Burroughs

B 6000 Series System Notes

MARK III.1 RELEASE

PRICED ITEM

Printed in U.S. America

September 1979

5011257

Burroughs believes that the information described in this manual is accurate and reliable, and much care has been taken in its preparation. However, no responsibility, financial or otherwise, is accepted for any consequences arising out of the use of this material. The information contained herein is subject to change. Revisions may be issued to advise of such changes and/or additions.

Correspondence regarding this document should be addressed directly to Burroughs Corporation, P.O. Box 4040, El Monte, California 91734, Attn: Publications Department, TIO—West.

MARK 3.1

TABLE OF CONTENTS

MARK 3.1

DOCUMENT CHANGES NOTES (D NOTES)

GENERAL

D2465 GENERAL - INSTALLATION OF THE "III.1" SOFTWARE SYSTEM

This note describes procedures for implementing the Mark III.1 software system. It mentions certain operational procedures, and references some system notes describing major III.1 features. Other major III.1 features are described in the appendices to the D and P Notes. These system notes should be studied before implementation of III.1 is attempted.

The major change for III.1 for most installations is the conversion of the MCP from the ESPOL language to a new ALGOL-based programming language named NEWP. Salient features of the NEWP language and information concerning the conversion of the MCP to NEWP may be found in the following notes: NEWP note D2804, "Implementation of NEWP" and MCP note P2288, "MCP Conversion to NEWP". In addition to the conversion to NEWP, the MCP has been reorganized into modules which group together logically related features, which is described in MCP note D2430, "MCP Restructuring". This reorganization has also required a complete resequencing,

Since NEWP's internal information table structures are different from those of the other ALGOL-based languages, it is no longer possible to LOADINFO from the MCP. The major effect of this restriction is on the system intrinsics. Many of the intrinsics have been converted to other languages (e.g. ESPOL to DCALGOL) or have been absorbed into the MCP symbolic (e.g. SORT). Any local modifications to the system intrinsics must therefore be checked for the proper languge constructs and intrinsic symbolic location.

A sample WFL job for compiling all system software items may be found in the System Software Operational Guide, Volume II (Form No. 5001688), Section 14. The changes for III.1 software will be described in a revision to this manual.

The III.1 MCP, WFL, Controller, ALGOL, DCALGOL and NEWP are all compilable on the Mark III.0 level software. This allows compilation of the III.1 software on III.0 as the first phase of the bootstrapping process. Note that the MCP on the SYSTEM tape is usable "as is" if no modifications are required by the installation. If such modifications are required, they may be made and the resulting symbolic compiled on the III.0 system.

When the installation is ready to move to III.1, the CM command may be used to change to the new MCP. It is recommended that the mix be empty, as a loss of information about running jobs may occur if the "??CM" command is used with a non-empty mix. Following the CM, the III.1 intrinsics should be loaded, and then any system software necessary to continue operation may be copied from the system tape.

A number of new features have been added to the system. Careful review of the system notes will allow the installation to decide which features might be relevant to its operation, and will provide information as to the use of those features.

Those installations planning to move system files off the halt-load pack (or disk) by use of the DISKLOCATION (DL) command should be aware that the III.0 software is not sensitive to this command. If operational system files such as the job description, SUMLOG and BD files are moved from their normal locations to some other family, a loss of information could result if it is necessary to CM back to III.0. For this reason it may be better not to use the DL feature until the installation is satisfied with the stabilty of III.1 and has decided that a CM to III.0 is unlikely.

Customers using the early ("TC") versions of the MCP to support * GLOBAL tm Memory systems should note the easing of several restrictions which were imposed by that MCP. In particular, the requirement that all exchangeable devices be exchanged to all processors has been removed. The restriction of SWAPPER to one local memory structure has also been removed.

* "GLOBAL Memory" is a trademark of Burroughs Corporation.

In order to improve operator communication with the MCP, many of the ODT input messages will be revised in future software releases. The revised messages have been incorporated in the III.1 release without deleting the old message. This enables operators to make an orderly transition to the new syntax. It is recommended that installations study GENERAL note D2535, "Revised ODT Messages", and begin using the new forms. In addition to the proposed revisions, new messages are documented in this note. desired.

SWAPPER has been modified to provide more flexibility in the program structures which are supported. It may also be used in multiple local memory subsystems on B6800 multiprocessor systems. In addition to the flexibility of this implementation, a load-balancing algorithm has been incorporated to improve performance on B6800 multiprocessors. See GENERAL note D2906, "Changes in the SWAPPER Mechanism", for details of the new facilities.

As was announced at Spring '79 CUBE, a new and more comprehensive printer control facility is planned for a future release. In conjunction with this program, the MCP option PRINTERLABELS has been removed along with the pertinent MCP code.

D2535 GENERAL - REVISED "ODT" MESSAGES

This system note documents changes to the ODT (Operator Display Terminai) messages.

Primarily, the changes made now allow the standard mix-related messages to appear only in the following form:

    <mix number list> ODT MESSAGE.

Previously, ODT MESSAGE <mix number list> has been also allowed.

The first release of the B6800 Multiprocessor system and the III.1 system release will allow either form; the III.3 system release will allow only the syntax described herein.

The following list shows the old and new mnemonics where applicable:

Old	New	
AP	AB	Auto Backup
BRK	CQ	Clear Queued Messages
CI	SI	Display System Intrinsics
DC	ID	Initialize Datacom
DP	DUMP	Dump Memory
EI	HS	Prevent Schedule Jobs Entering Mix
EQ	MQ	Make or Modify Queue
IV	RC	Reconfigure
LR	TL	Transfer Log
M	MX	Display Mix Entries
MIXL	ML	Mixlimit
P	PER	Display Peripheral Status
PC	SC	Display System Configuration
PU	MU	Make User
RESTORE	SUPPRESS-	Suppresses Display of Active Job
RO	OP	Reset Options
RR	SR	Remove Reader Security Restrictions
SO	OP	Set Options
TD	TDIR	Tape Directory
TO	OP	Options
UA	UR	Unit Reserved,Available
WD	TD	Display Date
WI	SI	Display System Intrinsics
WT	TD	Display Time
XS	FS	Force From Schedule Queue

The following messages have been added or modified for the the first release of the B6800 Multiprocessor system:

A (Active Mix Entries)
ACQUIRE
CF (Configuration File)
CU (Core Usage)
DL (Disk Location)
FREE
GC (Group Configuration)
HN (Hostname)
HU (Host Usercode)
J (Job and Task Structure)
LP (Lock Program)
ML (Mixlimit)
MS (Make Subsystem)
MX (Mix Entries)
NET (Network)
PG (Purge)
PP (Privileged Program)
RECONFIGURE
RY (Ready)
S (Scheduled Mix Entries)
SN (Serial Number)
SV (Save)
SW (Swapper)
W (Waiting Mix Entries)
WM (What MCP)

See GENERAL note D2861 for a description of "B6800 Multiprocessor Systems"; see GENERAL note D2867 for a description of "Soft Configuration of Global Memory Systems".

A (Active Mix Entries) Message

Revised Syntax:

```
-- A -------------------------------------------------------------->
      |-/1\- ALL -|

>------------------------------------------------------------------|
  |                                               |
  | |<------------------------------------------| |
  | |                                          |  |
  |-----/1\- SWAPPER -----------------------------|
      |      --                               |
      |-/1\- MCSNAME ---------<mcsname>-------|
      |      ---      |     |                 |
      |               |- = -|                 |
      |                                       |
      |-/1\- IN ---<subsystem id>-------------|
                 |                            |
                 |- ( <processor id list> ) - |
```

```
<processor id list>

  |<-------------|
  |   |      |   |
  |   |<- , -|   |
  |              |
------ GLOBAL ------|
    |  -       |
    |- 1 ------|
    |          |
    |- 2 ------|
    |          |
    |- 3 ------|
    |          |
    |- 4 ------|
```

Revised Semantics:

The A (Active Mix Entries) message lists all active jobs and tasks. If ALL is used, any active jobs or tasks which have been suppressed by the SUPPRESS message are displayed in addition to unsuppressed tasks.

When SWAPPER is specified, only jobs running in swapspace will be displayed.

When MCSNAME is specified, only jobs that originated from the specified MCS will be displayed.

On a B6800 multiprocessor system, each displayed task is preceded by a subsystem indicator: processor id for a local memory task, "G" for a global memory task, or blank for a task whose subsystem location is currently unassigned.

When IN is specified, only jobs with stacks running in the subsystem identified or the processor identified will be displayed.

A typical response to the A message is as follows:

```
 -----7 ACTIVE ENTRIES-----

 0232 JOB 80DCP/0
 0234 JOB 70SYSTEM/CANDE
 0243/0244 55 ESPOL JKL
 0254/0254 55#SEPCOMP JHOST2/HOST
*0255x0256 55 ALGOL KAP
```

The lower case "x" between the job and task number in the last entry of the example indicates that the job is under control of an MCS. Swap jobs are flagged with an "#" between the priority number and the file name. The number in the active entry heading (7) is the total number of active entries including suppressed entries.

New Example:

```
A ALL MCSNAME=SYSTEM/CANDE
--------------------------
```

This message will display all active jobs (including suppressed) that originated from SYSTEM/CANDE.

New Example: (B6800 Multi-Processor System):

```
A SW IN(3)
----------
```

This message will display all active jobs (excluding suppressed) whose stacks are running in swapspace in processor 3.

AB (Auto Backup) Message

AB replaces the AP (Auto Backup) Message.

Syntax:

```
-- AB ---------------------------------|
       |                            |
       |-<number>-------------------|
       |           |                |
       |           |- CP -----------|
       |           |                |
       |           |- LP -----------|
       |                            |
       |------------ CP ---<number>-|
       |        | |      |
       |-  -   -| |- LP -|
```

Semantics:

The AB (Auto Backup) message sets the maximum number of ABed line printers and/or card punches. This number is set to zero at Halt/Load time if the MCP option AUTORECOVERY is reset. When AUTOBACKUP is looking for a line printer or card punch to use for automatic output of backup files, it begins looking for on-line ABed units. If AUTOBACKUP cannot find any ABed units, it tries any on-line line printer or card punch.

This message may appear simply as AB, in which case the system responds by displaying the current number of line printers and card punches available to AUTOBACKUP.

Example:

```
AB
--

   AB MAX=2; CP MAX=1; AB-ED LPS=0; AB-ED CPS=0
```

When a number immediately follows the AB, that number is used as the maximum number of copies of AUTOPRINT allowed.

Example:

```
AB 3
----

   AB MAX=3; CP MAX=1; AB-ED LPS=0; AB-ED CPS=0
```

When a number and device immediately follow the AB, that number is used as the maximum number of copies of AUTOPRINT allowed on that device.

Example:

```
AB 3 LP
-------

   AB MAX=3; CP MAX=0; AB-ED LPS=1; AB-ED CPS=0
```

When a device and unit number immediately follow the AB, the indicated output device is ABed.

Example:

```
AB LP 12
--------
```

The number of devices assigned may exceed the number of allowed copies of AUTOBACKUP. The following sequence of AB messages allows one AUTOBACKUP printing on LP 5.

Example:

```
AB 0
----

AB LP 5
-------
```

AUTOBACKUP may be disabled on the indicated device by preceding the specified device with a hyphen. When such a message references a device upon which output is currently being generated, this activity is allowed to proceed to a normal termination before the device is disabled.

Example:

```
   AB-CP 13
   --------
```

When use of AB results in all line printers and card punches being freed of the ABed status and AB is set to 0, the queue of disk/diskpack backup yet to be printed is forgotten. This does not affect the actual backup files, except they are not printed automatically by AUTOBACKUP. If subsequently any LP is given AB status or AB is set to a non-zero number, all disk and native mode diskpacks will be searched for backup files which are enqueued for printing. A similar situation holds for card punch.

Backup files introduced by Library Maintenance (i.e., COPY, COPY&COMPARE and ADD) from tape are picked up by this rebuilding of the queues and will be printed in turn.

Example:

```
   COPY BD/0000365/0000366/000TASKFILE
   COPY BP/0000367/0000368/000TASKFILE

      Backup files of this form will be picked up.

   COPY BD/0000365/0000366/000TASKFILE AS JEV/TASKFILE

      Backup files of this form will not be picked up.
```

ACQUIRE Message

Syntax:

```
-- ACQUIRE --- MOD -----<mod range>----------|
            |        |                       |
            |- MODS -|                       |
            |                                |
            |- PROC --<node id>--------------|
            |                                |
            |-<unit type>--<unit number>-----|
```

Semantics:

Existing groups can be altered by the FREE and ACQUIRE messages. The ACQUIRE message allows an active group to acquire additional resources. Devices "acquired" will be displayed as saved units. Disk packs will also be closed. The "new" devices must be RYed before they can be used.

Example:

```
   PER MT
   ------

      ----- MT STATUS -----
      81*P  [HALL  ] 1600 #1 1:0 MEMORY/DUMP [2,1]
      82  NOT AVAILABLE TO GROUP

   ACQUIRE MT 82
   -------------

      MT82 WILL BE ACQUIRED

   RY MT 82
   --------

   PER MT
   ------

      ----- MT STATUS -----
      81*P  [HALL  ] 1600 #1  1:0 MEMORY/DUMP [2,1]
      82*P  [MIKE  ] 1600     S C R A T C H [2,1]
```

AD (Access Duplicate) Message

Revised Syntax:

```
-- AD ----------------<familyname>-- ( --<familyindex>-- ) --|
        |             |
        |- - ---------|
              |       |
              |- ON -|
```

Semantics:

The AD (Access Duplicate) message causes the access structure (SYSTEM/ACCESS or SYSTEM/CATALOG) to be duplicated. The duplicate structure has the family index number appended; e.g., SYSTEM/ACCESS/002 or SYSTEM/CATALOG/003. The family index represents a member of the Catalog or Access family.

Example:

```
AD 2
----

   *608 JOB 80 COPYDIR
```

AD- causes the referenced access structure to no longer be considered a duplicate. The actual file is not removed.

Example:

```
AD- 2
-----

   SYSTEM/ACCESS/002 REMOVED
```

ADM (Automatic Display Mode) Message

ADM has been revised, as follows:

MIX has been changed to MX.

Revisions made to the following messages have been applied to ADM also:

```
A  (Active Mix Entries)
J  (Job and Task Structure Display)
ML (Mixlimit)
MX (Mix Entries)
S  (Scheduled Mix Entries)
W  (Waiting Mix Entries)
```

See the description of the individual messages for details.

Syntax:

Unchanged

Semantics:

Unchanged

Example:

Unchanged

AP (Auto Backup)

AP has been eliminated; see AB (Auto Backup) for new message.

AX (Accept) Message

Revised Syntax:

```
--<mix number list>-- AX --<text>--|
```

Semantics:

Unchanged

Example:

Unchanged

BRK (Break) Message

BRK has been eliminated; see CQ (Clear Queue) for new message.

CF (Configuration File)

Syntax:

```
-- CF -------------------|
          |              |
          |- + ---------|
          |              |
          |- - ---------|
          |              |
          |-<file name>-|
```

Semantics:

The CF message is used to designate or display the title of the current configuration file.

CF with no options displays the title of the current configuration file.

CF+ selects SYSTEM/CONFIGURATION, the default, as the configuration file.

CF- removes the designation of the configuration file.

CF <file name> selects the named file as the configuration file.

Example:

```
CF
--

   CONFIGURATION FILE: (EVANS)SYSTEM/GMMCONFIG/022079

CF +
----

   SYSTEM/CONFIGURATION IS THE CONFIG. FILE

CF XYZ
------

   XYZ IS THE CONFIG. FILE
```

CI (Change Intrinsics) Message

CI has been eliminated; see SI (System Intrinsics) for new message.

CQ (Clear Queue) Message

Syntax:

```
-- CQ --|
```

Semantics:

The CQ (Clear Queue) message clears system messages queued for the terminal.

A typical response to the CQ message is the following:

```
MESSAGES FLUSHED
```

CS (Change Supervisor) Message

CS incorporates the WS (What Supervisor) message.

Revised Syntax:

```
-- CS ---------------------------------------|
        |                                |
        |-  -  --------------------------|
        |                                |
        |-<file title>-- ON --<familyname>-|
```

Semantics:

The CS (Change Supervisor) message displays, cancels or designates as a "supervisor" program a code disk file specified by a file title, as follows:

```
CS                 Displays the current supervisor
CS-                Cancels the current supervisor
CS <file title>    Designates the supervisor
```

The file must be present when CS is executed.

At Halt/Load time, the supervisor program is automatically entered into the mix. If the run-time system option DUPSUPERVISOR is set, the MCP will attempt to execute a code file <file title>/FMLYINX<nnn>, where <nnn> is the family index. If DUPSUPERVISOR is reset, the MCP will attempt to execute the designated supervisor program.

Example:

```
CS
--

   SUPERVISOR: SYSTEM/SUPERVISOR

CS-
---

   SUPERVISOR: NOT SPECIFIED

CS SUPPRESS/MIX/NOS
-----

   SUPERVISOR: SUPRESS/MIX/NOS
```

CU (Core Usage) Message

Revised Syntax:

```
------------------------ CU --|
  |                     |
  |-<mix number list>-|
```

Semantics:

Unchanged

New Example:

The resulting display for a B6800 multiprocessor system will depend upon the number of local processors comprising the system. A 2-processor system could produce the following:

```
CORE USAGE AT    10:21:04
GLOBAL MEMORY                LOCAL MEMORY 1              LOCAL MEMORY 2
  AVAILABLE    255028          AVAILABLE    111967         AVAILABLE     31843
  NON SAVE     105882          NON SAVE     301632         NON SAVE     357335
  SAVE          32306          SAVE          98401         SAVE         122822
   TOTAL       393216           TOTAL       512000          TOTAL       512000

SYSTEM TOTAL
  AVAILABLE    398838
  NON SAVE     764849
  SAVE         253529
   TOTAL      1417216
```

As the number of processors increase, the number of local memory statistic blocks will increase.

For a single B6800 with global memory, the display will be of just the local memory **with** one additional line giving the total of the shared (global) memory available.

DC (Datacom Initiation) Message

DC has been eliminated; see ID (Initialize Datacom) for new messages.

DD (Directory Duplicate) Message

Revised Syntax:

```
-- DD ------------------<familyname>-- ( --<familyindex>-- ) --|
      |     | |      |
      |- - -| |- ON -|
```

Semantics:

Unchanged

New Example:

```
DD TIOMASTER 2
--------------

    -----ACTIVE ENTRIES-----
    .
    .
    .
    *9303 JOB 80 COPYDIR
```

DL (Disklocation) Message

Syntax:

```
---- DL ---------------------------------------------------------->
   |               |
   |- DISKLOCATION -|

>-----------------------------------------------------------------|
  |                                                               |
  |- BACKUP ------------------------------------------------------|
  |                                     | |                       |
  |- USERDATA --------------------------| |- ON --<familyname>-|
  |                                     |
  |- OVERLAY -- ( <processor id list> ) -|
  |                                     |
  |- JOBS ------------------------------|
  |                                     |
  |- CATALOG ---------------------------|
  |                                     |
  |- LOG --<familyname>-----------------|
```

<processor id list>

```
   |<-------------|
   |    |     |   |
   |    |<- . -|   |
   |             |
------ GLOBAL ------|
     |   -     |
     |- 1 ------|
     |         |
     |- 2 ------|
     |         |
     |- 3 ------|
     |         |
     |- 4 ------|
```

Semantics:

The DL (Disklocation) message specifies that system files reside on families other than the Halt/Load family.

The simple form of "DL" is used to interrogate the current settings of the various "DL" families.

A typical response to the DL message is as follows:

```
DISK LOCATION:
  CATALOG ON DISK
  JOBS ON DISK
  USERDATA ON DISK
  BACKUP ON BKPACK
  LOG ON DISK
  OVERLAY ON DISK
```

The DL OVERLAY specification allows the user to direct system OVERLAY file allocation (and the consequent overlaying I/O activity) to any desired family. For a B6800 Multiprocessor System, each processor has its own DL OVERLAY specification to ensure direct I/O access to the memory being overlayed. Different local processors may not be able to access directly the same disk-type devices.

The <processor id list> specification for an OVERLAY family is meaningful only for a B6800 Multiprocessor System, and is rejected on ODT inputs to a monolithic system.

The presence of each pertinent DL OVERLAY family is checked at system initialization time, and whenever an ODT input attempts to change a specification. (This "presence" check includes a test for proper I/O visibility on a B6800 Multiprocessor System.) If a family is not present, an RSVP wait allows the user the following options:

1. Mount the family baseunit and "OK" the RSVP wait.

2. "OF" or "DS" the RSVP wait to disassociate DL OVERLAY from the missing family. (At initialization time, the unsatisfied DL OVERLAY specification defaults to the <Halt/Load familyname>; on a B6800 Multiprocessor System, the default for a local processor is the <Halt/Load familyname> for that processor. If the wait arose from an ODT input, the previous family specification for the DL OVERLAY case in question is retained.)

Whenever a DL OVERLAY specification is changed, the MCP overlay file for the system (monolithic) or processor (multiprocessor) is immediately moved during the new family. This file is also moved at system initialization if a DL OVERLAY specification is other than the corresponding <Halt/Load familyname>; other overlay files, however, once allocated on a given family, are never moved. Thus, changing a DL OVERLAY family while the system is running affects only the allocation of new overlay files.

Example:

```
DL OVERLAY MYPACK
-- ------- ------

    <job#> OVERLAY SET TO MYPACK
```

The DL BACKUP specification, and a corresonding extension to the "SB" ODT message, allow a site to direct its PRINTER and PUNCH BACKUP files to any desired family. The "SB" extension recognizes "DLBACKUP" as a valid substitute backup specification.

A new task attribute, "BACKUPFAMILY", has been implemented; it may be set only by the MCP or an MCS. Specifying DLBACKUP as a substitute backup medium causes the BACKUPFAMILY in the TASK (if valid, otherwise the current DL BACKUP family specification) to be the family on which printer and punch backup files are opened. The DL BACKUP familyname is stored into the BACKUPFAMILY of the job stack when the job is initiated; task attribute inheritance propagates this BACKUPFAMILY to all stacks initiated under the job. CANDE stores the DL BACKUP familyname at log-on time, and propagates it to all tasks begun under the session. Thus, all backup files for a job/session normally go to a single family, even if the DL BACKUP family is changed during a job or CANDE session. (This, of course, need not be true if SB specifications were changed during the job/session, and is not true for backup files explicitly directed elsewhere.)

The presence of the specified DL BACKUP family is checked at system initialization time, and whenever an attempt is made to change it via an ODT input. If the family is not present, an RSVP wait allows the user the following options:

1. Mount the family baseunit and "OK" the RSVP wait.

2. "OF" the RSVP wait to keep the missing family as the DL BACKUP specification.

3. "DS" the RSVP wait to disassociate DL BACKUP from the missing family. (At initialization time the DL BACKUP specification defaults to the <Halt/Load familyname>; DSing a wait resulting from an ODT input causes the previous DL BACKUP family specification to be retained.)

The CANDE commands BACK and BDREMOVE automatically find files on DISK, PACK and the session's BACKUPFAMILY.

Example:

```
DL BACKUP MYPACK
-- ------ ------

    <job#> DL BACKUP FAMILY SET TO:  MYPACK
```

The DL JOBS specification allows the operator to specify the family on which JOBDESC is located. No JOBDESC file is actually be moved to that family, but all Halt/Loads subsequent to the specification will search for the family. If it is not present, the operator may IL the search to another family (this updates DL to the ILed family) or OF the search (this causes JOBS DL to default to the Halt/Load family).

Example:

```
DL JOBS ON MYPACK

    DISKLOCATION FOR JOBS WILL BE CHANGED
```

The DL CATALOG specifications allows the operator to specify the catalog family. It works in the same fashion as JOBS.

Example:

```
DL CATALOG ON MYPACK

    DISKLOCATION FOR CATALOG WILL BE CHANGED
```

The DL LOG specification allows a site to have its SYSTEM/SUMLOG file maintained on any desired family. Efficiency considerations suggest that the DL LOG family chosen for a B6800 Multiprocessor System be directly visible to each local processor, but this need not be the case. SYSTEM/LOGANALYZER has been changed to search the DL LOG family for any log file whose location has not been explicitly specified.

During system initialization, the specified DL LOG family must be present or the DL LOG specification defaults to the <Halt/Load familyname> as a non-IAD file.

When an attempt is made to change the DL LOG specification via an ODT input, an RSVP wait occurs if the newly specified family is not present and the user has the following options:

1. Make the family baseunit available and "OK" the RSVP wait.

2. "OF" or "DS" the RSVP wait to discard the requested change and maintain the previous DL LOG specification.

Assuming any required family is present, MCP action when changing the DL LOG specification is as follows:

1. A Log Release ("LR") is done on the SYSTEM/SUMLOG on the old DL LOG family.

2. If a valid SYSTEM/SUMLOG file already exists on the new DL LOG family, it is continued; otherwise, a new SYSTEM/SUMLOG is opened.

Example:

```
DL LOG MYPACK
-- --- ------

    <job#> SUMLOG IS ON MYPACK
```

The DL USERDATA specification, along with a corresponding change to SYSTEM/MAKEUSER, allows a site to maintain its SYSTEM/USERDATAFILE on any desired family.

An attempt to change the DL USERDATA family via an ODT input succeeds provided there was no SYSTEM/USERDATAFILE on the old DL USERDATA family (or the old family was not present). If a SYSTEM/USERDATAFILE does exist, MCP action is as follows:

1. The presence of the new DL USERDATA family is verified; if it is not present, the user may "OF" or "DS" the resulting RSVP wait to abort the change and retain the previous DL USERDATA specification.

2. A check is made to determine whether a SYSTEM/USERDATAFILE already exists on the new family; if it does, an RSVP wait allows the user the following options:

   a. "DS" the RSVP wait to abort the DL USERDATA change, or

   b. "RM" the RSVP wait to remove the file, or

   c. dispose of the file in some other way, and "OK" the wait.

3. The current SYSTEM/USERDATAFILE is copied to the new USERDATA family; the SYSTEM/USERDATAFILE on the old family is renamed "USERDATAFILE/<date-time>". (Should

the copy fail, the DL USERDATA change is aborted.)

If there is no SYSTEM/USERDATAFILE on the USERDATA family, or that family is not present, and a non-interactive request is made for userdata information, MCP action is as follows:

1. If there is no family, a "REQUIRES PK" RSVP message is displayed, with OK, DS or OF options. OK causes repeated demand for the family; DS terminates the requestor; OF causes the code to proceed without the family.

2. If the family is present but has no SYSTEM/USERDATAFILE, or if "REQUIRES PK" message was answered "OF", a "NO FILE" RSVP message is displayed. If the "IL" option is used, the USERDATA DL family is set to the familyname of the resulting unit.

Example:

```
DL USERDATA MYPACK
-- -------- ------

   SYSTEM/USERDATAFILE WAS MOVED TO MYPACK
```

DUMP (Dump Memory) Message

DUMP replaces the DP (Dump Memory) Message.

Revised Syntax:

```
---- DUMP ----------------------------------------|
  |         |                                   |
  |         |-<nonnumeric text>-----------------|
  |         |                                   |
  |         |- " <text> " ----------------------|
  |                                             |
  |-<mix number list>-- DUMP -------------------|
                              |                 |
                              |-<option list>-|
```

<option list>

```
--------------------------------|
  |                           |
  | |<---------- , ---------| |
  | |                       | |
  |----- * -------------------|
      |                  |
      |- ALL ------------|
      |                  |
      |- NONE -----------|
      |                  |
      |- FAULT ----------|
      |                  |
      |- DSED -----------|
      |                  |
      |- BASE -----------|
      |                  |
      |- ARRAYS ---------|
      |                  |
      |- CODE -----------|
      |                  |
      |- FILES ----------|
      |                  |
      |- DBS ------------|
      |                  |
      |- PORTS ----------|
      |                  |
      |- LIBRARIES ------|
      |                  |
      |- PRIVATELIBRARIES -|
      |                  |
      |- SIBS -----------|
```

Semantics:

The DUMP (Dump Memory) message may be used to dump the entire contents of memory to tape or invoke a program dump on a particular program.

When text appears following the DUMP, that string appears in the memory dump tape as the reason for the dump.

Options may be used on a program dump as follows:

* The options appearing in the list will be ORed with any compiled-in options and the net result used to control the program dump.

ALL Dump all items in the stack and also list code segments.

NONE Use the default dump options.

FAULT A program dump will occur if a fault occurred in the program (divide by zero, segment array, etc.). A program dump is not invoked at the time the message is entered.

DSED A program dump will occur if the program is DSed. A program dump is not invoked at the time the message is entered.

BASE Dump the base of the stack which is used by the operating system.

ARRAYS Dump all present arrays.

CODE Dump all code segments.

FILES Dump all areas used by files.

DBS Dump SIB and data base stack.

PORTS Dumps ports and signals.

LIBRARIES Dumps library stack.

PRIVATELIBRARIES
Dumps only Private Library stacks.

SIBS Dumps only SIB.

Example:

```
3132 DUMP ALL
-----------
```

DP (Dump) Message

DP has been eliminated; see DUMP (Dump Memory) for new message.

DR (Date Reset) Message

Revised Syntax:

```
-- DR --<mm>-- / --<dd>-- / --<yy>--|
```

Revised Semantics:

The DR (Date Reset) message may be used to change the date currently in use by the MCP. The desired date is specified by <mm>, <dd>, and <yy>, which are one- to two-digit numeric indications of the month, date, and year, respectively.

The delimiter must be a slash ("/").

New Example:

```
DR 04/22/75
-- --------

   DATE IS TUESDAY APR 22, 1975 (75112)
```

(75112) indicates that the year is 1975 and the date is day 112.

DS (Discontinue) Message

Revised Syntax:

```
--<mix number list>-- DS -------------------|
                         |                  |
                         |-<option list>-|
<option list>

--------------------------------|
   |                          |
   |  <---------- , ---------| |
   |                         | |
   |----- ALL ----------------|
         |                 |
         |- NONE ----------|
         |                 |
         |- FAULT ---------|
         |                 |
         |- DSED ----------|
         |                 |
         |- BASE ----------|
         |                 |
         |- ARRAYS --------|
         |                 |
         |- CODE ----------|
         |                 |
         |- FILES ---------|
         |                 |
         |- DBS -----------|
         |                 |
         |- PORTS ---------|
         |                 |
         |- LIBRARIES -----|
         |                 |
         |- PRIVATELIBRARIES -|
         |                 |
         |- SIBS ----------|
```

Revised Semantics:

With the DS (Discontinue) message, all programs associated with the given mix numbers are terminated or removed from the mix schedule or scheduling queue.

If the program has been initiated and options appear in the DS message, these options will be used to control a program dump taken at the time of termination.

Options for the program dump are as follows:

ALL Dump all items in the stack and also list code segments.

NONE Use the default dump options.

FAULT A program dump will occur only if a fault occurred in the program (divide by zero, segment array, etc.). A program dump is not invoked at the time the message is entered.

DSED A program dump will occur only if the program is DSed. A program dump is not invoked at the time the message is entered.

BASE Dump the base of the stack which is used by the operating system.

ARRAYS Dump all present arrays.

CODE Dump all code segments.

FILES Dump all areas used by files.

DBS Dump SIB and data base stack.

PORTS Dumps ports and signals.

LIBRARIES
Dumps library stack.

PRIVATELIBRARIES
Dumps only Private Library stacks.

SIBS Dumps only SIB.

Example:

Unchanged

EI (Emergency Interrupt) Message

EI has been eliminated; see HS (Hold Schedule) for new message.

EQ (Eliminate Queue) Message

EQ has been eliminated; see MQ (Make or Modify Queue) for new message.

FA (File Attribute) Message

Revised Syntax:

```
                            |<------------- , ------------|
                            |                             |
--<mix number list>-- FA ---<file attribute assignment>----|
```

Semantics:

Unchanged

New Example:

```
7988 FA VERSION=12, CYCLE=13
----------------------------
```

FM (Form Message) Message

Revised Syntax:

```
--<mix number list>-- FM --<device>--<unit number>--|
```

Semantics:

Unchanged

Example:

Unchanged

FR (Final Reel) Message

Revised Syntax:

```
--<mix number list>-- FR --|
```

Semantics:

Unchanged

Example:

```
4423 FR
-------
```

FREE Message

Syntax:

```
-- FREE --- MOD ----<mod range>----------|
          |          |                  |
          |- MODS -|                    |
          |                             |
          |- PROC --<node id>-----------|
          |                             |
          |-<unit type>--<unit number>-|
```

Semantics:

Existing groups can be altered by the FREE and ACQUIRE messages. The FREE message is used to detach resources from an active group. Devices must be SAVEd before they can be "freed". Disk packs must also be closed first.

Example:

```
   PER MT
   ------

      ----- MT STATUS -----
      81*P  [HALL  ] 1600 #1 1:0 MEMORY/DUMP [2,1]
      82*P  [MIKE  ] 1600    S C R A T C H [2,1]

   SV MT 82
   --------

   FREE MT 82
   ----------

      UNIT IN USE

   FREE MT 82
   ----------

      MT82 FREED

   PER MT
   ------

      ----- MT STATUS -----
      81*P  [HALL  ] 1600 #1 1:0 MEMORY/DUMP [2,1]
      82  NOT AVAILABLE TO GROUP
```

FS (Force from Schedule) Message

FS replaces the XS (Exceed Schedule) message.

Syntax:

```
--<mix number list>-- FS --|
```

Semantics:

When the FS (Force Schedule) message is entered, the execution of the indicated scheduled job is unconditionally initiated. Also, it may be used to force a job out of a scheduling queue, as well as one initiated but scheduled by the MCP. The system does not display a response to this message.

Example:

```
   7852 FS
   -------
```

GC (Group Configuration) Message

Syntax:

```
-- GC --|
```

Semantics:

The GC (Group Configuration) message displays the current group configuration.

Example:

```
   GC
   --
```

Response for Multiprocessor System:

```
     ***** GROUP CONFIGURATION *****
GROUP ID: BLUE
PROCESSOR PORTS: A000   PROC ID. = 1
PERIPHERALS ALLOWED TO GROUP:
  12-13,32-33,42-43,69-71,81-83,193-194
GLOBAL MEMORY STATUS:
  PRIVATE MEMORY AVAILABLE: NONE
  PRIVATE MEMORY IN USE:    NONE
  SHARED MEMORY AVAILABLE:  32-33,38-39,44-45,50-51,56-57,62-63
  SHARED MEMORY IN USE:     32-33,38-39,44-45,50-51
```

Response for Monolithic System:

```
     ***** GROUP CONFIGURATION *****
GROUP ID: DEFAULT
PERIPHERALS ALLOWED TO GROUP:
  1-255
GLOBAL MEMORY STATUS:
  PRIVATE MEMORY AVAILABLE: NONE
  PRIVATE MEMORY IN USE:    0-63
  SHARED MEMORY AVAILABLE:  NONE
  SHARED MEMORY IN USE:     NONE
```

HI (Exceptionevent) Message

Revised Syntax:

```
--<mix number list>-- HI ---------------|
                         |              |
                         |-<number>-|
```

Semantics:

Unchanged

Example:

```
4312 HI
-------
```

HN (Hostname) Message

Syntax:

```
-- HN -----------------|
      |               |
      |-<hostname>-|
```

Semantics:

The HN (Hostname) message displays the hostname of the system. HN <hostname> designates the hostname of the system. The new hostname does not take effect until the next Halt/Load.

Example:

```
HN
--

   HOSTNAME: BLUE

HN GAZORBAFLEX
--------------

   HOSTNAME: BLUE
   HOSTNAME WILL BE >> GAZORBAFLEX << AFTER NEXT HALT/LOAD
```

HS (Hold Schedule) Message

HS replaces the EI (Emergency Interrupt) message.

Syntax:

```
-- HS ------------|
        |        |
        |- - -   |
        |        |
        |- ? ---|
```

Semantics:
The HS (Hold Schedule) message stops the selection of jobs from the schedule into execution. HS- resumes job selection. HS? displays the current status of job selection.

Example:

```
   HS
   --

      JOB SELECTION STOPPED
```

and to resume normal operations

```
   HS-
   ---

      JOB SELECTION RESUMED

   HS?
   ---

      HOLD SCHEDULE IS RESET
```

HU (Host Usercode) Message

Syntax:

```
-- HU -----------------|
        |             |
        |-<usercode>- |
        |             |
        |- - -------- |
```

Semantics:

The HU (Host Usercode) message designates the usercode to be used for inter-host system communications. Networking must not be in effect when HU is used.

HU is referenced in two ways:

1. The usercode attached to inter-host system communications (i.e., non-user communications).
2. Used for identity in inter-host communications where a usercode is not available; e.g., ODT input.

Example:

```
   HU <usercode> sets the "SYSTEM" usercode
   HU            displays the current usercode
   HU-           removes the usercode.
```

A new "LOCATOR" for the USERDATA interface is used to determine whether the usercode is a "SYSTEM" usercode or not. This "LOCATOR" is called "SYSTEMUSER". This same mnemonic is used by MAKEUSER to designate a usercode as a "SYSTEM" usercode.

If no system usercode is set by the HU command and inter-host system communication is attempted, an error condition is generated.

If an inter-host system communication is received and the usercode associated with the message is not recognized by the receiving host as a "SYSTEM" usercode in the USERDATAFILE, an error condition is generated.

ID (Initialize Datacom) Message

ID replaces the DC (Datacom Initiation) message.

Syntax:

```
-- ID -----------------------------------------------------|
         |                |
         |-<number>-|  |-<identifier>----------------------|
         |                                |                 |
         |                                |- ON --<familyname>-|
         |                                                  |
         |-  -  ------------------------------------------|
```

Semantics:

The ID (Initialize Datacom) message is provided to allow initialization of data communications processors and to yield information regarding the DCP files.

When a number is used in the message, the DCP identified by that number is initialized. When an identifier appears in the message, that identifier is used as the prefix of the NIF and DCPCODE files to be employed by the DCP.

Example:

```
ID 6 DATACOM2
-------------
```

The DCP numbered 6 will be initialized; it will employ the files titled DATACOM2/NIF and DATACOM2/DCPCODE.

```
ID 6
----
```

The DCP numbered 6 will be initialized; by default, it will employ the NIF and DCPCODE files with the present prefix.

```
ID-
---
```

The DCP prefix SYSTEM is made.

```
ID
--
```

The system responds by indicating the current NIF and DCPCODE file prefix, as follows:

```
NIF:TIONDL
```

IL (Ignore Label) Message

Revised Syntax:

```
--<mix number list>-- IL --<device>--<unit number>--|
```

Semantics:

Unchanged

Example:

```
7924 IL MT113
-------------
```

IV (Initialize and Verify) Message

IV has been eliminated; see RC (Reconfigure) for new message.

J (Job and Task Structure Display) Message

Revised Syntax:

```
-- J ------------------------------------------------------------>
      |          |
      |-/1\- ALL -|

>-----------------------------------------------------------------|
  |                                              |
  | |<-----------------------------------------| |
  | |                                          | |
  |-----/1\- SWAPPER ----------------------------|
      |      --                              |
      |-/1\- MCSNAME ---------<mcsname>------|
      |      ---      |     |                |
      |               |- = -|                |
      |                                      |
      |-/1\- IN ---<subsystem id>------------|
                 |                           |
                 |- ( <processor id list> ) -|
```

```
<processor id list>

  |<-------------|
  |   |      |   |
  |   |<- , -|   |
  |              |
------ GLOBAL ------|
    |  -       |
    |- 1 ------|
    |          |
    |- 2 ------|
    |          |
    |- 3 ------|
    |          |
    |- 4 ------|
```

Revised Semantics:

The J (Job and Task Structure Display) message lists the tasks by job structure. If ALL is used, any active jobs or tasks which have been suppressed by the SUPPRESS message are displayed in addition to unsuppressed tasks. J does not show jobs in the job queue; the SQ (Show Queue) message serves that purpose.

When SWAPPER is specified, only jobs running in swapspace will be displayed.

When MCSNAME is specified, only jobs that originated from the specified MCS will be displayed.

On a B6800 multiprocessor system, each displayed task is preceded by a subsystem indicator: processor id for a local memory task, "G" for a global memory task, or blank for a task whose subsystem location is currently unassigned.

When IN is specified, only jobs with stacks running in the subsystem identified or the processor identified will be displayed.

See "Queue-Level Scheduling" and Figure 1-2 for a discussion of the queue-matching algorithm.

The mix picture for a job running three tasks in parallel might look like the following:

```
0230 JOB 55
 . . 0231 55 COBOL TASK/A
 . . 0233 55 COBOL TASK/B
S. . 0234 55 ALGOL TASK/C
```

Note that every task and job has a different mix number. The number 55 is the priority, equal in this case for all the tasks shown.

The different tasks in a job may be in different states. Any of them may be active, scheduled, completed or waiting for operator action. The left margin has a letter flagging any of the job's tasks which are not active, as follows: S for scheduled tasks, W for waiting tasks (RSVP required) and E for compiles which have a syntax error. Completed tasks are not shown within the job structure. An asterisk "*" preceding a job or task indicates the task is displayed for the first time in a particular category. Swap jobs are flagged with an "#" between the priority number and the file name.

A job placed in the reader may never show in the job structure for any of three reasons. First, the job may have finished before a mix display (MX) was requested. In this case, it appears in the completed table if it is one of the list of most recently completed jobs. If Autoprint (AP) is running, the job's wrapup sheet is printed. Second, the job may have been rejected without being run. This condition shows up as a syntax error printout if Autoprint is running. Finally, the job may be in a queue waiting to be initiated, in which case the SQ message will show it.

A typical response to a J message is shown in the example.

Example:

```
-----JOB STRUCTURE-----

* 3044 JOB 50 COPY & COMPARE SY
*W.. 3045 50 LIBRARY/MAINTENANCE
  3072 JOB 60 INTERIMFR26
 E.. 3041 50 SYSTEM/REL/ALGOL ON INTERIM26
*E.. 3060 50#ALGOL (JONES) CANDE/CODE60
 W.. 3080 50 ALGOL (GORD) SYSTEM/PL1
*S.. 3097 50 PL/I (GORD)CANDE/CODE1080
```

Swap jobs are flagged with an "#" between the priority number and the file name. The number in the active entry heading is the total number of active entires including suppressed entries.

New Example:

```
J ALL MCSNAME=SYSTEM/CANDE
--------------------------
```

This message will display all active jobs (including suppressed) that originated from SYSTEM/CANDE.

New Example: (B6800 Multi-Processor System):

```
J SW IN(3)
----------
```

This message will display all active jobs (excluding suppressed) whose stacks are running in swapspace in processor 3.

LJ (Log to Joblog) Message

Revised syntax:

```
--<mix number list>-- LJ --<text>--|
```

Semantics:

Unchanged

Example:

Unchanged

LP (Lock Program) Message

Syntax:

```
--<mix number list>-- LP ------------|
                          |     |
                          |- - -|
```

Semantics:

The LP (Lock Program) message is provided to disable certain ODT messages which may interfere with program execution. Once locked, the DS and QT messages will be invalid for the locked task.

LP- removes the LP restriction.

Example:

```
7094 LP
-------

   7094 PROGRAM LOCKED.

7094 LP -
---------

   7094 NOT LOCKED.
```

LR (Log Release) Message

LR has been eliminated; see TL (Transfer Log) for new message.

M (Mix Entries) Messge

M has been eliminated; see MX (Mix Entries) for new message.

MIXL (Mixlimit) Message

MIXL has been eliminated; see ML (Mixlimit) for new message.

ML (Mixlimit) Message

ML replaces the MIXL (Mixlimit) message.

Revised Syntax:

```
-- ML ---------<number>----|
       |       |           
       |- = -|          |
       |                |
       |-  -  ----------|
```

Semantics:

The ML (Mixlimit) message is used to set or interrogate the current mixlimit. The system responds to the interrogation ML by displaying queue class, active count, mixlimit and the number of jobs queued for every job queue. If a default queue has been set, the letter D appears in the left margin of the display for that queue. A typical response to a ML message is as follows:

```
   QUEUE      ACTIVE     LIMIT      QUEUED

  D   0           2         10          0
      3           0          2          0
      5           4          4          1
  TOTAL           7       NONE
```

ML <number> sets the mixlimit used for introducing new jobs into the system. This message establishes the maximum number of jobs which may be introduced regardless of the sum of all mixlimits set for all queues. Setting the mixlimit equal to zero will allow no jobs to be run. The limits set on each queue need not be changed.

Example:

```
ML 20

QUEUE      ACTIVE     LIMIT      QUEUED
           .
           .
           .
           TOTAL                 7          20
```

ML- removes the mixlimit.

Example:

```
ML-
---
QUEUE      ACTIVE     LIMIT      QUEUED
           .
           .
           .
           TOTAL                 7          NONE
```

MODE (In or Output Mode) Message

Syntax:

Unchanged

Revised Semantics:

The MODE message is used to control the input/output capability of the indicated device. The IN option (read only) prevents new files from being created, but it does not prevent writes to the given unit. Old files can be updated and/or removed even if the IN option is set. The IO and OUT options allow the device to revert to normal operation. The MODE message allows the operator to inform the MCP that the WRITE ENABLE switch on a given disk pack unit has been changed.

Example:

```
MODE PK 096 OUT
---------------

   PK096 MODE IS OUT
```

This allows files to be opened for output.

GETUSERDISK gives a "SECTORS REQUIRED" message if a request is made for space on a write locked out disk pack.

MQ (Make or Modify Queue) Message

In addition to the pre-III.1 MQ syntax, the EQ (Eliminate Queue) option has been incorporated.

Revised Syntax:

```
-- MQ ---<number>---------------------------|
       |         |                         |
       |         | |<-------- , --------| |
       |         | |                    | |
       |         |---<queue attributes>---|
       |                                  |
       |-  -  --<number>------------------|
```

The syntax and semantics for <queue attributes> are unchanged, except as follows:

A new attribute has been added to the list of queue attributes,

Syntax for SUBSYSTEM:

```
--/1\- SUBSYSTEM -- = --<subsystem id>--|
```

Semantics for SUBSYSTEM:

This queue attribute is used to specify a subset of the memory subsystems in which the job queue is to be located.

Semantics:

The entry of the MQ (Make or Modify Queue) message results in the modification or creation of a job queue identified by the number following the MQ.

MQ- <number> eliminates the specified queue from the system.

Example:

```
MQ 37 MIXL = 2
--------------

     QUEUE 37:
       MIXLIMIT = 2
       DEFAULTS:
         NONE
       LIMITS:
         NONE
```

MS (Make Subsystem) Message

Syntax:

```
-- MS ---------------------------------------------------------|
        |                                                    |
        |---------<subsystem id>-----------------------------|
        |                                                    |
        |- - -|                                              |
        |                                                    |
        |-<subsystem id>--------- ( --<processor id list>-- ) -|
                       |         |
                       |- = -|
```

```
<processor id list>

  |<-------------|
  |    |    |    |
  |    |<- , -|  |
  |              |
------ GLOBAL ------|
     |  -        |
     |- 1 ------|
     |          |
     |- 2 ------|
     |          |
     |- 3 ------|
     |          |
     |- 4 ------|
```

Semantics:

The MAKE SUBSYSTEM message defines a logical subsystem within a system.

MS without qualification displays all the current subsystem definitions.

MS <subsystem id> displays that subsystem definition.

MS - <subsystem id> eliminates a subsystem definition.

MS <subsystem id>=(<processor id list>) defines or redefines a subsystem.

Example:

```
MS
--

   MS SYSTEM      = (GLOBAL,1,2,3,4) ,   CURRENTLY (GLOBAL,1,2)
   MS GLOBAL      = (GLOBAL)         ,   CURRENTLY (GLOBAL)
   MS TWO         = (2)              ,   CURRENTLY (2)
   MS BARBARA     = (2)              ,   CURRENTLY (2)
   MS RED         = (3)              ,   CURRENTLY EMPTY
   MS YELLOW      = (3)              ,   CURRENTLY EMPTY
   MS BLUE        = (1)              ,   CURRENTLY (1)

MS JUNK = (2)
-------------

   MS JUNK        = (2)              ,   CURRENTLY (2)

MS - JUNK
---------

   JUNK REMOVED
```

MU (Make User) Message

MU incorporates the PU (Privileged User) message.

Syntax:

```
-- MU ---<identifier>-----------------------------------------|
       |                  |                 | |             |
       |                  |- / <identifier> -| |- PRIVILEGED -|
       |                                                    |
       |-  -  --<identifier>-- PRIVILEGED -----------------|
```

Semantics:

The MU (Make User) message causes the first specified identifier to be entered into the userdirectory as a valid usercode, and the second specified identifier (if any) to be associated as a password with that usercode. The usercode may be specified as a privileged user. A usercode preceded by a minus sign has privileges removed from it. The MU action is subject to regulation by the "MU MODEL" entry in the SYSTEM/USERDATAFILE; MU may be disallowed entirely, or the PRIVILEGED option may be disabled.

Example:

```
   MU JOHN/DOE
   -----------

      JOHN/DOE PRIVILEGED

   MU- JOHN
   --------

      JOHN NOT PRIVILEGED
```

MX (Mix Entries) Message

MX replaces the M (Mix) message.

Revised Syntax:

```
-- MX ------------------------------------------------------------>
      |            |
      |-/1\- ALL -|

>------------------------------------------------------------------|
  |                                               |
  | |<------------------------------------------| |
  | |                                           | |
  |-----/1\- SWAPPER -----------------------------|
      |      --                               |
      |-/1\- MCSNAME ---------<mcsname>-------|
      |      ---      |      |                |
      |               |- = -|                 |
      |                                       |
      |-/1\- IN ---<subsystem id>-------------|
                  |                           |
                  |- ( <processor id list> ) -|
```

<processor id list>

```
  |<---------------|
  |   |      |     |
  |   |<- , -|     |
  |                |
------ GLOBAL ------|
    |  -       |
    |- 1 ------|
    |          |
    |- 2 ------|
    |          |
    |- 3 ------|
    |          |
    |- 4 ------|
```

Revised Semantics:

The MX (Mix Entries) request yields the same response as the J (Job and Task Structure Display) request, except that display lines (RSVP messages and DISPLAY) are printed with each task. If ALL is used, any active jobs or tasks which have been suppressed by the SUPPRESS message are displayed in addition to unsuppressed tasks.

When SWAPPER is specified, only jobs running in swapspace will be displayed.

When MCSNAME is specified, only jobs that originated from the specified MCS will be displayed.

On a B6800 multiprocessor system, each displayed task is preceded by a subsystem indicator: processor id for a local memory task, "G" for a global memory task, or blank for a task whose subsystem location is currently unassigned.

When IN is specified, only jobs with stacks running in the subsystem identified or the processor identified will be displayed.

Example:

```
MX
--

 -----JOB STRUCTURE-----

 S4087 JOB 50 ?RUN DICKEYIN("SUP
 4070 JOB 50? RUN DATABASE/DU
 S..L4083 50 DATABASE/DUMPANALYZER ON DMS
 4053 JOB 70 SYSTEM/CANDE
 D:DISPLAY:#
 E..4075 60 DMALGOL ON DMS (JKD) CANDE/CODE1100
 *S..4120 50 DMALGOL ON DMS
 3643 JOB 60 INTERIMFR26
 D: (JHHH) JHHHPATCH REMOVED ON INTERIM26 PK065
 W..4128 60 LIBRARY/MAINTENANCE
 R: RECOPY REQD: SYSTEM/FORTRAN
 D: COMPARE ERROR: MT114*
```

Example:

```
MX ALL MCSNAME=SYSTEM/CANDE
---------------------------
```

This message will display all active jobs (including suppressed) that originated from SYSTEM/CANDE.

Example: (B6800 Multi-Processor System):

```
MX SW IN(3)
-----------
```

This message will display all active jobs (excluding suppressed) whose stacks are running in swapspace in processor 3.

NET (Network) Message

Syntax:

```
-- NETWORK ----------|
   ---        |     |
              |- + -|
              |     |
              |- - -|
```

Semantics:

The NET (Network) message is used to attach a host to the network.

NET displays the current status of the network including the names, states and address of each of the known hosts in the network.

NET+ initiates contact with other hosts by initiating the SYSTEM/HOSTSERVICES library, which provides network communication functions to users and programs. Until this is done, no inter-host communications can take place.

NET- disconnects the host from the network by causing the SYSTEM/HOSTSERVICES library to go to EOJ.

Example:

```
NET
---

   LOCAL HOST IS NOT NETWORKING
   ------------------------------------------------------------
   BLUE[0]          LOCAL HOST(1)                DISCONNECTED

NET+
----

   This message will cause the execution of STARTNETWORK.

NET-
----

   This message will cause the execution of NETWORKDISCONNECT.
```

NEXT (Next Screen) Message

NEXT has been eliminated; see NS (Next Screen) for new message.

NS (Next Screen) Message

Syntax:

```
-- NS --|
```

Semantics:

The function of NS (Next Screen) is to bring up the next screen on a screen terminal if there is a next screen. The message is automatically typed in and the cursor positioned at the lower right corner of the screen.

OF (Optional File) Message

Revised Syntax:

```
--<mix number list>-- OF --|
```

Semantics:

Unchanged

Example:

Unchanged

OG (Overlay Goal) Message

Revised Syntax:

```
--<mix number list>-- OG ---------------|
                         |             |
                         |-<number>-|
```

Semantics:

Unchanged

Examples:

```
4869,5269 OG 18
---------------

   4869 OVERLAY GOAL=18%
   5269 OVERLAY GOAL=18%

5269 OG
-------

   5269 OVERLAY GOAL=11% PER MINUTE

4869 OG 20
----------

   4869 OVERLAY GOAL=20%
```

OK (Reactivate) Message

Revised Syntax:

```
--<mix number list>-- OK --|
```

Semantics:

Unchanged

Examples:

0963 OK

0931,0935 OK

OP (Options) Message
-- --------- -------

OP incorporates the SO (Set Options), RO (Reset Options) and TO (Test Options) messages.

Syntax:

```
-- OP ----------------------------|
        |     | |                |
        |- + -| |-<option list>-|
        |     |
        |- - -|
<option list>
 --------------------------------|
                                 |
      <----------------------|   |
                             |   |
   -----<number>-----------------|
         - OPEN ------------|
         - TERMINATE -------|
         - NOCHECK ---------|
         - LPBDONLY --------|
         - AUTORM ----------|
         - DIAGNOSTICS -----|
         - CDONLY ----------|
         - AUTORECOVERY ----|
         - DUPSUPERVISOR ---|
         - DUPINTRINSICS ---|
         - AUTODC ----------|
         - NODUMP ----------|
         - CPBDONLY --------|
         - CRUNCH ----------|
         - BACKUPBYJOBNR ---|
         - FULLTRANSLATION -|
         - NOFETCH ---------|
         - RESOURCECHECK ---|
         - NOSUMMARY -------|
         - DIRDEBUG --------|
         - CATALOGING ------|
         - NEWPERETRY ------|
         - OKTIMEANDDATE ---|
         - LOGPOSITIONING --|
         - SERIALNUMBER ----|
         - ARCHIVING -------|
         - CONTROLOLDWFL ---|
         - IORANGECHECK ----|
         - IODIAGNOSTICS ---|
         - USECATDEFAULT ---|
         - CATTEST ---------|
         - MCPTEST ---------|
```

Semantics:

The OP (Options) message displays, sets and resets options as follows:

OP   Displays all options and their respective states.

OP <option list>   Displays the options in the list and their respective states.

OP+   Displays all options that are set.

OP+<option list>   Sets all options in the option list.

OP- Displays all options that are reset.

OP-<option list> Resets the options in the option list.

Example:

```
OP
--

   ----- OPTIONS -----
    1 OPEN              2*TERMINATE
    3 NOCHECK           4*LPBDONLY
    5*AUTORM            6 DIAGNOSTICS
    7 CDONLY            8*AUTORECOVERY
    9 DUPSUPERVISOR    10 DUPINTRINSICS
   12*AUTODC           13 NODUMP
   14*CPBDONLY         16*CRUNCH
   17 BACKUPBYJOBNR    18 FULLTRANSLATION
   19*NOFETCH          20*RESOURCECHECK
   21*NOSUMMARY        22 DIRDEBUG
   23*CATALOGING       24 OKTIMEANDDATE
   25*NEWPERETRY       26*LOGPOSITIONING
   27*SERIALNUMBER     28*ARCHIVING
   29 CONTROLOLDWFL    31 IORANGECHECK
   43 IODIAGNOSTICS    45*USECATDEFAULT
   46*CATTEST          47 MCPTEST

OP + OPEN
---------

    1 OPEN SET

OP -
----

   ----- RESET OPTIONS -----
    1 OPEN              3 NOCHECK
    6 DIAGNOSTICS       7 CDONLY
    9 DUPSUPERVISOR    10 DUPINTRINSICS
   13 NODUMP           17 BACKUPBYJOBNR
   18 FULLTRANSLATION  22 DIRDEBUG
   24 OKTIMEANDDATE    29 CONTROLOLDWFL
   31 IORANGECHECK     43 IODIAGNOSTICS
   47 MCPTEST

OP - OPEN
---------

    1 OPEN RESET
```

The III.1 run-time system options are as follows:

<number>

A number appearing in this message must correspond to the number which identifies the option desired. (The run-time system options are not to be confused with the option word assigned to each job.)

OPEN (option 1)

When this option is set, a file-open message is displayed for each job whenever a file is opened.

TERMINATE (option 2)

When this option is set, abnormal job terminations result in an attempted program dump rather than a full memory dump. If this option is reset, such abnormal terminations result in a full memory dump.

NOCHECK (option 3)

When this option is set, memory dumps under both abnormal termination and FORGETCHECK conditions are inhibited. These dumps are automatic when NOCHECK is reset.

LPDONLY (option 4)

When this option is set, all printer output files are assigned to printer backup disk. These files can then be printed by the AUTOBACKUP routine.

AUTORM (option 5)

When this option is set, the MCP automatically removes the old file when a duplicate-file condition occurs. When AUTORM is reset, an RM or OF message is required when such a condition occurs.

DIAGNOSTICS (option 6)

When this option is set, an RSVP message (e.g., RF DEGRADATION) is displayed at the console any time the reliability of a hardware unit is degraded by a set amount. This option is meaningless when the MCP has been compiled without setting the MTBF option.

CDONLY (option 7)

With this option set, any job opening card input which is not internal to the jobfile is DSed. Also, no card reader may be labeled.

AUTORECOVERY (option 8)

When this option is set, a Halt/Load is attempted following all system fatal memory dumps (except a hung processor). DCP's which were running prior to the Halt/Load are subsequently reinitialized and the AP number is restored to the value previous to the halt.

If this option is reset, the above conditions do not happen. Furthermore, the mix limit of all queues is set to zero so that no jobs will be automatically restarted.

DUPSUPERVISOR (option 9)

This option is provided for use with directory reconstruction. If a code file <file title> has been designated as the supervisor program by a CS message and this option is set, at Halt/Load time the MCP will attempt to execute a code file <file title>/FMLYINX<nnn>. If this option is reset, the MCP will attempt to execute the designated supervisor program.

DUPINTRINSICS (option 10)

This option is provided for use with directory reconstruction. If a file <file title> has been designated as the intrinsics file by a CI message and this option is set, at Halt/Load time the system will attempt to use as the intrinsics file the code file <file title>/<family index>. If this option is reset, the code file <file title> will be used as the intrinsics file at Halt/Load time, regardless of the EU involved.

AUTODC (option 12)

When this option is set, the automatic generation of a Data Comm control stack is provided for when an executing job requests it.

NODUMP (option 13)

When this option is set, the MCP is prevented from attempting dumps to tape. Potential nonfatal dumps are denoted by a message at the supervisory console and logged. The source of a fatal dump is displayed in a system message at Halt/Load time. When this option is reset, dumps are taken in the normal fashion.

CPBDONLY (option 14)

When this option is set, all card punch output files are assigned to punch backup disk. These files can be punched by the AUTOBACKUP routine.

CRUNCH (option 16)

When this option is set, code files and backup disk files are automatically CRUNCHed when they are closed. Note that if this option is reset, no file can be CRUNCHed although the file may have been explicitly CRUNCHed by other program constructs.

BACKUPBYJOBNR (option 17)

When this option is set, jobs are printed by order of the job number. When reset, jobs are printed by order of lowest output print quantity to highest output print quantity.

FULLTRANSLATION (option 18)

When this option is set, every logical file is initialized with the file attribute TRANSLATE set to FULLTRANS. This allows software translation whenever translation is required and hardware translation is not provided.

NOFETCH (option 19)

When this option is set, FETCH statements entered by a WFL deck are disabled.

RESOURCECHECK (option 20)

When this option is set, it enables tape resource management.

NOSUMMARY (option 21)

When this option is set, the job summary output is suppressed if no backup files are produced. The job summary is printed if a task terminates abnormally.

DIRDEBUG (option 22)

CAUTION

> This option is intended for use only by the Burroughs Large Systems Plant.

CATALOGING (option 23)

At Halt/Load time CATALOGING is tested. If it is true, CATALOGLEVEL is initialized to CATALOGLEVELSET; if it is false, CATALOGLEVEL is set to zero. Note that a CATALOGING MCP is any MCP whose CATALOGLEVEL is greater than zero.

Note: A Halt/Load must occur after this option is set for CATALOGING to take effect.

OKTIMEANDDATE (option 24)

This option is used to verify that system TIME and DATE values are valid.

Verification will be required on B6800 multiprocessor systems regardless of the option setting if the clock of any processor varies from that of the leader by more than 60 seconds during the system initialization sequence. (Whenever B6800 time-of-day registers disagree as a multiprocessor system is initialized, they are all synchronized at the maximum time found in any of them.)

When verification is required, the current TIME and DATE settings will be displayed on the ODT and updated whenever ADM would ordinarily be updated. The operator may enter "TIMEOK" to resume normal processing after resetting any invalid time or date via the "DR" and "TR" messages.

NEWPERETRY (option 25)

This option invokes a special method of repositioning the tape after a write error occurs on any PE tape unit. Do not set NEWPERETRY unless all PE tape controls have attained the following RIN level:

CE-L3486-1
CON-5-L2189-6
MECH-R2254-19

LOGPOSITIONING (option 26)

When this option is set, the MCP will log the positioning actions of tape parity retry as well as the actual retries. This option should be set by a site having trouble in retrying tape errors; e.g., lost blocks on tape.

SERIALNUMBER (option 27)

When this option is set, scratch tapes will not be assigned to output tapes unless the SERIALNO attribute is used or the operator OUs the tape.

ARCHIVING (option 28)

This option enables the MCP archiving function. If this option is set and the catalog level of the system is greater than zero, an archive log will be created in which pertinent information will be stored for later processing by the SYSTEM/ARCHIVE utility. The name of the archive log will be "ARCHIVELOG/<date>/<time>", where <date> and <time> refer to the creation date and time of the file. If the archive log is not successfully set up, the option will automatically be reset by the system.

CONTROLOLDWFL (option 29)

When this option is set, all WFL input from the ODT is treated as new WFL. This option affects only job syntax entered through the ODT.

IORANGECHECK (option 31)

This option verifies that the disk address requested for an I/O is, in fact, within the range of any other rows of the file.

IODIAGNOSTICS (option 43)

CAUTION

> This option is intended for use only by the Burroughs Large Systems Plant.

USECATDEFAULT (option 45)

At Halt/Load time, this option and cataloging are tested. If both are true, all files have the attribute USECATALOG set to true by default.

CATTEST (option 46)

CAUTION

This option is intended for use only by the Burroughs Large Systems Plant.

MCPTEST (option 47)

CAUTION

This option is intended for use only by the Burroughs Large Systems Plant.

Setting or resetting any of these options (by number) should have no effect.

OT (Stack Cell Inspection) Message

Revised Syntax:

```
--<mix number>-- OT --<number>--|
```

Semantics:

Unchanged

Examples:

Unchanged

OU (Output Unit) Message

Revised Syntax:

```
--<mix number list>-- OU --<output device>--|
```

Semantics:

Unchanged

Examples:

Unchanged

P (Peripheral Status) Message

P has been eliminated; see PER (Peripheral Status) for new message.

PC (Print Configuration) Message

PC has been eliminated; see SC (System Configuration) for new message.

PER (Peripheral Status) Message

PER replaces the P (Peripheral) message.

Revised Syntax:

```
-- PER --------------------------------|
        |                              |
        | |<--------- , ---------|     |
        | |                      |     |
        |-----<device>-----------------|
             |                  |
             |- = --------------|
             |                  |
             |-<device>--- = ---|
                          |     |
                          |- - -|
```

Revised Semantics:

When the PER (Peripheral Status) message is entered, the status of the specified peripheral unit(s) is displayed on the supervisory console at which the message was entered.

If PER= or PER <device>= is used, unlabeled and not-ready devices are included; otherwise, they are not.

On a B6800 multiprocessor system, the display for each unit is followed by a bracketed list of processor numbers showing which subsystems are connected to the unit.

The remainder of the semantics remain unchanged.

New Example: (B6800 Multiprocessor System)

```
PER PK
------

----- PK STATUS -----
64  NOT AVAILABLE TO GROUP
66  NOT AVAILABLE TO GROUP
68*B  [152000] #1 DISK (26)[2,1]
70*B  [144732] #1 PACK (02)[2,1]
71*B  [004092] #1 USERS (00)[2,1]
```

PG (Purge) Message

The PG (Purge) message now allows the specification of tape density.

Revised Syntax:

```
---- PG ------ MT ---<unit number list>--------------------------|
   |         | |    |                      |                    |
   |- PGL -| |- PK -|                      |- ( --<density>-- ) -|
```

<density>

```
---- 200 -----|
  |           |
  |- 556 --|
  |           |
  |- 800 --|
  |           |
  |- 1600 -|
  |           |
  |- 6250 -|
```

Revised Semantics:

The PG (Purge) message may be used to purge specified tape units or disk packs if they are ready, not in use, and are write-enabled. For tapes, use of this message requires that the tape have serial numbers; these serial numbers are not disturbed by the message. If the tapes in question do not have serial numbers, use of the SN message is required. The density of the tape may be specified; however, when the density is specified, it applies to all tapes in the list.

The PGL form of this message, in addition, causes the specified tape units to be locked so that no job can automatically pick up the scratch tapes.

New Example:

```
PG MT 82 (1600)
---------------
```

MT82 WILL BE PURGED

For disk packs, the PG message causes the pack to be relabeled with the packname SCRATCH and made available to SCR programs. This message, intended primarily for maintenance, should not be used for disk packs in the normal course of work.

PP (Privileged Program) Message

Syntax:

```
-- PP ---------<file title>--|
       |     |
       |- - -|
```

Semantics:

The PP (Privileged Program) message designates a code file as a privileged program; i.e., it has the same capability as a program running under a privileged usercode. A code file title preceded by a minus sign has privileges removed from it.

Example:

```
PP (MIKE)OBJECT/HARD
--------------------

      (MIKE)OBJECT/HARD IS A PRIVILEGED PROGRAM

PP- (MIKE)OBJECT/HARD
---------------------

      (MIKE)OBJECT/HARD NON PRIVILEGED PROGRAM
```

PR (Priority) Message

Revised Syntax:

```
--<mix number list>-- PR --<number>--|
```

Semantics:

Unchanged

Examples:

```
4972, 4980 PR 90
----------------

4975 PR 30
----------
```

PU (Privileged User) Message

PU has been eliminated; see MU (Make User) for new message.

QT (Quit) Message

Revised Syntax:

```
--<mix number list>-- QT --|
```

Semantics:

Unchanged

Example:

```
6823 QT
-------
```

RC (Reconfigure) Message

In addition to the pre-III.1 syntax, the IV (Initialize and Verify) message has been incorporated.

Revised Syntax:

```
-- RC --- PK ---<unit number>---------------------------------------->
        |      |               |      |
        |- DK -|               |- IV -|

>--------------------------------------------------------------------|
   |                       |
   | |<------------------| |
   | |                   | |
   |---<parameter options>---|
```

<parameter options>

```
----/1\- OWNER ---------<name>------------------|
   |                |     |                     |
   |                |- = -|                     |
   |                                            |
   |-/1\- NAME ---------<packname>-------------|
   |               |     |                      |
   |               |- = -|                      |
   |                                            |
   |-/1\- SERIAL ---------<integer>------------|
   |                 |     |                    |
   |                 |- = -|                    |
   |                                            |
   |-/1\- BP ---------<integer>----------------|
   |             |     |                        |
   |             |- = -|                        |
   |                                            |
   |-/1\- FAMILYINDEX -- = --<number>----------|
   |                                  |         |
   |                                  |- KEEP -|
   |                                            |
   |-/1\- IC ----------------------------------|
   |                                            |
   |-/1\- SR ----------------------------------|
   |                                            |
   |-/1\- IAD ---------------------------------|
```

Revised Semantics:

The RC (Reconfigure) message is used to create a new set of volume labels on a disk pack or head-per-track disk. If the disk pack is to be a native mode base pack or interchange pack, a new master directory is also created. The IV option may be used to format and analyze all of the tracks of the specified pack.

The semantics for <parameter options> is unchanged.

RECONFIGURE Message

Syntax:

```
-- RECONGIFURE --- INSTALLATION --- AS ---<group list>--------|
                |                |      |                     |
                |- GROUP --------|      |- DEFAULT ---------|
                |                                             |
                |- INSTALLATION -- AS ---<installation id>-|
                                        |                     |
                                        |- DEFAULT ---------|
```

Semantics:

The RECONFIGURE message regroups the hardware resources according to the groups specified in the group list. If INSTALLATION is specified, all installation resources are reconfigured to the new configuration. If GROUP is specified, all resources assigned to the group to which the operator display terminal at which the message was entered will be reconfigured into the new configuration. A reconfiguration may cause a Halt/Load of existing groups which are being reconfigured into new groups.

Any subset of the <group id>s defined in the configuration file may be specified in the new configuration. Not all defined <group id>s need be specified and a <group id> cannot be specified more than once.

Example:

```
RECONFIGURE INSTALLATION AS THREEBY
RECONFIGURE INSTALLATION AS SYSA, SYSB, SYSC
RECONFIGURE GROUP AS TWOBY
RECONFIGURE GROUP AS RED
```

RES (Reserve) Message

RES incorporates the RET (Return) Message.

Revised Syntax:

```
-- RES ------------------------------------------------------->

>--- DK ---<unit number>--<reserve options>-----------------------|
  |      |                              |                   |
  |- PK -|                              |- COPY ERRORS -|
  |      |                              |                   |
  |- EU -|                              |- REMOVE ------|
  |                                                          |
  |-  -  --- DK ---<unit number>--<su or switch>------------|
            |      |
            |- PK -|
            |      |
            |- EU -|
```

<reserve options>

```
-------------------------------------------------------------|
  |                  | |                                     |
  |-<su or switch>-| |- AS BADDISK ---------------------|
  |                                                        |
  |- AS MAINT ------------------------------------------|
  |                                                        |
  |- LABEL ---------------------------------------------|
  |                                                        |
  |- SEGMENT --<number>----------------------------------|
                           |                              |
                           |- FOR --<number>---------------|
                           |                              |
                           |- THRU ---------------<number>-|
                                    |           |
                                    |- SEGMENT -|
```

<su or switch>

```
----------------------------------------------------------------|
  |                                                          |
  | |<-------- , -------|                                    |
  | |                   |                                    |
  | |        |<--- , ---| |                                  |
  | |        |          | |                                  |
  |--- SU ---<number>----------------------------------------|
  |                                                          |
  |- SU --<number>--- THRU ----------<number>----------------|
  |               |          |      |                        |
  |               |          |- SU -|                        |
  |               |                 |                        |
  |               |- FOR ---------|                          |
  |                                                          |
  | |<---------------------- , -------------------------|    |
  | |                                                   |    |
  | |            |<--- , ---|                           |    |
  | |            |          |                           |    |
  |--- SWITCH -----<number>-----------------------------------|
                 |                                     |
                 |-<number>--- THRU --------------<number>-|
                            |          |          |
                            |          |- SWITCH -|
                            |                     |
                            |- FOR -------------|
```

Revised Semantics:

The RES (Reserve) message allows a specified area to be removed from or returned to the head-per-track or disk pack subsystem.

Such an area to be reserved is placed in the IAD-disk pool or marked as a BADDISK or RESIDISK file. Only non-IAD disk areas may be so reserved.

Disk areas specified as IAD-disk may be returned to the available disk pool. If an area is not IAD-disk, an appropriate error message is displayed and the area is not returned. If an area is designated as not ready or write lockout, it is not returned.

Only one RES request can be active at any given time.

The semantics for <reserve options> and <su or switch> are unchanged.

Revised Example:

```
RES PK 96 SEGMENT 111111 FOR 25
-------------------------------

   1281/JOB 99 RESERVEDISK
```

and displays the following messages:

```
   1281      DATA MOVED IN SYSTEM/MCP117
   1281      PK096 BADDISK/FMLYINX1/UNIT96/AD111111 CREATED
```

Note that the files that are being affected by the copying action are displayed.

```
RET DK33 SWITCH 4 FOR 1
---------------

   3444 JOB 99 RETURNIADISK
```

RESTORE (Restore) Message

RESTORE has been eliminated; see SUPPRESS for new message.

RET (Return) Message

RET has been eliminated; see RES (Reserve) for new message.

RM (Remove) Message

Revised Syntax:

```
--<mix number>-- RM --|
```

Semantics:

Unchanged

Example:

Unchanged

RO (Reset Option) Message

RO has been eliminated; see OP (Options) for new message.

RR (Release Reader) Message

RR has been eliminated; see SR (Secure Reader) for new message.

RY (Ready) Message

In addition to the pre-III.1 syntax, the RY message has been modified to accommodate multiprocessor systems.

Revised Syntax:

```
             <------------- , -------------

-- RY -----<device>--<unit number list>-------------------
                                                          :
          - CPU --<processor number>----------------------:
                                                          :
          - MOD --<mod number>----------------------------:
                                  :                       :
                                  :- IN <processor number> -
```

Revised Semantics:

The RY (Ready) message causes the indicated devices to be made ready for use by the system if they are in remote status and have been made previously inaccessible via the SV message or rewind and lock. If the <mod number> is greater than or equal to the first mod of * GLOBAl tm Memory, the IN phrase is not used; global is used by default.

An off-line processor (CPU) may also be restored by the RY message as part of the restoring sequence, as follows:

a. Place the processor in LOCAL.
b. Enter RY CPU<nnn>. The system will display the following typical message:

    IC20(ZERO)30(some hexadecimal starting value); IIHF START

c. Make sure the processor is in LOCAL.
d. Set the B-register to hexadecimal 20. Note that everything in the registers must be right-justified.
e. Set the A-register to hexadecimal 0.
f. Set AROF and BROF.
g. Do a WRITE-IC.
h. Do a UNIT CLEAR.
i. Set the B-register to hexadecimal 30.
j. Set the A-register to the starting value.
k. Set AROF and BROF.
l. Do a WRITE-IC.
m. Do a UNIT CLEAR.
n. Set IIHF.
o. Set the processor to REMOTE.
p. Make certain that SCII, SCC1 and SCC2 are cycling.
q. Press START.

The RY CPU message cannot currently be used on B6800 multiprocessor systems, due to the architecture and relationship of CPUs to I/Os in the B6800; this restriction may be removed on a future software release.

* "GLOBAL Memory" is a trademark of Burroughs Corporation.

Revised Example:

```
RY MT 113
---------

   MT 113 READY

RY MOD 1 IN 2
-------------

   MOD 1 IN LOCAL 2 WILL BE READY
```

S (Scheduled Mix Entries) Message

Revised Syntax:

```
-- S ---------------------------------------------------------------->
        |           |
        |-/1\- ALL -|

>--------------------------------------------------------------------
    |                                                       |
    |  |<-------------------------------------------------| |
    |  |                                                  | |
    |-----/1\- SWAPPER -----------------------------------|
           |           --                                 |
           |-/1\- MCSNAME -----------<mcsname>------------|
           |      ---           |   |                     |
           |                    |- = -|                   |
           |                                              |
           |-/1\- IN ---<subsystem id>--------------------|
                          |                               |
                          |- ( <processor id list> ) -----|
```

```
<processor id list>

   |<-------------|
   |   |     |    |
   |   |<- , -|   |
   |              |
------ GLOBAL ------|
     |  -       |
     |- 1 ------|
     |          |
     |- 2 ------|
     |          |
     |- 3 ------|
     |          |
     |- 4 ------|
```

Revised Semantics:

The S (Scheduled Mix Entries) message causes a display of those tasks which are scheduled. This display does not include jobs still in one of the job queues.

When SWAPPER is specified, only jobs running in swapspace will be displayed.

When MCSNAME is specified, only jobs that originated from the specified MCS will be displayed.

On a B6800 multiprocessor system, each displayed task is preceded by a subsystem indicator: processor id for a local memory task, "G" for a global memory task, or blank for a task whose subsystem location is currently unassigned.

When IN is specified, only jobs with stacks running in the subsystem identified or the processor identified will be displayed.

A typical response to an S message is as follows:

```
-----2 SCHEDULED ENTRIES-----

6950 JOB 70 SYSTEM/CANDE
6962/6963 55 SYSTEM/DUMPANALYZER
```

Swap jobs are flagged with an "#" between the priority number and the file name. The number in the active entry heading is the total number of active entries including suppressed entries.

New Example:

```
S ALL MCSNAME=SYSTEM/CANDE
--------------------------
```

This message will display all scheduled jobs (including suppressed) that originated from SYSTEM/CANDE.

New Example: (B6800 Multi-Processor System):

```
S SW IN(3)
----------
```

This message will display all scheduled jobs (excluding suppressed) whose stacks are running in swapspace in processor 3.

SB (Substitute Backup) Message

Revised Syntax:

```
        |<----------------------- , ----------------------|
        :                                                 |
        :                          |<-------------------| |
        :                          |                    | |
-- SB -----/1\- DISK -----------------/1\- DISK -------------|
         |              |  |     |   |                |
         |-/1\- PACK ---|  |- = -|   |-/1\- PACK -----|
         |              |            |                |
         |-/1\- TAPE ---|            |-/1\- TAPE -----|
         |              |            |                |
         |-/1\- TAPE7 --|            |-/1\- TAPE7 ----|
         |              |            |                |
         |-/1\- TAPE9 --|            |-/1\- TAPE9 ----|
         |              |            |                |
         |-/1\- PETAPE -|            |-/1\- PETAPE ---|
                                     |                |
                                     |-/1\- DLBACKUP -|
```

Semantics:

Semantics are unchanged, except for the following:

The default SB specifications are DISK = DLBACKUP and PACK = PACK.

Example:

Unchanged

SC (System Configuration) Message

SC replaces the PC (Print Configuration) message.

Syntax:

```
-- SC --|
```

Semantics:

When the SC (System Configuration) message is entered, the system responds by displaying the current system configuration at the supervisory console from which the message is entered.

A typical response to an SC message, which yields the total configuration of the system, is as follows:

```
3 PROCESSORS 1-3

    D0=02100

3 MULTIPLEXORS 1-3

MPX            LIMIT          TRAFFIC

1 (MOD III)     15               2

2 (MOD III)     15               0

3 (MOD III)     13               2

MEMORY STATUS

    24 IN USE 0-23

PROGRAMMED HALT/LOADS:  0

STRINGS PRESENT: 1-3
```

SI (System Intrinsics) Message

SI incorporates the CI (Change Intrinsics) and WI (What Intrinsics) messages.

Syntax:

```
-- SI -------------------|
        |              |
        |- + ----------|
        |              |
        |-<file title>-|
```

Semantics:

The SI (System Intrinsics) message loads or displays the system intrinsics as follows:

SI Displays the current intrinsics.

SI <file title>
Loads the intrinsics named in <file title>.

SI+
Loads the default intrinsics. The default is a define in the Controller such that an installation may change it if so desired.

Jobs that are currently active use the "old" intrinsics stack; new jobs use the "new" intrinsics stack.

Examples:

```
   SI SYSTEM/X
   -----------

      6029 JOB 99 INITIALIZEINTRINSICSTUFF

   then

      SYSTEM/X (LOADED)

   SI+
   ---

      6039 JOB 99 INITIALIZEINTRINSICSTUFF

   then

      SYSTEM/INTRINSICS (LOADED)

   SI
   --

      INTRINSICS:  SYSTEM/INTRINSICS
```

SM (Send Message) Message

Revised Syntax:

```
--<mix number list>-- SM ---------<control message>--|
                        |       |
                        |- : -|
```

Semantics:

Unchanged

Example:

Unchanged

SN (Serial Number) Message

The SN (Serial Number) message now allows the specification of tape density.

Revised Syntax:

```
---- SN ---- MT --<sn option list>--------------------------|
  |         |                   |                   |
  |- SNL -|                   |- ( --<density>-- ) -|
```

<sn option list>

```
   |<---------------------------- , ----------------------------|
   |                                                            |
   |                              |<-----------------|          |
   |                              |                  |          |
----<unit number list>-------------/6\---<digit>-----------------|
                       |      |  |        |          |       |
                       |- : -|  |        |-<letter>-|       |
                                 |                             |
                                 |      |<-----------------|   |
                                 |      |                  |   |
                                 |- " ---/6\---<digit>------ " -|
                                                |          |
                                                |-<letter>-|
```

<density>

```
---- 200 -----|
  |           |
  |- 556 --|
  |        |
  |- 800 --|
  |        |
  |- 1600 -|
  |        |
  |- 6250 -|
```

The SN (Serial Number) message is used to purge (with optional locking) and assign a serial number a tape unit. The serial number may consist of up to six alphanumeric characters. If a number is used, it is right justified with leading zeros added. Any serial number containing alphabetic characters or any quoted alphanumeric string is left justified with trailing blanks. The density of the tape may be specified; however, it applies to all tapes in the list.

When the SNL form of the message is used, the tape is locked as well as purged so that no job can automatically pick up the scratch tape.

New Example:

```
SN MT 82 MIKE (1600)
--------------------

   MT82 WILL BE SN-ED

P MT
----

   82*P [MIKE  ] 1600 #1 1:00 S C R A T C H
```

An attempt to SN a tape that has been locked causes the UNIT LOCKED message to be displayed.

SO (Set Option) Message

SO has been eliminated; see OP (Options) for new message.

SR (Secure Reader) Message

SR incorporates the RR (Release Reader) message.

Revised Syntax:

```
-- SR ----------- CR --<unit number list>--|
      |         |
      |- - -|
```

Revised Semantics:

The SR (Secure Reader) messages causes the system to check the usercode/password combination and reject all card decks entered into the indicated card reader which do not contain a USER system control card. If SR- is entered, the security restrictions imposed on the indicated card reader by a prior SR message will be removed.

```
SR CR10
-------

   CR10 SECURED
```

If an attempt is made to enter a deck without a USER system control card, the system responds with the typical message:

```
   0941 CONTROL CARD ERROR

SR- CR10
--------

   CR10 RELEASED
```

ST (Stop) Message

Revised Syntax:

```
--<mix number list>-- ST --|
```

Semantics:

Unchanged

Example:

Unchanged

SUPPRESS (Suppress) Message

SUPPRESS incorporates the RESTORE message.

Revised Syntax:

```
--<mix number list>-- SUPPRESS ------------|
                                 |       |
                                 |-  -  -|
```

Semantics:

The SUPPRESS message prevents indicated jobs from appearing in a display of the job mix when they are active. If SUPPRESS- is entered, the suppression is lifted. Only a mix (e.g., A,C,J) message containing the ALL qualifier may cause such jobs to appear in the mix display.

If an active suppressed task goes into a waiting state, it is displayed. When it returns to an active state, it is again suppressed.

Example.

```
180 SUPPRESS
------------

   -----ACTIVE ENTRIES-----

   180 JOB 70 SYSTEM/CANDE
   ......193 70 STACK2/CANDE
   179 JOB 80 DCP/0
```

results in

```
   -----ACTIVE ENTRIES-----

   180x 193 70 STACK2/CANDE
   179 JOB 80 DCP/0
```

SV (Save) Message

In addition to the pre-III.1 syntax, the SV message has been modified to accommodate multiprocessor systems.

Revised Syntax:

```
              |<------------- , -------------|
              |                              |
-- SV -----<device>--<unit number list>-------------------|
          |                                             |
          |- CPU --<processor number>--------------------|
          |                                             |
          |- MOD --<mod number>--------------------------|
                                   |                    |
                                   |- IN <processor number> -|
```

Revised Semantics:

The SV (Save) message is provided to allow a unit to be made inaccessible to the system. If the <mod number> is greater than or equal to the first mod of * GLOBAL tm Memory, the IN phrase is not used; global is used by default.

This message may be used in response to an RF DEGRADATION message to take the indicated peripheral unit off-line as soon as it is not in use by the current job which is using it. Such an SV message will inhibit further RSVP messages regarding this unit until an appropriate RY message is input. In addition, the unit is automatically placed in a "saved" status so that upon completion of this job the unit cannot be reassigned. Once a unit has been saved by an SV message, the reliability factor for that unit will remain unchanged until an RY message is entered for that unit.

The SV message may also be used to take a memory module off-line. The restrictions are that this message may not reference a module which is already saved or off-line, may not reference module #0, and may not reference the numerically highest module which was on-line at the time of the last Halt/Load. An arbitrarily long period of time may expire between the input of the SV message and the time the module becomes completely saved; all in-use areas must be returned to the system before the process is complete. A system message is issued when the module is off-line.

The SV CPU message cannot currently be used on B6800 multiprocessor systems, due to the architecture and relationship of CPUs to I/Os in the B6800; this restriction may be removed on a future software release.

Revised Example:

```
SV MT 114
---------

   MT 114 SAVED

   -----MT STATUS-----
   114*P [000001] SAVED

SV MOD 1 IN 2
-------------

   MOD 1 IN LOCAL 2 WILL BE SAVED
```

A processor (CPU) may be taken off-line by the SV CPU<nnn> message once the CONDITIONAL HALT switch for the desired CPU has been turned on. In order to save one CPU, the other CPUs in a multiprocessor system must be on-line. Use the following procedure to save a CPU:

a. Put the CONDITIONAL HALT switch in the up position. If the switch is up, OFF is displayed on the MDL panel in the A register. If the switch is not up, the system responds with

   SET CONDITIONAL HALT SWITCH FOR CPU<nnn>

b. Enter SV CPU<nnn>
c. Put the processor in local mode and CLEAR (CL) the unit.

Note: The processor must remain in local mode; if it is in remote mode, the next Halt/Load will pick up the processor.

SW (Swapper) Message

The SW (Swapper) message has been revised to accommodate B6800 multiprocessor systems.

Revised Syntax:

```
-- SW ----------------------------------------------------------->

>-----------------------------------------------------------------|
   |                                  |                           |
   |- ( --<processor id list>-- ) -   |-<swapper control parameters>-|
   |                                                              |
   |- - --------------------------------------------------------- |
   |          |                                                   |
   |          |- ( --<processor id list>-- ) ---------------------|
   |                                                              |
   |- + ----------------------------------------------------------|
           |                                |  |                  |
           |- ( --<processor id list>-- ) -|  |-<start options>--------|
```

<processor id list>

```
    |<----------|
    |      |    |
    |      |<- , -|
    |           |
------ 1 ---------|
      |    |
      |- 2 -|
      |    |
      |- 3 -|
      |    |
      |- 4 -|
```

<swapper control parameters>

```
---- MINTIMESLICE ------ = --<value>--|
  |                 |
  |- EXPTIME ---------|
  |                 |
  |- MAXSLICENUMBER --|
  |                 |
  |- MAXCORE ---------|
  |                 |
  |- RATIO -----------|
  |                 |
  |- CORESIZE --------|
  |                 |
  |- EXPRESSRESERVE --|
  |                 |
  |- EXPMAXCORE ------|
  |                 |
  |- PRIORITYBIAS ----|
  |                 |
  |- UTILIZATIONBIAS -|
  |                 |
  |- IOBIAS ----------|
  |                 |
  |- MEMORYBIAS ------|
```

<start options>

```
    |<--------------------------------|
    |                                 |
------ CORESIZE ---------<integer>------|
    |            |    |          |
    |            |- = -|          |
    |                            |
    |- ON --<family>---------------|
```

Revised Semantics:

The SW (Swapper) message is used to interrogate, control, initiate and terminate the SWAPPER mechanism, which transfers tasks between central memory and mass storage on a demand or time-slice basis. The central memory area is procured automatically according to parameters which may be specified in this message; the mass storage area must be present as a file with the title *SYSTEM/SWAPDISK. A visible independent runner named SWAPPER is run to perform the swapping. In a B6800 Multiprocessor (Tightly-Coupled) system, SWAPPER may be used in each local subsystem.

See System Software Operational Guide, Vol. 2 (Form No. 5001688), Article 16, for a description of <swapper control parameters> and <start options>. See also GENERAL note D2906, "Changes in the SWAPPER Mechanism".

The SW+ and SW- forms are used to initiate and terminate swapping, respectively. The SW forms are used to interrogate or alter parameters while SWAPPER is running. (An additional temporary feature is present to enhance compatibility between III.1 and earlier systems: the command "SW", when entered while no SWAPPER is active, is equivalent to "SW+".)

A <processor id list> is significant only on a TC system. If the list is present, the SW message applies to each of the named local subsystems. If the list specifies a subsystem that is not present on the system, the message is rejected. If no list is specified on a TC system, the SW+ message causes SWAPPER to be initiated in each local subsystem, while the SW- and SW messages are applied to each SWAPPER that is running.

<Swapper control parameters> may be entered to alter the specified parameter values while SWAPPER is running.

<Start options> may be entered in a SW+ command to specify the CORESIZE parameter (number of 990-word slots to be allocated in central memory) and/or the disk/pack <family> on which *SYSTEM/SWAPDISK resides. Every row of *SYSTEM/SWAPDISK must reside on a family which is exchanged to each local processor in which SWAPPER is to be run. If a SWAPPER is already running, this clause may not be specified on the SW+ message to initiate additional SWAPPER(s).

Examples:

SW+

Initiate SWAPPER (one copy in a monolithic system, one copy in each local subsystem on a TC system).

SW+(1,3) ON SWD CORESIZE=180

Initiate SWAPPER in the local memory of processors 1 and 3; find *SYSTEM/SWAPDISK on SWD; set its CORESIZE to 180 slots for each SWAPPER.

SW+(2) CORESIZE 140

Initiate SWAPPER in the local memory of processor 2; set its CORESIZE to 140 slots. SWAPPER may or may not be already running in some other subsystem.

SW (1) EXPRESSRESERVE 40 EXPMAXCORE 25

For the SWAPPER running in processor 1, set the EXPRESS reserve area size to 40 slots, with an upper bound of 25 slots on the size of subspace for an express task.

SW

Cause each SWAPPER running to display its current parameter settings. The data are displayed in a series of messages which may be seen in response to a MSG input message.

SW (3)

Cause the SWAPPER running in processor 3 to display its parameters, as above. SW - (3) Cause the SWAPPER running in processor 3 to terminate. It will immediately acknowledge "<mix> GOING AWAY"; it will terminate when there are no longer any swap tasks that require that subsystem. Meanwhile, no new tasks except offspring of existing processor-3 swaptasks will be initiated as swaptasks in processor 3.

SW-

Cause all SWAPPERs to terminate. On a TC system, each one will acknowledge and then terminate when not required, as above. On a monolithic system, SWAPPER will acknowledge and then terminate when there are no more swaptasks running. Meanwhile, no new swaptasks will be started except offspring of existing swaptasks.

TD (Time and Date) Message

TD incorporates the WD (What Date) and WT (What Time) messages.

Syntax:

```
-- TD --|
```

Semantics:

When the TD (Time and Date) message is entered, the system responds by displaying the current time and date in the following manner:

```
DATE IS MONDAY APR 23,1979 (79113) hh:mm:ss
```

where hh, mm and ss are digits denoting hours, minutes and seconds respectively; 79113 indicates the year is 1979 and the date is day 113.

TD (Tape Directory) Message

TD (Tape Directory) has been replaced by TDIR (Tape Directory); see TDIR for new message.

TDIR (Tape Directory) Message

Syntax

```
                          |<--------- , --------|
                          |                     |
-- TDIR -----------------<unit number>------|
         |         |    |                   |
         |- SPO ---|    |-<tape name>---|
         |         |
         |- PUNCH -|
```

Semantics:

TDIR (Tape Directory) message initiates SYSTEM/FILEDATA, which reads the directory of the specified tape. The default output is to the printer. As many intermixed tape names or unit numbers as desired may be specified, each being separated from the next by commas.

Example:

```
TDIR SPO X

 B6700 SYSTEM 277 REPORT OF 06/09/75 AT 09:48:01.
 VERSION 2.7.380
 TAPE = X/FILE000. ON UNIT   115
 SERIAL#=123456 CREATED 6/09/75
 9-TRACK (PE) 1600 BPI
 *SYSTEM/DUMPALL
           TAPEDIRECTORY INPUT WAS:
 "TPDIR SPO X"
                  .
                  .
                  .
```

TI (Times) Message

Revised Syntax:

```
--<mix number list>-- TI --|
```

Semantics:

Unchanged

Example:

Unchanged

TL (Transfer Log) Message

TL replaces the LR (Log Release) Message.

Syntax:

```
-- TL --|
```

Semantics:

A new system log is generated by entering a TL (Transfer Log) message.

The previous log is named:

SUMLOG/<system number>/<mmddyy>/<number>

as appropriate (where number is a log serial number from 1 to 999999) and may be saved or discarded as desired. IAD logs specified at cold start time do not have a new log file created; instead, the rows of the system log are rotated. A typical response to a TL message is as follows:

```
7275 SYSTEM/SUMLOG CHANGED TO SUMLOG/227/053075/000005 ON DISK DK033
```

TO (Test Option) Message

TO has been eliminated; see OP (Options) for new message.

UA (Unit Available) Message

UA has been eliminated; see UR (Unit Reserved) for new message.

UL (UnLabeled) Message

Revised Syntax:

```
--<mix number list>-- UL ---------<device>--<unit number>--|
                         |   |
                         |- : -|
```

Semantics:

Unchanged

Example:

Unchanged

UR (Unit Reserved) Message

UR incorporates the UA (Unit Available) message.

Syntax:

```
-- UR -----------<device specifier list>--|
       |     |
       |-  - -|
```

<device specifier list>

```
      |<------------ , ------------|
      |                            |
      |          |<------ , -----| |
      |          |               | |
-------<device>---<unit number>-----------------------------------|
 |                                                                |
 |- DK ---<unit number>-- MPX --<mpx no>-- PATH --<path no>-|
 |      |
 |- PK -|
 |      |
 |- MT -|
```

Semantics:

The UR (Unit Reserved) message is used to reserve a unit to allow FE maintenance to be performed. The unit is no longer available for assignment. When a path is specified, only the path is reserved, not the unit. The unit may be restored to the system by entering UR-.

Example:

```
   UR LP13
   -------

        LP13 RESERVED
```

W (Waiting Mix Entries) Message

Revised Syntax:

```
-- W ---------------------------------------------------------->
        |            |
        |-/1\- ALL  -|

>-------------------------------------------------------------|
   |                                                        |
   | |<-----------------------------------------|           |
   | |                                          |           |
   |-----/1\- SWAPPER --------------------------|-----------|
         |        --                            |
         |-/1\- MCSNAME ----------<mcsname>------|
         |      ---         |     |              |
         |                  |- = -|              |
         |                                       |
         |-/1\- IN ---<subsystem id>-------------|
                         |                       |
                         |- ( <processor id list> ) -|
```

```
<processor id list>

   |<-------------|
   |   |     |    |
   |   |<- , -|   |
   |              |
------ GLOBAL ------|
     |  -      |
     |- 1 ------|
     |          |
     |- 2 ------|
     |          |
     |- 3 ------|
     |          |
     |- 4 ------|
```

Revised Semantics:

The W (Waiting Mix Entries) message causes a display of those tasks which require operation action in order to continue; i.e., those suspended on an RSVP condition. The reason for suspension is included in the display.

When SWAPPER is specified, only jobs running in swapspace will be displayed.

When MCSNAME is specified, only jobs that originated from the specified MCS will be displayed.

On a B6800 multiprocessor system, each displayed task is preceded by a subsystem indicator: processor id for a local memory task, "G" for a global memory task, or blank for a task whose subsystem location is currently unassigned.

When IN is specified, only jobs with stacks running in the subsystem identified or the processor identified will be displayed.

A typical response to a W message is as follows:

```
        -----1 WAITING ENTRIES-----

        3570/3571 50 LIBRARY MAINTENANCE
        1100  SECT REQ ON MCPMAST PK068*
```

Swap jobs are flagged with an "#" between the priority number and the file name. The number in the active entry heading is the total number of active entries including suppressed entries.

New Example:

```
   W ALL MCSNAME=SYSTEM/CANDE
   --------------------------
```

This message will display all waiting jobs (including suppressed) that originated from SYSTEM/CANDE.

New Example: (B6800 Multi-Processor System):

```
W SW IN(3)
----------
```

This message will display all waiting jobs (excluding suppressed) whose stacks are running in swapspace in processor 3.

WD (What Date) Message

WD has been eliminated; see TD (Time and Date) for new message.

WI (What Intrinsic) Message

WI has been eliminated; see SI (System Intrinsics) for new message.

WM (What MCP) Message

Syntax:

Unchanged

Semantics:

Unchanged, except as follows:

A typical response to the WM message is:

```
MCP: SYSTEM/MCP 31.200.900
H/L UNIT: 64
COMPILED: 3/22/79 @ 16:27:48 (NEWP 31.197)
      COMPILE TIME OPTIONS ARE:
      DIAGNOSTICS         DISKCHECK       MTBF
      LINEINFO            LOCKTRACE
      REVERSEPAPERTAPE    SWAPTRACE
H/L REASON: MANUAL
GROUPID: ONEBY
HOSTNAME: BLUE
SYSTEM SERIAL NO: 277
CATALOG LEVEL: 0
NEXT MCP: NOT SPECIFIED
```

WS (What Supervisor) Message

WS has been eliminated; see CS (Change Supervisor) for new message.

WT (What Time) Message

WT has been eliminated; see TD (Time and Date) for new message.

XS (Exceed Schedule) Message

XS has been eliminated; see FS (Force From Schedule) for new message.

PRIMITIVE Messages

The following "primitive" messages have been added:

??DUMP

```
-- ?? ----------------- DUMP --|
        |                 |
        |-<mix number>----|
```

The ??DUMP message will invoke a non-fatal memory dump (through the MCP procedure KEYIN).

??FS

```
-- ?? ------------------ FS --|
        |                 |
        |-<mix number>-|
```

The ??FS message causes the execution of the indicated scheduled job. If ??FS is used without the mix number, all scheduled jobs are executed.

??HS

```
-- ?? ------------------ HS --|
        |                 |
        |-<mix number>-|
```

The ??HS message is used to suspend job selection when the CONTROLLER will not accept the HS message. An example of the use of the ??HS message is to inhibit job selection immediately following a Halt/Load.

D2604 GENERAL - LIBRARIES

A library mechanism has been implemented, which provides a set of procedural 'entry points' which can be called by other programs.

See Appendix B, User Interface to Libraries, for details of implementation.

D2782 GENERAL - "KIND=DISK" VS. "FAMILYNAME"

With the implementation of named head-per-track disk families (especially the ability to name them something other than "DISK") and the ability to name a disk-pack family "DISK", the last major distinction between the two forms of random access mass storage devices has been eliminated. Ultimately the differences in the behavior of the operating system when assigning files declared as KIND=DISK and KIND=PACK will also be eliminated.

Currently (III.1) the selection of random access mass storage devices varies depending upon whether or not the KIND attribute has a value of DISK or PACK and whether or not the FAMILYNAME attribute has been assigned a value. If the KIND value is DISK then the system supplies the FAMILYNAME of "DISK" whether or not the FAMILYNAME attribute has been given another value by declaration, label-equation, or direct assignment. If the KIND value is PACK then the system supplies the FAMILYNAME of "PACK" unless it has been explicitly set to another value. In other words, the FAMILYNAME (even if assigned a value) is ignored if KIND is DISK and has two different default values if one has not been explicitly assigned.

On III.3 the values of DISK and PACK for the KIND attribute will become synonymous and the default value for FAMILYNAME will be "DISK". This has two implications:

1. The FAMILYNAME attribute will be used when assigning files whose KIND is DISK, eliminating the phenomenon of label-equated FAMILYNAMEs being ignored.

2. The assignment of files on the family "PACK" will require the explicit setting of the FAMILYNAME attribute (or the intervention of family substitution).

To aid in the transition to a single default value for FAMILYNAME, on III.2 a run-time warning will be given whenever a file from the family named "PACK" is assigned to a logical file that is declared with KIND=PACK and the FAMILYNAME has not been assigned. The warning and the noted problem can be eliminated by changing the logical file declaration to include FAMILYNAME="PACK."

D2794 GENERAL - DEIMPLEMENTATION OF "XALGOL" COMPILER

The XALGOL language will no longer be supported beginning with the III.2 system release; the compiler will not be supplied, the MCP will not allow execution of XALGOL code files, CANDE will not MAKE or otherwise handle XALGOL symbolic files, WFL will not recognize XALGOL as a legitimate compiler name.

The III.0 PR1 and III.1 XALGOL compiler will unconditionally produce a warning message describing its unavailability beginning with the III.2 release.

The III.0 PR1 and III.1 MCPs will unconditionally produce a warning in the job summary indicating that the code file will no longer run beginning on III.2. This same message will be propagated back to CANDE users attempting to execute XALGOL code files (if their CANDEGETMSG flag is set in the USERDATAFILE) (or they have explicitly set messages for their session).

D2797 GENERAL - DEIMPLEMENTATION OF OLD INTRINSICS

The intrinsics that support formatted and freefield READ and WRITE statements in pre-II.7 **ALGOL** code files (and II.7 and II.8 code files that were compiled by an ALGOL compiler that **was** itself compiled with the $OLDFORMATIO compiler generation option) will be deimplemented on **the** III.2 system release.

The III.0 PRI and III.1 MCPs will produce a warning message in the job summary (when attempting to link to these intrinsics) indicating that the code files will no longer run beginning on the III.2 system release. This same message will be displayed to CANDE users if their CANDEGETMSG flag is set in the USERDATAFILE (or they have explicitly set messages for their session).

D2798 GENERAL - DEIMPLEMENTATION OF OLD INTRINSICS

The intrinsics that support formatted and freefield READ, WRITE, PRINT and PUNCH statements for pre-II.4 FORTRAN code files will be deimplemented on the III.2 system release.

Also, the intrinsic that supported the OPEN, CHANGE and INQUIRE statements (dealing with file attributes) for pre-II.8 PR2 FORTRAN code files will be ueimplemented.

The III.0 PRI and III.1 MCPs will produce a warning message in the job summary (when attempting to link to these intrinsics) indicating that the code files will no longer run beginning on the III.2 system release. This same message will be displayed to CANDE users if their CANDEGETMSG flag is set in the USERDATAFILE (or they have explicitly set messages for their session).

D2799 GENERAL - DEIMPLEMENTATION OF OLD INTRINSICS

The intrinsics that perform I/O operations for pre-II.4 BASIC code files (and BASIC code files between II.4 and II.8 that were compiled with $OLDBASIC set) will be deimplemented on the III.2 system release.

The III.0 PRI and III.1 MCPs will produce a warning message in the job summary (when attempting to link to these intrinsics) indicating that the code files will no longer run beginning on the III.2 system release. This same message will be displayed to CANDE users if their CANDEGETMSG flag is set in the USERDATAFILE (or they have explicitly set messages for their session).

D2800 GENERAL - DEIMPLEMENTATION OF OLD INTRINSICS

The intrinsics that handled file and task attributes for pre-II.8 PRI PL/I code files will be deimplemented on the III.2 system release.

The III.0 PRI and III.1 MCPs will produce a warning message in the job summary (when attempting to link to these intrinsics) indicating that the code files will no longer run beginning on the III.2 system release. This same message will be displayed to CANDE users if their CANDEGETMSG flag is set in the USERDATAFILE (or they have explicitly set messages for their session).

D2861 GENERAL - "B6800" MULTIPROCESSOR SYSTEMS

INTRODUCTION

The architecture of the B6800 multiprocessor systems represents a significant departure by Burroughs from past practices for large systems. Previous systems had been 'monolithic' from both the hardware and software viewpoints. This meant that all hardware modules had direct access to all other hardware modules, and that the machine was presented to the operating system in a 'flat' manner, such that memory, processors, and peripheral channels were identical and interchangeable. This system organization worked very well for single or dual processor systems. However, with the advent of three, four, or more processors on a system certain problems can be encountered that give rise to increasing operating system overhead on monolithic systems. The basis for certain of these problems is contention for resources; the resources may be memory, processor time, channel time, or system data structures. Other problems, in terms of cost in dollars, stem from the fact that all systems, even single processor systems, paid the cost of the hardware to allow for the possibility of multiprocessor configurations. Also, prior architectures allowed for no convenient method of resource sharing by separately controlled systems, and had only minimal capabilities to reconfigure hardware resources in order to cope with dynamically changing requirements. In order to avoid these problems, the B6800 systems were designed such that:

single processor systems do not pay for the additional cost of providing for the possibility of multiprocessor systems,

resources are arranged in such a way that contention on multiprocessor systems is reduced as much as possible,

resources may be shared by autonomous systems, which cooperate in such a manner that a failure by either system will not cause the other system to fail,

resources may be re-configured via software, under the users control, in order to cope with shifts in load, or failure of portions of a system.

These design goals of the B6800 systems are met through a combination of hardware **and** software features which are described in this and related D-Notes.

B6800 HARDWARE OVERVIEW

The B6800 single processor system consists of a central processor, a maintenance processor, a memory cabinet containing up to one-half million words of memory, a peripheral control cabinet, and a power supply cabinet.

The central processor of the B6800 is code compatible with other members of the B6000/B7000 series of machines.

The memory cabinet of the B6800 system is dedicated to a single processor, thus allowing simpler, less costly, and faster hardware in the memory control. The maximum memory configuration allowed is 524,288 words. This memory may consist of either four stacks of 131,072 words each or eight stacks of 65,536 words each. The two types of stacks may not be intermixed. Memory which is attached to the processor in this manner is termed 'local memory', and is not visible to other processors in a multiprocessor configuration. The memory control of the processor contains a jumper block that will provide for local memory to be either two-way or four-way interlaced, depending on the physical memory configuration available. As the interlacing of the memory modules can significantly improve the systems performance, it is recommended that the memory be interlaced whenever possible.

The multiplexor on the B6800 is physically part of the central processor, and operates much like an additional operator family. However, the multiplexor is capable of running independently of the processor in order to service the peripherals, and interrupts the processor operations only when necessary to transfer data to or from memory. The multiplexor on a B6800 may have up to twenty I/O channels. These channels are dedicated to the processor to which they are attached and are not visible to other processors in a multiprocessor configuration.

In order to configure individual B6800 processor systems into multiprocessor systems, additional hardware is required. This hardware consists of an adapter in each processor which allows access to the * GLOBAL tm Memory cabinet. The GLOBAL Memory cabinet contains an independent power supply, logic to control access to the memory in the cabinet, and up to one-half million words of memory. This memory may be configured in the same manner as described for the local memory. In addition, the cabinet contains one processor adapter port for each processor that will be attached to the cabinet, up to a maximum of four. This allows a B6800 multiprocessor system to have up to four processors, with a total possible memory capacity of two and one-half million words, and up to eighty I/O channels.

* "GLOBAL Memory" is a trademark of Burroughs Corporation.

GLOBAL MEMORY Description

The * GLOBAL tm Memory Module has a control panel with indicators and switches for maintenance and configuration. The following information about system configuration will be of interest:

The GMM provides ports to four requestors, named A, B, C and D, and hubs to four modules of memory, named L, M, N and P. Each requestor port is cabled to a specific B6800 processor. Each memory hub provides 65536 or 131072 words, depending on memory type. A set of LOCKOUT switches near the upper center of the panel can be used to lock out each port (from every hub) and each hub (from every port). The hub lockout switches may be used to put part of the memory off-line. The port lockout switches may be used to prevent a particular B6800 processor from accessing the GMM subsystem; that processor can then run as a simplex system.

There is a matrix of REQUESTOR PORT LOCKOUT switches in the upper right portion of the panel; these may be used to set any combination of paths from particular requestors to particular memory ports. These switches should all be NORMAL for multiprocessor operation. While they may be used to divide the GLOBAL memory resources among simplex systems in a split system, it is more convenient under most circumstances to use the soft configuration capabilities of the system, which are explained in D-Note 2867.

The MEMORY IN MAINTENANCE MODE switches should be down for normal MCP operation; they must be up for certain stand-alone diagnostic and dump programs.

The HUB INTERLACE switches provide for interlacing memory addresses to improve access time to sequential locations. Specific pairs of hubs may be two-way interlaced, providing both are accessible to all requestor ports in use. (Which hub can be paired with which depends on the address ranges associated with each memory hub, as set by field engineering personnel. In the typical configuration where L spans 80000-9FFFF, M spans A0000-BFFFF, etc., the pairs are L-M and N-P.) If all four hubs are accessible to all requestor ports in use, they may be four-way interlaced. As interlacing the memory modules can significantly improve the system's performance, it is recommended that memory be interlaced whenever possible.

In addition to its function as a memory subsystem, the * GLOBAL tm Memory cabinet contains a Global System Control facility. The GSC provides communication and control paths between the various B6800 processors comprising the system.

Peripheral Configuration Cabinet

The system may include a Peripheral Control Cabinet, with switch panels for various groups of exchanged units. When the system is being split, individual units can be made ready on paths to only the desired processors. When the system is joined (tightly-coupled), any unit ready for one processor must be ready for all processors to which it is exchanged. While the use of the Peripheral Control Cabinet will physically disable a processor from using a particular unit, the use of a system configuration file is sufficient to logically disable one or more units for any processor or group of processors. If a configuration file is in use, it is recommended that all exchanged devices be ready to all processors via the Peripheral Configuration Cabinet. This will prevent accidental "UNIT NOT READY" situations following a reconfiguration. See D-Note 2867 for an explanation of soft configuration of GLOBAL memory systems.

SOFTWARE OVERVIEW

B6800 multiprocessor systems can be operated in three basic modes:

Independent or simplex, where each processor operates as a single processor system,

Multiprocessor (Tightly-Coupled or TC), where two or more processors operate as a multiprocessor system under the control of one MCP.

Shared Resources (Loosely-Coupled or LC), where two or more independent systems may cooperate in order to share resources.

In an installation with four processors attached to a GLOBAL memory cabinet each of the three modes can exist simultaneously. Each mode offers certain capabilities, and can be used to advantage, depending on the needs of the user.

If access to GLOBAL memory is allowed, the memory may be used either as PRIVATE memory, dedicated to the processor as an extension of local memory or as SHARED memory, which can be used to communicate and cooperate with other systems. Both modes of memory may exist simultaneously, or either without the other. Management of memory mode is explained in D-Note 2867 on soft configuration of GLOBAL memory systems.

Access to peripherals may be controlled through either the Peripheral Control Cabinet, **or** via the configuration file. It is more convenient to utilize the capabilities of **the** configuration file. If communication has been established with other systems, **then** peripherals may be logically "moved" between systems to which they are exchanged via **the** "ACQUIRE" and "FREE" commands. These commands are explained in D-Note 2867 on **soft** configuration.

SIMPLEX SYSTEMS

A B6800 processor operating in simplex mode is operating as a single processor system. In this mode the processor may or may not have access to GLOBAL memory. If physical access to GLOBAL memory is prohibited, the processor will be unable to communicate with other processors that may be attached to the GMM cabinet. Access to the GMM can be controlled physically via the LOCAL/REMOTE switch on the processor maintenance panel, or via the REQUESTOR LOCKOUT switch on the GMM cabinet.

The LOCAL/REMOTE switch also controls access to DATACOM processors and Reader/Sorter processors. The switch must be in the REMOTE position in order to allow these devices to be accessed. The processor may be Halt/Loaded with the switch in the LOCAL position to prohibit access to GLOBAL memory, and then switched to the REMOTE position after the system is initialized in order to access either the DCP or Reader/Sorter subsystem.

If physical access from a processor to the * GLOBAL tm Memory cabinet is allowed, the GSC can be used for configuration control, even though hardware switches or software configuration permit no Global Memory access by that processor; thus, even simplex systems may cooperate via the "ACQUIRE" and "FREE" commands.

MULTIPROCESSOR SYSTEMS

Multiprocessor (Tightly-Coupled or TC) mode involves two or more B6800 processors operating as' a single multiprocessor system. Each processor has local memory, and GLOBAL memory is utilized for any code or data structures that must be visible to all processors. Peripheral devices on TC systems may either be exchanged or non-exchanged. All devices may be used from any processor, even if the processor does not have a physical path to the device; in that case, the actual I/O will be done by a processor that has a physical path.

Memory Subsystems

The B6800 Multiprocessor architecture allows program stacks, data and code to reside in either Global or local memory. Any item residing in local memory is visible only to one processor; items in Global memory are visible to all processors. Since the B6800 hardware is optimized for quick access to local memory, and since there is no contention for local memory from other processors, it is the policy of the MCP to keep as much information as practical in local memory.

The MCP ("D0") stack, with its code and data, resides in Global memory, along with all datacom messages and most independent-runner stacks. In addition, by default, all MCS and data base stacks and buffers are allocated in GLOBAL memory. As access times to local memory are faster than to GLOBAL memory it is preferable that, whenever possible, the user specify via the SUBSYSTEM attribute that MCS stacks and data bases run in local memory.

User processes ("task" stacks), data, segment dictionaries ("D1" stacks) and code generally reside in local memory. (With rare exceptions, the data allocated by a stack reside in the same location as the stack.) Once a task is initiated in the local memory of one processor, it normally remains there throughout its life. If the task is a SWAPTASK and the task is not restricted by its SUBSYSTEM attribute or parental visibility, and SWAPPER has established SWAPSPACES in two or more local memories, then the task can migrate from one local memory subsystem to another. See GENERAL note D2906 for a complete explanation of SWAPPER on TC systems.

SUBSYSTEM Definitions

A multiprocessor B6800 system contains one GLOBAL memory subsystem and a local memory subsystem for each processor. The local memory subsystems are identified by their processor ID numbers; the Global memory subsystem is identified by the letter G or the word GLOBAL.

Subsystems may also be given names in the form of alphanumeric identifiers. Subsystem identifiers are defined by the system operator through the ODT command MS (Make Subsystem; see GENERAL note D2535). The reserved identifier SYSTEM always stands for the entire system as currently configured. The reserved identifier GLOBAL always stands for just the Global memory subsystem. (With software release III.1, any B6700 simplex or multiprocessor system, or any B6800 simplex system, is considered as comprising a single subsystem in which all the main memory is "GLOBAL.")

Other subsystems may be defined including any one or more memory subsystems in the system. The set of defined subsystems can be overlapping. For example, four subsystems could be defined by the commands

```
    MS RED(1)
    MS BLUE(2)
    MS PURPLE(1,2)
    MS PINK(1,G)
```

by which processor 1 is named RED, processor 2 is named BLUE, PURPLE defines a subsystem containing both processors (but not Global), and PINK refers to processor 1 and Global.

If a subsystem identifier has not been defined via an MS command, or has been removed via an MS- command, that identifier is undefined. If an identifier has been defined, but none of the subsystems listed exist on the system, that subsystem is said to be "empty." For example, MS YELLOW(3) would define an empty subsystem on a system which had no processor numbered 3.

Subsystem identifiers may be used as values of the SUBSYSTEM attribute in tasks and queues; see the next subsection.

The system operator will see and will sometimes use the processor ID numbers identifying subsystems, but any specifications from the user or programmer must use identifiers defined via MS. The mapping of subsystem identifiers to processor numbers may be changed by the system operator without changing user programs or control decks.

The subsystem identifier definitions are stored in the JOBDESC file, and retained across Halt/Loads in the same manner as ADM specifications and job queue parameters.

All communication with the operator and user employs the notations defined above. However, users of SYSTEM/DUMPANALYZER and of such MCP interfaces as SYSTEMSTATUS and GETSTATUS may encounter the "box" notation used internally by the MCP: memory subsystems are an array of "boxes" indexed by "box numbers." Global memory has number 1 (GLOBALBOX); local memory is indexed by the processor ID number +1. The index zero indicates that no box assignment has been made.

SUBSYSTEM Attribute

The user or programmer specifies a subsystem by using the TASK attribute SUBSYSTEM, whose value is a simple name.

Subsystem may be assigned an EBCDIC character string containing up to 17 upper-case alphanumeric characters, starting with a letter. The value returned is the last value assigned, or a null string if no value has been assigned. The assigned value is translated into a set of memory subsystems (locals specified by processor, and/or Global) according to the MS definitions currently in force. The SUBSYSTEM attribute is a <simple name> in WFL or CANDE, POINTER-valued in ALGOL, ALPHA in COBOL and STRING-valued in PL/I.

The subsystem attribute may be assigned a value only when the TASK is inactive. It is not automatically propagated from a parent task to its offspring.

A SUBSYSTEM attribute may be assigned to a job queue, via the ODT command MQ; e.g., "MQ 0 SUBSYSTEM=BLUE". Any job entered in that queue will be assigned the specified subsystem. The attribute specification may be removed from the queue with "MQ 0 SUBSYSTEM.", where a period follows the word SUBSYSTEM. The primary use of a SUBSYSTEM specification is to influence the choice of subsystem in which to locate a task. Such use will be discussed further under "Task Initiation."

Process Families

Processes on the B6700 or B6800 may be related to each other in various ways. A "job" is the manifestation of a set of one or more WFL statements entered via a card reader, typed in at an ODT, or initiated via such language constructs as START, STARTJOB or ZIP. (The CANDE commands WFL and COPY do not create jobs in this sense.) A job may spawn other tasks, and these may spawn more. Such offspring tasks are generally dependent on and remain related to their parents; the exceptions are creation of new jobs (via STARTJOB or ZIP) and of independent tasks (via ALGOL RUN, etc.).

Task Initiation

By default, the MCP on a multiprocessor system will allocate jobs and subroutines of jobs to Global memory, but it will allocate all other user tasks (with the exceptions noted above) to local memory. Jobs are run by default in Global to permit maximum flexibility in the placement of offspring tasks.

The algorithm for selecting the subsystem for a new task is as follows:

> If the task will be dependent upon an ancestor, and that ancestor is in a local-memory subsystem, the new task will go into the same subsystem.
>
> If the task will be dependent upon an ancestor in the Global memory subsystem, and there is any possibility that it could store an uplevel pointer reference into any ancestor, the new task will go into the same (Global) subsystem. [See MCP note D2654 for further discussion of uplevel pointers.]
>
> If the task is not constrained to the same subsystem as an ancestor, then any SUBSYSTEM specification will be considered: If no SUBSYSTEM attribute has been specified for this task, or if the the named SUBSYSTEM is undefined, or if the named SUBSYSTEM is empty, then all local subsystems in the system are considered. If a non-empty SUBSYSTEM has been specified, then if it contains only Global memory the task will be placed in Global memory; otherwise, all local subsystems included in

the named SUBSYSTEM are considered.

If the task is to be a swaptask, and there is a swapspace active in a subsystem being considered, the task will be assigned to the swapspace. There may be one swapspace in each local memory. Only those swapspaces which fall within any subsystem restrictions placed upon the task by either the user or the system will be considered.

All non-swap tasks pending initiation are entered into a schedule queue, from which they are removed in priority order. (If some task in a subsystem has been SUSPENDED BY SYSTEM, or if a higher-priority task eligible for that subsystem is scheduled, other tasks will not be initiated in that subsystem, but tasks may be started in another subsystem.) If more than one local subsystem is being considered for a task, the task will be tested for fit in each subsystem. If it fits in none, it will remain in the schedule and be reconsidered from time to time. If it fits in one, it will be initiated in that one. If it fits in more than one, it will be initiated in the subsystem of whichever processor has recently been least busy. (The notion of "fit" is based on current and projected memory utilization.)

The SUBSYSTEM attribute is treated as a request, not a binding demand, to guide placement of a task. There are three circumstances in which the attribute setting must be disregarded:

1. The named subsystem is undefined.

2. The named subsystem is empty.

3. The task was constrained by ancestral relationship to a subsystem outside the named set. The firm rule is that a dependent task must not be more global than its parent.

Whenever a SUBSYSTEM has been specified but had to be disregarded, a warning message is displayed (and logged) at the start of the task.

Peripheral Subsystem

On a TC system it is possible to have exchanged devices, non-exchanged exchangeable devices, and non-exchangeable devices. As each processor has its own multiplexor, I/O will be more efficient if, for those devices which are exchangeable, those devices are exchanged to each processor. The rationale for this is as follows: the MCP attempts to place all user stacks and associated data in local memory structures. This includes file buffers, and for direct files, the direct arrays. If the device associated with the file is exchanged to the processor in whose local memory the task is executing, that processor will do the I/O in the normal manner. If the processor executing the task does not have a physical path to the device, two actions must be taken: first, the buffers for the file must be moved such that the processor (multiplexor) that will actually do the I/O can see them (i.e., to GLOBAL memory); second, the appropriate processor must be interrupted in order to initiate the physical I/O. For large numbers of I/Os this additional overhead can have a significant impact on throughput of the system. Performance will therefore be improved by either a) exchanging all exchangeable devices to all processors or b) running programs requiring non-exchanged devices on a processor that can see the required device or c) executing the task from GLOBAL memory. The choices are listed in order of preference.

Multiple Halt/load Units

Each processor in a TC system may have its own Halt/Load unit, or family. Alternatively, two or more processors of a TC system may share the same Halt/Load unit or family. The only restrictions are a) all MCP code files on all Halt/Load units must be identical, b) all members of the Halt/Load family must be exchanged to each of the proce:sors using that family. Identical MCPs means that the code files must have the same timestamp. The code files do not have to have the same title.

Memory Dumps

Due to the configuration of TC systems, memory dumps are slightly different from simplex systems. The processor that detects the fault causing the dump, or the processor that answered the operator input requesting a dump, becomes the "lead" processor for the dump. This processor will control the dump routine, and all other processors will act to cooperate in the dump at the request of the lead processor. The lead processor is the only processor which will display the dump cause on the console. The lead processor must have a tape drive and an ODT available in order to take the dump.

The memory dump will progress as follows: first, the lead processor will dump its local memory to tape; second, the lead processor will dump GLOBAL memory to tape; third, the lead processor will request each of the other processors, one at a time, to transfer their local memory to a buffer in GLOBAL, which the lead processor will write to tape. The lead processor will display on the console the memory address and processor number of the memory being dumped. GLOBAL memory addresses will not have a processor number displayed with them.

Configuration rules

The following configuration rules apply to Multiprocessor systems:

1. Each processor must have access to at least one tape drive and an ODT in order to take memory dumps.

2. At least one processor must have a card reader in order to load off-line (stand-alone) tests, and to perform the initial cold start.

3. Operator Display Consoles (ODTs) must be configured such that the ODTs attached to different processors fall into different MINTERM groups; i.e., for a multiprocessor system with two processors, if processor one has ODTs with unit numbers of 42 and 43, the second processor must not have ODTs with unit numbers in the range 44 - 57.

4. Peripheral unit numbers must be unique throughout the multiprocessor system: when a given unit is exchanged to different multiplexors, it must have the same unit number designation in each. When a non-exchanged peripheral unit appears on one multiplexor, that number must not be used for any unit on any other multiplexor in the same multiprocessor system.

SHARED RESOURCES SYSTEMS

B6800 systems may be configured into what are termed Shared Resources (Loosely-Coupled or LC) systems. Each LC system may consist of one to four processors with their associated local memories, and some quantity of GLOBAL memory. Each system has its own MCP and set of peripherals. Peripherals are not shared between LC systems simultaneously but control of units may be passed from one system to another through the "FREE" and "ACQUIRE" operator input commands. Each LC system is both logically and physically independent of any other system attached to the GLOBAL memory cabinet. Each may be independently Halt/Loaded, and failure of one system will not cause any of the other systems to fail.

Using the features provided by the system software, an installation may create a set of independent systems which can communicate and cooperate with each other. The workload may be separated based on criteria developed by the installation, and resources may be allocated dynamically among the systems based upon the load.

See GENERAL note D2867 for a detailed description of "Soft Configuration of Global Memory Systems".

GLOBAL Memory Cabinet

In addition to the features mentioned for TC systems, the hardware of the GMM provides capabilities to support LC systems. The GMM contains a communications bus which allows one processor to send a message to another processor or to one of a set of processors. The GMM also contains registers which allow the software to "name" processors or sets of processors. Each set of processors with the same "name" is considered to be a member of the same system. Each processor adapter port in the GMM contains a set of "access registers". These registers allow the software to specify which processor families (i.e., processors with the same name) are allowed to access which areas of GLOBAL memory. Access is controlled on 4k word chunks, with any processor family having either NO access, READ ONLY access, or READ WRITE access to each chunk. Initial allocation of GLOBAL memory is controlled via the system configuration file. See D-Note 2867 on soft configuration for details. Each system which expects to operate as an LC system must have access to at least one 4K chunk of GLOBAL memory. This area will belong to the system, which will have READ WRITE access to the area. The area will be SHARED with other systems as a communication area. The other systems will be granted READ access to the area, but not write access.

The granularity of GLOBAL memory allocation for the III.1 system release is restricted to 16K chunks, i.e. the system will allocate a minimum of four of the 4K chunks as a communications area. This restriction will be lifted on a future release.

GLOBAL Memory Networks

Each system operating in a loosely-coupled mode is considered to be a HOST in a network. A GLOBAL memory network may consist of up to four hosts. All communication between systems operating in a GLOBAL memory network occur either through the GMM communications bus or via shared areas in GLOBAL memory. The III.1 system release does not support loosely coupling via data communications networks.

HOSTNAME

Each HOST in the network is identified by a HOSTNAME. HOSTNAMEs must be unique; i.e., no two systems in the network may have the same HOSTNAME at the same time. This name is used by programs and users to identify the location of resources to which access is desired. A hostname consists of 1 to 17 alphanumeric characters, beginning with a letter. A system's hostname may be set or modified at any time that the system is not in actual contact with another host. Each host must have a hostname assigned before attempting to establish communication with another host. Hosts which are in communication are said to be "attached to the network". Hosts which are not in communication are said to be "disconnected from the network". Hostnames are set via the ODT HN (HostName) command. HN <name> will establish a hostname. HN- will remove the hostname. HN will display the current hostname.

See D-Note 2535 concerning ODT commands for the full semantics of the HN command.

Host Usercodes

Each host in the network must have a "host" usercode. This user code is used for system, as opposed to user, communications. For instance, an inquiry from the ODT which is directed to another host will have the host usercode passed along with the inquiry. The host usercode is established for a given host through the HU ODT command. This command specifies the usercode to be sent with all "system" requests to another host. HU <usercode> will set the host usercode. HU- will delete the current host usercode. HU will display the current host usercode.

If an attempt is made to perform an inter-host system communication, and a host usercode has not been established via the HU command, or the other host does not recognize the usercode supplied as a "SYSTEMUSER" then an error will be generated, and the request will be aborted.

A new locator for the userdata interface exists in order to support the concept of host usercodes. The locator is "SYSTEMUSER". This is the mnemonic to establish any usercode as a "system" user. See D-Note 2535 on SYSTEM/MAKEUSER for further details concerning the SYSTEMUSER function.

Network Initialization

Once a hostname and host usercode have been established, the system can attempt to join an existing network or initiate an network activity. The NET (NETwork) ODT command is used to attach a host to the network. NET+ will initiate contact with other hosts. NET- will disconnect the host from the network. NET will display the current status of the network. The display will present the names, states, and addresses of each of the known hosts in the network. In addition to establishing contact with other hosts in the network, the NET+ command will cause the SYSTEM/HOSTSERVICES library to be fired up. This library provides network communications functions to users and programs. The SYSTEM/HOSTSERVICES library will be discussed in more detail below. The NET- command will cause the SYSTEM/HOSTSERVICES library to go to EOJ. See GENERAL note D2535 for details concerning the NET command.

HOSTNAME Attribute

The HOSTNAME attribute exists as both a task and a file attribute. When the HOSTNAME attribute has been set to a hostname that is not the local hostname, an attempt will be made to initiate the task, or open the file at the specified host. The attribute may be set at any time prior to task initiation or file open time. The HOSTNAME attribute may be read at anytime. HOSTNAME may be label equated as a file attribute. The HOSTNAME attribute is the mechanism used to indicate to the system where a resource is to be found, or where an action is to take place.

SYSTEM/HOSTSERVICES Library

The SYSTEM/HOSTSERVICES library provides functions to users and programs that are necessary to implement communication across the network. The functions implemented for the III.1 system release are:

1. Operator inquiry and control. This allows either the system operator or users at terminals to check the status of and control tasks running at another host, or to inquire about the status of the host itself.

2. File transfer. Using the COPY syntax, it is possible to transfer a file from one host to another.

3. Logical I/O. Files may be created, read, written, or updated at another host.

4. Job transfer. WFL decks may be transferred from one host to another and executed at the second host.

5. Tasking. Using normal IPC syntax, WFL or a user program, it is possible to initiate and control a task or tasks at another host.

6. Inter-program communication: Using PORTS and SIGNALS or PORT files user programs may establish direct program to program communication across the network. The same techniques may be used to communicate between programs running at the same host.

Operator Inquiry and Control

In order for either a system operator or user to make inquiries or attempt to control actions taking place at another host in the network, there must be a method of detecting messages that are destined for a remote host. This is done through the system console and through CANDE by recognizing messages that start with the keyword "AT". From the console, a message may be directed to another host, e.g., one with the hostname of "SOMEWHERE", by entering the following:

```
AT SOMEWHERE <ANY TEXT STRING> ETX
```

The Controller recognizes the AT as indicating that the message is destined for a remote host. The Controller will not syntax check the message <ANY TEXT STRING>, but will simply pass the message to the Host Services library. The library will establish a dialog with the host SOMEWHERE if a) the host is known, and b) a dialog does not already exist. The library will transmit the message, receive the reply, and pass the reply to the Controller to be displayed on the console. CANDE users may direct commands to remote hosts by entering the following:

```
?AT <hostname> <command>
```

CANDE will pass the command to the Controller. The message will follow the same path outlined above, but the reply will be routed back to the users terminal.

A usercode is contained in each message. If no usercode is available (i.e., terminal usercode not set or command not from CANDE), the host usercode is used.

When receiving a message, the usercode in the message is tested for its privilege (i.e., whether it is a "systemuser" or not).

File Transfer

Using the COPY syntax of Library Maintenance, files may be copied from one host to another. Files may be copied either from the local host to a remote host, or from a remote host to the local host.

Examples:

```
COPY SYSTEM/MCP TO PACK(HOSTNAME=A)

COPY X FROM YOURS(KIND=PACK,HOSTNAME=B) TO OURS(PACK)

COPY MY/FILE AS YOUR/FILE FROM PACK TO PACK(HOSTNAME=C)
```

The following restrictions exist for the III.1 release:

1. The file transfer capability is available only in new WFL.

2. A minimal subset of the COPY syntax is supported for the III.1 release. Not supported are ADD, COMPARE, BACKUP, CATALOG, lists of multiple files, multiple sources or destinations, AREACLASS, FAMILYINDEX, SINGLEPACK, INTERCHANGE volume specifications, copies involving tape as source or destination, copies ONTO, copies involving directories, copies with the local host as both the source and the destination. When a command to copy files is entered on the ODT, any files which have any usercode (<usercode> or *) omitted will assume the host usercode (not *).

3. The following file types cannot be transferred with the COPY available on III.1: FILETYPE not equal zero, files with BLOCKSIZE larger than 1600 words, BCL files, files with unallocated areas in the logical middle of the file, non disk files, duplicated files, installation allocated disk files, interchange pack files.

4. The restrictions listed under Logical I/O regarding file names and mnemonic-valued attributes also apply to files being transferred across systems.

Family substitution will not occur on the cooperating host (the host at which the transfer was not initiated). The file transfer must be performed by a job that has a usercode associated with it and that usercode will be assumed to all filenames without an explicit usercode (as usual).

These restrictions may be removed on future releases.

Logical I/O

The Host Services library will support logical I/O across the network in order to allow the user to read, write, and create files at another host. The file attribute HOSTNAME can be set either in the users program, or via label equation. This protocol requires the task to be running under a usercode. No checks are made for "systemuser" usercode.

Examples

```
1)  ?BEGIN JOB FOREIGN/COMPILE;
       COMPILE PROG COBOL;
       COBOL FILE CARD(KIND=DISK,TITLE=S/PROG,HOSTNAME=D);
    ?END JOB


2)  BEGIN
      FILE F(KIND=DISK,FILETYPE=8,TITLE="THE/FILE.",
            HOSTNAME="E.");
      ARRAY A[0:12];
      LABEL EOF;
      WHILE TRUE DO
```

```
          BEGIN
            READ(F,12,A)[EOF];
            .
            .
          END;
      EOF:
        CLOSE(F);
      END.
```

```
  3)  BEGIN
        FILE F(KIND=PACK,MAXRECSIZE=14,TITLE="OVER/THERE.",
               NEWFILE=TRUE,AREAS=10,AREASIZE=100,
               HOSTNAME="E.");
        ARRAY A[0:12];
        BOOLEAN DONE;
        DO
          BEGIN
            .
            WRITE(F,12,A);
            .
            .
          END
        UNTIL DONE;
        LOCK(F);
      END.
```

In the III.1 system software release, the following restrictions are in effect on logical I/O operations performed across systems via HOSTSERVICES:

1. Only FILETYPEs of 0, 3 and 8 are allowed.

2. The attribute KIND must be set. The only values allowed for KIND are TAPE, TAPE7, TAPE9, TAPEPE, TAPEGCR, PACK, DISK, READER, PRINTER and PUNCH.

3. No UPDATE files are allowed. UPDATEFILE must be FALSE.

4. MAXRECSIZE must be less than 10,000 characters.

5. USE routines are not supported.

6. In order to create a file, the attribute NEWFILE must be TRUE. If the attribute is not set, or is set to FALSE, HOSTSERVICES will search for an existing file. Note that NEWFILE cannot be changed by the FA command if the program hangs on a "no file" condition.

   If NEWFILE=TRUE, UNITS=CHARACTERS and INTMODE=SINGLE, the file actually created will have UNITS=WORDS.

7. No relative I/O; no keyed I/O.

8. No direct I/O.

9. RSVPs will not propagate back to the user's host.

10. There must be a USERCODE associated with the task performing logical I/O operations.

11. Use of the following attributes is not supported by HOSTSERVICES. If the attribute can be set, any value assigned before the file is opened will be ignored at file assignment; any attempt to read or write the attribute after the file has been opened will generate an attribute error.

    AREACLASS
    ASSIGNTIME
    ATTVALUE
    ATTYPE
    BLOCK
    CYLINDERMODE
    ENABLEINPUT
    EOF
    ERRORTYPE
    EXCLUSIVE
    IAD
    INTERCHANGE
    IOCLOCKS
    IOINERROR
    MYUSE
    POPULATION
    PRESENT
    RECEPTIONS
    RECORD
    RECORDINERROR
    RESIDENT
    ROWADDRESS

ROWSINUSE
SIZEMODE
SIZE2
SPEED
STATE
TANKING
TRANSMISSIONS
UNITNO

12. The following attributes may be set before a file is opened and the value specified will be used by HOSTSERVICES; however, an attempt to access the attribute after the file is opened will generate an attribute error.

    AREASIZE
    FILETYPE
    UNITS

13. For each of the following mnemonic-valued attributes, the values listed below it are not supported by HOSTSERVICES. A request to open or copy a file will be refused by HOSTSERVICES if one of these attributes has been set to a value that is not supported.

    A. SECURITY TYPE
       CONTROLLED

    B. FILEKIND
       APLDATA
       APLWORKSOURCE
       ARCHIVELOG
       BACKUPPRINTER
       BDDATA
       CATALOG
       FIRMWARE
       INFOFILE
       LCOBOLSL3CODE
       LCOBOLSL5CODE
       LCOBOLSYMBOL
       MDLCODE
       MDLSYMBOL
       NULLFILE
       RECOVERYFILE
       REMOTEAUDIT
       REMOTEBACKUP
       RSNETFILE
       RSPCODE
       RSSORTTABLE
       SCHEDULEFILE
       UCRFILE

    C. EXTMODE or INTMODE
       BCL
       HEX

14. The attributes FILEKIND and PROTECTION are applicable only to disk files. If a file is assigned to a non-disk device, any attempt to access these attributes will generate an attribute error.

15. The file attribute OPEN may not be set to FALSE to close a file; an explicit close must be performed.

16. The file attributes FILEKIND, SECURITYTYPE, SECURITYUSE and SECURITYGUARD are ignored if a permanent disk file is assigned (NEWFILE=FALSE). If, before the file is opened, any of these attributes is set to a value different from that of the physical file on disk, the value of the file attribute will not be changed in the physical file at file assignment time. However, if the attribute is set after the file has been opened, the physical file will be changed accordingly.

17. Under HOSTSERVICES logical I/O, the default value for the file attribute TRANSLATE is FULLTRANS. The value DEFAULTTRANS is not supported; any request for DEFAULTTRANS will be changed to FULLTRANS.

18. Only file names which are valid under "new" (post-II.9) WFL are acceptable to HOSTSERVICES. This restriction applies to any attribute which has a file name as its value, including TITLE and SECURITYGUARD.

19. If the SERIALNO attribute is set before a file is opened, serial numbers for other than the first reel are ignored.

20. The TITLE of a disk file cannot be changed while the file is open.

21. If a file has SECURITYTYPE GUARDED, the associated guard file must grant access only by usercode; i.e., the access specification in the guardfile must be of the following form:

    USERCODE X;

If the access specification is in either of the two following forms, access to the file will not be granted to the specified program:

```
PROGRAM Y;
USERCODE X USING PROGRAM Y;
```

22. For files accessed via HOSTSERVICES, the default values for INTMODE and EXTMODE are different from the defaults used for normal logical I/O. INTMODE and EXTMODE should be explicitly set if the defaults used by HOSTSERVICES do not give the desired results.

Some of these restrictions may be removed on future releases.

Job Transfer

WFL has been modified to interface to the Host Services library. This allows users to transfer entire job decks to a remote host for interpretation and execution. This feature is invoked by prefacing the job deck with a card as follows:

```
<I> AT <hostname> BEGIN JOB
```

and ending the job deck with a card as follows:

```
<I> END JOB
```

The <I> indicates the invalid character in column one of the card image. The invalid character is not allowed between the AT <hostname> and the BEGIN JOB. This insures that a deck dropped into a reader need only start with <I> BEGIN JOB in order to avoid a <I> AT <hostname> that has been left in the card reader. No status will be returned for WFL remote jobs. be assumed for WFL remote jobs. A usercode/password must appear explicitly within the job deck. No check will be made for "systemuser" usercode.

For the III.1 release, the transfer of BINARY formatted decks is not supported.

Tasking

Tasks may be initiated at another host by setting the task attribute HOSTNAME to a hostname that is not the local host. The task attribute is available in WFL and in the user languages. It may be set at any time prior to task initiation.

This protocol requires the job to be running under a usercode. No checks are made for "systemuser" usercodes.

Examples

```
 1)    ?BEGIN JOB FOREIGN/TASK;
          RUN T;
            OPTION=FAULT,ARRAYS;
            PRIORITY=40;
            HOSTNAME=F;
          PROCESS RUN Z;
            HOSTNAME=G;
       ?END JOB


 2)    BEGIN
         TASK T;
         PROCEDURE EXT; EXTERNAL;
           REPLACE T.NAME BY "FOREIGN/PROG.";
           REPLACE T.HOSTNAME BY "H.";
           PROCESS EXT[T];
           WHILE T.STATUS > 0 DO
             WAITANDRESET(MYSELF.EXCEPTIONEVENT);
       END.
```

The following limitations exist on III.1:

1. Compile and go is not supported across the network.

2. The CONTINUE statement is not supported across the network.

3. The following attributes are not allowed to be used across the network: EXCEPTIONTASK, EXCEPTIONEVENT, PARTNER, TASKFILE.

4. Tasks initiated at a remote host may have one parameter, which is an array of less than 512 words in length. The array will be passed by value.

5. If the task attribute FILECARDS is used, the following restrictions apply to the label equation string:

   - All file names must conform to new (post-II.9) WFL syntax. File names must be composed of only alphanumeric characters.

Any specification of file KIND must conform to new (post-II.9) WFL syntax. No KIND lists are allowed.

These restrictions may be removed on a later release.

INTER-PROGRAM COMMUNICATION

An entirely new mechanism for supporting inter-program communication has been introduced on III.1. This mechanism, called PORTS and SIGNALS, is a very general, very flexible program to program communication tool. Any two programs located anywhere in the network will be able to establish a communication path and exchange data. A new file kind, PORT, has been introduced to allow easy access to this mechanism for the user languages. See Appendix D, "Implementation of Inter-Process Communication", for details on the use of this feature.

D2867 GENERAL - SOFT CONFIGURATION OF GLOBAL MEMORY SYSTEMS

Release III.1 of the B6700/B6800 system software offers users the ability to reconfigure a B6800 multiprocessor system as a shared resources or loosely-coupled (LC) system. Use of this feature allows a site to create a set of cooperating and/or independent subsystems, each consisting of a group of shared or dedicated hardware resources. A companion feature is the Multiprocessor System, which is described in D-note D2861. The reader should be familiar with D-note D2861 before continuing.

Soft configurations of * GLOBAL tm Memory systems are system configurations controlled by the operating system. Although a processor on a Global Memory System may physically be capable of using particular hardware resources such as a pack or memory module, it may be restricted logically from doing so. A processor may also have software restrictions as to other processors with which it may cooperate and/or communicate.

* "GLOBAL Memory" is a trademark of Burroughs Corporation.

A global memory module (GMM) has 4 requestor ports to which B6800 processors may be attached. Consider the following diagram of a single GMM:

```
-----------
|         |
|   GMM   |
| A B C D |
-----------
  | | | |    requestor ports
```

The requestor ports are named A, B, C and D for purpose of addressing any B6800 which may be attached to them.

A B6800 installation may consist of one to four B6800 processors. In order to couple more than one processor, a global memory module (GMM) is required. The processors are simply connected to the requestor ports of the GMM (see D-note D2861). For example, a two-processor shared resources system may attach B6800 processors to ports A and B as in the following diagram:

```
                -----------
               |           |
               |    GMM    |
               |  A B C D  |
                -----------
                   | | | |
                   | |
         --------    --------
         |                  |
      ---------          ---------
     |         |        |         |
     |  B6800  |        |  B6800  |
     |         |        |         |
      ---------          ---------
```

Ports C and D remain unused at this installation.

At any given time, the installation manager may choose to have a set of processors work together in a multiprocessor environment (tightly-coupled), or work independently in separate working groups (loosely-coupled). The shared resources system provides the manager with a method of defining working groups of resources, reconfiguring the installation resources into a desired set of working groups, and dynamically redistributing the resources among the working groups.

Example 1

Assume that an installation consists of three B6800 processors, a GMM, and a complement of peripheral devices. The three processors are attached to GMM ports A, B, and C as in the following diagram:

```
                     -----------
                    |           |
                    |    GMM    |
                    |  A B C D  |
                     -----------
                       | | | |
                       | | |
       --------------  |  -------------
       |               |               |
   ---------       ---------       ---------
  |         |     |         |     |         |
  |  B6800  |     |  B6800  |     |  B6800  |
  |         |     |         |     |         |
   ---------       ---------       ---------
```

Port D is unused.

The installation has five choices of how the three processors may be distributed among working groups:

```
     A. Three one-processor groups.
            (A, B, and C loosely-coupled)
     B. One three-processor group.
            (A, B, and C tightly-coupled)
     C. One two-processor group and a one-processor group.
            (A and B tightly-coupled, loosely-coupled to C)
            (A and C tightly-coupled, loosely-coupled to B)
            (B and C tightly-coupled, loosely-coupled to A)
```

Note that any of the processors may be omitted (i.e not used).

Characteristics of a Group

A group is a set of hardware resources working together as one system. Each group has its own operating system and is designed to keep running no matter what happens to any other group. A group may consist of only one processor or as many as four processors tightly-coupled. A group can Halt/Load, CM, dump, power off, etc. with no effect on other groups. Each group looks somewhat like a separate computer system. HALT and CLEAR of any processor will propagate to any other processor in the same group but will not propagate to processors in other groups.

To some extent, the Global System Control (GSC) manages the requestor ports. All ports which are attached to processors in the same group will be given the same system ID within the GSC controls. The system ID enables the GSC to control the extent to which a processor can utilize other processors and global memory modules. No other group may acquire a processor port which has a valid ID. The HALT and CLEAR function buttons of the processor will clear the port IDs within the GSC. HALT and CLEAR are also propagated to all processors at ports with processors of the same ID. The global system ID is displayed via the "SC" ODT command whenever a processor is on-line to the Global System Control (GSC).

Example 2

Assume that the installation is the same as in example 1.

```
                   -----------
                  |           |
                  |    GMM    |
                  |  A B C D  |
                   -----------
                     | | | |
                     | | |
   ----------------- |  ----------------
   |                 |                 |
---------        ---------         ---------
|       |        |       |         |       |
| B6800 |        | B6800 |         | B6800 |
| BLUE  |        | BLUE  |         |  RED  |
---------        ---------         ---------
```

Ports A and B are tightly-coupled as the two-processor BLUE group. Port C is the one-processor RED group. No matter what happens to one of the two groups, the other should not be affected. The RED group need not even run on the same MCP as the BLUE group. RED and BLUE are autonomous.

If either processor in the BLUE group is HALTed or CLEARed, the other processor in the group will also HALT or CLEAR. However, HALT or CLEAR of the BLUE group will not affect RED.

Global Memory Assignment

It is obvious from the diagrams of global memory systems that the GMM is common to **all** processors and therefore must be common to all groups. Whenever a group is initialized, it may be assigned some amount of the global memory which it can control.

Whenever a tightly-coupled group is created, it must have access to some amount of global memory since it must necessarily have a common memory area. Furthermore, no other group will be allowed access to that area. This global memory area is termed PRIVATE global memory. Without it, processors cannot be tightly-coupled.

During initialization, the MCP will attempt to acquire its private memory allocation from the set of installation resources. The acquisition will be successful if (a) the memory physically exists, and (b) no other group has previously acquired the memory.

The GMM contains access switches which can be set by the processors. Once set, a processor may access the memory controlled by the switch, and all other processors will be denied access. There is, however, provision to share the memory with other processors on a read-only or read/write basis.

Two or more groups which are loosely-coupled may communicate via Host Services provided each group allows each of the other groups read access to some of the global memory it controls. This memory is termed **SHAREREAD** memory and is necessary for Host Services communication between groups. The group to which the **SHAREREAD** memory is assigned will have read/write access. That group will grant read-only access to any group to which it wishes to send a message. No other group is allowed to write into its memory space.

Example 3

Assume an installation as in example 2:

```
                 -----------
                |           |
                |    GMM    |
                |  A B C D  |
                 -----------
                   | | | |
                   | | |
   ---------------- | ---------------
  |                 |                |
---------       ---------       ---------
|       |       |       |       |       |
| B6800 |       | B6800 |       | B6800 |
| BLUE  |       | BLUE  |       |  RED  |
---------       ---------       ---------
```

Each B6800 has local memory (see D-note D2861). In this example, assume that each local memory has 32 16K memory modules. Therefore, mods 0-31 are local to each processor. Also assume that there are 32 16K modules of global memory. An attempt by any processor to reference mods 32-63 will be a global memory request.

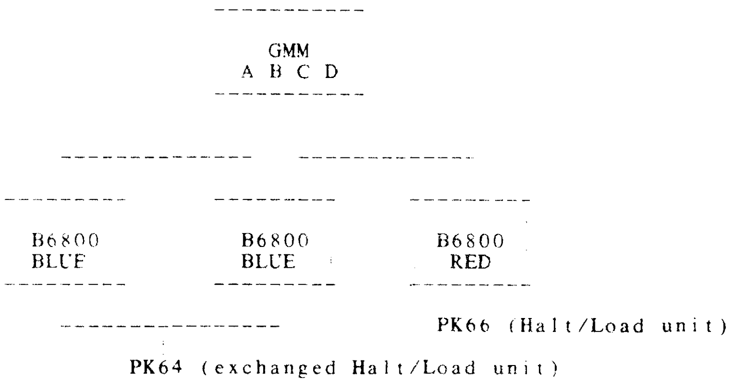
The BLUE group must have some PRIVATE global memory in order for the two processors to tightly couple. In this example, the modules 32-59 will be allocated to the BLUE group as PRIVATE memory. Modules 60-61 will also be assigned to the BLUE group as SHAREREAD memory (used to send messages to RED). Modules 62-63 will be assigned to the RED group as SHAREREAD memory (used to send messages to BLUE). The RED group requires no PRIVATE global memory as it is a one-processor group. However, PRIVATE memory could very well have been assigned to the RED group and it would have simply been used as an extension of RED's local memory. Since both groups have allocated SHAREREAD memory, they are capable of communicating via the Host Services mechanism.

Peripheral Unit Assignment

The GSC does not manage peripheral units. Each group must poll every other group to determine if a peripheral unit may be acquired. A group being polled will reject the request if the unit is in use. Units which are physically exchanged among groups must be disallowed to all but one group. Note that two processors which Halt/Load from the same unit may not belong to separate groups.

Example 4

Assume the installation of example 3:



The two processors of the BLUE group share the same Halt/Load unit via a peripheral exchange. Since they are managed by the same operating system, it is OK. However, RED must have its own Halt/Load unit because no tables are shared between the two autonomous systems. There is no reason to disallow a physical exchange of PK64 or PK66 among all three processors; however, simultaneous use of the same peripheral units by the two groups must be logically disallowed.

If, by chance, PK66 were already assigned to the BLUE group and the RED group were subsequently loaded, the RED group would poll the BLUE group and BLUE would reply that access to PK66 is denied. RED would then inform the operator:

GROUP DOES NOT OWN HALT/LOAD UNIT-OK TO USE PK66

The system operator then must resolve the problem by forcing BLUE to relinquish control of PK66. He may respond OK to RED in order to allow system initialization to continue.

The same scheme is true of non-Halt/Load units. However, since those units are not critical to initialization, the RED group will simply initialize without them. The operator may later have the RED group acquire the peripherals.

Configuration File

The installation manager must decide how the set of installation resources is to be divided into groups. That task is accomplished by building a CONFIGURATION FILE which contains an initial description of all groups. There is a symbolic form of the file (created by the user) and an object form (created by giving the symbolic form as input to a Burroughs supplied utility program). The utility program is called SYSTEM/CONFIGURATION which is a new III.1 release item.

The symbolic form of the CONFIGURATION FILE is very easy to generate. It may be a card deck, a CANDE sequenced data file, etc. It allows groups and entire installation configurations to be defined. The symbolic file is a series of initial group definitions and optional installation definitions. The syntax of the statements is as follows:

GROUP Statement

```
-- BEGIN -- GROUP --<group id>--<processor section>----------->

>-------------------------------------------------------------->
   |                         | |                      |
   |-<global memory section>-| |-<operations section>-|

>-------------------------- END GROUP -- ; -------------------|
   |                      |
   |-<peripheral section>-|
```

<group id>

```
--<letter>----------------------------------------------------|
           |                         |
           | |<--------------------| |
           | |                     | |
           |--- /16\ ---<letter>-----|
                      |          |
                      |-<digit>--|
```

<processor section>

```
-- PROCESSORS: ----------------------------------------------->

  |<-/3\------------------------------------------|
  |                                               |
>--- PROC --<node id>-------------------------- ; -----------|
                      |                       |
                      |- LOCALIZE <mem list> -|
```

<node id>

```
---- A -------------------------------------------------------|
  |     |
  |- B -|
  |     |
  |- C -|
  |     |
  |- D -|
```

<operations section>

```
-- OPERATIONS: -- <ODT commands> -----------------------------|
```

<global memory section>

```
            |<----------------------------------|
            |                                   |
-- MEMORY: ----- PRIVATE -----<mem list>-- ; ----------------|
              |               |
              |- SHAREREAD -|
```

<peripheral section>

```
-- PERIPHERALS: --<unit list>-- ; ---------------------------|
```

<mem list>

```
-------------------------------------------------------------|
  |                      |
  |- MODS --<mod range>-|
<mod range>

   |<------------------ , ------------------|
   |                                        |
----<module number>--------------------------------------------|
                          |                 |
                          |- - <module number> -|


<unit list>

   |<---------------- , ----------------|
   |                                    |
----<unit number>----------------------------------------------|
                        |               |
                        |- - --<unit number>-|
```

Semantics:

The GROUP statement is used to define autonomous subsystems. It allows a site to define hardware resources which will be dedicated to that particular group. Those resources include processors, global memory modules, and peripheral units. It also assigns to the group a name (group id) which is displayed in response to "WM" and "GC" ODT commands.

The <processor section> consists of a list of not more than four processor nodes that run under control of the same operating system. If there is only one node in the list, the group is monolithic (see D-note D2861). During initialization, acquisition of all specified nodes is attempted. If the node is unused or has already been assigned a family name via the Global System Control (GSC), the acquisition will fail and the group will initialize without the node. Any nodes which cannot be incorporated at group initialization time may later be brought into the group by making the node available and Halt/Loading the group.

The LOCALIZE command following a processor node designates that the memory is to be treated as local to that processor. That memory must have lower addresses than any other global memory designated to the group.

ODT commands may be entered under the <operations section>. Currently, only the ODT DL command is allowed. See GENERAL note D2535 for DL syntax and semantics. Commands specified in this section will be performed during the initialization process of the group.

The <peripheral section> consists of a list of peripherals to which communications paths may be established. During system initialization, each group will poll all other active groups as to whether it is appropriate to use the peripherals assigned to it. Any peripheral which is assigned to another group will be deleted from the polling group's description. The processors of a group will not attempt to use any peripheral unit that is not included in the list even though physical paths may exist to those units. If the section is not specified, the group will establish I/O paths only to non-exchangeable units and units designated as Halt/Load units. Such Halt/Load units will not be written upon until the system operator has notified the MCP that it is OK to do so.

The <global memory section> consists of a list of global memory which the group will manage. That memory may be specified as PRIVATE or SHAREREAD.

The PRIVATE modifier specifies exclusive use of the global memory. If two or more processors are in the group, the PRIVATE memory will be used for tightly coupling the processors. If the group is monolithic, any private global memory which is specified will simply be dedicated to the monolithic processor as an extension to local memory.

The SHAREREAD modifier specifies memory which the group may share with other groups for the purpose of loosely-coupled global memory communications. The shared memory must be visible to any groups that attempt such communications with the group.

If the <global memory section> is not specified, the group will not attempt access to any global memory. Lack of shared memory prohibits the group from establishing loosely-coupled global memory communications with other groups. Lack of private memory prohibits tightly-coupled processor configurations.

INSTALLATION Statement

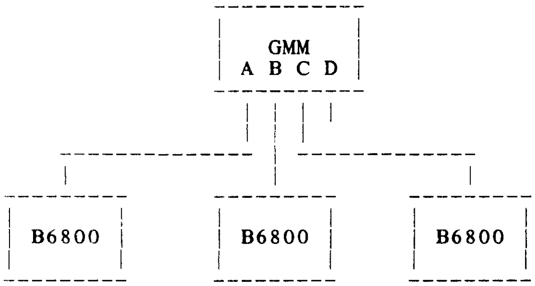
```
-- BEGIN -- INSTALLATION --<installation id>-- GROUPS -- : --->

>-<group list>-- END -- INSTALLATION -- ; -------------------|
<installation id>

--<letter>------------------------------------------------|
          |                        |
          | |<-------------------| |
          | |                    | |
          |--- /16\ ---<letter>-----|
                     |         |
                     |-<digit>--|


<group list>

  |<---- , ----|
  |            |
----<group id>--------------------------------------------|
```

Semantics:

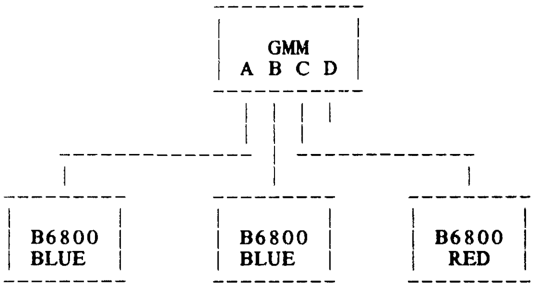
INSTALLATION statements may also be included in a configuration file. The purpose is merely to allow total installations to be defined within the configuration file. Any number of installations may be defined. INSTALLATION statements are optional and, as will be shown later, are merely a convenience mechanism.

Example 5

Assume the installation resources of example 1:



This installation has been running in the default mode (i.e it has been running as a 3-processor tightly-coupled installation) because no groups have ever been specified. Now, the installation manager wishes to assume the configuration of example 3:



Assume also that processors A and B Halt/Load from PK64 and processor C Halt/Loads from PK66. There are three other packs PK68-70 which are exchanged to all three processors. The installation has 5 tape drives, MT81-85 which are exchanged to all three processors. There are three ODTs, SC44, SC60 and SC61. SC44 is attached to processor A, SC60 is attached to processor B, and SC61 is attached to processor C. There are two line printers. LP13 is attached to processor A, LP14 is attached to processor C, and there is no line printer attached to processor B.

In addition, the installation manager would like to define a 3-processor tightly-coupled group which controls all the resources. He would like to run on the BLUE-RED configuration during the day, then reconfigure to the 3-processor tightly-coupled configuration at night. The symbolic configuration file may look like the following:

```
 BEGIN GROUP BLUE
   PROCESSORS:
     PROC A;
     PROC B;
   MEMORY:
     PRIVATE MODS 32-59;
     SHAREREAD MODS 60-61;
   PERIPHERALS:
     UNITS 13,44,60,64,68,69;
  END GROUP; % BLUE
  BEGIN GROUP RED
    PROCESSORS:
      PROC C;
    MEMORY:
      SHAREREAD MODS 62-63;
    PERIPHERALS:
      UNITS 14,61,66,70,81-85;
  END GROUP; % RED
  BEGIN GROUP THREEBY
    PROCESSORS:
      PROC A;
      PROC B;
      PROC C;
    MEMORY:
      PRIVATE MODS 32-63;
    PERIPHERALS:
      UNITS 1-255;
  END GROUP; % THREEBY
  BEGIN INSTALLATION DAYTIME
    GROUPS: BLUE,RED;
  END INSTALLATION;
  BEGIN INSTALLATION NIGHTTIME
```

```
    GROUPS: THREEBY;
  END INSTALLATION;
```

The installation now has specified all the groups it will need.

Creation of the Configuration File Object

After having created the symbolic CONFIGURATION FILE, the SYSTEM/CONFIGURATION program is run to create the object file which the system will actually use (this is similar to compiling a symbolic file to create an object code file which is executed). The input file to SYSTEM/CONFIGURATION has an internal name of "SOURCE", the output file has an internal name of "OBJECT". The file titles of these files are defaulted to "SYMBOL/CONFIGURATION" and "SYSTEM/CONFIGURATION".

Example 6

Assume the symbolic file created in example 5 is called "SYMCONFIGURATION". The object file is to be called "SYSCONFIGURATION". A job to create the object file may look as follows:

```
 ?RUN SYSTEM/CONFIGURATION;
 FILE SOURCE(TITLE = SYMCONFIGURATION);
 FILE OBJECT(TITLE = SYSCONFIGURATION);
```

The MCP must now be informed of the name of the configuration file. The "CF" ODT command is provided for that purpose.

CONFIGURATION FILE (CF) Command

```
-- CF ------------------|
        |               |
        |- + ---------|
        |               |
        |- - ----------|
        |               |
        |-<file name>-|
```

Semantics:

The CF command is used to designate or display the title of the current configuration file.

CF with no options displays the title of the current configuration file.

CF+ selects SYSTEM/CONFIGURATION, the default, as the configuration file.

CF- undesignates the configuration file.

CF <file name> selects the named file as the configuration file.

Examle 7

After having compiled the object configuration file of example 6, the following could be entered via ODT to designate "SYSCONFIGURATION" as the configuration file:

```
 CF SYSCONFIGURATION
```

The CF command does nothing to the configuration file. It merely stores the file name into the system tables so that any reconfiguration will know where to find a description of the groups. It need only be done if a reconfiguration is initiated from a group. If the configuration file were never specified or the file did not exist, CONFIGURATOR, the MCP procedure which does the reconfiguration, would generate a "no file" condition.

Use of the Configuration File

The CONFIGURATION file is only used when a RECONFIGURATION is initiated. It must contain a description of each group which is to be in the new configuration. During the RECONFIGURATION, the new group descriptions are extracted from the file and stored in the system information tables of each new group. A group is not required to have a configuration file unless it is chosen to initiate a RECONFIGURATION. The file is not used until another RECONFIGURATION is initiated.

In order to maintain a group's integrity, no group has access to any other group's current system information tables. Via the CONFIGURATION file, only the initial configuration of any other group is available at RECONFIGURATION time. Subsequent FREEs or ACQUIREs may have altered the group's current tables, but the group's description in the configuration file is not disturbed. Thus any FREEs and ACQUIREs done previous to a RECONFIGURATION are lost.

RECONFIGURE Statement

```
-- RECONFIGURE --- INSTALLATION --- AS ---<group list>-------|
                |                |      |                 |
                |- GROUP --------|      |- DEFAULT -------|
                |                                         |
                |- INSTALLATION -- AS --<installation id>-|
```

Semantics:

The RECONFIGURE command regroups the hardware resources according to the groups specified in the group list. If INSTALLATION is specified as the current configuration, all installation resources are reconfigured to the new configuration. If GROUP is specified as the current configuration, all resources assigned to the group will be reconfigured into the new configuration. A reconfiguration may cause a Halt/Load of existing groups which are being reconfigured into new groups.

Any subset of the <group id>s defined in the configuration file may be specified in the new configuration. Not all defined <group id>s need be specified and a <group id> cannot be specified more than once.

Example 8

After having specified the configuration file via the "CF" ODT command, the installation may be reconfigured. In order to go to the BLUE-RED configuration, the operator could enter:

```
     RECONFIGURE INSTALLATION AS BLUE,RED
                    or
     RECONFIGURE INSTALLATION AS DAYTIME
```

The second form is convenient as it allows the creator of the configuration file to designate an entire configuration by one name.

At night, when the installation is ready to reconfigure to the three-processor tightly-coupled configuration, the operator can enter:

```
     RECONFIGURE INSTALLATION AS THREEBY
                    or
     RECONFIGURE INSTALLATION AS NIGHTTIME
```

Default System Configurations

The configuration file should be used by any site that wants sophisticated system configurations. However, some sites want only to Halt/Load their machines and have the system automatically configured. The default GROUP will consist of:

1. All processors directly attached to the GMM to which the lead Halt/Loading processor is attached.

2. All peripherals (1-255).

3. All memory within the GMM to which the Halt/Loading processor is attached.

At any time, the operator may enter:

```
     RECONFIGURE INSTALLATION AS DEFAULT
                   or
     RECONFIGURE GROUP AS DEFAULT
```

DEFAULT is system-defined and requires no configuration file.

Modification of Existing Groups

FREE and ACQUIRE Statements

```
---- FREE -------- SHAREMODS -----<mod range>----|
  |             |  |             |               |
  |- ACQUIRE -| |  |- PRIVATEMODS -|             |
                   |                             |
                   |- PROC --<node id>-----------|
                   |                             |
                   |-<unit type>--<unit number>--|
```

Semantics:

Existing groups can be altered by the FREE and ACQUIRE commands.

The FREE command is used to detach resources from an active group.

The ACQUIRE command allows an active group to acquire additional resources.

If the resouces requested are assigned to another active group, a dialog will be established between the two groups such that the other group will be asked to relinquish control of the resource.

Any change permanently alters a group; i.e., the results of alterations are maintained across Halt/Loads but not RECONFIGURATIONs. The original group definition, whether obtained from the CONFIGURATION FILE or through default configuration, specifies the initial definition of a group. Though a modification alters the current definition of a group, it does not alter the group definition in a CONFIGURATION FILE.

Group Configuration (GC) Command

```
-- GC --|
```

Semantics:

The GC command displays the current configuration.

D2905 GENERAL - "LH" COMMAND

The new disk pack subsystems should be loaded from the host via the LH ODT command, rather than from cassettes.

As a reminder, the LH syntax is as follows:

LH

```
-- LH -- PK --<unit number>-- MPX --<mpxno>-- PATH --<pathno>--------->

>--------------------------------------------------------------------|
   |             |
   |-<filetitle>-|
```

LH (Load Host) loads the pack firmware file to the disk pack controller. The default firmware file is SYSTEM/FIRMWARE on DISK OTHERWISE PACK. Blanks are required between MPX and <mpxno> and between PATH and <pathno>.

D2906 GENERAL - CHANGES IN THE "SWAPPER" MECHANISM

The following system note references various other documentation, as follows:

GENERAL ODT III.1 note D2535 - "ODT Note"
MCP II.9 Note D2146 - "MCP Express Note"
MCP III.1 Note D2654 - "MCP Pointers Note"

NEW FEATURES

New parameter - "EXPTIME"

The size of a time-slice for an express swapjob is now a SWAPPER parameter that can be set through the SW ODT message (see "ODT Note" for syntax information). Previously, this parameter existed but was not settable by the installation. The default value of this parameter is .75 seconds which is the value used on the III.0 and II.9 MCP releases (see "MCP Express Note" for more information on the EXPRESS facility of SWAPPER).

Timelimit vs Datacom input

The TIMELIMIT option of the READ statement (in ALGOL) now functions properly for swaptasks.

Inter-Program Communication (IPC) for Swaptasks

The SWAPPER mechanism has been redesigned to handle most forms of IPC. This includes co-routines and asynchronous processes. In general any tasking structure can be handled. The principal rule governing swappable IPC families is:

> If the critical block for a process is in a swappable stack, then that process must also be swapped.

Any son of a swaptask must be a swaptask. For example, if a batch process "A" spawns a swappable process "B" and that process subsequently spawns another process "C", then if the critical block for "C" is in "B", "C" is required to be swapped (any user specification to the contrary is overridden). However, if the critical block was in "A", then "C" is not required to be swapped, nor is it excluded from being so.

Restrictions on IPC Families

1. Bad Pointers — See "MCP Pointers Note"

2. LIBRARYs — Library stacks are not permitted in swapspace. Any program running in swapspace that attempts to FREEZE will be DSed; however, programs which use libraries are permitted in swapspace (only the library stacks themselves are excluded).

3. DATABASEs — No major changes have been made in this area. DATABASE stacks are still

not permitted in swapspace but the programs which use DATABASE stacks are. Some problems that previously existed in this area have been corrected.

4. JOBs — Job stacks are not permitted in swapspace. However internal processes of jobs can be swapped.

Getting IPC-capable Programs to Run in Swapspace

On III.0 and previous releases of the MCP, code files which were marked IPC-capable were not permitted in swapspace even if the SUBPSACES task attribute requested swapping. On III.1 this rule has been changed. Any codefile compiled by a III.0 or III.1 compiler will run in swapspace if the SUBSPACES attribute is non-zero. Any codefile compiled by a II.9 or older compiler that is IPC-capable will not run in swapspace unless the task OPTION word has the IPCOVERRIDE bit set (OPTIONS=IPCOVERRIDE in WFL or CANDE). IPC-capable codefiles compiled II.9 or older without the IPCOVERRIDE bit being set will continue to behave exactly as in III.0 (with regard to swapping).

New SWAPPER Priority Parameters

There are two new SWAPPER priority algorithm parameters. These parameters are set using the SW ODT message (see "ODT Note" for more information on syntax). The new formula for calculating the SWAPPER queue priority is:

```
 MIN(255,MAX(0,<old formula>-
      (  (IOBIAS*<num I/Os>) +
         (MEMORYBIAS*(<memory size> / MAXCORE))
      ) ))
```

Where:

<old formula> The value of the SWAPPER priority algorithm on the III.0 MCP release.

IOBIAS — The larger the value of this parameter, the lower the priority of multiprocess IPC families.

<num I/Os> — The number of I/O operations required to bring in all of the swappable parents of the task whose priority is being calculated.

MEMORYBIAS — The larger the value of this parameter, the lower the priority of processes with large memory requirements.

<memory size> The amount of memory required (in slots) by all of the swappable parents of the task whose priority is being calculated.

MAXCORE — The value of the SWAPPER MAXCORE parameter.

Usage of the Express area in an IPC family

Any time that SWAPPER brings in a process that was not "sliced", it will try and bring it into the express area, provided that the process meets the express criteria (size < EXPMAXCORE and not sliced). This facility has been modified to handle IPC families by using the following rule:

> Only the leaf of an IPC family may be placed in the express area.

Any process with no swappable sons is eligible to run in express space. This includes all old programs that were swappable.

SWAPPER on a B6800 Multiprocessor System

The SWAPPER mechanism has been extended for B6800 multiprocessor systems. Now a SWAPPER independent runner and an associated swapspace may be initiated in each memory subsystem (except Global). Each of these swappers is independent in the sense that they each have their own set of parameters and can start and stop independently of each other. For example, it would be easy to run express swaptasks in one processor and most of the non-express swaptasks in the another processor (by setting the EXPRESERVE parameter to CORESIZE-MAXCORE for the SWAPPER in the first processor and setting EXPRESERVE to zero on the other). For details on the syntax of the SW ODT message, see "ODT Note".

*SYSTEM/SWAPDISK for a Multiprocessor System

The *SYSTEM/SWAPDISK file used by the III.1 SWAPPER is compatible with the file used on the III.0 and previous releases, and vice versa; however, for B6800 multiprocessor systems, an extra restriction is added:

> Each row of *SYSTEM/SWAPDISK must reside on a unit that is exchanged to each processor in which SWAPPER is to be run.

The SUBSYSTEM Task Attribute

The SUBSYSTEM task attribute is used to specify a subset of the memory subsystems where it is desirable to run this task. For batch tasks, one of the subsystems is choosen from the specified set and then permanently placed in that subsystem (no capability for moving batch tasks between different subsystems exists). For swaptasks, the meaning of this attribute is not changed but the capability of moving the task is readily available. (The task is swapped out of one subsystem and into another. It is this capability that requires *SYSTEM/SWAPDISK to be exchanged to all processors running SWAPPER.) If a task requests to be run in swapspace (task.SUBSPACES neq 0), this task will be run in any of the swapspaces that are in the subset of memory subsystems specified by the SUBSYSTEM attribute.

Example:

Task "A" is started with SUBSPACES=3 and SUBSYSTEM=BOTH, where BOTH is setup to be processors 1 and 3. At task initiation time there is a SWAPPER only in processor 3. This task would be run as a swaptask constrained to processor 3. (Even if there were a SWAPPER in processor 2, "A" would never run there.) If a SWAPPER were subsequently started in processor 1, task "A" could now be either processor 1 or 3. It is even possible to then shut down the SWAPPER in processor 3, then "A" would only run in processor 1.

SUBSYSTEM vs. SUBSPACES: Conflicting Specifications

When a conflict arises between the specifications of the SUBSYSTEM and SUBSPACES task attributes, the SUBSYSTEM specification is considered more binding than SUBSPACES. In case of conflict, the SUBSPACES attribute will be modified in order to make it consistent. Some examples of conflicts and their resolution are:

1. SUBSPACES NEQ 0 but no SWAPPER running within the specified subsystem. In this case remedial action is to assume SUBSPACES=0.

2. SUBSYSTEM specifies more than 1 processor and SUBSPACES=1. The inconsistency in this situation is that the user has requested that the segment dictionary be placed in main memory but has said that the data can run in more than one processor. In this case SUBSPACES=3 is assumed and a warning message is logged for the task. This situation can also occur if SUBSPACES=2 is in effect.

Process Family Migration on a B6800 Multiprocessor System

In the same fashion that a single swaptask may migrate between different swapspaces (on different processors), an IPC family of swaptasks may also migrate. As with a single task, migration is controlled by the SUBSYSTEM task attribute. However the full generality of the subsystem specification mechanism is not supported. The following restriction is made:

> Any task spawned by a swappable task will run in the same set of subsystems in which the parent will run.

A son of a swappable task inherits the subsystem specification of the parent regardless of any user specification which might be present.

Swaptasks and Non-exchanged Units

On a B6800 multiprocessor system, whenever a task attempts to use a peripheral which is not exchanged to the processor running the task, then the buffer for that I/O operation is moved into global memory and the I/O operation is performed by one of the processors to which the unit is exchanged. For swappable tasks the buffer is moved into global memory if the peripheral is not exchanged to every processor in which this task might run (processors in which SWAPPER is not currently running are also included in this test because SWAPPER might be started in that processor at some point in the future). In order to reduce the amount of global memory used, any swaptasks which use non-exchanged units should have a subsystem specified which includes only those processors to which the peripherals are exchanged.

CHANGED FEATURES

Changes to the CONTROLLER "Y" Display

The text of the "SWAP STATUS" line of the Controller "WY/Y" display has been changed to reflect the new implementation of SWAPPER. The text of the new displays and their meaning is described below.

TEXT	MEANING
NEW	This task is new, it has just been initiated.
IN CORE	This task is currently in core and there is no outstanding request to swap it out.
QUIESCING	This task is currently in core and there is an outstanding request to swap

it out; however, the task is not currently in a state that would permit it to be swapped. There are 3 principal reasons that would prevent the task from being swapped:
1. The task has an I/O operation in progress.
2. A son of this task is still in core.
3. The task is executing certain sections of MCP code.

WRITE REQUEST This task has requested that it be swapped out and it is currently possible to honor that request; however, SWAPPER has not yet started to process the swap-out request.

WRITING OUT This task is currently in the process of being written to *SYSTEM/SWAPDISK.

ON DISK This task is resident on *SYSTEM/SWAPDISK, no outstanding request to swap it in exists.

READ REQUEST A request to swap this task into main memory has been made but SWAPPER has not yet processed this request.

WAITING FOR CORE A request to swap this task in has been processed, but insufficient memory in a swapspace is available to honor this request. Tasks forced out for exceeding their timeslice will also appear in this state.

READING IN This task is currently in the process of being read from *SYSTEM/SWAPDISK.

TERMINATING SWAPPER is currently in the process of terminating this task.

OUT OF DISK A request to swap this task out has been made; however, currently there is insufficient room on *SYSTEM/SWAPDISK to honor that request.

NEEDS PARENT This task is in core but is unable to run because one or more of its ancestors is not yet present.

LINGERING Certain statements (primarily those associated with IPC) leave a task in an inactive state that may last for only a short period of time. In these cases the task is left in a state such that if the expected event does not occur within 2 seconds the task is swapped out. Tasks in that state are said to be lingering.

CONSIDERING FOR SWAP IN
A task in this state is currently under consideration for swap in. If the resources are available, then this task will be swapped in; if not, then it goes back to the WAITING FOR CORE state.

ZIPS from Swaptasks

Formerly when a swaptask executed a ZIP statement, the WFL compiler was started in batch as a co-routine of the swaptask but the swaptask remained locked in core until WFL terminated (which could be a long time if WFL were suspended). Now when a swaptask executes a ZIP statement, the WFL compiler is run in swapspace as the son of the ZIPing task. This allows WFL and the ZIPing task to be timesliced or swapped out for any other reason. The differences in this mechanism represent no net change in the semantics of the ZIP construct.

ABORTS from Swaptasks

The Data Management abort routine is run by the user of a data base to exit transaction state after a user already in transaction state terminates abnormally. Previously if the task that initiated abort was swappable, then abort was run as a batch task and the initiating task was locked into core until abort terminated. Now when a swappable task initiates abort, abort will run as a swappable task.

Stackstretching for Swappable Tasks

Previously when a swaptask needed to have its stack stretched, it was swapped out and an extra "slot" was added to the subspace. One effect of this was that swaptasks were stackstretched by only 990 words at a time; another was that if the subspace were already of maximum size, the stack could not be stretched (even if there were plenty of room). Now stackstretching for swaptasks is done by the same mechanism as stackstretching for batch tasks; the subspace may grow if necessary.

Changes in the Internal Structure of a Subspace

Many changes have been made in the internal structure of some types of data areas in a subspace. This was done because the old methods had some built-in restrictions that could cause certain pathological programs to be erroneously DSed for "MEMORY EXCEEDED". A brief survey of these changes:

1. The processing stack in a subspace is no longer constrained to be the first data area of a subspace, this is a side effect of the changes in the area of stackstretching.

2. FIBs and SIBs are no longer required to be in the first 65K words of a subspace.

Changes in the Swapping Mechanism

There are two principal mechanisms for getting a swappable task swapped out. These are called "delayswap" and "immediateswap".

Immediateswap is used whenever a task is to swapped out as soon as possible. This mechanism is used when a task is to be swapped out for exceeding its time slice, waiting for datacom input (or output), waiting for a reply or other situations where the MCP has determined that it could be a long time before this task resumes.

Delayswap is used when the amount of time before this task resumes is likely to be small (i.e., a CONTINUE statement in ALGOL). If the delayswap mechanism is specified for a task, then that task is allowed to "linger" in core for 2 seconds. If at that time it has not resumed, then a normal swapout using the immediateswap mechanism is used. Some examples where delayswap is used are:

WAIT statements (from user code).
ALGOL-type CALL statements.
ALGOL-type CONTINUE statements.

In conjunction with allowing IPC-capable programs to run in swapspace, many new points at which a task may be swapped have been added. A list of these points and the type of swapping mechanism to be used follows:

1. Any point in user code	Immediateswap (time-sliced only)
2. Waiting for a reply	Immediateswap
3. ALGOL-type CALL statement	Delayswap
4. ALGOL-type CONTINUE statement	Delayswap
5. WAIT statement	Delayswap
6. WHEN(<time>)	Not swapped if <time> is less than 2 seconds, otherwise Immediateswap
7. At points where extra storage may be needed.	Immediateswap

This list is not meant to be complete; it only provides the user with some understanding of the mechanisms involved.

DEIMPLEMENTED FEATURES

SWAPPER AX Facility

SWAPPER parameters are no longer set using the AX input message. Refer to "ODT Note" for syntax information for the SW ODT message which replaces this facility.

Auto-Restart Enhancements

After a Halt/Load, if SWAPPER is to be restarted, the system will not complete initialization and restart user jobs until SWAPPER is completely initialized. If SWAPPER initialization is delayed, the operator can force initialization to continue by responding to the RSVP that will be outstanding.

If the schedule is held (via EI or HS) and there are one or more swappers in the schedule, when the operator releases the schedule (via EI, HS-, etc.), the schedule will continue to be held until the swappers are initialized. (Again, if initialization is delayed, the operator can discontinue waiting for SWAPPER.)

D2913 GENERAL - "GLOBAL MEMORY" TRADEMARK

"GLOBAL Memory" is a trademark of Burroughs Corporation.

D2934 GENERAL - "SPO" REPLACED BY "ODT"

For documentation purposes, all occurrences of SPO (Supervisory Printer Output) have been replaced by ODT (Operator Display Terminal). The two terms were synonymous and interchangeable.

D2953 GENERAL - NEW "COMPILERINFO" FORMAT

The format of the COMPILERINFO version data has been changed to permit the CYCLE number to exceed 255.

COMPILERINFO is a word of data placed in a code file (segment 0, word 8) by the compiler, kept by the MCP, and used in various places to identify the compilation. The bottom 24 bits of the word contain VERSION data for the compiler that created the code file.

The old and new formats are shown here, with example values for a compiler version 31.234 (cycle 234 of III.1):

Old			New		
Bits	Contents	Example	Bits	Contents	Example
23:8	MARK	3	23:8	MARKLEVEL	31
15:8	LEVEL	1	15:6	unused	
7:8	CYCLE	234	9:10	CYCLE	234

In the initial release of III.1, the NEWP compiler uses the new format and all others continue to use the old one. On a subsequent update of the III.1 system, the other compilers will be changed.

The III.1 MCP translates the old format to the new whenever storing COMPILERINFO for a task, and outputs only the new format. This change appears when COMPILERINFO is returned by SYSTEMSTATUS or GETSTATUS.

The following software items have been modified to accept COMPILERINFO in the new format: CONTROLLER, DUMPANALYZER and JOBFORMATTER. (LOGANALYZER and LOGGER are updated by inclusion from JOBFORMATTER. Logs from systems prior to III.1 do not include COMPILERINFO in either format.)

D2979 GENERAL - PORTS AND SIGNALS

Tasks may now communicate with each other by means of the Inter-Process Communication facilities provided by Host Services.

See Appendix D, Inter-Process Communication, for details of implementation.

D2981 GENERAL - MARK LEVEL DOCUMENTATION

Beginning with the 3.2 system release, for documentation purposes, mark levels will appear in Arabic rather than Roman numerals; e.g., Mark 3.2 rather than Mark III.2, or Mark 2.7 rather than Mark II.7.

SOFTWARE IMPROVEMENTS NOTES (P NOTES)

GENERAL

P1414 GENERAL - COPYRIGHTS, VERSION LEVELS UPDATED

The copyright notices have been updated for 1978, 1979.

Version levels for all software products have been set to the current level.

P2219 GENERAL - "KIND=DISK" VS. "FAMILYNAME"

The use of default values for the FAMILYNAME attribute in system software has been eliminated by explicitly declaring KIND=PACK, FAMILYNAME="DISK." or KIND=PACK, FAMILYNAME="PACK." where only KIND=DISK or KIND=PACK respectively were declared previously. The symbolics have also been changed to use file attribute mnemonics instead of integers, and to take advantage of the attributes BACKUPKIND and UPDATEFILE.

P2578 GENERAL - FOUR DIGIT PATCH NUMBERS

The system now recognizes 4-digit patch numbers.

ALGOL

D2623 ALGOL - EQUATION DISALLOWED IN GLOBAL DECLARATIONS

The following additions should be made to the ALGOL Manual, Form No. 5001639:

Page 4-5 Under "RESTRICTIONS", add the following

"<array row equivalence> is not permitted in <global part>."

Page 4-7 Add "RESTRICTIONS"

"<equation list> is not permitted in <global part>."

Page 4-11 Add "RESTRICTIONS"

"<direct array equivalence> is not permitted in <global part>."

Page 4-43 Add "RESTRICTIONS"

"<equation list> is not permitted in <global part>."

Page 4-59 Add "RESTRICTION"

"<equation list> is not permitted in <global part>."

D2649 ALGOL - NEW INFORMATION IN THE LISTING TRAILER

The trailer printed at the end of a listing will now also contain "SOURCE TAPE:<source tape name>" if dollar option MERGE is set for the compile, and "NEW SYMBOLIC:<new symbolic name>" if dollar option NEW is set for the compile.

D2661 ALGOL - HEADER IMPROVEMENT

The header that the compiler places on listings has been modified to include the patch identification with the version. An example of the new format is as follows:

```
31.002.005
```

The 31 indicates the release level, the 002 indicates the cycle (of the entire system), and the 005 indicates the number of patches above 31.000.000 to the ALGOL compiler.

D2662 ALGOL - MANY INSTALLATION INTRINSICS

ALGOL can now handle more than 256 installation intrinsics.

D2664 ALGOL - "STACK" AND "CODE" LISTING IMPROVEMENTS

The STACK and CODE listing produced by the compiler has been improved to only print four characters of displacement in address couples, and to print the names of all intrinsics and MCP procedures which are referenced in a program in the stack listing rather than the words "SEPARATE INTRINSIC".

D2677 ALGOL - "FIRST" FUNCTION FOR STRINGS

```
<FIRST function>

-- FIRST -- ( --<string expression>-- ) --|
```

The FIRST function is a real-valued function which returns as its value the ordinal position in the EBCDIC, hex, or ASCII collating sequence of the first character in the <string expression>. The function will return an ordinal position in the EBCDIC collating sequence if the <string expression> is EBCDIC, an ordinal position in the hex collating sequence if the <string expression> is hex, and an ordinal position in the ASCII collating sequence if the <string expression> is ASCII. A fault will occur if the <string expression> is empty.

Examples:

```
FIRST("ABC")      returns 193   (=4"C1")
FIRST(7"NNXX")    returns 78    (=4"4E")
FIRST(4"F1F2")    returns 15    (=4"0F")
```

D2684 ALGOL - "COMPLEX" IMPLEMENTATION

INTRODUCTION

Complex variables allow for the storage and manipulation of complex values in a program. The interpretation of complex values is the usual mathematical one. The real and imaginary parts of complex values are always single precision reals.

The set of real values is a subset of the set of complex values. Arithmetic values may be assigned to complex variables. Complex values may not be assigned to arithmetic variables.

The syntactic items used in this description match those in the ALGOL Manual.

COMPLEX DECLARATION

All complex variables must be explicitly declared in a <complex declaration>.

<complex declaration>

```
--<local or own>-- COMPLEX --<identifier list>--|
```

Examples:

```
        COMPLEX C1, C2
        COMPLEX CURRENT, VOLTAGE, IMP
```

Semantics:

A <complex declaration> is used to declare <simple variable>'s of type complex.

COMPLEX ARRAY DECLARATION

The syntax for an array declaration is extended to allow complex arrays. Direct complex arrays are not allowed.

<complex array declaration>

```
--<long/own specification>-- COMPLEX -- ARRAY --<array list>--|
```

<complex array reference declaration>

```
-- COMPLEX -- ARRAY -- REFERENCE --<array reference list>--|
```

<complex value array declaration>

```
-- COMPLEX -- VALUE -- ARRAY --<value array list>--|
```

Examples:

```
        COMPLEX ARRAY A,B[-1:2], CURR[1:15]
        COMPLEX ARRAY C [0:2, 0:60]
        COMPLEX ARRAY REFERENCE CREF1[0], CREF2[0, 10, 10]
        COMPLEX VALUE ARRAY CVAL( COMPLEX(8,1.5), 1.7*COMPLEX(3,(3/7)))
```

COMPLEX PROCEDURE DECLARATION

The syntax for a procedure declaration is extended to allow for complex-valued procedures.

<complex procedure declaration>

```
-- COMPLEX -- PROCEDURE --<procedure heading>-- ; --<procedure body>--|
```

Examples:

```
        COMPLEX PROCEDURE IMP(R); REAL R; ...
        COMPLEX PROCEDURE P; ...
```

COMPLEX ASSIGNMENT
------- ----------

A complex variable is assigned a value via the <complex assignment statement>.

<complex assignment statement>

```
--<complex variable>-- := ---<complex expression>-----------|
                           |                             |
                           |-<complex update expression>-|
```

<complex variable>

```
--<variable>--|
```

COMPLEX EXPRESSIONS
------- -----------

The syntax of a <complex expression> is

<complex expression>

```
----<arithmetic expression>-------------|
   :                                  |
   |-<simple complex expression>------|
   :                                  |
   |-<conditional complex expression>-|
```

<simple complex expression>

```
  |<------ + ------|
  |    :      |    |
  |    |<- - -|    |
  |                |
----<complex term>----|
```

<complex term>

```
----<term>---------------------------------|
   :                                     |
   : <-<complex multiplying operator>-|  |
   |                                  |  |
   |---<complex factor>------------------|
```

<complex factor>

```
----<factor>------------------------------------|
                                             |
   |-<complex primary>-----------------------|
                       |                     |
                       | |<----------------| |
                       | |                 | |
                       |--- ** --<primary>---|
```

<complex primary>

```
----<complex variable>----------------|
  |                                |
  |-<complex assignment statement>-|
  |                                |
  |-<complex function designator>--|
  |                                |
  |- ( --<complex expression>-- ) -|
  |                                |
  |-<complex case expression>------|
```

<complex function designator>

```
--<complex function identifier>--<actual parameter part>--|
```

<complex function identifier>

```
----<procedure identifier>------|
  |                          |
  |-<complex intrinsic name>-|
```

<complex case expression>

```
                  |<--------- , ---------|
                  |                      |
--<case head>-- ( ---<complex expression>--- ) --|
```

<conditional complex expression>

```
--<if clause>--<complex expression>-- ELSE --<complex expression>--|
```

<complex multiplying operator>

```
---- * ----|
  |     |
  |- / -|
```

The interpretation of complex expressions is the usual mathematical one. An arithmetic (non-complex) expression is (whenever necessary) considered to be the real part of a complex expression with a zero imaginary part. There is no automatic type conversion from complex to arithmetic.

Valid complex intrinsic names are listed in the last paragraph of this document.

COMPLEX UPDATE EXPRESSIONS

The syntax of a <complex update expression> is:

<complex update expression>

```
-- * --<complex update tail>--|
```

<complex update tail>

```
----<complex update term>------------------------------------|
  |                        ||                              |
  |- + ---<complex term>---|| |<------------------------|  |
  |     |                   | |                         |  |
  |- - -|                   |----- + ---<complex term>---|
                                 |     |
                                 |- - -|
<complex update term>

----<complex update factor>---------------------------------------|
  |                          | |                              |
  |- * ---<complex factor>--| | |<------------------------| |
  |     |                    | | |                        | |
  |- / -|                    | |----- * ---<complex factor>---|
                                   |     |
                                   |- / -|


<complex update factor>

  |<----------------|
  |                 |
---- ** --<primary>----|
```

The interpretation of a <complex update expression> is the same as that of a <complex expression> after substitution of the * following the assignment operator by the complex variable that precedes the assignment operator.

COMPLEX PARAMETERS TO PROCEDURES
------- ---------- -- ----------

If a formal parameter of a procedure is a simple variable of type complex, any complex expression may be passed as an actual parameter. If the type of a formal parameter is not complex, the actual parameter may not be of type complex.

If the formal parameter is a complex name parameter and the actual parameter is not of type complex, an assignment within the procedure body to the formal parameter will cause the program to abort with an invalid operator.

If the formal (actual) parameter of an unspecified formal procedure is of type complex, the actual (formal) parameter must also be of type complex.

Complex arrays are allowed as parameters to procedures; in the parameter specification the lower bounds of these arrays must be specified with either an integer constant or an asterisk. If the formal (actual) parameter is a complex array, the actual (formal) parameter must also be a complex array.

Complex procedures are allowed as parameters to procedures; in the parameter specifications the parameters of these procedures may either be fully specified (with FORMAL) or be unspecified. If the formal (actual) parameter is a complex procedure, the actual (formal) parameter must also be a complex procedure.

COMPARISON OF COMPLEX VALUES
---------- -- ------- ------

In a relation, a complex operand may be compared with another complex operand or an arithmetic operand. If one of the operands of a relation is of type complex, the only allowed relational operators are EQL (=) and NEQ (=).

INPUT AND OUTPUT OF COMPLEX VALUES
----- --- ------ -- ------- ------

A complex variable or value occurring in the list part of all types of READ and WRITE statements is considered to be a pair of real variables or values; the first real of the pair being the real part, the second being the imaginary part. A complex array row occurring in the list part of a READ or WRITE statement is considered to be a real array row containing the real and imaginary parts of the elements of the complex array row in the following order: real part of first element, imaginary part of first element, real part of second element, etc.

BINDER INTERFACE
------ ---------

Procedures with complex parameters and complex procedures may be declared external and bound in. Complex variables and arrays may be specified in a <global part>.

The complex implementation in ALGOL is compatible with the complex implementation in FORTRAN so that interlanguage binding with complex types involved is allowed; the normal by-name, by-value and array-lower-bound rules are applicable.

COMPLEX INTRINSICS AND INTRINSICS WITH COMPLEX ARGUMENTS
------- ---------- --- ---------- ---- ------- ---------

Function	Parameters	Result
CABS	(<ce>)	Real. Absolute value of <ce>.
CCOS	(<ce>)	Complex. Cosine of <ce>.
CEXP	(<ce>)	Complex. e ** <ce>.
CLN	(<ce>)	Complex. Natural logarithm of <ce>.
COMPLEX	(<ae1>, <ae2>)	Complex. <ae1> + i * <ae2>.
CONJUGATE	(<ce>)	Complex. Conjugate of <ce>.
CSIN	(<ce>)	Complex. Sine of <ce>.
CSQRT	(<ce>)	Complex. Square root of <ce>.
IMAG	(<ce>)	Real. Imaginary part of <ce>.
REAL	(<ce>)	Real. Real part of <ce>.

D2711 ALGOL - DEIMPLEMENTATION OF BACKSLASH

The compiler will no longer accept the backslash (\) character in an <identifier> or <user option>. The backslash character will now be flagged with an error message.

Note: III.0 ALGOL note D2563 gave advance warning of this change.

D2712 ALGOL - FLAG VECTORMODE

Because vectormode is not portable to non-vectormode machines, the compiler will flag with a warning message the first occurrence of an explicit vectormode statement in a program, batch job or separately-compiled procedure. If the warning is given, a message will also appear in the listing trailer.

D2713 ALGOL - TOGGLE, OVERFLOW, "REAL(<POINTER EXP>)"

The functions TOGGLE and OVERFLOW have been deimplemented.

Use of the function REAL(<pointer expression>) will now be flagged with a warning message. The OFFSET function can be used to find the offset of a pointer within an array.

Example:

If P is an 8-bit pointer into a non-segmented array,

```
REAL(P).[35:16]*6+REAL(P).[39:4]
```

returns the same value as does

```
OFFSET(P)
```

Note: The OFFSET function is documented in 3.0 ALGOL note D2289.

D2721 ALGOL - STRING ARRAYS, PROCEDURES AS PARAMETERS

Passing string arrays and string procedures (unspecified as well as specified) as parameters has been implemented.

In the specification of string array parameters, the lower bounds must be integer constants or asterisks (*).

In the specification of formal string procedures, the parameter list must be unspecified or fully specified by the <formal parameter specifier>.

If the string type of the formal array or procedure is not EBCDIC, this type must also be specified.

D2761 ALGOL - "MAKEHOST" ENVIRONMENTS

The B7000/B6000 ALGOL Reference Manual (Form No. 5001639) should be corrected. On page D-22, under the compiler option MAKEHOST, remove the sentence:

> "The current implementation, for reasons of simplicity, requires environments to be fully qualified through level three procedure identifiers."

Replace it with the following sentence:

> "Environments must be fully qualified through the outermost level of procedure declaration, with the exception that for a program which is a procedure, the name of that procedure must not appear."

D2762 ALGOL - "VECTORMODE" LOOPS WITH LENGTH <=0

The following addition should be made to the B7000/B6000 ALGOL Reference Manual (Form No. 5001639) under "VECTORMODE Statement" on Page 5-113:

> "The hardware supports two modes of vectormode, single word and multi word; the compiler will attempt to use single word mode whenever possible. There is a noticeable difference between the two when the length specified by the FOR <arithmetic expression> part has a value less than or equal to zero; the single word mode will not execute the loop at all, the

multi word mode will execute the loop one time."

D2763 ALGOL - CLARIFICATION OF "REPLACE" STATEMENT

The B7000/B6000 ALGOL Reference Manual (Form No. 5001639) should be clarified with respect to the REPLACE statement, as follows:

On Page 5-81, in the second paragraph following the list, change the following:

"equal to or less than 48 bits" should be "equal to or less than 96 bits".

"longer than 48 bits" should be "longer than 96 bits".

In the following discussion, "<<" is a left brace as defined on Page 5-81 and ">>" is a right brace.

Replace the third through sixth paragraphs of Page 5-81 with the following:

"For a <<short string>>, once the string literal is evaluated at compile time, all traces of its character size are discarded. At run time, if the string literal is used in a <source part> where individual characters are to be copied from the string literal, the character size for the characters to be copied is determined by the character size of the stack-destination-pointer. If the character size of the stack-destination-pointer and the character size of the <<short string>> are not the same, the results are likely to be unexpected and undesired by the programmer.

Each <<long string>> is stored at compile time in a portion of an array, called a pool array, created by the compiler for use at run time. A pointer to the beginning of the string literal in the pool array is automatically created at run time with a character size equal to the maximum character size of the string literal.

The number of characters represented by a string literal is determined based on the maximum character size of the literal. For example,

```
4"C1"     is 2 characters long,
8"AB"     is 2 characters long,
48"01"    is 1 character long,
4"01""A"  is 2 characters long (if the default character
                                 size is EBCDIC)
```

<arithmetic expression>

<<short string>>. The stack-source-operand is initialized by the value of the <<short string>>, left-justified with zero fill into one word if the string literal is less than or equal to 48 bits in length, or into two words if the string literal is greater than 48 bits in length. The stack-integer-counter is initialized to the number of characters represented by the <<short string>>. The number of characters specified by the stack-integer-counter, of the size specified by the stack-destination-pointer, is copied from the stack-source-operand to where the stack-destination-pointer indicates.

<<long string>>. The stack-source-pointer is initialized with the pointer that points to the first character of the <<long string>> in a pool array. The stack-integer-counter is initialized to the number of characters represented by the <<long string>>. The specified number of characters is copied from the <<long string>> to where the stack-destination-pointer indicates."

On Page 5-82, in the paragraph starting with "<<short string>> FOR <arithmetic expression>.", change the first sentence to say:

"The stack-source-operand is initialized by the value of the <<short string>>, left-justified and repeated, if necessary, to fill the operand."

On Page 5-82, in the paragraph starting with "<<short string>> FOR <arithmetic expression> WORDS.", change the first sentence to say:

"The stack-source-operand is initialized by the value of the <<short string>>, left-justified and repeated, if necessary, to fill the operand."

D2764 ALGOL - "OWN" ARRAYS IN A PROCESSED PROCEDURE

The following addition should be made to the B7000/B6000 ALGOL Reference Manual (Form No. 5001639) under "PROCESS Statement" on Page 5-62:

"The PROCESSed procedure must not declare any OWN arrays. An attempt to do so will result in a run-time error."

D2765 ALGOL - LARGE NUMERIC LITERALS

The following paragraph should be added to the B7000/B6000 ALGOL Reference Manual (Form No. 5001639) on Page 2-6 as the second paragraph under "COMPILER NUMBER CONVERSION":

"The compiler will accept as an <unsigned number> any value that can be represented in double precision (that is, one with not more than 24 significant decimal digits). If this <unsigned number> does not contain an <exponent part> with @@ (specifying a double precision value), the single precision representation of that value will be used. If such an

<unsigned number> contains more than 12 significant digits, some precision will be lost **converting it to single** precision."

D2776 ALGOL - "MOD" OPERATOR WHEN SECOND PARAMETER <1

The B7000/B6000 ALGOL Reference Manual (Form No. 5001639), Page 6-4, should be changed as **follows:**

> **"The MOD** operator denotes remainder and has the following meaning:"

should be changed to read

> **"The MOD** operator denotes remainder and has the following meaning (for Z<1 the results of **the MOD** operator are undefined by the current implementation):"

D2777 ALGOL - MISSING **ACTUAL** PARAMETER IN DEFINE INVOCATION

The B7000/B6000 **ALGOL** Reference Manual (Form No. 5001639) should be changed as follows:

On Page 4-9 before the subheading "Pragmatics", add the following paragraph:

```
"It is possible for the <closed text> to be empty in a <define invocation>. In this case,
all occurrences of the corresponding <formal symbol> in the <text> are replaced by no text.
For example, if define F is declared by

    DEFINE F(M,N)=M+N#;

then the invocation of F in the statement
    R:=F(,1);
expands to
    R:=+1;
which is syntactically correct; however, the statement
    R:=F(2,);
expands to
    R:=2+;
which is syntactically incorrect."
```

D2778 ALGOL - FILES "LINE" AND "ERRORFILE"

The B7000/B6000 ALGOL Reference Manual (Form No. 5001639) should be corrected as follows.

The table on Page E-7/E-8 for file LINE, under KIND, should read:

> "LINE PRINTER"

For file ERRORFILE, under KIND, the table should read:

> "LINE PRINTER or REMOTE".

D2786 ALGOL - "$LOADINFO, $DUMPINFO" YIELD SYNTAX ERRORS

The following paragraph should be added to the description of $LOADINFO and $DUMPINFO in the B7000/B6000 ALGOL Reference Manual (Form No. 5001639) on Page D-20 and D-21:

> "Care must be taken that the release level of the compiler which performed the $DUMPINFO of a file and the release level of the compiler which performs the $LOADINFO of that file are the same. If they are not the same, a syntax error will be given and the compilation will be aborted."

D2787 ALGOL - "READ" ON THE "B5500" VS. THE "B6700"

The B7000/B6000 ALGOL Reference Manual (Form No. 5001639) should be corrected, as follows:

On Page 5-69, replace the last line of the second paragraph:

> "the record pointer is not adjusted after the READ operation."

with the following:

> "the record pointer is adjusted after the READ operation to point to the next record."

Also on Page 5-69, replace the line in the third paragraph:

> "the buffer is not released after it has been read or written;"

with the following:

> "the record pointer of the file is not modified;".

D2788 ALGOL - CLARIFICATION OF "INTEGER" FUNCTION

The B7000/B6000 ALGOL Reference Manual (Form No. 5001639) Page 6-24, under the description of INTEGER (<update pointer> <pointer expression>,<ae>), should be corrected by deleting the sentence:

"With 4-bit characters, a leading 1101 results in a negative value."

and adding the following paragraph:

"If the <pointer expression> results in a 4-bit (hex) pointer, then if the first character pointed at is numerically greater than 4"9", the numeric value of the function is determined by the <ae> characters beginning at <pointer expression> +1; otherwise, the numeric value of the function is determined by the <ae> characters beginning at <pointer expression>. If the first character pointed at is 4"D", then the result of the function will be negative."

D2814 ALGOL - COMPILER LABEL EQUATION TO "XREF"

The label-equation statements passed to the ALGOL compiler will now be passed to SYSTEM/XREFANALYZER when $XREF is set.

D2836 ALGOL - ARITHMETIC FUNCTION "VALUE"

The B7000/B6000 ALGOL Language Reference Manual (Form No. 5001639) needs clarification. On page 6-28, change the "parameter(s)" to VALUE to read "Attribute Mnemonic". Change the explanation under "result" to read as follows:

"The parameter to VALUE is any of the mnemonic names valid for the value of the file or task attribute or fields of an attribute; the value returned is the integer value of that mnemonic as known by the system.

The mnemonics valid for the value of the file attribute ATTYPE are the names of the file attributes themselves. The mnemonics valid for the value of task attribute ERROR are the names of the task attributes themselves. The value returned for these "mnemonics" is the attribute number corresponding to the file or task attribute named.

If the parameter to VALUE is a mnemonic of the task attribute OPTION, the value returned is the number of the bit in the OPTION word associated with that mnemonic (refer to the B7000/B6000 Series I/O Subsystem Reference Manual, Form No. 5001779, and to the B7000/B6000 Series Work Flow Language Reference Manual, Form No. 5001555)."

D2914 ALGOL - UNDEFINED RESULTS IN "REPLACE" STATEMENTS

The following paragraphs should be added to the ALGOL Language Reference Manual (Form No. 5001639) at the bottom of page 5-87:

"During a replace operation involving a <source>, if the stack-source-pointer ever references the first character position of the <destination>, the results of the operation are undefined. For example, the following REPLACE statement has an undefined result:

```
REPLACE POINTER(A)+6 BY POINTER(A) FOR 12;
```

On the other hand, the following REPLACE statement has a well defined result:

```
REPLACE POINTER(A) BY POINTER(A)+6 FOR 12;"
```

D2919 ALGOL - INTRINSICS WITH COMPLEX AND STRINGS

The possiblity of calling installation intrinsics of type complex and type string has been implemented. Also, complex and string parameters to installation intrinsics are now allowed.

ALGOL

P1279 ALGOL - CHANGE "INFO LEVEL"

The ALGOL compiler on the III.1 release now has INFO LEVEL set to 31.

P1373 ALGOL - STORING SINGLES INTO DOUBLES

Code will now be emitted to always extend a real or integer quantity whenever it is stored into a double precision item. The previous code was incorrect under certain circumstances.

P1449 ALGOL - SMALL POOLS FOR FORMATS

Due to an error in the calculation used to determine whether a format would fit into the current format pool, formats were sometimes mapped into very small pools. The calculation has been corrected.

P1759 ALGOL - BINDINFO FOR STRING PROCEDURES

Incorrect bindinfo was being generated for string valued procedures. This problem has been corrected.

P1760 ALGOL - NO ERROR ON "REAL (P:P EQL "AB")"

The compiler no longer gives a syntax error for the use of the function REAL (<boolean expression>), where <boolean expression> is of the following form:

```
    <pointer update><relational op>. . .
```

Example:

```
    REAL (P:P= "AB")
```

This expression will now compile correctly.

P1791 ALGOL - NO ERROR AFTER DIRECT "I/O" STATEMENT

The compiler will no longer generate a spurious syntax error on a fault statement immediately following a direct I/O statement.

Example:

```
    .
    .
    .
    EBCDIC ARRAY E[0:79];
    DIRECT FILE D;
    DIRECT ARRAY DA;
    READ (D,1,DA);
    ON ANYFAULT [E[0]], BEGIN END;
    .
    .
    .
```

No syntax error will be given on the "ON ANYFAULT" statement.

P1792 ALGOL - NO ERROR ON "TITLE=<PTR>"

Providing that the pointer has been previously declared, a syntax error will no longer occur on a file declaration with the following attribute assignment:

```
    TITLE=<pointer expression>
```

Example:

```
    .
    .
    .
    POINTER P;
    FILE REM(TITLE=P);
    .
    .
    .
```

This will now compile properly.

P1817 ALGOL - ARRAY REFERENCE ASSIGNMENT PROBLEMS

When a formal double array was assigned to a single reference array (or vice versa), no type conversion would take place.

Example:

```
    PROCEDURE P(D); DOUBLE ARRAY D[*];
    BEGIN ARRAY REFERENCE A[0]; REAL X; A:=D[0]; X:=A[0];
    END;
```

The combined assignments were equivalent to X:=D[0] instead of X:=FIRSTWORD(D[0]), as it should be according to the documentation. This problem has been corrected.

P1819 ALGOL - UPDATE POINTER IN "REPLACE BY" STRING

The destination pointer was not being updated in statements such as the following, where P is a pointer variable and STR is a string variable:

```
    REPLACE P:P BY STR;
```

This problem has been corrected.

P1878 ALGOL - "IF FALSE" IN PROCEDURES TO BE BOUND

If pool data, such as in-line formats, occurred in a procedure to be bound, it was possible for an IF FALSE THEN statement to corrupt the procedure's codefile such that an INVALID OP would occur if the bound codefile were executed. This problem has been corrected.

P1879 ALGOL - SYNTAX ERRORS NOT BEING REPORTED TO "WFL"

When separately compiling several procedures, if one of the procedures contained one or more syntax errors but the last procedure compiled without error, the existence of syntax errors would not be reported to WFL. Also, when SEPCOMPing several procedures, if one of the procedures compiled with one or more syntax errors but the last procedure compiled without error, the existence of syntax errors would not be reported to WFL. Both of these problems have been corrected.

P1880 ALGOL - "AUTOBIND" AND ONE LEVEL CODEFILE TITLE

Incorrect control card images were being passed to the binder for an AUTOBIND when the title of the codefile consisted of only one level; i.e., when the title contained no slashes. This problem has been corrected.

P1883 ALGOL - REUSE STRING TEMPORARIES

The ALGOL compiler now reuses, when possible, the EBCDIC string temporaries which are required for string expressions.

It was possible to get stack overflow on a multiple string assignment if the string expression consisted only of the EMPTY string function. This problem has been corrected.

P1933 ALGOL - "SERIALNO" FILE ATTRIBUTE

Possible faults have been corrected which could occur when the SERIALNO file attribute was assigned the value 0 in a file declaration or in a multiple attribute assignment statement.

P1953 ALGOL - MORE DESCRIPTIVE ERROR MESSAGE

A more descriptive error message is now generated when items cannot be monitored: CANNOT MONITOR THIS ITEM.

P1954 ALGOL - STRING TOO LONG ERROR MESSAGE

A possible compiler fault when emitting the error message that a string was too long has been corrected.

P1956 ALGOL - CORRECTLY HANDLE "$MCP"

The token after the dollar option MCP is no longer ignored.

P1957 ALGOL - ERRONEOUS SYNTAX ERROR IN VECTORMODE

A rare problem has been eliminated in which a GO TO from vectormode code could cause an erroneous syntax error.

P1958 ALGOL - LIST DECLARATIONS

The ALGOL compiler will no longer get an INVALID INDEX when many lists are part of the same declaration.

P1959 ALGOL - BAD "GO TO" FROM A SEPARATE PROCEDURE

A problem has been corrected where a bad GO TO from a procedure that had been bound into a host could cause unpredictable results.

P1960 ALGOL - ERRONEOUS SYNTAX ERRORS ON FORMATS

A problem of erroneous syntax errors being given on formats because of characters left over from a previous format has been corrected.

P1961 ALGOL - EXTERNAL PROCEDURES IN "PROCESS" STATEMENTS

Procedures declared external and then bound into a host can now be passed as parameters in a PROCESS statement.

P1962 ALGOL - NUMBERED "CASE" STATEMENTS

Numbered CASE statements occasionally could cause an incorrect branch. This problem has been corrected.

P1964 ALGOL - STACK BUILDING CODE

A rare problem in which incorrect stack building code was created has been corrected.

P1966 ALGOL - OPTIMIZED "IF" STATEMENT

A problem in which erroneous syntax errors occurred has been corrected. This problem occurred only if OPTIMIZE were set and an I/O statement with an in-line format occurred as a Boolean primary in an IF, WHILE or THRU statement. An example of a program which got erroneous syntax errors is the following:

```
$SET OPT
BEGIN
FILE X;
IF WRITE(X,</>) THEN;
END.
```

P1967 ALGOL - BINDING "ALPHA6" OR "ALPHA7" TRUTHTABLES

A problem with binding the ALPHA6 or ALPHA7 truthtables has been corrected.

P1968 ALGOL - "$XREFFILES" INTERACTION WITH "$XREF"

A rare problem in which using SET and RESET of $XREFFILES and $XREF could cause improper results has been corrected.

P1969 ALGOL - FLAG EXTRANEOUS CROSSHATCH

Extraneous crosshatches are now flagged properly. Previously, they were only flagged if the compiler were compiled with CTPROC set.

P1970 ALGOL - SYNTAX ERROR ON "IF" STATEMENT

A problem in which an optimized IF statement could get an erroneous syntax error has been corrected. An example of such a statement is the following:

```
THRU 3 DO
   IF WRITE(X,<H12>) THEN;
```

P1971 ALGOL - "FIRSTWORD" AND "SECONDWORD" OF CONSTANT

A problem in which FIRSTWORD or SECONDWORD of a constant could return a double precision constant with a secondword of zero has been corrected. This problem only occurred under extremely rare conditions and in most cases, program results were unaffected.

P1972 ALGOL - ERROR LIMIT OF ZERO

Setting the dollar option LIMIT to zero no longer limits the number of syntax errors. Previously, if $LIMIT were zero, a warning or an error message caused an immediate message of "ERROR LIMIT EXCEEDED" and termination of the compile.

P1974 ALGOL - ELIMINATE COMPILER ERROR

A possible compiler error has been eliminated. This error would occur if a user dollar option were used between the word PROCEDURE and the procedure identifier; e.g.,

```
PROCEDURE
  $SET OMIT=X
  $POP OMIT
K;
```

P1975 ALGOL - "FOR" STATEMENT SYNTAX

A FOR statement with an embedded ":=*" assignment to the control variable will now be compiled correctly. Previously, if the dollar options OPTIMIZE and B7700 were set, an incorrect syntax error would occur.

P1976 ALGOL - "COMPILETIME" SYNTAX ERRORS

The use of the intrinsic COMPILETIME will no longer cause erroneous syntax errors. An example of a program which would previously syntax is the following:

```
BEGIN
REAL X;
IF FALSE THEN X:=COMPILETIME(21);
END.
```

P1977 ALGOL - "MAKEHOST" AND INNER BLOCKS

Compiling with $MAKEHOST set and with extra environments specified will no longer lead to a rare end-of-file on the code file or to an erroneous error message.

P1978 ALGOL - "BREAKPOINT/SEPCOMP" CORRECTION

A problem in which the compiler emitted bad code has been corrected. This problem occurred when a series of separate procedures were compiled where one of the procedures ended with the dollar option BREAKPOINT set. The following procedure would have incorrect code after each statement until a BREAKPOINT dollar card or statement was encountered.

P1979 ALGOL - "IF" EXPRESSION COMPATIBILITY

An IF expression incompatibility between the B6700 and B7700 has been eliminated.

P1981 ALGOL - "STATISTICS" CORRECTION

A problem in which an ALGOL program compiled with dollar option STATISTICS set could get an INVALID OP in stack building code has been corrected. An example of a program which previously failed is the following:

```
$ SET STATISTICS
BEGIN
IF FALSE THEN
  BEGIN
  REAL X;
  END;
END.
```

P1982 ALGOL - LARGE FORMATS

The use of extremely large formats will no longer cause system dumps.

P1983 ALGOL - HANDLE "TASKFILE" CORRECTLY

The ALGOL compiler will now emit correct code for interrogating the value of the following:

```
<task>.TASKFILE.<file attribute>
```

P1984 ALGOL - BINDER INFORMATION FOR A LARGE PROGRAM

A problem in which the ALGOL compiler output incorrect binder information for an extremely large program has been corrected.

P1988 ALGOL - CORRECT XREF OF FILE MONITOR DECLARATIONS

File monitor declarations will be XREFed correctly. Previously, the compiler could fault with an INVALID INDEX.

P1989 ALGOL - DIRECT FILES AND ATTRIBUTES

The order of assigning file attributes within a DIRECT file declaration has been modified so that the file attribute DIRECT will be assigned before any user specified attributes. Previously, a file declaration such as the following:

```
DIRECT FILE F(KIND=DISK, NORESOURCEWAIT)
```

would cause a file attribute error because the NORESOURCEWAIT attribute (which is invalid for non-direct files) was assigned before the DIRECT attribute.

P1990 ALGOL - PREVENT COMPILER FAULT

A problem has been corrected which caused the compiler to fault under a set of circumstances that included the following: $LINEINFO set, a multi-procedure source being compiled, blank records or comments between procedures, and a syntax error in one of the procedures.

P1991 ALGOL - RECOVERING FROM SYNTAX ERROR

A possible INVALID INDEX that could occur when recovering from syntax errors has been eliminated. The following program could cause this problem:

```
    BEGIN PROCEDURE P(I) BEGIN END; END.
```

P1992 ALGOL - CORRECT "IF" STATEMENT BRANCHES

A problem where the ALGOL compiler was generating incorrect branches when compiling nested IF statements has been corrected. A program for which incorrect code was generated is the following:

```
   BEGIN
   REAL GR1;
   IF GR1=1 THEN GR1:=2
   ELSE IF GR1 NEQ 3 THEN
        IF FALSE THEN
             GR1:=4;
   END.
```

P1993 ALGOL - EXTERNAL PROCEDURE PARAMETER MISMATCH

A possible parameter mismatch error on external procedures has been corrected.

P1994 ALGOL - TRUTHSETS IN "IF" CONDITIONS

A syntax error will no longer occur when $OPTIMIZE is set and truthsets ALPHA6, ALPHA7, or ALPHA8 occur in the <condition> of an IF statement. An example of a program which previously syntaxed is the following:

```
   $SET OPTIMIZE
   BEGIN
     POINTER P;
     IF P IN ALPHA6 THEN P:=P+5;
   END.
```

P1995 ALGOL - "$INCLUDE" USING STRINGS

When an error is encountered within a $INCLUDE compiler control statement using string literals, the correct error message is now generated. Previously, the compiler faulted with a SEG ARRAY error.

P1996 ALGOL - "FOR" STATEMENT WITH VARIABLE STEP

Previously, a program could fault with exponent overflow in executing the check for loop completion in a FOR statement with a variable step and upper limit when these values were very large. This problem has been corrected.

P1997 ALGOL - "CTPROC" COMPILER HANG

Under certain conditions, the CTPROC compiler was trying to read from CTCOPYFILE before it was opened. The compiler will now check for CTCOPYFILE being open.

P2028 ALGOL - UPLEVEL POINTER DETECTION

Uplevel pointer checking for OWN pointers has been corrected. The checking will now be based upon the level at which the pointer actually exists rather than upon the level at which the pointer was declared.

P2029 ALGOL - WRITING STRINGS IN ERROR MESSAGES

When the scanner is pointing to a string literal and an error is encountered, the internal evaluation of the string was written in the error message. This string could produce unpredictable results on certain terminals. To remedy this problem, in the case of strings, the token at which the scanner is pointing when the error is encountered will not be displayed after the error message.

P2030 ALGOL - ERRONEOUS REPEAT COUNT IN VALUE ARRAYS

If in the following value array declaration:

```
    VALUE ARRAY VA (<repeat count> (<constant list>))
```

the <repeat count> had the value 0, the <constant list> was put twice in the array. Also, if the <repeat count> was a double precision value, the first word of that value was taken as the repeat count. These problems have been corrected.

P2110 ALGOL - "INVALID OP" IN FREEFIELD "WRITE" STATEMENT

If a freefield WRITE statement (with the option */), in which one or more of the list elements are arrays or array elements, were followed by an inner block, execution of the WRITE statement caused an INVALID OP. This problem has been corrected.

The following example previously failed and now works correctly:

```
BEGIN FILE F (KIND=PRINTER);
      ARRAY A[0:10];
      WRITE (F,*/,A);
      BEGIN INTEGER I;
            I:=0;
      END;
END.
```

P2111 ALGOL - WARNING COUNT RESET

The listing trailer for a batch job will now contain a warning count. The warning count is now reset to zero between separately-compiled procedures and between batch jobs so that the warning count printed in the trailer of a separately-compiled procedure or batch job reflects only the warnings which occurred in that procedure or batch job.

P2112 ALGOL - "BCL" WARNING MESSAGE OCCURS CORRECTLY

The BCL warning message which appears in the listing trailer has been corrected so that the trailer of a separately-compiled procedure containing no BCL constructs following a procedure containing BCL constructs and the trailer of a batch job containing no BCL constructs following a batch job containing BCL constructs will not contain the BCL trailer message.

P2113 ALGOL - "INVALID OP" IN FREEFIELD "WRITE"

The following WRITE statement, which previously caused an INVALID OP under certain circumstances, now works correctly:

```
WRITE (F,/,*A[I])
```

P2114 ALGOL - "INVALID OP" IN "READ, WRITE" STATEMENT

If a FORMAT or LIST declaration was followed by a PROCEDURE declaration whose body was not a block, the use of the format or list in a READ or WRITE statement would, in some cases, cause an INVALID OP. This problem has been corrected.

P2200 ALGOL - "<FILE ID>.<FILE ATTRIBUTE>:=*"

Incorrect code is no longer produced for an assignment statement of the following form:

```
<file id> . <file attribute> := * . . .
```

P2218 ALGOL - WARNING MESSAGE GIVEN FOR FORMAT "A4030"

The text of the warning message given for a format specification with a field width that is too large has been improved. The previous message:

"NUMBER TOO LARGE. 2*39-2 WAS USED."

was inaccurate. The new message is the following:

"NUMBER TOO LARGE FOR THIS FORMAT SPECIFICATION"

P2236 ALGOL - "IF FALSE" AND "$LINEINFO"

A problem has been corrected which caused a SEG ARRAY fault to occur in the ALGOL compiler for certain IF FALSE THEN <statement> constructs when $LINEINFO was set. For example, the following program, which previously would cause the compiler to abort, now compiles correctly:

```
$SET LINEINFO
BEGIN
IF FALSE THEN
     BEGIN
     REAL I;
     IF I=0 THEN
          BEGIN
          END;
     END;
END.
```

P2237 ALGOL - SYNTAXING OF FORMAT SPECIFICATION DELIMITERS

The ALGOL compiler will now correctly give a syntax error on an in-line format specification which ends in a right parenthesis instead of a right angle bracket (>), on a format or switch format declaration in which the <editing specifications> are delimited by one parenthesis and one angle bracket, and on <editing specifications> within <editing specifications> which are delimited by a left parenthesis and a right angle bracket. For example, the following will now get syntax errors:

```
WRITE(F,<I5),R);
FORMAT FMT(A6>;
SWITCH FORMAT SFMT:=<F13.3);
FORMAT HX(5(H12,X2>);
```

P2238 ALGOL - "$PAGE" AND THE COMPILER LINECOUNT

Improvements have been made to the updating of the compiler's internal linecount, the number of lines printed per page, which is used for $FORMAT. In particular, the linecount will now initialized to zero whenever a $PAGE is encountered.

P2239 ALGOL - COMPILER LOOP ON "DATABASE" DECLARATION

A problem has been corrected in which the ALGOL compiler looped giving syntax errors for declarations beginning with the words DATABASE, TRANSACTION (for III.1 only) or NUMBER.

P2240 ALGOL - REMOVE "?" FROM ALPHA6

The character "?" has been removed from the truthset ALPHA6. ALPHA6 now correctly consists only of the BCL letters and digits.

P2248 ALGOL - EXCLAMATION MARK AND UNDERSCORE

A problem has been corrected in which incorrect bit patterns were being given by the compiler for ASCII or BCL string literals containing the exclamation mark or underscore.

P2289 ALGOL - NO "COMPILER ERROR IN BUILDITEMDESC"

The ALGOL compiler will no longer generate the "COMPILER ERROR - IN BUILDITEMDESC" message after the "FORMAL PARAMETER NOT SPECIFIED" message.

P2290 ALGOL - NO SPURIOUS "UNKNOWN DOLLAR CARD" ERROR

The ALGOL compiler will now generate the correct error when the dollar options XREF and XREFFILES occur after the first source image.

P2439 ALGOL - "CASE" STATEMENTS WITH ALL CASES "GO TO"

CASE statements of the form described below will no longer "fall through" (no case executed) when the arithmetic expression has its highest possible value.

All the following conditions had to have been met for the failure to occur:

1. The case selection expression ended with an isolate of 5 or fewer bits.

2. The case statement had an "ELSE:" clause.

3. All cases, including the ELSE, were unconditional GO TO statements.

4. The highest value for the expression (after the isolate) was not an explicit case label.

The problem has been corrected.

P2579 ALGOL - STRING IN "DEFINE" TOO LONG

A SEG ARRAY fault termination has been corrected which occurred when a quoted string which was too long was included in a DEFINE.

P2587 ALGOL - SYNTAX ERROR WITH "$INTRINSICS" SET

An INVALID INDEX in the ALGOL compiler following a syntax error in global declarations with $INTRINSICS set has been corrected.

P2593 ALGOL - SEPCOMP OF PROCEDURES WITH STRINGS

If a procedure which contained strings was sepcomped into, it was possible for the resultant code file to fault with STRING POOL EXCEEDED. This problem has been corrected.

P2594 ALGOL - NO STRING PARAMETERS BY VALUE

Strings are no longer allowed as by-value parameters for CALL or PROCESS statements and are not allowed either by-name or by-value for RUN statements. This prohibition prevents the untimely disappearance of the contents of string variables and the occurrence of a variety of system fatal dump situations.

ALGOL INTRINSICS

D2565 ALGOLINTRN - INTRINSICS

Some of the intrinsics previously coded in ESPOL or DCALGOL have been recoded in ALGOL, and placed in separate symbol files. Those which were previously in the file SYMBOL/PLINTRINSICS have been placed in the file SYMBOL/ALGOLPLINTRINSICS. Those which were previously in the files SYMBOL/ESPOLINTRINSICS and SYMBOL/DCALGOLINTRINSICS have been placed in the file SYMBOL/ALGOLINTRINSICS.

When creating an intrinsic file from scratch, SYMBOL/ALGOLINTRINSICS and SYMBOL/ALGOLPLINTRINSICS should be compiled and the resulting object files bound with the other intrinsics. There are no special dollar options to set when compiling these two symbolic files, with the possible exception of the "LIBRARY" option.

D2626 ALGOLINTRN - "DISPLAYTOSTANDARD" DESTINATION POINTER

Previously, the DISPLAYTOSTANDARD intrinsic did a word replace into the destination pointer, the second parameter, so if that pointer were not at a word boundary, the standard form name would begin at the next word boundary. Now, the standard form name will begin exactly where the destination pointer specifies, regardless of whether it is at a word boundary.

ALGOL INTRINSICS

P2140 ALGOLINTRN - "CSQRT" WITH ZERO IMAGINARY PART

CSQRT will no longer return different results depending upon the imaginary part being positive or negative zero.

ATTABLEGEN

D2588 ATTABLEGEN - "IPCOVERRIDE" RECOGNIZED AS TASK OPTION

The mnemonic IPCOVERRIDE is now recognied as a value for the task attribute OPTION.

BACKUP

D2583 BACKUP - "BACKUPBYJOBNR"

The discussion of BACKKUPBYJOBNR SYSTEM OPTION in "System Software Operational Guide", Volume 1, page 1-5-2 has been revised. The second paragraph has been corrected and now reads as follows:

> When this option is set, jobs are printed by order of the job number. When reset, jobs are printed in reverse order of print quantity.

D2585 BACKUP - "END" AS RANGE STOP INDICATOR

The System Software Operational Guide, Volume 1, No. 5001563, should be corrected as follows:

Page 1-7-3

The syntax diagram for <range part> should be corrected to read as follows:

<range part>

```
    |<-------------------------------------------|
    |                                            |
------ RANGE ---<number>-------------<number>------|
      |            |        | |     | |          |
      |            |-<string>-| |- : -| |-<string>-|
      |                                 |          |
      |                                 |- "END" --|
      |                                            |
      |- EQUAL ---<number>-------------------------|
                   |                               |
                   |-<string>----------------------|
```

Page 1-7-7

The semantic description of END should be corrected to read as follows:

"END" Is a range stop indicator for an EBCDIC string (besides either <number> or <string>) which is equivalent to setting the stop integer to 99999999.

D2789 BACKUP - "NEWP" KEY SPECIFIED

A NEWP key specifier has been added to SYSTEM/BACKUP.

The B7000/B6000 System Software Operational Guide, Volume 1 (Form No. 5001563) should be corrected, as follows:

On Page 1-7-3, the <key part> syntax diagram now reads as follows:

<key part>

```
----<key start>--<key length>----|
   |                       |
   |- ALGOL ---------------|
   |                       |
   |- COBOL ---------------|
   |                       |
   |- FORTRAN -------------|
   |                       |
   |- NEWP ----------------|
   |                       |
   |- REPORT --------------|
```

On Page 1-7-6, add the following semantic description of NEWP after the semantic description of FORTRAN:

> "NEWP Is a key specifier indicating the appropriate columns for the NEWP sequence numbers on compilation listings generated by this compiler."

D2870 BACKUP - "BACKUP" SYNTAX

The System Software Operational Guide, Volume 1 (Form No. 5001563), Page 1-7-1, shows incorrect syntax for <disk file>. The syntax diagram should be the following:

```
-------------------------------------------------------------|
   |                                                       |
   | |<-/10\--------|                                      |
   | |              |                                      |
   |--- / <number> ---<number>-- <printer/punch filename> -|
```

BACKUP

P1761 BACKUP - FILENAMES WITH SPECIAL CHARACTERS

BACKUP can now handle filenames with special characters; e.g., backup files with hyphenated names generated by COBOL programs may now be selectively PBed.

P1762 BACKUP - ALL COPIES NOT PRINTED

The following command would print the first copy, but not print further copies:

```
    PB <filename> ON <packname> COPIES=<n>
```

where <n> was greater than 1. This problem has been corrected; now, all copies are printed.

P1885 BACKUP - LP# LOST AFTER FORMS FILE

If one file of a list of files has a forms request, the designated printer is lost after the forms file is printed; e.g.,

```
    ?PB "XYZ" LP#
```

Now if one of the "XYZ" files has a forms request, the remaining files are printed on the designated printer after the forms file is printed.

P2003 BACKUP - BACKWARD SKIP

BACKUP would occasionally get a SEG ARRAY error after a backward skip, because the number of lines to space backward were incorrectly counted. This has been corrected.

P2004 BACKUP - "BFILE" LABEL EQUATON

If the following were entered:

```
    PB ND; FILE BFILE(KIND=DISK)
```

BACKUP would not lock the file if DISK were physically a disk pack. This has been corrected.

P2567 BACKUP - BUFFER REDUCTION FOR "LIN" FILE

BACKUP will now use one BUFFER for the LIN file, reducing the save memory requirements by approximately 300 words.

P2568 BACKUP - LOOPS WHEN "<SKIP COUNT>" OMITTED

BACKUP would loop when the operator changed printers without skipping lines. This problem has been corrected.

P2569 BACKUP - "REEL" ATTRIBUTE NOT SET BEFORE "OPEN"

BACKUP will now set the REEL specification in the PB command before it tries to OPEN the file.

Example:

```
    ?PB MT "<filename>" REEL=8 (KIND=PRINTER)
```

This input will cause the following message if the tape is not mounted:

```
    "NO FILE BACKUP/<filename> (MT)=8"
```

Note: The use of the following will cause the tape on the requested unit to be opened, regardless of the setting of the REEL value:

```
        ?PB MT113 REEL=8 (KIND=PRINTER)
```

P2570 BACKUP - FILENAMES LIMITED TO "60" CHARACTERS

BACKUP will now allow filenames of length greater than 60 characters.

P2571 BACKUP - INVALID OPTION COMBINATION

The use of BACKUP within a JOB to print BD files before the JOB has terminated is restricted to direct output only. The use of "ND" and either "*" or any other method of supplying the JOB's JOBNUMBER to BACKUP will result in the following error:

```
    "INV OPTION COMBINATION"
```

The following example will produce the above error:

```
?BEGIN JOB X
 STRING S;
 RUN Y;
 S:="D" & STRING (MYJOB(STACKNO),4) & "ND";
 RUN *SYSTEM/BACKUP(S)
?END JOB
```

P2575 BACKUP - "ODT" DISPLAY FORMAT

The ODT display made by BACKUP has been reformatted to remove blank lines and "floating" periods at the end of each message.

P2576 BACKUP - FIRST LEVEL FILENAME VS. USERCODE

In some cases, BACKUP would interpret the first level of a file name returned by GETSTATUS as a USERCODE, when, in fact, it was a SYSTEM file. This problem has been corrected.

P2580 BACKUP - "BACKUP" VS. SYSTEM WITHOUT "DISK"

BACKUP will no longer hang on a "<filename> REQUIRES DK DISK" message on systems that do not have a disk or pack labeled "DISK". BACKUP now makes a FILEIN.AVAILABLE check instead of a FILEIN.PRESENT which allows it to continue on a "NO FAMILY PRESENT" error.

BARS

D2978 BARS - "SYSTEM/BARS" UTILITY

This note describes the new performance reporting utility SYSTEM/BARS. This utility monitors the system's performance and displays it in the form of numeric values and bar graphs. Various system performance parameters are sampled and may be displayed in a user-controllable format on screen-type terminals.

Performance elements are represented by a bar on the screen. The format of each bar is ###XXX... where the #'s extend to the minimum value seen, the X's extend to the current value, and the .'s extend to the maximum value seen. The display is updated every 'cycle' seconds, where cycle is a parameter which is dynamically variable from the terminal, and a running average is maintained over a user-settable period.

The following verbs are used to control the display:

HELP (alias TEACH)
Displays this information.

CYCLE
Controls the sampling and terminal update interval.

PERIOD
Several values are computed as running averages using the formula:

```
NEWAVERAGE:=
(OLDAVERAGE * (PERIOD-CYCLE) + NEWVALUE * CYCLE) / PERIOD
```

PERIOD changes the value of period. If PERIOD < CYCLE, no averaging is done (i.e. the exact value is displayed).

NEWDISPLAY
Modifies the format and content of the screen. Each item on the screen is specified by giving a key word followed by N's for the value field and B's for the bar graph (both N's and B's are optional); for example:

```
Idle NN BBBBBBB
```

will produce a display of the form: Idle 45 ###X. Text may be placed on the screen by placing it within single quotes: 'xyz'. The valid key words and their meanings can be displayed with the WORDS verb. A default display is supplied by the program. The NEWDISPLAY input that creates it can be displayed with the DISPLAY verb.

DISPLAY
Displays the input that would create a specified display screen. Three options exist:

DISPLAY Shows the NEWDISPLAY input for the current screen.

DISPLAY DEFAULT
Shows the NEWDISPLAY input for the default screen.

DISPLAY <file name>
Shows the NEWDISPLAY input for a screen SAVEd in the specified file; e.g. DISPLAY X/Y ON P.

SAVE <file name>
Saves the current display in the specified file; e.g. SAVE X/Y ON P.

LOAD
Loads a previously generated screen as the new screen. The options for DISPLAY are also valid for LOAD:

LOAD Loads the current display. The screen does not change.

LOAD DEFAULT
Loads the default screen for monolithic systems.

LOAD TCDEFAULT
Loads the default screen for a B6800 multiprocessor system.

LOAD TCMEMORY
Loads a screen giving overlay information for each memory structure of a B6800 multiprocessor system.

LOAD <file name>
Loads a previously SAVEd display.

WORDS
Displays the allowed key words for the NEWDISPLAY input plus a short description of their meaning.

PACK (alias PK, alias PERPK)
Displays the names, channel numbers, and family indicies of the packs currently on-line.

BYE Ends the program.

The program begins by executing a LOAD DEFAULT command. The program is initialized to a different display if the file DISPLAY is label-equated to a SAVEd file; for example, FILE DISPLAY = MY/SCREEN.

If the file MONITOR is label-equated, the program will write the raw performance data to that file as it is received from the MCP; for example, FILE MONITOR = XXX.

BINDER

D2589 BINDER - BINDING "NEWP"

The BINDER will now handle binding to MCP codefiles compiled in NEWP. The following restrictions apply to a subprogram being bound in:

1. The subprogram cannot have been compiled in NEWP.
2. The subprogram must have been declared external in the host.
3. The subprogram cannot add any globals to the host or contain any OWN variable declarations.
4. The only global variables that can be referenced by the subprogram are ones that were declared in the outerblock of the MCP.

The BINDER will not handle binding to non-MCP hosts compiled in NEWP.

D2714 BINDER - TRANSACTION RECORD PARAMETERS

Separately-compiled program units may not be bound if they declare transaction records as parameters.

D2722 BINDER - BINDING OF PORTS AND SIGNALS

The BINDER is now able to bind DCALGOL PORT and SIGNAL variables and arrays.

Ports and signals that are parameters or globals are handled. Ports and signals that are globals that are added to a host will not have their attribute information copied into the host (as is traditionally the case with files added to hosts).

D2766 BINDER - INTERNAL HANDLING OF WARNINGS

The BINDER's handling of warning messages has been improved.

Also, one "BINDER ERROR" error message has been changed from a warning into an error. The handling of the "SOME PROCEDURES NOT BOUND" message has been improved.

D2770 BINDER - PLACEMENT OF THE "MCP D0" STACK AND SAVE CODE

When binding into an MCP, the BINDER will no longer place the D0 stack and the save-code at the beginning of a disk row. Now, if the two pieces will fit in the current row, they will be put there.

D2771 BINDER - REBINDING EXTERNAL PROCEDURES

It is now possible to rebind the various DCALGOL procedures into a NEWP MCP host. Replacement binding is not allowed for procedures in the NEWP host, except for externals which were bound in and need to be rebound.

Because of the interaction with the NEWP and SEPCOMP facility, the old code of the procedures being rebound is retained in the code file (although it is not pointed at and cannot be executed); thus, a substantial number of rebindings can cause the MCP code file to grow undesirably large.

D2815 BINDER - MULTIPLE "BIND=" CARDS

If more than one BIND= statement is given to the BINDER, all but the last one will be ignored. An appropriate warning message will be given.

SOFTWARE IMPROVEMENTS NOTES (P NOTES)

BINDER

P1513 BINDER - ALLOW MANY INSTALLATION INTRINSICS

The BINDER now handles many installation intrinsics by giving the message "CODE.AREASIZE TOO SMALL-INCREASE IT BY LABEL EQUATION OF FILE CODE". When the AREASIZE is increased, the large number of installation intrinsics will no longer cause a problem.

P1514 BINDER - DETECT LARGE "INTRINSICINFO"

The BINDER will now detect when INTRINSICINFO does not fit on a disk row.

P1515 BINDER - BACK OUT CORRECTLY

If the BINDER discontinues a BIND of one subprogram, incorrect placement of global variables will not occur when binding another subprogram.

P1516 BINDER - BINDING ENTRY POINTS

The binding of subprograms found in multi-procedure code files (produced when the compiler option $LIBRARY is SET) has been corrected.

Some problems with binding FORTRAN entry points, whether these entry points were in multi-procedure code files or in separate code files, have also been corrected.

Symptoms of the erroneous binding were unbound procedures and entry points, binding procedures more than once, parameter mismatches, and improper internal generation of USE statements.

The search for FORTRAN entry points has been isolated to FORTRAN and bound code files only and has been optimized to use much less processor and I/O time.

P1517 BINDER - PRESERVE CONTROL STATE

Installation intrinsics declared to be control state procedures will still be control state after binding. Previously, the BINDER was turning off the control state bit.

P1519 BINDER - CORRECT IDENTIFIER RECOGNITION

The BINDER now correctly recognizes identifiers containing the following character:

' (apostrophe)

P2031 BINDER - PREVENT "INVALID OP" IN BOUND PROGRAM

The BINDER would previously generate bad parameter checking code under the following condition: Binding a procedure that invoked a formal parameter procedure that had a double precision parameter. This has been corrected.

P2032 BINDER - CLEAR INTERNAL ARRAY

The BINDER occasionally failed to clear an internal array. This would cause either an INVALID INDEX termination of the BINDER later during the bind, or it would cause a global to be incorrectly added to the bound program at address (0,1). This has been corrected.

P2178 BINDER - PROPER REBINDING OF STRING PROCEDURES

The BINDER now properly preserves the procedure descriptions for ALGOL type STRING procedures in the code file that it creates. The major effect is that programs containing these STRING procedures may be correctly rebound.

P2179 BINDER - UPDATE ERROR MESSAGES

The BINDER is now able to give proper "TYPE MISMATCH" error messages for data bases and ALGOL string variables, arrays and procedures.

P2434 BINDER - INSTALLATION INTRINSICS USING "MCP" PROCEDURES

The BINDER will now correctly bind installation intrinsics that reference MCP procedures and tables that were not referenced by any system-supplied intrinsics.

Previously, the BINDER would correctly add the first installation intrinsic that added an MCP "intrinsic" reference; subsequent installation intrinsics that referenced this MCP intrinsic would get a bad address couple generated.

P2465 BINDER - ENTRY BINDING CORRECTIONS

Instances where the BINDER would look in the wrong code file when trying to find FORTRAN entry points have been corrected. The symptom of this problem was the following erroneous error message:

"IDENTIFIER OF SEPARATE PROCEDURE AND HOST DIFFER"

The BINDER now looks in the proper code files.

P2492 BINDER - REBINDING WITH "FORTRAN" "DATA" STATEMENTS

When rebinding a FORTRAN procedure that contained a DATA statement that initialized an array, the BINDER was throwing away too many segment dictionary items from the host. The effect of this was an INV OP termination when attempting to run the re-bound program. This problem has been corrected.

CANDE

D2590 CANDE - "III.1" CONTROLLER KEYINS

CANDE will now recognize III.1 controller keyins as synonyms for existing controller keyins, as follows:

```
    "?ID"    is recognized as a synonym for "?DC"
    "?SC"     "     "               "       "?PC"
    "?TD"     "     "               "       "?WD"
    "?SI"     "     "               "       "?WI"
    "?MXA"    "     "               "       "?MIXA"
    "?DUMP"   "     "               "       "?DP"
```

D2634 CANDE - "SUBSYSTEM" TASK ATTRIBUTE

The SUBSYSTEM task attribute is now recognized as a modifier for CANDE task commands; e.g., RUN, COMPILE, LFILES. Compile-time specification of an object-file SUBSYSTEM is not possible in CANDE.

Syntax:

```
-- SUBSYSTEM ---------<name>--|
              |    |
              |- = -|
```

```
<name>

  |<-----------------|
  |                  |
----/17\---<letter>------|
          |        |
          |-<digit>--|
```

Example:

```
    RUN PROG; SUBSYSTEM=ABC
```

D2636 CANDE - "Q-DS" MESSAGE FOR START JOB

CANDE will now give a descriptive message when a job started via a START command is Q-DSed.

D2665 CANDE - "LSN" RANGES

The following control commands will now accept an LSN range; i.e., execute the respective command for each LSN in the range specified:

```
   ATTACH     READY
   AUTOANS    RELEASE
   NOAUTO     SAVE
   CLEAR      SS
   DISABLE    STATUS
   ENABLE
```

The syntax for an LSN range is as follows:

```
--<lsn>-----------------|
        |              |
        |- - --<lsn>-|
```

The syntax for each command changed is follows:

```
                     |<-------- , --------|
                     |                    |
-- ? -- ATTACH -----<lsn range>---------|
        --            |                   |
                      |-<dls>----------|
                      |                   |
                      |-<station name>-|
                                    |<-------- , --------|
                                    |                    |
-- ? --- AUTOANSWER ---------<lsn range>---------|
         |  ----              |   |                 |
         |- NOAUTOANSWER -|   |-<dls>----------|
            ----                  |                 |
                                  |-<station name>-|


                    |<-------- , --------|
                    |                    |
-- ? -- CLEAR -----<lsn range>---------|
                      |                 |
                      |-<dls>----------|
                      |                 |
                      |-<station name>-|


                      |<-------- , --------|
                      |                    |
-- ? -- DISABLE -----<lsn range>---------|
        ----            |                 |
                        |-<dls>----------|
                        |                 |
                        |-<station name>-|


                     |<-------- , --------|
                     |                    |
-- ? -- ENABLE -----<lsn range>---------|
        ----           |                 |
                       |-<dls>----------|
                       |                 |
                       |-<station name>-|


                    |<-------- , --------|
                    |                    |
-- ? -- READY -----<lsn range>---------|
                      |                 |
                      |-<dls>----------|
                      |                 |
                      |-<station name>-|


                      |<-------- , --------|
                      |                    |
-- ? -- RELEASE -----<lsn range>-------- TO --<file title>--|
        ----            |                 |
                        |-<dls>----------|
                        |                 |
                        |-<station name>-|


                   |<-------- , --------|
                   |                    |
-- ? -- SAVE -----<lsn range>---------|
        --           |                 |
                     |-<dls>----------|
                     |                 |
                     |-<station name>-|


                     |<-------- , --------|
                     |                    |
-- ? -- STATUS -----<lsn range>---------|
        ---            |                 |
                       |-<dls>----------|
                       |                 |
                       |-<station name>-|
-- ? -- SS --- SPO -------------------<text>--|
              |                         |
              |- ALL ------------------|
              |                         |
              | |<-------- , --------| |
              | |                     | |
              |-----<lsn range>--------|
                   |                |
                   |-<dls>----------|
                   |                |
                   |-<station name>-|
```

Examples:

```
?STA 10-20,TI8
?ATTACH 3-4,35-49
?RELEASE 6-10 TO SYSTEM/DIAGNOSTICMCS
?ENAB TTY1, 4:14, 50-53, ADM10
```

D2666 CANDE - "OPTIONS COMMANDS"

The RO and SO commands have been changed to allow the specification of more than one option with a single command.

The syntax for these verbs is now as follows:

```
                |<--------------|
                |               |
---- RO -----------------------|
   |        |  |            |
   |- SO -|  |- MESSAGE -|
                 ----
              |- MSG -----|
                 ---
              |- QWAIT ---|
                 ---
```

Example:

```
RO QWAIT MSG
```

D2667 CANDE - "CONTROL" COMMAND EXTENSIONS

The control command CONTROL has been extended to include a few new functions. The syntax for the command is as follows:

```
-- ? -- CONTROL ------------------------------|
        ----       |                        |
                   |---------<lsn>----------|
                   |     |  |               |
                   |- - -|  |-<dls>---------|
                            |               |
                            |-<station name>-|
                            |               |
                            |- ALL ---------|
```

The semantics of the command are exactly as they are previously, with the following extensions:

1. If no LSN is specified (?CONTROL), all control stations are displayed in a list.
2. If a minus (-) LSN follows CONTROL (?CONTROL-<lsn>), that station is removed from the control station list and its control station status removed.

D2668 CANDE - "WHAT" NEXT EXTENSIONS

The WHAT next has been extended to check that families and files referenced by the workfile are available, and displays the title if they are not. It also extends recovery to do an automatic check on these families and files after they have been recovered.

D2670 CANDE - MULTIPLE LOG STATIONS

A CANDE LOGSTATION is a station which receives CANDE log messages and, if LGSPO is enabled, receives ?SS messages directed to "ODT". Previously, only one such station could be identified at a time. It is now possible to designate up to thirty (30) stations as CANDE LOGSTATIONs, each with all or its own subset of the possible logging functions set. Three new verbs (LGSTA, DSLGSTA, and LGOP) have been implemented to initiate, terminate, interrogate and/or change the logging functions of a designated station. The syntax, semantics and examples of each of the new commands follows.

For all the syntax, the following definitions apply:

```
<station list>

   |<--------- , ---------|
   |                      |
------ <station name> ------------------------------------------|
     |                  |
     |- <lsn> ----------|
     |                  |
     |- <dls> ----------|
```

```
<option list>

   |<------ , ------|
   |                |
------- LGATTACH ----------------------------------------------|
     |   ----         |
     |-- LOGON ----   |
     |   ----         |
     |-- LOGOFF ---   |
     |   -----        |
     |-- LGCHARGE -   |
     |   -----        |
     |-- LGERROR --   |
     |   -----        |
     |-- LGBOT ----   |
     |   -----        |
     |-- LGEOT ----   |
     |   -----        |
     |-- LGSECURE -   |
     |   -----        |
     |-- LGFAULT --   |
     |   -----        |
     |-- LGUNABLE -   |
     |   -----        |
     |-- LGSABORT -   |
     |   -----        |
     |-- AUTOINFO -   |
     |   -----        |
     |-- LGSPO ----   |
         ----
```

LGSTA COMMAND

This command causes the designated station(s) to be initiated as a CANDE LOGSTATION.

```
-- ? --- LGSTA -- <station list> ---------------------------------------|
         --                      |                    |
                                 |- : ---- ALL -------|
                                        |  ---        |
                                        |- <option list> -|
```

?LGSTA

This form of the command is only valid from a control station; it is not valid from the main console. It will cause the station from which the message originates to be initiated as a CANDE LOGSTATION. If the station is already a LOGSTATION, the error message "#<lsn> ALREADY A LOGSTATION" will be displayed. All logging functions for the station will be set to FALSE.

Examples:

```
?LG
#T17 LG OPTIONS: -=AUTOINFO -=LGFAULT -=LGSABORT -=LGUNABLE
     -=LGSPO -=LGSECURE -=LGEOT -=LGBOT -=LGERROR -=LGCHARGE
     -=LGOFF -=LGON -=LGATTACH

?LG
# 13 IS ALREADY A LOGSTATION
```

?LGSTA : <option list>

This form of the command is only valid from a control station; it is not valid from the main console. It will make the station from which the command is initiated a LOGSTATION. If the station is already a LOGSTATION, an error message will be displayed. The logging bits for the specified option(s) will be set. The absence of an option will cause it to be set FALSE.

Example:

```
    ?LG:LGON,LGOFF,LGCHARGE
    #T17 LG OPTIONS: -=AUTOINFO -=LGFAULT -=LGSABORT -=LGUNABLE
         -=LGSPO -=LGSECURE -=LGEOT -=LGBOT -=LGERROR +=LGCHARGE
         +=LGOFF +=LGON -=LGATTACH
```

?LGSTA <station list> This form of the command is valid from the main console and a control station. It will make the designated station(s) a LOGSTATION and set all logging bits FALSE. If no such station exists, or the station is already a LOGSTATION, an error message will be displayed.

Examples:

```
    ?LGSTA T17,8
    #T17 LG OPTIONS: -=AUTOINFO -=LGFAULT -=LGSABORT -=LGUNABLE
         -=LGSPO -=LGSECURE -=LGEOT -=LGBOT -=LGERROR -=LGCHARGE
         -=LGOFF -=LGON -=LGATTACH
    #T12 LG OPTIONS: -=AUTOINFO -=LGFAULT -=LGSABORT -=LGUNABLE
         -=LGSPO -=LGSECURE -=LGEOT -=LGBOT -=LGERROR -=LGCHARGE
         -=LGOFF -=LGON -=LGATTACH
```

?LGSTA <station list> : <option list>
This form of the command is valid both from a control station and from the main console. It will cause the station(s) specified to be made a LOGSTATION. If no such station exists, or the station is already a LOGSTATION, an error message will be returned. The logging functions specified in the <option list> will be set. The absence of an option will cause it to be set FALSE.

Examples:

```
    ?LG T17:LGERROR,LGFAULT
    #T17 LG OPTIONS: -=AUTOINFO +=LGFAULT -=LGSABORT -=LGUNABLE
         -=LGSPO -=LGSECURE -=LGEOT -=LGBOT +=LGERROR -=LGCHARGE
         -=LGOFF -=LGON -=LGATTACH
```

It should be noted that only a control station may create a CANDE LOGSTATION; however, the station being made a LOGSTATION need not be a control station. It should be emphasized that a non-control CANDE LOGSTATION cannot terminate itself as a LOGSTATION.

DSLGSTA COMMAND

This command causes the designated station(s) to be terminated as CANDE LOGSTATIONs.

```
-- ? --- DSLGSTA --------------------------------------------------|
         ----       |                  |
                    |- <station list> -|
                    |                  |
                    |-- ALL -----------|
                        ---
```

?DSLG This form of the command is valid from a control station only; it is not valid from the main console. It terminates the station from which the message originates as a LOGSTATION. If the originating station is not a LOGSTATION, an error message will be displayed.

Example:

```
    ?DSLG
    #T17(13) DISCONTINUED AS LOGSTATION.
```

?DSLGSTA <station list> This form of the command is valid from a control station or the main console. It terminates the designated station(s) as a CANDE LOGSTATION.

Examples:

```
    ?DSLG 8,13
    #T12(8) DISCONTINUED AS LOGSTATION
    #T17(13) DISCONTINUED AS LOGSTATION
```

?DSLG ALL This command is valid from both a control station and and the main console. It terminates all CANDE LOGSTATIONs.

Example:

```
    ?DSLG ALL
    #ALL LOGSTATIONS DISCONTINUED
```

LGOP COMMAND

The purpose of this command is to furnish the capability of interrogating, setting or resetting any or all of the logging functions for the specified LOGSTATION(s).

```
-- ? --- LGOP ------------------------------------------------------------|
         ----   |     |  |                  |  |                         |
                |- + -|  |- <station list> -|  |- : -- <option list> -|
                |     |  |                  |
                |- - -|  |-- ALL -----------|
                            ---
```

```
?LGOP
?LGOP : <option list>
?LGOP-
?LGOP+
```

The above forms of the LGOP command are for interrogation of the logging functions of the station from which the message originates. These forms may be used from both a control and non-control station. None are valid from the main console. In each case, if the station from which the message originates is not a CANDE LOGSTATION, an error message will be displayed. "?LGOP" displays the complete list of logging functions with their current value; i.e., SET or RESET. "?LGOP : <option list>" displays the current setting for the options specified by <option list>. "?LGOP-" and "?LGOP+" display the list of RESET and SET options, respectively.

Examples:

```
?LG:ALL
#T17 LG OPTIONS: +=AUTOINFO +=LGFAULT +=LGSABORT +=LGUNABLE
     +=LGSPO +=LGSECURE +=LGEOT +=LGBOT +=LGERROR +=LGCHARGE
     +=LGOFF +=LGON +=LGATTACH
?LGOP
#T17 LG OPTIONS: +=AUTOINFO +=LGFAULT +=LGSABORT +=LGUNABLE
     +=LGSPO +=LGSECURE +=LGEOT +=LGBOT +=LGERROR +=LGCHARGE
     +=LGOFF +=LGON +=LGATTACH
?LGOP:LGFAULT
#T17 LG OPTIONS: +=LGFAULT
?LGOP -
#T17 LG OPTIONS RESET: [NONE]
?LGOP +
#T17 LG OPTIONS SET: AUTOINFO LGFAULT LGSABORT LGUNABLE
     LGSPO LGSECURE LGEOT LGBOT LGERROR LGCHARGE LGOFF
     LGON LGATTACH
```

```
?LGOP - : <option list>
?LGOP + : <option list>
```

Each of the above LGOP commands is valid from a control station only; neither is valid from the main console. If a minus (-) is used, the list of options specified will be reset. If a plus (+) is used, the options will be set. If the station is not a LOGSTATION, an error message will be displayed.

Examples:

```
?LGOP-:LGBOT
#T17 LG OPTIONS RESET: LGBOT
?LGOP
#T17 LG OPTIONS: +=AUTOINFO +=LGFAULT +=LGSABORT +=LGUNABLE
     +=LGSPO +=LGSECURE +=LGEOT -=LGBOT +=LGERROR +=LGCHARGE
     +=LGOFF +=LGON +=LGATTACH
?LGOP+:LGBOT
#T17 LG OPTIONS SET:LGBOT
```

```
?LGOP<station list>
?LGOP<station list>:<option list>
?LGOP- <station list>
?LGOP+ <station list>
```

The above forms of the LGOP command are for interrogation of the logging functions of the station(s) specified by <station list>. These forms are valid from a control station and the main console. In each case, if any station specified is not a LOGSTATION, an error message will be displayed. "? LGOP<station list>" displays the complete list of logging functions with their current value for the designated station(s). "? LGOP <station list:<option list>" displays the current setting of the option(s) specified by <option list> for the station(s) specified. "? LGOP- <station list>" and "? LGOP+ <station list>" display the list of RESET and SET options respectively for the station(s) specified.

Examples:

```
    ?LGOP T12
    #T12 LG OPTIONS: +=AUTOINFO -=LGFAULT +=LGSABORT -=LGUNABLE
         +=LGSPO -=LGSECURE +=LGEOT +=LGBOT -=LGERROR -=LGCHARGE
         -=LGOFF -=LGON +=LGATTACH
    ?LGOP T17:LGON
    #T17 LG OPTIONS: -=LGON
    ?LGOP-T12
    #T12 LG OPTIONS RESET: LGFAULT LGUNABLE LGSECURE LGERROR
         LGCHARGE LGOFF LGON
    ?LGOP+T12
    #T12 LG OPTIONS SET: AUTOINFO LGSABORT LGSPO LGEOT LGBOT
         LGATTACH
```

```
?LGOP - <station list>:<option list>
?LGOP + <station list>:<option list>
```

The above forms of the LGOP command are valid from a control station or a main console. If a minus (-) is used, the list of options specified will be reset for the designated station(s). If a plus (+) is used, the options will be set. In all cases, if a station is not a LOGSTATION, an error message is displayed.

Examples:

```
    ?LGOP T17
    #T17 LG OPTIONS: +=AUTOINFO -=LGFAULT +=LGSABORT -=LGUNABLE
         +=LGSPO -=LGSECURE +=LGEOT +=LGBOT -=LGERROR -=LGCHARGE
         -=LGOFF -=LGON +=LGATTACH
    ?LGOP-T17:LGSPO
    #T17 LG OPTIONS RESET: LGSPO
    ?LGOP T17
    #T17 LG OPTIONS: +=AUTOINFO -=LGFAULT +=LGSABORT -=LGUNABLE
         -=LGSPO -=LGSECURE +=LGEOT +=LGBOT -=LGERROR -=LGCHARGE
         -=LGOFF -=LGON +=LGATTACH
    ?LGOP+T17: LGSPO
    #T17 LG OPTIONS SET: LGSPO
    ?LGOP T17
    #T17 LG OPTIONS: +=AUTOINFO -=LGFAULT +=LGSABORT -=LGUNABLE
         +=LGSPO -=LGSECURE +=LGEOT +=LGBOT -=LGERROR -=LGCHARGE
         -=LGOFF -=LGON +=LGATTACH
```

```
?LGOP ALL
?LGOP ALL: <option list>
?LGOP- ALL
?LGOP+ ALL
```

Each of the above LGOP commands is valid from a control station or the main console, and is for the interrogation of all CANDE LOGSTATIONS. The presence of an option list will cause only the specified options to be displayed. The presence of a plus (+) or minus (-) will cause only the set or reset options to be displayed for all LOGSTATIONS.

Examples:

```
    ?LGOP ALL
    #T17 LG OPTIONS: +=AUTOINFO -=LGFAULT +=LGSABORT -=LGUNABLE
         +=LGSPO -=LGSECURE +=LGEOT +=LGBOT -=LGERROR -=LGCHARGE
         -=LGOFF -=LGON +=LGATTACH
    #T12 LG OPTIONS: +=AUTOINFO -=LGFAULT +=LGSABORT -=LGUNABLE
         +=LGSPO -=LGSECURE +=LGEOT +=LGBOT -=LGERROR -=LGCHARGE
         -=LGOFF -=LGON +=LGATTACH
    ?LGOP- ALL
    #T17 LG OPTIONS RESET: LGFAULT LGUNABLE LGSECURE LGERROR
         LGCHARGE LGOFF LGON
    #T12 LG OPTIONS RESET: LGFAULT LGUNABLE LGSECURE LGERROR
         LGCHARGE LGOFF LGON
```

```
?LGOP- ALL: <option list>
?LGOP+ ALL: <option list>
```

These commands are valid from a control station or the main console. They will cause the specified options to be set (+) or reset (-) for all CANDE LOGSATIONS. If there are no LOGSTATIONS, an error message will be displayed.

Examples:

```
    ?LGOP-ALL:AUTOINFO,LGSPO
    #T17 LG OPTIONS RESET: AUTOINFO LGSPO
    #T12 LG OPTIONS RESET: AUTOINFO LGSPO
    ?LGOP ALL
    #T17 LG OPTIONS: -=AUTOINFO -=LGFAULT +=LGSABORT -=LGUNABLE
         -=LGSPO -=LGSECURE -=LGEOT -=LGBOT -=LGERROR -=LGCHARGE
         -=LGOFF -=LGON +=LGATTACH
    #T12 LG OPTIONS: -=AUTOINFO -=LGFAULT +=LGSABORT -=LGUNABLE
         -=LGSPO -=LGSECURE -=LGEOT -=LGBOT -=LGERROR -=LGCHARGE
```

```
-=LGOFF -=LGON +=LGATTACH
```

OP COMMAND

The ?RO and ?SO commands have been replaced by the ?OP command. ?OP will allow the interrogation, setting or resetting of any or all CANDE options.

```
-- ? -- OP ----------------------------------------------------------|
             |     | |                                |
             |- + -| | |<-------- , --------|  |
             |- - -| |------- SWAPALL ----------|
                            ----
                       |-- ALLLOGIN ----|
                            ----
                       |-- DIALLOGIN ---|
                            ----
                       |-- KEEPSTA -----|
                            ----
                       |-- ALLMSG ------|
                            ----
                       |-- CATDEFAULT --|
                            ----
                       |-- CATALOGOK ---|
                            ----
                       |-- DUMPOK ------|
                            ----
```

?OP
?OP-
?OP+

These commands are valid from a control station and the main console. "?OP" will cause a complete list of the CANDE options with their current settings to be displayed. "?OP-" and "?OP+" will cause the list of reset and set options respectively to be displayed.

Examples:

```
?OP-
#OPTIONS RESET: KEEPSTA DIALLOGIN ALLLOGIN CATDEFAULT
     CATALOGOK SWAPALL
?OP+
#OPTIONS SET: DUMPOK DOSWAPTO DOWAITGO ALLMSG
```

?OP- <option list>
?OP+ <option list>

These forms of the command are valid from the main console and from a control station. They will cause the designated option(s) to be set (+) or reset (-).

Examples:

```
?OP+
#OPTIONS SET: DUMPOK DOSWAPTO DOWAITGO
?OP+SWAPALL
#OPTIONS SET: SWAPALL
?OP+
#OPTIONS SET: DUMPOK DOSWAPTO DOWAITGO SWAPALL
```

The logging options LGFAULT, LGUNABLE, LGSABORT and LGSPO have been added to the list of available logging functions.

LGFAULT cause CANDE faults to be noted at any LOGSTATION with the option set.

LGUNABLE note CANDE service unavailable at any station with the option set.

LGSABORT cause any abnormal termination of a CANDE SCHEDULE session to be reported at any LOGSTATION with the option set.

LGSPO cause any "?SS" messages addressed to the ODT to go to all LOGSTATIONs with the option set.

The ?WHERE (without a usercode) command will report both the control station and LOGSTATION characters (i.e., "C" and "L") for any station that is a control or LOGSTATION.

D2672 CANDE - "SCHD" MESSAGE

When a CANDE task is scheduled by the system, the SCHD message will now include the mix number of the scheduled task.

D2673 CANDE - "PUBLIC" ABBREVIATION

PUB will now be recognized as an abbreviation for PUBLIC in a SECURITY command.

D2674 CANDE - DATA FILES UPDATE

The implementation of CANDE is such that if a sufficient number of changes, none requiring an update, is made to the workfile, more than one update may be required to accomplish a complete update of the file; i.e., to produce a file that represents all outstanding workfile changes. In the case where the workfile is either type DATA or CDATA, this sometimes results in an incorrect update. Now, CANDE will discard any outstanding workfile changes that cannot be made with one update of the file and will display the following error message:

"DATA LOST, LIMITED CHANGES ALLOWED TO DATA FILES"

To prevent this from occurring, it is suggested that a periodic update of a DATA or CDATA file be performed using the UPDATE verb.

D2675 CANDE - RESEQUENCE OF "CANDE"

The CANDE symbolic has been resequenced.

D2679 CANDE - "WHO" CONTROL COMMAND

The ?WHO control command allows an operator to easily determine the user of a particular station. If someone is using the station being queried, the information returned is in the same format returned for a blanket ?WHERE control command. If no one is logged on at that station, a message to that effect is returned. The ?WHO command is only valid from a control station.

Syntax:

```
                 |<------------------|
                 |                   |
                 |       |<- , -|    |
                 |                   |
-- ? -- WHO -----<lsn range>---------|
                  |                |
                  |-<dls>----------|
                  |                |
                  |-<station name>-|
```

Example:
?WHO 15, 100
#15 NOT ON
1998 LIST JONES ON TTY23(100)

D2740 CANDE - ACCESSCODE

Two new commands have been added to CANDE: ACCESS and APASSWORD.

```
-- ACCESS ---------------------------------------|
   ----        |                                |
               |- . ----------------------------|
               |                                |
               |-<accesscode>-------------------|
                              |                 |
                              |-------<password>-|
                              |      |
                              |- / -|
```

"ACCESS" will return the current accesscode for the session (not including the password).

"ACCESS ." will assign a null accesscode to the session.

"ACCESS <accesscode>/<password>" will assign or change the session accesscode after validation in the USERDATAFILE.

Change of accesscode to a session is logged.

```
-- APASSWORD ---------------------------------------------|
   ----         |                                        |
                |-<old password>-------------------------|
                            |     | |                    |
                            |- = -| |-<new password>-----|
```

APASSWORD allows the password to be changed for the current accesscode of a session. If "APASSWORD" is entered, CANDE will prompt through the rest of the command, as follows:

    ENTER CURRENT ACCESSCODE PASSWORD PLEASE

No APASSWORD request may be entered from a CANDE SCHEDULE session.

When logging onto CANDE, CANDE will check the USERDATAFILE to see if the usercode has the ACCESSCODENEEDED bit set (TRUE). If the ACCESSCODENEEDED bit is set, CANDE will ask the following:

    ENTER ACCESSCODE PLEASE.

If no password is given (the password may be "." here, as for the USERCODE PASSWORD command), CANDE will ask for a password (using blotting or other special security measures, if needed, as for the USERCODE PASSWORD command). The accesscode/password is validated in the USERDATAFILE. If the accesscode is not valid, CANDE will send the following:

    INVALID ACCESSCODE/ACCESSCODEPASSWORD; ENTER ACCESSCODE PLEASE

Logon will not be complete until a valid accesscode is supplied.

"HELLO" will abort the logon attempt. In this case, CANDE will respond with the following:

    ENTER USERCODE PLEASE

When MAKEing a file, it may be TYPEd to "CONTROLLED" in the same manner in which a "GUARDED" file is typed, as follows:

    MAKE A/B ALGOL: CONTROLLED GUARDFILE

The security of a file may be changed to "CONTROLLED" in the same manner in which it may be changed to "GUARDED".

To change the security from "CONTROLLED" to any other security, read/write access to the file is required.

The new form for the SECURITY command is as follows:

```
-- SECURITY -------------------<filename>------------------------------>
   ---       |                |
             |- SOURCE -------|
             |                |
             |- OBJECT -------|
                        |     |
                        |- $ -|

>--- GUARDED ------<guard file name>------------------------------------|
  |  -----       |                   |
  |- CONTROLLED -|                   |
  |  ----                            |
  | |<--------------------|          |
  | |                     |          |
  |-----/1\--- PRIVATE --------------|
      |     |  --       |
      |     |- PUBLIC --|
      |                 |
      |-/1\--- SECURED -|
            |  ---      |
            |- IN ------|
            |           |
            |- OUT -----|
            |           |
            |- IO ------|
```

If an accesscode has read-only access (IN) to a file under its own usercode, doing a GET on the file will result in an unnamed workfile whose source is that file.

CANDE will store the current accesscode of a CANDE session at the time of the creation of a workfile (GET/MAKE) in the tankfile for each user. At recovery time, if the accesscode is not null, the accesscode on the session must be the same as in the tankfile in order to recover a workfile. If an attempt is made to recover a workfile with a different accesscode, the following error message will be sent:

    INCOMPATIBLE ACCESSCODE

The recovery file will not be recovered or purged. If the recovery file has an accesscode, it will be indicated in the list of recovery files as follows:

# Accesscode is different from the session's accesscode.

* Accesscode is the same as the session's accesscode.

CANDE will recognize the ACCESSCODE task attribute. For example, assume a session is running under usercode "UA" with accesscode "AA/PA". The following is entered:

```
    RUN (UB)P ON HISPACK; AC AB/PB
```

"AB" will be the accesscode for the task "(UB)OBJECT/P". At task startup, "AB/PB" must be a valid accesscode/password for user "UA"; if not, the task will not be run. Access to the program file "(UB)OBJECT/P" will be determined by the security of the program code file; if "GUARDED" or "CONTROLLED", by the usercode "UA" and the session's accesscode "AA". Supplying an ACCESSCODE task attribute on a task does not change the session's accesscode. The ACCESSCODE task attribute cannot be provided as a task attribute to a program being compiled from CANDE, as follows:

```
    "C; C AC=A/B" is valid
    "C; AC=A/B"   is not valid
```

Accesscodes are only required on tasks that access files which are protected by a guardfile which uses accesscodes to control access rights.

Any CANDE SCHEDULE sessions will inherit the accesscode of the CANDE session. The ACCESS and APASSWORD commands are not valid within a CANDE SCHEDULE session. If a SCHEDULE session contains an ACCESS command with an invalid accesscode, the SCHEDULE session will be aborted.

CANDE will include the accesscode in all log records about a session, in particular logon and logoff records.

D2827 CANDE - "?AT" CONTROLLER KEYIN

The "?AT" controller keyin is now available in CANDE.

Syntax:

```
-- ? -- AT --<text>--%
```

As a result of this implementation, the minimum abbreviation for the "?ATTACH" control command is now three characters.

Syntax:

```
                   |<-------- , --------|
                   |                    |
-- ? -- ATTACH -----<lsn range>---------|
        ---         |                  |
                    |-<dls>------------|
                    |                  |
                    |-<station name>---|
```

D2871 CANDE - "FILEORGANIZATION"

In the CANDE Reference Manual (Form No. 5010259), Page 4-40, the syntax diagram for LFILES should be changed to read as follows:

```
-- LFILES ------------------------------------------------------------>
   ----      |                  | |                   |
             |-<filename>------| |- ON <familyname> -|
             |                 |
             |-<directoryname>-|

>----------------------------------------------------------------------|
  |                                 | |                    |
  |- : -----------------------------| | |<---------------| |
       |                            | | |                | |
       | |<-----------------------| | |--- ; <modifier> ---|
       | |                        | |
       |-----<depth>----------------|
              |                   |
              |- ABBREVIATED ------|
              |  ---
              |- ALL --------------|
              |
              |- AREAS ------------|
              |  ----
              |- AREASIZE ---------|
              |  -----
              |- BLOCKSIZE --------|
              |  --
              |- CATALOGUE --------|
              |  ---
              |- CREATIONDATE -----|
              |  ---
              |- CRUNCHED ---------|
              |  ---
              |- CYCLE ------------|
              |  --
              |- DOUBLE -----------|
              |  --
              |- FILEKIND ---------|
              |  -----
              |- FILEORGANIZATION -|
              |  -----
              |- FILETYPE ---------|
              |  -----
              |- INTMODE ----------|
              |  --
              |- LASTACCESSDATE ---|
              |  -----
              |- LASTRECORD -------|
              |  -----
              |- MAXRECSIZE -------|
              |  --
              |- MINRECSIZE -------|
              |  --
              |- PRINTER ----------|
              |  --
              |- SAVEFACTOR -------|
              |  --
              |- SECURITY ---------|
              |  --
              |- TIMESTAMP --------|
              |  --
              |- UNITS ------------|
              |  -
              |- VERSION ----------|
                 -
```

On Page 4-41 add the following to the list of file attribute options: FILEORGANIZATION.

D2916 CANDE - ADD "HOSTNAME" AS TASK MODIFIER

The ability to run (compile, execute, etc.) a task on another host has been implemented.

The following should be added to the definition of <modifier> in the CANDE Reference Manual (Form No. 5010259), Page 3-4:

```
-- HOSTNAME -- = --<host identifier>--|
```

"If the host cannot be reached, CANDE will return the message "HOST NOT REACHABLE"."

CANDE

P1998 CANDE - HUNG SWAP JOB

When the System DSs a CANDE task that is running in swapspace due to an unrecoverable I/O error, CANDE will now report this to the terminal with the message "TASK HUNG ON SWAPDISK DUE TO IRRECOVERABLE I/O ERROR".

P1999 CANDE - SCHEDULE FAULT CORRECTED

A problem has been corrected where CANDE would get a DIVIDE BY ZERO fault if a schedule session were initiated and a problem was encountered opening the schedule output file. This would only happen if a prior schedule session had run to completion and an internal CANDE resource were reused.

P2000 CANDE - "MATCH" VERB CORRECTION

When a MATCH of two files is attempted and the newfile is not present, CANDE will now display the name of the newfile; e.g.,

```
MATCH A TO B
# NO FILE: B
```

P2292 CANDE - "REPLACE" IN SEQUENCE NUMBER FIELD

A problem has been corrected where CANDE would allow the following syntax of the REPLACE verb to include the sequence number range of the workfile:

```
@ <start column> - <end column>
```

This would only occur when the MAKE verb had been used to create the workfile and the workfile had never been updated.

Example:

```
MAKE EXAMPLE ALGOL
100 LINE #1
REPL FIRST LIT /00//AA/ @ 73-80:t
```

P2293 CANDE - "REPLACE" ON "ID" FIELD

A REPLACE verb done of the ID field of the workfile will now work correctly on all single line entries succeeding the first.

P2294 CANDE - "INSERT AT END" ON EMPTY WORKFILE

If the following INSERT syntax is used:

```
INSERT <file title> AT END
```

and the workfile is empty, CANDE displays the error message "EMPTY WORKFILE"; however, the workfile is left in such a state that it was possible to get the following CANDE errors on subsequent LIST commands on that workfile:

```
ORPHAN
XSBUF
```

This problem has been corrected.

P2553 CANDE - "VALUE" SIGN ON "EXECUTE"

A plus sign following VALUE clause on a CANDE EXECUTE was being treated as a minus sign. A plus sign in this context will now be handled correctly.

CARDLINE

P1886 CARDLINE - JOB CARDS TRANSFERRED TO DISK

CARDLINE will now allow the transfer of job cards to disk (TASKVALUE=10). The maxrecsize of the output file will be as requested in the file LINE equate statement. Previously, a 14-word maxrecsize was always used.

COBOL

D2676 COBOL - INSPECT ALGORITHM MEETS "ANSI74" SPECIFICATION

The comparison cycle algorithm used for INSPECT statements has been changed slightly. This change only affects INSPECT statements having multiple tallying operands. (The tallying operand is the identifier designated as IDENTIFIER-2 in the syntax for the INSPECT statement on Page B-23 of the COBOL Reference Manual, Form No. 5001464.)

Previously, such statements were treated as if they were separate INSPECT statements, with the full comparison cycle applied to each tallying operand even though a match may have occurred on a prior tallying operand. This was incorrect according to the ANSI74 standard.

Now, the comparison cycle encompasses all tallying operands in the INSPECT statement, with a match causing subsequent tallying operands to be made ineligible for matching until the next cycle.

An example is necessary to illustrate this delicate distinction. Assume the identifier "I" is 3 characters long and has a value of "AAA":

```
INSPECT I TALLYING
     X FOR ALL "A"
     Y FOR ALL "A".
```

Previously, both tallying operands, X and Y, were incremented by 3. Now, only X is incremented by 3.

D2683 COBOL - DOLLAR OPTION "BINARYCOMP"

A new dollar option, $BINARYCOMP, has been provided to override the default interpretation of computational data items as 4-bit quantities when the B2500 option is set; i.e., when BINARYCOMP is set in conjunction with B2500, computational items will be maintained in internal binary format. The default setting is RESET.

The setting of the option must appear before the IDENTIFICATION DIVISION and may not be modified subsequently.

The option may be invoked at compile time via a dollar card or by recompiling the COBOL compiler with the BINARYCOMP option set to establsh a different default.

D2816 COBOL - "NOTE" VERB EXTENSIONS

The NOTE construct has been extended to be compatible with the Medium Systems COBOL. The following formats are now acceptable.

PARAGRAPH (starting in Margin A) NOTE
1. <procedure-name> SECTION. NOTE <comment>
2. <procedure-name> .NOTE <comment>
3. NOTE <comment>

SENTENCE (starting in Margin B) NOTE
1. NOTE <comment> .

D2839 COBOL - LIBRARY FACILITY IMPLEMENTED

The Library facility has been implemented in COBOL and COBOL74. As documentated in Appendix B, User Interface to Libraries, all COBOL programs are now made library-capable and may use libraries via the CALL statement. The EXIT PROGRAM statement has been expanded to cause a library to return to its caller if it indeed was called as a library.

D2850 COBOL - "B7700 AUDIT" FEATURE

AUDIT is now a reserved word to support the B7700 AUDIT feature.

D2852 COBOL - SEPARATORS

In the discussion of the separators comma and semicolon on Page 7-6 of the COBOL Reference Manual (Form No. 5001464), Rule (e) is incorrect and should be replaced by Rules (e) and (f), as follows:

"e. Semicolons and commas may appear only where shown in the syntax formats and may not be used interchangeably.

f. Semicolons may be used only in the following places:

1. Between statements
2. In a conditional statement:
   (a). Between the condition and statement-1
   (b). Between statement-1 and ELSE"

D2854 COBOL - "MOVE" STATEMENT

The method used by the COBOL compiler to compile MOVE statements with multiple receiving fields is incorrect and is in conflict with the correct method specified in the COBOL Language Reference Manual (Form No. 5001464) on Page 7-74 and in the American National Standard 1974 version of the COBOL Language. Consider the following statement:

```
    MOVE A(I) TO I,X.
```

Currently, the effect of this statement is the same as the two following consecutive statements:

```
    MOVE A(I) TO I.
    MOVE A(I) TO X.
```

This is incorrect, since the subscript for the source is re-evaluated for the second move, and the value of I may be different.

The source expression should be evaluated only once, with the MOVE operations being the following:

```
    MOVE A(I) TO <temporary>
    MOVE <temporary> TO I
    MOVE <temporary> TO X
```

The correct method will be implemented on a future release. MOVE statements with multiple receiving fields which could produce incorrect side effects should be avoided.

D2872 COBOL - "TIME" AND "COMPILETIME" FUNCTIONS

The following text should be added to the note on Page 2-14 of the COBOL Reference Manual (Form No. 5001464):

> "It is advisable to move TIME and COMPILETIME functions to numeric or numeric edited receiving fields to ensure decimal point alignment. When these functions are moved to alphanumeric receiving fields, different move rules apply and frequently cause unexpected results."

D2873 COBOL - SORT VARIABLE LENGTH RECORDS

The COBOL Reference Manual (Form No. 5001464), Page 7-120, should be corrected as follows:

> "(b) The input files to be merged may have either fixed- or variable-length records."

D2874 COBOL - "OCCURS DEPENDING" OPTION

Several corrections have been made with regard to the use of the DEPENDING option of the OCCURS clause when the ANSI74 dollar option is set. In ANSI74 COBOL, an OCCURS...DEPENDING clause causes all group items which span the occurring item to be considered as variable length items whose length depends indirectly on the number of valid occurrences at the time of the reference. This situation conflicts with any SIZE...DEPENDING clause on any of these group items and will now be given a syntax error. Further, three other restrictions of ANSI74 COBOL will also be enforced:

1. No group item spanning an OCCURS...DEPENDING item may itself have an OCCURS clause.

2. The only data items that may be declared in the structure following the OCCURS...DEPENDING clause are those which are subordinate to the item having the OCCURS clause.

3. The minimum number of valid occurrences is one (1). Setting the depending item to zero will cause an INVALID INDEX interrupt if either a group item spanning the occurring item or one of the subordinate data items to the occurring item is referenced.

In addition, DISPLAY statements referencing variable length data items will now display only the appropriate number of characters, rather than the appropriate number of characters followed by spaces representing the unused character positions.

D2893 COBOL - "DMS" EXCEPTION HANDLING

Page 6-1 of the DMS II Host Reference Manual (Form No. 5001498) states that the entire DMSTATUS register may be referenced.

This is incorrect. Only the individual fields within the register may be referenced by specifying a DMSTATUS attribute name in parentheses following the word DMSTATUS.

D2921 COBOL - "DATE-COMPILED" CLAUSE HAS PERIOD

The period for the DATE-COMPILED clause is now allowed in columns 59-70.

D2922 COBOL - EXCEPTION CODE FOR "ANSI74" INDEXED FILES

The exception handling code for indexed I/O files under the ANSI74 option has been improved.

D2923 COBOL - "MONITOR" OUTPUT

The MONITOR statement now observes the ANSI74 dollar option default advancing to AFTER, not BEFORE.

D2958 COBOL - SETTING OF SYSTEM COMPATIBILITY OPTIONS

The setting of system compatibility options are not mutually exclusive; i.e., setting a system option will not reset all other system options.

The only exception is setting the ANSI74 option using a dollar card containing ANSI74 option that is not preceded by an option indicator. In this case, the B6700 option is reset.

D2959 COBOL - FILE ATTRIBUTE "IORECORDNUM"

The file attribute IORECORDNUM in COBOL is implemented so that it counts relative to 1.

D2960 COBOL - COMPARISON OF NONNUMERIC OPERAND

The following should be added to Page 7-18 of the COBOL Language Reference Manual (Form No. 5001464) before the comparison of numeric operands:

> "Comparison of Non-Numeric Operand Versus Arithmetic Expression
> ---------- -- ----------- ------- ------ ---------- ----------
>
> A non-numeric data item cannot be used in a relation condition with an arithmetic expression."

D2962 COBOL - "OBJECT-COMPUTER" PARAGRAPH

Any text appearing in the OBJECT-COMPUTER paragraph which is not part of the MEMORY, DISK, SEGMENT-LIMIT, or PROGRAM COLLATING SEQUENCE clauses is considered as a comment.

The syntax diagram of the OBJECT-COMPUTER paragraph on Page 5-5 of the COBOL Reference Manual (Form No. 5001464) should be amended to include a comment entry.

COBOL

P1434 COBOL - PROGRAM COLLATING SEQUENCE

Several problems have been corrected regarding compares of data items against figurative constants when a program collating sequence clause was specified.

P1662 COBOL - EXCEPTIONCODE FOR "ANSI74" CORRECTED

Under the ANSI74 dollar option, a READ statement with an AT END clause for a file which had a USE procedure was leaving the I/O result on the stack for any exception other than an AT END. This has been corrected.

P1663 COBOL - SCALE NON-INTEGER MOVED TO "DM" FIELD

A non-integer numeric item moved to a DM field was not being scaled (i.e., integerized) as was expected by the DM field. The scaling now takes place.

P1664 COBOL - GRACEFUL EXIT FROM SYNTAXED "STRING" STATEMENT

When 'DELMITED' was misspelled in the STRING statement, the compiler would terminate with an END OF FILE NO LABEL. This error no longer causes the compiler to be DSed; instead, it continues on to a normal EOJ with a syntax error.

P1666 COBOL - "FILE-LIMIT" STATEMENT CAUSED "INVALID INDEX"

A bad range check allowed an INVALID INDEX to occur in the compiler when the FILE-LIMITS clause was mal-formed. The range check has been corrected so that the compiler continues after syntaxing the mal-formed FILE-LIMITS clause.

P1763 COBOL - USER INTRINSIC CALLS NOT SCANNING END PERIOD

On a CALL statement, when a user intrinsic is called with parameters by value, the period at the end of the statement was being skipped over. This could cause different logical structure of a nested IF statement than the structure that might have been intended. Scanning is now done properly.

P2033 COBOL - SPURIOUS ERROR MESSAGES

A problem has been corrected where an error message would refer to a previous correctly defined 88 level item.

P2035 COBOL - COMPILE TIME INFORMATION

The compiler will now print 4 digits before the decimal point to avoid truncation on long elapsed times in the trailer summary.

P2036 COBOL - HEX LITERAL COMPARES

The compiler will now emit proper code to compare large COMP-2 data items with single digit hex literals.

P2037 COBOL - "E" FORMAT SCANNER ERROR

A problem has been corrected where the COBOL compiler would misinterpret the string ".4ELSE". This would appear if the "4" appeared in column 72 and the "ELSE" on the following card.

P2038 COBOL - COMPARISON OF INDEX DATA NAMES

Several problems have been corrected in the handling of Index-Data name compares:

1. Index-names and Index-data names will now compare properly in all cases.
2. A syntax error will be given on compares between index-data-names with a data-name or a literal.
3. Subscripting of index-data-names will now be correctly compiled.

P2039 COBOL - "VALUE" CLAUSE

An illegal VALUE clause subordinate to an OCCURS clause gave the proper syntax error, but would cause compiler termination with INVALID INDEX. This has been corrected.

P2040 COBOL - IMPROVED STRING COMPARISONS

Improved code is now generated for comparing unequal length strings. The technique consists of breaking the comparison into two parts, thus allowing detection of early termination of the compare and reducing overhead by elimination of space fill on the shorter string. The improved code is generated for equal or not equal relationals.

P2041 COBOL - "E" FORMAT ON CONTINUATION CARD

If a constant in E-Format was split across a card boundary such that "E" was the first character of the continuation card and the exponent contained a "+" or "-", an invalid syntax error would result (e.g., ".4E.25" where the "4" and "E" were not on the same card). This has been corrected.

P2042 COBOL - DICTIONARY WRAPAROUND

The compiler will now ensure that an error is generated whenever its internal table limits are exceeded.

P2043 COBOL - DECLARATIVE EXECUTION

A problem has been corrected where both standard error and record size USE procedures were present. Certain tape read parity errors can result in multiple bits being on in the result descriptor, thus simulating a record size error. This could result in improper execution of the record size routine. In addition, if subsequent end of file processing was specified, it would be performed regardless of the actual end of file. Code is now emitted to ensure that the correct sequence of actions occur when the above situation arises.

P2044 COBOL - HEX LITERAL FIGURATIVES

The handling of hex literals for VALUE, IF and MOVE statements has been corrected. In particular, one character ALL figuratives now work correctly for all cases.

P2045 COBOL - SIMPLE LITERAL COMPARES

Correct code is now emitted for the following cases:

```
IF A>(-<literal>)
IF A>(<literal>)
```

P2046 COBOL - CORRECT STACK ESTIMATE

The stack estimate passed to the MCP has been updated to prevent double counting of local stack cells for IPC capable jobs.

P2047 COBOL - IMPROVED ERROR MESSAGE

The compiler will now give an error when the BLOCKSIZE file attribute exceeds 65535.

P2048 COBOL - CORE ESTIMATE OF SORTING PROGRAMS

The compiler will now correctly increment the core estimate for programs which specify the amount of memory in the SORT statement. If the memory specification is a constant, that value will be added to the estimate; otherwise, a value of 12,000 words will be used.

P2049 COBOL - "PICTURE" CLAUSE

For edited numeric and alphanumeric items, the PICTURE clause will now properly allow elements as described in pages 6-60 through 6-68 of the COBOL Language Manual (5001464 August 1977). In addition, when ANSI74 $ option is set, non-reserved picture characters are allowed as simple insert characters.

P2050 COBOL - MIXED GROUP MOVES

Correct code will now be generated for group moves when the sending and receiving fields usage differs and one operand in the MOVE is a two-dimensional array.

P2051 COBOL - "VALUE" CLAUSE FOR "COMP-4" ITEMS

Scaled decimal values in the VALUE clause of COMP-4 data items is now permitted.

Example:

```
02 A COMP-4 VALUE 123.45
```

P2052 COBOL - WARNING FOR VARYING SIZE RECORDS

A warning message will now be given if the B2500 option is set and there is no "RECORD CONTAINS" clause and all the record descriptions are not the same size for a given file description.

P2053 COBOL - "COPY" STATEMENT LISTING

The listing of COPY text has been improved when the COPY statement ends with a period in column 72. It also improves the matching of error text with the source line when syntax errors occur.

P2054 COBOL - CORRECT WARNING MESSAGE

A warning message is now issued whenever an index-data name is set to literal zero.

P2055 COBOL - REPORT WRITER CONTROL FOOTING

The compiler no longer improperly splits a CONTROL-FOOTING report group across a page boundary.

P2056 COBOL - CORRECT OFFSET LISTING

The correct offset will now be printed for LOCK and EVENT data-items when $OFFSET is used.

P2058 COBOL - NEGATIVE VALUED ATTRIBUTE MNEMONICS

The compiler will now generate the correct value of negative valued attribute mnemonics. The following example will now work correctly:

```
SET MYSELF(STATUS) TO VALUE(TERMINATED)
```

P2059 COBOL - "XREF"

COBOL will now XREF the occurrence of a file-name in the SELECT clause as being the point of declaration. Previously, the FD was noted as being the point of declaration.

P2060 COBOL - TWO DIMENSIONAL SUBSCRIPTING

An obscure problem has been corrected where the sending field operand of a group MOVE was a two dimensional array. If the receiving field usage differed from the sending field, an incorrect offset was generated for the sending field.

P2061 COBOL - "CLOSE" WITH "PURGE"

CLOSE with PURGE on a DIRECT switch file will now take the correct action. Previously, a regular CLOSE was done.

P2062 COBOL - "SELECTION" EXPRESSIONS

Correct code is now generated for the use of figuratives in SELECTION expressions.

Example:

```
FIND <set-name> AT <key> = SPACES
```

P2063 COBOL - WORD ALIGNED MOVES

The compiler will now generate improved indexing code whenever the sending or receiving field offset is aligned on a word boundary.

P2161 COBOL - "OPTIMIZE" OPTION

Incorrect code is no longer generated for PERFORM statements referencing paragraphs that have a small number of statements. The setting of the OPTIMIZE option causes such paragraphs to be compiled "in-line" at the site of the PERFORM statement, and in rare circumstances, incorrect code was being generated for certain statements.

P2177 COBOL - "COPY" IN "LD" SECTION

The compiler will now handle a COPY statement in the LD section.

P2187 COBOL - "TIME" FUNCTIONS

Incorrect results will no longer be obtained when using either the TIME(10), TIME(15) or TODAYS-DATE functions in a COMPUTE statement.

P2252 COBOL - STATISTICS NOT CORRECT FOR "STOP RUN"

Paragraphs containing a STOP RUN statement were not being clocked off before blockexit, thus distorting the statistics timings for that paragraph. A STOP RUN statement now correctly clocks off statistics code for its paragraph.

P2253 COBOL - "SELECT" STATEMENT SYNTAX ERRORS

A problem has been corrected where a syntax error in a SELECT statement with a FILE LIMITS clause could cause compiler termination with an INVALID INDEX.

P2254 COBOL - EXTERNAL "PROCEDURE" DECLARATIONS

The compiler no longer terminates with an integer overflow while attempting to compile external PROCEDURE declarations which are syntactically invalid.

P2255 COBOL - "UNITS" ATTRIBUTE

The compiler no longer unconditionally generates a VALUE(CHARACTERS) for the UNITS attribute in the file description for all COBOL files. UNITS is now set to VALUE(WORDS) if the INTMODE attribute is equal to words (the usage of the first record is COMPUTATIONAL).

P2256 COBOL - CODE SEGMENT SIZE EXCEEDED

A large paragraph causing a code segment greater than 4095 words would not be syntaxed if the size were so extremely large as to exceed 5460 words. This situation has been corrected.

P2257 COBOL - TOO MANY "REPORT" WRITER CODE CLAUSES

More than 15 code clauses in a REPORT section caused the compiler to get a SEG ARRAY error. This has been corrected.

P2258 COBOL - "INVALID INDEX" IN BAD "REPORT" WRITER

A REPORT writer data division item that does not end with a period caused the compiler to fault with an INVALID INDEX. This error has been corrected; a syntax error is now generated.

P2259 COBOL - "ANSI74 LINAGE" "WRITE AT END OF PAGE"

A file using the ANSI74 LINAGE clause and the WRITE AT END OF PAGE clause would get an INVALID OP fault because the compiler generated bad code. This error has been corrected.

P2260 COBOL - RANDOM FILE GETTING SERIAL WRITE ERRONEOUSLY

A random access file written with a WRITE statement not having an INVALID KEY clause would erroneously get serial write action instead of random write action under the ANSI74 dollar option. This error has been corrected.

P2295 COBOL - EXCEPTION CODE ERRONEOUSLY CHECKING DATA BIT

The code generated for exception handling on an INVALID KEY clause was erroneously checking the data bit result instead of the EOF bit. Correct code is now generated which checks the EOF bit.

P2313 COBOL - INCORRECT "SORT" STATEMENT

SORT statements which failed to include one of the required reserved words MODULES, WORDS or CHARACTERS on the disk size clause would not receive a syntax error. Now, an appropriate syntax error is generated in this situation.

P2314 COBOL - "LOCK" STATEMENTS

Correct code is now generated for LOCK statements referencing COMP-1 data items which are both global and reference parameters.

P2315 COBOL - "INVALID INDEX" IN LARGE CODE SEGMENT

The compiler would fault with an INVALID INDEX when a code segment was large enough to be near 4095 words. The compiler will no longer get an INVALID INDEX; instead, it will create the code segment so long as it does not exceed 4095 words.

P2316 COBOL - MAXIMUM FILES ALLOWED

The maximum number of files allowed to be declared in a COBOL program has been increased from 99 to 127.

P2317 COBOL - INFO TABLE OVERFLOW

The compiler no longer occasionally terminates with an INVALID INDEX because the maximum number of info table entries was exceeded.

P2318 COBOL - "XREF"

Sequence number indications have been improved for references to receiving fields in arithmetic statements.

P2319 COBOL - CONDITION NAMES

Erroneous syntax errors (error #19) are no longer occasionally produced for legitimate references to condition names associated with a COMP-2 filler item in a display array.

P2320 COBOL - MOVES OF NUMERIC LITERALS

Correct code is now generated for moving numeric literals whose value is greater than 10**11 to single-precision computational data items which are not "word-oriented" within a data structure.

P2338 COBOL - LARGE REPORT WRITER EXCEEDED CODE SEGMENT

The compiler failed to syntax a program with an excessively large code segment in the Report Writer section. A syntax error will now be given for code segments which are too large in the Report Writer section.

P2339 COBOL - ANALYZE OPTION DUPLICATING LINES

Information from a previous line of output from the ANALYZE option was being duplicated on the listing. This problem has been corrected.

P2340 COBOL - NEGATIVE ZEROS IN "VALUE" CLAUSES

Incorrect non-zero values are no longer given to 4-bit character data items having a negative zero in the VALUE clause of their data description.

P2341 COBOL - "COBOL" NO LONGER IGNORES MISPLACED "NOT"

A statement of the form:

```
    IF <item> = NOT "ALL"
```

was compiled the same as

```
    IF <item> = "ALL"
```

Now, the compiler gives a syntax error.

P2342 COBOL - "SEARCH" NESTED IN "IF" STATEMENT

A SEARCH statement nested in an IF statement caused a logic flow problem when the OPTIMIZE dollar option was used. The ELSE part of the IF statement would be executed as well as the IF part on a complex IF-SEARCH-ELSE-IF construct. This worked correctly without the OPTIMIZE option set; now, that option also works correctly.

P2365 COBOL - LARGE RECORD DESCRIPTIONS

Subscripted 4-bit character data subordinate to an EBCDIC group item having an OCCURS clause and an offset (in terms of 8-bit characters) greater than 2**15-1 are no longer addressed incorrectly.

The following example declaration would have caused the elementary item "C" to be addressed incorrectly.

Example:

```
  01 TAB
     03  FILLER PIC X(50000).
     03  G OCCURS 10 TIMES.
         05 C COMP-2 PIC 9.
```

P2373 COBOL - CONDITION NAMES

The invalid use of a numeric literal value in the condition name declaration of a non-numeric conditional variable no longer causes the compiler to fault with an INVALID INDEX while attempting to compile PROCEDURE DIVISION references to the condition name.

The following example declaration would have cause the INVALID INDEX termination:

```
  77 X PIC XXX.
     88 LX VALUE 555.
```

P2377 COBOL - "INSPECT" VERB SLOW EXECUTION

When using the INSPECT verb in the same way as using EXAMINE, INSPECT ran with a "tally" speed of only 20 characters per second.

This has been improved; code generated by the compiler has been optimized.

P2380 COBOL - "SORT" DYNAMIC FILE ATTRIBUTES

SORT statements using files with dynamic attributes in the GIVING or USING clauses would cause the COBOL compiler to get an INTEGER OVERFLOW fault. The compiler will not get a fault; correct code will be generated.

P2381 COBOL - ERRONEOUS SYNTAX ERROR

When comparing a figurative constant against an arithmetic expression with subscripted variables, the compiler erroneously emitted a syntax error. The syntax error will no longer be emitted, and the comparison will be allowed.

COBOL74

D2839 COBOL74 - LIBRARY FACILITY IMPLEMENTED

The Library facility has been implemented in COBOL and COBOL74. As documentated in Appendix B, User Interface to Libraries, all COBOL programs are now made library-capable and may use libraries via the CALL statement. The EXIT PROGRAM statement has been expanded to cause a library to return to its caller if it indeed was called as a library.

D2875 COBOL74 - "COBOL74" IMPLEMENTATION

COBOL74 has been implemented, conforming to the COBOL ANSI74 Standard.

COBOL TABLEGEN

P1811 COBOLTABLE - "LISTACK" OPTION

When compiled with the LISTACK option, COBOL TABLEGEN now correctly produces a listing of the reserved word table stack heads.

COBOL74 TABLEGEN

D2983 COBOLTABLE74 - "COBOL74 TABLEGEN" IMPLEMENTATION

SYSTEM/COBOLTABLEGEN74 generates the text required for the declaration of reserved word defines and arrays in the COBOL74 compiler, obtaining its input from its own symbol file.

The format for each reserved word is as follows:

Col (inclusive)	Use
1-19	The actual spelling of the reserved word.
21-39	The ALGOL identifier by which the reserved word is known within the compiler. If blank, the ALGOL identifier is spelled the same as the reserved word.
41	Contains an "*" if a synonym for the previous reserved word.
43-45	Contains the subclass (SCF) value for this word. The defines for these values are in the symbol file for the COBOL compiler.
47	Contains the stackhead priority number. The larger the number the more quickly it will be found by the compiler's reserved word lookup routine.

Desired patches should be made in accordance with the record format described above.

The program should be compiled with "NEW" set. The ALGOL newtape file should be label-equated to the title of "SYMBOL/COBOLTABLEGEN74".

When the program is executed, it will:

Read the new SYMBOL/COBOLTABLEGEN74 under the title "SYMBOL/COBOLTABLEGEN74" with the file "GEN".

Create a patch to be used to create a new SYMBOL/COBOL74 under the title "PATCH/COBOLTABLEGEN74" with the file named "PATCH".

Produce a listing of the generated defines if compiled with "LISTUFF" set.

Produce a listing of the RSWD stackheads if compiled with "LISTACK" set.

COMPARE

P1887 COMPARE - USERCODE ATTACHED FILE TITLES

COMPARE will now handle file titles which include a USERCODE.

CONTROLLER

D2267 CONTROLLER - FOUR VS THREE DIGIT SYSTEM SERIAL NUMBERS

All system serial number fields have been expanded to allow four-digit numbers.

D2404 CONTROLLER - NEW DISPLAY OPTIONS

Qualifications have been added to the following ODT commands: A, JOB, MIX, S, W and their respective entries in the ADM. See GENERAL note D2535 for complete syntax and semantics.

D2571 CONTROLLER - DISALLOW "PRINTERLABELS" OPTION

The PRINTERLABELS option no longer exists. Option 24 is now the OKTIMEANDDATE option; see MCP note D2592.

D2594 CONTROLLER - REMOVE "DCKEYIN" RESTRICTIONS

DCKEYIN now allows all requests to be passed on to the CONTROLLER. The CONTROLLER now enforces any necessary restrictions.

D2637 CONTROLLER - REASON FOR QUEUE "DS"

The system will now display the reason for queue insertion failure.

This message is passed to the MCS if ORGWANTSMSG is TRUE. The message has the following format:

```
Word 0 .[47:8]  = 21 Controller result
       .[39:8]  =  6 Miscellaneous message
       .[31:8]  =  3 Queue reject notice
       .[14:15] = LSN
Word 1          = Length of message in characters
Words 2-N       = Text
```

D2940 CONTROLLER - LEADING BLANKS IN "ACCEPT" MESSAGES

ACCEPT messages (<mix number> AX) will not have leading blanks scanned off if the first non-blank character following the AX is a colon (:). In that case, the string following the colon is passed on to the program untouched. If the first non-blank character after the AX is not a colon, the message will be deblanked as before.

CONTROLLER

P1073 CONTROLLER - "ADMEVENT M"

ADMEVENT M now functions after a Halt/Load.

P1077 CONTROLLER - "PC" VS. "MPX3"

SC (or PC) no longer gives "PROGRAMMED H/L" and "STRINGS PRESENT" messages on a system with a Model 3 Multiplexor.

P1705 CONTROLLER - "REMOTESPO"

REMOTESPO no longer goes into a loop when it is DSed.

P1706 CONTROLLER - "NEXT" ON "RJE" PERIPHERAL STATUS REQUEST

The NEXT input message on an RJE RSC will now work correctly when more than one "PAGE" of output (as specified by the RSC's TERM description) is generated by a request for peripheral status.

P1765 CONTROLLER - "SQ" COMMAND CORRECTION

The SQ command no longer misses printing "NO ENTRIES" after printing the first non-empty queue.

P2308 CONTROLLER - "CONT6 GETDISK"

Occasionally, JOBFORMATTER would abort because a job file header was spread across two rows of JOBDESC. This problem has been corrected.

P2399 CONTROLLER - MESSAGE GREATER THAN "1896" CHARACTERS

A terminal sending a message with a length greater than 1896 characters (3/6 words) such that the MCS passes it to the CONTROLLER (through REMOTEINPUTQ) can cause a non-fatal dump by string protect. This problem has been corrected.

P2400 CONTROLLER - CORRECT "MQ- 1023"

Entering "MQ- 1023" (or any other number >30) will no longer hang the system.

P2440 CONTROLLER - "SCR" LENGTH

The length of input for the SCR command passed to SETSTATUS is now correct.

P2466 CONTROLLER - MNEMONIC IMPROVEMENT

Control card commands and PD commands from RJE are now allowed. The ADM mnemonic JOBS now work properly.

P2467 CONTROLLER - PRIVILEGED "MCS"

The CONTROLLER now recognizes an MCS as privileged when bit 45 in word 1 of the message is on. This gives the inquirer full ODT privileges.

The following restriction is applicable, however: If a usercode is included in the message that is designated from a privileged MCS, a "NOT ALLOWED" message will be returned.

P2489 CONTROLLER - "PRINTLABEL"

A possible dump for an empty line when PRODUCEALABEL calls HARDCOPY has been corrected.

P2525 CONTROLLER - "ODT" TIMOUT

The timeout for ODTs has been corrected.

P2541 CONTROLLER - "CU" VS. "HARDCOPY"

The CONTROLLER will no longer get an INVALID INDEX fault when hardcopy is running and a message of 0 length is queued for output.

P9217 CONTROLLER - "SQ NO ENTRIES"

When the CANDE command "?SQ" is entered and there are jobs in Q<n>, if the user's job is not in Q<n> it will show "NO ENTRIES". Previously, it would show a blank line.

D2903 DATACOM - STATION INTERROGATE DCWRITE EXTENSION

An extension to the "interrogate station environment" DCWRITE (TYPE=4) has been made to allow access to the station's NDL-declared terminal name. The name can be accessed by setting MSG[0].[31:1] to 1 in the interrogate station DCWRITE message. The terminal name will be returned via the interrogate station environment result (CLASS=15), which is similar to the mechanism used to return station names. In INX:=INDEXWORD.[47:8], there will be an index into the result message (MSG) for the terminal name. The name can be retrieved by scanning EBCDIC characters, starting in MSG[INX] until a terminating period (.) occurs. WARNING: INDEXWORD.[47:8] will contain zero for a SCHEDULE station or if a disk I/O error occurs when an attempt is made to read the NIF for the terminal name.

DATA COMMUNICATIONS

P2106 DATACOM - SEGMENTED ARRAY DISK "I/O"

Datacom I/O into and from segmented arrays now functions properly. DCINITIAL will no longer get an INVALID INDEX or INVALID OP fault while attempting to initialize a datacom system that has more than 1023 stations defined.

P2107 DATACOM - RECONTIMEOUT FAILURE

DCRECON's wait for reply loop has been rewritten to give it an adequate opportunity to get processor time, even in the heaviest load conditions, so that spurious system dumps by DCRECON should no longer occur.

P2108 DATACOM - "DCCONTROL" INVALID INDEX

DCCONTROL should no longer get an INVALID INDEX when a move/swap cluster DCWRITE is requested.

P2109 DATACOM - SET UP LINE CONTROL INDEX

DCRECON now properly sets up the new line descriptor and line table on a move/add/subtract DCWRITE request. The primary and auxiliary bits, [47:2], of the descriptor are no longer overwritten. Line tallys are properly cleared. The correct field is used for ensuring line control indices are valid.

P2397 DATACOM - RESTARTING DSED "DCP"

The MCP keeps internal statistics on each DCP, which remain in existence as long as the datacom tables are present. One piece of information is the fact that this DCP stack has been DSed. If that DCP is later restarted and the datacom tables are still present, there was one path through initialization which did not clear the DSed bit. That caused two problems: the message "DCP <n> RUNNING" would constantly appear on the ODT, and the MCP procedure DCINFOINT would get a fault and programdump when SYSTEM/DCSTATUS was run for that DCP. Both these problems have been corrected.

P2450 DATACOM - "DCP" TERMINATION

Two problems which could occur when a DCP terminated have now been corrected.

In the first, DCTERMINATE was clearing the DCP file status for all stations and not just for those stations assigned to lines on the terminating DCP. That would cause the MCP to forget about auditing or file mode operations that were in progress for stations on other DCPs.

The second problem occurred when DCCLEAR was invoked to purge object job output while a DCP was terminating. A memory dump by INVALID DCCLEAR would result, even though DCTERMINATE had already purged the output.

P2583 DATACOM - SUBTRACT STATION FAULT WITH NO "DCP 0"

A "FAULT IN DO CODE" dump would occur in DCRECON when attempting to subtract a station from a line in a network which had no DCP 0 defined. This problem has been corrected.

P2596 DATACOM - NOT READY LINES AFTER CLUSTER EXCHANGE

For DCPs with local memory tables, if clusters were transferred from one running DCP to another, it was possible for the lines on those clusters to be marked Not Ready, even though the Exchange Clusters DCWRITE (Type 129) requested that they be left Ready. That problem has now been corrected by reordering the sequence of steps performed by the MCP: the line and station tables for the affected clusters will now be copied from one DCP to the other before the clusters are transferred, rather than after. In this manner, the line and station tables will be correct when the lines are reinitialized by the Add Clusters request and will be left in the specified Ready/Not Ready state.

If any of the transferred clusters fail to initialize on the new DCP, the MCP will now notice this fact and report the failure(s) on the ODT, just as at DCP initialization. The MCS will be informed by DCWRITE Error #171 in its Exchange Clusters result and by having the mask bit(s) for any failed cluster(s) reset to 0 in the cluster transfer mask (word 0, bits [15:16]). By comparing the original mask with this mask returned in the result message, the MCS can determine which cluster(s) failed to initialize after transfer.

P2622 DATACOM - SET "ICANWAITF" IN DCINSERT GETAREA CALL

DCALGOL programs doing DCINSERTs with arrays that are longer than the size specified in the insert will no longer cause fatal GETAREA dumps.

P2623 DATACOM - "DCTANKING" ERROR HANDLING CORRECTIONS

Two problems in DCTANKING have been corrected.

1. Under some conditions, an error on the tank read could cause the entire GETAREA row to be forgotten erroneously with numerous fatal after effects.

2. Dumps by "GETAREA SIZE" would occur if a queue were emptied while DCTANKING was getting disk for it.

P9144 DATACOM - "ADD STATION TO FILE" "DCWRITE"

The ADD STATION TO FILE DCWRITE (TYPE=67) may now be used with tasks running in swapspace. An MCS performing this DCWRITE will no longer get a DCWRITE ERROR when attempting to add a station to a file for a task running in swapspace.

P9157 DATACOM - INSERTING NULL MESSAGE IN INACTIVE QUEUE

Inserting a null message into an inactive queue will now activate that queue.

DCALGOL

D2838 DCALGOL - SETTING READ-ONLY ATTRIBUTES

In DCALGOL, it is now permitted to assign values to read-only file, task, port and signal attributes. An attempt to assign a value to a direct read-only attribute is still a syntax error.

D2955 DCALGOL - INTERROGATION RESULTS

The second paragraph on Page B-13 of the DCALGOL Reference Manual (Form No. 5000052), should be replaced with the following:

> "In the most common case, interrogation results are returned in one or more messages which have the format described under the "Interrogate Station Environment Result" (Class=15) message. In the case of an error condition or an MCS issuing a blanket interrogate and having no stations assigned, no Class 15 messages are returned."

Add the following at the end of the fourth paragraph on Page B-13:

> "Also of interest is the fact that if an MCS with no stations defined as assigned to is issued such a blanket interrogate, no Class 15 messages are returned and MSG[6] will contain 0 to indicate that fact."

DCALGOL

P1882 DCALGOL - ERRONEOUS PARAMETER CHECKING INFORMATION

The way information about the procedure type is put in the actual and formal parameter checking arrays has been corrected. The preparation of an actual parameter checking array for epilog procedures has been corrected; previously, a parameter mismatch would result when calling a formal epilog procedure.

DCALGOL INTRINSICS

P1888 DCALGOLINT - CORRECT "DCP" SLEEP ADDRESS INTERPRETATION

Procedure DCERRORANALYSIS in the DCALGOLINTRINSICS attempts to interpret a DCP sleep address of the form LOB3F:1 into label information of the form USER(12345678)+9. It calls DCSYSTEMTABLES to perform this function for it, supposedly passing it the DCP number as well as the sleep address. Actually, DCERRORANALYSIS was passing only the sleep address, so all label interpretation was being done for DCP 0. This has been corrected, so that label interpretation will be against the proper DCP and code file.

DCP PROGRAM GENERATOR

D2772 DCPPROGEN - IMPLEMENT VARIABLE SIZE STATION TABLE

In the past, a station table base size was either 6 words (without DCP sequencing) or 8 words (with DCP sequencing). Certain message-oriented constructs for autonomous DCP systems (like auditing and filemode) also needed extra words in the station table. Rather than always making the station table base the maximum possible size, each NDL compilation will now determine what constructs were used that require extra words in the station table and will assign and allocate only those locations needed; thus, with this change, the station base may be any (fixed) size from 6 to 10 words in length. The message-oriented MCP has been similarly modified to allow it to make use of this new information and to initialize a datacom system whose stations tables have either the old or new sizes.

DCP PROGRAM GENERATOR

P2108 DCPPROGEN - "DCCONTROL" INVALID INDEX

DCCONTROL should no longer get an INVALID INDEX when a move/swap cluster DCWRITE is requested.

P2201 DCPPROGEN - "OR" AND "XOR" CODE GENERATION

Under some circumstances, DCPPROGEN was not generating the proper code sequence for implementing the NDL operators OR and XOR. The problem, which occurred only when both operands were byte-variables and did not reside in the same scratchpad or word of memory, has been corrected.

P2325 DCPPROGEN - FURTHER AUXILIARY LOGIC CORRECTIONS

Two problems dealing with the selective placement of portions of request sets or line controls into auxiliary (main) memory have been corrected in the NDL compiler. Both problems concerned the usage of $SET AUXLOGIC/$RESET AUXLOGIC dollar cards in NDL source programs in which the DCP AUXILIARY statement was not used. 'If the AUXLOGIC dollar card option was changed only between request sets and line controls, its occurrence might be ignored. Furthermore, when the AUXLOGIC options was changed within a request set or line control, it sometimes would not take effect until after the following statement, rather than before. Both of these problem areas have been corrected; some additional code optimization in the switching between local and auxiliary memory has also been performed.

P2401 DCPPROGEN - EXTEND LABELS FOR "20" BIT ADDRESSING

With the advent of Message Oriented Autonomous DCPs, it is possible to use much more than 16K of local (autonomous) memory (for tables and buffers). The code generated for an autonomous DCP by the compiler already uses 20-bit addresses, but the internal variables of the compiler (labels, etc.) still accept only 16-bit addresses. Thus the statistics printed out at the end of a compilation by DCPPROGEN are currently incorrect when more than 16K of autonomous memory is required.

This change to the compiler corrects those statistics and other internal variables and error checking; it makes no change to any code that is generated.

P2402 DCPPROGEN - FULL DUPLEX TERMINATION

Several problems associated with terminating a full-duplex station that was operating on a Message-Oriented Autonomous DCP have been corrected.

P2441 DCPPROGEN - "LOSSOFCARRIER=DISCONNECT" VS. "SECUREDLINES"

If the SECUREDLINES option was used in the DCP section of a SOURCENDL program, a loss of data set carrier could be treated as a disconnect condition, even for those lines using modems which had not specified LOSSOFCARRIER=DISCONNECT. This situation has been corrected.

P2491 DCPPROGEN - CORRECT CODE LISTING FOR 20-BIT LABELS

A recent change to DCPPROGEN converted internal labels from 14 to 20 bits to handle larger memory sizes. When $CODE is set to produce a code listing, all branch addresses must be converted into the new form of internal label before they can be displayed. This was not being done, such that a branch to address LOXXXX:X printed out as address M4XXX:X. This problem has been corrected.

P2493 DCPPROGEN - "TERMINATE OUTPUTREQUEST" LABEL ERRORS

Because of improper code optimization, DCPPROGEN could get a label error during code generation for an NDL TERMINATE OUTPUTREQUEST statement. The problem has been corrected.

P2554 DCPPROGEN - SAVE AND USE "ADDCLUSTERS" VARIANT

An inadvertent change to III.0.1 and higher DCPPROGEN caused the variant field of the ADDCLUSTERS request to be ignored. This field is used to indicate whether the lines on the affected clusters should initially be made READY or NOT READY. By ignoring the variant, the usual effect was to leave the line in whatever state it had been in previously. The variant will now be saved and properly used in determining the initial line state.

DCP TEST GENERATOR

P2366 DCPTESTGEN - "MAKE NEWTAPE FILEKIND=DCPSYMBOL"

When using "$SET NEW" using DCPTESTGEN, the FILEKIND of the new symbolic created will no longer default to DATA; instead, it will be set to DCPSYMBOL by the compiler.

DCSTATUS

D2696 DCSTATUS - EXTENSIONS

SYSTEM/DCSTATUS has been extended to provide more information from the datacom network files.

1. A <modem option> has been added. This option will report modem information from the NIF file for a specific modem or for all modems defined in the network.

   The syntax for the <modem option> is:

```
-- MODEM ----------------------|
           |                |
           |-<modem number>-|
```

2. The <terminal option> has been extended to report the names of the application sets defined for that terminal.

   No additional syntax is required.

3. The <dcp option> has been extented to report DCP information from the NIF file.

   The modified syntax for the <dcp option> is:

```
-- DCP ---<dcp number>------------|
        |                         |
        |---------------- NDL -|
        |                |
        |-<dcp number>-|
```

4. A <network option> has been added to produce a brief tabular network configuration report. The report contains the following fields and information:

```
DCP number and exchange status
Cluster number
LINE info
     Address (DCP number: cluster number: LINE number)
     STATION information
          Station name (20 characters)
          Special characters or address characters
          LSN
          Terminal name (17 characters)
          Connection medium (DIRECT or modem name, 17 char)
     LINE speed, character size, adapter mode
     MCS name (30 characters)
```

   The syntax for the <network option> is:

```
-- NETWORK ----------------------------|
            |                      |
            |-<datacom file prefix>-|
```

   <datacom file prefix>

```
--<file name>--------------------------|
               |                   |
               |- ON --<familyname>-|
```

   The default files are those of the currently active network or those specified by the <file option>. If datacom is not running or a report on a different set of files is desired, the <datacom file prefix> must be specified.

   Note: SYSTEM/DCSTATUS will append DCPCODE and NIF in the correct places. Usercode files are valid.

5. A <file option> has been added to permit access to datacom files other than the currently active network files. If no <datacom file prefix> is specified, file access defaults to the currently active network.

   -- CAUTION --

   Changing the network file prefix precludes reporting information requiring active network information; i.e., the following are the only allowable options.

```
          DCP <dcp number> NDL
          STATION <station number> NDL
          TERMINAL
          MODEM
          GRAPH
          NETWORK
```

The syntax for the <file option> is:

```
-- FILE ---------------------------|
           |                   |
           |-<datacom file prefix>-|
```

Examples:

```
       RUN SYSTEM/DCSTATUS("DCP 1 NDL;NETWORK MIDNIGHT/NET")
       RUN SYSTEM/DCSTATUS("FILE NOON/NET;TERMINAL;FILE;LINE 0:20")
      *DCSTATUS NETWORK
     **DP REMOTE ("FILE TEST/NET;MODEM;FILE;DCP NDL")

      * Syntax for CANDE
     ** Syntax for DIAGNOSTICMCS
```

DMS II - GENERAL

D2611 DMSII - TRANSACTION PROCESSING SYSTEM

Extensions required to support the DMSII Transaction Processing System have been implemented.

See Appendix C, Implementation of Transaction Processing, for details of implementation.

D2709 DMSII - "B7700" OPTIMIZATION

To optimize DMS II programs on the B7000 series, an attempt is made to not purge the associative memory when executing the READLOCK operator. All the DMS II products and the DMS II compilers (DMALGOL, COBOL, ALGOL, and PL/I) have been affected.

The compilers normally released to B7000 series sites are generated with the dollar option B7700CODE set. This means they have the potential to generate code optimized for the B7000 series under control of the dollar option B7700. Furthermore, the default value of the B7700 dollar option for these compilers is SET. Thus, they automatically generate code optimized for the B7000 series machines by default.

The compilers normally released to B6000 series sites are generated with the dollar option B7700CODE reset. This means that they do not have the ability to generate code optimized for the B7000 series regardless of the settings of any dollar options in the user program symbolics.

All code files will run correctly on either series of machines. Furthermore, codefiles optimized for the B7000 series will probably not incur too much extra processor overhead when running on B6000 series machines.

The DASDL compiler has the dollar option B7700 which is reset by default. The setting of this dollar option is remembered in the description file, and ACCESSROUTINES compiled from such a description file will be optimized for the B7000 series if the DMALGOL compiler used has this capability. The DASDL dollar option B7700 should normally be superfluous, however, since DMALGOL compilers capable of generating code optimized for the B7000 series have the B7700 dollar option set by default.

D2802 DMSII - RETAIN OR REUSE OLD STRUCTURE NUMBERS

The algorithm for assigning structure numbers in DASDL has been changed. Structures now retain their structure number when they are updated or reorganized. Unassigned structure numbers, which are created by structure deletion, are now reused by newly added structures on the next DASDL update. This keeps the size of the database description file at a minimum.

Programs that attempt to use deleted structures or structures that have changed since the program was compiled will now receive an exception, VERSIONERROR Subcategory 2, when attempting to access the structure rather than being terminated with an INVALID OP fault.

D2806 DMSII - DATA DICTIONARY

The Data Dictionary consists of a data base containing information about data and transaction bases and the programs which reference them, together with programs for entering and deleting this information, and for collating it and displaying various reports.

See Appendix A, Implementation of Data Dictionary, for details of implementation.

D2868 DMSII - NEW ERROR MESSAGES

As a result of the implementation of Transaction Processing, the following error messages have been implemented:

Auditerror

3	Midtransacton and not in transaction state
4	Normal Begintransaction or Endtransaction while data base opened by Transaction Processing System
5	Transaction Processing System Begintransaction or Endtransaction while data base not opened by Transaction Processing System
6	Transaction Record has improper TRSTATE value
7	Midtransaction while data base not opened by Transaction Processing System

Abort

2	Abort has occurred and Transaction Processing System rerun not finished

Openerror

44	Data base opened in incompatible subsystem
45	This option not permitted while Transaction Processing System active
46	This option not permitted until Transaction Processing System recovery completed

D2930 DMSII - "III.1 DMSII" ON "III.0 MCP"

III.1 DMSII software may be used with the III.0 MCP on either the B6000 or B7000. B6000 systems must use the III.0 PR1 or later MCP; B7000 systems must use the B7000 III.0 PR1 MCP.

Both the UTILTY Dump Tape Directory and the Transaction Processing System employ libraries; therefore, they require the III.1 MCP.

D2954 DMSII - "II.8" DATA BASE SOFTWARE ON "III.2"

III.2 ACCESSROUTINES will not be usable by application programs compiled on II.8 or earlier releases.

D2965 DMSII - UPDATING FROM "III.0" TO "III.1 DMSII"

The following procedure may be used to update to the III.1 DMSII release.

1. Copy and compare the III.0 DASDL source, description file and system software to tape using Library Maintenance.

2. Dump the data base files to tape using the III.0 UTILITY program.

3. Load the III.1 DMSII software and perform DASDL update. In order to ensure an easy return to III.0, do not make any changes to the DASDL description of the data base during the update. Following successful update, III.1 system software will automatically be compiled.

4. As a precaution, the data base files may be dumped using III.1 UTILITY. If ADDRESSCHECK or CHECKSUM are enabled for the data base, III.1 UTILITY will check these values as the data base is dumped. Structures which contain errors may be reconstructed using III.1 software from III.0 UTILITY dumps and III.0 audit trails.

5. User programs will continue to run on III.1 without being recompiled.

The following procedure may be used to return to III.0 software provided no changes were made to the DASDL description of the data base during DASDL update.

1. Dump the data base files to tape using III.1 UTILITY.

2. Backup copies of the audit may be made using COPYAUDIT.

3. Reload the III.0 DASDL source, description file and system software from tape.

4. Perform a RECOVER UPDATE of the control file using DMCONTROL. This will transfer the information from the III.1 control file to the III.0 control file. Do not RECOVER INITIALIZE the control file.

5. User programs will continue to run normally. Programs compiled with III.1 compilers will run using III.0 ACCESSROUTINES if the update level in the description file was not changed by the DASDL update. This value is only changed when the description of the data base changes.

Several D-Notes contain information which is especially useful for conversion to III.1 DMSII.

a. ACR D2617

   The visible DBS operator interface has been substantially enhanced. This D-Note collects all information concerning the visible DBS in a single document.

b. DASDL D2751

   DASDL now starts a WFL file titled DATABASE/WFL/COMPILEACR to compile all tailored DMSII software.

c. REORG D2736

   This D-Note collects all information concerning REORGANIZATION in a single document.

d. UTILITY D2710

   I/O error handling has been improved. UTILITY now verifies checksums and addresschecks when dumping the data base. UTILITY now detects errors which previously went undetected.

e. UTILITY D2835

   This D-Note collects all information concerning UTILITY in a single document.

DMS II - ACCESSROUTINES

D2566 ACR - ADDITIONAL TRANSACTION STATISTICS

This change to the ACCESSROUTINES adds some additional statistics to the transaction statistics and corrects a number of minor problems with the transaction statistics.

The following problems have been eliminated:

1. The I/O WAIT TIME for the audits was not being cleared on a STATISTICS RESTART through the visible DBS.

2. WAIT FOR SYNCPOINT time was not being accumulated for programs that did an ENDTRANSACTION SYNC that had to wait for syncpoint. Also, WAIT FOR SYNCPOINT time was not being accumulated for programs taking a syncpoint because of a close of the data base.

3. The WAIT FOR SYNCPOINT time could occasionally be incorrect and show up as a very large value. PRINTSTATISTICS was incorrectly resetting the value when multiple programs were waiting for syncpoint when PRINTSTATISTICS ran.

A number of new statistics have been added to the transaction statistics summary. An example of the new statistics and a description of them is given below.

```
                                           TRANSACTION STATISTICS
                                           ----------- ----------
TOTAL TRANSACTION COUNT                       241
TOTAL SYNCPOINT COUNT                          25
TOTAL CONTROLPOINT (CP) COUNT                  12
WAIT FOR SYNCPOINT/CONTROLPOINT COUNT           5
AVERAGE WAIT FOR SYNCPOINT/CONTROLPOINT     0.297  SECONDS
AVERAGE TIME TO TAKE CONTROLPOINT           0.480  SECONDS
AVERAGE NUMBER OF BUFFERS FLUSHED AT CP       9.9
PERCENT OF MODIFIED BUFFERS FLUSHED AT CP    49.8 %
```

1. TOTAL TRANSACTION COUNT

   This is a count of the number of times that BEGINTRANSACTION has been executed.

2. TOTAL SYNCPOINT COUNT

   This is a count of all syncpoints that have been taken. This does not include syncpoints that turn into controlpoints; thus, if the syncpoint frequency is 10, and the controlpoint frequency is 10, every 100 transactions the statistics will reflect 9 syncpoints and 1 controlpoint.

3. TOTAL CONTROLPOINT (CP) COUNT

   This is a count of all controlpoints taken against the data base.

4. WAIT FOR SYNCPOINT/CONTROLPOINT COUNT

   Basically, this is a count of how many times programs had to wait at BEGINTRANSACTION for a pending syncpoint to be taken. This also includes the number of times programs had to wait for a syncpoint because of an ENDTRANSACTION SYNC statement or because of an implicit syncpoint at data base close.

5. AVERAGE WAIT FOR SYNCPOINT/CONTROLPOINT

   This is the average amount of time a program had to wait for a syncpoint to be taken. Since some syncpoints cause controlpoints to be taken, this wait time also includes wait for controlpoint time. Since controlpoints typically take longer than syncpoints, the wait for syncpoint time may be severely biased by the number of times controlpoints are taken. The next statistic, TIME TO TAKE CONTROLPOINT, should help to determine the amount of bias.

6. AVERAGE TIME TO TAKE CONTROLPOINT

   This is the average amount of elapsed time required to accomplish a controlpoint. This includes the time to flush the current audit buffers, flush all modified data buffers not flushed since before the last controlpoint, flush all storage control information, and update all disk headers for structures whose LASTRECORD attribute has changed. This time is directly affected by the number of buffers per structure being used and by the audit buffer size. Large numbers of buffers available for a structure or a large audit buffer size can cause controlpoint times to be excessively large. In addition, high controlpoint frequencies can increase total controlpoint overhead. The audit buffer size should not be made so small that too many audit I/O's occur and WAIT FOR AUDIT time becomes excessive.

7. AVERAGE NUMBER OF BUFFERS FLUSHED AT CP

   This is the average number of modified buffers forced back to the data base at controlpoint time.

8. **PERCENT OF MODIFIED BUFFERS FLUSHED AT CP**

This is the percent of all modified buffers present at controlpoint time that were actually flushed back to the data base at that time. Careful control of the audit buffer size, number of buffers per structure, and control point frequency as indicated in 6) above can reduce this figure from its potentially worst value of 50% down to a value close to the ideal of 0%.

D2607 ACR - NEW "I/O" ERROR HANDLING PROCEDURES

Procedure IOERRORHANDLER has been implemented to handle I/O errors and ACCESSROUTINES detected errors (ADDRESSCHECK and CHECKSUM). IOERRORHANDLER will display appropriate information regarding the error, retry the I/O if called for, display the results of any retries, and, if the retries fail, set the DMROWLOCK bits and report that fact.

Upon entering IOERRORHANDLER, two messages are displayed. The first specifies whether the I/O was a read or a write, the structure the I/O was for, and the file name.

Example:

```
    DISPLAY:***READ ERROR ON STR #2, FILE:(DMSII)DB/D/DATA ON DMS.

    DISPLAY:***WRITE ERROR ON STR #3, FILE:(DMSII)DB/D/S ON DMS.
```

The second message gives more detailed information about the I/O as follows:

RSLT the processed I/O result descriptor from the DCB followed by an explanation of the error
FAMILYNAME the family where the file in question resides - <fid>.FAMILYNAME
FAMILYINDEX the family index of the unit to which the I/O was attempted - <fid>(<row>).ROWADDRESS.[29:8]
ADDRESS the hardware address of the block.
ROW the row of the file on which the I/O was attempted - <block> DIV <fid>.AREASIZE
BLOCK the relative segment number of the first segment of the block of the file on which the I/O was attempted - <buffer>.IORECORDNUM

Examples:

```
    DISPLAY:***RSLT=CHECKSUM FAILED, FAMILYNAME=DMS, FAMILYINDEX=2,
       ADDRESS=108355, ROW=3, BLOCK=42.

    DISPLAY:***RSLT=400002860009 : UNIT NOT READY, FAMILYNAME=DMS,
       FAMILYINDEX=1, ADDRESS=2917, ROW=1, BLOCK=1.
```

NOTE: Since ADDRESSCHECK and CHECKSUM errors are ACR detected after good reads, no result descriptor will be displayed.

Following these displays, if the error was on a read:

If the error is an ADDRESSCHECK or a CHECKSUM error, the read will be retried up to MAXRETRIES (a DATABASE/PROPERTIES define, currently 3) times in an attempt to get around any transient error. If the read was retried, a message will be displayed telling the number of retries attempted and whether they succeeded or not.

Example:

```
    DISPLAY:***READ FOR STR #2 ON DMS HAS BEEN RETRIED 3 TIMES
       WITHOUT SUCCESS.

    DISPLAY:***READ FOR STR #2 ON DMS WAS RETRIED 2 TIMES BEFORE
       SUCCEEDING.
```

If the error was on a write:

The write will automatically be retried 1 time and if this fails, the operator will be asked whether the write should be retried again or not. If told to retry, the write will be retried up to MAXRETRIES times, and, if still not successful, will repeat its accept and retry loop. Upon leaving the retry loop (the retries succeeded, or the operator responded saying to lock the row), a message will be displayed telling the number of retries attempted and whether they succeeded or not.

Examples:

```
    DISPLAY:***WRITE FOR STR #3 ON DMS HAS BEEN RETRIED 1 TIMES
       WITHOUT SUCCESS.
    ACCEPT:R TO RETRY WRITE FOR STR #3 ON DMS, OR L TO LOCK ROW #2
       AND PROCEED.
```

NOTE: In response to the ACCEPT exactly R or L must be entered. No trailing characters or blanks are permitted.

```
    DISPLAY:***WRITE FOR STR #3 ON DMS WAS RETRIED 14 TIMES BEFORE
       SUCCEEDING.
```

NOTE: All successful writes will be re-read and compared against the original for correctness before the write is considered good.

If the retries are not successful, the appropriate DMROWLOCK bit will be set and the operator will be told that the READERROR or ROWLOCK bit was set.

Examples:

```
DISPLAY:***READERROR BIT FOR ROW #3 OF STR #2 ON DMS HAS BEEN SET.

DISPLAY:***ROW #2 OF STR #3 ON DMS HAS BEEN LOCKED OUT.

DISPLAY:***READERROR BIT FOR ROW #3 OF STR #2 ON DMS WAS
   ALREADY SET.
```

If the database is audited and the retries fail, a special record (RDERR for reads, BLKIMG for writes) will be written to the audit trail for later analysis. These records contain the result descriptor and the block in error so that frame shifts, dropped or added bits, etc., might be seen.

Some examples of an entire dialog are:

1. A CHECKSUM error is detected which is corrected after 1 retry

```
DISPLAY:***READ ERROR ON STR #19, FILE:(DMSII)DB/D/E/DATA ON DMS.
DISPLAY:***RSLT=CHECKSUM FAILED, FAMILYNAME=DMS, FAMILYINDEX=3,
   ADDRESS=671124, ROW=26, BLOCK=263.
DISPLAY:***READ FOR STR #19 ON DMS WAS RETRIED 1 TIMES BEFORE
   SUCCEEDING.
```

2. A non retriable read error occurs (only ADDRESSCHECK and CHECKSUM errors are retried for reads)

```
DISPLAY:***READ ERROR ON STR #63, FILE:(DMSII)DB/D/ISSET ON DMS.
DISPLAY:***RSLT=400006C00901 : PACK DRIVE ERROR, FAMILYNAME=DMS,
   FAMILYINDEX=2, ADDRESS=291376, ROW=43, BLOCK=441.
DISPLAY:***READERROR BIT FOR ROW #43 OF STR #63 ON DMS HAS BEEN
   SET.
```

3. An ADDRESSCHECK error is detected which fails retrying

```
DISPLAY:***READ ERROR ON STR #7, FILE:(DMSII)DB/D/D1/DATA ON DMS.
DISPLAY:***RSLT=ADDRCHECK FAILED, FAMILYNAME=DMS, FAMILYINDEX=1,
   ADDRESS=399440, ROW=2, BLOCK=513.
DISPLAY:***READ FOR STR #7 ON DMS HAS BEEN RETRIED 3 TIMES
   WITHOUT SUCCESS.
DISPLAY:***READERROR BIT FOR ROW #2 OF STR #7 ON DMS HAS BEEN SET.
```

4. An uncorrectable or non retriable read error is detected in a row which already has its READERROR bit set

```
DISPLAY:***READ ERROR ON STR #63, FILE:(DMSII)DB/D/ISSET ON DMS.
DISPLAY:***RSLT=8027778A0881 : PARITY, FAMILYNAME=DMS,
   FAMILYINDEX=2, ADDRESS=291586, ROW=43, BLOCK=493.
DISPLAY:***READERROR BIT FOR ROW #43 OF STR #63 ON DMS WAS ALREADY
   SET.
```

5. A transient write error occurs which is corrected on the first retry

```
DISPLAY:***WRITE ERROR ON STR #63, FILE:(DMSII)DB/D/ISSET ON DMS.
DISPLAY:***RSLT=400010B20501 : WRITE LOCK OUT, FAMILYNAME=DMS,
   FAMILYINDEX=4, ADDRESS=495092, ROW=4, BLOCK=4015.
DISPLAY:***WRITE FOR STR #63 ON DMS WAS RETRIED 1 TIMES BEFORE
   SUCCEEDING.
```

6. A write error occurs because of a problem which can easily be corrected (reload pack firmware or move a pack to a new drive)

```
DISPLAY:***WRITE ERROR ON STR #1, FILE:(DMSII)DB/DATA ON DMS.
DISPLAY:***RSLT=400007840801 : MPX OR CONTROLLER ERROR,
   FAMILYNAME=DMS, FAMILYINDEX=1, ADDRESS=493688,
   ROW=0, BLOCK=0.
DISPLAY:***WRITE FOR STR #1 ON DMS HAS BEEN RETRIED 1 TIMES
   WITHOUT SUCCESS.
ACCEPT:R TO RETRY WRITE FOR STR #1 ON DMS, OR L TO LOCK ROW #0
   AND PROCEED.

   Operator diagnoses and corrects the problem.

OPERATOR ENTERED: AX:R.
DISPLAY:***WRITE FOR STR #1 ON DMS WAS RETRIED 2 TIMES BEFORE
   SUCCEEDING.
```

7. A write error occurs because of a problem which cannot be easily diagnosed or corrected

```
DISPLAY:***WRITE ERROR ON STR #19, FILE:(DMSII)DB/D/E/DATA ON DMS.
DISPLAY:***RSLT=800F17824101 : SECTOR FORMAT ERROR, FAMILYNAME=DMS,
   FAMILYINDEX=3, ADDRESS=688267, ROW=29, BLOCK=304.
DISPLAY:***WRITE FOR STR #19 ON DMS HAS BEEN RETRIED 1 TIMES
   WITHOUT SUCCESS.
ACCEPT:R TO RETRY WRITE FOR STR #19 ON DMS, OR L TO LOCK ROW #29
   AND PROCEED.
```

Even if the problem cannot be diagnosed and fixed immediately, it is generally a good idea to retry a couple of times rather than just locking the row.

```
OPERATOR ENTERED: AX:R.
DISPLAY:***WRITE FOR STR #19 ON DMS HAS BEEN RETRIED 4 TIMES
   WITHOUT SUCCESS.
ACCEPT:R TO RETRY WRITE FOR STR #19 ON DMS, OR L TO LOCK ROW #29
   AND PROCEED.
OPERATOR ENTERED: AX:L.
DISPLAY:***ROW #29 OF STR #19 ON DMS HAS BEEN LOCKED OUT.
```

NOTE: If the operator responds to the ACCEPT incorrectly, the retries DISPLAY and the ACCEPT will be repeated until the response is correct.

Example:

```
DISPLAY:***WRITE FOR STR #19 ON DMS HAS BEEN RETRIED 1 TIMES
   WITHOUT SUCCESS.
ACCEPT:R TO RETRY WRITE FOR STR #19 ON DMS, OR L TO LOCK ROW #29
   AND PROCEED.
OPERATOR ENTERED: AX:RETRY.
DISPLAY:***WRITE FOR STR #19 ON DMS HAS BEEN RETRIED 1 TIMES
   WITHOUT SUCCESS.
ACCEPT:R TO RETRY WRITE FOR STR #19 ON DMS, OR L TO LOCK ROW #29
   AND PROCEED.
OPERATOR ENTERED: AX:R .
```

Note tbe trailing blank

```
DISPLAY:***WRITE FOR STR #19 ON DMS HAS BEEN RETRIED 1 TIMES
   WITHOUT SUCCESS.
ACCEPT:R TO RETRY WRITE FOR STR #19 ON DMS, OR L TO LOCK ROW #29
   AND PROCEED.
OPERATOR ENTERED: AX:R.         O.K.
```

D2608 ACR - FAULT HANDLING IN "ACCESSROUTINES"

The capability of handling faults within the ACCESSROUTINES has been added. Previously, such faults would either cause termination of the user program or would be trapped by the user program fault handling code. With this change such faults will cause a SYSTEMERROR exception to be returned to the user program and eventual termination of the data base.

A side effect of this change is that fatal data base errors detected within the ACCESSROUTINES will no longer cause the user program to be DS-ed because of an EXPON OVERFLOW fault. Instead, an exception will be returned to the user program. The only condition which will cause the ACCESSROUTINES to DS a user program is operator termination of the data base stack.

Once an internal ACCESSROUTINES fault or fatal error is detected, no further processing can occur against the data base. All current data buffers are discarded and, if the data base is audited, auditing is halted. All future attempts to perform operations against the data base will cause immediate exceptions to be returned to the user program. The data base stack will remain in the mix until all users of the data base close the data base (either explicitly or via BLOCKEXIT close).

Potentially, this change will allow user programs to stay up and active even through a total data base failure. Presumably, such programs could then respond to, or possibly tank for later processing, inputs from an on-line network.

Consider, for example, the following situation. A fatal error has been detected in the data base. As soon as the fatal error is detected, all programs using the data base will start receiving SYSTEMERROR exceptions to data base operations. Upon first detecting a SYSTEMERROR, the user program could close the data base and activate interrupt code which would cause the program to open the data base and return to normal operation upon operator notification. In the meantime, the data base error could be analyzed and corrected, perhaps by reconstructing some corrupted structure. Once the data base has been corrected, the user programs could be notified and processing could continue. Obviously, a restart will be required for all programs.

D2609 ACR - REDUCE CONTROL POINT OVERHEAD

The amount of time required to complete a control point against an audited data base has been reduced by the following improvements in the control point mechanism.

1. All write operations required to flush modified data buffers not written at the last control point are now initiated in parallel rather than being done serially.

2. Storage control information such as DKTABLES for STANDARD and STANDARD variable format data sets and NEXTTABLEADDRESS and LASTTABLEADDRESS for tables will be flushed only when modified and only at every other control point. Previously, this information was flushed at every control point whether or not it had been modified.

D2610 ACR - DISPLAY CORE IN USE FOR "ALLOWEDCORE"

Whenever ALLOWEDCORE is interrogated or changed via the SM message to the data base stack, the current total buffer space in use is displayed in the response.

D2612 ACR - TWO CONTROL POINTS AT AUDIT REEL SWITCH

The ACCESSROUTINES will now force two control points immediately after a normal audit reel switch. This will allow the previous audit to be verified or copied sooner and minimize the necessity for using the previous audit in case recovery is required.

D2617 ACR - VISIBLE "DBS" CHANGES AND ENHANCEMENTS

I. INTRODUCTION

With the implementation of reblocking, it has become necessary to allow the user more dynamic control over system resources on a structure by structure basis. This has been achieved via the Visible DBS. Through the operator's console, the user can now:

1. Interrogate the "status" of any physical structure. This displays various settings and states concerning the structure.

2. Change the values of REBLOCK, REBLOCKFACTOR, and buffer specifications for a structure.

In order to implement the new Visible DBS capabilities in a clear and consistent way, it was necessary to change some of the existing commands. All syntax and semantics for the Visible DBS are given below.

II. SYNTAX

All commands to the Visible DBS are entered using the following ODT message:

<DBS mix no.> SM <message>

The <DBS mix no.> must be that of a data base stack, which can be determined by entering the ODT message DBS. The text of the <message> is free field and is comprised of any one of the several commands.

A. DBS status command:

```
-- STATUS --|
```

B. DBS change command:

```
   |<--------------------- , --------------------|
   |                                             |
------/1\- ALLOWEDCORE ---- = <unsigned integer> ----|
      |                |
      |-/1\- SYNCPOINT ----|
      |                |
      |-/1\- CONTROLPOINT -|
```

C. Statistics command:

```
-- STATISTICS ----------------|
                |             |
                |- RESTART -|
```

D. Audit close command:

```
-- AUDIT CLOSE ------------------------------------------------------>
                 |                           |
                 |- PRIMARY = --- NORMAL ----|
                               |             |
                               |- ALTERNATE -|

>--------------------------------------------------------------------|
  |                             |
  |- SECONDARY = --- NORMAL ----|
                  |             |
                  |- ALTERNATE -|
```

E. Structure status command:

```
-- STATUS STRUCTURE --- * ---------------------------|
                      |                          |
                      |-<physical structure list>-|
```

F. Structure change command:

```
-- STRUCTURE -------------------------------------------------------->
>--- = (<parameter list>) -------------------------------------------|
  | |                                                   |
  | |<---------------------- , ------------------------| |
  | |                                                 | |
  |--- <physicial structure list> (<parameter list>) ---|
```

<parameter list>

```
  |<-------------------- , -------------------------|
  |                                                 |
------/1\- REBLOCK ------------------------------------|
    |               |                           |
    |               |- SET ---------------------|
    |               |- RESET -------------------|
    |-/1\- REBLOCKFACTOR = <unsigned integer> --|
    |-/1\- BUFFERS = <int1> + <int2> OR <int3> -|
```

<physical structure list>

```
  |<------------ , -----------|
  |                           |
----<structure specification>----|
```

<structure specification>

```
----<structure no.>-------|
  |                     |
  | |<------ . ------|  |
  | |                |  |
  |---<structure id>---|
```

Examples:

a) 1357 SM STATUS

b) 2468 SM SYNCPOINT=100, CONTROLPOINT=10, ALLOWEDCORE = 25000

c) 1234 SM STATUS STRUCTURE PARTSET, PARTDATA

d) 7654 SM STRUCTURE 22,25 (REBLOCK)

e) 9988 SM STRUCTURE PARTDATA, PARTSET
(REBLOCK RESET, BUFFERS=2+0 OR 0),
22,25(BUFFERS=1+1 OR 3,REBLOCK)

f) 9182 SM STRUCTURE * (REBLOCK,BUFFERS=1+1 OR 2)

g) 2252 SM STATUS STRUCTURE PARTINFO,PARTINFO.NAMES,
PARTINFO.NAMES.SPECS

A. The DBS status command will cause the following information to be displayed:

1. The condition of the database. One of five possible displays will occur:

   a) WAITING FOR ABORT

   b) WAITING FOR RECOVERY

   c) TERMINATION IN PROGRESS

   d) OPEN INITIALIZE

   e) OPEN COUNTS: INQUIRY = <integer>, UPDATE = <integer>

2. The value of ALLOWEDCORE and the amount of buffer core currently in use.

3. The values of SYNCPOINT and CONTROLPOINT.

4. Information about the audit file:

   a) Block size, area size, areas.

   b) Current audit file number, audit block serial number.

   c) Information about the relative position of the ACCESSROUTINES in current audit file.

B. The DBS change command is used to change the values of ALLOWEDCORE, SYNCPOINT, and/or CONTROLPOINT. When this command is used, the specified parameters are changed immediately. The new values are retained until the next time this command is used or the next DASDL/Control File update.

C. The statistics command causes the current statistics to be printed. If "RESTART" is specified, all internal counters and timers are cleared after printing, thus restarting statistics as if the database had just been opened. This command is only valid if STATISTICS is specified in DASDL.

D. The audit close command forces an audit file switch. This command is only valid for audited databases.

The family of the next primary or secondarv (duplicated) audit file may be specified with "PRIMARY =" or "SECONDARY =". "NORMAL" causes the next family to be the normal audit family specified in DASDL; "ALTERNATE" causes the next family to be the alternate audit family specified in DASDL.

If "ALTERNATE" is specified, the attempt by the ACCESSROUTINES to switch back to the normal family after every audit file on the alternate family is disabled--all subsequent audit files will be placed on the alternate family until another "AUDIT CLOSE" message is entered, or until the ACCESSROUTINES go away and are fired up again.

If "NORMAL" is specified, the next audit file will be placed on the normal family, and automatic switching to the alternate family by the ACCESSROUTINES in case of TIMEOUT or SECTORS REQUIRED is enabled.

If neither "PRIMARY =" nor "SECONDARY =" is specified, the ACCESSROUTINES will behave as if a normal audit file switch had occurred.

E. The structure status command is used to find out information about individual structures. Only physical structures (data sets, sets, and subsets - not accesses) may be specified in the <physical structure list>. Qualification is necessary only when the identifier is not unique. Qualification is done by specifying the <structure id>s from outermost to innermost in the hierarchy.

When an "=" is specified, the command applies to all physical structures in the data base.

The structure status command causes the following information to be displayed for every structure in the <physical structure list> (or for all structures if "*" is used):

1. General

   a) Whether the structure is opened or closed.

   b) Number of random and serial users of the structure.

2. Current settings

   a) Whether REBLOCK is SET or RESET for the structure.

b) Buffer specifications.

c) Number of "small" or "large" buffers allocated.

d) The current REBLOCKFACTOR.

e) The next REBLOCKFACTOR which will be used when the structure is closed and re-opened.

F. The structure change command is composed of one or more <physical structure list> and corresponding <parameter list> pairs. Each <parameter list> is applied only to the structures specified in the immediately preceding <physical structure list>. The rules for specification of structures in the <physical structure list> are identical to the ones given under E.

The structure change is used to set the current parameters of the structure. The parameters specified through this command are stored in the Control File and are "remembered" upon subsequent DBS runs. When this command is used with the <physical structure list>, there will also be a status display for every structure specified. No status displays will occur if "*" is used.

The structure parameters which may be changed are:

1. REBLOCK

A. REBLOCK or REBLOCK SET informs the ACCESSROUTINES to employ the REBLOCK algorithm for serial users of the structure. This will only take effect if REBLOCK was TRUE for that structure in DASDL. REBLOCK may never be SET when the "serial buffers" specification is less than 2.

B. REBLOCK RESET prevents all reblocking on the structure. Any current serial users will discontinue reblocking and the "large" buffers will be overlayed.

2. BUFFERS - <int1>, <int2>, and <int3> must be integers from 0 to 254 inclusive. This specification corresponds to the DASDL specification:

```
BUFFERS = <int1> + <int2> PER RANDOM USER
                 + <int3> PER SERIAL USER
```

<int1> specifies the number of base buffers (small) to allocate for the structure. <int2> specifies the number of additional buffers (small) to allocate for each random user. <int3> specifies the number of additional buffers to allocate for each serial user. If REBLOCK is SET, <int3> specifies the number of "large" buffers. <int3> must be at least 2 when REBLOCK is set. If REBLOCK is RESET and <int3> is greater than or equal to 2, the ACCESSROUTINES perform read aheads (with small blocks) for serial users of the structure. When <int3> is less than 2, read aheads do not occur. Changing buffer specifications may cause overlays to adjust the number of buffers in the buffer pool.

NOTE: The number of buffers in the buffer pool may exceed the buffer specifications. This is especially true of audited databases which are running well below ALLOWEDCORE.

3. REBLOCKFACTOR - The user may change the REBLOCKFACTOR for a structure provided REBLOCK was TRUE in DASDL. REBLOCKFACTOR will not actually change until the structure is completely closed and reopened.

III. SYNTAX ERRORS AND WARNINGS

If the input to the Visible DBS contains a syntax error, then an error message prefixed by SYNTAX- will be displayed and the input will not be processed. If the user input is syntactically correct, but certain specifications are illegal, a warning prefixed by WARNING- will be displayed and the remainder of the input will be processed. For example, if the user specified REBLOCK on a structure for which REBLOCK was FALSE in DASDL, a warning is returned and the remaining input is applied.

D2619 ACR - "REBLOCKING"

I. PURPOSE

Previously, a user could optimize either serial or random access to a DMSII structure. However, it was very difficult to optimize both simultaneously. The following example demonstrates the problem:

Example:

A data base contains a data set "D" which has a high volume of random inquiry and/or update against it during business hours. This same data set, however, is used to generate reports at night. In general, a large percentage of the data set must be accessed serially for report generation.

The user has a dilemma in choosing a block size for "D". He can optimize random access by choosing a small block size. This cuts down the core usage and I/O time during the day. However, the report generation phase may be inefficient because of excessive serial I/O time.

If the user optimizes the report generation by choosing large blocks for "D", random access during the day may use too much core and I/O time.

Reblocking addresses this problem by allowing the user to specify two block sizes for the same structure - a small one for random users and a large one for serial users. These block sizes are specified via the following physical attributes:

A. BLOCKSIZE - This has the same meaning it always had. In terms of reblocking, this is the attribute which controls the size of the "small" block to be used for random access.

B. REBLOCKFACTOR - This is a new physical attribute used to specify an integral number of "small" (BLOCKSIZE) blocks which will comprise a large block for serial access.

II. DEFINITIONS

A. Reblocking - Reblocking is the capability of the ACCESSROUTINES to handle two block sizes simultaneously for the same structure. Small blocks are used for random access and large blocks are used for serial access.

B. Small block - A small block is an I/O buffer of size BLOCKSIZE used for random access. If the structure is not reblocked, all buffers are small.

C. Large block - A large block is an I/O buffer whose size is a specified integral number of small blocks. The large block is used for serial access. If the structure is not reblocked, there are no large blocks.

D. Reblock factor - Reblock factor is the number of small blocks in each large block.

III. Features of DMSII reblocking

A. The user may take advantage of reblocking through a normal DASDL update. Reorganization is only necessary if the BLOCKSIZE is to be changed. Changing BLOCKSIZE is discussed under topic VIII.

B. Reblocking does not require reprogramming or recompilation of user programs. Random vs. serial access is determined automatically by the ACCESSROUTINES at run time on a program basis. There are no host language constructs for reblocking.

C. Random and serial access can occur simultaneously. The ACCESSROUTINES will assign small blocks to random users and large blocks to serial users.

D. Reblocking can be turned on and off through the Visible DBS for any structure which had reblocking set in DASDL. Reblock factors and buffers may also be changed for a structure. (See ACCESSROUTINES note D2617.) These features allow the user a tremendous amount of control over core utilization, especially during critical periods (e.g. high volume daily access).

E. The 3.1 DMSII release permits reblocking for DIRECT and STANDARD (fixed or variable format) data sets only, except for the restart data set.

IV. SYNTAX

Two new physical attributes have been implemented in DASDL.

Reblock attribute:

```
-- REBLOCK -------------------------------------------------------|
                  |            |
                  |- = TRUE  --|
                  |            |
                  |- = FALSE  -|
```

Reblock factor attribute:

```
-- REBLOCKFACTOR = <unsigned integer> ----------------------------|
```

These attributes are only valid for DIRECT and STANDARD data sets. Either or both attributes may be specified in the "global" or DATASET defaults statement. These defaults will only be applied to DIRECT and STANDARD data sets.

The REBLOCK attribute informs the ACCESSROUTINES to employ reblocking at run time. A structure cannot take advantage of reblocking unless this attribute is TRUE in DASDL.

The REBLOCKFACTOR attribute specifies the number of small blocks (BLOCKSIZE sized blocks) in each large block. This attribute is meaningful only if REBLOCK is TRUE for the structure. If REBLOCK is FALSE, this attribute is ignored. <unsigned integer> must be an integer from 1 to 60 inclusive.

When REBLOCK is TRUE, both attributes are stored in the Control File. The user may change them through the Visible DBS without recompiling DASDL. (See ACCESSROUTINES note D2617.)

The default settings for these attributes are as follows:

1. If REBLOCK is not specified, then REBLOCK is FALSE and REBLOCKFACTOR is ignored.

2. If REBLOCK is TRUE and REBLOCKFACTOR is not specified, then a REBLOCKFACTOR will be calculated to give a large block size of approximately 1000 words.

V. Buffers

The syntax for the buffer specifications has not changed. However, the "<integer> PER SERIAL USER" specification affects reblocking. REBLOCK may only be SET at run time when the "serial buffers" is greater than or equal to 2. (See ACCESSROUTINES note D2617.) When REBLOCK is SET, the serial buffer specification informs the ACCESSROUTINES how many "large" buffers to use for each serial user. The buffers specification may be changed through the Visible DBS.

VI. Host language statements which invoke reblocking

Several host language statements may cause the ACCESSROUTINES to employ reblocking. These statements are given below. ("D" is a data set. "S" is a SET, SUBSET, or ACCESS of "D"):

```
1. FIND NEXT D;

2. FIND S AT . . . . . . . .
   FIND NEXT S AT . . . . .
   FIND D VIA S AT . . . . .
   FIND D VIA NEXT S AT . .
   FIND FIRST S AT . . . . .
   FIND D VIA FIRST S AT . .

3. STORE D;     (after CREATE or LOCK)
```

NOTE: LOCK, MODIFY, or DELETE may be substituted for FIND above.

Reblocking may only occur when:

1. REBLOCK is TRUE in DASDL.

2. REBLOCK is SET for "D" at run time. This implies that "serial buffers" must be greater than or equal to 2. (See ACCESSROUTINES note D2617.)

3. The user program accesses "D" sequentially. This is automatically detected by the ACCESSROUTINES.

If these criteria are not met, only small blocks will be handled by the ACCESSROUTINES.

VII. Interaction of reblocking with the Visible DBS

Both REBLOCK and REBLOCKFACTOR are dynamic parameters of a structure stored in the Control File. If REBLOCK is FALSE in DASDL, then neither can be modified through the Visible DBS. If REBLOCK is TRUE in DASDL, then both may be modified through the Visible DBS. In this case, REBLOCK may be SET or RESET (turned on or off), and REBLOCKFACTOR may be altered to any integer from 1 to 60 inclusive. (See ACCESSROUTINES note D2617.)

NOTE: The dynamic parameter, REBLOCK, can only be SET when "serial buffers" is greater than or equal to 2. Therefore, if REBLOCK is TRUE in DASDL, then:

1. If "serial buffers" is greater than or equal to 2 then REBLOCK will be initialized to SET in the Control File.

2. If "serial buffers" is less than 2 then REBLOCK will be initialized to RESET in the Control File.

VIII. User considerations for choice of BLOCKSIZE

The most critical choice the user has to make for reblocking is the BLOCKSIZE. Once an effective BLOCKSIZE is chosen, it is fairly easy to choose an optimal REBLOCKFACTOR.

Choosing a BLOCKSIZE often requires a tradeoff between pack space versus core utilization and I/O time. The user should minimize BLOCKSIZE as much as possible without wasting too much pack space.

Example:

Data set "D" has a record size of 8 words and CHECKSUM and ADDRESSCHECK are set. The ideal BLOCKSIZE is the smallest possible I/O, 1 segment (30 words). If the user chooses BLOCKSIZE = 3 RECORDS to achieve this minimum I/O, the file would look like this:

```
             /-------- record/word layout --------\

                  8         8         8      1 1   4
              |---------|---------|---------|-|-|-----|
    Block 1   |record 1 |record 2 |record 3 |C|A|waste|   (1 segment)
              |---------|---------|---------|-|-|-----|

              |---------|---------|---------|-|-|-----|
    Block 2   |record 4 |record 5 |record 6 |C|A|waste|   (1 segment)
              |---------|---------|---------|-|-|-----|
                                   .

                                   .


              |---------|---------|---------|-|-|-----|
    Block n   |record   |record   |record   |C|A|waste|   (1 segment)
              |---------|---------|---------|-|-|-----|
```

STRUCTURE D

BLOCKSIZE = 3 RECORDS (24 WORDS)
(C = checksum word, A = addresscheck word)

Only 80% of the file is user data, while 7% is control information and 13% is wasted. This is the optimal BLOCKSIZE for the user who can afford the wasted pack resource. However, for most users a BLOCKSIZE of 7 RECORDS would utilize pack much better:

```
             /------- record/word layout -------\

                  8         8         8        6
              |---------|---------|---------|-------|
              |record 1 |record 2 |record 3 |record |
    Block 1   |-------------------------------------|   (2 segments)
              |4|record 5|record 6|record 7|CA|Ws|
              |-|--------|--------|--------|--|--|
               2     8        8        8     2  2
                                     .

                                     .

                                     .

              |---------|---------|---------|-------|
              |record   |record   |record   |record |
    Block n   |-------------------------------------|   (2 segments)
              | |record  |record  |record  |CA|Ws|
              |-|--------|--------|--------|--|--|
```

STRUCTURE D

BLOCKSIZE = 7 RECORDS (56 WORDS)
(CA = checksum addresscheck words, Ws = waste)

Here 93% is user data and only 7% is non-user data. This increase to 56 word blocks has bettered pack utilization and may not impact core utilization significantly.

An even larger BLOCKSIZE of 11 RECORDS would eliminate all wasted space. Since the block would be larger, a user should evaluate this BLOCKSIZE in terms of core utilization.

The choice of REBLOCKFACTOR is not as critical as BLOCKSIZE. The larger the REBLOCKFACTOR, the less I/O time there is for the serial user. However, there is a point at which an increase in the REBLOCKFACTOR will not decrease the overall elapsed time of the serial programs. If serial programs are apt to be run when system resources are precious, a smaller REBLOCKFACTOR will help minimize core usage. Experimentation with REBLOCKFACTOR through the Visible DBS can help in finding the proper REBLOCKFACTOR.

IX. REBLOCK algorithm
------- ---------

If REBLOCK is TRUE in DASDL and SET at run time, the ACCESSROUTINES employ the following REBLOCK algorithm:

1. Serial detection - The ACCESSROUTINES detect whether a user program is accessing the structure serially or randomly. If the program is serial, it enters "reblock mode" and the ACCESSROUTINES allocate large buffers for it.

2. Implicit read ahead - Once a program is in "reblock mode", large block "read ahead" type reads will occur.

3. Small block reads for random I/O - If a serial program happens to perform a random access, the record will be read using a small block. Using small blocks instead of large blocks may greatly reduce I/O time depending upon the frequency of these "spurious" reads.

4. Oscillaton detection - If a program oscillates between random and serial access, the ACCESSROUTINES detect this situation and make it increasingly hard for the program to change modes. This prevents spurious "read aheads" and buffer overlays.

D2657 ACR - FIND CORRECT END OF DISK TYPE AUDITS

In the past a number of problems have been reported which seem to have been attributable to Halt/Load recovery finding the wrong end of disk or pack audit files. RECOVERY's mechanism for finding the end of disk type audits was susceptible to error when undetected I/O errors occurred in reading the audit, e.g., the I/O was reported as successful but the data was transferred to memory incorrectly. RECOVERY would assume that any time a timestamp discontinuity occurred in the audit, that this was indeed the correct end of the audit. Once RECOVERY ran with the wrong end of audit, the data base would be corrupted and the only appropriate recourse would be to rebuild the data base to a point in time short of the problem in the audit.

As a result of such problems, a new mechanism has been implemented which virtually ensures that Halt/load recovery cannot find the wrong end of the audit. For disk type audits, the ACCESSROUTINES will now always write a detectable stopper pattern after each audit block except for the last block in each area. Halt/load recovery will search for this pattern when looking for the end of the audit. Since the audit end-of-file is always after the stopper pattern, an unrecovered audit file can only end with either the stopper pattern or normal end-of-file. Any other error, such as audit checksum, timestamp discontinuity, or unexpected audit block serial number, will be fatal when encountered. A stopper pattern will not exist in a recovered audit file or in an unrecovered audit file when the last block written was written at the end of an area (in which case audit end-of-file points at that block).

Should RECOVERY detect such a fatal error in searching for the end of the audit, the user has two possible recourses. If the primary audit can be fixed by copying the secondary audit, then this should be done and RECOVERY rerun. If the audit error cannot be fixed, a rebuild must be done to a point short of the audit error.

D2658 ACR - IMPROVE BUFFER MANAGEMENT

The buffer deallocation algorithm which is executed when total buffer storage exceeds ALLOWEDCORE has been improved. It is simpler and more efficient. Also, buffers can now be deallocated in parallel by multiple processes intead of being deallocated serially by a single process.

DMS II - ACCESSROUTINES

P1518 ACR - ZIP "COPYAUDIT" AT FINAL CLOSE

If the final close of audit file N was done before two control points were performed, COPYAUDIT would not be zipped by the ACCESSROUTINES for audit file N-1. This problem has been corrected.

P1582 ACR - "STATISTICS" ON "CHECKSUM" RETRIES

STATISTICS are now gathered on retries caused by CHECKSUM failures and reported with the number of reads and read wait times.

P1585 ACR - RETRY "ADDRESSCHECK" FAILURES

The ACCESSROUTINES will now retry all ADDRESSCHECK errors up to MAXRETRIES (default=3) times. If it still fails, the row will have its READERROR bit set and IOERROR 256 (ADDRESSCHECK FAILED) will be returned.

P1738 ACR - STATISTICS MIDNIGHT OVERLAP

Several midnight overlap problems in printing statistics have been corrected. In particular, the statistics interval was incorrect, and the data base open interval did not account for two or more midnight crossovers. A timing problem involving multiple calls on the MCP TIME intrinsic has also been corrected.

P1742 ACR - "CREATE/STORE" INCORRECT FOR REMAP

Records created using a remap of a STANDARD data set contain incorrect values. Items which were REQUIRED in the data set and HIDDEN in the remap were initialized to NULL value regardless of the INITIALVALUE specified in DASDL.

P1773 ACR - READ AHEAD ON COARSE TABLES

When index sequential is in a "read ahead" state, the ACCESSROUTINES will now read ahead the coarse tables. Reading ahead a coarse table will only occur when the last fine table for the current coarse table is being accessed. Formerly, read ahead never occurred for coarse tables.

P1774 ACR - "GENERATE NULL" BITVECTOR

After GENERATE NULL on a bit vector and subsequent Halt/Load, the end of file for the bit vector may have been set up wrong by RECOVERY. This caused a "read past end of file" on a subsequent GENERATE on the bit vector. This problem has been corrected.

P1802 ACR - AUDIT FILE SWITCH FOR DISK OR PACK AUDIT

If the first attempt to open a new audit file was unsuccessful, the second and subsequent attempts could cause the wrong audit file name to be used. The audit pack family name was used in place of the proper audit file title. This problem has been corrected.

P1820 ACR - FAULTY AUDIT CONSTRAINT

On the II.9 and III.0 releases, the ACCESSROUTINES constraint that the audit buffers must be written before the data buffers was faulty. The result was that if the timing were just right, Halt/Load recovery would recover the data base improperly. The most common symptoms would be key and data mismatch, set entries pointing at deleted records, missing set entries, etc. This has been corrected.

P1821 ACR - "INVALID INDEX" AFTER AUDIT TIMEOUT

The ACCESSROUTINES would sometimes fault with INVALID INDEX following a timeout on a pack audit write, caused by an incorrect FOR statement in an I/O list. This problem has been corrected.

P1822 ACR - BLOCK ZERO OF CONTROL FILE CORRUPTED

Block zero of the control file no longer becomes corrupted (with all zeros) when an update program is DSed attempting to open the control file.

P1900 ACR - MAXBUFFERS STATISTIC INCORRECT

Reconstruction was making DCBs present unnecessarily when stopping the data base for the final phase.

Also, the statistics printout of the value of the maximum number of buffers used was too large by a value of 0 to 5. These problems have been corrected; there is also less resizing activity on DCBSET, which should decrease the overhead in running the ACCESSROUTINES.

P1901 ACR - "LOCK TO MODIFY DETAILS"

An INVALID OP would occur in the ACCESSROUTINES when an OPEN INITIALIZE was done on a partitioned data set which was declared in DASDL with the option "LOCK TO MODIFY DETAILS". This has been corrected.

P1907 ACR - RESET SYNCPOINT TO LOWER VALUE

When resetting the value for syncpoint through the visible DBS, it was possible to hang the data base if the new value was lower than the old syncpoint value. This has been corrected.

P1908 ACR - UNLOCK GLOBAL DATA AND ORDERED BUFFERS

Some code for ordered and global data in which some buffers were not unlocked prior to close has been corrected. The problem had no known user impact.

P1909 ACR - READ PAST "EOF" ON INDEX SEQUENTIAL

A data base will no longer terminate with a READ PAST END OF FILE error on an INDEX SEQUENTIAL following an abort. This could only happen if READAHEAD were set on the INDEX SEQUENTIAL.

P1910 ACR - CORRUPTION OF UNORDERED

A problem in which unordered "next prior" links may become corrupted following a Halt/Load has been corrected.

P1911 ACR - REMAPS OF ORDERED DATA SETS WITH SUBBLOCKS

Loading an embedded ordered data set through a remap did not utilize the subblocking feature when SUBBLOCKSIZE was specified for the data set in DASDL. As a result, blocks would contain only records belonging to a single master. The problem results since a SUBBLOCKSIZE of 0 (no subblocking) is always used when the embedded data set is invoked via a remap. This problem has been corrected along with other possible problems caused by the improper setting of SUBBLOCKSIZE when embedded ordered data sets are referenced through remaps.

P1912 ACR - REORGANIZATION OF COMPACT DATA SETS

After reorganization of a compact data set, an INVALID INDEX could occur in the ACCESSROUTINES upon record update or deletion on the compact data set. In addition, REORGANIZATION would under special circumstances insert an invalid entry in the compact data set available space table for the last data block. This entry, however, would never be accessed and, therefore not cause any problems. Both of these problems have been corrected.

P1913 ACR - DUPLICATE AUDIT BLOCK IF SWITCH TO TAPE

In some cases, the ACCESSROUTINES would duplicate the first block of an audit file by writing it twice. This happened rarely, and only when the audit file was switched from or pack to tape. This has been corrected.

P1914 ACR - "FILEDC" A CONTROL RECORD

FILEDC is now treated as a control record by the ACCESSROUTINES and PRINTAUDIT. This reduces the likelihood that FILEDC will be overlooked when scanning the audit.

P1915 ACR - "COPYAUDIT" NOT ZIPPED IF ERROR OR TIMEOUT

COPYAUDIT was not being zipped if the audit file was closed because of a write error or timeout on a write. The problem has been corrected.

P1916 ACR - TWO HALT/LOADS CAUSES RECOVERY FAILURE

If the following sequence of events occurred:

> There was a Halt/Load;
> RECOVERY ran almost to completion;
> A program opened the data base inquiry and closed it immediately;
> There was another Halt/Load immediately;
> The second RECOVERY was run.

The second RECOVERY would fail due to an extraneous SDSEOF audit record at the end of the audit. The problem has been corrected.

P1917 ACR - UNAUDITED "DB" CORRUPTION INQUIRY VS. UPDATE

For unaudited databases, if inquiry and update programs were running simultaneously and all update programs discontinued use of the data base leaving only the inquiry program, data base corruption would occur. This problem has been corrected.

P1947 ACR - LESS RESIZING OF ARRAYS

As an efficiency measure, the ACCESSROUTINES now resize internal arrays less often and by more intelligent values.

P2009 ACR - RETURN RESTART AREAS

ACCESSROUTINES will no longer suppress the returning of restart areas when a data base has been DSed and Halt/Load recovery run.

P2010 ACR - CHECKSUM ERROR

Erroneous checksum errors no longer occur in the after image phase of any type of recovery for standard data sets. Also, ROLLBACK no longer gets false checksum or invalid table serial number errors when rolling back through a section of the audit backed out by a previous Halt/Load recovery.

Previously, it was possible to receive false address check errors in an after image phase of recovery on a standard data set without checksum. This problem has been corrected.

P2095 ACR - CORRECT DELETION IN UNORDERED

When a data base contained both variable format Unordered and fixed format Unordered simultaneously, DELETEs would fault. This problem has been corrected.

P2096 ACR - "DB" CORRUPTION DUE TO MISSING ERROREXIT

The ACCESSROUTINES no longer corrupts the data base if an error is encountered during initialization of the audit. Formerly, the following could occur:

1. Invalid increment of the audit file number in the control file.

2. Data base corruption by subsequent Halt/Load recovery.

P2097 ACR - "DS" OF RECOVERY

Halt/Load recovery no longer turns off the Halt/Load flag if DSed when attempting to open an audit file to write restart areas.

P2098 ACR - RETRY AUDIT OPENS

The ACCESSROUTINES will now attempt to retry the open of the audit if a fault or attribute error occurs on the initial open.

P2127 ACR - CLEAR READ/WRITE AHEAD STATISTICS

The ACCESSROUTINES now zero out the read and write ahead counters when the command STATISTICS RESTART is issued to the visible DBS.

P2128 ACR - READAHEAD CORRECTIONS

1. The ACCESSROUTINES now read ahead correctly for compact data sets using "FIND NEXT D;".

2. The user program will no longer fault in the ACCESSROUTINES when find current is executed against a compact data set.

3. The ACCESSROUTINES will no longer attempt to read ahead deleted records. (The error was formerly possible only when deletes were multiprocessed with finds against the same compact data set.)

P2129 ACR - CONTROL FILE OPEN ON ABORT

If the task which was waiting for abort were DSed, the attempt to reopen the control file following the completion of abort processing caused the DBS to terminate abnormally. This has been corrected.

P2130 ACR - "ROWLOCKOUTAUDIT" ERRORS

RECOVERY no longer terminates abnormally because it failed to find the end of the ROWLOCKOUTAUDIT properly. Also, RECOVERY no longer gets attribute errors on DMROWLOCK when the ROWLOCKOUT audit file is present. In addition, a stopper was added to the ROWLOCKOUTAUDIT for positive end of file indication, checksum was added to it, and the data base date-time stamp was added to it. I/O errors and checksum errors on the ROWLOCKOUTAUDIT are retried, and the ROWLOCKOUTAUDIT is read after each write to ensure the write occurred properly.

P2131 ACR - RECONSTRUCT MAKES EMPTY AUDIT FILE

If a program has the data base open inquiry when the last update program closes the data base, the audit file is still held open. If DATARECOVERY goes into its final phase at this time, an empty audit file is generated, which will cause recovery to terminate abnormally if scanned during a subsequent recovery. The empty audit file is no longer generated.

P2132 ACR - ELIMINATE UNNECESSARY DISPLAYS

Unnecessary displays are eliminated when close fails to store a last good restart area because the program died outside of transaction state. When an unexpected exception does occur, complete information on it is displayed.

P2133 ACR - CREATE/STORE AT END OF "DIRECT" DATA SET

When multiple programs were creating records at the end of a DIRECT data set, it was possible for the data set to become corrupted. This problem has been corrected.

P2183 ACR - CHECKSUM ERRORS

It was possible to get false checksum errors in RECOVERY during an after image phase when CREATE/STOREs in the audit extended to end of file of a variable format standard data set. The error has been corrected.

P2184 ACR - ERROREXIT TIMING PROBLEM

It was possible for the data base to become corrupted when the operator DSed the visible DBS or a fault in the ACCESSROUTINES occurred, because the audit was not disabled soon enough after writes to the data base files were disabled. This has been corrected.

P2244 ACR - RETURN OF COMPACT RECORD

A fault occurs in the ACCESSROUTINES when a modified compact data set record is returned to its parent block from an overflow block. This problem occurs only on audited data bases and only when the parent block is empty. This problem has been corrected.

P2329 ACR - INVALID COUNT OF READERS

The internal count of readers of the data base is now maintained correctly for all structures.

P2347 ACR - ORDERED AVAILABLE TABLE CORRUPTED

The ACCESSROUTINES no longer fail to enter an available address at word 3 of an available table for ORDERED data sets.

P2348 ACR - DATA BASE HUNG AFTER CLOSE

The ACCESSROUTINES no longer hang after a close of an access-typed structure. The following conditions are necessary to cause the problem:

1. A DIRECT, RANDOM or ORDERED data set must be invoked by the user program before the restart data set.

2. The program must close the data base after an ABORT recovery without executing BEGINTRANSACTION.

3. Any subsequent program attempting to cause an ABORT hangs the data base.

P2349 ACR - "ORDERED" DATA SET "INVALID OP"

The ACCESSROUTINES no longer die with an INVALID OP when performing a store on an ORDERED data set.

P2350 ACR - "ORDERED" DATA SET IMPROPER AUDIT

The ACCESSROUTINES now correctly audit changes to ORDERED available tables. Formerly, corruption could be left in the tables after a recovery process causing a subsequent fatal error to the data base.

P2351 ACR - "ORDERED ADDRESSCHECK, CHECKSUM" CORRUPTED

The ACCESSROUTINES no longer overwrite the ADDRESSCHECK or CHECKSUM word(s) with address words. Formerly, this could occur in available tables.

P2352 ACR - "ORDERED" DATA SET LOOP

The ACCESSROUTINES no longer get into a loop when linear searching ORDERED data sets.

P2359 ACR - RESEQUENCE DATA BASE AND "RECOVERY" SYMBOLICS

DATABASE and RECOVERY symbolics have been partially resequenced.

P2360 ACR - STORE OF "ORDERED" WITH "BLOCKSIZE=1"

The ACCESSROUTINES no longer go into an infinite loop when a store is attempted on an ORDERED data set with a BLOCKSIZE equal to one record.

P2362 ACR - ABORT HANDLING MECHANISM

The interface between the HOSTLIB and the ACCESSROUTINES was redesigned to close timing windows and correct some problems.

P2379 ACR - DESIGNATED SERIAL NUMBERS AND STATISTICS

An ACCESSROUTINES syntax error is no longer generated when designated serial numbers and statistics are specified.

P2387 ACR - ERRONEOUS NOT FOUND RESULT

INQUIRY could get an erroneous "not found" result from the ACCESSROUTINES for random data sets whose LASTBLOCKADDRESS was a negative value. This problem has been corrected.

P2409 ACR - ELIMINATE VARIABLE FORMAT CODE

GETDATAADDRESS and FORGETDATAADDRESS for random data sets contained code for handling variable format records. This code has been eliminated.

P2410 ACR - NO ABORT ACCEPT WHEN "DBS" IS DSED

When the DBS is DSed, abort recovery no longer stops with the ACCEPT "RDS ERROR, OK TO CONTINUE".

P2411 ACR - PROGRAM "DS" CAUSES ABORT TO DIE

Dsing a program just prior to firing up ABORT recovery would sometimes prohibit an ABORT run and DS the data base. This problem has been corrected.

P2453 ACR - ERRONEOUS "DIRECT KEY" CORRUPTED

When one stack was trying to find a record in a direct data set and a second stack was trying to store the same record, the DBS could DS with an erroneous DIRECT KEY CORRUPTED error. The timing window responsible for this situation has been closed.

P2454 ACR - DESIGNATED SERIAL NUMBERS

User programs may now be DSed when attempting to open audits using designated serial numbers. Formerly, this would hang the data base.

Note: A program attempting to open an audit should never be DSed, since that may take down the data base. This change allows the data base to be cleared without a Halt/Load.

P2473 ACR - INCORRECT STATISTICS TOTALS

The following statistics problems have been corrected:

1. If statistics were printed multiple times
   a. the I/O statistics totals would be incorrect
   b. the I/O time statistics (column 1) would be incorrect

2. Statistics restart was not clearing
a. the file I/O time statistics (column 1)
b. all of the audit statistics

3. A timing problem could cause data base statistics for one session to be carried forward to a new session.

4. The statistics interval was incorrectly reflecting the time since the statistics were last printed, instead of the time since the statistics were last cleared.

P2474 ACR - FREE RECORDS ON ABORT EXCEPTION

The ACCESSROUTINES now release all records locked by a program when returning an abort exception.

P2495 ACR - "DS DBS" VS. FILE ATTRIBUTES

Operator DS of a user task could cause the ACCESSROUTINES to incorrectly read or set some file attributes. Although situations leading to invalid file attribute handling have been restricted, they have not been completely eliminated. Errors in reading or setting file attributes may still occur if the operator issues more than one DS of a user task.

P2496 ACR - ZERO "TPS" TIMESTAMP AT CLOSE

The Transaction Processing System's timestamp which is passed to the ACCESSROUTINES at the first open is now reset to zero upon the final close.

P2497 ACR - "COPYAUDIT" VS. AUDIT IOERROR SWITCH

Following an ioerror on an audit file, COPYAUDIT was being ZIPped with an ending audit block serial number one higher than the last block in the audit file to be copied. The problem has been corrected.

P2498 ACR - "INVALID INDEX" GLOBAL DATA AND ONLINE DUMP

UTILITY no longer dies with an INVALID INDEX in the ACCESSROUTINES when performing an online dump. This formerly occurred on audited data bases with global data.

P2600 ACR - REMOVE OVERLAY POINT

The ACCESSROUTINES no longer displays OLAYPOINT for a visible DBS STATUS command, nor does it print OLAYPOINT for statistics.

OLAYPOINT is an ACCESSROUTINES internal variable which the user cannot control; therefore, its visibility has been deimplemented.

P2601 ACR - PROGRAM NOT "DSED" IF AUDIT OPEN FAILS

If a program attempts to open a new audit, the open fails, the operator requests that the open be retried, and the open succeeds, the program will no longer be DSed.

P2602 ACR - INITIALIZE PARTITION AUDITED INCORRECTLY

When a new partition was created via a remap, the wrong structure number was placed in the Open Initialize Partition audit record. The structure number of the remap was audited rather than the structure number of the data set. This could cause RECOVERY to fail when the audit record was encountered. This problem has been corrected.

P2603 ACR - PARTITIONED DATA SET INACCESSABLE VIA REMAPS

Partitioned data sets could not be accessed using remaps. Programs which opened the logical data base INQUIRY received NOTFOUND Subcategory 7 exceptions. Programs which opened the logical data base UPDATE received the same exception if they attempted to access the remap outside transaction state. Programs which opened the data base UPDATE and were in transaction state caused the partition directory to be corrupted and could terminate with the message "PARTITION ALREADY PRESENT". This problem has been corrected.

P2604 ACR - "NOT FOUND" FOR PARTITIONED ORDERED DATA SETS

The first attempt to FIND or LOCK a record in a partitioned ordered data set following the open of the partition resulted in a spurious NOT FOUND 01 exception. Subsequent references to the same record worked correctly. This problem has been corrected.

P2605 ACR - CORRECTION OF INTERNAL DELETE COUNTER

An internal counter of deletes on a block was not being incremented correctly. This could very rarely cause a deleted record to be retrieved. This problem has been corrected.

P2606 ACR - AUDIT SMALL BLOCKS FOR "IO" ERRORS

The ACCESSROUTINES now audits only the small blocks in a "reblocked" block in case of an IO error. Formerly, the "reblocked" block was audited causing Quick Fix to die with a SEG ARRAY error.

P2627 ACR - DEADLOCK DURING UPDATE AND DELETE

When concurrent updates and deletes are occurring to compact data sets, a deadlock may occur. One process has locked a DCB and requires storelock; the other has storelock and needs the locked DCB. This problem has been corrected.

P2636 ACR - MISSING "STARTDB" IN ORDERED PATHFINDER

The STARTDB call at the beginning of Pathfinder in Ordered data sets was erroneously eliminated; it has now been restored.

P2637 ACR - TIMING WINDOW IN "OVERLAY"

A timing problem in the new OVERLAY procedure could cause a fatal error in the data base. This problem has been corrected.

DMS II - BDMSALGOL

D2678 BDMSALGOL - "OPEN INITIALIZE" WARNING

The OPEN INITIALIZE statement will now be flagged with a warning stating that the statement will be deimplemented on the III.2 release.

D2760 BDMSALGOL - "B7000" SERIES "DM" OPTIMIZATION

On the B7000 series systems, the BDMSALGOL compiler now generates improved locking code prior to entering the ACCESSROUTINES; this new code prevents purges of the associative memory. This new locking code is generated only if the compiler itself was compiled with the compiler generation $ option B7700CODE set and the $ option B7700 is not explicitly reset within the user program.

While the improved locking code is not optimal for the B6000 series systems, the code will execute correctly on the B6000 systems. The B6000 series BDMSALGOL compiler (and the B7000 compiler with $B7700 reset) will continue to generate code optimal for B6000 systems.

DMS II - BDMSALGOL

P1980 BDMSALGOL - "STRUCTURENUMBER" CORRECTION

Use of the construct STRUCTURENUMBER will no longer cause incorrect code to be generated in rare situations.

DMS II - BDMSCOBOL

D2848 BDMSCOBOL - "B7000" SERIES "DM" OPTIMIZATION

On the B7000 series systems, the BDMSCOBOL compiler now generates improved locking code prior to entering the ACCESSROUTINES; this new code prevents purges of the associative memory. This new locking code is generated only if the compiler itself was compiled with the compiler generation $ option B7700CODE set and the $ option B7700 is not explicitly reset within the user program.

While the improved locking code is not optimal for the B6000 series systems, the code will execute correctly on the B6000 systems. The B6000 series BDMSCOBOL compiler (and the B7000 compiler with $B7700 reset) will continue to generate code optimal for B6000 systems.

D2860 BDMSCOBOL - USE OF <STRUCTURE NUMBER> CONSTRUCT

The syntax diagram on Page 5-7 of the DMS II HOST Reference Manual (Form No. 5001498) is incorrect. It should appear as follows:

```
----<data set name>--- ( DMSTRUCTURE ) --|
   |                 |
   |-<set name>------|
   |                 |
   |-<subset name>---|
```

The semantics should read as follows:

"<structure number> allows the programmer to programmatically determine the structure number of a data set, set or subset."

D2884 BDMSCOBOL - "<DATA-ITEM-1> IN <KEY-CONDITION>"

The BDMSCOBOL compiler has been corrected to allow <data set name> as a qualifier of <data-item-1> in <key-condition> if <data-item-1> is ALPHA or <data-item-1> is NUMBER. The DMS II HOST Reference Manual (Form No. 5001498), Page 13-2, should be changed accordingly.

D2961 BDMSCOBOL - "DM" ATTRIBUTE <STRUCTURE NUMBER>

The DM attribute <structure number> example on Page 5-7 of the DMSII Host Reference Manual (Form No. 5001498) is incorrect. The statement should read as follows:

```
IF D(DMSTRUCTURE)=DMSTATUS(DMSTRUCTURE) DISPLAY "D FAULT"
```

DMS II -- BDMSCOBOL

P2057 BDMSCOBOL - "DATABASE" DECLARATION

BDMSCOBOL will now successfully recovery from an improperly specified DATABASE declaration, instead of terminating with a "VISIT NONACTIVE TASK" message.

P2202 BDMSCOBOL - BAD "DMS" INVOKE LISTING

An error in an invocation of a data base in the 'DATA-BASE SECTION' would cause the compiler to get an INVALID INDEX fault when $LIST was set. This has been corrected.

P2299 BDMSCOBOL - BAD SYNTAX CHECKING IN "DB" STATEMENT

The COBOL compiler no longer waits on a "NO FILE DESCRIPTION/<database name>" when the reserved word "OF" is misspelled in the <logical data base option>. Now, the compiler will give a syntax error.

P2330 BDMSCOBOL - LISTING OF THE VARIABLE FORMAT INVOCATION

The format line for the variable format invocation will now appear in the proper place.

P2331 BDMSCOBOL - DECLARATION OF ITEM WITH SAME NAME AS SET

BDMSCOBOL did not allow the declaration of an item having the same name as a DMSII set invoked by the program, even though the name can be uniquely qualified. This has been corrected.

P2332 BDMSCOBOL - "INVALID INDEX"

Syntax checking in statements that handles DM attributes was incorrect, which caused the compiler to abort with an INVALID INDEX. This has been corrected.

P2333 BDMSCOBOL - INCONSISTENT HANDLING OF KEY CONDITION

If a data-item (numeric) larger than the size of the key-item is used in a key-condition, movement to user key area follows normal COBOL MOVE rules resulting in truncation of the data-item. Thus, comparison in PATHFINDER is on only the least significant digits of the data-item, and records may be returned which do not meet the key-condition if non-zero digits were truncated. The compiler will give a warning message.

DMS II - BDMS/PL/I

D2849 BDMSPLI - "B7000" SERIES "DM" OPTIMIZATION

On the B7000 series systems, the BDMSPLI compiler now generates improved locking code prior to entering the ACCESSROUTINES; this new code prevents purges of the associative memory. This new locking code is generated only if the compiler itself was compiled with the compiler generation $ option B7700CODE set and the $ option B7700 is not explicitly reset within the user program.

While the improved locking code is not optimal for the B6000 series systems, the code will execute correctly on the B6000 systems. The B6000 series BDMSPLI compiler (and the B7000 compiler with $B7700 reset) will continue to generate code optimal for B6000 systems.

DMS II - BDMS/PL/I

P1771 BDMSPLI - PROPER PRINTING OF LENGTH ATTRIBUTE

The compiler will now print the lengths of data base items in the ATTRIB and EXTERN listings correctly, avoiding INVALID INDEX faults in certain circumstances.

DMS II - BUILDINQUIRY

D2695 BUILDINQ - SETS WITH GROUP KEYS

Sets with group keys are now entered twice: once with the group as key (as before) and again with the group elementary items as keys. This makes it possible for INQUIRY to select the best search algorithm if either the group item or the elementary items is specified in the selection condition.

D2863 BUILDINQ - LOGICAL DATA BASES WITH MULTIPLE INVOKES

DMSII INQUIRY cannot be generated for a logical data base which invokes the same structure more than once. For example, the following logical data base is not valid for INQUIRY:

```
    LDB DATABASE(DS1, DSX=DS1, DS2);
```

Such a logical data base will result in a fatal error message in BUILDINQ.

DMS II - BUILDINQUIRY

P1444 BUILDINQ - AUTOMATICALLY SELECTS RESTART DATA SET

When the user chose to select which data sets INQUIRY would handle, it was possible that the restart data set would not be selected. If the selection process were terminated by entering a "C" before the restart data set was reached, the restart data set was not selected. Now the restart data set, if one exists, is unconditionally selected.

P1823 BUILDINQ - ENFORCE ANSWER ON UPDATE QUESTION

Previously, when BUILDINQ asked if update was requested, any answer other than YES was assumed to mean NO. Now, BUILDINQ insists that either YES or NO be input before it continues.

P1824 BUILDINQ - DATA SET SELECTION OPTION ENFORCED

BUILDINQ asks the user to specify whether an entire data base, selected data sets or logical data bases are required. Previously, any response other than 1,2 or 3 caused program termination. Now, invalid responses cause an error message to be displayed and the question repeated.

P1918 BUILDINQ - DELETED LOGICAL DATA BASE

BUILDINQ was not skipping deleted logical data bases when searching for a given logical data base. Thus, if a logical data base were deleted by a DASDL update and then added back by a subsequent DASDL update, BUILDINQ for that logical data base would find the first (deleted) entry and subsequently fail (e.g., INVALID INDEX, SEG ARRAY, etc). This has been corrected.

P1919 BUILDINQ - VERIFIED LINK

If a data set containing a verified link is included, but the data set to which it refers is not (in the Selected Data Set mode), an INVALID INDEX fault may occur. This has been corrected.

P2011 BUILDINQ - OCCURRING GROUPS

BUILDINQ was not correctly handling subscripts for non-occurring items which exist in a non-occurring group, which in turn exists in an occurring group. The error causes an erroneous "SUBSCRIPT OUT OF RANGE" message when the item is referenced with a valid subscript. This has been corrected.

P2309 BUILDINQ - VAGUE LINK AND SUBSET REFERENCES

BUILDINQ now resolves vague references of links and subsets included in remaps. Previously, items which referenced a data set that was not selected for INQUIRY also were not selected. Now, BUILDINQ checks to see if a remap of the original data set has been selected; if so, the item is selected and fixed to reference the remap.

P2334 BUILDINQ - LOGICAL DATA BASE WITH "<SET PART>" OF NONE

When BUILDINQ was run for a logical data base which contained a <set part> of NONE in the <logical data base> declaration in DASDL, the error message "TABLE EXCEEDED" was incorrectly output. This problem has been corrected.

P2353 BUILDINQ - INVALID "TABLE EXCEEDED" ERROR

A problem existed in BUILDINQ which could cause a fatal TABLE EXCEEDED error to be inappropriately displayed. The problem occurred when one or more sets in the data base contained keydata. Such sets could, in general, have caused unpredictable results.

P2388 BUILDINQ - EMBEDDED SETS IN LOGICAL DATA BASES

An INQUIRY program built for a logical data base would not allow for the use of automatic sets which spanned embedded data sets. This problem has been corrected.

P2412 BUILDINQ - MAXIMUM SIZE OF "DMINQDIRECTORY"

For a very large DASDL description, an INVALID INDEX would occur in BUILDINQ if the maximum size of the DMINQDIRECTORY as defined DMINQ/BUILDSYMBOLIC was exceeded. If the maximum size of the DMINQDIRECTORY as defined in DATABASE/PROPERTIES was exceeded, an INVALID INDEX would occur during the DMALGOL compilation of the INQUIRY program. These problems have been eliminated. An error message "MAXIMUM DIRECTORY SIZE EXCEEDED" is now issued by BUILDINQ when this system limitation is encountered. The size of the DMINQDIRECTORY may be increased by patching DATABASE/PROPERTIES (DASPROBUFFSZ may be changed to 2**16) and recompiling BUILDINQ and DMALGOL.

P2607 BUILDINQ - SELECTION USING KEY DATA ITEMS

A selection expression involving key data items caused a SOFTWARE ERROR when accessing occurred via an index. This problem has been corrected.

P2638 BUILDINQ - ACCEPTANCE OF QUEUE SPECIFICATION

BUILDINQ ignored any queue specification given with card input. Also, a specified queue value of zero was ignored. These problems have been corrected.

DMS II - BUILDREORGANIZATION

D2591 BUILDREORG - CHECKING "BUILDREORG" STATUS IN "WFL"

BUILDREORG now sets its STATUS to TERMINATED if any errors are encountered. Thus, comparison of the task variable associated with BUILDREORG with COMPLETEDOK is now valid in WFL. In addition, an error-free run of BUILDREORG returns as TASKVALUE of one. A run with errors returns a TASKVALUE of zero.

D2754 BUILDREORG - SORT ENVIRONMENT SYNTAX

The clauses of the <sort environment specs> may now be given in any order, separated by commas. These clauses are the following: <extract specs>, <sort specs> and <loadfactor specs>.

See REORGANIZATION note D2736 for complete syntax and semantics.

D2775 BUILDREORG - "REORGANIZATION" LIMITATIONS

Certain types of data base reorganization which were previouly documented as valid have not as yet been fully implemented in the DATABASE/REORGSYMBOLIC program. When these types of reorganization are detected by SYSTEM/BUILDREORG, a fatal "REORGANIZATION LIMITATION" error results after the printing of the SYSTEM/BUILDREORG report. The limitations documented below will be eliminated on a subsequent DMSII release.

The following types of reorganization represent current REORGANIZATION limitations. Where possible, it is noted how the limitation might be overcome by changing the specifications given to SYSTEM/BUILDREORG.

1. Generation of an index sequential structure using an index random or ordered list structure where the index sequential structure has DUPLICATES allowed but FIRST or LAST unspecified. Generation of the index structure using itself or generation of the index structure from the data set, if the index is a disjoint set or automatic subset, are possible alternatives that can be taken to avoid this limitation.

2. Generation of an unordered list using an index structure or fixup of an unordered list when the data set is generated and one of the following conditions also exist:

   A. The data set is ordered by a prime index which is not the generated unordered list.

   B. The data set is of type RANDOM or UNORDERED.

   Generation of the index from the generated data set, if the index is a disjoint set or automatic subset, or using the unordered list as the prime set, if it is an automatic set. are possible alternatives to avoid this limitation.

3. Generation of an unordered list from a generated RANDOM data set.

4. Generation of an ordered list using an index structure where the ordered list has DUPLICATES allowed but FIRST or LAST unspecified and one of the following conditions also exist:

   A. The data set is ordered by a prime index which is not the generated ordered list.

   B. The data set is of type RANDOM or UNORDERED.

   Using the ordered list as the prime set or generation of the ordered list from the data set are possible alternatives to overcome this limitation.

5. Generation of an embedded index sequential structure where the index structure has DUPLICATES allowed but FIRST or LAST unspecified and the following conditions also exist:

   A. The data set is generated.

   B. The index sequential structure is not the prime set.

   Using the index sequential structure as the prime set is a possible alternative to avoid this limitation.

D2803 BUILDREORG - AUTOMATIC CHECKSUM OF INTERMEDIATE TAPE FILES

Checksumming is now performed automatically on all tapes written and read during reorganization. A warning is now issued when the CHECKSUM specification is given to SYSTEM/BUILDREORG; this specification will be deimplemented on the III.3 system release.

DMS II - BUILDREORGANIZATION

P1465 BUILDREORG - SEQUENCING OF PRIME INDEX, DEFAULT GENERATE

An "ILLEGAL SEQUENCE" error could occur if the prime set generate preceded the data set generate in the SEQUENCE statement and the prime set was generated by default. Also, a default generate for an INDEX RANDOM structure with DUPLICATES allowed, but FIRST or LAST unspecified, was inappropriately taken. Such a structure should be fixed up by default. Both these problems have been corrected.

P2185 BUILDREORG - REPORTING OF "ORDERED BY" FOR INDEXES

When an index structure was GENERATEd USING an index structure, the USING structure was improperly reported as the ORDERED BY structure on the BUILDREORG report. This is no longer the case.

P2477 BUILDREORG - "INVALID INDEX" AT "30139840"

Due to an error in BUILDREORG, a DASDL statement of KIND=PACK could cause an INVALID INDEX error in the compile-time processor at 30139840 when compiling REORG. This has been corrected.

DMS II - COPY AUDIT

P1548 COPYAUD-II - CORRECT BLOCK "0" SIZE ERROR MESSAGE

If COPYAUDIT detects an invalid blocksize on block 0 of the input tape, it now displays the following, where MM is the requested blocksize and NN is the actual blocksize:

"READ TAPE [0] MM WORDS BUT READ NN"

P2217 COPYAUD-II - BLOCK "0" CHECKSUM

The checksum in block 0 was sometimes computed and stored incorrectly when copying from tape to disk. This and other incompatibilities with the audit stopper implementation have been corrected.

P2478 COPYAUD-II - CLOSE NEW AUDIT BEFORE OLD

The new audit file(s) created by COPYAUDIT are now closed before removing the old audit. Previously, it was possible to remove the old audit and then get errors closing the new audit, resulting in no good audits being available.

DMS II - DASDL

D2567 DASDL - "VERSION" COMPILER OPTION

The VERSION compiler option is now available in DASDL.

Syntax:

```
-- $ VERSION --- <version> . <cycle> ------------------------|
                |                         | |                |
                |- + <version> . + <cycle> -| |- . <patch> -|
```

<version> is a two-digit <unsigned integer>; <cycle> and <patch> are three-digit <unsigned integer>s.

The VERSION compiler option allows the user to specify an initial version number or to replace or increment an existing version number.

The VERSION option is initially inserted in the DASDL symbolic file by setting the NEW option and including a VERSION statement with the dollar sign in column two in the CARD input file.

During subsequent compiles, the version can be updated by setting the MERGE and NEW options and including a VERSION option in the CARD file. Two techniques are available for updating the VERSION card. When plus signs precede the <version> and <cycle> numbers in the CARD file, the existing cycle and version numbers in the TAPE file are incremented by the specified values. When plus signs are not specified, the version information from the CARD file replaces the existing version information. In either case, the updated VERSION card is written to the new symbolic file.

The VERSION option in the CARD file must have a lower sequence number than that in the TAPE file. If more than one VERSION option appears in the CARD file, the one with the highest sequence number is used.

<patch> numbers are optional. The new patch number is the patch number in the CARD file, if any; otherwise, it is the patch number in the TAPE file. If neither VERSION option contains a patch number, the patch number is omitted in the new symbolic.

D2568 DASDL - ALLOW MORE STRUCTURES, ITEMS PER STRUCTURE

DASDL will now properly allow a single data base to describe up to 4095 active and deleted structures (including data sets, sets, subsets, accesses, remaps). A single data base may contain up to 1023 data sets, sets and subsets. In the case of partitioned structures, the number of structures is multiplied by the value of "OPEN PARTITIONS", but is independent of the actual number of partitions that currently exist.

Each data set may contain up to a total of 1500 items (including data items and embedded data sets, sets, subsets, accesses and remaps). Each data base may contain up to 1500 global items, which include global data items and disjoint data sets, sets, subsets, accesses and remaps.

These limits can be further increased to a maximum of approximately 3000 items per data set and 3000 global items per data base. If DASDL terminates with the message "PROPERTY ARRAY SIZE EXCEEDED", the following procedure can be used to reach these larger limits.

a. Modify the DATABASE/PROPERTIES file as follows:

```
DASTEXTBUFFSZ=2**16 #,      23811000
DASPROPBUFFSZ=2**16 #,      23812000
```

Note: 2**16 is the maximum value which can be specified due to hardware addressing limitations.

b. Run DATABASE/PROPERTIES

This creates a new data base properties file.

c. Recompile the following programs:

SYSTEM/DASDL
SYSTEM/DMALGOL
SYSTEM/DMCONTROL
SYSTEM/BUILDINQ
SYSTEM/BUILDREORG
SYSTEM/LOADDUMP
DATABASE/INTERFACE

D2613 DASDL - OPTIMIZE "CLEARDATA" TEXT

DASDL now emits optimized CLEARDATA text for occurring items.

D2614 DASDL - LIST "TAPE" AND "NEWTAPE" FILE TITLES

DASDL now lists the titles of the TAPE and NEWTAPE files when the LIST dollar option is set.

D2615 DASDL - "DASDL" DEFAULTS FOR DATA SETS AND SETS

DASDL's physical option calculations have been changed in three ways. Default attribute selection has been improved, output from the FILE dollar option has been enhanced, and modification of the POPULATION option has been simplified.

A. Default attribute selection

Default selection has been improved. These new defaults apply to new data bases, new structures added to existing data bases, and existing structures changed using reorganization ITEMS SAME or ITEMS CHANGED. DASDL update will not change options for existing structures. Defaults for the following options have been changed.

AREAS and AREASIZE

1. For RANDOM data sets and INDEX RANDOM sets and subsets, the default AREAS and AREASIZE are large enough to contain the entire POPULATION despite an uneven distribution of records among the basic blocks.
2. For variable format data sets, AREAS and AREASIZE are now based upon the largest variable format type. LIMITERRORs will not occur even when the entire POPULATION consists of maximum size records. Note that BLOCKSIZE is still calculated using the length of the fixed portion of the record.

BLOCKSIZE and TABLESIZE

1. Default BLOCKSIZEs and TABLESIZEs are now selected based upon structure type. This makes better use of space. In addition, ADDRESSCHECK, CHECKSUM, and other control words are considered when default BLOCKSIZEs are selected.
2. Extremely large TABLESIZEs are now avoided. For INDEX SEQUENTIAL sets and subsets, DASDL chooses a TABLESIZE which is as small as possible yet minimizes the number of table levels. For INDEX RANDOM sets and subsets which specify a MODULUS, DASDL selects a TABLESIZE which is as small as possible while minimizing the number of overflow tables.
3. BIT VECTOR TABLESIZEs have been optimized to avoid wasted space.

MODULUS

To provide a better distribution of key entries for INDEX RANDOM sets and subsets, DASDL selects an odd MODULUS value.

POPULATION

POPULATION, AREAS, and AREASIZE or any combination of the three may now be specified for a structure. When POPULATION alone is specified, default AREAS and AREASIZE are selected by DASDL based upon the POPULATION. When only POPULATION and AREAS are provided, DASDL uses these values to compute the AREASIZE. Similiarly, DASDL computes AREAS if POPULATION and AREASIZE are stipulated. If both AREAS and AREASIZE are specified, then the POPULATION is ignored.

B. FILE dollar option output

Output produced by the FILE dollar option has been expanded. DASDL lists the following information for each structure.

1. File title and family.
2. Buffers.
3. Options including ADDRESSCHECK, CHECKSUM, etc.
4. Record size and block size for data sets.
5. Key entry size and table size for sets and subsets.
6. Subblock size for ORDERED data sets.
7. Modulus for RANDOM data sets and INDEX RANDOM sets and subsets.
8. Wasted space per block or table.
9. Loadfactor for sets and subsets.
10. Areas, areasize, and population.

All of the information which was listed when the ACRINFO dollar option was set during the Accessroutines compile is also listed by DASDL. The values listed by DASDL are the values used by the Accessroutines; physical options are no longer changed when the Accessroutines are compiled.

C. Updating POPULATION
--------- ----------

The POPULATION option may now be changed using any form of DASDL update. DASDL automatically maintains the existing AREASIZE, BLOCKSIZE, and TABLESIZE if this is necessary. To change the capacity of the structure, simply change the POPULATION.

D2616 DASDL - "INITIALIZE" STATEMENT ZIPS "UTILITY"

When the INITIALIZE statement is present, DASDL zips UTILITY to initialize all data base files. UTILITY will be run when the ACCESSROUTINES and other data base software has been compiled. Formerly, this function was performed by a DASDL-generated program which did an OPEN INITIALIZE.

D2682 DASDL - CONTROL FILE TITLE ON UPDATE

DASDL now uses the data base name stored in the description file to select the input control file during update. Previously, operator intervention was required if the old description file title and the data base name stored in the description file were different.

D2729 DASDL - ADD SUBSYSTEM "ID" DOLLAR CARD

The data base administrator can specify that the data base stack for a particular data base will run in the subsystem of his choice by having a value for the SUBSYSTEM task attribute compiled into the ACCESSROUTINES code file. This can be accomplished in two ways.

1. The ACCESSROUTINES can be compiled with the desired value for SUBSYSTEM label equated.

2. The DASDL dollar option SUBSYSTEM can be set. Then if DASDL is compiled to LIBRARY, and SUBSYSTEM has been set, the value specified on the dollar card will be passed as a parameter to the Workflow Job that DASDL will start. This will cause that value of SUBSYSTEM to be label equated for the ACCESSROUTINES compile. The syntax is $ SUBSYSTEM = "<simple identifier>" where <simple identifier> is the subsystem name.

When the data base stack is created, it will be placed in the globalmost box included in the subsystem. Thus, if * GLOBAL tm Memory is included in the system, the data base stack will be placed in global memory. If it is desired to have the data base stack in local memory, the local memory for the appropriate box should be included in the subsystem definition and the other local memory boxes and the global memory box should be excluded.

* "GLOBAL Memory" is a trademark of Burroughs Corporation

If the appropriate box or boxes are not present when the data base stack is first created, it will be placed in the default subsystem.

During the first or any subsequent OPEN of the data base, if the box in which the data base stack is placed is not visible to the program trying to OPEN the data base, the program will get a DMOPENERROR exception on the OPEN statement.

There is at most one data base stack for a given data base present anywhere on the total system at any point in time. Once built, the data base stack is not moved to another box.

If the subsystem in which the data base is to be placed includes more than one local memory box but excludes global memory, the data base stack will be created in the local memory box in which the program that initially OPENs the data base is running. This situation should normally be avoided for two reasons. First, in general, a subsystem with more than one local memory box without global memory is not very useful. Second, the placement of the data base stack can vary depending on which program happens to OPEN the data base first. When the data base stack is in local memory, only programs running in that local memory box can successfully OPEN the data base.

It is possible to determine which box the data base stack is running in by examining the DBS page of the ODT display.

The SUBSYSTEM attribute is described in the III.1 notes MCP D2405 and CONTROLLER D2535.

D2730 DASDL - CHANGES TO "READAHEAD," BUFFERS SPECIFICATION

The buffer specification is now a dynamic data base parameter which can be changed through the Visible DBS. This provides two capabilities:

1. The number of buffers allocated to a structure may be changed at run time.

2. Serial optimization (read ahead) can be activated by specifying "serial buffers" greater than or equal to 2. (Serial optimization can be deactivated by setting "serial buffers" less than 2.) This feature is available whether READAHEAD has been specified in DASDL or not.

These capabilities make the DASDL READAHEAD attribute unnecessary; therefore, the following changes have been made to DASDL:

1. The user may now specify "serial buffers" (<integer> BUFFERS PER SERIAL USER) from 0 to 254 inclusive. Formerly, a syntax error was generated if READAHEAD were TRUE and "serial buffers" were less than 2. The default for "serial buffers" is 0.

2. The physical attribute READAHEAD will be deimplemented on the DMSII III.3 release. If DASDL encounters READAHEAD on the III.1 or III.2 release, the following warning is issued:

   "THE READAHEAD ATTRIBUTE WILL BE DEIMPLEMENTED ON 3.3"

3. On III.1 and III.2 releases, READAHEAD has the sole effect of controlling the default value of the dynamic "serial buffers" specification:

   a. If either READAHEAD or REBLOCK is TRUE, dynamic "serial buffers" defaults to the DASDL "serial buffers" specification. (The DASDL default is 2 when there is no user specification.)

   b. If neither READAHEAD nor REBLOCK is TRUE, dynamic "serial buffers" defaults to 0.

Please refer to ACCESSROUTINES note D2617 for additional information on dynamic parameters and the Visible DBS.

D2751 DASDL - ZIP STANDARD "WFL" FILE

Previously, when the ZIP option was set and no errors were detected, DASDL built and zipped a job deck to compile the tailored data base software (ACCESSROUTINES, UTILITY, etc.).

Now, under the same circumstances DASDL zips a WFL file titled "DATABASE/WFL/COMPILEACR", which compiles the tailored DM software.

This WFL file can be started anytime to compile a user's tailored data base software. This WFL file must be started with a string parameter which reflects the given situation. The string parameter is a list of "<keyword>=<value>" pairs. The following is a description of the possible keywords, their default values, and their meanings.

Keyword	Default	Description
DB	DEFAULTDB	Data base name
SOURCE	DISK	Family of the DM system software (SYSTEM/= and DATABASE/= files)
OBJECT	DISK	Family on which to put the tailored DM software
DESCRIPTION	DISK	Family of description file
AUDIT	RESET	SET if data base is audited
SUBSYSTEM	SYSTEM	Subsystem identifier from DASDL (For * GLOBAL tm Memory systems only)
PARTITIONS	RESET	SET if partitioned structures exist (if SET, this will enable the WFL to compile PARTITIONCONTROL).
INITPARTITIONS	RESET	SET if the initialization of partitions via the running of SYSTEM/DMCONTROL is desired.
PRE29	RESET	SET if mark level of old description is prior to level II.9
INITIALIZE	RESET	SET if a run is desired of UTILITY which will initialize the data base files.

* "GLOBAL Memory" is a trademark of Burroughs Corporation.

Example :

```
STARTJOB DATABASE/WFL/COMPILEACR ("DB=TESTDB DBPK=DEVELPK AUDIT=SET")
```

This statement compiles all the tailored DM software for an audited data base named "TESTDB" and places that software on a pack labeled "DEVELPK". The name of the new description file is "DESCRIPTION/TESTDB".

By default, DASDL zips the job file titled "DATABASE/WFL/COMPILEACR ON DISK". The title of the WFL file to be zipped may be altered using the ZIPFILE dollar option.

Example :

```
    $ ZIPFILE = "(USER1)MYSPECIALWFL ON MYPACK"
    $ ZIPFILE = "WFL/COMPILEALL"
```

Additionally, the implementation of the WFL zipfile in DASDl caused the deletion of the DASDL source dollar options ACRINFO and ACRDISPLAY and the DASDL compile-time options ACRINFO, ACRDISPLAY, CONTROLCARD and CLASS40. The ACRINFO and ACRDISPLAY options for the ACCESSROUTINES compile can be set in the WFL zipfile directly. The WFL zipfile is always zipped now; the CONTROLCARD option is no longer needed. The desired queue for the zipfile run can be set directly in the zipfile; the CLASS40 option is now eliminated.

D2889 DASDL - "TPSDUALUPDATE" OPTION ADDED

TPSDUALUPDATE has been added to DASDL to allow a data base to be used by both the transaction system and normal DM programs.

When this option is set, a data base may be opened by both a regular DM program (OPEN UPDATE) and a transaction system program (OPEN TRUPDATE). If this option is not set, a mixture of the two types of open results in an open error.

The syntax diagram on Page 4-14 of the DASDL Reference Manual (Form No.5001480) should be changed to show the following:

```
                 |<---------------- , ----------------|
                 |                                    |
-- OPTIONS -- ( -----/1\- ADDRESSCHECK ------------------ ) --|
                   |                                |
                   |-/1\- AUDIT ---------------------|
                   |                                |
                   |-/1\- BACKOUTTOSPT --------------|
                   |                                |
                   |-/1\- DATACHECK1 ----------------|
                   |                                |
                   |-/1\- DATACHECK2 ----------------|
                   |                                |
                   |-/1\- RESTART TRANSACTIONS ONLY -|
                   |                                |
                   |-/1\- STATISTICS ----------------|
                   |                                |
                   |-/1\- TPSDUALUPDATE -------------|
```

D2899 DASDL - CAPABILTIES AND LIMITATIONS

The table in the DMS II DASDL Reference Manual (Form No. 5001480), Page 6-48, is incorrect. The three "NO"s opposite "Change Order in which items are declared" should read as follows:

"NO NO YES".

D2900 DASDL - FORMAT OF A VERIFIED LINK

The diagram on Page 4-98 of the DMS II DASDL Reference Manual (Form No. 5001580) is incorrect. The diagram should show that the disk address in a verified link (L) precedes the value of the verification item (K) in the record for the data set (D).

D2927 DASDL - BLOCK ZERO FORMAT

Page C-52 of the DMS II DASDL Reference Manual (Form No. 5001480) shows block zero format. The format is incorrect; the locations of EMPTY BLOCK and EOF BLOCK are reversed. The order of the blocks should read as follows:

```
CONTROL WORD
    .
    .
    .
CONTROL WORD
EOF BLOCK
CURRENT BLOCK
EMPTY BLOCK
```

DMS II - DASDL

P1038 DASDL - "INVALID INDEX"

DASDL dies with an INVALID INDEX when an update compile is done on a data base containing a partitioned data set. This problem has been corrected.

P1775 DASDL - ELIMINATE "SEG ARRAY" ERROR

DASDL no longer terminates with a SEG ARRAY error while formatting printed output.

P1776 DASDL - RECOMPILATION MESSAGE FOR UPDATE

If DASDL update is performed with the ZIP compiler option set, DASDL now displays a message indicating which items of system software must be recompiled.

P1777 DASDL - PREVENT ABNORMAL TERMINATION

DASDL no longer terminates abnormally when a semicolon is omitted following the DEFAULT statement.

P1803 DASDL - PREVENT "INVALID INDEX"

DASDL no longer terminates with an INVALID INDEX at sequence number 77885000 while creating the block directory.

P1804 DASDL - ENSURE "AREASIZE" IS SPECIFIED PROPERLY

DASDL now ensures that the AREASIZE attribute is specified in the proper units; e.g., BLOCKS is only permitted for the audit trail, while either RECORDS or SEGMENTS is permitted for a data set.

P1840 DASDL - IMPROVE "DASDL" SCANNER

All scanning has been incorporated in a single procedure.

P1920 DASDL - VALID "OPEN PARTITIONS" ATTRIBUTE

DASDL now ensures that the OPEN PARTITIONS attribute is not specified for disjoint structures.

P1921 DASDL - GLOBAL CHAIN NOT OVERLAYED

When a logical data base was encountered which referenced an undeclared data set, DASDL produced unpredictable results. This occurred because DASDL inadvertently overlayed the global chain. This has been corrected.

P2034 DASDL - SYNTAX ERROR FOR ONE GLOBAL DATA SET

All data bases must contain at least one data set. DASDL now gives a syntax error if no data sets are present.

P2191 DASDL - ERRONEOUS KEY CHANGED ERRORS

DASDL generated incorrect KEYERROR text for remaps containing virtual items. As a result, the ACCESSROUTINES reported KEY CHANGED exceptions when remaps were used. This has been corrected.

P2207 DASDL - "INFO" TABLE INCORRECT

DASDL was not storing information in its INFO tables correctly. As a result, the description file produced by DASDL was incorrect. This caused a variety of errors when tailored software was compiled or run. This has been corrected.

P2354 DASDL - TEXT GENERATED FOR GROUP KEY WITH FIELD

DASDL was generating incorrect text for a set whose key was a group which contained a field.

P2355 DASDL - LAST SCAN SAVED

An INVALID INDEX no longer occurs while scanning variable format declarations.

P2356 DASDL - CORRECTS ERROR CHECKING FOR "READONLY" OPTION

DASDL was erroneously generating the message "MUTUALLY EXCLUSIVE OPTIONS" for some specifications of the READONLY option.

P2455 DASDL - PRINT STORE CARD IMAGE

DASDL was printing out each source card image after it was scanned for syntax. This caused DASDL information such as:

"<structure name> IS STRUCTURE <struture #>"

to be printed on the output listing before the card image to which it referred.

DASDL now prints the card image and the associated information in the proper order.

P2456 DASDL - ALLOW "ALTERNATE" SPECS AFTER "PACKNAME"

Following an audit trail PACKNAME specification, DASDL now allows an ALTERNATE media specification.

P2457 DASDL - ALWAYS COMPILE "PARTITIONCONTROL"

The WFL job zipped by DASDL to compile the tailored data base software now compiles PARTITIONCONTROL whenever partitioned structures are specified in the DASDL source. Previously, PARTITIONCONTROL was compiled onlly when partitioned structures were added to an existing data base with no partitioned structures.

P2479 DASDL - MAXIMUM LENGTH "DBNAME" IN QUOTES

DASDL was erroneously displaying the error message "IDENTIFIER EXCEEDS 17 CHARACTERS" for long data base names enclosed in quotes. DASDL was mistakenly counting the quotes in computing the name size. DASDL now allows a full 17 character string enclosed in quotes as a data base name.

P2480 DASDL - DISALLOW DATA SET WITH NO NAMED ITEMS

DASDL now displays an error message for data sets which have no named items.

P2608 DASDL - WARNING FOR "ACRINFO, ACRDISPLAY"

ACRINFO and ACRDISPLY DASDL options were deimplemented with the implementation of the WFL zipfile; however, these options could still be specified in DASDL source without errors. Now, DASDL will give a warning when these options are specified.

P2609 DASDL - "DECKLIST" OPTION

Since DASDL now uses a standard WFL zipfile, the DECKLIST option is now used to list only the parameters with which the WFL zipfile is to be zipped.

P2610 DASDL - WRITE OMITTED CARDS TO LISTING AND "NEWTAPE"

Previously, DASDL was not writing omitted cards to a line printer listing or newtape file. Now, if LIST or NEW dollar options are set, the omitted cards will be written correctly.

P2639 DASDL - INCREASE "TEXTGENMAX" TO "4095"

The TEXTGENMAX define has been changed from 1200 to 4095. Previously, DASDL could get a SEG ARRAY error while generating text for very large compact data sets.

DMS II - DMALGOL

D2759 DMALGOL - "B7000" SERIES "DM" OPTIMIZATION

The **READLOCKNOPURGE** function and the **PURGEASM** statement have been implemented for B7000 series optimization.

The **READLOCKNOPURGE** function has the same syntax as the **READLOCK** function. For B7000 series compilers, if the compiler $ option B7700 is set, **READLOCKNOPURGE** emits code which prevents the purge of associative memory before the readlock operation if the code is executing on a B7000 series machine. If the code is executing on a B6000 series machine, the action of **READLOCKNOPURGE** is identical to that of **READLOCK**. If $B7700 is reset, the code for **READLOCKNOPURGE** is identical to the code for **READLOCK**.

The syntax for the **PURGEASM** statement is the following:

```
-- PURGEASM --%
```

For B7000 series compilers, if the compiler $ option B7700 is set, **PURGEASM** emits code to purge the associative memory if the code is executing on a B7000 series machine. If the code is executing on a B6000 series machine, no action is taken. If $B7700 is reset, no code is emitted for **PURGEASM**.

D2837 DMALGOL - COMPILE TIME EXTENSION TO "<DEFINE DECL>"

The following changes should be made to the **ALGOL** Language Reference Manual (Form No. 5001639):

On Page H-1, change the third line from "and (4)" to "(4) an extension to the <define declaration>; and (5)".

On Page H-4, before heading "Compiler Options", add the following:

> "EXTENSION TO <define declaration>
>
> The following extension to <define declaration> is available when using the compile-time facility:
>
> ```
> <definition>::=<defined identifier><formal symbol part>=<text>#|
>                <defined identifier><formal symbol part>:=<text>#
> ```
>
> If a define identifier is declared with ":=", then any compile-time variables, identifiers, and statements in the <text> are evaluated once, at the time of and in the scope of, the define declaration. If a define is declared with "=", then the compile-time items in the <text> are evaluated upon each invocation of the define identifier and in the scope of the invocation."

D2980 DMALGOL - "DMALGOL" DOCUMENTATION

The DMALGOL language, previously released but largely undocumented, has been documented for the III.1 release. It is described in Appendix E, "DMALGOL Implementation".

DMS II - DMALGOL

P2115 DMALGOL - COMPILE TIME PROCESSOR "INVALID INDEX"

The compile-time processor no longer terminates with an INVALID INDEX on the CTLABELS array. The size of CTLABELS has been increased.

DMS II - DMCONTROL

P1739 DMCTL - "OPEN INITIALIZE" VS. "RECOVERY"

If OPEN INITIALIZE is run against an audited data base which requires Halt/Load recovery, the control file is not locked until Halt/Load recovery has been run successfully. Formerly, OPEN INITIALIZE locked the control file prior to recovery. This prohibited any program except an OPEN INITIALIZE to be run if the recovery terminated unsuccessfully.

If this change is not contained on a DM standard system tape, it must be first compiled into DMCONTROL and a new DATABASE/DMCONTROL must be made with the patch in it. Next, the ACCESSROUTINES must be recompiled to include the new DATABASE/DMCONTROL.

P1922 DMCTL - "INV OP" IN "IODISKADDRESS"

Assume a data base data file had an incorrect version timestamp for some reason. (For example, the file was dumped using UTILITY, but "OFFLINE DUMP" was not specified; then later, the file was loaded by UTILITY using the "COPY" statement.) Assume a program opened the data base inquiry and proceeded to run, because the operator overrode the timestamp checking. Finally, assume another program opens the data base update. Then a timing problem exists if the operator overrides the timestamp checking again for the update program. Unpredictable results might occur including an INVALID OP in the MCP procedure IODISKADDRESS. The problem has been corrected.

DMS II - DUMPDIR

D2708 DUMPDIR - DUMP TAPE DIRECTORY

DMSII is now capable of maintaining a directory which records information about UTILITY data base dumps. Using the directory, the system can determine on which dump tape a row of the data base resides. This information makes data base recovery much easier.

The Dump Directory is entirely optional. It may be enabled and disabled at any time.

Main Directory

Information about dumps is maintained in a two-level directory. Each data base has one Main Directory. It contains one entry for each dump that has been processed (and not deleted) since the directory was enabled.

The Main Directory is stored on the same pack as the data base Control file.

Naming Convention

```
          <data base name>/DMDUMPDIR
```

Dump Directory

There is one Dump Directory for each entry in the Main Directory. A Dump Directory entry is created each time a UTILITY dump is performed. Each Dump Directory contains the dump name and a list of the data base files dumped to the tape. The user may specify the pack on which the dump directories are maintained using the ENABLE statement. If no pack is specified, the dump directories are stored on the same pack as the data base Control file.

Naming Convention

```
          <data base name>/DMDUMPDIR/<dump time>/<dump name>
```

<dump name> The name given to the tape in the UTILITY DUMP statement.

<dump time> The time at which the dump was taken. This number is used when directory entries are manually added or deleted.

Example:

DEC2179152116

This means the dump was taken on December 21, 1979 @ 15:21:16.

DM DUMP DIRECTORY Program

Dump directory information is retrieved and modified using SYSTEM/DMDUMPDIR. DMDUMPDIR provides the following functions:

Syntax:

```
---- ENABLE --------------------------------------------------|
  |           |                                            |
  |           | |<----------------   ,   ----------------| |
  |           | |                                        | |
  |           |-----/1\- PACKNAME -- = --<familyname>------|
  |               |                                    |   |
  |               |-/1\- RETAIN -- = --<integer>-------|   |
  |- DISABLE ----------------------------------------------|
  |        |<------  ,  ------|                            |
  |        |                  |                            |
  |- ADD ---<dumpidentifier>-------------------------------|
  |             |<------  ,  ------|                       |
  |             |                  |                       |
  |- DELETE -----<dumpidentifier>--------------------------|
  |           |                                            |
  |           |- OLDEST --<integer>------------------------|
  |- WRITE -----<filelist>---------------------------------|
  |         | |                                            |
  |- LIST --| |- = ----------------------------------------|
              |- MAINDIRECTORY ----------------------------|
              |                  |<-------  ,  -------|    |
              |                  |                    |    |
              |- DUMPDIRECTORY ---<dumpidentifier>---------|
```

<dumpidentifier>

```
--<dumptime>-- / --<dumpname>--|
```

<file list>

```
  |<------------   ,   ------------|
  |                                |
----<file name>-----------------------|
                 |-<rowselector>-|
```

<rowselector>

```
       |<-------------------- AND -----------------------|
       |                   |<- & ---|                    |
       |                                                 |
-- ( -----/1\- ROW ------------ = --<range options>------- ) --|
        |-/1\- FAMILYINDEX -|                          |
        |                                              |
        |                     |<--------  ,  --------| |
        |                     |                      | |
        |-/1\- ROWLOCK -- = ------/1\- LOCKEDROW ------|
        |                       |-/1\- READERROR -|    |
        |-/1\- PACKNAME -- = --<familyname>------------|
```

<range options>

```
  |<------------   ,   ------------|
  |                                |
----<integer>------------------------|
               |- - --<integer>-|
```

Running DMDUMPDIR

To run DMDUMDIR, label-equate file DASDL to the description file and pass the appropriate input command as a string parameter.

Example:

```
RUN SYSTEM/DMDUMPDIR("LIST =")
FILE DASDL=DESCRIPTION/DBNAME ON DBPACK;
```

Functions

ENABLE This establishes a Main Directory for the data base, causing a boolean in the Control file to be turned on and the global section of the Main Directory to be set up.

PACKNAME This may be used to specify the location of the dump directories. By default, dump directories are stored on the same family as the data base Control file.

RETAIN This may be used to control how many dumps are maintained in the directory. By default, fifty (50) dumps are retained.

Example:

"ENABLE PACKNAME=DBPACK,RETAIN=10"

This causes a directory to be associated with the data base. All Dump Directory files reside on DBPACK. The directory retains information for a maximum of 10 dumps.

DISABLE This disables the Dump Directory and removes all available files associated with it.

ADD This allows entrys to be manually added to the Main Directory. <dumpidentifier> specifies the entries to be added.

Example:

"ADD FEB0179125205/DBDUMP"

This causes dump FEB0179125205/DBDUMP to be added to the Main Directory.

DELETE This deletes entries from the Main Directory and removes the corresponding Dump Directory from disk. <dumpidentifier> specifies the entry to be removed. OLDEST deletes the oldest <integer> entries.

Example:

"DELETE DEC2578172125/DMSDUMP"

This deletes the entry for DEC2578172125/DMSDUMP and removes its Dump Directory file.

WRITE/LIST This displays the name of the dump tapes on which the most current copies of the specified rows reside. WRITE directs output to the printer; LIST sends output to the terminal. When a <filelist> appears, information is listed for the specified files. The equal sign option ("=") displays information for all data base files. MAINDIRECTORY displays the contents of the Main Directory. DUMPDIRECTORY displays the contents of the specified Dump Directories.

Examples:

"WRITE MAINDIRECTORY"

This displays the Main Directory on the printer.

"LIST DB/A/DATA(ROW=1-5)"

This lists the tapes on which the most current copies of rows 1 through 5 of DB/A/DATA reside.

Adding and Deleting Entries
------ --- -------- -------

Adding Entries

Each time a dump is taken, an entry for the dump will be automatically entered in the Main Directory and a Dump Directory file will be created. Dump entries may be added to the Main Directory manually if a Dump Directory exists for the entry.

Deleting Entries

Entries may be deleted automatically or manually. Automatic deletion occurs if the number of entries being added causes the number of entries in the Main Directory to exceed the number to be retained. The oldest entries will be deleted.

DMS II - DUMPDIR/LIBRARY

P2582 DUMPDIRLIB - LOCATION FOR LENGTH OF DUMP DIRECTORY

The complete dump directory was not being read in because the length of the directory was not being picked up from the proper location. This problem has been corrected.

DMS II - INQUIRY

P1825 INQ - DISPLAY OF ALL-FRACTION ITEMS

An all-fraction numeric item (e.g., NUMBER(5,5) in DASDL) was displayed incorrectly. The digit in the record immediately preceding the item was displayed before the decimal point as if it were part of the value. This has been corrected.

P1826 INQ - DISPLAY OF NULL ITEMS

When the option NULL=FALSE is used, null items are not displayed in TAB or SINGLE format. In the case of TAB, it was possible for extraneous characters to appear at the end of one value prior to the next item, instead of the required blanks. This has been corrected.

P2012 INQ - VALUE OUT OF RANGE

An erroneous VALUE OUT OF RANGE error might occur with certain decimal values used in a selection condition. This has been corrected.

P2208 INQ - "INVALID INDEX" USING DEFINES

An INVALID INDEX fault occurred if more than one simple define were used in a single input. This has been corrected.

P2251 INQ - DISPLAY OF NUMERIC ITEMS

The following problems have been corrected:

1. Invalid printing of units digit (always 0) when displaying or reporting a number defined in DASDL with only fractional digits.

2. Display of 0 value not properly aligned when displaying a number with no fractional digits.

3. INVALID INDEX when displaying a zero-valued numeric item defined in DASDL as the last item of a data set record and defined in DASDL with no fractional digits.

P2276 INQ - DISPLAY LIMIT FOR "OPTION PRINTER"

When OPTION PRINTER was set and a <limit> was specified for a DISPLAY, all records which meet the selection criteria were automatically output to the printer. The <limit> specification was essentially ignored. This problem has been corrected. Now, if a <limit> is specified for the display, after this number of records is printed, the following message will be displayed at the terminal:

"# <limit> RECORDS FOR <structure name> PRINTED"

A NEXT command is required to display additional records.

P2389 INQ - FIND VIA SELF-CORRECTING LINK ITEM

A SEG ARRAY error sometimes occurred when selecting a record via a self-correcting link item. This problem has been corrected.

P2481 INQ - FAULT ALPHA CONTROL ITEM BREAK

Only the first item of an alpha field was used to check for a control item change. Thus, control breaks for alpha control items were sometimes not properly detected. This problem has been corrected.

P2500 INQ - USE OF DEFINES IN CERTAIN CONTEXTS

Invocation of defines in certain contexts (e.g., at the beginning of a REPORT statement or SET statement) would cause an INVALID INDEX. This problem has been corrected.

P2501 INQ - LOSS OF MINUS SIGN

The minus sign for a negative value was lost when a signed item was printed via the REPORT statement. This problem has been corrected.

P2612 INQ - COMPARISONS OF LITERALS

Comparison of an alpha item of a length greater than 275 characters with a literal in a selection expression caused a SEG ARRAY error. This problem has been corrected.

DMS II - INTERFACE

P2158 INTERFACE - "INVALID INDEX" IN DATA BASE INTERFACE

Logical data bases which contain global data could cause data base interface to die with an INVALID INDEX. This problem has been corrected.

P2249 INTERFACE - "GROUP" ITEMS INVOKED INCORRECTLY

Remaps which contained groups having HIDDEN items were not invoked correctly. Items which followed the group were erroneously included in the group. This problem has been corrected.

P2357 INTERFACE - LINK TO SET QUALIFICATION

Previously, INTERFACE passed an incorrect identifier to the compilers to qualify a link which referred to a set. The erroneous output on the invocation listing was the following:

```
    <linkname> REFERENCE TO <setname> OF <setname>
```

Correct output now is the following:

```
    <linkname> REFERENCE TO <set name> OF <data set name>
```

DMS II - LOADDUMP

P1923 LOADDUMP - "INVALID INDEX" WITH DELETED STRUCTURE

A problem has been corrected in which LOADDUMP would terminate with an INVALID INDEX if the specified data set had been deleted and added back into the DASDL.

P1924 LOADDUMP - TITLE ATTRIBUTE ERROR

A problem has been corrected which produced a file attribute error on DASDL.TITLE when DEBUG was set and a logical data base was specified.

DMS II - PRINTAUDIT

D2618 PRINTAUDIT - DECODE DATE AND TIME

PRINTAUDIT will now decode all date-timestamps in records containing them. In interactive mode, this information will only be displayed if hex dump information is also displayed.

DMS II - PRINTAUDIT

P2027 PRINTAUDIT - HEX DUMP OF BLOCK

If PRINTAUDIT detects a timestamp or audit block serial number discontinuity, the block is printed in hex. Formerly, it tried to print it normally and often got a fault-DS because the block was abnormal. Now, if this is the case, at least a hex dump of the block will be printed.

P2502 PRINTAUDIT - BAD CHECK FOR LABEL EQUATION

Because of the use of the FAMILYNAME file attribute, PRINTAUDIT was incorrectly checking for a label-equated audit file. The problem has been corrected.

P2504 PRINTAUDIT - FAULT ON PARTIAL RECORD

PRINTAUDIT could fault with an INVALID INDEX if the first record of a file was a partial record. The problem has been corrected.

DMS II - PROPERTIES

D2832 PROPERTIES - "DUPLICATES" SUBCATEGORY "2" EXPLANATION

The explanation of subcategory 2 for a DUPLICATES exception in the DMSII Host Reference Manual (Form No. 5001598), Page B-2, should be corrected to read as follows:

"DUPLICATES

2 Duplicates not allowed in this subset or data set record already in subset (INSERT)."

DMS II - PROPERTIES

P2361 PROPERTIES - OVERLAPPING USE OF OPENERRORS

The same value of openerror was being used for two different purposes. This error has been corrected.

P2549 PROPERTIES - LEVEL FOR "RECONINFO" FILES

The format of the RECONINFO files changed slightly on III.1; therefore, the format level of the file has been incremented.

P9236 PROPERTIES - OPTIMIZE EVALUATION OF CHECKSUM LOCATION

The code to access checksum locations in buffers has been optimized.

DMS II - RECOVERY

D2659 RECOVERY - AUDIT HANDLING ENHANCEMENTS

A number of changes have been made to RECOVERY's audit handling pro- cedures which provide some new capabilities and enhancements.

1. Improved positioning at end of audit for Halt/Load recovery.

   RECOVERY will now position both primary and secondary audits in parallel when finding the end of the audits for the Halt/Load recovery and rollback. As a side effect of this change, RECOVERY will ensure that the ends of the primary and secondary audits are the same. Any errors encountered in the audits while positioning will be ignored if they can be bypassed via the duplicated audit. In addition, RECOVERY will now expect audit tapes to end with a good end-of-file and disk audits to end with a good end-of-file or the stopper pattern. If RECOVERY finds an audit that ends abnormally, it will dump to the printer the bad block at the end of the audit and the last good block found on the audit and then terminate. COPYAUDIT will be required to fix the audit file (if indeed the problem is correctable.)

2. Better error recovery on audit file errors when using duplicated audit.

   RECOVERY will now attempt to recover from any type of errors found on the primary audit by using the secondary audit. Previously, only hard I/O errors and checksum errors would be retried from the secondary audit. Now soft errors like timestamp discontinuities and audit block serial number errors will also cause the secondary audit to be used.

3. Last resort error recovery on audit file errors.

   If RECOVERY cannot recover from an error on the primary audit by using the secondary audit (or no secondary audit exists), RECOVERY will close and release the current audit file and then wait for an operator OK to reposition and retry the operation. This will give the operator the chance to move an audit tape or pack to possibly correct the problem. If repositioning fails, RECOVERY will terminate with a fatal error.

4. Recovery from write errors on audit.

   When RECOVERY goes to fix the last block on the audit in the case of Rebuild, Rollback, or Halt/Load recovery, if a write error is detected, RECOVERY will release the audit and wait for an operator OK to reposition and retry. In particular, if the write ring was not inserted in the final audit tape for Rebuild and Rollback, this will allow the write ring to be inserted and RECOVERY to continue.

5. Additional diagnostics on audit errors.

   Whenever an unexpected error condition is encountered when reading an audit file, the audit block and the error condition encountered are listed on the printer before attempting to recover from the error.

6. Locating audit files.

   Audit files can now be used and read by RECOVERY from tape or disk or any pack regardless of the declared source of audits in DASDL. If RECOVERY cannot find an audit file, RECOVERY will wait on an ACCEPT message on the console. The operator can respond "OK" to retry looking for the audit or can enter "FAMILY = <family name>" to divert RECOVERY to a different audit family. Entering "FAMILY = DISK" will cause RECOVERY to look on the DISK family for the audits. This alternate audit family will be remembered by RECOVERY and will be searched for all other audits for the rest of the run. A different alternate audit family may be specified for the duplicated audit from the primary audit.

   If for some reason an audit file is missing (possibly lost or misplaced) and Rebuild, Rollback, or Reconstruct requires that audit file, RECOVERY will wait on an accept which will allow the operator to enter "USE DUP" to indicate that the desired audit file is missing but that the secondary audit is present. The secondary audit will then be opened and used.

D2807 RECOVERY - ENHANCEMENTS TO QUICKFIX

The III.0 Quickfix capability is useful for recovering from transient failures in the electronic parts of the I/O hardware. However, it was not so useful for recovering from solid failures in the magnetic recording media itself. The present ehancements to the Quickfix capability address this problem.

Quickfix now has a copy option in which the locked out rows of the data base files to be rebuilt are first copied to temporary files. Any compare or I/O errors are retried, and any errors which are irrecoverable are recorded. During the initial reverse scan of the audit (the prepass), such I/O errors can be "canceled" by encountering a BLKIMG audit record which spans the sectors in error. The BLKIMG audit record is written to the temporary file. Thus the write of the block image can succeed even if the corresponding sectors in the data base file cannot be written. Such irrecoverable errors can also be "canceled" during the prepass if an appropriate decrease in end-of-file occurs in the audit.

The UTILITY syntax to initiate the new form of Quickfix is as follows. RECOVER (ROWS USING AUDIT ONLY, <limit>) This is the same as the syntax to initiate the old (3.0) form of Quickfix. The syntax to initiate the old form is as follows. RECOVER (ROWS IN PLACE USING AUDIT ONLY, <limit>) This change to the syntax enables users to reap the added benefits of the new form without changing any existing syntax. It also makes the syntax consistent with the two forms of DATARECOVERY which use backup dumps.

It is expected that the old form of Quickfix will seldom be used, since the new form is more likely to succeed, and since many transient I/O problems can be handled by the III.1 I/O retry mechanism in the ACCESSROUTINES without ever locking out a row.

Two other enhancements to Quickfix have been made. First, the prepass will terminate as soon as the recoverability status of all rows to be recovered has been determined, even if the <limit> has not yet been reached. The other enhancement is that a displacement from the current end of the audit can be specified, resulting in UTILITY syntax which is invariant over time. The displacement can be specified as a certain amount of time or a certain number of audit files. For example, to specify that the prepass is to go back one hour and thirty-five minutes in the audit, <limit> would be as follows.

```
    LIMIT = * - 1:35
```

To specify that the prepass is to go back through the current audit file plus the six previous audit files, <limit> would be as follows.

```
    LIMIT = * - 7 AUDIT FILES
```

The complete UTILITY syntax is presented in III.1 UTILITY note D2835.

DMS II - RECOVERY

P1805 RECOVERY - ORDERED "EOF" SET INCORRECTLY

If the last block allocated in an ORDERED data set was also the last block of a disk row, Halt/Load recovery would die with a FAILED TO SET END-OF-FILE ON DB FILE error. This has been corrected.

P1806 RECOVERY - TAPE AUDIT HANDLING

When the audit file is used by RECOVERY during a rebuild or rollback was on tape, the fatal error UNEXPECTED AUDIT BLOCK SERIAL NUMBER would sometimes occur erroneously. This would happen in the final Halt/Load recovery phase when backing up two control points, if either control point was in the last block of the previous audit file. This has been corrected.

P1807 RECOVERY - "48" USERS OF RESTART DATA SET

When more than 47 invokes of the restart data set were active simultaneously, Halt/Load recovery and abort recovery and the final phase of rebuild and rollback could get an INVALID OP when processing the corresponding section of the audit. This problem has been corrected.

P1808 RECOVERY - AUDIT REEL SWITCH

When the primary audit was on pack, but the actual audit was on tape, Halt/Load recovery and rebuild and rollback (if started at the final Halt/Load recovery phase) would get a fatal error: AUDIT REEL SWITCH SHOULD NOT BE NECESSARY. This has been corrected.

P1809 RECOVERY - INCORRECT TIME PRINTED

When printing the stop time for a rebuild or rollback, RECOVERY and UTILITY would print the seconds incorrectly. RECOVERY printed the wrong seconds; UTILITY printed the wrong fractional seconds. The actual time value used was correct, however. RECOVERY now prints the correct time truncated to the nearest second; UTILITY now prints the correct time truncated to the nearest millisecond.

P1925 RECOVERY - "ABORT" GETS "WAITING ON ROWLOCKOUTAUDIT"

If ABORT recovery encountered an error, the ACCESSROUTINES would get an RSVP "WAITING ON <db>/ROWLOCKOUTAUDIT" and the data base would hang until it was DSed, because ABORT called the ACCESSROUTINES before closing the rowlockout audit file, which had been opened exclusively. The problem has been corrected.

P2013 RECOVERY - AUDIT POSITIONING ERROR

There is no longer a problem of positioning the audit at the end of the Quickfix prepass on partitioned data bases.

P2014 RECOVERY - CORRECT REBUILD RESTART

Rebuild recovery no longer goes into a processor loop when restarting.

P2101 RECOVERY - DATA BASE CORRUPTION BY ROLLBACK

Rollback no longer corrupts the data base when terminated in the region of an audit where a prior Halt/Load recovery had taken place.

P2102 RECOVERY - REBUILD, ROLLBACK RESTART ERROR

If RECOVERY were doing a rebuild or rollback and died or was Halt/Loaded between the afterimage phase (if a rebuild) or the beforeimage phase (if a rollback) and the Halt/Load recovery phase, it would not restart properly and would terminate abnormally. This problem has been corrected.

P2159 RECOVERY - MAKE "ODT" ARRAY LOCAL TO INTERRUPT

The array used for displaying messages has been made local to the software interrupt to avoid any possible conflict.

P2186 RECOVERY - OPENERROR MESSAGE IMPROVED

If RECOVERY gets an open error in opening the data base, the category, subcategory and structure number are now displayed.

P2284 RECOVERY - PREVENT CHECKSUM ERROR

RECOVERY will no longer terminate with spurious checksum errors when an unordered or random data set is extended.

P2335 RECOVERY - "EOF" TOO LARGE

When backing out changes to certain structures (I-S, O-L, UO-L, I-R and compact data sets) that extended the end of file, RECOVERY would set the end of file one block greater than the actual value. This has not caused any problems previously; however, it could cause false checksum and address check errors on a III.1 UTILITY dump.

P2413 RECOVERY - ABORT BACKS UP TOO FAR

ABORT No longer improperly backs up the data base. Formerly, it unconditionally backed out to syncpoint when it could have backed up to a null transaction point.

P2414 RECOVERY - "REBUILD" AND "RECONSTRUCT" BACK UP TOO FAR

Formerly, when REBUILD and RECONSTRUCT encountered a RECOV record in the audit file while applying AIs, they would behave as if they were in the Halt/Load phase of recovery (i.e., they would apply BIs to the last null transaction point preceding the RECOV record, back up two control points, and apply AIs up to the last null transaction point before continuing on to apply AIs beyond the RECOV record). They no longer unnecessarily back up these two control points (and apply AIs to the last null transaction point);. instead, they begin applying AIs beyond the RECOV record immediately after applying the BIs to the null transaction point.

P2449 RECOVERY - "DS" ABORT CAUSES SYSTEM HANG

Abort RECOVERY no longer hangs the system with a fatal STACK OVERFLOW when DSed.

P2458 RECOVERY - BAD RETRY LOGIC

If the write ring was not in place at the time REBUILD or ROLLBACK attempted to end-of-file the audit and the RETRY option was used, then the second attempt to end-of-file the audit would fail with an I/O DESCRIPTOR error or possible corruption of the audit. The RETRY will now be done correctly.

P2459 RECOVERY - TIMING PROBLEM IN "REBUILD/ROLLBACK" RESTART

If REBUILD or ROLLBACK were DSed when attempting to fix up the last audit block on tape, a subsequent restart could fail. REBUILD would fail with "MUST REBUILD TO DATE-TIME > END OF DUMP OF END OF AUDIT".

P2482 RECOVERY - "FAILED TO SET END-OF-FILE" MESSAGE

RECOVERY no longer terminates with "FAILED TO SET END-OF-FILE ON DB FILE" when attempting to set end of file on an ORDERED data set.

P2483 RECOVERY - QUICKFIX MAY FAIL ON CHECKSUMMED DATA BASE

Quickfix could fail with a SEG ARRAY error in its prepass if one of the structures being fixed had the largest block size in the data base, was checksummed, and had a blocksize one larger than a multiple of 30. Data recovery was ignoring the checksum word when computing the maximum buffer size to use.

P2503 RECOVERY - COPE WITH EMPTY AUDIT FILES

Halt/Load recovery and PRINTAUDIT could fail if an audit contained block zero only and no data. The problem has been corrected.

P2505 RECOVERY - RECONSTRUCT IGNORES QUIET POINT

Upon encountering a recovery point in the audit, Reconstruct could incorrectly apply before images. If the restart data set was not being reconstructed, Reconstruct would apply before images until a control record was encountered, rather than stopping on a quiet point (a BTR record indicating no one in transaction state) as it should. This problem, which could potentially corrupt the structure(s) being reconstructed, has been corrected.

P2640 RECOVERY - "ZOT CURSTAMP" AT END OF AUDIT

Previously, when READAUDIT encountered the end of a disk audit, it did not always set the audit block timestamp check variable to undefined. This caused a subsequent call on READAUDIT to report a TIMESTAMP DISCONTINUITY error. READAUDIT now sets this variable to undefined upon encountering end of audit.

P2641 RECOVERY - "FIXLASTAUDITBLOCK" AUDIT POSITIONING

Previously, for disk audits FIXLASTAUDITBLOCK left the audit positioned in such a way that, under certain conditions, a subsequent attempt to read the audit would report a TIMESTAMP DISCONTINUITY error. This problem has been corrected.

DMS II - REORGANIZATION

D2731 REORG - "REORGANIZATION" OPTIMIZATION

The following enhancements and changes are related to improvements made in the overall efficiency of the REORGANIZATION program.

DEFAULTS OF GENERATION

The following defaults apply to the GENERATE statement:

1. If UPDATE is specified in DASDL, then every "REORGANIZEd" indexing structure in the data set family which is not explicity generated will have the default generate:

   GENERATE <istr1> USING <istr1>;

2. If the data set is generated, then all Bit-vectors and manual subsets in the data set family will have the default generate:

   GENERATE <istr1> USING <istr1>;

3. If the data set is generated, then all index sequential and ordered list structures in the data set family with DUPLICATES allowed, but FIRST or LAST unspecified, will have the default generate:

   GENERATE <istr1> USING <istr1>;

   On previous releases the above default was inappropriately taken for index random structures in the data set family when the data set was generated and the stated DUPLICATES condition existed. This is no longer the case. Such an index random structure will be merely fixed up by default.

CHANGES TO THE DATA COPY SPECIFICATIONS

The following is the simplified syntax of the <data copy specs>:

```
 |<-- , -|
 |       |
----<str>--- ( COPY TO ---<medium>------------------------------ ) ; -|
                          |          |                          |
                          |          |- , COPY BACK ------------|
                          |                              |      |
                          |                              |- AT END -|
                          |                                     |
                          |- FINAL MEDIUM ----------------------|
```

The NO COPY option has been eliminated. In addition, COPY specifications can now only be given for structures which are generated, either explicitly or by default. An improved NO COPY algorithm offering a significant speed up in the fixup process has been implemented which will now be automatically employed to do all fixups "in place."

In order not to immediately invalidate existing specifications, COPY specifications for structures requiring fixup will be ignored and a warning message will be issued for this and the following release.

NEW RESTRICTION ON THE GENERATE STATEMENT

A new restriction has been placed on the GENERATE statement which facilitates better optimization of the reorganization processes. If <istr1> is generated USING <istr2>, then <istr2> cannot be generated USING <istr1> or using any other index structure generated using <istr1>. For example, the following GENERATE statements are invalid:

```
GENERATE S1 USING S2, S2 USING S1;
GENERATE S1 USING S2, S2 USING S3, S3 USING S1;
```

NEW CRITERIA FOR SEQUENCING REORGANIZATION PROCESSES

New criteria is now used to determine the sequence of certain reorganization processes. If <istr1> is generated using <istr2>, then any GENERATE or FIXUP required for <istr2> is done before the GENERATE for <istr1>. In addition, if <istr1> is generated using <istr2> and the data set which these indexes span is generated, then the generation of <istr2> is not done simultaneously with the generation of any other index structure which is also generated using <istr2>. These criteria facilitate better optimization of the reorganization process and permit the SEQUENCE statement to be completely optional.

To enforce the new criteria when explicit sequencing is specified, the following new restrictions are placed on the SEQUENCE statement:

1. If <istr1> is not the <prime> set and is generated USING <istr2> and <istr2> is generated or fixed up, then <istr2> must appear before <istr1> and cannot be ANDed with <istr1>.

2. If the data set is generated and <istr1> is generated USING <istr2>, then <istr1> cannot be ANDed with any other index structure which is also generated USING <istr2>.

As a result of the new sequence criteria, a SEQUENCE statement is no longer required when REORGANIZATION specifications similiar to the following are given:

GENERATE D, S1 USING S1, S2 USING S1;

Assuming D is a data set and S1 and S2 are spanning sets, a SET SEQUENCE ERROR was previously given forcing either a SEQUENCE statement or <data copy specs> to be specified. The above change eliminates the need for this additional specification.

SPEED UP OF CERTAIN GENERATE PROCESSES

The new fixup algorithm has been employed to significantly increase the speed of certain generate processes. When an index structure is generated USING itself or another index structure (i.e., <istr1> USING <istr1> or <istr1> USING <istr2>), the new fixup algorithm is now applied when required within the index structure GENERATE process to fixup the USING structure (which is usually the old structure) in-place. This occurs before the actual generate of the new index structure begins and provides for improved overall efficiency of the reorganization process.

D2732 REORG - ELIMINATION OF "REORGPENDING"

A SYSTEM/DMCONTROL("UPDATE") run is no longer zipped immediately after a DASDL UPDATE run with REORGANIZATION specified, nor is the control file marked as "REORGANIZATION PENDING". Thus the control file is unchanged by the intended reorganization until the actual run of the REORGANIZATION program.

If the user should execute SYSTEM/DMCONTROL("UPDATE") immediately after a DASDL REORGANIZATION run, a warning is given and the program is terminated normally (i.e., essentially a no-op run occurs).

This change was made to facilitate continual on-line operation of the data base up until the time when the REORGANIZATION program is run. In addition, it facilitates proper dumping of the control file and data base files at this point.

D2736 REORG - DATA BASE REORGANIZATION

The following represents an updated version of the original documentation on REORGANIZATION which appeared in the II.8 D-Notes. All documentation changes which have occurred since II.8, including III.1 changes, have been included.

PURPOSE

During the life of a data base, the uses and demands on it may occasionally change. For instance, the data base administrator may find it necessary or desirable to:

REORGANIZATION CAPABILITIES

1. Reorder data sets according to a specific index set (prime index) if sequential accessing through that set involved the greatest number of applications for the data set.

2. Garbage collect on both data sets and indexing structures. During the normal running of a data base, the physical size of a file for a structure does not decrease. Thus, if a data set once had a large population and then reverted to a smaller population, the data set would still have the physical size equivalent to the largest population. However, it would consist of a small population of data records and a large population of deleted (available) records. In anticipation that this data set would never increase to the former high population (within some reasonable time frame), the available "holes" could be removed to return disk or pack space back to the MCP. This process is known as "garbage collection".

3. Generate sets or automatic subsets if a new method of access is desired for the data set.

4. Balance index tables and achieve a uniform coarse/fine table distribution to optimize accesses through sets.

5. Change structure physical attributes such as blocksize, areas, population, modulus, load factor, etc.

6. Change records by adding record fields.

NOTE: If the user wishes to do this, he should first read Section 5 of the DMSII DASDL Reference Manual (Form No. 5001480) - "Remaps and Logical Data Bases".

To accomplish any of the above obviously requires physical transformations of structures. These transformations are known as "data base reorganization".

This document describes the data base reorganization program capabilities.

USER DESCRIPTION

Database Reorganization must be performed with the data base off-line. The object of one "Reorganization run" is to modify at most one data set and its spanning sets. Thus, a reorganization run may be applied to any data set and any of its spanning sets, or any spanning sets with a common data set.

COMPONENTS OF REORGANIZATION

REORGANIZATION
STEPS

```
          \--------------\
           |              |-|
           | DESCRIPTION/ | |
           |  "DB-NAME"   | |
           |              | |
          /--------------/  |
                            |
         \---------------\  |   |---------------|     \---------------\
          |               | |-> |               |      |               |
2&7       |    DASDL      |---> |    DASDL      |----->|  DESCRIPTION/ |
          |   SYMBOLIC    |     |   COMPILER    |      |   "DB-NAME"   |
         /---------------/      |---------------|     /---------------/

         \---------------\
          |               |
          |  DESCRIPTION/ |-|
          |   "DB-NAME"   | |   |---------------|     \---------------\
         /---------------/  |-> |   SYSTEM/     |      | DESCRIPTION/  |
3                               |  BUILDREORG   |----->|REORGANIZATION/|
          /--------------|  |-> |  PROGRAM      |      |  "DB-NAME"    |
         /               |  |   |---------------|     /---------------/
         |    USER       |--|
         |    INPUT      |
         |---------------|

         \---------------\
          | DESCRIPTION/  |
          |REORGANIZATION/|-|
          |  "DB-NAME"    | |   |---------------|     \---------------\
         /---------------/  |-> |               |      |               |
4                               |   DMALGOL     |----->|REORGANIZATION/|
         \---------------\  |-> |   COMPILER    |      |   "DB-NAME"   |
          |               | |   |---------------|     /---------------/
          |   DATABASE/   |-|
          |  REORGSYMBOLIC|
         /---------------/


                                                       \---------------\
                                                      \---------------\ |
                                                       |               ||
                                                    |->| INTERMEDIATE  ||
          \---------------\                         |  |    FILES      ||
         \---------------\ |    |---------------|   | /---------------//
          |               ||    |               |<--|
6         |   DATABASE    ||--> |REORGANIZATION/|      \---------------\
          |    FILES      || -> |   "DB-NAME"   |<--|  \---------------\ |
         /---------------//  |  |---------------|<-||  |               ||
                             |                     ||->|  REORGANIZED  ||
         \---------------\   |                     |   |     FILES     ||
          |               |  |                     |  /---------------//
          |  "DB-NAME"    |--|                     |
          |  CONTROL      |                        |  \---------------\
         /---------------/                         |   |               |
                                                   |-->|  "DB-NAME"/   |
                                                       |   CONTROL     |
                                                      /---------------/

         \---------------\        |---------------|   \---------------\
          |               |       |               |    |               |
6&7       |  "DB-NAME"/   |-----> |   SYSTEM/     |--->|  "DB-NAME"/   |
          |  CONTROL      |       |  DMCONTROL    |    |   CONTROL     |
         /---------------/        |---------------|   /---------------/
```

USER STEPS FOR REORGANIZATION

The data base administrator must decide which data base structures require reorganization. The data base administrator may perform as many "reorganization runs" as necessary to achieve the desired resultant data base. A "reorganization run" is the application of this package to at most one data set and its spanning structures and consists of the following steps:

1. If step 2 below is to be done, a backup copy of the DASDL symbolic and description file should first be made.

2. (Optional) DASDL run with UPDATE option set and REORGANIZE clause specified. DASDL UPDATE is necessary when physical attributes such as AREASIZE, BLOCKSIZE and TABLESIZE are changed, and when new fields are added to records. The new description file will have the update level incremented. Only user programs may be compiled against the description--ACCESSROUTINES, etc., will get syntax errors. Such user programs will not run with the old (i.e., current) ACCESSROUTINES, however, because their DASDL UPDATE level is greater. Old programs can still run with the old ACCESSROUTINES to update the data base.

3. SYSTEM/BUILDREORG run. The user must create input to the SYSTEM/BUILDREORG program specifying the structures to be reorganized. In addition, resource allocation and various other options may be given. SYSTEM/BUILDREORG syntax checks this input and if all structures specified belong to the family of one data set and its spanning structures and other rules are adhered to, then the program will:

   A. Create the "DESCRIPTION/REORGANIZATION/<data base name>" file containing the data base description and the reorganization specifications.

   B. Generate a report which shows both the user and default reorganization specifications. All structures which will be changed by the reorganization program are noted on this report.

4. DMALGOL compilation of the actual reorganization program, "REORGANIZATION/<data base name>". This compilation utilizes "DATABASE/REORGSYMBOLIC" and "DESCRIPTION/REORGANIZATION/<data base name>". This compilation may optionally be "zipped" from SYSTEM/BUILDREORG.

5. A backup copy of the control file and all data base files should be made. If dumping the entire data base is impossible, at least the control file and all data files which will be reorganized or which require address fixup should be dumped.

6. "REORGANIZATION/<data base name>" run performs the actual data base reorganization. The run, which is actually performed in two or three phases, is explained later in more detail. (See "REORGANIZATION RUN".) The data base must be taken off-line and the following files must be present at some time during this run:

   A. All structures to be reorganized.

   B. If a reorganized structure has a structure with links pointing to it, then all such linking structures (for address fixing).

   C. If a reorganized structure is embedded, then its master structure (for root address fixup).

   This program ignores all FAMILY statements at run time. The reorganization program always expects a structure to be present on the FAMILY resulting from the specifications in the last DASDL run; thus, when the <disk/pack> option for a structure is changed as a result of a DASDL UPDATE run, REORGANIZATION expects to find the structure on the new device. The structure can be copied from the old device to the new device using Library Maintenance, or the old device can be appropriately relabeled.

   If Step 2. above is done, the REORGANIZATION program immediately runs the control file program to update the control file. (Consequently, the description built by BUILDREORG must be present at this time, since it is needed by the control file program.) The control file is marked as in exclusive use by REORGANIZATION. The new structures are added to the control file at this time, and the DASDL UPDATE level is incremented in the control file. Until this step, the old ACCESSROUTINES were able to run. Now, they will get an open error on the control file because its DASDL UPDATE level is greater and because it is marked as being in exclusive use by REORGANIZATION. The REORGANIZATION program then reorganizes the data base. In the (final) remove phase, the REORGANIZATION program unlocks the control file from from the "IN EXCLUSIVE USE BY REORG" state.

7. If Step 2. above is done, then a normal DASDL UPDATE run (with the REORGANIZE clause(s) removed) is necessary. The purpose of this run is to take the description file out of "reorganize" mode and restore it to a state for running the data base. This also notifies DASDL that the "reorganization run" has been successfully completed and allows the new ACCESSROUTINES, etc., to be compiled. The user can then run all programs compiled against the current or previous descriptions with the new ACCESSROUTINES.

8. A backup copy of the new control file and all data files which were generated or fixed up should be made because of the discontinuity created in these files by their reorganization.

9. The reorganization program should be discarded and never reused.

OVERVIEW OF REORGANIZATION ALGORITHM

Reorganization is not done in place. Thus, the reorganization program will use an old structure as input and output a new structure onto the proper medium. For very large structures there may be severe space problems. Therefore, reorganization provides an optional two step algorithm which uses an intermediate medium. After input is completed, the reorganization program removes the old structure and copies the new structure from the intermediate medium to the final medium.

The user must determine which structures should be reorganized to intermediate mediums.

Structures which are not reorganized but have links which point into reorganized structures must be updated in the reorganization process. These "fixups" are automatically performed in place.

No partitioned structure or structure with a partitioned master may be reorganized nor may any data set being linked to by a partitioned structure be reorganized.

SPECIFICATION OF THE REORGANIZATION PROCESS (BUILDREORG)

SYSTEM/BUILDREORG is used to control the reorganization process. The input to BUILDREORG specifies which structures are to be reorganized, the intermediate mediums to use, and various other options.

SYNTAX:

Control program run:

```
? RUN SYSTEM/BUILDREORG
? FILE DASDL=DESCRIPTION/"<data base name>"
? DATA CARD
    .
    .   (user specified reorg &
    .                   resource alloc.)
    .
? END
```

For the following syntax let

<d>	data set
<str>	structure
<istr>	indexing structure (set or subset)
<id>	identifier
<db-name>	data base name

```
--<generation identification>------------------------------------------->
                                    |                              |
                                    | |<-------------------------| |
                                    | |                          | |
                                    |---<sort environment specs>---|

>----------------------------------------------------------------------->
   |                      | |                               |
   | |<-----------------| | |-<procedure sequence specs>-|
   | |                  | |
   |---<data copy specs>---|

>-----------------------------------------------------------------------|
   |                                      |
   |-<reorganization global control specs>-|
```

<generation identification>

```
-- GENERATE ------------------------------------------------------------>
              |                           |
              |-<d>-----------------------|
                       |                  |
                       |- ORDER BY --<str>-|

>-----------------------------------------------------------------------|
   |                                     |
   | |<------------- , -------------|   |
   | |                               |   |
   |---<istr1>---------------------------|
                |                 |
                |- USING --<istr2>-|
```

<sort environment specs>

```
  |<-- , -|      |<------------ , ------------|
  |       |      |                            |
----<str>--- ( -----/1\-<extract specs>-------- ) -- ; --|
                  |                         |
                  |-/1\-<sort specs>-------|
                  |-/1\-<loadfactor specs>-|
```

<extract specs>

```
-- EXTRACT KEYS TO --<medium>--|
```

<sort specs>

```
               |<--------------- , --------------|
               |                                 |
-- SORT USING -----/1\-<integer>-- TAPES -----------|
                 |-/1\-<integer>-- SEGMENTS ---|
                 |-/1\- FAMILYNAME = --- DISK -|
                                      |- PACK -|
                                      |-<id>---|
```

<loadfactor specs>

```
-- LOADFACTOR = --<integer>--|
```

<data copy specs>

```
  |<-- , -|
  |       |
----<str>--- ( COPY TO ---<medium>------------------------------- ) ; -|
                        |          |                          |
                        |          |- , COPY BACK ------------|
                        |                          |- AT END -|
                        |- FINAL MEDIUM ----------------------|
```

<procedure sequence specs>

```
             |<---- AND ---|
             |  |<- , ---| |
             |  |        | |
-- SEQUENCE ---<str>--------- ; --|
```

<reorganization global control specs>

```
  |<------------------------------------------------------------|
  |                                                             |
------/1\- INTERNAL FILES -- ( FAMILYNAME = --- DISK --- ) -- ; ------|
    |                                         |- PACK -|         |
    |                                         |-<id>---|         |
    |-/1\- TASKLIMIT = --<integer>-- ; --------------------------|
    |-/1\- ALLOWEDCORE = --<integer>-- ; ------------------------|
<medium>

---- TAPE ----|
   |          |
   |- PACK  - |
   |          |
   |- DISK  - |
   |          |
   |-<id>---  |
```

EXAMPLE:

```
GENERATE D1A  ORDER BY S5, S1, S2, S3B USING S3, S5, S7;

S1(EXTRACT KEYS TO SORTPACK, SORT USING 1000000 SEGMENTS,
  FAMILYNAME=SORTPACK, LOADFACTOR=90);

S2(EXTRACT KEYS TO SORTPACK, SORT USING 1000000 SEGMENTS ,
  FAMILYNAME=SORTPACK);

S3A(SORT USING 40000 SEGMENTS);

S5(EXTRACT KEYS TO TAPE, SORT USING 5 TAPES, FAMILYNAME=SORTPACK,
  1000000 SEGMENTS, LOADFACTOR=80);

S7(EXTRACT KEYS TO DISK, SORT USING 8 TAPES);

D1A(COPY TO TAPE, COPY BACK AT END);

S1,S2 (COPY TO DISK, COPY BACK);

S3B(COPY TO FINAL MEDIUM);

S5,S7(COPY TO TAPE, COPY BACK AT END);

DX,DY(COPY TO PACK, COPY BACK);

DW,DZ(COPY TO TAPE, COPY BACK);

SEQUENCE S5, D1A, S1 AND S7 AND S3B,S2 AND DX AND DW AND DZ AND DY;

INTERNAL FILES(FAMILYNAME=SYSPACK);

TASKLIMIT=3; ALLOWEDCORE=300000;
```

SEMANTICS:

SYSTEM/BUILDREORG uses the data base description file and the user card input to create the reorganization description file titled "DESCRIPTION/REORGANIZATION/<data base name>". This reorganization description file is used to drive the DMALGOL compiler over the standard DATABASE/REORGSYMBOLIC to create the reorganization program. The resultant reorganization program is "tailor-made" and suitable for only one reorganization run.

GENERATION IDENTIFICATION

The generation identification specifies which structures will be reorganized. The options are as follows:

1. GENERATE <d>;

   The reorganization program reads the original data set, and writes a new data set, performing complete garbage collection. If a <prime> set for <d> is specified in DASDL, then the new data set will be ordered in the key order of the <prime> set. If no <prime> set is specified, the original data seta order is maintained.

2. GENERATE <d> ORDER BY <str>;

   <str> must be <d> or an I-S, U-L, or O-L spanning set of <d>. The system performs the same function as (1) but will order on the <str>. If the <str> is <d>, then the original data set order is maintained.

3. GENERATE <istr>;

   <istr> may be any automatic indexing structure against the data set. The system will use the data set as input and completely generate <istr>, performing complete table balancing.

   With this method of generation, it is impossible for the system to determine the order in which duplicates were originally entered. Therefore, if "DUPLICATES", "DUPLICATES FIRST", or "DUPLICATES LAST" is specified in DASDL, the duplicate entries will be entered into the new structure in a random sequence. Also, if there is no duplicates condition on the key of <istr> and duplicates is encountered, the reorganization program will abort with a "DUPLICATES" exception.

4. GENERATE <istr1> USING <istr2>;

<istr1> and <istr2> must be spanning sets with the same key, duplicates condition, where condition, and "data in key" against the data set. The system will use <istr2> as input and generate <istr1>, performing complete table balancing. The most common use of this statement is "GENERATE <istr> USING <istr>", where the original and final structures are the same. This method of generation has two advantages over generation of <istr1> from the data set (Option 3):

A. The order of duplicates will be preserved.
B. If <istr2> is not an I-R structure, the extraction of keys and key sorting is not required.

This syntax also permits newly added sets and subsets to be generated from existing ones.

If <istr1> is a manual subset and the data set is GENERATEd, then all key entries pointing to deleted records in the data set will be deleted. This prevents these keys from pointing past "end of file" or to valid data.

RESTRICTIONS OF GENERATION

1. Structures may be generated only once.
2. If UPDATE with REORGANIZE is specified in DASDL and the data set is specified REORGANIZE, then it must be explicitly GENERATEd.
3. All generated structures must belong to the family of the data set and its spanning sets.
4. If <istr1> is generated USING <istr2>, then <istr2> cannot be generated USING <istr1> or using any other index structure generated using <istr1>. For example, the following GENERATE statements are invalid:

```
GENERATE S1 USING S2, S2 USING S1;
GENERATE S1 USING S2, S2 USING S3, S3 USING S1;
```

5. Manual subsets must have a "USING" clause.
6. Embedded indexing structures must have a "USING" clause, and <istr2> must be identical to <istr1>.
7. Restart data sets must not be generated.
8. Partitioned structures must not be generated.
9. Structures with partitioned masters or partitioned linking structures must not be generated.

(See also BUILDREORG note D2775 - "Reorganization Limitations".)

DEFAULTS OF GENERATION

1. If UPDATE is specified in DASDL, then every "REORGANIZEd" indexing structure in the data set family which is not explicitly generated will have the default generate:

```
GENERATE <istr1> USING <istr1>;
```

2. If the data set is generated, then all Bit-vectors and manual subsets in the data set family will generate:

```
GENERATE <istr1> USING <istr1>;
```

3. If the data set is generated, then all index sequential and ordered list structures in the data set family with DUPLICATES allowed, but FIRST or LAST unspecified, will have the default generate:

```
GENERATE <istr1> USING <istr1>;
```

FIXUP (ADDRESS UPDATE)

A fixup process is done for structures which are not to be reorganized but must be updated in the reorganization process. There are three cases:

1. If the reorganized structure is embedded, then any root word links must be updated in the master structure.
2. If the data set is generated, then all spanning sets which are not generated must have their links to the data sets updated.
3. If a data set is generated, then all structures with DASDL links pointing to it must have their links updated.

Each such structure is automatically fixed up by the reorganization program.

SORT ENVIRONMENT SPECIFICATIONS

The reorganization program uses the Sort Intrinsic. The user can control the sort resources via the following specifications:

1. "EXTRACT KEYS TO <medium>". If an indexing structure is generated from a data set or an I-R structure, then the system unconditionally creates a file consisting of the "keys" for that indexing structure. Since this "key" file is an "unordered" version of the indexing structure, it may be quite large. This statement controls where the key file is written. By default, the key file is written to the medium on which the reorganized structure will reside.

2. "SORT USING". This specifies the resources which can be used by the SORT intrinsic. Disk (or pack), tape, or a combination of disk (or pack) and tape sorting is permitted. The defaults for this specification are as follows:

   a. If this clause is not specified, the total default is "SORT USING 3 TAPES".

   b. If "SORT USING 0 TAPES" is specified, the total sort specification is "SORT USING 0 TAPES, FAMILYNAME=<id>, 20000 SEGMENTS", where <id> is the internal files family. (See INTERNAL FILES SPECIFICATION).

   c. If "SORT USING <integer> SEGMENTS" is specified, the total sort specification is "SORT USING 0 TAPES, FAMILYNAME=<id>, <integer> SEGMENTS", where <id> is the internal files family.

   d. If "SORT USING FAMILYNAME=<id>" is specified, the total sort specification is "SORT USING 0 TAPES, FAMILYNAME=<id>, 20000 SEGMENTS".

   e. If "SORT USING <integer> TAPES, FAMILYNAME=<id>" is specified, the total sort specification is "SORT USING <integer> TAPES, FAMILYNAME=<id>, 20000 SEGMENTS".

   f. If "SORT USING <integer> TAPES, <integer> SEGMENTS" is specified, the total sort specification is "SORT USING <integer> TAPES, FAMILYNAME=<id>, <integer> SEGMENTS", where <id> is the internal files family.

   g. If "SORT USING FAMILYNAME=<id>, <integer> SEGMENTS" is specified, the total sort specification is "SORT USING 0 TAPES, FAMILYNAME=<id>, <integer> SEGMENTS".

3. "LOADFACTOR=<integer>". This allows the user to override the DASDL loadfactor for reorganization only. The LOADFACTOR and "integer" have the same semantics as in DASDL. The default is the DASDL LOADFACTOR.

DATA COPY SPECIFICATIONS

The copy specifications indicate which medium to reorganize to and whether or not to copy the structure back to its final medium. Each generated structure has either a default or explicit copy specification. If no copy specifications are given, all generated structures will be reorganized to the final medium on which they reside.

The copy options are the following:

1. (COPY TO <medium>, COPY BACK) causes the system to reorganize to the specified medium and immediately copy the structure(s) back to the final medium.

2. (COPY TO <medium>, COPY BACK AT END) causes the system to reorganize to the specified medium and copy the structure(s) back to the final medium after all structures for this run have been reorganized.

3. (COPY TO <medium>) causes the system to reorganize to the specified medium. Copying the structure to the final medium requires a special run of the reorganization program (see REORGANIZATION RUN).

4. (COPY TO FINAL MEDIUM) is the default for all generated structures and causes the system to reorganize to the final medium on which the final structure will reside. No COPY BACK is necessary.

NOTE: The intermediate files generated by any copy specification are temporary and are not intended for archiving purposes.

RESTRICTIONS OF DATA COPY SPECIFICATIONS

1. Data copy specifications only apply to structures which are generated, either explicitly via the GENERATE statement or by default.

2. If the data set's "ORDER BY" structure is an index set which is GENERATEd, then it may only have one of the following specifications:

   (COPY TO FINAL MEDIUM)
   (COPY TO <medium>, COPY BACK)

DEFAULT DATA COPY SPECIFICATIONS

For all generates:

```
<str>(COPY TO FINAL MEDIUM);
```

PROCEDURE SEQUENCE SPECIFICATION

The procedure sequence specification provides the maximum control possible over the order in which processes occur so that resources and throughput can be optimized. For most applications, default sequencing together with the possible specification of TASKLIMIT and <sort environment specs>, should provide ample optimization.

The same criteria used to determine the default sequence of processes must be followed when explicitly specifying the sequence. A certain degree of flexibility is allowed in determining which independent processes should be done in parallel. All structures which are ANDed together within the same commas in the SEQUENCE clause will be processed simultaneously within the "TASKLIMIT" before proceeding to the next set of structures (delimited by commas) in the clause. The rules for sequencing are as follows:

1. If the data set is generated and its "ORDER BY" structure (<prime> set) is a generated set, then this index set must appear first and may not be ANDed.
2. If the <prime> set is specified first according to rule 1, then the generated data set must appear second. If the <prime> set is not first, then the generated data set must appear first. In either case, the generated data set may not be ANDed.
3. A "fixed" master structure of an embedded generated structure must appear after all of its embedded generated structures. It may not be ANDed with any of its embedded generated structures.
4. If <istr1> is not the <prime> set and is generated USING <istr2> and <istr2> is generated or fixed up, then <istr2> must appear before <istr1> and cannot be ANDed with <istr1>.
5. If the data set is generated and <istr1> is generated USING <istr2>, then <istr1> cannot be ANDED with any other index structure which is also generated USING <istr2>.
6. If a structure is not explicitly sequenced, then it will be placed in sequence in accordance with the above rules, and will be ANDed as much as possible.

INTERNAL FILES SPECIFICATION

The "INTERNAL FILES" specification causes the reorganization program to maintain its internal files on the given "FAMILYNAME".

There are several of these internal files depending on the number of structures and the reorganization program's internal algorithms for a particular reorganization process. These files are used to accomplish certain functions, such as program restart, retrieving records and updating addresses. All of these files exist in the directory: "<data base name>/REORGANIZATION/CONTROL".

This specification is also used in conjunction with the SORT ENVIRONMENT SPECIFICATIONS to determine the sort medium when "SEGMENTS" is specified but "FAMILYNAME" is not. (See SORT ENVIRONMENT SPECIFICATIONS.)

NOTE: This specification does not determine the default mediums for "COPY TO" files controlled by the DATA COPY SPECIFICATIONS nor the "EXTRACT KEYS TO" files controlled by the SORT ENVIRONMENT SPECIFICATIONS.

TASKLIMIT SPECIFICATION

The reorganization program will multi-process reorganization processes which have been either explicitly or implicitly ANDed together (see PROCEDURE SEQUENCE SPECIFICATION). The "TASKLIMIT" will put an upper limit on the number of these processes. The default is: "TASKLIMIT = 1;"

ALLOWED CORE SPECIFICATIONS

The "ALLOWEDCORE" clause controls the amount of sort core used by the system. The default "allowed core" in words is approximately 12000 times the number of tasks which can be sorting at the same time (controlled by SEQUENCE and TASKLIMIT).

DOLLAR CARD OPTIONS

SYSTEM/BUILDREORG has the following dollar options which are initially "SET":

1. LIST - When LIST is SET, REORGANIZATION lists the input and the resulting report.

2. ZIP - When ZIP is SET, the reorganization program compile is is automatically zippd.

NOTE: Columns 73-80 are ignored by SYSTEM/BUILDREORG.

The user may want to "RESET ZIP" and compile the reorganization program separately. The compilation deck for the reorganization program is as follows:

```
?COMPILE REORGANIZATION/"<data base name>" WITH DMALGOL LIBRARY
?COMPILER FILE TAPE=DATABASE/REORGSYMBOLIC
?COMPILER FILE DASDL=DESCRIPTION/REORGANIZATION/"<data base name>"
?COMPILER FILE PROPERTIES=DATABASE/PROPERTIES
?DATA CARD
$MERGE
?END
```

REORGANIZATION RUN

The title of the reorganization program is:
REORGANIZATION/"<data base name>"

This program may be run with three options:

1. RUN REORGANIZATION/"<data base name>"("GENERATE");

   This performs the reorganization except for copying back to the final medium for structures which specify:

   <str>(COPY TO <medium>);

2. RUN REORGANIZATION/"<data base name>"("COPY");

   This performs only the "copy back" to final medium for structures which specify:

   <str>(COPY TO <medium>);

   This is only necessary if there are structures with this copy specification. It may be used more than once.

   If REORGANIZATION is run with this option and there are no structures to copy, then REORGANIZATION will go to immediate EOJ. No structure will be copied and no errors will result.

   The reorganization control files must be present on the internal file family during the copy.

   "<data base name>/REORGANIZATION/CONTROL/="

   The intermediate files generated by REORGANIZATION must be resident. The title convention for these files is as follows:

   a. Tape: IIII...NNN, where the I's are the first alphanumeric part of the structure identifier and the N's are the three-digit structure number.

   b. Disk or pack: <data base name>/REORGANIZATION/<structure id>/ <structure number>

   NOTE: The above files are not intended for archiving purposes and should be copied back as soon after the "GENERATE" run as possible. It is also important that any file created by the reorganization program should not be purged by the user until it has been copied back. To facilitate the scratching and reusing of tapes, the reorganization program informs the user via a display when a tape can be scratched. Disk files are purged automatically.

3. RUN REORGANIZATION/"<data base name>" ("REMOVE");

   This causes REORGANIZATION program to unlock the control file and remove all residual files. This option must be used after the "GENERATE" phase and the "COPY" phase, if any, are completed.

   Intermediate structure files which are utilized by the "COPY" phase and certain control files necessary for restart are left on disk to allow multiple runs of the "COPY" phase in case of parity errors on the final structure files. These files are removed to return storage space to the MCP and to ensure that a subsequent reorganization program will not try to read any residual control files left by this run.

   NOTE: The reorganization program makes a check to ensure that the restart file it picks up was created by it and not by another version of the reorganization program. If this check fails, the program will terminate with an error when it opens the control restart file. a previous program). The error message displayed by the reorganization program in this case is: "MISMATCH BETWEEN REORGANIZATION PROGRAM AND <data base name>/REORGANIZATION/CONTROL DIRECTORY".

The following WFL example illustrates the proper sequence of execution of the reorganization program options.

EXAMPLE:

```
?JOB REORGANIZEDB;
BEGIN

    RUN REORGANIZATION/DB ("GENERATE")[GENTSK];

    IF GENTSK ISNT COMPLETEDOK THEN ABORT
    "***BAD GENERATE RUN***";

    RUN REORGANIZATION/DB ("COPY")[COPYTSK];

    IF COPYTSK ISNT COMPLETEDOK THEN ABORT
    "***BAD COPY RUN***";

    RUN REORGANIZATION/DB ("REMOVE")[REMTSK];

    IF REMTSK ISNT COMPLETEDOK THEN ABORT
    "***BAD REMOVE RUN***";

?END JOB
```

REORGANIZATION PROGRAM RESTART

In the event of a Halt/Load, DS, or various run time error, the reorganization program can restart on a process basis. If a task aborts, the reorganization program waits for all tasks ANDed within the same SEQUENCE to complete, and then terminates itself to allow the user to correct the condition. To restart from the point of the abort, run the reorganization program with the proper option (1-3 above). It will continue at the latest point possible. Successfully completed processes will not be restarted. To achieve this restart, it is necessary that "<data base name>/REORGANIZATION/CONTROL/=" be resident on the "internal file family". To restart from the very beginning of the generation, remove this directory and reload the data base. A parity error on a control file is a fatal error and requires that the data base files be re-loaded and the program be started from the beginning.

D2910 REORG - "FAMILY" ATTRIBUTE FOR REORGANIZED STRUCTURES

The reorganization program always expects a structure to be present on the FAMILY resulting from the specifications in the last DASDL run. Thus, when the <disk/pack> option for a structure is changed as a result of a DASDL UPDATE run, REORGANIZATION will expect to find the structure on the new device. The structure can be copied from the old device to the new device using Library Maintenance or the old device can be appropriately relabeled.

DMS II - REORGANIZATION

P1778 REORG - WARNINGS WHEN COMPILING VARIABLE FORMAT

When compiling REORGANIZATION against a variable format data set with no variable parts, a warning is no longer issued.

P1827 REORG - "DATAERROR 001" WHEN REORGANIZING GLOBAL DATA

REORGANIZATION no longer terminates with a DATAERROR 001 fault when new global data items are added.

P1828 REORG - LINK FIXUP IN VARIABLE FORMAT DATA SETS

Two problems associated with fixup of links within variable format data sets have been corrected. Previously, if links referencing the GENERATEd data set occurred only in the fixed format portion of the data set record, an INVALID INDEX occurred when a variable format record was processed. If links referencing the GENERATEd data set occurred in both the fixed and variable portion of the data set record, only the link in the variable portion of the record was "fixed up" for processed records having a variable portion.

P1829 REORG - "INVALID INDEX" ON RESTART OF GENERATE PHASE

If REORGANIZATION/<data base name> ("GENERATE") were Halt/Loaded or DSed during the later phases of its processing (i.e., after basic reorganization of files is complete and the AT END COPY or self-link fixup phase is in progress), an INVALID INDEX would occur on subsequent restart. This problem has been corrected.

P1902 REORG - COMPACT DATA SETS WITH RECORD RELOCATION

The following problems associated with compact data sets which contain records relocated from the parent, or original, block have been corrected:

1. If the compact data set was GENERATED and ordered using a spanning set, REORGANIZATION would abort with a SEG ARRAY error, or, in some cases, corruption of data in the compact data set would occur.

2. When GENERATing a set from the compact data set, entries in the set for relocated records would point at the resident record instead of the parent. Unpredictable results could occur later when updating the compact data set.

3. Fixups of links and spanning sets required as a result of reorganizing the compact data set would result in inappropriate NO RECORD errors generated by REORGANIZATON; or, in the case of unprotected, verified, or self-correcting links, the link would be inappropriately set to null.

P1903 REORG - GENERATION OF MANUAL SUBSETS

When reorganizing a data set having a spanning manual subset, it was possible that valid entries in the spanning subset were deleted. This could occur when the manual subset structure contained few blocks (or few blocks per master in the case of an embedded subset) and the new AA for a valid set entry was not also a valid old AA. In addition, under these circumstances, it was possible that set entries would be updated with an improper AA. These problems has been corrected.

P1904 REORG - "IOERRORTYPE 7" ON FIXUP OF STANDARD DATA SET

Previously, it was possible for an IOERRORTYPE of 7 (attempt to exceed disk row or read past EOF) to occur during the FIXUP of a standard data set. This would occur when a COPY TO specification was given for the data set in BUILDREORG. In addition, for a standard data set with variable format records, it was possible for the last words of block zero to be corrupted upon FIXUP. This would occur only when a COPY TO <medium> specification was given for the data set in BUILDREORG and would cause unpredictable results on subsequent update of the data set.

P1926 REORG - BLOCK INITIALIZATION FOR DIRECT DATA SETS

Unused records existing at the end of the last block of a reorganized DIRECT data set file were not properly marked as "INVALID" by REORGANIZATION/<data base name>, thus causing a fatal error in the ACCESSROUTINES when these records are accessed. The problem has been corrected by appropriately setting the key values in these records to "UNDEFINED".

P1927 REORG - MULTIPLE REEL INTERMEDIATE TAPE MEDIUM

A fatal IOERROR #6 would result when an intermediate tape medium was specified and multiple reels were required. This error would occur on the COPY BACK phase of program operation. On the COPY TO, or original output phase, more reels were requested by the program than were required. Reel switching is now properly handled by REORGANIZATION and these problems have been eliminated.

P2103 REORG - FAULTY OUTPUT TO TAPE

A fatal software error would occur if CHECKSUM were specified in BUILDREORG and blocks of size equal to an integral number of 30 words required output to an intermediate tape file. This has been corrected.

P2134 REORG - DUPLICATES OF ORDERED LIST

If an ordered list indexing structure having DUPLICATES specified but not DUPLICATES FIRST or DUPLICATES LAST were generated from a data set or index random structure, the resulting structure would contain duplicate entries which were not in the proper sequence. As a result, a fatal error (SYSTEMERROR 07) would occur when the ACCESSROUTINES attempted to delete a duplicate entry in the ordered list. This same problem also occurred when an index sequential structure having DUPLICATES specified but not DUPLICATES FIRST or DUPLICATES LAST was generated from an index random structure. This has been corrected.

P2135 REORG - GENERATE OF INDEX USING DIFFERENT INDEX

If an index structure were generated from a different index structure which was not itself being generated or fixed up, syntax errors would occur in the compilation of the reorganization program. This problem has been corrected.

P2136 REORG - UNUSED FOLD WORDS

Unused fold words were not properly initialized for a generated index random structure. As a result, new entries added to an index structure after its reorganization could not be properly located. This problem has been corrected.

P2209 REORG - INVALID FIXUP OF DATA SETS

Only AA words for roots or references which existed in the last physical subblock of the blocks of an embedded ordered data set were properly fixed up. Links which existed in other subblocks were not fixed up. This could lead to later problems in accessing the referenced data via these links. The problem has been corrected.

P2210 REORG - INDEX RANDOM, RANDOM SHADOW FILE AREASIZE

The AREASIZE for certain internal shadow files used in the generate of index random and random structures accommodated only 240,000 entries or records in these structure. The AREASIZE has been increased to now accommodate over eleven million entries or records.

P2211 REORG - FIXUP OF EMPTY VARIABLE FORMAT STRUCTURE

A fatal software error occurred when REORGANIZATION attempted to fixup links in a variable format standard data set which contained no records. This has been corrected.

P2506 REORG - REORGANIZATION OF COMPACT DATA SET

When a compact data set was reorganized with (ITEMS CHANGED) specified in DASDL, the REORGANIZATION program would cause any count or population items occurring in the compact data set record to be set to zero. This problem has been corrected.

P2589 REORG - RECORD FORMAT CHANGE

A SEG ARRAY error could occur or corrupted data could result when a REORGANIZE (ITEMS CHANGED) is specified for a compact data set in DASDL and the data set is generated by REORGANIZATION. This problem has been corrected.

DMS II - UTILITY

D2620 UTIL - MISCELLANEOUS CHANGES

1. The input request of the WRITE/LIST command will be printed.

2. Implements the USEMOREPAPER dollar option.

   Up to now the report from the WRITE command would skip to channel 1 prior to every structure it was writing, and many of the other reports would skip to channel 1 after printing the heading and input command. The new default is to skip 3 lines instead of skipping to channel 1, thus possibly saving a great deal of paper where many structures were involved. If the old method is preferred, UTILITY may be compiled with the USEMOREPAPER dollar option set.

3. Quotes are now optional in all cases.

   Prior to this most actions which required a structure list absolutely disallowed quotes around the structure names and those which required a file list required quotes around hyphenated file name nodes.

   The following are now all allowed:

```
RUN UTILITY/DB("WRITE 'D-S'");
RUN UTILITY/DB("WRITE D-S");
RUN UTILITY/DB("DUMP 'D-B'/D-S/'S-1' TO 'T-N'
   (SERIALNO='ABC')");
RUN UTILITY/DB("COPY D-B/'D-S'/= FROM T-N
   (SERIALNO=ABC)");
RUN UTILITY/DB("INITIALIZE 'D-S'");
RUN UTILITY/DB("INITIALIZE D-S");
```

4. Some of the reports have been changed to make them more readable and in some cases to add more information.

5. TIME(23) is used to extract the machine type for the heading.

6. Within the WRITE/LIST command the options (RECORD, HEX, and CONTROL) have been changed slightly so that they control what really is reported; i.e., RECORD now gives only the RECORD output rather than a mixture of RECORD and CONTROL. If more than one form of report is desired (record output with control information), both options must be input to UTILITY; e.g,

```
RUN UTILITY/DB("WRITE D-S(CONTROL,RECORD)");
```

D2621 UTIL - VERIFY "UTILITY" DUMP TAPES REVERSE

UTILITY will now verify its dump tapes in the reverse direction rather than rewinding and verifying in the forward direction.

D2710 UTIL - "CHECKSUM" AND "I/O" RETRY

UTILITY I/O handling has been expanded, as follows:

1. CHECKSUMs and ADDRESSCHECKs are verified when dumping data base files (structures in which CHECKSUM and/or ADDRESSCHECK are set).

2. I/O errors (both hardware and CHECKSUM/ADDRESCHECK) are retried and diagnostic information is displayed.

UTILITY now opens the data base for on-line dumps. This is done to determine end of files for CHECKSUM/ADDRESSCHECK.

Procedure RETRYIO has been implemented to handle I/O errors and UTILITY detected errors (ADDRESSCHECK and CHECKSUM). RETRYIO will display appropriate information regarding the error, retry the I/O if called for, display the results of any retries, and, if the retries fail, DS that one WORKER.

Upon entering RETRYIO, two messages are displayed. The first specifies whether the I/O was a read or a write, for what structure the I/O was intended, and the file name; e.g.,

```
DISPLAY:***READ ERROR ON STR #2, FILE:(DMSII)DB/D/DATA ON DMS.

DISPLAY:***WRITE ERROR ON STR #3, FILE:(DMSII)DB/D/S ON DMS.
```

The second message gives more detailed information about the I/O as follows:

RSLT - the I/O result descriptor followed by an explanation
   of the error - <buffer>.IORESULT

FAMILYNAME - the family where the file in question resides - <fid>.FAMILYNAME

FAMILYINDEX - the family index of the unit the I/O was attempted to - <fid>(<row>).ROWADDRESS.[29:8]

ADDRESS - the hardware address of the block

ROW - the row of the file the I/O was attempted on - <block> DIV <fid>.AREASIZE

BLOCK - the relative segment number of the first segment of the block of the file the I/O was attempted on - <buffer> .IORECORDNUM

Examples:

```
DISPLAY:***RSLT=CHECKSUM FAILED, FAMILYNAME=DMS, FAMILYINDEX=2,
   ADDRESS=108355, ROW=3, BLOCK=42.

DISPLAY:***RSLT=400002860009 : UNIT NOT READY, FAMILYNAME=DMS,
   FAMILYINDEX=1, ADDRESS=2917, ROW=1, BLOCK=1.
```

NOTE: Since ADDRESSCHECK and CHECKSUM errors are UTILITY detected after good reads, no result descriptor will be displayed.

Following these displays, the I/O will automatically be retried 1 time. If this retry fails, RETRYIO will ask the operator whether the I/O should be retried again or not. If told to retry, the I/O will be retried up to MAXRETRIES times, and, if still not successful, RETRYIO will repeat its DISPLAY and ACCEPT loop. If the operator responds that retries are not to be attempted, that one WORKER will be DSed, but UTILITY may be restarted to recover the work that WORKER didn't finish.

```
DISPLAY:***WRITE FOR STR #3 ON DMS HAS BEEN RETRIED 1 TIMES
   WITHOUT SUCCESS.

ACCEPT:R TO RETRY WRITE FOR STR #3 ON DMS, OR QUIT TO ABORT
   THIS WORKER.
```

NOTE: In response to the ACCEPT, anything beginning with the letter "R" will be construed to mean RETRY, and anything beginning with the letters "QUIT" to mean QUIT.

```
DISPLAY:***WRITE FOR STR #3 ON DMS WAS RETRIED 14 TIMES BEFORE
   SUCCEEDING.
```

NOTE: All successful writes will be re-read and compared against the original for correctness before the write is considered good.

Some examples of an entire dialog are:

1. A CHECKSUM error is detected which is corrected after 1 retry

```
DISPLAY:***READ ERROR ON STR #19, FILE:(DMSII)DB/D/E/DATA ON DMS.
DISPLAY:***RSLT=CHECKSUM FAILED, FAMILYNAME=DMS, FAMILYINDEX=3,
   ADDRESS=671124, ROW=26, BLOCK=263.
DISPLAY:***READ FOR STR #19 ON DMS WAS RETRIED 1 TIMES BEFORE
   SUCCEEDING.
```

2. A transient write error occurs which is corrected on the first retry

```
DISPLAY:***WRITE ERROR ON STR #63, FILE:(DMSII)DB/D/ISSET ON DMS.
DISPLAY:***RSLT=400010B20501 : WRITE LOCK OUT, FAMILYNAME=DMS,
   FAMILYINDEX=4, ADDRESS=495092, ROW=4, BLOCK=4015.
DISPLAY:***WRITE FOR STR #63 ON DMS WAS RETRIED 1 TIMES BEFORE
   SUCCEEDING.
```

3. A write error occurs because of a problem which can easily be corrected (reload pack firmware or move a pack to a new drive)

```
DISPLAY:***WRITE ERROR ON STR #1, FILE:(DMSII)DB/DATA ON DMS.
DISPLAY:***RSLT=400007840801 : MPX OR CONTROLLER ERROR, FAMILYNAME=
   DMS, FAMILYINDEX=1, ADDRESS=493688, ROW=0, BLOCK=0.
DISPLAY:***WRITE FOR STR #1 ON DMS HAS BEEN RETRIED 1 TIMES
   WITHOUT SUCCESS.
ACCEPT:R TO RETRY WRITE FOR STR #1 ON DMS, OR QUIT TO ABORT
   THIS WORKER.

        ...   operator diagnoses and corrects the problem   ...

OPERATOR ENTERED: AX:R.
DISPLAY:***WRITE FOR STR #1 ON DMS WAS RETRIED 2 TIMES BEFORE
   SUCCEEDING.
```

4. A write error occurs because of a problem which cannot be easily diagnosed or corrected

```
DISPLAY:***WRITE ERROR ON STR #19, FILE:(DMSII)DB/D/E/DATA ON DMS.
DISPLAY:***RSLT=800F17824101 : SECTOR FORMAT ERROR, FAMILYNAME=DMS,
   FAMILYINDEX=3, ADDRESS=688267, ROW=29, BLOCK=304.
DISPLAY:***WRITE FOR STR #19 ON DMS HAS BEEN RETRIED 1 TIMES
   WITHOUT SUCCESS.
ACCEPT:R TO RETRY WRITE FOR STR # 10 ON DMS, OR QUIT TO ABORT
   THIS WORKER.

        ...   even if the problem cannot be diagnosed and    ...
        ...   fixed immediately, it is generally a good      ...
        ...   idea to retry a couple of times rather than    ...
        ...   to DS the worker                               ...

OPERATOR ENTERED: AX:R.
DISPLAY:***WRITE FOR STR #19 ON DMS HAS BEEN RETRIED 4 TIMES
   WITHOUT SUCCESS.
ACCEPT:R TO RETRY WRITE FOR STR #19 ON DMS, OR L TO LOCK ROW #29
   AND PROCEED.
OPERATOR ENTERED: AX:QUIT.
```

NOTE: The WORKER handling this I/O will be DSed, but the rest of the active workers will continue. If the problem can be corrected, UTILITY may be restarted to finish this WORKERs work.

D2808 UTIL - ELIMINATE "II.8" RECONSTRUCT SYNTAX

The RECONSTRUCT statement has been eliminated as has been promised now for two prior system releases. RECONSTRUCT now results in a syntax error. The function of the RECONSTRUCT statement has been included in the RECOVER statement. The complete syntax for UTILITY is summarized in III.1 UTILITY note D2835.

D2835 UTIL - "UTILITY" FACILITY

The following represents an updated version of the original documentation on UTILITY which appeared in the II.8 D-Notes. All documentation changes which have occurred since II.8, including III.1 changes, have been included.

UTILITY Statements
------- ----------

```
----<cancel statement>-----------|
   |                            |
   |-<copy statement>----------|
   |                            |
   |-<dbdirectory statement>---|
   |                            |
   |-<dump statement>----------|
   |                            |
   |-<initialize statement>----|
   |                            |
   |-<list/write statement>----|
   |                            |
   |-<recover statement>-------|
   |                            |
   |-<tapedirectory statement>-|
```

CANCEL	Clears the offline dump flag in the control file following an unsuccessful offline dump.
COPY	Copies data base files from tape to disk.
DUMP	Copies data base files from disk to tape.
DBDIRECTORY	Prints the current status of rows in data base files.
INITIALIZE	Initializes data base data files.
LIST/WRITE	Prints the contents of data base files.
RECOVER	Initiates the non-automatic forms of recovery for audited data bases.
TAPEDIRECTORY	Prints information about rows written to tape by UTILITY DUMP.

Common Syntactic Items

<familyname>

```
--<identifier>--|
```

<file name>

```
---- = ---------------------------|
   |                              |
   | |<------- / ------|          |
   | |                 |          |
   |---<hyphenated id>------------|
                          |       |
                          |- /= - |
```

<hyphenated id>

```
-- a sequence of 1 to 17 letters, digits, and hyphens --|
```

The first character must be a letter; the last character must not be a hyphen.

<identifier>

```
-- a sequence of 1 to 17 letters and digits --|
```

The first character must be a letter.

<integer>

```
-- a sequence of decimal digits with no embedded blanks --|
```

<partition name>

```
-- 1 to 17 letters and digits --|
```

<range>

```
   |<-------------- , -------------|
   |                               |
---- <integer> -----------------------|
                |                 |
                |- - <integer> -  |
```

If the second integer is present, it must be larger than the first integer.

<string6>

```
-- 1 to 6 letters and digits enclosed in quotes --|
```

<structure name>

```
-- structure name as it appears in DASDL source --|
```

<tape id>

```
--<tape name>----------------------------------------------------|
              |                                                 |
              |      |<---------------- , ---------------|      |
              |      |                                   |      |
              |- ( -----/1\- VERSION = <integer> --------- ) -|
                       |                               |
                       |-/1\- CYCLE = <integer> -------|
                       |                               |
                       |-/1\- SERIALNO =  ---<integer>-|
                                           |           |
                                           |-<string6>-|
```

<tape name>

```
--<identifier>--|
```

CANCEL Statement

```
-- CANCEL --|
```

In order to perform an OFFLINE dump, the data base must be in exclusive use by UTILITY. In order to guarantee this, the data base control file is marked by UTILITY. If for any reason UTILITY is DSed before completion of the dump, the control file will have to be unmarked before any processing against the data base other than a restart of the OFFLINE dump can proceeed. The CANCEL statement may be used to unmark the control file.

COPY Statement

```
                            |<--------------- ; ---------------|
                            |                                  |
--------------------- COPY --- <copy list> FROM <copy source> ----|
  |                  |
  |-<copy options>-|
```

<copy options>

```
-- OPTIONS ( WORKERS = <integer> ) --|
```

<copy list>

```
  |<-------------------------- , --------------------------|
  |                                                         |
------<file name>----------------------------------------------------|
    |                  | |                  | |             |
    |- ( <copy list> ) -| |-<copy selector>-| |-<copy dest>-|
```

<copy selector>

```
       |<------------- AND ------------|
       |           |<- & ---|          |
       |                               |
-- ( ----- FAMILYINDEX = <range> ------- ) --|
         |                          |
         |- ROW = <range> -----------|
         |                          |
         |- PACKNAME = <familyname> -|
<copy dest>

----<as option>-----------------|
   |              |             |
   |              |-<to option>-|
   |                            |
   |-<onto option>--------------|
   |                            |
   |-<to option>----------------|


<as option>

-- AS <file name> ----------------------|
                     |                 |
                     |- ON <familyname> -|


<onto option>

-- ONTO <file name> ----------------------|
                       |                 |
                       |- ON <familyname> -|


<to option>

-- TO ( FAMILYINDEX =  ---<integer>--- ) --|
                         |          |
                         |- RETAIN --|


<copy source>

  |<---- , ---|
  |           |
----<tape id>----|
```

The COPY statement is primarily used to copy an OFFLINE dump of the entire data base from tape to disk. If used for any other reason, the COPY statement will change the version date-time stamps of the files it copies to pack so that they no longer match the control file.

It is normally unsafe to override the version date-time stamps. The most common reason for version date-time stamp mismatch is that the user forgot to specify OFFLINE on the DUMP or did not reload all of the rows of a structure.

    Example

      COPY = FROM T

    This copies all data base files from tape T to disk.

DBDIRECTORY Statement
----------- ---------

```
-- DBDIRECTORY --<dbdir list>--|
   -----


<dbdir list>

  |<------------------ , ------------------|
  |                                        |
-------<file name>-------------------------------|
      |                    | |                |
      |- ( <dbdir list> ) -| |-<rowselector>-|
<rowselector>

        |<------------------ AND ------------------|
        |                 |                        |
        |                 |<- & ---|               |
        |                                          |
-- ( ----- FAMILYINDEX = <range> ------------------ ) --|
          |                                    |
          |- ROW = <range> ----------------------|
          |- PACKNAME = <familyname> ------------|
          |                |<--------- , --------|  |
          |                |                     |  |
          |- ROWLOCK =  -----/1\- LOCKEDROW -----|
                             |-/1\- READERROR -|
```

The DBDIRECTORY statement lists the status of rows of data base files.

Example

```
 DBDIRECTORY = (ROWLOCK=READERROR,LOCKEDROW)
```

This command reports on all rows which are write locked out or have read errors. Row recovery may be used to restore damaged rows.

DUMP Statement

```
--<dump options>--------------- DUMP --------------------------------->
                   |            |
                   |- OFFLINE -|

  |<------------- ; -------------|
  |                              |
>--- <dump list> TO <dump tape> -------------------------------------|


<dump options>

-----------------------------------------|
  |                                    |
  |- OPTIONS ( WORKERS = <integer> ) -|


<dump list>

  |<------------------- , ------------------|
  |                                         |
------<file name>-------------------------------|
    |                  |  |                  |
    |- ( <dump list> ) -|  |-<rowselector>-|


<rowselector>

        |<------------- AND ------------|
        |              |                |
        |              |<- & ---|       |
        |                               |
-- ( ----- FAMILYINDEX = <range> ------- ) --|
          |                          |
          |- ROW = <range> -----------|
          |- PACKNAME = <familyname> -|
<dump tape>

--<tape name>------------------------------------------------|
               |                                          |
               |     |<-------------- , ------------|     |
               |     |                              |     |
               |- (  -----/1\- TAPES = <integer> ----- ) -|
                       |                          |
                       |-<serialno spec>----------|


<serialno spec>

                                      |<------- , -------|
                                      |                  |
-- SERIALNO ---------------------- = ----- <integer> ------|
              |                 |       |              |
              |- ( <integer> ) -|       |- <string6> -|
```

The DUMP statement is used to copy data base files from disk to tape. Two types of dumps are available: offline dumps and online dumps.

When OFFLINE is specified, the system ensures that no programs which update the data base are active during the dump. Offline dumps are useful for backing-up and recovering both audited and unaudited data bases. For unaudited data bases, all dumps are offline dumps and OFFLINE need not be specified. For audited data bases, OFFLINE must be specified if an offline dump is desired.

Online dumps may be performed while the data base is being updated. Online dumps are only permitted for audited data bases because such dumps must be used in conjunction with the audit. For audited data bases, all dumps are online dumps unless OFFLINE is specified.

Only OFFLINE dumps guarantee that the data base will not be updated during the dump. LIBRARY/MAINTENANCE dumps taken while the data base is being updated should not be reloaded or data corruption may occur.

The <dump list> designates which rows are to be dumped. The slash equal sign form may be used to dump a family of files. The equal sign alone designates all files in the data base.

If the <rowselector> is present, the rows are restricted to those designated by the <rowselector>. If a <dump list> is enclosed by parentheses and the <rowselector> is present, all files in that <dump list> are restricted by that <rowselector>. Files in a <dump list> which already have a <rowselector> have the outer selection constraints "OR"ed to the inner selection constraints.

<dump tape> identifies the tapes to be used for dumping.

When "TAPES = N" is specified, the data to be dumped is divided into N equal parts and each part is dumped to a separate family of tapes. Output dump tapes are labeled with unique CYCLE and VERSION numbers. The CYCLE number indicates the tape's family; the VERSION indicates its position within the family. For example, if "TAPES = 2" the tapes in the first family are labeled CYCLE = 1 VERSION = 1, CYCLE = 1 VERSION = 2, etc. Tapes in the second family are labeled CYCLE = 2 VERSION = 1, CYCLE = 2 VERSION = 2, etc.

WORKERS controls how many tape will be dumped in parallel. When "WORKERS = M", UTILITY dumps to M tapes concurently. If workers is not specified, TAPES controls the numbers of workers.

SERIALNO may be used to designate tape serial numbers. "SERIALNO (<integer>)" allows the serial numbers to be specified for a single cycle of tapes. The <integer> designates the tape cycle number. This form is only vaild when the number of cycles is greater than one.

Examples

(a) DUMP = TO DBONE020379

Dumps all rows of the data base to tape "DBONE020379".

(b) DUMP = (FAMILYINDEX=3) TO TAPEX

Dumps all rows on family index 3 to tape "TAPEX".

(c) DUMP = TO TAPEX(TAPES=3)

Divides the data base into three equal parts and dumps one third of it to each tape. Three tapes will be written in parallel. The first tape has CYCLE=1, VERSION=1; the second tape has CYCLE=2, VERSION=1; and the third tape has CYCLE=3, VERSION=1. If the first tape overflows, it overflows to a tape with CYCLE=1, VERSION=2, etc.

(d) DUMP = TO TAPEX(SERIALNO=231,232,233)

Dumps the entire data base to tapes with the specified serial numbers.

```
(e)    DUMP = TO TAPEX(TAPES=2, SERIALNO(1)=240,250,
                                SERIALNO(2)=251,252)
```

Divides the data base into two equal parts and dumps one half of it to each tape. Two tapes will be written in parallel. The first tape has CYCLE=1, VERSION=1 and SERIALNO=240. The second tape has CYCLE=2, VERSION=1 and SERIALNO=251. If the first tape overflows, it is labeled CYCLE=1, VERSION=2 and SERIALNO=250. If the second tape overflows, it is labeled CYCLE=2, VERSION=2 and SERIALNO=252.

```
(f)    DUMP DB/A TO TAPE1; DB/B TO TAPE2; DB/C TO TAPE3
```

This example explicitly partitions the files rather than letting UTILITY do it implicitly.

```
(g)    OPTIONS(WORKERS=3) DUMP
       = (FAMILYINDEX=1) TO T1(TAPES=3);
       = (FAMILYINDEX=2) TO T2(TAPES=3);
       = (FAMILYINDEX=3) TO T3(TAPES=3)
```

UTILITY will dump each family index to a separate set of tapes. The rows on each pack will be divided up by UTILITY into three parts, which will be dumped to different cycles of the same tape name. Nine tapes could be dumped simultaneously, but WORKERS=3 limits UTILITY to three tapes at a time.

```
(h)    OPTIONS(WORKERS=3) DUMP
                  = (FAMILYINDEX=1) TO T1(TAPES=3,
                     SERIALNO(1)=100,101,
                     SERIALNO(2)=200,201,
                     SERIALNO(3)=300,301);
                  = (FAMILYINDEX=2) TO T2(TAPES=3,
                     SERIALNO(1)=110,111,
                     SERIALNO(2)=210,211,
                     SERIALNO(3)=310,311);
                  = (FAMILYINDEX=3) TO T3(TAPES=3,
                     SERIALNO(1)=120,121,
                     SERIALNO(2)=220,221,
                     SERIALNO(3)=320,321)
```

This example is identical to the previous one except that the tapes have been assigned serial numbers. SERIALNO(1) means the first cycle of that tape name. The first reel of cycle 1 of T1 will be assigned serial number 100. If it overflows, the second reel (version 2) of cycle 1 will be assigned number 101, etc.

```
(i)    DUMP = (FAMILYINDEX=1,4 AND PACKNAME=DBDATA) TO TAPEX
```

All rows that reside on family indexes 1 or 4 of family DBDATA will be dumped to TAPEX.

```
(j)    DUMP (DB/A/=, DB/B/=, DB/RDS/= (PACKNAME=DBDATA))
            (PACKNAME=DBDATA&FAMILYINDEX=1) TO TAPEX
```

All rows in files DB/A/= and DB/B/= which are on FAMILYINDEX 1 of pack DBDATA will be dumped. The rows of DB/RDS/= will be dumped if they satisfy either the inner condition PACKNAME=DBDATA or the outer condition that PACKNAME is DBDATA and FAMILYINDEX=1. Since the first condition is less restrictive, all the rows of DB/RDS/= on DBDATA will meet it and be dumped.

INITIALIZE Statement

```
-- INITIALIZE --- = ---------------------|
                  |                      |
                  | |<------- , -------| |
                  | |                  | |
                  |---<structure name>---|
```

INITIALIZE creates empty data base data files. Following the initialize, the structure's file-state field in the control file is assigned the value CFAUDINZ for audited data bases or CFFILENORMAL for unaudited data bases.

UTILITY ensures that there are no dangling pointers (sets, root words, or links) which refer to initialized structures.

When a disjoint data set is initialized, all sets which refer to it and all embedded structures within it are automatically initialized. If a disjoint structure for which a population item exists in global data is initialized, all such structures must be initialized along with the global data record. If a structure containing a counted link is initialized, the structure referenced by the link must also be initialized.

Disjoint automatic sets can only be initialized when the data set they reference is also initialized. Disjoint manual subsets may be initialized without initializing the data sets they reference.

While UTILITY is initializing structures, it locks the control file to ensure that it has exclusive use of the data base. Initialization can be executed in several steps. Different structures may be initialized in each step. Initialization may be repeated if it is interrupted by a halt/load or other failure.

LIST/WRITE Statement

```
---- LIST ------------------------------------------------|
   |                                                    |
   |- WRITE --------------------------------------------|
   |                                                    |
   |  |<---------------------- ; ---------------------| |
   |  |                                               | |
   |---<structure spec>---------------------------------|
                        |          | |               |
                        |-<format>-| |-<block range>-|
```

```
<structure spec>

----<structure spec>---------------------------|
   |                  |                         |
   |                  |- / <partition name> -   |
   |                                            |
   |- <database name>/CONTROL ------------------|
   |                                            |
   |- ALL --------------------------------------|
   |                                            |
   |- = ----------------------------------------|
```

```
<format>

         |<------ , -----|
         |               |
-- ( ----- HEX --------- ) --|
           |            |
           |- RECORD ---|
           |            |
           |- CONTROL --|
```

```
<block range>

-- <hex block address> -------------------------------|
                        |                            |
                        |- - <hex block address> -   |
```

LIST and WRITE display the contents of data base files. LIST directs output to the terminal; WRITE sends output to the printer. CONTROL can be used to list control information only.

RECOVER Statement

```
------------------------ RECOVER ( ---<recover rows>------- ) -------->
  |                  |              |                  |
  |-<recover options>-|              |-<recover whole db>-|

  |<------------------- ; ------------------|
  |                                         |
>-----------------------------------------------------------------------|
    |                   | |              |
    |-<recover filelist>-| |-<sourcelist>-|
<recover options>

              |<------------------ , -----------------|
              |                                       |
-- OPTIONS ( -----/1\- NOZIP --------------------------- ) --|
                |                                   |
                |-/1\- FLUSHDB = <integer> ---------|
                |                           |       |
                |                           |- MIN -|
                |                                   |
                |-/1\- WORKERS = <integer> ---------|
```

```
<recover rows>

-- ROWS ------------------ USING BACKUP ------------------|
         |            | |                              |
         |- IN PLACE -| |- USING AUDIT ONLY , <limit> -|
```

```
<limit>

-- LIMIT =  --- <integer> --- CONTROL --- POINTS ----|
             |             |           |          |
             |             |- SYNC ----|          |
             |                                    |
             |- THRU AUDIT <integer> -------------|
             |                                    |
             |-<date-time>------------------------|
             |                                    |
             |- * -  --- <integer> : <min> -------|
                      |                           |
                      |- <integer> AUDIT FILES ---|
```

```
<recover whole db>

---- ROLLBACK ----- THRU AUDIT <integer> ------|
  |            | |                          |
  |- REBUILD --| |- TO ---<boj/eoj point>---|
                        |                   |
                        |-<date-time point>-|
```

```
<boj/eoj point>

---- BOJ --- OF <job no> / <task no> -----------------------|
  |       |                           |                  |
  |- EOJ -|                           |- ON <date-time> -|
```

```
<date-time point>

---- GEQ ---<date-time>--|
  |       |
  |- LEQ -|
```

```
<date-time>

-- <month> <day> ---------------- AT <hrs> : <min> ------------------->
                  |           |
                  |- , <year> -|

>---------------------------------------------------------------------|
  |                        |
  |- : <sec> --------------|
              |            |
              |- . <fsec> -|
```

<job no>
Mix number for job. Unsigned integer of not more than 4 digits.

<task no>
Mix number for task. Unsigned integer of not more than 4 digits.

<month>

JAN, JANUARY, FEB, FEBRUARY, etc.

<day>
Day of month. Unsigned integer of not more than 2 digits.

<year>
Year. Unsigned integer of not more than 4 digits.

<hrs>
Hours. Unsigned integer of not more than 2 digits.

<min>
Minutes. Unsigned integer of not more than 2 digits.

<sec>
Seconds. Unsigned integer of not more than 2 digits.

<fsec>
Fractional seconds. Unsigned integer of not more than 3 digits.

<recover filelist>

```
  |<---------------------------------- , ----------------------------|
  |                                                                  |
------<file name>-----------------------------------------------------|
     |                        | |                | |                |
     |- (<recover filelist>) -| |-<rowselector>-| |-<recover dest>-|
```

<rowselector>

```
       |<----------------- AND -----------------|
       |               |<- & ---|                |
       |                                         |
-- ( ----- FAMILYINDEX = <range> ----------------- ) --|
        |- PACKNAME = <familyname> -----------|
        |- ROW = <range> ---------------------|
        |                 |<--------- , -------|  |
        |                 |                    |  |
        |- ROWLOCK = -----/1\- LOCKEDROW -----|
                         |-/1\- READERROR -|
```

<recover dest>

```
----<as option>-------------------|
  |             |                |
  |             |-<to option>-|
  |                            |
  |-<onto option>--------------|
  |                            |
  |-<to option>----------------|
```

<as option>

```
-- AS <file name> --------------------------|
                    |                    |
                    |- ON <familyname> -|
```

<onto option>

```
-- ONTO <file name> -------------------------|
                      |                    |
                      |- ON <familyname> -|
```

<to option>

```
-- TO ( FAMILYINDEX =  ---<integer>--- ) --|
                        |            |
                        |- RETAIN --|
```

<source list>

```
          |<--- , ---|
          |          |
-- FROM ---<tapeid>----|
```

Restrictions:

Some forms of the RECOVER statement require both a <recover filelist> and a <sourcelist>; others do not. The following table summarises these requirements.

Recovery Statement	Filelist	Sourcelist
RECOVER(ROWS USING BACKUP)	Required *	Required
RECOVER(ROWS IN PLACE USING BACKUP)	Required **	Required
RECOVER(ROWS---USING AUDIT ONLY---)	Not Permitted	Not Permitted
RECOVER(ROLLBACK----)	Not Permitted	Not Permitted
RECOVER(REBUILD----)	Not Required ***	Required

* If "ONTO <file name>" or "AS <file name>" is used, the <file name> must not be the same as any data base file name in this or any other data base.

** <recover dest> is not permitted.

*** "ONTO <file name>" and "AS <file name>" are not permitted.

The RECOVER statement is used to initiate all manual forms of recovery for audited data bases (i.e., all forms except halt/load and abort recovery, which are initiated automatically). Manual forms of recovery fall into two classes: those that recover damaged rows, and those that recover the entire data base.

"RECOVER(ROWS ..." is used to recover rows of data base files. The data base may be in use during row recovery. UTILITY starts row recovery; Reconstruct completes the process. Reconstruct processes an external coroutine called Datarecovery which reads the audit trail and applies the audit images. Reconstruct is zipped by UTILITY unless NOZIP is specified. NOZIP permits the entire row recovery to be controlled by a single WFL job.

"RECOVER(REBUILD..." moves the entire data base forward in time. The data base must not be in use during rebuild. UTILITY begins the rebuild by copying the entire data base from one or more sets of dump tapes. Then the application of audit trail images is performed by RECOVERY/<data base name>. RECOVERY is ZIPped automatically by UTILITY unless NOZIP is specified. If rebuild is not successful, the last resort is to reload a backup copy of the data base and rerun all updates.

"RECOVER(ROLLBACK..." moves the entire data base backwards in time. Rollback must have exclusive use of the data base. Rollback does not copy in a backup dump of the data base files--the current files are used. During rollback, RECOVERY reads the audit trail and applies the audit images. RECOVERY is automatically initiated unless the NOZIP option is used.

Unlike rebuild, rollback cannot be used to recover from data corruption. It can potentially save time over rebuild, however, to correct logical errors such as running an update program with incorrect input.

The FLUSHDB option controls how often a restart point is taken by Datarecovery or RECOVERY. The default is 20 minutes. Thus, by default at most 20 minutes of processing can be lost due to a system crash while Datarecovery or RECOVERY is running.

The WORKERS option controls how many dump tapes are processed in parallel for those forms of recovery which require the use of such tapes.

Example

```
OPTIONS(NOZIP) RECOVER(ROWS USING BACKUP)
          = (ROWLOCK = LOCKEDROW, READERROR)
          FROM T1,T2,T3,T4,T5,T6,T7,T8
```

All rows which have write errors (ROWLOCK=LOCKEDROW) or read errors (ROWLOCK=READERROR) will be recovered using the data in the dumps plus the changes recorded in the audit trail since the dumps. Since the WORKERS=<integer> construct is not used, UTILITY will process each dump serially, and process the cycles of each dump in parallel. The number of cycles is specified when the dump is taken.

Example

```
 OPTIONS(NOZIP,WORKERS=3) RECOVER(ROWS USING BACKUP)
        = (ROWLOCK = LOCKEDROW, READERROR)
        FROM T1,T2,T3,T4,T5,T6,T7,T8
```

This example is the same as the previous one except for the WORKERS option which causes three reels to be processed at a time. If, for example, a worker finishes its part of dump T1, it will go on to T2, even if the other two workers have not finished their parts of T1.

Example

```
 RECOVER(ROWS USING BACKUP)
        = (ROWLOCK=READERROR, LOCKEDROW AND
           PACKNAME=DBDATA AND
           FAMILYINDEX=1-3,5)
        FROM TAPEX
```

If there are 5 packs in the DBDATA family numbered 1-5, none of the rows on familyindex 4 will be selected for row recovery. All of the rows of familyindex 1,2,3 or 5 that either have a READERROR or are LOCKED OUT will be row-recovered.

Example

```
 RECOVER(ROWS USING AUDIT ONLY, LIMIT = * - 1:30)
```

This form of row recovery is often referred to as "quickfix". The three previous examples required the use of backup dump tapes. In those examples, UTILITY began by copying old versions of the rows to be repaired from backup dumps to temporary files. Datarecovery then applied the audit images to the rows in the temporary files. Finally the recovered rows were exchanged from the temporary files to the data base files. The advantage of that form of row recovery is that, barring irrecoverable I/O errors, it always works. The disadvantage is the time consumed locating the proper dump tapes, loading the rows, and processing all the audit images from the time of the dump to the end of the audit trail.

Quickfix reduces the time required for row recovery. It may not, however, always be able to repair all locked out rows. Quickfix starts by scanning the audit in reverse starting from the end of the audit. The user specifies a limit to this reverse scan using the LIMIT syntax. The example limits the reverse scan to audit records created at most one and one half hours before the time in the last audit record. At the end of the reverse scan, DATARECOVERY determines if any rows can be repaired. If so, it reverses directions in the audit trail and performs a normal row recovery. The backward scan will stop short of the <limit> if the recoverability status of all locked rows is determined prior to reaching the <limit>.

For all forms of row recovery, if "IN PLACE" is used, the audit images are applied to the data base files directly. This requires that they be locked out by UTILITY if they are not already locked out. If "IN PLACE" is not used, the audit images are applied to temporary files, and the appropriate rows are EXCHANGEd into the data base files upon completion of image application.

If "USING BACKUP" is specified and "IN PLACE" is not specified, any recovered rows which initially have a ROWLOCK value of 0 (normal) will be changed to a value of 2 (READERROR) by UTILITY. This is done so that in the event of a halt/load during the EXCHANGE phase, Datarecovery can tell which rows have been exchanged and which have not. The ROWLOCK value of READERROR will not inhibit normal processing, dumping, or recovering of those rows in any way.

Quickfix IN PLACE is not as powerful as normal quickfix because it cannot recover rows damaged by irrecoverable errors in the magnetic recording media.

Example

```
  RECOVER(REBUILD THRU AUDIT 4567) = FROM T1
```

The entire data base is loaded from dump T1 and brought forward to the end of audit 4567.

Example

```
  RECOVER(ROLLBACK TO BOJ OF 1234/1235)
```

This rolls the current data base files back to the beginning of task 1234/1235.

Both rollback and rebuild perform a halt/load recovery before they terminate. This produces a discontinuity in the audit and it may cause some of the original audit to be discarded. Thus, care must be taken not to confuse the audit which has been discarded with the new audit which will be created.

Consider a data base for which the last audit is 6000. If rollback terminates in audit file 5888 then any old copies of audit files 5888 through 6000 are invalid. Auditing will begin at the end of audit file 5888 when normal updating resumes.

TAPEDIRECTORY Statement

```
-- TAPEDIRECTORY -- <tape id> --|
   -------
```

The TAPEDIRECTORY statement lists the tape directory written by UTILITY at the beginning of each dump tape. Directories of successive reels are cumulative; thus, a directory of the last reel provides information about all rows dumped by UTILITY. This information is useful during recovery. From it, UTILITY can be told exactly which tapes contain the rows of interest; this eliminates extra search time.

Example

```
TAPEDIRECTORY DBNAME010279(CYCLE=2, VERSION=3)
```

Operator Interface to UTILITY

The number of workers can be changed while UTILITY is running by entering an accept message. The syntax for the message is:

```
<mixno> AX WORKERS = <integer>
```

This can be entered even if the WORKERS construct was not used in the original input. The mix number must be that of the main UTILITY stack.

Restarting UTILITY

UTILITY may be restarted if it is DSed while processing a DUMP or RECOVER request, or reading or writing tape. It may be restarted whether it was DSed internally due to a fatal I/O errors or was DSed by the operator. UTILITY restarts with the tape being processed when it was DSed; completed tapes are not reprocessed. Since UTILITY does not rescan its input during a restart, the input string may be empty. The job number of the UTILITY run to be restarted is passed as a negative value via the taskvalue attribute.

Example

```
RUN UTILITY/TEST (" "); VALUE = -1025
```

UTILITY restarts using the HLDUMPINFO files created by the original run of UTILITY. The HLDUMPINFO file is titled <data base name>/HLDUMPINFO/<job number>. In addition each tape or dump worker that was active has its own HLDUMPINFO file titled <data base name>/HLDUMPINFO/<job number>/<worker number>. All of these files must be present or UTILITY will not restart.

To avoid conflicting restart file titles, only one copy of UTILITY should be run in a single WFL job.

Continuing UTILITY

UTILITY runs which "RECOVER(ROWS..." and "RECOVER(REBUILD..." may be continued when needed files are not loaded from tape due to I/O errors or errors in the UTILITY input. A special option allows the missing files and rows to be added before RECOVERY is initiated.

A taskvalue of 8 identifies a continued UTILITY run. The RECOVER request must not be changed for the continued run. Only the filelist or tapelist may be changed to add the necessary files. UTILITY will use the RECONSTRUCTINFO or REBUILDINFO file created by the previous UTILITY run and simply add to it. This file must be present.

Example

Assume the data base was dumped to four tapes (Tl thru T4). Then a rebuild was attempted as follows:

```
RUN UTILITY/TEST ("RECOVER(REBUILD THRU AUDIT 5) FROM T1,T2,T4 ")
```

Since a REBUILD must load the entire data base, UTILITY would notice that the files from T3 were missing and would prevent the rebuild. Using the continue request, the missing tape could be loaded without loading the other tapes. Following this RECOVERY would be initiated normally.

```
RUN UTILITY/TEST ("RECOVER(REBUILD THRU AUDIT 5) FROM T3 ");
VALUE=8
```

I/O Errors During Dump

UTILITY handles both hard I/O errors and timeouts while writing to tape. An appropriate error message is displayed and the tape on which the error occurred is labeled "BADTAPE" and closed. The following message is then displayed:

```
AX "OK" FOR RETRY OR NEWTAPE
```

The operator may enter "AX:OK" or DS the job. "OK" causes UTILITY to re-dump the tape which failed. When a failure occurs, the operator can switch tapes, clean the tape in error, change tape drives, etc. Other UTILITY workers can proceed with their dumping while this worker is waiting.

I/O errors that occur while UTILITY is reading from disk are fatal. UTILITY cannot successfully complete a dump unless all the rows that are specified in the dumplist are dumped. The row in error must be recovered before it can be dumped. UTILITY will DS itself in this case, leaving its restart files present so that it can restart the dump where it left off.

I/O Errors Processing RECOVER

If UTILITY encounters hard I/O errors or timeouts while reading tape, an error message is printed and the following message is displayed:

```
AX 'RETRY' OR 'SKIP ROW' OR 'QUIT TAPE'
```

If "RETRY" is entered, UTILITY starts over at the beginning of the tape, and attempt to read it again. "SKIP ROW" causes the row on the tape that got the I/O error to be passed over and that row is omitted from the recovery operation. If UTILITY skips a row in this manner, RECOVERY is not automatically ZIPped. This allows the row to be reloaded via a UTILITY continue request before RECOVERY is initiated.

"QUIT TAPE" causes the rest of the rows on this and any subsequent reels to be skipped. If there are other input tapes, UTILITY will load these tapes. As with "SKIP", RECOVERY is not ZIPped. This allows the missing rows to be loaded from other tapes.

DMS II - UTILITY

P1600 UTIL - HALT/LOAD INFORMATION

1. If a Halt/Load occurs when a worker, other than the first, is putting its information into the RECON file, the RECON file is removed and information from all previous workers is lost.
2. If a Halt/Load occurs after a worker is done with the RECON file but before a HLDUMPINFO file is created for the worker, after the Halt/Load the worker will again try to put its information into the RECON file; however, since all its information is already in the RECONSTRUCT file, it will think that it has nothing to do.

Both these problems have been corrected.

P1757 UTIL - PRINT PATCH LEVEL IN HEADING

UTILITY now prints its patch level in the heading on listings rather than its internal version level.

P1779 UTIL - "INVALID INDEX" PRINTING ORDERED DATA SET

UTILITY terminated with an INVALID INDEX while printing block zero of an ORDERED data set containing subblocks. This failure only occurred when the data set record size was one word. This has been corrected.

P1780 UTIL - ALLOW RECONSTRUCTION OF NEW ROWS SINCE DUMP

If all rows to be reconstructed in a run of UTILITY were new since the dump, UTILITY would fail to create the temporary reconstruct files and datarecovery would end up waiting on a "NO FILE". This has been corrected.

P1809 UTIL - INCORRECT TIME PRINTED

When printing the stop time for a rebuild or rollback, RECOVERY and UTILITY would print the seconds incorrectly. RECOVERY printed the wrong seconds; UTILITY printed the wrong fractional seconds. The actual time value used was correct, however. RECOVERY now prints the correct time truncated to the nearest second; UTILITY now prints the correct time truncated to the nearest millisecond.

P1810 UTIL - REMOVE "ROWLOCKOUTAUDIT" AT START OF "REBUILD"

UTILITY now removes the ROWLOCKOUTAUDIT file at the start of REBUILD. This file is normally removed at the end of a successful recovery run, if it is ever present at all. If recovery fails, however, it may be left on disk. Its presence could cause a rebuild to erroneously lock out some rows upon completion because of the prior unsuccessful recovery run. Removing it at the start of a rebuild prevents this problem.

P1868 UTIL - LOCATE "H/L" FILE FOR UNAUDITED DATA BASE

UTILITY placed HLDUMPINFO files for an unaudited data base on a family pack. When looking for HLDUMPINFO files for an unaudited data base, the code did not set up the pack name to the family of the user and so looked for files on disk. As a result, UTILITY could not find HLDUMPINFO files for the unaudited data base. This has been corrected.

P1869 UTIL - "DBDIR" WITH "ROWLOCK" OPTION

If any row in a structure was DMROWLOCKed or if any row(s) in a structure was specified to be printed, structure information was printed.

However, row information for a structure was printed only if specified by the user, whether or not it was DMROWLOCKed. This problem has been corrected.

P1871 UTIL - MISSING ENTRY IN "HL" FILE

UTILITY will issue an error when it cannot find an entry in the HLDUMPINFO file only for a dump. It is possible that a UTILITY recover function would search an entire reel of tape and do a reelswitch without having found anything to load from the previous reel, and no entry for it would have been made in the file. This problem has been corrected.

P1884 UTIL - "TDATABEGIN" FIELD NOT INITIALIZED

If an IOERROR occurs while a data base dump is being checked, bad data may be written in bits 47:20 of word zero of each tape block. This may cause UTILITY to fail later with an INVALID INDEX. This problem has been corrected.

P1928 UTIL - TAPE BLOCK COUNT NOT PROPERLY STORED

When a reel switch occurred under the following conditions, the information in the Halt/Load file was not updated accurately, thus causing an INVALID INDEX after a Halt/Load occurs:

1. The data on the new reel starts at the beginning of a row.

2. Using any version of the directory for the tapes that shows where the row actually lives.

This has been corrected.

P1929 UTIL - CORRECT FAMILY NOT SET

If the FAMILY statement is being used, then after a Halt/Load the correct family is not set up for the tapeworker hldumpinfo files. This causes UTILITY to look in the wrong place for the Halt/Load file; therefore, it does not use the information from those files. This has been corrected.

P1930 UTIL - REEL VERSION LOCATION OF ROW

During a dump, a reelswitch that causes the next version to start at the beginning of a row does not update the tape directory properly. The directory value (DKOWREEL) for the row that starts at the beginning of the next version is not updated to show that it is being placed on the next version. This has been corrected.

P1931 UTIL - CYCLE "9" AND ABOVE MAY NOT RESTART

Information as to which cycles and versions of tapes are required after a Halt/Load is not complete starting with Cycle 9. Cycle 9 may restart depending on version. Cycle 10 and above will not restart. This has been corrected.

P1932 UTIL - "IOERROR" ON TAPE

If an I/O error occurred before UTILITY had a chance to save the variables needed for restarting, software errors could occur. This has been corrected.

P1948 UTIL - "CANCEL" OPTION CAUSES ATTRIBUTE ERROR

The procedure to perform the cancel function was not setting up the title for the control file before trying to open it. This has been corrected.

P1949 UTIL - HALT/LOAD FILE TITLE

After a Halt/Load, the file title was not properly set up, causing an attribute error. This has been corrected.

P1950 UTIL - "LIST" FORMAT ERROR

For LIST, if five words of data were printed on one line, the number of characters exceeded the number allowed (80). This has been corrected.

P1951 UTIL - INVALID STRUCTURE NAME

In a LIST/WRITE function, although <data base name>/CONTROL was a valid option, it was getting an INVALID STRUCTURE NAME error. This has been corrected.

P2015 UTIL - "DUMPTIME" TAKEN AT WRONG TIME

The DUMPTIME telling RECOVERY where to start in the audit was taken after the data files were opened and the number of rows determined. This created a timing window that has been closed.

P2016 UTIL - "CONTINUE" REQUEST DOES NOT RECREATE FILES

When doing a CONTINUE request, UTILITY was recreating the file to be loaded to in some cases, thus losing data from the previous load. This has been corrected.

P2017 UTIL - RESETS "DATABEGIN" FOLLOWING RESTART

TDATABEGIN was not being saved in the Halt/Load file and was not reset to its original value following a UTILITY restart. This has been corrected.

P2018 UTIL - ERASE RECPNINFO FOR ROWS SKIPPED

Rather than altering the reconstructinfo files as soon as a row is skipped on a tape, it is done at the end of a reel. Thus, if a restart happens and the rows are loaded successfully on the second try, the information about the row is still left in the reconinfo file.

P2019 UTIL - CORRECTS "TIMESTAMP" COMPARES

UTILITY was incorrectly comparing TIMESTAMPS as reals, which would result in an incorrect comparison being made when there were bits turned on in the exponent field and the sign of the exponent was negative. This situation will occur on NOVEMBER 15, 1978. This has been corrected.

P2104 UTIL - MULTIPROCESSOR PROBLEMS

The following problems have been corrected:

1. Locking problems caused UTILITY to try to initiate an already-active task when firing up its read/write volumes.

2. When processing more than one tapeworker, UTILITY may get an INVALID INDEX after a Halt/Load when trying to restore its worker information.

P2137 UTIL - CONTINUATION REQUEST FOR "COPY"

A continuation request for a COPY did not work correctly and resulted in a NO FILE on RECONSTRUCTINFO. Now COPY does not expect to find a pre-existing RECONSTRUCTINFO file when it is continued.

P2138 UTIL - "FILELISTREQUIRED" ON RESTART

If a Rebuild had to restart and there was a filelist provided in the input, Rebuild could scan the wrong thing. This problem has been corrected.

P2139 UTIL - TIMING PROBLEM IN SCANNER

Scanning the input could become confusing if both UTILITY and a tapeworker were scanning the input at the same time. This has been corrected.

P2216 UTIL - "TAPEDIRECTORY" COMMAND PROBLEMS

The following problems have been corrected:

1. A tape name containing "=" was not giving a syntax error.

2. The TAPEDIRECTORY command was not printing the report.

P2390 UTIL - "CFNAME" ARRAY SIZE INCREASED

The size of the CFNAME array has been increased to contain the entire title including the packname.

P2398 UTIL - INCORRECT SELECTION OF STRUCTURES

If a data base resides on families PACKX and PACKXY, a UTILITY request with a file selection based on PACKNAME=PACKX would cause the request to be performed with families PACKX and PACKXY. This problem has been corrected.

P2415 UTIL - REPORT ON NEW QUICKFIX LIMITS

UTILITY's report showing the recovery syntax now correctly displays the new Quickfix limits.

P2416 UTIL - PRINT NEW CONTROL FILE INFORMATION

UTILITY now prints the control file information which was added on the III.1 release.

P2417 UTIL - "FLUSHDB" OPTION CORRECTION

When FLUSHDB is given in the OPTIONS list, UTILITY would incorrectly pass the default value of 20 minutes to recovery.

Beginning with system release 3.2, UTILITY will no longer allow FLUSHDB in the <recover specification>. It will only be permitted in the <options list>.

P2460 UTIL - "END OF FILE" ERRORS ON DUMPS

UTILITY no longer incurs end-of-file errors. Formerly, this could happen for on-line dumps when RECOVERY was required.

P2461 UTIL - "COPY AS" ERROR

When a 'COPY AS' is done that includes the control file, UTILITY gets a software error in the control file module. This problem has been corrected.

P2484 UTIL - "INITIALIZE" OF GLOBAL DATA

INITIALIZE of global data was erroneously stating that all structures in the data base must be initialized along with the global data. This has been corrected.

P2642 UTIL - FAILURE TO DUMP PARTITIONS

UTILITY failed to dump partitioned structures for data bases not under a usercode. Partitions stored under a usercode were dumped correctly. Attempts to explicitly dump partitioned structures resulted in the message "NO DATABASE FILES MATCH THIS IDENTIFIER". This problem has been corrected.

DMS II - TFL

P2507 TFL - "TFL" UPDATE

The TFL update level will only change when formats and subformats are affected by the update. Changes to subbases, journals and globals do not increment the update level.

P2527 TFL - SCANNER PROBLEMS

Problems in scanning both usercodes and global options have been corrected.

P2550 TFL - DEFAULT TEXT

TFL was improperly copying the usercode and packname text when specified for the default journal. Any subsequent journals specified in TFL should pick up the text from the default journal unless a usercode or packname was specified.

DMS II - TRUTILITY

P2462 TRUTILITY - MISCELLANEOUS CORRECTIONS

Miscellaneous problems in the original implementation have been corrected.

DMS II - HOSTLIB

P2418 HOSTLIB - MISCELLANEOUS CORRECTIONS

Miscellaneous problems have been corrected.

P2548 HOSTLIB - READING, WRITING OVER ROWS

Various problems in reading and writing blocks near the end of a row have been corrected.

DMS II - HOSTINTERFACE

P2547 HOSTINTFACE - "INCLUDE" "PROPERTIES"

The HOSTINTERFACE will now include defines from TRPROPERTIES.

DMS II - TRPROPERTIES

P2551 TRPROPERTY - FIELDS IN "PORT" MESSAGE

The layout of the PORT message used by the Remote Library and Hostinterface is now defined in the TRPROPERTIES.

DMS II - DDDASDL
--- -- - -------

P2613 DDDASDL - MAINTAIN LEVEL NUMBER AND OCCURRENCES

The COBOL level number and number of occurrences are now maintained in the Data Dictionary for each item.

DMS II - DDUPDATE

P2614 DDUPDATE - TEXT FILES WITHOUT VALID HEADER

DDUPDATE will now produce an appropriate error message if a text file which does not contain a valid header is INSERTed. Previously, DDUPATE died with an INV OP at 30412000.

DUMPALL

P1766 DUMPALL - "UL" OPTION CORRECTION

DUMPALL now asks for an unlabeled tape when the input file title is "UL".

P1767 DUMPALL - "ROUTINE" AND "COPY" STATEMENTS REWRITTEN

DUMPALL's "ROUTINE" procedure (described in the output from the TEACH verb) has been rewritten, thus correcting many problems relating to the "PACK=<packname>" and "SKIPTM<integer>" phrases for the ROUTINE statement. The syntax and semantics of the ROUTINE statement and the COPY statement remain unchanged.

P1768 DUMPALL - "DMPMT" SHORT RECORDS

The DMPMT verb now dumps the entire record when the UL input file type is specified.

P1769 DUMPALL - EQUAL SIGN IN FILE TITLE

DUMPALL will now accept file title with embedded equal signs.

P2261 DUMPALL - "INTMODE" OF "HEX" OR "SINGLE"

DUMPALL will now handle files with INTMODEs of HEX or SINGLE and UNITS of CHARACTERS when using the LIST, LISTAN or LAN options.

P2343 DUMPALL - UNALLOCATED ROWS

When DUMPALL is listing fixed record length disk files, the unallocated rows will not be listed or allocated. The following message is written to the printer for all unallocated rows:

"***ROW NUMBER <nn> NOT ALLOCATED***"

P2442 DUMPALL - LOOP IN "DMPMT" IF FILE FOUND ON DISK

DUMPALL will now no longer loop dumping the same file over and over when the DMPMT option is used but a file on disk with the same title as the requested tape is found. The input file in the DMPMT routinc now has KIND=TAPE.

P2443 DUMPALL - NO FILE WHEN EQUATING TAPE TO "LIST" OPTION

DUMPALL will now handle LIST requests where the input file has been equated to KIND=TAPE.

P2528 DUMPALL - "NEWFILE" VS. OLD FILE

DUMPALL now sets the NEWFILE attribute for all output disk and pack files to cause the creation of a new file and not to update an old file with the same title.

P2529 DUMPALL - ERROR MESSAGES

DUMPALL will now output error messages that are more consistent with the error.

Example:

Input: "L MYFILE DLB"

"DLB" was input instead of "DBL"

New Error Message: "EXTRANEOUS CHARACTERS IN INPUT STRING"

Old Error Message: "OUTPUT PARAMETERS NOT ALLOWED FOR COPY"

DUMP ANALYZER

D2451 DUMPANALY - NEW "IOCB" WORD, REMOTE BACKUP FILES

The new IOCB word RDEXTENSIONWORD has been added to the DUMPANALYZER.

The new file type REMOTE BACKUP has been added to the DUMPANALYZER. FIB's of this type will be analyzed correctly.

D2453 DUMPANALY - "HELP" COMMAND IMPROVEMENTS

The HELP command has been expanded to provide a railroad syntax diagram and a brief summary of the various DUMPANALYZER commands. The new syntax for the HELP command is:

```
-- HELP ------------------------------------------------------------|
          |                        |
          |-<dumpanalyzer command>-|
          |                        |
          |-<meta item>------------|
```

As before, entering "HELP" provides a list of available commands. Entering "HELP <dumpanalyzer command>" provides a summary of the requested command. Entering "HELP <meta item>" provides the meta syntactic items used in the syntax of the commands.

D2551 DUMPANALY - "STANDARD" COMMAND

A new command, STANDARD, has been added to the DUMPANALYZER. This command, effective in both interactive and ODT mode, allows the user to revert to the noninteractive analysis method; i.e., the user may give up an interactive analysis and analyze a dump by the old methods.

Syntax:

```
-- STANDARD --|
```

The command must be verified by entering YES when asked. The user may then enter the card-mode images at the terminal or ODT, followed by end of file (?END).

D2593 DUMPANALY - "STACK" BOUNDS

The syntax of the STACK request in the Interactive Dumpanalyzer has been extended to permit specification of a lower-bound offset as an optional third number.

Example:

```
    STACK 3E 2F7 2A0
```

This example will dump stack number 3E, starting at offset 2F7 and stopping after offset 2A0.

The upper bound is effectively unrestricted; the lower bound must not exceed the upper.

D2635 DUMPANALY - PORT AND SIGNAL ANALYSIS

Analysis of ports and signals may be invoked via the batch options "PORT" or "SIG". Setting either option will cause analysis of both ports and signals.

DUMPANALYZER will recognize modes of "PORT" or "SIG". This allows syntax such as the following:

```
    MODE + PORT
    PV M[03BAC] PORT
```

As with batch analysis, "PORT" and "SIG" are synonymous and setting either one will cause analysis of both ports and signals.

D2656 DUMPANALY - DUMP ANALYZER CHANGES

Several changes in the structure of the III.1 MCP have necessitated changes in the DUMPANALYZER, including some new and altered syntax in its control statements.

STRUCTURAL CHANGES

Task and Stack Structure

The stackvector has been rearranged, with genuine process stacks and segment dictionaries at the bottom and other things at the top end. These other things include the diskfile header "stack", the DCALGOL queue "stack", and the memory-spanning pseudostacks.

The "TASK" variable associated with a process stack is now called a PIB (Program Information Block).

The descriptors for active PIBs no longer live in the base of the process stack; instead they are in a SPIBVECTOR parallel to STACKVECTOR.

Boxes

Each memory subsystem in a B6800 Multiprocessor (tightly-coupled) system is called a "box": the Global memory subsystem is box 1; the local memory subsystems are numbered with the associated processor id+1. The box number for each memory word is recorded on the dump tape as the tape is written. Global memory addresses are unique; they are all above first-word-address-of-global. Local memory addresses are ambiguous, since each local memory subsystem has addresses beginning at zero.

The memory subsystem for each stack is displayed in the first column of the SUMMARY output for a B6800 Multiprocessor system: G for global, 1 for processor 1 (box 2), etc. The column is blank for a stack whose box is not currently assigned. Box and/or processor numbers appear in the heading for each stack dump, for the area dump from each box and for the descriptor analysis.

NON-INTERACTIVE SYNTAX

Two new option keywords have been added to the STANDARD-mode input:

HDR causes the disk-file header stack (and its associated locks) to be printed.

QUE causes the DCALGOL queue stack to be printed.

The MIX=ALL specification will continue to print these "stacks", but STK=11 will no longer get the header stack, for example.

INTERACTIVE SYNTAX

The <simple loc> construct no longer includes the forms

```
            STK <stack number> TASK <attr name>
        and STK <stack number> TASK # <offset>
```

These have been replaced with the new <simple loc> constructs

```
            PIB <stack number> <attr name>
        and PIB <stack number> # <offset>
```

Examples:

```
        MD PIB 2E7 SERIAL
        MD PIB 2E7 # 5F
```

These examples print information from the TASK variable associated with stack number 2E7. The first prints the SERIAL word; the second prints word 5F (zero-relative).

Several new commands are available:

BOX <number>

causes the dump analyzer to interpret all non-global addresses as being in the specified box. The box specification is also changed by a STACK command for a stack in a local box, to designate that box. The command "BOX" with no <number> shows the current setting.

BOXINFO

prints (as raw arrays) the BOXINFO array for each box in the system.

PIB <stack number>

prints the task assocated with the specified stack.

PIB <simple loc>

prints the task whose descriptor is at the specified location.

QUE prints all the DCALGOL queues.

QUE <number>

prints the queue whose queue number is specified.

The HDR <number> command (and its variations) has been modified to print the single diskfile header whose header index is specified.

The SEARCH command operates on only one memory subsystem in a tightly-coupled system. The BOX command may be used to select the subsystem to be searched.

D2720 DUMPANALY - NETWORK ANALYSIS

The standard option NETWORK will now print an analysis of the network if the dumping host was part of a network at the time of the dump.

Two new interactive commands have been implemented.

HOSTINFO

```
-- HOSTINFO --|
```

The HOSTINFO command displays information about all network hosts.

SHAREMEM

```
-- SHAREMEM ----------------------|
                |                 |
                |-<logical mod #>-|
```

The SHAREMEM command is used to get the status of shared memory on a loosely-coupled system.

D2845 DUMPANALY - "MCP" NAME HANDLING

The dump analyzer handling of names for MCP stack locations has been redesigned. Noticeable effects of the change are the following:

1. The LINEINFO/NAMES portion of dump analyzer initialization is much faster.
2. All aliases for MCP stack cells are retained and will be returned by the WHO command and recognized by the WHERE command.
3. Interpretation of code addresses from RCWs or PCWs has been improved as follows:
   a. The UNKNOWN/INNER BLOCK designation will no longer appear. In most cases a name is known and will be displayed; if no name is known, none will appear.

   b. The $USERSEGMENT designation will no longer appear. Instead, the name of the first procedure in the segment will be used, suffixed by ",ETC".

   c. When several outer-block procedures have code in the same segment, the name of the particular procedure containing the addressed code will be displayed. (This feature was formerly available only for segment-5 procedures, and was often inaccurate.) When inner procedures reside in the same code segment as the containing block, the name of the containing block will be displayed, as before.
4. The NONAMES specification will stop production of the tables of names and addresses in a non-interactive dump, but will not prevent initialization of the name data and its display in association with stack blocks, memory areas, etc.
5. When a table of names and addresses is produced, either non-interactively or in response to the NAMES command, the SORT intrinsics is invoked to sort the names alphabetically. During this time, the HI response is "SORTING MCP NAMES". There is no longer any sorting during initialization, so in general the interactive dump analyzer working set is substantially reduced.

D2846 DUMPANALY - AREA DUMP HEADINGS

The information in the in-use-area headings for an areadump have been modified. Such headings may comprise three lines: If the mom address for the area is in the MCP D0 stack and the name of that cell is available, the name is printed as the first line. The next line is the basic heading for the area. The final line is optional under the linkdump option; it displays the three links before and the link after the area.

The D0-item name, if any, is now right-justified in the blank line which separates adjacent area listings.

The heading line for an in/use area has been revised:

```
The first word is   SAVE   for a permanently-save area,
                   CSAVE   for a currently-save area,
                    OLAY   for an overlayable area.

The second word is  CODE   for a code segment,
                    DATA   for a normal data area,
                     R/O   for a read-only data area.
```

Minor punctuation changes have been made in the same line.

The memory-link line for an in-use area has been revised:

> The USE-nnn field has been eliminated and the line left justified. (The USAGEF field is the second quarter of the second word of links. Parts of this field are already interpreted on the heading line in deriving the notations SAVE, CODE, R/O, and type (FIBMARK, DOPEVECT, etc.).
>
> The LINKC word of most areas is an RCW that indicates where the area was made present. If the RCW is D0-relative, the line number and name of the MCP code segment is displayed. Such interpretation will be invalid for some areas at some times, as when an IOCW is in place for overlay. The interpretation is not attempted for FIBs, SIBs or Direct Arrays, since these never have an RCW in LINKC. The interpretation is also suppressed for code segments, to avoid clutter.

D2856 DUMPANALY - INTERACTIVE SYNTAX ERRORS

The syntax of the interactive request IO has been corrected. The following diagram should replace the syntax diagram for the HELP command on Page 14-38 of the System Software Operational Guide, Vol. 1 (Form No. 5001563).

```
<unit part>

------------------------------|
   |                          |
   |- ACTIVE ---------------- |
   |                          |
   |- UNIT --<simple value>-  |
```

The following paragraph should replace the first full paragraph of semantics on Page 14-38.

> "The IO (input output) command invokes I/O analysis. (See figure 5.) If the <unit part> is empty, all units are analyzed. If a particular unit is specified, only that unit is analyzed; otherwise, all units satisfying the condition specified are analyzed. An active unit is one whose I/O queue is non-empty; it may also be assigned to an MCP stack, etc."

D2939 DUMPANALY - CODE DUMP

If the CODE option is not set, DUMPANALYZER does not print code areas. Now read-only data areas are suppressed under the same option.

DUMPANALYZER will no longer scan code areas to verify tag=3 for each word. Previously, it forced printing of such areas and recorded them as having bad memory links; however, such instances are not usually pathological.

D2944 DUMPANALY - INTERACTIVE MODE

The initial options when DUMPANALYZER is run are:

> SELECT RUN MODE: STANDARD OR INTERACTIVE

The minimum acceptable response is "S" or "I". For compatibility with previous versions, the response "EXTEN" is synonymous with "I".

If DUMPANALYZER is run with an odd task VALUE, the query is skipped, INTERACTIVE mode is selected, and the "HELPful" list of commands is omitted.

D2945 DUMPANALY - "IO" UNIT SELECTION

The criteria for printing I/O units have been revised to print fewer non-existing units when the interactive I/O command is used or when DUMPANALYZER is run in STANDARD mode.

When the interactive analyzer is given an "IO UNIT <n>" command, the unit information is supplied for any <n> in the unit-table range. (Non-existent peripheral controls are designated "pseudo reader".)

In addition to a decimal integer, the following catalog pseudo-unit names are accepted:

> CATUNIT
> OFFLINEUNIT
> VOLUNIT
> TAPEUNIT

The first syllable is sufficient as an abbreviation.

D2949 DUMPANALY - DESCRIPTOR ANALYSIS

FORMATTING

The report has been reformatted for increased readability. The old format appeared as follows*

```
    AREA        LENGTH  DESCRIPTOR     AT      DESCRIPTOR     AT
 SV 020F9-02E7F (03463) 8000D8 4020F9 (5E087) E25000 0023AD (023CB)
 OV 0358E-03599 (00114) C00000 C0358E (02F38)
```

In the new format, the parentheses are eliminated, the decimal length is shown with leading zeros suppressed, the locations precede the descriptors:

```
    AREA        LENGTH   LOC  DESCRIPTOR     LOC  DESCRIPTOR
 SV 020F9-02E7F   3463  5E087 8000D84020F9  023CB E250000023AD
 OV 0358E-03599    114  02F38 C00000C0358E
```

In a B6800 multiprocessor (Tightly-Coupled) system, a separate report is printed for the descriptors in each memory subsystem, and the box number precedes each descriptor location. For example:

```
 = = = = = = = = = = = = = GLOBAL MEMORY (BOX 1) = = = = = = = = = = = =

    AREA        LENGTH  BX LOC  DESCRIPTOR     BX LOC  DESCRIPTOR
 SV 820F9-82E7F   3463  1 5E087 8000D84820F9   3 023CB E250000823AD
 CS 9358E-93599    114  2 02F38 C00000C9358E
```

Note that the box number, not the processor id, is shown. Note also that a descriptor in Global memory which contains a local address is ambiguous; it cannot be matched uniquely with an area in a particular box. Such descriptors will be listed in the final "MISC DESCRIPTOR" report.

OTHER CHANGES

DUMPANALYZER capacity has been increased from 5000 areas and 14999 descriptors to 15000 areas and 50000 descriptors. Should these limits be exceeded, DUMPANALYZER will disregard the excess data and report the number of areas/descriptors not printed. (Formerly, the program got invalid-index faults when the arrays were exceeded.)

Code areas are no longer considered in the descriptor analysis. Each area is now reported as SV, CS or OV for save, current-save or overlayable memory, respectively.

D2950 DUMPANALY - LOCKS, EVENTS, "DEADLOCK" ANALYSIS

DUMPANALYZER handling of locks and EVENTs has been revised.

LOCKS

The LOCKS command is available in interactive mode, or invoked automatically in standard mode. Several changes have been made:

1. The "hard" locks are no longer reported.

2. All EVENTs and EVENT ARRAYs declared in the MCP outer block are now considered; only those which have been PROCUREd or are being awaited are listed.

3. The "ABNORMAL" qualifier has been eliminated from the LOCKS command.

DEADLOCK

The DEADLOCK command is available in interactive mode, or invoked automatically in standard mode. Its output has been reformatted, error checking has been improved, and it now considers all EVENTs and EVENT ARRAYs in the MCP outer block. It no longer considers the "hard" locks.

HDR

The HDR command is available in interactive mode, or invoked in standard mode by the HDR option or MIX=ALL specification. The resulting report no longer shows the HEADERLOCK event, since these are reported by the LOCKS and DEADLOCK commands. Therefore, the LOCKS and LOCKS ABNORMAL qualifiers have been deleted from the HDR command.

EVENT Analysis

The analysis of EVENTs has been abbreviated by stating only present conditions. Thus, for example, an EVENT may be marked HAPPENED or UNAVAILABLE; it will no longer be marked NOT HAPPENED or AVAILABLE (the default conditions).

Warning

The MCP global EVENTs and EVENT arrays are located from (BINDINFO) data in the MCP code file. If the analyzed code file does not match the dumping MCP code file, the event analysis may be garbage.

D2966 DUMPANALY - "NOIO AND UINFO" OPTIONS

Some options have been changed for DUMPANALYZER's STANDARD mode.

A new option, NOIO, will suppress the analysis of I/O units that is produced by default. DATACOMONLY and DMSIIONLY implicitly set NOIO.

The UINFO option no longer analyzes the long arrays which hang off the UINFO array; thus, UINFO (which is set by default) is synonymous with QUICKUINFO. A new option, UINFOALL, will cause these arrays to be analyzed; that option is not set by default. (As before, NOUINFO will reset the UINFO option and suppress UINFO analysis.)

DUMP ANALYZER

P1109 DUMPANALY - HARD LOCK AND "RCW" ANALYSIS

Whenever the DUMPANALYZER encounters an RCW referring to the MCP stack, the procedure name is printed if available.

If the READLOCK MCP option were set, hard locks were BUZZ'd with a value which could be treated as an RCW; now, these locks are shown with sequence number and procedure name. The displays affected include "PV ... LOCK".

P1110 DUMPANALY - "MCP" TITLE FROM HEADER

The DUMPANALYZER now uses the title of the MCP code file derived from its header, not the value passed on the dump tape from MCPINFO.

The change means that a correct file name will be displayed and used as the default MCPCODEFILE title in cases where MCPINFO is wrong or incomplete, such as after a CM# or with duplicated MCP.

P1213 DUMPANALY - "SEARCH" COMMAND ACCELERATION

The SEARCH command has been accelerated considerably. It should be noted that the search now occurs from the high end of memory toward 0, so that memory indices printed in response to "?AX WHERE" will decrease.

P1520 DUMPANALY - "IO" DEADLOCK ANALYSIS

The analysis of deadlock has been refined for stacks waiting on I/O, as follows: A stack will be reported as waiting for unit/path only when actually waiting, not just because it has I/O requested.

P1707 DUMPANALY - UNKNOWN RUN TIME OPTIONS IN "MCP"

A problem which could cause truncation of the listing of an "**UNKNOWN**" option in the list of run-time MCP options has been corrected.

P1708 DUMPANALY - "MOD 63" GIVES FALSE TAPE ERROR MESSAGE

A problem has been corrected in which having a memory module #63 in the system when a dump is taken leads to an extraneous "BAD TAPE INFORMATION..." error message.

P1793 DUMPANALY - MISCELLANEOUS CORRECTIONS

A problem has been corrected in which a dangling "STACK" specification card could cause a failure.

Other problems relative to III.1 implementation have been corrected.

P1893 DUMPANALY - "SIRW" ANALYSIS

SIRWs in stacks are now analyzed correctly. Previously, extraneous bits not in the DELTA field could cause the offset to be off.

P2083 DUMPANALY - SEQUENCE NUMBERS FROM "RCW"

RCWs are now correctly converted to line numbers.

P2084 DUMPANALY - FAULT HANDLING

The reasons displayed for DUMPANALYZER faults will now be correct.

P2321 DUMPANALY - CLOSE INTERACTIVE "OPTIONS" FILE

When running from a terminal in STANDARD mode, DUMPANALYZER will now close its remote I/O file when it detects EOF (as ?END is entered) before it possibly tries to write error message to that file; thus, any message will reopen the file instead of being rejected.

P2432 DUMPANALY - COPE WITH "FIB" INDEX ARRAY NOT YET SETUP

If a tape dump is taken during Halt/Load initialization before the MCP has set up the FIBINDEXARRAY, DUMPANALYZER will fault during initialization. This fault is now trapped, appropriate messages displayed, and the dump analysis continues without the table (which provides names for words in the FIB). There are no FIBs at this point anyway.

P2515 DUMPANALY - "DCC" AND "DCP" STATION INFO

DUMPANALYZER will now print the correct data for the DCC station table; these data have been incorrectly displayed since the II.9 release.

Only the first six words of DCP station information are now displayed for each station on each valid line. These words have the same structure for all stations on all systems. Subsequent words vary in number and content for various stations and various systems, and will not be displayed by DUMPANALYZER. (On earlier systems, the seventh and eighth words were sometimes displayed as sequence parameters.)

P2520 DUMPANALY - SAVE FILE "AREASIZE"

The AREASIZE of the NEWTAPE file (used for SAVE) has been reduced from 40 to 15 records (2000 to 750 segments).

P2521 DUMPANALY - BREAK ON OUTPUT

The handling of Break on Output has been improved in DUMPANALYZER. In particular, a break during initialiation will no longer cause part of initialization to be skipped.

P2523 DUMPANALY - SPECIAL ARRAYS

The analysis and display of special-type arrays in a stackdump have been corrected and improved, as follows:

1. If the array is a GETAREA area (such as a DCALGOL MESSAGE), it will be noted as such and not confused with a GETSPACE area.

2. All GETSPACE areas with ODDBALLF non-zero are identified as to type, not just as FIBs.

P2524 DUMPANALY - REMOTE LINE WIDTH

The handling of line width for the remote terminal has been improved, as follows:

1. The station is attached to the remote file before the station attributes are sensed, so the specification in a CANDE TERM command will be effective.

2. The full line width is used for hard-copy terminals, one less for screen devices. (Formerly, one was subtracted for all terminals.)

P2540 DUMPANALY - IMPROVED INITIALIZATION AND FILENAME HANDLING

DUMPANALYZER stack-information handling has been reorganized. The most noticeable effects are the following:

1. Dumps taken with a large mix initialize faster.

2. Complete file names are kept, regardless of length; complete names are displayed (on multiple lines if necessary) except in tabular output.

3. Long MCP names will no longer cause attribute errors on MCPCODEFILE.

4. If DUMPANALYZER is running with FAMILY substitution in effect, it looks for MCP and program code files first without and then with the FAMILY specifications.

P2565 DUMPANALY - "TAG 4" AND "TAG 6" ANALYSIS

DUMPANALYZER now indicates library template markers in TAG 6 words, and distinguishes various types of TAG 4 words by their FAULTMARKF field.

P2581 DUMPANALY - "PV" AND STACKDUMP OPERANDS

The formatting of decimal operand values is now more flexible; integer, fixed-point and exponential notation will be used as appropriate.

EBCDIC operands are so printed only if all characters are graphics; however, if an operand contains right-justified EBCDIC characters with NULL fill, the EBCDIC characters will be printed.

In the construct "PV M[a] . . .", if M[a] and M[a+1] are both Tag-2 words, they will be analyzed together as a double operand. This is effective in DEC, OCT, BCL, EBC and LOCK mode.

P9248 DUMPANALY - STANDARD MODE FROM THE TERMINAL GETS ERRORS

A problem has been corrected which could sometimes cause the DUMPANALYZER to erroneously issue a "BAD DUMPANALYZER INPUT CARDS" message when run in STANDARD mode from a terminal.

P9249 DUMPANALY - CORRECT "CANNOT ANALYZE" MESSAGE

The problem which caused the "CANNOT ANALYZE - MCP INCOMPATIBLE WITH DUMPANALYZER" message to be truncated has been corrected.

P9277 DUMPANALY - SUPPLY INFORMATION ON INCOMPATIBILITY "DS"

More information wil! now be supplied when a situation is encountered which results in the message "UNABLE TO ANALYZE--MCP NOT COMPATIBLE WITH DUMPANALYZER". The version level word, which is an indicator of the general MCP-DUMPANALYZER compatibility, will be printed along with the specific MCP level word from the dump tape. If the DEBUG option is set, the debug information, consisting of the first 130 words of the dump tape, will now be printed; previously, it was not. The formatting of this information has been improved.

ESPOL INTRINSICS

D2595 ESPOLINTRN - COMPILING "ESPOL" INTRINSICS

An "INFO/MCP" file is no longer required for compiling the ESPOL intrinsics. Instead, a few critical defines from "SYMBOL/MCP" are included during the compile, using the ESPOL file named "MCP"

D2680 ESPOLINTRN - FREEFIELD OUTPUT FOR COMPLEX VALUES

Named output for complex expressions and variables in ALGOL have been implemented. The named output for complex values in FORTRAN has been changed.

ALGOL:

The statement

```
WRITE (F,*1,COMPLEX(3,4))
```

generates the following output:

```
COMPLEX(3,4):R=3,:I=4
```

FORTRAN:

The program

```
COMPLEX C1,C2,CA(2)
WRITE (1,*1),C1*C2,CA
STOP
END
```

now generates the following output:

```
<exp>:R=0.0,:I=0.0,CA(1):R=0.0,:I=0.0
CA(2):R=0.0,:I=0.0
```

D2817 ESPOLINTRN - "BASIC," BLOCKING OF DISK FILES

The blocking of BASIC disk files created after a margin statement has been redesigned to provide for better disk and memory usage and readability through CANDE. The redesign attempts to block the disk file so that block fits exactly on a disk segment boundary and occupies as close to and not exceeding 3060 characters as possible. If this is impossible, the blocking will be allocated so that block fits on a word boundary and does not exceed 3060 characters. If this is impossible, the file will be blocked six records per block.

D2818 ESPOLINTRN - "BASIC," FILE HANDLING CHANGES

If a file is declared in a BASIC program, unreferenced and non-existent, an empty file will no longer be created at the end of the program.

D2819 ESPOLINTRN - NEW DISPLAY OF FORMAT ERRORS

The display of format errors has been changed. The non-binary display has been enhanced by including an English description of the error, the filename, the record number from the file, and the character position at which the formatting intrinsics were editing. The binary display will include the words FORMAT ERROR before the already-existing English description, and a format error number, the filename, the record number and the character position.

The new display of format errors is as follows:

```
<task number> FORMAT ERROR (<error message>) #<error number>:
   <filename> @ REC #<number>, CHAR #<number>
```

Example:

```
5367 FORMAT ERROR (RECORD OVERFLOW) #217: FILEF @ REC #13
   CHAR #72
```

This error message would indicate that the task number 5367 had a format error number 217, which is a RECORD OVERFLOW, while reading the file FILEF at record number 13 and at character position 72.

If the task is reading or writing to an array row rather than a file, the record number displayed will be zero; instead of a filename, "ARRAY ROW" will be displayed. The display of format errors pertaining to format specifications in arrays will not include the record number and character position, and FORMAT SYNTAX ERROR will be displayed in place of FORMAT ERROR.

The new format error display will not include the display of the history. Such a display was uninformative because it contained no LINEINFO, and the address specified was simply an address within the intrinsics. It would also be redundant due to the fact that after the format is encountered and the program is DSed, the user receives a complete stack history at the time of

ESPOL INTRINSICS

P1453 ESPOLINTRN - "STATISTICS" OPTION

The intrinsic which prints timing information for the STATISTICS option now uses the TIME(12) (process time) instead of the TIME(11) function in improving upon the times noted with asterisks.

P1841 ESPOLINTRN - "BASIC" BINARY FILES

Attempting to create a binary file with the BASIC FILE statement no longer causes an INVALID INDEX termination in the BASIC file manipulation intrinsic.

P2141 ESPOLINTRN - CORRECT "R" FORMATS

A problem with the following format has been corrected:

```
    wRf.d where w is a constant
                d is a *
```

P2142 ESPOLINTRN - ELIMINATE LOOP ON INVALID CHARACTER

The intrinsic FORTALGFORMATENCODER will no longer loop when trying to evaluate an invalid character as a number.

P2143 ESPOLINTRN - DETERMINE LENGTH OF CHARACTER ARRAYS

A problem has been corrected that occurred when doing a read or write using a non-indexed character array as the file part. The number of characters in the array was incorrectly calculated.

P2144 ESPOLINTRN - CORRECT INVALID INDEX

Previously, F3.0 and F4.0 formats would get an INVALID INDEX in the intrinsics if the number did not fit in the fields specified. Now, asterisks are placed in the field.

P2145 ESPOLINTRN - "BASIC," BINARY FILES

Binary files will now be created with records of the correct size (i.e., 14 words).

P2146 ESPOLINTRN - "MONITOR" DOUBLE VARIABLE

The MONITOR intrinsics will no longer get an exponent overflow when formatting a very large double precision numbers.

P2147 ESPOLINTRN - CORRECT "E" FORMAT

Values approaching one will be formatted properly for output from the following format:

```
  Ew.d with d greater than 22 for ALGOL
            d greater than 24 for FORTRAN
```

P2148 ESPOLINTRN - "BASIC," CORRECT BUFFER FLUSHING

BASIC file buffers will no longer be flushed inappropriately at wrap-up.

P2149 ESPOLINTRN - CORRECT NUMERIC REPLACE

A problem in which the intrinsic OUTPUTCONVERT would incorrectly print the least significant part of a double precision number has been eliminated.

P2150 ESPOLINTRN - PREVENT EXTRA PRINTING

A FORTRAN format ending with a "/" being written to a file with UNITS=CHARACTERS and with a MAXRECSIZE divisible by 6 will no longer output a character left over in the buffer instead of a blank line.

P2151 ESPOLINTRN - "BASIC, PRINTUSING"

PRINTUSING will now correctly print an integer if the integer field is all of the following:

1. preceded by a $, and

2. the last element in the image, and

3. the image is declared via the format

```
        <line number> : <image element>
```

or in an image declared via the format

```
<string exp> : "<image element>"
```

where there is no space or other character between the integer field and trailing quote.

P2152 ESPOLINTRN - "BASIC," EMPTY FILES

The following BASIC construct has been corrected:

```
FILE # <number>, "*"
```

Previously, if a file were created in the progiam, closed in this manner, and never again used, an empty file would result. An example of such a program is the following:

```
FILES DEF
SCRATCH #1
WRITE #1, "ANYTHING"
FILE #1, "*"
```

Upon execution of the FILE statement, the file DEF was closed, containing "ANYTHING". At the end of the program, file #1 was automatically closed, with an empty file replacing the previous file DEF. The additional closing at the end of the program was erroneous. Now a valid file will not be replaced by an empty file in this situation.

P2153 ESPOLINTRN - TASK "DS"

A task that is DSed from the ESPOLINTRINSICS will now have the task history word properly set up indicating the reason for the error termination. In addition, the task will be recorded and displayed as P-DS. The following program-caused DS reasons have been added:

INVALIDPARAMV	82
FORTRANERRV	83
PLIRUNTIMEERRV	84
MATHERRV	86
INTRINSICSERRV	85
FORMATERRV	87

P2154 ESPOLINTRN - MONITOR DOUBLE ARRAYS

Double precision arrays are now monitored correctly.

P2155 ESPOLINTRN - SCALE FORMAT PHRASE

Scaled formats via the "S" format phrase will now produce correct results. Incorrect results depended upon input field position.

P2156 ESPOLINTRN - FREE FIELD OUTPUT

Incorrect subscripts (in the range of 65536 to 99999) will no longer be printed for writes such as the following:

```
WRITE(F,*//,A[66000]).
```

P2157 ESPOLINTRN - BINARY WRITE QUESTION MARKS

Question marks were incorrectly being printed for certain cases of binary I/O. An example of such a case if the following:

```
WRITE(F [SPACE 2],*,A).
```

P2160 ESPOLINTRN - "BASIC," RESIZE FORMAT BUFFER

ESPOLINTRINSICS will now properly resize the format buffer. Previously, the intrinsics BASICINEI, BASICOUTEI and BASICPRINTUSING made the occasionally incorrect assumption that the MAXRECSIZE of the file was in terms of characters.

P2162 ESPOLINTRN - "BASIC," STRING COMPARISONS

An extremely rare problem has been corrected where a BASIC program doing string comparisons could cause a system failure due to manipulation of absolute memory addresses in normal state.

P2542 ESPOLINTRN - FORMATTING INTRINSICS

The formatting intrinsics no longer incorrectly call the FORGETSPACE procedure of the MCP. Previously, a dump by BLOCKEXIT MOM could have been caused if a task were DSed when an intrinsic had just called FORGETSPACE for a local array.

P2543 ESPOLINTRN - "GMM" ROUTINES

The P2 and P3 values returned by global intrinsics GMMSCAN and GLOBALMEMORYTEST were swapped; they are now correct.

P2591 ESPOLINTRN - "FORTRAN BCL" FORMATTING

The use of BCL strings in run-time formats no longer produces spurious results.

FILECOPY

D2826 FILECOPY - "II.9 WFL"

It should be noted that if the WFL deck is punched by FILECOPY and then read into a SECURED READER, an error will abort the WFL. A user card must be added to the punched deck.

D2941 FILECOPY - TIME VALUE OF SECOND TIMESTAMP IN "BETWEEN"

FILECOPY will now correctly copy files when the UPDATED modifier BETWEEN is used and no time part is supplied with the timestamps (or TODAY is used). The problem was caused by the time field of the file's timestamp being greater than the time field of the second timestamp of the BETWEEN statement.

D2946 FILECOPY - VOLUME SPECIFICATION NOT ALLOWED FOR "DISK"

FILECOPY will now give a syntax error when volume specifications (e.g., KIND=PACK) are supplied for media names of DISK or PACK. The error messages reads as follows:

```
"NO VOLUME SPEC. ALLOWED FOR DISK OR PACK"
```

FILECOPY

P1847 FILECOPY - "II.9 WFL"

The WFL decks that are punched or zipped by FILECOPY will now be in II.9 WFL form.

P2188 FILECOPY - PUNCHED OUTPUT

FILECOPY will now create its punched output decks without the error INCOMPAT BLOCKING:T1OUTPUT.

P2555 FILECOPY - "(X)" VS. *X IN "EXCLUDE" LIST

FILECOPY was not distinguishing between system (*) files and user ( ) files in the EXCLUDE list. When the first directory identifier of a system file in an EXCLUDE list was identical to a USERCODE identifier in a FILE list, the USERCODE would be EXCLUDEd. This problem has been corrected.

FILEDATA

D2876 FILEDATA - "FILEORGANIZATION" ATTRIBUTE

A new attribute, FILEORGANIZATION, has been added to the valid attributes for FILEDATA. It will only be shown of "data" files; i.e., files with FILEKIND > OR = VALUE(DATA).

The System Software Operational Guide, Volume 1 (Form No. 5001563), should be changed as follows:

Page 5-1-8:

The syntax diagram should be changed to read as follows:

```
-- ATTRIBUTES -----------------------------------------------------|
              |                                                   |
              |- : -----------------------------------------------|
                   |                                              |
                   | |<-----------------------------------|       |
                   | |                                    |       |
                   |----- ABBREVIATED ------------------------|
                      |   ----                          |
                      |- ALL ---------------------------|
                      |                                 |
                      |- CATALOGUE ---------------------|
                      |  ---                            |
                      |- DOUBLE ------------------------|
                      |  ---                            |
                      |-<file attribute>----------------|
                      |                                 |
                      |- LASTACCESSDATE ----------------|
                      |  -----                          |
                      |- LEVEL -- = <integer> ----------|
                      |  ---                            |
                      |- NAMESONLY ---------------------|
                      |  ---                            |
                      |- NEWDATABASE -- = <filetitle> --|
                      |  ----                           |
                      |-/1\--- PRINTER -----------------|
                      |     |  ---                      |
                      |     |- PUNCH -------------------|
                      |     |  ---                      |
                      |     |- SCREEN ------------------|
                      |        ---                      |
                      |- DATABASE ---- = <file title> --|
                      |  ---        |                   |
                      |- DIRECTORY -|                   |
                      |  ---                            |
                      |- PACKNAME -- = <identifier> ----|
                      |  ---                            |
                      |- TAPE -- = ---<tape name>-------|
                      |  ---        |                   |
                      |             |-<unit no>---------|
                      |                                 |
                      |- TIMESTAMP ---------------------|
                         --
```

Page 5-1-9:

The description of <file attributes> should be replaced by the following syntax diagram:

```
<file attribute>

---- AREAS ----------------|
   |                      |
   | - AREASIZE ---------|
   |   -------           |
   | - BLOCKSIZE --------|
   |   --                |
   | - CREATIONDATE -----|
   |   ---               |
   | - CRUNCHED ---------|
   |   ---               |
   | - CYCLE ------------|
   |   --                |
   | - FILEKIND ---------|
   |   -----             |
   | - FILEORGANIZATION -|
   |   -----             |
   | - FILETYPE ---------|
   |   -----             |
   | - INTMODE ----------|
   |   --                |
   | - LASTRECORD -------|
   |   -----             |
   | - MAXRECSIZE -------|
   |   --                |
   | - SAVEFACTOR -------|
   |   --                |
   | - SECURITY ---------|
   |   --                |
   | - UNITS ------------|
   |   -                 |
   | - VERSION ----------|
       -
```

Page 5-1-21:

Add the following description of FILEORGANIZATION:

"FILEORGANIZATION

FILEORGANIZATION is the organization under which the file was opened.

FILEORGANIZATION is shown only if FILEKIND > OR = VALUE(DATA)."

Page 5-1-23:

Add the following description of TIMESTAMP:

"TIMESTAMP

The date and time the last alteration was made to the file."

FILEDATA

P1364 FILEDATA - ACCESSCODE

SECURITYTYPE CONTROLLED (CLASSC) is now recognized.

P2296 FILEDATA - ALLOW "CANDE" "LFILES" WITH NO DISK SYSTEM

A CANDE request for LFILES <filename> ON <packname> on a system with no disk no longer hangs on a "REQUIRES DK DISK". Note that the system will still mention "NO FAMILY DISK" in passing.

P2468 FILEDATA - "6250 BPI" TAPES RECOGNIZED

FILEDATA obtains tape density information from GETSTATUS, which it then uses to index a value array for report headings. An entry has been added to that value array for the 6250 BPI density setting.

FORTRAN

D2532 FORTRAN - COMPILER FILE LINE

Table 18-1 of the FORTRAN Manual (Form 5000458) should be corrected. The COMMENTS for file LINE should say:

> "Optional and label-equatable to REMOTE file produced
> when either the compiler option LIST or TIME is set."

D2821 FORTRAN - FORMAT-REPEAT COUNT

If a repeat count greater than 65535 is used, a warning is issued and a repeat count of 65535 is used.

D2822 FORTRAN - BINDING AND STATISTICS

It is not possible to bind programs with DEBUG STATISTICS. If a DEBUG STATISTICS statement is encountered, no bind information will be generated. If SEPARATE, LIBRARY or AUTOBIND is set, an error will be flagged; otherwise, if NOBINDINFO is not already set, a warning will state that no bind information will be generated.

FORTRAN

P1842 FORTRAN - INVALID SYNTAX, "IMPLICIT" STATEMENT

It was possible for an invalid type specification in an IMPLICIT statement to pass syntax. An appropriate error message will now be generated.

P1843 FORTRAN - DOUBLE PRECISION CONSTANTS "OPT=1"

Double Precision constants whose magnitudes were less than 2**16 were being interpreted as single precision in OPT=1, causing a loss of accuracy when dividing. This has been corrected.

P1844 FORTRAN - INVALID TYPE CHECKING OF SUBROUTINES

When the name of a subroutine, which was passed as a parameter, was included in the scope of an IMPLICIT statement, a spurious syntax error was generated.

Example:

```
IMPLICIT DOUBLE PRECISION (A-Z)
EXTERNAL SUB2
CALL SUB1(SUB2)
END
SUBROUTINE SUB1
CALL P1
END
SUBROUTINE SUB2
END
```

This has been corrected.

P1845 FORTRAN - MONITOR BINDER INTERFACE

If the binding of a subroutine caused the addition of D2 stack cells and the subroutine used the monitor debug facility, erroneous assignment of stack cell locations could result. This has been corrected.

P1848 FORTRAN - RUN-TIME GROUP REPEAT

If the run-time FORMAT option asterisk (*) appeared as a repeat count and control reverted to the asterisk, the group would be repeated indefinitely with no list item accessed to replace the asterisk. This has been corrected.

P1849 FORTRAN - BAD "PCW" ON ACTION LABEL

In subroutines which occupy more than one segment, the compiler failed to properly generate a PCW for an I/O exception branch. This would typically result in program termination with INVALID PROGRAM SYLLABLE. This has been corrected.

P1850 FORTRAN - MULTIPLE ENTRY POINTS WITH "OPT=1"

When a subroutine contained several ENTRY points, the possibility existed of bad code being emitted or a COMPILER ERROR being generated. This occurred ony with OPT=1. This has been corrected.

P1874 FORTRAN - BINDING "FORTRAN" AND "ALGOL"

The BINDINFO generated by the III.0 FORTRAN compiler for some intrinsics (such as FORMATTEDOUTPUT) was not compatible with the BINDINFO generated by the ALGOL compiler or the BINDINFO generated by the FORTRAN compiler before III.0. This caused interlanguage binding and binding of III.0 FORTRAN to previous levels to fail in some cases. This has been corrected.

P1934 FORTRAN - INVALID SYNTAX IN DATA LISTS

When the letter O is used in a data list for arrays, followed by a comma and another list element, the next element is taken as an actual digit and an incorrect error is produced. The correct error, ILLEGAL CONSTANT IN DATA OR INITIAL VALUE LIST, is now flagged.

P1935 FORTRAN - REPEAT COUNT, LEADING ZEROS

Leading zeros in repeat counts of a format phrase were being treated as significant digits, resulting in syntax errors under some circumstances. This situation has been corrected.

P1936 FORTRAN - "FREE" FORMAT VS. "INCLUDE"

The compiler structures for manipulating INCLUDEs and handling FREE formatting have been cleaned up, thus resolving contention problems for the previously shared structure which could sometimes lead to spurious syntax errors.

P1937 FORTRAN - "INVALID INDEX," SUBROUTINE CALL

When a subroutine is incorrectly called with no parameters, it is correctly flagged as an error; but, if the same subroutine is then correctly called, the compiler faults with an INVALID INDEX. This has been corrected.

P1938 FORTRAN - INTERNAL COMPILER FIELDS EXCEEDED WITH "OPT=1"

When large programs were being compiled with OPT=1, it was possible to get a compiler error if the internal compiler fields were exceeded. An error will now be generated if the internal compiler fields are exceeded. In addition, the fields involved have been expanded such that. if the FORTRAN compiler is compiled with compiler generation option BIGOPTJOB set, larger program units can be compiled.

P1939 FORTRAN - RUN TIME PROBLEMS WITH VERY LARGE PROGRAMS

It was possible for various run-time problems to occur when a program was compiled with more than 2047 variables at lex level 2. The execution of a program with many COMMON blocks and OWN arrays may have resulted in a problem such as an INVALID INDEX or a system loop. This has been corrected.

P2302 FORTRAN - INVALID "PARAMETER MISMATCH"

A PARAMETER MISMATCH occurred when a subroutine, call it SUB1, had an array name as a dummy argument, and it was called with an array element as an actual argument, and the call occurred in a subroutine to which SUB1 was a dummy argument.

Example:

```
    EXTERNAL SUB1
    CALL SUB2 (SUB1)
    END
    SUBROUTINE SUB2 (SUB1)
    REAL A(20)
    CALL SUB2 (A(5))
    END
    SUBROUTINE SUB1 (Y)
    END
```

This problem has been corrected.

P2322 FORTRAN - "FORTRAN XREF"

The following problems, which occurred in the FORTRAN XREF, have been corrected:

Items were erroneously xreffed (e.g., beginning of DO statements,
internal compiler labels, etc.)
Items had missing occurrences
Items had extra occurrences
Items were not xreffed
Items were xreffed with the incorrect type
Labels on DO statements had no "OCCURS AT" clause
References to variables which occurred on the left of assignment
statements for OPT=1 were not marked with asterisks.

P2592 FORTRAN - CORE ESTIMATE WITH "$SET SEPARATE"

The estimate core usage for the first subroutine was used for all subsequent subroutines when compiling with the dollar option SEPARATE set. This problem has been corrected by calculating an individual core estimate for each separate SUBROUTINE.

GUARDFILE

D2494 GUARDFILE - ACCESSCODE

GUARDFILE has been modified so that the type of access (NO, RO, WO, RW, XO) may be determined by usercode, program name, accesscode or any combination of these three items.

Syntax Changes:

```
    <entry type> ::= USERCODE//PROGRAM//ACCESSCODE

    <qualification info> ::= <usercode>//<program name>//<accesscode>
```

For multiple users with the same usercode, it is necessary to provide additional security to the guardfile. This can be accomplished by creating a guardfile with SECURITYTYPE=CONTROLLED and having itself as its guardfile, or having a special guardfile for all guardfiles (including itself) which restricts access to only the creator's usercode and accesscode.

Example

Consider a group of three users (USER1, USER2, USER3) with the same usercode UA. Consider also a fourth user (USER4) with usercode UB. The USERDATAFILE could be set up as follows:

Usercode	Passwords	Accesscodes/Passwords
UA	PA1, PA2, PA3	AA1/AP1, AA2/AP2, AA3/AP3
UB	PB	AB/BP

Each user (person) in usercode UA can be assigned his own usercode password and his accesscode and corresponding accesscode password.

Note: Any valid user under usercode UA could use any of the accesscodes AA1, AA2, AA3 as long as he also knows the corresponding accesscode password AP1, AP2, AP3.

Assume the following:

USER1 is told UA/PA1 and AA1/AP1 only
USER2 is told UA/PA2 and AA2/AP2 only
USER3 is told UA/PA3 and AA3/AP3 only
USER4 is told UB/PB and AB/PB only

Consider four files with the following access requirements:

	Access	Requirements	for	
Filename	USER1	USER2	USER3	USER4
(UA)F1	RW	RW	RW	NO
(UA)F2	RW	RW	RW	RW
(UB)F3	RO	RO	RO	RW
(UA)F4	RW	RO	NO	WO

These files can be set up with the following security:

(UA)F1 SECURITYTYPE=PRIVATE IO

(UA)F2 SECURITYTYPE=PUBLIC IO

(UB)F3 SECURITYTYPE=GUARDED, GUARDFILE:DEFAULT=NO, USERCODE UA = RO

(UA)F4 SECURITYTYPE=CONTROLLED, GUARDFILE: DEFAULT=NO, USERCODE US USING ACCESSCODE AA1=RW, AA2=RO, AA3=NO; USERCODE UB=WO.

The above demonstrates the ability to restrict access to files between different users (people) with the same usercode. Note also that accesscodes could be required in any case (e.g., for F3) to ensure validity of access.

If USER3 obtained USER1's accesscode AA1, he would still need to find the corresponding accesscode password AP1 before obtaining RW access to F4, even though USER3 can sign on under the creating usercode UA.

Assume the "owner" of F4 is USER1 (accesscode AA1), the guardfile for F4 should have its SECURITYTYPE=CONTROLLED and either itself as its guardfile or a self-guarded guardfile containing the following:

```
    DEFAULT=NO;
    USERCODE UA USING ACCESSCODE AA1=RW.
```

Otherwise, any user with usercode UA could change F4's guardfile (and thus gain access to and control of the file).

Note: This self-guarded guardfile allowing access only to usercode UA with accesscode AA1 could serve to guard all of USER1's guardfiles.

Each individual user can change his accesscode's password at any time, using the "APASSWORD" in CANDE or the WFL "ACCESS PASSWORD" statement. The owner of a file (the owner of the guardfile) can change the contents of the guardfile to change who is allowed access and what kind of access. Note that the file owner only knows the accesscode of the potential users, not their accesscode passwords.

HOSTSERVICES

P2403 HOSTSERVICES - CORRECT ATTRIBUTE SETTING

The attribute setting for BOT and EOT status change messages has been corrected.

INPUT-OUTPUT

D2640 IN-OUTPUT - UPDATE "I/O" IMPLEMENTATION CHANGED

Update I/O used to be performed by the MCP logical I/O routines (FIBSTACK) with the assistance of the MCP physical I/O routines (IOEXCEPTION and PACKPARITYRETRY). In the future, the FIBSTACK update disk release routines will handle update I/O all by themselves. In order to retain the efficiency gained via update I/O, the default number of buffers for a file with UPDATEFILE=TRUE is now three instead of two. Having less than three buffers results in FIBSTACK waiting for look-ahead reads to complete.

D2646 IN-OUTPUT - "SERIALNO" ATTRIBUTE VALIDITY CHECK

The SERIALNO attribute may be assigned a REAL value via an ALGOL statement or file declaration; the value may be the following:

1) An integer value from 1 through 999999

2) One to six EBCDIC alphanumeric characters, left-justified with blank fills

3) Normal zero (the null value for SERIALNO).

The following ALGOL and WFL text specify the same SERIALNO values:

ALGOL	WFL	SERIALNO
F.SERIALNO:=123	F(SERIALNO=123)	[000123]
F.SERIALNO:="AB "	F(SERIALNO="AB")	[AB]
F.SERIALNO:="AB"	F(SERIALNO=49602)	[049602]

The third line illustrates an easy mistake, since ALGOL short strings are right-justified with NUL fill and thus have integer values.

These rules are now strictly enforced; any violation will produce an attribute error. It is no longer possible to cause an integer overflow fault in ATTRIBUTEHANDLER by an invalid value.

D2783 IN-OUTPUT - "I/O" ERROR MESSAGES

File open, I/O and close error messages are now generated with the same format, as follows:

```
FILE <title>  OPEN ERROR: <error message>
FILE <title>  I/O  ERROR: <error message>
FILE <title> CLOSE ERROR: <error message>
```

All of the errors are fatal, including the CLOSE errors, which previously were displayed for information only.

Previously undocumented messages related to direct I/O have been replaced, as follows:

Previous Message	New Message
I/O ERROR #17	BUFFER IN USE
I/O ERROR #18	UP-LEVEL EVENT
I/O ERROR #19	SECURITY ERROR

D2785 IN-OUTPUT - "HOSTNAME" FILE ATTRIBUTE

The file attribute HOSTNAME has been implemented. HOSTNAME is a pointer valued attribute used during file assignment to select the host on which the file resides.

HOSTNAME may be set when the file is closed and unassigned.

HOSTNAME may be read at any time. If the file is closed and unassigned and the attribute has never been set, it will return a null (".") value. If the file is assigned, HOSTNAME will return the name of the host at which the file resides.

D2809 IN-OUTPUT - "READ" ON THE "B5500" VS. THE "B6700"

The B7000/B6000 I/O Subsystem Reference Manual (Form No. 5001779), Page 3-42 under "RECORD", should be corrected by adding the following between the second and third lines of the paragraph:

"or if the file was opened with a READ[NO] operation," D2866 IN-OUTPUT - RELATIVE "I/O"

Add the following new field for the STATE attribute on Page 3-51 of the I/O Subsystem Reference Manual (Form No. 5001779):

"[5:1] If this bit is on, a deleted or duplicated record was referenced in file whose FILEORGANIZATION is RELATIVE."

Add the following new attribute to the I/O Subsystem Reference Manual:

"FILEORGANIZATION

Disk only, read/write, anytime/closed, Integer

The FILEORGANIZATION attribute is available so that files may be restricted as to the organization under which they were opened. The attribute must be set to the value used when the file was created in order to open the file, unless the file is direct. The currently defined values, mnemonics, and meanings of FILEORGANIZATON are as follows:

0	NOTRESTRICTED	The file does not adhere to any FILEORGANIZATION restrictions.
1	RELATIVE	The file was created under a RELATIVE file organization and adheres to the restrictions of such. This organization is derived from COBOL74.

The default value is NOTRESTRICTED."

D2883 IN-OUTPUT - COUNT LOGICAL "I/O" REQUESTS

This feature causes requests made on logical I/O read and write procedures to be counted. Each call will cause a count kept in the FIB to be incremented. This count will be logged at file close time, and the results printed by log requests. This feature supercedes recording the current record number in the log; consequently, the log value will now reflect the actual number of requests and not the number of the last record read/written/seeked.

D2902 IN-OUTPUT - "NEWFILE" VS. TAPE LABELS

Tape labels are now written when a tape is opened with NEWFILE=TRUE and MYUSE not specified. Also, if MYUSE is unspecified and NEWFILE is true and the device is an I/O device (TAPE or DISK), then MYUSE is set to OUT.

D2952 IN-OUTPUT - NEW TASK, FILE ATTRIBUTES

Values for the task attributes HISTORYTYPE, HISTORYCAUSE and HISTORYREASON may now be referenced mnemonically, as follows:

HISTORYTYPE:

NORMALV
DUMPINGV
QTEDV
STEDV
DSEDV
NORMALEOTV
SYNTAXERRORV
UNKNOWNEOTV
UNINITIATEDV

HISTORYCAUSE:

OPERATORCAUSEV
PROGRAMCAUSEV
RESOURCECAUSEV
FAULTCAUSEV
SYSTEMCAUSEV
DCERRCAUSEV
DCERRV = DCERRCAUSEV
IOERRCAUSEV
IOERRV = IOERRCAUSEV
SOFTIOERRCAUSEV
SOFTIOERRV = SOFTIOERRCAUSEV
NEWIOERRCAUSEV
NEWIOERRV = NEWIOERRCAUSEV
UNIMPLEMENTEDCAUSEV
UNSPECIFIEDCAUSEV
EBDMSERRCAUSEV
EBDSMERRV = EBDMSERRCAUSEV
NETWORKCAUSEV

HISTORYREASON:
DSed for OPERATORCAUSE:

RSVPV
CLEARUNITV
JUSTDSEDV

DSed for PROGRAMCAUSE:

MISSINGCODEFILENAMEV
MISSINGCODEFILEV
INITACTIVETASKV
NOEXTERNALRUNV
INVALIDACCESSCODEV
DEATHINFAMILYV
CRITICALBLOCKV
BADGOTOV
NOTEXECUTABLEV
UNMATCHEDPARAMSV
INVCOMPILERVV
SECURITYERRORV
LIBMAINTV
BADRESIZEDEALLOCV
MISSINGINTRINSICV
INCOMPATIBLELEVELV
INFANTICIDEV
NOTBOUNDV
ILLEGALOWNARRAYV
DIMSIZERRORV
UPLEVELATTACHV
ILLEGALSWAPV
SWAPDOESNTALLOWV
BADTASKATTRIBUTEV
MISSINGCARDDECKV
BADRESTARTV
BADEVENTUSAGEV
BADGIVELOCKV
BADGETLOCKV
COMPILERSONLYV
TASKLIMITEXCEEDEDV
AXBADARRAYV
RUNTIMEWFLV
COMPILERERRORV
LIBMISSINGNAMEV
LIBTYPEMISMATCHV
LIBDIRECTORYNOTFOUNDV
CYCLICPROVISIONV
PREVIOUSLYFROZENLIBV
LIBIMPLEMENTATIONERRORV
BADPTRLIBV
NONUNIQLIBV
NOINITPARAMV
SWAPJOBCANTBELIBV
GLOBALLIBSONLYV
INVALIDLIBREFV
LIBFEATURENOTIMPLEMENTEDV
BADCOMPILERINDEXV
LIBNOTPROCESSEDORRUNV
LIBMUSTBESEPARATESTACKV

STed for PROGRAMCAUSE:

RESPONSEREQUIREDV

DSed for RESOURCECAUSE:

PROCESSEXCEEDEDV
IOEXCEEDEDV
STACKEXCEEDEDV
PRINTEXCEEDEDV
PUNCHEXCEEDEDV
CARDREADEXCEEDEDV
MEMORYEXCEEDEDV
DIRECTORYEXCEEDEDV
TAPEEXCEEDEDV
WAITEXCEEDEDV
ELAPSEDEXCEEDEDV
DISKLIMITEXCEEDEDV
STRINGPOOLEXCEEDEDV

DSed for FAULTCAUSE:

DIVIDEBYZEROV
EXPOVERFLOWV
EXPUNDERFLOWV
INVALIDINDEXV
INTEGEROVERFLOWV
INACTIVEQV
MEMORYPROTECTV
INVALIDOPV
LOOPV
MEMORYPARITYV
SCANPARITYV
INVALIDADDRESSV
STACKOVERFLOWV
STRINGPROTECTV
PROGRAMMEDOPV
BOTTOMOFSTACKV
SEQUENCEERRORV
INVALIDPCWV
STACKUNDERFLOWV
ZAPPEDV
DISKPARITYV

DSed for SYSTEMCAUSE:

NOMEMV
PARITYONPBITV
ARRAYTOOLARGEV

STed for SYSTEMCAUSE:

WSSIZEEXCEEDEDV

DSed for UNIMPLEMENTEDCAUSE:

DYNAMICOWNARRAYV

DSed for NETWORKCAUSE:

NOHOSTV
HOSTNOTREACHABLEV
HOSTHALTLOADEDV

STed for NETWORKCAUSE:

DISCONNECTEDV
SUSPENDEDV

Several new file attributes have been added to support upgrading to III.1 Libraries and BNA, as follows:

FILEUSE:

A new file attribute, FILEUSE, is now available. This attribute is based on the MYUSE attribute, except as follows:

1. It is restrictive; i.e., if an output operation is attempted on a file whose FILEUSE is declared IN, an error occurs, and vice versa.

2. The default value is IO.

3. The value CLOSED is not available.

4. The value is retained during file operations.

5. It does not cause the creation of a new file when a search for an output file is being conducted; it has no other side effects with respect to file search or creation - it is strictly a usage limitation.

6. It is readable at any time; it is only settable when closed (including close with retention).

7. The error given when FILEUSE is violated is "WRITE ON INPUT FILE" or "READ ON OUTPUT FILE".

DENSITY:

The following mnemonics are now the "preferred" mnemonics for the DENSITY file attribute:

BPI200
BPI556
BPI800
BPI1600
BPI6250

The mnemonic BPI6250 is for the new GCR tape facility.

The old mnemonics for the DENSITY attribute will continue to be supported as synonyms to the above, as follows:

LOW	= BPI200
MEDIUM	= BPI556
HIGH	= BPI800
SUPER	= BPI1600

NEWFILE:

A new file attribute, NEWFILE, has been implemented to specify whether a new file is to be created or an existing file is to be accessed.

Its usage and access specifications are:

General, read/write, anytime/closed, Boolean

The attribute actually has three states: never-set, TRUE and FALSE. If the attribute has never been set, then it has no effect: a new file will be created or a permanent file sought, depending upon the interaction of the MYUSE, AREASIZE and KIND attributes. If NEWFILE is set to TRUE, then a new file will be created irrespective of the settings of other attributes. In the case of DISK and PACK files, a default AREASIZE will be chosen by the MCP if none was specified. If NEWFILE is set FALSE, then an existing file will be sought, regardless of the settings of other attributes. A file-open error results if NEWFILE is set incompatibly with the device type, such as FALSE for a PRINTER or TRUE for a READER.

The NEWFILE attribute may be read at any time: the value is TRUE if NEWFILE has been explicitly set TRUE and FALSE otherwise. If the attribute has not been set, the value is FALSE whether or not a new file was created according to the default criteria.

INPUT-OUTPUT

P1030 IN-OUTPUT - CHANGE "INTNAME"

Setting the INTNAME file attribute via a multiple file attribute assignment will now cause label-equation statements for the new INTNAME to be processed. The following example will now attempt to open a disk file:

Example:

```
?COMPILE X ALGOL GO;
?FILE NEWINT (KIND=DISK);
?DATA
BEGIN
FILE F (KIND=TAPE);
F (MAXRECSIZE=22, INTNAME="NEWINT.");
F.OPEN:=TRUE;
END.
?END
```

P1128 IN-OUTPUT - "CHECKCOUNT" VS. "5500" TAPES

Block count errors on B5500 data tapes will now be reported by CLOSE.

P1136 IN-OUTPUT - TAPE CLOSE VS. "MYUSE"

When closing an output tape, MYUSE will now be left as output.

P1603 IN-OUTPUT - EXTRACT IOCBS FROM BUFFERS

I/O control blocks are now disjoint from logical I/O buffers. They are linked to the buffers, and continue to be displayed by PROGRAMDUMP and DUMPANALYZER.

P1736 IN-OUTPUT - NEGATIVE BOOLEAN

Boolean file attributes will no longer cause an attribute error if set to a negative value.

P1781 IN-OUTPUT - "CANCEL/DIRECTDCREAD"

CANCEL/DIRECTDCREAD will now check to be sure there is an IRW before cancelling the READ.

P1856 IN-OUTPUT - LOG VS. TITLE

Log open and close records will now show the title of the file actually opened, even if the operator IL's a "no file" RSVP. Previously, log open and close records showed the title of the file requested, which was not always the title of the file that was actually used.

P1860 IN-OUTPUT - "FIBEOF"

Sometimes the MCP would give EOF when user programs attempted to read from a small disk file. This has been corrected.

P1891 IN-OUTPUT - BADLY BLOCKED UPDATE "I/O"

Certain combinations of update I/Os, when applied to a badly blocked file, could result in the data being incorrectly written. This has been corrected.

P9024 IN-OUTPUT - "FIBLOCK" AND "TIMESTAMP"

1. Binary disk I/O will no longer hang.

2. READ(F,*,A) will no longer change the disk file TIMESTAMP.

P9167 IN-OUTPUT - "NORESOURCEWAIT" FILE ATTRIBUTE CORRECTION

The NORESOURCEWAIT direct file attribute may now be reset as well as set. Formerly, the following ALGOL statement would set the attribute (the value of the Boolean expression was ignored):

```
F.NORESOURCEWAIT:=FALSE
```

P9228 IN-OUTPUT - "AVAILABLE"

<FILE>.AVAILABLE will now return a value of 2 (no file) in cases where the file cannot be opened because of incompatible blocksizes, etc.

P9272 IN-OUTPUT - "FIBLOCKSNR"

A COBOL program can reference file attributes when executing in the "USE AFTER STANDARD LABEL" declarative procedure, even if the FIB is IPC capable. (Formerly, the program would hang in the user's label procedure when an IPC-capable FIB was accessed.)

P9273 IN-OUTPUT - "PROGRAMDUMP," NEW "IOCB"

The output of the file section of PROGRAMDUMP has been modified to correctly identify the new words in the IOCB.

Tape labels now contain four-digit system serial numbers. The USYSID field in the HDR2.EOF2 label now begins at character number 45 (starting at character one). The old three-digit field, beginning at character 37, will contain the lower three digits of the system serial number until the III.1 release, at which time the three-digit field will change to blanks. The tape level value (characters 35 and 36 of the VOL1 label) has been changed to 3.

P9274 IN-OUTPUT - "CLOSE HERE" ON EMPTY TAPE FILE

A COBOL CLOSE HERE statement executed upon an empty tape file will no longer either cause a block count error or an incorrect number of records being written on the tape.

P9275 IN-OUTPUT - "EVEN" PARITY VS. "EOF1"

CLOSE was unable to recognize EOF labels written with EVEN parity causing extraneous "I/O ERR IN CLOSE" messages. This problem has been corrected.

INTERACTIVEXREF

D2476 IXREF - "INTERACTIVEXREF" ENHANCEMENTS

Text lines from the symbol file used by INTERACTIVEXREF may now be sent to a disk file. The revised syntax and semantics of the commands affected by this option are explained below.

<identifier specification>

```
--<identifier>-----------------------------------|
                |                               |
                |-<identifier qualification>-|
```

<identifier qualification>

```
---- AT FIRST ----------------------------------------------|
   |                                                      |
   |- AT --<sequence number>-------------------------------|
   |                                                      |
   |- IN --<procedure specification>-----------------------|
   |                                                      |
   |- OF --<procedure specification>-----------------------|
   |                                                      |
   |- AT --<sequence number>-- IN --<procedure specification>-|
```

<procedure specification>

```
   |<---------- OF ---------|
   |                        |
----<procedure identifier>----|
```

<range spec>

```
---------- * ----------------------|
   |     |                        |
   |- - -|                        |
   |                              |
   |-------------<sub range spec>-|
   |           |
   |------- * -|
   |     |
   |- - -|
```

<sub range spec>

```
     |<------------------------ , -------------------------|
     |                                                     |
---------------<procedure spec>-----------------------------------|
   |    |     |                 |                          |      |
   |    |- - -|                 |- THRU <procedure spec> -|      |
   |    |                                                  |      |
   |    |------- ( <procedure spec> ) ---------------------|      |
   |    |     |                                                   |
   |    |- - -|                                                   |
   |                                                                |
   |  |<---------------- , ---------------|                         |
   |  |                                   |                         |
   |---<sequence no>-------------------------------------------------|
                      |                 |
                      |- - <sequence no> -|
                      |- - END -----------|
```

REFERENCES

```
-- REFERENCES -----------------------------------------------------|
   ---         |                                                  |
               | |<-------------------------------------------| |
               | |                                            | |
               |-----/1\-<identifier specification>---------------|
                    |                                            |
                    |-/1\- : -- RANGE --<range spec>-----------|
                    |           --                             |
                    |-/1\- : -- CHANGED -----------------------|
                    |           --                             |
                    |-/1\- : -- ALIASES -----------------------|
                    |           -                              |
                    |-/1\- : -- TEXT --------------------------|
                    |           -      |                       |
                    |                  |-<integer>-------------|
                    |                                          |
                    |-/1\--- : -- ENVIRONMENTS ----------------|
                    |     |       ---          |               |
                    |     |                    |- ONLY -------|
                    |     |                                    |
                    |     |- : -- GLOBALENVIRONMENTS ----------|
                    |             ----             |           |
                    |                              |- ONLY -|
                    |                                          |
                    |-/1\- : -- PRINTER -----------------------|
                    |           -                              |
                    |-/1\- : -- REMOTE ------------------------|
                    |                                          |
                    |-/1\- : -- FILE --<file name>-------------|
```

SEMANTICS:

References to the specified identifier will be listed. If no identifier is specified, and the previous command was **LOCATE**, **REFERENCES**, **EXPAND** or **SUMMARY**, then the identifier specified in that command will be used. This identifier, which is remembered from command to command, is known as the "work identifier".

Unless modified by some of the options described below, the references will be printed in the form familiar from printed **XREF**s. The eight digit sequence number of each line where the identifier is referenced will be printed. It will be preceeded by an asterisk (*) if the value might be changed by the statement, and followed by a pound sign (#) if the reference occurred as part of an expanded define. The available options are:

RANGE — allows the user to restrict the range over which references are to be printed. If this option is not specified, then the default reference range, as specified by the **RANGE** command, will be used. The default value of the default reference range is the entire program.

CHANGED — only those references where the value of the identifier might be changed by the statement will be listed.

ALIASES — causes a merged list of references to the identifier and all of its aliases (if any) to be listed. Those sequence numbers where an alias is referenced will be marked with "+". Currently, only ESPOL keeps track of aliases.

TEXT — causes the text from the symbol file to be printed with each reference. If an integer is specified with this option, a sample of that many lines of text, centered at the line containing the reference, will be printed with each reference. Note that the symbol file must be loaded to use this option.

ENVIRONMENTS — causes the names of the environments (procedures and blocks) where the references occur to be printed, appropriately interleaved with the references. If modified by "ONLY", then only the environments, and not the references, will be printed. Note that **ENVIRONMENTS ONLY** and **TEXT** are mutually exclusive.

GLOBALENVIRONMENTS — similar to **ENVIRONMENTS**, except that references are broken down only by global environment (global procedure). Note that **GLOBALENVIRONMENTS ONLY** and **TEXT** are mutually exclusive.

PRINTER — causes output to go to the line printer, by way of a file internally named "LINE".

REMOTE — for use when one wishes output to go to both the line printer and the terminal.

FILE — causes all referenced text lines from the symbol file to be output to the user specified disk file; this file cannot already exist.

It will be made with the same filetype as the file loaded by the SYMBOL command. If no symbol file is loaded, an error message is output to the terminal.

MERGE/COINCIDENCE

```
                      |<-------------------------------------------|
                      |                                            |
                      |          |<------------ , ------------|    |
                      |          |                            |    |
---- MERGE -------------/1*\---<identifier specification>-------------|
  |  ---            |  |                                          |
  |- COINCIDENCE -| |-/1\- : -- RANGE --<range spec>-----------|
     -----            |              --                           |
                      |-/1\- : -- CHANGED -----------------------|
                      |              --                           |
                      |-/1\- : -- ALIASES -----------------------|
                      |              -                            |
                      |-/1\- : -- TEXT --------------------------|
                      |              -       |                    |
                      |                      |-<integer>--------------|
                      |                                           |
                      |-/1\--- : -- ENVIRONMENTS ----------------|
                      |      |         ---              |         |
                      |      |                          |- ONLY -------|
                      |      |                                    |
                      |      |- : -- GLOBALENVIRONMENTS ----------|
                      |                ----                  |    |
                      |                                      |- ONLY -|
                      |                                           |
                      |-/1\- : -- PRINTER -----------------------|
                      |              -                            |
                      |-/1\- : -- REMOTE ------------------------|
                      |                                           |
                      |-/1\- : -- FILE --<file name>-------------|
                                     -
```

SEMANTICS:

A merged list of the references to the specified identifiers will be produced. To avoid confusion, The "work identifier" will be nullified. All of the options described under the command REFERENCES apply.

DECLARATIONS

```
-- DECLARATIONS --------------------------------------%
   ---             |                              |
                   |-<identifier>-----------------|
                                  |               |
                                  |- -<identifier> -|
                                  |               |
                                  |- : -- LITERAL --|
                                          ---
```

```
--------------------------------------------------------------------|
  |                                                                 |
  | |<------------------------------------------------------------| |
  | |                                                             | |
  |-----/1\- : -- DECLARED --<range spec>---------------------------|
      |               ---                                         |
      |-/1\--- : -- USED -----------------------------------------|
      |         |                                                 |
      |         |- : -- UNUSED -| |-<range spec>------------------|
      |                                                           |
      |-/1\- : -- ONLYUSED --<range spec>-------------------------|
      |                                                           |
      |                    |<-------------------|                 |
      |                    |                    |                 |
      |- : -- CLASS ---<alpha identifier>-------------------------|
      |                                                           |
      |- : -- KEYWORD --<alpha identifier>------------------------|
      |                                                           |
      |-/1\- : -- LEVELS --<hex int>------------------------------|
      |           -----                   |                       |
      |                                   |- - <hex int> ---------|
      |                                                           |
      |-/1\- : -- DISPLACEMENTS --<hex int>-----------------------|
      |           ----                          |                 |
      |                                         |- - <hex int> ---|
      |                                                           |
      |-/1\- : -- IDSONLY ----------------------------------------|
      |           ---        |                                    |
      |                      |-<integer>--------------------------|
      |                                                           |
      |-/1\- : --<references>-------------------------------------|
      |                                                           |
      |-/1\- : --<expand>-----------------------------------------|
      |                                                           |
      |-/1\- : -- SUMMARY ----------------------------------------|
      |                                                           |
      |-/1\- : -- SHORT ------------------------------------------|
      |                                                           |
      |-/1\- : --<sort>-------------------------------------------|
      |                                                           |
      |-/1\- : -- PRINTER ----------------------------------------|
      |           -                                               |
      |-/1\- : -- REMOTE -----------------------------------------|
      |                                                           |
      |-/1\- : -- FILE --<file name>------------------------------|
                  -
```

```
<references>

-- REFERENCES ------------------------------------------------------|
   ---          |                                                   |
                | |<---------------------------------------------|  |
                | |                                              |  |
                |-----/1\- . -- CHANGED ----------------------------|
                    |            --                              |
                    |-/1\- . -- ALIASES -------------------------|
                    |            -                               |
                    |-/1\- . -- TEXT ----------------------------|
                    |            -         |                     |
                    |                      |-<integer>-----------|
                    |                                            |
                    |-/1\--- . -- ENVIRONMENTS ------------------|
                    |      |      ---                |           |
                    |      |                         |- ONLY ----|
                    |      |                                     |
                    |      |- . -- GLOBALENVIRONMENTS -----------|
                    |              ----                     |    |
                    |                                       |- ONLY -|
                    |                                            |
                    |-/1\- . -- RANGE --<range spec>-------------|
                                 --
```

<expand>

```
-- EXPAND ----------------------------------|
-            |                            |
             | |<----------------------| |
             | |                       | |
             |-----/1\- . -- FULL --------|
                  |-/1\- . -- BLOCKED -|
                  |           -        |
                  |-/1\- . --<integer>-|
```

<sort>

```
         |<-----------------------|
         |                        |
-- SORT -----/1\- SEQNUMBER ----------|
            |     ---             |
            |-/1\- ADDRESSCOUPLE -|
            |      -------        |
            |-/1\- ALPHABETICAL --|
                   -----
```

SEMANTICS:

A specified set of declarations is isolated and listed along with optional information. The options available fall into three categories: those which select the set of declarations; those which specify the output to be produced for each selected declaration; and those which specify the order and destination of the output. To avoid confusion, the "work identifier" will be nullified.

The defaults, if no options are specified, are as follows: all declarations are included; the header line (name, environment where declared, compiler class, sequence number where declared, aliases, et cetera) will be printed for each declaration; the output will be ordered alphabetically by identifier name, and will come to the terminal.

Options which control the selection of the set of declarations are:

<identifier> restricts the set to the occurrences of a given indentifer.

<identifier>-<identifier> restricts the set to a given alphabetic range. The identifier pair must be ordered alphabetically.

<identifier> :LITERAL restricts the set to those identifiers which contain this specified identifier as a substring.

DECLARED restricts the set to those identifiers declared within the specified range.

USED if no range specification is included, restricts the set to those identifiers referenced somewhere in the program. If a range is specified, restricts the set to those referenced within the range.

UNUSED if no range specification is included, restricts the set to those identifiers which are declared but never referenced; otherwise, restricts the set to those which are not referenced within the specified range.

ONLYUSED restricts the set to those identifiers which are only used within the specific range and not referenced elsewhere.

CLASS restricts the set to a particular compiler class or group of compiler classes. The compiler class must appear exactly as it does in a header line, e.g., BOOLEAN ARRAY, INTEGER, FORMAL NAME REAL, et cetera. Only one compiler class may be specified for each CLASS option, though the compiler class may contain more than one <alpha identifier>; e.g., REAL PROCEDURE. Note from the syntax diagram that CLASS may be specified as often as desired, thus specifying a group of classes.

CLASS - restricts the set to all compiler classes except those specified in the alpha identifier list.

KEYWORD restricts the set to a group of compiler classes which contain the specified alpha identifier. For example, KEY BOOLEAN would cause BOOLEAN, BOOLEAN ARRAY, BOOLEAN PROCEDURE, et cetera, to be included. KEYWORD and CLASS may be specified as often as desired to generate the desired group of classes.

KEYWORD - selects a group of compiler classes which do not contain the specified alpha identifier. For example, KEY- BOOLEAN would include exactly the complement of the classes included by KEY BOOLEAN.

LEVELS restricts the set to those identifiers which have stack cells with lex levels as specified.

DISPLACEMENTS restricts the set to those identifiers which have stack cells with displacements as specified.

Options which specify the output to be produced for each selected declaration are:

IDSONLY displays only identifier names and not any of the other header information. The output is displayed in ascending alphanumeric order, with a default field width of 20. The field width may be altered by specifying an optional field width.

REFERENCES references to the selected declaration will be listed. This option may itself be modified by any of the options listed under the REFERENCE command, except PRINTER or REMOTE, with the same effects.

EXPAND the text of the selected declaration will be written out (expanded) if the identifier is an item such as a define, array or file, which has text associated with its declaration. This option may itself be modified by the options FULL and BLOCKED, as described under the command EXPAND. Note that only the first or the final expansion may be obtained. (If a full expansion of a define is requested, it will be done in the context of its first use.) EXPAND may be modified by an integer, which allows specification of an approximate limit on lines of text printed for each declaration. The default is ten.

SUMMARY a summary of the number and kinds of references to the selected declaration will be printed.

SHORT the aliases of the selected declaration will not be printed. Currently, only ESPOL keeps track of aliases; for other languages, SHORT has no effect.

Options which specify the order and destination of output are:

SORT controls the order in which the selected declarations will be printed. When SORT is followed by SEQNUMBER, the output will show the declarations sorted on sequence number where declared. When SORT is followed by ADDRESSCOUPLE, the output will show the declarations sorted first on address couple (lex level and displacement of stack cell), then on sequence number where declared. When SORT is followed by ALPHABETICAL, the output will show the declarations sorted first alphabetically, then in order of occurrence. (This is the same output that would be produced if SORT were not specified). When SORT is followed by more than one item, then multiple sets of output will be produced.

PRINTER causes output to go to the line printer, by way of a file internally named "LINE".

REMOTE causes output to come to the terminal, even if it is also going to the line printer.

FILE the FILE option is only valid when used with the REFERENCE option. Its semantics are explained with the REFERENCE command.

LIST

```
-- LIST -------------------------------------------------------------->
   -
>---------------------------------------------------------------------|
  |                                                             |
  | |<--------------------------------------------------------| |
  | |                                                         | |
  | |        |<-------------------- , --------------------|   | |
  | |        |                                            |   | |
  |-----/1\-----<seq no>----------------------------------------|
      |         |         |           | |               |   |
      |         |         |-<integer>-| |- - --<seq no>-|   |
      |         |                       |               |   |
      |         |                       |- - -- END ----|   |
      |         |                                       |   |
      |         |-<procedure id>------------------------|   |
      |         |                                       |   |
      |         |-<interface id>------------------------|   |
      |                                                     |
      |-/1\- : -- PRINTER ----------------------------------|
      |           -                                         |
      |-/1\- : -- REMOTE -----------------------------------|
      |                                                     |
      |-/1\- : -- FILE --<file name>------------------------|
                  -
```

SEMANTICS:

Text is listed from the symbol file. If no sequence numbers are specified, the entire file is listed. If sequence number(s) and/or sequence number pair(s) are specified, only those lines will be listed. When the sequence number, or first sequence number of a pair, is followed by "<" and an integer, the command will back up that many records before beginning to list. Entire procedures and interfaces may be listed by using the procedure or interface identifiers in place of the sequence range.

The available options are:

PRINTER causes output to go to the line printer, by way of a file internally named "LINE".

REMOTE causes output to come to the terminal, even if it is also going to the line printer.

FILE causes all lines in the sequence range or all lines of the requested procedures or interfaces to be written to the user requested disk file; this file cannot already exist. It will be made with the same filetype as the file loaded by the SYMBOL command.

D2523 IXREF - IMPROVE QUALFICATION

Previously, <procedure specification> had to be specified all the way to a global procedure; i.e., "HARRY OF JOE" was an acceptable specification if JOE was a global procedure and HARRY was a procedure declared within JOE. HARRY by itself, however, was only acceptable if there was a global procedure HARRY.

The <procedure specification> now only need be long enough so that its outermost environment is the "best candidate" of the possible environments specified by that identifier. The "best candidate" is defined as follows:

1. If there exists only one environment (i.e., MODULE or procedure) with this name, use it.

2. If more than one environment exist for this name, limit the possible candidates to the most global.

3. If there are equally global environments for this name, use the first.

If more than one environment exist for a specified name, a warning of possible ambiguity and the chosen environment are output.

Example:

```
JOE
     HARRY
HARRY
FRANK
     TOM
          STEVE
     BOB
          STEVE
               HARRY
```

The environments are the following:

```
JOE
HARRY OF JOE
HARRY
FRANK
TOM OF FRANK
STEVE OF TOM OF FRANK
BOB OF FRANK
STEVE OF BOB OF FRANK
HARRY OF STEVE OF BOB OF FRANK
```

"TOM" would locate "TOM OF FRANK". TOM is a unique environment.

"STEVE" would locate "STEVE OF TOM OF FRANK". Since both STEVE's are at the same level, the first is used and a warning is emitted.

"STEVE OF BOB" would locate "STEVE OF BOB OF FRANK". BOB is a unique environment.

"HARRY" would locate "HARRY"; i.e., it is the most global, and a warning would be emitted.

"HARRY OF STEVE" would be an error. The environment used for STEVE is "STEVE OF TOM OF FRANK" (i.e., the first of the STEVE's on the same level) and there is no HARRY in that environment.

"HARRY OF STEVE OF BOB" would locate "HARRY OF STEVE OF BOB OF FRANK". BOB is a unique environment.

"HARRY OF JOE" would locate "HARRY OF JOE". JOE is unique.

D2524 IXREF - ENVIRONMENT IMPROVEMENT

The ENVIRONMENTS option to the REFERENCE, MERGE, COINCIDENCE and DECLARATION commands has been modified so that any level of environments can be listed by specifying the level. Previously, only global environments (through the GLOBALENVIRONMENTS option), or the complete list of environments (through the ENVIRONMENTS option) could be obtained. The new syntax for the ENVIRONMENTS option is the following:

```
-- ENVIRONMENTS ------------------------------------------------|
                 |            |  |        |
                 |-<number>-|  |- ONLY -|
```

<number> specifies the maximum number of environments to be listed.

D2555 IXREF - RECOGNIZE LOWER CASE INPUT

INTERACTIVEXREF now translates to upper case all lower case characters in a line of input read from a remote terminal.

D2823 IXREF - "*BREAK*"

Previously, whenever INTERACTIVEXREF was waiting for a null input as the signal to transmit the next page of output, any nonblank string was discarded and a "*BREAK*" response was given. INTERACTIVEXREF will now process any nonblank "break string". If the "break string" is not a recognized command, the normal "*BREAK*" is given; otherwise, the command will be processed.

INTERACTIVEXREF

P1652 IXREF - FILES FROM "NEWP" COMPILER

INTERACTIVEXREF files generated by the NEWP compiler are now accepted and used.

JOB FORMATTER

D2637 JOBFORMAT - REASON FOR QUEUE "DS"

The system will now display the reason for queue insertion failure.

This message is passed to the MCS if ORGWANTSMSG is TRUE. The message has the following format:

Word 0	.[47:8]	= 21 Controller result
	.[39:8]	= 6 Miscellaneous message
	.[31:8]	= 3 Queue reject notice
	.[14:15]	= LSN
Word 1		= Length of message in characters
Words 2-N		= Text

D2967 JOBFORMAT - LOG BOXES FOR JOBS

The MCP now maintains a list of "boxes" that a task/job has occupied during the life of its execution, as well as a list of "boxes" that the task/job could have occupied. These lists are logged in the EOT and EOJ log records in the "EOTBOXNOX" word. EOTBOXNOX contains four 8-bit fields which are the following BOX lists: DATA ALLOWED, CODE ALLOWED, DATA OCCUPIED and CODE OCCUPIED.

JOBFORMATTER (and LOGANALYZER) will now display two new lines in the EOT/EOJ record print out, as follows:

```
DATA ALLOWED IN XXX; CODE ALLOWED IN XXX
DATA OCCUPIED XXX; CODE OCCUPIED XXX
```

where XXX is a list of BOX names (GLOBAL, LOCAL1, LOCAL2, etc.) or "NO BOX" (which is used when the "box" list is empty).

JOB FORMATTER

P1074 JOBFORMAT - "JOBFORMATTER" LOOP

It was possible for JOBFORMATTER to loop while printing the I/O error summary of a job. This problem has been corrected.

P1123 JOBFORMAT - "SEG ARRAY" PRINTING "MCS" STRING

JOBFORMATTER will no longer be DS'ed with a SEG ARRAY error when trying to print a long (more than 80 characters) MCS MESSAGE RECORD.

P1709 JOBFORMAT - INCLUDES "LSN" FOR JOB SUMMARY

JOBFORMATTER will now only include the LSN to the block character printout of a job summary if the BCLSNF bit is set in the printmask.

P2271 JOBFORMAT - ADD CASE "22, LP"

Case 22 has been added to JOBFORMATTER to accommodate the LP command.

P2272 JOBFORMAT - "SCR" JOBS GENERATE "JOBFORMATTER" FAULT

JOBFORMATTER will now handle "JOBLOGS" that do not have a BOJ entry. This was causing an INVALID INDEX fault in JOBFORMAT on systems with OPEN set when trying to print the block character heading of an SCR job.

P2323 JOBFORMAT - NOT SKIPPING BAD RECORDS

JOBFORMATTER will no longer, under certain circumstances, allow bad records to slip through.

P2324 JOBFORMAT - PREVENT "INTEGER OVERFLOW"

When JOBFORMATTER is converting a very large data memory integral to display form, an INTEGER OVERFLOW will no longer occur.

P2382 JOBFORMAT - "CLOSE" TYPES

JOBFORMATTER (and LOGANALYZER) will now display the correct CLOSE types.

P2383 JOBFORMAT - "JOB ENTERED SYSTEM" TIME

The "JOB ENTERED SYSTEM" message will now only be printed for WFL jobs, not tasks that look like jobs (TASKNUMBER=JOBNUMBER) or tasks run by the ALGOL compiler (XREFANALYZER). Also, the values and descriptions used by JOBFORMATTER for TASK TYPE now match those used by the MCP.

LCOBOL

P2005 LCOBOL - "TU" FIRMWARE BUFFER LENGTHS

The word in the codefile which contains the lengths of the TU Firmware buffers was incorrect in some cases. This has been corrected.

P2245 LCOBOL - "IF" STATEMENT WITH "OPTIM" CORRECTION

Incorrect code was being generated in some cases for the IF statement when the dollar option OPTIM was set. This problem has been corrected.

P2246 LCOBOL - "DISPLAY" STATEMENT CORRECTION

Incorrect code was being generated in some cases for the DISPLAY statement. This problem has been corrected.

P2463 LCOBOL - INCORRECT CODE, INVALID SYNTAX ERROR

1. It was possible to get an invalid syntax error "DUPLICATE LABEL" for labels with similar names or for no error to be generated for a statement such as: "GO TO <label>.", where the label did not occur in the program.

2. It was also possible for incorrect code to be generated or for the compiler to loop following a statement of the form:

```
WRITE <record> BEFORE <integer> LINE>
```

3. The SRR instruction was sometimes generated as three bytes long instead of two bytes.

These problems have been corrected.

LOADER

D2572 LOADER - DISALLOW "PRINTERLABELS" OPTION

The PRINTERLABELS option (option 24) has been removed from the MCP and all related software. "PRINTERLABELS" is no longer a valid option name for SET and RESET statements.

LOADER

P1078 LOADER - THREE-MULTIPLEXOR SYSTEM

LOADER will now properly handle 3-multiplexor systems. Previously, an attempt to do I/O via Multiplexor C would abort the load.

P1155 LOADER - "LOADER" VS. "IV"

The LOADER IV routine will now relocate sectors in error.

P1178 LOADER - BINARY DISK ADDRESSING

The LOADER can now handle diskpacks that need binary addresses.

P1388 LOADER - CHANGE "ODT," PANEL DISPLAYS

The ODT BOJ message has been changed to indicate the appropriate B6700/B6800 system.

The processor panel display has been changed to show the correct LOADER level.

P1770 LOADER - "OLAYROW" GREATER THAN "1200"

LOADER can now handle OLAYROW sizes greater than 1200 segments. The maximum row size depends on the amount of disk available for this function.

P1818 LOADER - "LOADER" VS. "GMM"

The LOADER now may run with global memory on line. The LOADER will run entirely in local memory and will neither interfere with other systems nor be bothered by other systems. Care should be taken not to CM to the LOADER with global memory on line.

P1852 LOADER - "TD850 ODT"

The LOADER will now work with TD850 ODTs.

P2006 LOADER - UNIT DISPLAY

The LOADER will no longer fault if more than 43 disks or packs are on line.

P2194 LOADER - "BX387" FIRMWARE LOAD

SYSTEM/LOADER can now load diskpack FIRMWARE files. The format of the command is the following:

```
LH PK<nnn> MPX<m> PATH<p> <filename> FROM <tapename>
```

P2577 LOADER - "B9246" DRUM PRINTER IN DUMP

The memory dump of LOADER has been corrected to handle B9246 drum printers.

LOG ANALYZER

D2647 LOGANALY - "LOG IOCONDITION" RESULT DESCRIPTORS

A new maintenance log entry subtype=9 has been added to log I/O results that indicate retries that were not errors (diskpack read and write 4"C801" and read 4"0C01").

These entries will be printed in response to a LOG MAINT or a LOG IOCONDITION request; they will not be printed for a LOG IOERROR request.

These diskpack result descriptors are accepted as successful I/Os and are not retried.

D2841 LOGANALY - "MAJOR TYPE=2 MINOR TYPE=5" VS. BAD LOG TYPE

LOGANALYZER will no longer consider MAJOR TYPE=2 MINOR TYPE=5 log records as being bad. These records are output from several SCR maintenance routines and have no meaning for LOGANALYZER. The RADAR software item is designed to use these records.

D2842 LOGANALY - "ABORT, ERRORS" VS. OTHER OPTIONS

The ABORT, ERRORS and DATE options cannot be used with any of the other LOGANALYZER options due to conflicts in the manner in which the data is manipulated. The following error message will result if one of these options is used with any other option:

"DATE, ABORT, ERRORS CANNOT BE USED WITH ANY OTHER OPTION"

The following syntax diagram should replace the diagram in the System Sofware Operational Guide, Vol 2 (Form No. 5001688), Page 6-1-1.

```
-- LOG ---------------------------------------------------------------->
          |                   | |                 |
          |- / <INTEGER> -----| |-<time>----------|
          |                   |           |       |
          |- " --<logname>-- " -|         |-<date>-|

>---------------------------------------------------------------------|
   |                  | |                      |
   |- TO --<time>-----| |- ABORT --------------|
              |       | |                      |
              |-<date>-| |- ERRORS ------------|
                        |                      |
                        |- DATE ---------------|
                        |                      |
                        |-<loganalyzer options>-|


<loganalyzer options>

  |<---------------------------------------------------|
  |                                                    |
------ ALL ------------------------------------------------|
     |                                               |
     |- BOJ -----------------------------------------|
     |                                               |
     |- BOT -----------------------------------------|
     |                                               |
     |- EOT -----------------------------------------|
     |                                               |
     |- EOJ -----------------------------------------|
     |                                               |
     |- IO ------------------------------------------|
     |                                               |
     |- MSG -----------------------------------------|
     |                                               |
     |- MCS -----------------------------------------|
     |                                               |
     |-/1\--- JOB --------- " --<filename>----------- " -
     |        |           | |                |      |
     |        |- SESSION -| |                |- / = -|
     |        |           | |                        |
     |        |- TASK ----| |-<integer>--------------|
     |                                 |             |
     |                                 |- THRU <integer> --
     |                                               |
     |-/1\- MIX -------------------------------------|
     |            |                                  |
     |            |-<integer>------------------------|
     |                         |                     |
     |                         |- THRU <integer> ----|
     |                                               |
     |- HL ------------------------------------------|
|                                                   |
|- DCP ---------------------------------------------|
|                                                   |
|                    |<------------------------|    |
|                    |                         |    |
|- MAINT -------------------------------------------|
|- IOERROR -----|     |- DK -----|  |-<unitno>-|    |
|- IOCONDITION -|     |- SC -----|                  |
|                     |- PR -----|                  |
|                     |- PP -----|                  |
|                     |- LP -----|                  |
|                     |- CR -----|                  |
|                     |- CP -----|                  |
|                     |- PK -----|                  |
|                     |- MT -----|                  |
|                     |- PEMT ---|                  |
|                     |- NRZMT --|                  |
|                     |- NRZ7MT -|                  |
|                     |- NRZ9MT -|                  |
|                     |- CPU ----|                  |
|- RAW ---------------------------------------------|
|- FILE --------------------------------------------|
|- UNKNOWN -----------------------------------------|
|- OPERATOR ----------------------------------------|
|               |                                   |
|               | |<-------|                        |
|               | |        |                        |
|               |---<type>--------------------------|
|- DUMP --------------------------------------------|
|- SECURITY ----------------------------------------|
|- COMMENT -----------------------------------------|
|- SORTSIZE -- = --<integer>------------------------|
|- UNSORTED ----------------------------------------|
|- CPUERROR ----------------------------------------|
```

Also, change the seventh example on Page 6-1-2 to read as follows:

"LOG 1200 5/11/76 TO 1700 5/11/76 MIX 345"

LOG ANALYZER

P1370 LOGANALY - ACCESSCODE

The log file will store the accesscode (without password) in the following log records:

BOT/BOJ	(item #11)
EOT/EOJ	(item #25)
FILE OPEN	(item # 8)
BEGINNING OF MCS SESSION	(item #11)
END OF MCS SESSION	(item #25)
SECURITY VIOLATION	(item # 7)

LOGANALYZER will include the accesscode whenever it is not null when analyzing any of the log records listed above. LOGANALYZER may be used to obtain information concerning accesscodes using CONTROLLED files from the FILE OPEN records.

P1812 LOGANALY - SYSTEM SERIAL NUMBER IS * DIGITS

As part of the effort to expand system serial numbers, LOGANALYZER now constructs file names with system serial number length of * digits.

P1859 LOGANALY - "DT"

LOGANALYZER will now analyze DISKETTE I/O error retries found in log.

P1940 LOGANALY - FIRMWARE "ID" IN "LH ODT" COMMAND

LOGANALYZER will now display the <firmware id> of the LH ODT command (if one was supplied).

P2189 LOGANALY - CARDNUMBER VS. "P3" WORD

On a B6800 system, LOGANALYZER will now only display the "CARD NO:" for PROM CARD PARITY and RAM CARD PARITY errors and not take a fault on the other error conditions.

P2193 LOGANALY - HALT/LOAD RECORDS

LOGANALYZER will now display the system serial number correctly for HALT/LOAD log entries. All four digits of the serial number will be displayed for system with 4-digit serial numbers.

P2262 LOGANALY - GARBAGE OUTPUT ON "UNSORTED"

LOGANALYZER occasionally produced extra or garbage outputs when the UNSORTED option was used. This problem has been corrected.

P2378 LOGANALY - CHARACTER COUNT ON LOG TITLE

LOGANALYZER will now handle quoted log titles correctly.

P2586 LOGANALY - FAMILY SUBSTITUTION VS. "READK(LOCHDR)"

LOGANALYZER will now be able to find SYSTEM/SUMLOG on DISK when run with the following FAMILY statement:

```
FAMILY DISK=PACK OTHERWISE ANOTHERPACK.
```

LOGGER

D2502 LOGGER - ACCESSCODE

The log file will now store the accesscode (without password) in the following log records:

BOT/BOJ	(item #11)
EOT/EOJ	(item #25)
FILE OPEN	(item # 8)
BEGINNING OF MCS SESSION	(item #11)
END OF MCS SESSION	(item #25)
SECURITY VIOLATION	(item # 7)

LOGGER will allow "ACCESSCODE" as a data type for record selection or reporting. The accesscode is stored as a string item in the JOBSUMMARY file created by LOGGER, and may be used as any other string item by LOGGER. It is not stored in the FILEIODATA file since this is generated using only the FILE CLOSE records, which do not contain the accesscode (the accesscode of a task may be changed between file open and file close time).

D2843 LOGGER - SUMMARY REPORTS

LOGGER will now produce correct summary reports when more than one item is specified in a BREAK statement.

Example:

```
    BREAK ON UNITNO, KIND TOTALING IOTIME
```

The totals for the BREAK items in the detailed reports will also be correct and not totaled under the last BREAK item.

D2844 LOGGER - LAST "STATISTICS" INTERVAL SAVED

LOGGER would not write the last statistics interval to the STATISTICS file at EOF of the last input log file. This problem has been corrected.

LOGGER

P1813 LOGGER - ITEMS EXTENDING PAST COLUMN "72"

LOGGER will now give a syntax error for input items extending past column 72 of the input record.

P2263 LOGGER - ELAPSED TIME FOR RESTART "MCS" SESSIONS

LOGGER will now compute the ELAPSEDTIME value for RESTART MCS sessions from the LOGON and LOGOFF log records instead of using the ELAPSEDTIME value in the LOGOFF record. This will correct several cases of incorrect ELAPSEDTIME logging.

P2264 LOGGER - ELAPSED TIME OF JOBS RUN THROUGH MIDNIGHT

LOGON will now correctly compute the ELAPSEDTIME of jobs that run through midnight.

P2384 LOGGER - "YTDFILE" UPDATE

LOGGER will now allow the "YEAR TO DATE" file to run longer than one calendar year. The date stored in each of the YTDFILE records is now in Julian form (YYDDD). All old YTDFILEs are still compatible with this version of LOGGER.

P2556 LOGGER - "ALINK" VS. "MATCHM" "(BOJ,EOJ)"

The size of the linked list (ALINK) used by the procedure MATCHM to store BOJ and EOJ record pairs has been increased to handle very large log files with jobs that run for a very long time.

LTTABLEGEN

P1124 LTTABLEGEN - USER SPECIFIED TABLES

SYSTEM/TRAINTABLES will now be properly modified to reflect user specified train tables.

P1130 LTTABLEGEN - "450/750 LPM" TRAIN PRINTERS

450/750 LPM train printers will now have appropriate train tables loaded for the following trains:

16 char	EBCDIC-3	-TRAINID=1
64 char	EBCDIC-3	-TRAINID=5
64 char	SWEDEN OCR-B	-TRAINID=14

Note: Although the appropriate tables are loaded for TRAINID's 1 and 5, they will show respectively as EBCDIC18 and EBCDIC72 instead of EBCDIC16 and EBCDIC64. This discrepancy will be resolved later.

MAKEUSER

D2503 MAKEUSER - ACCESSCODE

MAKEUSER and USERDATAREBUILD have been modified to handle the ACCPSW LIST nodes, node type 25, including handling the crunching of the accesscode password. An accesscode may be added to a usercode's entry in the USERDATAFILE through MAKEUSER, using the node id "ACCESSCODELIST". The syntax is the same as for any other list item in MAKEUSER. An ACCESSCODELIST item consists of either an accesscode alone, or an accesscode followed by a slash followed by an accesscode password. If a password is desired, one must be supplied when the accesscode is created, or one may be supplied later only through MAKEUSER. If an accesscode is created with a password, the password may only be deleted through MAKEUSER. The password on an accesscode may be changed (through CANDE or WFL), but a password may not be added on an accesscode created without one, or deleted from an accesscode created with a password (unless a privileged user is running MAKEUSER). If an accesscode already exists for a usercode, with or without a password, when MAKEUSER tries to create it, the old one will be deleted and the new one inserted in the USERDATAFILE. This is done to avoid accidentally defining the same accesscode with and without a password or with different passwords. MAKEUSER passes the accesscode password to USERDATAREBUILD uncrunched; USERDATAREBUILD then crunches it the same way the usercode password is crunched and stores the accesscode and crunched password (if any) in the USERDATAFILE.

The "ACCESSCODENEEDED" bit may be set or reset through MAKEUSER, using the same syntax as for any other bit item in the USERDATAFILE.

D2964 MAKEUSER - ENHANCEMENTS FOR "DL USERDATA"

MAKEUSER has been modified to read and write the *SYSTEM/USERDATAFILE on whatever disk/pack family has been specified via the ODT Disk Location command DL USERDATA (see GENERAL note D2535).

The automatic family selection for an input file (via OLDFILE) or an output file (via USERFILE) will be bypassed if either FAMILYNAME (PACKNAME) or TITLE has been equated for that file.

The System Software Operational Guide, Volume 2 (Form No. 5001688), should be changed as follows with respect to label equation of these files:

for OLDFILE:	discard the clause " but in this case, the disposition options ... lost if a new one is locked".
for USERFILE:	Add a new second sentence: "If the USERFILE is equated during a COPY NEW, then upon completion the same TITLE and FAMILYNAME is applied to OLDFILE; any subsequent COPY commands will use the equated OLDFILE."

MCP-GENERAL

D2935 MCP-GENERAL - "MCP" GENERAL INFORMATION

New features have been added to the ODT messages as a result of the following:

B6800 Multiprocessor Systems
Global Memory Systems
Inter-Process Communication
Libraries
Disk Location
7A Tape Control

See GENERAL note D2535, "Revised ODT Messages", for complete syntax and semantics.

MCP

D2246 MCP - "B6800 MCP"

New hardware interrupt literals are now recognized, and, where appropriate, logged.

The format of log records reporting B6800 processor errors or diagnostic interrupts is as follows:

Word	Contents	
0	Link	
1	Date	
2	Time	
3	Type	31:16 = 2 (LOGMAJMAINT) 15:16 = 15 (MLCONFIDENCEERROR)
4	MCPID	47:12 = System Serial # 35:12 = Mark Digit 23:12 = Mark Level 11:12 = Patch Level
5	ProcessorID, WHOI Value	
6	P1 Parameter, P1 Interrupt Parameter with Tag set to 0	
7	P3 Parameter, P3 Interrupt Parameter with Tag set to 0	
8	P2 Parameter Tag	
9	P2 Parameter, P2 Interrupt Parameter with Tag set to 0	

The interpretation of P1, P2 and P3 is based on B6800 processor specifications.

D2249 MCP - "OF" OF "GETSTATUS" WAITING ON <FILENAME>

OF has been added to the possible replies of the MCP GETSTATUS RSVP of "WAITING ON <filename>". The possible replies are now "OK", "DS" and "OF".

D2268 MCP - IGNORE UNKNOWN UNIT TYPES

The MCP will now ignore units which are present at Halt/Load time but have a unit type (as supplied by the PC encoder card) which is greater than any type the MCP is programmed to handle. Previously, presence of such a unit on the system would cause Halt/Load initialization to fail with an INVALID INDEX.

Note that it is still not possible to Halt/Load with an unknown unit that uses a status-change interrupt, since the MCP must correctly map these by control and unit. For example, this change makes it possible for an older MCP to ignore diskette but not model 235 diskpacks.

D2269 MCP - "RESERVE" ENHANCEMENTS

The operational characteristics of RESERVEDISK have been improved. Previously, RESERVEDISK would hang waiting for exclusive use of a file before attempting to copy it. This caused delays - some of quite extended duration - during the pass of the permanent files residing in the area being reserved. RESERVEDISK may now continue through the list, and later recycle through it on operator request, presumably after some action has been taken to correct the blockage. If the operator is unable to correct the hangup, he may "QT" RESERVEDISK, which will create a new directory of XDISK titled "RESDISK".

RESDISK is an XDISK file which is titled in the same way as "BADDISK":

RESDISK/FMLYINX<family index no.>/UNIT<unitno>/AD<address>.

If a RESDISK file is encountered on a subsequent reserve, it will be removed. The effect of this is that any part of the old RESDISK which intersects with the new RESERVEDISK specification is absorbed into the new set of reserved disk, and the non-intersecting part is released as available for system use. The rationale of this operation is that it is a simple solution to an otherwise complex naming problem.

RESERVEDISK, at the end of a pass through the permanent files which intersect the area being reserved, will inform the operator of any files which could not be moved. The operator may then take one of three actions, as follows:

DS Releases all space reserved during this run of RESERVEDISK. This is equivalent to II.7 action.

OK Causes another pass through the permanent intersecting files. If any are still blocking the reserve, they will be displayed. If no blockage now exists, RESERVEDISK will proceed to handle temporary files.

QT Causes creation of RESDISK files out of areas which would otherwise be released, and then send the reserve to end-of-job. This option may be used to cut off the RESERVEDISK overhead, while preventing the subsequent allocation to areas which later may be required by a reserve.

RESERVEDISK will move executable and intrinsic files (except JOBCODE) even if they are open. Access to the headers is obtained with a "conditional exclusive" request and is gained if the file is not being used; otherwise, the file is opened as "shared". This may prove valuable for files such as CANDE and intrinsics, which are accessed via PRESENCEBIT and not logical I/O; therefore, disk addresses will not be left floating in buffers. Since they are read-only files, no WRITE to the wrong area after copying will occur. If a datafile is encountered open with only inactive rows intersecting the reserve area, these rows will be moved up to the first active row encountered.

The job number of the first stack to open a temporary file is displayed, if the stack is still alive, which should assist the operator in identifying which job is blocking the reserve. At this time (during a hang on scanning for the existence of temporary files), the operator may do a QT, the effects of which are described above.

Old BADDISK files are retained in the directory, even though the rows are released, if a subsequent RESERVEDISK absorbs them. The rowsize is set to zero, identifying the situation to the operator on a "PD" message.

If a BADDISK file exists and a reserve is done overlapping its high order address end, RESERVEDISK will no longer hang on a duplicate name.

D2271 MCP - "GETSTATUS MCSNAME"

The following are examples of GETSTATUS calls to convert MCSNAME to MCSNUMBER and MCSNUMBER to MCSNAME.

Name to number:

```
A[0] := 2;
A[1] := 0;
REPLACE POINTER(A[3]) BY <mcsname in display form>, NULL;
GETSTATUS(0 &1 [46:1] & 2 [7:8] & 1 [15:8], 0,
          0 & 1 [37:1], A);
```

The MCSNUMBER is returned in A[2+37+1].

Errors are returned in the normal GETSTATUS method if there is no MCS by that name or if the DCP is not is not running.

Number to name:

```
A[0] := 2;
A[1] := 0;
A[3] := <mcsnumber>;
GETSTATUS(0 & 2 [7:8] & 1 [15:8], 0, 0 & 1 [37:1], A);
```

The MCSNAME is returned in standardform in A[2 + A[2+37+1].[32:17]].

Errors are returned in the normal GETSTATUS method if there is no MCSNAME by that number or if the DCP is not running.

D2273 MCP - "IV" VS "235"

The IV routines for 235 diskpacks have been changed. 235 packs come from the factory already initialized. If the packs are damaged, they must be returned to the factory to be reinitialized.

The INITIALIZE and VERIFY routines in the MCP or LOADER must be used to put a label on 235 packs. However, these routines will not reinitialize any cylinders on the 235 packs.

D2430 MCP - "MCP" RESTRUCTURING

Several changes have been made to improve the maintainability of the MCP on the III.1 release, as follows:

The MCP has been converted from ESPOL to NEWP. Details of the relevant language features are described in the following notes:

NEWP Note D2804 - "Implementation of NEWP"
MCP Note P2288 - "MCP Conversion to NEWP"

The MCP has been grouped into logical modules. Each global item belongs to a particular module. Various "interfacing" defines have been added for accessing data from other modules.

Several external procedures which used MCP data structures have been moved into the MCP symbolic. These include SORT, MAINTENANCE, Reader/Sorter routines and some of the intrinsics which had previously been written in ESPOL.

These changes have affected the way in which an MCP is generated, as follows:

Compilation in NEWP

- DUMPINFO and LOADINFO are not supported.
- Dollars cards not beginning with SET or RESET will not affect unmentioned options. To reset all options, $CLEAR can be used.
- The compiler produces its own cross-reference; SYSTEM/XREFANALYZER need not be run.
- A MAKEHOST and a SEPCOMP facility exist similar to ALGOL, except that NEWP does its own binding.

Symbolic Consolidation

- SORT and MAINTENANCE need not, and cannot, be separately compiled and bound. They can, however, be changed using SEPCOMP.
- The DCALGOL components of the MCP are compiled and bound as before.
- The ESPOL and PLI intrinsics no longer do a LOADINFO from the MCP.

SAMPLE HOST COMPILE DECK

```
JOB NEWP/MCP;
BEGIN

PT(STATUS=0);
PROCESS SYSTEM/PATCH [PT];
 FILE TAPE = SYMBOL/MCP,
      PATCH = PATCH/NEWPMCP;
DATA
$.COMPARE MARK
$#
$CLEAR MERGE LINEINFO MCP MAKEHOST
<patches>
? % END PATCH INPUT

WAIT(PT(VALUE)=1);
IF PT(VALUE) NEQ 1 THEN ABORT "BAD PATCH";

COMPILE NEWP/HOST WITH NEWP [CT] LIBRARY;
 COMPILER FILE TAPE = SYMBOL/MCP ;
 COMPILER FILE CARD = PATCH/NEWPMCP DISK;

IF CT ISNT COMPILEDOK THEN ABORT "BAD COMPILE";
REMOVE PATCH/NEWPMCP;

BIND NEWP/MCP BINDER[BT] LIBRARY;
 BINDER FILE HOST= NEWP/HOST;
DATA
 BIND JOBFORMATTER      FROM SYSTEM/JOBFORMATTER;
 BIND WFL               FROM SYSTEM/WFL ;
 BIND CONTROLLER        FROM SYSTEM/CONTROLLER ;
 BIND CCSTRINGCONV      FROM SYSTEM/CCSTRINGCONV;
 BIND CCVARIABLEPPB     FROM SYSTEM/CCVARIABLEPPB;
 BIND CCSTRINGFUNCTION FROM SYSTEM/CCSTRINGFUNCTION;
?
IF BT ISNT COMPILEDOK THEN ABORT "BAD BIND";

END JOB.
```

SAMPLE SEPCOMP DECK

```
JOB NEWP/SEP;
BEGIN
PROCESS SYSTEM/PATCH [PT];
 FILE TAPE = SYMBOL/MCP,
      PATCH = PATCH/NEWPSEP;
DATA
$.COMPARE MARK
$#
$CLEAR LINEINFO MCP SEPCOMP "SYSTEM/MCP."
<patches>
? % END PATCH INPUT
WAIT(PT(VALUE)=1);
IF PT(VALUE) NEQ 1 THEN ABORT "BAD PATCH";

COMPILE SEP/MCP WITH NEWP [CT] LIBRARY;
 COMPILER FILE TAPE = SYMBOL/MCP ;
 COMPILER FILE CARD = PATCH/NEWPSEP DISK;

IF CT ISNT COMPILEDOK THEN ABORT "BAD COMPILE";
REMOVE PATCH/NEWPSEP;

END JOB.
```

D2439 MCP - DIRECTORY CONVERSION

The MCP will no longer recognize or convert disks or packs that use pre-II.7 directory format.

D2447 MCP - "COPY" VS. SERIAL NUMBERS

Library Maintenance has been changed to use serial numbers when they are specified for source disk and destination disk volumes.

Note that when a non-zero serial number is specified, family substitution does not take place.

D2452 MCP - BINARY PACK ADDRESSES

The MCP now supports binary addressing of diskpacks. The controlware loaded into the diskpack control can be either binary or decimal.

In order to change from one type of controlware to the other, do the following:

1. CM to the MCP to be used.
2. Halt/Load on the MCP to be used. In the case of duplicated MCP's, be sure to Halt/Load on every backup pack involved.
3. Halt the system and load the new controlware into every diskpack control.
4. Halt/Load the system.

Do not use the HOST LOAD command (LH) to change types of firmware.

D2463 MCP - "RESOURCECHECK"

The tape resource subsystem (ODT option RESOURCECHECK) has been changed. The count of available tapes in the system pool is now taken as the number of "existing" tape units minus the count of reserved, saved, in use and "set aside" (tapes already requested via a RESOURCE statement in a job) tapes. An "existing" tape unit is a tape unit that has been READY'ed at least once since the last Halt/Load. Formerly, the count of available tapes in the system pool was taken as the number of ready tapes minus the count of reserved, in use and "set aside" tapes.

D2493 MCP - ACCESSCODE

Overview

The B6000/B7000 system software has been modified so that a file can require an ACCESSCODE before access is granted. One of the results of this is to enable control of access rights to a file among different users with the same usercode.

An additional item has been added to the USERDATAFILE called "ACCESSCODELIST", consisting of ACCESSCODEs optionally protected by passwords. A new task attribute called "ACCESSCODE" has been added (the task's ACCESSCODE). An ACCESSCODE may be assigned to a session, job or task through CANDE, RJE, WFL and the other compilers; however, it is validated first in the USERDATAFILE. SYSTEM/GUARDFILE has been modified so that the type of access granted to a file (NO, RO, WO, RW, XO) can be determined by usercode, program name, accesscode or any combination of these three items. A new SECURITYTYPE file mnemonic called "CONTROLLED" has been added. It is equivalent to "GUARDED" (CLASSB) except the guardfile will be invoked even for the creating (owning) usercode. File open in the MCP has been changed so that when a guardfile is invoked, the accesscode of the requesting task is also checked.

ACCESSCODE Task Attribute

A new task attribute, ACCESSCODE, has been added to the MCP. It is a pointer valued task attribute. An accesscode may be protected by a password, as the usercode is, but only the accesscode is stored in the task, not its password. An accesscode may have no more than one password. Both the accesscode and its password are stored in the USERDATAFILE, but the password is crunched (like the usercode password), so it is never listed explicitly. If an accesscode has a password, it must be specified if the attribute is to be changed, but only the accesscode is returned on read access.

Example:

```
    REPLACE MYSELF.ACCESSCODE BY "ACCESS/ITSPW.";
```

The following statement would return "ACCESS". in X[*]:

```
    REPLACE POINTER (X) BY MYSELF.ACCESSCODE;
```

The MCP will check the validity of the accesscode and password each time it is changed on an active task. Attempting to change to an invalid accesscode/password will be a fatal run-time error. The accesscode of the parent task will be inherited by any subtasks (including a job, CANDE session or RJE subtask) unless the usercode or accesscode is explicitly specified to the subtask. If the usercode of an active task is changed, the accesscode for the task becomes null.

In order to change the title of a file or the security of a file or to remove a file, the user must be running under the usercode that owns the file (except for privileged users). If the file is "CONTROLLED", the user must also supply or have assigned to his session an accesscode which has read/write access to the file.

USERDATA

USERDATA in the MCP has two new functions to handle accesscodes.

Function 11: VALIDATE ACCESSCODE/PASSWORD

TSK may be a task variable to which the accesscode will be assigned, or it may be empty. The LOCATOR is not used. If ACTION.COPY is set, the USERDATAFILE entry will be copied into OUTSTUFF, as with Function 3. ACTION.STANDARD must be set, and USERDATA must be called from an active MCS or the MCP. INSTUFF must contain the usercode, accesscode and (optionally) the accesscode password as a three (or two) level file id in standard form with the qualification byte either 0 or 1. If the qualification byte is 0, the accesscode will be assigned to the task TSK without being validated first; otherwise, the accesscode and its password (if any) are validated in the USERDATAFILE before the accesscode is assigned to the task TSK. The accesscode password, if there is one, is passed to USERDATA uncrunched, and USERDATA will crunch it before validating the accesscode. Examples of input to this function (in standard form notation) are as follows:

```
    4"0F010204"  8"USER" 4"06" 8"ACCESS"
    4"18010304"  8"USER" 4"06" 8"ACCESS" 4"08"  8"PASSWORD"
```

Function 12: CHANGE ACCESSCODE PASSWORD

There are two forms for this call: when called by a WFL job, and when called by an MCS. When called from a WFL job, INSTUFF must be the new password in display form. The calling task must have a usercode and a valid accesscode.

Otherwise, INSTUFF must contain the usercode, accesscode, old accesscode password and the new accesscode password as a four level standard form file id. USERDATA must be called from an active MCS or the MCP, and ACTION.STANDARD must be set. TSK, LOCATOR and OUTSTUFF are ignored. The accesscode password may be changed, not added or deleted. The accesscode and old password are validated first and the password is only changed if validation is successful. An example of input to this function (in standard form notation) is as follows:

```
    4"1B010404"  8"USER"  4"06"  8"ACCESS"  4"05"  8"OLDPW"
       4"05"  8"NEWPW"
```

Four new error codes have been added to USERDATA, as follows:

50 INSTUFF not in standard from
51 ACCESSCODE/PASSWORD not found in USERDATAFILE
52 more than one accesscode password (INSTUFF too long)
53 task does not have an ACCESSCODE

MAKEUSER and USERDATAREBUILD have been modified to handle the ACCPSW LIST nodes, node type 25, including handling the crunching of the accesscode password. If an accesscode already exists for a usercode, with or without a password, when MAKEUSER tries to create it, the old one will be deleted and the new one inserted in the USERDATAFILE. This is done to avoid accidentally defining the same accesscode with and without a password, or with different passwords. MAKEUSER passes the accesscode password to USERDATAREBUILD uncrunched; USERDATAREBUILD then crunches it the same way the usercode password is crunched and stores the accesscode and crunched password (if any) in the USERDATAFILE.

Logging

The log file will store the accesscode (without password) in the following log records:

BOT/BOJ	(item #11)
EOT/EOJ	(item #25)
FILE OPEN	(item # 8)
BEGINNING OF MCS SESSION	(item #11)
END OF MCS SESSION	(item #25)
SECURITY VIOLATION	(item # 7)

The accesscode will appear in the jobsummary, and will be printed in the banner. (If the accesscode is not desired in the banner, it may be eliminated by resetting the BCACCESSCODEF bits in the PRINTMASK of the MCP.)

A change of accesscode in a session is logged exactly as a change of chargecode in a session with a "SIGN ON" record.

SECURITYTYPE File Mnemonic

A new value for SECURITYTYPE has been added called "CONTROLLED". A CONTROLLED file is identical to a GUARDED file except the guardfile will be checked even for the creating (owning) usercode.

AVAILABLE File Attribute

A value of 14 will be returned for <file>.AVAILABLE if the file exists in the user's directory but it not available due to accesscode. This new value is returned only for files in the user's directory.

File Open

The MCP file open procedure has been changed so that an attempt to open a file with SECURITYTYPE=CONTROLLED will cause a call on CHECKGUARDFILE in all cases. In CHECKGUARDFILE, all three items (usercode, program name and accesscode) will be used to determine access rights.

Task Initialization

At task initialization time, the MCP will determine if an accesscode has been explicitly assigned to the task. If so, the accesscode and password will be validated in the USERDATAFILE for the task's usercode. If no accesscode has been explicitly assigned but a usercode has been explicitly assigned, the task will have a null accesscode. If no accesscode and no usercode have been explicitly assigned, the task will be given the parent task's usercode and accesscode.

D2518 MCP - SEGMENTED STRING VARIABLES

An ALGOL string is initialized to a maximum length of 132. If this length is exceeded, the string is resized (to multiples of 32 words) to a maximum of 256 words. If this length is exceeded, the string is segmented and increased in increments of 256 words to its specified maximum size (currently 65534 characters).

D2570 MCP - DISALLOW "PRINTERLABELS" OPTION

The PRINTERLABELS option has been removed from the MCP options list. Printer file volume and header labels are no longer printed at the time the files are opened and closed. Option 24 has been redefined as OKTIMEANDDATE; see MCP note D2592.

D2576 MCP - "RERUN" VS. "EOF"

RERUN has been corrected; it now correctly repositions tape files that were at or near EOF. Formerly, RERUN aborted with the message "FILE POSITIONING ERROR".

The following new error messages have been added to checkpoint RERUN:

1. UNSUPPORTED KIND=nn:<file id>

   Cannot restart from checkpoint taken with file of given KIND open (e.g., CARDPUNCH).

2. ERROR REPOSITIONING TAPE: MTnn <file id>

   Either an I/O error occurred or an extra or missing EOF was detected.

3. ERROR IN IODISKADDRESS: PKnn <file id>
   ERROR IN FINDINPUT: <file id>
   ERROR IN OPEN: <file id>

4. **INCOMPATIBLE FILE ATTRIBUTES: PKnn <file id>**

File attributes have changed since the checkpoint was taken (e.g., EOF moved backwards, different AREASIZE, etc.).

D2580 MCP - "PRE-2.4 ON <FAULT> STATEMENTS"

The MCP version 3.1 no longer supports the 'ON <fault>' statements compiled by the ALGOL, DCALGOL or ESPOL compilers prior to the 2.4 release. An entirely new mechanism was implemented in 2.4 (released mid-1973).

Any existing code file which was compiled with a 2.3 or older compiler and which uses 'ON <fault>' statements must be recompiled prior to being run on a 3.1 system.

An attempt to run the old object code on a 3.1 MCP will cause the program to be terminated whenever an 'ON' statement is executed, with the following message

RECOMPILE PRE-2.4 CODE WITH 'ON' STATEMENT @ <addr>

The <addr> is the line number, or seg:word:syl address if no lineinfo is present, for the 'ON' statement. Only one message is generated for any one task. The message will appear in the job log, and will be sent to the user's terminal if the task was run via CANDE and the user's MESSAGE options is set.

Note that the 'ON' statement is "executed" whenever it appears in the flow of control, to arm the fault trap. It is at arming time, not fault-detection time, that the program will be aborted.

D2592 MCP - TIME AND DATE VERIFICATION

Sites may now require their operators to verify that system TIME and DATE values are valid. A new system option (OPTION 24 = OKTIMEANDDATE) may be set from the ODT via the "OP" (or "SO") command to require the verification. In addition, the verification will be required on B6800 multiprocessor systems regardless of the option setting if the clock of any processor varies from that of the first processor to halt/load by more than 60 seconds during the system initialization sequence. (Whenever B6800 time-of-day registers disagree as a multiprocessor system is initialized, they are all synchronized at the maximum time found in any of them.)

When verification is required, the current TIME and DATE settings will be displayed on the ODT and updated whenever ADM would ordinarily be updated. The operator may enter "TIMEOK" to resume normal processing after resetting any invalid time or date via the "DR" and "TR" commands.

D2605 MCP - NEW FUNCTION FOR "DMSWAIT"

DMSWAIT has been modified to indicate whether or not the calling stack is in transaction state. No other function of DMSWAIT is affected. In addition, the meanings of the various parameters value is documented.

D2606 MCP - "UNITS=CHARACTERS" VS. "MODE=SINGLE"

If a user makes the inconsistent file specification UNITS=CHARACTERS, MODE=SINGLE, the file is treated as word-oriented; the UNITS specification is overridden. Effective with the III.1 MCP release, the attributes recorded in a permanent DISK or PACK file with these specifications will be adjusted to be consistent; i.e., the header attributes will be UNITS=WORDS, MODE=SINGLE.

D2642 MCP - "SYSTEMSTATUS" INTRINSIC

SYSTEMSTATUS is subject to constant change or new features are implemented. See Appendix F, "Changes to SYSTEMSTATUS", for details of the changes for the III.1 release.

Because of these changes, the III.0 version of the SPARK package may not be used on the III.1 system.

D2644 MCP - SUSPEND AND RESUME TASKS BY BOX

It is now possible to suspend and resume tasks on a box basis with SETSTATUS. The call to SETSTATUS is TYPE=2, SUBTYPE=37 with the value parameter as follows:

Bit 0 = 1 Suspend
Bit 0 = 0 Resume
Bit 1 = 1 On a box basis
Bit 1 = 0 Same as old call

If Bit 1=1, arrayrow [1] must contain the number of the box to be used. Allowable box numbers are 1 through 5 inclusive. If the box number passed is not valid or not present, SETSTATUS will return an appropriate error. This change applies only to B6800 Multiprocessor systems where the boxes are defined as follows:

1 = global
2 = processor 1
3 = processor 2
4 = processor 3
5 = processor 4

D2647 MCP - "LOG IOCONDITION" RESULT DESCRIPTORS

A new maintenance log entry subtype=9 has been added to log I/O results that indicate retries that were not errors (diskpack read and write 4"C801" and read 4"0C01").

These entries will be printed in response to a LOG MAINT or a LOG IOCONDITION request; they will not be printed for a LOG IOERROR request.

These diskpack result descriptors are accepted as successful I/Os and are not retried.

D2652 MCP - NEW "SYSTEMSTATUS" CASE "14"

A new case has been added to SYSTEMSTATUS; it is type 14, subtype 0. Its exclusive purpose is to support SUSPENDER's information needs.

D2654 MCP - HANDLING UPLEVEL POINTERS

The handling of potential uplevel pointers in the MCP has been modified, as follows:

1. An ALGOL, DCALGOL or DMALGOL program is deemed internally free of uplevel pointers if it was compiled by a compiler whose version is III.0 or later. Any older ALGOL-type codefile is considered as potentially containing uplevel pointer assignments. The bit set by the ALGOL compilers in segment 0 of the code file (COMPILERINFO.NOBADPTRF) is disregarded, since it is always set in III.0 or III.1 and not reliable in II.8 or II.9.

   Programs compiled by COBOL, FORTRAN, BASIC, PL/I or WFL continue to be considered free of uplevel pointer problems whatever the compiler version.

   As before, the possibility of uplevel pointer assignment into a parent task is ruled out if the offspring is external (separate code file) and the parameters passed could not permit such assignment.

2. If the MCP cannot ascertain that uplevel pointer assignments from offspring to parent are impossible, it will force the offspring to run in the same memory environment as the parent. Thus the offspring could not become a swapped task, nor could the offspring run in a different memory subsystem from the parent on a B6800 multiprocessor system.

D2655 MCP - "STOPPOINT" TASK ATTRIBUTE

The STOPPOINT attribute reports the point in a program at which something abnormal occurred. If a fault is detected, the STOPPOINT value takes on a value consisting of the Return Control Word (RCW) information in bits 35:36 and the fault REASON in bits 47:8.

The RCW fields are as determined by the processor:

bits	12:13	SDI	Segment Dictionary Index
	13:1		0/1 for MCP/non-MCP code
	32:13	PIR	Index of word in code segment
	35:3	PSR	Index of syllable in code word (0-5)

SID,PIR and PSR are generally reported in compler listings and error messages in the form 017:02B3:5; they uniquely specify the syllable of code in a given program file.

The REASON values are the same as those reported in the HISTORY task attribute bits 23:8 when bits 15:8 specify FAULTCAUSE (4):

1: Divide by zero
2: Exponent overflow
3: Exponent underflow
4: Invalid index
5: Integer overflow
6: Inactive queue (software-detected)
7: Memory protect
8: Invalid operand or NVLD operator
9: Instruction timeout
10: Memory parity error
11: Scan parity error
12: Invalid address
13: Stack overflow
14: String protect (segmented array) fault
15: Programmed operator
16: Bottom-of-stack fault
17: Sequence error
18: Invalid program word
19: Stack underflow

When a task terminates normally, STOPPOINT is zero. For an abnormally-terminated task, STOPPOINT contains the RCW information. If the task was terminated for a fault, REASON will specify the fault; if it was terminated for some other cause the REASON will be zero.

D2694 MCP - IMPLEMENTATION OF "DL"

System file handling has been redesigned to allow OVERLAY, BACKUP, JOBDESC, CATALOG, SYSTEM/SUMLOG and SYSTEM/USERDATAFILE to reside on families other than the Halt/Load family. The user may explicitly specify the families on which these system files reside via the DISKLOCATION ODT message. Any unspecified DISKLOCATION familyname defaults to the <Halt/Load family> during MCP initialization.

Complete syntax and semantics for the DISKLOCATION ODT message are described in III.1 GENERAL note D2535.

D2705 MCP - "7A" TAPE CONTROL

The new magnetic tapes have the following features. A given tape drive will be able to read and write 800 BPI 9 track tapes and 1600 BPI PE tapes or 1600 BPI PE tapes and 6250 GCR tapes. The control 7A (with scanin unit type code of 4"0C") provides extended status (similar to that provided by all new unit controls) that can be used to help determine the cause of I/O errors.

The density setting is automatically determined by the tape drive. Once the first record on the tape has been read or written, the drive remembers the density setting; this cannot be overridden by bits in the IOCW. Furthermore, a read at load point automatically determines the proper density from "id burst"s on the front of the tape, no matter what the density bit settings in the IOCW request. Only in the case of a write at load point does the subsystem use the code bits in the IOCW to set the density, which is remembered until the tape is rewound. Note that a test op at load point does not return the proper density in the result descriptor (because the "id burst" has not been read); rather, it always returns the code for 1600 BPI.

These tape drives are treated as PE tapes that have two different densities; i.e., in the same manner that 7 track drives are treated as having three different densities.

Kind:

The unit KIND is PETAPE (VALUE 15).

Density:

The new density for 6250 BPI is 4. The menmonic is BPI6250 (see ATTABLEGEN note D2723).

FINDOUTPUT's masksearches have been modified since a request for KIND=PETAPE with density of 4 cannot be satisfied by just any PETAPE, only a new one.

READALABEL and PURGIT

READALABEL and PURGIT have been modified to delay testing the density until after reading the first record on the tape. An optional modifier has been added on the operators SN and PG so they may change the density of a given tape volume. (See GENERAL note D2535 for syntax.)

TAPEPARITYRETRY

IOFINISH68 reads the log via the "IOCREADLOG-IOLOGAREA-MASKGETAREAF" mechanism. IOEXCEPTION passes this to IOERROR and TAPEPARITYRETRY, which logs it via MLEXTRDS and/or FORGETAREA as required.

SYSTEM/LOGANALYZER

LOGANALYZER prints out the extended status stored by TAPEPARITY for I/O errors.

FIBSTACK

The OPEN routine is aware of new IOCW density settings. OPEN also checks for the new allowable DENSITY/KIND combinations. CLOSE records the new density value in the HDR2 UDNSTY flag.

D2737 MCP - PRIVILEGED PROGRAMS

The privileged program feature associates security of a running program with the code file being executed in addition to the usercode under which it is being run.

The following are the privileges of running programs:

1. The privileges of a program running under a privileged usercode do not change.

2. Conferring of privileged usercode in the MCP does not change (i.e., task initiation or MCS),

3. A privileged program has the same privileges as a program running under a privileged usercode.

4. Code belonging to a privileged program is always privileged, regardless of the usercode of the stack in which it is running.

5. A program can only be made privileged by the ODT command "PP". The command designates a code file as a privileged program. code file.

```
-- PP ---------<filetitle>--|
        |   |
        |- - -|
```

   <filetitle> references the code file to be marked as a privileged program. A minus preceding the file title denotes that the code file is no longer a privileged program.

6. A control program is one which is not scheduled and will not be suspended. This notion is now independent of the privileged status of the program.

7. The ability to access the directory past non-visible nodes is subsumed by privileged programs.

8. COMPILERINFO (which includes the privileged program bit) will reside in the task of the segment dictionary as well as the process task.

D2738 MCP - DELETE "CPUTEST"

The CPUTEST feature in the MCP has been removed.

D2757 MCP - NEW "DMS" STRUCTURE NUMBERS

MCP support is now provided in DMSOPEN for new ACR structure number format.

D2781 MCP - DATA BASE STACKS RUN IN LOCAL MEMORY

Data bases may define the subsystem in which they are allowed to run (via the SUBSYSTEM attribute). This is checked against the subsystem which is being used by the user who is opening the data base to see if the data base can be seen by the user. If the data base is already open and the memory in which the DBS exists cannot be seen by the task attempting to open the data base, the task will be terminated. Note that if the data base SUBSYSTEM includes * GLOBAL TM Memory, the DBS will be placed there.

* "GLOBAL Memory" is a trademark of Burroughs Corporation.

D2784 MCP - "NIF" AND "DCPCODE" DISK "I/O" ERRORS

Disk I/O error checking code has been implemented for datacom I/O. The datacom subsystem depends on two files, NIF and DCPCODE, for information about the datacom environment. The NIF file contains logical and physical specifications of the datacom network, and it never changes after it is created by the NDL compiler. The DCPCODE file contains the DCP hardware instructions and current datacom station/line information, and it will change as the datacom environment is altered by DCRECON requests.

When a datacom disk error occurs, one of the following messages will be displayed:

1. NIF FILE DISK INPUT ERROR
2. DCPCODE FILE DISK INPUT ERROR
3. DCPCODE FILE DISK OUTPUT ERROR

   One of these messages will inform the operator as to the type of disk I/O error that occurred. No immediate response is required. A disk I/O message will be displayed prior to this message. The MCP will then appropriately respond to the error. Depending on the frequency of the errors, it may be necessary to terminate datacom and either move or reload the datacom files (see "Processing a DCRECON Request", below). WARNING: reloading the DCPCODE file will cause the loss of all datacom reconfigurations that have completed.

4. BAD NIF FILE RECORD NUMBER
5. BAD DCPCODE FILE RECORD NUMBER

   One of these messages will inform the operator that an attempt was made to perform I/O to a non-existent part of a datacom file. No immediate response is required. If the compile-time option DIAGNOSTICS was set in the MCP, a non-fatal dump will occur. The MCP will then appropriately respond to the error.

After displaying one of the above messages, the MCP will respond to the error. The most important responses that can occur are:

1. During datacom initialization.

   One of the following messages will be displayed: "DCP INITIALIZATION FAILED" or "DATACOM INITIALIZATION FAILED", which indicate respectively that a DCP could not be initialized or that the datacom tables could not be built. The MCP will then terminate the DCP concerned, and if no DCPS are running, the datacom tables will be deallocated.

2. Processing a DCRECON request.

   Two messages will be displayed: (1) "DCRECON (TYPE=<RECALLJOBOUTPUT, LINESWAP, CLUSTEREXCHANGE, STATIONMOVE or UPDATELINE>): BAD DATACOM DISK I/O." and (2) "DCRECON: PERMISSION TO ATTEMPT RECOVERY.". The latter message will require a response of "OK" or "DS" from the operator. If "OK" is entered, the current request will be restarted, but it may not be possible to recover from the failure. If "DS" is entered, the current request will be discontinued, and the next request will be started. If "DS" is chosen, however, the datacom subsystem may be incompletely initialized. If datacom is not functioning correctly because of this (e.g. missing lines or stations), then it will be necessary to terminate datacom and re-initialize.

3. MCS Interface.

   A new DCWRITE error (170) has been created for datacom I/O errors. In addition, if an I/O error occurs while an interrogate station environment result (class=15) is being built, the result will not contain the station's name; this is also true for file OPEN and CLOSE results (MSG[8].47:8 = 0).

All other responses are self explanatory.

D2795 MCP - "DS" PRIORITY

When a task is externally DSed, its priority class is increased to force the task to clean up and leave quickly. (Formerly, the base priority was increased but not the invisible, high-order priority class; this has proved insufficient.)

If, for some reason, the task continues to loop in non-DSable code, there is now a recourse, as follows:

The operator may enter the following ODT message:

```
    <mix no.> PR 0
```

Setting the base priority explicity to zero will also remove the priority-class enhancement due to DS, so that task will become uncompetitive for a processor.

D2796 MCP - DEIMPLEMENTATION OF OLD INTRINSICS

A warning will now be generated when executing code files using certain old intrinsics which will be deimplemented on the III.2 system release. Please consult the following GENERAL D-Notes for details about the relevant languages:

ALGOL	D2797
FORTRAN	D2798
BASIC	D2799
PL/I	D2800

D2833 MCP - MPX#,PATH# VS. "LH,UA,UR"

The MCP will now log all SETSTATUS calls before anything can happen to the input array. For example, the MPX# and PATH# for LH, UA and UR commands will be logged correctly.

D2834 MCP - CHECK STACK IMAGE COMPATIBILITY

WFL jobs that are restarted after a Halt/Load are now checked to ensure that the stack image is compatible with the current MCP level. Incompatible jobs are DSed. Jobs are compatible between MCP cycle III.1.120 (TC) and the forthcoming III.1 release. A CM from any III.0 MCP or any earlier III.1 (TC) MCP will cause any WFL jobs that were active before the CM to be DSed. CM between any III.0 or III.1 level will have no effect upon jobs that are in the queues but never started.

D2855 MCP - TIME OUT FOR MEMORY DUMPS

The MCP "clock" is stopped while taking a memory dump; thus, the processor and I/O times for certain unlucky actions are not distorted by the time to take the dump.

The TIME(14) intrinsic now returns the time since Halt/Load less any time spent taking memory dumps.

D2864 MCP - RELATIVE "I/O" FOR "COBOL74"

This note describes the relative I/O subsystem. The implementation is designed to support the requirements of COBOL74. Mapping of logical I/O to physical I/O is accomplished by the MCP I/O subsystem; other features relating more to programmatic views of the state of the relative I/O file are maintained by the COBOL compiler.

The relative I/O facility maintains records in a disk or pack file which was created as a relative file. Records are marked as either valid or deleted.

Each block contains a fixed number of fixed length record positions. Flags are provided at the beginning of each block to contain the validity information; there is one bit per record position in the block. The first record starts on a byte boundary.

The physical file is denoted as a relative file by use of the attribute FILEORGANIZATION. See IN-OUTPUT note D2866 for a description of this attribute.

As each row of a file is allocated, the MCP initializes each block of the new row to indicate that each record of the block is deleted. Note that no end-of-file action is implied by this process; EOF action is unchanged from prior implementation.

Creation, deletion and overwriting of records in a relative file is accomplished in accordance with the ANSI74 COBOL Standard.

D2865 MCP - VERIFY "235" PACKS IN "IV" MESSAGE

The Operator Display Terminal Reference Manual (Form No. 5001704) should be corrected as follows. The note on Page 2-47-A should read:

> "The IV routines in the MCP or the LOADER used to label 235 packs will not reinitialize any cylinders in the 235 packs; they will write out a master available label and a volume label."

D2869 MCP - "MCP" LEVEL INDICATORS

Some changes are being made in MCP level indicators.

BACKGROUND

Each released software item has a level indicator specified at compile time.

Example:

```
$ VERSION 31.190.728
```

This indicates marklevel 31 (often denoted as III.1), cycle 190, patchnumber 728. The cycle number increases throughout the development process, and is synchronized for all products in any release to the field. The patchnumber varies with the number of changes in each software item. The patchnumber concept predates the cycle reckoning, and used to be the more significant number for the MCP, but the cycle number is now much more convenient.

DISPLAYS

The displays from the CONTROLLER, JOBFORMATTER, PROGRAMDUMP and DUMPANALYZER have been changed to use the same format as the original $VERSION specification. Thus what might have been "III.1.728" or "III.1.190.728" is now "31.190.728".

INTERFACES

Those MCP interfaces that have returned binary representations of MCP mark.level.patch will now return mark.level.cycle. These include SYSTEMSTATUS, GETSTATUS, and the system LOG.

D2881 MCP - PROVIDE FIRMER "MAKEUSER-USERDATA" INTERFACE

A new USERDATA function has been provided to allow SYSTEM/MAKEUSER to ascertain the exact location of the SYSTEM/USERDATAFILE in use by the MCP.

Only the ACTION (first) and OUTSTUFF (fourth) parameters are used; the others should all be zero. The ACTION parameter = 14 & x [4:1], where x specifies how USERDATA will behave if the SYSTEM/USERDATAFILE is unavailable: if x=0 then USERDATA will wait displaying RSVP messages ("REQUIRES PK" or "NO FILE"); if x=1 then USERDATA will return an error value. The OUTSTUFF parameter should be a pointer. If the file is present and has the correct format, the USERDATA function value will be even. If the file was not present or was not a valid userdata file, the returned value will be odd and contain an error code as specified in the USERDATA/MAKEUSER documentation. If the file is present, the function value will contain the base-unit serial number (as an integer) in the UDVLOCF field, [35:20]. In any case, the DL userdata family name (followed by a period) will have been transferred via the pointer.

Action 14 is usable only by a program with privileged-user status.

If this function is used while the USERDATAFILE is "frozen", the calling program is assured that the file exists and will remain present and will not be modified by the MCP.

A new RSVP message is now used by USERDATA if some process is attempting to modify the userdata file while it is frozen, and has chosen to wait. The message is "WAITING FOR USERDATA FILE FROZEN BY TASK nnnn", where nnnn is the mix number of the task which has frozen the userdata file (typically SYSTEM/MAKEUSER doing a COPY or CREATE statement.)

D2888 MCP - LIBRARY MAINTENANCE RESULT REPORTING

Library Maintenance will now return the following in its TASKVALUE:

0 - all actions carried out correctly
1 - one or more files not copied

The latter category indicates that either Library Maintenance DSed, or in case of some exception condition, the operator chose not to retry the failing action (by OF or FR input).

D2907 MCP - AUTOMATIC MOVE

The MCP now permits the operator to move an in-use pack from one unit to another without entering a "MOVE <pk>" command. This will allow recovery from disk pack problems even when the controller is hung.

D2908 MCP - INITIALIZATION NEEDS MORE MEMORY

If the operating system requires additional global memory during the early part of initialization, it will now defunct displaying "NEED MEMORY - HALT LOAD".

After initialization is capable of taking a non-fatal memorydump, it will check to make certain that there are at least four mods (16K each) of private global memory. If not, there will be a non-fatal dump by "NEED GLOBAL MEM". The operator may choose to Halt/Load or DS the dump and continue.

D2911 MCP - LOCAL DIRECT ARRAYS

In the initial release of Tightly-Coupled systems, all direct arrays were forced global. In the III.1 MCP to be formally released, the MCP will automatically site direct arrays into global as needed for I/O visibility, but by default direct arrays of local processes will be in local. Meanwhile, the TC field-release MCP has been modified to permit the careful use of direct arrays in local.

Direct arrays will be global by default (as before), except:

Those belonging to a local DBS will be local.

Those belonging to a local MCS will be local unless the MCS has set MYSELF.OPTION:=MYSELF.OPTION & 1 [30:1].

While system option 35 is set (operator enters OP + 35), direct arrays belonging to any local stack will be local unless that process has set its OPTION.[30:1].

The only reason for forcing direct arrays global on a TC system with all tape and disk-type devices exchanged to both processors is to do direct I/O to a non-exchangeable unit (printer, card reader, etc.) from the other side of the system. The only standard software that does so is SYSTEM/BACKUP.

This feature involving system option 35 and task option bit 30 is temporary for the TC release, and no mnemonics have been provided for setting them via the CONTROLLER, CANDE or WFL. In the forthcoming III.1 MCP, these bits will revert to their unused status.

D2917 MCP - NEW "BOJ/BOT, EOJ/EOT" LOG INFORMATION

Several new items of information have been added to the BOJ/BOT and EOJ/EOT system log entries. They contain data about the tightly or loosely coupled system environment and provide additional information for performance measurement. The new words and their contents are:

BOT WORD	EOT WORD		CONTENTS
13	25		Compiler information from COMPILERINFO
		[47: 1]	IPC capable
		[46: 1]	Sort capable
		[45: 1]	Control program (CP)
		[44: 1]	DMS capable
		[43: 1]	No uplevel pointers see D-Note 2654 for details
		[42: 1]	Privileged program (PP)
		[41: 1]	Library capable
		[31: 8]	Language type 0 = ALGOL 1 = COBOL 2 = FORTRAN 3 = XALGOL 4 = PL/I 5 = JOVIAL 6 = NEWP 7 = ESPOL 8 = DCALGOL 9 = BASIC 10 = WFL 254 = INTRINSICS ($INTRINSICS set) 255 = MCP ($MCP set)
		[23: 8]	Compiler mark and level number (e.g., 31)
		[9:10]	Compiler cycle number
14	19		SWAPPER information from SWAPSPEX and SWAPINFO
		[39:10]	Number of swap space core slots in use
		[15: 8]	SUBSPACE attribute requested 0 = Do not run in swap space 1 = Data (D2) stack only in swap space 2 = Data in swap space. Code (D1) stack in swap space if code file in my directory 3 = Data and code in swap space
		[7: 8]	SUBSPACE actually granted same as [15:8] but 2 is not valid
15	26		Task's box numbers
		[39:20]	Box number of code (D1) stack
		[19:20]	box number of data (D2) stack
16	27		Link to value of SUBSYSTEM attribute
17	28		Link to name of originating host
18	--		Amount of time spent in the schedule
--	29		Amount of time spent in the ready queue

Also, the job entry time in BOT word 12 (added in III.0 P-Note 1383) is the time stamp of the code file. Thus, for a job it is the time the WFL compiler finished with it and it entered the system. For a task, it is the timestamp of the object code file being run. It is zero if the code stack was being shared with an already running task. The format is: [47:16] (Julian date - 70000) [31:32] (Time of day in ticks DIV 16)

D2928 MCP - ELIMINATE VECTOR MODE, IMPROVE CHECKSUM

The ability for the MCP to use vector mode has been eliminated. Also, substantial changes have been made to the CHECKSUM procedure to make it competitive (on the B6700) with the vector mode checksum. (This new checksum is substantially faster on the B6800 than the vector mode version.) The results generated by this new CHECKSUM are identical to results generated by older versions.

Since the MCP is no longer aware of the vector mode capability of the processor, bit 47 in the value of TIME(23), as described in II.9 MCP note D2213, will no longer be set.

D2929 MCP - FIXED PORTION MARKER OF LOG RECORDS

The format of word 3 (LOGTYPEX) of a LOG record has been changed to include the length of the FIXED portion of the record. This new field is called LOGFIXEDLENGTHF and is defined to be [47:6]. The old field LOGLENGTHF is now [41:10]; NEWFORMATF has been deleted.

D2936 MCP - "COBOL" FILE ALREADY CLOSED ERRORS

In order to conform to the requirements of COBOL, an attempt by a COBOL program to close a file which is already closed will cause a fatal error on the III.3 release. This warning will be noted on the job summary during the III.1 and III.2 releases, along with an indication of the file and location of the offending CLOSE statement.

D2937 MCP - RESERVE TIME CASES FOR "BSP"

TIME intrinsic calls with parameter values of 24 and 30 through 45 are now reserved for use by BSP. Any calls on a non-BSP system using these values will return a value of 0.

D2938 MCP - GETSTATUS: TEST FOR COMPILER

A new case in GETSTATUS has been added for TYPE=0, SUBTYPE=2, SUBCLASS=2, BIT NO=42. Reply type is REAL. The values are:

0 = Task is not a compiler
1 = Task is a compiler

If the task is not a compiler, a "SOFT-GETSTATUS" error is also returned.

D2942 MCP - VERIFY MEMORY DUMP TAPE

When writing a memory dump to tape, the MCP will now attempt to verify that the tape is assigned to the group. If not, the following message will be displayed:

"GROUP DOES NOT OWN MT<nn>. XMIT AGAIN TO OVERRIDE."

The operator may OK the request or choose another tape.

D2963 MCP - ANOTHER GROUP NOT RESPONDING

The message displayed when another group on a * GLOBAL tm Memory system does not respond is now the following:

NO RESPONSE FROM ANOTHER GROUP (GNS=#). . .

* "GLOBAL Memory" is a trademark of Burroughs Corporation.

D2967 MCP - LOG BOXES FOR JOBS

The MCP now maintains a list of "boxes" that a task/job has occupied during the life of its execution, as well as a list of "boxes" that the task/job could have occupied. These lists are logged in the EOT and EOJ log records in the "EOTBOXNOX" word. EOTBOXNOX contains four 8-bit fields which are the following BOX lists: DATA ALLOWED, CODE ALLOWED, DATA OCCUPIED and CODE OCCUPIED.

JOBFORMATTER (and LOGANALYZER) will now display two new lines in the EOT/EOJ record print out, as follows:

```
DATA ALLOWED IN XXX; CODE ALLOWED IN XXX
DATA OCCUPIED XXX; CODE OCCUPIED XXX
```

where XXX is a list of BOX names (GLOBAL, LOCAL1, LOCAL2, etc.) or "NO BOX" (which is used when the "box" list is empty).

D2969 MCP - IDLE PATTERNS

The MCP often displays a recognizable pattern in the stack registers while waiting in an IDLE operator. These patterns have been changed slightly for the B6700 and extensively for the B6800.

The principal change for the B6700 is that during system initialization, the normal pattern is a big "I" instead of the big "B".

The principal change for the B6800 is to use patterns designed for hex display, rather than the B6700-type patterns which were designed for dot-matrix use of a binary display. (In the II.9 and III.0 releases, only the normal system idle pattern, B6800FFB6800, was designed for hex display. This pattern has also been changed, for greater visibility.)

This note provides an annotated list'of all the idle patterns displayed by the MCP. (There are other instances of the IDLE operator, mostly very early in system initialization, that show no distinct pattern.)

For the most common cases, the B6700 and B6800 have different patterns. B6700 patterns are designed to show one, two or four letters in the A, B, X and Y registers taken as a 12-by-16 dot matrix; those letters are shown here.

B6800 patterns are shown as the hex value actually displayed in each register, except:

(a) "Linenumber" means that the sequence number of the line in
    the MCP where the IDLE occurs is displayed in decimal.
(b) "RCW" means that the return-control-word of the procedure
    that contains the IDLE is displayed in raw form.
(c) Occasionally some variable data are displayed, as noted.

The Y register contents are unspecified unless the B register tag is 2.

B6700	B6800	Significance (and location)
I	A: 2 AAAA1111AAAA X: 2 A00A00A00A00 B: 2 BBBB1111BBBB Y: 2 B00B00B00B00	Normal idle during system initialization. (PAWS)
B	A: 2 A1111111111A X: 2 A00A00A00A00 B: 2 B1111111111B Y: 2 B00B00B00B00	Normal idle. (PAWS)
C M	A: 2 C0C0C0C0C0C0 X: 2 linenumber B: 0 C3D440404040 =	Changing MCP. (CHANGEMCP) "CM " in EBCDIC
D U M P	A: 2 DBDBDBDBDBDB X: 2 linenumber B: 3 RCW	Normal idle during initiation and termination of dump. (LOITER of DUMPBOOTSTRAPPER)
D U M P	A: 2 DBDBDBDBDBDB X: 2 linenumber B: 0 C7C5E3D9C4E2 =	Dump waiting for system I/O to finish. (GETRDS of DUMPBOOTSTRAPPER) "GETRDS" in EBCDIC
D U M P	A: 2 DDDDDDDDDDDD X: 2 linenumber B: 3 RCW	Normal idle during dump. (DPPAUSE of TAPEDUMP)
D E A D	A: 2 DEADDEADDEAD X: 2 linenumber B: 3 RCW	System is hung, displaying "... PLEASE HALT LOAD". (DEFUNCT)
D E A D	A: 2 DEADDEADEAD2 X: 2 linenumber B: 3 RCW	System is hung; some other processor is already in DEFUNCT. (DEFUNCT)
D E A D	A: 2 DDEADEADEADD X: 2 linenumber B: 3 RCW	System hung taking a dump. (SPOUTP OF TAPEDUMP)

The following patterns occur only during the initialization of B6800 multiprocessor systems and never appear on the B6700.

B6800		Significance (and location)
A: 2 CCCCCCCCCCC0		Lead processor waiting to send a
X: 2 linenumber		message to a follower.
B: 2 addressee proc		(MAILACARD of SECONDARYINITIALIZE)
Y: 2 message		
A: 2 CCCCCCCCCCC1		Follower processor waiting for a message
X: 2 linenumber		from the leader.
B: 0 000000000000		(PICKUPMAIL of SECONDARYINITIALIZE)
A: 2 CCCCCCCCCCC2		Lead processor waiting for follower to
X: 2 linenumber		receive a message.
B: 2 addressee proc		(MAILACARD of SECONDARYINITIALIZE)
Y: 2 message		
A: 2 CCCCCCCCCCCF		Lead processor waiting for followers to
X: 2 linenumber		complete initialization. (MERGER)
B: 0 D3D4C5D9C7C5	=	"LMERGE" in EBCDIC

The printer-dump program, which is invoked for system errors early in Halt/Load initialization, uses a single set of patterns for both the B6700 and B6800. The A and B registers are designed for hex display, while the X and Y registers are used as a bit matrix.

X&Y	Registers		Significance (and location)
	A: 2 DEDEDEDEDED1		Normal idle in printer dump.
M D	X: 2 F424F0F88430		(DOIOP of MEMDUMP)
	B: 2 dump address		
	Y: 2 F000F0F112C0		
	A: 2 DEDEDEDEDED0		No printer is on line.
N O	X: 2 F610F0788870		(DUMPITP of MEMDUMP)
	B: 2 D7D9C9D5E3D9	=	"PRINTR" in EBCDIC
	Y: 2 F086F0E111E0		
	A: 2 DEDEDEDEDED2		Printer went not ready.
N R	X: 2 F610F0788870		(DOIOP of MEMDUMP)
	B: 2 D9C5C1C4E840	=	"READY " IN EBCDIC
	Y: 2 F086F0E111E0		
	A: 2 DEDEDEDEDED3		Printer dump has no path to its printer.
N P	X: 2 F610F0F99960		(DOIOP of MEMDUMP)
	B: 2 D7C1E3C84040	=	"PATH " in EBCDIC
	Y: 2 F086F0F00000		
	A: 2 DEDEDEDEDEDF		Printer dump is finished.
FIN	X: 2 F9980F0F610F		(DUMPITP of MEMDUMP)
	B: 2 C6C9D5C9E2C8	=	"FINISH" IN EBCDIC
	Y: 2 F0000F0F086F		

D2971 MCP - SYSTEM MAINTENANCE FOR "B6800" MULTIPROCESSORS

Maintenance for B6800 multiprocessor systems has been implemented. For the III.1 release, multiplexor-related operations may only be directed to one multiplexor during that operation. If no subsystem is specified, maintenance will run in * GLOBAL tm Memory. If operation in local memory is desired, a subsystem specification denoting only the local memory should be used. If an invalid local memory is used (i.e., one which has no direct path to the unit involved), the requested operation will not be performed. An appropriate subsystem may be set up using the MS ODT command (see GENERAL note D2535 for details).

* "GLOBAL Memory" is a trademark of Burroughs Corporation.

MCP

P1026 MCP - "PATHRES"

The unit number displayed in the following message will now be correct:

PK<nn> MPX<m> PATH<p> ERROR ON HOST LOAD.

Error messages from PATHRES will no longer display the phony RD=20001. Instead, an appropriate diagnostic will be given:

(NO IO FINISH) or (NO PATH FOR IO).

P1027 MCP - "WASTEMPATCHECK"

Under certain conditions, disk files that were opened when a checkpoint was taken would be handled incorrectly when the task was restarted. It was possible for the file to be closed with the wrong TITLE. This problem has been corrected.

P1029 MCP - QUEUE ERROR VS. "THEQUE"

Certain "QUEUE ATTRIBUTE ERR" errors were causing the given queue to remain locked. If the program referenced that queue again, the program would hang and could not be DS'ed. This problem has been corrected.

P1053 MCP - "SUSPENDER" VS. "OK" FROM "ODT"

SUSPENDER will no longer hang the system if an OK input from the operator is entered while SUSPENDER is also attempting to resume a suspended task.

P1054 MCP - "IC" DISKPACKS

There were several problems with interchange mode diskpacks on the II.8 and II.9 releases which have been corrected.

P1076 MCP - "FLATREADER" ERRORS

The error checking in FLATREADER has been improved.

P1081 MCP - "NON ANCESTRAL TASKFILE"

In order to prevent illegal memory addressing, it has become necessary to disallow any task from referencing the taskfile of a non-ancestral task; i.e., a task can only reference the taskfile of its father, grandfather, etc. When such illegal references are attempted, the violating task will be DS'ed with the message "NON ANCESTRAL TASKFILE".

P1082 MCP - "LEIBNITZ" VS. OVERLAY

Library Maintenance would occasionally get a dump in BILDAFID when attempting to issue the error message "<filename> NOT ON <source name>". This problem has been corrected.

P1083 MCP - "CATALOG ADD" TAPE FILES

CATALOG ADD= of a multi-file tape volume will now work correctly.

P1085 MCP - "STATUSCHANGE" EVENT BATCHED

Instead of being caused every time any of many operations occur, STATUSCHANGEEVENT will now be caused every 3 seconds when an appropriate operation occurred in the interval.

P1086 MCP - UNLABELED TAPE CORRECTIONS

When attempting, on a cataloging system, to open output unlabeled to tape, the system will insist that the tape not be volumed. If so, the FSVP "REQUIRES NON-VOLUMED MT" will be displayed, so that the operator may find an appropriate tape volume or delete a volume from the volume library. This change is necessary because the serial number will be destroyed on the tape, and thus no longer correspond with any volume library entries. It will not be deleted automatically from the volume library, as policy requires manual intervention to delete a serial number from the volume library. In addition, if the LABELTYPE attribute of a file is changed from an output tape file via the "FA" ODT message, the tape labels will now reflect the new LABELTYPE rather than the original.

P1087 MCP - "MCSREADY" ON "SM" INPUT

An attempt to send a message to an MCS via the "SM" ODT message when the MCS has either not yet attached or just detached itself from the primary queue will now get the reply "INVALID MCS" and will no longer cause an MCP fault.

P1101 MCP - "RESERVE"

RESERVEDISK will no longer corrupt memory links when moving a row of the JOBDESC file.

P1102 MCP - ADD "ETX" TO ALL "ODT" OUTPUT FILE WRITES

If a TD830 ODT is connected to an SLC and a WRITE is done that does not include an ETX in the character string, the output will not appear on the screen. This would occur if the last character written was a control character (from column 0-3 of the EBCDIC chart). This will no longer occur.

This change is required for general use of the TD830 with the SLC.

P1111 MCP - "AUTOPRINT" VS. MULTIPLE "SKIP TO CHANNEL 1"

If two or more "SKIP TO CHANNEL 1" records occurred in a print file, it was possible for AUTOPRINT to incorrectly print a preceding record in the BD file. This has been corrected.

P1112 MCP - "AUTOPRINT" VS. "FM" REPLY

If the operator replies "FM" to AUTOPRINT's "REQ FM:<formname> LP" RSVP to a printer with a different FORMMESSAGE, AUTOPRINT will display "INVALID UNIT LP<nn>" and QT the print file.

P1125 MCP - UNIT LEFT ASSIGNED

The MCP will now check for units which are left assigned to a stack at EOJ even if the DIAGNOSTICS compile-time option is reset.

P1127 MCP - BINARY PACK ADDRESSES

The MCP now supports binary addressing of diskpacks. The controlware loaded into the diskpack controls can now be either binary or decimal. The following restrictions must be observed:

1. All of the controls that can access a particular diskpack must be loaded with the same kind of controlware (binary or decimal).

2. The controlware cannot be changed from binary to decimal or decimal to binary without closing all the packs accessible from the control (or Halt/Loading the system).

3. When a CM is performed on a pack, it must be Halt/Loaded at least once on the same kind of controlware. It then can be Halt/Loaded on either kind of firmware. Because the initial bootstrap put on the pack by the CM routine is not dynamic, after the first Halt/Load a dynamic bootstrap (one that can run on binary or decimal controlware) is automatically written on the pack. In the case of duplicated CM's, at least one Halt/Load must be performed on each backup copy of the MCP before the controlware can be changed.

P1129 MCP - FILE REMOVED MESSAGE

When an attempt is made to REMOVE a cataloged file which has no resident entries, the message "<filename> REMOVED ON ..." will no longer be erroneously displayed.

P1130 MCP - "450/750 LPM" TRAIN PRINTERS

450/750 LPM train printers will now have appropriate train tables loaded for the following trains:

16 char	EBCDIC-3	-TRAINID=1
64 char	EBCDIC-3	-TRAINID=5
64 char	SWEDEN OCR-B	-TRAINID=14

Note: Although the appropriate tables are loaded for TRAINID's 1 and 5, they will show respectively as EBCDIC18 and EBCDIC72 instead of EBCDIC16 and EBCDIC64. This discrepancy will be resolved later.

P1134 MCP - "UR" VS. RESOURCE

UR'ed tape units will no longer be subtracted from the pool of tapes used in starting jobs with RESOURCECHECK set.

P1135 MCP - MEMORY ACCESS ERROR VS. TIME OF DAY

The starting and ending times for memory access error counts will now be logged correctly.

P1154 MCP - TAPE WRITE PARITY AFTER A TAPEMARK

If the ODT option NEWPERETRY were reset and a write parity occurred on the tape record immediately following a tapemark, an extra record would be left on the tape. This has been corrected.

P1180 MCP - "TAPEUNIT" VS. "VOLUNIT"

When purging a tape on a catalog system, a deadly embrace between DIRLOCK(TAPEUNIT) and DIRLOCK(VOLUNIT) was possible. This has been corrected.

P1181 MCP - "II.6" VS. TIMESTAMP

The timestamp will no longer be destroyed when copying BD and BP files to tape.

P1182 MCP - "DBS D3 NOMEM"

DMSOPEN will now hang with an RSVP if insufficient memory is available for the D3 stack. If this RSVP is DS'ed, DMSOPEN will return ERROR 42.

P1184 MCP - "EOT" VS. "TAPEPARITYRETRY"

Mag tape I/O errors at EOT, EOF and BOT will now be logged. TAPEPARITYRETRY has been changed to allow erasing up to 12000 words beyond EOT during recovery from write parity errors on PE tapes. The old limit of 6000 words still applies for lower densities (800 BPI, 556 BPI and 200 BPI).

P1191 MCP - "FINDINPUT" VS. "UNBRF"

An INVALID INDEX in FINDINPUT when opening a file without a kind specification has been corrected.

P1192 MCP - "SWAPPER" VS. CRUNCH

SWAPPER will now quit if the swapdisk file is crunched.

P1193 MCP - "FM" VS. "SU"

Saved (or reserved) printers will not be selected by FINDOUTPUT even if the printer has the correct form.

P1214 MCP - "GIVEBACKDISK"

If a problem occurs complementing the rows of a particular disk file during a family rebuild, the following message will be issued:

    HEADER DAMAGE ON <familyname> <filename>

The file will then be changed to a "bad disk" file, and the row in question will be deleted from the file.

P1216 MCP - "OF" VS. EXCLUSIVE

When trying to copy a disk file that is opened exclusively, the operator can now reply "OF" to the "WAITING FOR FILE" RSVP.

P1217 MCP - FAMILY SUBSTITUTION VS. CATALOG

With "FAMILY DISK=USERPACK OTHERWISE PACK", CATALOG statements; i.e., CATALOG PURGE <filename>, CATALOG ADD <filename>, will reference <filename> on USERPACK (if USERPACK is on-line).

CATALOG statements that specify a serial number will not have family substitution applied.

P1219 MCP - "REMOVE" VS. SECURITY

When a non-privileged user tries to remove or change files in a disk directory not under his usercode, he will get a security error without a display of all of the filenames in that directory.

P1225 MCP - "KANGAROO-INTERCEDE" MECHANISM

The mechanism used by the MCP to interrupt the flow of control in a user program is embodied in the procedures KANGAROO and INTERCEDE, which are used to "get the attention" of a task for such purposes as swapping it out, suspending it, terminating it, or processing a software interrupt. This mechanism has been redesigned to avoid many flaws in the earlier implementation.

The changes should not be visible to running programs, with some minor exceptions: The STOPPOINT task attribute value will be changed only when a fault is detected, or when a task is terminated abnormally; it will no longer be changed every time a swap task is timesliced, for example. The HISTORY task attribute will only be changed at major state transitions in a task, such as initiation, suspension, resumption and termination; for example it will no longer be set to STED (3 in bits 7:8) to timeslice a swaptask.

P1226 MCP - "EVENT" MECHANISM CHANGES

The implementation of EVENTs has been substantially modified. Most of the changes improve efficiency and reliability without affecting the semantics of existing constructs. The following changes may be noticable in some programs.

LIBERATE (or CAUSE after FREE) will awaken all tasks WAITing on the EVENT, but only one PROCUREr: the highest-priority task waiting in a PROCURE is awakened as the new "owner" of the event, while the EVENT remains UNAVAILABLE and all other PROCURErs remain asleep.

WAIT((<time>),...) returns 1 immediately if the time interval is zero (or nearly zero or negative), whether or not any EVENTs designated in the WAIT list have HAPPENED. This is consistent with taking the first "event" already "caused" (from left to right).

WHEN(<time>) is now semantically equivalent to WAIT((<time>)). Thus WHEN(0) is a no-op, and a WHEN statement with a time less than two seconds no longer swaps out a swaptask.

The software interrupt implementation has been generalized to apply to swap tasks; see MCP note D2645.

The limit of 250 parameters (time and EVENT designators) is enforced in the multiple WAIT statement.

P1231 MCP - "235" PACKS VS. MODEL "3 MPX"

Extraneous "read log" I/O's and pathwait problems involving 235 packs and level 2.0 firmware have been corrected.

P1232 MCP - DISK STATUS VS. UNIT MOVED

Problems involved with "moving" packs using level 2.0 firmware have been corrected.

P1252 MCP - "SCR" VS. FIRST ACTION

When SCR got a "first action" result descriptor from a disk pack unit, it was possible for a dump by "IOERROR LEFT ON" and for all I/O's to the pack to hang. This has been corrected.

P1253 MCP - ZIP WITH LARGE ARRAY

ZIP WITH ARRAY will now display the "WORDS REQ" RSVP message if core is needed to make a copy of the array.

P1254 MCP - "DISKLIMIT" VS. "RSVP"

Tasks with DISKLIMIT set were incorrectly being DS'ed if they issued "SEGMENTS REQ" RSVP messages. This problem has been corrected.

P1256 MCP - "COPYIT" VS. "GETUSERDISK"

GETSTATUSERROR #115 (no user disk) will not be returned unless the operator answers "DS" to the "SECT REQ" RSVP message.

P1312 MCP - IMPROVE "AVAILIST" INSERT ALGORITHM

The procedure LINKLISTINSERT and the structure of disk available lists have been modified to reduce overhead for consecutive calls involving monotonically non-decreasing addresses.

P1374 MCP - "MAXIOTIME"

Setting MAXIOTIME will no longer result in MAXIOTIME becoming zero.

P1459 MCP - "MISSINGPROCEDURE" CALL

The call on MISSINGPROCEDURE in HARDWAREINTERRUPT67 was using the wrong field of the result from INTERRUPTEDOPERATOR. This has been corrected.

P1552 MCP - NEW FIRMWARE

The MCP host load procedure PATHRES can now load multisegment firmware files.

P1609 MCP - GUARDFILE

Checkguardfile would not wait for systems resources (i.e., guardfile or pack is not mounted) for available present or resident. This has been corrected.

P1712 MCP - "VOLUME DELETE"

Several problems with the VOLUME DELETE statement have been corrected.

P1713 MCP - LIBRARY MAINTENANCE "BADFILE" VS. "REELSWITCH"

If a COPY without compare did a reel switch and subsequently produced the message "UNEXPECTED TAPE MARK-NOT COPIED", IOREQUEST would blow up. This problem has been corrected.

P1715 MCP - DYING STACK IN "WSSHERIFF"

WSSHERIFF no longer tries to suspend dying stacks.,

P1716 MCP - "PB 2" LEVEL NAME

AUTOBACKUP can now handle two level file names (e.g., BD/0001111) without getting an INVALID OP fault.

P1717 MCP - "MOVE" VS. "LOG"

After MOVEing a pack, I/O error records in the log were incorrectly showing the original unit number. This problem has been corrected.

P1718 MCP - "IOTRACE"

In building the descriptor in SYSTEMSTATUS, V2 is now integerized to use for the length instead of normalized.

P1719 MCP - "ODT ETX" POSITIONING

When clearing or readying an ODT, SYSTEMSTATUS zeroed the density field, not checking to see if it is an ODT and then leave it intact. This has been corrected.

P1720 MCP - CHECKPOINT

When more than one file was reopened, one variable was not reset so that the next file could not be opened properly. This has been corrected.

P1721 MCP - "LINKLISTINSERT" VS. "XTEND"

A conflict between LINKLISTINSERT and XTEND that was causing "DUMP BY FILEHANDLER" has been corrected.

P1722 MCP - READER BUSS PARITY ERROR

A "READ PARITY ERROR" message will now be displayed when an I/O buss parity error (RD=891) occurs while reading from a card reader.

P1723 MCP - "DP" VS. "MPXIII"

A problem in tape dumps occasionally caused superhalts on machines with Model 3 or Model 4 Multiplexors. This has been corrected.

P1724 MCP - "CATBLKSIZEF"

Under certain circumstances, a CATALOG DELETE could cause a corruption in the catalog record. The next Halt/Load would then get the following message in SYSTEM/CATALOG: "OK TO ERASE BAD RECORD". This has been corrected.

P1725 MCP - "GENERATION"

The limit on the file attribute GENERATION is now CATALOGLEVEL (CATALOGLEVEL limit is 7). Formerly, GENERATION was limited to 4.

P1726 MCP - "DUMP" VS. "RD"

Occasionally, an INV PW word interrupt would occur in MCP Seg 5 code after taking a memory dump. This has been corrected.

P1727 MCP - "AUTOBACKUP" PRIORITY

AUTOBACKUP now has a higher priority than MCSs and their offspring tasks.

P1729 MCP - "SIRWOFADDRESS"

The subroutine SIRWOFADDRESS will now return SIRW of true stack and save areas. Hardware interrupt calls this routine to ensure that a valid SIRW is passed to the attributehandler on an uninitialized FIB.

P1730 MCP - CHECKPOINT FILE HEADER TIMESTAMP

File handler will now put the timestamp in the file header of a checkpoint file.

P1731 MCP - MODEL "III" MULTIPLEXOR

Model 3 Multiplexor paths were not being reserved properly by the MCP, as follows:

1. Memory dump was releasing all reserved paths to the Halt/Load disk.

2. PRIMARYINITIALIZE was not setting the pseudobusy FF at Halt/Load time.

These have been corrected.

P1732 MCP - "ELAPSEDTIMELIMIT"

Occasionally, a job with ELAPSEDTIMELIMIT set would be DSed for ELAPSED LIMIT EXCEEDED immediately after it was initiated. This has been corrected.

P1733 MCP - "READALABEL" VS. "EXTMODE"

If a HDR2 tape label has an external mode field that is not 0,2,3,4 or 5, the following warning message will now be issued by READALABEL: "MTnn HDR2 LABEL HAS INVALID EXTERNALMODE". The external mode will be assumed to be 4=EBCDIC (or 3=BCL in the case of 7 track tapes).

P1734 MCP - "FM" VS. SAVED "LP"

1. The FM reply will now override the save status of a printer.

2. If the FM reply is used to an unformed printer, that printer will automatically be saved when the print file is closed.

P1735 MCP - REELSWITCH VS. CATALOG

COPY&CATALOG of a file from tape to disk will now put the correct tape serial numbers in the catalog, even if the file is on a reel other than the first reel.

P1752 MCP - PROGRAMDUMP DIAGNOSTIC

The RCW of the code getting a fault will now be printed rather than a hardwareinterrupt RCW. Also, the sequence number will be printed if possible.

P1753 MCP - ACCEPT "DUMMMP" TAPE ANYTIME

TAPEDUMP will now use a scratch tape with the "DUMMMP" serial number anytime when it is looking for a tape - even after it has given up and asked for a scratch tape. Just make the "DUMMMP" tape ready and it will be found.

P1754 MCP - PROCESSOR BOUND JOBS IN SWAPSPACE

Processor bound jobs that run in swapspace will no longer dominate processor utilization. Non-interactive swap jobs will now have their priority class lowered to the level of a non-swap job. An interactive job in this case is one that does not exceed its timeslice. It is possible for a job to switch back and forth between interactive and non-interactive state; in this case, the priority will be updated accordingly.

P1764 MCP - "B6700/B7700" COMPATIBILITY

The following changes have been made:

1. B7700 commonality patches have been included.

2. Magnetic tape unit type on error listings from magnetic tape confidence is now reported correctly.

3. Result descriptor on error listings in all cases is now reported correctly.

P1782 MCP - GUARDFILE VS. "DS"

Under certain circumstances, if a task waiting for a disk containing a guardfile were DSed, it would come up a second time waiting for a disk containing the guardfile. This has been corrected.

P1783 MCP - "MASKGETAREAF"

A problem with logging I/O errors occurring on Halt/Load disk during the WRITE part of update I/O has been corrected.

P1784 MCP - "CRUNCH" VS. "CP"

The MCP will no longer automatically CRUNCH BACKUP files if the task has taken a checkpoint. This allows rerun to reopen the backup file and write output into it.

P1785 MCP - SOFTWARE INTERRUPT EXECUTION

The operating system now executes software interrupts that are enabled after having been caused while being disabled.

The operating system will now execute software interrupts for all interrupts that are not disabled.

P1786 MCP - GUARDFILE VS. STACKOVERFLOW

Opening guarded disk files would occasionally cause stackoverflow. This problem has been corrected.

P1787 MCP - "TAPEPARITYRETRY" VS. "BLOCKEXIT"

A conflict between TAPEPARITYRETRY and BLOCKEXIT involving direct arrays would occasionally cause a dump by BAD FORGET ADDR. This problem has been corrected.

P1788 MCP - "LOOKFORIT" CONTROL STATE

The MCP procedure LOOKFORIT will now only be executed in control state.

P1789 MCP - READ "LCC"

The response to the PC command on a system with Model III MPXs includes the "halt load count". Shortly after a Halt/Load, if a PC were issued, the system would occasionally get a SCAN BUS PARITY error. This problem has been corrected.

P1790 MCP - INVALID "CRITICAL BLOCK" TERMINATION

BLOCKSEARCH could falsely detect CRITICAL BLOCK EXIT if the offspring task went away after BLOCKEXIT action had begun (as during an EPILOG procedure). The error has been corrected.

P1796 MCP - TIMESTAMP RECALLED OBJECT-JOB MESSAGE

When object-job output is being recalled by DCRECON because a station is being cleared (e.g., for a DSed stack), the messages have not been timestamped properly. This has been corrected.

P1798 MCP - DMS PROGRAMS SWAPPED OUT "FOREVER"

When trying to lock a record or when waiting for a syncpoint, it was possible for a DMS swap job to be swapped out but not be swapped back in (until DSed). This could only happen on a three-processor system, and only occurred rarely. This has been corrected.

P1799 MCP - "PRINTLIMIT" VS. "PROGRAMDUMP"

PRINTLIMIT and PROGRAMDUMP will now work as follows:

1. If a program has a PRINTLIMIT of 0, a PROGRAMDUMP will never be taken.
2. In all other cases, a PROGRAMDUMP will never be terminated or suppressed by exceeding the PRINTLIMIT; however, all lines printed by PROGRAMDUMP will be counted against the PRINTLIMIT. If the PRINTLIMIT is exceeded at the end of a PROGRAMDUMP, the program will be R-DSED.

P1800 MCP - "WAITLIMIT"

The job queue attribute WAITLIMIT will now work as documented In II.9 MCP note D2140.

P1801 MCP - "FIBLOCK" VS. "PROGRAMDUMP"

If a task blew up while using MYSELF.TASKFILE, it was possible for PROGRAMDUMP to hang. This problem has been corrected.

P1862 MCP - "RSNINVALID"

Remote ODT would occasionally hang the CONTROLLER when a remote terminal was shut down. This has been corrected.

P1863 MCP - "DISCSTATUS"

Sometimes the MCP procedure DISCSTATUS, after an I/O error on a test op to a pack, would mark that unit as having "NEW FIRMWARE". This would cause DISCSTATUS to fail for any unit in that minterm group until the next Halt/Load. This has been corrected.

P1864 MCP - "FORGETSPACE" VS. "FORM"

When LH or LT commands were entered specifying a file name, the MCP was not deallocating the array used to hold the name. This has been corrected.

P1865 MCP - LIBRARY MAINTENANCE OPEN ERRORS

Library Maintenance will now issue an open error message when an error occurs in OPEN.

P1876 MCP - "RESERVE AS"

Errors in RESERVE commands with an AS clause would sometimes cause a dump by RESERVE FAULT. This has been corrected.

P1877 MCP - LIBRARY MAINTENANCE REEL SWITCH

Occasionally library maintenance would get a fault in DO code after doing a reel switch during a COPY&COMPARE. This has been corrected.

P1894 MCP - MOVE VS. "RC"

The following message will have the correct unit number, even if the pack has been the subject of a MOVE:

```
    VERIFY REQ TO RC PK<nn>
```

MOVE can be used to fix the "BLASTED" state of a pack. A successful completion of a MOVE will fix a pack's BLASTED status.

P1896 MCP - NEW "GOTOSOLVER"

Several problems involving bad GOTO handling have been corrected. If multiple GOTOs occur (e.g., a GOTO around a block with an EPILOG that calls a procedure that also tries to GOTO around that block), the resulting destination is whichever label PCW was lower in the stack.

P1897 MCP - NO BAD POINTERS IN "WFL"

WFL has been added to the list of compilers whose code files never store "uplevel pointers".

P1898 MCP - COMPILE-AND-GO WITH "SUBSPACES=2"

A compile-and-go with a job or queue SUBSPACES=2 qualification was not running the go part as a swap task. This has been corrected.

P1952 MCP - TAPE OPEN

An ALGOL file declaration of the following form would cause the MCP to get dumps by GETSTATUS UNIT:

```
    FILE T(KIND=TAPE,LABELTYPE=OMITTED,OPEN...)
```

This problem has been corrected.

P2021 MCP - ALLOW STATISTICS TO BE PRINTED

After a PRINT LIMIT EXCEEDED fault, the statistics can now be printed and the DBS will not hang.

P2022 MCP - REBUILD VS. "AD"

Family rebuild and AD (ACCESS DUPLICATE) will conflict if they both occur at the same time. These two procedures now lock each other out such that only one will occur at a time.

The catalog or access structure now contains a TIMESTAMP so backup copies can be checked to ensure they are all current.

P2023 MCP - "SEGO" OF FLATROWS

FLATREADER will now be able to fix SEGO of a flat directory row. Formerly a reply of OK to the RSVP "OK TO ERASE BAD RECORDS" would not correctly fix such an error.

P2024 MCP - "GETSTATUS" VS. "UINFO"

If LABELLED were not TRUE, LEB would not set up so the PACKBASEEU define could not be used. This has been corrected.

P2066 MCP - TRAP FOR BAD DISK "I/O"

If the MCP is compiled with DIAGNOSTICS set and option 43 (DISKCHECK) is turned on, a dump will occur whenever an attempt is made to do logical disk I/O outside the range of the file.

P2068 MCP - "PRESERVER" ARRAY PARAMETER

The array parameter to PRESERVER has been deleted. This avoids an error which could occur if PRESERVER were FORKed and the array was then overlayed.

P2105 MCP - "DUP FAMILY"

When two disk families with the same name cause a "no file" condition, the MCP will now use the message "DUP FAMILY" rather than "NO FAMILY".

GETSTATUS error code 134 will now mean DUP FAMILY.

P2176 MCP - LOG "MPX" SCRATCHPAD MEMORY

The system now does logging of the MPX scratch-pad memory for packs.

P2180 MCP - SYNCHRONIZE CLOCKS

All time-of-day clocks on a B6800 multiprocessor system are now synchronized every 500 seconds.

P2181 MCP - INCREASE STACK SIZE FOR "TAPEDUMP"

The stack size in TAPEDUMP has been increased to avoid overflowing into the buffers. Also, LOSR is now set, so future stack overflows will get a stack overflow interrupt.

P2182 MCP - EXPANDAROW FOR DOPE VECTORS

Expandarow will now work on vectors of descriptors to GETAREAs.

P2198 MCP - "GETAROW" VS. CATALOG

Occasionally, when the CATALOG or ACCESS header was stretched, it would overlay the next header in the FLAT directory. This has been corrected.

P2199 MCP - "PATHRES, FLATREADER, ERRORHANDLER"

The following corrections have been made:

1. PATHRES will now distinguish between NO PATH and a BLOCKED I/O QUEUE when producing error messages.

2. FLATREADER will now produce an error message when it aborts. FLATREADER will now abort if a disk unit is in BLASTed state.

3. ERRORHANDLER will now abort if it is called for a BLASTed unit.

4. PACKMOUNT will no longer allow access to a BLASTed pack.

P2212 MCP - SCRATCHPAD PARITY ERROR

The MCP will now log B6800 SCRATCHPAD PARITY ERROR external interrupts.

P2213 MCP - "BX380" VS. CHANNEL ADDRESSING

The MCP will no longer zero out bits 12 and 13 of result descriptors from 215 BX380 diskpacks.

P2214 MCP - "AB" VS. "EOF"

When a backup file has PROTECTION=PROTECTED set, AUTOBACKUP will now print up to the last block written, even if a Halt/Load prevented the backup file from being closed.

P2215 MCP - "TURNOVERLAYKEY"

TURNOVERLAYKEY will no longer get an invalid address if passed a procedure with the once-only-independent-runner bit on.

P2221 MCP - "STATISTICS"

The MCP STATISTICS option may now be used while SWAPPER is running and no longer requires that the intrinsics be recompiled.

P2222 MCP - "NDL" COMPATIBILITY

The III.0 and III.1 MCPs are now compatible with either III.0 or III.1 NDL files.

P2223 MCP - "FORGETAREA RCW" TRACE

An RCW trace option, triggered by the same bit that controls the FORGETSPACE RCW trace, has been implemented in FORGETAREA. When enabled, the first word after the links in the forgotten area will contain the stack number of the stack that called FORGETAREA followed by as many of the RCWs in the stack as will fit in the space available.

P2224 MCP - IMPROVE HANDLING OF DSING DCPS AND MCSS

If a controlling MCS or DCP abnormally terminates or is DS-ed, programs with open remote files will receive an EOF on their next I/O operation to an affected station. Affected stations will have DISPOSITION=DENIED, and the remote file's POPULATION and STATIONSDENIED counts will be updated appropriately.

P2225 MCP - "FA" OF UNLABELED FILE TO REMOTE

A system dump by "STACK KILLED WITH PCL" will no longer occur when an unlabelled file is equated to a remote file via FA operator input in response to a "NO FILE" condition.

P2226 MCP - DSED "MCS" WAS CAUSING HUNG TASKS

If an MCS abnormally terminates or is DS-ed, programs with remote files will no longer hang in DCCLEAR with a positive datacomm write count (DCWCOUNT), because outstanding WRITEs will have normally finished (via DCIOFINISH).

P2227 MCP - SCHEDULE STATIONS MUST BE ATTACHED

When attempting to assign a schedule station to a remote file, the station must have been previously attached by an MCS; otherwise, an OPEN error will occur, and "SCHEDULE STATION MUST BE ATTACHED" will be displayed.

P2228 MCP - "DCP" FAULT WITH SPACE QUEUE LOCKED

If the DCP abnormally stops (e.g., faults) with DCPSPACEHEAD[*] locked, the MCP will no longer hang in an infinite loop.

P2229 MCP - "AUTODC" AND STATION ATTACH

When attaching a station by name (DCWRITE type = 01), AUTODC (option 12) will be checked prior to invoking FIREOFFDCP.

P2230 MCP - "PC" LOCK VS. "NIFSEARCH"

PROCESSCHANGELOCK is released in MUTATE to avoid a deadlock with DIRLOCK during FINDINPUT in NIFSEARCH.

P2231 MCP - CORRECT PRIMARY QUEUE NUMBER

The procedure DCINITIATEMCS will now return the correct primary queue number.

P2232 MCP - TANKED OUTPUT MESSAGES

The appropriate bit in a station list will be used to specify that tanked output messages were in progress at file CLOSE time for a remote file with asynchronous tanking.

P2233 MCP - "DCALGOL EPILOG" PROCEDURES

An error has been corrected in the handling of DCALGOL EPILOG procedures at BLOCKEXIT time. An INVALID INDEX could occur in the case where the block containing the EPILOG was a formal procedure.

P2235 MCP - REBUILD VS. "DONTWAIT"

GETFAMILY of FILEHANDLER will now return NOTFOUND if a disk being sought is rebuilding (in ERRORHANDLER) if the caller set FHDONTWAITF; consequently, CANDE will not hang so often waiting for ERRORHANDLER.

P2241 MCP - HANDLE LONG "AX" INTO SHORT ARRAY

It is no longer possible to cause a fault in MCP code by entering an AX input with a message text longer than the program's array.

P2242 MCP - "PROGRAMDUMP" CORRECTIONS

Two small long-standing errors in PROGRAMDUMP have been corrected, as follows:

1. The TFFFF and TFFFOFFF of an RCW are now printed independently as "TRUE" and "OCCUPIED", respectively. Formerly, the former was printed only if the latter were set.
2. DCALGOL MESSAGEs and MESSAGE ARRAY elements will no longer be mistaken for special objects like FILEs.

P2265 MCP - "BUFFERS" ATTRIBUTE FOR OPEN FILE

The BUFFERS attribute was returning zero for open files rather than the number of buffers allocated. This problem has been corrected.

P2266 MCP - SECURITY VS. BACKUP

Non-privileged users would occasionally get security violations when entering printer or punch backup files in the directory. The security violation would occur at close time if the file had been removed after it was opened and before it was closed. This problem has been corrected.

P2267 MCP - LOCKING IN "MIXREQUEST"

It is no longer possible to take the ELSE part of MIXREQUEST's case and exit with PROCESSCHANGELOCK procured.

P2268 MCP - "CL" VS. "AB"

Occasionally, after entering CL a printer or punch AUTOBACKUP would hang. Normally, this condition could be circumvented by CLing the unit again. AUTOBACKUP has been changed so that it will usually CL the unit automatically until the I/O queue is clear.

P2269 MCP - "DCALGOL EPILOG" REVISITED

BLOCKEXIT handling of DCALGOL EPILOG procedures has been corrected yet again. The policy remains that an epilog in non-MCP code will not be processed if there is stack overflow pending, unless the code has been marked non-DSABLE. The latter distinction is now based on the segment description for the EPILOG procedure.

P2273 MCP - "RESERVE" LOOP

RESERVE would often loop after issuing the message "WAIT ON TEMP FILE <filename>". This problem has been corrected.

P2274 MCP - "CLOSE" ERRORS FATAL

"IO ERR IN CLOSE", which is generated in CLOSE, is now a fatal error.

Also, trying to write to disk past the EOF is now a fatal error.

Errors which now become fatal are the following:

1. Closing COBOL files not opened
2. Errors rewinding a tape at CLOSE
3. Errors writing tape marks at CLOSE
4. Errors writing tape labels
5. User not explicitly closing file requiring user labels
6. Error reading user labels at CLOSE
7. Writing past the EOF when flushing buffers at CLOSE

P2275 MCP - "COPY =" MANY FILES

When Library Maintenance attempted to copy a very large number of files, it would occasionally abort. This was caused by overflow of a 16-bit field. These fields are now 20 bits.

P2286 MCP - "GETSTATUS UNIT" VS. "DISKPACKSEARCH"

After a GETSTATUS UNIT memory dump, DISKPACKSEARCH will now be liberated if it was owned by the dumping stack.

P2288 MCP - "MCP" CONVERSION TO "NEWP"

1 INTRODUCTION
- ------------

This note describes the differences between NEWP and ESPOL that resulted in changes to the III.1 MCP symbolic. Full documentation of NEWP features used in the MCP is contained in "D2804 -- Implementation of NEWP".

2 BLOCKS AND PROCEDURES
- ------ --- ----------

"Cheap" Blocks

Blocks which are not procedures are not entered with an ENTR operator in NEWP. Items which are declared in the block are simply added to the stack at the current lex level. Since this implementation requires less execution-time overhead, NEWP blocks are called "cheap" blocks.

Because cheap blocks are not entered via an ENTR operator, it is illegal to leave a block by performing an EXIT or RETURN. Cheap blocks do not change the addressing environment; thus, the lex level does not change and there is no MSCW for the block.

Segmentation

In NEWP, by default, procedures declared at lex levels less than or equal to the specified SEGMENTLEVEL are in separate segments, regardless of whether or not local variables are declared, and blocks are not in separate segments. Default segmentation can be overridden by using block directions, as described in the NEWP D-Note.

The ESPOL constructs BEGINSEGMENT, ENDSEGMENT, RESIDENT, CONTROL, and SAVE (for procedures) do not exist in NEWP. Instead, the SEGMENT and CONTROLSTATE block directions can be used to perform the same functions (e.g. an outer block "SAVE" procedure is specified in NEWP as [SEGMENT=5, CONTROLSTATE]).

3 POINTERS

As described in the NEWP D-Note, pointers can be declared as EBCDIC, ASCII, BCL, or HEX. If no size is specified, EBCDIC is assumed. Strings are assumed to be EBCDIC if no size specification is given. In NEWP, POINTER(<array name>) generates an EBCDIC pointer, not a word pointer as it would in ESPOL. POINTER(<array name>,0) generates a word pointer, as does the construct POINTER(MEMORY[n]). Word pointers are syntactically interchangeable with EBCDIC pointers.

NEWP does not allow pointer and string sizes to be mismatched. For example, a REPLACE statement is not allowed to replace an EBCDIC pointer by a hexadecimal string.

The CORRECTLY clause, allowed on REPLACE statements in ESPOL, is not implemented in NEWP.

4 ARRAYS

Arrays in ESPOL can be used as arrays (in the ALGOL sense), as array references, and as descriptors. In NEWP, these concepts are separated; arrays are the same as ALGOL arrays, array references are declared explicitly, and descriptor-related operations are performed using variables of a new data type called DESCRIPTOR. DESCRIPTOR variables are described in the NEWP D-Note.

Array bounds in NEWP differ from ESPOL in the following ways:

- Both the upper and the lower bounds of the array must be specified in an array declaration.
- The array bound "*" is allowed only for formal parameters.
- If "*" is specified as the lower bound of an array formal parameter, the lower bound is passed to the procedure as a hidden parameter (as in ALGOL).
- Address-equated arrays must have "0" as the lower bound, not "*".

In NEWP, arrays can be passed as parameters by reference only. NEWP call-by-reference arrays have the same semantics as ESPOL call-by-value arrays, as they both pass a copy descriptor. ESPOL call-by-name arrays are not available in NEWP; call-by-reference DESCRIPTORs are used instead.

5 OTHER CHANGES

This section describes additional differences in syntax and semantics between NEWP and ESPOL. The following table lists some ESPOL constructs that are not available in NEWP and how they were converted when they were encountered in the MCP symbolic:

ESPOL Construct	NEWP Conversion
DABS, DNABS	None to convert.
DUPLICATE	Deleted by recoding.
ENTIER	Replaced by a DEFINE.
EVENT(x)	Recoded using "EVENT AT".
EXCHANGE(x)	Deleted by recoding.
EXCHANGE(x,a)	DEFINEd to be READLOCK.
FIRSTWORD(<event>)	Deleted by recoding.
IIO	DEFINEd using SCANOUT.
LOAD	Recoded using "DESCRIPTOR AT".
M	DEFINEd to be MEMORY.
NAME(x)	Recoded using LEXOFFSET or "REFERENCE TO".
NAME <type>	Recoded using "<type> VIA".

OCRX	Replaced by a DEFINE.
REWIND statement	Replaced by CLOSE(f,RETAIN).
SCALELEFT, SCALERIGHT, SCALERIGHTS, SCALERIGHTT	None to convert.
SCALERIGHTF	Replaced by PACKDECIMAL.
SPACE Statement	Replaced by READ(f[SPACE n]).
STACK	Declared as an array at (0,2).
STFF	Recoded or defined using "REFERENCE TO".
TOGGLE	Deleted by recoding.
TOPOFSTACK	Deleted by recoding.
TOUCH	Replaced by a DEFINE.
UNPACKU	Deleted by recoding.
Software interrupts	Deleted by recoding.
[<exp>] := <expl>	Recoded using "<type> VIA <exp>"
@<octal constant>	Replaced by 3"<octal constant>".
GO TO <word variable>	Replaced by procedure calls.
GO TO <designational exp>	Expanded into statements.
Initialized arrays	None to convert.
Initialized variables	Statements added to initialize variables.
Dynamic procedures e.g. PROCEDURE (P)	Recoded as normal procedures.
Space-saver procedures e.g. PROCEDURE P [10]	Recoded as dummy procedures of the desired length.
FIELDs	Redeclared as DEFINEs.
LAYOUTs	Redeclared using concatenation.
PICTUREs	Deleted by recoding.
QUEUEs	Redeclared items using CONSTANT declarations.
Compile-time variables e.g. DEFINE X:=3	Redeclared as CONSTANTs or deleted by recoding.
Address-equation of the form "<id> = (-0,n)"	-0 replaced by *.
Undeclared labels ("seems to be a label")	LABEL declarations added.
Vectormode	Recoded using VECTORCHECKSUM.
Word expressions as <translate table>s in REPLACE statements	Recoded by declaring table.
IF <ptr>=<arith exp> FOR <n>	Recoded using 48"...".
ON statements	Recoded using ON declarations.
ONES intrinsic used as a statement	Replaced by DAWDLE intrinsic.

The following ESPOL constructs are implemented differently in NEWP:

- The ESPOL intrinsic MYSELF is WHOAMI in NEWP; MYSELF in NEWP is the intrinsic for accessing the task for the executing program, as in ALGOL.

- Multi-character operators are treated as single tokens in NEWP. The characters cannot be separated by embedded blanks, by card boundaries, or by parametric defines. However, one blank is permitted between the "=" and the "*" in the update replacement operator (":=*").

- Assignments to REGISTERS may not use the update replacement operator (":=*") and may not be embedded within expressions.

- Accidental entries are never generated in NEWP. Thus, expressions may not be passed to call-by-reference parameters.

- NEWP makes no distinction between SAVE ARRAYs declared in the outer block and those declared locally. That is, NEWP will not pre-allocate global SAVE ARRAYs in the D[0] stack image.

- Parameters associated with formal procedures must be specified.

- Implicit type transfers ("coercions") of WORD variables to other data types have been restricted in NEWP. Specifically, WORD variables are not coerced to ARRAYs, PROCEDUREs, or POINTERs. Coercions between WORDs and BOOLEANs are not allowed in assignment operations.

- REAL variables appearing in concatenations are not coerced to BOOLEANs.

- Certain D[0] variables that are initialized by ESPOL are not initialized by NEWP, such as the empty descriptor at (0,92) and the empty tag 6 word at (0,90).

- The STOP intrinsic allows two additional parameters in NEWP; these parameters specify locations into which the contents of the A and B registers are to be stored following the HALT operator.

- A BEGIN/END pair is required around all procedure bodies.

P2300 MCP - "MPX" VS. TAPE PARITY

Occasionally, the I/O queue for READALABEL would become garbled, apparently caused by the MPX setting the RDERROR field to 0 even though IOATTENTION was set in the IOCW. This problem has been corrected; HARDWAREINTERRUPT now sets bits 0 and 1 of RDERROR if IOERROR is running.

P2301 MCP - "AB" BUFFER SIZE

The buffer size of AUTOBACKUP has been doubled, thus increasing printout speed.

P2303 MCP - "SYSTEMSTATUS" STRING PROTECT

Previously, it was possible to put the system in a control state loop by calling SYSTEMSTATUS CASE 10 with too short an array. This problem has been corrected.

P2304 MCP - RESTORE "LOSR" AFTER DUMP

DUMPBOOTSTRAPPER now restores the LOSR register after a non-fatal dump.

P2305 MCP - "STICKY" MEMORY

A conflict between two different stacks in GETSPACE occasionally occurred when STICKY memory was being used. This would lead to INV PW in GETSPACE. This problem has been corrected.

P2306 MCP - "USERPARITYBIT"

For certain I/O errors (such as DESCRIPTOR ERROR, MEMORY PARITY ERROR, etc.), IOERROR was not setting the USERPARITYBIT, thus causing such I/O errors to go undetected by most programs. This problem has been corrected.

P2307 MCP - "VERIFYFAMILY" VS. "BU"

Occasionally, VERIFYFAMILY would incorrectly set up a disk family that was using duplicate directories. This problem has been corrected.

P2310 MCP - CLASS COUNT "UFLO"

MEMDUMPs by CLASS COUNT UFLO would occasionally occur when running RESERVE on HPT DISK. This problem has been corrected.

P2311 MCP - "EOJ" VS. "USTKASSIGNED"

Occasionally, after a program does a CLOSE, ASSIGNSTKV, READALABEL will get a dump by UNOWNED I/O. This problem has been corrected.

P2312 MCP - "SWAPPER" GETS "DIV BY ZERO"

SWAPPER no longer gets a "DIV BY ZERO" error after entering "<mix> AX MINTIME".

P2336 MCP - CLEAR PARAMETERS IF TASK INITIATION FAILS

An error has been corrected which caused garbage parameters to be passed if a TASK variable had previously been used in an unsuccessful task initiation.

P2358 MCP - "OVERLAYCF"

On a B6800 multiprocessor system, a problem with the handling of LINKB occasionally caused faults in LOSEOLAYSPACE (for one) because the wrong stack number was used. This problem has been corrected.

P2364 MCP - "SIB" ENVIRONMENT

If a SIB environment in the codefile ended on a row boundary, DMSOPEN would read the environment incorrectly. This problem has been corrected.

P2391 MCP - CATALOG REMOVE

Occasionally, after a catalog (or access) has been removed, STARTSYSTEM would get a dump by VERIFY ERR 9. This problem has been corrected.

P2392 MCP - SWAP OUT WAITING FOR SCHEDULE READ

Swapjobs that read data from schedule stations are now swapped out if the data is not available.

P2396 MCP - "CPRESTART EOJ"

Occasionally, when a checkpoint restart job was aborted, the program name in the COMPLETED mix picture would be corrupted. Sometimes this corruption would cause a fault in the CONTROLLER. This problem has been corrected.

P2419 MCP - IMPROVED "I/O" ERROR MESSAGE

When a program tries to write on a crunched file using different blocking factors from than those of the physical file, the error message "INCOMPATIBLE BLOCKING" is now given; previously, the message "UNKNOWN ERROR" was given.

P2422 MCP - ARRAY PARAMETERS BY VALUE

In certain cases, arrays are being passed "by value" to external procedures. The circumstances are the following:

1. The only parameter is a single-precision array passed with lower bound=0.

2 a. The passing program is WFL code, or

2 b. The passing program is an MCS and the receiving program is a task of a session.

"by value" means that the data are copied into a new array, whose main descriptor resides within the receiving task.

The current specifications of this feature are temporary for the III.1 release, and will be changed. This implementation serves two purposes:

1. Utility programs and compilers run as clients of jobs or of CANDE will no longer need to be searched for descriptors belonging to the parent process.

2. Utility programs and compilers may be proessed as an external host, even though parameter-passking in general is not supported in III.1 Host Services.

P2423 MCP - MEMORY PRIORITY FOR SWAPJOBS

The memory priority algorithms now work properly for swapjobs.

P2425 MCP - SIZE CHECK OF USER ARRAY

In GETUSER, a test is made to determine if the array "userarray" is large enough to hold the information. The test was wrong; it has been corrected.

P2427 MCP - "MPX" PATHS

The OL display for disks and tapes will now show which multiplexors have paths to a minterm group. Formerly, this display was accurate only for systems with Model 3 and Model 4 multiplexors.

P2428 MCP - UNINITIATED "I/O" DUMP

A dump by uninitiated I/O would sometimes occur when a tape drive was cleared. This problem has been corrected.

P2429 MCP - ILLEGAL "I/O"

The MCP traps attempts to initiate I/O from one local processor when the buffer is in local memory of another processor. A DIAGNOSTICS MCP has been taking a dump by ILLEGAL I/O. In the forthcoming III.1 MCP, such an occurrance would indicate a system failure. However, in initial-release TC systems, the situation may arise from violation of the configuration rules, such as using a non-exchanged mag tape or attempting direct printer output with the printer on the wrong side of the system and the direct arrays forced to local (see MCP note D2911).

Because user error has proved more likely than system error in this diagnostic, the TC initial-release software has been changed. In addition to cancelling the I/O, as before, the MCP will now terminate the offending process with the message

    ILLEGAL IO: UNIT NOT EXCHANGED TO LOCAL PROCESSOR

The dump by ILLEGAL IO is now produced only if the run-time option DIAGNOSTICS is set; it is independent of the compile-time $DIAGNOSTICS option.

A task DSed for ILLEGAL I/O will have HISTORYCAUSE=SYSTEMCAUSE=5 and HISTORYREASON=ILLEGALIOV=5.

P2430 MCP - NEW HIGH ORDER PRIORITY SCHEME

The priority of a process stack for acquiring a processor must often be raised above the normal visible priority range 0-99. The exalted priority is sometimes permanent, as for certain independent runners, MCS programs, and the like; for other stacks, the priority must be raised temporarily, as during a period that MCP global locks are held. (If a process cannot reacquire a processor to unlock some resource, many other processes can be starved of that resource.)

The MCP keeps processor priority in a three-part field. The declared visible priority is in the middle, with fine priority (biasing towards I/O-bound tasks) at the low end and the special-case priorities in the high end. The management of this high-order field has been changed.

Since the II.8 release, the MCP has initialized this field, called PRIORCLASSF, to various values according to the type of process. Then the field was used as a counter, bumped up and down as the code PROCUREd and LIBERATEd events, etc. In the new scheme, the field is divided into several subfields which may be used orthogonally. The function of lock counting has been moved to a new word in the base of the stack; one bit in the priority field records the non-zero state of that counter. The field as a whole is called PRIORBIASF; the subfields are as follows:

HIGHESTBIASF	[47:1]	Set for invisible independent runners.
LOCKBIASF	[46:1]	Set if LOCKCOUNT is non-0; used for soft locks and similar resources.
IRBIASF	[45:2]	Used for special independent runners, both visible and invisible. Among the former are AUTOBACKUP and DCCONTROL.
MCSBIASF	[42:2]	Used for special user programs: an MCS gets 2, a CPed non-MCS gets 1.
DSEDBIASF	[39:1]	Used to accelerate a DSed process off the system, freeing any resources it has used.
INTERACTIVEBIASF	[38:1]	Set for a swap task currently in an interactive (not time-sliced) mode.
JOBBIASF	[37:1]	Set for all WFL JOB stacks.

(The fields have been described in the positive sense; the data are actually kept in ones-complemented form to utilize the LLLU operator.)

In an additional change, any time PRESENCEBIT locks a descriptor, the LOCKCOUNT is incremented as though an event had been PROCURED.

P2431 MCP - "PROGRAMDUMP" BE WARY IN DEFUNCT BLOCK

PROGRAMDUMP attempts to determine the owning stack for any copy descriptors encountered. If the area is owned by some other stack and the ARRAY option is selected, the contents are dumped for arrays referenced by unindexed copy descriptors. This useful feature can be dangerous if BLOCKEXIT has begun to tear down the block in question, since the copies may be of moms which formerly resided in this block and have been deallocated.

BLOCKEXIT now marks a block as DEFUNCT, by setting a bit in the software control word. When PROGRAMDUMP finds such a mark at the top of a block, it displays that fact on the SCW line. Then it ignores any copy descriptors within that block which do not point into the stack or segment dictionary directly. That is, the copy descriptors are shown, but their analysis and possible area dumping is curtailed.

PROGRAMDUMP now also notices the NOGOPASTF bit in special SCWs used within the MCP. Such an SCW will now be identified as such, rather than subjected to a garbage "analysis."

P2433 MCP - CORRECT "TERMINATE" VS. "BLOCKEXIT" RACE

A timing race has been eliminated between TERMINATE (tearing down a stack and cleaning up its TASK variable) and BLOCKEXIT (discarding the TASK in question).

P2435 MCP - REVISE "GETAREA" POOL

The management of the pool of spare GETAREA rows has been revised, as follows:

1. The basic number of spare rows is now 3, with a window of 3. Thus new rows will be acquired when the number of empty rows falls below 3; empty rows will be discarded when their number exceeds 6.

2. The basic number (but not the window) is increased by one for each 34 swaptasks.

P2436 MCP - STATISTICS IMPROVEMENTS

A number of improvements have been made in the reliability and completeness of the MCP statistics mechanism.

Attempts to SV or RY a processor will be rejected while statistcs is running.

P2438 MCP - LIBRARY PROGRAM DUMP OPTIONS

The following new program dump options have been added as a result of the Libraries implementation:

LIBRARIES	Dump all library stacks
PRIVATELIBRARIES	Dump only Private Library stacks
SIBS	Dump SIBs

The existing DBS option dumps all DBS and SIBs.

See GENERAL note D2535, "Revised ODT Commands", for syntax and semantics.

P2448 MCP - "CL READALABEL"

When the operator clears a tape in READALABEL, the MCP will now issue the message:

```
        MT#  READALABEL DSED
```

P2485 MCP - "GIVEBACKHDR" CORRECTION

If files happen to overlap on a pack and one (or both) is (are) a code file(s), we eventually get an "I/O TO UNIT ZERO" dump because DOCTOR used a "bad disk header" out of which the unit number was derived while trying to bring in some code.

Now, when overlap is detected, an overlap message is given (only one per file) on the ODT naming the pack, from segment, to segment, and file name. In addition, if the FILEKIND of the file is "dangerous", it is marked as an "XDISKFILE" (i.e., bad disk) and an ODT message is issued telling the operator that the file has been so marked.

P2488 MCP - AUTOMATIC SHORTENING OF "SEARCHQ" ENTRIES

When the number of SEARCHQ entries in the system gets too large, SWAPPER will swap in swapjobs in an attempt to reduce that number.

P2513 MCP - "CATALOG PURGE"

CATALOG PURGE and CATALOG DELETE were not clearing the CATALOGED bit in the resident disk file header. This meant the MCP often restored the cataloged status of a disk file the next time it used the header. This problem has been corrected.

P2514 MCP - "PURGIT RD" MESSAGE

An I/O error while purging a 7-track tape would cause an INVALID INDEX in PURGIT. This problem has been corrected.

P2519 MCP - NEW TRAPS

Two new traps have been installed in the MCP. Both are of low execution cost.

Any fault in or of NORMALEOJ causes a fatal dump by FAULT IN EOJ.

Any fault termination in D0-relative (MCP) code while the faulting stack has a soft lock now causes a non-fatal dump by MCP FAULT LOCKED.

P2530 MCP - "CHECKPOINT" VS. "OLAYHEADER"

Occasionally, CHECKPOINT would blow up counting the number of allocated rows in the overlay file. This problem has been corrected.

P2531 MCP - "CHANGEMCP" "INVALID INDEX"

CHANGEMCP will no longer terminate with a fault if the MCP being CMed to has no unallocated rows in the code file header.

P2532 MCP - "CHECKPOINTFILE" VS. ABORT

In certain cases when a checkpoint was aborted, the checkpoint file would accidentally be left opened. This problem has been corrected.

P2533 MCP - DISALLOW CRUNCHED "USERDATAFILE"

The code in the MCP to disallow crunched SYSTEM/USERDATAFILE was incorrect; it has been corrected.

P2534 MCP - "CHECKPOINT" VS. "JEDGARHOOVER"

A program run under a non-privileged usercode that took a checkpoint with a temporary disk file open would get a security error if it rewound the file and then read the file. This problem has been corrected.

P2536 MCP - "PATH MARKED OFFLINE"

The PATH MARKED OFFLINE message for disk packs has been changed to an RSVP, so the operator will be sure to notice it. A path will not be automatically marked offline until it receives three consecutive 809 result descriptors.

P2537 MCP - REEL SWITCH VS. PARITY ERROR

COPY (no compare) to a tape would occasionally hang the system. This would happen if EOT occurred immediately after the operator replied OK to RECOPY REQUIRED. This problem has been corrected.

P2538 MCP - SPURIOUS "PACK IN USE" MESSAGE

On a B6800 Multiprocessor system, a spurious PK<nn> IN USE message would occasionally be generated, as well as a spurious SECTORS REQUIRED message. These problems have been corrected.

P2545 MCP - "GETAREA" VS. LOCKS

A possible system deadlock involving GETAREA, ETERNALIR and RSVPLOCK has been corrected.

P2546 MCP - "DD" VS. "CM"

COPYDIR no longer sets DKBACKUPF before completion. PACKMOUNT RSVP for JOBCODE no longer attempts to display the file title.

P2552 MCP - CARD READER SECURED STATUS

If a card reader is secured using the SR command and is later cleared using the CL command, it loses its secured status. This problem has been corrected.

P2566 MCP - PUT "HOSTNAME" AND "GROUPID" IN PROGRAMDUMP

HOSTNAME and GROUPID now appar in the programdump heading.

P2584 MCP - AVOID EXTRA MOM ON "B6800" NEGATIVE INDEX

A logic error in the B6800 processor has the following effect:

> If a Mom descriptor is indexed by a negative value, a replica of the mom is left on the stack (just below the interrupt MSCW).

If the fault causes a PROGRAMDUMP which causes a stack stretch, the extra mom descriptor results in a dump by "BAD STRETCH MOM".

Pending correction of the hardware, the MCP will avoid the bad effects by setting the copy bit of any such replica mom.

P2588 MCP - INITIALIZING "FIB" IN DIFFERENT STACK

In certain cases, the MCP was improperly initializing FIBs declared in other stacks. The problem has been corrected.

P2590 MCP - "CHECKPOINT RESTART"

CHECKPOINT RESTART has been corrected, as follows:

1. It will now correctly reestablish software interrupts.

2. Rerun will not do family substitution when looking for backup punch and printer files.

3. Absent copy descriptors are now handled correctly.

P2595 MCP - SETTING FAMILY NAME AFTER ASSIGNMENT

Familyname will now give an attribute error if an attempt is made to set it after the file has been assigned. Previously, the assignment did not give an error; however, the familyname was not altered.

P2597 MCP - "PROGRAMDUMP" HEADING

The heading of the program dump has been reformatted and augmented for improved readability. Among the improvements are the following:

1. Correctly identify the machine (B6700, B6800)

2. Fully display longer program and MCP names

3. Show subsystem (local processor or Global) on a B6800 Multiprocessor system

P2599 MCP - "PROGRAMDUMP" HANG

On a diagnostic MCP, a program that had OPTION=FILES set and then got an attribute error in the file description could hang in a non-DSable state. This problem has been corrected.

P2624 MCP - PROTECTION FROM TAG-7 INTERRUPTS

HARDWAREINTERRUPT now prevents any tag-7 interrupt parameter from confusing the algorithm that searches for PCW or memory link words to determine the owner stack of a memory address.

P2625 MCP - PRINTER DUMP LOOP

Occasionally, the raw printer dump program would go into a loop doing futile outputs to the printer, which never moves. The problem occurred only when the paper had been left aligned at the end-of-page line; this has been corrected.

P8629 MCP - "PD" TO MISSING FAMILY

PD commands directed to missing families will now get the response "NO FAMILY".

P9117 MCP - "235" PACKS

The MCP and LOADER can now handle the new diskpack controlware releases and 235 diskpacks.

P9143 MCP - FATAL STACK OVERFLOW

System fatal stack overflow dumps after a user stack is stretched to the limit will no longer occur (only for non-SWAPPER cases).

P9145 MCP - "CRUNCH" VS. "COPY"

When files are copied to disk, the rows will be allocated in order. This will avoid checkerboarding when several crunched files are copied to disk.

P9146 MCP - RESERVE LOCKS

The MCP RESERVE procedure is no longer subject to a deadly embrace involving USERDISKLOCK and RESERVINGCOMPLETE.

P9149 MCP - "BADROW"

If FLATREADER encounters a row that is allocated on a FAMILY member that is not in the family (DFAMILYINDEX too large), the row will be deallocated.

P9150 MCP - "FILEKIND"

PD on a cataloging system will now show the FILEKIND of each non-resident disk file.

P9151 MCP - "CDONLY"

The message "OPEN WITH CDONLY SET:<fileid>" has been changed to "NO FILE (CR) BUT CDONLY IS SET:<fileid>". This message is issued when a task is DS'ed for a no file on a card input file with the CDONLY system option set.

P9153 MCP - CHANGE TITLE OF CATALOGED FILE

Changing the title of a cataloged file will only affect the resident file. The catalog information under the original title will remain unchanged, except that the file is now non-resident. Users may obtain the file under its original title by specifying the original title and the USECATALOG file attribute. .The resident file whose title has been changed will have no backup copies until some user action is taken to produce them.

P9156 MCP - TASK ATTRIBUTE ERRORS

Several cases where task attribute errors displayed no message have been eliminated. In general, a user task attribute error will display a message and DS the task. Certain exceptions to this rule remain for MCS's.

Also, some improvements to the task error messages have been made; e.g., the "INCORRECT SYNTAX" message will name the task attribute invoked.

P9158 MCP - "WFL" "USER" STATEMENT IN SUBROUTINE

A USER statement in a WFL subroutine which is PROCESS'ed will now properly change the usercode of the processed subroutine.

Previously, a USER statement in a PROCESS'ed WFL subroutine was ignored.

P9160 MCP - "CANDE" "WFL" COMMAND WITH NEW "WFL" SYNTAX

A WFL RUN statement in a CANDE WFL command would abort for "FAULT OR BAD TASK ATTRIBUTE" if the WFL command used the II.9 BEGIN JOB syntax. This problem has been corrected.

P9161 MCP - "SCHEDCORE"

SCHEDCORE will now be counted for invisible jobs as well as visible stacks.

P9162 MCP - "NO MEM" IN SWAPSPACE

The core initially allocated for arrays, code, etc., is no longer protected by memory priority. This should alleviate problems with large amounts of array initialization.

P9164 MCP - "SWAPPER" QUEUE COUNTS AND AGING

The values for CORENQ count and SLICEQ count are now correctly maintained. The "aging" algorithm for tasks in the core queue has also been modified.

P9166 MCP - PROTECTED TAPES

The protected bit will be turned off during TAPEPARITYRETRY action on output tape files. This will prevent a tapemark from being written on the tape if a Halt/Load occurs before TAPEPARITYRETRY finishes. (Since TAPEPARITYRETRY will backup over good records during its recovery operations, this prevents those good records from being lost if a Halt/Load occurs before the tape is spaced forward again.)

P9169 MCP - PROCESSOR PRIORITY

Some problems in the processor priority algorithms, which occasionally resulted in a task stack being given excessive internal priority, have been corrected.

P9170 MCP - "WAITLIMIT" CORRECTION

An error in the implementation of WAITLIMIT task attribute has been corrected. It was possible to improperly arm the limit by executing a multiple-event WAIT statement, and then be DS'ed, even though the statement did not actually wait.

P9171 MCP - STACKSEARCH CONTROL

An error in the parameter analysis for PROCESS's and COROUTINE's has been corrected. If the procedure being initiated had a formal name POINTER parameter preceding a formal PROCEDURE, there was a possibility of inadequate stacksearching for uplevel pointer assignments.

P9172 MCP - "BADDISK" VS. DIRECTORY

When a disk pack is RC'ed, the directory will be allocated so it will not conflict with any BADDISK files that were created when the pack was IV'ed.

P9173 MCP - "LEIBNITZ EOTINHDR"

When copying empty disk files to tape, Library Maintenance would not do a reel switch when EOT was encountered. This problem has been corrected.

P9176 MCP - "SWAPPER" QT'S "BACKUP"

BACKUP will no longer be QT'ed by SWAPPER when running in swapspace.

P9209 MCP - HALT/LOAD WITH "64" MODS OF MEMORY

It is now possible to Halt/Load a B6700/B6800 with 64 mods of memory on line.

P9218 MCP - "BD" FILES VS "EOF=0"

Empty backup disk files (BD and BP files) will now automatically be removed from disk by AUTOBACKUP.

P9219 MCP - "DBS" VS "CHECKPOINT"

INITIATE was changed to allow running programs out of DBS stacks. This change introduced a problem into TASK-RESTARTER when restarting a CHECKPOINT. This problem has been corrected.

P9220 MCP - "PLANT MESSAGE" VS "AUTOBACKUP"

If a printer (or punch) were CL'ed using AUTOBACKUP and subsequently went not ready, PLANT MESSAGE would blow up indexing stack 0. This problem has been corrected.

P9221 MCP - "DBSSTACK" VS "GETSTATUS"

When GETSTATUS was called with TYPE = 0 and SUBTYPE = 2 for a DBS stack, it would get a dump by GETSTATUS MIX. This problem has been corrected.

P9222 MCP - "CP" VS "SWAPPER"

1. Jobs in subspaces can now take checkpoints.

2. When a checkpoint is RERUN, the restart will occur outside of the subspaces.

P9223 MCP - "SUBSPACES" VS "USERCODE"

If SUBSPACES=2 and the code file is under the user's own usercode, SUBSPACES will be set to 3. If SUBSPACES=2 and the code file is under another usercode or does not have a usercode, SUBSPACES will be set to 1.

P9224 MCP - "VERIFYFAMILY" VS "HDRVECTORLOCK"

If VERIFYFAMILY gets a fault while processing the system directory on a disk, it will liberate HDRVECTORLOCK before aborting.

P9225 MCP - "DESCRIPTOR" ERROR ON PACKS

Under certain conditions if a disk pack on a MODEL I multiplexor got a DESCRIPTOR error, it was possible for the MCP to loop. This problem has been corrected.

P9226 MCP - STICKY "MEM" RECURSIVE "GETSPACE"

It was possible that GETSPACE could destroy a RESIDENT array or procedure when moving it. This problem has been corrected.

P9227 MCP - "HDREXPAND"

If a disk file header were stretched while DISKFILEHEADERS was being stretched, a BAD MOM ADDRESS dump would occur. This problem has been corrected.

P9229 MCP - "ANABOLISM" VS "MAKEJOBFILE"

It was possible to hang the system when starting a SCR, RES, RET or XD task. If a disk pack I/O error occurred at the same time INITIATE (who was called by ANABOLISM) called MAKEJOBFILE, ANABOLISM could hang waiting for a lock. This problem has been corrected.

P9230 MCP - "HISTORY" VS "PROGRAMDUMP"

Faults occurring in PROGRAMDUMP will no longer change the HISTORY word of the task.

P9231 MCP - USER USES OF "TD830" ODT'S FOR "ODT" FILES

Logical record handling has been modified so that all writes to ODT files terminate with ETX's. If the user program has not provided an ETX, an attempt is made to append on to the record. If the record is already maxrecsize long, the last character of the user's data is overwritten. This is necessary for compatibility with TD830 ODT's. In addition, the sequence ESC "("; (i.e., escape left parenthesis) is replaced by blank.

Similar changes have been made to the DCALGOL intrinsic WRITESPO.

P9240 MCP - TAPE PARITY RETRY VS. TAPEMARK

If an I/O error occurred on a WRITE to the first record after a tapemark on a tape, the bad record was not being erased. This problem has been corrected.

P9245 MCP - REGULATE "STACKINFO" LOCKING

A new hard lock, SILOCK, is used to protect STACKINFO words when changing field values. This is an innermost lock. SILOCK has replaced SEARCHLOCK in the SEARCH/STACKSEARCH interface.

P9246 MCP - PAPERTAPE VS REELSWITCH

The paper tape punch and the reel switching will now work correctly.

P9269 MCP - DENSITY VS. PRINTLABEL

Creating an unlabeled PE tape no longer causes ADM PRINTLABEL to mark as 800bpi on the label.

P9270 MCP - "DBS PROGRAMDUMP"

It was possible for PROGRAMDUMP to get a SEG ARRAY error when printing a dump for a task using a DBS. This problem has been corrected.

P9271 MCP - "CP" VS. BACKUP TAPE

When restarting a checkpoint job that was using a printer backup tape, it was possible to get a SEG ARRAY error. This problem has been corrected.

P9276 MCP - DELETE ZERO SERIAL NUMBER

The VOLUME DELETE statement will now delete volumes with serial number zero from the volume library and the pack access structure.

P9285 MCP - TASK RESTRUCTURING

The task and base-of-stack structures have been reorganized; many items were moved from the stack to the task.

The changes were required for B6800 multiprocessor systems. They will also improve control and monitoring of swap jobs. Most interaction that previously was rejected for swap tasks is now processed normally.

P9286 MCP - "DM6700" VESTIGES REMOVED

The vestigial code and external declarations for DM6700 have been removed from the MCP.

P9288 MCP - "EOJ" VARIABLES

The temporary variables used in NORMALEOJ have been renamed and all unnecessary address equation has been eliminated. No logic was changed.

P9304 MCP - "IC" SPECS

A revised format for interchange disk packs has been defined; the system now conforms to the revised version.

NETWORK DEFINITION LANGUAGE

D2697 NDL - DATACOM NETWORK CONFIGURATION REPORT

A brief tabular Network Status Report is now available at compile time through the NDL Processor.

The report contains the following fields and information:

```
DCP number and exchange status
Cluster number
LINE information
     Address (DCP number: cluster number: LINE number)
     STATION(s) information
     Special characters or address characters if specified
     LSN
     Terminal name (17 characters)
Connection medium (direct or modem name, 17 characters)
LINE speed, character size, adapter mode
MCS name (30 characters)
```

Enabling the new dollar option NETWORK will effect the printing of the network configuration report. A report will be generated even if the dollar option SYNTAX is set or if the dollar option LIST is RESET.

Examples:

```
$SET LIST SYNTAX NETWORK
$SET NETWORK
```

D2702 NDL - USE PROPER PROCESSOR "ID" IN HEADINGS

The NDL compiler will now use the TIME(23) intrinsic to determine the proper machine identifier to be used in the heading of compiler listings.

D2703 NDL - IMPLEMENT APPLICATION NUMBER VARIABLE

A new variable, APPLICATION, has been added to the NDL language to allow interrogation of the application number within requests. The following is a summary of the variable's characteristics:

```
APPLICATION
Interrogate, CTR, 8
```

This read-only station-oriented variable contains the value supplied in the most recent SET APPLICATION NUMBER DCWRITE (type 38). It is limited to integers in the range of 1 to 6. When the station is initialized, APPLICATION is given a value of 1.

Example:

```
IF APPLICATION=6 THEN . . .
```

The terminal REQUEST statement has been extended to allow a list of application numbers to appear within the brackets. The new syntax is as follows:

```
-- REQUEST --------------------------------- = ----------------------->
            |                             |
            |     |<------- , -------|    |
            |     |                  |    |
            |- [ ---<integer range>--- ] -|

  |<----------------- , -----------------|
  |                                      |
>-----/1\-<request id>-- : -- RECEIVE ------ . ----------------------|
     |                                   |
     |-/1\-<request id>-- : -- TRANSMIT -|


<integer range>

-- 1 to 6 --|
```

This new syntax specifies that all of the application numbers will refer to the same pair of requests. All statements which were valid under the old syntax are still valid. Syntax checks will be performed to ensure that all application numbers are within the range of 1 to 6, and that no application number is specified more than once.

Examples:

```
    REQUEST[1,5,6]=WRITETTY:TRANSMIT,READTTY:RECEIVE.
    REQUEST[3,4]=WRITESCREEN:TRANSMIT,READSCREEN:RECEIVE.
```

D2772 NDL - IMPLEMENT VARIABLE SIZE STATION TABLE

In the past, a station table base size was either 6 words (without DCP sequencing) or 8 words (with DCP sequencing). Certain message-oriented constructs for autonomous DCP systems (like auditing and filemode) also needed extra words in the station table. Rather than always making the station table base the maximum possible size, each NDL compilation will now determine what constructs were used that require extra words in the station table and will assign and allocate only those locations needed; thus, with this change, the station base may be any (fixed) size from 6 to 10 words in length. The message-oriented MCP has been similarly modified to allow it to make use of this new information and to initialize a datacom system whose stations tables have either the old or new sizes.

NETWORK DEFINITION LANGUAGE

P2085 NDL - "FILE" <FAMILY> STATEMENT

The NDL FILE <family> statement has been corrected to work as specified in the NDL Manual, Form No. 5001522, when a previously-declared <file identifier> is included in the list of stations comprising the family. In the past, a family declared in that manner was not correctly defined and could cause an INVALID INDEX in the MCP procedure DOCTOR when the file was opened.

P2325 NDL - FURTHER AUXILIARY LOGIC CORRECTIONS

Two problems dealing with the selective placement of portions of request sets or line controls into auxiliary (main) memory have been corrected in the NDL compiler. Both problems concerned the usage of $SET AUXLOGIC/$RESET AUXLOGIC dollar cards in NDL source programs in which the DCP AUXILIARY statement was not used. If the AUXLOGIC dollar card option was changed only between request sets and line controls, its occurrence might be ignored. Furthermore, when the AUXLOGIC options was changed within a request set or line control, it sometimes would not take effect until after the following statement, rather than before. Both of these problem areas have been corrected; some additional code optimization in the switching between local and auxiliary memory has also been performed.

P2404 NDL - ELIMINATE LOOPING ON SEVERE ERRORS

In many sections of the NDL compiler, it was possible to get into an endless loop if end-of-file was encountered while attempting to discard tokens after a severe error until a comma or period was found. End-of-file has been made a termination condition in all these cases also. In addition, it was discovered that if a token were requested after the one which showed end-of-file was presented, the last line of input was being rescanned. This problem has also been eliminated.

P2405 NDL - BASIC CONTROL/ACII POSITION CHECKING

In Message Oriented datacom, one basic control unit (to which four front-end controls may be attached) takes up four Cluster II positions. In a network which contains both front-end controls (attached to a basic control) and Cluster II controls, the NDL must ensure that no attempt is made to configure a ntwork that has both front-end controls and Cluster II controls within the same group of four cluster positions. The previous syntax checking in the NDL compiler was incorrect, such that some erroneous errors would be reported, while other actual error situations went undetected. That problem has been corrected.

P2406 NDL - AUTONOMOUS "DCP" BUFFERS FOR STATIONLESS LINES

In autonomous operation of a Message Oriented datacom system, the DCP must have message buffers allocated in autonomous memory for all lines. For lines which have stations assigned to them, the NDL compiler can automatically allocate message buffers of the proper size, or the line MSGSPACE statement can be used to control the allocation explicitly. The compiler gives an error if a line has no buffers allocated by either method. This error checking was incorrect in three cases, all involving lines which had no stations assigned. If the line was an auxiliary line of a full-duplex pair, by default the buffer allocation of the primary line is now used. For an ACU line, the compiler allocates the proper sie buffer automatically. Only for a normal stationless lines whose MAXSTATIONS value is greater than zero does the compiler now require that buffers be allocated (with the MSGSPACE statement); in this case, the user must make the buffer size as large as that required by whatever station may later be added to the line.

P2444 NDL - ERRONEOUS SYNTAX ERROR ON DEFINE "ID" USAGE

If the first Request or Control in SOURCENDL referenced a global define and did not declare any local defines, it would receive an erroneous syntax error. This problem, which is now corrected, was due to an incorrect context level which caused the text of the define identifier not to be expanded.

A different problem, which was previously corrected in NDL patch 30.007, occurred when a locally declared define identifier was immediately used as the first token of the first statement of a Request or Control. While similar in cause, these two problems were not related. They have both been corrected.

P2469 NDL - SUPPRESS "IF B1 OR B2" CODE WHEN FROZEN

The NDL compiler sets a variable called FROZEN and suppresses code generation following an unconditional GOTO or terminating construct until a label definition is seen. However, if an IF statement of the form "IF B1 OR B2 THEN..." was used while FROZEN was set, an attempt was being made to fixup the branch condition for code that was suppressed, leading to possible corruption of the previous macro. This spurious code fixup has been eliminated.

P2557 NDL - EXCHANGED "DCP" VS. AUXILIARY LOGIC

A problem could occur in DCPPROGEN compiling a source NDL program that contained exchanged DCPs, one with local memory and auxiliary logic, the other with main memory only. Various label errors, faults and/or looping in the compiler might result. This condition has been resolved.

NEW IMPLEMENTATION LANGUAGE

D2804 NEWP - IMPLEMENTATION OF "NEWP"

1 INTRODUCTION

NEWP is an ALGOL-like language that has replaced ESPOL as the language in which the MCP is written. Although NEWP has been designed as a general-purpose language, the III.1 NEWP compiler is specifically aimed at aiding the conversion of the MCP symbolic from ESPOL to NEWP. This document describes only the features of NEWP that appear in the III.1 MCP symbolic. Subsequent releases of the NEWP compiler will be accompanied by documentation describing additional NEWP constructs.

The III.1 NEWP compiler will produce only non-executable (MCP) code files. Some features are described as "temporary". These features were implemented to facilitate the MCP conversion and will be deleted from the language as soon as it is practical to do so.

Aspects of NEWP which are not specified in this document are identical in operation to ALGOL as described in the B7000/B6000 ALGOL Language Reference Manual (Form No. 5001639). The NEWP implementation also includes the interface to libraries as it is implemented in ALGOL.

For details regarding those differences between ESPOL and NEWP that affected the MCP conversion, see "P2288 -- MCP Conversion to NEWP".

2 LANGUAGE COMPONENTS

All reserved words which have meaning only in limited contexts are type 3 reserved words. This type includes file attributes, task attributes, CLOSE options, PROGRAMDUMP options, carriage control parameters in I/O statements, and fault names. When context indicates that a type 3 reserved word is appropriate, the compiler first looks for a reserved word. If one is not found, defines are expanded and the compiler looks again for a reserved word.

3 PROGRAM STRUCTURE

In NEWP, lex levels are determined by procedure nesting, not block nesting as in ALGOL. Thus, a procedure which is not a block changes the lex level, while a block which is not a procedure does not.

Segmentation is based upon lex levels. By default, a new segment is assigned to each procedure declared at lex level 15 or below (whether or not local variables are declared). The lex level at which segmentation occurs can be changed by using the SEGMENTLEVEL block direction. In addition, automatic segmentation can be overridden on a block-by-block basis by use of the SEGMENT block direction.

Note: The problem that exists in ALGOL where variables declared at a low lex level with a high offset may become inaccessible at higher lex levels has been alleviated in NEWP.

* See also
8.2 Block Directions

4 DECLARATIONS

In addition to the differences and additions described in the following subsections, NEWP declarations differ from ALGOL declarations in the following aspect:

- DEFINE invocations are expanded within FORMAT declarations and in-line formats when an item is not recognized as a format specifier.

4.1 CONSTANT DECLARATION

<constant declaration>

```
                |<----------------- , -----------------|
                |                                      |
-- CONSTANT ---<constant-id>-----------------------------------|
                               |                       |
                               |- = --<constant value>-|
```

```
     <constant value>

     --<constant arithmetic expression>-----------------------|
```

Example:

```
CONSTANT
   TASKMSCW,              % AUTOMATICALLY ASSIGNED 0
   TASKPARAMS,            % 1
   CODEHEADERINDEX = 0,   % STARTS AGAIN AT 0
   RUNNINGCOUNT,          % 1
   CODELINKS,             % 2
   MARKER = CODELINKS,    % 2 ALSO
   COMPILERINFO,          % 3
   TOFFSET = MARKER+5,    % 7
   NEXTWORD;              % 8
```

The CONSTANT declaration can be used to declare arithmetic constants. It is particularly useful for declaring a series of integer constants where each value is to be one greater than the previous value.

If a <constant-id> appears with a <constant value>, then the identifier is associated with the specified value. If no <constant value> is given, the compiler assigns the identifier the value of the previous identifier plus one (or zero, if it is the first identifier in the list).

The <constant value> is evaluated in the context of the CONSTANT declaration (unlike defines, for which the text is expanded in the context of the invocation).

4.2 LABEL DECLARATION

```
-- LABEL -------------<label identifier list>-----------------|
           |           |
           |- [BAD] -|
```

Example:

```
   LABEL [BAD] ENDITALL, ERROREXIT;
```

In NEWP, a "bad go to" is a GO TO statement that branches out of the segment or procedure in which the GO TO statement appears. A label which is the object of a "bad go to" must be declared in a label declaration which includes the "[BAD]" syntax.

4.3 MODULE DECLARATION

<module-declaration>

```
-- MODULE --<module-id>-- ; ----------------------------------->
                              |                |
                              |-<module-head>-|

>- BEGIN --<module-id>-- ; ----------------------------------->

>-<module-body>---------------------------------------------->

>- END --<module-id>-- ; -------------------------------------|
```

<module-head>

```
                            |<------ ; -----|
                            |               |
---------------------------<declaration>----------------------|
  |                      |
  |-<export-list>-- ; -|
<module-body>

  |<-------- ; -------|
  |                   |
------------------------------------------------------------------|
     |                |
     |-<declaration>-|
     |                |
     |-<import-list>-|


<export-list>

-- EXPORT --<id-list>-------------------------------------------|


<import-list>

             |<------------------ , -------------------|
             |                                         |
-- IMPORT --- FROM --<module-id>-- ( --<id-list>-- ) ---------|


<id-list>

  |<----- , -----|
  |              |
----<identifier>-----------------------------------------------|
```

The MODULE declaration allows logically-related declarations to be grouped together. Items declared within a module are "protected" in the sense that they are not visible to other modules unless specified in an EXPORT list; even if exported, items which are IMPORTed into another module may not be changed by that module.

EXPORTed identifiers must be declared in the <module-head>. Procedures to be EXPORTed are declared in the <module-head> as they would be declared anywhere else, except that the <procedure body> must be EXTERNAL, NULL, or FORWARD. If the <procedure body> is FORWARD, the procedure must be fully declared in the <module-body>.

A module that requires access to an EXPORTed identifier must specify that identifier in an IMPORT list. Access to IMPORTed variables is always read-only.

"MODULE <module-id>", "BEGIN <module-id>", and "END <module-id>" must be the first tokens on the card images on which they appear.

TEMPORARY -- As a temporary convenience, exported identifiers are implicitly imported into all modules; thus, IMPORT lists are temporarily optional.

TEMPORARY -- Temporarily, modules that import identifiers are allowed both read and write access to those identifiers.

Example of Modules:

```
BEGIN
...
MODULE X;
  EXPORT RX, PX;
  REAL RX;
  PROCEDURE PX(B);
    VALUE B; REAL B;
    FORWARD;
BEGIN X;
  IMPORT FROM Y (RY, PY);
  PROCEDURE PX(B);
    VALUE B; REAL B;
    BEGIN
    RX := RY;
    PY;
    END;
END X;
...
MODULE Y;
  EXPORT RY, PY;
  REAL RY;
  PROCEDURE PY;
    FORWARD;
BEGIN Y;
  REAL LOCALR;
  IMPORT FROM X(RX);
  PROCEDURE PY;
    BEGIN
    LOCALR := RX;
    END;
END Y;
...
END.
```

4.4 ON DECLARATION

<on declaration>

```
-- ON --<fault list>-- , --<statement>------------------------|
```

Example:

```
ON ANYFAULT,
   BEGIN
   WRITE(TTYFILE,<"FAULT IN FIRSTPROC">);
   GO TO ERRLABEL;
   END;
```

The ON declaration provides a mechanism for handling faults. When a fault named in the fault list occurs, control is transferred to the fault-handling statement appearing in the ON declaration. In order to resume normal execution, the programmer must perform a GO TO (except in the case of EXPONENT UNDERFLOW).

If a GO TO is not performed by the programmer, the system searches down the program's execution stack for another enabled fault-handling declaration. If none is found, the program is DSed (except if the fault was EXPONENT UNDERFLOW, in which case the program will continue processing with the operand that caused the underflow set to zero).

The <fault name>s and <fault number>s in NEWP are identical to those in ALGOL, except for the NEWP faults MEMORYFAIL1 (23), PRIVILEGEDINSTRUCTION (24), and PROCESSORINTERNAL (20), which are available when the compiler option B7000 is set. Also, if B7000 is set, the following faults are not included in ANYFAULT, although they may be specified as individual <fault name>s: LOOP, MEMORYPARITY, INVALIDADDRESS, SCANPARITY, INVALIDPROGRAMWORD, MEMORYFAIL1, PROCESSORINTERNAL.

4.5 POINTER DECLARATION

```
--------------- POINTER --<pointer identifier list>-----------|
  |          |
  |- ASCII --|
  |          |
  |- BCL ----|
  |          |
  |- EBCDIC -|
  |          |
  |- HEX ----|
```

Example:

```
   EBCDIC POINTER PTRIN, PTROUT;
```

Pointers may be declared with a size specification (e.g. HEX). If the character size is not specified, it defaults to EBCDIC.

Syntax errors are given for pointer and string size mismatches. For example, if PTR is declared as an EBCDIC pointer, the following statement will cause a syntax error:

```
         REPLACE PTR BY 4"FF00";   % SHOULD BE 48"FF00"
```

4.6 PROCEDURE DECLARATION

Procedures in NEWP are similar to procedures in ALGOL, except 1) every <procedure body> must be delimited by a BEGIN/END pair and 2) parameters may not be passed by name.

Parameters may be passed as call-by-value or call-by-reference (call-by-name parameters and "thunks" are not implemented in NEWP). The default is call-by-reference. To pass a parameter as call-by-value, the <value part> must appear in the procedure heading.

Actual parameters passed to call-by-reference formal parameters must generate address references. Constants and arithmetic expressions do not generate address references. However, conditional and case expressions are allowed if each branch generates an address reference. For parameters passed by reference, the types of the actual and formal parameters must agree (e.g. a variable of type REAL cannot be passed by reference to a formal parameter of type DOUBLE or INTEGER). Procedures of type POINTER may be declared in NEWP.

The syntax for specifying procedures as formal parameters differs between NEWP and ALGOL. NEWP does not support run-time parameter checking; therefore, all parameters of formal procedures must be specified. The following diagram describes the syntax for the <specification> of a procedure which is a formal parameter (refer to the ALGOL Language Reference Manual page 4-55):

```
--<procedure type>-- PROCEDURE --<identifier>----------------->

>-<formal parameter part>-- ; -- FORMAL ---------------------|
```

5 STATEMENTS

The following list describes the differences between NEWP statements and ALGOL statements. Page numbers refer to applicable documentation in the ALGOL Language Reference Manual.

- A <partial word part> may appear on the left-hand side of an update replacement (page 5-4), as shown in the following example: X.[5:3] :=* + 1.

- The function of the DEALLOCATE statement in ALGOL (page 5-31) is performed by the RESIZE statement in NEWP (page 5-91), as follows: RESIZE(<array row>,DEALLOCATE).

- When an <arithmetic expression> appears as the <source part> in a REPLACE statement and no FOR clause is specified, exactly 48 or 96 bits of data (depending on whether the <arithmetic expression> is single-precision or double-precision) are transferred as characters of the size determined by the <destination part> (page 5-78). This implementation differs from ALGOL, where either 6 or 8 characters would be transferred, depending on the value of the BCL dollarcard option.

- The SET and RESET statements for events have been renamed SETEVENT and RESETEVENT (pages 5-98 and 5-90).

- The function performed by the REWIND statement in ALGOL (page 5-92) is performed by the CLOSE statement in NEWP (page 5-25), as follows: CLOSE(<file designator>,RETAIN).

- Branching into THRU loops (page 5-109) is not permitted. Branching within a THRU loop is allowed, provided the label is declared within the loop.

- The <time> specified in a WAIT statement (page 5-114) need not appear within its own set of parentheses and need not appear first in the <wait parameter list>. If a <time> appears in the parameter list, it will be evaluated before any event parameters.

- The function of the WHEN statement (page 5-117) in ALGOL is performed by the WAIT statement (page 5-114) in NEWP, as follows: WAIT(<arithmetic expression>).

6 INTRINSICS

This section describes the intrinsics available in NEWP; page numbers refer to the ALGOL Language Reference Manual.

The following intrinsics are implemented in NEWP as they are in ALGOL (pages 6-19 to 6-30):

ABS, AVAILABLE
BOOLEAN
COMBINEPPBS, COMPILETIME
DAND, DEQV, DNOT, DOR, DOUBLE
FIRSTONE, FIRSTWORD
HAPPENED
INTEGER, INTEGERT
LISTLOOKUP
MASKSEARCH, MAX, MIN
NABS
OFFSET, ONES
POINTER, POTC, POTH, POTL
READLOCK, REAL(<ae>), REAL(<be>), REAL(<pe>,<ae>)
SECONDWORD, SINGLE, SIZE(<array designator>)
TIME

The function provided by the LINENUMBER intrinsic in ALGOL (page 6-24) is available as COMPILETIME(23) in NEWP.

The operation performed by the SCALERIGHTF intrinsic in ALGOL (page 6-26) is performed by the PACKDECIMAL intrinsic in NEWP.

The POINTER intrinsic in NEWP (page 6-30) allows the <character size> to be specified as 0, creating a word pointer which is considered as EBCDIC when pointer and string sizes are being matched.

The DAWDLE intrinsic in NEWP is an untyped intrinsic that takes an integer parameter. DAWDLE is used in the MCP to delay without accessing memory. The integer parameter specifies the number of Count Binary Ones (CBON) operators that are to be executed to effect the delay.

* See also
4.5 POINTER Declaration

UNSAFE MODE

Some constructs which the MCP requires in order to perform hardware-related functions are considered unsafe for general use. NEWP requires the MCP programmer to specify when and which unsafe constructs are to be used. This specification is made on a block-by-block basis through the use of the UNSAFE block direction. Programs that use the UNSAFE block direction are marked as non-executable.

* See also
8.2 Block Directions

7.1 DECLARATIONS

There are two additional data types available in UNSAFE mode, DESCRIPTOR and WORD.

DESCRIPTOR
In UNSAFE(DESCRIPTOR) mode, DESCRIPTOR variables are allowed. The DESCRIPTOR data type is described in the following subsection.

WORD
In UNSAFE(WORD) mode, WORD variables are allowed, with basically the same syntax and semantics as in ESPOL. Implicit type transfers ("coercions") between type WORD and other data types is more restricted in NEWP than in ESPOL. In particular, WORD/BOOLEAN coercion is disallowed in assignment operations.

In UNSAFE(MISC) mode, address equation as it exists in ESPOL is allowed in NEWP. Also, a <procedure body> can be specified as NULL, as in ESPOL, but only if the procedure has been address-equated. Array declarations may include the SAVE specification, as in ESPOL.

7.1.1 DESCRIPTOR Data Type

Variables of type DESCRIPTOR are used to store unindexed mom or copy descriptors. Simple variables of type DESCRIPTOR are declared in a DESCRIPTOR declaration (similar to a REAL, INTEGER, or BOOLEAN declaration). Arrays, procedures, and formal parameters may also be specified as type DESCRIPTOR. Variables of type DESCRIPTOR are initialized to "uninitialized operand".

When values of type DESCRIPTOR are evaluated, copy bit action occurs if the target is a data descriptor.

The type transfer function DESCRIPTOR can be applied to both WORD variables and array rows. Implicit type transfers ("coercions") allow array references to be assigned from DESCRIPTOR values and DESCRIPTORs to be assigned from array references or array rows. The same coercions are applied between formal and actual parameters.

TEMPORARY -- Temporarily, WORDs and DESCRIPTORs are mutually coerced.

7.2 STATEMENTS

The FORK statement is allowed in UNSAFE(FORK) mode, with similar syntax and semantics to ESPOL. A new required parameter, <box number>, must appear as the first parameter in the parameter list.

The OVERWRITE option on the REPLACE statement (as in ESPOL) is allowed in UNSAFE(MISC) mode.

In UNSAFE(REGISTERS) mode, the D-register override clause ("@ [<exp>]") is allowed following EXIT, RETURN, and <procedure statement>s (e.g. "RETURN @ [TASKADDR]").

WAIT and WAITANDRESET statements may have option lists in UNSAFE(MISC) mode. This syntax is described in the following subsection.

TEMPORARY -- ALLOW and DISALLOW statements are temporarily allowed in NEWP as they are in ESPOL.

7.2.1 WAIT Statement

```
---- WAIT ----------------------------------------------------->
   |                |  |                          |
   |- WAITANDRESET -|  |- [ --<option list>-- ] -|

>- ( --<wait list>-- ) ---------------------------------------|
```

<option list>

```
   |<---------- , ---------|
   |                       |
------/1\--- DSABLE -------------------------------------------|
      |       |               |
      |       |- NOTDSABLE -|
      |                       |
      |-/1\--- SWAPNOW ---|
              |               |
              |- DELAYSWAP -|
              |               |
              |- NOSWAP ----|
```

In UNSAFE(MISC) mode, the WAIT and WAITANDRESET statements may include an <option list> to specify whether or not the waiting process can be DSed or swapped out while waiting, as follows:

DSABLE
WAIT and WAITANDRESET with [DSABLE] perform the same functions that DSWAIT and DSWAITANDRESET do in ESPOL. The process will not wait if it is already DSed or will not continue to wait if it is externally DSed while waiting. The value returned by the WAIT function (if it occurs in an arithmetic expression) will be zero in either case.

NOTDSABLE
The process will wait even if it is or becomes DSed.

DELAYSWAP
The process will be swapped out if it waits longer than the maximum swap wait, defined in the MCP as MAXSWAPWAIT.

SWAPNOW
The process is swapped out as soon as the wait starts.

NOSWAP
The process must not be swapped out while waiting.

If an <option list> is not given, default values are assigned according to the type o1 process. System processes (D[0] and pseudo-D[0] relative code) are NOTDSABLE and NOSWAP. User processes default to DELAYSWAP and DSABLE (if not already DSed; if the process has already been DSed, it will wait).

7.3 INTRINSICS

The following list describes the intrinsic identifiers recognized in UNSAFE mode. The UNSAFE category for each intrinsic is included in parentheses.

AT (REFERENCE)
The syntax "<type> AT <location designator>" allows the item at the location specified by the <location designator> to be referenced or assigned to as if it had been declared of the specified <type>. <Type> must be one of the following simple types: BOOLEAN, INTEGER, REAL, DOUBLE, POINTER, EVENT, DESCRIPTOR, FILE, TASK, WORD. <Location designator> can be a data item, a partially subscripted array (the descriptor is accessed), or a procedure identifier (the PCW for the procedure is accessed). If the AT syntax is used in an expression, the item is fetched from the location in the manner appropriate for a value of the target <type>.

For example, if D is a variable of type DESCRIPTOR, the syntax "WORD(D)" causes D to be fetched as a DESCRIPTOR, and copy bit action is performed. The syntax "WORD AT D", on the other hand, causes the item at D to be fetched as a WORD, and no copy bit action is performed.

Note that if the specified <location> is a formal parameter, "<type> AT <location>" references the actual parameter, not the local SIRW.

BUZZ (MISC)
The BUZZ intrinsic is implemented in NEWP as in ESPOL, with the additional restriction that BUZZ can be used only in CONTROLSTATE blocks.

TEMPORARY -- BUZZ is temporarily allowed for NORMALSTATE blocks.

BUZZ47 (MISC)
BUZZ47 performs the same function that BUZZ does, except that bit 47 of the parameter is tested instead of bit 0. BUZZ47 is valid only when the B7000 compiler option is set.

DESCRIPTOR (DESCRIPTOR)
DESCRIPTOR is a type transfer function that can be applied to a WORD variable or an array row.

EVAL (MACHINEOPS)
TEMPORARY -- EVAL(<param>) in NEWP performs the same function that EVAL(NAME(<param>)) performs in ESPOL. EVAL is a temporary feature.

EXIT (MACHINEOPS)
EXIT is implemented in NEWP as it is in ESPOL.

FAILREGISTER (MACHINEOPS)
FAILREGISTER(<integer>) generates the Fetch Main Memory Fail Register (FMFR) operator for B7000 Series machines. The <integer> parameter is the memory module number. FAILREGISTER is valid only if the B7000 compiler option is set.

FMMRREADLOCK (MACHINEOPS)
FMMRREADLOCK performs the same function that READLOCK does, except that the compiler emits the B7000 operator Fetch Main Memory Reference (FMMR) prior to emitting the Read With Lock (RDLK) operator. FMMRREADLOCK is valid only if the B7000 compiler option is set.

HEYOU (MACHINEOPS)
HEYOU is implemented in NEWP as it is in ESPOL.

IGNOREPARITY (MACHINEOPS)
IGNOREPARITY is an untyped procedure that generates the B7000 operator Ignore Parity (IGPR). IGNOREPARITY is valid only if the B7000 compiler option is set.

INTERRUPTCHANNEL (MACHINEOPS)
INTERRUPTCHANNEL(<real>) is an untyped procedure that generates the B7000 operator Interrupt Channel (INCN), where the <real> parameter is a mask indicating which channel is to be interrupted. INTERRUPTCHANNEL is valid only if the B7000 compiler option is set.

LEXOFFSET (MISC)
LEXOFFSET(<identifier>), where <identifier> represents a data item with an address couple, returns the MSCW-relative offset of that data item. For example, if X is declared at (1,9), LEXOFFSET(X) is 9.

MAKEPCW (MACHINEOPS)
TEMPORARY -- MAKEPCW is implemented in NEWP as it is in ESPOL. MAKEPCW is a temporary feature.

MEMORY (MEMORY)
MEMORY is implemented in NEWP as it is in ESPOL, except that M is not recognized as a synonym.

MOVESTACK (MACHINEOPS)
MOVESTACK is implemented in NEWP as it is in ESPOL.

PAUSE (MACHINEOPS)
PAUSE generates the IDLE operator. If two parameters are specified, the two values are loaded into the A and B registers before the IDLE operator is executed and are deleted afterwards.

POINTER(<word exp>) (WORD)
POINTER(<word exp>) performs a type transfer between type WORD and type POINTER.

REFERENCE TO (REFERENCE and WORD)
The function represented by the syntax "REFERENCE TO <primary>" returns a WORD which is equivalent to the value that would be generated to access <primary> if it were a call-by-reference parameter (e.g. an SIRW for simple data type, a data descriptor for array rows, an indexed data descriptor for subscripted variables). <Primary> can be of any data type that NEWP allows to be passed as a call-by-reference parameter.

REGISTERS (REGISTERS)
REGISTERS is implemented in NEWP as it is in ESPOL.

RETURN (MACHINEOPS)
RETURN is implemented in NEWP as it is in ESPOL.

SCANIN (MACHINEOPS)
SCANIN is implemented in NEWP as it is in ESPOL.

SCANOUT (MACHINEOPS)
SCANOUT is implemented in NEWP as it is in ESPOL, except that it is not valid when the B7000 compiler option is set.

SETINHIBIT (MACHINEOPS)
SETINHIBIT(<real>,<integer>) generates the B7000 operator Set Memory Inhibits (SINH), where the low-order eight bits of the <real> parameter contain the inhibit mask and the low-order four bits of the <integer> parameter represent the memory module. SETINHIBIT is valid only when the B7000 compiler option is set.

SETLIMITS (MACHINEOPS)
SETLIMITS(<real>, <integer1>, <integer2>, <integer3>) generates the B7000 operator Set Memory Limits (SLMT). The low-order four bits of the <real> parameter represent the availability mask, <integer1> and <integer2> indicate the lower and upper addresses, respectively, and <integer3> is the memory module number. SETLIMITS is valid only when the B7000 compiler option is set.

STOP (MACHINEOPS)
STOP generates the HALT operator. If two parameters are specified, the two values are loaded into the A and B registers before the HALT operator is executed. If four parameters are specified, the third and fourth parameters are interpreted as addresses into which the values in the A and B registers are to be stored following execution of the HALT operator.

STOP77 (MACHINEOPS)
The STOP77 intrinsic is similar to the STOP intrinsic, except that the B7000 operator STOP is generated. STOP77 is valid only when the B7000 compiler option is set.

SUSPEND (MACHINEOPS)
The SUSPEND intrinsic is similar to the PAUSE intrinsic, except that the B7000 operator Pause Until Interrupt (PAUS) is generated. SUSPEND is valid only when the B7000 compiler option is set.

TAG (WORD)
TAG is implemented in NEWP as it is in ESPOL.

TIMER (MACHINEOPS)
TIMER is implemented in NEWP as it is in ESPOL.

UNLOCK (MISC)
UNLOCK is implemented in NEWP as it is in ESPOL, with the additional restriction that UNLOCK can be used only in CONTROLSTATE blocks.

TEMPORARY -- UNLOCK is temporarily allowed in NORMALSTATE blocks.

VECTORCHECKSUM (MISC)
VECTORCHECKSUM(<subscripted variable>,<integer>) returns the checksum calculated over <integer> words starting at the location described by the <subscripted

variable>, which must be an element of an INTEGER, REAL, BOOLEAN, or WORD array. VECTORCHECKSUM uses vectormode operators; an INVALID OP interrupt is generated if the processor does not support vectormode.

VIA (REFERENCE)
The syntax "<type> VIA <word primary>" is used to access an item referenced by <word primary> (which is assumed to be a reference generated by the "REFERENCE TO <primary>" syntax) as a value of the specified <type>. <Type> may be any of the following simple types: INTEGER, BOOLEAN, REAL, DOUBLE, POINTER, EVENT, DESCRIPTOR, FILE, TASK, WORD.

WHOAMI (MACHINEOPS)
WHOAMI performs the same function in NEWP as the MYSELF intrinsic does in ESPOL.

WORD (WORD)
WORD is implemented in NEWP as it is in ESPOL.

ZAP (MACHINEOPS)
ZAP is implemented in NEWP as it is in ESPOL.

8 COMPILER CONTROLS

8.1 COMPILER OPTIONS

The compiler options available in NEWP are identical to those in ALGOL, except for the changes, additions, and deletions noted below. The following features differ from the ALGOL semantics:

- Specifying an option without including SET, RESET, or POP causes no action other than setting that option. Specifically, STANDARD OPTIONS ARE NOT RESET (e.g. $MERGE does NOT reset LIST). The CLEAR option, described below, can be used to reset all standard options. Examples:

```
$ MERGE          % IN NEWP HAS THE SAME EFFECT AS:
$ SET MERGE      % IN ALGOL OR NEWP

$ CLEAR MERGE    % IN NEWP HAS THE SAME EFFECT AS:
$ MERGE          % IN ALGOL
```

- The dollar sign ("$") must appear in column 1, 2, or 3.

- Compiler-provided (standard) options are recognized in option expressions (e.g. in the expression "$SET LIST=MYOPTION OR OMIT", OMIT is recognized as the compiler-provided option and is not interpreted as a user option).

- The options MERGE, LIST, SEPCOMP, and NEW allow a file name to be specified as a string enclosed in quotes (the string may not include the quote character). The specified file name is used to set the TITLE attribute of the compiler files TAPE, LINE, HOST, and NEWTAPE, respectively.

```
$ SET MERGE "(FREDS)ALTERFILE"
```

- The INSTALLATION option accepts either a single intrinsic number or a range of numbers.

- The MAKEHOST option does not allow an <environment-list> to be specified. SEPCOMPable environments in NEWP are controlled by the SEPCOMPLEVEL block direction.

- The SEPCOMP option has different semantics in NEWP than it does in ALGOL. SEPCOMP in NEWP is described in a later section of this document.

- The STATISTICS option produces MCP statistics (as in ESPOL, not ALGOL) for each procedure. Also, statistics will be summarized for any block for which the block direction STATSUMMARY appears.

- Cross-reference generation (XREF) is performed by the NEWP compiler itself and is controlled by two compiler options, XREF and XREFFILES. If XREF is set, a cross-reference listing is produced. If XREFFILES is set, cross-reference files for use by SYSTEM/INTERACTIVEXREF are generated. These options can be set or reset independently of each other.

* See also
8.2 Block Directions
8.3 SEPCOMP

The following compiler options are implemented in NEWP in addition to the standard ALGOL options:

B7000 (RESET if B6000, SET if B7000)
The B7000 option, if SET, specifies that the machine on which the compiler is running is a B7000 Series machine. If RESET, the compilation is assumed to be running on a B6000 Series machine.

CLEAR (cannot be SET or RESET)
The CLEAR option resets all compiler-provided, settable options.

LISTO (RESET)
If LISTO is set, all records from the secondary input file (TAPE) that are voided or replaced and all records from the primary input file (CARD) that are omitted will be printed.

LIST1 (RESET)
If LIST1 is set, a listing is produced during the compiler's first pass compiling modules.

MCP (RESET)
MCP is implemented in NEWP as it is in ESPOL.

NOCOUNT (RESET)
NOCOUNT is implemented in NEWP as it is in ESPOL.

PROCREF (SET)
If PROCREF is set, each line which references a procedure will be listed with the line number of that procedure's declaration.

READLOCK (RESET)
READLOCK is implemented in NEWP as it is in ESPOL.

UNDERLINE (RESET)
If UNDERLINE is set, all procedure names in procedure declarations will be underlined on the output listing.

The following ALGOL compiler options are not implemented in NEWP:

AREACLASS
AUTOBIND
BCL
BEGINSEGMENT
BIND
BINDER
BREAKHOST
BREAKPOINT
B7700
CHECK
DOUBLESPACE
DUMPINFO
ENDSEGMENT
EXTERNAL
FORMAT
GO
GO TO
HOST
INITIALIZE
INTRINSICS
LEVEL
LIBRARY
LISTDELETED
LISTOMITTED
LOADINFO
NOSTACKARRAYS
NOXREFLIST
OPTIMIZE
PURGE
SEGDESCABOVE
SEQERR
STOP
USE
WRITEAFTER
XDECS
XREFS

8.2 BLOCK DIRECTIONS

```
         |<-------------------- , --------------------|
         |                                            |
-- [ ----- CONTROLSTATE -------------------------------- ] -----|
          |                                        |
          |- NORMALSTATE --------------------------|
          |                                        |
          |- SAFE ---------------------------------|
          |                                        |
          |- SEGMENT ------------------------------|
          |              |                         |
          |              |- = --<procedure identifier>-|
          |              |                         |
          |              |-<segment name>----------|
          |              |                         |
          |              |- = --<segment name>-----|
          |              |                         |
          |              |- = --<integer>----------|
          |                                        |
          |- SEGMENTLEVEL -- = --<integer>---------|
          |                                        |
          |- SEPCOMPLEVEL -- = --<integer>---------|
          |                                        |
          |- STATSUMMARY --------------------------|
          |                                        |
          |              |<------- , -------|      |
          |              |                  |      |
          |- UNSAFE -- ( ----- DESCRIPTOR ----- ) --|
                            |                |
                            |- FORK ---------|
                            |                |
                            |- MACHINEOPS ---|
                            |                |
                            |- MEMORY -------|
                            |                |
                            |- MISC ---------|
                            |                |
                            |- REFERENCE ----|
                            |                |
                            |- REGISTERS ----|
                            |                |
                            |- WORD ---------|
```

Example:

```
BEGIN [SEGMENT OUTERBLOCK, SEPCOMPLEVEL=5]
  ...
  PROCEDURE P(X);
    VALUE X;  REAL X;
    BEGIN
    [UNSAFE(MEMORY,WORD),SEGMENT=5,CONTROLSTATE]
    ...
    END P;
  ...
END.
```

Through the use of block directions, the programmer can control segmentation, the use of potentially dangerous constructs, and other compilation details. Block directions may occur inside brackets immediately after any BEGIN. In order to make the following discussion more readable, the word "block" has been used for the concept of "block or procedure".

In general, block directions are inherited by nested blocks unless overridden by block directions appearing in the nested block. However, the UNSAFE block direction is not inherited. TEMPORARY -- Temporarily, UNSAFE is inherited by nested blocks.

The following keywords are recognized as directions to the compiler:

CONTROLSTATE
: The block is to run in control-state. (UNSAFE only)

  TEMPORARY -- CONTROLSTATE may be set for procedures only.

NORMALSTATE
: The block is to run in normal-state.

  TEMPORARY -- NORMALSTATE may be set for procedures only.

SAFE (TEMPORARY)
: Generate syntax errors for all UNSAFE constructs used in this block, except those enabled by subsequent UNSAFE specifications.

SEGMENT
: Segmentation information for the block may be specified by one of the following phrases:

a. SEGMENT -- Make a new segment.

b. SEGMENT = <procedure identifier> -- Add the code for this block to the segment which contains the specified procedure. The <procedure identifier> must have appeared previously in a <procedure declaration> or a <forward procedure declaration>.

c. SEGMENT <segment name> -- Make a new segment and associate the specified <segment name> with it.

d. SEGMENT = <segment name> -- Add the code for this block to the segment associated with this segment name. The <segment name> must have appeared previously in a "SEGMENT <segment name>" compiler direction.

e. SEGMENT = <integer> -- Add to segment <integer>. (UNSAFE only)

SEGMENTLEVEL
The SEGMENTLEVEL block direction must be followed by an integer which specifies the lex level at or below which segmentation takes place.

SEPCOMPLEVEL
The SEPCOMPLEVEL block direction must be followed by an integer which specifies the lex level at or below which separate compilation can occur for declarations.

STATSUMMARY
If the compiler option STATISTICS is set, statistics will be summarized for the block(s) on which the STATSUMMARY block direction appears (statistics are automatically summarized for each procedure when STATISTICS is set).

UNSAFE
Allow the use of the potentially dangerous constructs specified by the following keywords:

DESCRIPTOR
Allow the DESCRIPTOR data type and DESCRIPTOR type transfer function.

FORK
Allow FORK statements.

MACHINEOPS
Allow the use of the following intrinsics: EVAL (temporary), EXIT, FAILREGISTER, FMMRREADLOCK, HEYOU, IGNOREPARITY, INTERRUPTCHANNEL, MAKEPCW (temporary), MOVESTACK, PAUSE, RETURN, SCANIN, SCANOUT, SETINHIBIT, SETLIMITS STOP, STOP77, SUSPEND, TIMER, WHOAMI, ZAP.

MEMORY
Allow the MEMORY array intrinsic.

MISC
Allow address equation, NULL as a <procedure body>, <option list>s on WAIT and WAITANDRESET statements, the OVERWRITE option on the REPLACE statement, SAVE ARRAY declarations, and the following intrinsics: BUZZ, BUZZ47, LEXOFFSET, UNLOCK.

REFERENCE
Allow "REFERENCE TO <primary>", "<type> VIA <word primary>", "<type> AT <address primary>" syntax.

REGISTERS
Allow the REGISTERS array intrinsic and D-register override clause ("@ [<exp>]").

WORD
Allow WORD data type, WORD type transfer function, POINTER(<word exp>), and TAG.

* See also
3 Program Structure
7 UNSAFE Mode

8.3 SEPCOMP

Separate Compilation (SEPCOMP) is implemented in the NEWP compiler itself and does not require invocation of the BINDER. In order to use the SEPCOMP facility, a "host" must be generated by compiling a program with the compiler option MAKEHOST set, causing the compiler to place information necessary for SEPCOMP in the codefile. Patches to the host symbolic can be compiled with SEPCOMP set, resulting in an abbreviated compilation, since only the affected areas of the program are actually compiled.

A SEPCOMP "region" is any declaration that occurs at a lex level less than or equal to the SEPCOMPLEVEL (either the one specified by the SEPCOMPLEVEL block direction or the default). The executable statements of a procedure (or outer block) are considered to be a "text" region. A region can be changed, added, or deleted during SEPCOMP. The text region of a procedure (or outer block) is recompiled if any declaration region associated with it is separately compiled.

At the present time, the programmer is responsible for observing the following restrictions:

a) Do not change a VALUE ARRAY, ARRAY, DEFINE, FORMAT, FILE, TRANSLATETABLE, or TRUTHSET declaration without explicitly causing all references to the changed identifier to be recompiled.

b) Do not patch (via SEPCOMP) COMMENTs and areas of the symbolic which were omitted because the compiler option OMIT was set.

c) Do not change the starting or ending sequence number of a region.

d) Do not put multiple regions on one line.

These restrictions do not apply to patches to declarations or text in procedures above the SEPCOMPLEVEL, since a patch to such a procedure causes the entire procedure to be recompiled.

The user compiler options that were set when the host was compiled and the VERSION that was supplied for the host are preserved and reinstated during a SEPCOMP.

The title of the symbolic file which was compiled as the host is saved in the codefile generated during a MAKEHOST or SEPCOMP compile. This title is used as the default title for the TAPE file when SEPCOMPing (but may be overridden by label-equation) and is selected as follows: if MAKEHOST is set, the titles of the NEWTAPE, TAPE, and CARD (if it is a disk file) files are examined in order, and the first valid title is saved; if SEPCOMP is set, the title of the TAPE file is saved.

* See also
8.1 Compiler Options
8.2 Block Directions

9 ALGOL FEATURES NOT IMPLEMENTED IN NEWP

The following ALGOL features are not implemented in NEWP. Some of these features were considered undesirable or inappropriate in the context of NEWP, while others have been or will be replaced by NEWP features which perform approximately the same function. Page numbers refer to the ALGOL Language Reference Manual.

DECLARATIONS

- The ALPHA declaration, page 4-2, and ALPHA as a <type> in the <array declaration>, page 4-3.
- OWN variables (OWN as the <local or own> part), page 4-2.
- <Array row equivalence>, page 4-5.
- The DUMP declaration, page 4-13.
- <In-out part>s in FORMAT declarations, page 4-20.
- Non-EBCDIC <simple string>s in formats, page 4-24.
- The O format phrase, page 4-36, and the R format phrase, page 4-37.
- <Forward interrupt declaration>s and <forward switch declaration>s, page 4-42.
- The INTERRUPT declaration, page 4-44.
- The LIST declaration, page 4-46.
- The MONITOR declaration, page 4-48.
- The PICTURE declaration, page 4-51.
- <Parameter delimiter>s other than comma (","), page 4-55.
- Labels and formats as formal parameters (LABEL and FORMAT as <specifier>s in <formal parameter part>s), page 4-55.
- <Switch declaration>s, page 4-60.

STATEMENTS

- The ACCEPT statement, page 5-2.
- The update replacement (":=*") construct for assignment to task or file attributes, page 5-6.
- The ATTACH statement, page 5-11.
- The BREAKPOINT statement, page 5-12.
- The CALL statement, page 5-13.
- The CHANGEFILE statement, page 5-19.
- The CHECKPOINT statement, page 5-20.
- The CONTINUE statement, page 5-30.
- The DEALLOCATE statement, page 5-31.
- The DETACH statement, page 5-32.
- The DISABLE statement, page 5-33.
- The ENABLE statement, page 5-36.
- The EXCHANGE statement (for disk file rows), page 5-38.
- The FILL statement, page 5-39.
- Direct I/O, page 5-49.
- The MERGE statement, page 5-56.
- The multiple (file) attribute assignment statement, page 5-57.
- The ON statement, page 5-58.
- The PROCESS statement, page 5-62.
- Core-to-core I/O (i.e. <core-to-core part>s as <file part>s), page 5-66.
- Formatted I/O (i.e. <format designator>s or in-line editing specifications in <format and list part>s), page 5-66.
- Binary I/O (i.e. "*" as the format in a <format and list part>), page 5-66.
- Free-field I/O (i.e. <free field part>s in <format and list part>s), page 5-66.
- The WHILE clause for FOR-loops occurring in the <format and list part>s of READ and WRITE statements, page 5-66.
- The update replacement (":=*") construct for assignment to the control variable of a FOR-loop occurring in the <format and list part> of a READ or WRITE statement, page 5-66.
- The REMOVEFILE statement, page 5-77.
- All intrinsic <translate table>s that refer to BCL (these translate tables can be declared, however), page 5-79.
- The <replace family-change statement>, page 5-88.
- The REWIND statement, page 5-92.
- The RUN statement, page 5-93.
- The SPACE statement, page 5-104.
- The SWAP statement, page 5-108.
- The VECTORMODE statement, page 5-111.
- The WAIT statement with no parameters (wait for interrupt), page 5-114.
- The WHEN statement, page 5-117.

EXPRESSIONS

- Vertical bar ("|") as the logical operator OR in <Boolean term>, page 6-9.
- <Designational expression>s, except <label identifier>, page 6-16.

- The following intrinsics (page 6-19 to 6-30):

  ARCCOS, ARCSIN, ARCTAN, ARCTAN2, ATANH
  CHECKSUM, COS, COSH, COTAN
  DABS, DARCCOS, DARCSIN, DARCTAN, DARCTAN2, DCOS,
  DCOSH, DELTA, DEF, DEFC, DEXP, DGAMMA, DIMP, DINTEGER,
  DLGAMMA, DLN, DLOG, DMAX, DMIN, DNABS, DSCALELEFT,
  DSCALERIGHT, DSIN, DSINH, DSQURT, DTAN, DTANH
  ENTIER, ERF, ERFC, EXP
  GAMMA
  LINENUMBER, LNGAMMA, LOG
  NORMALIZE
  RANDOM
  SCALELEFT, SCALERIGHT, SCALERIGHTF, SCALERIGHTT, SIGN,
  SIN, SINH, SIZE(<pointer identifier>), SQRT
  TAN, TANH

MISCELLANEOUS

- Identifiers, numbers, and strings continued across card images.
- Multi-character operators with embedded blanks. As an exception, the update replacement operator, ":=*", is allowed to have one blank between the "=" and the "*" (i.e. ":= *" is allowed).
- <Global part> in a program unit, page 3-1.
- KIND=READER for the compiler file CARD (CARD must be label-equated to a disk file), Appendix E (of the ALGOL Language Reference Manual).
- Batch Facility, Appendix F (of the ALGOL Language Reference Manual).
- BDMS statements (see B7000/B6000 Series DMSII HOST Reference Manual, #5001498).

MARK 3.1

DOCUMENT CHANGES NOTES (D NOTES)

PATCH

D2596 PATCH - LABEL EQUATION FOR "$." CARDS

Label-equation is now allowed for file title specifications of one node for the following commands: $.FILE, $.DISK, $.PATCHDECK and $.DISK$.

Also, other $. commands may appear on the same $. card after $.FILE, $.DISK, $.PATCHDECK and $.DISK$.

Example:

```
FILE PATCH/RUN:

?BEGIN JOB PATCHER(STRING PATCHFILE);
   RUN SYSTEM/PATCH;
   FILE TAPE(TITLE=SYMBOL/SOURCE ON PACK1);
   FILE INPUT(TITLE=#PATCHFILE);
   DATA CARD
$#PATCH SEPARATOR CARD 1
$ SET LIST MERGE NEW
$#PATCH SEPARATOR CARD 2
$.FILE INPUT COMPARE
?END JOB
```

CANDE or ODT or WFL Input:

```
START PATCH/RUN (" PATCH/SOURCE/23 ON PACK2")
```

The result of this START command will be a SYSTEM/PATCH run with the file "PATCH/SOURCE/23 ON PACK2" being used. Also, a compare listing is generated, even though the $.COMPARE command comes after the $. FILE command.

D2956 PATCH - CLARIFICATION OF TEXT FIELDS

System Software Operational Guide, Volume 1 (Form No. 5001563), SYSTEM/PATCH documentation, should be changed.

Add the following before the last sentence of the first paragraph on Page 10-1-3:

> "The text field of a $.CARD is columns 3-80 (9-80 if $.COBOL is set). Parsing of this text field is terminated by a "%" character."

D2957 PATCH - IMPROVE DOCUMENTATION

System Software Operational Guide, Volume 1 (Form No. 5001563), SYSTEM/PATCH documentation, should be changed as follows:

1. Insert the following between SEQ and VOID in the list of recognized $ options on Page 10-1-1:

   "<sequence base>
   <sequence increment>"

2. Insert the following after the list of recognized $ options on Page 10-1-1:

   "These options' functions are performed by SYSTEM/PATCH and not by the compiler. Because of this, they are (except for MERGE) erased from the card on which they appear before that card is written to the patch file. SYSTEM/PATCH creates SET and POP VOIDT cards as needed to simulate the functions of each of these options."

PLI

D2879 PLI - CLARIFICATION OF "FIXEDOVERFLOW"

Replace the first paragraph of the description of the FIXEDOVERFLOW condition in the PL/I Reference Manual (Form No. 5001530), Page A2-2, with the following:

> "This condition occurs during fixed point arithmetic operations if the results of these operations exceed the integer capacity of the hardware. See SIZE for a related condition which does checking for overflow related to declared attributes of data items."

PLI

P1941 PLI - DEALLOCATE DYNAMIC BOUNDS AUTO ARRAYS

Automatic arrays with dynamic bounds are now deallocated on block exit.

P2163 PLI - BASED DOUBLE VARIABLES

Overlay defining using based storage will now work properly for double float variables.

P2164 PLI - EXTRA SUBSCRIPT SYNTAX ERROR

Cases in which an array is referenced with an extra subscript and the extra subscript is an asterisk now cause a syntax error.

P2165 PLI - BAD PROCEDURE DIRECTORY

A problem has been corrected where the compiler could produce a bad procedure directory which caused an "END OF FILE" error in the BINDER.

P2166 PLI - DEFINES FOR DOUBLES

The length attribute of double precision items is now correctly checked against the length of the base items on which they are defined.

P2167 PLI - FLAG ILLEGAL ARRAY BOUNDS

Arrays declared with illegal constant bounds will be flagged. An example of an array which would formerly pass syntax is the following:

```
    DCL A (1:-1) FIXED;
```

P2168 PLI - LOOP ON DEFINE

A problem has been corrected where it was possible for the compiler to loop infinitely when compiling a define declaration in which the base is a fully subscripted array.

P2169 PLI - BAD DUMMY FOR STRING INTRINSIC

A problem has been corrected in which the dummy for a bit string was not getting set up properly for a call on a string intrinsic.

P2170 PLI - "CALL" VERB WITH "EVENT" OPTION

Using the CALL verb with the EVENT option has not been implemented. The compiler was not issuing a warning for an attempt to use this construct as it does for other similar cases. This has been corrected; a warning for a level 6 error will now be issued.

P2385 PLI - MISSING COMMAS IN FILE OPTIONS

An error is now given when commas are omitted from file option clauses (e.g., OPTIONS(KIND='DISK' FILETYPE=8)). The error is level 5 since any following option specifications are ignored.

P2386 PLI - WRITE STATEMENT WITHOUT "FROM"

A level 6 syntax error is now given for a WRITE statement without a "FROM" option.

P2445 PLI - MISCELLANEOUS FAULTS USING "PIC 'Z'" ITEMS

Incorrect code for certain operations involving items of type PIC 'Z' is no longer generated. The problem appeared when assigning from a PIC item which had been declared DEFINED on some other item and had a zero value.

Symptoms included intrinsic faults, system faults, and super halts. These no longer occur.

P2446 PLI - "DO STATEMENT" PROBLEM

DO statements with a "BY" and a "TO" will no longer leave an extra value on the stack. This problem could have caused program failure for STACK OVERFLOW.

P2447 PLI - LEVEL "9" ERROR

It is not permitted to declare an array parameter as in the following:

```
        P : PROCEDURE (A);
            DECLARE A (1:N);
        END;
```

Since running a program with such a declaration can have serious consequences, the error now given is at level 9 and the program will be non-executable.

PLI INTRINSICS

P2171 PLINTRN - REAL BIT CONSTANTS

A problem has been corrected in which bit constants were not being read correctly via stream input into fixed length bit variables in some cases.

P2172 PLINTRN - FILLING OUT BIT STRINGS

A problem has been corrected in which bit strings of fixed length assigned a string of shorter length were not always getting padded properly with zeroes.

P2173 PLINTRN - CONVERSION ERROR

An invalid character condition is no longer raised when converting a numeric picture to a decimal float variable when a drifting "+" picture character is used and the arithmetic value of the picture variable is a negative number, or conversely when a drifting '-" picture character is used and the arithmetic value of the picture is a positive number. An example of the program which would fail is the following:

```
    EXAMPLE: PROCEDURE;
    DCL A PICTURE '----9';
    A=25;
    X=A;
    END EXAMPLE;
```

P2174 PLINTRN - "ISAM" INTRINSICS, INFINITE LOOP

If an ISAM file were opened output and immediately closed without writing any records to it, and subsequently opened I/O, it was possible for the intrinsics to go into an infinite loop if two records were written to the file, the second having a key less than the first. This problem has been corrected.

P2250 PLINTRN - "ISAM," ELIMINATE SUPERHALT

The ISOPEN intrinsic (for opening KEYED files) put a TAG6 SCW in a variable local to its block. Prior to the III.0 system release, this caused no problem; the III.0 release notices this extra SCW and presumes it to be a "subblock" on the stack. If a program is DSed while running in ISOPEN, a superhalt occurs.

Now, ISOPEN is prevented from storing the TAG6 SCW; the SCW is therefore eliminated.

P2346 PLINTRN - ELIMINATE "SEG ARRAY" ERROR IN "TRACE"

SEG ARRAY errors caused by incorrect code in the PLITRACE intrinsic no longer occur.

PRINT BINDER INFO

D2685 PRINTBIND - PRINTING OF PROGRAM UNIT DIRECTORY

Code files which contain a "Program Unit Directory" (created by the $SET LIBRARY option in the ALGOL and FORTRAN compilers) will now be correctly analyzed.

Specifically, each separate procedure will be independently analyzed, rather than only the first.

D2686 PRINTBIND - LISTING FORMAT IMPROVEMENTS

Various improvements have been made to the style of the printout produced by PRINTBINDINFO.

In particular, the indenting at each level has been reduced from 6 columns to 4.

D2687 PRINTBIND - IMPROVE HANDLING OF "DATABASE" INFORMATION

The printout of DATABASE information has been improved.

Specifically, the timestamp is printed in a meaningful date-time fashion, rather than in hex.

Also, the names of items within the database are printed (with address couples when necessary) in EBCDIC rather than hex.

PRINT BINDER INFO

P2025 PRINTBIND - IMPROVED HANDLING OF UNKNOWN ITEMS

PRINTBINDINFO will now handle unknown universal class, subclass pairs in a more useful fashion.

If the universal class is zero and the subclass is not known to PRINTBINDINFO, the entire contents of the item will be printed in hex.

If the universal class is unknown, the universal class, subclass pair is printed and the contents of the item are recursively analyzed.

In both cases, analysis of the binding information continues in a normal fashion.

Finally, PRINTBINDINFO is brought up to date with knowledge of all current universal class and subclass items.

P2026 PRINTBIND - CORRECT HANDLING OF "PL/I" ITEMS

PL/I style variables, constants and arrays are correctly identified by PRINTBINDINFO. ARRAYs and CONTROLLED items are properly identified.

PRINTBINDINFO will now detect PL/I items it does not understand and will no longer abort with an INVALID INDEX when they occur.

P2065 PRINTBIND - LARGE CODE FILES

PRINTBINDINFO can now handle programs that contain more than 20,000 words of binding information.

P2470 PRINTBIND - PREVENT PRINTING TOO MANY "EXTRASEGS"

PRINTBINDINFO has been prevent from analyzing erroneous "ADDITIONAL REENTRANT LOCATIONS" following the last procedure in the procedure directory.

REMOTE JOB ENTRY

D2578 RJE - "BUG DUMP <ARRAYNAME>"

A new DEBUG option has been added to RJE. The "SM BUG" command has been expanded to allow the dumping to the PRINTFILE of any of the global arrays used by by RJE.

Example:

```
    <rje mix#> SM BUG DUMP <arrayname>
```

The array specified by <arrayname> will be dumped in HEX and ALPHA.

If <arrayname> does not exist, the following message will be displayed:

```
"NO SUCH ARRAY: <arrayname>"
```

If the request is correct, the following will be displayed:

```
"#OK"
```

This option will allow snap shots of the arrays used by RJE to be taken in real time.

D2597 RJE - "DEBUG" OUTPUT CHANGE

The JOBNUMBER of REMLPXX files will now be displayed in the DEBUG printer output for all UPDATEPRINTQUEUE, FIREUPINITIALIZATION and RECORDMANAGER display lines. This will aid in the debugging of AUTOBACKUP problems.

D2625 RJE - "CLEAR" <LSN> COMMAND

RJE will now allow the clearing of all stations on a line. A new control command "CLEAR" has been added, which will cause the following:

1. Save all stations on the line.
2. Recall all messages.
3. Force the line not ready.
4. DS any WFLTASK, AUTOBACKKUP, SERVICETASK (*RS) that is associated with the line.
5. RESET all RJE control states.
6. LOG OFF stations with cause being "STATION CLEARED".
7. UPDATE LINKFILE.

The stations on the line are left in a NOT/READY status and must be readied to be used.

Syntax:

```
-- CLEAR --<lsn>--|
   --
```

D2627 RJE - ACCESSCODE

If an RJE must be logged on before use, an accesscode may be required on all RJE sessions. To require a default accesscode on all logons with a default usercode, RJE may be compiled with "ACCESSDEFAULT" set. To require default accesscodes later, the SM message "SO ACCESSCODE" or "RO ACCESSCODE" may be used to turn off default accesscode.

When logging onto an RJE with a default usercode, if an accesscode is required for all users, or if the "ACCESSCODENEEDED" indicator for the usercode is set in the USERDATAFILE, RJE will ask the following: "ACCESSCODE?". After an accesscode is entered, RJE will ask the following: "AND YOUR ACCESSCODE PASSWORD". A "." may entered if the accesscode has no password. The accesscode and password is validated in the USERDATAFILE. If valid, it will be applied to all jobs entered at the RJE unless an explicit accesscode or usercode specification appears in the job input. If it is not valid, RJE will respond with the following:

```
"INVALID ACCESSCODE/PASSWORD, ENTER ACCESSCODE"
```

The RJE will not be logged on until a valid accesscode and password has been entered. If "HELLO" is entered, RJE will abort the logon attempt and ask for the usercode again.

If an RJE is logged on with a default usercode, the ACCESS command may be used, as follows:

The accesscode may be changed.
The accesscode may be added if there is no accesscode.
The accesscode may be deleted.

The syntax is the same as the ACCESS command in CANDE, except that it is preceded by an "*".

If an RJE does not require logon, and no default usercode is assigned, no default accesscode can be assigned.

RJE will record the accesscode in the beginning of session messages and end of session messages.

D2632 RJE - "MAXTERMINALS" NOT A POWER OF "2"

RJE will now allow the setting of the define MAXTERMINALS to any positive integer value (it is used to determine the size of RJE working arrays). This removes the restriction that MAXTERMINALS be a power of 2 and will allow sites with large RJE networks to reduce the amount of overhead in the MCS.

Note: MAXTERMINALS must be set to at least the number of RJE stations declared in the site's NDL. Values much greater will cause an increase in overhead and memory requirements for RJE.

D2638 RJE - NUMBER OF STATIONS ATTACHED

The number of attached stations at BOJ of RJE will be stored in word 55 of the first record of the LINKFILE.

D2641 RJE - "LINKFILE"

The RJE MCS employs a disk file with the internal name "LINK" and the title "SYSTEM/RJE/LINKFILE" to save information regarding the terminals under its control in the event of a halt/load.

Saved in this file are such items as the states of the various run-time terminal options, usercodes, phone numbers, etc. All saved information is discarded if RJE is restarted after changing "NIF" and "DCPCODE" files as all saved information is considered as possibly incompatible with the new datacom environment.

If the title of the RJE MCS is changed and then executed (assuming the new title is declared in the site's NDL), a new linkfile will be created called <mcs name>/LINKFILE. The linkfile is always created and/or looked for on the family that the MCS is running.

The file consists of a minimum of two 240 word records. The first record contains RJE's mix number, version, and date and time of latest "NIF" and "DCPCODE" files. The remaining records consist of 48 word blocks containing information about each terminal followed by 48 word blocks containing autobackup restart record.

The layout of the "LINKFILE" is as follows:

```
%                                                                      %
%                          LINKFILE INFO RECORD                        %
%                          --------------------                        %
%                                                                      %
%                          240 WORDS PER RECORD                        %
%                                                                      %
%RECORD 0 :                                                            %
%                                                                      %
%        WORD [0]                 =  RJE MIX NUMBER                    %
%             [50]                =  NDL COMPILATION DATE              %
%             [51]                =  NDL COMPILATION TIME              %
%             [52]-[53]           =  RJE LEVEL (VERSION)               %
%             [54]                =  MAXTERMINALS                      %
%             [55]                =  NUMBER OF ATTACHED STATIONS       %
%                                                                      %
%RECORD 1 - (MAXTERMINALS+4) DIV 5 :                                   %
%                                                                      %
%                          TERMINAL INFO BLOCKS                        %
%                          --------------------                        %
%                                                                      %
%                      5 - 48 WORD BLOCKS PER RECORD                   %
%                                                                      %
%        WORD [0]                 =  TERMINAL STATUS WORD              %
%             [1]                 =  SESSION NUMBER                    %
%             [2].[47:8]          =  LENGTH OF PHONE NUMBER            %
%             [2].[39:40]-[3]     =  PHONE NUMBER                      %
%             [4]-[7].[47:24]     =  USERCODE                          %
%             [7].[23:24]-[10]    =  ACCESSCODE                        %
%             [11]                =  CALL BACK INFO                    %
%             [12]                =  PRINT QUEUE INFO WORD             %
%             [13]-[47]           =  JOB NO.S QUEUED FOR OUTPUT        %
%                                                                      %
%RECORDS ((MAXTERMINALS+4) DIV 5)+1  -  (MAXTERMINALS*2+4) DIV 5 :     %
%                                                                      %
%                       AUTOBACKUP RESTART RECORDS                     %
%                       --------------------------                     %
%                                                                      %
%                      5 - 48 WORD BLOCKS PER RECORD                   %
%                                                                      %
%        WORD [0].[15:16]         =  JOB NUMBER BEING PRINTED          %
%             [0].[23:08]         =  RJE SUPPLIED UNIQUENESS NUMBER    %
%             [1].[47:16]         =  BACKUP RECORD TO RESUME UPON      %
%             [1].[31:12]         =  INDEX WITHIN BACKUP RECORD        %
%             [2].[47:08]         =  # OF CHARACTERS IN DIR PB'ED      %
%             [2].[38:12]         =  USER SPECIFIED NUMBER OF COPIES   %
%             [2].[27:12]         =  USER SPECIFIED SAVE OR NOT        %
%             [2].[23:04]         =  # OF LEVELS IN DIRECTORY          %
%             [2].[19:04]         =  DIRECTORY FULL FILE NAME          %
%             [3].[47:12]         =  COPIES PRINTED                    %
%             [3].[35:12]         =  # OF CHARACTERS OF CURRENT FILE   %
%             [5]-[47]            =  NAME OF FILE CURRENTLY PRINTING   %
%                                                                      %
```

The structure of this file is subject to change at a future time.

If the RJE MCS should ever be terminated with a fault, it is recommended that the "LINKFILE" be removed before the MCS is reactivated.

D2810 RJE - FILE TRANSFER

The implementation requirements of the following new features have caused an increase in the number of "system control messages" (device address 00 messages) plus format changes to some of the old messages. The use of these new "system control messages" may cause compatibility problems with old or non-Burroughs RJE terminals and III.1 SYSTEM/RJE. The use of the File Transfer features in this release are applicable only to B6000/B7000 series systems.

SYSTEM/RJE has been modified to allow files to be transferred between host systems and between host systems and terminals. The file transfer is done through a new station added to a site's NDL for each RJE station family. This new station has a device address of 04 but no other special attributes. An example of this new station can be found in the SYMBOL/SOURCENDL file for this software release.

Since this new feature is symmetric, it may be invoked between any two systems without regard to "HOST" vs. "TERMINAL" status; thus, the term "HOST" will be used in the remainder of this document.

File transfer is supported for disk and pack files with the following restrictions:

1. Only code and 8-bit data files (including text files) may be transferred.

2. Only one file at a time may be transferred in each direction.

3. File titles cannot be greater than 100 characters in length.

The user initiates a file transfer by entering a variation of a WFL "COPY" statement on the host ODT. This command specifies source and destination hosts and file titles. The hosts are specified by the use of their "HOSTNAME"; in the case of a "LARGE SYSTEM" host (B6700, B6800, B7700, etc.), this hostname is created in the following fashion:

> If the "HN" (HOSTNAME) ODT command has not been used, then SYSTEM/RJE creates a hostname at run time. This hostname is the letter "B" followed by the 4 digit system model number (6700, 6800, etc.) and then the 3 or 4 digit system serial number. For sites that have specified a hostname by the "HN" ODT command, that hostname is used. Note that if two sites are connected to a host with the same hostname, requests to transfer files may not produce the desired results.

The actual transfer of a file is initiated by the exchange of control information between the hosts. If the specified source host is the user's local host, the transfer is a "PUT"; if the source host is the remote host, then the transfer is a "FETCH". This means that if the file to be transferred from my host to your host is on my host, the "COPY" request will generate a "PUT" control message; if the file is on your host, it will generate a "FETCH" control message. All file transfers occur as "PUTS"; if one host wishes to do a "FETCH", it sends a "FETCH" message to the other host which causes it to initiate a "PUT" sequence. The "PUT" message elicits a "PUT REPLY" which signals that data transfer may begin or supplies a reason for the rejection of the "PUT" request. If the transfer is OKed, the sending host then sends one or more data messages to device address 04 followed by an end-of-file message. The sending host processes a task called "FILEX[<lsn>]", where <lsn> is the LSN of the station connected to the remote host. This task runs independently of SYSTEM/RJE. It opens the requested source file and performs any data compaction or translation required in the course of the transfer. The receiving host processes a task called "FILER[<lsn>]" which also runs independently of SYSTEM/RJE. This task reverses whatever FILEX did to the data and creates the requested destination file. Upon completion of the file transfer, the receiving host informs the requestor of completion of the request.

Any file transfer requests that occur while another transfer is in progress are queued and then executed when possible. When a request is queued, the following message is displayed:

```
    #FILE TRANSFER STATION IN USE - COPY REQUEST WILL BE QUEUED.
```

All requests are queued as "PUTS" , so if a user requests a "COPY" that results in a "FETCH" being generated, the request is sent to the other host and may be queued for the reason given above. If the "COPY" request is syntactically correct and passes resident and security checks, the following message is displayed:

```
    #COPY REQUEST VALID - WILL FORWARD.
```

The other host responds either with an error display or a request accepted display, in which case the transfer will start.

All files on the destination hosts will be locked as type data.

To implement this new feature, a set of new commands has been added to SYSTEM/RJE and several old commands have been modified. The file transfer requests and their progress displays are made on the host ODT by using SYSTEM/RJE "SM" message syntax.

New SYSTEM/RJE SM Commands

COPY

```
-- COPY --<filename>----------------------------------------------------->
   --                  |                 |
                       |-<usercode spec>-|

>------------------------------------------- FROM --<volume spec>----->
   |                                       |
   |- AS -----<filename>-------------------|
   |          |            |               |
   |- ONTO -|              |-<usercode spec>-|

>- TO --<volume spec>--------------------------------------------------|
```

<usercode spec>

```
-- ( -- USER -- = --<usercode>-------------------------------- ) --|
                                |                            |
                                |- , PASSWORD = <password> -|
                                |                            |
                                |- / <password> ------------|
```

<volume spec>

```
---- DISK --------- ( ----------------------------------------------->
  |                 |    |                                         |
  |- PACK ------- |    |-------------- TAPE ----- , -|
  |                 |    |                            
  |-<volumename>-|    |- KIND = -| |- TAPE7 --|
                                   |            |
                                   |- TAPE9 --|
                                   |            |
                                   |- PETAPE -|
                                   |            |
                                   |- PACK ---|

>- HOST = <hostname> -- ) -------------------------------------------|
```

The "COPY" command transfers the requested file from the source host to the destination host. The <usercode spec> is used to verify access rights to the specified file and to the use of the file transfer station. It may also affect defaults for parts of the file title. For file names that start with the characters "*" or "(", the <usercode spec> usercode is not used as a prefix. The <usercode spec> is required for any source or destination that is a large system host. The default volume kind is "PACK". The "COPY" request will not overwrite an existing file on the destination host unless the "ONTO" modifier is used.

Examples:

```
<rje mix#> SM COPY TESTFILE (USER=MIKE) FROM MCPMAST (HOST=B6700282)
     TO DISK (HOST=B68001004)
```

Transfers "(MIKE)TESTFILE" from a pack called "MCPMAST" on host "B6700282" to a pack named "DISK" on host "B68001004". Usercode "MIKE" (with no password) must be a valid usercode on both hosts.

```
<rje mix#> SM COPY TESTFILE (USER=MIKE) AS X (USER=RJE/RJE) FROM
     MCPMAST (HOST=B6700282) TO DISK (HOST=B68001004)
```

Same as first example except the destination file title is "(RJE)X ON DISK". Usercode "MIKE" is used for all security and resident checks on host "B6700282" while "RJE/RJE" is used on host "B68001004".

```
<rje mix#> SM COPY *WFL (USER=MIKE) ONTO TEMP (USER=WHY,PASSWORD=ME)
     FROM DISK (HOST=B6700282) TO PACK (HOST=B7700003)
```

Transfers the system file "*WFL" from disk on host "B6700282" as a file called "(WHY)TEMP ON PACK" (overwrites the file if it already exists) on host "B7700003".

The following error messages are produced as a result of an invalid "COPY" command:

FILENAME GTR 100 CHAR.

INVALID FILENAME: <filename>.

NO FROM OR TO PARTS.

NO FROM PART.

INVALID VOLUMENAME: <volumename>.

KIND MAY NOT BE SET THERE.

Example: setting "KIND=DISK" when volumename is "DISK".

MISSING EQUAL.

Equal signs are required after key words "USER", "PASSWORD","HOST","KIND".

INVALID MEDIA TYPE: <media type>.

NO HOSTNAME.

EXPECTING PARENTHESIS.

NO TO PART.

NO USERCODE/PASSWORD.

INVALID USER/PASS SYNTAX.

SOURCE AND DEST HOSTNAMES THE SAME.

THIS HOST NOT SOURCE OR DEST.

USERDATAFILE FROZEN - TRY LATER.

The userdatafile was not available at security check time.

SECURITY ERROR ON USER = <usercode>.

FILE <filename> NOT AVAILABLE.

FILE <filename> NOT RESIDENT.

FILE <filename> ALREADY EXISTS.

FILE <filename> IS OPEN EXCLUSIVE AND CANNOT BE REPLACED.

FAMILY <familyname> IS NOT PRESENT.

ABORT

```
-- ABORT --<lsn>--- SEND -------------------|
   ----            |        |  |             |
                   |- RECV -|  |- = --------|
                               |             |
                               |-<integer>--|
                               |             |
                               |-<filename>-|
```

The ABORT command is used to stop an active file transfer or delete one or all entries that are queued for a host. The key words "SEND" and "RECV" are used to specify the type of transfer to abort. The <lsn> is the LSN of the station to which the remote host is connected. See the "FTS" command for more information concerning the <integer> and <filename> formats and meanings.

Examples:

<rje mix#> SM ABORT <lsn> SEND

Aborts the current active "FILEX" task (the sender).

<rje mix#> SM ABORT <lsn> RECV

Aborts the current active "FILER" task (the receiver).

<rje mix#> SM ABORT <lsn> SEND =

Deletes all requests queued at this host for the requested <lsn>.

<rje mix#> SM ABORT <lsn> RECV =

Deletes all requests queued at the remote host for transmission to the requested <lsn>.

<rje mix#> SM ABORT <lsn> SEND <integer>

Deletes the <integer> entry from the queued list but does not delete the active entry (queued number 0).

<rje mix#> SM ABORT <lsn> RECV <integer>

Deletes the <integer> entry from the queued list of the remote host connected to the requested <lsn>, but does not delete active entry (queued number 0).

<rje mix#> SM ABORT <lsn> SEND <filename>

Deletes the first queued entry from the queued list for the <lsn> requested where the queued source filename matches <filename>.

<rje mix#> SM ABORT <lsn> RECV <filename>

Deletes the first queued entry from the queued list of the host that is using the <lsn> where the queued source filename matches <filename>.

ONLINE and OFFLINE

```
---- ONLINE ----<lsn>--|
   |  ----           |
   |- OFFLINE -|
      ----
```

The ONLINE command initiates the connection between two hosts. Depending on the type of datacom connection, this command has different effects.

SWITCHED LINE WITHOUT AUTOMATIC CALLING UNIT

The request has no effect and elicits a response of :

#NON DIALOUT CAPABLE STATION.

SWITCHED LINE WITH AUTOMATIC CALLING UNIT

Causes dialout if phone number is set up (see PH command).
RJE will respond with either :

#NO DIALOUT PHONE NUMBER
or
#DIALING OUT STARTED

DIRECT OR LEASED LINE

Causes the ##RJE and 09 station-id messages to be sent.
RJE will respond with :

#OK

If the <lsn> is already connected and active RJE displays:

#STATION ALREADY ACTIVE.

The OFFLINE command causes termination of the datacom connection between two host systems. Depending on the type of datacom connection, this command has different effects.

SWITCHED LINE - causes log-off and line disconnect.

DIRECT OR LEASED LINE - causes log-off, but leaves line active.

RJE responds with the following message to the OFFLINE command:

#OK.

PH (Phone Number)

```
-- PH --<lsn>----------------------|
             |                   |
             |- CLEAR --------|
             |                   |
             |-<phone number>-|
             |                   |
             |- INT ----------|
```

<phone number>

```
-- Maximum of 11 digits --|
```

The PH (phone number) command is used to display, clear, change or retry the automatic calling unit phone number associated with a station.

Examples:

```
<rje mix#> SM PH <lsn>
```

Displays the current phone number (if any).

```
<rje mix#> SM PH <lsn> CLEAR
```

Clears the phone number.

```
<rje mix#> SM PH <lsn> <phone number>
```

Enters new phone number.

```
<rje mix#> SM PH <lsn> INT
```

Causes dialout retry if the callback option is set for <lsn> station.

SF (Set Factors)

```
                 |<-----------------------------|
                 |                              |
-- SF --<lsn>----- BLKSZ -----------<integer>----|
                  |                | |
                  |- BUFSZ -|  |- = -|
                  |          |
                  |- FTBLK -|
```

This command specifies the characters per transmission blocking factor (BLKSZ), buffer size in characters of remote printer and or punch output (BUFSZ) and the maximum number of characters allowed in a file transfer block (FTBLK). The following defaults and limits are imposed :

	Defaults	Limits
BLKSZ	400	132 through 405 (inclusive)
BUFSZ	820	132 through 2000 (inclusive)
FTBLK	400	400 through 2000 (inclusive)

Changing any of these values should be done with the utmost care to avoid exceeding the limits of the remote terminal or of the datacom line.

TF (Type Factors)

```
-- TF --|
```

This command has been modified to display the characters per transmission blocking factor (BLKSZ), buffer size in characters of remote printer and or punch output (BUFSZ) and the maximum number of characters allowed in a file transfer block (FTBLK). The defaults and limits imposed are the same as for the SF command.

FTS (File Transfer Status)

```
-- FTS ------------|
         |          |
         |-<lsn>-|
```

The FTS command monitors the file transfer activity. This command without the optional <lsn> displays one or three lines for each active station attached to SYSTEM/RJE. If the station has no "HOSTNAME", or if there are no file transfers active on the station, one line will be displayed showing LSN, station name, hostname (if any) and a comment of "NO ACTIVE FILE TRANSFER". For stations with an active file transfer, three lines are displayed showing the status of the station and of the active file transfers.

If the optional <lsn> is used with the FTS command, only the information for that station is displayed. In this case, the display is either one or four or more lines depending on the number of queued requests.

Example outputs:

The examples that follow are the results of using the FTS command without the optional <lsn>.

```
[034] RJE1   B6800282  NO ACTIVE FILE TRANSFER
[039] RJETIO  NO HOSTNAME  NO ACTIVE FILE TRANSFER
[044] RJEGMM   B6700282 2 REQUEST QUEUED
  RECV: FILER ACTIVE (4321) 21 OF 54
  SEND: FILEX ACTIVE (4401) 1053 OF 10049 (PACING)
```

The first line of the above example has the following meanings:

```
[034]      = The base LSN of the RJE station
RJE1       = Station name
B6800282   = Remote host name that is connected to this station
```

In the second line of the above example the host or terminal using station RJETIO has not sent its host name (if it has one) and no file transfers are allowed.

In lines three through five of the above example, the host B6700282 is active on station RJEGMM and there is a file transfer running in each direction. The "RECV" line shows the status of the receiving task FILER. The number in parentheses is the mix number of FILER while the XXX OF YYY numbers are the current count of records transferred of the total for the file. The "SEND" line shows the status of the sending task FILEX. The number in parentheses is the mix number of FILEX while the XXX OF YYY numbers are the current count of records transferred of the total for the file. The word "PACING" indicates that FILEX is waiting for a pacing flag before continuing with the next file transfer block of records.

The example that follows is of LSN 44 in the last example using the optional <lsn>.

```
[044] RJEGMM   B6700282 2 REQUEST QUEUED
  RECV: FILER ACTIVE (4321) 21 OF 54
     <00> (MIKE)TESTFILE ON TESTPACK
         AS (MIKE)NEW/TESTFILE ON MCPMAST.
  SEND: FILEX ACTIVE (4401) 1053 OF 10046 (PACING)
     <00> (MIKE)RJEX ON MCPMAST.
     <01> *SYMBOL/WFL
         AS (MIKE)WFL ON TESTPACK.
     <02> (RJE)DUMMY ON PACK
         ONTO (MIKE)SAVE/THIS/FILE/FOREVER.
```

The lines that start with the items [044], RECV: and SEND: have the same meanings as above. The lines that start with <00> are the file names of the active file transfers and the counts displayed in the recv and send lines refer to them. The lines under the send line that start with <01> through <02> are requests that have been queued. The numbers displayed within the <> and the file names on those lines are the numbers and file names used in the "ABORT" command described above.

*SF and *TF

The RSC keyin commands *SF and *TF have been changed to reflect the addition of the new factor "FTBLK". The syntax of these commands is the same as their SM command counterparts.

File Transfer Recovery

If during a file transfer, one or both of the hosts stops (Halt/Load, RJE fault, DCP death, etc.), SYSTEM/RJE stores enough information about the transfer to restart it where it left off. This is done by storing the current record counts and the "COPY" requests in RJE's "LINKFILE".

On the receiving host, the destination file is not created directly but a temporary file is created to allow restart and recovery. This temporary file is called "FILERFILE[<lsn>]" and is created under the destination usercode, unless it is a system file (*), on the destination <volumename>. Upon successful completion of the transfer, the temporary file is re-titled and locked.

At initialization RJE checks to see if it was involved in a file transfer that was not completed. The requirements to continue a file transfer are the following:

1. Connection is made to the same host on the same station (datacom line) that was last using it.

2. The linkfile on both hosts are still intact (do not remove the linkfile).

3. The temporary output file on the receiving host is available.

If the above conditions exist, then at initialization the receiving host requests the file transfer to continue. The "SEND" and "RECV" lines of the "FTS" command display have the word "RESTARTED" in parenthenses for any file transfer that has been continued.

If the re-connection of the datacom line is to a different host, all queued request and the active requests are deleted and all restart information is discarded.

Linkfile Restructure

Several changes have been made to SYSTEM/RJE to support the file transfer feature. One of these is the restructuring of the linkfile. To ensure the correct linkfile format, SYSTEM/RJE checks at initialization the "LINKFILE VERSION" which is stored in word 56 of record 0. If the version of SYSTEM/RJE and its linkfile do not agree, the linkfile will be discarded. There are currently four reasons for discarding an old linkfile :

1. NDL file (DCPCODE and NIF) changes.

2. Maxterminals in SYSTEM/RJE has changed.

3. Linkfile version not correct.

4. Parity error reading old file.

Each of these conditions causes SYSTEM/RJE to display an appropriate message.

D2828 RJE - "WAIT" STATE MAINTAINED

The state of the WAIT SM command will be maintained over Halt/Loads and RJE being forced to EOJ by the QUIT command. The WAIT state will be stored in the LINKFILE in record 0 word 57.

D2862 RJE - "WFLMESSAGE ARRAY" USAGE INCREASE

RJE will now better utilize the WFLMESSAGE ARRAY when handling card input from the remote site. RJE was causing insertion in the WFLQUEUE at 15% of capacity where now insertion will not take place until a minimum of 80%.

REMOTE JOB ENTRY
------ --- -----

P1037 RJE - RECONFIGURATION WITH ACTIVE "AUTOBACKUP" TASKS

RJE will no longer allow reconfiguration to a station that has logged off but still has an active AUTOBACKUP task. The error message "# STATION IN USE, ENTER STATION NAME" will be displayed at the site trying to log on.

P1579 RJE - "STATIONID" OR "USER" CHANGES

Changing the status of STATIONID or USER with the SM, SO or RO commands will now cause an update of RJE's LINKFILE.

P1610 RJE - "RJELEVEL" DISPLAY CHANGE

The form of the RJE patch level that is displayed at logon time has been changed to include the patch number. The RJE LEVEL message is also displayed on the host site ODT at BOJ of RJE and is written to the PRINTFILE if DEBUG is set.

P1772 RJE - "RJE AUTOPRINT" HANG

RJE's AUTOPRINT routine will now check that the PRINTER NOT READY message removed from its CURRENT QUEUE has been inserted by the REMOTECONTROL procedure and is not a message left from an earlier run.

P1794 RJE - "WFLCOMPILER" NOT GOING TO "EOT"

It was possible for the WFLCOMPILER processed off by RJE to hang in the mix because of garbage characters in the QUEUEIN array. This has been corrected by blanking one character past the last character sent by the remote terminal to the host system.

P1795 RJE - "LSNRAY" VS. STATIONS WITH "NO-LINES"

RJE will now build the LSNRAY correctly at BOJ when stations with NO-LINES are found. The problem could cause several error conditions, bad data or INVALID INDEX. Stations that have NO-LINES have only their processor (DA="00") entry updated in LSNRAY. The card reader, printer and RSC information are not needed when reconfiguration is requested by the STATIONID option.

P1814 RJE - SWAPPING PRINT QUEUES VS. STATION ID

Several problems involving reconfiguration and exchanging of print queues have been corrected. These involved possible SEG ARRAY errors and storage of incorrect information in a print queue.

P1815 RJE - "CONT" AND "PRINT" BANNERS

RJE will now print the CONT and PRINT banners when the NEWLINE option is reset.

P1816 RJE - "QT" PRINTER FILES SAVED OVER HALT/LOAD

While printing a job with several print files, the remote site operator QTed a file to be printed later. A Halt/Load occurred at the host site; thus, on restart the remote printing continued from where it left off. However, on completion of the last file of the job, all files were removed including the file previously QTed. This problem has been corrected.

P1942 RJE - RESTARTFILE NAME "SEG ARRAY" ERROR

RJE would die with a SEG ARRAY error if the title of the REMLP file used to restart an AUTOBACKUP task ended on a word boundary (<length of title> MOD 6=0). This caused the procedure DISPLAYTOSTANDARD not to find a delimiting "." or NUL. This problem has been corrected by increasing the title length by one to include the trailing ".".

P2086 RJE - "MAXTERMINALS" GREATER THAN "48"

RJE will now allow the setting of MAXTERMINALS to an integer value greater than 48 and not die with an INVALID OP. Several REAL and Boolean identifiers have been changed to EBCDIC ARRAY to remove the 48 MAXTERMINALS limit. (On III.0 and II.9 levels of RJE, the requirement that MAXTERMINALS be a power of 2 still holds true; therefore, setting MAXTERMINALS to 64 will now work.)

P2204 RJE - "*ME COPIES 2" VS. RESIZE OF FILEINFO

RJE's AUTOPRINT procedure will no longer DS with a SEG ARRAY error after the following is entered:

```
    *ME <job no>/<task no> COPIES 2 SAVE
```

P2205 RJE - NOLOGON VS. USERCODE "."

For RJE stations with NOLOGON set, the MCS LOGON and LOGOFF log records will have zeroed word #7 (USERCODE). This will cause the line in the job summary output and in LOGANALYZER output "USERCODE:." to be deleted, as well as cause blank USERCODE lines in the summary block character heading.

P2374 RJE - UNIT DEVICE NUMBER

The first block of input from an RJE card reader could be lost under certain circumstances, if a card reader block is the first input from an uninitialized RJE terminal. This problem has been correccted.

P2561 RJE - "*ME" FAILS ON SPECIFIC FILENAME IF "SB=PACK"

RJE's autoprint will now print backup entries created by the *ME request when a specific filename is supplied and the REMCP files are on "PACK" (SB DISK=PACK).

P2562 RJE - "DESTNAME" DIRECTED "REMCP" FILES

RJE will now automatically punch REMCP files which have been directed to it at the host site with REMOTEPUNCH reset.

P2563 RJE - "REMCP" FILE TITLES ON WORD BOUNDARY

It was possible for AUTOBACKKUP to DS with a SEG ARRAY error in the DISPLAYTOSTANDARD procedure while building the restart record. This problem has been corrected.

READER SORTER - GENERAL

D2859 READ-SORT - "RSC3" CONTROLS

The RSC3 control for the B9137-III Reader/Sorter has been implemented. As a result, the B7000/B6000 Sortercontrol Language Reference Manual (Form No. 5001183) should be revised as follows.

Page 2-17, the list of Procedure Description Division should read as follows:

"BREAKOUT
FLOW-STOPPED
INITIATION
INVALID-ITEM
NORMAL-ITEM=MICR-ITEM
RESTART
TERMINATION"

Page 2-20, under ITEM-TYPE, the description should read as follows:

"A read-only six-digit register that contains a value indicating the type of invalid document detected. ITEM-TYPE is valid only during execution of the INVALID-ITEM SECTION and the FLOW-STOPPED SECTION.

For a B9134, only one digit is returned (right-justified).

For a B9137, two digits are returned (right-justified). The right-most digit is the value for HEAD1; the other digit is the value for HEAD2."

The values remain unchanged.

In sections 3 and 4, references to the following sections have been combined into the NORMAL-ITEM SECTION (synonym=MICR-ITEM SECTION), because the Reader/Sorter subsystem considers data to be no more than data transmitted to the UCR:

MICR-ITEM SECTION
OCR1-ITEM SECTION
OCR2-ITEM SECTION

Page 3-3, replace the syntax diagram with the following:

```
-- CONTROL --- ( -- HALT -- ) ------------------------|
              |                                      |
              |- ( -- SLEW -- ) ---------------------|
              |                                      |
              |- ( -- COUNT -- ) --------------------|
              |                                      |
              |- ( -- LIGHT --<numeric operand>-- ) -|
              |                                      |
              |- ( -- ICM --<numeric operand>-- ) ---|
              |                                      |
              |- ( -- SELECT --<select options>-- ) -|
```

```
<select options>

----<numeric operand>------------------------------------------->
   |                     | |                    | |
   |- POCKET ----------| | |<---------------| |
   |                     | | |                  | |
   |- REJECT-POCKET ---| |--------------------|
                              |             |
                              |- ENDORSE -|
                              |             |
                              |- CLICK ---|

>----------------------------------------------------------------|
   |                                                        |
   |- ON EXCEPTION --<statement>----------------------------|
                                  |                         |
                                  |- ELSE ---<statement>-----|
                                             |               |
                                             |- NEXT SENTENCE -|
```

Add the following as the first pararaphs of the semantics on Page 3-3:

"The CONTROL options are valid for RSC1 and/or RSC3 as follows:

Option	RSC1	RSC3
HALT	X	X
SLEW		X
COUNT	X	
LIGHT	X	X
ICM		X
SELECT(<num op>)	X	X
SELECT(POCKET)	X	X
SELECT(REJ-POC)	X	X
SELECT(ENDORSE)		X
SELECT(CLICK)		X

If there are two heads, then ITEM-TYPE returns two digits; therefore, it must be declared 99 in the UCR to get the information.

The following options are exclusive; i.e., they cannot be used in combination with any other options:

HALT, SLEW, COUNT, LIGHT, ICM"

The third paragraph under Semantics should read as follows:

"CONTROL(HALT) is valid in the NORMAL-ITEM SECTION."

Add the following to Semantics on Page 3-3:

"CONTROL(SLEW) is a microfilm command that advances the microfilm. It is valid only when flow is stopped and the Reader/Sorter is ready. Also, it is valid only by itself.

CONTROL(ICM <numeric operand>) is a microfilm command that causes the Reader/Sorter to use the last used microfilm identification number on the next <numeric operand> frames (maximum=9). Flow must be stopped and the Reader/Sorter must be ready.

CONTROL(COUNT) causes the batch counter to advance."

Page 3-4, the first paragraph should read as follows:

"CONTROL(SELECT) is valid in the following procedure sections:

INVALID-ITEM SECTION
NORMAL-ITEM SECTION=MICR-ITEM SECTION"

Page 3-4, add the following paragraphs to the semantics for CONTROL:

"The following semantics apply only to the B9137; they are invalid for the B9134.

CONTROL(SELECT <numeric operand> ENDORSE) means

a. Route this check to POCKET #<numeric operand>
b. Load all four bands in the UCR endorser record
c. Endorse all four bands in the UCR endorser record

CONTROL(SELECT <numeric operand> CLICK) means

a. Route this check to POCKET #<numeric operand>
b. Microfilm this check

CONTROL(SELECT <numeric operand> ENDORSE CLICK) means

Select pocket, load all bands, endorse all bands and microfilm the check."

Page 3-23, the first paragraph should read as follows:

"The REMAP statement is valid only in a repass UCR that performs image-matching procedures. REMAP may be specified in the following sections:

FLOW-STOPPED SECTION
NORMAL-ITEM SECTION
RESTART SECTION"

In Section 4, Unit Control Routine (UCR), all references to the following SECTIONs should be deleted, as they are now handled as NORMAL-ITEMs:

BLACK-BAND-MICR SECTION
BLACK-BAND-OCR1 SECTION
BLACK-BAND-OCR2 SECTION

Page 4-2, replace the syntax diagram with the following:

```
-- ROUTINE ------------------------------------------------------->
              |                              |  |       |
              |- " --<comment information>-- " -|  |- . -|

   |<------------------------------------|
   |                                     |
>-----/1**\- DATA SECTION ------------------- PROCEDURE . -------->
      |                                  |
      |-/1\- ENDPOINT-ATTRIBUTES SECTION ---|
      |                                  |
      |-/1\- IMAGE-FILE ATTRIBUTES SECTION -|
      |                                  |
      |-/1\- POCKET-ATTRIBUTES SECTION -----|
      |                                  |
      |-/1**\- READER-SORTER SECTION -------|
      |                                  |
      |-/1\- RECOVERY-FILE SECTION ---------|
      |                                  |
      |-/1\- SORT-FIELDS SECTION -----------|

   |<-------------------------------|
   |                                |
>-----/1\- BREAKOUT SECTION ---------------------------------------|
      |                          |
      |-/1\- FLOW-STOPPED SECTION --|
      |                          |
      |-/1\- INITIATION SECTION ----|
      |                          |
      |-/1\- INVALID-ITEM SECTION --|
      |                          |
      |-/1\--- NORMAL-ITEM SECTION -|
      |      |                   |
      |      |- MICR-ITEM SECTION ---|
      |                          |
      |-/1\- RESTART SECTION -------|
      |                          |
      |-/1\- TERMINATION SECTION ---|
```

Page 4-13, replace the syntax diagram for **READER-SORTER SECTION** with the following:

```
-- READER-SORTER SECTION . ------------------------------------------>

   |<--------------------------------------------------------------|
   |                                                               |
>-----/1\- HEAD1-RECORD --- REVERSE -----------<record description>--->
      |                |               | |       |
      |-/1\- HEAD2-RECORD -|               | |- . -|
      |                                |
      |-/1\- ENDORSER-RECORD ----------|

>--------------------------------------------------------------------|
   |                    |
   |-<pattern expression>-|
```

Page 4-14, replace the first four paragraphs under "Semantics" with the following:

"The READER-SORTER SECTION describes the format(s) of acceptable document images and the endorser area.

HEAD1-RECORD and HEAD2-RECORD (for B9137 only) identify a particular <record description> (data area) into which document images are read from the Reader/Sorter HEAD1 and HEAD2 read heads, respectively. HEAD1 HEAD2 <record description>s are referred to as READER-SORTER <record description>s.

REVERSE must be specified for HEAD1 and HEAD2 records. REVERSE indicates that the <record description> is interpreted with the individual data fields described from right-to-left. (Check formats require a right-to-left scan to detect delimiters.) Document format description must be from right-to-left.

<record description> must contain a complete description of the documents to be processed. The HEAD1, HEAD2 and ENDORSER <record description> represents the default or most commonly-expected document format. The <record description> must describe the document format from right-to-left.

The endorser area (for B9137 only) describes the data sent to the Reader/Sorter when the CONTROL(SELECT ENDORSER) statement is executed. REVERSE is not valid. The contents of the endorser area are sent to the control every time this statement is executed. All four 32-byte bands are loaded and endorsed."

Page 4-15, change "MICR" to "HEAD2".

Page 4-18, the list of procedure description sections should read as follows:

```
BREAKOUT
FLOW-STOPPED
INITIATION
INVALID-ITEM
NORMAL-ITEM=MICR-ITEM SECTION
RESTART
TERMINATION
```

Page 4-19 should be deleted.

Page 4-20 should be deleted

Page 4-21 should be deleted

Page 4-23, under Semantics, the fifth paragraph should read as follows:

"Detection of a Black-Band item causes the NORMAL-ITEM SECTION to be invoked. Upon exiting the NORMAL-ITEM SECTION, the FLOW-STOPPED SECTION is invoked. (If the SET END-OF-RUN statement is issued, the TERMINATION SECTION is invoked rather than the FLOW-STOPPED SECTION.)"

Page 4-25, under Semantics, the first paragraph should read as follows:

"The INVALID-ITEM SECTION is invoked when a READ COMPLETE is received but there is no data (i.e., unencoded) or the data is not valid."

Page 4-26, replace the syntax diagram with the following:

```
                                 |<------------|
                                 |             |
-- USE FOR --- NORMAL-ITEM . -----<paragraph>--- END -------------->
              |              |
              |- MICR-ITEM . ---|

>--- NORMAL-ITEM . ------------------------------------------------|
  |                  |
  |- MICR-ITEM . ---|
```

The semantics should read as follows:

"The NORMAL-ITEM SECTION is invoked when a read complete is received and valid data is sent from the Reader/Sorter. The NORMAL-ITEM SECTION must contain the code necessary to fully process MICR documents."

Page 4-27 should be deleted.

Page 4-28 should be deleted.

Page 4-31, replace the syntax diagram with the following:

```
-- LABEL-ROUTINE -------------------------------------------------->
                   |                                  |  |      |
                   |- " --<comment information>-- " -|  |- . -|

  |<--------------------------------|
  |                                 |
>-----/1**\- DATASECTION -------------- PROCEDURE . --------------->
    |                                |
    |-/1**\- READER-SORTERSECTION -|

  |<------------------------------|
  |                               |
>-----/1**\- FLOW-STOPPEDSECTION ----------------------------------|
    |                            |
    |-/1**\- NORMAL-ITEMSECTION --|
```

Page 4-32, the fourth paragraph should read as follows:

"The NORMAL-ITEM SECTION must be declared. This section must release exactly one display-form label. The label must be terminated with a period; e.g., "RS/TWO.". This procedure must also select the document to a pocket."

Appendix A, revise the list of reserved words as follows:

Delete the following words:

BLACK-BAND-MICR
BLACK-BAND-OCR1
BLACK-BAND-OCR2
OCR1-ITEM
OCR2-ITEM
OCR1-RECORD
OCR2-RECORD

Add the following words:

ENDORSER
HEAD1
HEAD2
NORMAL
NORMAL-ITEM

READER SORTER - INITIALIZER

D1744 INITIALIZER - "RSC3"

INITIALIZER now recognizes RSC3 controls. A change has been made to send RSMONITOR only a code number to report an RSCl or RSC3 or disk control (previously, INITIALIZER sent the result descriptor). Now RSPINITIALIZER does not know the format of the RD, only the defines for Reader/Sorter controls. INITIALIZER only acknowledges the presence of Reader/Sorter controls. Any other control is ignored by INITIALIZER and no record of it is sent to RSPINITIALIZER.

READER SORTER - MIL6700

D2743 MIL6700 - DEBUG

To activate DEBUGTRACE, set DEBUG and TRACE.

READER SORTER - READERSORTER
------ ------ - ------------

D2745 READERSORTER - "RSC3"

RSPINITIALIZER gets a code from INITIALIZER for the types of control present. RSPINITIALIZER checks the code against the RSUNITTABLE created from RSNET, then logs what it finds. RSPINITIALIZER no longer knows about the RD format in the RSP.

READER SORTER - READERSORTER

P1435 READERSORTER - SCAN ERROR WORD

RSSCANIN and RSSCANOUT now use the stored fault information upon detection of a fault condition.

READER SORTER - RSANALYZER

D2746 RSANALYZER - CHANGE CARD FILE NAME

The internal name of the program's card file has been changed from "CD" to "CARD".

READER SORTER - RSLOG

P1441 RSLOG - "RSC3"

RSLOG has been modified as follows:

1. To recognize RSC3 controls.
2. To show the reason for initiation (RSMONITOR option).
3. To show the fault reason in BICERROR during scans.

READER SORTER - RSMCP

P2487 RSMCP - "RSMONITOR"

RSMONITOR now looks for a BIC, then forks RSMONITOR in the local box that can see the BIC.

READER SORTER - SORTERCONTROL

D2747 SORTERCONTRL - "RSC3"

New error messages have been added. Other message declarations have been tidied.

D2748 SORTERCONTRL - "RSC3"

The compiler has been modified to allow for the compilation of multiple Reader/Sorter record descriptions.

READER SORTER - SORTERCONTROL

P1372 SORTERCONTRL - ALLOW FILLER WITH "USAGE"

Filler can now specify usage.

P1463 SORTERCONTRL - "RSC3"

The following changes have been made:

1. Added the following reserved words:

   NORMAL-ITEM
   HEAD1-RECORD
   HEAD2-RECORD
   ENDORSER-RECORD
   ENDORSE, SLEW, ICM, CLICK

2. Deleted all references to the following:

   MICR
   OCR1
   OCR2
   BLACK-BAND SECTIONS

3. Renumbered UCR sections to take account of deleted OCR and BLACK-BAND sections.

4. Added communicate codes for SLEW and ICM on microfilm.

5. Modified compiler to handle one or two read head declarations.

P1464 SORTERCONTRL - "RSC3"

The CONTROL statement, used for controlling the Reader/Sorter, has been modified to allow the following additional functions:

1. ENDORSE - To activate the non-impact endorser.
2. CLICK - To activate the microfilm camera.
3. SLEW - To slew the microfilm.
4. ICM - To place image count marks on microfilm.

SCRMCP

D2943 SCRMCP - "MCP" SYMBOLIC CONSOLIDATION

The external procedures MAINTENANCE and SORT have been moved into the MCP symbolic. They can no longer be separately compiled and bound; however, they can be changed using SEPCOMP.

D2947 SCRMCP - "CURRENTDATE"

A CURRENTDATE function has been added to the MAT language. It returns a six-digit number whose value is the current date in mmddyy form.

As a result, the following changes should be made to the MAT Language Information Manual (Form No. 5000169):

Page 2-15:

Add "| CURRENTDATE" to the list of options for <primary>.

Page 2-20:

Add the following:

"CURRENTDATE

This primary returns the current date as a six-digit number of the form mmddyy. This construct is provided to allow for self-identification of the printed results from a MAT program."

SCRMCP

P2486 SCRMCP - "IOTIMEOUT"

IOTIMEOUT default is now set at 90,000 milliseconds instead of 90. TIMELIMIT default for datacom files is now set at 90 seconds instead of 90,000.

P2535 SCRMCP - LABEL "CASE" STATEMENTS

All unlabeled CASE statements are now labeled.

P2615 SCRMCP - HANDLE "VOIDT" PROPERLY

Previously, $SET VOIDT would delete lines from the file CARD as well as the file TAPE. Now it deletes lines only from the file TAPE.

P2621 SCRMCP - TEST RESTRICTED TO PACK TYPES

The following tests (or utilities) have been restricted to 215 and 225 packs only:

IV (SCR function)
SUPERIV
PK15
PEP

The following tests (or utilities) have been restricted to 215, 225 and 235 packs only:

PKTEST
PK16
PK11
PK10
PK09
PK08
PK07
PK06
PK04

SCR

D2246 SCR - "B6800 MCP"

New hardware interrupt literals are now recognized, and, where appropriate, logged.

The format of log records reporting B6800 processor errors or diagnostic interrupts is as follows:

Word	Contents	
0	Link	
1	Date	
2	Time	
3	Type	31:16 = 2 (LOGMAJMAINT) 15:16 = 15 (MLCONFIDENCEERROR)
4	MCPID	47:12 = System Serial # 35:12 = Mark Digit 23:12 = Mark Level 11:12 = Patch Level
5	ProcessorID, WHOI Value	
6	P1 Parameter, P1 Interrupt Parameter with Tag set to 0	
7	P3 Parameter, P3 Interrupt Parameter with Tag set to 0	
8	P2 Parameter Tag	
9	P2 Parameter, P2 Interrupt Parameter with Tag set to 0	

The interpretation of P1, P2 and P3 is based on B6800 processor specifications.

D2414 SCR - ADD "RSC-3" TEST NAME

The name "RSC3MAINT" has been added to the list of names which may be specified as test routines files to be loaded to an RSP. This is to allow the new system file SYSTEM/RS/RSC3MAINT to be loaded. This file is the test routine which will be used in conjunction with the RSC-3 Control and B9137 Reader/Sorter.

D2758 SCR - "GCR" TAPE ALLOWED

Code has been added to SCR (MAT Language) to allow for the testing of GCR type tape units. When generating a "VERIFY" "TESTAPE", the following densities MUST be specified at some point after "TESTAPE": "DENSITY 6250" or "DENSITY 1600". This is required only on GCR type units. "DENSITY" is not required during options "CERTIFY" or "SELECT".

Examples:

```
    (1) SCR; VERIFY TAPE MT80 (TESTAPE DENSITY 6250 REEL 1)
    (2) SCR; VERIFY TAPE MT80 (SELECT ALL)
```

D2933 SCR - "IOSTATEMENT" DOCUMENTATION

The following changes should be made to the MAT Language Information Manual (Form No. 5000169) to reflect the implementation of the previously undocumented IOSTATEMENT:

Add the following to the right side of <PRIMARY> on page 2-15:

```
    "/ <IOSTATEMENT IDENTIFIER> ( <IOSTATEMENT ID PARAMETER> )"
```

Add the following to the syntax on page 2-16:

```
   "<IOSTATEMENT ID PARAMETER> ::= MPXNO / RESULT / UNITNO /
         UNITTYPE / IOLENGTH / IOSIZE / PCROUTE / IOCW.ADDRESS /
         IOCW.CONTROL"
```

Add the following to examples on page 2-17:

```
   "12. READPK(UNITNO)
    13. READPK(IOCW.ADDRESS)"
```

Add the following paragraphs to semantics on page 2-20:

"<IOSTATEMENT IDENTIFIER> (MPXNO)

THIS PRIMARY RETURNS THE MULTIPLEXOR ROUTE NUMBER USED BY THE MOST RECENT EXECUTION OF THE I/O STATEMENT ASSOCIATED WITH <IOSTATEMENT IDENTIFIER>.

<IOSTATEMENT IDENTIFIER> (RESULT)

THIS PRIMARY RETURNS THE RESULT DESCRIPTOR OF THE MOST RECENT EXECUTION OF THE I/O STATEMENT ASSOCIATED WITH <IOSTATEMENT IDENTIFIER>. THE ERROR FIELD OF THE RESULT DESCRIPTOR IS THE ONLY PORTION OF THE RESULT DESCRIPTOR RETURNED.

<IOSTATEMENT IDENTIFIER> (UNITNO)

THIS PRIMARY RETURNS THE PHYSICAL UNIT NUMBER OF THE UNIT USED BY THE MOST RECENT EXECUTION OF THE I/O STATEMENT ASSOCIATED WITH <IOSTATEMENT IDENTIFIER>.

<IOSTATEMENT IDENTIFIER> (UNITTYPE)

THIS PRIMARY RETURNS THE RESULTS OF THE SCANIN FOR UNIT TYPE OF THE UNIT USED BY THE MOST RECENT EXECUTION OF THE I/O STATEMENT ASSOCIATED WITH <IOSTATEMENT IDENTIFIER>.

<IOSTATEMENT IDENTIFIER> (IOLENGTH)

THIS PRIMARY RETURNS THE WORD COUNT USED BY THE MOST RECENT EXECUTION OF THE I/O STATEMENT ASSOCIATED WITH <IOSTATEMENT IDENTIFIER>.

<IOSTATEMENT IDENTIFIER> (IOSIZE)

THIS PRIMARY RETURNS THE ACTUAL SIZE OF THE RECORD (IN CHARACTERS) USED BY THE MOST RECENT EXECUTION OF THE I/O STATEMENT ASSOCIATED WITH <IOSTATEMENT IDENTIFIER>.

<IOSTATEMENT IDENTIFIER> (PCROUTE)

THIS PRIMARY RETURNS THE PATH NUMBER USED BY THE MOST RECENT EXECUTION OF THE I/O STATEMENT ASSOCIATED WITH <IOSTATEMENT IDENTIFIER>.

<IOSTATEMENT IDENTIFIER> (IOCW.ADDRESS)

THIS PRIMARY RETURNS THE LOWER 28 BITS OF THE IOCW USED BY THE MOST RECENT EXECUTION OF THE I/O STATEMENT ASSOCIATED WITH <IOSTATEMENT IDENTIFIER>.

<IOSTATEMENT IDENTIFIER> (IOCW.CONTROL)

THIS STATEMENT RETURNS THE UPPER 20 BITS OF THE IOCW USED BY THE MOST RECENT EXECUTION OF THE I/O STATEMENT ASSOCIATED WITH <IOSTATEMENT IDENTIFIER>."

Modify the syntax of I/O STATEMENTS on page 3-17 as follows:

```
"SYNTAX:
 ------

 <I/O STATEMENTS> ::= <I/O STATEMENT> /
       * <IOSTATEMENT IDENTIFIER> * <I/O STATEMENT>
 <I/O STATEMENT> ::= <BACKSPACE I/O STATEMENT> /
                                        .
                                        .
                                        ."
```

Add the following to the examples on page 3-18:

"15. *READPK* READ ...;"

Add the following paragraph to the semantics on page 3-19:

"THE EXECUTION OF AN <I/O STATEMENT> PRECEDED BY AN <IOSTATEMENT IDENTIFIER> ASSOCIATES THE <I/O STATEMENT> WITH THE <IOSTATEMENT IDENTIFIER>. THE <IOSTATEMENT IDENTIFIER> MAY BE REFERENCED AS A <PRIMARY> IN ORDER TO GET CERTAIN CHARACTERISTICS OF THE LAST EXECUTION OF THE <I/O STATEMENT>. THE <IOSTATEMENT IDENTIFIER> MAY BE USED IN A LOG STATEMENT IN ORDER TO LOG THE RESULTS OF THE LAST EXECUTION OF THE <I/O STATEMENT>."

Add the following to the right side of <DECLARATION> on page 4-2:

"/ <IOSTATEMENT DECLARATION>"

Add the following subsection to SECTION 4:

"IOSTATEMENT DECLARATION

```
SYNTAX:
------

<IOSTATEMENT DECLARATION> ::= IOSTATEMENT <IOSTATEMENT IDENTIFIER>
<IOSTATEMENT IDENTIFIER> ::= <IDENTIFIER>

EXAMPLE:
-------

     IOSTATEMENT READPK;
```

SEMANTICS:

AN <IOSTATEMENT DECLARATION> DECLARES THE <IDENTIFIER> AS AN <IOSTATEMENT IDENTIFIER>. IF ANY <I/O STATEMENT> IS LABELED *<IOSTATEMENT IDENTIFIER>*, THE <IOSTATEMENT IDENTIFIER> MUST BE DECLARED. THE <IOSTATEMENT IDENTIFIER> MAY BE REFERENCED AS A <PRIMARY> OR IN A LOG STATEMENT."

Add the following paragraph to the semantics for SUBROUTINE DECLARATION on page 4-10:

"IOSTATEMENT IDENTIFIERS MAY BE PASSED TO SUBROUTINES ONLY AS CALL-BY-VALUE PARAMETERS. IF A PARAMETER OF A SUBROUTINE DECLARATION IS AN IOSTATEMENT IDENTIFIER, IT MUST BE DECLARED AS AN IOSTATEMENT IDENTIFIER WITHIN THE SUBROUTINE, AND AN I/O STATEMENT WITHIN THE SUBROUTINE MAY NOT BE LABELED WITH THIS IOSTATEMENT IDENTIFIER. FOR EXAMPLE, CONSIDER THE FOLLOWING PROGRAM:

```
      BUFFER A = 80 CHARACTERS;                         %  1
      VARIABLE CURMPX;                                  %  2
      VARIABLE CURRES;                                  %  3
      IOSTATEMENT READPK;                               %  4
      SUBROUTINE WINGIT(LASTIO);                        %  5
            IOSTATEMENT LASTIO;                         %  6
            BEGIN                                       %  7
            SET CURMPX = LASTIO(MPXNO);                 %  8
            STATUS PK 67;                               %  9
            END;                                        % 10
      FOR 5 DO                                          % 11
*READPK*  READ PK 67 CYLINDER = 2 INTO BUFFER A;        % 12
      CALL WINGIT(READPK);                              % 13
      SET CURRES = READPK(RESULT);                      % 14
      LOG RESULTS OF READPK;                            % 15
```

"CURMPX" IS SET TO THE MULTIPLEXOR NUMBER OF THE FIFTH EXECUTION OF THE I/O STATEMENT AT LINE 12. "CURRES" IS SET TO THE ERROR FIELD OF THE RESULT DESCRIPTOR OF THE FIFTH EXECUTION OF THE SAME I/O STATEMENT. THE RESULTS OF THIS I/O STATEMENT ARE LOGGED AT LINE 15. THE I/O STATEMENT AT LINE 9 MAY NOT BE LABELED BY THE IOSTATEMENT IDENTIFIER "LASTIO"."

SCR

P1014 SCR - "TD830," DISKPACK "235" CORRECTIONS

TD830 now returns CR only instead of CR-LF.

SCR and MAT now can run on BX385-DPK235.

P1171 SCR - "VERIFY" DOES NOT RECOGNIZE "5N" DISK

DISK VERIFY can now run on 5N disk.

P1172 SCR - DEADLY EMBRACE

A deadly embrace between SCR and CONTROLLER has been prevented.

P1173 SCR - ADDRESS CYLINDER "811"

SCR can now address cylinder 811 on disk pack type 235.

P1174 SCR - SERIAL NUMBER EXPANDED

The SCR system serial number has been expanded from 3 to 4 digits.

P1175 SCR - "SCR" TEST "9"

SCR Test 9 now works properly on disk pack type 235.

P1176 SCR - DEFAULT "VERSION"

"$ VERSION ML.NN" now sets the version to "ML.NN.000".

P1177 SCR - BUFFER FILL CORRECTION

Buffer fill of N"X" has been corrected.

P1241 SCR - MAKE FILE CARD "FILETYPE=8"

The file card (patch deck or symbol) is now made FILETYPE=8 at run time so that the file card can come from disk or disk pack by using a file-equation card.

P1242 SCR - RETURN SAVED MEMORY

The system now returns saved memory when address is used in a buffer declaration.

P1243 SCR - NO DECODE OF "RD=1FFFF"

The system no longer decodes all bits on a result descriptor of 1FFFF (I/O timeout).

P1244 SCR - "NRZ" TAPES

SCR MAKE TESTAPE now works correctly for 7- and 9-track NRZ tapes.

P1245 SCR - "SCR" "HI TESTAPE" MESSAGE

An extra Cr (carriage return) in the MAKE TESTAPE response to a Hi message, which caused an overflow in ADM to line 1, has been removed.

P1246 SCR - "TESTOP RD" ANALYSIS

SCR now decodes TESTOP RD results correctly.

P1247 SCR - PACK UNIT SUBTYPE FIELD

SCR disk pack unit subtype field has been corrected. SCR now allows I/O to greater than cylinder 405 for disk pack type 235.

P1251 SCR - "SCR MARKNO III"

The SCR MARKNO is now III in SCR printouts.

P1252 SCR - "SCR" VS. FIRST ACTION

When SCR got a "first action" result descriptor from a disk pack unit, it was possible for a dump by "IOERROR LEFT ON" and for all I/O's to the pack to hang. This has been corrected.

P1307 SCR - ALLOW "SET, RESET, POP" OF "VOIDT"

The system now allows SET, RESET and POP of VOIDT if MERGING is set.

P1764 SCR - "B6700/B7700" COMPATIBILITY

The following changes have been made:

1. B7700 commonality patches have been included.

2. Magnetic tape unit type on error listings from magnetic tape confidence is now reported correctly.

3. Result descriptor on error listings in all cases is now reported correctly.

P2430 SCR - NEW HIGH ORDER PRIORITY SCHEME

The priority of a process stack for acquiring a processor must often be raised above the normal visible priority range 0-99. The exalted priority is sometimes permanent, as for certain independent runners, MCS programs, and the like; for other stacks, the priority must be raised temporarily, as during a period that MCP global locks are held. (If a process cannot reacquire a processor to unlock some resource, many other processes can be starved of that resource.)

The MCP keeps processor priority in a three-part field. The declared visible priority is in the middle, with fine priority (biasing towards I/O-bound tasks) at the low end and the special-case priorities in the high end. The management of this high-order field has been changed.

Since the II.8 release, the MCP has initialized this field, called PRIORCLASSF, to various values according to the type of process. Then the field was used as a counter, bumped up and down as the code PROCUREd and LIBERATEd events, etc. In the new scheme, the field is divided into several subfields which may be used orthogonally. The function of lock counting has been moved to a new word in the base of the stack; one bit in the priority field records the non-zero state of that counter. The field as a whole is called PRIORBIASF; the subfields are as follows:

HIGHESTBIASF	[47:1]	Set for invisible independent runners.
LOCKBIASF	[46:1]	Set if LOCKCOUNT is non-0; used for soft locks and similar resources.
IRBIASF	[45:2]	Used for special independent runners, both visible and invisible. Among the former are AUTOBACKUP and DCCONTROL.
MCSBIASF	[42:2]	Used for special user programs: an MCS gets 2, a CPed non-MCS gets 1.
DSEDBIASF	[39:1]	Used to accelerate a DSed process off the system, freeing any resources it has used.
INTERACTIVEBIASF	[38:1]	Set for a swap task currently in an interactive (not time-sliced) mode.
JOBBIASF	[37:1]	Set for all WFL JOB stacks.

(The fields have been described in the positive sense; the data are actually kept in ones-complemented form to utilize the LLLU operator.)

In an additional change, any time PRESENCEBIT locks a descriptor, the LOCKCOUNT is incremented as though an event had been PROCURED.

P9229 SCR - "ANABOLISM" VS "MAKEJOBFILE"

It was possible to hang the system when starting a SCR, RES, RET or XD task. If a disk pack I/O error occurred at the same time INITIATE (who was called by ANABOLISM) called MAKEJOBFILE, ANABOLISM could hang waiting for a lock. This problem has been corrected.

P9285 SCR - TASK RESTRUCTURING

The task and base-of-stack structures have been reorganized; many items were moved from the stack to the task.

The changes were required for B6800 multiprocessor systems. They will also improve control and monitoring of swap jobs. Most interaction that previously was rejected for swap tasks is now processed normally.

SCTABLEGEN

D2648 SCTABLEGEN - ADD RESERVED WORDS

The following reserved words have been added:

AT
FROM
GLOBAL=G

D2681 SCTABLEGEN - NEW "ODT" OPTION "IODIAGNOSTICS"

A new ODT option, IODIAGNOSTICS, has been implemented. When set, it provides extensive error checking in the PHYSICALIO module's interface. Memory dumps are taken when errors are detected. When reset, no error checking is performed. This is option #36.

SCTABLEGEN

P1015 SCTABLEGEN - ADD RESERVED WORDS TO DICTIONARY

The following have been added to the dictionary of reserved words:

GLOBAL
GLO
LOCAL
LOC
MCSNAME
MCS

P1080 SCTABLEGEN - REMOVE UNUSED "TO"

Since the following test options are no longer referenced in the MCP, they have been removed from the test option list in the CONTROLLER via SCTABLEGEN:

RETENTION (0)
RECONDUMP (11)

SORT

P1016 SORT - "KANGAROO"

Problems often occurred in handling the HISTORY and ABORTEDRCW of SORT jobs on multiprocessor and SWAPPER systems. This problem has been corrected.

P1710 SORT - SECURITY VIOLATION ON "SORT/STATISTICX"

A problem has been corrected where a security violation could occur if a non-privileged user created and then attempted to lock the SORT/STATISTICX file.

P1711 SORT - "MERGE" VS. "SWAPPER"

A MERGE being run in SWAPSPACE will no longer fail as a result of invalid memory addresses being returned in the stack subsequent to a swap-in.

P1854 SORT - ERROR FOR INSUFFICIENT CORE

SORT would get an INVALID OP if (CORE SIZE)/(RECORD SIZE) < 1 and disk size and number of tapes were zero. Now, SORT gives ERROR #3 when it is passed these parameters.

P9285 SORT - TASK RESTRUCTURING

The task and base-of-stack structures have been reorganized; many items were moved from the stack to the task.

The changes were required for B6800 multiprocessor systems. They will also improve control and monitoring of swap jobs. Most interaction that previously was rejected for swap tasks is now processed normally.

SORTMCP

D2943 SORTMCP - "MCP" SYMBOLIC CONSOLIDATION

The external procedures MAINTENANCE and SORT have been moved into the MCP symbolic. They can no longer be separately compiled and bound; however, they can be changed using SEPCOMP.

SOURCENDL

D2698 SOURCENDL - MESSAGE ORIENTED DATACOM "(BBSC)"

SYMBOL/SOURCENDL has been updated to include the line control and message request sets for use with the new message-oriented broadband bisync control (BBSC), together with sample terminal, station, line and DCP definitions. In addition, appropriate changes have been made in the other requests and controls to allow them to operate correctly in a message-oriented datacom system and make use of features like auditing, remote filemode output, and autonomous operation. However, since many sites may not require these new features at present and also to allow continued use of older versions of the NDL compiler, all these new statements are conditionally omitted or included in SOURCENDL as controlled by a dollarcard called "ENHANCED" at sequence number 00000040. The default state of ENHANCED is RESET, so that the new statements are omitted.

D2699 SOURCENDL - "AUXLOGIC" REDUCES LOCAL MEMORY USAGE

Since its implementation in II.9 (see SOURCENDL note D1884), auxiliary logic usage has been available to reduce local memory requirements by placing parts or all of various request sets in main memory. The error recovery and "?" command processing portions of all requests in SYMBOL/SOURCENDL have been moved into main memory by surrounding each section of code with "$SET AUXLOGIC=AUXCODE" . . . "$POP AUXLOGIC". All usages of auxiliary logic in SYMBOL/SOURCENDL can be enabled/disabled by setting/resetting a single dollarcard option "AUXCODE" at sequence number 00000050. The default state of AUXCODE is SET.

D2700 SOURCENDL - AUTOMATIC RECALL OF OUTPUT MESSAGES

Currently, output messages to a powered-down poll/select device are kept around almost forever, even though the station may not have responded to a select for hours. Besides using main memory and the DCP's time in an unnecessary manner, if one of the messages is CANDE's "TIMELIMIT ELAPSED" message, a further bad action occurs. CANDE waits for a response saying that the message was actually received (hours, if necessary). Then it recalls all messages for that station, discards any input, and resets everything associated with the station. The user usually learns of this when he turns on his terminal in the morning, enters his usercode/password, and suddenly receives half a dozen messages sent the night before, including a timelimit elapsed message which purges his input. That problem has been corrected by using the RETRY value to count the number of times a select is done with no response (terminal powered down) and then simulates a "?BRK" input to cause CANDE to recall all its outstanding output messages.

D2701 SOURCENDL - IMPLEMENTATION OF "?BRK" FOR SCREENTTY

CANDE now treats a BREAK condition on transmit as a signal to recall all outstanding output to the station or, if none, to discontinue the command it is executing and to proceed with the next one. The "?BRK" input is used in the POLLTD820 request as the signal from the user to simulate the break on transmit condition. This implementation adds that same ability to the READSCREENTTY request.

SOURCENDL

P2087 SOURCENDL - CORRECT SCROLLING ON "TD820/TD830" REQUESTS

Two major problems have been corrected. First, when in scrolling mode with the POLLTD820 AND SELECTTD820 request sets, input and output which contain more than 80 characters (total, or between embedded carriage returns) will now be properly scrolled. Second, output whose length is an exact multiple of 80 characters will no longer be displayed with a blank line between each line of output.

Numerous other changes and improvements have been made to the POLLTD820 and SELECTTD820 request sets in the process of implementing these corrections. Most important is the implementation of DCP paging, enabled by "?+P" and disabled by "?-P". (Note that a TD830 must be configured for DC1 programmatic mode control and must not automatically remain in receive mode if paging is to work correctly.) Paging is accomplished by dropping the terminal into LOCAL mode when the proper number of messages have been received to fill the display. Pressing the RCV (Receive) key allows the next page of information to be displayed. The page size and initial state of paging (enabled/disabled) is specified in the station definition.

The other major change immediately apparent is that the request sets have been reformatted and rewritten to use alphanumeric labels and defines for all tallys and togs, thus making them much easier to read and understand. Other minor changes are: (1) transparent output is now really transparent (the line is not cleared first, (2) SKIP <n> carriage control now homes the cursor and then skips <n>-1 lines down (without changing lines skipped over), and (3) NDL "?" command processing has been made more efficient.

P2088 SOURCENDL - CHANGES TO "POLLCONTENTION" LINE CONTROL

As implemented, the POLLCONTENTION line control routine stays in contention mode only half a second, then exits to check if any queued output can be sent, and returns (still in contention mode) to wait another half second if it finds no queued output to send. There are two problems with this implementation. One, contention mode is exited frequently when it need not be; two, a station on a single-drop (or even multi-drop) line may sometimes see noise on the line and leave contention mode without the DCP being aware that it has. In the latter case, since the DCP is not polling the station, it is unable to transmit until contention mode is restored; therefore, the following changes have been made to POLLCONTENTION. As long as output is queued for some powered-up station on the previous poll/select cycle, contention mode is exited every 150 milliseconds (giving faster response than the current value of 500 milliseconds). When no output is queued, however, contention mode will be exited every 30 seconds and the "go to contention" message will be reissued; thus, noise on the line cannot make a station unable to transmit for more than 30 seconds.

P2089 SOURCENDL - PREVENT INFINITE LOOP

When RJE terminals are attached to the B6700 datacom subsystem, they may get into infinite loops under some error conditions. Specifically, these conditions occur with synchronous modems with constant carriers if the terminal sends all zeroes on the line when it is not transmitting. This problem has been corrected.

P2090 SOURCENDL - CHANGES TO "POLLDISPLAYS" REQUEST

Several problems in the POLLDISPLAYS request set have been corrected. The receive code to discard a sequence mode form was checking for US instead of for RS as its first character. If the form was not present and a control input were entered, no checking was done for "?BRK". If "?BRK" were entered, it was being returned as an input message instead of being discarded. Finally, group polling was not implemented properly for receiving another message automatically after the ACK of the first one.

P2091 SOURCENDL - CORRECT TRANSMISSION OF "NUL" CHARACTER

A recent change to the SELECTTD800 request set (see II.9 note P9303) allowed the NUL character to be transmitted rather than discarded; however, it should not be counted, since it is not a visible character. Counting the NUL leads to the cursor not being properly positioned in column one after the transmission. This problem has been corrected; now the NUL is transmitted without being counted.

P2092 SOURCENDL - PREVENT RESET OF TALLY BIT

In the RJE request set, the DLE padding control bit is no longer reset incorrectly.

P2093 SOURCENDL - RECOGNIZE "ENQ" CORRECTLY

SOURCENDL now handles DLE ENQ and ENQ correctly for RJE.

P2203 SOURCENDL - LINE "TOGS" AND "TALLYS" CLEARED

The RJE request sets will now clear all line TOGS and TALLYS at disconnect time for switched, leased and direct connect lines. This change will eliminate the failure at log-on time of the terminal and subsequent host rejection of messages due to ARM character mismatch.

STATSTOPPER

D2912 STATSTOPPER - FILE NAMES, NEW MODES, FLAG SIGMA<DELTA

The name (INTNAME and default TITLE) has been changed for two files manipulated by STATSTOPPER:

1. The name of the input file of procedure names and numbers has been changed from COUNTS to STATNAMES. This file is generated by the compiler; the corresponding name change has been made in NEWP but not in ESPOL, so either the ESPOL file COUNTS or the STATSTOPPER file STATNAMES must be equated.

2. The name of the intermediate file of statistical data has been changed from SRT to STATS.

Two new modes of operation have been provided:

1. If the TASKVALUE for STATSTOPPER is set odd, the program will produce only its first report (in order of procedure declaration); it will skip all sorting and the other five reports.

2. If the file STATS is equated (NEWFILE=FALSE), the program will not make any GETSTATUS calls to shut off statistics; instead it will read the STATS file produced by an earlier run of STATSTOPPER and generate the reports from those data.

In the first report, any line on which the total DELTA time exceeds the total SIGMA time will be flagged "** S<D". This anomaly should not occur if the statistics mechanism is working properly.

STATSTOPPER

P2220 STATSTOPPER - OUTPUT FORMATTING

The output format has been changed so that longer statistics runs may be made without overflowing the format.

SUSPENDER

P1890 SUSPENDER - MULTIPROCESSOR SYSTEM "SUSPENDER"

SUSPENDER now runs on a B6800 Multiprocessor system as well as a monolith. SUSPENDER will control thrashing in a B6800 Multiprocessor system on a box-by-box basis. The output format used by SUSPENDER when the PRINT or MONITOR option is set is slightly different. The action format is now as follows:

<box no. where action occurred> <action> <mix no. that action affected>

The rest of SUSPENDER's output remains unchanged in format.

SUSPENDER will not monitor nor attempt to control the mix in GLOBAL on a B6800 Multiprocessor (Tightly-Coupled) system.

UDSTRUCTURE TABLE

D2505 UDSTRCTTAB - ACCESSCODE

Two nodes have been added to the USERDATAFILE. The "ACCESSCODELIST" is a list of accesscodes with optional crunched passwords. "ACCESSCODENEEDED" is a bit which may be set to require an accesscode whenever a user logs onto CANDE or RJE.

D2801 UDSTRCTTAB - RECOMPILATION OF "UDSTRUCTURETABLE"

The example shown on Page 9-2-12 of the B7000/B6000 System Software Operational Guide (Form No. 5001688-2) for recompilation of the UDSTRUCTURETABLE is incorrect. The following example should replace the example given:

```
<i> BEGIN JOB;
    COMPILE SEP/UDSTRUCTURETABLE WITH USERSTRUCTURE TO LIBRARY;
    COMPILER FILE CARD=SYMBOL/UDSTRUCTURE ON DISK;
<i> BIND NEW/INTRINSICS WITH BINDER TO LIBRARY;
    COMPILER DATA
    HOST IS SYSTEM/INTRINSICS;
    BIND UDSTRUCTURETABLE FROM SEP/=;
<i> END JOB
```

D2892 UDSTRCTTAB - "SYSTEMUSER" BIT

Bit 45 in node 1 is now designated the SYSTEMUSER bit for distributed processing. When this bit is set in the usercode entry, it designates the usercode as one associated with inter-host system communicati'ons (i.e., non-user communicatins). This bit may be interrogated using the locater SYSTEMUSER, as follows:

MAKEUSER Syntax:

```
-- MU --------- SYSTEMUSER --|
        |     |
        |- + -|
        |     |
        |- - -|
```

Sample USERDATA Interface:

```
ARRAY ENTRY[0:255];
REAL UDRSLT;
<assume array entry already contains the usercode entry>

UDRSLT := USERDATA(,0,USERDATALOCATER("SYSTEMUSER"),0,ENTRY);

IF BOOLEAN(UDRSLT).[46:1] THEN
     <usercode in entry is a system usercode>

ELSE
     <usercode in entry is a normal (non-system) usercode>
```

USERSTRUCTURE COMPILER

D2506 USERSTRUCT - ACCESSCODE

A new type of typed node has been added to the USERDATASTRUCTURE, the ACCPSW LIST (node type 25), for storing accesscodes and passwords. The passwords are crunched, using the same procedure used to crunch usercode passwords. The accesscode/password combination is treated like a file name, consisting of either one name (an accesscode with no password), or as a name, a slash, and a 6-character crunched password:

```
ACCESS/"??????".
```

WORK FLOW LANGUAGE

D2733 WFL - FLEXIBILITY IN "REMOVE, CHANGE" STATEMENTS

The following changes have been made to the syntax for the WFL REMOVE and CHANGE statements, to provide increased flexibility:

1. Familynames may now be used as names for individual volumes. For example:

```
REMOVE X FROM PACK(KIND=DISK);
```

2. The KIND attribute may be specified for any volume name. If KIND is not specified, the WFL compiler will use the following default values:

Volume Name	Default Kind
DISK	DISK
PACK	PACK
<other>	TAPE

3. Remove and Change statements now allow SERIALNO specifications.

4. INTERCHANGE and SERIALNO attributes may be specified on all DISK and PACK families.

D2741 WFL - ACCESSCODE

The only part of this discussion which will apply to pre-II.9 WFL is the ACCESSCODE task attribute.

An accesscode may be applied to a job or task. If no accesscode is supplied for a task, it will inherit the job's accesscode.

```
-- ACCESSCODE ------------------------------------------|
               |                                       |
               |- = --<accesscode>-------------------|
                                     |                 |
                                     |- / --<password>-|
```

"ACCESSCODE" specifies that the job (or task) has no accesscode.

"ACCESSCODE=<accesscode>" specifies the accesscode for the job (or task). The accesscode/password will be validated in the USERDATAFILE.

WFL will understand the ACCESSCODE task attribute whenever it is supplied with a task, as described for CANDE. Supplying an accesscode for a task does not change the job's accesscode.

The accesscode password will be suppressed in the job summary printout.

The "SECURITY" WFL statement will accept "CONTROLLED <filename>".

A new executable statement has been added to WFL so that the password on the current accesscode of the job may be changed.

```
-- ACCESS -- PASSWORD -- = --<new password>--|
```

At the time this statement is executed, the job must have a usercode and a valid accesscode. The password for the current accesscode is changed to <new password>.

D2752 WFL - NEW COMPILER NAMES

The following compiler names should be added to the <compiler name list> on page 8-9 of the Work Flow Language Reference Manual (Form No. 5001555):

COBOL74
LCOBOL
NEWP

D2793 WFL - "OF <HOSTNAME>" RECOGNIZED IN FILE TITLES

WFL now recognizes "OF <hostname>" (as well as "ON <packname>") in file titles for use with Shared Resource systems and Burroughs Network Architecture. If both "ON <packname>" and "OF <hostname>" occur, "OF <hostname>" must occur last. The following is an example of a valid file title:

```
ABC ON PACK2 OF HOST2
```

D2894 WFL - "WFL" MANUAL CORRECTIONS

The following corrections should be made to the Work Flow Language Reference Manual (Form No. 5001555):

1. The note on page 6-9 of the WFL manual should be changed to read as follows:

   NOTE

   For the II.9, III.0 and III.1 System Releases, <serial number> must be a constant.

2. The following note should be added to page 7-1 of the WFL manual, after the railroad diagram for the <"single" file title>:

   NOTE

   For the II.9, III.0 and III.1 System Releases, the <"single" file title> has NOT been implemented.

3. The note on page 7-2 of the WFL manual should be changed to read as follows:

   NOTE

   For the II.9, III.0 and III.1 System Releases, the <"improved" file title list> has NOT been implemented.

4. The following note appears on pages 8-8, 8-15, 8-33 and 8-41 of the WFL manual:

   NOTE

   For the II.9 System Release, the <"improved" file title list> has NOT been implemented.

   On pages 8-8, 8-15 and 8-33, the note should be changed to read as follows:

   NOTE

   For the II.9, III.0 and III.1 System Releases, the <"improved" file title list> and <"single" file title> have NOT been implemented.

   On page 8-41, the note should read as follows:

   NOTE

   For the II.9, III.0 and III.1 System Releases, the <"improved" file title list>, <"single" file title>, and <"old" file name list> have NOT been implemented. One single-level file name must be used, as shown in the examples above.

D2895 WFL - FILE ATTRIBUTE "SERIALNO" NOT IMPLEMENTED

The following note should be added to page 4-8 of the Work Flow Language Manual (Form No. 5001555), after the railroad diagram:

NOTE

For the II.9, III.0 and III.1 System Releases, the ability to the use file attribute SERIALNO as a string primary has NOT been implemented.

D2896 WFL - CHANGE FILENAME USING "(PACK)"

Previously, an attempt to change a filename using a "(PACK)" specification would not change the filename. In addition, no error message or warning would be displayed to indicate that the filename was not changed. If the "(PACK)" specification were removed, the filename would be changed.

WFL will now change filenames when the (PACK) specification is included.

D2924 WFL - VARIABLE NOT USER AND SYSTEM DEFINED

On page 3-2 of the Work Flow Language Reference Manual (Form No. 5001555), the following paragraph should be added:

> "System attributes and mnemonics (e.g., "DISK") may be redefined by the user within a WFL job. When this is done, the system's definition for that variable is not valid in the job. For example, if F is a <file id>, then the following is not valid:
>
> ```
> STRING DISK;
> DISK:="PRINTER";
> F(KIND=DISK);
> F(KIND=#DISK);
> ```
>
> "DISK" must only be used as DISK or "PRINTER"."

On page 4-11, the following should be omitted:

> "F(KIND=DISK) yields KIND=DISK".

D2925 WFL - "WFL" JOB TRANSFER TO/FROM REMOTE HOST

WFL jobs may now be sent from one host to another, using an "AT <hostname>" between the invalid character and the "BEGIN JOB" on the first card. The following example will send the job to HOSTC:

```
?AT HOSTC BEGIN JOB X;
.
.
.
?END JOB
```

The WFL compiler at the receiving host system compiles the job; the sending host does not analyze the contents of the job (except for the ?END JOB).

D2926 WFL - COPY FILES TO REMOTE HOSTS

The file attribute "HOSTNAME=<hostname>" will cause a specified file to be copied to the specified host via the COPY statement.

Example:

```
COPY ABC FROM BLUE(KIND=PACK, HOSTNAME=HOSTA) TO RED
   KIND=PACK, HOSTNAME=HOSTB)
```

The above example will transfer file "ABC" from the BLUE pack on HOSTA to the RED pack on HOSTB.

The COPY statement will only transfer files from Post-II.9 WFL jobs; "old" WFL jobs containing the HOSTNAME attribute will get a syntax error.

D2931 WFL - TASK ID IN TASK FAMILY DECLARATION

The following information should be added to the Work Flow Language Reference Manual (Form No. 5001555):

On page 3-2, add a new paragraph after the second paragraph, as follows:

> "File and task attributes in a file or task declaration may only be assigned constant values."

On page 6-4, add the following sentence to the first paragraph:

> "Attribute values in <task declaration> statements must be constants or constant expressions."

On page 6-8, add the following sentence to the first paragraph:

> "Attribute values in <file declaration> statements must be constants or constant expressions."

D2932 WFL - "STATUS" ATTRIBUTE

The following should be added at the end of the STATUS task attribute description on page B-6 of the Work Flow Language Manual (Form No. 5001555):

> "The STATUS task attribute may be set in a task declaration statement, but may not be used in the task equation list of a task initiation. Thus, in the following jobs, job (1) is valid and job (2) is not valid:

```
(1) 100 ?BEGIN JOB STEST;             (2) 100 ?BEGIN JOB STEST2;
    200 TASK T(STATUS=NEVERUSED);         200 TASK T;
    300 RUN X[T];                         300 RUN X[T];
    400 ?ENDJOB.                          400 STATUS=NEVERUSED;
                                          500 ?ENDJOB."
```

Job (2) will now get a syntax error on line 400.

WORK FLOW LANGUAGE

P1830 WFL - PREVENT "SEG ARRAY" ERROR WHEN JOB RESTARTS

A SEG ARRAY error will no longer occur when a job is restarted in a subroutine with parameters.

P1873 WFL - INCORRECT CODE FOR FILE TITLE ASSIGNMENTS

A problem has been corrected whereby some dynamic file titles could cause incorrect code to be generated for a subsequent file title assignment. This could cause a FAULT IN DO CODE system dump. The following is an example of the kind of job for which incorrect code was generated:

```
<i>BEGIN JOB X;
 FILE GRDTST;
 STRING STR;
 STR :="C";
 IF FILE A/B/#STR IS RESIDENT THEN;
 CRDTST(TITLE=CARDTEST);
<i>END JOB
```

P1875 WFL - CONCATENATION OF MORE THAN "15" STRINGS

WFL will no longer generate incorrect code for concatenation of more than fifteen string expressions.

P2001 WFL - ZIP WITH ARRAY

A zip with a long array will no longer result in a SEG ARRAY error in the WFL compiler.

P2002 WFL - ZIP WITH ARRAY

Previously, a zip with a long array could result in a MSG SIZE error in the WFL compiler. WFL will no longer generate the MSG SIZE error.

P2007 WFL - "REMOVE" COMMAND

Previously, an attempt to remove a file from a named pack using a (PACK) or (KIND=PACK) specification would not remove the file. In addition, no error message or warning would be displayed to indicate that the file was not removed. If the (PACK) or (KIND=PACK) specification was taken out of the statement, the file would be removed.

Example:

```
REMOVE X FROM ARK(PACK) would not remove file X
REMOVE X FROM ARK       would remove file X
```

WFL will now remove files from named packs when the (PACK) or (KIND=PACK) specifications are used.

P2008 WFL - "ELSE" NO LONGER IGNORED

Previously, if a REMOVE, CHANGE, SECURITY, START or RERUN statement appeared in the THEN portion of an IF-THEN-ELSE statement, the ELSE portion of the statement would be executed unconditionally unless a semicolon appeared immediately before the ELSE.

Example:

```
IF X THEN REMOVE TEST1 ELSE DISPLAY "TEST1 NOT REMOVED";
```

If X is true, TEST1 would be removed and the message would appear (erroneously). If X is false, the message would be displayed.

WFL will no longer incorrectly execute both parts of the conditional.

P2077 WFL - EXTRA "BEGIN" IN "WFL" JOB DECK

An extra BEGIN at the end of a ?BEGIN JOB card caused the WFL compiler to loop and print error messages.

The following sample job headings would all cause the looping to occur:

```
?BEGIN JOB X; CLASS=IO; BEGIN
?BEGIN JOB B; USER=XYZ; BEGIN
?BEGIN JOB K; BEGIN
```

The WFL compiler would display "MISSING END JOB CARD", the "?BEGIN JOB" card and the message "COMPILATION ABORTED". It would then attempt to compile the job again.

The WFL compiler will no longer loop under these circumstances.

P2116 WFL - "BCL" JOBFILE LIBRARY PROBLEM

WFL jobs that contain a BCL data deck, and were compiled with NEWSOURCE (or LIBRARY for pre-II.9 WFL), would not run correctly when reexecuted using the START WFL statement. This problem has been corrected.

P2117 WFL - AMBIGUOUS "COPY" STATEMENTS

Syntax errors will now be generated for ambiguous COPY AS statements. Examples of such ambiguities include the following:

```
COPY A/= AS B
COPY A AS B/=
COPY = AS =
```

P2118 WFL - ZIP WITH ARRAY LOOPING

The WFL compiler no longer gets into an infinite loop when the following conditions are true: (1) the job is a zipped array, (2) the job card gets a syntax error, (3) and there are no other statements to be compiled or an END JOB card is not seen.

P2119 WFL - INSTRUCTION BLOCKS

WFL will no longer get an INVALID INDEX on large INSTRUCTION blocks within a WFL job.

P2120 WFL - HISTORY

The following task attributes will now return the proper fields of the history word:

```
HISTORYTYPE
HISTORYREASON
HISTORYCAUSE
```

P2121 WFL - "WFL" FAULT IN ERROR HANDLING PROCEDURE

WFL will no longer get an INVALID INDEX on extraneous blank cards after the END JOB card.

P2122 WFL - RESERVED WORDS AS FAMILYNAMES

WFL will now allow reserved words as family names.

P2123 WFL - "SEG ARRAY" IN JOB STACK

WFL will no longer cause a job stack to get a SEG ARRAY error because a string constant in a RUN statement was improperly reused.

P2124 WFL - DECIMAL NUMBER

WFL will now correctly handle decimal numbers that are preceded only by a decimal point.

Example:

```
X + .5
```

P2125 WFL - FAULT ON BAD TASK ATTRIBUTE

WFL will no longer generate a "BAD TASK ATTRIBUTE" error message on COMPILE TO LIBRARY AND GO. The bad task attribute would appear on a subsequent running of the object file.

P2126 WFL - CHARGECODE VS. "DECK" STATEMENT

Chargecode is now allowed with DECK statements.

P2175 WFL - CHARGECODE VS. ONE STATEMENT "ODT" JOB

Chargecodes will now be allowed on one statement ODT jobs in pre-II.9 WFL.

P2464 WFL - WFL "BEGIN; JOB" CORRECTION

The WFL compiler will no longer loop when it encounters a job card of the following type.

```
?BEGIN; JOB
```

Instead, it will display the message:

```
"ILLEGAL JOB CARD"
```

P2471 WFL - "COMPILATION ABORTED" NOT TRUNCATED

Previously, the following message would be truncated on a remote terminal after "ABOR":

"***ERROR LIMIT EXCEEDED, COMPILATION ABORTED*****"

Now, the message will be displayed in its entirety.

P2509 WFL - ELIMINATE RECURSIVE HANDLING OF QUOTES

WFL will no longer handle missing quotes recursively, as this led to problems when WFL kept looking for the second quote even after it passed a record boundary.

P2510 WFL - "COMPILE" STATEMENT WITHOUT COMPILER NAME

Previously, when WFL encountered a COMPILE statement with no compiler name, it would continue reading cards as if they were WFL statements. If a DATA statement preceded by a compiler name (e.g., "ALGOL DATA") followed after the COMPILE statement, it was not recognized, so that the data deck was also read as WFL statements.

WFL will now give the correct syntax error for a missing compiler name, and will correctly handle data decks after that point.

XREF ANALYZER

D2898 XREFANALY - CHANGE HEADING

The XREFANALYZER will now print its own version in the form "ll.ccc.ppp", where "ll" is the software level, "ccc" is the cycle, and "ppp" is the most recent patch number.

XREF ANALYZER

P1556 XREFANALY - MORE INFORMATION IN "REF" FILE

Information about the procedure context in which a sequence numbers occurs has been appended to the REFs file.

P2206 XREFANALY - PORT AND SIGNAL ARRAYS

The INTERACTIVEXREF will now give information on port arrays and signal arrays.

P9193 XREFANALY - MEMORY REQUIREMENTS

All calls on system SORT will be disk sorts; no core-only sorts will be attempted. This corrects a problem which caused XREFANALYZER to ask for a large block of contiguous memory words.

SYSTEST - DCP/MAINTMCS
------- - ------------

D2890 MAINTMCS - EXTENSION TO "ADAPTER ADDRESS"

The patch mark for this note is 31.195.002.

ADAPTER ADDRESS (AA) syntax for DCP test execution has been extended to optionally allow !ine adapter position within a cluster to be specified. The value of the line adapter position may range from 0 to 15. If this optional parameter is omitted, position 0 is assumed. The extended syntax is intended for use in Cluster III tests (and is ignored by those tests which do not require this information). Both of the following examples specify line adapter position 0 of front end control 1 of basic control 2 of DCP2:

```
    <mix no>SM DCPTEST DCP 3, TEST (AA2:1:0)4-5
    <mix no>SM DCPTEST DCP 3, TEST (AA 2:1)3-8
```

SYSTEST - DCP/MAINTMCS

P2370 MAINTMCS - ERROR MESSAGE DISPLAYS CORRECT LENGTH TOKEN

The patch mark for this note is 31.195.001.

Certain erroneous syntax within a DCPTEST list of tests following a left parenthesis failed to display the error token. The token is now displayed correctly.

SYSTEST - DOCUMENTOR

P1832 DOCUMENTOR - UPDATE DOCUMENT PRINTER FOR "GMM" FILES

After adding "GMM" to the tables, the document program printed "BIC" in the directory. This problem has been corrected.

P1833 DOCUMENTOR - MODIFIED MARK # IN DIRECTORY PRINT FILE

Previously, the mark # printing was a define of a real #; now, it is a define to the compiler (20).

P2069 DOCUMENTOR - REMOVE "PROGRAMDOC" FROM EXCLUDED GROUP

SYMTEST/GMM/PROGRAMDOC has been removed from the EXCLUDEd group.

P2629 DOCUMENTOR - MISCELLANEOUS CHANGES

1. DOCUMENTOR would DS when it processed a symbolic without documentation which was not explicitly excluded. Now, DOCUMENTOR skips the symbolic and continues with the next one.
2. If DOCUMENTOR calls PROGRAMDUMP, it now prints files as well as arrays.

SYSTEST - IO/IOP

P2490 IOIOP - TAPE DENSITY

To test 1600/6250 tape units, enter the following from the ODT:

```
    SN MT <unit #> <serial #> (<tape density>)
```

The tape density, 1600 or 6250, selects the mode of operation.

SYSTEST - IO/IOTEST

P2630 IOTEST - INCLUDE "PK" AS UNITTYPE

IOTEST may be run for packs; therefore, PK may be included as unittype.

SYSTEST - OFF/MPX
------- - -------

P1836 MPX - "TD850" REVISIONS

The following changes have been made in handling TD850s:

A delay has been added to subsequent ODT writes such that the TD850s will be in a READY state prior to the next WRITE operation.

The capability has been added to recognize status change interrupts during the PRESENTCOLDSTART interrupt handling procedure to prevent loops under certain status change conditions.

The "ETX" character has been added at the end of ODT write data to ensure proper operations with TD830/850s.

The memory addresses for non-save identifiers have been changed to prevent sequence errors.

P1837 PEP - "TD850" TEST RUNS WITH NO READ BACK

When the device specified is a "SC2" and the option selected is "write only", a timer has been added to allow the TD850 to accomplish this.

When the device specisied is a "SC2" and the option selected if "write with read back", the "XMT" key may be depressed only after "RECEIVING" has been flashed three times on the respective ODT and before the next three "RECEIVING" appear.

P2621 PEP - TEST RESTRICTED TO PACK TYPES

The following tests (or utilities) have been restricted to 215 and 225 packs only:

IV (SCR function)
SUPERIV
PK15
PEP

The following tests (or utilities) have been restricted to 215, 225 and 235 packs only:

PKTEST
PK16
PK11
PK10
PK09
PK08
PK07
PK06
PK04

SYSTEST - SCR/PKXD

P2619 PKXD - MISCELLANEOUS CHANGES

PKXD has been modified, as follows:

1. It now runs for all disk pack types.
2. The interval of the XDed area to test has been corrected.
3. Now the address portion of the record read is compared to the address portion of the record written.

SYSTEST - SCR/PK04
------- - --------

P2621 PK04 - TEST RESTRICTED TO PACK TYPES

The following tests (or utilities) have been restricted to 215 and 225 packs only:

IV (SCR function)
SUPERIV
PK15
PEP

The following tests (or utilities) have been restricted to 215, 225 and 235 packs only:

PKTEST
PK16
PK11
PK10
PK09
PK08
PK07
PK06
PK04

SYSTEST - SCR/PK06

P2621 PK06 - TEST RESTRICTED TO PACK TYPES

The following tests (or utilities) have been restricted to 215 and 225 packs only:

IV (SCR function)
SUPERIV
PK15
PEP

The following tests (or utilities) have been restricted to 215, 225 and 235 packs only:

PKTEST
PK16
PK11
PK10
PK09
PK08
PK07
PK06
PK04

SYSTEST - SCR/PK07
------- - --------

P2621 PK07 - TEST RESTRICTED TO PACK TYPES

The following tests (or utilities) have been restricted to 215 and 225 packs only:

IV (SCR function)
SUPERIV
PK15
PEP

The following tests (or utilities) have been restricted to 215, 225 and 235 packs only:

PKTEST
PK16
PK11
PK10
PK09
PK08
PK07
PK06
PK04

SYSTEST - SCR/PK08

P2621 PK08 - TEST RESTRICTED TO PACK TYPES

The following tests (or utilities) have been restricted to 215 and 225 packs only:

IV (SCR function)
SUPERIV
PK15
PEP

The following tests (or utilities) have been restricted to 215, 225 and 235 packs only:

PKTEST
PK16
PK11
PK10
PK09
PK08
PK07
PK06
PK04

SYSTEST - SCR/PK09

P2621 PK09 - TEST RESTRICTED TO PACK TYPES

The following tests (or utilities) have been restricted to 215 and 225 packs only:

IV (SCR function)
SUPERIV
PK15
PEP

The following tests (or utilities) have been restricted to 215, 225 and 235 packs only:

PKTEST
PK16
PK11
PK10
PK09
PK08
PK07
PK06
PK04

SYSTEST - SCR/PK10

P2621 PK10 - TEST RESTRICTED TO PACK TYPES

The following tests (or utilities) have been restricted to 215 and 225 packs only:

IV (SCR function)
SUPERIV
PK15
PEP

The following tests (or utilities) have been restricted to 215, 225 and 235 packs only:

PKTEST
PK16
PK11
PK10
PK09
PK08
PK07
PK06
PK04

SYSTEST - SCR/PK11

P2621 PK11 - TEST RESTRICTED TO PACK TYPES

The following tests (or utilities) have been restricted to 215 and 225 packs only:

IV (SCR function)
SUPERIV
PK15
PEP

The following tests (or utilities) have been restricted to 215, 225 and 235 packs only:

PKTEST
PK16
PK11
PK10
PK09
PK08
PK07
PK06
PK04

SYSTEST - SCR/PK15

P2621 PK15 - TEST RESTRICTED TO PACK TYPES

The following tests (or utilities) have been restricted to 215 and 225 packs only:

IV (SCR function)
SUPERIV
PK15
PEP

The following tests (or utilities) have been restricted to 215, 225 and 235 packs only:

PKTEST
PK16
PK11
PK10
PK09
PK08
PK07
PK06
PK04

SYSTEST - SCR/PK16

P2621 PK16 - TEST RESTRICTED TO PACK TYPES

The following tests (or utilities) have been restricted to 215 and 225 packs only:

IV (SCR function)
SUPERIV
PK15
PEP

The following tests (or utilities) have been restricted to 215, 225 and 235 packs only:

PKTEST
PK16
PK11
PK10
PK09
PK08
PK07
PK06
PK04

SYSTEST - SCR/PKSCAN

P2072 PKSCAN - LOGGING AUTOCORRECTION "RD"

PKSCAN now correctly logs results to CHECKP and READPU.

P2616 PKSCAN - CHECKING FOR USED SPARE SECTORS

Checking for used spare sectors is now done only for old firmware.

P2617 PKSCAN - LOG ERROR FOR "AUTOCORRECT"

If AUTOCORRECT is set and an I/O gets an error, it is printed saying that the RD should be "C01". If the error is corrected (i.e., RD="C01"), the message is no longer printed.

P2618 PKSCAN - "PKSCAN" FOR WRITE DISABLED PACKS

Since PKSCAN does not do WRITEs to pack, write-disabled packs may now be scanned.

P2620 PKSCAN - WORK WITH ALL PACK TYPES

The PKSCAN utility now works with all types of disk packs.

SYSTEST - SCR/PKSEEK

P2631 PKSEEK - DISK PACK CONTROLLER ERROR MESSAGE

Previously, if the disk pack controller was not a BX380, BX383 or BX385, the program would give the following erroneous error message:

```
">>>ERROR<<< UNIT <#> NOT OF PACK TYPE
 >>>ERROR<<< UNIT <#> INVALID
 TYPE CONTROL BX<ctype> UNIT TYPE <utype>"
```

where <#>, <ctype>, <utype> are numbers. In particular, <ctype> would be the result of doing a scanin for controller type, not the BX number.

Now, the program prints the following error message:

```
">>>ERROR<<< CONTROLLER TYPE FOR UNIT <#> IS NOT BX380/383/385"
```

SYSTEST - SCR/PKBASIC

P2631 PKBASIC - DISK PACK CONTROLLER ERROR MESSAGE

Previously, if the disk pack controller was not a BX380, BX383 or BX385, the program would give the following erroneous error message:

```
"›››ERROR‹‹‹ UNIT <#> NOT OF PACK TYPE
 ›››ERROR‹‹‹ UNIT <#> INVALID
 TYPE CONTROL BX<ctype> UNIT TYPE <utype>"
```

where <#>, <ctype>, <utype> are numbers. In particular, <ctype> would be the result of doing a scanin for controller type, not the BX number.

Now, the program prints the following error message:

```
"›››ERROR‹‹‹ CONTROLLER TYPE FOR UNIT <#> IS NOT BX380/383/385"
```

SYSTEST - SCR/SUPERIV

P2076 SUPERIV - "SUPERIV" CORRECTIONS

1. SUPERIV will now detect whether the 2.0 or 1.5 firmware is running. If the 2.0 firmware is running the firmware itself will do the head positioning; thus, SUPERIV will not run extra verification cycles.

2. The VIA construct handling has been modified so that it will not expend excessive processor time. It is suggested that a nonexistent IOM (e.g., 4) be specified when asked to input the IOM number to bypass the VIA mechanism.

3. The message asking for "MPX#" has been modified to ask for "MPX# OR IOM#".

4. The value of 0 is now permitted as a legal IOM number. The maximum value permitted is still 4, since IOMs are not numbered above 3; this maintains compatibility with the B 6800/B 6700.

5. When asked to "SET HEADS CENTER <1 OR 2>", the reply should be "VALUE 2". The value of 1 is used for an earlier model controller when intervention was necessary to manually offset the heads for testing.

P2621 SUPERIV - TEST RESTRICTED TO PACK TYPES

The following tests (or utilities) have been restricted to 215 and 225 packs only:

IV (SCR function)
SUPERIV
PK15
PEP

The following tests (or utilities) have been restricted to 215, 225 and 235 packs only:

PKTEST
PK16
PK11
PK10
PK09
PK08
PK07
PK06
PK04

SYSTEST - SCR/PKHDOVER
------- - ------------

P2631 PKHDOVER - DISK PACK CONTROLLER ERROR MESSAGE

Previously, if the disk pack controller was not a BX380, BX383 or BX385, the program would give the following erroneous error message:

```
"&gt;&gt;&gt;ERROR&lt;&lt;&lt; UNIT &lt;#&gt; NOT OF PACK TYPE
 &gt;&gt;&gt;ERROR&lt;&lt;&lt; UNIT &lt;#&gt; INVALID
 TYPE CONTROL BX&lt;ctype&gt; UNIT TYPE &lt;utype&gt;"
```

where <#>, <ctype>, <utype> are numbers. In particular, <ctype> would be the result of doing a scanin for controller type, not the BX number.

Now, the program prints the following error message:

```
"&gt;&gt;&gt;ERROR&lt;&lt;&lt; CONTROLLER TYPE FOR UNIT &lt;#&gt; IS NOT BX380/383/385"
```

SYSTEST - SCR/PKINTERCH

P2631 PKINTERCH - DISK PACK CONTROLLER ERROR MESSAGE

Previously, if the disk pack controller was not a BX380, BX383 or BX385, the program would give the following erroneous error message:

```
"&gt;&gt;&gt;ERROR&lt;&lt;&lt; UNIT &lt;#&gt; NOT OF PACK TYPE
 &gt;&gt;&gt;ERROR&lt;&lt;&lt; UNIT &lt;#&gt; INVALID
 TYPE CONTROL BX&lt;ctype&gt; UNIT TYPE &lt;utype&gt;"
```

where <#>, <ctype>, <utype> are numbers. In particular, <ctype> would be the result of doing a scanin for controller type, not the BX number.

Now, the program prints the following error message:

```
"&gt;&gt;&gt;ERROR&lt;&lt;&lt; CONTROLLER TYPE FOR UNIT &lt;#&gt; IS NOT BX380/383/385"
```

SYSTEST - SCR/PKACTUATOR

P2631 PKACTUATOR - DISK PACK CONTROLLER ERROR MESSAGE

Previously, if the disk pack controller was not a BX380, BX383 or BX385, the program would give the following erroneous error message:

```
">>>ERROR<<< UNIT <#> NOT OF PACK TYPE
 >>>ERROR<<< UNIT <#> INVALID
 TYPE CONTROL BX<ctype> UNIT TYPE <utype>"
```

where <#>, <ctype>, <utype> are numbers. In particular, <ctype> would be the result of doing a scanin for controller type, not the BX number.

Now, the program prints the following error message:

```
">>>ERROR<<< CONTROLLER TYPE FOR UNIT <#> IS NOT BX380/383/385"
```

SYSTEST - SCR/PKHEADISOL

P2631 PKHEADISOL - DISK PACK CONTROLLER ERROR MESSAGE

Previously, if the disk pack controller was not a BX380, BX383 or BX385, the program would give the following erroneous error message:

```
    ">>>ERROR<<< UNIT <#> NOT OF PACK TYPE
     >>>ERROR<<< UNIT <#> INVALID
     TYPE CONTROL BX<ctype> UNIT TYPE <utype>"
```

where <#>, <ctype>, <utype> are numbers. In particular, <ctype> would be the result of doing a scanin for controller type, not the BX number.

Now, the program prints the following error message:

```
    ">>>ERROR<<< CONTROLLER TYPE FOR UNIT <#> IS NOT BX380/383/385"
```

SYSTEST - SCR/PKWRITEREAD

P2631 PKWRITEREAD - DISK PACK CONTROLLER ERROR MESSAGE

Previously, if the disk pack controller was not a BX380, BX383 or BX385, the program would give the following erroneous error message:

```
    ">>>ERROR<<< UNIT <#> NOT OF PACK TYPE
     >>>ERROR<<< UNIT <#> INVALID
     TYPE CONTROL BX<ctype> UNIT TYPE <utype>"
```

where <#>, <ctype>, <utype> are numbers. In particular, <ctype> would be the result of doing a scanin for controller type, not the BX number.

Now, the program prints the following error message:

```
    ">>>ERROR<<< CONTROLLER TYPE FOR UNIT <#> IS NOT BX380/383/385"
```

SYSTEST - UTIL/PKTEST

P2073 PKTEST - REMOTE USAGE OF "SPOTAB" OPTION

Executing PKTEST via remote terminals, the input "AX SPOTAB" will now show the display every time it is transmitted. File CON must be label-equated prior to execution.

P2074 PKTEST - "TAB" OPTION CAUSING "INVALID INDEX"

The use of the TAB option will no longer produce an erroneous INVALID INDEX.

P2075 PKTEST - DIRECT "I/O" HANDLING

PKTEST now reports all failing direct I/O result descriptors which occur during execution.

P2621 PKTEST - TEST RESTRICTED TO PACK TYPES

The following tests (or utilities) have been restricted to 215 and 225 packs only:

IV (SCR function)
SUPERIV
PK15
PEP

The following tests (or utilities) have been restricted to 215, 225 and 235 packs only:

PKTEST
PK16
PK11
PK10
PK09
PK08
PK07
PK06
PK04

SYSTEST - GMM/EVENTS

P1834 EVENTS - TEST "309" FAILED WITH "GSC" ABSENT

Test 309 had a GSC option. If GSC were not present, the test would fail because the MCP select did not get the scan bits off the 68-GMM interface. This problem has been corrected.

SYSTEST - GMM/FLIPFLOPS

D2977 FLIPFLOPS - ERROR MESSAGES IN HEX

Previously, scan in/out information was printed as a string of bits. Now, it will be done in hex.

SYSTEST - GMM/FUNCTIONAL

D2976 FUNCTIONAL - DELETED "MISSING PORT" MESSAGE

Because most * GLOBAL tm Memory systems have one or more ports absent, the "MISSING PORT" message will no longer be printed.

* "GLOBAL Memory" is a trademark of Burroughs Corporation.

SYSTEST - GMM/MEMTEST

P2071 MEMTEST - STOP PAGING WHILE LOOPING

The system will now ignore ODT paging option if looping.

SYSTEST - GMM/INTERPRETER

D2975 INTERPRETER - INTERRUPT ERROR HANDLING

The following changes have been made to the handling of hardware interrupts:

1. The printing of hardware interrupt error messages has been modified.
2. Tests have been added for EMNT and EVNT FFs while in SNAP compare mode.
3. Previously, the program terminated after hardware interrupts; now, it stays in the mix.

SYSTEST - GMM/INTERPRETER

P1835 INTERPRETER - "SNAP" COMPARE MODE

Priority resolution, which caused problems on SNAP compare when the multi-cabinet adapter was missing, has been corrected.

P2070 INTERPRETER - PRINT LAST AND CURRENT RECORDS

The following changes have been made:

1. The format of the GMM page dump has been changed from 64 half-pages to 32 full pages; pages in the printout now match the display pages.
2. Last and current snap records are printed when snap compare errors occur.
3. Miscellaneous changes have been made.

SYSTEST - DCP/BBSC

P1839 BBSC - CORRECT "ASCII" CHARACTER DEFINES

The ASCII control character defines for WAC and RVT were incorrect, which caused test 26 to fail. This only had an effect if the dollar option ASCII were SET. This has been corrected.

P2368 BBSC - REPLACE "$POP LIST," ALLOW "$SET LIST"

The patch mark for this note is 31.195.0002.

When LIST is set during compilation, the printout stops at line 25601600 due to "$RESET LIST" in the symbolic. The matching "$POP LIST" at line 25602000 was omitted due to the dollar sign being in column 1 instead of column 2. This problem has been corrected.

SYSTEST - DCP/AUTONBBSC

P1839 AUTONBBSC - CORRECT "ASCII" CHARACTER DEFINES

The ASCII control character defines for WAC and RVT were incorrect, which caused test 26 to fail. This only had an effect if the dollar option ASCII were SET. This has been corrected.

P2368 AUTONBBSC - REPLACE "$POP LIST," ALLOW "$SET LIST"

The patch mark for this note is 31.195.0002.

When LIST is set during compilation, the printout stops at line 25601600 due to "$RESET LIST" in the symbolic. The matching "$POP LIST" at line 25602000 was omitted due to the dollar sign being in column 1 instead of column 2. This problem has been corrected.

FTR ACTION TABLE

FTR	NOTE	SOFTWARE	PATCH	DESCRIPTION
0060-7145	D2860	BDMSCOBOL	31.0.	Use of <structure number> Cons
0060-7152	D2267	CONTROLLER	31.0.0011	Four vs Three Digit System Ser
0060-7152	D2267	DUMPANALY	31.0.0011	Four vs Three Digit System Ser
0060-7152	D2267	JOBFORMAT	31.0.0004	Four vs Three Digit System Ser
0060-7155	P2357	INTERFACE	31.0.0014	Link to Set Qualification
0060-7155	P2357	INTERFACE	31.0.0016	Link to Set Qualification
0060-7156	P1921	DASDL	31.0.0030	Global Chain Not Overlayed
0113-0819	P2302	BINDER	31.0.0044	Invalid PARAMETER MISMATCH
0113-0819	P2302	FORTRAN	31.0.0040	Invalid PARAMETER MISMATCH
0113-0819	P2302	MCP	31.0.0720	Invalid PARAMETER MISMATCH
0113-0879	D2815	BINDER	31.0.0015	Multiple BIND= Cards
0113-0882	P1516	BINDER	31.0.0012	Binding Entry Points
0113-0885	D2815	BINDER	31.0.0015	Multiple BIND= Cards
0113-0896	P1662	COBOL	31.0.0035	Exceptioncode for ANSI74 Corre
0114-0377	P2591	ESPOLINTRN	31.0.0067	FORTRAN BCL Formatting
0114-0385	P1081	MCP	31.0.0166	NON ANCESTRAL TASKFILE
0114-1001	P9146	MCP	31.0.0062	Reserve Locks
0114-1001	P9146	MCP	31.0.0073	Reserve Locks
0114-1001	P9146	MCP	31.0.0125	Reserve Locks
0114-1001	P9146	MCP	31.0.0135	Reserve Locks
0114-1004	P2238	ALGOL	31.0.0149	$PAGE and the Compiler Linecou
0114-1026	P1244	SCR	31.0.0025	NRZ Tapes
0114-1040	P1760	ALGOL	31.0.0037	No Error on "REAL (P:P EQL "AB
0114-1042	P1516	BINDER	31.0.0012	Binding Entry Points
0114-1046	D2623	ALGOL	31.0.0030	Equation Disallowed in Global
0114-1046	D2623	ALGOL	31.0.0031	Equation Disallowed in Global
0114-1046	D2623	ALGOL	31.0.0036	Equation Disallowed in Global
0114-1046	D2623	ALGOLTABLE	31.0.0005	Equation Disallowed in Global
0114-2001	P2289	ALGOL	31.0.0163	No "COMPILER ERROR IN BUILDITE
0114-2004	D2841	JOBFORMAT	31.0.0029	"MAJOR TYPE=2 MINOR TYPE=5" Vs
0114-2004	D2841	LOGANALY	31.0.0031	"MAJOR TYPE=2 MINOR TYPE=5" Vs
0114-3106	P2624	MCP	31.0.0875	Protection from Tag-7 Interrup
0116-0106	P1073	CONTROLLER	31.0.0015	ADMEVENT M
0116-0112	P1193	MCP	31.0.0213	FM Vs. SU
0116-0118	P2583	DATACOM	31.0.0979	Subtract Station Fault with No
0120-0170	D2249	MCP	31.0.0059	OF of GETSTATUS Waiting on <fi
0120-0184	D2778	ALGOL	31.0.	Files LINE and ERRORFILE
0120-0191	P1767	DUMPALL	31.0.0003	ROUTINE and COPY Statements Re
0120-0197	P2007	WFL	31.0.0021	REMOVE Command
0120-0207	P2252	COBOL	31.0.0097	Statistics Not Correct for STO
0120-0207	P2252	COBOL74	31.0.0149	Statistics Not Correct for STO
0120-0215	P2062	COBOL	31.0.0084	SELECTION Expressions
0121-0187	P2237	ALGOL	31.0.0148	Syntaxing of Format Specificat
0121-0187	P2237	ALGOL	31.0.0152	Syntaxing of Format Specificat
0121-0187	P2237	ALGOLTABLE	31.0.0023	Syntaxing of Format Specificat
0122-5013	P1850	FORTRAN	31.0.0020	Multiple Entry Points with OPT
0122-5026	P1898	MCP	31.0.0489	Compile-and-Go with SUBSPACES=
0122-5037	P1715	MCP	31.0.0356	Dying Stack in WSSHERIFF
0122-5040	P1516	BINDER	31.0.0012	Binding Entry Points
0122-5043	P1709	JOBFORMAT	31.0.0019	Includes LSN for Job Summary
0122-5046	P1765	CONTROLLER	31.0.0046	SQ Command Correction
0122-5048	P1734	MCP	31.0.0416	FM Vs. Saved LP
0122-5051	P1516	BINDER	31.0.0012	Binding Entry Points
0125-0222	P2464	WFL	31.0.0073	WFL "BEGIN; JOB" Correction
0125-0223	P1733	MCP	31.0.0413	READALABEL Vs. EXTMODE
0125-0228	P2358	DUMPANALY	31.0.0067	OVERLAYCF
0125-0228	P2358	MCP	31.0.0783	OVERLAYCF
0125-0241	P1794	RJE	31.0.0027	WFLCOMPILER Not Going to EOT
0128-0358	P9162	MCP	31.0.0009	NO MEM in Swapspace
0128-0389	P9024	IN-OUTPUT	31.0.0017	FIBLOCK and TIMESTAMP
0128-0389	P9024	IN-OUTPUT	31.0.0132	FIBLOCK and TIMESTAMP
0128-0390	P1217	MCP	31.0.0221	Family Substitution Vs. Catalo
0128-0397	P1129	MCP	31.0.0189	File Removed Message
0128-0399	P9150	CONTROLLER	31.0.0008	FILEKIND
0128-0399	P9150	MCP	31.0.0067	FILEKIND
0128-0410	P9153	MCP	31.0.0071	Change Title of Cataloged File
0128-0421	D2765	ALGOL	31.0.	Large Numeric Literals
0128-0436	P1086	MCP	31.0.0174	Unlabeled Tape Corrections
0128-0441	P1192	MCP	31.0.0212	SWAPPER Vs. Crunch
0128-0479	P1721	MCP	31.0.0371	LINKLISTINSERT Vs. XTEND
0128-0487	D2822	FORTRAN	31.0.0034	Binding and Statistics
0128-0492	P1991	ALGOL	31.0.0104	Recovering from Syntax Error
0128-0493	P2023	MCP	31.0.0499	SEG0 of Flatrows
0128-0497	P1715	MCP	31.0.0356	Dying Stack in WSSHERIFF
0128-0501	P1792	ALGOL	31.0.0040	No Error on "TITLE=<ptr>"
0128-0504	P1712	MCP	31.0.0352	VOLUME DELETE
0128-0505	P1709	JOBFORMAT	31.0.0019	Includes LSN for Job Summary
0128-0506	P1939	FORTRAN	31.0.0033	Run Time Problems with Very La
0128-0522	P2322	FORTRAN	31.0.0041	FORTRAN XREF
0128-0523	P2022	MCP	31.0.0492	Rebuild Vs. AD
0128-0523	P2022	MCP	31.0.0637	Rebuild Vs. AD
0128-0524	P1781	IN-OUTPUT	31.0.0376	CANCEL/DIRECTDCREAD

FTR ACTION TABLE

FTR	NOTE	SOFTWARE	PATCH	DESCRIPTION
0128-0525	P2542	ESPOLINTRN	31.0.0063	Formatting Intrinsics
0128-0528	P1794	RJE	31.0.0027	WFLCOMPILER Not Going to EOT
0128-0529	P2391	MCP	31.0.0797	Catalog Remove
0128-0538	P2534	MCP	31.0.0913	CHECKPOINT Vs. JEDGARHOOVER
0130-0682	P9158	MCP	31.0.0077	WFL USER Statement in Subrout
0130-0710	P1111	MCP	31.0.0163	AUTOPRINT Vs. Multiple SKIP TO
0130-0778	P2113	ALGOL	31.0.0134	INVALID OP in Freefield WRITE
0132-0105	P2381	COBOL	31.0.0139	Erroneous Syntax Error
0134-0156	P1184	MCP	31.0.0207	EOT Vs. TAPEPARITYRETRY
0134-0158	P2414	RECOVERY	31.0.0048	REBUILD and RECONSTRUCT Back U
0139-0107	D2761	ALGOL	31.0.	MAKEHOST Environments
0141-0271	D2959	COBOL	31.0.	File Attribute IORECORDNUM
0141-0273	P1664	COBOL	31.0.0038	Graceful Exit from Syntaxed ST
0141-0273	P1664	COBOL74	31.0.0099	Graceful Exit from Syntaxed ST
0141-0275	P2062	COBOL	31.0.0084	SELECTION Expressions
0141-0278	P1128	IN-OUTPUT	31.0.0182	CHECKCOUNT Vs. 5500 Tapes
0141-0292	P2274	MCP	31.0.0692	CLOSE Errors Fatal
0141-0292	P2274	MCP	31.0.0825	CLOSE Errors Fatal
0141-0292	P2274	MCP	31.0.0976	CLOSE Errors Fatal
0141-0293	P2274	MCP	31.0.0692	CLOSE Errors Fatal
0141-0293	P2274	MCP	31.0.0825	CLOSE Errors Fatal
0141-0293	P2274	MCP	31.0.0976	CLOSE Errors Fatal
0141-0298	P1709	JOBFORMAT	31.0.0019	Includes LSN for Job Summary
0141-0299	P9176	MCP	31.0.0090	SWAPPER QT's BACKUP
0141-0307	D2816	COBOL	31.0.0049	NOTE Verb Extensions
0143-0232	P1082	MCP	31.0.0162	LEIBNITZ Vs. Overlay
0143-0236	P9220	MCP	31.0.0113	PLANT MESSAGE vs AUTOBACKUP
0143-0250	P1856	IN-OUTPUT	31.0.0449	Log Vs. Title
0143-0250	P1856	IN-OUTPUT	31.0.0504	Log Vs. Title
0143-0251	P1767	DUMPALL	31.0.0003	ROUTINE and COPY Statements Re
0143-0252	P1459	MCP	31.0.0308	MISSINGPROCEDURE Call
0143-0256	P2311	MCP	31.0.0736	EOJ Vs. USTKASSIGNED
0143-0260	P2515	DUMPANALY	31.0.0086	DCC and DCP Station Info
0144-0083	P2005	LCOBOL	31.0.0007	TU Firmware Buffer Lengths
0148-0049	P1767	DUMPALL	31.0.0003	ROUTINE and COPY Statements Re
0148-0117	P2245	LCOBOL	31.0.0009	IF Statement with OPTIM Correc
0148-0118	P2246	LCOBOL	31.0.0010	DISPLAY Statement Correction
0148-0119	P2463	LCOBOL	31.0.0014	Incorrect Code, Invalid Syntax
0148-0120	P2463	LCOBOL	31.0.0014	Incorrect Code, Invalid Syntax
0149-0379	P1177	SCR	31.0.0017	Buffer Fill Correction
0149-0422	P1030	IN-OUTPUT	31.0.0152	Change INTNAME
0149-0435	P9272	DUMPANALY	31.0.0013	FIBLOCKSNR
0149-0435	P9272	IN-OUTPUT	31.0.0136	FIBLOCKSNR
0149-0452	P9227	MCP	31.0.0131	HDREXPAND
0149-0458	P9224	MCP	31.0.0097	VERIFYFAMILY vs HDRVECTORLOCK
0149-0461	P9225	MCP	31.0.0098	DESCRIPTOR error on Packs
0149-0463	P9229	MCP	31.0.0129	ANABOLISM vs MAKEJOBFILE
0149-0463	P9229	SCR	31.0.0006	ANABOLISM vs MAKEJOBFILE
0149-0464	P1172	SCR	31.0.0012	Deadly Embrace
0149-0473	P1087	MCP	31.0.0175	MCSREADY on SM Input
0149-0486	P1729	MCP	31.0.0386	SIRWOFADDRESS
0149-0489	P2003	BACKUP	31.0.0010	Backward Skip
0149-0493	P1762	BACKUP	31.0.0008	All Copies Not Printed
0151-0361	P1054	MCP	31.0.0156	IC Diskpacks
0151-0372	P2397	DATACOM	31.0.0821	Restarting DSed DCP
0151-0387	P1726	MCP	31.0.0381	DUMP Vs. RD
0154-0167	P2381	COBOL	31.0.0139	Erroneous Syntax Error
0154-0173	P2085	NDL	31.0.0001	FILE <family> Statement
0154-0201	P2085	NDL	31.0.0001	FILE <family> Statement
0154-0203	P2319	COBOL	31.0.0123	Condition Names
0154-0210	P2381	COBOL	31.0.0139	Erroneous Syntax Error
0154-0217	P2381	COBOL	31.0.0139	Erroneous Syntax Error
0156-0066	P9143	MCP	31.0.0057	Fatal Stack Overflow
0156-0073	P1125	MCP	31.0.0178	Unit Left Assigned
0156-0075	P1609	MCP	31.0.0349	Guardfile
0156-0080	P1767	DUMPALL	31.0.0003	ROUTINE and COPY Statements Re
0156-0083	P1715	MCP	31.0.0356	Dying Stack in WSSHERIFF
0156-0092	P1715	MCP	31.0.0356	Dying Stack in WSSHERIFF
0156-0093	P1715	MCP	31.0.0356	Dying Stack in WSSHERIFF
0156-0100	P1609	MCP	31.0.0349	Guardfile
0156-0102	D2852	COBOL	31.0.	Separators
0156-0110	P2580	BACKUP	31.0.0026	BACKUP Vs. System Without DISK
0156-0112	P2358	DUMPANALY	31.0.0067	OVERLAYCF
0156-0112	P2358	MCP	31.0.0783	OVERLAYCF
0156-0117	P2537	MCP	31.0.0907	Reel Switch Vs. Parity Error
0161-0114	D2958	COBOL	31.0.	Setting of System Compatibilit
0163-0078	P2387	ACR	31.0.0118	Erroneous Not Found Result
0166-0211	P9274	IN-OUTPUT	31.0.0142	CLOSE HERE on Empty Tape File
0166-0211	P9274	IN-OUTPUT	31.0.0258	CLOSE HERE on Empty Tape File
0166-0214	P9274	IN-OUTPUT	31.0.0142	CLOSE HERE on Empty Tape File
0166-0214	P9274	IN-OUTPUT	31.0.0258	CLOSE HERE on Empty Tape File
0166-0219	P9222	MCP	31.0.0118	CP vs SWAPPER

FTR ACTION TABLE

FTR	NOTE	SOFTWARE	PATCH	DESCRIPTION
0166-0225	P9219	MCP	31.0.0116	DBS vs CHECKPOINT
0166-0229	P9220	MCP	31.0.0113	PLANT MESSAGE vs AUTOBACKUP
0166-0268	P1767	DUMPALL	31.0.0003	ROUTINE and COPY Statements Re
0166-0269	P1664	COBOL	31.0.0038	Graceful Exit from Syntaxed ST
0166-0269	P1664	COBOL74	31.0.0099	Graceful Exit from Syntaxed ST
0166-0271	D2249	MCP	31.0.0059	OF of GETSTATUS Waiting on <fi
0166-0289	P2025	PRINTBIND	31.0.0005	Improved Handling of Unknown I
0166-0293	D2576	MCP	31.0.0351	RERUN Vs. EOF
0166-0293	D2576	MCP	31.0.0401	RERUN Vs. EOF
0166-0296	P2258	COBOL	31.0.0105	INVALID INDEX in Bad REPORT Wr
0166-0302	P1662	COBOL	31.0.0035	Exceptioncode for ANSI74 Corre
0166-0306	D2576	MCP	31.0.0351	RERUN Vs. EOF
0166-0306	D2576	MCP	31.0.0401	RERUN Vs. EOF
0166-0308	P1720	MCP	31.0.0366	Checkpoint
0166-0311	P1792	ALGOL	31.0.0040	No Error on "TITLE=<ptr>"
0166-0315	P2296	FILEDATA	31.0.0008	Allow CANDE LFILES with No Di
0166-0318	P2580	BACKUP	31.0.0026	BACKUP Vs. System Without DISK
0166-0321	P1709	JOBFORMAT	31.0.0019	Includes LSN for Job Summary
0166-0324	P2576	BACKUP	31.0.0024	First Level Filename Vs. Userc
0166-0324	P2576	BACKUP	31.0.0025	First Level Filename Vs. Userc
0166-0327	P1766	DUMPALL	31.0.0002	UL OPtion Correction
0166-0328	P2382	JOBFORMAT	31.0.0036	CLOSE Types
0166-0332	P2532	MCP	31.0.0924	CHECKPOINTFILE Vs. Abort
0168-0262	P1941	PLI	31.0.0016	Deallocate Dynamic Bounds Auto
0168-0323	P1786	MCP	31.0.0430	Guardfile Vs. Stackoverflow
0168-0336	P2322	FORTRAN	31.0.0041	FORTRAN XREF
0168-0343	P2591	ESPOLINTRN	31.0.0067	FORTRAN BCL Formatting
0168-0346	P1781	IN-OUTPUT	31.0.0376	CANCEL/DIRECTDCREAD
0168-0357	P1516	BINDER	31.0.0012	Binding Entry Points
0168-0358	P1933	ALGOL	31.0.0119	SERIALNO File Attribute
0168-0372	P1609	MCP	31.0.0349	Guardfile
0168-0374	P1768	DUMPALL	31.0.0004	DMPMT Short Records
0168-0384	P2471	WFL	31.0.0074	COMPILATION ABORTED Not Trunca
0168-0387	P2312	MCP	31.0.0738	SWAPPER Gets DIV BY ZERO
0168-0389	P2125	WFL	31.0.0044	Fault on Bad Task Attribute
0168-0397	P1819	ALGOL	31.0.0043	Update Pointer in REPLACE BY S
0169-0131	P1449	ALGOL	31.0.0145	Small Pools for Formats
0169-0148	P9024	IN-OUTPUT	31.0.0017	FIBLOCK and TIMESTAMP
0169-0148	P9024	IN-OUTPUT	31.0.0132	FIBLOCK and TIMESTAMP
0169-0158	P9230	MCP	31.0.0128	HISTORY vs PROGRAMDUMP
0169-0161	P1016	SORT	31.0.0002	KANGAROO
0169-0181	P1785	MCP	31.0.0428	Software Interrupt Execution
0169-0183	P1516	BINDER	31.0.0012	Binding Entry Points
0169-0185	P9272	DUMPANALY	31.0.0013	FIBLOCKSNR
0169-0185	P9272	IN-OUTPUT	31.0.0136	FIBLOCKSNR
0169-0186	P1715	MCP	31.0.0356	Dying Stack in WSSHERIFF
0169-0188	P2257	COBOL	31.0.0104	Too Many REPORT Writer Code Cl
0169-0189	P1786	MCP	31.0.0430	Guardfile Vs. Stackoverflow
0170-0075	P9193	XREFANALY	31.0.0003	Memory Requirements
0170-0100	P2338	COBOL	31.0.0127	Large Report Writer Exceeded C
0171-0004	P2252	COBOL	31.0.0097	Statistics Not Correct for STO
0171-0004	P2252	COBOL74	31.0.0149	Statistics Not Correct for STO
0174-0127	P2237	ALGOL	31.0.0148	Syntaxing of Format Specificat
0174-0127	P2237	ALGOL	31.0.0152	Syntaxing of Format Specificat
0174-0127	P2237	ALGOLTABLE	31.0.0023	Syntaxing of Format Specificat
0174-0141	P2218	ALGOL	31.0.	Warning Message Given for Form
0174-0141	P2218	ALGOLTABLE	31.0.0022	Warning Message Given for Form
0174-0153	P1254	MCP	31.0.0237	DISKLIMIT Vs. RSVP
0174-0157	D2776	ALGOL	31.0.	MOD Operator When Second Param
0174-0162	P2253	COBOL	31.0.0048	SELECT Statement Syntax Errors
0174-0165	P2205	RJE	31.0.0041	NOLOGON Vs. Usercode "."
0174-0172	P2349	ACR	31.0.0109	ORDERED Data Set INVALID OP
0174-0175	P1767	DUMPALL	31.0.0003	ROUTINE and COPY Statements Re
0174-0176	P2347	ACR	31.0.0106	Ordered Available Table Corrup
0174-0180	P2237	ALGOL	31.0.0148	Syntaxing of Format Specificat
0174-0180	P2237	ALGOL	31.0.0152	Syntaxing of Format Specificat
0174-0180	P2237	ALGOLTABLE	31.0.0023	Syntaxing of Format Specificat
0174-0186	P2529	DUMPALL	31.0.0014	Error Messages
0175-0025	P1516	BINDER	31.0.0012	Binding Entry Points
0175-0032	P2125	WFL	31.0.0044	Fault on Bad Task Attribute
0175-0033	P2259	COBOL	31.0.0110	ANSI74 LINAGE WRITE AT END OF
0176-0174	P1767	DUMPALL	31.0.0003	ROUTINE and COPY Statements Re
0176-0176	P2051	COBOL	31.0.0069	VALUE Clause for COMP-4 Items
0178-3001	P1819	ALGOL	31.0.0043	Update Pointer in REPLACE BY S
0180-0459	D2762	ALGOL	31.0.	VECTORMODE Loops with Length <
0180-0536	P2125	WFL	31.0.0044	Fault on Bad Task Attribute
0180-0538	P1516	BINDER	31.0.0012	Binding Entry Points
0180-0543	P1516	BINDER	31.0.0012	Binding Entry Points
0180-0550	P2125	WFL	31.0.0044	Fault on Bad Task Attribute
0180-0554	P1942	RJE	31.0.0037	Restartfile Name SEG ARRAY Err
0181-0086	P9159	MCP	31.0.0078	Information Carry Over
0183-2904	P1928	UTIL	31.0.0025	Tape Block Count Not Properly

FTR ACTION TABLE

FTR	NOTE	SOFTWARE	PATCH	DESCRIPTION
0184-0044	P1734	MCP	31.0.0416	FM Vs. Saved LP
0184-0046	P2210	REORG	31.0.0024	Index Random, Random Shadow Fi
0184-0047	P2136	REORG	31.0.0017	Unused Fold Words
0184-0053	P1816	RJE	31.0.0033	QT Printer Files Saved Over Ha
0184-0058	P2473	ACR	31.0.0129	Incorrect Statistics Totals
0184-0058	P2473	ACR	31.0.0132	Incorrect Statistics Totals
0184-0063	P1884	UTIL	31.0.0051	TDATABEGIN Field Not Initializ
0184-0067	P2579	ALGOL	31.0.0181	String in DEFINE Too Long
0184-5001	D2591	BUILDREORG	31.0.0015	Checking BUILDREORG Status in
0185-0236	D2932	WFL	31.0.0075	STATUS Attribute
0185-0239	D2463	CONTROLLER	31.0.0021	RESOURCECHECK
0185-0239	D2463	MCP	31.0.0222	RESOURCECHECK
0185-0260	P1768	DUMPALL	31.0.0004	DMPMT Short Records
0185-0263	P1732	MCP	31.0.0390	ELAPSEDTIMELIMIT
0185-0264	P1781	IN-OUTPUT	31.0.0376	CANCEL/DIRECTDCREAD
0185-0269	P2251	INQ	31.0.0039	Display of Numeric Items
0185-0270	P2373	COBOL	31.0.0135	Condition Names
0187-0172	P2108	DATACOM	31.0.0528	DCCONTROL Invalid Index
0187-0172	P2108	DCPPROGEN	31.0.0003	DCCONTROL Invalid Index
0187-0186	P1078	LOADER	31.0.0005	Three-Multiplexor System
0187-0188	P2397	DATACOM	31.0.0821	Restarting DSed DCP
0187-0191	P2108	DATACOM	31.0.0528	DCCONTROL Invalid Index
0187-0191	P2108	DCPPROGEN	31.0.0003	DCCONTROL Invalid Index
0187-0194	P2002	WFL	31.0.0023	Zip with Array
0187-0197	P1863	MCP	31.0.0456	DISCSTATUS
0187-0197	P1863	MCP	31.0.0534	DISCSTATUS
0187-0197	P1863	MCP	31.0.0549	DISCSTATUS
0188-0110	P1879	ALGOL	31.0.0049	Syntax Errors Not Being Report
0188-0141	P9169	MCP	31.0.0085	Processor Priority
0188-0143	P9228	IN-OUTPUT	31.0.0130	AVAILABLE
0188-0146	P9169	MCP	31.0.0085	Processor Priority
0188-0150	P1191	MCP	31.0.0211	FINDINPUT Vs. UNBRF
0188-0151	P1253	MCP	31.0.0235	Zip with Large Array
0188-0163	P1516	BINDER	31.0.0012	Binding Entry Points
0188-0168	P1516	BINDER	31.0.0012	Binding Entry Points
0188-0169	P2001	WFL	31.0.0022	Zip With Array
0188-0170	P1880	ALGOL	31.0.0050	AUTOBIND and One Level Codefil
0188-0171	P2002	WFL	31.0.0023	Zip with Array
0188-0172	D2649	ALGOL	31.0.0048	New Information in the Listing
0188-0174	P1878	ALGOL	31.0.0047	IF FALSE in Procedures to be B
0188-0177	P1459	MCP	31.0.0308	MISSINGPROCEDURE Call
0188-0183	P2383	JOBFORMAT	31.0.0037	JOB ENTERED SYSTEM Time
0188-0186	P2492	BINDER	31.0.0048	Rebinding with FORTRAN DATA S
0188-0187	P2465	BINDER	31.0.0047	Entry Binding Corrections
0189-0176	D2865	MCP	31.0.	Verify 235 Packs in IV Message
0191-0120	P2340	COBOL	31.0.0129	Negative Zeros in VALUE Clause
0191-0125	P2545	MCP	31.0.0931	GETAREA Vs. Locks
0192-0299	P1794	RJE	31.0.0027	WFLCOMPILER Not Going to EOT
0192-0300	P2552	MCP	31.0.0978	Card Reader Secured Status
0193-0519	P1863	MCP	31.0.0456	DISCSTATUS
0193-0519	P1863	MCP	31.0.0534	DISCSTATUS
0193-0519	P1863	MCP	31.0.0549	DISCSTATUS
0193-0525	P2188	FILECOPY	31.0.0008	Punched Output
0194-0222	D2576	MCP	31.0.0351	RERUN Vs. EOF
0194-0222	D2576	MCP	31.0.0401	RERUN Vs. EOF
0194-0234	P2530	MCP	31.0.0933	CHECKPOINT Vs. OLAYHEADER
0194-0239	D2960	COBOL	31.0.	Comparison of Nonnumeric Opera
0200-0203	P1182	MCP	31.0.0203	DBS D3 NOMEM
0200-0264	P1715	MCP	31.0.0356	Dying Stack in WSSHERIFF
0201-0220	P1936	FORTRAN	31.0.0030	FREE Format Vs. INCLUDE
0201-0225	P9146	MCP	31.0.0062	Reserve Locks
0201-0225	P9146	MCP	31.0.0073	Reserve Locks
0201-0225	P9146	MCP	31.0.0125	Reserve Locks
0201-0225	P9146	MCP	31.0.0135	Reserve Locks
0201-0226	P1763	COBOL	31.0.0041	User Intrinsic Calls Not Scann
0201-0230	P9218	MCP	31.0.0111	BD Files vs EOF=0
0201-0242	D2762	ALGOL	31.0.	VECTORMODE Loops with Length <
0201-0256	P2003	BACKUP	31.0.0010	Backward Skip
0201-0266	P1794	RJE	31.0.0027	WFLCOMPILER Not Going to EOT
0202-0268	P1784	MCP	31.0.0425	CRUNCH Vs. CP
0202-0291	P1788	MCP	31.0.0433	LOOKFORIT Control State
0202-0306	P1893	DUMPANALY	31.0.0043	SIRW Analysis
0202-0315	P2396	MCP	31.0.0794	CPRESTART EOJ
0203-0154	P1453	ESPOLINTRN	31.0.0060	STATISTICS Option
0203-0156	P2568	BACKUP	31.0.0019	Loops When "<skip count>" Omit
0203-0160	P1879	ALGOL	31.0.0049	Syntax Errors Not Being Report
0203-0164	P2569	BACKUP	31.0.0020	REEL Attribute Not Set Before
0203-0165	P2509	WFL	31.0.0078	Eliminate Recursive Handling o
0203-0167	P2399	CONTROLLER	31.0.0094	Message Greater Than 1896 Char
0203-0171	P2510	WFL	31.0.0079	COMPILE Statement Without Comp
0203-0175	P1933	ALGOL	31.0.0119	SERIALNO File Attribute
0203-0183	P1915	ACR	31.0.0068	COPYAUDIT Not Zipped if Error

FTR ACTION TABLE

FTR	NOTE	SOFTWARE	PATCH	DESCRIPTION
0205-0616	P2492	BINDER	31.0.0048	Rebinding with FORTRAN DATA S
0205-0617	P2290	ALGOL	31.0.0164	No Spurious UNKNOWN DOLLAR CAR
0205-0617	P2290	ALGOL	31.0.0168	No Spurious UNKNOWN DOLLAR CAR
0205-0617	P2290	ALGOLTABLE	31.0.0026	No Spurious UNKNOWN DOLLAR CAR
0207-0125	D2674	CANDE	31.0.0035	Data Files Update
0207-0127	P9156	MCP	31.0.0074	Task Attribute Errors
0208-7708	P1112	MCP	31.0.0165	AUTOPRINT Vs. FM Reply
0208-7801	P1787	MCP	31.0.0431	TAPEPARITYRETRY Vs. BLOCKEXIT
0208-7805	P2077	WFL	31.0.0026	Extra BEGIN in WFL Job Deck
0208-7808	P1734	MCP	31.0.0416	FM Vs. Saved LP
0208-7817	P1820	ACR	31.0.0054	Faulty Audit Constraint
0208-7823	P1781	IN-OUTPUT	31.0.0376	CANCEL/DIRECTDCREAD
0209-0029	P9223	MCP	31.0.0099	SUBSPACES vs USERCODE
0209-0048	P2562	RJE	31.0.0052	DESTNAME Directed REMCP Files
0209-0049	P2461	UTIL	31.0.0062	COPY AS Error
0209-0049	P2461	UTIL	31.0.0063	COPY AS Error
0209-0049	P2461	UTIL	31.0.0067	COPY AS Error
0210-0115	P1112	MCP	31.0.0165	AUTOPRINT Vs. FM Reply
0210-0117	P1112	MCP	31.0.0165	AUTOPRINT Vs. FM Reply
0210-0124	P2007	WFL	31.0.0021	REMOVE Command
0210-0126	P2062	COBOL	31.0.0084	SELECTION Expressions
0210-0127	P2202	BDMSCOBOL	31.0.0095	Bad DMS Invoke Listing
0210-0137	P2086	RJE	31.0.0038	MAXTERMINALS Greater Than 48
0210-0142	P2202	BDMSCOBOL	31.0.0095	Bad DMS Invoke Listing
0211-7743	P9157	DATACOM	31.0.0075	Inserting Null Message in Inac
0211-7747	P9153	MCP	31.0.0071	Change Title of Cataloged File
0213-0075	P2264	LOGGER	31.0.0020	Elapsed Time of Jobs Run Throu
0214-0030	P1516	BINDER	31.0.0012	Binding Entry Points
0215-0149	D2623	ALGOL	31.0.0030	Equation Disallowed in Global
0215-0149	D2623	ALGOL	31.0.0031	Equation Disallowed in Global
0215-0149	D2623	ALGOL	31.0.0036	Equation Disallowed in Global
0215-0149	D2623	ALGOLTABLE	31.0.0005	Equation Disallowed in Global
0215-0157	P1516	BINDER	31.0.0012	Binding Entry Points
0215-0159	P1792	ALGOL	31.0.0040	No Error on "TITLE=<ptr>"
0215-0160	P2077	WFL	31.0.0026	Extra BEGIN in WFL Job Deck
0216-0068	P2392	MCP	31.0.0814	Swap Out Waiting for Schedule
0216-0075	P2591	ESPOLINTRN	31.0.0067	FORTRAN BCL Formatting
0216-0076	P1819	ALGOL	31.0.0043	Update Pointer in REPLACE BY S
0222-0366	P1941	PLI	31.0.0016	Deallocate Dynamic Bounds Auto
0222-0368	P2591	ESPOLINTRN	31.0.0067	FORTRAN BCL Formatting
0222-0373	P1516	BINDER	31.0.0012	Binding Entry Points
0222-0380	P1516	BINDER	31.0.0012	Binding Entry Points
0222-0383	P2125	WFL	31.0.0044	Fault on Bad Task Attribute
0222-0384	P1373	ALGOL	31.0.0142	Storing Singles into Doubles
0222-0388	P2025	PRINTBIND	31.0.0005	Improved Handling of Unknown I
0225-0021	P2024	MCP	31.0.0506	GETSTATUS Vs. UINFO
0226-0352	P9275	IN-OUTPUT	31.0.0143	EVEN Parity Vs. EOF1
0226-0354	P9269	MCP	31.0.0160	Density Vs. Printlabel
0226-0355	P9143	MCP	31.0.0057	Fatal Stack Overflow
0226-0363	D2765	ALGOL	31.0.	Large Numeric Literals
0226-0370	P1715	MCP	31.0.0356	Dying Stack in WSSHERIFF
0226-0383	P1800	CONTROLLER	31.0.0049	WAITLIMIT
0226-0383	P1800	MCP	31.0.0443	WAITLIMIT
0226-0388	P1925	RECOVERY	31.0.0025	ABORT Gets WAITING ON ROWLOCKO
0226-0395	P1863	MCP	31.0.0456	DISCSTATUS
0226-0395	P1863	MCP	31.0.0534	DISCSTATUS
0226-0395	P1863	MCP	31.0.0549	DISCSTATUS
0226-0399	P1827	REORG	31.0.0005	DATAERROR 001 when Reorganizin
0226-0410	P2642	UTIL	31.0.0072	Failure to Dump Partitions
0226-0418	P2136	REORG	31.0.0017	Unused Fold Words
0226-0422	P1903	REORG	31.0.0012	Generation of Manual Subsets
0227-0115	P2006	LOADER	31.0.0025	Unit Display
0227-0116	P2274	MCP	31.0.0692	CLOSE Errors Fatal
0227-0116	P2274	MCP	31.0.0825	CLOSE Errors Fatal
0227-0116	P2274	MCP	31.0.0976	CLOSE Errors Fatal
0227-0117	P2274	MCP	31.0.0692	CLOSE Errors Fatal
0227-0117	P2274	MCP	31.0.0825	CLOSE Errors Fatal
0227-0117	P2274	MCP	31.0.0976	CLOSE Errors Fatal
0227-0123	P2276	INQ	31.0.0040	Display Limit for OPTION PRINT
0231-0381	P9143	MCP	31.0.0057	Fatal Stack Overflow
0231-0386	P2331	BDMSCOBOL	31.0.0120	Declaration of Item With Same
0231-0403	P2515	DUMPANALY	31.0.0086	DCC and DCP Station Info
0231-0404	P1862	MCP	31.0.0454	RSNINVALID
0231-0409	P1715	MCP	31.0.0356	Dying Stack in WSSHERIFF
0231-0410	P1766	DUMPALL	31.0.0002	UL OPtion Correction
0231-0411	P2002	WFL	31.0.0023	Zip with Array
0231-0428	P2295	COBOL	31.0.0109	Exception Code Erroneously Che
0231-0433	P2260	COBOL	31.0.0111	Random File Getting Serial Wri
0231-0439	D2687	PRINTBIND	31.0.0009	Improve Handling of DATABASE I
0232-0026	P9143	MCP	31.0.0057	Fatal Stack Overflow
0238-0167	P2316	COBOL	31.0.0118	Maximum Files Allowed
0238-0168	P1734	MCP	31.0.0416	FM Vs. Saved LP

FTR	NOTE	SOFTWARE	PATCH	DESCRIPTION
0239-0807	P9271	MCP	31.0.0137	CP Vs. Backup Tape
0242-0004	P1765	CONTROLLER	31.0.0046	SQ Command Correction
0246-0048	P2274	MCP	31.0.0692	CLOSE Errors Fatal
0246-0048	P2274	MCP	31.0.0825	CLOSE Errors Fatal
0246-0048	P2274	MCP	31.0.0976	CLOSE Errors Fatal
0248-0118	P1768	DUMPALL	31.0.0004	DMPMT Short Records
0249-0138	P1888	DCALGOLINT	31.0.0004	Correct DCP Sleep Address Inte
0249-0162	P2198	MCP	31.0.0575	GETAROW Vs. catalog
0250-0030	P1716	MCP	31.0.0357	PB 2 Level Name
0250-0031	P2022	MCP	31.0.0492	Rebuild Vs. AD
0250-0031	P2022	MCP	31.0.0637	Rebuild Vs. AD
0250-0035	P2580	BACKUP	31.0.0026	BACKUP Vs. System Without DISK
0250-0037	P2330	BDMSCOBOL	31.0.0116	Listing of the Variable Format
0250-0039	P1871	UTIL	31.0.0024	Missing Entry in HL File
0250-0040	P2642	UTIL	31.0.0072	Failure to Dump Partitions
0250-0042	P2604	ACR	31.0.0145	NOT FOUND for Partitioned Orde
0250-0043	P2546	MCP	31.0.0850	DD Vs. CM
0250-0043	P2546	MCP	31.0.0860	DD Vs. CM
0253-0058	P9270	MCP	31.0.0138	DBS PROGRAMDUMP
0254-0018	P1767	DUMPALL	31.0.0003	ROUTINE and COPY Statements Re
0254-0022	P1715	MCP	31.0.0356	Dying Stack in WSSHERIFF
0254-0023	P2125	WFL	31.0.0044	Fault on Bad Task Attribute
0254-0025	P2580	BACKUP	31.0.0026	BACKUP Vs. System Without DISK
0254-0027	P1794	RJE	31.0.0027	WFLCOMPILER Not Going to EOT
0255-1004	P1219	MCP	31.0.0226	REMOVE Vs. Security
0255-2014	P1516	BINDER	31.0.0012	Binding Entry Points
0255-2024	P1936	FORTRAN	31.0.0030	FREE Format Vs. INCLUDE
0258-0032	P2353	BUILDINQ	31.0.0021	Invalid TABLE EXCEEDED Error
0258-0039	D2568	DASDL	31.0.0013	Allow More Structures, Items P
0258-0039	D2568	PROPERTIES	31.0.0002	Allow More Structures, Items P
0260-0064	P2248	ALGOL	31.0.0153	Exclamation Mark and Underscor
0260-0114	P2077	WFL	31.0.0026	Extra BEGIN in WFL Job Deck
0260-0115	D2822	FORTRAN	31.0.0034	Binding and Statistics
0260-0118	D2874	COBOL	31.0.0134	OCCURS DEPENDING Option
0260-0118	D2874	COBOL74	31.0.0183	OCCURS DEPENDING Option
0261-0134	P1845	FORTRAN	31.0.0015	Monitor Binder Interface
0261-0158	P9217	CONTROLLER	31.0.0009	SQ NO ENTRIES
0261-0160	P9173	MCP	31.0.0091	LEIBNITZ EOTINHDR
0261-0171	P9145	MCP	31.0.0061	CRUNCH Vs. COPY
0261-0200	P1216	MCP	31.0.0220	OF Vs. Exclusive
0261-0210	D2822	FORTRAN	31.0.0034	Binding and Statistics
0261-0213	P1886	CARDLINE	31.0.0002	Job Cards Transferred to Disk
0261-0214	P2001	WFL	31.0.0022	Zip With Array
0261-0239	P1727	MCP	31.0.0383	AUTOBACKUP Priority
0261-0243	P1765	CONTROLLER	31.0.0046	SQ Command Correction
0261-0244	P1813	LOGGER	31.0.0014	Items Extending Past Column 72
0261-0253	P2275	MCP	31.0.0701	"COPY =" Many Files
0261-0262	P2077	WFL	31.0.0026	Extra BEGIN in WFL Job Deck
0261-0268	P2322	FORTRAN	31.0.0041	FORTRAN XREF
0261-0269	P2322	FORTRAN	31.0.0041	FORTRAN XREF
0264-0080	P1710	SORT	31.0.0011	Security Violation on SORT/STA
0264-0098	P2596	DATACOM	31.0.1011	Not Ready Lines After Cluster
0264-0101	P2477	BUILDREORG	31.0.0013	INVALID INDEX at 30139840
0264-0102	P2556	LOGGER	31.0.0024	ALINK Vs. MATCHM (BOJ,EOJ)
0271-0344	P2464	WFL	31.0.0073	WFL "BEGIN; JOB" Correction
0271-9004	P2236	ALGOL	31.0.0147	IF FALSE and $LINEINFO
0278-0031	D2777	ALGOL	31.0.	Missing Actual Parameter in De
0285-0032	P1801	MCP	31.0.0444	FIBLOCK Vs. PROGRAMDUMP
0285-0033	P2202	BDMSCOBOL	31.0.0095	Bad DMS Invoke Listing
0286-0035	D2763	ALGOL	31.0.	Clarification of REPLACE State
0286-0047	P1734	MCP	31.0.0416	FM Vs. Saved LP
0286-0048	P2274	MCP	31.0.0692	CLOSE Errors Fatal
0286-0048	P2274	MCP	31.0.0825	CLOSE Errors Fatal
0286-0048	P2274	MCP	31.0.0976	CLOSE Errors Fatal
0286-0053	P1862	MCP	31.0.0454	RSNINVALID
0286-0068	P2301	MCP	31.0.0712	AB Buffer Size
0286-0068	P2301	MCP	31.0.0750	AB Buffer Size
0286-0077	P2127	ACR	31.0.0081	Clear Read/Write Ahead Statist
0286-0078	P2414	RECOVERY	31.0.0048	REBUILD and RECONSTRUCT Back U
0292-0105	P2397	DATACOM	31.0.0821	Restarting DSed DCP
0292-0119	P1077	CONTROLLER	31.0.0017	PC Vs. MPX3
0292-0127	P1767	DUMPALL	31.0.0003	ROUTINE and COPY Statements Re
0292-0132	P1767	DUMPALL	31.0.0003	ROUTINE and COPY Statements Re
0292-0133	P1766	DUMPALL	31.0.0002	UL OPtion Correction
0292-0141	P1661	COBOL74	31.0.0097	Task to Task String Attribute
0292-0143	P1706	CONTROLLER	31.0.0037	NEXT on RJE Peripheral Status
0292-0153	P2464	WFL	31.0.0073	WFL "BEGIN; JOB" Correction
0292-0159	P2105	CONTROLLER	31.0.0061	DUP FAMILY
0292-0159	P2105	MCP	31.0.0511	DUP FAMILY
0292-0163	P2317	COBOL	31.0.0119	Info Table Overflow
0292-0164	D2832	PROPERTIES	31.0.0023	DUPLICATES Subcategory 2 Expla
0292-0165	P1732	MCP	31.0.0390	ELAPSEDTIMELIMIT

FTR ACTION TABLE

FTR	NOTE	SOFTWARE	PATCH	DESCRIPTION
0292-0180	P2325	DCPPROGEN	31.0.0008	Further Auxiliary Logic Correc
0292-0180	P2325	NDL	31.0.0009	Further Auxiliary Logic Correc
0292-0181	P1884	UTIL	31.0.0051	TDATABEGIN Field Not Initializ
0292-0184	P2515	DUMPANALY	31.0.0086	DCC and DCP Station Info
0292-0186	P2384	LOGGER	31.0.0023	YTDFILE Update
0292-0193	D2949	DUMPANALY	31.0.0093	Descriptor Analysis
0293-0068	P1705	CONTROLLER	31.0.0034	REMOTESPO
0293-0085	P1705	CONTROLLER	31.0.0034	REMOTESPO
0293-0087	P1191	MCP	31.0.0211	FINDINPUT Vs. UNBRF
0293-0088	D2784	MCP	31.0.0605	NIF and DCPCODE Disk I/O Error
0297-0021	D2931	WFL	31.0.	Task Id in Task Family Declara
0299-0153	P1771	BDMSPLI	31.0.0008	Proper Printing of Length Attr
0299-0159	P1718	MCP	31.0.0367	IOTRACE
0299-0161	P2107	DATACOM	31.0.0527	Recontimeout Failure
0299-0165	P2567	BACKUP	31.0.0018	Buffer Reduction for LIN File
0299-0172	D2822	FORTRAN	31.0.0034	Binding and Statistics
0300-7624	P1083	MCP	31.0.0168	CATALOG ADD Tape Files
0300-7729	P1125	MCP	31.0.0178	Unit Left Assigned
0300-7759	D2576	MCP	31.0.0351	RERUN Vs. EOF
0300-7759	D2576	MCP	31.0.0401	RERUN Vs. EOF
0300-7803	P1730	MCP	31.0.0387	Checkpoint File Header Timesta
0300-7805	P1784	MCP	31.0.0425	CRUNCH Vs. CP
0300-7838	P2272	JOBFORMAT	31.0.0030	SCR Jobs Generate JOBFORMATTER
0303-0002	P1885	BACKUP	31.0.0009	LP# Lost After Forms File
0306-0006	P1767	DUMPALL	31.0.0003	ROUTINE and COPY Statements Re
0306-0032	P2129	ACR	31.0.0084	Control File Open on Abort
0306-0034	P2579	ALGOL	31.0.0181	String in DEFINE Too Long
0307-0027	P1820	ACR	31.0.0054	Faulty Audit Constraint
0308-0002	P1768	DUMPALL	31.0.0004	DMPMT Short Records
0308-0003	P1734	MCP	31.0.0416	FM Vs. Saved LP
0308-0008	P2077	WFL	31.0.0026	Extra BEGIN in WFL Job Deck
0308-0010	P1927	REORG	31.0.0009	Multiple Reel Intermediate Tap
0309-0022	P2214	MCP	31.0.0584	AB Vs. EOF
0309-7155	P1715	MCP	31.0.0356	Dying Stack in WSSHERIFF
0311-0166	P9024	IN-OUTPUT	31.0.0017	FIBLOCK and TIMESTAMP
0311-0166	P9024	IN-OUTPUT	31.0.0132	FIBLOCK and TIMESTAMP
0311-0194	P1896	MCP	31.0.0346	New GOTOSOLVER
0311-0195	P1896	MCP	31.0.0346	New GOTOSOLVER
0312-0042	P1791	ALGOL	31.0.0039	No Error After Direct I/O Stat
0312-0065	P1731	MCP	31.0.0388	Model III Multiplexor
0312-0066	P2106	DATACOM	31.0.0526	Segmented Array Disk I/O
0312-0069	P1877	MCP	31.0.0469	Library Maintenance Reel Switc
0314-0015	P1181	MCP	31.0.0200	II.6 Vs. Timestamp
0314-0021	P2378	LOGANALY	31.0.0036	Character Count on Log Title
0316-0015	P2108	DATACOM	31.0.0528	DCCONTROL Invalid Index
0316-0015	P2108	DCPPROGEN	31.0.0003	DCCONTROL Invalid Index
0316-0022	P1125	MCP	31.0.0178	Unit Left Assigned
0316-0024	P1191	MCP	31.0.0211	FINDINPUT Vs. UNBRF
0318-0011	P1516	BINDER	31.0.0012	Binding Entry Points
0319-0049	P2333	BDMSCOBOL	31.0.0126	Inconsistent Handling of Key C
0319-0064	P1762	BACKUP	31.0.0008	All Copies Not Printed
0319-0065	P2002	WFL	31.0.0023	Zip with Array
0319-0066	P2025	PRINTBIND	31.0.0005	Improved Handling of Unknown I
0319-0068	P2106	DATACOM	31.0.0526	Segmented Array Disk I/O
0319-0070	D2764	ALGOL	31.0.	OWN Arrays in a PROCESSed Proc
0319-0072	P1768	DUMPALL	31.0.0004	DMPMT Short Records
0319-0074	P2441	DCPPROGEN	31.0.0013	LOSSOFCARRIER=DISCONNECT Vs. S
0319-0075	D2949	DUMPANALY	31.0.0093	Descriptor Analysis
0323-0016	P2591	ESPOLINTRN	31.0.0067	FORTRAN BCL Formatting
0323-0019	P1134	MCP	31.0.0193	UR Vs. Resource
0323-0025	D2705	LOADER	31.0.0027	7A Tape Control
0323-0025	D2705	MCP	31.0.0541	7A Tape Control
0323-0025	D2705	MCP	31.0.0554	7A Tape Control
0323-0025	D2705	MCP	31.0.0573	7A Tape Control
0323-0025	D2705	MCP	31.0.0593	7A Tape Control
0323-0025	D2705	MCP	31.0.0634	7A Tape Control
0323-0025	D2705	MCP	31.0.0934	7A Tape Control
0323-0025	D2705	MCP	31.0.0963	7A Tape Control
0323-0025	D2705	UTILOADER	31.0.0005	7A Tape Control
0323-0025	D2705	UTILOADER	31.0.0006	7A Tape Control
0323-0026	P2125	WFL	31.0.0044	Fault on Bad Task Attribute
0324-0009	P2315	COBOL	31.0.0117	INVALID INDEX in Large Code Se
0324-0018	D2874	COBOL	31.0.0134	OCCURS DEPENDING Option
0324-0018	D2874	COBOL74	31.0.0183	OCCURS DEPENDING Option
0324-0038	P2623	DATACOM	31.0.1015	DCTANKING Error Handling Corre
0327-0012	P1112	MCP	31.0.0165	AUTOPRINT Vs. FM Reply
0329-1008	P1734	MCP	31.0.0416	FM Vs. Saved LP
0331-0100	P1766	DUMPALL	31.0.0002	UL OPtion Correction
0332-2904	P2478	COPYAUD-II	31.0.0007	Close New Audit Before Old
0332-3005	P1713	MCP	31.0.0353	Library Maintenance BADFILE Vs
0335-0037	P2126	WFL	31.0.0046	Chargecode Vs. DECK Statement
0335-0046	P1819	ALGOL	31.0.0043	Update Pointer in REPLACE BY S

FTR ACTION TABLE

FTR	NOTE	SOFTWARE	PATCH	DESCRIPTION
0335-0049	P2272	JOBFORMAT	31.0.0030	SCR Jobs Generate JOBFORMATTER
0335-0052	P2571	BACKUP	31.0.0022	Invalid Option Combination
0335-0053	P2382	JOBFORMAT	31.0.0036	CLOSE Types
0338-0011	P2397	DATACOM	31.0.0821	Restarting DSed DCP
0339-0038	P1937	FORTRAN	31.0.0031	INVALID INDEX, Subroutine Call
0342-0012	P9240	MCP	31.0.0159	Tape Parity Retry Vs. Tapemark
0342-0013	P2583	DATACOM	31.0.0979	Subtract Station Fault with No
0342-0032	P1763	COBOL	31.0.0041	User Intrinsic Calls Not Scann
0342-0034	P1820	ACR	31.0.0054	Faulty Audit Constraint
0342-0035	P1820	ACR	31.0.0054	Faulty Audit Constraint
0342-0036	P2009	ACR	31.0.0073	Return Restart Areas
0342-0038	P2272	JOBFORMAT	31.0.0030	SCR Jobs Generate JOBFORMATTER
0342-0041	P2009	ACR	31.0.0073	Return Restart Areas
0342-0042	P2132	ACR	31.0.0087	Eliminate Unnecessary Displays
0342-0051	P2483	RECOVERY	31.0.0052	Quickfix May Fail on Checksumm
0342-0052	P1912	ACR	31.0.0064	Reorganization of Compact Data
0342-0052	P1912	REORG	31.0.0010	Reorganization of Compact Data
0342-0059	P2412	BUILDINQ	31.0.0024	Maximum Size of DMINQDIRECTORY
0342-0065	P2536	MCP	31.0.0906	PATH MARKED OFFLINE
0342-0069	P2497	ACR	31.0.0138	COPYAUDIT Vs. Audit Ioerror Sw
0343-0020	P1715	MCP	31.0.0356	Dying Stack in WSSHERIFF
0343-0025	P1772	RJE	31.0.0026	RJE AUTOPRINT Hang
0343-0026	P2414	RECOVERY	31.0.0048	REBUILD and RECONSTRUCT Back U
0343-0033	P2129	ACR	31.0.0084	Control File Open on Abort
0343-0039	P1820	ACR	31.0.0054	Faulty Audit Constraint
0345-0001	P9209	MCP	31.0.0164	Halt/Load with 64 Mods of Memo
0345-0909	P1761	BACKUP	31.0.0007	Filenames With Special Charact
0345-0910	P2575	BACKUP	31.0.0023	ODT Display Format
0345-2901	P1732	MCP	31.0.0390	ELAPSEDTIMELIMIT
0345-2902	D2842	LOGANALY	31.0.0032	ABORT, ERRORS Vs. Other Option
0345-2903	P2008	WFL	31.0.0025	ELSE No Longer Ignored
0345-2908	P1709	JOBFORMAT	31.0.0019	Includes LSN for Job Summary
0345-2915	P1708	DUMPANALY	31.0.0035	MOD 63 Gives False Tape Error
0345-2916	P1799	MCP	31.0.0442	PRINTLIMIT Vs. PROGRAMDUMP
0347-0046	P1181	MCP	31.0.0200	II.6 Vs. Timestamp
0347-0048	P1191	MCP	31.0.0211	FINDINPUT Vs. UNBRF
0347-0052	P1735	MCP	31.0.0417	Reelswitch Vs. Catalog
0348-0042	P1181	MCP	31.0.0200	II.6 Vs. Timestamp
0348-0044	P1852	LOADER	31.0.0023	TD850 ODT
0348-0044	P1852	LOADER	31.0.0024	TD850 ODT
0348-0056	P2008	WFL	31.0.0025	ELSE No Longer Ignored
0348-0057	P1896	MCP	31.0.0346	New GOTOSOLVER
0348-0060	P2087	SOURCENDL	31.0.0002	Correct Scrolling on TD820/TD8
0348-0064	P1896	MCP	31.0.0346	New GOTOSOLVER
0348-0076	P1736	IN-OUTPUT	31.0.0418	Negative Boolean
0348-0080	P1731	MCP	31.0.0388	Model III Multiplexor
0348-0102	P1864	MCP	31.0.0457	FORGETSPACE Vs. FORM
0348-0103	P1864	MCP	31.0.0457	FORGETSPACE Vs. FORM
0348-0110	P2236	ALGOL	31.0.0147	IF FALSE and $LINEINFO
0348-0123	P2160	ESPOLINTRN	31.0.0055	BASIC, Resize Format Buffer
0348-0128	P2461	UTIL	31.0.0062	COPY AS Error
0348-0128	P2461	UTIL	31.0.0063	COPY AS Error
0348-0128	P2461	UTIL	31.0.0067	COPY AS Error
0348-0129	P1929	UTIL	31.0.0026	Correct Family Not Set
0348-0139	P2374	RJE	31.0.0049	Unit Device Number
0348-0147	P2528	DUMPALL	31.0.0015	NEWFILE Vs. Old File
0348-0148	P2514	MCP	31.0.0968	PURGIT RD Message
0348-0151	P1733	MCP	31.0.0413	READALABEL Vs. EXTMODE
0348-0153	P2212	LOGANALY	31.0.0024	Scratchpad Parity Error
0348-0153	P2212	MCP	31.0.0582	Scratchpad Parity Error
0348-0153	P2212	MCP	31.0.0621	Scratchpad Parity Error
0348-0153	P2212	MCP	31.0.0648	Scratchpad Parity Error
0349-0081	P1862	MCP	31.0.0454	RSNINVALID
0353-0011	P1054	MCP	31.0.0156	IC Diskpacks
0353-0020	P1733	MCP	31.0.0413	READALABEL Vs. EXTMODE
0353-0027	P1786	MCP	31.0.0430	Guardfile Vs. Stackoverflow
0353-0030	P2576	BACKUP	31.0.0024	First Level Filename Vs. Userc
0353-0030	P2576	BACKUP	31.0.0025	First Level Filename Vs. Userc
0361-0026	P1771	BDMSPLI	31.0.0008	Proper Printing of Length Attr
0361-0072	P2026	PRINTBIND	31.0.0006	Correct Handling of PL/I Items
0361-0075	P9221	MCP	31.0.0114	DBSSTACK vs GETSTATUS
0361-0095	P2025	PRINTBIND	31.0.0005	Improved Handling of Unknown I
0361-0101	P1709	JOBFORMAT	31.0.0019	Includes LSN for Job Summary
0361-0104	P2008	WFL	31.0.0025	ELSE No Longer Ignored
0361-0105	D2895	WFL	31.0.	File Attribute SERIALNO Not Im
0361-0107	P1925	RECOVERY	31.0.0025	ABORT Gets WAITING ON ROWLOCKO
0361-0109	D2843	LOGGER	31.0.0019	Summary Reports
0361-0112	P2346	PLINTRN	31.0.0031	Eliminate SEG ARRAY Error in T
0361-0115	P2208	INQ	31.0.0034	INVALID INDEX Using Defines
0361-0120	P2264	LOGGER	31.0.0020	Elapsed Time of Jobs Run Throu
0364-0017	P2579	ALGOL	31.0.0181	String in DEFINE Too Long
0366-0036	P2365	COBOL	31.0.0131	Large Record Descriptions

FTR ACTION TABLE

FTR	NOTE	SOFTWARE	PATCH	DESCRIPTION
0366-0132	P1180	MCP	31.0.0199	TAPEUNIT Vs. VOLUNIT
0366-0132	P1180	MCP	31.0.0230	TAPEUNIT Vs. VOLUNIT
0366-0133	P1725	MCP	31.0.0380	GENERATION
0366-0142	P1720	MCP	31.0.0366	Checkpoint
0366-9021	P1712	MCP	31.0.0352	VOLUME DELETE
0366-9022	P1712	MCP	31.0.0352	VOLUME DELETE
0366-9023	P1724	MCP	31.0.0377	CATBLKSIZEF
0367-0010	P2034	DASDL	31.0.0043	Syntax Error for One Global Da
0368-0051	D2873	COBOL	31.0.	Sort Variable Length Records
0368-0080	P2250	PLINTRN	31.0.0030	ISAM, Eliminate Superhalt
0368-0084	P2272	JOBFORMAT	31.0.0030	SCR Jobs Generate JOBFORMATTER
0368-0090	P2473	ACR	31.0.0129	Incorrect Statistics Totals
0368-0090	P2473	ACR	31.0.0132	Incorrect Statistics Totals
0369-0031	P9169	MCP	31.0.0085	Processor Priority
0369-0045	P1770	LOADER	31.0.0020	OLAYROW Greater Than 1200
0369-0051	P2309	BUILDINQ	31.0.0020	Vague Link and Subset Referenc
0369-0054	P2207	DASDL	31.0.0039	INFO Table Incorrect
0370-0010	P2259	COBOL	31.0.0110	ANSI74 LINAGE WRITE AT END OF
0372-0203	P2274	MCP	31.0.0692	CLOSE Errors Fatal
0372-0203	P2274	MCP	31.0.0825	CLOSE Errors Fatal
0372-0203	P2274	MCP	31.0.0976	CLOSE Errors Fatal
0372-0300	P2468	FILEDATA	31.0.0010	6250 BPI Tapes Recognized
0372-1201	P1921	DASDL	31.0.0030	Global Chain Not Overlayed
0372-2598	P2250	PLINTRN	31.0.0030	ISAM, Eliminate Superhalt
0373-0034	P2087	SOURCENDL	31.0.0002	Correct Scrolling on TD820/TD8
0373-0035	P2088	SOURCENDL	31.0.0001	Changes to POLLCONTENTION Line
0373-0045	P1902	REORG	31.0.0011	Compact Data Sets with Record
0373-0046	P1912	ACR	31.0.0064	Reorganization of Compact Data
0373-0046	P1912	REORG	31.0.0010	Reorganization of Compact Data
0373-0047	P1038	DASDL	31.0.0042	INVALID INDEX
0374-0015	P2314	COBOL	31.0.0115	LOCK Statements
0374-0033	P1124	LTTABLEGEN	31.0.0002	User Specified Tables
0374-0045	P2236	ALGOL	31.0.0147	IF FALSE and $LINEINFO
0374-0063	P1788	MCP	31.0.0433	LOOKFORIT Control State
0374-0065	P2310	MCP	31.0.0729	Class Count UFLO
0374-0074	P2310	MCP	31.0.0729	Class Count UFLO
0374-0080	P2305	MCP	31.0.0737	STICKY Memory
0374-0090	P1722	MCP	31.0.0373	Reader Buss Parity Error
0374-0102	P2008	WFL	31.0.0025	ELSE No Longer Ignored
0374-0121	P2258	COBOL	31.0.0105	INVALID INDEX in Bad REPORT Wr
0374-0135	P2480	DASDL	31.0.0056	Disallow Data Set with No Name
0374-0136	P2489	CONTROLLER	31.0.0102	PRINTLABEL
0374-0140	P2382	JOBFORMAT	31.0.0036	CLOSE Types
0374-0141	P2561	RJE	31.0.0051	"*ME" Fails on Specific Filena
0380-0010	P1865	MCP	31.0.0459	Library Maintenance Open Error
0381-0027	P1715	MCP	31.0.0356	Dying Stack in WSSHERIFF
0382-0024	P1715	MCP	31.0.0356	Dying Stack in WSSHERIFF
0384-0005	D2788	ALGOL	31.0.	Clarification of INTEGER Funct
0386-0009	P2091	SOURCENDL	31.0.0008	Correct Transmission of NUL Ch
0386-0021	P1734	MCP	31.0.0416	FM Vs. Saved LP
0386-0026	P2014	RECOVERY	31.0.0026	Correct Rebuild Restart
0390-0013	P1715	MCP	31.0.0356	Dying Stack in WSSHERIFF
0390-0014	P2120	WFL	31.0.0036	History
0390-0016	P1873	WFL	31.0.0018	Incorrect Code for File Title
0390-0017	P1876	MCP	31.0.0467	RESERVE AS
0393-0075	P1027	MCP	31.0.0148	WASTEMPATCHECK
0393-0094	P1123	JOBFORMAT	31.0.0006	SEG ARRAY Printing MCS String
0393-0134	P1717	MCP	31.0.0364	MOVE Vs. LOG
0393-0152	P2244	ACR	31.0.0098	Return of Compact Record
0393-0162	P1916	ACR	31.0.0069	Two Halt/Loads Causes Recovery
0393-0162	P1916	ACR	31.0.0074	Two Halt/Loads Causes Recovery
0393-0187	P2627	ACR	31.0.0149	Deadlock During Update and Del
0393-0191	P2627	ACR	31.0.0149	Deadlock During Update and Del
0397-0003	P1516	BINDER	31.0.0012	Binding Entry Points
0397-0005	P1171	SCR	31.0.0011	VERIFY Does Not Recognize 5N D
0397-0020	P1819	ALGOL	31.0.0043	Update Pointer in REPLACE BY S
0397-0021	P2583	DATACOM	31.0.0979	Subtract Station Fault with No
0399-9011	P1054	MCP	31.0.0156	IC Diskpacks
0399-9042	P2253	COBOL	31.0.0048	SELECT Statement Syntax Errors
0399-9100	P9226	MCP	31.0.0133	Sticky MEM Recursive GETSPACE
0399-9132	P2623	DATACOM	31.0.1015	DCTANKING Error Handling Corre
0399-9191	P2255	COBOL	31.0.0102	UNITS Attribute
0399-9204	P2320	COBOL	31.0.0125	Moves of Numeric Literals
0399-9226	P1940	LOGANALY	31.0.0027	Firmware ID in LH ODT Command
0399-9228	D2854	COBOL	31.0.	MOVE Statement
0399-9905	P1029	MCP	31.0.0150	Queue Error Vs. THEQUE
0401-0025	P1516	BINDER	31.0.0012	Binding Entry Points
0401-0026	P1816	RJE	31.0.0033	QT Printer Files Saved Over Ha
0401-0027	P2295	COBOL	31.0.0109	Exception Code Erroneously Che
0401-0031	P1516	BINDER	31.0.0012	Binding Entry Points
0401-0035	P2501	INQ	31.0.0045	Loss of Minus Sign
0401-0036	P2607	BUILDINQ	31.0.0026	Selection Using Key Data Items

FTR ACTION TABLE

FTR	NOTE	SOFTWARE	PATCH	DESCRIPTION
0402-0002	P1791	ALGOL	31.0.0039	No Error After Direct I/O Stat
0402-0018	P1516	BINDER	31.0.0012	Binding Entry Points
0402-0100	P2592	FORTRAN	31.0.0047	Core Estimate with $SET SEPARA
0405-0006	P2038	COBOL	31.0.0053	Comparison of Index Data Names
0405-0022	P2062	COBOL	31.0.0084	SELECTION Expressions
0405-0041	P1717	MCP	31.0.0364	MOVE Vs. LOG
0405-0045	P2495	ACR	31.0.0131	DS DBS Vs. File Attributes
0405-0045	P2495	ACR	31.0.0135	DS DBS Vs. File Attributes
0405-0045	P2495	ACR	31.0.0136	DS DBS Vs. File Attributes
0405-0045	P2495	ACR	31.0.0137	DS DBS Vs. File Attributes
0405-0045	P2495	DMCTL	31.0.0016	DS DBS Vs. File Attributes
0405-0045	P2495	DMCTL	31.0.0020	DS DBS Vs. File Attributes
0405-0046	P1802	ACR	31.0.0050	Audit File Switch for disk or
0405-0048	P1923	LOADDUMP	31.0.0004	INVALID INDEX with Deleted Str
0405-0053	P2095	ACR	31.0.0077	Correct Deletion in Unordered
0406-0022	P2332	BDMSCOBOL	31.0.0122	INVALID INDEX
0406-0027	P1715	MCP	31.0.0356	Dying Stack in WSSHERIFF
0406-0030	P2008	WFL	31.0.0025	ELSE No Longer Ignored
0406-0039	P1830	WFL	31.0.0017	Prevent SEG ARRAY Error when J
0406-0044	P1919	BUILDINQ	31.0.0008	Verified Link
0406-0046	P1821	ACR	31.0.0055	INVALID INDEX after Audit Time
0406-0049	D2695	BUILDINQ	31.0.0011	Sets with Group Keys
0406-0051	P2007	WFL	31.0.0021	REMOVE Command
0406-0055	P2010	ACR	31.0.0075	Checksum Error
0406-0055	P2010	PROPERTIES	31.0.0011	Checksum Error
0406-0055	P2010	RECOVERY	31.0.0027	Checksum Error
0406-0055	P2010	RECOVERY	31.0.0032	Checksum Error
0406-0056	P1775	DASDL	31.0.0021	Eliminate SEG ARRAY Error
0406-0057	P1786	MCP	31.0.0430	Guardfile Vs. Stackoverflow
0406-0059	P1925	RECOVERY	31.0.0025	ABORT Gets WAITING ON ROWLOCKO
0406-0060	P2130	ACR	31.0.0085	ROWLOCKOUTAUDIT Errors
0406-0060	P2130	PROPERTIES	31.0.0013	ROWLOCKOUTAUDIT Errors
0406-0060	P2130	RECOVERY	31.0.0031	ROWLOCKOUTAUDIT Errors
0406-0069	P2570	BACKUP	31.0.0021	Filenames Limited to 60 Charac
0406-0072	P2484	UTIL	31.0.0065	INITIALIZE of Global Data
0410-0003	P1136	IN-OUTPUT	31.0.0195	Tape Close Vs. MYUSE
0410-0006	P1939	FORTRAN	31.0.0033	Run Time Problems with Very La
0410-0021	P1715	MCP	31.0.0356	Dying Stack in WSSHERIFF
0410-0026	P2204	RJE	31.0.0040	"*ME COPIES 2" Vs. Resize of F
0410-0027	P2011	BUILDINQ	31.0.0009	Occurring Groups
0410-0029	P2188	FILECOPY	31.0.0008	Punched Output
0410-0030	P1847	FILECOPY	31.0.0012	II.9 WFL
0410-0030	P1847	FILECOPY	31.0.0013	II.9 WFL
0410-0030	P1847	FILECOPY	31.0.0015	II.9 WFL
0410-0031	P2125	WFL	31.0.0044	Fault on Bad Task Attribute
0410-0037	P2563	RJE	31.0.0054	REMCP File Titles on Word Boun
0411-0006	P2125	WFL	31.0.0044	Fault on Bad Task Attribute
0415-0025	P2021	MCP	31.0.0483	Allow Statistics To Be Printed
0415-0032	P9149	MCP	31.0.0066	BADROW
0415-0033	P9149	MCP	31.0.0066	BADROW
0415-0060	P2299	BDMSCOBOL	31.0.0112	Bad Syntax Checking in DB Stat
0415-0108	P1738	ACR	31.0.0041	Statistics Midnight Overlap
0415-0120	P2330	BDMSCOBOL	31.0.0116	Listing of the Variable Format
0415-0154	P1738	ACR	31.0.0041	Statistics Midnight Overlap
0415-0159	P1924	LOADDUMP	31.0.0005	Title Attribute Error
0415-0163	P2239	ALGOL	31.0.0150	Compiler Loop on DATABASE Decl
0415-0168	P1738	ACR	31.0.0041	Statistics Midnight Overlap
0415-0170	P1922	ACR	31.0.0066	INV OP in IODISKADDRESS
0415-0170	P1922	DMCTL	31.0.0007	INV OP in IODISKADDRESS
0415-0172	D2608	ACR	31.0.0021	Fault Handling in ACCESSROUTIN
0415-0173	P1820	ACR	31.0.0054	Faulty Audit Constraint
0415-0176	P1915	ACR	31.0.0068	COPYAUDIT Not Zipped if Error
0415-0190	P1798	MCP	31.0.0439	DMS Programs Swapped Out "FORE
0415-0193	P1932	UTIL	31.0.0033	IOERROR on Tape
0415-0195	P1907	ACR	31.0.0057	Reset Syncpoint to Lower Value
0415-0198	P2495	ACR	31.0.0131	DS DBS Vs. File Attributes
0415-0198	P2495	ACR	31.0.0135	DS DBS Vs. File Attributes
0415-0198	P2495	ACR	31.0.0136	DS DBS Vs. File Attributes
0415-0198	P2495	ACR	31.0.0137	DS DBS Vs. File Attributes
0415-0198	P2495	DMCTL	31.0.0016	DS DBS Vs. File Attributes
0415-0198	P2495	DMCTL	31.0.0020	DS DBS Vs. File Attributes
0415-0199	P1798	MCP	31.0.0439	DMS Programs Swapped Out "FORE
0415-0202	P2202	BDMSCOBOL	31.0.0095	Bad DMS Invoke Listing
0415-0204	P1821	ACR	31.0.0055	INVALID INDEX after Audit Time
0415-0207	P1916	ACR	31.0.0069	Two Halt/Loads Causes Recovery
0415-0207	P1916	ACR	31.0.0074	Two Halt/Loads Causes Recovery
0415-0208	P1913	ACR	31.0.0065	Duplicate Audit Block If Switc
0415-0209	P2027	PRINTAUDIT	31.0.0005	Hex Dump of Block
0415-0214	P1921	DASDL	31.0.0030	Global Chain Not Overlayed
0415-0217	P1798	MCP	31.0.0439	DMS Programs Swapped Out "FORE
0415-0219	P1798	MCP	31.0.0439	DMS Programs Swapped Out "FORE
0415-0221	P2305	MCP	31.0.0737	STICKY Memory

FTR ACTION TABLE

FTR	NOTE	SOFTWARE	PATCH	DESCRIPTION
0415-0230	P2295	COBOL	31.0.0109	Exception Code Erroneously Che
0415-0240	D2962	COBOL	31.0.	OBJECT-COMPUTER Paragraph
0415-0242	P2497	ACR	31.0.0138	COPYAUDIT Vs. Audit Ioerror Sw
0417-2012	P2001	WFL	31.0.0022	Zip With Array
0417-2014	P1721	MCP	31.0.0371	LINKLISTINSERT Vs. XTEND
0417-2015	P1713	MCP	31.0.0353	Library Maintenance BADFILE Vs
0417-2023	P2296	FILEDATA	31.0.0008	Allow CANDE LFILES with No Di
0419-0008	D2874	COBOL	31.0.0134	OCCURS DEPENDING Option
0419-0008	D2874	COBOL74	31.0.0183	OCCURS DEPENDING Option
0419-0009	D2874	COBOL	31.0.0134	OCCURS DEPENDING Option
0419-0009	D2874	COBOL74	31.0.0183	OCCURS DEPENDING Option
0421-0005	P2122	WFL	31.0.0039	Reserved Words as Familynames
0421-0007	P2300	MCP	31.0.0719	MPX Vs. Tape Parity
0422-0002	P2158	INTERFACE	31.0.0011	INVALID INDEX in Data Base Int
0427-0007	P2577	LOADER	31.0.0035	B9246 Drum Printer in Dump
0427-0010	P1860	IN-OUTPUT	31.0.0452	FIBEOF
0429-0041	P2240	ALGOL	31.0.0151	Remove "?" from ALPHA6
0429-9005	P2358	DUMPANALY	31.0.0067	OVERLAYCF
0429-9005	P2358	MCP	31.0.0783	OVERLAYCF
0430-0004	P1715	MCP	31.0.0356	Dying Stack in WSSHERIFF
0430-0007	P1825	INQ	31.0.0031	Display of All-Fraction Items
0430-0009	P1828	REORG	31.0.0006	Link Fixup in Variable Format
0430-0010	P1903	REORG	31.0.0012	Generation of Manual Subsets
0430-0011	P1926	REORG	31.0.0008	Block Initialization for Direc
0430-0014	P2387	ACR	31.0.0118	Erroneous Not Found Result
0430-0017	P2189	LOGANALY	31.0.0022	Cardnumber Vs. P3 Word
0431-0007	D2836	ALGOL	31.0.	Arithmetic Function VALUE
0435-0043	P2088	SOURCENDL	31.0.0001	Changes to POLLCONTENTION Line
0435-0044	P1901	ACR	31.0.0072	LOCK TO MODIFY DETAILS
0435-0047	P2249	INTERFACE	31.0.0010	GROUP Items Invoked Incorrectl
0435-0049	P2266	MCP	31.0.0651	Security Vs. Backup
0435-0050	P2273	MCP	31.0.0689	RESERVE Loop
0435-0053	P2388	BUILDINQ	31.0.0023	Embedded Sets in Logical Data
0435-0054	P2389	INQ	31.0.0042	Find Via Self-Correcting Link
0436-0003	P2296	FILEDATA	31.0.0008	Allow CANDE LFILES with No Di
0439-0001	P1763	COBOL	31.0.0041	User Intrinsic Calls Not Scann
0439-0017	P1879	ALGOL	31.0.0049	Syntax Errors Not Being Report
0439-0018	P2008	WFL	31.0.0025	ELSE No Longer Ignored
0439-0038	D2896	WFL	31.0.	Change Filename Using "(PACK)"
0444-0002	P1715	MCP	31.0.0356	Dying Stack in WSSHERIFF
0445-0015	D2959	COBOL	31.0.	File Attribute IORECORDNUM
0447-0021	P2109	DATACOM	31.0.0529	Set Up Line Control Index
0451-0007	P2333	BDMSCOBOL	31.0.0126	Inconsistent Handling of Key C
0454-9003	P1666	COBOL	31.0.0040	FILE-LIMIT Statement Caused IN
0454-9003	P1666	COBOL74	31.0.0100	FILE-LIMIT Statement Caused IN
0454-9014	P2448	MCP	31.0.0858	CL READALABEL
0454-9902	P2262	LOGANALY	31.0.0030	Garbage Output on UNSORTED
0454-9904	P1875	WFL	31.0.0019	Concatenation of More Than 15
0455-0002	P1712	MCP	31.0.0352	VOLUME DELETE
0455-0006	P1725	MCP	31.0.0380	GENERATION
0455-0007	P1715	MCP	31.0.0356	Dying Stack in WSSHERIFF
0460-0005	P1850	FORTRAN	31.0.0020	Multiple Entry Points with OPT
0460-0007	D2822	FORTRAN	31.0.0034	Binding and Statistics
0460-0010	P1874	FORTRAN	31.0.0026	Binding FORTRAN and ALGOL
0461-0078	P1734	MCP	31.0.0416	FM Vs. Saved LP
0462-0005	P2500	INQ	31.0.0044	Use of Defines in Certain Cont
0462-0008	P2398	UTIL	31.0.0060	Incorrect Selection of Structu
0463-9017	D2879	PLI	31.0.	Clarification of FIXEDOVERFLOW
0463-9036	P2308	CONTROLLER	31.0.0083	CONT6 GETDISK
0464-0007	P1826	INQ	31.0.0032	Display of Null Items
0464-0008	P2580	BACKUP	31.0.0026	BACKUP Vs. System Without DISK
0464-0010	P2557	NDL	31.0.0017	Exchanged DCP Vs. Auxiliary Lo
0464-0053	D2787	ALGOL	31.0.	READ on the B5500 Vs. the B670
0467-0002	D2894	WFL	31.0.	WFL Manual Corrections
0467-0006	P2318	COBOL	31.0.0121	XREF
0467-0016	P1782	MCP	31.0.0389	Guardfile Vs. DS
0467-0021	P1938	FORTRAN	31.0.0032	Internal Compiler Fields Excee
0467-0030	P1887	COMPARE	31.0.0003	Usercode Attached File Titles
0467-0040	P2008	WFL	31.0.0025	ELSE No Longer Ignored
0467-0049	P2272	JOBFORMAT	31.0.0030	SCR Jobs Generate JOBFORMATTER
0467-0053	P2318	COBOL	31.0.0121	XREF
0467-0054	P1768	DUMPALL	31.0.0004	DMPMT Short Records
0471-0001	D2786	ALGOL	31.0.	$LOADINFO, $DUMPINFO Yield Syn
0473-0001	P2443	DUMPALL	31.0.0013	No File when Equating Tape to
0473-0006	P1794	RJE	31.0.0027	WFLCOMPILER Not Going to EOT
0474-1020	P1820	ACR	31.0.0054	Faulty Audit Constraint
0474-1022	P1738	ACR	31.0.0041	Statistics Midnight Overlap
0474-1024	P1715	MCP	31.0.0356	Dying Stack in WSSHERIFF
0474-1040	P2129	ACR	31.0.0084	Control File Open on Abort
0474-1042	P1825	INQ	31.0.0031	Display of All-Fraction Items
0475-0005	P2250	PLINTRN	31.0.0030	ISAM, Eliminate Superhalt
0476-0001	P2364	MCP	31.0.0779	SIB Environment

FTR ACTION TABLE

FTR	NOTE	SOFTWARE	PATCH	DESCRIPTION
0479-0002	P2125	WFL	31.0.0044	Fault on Bad Task Attribute
0488-0003	P1444	BUILDINQ	31.0.0013	Automatically Selects Restart
0490-0001	P2202	BDMSCOBOL	31.0.0095	Bad DMS Invoke Listing
0490-0002	P1768	DUMPALL	31.0.0004	DMPMT Short Records
0490-0004	P1920	DASDL	31.0.0029	Valid OPEN PARTITIONS Attribut
0491-0001	P2515	DUMPANALY	31.0.0086	DCC and DCP Station Info
0491-0002	P2341	COBOL	31.0.0132	COBOL No Longer Ignores Mispla
0491-0003	P2474	ACR	31.0.0134	Free Records on Abort Exceptio
0502-0002	P2250	PLINTRN	31.0.0030	ISAM, Eliminate Superhalt
1005-0006	P2113	ALGOL	31.0.0134	INVALID OP in Freefield WRITE
1005-0011	P2428	MCP	31.0.0848	Uninitiated I/O Dump
1005-0012	P2430	MCP	31.0.0867	New High Order Priority Scheme
1005-0012	P2430	SCR	31.0.	New High Order Priority Scheme
1005-0013	P2433	MCP	31.0.0869	Correct TERMINATE Vs. BLOCKEXI
1009-0021	P2308	CONTROLLER	31.0.0083	CONT6 GETDISK
1009-0025	P2208	CONTROLLER	31.0.0083	CONT6 GETDISK
1009-0029	P2255	COBOL	31.0.0102	UNITS Attribute
1009-0030	P1823	BUILDINQ	31.0.0005	Enforce Answer on Update Quest
1009-0033	P1824	BUILDINQ	31.0.0006	Data Set Selection Option Enfo
1011-0003	P2583	DATACOM	31.0.0979	Subtract Station Fault with No
1012-0001	P2256	COBOL	31.0.0103	Code Segment Size Exceeded
1013-0001	P2211	REORG	31.0.0025	Fixup of Empty Variable Format
1016-0016	P1763	COBOL	31.0.0041	User Intrinsic Calls Not Scann
1016-0032	P2580	BACKUP	31.0.0026	BACKUP Vs. System Without DISK
1016-9004	P2286	MCP	31.0.0652	GETSTATUS UNIT Vs. DISKPACKSEA
1016-9012	P1952	MCP	31.0.0645	Tape Open
1016-9025	P2296	FILEDATA	31.0.0008	Allow CANDE LFILES with No Di
1016-9027	P2004	BACKUP	31.0.0011	BFILE Label Equaton
1020-5007	P2126	WFL	31.0.0046	Chargecode Vs. DECK Statement
1020-5014	P1705	CONTROLLER	31.0.0034	REMOTESPO
1021-0009	P2007	WFL	31.0.0021	REMOVE Command
1021-0011	P2587	ALGOL	31.0.0182	Syntax Error with $INTRINSICS
1021-0504	P2417	UTIL	31.0.0061	FLUSHDB Option Correction
1024-0005	P2313	COBOL	31.0.0114	Incorrect SORT Statement
1027-0015	P2255	COBOL	31.0.0102	UNITS Attribute
1028-0007	P2045	COBOL	31.0.0062	Simple Literal Compares
1033-0001	P2189	LOGANALY	31.0.0022	Cardnumber Vs. P3 Word
1035-0011	P2252	COBOL	31.0.0097	Statistics Not Correct for STO
1035-0011	P2252	COBOL74	31.0.0149	Statistics Not Correct for STO
1035-0013	P2377	COBOL	31.0.0136	INSPECT Verb Slow Execution
1035-0014	D2894	WFL	31.0.	WFL Manual Corrections
1036-9006	P2261	DUMPALL	31.0.0009	INTMODE of HEX or SINGLE
1036-9401	P1860	IN-OUTPUT	31.0.0452	FIBEOF
1036-9918	P2272	JOBFORMAT	31.0.0030	SCR Jobs Generate JOBFORMATTER
1037-0009	P2481	INQ	31.0.0043	Fault Alpha Control Item Break
1038-9901	P1811	COBOLTABLE	31.0.0003	LISTACK Option
1038-9904	P2261	DUMPALL	31.0.0009	INTMODE of HEX or SINGLE
1039-0001	P1772	RJE	31.0.0026	RJE AUTOPRINT Hang
1040-0002	P1795	RJE	31.0.0028	LSNRAY Vs. Stations with NO-LI
1040-0005	P1814	RJE	31.0.0030	Swapping Print Queues Vs. Stat
1040-0007	P2536	MCP	31.0.0906	PATH MARKED OFFLINE
1041-0001	P1841	ESPOLINTRN	31.0.0030	BASIC Binary Files
1042-0012	P1772	RJE	31.0.0026	RJE AUTOPRINT Hang
1042-0017	P2580	BACKUP	31.0.0026	BACKUP Vs. System Without DISK
1042-0018	P2296	FILEDATA	31.0.0008	Allow CANDE LFILES with No Di
1042-0020	P2397	DATACOM	31.0.0821	Restarting DSed DCP
1042-0023	P2203	SOURCENDL	31.0.0012	Line TOGS and TALLYS Cleared
1042-0036	P2383	JOBFORMAT	31.0.0037	JOB ENTERED SYSTEM Time
1046-0001	P2333	BDMSCOBOL	31.0.0126	Inconsistent Handling of Key C
1046-0013	P1715	MCP	31.0.0356	Dying Stack in WSSHERIFF
1046-0018	P2295	COBOL	31.0.0109	Exception Code Erroneously Che
1046-0021	P2347	ACR	31.0.0106	Ordered Available Table Corrup
1046-0022	P2350	ACR	31.0.0110	ORDERED Data Set Improper Audi
1046-0022	P2350	ACR	31.0.0124	ORDERED Data Set Improper Audi
1046-0025	P2210	REORG	31.0.0024	Index Random, Random Shadow Fi
1046-0027	P2260	COBOL	31.0.0111	Random File Getting Serial Wri
1046-0028	D2856	DUMPANALY	31.0.	Interactive Syntax Errors
1046-0029	P2350	ACR	31.0.0110	ORDERED Data Set Improper Audi
1046-0029	P2350	ACR	31.0.0124	ORDERED Data Set Improper Audi
1046-0030	P2136	REORG	31.0.0017	Unused Fold Words
1046-0034	P2295	COBOL	31.0.0109	Exception Code Erroneously Che
1046-0035	P1768	DUMPALL	31.0.0004	DMPMT Short Records
1046-0037	P2193	LOGANALY	31.0.0021	Halt/Load Records
1046-0038	P2576	BACKUP	31.0.0024	First Level Filename Vs. Userc
1046-0038	P2576	BACKUP	31.0.0025	First Level Filename Vs. Userc
1046-0048	P2354	DASDL	31.0.0047	Text Generated for Group Key w
1047-0005	P2268	MCP	31.0.0662	CL Vs. AB
1049-0008	P2203	SOURCENDL	31.0.0012	Line TOGS and TALLYS Cleared
1050-0002	P2404	NDL	31.0.0012	Eliminate Looping on Severe Er
1050-0009	P2203	SOURCENDL	31.0.0012	Line TOGS and TALLYS Cleared
1050-0013	P1942	RJE	31.0.0037	Restartfile Name SEG ARRAY Err
1050-0018	P1037	RJE	31.0.0046	Reconfiguration with Active AU

FTR ACTION TABLE

FTR	NOTE	SOFTWARE	PATCH	DESCRIPTION
1050-0019	P1814	RJE	31.0.0030	Swapping Print Queues Vs. Stat
1050-0022	P2339	COBOL	31.0.0128	Analyze Option Duplicating Lin
1050-0024	D2844	LOGGER	31.0.0021	Last STATISTICS Interval Saved
1050-0026	P1579	RJE	31.0.0045	STATIONID or USER Changes
1052-0003	P1742	ACR	31.0.0042	CREATE/STORE Incorrect for Rem
1052-0029	P2536	MCP	31.0.0906	PATH MARKED OFFLINE
1056-0005	P2208	INQ	31.0.0034	INVALID INDEX Using Defines
1060-0001	P1912	ACR	31.0.0064	Reorganization of Compact Data
1060-0001	P1912	REORG	31.0.0010	Reorganization of Compact Data
1062-0004	P2062	COBOL	31.0.0084	SELECTION Expressions
1067-0001	P1779	UTIL	31.0.0015	INVALID INDEX Printing Ordered
1067-0003	P1911	ACR	31.0.0063	Remaps of Ordered Data Sets wi
1067-0003	P1911	PROPERTIES	31.0.0010	Remaps of Ordered Data Sets wi
1067-0005	P2352	ACR	31.0.0112	ORDERED Data Set Loop
1067-0009	P2202	BDMSCOBOL	31.0.0095	Bad DMS Invoke Listing
1067-0012	P2207	DASDL	31.0.0039	INFO Table Incorrect
1067-0013	P2191	DASDL	31.0.0036	Erroneous Key Changed Errors
1067-0015	P2347	ACR	31.0.0106	Ordered Available Table Corrup
1067-0016	P2351	ACR	31.0.0111	ORDERED ADDRESSCHECK, CHECKSUM
1067-0016	P2351	ACR	31.0.0123	ORDERED ADDRESSCHECK, CHECKSUM
1067-0018	P2380	COBOL	31.0.0138	SORT Dynamic File Attributes
1067-0022	P1916	ACR	31.0.0069	Two Halt/Loads Causes Recovery
1067-0022	P1916	ACR	31.0.0074	Two Halt/Loads Causes Recovery
1067-0024	P1820	ACR	31.0.0054	Faulty Audit Constraint
1067-0030	P2482	RECOVERY	31.0.0051	"FAILED TO SET END-OF-FILE" Me
1067-0033	P2546	MCP	31.0.0850	DD Vs. CM
1067-0033	P2546	MCP	31.0.0860	DD Vs. CM
1069-0009	P2390	UTIL	31.0.0055	CFNAME Array Size Increased
1069-0013	P2589	REORG	31.0.0032	Record Format Change
1069-0014	P2355	DASDL	31.0.0048	Last Scan Saved
1071-0004	P2360	ACR	31.0.0104	Store of ORDERED with BLOCKSIZ
1071-0005	P2360	ACR	31.0.0104	Store of ORDERED with BLOCKSIZ
1079-0001	P2351	ACR	31.0.0111	ORDERED ADDRESSCHECK, CHECKSUM
1079-0001	P2351	ACR	31.0.0123	ORDERED ADDRESSCHECK, CHECKSUM
1079-0002	P2347	ACR	31.0.0106	Ordered Available Table Corrup
1081-0001	P2444	NDL	31.0.0015	Erroneous Syntax Error on Defi
1086-0009	P1802	ACR	31.0.0050	Audit File Switch for disk or
1094-1064	P2473	ACR	31.0.0129	Incorrect Statistics Totals
1094-1064	P2473	ACR	31.0.0132	Incorrect Statistics Totals
1100-0004	P2342	COBOL	31.0.0133	SEARCH Nested in IF Statement
1100-0007	P2010	ACR	31.0.0075	Checksum Error
1100-0007	P2010	PROPERTIES	31.0.0011	Checksum Error
1100-0007	P2010	RECOVERY	31.0.0027	Checksum Error
1100-0007	P2010	RECOVERY	31.0.0032	Checksum Error
1101-0001	P2254	COBOL	31.0.0099	External PROCEDURE Declaration
1101-0006	P2254	COBOL	31.0.0099	External PROCEDURE Declaration
1104-9001	P2360	ACR	31.0.0104	Store of ORDERED with BLOCKSIZ
1108-0005	P1794	RJE	31.0.0027	WFLCOMPILER Not Going to EOT
1116-0012	P2484	UTIL	31.0.0065	INITIALIZE of Global Data
1127-0014	P2479	DASDL	31.0.0055	Maximum Length DBNAME In Quote
1127-0023	P2584	MCP	31.0.0983	Avoid Extra Mom on B6800 Negat
1143-0005	P2385	PLI	31.0.0050	Missing Commas in File Options
1143-0007	P2513	MCP	31.0.0969	CATALOG PURGE
1143-0008	P2445	PLI	31.0.0051	Miscellaneous Faults Using PIC
1143-0012	P2386	PLI	31.0.0049	WRITE Statement Without "FROM"
1143-0022	P2580	BACKUP	31.0.0026	BACKUP Vs. System Without DISK
1143-0027	P2555	FILECOPY	31.0.0019	(X) Vs. *X in EXCLUDE List
1143-0029	D2568	DASDL	31.0.0013	Allow More Structures, Items P
1143-0029	D2568	PROPERTIES	31.0.0002	Allow More Structures, Items P
1143-0030	D2961	BDMSCOBOL	31.0.	DM Attribute <structure number
1143-0043	P2461	UTIL	31.0.0062	COPY AS Error
1143-0043	P2461	UTIL	31.0.0063	COPY AS Error
1143-0043	P2461	UTIL	31.0.0067	COPY AS Error
1143-0061	P2556	LOGGER	31.0.0024	ALINK Vs. MATCHM (BOJ,EOJ)
1143-0068	P1775	DASDL	31.0.0021	Eliminate SEG ARRAY Error
7116-0063	P2130	ACR	31.0.0085	ROWLOCKOUTAUDIT Errors
7116-0063	P2130	PROPERTIES	31.0.0013	ROWLOCKOUTAUDIT Errors
7116-0063	P2130	RECOVERY	31.0.0031	ROWLOCKOUTAUDIT Errors
7122-0018	D2893	COBOL	31.0.	DMS Exception Handling
7126-0113	P2612	INQ	31.0.0047	Comparisons of Literals
7142-0049	P1916	ACR	31.0.0069	Two Halt/Loads Causes Recovery
7142-0049	P1916	ACR	31.0.0074	Two Halt/Loads Causes Recovery
7144-0030	P2010	ACR	31.0.0075	Checksum Error
7144-0030	P2010	PROPERTIES	31.0.0011	Checksum Error
7144-0030	P2010	RECOVERY	31.0.0027	Checksum Error
7144-0030	P2010	RECOVERY	31.0.0032	Checksum Error
7144-0031	P2453	ACR	31.0.0127	Erroneous DIRECT KEY Corrupted
7144-0038	P2130	ACR	31.0.0085	ROWLOCKOUTAUDIT Errors
7144-0038	P2130	PROPERTIES	31.0.0013	ROWLOCKOUTAUDIT Errors
7144-0038	P2130	RECOVERY	31.0.0031	ROWLOCKOUTAUDIT Errors
7144-0042	P2495	ACR	31.0.0131	DS DBS Vs. File Attributes
7144-0042	P2495	ACR	31.0.0135	DS DBS Vs. File Attributes

FTR ACTION TABLE

FTR	NOTE	SOFTWARE	PATCH	DESCRIPTION
7144-0042	P2495	ACR	31.0.0136	DS DBS Vs. File Attributes
7144-0042	P2495	ACR	31.0.0137	DS DBS Vs. File Attributes
7144-0042	P2495	DMCTL	31.0.0016	DS DBS Vs. File Attributes
7144-0042	P2495	DMCTL	31.0.0020	DS DBS Vs. File Attributes
7149-0160	P2501	INQ	31.0.0045	Loss of Minus Sign
7149-0164	P2603	ACR	31.0.0144	Partitioned Data Set Inaccessa
7149-0173	P1871	UTIL	31.0.0024	Missing Entry in HL File
7169-0001	P2216	UTIL	31.0.0050	TAPEDIRECTORY Command Problems
7169-0002	P2104	UTIL	31.0.0039	Multiprocessor Problems
7169-0003	P2398	UTIL	31.0.0060	Incorrect Selection of Structu
7169-0004	P2459	RECOVERY	31.0.0050	Timing problem in REBUILD/ROLL
7169-0004	P2459	RECOVERY	31.0.0053	Timing problem in REBUILD/ROLL
7169-0005	P2458	RECOVERY	31.0.0049	Bad Retry Logic
7169-0609	P2505	RECOVERY	31.0.0055	Reconstruct Ignores Quiet Poin

PATCH TABLE

SOFTWARE	PATCH	PRI	NOTE	DESCRIPTION
ACR	31.0.0009	30012	P1585	Retry ADDRESSCHECK Failures
ACR	31.0.0010	32025	P1582	STATISTICS on CHECKSUM Retries
ACR	31.0.0014	32003	D2566	Additional Transaction Statist
ACR	31.0.0019	33797	D2607	New I/O Error Handling Procedu
ACR	31.0.0021	33760	D2608	Fault Handling in ACCESSROUTIN
ACR	31.0.0022	33758	D2609	Reduce Control Point Overhead
ACR	31.0.0024	33785	P1773	Read Ahead on Coarse TAbles
ACR	31.0.0034	34293	D2610	Display Core in Use for ALLOWE
ACR	31.0.0039	34304	D2611	Transaction Processing System
ACR	31.0.0040	34358	D2615	DASDL Defaults for Data Sets a
ACR	31.0.0041	34357	P1738	Statistics Midnight Overlap
ACR	31.0.0042	34354	P1742	CREATE/STORE Incorrect for Rem
ACR	31.0.0043	33759	D2619	REBLOCKING
ACR	31.0.0044	34340	D2612	Two Control Points at Audit Re
ACR	31.0.0047	34336	P1774	GENERATE NULL Bitvector
ACR	31.0.0050	34332	P1802	Audit File Switch for disk or
ACR	31.0.0052	34329	D2658	Improve Buffer Management
ACR	31.0.0053	34334	D2657	Find Correct End of Disk Type
ACR	31.0.0054	34498	P1820	Faulty Audit Constraint
ACR	31.0.0055	34493	P1821	INVALID INDEX after Audit Time
ACR	31.0.0056	34492	P1822	Block Zero of Control File Cor
ACR	31.0.0057	34490	P1907	Reset Syncpoint to Lower Value
ACR	31.0.0058	34478	P1908	Unlock Global Data and Ordered
ACR	31.0.0060	34328	P1909	Read Past EOF on Index Sequent
ACR	31.0.0061	34477	P1947	Less Resizing of Arrays
ACR	31.0.0062	34476	P1910	Corruption of Unordered
ACR	31.0.0063	34489	P1911	Remaps of Ordered Data Sets wi
ACR	31.0.0064	34471	P1912	Reorganization of Compact Data
ACR	31.0.0065	34470	P1913	Duplicate Audit Block If Switc
ACR	31.0.0066	34480	P1922	INV OP in IODISKADDRESS
ACR	31.0.0067	34468	P1914	FILEDC a Control Record
ACR	31.0.0068	34472	P1915	COPYAUDIT Not Zipped if Error
ACR	31.0.0069	34467	P1916	Two Halt/Loads Causes Recovery
ACR	31.0.0070	34458	P1917	Unaudited DB Corruption Inquir
ACR	31.0.0071	34455	P1900	Maxbuffers Statistic Incorrect
ACR	31.0.0072	34450	P1901	LOCK TO MODIFY DETAILS
ACR	31.0.0073	34440	P2009	Return Restart Areas
ACR	31.0.0074	34467	P1916	Two Halt/Loads Causes Recovery
ACR	31.0.0075	34438	P2010	Checksum Error
ACR	31.0.0077	34432	P2095	Correct Deletion in Unordered
ACR	31.0.0078	34431	P2096	DB Corruption Due to Missing E
ACR	31.0.0079	34430	P2097	DS of Recovery
ACR	31.0.0080	34427	P2098	Retry Audit Opens
ACR	31.0.0081	34422	P2127	Clear Read/Write Ahead Statist
ACR	31.0.0082	34421	P2128	Readahead Corrections
ACR	31.0.0083	34305	D2617	Visible DBS Changes and Enhanc
ACR	31.0.0084	34416	P2129	Control File Open on Abort
ACR	31.0.0085	35303	P2130	ROWLOCKOUTAUDIT Errors
ACR	31.0.0086	35304	P2131	Reconstruct Makes Empty Audit
ACR	31.0.0087	35306	P2132	Eliminate Unnecessary Displays
ACR	31.0.0089	35307	P2133	Create/Store at End of DIRECT
ACR	31.0.0091	35344	P2183	Checksum Errors
ACR	31.0.0092	35302	D2709	B7700 Optimization
ACR	31.0.0093	35340	P2184	Errorexit Timing Problem
ACR	31.0.0097	35623	P2219	KIND=DISK Vs. FAMILYNAME
ACR	31.0.0098	35443	P2244	Return of Compact Record
ACR	31.0.0099	35318	D2802	Retain or Reuse Old Structure
ACR	31.0.0100	35449	P1518	Zip COPYAUDIT at Final Close
ACR	31.0.0102	34420	D2710	CHECKSUM and I/O Retry
ACR	31.0.0103	35465	P2284	Prevent Checksum Error
ACR	31.0.0104	35456	P2360	Store of ORDERED with BLOCKSIZ
ACR	31.0.0105	35472	P2329	Invalid Count of Readers
ACR	31.0.0106	35457	P2347	Ordered Available Table Corrup
ACR	31.0.0108	35479	P2348	Data Base Hung After Close
ACR	31.0.0109	35460	P2349	ORDERED Data Set INVALID OP
ACR	31.0.0110	35458	P2350	ORDERED Data Set Improper Audi
ACR	31.0.0111	35455	P2351	ORDERED ADDRESSCHECK, CHECKSUM
ACR	31.0.0112	35459	P2352	ORDERED Data Set Loop
ACR	31.0.0113	35426	P2362	Abort Handling Mechanism
ACR	31.0.0115	35912	P2359	Resequence Data Base and RECOV
ACR	31.0.0116	35912	P2359	Resequence Data Base and RECOV
ACR	31.0.0117	35915	P2379	Designated Serial Numbers and
ACR	31.0.0118	35917	P2387	Erroneous Not Found Result
ACR	31.0.0119	35919	P2409	Eliminate Variable Format Code
ACR	31.0.0120	35922	P2410	No Abort Accept When DBS is DS
ACR	31.0.0121	35426	P2362	Abort Handling Mechanism
ACR	31.0.0122	36161	P2411	Program DS Causes Abort to Die
ACR	31.0.0123	35455	P2351	ORDERED ADDRESSCHECK, CHECKSUM
ACR	31.0.0124	35458	P2350	ORDERED Data Set Improper Audi
ACR	31.0.0127	36166	P2453	Erroneous DIRECT KEY Corrupted
ACR	31.0.0128	37092	P2454	Designated Serial Numbers
ACR	31.0.0129	37085	P2473	Incorrect Statistics Totals

PATCH TABLE

SOFTWARE	PATCH	PRI	NOTE	DESCRIPTION
ACR	31.0.0131	37084	P2495	DS DBS Vs. File Attributes
ACR	31.0.0132	37085	P2473	Incorrect Statistics Totals
ACR	31.0.0133	37071	P2496	Zero TPS Timestamp at Close
ACR	31.0.0134	37072	P2474	Free Records on Abort Exceptio
ACR	31.0.0135	37084	P2495	DS DBS Vs. File Attributes
ACR	31.0.0136	37084	P2495	DS DBS Vs. File Attributes
ACR	31.0.0137	37084	P2495	DS DBS Vs. File Attributes
ACR	31.0.0138	37345	P2497	COPYAUDIT Vs. Audit Ioerror Sw
ACR	31.0.0139	37339	P2498	INVALID INDEX Global Data and
ACR	31.0.0141	37531	P2600	Remove Overlay Point
ACR	31.0.0142	37718	P2601	Program not DSED if Audit Open
ACR	31.0.0143	37745	P2602	Initialize Partition Audited I
ACR	31.0.0144	37746	P2603	Partitioned Data Set Inaccessa
ACR	31.0.0145	38005	P2604	NOT FOUND for Partitioned Orde
ACR	31.0.0146	33797	D2607	New I/O Error Handling Procedu
ACR	31.0.0147	38025	P2605	Correction of Internal Delete
ACR	31.0.0148	38024	P2606	Audit Small Blocks for IO Erro
ACR	31.0.0149	38023	P2627	Deadlock During Update and Del
ACR	31.0.0153	38028	P2636	Missing STARTDB in Ordered Pat
ACR	31.0.0154	38041	P2637	Timing Window in OVERLAY
ACR	31.0.0157	38024	P2606	Audit Small Blocks for IO Erro
ALGOL	31.0.0095	33518	P1279	Change INFO LEVEL
ALGOL	31.0.0019	32685	D2265	Standardization of Compiler Fi
ALGOL	31.0.0020	30376	D2979	Ports and Signals
ALGOL	31.0.0021	30376	D2979	Ports and Signals
ALGOL	31.0.0030	34277	D2623	Equation Disallowed in Global
ALGOL	31.0.0031	34277	D2623	Equation Disallowed in Global
ALGOL	31.0.0036	34277	D2623	Equation Disallowed in Global
ALGOL	31.0.0037	34276	P1760	No Error on "REAL (P:P EQL "AB
ALGOL	31.0.0038	34122	P1759	Bindinfo for String Procedures
ALGOL	31.0.0039	34275	P1791	No Error After Direct I/O Stat
ALGOL	31.0.0040	34274	P1792	No Error on "TITLE=<ptr>"
ALGOL	31.0.0042	33838	D2604	Libraries
ALGOL	31.0.0043	34121	P1819	Update Pointer in REPLACE BY S
ALGOL	31.0.0045	34507	P1817	Array Reference Assignment Pro
ALGOL	31.0.0047	34576	P1878	IF FALSE in Procedures to be B
ALGOL	31.0.0048	34575	D2649	New Information in the Listing
ALGOL	31.0.0049	34574	P1879	Syntax Errors Not Being Report
ALGOL	31.0.0050	34573	P1880	AUTOBIND and One Level Codefil
ALGOL	31.0.0054	34539	P1883	Reuse String Temporaries
ALGOL	31.0.0055	34582	D2721	String Arrays, Procedures as P
ALGOL	31.0.0058	34736	D2814	Compiler Label Equation to XRE
ALGOL	31.0.0060	34734	P1953	More Descriptive Error Message
ALGOL	31.0.0061	34732	P1954	String Too Long Error Message
ALGOL	31.0.0063	34731	P1956	Correctly Handle $MCP
ALGOL	31.0.0064	34757	P1957	Erroneous Syntax Error in Vect
ALGOL	31.0.0065	34765	P1958	List Declarations
ALGOL	31.0.0066	34764	P1959	Bad GO TO From a Separate Proc
ALGOL	31.0.0067	34730	P1960	Erroneous Syntax Errors on For
ALGOL	31.0.0068	34753	P1961	External Procedures in PROCESS
ALGOL	31.0.0070	34751	P1962	Numbered CASE Statements
ALGOL	31.0.0073	34749	P1964	Stack Building Code
ALGOL	31.0.0075	34727	D2661	Header Improvement
ALGOL	31.0.0076	34790	P1966	Optimized IF Statement
ALGOL	31.0.0077	34789	P1967	Binding ALPHA6 or ALPHA7 Truth
ALGOL	31.0.0079	34786	P1968	$XREFFILES Interaction with $X
ALGOL	31.0.0080	34785	P1969	Flag Extraneous Crosshatch
ALGOL	31.0.0081	34725	P1970	Syntax Error on IF Statement
ALGOL	31.0.0082	34724	P1971	FIRSTWORD and SECONDWORD of Co
ALGOL	31.0.0083	34723	P1972	Error Limit of Zero
ALGOL	31.0.0085	34721	P1975	FOR Statement Syntax
ALGOL	31.0.0086	34738	P1974	Eliminate Compiler Error
ALGOL	31.0.0087	34767	P1976	COMPILETIME Syntax Errors
ALGOL	31.0.0088	34801	P1977	MAKEHOST and Inner Blocks
ALGOL	31.0.0089	34741	P1978	BREAKPOINT/SEPCOMP Correction
ALGOL	31.0.0090	34720	P1979	IF Expression Compatibility
ALGOL	31.0.0092	34739	P1981	STATISTICS Correction
ALGOL	31.0.0093	34718	D2662	Many Installation Intrinsics
ALGOL	31.0.0094	34742	P1982	Large Formats
ALGOL	31.0.0095	34783	P1983	Handle TASKFILE Correctly
ALGOL	31.0.0096	34781	P1984	Binder Information for a Large
ALGOL	31.0.0100	34778	P1988	Correct Xref of File Monitor D
ALGOL	31.0.0101	34714	P1989	Direct Files and Attributes
ALGOL	31.0.0102	34776	P1990	Prevent Compiler Fault
ALGOL	31.0.0104	34775	P1991	Recovering from Syntax Error
ALGOL	31.0.0105	34774	P2028	Uplevel Pointer Detection
ALGOL	31.0.0106	34773	P1992	Correct IF Statement Branches
ALGOL	31.0.0107	34712	P1993	External Procedure Parameter M
ALGOL	31.0.0108	34711	P1994	Truthsets in IF conditions
ALGOL	31.0.0109	34772	P1995	$INCLUDE Using Strings
ALGOL	31.0.0112	34770	P1996	FOR Statement with Variable St
ALGOL	31.0.0113	34769	P2029	Writing Strings in Error Messa

PATCH TABLE

SOFTWARE	PATCH	PRI	NOTE	DESCRIPTION
ALGOL	31.0.0114	34787	P1997	CTPROC Compiler Hang
ALGOL	31.0.0115	34572	D2664	STACK and CODE Listing Improve
ALGOL	31.0.0117	34571	D2677	FIRST Function for Strings
ALGOL	31.0.0119	34569	P1933	SERIALNO File Attribute
ALGOL	31.0.0123	34584	D2684	COMPLEX Implementation
ALGOL	31.0.0124	34584	D2684	COMPLEX Implementation
ALGOL	31.0.0125	34585	P2030	Erroneous Repeat Count in Valu
ALGOL	31.0.0126	34586	P2110	INVALID OP in Freefield WRITE
ALGOL	31.0.0129	34566	D2713	TOGGLE, OVERFLOW, "REAL(<point
ALGOL	31.0.0130	34398	P2112	BCL WArning Message Occurs Cor
ALGOL	31.0.0131	34396	D2712	Flag Vectormode
ALGOL	31.0.0133	34397	P2111	Warning Count Reset
ALGOL	31.0.0134	34589	P2113	INVALID OP in Freefield WRITE
ALGOL	31.0.0135	34590	P2114	INVALID OP in READ, WRITE Stat
ALGOL	31.0.0136	34566	D2713	TOGGLE, OVERFLOW, "REAL(<point
ALGOL	31.0.0137	35321	D2711	Deimplementation of Backslash
ALGOL	31.0.0139	34565	P2200	"<file id>.<file attribute>:=*
ALGOL	31.0.0140	34564	D2952	New Task, File Attributes
ALGOL	31.0.0142	34562	P1373	Storing Singles into Doubles
ALGOL	31.0.0143	35623	P2219	KIND=DISK Vs. FAMILYNAME
ALGOL	31.0.0145	35523	P1449	Small Pools for Formats
ALGOL	31.0.0147	35600	P2236	IF FALSE and $LINEINFO
ALGOL	31.0.0148	35601	P2237	Syntaxing of Format Specificat
ALGOL	31.0.0149	35602	P2238	$PAGE and the Compiler Linecou
ALGOL	31.0.0150	35603	P2239	Compiler Loop on DATABASE Decl
ALGOL	31.0.0151	35605	P2240	Remove "?" from ALPHA6
ALGOL	31.0.0152	35601	P2237	Syntaxing of Format Specificat
ALGOL	31.0.0153	35606	P2248	Exclamation Mark and Underscor
ALGOL	31.0.0154	35628	D2838	Setting Read-Only Attributes
ALGOL	31.0.0160	34564	D2952	New Task, File Attributes
ALGOL	31.0.0163	35803	P2289	No "COMPILER ERROR IN BUILDITE
ALGOL	31.0.0164	35802	P2290	No Spurious UNKNOWN DOLLAR CAR
ALGOL	31.0.0165	33838	D2604	Libraries
ALGOL	31.0.0166	33838	D2604	Libraries
ALGOL	31.0.0168	35802	P2290	No Spurious UNKNOWN DOLLAR CAR
ALGOL	31.0.0169	30376	D2979	Ports and Signals
ALGOL	31.0.0170	30376	D2979	Ports and Signals
ALGOL	31.0.0173	33464	P2439	CASE Statements with all Cases
ALGOL	31.0.0174	37297	D2919	Intrinsics with Complex and St
ALGOL	31.0.0178	33849	D2535	Revised ODT Messages
ALGOL	31.0.0180	37448	P2578	Four Digit Patch Numbers
ALGOL	31.0.0181	37447	P2579	String in DEFINE Too Long
ALGOL	31.0.0182	37444	P2587	Syntax Error with $INTRINSICS
ALGOL	31.0.0183	34520	P2594	No String Parameters by Value
ALGOL	31.0.0184	34518	P2593	Sepcomp of Procedures with Str
ALGOL	31.0.0187	34520	P2594	No String Parameters by Value
ALGOLINTRN	31.0.0001	33913	D2565	Intrinsics
ALGOLINTRN	31.0.0006	33927	D2626	DISPLAYTOSTANDARD Destination
ALGOLINTRN	31.0.0009	35251	P2140	CSQRT With Zero Imaginary Part
ALGOLINTRN	31.0.0011	35623	P2219	KIND=DISK Vs. FAMILYNAME
ALGOLPLINTRN	31.0.0001	33913	D2565	Intrinsics
ALGOLPLINTRN	31.0.0004	35623	P2219	KIND=DISK Vs. FAMILYNAME
ALGOLTABLE	31.0.0004	30376	D2979	Ports and Signals
ALGOLTABLE	31.0.0005	34277	D2623	Equation Disallowed in Global
ALGOLTABLE	31.0.0007	34304	D2611	Transaction Processing System
ALGOLTABLE	31.0.0008	33838	D2604	Libraries
ALGOLTABLE	31.0.0009	34580	P1882	Erroneous Parameter Checking I
ALGOLTABLE	31.0.0012	34571	D2677	FIRST Function for Strings
ALGOLTABLE	31.0.0014	34399	D2678	OPEN INITIALIZE Warning
ALGOLTABLE	31.0.0016	34584	D2684	COMPLEX Implementation
ALGOLTABLE	31.0.0017	34566	D2713	TOGGLE, OVERFLOW, "REAL(<point
ALGOLTABLE	31.0.0018	34396	D2712	Flag Vectormode
ALGOLTABLE	31.0.0020	35623	P2219	KIND=DISK Vs. FAMILYNAME
ALGOLTABLE	31.0.0021	35524	D2759	B7000 Series DM Optimization
ALGOLTABLE	31.0.0022	35522	P2218	Warning Message Given for Form
ALGOLTABLE	31.0.0023	35601	P2237	Syntaxing of Format Specificat
ALGOLTABLE	31.0.0024	33453	D2806	Data Dictionary
ALGOLTABLE	31.0.0026	35802	P2290	No Spurious UNKNOWN DOLLAR CAR
ALGOLTABLE	31.0.0027	33838	D2604	Libraries
ALGOLTABLE	31.0.0028	30376	D2979	Ports and Signals
ALGOLTABLE	31.0.0031	34520	P2594	No String Parameters by Value
ARCHDASDL	31.0.0002	35623	P2219	KIND=DISK Vs. FAMILYNAME
ARCHINQ	31.0.0002	35623	P2219	KIND=DISK Vs. FAMILYNAME
ARCHUPDATE	31.0.0002	35623	P2219	KIND=DISK Vs. FAMILYNAME
ATTABLEGEN	31.0.0003	33538	D2493	Accesscode
ATTABLEGEN	31.0.0004	34976	D2430	MCP Restructuring
ATTABLEGEN	31.0.0008	34564	D2952	New Task, File Attributes
ATTABLEGEN	31.0.0011	34381	D2588	IPCOVERRIDE Recognized as Task
ATTABLEGEN	31.0.0015	34564	D2952	New Task, File Attributes
ATTABLEGEN	31.0.0016	35373	D2694	Implementation of DL
ATTABLEGEN	31.0.0017	34564	D2952	New Task, File Attributes
ATTABLEGEN	31.0.0019	35623	P2219	KIND=DISK Vs. FAMILYNAME

PATCH TABLE

SOFTWARE	PATCH	PRI	NOTE	DESCRIPTION
ATTABLEGEN	31.0.0020	35500	D2864	Relative I/O for COBOL74
ATTABLEGEN	31.0.0022	35858	P2438	Library Program Dump Options
ATTABLEGEN	31.0.0030	33849	D2535	Revised ODT Messages
AUTONBBSC	31.0.0001	34414	P1839	Correct ASCII Character Define
AUTONBBSC	31.0.0002	36009	P2368	Replace $POP LIST, Allow $SET
BACKUP	31.0.0007	34237	P1761	Filenames With Special Charact
BACKUP	31.0.0008	34225	P1762	All Copies Not Printed
BACKUP	31.0.0009	34032	P1885	LP# Lost After Forms File
BACKUP	31.0.0010	34836	P2003	Backward Skip
BACKUP	31.0.0011	34809	P2004	BFILE Label Equaton
BACKUP	31.0.0014	35623	P2219	KIND=DISK Vs. FAMILYNAME
BACKUP	31.0.0015	35511	D2789	NEWP Key Specified
BACKUP	31.0.0018	37563	P2567	Buffer Reduction for LIN File
BACKUP	31.0.0019	37562	P2568	Loops When "<skip count>" Omit
BACKUP	31.0.0020	37561	P2569	REEL Attribute Not Set Before
BACKUP	31.0.0021	37479	P2570	Filenames Limited to 60 Charac
BACKUP	31.0.0022	37598	P2571	Invalid Option Combination
BACKUP	31.0.0023	37597	P2575	ODT Display Format
BACKUP	31.0.0024	37595	P2576	First Level Filename Vs. Userc
BACKUP	31.0.0025	37595	P2576	First Level Filename Vs. Userc
BACKUP	31.0.0026	37593	P2580	BACKUP Vs. System Without DISK
BASIC	31.0.0003	32685	D2265	Standardization of Compiler Fi
BASIC	31.0.0008	35623	P2219	KIND=DISK Vs. FAMILYNAME
BBSC	31.0.0001	34414	P1839	Correct ASCII Character Define
BBSC	31.0.0002	36009	P2368	Replace $POP LIST, Allow $SET
BDMSALGOL	31.0.0041	34304	D2611	Transaction Processing System
BDMSALGOL	31.0.0044	34304	D2611	Transaction Processing System
BDMSALGOL	31.0.0091	34740	P1980	STRUCTURENUMBER Correction
BDMSALGOL	31.0.0120	34399	D2678	OPEN INITIALIZE Warning
BDMSALGOL	31.0.0155	33453	D2806	Data Dictionary
BDMSALGOL	31.0.0156	33453	D2806	Data Dictionary
BDMSALGOL	31.0.0162	33453	D2806	Data Dictionary
BDMSALGOL	31.0.0167	33453	D2806	Data Dictionary
BDMSCOBOL	31.0.0078	34623	P2057	DATABASE Declaration
BDMSCOBOL	31.0.0089	34304	D2611	Transaction Processing System
BDMSCOBOL	31.0.0095	34669	P2202	Bad DMS Invoke Listing
BDMSCOBOL	31.0.0101	35776	D2848	B7000 Series DM Optimization
BDMSCOBOL	31.0.0112	35817	P2299	Bad Syntax Checking in DB Stat
BDMSCOBOL	31.0.0113	33453	D2806	Data Dictionary
BDMSCOBOL	31.0.0116	35823	P2330	Listing of the Variable Format
BDMSCOBOL	31.0.0120	35818	P2331	Declaration of Item With Same
BDMSCOBOL	31.0.0122	35821	P2332	INVALID INDEX
BDMSCOBOL	31.0.0124	33453	D2806	Data Dictionary
BDMSCOBOL	31.0.0126	35819	P2333	Inconsistent Handling of Key C
BDMSCOBOL	31.0.0130	35820	D2884	"<data-item-1> IN <key-conditi
BDMSCOBOL74	31.0.0138	34304	D2611	Transaction Processing System
BDMSCOBOL74	31.0.0162	33453	D2806	Data Dictionary
BDMSCOBOL74	31.0.0174	33453	D2806	Data Dictionary
BDMSCOBOL74	31.0.0179	35820	D2884	"<data-item-1> IN <key-conditi
BDMSPLI	31.0.0008	33443	P1771	Proper Printing of Length Attr
BDMSPLI	31.0.0018	34304	D2611	Transaction Processing System
BDMSPLI	31.0.0021	34399	D2678	OPEN INITIALIZE Warning
BDMSPLI	31.0.0033	33453	D2806	Data Dictionary
BDMSPLI	31.0.0036	35777	D2849	B7000 Series DM Optimization
BDMSPLI	31.0.0038	33453	D2806	Data Dictionary
BDMSPLI	31.0.0041	34304	D2611	Transaction Processing System
BDMSPLI	31.0.0042	34304	D2611	Transaction Processing System
BDMSPLI	31.0.0043	33453	D2806	Data Dictionary
BDMSPLI	31.0.0044	33453	D2806	Data Dictionary
BDMSPLI	31.0.0045	33453	D2806	Data Dictionary
BDMSPLI	31.0.0046	34304	D2611	Transaction Processing System
BINDER	31.0.0009	33727	P1513	Allow Many Installation Intrin
BINDER	31.0.0010	33731	P1514	Detect Large INTRINSICINFO
BINDER	31.0.0011	33730	P1515	Back Out Correctly
BINDER	31.0.0012	33735	P1516	Binding Entry Points
BINDER	31.0.0014	33732	P1517	Preserve Control State
BINDER	31.0.0015	33729	D2815	Multiple BIND= Cards
BINDER	31.0.0016	33728	P1519	Correct Identifier Recognition
BINDER	31.0.0020	33013	D2589	Binding NEWP
BINDER	31.0.0023	35112	D2722	Binding of Ports and Signals
BINDER	31.0.0024	33034	P2031	Prevent INVALID OP in Bound Pr
BINDER	31.0.0025	34708	P2032	Clear Internal Array
BINDER	31.0.0026	34673	D2714	Transaction Record Parameters
BINDER	31.0.0029	33033	P2179	Update Error Messages
BINDER	31.0.0030	35109	P2178	Proper Rebinding of String Pro
BINDER	31.0.0033	35623	P2219	KIND=DISK Vs. FAMILYNAME
BINDER	31.0.0034	35406	D2766	Internal Handling of Warnings
BINDER	31.0.0038	35411	D2770	Placement of the MCP D0 Stack
BINDER	31.0.0039	35410	D2771	Rebinding External Procedures
BINDER	31.0.0043	33453	D2806	Data Dictionary
BINDER	31.0.0044	35811	P2302	Invalid PARAMETER MISMATCH
BINDER	31.0.0046	36105	P2434	Installation Intrinsics Using

PATCH TABLE

SOFTWARE	PATCH	PRI	NOTE	DESCRIPTION
BINDER	31.0.0047	37293	P2465	Entry Binding Corrections
BINDER	31.0.0048	37291	P2492	Rebinding with FORTRAN DATA S
BUILDINQ	31.0.0004	34358	D2615	DASDL Defaults for Data Sets a
BUILDINQ	31.0.0005	34504	P1823	Enforce Answer on Update Quest
BUILDINQ	31.0.0006	34505	P1824	Data Set Selection Option Enfo
BUILDINQ	31.0.0007	34487	P1918	Deleted Logical Data Base
BUILDINQ	31.0.0008	34488	P1919	Verified Link
BUILDINQ	31.0.0009	34443	P2011	Occurring Groups
BUILDINQ	31.0.0011	34442	D2695	Sets with Group Keys
BUILDINQ	31.0.0013	35434	P1444	Automatically Selects Restart
BUILDINQ	31.0.0014	35623	P2219	KIND=DISK Vs. FAMILYNAME
BUILDINQ	31.0.0017	35302	D2709	B7700 Optimization
BUILDINQ	31.0.0018	35466	D2863	Logical Data Bases with Multip
BUILDINQ	31.0.0019	35468	P2334	Logical Data Base with "<set p
BUILDINQ	31.0.0020	35469	P2309	Vague Link and Subset Referenc
BUILDINQ	31.0.0021	35475	P2353	Invalid TABLE EXCEEDED Error
BUILDINQ	31.0.0023	35916	P2388	Embedded Sets in Logical Data
BUILDINQ	31.0.0024	35923	P2412	Maximum Size of DMINQDIRECTORY
BUILDINQ	31.0.0026	37721	P2607	Selection Using Key Data Items
BUILDINQ	31.0.0027	38030	P2638	Acceptance of Queue Specificat
BUILDINQ	31.0.0028	38030	P2638	Acceptance of Queue Specificat
BUILDREORG	31.0.0004	35347	P2185	Reporting of ORDERED BY for In
BUILDREORG	31.0.0005	35336	D2731	REORGANIZATION Optimization
BUILDREORG	31.0.0006	35429	D2754	Sort Environment Syntax
BUILDREORG	31.0.0008	35623	P2219	KIND=DISK Vs. FAMILYNAME
BUILDREORG	31.0.0009	35435	D2775	REORGANIZATION Limitations
BUILDREORG	31.0.0010	35436	P1465	Sequencing of Prime Index, Def
BUILDREORG	31.0.0011	35440	D2803	Automatic Checksum of Intermed
BUILDREORG	31.0.0013	37083	P2477	INVALID INDEX at 30139840
BUILDREORG	31.0.0015	37533	D2591	Checking BUILDREORG Status in
CANDE	31.0.0013	32685	D2265	Standardization of Compiler Fi
CANDE	31.0.0016	35325	D2740	Accesscode
CANDE	31.0.0018	34322	D2590	III.1 Controller Keyins
CANDE	31.0.0019	33605	D2634	SUBSYSTEM Task Attribute
CANDE	31.0.0020	33604	D2636	"Q-DS" Message for START Job
CANDE	31.0.0021	34900	D2665	LSN Ranges
CANDE	31.0.0022	34899	D2666	OPTIONS COMMANDS
CANDE	31.0.0023	34898	D2667	CONTROL Command Extensions
CANDE	31.0.0025	34896	D2668	WHAT Next Extensions
CANDE	31.0.0026	34895	P1998	Hung Swap Job
CANDE	31.0.0028	34880	P1999	Schedule Fault Corrected
CANDE	31.0.0029	34881	D2670	Multiple Log Stations
CANDE	31.0.0032	34884	P2000	MATCH Verb Correction
CANDE	31.0.0033	34886	D2672	SCHD Message
CANDE	31.0.0034	34887	D2673	PUBLIC Abbreviation
CANDE	31.0.0035	34888	D2674	Data Files Update
CANDE	31.0.0036	33603	D2675	Resequence of CANDE
CANDE	31.0.0038	33602	D2679	WHO Control Command
CANDE	31.0.0040	35623	P2219	KIND=DISK Vs. FAMILYNAME
CANDE	31.0.0041	33601	D2827	"?AT" Controller Keyin
CANDE	31.0.0043	35373	D2694	Implementation of DL
CANDE	31.0.0044	34881	D2670	Multiple Log Stations
CANDE	31.0.0045	35727	P2292	REPLACE in Sequence Number Fie
CANDE	31.0.0046	35728	P2293	REPLACE on ID Field
CANDE	31.0.0047	35729	P2294	INSERT AT I ID on Empty Workfil
CANDE	31.0.0055	36022	D2916	Add HOSTNAM, as Task Modifier
CANDE	31.0.0056	35325	D2740	Accesscode
CANDE	31.0.0057	35325	D2740	Accesscode
CANDE	31.0.0059	37484	P2553	VALUE Sign on EXECUTE
CANDE	31.0.0062	34322	D2590	III.1 Controller Keyins
CANDE	31.0.0063	34322	D2590	III.1 Controller Keyins
CARDLINE	31.0.0002	34018	P1886	Job Cards Transferred to Disk
CARDLINE	31.0.0004	35623	P2219	KIND=DISK Vs. FAMILYNAME
CCTABLEGEN	31.0.0002	34063	D2627	Accesscode
CCTABLEGEN	31.0.0010	34912	D2752	New Compiler Names
CCTABLEGEN	31.0.0012	35623	P2219	KIND=DISK Vs. FAMILYNAME
COBOL	31.0.0012	33679	P1434	Program Collating Sequence
COBOL	31.0.0017	32685	D2265	Standardization of Compiler Fi
COBOL	31.0.0035	34145	P1662	Exceptioncode for ANSI74 Corre
COBOL	31.0.0036	34129	D2676	Inspect Algorithm Meets ANSI74
COBOL	31.0.0037	34130	P1663	Scale Non-Integer Moved to DM
COBOL	31.0.0038	34131	P1664	Graceful Exit from Syntaxed ST
COBOL	31.0.0039	30376	D2979	Ports and Signals
COBOL	31.0.0040	34132	P1666	FILE-LIMIT Statement Caused IN
COBOL	31.0.0041	34143	P1763	User Intrinsic Calls Not Scann
COBOL	31.0.0043	34134	D2850	B7700 AUDIT Feature
COBOL	31.0.0047	34654	P2033	Spurious Error Messages
COBOL	31.0.0048	34653	P2253	SELECT Statement Syntax Errors
COBOL	31.0.0049	34652	D2816	NOTE Verb Extensions
COBOL	31.0.0050	34651	P2035	Compile Time Information
COBOL	31.0.0051	34650	P2036	Hex Literal Compares
COBOL	31.0.0052	34649	P2037	E Format Scanner Error

PATCH TABLE

SOFTWARE	PATCH	PRI	NOTE	DESCRIPTION
COBOL	31.0.0053	34648	P2038	Comparison of Index Data Names
COBOL	31.0.0054	34647	P2039	VALUE Clause
COBOL	31.0.0055	34646	P2040	Improved String Comparisons
COBOL	31.0.0056	34645	P2041	E Format on Continuation Card
COBOL	31.0.0057	34644	D2683	Dollar Option BINARYCOMP
COBOL	31.0.0059	34642	P2042	Dictionary Wraparound
COBOL	31.0.0060	34641	P2043	Declarative Execution
COBOL	31.0.0061	34640	P2044	Hex Literal Figuratives
COBOL	31.0.0062	34639	P2045	Simple Literal Compares
COBOL	31.0.0063	34638	P2046	Correct Stack Estimate
COBOL	31.0.0064	34637	P2047	Improved Error Message
COBOL	31.0.0065	34636	P2177	COPY in LD Section
COBOL	31.0.0066	34635	P2048	Core Estimate of Sorting Progr
COBOL	31.0.0067	34634	P2049	PICTURE Clause
COBOL	31.0.0068	34633	P2050	Mixed Group Moves
COBOL	31.0.0069	34632	P2051	VALUE Clause for COMP-4 Items
COBOL	31.0.0070	34631	P2052	Warning for Varying Size Recor
COBOL	31.0.0072	34629	P2053	COPY Statement Listing
COBOL	31.0.0075	34626	P2054	Correct Warning Message
COBOL	31.0.0076	34624	P2055	Report Writer Control Footing
COBOL	31.0.0077	34625	P2056	Correct Offset Listing
COBOL	31.0.0079	34622	P2058	Negative Valued Attribute Mnem
COBOL	31.0.0080	34621	P2059	XREF
COBOL	31.0.0081	34619	P2060	Two Dimensional Subscripting
COBOL	31.0.0082	34618	P2061	CLOSE with PURGE
COBOL	31.0.0084	34620	P2062	SELECTION Expressions
COBOL	31.0.0086	34677	P2063	Word Aligned Moves
COBOL	31.0.0087	34676	P2161	OPTIMIZE Option
COBOL	31.0.0090	34674	D2839	Library Facility Implemented
COBOL	31.0.0092	34129	D2676	Inspect Algorithm Meets ANSI74
COBOL	31.0.0094	34670	P2187	TIME Functions
COBOL	31.0.0097	34668	P2252	Statistics Not Correct for STO
COBOL	31.0.0098	35623	P2219	KIND=DISK Vs. FAMILYNAME
COBOL	31.0.0099	34664	P2254	External PROCEDURE Declaration
COBOL	31.0.0100	34674	D2839	Library Facility Implemented
COBOL	31.0.0102	34663	P2255	UNITS Attribute
COBOL	31.0.0103	35769	P2256	Code Segment Size Exceeded
COBOL	31.0.0104	35770	P2257	Too Many REPORT Writer Code Cl
COBOL	31.0.0105	35771	P2258	INVALID INDEX in Bad REPORT Wr
COBOL	31.0.0108	34674	D2839	Library Facility Implemented
COBOL	31.0.0109	35768	P2295	Exception Code Erroneously Che
COBOL	31.0.0110	35767	P2259	ANSI74 LINAGE WRITE AT END OF
COBOL	31.0.0111	35765	P2260	Random File Getting Serial Wri
COBOL	31.0.0114	35763	P2313	Incorrect SORT Statement
COBOL	31.0.0115	35752	P2314	LOCK Statements
COBOL	31.0.0117	35755	P2315	INVALID INDEX in Large Code Se
COBOL	31.0.0118	35762	P2316	Maximum Files Allowed
COBOL	31.0.0119	35756	P2317	Info Table Overflow
COBOL	31.0.0121	35757	P2318	XREF
COBOL	31.0.0123	35758	P2319	Condition Names
COBOL	31.0.0125	35759	P2320	Moves of Numeric Literals
COBOL	31.0.0127	35934	P2338	Large Report Writer Exceeded C
COBOL	31.0.0128	35933	P2339	Analyze Option Duplicating Lin
COBOL	31.0.0129	35760	P2340	Negative Zeros in VALUE Clause
COBOL	31.0.0131	35932	P2365	Large Record Descriptions
COBOL	31.0.0132	35822	P2341	COBOL No Longer Ignores Mispla
COBOL	31.0.0133	35946	P2342	SEARCH Nested in IF Statement
COBOL	31.0.0134	35947	D2874	OCCURS DEPENDING Option
COBOL	31.0.0135	35949	P2373	Condition Names
COBOL	31.0.0136	35824	P2377	INSPECT Verb Slow Execution
COBOL	31.0.0138	35968	P2380	SORT Dynamic File Attributes
COBOL	31.0.0139	35948	P2381	Erroneous Syntax Error
COBOL	31.0.0140	35965	D2921	"DATE-COMPILED" Clause Has Per
COBOL	31.0.0141	35966	D2922	Exception Code for ANSI74 Inde
COBOL	31.0.0142	34674	D2839	Library Facility Implemented
COBOL	31.0.0143	35967	D2923	MONITOR Output
COBOL	31.0.0145	34674	D2839	Library Facility Implemented
COBOL	31.0.0146	34674	D2839	Library Facility Implemented
COBOLTABLE	31.0.0002	30376	D2979	Ports and Signals
COBOLTABLE	31.0.0003	34136	P1811	LISTACK Option
COBOLTABLE	31.0.0004	34134	D2850	B7700 AUDIT Feature
COBOLTABLE	31.0.0007	34304	D2611	Transaction Processing System
COBOLTABLE	31.0.0008	34674	D2839	Library Facility Implemented
COBOLTABLE	31.0.0010	35623	P2219	KIND=DISK Vs. FAMILYNAME
COBOLTABLE74	31.0.0005	34304	D2611	Transaction Processing System
COBOLTABLE74	31.0.0006	34674	D2839	Library Facility Implemented
COBOLTABLE74	31.0.0008	35623	P2219	KIND=DISK Vs. FAMILYNAME
COBOLTABLE74	31.0.0009	30376	D2979	Ports and Signals
COBOL74	31.0.0097	34146	P1661	Task to Task String Attribute
COBOL74	31.0.0098	34130	P1663	Scale Non-Integer Moved to DM
COBOL74	31.0.0099	34131	P1664	Graceful Exit from Syntaxed ST
COBOL74	31.0.0100	34132	P1666	FILE-LIMIT Statement Caused IN

PATCH TABLE

SOFTWARE	PATCH	PRI	NOTE	DESCRIPTION
COBOL74	31.0.0139	34674	D2839	Library Facility Implemented
COBOL74	31.0.0149	34668	P2252	Statistics Not Correct for STO
COBOL74	31.0.0150	35623	P2219	KIND=DISK Vs. FAMILYNAME
COBOL74	31.0.0160	34674	D2839	Library Facility Implemented
COBOL74	31.0.0171	30376	D2979	Ports and Signals
COBOL74	31.0.0172	30376	D2979	Ports and Signals
COBOL74	31.0.0183	35947	D2874	OCCURS DEPENDING Option
COBOL74	31.0.0190	34674	D2839	Library Facility Implemented
COBOL74	31.0.0195	34674	D2839	Library Facility Implemented
COMPARE	31.0.0003	34019	P1887	Usercode Attached File Titles
COMPARE	31.0.0005	35623	P2219	KIND=DISK Vs. FAMILYNAME
CONFIGURATOR	31.0.0002	35623	P2219	KIND=DISK Vs. FAMILYNAME
CONFIGURATOR	31.0.0004	35985	D2867	Soft Configuration of Global M
CONFIGURATOR	31.0.0007	35985	D2867	Soft Configuration of Global M
CONFIGURATOR	31.0.0008	35985	D2867	Soft Configuration of Global M
CONFIGURATOR	31.0.0009	35985	D2867	Soft Configuration of Global M
CONFIGURATOR	31.0.0011	35985	D2867	Soft Configuration of Global M
CONTROLLER	31.0.0006	35987	D2861	B6800 Multiprocessor Systems
CONTROLLER	31.0.0008	32463	P9150	FILEKIND
CONTROLLER	31.0.0009	32493	P9217	SQ NO ENTRIES
CONTROLLER	31.0.0011	32636	D2267	Four vs Three Digit System Ser
CONTROLLER	31.0.0012	32604	D2404	New Display Options
CONTROLLER	31.0.0015	32591	P1073	ADMEVENT M
CONTROLLER	31.0.0017	32575	P1077	PC Vs. MPX3
CONTROLLER	31.0.0021	33252	D2463	RESOURCECHECK
CONTROLLER	31.0.0026	33538	D2493	Accesscode
CONTROLLER	31.0.0027	34976	D2430	MCP Restructuring
CONTROLLER	31.0.0032	33849	D2535	Revised ODT Messages
CONTROLLER	31.0.0034	33319	P1705	REMOTESPO
CONTROLLER	31.0.0035	33849	D2535	Revised ODT Messages
CONTROLLER	31.0.0036	34063	D2627	Accesscode
CONTROLLER	31.0.0037	34057	P1706	NEXT on RJE Peripheral Status
CONTROLLER	31.0.0038	33849	D2535	Revised ODT Messages
CONTROLLER	31.0.0039	33838	D2604	Libraries
CONTROLLER	31.0.0040	34263	D2594	Remove DCKEYIN Restrictions
CONTROLLER	31.0.0042	33849	D2535	Revised ODT Messages
CONTROLLER	31.0.0046	34213	P1765	SQ Command Correction
CONTROLLER	31.0.0047	34210	D2592	Time and Date Verification
CONTROLLER	31.0.0049	34200	P1800	WAITLIMIT
CONTROLLER	31.0.0051	30499	D2637	Reason for Queue DS
CONTROLLER	31.0.0052	33849	D2535	Revised ODT Messages
CONTROLLER	31.0.0053	33849	D2535	Revised ODT Messages
CONTROLLER	31.0.0054	33849	D2535	Revised ODT Messages
CONTROLLER	31.0.0057	33849	D2535	Revised ODT Messages
CONTROLLER	31.0.0058	33849	D2535	Revised ODT Messages
CONTROLLER	31.0.0060	33849	D2535	Revised ODT Messages
CONTROLLER	31.0.0061	34968	P2105	DUP FAMILY
CONTROLLER	31.0.0064	33849	D2535	Revised ODT Messages
CONTROLLER	31.0.0066	33849	D2535	Revised ODT Messages
CONTROLLER	31.0.0067	33849	D2535	Revised ODT Messages
CONTROLLER	31.0.0068	33849	D2535	Revised ODT Messages
CONTROLLER	31.0.0070	33849	D2535	Revised ODT Messages
CONTROLLER	31.0.0073	35373	D2694	Implementation of DL
CONTROLLER	31.0.0074	35623	P2219	KIND=DISK Vs. FAMILYNAME
CONTROLLER	31.0.0083	35736	P2308	CONT6 GETDISK
CONTROLLER	31.0.0085	33849	D2535	Revised ODT Messages
CONTROLLER	31.0.0086	35985	D2867	Soft Configuration of Global M
CONTROLLER	31.0.0089	33849	D2535	Revised ODT Messages
CONTROLLER	31.0.0090	33849	D2535	Revised ODT Messages
CONTROLLER	31.0.0091	36102	P2400	Correct MQ- 1023
CONTROLLER	31.0.0093	33849	D2535	Revised ODT Messages
CONTROLLER	31.0.0094	36054	P2399	Message Greater Than 1896 Char
CONTROLLER	31.0.0095	36070	P2440	SCR Length
CONTROLLER	31.0.0096	33849	D2535	Revised ODT Messages
CONTROLLER	31.0.0097	33849	D2535	Revised ODT Messages
CONTROLLER	31.0.0098	33849	D2535	Revised ODT Messages
CONTROLLER	31.0.0099	36081	P2466	Mnemonic Improvement
CONTROLLER	31.0.0101	37270	P2467	Privileged MCS
CONTROLLER	31.0.0102	36084	P2489	PRINTLABEL
CONTROLLER	31.0.0103	35985	D2867	Soft Configuration of Global M
CONTROLLER	31.0.0105	33849	D2535	Revised ODT Messages
CONTROLLER	31.0.0106	37246	P2541	CU Vs. HARDCOPY
CONTROLLER	31.0.0108	33849	D2535	Revised ODT Messages
CONTROLLER	31.0.0110	37496	P2525	ODT Timout
CONTROLLER	31.0.0111	37485	D2940	Leading Blanks in ACCEPT Messa
CONTROLLER	31.0.0118	37665	D2953	New COMPILERINFO Format
COPYAUD-II	31.0.0003	34464	P1548	Correct Block 0 Size Error Mes
COPYAUD-II	31.0.0004	35623	P2219	KIND=DISK Vs. FAMILYNAME
COPYAUD-II	31.0.0005	35441	P2217	Block 0 Checksum
COPYAUD-II	31.0.0007	37080	P2478	Close New Audit Before Old
DASDL	31.0.0012	33123	P1840	Improve DASDL Scanner
DASDL	31.0.0013	33118	D2568	Allow More Structures, Items P

PATCH TABLE

SOFTWARE	PATCH	PRI	NOTE	DESCRIPTION
DASDL	31.0.0014	33119	D2567	VERSION Compiler Option
DASDL	31.0.0017	32015	D2613	Optimize CLEARDATA Text
DASDL	31.0.0018	32007	D2614	List TAPE and NEWTAPE File Tit
DASDL	31.0.0021	34301	P1775	Eliminate SEG ARRAY Error
DASDL	31.0.0022	34358	D2615	DASDL Defaults for Data Sets a
DASDL	31.0.0023	34352	P1776	Recompilation Message for Upda
DASDL	31.0.0024	34341	D2616	INITIALIZE Statement Zips UTIL
DASDL	31.0.0025	34339	P1777	Prevent Abnormal Termination
DASDL	31.0.0026	33759	D2619	REBLOCKING
DASDL	31.0.0027	34331	P1803	Prevent INVALID INDEX
DASDL	31.0.0028	34330	P1804	Ensure AREASIZE is Specified P
DASDL	31.0.0029	34481	P1920	Valid OPEN PARTITIONS Attribut
DASDL	31.0.0030	34461	P1921	Global Chain Not Overlayed
DASDL	31.0.0031	34456	D2682	Control File Title on Update
DASDL	31.0.0032	35318	D2802	Retain or Reuse Old Structure
DASDL	31.0.0033	35302	D2709	B7700 Optimization
DASDL	31.0.0034	35320	D2729	Add Subsystem ID Dollar Card
DASDL	31.0.0035	35335	D2732	Elimination of REORGPENDING
DASDL	31.0.0036	35348	P2191	Erroneous Key Changed Errors
DASDL	31.0.0037	35316	D2730	Changes to READAHEAD, Buffers
DASDL	31.0.0038	35332	D2751	Zip Standard WFL File
DASDL	31.0.0039	35431	P2207	INFO Table Incorrect
DASDL	31.0.0041	35623	P2219	KIND=DISK Vs. FAMILYNAME
DASDL	31.0.0042	35446	P1038	INVALID INDEX
DASDL	31.0.0043	35447	P2034	Syntax Error for One Global Da
DASDL	31.0.0045	35302	D2709	B7700 Optimization
DASDL	31.0.0046	35473	D2889	TPSDUALUPDATE Option Added
DASDL	31.0.0047	35476	P2354	Text Generated for Group Key w
DASDL	31.0.0048	35477	P2355	Last Scan Saved
DASDL	31.0.0049	35480	P2356	Corrects Error Checking for RE
DASDL	31.0.0051	36172	P2455	Print Store Card Image
DASDL	31.0.0052	36172	P2455	Print Store Card Image
DASDL	31.0.0053	36180	P2456	Allow ALTERNATE Specs after PA
DASDL	31.0.0054	37091	P2457	Always Compile PARTITIONCONTRO
DASDL	31.0.0055	37090	P2479	Maximum Length DBNAME In Quote
DASDL	31.0.0056	37087	P2480	Disallow Data Set with No Name
DASDL	31.0.0059	37448	P2578	Four Digit Patch Numbers
DASDL	31.0.0060	37537	P2608	Warning for ACRINFO, ACRDISPLA
DASDL	31.0.0062.	37535	P2609	DECKLIST Option
DASDL	31.0.0063	37731	P2610	Write Omitted Cards to Listing
DASDL	31.0.0064	38029	P2639	Increase TEXTGENMAX to 4095
DATACOM	31.0.0060	31593	P9144	ADD STATION TO FILE DCWRITE
DATACOM	31.0.0075	32813	P9157	Inserting Null Message in Inac
DATACOM	31.0.0526	34952	P2106	Segmented Array Disk I/O
DATACOM	31.0.0527	34951	P2107	Recontimeout Failure
DATACOM	31.0.0528	32568	P2108	DCCONTROL Invalid Index
DATACOM	31.0.0529	32570	P2109	Set Up Line Control Index
DATACOM	31.0.0530	35215	D2703	Implement Application Number V
DATACOM	31.0.0821	36018	P2397	Restarting DSed DCP
DATACOM	31.0.0822	36044	D2903	Station Interrogate Dcwrite Ex
DATACOM	31.0.0855	37021	P2450	DCP Termination
DATACOM	31.0.0979	37028	P2583	Subtract Station Fault with No
DATACOM	31.0.1011	37029	P2596	Not Ready Lines After Cluster
DATACOM	31.0.1014	37759	P2622	Set "ICANWAITF" in DCINSERT GE
DATACOM	31.0.1015	37760	P2623	DCTANKING Error Handling Corre
DCALGOL	31.0.0053	34580	P1882	Erroneous Parameter Checking I
DCALGOL	31.0.0056	34580	P1882	Erroneous Parameter Checking I
DCALGOLINT	31.0.0002	33913	D2565	Intrinsics
DCALGOLINT	31.0.0004	34604	P1888	Correct DCP Sleep Address Inte
DCPPROGEN	31.0.0002	35215	D2703	Implement Application Number V
DCPPROGEN	31.0.0003	32568	P2108	DCCONTROL Invalid Index
DCPPROGEN	31.0.0005	34615	P2201	OR and XOR Code Generation
DCPPROGEN	31.0.0007	35535	D2772	Implement Variable Size Statio
DCPPROGEN	31.0.0008	35539	P2325	Further Auxiliary Logic Correc
DCPPROGEN	31.0.0009	36016	P2402	Full Duplex Termination
DCPPROGEN	31.0.0010	36015	P2401	Extend Labels for 20 Bit Addre
DCPPROGEN	31.0.0013	36019	P2441	LOSSOFCARRIER=DISCONNECT Vs. S
DCPPROGEN	31.0.0014	37023	P2491	Correct Code Listing for 20-Bi
DCPPROGEN	31.0.0015	37024	P2493	TERMINATE OUTPUTREQUEST Label
DCPPROGEN	31.0.0016	36015	P2401	Extend Labels for 20 Bit Addre
DCPPROGEN	31.0.0017	37026	P2554	Save and Use ADDCLUSTERS Varia
DCPTESTGEN	31.0.0003	35623	P2219	KIND=DISK Vs. FAMILYNAME
DCPTESTGEN	31.0.0005	36008	P2366	MAKE NEWTAPE FILEKIND=DCPSYMBO
DCSTATUS	31.0.0002	32571	D2696	Extensions
DCSTATUS	31.0.0004	35623	P2219	KIND=DISK Vs. FAMILYNAME
DDDASDL	31.0.0001	33453	D2806	Data Dictionary
DDDASDL	31.0.0008	37748	P2613	Maintain Level Number and Occu
DDINITIAL	31.0.0001	33453	D2806	Data Dictionary
DDUPDATE	31.0.0001	33453	D2806	Data Dictionary
DDUPDATE	31.0.0002	33453	D2806	Data Dictionary
DDUPDATE	31.0.0003	33453	D2806	Data Dictionary
DDUPDATE	31.0.0004	33453	D2806	Data Dictionary

PATCH TABLE

SOFTWARE	PATCH	PRI	NOTE	DESCRIPTION
DDUPDATE	31.0.0005	33453	D2806	Data Dictionary
DDUPDATE	31.0.0006	33453	D2806	Data Dictionary
DDUPDATE	31.0.0007	33453	D2806	Data Dictionary
DDUPDATE	31.0.0008	33453	D2806	Data Dictionary
DDUPDATE	31.0.0009	33453	D2806	Data Dictionary
DDUPDATE	31.0.0010	33453	D2806	Data Dictionary
DDUPDATE	31.0.0016	33453	D2806	Data Dictionary
DDUPDATE	31.0.0021	37748	P2613	Maintain Level Number and Occu
DDUPDATE	31.0.0022	37747	P2614	Text Files Without Valid Heade
DIAGNOSTMCS	31.0.0005	35623	P2219	KIND=DISK Vs. FAMILYNAME
DMALGOL	31.0.0138	34417	P2115	Compile Time Processor INVALID
DMALGOL	31.0.0144	35524	D2759	B7000 Series DM Optimization
DMALGOL	31.0.0146	35524	D2759	B7000 Series DM Optimization
DMCTL	31.0.0001	33759	D2619	REBLOCKING
DMCTL	31.0.0003	34304	D2611	Transaction Processing System
DMCTL	31.0.0004	34305	D2617	Visible DBS Changes and Enhanc
DMCTL	31.0.0006	34359	P1739	OPEN INITIALIZE Vs. RECOVERY
DMCTL	31.0.0007	34480	P1922	INV OP in IODISKADDRESS
DMCTL	31.0.0008	34419	D2708	Dump Tape Directory
DMCTL	31.0.0009	35318	D2802	Retain or Reuse Old Structure
DMCTL	31.0.0010	35335	D2732	Elimination of REORGPENDING
DMCTL	31.0.0012	35623	P2219	KIND=DISK Vs. FAMILYNAME
DMCTL	31.0.0013	35302	D2709	B7700 Optimization
DMCTL	31.0.0015	35426	P2362	Abort Handling Mechanism
DMCTL	31.0.0016	37084	P2495	DS DBS Vs. File Attributes
DMCTL	31.0.0020	37084	P2495	DS DBS Vs. File Attributes
DOCUMENTOR	31.0.0001	34543	P1832	Update Document Printer for GM
DOCUMENTOR	31.0.0002	34544	P1833	Modified Mark # in Directory P
DOCUMENTOR	31.0.0003	35114	P2069	Remove PROGRAMDOC From Exclude
DOCUMENTOR	31.0.0004	37994	P2629	Miscellaneous Changes
DUMPALL	31.0.0002	34254	P1766	UL OPtion Correction
DUMPALL	31.0.0003	34253	P1767	ROUTINE and COPY Statements Re
DUMPALL	31.0.0004	34251	P1768	DMPMT Short Records
DUMPALL	31.0.0005	34250	P1769	Equal Sign in File Title
DUMPALL	31.0.0008	35623	P2219	KIND=DISK Vs. FAMILYNAME
DUMPALL	31.0.0009	35805	P2261	INTMODE of HEX or SINGLE
DUMPALL	31.0.0010	35873	P2343	Unallocated Rows
DUMPALL	31.0.0012	36028	P2442	Loop in DMPMT if File Found on
DUMPALL	31.0.0013	36029	P2443	No File when Equating Tape to
DUMPALL	31.0.0014	37259	P2529	Error Messages
DUMPALL	31.0.0015	37258	P2528	NEWFILE Vs. Old File
DUMPANALY	31.0.0009	35987	D2861	B6800 Multiprocessor Systems
DUMPANALY	31.0.0010	30369	P9249	Correct "CANNOT ANALYZE" Messa
DUMPANALY	31.0.0011	32636	D2267	Four vs Three Digit System Ser
DUMPANALY	31.0.0012	30370	P9248	Standard Mode From the Termina
DUMPANALY	31.0.0013	32857	P9272	FIBLOCKSNR
DUMPANALY	31.0.0014	30372	P9277	Supply Information on Incompat
DUMPANALY	31.0.0015	33081	P1109	Hard Lock and RCW Analysis
DUMPANALY	31.0.0016	33080	P1110	MCP Title from Header
DUMPANALY	31.0.0017	35987	D2861	B6800 Multiprocessor Systems
DUMPANALY	31.0.0018	30374	D2451	New IOCB Word, Remote Backup F
DUMPANALY	31.0.0019	30411	D2453	HELP Command Improvements
DUMPANALY	31.0.0021	30375	P1213	SEARCH Command Acceleration
DUMPANALY	31.0.0022	33208	P1603	Extract IOCBs from Buffers
DUMPANALY	31.0.0028	33087	P1226	EVENT Mechanism Changes
DUMPANALY	31.0.0029	33946	P1520	IO Deadlock Analysis
DUMPANALY	31.0.0030	30378	D2551	STANDARD Command
DUMPANALY	31.0.0032	30376	D2979	Ports and Signals
DUMPANALY	31.0.0034	30381	P1707	Unknown Run Time Options in MC
DUMPANALY	31.0.0035	30382	P1708	MOD 63 Gives False Tape Error
DUMPANALY	31.0.0036	33838	D2604	Libraries
DUMPANALY	31.0.0037	34106	D2593	STACK Bounds
DUMPANALY	31.0.0038	30376	D2979	Ports and Signals
DUMPANALY	31.0.0039	30388	P1793	Miscellaneous Corrections
DUMPANALY	31.0.0040	30498	D2635	Port and Signal Analysis
DUMPANALY	31.0.0043	30410	P1893	SIRW Analysis
DUMPANALY	31.0.0045	30390	P2083	Sequence Numbers from RCW
DUMPANALY	31.0.0046	30391	P2084	Fault Handling
DUMPANALY	31.0.0047	33849	D2535	Revised ODT Messages
DUMPANALY	31.0.0048	30502	D2720	Network Analysis
DUMPANALY	31.0.0051	33849	D2535	Revised ODT Messages
DUMPANALY	31.0.0054	35623	P2219	KIND=DISK Vs. FAMILYNAME
DUMPANALY	31.0.0058	35614	D2845	MCP Name Handling
DUMPANALY	31.0.0059	35615	D2846	Area Dump Headings
DUMPANALY	31.0.0064	35842	P2321	Close Interactive OPTIONS File
DUMPANALY	31.0.0067	35891	P2358	OVERLAYCF
DUMPANALY	31.0.0068	35850	D2883	Count Logical I/O Requests
DUMPANALY	31.0.0070	36109	P2423	Memory Priority for Swapjobs
DUMPANALY	31.0.0071	37110	P2432	Cope with FIB Index Array Not
DUMPANALY	31.0.0076	36117	P2488	Automatic Shortening of SEARCH
DUMPANALY	31.0.0077	37120	D2944	Interactive Mode
DUMPANALY	31.0.0078	37511	D2945	IO Unit Selection

PATCH TABLE

SOFTWARE	PATCH	PRI	NOTE	DESCRIPTION
DUMPANALY	31.0.0079	37122	P2540	Improved Initialization and Fi
DUMPANALY	31.0.0080	37512	P2523	Special Arrays
DUMPANALY	31.0.0081	37513	P2524	Remote Line Width
DUMPANALY	31.0.0082	37514	D2939	Code Dump
DUMPANALY	31.0.0083	37515	P2520	Save File AREASIZE
DUMPANALY	31.0.0084	37516	P2521	Break on Output
DUMPANALY	31.0.0086	37519	P2515	DCC and DCP Station Info
DUMPANALY	31.0.0088	37520	P2565	"TAG 4" and "TAG 6" Analysis
DUMPANALY	31.0.0091	37714	P2581	PV and Stackdump Operands
DUMPANALY	31.0.0092	37713	D2950	Locks, Events, DEADLOCK Analys
DUMPANALY	31.0.0093	37712	D2949	Descriptor Analysis
DUMPANALY	31.0.0094	37665	D2953	New COMPILERINFO Format
DUMPANALY	31.0.0096	37972	D2966	NOIO AND UINFO Options
DUMPDIR	31.0.0002	34419	D2708	Dump Tape Directory
DUMPDIRLIB	31.0.0002	34419	D2708	Dump Tape Directory
DUMPDIRLIB	31.0.0003	34419	D2708	Dump Tape Directory
DUMPDIRLIB	31.0.0004	34419	D2708	Dump Tape Directory
DUMPDIRLIB	31.0.0007	37539	P2582	Location for Length of Dump Di
DUMPDIRLIB	31.0.0009	34419	D2708	Dump Tape Directory
DUMPDIRLIB	31.0.0010	34419	D2708	Dump Tape Directory
DUMPDIRLIB	31.0.0011	34419	D2708	Dump Tape Directory
ESPOL	31.0.0017	35623	P2219	KIND=DISK Vs. FAMILYNAME
ESPOLINTRN	31.0.0001	32078	P9285	Task Restructuring
ESPOLINTRN	31.0.0012	33913	D2565	Intrinsics
ESPOLINTRN	31.0.0024	34263	D2594	Remove DCKEYIN Restrictions
ESPOLINTRN	31.0.0026	34140	D2595	Compiling ESPOL Intrinsics
ESPOLINTRN	31.0.0030	34135	P1841	BASIC Binary Files
ESPOLINTRN	31.0.0031	34583	D2680	Freefield Output for Complex V
ESPOLINTRN	31.0.0034	35232	P2141	Correct R Formats
ESPOLINTRN	31.0.0035	35233	P2142	Eliminate Loop on Invalid Char
ESPOLINTRN	31.0.0036	35236	P2143	Determine Length of Character
ESPOLINTRN	31.0.0037	35237	P2144	Correct Invalid Index
ESPOLINTRN	31.0.0038	35238	P2145	BASIC, Binary Files
ESPOLINTRN	31.0.0039	35239	P2146	MONITOR Double Variable
ESPOLINTRN	31.0.0040	35240	P2147	Correct E Format
ESPOLINTRN	31.0.0041	35241	P2148	BASIC, Correct Buffer Flushing
ESPOLINTRN	31.0.0042	35242	P2149	Correct Numeric Replace
ESPOLINTRN	31.0.0043	35243	P2150	Prevent Extra Printing
ESPOLINTRN	31.0.0044	35244	P2151	BASIC, PRINTUSING
ESPOLINTRN	31.0.0045	35245	P2152	BASIC, Empty Files
ESPOLINTRN	31.0.0046	35246	P2153	Task DS
ESPOLINTRN	31.0.0047	35247	P2154	Monitor Double Arrays
ESPOLINTRN	31.0.0048	35248	P2155	Scale Format Phrase
ESPOLINTRN	31.0.0051	35252	P2156	Free Field Output
ESPOLINTRN	31.0.0052	35253	P2157	Binary Write Question Marks
ESPOLINTRN	31.0.0053	35254	D2817	BASIC, Blocking of Disk Files
ESPOLINTRN	31.0.0054	35255	D2818	BASIC, File Handling Changes
ESPOLINTRN	31.0.0055	35256	P2160	BASIC, Resize Format Buffer
ESPOLINTRN	31.0.0056	35257	D2819	New Display of Format Errors
ESPOLINTRN	31.0.0057	35258	P2162	BASIC, String Comparisons
ESPOLINTRN	31.0.0060	34659	P1453	STATISTICS Option
ESPOLINTRN	31.0.0063	35953	P2542	Formatting Intrinsics
ESPOLINTRN	31.0.0064	37400	P2543	GMM Routines
ESPOLINTRN	31.0.0066	35255	D2818	BASIC, File Handling Changes
ESPOLINTRN	31.0.0067	35956	P2591	FORTRAN BCL Formatting
ESPSIM	31.0.0002	35623	P2219	KIND=DISK Vs. FAMILYNAME
EVENTS	31.0.0001	34561	P1834	Test 309 Failed with GSC Absen
FILECOPY	31.0.0008	34993	P2188	Punched Output
FILECOPY	31.0.0011	35623	P2219	KIND=DISK Vs. FAMILYNAME
FILECOPY	31.0.0012	35647	P1847	II.9 WFL
FILECOPY	31.0.0013	35647	P1847	II.9 WFL
FILECOPY	31.0.0015	35647	P1847	II.9 WFL
FILECOPY	31.0.0016	37255	D2941	Time Value of Second Timestamp
FILECOPY	31.0.0017	37254	D2946	Volume Specification Not Allow
FILECOPY	31.0.0019	37489	P2555	(X) Vs. *X in EXCLUDE List
FILEDATA	31.0.0003	33530	P1364	Accesscode
FILEDATA	31.0.0007	35623	P2219	KIND=DISK Vs. FAMILYNAME
FILEDATA	31.0.0008	35743	P2296	Allow CANDE LFILES with No Di
FILEDATA	31.0.0010	36092	P2468	6250 BPI Tapes Recognized
FLIPFLOPS	31.0.0001	37404	D2977	Error Messages in Hex
FORTRAN	31.0.0009	32685	D2265	Standardization of Compiler Fi
FORTRAN	31.0.0015	34116	P1845	Monitor Binder Interface
FORTRAN	31.0.0017	34111	D2821	Format-Repeat Count
FORTRAN	31.0.0018	34113	P1848	Run-Time Group Repeat
FORTRAN	31.0.0019	34541	P1849	Bad PCW on Action Label
FORTRAN	31.0.0020	34118	P1850	Multiple Entry Points with OPT
FORTRAN	31.0.0022	34115	P1842	Invalid Syntax, IMPLICIT State
FORTRAN	31.0.0023	34119	P1843	Double Precision Constants OPT
FORTRAN	31.0.0024	34120	P1844	Invalid Type Checking of Subro
FORTRAN	31.0.0026	34512	P1874	Binding FORTRAN and ALGOL
FORTRAN	31.0.0028	34537	P1934	Invalid Syntax in Data Lists
FORTRAN	31.0.0029	34542	P1935	Repeat Count, Leading Zeros

PATCH TABLE

SOFTWARE	PATCH	PRI	NOTE	DESCRIPTION
FORTRAN	31.0.0030	34117	P1936	FREE Format Vs. INCLUDE
FORTRAN	31.0.0031	34536	P1937	INVALID INDEX, Subroutine Call
FORTRAN	31.0.0032	34535	P1938	Internal Compiler Fields Excee
FORTRAN	31.0.0033	34533	P1939	Run Time Problems with Very La
FORTRAN	31.0.0034	34534	D2822	Binding and Statistics
FORTRAN	31.0.0036	30376	D2979	Ports and Signals
FORTRAN	31.0.0038	35623	P2219	KIND=DISK Vs. FAMILYNAME
FORTRAN	31.0.0039	33838	D2604	Libraries
FORTRAN	31.0.0040	35811	P2302	Invalid PARAMETER MISMATCH
FORTRAN	31.0.0041	34527	P2322	FORTRAN XREF
FORTRAN	31.0.0047	34519	P2592	Core Estimate with $SET SEPARA
FUNCTIONAL	31.0.0001	37403	D2976	Deleted MISSING PORT Message
GUARDFILE	31.0.0002	33533	D2494	Accesscode
GUARDFILE	31.0.0004	35623	P2219	KIND=DISK Vs. FAMILYNAME
HARDCOPY	31.0.0003	35623	P2219	KIND=DISK Vs. FAMILYNAME
HOSTINTFACE	31.0.0003	37550	P2547	INCLUDE PROPERTIES
HOSTLIB	31.0.0002	34304	D2611	Transaction Processing System
HOSTLIB	31.0.0003	35623	P2219	KIND=DISK Vs. FAMILYNAME
HOSTLIB	31.0.0004	35426	P2362	Abort Handling Mechanism
HOSTLIB	31.0.0005	35426	P2362	Abort Handling Mechanism
HOSTLIB	31.0.0007	35313	P2418	Miscellaneous Corrections
HOSTLIB	31.0.0008	35426	P2362	Abort Handling Mechanism
HOSTLIB	31.0.0009	35313	P2418	Miscellaneous Corrections
HOSTLIB	31.0.0010	35313	P2418	Miscellaneous Corrections
HOSTLIB	31.0.0011	35313	P2418	Miscellaneous Corrections
HOSTLIB	31.0.0012	35313	P2418	Miscellaneous Corrections
HOSTLIB	31.0.0013	35313	P2418	Miscellaneous Corrections
HOSTLIB	31.0.0014	35313	P2418	Miscellaneous Corrections
HOSTLIB	31.0.0015	37222	P2548	Reading, Writing Over Rows
HOSTLIB	31.0.0016	35313	P2418	Miscellaneous Corrections
HOSTLIB	31.0.0017	35313	P2418	Miscellaneous Corrections
HOSTLIB	31.0.0019	35313	P2418	Miscellaneous Corrections
HOSTLIB	31.0.0020	35313	P2418	Miscellaneous Corrections
HOSTLIB	31.0.0021	35313	P2418	Miscellaneous Corrections
HOSTLIB	31.0.0022	35313	P2418	Miscellaneous Corrections
HOSTLIB	31.0.0023	35313	P2418	Miscellaneous Corrections
HOSTLIB	31.0.0024	35313	P2418	Miscellaneous Corrections
HOSTLIB	31.0.0025	35313	P2418	Miscellaneous Corrections
HOSTLIB	31.0.0026	35313	P2418	Miscellaneous Corrections
HOSTLIB	31.0.0027	35313	P2418	Miscellaneous Corrections
HOSTLIB	31.0.0028	35313	P2418	Miscellaneous Corrections
HOSTLIB	31.0.0029	35313	P2418	Miscellaneous Corrections
HOSTLIB	31.0.0030	35313	P2418	Miscellaneous Corrections
HOSTSERVICES	31.0.0037	36042	P2403	Correct Attribute Setting
HOSTSERVICES	31.0.0055	35987	D2861	B6800 Multiprocessor Systems
HOSTSERVICES	31.0.0057	35987	D2861	B6800 Multiprocessor Systems
HOSTSERVICES	31.0.0072	35987	D2861	B6800 Multiprocessor Systems
IADMAPPER	31.0.0004	35623	P2219	KIND=DISK Vs. FAMILYNAME
IN-OUTPUT	31.0.0017	32406	P9024	FIBLOCK and TIMESTAMP
IN-OUTPUT	31.0.0082	32693	P9167	NORESOURCEWAIT File Attribute
IN-OUTPUT	31.0.0130	32862	P9228	AVAILABLE
IN-OUTPUT	31.0.0132	32406	P9024	FIBLOCK and TIMESTAMP
IN-OUTPUT	31.0.0136	32857	P9272	FIBLOCKSNR
IN-OUTPUT	31.0.0141	30477	P9273	PROGRAMDUMP, New IOCB
IN-OUTPUT	31.0.0142	30478	P9274	CLOSE HERE on Empty Tape File
IN-OUTPUT	31.0.0143	32867	P9275	EVEN Parity Vs. EOF1
IN-OUTPUT	31.0.0152	32595	P1030	Change INTNAME
IN-OUTPUT	31.0.0155	34564	D2952	New Task, File Attributes
IN-OUTPUT	31.0.0182	33057	P1128	CHECKCOUNT Vs. 5500 Tapes
IN-OUTPUT	31.0.0195	33051	P1136	Tape Close Vs. MYUSE
IN-OUTPUT	31.0.0239	33208	P1603	Extract IOCBs from Buffers
IN-OUTPUT	31.0.0258	30478	P9274	CLOSE HERE on Empty Tape File
IN-OUTPUT	31.0.0259	30477	P9273	PROGRAMDUMP, New IOCB
IN-OUTPUT	31.0.0376	33313	P1781	CANCEL/DIRECTDCREAD
IN-OUTPUT	31.0.0418	34236	P1736	Negative Boolean
IN-OUTPUT	31.0.0424	34564	D2952	New Task, File Attributes
IN-OUTPUT	31.0.0449	34190	P1856	Log Vs. Title
IN-OUTPUT	31.0.0452	33991	P1860	FIBEOF
IN-OUTPUT	31.0.0464	33972	D2640	Update I/O Implementation Chan
IN-OUTPUT	31.0.0466	34403	D2646	SERIALNO Attribute Validity Ch
IN-OUTPUT	31.0.0470	34805	P1891	Badly Blocked Update I/O
IN-OUTPUT	31.0.0504	34190	P1856	Log Vs. Title
IN-OUTPUT	31.0.0601	35537	D2783	I/O Error Messages
IN-OUTPUT	31.0.0624	31272	D2785	HOSTNAME File Attribute
IN-OUTPUT	31.0.0659	34564	D2952	New Task, File Attributes
IN-OUTPUT	31.0.0787	35850	D2883	Count Logical I/O Requests
IN-OUTPUT	31.0.0816	31282	D2902	NEWFILE Vs. Tape Labels
INITIALIZER	31.0.0001	33384	D1744	RSC3
INQ	31.0.0031	34501	P1825	Display of All-Fraction Items
INQ	31.0.0032	34502	P1826	Display of Null Items
INQ	31.0.0033	34453	P2012	Value out of Range
INQ	31.0.0034	35353	P2208	INVALID INDEX Using Defines

PATCH TABLE

SOFTWARE	PATCH	PRI	NOTE	DESCRIPTION
INQ	31.0.0036	35623	P2219	KIND=DISK Vs. FAMILYNAME
INQ	31.0.0038	35302	D2709	B7700 Optimization
INQ	31.0.0039	35453	P2251	Display of Numeric Items
INQ	31.0.0040	35464	P2276	Display Limit for OPTION PRINT
INQ	31.0.0042	35921	P2389	Find Via Self-Correcting Link
INQ	31.0.0043	37074	P2481	Fault Alpha Control Item Break
INQ	31.0.0044	37227	P2500	Use of Defines in Certain Cont
INQ	31.0.0045	37340	P2501	Loss of Minus Sign
INQ	31.0.0047	37532	P2612	Comparisons of Literals
INTERFACE	31.0.0005	34304	D2611	Transaction Processing System
INTERFACE	31.0.0009	35623	P2219	KIND=DISK Vs. FAMILYNAME
INTERFACE	31.0.0010	35445	P2249	GROUP Items Invoked Incorrectl
INTERFACE	31.0.0011	35450	P2158	INVALID INDEX in Data Base Int
INTERFACE	31.0.0013	35302	D2709	B7700 Optimization
INTERFACE	31.0.0014	35481	P2357	Link to Set Qualification
INTERFACE	31.0.0016	35481	P2357	Link to Set Qualification
INTERPRETER	31.0.0001	34560	P1835	SNAP Compare Mode
INTERPRETER	31.0.0002	35115	P2070	Print Last and Current Records
INTERPRETER	31.0.0003	37401	D2975	Interrupt Error Handling
IOIOP	31.0.0001	36085	P2490	Tape Density
IOTEST	31.0.0001	36085	P2490	Tape Density
IOTEST	31.0.0002	36085	P2490	Tape Density
IOTEST	31.0.0003	37995	P2630	Include PK as Unittype
IXREF	31.0.0004	33167	D2476	INTERACTIVEXREF Enhancements
IXREF	31.0.0006	33167	D2476	INTERACTIVEXREF Enhancements
IXREF	31.0.0011	33697	D2524	Environment Improvement
IXREF	31.0.0012	33696	D2523	Improve Qualfication
IXREF	31.0.0017	31261	P1556	More Information in REF File
IXREF	31.0.0018	33976	D2823	"*BREAK*"
IXREF	31.0.0019	33977	D2555	Recognize Lower Case Input
IXREF	31.0.0020	33441	P1652	Files from NEWP Compiler
IXREF	31.0.0021	30376	D2979	Ports and Signals
IXREF	31.0.0022	34584	D2684	COMPLEX Implementation
IXREF	31.0.0024	34395	P2206	Port and Signal Arrays
IXREF	31.0.0025	35623	P2219	KIND=DISK Vs. FAMILYNAME
IXREF	31.0.0026	33838	D2604	Libraries
JOBFORMAT	31.0.0003	32687	D2246	B6800 MCP
JOBFORMAT	31.0.0004	32636	D2267	Four vs Three Digit System Ser
JOBFORMAT	31.0.0005	33067	P1074	JOBFORMATTER Loop
JOBFORMAT	31.0.0006	33199	P1123	SEG ARRAY Printing MCS String
JOBFORMAT	31.0.0012	33538	D2493	Accesscode
JOBFORMAT	31.0.0018	34063	D2627	Accesscode
JOBFORMAT	31.0.0019	34045	P1709	Includes LSN for Job Summary
JOBFORMAT	31.0.0020	33849	D2535	Revised ODT Messages
JOBFORMAT	31.0.0021	30499	D2637	Reason for Queue DS
JOBFORMAT	31.0.0025	35623	P2219	KIND=DISK Vs. FAMILYNAME
JOBFORMAT	31.0.0027	34021	D2647	LOG IOCONDITION Result Descrip
JOBFORMAT	31.0.0028	34385	P2271	Add Case 22, LP
JOBFORMAT	31.0.0029	35668	D2841	"MAJOR TYPE=2 MINOR TYPE=5" Vs
JOBFORMAT	31.0.0030	35806	P2272	SCR Jobs Generate JOBFORMATTER
JOBFORMAT	31.0.0031	35869	P2323	Not Skipping Bad Records
JOBFORMAT	31.0.0032	35871	P2324	Prevent INTEGER OVERFLOW
JOBFORMAT	31.0.0033	35844	D2869	MCP Level Indicators
JOBFORMAT	31.0.0034	35850	D2883	Count Logical I/O Requests
JOBFORMAT	31.0.0036	36100	P2382	CLOSE Types
JOBFORMAT	31.0.0037	36023	P2383	JOB ENTERED SYSTEM Time
JOBFORMAT	31.0.0038	37016	D2917	New BOJ/BOT, EOJ/EOT Log Infor
JOBFORMAT	31.0.0039	37014	D2929	Fixed Portion Marker of Log Re
JOBFORMAT	31.0.0041	37016	D2917	New BOJ/BOT, EOJ/EOT Log Infor
JOBFORMAT	31.0.0042	37665	D2953	New COMPILERINFO Format
JOBFORMAT	31.0.0043	37681	D2967	Log Boxes for Jobs
LCOBDOWNLINE	31.0.0004	35623	P2219	KIND=DISK Vs. FAMILYNAME
LCOBLIBRARY	31.0.0004	35623	P2219	KIND=DISK Vs. FAMILYNAME
LCOBOL	31.0.0007	34532	P2005	TU Firmware Buffer Lengths
LCOBOL	31.0.0009	34530	P2245	IF Statement with OPTIM Correc
LCOBOL	31.0.0010	34529	P2246	DISPLAY Statement Correction
LCOBOL	31.0.0011	35623	P2219	KIND=DISK Vs. FAMILYNAME
LCOBOL	31.0.0014	34522	P2463	Incorrect Code, Invalid Syntax
LISTVOLLIB	31.0.0001	34976	D2430	MCP Restructuring
LISTVOLLIB	31.0.0004	35623	P2219	KIND=DISK Vs. FAMILYNAME
LOADDUMP	31.0.0004	34475	P1923	INVALID INDEX with Deleted Str
LOADDUMP	31.0.0005	34469	P1924	Title Attribute Error
LOADDUMP	31.0.0007	35623	P2219	KIND=DISK Vs. FAMILYNAME
LOADER	31.0.0002	32076	P9117	235 Packs
LOADER	31.0.0003	32076	P9117	235 Packs
LOADER	31.0.0004	32866	D2273	IV vs 235
LOADER	31.0.0005	32576	P1078	Three-Multiplexor System
LOADER	31.0.0006	33247	P1155	LOADER Vs. IV
LOADER	31.0.0007	33249	P1178	Binary Disk Addressing
LOADER	31.0.0012	33627	P1388	Change ODT, Panel Displays
LOADER	31.0.0013	32866	D2273	IV vs 235
LOADER	31.0.0019	34092	D2572	Disallow PRINTERLABELS Option

PATCH TABLE

SOFTWARE	PATCH	PRI	NOTE	DESCRIPTION
LOADER	31.0.0020	34238	P1770	OLAYROW Greater Than 1200
LOADER	31.0.0021	34210	D2592	Time and Date Verification
LOADER	31.0.0022	34188	P1818	LOADER Vs. GMM
LOADER	31.0.0023	34014	P1852	TD850 ODT
LOADER	31.0.0024	34014	P1852	TD850 ODT
LOADER	31.0.0025	34965	P2006	Unit Display
LOADER	31.0.0026	35356	P2194	BX387 Firmware Load
LOADER	31.0.0027	34979	D2705	7A Tape Control
LOADER	31.0.0035	37458	P2577	B9246 Drum Printer in Dump
LOGANALY	31.0.0009	33531	P1370	Accesscode
LOGANALY	31.0.0017	33849	D2535	Revised ODT Messages
LOGANALY	31.0.0018	34211	P1812	System Serial Number is * Digi
LOGANALY	31.0.0019	34159	P1859	DT
LOGANALY	31.0.0021	35361	P2193	Halt/Load Records
LOGANALY	31.0.0022	35364	P2189	Cardnumber Vs. P3 Word
LOGANALY	31.0.0024	35391	P2212	Scratchpad Parity Error
LOGANALY	31.0.0025	35623	P2219	KIND=DISK Vs. FAMILYNAME
LOGANALY	31.0.0027	35648	P1940	Firmware ID in LH ODT Command
LOGANALY	31.0.0028	35649	D2833	MPX#,PATH# Vs. LH,UA,UR
LOGANALY	31.0.0029	34021	D2647	LOG IOCONDITION Result Descrip
LOGANALY	31.0.0030	35661	P2262	Garbage Output on UNSORTED
LOGANALY	31.0.0031	35668	D2841	"MAJOR TYPE=2 MINOR TYPE=5" Vs
LOGANALY	31.0.0032	35801	D2842	ABORT, ERRORS Vs. Other Option
LOGANALY	31.0.0033	35373	D2694	Implementation of DL
LOGANALY	31.0.0034	33849	D2535	Revised ODT Messages
LOGANALY	31.0.0036	36021	P2378	Character Count on Log Title
LOGANALY	31.0.0037	37014	D2929	Fixed Portion Marker of Log Re
LOGANALY	31.0.0039	37700	P2586	Family Substitution Vs. READK(
LOGGER	31.0.0003	33532	D2502	Accesscode
LOGGER	31.0.0008	34063	D2627	Accesscode
LOGGER	31.0.0014	34199	P1813	Items Extending Past Column 72
LOGGER	31.0.0017	35623	P2219	KIND=DISK Vs. FAMILYNAME
LOGGER	31.0.0018	35660	P2263	Elapsed Time for Restart MCS S
LOGGER	31.0.0019	35671	D2843	Summary Reports
LOGGER	31.0.0020	35673	P2264	Elapsed Time of Jobs Run Throu
LOGGER	31.0.0021	35675	D2844	Last STATISTICS Interval Saved
LOGGER	31.0.0023	36026	P2384	YTDFILE Update
LOGGER	31.0.0024	37488	P2556	ALINK Vs. MATCHM (BOJ,EOJ)
LTTABLEGEN	31.0.0001	33201	P1130	450/750 LPM Train Printers
LTTABLEGEN	31.0.0002	33200	P1124	User Specified Tables
LTTABLEGEN	31.0.0005	35623	P2219	KIND=DISK Vs. FAMILYNAME
MAINTMCS	31.0.0001	35484	P2370	Error Message Displays Correct
MAINTMCS	31.0.0002	35483	D2890	Extension to ADAPTER ADDRESS
MAKEUSER	31.0.0003	33527	D2503	Accesscode
MAKEUSER	31.0.0006	35623	P2219	KIND=DISK Vs. FAMILYNAME
MAKEUSER	31.0.0011	37646	D2964	Enhancements for DL USERDATA
MCP	31.0.0002	35987	D2861	B6800 Multiprocessor Systems
MCP	31.0.0003	32078	P9285	Task Restructuring
MCP	31.0.0004	32076	P9117	235 Packs
MCP	31.0.0007	32703	P9164	SWAPPER Queue Counts and Aging
MCP	31.0.0009	32701	P9162	NO MEM in Swapspace
MCP	31.0.0047	32687	D2246	B6800 MCP
MCP	31.0.0049	35987	D2861	B6800 Multiprocessor Systems
MCP	31.0.0054	32690	P9286	DM6700 Vestiges Removed
MCP	31.0.0055	34976	D2430	MCP Restructuring
MCP	31.0.0056	32691	P9288	EOJ Variables
MCP	31.0.0057	32664	P9143	Fatal Stack Overflow
MCP	31.0.0058	32663	P1374	MAXIOTIME
MCP	31.0.0059	32678	D2249	OF of GETSTATUS Waiting on <fi
MCP	31.0.0061	32464	P9145	CRUNCH Vs. COPY
MCP	31.0.0062	32466	P9146	Reserve Locks
MCP	31.0.0063	32465	P9161	SCHEDCORE
MCP	31.0.0066	32468	P9149	BADROW
MCP	31.0.0067	32463	P9150	FILEKIND
MCP	31.0.0068	32818	P9151	CDONLY
MCP	31.0.0069	32076	P9117	235 Packs
MCP	31.0.0071	32666	P9153	Change Title of Cataloged File
MCP	31.0.0073	32466	P9146	Reserve Locks
MCP	31.0.0074	32661	P9156	Task Attribute Errors
MCP	31.0.0077	32872	P9158	WFL USER Statement in Subrout
MCP	31.0.0078	32880	P9159	Information Carry Over
MCP	31.0.0080	32882	P9160	CANDE WFL Command with New WF
MCP	31.0.0081	32820	P9166	Protected Tapes
MCP	31.0.0085	31099	P9169	Processor Priority
MCP	31.0.0087	32694	P9170	WAITLIMIT Correction
MCP	31.0.0088	32695	P9171	Stacksearch Control
MCP	31.0.0090	32657	P9176	SWAPPER QT's BACKUP
MCP	31.0.0091	32823	P9173	LEIBNITZ EOTINHDR
MCP	31.0.0092	32825	P9172	BADDISK Vs. Directory
MCP	31.0.0096	32076	P9117	235 Packs
MCP	31.0.0097	32835	P9224	VERIFYFAMILY vs HDRVECTORLOCK
MCP	31.0.0098	32838	P9225	DESCRIPTOR error on Packs

PATCH TABLE				
SOFTWARE	PATCH	PRI	NOTE	DESCRIPTION
MCP	31.0.0099	32836	P9223	SUBSPACES vs USERCODE
MCP	31.0.0101	32078	P9285	Task Restructuring
MCP	31.0.0102	32697	P9245	Regulate STACKINFO Locking
MCP	31.0.0103	32698	D2268	Ignore Unknown Unit Types
MCP	31.0.0104	32076	P9117	235 Packs
MCP	31.0.0105	34976	D2430	MCP Restructuring
MCP	31.0.0110	32644	P9231	User Uses of TD830 ODT's for O
MCP	31.0.0111	32840	P9218	BD Files vs EOF=0
MCP	31.0.0112	32076	P9117	235 Packs
MCP	31.0.0113	32846	P9220	PLANT MESSAGE vs AUTOBACKUP
MCP	31.0.0114	32839	P9221	DBSSTACK vs GETSTATUS
MCP	31.0.0116	32854	P9219	DBS vs CHECKPOINT
MCP	31.0.0118	32844	P9222	CP vs SWAPPER
MCP	31.0.0122	32833	P9246	Papertape vs Reelswitch
MCP	31.0.0124	32183	D2269	RESERVE Enhancements
MCP	31.0.0125	32466	P9146	Reserve Locks
MCP	31.0.0128	32843	P9230	HISTORY vs PROGRAMDUMP
MCP	31.0.0129	32828	P9229	ANABOLISM vs MAKEJOBFILE
MCP	31.0.0131	32859	P9227	HDREXPAND
MCP	31.0.0133	32860	P9226	Sticky MEM Recursive GETSPACE
MCP	31.0.0134	32616	D2271	GETSTATUS MCSNAME
MCP	31.0.0135	32466	P9146	Reserve Locks
MCP	31.0.0137	32858	P9271	CP Vs. Backup Tape
MCP	31.0.0138	32863	P9270	DBS PROGRAMDUMP
MCP	31.0.0139	32607	P9276	Delete Zero Serial Number
MCP	31.0.0144	32866	D2273	IV vs 235
MCP	31.0.0145	32076	P9117	235 Packs
MCP	31.0.0146	32687	D2246	B6800 MCP
MCP	31.0.0147	32849	P1026	PATHRES
MCP	31.0.0148	32848	P1027	WASTEMPATCHECK
MCP	31.0.0149	32970	P1028	DASDL Update Timestamp
MCP	31.0.0150	32851	P1029	Queue Error Vs. THEQUE
MCP	31.0.0151	31687	P8629	PD to Missing Family
MCP	31.0.0153	32593	P1053	SUSPENDER Vs. OK from ODT
MCP	31.0.0156	33050	P1054	IC Diskpacks
MCP	31.0.0157	34976	D2430	MCP Restructuring
MCP	31.0.0158	33066	P1076	FLATREADER Errors
MCP	31.0.0159	33065	P9240	Tape Parity Retry Vs. Tapemark
MCP	31.0.0160	32586	P9269	Density Vs. Printlabel
MCP	31.0.0161	33064	P9304	IC Specs
MCP	31.0.0162	33063	P1082	LEIBNITZ Vs. Overlay
MCP	31.0.0163	33062	P1111	AUTOPRINT Vs. Multiple SKIP TO
MCP	31.0.0164	32580	P9209	Halt/Load with 64 Mods of Memo
MCP	31.0.0165	33061	P1112	AUTOPRINT Vs. FM Reply
MCP	31.0.0166	32579	P1081	NON ANCESTRAL TASKFILE
MCP	31.0.0168	32577	P1083	CATALOG ADD Tape Files
MCP	31.0.0170	33079	P1085	STATUSCHANGE Event Batched
MCP	31.0.0171	32889	D2434	Clearing Card Reader in Use by
MCP	31.0.0172	32574	D2439	Directory Conversion
MCP	31.0.0173	32076	P9117	235 Packs
MCP	31.0.0174	32573	P1086	Unlabeled Tape Corrections
MCP	31.0.0175	32572	P1087	MCSREADY on SM Input
MCP	31.0.0176	33058	D2447	COPY Vs. Serial Numbers
MCP	31.0.0178	33215	P1125	Unit Left Assigned
MCP	31.0.0180	33059	P1127	Binary Pack Addresses
MCP	31.0.0181	32687	D2246	B6800 MCP
MCP	31.0.0183	33209	P1101	RESERVE
MCP	31.0.0185	33206	P1102	Add ETX to All ODT Output File
MCP	31.0.0188	32687	D2246	B6800 MCP
MCP	31.0.0189	33205	P1129	File Removed Message
MCP	31.0.0190	33201	P1130	450/750 LPM Train Printers
MCP	31.0.0193	33053	P1134	UR Vs. Resource
MCP	31.0.0194	33052	P1135	Memory Access Error Vs. Time o
MCP	31.0.0197	34976	D2430	MCP Restructuring
MCP	31.0.0199	33244	P1180	TAPEUNIT Vs. VOLUNIT
MCP	31.0.0200	33245	P1181	II.6 Vs. Timestamp
MCP	31.0.0201	33247	P1155	LOADER Vs. IV
MCP	31.0.0202	33247	P1155	LOADER Vs. IV
MCP	31.0.0203	33250	P1182	DBS D3 NOMEM
MCP	31.0.0205	33185	P1154	Tape Write Parity After a Tape
MCP	31.0.0207	33248	P1184	EOT Vs. TAPEPARITYRETRY
MCP	31.0.0210	33254	P1231	235 Packs Vs. Model 3 MPX
MCP	31.0.0211	33256	P1191	FINDINPUT Vs. UNBRF
MCP	31.0.0212	33253	P1192	SWAPPER Vs. Crunch
MCP	31.0.0213	33257	P1193	FM Vs. SU
MCP	31.0.0215	34165	D2580	"Pre-2.4 ON <fault> Statements
MCP	31.0.0216	33086	P1225	KANGAROO-INTERCEDE Mechanism
MCP	31.0.0217	33087	P1226	EVENT Mechanism Changes
MCP	31.0.0219	33258	P1232	Disk Status Vs. Unit Moved
MCP	31.0.0220	33259	P1216	OF Vs. Exclusive
MCP	31.0.0221	33260	P1217	Family Substitution Vs. Catalo
MCP	31.0.0222	33252	D2463	RESOURCECHECK

PATCH TABLE				
SOFTWARE	PATCH	PRI	NOTE	DESCRIPTION
MCP	31.0.0225	33264	P1214	GIVEBACKDISK
MCP	31.0.0226	33263	P1219	REMOVE Vs. Security
MCP	31.0.0228	33264	P1214	GIVEBACKDISK
MCP	31.0.0229	30376	D2979	Ports and Signals
MCP	31.0.0230	33244	P1180	TAPEUNIT Vs. VOLUNIT
MCP	31.0.0231	33265	P1256	COPYIT Vs. GETUSERDISK
MCP	31.0.0234	34976	D2430	MCP Restructuring
MCP	31.0.0235	33267	P1253	Zip with Large Array
MCP	31.0.0237	33266	P1254	DISKLIMIT Vs. RSVP
MCP	31.0.0238	33268	P1252	SCR Vs. First Action
MCP	31.0.0242	33160	P1312	Improve AVAILIST Insert Algori
MCP	31.0.0243	33059	P1127	Binary Pack Addresses
MCP	31.0.0260	34976	D2430	MCP Restructuring
MCP	31.0.0286	33538	D2493	Accesscode
MCP	31.0.0290	34976	D2430	MCP Restructuring
MCP	31.0.0291	34976	D2430	MCP Restructuring
MCP	31.0.0308	33854	P1459	MISSINGPROCEDURE Call
MCP	31.0.0321	33814	P1552	New Firmware
MCP	31.0.0335	35987	D2861	B6800 Multiprocessor Systems
MCP	31.0.0343	33947	D2570	Disallow PRINTERLABELS Option
MCP	31.0.0346	33944	P1896	New GOTOSOLVER
MCP	31.0.0349	33370	P1609	Guardfile
MCP	31.0.0351	33956	D2576	RERUN Vs. EOF
MCP	31.0.0352	34074	P1712	VOLUME DELETE
MCP	31.0.0353	34071	P1713	Library Maintenance BADFILE Vs
MCP	31.0.0356	33317	P1715	Dying Stack in WSSHERIFF
MCP	31.0.0357	34067	P1716	PB 2 Level Name
MCP	31.0.0358	34063	D2627	Accesscode
MCP	31.0.0360	34060	P1723	DP Vs. MPXIII
MCP	31.0.0364	34062	P1717	MOVE Vs. LOG
MCP	31.0.0365	33314	P1719	ODT ETX Positioning
MCP	31.0.0366	33315	P1720	Checkpoint
MCP	31.0.0367	33316	P1718	IOTRACE
MCP	31.0.0368	33539	D2518	Segmented String Variables
MCP	31.0.0371	34061	P1721	LINKLISTINSERT Vs. XTEND
MCP	31.0.0373	34049	P1722	Reader Buss Parity Error
MCP	31.0.0374	34060	P1723	DP Vs. MPXIII
MCP	31.0.0375	33838	D2604	Libraries
MCP	31.0.0377	34044	P1724	CATBLKSIZEF
MCP	31.0.0380	34042	P1725	GENERATION
MCP	31.0.0381	34078	P1726	DUMP Vs. RD
MCP	31.0.0383	34271	P1727	AUTOBACKUP Priority
MCP	31.0.0386	33353	P1729	SIRWOFADDRESS
MCP	31.0.0387	34269	P1730	Checkpoint File Header Timesta
MCP	31.0.0388	34043	P1731	Model III Multiplexor
MCP	31.0.0389	34267	P1782	Guardfile Vs. DS
MCP	31.0.0390	34241	P1732	ELAPSEDTIMELIMIT
MCP	31.0.0392	34102	P1796	Timestamp Recalled Object-Job
MCP	31.0.0394	33849	D2535	Revised ODT Messages
MCP	31.0.0398	30376	D2979	Ports and Signals
MCP	31.0.0399	34287	D2605	New Function for DMSWAIT
MCP	31.0.0400	34257	P1752	Programdump Diagnostic
MCP	31.0.0401	33956	D2576	RERUN Vs. EOF
MCP	31.0.0404	35987	D2861	B6800 Multiprocessor Systems
MCP	31.0.0405	35987	D2861	B6800 Multiprocessor Systems
MCP	31.0.0407	35987	D2861	B6800 Multiprocessor Systems
MCP	31.0.0409	35987	D2861	B6800 Multiprocessor Systems
MCP	31.0.0413	34268	P1733	READALABEL Vs. EXTMODE
MCP	31.0.0414	35987	D2861	B6800 Multiprocessor Systems
MCP	31.0.0415	31268	P1753	Accept DUMMMP Tape Anytime
MCP	31.0.0416	33989	P1734	FM Vs. Saved LP
MCP	31.0.0417	34235	P1735	Reelswitch Vs. Catalog
MCP	31.0.0422	34228	P1783	MASKGETAREAF
MCP	31.0.0423	34378	P1764	B6700/B7700 Compatibility
MCP	31.0.0425	34229	P1784	CRUNCH Vs. CP
MCP	31.0.0426	34107	D2606	UNITS=CHARACTERS Vs. MODE=SING
MCP	31.0.0427	34393	P1754	Processor Bound Jobs in Swapsp
MCP	31.0.0428	34217	P1785	Software Interrupt Execution
MCP	31.0.0430	34218	P1786	Guardfile Vs. Stackoverflow
MCP	31.0.0431	34216	P1787	TAPEPARITYRETRY Vs. BLOCKEXIT
MCP	31.0.0433	34208	P1788	LOOKFORIT Control State
MCP	31.0.0436	34210	D2592	Time and Date Verification
MCP	31.0.0437	34209	P1789	Read LCC
MCP	31.0.0438	34400	P1790	Invalid CRITICAL BLOCK Termina
MCP	31.0.0439	34346	P1798	DMS Programs Swapped Out "FORE
MCP	31.0.0442	34204	P1799	PRINTLIMIT Vs. PROGRAMDUMP
MCP	31.0.0443	34200	P1800	WAITLIMIT
MCP	31.0.0444	34196	P1801	FIBLOCK Vs. PROGRAMDUMP
MCP	31.0.0454	33995	P1862	RSNINVALID
MCP	31.0.0456	33993	P1863	DISCSTATUS
MCP	31.0.0457	33990	P1864	FORGETSPACE Vs. FORM
MCP	31.0.0459	33997	P1865	Library Maintenance Open Error

PATCH TABLE

SOFTWARE	PATCH	PRI	NOTE	DESCRIPTION
MCP	31.0.0467	34005	P1876	RESERVE AS
MCP	31.0.0468	34016	D2644	Suspend and Resume Tasks by Bo
MCP	31.0.0469	34022	P1877	Library Maintenance Reel Switc
MCP	31.0.0476	34838	D2652	New SYSTEMSTATUS Case 14
MCP	31.0.0479	31265	P2066	Trap for Bad Disk I/O
MCP	31.0.0481	34817	P1894	Move Vs. RC
MCP	31.0.0483	34810	P2021	Allow Statistics To Be Printed
MCP	31.0.0485	34806	P2068	PRESERVER Array Parameter
MCP	31.0.0488	34405	P1897	No Bad Pointers in WFL
MCP	31.0.0489	34406	P1898	Compile-and-Go with SUBSPACES=
MCP	31.0.0492	34026	P2022	Rebuild Vs. AD
MCP	31.0.0496	33849	D2535	Revised ODT Messages
MCP	31.0.0499	34029	P2023	SEGO of Flatrows
MCP	31.0.0506	34964	P2024	GETSTATUS Vs. UINFO
MCP	31.0.0511	34968	P2105	DUP FAMILY
MCP	31.0.0515	33814	P1552	New Firmware
MCP	31.0.0520	34976	D2430	MCP Restructuring
MCP	31.0.0521	33849	D2535	Revised ODT Messages
MCP	31.0.0522	34407	D2795	DS Priority
MCP	31.0.0531	35257	D2819	New Display of Format Errors
MCP	31.0.0532	35246	P2153	Task DS
MCP	31.0.0534	33993	P1863	DISCSTATUS
MCP	31.0.0541	34979	D2705	7A Tape Control
MCP	31.0.0544	34992	P2176	Log MPX Scratchpad Memory
MCP	31.0.0545	34998	P2180	Synchronize Clocks
MCP	31.0.0549	33993	P1863	DISCSTATUS
MCP	31.0.0550	33849	D2535	Revised ODT Messages
MCP	31.0.0551	35355	P2181	Increase Stack Size for TAPEDU
MCP	31.0.0554	34979	D2705	7A Tape Control
MCP	31.0.0556	30392	P2182	Expandarow for Dope Vectors
MCP	31.0.0559	35359	D2737	Privileged Programs
MCP	31.0.0560	35403	D2733	Flexibility in REMOVE, CHANGE
MCP	31.0.0562	31266	D2738	Delete CPUTEST
MCP	31.0.0564	35373	D2694	Implementation of DL
MCP	31.0.0566	35376	D2781	Data Base Stacks Run in Local
MCP	31.0.0567	34976	D2430	MCP Restructuring
MCP	31.0.0569	33849	D2535	Revised ODT Messages
MCP	31.0.0571	35383	P2222	NDL Compatibility
MCP	31.0.0573	34979	D2705	7A Tape Control
MCP	31.0.0575	35365	P2198	GETAROW Vs. catalog
MCP	31.0.0579	35386	P2199	PATHRES, FLATREADER, ERRORHAND
MCP	31.0.0580	35393	D2757	New DMS Structure Numbers
MCP	31.0.0582	35391	P2212	Scratchpad Parity Error
MCP	31.0.0583	35392	P2213	BX380 Vs. Channel Addressing
MCP	31.0.0584	35395	P2214	AB Vs. EOF
MCP	31.0.0585	35486	P2215	TURNOVERLAYKEY
MCP	31.0.0589	35623	P2219	KIND=DISK Vs. FAMILYNAME
MCP	31.0.0593	34979	D2705	7A Tape Control
MCP	31.0.0594	35491	P2221	STATISTICS
MCP	31.0.0595	35493	P2223	FORGETAREA RCW Trace
MCP	31.0.0604	35500	D2864	Relative I/O for COBOL74
MCP	31.0.0605	35501	D2784	NIF and DCPCODE Disk I/O Error
MCP	31.0.0606	35502	P2224	Improve Handling of DSing DCPs
MCP	31.0.0607	35503	P2225	FA of Unlabeled File to Remote
MCP	31.0.0608	35504	P2226	DSed MCS was Causing Hung Task
MCP	31.0.0609	35505	P2227	Schedule Stations Must Be Atta
MCP	31.0.0610	34411	P2241	Handle Long AX into Short Arra
MCP	31.0.0611	35506	P2228	DCP Fault with Space Queue Loc
MCP	31.0.0612	35507	P2229	AUTODC and Station Attach
MCP	31.0.0613	35508	P2230	PC Lock Vs. NIFSEARCH
MCP	31.0.0614	34412	P2233	DCALGOL EPILOG Procedures
MCP	31.0.0615	35509	P2231	Correct Primary Queue Number
MCP	31.0.0616	35510	P2232	Tanked Output Messages
MCP	31.0.0617	35608	P2242	PROGRAMDUMP Corrections
MCP	31.0.0621	35391	P2212	Scratchpad Parity Error
MCP	31.0.0622	35396	P2235	Rebuild Vs. DONTWAIT
MCP	31.0.0633	34021	D2647	LOG IOCONDITION Result Descrip
MCP	31.0.0634	34979	D2705	7A Tape Control
MCP	31.0.0637	34026	P2022	Rebuild Vs. AD
MCP	31.0.0638	35649	D2833	MPX#,PATH# Vs. LH,UA,UR
MCP	31.0.0645	35656	P1952	Tape Open
MCP	31.0.0646	34390	P2265	BUFFERS Attribute for Open Fil
MCP	31.0.0647	35662	D2834	Check Stack Image Compatibilit
MCP	31.0.0648	35391	P2212	Scratchpad Parity Error
MCP	31.0.0651	35659	P2266	Security Vs. Backup
MCP	31.0.0652	35664	P2286	GETSTATUS UNIT Vs. DISKPACKSEA
MCP	31.0.0656	34384	P2267	Locking in MIXREQUEST
MCP	31.0.0657	35110	D2794	Deimplementation of XALGOL Com
MCP	31.0.0658	35111	D2796	Deimplementation of Old Intrin
MCP	31.0.0661	35500	D2864	Relative I/O for COBOL74
MCP	31.0.0662	35669	P2268	CL Vs. AB
MCP	31.0.0672	35616	P2269	DCALGOL EPILOG Revisited

PATCH TABLE

SOFTWARE	PATCH	PRI	NOTE	DESCRIPTION
MCP	31.0.0678	34021	D2647	LOG IOCONDITION Result Descrip
MCP	31.0.0685	35373	D2694	Implementation of DL
MCP	31.0.0689	35713	P2273	RESERVE Loop
MCP	31.0.0691	35373	D2694	Implementation of DL
MCP	31.0.0692	35693	P2274	CLOSE Errors Fatal
MCP	31.0.0701	35670	P2275	"COPY =" Many Files
MCP	31.0.0705	35620	D2881	Provide Firmer MAKEUSER-USERDA
MCP	31.0.0712	35722	P2301	AB Buffer Size
MCP	31.0.0715	35111	D2796	Deimplementation of Old Intrin
MCP	31.0.0719	35740	P2300	MPX Vs. Tape Parity
MCP	31.0.0720	35811	P2302	Invalid PARAMETER MISMATCH
MCP	31.0.0724	35622	D2855	Time Out for Memory Dumps
MCP	31.0.0725	35839	P2304	Restore LOSR After Dump
MCP	31.0.0727	35388	D2906	Changes in the SWAPPER Mechani
MCP	31.0.0729	35739	P2310	Class Count UFLO
MCP	31.0.0735	35747	P2303	SYSTEMSTATUS String Protect
MCP	31.0.0736	35738	P2311	EOJ Vs. USTKASSIGNED
MCP	31.0.0737	35737	P2305	STICKY Memory
MCP	31.0.0738	35854	P2312	SWAPPER Gets DIV BY ZERO
MCP	31.0.0740	35858	P2438	Library Program Dump Options
MCP	31.0.0743	35843	P2336	Clear Parameters if Task Initi
MCP	31.0.0745	35856	P2306	USERPARITYBIT
MCP	31.0.0749	35829	P2307	VERIFYFAMILY Vs. BU
MCP	31.0.0750	35722	P2301	AB Buffer Size
MCP	31.0.0755	31280	P2419	Improved I/O Error Message
MCP	31.0.0762	35500	D2864	Relative I/O for COBOL74
MCP	31.0.0769	35500	D2864	Relative I/O for COBOL74
MCP	31.0.0773	35844	D2869	MCP Level Indicators
MCP	31.0.0776	35971	P2422	Array Parameters by Value
MCP	31.0.0777	35987	D2861	B6800 Multiprocessor Systems
MCP	31.0.0779	35896	P2364	SIB Environment
MCP	31.0.0781	34803	D2642	SYSTEMSTATUS Intrinsic
MCP	31.0.0783	35891	P2358	OVERLAYCF
MCP	31.0.0789	35985	D2867	Soft Configuration of Global M
MCP	31.0.0793	33849	D2535	Revised ODT Messages
MCP	31.0.0794	35832	P2396	CPRESTART EOJ
MCP	31.0.0795	35901	D2888	Library Maintenance Result Rep
MCP	31.0.0796	33849	D2535	Revised ODT Messages
MCP	31.0.0797	35903	P2391	Catalog Remove
MCP	31.0.0804	35985	D2867	Soft Configuration of Global M
MCP	31.0.0805	35373	D2694	Implementation of DL
MCP	31.0.0807	35985	D2867	Soft Configuration of Global M
MCP	31.0.0808	35985	D2867	Soft Configuration of Global M
MCP	31.0.0809	35985	D2867	Soft Configuration of Global M
MCP	31.0.0810	35985	D2867	Soft Configuration of Global M
MCP	31.0.0811	35388	D2906	Changes in the SWAPPER Mechani
MCP	31.0.0812	36109	P2423	Memory Priority for Swapjobs
MCP	31.0.0814	36112	P2392	Swap Out Waiting for Schedule
MCP	31.0.0823	33538	D2493	Accesscode
MCP	31.0.0824	33849	D2535	Revised ODT Messages
MCP	31.0.0825	35693	P2274	CLOSE Errors Fatal
MCP	31.0.0826	35376	D2781	Data Base Stacks Run in Local
MCP	31.0.0827	36048	P2425	Size Check of User Array
MCP	31.0.0844	36144	D2908	Initialization Needs More Memo
MCP	31.0.0847	35373	D2694	Implementation of DL
MCP	31.0.0848	36032	P2428	Uninitiated I/O Dump
MCP	31.0.0849	36189	P2427	MPX Paths
MCP	31.0.0850	36059	P2546	DD Vs. CM
MCP	31.0.0854	36062	D2907	Automatic Move
MCP	31.0.0856	37016	D2917	New BOJ/BOT, EOJ/EOT Log Infor
MCP	31.0.0858	36190	P2448	CL READALABEL
MCP	31.0.0860	36059	P2546	DD Vs. CM
MCP	31.0.0862	33849	D2535	Revised ODT Messages
MCP	31.0.0867	37097	P2430	New High Order Priority Scheme
MCP	31.0.0868	37098	P2431	PROGRAMDUMP Be Wary in Defunct
MCP	31.0.0869	37099	P2433	Correct TERMINATE Vs. BLOCKEXI
MCP	31.0.0870	37104	P2436	Statistics Improvements
MCP	31.0.0872	35987	D2861	B6800 Multiprocessor Systems
MCP	31.0.0873	37107	P2435	Revise GETAREA Pool
MCP	31.0.0875	37109	P2624	Protection from Tag-7 Interrup
MCP	31.0.0876	33849	D2535	Revised ODT Messages
MCP	31.0.0881	35987	D2861	B6800 Multiprocessor Systems
MCP	31.0.0885	36107	D2928	Eliminate Vector Mode, Improve
MCP	31.0.0887	37202	P2485	GIVEBACKHDR Correction
MCP	31.0.0888	37014	D2929	Fixed Portion Marker of Log Re
MCP	31.0.0896	35987	D2861	B6800 Multiprocessor Systems
MCP	31.0.0898	36117	P2488	Automatic Shortening of SEARCH
MCP	31.0.0903	37117	P2538	Spurious PACK IN USE Message
MCP	31.0.0906	37262	P2536	PATH MARKED OFFLINE
MCP	31.0.0907	37269	P2537	Reel Switch Vs. Parity Error
MCP	31.0.0913	36195	P2534	CHECKPOINT Vs. JEDGARHOOVER
MCP	31.0.0914	35985	D2867	Soft Configuration of Global M

SOFTWARE	PATCH	PRI	NOTE	DESCRIPTION
MCP	31.0.0916	33849	D2535	Revised ODT Messages
MCP	31.0.0917	37121	P2533	Disallow Crunched USERDATAFILE
MCP	31.0.0920	33849	D2535	Revised ODT Messages
MCP	31.0.0922	35985	D2867	Soft Configuration of Global M
MCP	31.0.0924	37247	P2532	CHECKPOINTFILE Vs. Abort
MCP	31.0.0931	37237	P2545	GETAREA Vs. Locks
MCP	31.0.0932	37238	P2531	CHANGEMCP INVALID INDEX
MCP	31.0.0933	37235	P2530	CHECKPOINT Vs. OLAYHEADER
MCP	31.0.0934	34979	D2705	7A Tape Control
MCP	31.0.0941	37518	P2519	New Traps
MCP	31.0.0942	37517	D2938	GETSTATUS: Test for Compiler
MCP	31.0.0943	36150	D2942	Verify Memory Dump Tape
MCP	31.0.0944	35985	D2867	Soft Configuration of Global M
MCP	31.0.0946	35987	D2861	B6800 Multiprocessor Systems
MCP	31.0.0951	35388	D2906	Changes in the SWAPPER Mechani
MCP	31.0.0958	36094	D2937	Reserve Time Cases for BSP
MCP	31.0.0963	34979	D2705	7A Tape Control
MCP	31.0.0964	37481	D2936	COBOL File Already Closed Erro
MCP	31.0.0968	37474	P2514	PURGIT RD Message
MCP	31.0.0969	37475	P2513	CATALOG PURGE
MCP	31.0.0971	31309	P2566	Put HOSTNAME and GROUPID in Pr
MCP	31.0.0976	35693	P2274	CLOSE Errors Fatal
MCP	31.0.0978	37467	P2552	Card Reader Secured Status
MCP	31.0.0983	37521	P2584	Avoid Extra Mom on B6800 Negat
MCP	31.0.0987	37704	P2588	Initializing FIB in Different
MCP	31.0.0994	37710	D2969	Idle Patterns
MCP	31.0.0996	37705	P2590	CHECKPOINT RESTART
MCP	31.0.1005	37665	D2953	New COMPILERINFO Format
MCP	31.0.1009	37709	P2597	PROGRAMDUMP Heading
MCP	31.0.1017	37804	P2595	Setting Family Name After Assi
MCP	31.0.1021	37793	D2963	Another Group Not Responding
MCP	31.0.1027	37681	D2967	Log Boxes for Jobs
MCP	31.0.1035	37975	P2599	PROGRAMDUMP Hang
MCP	31.0.1040	37992	D2971	System Maintenance for B6800 M
MCP	31.0.1041	35318	D2802	Retain or Reuse Old Structure
MCP	31.0.1042	37988	P2625	Printer Dump Loop
MCP	31.0.1053	34837	P1890	Multiprocessor System SUSPENDE
MEMTEST	31.0.0001	35116	P2071	Stop Paging While Looping
MILO	31.0.0005	33380	P1463	RSC3
MIL6700	31.0.0001	33397	D2743	Debug
MIL6700	31.0.0007	35623	P2219	KIND=DISK Vs. FAMILYNAME
MPX	31.0.0001	34546	P1836	TD850 Revisions
NDL	31.0.0001	32563	P2085	FILE <family> Statement
NDL	31.0.0003	35217	D2702	Use Proper Processor ID in Hea
NDL	31.0.0004	35215	D2703	Implement Application Number V
NDL	31.0.0007	35623	P2219	KIND=DISK Vs. FAMILYNAME
NDL	31.0.0008	35535	D2772	Implement Variable Size Statio
NDL	31.0.0009	35539	P2325	Further Auxiliary Logic Correc
NDL	31.0.0010	36012	P2405	Basic Control/ACII Position Ch
NDL	31.0.0011	36013	P2406	Autonomous DCP Buffers for Sta
NDL	31.0.0012	36014	P2404	Eliminate Looping on Severe Er
NDL	31.0.0013	36015	P2401	Extend Labels for 20 Bit Addre
NDL	31.0.0015	37020	P2444	Erroneous Syntax Error on Defi
NDL	31.0.0016	37022	P2469	Suppress "IF B1 OR B2" Code Wh
NDL	31.0.0017	37027	P2557	Exchanged DCP Vs. Auxiliary Lo
NEWP	31.0.0069	35623	P2219	KIND=DISK Vs. FAMILYNAME
NEWP	31.0.0071	35410	D2771	Rebinding External Procedures
NEWP	31.0.0110	33849	D2535	Revised ODT Messages
NEWP	31.0.0119	37665	D2953	New COMPILERINFO Format
PATCH	31.0.0002	34321	D2596	Label Equation for "$." Cards
PATCH	31.0.0005	35623	P2219	KIND=DISK Vs. FAMILYNAME
PEP	31.0.0001	34545	P1837	TD850 Test Runs with No Read B
PEP	31.0.0002	37651	P2621	Test Restricted to Pack Types
PKACTUATOR	31.0.0001	37843	P2631	Disk Pack Controller Error Mes
PKBASIC	31.0.0001	37843	P2631	Disk Pack Controller Error Mes
PKHDOVER	31.0.0001	37843	P2631	Disk Pack Controller Error Mes
PKHEADISOL	31.0.0001	37843	P2631	Disk Pack Controller Error Mes
PKINTERCH	31.0.0001	37843	P2631	Disk Pack Controller Error Mes
PKSCAN	31.0.0001	34829	P2072	Logging Autocorrection RD
PKSCAN	31.0.0002	37661	P2620	Work with All Pack Types
PKSCAN	31.0.0003	37660	P2616	Checking for Used Spare Sector
PKSCAN	31.0.0004	37659	P2617	Log Error for AUTOCORRECT
PKSCAN	31.0.0005	37658	P2618	PKSCAN for Write Disabled Pack
PKSEEK	31.0.0001	37843	P2631	Disk Pack Controller Error Mes
PKTEST	31.0.0001	34821	P2073	Remote Usage of SPOTAB Option
PKTEST	31.0.0002	34822	P2074	TAB Option Causing INVALID IND
PKTEST	31.0.0003	34823	P2075	Direct I/O Handling
PKTEST	31.0.0004	37651	P2621	Test Restricted to Pack Types
PKWRITEREAD	31.0.0001	37843	P2631	Disk Pack Controller Error Mes
PKXD	31.0.0001	37654	P2619	Miscellaneous Changes
PK04	31.0.0001	37651	P2621	Test Restricted to Pack Types
PK05	31.0.0001	37661	P2620	Work with All Pack Types

PATCH TABLE

SOFTWARE	PATCH	PRI	NOTE	DESCRIPTION
PK06	31.0.0001	37651	P2621	Test Restricted to Pack Types
PK07	31.0.0001	37651	P2621	Test Restricted to Pack Types
PK08	31.0.0001	37651	P2621	Test Restricted to Pack Types
PK09	31.0.0001	37651	P2621	Test Restricted to Pack Types
PK10	31.0.0001	37651	P2621	Test Restricted to Pack Types
PK11	31.0.0001	37651	P2621	Test Restricted to Pack Types
PK15	31.0.0001	37651	P2621	Test Restricted to Pack Types
PK16	31.0.0001	37651	P2621	Test Restricted to Pack Types
PLI	31.0.0007	32685	D2265	Standardization of Compiler Fi
PLI	31.0.0012	30376	D2979	Ports and Signals
PLI	31.0.0014	33838	D2604	Libraries
PLI	31.0.0016	33450	P1941	Deallocate Dynamic Bounds Auto
PLI	31.0.0022	35223	P2163	Based Double Variables
PLI	31.0.0023	35282	P2164	Extra Subscript Syntax Error
PLI	31.0.0026	35230	P2165	Bad Procedure Directory
PLI	31.0.0027	35229	P2166	Defines for Doubles
PLI	31.0.0028	35225	P2167	Flag Illegal Array Bounds
PLI	31.0.0029	35226	P2168	Loop on Define
PLI	31.0.0030	35227	P2169	Bad Dummy for String Intrinsic
PLI	31.0.0031	35228	P2170	CALL Verb with EVENT Option
PLI	31.0.0035	35623	P2219	KIND=DISK Vs. FAMILYNAME
PLI	31.0.0040	33838	D2604	Libraries
PLI	31.0.0049	33459	P2386	WRITE Statement Without "FROM"
PLI	31.0.0050	33460	P2385	Missing Commas in File Options
PLI	31.0.0051	33461	P2445	Miscellaneous Faults Using PIC
PLI	31.0.0052	33462	P2446	DO STATEMENT problem
PLI	31.0.0053	33463	P2447	Level 9 Error
PLINTRN	31.0.0001	32078	P9285	Task Restructuring
PLINTRN	31.0.0002	34165	D2580	"Pre-2.4 ON <fault> Statements
PLINTRN	31.0.0009	33913	D2565	Intrinsics
PLINTRN	31.0.0019	30376	D2979	Ports and Signals
PLINTRN	31.0.0024	35262	P2171	Real Bit Constants
PLINTRN	31.0.0025	35263	P2172	Filling Out Bit Strings
PLINTRN	31.0.0026	35275	P2173	Conversion Error
PLINTRN	31.0.0027	35276	P2174	ISAM Intrinsics, Infinite Loop
PLINTRN	31.0.0030	35707	P2250	ISAM, Eliminate Superhalt
PLINTRN	31.0.0031	33458	P2346	Eliminate SEG ARRAY Error in T
PRINTAUDIT	31.0.0001	33798	D2618	Decode Date and Time
PRINTAUDIT	31.0.0002	33797	D2607	New I/O Error Handling Procedu
PRINTAUDIT	31.0.0003	34304	D2611	Transaction Processing System
PRINTAUDIT	31.0.0004	34468	P1914	FILEDC a Control Record
PRINTAUDIT	31.0.0005	34451	P2027	Hex Dump of Block
PRINTAUDIT	31.0.0007	35623	P2219	KIND=DISK Vs. FAMILYNAME
PRINTAUDIT	31.0.0008	35302	D2709	B7700 Optimization
PRINTAUDIT	31.0.0010	37214	P2502	Bad Check for Label Equation
PRINTAUDIT	31.0.0011	37348	P2503	Cope with Empty Audit Files
PRINTAUDIT	31.0.0012	37346	P2504	Fault on Partial Record
PRINTBIND	31.0.0005	35103	P2025	Improved Handling of Unknown I
PRINTBIND	31.0.0006	35108	P2026	Correct Handling of PL/I Items
PRINTBIND	31.0.0007	35102	D2685	Printing of Program Unit Direc
PRINTBIND	31.0.0008	35104	D2686	Listing Format Improvements
PRINTBIND	31.0.0009	35105	D2687	Improve Handling of DATABASE I
PRINTBIND	31.0.0010	35106	P2065	Large Code Files
PRINTBIND	31.0.0012	35112	D2722	Binding of Ports and Signals
PRINTBIND	31.0.0013	34673	D2714	Transaction Record Parameters
PRINTBIND	31.0.0015	35623	P2219	KIND=DISK Vs. FAMILYNAME
PRINTBIND	31.0.0017	37292	P2470	Prevent Printing Too Many EXTR
PRINTCOPY	31.0.0003	35623	P2219	KIND=DISK Vs. FAMILYNAME
PROPERTIES	31.0.0001	30012	P1585	Retry ADDRESSCHECK Failures
PROPERTIES	31.0.0002	33118	D2568	Allow More Structures, Items P
PROPERTIES	31.0.0003	33797	D2607	New I/O Error Handling Procedu
PROPERTIES	31.0.0004	33759	D2619	REBLOCKING
PROPERTIES	31.0.0005	34304	D2611	Transaction Processing System
PROPERTIES	31.0.0006	34358	D2615	DASDL Defaults for Data Sets a
PROPERTIES	31.0.0007	33759	D2619	REBLOCKING
PROPERTIES	31.0.0008	34334	D2657	Find Correct End of Disk Type
PROPERTIES	31.0.0009	34329	D2658	Improve Buffer Management
PROPERTIES	31.0.0010	34489	P1911	Remaps of Ordered Data Sets wi
PROPERTIES	31.0.0011	34438	P2010	Checksum Error
PROPERTIES	31.0.0012	35302	D2709	B7700 Optimization
PROPERTIES	31.0.0013	35303	P2130	ROWLOCKOUTAUDIT Errors
PROPERTIES	31.0.0014	35336	D2731	REORGANIZATION Optimization
PROPERTIES	31.0.0015	35318	D2802	Retain or Reuse Old Structure
PROPERTIES	31.0.0016	35302	D2709	B7700 Optimization
PROPERTIES	31.0.0017	35350	P9236	Optimize Evaluation of Checksu
PROPERTIES	31.0.0018	35339	D2807	Enhancements to Quickfix
PROPERTIES	31.0.0020	35623	P2219	KIND=DISK Vs. FAMILYNAME
PROPERTIES	31.0.0021	35440	D2803	Automatic Checksum of Intermed
PROPERTIES	31.0.0023	35452	D2832	DUPLICATES Subcategory 2 Expla
PROPERTIES	31.0.0025	35465	P2284	Prevent Checksum Error
PROPERTIES	31.0.0026	35473	D2889	TPSDUALUPDATE Option Added
PROPERTIES	31.0.0028	35427	P2361	Overlapping Use of Openerrors

PATCH TABLE SOFTWARE	PATCH	PRI	NOTE	DESCRIPTION
PROPERTIES	31.0.0030	35427	P2361	Overlapping Use of Openerrors
PROPERTIES	31.0.0031	37223	P2549	Level for RECONINFO Files
READERSORTER	31.0.0002	33086	P1225	KANGAROO-INTERCEDE Mechanism
READERSORTER	31.0.0004	33389	P1435	Scan Error Word
READERSORTER	31.0.0006	33385	D2745	RSC3
RECOVERY	31.0.0016	34304	D2611	Transaction Processing System
RECOVERY	31.0.0017	34305	D2617	Visible DBS Changes and Enhanc
RECOVERY	31.0.0018	34334	D2657	Find Correct End of Disk Type
RECOVERY	31.0.0019	34333	P1805	Ordered EOF Set Incorrectly
RECOVERY	31.0.0020	34350	P1809	Incorrect Time Printed
RECOVERY	31.0.0021	34344	P1806	Tape Audit Handling
RECOVERY	31.0.0022	34343	P1807	48 Users of Restart Data Set
RECOVERY	31.0.0023	34342	P1808	Audit Reel Switch
RECOVERY	31.0.0024	34495	D2659	Audit Handling Enhancements
RECOVERY	31.0.0025	34473	P1925	ABORT Gets WAITING ON ROWLOCKO
RECOVERY	31.0.0026	34441	P2014	Correct Rebuild Restart
RECOVERY	31.0.0027	34438	P2010	Checksum Error
RECOVERY	31.0.0028	34439	P2013	Audit Positioning Error
RECOVERY	31.0.0029	34426	P2101	Data Base Corruption by Rollba
RECOVERY	31.0.0030	34423	P2102	Rebuild, Rollback Restart Erro
RECOVERY	31.0.0031	35303	P2130	ROWLOCKOUTAUDIT Errors
RECOVERY	31.0.0032	34438	P2010	Checksum Error
RECOVERY	31.0.0033	35341	P2186	Openerror Message Improved
RECOVERY	31.0.0035	35339	D2807	Enhancements to Quickfix
RECOVERY	31.0.0037	35344	P2183	Checksum Errors
RECOVERY	31.0.0038	35623	P2219	KIND=DISK Vs. FAMILYNAME
RECOVERY	31.0.0039	35448	P2159	Make ODT Array Local to Interr
RECOVERY	31.0.0040	35302	D2709	B7700 Optimization
RECOVERY	31.0.0041	35465	P2284	Prevent Checksum Error
RECOVERY	31.0.0042	35470	P2335	EOF Too Large
RECOVERY	31.0.0043	35912	P2359	Resequence Data Base and RECOV
RECOVERY	31.0.0045	35918	P2449	DS Abort Causes System Hang
RECOVERY	31.0.0046	35912	P2359	Resequence Data Base and RECOV
RECOVERY	31.0.0047	36170	P2413	Abort Backs Up Too Far
RECOVERY	31.0.0048	36169	P2414	REBUILD and RECONSTRUCT Back U
RECOVERY	31.0.0049	37089	P2458	Bad Retry Logic
RECOVERY	31.0.0050	37088	P2459	Timing problem in REBUILD/ROLL
RECOVERY	31.0.0051	37086	P2482	"FAILED TO SET END-OF-FILE" Me
RECOVERY	31.0.0052	37078	P2483	Quickfix May Fail on Checksumm
RECOVERY	31.0.0053	37088	P2459	Timing problem in REBUILD/ROLL
RECOVERY	31.0.0054	37348	P2503	Cope with Empty Audit Files
RECOVERY	31.0.0055	37342	P2505	Reconstruct Ignores Quiet Poin
RECOVERY	31.0.0057	38027	P2640	ZOT CURSTAMP at End of Audit
RECOVERY	31.0.0058	38043	P2641	FIXLASTAUDITBLOCK Audit Positi
REMOTELIB	31.0.0006	35313	P2418	Miscellaneous Corrections
REORG	31.0.0001	32008	P1778	Warnings When Compiling Variab
REORG	31.0.0004	34358	D2615	DASDL Defaults for Data Sets a
REORG	31.0.0005	34324	P1827	DATAERROR 001 when Reorganizin
REORG	31.0.0006	34499	P1828	Link Fixup in Variable Format
REORG	31.0.0007	34494	P1829	INVALID INDEX on Restart of Ge
REORG	31.0.0008	34491	P1926	Block Initialization for Direc
REORG	31.0.0009	34479	P1927	Multiple Reel Intermediate Tap
REORG	31.0.0010	34471	P1912	Reorganization of Compact Data
REORG	31.0.0011	34459	P1902	Compact Data Sets with Record
REORG	31.0.0012	34457	P1903	Generation of Manual Subsets
REORG	31.0.0013	34454	P1904	IOERRORTYPE 7 on Fixup of Stan
REORG	31.0.0014	34437	P2103	Faulty Output To Tape
REORG	31.0.0015	34425	P2134	Duplicates of Ordered List
REORG	31.0.0016	34424	P2135	Generate of Index Using Differ
REORG	31.0.0017	34418	P2136	Unused Fold Words
REORG	31.0.0019	35336	D2731	REORGANIZATION Optimization
REORG	31.0.0020	35318	D2802	Retain or Reuse Old Structure
REORG	31.0.0021	35335	D2732	Elimination of REORGPENDING
REORG	31.0.0023	35352	P2209	Invalid Fixup of Data Sets
REORG	31.0.0024	35430	P2210	Index Random, Random Shadow Fi
REORG	31.0.0025	35432	P2211	Fixup of Empty Variable Format
REORG	31.0.0027	35440	D2803	Automatic Checksum of Intermed
REORG	31.0.0028	35623	P2219	KIND=DISK Vs. FAMILYNAME
REORG	31.0.0030	35302	D2709	B7700 Optimization
REORG	31.0.0032	36176	P2589	Record Format Change
REORG	31.0.0034	37337	P2506	Reorganization of Compact Data
RESEQBASIC	31.0.0005	35623	P2219	KIND=DISK Vs. FAMILYNAME
RJE	31.0.0018	33948	P1610	RJELEVEL Display Change
RJE	31.0.0019	34073	D2578	"BUG DUMP <arrayname>"
RJE	31.0.0020	34063	D2627	Accesscode
RJE	31.0.0024	34272	D2597	DEBUG Output Change
RJE	31.0.0026	34243	P1772	RJE AUTOPRINT Hang
RJE	31.0.0027	34233	P1794	WFLCOMPILER Not Going to EOT
RJE	31.0.0028	34227	P1795	LSNRAY Vs. Stations with NO-LI
RJE	31.0.0029	34224	D2625	CLEAR <lsn> Command
RJE	31.0.0030	34194	P1814	Swapping Print Queues Vs. Stat
RJE	31.0.0031	34193	P1815	CONT and PRINT Banners

SOFTWARE	PATCH	PRI	NOTE	DESCRIPTION
RJE	31.0.0032	34197	D2632	MAXTERMINALS Not a Power of 2
RJE	31.0.0033	34187	P1816	QT Printer Files Saved Over Ha
RJE	31.0.0035	34004	D2638	Number of Stations Attached
RJE	31.0.0036	34186	D2641	LINKFILE
RJE	31.0.0037	34033	P1942	Restartfile Name SEG ARRAY Err
RJE	31.0.0038	34985	P2086	MAXTERMINALS Greater Than 48
RJE	31.0.0040	35370	P2204	"*ME COPIES 2" Vs. Resize of F
RJE	31.0.0041	35371	P2205	NOLOGON Vs. Usercode "."
RJE	31.0.0043	35623	P2219	KIND=DISK Vs. FAMILYNAME
RJE	31.0.0044	35518	D2810	File Transfer
RJE	31.0.0045	35515	P1579	STATIONID or USER Changes
RJE	31.0.0046	35646	P1037	Reconfiguration with Active AU
RJE	31.0.0047	35635	D2828	WAIT State Maintained
RJE	31.0.0048	35848	D2862	WFLMESSAGE ARRAY Usage Increas
RJE	31.0.0049	35904	P2374	Unit Device Number
RJE	31.0.0051	37487	P2561	"*ME" Fails on Specific Filena
RJE	31.0.0052	37486	P2562	DESTNAME Directed REMCP Files
RJE	31.0.0054	37477	P2563	REMCP File Titles on Word Boun
RLTABLEGEN	31.0.0003	35623	P2219	KIND=DISK Vs. FAMILYNAME
RSANALYZER	31.0.0002	32903	D2746	Change Card File Name
RSLOG	31.0.0001	33391	P1441	RSC3
RSLOG	31.0.0003	35623	P2219	KIND=DISK Vs. FAMILYNAME
RSMCP	31.0.0894	36090	P2487	RSMONITOR
RSNETL	31.0.0002	35623	P2219	KIND=DISK Vs. FAMILYNAME
RSPANALYZER	31.0.0002	35623	P2219	KIND=DISK Vs. FAMILYNAME
SCR	31.0.0001	32078	P9285	Task Restructuring
SCR	31.0.0002	32687	D2246	B6800 MCP
SCR	31.0.0003	32802	D2414	Add RSC-3 Test Name
SCR	31.0.0004	34976	D2430	MCP Restructuring
SCR	31.0.0006	32828	P9229	ANABOLISM vs MAKEJOBFILE
SCR	31.0.0007	32850	P1014	TD830, Diskpack 235 Correction
SCR	31.0.0008	32850	P1014	TD830, Diskpack 235 Correction
SCR	31.0.0009	32850	P1014	TD830, Diskpack 235 Correction
SCR	31.0.0011	33303	P1171	VERIFY Does Not Recognize 5N D
SCR	31.0.0012	33304	P1172	Deadly Embrace
SCR	31.0.0013	33305	P1173	Address Cylinder 811
SCR	31.0.0014	33306	P1174	Serial Number Expanded
SCR	31.0.0015	33307	P1175	SCR Test 9
SCR	31.0.0016	33308	P1176	Default VERSION
SCR	31.0.0017	33309	P1177	Buffer Fill Correction
SCR	31.0.0018	33086	P1225	KANGAROO-INTERCEDE Mechanism
SCR	31.0.0021	33482	P1241	Make File Card FILETYPE=8
SCR	31.0.0022	33483	P1242	Return Saved Memory
SCR	31.0.0023	33484	P1251	SCR MARKNO III
SCR	31.0.0024	33485	P1243	No Decode of RD=1FFFF
SCR	31.0.0025	33486	P1244	NRZ Tapes
SCR	31.0.0026	33487	P1245	SCR HI TESTAPE Message
SCR	31.0.0027	33488	P1246	TESTOP RD Analysis
SCR	31.0.0028	33489	P1247	Pack Unit Subtype Field
SCR	31.0.0029	33492	P1307	Allow SET, RESET, POP of VOIDT
SCR	31.0.0036	34976	D2430	MCP Restructuring
SCR	31.0.0043	34378	P1764	B6700/B7700 Compatibility
SCR	31.0.0049	33849	D2535	Revised ODT Messages
SCR	31.0.0053	35623	P2219	KIND=DISK Vs. FAMILYNAME
SCR	31.0.0054	31554	D2758	GCR Tape Allowed
SCRMCP	31.0.0890	36082	P2486	IOTIMEOUT
SCRMCP	31.0.0908	37268	P2535	Label CASE Statements
SCRMCP	31.0.0909	37257	D2947	CURRENTDATE
SCRMCP	31.0.1002	37651	P2621	Test Restricted to Pack Types
SCRMCP	31.0.1031	37978	P2615	Handle VOIDT Properly
SCTABLEGEN	31.0.0003	32603	P1015	Add Reserved Words to Dictiona
SCTABLEGEN	31.0.0004	32581	P1080	Remove Unused TO
SCTABLEGEN	31.0.0009	33849	D2535	Revised ODT Messages
SCTABLEGEN	31.0.0011	34091	D2571	Disallow PRINTERLABELS Option
SCTABLEGEN	31.0.0012	33849	D2535	Revised ODT Messages
SCTABLEGEN	31.0.0013	33849	D2535	Revised ODT Messages
SCTABLEGEN	31.0.0015	34210	D2592	Time and Date Verification
SCTABLEGEN	31.0.0017	34011	D2648	Add Reserved Words
SCTABLEGEN	31.0.0018	31265	P2066	Trap for Bad Disk I/O
SCTABLEGEN	31.0.0019	34679	D2681	New ODT Option IODIAGNOSTICS
SCTABLEGEN	31.0.0020	33849	D2535	Revised ODT Messages
SCTABLEGEN	31.0.0021	33849	D2535	Revised ODT Messages
SCTABLEGEN	31.0.0023	33849	D2535	Revised ODT Messages
SCTABLEGEN	31.0.0025	33849	D2535	Revised ODT Messages
SCTABLEGEN	31.0.0026	35373	D2694	Implementation of DL
SCTABLEGEN	31.0.0029	35623	P2219	KIND=DISK Vs. FAMILYNAME
SCTABLEGEN	31.0.0033	35985	D2867	Soft Configuration of Global M
SCTABLEGEN	31.0.0035	35985	D2867	Soft Configuration of Global M
SCTABLEGEN	31.0.0036	31266	D2738	Delete CPUTEST
SCTABLEGEN	31.0.0037	33849	D2535	Revised ODT Messages
SORT	31.0.0001	32078	P9285	Task Restructuring
SORT	31.0.0002	32852	P1016	KANGAROO

PATCH TABLE				
SOFTWARE	PATCH	PRI	NOTE	DESCRIPTION
SORT	31.0.0004	33086	P1225	KANGAROO-INTERCEDE Mechanism
SORT	31.0.0007	34976	D2430	MCP Restructuring
SORT	31.0.0011	33955	P1710	Security Violation on SORT/STA
SORT	31.0.0012	34058	P1711	MERGE Vs. SWAPPER
SORT	31.0.0014	34012	P1854	Error for Insufficient Core
SORT	31.0.0015	33849	D2535	Revised ODT Messages
SORTERCONTRL	31.0.0001	33395	P1372	Allow Filler with USAGE
SORTERCONTRL	31.0.0002	33380	P1463	RSC3
SORTERCONTRL	31.0.0003	33893	P1464	RSC3
SORTERCONTRL	31.0.0004	33892	D2747	RSC3
SORTERCONTRL	31.0.0005	33891	D2748	RSC3
SORTERCONTRL	31.0.0010	35623	P2219	KIND=DISK Vs. FAMILYNAME
SORTSTAT	31.0.0003	35623	P2219	KIND=DISK Vs. FAMILYNAME
SOURCENDL	31.0.0001	34605	P2088	Changes to POLLCONTENTION Line
SOURCENDL	31.0.0002	34608	P2087	Correct Scrolling on TD820/TD8
SOURCENDL	31.0.0003	34607	D2701	Implementation of "?BRK" for S
SOURCENDL	31.0.0004	32527	P2089	Prevent Infinite Loop
SOURCENDL	31.0.0005	34609	D2700	Automatic Recall of Output Mes
SOURCENDL	31.0.0006	34611	D2699	AUXLOGIC Reduces Local Memory
SOURCENDL	31.0.0007	34612	P2090	Changes to POLLDISPLAYS Reques
SOURCENDL	31.0.0008	34613	P2091	Correct Transmission of NUL Ch
SOURCENDL	31.0.0009	35218	D2698	Message Oriented Datacom (BBSC
SOURCENDL	31.0.0010	34984	P2092	Prevent Reset of Tally Bit
SOURCENDL	31.0.0011	34983	P2093	Recognize ENQ Correctly
SOURCENDL	31.0.0012	35380	P2203	Line TOGS and TALLYS Cleared
SOURCENDL	31.0.0014	35518	D2810	File Transfer
STATSTOPPER	31.0.0004	35623	P2219	KIND=DISK Vs. FAMILYNAME
STATSTOPPER	31.0.0005	35492	P2220	Output Formatting
STATSTOPPER	31.0.0007	35976	D2912	File Names, New Modes, Flag SI
SUPERIV	31.0.0001	34824	P2076	SUPERIV Corrections
SUPERIV	31.0.0002	37651	P2621	Test Restricted to Pack Types
SUSPENDER	31.0.0003	34837	P1890	Multiprocessor System SUSPENDE
SUSPENDER	31.0.0009	34837	P1890	Multiprocessor System SUSPENDE
TABLEGEN	31.0.0004	32451	D2224	Warnings for Toggles, Overflow
TABLEGEN	31.0.0017	35623	P2219	KIND=DISK Vs. FAMILYNAME
TFL	31.0.0002	34304	D2611	Transaction Processing System
TFL	31.0.0003	35623	P2219	KIND=DISK Vs. FAMILYNAME
TFL	31.0.0005	37234	P2507	TFL Update
TFL	31.0.0006	37336	P2527	Scanner Problems
TFL	31.0.0007	37229	P2550	Default Text
TRINTERFACE	31.0.0002	34304	D2611	Transaction Processing System
TRPROPERTY	31.0.0002	34304	D2611	Transaction Processing System
TRPROPERTY	31.0.0003	35623	P2219	KIND=DISK Vs. FAMILYNAME
TRPROPERTY	31.0.0005	37229	P2550	Default Text
TRPROPERTY	31.0.0007	35315	P2551	Fields in PORT Message
TRUTILITY	31.0.0002	34304	D2611	Transaction Processing System
TRUTILITY	31.0.0003	35623	P2219	KIND=DISK Vs. FAMILYNAME
TRUTILITY	31.0.0005	36182	P2462	Miscellaneous Corrections
TRUTILITY	31.0.0006	36182	P2462	Miscellaneous Corrections
TRUTILITY	31.0.0007	36182	P2462	Miscellaneous Corrections
TRUTILITY	31.0.0008	36182	P2462	Miscellaneous Corrections
TRUTILITY	31.0.0009	36182	P2462	Miscellaneous Corrections
TRUTILITY	31.0.0010	36182	P2462	Miscellaneous Corrections
TRUTILITY	31.0.0011	36182	P2462	Miscellaneous Corrections
TRUTILITY	31.0.0012	36182	P2462	Miscellaneous Corrections
TRUTILITY	31.0.0013	36182	P2462	Miscellaneous Corrections
UDSTRCTTAB	31.0.0002	33528	D2505	Accesscode
UDSTRCTTAB	31.0.0004	34972	D2892	SYSTEMUSER Bit
USERSTRUCT	31.0.0002	33529	D2506	Accesscode
USERSTRUCT	31.0.0006	35623	P2219	KIND=DISK Vs. FAMILYNAME
UTIL	31.0.0010	33746	P1757	Print Patch Level in Heading
UTIL	31.0.0013	34303	D2620	Miscellaneous Changes
UTIL	31.0.0014	34355	D2621	Verify UTILITY Dump Tapes Reve
UTIL	31.0.0015	34353	P1779	INVALID INDEX Printing Ordered
UTIL	31.0.0016	34358	D2615	DASDL Defaults for Data Sets a
UTIL	31.0.0017	34351	P1780	Allow Reconstruction of New Ro
UTIL	31.0.0018	34348	P1810	Remove ROWLOCKOUTAUDIT at Star
UTIL	31.0.0019	34350	P1809	Incorrect Time Printed
UTIL	31.0.0020	34325	P1868	Locate H/L File for Unaudited
UTIL	31.0.0021	34326	P1869	DBDIR with ROWLOCK Option
UTIL	31.0.0022	34327	P1948	CANCEL Option Causes Attribute
UTIL	31.0.0024	34497	P1871	Missing Entry in HL File
UTIL	31.0.0025	34483	P1928	Tape Block Count Not Properly
UTIL	31.0.0026	34486	P1929	Correct Family Not Set
UTIL	31.0.0027	34484	P1949	Halt/Load File Title
UTIL	31.0.0028	34485	P1930	Reel Version Location of Row
UTIL	31.0.0030	34465	P1950	LIST Format Error
UTIL	31.0.0031	34463	P1951	Invalid Structure Name
UTIL	31.0.0032	34462	P1931	Cycle 9 and Above May Not Rest
UTIL	31.0.0033	34460	P1932	IOERROR on Tape
UTIL	31.0.0034	34448	P2016	CONTINUE Request Does Not Recr
UTIL	31.0.0035	34447	P2017	Resets DATABEGIN Following Res

PATCH TABLE				
SOFTWARE	PATCH	PRI	NOTE	DESCRIPTION
UTIL	31.0.0036	34446	P2018	Erase Recpninfo for Rows Skipp
UTIL	31.0.0037	34445	P2019	Corrects TIMESTAMP Compares
UTIL	31.0.0038	34444	P2015	DUMPTIME Taken at Wrong Time
UTIL	31.0.0039	34435	P2104	Multiprocessor Problems
UTIL	31.0.0040	34420	D2710	CHECKSUM and I/O Retry
UTIL	31.0.0041	35310	P2137	Continuation Request for COPY
UTIL	31.0.0042	35309	P2138	FILELISTREQUIRED on Restart
UTIL	31.0.0043	35308	P2139	Timing Problem in Scanner
UTIL	31.0.0044	35335	D2732	Elimination of REORGPENDING
UTIL	31.0.0045	35339	D2807	Enhancements to Quickfix
UTIL	31.0.0046	35423	D2808	Eliminate II.8 Reconstruct Syn
UTIL	31.0.0048	35623	P2219	KIND=DISK Vs. FAMILYNAME
UTIL	31.0.0049	34449	P1600	Halt/Load Information
UTIL	31.0.0050	35439	P2216	TAPEDIRECTORY Command Problems
UTIL	31.0.0051	35444	P1884	TDATABEGIN Field Not Initializ
UTIL	31.0.0052	34419	D2708	Dump Tape Directory
UTIL	31.0.0053	34419	D2708	Dump Tape Directory
UTIL	31.0.0055	35428	P2390	CFNAME Array Size Increased
UTIL	31.0.0056	35312	P2415	Report on New Quickfix Limits
UTIL	31.0.0057	35929	P2416	Print New Control File Informa
UTIL	31.0.0059	36165	P2460	END OF FILE Errors on Dumps
UTIL	31.0.0060	36164	P2398	Incorrect Selection of Structu
UTIL	31.0.0061	36174	P2417	FLUSHDB Option Correction
UTIL	31.0.0062	36177	P2461	COPY AS Error
UTIL	31.0.0063	36177	P2461	COPY AS Error
UTIL	31.0.0065	36178	P2484	INITIALIZE of Global Data
UTIL	31.0.0066	34419	D2708	Dump Tape Directory
UTIL	31.0.0067	36177	P2461	COPY AS Error
UTIL	31.0.0069	34419	D2708	Dump Tape Directory
UTIL	31.0.0070	34419	D2708	Dump Tape Directory
UTIL	31.0.0071	34420	D2710	CHECKSUM and I/O Retry
UTIL	31.0.0072	38032	P2642	Failure to Dump Partitions
UTILOADER	31.0.0002	34188	P1818	LOADER Vs. GMM
UTILOADER	31.0.0005	34979	D2705	7A Tape Control
UTILOADER	31.0.0006	34979	D2705	7A Tape Control
WFL	31.0.0003	34976	D2430	MCP Restructuring
WFL	31.0.0012	32685	D2265	Standardization of Compiler Fi
WFL	31.0.0014	35326	D2741	Accesscode
WFL	31.0.0017	34379	P1830	Prevent SEG ARRAY Error when J
WFL	31.0.0018	34380	P1873	Incorrect Code for File Title
WFL	31.0.0019	34554	P1875	Concatenation of More Than 15
WFL	31.0.0021	34551	P2007	REMOVE Command
WFL	31.0.0022	34552	P2001	Zip With Array
WFL	31.0.0023	34553	P2002	Zip with Array
WFL	31.0.0025	34906	P2008	ELSE No Longer Ignored
WFL	31.0.0026	34910	P2077	Extra BEGIN in WFL Job Deck
WFL	31.0.0027	35161	P2116	BCL Jobfile Library Problem
WFL	31.0.0028	35162	P2117	Ambiguous COPY Statements
WFL	31.0.0029	35163	P2118	Zip with Array Looping
WFL	31.0.0034	35168	P2119	Instruction Blocks
WFL	31.0.0036	35170	P2120	History
WFL	31.0.0037	35171	P2121	WFL Fault in Error Handling Pr
WFL	31.0.0039	35173	P2122	Reserved Words as Familynames
WFL	31.0.0040	35174	P2123	SEG ARRAY in Job Stack
WFL	31.0.0043	35177	P2124	Decimal Number
WFL	31.0.0044	35178	P2125	Fault on Bad Task Attribute
WFL	31.0.0045	35179	P2175	Chargecode Vs. One Statement O
WFL	31.0.0046	35180	P2126	Chargecode Vs. DECK Statement
WFL	31.0.0048	35403	D2733	Flexibility in REMOVE, CHANGE
WFL	31.0.0053	34915	D2793	"OF <hostname>" Recognized in
WFL	31.0.0059	35190	D2925	WFL Job Transfer To/From Remot
WFL	31.0.0064	35196	D2926	Copy Files to Remote Hosts
WFL	31.0.0073	35208	P2464	WFL "BEGIN; JOB" Correction
WFL	31.0.0074	35209	P2471	COMPILATION ABORTED Not Trunca
WFL	31.0.0075	35213	D2932	STATUS Attribute
WFL	31.0.0078	37372	P2509	Eliminate Recursive Handling o
WFL	31.0.0079	37373	P2510	COMPILE Statement Without Comp
XALGOL	31.0.0005	35623	P2219	KIND=DISK Vs. FAMILYNAME
XALGOL	31.0.0006	35110	D2794	Deimplementation of XALGOL Com
XREFANALY	31.0.0003	32630	P9193	Memory Requirements
XREFANALY	31.0.0004	33167	D2476	INTERACTIVEXREF Enhancements
XREFANALY	31.0.0013	31261	P1556	More Information in REF File
XREFANALY	31.0.0015	30376	D2979	Ports and Signals
XREFANALY	31.0.0016	34584	D2684	COMPLEX Implementation
XREFANALY	31.0.0018	34395	P2206	Port and Signal Arrays
XREFANALY	31.0.0019	35623	P2219	KIND=DISK Vs. FAMILYNAME
XREFANALY	31.0.0020	33838	D2604	Libraries
XREFANALY	31.0.0022	36104	D2898	Change Heading

IMPLEMENTATION

OF

DATA DICTIONARY

1. INTRODUCTION TO THE DATA DICTIONARY

1.1 Data Dictionary

The Data Dictionary consists of a data base containing information about data and transaction bases and the programs which reference them, together with programs for entering and deleting this information, and for collating it and displaying various reports.

It will make it easier for Data Base Administrators to control which programs reference data bases, and will facilitate program maintenance by listing which programs would be affected by a change to a data base.

This section will describe some of the ways in which the Data Dictionary could be used; more details will be found in later sections.

1.2 Entering Data Base and Program Information

Suppose that a number of programs use a data base called EMPLOYEES. After the DASDL compiler has processed the DASDL description of this data base it will write a file DESCRIPTION/EMPLOYEES, which is used to compile the access routines and for other purposes.

To enter this data base into the Data Dictionary, run the DDUPDATE program and enter the command

```
INSERT DATABASE EMPLOYEES;
```

This will take information from the description file (DESCRIPTION/EMPLOYEEES) and enter it into the Data Dictionary.

When programs which use this data base are being compiled, the compiler option DATADICTINFO should be SET for the whole compilation, by putting a

```
$ SET DATADICTINFO
```

card at the very beginning of the source program, before any source language statements. This will cause the compiler to save the usage of data base items, and to include this usage within the object code file.

Suppose that three programs which use the EMPLOYEES data base have been compiled into the code files OBJECT/PAYROLL, OBJECT/PENSION and OBJECT/MAILING, with the DATADICTINFO compiler option set.

The usage from these programs can be entered into the Data Dictionary by running the DDUPDATE program, and entering the command

```
INSERT PROGRAMS OBJECT/PAYROLL, OBJECT/PENSION, OBJECT/MAILING;
```

(All data bases accessed by these programs must already be in the Data Dictionary.)

A list of the bases and programs in the Data Dictionary can be obtained using the command

```
DISPLAY SOURCES;
```

1.3 Generating Reports from the Data Dictionary

Once the DDUPDATE prgram has entered a data base and programs which reference it, the DDREPORT program (and also the DDUPDATE program) can be run to produce cross reference reports showing which programs access each data base item, and which data bases and items are referenced by each program. For example, the commands

```
SELECT EMPLOYEES DATABASE;
DISPLAY XREF;
```

will generate a report showing which programs reference each item of the EMPLOYEES data base, and whether the program read or wrote that item. If the item has been given an internal name within the program, that internal name will also be shown.

The report may be confined to a single item, for example

```
DISPLAY XREF ZIP-CODE OF EMPLOYEE-ADDRESS;
```

which will report only on programs which reference the item ZIP-CODE of EMPLOYEE-ADDRESS.

These reports may either be displayed on a terminal, or printed, using the commands

```
OPTIONS TERMINAL;
or
OPTIONS PRINTER;
```

If there are remaps or logical data bases belonging to the EMPLOYEES data base, a change to one item may also affect remaps of that item, and logical data bases which include it. Also, programs which reference subitems of it, or group items which contain it may also be affected. The programs affected by such a change can be reported using the CHANGE IMPACT command:

```
SELECT EMPLOYEES DATABASE;
DISPLAY CHANGE IMPACT ZIP-CODE OF EMPLOYEE-ADDRESS;
```

A report showing which data bases are referenced by a particular program can be produced by a USAGE command, for example

```
DISPLAY USAGE BY PROGRAM PAYROLL;
```

A list of all items within a particular data base referenced by a particular program, and showing whether they are read or written, can be produced by the commands

```
SELECT EMPLOYEES DATABASE;
DISPLAY USAGE OF ITEMS BY PROGRAM PENSION;
```

Section 6 describes all the reporting facilities.

1.4 Documentation Facilities

The Data Dictionary can index files containing documentation text for items within a data base, and include this text in some of the reports. The documentation text may be in any format up to 72 characters wide, and extend over as many records as necessary. Section 5 describes how to prepare files of documentation text.

Suppose that documentation text for the EMPLOYEES data base has been prepared in a file titled EMPLOYEES/DOC. This file can be indexed by the Data Dictionary using an

```
INSERT TEXT EMPLOYEES/DOC;
```

command. The actual text is not entered into the Data Dictionary, but only its location within the file, and the title of the file.

This information could later be deleted from the Data Dictionary using a

```
DELETE TEXT EMPLOYEES/DOC;
```

command.

This documentation text can be displayed by DISPLAY ALPHABETIC and DISPLAY ATTRIBUTES commands, for example

```
SELECT EMPLOYEES DATABASE;
DISPLAY ALPHABETIC ITEMS WITH TEXT;
```

which lists alphabetically all the items within the EMPLOYEES data base together with the documentation text for each item.

The commands

```
SELECT EMPLOYEES DATABASE;
DISPLAY ATTRIBUTES OF ZIP-CODE WITH TEXT;
```

will display a summary of the attributes of the ZIP-CODE item, with its documentation text.

```
DISPLAY ATTRIBUTES OWNED BY EMPLOYEE-ADDRESS WITH TEXT;
```

would display that information for all items belonging to EMPLOYEE-ADDRESS.

2. USING DATA DICTIONARY PROGRAMS

2.1 Introduction to Data Dictionary Programs

There are three programs associated with the Data Dictionary. The first, DDINITIALIZE, simply creates a new Data Dictionary data base, and initializes it. Any existing Data Dictionary data base would be purged and its information lost. This initialization would probably be done very infrequently, and then only by the Data Base Administrator (DBA).

The second program, DDUPDATE, inserts new information into the Data Dictionary, and deletes old information which is no longer required. It would be run whenever a data base was updated, or a new program became operational, or an old program was no longer needed, and the DBA should supervise its use.

DDUPDATE would usually be run interactively from a terminal, the commands entered and executed one by one, and the result of each command displayed on the terminal. However, the command could be prepared in a disk file or card desk it the same commands were required frequently. Also, the results can be printed if a permanent record is wanted, and the program can be run in batch mode also.

All of the reporting facilities, described below for DDREPORT, are also provided by DDUPDATE.

The third program, DDREPORT, generates the various reports, collating the information within the Data Dictionary. It cannot update the Data Dictionary in any way, and so it would probably be available to any person who needed the information. It can be run either interactively or in batch mode, and its input and output may use terminals, disk files, cards or printer. It accepts only the reporting commands and will not act upon any updating commands.

2.2 Compiling Data Dictionary Software

The Data Dictionary data base DATADICTDB is compiled using the DASDL compiler from the DASDL file DATADICT/DATADICTDB, creating DESCRIPTION/DATADICTDB. The DASDL compiler will also ZIP WFL (if $ZIP is not reset) to compile the following:

ACCESSROUTINES/DATADICTDB
UTILITY/DATADICTDB
RECOVERY/DATADICTDB
DATARECOVERY/DATADICTDB
RECONSTRUCT/DATADICTDB

Example:

```
 100 BEGIN JOB COMPILE/DD/DASDL;
 200   USER=MYNAME/XYZ;
 300   CLASS=40;
 400   FAMILY DISK=DMS OTHERWISE DISK;
 500   COMPILE DATADICTDB WITH DASDL LIBRARY;
 600   COMPILER FILE CARD(KIND=DISK,
                 TITLE=*DATADICT/DATADICTDB);
 700 END JOB;
```

DDREPORT and DDUPDATE are both compiled from the symbol file DATADICT/DDUPDATE. DDREPORT is compiled with $INQUIRY set and DDUPDATE is compiled with $INQUIRY reset.

Example:

```
 100 BEGIN JOB COMPILE/DD/DDREPORT;
 200   USER=MYNAME/XYZ;
 300   CLASS=40;
 400   FAMILY DISK=DMS OTHERWISE DISK;
 500   COMPILE DDREPORT WITH DMALGOL LIBRARY;
 600   COMPILER FILE TAPE=*DATADICT/DDUPDATE;
 700   COMPILER FILE DASDL=*DESCRIPTION/DATADICTDB;
 800   COMPILER DATA CARD;
 900 $RESET LIST
1000 $SET INQUIRY
1100 $SET MERGE
1200 ?END JOB;
```

Example:

```
 100 BEGIN JOB COMPILE/DD/DDUPDATE;
 200   USER=MYNAME/XYZ;
 300   CLASS=40;
 400   FAMILY DISK=DMS OTHERWISE DISK;
 500   COMPILE DDUPDATE WITH DMALGOL LIBRARY;
 600   COMPILER FILE TAPE=*DATADICT/DDUPDATE;
 700   COMPILER FILE DASDL=*DESCRIPTION/DATADICTDB;
 800   COMPILER DATA CARD;
 900 $RESET INQUIRY LIST
1000 $SET MERGE
1100 ?END JOB;
```

Note that DESCRIPTION/DATADICTDB must exist before either DDREPORT or DDUPDATE can be compiled. The file DATADICT/DDHELP is not compiled, but it must be accessible from DDREPORT and DDUPDATE, as it contains the various HELP messages output by these two programs. The internal name of DATADICT/DDHELP in DDREPORT and DDUPDATE is HELPFILE; it has the following attributes:

KIND=PACK, FAMILYNAME="DISK", FILETYPE=8.

This set of attributes may be modified through the use of label-equation when compiling DDREPORT and DDUPDATE. For example, if DATADICT/DDHELP is stored on the family DBPACK, the following statement could be inserted in the WFL decks for compiling DDREPORT and DDUPDATE (at line 750 in the two examples above):

```
FILE HELPFILE(FAMILYNAME=DBPACK)
```

DDINITIALIZE is compiled from the symbol file DATADICT/DDINITIALIZE.

Example:

```
 100 BEGIN JOB COMPILE/DD/DDINITIALIZE;
 200   USER=MYNAME/XYZ;
 300   CLASS=40;
 400   FAMILY DISK=DMS OTHERWISE DISK;
 500   COMPILE DDINITIALIZE WITH DMALGOL LIBRARY;
 600   COMPILER FILE TAPE=*DATADICT/DDUPDATE;
 700   COMPILER FILE DASDL=*DESCRIPTION/DATADICTDB;
 800   COMPILER DATA CARD;
 900 $RESET LIST
1000 $SET MERGE
1100 ?END JOB;
```

Note that DESCRIPTION/DATADICTDB must exist before DDINITIALIZE can be compiled.

2.3 Initialization Program

The program DDINITIALIZE is run from a terminal, and firstly asks whether initialization is really wanted. It will not proceed further unless an affirmative reply is given. It then runs the Data Dictionary UTILITY/DATADICTDB program which creates empty files for all Data Dictionary structures, and removes any existing structures of a previous Data Dictionary. Finally, various Data Dictionary parameters are initialized.

Using CANDE, the program is run by typing

```
RUN DDINITIALIZE
```

2.4 Commands of the Update and Report Programs

The DDUPDATE and DDRREPORT programs are controlled by commands which specify what updates are to be done, or what reports are required. For example:

```
INSERT DATABASE PERSONNEL;
DELETE PROGRAM PAYROLL;
DISPLAY SOURCES;
SELECT PERSONNEL DATABASE;
DISPLAY XREF ZIP-CODE OF HOME-ADDRESS;
QUIT;
```

specifies that a data base is to be inserted, a program deleted, a report produced showing the sources of all the information in the Data Dictionary, and a cross reference report generated, showing all programs which reference the data item ZIP-CODE OF HOME ADDRESS, within the PERSONNEL data base. The QUIT command terminates execution of the program.

The rules for writing these commands are as follows:

1. All input is free format.

2. When Data Dictionary statements are entered via a terminal, the statements are executed immediately. Normally, statements can be entered on a single line; however, some statements may exceed the width of the terminal. A % character, entered at the end of the line, indicates that more input will follow. When the line is transmitted, the Data Dictionary responds with # and additional input may be entered. This process may be repeated as many times as necessary. When the statement is complete, the user can transmit the last line without a % and the entire statement will be executed.

   Several statements may be contained in a single input message. Each statement should be followed by a semicolon. The final semicolon is optional.

3. When Data Dictionary statements are entered from cards or disk, only the first 72 characters of each record may contain commands. The remaining columns are reserved for sequence numbers or comments. A command may extend over as many records as desired; however, no word may be split between two records. Every command must be terminated by a semicolon. Several commands may be contained on a single input record provided each is delimited by a semicolon.

4. A percent sign (%) denotes a comment, and anything between the percent sign and the end of the record is ignored. (except when the percent sign is enclosed in quotation marks, such as "ABC%DE").

5. Words, names and numbers must be separated by at least one space, or by punctuation or delimiter characters (such as ; , / ( ) *).

6. Names of DASDL identifiers, and within file titles may contain hyphens (-) and underscores (_), but not at the beginning or end of a name.

7. File identifiers containing any characters other than letters, digits, hyphens or underscores must be enclosed in quotation marks, (").

8. There is no restriction on the use of keywords such as INSERT, PROGRAM, DATABASE, and SOURCES as DASDL or file identifiers; the syntax causes no ambiguity.

9. If the first command is SYNTAX; the remaining commands are checked for syntax but are not executed. This allows card or disk files to be checked prior to actual use.

10. Many keywords may be abbreviated; refer to the syntax diagrams for details.

Each command is printed or displayed as soon as it has been read, and an explanatory message is then given. If the commands came from a card or disk file, the remaining commands will not be executed, but will be checked for syntax. However, if the commands came from a terminal a further explanatory message will tell the user what was expected, and the user may retype the word in error, followed by the rest of command. Typing a semicolon instead will abandon that command, if the user wishes to start it again from the beginning.

2.5 Running from Batch and CANDE

To run DDUPDATE or DDREPORT in batch mode, a card deck or JOBSYMBOL file must be made, containing WFL statements similar to those below:

```
? BEGIN JOB EXAMPLE;
? CLASS = 50;
? USER = BILL/PASSWORD;
? RUN DDREPORT;
? DATA CARD;
  SELECT DEBTORS DATABASE;
  OPTIONS PRINTER;
  DISPLAY XREF;
  QUIT;
? END JOB
```

(For a card deck, ? denotes an invalid punch code, such as a 1 2 3 multi punch in column 1). By default the commands are expected to be in a card file, with internal name CARD, and the output will be a printer file, with internal name LINE.

To run DDUPDATE or DDREPORT in interactive mode, with CANDE, first log on to CANDE, and then type:

```
RUN $DDREPORT;
```

(The $ tells CANDE to look for a program called DDREPORT, instead of OBJECT/DDREPORT as it normally would).

By default the commands will be expected to come from the terminal using a file with internal name CARD, and output will be displayed on the same terminal, using a file with internal name LINE. However, the user may override these defaults for the CARD and LINE files by using FILE statements in WFL or CANDE. For example, if the commands for run were in a disk file called DDCOMMANDS011079, and the output to be written to a disk file called DDOUTPUT022879, then the WFL "RUN DDREPORT;" statement, or the CANDE "RUN $DDREPORT;" would be followed by the following file statements:

```
FILE CARD (KIND=DISK, TITLE=DDCOMMANDS011079);
FILE LINE (KIND=DISK, TITLE=DDOUTPUT0228979);
```

The DDUPDATE and DDREPORT programs will try to handle any meaningful assignment of the CARD and LINE files.

2.6 Handling of Faults During Execution of Data Dictionary Programs

Care has been taken to prevent system or program faults from leaving the Data Dictionary data base in an inconsistent state. The data base is audited, and each insertion, deletion or replacement of a program or data base is treated as one transaction. Thus if the program is stopped for some reason in the middle of such an operation, the data base will, at the first opportunity, restore the data base to its state before the operation was commenced.

2.7 Security of Files

The DDUPDATE program has no special security privileges, so that if it needs to access files belonging to a usercode other than the one it is running under, that file must either be a PUBLIC file, or else DDUPDATE must be run by a privileged usercode.

3. ENTERING DATA AND TRANSACTIONS BASES

3.1 Overview of Data and Transaction Bases

The layout of data fields and their relationships within a data base are described by a source "program" written in the DASDL language. The DASDL compiler reads this and creates a description file, containing all the information in a form which can be accessed by the compilers when compiling the access routines for the data base, and application programs which access the data base.

If the data base name is ABCD, then the DASDL compiler will create a description file with the title DESCRIPTION/ABCD. By running the DASDL compiler with the UPDATE option, an existing data base description can be altered, and a new different description file created. To distinguish these various updates of the original base, each is assigned an update level, which begins at 1 for the original description file, and increases by 1 for each update. In addition, each description file has a creation timestamp, giving the date and time when the initial description file was created, and an update timestamp showing when the particular update level was created.

Similar considerations apply to transaction bases, except that the description file for a transaction base called ABCD would be called ABCD/TRDESCRIPTION.

The Data dictionary obtains its information about a base from the description file for that base.

A data base may also have logical data bases associated with it. Each logical data base may access only a portion of the full (physical) data base.

Similarly, a transaction base may have transaction subbases associated with it, each of which accesses only a portion of the full (physical) transaction base.

Each description file contains information about the physical base and all of its logical bases (or subbases), if any.

3.2 Naming Conventions for the Data and Transaction Bases

The Data Dictionary identifies a data or transaction base by its name and update level, and also by the usercode and family of its description file. These are all saved for each base entered into the Data Dictionary.

The Data Dictionary will not allow two data bases or two transaction ba bases with the same name, update level, usercode and family. Any request to enter a base which has the same names etc. as a base already in the Data Dictionary will be disallowed.

Every base is given a unique number when entered into the Data Dictionary, and this number may be used to specify a base uniquely. When bases are mentioned in Data Dictionary commands the name of the base and its type (whether a data base or transaction base) must be given. If the Data Dictionary contains more than one update level of this base, the update level must be specified also. If there is more than one base with the same name and update level (perhaps with different usercodes or families), this unique number must also be specified.

3.3 The INSERT Command for Data and Transaction Bases

Data and transaction bases are inserted into the Data Dictionary using the INSERT command, which specifies the type and name of the base to be inserted. One or more bases of the same type may be specified in the same command. For example, the two commands:

```
INSERT DATABASE DEBTORS;
INSERT TRBASES INVENTORY, SALES;
```

insert a database called DEBTORS and transaction bases INVENTORY and SALES into the Data Dictionary. Any logical data bases of DEBTORS, and transaction subbases of INVENTORY and SALES are also entered.

The INSERT command looks for description files DESCRIPTION/DEBTORS, and for the INVENTORY/TRDESCRIPTION and SALES/TRDESCRIPTION (according to the usual file naming conventions).

Regardless of the title of the description file, the database name is taken from the contents of the description file.

The usercode and family of the description file may be explicitly given, as for example:

```
INSERT DATABASE (BILL)DEBTORS ON ACCOUNTS;

INSERT TRBASE *INVENTORY;

INSERT TRBASE SALES ON RETAIL;
```

These description files should be public files, or else DDUPDATE should be run with a privileged usercode if files in other usercodes are to be read.

No insertion will be done if the Data Dictionary already contains a base of the same type, with the same name and update level, and entered from a description file with the same usercode and family as that found from the information in the INSERT command.

The keyword INSERT may be abbreviated to IN, and DATABASE or DATABASES to DA or DB, AND TRBASE or TRBASES to TR.

3.4 The DELETE Command for Data and Transaction Bases

Data and transaction bases are deleted from the Data Dictionary using the DELETE command, which specifies the type and name of the base to be deleted. One or more bases of the same type may be given in the same command. For example, the two commands

```
DELETE DATABASE DEBTORS;
DELETE TRBASES INVENTORY, SALES;
```

delete the data base DEBTORS, and the transaction bases INVENTORY and SALES from the Data Dictionary. Any logical bases of DEBTORS and transaction subbases of INVENTORY and SALES are also deleted.

The names given in the DELETE commands must be the actual names of the bases, not titles of description files, and must uniquely specify each base to be deleted.
The update level, and unique number of a base may also be specified. For example, if there were two update levels 3 and 5 of DEBTORS in the Data Dictionary, then the DELETE command could be

```
DELETE DATABASE DEBTORS LEVEL 3;
```

```
                 or
     DELETE DATABASE DEBTORS LEVEL 5;
```

to show which one was to be deleted.

If there are two bases with the same names and update levels, but with different usercodes or families, the unique number may be specified. Thus the following commands could be used

```
     DELETE DATABASE DEBTORS (33);
                 or
     DELETE DATABASE DEBTORS LEVEL 5 (41);
```

to show that the bases with unique numbers 33 and 41 were to be deleted. (The unique numbers of all bases in the Data Dictionary can be found using the DISPLAY SOURCES command, described in section 6.4).

The DELETE command will terminate if there is more than one base matching a given specification, or if no base matching that specification can be found.

The keyword DELETE may be abbreviated to DEL.

3.5 The REPLACE Command for Data and Transaction Bases

A REPLACE command is equivalent to a DELETE followed by an INSERT, and the unique number of the old base is assigned to the new one.

If the old base is not specified uniquely, or the new base has the same name, update level, usercode and family as a base of the same type already in the Data Dictionary then no deletion or insertion will be done.

The rules for specifying the new and old base are just those given for the INSERT and DELETE commands respectively.

Examples of REPLACE commands are:

```
     REPLACE TRBASE SALES BY SALES;
     REPLACE TRBASE SALES LEVEL 7 BY SALES;
     REPLACE TRBASE SALES BY SALES ON RETAIL;
     REPLACE TRBASE SALES (31) BY (LA) SALES;
```

The keyword REPLACE may be abbreviated to REP.

3.6 Syntax for Data and Transaction Base Commands

<base insert command>

```
                                   |<-------- , --------|
                                   |                    |
-- INSERT --- DB ------------<base insert name>--- ; -----------------|
   --      |             |
           |- DATABASE --|
              --
           |- DATABASES -|
              --
           |- TRBASE ----|
              --
           |- TRBASES ---|
              --
```

<base delete command>

```
                                   |<-------- , --------|
                                   |                    |
-- DELETE --- DB ------------<base delete name>--- ; -----------------|
   ---     |             |
           |- DATABASE --|
              --
           |- DATABASES -|
              --
           |- TRBASE ----|
              --
           |- TRBASES ---|
              --
```

```
<base replace command>

                              |<---------- , --------|
                              |                      |
-- REPLACE --- DB ------------<base replace name>--- ; ----------------|
   ---      |              |
            |- DATABASE --|
            |  --          |
            |- DATABASES -|
            |  --          |
            |- TRBASE ----|
            |  --          |
            |- TRBASES ---|
               --


<base insert name>

--<file title>---------------------------------------------------------|


<base delete name>

--<name>---------------------------------------------------------------|
       |                                                       |
       | |<-------------------------------------|               |
       | |                                      |               |
       |-----/1\- ( --<unique integer>-- ) -----|
             |                                 |
             |-/1\- LEVEL --<update level>----|
                    ---


<base replace name>

--<base delete name>-- BY --<base insert name>-------------------------|


<unique integer>

--<integer with no more than 11 digits>--------------------------------|


<update level>

--<integer with no more than 1! digits>--------------------------------|
```

4. COMPILING AND ENTERING PROGRAMS INTO THE DATA DICTIONARY

4.1 Overview of Saving the Usage Data

Programs whose usage of data and transaction bases is to be entered into the Data Dictionary must be compiled with the DATADICTINFO compiler option set. This causes the compiler to place this usage information into the object code file.

Following such a compilation, the Data Dictionary can be instructed to extract that usage from the code file and enter it into the Data Dictionary, which can then prepare various reports which collate this usage. The usage data for a program may also be deleted or replaced.

This usage information shows which bases structures and items were invoked by the program, and whether they appeared in source language statements in a read or write context.

4.2 Saving Usage Data When Compiling

To save data and transaction base usage data in the object code file, the compiler option DATADICTINFO must be set. This option must be set before the first source language statement, and cannot be set, reset or popped within the program.

The usage data shows which bases, structures and items were invoked by the program, and also whether they were used in source language statments in a read context, a write context, or both. For data bases use in OPEN INQUIRY or CLOSE statements is regarded as a read context, and OPEN UPDATE as a write context. For data sets and sets, use in a FIND or LOCK statement is a read context, while DELETE, CREATE and STORE are write contents. Statements which alter manual sets are a write context. Statements which read a data item are read contents, whereas statements which alter data items are write contents.

This data does not purport to show which data sets, sets and data items are actually updated or read by executing the program but only the context in which they appear in source language programs.

4.3 Program Naming Conventions

Programs within the Data Dictionary are identified by a short name up to 17 characters long, and by a unique number which must be specified only if the short name is not unique.

The short name may be explicitly specified when the program is entered, and otherwise will be the rightmost identifier of the object code file title. (Short names are used because lengthy file titles would be inconvenient).

There may be more than one program with the same short name in the Data Dictionary, so that different versions of a program may be kept in it at the same time. Every program, however, will have a different unique number. The DISPLAY SOURCES command, described in section 6.4, will print a list of all programs in the Data Dictionary, showing their short name, unique number, the full title of their object code file, their compilation date and time, and the date and time they were entered into the Data Dictionary. Thus it should be possible to identify every program from this information.

The Data Dictionary will only enter usage data from object code files which are executable, and not from object code files compiled with the LIBRARY or SEPARATE options. If any procedures were compiled with the DATADICTINFO option reset, a warning of possibly incomplete usage data will be given whenever that program is entered.

4.4 The INSERT PROGRAM Command

The INSERT PROGRAM command enters usage information from object code files and saves it in the Data Dictionary. For example the commands

```
INSERT PROGRAM OBJECT/STATETAX;
INSERT PROGRAMS PAYROLL, PARTS/MOD11;
```

insert usage information from the object code files with titles OBJECT/STATETAX, PAYROLL and PARTS/MOD11. (The keywords PROGRAM and PROGRAMS are equivalent). These files must be object code files containing Data Dictionary usage information, and the files must of course be available when the INSERT PROGRAM command is executed.

By default, the short name assigned to each program will be the rightmost identifier of its file title, which would be STATETAX, PAYROLL and MODII for the commands above. Each program will also be assigned a unique number. Thus it is permissible (but possibly confusing to the user) to have more than one program with the same short name, as they will have different unique numbers.

A specific short name can be assigned to a program as in the following examples:

```
INSERT PROGRAM PAYROLL = PAY79;
INSERT PROGRAM PARTS/MOD11 = PARTS11;
```

so that the program entered from the code file PAYROLL will have the short name PAY79, and that from the code file PARTS/MOD11 will have the short name PARTS11.

These short names must begin with a letter and comprise up to 17 letters, digits, hyphens or underscores (hyphens or underscores must not be the first or last characters).

If the object code file belongs to a usercode different from that used to run DDUPDATE, or to a different family, the usercode and family may be specified along with the file title, as for example:

```
INSERT PROGRAM (NY)PAYROLL;
INSERT PROGRAM PAYROLL ON PERSONNEL;
INSERT PROGRAM *PAYROLL ON PACK;
```

(The * specified that the file does not belong to a particular usercode).
If an object code file does not belong to the usercode which runs DDUPDATE, the normal file security restrictions will apply. Thus the file security must be PUBLIC (or CLASSA) or DDUPDATE must be run by a privileged usercode.

The file titles must conform to the usual rules for file titles. However, identifiers containing hyphens (-) or underscores (_) need not to be quoted (unless the hyphen or underscore is the first or last character). The following are all valid Data Dictionary commands:

```
INSERT PROGRAM PAY-ROLL;
INSERT PROGRAM FICA-DEDUCTIONS;
INSERT PROGRAM "5%PAYRISE";
```

All of the data bases invoked by a program must already have been entered into the Data Dictionary by INSERT DATABASE commands before the program is inserted. If a program being inserted invokes a data base which is not in the Data Dictionary, the command is terminated and any data already entered from that program is deleted from the Data Dictionary.

The description file current when the program was compiled must be the same one from which an INSERT DATABASE or REPLACE DATABASE command entered the base information into the Data Dictionary. (Their update timestamps are compared to ensure they are the same update level of the same data base). If there are several different update levels of a base in the Data Dictionary, the correct one will be found and used automatically.

The keyword INSERT may be abbreviated to IN, and PROGRAM or PROGRAMS may be abbreviated to PR.

4.5 The DELETE PROGRAM Command

The DELETE PROGRAM command deletes usage information of programs from the Data Dictionary. For example the commands

```
DELETE PROGRAMS STATETAX, PAYROLL;
DELETE PROGRAM MOD11;
```

will delete from the Data Dictionary all usage information from the programs with short names STATETAX, PAYROLL and MOD11 (assuming that there is only one program with each of these short names).

If there is more than one program with the same short name, the unique number must also be given to specify exactly which is to be deleted, as for example:

```
DELETE PROGRAM STATETAX (314);
DELETE PROGRAM MOD11 (19);
```

which delete the programs with short name STATETAX and inique number 314, and with short name MOD11 and unique number 19.

If a program specification is not unique, nothing at all will be deleted.

A DISPLAY SOURCES command will list all of the programs in the Data Dictionary with their unique numbers.

The keyword DELETE may be abbreviated to DEL, and PROGRAM or PROGRAMS (which are equivalent) to PR.

4.6 The REPLACE PROGRAM Command

The REPLACE PROGRAM Command is equivalent to a deletion followed by an insertion. For example:

```
REPLACE PROGRAM PAY79 BY OBJECT/PAYROLL = PAY80;
```

specifies that the program with short name PAY79 is to be deleted and that in the code file OBJECT/PAYROLL be inserted with a short name of PAY80.

No deletion or insertion is done unless the program to be deleted is specified uniquely, and the program to be inserted is available, and contains Data Dictionary usage information.

The program to be deleted is specified in exactly the same way as for the DELETE PROGRAMS command, and the program to be inserted is described in the same way as for the INSERT PROGRAMS command.

The key word REPLACE may be abbreviated to REP, and PROGRAM or PROGRAMS to PR.

4.7 Syntax for Program Commands

```
<program insert command>

                              |<---------- , ---------|
                              |                       |
-- INSERT --- PROGRAM -------<program insert name>--- ; ---------------|
   --       |  --        |
            |- PROGRAMS -|
               --


<program delete command>

                              |<------------- , -------------|
                              |                              |
-- DELETE --- PROGRAM ------<program program delete name>--- ; --------|
   ---      |  --        |
            |- PROGRAMS -|
               --


<program replace command>

                               |<---------- , ----------|
                               |                        |
-- REPLACE --- PROGRAM ------<program replace name>--- ; --------------|
   ---       |  --        |
             |- PROGRAMS -|
                --


<program insert name>

--<file title>---------------------------------------------------------|
                 |                 |
                 |- = --<short name>-|


<program delete name>

--<name>---------------------------------------------------------------|
             |                                |
             |- ( --<unique integer>-- ) -|


<program replace name>

--<program delete name>-- BY --<program insert name>-------------------|


<short name>

--<letter>-------------------------------------------------------------|
              |                                          |
              | |<-/15\----------------------------------|   |
              | |                                        |   |
              |---<letter or digit or hyphen or underscore>---|


<unique integer>

--<integer with no more than 11 digits>--------------------------------|
```

5. PREPARING AND ENTERING DOCUMENTATION TEXT

5.1 Overview of Documentation Text

Files of free format text which contain documentation for items in data and transaction bases can be indexed by the Data Dictionary, which can later retrieve the text and include it in various reports. The text itself is not stored in the Data Dictionary, but only its location within the file indexed.

This documentation text may be inserted independently of information entered from description files by INSERT DATABASE or INSERT TRBASE commands, and no association is made when either documentation text or description file data is entered. For example, a user may wish to index the documentation text before the description file has been generated.

The association between description file entities and their corresponding documentation text is made only when a report is being compiled. The rules by which this association is made are given in section 5.2, and the format of documentation text files is given in section 5.3.

5.2 Naming Conventions for Documentation Text

All documentation text belongs to a particular documentation base, and then to a particular item within that base, so whenever documentation text is required for a report, the documentation base must first be located, and then the correct item found within that base.

Documentation bases must always have their name and type (whether a data base or a transaction base) specified. In addition, the update level, and usercode and family of the corresponding description file may optionally be specified also. Whenever a documentation base is required for a particular description file base, the Data Dictionary looks first for a documentation base with the correct type and name. If there are several documentation bases with different update levels, it will choose the documentation base whose update level is equal to or less than the description file update level. For example, if there were three documentation data bases called SALES in the Data Dictionary, with update levels 1,5 and 7, then the following associations would be made with description file data bases called SALES.

Description file base Update Levels	Corresponding Documentation Base Update level
1,2,3,4	1
5,6	5
7 and greater	7

If the documentation base is still not unique, the usercode and family are then used to decide which base to use.

To avoid any ambiguity when searching for a documentation base to match a description file base, the Data Dictionary ensures that there are never two or more documentation bases which could match any one description file base. It ensures that all documentation bases differ in at least one of type (data or transaction base), name, update level, usercode or family. Properties for two bases are not considered different in this context if that property is not specified for one of the bases. For example, the following documentation data bases are all different in this context.

```
     SALES  LEVEL  3  ON RETAIL
     SALES  LEVEL  4  ON RETAIL
     SALES  LEVEL  3  ON BUYING
 (LA)SALES  LEVEL  1
(SFO)SALES            ON CREDIT
```

However, the following documentation data bases could not be inserted into the Data Dictionary, if it already held the ones above.

```
SALES
```

because it does not differ from any of the above.

```
(LA)SALES  LEVEL 4
```

because it does not differ from SALES LEVEL 4 ON RETAIL

```
(SFO)SALES LEVEL 6
```

because it does not differ from (SFO)SALES ON CREDIT.

A description file data base
    (LA)SALES LEVEL 1 ON DISK
would be matched by the documentation data base
    (LA)SALES LEVEL 1.

The description file data base
    (SD)SALES LEVEL 3 ON RETAIL
would be matched by the documentation data base
    SALES LEVEL 3 ON RETAIL.

5.3 Preparation of Documentation Text Files

A documentation text file must contain text for at least one data or transaction base, and its associated logical data bases or transaction subbases. All text for a base (and its logical bases) must be in the same file.

Text records may be of any length, but the Data Dictionary will show only the first 72 characters in its reports. CANDE files of type SEQDATA (SEQ) would be suitable. The documentation text for each item of the base may be in any format, and may comprise any number of records (including zero records). The item to which the text refers is identified by a heading record which precedes the text records for the item. These heading records have a unique character ($ by default) as their first character. A "level number" similar to those in COBOL or PL/I shows how items are owned by other items. Only the first 72 characters of heading records are significant.

The following example shows a text file for a data base SALES.

```
$ SALES DATABASE
    text for the data base

$ 1 DEPARTMENTS
    text for the data set DEPARTMENTS

$ 2 DEPT-NAME
    text for the data item DEPT-NAME

$ 2 DEPT-NUMBER
    text for the data item DEP-NUMBER

$ 2 CLOSING-DATE
    text for the group item CLOSING-DATE

$ 3 CLOSING-MONTH
    text for the item CLOSING-MONTH

$ 3 CLOSING-DAY
    text for the item CLOSING-DAY

$ 3 CLOSING-YEAR
    text for the item CLOSING-YEAR

$ 2 SALES-TO-CLOSING
    text for item SALES-TO-CLOSING

$ 2 SALES-BY-DEPT-NUM
    text for the set SALES-BY-DEPT-NUM which
    spans the data set DEPARTMENTS
```

The DASDL to describe these items may be:

```
     DEPARTMENTS DATA SET

          (DEPT-NAME ALPHA(30);
          DEPT-NUMBER NUMBER (6);
          CLOSING-DATE GROUP
               (CLOSING-MONTH NUMBER(2);
               CLOSING-DAY    NUMBER(2);
               CLOSING-YEAR   NUMBER(2);
               )
          SALES-TO-CLOSING NUMBER (6,2);
          );
     SALES-BY-DEPT-NUM SET OF DEPARTMENTS
          KEY IS DEPT-NUMBER;
```

The heading for the base must specify at least the base name and type. The usercode, update level and family of the description file may also be specified, as for example:

```
     $ (LA) SALES DATABASE LEVEL 3 ON RETAIL
```

If heading will not fit on one record it may be continued on successive records, by ending the unfinished heading with a hyphen, (preceded by at least one space), as for example:

```
     $ (LA) SALES DATABASE -
     $ LEVEL 3 ON RETAIL
```

The "level numbers" on the headings for each item in the base (not to be confused with the update level of a base) show that DEPARTMENTS owns DEPT-NAME, DEPT-NUMBER, CLOSING-DATE, SALE-TO-CLOSING and SALES-BY-DEPT-NUM, and that CLOSING-DATE owns CLOSING-MONTH, CLOSING-DAY and CLOSING-YEAR.

They must follow the same rules as for COBOL level numbers.

All sets, regardless of whether they are disjoint or embedded, or of where they are embedded, are assumed to be owned by the data set they span. GROUP items, and FIELD items which contain BOOLEANS, are assumed to own the items they contain.

Global data items are assumed to be owned by the global data set, and subformats by the format which contains them, and data items within a subformat are assumed to be owned by that subformat.

A documentation text file need not contain entries for all the items in the description file base. However, if an item is included, there must also be entries for every item which owns it. The Data Dictionary will not find a text item unless all of its owners have also been given.

The names for each item must be those given in the DASDL description of the data base (or TFL for a transaction base).

Anything following a percent sign (%) on a heading record is ignored, and treated as a comment.

Documentation text for logical data bases or transaction subbases must immediately follow that for the physical base. For example, if SALES had two logical data bases, SALES-1 and SALES-2, the text in the previous example could be followed by

```
$ SALES-1 OF SALES
    text for SALES-1

$ 1 DEPT-REMAP-1
    text for DEPT-REMAP-1

$ 2 DEPT-NUMBER
    text for DEPT-NUMBER

$ 2 SALES-TO-CLOSING
    text for SALES-TO-CLOSING

$ SALES-2 OF SALES
    text for SALES-2
```

Global data items within a logical data base are assumed to be owned by a global data set, with the name used in the DASDL description to include the global data in the logical data base.

5.4 The INSERT TEXT Command

Documentation text is indexed by the Data Dictionary, using an INSERT TEXT command. For example:

```
INSERT TEXT SALES/DOCTEXT;
```

indexes the documentation text given in the file with title SALES/DOCTEXT. This file title may include a usercode and family also, as for example:

```
INSERT TEXT (LA)SALES/DOCTEXT;
INSERT TEXT (LA)STAFF ON PERSONS;
```

If heading records are denoted by a character other than $, that character may be specified in the command, for example:

```
INSERT TEXT SALES/DOCTEXT CHAR #;
```

which states that heading records in that file begin with the character #. All headings in a text file must begin with the same character. If no character is specified, it will be $ by default.

The Data Dictionary will not use text in reports if the text file has been altered in any way since it was indexed, to prevent incorrect text being displayed if the file indexed has been replaced by an old or new file with the same title. (A file is regarded as having been altered if its title is changed.)

The INSERT TEXT command indexes all bases in the given file, including any logical data bases or transaction subbases.

5.5 The DELETE TEXT Command

Documentation text previously indexed by the Data Dictionary is deleted using the DELETE TEXT command. For example:

```
DELETE TEXT SALES/DOCTEXT;
```

deletes from the Data Dictionary all knowledge of the text for the bases in the file SALES/DOCTEXT. Usually the file specified in a DELETE TEXT command will be the same text file specified in a previous INSERT TEXT command.

Only headings for physical data bases or transaction bases are significant for a DELETE TEXT command. All text items in those bases, and in all of their logical data bases or transaction subbases are deleted. It is immaterial which items are given in the text file.

No deletion will occur unless the base is specified in exactly the same way as when it was inserted. The usercode, update level and family must each have been either not specified, or have been specified and be the same for deletion as they were for insertion. (If the same text file is specified for both insertion and deletion, the specifications will always match.)

The file specified in a DELETE TEXT command need not be the same one specified in the INSERT TEXT command.

5.6 The REPLACE TEXT Command

The REPLACE TEXT command is equivalent to a DELETE TEXT followed by an INSERT TEXT command. For example:

```
REPLACE TEXT SALES/DOCTEXT;
```

causes the file SALES/DOCTEXT to be scanned, and whenever a heading for a physical data or transaction base is found, that base together with all its items and logical data bases or transaction subbases is deleted. That base is then indexed again, together with all of its items and logical or subbases.

The REPLACE TEXT command would commonly be used whenever a documentation text file had been updated.

Remember that the Data Dictionary will not use text from a documentation text file which has been modified in any way since it was indexed by the Data Dictionary.

5.7 Syntax for Text Commands

```
<text command>

---- INSERT ---- TEXT --<file title>------------------------------ ; --|
   |  --      |                     |                          |
   |- DELETE --|                     |- CHARACTER --<character>-|
   |  ---     |                          ----
   |- REPLACE -|
      ---
```

5.8 Syntax for Documentation Text Files

```
<documentation text file>

  |<-----------------------------------------------|
  |                                                |
----<physical base text>------------------------------------------------|
                             |                                    |
                             |  |<--------------------|  |
                             |  |                     |  |
                             |---<logical base text>---|


<physical base text>

--<physical base heading>------------------------------------------------>
                               |          |
                               |-<text>-|

>-----------------------------------------------------------------------|
  |                                           |
  |  |<-----------------------------|  |
  |  |                              |  |
  |---<entity heading>--------------|
                            |        |
                            |-<text>-|
```

<logical base text>

```
--<logical base heading>----------------------------------------------->
                         |        |
                         |-<text>-|

>----------------------------------------------------------------------|
  |                                 |
  | |<----------------------------| |
  | |                             | |
  |---<entity heading>--------------|
                       |        |
                       |-<text>-|
```

<physical base heading>

```
-- $ --<physical base specification>-----------------------------------|
```

<physical base specification>

```
--------------------------<physical base name>--- DB ------------------>
  |                      |                      |            |
  |- * ------------------|                      |- DATABASE -|
  |                      |                      |  --        |
  |- ( --<usercode>-- ) -|                      |- TRBASE ---|
                                                   --
>----------------------------------------------------------------------|
  |                                 |
  | |<----------------------------| |
  | |                             | |
  |-----/1\- LEVEL --<integer>------|
      |      ---                |
      |-/1\- ON --<family name>-|
      |                         |
      |-/1\- OF --<host name>---|
```

<entity heading>

```
-- $ --<integer>--<entity name>----------------------------------------|
```

<logical base heading>

```
-- $ --<logical base name>-- OF --<physical base name>-----------------|
```

6. GENERATING DATA DICITIONARY REPORTS

6.1 Introduction to Data Dictionary Reports

The information about data and transaction bases, usage by programs, and documentation text, entered into the Data Dictionary by the various INSERT and REPLACE statements, can be presented in the various reports described in this section.

These reports can be generated by both the DDUPDATE and DDREPORT programs, although the DDREPORT program, which cannot alter the Data Dictionary, would normally be used.

The reports can be either displayed on a terminal or printed, depending on the option selected by the user as described in section 6.2.

For most reports, the data or transaction base, to be reported on must first be selected by a SELECT command, as described in section 6.3. The report is then generated by a DISPLAY command which specifies the particular report required, and in same cases gives details of a particular item to be reported on.

6.2 OPTION Command

The OPTION command specifies whether the reports are to be displayed on a terminal, or printed.

```
OPTION TERMINAL;
```

specifies that reports are to be displayed on a terminal. It is ignored for batch jobs.

The number of lines on the terminal screen may be specified with the PAGE option, as for example

```
OPTION TERMINAL PAGE = 20;
```

which stops output to the terminal every 20 lines. Output resumes when a blank line is sent from the terminal. If the first character sent from the terminal is not a space, the report will be terminated.

Setting PAGE = 0 results in continuous output.

```
OPTION PRINTER;
```

specifies that reports are to be printed.

```
OPTION;
```

causes the current option to be displayed.

The initial setting of this option is PRINTER for a batch run, and TERMINAL for a CANDE run. Options continue until changed by an OPTION command.

The keyword OPTION (or OPTIONS) may be abbreviated to OPT, TERMINAL may be abbreviated to T, and PRINTER to P.

<options command>

```
---- OPTION ------------------------------------------- ; -----------|
  |  ---      | |                                    |
  |- OPTIONS -| |- PRINTER --------------------------|
     ---        |  -                                 |
                |- TERMINAL -------------------------|
                   -        |                        |
                            |- PAGE -- = --<integer>-|
```

6.3 Selecting a Base on which to Report

The SELECT command specifies which data or transaction base will be reported on by the following DISPLAY commands. The information given by a SELECT command completely replaces all information given by a previous SELECT, and the new information remains in force until another SELECT command is given.

In its simplest form, the SELECT just gives the name and type of a base. For example,
SELECT SALES DATABASES;
selects a physical data base called SALES.
SELECT FOOD OF SALES DATABASE;
selects a logical data base FOOD of the physical data base SALES.

Similarly,
SELECT LOANS TRBASE;
selects a transaction base called LOANS, and
SELECT RV OF LOANS TRBASE;
selects a subbase RV of the LOANS transaction base.

If a base entered from a description file (by INSERT DATABASE or INSERT TRBASE commands) needs its update level or unique number to be specified also, they can be given, as for example:

```
SELECT SALES DATABASE LEVEL 3;
SELECT SALES DATABASE LEVEL 4 (39);
SELECT RV OF LOANS TRBASE (91);
SELECT LOANS TRBASE LEVEL 9;
```

If a documentation base is being selected, its usercode and family may be specified, as for example:

```
SELECT (LA)SALES DATABASE ON RETAIL;
SELECT RV OF LOANS TRBASE ON CREDIT;
```

In fact, thee SELECT may include all combinations of th above, as for example

```
SELECT (LA)SALES DATABASE LEVEL 3;
SELECT LOANS TRBASE (49) ON CREDIT;
```

However, only the base name, type, update level and unique number are used to select bases entered from description files, and only the usercode, base name, type, update level and family are used to select documentation text bases.

A SELECT command does not actually try to locate a base; the information given is just stored, and used by the DISPLAY commands when they need to locate a base.

The command SELECT; causes the current information to be displayed.

```
<select command>

-- SELECT ------------------------------------------------------------>
   ---        |                      |  |                            |
              |- ( --<usercode>-- ) -|  |-<logical base name>-- OF -|

>-<physical base name>--- DATABASE ------------------------------------>
                          |--         |
                          |- DB ------|
                          |           |
                          |- TRBASE --|
                             --
>------------------------------------------- ; ----------------------|
   |                                     |
   | |<--------------------------------| |
   | |                                 | |
   |-----/1\- LEVEL --<integer>------------|
          |         ---                 |
          |-/1\- ( --<unique integer>-- ) -|
          |                             |
          |-/1\- ON --<family name>--------|
          |                             |
          |-/1\- OF --<host name>----------|
```

6.4 SOURCES Report

The DISPLAY SOURCES command generates a report showing all description file bases (entered by INSERT DATABASE and INSERT TRBASE commands), programs (entered by INSERT PROGRAM commands) and documentation bases (entered by INSERT TEXT commands) which are currently in the Data Dictionary.

It does not use any information from SELECT commands.

```
<display sources command>

-- DISPLAY -- SOURCES -- ; -------------------------------------------|
   ---        --
```

6.5 Cross Reference Reports

The DISPLAY XREF command generates a report showing which programs reference each item of a base, any internal name by which that item was known within the program, and whether the usage was read or write. It has two forms, which report on all items in a base, and on just one item.

DISPLAY XREF; reports on all items within the selected base. Items are listed in nearly the same order as they were declared in DASDL or TFL.

DISPLAY XREF MONTHLY-REPAYMENT; reports just on the item MONTHLY-REPAYMENT within the selected base. The item may be qualified, as for example

DISPLAY XREF INTEREST OF LOAN-DETAILS OF LOAN-APPLICATION;

(Remember that sets are assumed to be owned by the data set they span, and that group items may be used as qualifiers in the Data Dictionary. Also, global data items are assumed to belong to the global data set).

All names used in PL/I programs will have underscores (_) replaced by hyphens (-).

<display xref command>

```
-- DISPLAY -- XREF --------------------- ; -------------------------|
   ---         -    |                    |
                    | |<----- OF ----|   |
                    | |              |   |
                    |---<identifier>---|
```

6.6 Usage Reports

The DISPLAY USAGE report has various forms which show, for a particular program, the various data and transaction bases used, or for a particular base the structures used, or all the items used. Another form shows which programs reference a given update level of the selected base.

The briefest usage report, lists the bases used by a program. For example:

    DISPLAY USAGE BY PROGRAM PAY;

lists all data and transaction bases referenced by the program whose short name is PAY. If the base was renamed within the program, this internal name will also be shown.

If there are several programs with the same short name, a unique number may also be given, as for example:

    DISPLAY USAGE BY PROGRAM PAY (49);

which reports on the program PAY with unique number 49.

Once the bases used by a program have been identified, more detailed usage reports may be generated showing which structures or items of one base are used by the program. For example, the commands

```
SELECT STAFF DATABASE;
DISPLAY USAGE OF STRUCTURES BY PROGRAM PAY;
```

show which data sets and sets of the STAFF data base are referenced by the PAY program. If they were renamed within the program, their internal names will also be shown.

A more detailed report, showing the use of all items (and data sets and sets) of the STAFF data base, by the PAY program would be obtained using the commands

```
SELECT STAFF DATABASE;
DISPLAY USAGE OF ITEMS BY PROGRAM PAY;
```

If necessary, the unique number of the program may be given also, as for example:

```
DISPLAY USAGE OF STRUCTURES BY PROGRAM PAY (216);
DISPLAY USAGE OF ITEMS BY PROGRAM PAY (324);
```

The programs which use a particular update level of the selected base can be listed as for example:

```
SELECT STAFF DATABASE;
DISPLAY USAGE OF LEVEL 3;
DISPLAY USAGE OF LEVEL 7;
```

These commands generate two reports, the first showing which programs reference update level 3 of the STAFF data base, and the second showing which programs reference update level 7 of the same base.

Any update level given in the SELECT command is ignored for the usage-of-level command, as the update level given in the usage command is always used. For example:

    SELECT STAFF DATABASE LEVEL 1;

```
DISPLAY USAGE OF LEVEL 3;
```

always reports on the usage of update level 3.

The following abbreviations may be used:

DIS	for	DISPLAY
US	for	USAGE
STR	for	STRUCTURES
LEV	for	LEVEL
PR	for	PROGRAM

<display base usage command>

```
-- DISPLAY -- USAGE ---------------------------------- PROGRAM ------>
   ---        --     |                       | |      |  --
                     |---------- ITEMS ------| |- BY -|
                     |      |  |             |
                     |- OF -|  |- STRUCTURES -|
                                  ---
>-<program short name>----------------------------- ; ---------------|
                       |                           |
                       |- ( --<unique number>-- ) -|
```

<display update level usage command>

```
-- DISPLAY -- USAGE ---------- LEVEL --<integer>-- ; ------------------|
   ---        --     |      |  ---
                     |- OF -|
```

6.7 Change Impact Report

The XREF and USAGE reports list references to items, without considering that different items may be related. However, the DISPLAY CHANGE IMPACT command lists not only those programs which reference the given item, but also those which reference subitems and owners of the given item, and also those which reference remaps of the given item, and the equivalent item in logical bases. Hence the change impact report attempts to identify all programs which would be affected by a change to the given item.

For example, the commands

```
SELECT STAFF DATABASE;
DISPLAY CHANGE IMPACT EMPLOYEE-NAME;
```

will generate a change impact report showing the effect of a change to the data item EMPLOYEE-NAME in the data base STAFF.

This change impact report will firstly list all subitems (if any) of EMPLOYEE-NAME, and then a list of all programs which reference EMPLOYEE-NAME or any of its subitems. If EMPLOYEE-NAME were deleted from the data base, or its size or other properties changed, logic changes may be required in those programs.

Next, all programs which reference owners of EMPLOYEE-NAME are listed. Although these programs may not specifically reference EMPLOYEE-NAME, they would probably need to be recompiled, as some properties of the owners may have changed.

A similar change impact report is then printed for each data item (in the STAFF data base or in one of its logical data bases) which remaps EMPLOYEE-NAME was on a remap, or in a logical data base).

Finally, a similar report is printed for every data item in logical data bases which is equivalent to EMPLOYEE-NAME. (This would not be done if EMPLOYEE-NAME was in a logical data base).

The item to be reported on may be qualified, as for example

```
DISPLAY CHANGE IMPACT ZIP-CODE OF ADDRESS OF EMPLOYEE-DATA;
```

Group items may be used as qualifiers within the Data Dictionary.

The following abbreviations are allowed:

DIS	for	DISPLAY
CH	for	CHANGE
IM	for	IMPACT

<display change impact command>

```
                                     |<----- OF ----|
                                     |              |
-- DISPLAY -- CHANGE -- IMPACT ---<identifier>--- ; -----------------|
   ---        --        --
```

6.8 Alphabetic Reports

The alphabetic reports list, in alphabetic order, all data and transaction bases, or all entities within a selected base. The command will list either description file bases, or documentation bases. Also, any documentation text for each entity can also be shown. For example:

```
DISPLAY ALPHABETIC;
```

displays an alphabetic list of all physical and logical data bases, and transaction and subbases, which have been extracted from description files.

Similarly,

```
DISPLAY ALPHABETIC DOCUMENTS;
```

displays an alphabetic list of all bases whose documentation text has been indexed by the Data Dictionary.

If the actual documentation text for each base is required, the commands would be

```
DISPLAY ALPHABETIC WITH TEXT;
DISPLAY ALPHABETIC DOCUMENTS WITH TEXT;
```

Alternatively, an alphabetic list of all entities within the selected base can be obtained from commands such as

```
SELECT STAFF DATABASE;
DISPLAY ALPHABETIC ITEMS;
DISPLAY ALPHABETIC DOCUMENT ITEMS;
```

The SELECT command just selects the base to be reported on, and would contain any of the specifications described in Section 6.3.

The first DISPLAY command generates a report showing all items in the DASDL description of the STAFF data base, while the second DISPLAY shows all items in the documentation text file for the STAFF data base.

Documentation text for each item would be given if WITH TEXT was added to each command, as for example:

```
DISPLAY ALPHABETIC ITEMS WITH TEXT;
DISPLAY ALPHABETIC DOCUMENT ITEMS WITH TEXT;
```

The following abbreviations are allowed:

DIS	for	DISPLAY
ALPHA	for	ALPHABETIC
DOC	for	DOCUMENT

<display description file alphabetic report command>

```
-- DISPLAY -- ALPHABETIC --------------------------------- ; ----------|
   ---        -----          |         | |                  |
                             |- ITEMS -| |---------- TEXT -|
                                         |         |
                                         |- WITH -|
```

<display documentation file alphabetic report>

```
-- DISPLAY -- ALPHABETIC -- DOCUMENT ---------------------------------->
   ---        -----         ---       |         | |                 |
                                      |- ITEMS -| |---------- TEXT -|
                                                  |         |
                                                  |- WITH -|

>- ; ---------------------------------------------------------------|
```

6.9 Attributes Reports

The attributes reports show the main attributes for either a given item, or all items owned by the given item. This information is extracted from DASDL or TFL description files, and will include any comment for each item in the DASDL or TFL description. Any documentation text for the item can also be included.

The commands

```
SELECT STAFF DATABASE;
DISPLAY ATTRIBUTES OF EMPLOYEE-NAME;
```

will display the attributes of the item EMPLOYEE-NAME in the STAFF data base.

The commands

```
SELECT STAFF DATABASE;
DISPLAY ATTRIBUTES OWNED BY EMPLOYEE-NAME;
```

will display the attributes of all items owned by EMPLOYEE-NAME in the STAFF data base. This form of the command is meaningful only for data sets and group items.

Adding WITH TEXT to the command will also display any documentation text for the items, as for example

```
DISPLAY ATTRIBUTES OF EMPLOYEE-NAME WITH TEXT;
DISPLAY ATTRIBUTES OWNED BY EMPLOYEE-NAME WITH TEXT;
```

The given item may be qualified for any of these commands. Names of data sets or groups may be used as qualifiers, as for example:

```
DISPLAY ATTRIBUTES OF ZIP-CODE OF ADDRESS OF
EMPLOYEE-DATA;
```

The following abbreviations are permitted:

```
DIS   for  DISPLAY
ATTR  for  ATTRIBUTES
```

<display attributes command>

```
                                              |<----- OF ----|
                                              |              |
-- DISPLAY -- ATTRIBUTES ---- OF --------------<identifier>--------------->
   ---        ----         |                |
                           |- OWNED -- BY -|
                              ---
>---------------------- ; ------------------------------------------------|
  |                  |
  |---------- TEXT -|
  |         |
  |- WITH -|
```

6.10 Differences Report

The differences report compares two different update levels of the same data or transaction base and reports the structures which differ. The two bases compared must have the same creation timestamps.
For example:

```
SELECT STAFF DATABASE LEVEL 3;
DISPLAY DIFFERENCES FROM LEVEL 4;
```

display the structures which differ between update levels 3 and 4 of the STAFF data base.

The following abbreviations are allowed:

```
DIS   for  DISPLAY
DIFF  for  DIFFERENCES
LEV   for  LEVEL
```

<display differences command>

```
-- DISPLAY -- DIFFERENCES -- FROM -- LEVEL --<integer>-- ; ------------|
   ---        ----                   ---
```

6.11 REPEAT Command

The command

REPEAT;

causes the previous DISPLAY command to be repeated, but only if that command completed successfully without error.

The REPEAT command would usually be given after an OPTION command to change the output medium, or after a SELECT command to alter the base reported on.

It may be abbreviated to R;

6.12 Reports during Updating of the Data Dictionary

If a DISPLAY command was given by one user while another was updating the Data Dictionary, the information given in the report could be incorrect.

Thus, if an update is in progress when a DISPLAY command is given, the user will be advised of the updating and the command will not be executed. If an update was in progress at the end of a DISPLAY command, or if an update occurred between the beginning and the end of the DISPLAY command, a warning will be displayed at the end of the report.

USER INTERFACE

TO

LIBRARIES

USER INTERFACE TO LIBRARIES

1. INTRODUCTION

Libraries provide a new process structuring technique. Specifically, a library is a program that provides a set of procedural 'entry points' which can be called by other programs. A call upon a procedure from a library is equivalent to a call upon a procedure within the user's program.

Collecting functions which are logically related into a library program makes programming easier and program structure more visible.

Any user can create and use libraries. Sharing of libraries is possible across multiple programs and multiple users.

Libraries offer the following improvements over binding:

1) Inter-language communication is much improved.

2) Sharing of globals is prevented.

3) Standard packages of routines (such as plotting routines and statistical routines) need not be copied into any user programs.

Libraries offer the following improvements over Installation Intrinsics:

1) A 'CI' by the system operator is not required.

2) There can be more than one version of a library in use at a time.

3) Libraries can be written in more languages.

4) Individual users can create their own libraries without needing special privileges.

5) A libraries can have its own global files, data bases, transaction bases and so on.

6) Libraries can contain initialization and termination code.

7) Library programs may themselves be user programs of other libraries.

2. FUNCTIONAL DESCRIPTION OF LIBRARIES

2.1 Library programs

A library program provides entry points for use by user programs. It is distinguished by the appearance of EXPORT declarations and FREEZE statements. (COBOL and COBOL74 differ from the other languages on this point; see sections 6 and 7.)

2.2 User programs

A user program uses entry points provided by library programs. A user program is distinguished by the appearance of LIBRARY declarations and entry point declarations for each of the declared libraries. (Again COBOL and COBOL74 differ on this point; see sections 6 and 7.)

It is permissible for a library program to also be a user of other libraries. The only restriction is that a chain of library linkages must never be circular. In other words, a library cannot reference itself, either directly or via some other chain of library references.

2.3 Library Directories and Templates

The information needed to make the linkage between the library program and the user program is contained in a pair of data structures called the 'directory' and the 'template' which are built by the compilers. These structures are used by the MCP to match the entry points in a library program with the corresponding entry points in the user program. The code file for a library program contains exactly one directory which describes all the entry points, their parameters, their result types and how to link to the entry points. The code file for a user program contains one template for each library declared in the program. The template describes the library attributes and contains the entry point descriptions for the procedures being used.

2.4 Library initiation

When a library entry point is first called, the user program is suspended, and the library (if it has not already been initiated) is automatically initiated by the MCP. The library runs as a normal program until it executes a FREEZE statement; this makes the EXPORTed entry points available. All of the entry points that the user program declared for the library are linked to the entry points provided by the library and the user program resumes execution.

The PERMANENT and TEMPORARY specifications on the FREEZE statement control the duration of the library. A permanent library remains available until it is DSed. A temporary library remains available as long as there are users of the library. A temporary library which is no longer in use will un-freeze and resume running as a regular program to 'end-of-task'. Once a library un-freezes, it may not execute a FREEZE statement in an attempt to become a library again.

Actually, any running program which executes a FREEZE statement becomes a library. To be useful in this case, the FREEZE statement should specify PERMANENT (the specification of TEMPORARY would cause the library program to immediately un-freeze, since the active user count would be zero).

Since a library program initially runs as a regular program, the flow of execution can be such that the execution of a FREEZE statement is conditional and can occur anywhere in the outer block of the program.

If a user program causes a library to be initiated, and this library does not execute a FREEZE statement (for whatever reason, perhaps it was not even a library program and thus had no FREEZE statement), the attempted linkage to the library entry points cannot be made and the user program will be DSed.

2.5 Linkage provisions

The linkage of entry points declared in a user program to entry points provided by a library program is done in one of three ways: directly, indirectly, or dynamically. The library program specifies the form of linkage; indirect and dynamic linkages allow linkage to be established to libraries other than the one specified by the user program. Other than by specifying the code file title of the library (via the library TITLE attribute) and for dynamic linkages by specifying the LIBPARAMETER library attribute, the user program has no control over which actual library invocation it gets linked to.

Direct linkage occurs when the library program actually contains the procedure that is named in the library's EXPORT declaration.

Indirect linkage occurs when the library program EXPORTs a procedure that it declared as an entry point of some other library. The MCP will then attempt to link the user program to this second library, which can provide the entry point directly, indirectly (again), or dynamically.

Dynamic linkage allows a library program to determine to which other library the actual linkage will be made. For an entry point that a library provides dynamically, the library also specifies a 'selection' procedure. This selection procedure is invoked by the MCP when the initial linkage between the user program and the library program is being established for this entry point. The selection procedure is passed an EBCDIC string whose value is the LIBPARAMETER attribute value given in the user program for this library. The selection procedure returns information to the MCP indicating which task (which must be a sub-task of the library program) will provide the entry point. This library task can provide the entry point directly, indirectly, or dynamically (again). The ability to provide entry points dynamically is available only to libraries written in the ALGOL language.

The only restrictions on the complexity of indirect and dynamic linkages are that 1) eventually some library must provide the entry point directly, and 2) the chain of referenced libraries must never become circular.

2.6 Error Handling

Any fault that is caused (and ignored) by a library procedure that is invoked by a user program will be treated as a fault in the user program, and if ignored by the user program will cause the user program to be DSed. The status of library program will be unaffected by the fault.

If a library faults (or is otherwise DSed) prior to execution of a FREEZE statement, then all user programs that are waiting to link to that library will also be DSed.

If a library program is DSed while user programs are linked to it, these user programs will also be DSed.

3. CREATING LIBRARIES

Libraries may be created by any user within the limitations that the language used imposes upon the library facility. The programmer of a library program need only define a user-server relationship where the library serves a specified function via procedural entry points. A library program differs from other programs only in the declaration of procedures to be exported as library entry points and the use of the FREEZE statement to make the program available as a library. (COBOL and COBOL74 differ from the other languages in that all programs written in these languages are capable of being invoked as either a normal program or as a library program without the need to use the FREEZE statement.)

3.1 Library Sharing Specifications

The allowed simultaneous uses of a library is specified by the creator of the library at compile time via the dollar option SHARING, one of: SHARING=PRIVATE, SHARING=SHAREDBYALL, or SHARING=DONTCARE. If the sharing is PRIVATE then for each user a separate instance of the library is started. If the sharing is SHAREDBYALL then all simultaneous users share the same instance of the library. If the sharing is DONTCARE then any combination of sharing will be used according to the MCP, but unknown to the creator and user of the library. Users of a library can be restricted, not only via the different library sharing specifications, but also through the normal file access restrictions of the system.

4. REFERENCING LIBRARIES

In order to use a library, the user program declares a library, its attributes, and the entry points from that library that it will use. Upon invoking an entry point the library linkage is automatically made by the MCP. If the library is not already started, the MCP starts it, and upon freezing it is connected to the user. The MCP will attempt to make linkages to all the entry points in a library at the first invocation of any one entry point in the library.

4.1 Library Attributes

Libraries have attributes much the same as files have attributes. Library attributes can be set and tested programmatically much like file attributes. Libraries have the attributes: INTNAME, TITLE, and LIBPARAMETER. All three of these attributes are EBCDIC string type attributes. The INTNAME is the internal identifier for the library. The TITLE is the code file name of the library. The LIBPARAMETER is an EBCDIC string used for transmitting information from the user program to the selection procedures of library programs that provide dynamic entry point linkage. Library attributes may be changed dynamically by the user program; these changes however must occur prior to the first attempt to link to the library. Changes to library attributes in the user program that are made after the user program is linked to the library are ignored by the MCP.

4.2 Type Matching

Library procedural entry points can be either typed or untyped. Type matching is performed on the entry point during the library linkage. If the description in the template does not match the description in the directory the linkage is not made and the referencing program is DSed. Matching includes several factors: the procedure type, the number of parameters, the parameter types, and how the procedure and parameters are provided. The provision of the procedure and parameters can be by value, by reference, or by name.

If a library declares a parameter to be provided by name, the user of the library can pass the parameter by name, by reference, or by value. If the library declares the parameter to be passed by reference, then the user can pass it by reference or by value. If the parameter is declared by value in the library, it can only be passed by value from the user of the library.

ALGOL generally passes and expects parameters by name or by value; strings, arrays, and formal procedures are passed by reference. COBOL, FORTRAN, and PL/I pass parameters by reference unless specified otherwise.

4.3 Parameter Passing

A formal parameter array in ALGOL that is declared using the '[*]' lower bound construct is actually passed as two parameters: the array and the lower bound of the array as an integer parameter. COBOL (when it is declared as 'with lower bounds') and FORTRAN also pass array parameters in this fashion. Note that, for ALGOL there is an identical relationship between passing an array with '[*]' lower bounds and passing two parameters: an array row with constant (for example '[0]') lower bounds and an integer by value.

For multi-dimensioned (ALGOL and COBOL) arrays, if any lower bound is an asterisk, then all the lower bounds are passed, one integer for each bound. All FORTRAN arrays (regardless of the number of dimensions) are passed as an ALGOL single dimension array, with a lower bound.

5. LIBRARIES IN ALGOL

ALGOL supports the full library facility, including the feature of creating a library which provides dynamic entry point linkages.

5.1 Creating Libraries

There are two constructs in ALGOL for creating libraries - The <export declaration> and the <freeze statement>:

<export declaration>

```
                 |<------------------- , -------------------|
                 |                                          |
   -- EXPORT ---<procedure id>-----------------------------------|
                              |                           |
                              |- AS --<ebcdic string>-|
```

The <export declaration> declares procedures to be entry points in the library. All <procedure id>s must be declared before the <export declaration> and must be in the same block as the <export declaration>.

The <ebcdic string> in the AS clause cannot contain any leading, trailing, or embedded blanks and must be a <valid identifier>. The AS clause specifies the name that will be given for this procedure in the library directory built from this declaration.

If the EXPORTed procedure is declared in a <library entry point declaration> then the exported procedure is provided indirectly.

If the exported procedure is to be provided dynamically, then the exported procedure is declared as follows:

<dynamic procedure declaration>

```
     --<procedure heading>-- ; ---------------------------------->

     >- BY CALLING --<selection procedure id>--------------------|
```

<selection procedure id>

```
     --<procedure id>--------------------------------------------|
```

The <dynamic procedure declaration> declares a procedure which is to be EXPORTed dynamically. The dynamic procedure cannot be declared forward and cannot be a separately-compiled procedure. It cannot be referenced directly in the library program declaring it.

The <selection procedure id> must be untyped, with two parameters. The first parameter must be a call-by-value EBCDIC string. The second parameter must be a formally specified untyped procedure with one parameter which is a task. When the MCP invokes this selection procedure the task variable passed to its procedure parameter must already be associated with a library which has been PROCESSed using this task variable.

<freeze statement>

```
     -- FREEZE -- ( --- PERMANENT --- ) -------------------------|
                     |               |
                     |- TEMPORARY -|
```

When executed, the <freeze statement> changes the running program into a library. There must be at least one export declaration in the same block as the <freeze statement>. Those procedures affected by a <freeze statement> are the procedures which appear in export declarations in the same block as the <freeze statement>.

If TEMPORARY is specified, the library created by the <freeze statement> exists only as long as at least one user is referencing it. If no users are referencing the library, it resumes execution following the <freeze statement>. If PERMANENT is specified, the created library stack exists no matter how few users reference it.

Example showing 'Direct' library linkage:

```
 BEGIN
      ARRAY MSG[0:120];
      INTEGER PROCEDURE FACT(N);
         INTEGER N;
      BEGIN
           IF N LSS 1
           THEN FACT := 1
           ELSE FACT := N * FACT(N - 1);
      END; % OF FACT.
      PROCEDURE DATEANDTIME(TOARRAY, WHERE);
        ARRAY TOARRAY[*];
        INTEGER WHERE;
      BEGIN
        REAL T;
        POINTER PTR;
           T := TIME(7);
           PTR := POINTER(TOARRAY, 8) + WHERE;
           CASE T.[5:6] OF
           BEGIN
                0: REPLACE PTR:PTR BY "SUNDAY, ";
                1: REPLACE PTR:PTR BY "MONDAY, ";
                2: REPLACE PTR:PTR BY "TUESDAY, ";
                3: REPLACE PTR:PTR BY "WEDNESDAY, ";
                4: REPLACE PTR:PTR BY "THURSDAY, ";
                5: REPLACE PTR:PTR BY "FRIDAY, ";
                6: REPLACE PTR:PTR BY "SATURDAY, ";
           END;
           REPLACE PTR BY T.[35:6] FOR 2 DIGITS, "-",
                          T.[29:6] FOR 2 DIGITS, "-",
                          T.[47:12] FOR 4 DIGITS, ", ",
                          T.[23:6] FOR 2 DIGITS, ":",
                          T.[17:6] FOR 2 DIGITS, ":",
                          T.[11:6] FOR 2 DIGITS;
      END; % OF DATEANDTIME.
      EXPORT FACT, DATEANDTIME AS "DAYTIME";
          REPLACE POINTER(MSG, 8) BY
               "  - SAMPLE LIBRARY STARTED",
               " " FOR 94;
          DATEANDTIME(MSG, 60);
          DISPLAY(MSG);
          FREEZE(TEMPORARY);
          REPLACE POINTER(MSG, 8)+ 19 BY "ENDED  ";
          DATEANDTIME(MSG, 60);
          DISPLAY(MSG);
END.
```

Example showing 'Dynamic' and 'Indirect' library linkage:

```
BEGIN
    TASK LIB1TASK, LIB2TASK;
    LIBRARY SAMLIB(TITLE= "OBJECT/SAMPLE/LIBRARY.");
    INTEGER PROCEDURE FACT(N); INTEGER N; LIBRARY SAMLIB;
    PROCEDURE DYNLIB1;
    % LIBRARY PROVIDED DYNAMICALLY AND INDIRECTLY
      BEGIN % PRINTS DATE WITH TIME.
         PROCEDURE DAYTIME(TOARRAY, WHERE);
              ARRAY TOARRAY[*];
              INTEGER WHERE;
                    LIBRARY SAMLIB;
         EXPORT DAYTIME;
         FREEZE(TEMPORARY);
      END; % OF DYNLIB1.
    PROCEDURE DYNLIB2;
    % LIBRARY PROVIDED DYNAMICALLY
      BEGIN % PRINTS DATE WITHOUT TIME.
         PROCEDURE DAYTIME(TOARRAY, WHERE);
              ARRAY TOARRAY[*];
              INTEGER WHERE;
           BEGIN
              REAL T;
                   T := TIME(7);
                   REPLACE POINTER(TOARRAY, 8) + WHERE
                        BY T.[35:6] FOR 2 DIGITS, "-",
                           T.[29:6] FOR 2 DIGITS, "-",
                           T.[47:12] FOR 4 DIGITS;
           END; % OF DAYTIME.
         EXPORT DAYTIME;
         FREEZE(TEMPORARY);
      END; % OF DYNLIB2;
    % THE SELECTION PROCEDURE.
    PROCEDURE THESELECTIONPROC(LIBPARSTR, NAMINGPROC);
         VALUE LIBPARSTR;
         STRING LIBPARSTR;
```

Since COBOL data is global to the COBOL program and of course global to the entry point, and since the parameters must be kept global, it is necessary that use of the COBOL library be restricted to one user at a time. This restriction imposed upon shared COBOL libraries is maintained entirely by compiler code with a lock. Although this is automatic and invisible to the COBOL programmer, it should be noted that sharing of COBOL libraries is strictly serial, that is the next user must wait upon the previous user before he is allowed to enter the library entry point. This lock upon the library also disallows a user of a library from recursively or cyclically calling the library since he would be in a dead lock situation waiting for the release of the lock which he himself holds. If a user attempts a recursive or cyclical call upon a COBOL library with $SHARING = SHAREDBYALL, the library will DS that user.

6.1 Creating Libraries

The creation of a library in COBOL is completely automatic. All COBOL programs which are compiled at level two and which do not declare any parameters which are illegal for libraries are automatically made capable of being a library. If the 'Using' clause of the Procedure Division declares a parameter which cannot be used in a library, the program is not made callable as a library. Other than this, all level two programs are capable of being called as a library. ~~If the COBOL program contains a 'Program-Id' comment, the first word of this comment is used as the name of the library entry point. If no 'Program-Id' comment appears, the name of the entry point is PROCEDUREDIVISION.~~

To exit from a COBOL library and return to the calling program an 'Exit Program' statement must be used. If a COBOL library attempts to do a 'Stop Run' the user is DSed for attempting to have the library terminate prematurely.

Data names that are parameters to a COBOL program cannot be specified with a 'VALUE'clause in the DATADIVISION if the COBOL program is to be invoked as a library.

6.2 Referencing Libraries

There is one construct in COBOL for calling libraries - The <call non-numeric literal statement>:

<call non-numeric literal statement>

```
-- CALL -- " --<procedure id>--- IN ---<library id>-- " ---->
                                 |     |
                                 |- OF -|

>--------------------------------------------------------------|
 |                                   | |                       |
 |              |<----- , ----|      | |- GIVING --<data-name>-|
 |              |             |      |
 |- USING ---<data-name>---|         |
```

The contents of the non-numeric literal are interpreted as follows:

1) The <procedure id> is the name of the procedural entry point. In COBOL this would be the Procedure Division with the name PROCEDUREDIVISION. In other languages, such as ALGOL, this would be the procedure name.

2) The <library id> would be the name of the library to be called. This can be a file name comprised of letters, numbers, hyphens and underscores and separated by slashes. This name (if it does not contain slashes, otherwise the last name following a slash) is used as the INTNAME of the library. This name, including slashes, is also used as the TITLE of the library.

The 'USING' clause identifies the parameters that are being passed to the library procedure.

The 'GIVING' clause specifies the destination for the value (of type INTEGER) that is returned from the procedure.

6.3 Changing Attributes of Libraries

There is one construct in COBOL for changing attributes of libraries - The <change attribute of non-numeric literal statement>:

<change attribute of non-numeric literal statement>

```
-- CHANGE ATTRIBUTE --<library attribute>--- OF --- " ------>
                                            |      |
                                            |- IN -|

>-<library id>-- " -- TO ---<identifier>--------------------|
                            |              |
                            |-<literal>----|
```

<library attribute>

```
     ---- INTNAME ---------------------------------------------|
         |                    |
         |- TITLE --------|
         |                    |
         |- LIBPARAMETER -|
```

The <library id> in the non-numeric literal coincides with the <library id> in the <call non-numeric literal statement> above and will be used in the same manner.

The <identifier> must be an alphanumeric Display data-item. The <literal> must be a non-numeric literal.

6.4 Dollar Options for Libraries

A Library, when created by the COBOL compiler, can be specified to be Permanent or Temporary through the following compiler dollar option:

TEMPORARY
If set the library will be temporary and remain until there is no user referencing the library. If reset, as is the default, the library will remain until it is DSed.

Example:

```
$SET TEMPORARY
 IDENTIFICATION DIVISION.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01 DATIME.
   03 MSG PIC X(17) VALUE " 13 FACTORIAL IS ".
   03 AMT PIC 9(12).
   03 SEP PIC X(11) VALUE "     -     "
   03 DAT PIC X(81).
 77 NUM PIC 9(2) COMP VALUE 13.
 01 RA COMP WITH LOWER-BOUNDS.
   03 RAS PIC 9(6) COMP OCCURS 13 .
 77 WH PIC 9 COMP VALUE 0.
 PROCEDURE DIVISION.
 P1.
    CHANGE ATTRIBUTE LIBPARAMETER OF
         "OBJECT/SAMPLE/DYNAMICLIB" TO "WITH TIME".
    CALL "FACT OF OBJECT/SAMPLE/DYNAMICLIB"
         USING NUM GIVING AMT.
    CALL "DAYTIME IN OBJECT/SAMPLE/DYNAMICLIB"
         USING RA, WH.
    MOVE RA TO DAT.
    DISPLAY DATIME.
      STOP RUN.
```

6.5 Entry Points and their Parameters

COBOL Parameter	Corresponding ALGOL Parameter
Comp, 77	Integer variable
Comp, 01	Real array [*]
Comp-4, 77	Real variable
Comp-5, 77	Double variable
Comp-1, 77 single	Integer variable
Comp-1, 77 double	Double variable
Pic X, 77	Ebcdic string variable
Comp-2, 01 no occurs	Hex character array [0]
Comp-2, 01 occurs	Hex character array [0,0]
ASCII, 01 no occurs	ASCII character array [0]
ASCII, 01 occurs	ASCII character array [0,0]
Display, 01 no occurs	Ebcdic character array [0]
Display, 01 occurs	Ebcdic character array [0,0]
Transaction record	Transaction record
Transaction record array	Transaction record array [0]

7. LIBRARIES IN COBOL74

COBOL74 supports the entire library facility that COBOL supports in addition to some special features for the ANSI74 Inter-Program Communication facility. COBOL74 also differs from COBOL in the allowable parameter types for libraries (see above).

Since the ANSI74 IPC facility has been implemented in COBOL74 using the library facility a modified syntax has been used to designate a call upon a library which is to be used in accordance with the ANSI74 IPC specification. The significant difference is that linkage to the library is severed after each call upon the library, causing re-linkage for each new call upon the library. This is necessary due to the additional requirement that the library be able to be canceled.

7.1 Creating Libraries

As with COBOL, libraries in COBOL74 are automatically created for all level two COBOL74 programs which do not declare any parameters disallowed for libraries. The single entry point is the Procedure Division, as it is for COBOL, with the entry point name of PROCEDUREDIVISION.

7.2 Referencing Libraries

There are two constructs in COBOL74 related to calling libraries - The <call non-numeric-literal or data-name statement> and the <cancel statement>:

<call non-numeric-literal or data-name statement>

```
-- CALL --- " --<library id>-- " --------------------------->
          |                       |
          |-<data-name>-----------|

>---------------------------------------------------------->
   |                     |
   |         |<----- , ----|  |
   |         |             |  |
   |- USING ---<data-name>---|

>----------------------------------------------------------|
   |                                          |
   |-------- OVERFLOW --<imperative statement>-|
   |        |
   |- ON -|
```

The <call non-numeric-literal or data-name statement> in COBOL74 is quite similar to the <call statement> in COBOL, except the '<procedure id> OF/IN' is omitted from the non-numeric literal and the 'ON OVERFLOW <imperative statement>' optional clause is allowed.

The optional clause 'ON OVERFLOW <imperative statement>' is included for the ANSI74 IPC facility specifications, but has no effect upon the execution of a program.

~~The use of the non-numeric literal in the <call statement> is functionally the same as in COBOL, except since the <procedure id> is omitted, it is assumed to refer to an entry point named PROCEDUREDIVISION.~~ The contents of the <data-name> is used the same as the <library id>.

<cancel statement>

```
                |<----------- , -----------|
                |                          |
-- CANCEL ----- " --<library id>-- " -----------------------|
                  |                       |
                  |-<data-name>-----------|
```

The <cancel statement> is used to cancel a library, that is to say if there are no more users of the library the library will go to 'end-of-task'. Of course, this naturally will occur for temporary libraries, but also this allows a user to cancel a Permanent library. The <library id> is again used the same as in the <call statement>. Furthermore, the contents of the <data-name> will be used in the same manner as is the <library id>.

7.3 Changing Attributes of Libraries

Changing attributes of libraries in COBOL74 is exactly the same as changing attributes of libraries in COBOL. ANSI74 IPC library calls cannot be referenced by a CHANGE statement however. IPC libraries cannot have their attributes changed.

Example:

```
IDENTIFICATION DIVISION.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 PARAM PIC 9(21) COMP.
PROCEDURE DIVISION.
P1.
   CHANGE ATTRIBUTE TITLE OF "OBJECT/A/COBOL/LIB"
         TO "OBJECT/ANOTHER/COBOL/LIB".
   CALL "PROCEDUREDIVISION OF OBJECT/A/COBOL/LIB"
         USING PARAM.
* NOTE: THE ABOVE CALL REFERENCES "OBJECT/ANOTHER/COBOL/LIB",
*   WHEREAS THE FOLLOWING WILL REFERENCE
*     "OBJECT/A/COBOL/LIB".
   CALL "OBJECT/A/COBOL/LIB" USING PARAM.
```

```
CANCEL "OBJECT/A/COBOL/LIB".
  STOP RUN.
```

7.4 Entry Points and their Parameters

COBOL74 Parameter	Corresponding ALGOL Parameter
Comp, 01	Hex character array [0]
~~ASCII, 01~~	~~ASCII character array [0]~~
Display, 01	Ebcdic character array [0]
Transaction record	Transaction record
Transaction record array	Transaction record array [0]

8. LIBRARIES IN FORTRAN

Libraries may be created with or called from FORTRAN programs. A new non-executable program unit, the BLOCK GLOBALS subprogram, has been implemented for global declarations associated with libraries.

8.1 BLOCK GLOBALS Subprogram

A BLOCK GLOBALS Subprogram is used for declaring libraries, export lists, and files. Only one BLOCK GLOBALS subprogram is allowed in a FORTRAN program and it must occur after any old style FILE declarations (appendix B of the FORTRAN Manual) and must precede all other FORTRAN source statements.

The BLOCK GLOBALS subprogram is initiated by a BLOCK GLOBALS statement and terminated by an END statement. The only statements which are allowed in this subprogram are LIBRARY statements, EXPORT statements, and FILE statements.

Files may be declared in the BLOCK GLOBALS subprogram by use of the non-executable FILE statement.

<file statement>

```
-- FILE --<file designator>-- ( ---------------------------->

>---<file attribute>-- = ---<expression>--- ) --------------|
    |                       |                  |
    |                       |-<mnemonic>---|
    |                                          |
    |-<boolean file attribute>-------------|
```

The <file designator> is an unsigned integer from 1 to 99. The <file attribute> is a valid file attribute identifier as listed in the I/O Subsystem Reference Manual. The <expression> or <mnemonic> must be a valid value for the attribute identifier. The <boolean> is a file attribute identifier of type Boolean.

The second form of the file attribute assignment results in the value of TRUE being assigned to the file attribute identifier indicated.

8.2 Creating Libraries

There are two constructs in FORTRAN for creating libraries: the EXPORT statement and the CALL FREEZE statement.

The non-executable EXPORT statement is used to specify which subroutines and functions are to be provided as entry points to the library upon execution of the CALL FREEZE statement. EXPORT statements must appear in the BLOCK GLOBALS subprogram.

<export statement>

```
              |<--------------- , --------------|
              |                                 |
-- EXPORT ------<subprogram>-----------------------------------|
                |                               |
                |-<subprogram>-- = --<string>-|
```

The <subprogram> name is an entry point designator and the string indicates the name by which the subprogram is to be known to users of the library.

The executable statement CALL FREEZE changes the running program into a library stack.

<freeze statement>

```
-- CALL FREEZE -- ( --- TEMPORARY --- ) --------------------|
                    |                 |
                    |- PERMANENT -|
```

If TEMPORARY is specified, the library stack exists only as long as at least one user is referencing it. If PERMANENT is specified, the library stack continues to exist even if no users are referencing it.

The following is an example of a FORTRAN program which creates a library.

```
$ SHARING = PRIVATE
      BLOCK GLOBALS
            FILE 6(KIND=PRINTER)
            EXPORT SINE="SIN",COSINE
      END
      REAL FUNCTION SINE(X)
C*          PERFORM SINE CALCULATION...
            SINE=X
      END
      REAL FUNCTION COSINE(X)
C*          PERFORM COSINE CALCULATION...
            COSINE=X
      END
C*    MAIN PROGRAM
            WRITE(6,/)SINE(X),COSINE(X)
            CALL FREEZE(TEMPORARY)
      END
```

8.3 Referencing Libraries

There are two constructs in FORTRAN for calling libraries: the LIBRARY statement and the IN LIBRARY statement.

All libraries used by a program must be declared in a LIBRARY statement. LIBRARY statements must appear in the BLOCK GLOBALS subprogram.

<library statement>

```
-- LIBRARY --<library id>---------------------------------->

>-------------------------------------------------------------|
    |                                                         |
    |       |<---------------- , ----------------|           |
    |       |                                    |           |
    |- ( -----/1\- TITLE = --<string>------------ ) -|
             |                                 |
             |-/1\- INTNAME = --<string>------|
             |                                 |
             |-/1\- LIBPARAMETER = --<string>-|
```

Specification of the attributes is optional; by default the name being declared is used for TITLE and INTNAME. If the INTNAME is given, but the TITLE is not, the string given for the INTNAME will be used for the TITLE.

The IN LIBRARY statement is used to indicate that a subprogram is to be found in a library.

<in library statement>

```
-- IN LIBRARY --<library id>-------------------------------->

>-------------------------------------------------------------%
    |                                 |
    |- ( ACTUALNAME = --<string>-- ) -|
```

The <library id> name is declared in the BLOCK GLOBALS subprogram. The <string> indicates the actual name of the subprogram in the library.

The only statements which are allowed in a subprogram which contains an IN LIBRARY statement are type statements and DIMENSION statements. These must specify type and dimensionality of the subprogram's parameters.

8.4 Changing Attributes of Libraries

Library attributes may be changed before the library is accessed by use of the CHANGE statement. For example:

```
REAL A(4)/"LIBRARY/ONE."/
CHANGE(LIB1,TITLE=A)
```

The following is an example of a FORTRAN program which uses the two functions created by the above example and a subroutine which is in another library.

```
      BLOCK GLOBALS
        FILE 5(KIND=REMOTE)
        FILE 6(KIND=PRINTER)
```

```
            LIBRARY LIB1(TITLE="MATHINTRINSICS",
     *                   INTNAME="MATHINTRINSICS")
            LIBRARY LIB2
      END
      REAL FUNCTION SIN(X)
            REAL X
            IN LIBRARY LIB1
      END
      REAL FUNCTION COS(X)
            REAL X
            IN LIBRARY LIB1(ACTUALNAME="COSINE")
      END
      SUBROUTINE SUB
            IN LIBRARY LIB2
      END
C*    MAIN PROGRAM
            READ(5,/)X
            WRITE(6,/)SIN(X),COS(X)
            CALL SUB
      END
```

8.5 Entry points and their Parameters

FORTRAN Parameter	Corresponding ALGOL Parameter
Integer	Integer variable
Real	Real variable
Logical	Boolean variable
Double precision	Double variable
Complex	Complex variable
Integer array	Integer array [*]
Real array	Real array [*]
Logical array	Boolean array [*]
Double precision array	Double array [*]
Complex array	Complex array [*]

9. LIBRARIES IN PL/I

Libraries may be created with or called from PL/I programs. There are some limitations on parameter and entry point types, and all libraries written in PL/I must be PRIVATE.

9.1 Creating Libraries

The <option-list> of the main (i.e., outer block) <procedure-statement> is extended to include the <export-option>. The <export-option> specifies which entry points the library is making available to programs which reference it. Its syntax is the following:

<export-option>

```
                    |<----------------- , -----------------|
                    |                                      |
 -- EXPORT -- ( ---<procedure id>------------------------------>
                                      |                  |
                                      |- AS --<char-string>-|

 >- ) -----------------------------------------------------|
```

Example:

PROGRAM: PROCEDURE OPTIONS (MAIN, EXPORT (P1, P2 AS 'X'));

The <procedure id>s and <char-string>s must be constants. Procedure or entry names may be used. In the above example, P1 and P2 might be the following:

```
P1: PROCEDURE ....;
P2: ENTRY ........;
```

The procedures and entries being exported must be declared in the outer block.

In order to become a library at run-time, a program must become "frozen". The method for doing this in PL/I is the execution of a FREEZE statement. Its syntax is the following:

<freeze statement>

```
 -- FREEZE OPTIONS -- ( --- PERMANENT --- ) -- ; ------------|
                         |              |
                         |- TEMPORARY --|
```

The FREEZE statement must occur in the outer block of the program.

9.2 Referencing Libraries

Calling libraries is accomplished by declaring <library-name>s and specifying which <procedure id>s within the program reference a given library. Library declaration, which must be explicit, is done by using the <library attribute> in a declaration. The syntax is the following:

<library attribute>

```
     -- LIBRARY ------------------------------------------------|
                   |                                     |
                   |- ( --<library attribute list>-- ) -|
```

<library attribute list>

```
        |<---------------- , ----------------|
        |                                    |
     ------/1\- INTNAME = <char-string> ------------------------|
           |                                 |
           |-/1\- TITLE = <char-string> -------|
           |                                 |
           |-/1\- LIBPARAMETER = <char-string> -|
```

The TITLE, INTNAME and LIBPARAMETER attributes are as they are in the other languages. The <char-string>s must be constants. Specification of these attributes is optional; by default the name being declared is used for TITLE and INTNAME. If the INTNAME is given, but the TITLE is not, the string given for the INTNAME will be used for the TITLE.

Association of an entry point name with a library is accomplished with the <options-list> added to <entry point declaration>s. The syntax is the following:

<entry point options list>

```
                      |<--------------- , ----------------|
                      |                                   |
     -- OPTIONS -- ( -----/*1\- LIBRARY = <library id> ---------->
                           |                              |
                           |-/1\- ACTUALNAME = <char-string> -|

     >- ) ------------------------------------------------------|
```

Use of an <options-list> which includes LIBRARY or ACTUALNAME will imply the attributes of ENTRY and CONSTANT for the name being declared. These attributes are mutually exclusive with file attributes, the use of which implies that a file is being declared. The library must be named; the actualname is optional, the declared identifier being used by default.

Example:

```
     DECLARE LIB1 LIBRARY
         (TITLE =  'MATH/LIBRARY');

     DECLARE MY_LIBRARY LIBRARY
         (INTNAME = 'YOUR_LIBRARY');

     DECLARE DO_THIS ENTRY (BINARY FIXED, CHAR(*))
          RETURNS (CHARACTER(100) VARYING)
          OPTIONS (LIBRARY = MY_LIBRARY,
                   ACTUALNAME = 'DO_THAT');
```

In this example, calling the entry DO_THIS from the PL/I program would result in the MCP linking the program to a library called YOUR_LIBRARY with an entry point called DO_THAT.

9.3 Changing Attributes of Libraries

The attributes may also be set and tested dynamically using a syntax similar to that for file and task attributes.

<library attribute reference>

```
     --<library attribute>-- ( --<library id>-- ) --------------|
```

The values given for these attributes are subject to some constraints:

1) The TITLE must be a valid file title as judged by the system; *, (usercode) and ON PACKNAME specifications are allowed.

2) The INTNAME must be a single identifier.

3) The LIBPARAMETER may be any string and is limited only by the conventions of the

particular library.

9.4 Entry Points and their Parameters

The following attributes may be given to return types for library entries:

PL/I Parameter	Corresponding ALGOL Parameter
Binary fixed(p, q),	
...q = 0, p <= 39	Integer variable
...q = 0, 39 < p <= 78	unique to PL/I
...q neq 0, p <= 39	unique to PL/I
...q neq 0, 39 < p <= 78	unique to PL/I
Decimal Fixed(p, q),	
...q = 0, p <= 11	Integer variable
...q = 0, 11 < p <= 23	unique to PL/I
...q neq 0, p <= 11	unique to PL/I
...q neq 0, 11 < p <= 23	unique to PL/I
Binary Float(p),	
...p <= 39	Real variable
...39 < p <= 78	Double variable
Decimal Float(p)	
...p <= 11	Real variable
...11 < p <= 23	Double variable
Character(n)	Ebcdic string variable
Character(*)	Ebcdic string variable
Character(n) Varying	Ebcdic string variable
character(*) varying	Ebcdic string variable
Bit(n), n <= 48	Boolean variable
Transaction record	Transaction record
Transaction record array	Transaction record array [*]

TRANSACTION PROCESSING SYSTEM

I. TRANSACTION PROCESSING SYSTEM: AN OVERVIEW

1. Motivation

Several considerations have motivated the design and development of the Transaction Processing System (or "System").

On-line applications

Data management applications are moving towards transaction oriented systems which have high volume online update via video terminals. This trend is becoming increasingly important because of the great user demand.

It is difficult to solve the recovery problems of an online system while maintaining efficiency. DMS II recovery backs out a certain number of transactions from the data base. These must be resubmitted before new input is accepted, and, for most applications, must be reprocessed in such a way that the previous results are reproduced. Due to multiprogramming, it is not immediately apparent how to achieve this reproducibility. Reproducibility must be designed into the system from the beginning.

Unlike batch systems, online systems must save a history of the input data. If data base recovery fails because of irrecoverable I/O errors on the DMS II audit trail or for any other reason, the data base must be reloaded and the input rerun. This is the user's ultimate disaster mode backup. If this procedure is resorted to and fails, information is lost.

Isolate data base code

It is desirable to isolate user application programs from changes in the structure of the DMS II data base without sacrificing efficiency. Remaps are not sufficient for this isolation. What is needed is a way to factor out all the code which has knowledge of the data base into one module.

Minimize application code

It is highly desirable to minimize what the user has to write in order to bring up his application. The user should only have to describe what is unique about his application from other applications. The rest of the software should be supplied by Burroughs.

Shared Resource systems

It is impractical to attempt to implement the current DMS II architecture across shared resources systems; the volume of communication between two systems necessary to support the current architecture is prohibitive. Also, the fact that shared resources systems can be Halt/Loaded independently adds a completely new dimension to the problem.

Modularity

Modularity is essential in any large software system. The critical problems in achieving modularity arise in the interfaces. In an update application there are two main types of interfaces: the transaction-oriented interface and the data base-oriented interface. It must be possible to achieve simple, well-defined, efficient interfaces.

The System addresses all these considerations.

2. Overview

The first step in coming up on the Transaction Processing System (TPS) is to define a routine for every update to the data base. Each update routine is written by the user to (1) take a transaction record as input, (2) update and/or retrieve information from the data base, and (3) produce a transaction record as output. Update routines should have no knowledge of anything other than the input and output transaction records and the data base. In particular, they should have no knowledge of datacom or files or what program is calling them, or the results of previous transactions (except changes to the data base).

It is expected that there will be on the order of a few hundred transaction routines, most of which are used infrequently. A typical routine is expected to execute about a dozen data management operations. The update routines are collected together in one or a small number of libraries called the Update Library. The Update Library is never run as a program. It is only used as a collection of service routines.

The formats of the transaction records are logically defined by user in the Transaction Format Language (TFL) which is similar to DASDL. TFL source is processed by the TFL compiler supplied with the TPS and a description file is produced which describes the collection of transaction formats or "transaction base". The host language compilers (COBOL, ALGOL, PL/I) obtain information from the description file via the transaction base interface program supplied with the TPS. This scheme enables user programs to reference the centrally defined data items in transaction records in a way symbolically similar to the way they reference data items in DMS II data set workareas.

Once all data base update routines have been defined as described above, the next step is to convert all update programs to use them instead of updating the data base directly. Thus all the data base knowledgable code is collected into the Update Library. The job of all other update programs becomes: (1) collect input data (from a terminal, a file, etc.), (2) assemble a transaction input record from the input data, (3) call the appropriate routine in the Update Library, (4) disassemble the transaction output record to output the result (to a terminal, a file, the printer, etc.).

For step (3), the Update Library routine is not called directly, however. The Transaction Library supplied with the TPS is called instead. The Transaction Library in turn calls the Update Library to perform the transaction and returns the result to the user program. Interposing the Transacton Library between the user program and the Update Library enables the TPS to save a history of all transaction records and to automatically resubmit transactions backed out by DMS II recovery. Another advantage of the scheme is that it can be used to make the system on which the data base resides transparent to user programs in a loosely coupled environment.

By following a few simple conventions in coding the Update Library routines, reproduction of previous results can be achieved when transactions backed out by DMS II recovery are automatically resubmitted by the TPS. Reproducibility can be achieved even if the data base was being updated in a multiprogrammed mode. Reproducibility is discussed in greater detail in Section V below. Reproducibllty requires that when any updaters are using the TPS, all updaters are. An option is provided to enforce this, if desired. Inquiry-only programs and DMS II INQUIRY may, however, be run while the TPS is running.

The TPS can also be used for batch or online data collection of transaction records. This is achieved by having the Transaction Library save the input transaction records in a transaction journal file but not call the Update Library. The Transacton Library can later be called to retrieve the transaction records from the journal.

One reason the extra processor overhead imposed by the TPS scheme is low is that the Transaction Library and Update Library routines are called as procedures on top of the user program process stacks (similar to the way DMS II ACCESSROUTINES and MCP routines are called as procedures on top of user program stacks).

The details of the Transaction Processing System are presented in what follows.

3. Definitions

Transaction:
A transaction (from the software point of view) is a unit of processing against a single DMS II data base. Transactions are performed by transaction processing routines. A transaction may include, but is not to be confused with, a data base transaction, which is application code delimited by BEGIN-TRANSACTION and END-TRANSACTION.

Transaction processing routine:
A transaction processing routine takes a single transaction record as input and produces a single transaction record as output. The routine may retrieve data from and/or update a DMS II data base.

Transaction record:
A transaction record is a structured variable which contains user-defined data items and some system-defined control items. The user-defined items are similar to the data items in a DMS II data set record or a COBOL 01 level variable. Transaction records may be passed as parameters to and from procedures. They may also be read from and written to transaction journals.

Transaction Journal:
A transaction journal is a sequence of data files similar to a DMS II audit trail, plus a control file. It contains transaction records, and is manipulated via system-supplied software. When a transaction data file becomes full, the next one in sequence is automatically created.

Transaction Base:
A Transaction Base is the analog of a data base. It is a collection of user-defined transaction formats, journals, subbases, and other information related to processing transactions. A Transaction Base is designed for at most one data base.

Two phase:
If the first execution phase of a data base transaction only locks records but does not free any, and if the second and final execution phase of the transaction frees records but does not lock any, and if no records are retrieved without locking them, then the transaction is said to be two-phase.

Host and Remote Systems:
In a shared resource or BNA environment, the system (processor) which controls the data base is considered to be the "host" system; all other systems connected to it are "remote" systems.

4. Assumptions

The System, in order to achieve the objectives stated above, makes certain assumptions about the nature of the user's transactions.

a) Transactions are relatively "small" units of processing, containing at most one data base transaction. This implies that a single transaction is not designed to make massive and sweeping changes to a data base.

b) All input to a transaction processing routine is encoded in a single transaction record; similarly, all output from a transaction processing routine is encoded in a single transaction record.

c) No transaction uses any memory of the results of previous transactions (for example, it does not depend upon DMS II SIB paths from previous transactions). It depends only upon the state of the data base and its input transaction record; in this sense it "depends" upon previous transactions, but has no knowledge of them.

d) All update transactions are two-phase, or they use programmatic conventions to achieve the same result. In addition, they either force a sync point or keep all records locked until END-TRANSACTION. They observe certain other requirements established by the System.

e) While the System is in use to update the data base, no conventional update programs may run. Conventional inquiry programs may run in parallel with the Transaction System, however.

5. Summary of Features

The System enables the user to create a central definition of transaction record formats, to create and manipulate transaction records, and to manipulate transaction journals.

The user may initiate a transaction processing routine with an input transaction record, or may tank input transaction records into a transaction journal for later processing.

The System will automatically create an input transaction history and reprocess backed out transactions after data base recovery.

User programs are isolated from changes to the structure of the DMS II data base except for the Update Library and TFL.

Restart coding in user application programs is simplified. Programs need not code for abort recovery. For Halt/Loads, they merely ask the System for their last completed transaction, and reposition their input and output files accordingly.

The volume of communication between shared resources systems consists of signals carrying input and output transaction records. Isolation from record locking, recovery, etc. is achieved. Except for speed, the fact that the data base is on another system is transparent to the user program.

Transaction records and transaction journals permit clean, well-defined transaction interfaces to be developed, and the Update Library provides a clean, well-defined data base interface.

Although designed for use with a DMS II data base, the Transaction System may be a convenient tool for other applications. Transactions which do not access a data base may appear to the System as inquiry-only transactions. Of course, the tanking and journal reading functions are totally independent of data bases.

6. Software

6.1 Libraries

The following libraries are components of the Transaction Processing System.

1) The Update Library contains all the user-written transaction processing routines, or controls routing of input transaction records to the proper routines in other libraries. The user is responsible for creating and maintaining the Update Library, observing certain conventions required by the System.

2) The Transaction Library is provided by the System, and is tailored for a particular Transaction Base during compilation. It contains all the routines which are called by user-written application programs to process or tank transactions. The Transaction Library calls the Update Library to process each transaction, saves the transaction records in a transaction history file, and reprocesses them from the history file after DMS II recovery. In a shared resources system, the Transaction Library must reside on the host system, where it is known as the Host Library.

3) The Remote Library is provided for shared resources systems and resides on a remote system. The Remote Library has the same entry points as the Host Library and appears identical to the user. Application programs running on the remote system call the Remote Library instead of the Host Library to process transactions. The Remote Library uses ports and signals to communicate with its counterpart, the Host Interface. The Host Interface must reside on the host system, as it calls the Host Library to process transactions on behalf of the remote application program. Except for speed, this technique can make the whereabouts of the data base transparent to the application program.

6.2 TFL

A language called the Transaction Format Language (TFL) is used to define the Transaction Base. The TFL source, created and maintained by the Data Base Administrator (DBA) or equivalent, is the central definition of all transaction formats, journals and subbases used by application programs.

The TFL processor (SYSTEM/TFL) creates a description file from the TFL source. The description file is used to invoke transaction formats in application programs, and to compile tailored software.

6.3 Application Language Interface

The application languages (ALGOL, COBOL and PL/1) have been extended to support use of the Transaction System. Through the use of these extensions, application programs can invoke some or all transaction formats defined in a Transaction Base. The application compilers recognize a new data type, the TRANSACTION RECORD, which is a structured variable similar to a DMS II data set work area or a COBOL 01 record. Transaction records can be created in any of the invoked formats, and passed as parameters to and from procedures. The host compilers invoke and compile references to the Transaction Base by communicating with an external coroutine, SYSTEM/TRINTERFACE.

6.4 Special DMS Host Syntax

In order to achieve necessary control over the execution of the data base transaction in the Update Library, a few special forms of DMS host syntax have been implemented. These include a new variant to the data base OPEN statement, new forms of BEGIN-TRANSACTION and END-TRANSACTION, and a new statement, MID-TRANSACTION; they are restricted to the Update Library.

6.5 Utility

The System provides a Utility program which performs certain functions not appropriate for the Transaction Library. These functions include the initial creation of each new transaction journal; recovery of journals after some operational mishap; searching journal data files for certain transaction records, then printing and/or counting the selected records; printing the contents of the journal control file.

6.6 Released Files

The following table lists all of the special files that comprise the Transaction System.

Source File	Object File
SYMBOL/TFL	SYSTEM/TFL
SYMBOL/TRINTERFACE	SYSTEM/TRINTERFACE
TRBASE/UTILITY	<base name>/CODE/UTILITY
TRBASE/HOSTLIB	<base name>/CODE/HOSTLIB
TRBASE/REMOTELIB	<base name>/CODE/REMOTELIB
TRBASE/HOSTINTERFACE	<base name>/CODE/HOSTINTERFACE
TRBASE/PROPERTIES	(none)
(user defined)	<base name>/CODE/UPDATELIB
	Data Files
(user defined)	<base name>/TRDESCRIPTION
(journal files)	<base name>/<journal name>/CONTROL <base name>/<journal name>/<file num>

Note that all files for a particular transaction base (tailored programs, description file, journal files) begin with the common directory name <base name>. In addition, all tailored programs begin with <base name>/CODE.

7. Reproducibility

Reproducibility during reprocessing is achieved by assuring that each data base transaction is independent of all others at a particular point during its execution; that is, it depends only upon the state of the data base. At that point the input transaction record . is saved in the history file. Serial processing of the transactions in the order that they occur in the history file is equivalent to the parallel processing of the original transactions. It can be shown that each transaction, whether in parallel or equivalent serial mode, is dependent only upon the state of the data base, and that state is identical for the transaction in either mode. If all transactions are two-phase, or the equivalent, then there is such a point of independence. This is sufficient for reproducibility during reprocessing after a Halt/Load.

While necessary, two-phase is not sufficient for abort recovery. It must be assumed that any transaction which was completed successfully before rerun will be successful during rerun mode. Since no attempt will be made to reprocess a transaction which has caused an abort, no completed transaction may depend upon any changes made by a transaction which may ultimately abort. This requires that all data base transactions must either (1) wait for a sync point or equivalent (or force a sync point), or (2) free no records, but allow END-TRANSACTION to free all records. Condition (1) implies that no transaction will finish while other transactions are active; condition (2) means that no transaction will start while another, upon which it depends, can yet cause an abort. Condition (1) is is not recommended, since it is likely to interfere with performance. Also, there is no way at the present to "wait for a sync point" without actually forcing one. Condition (2) is easy to observe, and it only affects transactions which are actually dependent upon others.

8. Using the Transaction System

Unlike conventional DMS II application programs, which may perform any data base statements the programmer wishes, an application program using the Transaction System does not invoke the data base, and therefore cannot perform ANY data base statements. The program is restricted to creating pre-defined transaction records and initiating pre-defined transaction processing routines via calls on procedures in the Transaction Library. In order to assure the proper design of the Transaction System, and of the various application programs which are to use it, the co-ordinated efforts of several kinds of users are required.

There are at least four kinds of users of the Transaction System (they could be a single individual). Each of these users must be familiar with the basic characteristics of the System, although perhaps to differing degrees.

The first is the Data Base Administrator (DBA), or someone like him, who is responsible for the overall design and maintenance of the System. He must be completely familiar with the data base and all of the transactions which are to access the data base. He must understand how the System works, including questions of recovery and reproducibliity. He must co-ordinate the efforts of the other users of the System.

The second is the programmer of the Update Library, who implements some or all of the transaction processing routines. Like the DBA, he must be familiar with the data base and the requirements of each of the transactions he implements; he may not be familiar with ALL transactions. He must understand how to design a data base transaction so that it observes conventions and requirements established by the System.

The third is the application programmer, who must write the programs which cause the System to perform desired work. He need not know anything about the data base, but must understand the input and output requirements, including transaction record formats, of each of the transactions he wishes to initiate. He must be familiar with all of the entry points in the Transaction Library, since those procedures are the tools he must use.

The fourth might be someone who runs an application program, particularly one of possibly several terminal users when the application program is interactive. He must understand the requirements of the program he is running, although a well-designed interactive program will prompt the user whenever necessary.

In order to begin using the Transaction System, the DBA must have some idea of the major applications which are to access the data base. The transaction processing routines, together with their input and output transaction record formats, must be defined. A Transaction Base, which contains the symbolic definitions of all transaction record formats, must be designed and processed with SYSTEM/TFL. The Transaction Library, which is tailored for a particular Transaction Base, must be compiled. If necessary, a Remote Library must also be compiled. All of the transaction processing routines, which reside in the Update Library, must be designed, implemented, and compiled (or bound) together to form the Update Library.

II. TRANSACTION FORMAT DESCRIPTION LANGUAGE (TFL)

1. TFL Syntax and Semantics

TFL is a language which is used to define transaction record formats and other properties of a transaction base. It is similar in several respects to the DASDL language, which is used, among other things, to define the formats of data set records.

The TFL Processor creates a transaction base description file from the TFL source definition of a transaction base. This description file is used to invoke the transaction base in application programs, and to compile tailored transaction software.

2. Transaction Base

<transaction base> represents the complete definition of a transaction base. It consists of several components which are described in later sections of this document.

<transaction base>

```
    ------------------------------------------------------------------>
      |                 |
      |- UPDATE -- ; -|

    >-<base name>-- TRANSACTION BASE -- ; ---------------------------->

    >----------------------------------------------------------------->
      |                                                 |
      |  |<-------------- ; --------------|             |
      |  |                                |             |
      |-----/1\-<parameters statement>----- ; -|
             |                          |
             |-/1\-<defaults statement>---|

      |<------------ ; -----------|
      |                           |
    >-----<transaction format>-----------------------------------------|
         |                        |
         |-<transaction subbase>-|
         |                        |
         |-<transaction journal>-|
```

<base name>

```
    --<simple identifier>----------------------------------------------|
```

UPDATE
: If present, UPDATE instructs the TFL Processor to update a previous description file with the current input.

<base name> TRANSACTION BASE
: <base name> is the name of the new transaction base. It consists of at most 17 letters and digits, beginning with a letter. The description file created from this definition will be titled <base name>/TRDESCRIPTION.

<parameters statement>
: specifies the values of global properties of the transaction base.

<defaults statement>
: specifies defaults to be used in the absence of required information.

<transaction format>
: defines the format of a single transaction record. There may be at most 1023 <transaction format>s in a <transaction base>.

<transaction subbase>
: a list of transaction formats to be invoked together. There may be at most 1023 <transaction subbase>s in a <transaction base>.

<transaction journal>
: defines the properties of a transaction journal.

<simple identifier>
: a sequence of at most 17 alphanumeric characters, beginning with a letter.

3. Parameters Statement

<parameters statement> specifies the values of global properties of the transaction base.

<parameters statement>

```
                         |<------- , -------|
                         |                  |
    -- PARAMETERS -- ( ---<parameter spec>--- ) ----------------------|
```

<parameter spec>

```
    ----<boolean parameter>-------------------------------------------|
       |                     |                       |
       |                     |- SET -----------------|
       |                     |                       |
       |                     |- RESET ---------------|
       |                                             |
       |-<numeric parameter>-- = --<numeric literal>-|
       |                                             |
       |-/1\-<data base parameter>-------------------|
       |                                             |
       |-/1\-<restart data set parameter>------------|
       |                                             |
       |-/1\-<host system parameter>-----------------|
```

<data base parameter>

```
    -- DATABASE -- = --------------------------<data base name>-------->
                        |                   |
                        |- ( --<usercode>-- ) -|
                        |                   |
                        |- * ---------------|

    >-----------------------------------------------------------------|
       |               |
       |- ON --<family name>-|
```

<restart data set parameter>

```
    -- RESTARTDATASET -- = --<restart data set name>-------------------|
```

<host system parameter>

```
    -- HOSTSYSTEM -- = --<host system name>----------------------------|
```

<boolean parameter>
may be SET or RESET; if neither is specified, SET is assumed. The following boolean parameters may be used:

STATISTICS
When set, STATISTICS causes the Transaction Library to gather statistics concerning the operation of the system. A report summarizing these statistics is produced on a printer file when the last user calls CLOSETRBASE.

<numeric parameter>
may be given a numeric value. The following numeric parameters may be used:

(none yet specified)

<data base parameter)
During its initialization the Transaction Library must access the data base control file; the <data base parameter> gives the data base name, and where to find the control file. If not specified, data base-related code will be omitted from the Transaction Library. The <family name> is the family on which the data base control file resides; if not given, DISK is assumed. The <usercode> is the usercode of the data base control file; if not given, no user code is assumed.

<restart data set parameter>
the <restart data set name> must be given so that the Transaction Library can invoke that data set.

<host system parameter>

the <host system name> must be given if a remote version of the Transaction Library is to be compiled, since the remote system must know what system to communicate with. By definition, the system which contains the data base is the host system.

4. Defaults Statement

<defaults statement> specifies defaults to be used in the absence of required information.

<defaults statement>

```
                       |<--------------- , ---------------|
                       |                                  |
-- DEFAULTS -- ( -----/1\-<control file attributes>----- ) --------|
                         |                             |
                         |-/1\-<data file attributes>----|
                         |                             |
                         |-/1\-<item defaults>-----------|
```

<item defaults>

```
     |<------------------------ , ------------------------|
     |                                                    |
------/1\- ALPHA ----- ( --<initial value option>-- ) -------------|
       |              |
       |-/1\- BOOLEAN -|
       |              |
       |-/1\- NUMBER --|
       |              |
       |-/1\- REAL ----|
```

<item defaults>
specify particular default initial values to be used for any or all of the four data types. These initial values will be used in lieu of standard default values whenever individual initial value specifications are not provided. The standard default initial values for items are:

ALPHA	BLANKS
BOOLEAN	FALSE
NUMBER	0
REAL	0

5. Transaction Format

<transaction format> defines the format of a single transaction record.

<transaction format>

```
    --<format name>------------------ FORMAT --<record description>---|
                    |                |
                    |- TRANSACTION  -|
```

<record description>

```
    --<common part>---------------------------------------------------|
                    |                                 |
                    | |<------------------------|     |
                    | |                         |     |
                    |--- , --<subformat part>---|
```

<common part>

```
    --<subrecord description>-----------------------------------------|
```

<subformat part>

```
    --<subformat name>-- : --<subrecord description>------------------|
```

<subrecord description>

```
    -- ( --<item list>-- ) -------------------------------------------|
                            |                     |
                            |- VERIFY --<condition>-|
```

<item list>

```
         |<------- ; -------|
         |                  |
    ------<data item>-------------------------------------------------|
          |             |    |     |
          |-<group item>-|    |- ; -|
```

<format name>

```
    --<simple identifier>---------------------------------------------|
```

<subformat name>

```
    --<simple identifier>---------------------------------------------|
```

<format name>
consists of at most 17 letters and digits, beginning with a letter. Each <format name> is assigned a unique integer value, which is stored in the Control Part of the transaction record (the "TRFORMAT" Control Item). The <format name> may be used in an application program, in certain contexts, to refer to this value.

<record description>
a <common part> and, optionally, one or more <subformat part>s. There may be at most 1023 <subformat part>s in a <record description>.

<common part>
represents a portion of a transaction record which contains user data common to all transaction records of a given format.

<subformat part>
represents a portion of a transaction record which contains user data for that subformat only.

<subformat name>
consists of at most 17 letters and digits, beginning with a letter. Each <subformat name> is assigned an integer value, which is stored in the Control Part of the transaction record (the "TRSUBFORMAT" Control Item). The <subformat name> may be used in an application program, in certain contexts, to refer to this value.

<subrecord description>
a list of item descriptions, optionally followed by a verify condition. For any transaction record, the applicable verify condition(s) must yield the value TRUE when evaluated using the items of that record. Note that there may be no verify condition, or one, or two, for any given record. If any condition yields the value FALSE, the record is not acceptable.

6. Items

<group item>

```
    --<group item name>-- GROUP -- ( --<item list>-- ) ---------------->

    >-----------------------------------------------------------------|
       |                  |
       |-<occurs option>-|
```

<data item>

```
    ----<alpha item>--------------------------------------------------|
       |                |  |                                       |
       |-<boolean item>-|  | |<-------------- , ---------------|  |
       |                |  | |                                 |  |
       |-<number item>--|  |------/1\-<occurs option>------------|
       |                |        |                           |
       |-<real item>----|        |-/1\-<initial value option>-|
       |                |
       |-<field item>---|
```

<alpha item>

```
    --<alpha item name>-- ALPHA -- ( --<integer>-- ) ------------------>

    >-----------------------------------------------------------------|
       |                                              |
       |- SIZE VARYING -------------------------------|
       |                       |                      |
       |                       |- WITH --<numeric item name>-|
       |                                              |
       |- SIZE DEPENDING ON --<numeric item name>-----|
```

<boolean item>

```
    --<boolean item name>-- BOOLEAN ----------------------------------|
```

<number item>

```
    --<number item name>-- NUMBER ------------------------------------->

    >- ( ---------<integer>-------------------- ) ---------------------|
           |     |            |               |
           |- S -|            |- , --<integer>-|
```

<real item>

```
    --<real item name>-- REAL ----------------------------------------->

    >-----------------------------------------------------------------|
       |                                                       |
       |- ( ---------<integer>-------------------- ) -|
              |     |            |               |
              |- S -|            |- , --<integer>-|
<field item>

    --<field item name>-- FIELD ----------------------------------------->

    >- ( ---<integer>------------------------------ ) ------------------|
         |                                          |
         | |<--------------- ; ---------------|  |
         | |                                  |  |
         |---/48\---<bit name>-------------------|
                                   |           |
                                   |- BOOLEAN -|


<initial value option>

    -- INITIALVALUE ----------- BLANKS ----------------------------------|
                      |     | |           |
                      |- = -| |-<literal>-|


<occurs option>

    -- OCCURS --<integer>------------------------------------------------>
                          |         |
                          |- TIMES -|

    >-------------------------------------------------------------------|
       |                                      |
       |- DEPENDING ON --<numeric item name>-|


< ... item name>

    --<simple identifier>------------------------------------------------|
```

<group item>
defines the properties of a group item. A group item is an item that contains other items. When referenced as an item, a group item behaves like an alpha item of the same length. Unlike an alpha item, however, a group item may not have an initial value of its own; its initial value is determined from the initial values of its elementary items. If a group item has an <occurs option>, then each item within the group "inherits" that <occurs option>.

<alpha item>
defines the properties of an alpha item. An alpha item is a string of <integer> EBCDIC characters. Both the "SIZE VARYING" and the "SIZE DEPENDING" options specify data compaction. For "SIZE VARYING", trailing blanks in the alpha item will be eliminated prior to writing the record to a transaction data file. If "WITH <numeric item name>" is also specified, the number of "significant" characters will be stored by the System in the named item. For "SIZE DEPENDING", the number of significant characters is the value of the given numeric item, or the declared size, whichever is less. It is the user program's responsibility to set the numeric item to the proper value. The default initial value is BLANKS.

<boolean item>
defines the properties of a boolean item. A boolean item is a single 4-bit HEX digit, representing only the truth values of TRUE or FALSE. The default initial value is FALSE.

<number item>
defines the properties of a number item. A number item is a string of 4-bit HEX digits, representing a numeric value. The maximum size, including sign, is 23 digits. If "S" is specified, a sign digit is added; otherwise only the absolute value is stored. The first <integer> is the total number of digits stored. The second <integer>, if present, is the scale factor. The default initial value is zero.

<real item>
defines the properties of a real item. A real item is a word, representing a numeric value in one of two forms. The first form, specified without the "(...)", is internal binary floating point. The second form is a binary integer. If "S" is specified, the sign is maintained; otherwise only the absolute value is stored. The first <integer> represents the significance of the item; it is not used by the system, but is maintained for documentation purposes. The second <integer>, if present, is the scale factor. The default initial value is zero.

<field item>
defines the properties of a field item. A field item is a field of bits, stored as a string of 4-bit HEX digits. The first form will accommodate <integer> bits. The <integer> cannot exceed 48. The second form will contain up to 48 bits, each individually accessible as a boolean value by using the appropriate <bit name>. The

default initial value is all bits off (zero).

<initial value option>
specifies the value a data item is to be given when a record containing that item is created. BLANKS may be specified for alpha items only. The <literal> must be a <string literal> for alpha items, <boolean literal> for boolean items, or a <numeric literal> for numeric items. If the <initial value option> is not given, default initial values will be used.

<occurs option>
specifies that the item to which it applies occurs <integer> times in the transaction record. Most references to the item must be subscripted to indicate which occurrence of the item is intended. The "OCCURS DEPENDING" option specifies data compaction. In this case, only the first n occurrences, where n is the value of the given numeric item, will be retained when writing the record to a transaction data file. It is the user program's responsibility to set the numeric item to the proper value.

< ... item name>
each item name consists of at most 17 letters and digits, beginning with a letter.

Alpha and group items always begin on a byte boundary, while the other items begin on a digit boundary. Whenever necessary, a binary zero digit is used as filler.

The following table defines how each TFL items appears to the application languages:

TFL ITEM	APPLICATION LANGUAGE	
name ALPHA(n)		
	ALGOL:	STRING name
	COBOL:	name PIC X(n) DISPLAY
	PL/1:	name CHAR (n)
name NUMBER(n)		
	ALGOL:	INTEGER name
	COBOL:	name PIC 9(n) COMP-2
	PL/1:	name PICTURE '(n)H'
name NUMBER(Sn)		
	ALGOL:	INTEGER name
	COBOL:	name PIC S9(n) COMP-2
	PL/1:	name PICTURE 'S(n)H'
name NUMBER(n,m)		
	ALGOL:	REAL name
	COBOL:	name PIC 9(n-m)V9(m) COMP-2
	PL/1:	name PICTURE '(n)HV(m)H'
name NUMBER(Sn,m)		
	ALGOL:	REAL name
	COBOL:	name PIC S9(n-m)V9(m) COMP-2
	PL/1:	name PICTURE 'S(n)HV(m)H'
name REAL		
	ALGOL:	REAL name
	COBOL:	name COMP-4
	PL/1:	name DECIMAL FLOAT(11)
name REAL(n)		
	ALGOL:	INTEGER name
	COBOL:	name PIC 9(n) COMP
	PL/1:	name DECIMAL FIXED(n)
name REAL(Sn)		
	ALGOL:	INTEGER name
	COBOL:	name PIC S9(n) COMP
	PL/1:	name DECIMAL FIXED(n)
name REAL(n,m)		
	ALGOL:	REAL name
	COBOL:	name PIC 9(n-m)V9(m) COMP
	PL/1:	name DECIMAL FIXED(n,m)
name REAL(Sn,m)		
	ALGOL:	REAL name
	COBOL:	name PIC S9(n-m)V9(m) COMP
	PL/1:	name DECIMAL FIXED(n,m)
name BOOLEAN		
	ALGOL:	BOOLEAN name
	COBOL:	name BOOLEAN
	PL/1:	name BIT(1)
name FIELD(n)		
	ALGOL:	REAL name
	COBOL:	name FIELD SIZE IS n BITS
	PL/1:	name BIT(n)
name GROUP		
	ALGOL:	STRING name
	COBOL:	name
	PL/1:	name

7. Condition, Expression

```
<condition>

      |<------------ AND ------------|
      |          |<- OR --|          |
      |                              |
    --------------<simple condition>-------------------------------------|
        |- NOT -|
```

```
<simple condition>

    ----<boolean variable>-----------------------------------------------|
      |                                            |
      |-<boolean literal>--------------------------|
      |                                            |
      |- ( --<condition>-- ) ----------------------|
      |                                            |
      |-<expression>--<relation op>--<expression>-|
```

```
<expression>

              |<-<arithmetic op>-|
              |                  |
    ------------<simple value>-------------------------------------------|
      |     |
      |- + -|
      |     |
      |- - -|
```

```
<simple value>

    ----<variable>-------------------------------------------------------|
      |                         |
      |-<literal>---------------|
      |                         |
      |- ( --<expression>-- ) -|
```

```
<arithmetic op>

    ---- + --------------------------------------------------------------|
      |       |
      |- - ---|
      |       |
      |- * ---|
      |       |
      |- / ---|
      |       |
      |- DIV -|
      |       |
      |- MOD -|
```

<relational op>

```
---- EQL ----------------------------------------------------|
   |       |
   |- = ---|
   |       |
   |- GEQ -|
   |       |
   |- >= --|
   |       |
   |- GTR -|
   |       |
   |- > ---|
   |       |
   |- LEQ -|
   |       |
   |- <= --|
   |       |
   |- LSS -|
   |       |
   |- < ---|
   |       |
   |- NEQ -|
   |       |
   |- ¬= --|
```

<condition>
a boolean expression, used in a verify clause. Any items referenced in a condition must exist in the transaction record to which the verify clause applies.

<simple condition>
a boolean primary. A <boolean variable> is a fully-subscripted boolean item. In a <simple condition> of the form <expression> <relation op> <expression>, both expressions must be of the same type; that is, both must be numeric, or both must be alpha. If alpha, the expressions may be only alpha items or alpha literals; if the two alpha expressions are of different length, the shorter is assumed to contain trailing blanks (this is the COBOL rule for comparing ALPHA items).

8. Transaction Subbase
-- ----------- -------

<transaction subbase> lists transaction formats which are to be invoked together by an application program. Such formats would most likely be related in some way. For example, each subbase may correspond to a logical data base, so that a program that invokes only a subbase need only cause a specific portion of the data base to be opened.

<transaction subbase>

```
--<subbase name>------------------- SUBBASE ----------------------->
                            |                 |
                            |- TRANSACTION -|

          |<------- , -------|
          |                  |
>- ( ---<subbase format>--- ) -------------------------------------|
                                   |                          |
                                   |- , --<guard file name>-|
```

<subbase name>

```
--<simple identifier>----------------------------------------------|
```

<subbase format>

```
----------------------------<format name>-------------------------->
   |                      |
   |-<new format name>-- = -|

>------------------------------------------------------------------|
   |                                       |
   |      |<--------- , --------|          |
   |      |                     |          |
   |- ( ---<subbase subformat>--- ) -|
```

<subbase subformat>

```
--------------------------------<subformat name>-------------------|
  |                          |
  |-<new subformat name>-- = -|
```

<subbase name>
consists of at most 17 letters and digits, beginning with a letter. Each <subbase name> is assigned a unique integer value, which is stored in the Control Part of the transaction record (the "TRSUBBASE" Control Item). The <subbase name> may be used in an application program, in certain contexts, to refer to this value.

<guard file name>
may be associated with a subbase. The system will verify that the usercode and program name are permitted access to the subbase by that guard file. See separate documentation concerning security aspects of the Transaction System.

<subbase format>
identifies one of several transaction formats to be invoked as part of a particular transaction subbase. The <new format name>, if present, becomes the <format name> in the application program. The list of <subbase subformat>s, if present, restricts the invoked subformats to those listed; otherwise, all subformats are invoked.

<subbase subformat>
identifies one of several subformats to be invoked along with a format of a transaction subbase. The <new subformat name>, if present, becomes the <subformat name> in the application program.

9. Transaction Journal

<transaction journal> defines attributes of control and data files for a given <journal name>. Any transaction journal not defined here will be given default attributes.

<transaction journal>

```
    --<journal name>------------------- JOURNAL ----------------------->
                                 |           |
                                 |- TRANSACTION -|

              |<--------------- , ---------------|
              |                                  |
    >- ( -----/1\-<control file attributes>----- ) -------------------|
                |                              |
                |-/1\-<data file attributes>----|
```

<journal name>

```
    --<simple identifier>-------------------------------------------|
```

<control file attributes>

```
    -- CONTROL FILE -- ( --<control file attribute list>-- ) ----------|
```

<data file attributes>

```
    -- DATA FILE -- ( --<data file attribute list>-- ) ---------------|
```

<control file attribute list>

```
       |<----------------------- , -----------------------|
       |                                                  |
    ------/1\- AREAS -- = --<integer>----------------------------------|
         |                                                 |
         |-/1\- AREASIZE -- = --<integer>-- BLOCKS -------|   |- , -|
         |-/1\- BLOCKSIZE -- = --<integer>--- SEGMENTS ---|
         |                               |            |    |
         |                               |- WORDS ----|    |
         |-/1\- FAMILY -- = --- DISK --------------------|
         |                  |          |                   |
         |                  |- PACK -----|                 |
         |                  |-<packname>-|                 |
         |-/1\- USERCODE -- = ---<usercode>---------------|
         |                    |           |                |
         |                    |- * --------|               |
         |-/1\- CHECKSUM --- = -- TRUE -------------------|
                         |             |
                         |- = -- FALSE -|
<data file attribute list>

        |<----------------------- , -----------------------|
        |                                                  |
------/1\- AREAS -- = --<integer>------------------------------------|
          |                                            |     |      |
          |-/1\- AREASIZE -- = --<integer>-- BLOCKS -------|     |- , -|
          |                                            |
          |-/1\- BLOCKSIZE -- = --<integer>--- SEGMENTS ---|
          |                                |            |
          |                                |- WORDS ----||
          |                                            |
          |-/1\- FAMILY -- = --- DISK ----------------------|
          |                    |           |           |
          |                    |- PACK -----|           |
          |                    |           |           |
          |                    |-<packname>-|           |
          |                                            |
          |-/1\- USERCODE -- = ---<usercode>---------------|
          |                      |          |          |
          |                      |- * --------|          |
          |                                            |
          |-/1\- DUPLICATED ON --- DISK --------------------|
          |                      |          |          |
          |                      |- PACK -----|          |
          |                      |          |          |
          |                      |-<packname>-|          |
          |                                            |
          |-/1\- CHECKSUM --- = -- TRUE --------------------|
                            |              |
                            |- = -- FALSE -|
```

A <journal name> consists of at most 17 letters and digits, beginning with a letter.

A transaction journal is a collection of disk or pack files. There may be several transaction journals for any given transaction base. All have the transaction base name as their first directory name, followed by the journal name.

Each transaction journal contains the following physical files:

Transaction Data Files
A sequence of files which contain variable-length transaction data records in variable-length blocks. Each file has a title of the form <base name>/<journal name>/<file-number>, where <file-number> ranges from 1 thru 9999, wrapping around from 9999 to 1. A new file is created whenever the current file becomes full, or whenever the SWITCHTRFILE procedure in the Transaction Library is called.

Transaction Control File
This file contains "control" information, including a Halt/Load flag (to indicate recovery is required), and the current value of the <file-number>. For each user "known" to the Transaction System, the Control File contains a reference to either his last good input transaction record or last good restart transaction record, depending on whether he was using a restart transaction, and a copy of its response record. The Control File has a title of the form <base name>/<journal name>/CONTROL.

Transaction journals are used for two purposes: transaction tanking and transaction history. A tanking journal contains transaction records accumulated during a tanking operation, but not yet processed (applied to the data base). A transaction base may have several tanking journals. They may have any journal name of the user's choice except "TRHISTORY".

A history journal contains transaction records actually applied to the data base. They are written in the order in which they must be re-processed in a recovery operation. A transaction base has only one history journal; its journal name is always "TRHISTORY". There is no physical difference between tanking and history journals.

The standard default attribute values for both control and data files are the following:

AREAS	100
AREASIZE	30 BLOCKS
BLOCKSIZE	900 WORDS
FAMILY	DISK
USERCODE	none
DUPLICATED	not duplicated
CHECKSUM	TRUE

The USERCODE and FAMILY attributes permit the control and/or data files to be located independently of the user running the application program.

Only the data files may be duplicated. They contain original data which cannot otherwise be recovered if lost, whereas the control file can,be recovered by the TRUTILITY program. If specified, USERCODE applies to the duplicated file as well as the primary file.

10. Update

A Transaction Base can be modified by making changes to the TFL source and generating a new TRDESCRIPTION file. The entire Transaction Base must be redescribed in TFL with the desired changes.The UPDATE statement is inserted at the beginning and instructs TFL to update an existing TRDESCRIPTION file. When a TFL update is run file HOST must be label equated to the old TRDESCRIPTION.
Example :

```
RUN SYSTEM/TFL;
FILE CARD = PATCH/TEST;
FILE TAPE = TFL/TEST;
FILE HOST = TEST/TRDESCRIPTION;
```

The user may make any change to the items of a subformat or format. This generates a new format and invalidates the previous one. If the format name remains the same, then the new format has the same format number but can still be distinguished from the old format by the Transaction library. If the name is changed, or a format is deleted and another added, then the old format number is invalid and a new format number is assigned.

The Transaction base update level is incremented when a format is altered, added or deleted. The Transaction library is automatically recompiled when the update level is incremented. Although the user's Update library is not automatically recompiled, it must match the update level of the Transaction library. Therefore it is the user's responsibility to ensure that the Update library is recompiled also.

Application programs that do not use the invalidated formats can continue to run without being recompiled. Programs using an old format will get an error upon passing one to the Transaction library.

Global parameters and options may also be changed. The DATABASE, RESTARTDATASET, and HOSTSYSTEM specifications may all be changed through an update. The Transaction library will be recompiled to incorporate the changes, but applications programs need not be recompiled.

The specifications for the default journal and the transaction journals may be updated in the same manner. FAMILY, USERCODE and CHECKSUM of the Control File attribute list may be changed, but AREAS, AREASIZE and BLOCKSIZE may not. All Data File attributes may be changed, including the physical attributes. If the AREASIZE is changed, the Transaction library will create a new file with the new attributes, rather than continuing to add to an existing data file.

Subbases may be changed by renaming the subbase, and adding or deleting formats or subformats. Any subbase change deletes the previous subbase and adds a new one with a new subbase number. Any program creating a transaction record in an old subbase may pass the transaction through the Transaction library as long as that format is still valid. If the Update library is routing transactions based on the subbase number in the transaction record, the code should be altered to reflect the new subbase numbers assigned by the update.

11. The TFL Processor

The TFL Processor is a program, SYSTEM/TFL, which reads the TFL symbolic description of a Transaction Base, checks the description for proper syntax, and produces a description file called <base name>/TRDESCRIPTION. Depending upon dollar-card options specified by the user, SYSTEM/TFL performs other functions as well. It produces a printer listing of the source, including errors which have been detected; it creates an updated symbolic file; and it initiates compilation of tailored software.

11.1 Files

SYSTEM/TFL employs the following files (note that they are very similar to compiler files):

CARD this is the primary input file and is required. It is a card-image file, containing the entire TFL source description of a Transaction Base, unless the MERGE option is used; in that case, CARD contains only patches to the TAPE file.

TAPE this is the secondary input file. When the MERGE option is used, TAPE contains the TFL source and CARD contains patches. The title should be label-equated by the user.

HOST this is the previous description file used as input when the UPDATE statement is used. The title should be label-equated by the user.

CODE this is the primary output file. This is the description file which will be produced if no errors are detected. Its title, <base name>/TRDESCRIPTION, is set by SYSTEM/TFL.

LINE this is the printer output file. It contains various output, depending upon options set by the user.

NEWSOURCE
this is the updated source file, produced when the option NEW is set. The title should be label-equated by the user.

11.2 Dollard-Card Options

The following dollar-card options may be employed by the user (the default settings are given in parentheses):

ALLINFO (FALSE)
if TRUE will set the TEXT, LAYOUT, and DECKLIST options.

DEBUG (FALSE)
sets the TRACE BOOLEAN and TRACE will evoke the debugging function.

DECKLIST (FALSE)
if TRUE will list the ZIP file produced to compile the tailored software.

ERROR LIMIT (10 if via CANDE; otherwise 100)
is used to set the maximum number of errors.

HEADER (FALSE)
if set will generate the header blocks.

INCLNEW (FALSE)
if TRUE will include the source from the included file on the NEWSOURCE file.

INCLUDE
same as the ALGOL INCLUDE.

LAYOUT (FALSE)
if TRUE will generate listing of all items and their offsets.

LIST (FALSE if via CANDE; otherwise TRUE)
sets the LISTB BOOLEAN. If LISTB is FALSE no listing will be generated (except for errors). If TRUE then normal listings will be generated.

LISTP (FALSE)
if TRUE will list the PATCH cards only.

MERGE (FALSE)
if TRUE will include cards from a file called SOURCE.

NEW (FALSE)
if TRUE then a NEWSOURCE file is generated.

OMIT if TRUE will omit source file input.

PAGE will cause a page eject when it is encountered.

POP pops a 48 bit stack, thereby returning the BOOLEAN to its previous value.

RESET
pushes a 48 bit stack with a FALSE.

SET pushes a 48 bit stack with a TRUE.

SEQ (FALSE)
if TRUE will cause the output file to be resequenced. When SEQ is set and no <sequence base> and <sequence increment> are specified, sequencing begins at 1000 and continues in increments of 1000. Default action can be overridden by specifying a one-to-eight digit <sequence base> and <sequence increment>. If a <sequence base> is given, but no <sequence increment> is specified, sequencing begins at <sequence base> and an increment of 1000 is used. If a <sequence increment> is specified, but no <sequence base> is given, sequencing begins at 1000 and the specified increment is used.

SEQERR (FALSE)
if TRUE will make sequence errors Syntax errors.

SINGLE (TRUE)
if the LIST option is set then the generated listing will be single spaced.

TEXT (FALSE)
if set will cause a formatted listing of the generated texts.

VOIDT
same as ALGOL VOIDT function.

WARNING (TRUE)
if TRUE will allow the printing of warning messages for COBOL reserved words.

ZIP (TRUE)
if TRUE and if no syntax errors, will ZIP a job file to compile all tailored software.

12. Transaction Record Format

A transaction record consists of several words of control information, followed by the items defined by the user in the TFL description. The control information is read-only for an application program; it is assigned values either during execution of the CREATE statement, or by the Transaction Library. These control items are described elsewhere in this document.

12.1 Format in Main Memory

(all sizes are given in bytes)

Word	Bits	Control Item Name	Defined by	Content
0	47:01	TRLRBIT	Tran. Library	left-right bit
	46:15	unused		all bits zero
	31:08	TRCONTROLSIZE	CREATE stmt.	size of control part (=48)
	23:08	TRFORMATLEVEL	CREATE stmt.	format level
	15:16	TRTOTALSIZE	CREATE stmt.	control+data size (=48+n)
1	47:48	TRBASEDTS	CREATE stmt.	TIME(6) date-time stamp
2	47:48	TRUPDATEDTS	CREATE stmt.	TIME(6) update dts
3	47:20	TRUPDATELEVEL	CREATE stmt.	update level
	27:04	TRSTATE	Tran. Library	tran. state
	23:12	unused		all bits zero
	11:12	TRFILENUM	Tran. Library	file number
4	47:32	TRBLOCKNUM	Tran. Library	block number
	15:16	TROFFSET	Tran. Library	word offset
5	47:08	unused		all bits zero
	39:16	TRDATASIZE	CREATE stmt.	size of data part (=n)
	23:12	TRFORMAT	CREATE stmt.	format number
	11:12	TRSUBFORMAT	CREATE stmt.	subformat number
6	47:08	TRDISKCONTROL SIZE	Tran. Library	size of control part (=18) (on disk)
	39:18	unused		all bits zero
	11:12	TRCOUNTER	Tran. Library	tran. counter
7	47:01	TRREJECTED	Tran. Library	rejected flag

46:02	TRCOMPACTED	Tran. Library	compacted flag
44:02	TRRESTARTINFO	Tran. Library	restart info flag
42:11	unused		all bits zero
31:12	TRSUBBASE	CREATE stmt.	subbase number
19:20	TRUSERNUM	Tran. Library	user id number

8 first word of user data (n bytes total)

If the total record length in words, (TRTOTALSIZE+5) DIV 6, is greater than 1023 words, the arrays which hold transaction records may be segmented. the total record length may never exceed 16K words.

12.2 Format on Disk

When a transaction record is written to disk (to a transaction journal), word zero is repeated at the end to improve error detection.

word	content	
0	left control word	(word 0 above, with TRLRBIT = 1)
1-7	control information	(words 1-7 above)
8	data items	(n bytes, starting with word 8 above)
	...	
	right control word	(word 0 above, with TRLRBIT = 0)

The left and right control words are identical except for a single bit to distinguish left from right.

12.3 "SYSTEMTR" Transaction

There is one transaction format, "SYSTEMTR", which is reserved for use by the System; it is defined automatically in all transaction bases and subbases. Within that format, the following subformats are also defined:

REMOTEREQUEST
used to communicate between remote and host systems.

SYSTEMRESPONSE
created by the Transaction Library whenever a response is required but not otherwise provided (for example, after an error in a transaction processor).

USERINFO
contains a brief summary of all user informations contained in the Control File. It is written at the end of each Data File, and is used during recovery of the Control File.

CHANGEUSER
used to "audit" changes to the user information in the Control File. It is written in the Data File when each change occurs (CREATETRUSER and PURGETRUSER), and is used during recovery of the Control File.

III. TRANSACTION SYSTEM APPLICATION LANGUAGE SYNTAX

The user interface to the Transaction System consists of certain constructs implemented in the application languages (ALGOL, COBOL, PL/I), and a set of special procedures contained in the Transaction Library. The language constructs, defined here, permit the user to invoke a transaction base, declare and manipulate transaction records, and pass transaction records to procedures. The Transaction Library procedures are described elsewhere in this document.

1. Invocation of Transaction Base

A transaction base must be invoked in a user program before any references are made to formats or items defined by that transaction base. Invocation consists of specifying the transaction base to be invoked and, optionally, a list of transaction formats and subformats. The user may also give alternate internal names, to be used by his program, for the transaction base and any of the formats or subformats invoked.

ALGOL

```
-- TRANSACTION BASE --<base-spec>---------------------------------------------|
                                        |                      |
                                        |-  :  --<format-list>-|
```

<base-spec>

```
--------------------------------------------------------<base name>-----------|
  |                      |  |                          |
  |-<int-baseid>--  =   -|  |-<subbase name>--  OF    -|
```

<format-list>

```
  |<-----------------------  ,  ------------------------|
  |                                                     |
----<format-spec>-------------------------------------------------------------|
                     |                                |
                     |-  (  --<subformat-list>--  )  -|
```

<format-spec>

```
------------------------------<format name>-----------------------------------|
  |                         |
  |-<int-formatid>--  =    -|
```

<subformat-list>

```
---- ALL ---------------------------------------------------------------------|
  |                          |
  |- NONE -------------------|
  |  |<-------  ,  -------|  |
  |  |                    |  |
  |---<subformat-spec>-------|
```

<subformat-spec>

```
---------------------------------<subformat name>-----------------------------|
  |                            |
  |-<int-subformatid>--   =   -|
```

<base name>, <subbase name>, <format name>, <subformat name>
identifiers declared in the TFL source for the transaction description file.

<int-baseid>, <int-formatid>, <int-subformatid>
identifiers used as internal names by this program instead of the actual names.

COBOL

A new section in the Data Division called the "TRANSACTION SECTION" contains the invocations of all transaction bases to be referenced in a COBOL program.

The TRANSACTION SECTION must appear between the DATA-BASE SECTION and the WORKING-STORAGE SECTION.

```
                         |<----------------|
                         |                 |
-- TRANSACTION SECTION. ---<tr-base-invoke>--------------------|
```

<tr-base-invoke>

```
-- TB --<base-spec>--------------------------------------------|
                   |                        |
                   |- USING --<format-list>-|
```

<base-spec>
same as ALGOL.

<format-list>
same as ALGOL.

PL/I

```
---- TRBASE ------------<base-spec>---------------------------->
  |                    |
  |- TRANSACTION BASE -|

>--------------------------------------------------------------|
  |                                   |
  |- FORMAT -- ( --<format-list>-- ) -|
```

<base-spec>

```
----------------------------<base name>------------------------|
  |                      | |                                  |
  |- INTNAME --- = ------| |- <subbase name> ( <base name> ) -|
              |          |
              |- INVOKE -|
```

<format-list>
same as ALGOL, except that INVOKE may also be used in place of "=".

2. Declaration of Transaction Record Variables

A transaction record is an array row which may contain the transaction data of one of several transaction formats. A transaction record variable names one of these array rows. It must be associated with one transaction base, and may contain only formats and subformats which have been invoked from that base. Note that the size of the array row will be large enough to accommodate the largest format of all those invoked for it.

ALGOL

```
------------- TRANSACTION RECORD -- ( --<baseid>-- ) -------------->
   |          |
   |- LONG   -|

   |<---- , ---|
   |           |
>---<trvarid>-------------------------------------------------------|


-- TRANSACTION RECORD ARRAY -- ( --<baseid>-- ) ------------------->

   |<--------------------- , ---------------------|
   |                                              |
>---<trarrayid-list>-- [ --<bound-pair-list>-- ] ------------------|
```

```
<trarrayid-list>

   |<----- , ----|
   |             |
----<trarrayid>-----------------------------------------------------|


<bound-pair-list>

   |<---------- , ---------|
   |                       |
----<a-exp>-- : --<a-exp>-------------------------------------------|
```

<a-exp>
An arithmetic expression.

<trvarid>
An identifier which is to be the name of a transaction record variable.

<trarrayid>
An identifier which is to be the name of an array of transaction record variables.

<baseid>
the (internal) name of a transaction base invoked in this program.

COBOL

Each transaction record for a given transaction base is declared immediately after the appropriate TB entry in the Transaction Section using COBOL's 01 level indicator.

```
-- 01 --<tr-record-id>--------------------------------------------|
                          |                                |
                          |- OCCURS --<integer>-----------|
                                                  |        |
                                                  |- TIMES -|
```

PL/I

```
-- DECLARE --<trvarid-list>---------------------------------------->
                                  |                    |
                                  |-<dimension attribute>-|

>--- TRANSACTION RECORD --- ( --<baseid>-- ) -- ; -----------------|
   |                      |
   |- TRREC --------------|
```

```
<trvarid-list>

   ----<trvarid>---------------------------------------------------|
     |                                 |
     |        |<---- , ---|            |
     |        |           |            |
     |- ( ---<trvarid>--- ) -|
```

```
<dimension attribute>

                           |<--- , --|
                           |         |
   ------------------ ( ---<bound>--- ) ---------------------------|
     |                |
     |- DIMENSION -|
```

```
<bound>

   -----------------------<expression>-----------------------------|
     |                       |                      |
     |-<expression>-- : -|                      |
     |                                              |
     |- * ------------------------------------------|
```

Use of the <dimension attribute> implies that a transaction record array is being declared. If it is used, the keyword **DIMENSION** may be omitted only if the attribute appears immediately following the <trvarid-list>. If it follows the "(<baseid>)", then the keyword must be used. The asterisk may be used only for parameter specifications and means that the bounds of the actual array passed will be used.

3. Creation of Transaction Records

The contents of a transaction record variable are undefined until it is initialized with a **CREATE** statement to a particular format. **CREATE** will assign the initial values of all items in the format (and subformat, if given) to the record variable; control items are initialized as well. The record variable will continue to contain the given format until it is re-initialized with a subsequent **CREATE**. It is never "cleared" by the system.

ALGOL

```
-- CREATE --<trrec>-- . --<formatid>-------------------------------|
                                       |                  |
                                       |- . --<subformatid>-|
```

```
<trrec>

   ----<trvarid>---------------------------------------------------|
     |                                                   |
     |-<trarrayid>-- [ --<subscript-list>-- ] -|
```

<trrec>
the transaction record variable to be initialized. A transaction record array element must be fully subscripted.

<formatid>, <subformatid>
the format and subformat (if given) whose initial values are to be assigned to the record variable.

COBOL

```
-- CREATE ---------------------------<formatid>---------------------->
             |                        |
             |-<subformatid>--- OF -|
                               |      |
                               |- IN -|

>--- OF ---<tr-record-id>-------------------------------------------|
  |       |                |                        |
  |- IN -|                 |- ( --<subscript>-- ) -|
```

PL/I

```
-- CREATE ---<trvarid>--------------------------------------------->
             |                                  | |                |
             |- <trarrayid> - (subcript list) -| |- . <formatid> -|

>---------------------- ; ---------------------------------------|
  |                    |
  |- . <subformatid> -|
```

Either the <formatid> or <subformatid> must appear. If necessary for qualification, both may be used. The subcripts follow the usual PL/I rules and may follow any of the identifiers.

4. Transaction Record Data Items
-- ----------- ------ ---- -----

A transaction record may contain a transaction of any format declared for it. When it contains such a transaction, data items in the format of that transaction may be referenced as follows:

ALGOL

```
--<trrec>--------------------- . --<tr-itemid>--------------------->
           |                  |
           |- . --<formatid>-|

>-----------------------------------------------------------------|
  |                              |
  |- [ --<subscript-list>-- ] -|
```

<trrec>
fully-subscripted transaction record variable.

<formatid>
optional, unless required for qualification.

<tr-itemid>
item of the format presently occupying the record variable; must be fully subscripted.

This construct may be used as the left part of an assignment or replace statement, or as a primary in an expression. The type of the item (ALPHA, NUMBER, etc.) must be consistent with the context in which it is used, as follows:

ALPHA	STRING
NUMBER	REAL
REAL	REAL
FIELD	REAL
BOOLEAN	BOOLEAN
GROUP	STRING

COBOL

```
--<tr-itemid>--- IN ------------------------------------------->
               |      | |                     |
               |- OF -| |-<formatid>--- OF -|
                                      |      |
                                      |- IN -|

>-<tr-record-id>-- <COBOL-subscr> -------------------------------|
```

<COBOL-subscr>
Empty if neither the record variable nor the item is subscripted. Otherwise, it is the list of subscripts for the record variable followed by the list of subscripts for the item, the whole list being enclosed in parentheses.

REFERENCES:
Transaction section data items are considered as normal data items and may be referenced at any place in the procedure division acceptable for a normal data item.

CORRESPONDING OPTION:
Several special rules apply to the use of the CORRESPONDING option in referencing transaction section items. A transaction record name alone is not considered as a legitimate group name. Instead, a transaction format name, a transaction subformat name, or a subordinate group data item must be referenced. When a transaction format is referenced, only the data items in the common portion of the format will be considered as eligible to correspond.

PL/I

Data Items

```
--<trrec>---------------------------------------- . <tritem-id> ->
          |                 | |                    |
          |- . <formatid> -| |- . <subformatid> -|

>----------------------------------------------------------------|
  |                    |
  |- (subcript list) -|
```

This construct may be used on the left side of an assignment or as a primary in an expression. The types to which it will be converted are:

ALPHA	CHARACTER
NUMBER	PCN 'H'
REAL (n)	DECIMAL FIXED (n)
REAL	DECIMAL FLOAT (")
BOOLEAN	BIT (1)
GROUP	STRUCTURE
FIELD	BIT (n)

The subscripts may follow the identifier according to the usual PL/I rules.

5. Transaction Record Control Items
-- ----------- ------ ------- -----

Control items are system-defined items contained in every transaction record. These items are maintained by the transaction system, and are read-only in an application program. Some control items are of little interest to the user program; others must be used in order to process the transaction records properly. These items are defined only after a transaction record has been CREATEd with a particular format.

The control items and their meanings are as follows:

TRTOTALSIZE
Total size in bytes of the transaction record. Includes control and data parts. Size of the control part is a constant for all records.

TRCONTROLSIZE
Size of the control part in bytes.

TRSIZE (COBOL ONLY)
Return the actual size, in bytes, of the transaction record array row.

TRFORMATLEVEL
Format level of the transaction record layout. Currently defined to be 1.

TRBASEDTS
Date-time stamp (TIME(6)) used to co-ordinate the transaction description, user program, and Transaction Library.

TRUPDATEDTS
Update date-time stamp. Defines update level within the transaction base date-time stamp; also used to co-ordinate the various code files.

TRUPDATELEVEL
TFL update level. Used to eliminate masksearch of a value array when checking TRUPDATEDTS.

TRCOUNTER
A counter used to synchronize input and response transaction records between remote and

host systems.

TRFILENUM, TRBLOCKNUM, TROFFSET
The "record address" of a transaction record in a transaction file. Assigned by the Transaction Library when the transaction is first written to the transaction journal.

TRDATASIZE
The length, in bytes, of the user data portion of the record.

TRLRBIT
Bit to distinguish left from right control words on disk.

TRFORMAT
Unique numeric value which identifies the transaction format.

TRSUBFORMAT
Numeric value which identifies the transaction subformat; unique for a given TRFORMAT value.

TRSUBBASE
Unique numeric value which identifies the transaction subbase invoked.

TRDISKCONTROLSIZE
Size of the control portion of the transaction record on disk.

TRUSERNUM
Identifies the source of a sequence of transaction records, primarily to aid in recovery and restart after an abort or Halt/Load. The last good input record, and its response, are saved for each value of TRUSERNUM.

TRREJECTED
Used by the Transaction Library to mark a transaction record which has already been written to the history file, and is subsequently rejected by the system when faults occur during processing.

TRCOMPACTED
Used by the Transaction Library to mark a transaction record which is in its compacted form (always 0 for transactions which cannot be compacted).

ALGOL

```
--<trrec>-- . --<control-item-name>---------------------------------|
```

<trrec>
fully-subscripted transaction record variable.

<control-item-name>
one of the control items defined above. In DMALGOL only, control items may be changed just like data items.

COBOL

```
--<control-item-name>-- ( --<trrec>-- ) ---------------------------|
```

PL/I

Same as for COBOL.

6. Transaction Compile-time Functions
-- ----------- ------------ ---------

Transaction compile-time functions provide access to certain properties of transaction formats which are constant at compile time.

ALGOL

```
--<tr-c-t-function name>-- ( --<tr-c-t-function argument>-- ) -----|
```

<tr-c-t-function argument>

```
   -------------------<formatid>------------------------------------->
     |               |              |                    |
     |-<baseid>-- . -|              |- . --<subformatid>-|

   >-----------------------------------------------------------------|
     |                   |
     |- . --<tr-itemid>-|
```

Not all transaction compile-time functions apply to all arguments. The following specifies which functions apply to the possible arguments, and what the function represents:

<formatid>

TRFORMAT
Numeric value assigned to the <formatid>.

TRDATASIZE
Size of the common part of the transaction record in bytes.

<formatid>.<subformatid>

TRSUBFORMAT
Numeric value assigned to the <subformatid>.

TRDATASIZE
Size of the subformat record (including common part) in bytes.

<formatid>.<tr-itemid> or <formatid>.<subformatid>.<tr-itemid>
---------------------- -- -------------------------------------

TRBITS
Size of the item in bits.

TRDIGITS
Size of the item in digits.

TRBYTES
Size of the item in bytes.

TROCCURS
Number of occurrences of the item. Zero, if not specified as occurring in TFL source.

The values for TRBITS, TRDIGITS, and TRBYTES are computed according to the following table (note that it is not always the actual size of the item, but a value computed from the size):

Item Type	TRBITS	TRDIGITS	TRBYTES
ALPHA(N)	N*8	N*2	N
BOOLEAN	1	1	1
BOOLEAN (in FIELD)	1	1	1
NUMBER(N)	N*4	N	(N+1) DIV 2
NUMBER(N,M)	N*4	N	(N+1) DIV 2
NUMBER(SN,M)	N*4	N	(N+1) DIV 2
REAL	48	12	6
REAL(N)	N*4	N	(N+1) DIV 2
REAL(N,M)	N*4	N	(N+1) DIV 2
REAL(SN,M)	N*4	N	(N+1) DIV 2
FIELD(N)	N	(N+3) DIV 4	(N+7) DIV 8
FIELD(N BOOLEANS)	N	(N+3) DIV 4	(N+7) DIV 8
GROUP	N*4	N=TOTALSZ	N DIV 2

COBOL

```
   --<tr-c-t-function name>-- ( --<tr-c-t-function argument>-- ) -----|
```

<tr-c-t-function argument>

```
   -------------------------------------------------------<formatid>------->
    |                     |  |                              |
    |-<tr-itemid>--- OF -|  |-<subformatid>--- OF -|
                      |      |                     |        |
                      |- IN -|                     |- IN -|

   >------------------------------------------------------------------|
    |                   |
    |- OF ---<baseid>-|
    |       |
    |- IN -|
```

PL/I

Same as ALGOL, except that any of the four names may or may not appear, as needed for qualification.

7. Transaction Record Variables as Parameters
-- ----------- ------ --------- -- ----------

Most of the work in using transactions is carried out by Transaction Library procedures, to which transaction records are passed as parameters.

The formal and actual parameters must agree on transaction base, but need not specify the same list of transaction formats. Of course, if a procedure is given a transaction record in a format it has not invoked, the procedure is limited in what it can do with that record.

Transaction records may not be passed to intrinsics or to external procedures being initiated via a CALL or PROCESS statement, or to procedures which are bound.

ALGOL

The specification syntax for a formal transaction record variable is identical to the declaration syntax, except that the <bound-pair-list> is replaced by a <lower-bound-list> as for any arrays. Note that the transaction base must have been invoked prior to this procedure declaration.

COBOL

A transaction record is passed as a parameter in the same way that any 01 entry is passed.

PL/I

The declaration syntax for transaction record parameters is the same as normal transaction record declarations. For arrays, bounds specified as "*" may be used. For constant bounds, compile time checking is done to see that the formal and actual parameters match.

8. Transaction Record Variable Assignment
-- ----------- ------ -------- ----------

A transaction record variable may be assigned (copied) to another transaction record variable, provided they represent the same transaction base.

ALGOL

```
--<trrec-1>-- = --<trrec-2>----------------------------------------|
```

In DMALGOL (the implementation language for the Transaction Library), it is permitted to assign a transaction record variable to an ARRAY reference variable. This construct is not permitted in application programs.

COBOL

```
-- MOVE -- <tr-rec2> TO <tr-rec1> ---------------------------------|
```

PL/I

```
    --<trrec-1>-- = --<trrec-2>-- ; -----------------------------------|
```

9. Restrictions
-- ------------

Certain restrictions are enforced by the compilers on the usage of transaction record variables (as opposed to item references). Except as described above, it is not possible to use transaction record variables without a syntax error. For example, it is impossible to use them in lists, READ or WRITE statements, assignment statements except as described above, as primaries in expressions, etc.

10. Checks
--- ------

The host compilers check that all the usages of a particular transaction variable within a code file are compatible. The parameter checking code checks that the following attributes of the transaction variable are those which were expected when the variable is passed as a parameter to a separately compiled code file: format level, control length, and transaction base date-time stamp. Note that the contents of the variable do not have to be inspected in order to make this check.

The Transaction Library insures that its TFL update level is greater than or equal to that of both the user program and the Update Library. It does this by indexing a value array by the update level to retrieve an update date-time stamp, and comparing it for equality with the update date-time stamp in the record variable. They must match. If the update level is out of bounds of the value array, the program (or update library) is too old or too new with respect to the Transaction Library.

11. TRINTERFACE
--- -----------

The host compilers communicate with an external coroutine SYSTEM/TRINTERFACE in order to process the syntax described here.

12. Transaction Use Procedures
--- ----------- --- ----------

COBOL programs intended for the Update library may declare untyped parameterless procedures as parameters by naming them in the USING clause of the Procedure Division header and declaring them as a section header in the Declaratives. The CALL statement is used to invoke these procedures. The syntax for the section header of a transaction procedure is:

<sectionname> SECTION. USE AS TRANSACTION PROCEDURE.

An example:

```
            PROCEDURE DIVISION USING TR1, TR2.
            DECLARATIVES.
            TR1 SECTION. USE AS TRANSACTION PROCEDURE.
            TR2 SECTION. USE AS TRANSACTION PROCEDURE.
            END DECLARATIVES.
            P1. CALL TR1.
                CALL TR2.
                  .
                  .
                  .
```

13. Example Application Programs
--- ------- ----------- --------

Skeleton application programs in ALGOL and COBOL are shown below to demonstrate how the Transaction Base and library entrypoints are declared in each language.

ALGOL

```
BEGIN   % SAMPLE BATCH PROGRAM USING TRANSACTIONS

    TRANSACTION BASE TRB = BANKTR;
    LIBRARY L(TITLE="BANKTR/CODE/HOSTLIB.");
```

```
% DECLARE ALL ENTRYPOINTS TO BE USED


INTEGER PROCEDURE CREATETRUSER(IDSTRING, IDNUM);
    STRING IDSTRING; INTEGER IDNUM;
    LIBRARY L;
INTEGER PROCEDURE PURGETRUSER(IDNUM);
    INTEGER IDNUM;
    LIBRARY L;
INTEGER PROCEDURE LOGONTRUSER(IDSTRING, IDNUM);
    STRING IDSTRING; INTEGER IDNUM;
    LIBRARY L;
INTEGER PROCEDURE LOGOFFTRUSER(IDNUM);
    INTEGER IDNUM;
    LIBRARY L;
INTEGER PROCEDURE RETURNRESTARTINFO(IDNUM, TROUT);
    INTEGER IDNUM;
    TRANSACTION RECORD (TRB) TROUT;
    LIBRARY L;
INTEGER PROCEDURE RETURNLASTRESPONSE(IDNUM, TROUT);
    INTEGER IDNUM;
    TRANSACTION RECORD (TRB) TROUT;
    LIBRARY L;
INTEGER PROCEDURE TANKTRNORESTART(IDNUM, TRIN);
    INTEGER IDNUM;
    TRANSACTION RECORD (TRB) TRIN;
    LIBRARY L;
INTEGER PROCEDURE PROCESSTRNORESTART(IDNUM, TRIN, TROUT);
    INTEGER IDNUM;
    TRANSACTION RECORD (TRB) TRIN, TROUT;
    LIBRARY L;
INTEGER PROCEDURE OPENTRBASE(USEROPTION, TIMEOUT);
    INTEGER USEROPTION, TIMEOUT;
    LIBRARY L;
INTEGER PROCEDURE CLOSETRBASE;
    LIBRARY L;
INTEGER PROCEDURE SEEKTRANSACTION(FILENUM, BLOCKNUM, OFFSET);
    INTEGER FILENUM, BLOCKNUM, OFFSET;
    LIBRARY L;
INTEGER PROCEDURE READTRANSACTION (TRREC);
    TRANSACTION RECORD (TRB) TRREC;
    LIBRARY L;
INTEGER PROCEDURE SWITCHTRFILE;
    LIBRARY L;

 % DECLARE TRANSACTION RECORD VARIABLES TO BE USED

    TRANSACTION RECORD (TRB)
         TRIN,
         TROUT,
         LASTINPUT,
         LASTRESPONSE;
    STRING JOURNALNAME;

         .
         .
         .
         .
%   START OF PROGRAM
%   SET LIBPARAMETER IN DECLARATION OR BEFORE FIRST CALL ON ENTRYPOINT

    L.LIBPARAMETER := JOURNALNAME;

%   BODY OF PROGRAM

          .
          .
          .

END.
```

COBOL

```
 IDENTIFICATION DIVISION.
 ENVIRONMENT DIVISION.
        .
        .
        .
 DATA DIVISION.
 FILE SECTION.
        .
        .
        .
 DATA-BASE SECTION.
 TRANSACTION SECTION.
```

```
* SPECIFY THE TRANSACTION FORMAT NAMES TO BE USED

TB  TRB = BANKTR
    USING STATIS=STATUS, CREATEACCT, PURGEACCT, DEPOSIT,
         WITHDRAWAL, CHANGEACCT, STATEMENT, TEST,
         RESTARTDETANKER


* SPECIFY THE TRANSACTION RECORD VARIABLES

01  TRIN.
01  TROUT.
01  LASTINPUT.
01  LASTRESPONSE.
WORKING-STORAGE SECTION.
    .
    .
    .
******************************************************************
PROCEDURE DIVISION.
MAIN SECTION.
    .
    .
    .
STOP RUN.
******************************************************************
*
*
CREATETRUSER.
    CALL "CREATETRUSER         OF BANKTR/CODE/HOSTLIB"
       USING ID-X , IDNUM
       GIVING TPS-RESULT.
*
*
PURGETRUSER.
    CALL "PURGETRUSER          OF BANKTR/CODE/HOSTLIB"
       USING ID-X
       GIVING TPS-RESULT.
*
*
LOGONTRUSER.
    CALL "LOGONTRUSER          OF BANKTR/CODE/HOSTLIB"
       USING ID-X, IDNUM
       GIVING TPS-RESULT.
*
*
LOGOFFTRUSER.
    CALL "LOGOFFTRUSER         OF BANKTR/CODE/HOSTLIB"
       USING IDNUM
       GIVING TPS-RESULT.
*
*
RETURNRESTARTINFO.
    CALL "RETURNRESTARTINFO    OF BANKTR/CODE/HOSTLIB"
       USING IDNUM, TROUT
       GIVING TPS-RESULT.
*
*
RETURNLASTRESPONSE.
    CALL "RETURNLASTRESPONSE    OF BANKTR/CODE/HOSTLIB"
       USING IDNUM, TROUT
       GIVING TPS-RESULT.
*
*
TANKTRNORESTART.
    CALL "TANKTRNORESTART       OF BANKTR/CODE/HOSTLIB"
       USING IDNUM, TRIN
       GIVING TPS-RESULT.
*
*
PROCESSTRNORESTART.
    CALL "PROCESSTRNORESTART    OF BANKTR/CODE/HOSTLIB"
       USING IDNUM, TRIN, TROUT
       GIVING TPS-RESULT.
*
*
OPENTRBASE.
    CALL "OPENTRBASE            OF BANKTR/CODE/HOSTLIB"
       USING OPT, TIMEOUTV
       GIVING TPS-RESULT.
*
*
CLOSETRBASE.
    CALL "CLOSETRBASE           OF BANKTR/CODE/HOSTLIB"
```

```
       GIVING TPS-RESULT.
*
*
SEEKTRANSACTION.
    CALL "SEEKTRANSACTION      OF BANKTR/CODE/HOSTLIB"
       USING FILENUMBER, BLOCKNUMBER, OFFSETS
       GIVING TPS-RESULT.
*
*
READTRANSACTION.
    CALL "READTRANSACTION      OF BANKTR/CODE/HOSTLIB"
       USING TRIN
       GIVING TPS-RESULT.
*
*
SWITCHTRFILE.
    CALL "SWITCHTRFILE         OF BANKTR/CODE/HOSTLIB"
       GIVING TPS-RESULT.
*
*******************************************************************
```

IV. THE TRANSACTION LIBRARY

1. Introduction

The Transaction Library is a collection of procedures which constitute the heart of the Transction System. A standard symbolic is provided, which, when compiled by the user, produces a library code fiie tailored for a given Transaction Base. The Transaction Library contains all the procedures which are called by user-written application programs to process or tank transaction records, and to read them back from journals.

In a shared resources system, the Transaction Library must reside on the host system (that system which contains the data base), where it is known as the Host Library. A second version of the Transaction Library, the Remote Library, resides on any remote system which may interface with the host system. The Remote Library, described later in the document, appears identical to the Host Library from the application program's viewpoint; it communicates with the host system where the work is actually performed. This section describes the host version of the Transaction Library.

2. Library Initiation

As with any library, the first attempt to call a library procedure by an application program causes the MCP to connect the program to the library stack. Physically, the Transaction Library consists of a D2 (and D1) stack, shared by all users of the Transaction Base; a D3 stack for each active transaction journal for that Base, each shared by all users of that journal; and a private D4 stack for each running application program. The first call of a procedure in the Transaction Library causes any of the required shared stacks not already active, and the private D4 stack, to be initiated.

Prior to the first procedure call, the library parameter must be assigned. The parameter is, in this case, the journal name for the journal which is to be accessed. This determines which of perhaps several D3 stacks will be used by the program. For transaction processing, the journal name must be "TRHISTORY"; for tanking it may be any journal except TRHISTORY. For reading, any journal may be used.

3. Definition of Transaction User

A "transaction user" is the source of a sequence of transaction records for a particular journal. It may be a program, or one of several terminals serviced by a program, or both, whichever is convenient at any given time. The transaction user concept is intended primarily as an aid to program restart.

Each transaction user is identified by a string (IDSTRING), unique among all transaction users of that journal. The transaction user becomes "known" when CREATETRUSER is called, passing IDSTRING. CREATETRUSER verifies that the transaction user is not already known, and assigns a unique integer value, IDNUM, to represent the transaction user. IDSTRING must be an EBCDIC string of at most 17 characters. It is suggested that the string be strictly alphanumeric, although that is not a requirement of the system.

A transaction user remains known until PURGETRUSER is called. This may be in the same program, or days or weeks later, or not at all.

A transaction user is "active", however, only during the execution of a program. To become active, the program must pass a known IDSTRING to LOGONTRUSER; if the transaction user is not already active, LOGONTRUSER will mark the transaction user active and return the original IDNUM to the program. Only active transaction users are permitted to submit transaction records to the System. A transaction user becomes "inactive" by calling LOGOFFTRUSER, PURGETRUSER or CLOSETRBASE, or at the termination of the program.

A program may make several transaction users active by calling LOGONTRUSER several times, each time with a different IDSTRING. It is then the program's responsibility to pass the appropriate IDNUM value whenever it calls one of the library procedures requiring that parameter.

The system saves the IDNUM parameter in the control part of each transaction record submitted by the transaction user so that its source may be identified later, if necessary. In addition, a reference to the restart record, and the response record corresponding to the last transaction, for update processing, are saved in the control file of the TRHISTORY journal for each transaction user. For tanking, a reference to the last tanked transaction record is saved in the control file of the tank journal for each transaction user. These records may be retrieved at any time to facilitate job restart.

It is important to note that transaction users are known to a journal, and not to the entire Transaction System. A given IDSTRING may be known to one journal and not another; and if known to more than one, the IDNUM value is unlikely to be the same for all journals.

4. Transaction Library Procedures

The following procedures are the external entry points in the Transaction Library, to be called by the application program.

```
INTEGER PROCEDURE OPENTRBASE(USEROPTION, TIMEOUT);
    INTEGER USEROPTION, TIMEOUT;
```

This must be the first procedure in the Transaction Library called by an application program. The Transaction System is initiated, and the transaction journal (specified by the library parameter in the application program) is opened and prepared for subsequent use.

The TIMEOUT parameter is used only by the remote version of the Transaction Library; it indicates how long (in seconds) the Remote System is to wait for the Host System to recover in the event that System fails. If that time is exceeded, an exception is returned to the program.

The USEROPTION parameter determines what functions the application program is permitted to do (that is, what procedures may be called). USEROPTION is interpreted as follows (note that the given integer values must be passed; the names "UPDATE", "TANK", etc., are unknown to the compiler in this context):

Update (1)
Transactions records are to be processed (applied to the data base). All procedures except TANKTRANSACTION, READTRANSACTION and SEEKTRANSACTION may be called. This mode is permitted only for the TRHISTORY journal.

Inquiry (2)
Same as Update, except only data base inquiry may be performed; that is, only those transactions which perform no data base update operations may be used. Update transactions will fail when BEGIN-TRANSACTION is attempted.

Tank (3)
Transactions records are to be tanked, and saved for future data base update. All procedures except PROCESSTRANSACTION and its variants, READTRANSACTION and SEEKTRANSACTION may be called. This mode is not permitted for the TRHISTORY journal.

Read (4)
Transaction records are to be read from the journal. All procedures except CREATETRUSER, PURGETRUSER, TANKTRANSACTION, PROCESSTRANSACTION, and SWITCHTRFILE may be called. While a journal is in this mode, no other process may open the journal. The journal is not actually opened until SEEKTRANSACTION is called to specify the file number and position.

Exclusiveupdate (5)
This is equivalent to Update (1) with the additional constraint that NO other updaters are allowed. This might be useful for large batch updating where transactions are not guaranteed to be independent of all others, and therefore must run alone.

```
INTEGER PROCEDURE CREATETRUSER(IDSTRING, IDNUM);
    STRING IDSTRING;
    INTEGER IDNUM;
```

Used to identify a new transaction user for this journal. IDSTRING must be a unique string, not already known for this journal; a unique integer value (IDNUM) is assigned by the system to represent IDSTRING. IDSTRING must be an EBCDIC string of at most 17 characters.

```
INTEGER PROCEDURE PURGETRUSER(IDNUM);
    INTEGER IDNUM;
```

Used to "forget" a transaction user previously identified by CREATETRUSER. After PURGETRUSER is called, the transaction user is no longer known by this journal, and information about his transactions is discarded.

```
INTEGER PROCEDURE TRUSERIDSTRING(IDSTRING, IDNUM);
    INTEGER IDNUM;
    STRING IDSTRING;
```

Returns in the parameter IDSTRING the user identification string which corresponds to the value of the parameter IDNUM.

```
INTEGER PROCEDURE LOGONTRUSER(IDSTRING, IDNUM);
    STRING IDSTRING; INTEGER IDNUM;
```

Used to make a transaction user active for this program. IDSTRING must be a unique identification of that transaction user, already "known" to the Transaction System from a previous call on CREATETRUSER, and not yet active. The original integer assigned for this IDSTRING is returned via the parameter IDNUM. Note that there may be only one user of a given IDNUM active at a time. IDSTRING must be an EBCDIC string of at most 17 characters.

```
INTEGER PROCEDURE LOGOFFTRUSER(IDNUM);
    INTEGER IDNUM;
```

IDNUM must represent an active transaction user for this program. That transaction user will be marked not active.

```
INTEGER PROCEDURE RETURNRESTARTINFO(IDNUM, TRREC);
    INTEGER IDNUM;
    TRANSACTION RECORD (TRBASE) TRREC;
```

Used to help restart an application program. IDNUM must represent an active transaction user for this program. The restart transaction record corresponding to the last good transaction for this user is returned as the parameter TRREC.

```
INTEGER PROCEDURE RETURNLASTRESPONSE(IDNUM, TRREC);
    INTEGER IDNUM;
    TRANSACTION RECORD (TRBASE) TRREC;
```

Used to help restart an application program. IDNUM must represent an active transaction user for this program. The response transaction record corresponding to the last good transaction of this user is returned as the parameter TRREC.

```
INTEGER PROCEDURE PROCESSTRANSACTION(IDNUM, TRIN, TROUT, RESTARTTR);
    INTEGER IDNUM;
    TRANSACTION RECORD (TRBASE) TRIN, TROUT, RESTARTTR;
```

Used to send an input transaction record (TRIN) to a transaction processor. IDNUM must represent an active transaction user for this program. A response transaction record is returned to the program as the parameter TROUT. If successful, that is, if the exception flag returned by the procedure is 0, the response is that created by the transaction processor; otherwise, the response is a transaction with SYSTEMTR format and SYSTEMRESPONSE subformat, which will indicate the nature of the error. Here "successful" means that the transaction appeared to observe all requirements of the Transaction System; it may in fact not have been successful from the application's viewpoint. In that case, the response would have to indicate the problem. The parameter RESTARTTR is a transaction record which is intended to contain user-defined restart information. It is saved for the user, and can be retrieved by the RETURNRESTARTINFO procedure in the event a program restart is necessary.

```
INTEGER PROCEDURE PROCESSTRNORESTART(IDNUM, TRIN, TROUT);
    INTEGER IDNUM;
    TRANSACTION RECORD (TRBASE) TRIN, TROUT;
```

Similar to PROCESSTRANSACTION except that no restart transaction record is passed. In this case, the input transaction, TRIN, is saved in lieu of a restart record.

```
INTEGER PROCEDURE PROCESSTRFROMTANK(IDNUM, TRIN, RESTARTNUM, RESTARTTR);
    INTEGER IDNUM, RESTARTNUM;
    TRANSACTION RECORD (TRBASE) TRIN, RESTARTTR;
```

Similar to PROCESSTRANSACTION, except that a different user number is used for restart purposes than the user number of the input transaction. IDNUM will be assigned to TRIN; RESTARTNUM will be assigned to RESTARTTR. This procedure is intended primarily for use in processing transactions from a tank file. In this case, the output transaction TROUT of the other procedures is not returned, since it is of no interest to the detanking procedure.

```
INTEGER PROCEDURE TANKTRANSACTION(IDNUM, TRIN, RESTARTTR);
    INTEGER IDNUM;
    TRANSACTION RECORD (TRBASE) TRIN, RESTARTTR;
```

Used to tank a transaction record. IDNUM must represent an active transaction user for this program. RESTARTTR is a transaction record which is intended to contain user-defined restart information. It can be retrieved by the RETURNRESTARTINFO procedure if program restart is necessary.

```
INTEGER PROCEDURE TANKTRNORESTART (IDNUM,TRIN);
    INTEGER IDNUM;
    TRANSACTION RECORD (TRBASE) TRIN;
```

Similar to TANKTRANSACTION except that no restart transaction record is passed. In this case, the input transaction, TRIN, is saved in lieu of the restart record.

```
INTEGER PROCEDURE READTRANSACTION(TRREC);
    TRANSACTION RECORD (TRBASE) TRREC;
```

Read the next transaction record from the transaction journal and move it to TRREC. If the transaction record is marked "REJECTED", or if it is the last transaction record in the current physical file, an exception flag (in the ATTENTION group) is returned along with the record; if there are no more transaction records at all, an exception flag is returned instead of a record. READTRANSACTION can be called only after SEEKTRANSACTION has opened and positioned a specific file.

```
INTEGER PROCEDURE SEEKTRANSACTION(TRFILE, TRBLOCK, TROFFSET);
    INTEGER TRFILE, TRBLOCK, TROFFSET;
```

The parameters represent the "address" of a transaction record in the transaction journal. The current record pointer is set so that the next read operation will read the transaction record at that address. TRBLOCK = 0 selects the first record in the given file (TROFFSET is ignored in this case); TRBLOCK > 0 and TROFFSET = 0 selects the first record in the given block. TRFILE must always specify a valid file number. Except for the cases just described, the parameters must refer exactly to the left control word of a valid transaction record; otherwise, no seek is performed and an exception is returned. SEEKTRANSACTION must be called prior to the first READTRANSACTION call in order to open and position a specific file.

The user can obtain the address of a particular transaction that he wishes to seek to by using the TRUTILITY search function. TRUTILITY will print the address of whatever transactions are located according to the search specification.

INTEGER PROCEDURE SWITCHTRFILE;

Used to force a file switch on the current data file of the journal. The file is closed, the file number is incremented by one, and the next file in sequence is created. The next write to the journal will occur on the new file.

INTEGER PROCEDURE CLOSETRBASE;

Terminate use of the Transaction System. All active transaction users for this program are marked inactive; all subsequent calls on transaction procedures are illegal until OPENTRBASE is called again.

5. Exception Reporting

The value returned by these procedures is an execption flag. If the procedure performs as expected, then the exception flag is zero. If the exception flag is non-zero, then the value identifies an exception. The exception values have been arranged into groups of exceptions related in some way. The following groups have been defined:

Value	Meaning	
0	no exception.	
001 - 099	Attention: not necessarily an error, but some condition exists which the program should notice.	
	1: encountered end-of-file: no transaction returned.	
	2: transaction record returned has TRREJECTED = 1.	
	3: no restart transaction record for this IDNUM.	
	4: no response transaction record for this IDNUM.	
100 - 199	Remote-Host interface errors: remote system cannot communicate with host system.	
	100: Host system is not known to the network.	
	101: Host system is down.	
	102: Host system was down, is now available.	
	103: Response signal out of sync with request signal.	
200 - 299	Bad parameter: some parameter has improper value.	
	200: unknown USEROPTION parameter.	
	201: unknown IDNUM parameter.	
	202: IDNUM not logged on.	
	203: RESTARTNUM not logged on.	
	204: IDSTRING already created.	
	205: IDNUM already logged on.	
	206: invalid address parameters (SEEKTRANSACTION).	
300 - 399	Not permitted: procedure called out of order, or function out of context.	
	301: OPENTRBASE already performed by this program.	
	302: cannot PROCESS with tank journal.	
	303: cannot TANK with TRHISTORY journal.	
	305: OPENTRBASE has not yet been performed.	
	306: journal must be open for input.	
	307: journal must be open for output.	
	308: SEEK not permitted.	
	309: there is no response maintained for tank journal.	
	310: UPDATE USEROPTION not permitted for tank journal.	
	311: INQUIRY USEROPTION not permitted for tank journal.	
	312: USEROPTION conflicts with current use of journal.	
	313: transaction base is unusable after fatal error.	
	314: Exclusive open not allowed when transaction base in use.	
	315: Open update not permitted when transaction base in exclusive use.	
400 - 499	Rejected: transaction rejected by Transaction Library (not processed).	
	400: transaction not yet created.	
	401: invalid TRBASEDTS	(input record)
	402: invalid TRUPDATELEVEL	(input record)
	403: invalid TRUPDATEDTS	(input record)
	404: unknown TRFORMAT	(input record)
	405: format verify condition FALSE	(input record)
	406: unknown TRSUBFORMAT	(input record)
	407: subformat verify condition FALSE	(input record)
	408: unknown TRSUBBASE	(input record)
	409: failed guardfile check.	(input record)
	451: invalid TRBASEDTS	(restart record)
	452: invalid TRUPDATELEVEL	(restart record)
	453: invalid TRUPDATEDTS	(restart record)
	454: unknown TRFORMAT	(restart record)
	455: format verify condition FALSE	(restart record)
	456: unknown TRSUBFORMAT	(restart record)
	457: subformat verify condition FALSE	(restart record)

```
            458: unknown TRSUBBASE                (restart record)
            459: failed guardfile check.          (restart record)
500 - 599 Failed: transaction failed in transaction processor.
            501: failed after BEGIN-TRANSACTION.
            502: failed after SAVEINPUT.
            503: failed after MID-TRANSACTION.
            504: failed after SAVERESPONSE.
            505: did not call SAVERESPONSE.
            551: invalid TRBASEDTS                (response record)
            552: invalid TRUPDATELEVEL            (response record)
            553: invalid TRUPDATEDTS              (response record)
            554: unknown TRFORMAT                 (response record)
            555: format verify condition FALSE    (response record)
            556: unknown TRSUBFORMAT              (response record)
            557: subformat verify condition FALSE (response record)
            558: unknown TRSUBBASE                (response record)
            559: failed guardfile check.          (response record)
900 - 999 System error: A software, hardware or procedural error has occurred.
            901: maximum library stacks already in use.
            902: error while accessing journal control file.
            903: data base out of sync with TRHISTORY jorunal.
            904: error in EXPANDTRANSACTION.
            905: error in COMPACTTRANSACTION.
            906: USERINFO transaction expected.
            907: USERINFO out of sync with journal control file.
            908: cannot locate known user in USERINFO.
            909: error in SEEKTRANSACTION.
            910: cannot position TRHISTORY file for rerun.
```

6. Compilation of the Transaction Library

The Transaction Library compilation may be initiated automatically by SYSTEM/TFL whenever a new description file is created. Otherwise, the user may compile the Library by compiling the source file TRBASE/HOSTLIB with the DMALGOL compiler. All required dollar-card options are included in that source file. The compiler file DESCRIPTION must be label-equated by the user to the appropriate description file (TRDESCRIPTION/<base name>).

V. THE TRANSACTION UPDATE LIBRARY

The Update Library is a collection of user written transaction processing routines. The library contains the data base declaration and ALL of the code within the transaction system that performs data management statements against the data base. The Update Library is a private library, so that each user that invokes it will get a separate stack, and a separate SIB after the data base has been opened.

1. Locking, Consistency, and Reproducibility

Because DMS II data bases can be updated concurrently by many programs, the concept of consistency and how to achieve it are important. Because DMS II recovery backs out some previously completed transactions, reproducibility of the previously obtained results is an important issue. DMS II provides record level locking as a tool to aid in achieving consistency and reproducibility. These issues are discussed in this section. The next section gives a detailed discussion of how to program the Update Library.

1.1. Consistency

Every data base has a set of assertions which the data must satisfy. Although few of the assertions are ever stated formally anywhere, they are extremely important. When the data satisfy all the assertions (or consistency constraints), the data base is said to be consistent. Otherwise, it is said to be inconsistent.

Every transaction which updates the data base must preserve the consistency constraints when run alone. (Otherwise, the transaction is considered to be incorrect.) Even though all transaction routines preserve all the consistency constraints when run alone, that is no guarantee that all multiprogrammed mixes of these transactions will also preserve the consistency constraints. Record level locking is the tool DMS II provides for use by the transaction routines so that all the consistency constraints will be preserved in a multiprogramming mix.

How is record level locking to be used to preserve consistency? Fortunately, there is a solution which is characterized very simply: When an arbitrary mix of transaction routines is run concurrently, all transactions must be two-phase in order to preserve consistency. Unless all transactions are two-phase, it may be possible for the transactions routines to execute in such a manner that consistency is not preserved. In fact, if all transactions are two-phase, no transaction will ever see the consistency constraints violated except as it temporarily violates them itself during the course of its update.

What is a two-phase transaction? As explained above in the definitions in Section I, a transaction is two-phase if it can be divided into a growing phase followed by a shrinking phase. During the growing phase, it locks records but does not free any. During the shrinking phase, it frees records but does not lock any. Furthermore, if a record is to be updated, it must be retrieved with an exclusive lock. Records which are retrieved but not updated can be shared among readers (or locked exclusively), but must be locked against retrieval by potential updaters.

Two-phase transactions have the following ramifications for DMS II.

1) FIND cannot be used at all, only LOCK.
2) Since LOCK obtains the record exclusively, records cannot be shared among readers.
3) Care must be taken, because DMSII does an implicit FREE when appropriate before retrieving the requested record. Thus, a LOCK may be equivalent to a FREE followed by a LOCK, which is clearly a violation of the two-phase condition. If more than one record in a data set must be locked, the data set must be invoked more than once because of the implicit FREE. DELETE followed by a selection expression is equivalent to a LOCK followed by DELETE <data set name>. Thus, care must also be taken in using DELETE so that unwanted implicit FREEs are not executed.
4) Since ENDTRANSACTION frees all records, execution of the next two-phase transaction is facilitated.
5) It may be possible to minimize the potential bottleneck caused by records locked by many transactions. (An example of such a record would be a record containing a data item which holds a total--for example, a bank total in a banking application.) This can be done by locking such records INSIDE transaction state as the last record locked in the transaction, updating it, STOREing it, and immediately FREEing it. (Note: a deadlock cannot occur when locking a record if no records are ever locked after it by any transaction.) This scheme keeps the record locked for the shortest possible time.

To illustrate the concept of consistency, consider the following example. Suppose the consistency constraint to be preserved is that data item A of record R1 is equal to data item B of record R2. The following two transaction routines, T1 and T2, written in pseudo-COBOL preserve the consistency constraint that A=B when run alone.

```
T1:
   LOCK R1.
   COMPUTE A=A+100.
   STORE R1.
   FREE R1.
   LOCK R2.
   COMPUTE B=B+100.
   STORE R2.
   FREE R2.

T2:
   LOCK R1.
   COMPUTE A=A*2.
   STORE R1.
   FREE R1.
   LOCK R2.
   COMPUTE B=B*2.
   STORE R2.
   FREE R2.
```

When T1 and T2 are run concurrently, the consistency constraint may or may not be preserved. The following schedule happens to preserve consistency.

```
T1:                           T2:
   LOCK R1.
   COMPUTE A=A+100.
   STORE R1.
   FREE R1.
                                 LOCK R1.
                                 COMPUTE A=A*2.
                                 STORE R1.
                                 FREE R1.
   LOCK R2.
   COMPUTE B=B+100.
   STORE R2.
   FREE R2.
                                 LOCK R2.
                                 COMPUTE B=B*2.
                                 STORE R2.
                                 FREE R2.
```

It is not difficult to prove that if A=B initially, then if T1 and T2 execute with the following schedule, A will not be equal to B upon termination. (In fact, A will be equal to B+100.) Thus, the following schedule destroys consistency.

```
T1:                           T2:
   LOCK R1.
   COMPUTE A=A+100.
   STORE R1.
   FREE R1.
                                 LOCK R1.
                                 COMPUTE A=A*2.
                                 STORE R1.
                                 FREE R1.
                                 LOCK R2.
                                 COMPUTE B=B*2.
                                 STORE R2.
                                 FREE R2.
   LOCK R2.
   COMPUTE B=B+100.
   STORE R2.
   FREE R2.
```

T1 and T2 both violate the two-phase condition, because "FREE R1" is followed by "LOCK R2". If "FREE R1" were eliminated, then both T1 and T2 would satisfy the two-phase condition, and consistency would always be preserved. Note that if R1 and R2 were records from the same data set, then the data set would have to be invoked twice to eliminate the "FREE R1"--if R1 and R2 used the same workarea, "LOCK R2" would perform an implicit "FREE R1".

Achieving the two-phase condition with DMS II can be a problem because of the implicit FREE. It is not convenient to invoke a data set once for every record in the data set that might have to be locked concurrently unless the number of such records is very small. What, then, are the alternatives?

The alternatives to the two-phase condition are programmatic conventions. The two-phase theorem states that if an arbitrary set of transaction routines is two-phase, multiprogramming them will not cause loss of consistency. The key word here is "arbitrary". The user can use programmatic conventions to prevent loss of consistency, for then the mix of transactions is not arbitrary.

The idea behind such programmatic conventions is to always LOCK certain "control" records before retrieving or updating a well defined collection of certain other "protected" records. The control record(s) are kept locked while the protected records are accessed and/or updated. This insures that access to the protected records is single threaded.

One example is LOCK TO MODIFY DETAILS. The master record must be locked before any of its details or any of the details of its details etc. can be updated. Each master record is a control record, and a master record and all its descendants are the protected records. In this case the convention happens to be enforceable by DMS II, but it is only applicable to embedded structures.

Another convention, which is generally applicable, is to always lock a certain disjoint record, such as global data, as the first operation in all transactions. This record is kept locked until the completion of the transaction. This convention may not be pratical for some applications, however, because it effectively single threads update access to the entire data base and, consequently, decreases potential throughput.

Thus, it is expected that in many cases the user will define a locking strategy which is not two-phase but allows a reasonable degree of concurrency in updating the data base. Such strategies can be considered equivalent to two-phase as follows. During the first part of the transaction, control records are locked, and no records are freed. This corresponds to the first phase of a two-phase transaction. Any accesses to the protected records after the last control record is locked until the first FREE of a control record are considered to be in the MIDTRANSACTION phase. The protected records are accessed only in the midtransaction phase. The midtransaction phase corresponds to the point between the phases of a two-phase transaction. The freeing of the control records corresponds to the second phase of a two-phase transaction, and no control records may be locked during this phase.

The midtransaction phase turns out to be very important, as will be discussed in the following subsection on reproducibility.

1.2 Reproducibility

DMS II was designed to allow a high degree of concurrency in order to maximize total throughput. This implies that it is highly desirable to have multiple update programs running concurrently. When multiprogramming is used, the question of reproducibility of results arises. For example, if it is necessary to reload a backup dump of the data base and reprocess transactions, will the results be the same? The answer is, in general, no. If the data base was updated in a multiprogrammed fashion, reproducibility of the previous results will not be achieved unless it was designed into the application from the beginning. This lack of reproduciblity is caused by random timing behavior and the interaction of transactions. DMS II recovery aggravates the problem, because it backs out a number of transactions which must then be reprocessed before new transactions are processed. Reproducibility of previous results when reprocessing backed-out transactions is a problem.

It turns out that the midtransaction phase is the key to reproducibility. Concurrently executing transactions which are in their midtransaction phase are independent of each other. Two-phase transactions have disjoint collections of records locked. Transactions using programmatic conventions equivalent to two-phase have disjoint collections of control records locked, and do not share any protected records. (Note: If DMS II had a type of lock which would enable a record to be shared among readers but protected against access by updaters, the collection of records locked would not be strictly disjoint. However, the transactions would still be independent. None of the records shared by readers could be changed, and, thus, no decisions based on shared records which were made by any of the transactions could be invalidated.)

This independence means that it is easy to construct a serial schedule for executing such concurrent transactions which is equivalent to the original multiprogrammed schedule, viz., they can be executed serially in any order whatsoever. There is no ordering constraint, since the transactions are updating disjoint parts of the data base and do not interact.

Consider two concurrently executing transactions, A and B. Suppose A is in midtransaction phase, and B is in the first phase. Finally, suppose B is waiting on a record which A has locked. Then in general, the results of B may depend upon the results of A. Note, however, that B cannot enter midtransaction phase until A has completed midtransaction phase. Because of the locking strategy, if A is run alone followed by B, then the results will be the same. In fact, it can be proven that the order in which transactions enter midtransaction state determines an order in which the transactions can be run in an equivalent serial schedule.

Thus, midtransaction phase is the key to reproducibility. What the Transaction Processing System does is to have the Update Library transaction processing routine execute a new verb, MIDTRANSACTION, when it is in midtransaction phase. Among other things, this causes a procedure in the Transaction Library to be called which saves the transaction input record in a transaction history file. If the transaction input records in the transaction history file are reprocessed serially in the order in which they are encountered, then the results obtained when originally updating the data base will be reproduced.

There is one more wrinkle to reproducibility. Suppose there is a program error in a transaction processing routine, or an error in the data in a transaction input record such that when that record is processed, a fault always occurs causing an abort recovery. When this transaction record is reprocessed after the backout, an abort will occur again. In order to avoid an infinite loop, the Transaction Processing System notices that a particular transaction record always causes an abort and ultimately rejects it. This introduces a new reproducibility problem, because other transactions may have depended on some records that the erroneous transaction had freed in its second phase. Since the erroneous transaction will ultimately be discarded, the results of the rerun mode can then be different. This is undesirable, because output may already have been sent to remote terminals. This problem is called the terminal reproducibility problem.

There are two solutions to the problem of terminal reproducibility. The first is to have all transactions use the SYNC option at END-TRANSACTION to protect themselves from erroneous transactions. It is expected that this solution will not commonly be used, because of the increased overhead and decreased potential throughput. The other solution is to keep all records (or in the case of programmatic conventions equivalent to two-phase, all control records) locked until the end and let END-TRANSACTION free them. Once END-TRANSACTION has been executed, the transaction can no longer cause an abort, and the records it has modified can not be accesed until then. It is expected that this is the convention that will be followed (in addition to the two-phase condition or an equivalent convention) by the transaction processing routines in the Update Library.

1.3 Summary

In programming the Update Library, it is expected that a programmatic convention equivalent to two-phase will be used, and that all control records will be kept locked until freed by END-TRANSACTION. FIND may not be used (except on protected records). Care must be taken in handling the implicit FREE performed by FIND, LOCK, and DELETE.

In order to minimize the amount of time spent in transaction state, as many records as possible should be LOCKed outside of transaction state. If no records are locked inside transaction state, BEGIN-TRANSACTION is the last command which can return a DEADLOCK exception. Also, this makes deadlock handling as simple as possible--merely branch back to the beginning of the transaction (since the data base could not have been changed outside of transaction state).

2. Programming the Update Library

Although the Update Library is user written, it must conform to certain conventions established by the transaction system. The user is required to provide one procedure as an entry point to the library which has a function flag passed as the first parameter. This function flag indicates which of the five basic functions are to be performed. The entry point must be declared as follows:

```
PROCEDURE ACCESSDATABASE(FUNCTIONFLAG, TRIN, TROUT, SAVEINPUT,
                         SAVEOUTPUT );
VALUE FUNCTIONFLAG;
INTEGER FUNCTIONFLAG;
TRANSACTION RECORD (TRB) TRIN, TROUT;
PROCEDURE SAVEINPUT(); FORMAL;
PROCEDURE SAVEOUTPUT(); FORMAL;
```

Based on the value of FUNCTIONFLAG, one of the following will be performed :

```
OPENDATABASE
     for UPDATE  (functionflag = 1)
     for INQUIRY (functionflag = 2)

UPDATE (functionflag = 3)

FORCEABORT (functionflag = 4)

CLOSEDATABASE (functionflag = 5)
```

The user writing the Update Library is not restricted in how he chooses to implement these 5 functions. For convenience, we will describe them as procedures and give them the names suggested above, although they do not have to be implemented this way.

We have forseen three approaches to setting up the Update Library, and have established guidelines for the user for each approach. The first approach is the general case, where the user will declare and open the ENTIRE data base in the Update Library.

2.1 Method One : Invoke Entire Data Base

The five functions provided via ACCESSDATABASE within the update library must perform the following :

1. OPENDATABASE (update or inquiry)

Each of these procedures must open the entire data base. The Transaction Library expects the open to occur so that recovery will be initiated on the data base if it is required. Note that the OPEN option for update must be "TRUPDATE".

2. UPDATE(TRIN, TROUT, SAVEINPUTTR, SAVERESPONSETR)

This function is called by the Transaction Library to process a transaction (TRIN) which has been passed from the user's application program as a parameter to PROCESSTRANSACTION. The procedure must generate a response transaction (TROUT) which is passed back through the transaction system to the user program. SAVEINPUTTR and SAVERESPONSETR are procedures in the Transaction Library which are passed as formal parameters to the Update Library. These procedures are part of the mechanism by which the transaction system retains some control over what happens in the UPDATELIBRARY, and is able to ensure that recovery will work. The details of the recovery operations are discussed elsewhere in this document.

A user routine that processes a transaction record must execute at most a single BEGINTRANSACTION/ENDTRANSACTION pair. All Update Library routines must use the same locking strategy--either two-phase locking or an equivalent programmatic convention. The records should remain locked until freed by ENDTRANSACTION. The reasons for this are discussed in section V.1 above. The Transaction Processing System will not enforce two-phase locking. If two-phase locking or an equivalent programmatic convention is NOT used, reproducibility of previous results will probably not be achieved upon completion of a recovery.

In addition, the user routines should be programmed so as NOT to depend upon variables or DMS II paths from previous transactions. The routines should be programmed so that the input transaction record has complete information necessary to process the transaction. The Update Library is private. Thus, during normal processing its internal variables will be maintained between successive calls, since it is only attached to one user program. However, after a DMS II recovery occurs, the Transaction Processing System will use a single copy of the Update Library to reprocess backed out transactions. Therefore, this copy of the Update Library will see the requests for the various users interleaved when the Transaction System reprocesses the transactions backed out by DMS II recovery.

For example, suppose the user writes a pair of Update Library routines. The first routine is to process the first member of a data set, and the second is called repetitively to process successive members of the data set until there are no more. Finally, suppose that the data set is to be processed in order of an index-sequential set, S, which has key item K and NO DUPLICATES. Then the second routine should not use "FIND NEXT S". This is because the path of the set S can be changed during rerun mode by transactions interleaved from other users. The routine should instead use "FIND S AT K > OLDVALUE" where OLDVALUE is an item in the input transaction record which the calling user program sets equal the the value processed on the previous transaction. (The Update Library routine would have to return this value in the transaction output record.) This technique makes the Update Library routine independent of the current value of the path for S. If "FIND NEXT S" were used, the routine would appear to work properly during normal processing. It would, however, probably prevent reproducibility after a recovery. Note that this type of error may be difficult to track down.

Update Library routines must execute BEGINTRANSACTION, MIDTRANSACTION, and ENDTRANSACTION in that order, or the Transaction system will return an error. It is the responsibility of the Update Library programmer to ensure that MIDTRANSACTION is executed in the midtransaction phase of the transaction. The SAVEINPUT procedure must be mentioned in the MIDTRANSACTION statement and is implicitly called at that point. Similarly, the SAVEOUTPUT procedure is mentioned in the ENDTRANSACTION statement and is implicitly called at that point.

For inquiry transactions (the only kind permitted for inquiry mode), SAVERESPONSETR must be called explicitly and BEGINTRANSACTION, MIDTRANSACTION and ENDTRANSACTION are not allowed.

```
             .
             .
        < lock all required records >
             .
             .
        BEGINTRANSACTION (TRIN) RDS;
             .
             .
             .
        MIDTRANSACTION (TRIN, SAVEINPUT ) RDS;
             .
             .
             .
        ENDTRANSACTION (TRIN, SAVEOUTPUT ) RDS;
             .
```

The UPDATE procedure will probably be structured so that the input transaction is processed based on its format and subformat type. The following is an example :

```
PROCEDURE UPDATE(TRIN, TROUT, SAVEINPUT, SAVERESPONSE);
...
BEGIN
      ...
      CASE TRFORMAT(TRIN) OF
      BEGIN
(fmt 1):
              .
              .
```

```
(fmt 2):
          .
          .
(fmt n):
          CASE TRSUBFORMAT(TRIN) OF
          BEGIN
     (sfmt 1):
               .
               .
     (sfmt m):
               < process transaction with format n, subformat m >;
               .
               .
          END SUBFORMATS;
          .
          .
          .
     END FORMATS;
     ....

END;
```

3. FORCEABORT

This function must be provided in the Update Library for the Transaction Library to call in order to cause an abort. The procedure must CLOSE the data base (SIB) that was opened in the OPENDATABASE procedure. This procedure will be called only when the data base is in transaction state, so the CLOSE will cause the abort.

4. CLOSEDATABASE

This function also must CLOSE the data base. It will be called as the result of the application program calling CLOSETRBASE procedure or by the Transaction Library if the user forgot to call CLOSETRBASE and he is exiting the library.

In the general case the FORCEABORT and CLOSEDATABASE procedures perform the same function.

2.2 Method two : Invoke Part of the Data Base

The second approach to writing the Update Library arises out of the need to break up the data base into logical data bases and to be able to invoke only a portion of the data base.

Using the previous approach the user must open the ENTIRE data base in OPENDATABASE and close the ENTIRE data base in CLOSEDATABASE. With the second approach the user does not have to strictly conform to the restrictions of the previous method, but must be careful to provide the checks he needs to accomplish the same thing.

1. OPENDATABASE (inquiry or update)

The user must open some portion of the data base here so that recovery will run if needed. If opening the logical data bases is allocated to other procedures, then the user can get by with only invoking the restart data set here. The function flag indicating whether the open was inquiry or update should be saved so that subsequent opens are done consistently.

2. UPDATE(TRIN, TROUT, SAVEINPUTTR, SAVERESPONSETR)

Before processing a transaction, the relevant portion of the data base must be invoked or an invalid op will result. Since the entire data base is not opened in OPENDATABASE, the user is responsible for insuring that the needed portion is opened at some point prior to processing the transaction.

A suggested approach is for the user to group transaction formats into subbases of the transaction base to correspond to logical databases of the data base. Then it can be insured that transaction records in a given subbase will refer to only those structures invoked in the corresponding logical data base.

Using the subbase number in the control part of the transaction record being processed, the Update Library can call the appropriate procedure to handle that subbase. A global for each of the subbases can be kept to tell if the procedure is being called for the first time and if so, an OPEN must be performed on the logical database. This way only those parts of the data base actually being used by this application program will be invoked, and they will be invoked as needed.

3. FORCEABORT

When the Transaction Library is notified that the last transaction did not complete successfully in the Update Library, it calls FORCEABORT to close the SIB that was used by the transaction. Where there are multiple SIBs open, the Update Library must have a global value to keep track of the logical database referred to by the last transaction. Then, if FORCEABORT is called, it knows which portion of the data base (SIB) to close.

4. CLOSEDATABASE

The Update Library must be able to close ALL logical data bases that have been invoked by the application program. Again, a global value for each logical data base would be sufficient to determine which logical data bases need to be closed.

2.3 Method three : Multiple Libraries

A variation to the second approach might involve the use of libraries to divide up the code associated with each logical data base. Since the concept of the Update Library is to have ALL of the code accessing the data base logically in ONE CODEFILE, it is forseeable that a user might generate an enormous codefile for the Update Library. It might be necessary to break up the codefile into pieces and only use the portions that are required at any given time.

This can be easily accomplished with the use of libraries. The Update Library could contain library declarations for many other libraries, which would provide the appropriate UPDATE, OPENDATABASE , FORCEABORT and CLOSEDATABASE functions for groups of transactions records. The Update Library would then route each transaction to the appropriate library that is handling that format. To simplify this process the user can group related transactions together in subbases in TFL and then route each transaction record based on its subbase number.

If the user has related a transaction subbase to a logical data base, then the library that handles a particular subbase would only have to open a logical data base rather than the entire data base.

Since the subbase number has not yet been implemented as a compile time construct (as format numbers have) the actual number given a particular subbase must be written into the program. This will not be necessary when subbase numbers are generated at compile time based on the name of the subbase. Subbase number 1 is always given to the entire transaction base. Thus a transaction created where the entire transaction base was invoked will always get subbase number 1. The user should also be careful to notice any changes in subbase numbers as a result of adding or deleting subbases in a TFL update.

By allocating the task of determining whether a subbase has been opened to the update procedure in the subbase libraries, the Update Library would not have to keep track of what has been opened. Thus, the individual libraries would need to provide only two functions for the Update Library to use, UPDATE and FORCEABORT. Note that the Transaction Library will continue to call the ACCESSDATABASE procedure in the principle Update Library to perform the appropriate function.

2.4 Example of Update Library

Whatever approach the user takes in implementing the Update Library, the library must provide the specific external entrypoint discussed above, and must be compiled as <trbase name>/CODE/UPDATELIB so that the Transaction Library can find it.

An example of the form an Update Library would have is shown below in a "skeleton" written in Algol. Along the lines of method three above the example uses libraries to provide the code which does the actual processing of the transaction records. The code provided via these other libraries can be in any language.

```
 $ SHARING = PRIVATE
BEGIN  % TRANSACTION UPDATE LIBRARY

      LIBRARY DBSUBONE ( TITLE = "TRBASE/UPDATELIB/SUBONE ");

            PROCEDURE ACCESSSUBBASEONE(FUNCTIONFLAG,INQ,TRIN,TROUT,
                      SAVEINPUT, SAVERESPONSE );
                  VALUE FUNCTIONFLAG, INQ;
                  REAL FUNCTIONFLAG, INQ;
                  TRANSACTION RECORD (TRB) TRIN, TROUT;
                  PROCEDURE SAVEINPUTTR(); FORMAL;
                  PROCEDURE SAVERESPONSETR(); FORMAL;
                  LIBRARY DBSUBONE;
```

MARK 3.1

```
     LIBRARY DBSUBTWO ( TITLE = "TRBASE/UPDATELIB/SUBTWO ");

          PROCEDURE ACCESSSUBBASETWO(FUNCTIONFLAG, INQ,TRIN,TROUT,
                   SAVEINPUT,SAVERESPONSETR );
               VALUE FUNCTIONFLAG, INQ;
               REAL FUNCTIONFLAG, INQ;
               TRANSACTION RECORD (TRB) TRIN, TROUT;
               PROCEDURE SAVEINPUTTR(); FORMAL;
               PROCEDURE SAVERESPONSETR(); FORMAL;
               LIBRARY DBSUBONE;


     DEFINE UPDATEV = 1 #,
            FORCEABORTV = 2 #;

     % global variables
     REAL LASTSUBBASE, OPENTYPE;
     ....


     PROCEDURE FORCEABORT;
     BEGIN
          CASE LASTSUBBASE OF
          BEGIN
          ( 2 ) :
               ACCESSSUBBASEONE(FORCEABORTV,SAVEFUNCTIONFLAG,TRIN,TROUT,
                              SAVEINPUT,SAVERESPONSE);
          ( 3 ) :
               ACCESSSUBBASETWO(FORCEABORTV,SAVEFUNCTIONFLAG,TRIN,TROUT,
                              SAVEINPUT,SAVERESPONSE);
               ....
          END OF CASE;
          ...
     END;

     PROCEDURE UPDATE(TRIN,TROUT,SAVEINPUTTR,SAVERESPONSETR);
     TRANSACTION RECORD (TRB) TRIN, TROUT;
     PROCEDURE SAVEINPUTTR(); FORMAL;
     PROCEDURE SAVERESPONSETR(); FORMAL;
     BEGIN
          ....
          LASTSUBBASE := TRIN.TRSUBBASE;
          CASE TRIN.TRSUBBASE OF
          BEGIN
          ( 2 ) :
               ACCESSSUBBASEONE(UPDATEV,SAVEFUNCTIONFLAG,TRIN,TROUT,
                     SAVEINPUT,SAVERESPONSE);
               % invokes library DBSUBONE
          ( 3 ) :
               ACCESSSUBBASETWO(UPDATEV,SAVEFUNCTIONFLAG,TRIN,TROUT,
                     SAVEINPUT,SAVERESPONSE);
               % invokes library DBSUBTWO
               ...
          END OF CASES;
          ...
     END;

     PROCEDURE ACCESSDATABASE(FUNCTIONFLAG,TRIN,TROUT,SAVEINPUT,
               SAVERESPONSE);
          VALUE FUNCTIONFLAG;
          REAL FUNCTIONFLAG;
          TRANSACTION RECORD (TRB) TRIN, TROUT;
          PROCEDURE SAVEINPUT(); FORMAL;
          PROCEDURE SAVERESPONSE(); FORMAL;
     % EXTERNAL ENTRYPOINT
     BEGIN

          CASE FUNCTIONFLAG OF
          BEGIN
               1:     % OPEN UPDATE
                      OPENTYPE := FUNCTIONFLAG;
               2:     % OPEN INQUIRY
                      OPENTYPE := FUNCTIONFLAG;
               3:     % UPDATE
                      UPDATE(TRIN, TROUT, SAVEINPUT, SAVERESPONSE);
               4:     % FORCEABORT
                      FORCEABORT;
               5:     % CLOSE DATABASE
                      % LET BLOCKEXIT DO IT;
          END;
     END ACCESSDATABASE;

**********************************************************************
```

```
     EXPORT
           ACCESSDATABASE;

     FREEZE(TEMPORARY);

END OF LIBRARY.
```

2.5 DMS Exception Handling

The Transaction library reprocesses transactions backed out during an abort, so the Update library does not have to be concerned with abort detection and restart. The Update library does however have to be concerned with other DM exceptions that can be returned from DMSII. The exceptions are the same as for any other DM program, except for some differences at OPEN, BEGINTRANSACTION, MIDTRANSACTION and ENDTRANSACTION. The exceptions that can be returned at these DM statements are as follows:

at OPEN :

OPENERROR
(all previous values of openerror plus two new values)
45 - this open option not permitted while TPS active. Program tried to do a regular OPEN UPDATE while another program is active with OPEN TRUPDATE.
46 - this open option not permitted until TPS recovery completed an abort or Halt/Load has occured and TPS has not yet had a chance to reprocess the backed out transactions. A regular OPEN UPDATE can not be done until a TPS program runs, causing the rerun to occur.

at BEGINTRANSACTION :

ABORT
1 - the normal abort exception that indicates a program should restart. This will never occur in a program using the TPS form of BEGINTRANSACTION.
2 - an abort has happened and TPS has not yet performed the rerun. This transaction should just EXIT the update library. The Transaction library will take care of resubmitting this transaction after it does a rerun.

DEADLOCK
1,2 - the update library should branch back and relock its records if this occurs.

AUDITERROR
1 - tried to do BEGINTRANSACTION and already in transaction state.
4 - tried to do normal BEGINTRANSACTION in program that is open TRUPDATE. Should use new form of BEGINTRANSACTION passing a transaction record.
5 - used new form of BEGINTRANSACTION passing transaction record, but data base is not open TRUPDATE.
6 - unexpected state of the transaction record. Indicates system software problem.

DATAERROR
1 - some required items are null.

at MIDTRANSACTION:

AUDITERROR
3 - MIDTRANSACTION and not in transaction state.
6 - unexpected state of transaction record.
7 - data base not open TRUPDATE.

at ENDTRANSACTION :

ABORT
1 - cant get normal abort exception in program using TPS form of ENDTRANSACTION.

DEADLOCK
1,2 - the Update library should branch back and relock its records if this occurs.

AUDITERROR
2 - tried to do ENDTRANSACTION and not in transaction state
4 - tried to do normal ENDTRANSACTION in program that is open TRUPDATE. Should use new form of ENDTRANSACTION passing a transaction record.
5 - used new form of ENDTRANSACTION passing a transaction record, but data base not open TRUPDATE.
6 - unexpected state of transaction record. Indicates system software problem.

DATAERROR
1 - some required items are null.

If the Update library does not capture these exceptions the accessroutines will cause the application program to be DSed, after which the private Transaction library, Update library and SIB will go away.

If the Update library is in transaction state when the exception is returned, an abort will be caused. Since the application program has been DSed, no useful information can be returned about the cause of the ABORT.

The Update library should capture the exception on each DM statement. Depending on the value of the result, the Update library can either handle the situation and proceed, or just

EXIT the Update library. If the Update library exits before doing a successful BEGINTRANSACTION the Transaction library will notice and give a result to the application program (result 505 - did not save a response transaction). If a transaction fails after BEGINTRANSACTION then an appropriate result will be returned to the application program and an abort will be caused.

FAILEDAFTERBTR = 501
FAILEDAFTERSAVEINPUT = 502
FAILEDAFTERMIDTR = 503
FAILEDAFTERSAVERESPONSE = 504

An inquiry transaction that does not go into transaction state must still save a response transaction before returning to the Transaction library. This can be done with an explicit call on SAVERESPONSETR. If this is not done the Transaction library will return a result 505.

VI. REMOTE LIBRARY - HOST INTERFACE

Users who have a B6800 Shared Resources System may wish to have their data base on one system and run application programs on another system in the network. Because of the large amount of communication that normally takes place between a DM program and the accessroutines, trying to run a DM program on a remote system which interacts with accessroutines and data on a host system would be very inefficient.

The Transaction System addresses this problem by providing the Remotelib and Hostinterface programs. The Remotelib runs on the remote system with the application programs. It is analogous to the Transaction Library that runs on the host system and provides the same set of entrypoints discussed in Section IV.

The Remotelib, which is compiled on the host system using the TRDESCRIPTION file, must be copied over to the remote system, and can be copied as <trbase>/CODE/HOSTLIB. Thus an application program on the remote system thinks that it has called entrypoints in the Transaction Library, but in fact has called the Remotelib because it is running on a remote system.

When an application program calls the Remotelib for the first time, a library stack is set up which is common for all users of the Remotelib, and a private stack is set up. The private Remotelib stack then initiates the Hostinterface as a dependent process on the host system. There is one Hostinterface program per application program on a remote system.

The Remotelib and Hostinterface then communicate via port files which have been declared and opened in each program. The Remotelib tells the Hostinterface which entrypoint the application program was trying to call and passes along the parameters. The Hostinterface has declared the same set of entrypoints in the Transaction Library and simply makes the procedure call with the correct parameters.

Thus the procedure call that the application program was trying to make on the remote system is actually made by the Hostinterface on the host system. The response to the procedure call is passed back to the Remotelib and on to the application program. The net result is the entrypoint in the Transaction Library was called, which accessed the data base and the result was received in the application program.

Obviously, there is additional overhead involved in passing information back and forth through the port files, but other than a slower response, the application program should not be able to tell that it is not in direct communication with the Transaction Library and data base.

There are some additional error conditions that the application program might receive from a call on the Remotelib. The program should be written to handle the errors if it will ever be run on a remote system even though it may run on the host most of the time.

The Remotelib knows the ID of the system on which the data base is located because it is specified in TFL. If for some reason the identifier given in TFL is not a system ID known to the network, the Remotelib will return an error (100).

If the Remotelib loses communication with the Hostinterface, either because of a data communications problem or a failure of the host system, the Remotelib can not continue with the procedure that was called.

If the user specified a value for the TIMEOUT parameter of OPENTRBASE, then the Remotelib will wait for that length of time to see if the other system comes back up. If the TIMEOUT is zero or the other system does not come up, the program will get a (101) result. If the host does come up, a different result (102) will be returned to indicate that the program can try again.

The application program will have to handle these errors and resubmit the last input if necessary just as though it were restarting from a Halt/Load on its own system. The Returnrestartinfo and Returnlastresponse procedures should be used to determine where to restart.

The application program that receives the 101 result must first determine whether the host system is running before it can restart. It could periodically call an entrypoint in the Remotelib and receive (101) results until the host comes up, or simply terminate and let someone restart the program when the host system is back up.

Another error condition that may occur is (103) which means that the response received from the Hostinterface is not the correct response. This indicates a problem in the port and signal mechanism.

If for some reason the remote system goes down, the Hostinterface will terminate. Upon restarting, the Remotelib will initiate another Hostinterface if there is not an active one.

When the user of the Remotelib is done, the Remotelib informs the Hostinterface and both programs terminate normally.

VII. TRANSACTION SYSTEM RECOVERY

1. General

A major objective of the Transaction System is to guarantee that any transaction which has been processed successfully is permanent, provided that certain assumptions and requirements are observed. That is, the user never need be concerned with reprocessing transactions whose changes have been backed out by a data base recovery operation (Halt/Load or Abort Recovery).

It is neither desirable nor possible to prevent the Data Base System from backing out certain transactions during recovery. But those transactions can be automatically reprocessed by the Transaction System. The previous results must be reproduced during this process, since it must appear to the user that they have never been backed out. As a consequence, the innocent victim of another user's abort is unaware that the abort has occurred; after a Halt/Load, the user only has to be concerned with restarting his job and perhaps positioning his files. Note that if the user's program is running on a remote system when the host system suffers a Halt/Load, he may be unaware that the Halt/Load has occurred.

The Transaction System is concerned with the following basic recovery operations:

Data Base Abort Recovery
performed by the Data Base System.

Data Base Halt/Load Recovery
performed by the Data Base System.

Transaction Halt/Load Recovery
performed by the Transaction System. The end-of-file for the current journal is recovered to avoid losing any information written to the file. The backed-out transaction records are identified and reprocessed.

Transaction Abort Recovery
performed by the Transaction System. The backed-out transaction records are identified and reprocessed.

Transaction Journal Rollback
performed by the Transaction System. The system can recognize when a Data Base Rollback (or the equivalent using Rebuild) has occurred. The transaction history is then rolled back to remain in sync with the data base.

Disaster Recovery
performed by user-written program. If the data base and its audit files are destroyed or otherwise unusable, it is possible to reload an an off-line dump of the data base and reprocess all transaction records captured in the history file after a given time. This process is very similar to processing transaction records which have been tanked for later processing, which is a user-controlled process.

Application Program Restart
with the aid of the Transaction System, the program determines where it was interrupted and continues from that point.

2. Transaction Recovery Design Features

The Transaction System facilitates recovery through the following design features and assumptions:

1. All Data Base transactions are designed by the user to be mutually independent. That is, they must be two-phase, or the equivalent; they must do nothing which depends upon knowledge of a prior transaction, and must keep records locked intil END-TRANSACTION. If this assumption is false, previous results may not be reproduced during transaction recovery. Refer to the section on the Update Library which defines the requirements of the user's data base transactions.

2. All input transaction records are saved in a transaction history file. This guarantees that application programs need not recreate transactions already completed. Furthermore, the order in which the records are saved is precisely the order required to reprocess them in a single-threaded recovery operation which reproduces previous results. This order is achieved by saving each record while the data base transaction is between its two phases.

3. The Data Base System maintains in its control file, and writes to the audit, the history file address of the transaction record corresponding to each data base transaction. Any recovery operation which moves the data base backward or forward in time maintains this file address in the data base control file.

4. A Halt/Load flag is maintained in the control file of each transaction journal, so that the Transaction System can recognize when a Halt/Load has occurred.

5. The Transaction System can determine whether or not the transaction processor has performed the required sequence of operations, including BEGIN-TRANSACTION, MID-TRANSACTION, and END-TRANSACTION. It can also determine whether or not the transaction processor is in transaction state. These abilities permit the Transaction System to recognize when an abort has occurred, or when an abort should be forced by the System.

6. For each identifiable user, a reference to either the last successful input transaction record or the last restart transaction record, and the corresponding response transaction are saved in the journal control file. The user can retrieve these records in order to determine where to restart an interrupted program.

3. Transaction Halt/Load Recovery

When the Transaction Library is initiated, it checks the status of the Halt/Load Flag in the journal control file. This flag is always reset when the Transaction Library terminates normally; if it is still set then the previous use of the library was either interrupted (by a Halt/Load or DS; it doesn't matter which), or the last transaction caused a abort.

The end-of-file for the current journal data file must be recovered to avoid loss of information written before the interruption. Next the data base is opened by the Transaction Library, which will cause data base Halt/Load Recovery if it is necessary.

After data base recovery, the data base control file contains the history file address of the last input transaction record remaining in the data base. This address is retrieved from the data base control file, and compared with the address of the last transaction record in the history file. If they do not agree then transaction reprocessing is initiated.

In the case where the final transaction caused an abort, the abort has possibly backed good transactions out of the data base, and the Transaction Library has not yet had a chance to reprocess the transactions. Leaving the Halt/Load flag on allows the Transaction Library to do the rerun automatically the next time it becomes active. The address in the data base control file will not match the address of the last transaction in the history file, and these backed out transactions will be reprocessed.

Transaction reprocessing begins with positioning the history file to a point just beyond the address in the data base control file. Because of the order of transaction records in the history file, all the records prior to this point are in the data base, and all records after are not. Now the Transaction Library merely reads transaction records and passes them, one at a time, to the user's Update Library for reprocessing. It is important to note that the reprocessing operation must be successfully completed in order to assure reproduced results. Irrecoverable IO errors, for example, may have serious consequences.

During this recovery operation, users attempting to process new transactions must wait until recovery`is completed.

4. Transaction Abort Recovery

Transaction Abort Recovery is very similar to Transaction Halt/Load Recovery; however, it is much more difficult to recognize when an abort has occurred. In order to accomplish this, new forms of BEGIN-TRANSACTION and END-TRANSACTION must be used, as well as a new statement MID-TRANSACTION, in the user's data base transaction code. These statements pass the transaction record to the Data Base System, which marks the record to indicate the successful execution of those statements. Furthermore, since these statements are executed in the same stack that the Transaction Library is running in (although the data base SIB is in another stack), the Transaction Library can determine if the stack is still in transaction state at the completion of the user's update code. If END-TRANSACTION was not executed, and if the stack is no longer in transaction state, then that transaction was aborted; if the stack is still in transaction state, then an abort needs to be forced by the Transaction Library (this is accomplished by calling another procedure in the Update Library).

Once the occurrance of an abort is determined (or an abort is forced), other users are locked out and forced to wait until abort recovery is completed. The first user who tries to process a transaction after the abort has finished will be selected to initiate the rerun of backed out transactions, much like Halt/Load Recovery. After the abort has finished, all users can proceed with their transactions.

5. Transaction Journal Rollback

The initialization of the Transaction Library which checks for the occurrance of a Halt/Load may instead discover that the data base appears to be at a state prior to the state of the transaction journal. This situation is recognized when there has been no Halt/Load, but the history file address in the data base control file does not agree with the address of the last transaction record saved in the history file.

This mismatch could occur either because an old data base control file has been loaded which does not reflect the state of the data base, the control file was initialized, losing the values of the last good transaction, or the data base has been rolled back or rebuilt.

In the event that the user has done a rebuild or rollback of the data base, he should then run TRUTILITY to recover the journal control file to the same state as the data base and adjust the datafile end-of-file. This should be done prior to letting any programs access the transaction base, so that the transaction base and data base do not appear to be out of sync.

If the Transaction Library does encounter a mismatch, it will notify the user and ask for instructions to either go on, overlooking the inconsistent states, or DS the program allowing the user to run TRUTILITY to recover the control file before proceeding.

6. Application Program Restart

The only aspect of recovery that an application program need consider is how to restart itself after an interruption (Halt/Load or DS). Since the Transaction System saves a reference to the restart record of the last successful input transaction, and its response, for each user, all the program has to do is ask for these transaction records. This is accomplished by calling two of the Transaction Library procedures, RETURNRESTARTINFO and RETURNLASTRESPONSE, passing the user's id number. It must be assumed that these records have been designed to provide whatever information is required to restart. Because of other aspects of transaction recovery already discussed, the user is assured that the transaction record purported to be his last is in fact just that.

VIII. DMS II HOST SYNTAX FOR TRANSACTION PROCESSING

1. General

It is important to be able to syncronize recovery of the Transaction System with recovery of the Data Base System. To accomplish this, certain extensions to the present application language interface to DMS II have been defined.

These extensions include (1) a new statement, MID-TRANSACTION; (2) optional extensions to the present BEGIN-TRANSACTION and END-TRANSACTION statements; and (3) a new option for the data base OPEN statement (TRUPDATE). Present syntax will continue to support DMS II applications which do not use the Transaction System. However, user code in the Update Library must use the new syntax.

2. Statement Syntax

All three statements must be able to pass a transaction record, in addition to present parameters, to procedures in the accessroutines. For the new forms only, NO-AUDIT is the default for both BEGIN-TRANSACTION and END-TRANSACTION; in fact, it is not permitted to specify AUDIT or NO-AUDIT with these forms.

COBOL

```
    -- BEGIN-TRANSACTION ------------------------------------------->
                         |                                     |
                         |- ( --<transaction record variable>-- ) -|
                         |- AUDIT -----------------------------|
                         |- NO-AUDIT --------------------------|

    >-<restart data set name>--<exception>----------------------------|


    -- MID-TRANSACTION -- ( --<mid-transaction parameters>-- ) -------->

    >-<restart data set name>--<exception>----------------------------|


    -- END-TRANSACTION ----------------------------------------------->
                       |                                       |
                       |- ( --<end-transaction parameters>-- ) -|
                       |- AUDIT -------------------------------|
                       |- NO-AUDIT ----------------------------|

    >-<restart data set name>------------<exception>------------------|
                                |- SYNC -|


<mid-transaction parameters>

    --<transaction record variable>-- , ----------------------------->

    >-<saveinput procedure identifier>--------------------------------|


<end-transaction parameters>

    --<transaction record variable>-- , ----------------------------->

    >-<saveoutput procedure identifier>-------------------------------|
```

ALGOL

```
-- BEGINTRANSACTION ---------------------------------------------->
                         |                                    |
                         |- ( --<transaction record variable>-- ) -|
                         |- AUDIT -----------------------------|
                         |- NOAUDIT ---------------------------|

>-<restart data set name>--<exception>-----------------------------|


-- MIDTRANSACTION -- ( --<mid-transaction parameters>-- ) --------->

>-<restart data set name>--<exception>-----------------------------|


-- ENDTRANSACTION ------------------------------------------------->
                       |                                    |
                       |- ( --<end-transaction parameters>-- ) -|
                       |- AUDIT -----------------------------|
                       |- NOAUDIT ---------------------------|

>-<restart data set name>-------------<exception>------------------|
                           |- SYNC -|
```

For both languages, <transaction record variable> is the name of the formal input transaction record variable; <saveinput procedure identifier> is the name of the SAVEINPUT formal procedure; <saveoutput procedure identifier> is the name of the SAVEOUTPUT formal procedure. The procedures are untyped formal procedures with no parameters. In each case, the compiler will generate a call on the given procedure immediately prior to the call on the DMS procedure in the accessroutines.

3. OPEN Option

The new OPEN option, TRUPDATE, must be used in the Update Library instead of the UPDATE option. Only when this option is used are the statements described above permitted. Note that it is still permitted to use the INQUIRY option; in this case, no forms of BEGIN-TRANSACTION, etc., are allowed.

IX. TRANSACTION SYSTEM UTILITY PROGRAM

A Utility program serving several functions is available with the Transaction System. Currently there are four major functions allowing the user to initialize transaction journals , search through transaction data files, print information in the control file of a journal, and recover a control file that has been lost or damaged. The syntax for the Utility program is driven by the first word which presently can be INITIALIZE, SEARCH, RECOVERCF, or CONTROLINFO.

The program is tailored for a particular transaction base (it is compiled using the trdescription file). It also interprets the trdescription file at run time, if necessary, to provide formatted output for the specified transaction records.

1. Journal Initialization

When a user wants to create a new journal he must first run the Utility program to initialize one with the new name. UTILITY will check to see if any special attributes were specified for the journal in the TFL description. If there are no predefined attributes then UTILITY will use the default attributes specified in TFL. If there are no defaults given then standard defaults will be used.

UTILITY will create the file <trbase>/<journalid>/CONTROL. The data files will be created as needed by the Transaction Library. The user will then be able to refer to this journal name in an application program as the parameter to the Transaction Library.

The syntax for the INITIALIZE option is given below:

```
-- INITIALIZE <journalid> ------------------------------------------------|
```

2. Search Data Files

A major function of the Utility program is to search a specified transaction data file for certain transaction records and print them out. This is similar in function to PRINTAUDIT but has the capability to print formatted transaction records along the lines of INQUIRY.

The syntax for the SEARCH option is as follows :

```
                                  |<--------------- , --------------|
                                  |                                 |
-- SEARCH <journalid> -- , -----/1\- RANGE <range options> ------------|
                                  |-/1\- USERS <user option> ---|
                                  |-/1\- SELECT <tr options> ---|
```

```
<range options>

      |<------------------------- , -------------------------|
      |                                                      |
-- ( -----/1\- TIME =  --- BEGIN -------- TO --- END ------------ ) --|
        |                |               |     |              |
        |                |- <datetime> -|     |- <datetime> -|
        |                                                     |
        |-/1\- FILE =  --- <filenum> -------------------------|
                         |                                    |
                         |- <filenum> TO <filenum> -----------|
```

<datetime>

```
-- <month> -- <day> -------------- AT <hour>:<minute> ----------------->
                    |           |
                    |- <year> -|

>---------------------------------------------------------------------|
  |
  |- : <second> -----------------|
                |                 |
                |- .<fraction> -|
```

<user option>

```
-- ( --- ALL --------- ) --------------------------------------------|
      |                |
      | |<--- , --|    |
      | |         |    |
      |--- <int> ---|
```

<tr options>

```
-- ( --- ALL : <all options>  ---------------------- ) ---------------|
      |                                              |
      | |<----------------- ; ------------------| |
      | |                                        | |
      |--- <format selector> : <format options> ---|
```

<all options>

```
  |<------------ , -----------|
  |                           |
------/1\- PRINT ---------------------------------------------------|
    |                  |          |
    |                  |- HEX  -|
    |                            |
    |-/1\- COUNT -----------|
```

<format selector>

```
-- <format id> ------------------------------------------------------|
                |            | |                       |
                |- .<subfmt> -| |- WHERE <item cond> -|
```

<format options>

```
  |<---------------------- , ----------------------|
  |                                                |
------/1\- PRINT --------------------------------------------------|
    |                  |                                |
    |                  |- HEX -------------------------|
    |                  |                            |  |
    |                  |          |<--- , ---|      |  |
    |                  |          |          |      |  |
    |                  |- ITEMS ( --- <item> --- ) -|  |
    |                                                  |
    |-/1\- COUNT ----------------------------------|
```

<item cond>

```
-- <item name> ---  <  ------ <alpha value> ----------------------------|
                 |         | |                       |
                 |-  <=  -| |- <numeric value> -|
                 |          |
                 |-  =  --|
                 |          |
                 |-  ¬=  -|
                 |          |
                 |-  >=  -|
                 |          |
                 |-  >  --|
```

Three verbs follow the SEARCH command and journalid which must be specified by the user. RANGE gives a time range between which UTILITY will search for transaction records; USERS gives the userids that are to be searched for; SELECT gives the transaction record types to be searched for and the appropriate action to take when one is located.

Upon selecting a transaction record which satisfies the given selection criteria, the user can specify that it either be PRINTed or COUNTed or both. The printing of the records is the most useful function, and can optionally be done either formatted or in hex. The user can limit the printing to certain items within the record, and can even base the selection of a transaction record on the value of a particular numeric or alpha item.

3. Control File Recovery

```
-- RECOVERCF ------------------------------------------------------>

>--- TRHISTORY -----------------------------------------------------|
  |                                                    |
  |- <journalid> -- TO -- <stop point> -- FILE -- <filenum> -|

<stop point>

    ---- END OF ----------------------------------------------------|
      |                                         |
      |- BLOCK --<blocknum>-- OFFSET <int> -- OF -|
      |                                         |
      |-<datetime>-- IN --------------------------|
```

A function to allow the TRHISTORY file to be recovered in case the control file is corrupted or the transaction system is out of sync with the data base is provided by this command.

If the Transaction library notices that it is out of sync with the data base upon being invoked, it will refuse to run. This could be caused by a rollback of the data base. The user can run UTILITY to do a RECOVERCF TRHISTORY which will effectively ROLLBACK the TRHISTORY file to a point where it is in sync with the data base.

Another way that this function would be used is if the TRHISTORY control file had been lost or destroyed. The user will need to take some action to recover a control file to work with, preferrably a copy of a recent one that the user dumped. If no copy is available, the user could resort to doing an initialize to create the control file. The RECOVERCF TRHISTORY function will then get the control file in sync with the data base, using the TRHISTORY file which reflects changes to the control file up to the current point in time.

A means to recover the control file for a tank file is also provided, but with slightly different syntax. Since tank files are not considered to be in sync with any part of the transaction system, the user must provide a point at which the control file is to be recovered to.

The user can specify this point by giving either a datetime or a block and offset within a file, or specifying that recovery should continue to the end of the particular file. UTILITY will use the data files to make updates to the control file so that it will reflect the state of the tank file at the stopping point specified.

By specifying an earlier time, the user is able to roll back a tank file to a point in time. This allows the user to delete a series of transactions that he doesn't want to process later on. This function is not available for the TRHISTORY file.

4. Print Control File Information

```
-- CONTROLINFO --<journalid>-- , -------------------------------------|
                                  |           |
                                  |- USERS -|
```

Information in the control file of a journal can be printed with this function. The table of contents or first block of the control file is printed by default. The USERS syntax causes the information about each known user for the journal to be printed, including the last response transaction and the location of the last restart transaction.

5. Starting a UTILITY Run

When initiating a UTILITY run, the parameters discussed above are passed as an array. Since UTILITY interprets the description file at run time, the file TFL must be label equated at run time.
A example of a UTILITY job follows:

```
RUN TRTEST/CODE/UTILITY("INITIALIZE TRHISTORY");
FILE TFL = TRTEST/TRDESCRIPTION;
```

X. TRANSACTION SYSTEM SECURITY

1. Guardfile Security

The only security provided by the Transaction system is via guardfiles. A DBA will be able to restrict the use of transaction record types by program name and/or usercode.

A guardfile can be associated with a transaction subbase in TFL, just as with logical data base in DASDL. The DBA can use this to restrict access to transaction types which have logically been grouped together. To a certain degree, this ability overlaps with the guardfile security provided in DASDL, but it can be used independently or in conjunction with DM security. A user might wish to restrict access to certain transaction records independent of whether he has logical data bases in the DASDL.

The Transaction Library is responsible for checking the transaction records against the guardfiles as they come through to be processed. Rather than checking EVERY transaction record from a user, the library will check the guardfile for each subbase the first time it is used, and remember whether the user was "OK". The subbase number is placed in the control portion of each transaction record when it is initialized. A record without a subbase number was not properly initialized and will be rejected.

XI. CONVERTING TO THE TRANSACTION PROCESSING SYSTEM

Converting an existing data base with all its application programs to use TPS may prove to be impractical for some users. The only practical approach, where there are a large number of application programs, may be to convert some of the programs to TPS and continue to run the others as conventional DMS programs. To facilitate this step by step conversion, a variation of the Transaction System has been developed that allows both TPS and conventional DM programs to update a data base at the same time.

There are restrictions to running in a mixed TPS mode which make it desirable only as a temporary means of converting an existing DM data base to use TPS, and not as a permanent way of operating. In this mode the problems of reproducibility are greatly magnified and the burden of insuring reproducibility falls on the user.

When TPS is the only updater of the data base and the Update library has been programmed in a two-phase manner, then TPS guarantees reproducible results. But when other updaters are running, even if they are programmed to be independent, TPS can no longer guarantee reproducibility unless extra restrictions are imposed. If any TPS transaction requires the same data base records as a non TPS program running at the same time, then the results produced by the TPS program may not be reproduced if the transactions have to be resubmitted. This is because TPS will resubmit only the transactions it knows about, and the sequence of events may not be reproduced.

Reproducibility may be achieved even when TPS and conventional updaters run if suitable restrictions are imposed. The simplest restriction is to divide up the application programs into groups that access disjoint portions of the data base. If this can be done then one set of programs may be run through TPS and the other as conventional programs without affecting reproducibility of the TPS programs, since the order in which the non TPS transactions are reprocessed has no effect on the TPS transactions.

Another alternative is to divide the set of applications into two groups which have the least amount of overlap, and take additional steps to ensure that there is reproducibility. The problem now centers around the application programs that access common portions of the data base. Either programmatic or operational guidelines must be followed to ensure reproducibility. An operational strategy might be to only run the programs that would destroy independence when TPS is not running.

If this is not feasible, programmatic conventions can be followed to ensure that if a TPS program is referencing a certain portion of the data base, then a DM program can not. A global data set with a single record could be used as a lock to accomplish this. In addition to preventing an outside program from using this portion of the data base while the TPS program is active, it must also ensure that a sync point is caused when it is finished so that a subsequent program cannot cause its transactions to be backed out.

Any other programmatic conventions that accomplish the same degree of independence will also work. If a user does not care about results being reproduceable when transactions are reprocessed by TPS, then such restrictions are not necessary. Each user must analyze the degree of reproducibility they require and design their system accordingly.

TPS and non TPS update is allowed by specifying a global data base option in DASDL. The new option is TPSDUALUPDATE and the default value is FALSE. Its value can be changed through a DASDL update. During the conversion process, the option should be set to TRUE; when all application programs have been converted, it should be reset to FALSE.

IMPLEMENTATION

OF

INTER-PROCESS COMMUNICATION

INTER-PROCESS COMMUNICATION

Tasks executing on various hosts (or the same host) may communicate with each other by means of the Inter-Process Communication (IPC) facilities provided by Host Services. There are two forms of user interface to the IPC system: the full IPC interface and the logical input/output interface.

The full IPC system facilities are currently defined and available only in the B6000/B7000 series DCALGOL language. The elements of the full interface are called ports and signals. Ports and signals are very general mechanisms for performing synchronous and asynchronous communication.

The logical input/output interface is available in B6000/B7000 series ALGOL, FORTRAN, COBOL, COBOL74, and PL/I. The elements of this interface are simple files with a new KIND attribute value of PORT. Ordinary input/output operations may be used to perform IPC using these port files.

The port provides the end-point functions for a set of connection paths between two or more processes. It performs segmentation and reassembly of messages as required by the connection path(s); it provides reliable delivery of text between the processes; it delivers text in the order in which it was sent by the processes; it provides a mechanism for associating requests and responses between processes.

A port is described by a set of port attributes and a set of functions. If two processes wish to communicate, each one opens a port with attribute values which are compatible with those that the other process has specified for its port, and a path between them is established.

```
|--------------|                                |--------------|
|              |                                |              |
| Host   |-----|                                |-----|   Host |
|        | Port---------------------------------Port  |        |
|  A     |-----|        Connection Path         |-----|    B   |
|              |                                |              |
|--------------|                                |--------------|
```

Connections between processes which desire to communicate are established based on the attributes of the port for each process. The port attributes are exchanged and the port connection is established via a series of messages transmitted between the hosts. All port connections, whether for Host Services processes or for user processes, are established in the same manner.

1.0 DCALGOL LANGUAGE SYNTAX AND SEMANTICS

1.1 DECLARATIONS

Syntax:

<port declaration>

```
          |<------------------ , ------------------|
          |                                        |
-- PORT ---<port identifier>--<port attribute list>--------------------|
```

<port identifier>

```
--<identifier>--|
```

<port attribute list>

```
-------------------------------------------------------------|
  |                                                         |
  |     |<------------------- , -------------------|        |
  |     |                                          |        |
  |- ( ---<port attribute>------------------------------ ) -|
                              |                    |
                              |- = <attribute value> -|
```

<signal declaration>

```
            |<-------------------- , --------------------|
            |                                            |
-- SIGNAL ---<signal identifier>--<signal attribute list>----|
<signal identifier>

--<identifier>--|


<signal attribute list>

-------------------------------------------------------------|
   |                                                      |
   |      |<-------------------- , --------------------|  |
   |      |                                            |  |
   |- ( ---<signal attribute>----------------------------- ) -|
                                    |                  |
                                    |- = <attribute value> -|


<port array declaration>

-- PORT ARRAY ----------------------------------------------------->

   |<---------------------------- , ----------------------------|
   |                                                            |
>---<port array id>-- [<bound pair list>] --<port attribute list>------|


<port array id>

--<identifier>--|


<signal array declaration>

-- SIGNAL ARRAY --------------------------------------------------->

   |<-------------------------- , ----------------------------|
   |                                                          |
>---<signal array id>-- [<bound pair list>] --<signal attr list>-------|


<signal array id>

--<identifier>--|
```

Examples:

```
     PORT PORT1(MYNAME="TASKNAME."),
          PORT2;
     SIGNAL SIG(TIMEOUT=3);
     PORT ARRAY PARY1 [1:10] (SECURITYTYPE=PRIVATE);
     SIGNAL ARRAY SIGARY1 [0:2,0:4] (RESPONSEREQUIRED=FALSE),
                  SIGARY2 [0:9];
```

Semantics:

The <port declaration> declares an <identifier> to be a <port identifier> and specifies values for the port attributes associated with the declared port.

The <signal declaration> declares an <identifier> to be a <signal identifier> and specifies values for the signal attributes associated with the declared signal.

The <port array declaration> declares an <identifier> to be a <port array identifier>, specifies the number of dimensions and the bounds for each dimension of the port array, and specifies values for the port attributes associated with each element of the port array. A port array can have up to 15 dimensions.

The <signal array declaration> declares an <identifier> to be a <signal array identifier>, specifies the number of dimensions and the bounds for each dimension of the signal array, and specifies values for the signal attributes associated with each element of the signal array. A signal array can have up to 15 dimensions.

In the <port attribute list> and in the <signal attribute list>, the <attribute value>s must be constant.

1.2 PORT ATTRIBUTES

The following port attributes are available in the full interface to ports and signals. Pointer-valued attribute values must end with a period. Pointer-valued attribute values may be null ("."), meaning not specified.

DSONERROR

Data Type: Boolean
Get: anytime
Set: anytime

The DSONERROR attribute indicates whether or not the task using a PORT should be DSed (caused to cease execution) when a run-time error on a SIGNAL or PORT operation or attribute occurs. The default value of the attribute is implementation specific, but should normally be TRUE. If DSONERROR is set FALSE, the task will not be DSed; instead, the SIGNALERROR attribute of the SIGNAL being referenced will be set to indicate the nature of the error.

INTNAME

Data Type: pointer
Get: anytime
Set: when PORT is CLOSED

The INTNAME attribute of a PORT is the name by which it is declared and referenced by the task which is using it. It has the same form as the file attribute INTNAME.

MESSAGEEVENT

Data Type: event
Get: anytime
Set: RESET/anytime, no other action allowed

The MESSAGEEVENT attribute for a PORT is caused whenever one or more MESSAGEs have arrived at the PORT, and is reset when all MESSAGEs have been received. In particular, the MESSAGEEVENT is caused when the MESSAGE being queued was originated by a remote SENDMESSAGE operation. It is not caused by either a RESPONSE originated by a SENDRESPONSE operation or by a MESSAGE which is a continuation of a previous MESSAGE; these are placed directly in a target SIGNAL and affect the values of attributes of that SIGNAL.

MESSAGEQUEUELIMIT

Data Type: integer
Get: anytime
Set: anytime

The MESSAGEQUEUELIMIT attribute indicates how many MESSAGEs may be queued in the PORT. If the MESSAGEQUEUELIMIT is set to a value smaller than the current size of the queue, no additional inserts into the queue are permitted until the size of the queue falls below this MESSAGEQUEUELIMIT. Setting the value to 0 causes it to revert to the system default value.

If the number of MESSAGEs queued for a particular PORT equals or exceeds the MESSAGEQUEUELIMIT value, then incoming MESSAGEs are rejected.

MESSAGEQUEUESIZE

Data Type: integer
Get: anytime
Set: get only

The MESSAGEQUEUESIZE attribute of a PORT gives the number of MESSAGEs which are currently queued in the PORT (they have not yet been received by the task).

MYHOSTNAME

Data Type: pointer
Get: anytime
Set: get only

The MYHOSTNAME attribute contains the name of the host at which the task using the PORT is executing. It allows tasks to be independent of the host at which they execute, since interrogating this attribute provides this information. The range of values for this attribute is the set of logical host names.

MYNAME

Data Type: pointer
Get: anytime
Set: when PORT is CLOSED

The MYNAME attribute is the name by which this task wishes to be known when opening the PORT. Its default value is the TASKNAME of the task opening the PORT. It may take on the same range of values as a FILENAME attribute. The value may also be set to ".", meaning not specified.

PORTATTERROR

Data Type: Boolean
Get: read
Set: get only

The PORTATTERROR attribute has a value of TRUE if the last attribute action was in error.

PORTERROR

Data Type: integer
Get: anytime
Set: get only

The PORTERROR attribute indicates various error conditions of the PORT. Some of the possible values are:

NOERROR

There are no errors.

DISCONNECTED

The PORT has been disconnected from its correspondent.

NOHOST

The attribute "YOURHOSTNAME" refers to an unknown host.

HOSTNOTREACHABLE

The attribute "YOURHOSTNAME" refers to a host that cannot be contacted at the present time.

PORTNAME

Data Type: pointer
Get: anytime
Set: when PORT is CLOSED

The PORTNAME attribute of a PORT is the external name by which PORTs that are communication with each other are associated. The form of PORTNAME is that of a FILENAME. Its default value is the same as the INTNAME of the PORT. The value of PORTNAME may be changed at run time; however, the value of PORTSTATE must be CLOSED when this is done.

PORTSTATE

Data Type: integer
Get: anytime
Set: get only

The values of the PORTSTATE attribute are:

CLOSED

Either the task has not yet attempted to open the PORT or the task has completed a close operation on it. No MESSAGEs are queued for the PORT; SIGNALS which had used the PORT may or may not have a value of COMPLETED for their SIGNALSTATE. Any attempt to use the PORT in a SIGNAL operation will result in an error action.

OFFERED

The task has attempted to open the PORT but the opening of the PORT is not yet complete; the offer to open remains in the system until it is completed or withdrawn by the task. Any attempt to use the PORT in a SIGNAL operation will result in an error action.

OPENED

The PORT may be used for SIGNAL operations.

DEACTIVATED

The PORT has been closed by the other task engaged in the conversation or by a disconnection between the hosts where the tasks in communication reside. The MESSAGEs from that task which are queued for this PORT remain available to be dequeued and processed. However, any attempts to perform a SIGNAL operation other than RECEIVEMESSAGE using this PORT will result in an error action. When all queued MESSAGEs have been removed, the PORTSTATE of the PORT becomes CLOSED.

SUSPENDED

Communication between the local host and the remote host has been temporarily interrupted, and may be resumed or discontinued depending on the actions of the remote host.

PORTUSE

Data Type: integer
Get: anytime
Set: when PORT is CLOSED

The attribute PORTUSE specifies how the PORT may be used. Some of the possible values are:

IN

May be used to do RECEIVEMESSAGE and SENDRESPONSE.

OUT

May be used to do SENDMESSAGE and RECEIVERESPONSE.

IO

May be used to do all operations.

STATEEVENT

Data Type: event
Get: anytime
Set: RESET/anytime, no other action allowed

The STATEEVENT attribute for a PORT is caused whenever the value of its PORTSTATE changes.

YOURHOSTNAME

Data Type: pointer
Get: anytime
Set: when PORT is CLOSED

The YOURHOSTNAME attribute specifies the host in the network which contains the task with which this task wishes to communicate using this PORT. Its default value is the value of the MYHOSTNAME attribute.

YOURNAME

Data Type: pointer
Get: anytime
Set: when PORT is CLOSED

The YOURNAME attribute of a PORT in use by a task corresponds to and is the same value as the MYNAME of the task with which it is communicating. Its default value is "not specified" (".").

1.3 SIGNAL ATTRIBUTES

The following signal attributes are available in the full interface to ports and signals.

CONFIRMATION

Data Type: Boolean
Get: anytime
Set: when SIGNAL is COMPLETED

If the value of the CONFIRMATION attribute of a SIGNAL is TRUE, the user will be informed (via the SIGNALSTATE attribute) of Network level acknowledgements to MESSAGEs or RESPONSES sent. If the value of the CONFIRMATION attribute is FALSE, the user will not be informed of these acknowledgements.

MAXMESSAGETEXTSIZE

Data Type: integer
Get: anytime
Set: when PORT is CLOSED

The value of the MAXMESSAGETEXTSIZE attribute of a SIGNAL is the limit on the size of the MESSAGETEXT of the SIGNAL in characters. Its default value is the maximum size permitted by the host system implementation. The user may change the value of this attribute, except that the value may not exceed the default value.

MAXRESPONSETEXTSIZE

Data Type: integer
Get: anytime
Set: when PORT is CLOSED

The value of the MAXRESPONSETEXTSIZE attribute of a SIGNAL is the limit on the size of the RESPONSETEXT of the SIGNAL in characters. Its default value is the maximum size permitted by the host system implementation. The value of this attribute may be changed by the user to any other value smaller than the default value.

MESSAGETEXT

Data Type: string
Get: anytime
Set: when SIGNAL is COMPLETED

The MESSAGETEXT attribute of a SIGNAL is the text of the MESSAGE carried by the SIGNAL. Its value is used in SENDMESSAGE operations and is changed by RECEIVEMESSAGE operations. Its value is undefined if the MESSAGEEVENT is not in the HAPPENED state.

MESSAGETEXTSIZE

Data Type: integer
Get: anytime
Set: get only

The MESSAGETEXTSIZE attribute indicates the current size of the MESSAGETEXT of a SIGNAL in characters. Its value is defined when MESSAGETEXT is defined.

RESPONSEREQUIRED

Data Type: Boolean
Get: anytime
Set: when SIGNAL is COMPLETED

The RESPONSEREQUIRED attribute of a SIGNAL indicates that a response must be sent for this message.

RESPONSETEXT

Data Type: string
Get: anytime
Set: anytime

The RESPONSETEXT attribute of a SIGNAL is used much like the MESSAGETEXT attribute, except that it is used to hold the RESPONSE to the MESSAGE in MESSAGETEXT. As such, its value is used in the SENDRESPONSE operation; it is changed by a RECEIVERESPONSE operation. While it is accessible at any time, its value is undefined when the SIGNALSTATE is AWAITINGRESPONSE and the TEXTEVENT is not in the HAPPENED state.

RESPONSETEXTSIZE

Data Type: integer
Get: anytime
Set: get only

The RESPONSETEXTSIZE attribute indicates the size of the RESPONSETEXT in characters. Like MESSAGETEXTSIZE, this attribute is valid when the RESPONSETEXT is valid.

SIGNALERROR

Data Type: integer
Get: anytime
Set: get only

The value of the SIGNALERROR attribute of a signal indicates whether an error is currently associated with this SIGNAL, and if so, what kind of error. If the DSONERROR attribute of the PORT is TRUE, then the occurrence of one of these errors will cause the task performing the operation to be DSed. The SIGNALERROR attribute is reset to NOERROR when the SIGNAL is used in a SIGNAL operation.

Some of the possible values for SIGNALERROR are:

NOERROR

There is no error currently associated with the SIGNAL.

NETWORKTIMEOUT

The network control system encountered a timeout when it tried to transmit the MESSAGE or its RESPONSE.

MESSAGEREJECTED

A SENDMESSAGE has been performed and the MESSAGE sent has been rejected (for example, due to the MESSAGEQUEUELIMIT for the receiver's PORT being exceeded).

RESPONSEREJECTED

A SENDRESPONSE has been performed and the RESPONSE sent was rejected (for example, it could not be associated with a SIGNAL). It is possible to receive a SIGNAL claiming to be a RESPONSE to a MESSAGE for which no SIGNAL can be found because a SIGNAL may be CLEARed while at the same time the RESPONSE is in transit back to the originating host. If no SIGNAL can be found for the RESPONSE or if it is found and its SIGNALSTATE is not AWAITINGRESPONSE, the text of the RESPONSE will be discarded at the host where it is received, and the SIGNALERROR attribute will be set to RESPONSEREJECTED where the SENDRESPONSE was performed. If the SIGNAL used in the SENDRESPONSE has been reused by the time the rejection of the RESPONSE has been detected, then no error is reported.

CLEARED

A CLEAR operation has been performed on this SIGNAL, or the program waiting on some sort of signal operation was DSed by the operator.

EMPTYSIGNAL

A RECEIVEMESSAGE operation has been performed and there is no MESSAGE available at the PORT.

TIMEDOUT

The SIGNALSTATE of the SIGNAL has remained AWAITINGRESPONSE more than the number of seconds specified by the TIMEOUT attribute. The SIGNAL has been CLEARed, its SIGNALSTATE set to COMPLETED, and SIGNALERROR set to TIMEDOUT.

PROTOCOL

An operation was performed that is inconsistent with the current state of the SIGNAL; e.g., a SENDMESSAGE while the state of the SIGNAL is AWAITINGRESPONSE.

CHANNELERROR

The message could not be sent because the message transportation vehicle mechanism failed.

NOTTRANSMITTED

An attempt was made to send a SIGNAL through a PORT whose PORTSTATE is SUSPENDED.

SIGNALSTATE

Data Type: integer
Get: anytime
Set: get only

Possible values for the SIGNALSTATE attribute are:

COMPLETED

No segments of the MESSAGE or RESPONSE based on this SIGNAL remain to be transmitted and no such segments await confirmation of their receipt.

TRANSMITTING

The PORT is in the process of transmitting a MESSAGE or RESPONSE based on this SIGNAL. If the MESSAGE requires segmenting, then some segments remain to be transmitted.

AWAITINGACK

The PORT has finished transmitting all segments of the MESSAGE or RESPONSE based on this SIGNAL but has not received confirmation of their receipt.

AWAITINGRESPONSE

The local PORT has sent the MESSAGE associated with this SIGNAL whose RESPONSEREQUIRED attribute is TRUE and the PORT is ready for the remote user to send a RESPONSE to it.

RESPONSENEEDED

The PORT has received a MESSAGE from the remote user, delivered it to the user, and a RESPONSE must now be sent to the remote user.

STATEEVENT

Data Type: event
Get: anytime
Set: RESET/anytime, no other action allowed

The STATEEVENT attribute for a SIGNAL is caused whenever the SIGNALSTATE attribute becomes COMPLETED. It is reset when a SIGNAL operation is performed using the SIGNAL.

TEXTEVENT

Data Type: event
Get: anytime
Set: RESET/anytime, no other action allowed

The TEXTEVENT attribute for a SIGNAL is caused when a RESPONSE to a MESSAGE is received. Additionally, this event will be caused when the SIGNAL times out (see the TIMEOUT attribute). This event is reset when the last RESPONSE is received.

TIMEOUT

Data Type: integer
Get: anytime
Set: when SIGNAL is COMPLETED

The TIMEOUT attribute of a SIGNAL indicates the number of seconds which may elapse between the time the SIGNALSTATE of the SIGNAL changes from AWAITINGRESPONSE to COMPLETED. The occurrence of a timeout will cause the TEXTEVENT and the STATEEVENT for the SIGNAL and set the SIGNALERROR attribute to TIMEDOUT.

A value of 0, which is the default value for this attribute, indicates that the timeout function is not desired.

1.4 ATTRIBUTE ASSIGNMENT STATEMENTS

Syntax:

```
<port attribute assignment>

--<port designator>-- . <port attribute> -- := --<expression>--|


<port designator>

----<port identifier>--------------------------------|
  |                                                  |
  |-<port array identifier>-- [<subscript list>] -|


<signal attribute assignment>

--<signal designator>-- . <signal attribute> -- := --<expression>--|


<signal designator>

----<signal identifier>--------------------------------|
  |                                                    |
  |-<signal array identifier>-- [<subscript list>] -|
```

Examples:

```
     STRING STRG;
     PORT1.SECURITYTYPE:=VALUE(PRIVATE);
     SIG.MAXMESSAGETEXTSIZE:=1000;
     SIGARY1[1,4].MESSAGETEXT:=STRG;
```

Semantics:

Port and signal attributes may be assigned values at runtime through attribute assignment statements. The type of the attribute and the type of the expression must match; for example, string attributes must be assigned string expressions. The <port attribute> and the <signal attribute> cannot be pointer-valued attributes.

1.5 ATTRIBUTES IN EXPRESSIONS

Syntax and Semantics:

<port designator> . <port attribute> and <signal designator> . <signal attribute> can be used in arithmetic, boolean and pointer expressions where <file identifier> . <file attribute> would be allowed and where the type of the attribute is appropriate. <port designator> . <string-valued port attribute> and <signal designator> . <string-valued signal attribute> can be used as <string primary>s in <string expression>s.

Examples:

```
     INTEGER STATE;
     POINTER PTR;
     STRING STRG;
     STATE:=PORT1.PORTSTATE;
     IF PARY1[9].SECURITYTYPE = VALUE(PRIVATE) THEN;
     REPLACE PORT2.PORTNAME BY PTR;
     REPLACE PTR BY PARY1[2].PORTNAME;
     STRG:=SIG.MESSAGETEXT CAT "ABC";
     STRG:=SIGARY2[8].RESPONSETEXT;
```

1.6 EVENT HANDLING

Syntax and Semantics:

<port designator> . <event-valued port attribute> and <signal designator> . <event-valued signal attribute> can be used wherever an event can be used.

Examples:

```
     BOOLEAN B;
     REAL VAL;
     B:=AVAILABLE(PORT1.MESSAGEEVENT);
     VAL:=WAIT(PARY1[3].MESSAGEEVENT,SIG.TEXTEVENT);
     B:=AVAILABLE(SIGARY1[2,0].TEXTEVENT);
```

1.7 PARAMETER PASSING

Syntax and Semantics:

A <port designator> can be passed to a port formal parameter, and a <signal designator> can be passed to a signal formal parameter.

<port array designator>

```
----<port array identifier>-------|
   |                              |
   |-<port subarray designator>-|
```

<port subarray designator>

```
--<port array id>-- [ --<subscript part>--<subarray part>-- ] --|
<signal array designator>

----<signal array identifier>-------|
  |                                 |
  |-<signal subarray designator>-|


<signal subarray designator>

--<signal array id>-- [ --<subscript part>--<subarray part>-- ] --|
```

A <port array designator> can be passed to a port array formal parameter with the same number of dimensions as the <port array designator>, and a <signal array designator> can be passed to a signal array formal parameter with the same number of dimensions as the <signal array designator>.

Example:

```
     BEGIN
          PORT PORTID;
          PORT ARRAY PORTARY[0:9,0:4];
          PROCEDURE PROC(P); PORT P;
          BEGIN
          END;
          PROC(PORTID);
          PROC(PORTARY[5,1]);
     END.
```

1.8 PORT STATEMENTS

Syntax:

```
<open statement>

-- OPEN -- ( --<port designator>-- , --- WAIT -------- ) --|
                                        |              |
                                        |- OFFER -----|
                                        |              |
                                        |- AVAILABLE -|


<close statement>

-- CLOSE -- ( --<port designator>-- , --- CLEAR ---- ) --|
                                         |          |
                                         |- RETAIN -|
```

Examples:

```
     OPEN(PORT1,OFFER);
     IF CLOSE(PARY1[5],CLEAR) THEN;
```

Semantics:

The port statements can also be used as Boolean primaries, in which case a value of TRUE means that the operation was unsuccessful.

The OPEN statement opens the port. A list of ports waiting to be opened is maintained by the system. This list is searched at open time for a matching port. If a match is found, both ports are put into the OPENED state. The matching algorithm is the following:

A. The MYNAME of one port must match the YOURNAME of the other port and vice versa. Note that a null YOURNAME (".") will match any MYNAME but not vice versa.

B. The PORTNAMEs must be identical and may not be null (".").

C. The MYHOSTNAME of one port must match the YOURHOSTNAME of the other port and vice versa. Note that a null YOURHOSTNAME will match any MYHOSTNAME.

If WAIT is specified in the OPEN statement, the task is suspended until the port is completely open.

If OFFER is specified in the OPEN statement, an offer is made to the system to open the port. When a matching open request or offer is found by the system, then the open operation will be completed. Before the open operation is completed, the PORTSTATE of the port is OFFERED. The task is not suspended.

If AVAILABLE is specified in the OPEN statement, the port is opened if and only if there is an outstanding offer to open it. In either case, control is returned to the user program.

The CLOSE statement closes a port.

If CLEAR is specified in the CLOSE statement, any signals which have not yet been received by the port are cleared.

If RETAIN is specified in the CLOSE statement, any signals which have not yet been received by the port are allowed to be received before the port is closed. Responses may be sent if the port to which the response is directed is not closed. Before the close operation is completed, the PORTSTATE of the port is DEACTIVATED.

1.9 SIGNAL STATEMENTS

Syntax:

```
<sendmessage statement>

-- SENDMESSAGE -- ( --<signal designator>-- , --<port designator>-- ) -|


<sendresponse statement>

-- SENDRESPONSE -- ( --<signal designator>-- ) ------------------------|


<receivemessage statement>

-- RECEIVEMESSAGE -- ( --<signal designator>-- , --<port designator>--->

>- ) -----------------------------------------------------------------|


<receiveresponse statement>

-- RECEIVERESPONSE -- ( --<signal designator>-- ) ---------------------|


<clear statement>

-- CLEAR -- ( --<signal designator>-- ) -------------------------------|
```

Examples:

```
     BOOLEAN B;
     SENDMESSAGE(SIG,PARY1[1]);
     IF SENDRESPONSE(SIGARY1[0,3]) THEN;
     RECEIVEMESSAGE(SIG,PORT2);
     B:=RECEIVERESPONSE(SIGARY1[2,2]);
     CLEAR(SIGARY1[2,2]);
```

Semantics:

The signal statements can also be used as Boolean primaries, in which case a value of TRUE means that the operation was unsuccessful.

The SENDMESSAGE statement causes the message contained in the MESSAGETEXT of the specified signal to be sent via the specified port to the other task in the conversation.

The SENDRESPONSE statement causes the response contained in the RESPONSETEXT of the specified signal to be sent to the task which sent the message via the port through which the message arrived. The text of the response is placed in the RESPONSETEXT of the other task's signal.

The RECEIVEMESSAGE statement causes the first message in the message queue of the specified port to be placed into the MESSAGETEXT of the specified signal. If the message queue is empty, a SIGNALERROR of EMPTYSIGNAL is returned.

The RECEIVERESPONSE statement causes the first response in the response queue of the specified signal to be placed into the RESPONSETEXT of the specified signal. If the response queue is empty, a SIGNALERROR of EMPTYSIGNAL is returned.

The CLEAR statement causes the SIGNALSTATE of the signal to become COMPLETED; i.e., resets the TEXTEVENT, flushes outstanding responses, etc.

1.10 THE SIZE FUNCTION

Syntax and Semantics:

The following can be parameters to the SIZE function:

```
<port array designator>
<port subarray designator>
<signal array designator>
<signal subarray designator>
```

Examples:

```
REAL R;
R:=SIZE(PARY1);
R:=SIZE(SIGARY1[1,*]);
R:=SIZE(SIGARY2[*]);
```

1.11 THE RESIZE STATEMENT

Syntax and Semantics:

A row of a port array or signal array may appear in a RESIZE statement. RETAIN is always done. If an attempt is made to do a downward RESIZE, at runtime the message ATTEMPTED DOWNWARD RESIZE FAILED will be displayed, the RESIZE will not be done, and the program will continue executing.

Examples:

```
RESIZE(PARY1,50,RETAIN);
RESIZE(SIGARY1[1,*],20,RETAIN);
```

2.0 LOGICAL INPUT/OUTPUT INTERFACE TO PORTS AND SIGNALS

The logical input/output interface to ports and signals is available in ALGOL, FORTRAN, COBOL, COBOL74, and PL/I. In this interface, a port is a file whose KIND attribute has the value PORT.

2.1 PORT FILE ATTRIBUTES

The following file attributes have been added for use by port files. Pointer-valued attribute values must end with a period.

MYNAME
Pointer-valued
The value may be null ("."), meaning not specified. The default value is the TASKNAME of the task.

YOURNAME
Pointer-valued
The value may be null ("."), meaning not specified.

HOSTNAME
Pointer-valued
The value may be null ("."), meaning not specified.

For PORT files, the file title is equivalent to the PORTNAME of a PORT. The INTNAME of the file will be used if no title is specified. The TIMEOUT function for SIGNAL and PORT file READ or WRITE statements are equivalent.

The following mnemonic values have been added to the value of field [24:8] of the file attribute STATE. These are valid only if STATE.[3:1]=1.

NOERROR
NETWORKTIMEOUT
MESSAGEREJECTED
RESPONSEREJECTED
CLEARED
TIMEDOUT
PROTOCOL
CHANNELERROR
NOTTRANSMITTED
BUFFERSBUZY
NOTRESPONSEREQUIRED
BADSIGNALNUMBER

A new value for use by port files has been added to the possible values of file attribute AVAILABLE. This new value, equal to 24, indicates that an offer has been made to open the port file but the open operation is not yet complete. If the AVAILABLE attribute has a value of 26, the open cannot be completed because the host is not reachable.

Interrogating the value of file attribute PRESENT for a port file will perform a port offer for that port file, but the offer will not remain outstanding. If there is no outstanding port offer to match this offer, then FALSE will be returned as the value of PRESENT.

The file attribute RESIDENT behaves like file attribute PRESENT when applied to a port file.

Example:

```
IF F.AVAILABLE=26 THEN ERROR(BADHOST);
WHILE F.AVAILABLE=24 DO WHEN (1);  % WAIT UNTIL OPEN
IF F.AVAILABLE=1 THEN BEGIN        %FILE IS OPEN
END;
```

2.2 GENERAL RESTRICTIONS

Some restrictions hold for the logical I/O interface to ports and signals. Restrictions specific to a user language are mentioned in the section pertaining to that language. The general restrictions are:

1. For an explicit or an implicit open of a port file, the open option is WAIT; that is, the task is suspended until the port file is open.

2. When the port file is closed, it is closed with the CLEAR option. Any other file close options that are specified, such as CRUNCH, are ignored.

3. Only FILETYPEs 0 and 3 are allowed. FILETYPE 3 behaves similarly to remote files.

4. The BLOCKSIZE must be equal to MAXRECSIZE. MAXRECSIZE must be LEQ 20,000 characters.

2.3 ASYNCHRONOUS AND SYNCHRONOUS INPUT/OUTPUT

There are two kinds of port file I/O, asynchronous and synchronous. In asynchronous I/O, no signal number is assigned to the signal by the system (the signal number will be zero) and no response is allowed. In synchronous I/O, the system assigns a signal number (which is greater than zero) to each signal and responses are required. A task receiving a message can tell whether a response is required by checking for a nonzero signal number.

2.3.1 ASYNCHRONOUS INPUT/OUTPUT

A signal sent using asynchronous I/O will have a signal number of zero. Responses are not allowed and an attempt to read or write a response to an asynchronous signal will cause an error. The number of buffers declared for a port file will be used by the system to determine the number of signals that the task can have for that file.

Input:

If no input is available when a read operation is attempted, the task will be suspended until input is available.

Output:

Each signal can be reused as soon as the message has been delivered to its destination. A task is suspended until there is an available signal.

2.3.2 SYNCHRONOUS INPUT/OUTPUT

A signal sent using synchronous I/O will have a signal number greater than zero. Each message sent expects a response in return. The number of buffers declared for a port file will be used by the system to determine the number of signals requiring a response that the task can have for that file.

Input:

When a message is read, the system will supply the nonzero signal number of that message to the user program. When a response is read, the user program supplies the signal number and the response with that signal number is read. If, on any read operation, the expected message or response is not available, the operation will wait until the message or response is available.

Output:

When a message is written, the system will return the nonzero signal number of that message to the user program. When a response is written, the user program supplies the signal number and the response is sent to the task which sent the message with that signal number.

2.4 USER LANGUAGE SYNTAX

2.4.1 ALGOL

Declaration and Attribute Use

A port file cannot be a direct file and must have a FILETYPE of 0 or 3.

Examples:

```
FILE P(KIND=PORT,TITLE="A/PORT.");
P.SECURITYTYPE:=VALUE(PRIVATE);
```

Asynchronous I/O

The syntax for asynchronous I/O is that of formatted, freefield, or array row READ and WRITE statements referencing a port file.

Examples of asynchronous I/O:

```
     FILE P(KIND=PORT);
     REAL ARRAY A[0:9];
     READ(P,10,A);
     WRITE(P,<H12>,A[5]);
     WRITE(P,*/,A);
```

Synchronous I/O

The following two elements are added to the definition of <record number or carriage control>:

```
     [SIGNAL <identifier>]
     [RESPONSE <identifier>]
```

where the <identifier> is the name of an integer or real variable. The syntax for synchronous I/O is that of array row READ and WRITE statements referencing a port file and containing one of the above <record number or carriage control> elements. Only array row READ and WRITE statements are allowed to reference port files.

When a WRITE statement containing [SIGNAL <identifier>] is executed, the system assigns a signal number to the message and stores that number in the <identifier>. When a READ statement containing [SIGNAL <identifier>] is executed, the signal number of the message read is stored into the <identifier>. When a WRITE statement containing [RESPONSE <identifier>] is executed, the response is returned to the task which sent the message whose signal number is equal to the value of the <identifier>. When a READ statement containing [RESPONSE <identifier>] is executed, the response whose signal number is equal to the value of the <identifier> is read. The values of the <identifier>s appearing in [SIGNAL <identifier>] and in [RESPONSE <identifier>] should not be changed by the user between the time a message is written or read and the time its response is written or read.

Examples of synchronous I/O:

The following two program segments represent two different tasks communicating with each other through port files. The time sequence is implied by the order of the statements.

```
     FILE P(KIND=PORT);              FILE Q(KIND=PORT,TITLE="P.");
     INTEGER I;                      INTEGER J;
     REAL ARRAY A[0:9];              REAL ARRAY B[0:9];
     .
     .
     WRITE(P[SIGNAL I],10,A);

                                     READ(Q[SIGNAL J],10,B);
                                     .
                                     .
                                     WRITE(Q[RESPONSE J],10,B);

     READ(P[RESPONSE I],10,A);
```

2.4.2 FORTRAN

Declaration and Attribute Use

A port file must have a FILETYPE of 0 or 3.

Examples:

```
FILE  1(KIND=PORT,TITLE="A/PORT.")
      CHANGE(1,SECURITYTYPE=VALUE(PRIVATE))
```

Asynchronous I/O

The syntax for asynchronous I/O is that of READ, WRITE, PRINT, and PUNCH statements referencing a port file.

Examples of asynchronous I/O:

```
FILE  1(KIND=PORT)
      DIMENSION A(10)
 100  FORMAT(X,F13.3)
      READ(1) A
      WRITE(1,100) A[3]
      WRITE(1,*/) A
```

Synchronous I/O

The following two clauses are added to the list of result clauses:

```
     SIGNAL = <identifier>
     RESPONSE = <identifier>
```

where the <identifier> is the name of an integer or real simple variable. The syntax for synchronous I/O is that of unformatted READ and WRITE statements referencing a port file and containing one but not both of the above clauses. Only unformatted READ and WRITE statements are allowed to reference port files.

When a WRITE statement containing a SIGNAL= clause is executed, the system assigns a signal number to the message and stores that number in the <identifier>. When a READ statement containing a SIGNAL= clause is executed, the signal number of the message read is stored into the <identifier>. When a WRITE statement containing a RESPONSE= clause is executed, the response is returned to the task which sent the message whose signal number is equal to the value of the <identifier>. When a READ statement containing a RESPONSE= clause is executed, the response whose signal number is equal to the value of the <identifier> is read. The values of the <identifier>s appearing in the SIGNAL= clause and in the RESPONSE= clause should not be changed by the user between the time a message is written or read and the time its response is written or read.

Examples of synchronous I/O:

The following two program segments represent two different tasks communicating with each other through port files. The time sequence is implied by the order of the statements.

```
FILE  1(KIND=PORT)          FILE  2(KIND=PORT,TITLE="P.")
      DIMENSION A(10)             DIMENSION B(10)
      .
      .
      WRITE(1,SIGNAL=I) A

                                   READ(2,SIGNAL=J) B
                                   .
                                   .
                                   WRITE(2,RESPONSE=J) B

      READ(1,RESPONSE=I) A
```

2.4.3 COBOL and COBOL74

Declaration and Attribute Use

COBOL and COBOL74 have an additional restriction not found in any of the other languages. A file can perform either synchronous or asynchronous I/O, but not both. Which type of I/O a file can perform is specified in the file declaration.

A port file is a file declared in the 'select-clause' with a 'hardware-name' of PORT. For a file to be declared capable of performing synchronous I/O, the declaration of that file must contain a clause providing a numeric data-name for the signal number associated with a signal sent from or received by the port. The absence of this clause from a file declaration declares a file capable of performing asynchronous I/O.

A port file cannot be a direct file and must have a FILETYPE of 0 or 3.

Examples:

```
     SELECT FILE-NAME ASSIGN TO PORT.      % ASYNCHRONOUS I/O
     SELECT FILE-NAME ASSIGN TO PORT
          SIGNAL IS DATA-NAME.             % SYNCHRONOUS I/O
```

Asynchronous I/O

The syntax for asynchronous I/O is that of READ and WRITE statements referencing a port file.

Examples of asynchronous I/O:

```
     SELECT FILE-NAME ASSIGN TO PORT.
     .
     .
     .
     READ FILE-NAME.
     WRITE RECORD-OF-FILE-NAME.
```

Synchronous I/O

The syntax for READ and WRITE statements has been extended to specify reading and writing the response to a message:

```
     READ RESPONSE <filename>
     WRITE RESPONSE <recordname>
```

The syntax for synchronous I/O is that of READ, WRITE, READ RESPONSE, and WRITE RESPONSE statements referencing a port file. The READ RESPONSE and WRITE RESPONSE statements can only reference port files.

When a WRITE statement referencing a synchronous I/O port file is executed, the system assigns a signal number to the message and updates the data-name associated with the signal to that number. When a READ statement referencing a synchronous I/O port file is executed, the data-name associated with the signal is updated with the signal number of the message read. When a WRITE RESPONSE statement is executed, the response is returned to the task which sent the message whose signal number is equal to the value in the data-name associated with the signal. When a READ RESPONSE statement is executed, the response whose signal number is equal to the value in the data-name associated with the signal is read. The value in a data-name associated with a signal should not be changed by the user between the time a message is written or read and the time its response is written or read.

Examples of synchronous I/O:

The following two program segments represent two different tasks communicating with each other through port files. The time sequence is implied by the order of the statements.

```
SELECT P                      SELECT Q
     ASSIGN TO PORT                ASSIGN TO PORT
     SIGNAL IS I.                  SIGNAL IS J.
     .
     .
WRITE RECORD-OF-P.

                              READ Q.
                              .
                              .
                              WRITE RESPONSE RECORD-OF-Q.

READ RESPONSE P.
```

2.4.4 PL/I

Declaration and Attribute Use

The FILETYPE of a port file must be 0 or 3.

Examples:

```
DECLARE P FILE RECORD ENVIRONMENT (KIND='PORT');
SECURITYTYPE (P) = 'PRIVATE';
```

Asynchronous I/O

The syntax for asynchronous I/O is that of record READ and WRITE statements and stream GET and PUT statements referencing a port file.

Examples of asynchronous I/O:

```
DCL P FILE RECORD ENV (KIND='PORT',MAXRECSIZE=60);
DCL P2 STREAM FILE ENV (KIND='PORT');
DCL A CHARACTER (60);
READ FILE (P) INTO (A);
WRITE FILE (P) FROM (A);
PUT FILE (P2) LIST (X,Y,Z);
```

Synchronous I/O

The following two elements are added to the definition of <option-list> in an input/output statement:

```
SIGNAL (<identifier>)
RESPONSE (<identifier>)
```

where the <identifier> must be of arithmetic, character string, or bit string type. The syntax for synchronous I/O is that of record READ and WRITE statements referencing a port file and containing one of the above <opt on-list> elements. Only record READ and WRITE statements are allowed to reference port files.

When a WRITE statement containing SIGNAL (<identifier>) is executed, the system assigns a signal number to the message and stores that number in the <identifier>. When a READ statement containing SIGNAL (<identifier>) is executed, the signal number of the message read is stored into the <identifier>. When a WRITE statement containing RESPONSE (<identifier>) is executed, the response is returned to the task which sent the message whose signal number is equal to the value of the <identifier>. When a READ statement containing RESPONSE (<identifier>) is executed, the response whose signal number is equal to the value of the <identifier> is read. The values of the <identifier>s appearing in SIGNAL (<identifier>) and in RESPONSE (<identifier>) should not be changed by the user between the time a message is written or read and the time its response is written or read.

Examples of synchronous I/O:

The following two program segments represent two different tasks communicating with each other through port files. The time sequence is implied by the order of the statements.

```
DCL P FILE RECORD                  DCL Q FILE RECORD
     ENV (KIND='PORT',                  ENV (KIND='PORT',TITLE='P.',
          MAXRECSIZE=60);                    MAXRECSIZE=60);
DCL I FIXED BINARY;                DCL J FIXED BINARY;
DCL A CHARACTER (60);              DCL B CHARACTER (60);

WRITE FILE (P) FROM (A)
               SIGNAL (I);
                                   READ FILE (Q) INTO (B)
                                                 SIGNAL (J);

                                   WRITE FILE (Q) FROM (B)
                                                  RESPONSE (J);

READ FILE (P) INTO (A)
              RESPONSE (I);
```

3.0 WFL EXTENSIONS FOR INTER-PROCESS COMMUNICATION

3.1 FILE ATTRIBUTE HANDLING

There is a new file attribute, HOSTNAME, for specifying the name of the host system at which a file exists.

If a file's HOSTNAME is not specified, the file will be searched for on the host system where the file open request originates.

3.2 TASK ATTRIBUTE HANDLING

There is a new string-valued task attribute, HOSTNAME, which specifies the host system at which the task will be initiated. The default value is the host system where the task initiation request originates.

There is a new read-only string-valued task attribute, ORGHOSTNAME, which specifies the name of the host which caused this task to be initiated.

To task attribute HISTORYTYPE has been added the value UNKNOWNEOTV, which indicates that after the time a task was initiated at a remote host and before the time it completed, communication between the systems was lost and was not restored until after the task had completed.

To task attribute HISTORYCAUSE has been added the value NETWORKCAUSEV, and to task attribute HISTORYREASON has been added the value DISCONNECTEDV.

3.3 EXPANDED LIBRARY MAINTENANCE SYNTAX

The <volume specification> for Library Maintenance statements has been expanded to allow specification of a HOSTNAME.

<disk volume specification>

```
  |<---------------------- , ----------------------|
  |                                                |
----- AREACLASS -- = -- <integer expression>--------------------------|
    |                                             |
    |- FAMILYINDEX -- = -- <integer expression> --|
    |                                             |
    |- HOSTNAME -- = -- <string expression> ------|
```

<pack volume specification>

```
  |<---------------------- , ------------------------|
  |                                                  |
----- SINGLEPACK ---------------------------------------------------|
    |                 |                                |
    |                 |-- = -- <boolean expression> ---|
    |                                                  |
    |- FAMILYINDEX -- = -- <integer expression> -----|
    |                                                  |
    |- HOSTNAME -- = -- <string expression> ---------|
```

<family volume specification>

```
   |<------------------------- , -------------------------|
   |                                                      |
----- INTERCHANGE ----------------------------------------------------|
    |                    |                                 |
    |                    |-- = -- <boolean expression> ---|
    |                                                      |
    |----------------------- PETAPE ------------------|
    |    |             |    |                              |
    |    |-- KIND -- = --|   |- TAPE9 --------------------|
    |                         |                             |
    |                         |- TAPE7 --------------------|
    |                         |                             |
    |                         |- TAPE ---------------------|
    |                         |                             |
    |                         |- PACK ---------------------|
    |                                                      |
    |- FAMILYINDEX -- = -- <integer expression> ------|
    |                                                      |
    |- SERIALNO -- = -- <serial number list> ---------|
    |                                                      |
    |- HOSTNAME -- = -- <string expression> ----------|
```

Examples:

```
     COPY FILEID FROM SOMEPACK(KIND=PACK,HOSTNAME=HOSTID);
     REMOVE FILEID FROM DISK(HOSTNAME=HOSTID);
```

3.4 JOBS TO OTHER HOSTS

A WFL job may request to be executed at a host different from the one where it was submitted.

For example,

```
     <I> AT Q BEGIN JOB X;
     . . .
     <I> END JOB.
```

will cause the WFL source records to be transferred to host Q for interpretation and execution. The records are transferred transparently.

The notation "<I>" stands for the invalid character code required for jobs entering the system from the card reader. The invalid character is not allowed between the host specification and the "BEGIN". This insures that jobs intended to be executed locally need only begin with

```
     <I> BEGIN JOB R;
     . . .
```

in order to be caught before being sent to Q in the event that a

```
     <I> AT Q
```

card was left in the reader.

DMALGOL Implementation

The following note describes the DMALGOL language, which was implemented in previous releases but was largely undocumented. This note also incorporates significant D-notes from previous release documentation.

1. INTRODUCTION

DMSII requires a special compiler for two reasons: The compiler must be able to retrieve information from the description file and it must build stack structures which are different from those in ALGOL. DMALGOL consists of ALGOL with extensions to meet these requirements.

The DMALGOL language is an "implementation" language for the DMSII system and is not intended for general use. New features may be added at any time and existing features may be changed or deleted without notice. For this reason, users should not rely upon DMALGOL as an application programming language.

2. OVERVIEW

2.1 Compile-Time Facilities

DMALGOL compile-time facilities consist of elements which are of a general nature and have been documented in ALGOL, and elements specific to DMSII.

The basic technique used to write the ACCESSROUTINES is conditional compilation. Various pieces of code are omitted, included and parameterized based on information from the DASDL description file. This file is read by the DMALGOL compiler; the DMALGOL language contains elements used to conveniently reference its information.

NODE variables and PROPERTY definitions are used to reference "nodes" within the description file. A node is a data structure which consists of two parts: a list and a set of properties, either of which may be absent. The elements of lists are frequently other nodes, but they may be integers or other data items. The syntax in DMALGOL to reference list element I of node N is N[I]. The properties of a node are in a non-homogeneous substructure whose format is defined by PROPERTY declarations.

Example:

```
NODE N;
INTEGER T;
PROPERTY RECORDSZ=[11].[11:12];
T:=N.RECORDSZ;
```

This will extract the property RECORDSZ of the node N. The properties actually used in the ACCESSROUTINES are included from the file called DATABASE/PROPERTIES.

A special form of the 'FOR statement is used to access the members of a node's list. It uses the fact that DASDL stores the number of list items at OFFSET=0 in the list, so that the compiler knows how many elements are in the list.

Example:

```
NODE STRUCTURE, SPANSET;
'FOR EACH SPANSET OF STRUCTURE DO . . .
```

This statement goes through each member of the list belonging to the node STRUCTURE, assigning its successive list elements to the node SPANSET.

A special form of the 'INCLUDE statement is used to access the text section of the DASDL description file. This section consists of a set of source language DMALGOL constructs; these "texts" are always referenced via an appropriate property. For example, data set nodes contain a property called VERIFYSTORETEXT which checks whether a record meets all verification rules before it is stored. To access this text for node DS, the following would be written:

```
'INCLUDE DS.VERIFYSTORETEXT.
```

DMALGOL would extract the appropriate text and compile it.

2.2 Environments

A normal ALGOL program contains block initialization code to reserve space in the stack for its local variables. DMSII environments, however, are not contained in a running stack and cannot, therefore, execute code to build the stack. Rather, the MCP procedure DMSOPEN sets up the stack variables based on an "image" of the DBS and S1B stacks kept in the ACCESSROUTINES code file. Thus, one task of DMALGOL is to build these stack images. This is done by using the ENVIRONMENT declaration, which delimits the boundaries of each environment, in much the same way as a procedure declaration does. Environments may not be nested more than three deep.

DMSII uses environments at all three levels. The outermost environment contains global variables common to the entire data base. Intermediate level environments, DBS environments, contain variables unique to each data base structure. The innermost environments, SIB environments, contain the variables which are unique to a single invocation of a structure.

It is possible to call a procedure in a different environment using the following construct:

```
     <procedure name> ' <structure number>
```

<structure number> is the number declared in the ENVIRONMENT declaration containing the desired procedure.

Example:

```
     ENVIRONMENT X OF 2;
     BEGIN
          PROCEDURE P;
     END;

     ENVIRONMENT X OF 3;
     BEGIN
          PROCEDURE P;
          P;
          P'2
     END;
```

The call P will call the routine in its local environment; i.e., the second one. The call P'2 will call the routine in environment 2; i.e., the first one.

2.3 Reference Variables

DMALGOL provides the ability to declare procedure reference variables, file reference variables and direct file reference variables. These reference variables allow for dynamic selection of procedures, files and direct files within the DMSII system.

2.4 "ONCE ONLY" Compilation

Each structure of the data base has one D2 and one D3 environment compiled for it. The same lines in DATABASE/SYMBOLIC are typically compiled many times over. For example, the procedure STORE is compiled for every data set. Conditional compile-time statements alter it to fit each successive data set; each has its own VERIFYSTORETEXT included, for example, and calls to insert it into each of its automatic sets.

Some procedures, however, do not differ in any major respect between two structures of the same type. An example is GENERATE; the main difference in its operation is the block size and end-of-file, which are easily parameterized. To save compile time and code space, such procedures are compiled only once. It is not possible to use calls on the shared procedure using the <procedure name>'<structure number> syntax, since that would change the environment as well. Rather, a PCW is constructed for each structure; for a shared procedure, the PCW in each structure will point to the same segment descriptor. This allows the procedure to be both shared by different environments, but behave as if it were local to each environment.

3. DMALGOL EXTENSIONS

3.1 Compile-Time Facilities

3.1.1 NODE Declaration

Syntax:

```
               |<---------------------- , ----------------------|
               |                                                |
-- NODE ---<identifier>-----------------------------------------|
                           |                              |
                           |_ * __________________________|
                           |                              |
                           |- :=<arithmetic expression 1> -|
```

Semantics:

1. An identifier declared to be a NODE is a compile-time variable similar to a NUMBER; however, it must be used in conjunction with a compile-time array created by DASDL.
2. A node variable represents an index into the DASDL array.
3. The value of a node variable (index into the DASDL array) may be changed at any time during compilation by means of a compile-time 'LET statement.
4. <arithmetic expression 1>, if used, must be a constant arithmetic expression. It represents the initial index of the node variable. By default, the initial value is zero, which is otherwise an illegal value; thus, a node must be initialized or assigned a value before it is used.
5. If the "*" is used, that node variable is a special read-only node which may be used to reference any word in the entire DASDL array. If N is such a node, then N[<i>] is the <i>th word of the DASDL array.
6. Normally, the node variable represents the index of a "node" in the DASDL array. A list and a set of properties is associated with a node; the list is usually a list of nodes, and the properties contain values. The node variable may be used to reference members of the list and values of the properties, as follows: If N is a node variable and P is a property identifier, then N[<i>] is the <i>th member of the list of N, and N.P is the value of the property P of N (<i> is a constant or constant expression).

3.1.2 PROPERTY Declaration

Syntax:

```
------------- PROPERTY --------------------------------------->
  |        |
  |-<type>-|

  |<--------------- , ----------------|
  |                                   |
>---<identifier>-- = --<property spec>------------------------|
```

```
<property spec>

-- [ --<arithmetic expression 1>-- ] ------------------------>

>--------------------------------------------------------------|
  |                                                          |
  |- .[<arithmetic expression 2>:<arithmetic expression 3>] -|
```

Semantics:

1. A property identifier defines the location and format of a property value associated with a node in the DASDL array.
2. Each arithmetic expression must be a constant arithmetic expression. <arithmetic expression 1> specifies the word in the property set. <arithmetic expression 2> and <arithmetic expression 3> specify a field within the word; if they are not given, the entire word is used.
3. If a type is declared, it must be a single-precision arithmetic or Boolean; when no type is specified, REAL is assumed.
4. A property identifier is used only with a node variable, as follows:

   ```
   --<node variable>-- . --<property identifier>--|
   ```

   This construct represents the value of the property for the given node. It may be used wherever constants of the specified type may appear.
5. The "node property" construct is only used to retrieve a value from the DASDL array; it may not be used to change a property value (in fact, no construct can change the DASDL array: it is read-only).

3.1.3 'FOR Statement Extensions

Syntax:

```
-- 'FOR --- EACH -----<nodeid 1>---- OF --<nodeid 2>---------->
          |         | |                |
          |- ALL --| |-<number id>-|

>---------------------------------- DO ---<ct statement>-----|
  |                                  |      |                 |
  |- WHERE --<boolean expression>-|      |-<begin clause>-|
```

```
<begin clause>

-- 'BEGIN --<text 1>-- 'NEXT --<text 2>-- 'PRIOR --<text 3>--->

>- 'END ---------------------------------------------------|
```

Semantics:

1. This statement provides for iterative compilation of ALGOL source code. One iteration is made over the source code for each entry in the list belonging to <nodeid 2>.
2. <nodeid 2> must be the index of a node in the DASDL array.
3. For each iteration, the control variable is assigned the index of the list element (if <nodeid 1> is specified) or the contents of the list element (if <number id> is specified).
4. If the WHERE clause is used, a constant <boolean expression> must be specified. Before the statement following DO is processed but after the control variable has assumed its next value, the <boolean expression> is evaluated. If the expression is TRUE, the statement following DO is processed; otherwise, the statement is ignored.
5. If the body of the 'FOR statement is the 'BEGIN-'NEXT-'PRIOR-'END form, a "telescoping" form of iteration is performed. In that case, the 'NEXT behaves like an 'END, so that the <text 1> is processed for each iteration. After the last iteration, <text 2> is processed just once. Finally, the iteration is repeated, backwards, for <text 3>. If a WHERE clause is used, the backwards iteration processes exactly the same cases as the first iteration; i.e., the <boolean expression> is not re-evaluated while backing out of the telescope.
6. <ct statement> is a compile-time statement including any DMALGOL extensions.

3.1.4 'INCLUDE Statement Extensions

Syntax:

```
-- 'INCLUDE --<nodeid>-- . --<property id>--|
```

Semantics:

1. This statement causes the compiler to process text directly out of the DASDL array.
2. The value of the specified property is assumed to be an index of text in the DASDL array. That text must be terminated by a pound sign (#) and at least one null character (4"00").

3.1.5 'LET Statement Extensions

Syntax:

```
-- 'LET --<number variable>-- := --<procedure name>--|
```

Semantics:

This statement stores a copy of the procedure's PCW in the number variable. This construct is intended for use with compile once only procedures. Refer to the section on "ONCE ONLY" Compilation for additional details.

3.1.6 'PRINT, 'ERROR, 'DISPLAY Statements

Syntax:

```
---- 'PRINT ------------------------------------------|
  |                  |  |                             |
  |- 'ERROR ---|     |  | |<----------- , ----------| |
  |            |     |  | |                         | |
  |- 'DISPLAY -|     |  |-----<nodeid>-------------------|
                          |                       |
                          |-<string>--------------|
                          |                       |
                          |-<arithmetic expression>-|
```

Semantics

1. This statement causes one or more lines to be printed on the compilation listing.

2. Each line may contain up to 87 characters, starting in position 18.

3. If the 'ERROR form is used, the compiler's error count is incremented by one, and the printed line is bracketed by the ">" and "<" characters, as for a normal syntax error message.

4. The lines are created from the information given in the statement, as follows:

   a. If a <nodeid> is specified, the alpha name property for the node is inserted into the line.

   b. If a string is specified, it is inserted into the line.

   c. If an <arithmetic expression> is used, it must be a constant arithmetic expression. It is assumed to be an integer, and the value of the integer, zero suppressed, is inserted into the string.

   No blanks are inserted into the line between specified items.

5. If a 'PRINT or 'ERROR statement is processed (not skipped) by the compile-time processor, the lines are printed whether or not any listing $ card options are set (LIST, LISTOMIT, etc.).

3.1.7 PROCEDURE Declaration Extensions

Syntax:

Form 1:

```
-- PROCEDURE --<procedure id 1>-- := --<procedure id 2>--|
```

Form 2:

```
--<procedure type>-- PROCEDURE --<procedure heading>-- ; ----->
>- EXTERNAL --<number variable>---------------------------------|
```

Semantics:

Two forms of procedure declaration are used for compile once only procedures. Refer to the section on "ONCE ONY" Compilation for additional details.

3.2 ENVIRONMENT Declaration

Syntax:

```
-- ENVIRONMENT --<identifier>----------------------------------->
>- ( <arithmetic expression 1> ) ------------------------------->
>--------------------------------------- BEGIN ----------------->
  |                                 |
  |- OF --<arithmetic expression 2>-|

  |<------ ; -----|
  |               |
>---<declaration>--- END ---------------------------------------|
```

Semantics:

1. The ENVIRONMENT declaration is used to declare the contents of the Data Base Stack (DBS) and Set Information Block (SIB) stacks for a data base.

2. The entire "program" for a data base is an environment containing all data base globals.

3. Following the data base globals, one or more DBS environments may be declared. One DBS environment is present for each data set or set in the data base.

4. Within each DBS environment, and after other DBS items, one or more SIB environments must be declared.

5. The identifier specified in the ENVIRONMENT declaration is primarily for documentation; it is not used again. <arithmetic expression 1> is a constant used to aid structure allocation within the ACCESSROUTINES. <arithmetic expression 2> (a constant, normally a node variable) is the environment identification. Every DBS environment must have a distinct environment identification. Each SIB environment must have an environment id equal to that of the DBS which contains it. The outer level environment is not declared with an environment id.

6. In a DBS environment, only global items and DBS items declared in that DBS are accessible. In a SIB environment, all global items, all items in the corresponding DBS environment, and all SIB items are accessible. In addition, all procedures declared in prior SIB environments are accessible. Those procedures are referenced via dynamic identifiers, using the environment id after the apostrophe.

3.3 Reference Variables

3.3.1 PROCEDURE References

Declaration Syntax:

(Form 1)

```
-- PROCEDURE REFERENCE --<procedure reference id>-- := ------->

>-<procedure id>---------------------------------------------|
```

(Form 2)

```
-- PROCEDURE REFERENCE --<procedure reference id>------------->

>-<formal parameter part>--<procedure body>------------------|
```

Assignment Syntax:

```
--<procedure reference id 1>-- := --------------------------->

>---<procedure id>-------------------------------------------|
  |                |
  |-<procedure reference id 2>-|
```

Semantics:

Form 1 of the declaration declares the procedure references variable <procedure reference id>. The initial value of this variable refers to <procedure id>. The parameter description for <procedure reference id> is derived from <procedure id>.

Form 2 of the declaration declares the procedure reference variable <procedure reference id 1>, specifies the parameter description for the reference variable and specifies a body of code to be executed if <procedure reference id 1> is used before an assignment is done to it.

The procedure reference assignment statement causes the <procedure reference id 1> to refer to the procedure specified on the right side of the ":=". Both procedures must be of the same type and have the same parameter descriptions.

A <procedure reference id> may be used to invoke a procedure as if the procedure were invoked directly.

3.3.2 File References

Declaration Syntax:

```
                                   |<--------- , --------|
                                   |                     |
--------------- FILE REFERENCE ---<file reference id>--- ; --|
  |           |
  |- DIRECT -|
```

Assignment Syntax:

```
--<file reference id 1>-- := ---<file id>--------------- ; --|
                                |                       |
                                |-<file reference id 2>-|
```

Semantics:

1. Prior to being assigned a value, a procedure reference variable does not point to a file.

2. In the file reference assignment statement, the left and right file identifiers must both be DIRECT or must both be non-DIRECT.

3.4 "ONCE ONLY" Compilation

Compile once only procedures may be declared in two ways.

3.4.1 Form 1

Syntax:

<procedure declaration>

```
-- PROCEDURE --<procedure id 1>-- := --<procedure id 2>--|
```

Example:

```
BOOLEAN PROCEDURE GETADDRESS (A);
     VALUE A;
     REAL A;
     BEGIN
       .
       .
       .
     END;
PROCEDURE GETADDR := GETADDRESS;
```

Semantics:

This is the most common type of compile once only procedure. <procedure id 1> and <procedure id 2> are identical. The parameters and procedure type of the two procedures must be exactly the same. The body of the procedure is compiled only once. Subsequent references to the procedure merely copy the PCW for <procedure id 2> into the environment that contains <procedure id 1>.

This technique can only be used for procedures declared in the outer block of a DBS or SIB environment. Embedded procedures must use the alternate form of compile once only.

3.4.1 Form 2

Syntax:

<let statement>

```
-- 'LET --<number variable>-- := --<procedure name>--|
```

<procedure declaration>

```
--<procedure type>-- PROCEDURE --<procedure heading>----->

>- ; -- EXTERNAL --<number variable>---------------------|
```

Example:

```
NUMBER GETADDRESSPCW;
BOOLEAN PROCEDURE GETADDRESS (A);
     VALUE A;
     REAL A;
     BEGIN
        .
        .
        .
     END;

'LET GETADDRESSPCW := GETADDRESS;

BOOLEAN PROCEDURE GETADDR (A);
     VALUE A;
     REAL A;
     EXTERNAL GETADDRESSPCW;
```

Semantics:

This type of compile once only procedure must be used very carefully. The user must ensure that the procedure declarations are identical. DMALGOL does not check the procedure declarations for consistency; it simply copies the PCW for the original procedure into the environment containing the EXTERNAL procedure declaration.

3.5 DMIO File Attribute

The DMIO direct file attribute is used by the ACCESSROUTINES to indicate that the DMSII system is using the file.

3.6 DMALGOL Functions and Statements

3.6.1 ALLOW and DISALLOW Statements

Syntax:

```
---- ALLOW -------|
   |             |
   |- DISALLOW -|
```

Semantics:

The DISALLOW statement disables external interrupts. The ALLOW statement enables external interrupts. These statements are intended for use in the ACCESSROUTINES and are used in critical sections where a series of operations must be performed without interruption.

3.6.2 ATTACHDBS Statement

Syntax:

```
-- ATTACHDBS -- ( --<arithmetic expression>-- ) --|
```

Semantics:

<arithmetic expression> must be a constant expression which identifies a D2 environment to which the currently active D3 environment will be linked up. The Mark Stack Control Word (MSCW) at the base of the current D3 environment is linked to the MSCW at the base of the specified D2 environment. The effect of this operation is not immediate since the processor's D2 register is not effected by the relinking done by ATTACHDBS. Calling a procedure declared at lex level 2 is sufficient to force the necessary display register update.

3.6.3 DMINQ Functions

The DM INQUIRY interface permits direct communication with the DMSII ACCESSROUTINES. It is intended only as a tool for use in implementing special purpose packages, such as on-line inquiry, and is specifically not intended for general user interface to DMSII.

It is assumed that exactly one data base is invoked in the normal manner in the program which uses these constructs; otherwise, a syntax error will result.

I. DMINQ [<arithmetic expression>] (<array row>)

. The <arithmetic expression> is the SIB index for the desired structure. The <array row> is a one-dimensional array used to communicate with the ACCESSROUTINES. The contents of the array control the function performed by the system.

A [0] = identifies desired procedure

```
 1 = pathfinder (find key only)
10 = set to beginning
11 = data finder (find/lock next/current)
12 = getdata
13 = DMS read (access data portion only)
14 = get link
16 = store current
17 = free current
23 = create
24 = delete current
```

A [i], i>0, depend upon A[0] as follows:

1. Pathfinder (A[0] =1)

A[1] = FIND op :

```
-1 = current record
 0 = next in set
 1 = next in set = UKA
 2 = next in set > UKA
 3 = next in set >= UKA
 4 = link in set
```

A [2] = SZ parameter to pathfinder

This parameter specifies the size of the key passed to pathfinder in the user key area. If SZ is less than zero (0), then ABS(SZ) is the number of hex characters in the user key; otherwise, SZ is the number of bytes in the user key.

A [3] = SZ2 parameter to pathfinder

This parameter specifies the size of the major portion of the user key which must exactly match the retrieved key. If SZ is less than zero (0), then ABS(SZ2) is the number of hex characters in the user key; otherwise, SZ2 is the number of bytes in the user key.

As an example, if an index set had a concatenated key (A,B,C) with each portion two bytes long, to retrieve the next key where A=UKA and B=UKA requires SZ=4 and SZ2=4. To retrieve the next key where A=UKA and B>UKA requires SZ=4 and SZ2=2.

If a record is found as specified, the AA word is returned in A [1]. A getdata call must be used to move the record to the user's work area.

2. Set to beginning (A[0] = 10)

A [i], i>0, not used.

3. Datafinder (A[0] = 11)

A[1] = FIND op:

```
0 = find current data set
1 = lock current data set
2 = find next data set
3 = lock next data set
```

If a record is found, its AA word is returned in A[1], and the record is moved to the user's work area.

4. Getdata (A[0] = 12)

A[1] = AA word of desired record. Moves the desired record to the user's work area.

5. DMSREAD (A[0] = 13)

   A[1] = AA word of desired record.

   This function is similar to the Getdata call, except that only the data portion of the desired record is moved to the user's work area; structures embedded in the accessed record are unaffected.

6. Get link (A[0] = 14)

   "Fetchkey" call on control manager:

   A[1] = Unused
   A[2] = LLOC parameter
   A[3] = LLEN parameter

   The link entry is returned in the array A[*].

7. Store current (A[0] = 16)

   A[i], i>0, not used.

8. Free current (A[0] = 17)

   A[i], i>0, not used.

II. Three "intrinsic" arrays

Note that these arrays may be used only in array reference assignment statements. They may not be used as ordinary arrays.

1. DMKEYAREA

   Hex array; the user's key area for the one SIB invoked.

2. DMWORKAREA [X]

   Hex array; the user's work area for a particular structure.

   X is an arithmetic expression, value = SIB index for that structure (see below).

3. DMSIBDESC

   Real array; the SIB description for the SIB invoked. Used primarily to determine the SIB index for each invoked structure. The first N words of this array (0 to N-1) contain the structure number (plus other stuff) for each invoked structure. The index in the SIB description of this word is the value of the SIB index for that structure. N is equal to SIZE(DMSIBDESC)-8.

3.6.4 DMTRANSLOCK Statement

Syntax:

```
-- DMSTRANSLOCK -- ( --<formal array 1>-- , ------------------>

>-<formal array 2>-- ) ---------------------------------------|
```

Semantics:

DMTRANSLOCK is intended only for use in compilation of ACCESSROUTINES. It performs transaction locking for DMS jobs where <formal array 1> refers to a transaction lock and <formal array 2> is its new value.

3.6.5 DMSCAUSE Statement

Syntax:

```
-- DMCAUSE -- ( --<arithmetic expression>-- ) --|
```

Semantics:

The DMSCAUSE statement calls the MCP procedure DMSCAUSE and passes a single real-valued parameter. The effect of the call is dependent on the parameter, as follows:

parameter < 0 Indicates that the calling program has left transaction state. The program is delinked from the transaction state linkage chain.

parameter = 0 Indicates that a syncpoint has been completed. All programs waiting for a syncpoint for this data base are awakened.

parameter > 0 Indicates that a record for which other users are waiting has been freed. The parameter is the stack number of the previous owner. All programs waiting on that stack number will be awakened.

3.6.6 DMSFREE Statement

Syntax:

```
-- DMSFREE --|
```

Semantics:

This statement calls the MCP procedure DMSFREE, which causes all records locked by this process to be freed in every data base visible to the process.

3.6.7 DMSUPDATEDISKHEADER Statement

Syntax:

```
-- DMSUPDATEDISKHEADER -- ( --<file designator>-- ) --|
```

Semantics:

This statement causes the disk header for the designated file to be flushed to the directory.

3.6.8 DMSWAIT Function

Syntax:

```
-- DMSWAIT -- ( --<arithmetic exp. 1>-- , -------------------->

>-<arithmetic exp. 2>-- , --<arithmetic exp. 3>-- , ---------->

>-<array identifier>-- ) -------------------------------------|
```

Semantics:

The DMSWAIT function is a Boolean-valued function with four parameters. The DMSWAIT function calls the DMSWAIT procedure in the MCP. The first three parameters are real-valued and the last is an array. The effect of the DMSWAIT function is dependent on the values of parameters 2 and 3.

a. Parameter 2 = 0, parameter 3 = 0

Indicates that the calling program needs to wait for a syncpoint to complete on this data base. The program is linked into the sleeper chain for the data base and is suspended. When a syncpoint is complete on this data base (indicated by some other program doing a DMSCAUSE(0)), this program will be awakened and a result of FALSE returned. If a deadlock is detected, this program is awakened and returned a result of BOOLEAN(1). If the program has specified a wait limit and the limit expires before the syncpoint occurs, the program is awakened and returned a value of BOOLEAN(3).

Parameter 1 is a control word obtained from the location specified by parameter 3 in the array given as parameter 4. If the value of the control word from the array changes before suspending the program, a value of FALSE is immediately returned.

b. Parameter 2 = 0, parameter 3 = -1

Indicates that the calling program has entered transaction state. The program is linked into the transaction state linkage for the appropriate data base. If at the time of this call the program is already in transaction state, a result of TRUE is returned; otherwise, a result of FALSE is returned.

c. Parameter 2 = -1, parameter 3 = -1

If the number of processes waiting for locked records in this data base is less than the first parameter, a result of TRUE is returned; otherwise, a result of FALSE is returned.

d. Parameter 2 > 0, parameter 3 > 0

Indicates that the calling program needs to wait for a locked record. Parameter 2 is the stack number of the current owner. The calling program is linked into the sleeper chain for the current owner of the record, the sleep count for this data base is incremented by one, and the program is suspended. When the current owner of the record frees it (indicated by doing a DMSCAUSE (<owner's stack number>)), this program is awakened and a result of FALSE returned. If a deadlock is detected, a result of BOOLEAN(1) is returned. If the program has specified a wait limit and the limit expires before the record is freed, the program is awakened and returned a value of BOOLEAN(3).

Parameter 1 is a control word obtained from the location specified by parameter 3 in the array given as parameter 4. If the value of the control word from the array changes before suspending the program, a value of FALSE is immediately returned.

3.6.9 DSED Function

Syntax:

```
-- DSED --|
```

Semantics:

The DSED function is a parameterless function which returns a Boolean result of TRUE if the program is DSed, and FALSE otherwise.

3.6.10 DSWAIT and DSWAITANDRESET Functions

DSWAIT and DSWAITANDRESET constructs are identical to WAIT and WAITANDRESET, except that a result of -1 indicates the job was DSed while waiting. Note that these constructs are intended for DMSII ACCESSROUTINES only.

3.6.11 NEWDOPEVECTOR Function

Syntax:

```
-- NEWDOPEVECTOR -- ( --<array identifier>-- , --------------->

>-<arithmetic exp>-- ) ---------------------------------------|
```

Semantics:

The NEWDOPEVECTOR function is a Boolean function with two parameters. The first parameter is a 2-dimensional array designator and the second is the new size for the first dimension. The new size may be larger or smaller than the current size. If the new size is smaller, rows will be deallocated. If the new size is larger, unallocated rows will be added. Care must be taken when using this function, since no copy descriptors will be fixed up.

3.6.12 SIBOFFSET Function

Syntax:

```
-- SIBOFFSET -- ( --<procedure name>-- ) --|
```

Semantics:

The SIBOFFSET function is a compile-time function that accepts a procedure name as its only parameter. The result of this function is the offset of this procedure in its environment.

3.6.13 SNR Function

Syntax:

```
-- SNR --|
```

Semantics:

The SNR function is a parameterless function which returns as a real value the current value of the processor's stack number register.

4. DMALGOL Reserved Words

The reserved word type is indicated following each word. Refer to the ALGOL Language Reference Manual (Form No. 5001639) for details.

ALLOW (2)
ATTACHDBS (2)
DMINQ (2)
DMKEYAREA (2)
DMWORKAREA (2)
DMSIBDESC (2)
DISALLOW (2)
DSWAIT (2)
DSWAITANDRESET (2)
DMIO (2)
DMSCAUSE (2)
DMSFREE (2)
DMTRANSLOCK (2)
DMSWAIT (2)
DMSUPDATEDISKHEADER (2)
DSED (2)
ENVIRONMENT (2)
NEWDOPEVECTOR (2)
NODE (2)
PROPERTY (1)
SIBOFFSET (2)
SNR (2)

CHANGES

TO

SYSTEMSTATUS

INTRODUCTION

This note details differences between SYSTEMSTATUS in the III.1 MCP and previously-documented versions.

The bulk of the changes fall into two categories:

a. Extension of types 1, 2 and 7 to provide "box-by-box" data on the various subsystems of a B 6800 Multiprocessor (tightly-coupled or TC) system.

b. Revision of data returned by types 3 and 9 because of substantial changes within the MCP.

Additional changes include the following:

c. A few new data items in type 2.
d. Minor changes to type 4 data.
e. Minor changes to IOTRACE data in type 7.
f. New process-time breakdown in type 7.
g. Revision of data in type 10.
h. New type 11 - Swapper parameters.
i. New type 14 - Suspender parameters.

The next section explains the scheme used for box-relative data. The succeeding sections discuss specific changes by type.

A B 6800 Multiprocessor system contains a Global memory subsystem and one or more local B 6800 subsystems. Each subsystem is called a "box" within the MCP. Some of the data which are reported by SYSTEMSTATUS will vary from one box to another. The SYSTEMSTATUS interface has been extended to make such data available.

To accomodate tightly-coupled systems, the V1 parameter used in calling SYSTEMSTATUS has been redefined as follows:

Field	Meaning
4:5	Group selection field (previously the entire word).
29:5	Bit mask field for requesting box relative information:

Bit	Box Number
29	5 (Local Processor 4)
28	4 (" " 3)
27	3 (" " 2)
26	2 (" " 1)
25	1 (Global Memory)

The bit mask field is relevant only for types 1, 2 and 7; it is ignored in all other types. If a bit mask is provided for one of these valid types, a separate block of data will be returned for each selected box. The structure of the data returned by SYSTEMSTATUS is as follows:

```
 ---------------------     ------------
 : GS : M : Box Info : . . . : Box Info :
 ---------------------     ------------
          \_________/       \_________/
               :                 :
             Lowest           Highest
            Numbered          Numbered
              Box               Box
```

The above structure contains first an overall group-size word (at index 0), a bit mask word (at index 1) indicating the boxes for which data are being returned, and then groupings of box relative information (each with its own group-size word). A minor exception to this rule is the Type 7 call where only the times and count information is repeated (see Type 7 call for further explanation). Note that requests for information pertaining to non-existent boxes are ignored and the box-relative data groupings are returned in ascending box number order.

A monolithic system is considered as having one Global box. If a bit mask is passed to SYSTEMSTATUS for the appropriate cases, the data will be returned in a degenerate multi-box structure, as follows:

```
---------------------------------
: GS : 0&1[1:1] : GS : Box Info :
---------------------------------
                \_______________/
                        :
                     Global
                      Box
```

The information structure employed in this text appears as follows:

Item — Meaning

WORD — Group-Relative word index

If a SYSTEMSTATUS call of type 1, 2 or 7 is made with a box mask (V1.[29:5]), the word index is relative to the information for each box returned; otherwise, one major group is returned and the index is absolute.

FIELD — For those words which contain multiple fields, the starting (high-order) bit and field lengths are indicated; e.g., 10:5 implies that the field starts with bit 10 and extends for 5 bits.

TYPE — Data types may be indicated by a letter, as follows:

I Integer
M Bit-mask
R Floating-point number ("real")
S Text string

The type is blank for word structures which are subdivided, and for single-bit subdivisions.

-CAT — In call types 1, 2 and the first group of type 7, a category code is appended in the TYPE column to indicate the category of the data returned, as follows:

-CAT — MEANING

(blank) — These items may vary from one box to another and are reported appropriately for each box. If no bit mask is specified when calling SYSTEMSTATUS, the quantitative box relative information will be summed over all boxes and this sum returned as if the system had only one box (namely Global).

1 — These items, typically descriptive or parametric, apply to all boxes equally and are reported identically for each box. Examples of this category are Intrinsic name and option settings.

2 — Certain quantitative items are maintained only for the MCP as whole; e.g., MIXCOUNT. These items are reported with the Global box data only and zeroes will appear in their corresponding positions within the local box information.

>2 — These categories indicate that the data item does not fall into one of the other categories. Because these categories represent exception items, their interpretation will be explained as these items occur.

NAME & MEANING

The item identifier as declared in the MCP, accompanied with explanatory comments in many cases.

It should be noted that times and counts reported for the Global box refer to processes (stacks) residing in Global memory. There is no Global processor; each local processor spends part of its time serving Global needs.

In the following text, an effort has been made to minimize redundant verbage. The definitions referring to single bit fields apply only when the bit is on.

Unless otherwise noted, all time values are reported in "clicks" of 2.4 microseconds, and all times and counts accumulate values since Halt/Load.

MAINFRAME INFORMATION (Type 1)

This information group contains the "hardcore" information pertaining to the system, such as number of processors, amount of memory, name of the running MCP, etc.

The structures of information returned by SYSTEMSTATUS for a Type 1 call on both monolithic and tightly-coupled systems are as follows:

Without bit mask:

```
    ---------------------
    : GS : Group 1 Info :
    ---------------------
```

With bit mask:

```
    ------------------------------     ---------------------
    : GS : M : GS : Group 1 Info : ... : GS : Group 1 Info :
    ------------------------------     ---------------------
              \__________________/     \__________________/
                       :                        :
                Lowest Numbered          Highest Numbered
                     Box                       Box
```

Because of variable length items returned with this SYSTEMSTATUS call, the following convention has been adopted to indicate item displacement:

```
    X = 10 + [10]
    Y =  X + [10+[10]]
```

where [N] is defined to be the contents of word N.

Graphically, the above algorithm appears as follows:

```
                  _______________         ___________
                  :              :        :          :
                  :              V        :          V
      ------------------     ------------------     ----------
  ... :       MCP       : ... :   INTRINSIC     : ... : Memory :
      : name-size word  :     : name-size word  :     :  Info  :
      ------------------     ------------------     ----------
         (word 10)              (word X)            (word Y)
```

All items returned by the type-1 call are listed here.

WORD	FIELD	TYPE -CAT	NAME & MEANING
0		I	(Group-Size Word)
1		I	(System Serial Number)
2		M	PROCMASK Bit "n" indicates whether there is a processor "n".
3		M	MULTIPLEXORMASK Bit "n" indicates whether there is a multiplexor "n".
4		I-1	MAXUNIT The upper bound of the tables indexed by unit number.
5		M-3	MA This word contains a true (one) for each memory mod that is "in-use" by the MCP; bit 0 is memory mod 0, bit 31 is memory mod 31. (MB contains the bits for memory mods 32 through 63.) The contents of this word and the next 3 words will be zero if the system is tightly coupled and no bit mask was specified with the SYSTEMSTATUS call.
6		M-3	PMA The bits in this word correspond with the bits in MA above, except that a true (one) means that the memory mod is present (i.e., it was "found" by the last Halt/Load). It may or may not be "in-use" depending on MA.
7		M-3	MB This word has the same function as MA, but it reflects the "in-use" status of memory mods 32 through 63.
8		M-3	PMB This word has the same function as PMA, but it reflects the presence of memory mods 32 through 63.
9			MCPLEVEL
	47:16	I-1	MARKDIGIT MCP's "MARK LEVEL" in binary.
	31:16	I-1	LEVELNO MCP's "LEVEL NUMBER" in binary.
	15:16	I-1	CYCLENO MCP's "CYCLE NUMBER" in binary.
10		I	(MCP NAME-SIZE WORD) Length of the standard form MCP name.
11		S	(MCP NAME) Name of the running MCP in standard form.
X		I-1	(INTRINSIC name-size word) Length of standard form INTRINSIC name.
X+1		S-1	(INTRINSICS name) Name of running INTRINSICS in standard form.

WORD	FIELD	TYPE -CAT	NAME & MEANING
Y		I	(Total memory mods present and in-use)
	23:12		Total number of mods present
	11:12		Total number of mods in-use
Y+1	N:4	I-1	DCPHEYUMAP where N is the DCP number i.e., N=[DCP number]*6+5. Indicates multiplexor/processor to which each running DCP is physically connected. The value stored in the 4-bit field is the processor ID for B 6800s and the MPX ID for B 6700s.

OPERATIONAL GROUP (Type 2)

This information group contains the kind of information which an operator can affect as well as interrogate in order to make operational decisions.

The Operational Information structure as returned by SYSTEMSTATUS appears as follows:

Without bit mask:

```
 --------------------
 : GS : Group 2 Info :
 --------------------
```

With bit mask:

```
 -----------------------------      --------------------
 : GS : M : GS : Group 2 Info : ... : GS : Group 2 Info :
 -----------------------------      --------------------
           \_________________/       \_________________/
                    :                         :
                Lowest Box               Highest Box
                  Number                   Number
```

Of those items returned by the type-2 call, only those which have been changed or added since the original documentation are listed here.

WORD	FIELD	TYPE -CAT	NAME & MEANING
1		M	OPTIONS
			The bits in this word indicate the settings of the run-time options:
	0:1		(Not used)
	1:1		OPEN
	2:1		TERMNATE
	3:1		NOCHECK
	4:1		LPBDONLY
	5:1		AUTORM
	6:1		DIAGNOSTICS
	7:1		CDONLY
	8:1		AUTORECOVERY
	9:1		DUPSUPERVISOR
	10:1		DUPINTRINSICS
	11:1		(Not used)
	12:1		AUTODC
	13:1		NODUMP
	14:1		CPBDONLY
	15:1		(Not used)
	16:1		CRUNCH
	17:1		BACKUPBYJOBNR
	18:1		FULLTRANSLATION
	19:1		NOFETCH
	20:1		RESOURCECHECK
	21:1		NOSUMMARY
	22:1		DIRDEBUG
	23:1		CATALOGING
	24:1		OKTIMEANDDATE
	25:1		NEWPERETRY
	26:1		LOGPOSITIONING
	27:1		SERIALNUMBER
	28:1		ARCHIVING
	29:1		CONTROLOLDWFL
	31:1		IORANGECHECK
	43:1		IODIAGNOSTICS
	45:1		USECATDEFAULT
	46:1		CATTEST
	47:1		MCPTEST

WORD	FIELD	TYPE -CAT	NAME & MEANING
6		I-2	MIXCOUNT The number of jobs and tasks in the mix.
27		I	OLAYCHANNELS A decaying average of recent overlay I/O time, used in scheduling.
28		I	SCHEDCORE The amount of memory which is "scheduled"; i.e., the total number of words temporarily committed for recently initiated programs.
29		I	SUSPENDEES The number of tasks which are suspended.
30			(Contains a -1, previously IRJOBS)
31		I-2	NUMBEROFSWAPTASKS The number of swap tasks running on the system.
32		I	BATCHTASKS The number of batch tasks running on the system.
33		I-1	Memory Priority Factor The system's memory priority factor.
34		I-1	OLAYROWSIZE The system's overlay row size. The maximum overlayable item cannot exceed OLAYROWSIZE*30 words.
35		I	READYTIME The total time accumulated by tasks in the READY queue (since Halt/Load).
36		I-4	PROCNET This item and the next are used for scheduling in a B6800 multiprocessor (TC) system. PROCNET is calculated once per second for each local processor as the total since Halt/Load of three times: processor time spent in global tasks plus "true" idle time minus ready time. Zero is reported for the GLOBAL box.
37		R-4	RECENTNET This item is a decaying average of incremental values of PROCNET. It may be regarded as approximating the time a local processor has recently had available to spend on additional local tasks. Its units are clicks per second. RECENTNET may be "normalized" by multiplying it by 2.4@-6 (seconds/click); it will then range from about +1 for a completely idle processor to -n for a completely busy processor with n tasks always in the ready queue.
38		I	SCHEDPROC SCHEDPROC is used to bias the scheduling algorithm against successively placing many new tasks in the same box. It is incremented by 1/3 second (in clicks) for each new task, and decayed by .5 each second.
39		I	SWAPSPACESIZE The size of the swapspace in words.
40		I-3	HLUNITNO The Halt/Load unit number.

On a TC system, each subsystem may have its own Halt/Load units or a unit may be shared. When a boxmask is specified, the Halt/Load unit for each processor is reported for its local box and the Halt/Load unit for the "lead" processor is reported for the Global box. If no boxmask is supplied, the "lead" Halt/Load unit is returned.

GENERAL MIX INFORMATION (Type 3)

A general mix request returns a series of mix entries, each of which contains pertinent information about a task in the mix. The general format is mix entry, followed by another mix entry, followed by another mix entry, etc. Each entry has a size word to facilitate indexing to the next entry.

The structure of the information returned for the general mix call appears as follows:

```
 -----------------------------     ------------------------
 : GS : GS : General Mix Info : ... : GS : General Mix Info :
 -----------------------------     ------------------------
```

The first group-size word (GS) specifies the entire length of all the information returned. The second group-size word indicates the length of the first general mix entry. This is followed immediately by the general mix information. Subsequent group-size words and general mix information structures are repeated for all tasks running on the system.

Only those general-mix items which have been changed or added since the original documentation are listed here.

WORD	FIELD	TYPE	NAME & MEANING
3			SERIAL (previously STACKNO only)
	45:10	I	Stack Number (STACKNO)
	31:16	I	Job Number
	15:16	I	Task Number
5			SWAPSPEX and STACKSTATE (previously STACKSTATE only)
	47:1		WASSLICEDF
			The associated swaptask is being or has been forced out of the swapspace because of timeslice lapsed or need to enlarge the swapspace.
	46:1		EXPRESSF
			The associated swaptask presently resides in or will be swapped into the express portion of the swapspace.
	39:8	I	SLICENRF
			The number of times this swaptask has been "sliced" by SWAPPER. The term sliced means that the task used its portion of processor or elapsed time and may have been forcibly swapped out of the swapspace to give other swaptasks usage of the system's resources.
	31:5	I	LOCATIONF
			SWAPPER uses this field as a state variable to control the swaptasks.

Value	Name & Meaning
0	NEWV
	A new swaptask, not yet set up.
1	INCOREV
	Able to run.
2	QUIESCINGV
	Swapout requested; waiting because of IO in progress, dependent tasks in core, etc.
3	WRITEREQV
	Swapout is requested and pending SWAPPER attention.
4	WRITEV
	Swapout write is being considered or performed.
5	ONDISKV

WORD FIELD TYPE NAME & MEANING
---- ------ ---- ---- - -------

Task is on disk until demand (interactive).

6 READREQUV

Swapin is requested and pending SWAPPER attention.

7 NEEDCOREV

Needs central memory.

8 READINV

Swapin read being set up or performed.

9 DONEV

Swaptask is being terminated.

10 OUTOFDISKV

Needs swapdisk for swapout (exception case).

11 NEEDPARENTV

Waiting for antecedent task to be in core.

12 LINGERINGV

Swapout requested; waiting to see if task will be resumed shortly.

13 CONSIDERINGV

Special transition state.

23:4 I STACKSTATUSF (STACKSTATE)

There are three groupings of STACKSTATEs. The first group applies to stacks which cannot process. The second group is for process tasks which have no need for a processor at the moment. The third group is for tasks which presently have a processor or are waiting for an available processor.

Group 1 -- Stacks which cannot process.

Value Name & Meaning

0 NOTUSED

No stack exists (SYSTEMSTATUS will never return this value).

1 BEINGSETUP

The task is being established by the INITIATE procedure of the MCP.

2 DYING (Formerly 8)

The task is in the process of termination.

3 SCHEDULED (Formerly 2)

The task is in the schedule queue.

4 SELECTED (Formerly 3)

The transition state between BEINGSETUP or SCHEDULED and task initiation.

5 UNEMPLOYED (Formerly 9)

This stack state only exists for segment dictionary stacks; the MCP never assigns a processor to a stack with this value.

Group 2 -- Tasks which have no need for a processor at the moment.

6 TOBECONTINUED (Formerly 7)

WORD FIELD TYPE NAME & MEANING
---- ----- ---- ---- - -------

The task is waiting to be resumed after execution of a "CALL" or "CONTINUE" statement.

7 FROZEN (New state)

This task is a library which has executed a "FREEZE" statement.

8 WAITING (Formerly 5)

The task is waiting (on an event, DM block, DM transaction, etc.).

9 HOLDING (Formerly 6)

The task is waiting for a software interrupt.

12 ASLEEP (New state)

The task is waiting according to a special MCP convention.

Group 3 -- Tasks which presently have a processor or are waiting for an available processor.

13 READYSWAPPED (New state)

The task is waiting for SWAPPER action (e.g., to be swapped in) before it can get a processor.

14 READYQUEUED (New state)

The task is in the ready queue, waiting for an available processor.

15 ALIVE (Formerly 4)

A processor is running this task.

15:8 I ORIGINALMTYPEF

The SUBSPACE value requested by the task:

Value	Meaning
0	do not run in subspace.
1	put data but not code in subspace.
2	put data in subspace; put code in subspace if code file is in my directory.
3	put both data and code in subspace.

7:8 I ACTUALMTYPEF

The actual SUBSPACE assignment made when the task was executed. The values are the same as ORIGINALMTYPEF, except that 2 will have been resolved to 1 or 3.

7 USAGE

47:1 STKISMCSF

Indicates that the stack is an MCS.

46:7 I STKMCSNRF

The MCS number if the stack is or has been an MCS. If datacom is terminated, each active MCS has STKISMCSF reset but retains STKMCSNRF.

39:7 I QPRIORLIMTF

The priority limit obtained from the queue from which the task was initiated.

WORD	FIELD	TYPE	NAME & MEANING
	32:4	I	SHARINGSPEC Indicates sharing specifications for a library. Value Meaning 0 PRIVATELIB 1 SHAREDBYALL 2 DONTCARE
	28:1		PROGRAMLOCKEDF The task may not be DSed if this bit is on.
	27:1		GENDISABLEF All software interrupts for this stack are disabled.
	26:1		DOINGSOFTINTF The stack is processing a soft interrupt.
	25:2	I	PBITSTATEF Value Meaning 0 Not PRESENCEBITting 1 Initial PRESENCEBITting 2 Free PRESENCEBITting 3 Normal PRESENCEBITting
	23:1		(Not used, previously PERMANENTF)
	21:1		PROCSELECTEDF
	20:1		NOMEMF The associated stack has been stopped due to insufficient memory resources.
	19:1		DUMPINGF The stack is PROGRAMDUMPing.
	18:1		SORTTINGF The stack has called the sort intrinsic and sorting is in process.
	17:1		UNITATTACHEDF
	16:3	I	LIBSTATEF Value Meaning 1 LIBINITBYLINKER The library stack was initiated by the MCP's library linking routine. 2 LIBINITBYUSER The library stack was explicitly initiated; e.g., by a RUN command. 3 LIBFROZEN The normal state of a usable library stack. It is the only state in which programs may be linked to the library stack. 4 LIBTHAWING The library stack has at least one program linked to it, but no additional links are being accepted. 5 LIBRESUMED The library stack has no programs linked to it and no links are being accepted. 6 LIBNEVERFROZE

WORD	FIELD	TYPE	NAME & MEANING
			This state represents an error condition signifying that a library stack which has been initiated by the MCP's library linking routine never achieved a LIBFROZEN state.
	13:1		NOHISTORYF The MCP uses this bit as a temporary flag to prevent recursive calls of STACKHISTORY.
	11:2	I	BLOWBY Value Meaning 0 Normal (no stack overflow) 1 This is an independent runner; stack overflow is not permitted. 2 Stack overflow has occurred; stack stretch or termination pending. 3 Stack overflow has occurred and stack cannot be stretched; task is being terminated.
	8:8	I	BOXMASKF Bit "K" of USAGE is set if this stack is or can be in box "K".
	0:1	I	LOAFING GEORGE is operating out of this stack. This bit is used to prevent calling GEORGE a second time.

SPECIFIC MIX INFORMATION (Type 3)

When specific mix entries are requested, SYSTEMSTATUS will first load the front of the array row with the GENERAL MIX INFORMATION for that mix entry, and then append the SPECIFIC MIX INFORMATION (refer to GENERAL MIX INFORMATION section for the first portion of the returned data).

The structure of the data returned by the SPECIFIC MIX call appears as follows:

```
 ---------------------------------------------------------
 : GS : GS : General Mix Info : GS : Specific Mix Info :
 ---------------------------------------------------------
```

The first group-size (GS) word indicates the entire length of the information returned. The second group-size word specifies the length of the general mix information which is followed immediately by the general mix data. The third group-size word indicates the length of the specific mix information and is followed by the specific mix data.

Because of variable length items returned by this call, item displacement is defined as follows:

X = 1 + [1]

where [N] is defined to be the contents of word N.

Graphically the above algorithm appears as follows:

```
                        _____________________
                        :                   :
                        :                   V
    ---------------------------   ---------------------------
... : General entry size word : ... : Specific entry size word : ..
    ---------------------------   ---------------------------
            (word 1)                       (word X)
```

Only those specific-mix items which have been changed or added since the original documentation are listed here.

WORD	FIELD	TYPE	NAME & MEANING
X+2			COMPILERINFO This word contains information supplied by the compiler that created the code file for this task.
	47:1		IPCCAPABLEF
	46:1		SORTCAPABLEF
	45:1		CONTROLPROGRAMF
	44:1		DMSCAPABLEF
	43:1		NOBADPTRF (not used by MCP)
	42:1		CONTROLPROGRAMF
	41:1		LIBRARYCAPABLEF
	31:8	I	LANGUAGETYPEF

Value	Type
0	ALGOL
1	COBOL
2	FORTRAN
3	XALGOL
4	PL1
6	NEWP
7	ESPOL
8	DCALGOL
9	BASIC
10	WFL
11	VFORTRAN
12	VLINKERTYPE
13	SFORTRANTYPE

WORD	FIELD	TYPE	NAME & MEANING
			254 INTRINSICSLTYPE
	23:8	I	MARKLEVELF (e.g. 31)
	9:10	I	CYCLENUMBERF (e.g. 240)
X+15			STACKINFO
			This word is used by the MCP in performing necessary work for this task.
			NOTE
			This word has been completely reorganized since the II.9 release. SWAPPER information previously kept here is now located in word 5 (SWAPSPEX and STACKSTATE) of General Mix Information (Type 3).
	47:4	I	BOXNOF
			This field identifies the memory subsystem in which this stack resides.
			0 = None assigned (task scheduled or swapped out). 1 = Global memory n+1 = Local memory for processor "n" where n=1,2,3,4.
	43:1		URSVPWAITINGF
			The task is waiting on a unit RSVP.
	40:1		VISIBLEF
			The task is to be displayed and logged.
	39:1		CANTSUSPENDF
			The task is not to be stopped by the working-set or SUSPENDER mechanism.
	38:1		SUSPF
			The task has been suspended.
	35:1		PRIVILEGEDUSERF
			The task is "privileged".
	34:1		COMPILERF
			The task is a compiler.
	33:1		ACTIVEDBSF
			The stack is an "active" DBS stack.
	31:8	M	STACKKINDF: the next eight bits
	31:1		MCPSTACKF
			The MCP (D0) segment dictionary stack.
	30:1		DBSF
			A data-base segment dictionary stack.
	29:1		INTRINSICSF
			An intrinsic segment dictionary stack.
	28:1		SEGDICTSTACKF
			A segment dictionary (D1) stack.
	27:1		LIBRARYSTACKF
			A library stack.

WORD	FIELD	TYPE	NAME & MEANING
	26:1		JOBSTACKF A job stack.
	25:1		IRINDICATORF An MCP independent runner.
	24:1		TASKSTACKF A task (processing) stack.
	22:1		VIEWABLEF The task variable is stable and can be examined. (This bit is always on for tasks reported by SYSTEMSTATUS.)
	21:1		LOGGABLEF Logging to the task's JOBLOG is allowed.
	20:1		STACKPRESENTF The stack has been built and is in memory.
	7:7	M	PROCF Contains a processor mask: bit "i" of the STACKINFO word is on if processor "i" can run in this stack.
	0:1		SWAPJOBF The task is a swap task.
X+25		R	STOPPOINT (previously ABORTEDRCW) This word contains an RCW, with tag of zero, pointing to the location where a fault occurred or where the task was abnormally terminated. The 47:8 field will contain the fault REASON if a hardware fault has occurred; otherwise, it will contain zero.
X+27			PATHCONTROL
	45:6	I	DESTMCSF Contains an MCS number (usually RJE).
	39:1		DESTISREMOTEF Output is to be sent to a remote station (usually RJE).
	38:15	I	DESTUNITF If this field is empty, output is directed to the SITE printers; otherwise, output will be sent to the remote station designated by this field. This field is usually associated with RJE.
	23:1		ORGWANTSSUMMARYF The job summary will be generated as a printer backup disk file (used by RJE).
	22:1		ORGWANTSMESSAGEF The MCP will send messages to an associated MCS. This feature is used by RJE and by CANDE when "SO MESSAGES" has been specified.
	21:6	I	ORGMCSF Contains the originating MCS number, or 0.
	15:1		ORGISREMOTEF The task was originated from a remote station.

WORD	FIELD	TYPE	NAME & MEANING
	14:15	I	ORGUNITF Contains the originating LSN if ORGISREMOTEF is on; otherwise, it is the originating peripheral unit number.
X+28			STATIONINFO
	33:2	I	TASKTANKF Contains the value of the TANKING attribute which controls output tanking for REMOTE files. 0 - Not specified 1 - No tanking 2 - Synchronous tanking 3 - Asynchronous tanking
	15:16	I	STATIONF Contains the value of the STATION attribute.
X+29			OPTION This word contains the value of the task's OPTION attribute.
	20:1		PRIVLIBDUMPF Causes a dump of any private LIBRARY connected to this task.
	19:1		LIBRARYDUMPF Produces a dump for any shared or private LIBRARY connected to this task.
	18:1		SIBDUMPF Causes the Structure Information Block (SIB), not the Data Base Stack (DBS), to be printed by PROGRAMDUMP.
	17:1		IPCOVERRIDEF Pre-III.0 IPC-capable code files will be run in subspace only if this option is set.
	16:1		PDSDUMPF Causes a dump of all ports and signals used by the task.
	15:1		DBSDUMPF Causes the Data Base Stack (DBS) and Structure Information Block (SIB) to be printed by PROGRAMDUMP.
	14:1		PRIVATETASKF Prevents parent tasks from inadvertently being terminated by their offspring tasks.
	13:1		(Not used)
	12:1		NOSUMMARYF A job which has normally terminated and has no associated printer backup files will not print a job summary.
	11:1		BACKUPPUNCHCLSF All punch files will go to backup-disk if this option is set.
	10:1		FILESCLSF Causes the File Information Block for each file to be printed by PROGRAMDUMP.
	9:1		CODECLSF Causes the code, segment dictionary, and value arrays to be printed by PROGRAMDUMP.

WORD	FIELD	TYPE	NAME & MEANING
	8:1		ARRAYSCLSF Causes arrays to be printed by PROGRAMDUMP.
	7:1		BASECLSF Causes PROGRAMDUMP to print the base of the stack and the Program Information Block (TASK variable).
	6:1		BDBASECLSF Used to establish a new BD BASE for backup files.
	5:1		AUTORMCLSF Automatic duplicate file removal option.
	4:1		BACKUPPRINTCLSF All printer files to backup-disk.
	3:1		(Not used)
	2:1		DSEDCLSF Program dump on external termination.
	1:1		FAULTCLSF Program dump on fault termination.
	0:1		LONGCLSF No arrays will be segmented.

PERIPHERAL INFORMATION (Type 4)

A SYSTEMSTATUS call of Type 4 with a V2 value greater than 255 will cause the system's UNITTABLE to be returned. The structure of the data returned by this call is:

```
-----------------------
: GS : UNITTABLE Info :
-----------------------
```

The following changes have occurred in the UNITTABLE field definitions:

WORD	FIELD	TYPE	NAME & MEANING
N			UNIT[N-1]
	41:1		UHLPACK The corresponding unit is a Halt/Load unit. (There may be more than one Halt/Load unit on a TC system.)
	41:6	I	UTRAINID If the unit is a train printer, this field contains the train identification.
	40:1		UNEWFIRM The pack firmware is 2.0 or later.
	38:1		UALTQR The HPT disk unit is an alternate DFO.
	37:1		UDECIMALF The unit requires decimal addressing.
	36:1		UPBUSYABLE
	17:3	I	DENSITYF If the UNITTYPE is DISK 0 = IA-2 1 = IC-3 2 = 5N 3 = IC-4 4 = IIB-2 5 = IIB-6 6 = IIB-4 If the UNITTYPE is PACK 0 = 215 1 = N215 2 = 225 3 = 235 4 = 206 sequential 5 = 206 interlaced 6 = 207 If the UNITTYPE is TAPE 0 = 800 bpi NRZ 1 = 556 bpi NRZ 2 = 200 bpi NRZ 3 = 1600 bpi PE 4 = 6250 bpi GCR

WORD	FIELD	TYPE	NAME & MEANING
	16:2	I	ULPKIND This field designates the train printer kind: 0 = 450/750 lines per minute 1 = 1100 lines per minute

TIMES AND COUNTS, DCP, IOTRACE, and PROCESSOR INFORMATION (Type 7)

The Type 7 SYSTEMSTATUS call is used to obtain information pertaining to miscellaneous MCP times and counts, DCP and I/O activity, and processor time distributions among visible and non-visible tasks.

The structures of information returned by the Type 7 call are:

Without bit mask:

```
 -------------------------
 : GS : GS : T & C  Info : --->
 -------------------------

      --------------------------------------
      : GS : DCP Info : GS : IOTRACE Info : --->
      --------------------------------------

           ------------------------------
           : GS : Processor Information :
           ------------------------------
```

With bit mask:

```
 --------------------------     -----------------
 : GS : M : GS : T&C Info : ... : GS : T&C Info : -->
 --------------------------     -----------------
           \______________/       \______________/
                  :                      :
            Lowest Box             Highest Box
              Number                 Number

      --------------------------------------
      : GS : DCP Info : GS : IOTRACE Info : --->
      --------------------------------------

           ------------------------------
           : GS : Processor Information :
           ------------------------------
```

The features of the above structures include the following:

First: A group-size word (GS) reflects the total amount of information returned.

Second: If a bit mask is used when calling SYSTEMSTATUS, a bit mask word (M) will be returned indicating the boxes for which information has been collected.

Third: The times and counts information and their associated group-size word is returned. Note that the times and counts information for each box has its own group-size word.

Fourth: The DCP information is as previously documented.

Fifth: The IOTRACE information and its associated group-size word is included provided that the IOTRACE mechanism has been previously activated by the Type 8 call. Note that the IOTRACE and DCP substructures are the same for both monolithic and tightly-coupled systems.

Sixth: The final entry is new information describing the distribution of processor times by Global/Local and visible/non-visible time delivered to users.

All items returned by the case-7 call are listed in this note, which incorporates D-notes from the II.8 and II.9 MCP releases.

Times and Counts Information:

Within this section, additional category codes are defined as follows:

Value	Meaning
4	On a TC system, these processor-related items are reported for local boxes only, the GLOBALBOX value being zero.
5	On a TC system, zero is reported as these items are not accumulated.

WORD*	FIELD	TYPE -CAT	NAME & MEANING
0			Group-size word
1		I-2	I/O time charged to the MCP stack.
2		I-4	Processor time spent IDLE.
3		I-4	Number of times a processor became IDLE.
4		I-4	Processor time spent non-idle in GEORGE.
5		I-4	Number of times a processor entered GEORGE or became non-idle.
6		I-4	Processor time spent in ANSWER.
7		I-4	Number of times ANSWER was entered.
8		I-4	Processor time spent in PRESENCEBIT.
9		I-4	Number of times PRESENCEBIT was entered.
10		I-2	Processor time logged for completed visible tasks.
11		I-2	I/O time logged for completed visible tasks.
12		I-2	Processor time accumulated by completed non-visible tasks and by completed visible tasks after they logged off.
13		I-2	I/O time accumulated by completed non-visible tasks and by completed visible tasks after they logged off.
14		I-2	Processor time spent in SEARCH.
15		I-2	Number of times SEARCH was entered.
16		I-2	Processor time in STACKSEARCH.
17		I-2	Number of stacks searched in STACKSEARCH.
18		I-2	Undefined
19		I-2	Undefined
20		I-4	TRUEIDLETIME: Time spent idle because of a lack of tasks.
21		I-4	TRUEIDLECOUNT: Number of times TRUEIDLETIME was incremented.
22		I-4	FALSEIDLETIME: Time spent idle when some stack is PBITting.
23		I-4	FALSEIDLECOUNT: Number of times FALSEIDLETIME was incremented.
24		I	BATCHPBITTIME: Process time spent handling normal PBIT for tasks in batch space.
25		I	BATCHPBITCOUNT: Number of normal PBITs handled for tasks in batch space.
26		I-5	BATCHSEARCHTIME: Processor time spent by other processors in ANSWER while a processor was doing normal PBIT for a task in batch space.

WORD*	FIELD	TYPE -CAT	NAME & MEANING
27		I-5	BATCHSEARCHCOUNT: Number of times BATCHSEARCHTIME was incremented.
28		I	INITPBITTIME: Processor time spent handling PBIT to make a data or code segment present for the first time.
29		I	INITPBITCOUNT: Number of initial PBITs handled.
30		I-5	INITSEARCHTIME: Processor time spent by other processors in ANSWER while a processor was doing initial PBIT.
31		I-5	INITSEARCHCOUNT: Number of times INITSEARCHTIME was incremented.
32		I	FREEPBITTIME: Processor time spent handling PBIT to restore the workingset of a previously suspended, recently resumed (batch) task.
33		I	FREEPBITCOUNT: Number of free PBITS handled.
34		I-5	FREESEARCHTIME: Processor time spent by other processors in ANSWER while a processor was doing free PBIT.
35		I-5	FREESEARCHCOUNT: Number of times FREESEARCHTIME was incremented.
36		I	SWAPPBITTIME: Processor time spent handling normal PBIT for a task in swap space.
37		I	SWAPPBITCOUNT: Number of normal PBITs handled for swap tasks.
38		I-5	SWAPSEARCHTIME: Processor time spent by other processors in ANSWER while a processor was doing normal PBIT for a swap task.
39		I-5	SWAPSEARCHCOUNT: Number of times SWAPSEARCHTIME was incremented.

* Word index is relative to start of group

DCP Information

The DCP information is as documented in II.8 MCP note D1781, incorporated here.

If the datacom subsystem is not running (the tables are not in storage), the group is null (its group-size word contains 1). If the datacom subsystem is running, there are four words for each DCP number from 0 through MAXDCPS, arranged in the following order:

0 Sum of character counts in WRITE and READ-ONCE-ONLY messages.
1 Count of all messages sent to the DCP.
2 Sum of character counts in enable-input and read-once messages.
3 Count of all messages received from the DCP.

The first pair of words pertains to output from DCWRITE to the DCP; the second pertains to input from the DCP to DCCONTROL. Note that the character counts in the messages do not necessarily reflect the information actually transmitted/received by the DCP.

IOTRACE Information

The IOTRACE information is as documented in II.8 MCP note D1781, incorporated here, with changes noted for the former USER word.

The first three words in the IOTRACE group are fixed, as follows:

WORD*	FIELD	TYPE	NAME & MEANING
0			Group-size word (3 if IO trace is inactive)

WORD*	FIELD	TYPE	NAME & MEANING
1			IOTRACECOUNT total IO finishes since trace activated
2	47:8		Number of words in an IOTRACE block
	39:20		Number of double-words in IOTRACE array
	19:20		N=number of IOTRACE blocks in this group

Then follow N IOTRACE blocks, describing the N most recent I/O operations completed. Each block comprises six words, arranged as follows:

0 TIMECELL of I/O Control Block (IOCB)
1 CLOCK value at I/O finish: [0]+[1] is I/O time.
2 IOSTUFF word of IOCB & MPX ID [19:4] (see below)
3 I/O (hardware) result descriptor
4 I/O area descriptor
5 I/O control word

The I/O control word, area descriptor and result descriptor are documented in the hardware handbook and elsewhere. The IOSTUFF word layout is defined in the MCP and is subject to change from release to release; it was formerly called the USER word. For the III.1 release, the IOSTUFF word has assigned a new field for the owner's stack number. The name of the field is now OWNERSTKNOF (previously IDNOF) and has been moved from 47:10 to 45:10. The MPX identification field (MPXIDF - 19:4) and unit number field (UNITNUMBERF - 31:8) are still in the same locations as in previous releases.

Activation and Use of IOTRACE

The IOTRACE facility is controlled by a Type 8 call on SYSTEMSTATUS, as documented previously in II.8 MCP note D1781. The array parameter is not used in this call. If the third parameter is zero or negative, the feature is deactivated (if active). Otherwise the third parameter specifies the number of IOTRACE blocks the MCP should allocate (the number of I/O finishes to be remembered). If sufficient storage is unavailable, IOTRACE is deactivated and SYSTEMSTATUS returns an error flag containing 37. If the storage is available, IOTRACE is activated and IOTRACECOUNT is set to zero. The upper bound for IOTRACE allocation is 2500 blocks; larger values will be truncated.

The number N of IOTRACE blocks returned in a Type 7 call is the minimum of:

> The number of IOTRACE blocks allocated by the MCP (max 2500),
> the capacity of the user's array parameter, and
> the difference between IOTRACECOUNT and the third (V2)
> parameter of the Type 7 call.

Thus if the sampling program passes IOTRACECOUNT from one Type 7 call as the V2 parameter in the next call, it will be given all the I/O-finish information since the previous call, unless too many I/O operations have been finished in the meanwhile, or unless his array is too short. The program can detect loss of data by comparing the incremental value of IOTRACECOUNT with the number N of blocks returned. The IOTRACE block allocation and the sampling rate may be varied to achieve the desired amount of retained/lost information. The user may pass a very large positive number as V2, in which case no IOTRACE blocks will be returned, since IOTRACECOUNT-V2 will always be negative.

Processor Information:

This group returns information about each processor on the system, for both monolithic and TC systems. For TC systems, information is returned for both Global- and Local-task utilization.

These accumulators record processor time allocated to tasks, which is all the time not accumulated as GEORGE, IDLE, PRESENCEBIT or ANSWER time. Therefore all these accumulators should add to the total elapsed time for all the processors on the system. (Since there are timing windows between updating the accumulators and sampling with SYSTEMSTATUS, there will be small discrepancies in the sums reported.) Note that if the PRESENCEBITCHARGED and OVERHEADCHARGED $-options are SET in an MCP compilation, then the PRESENCEBIT and ANSWER times, respectively, are included in the task processor times as well as in their separate accumulators.

The processor information is arranged in the following order.

Group-size word for all processor information

Processor Mask
Bit "i" is 1 if processor "i" is in use on the system. In the following subgroups, data are reported for each processor in increasing order of processor id number. Note that "i" is a processer ID, not a box number.

Group-size word for global processing
For each processor, there follow two words: the time spent by that processor in non-visible global tasks, followed by the time spent in visible global tasks. These data are present for all systems.

Group-size word for local processing

This word and the following are present only for a tightly-coupled system. For each processor, there follow two words: the time spent in non-visible local tasks, followed by the time spent in visible local tasks.

"Visible" means displayed and logged. "Non-visible" tasks include some independent runners, and tasks that are terminating and have already started to log off.

MIX AND TASK DATA (Type 9)

The structure and protocol for Type 9 are unchanged from that previously documented in II.8 MCP note D1781; that note is incorporated here. The data returned are subject to all the changes noted above for Type 3.

The Type 9 SYSTEMSTATUS calls retrieves general and specific mix data more efficiently and more conveniently than possible with type 3. It has the following advantages:

Data are returned about non-visible as well as visible stacks.

Fewer calls are required to get a complete mix survey.

A convenient fixed array size suffices for any mix.

The general group for each task is returned only once, not twice.

The data returned about each stack are the same general and specific groups defined for type 3, but the groups are arranged differently.

SYSTEMSTATUS scans all stacks in the system beginning at the stack number specified in the third (V2) parameter. It returns data for all process stacks until the maximum stack number is reached or there is insufficient room in the array for another entry. In the latter case, the stack number for a continuation call is returned in the array. V2 should be zero for the first call in a given sample.

The structure of the array is:

Word 0: Total number of words returned.
Word 1: Next stack number (for continuation-call third parameter), or zero (if no continuation required).

Then follow data about as many stacks as required or will fit: each stack is represented by a general group and a specific group, which contain the same data as returned by type 3. The specific group always follows the general group for the same stack; continuation breaks occur between stacks, not between the two groups for one stack.

The combined data about each stack vary in length, depending upon the length of task and compiler names, usercodes, etc. Typically, about 45 to 50 words suffice. To save time, SYSTEMSTATUS estimates whether another entry will fit before processing the stack in question; if an entry is unexpectedly long and does not fit, the partial entry is discarded and prior whole entries are returned. A fault error will be reported by SYSTEMSTATUS only if a single entry to be returned will not fit (or otherwise causes a fault).

Information is returned for all process stacks in a stable ("viewable") state, including both scheduled and active tasks; stacks in transition cannot be shown. In addition to visible stacks (those that appear in the CONTROLLER mix picture and in the SYSTEM/SUMLOG), the type 9 call shows MCP independent runners. Visible stacks are marked by a 1 in STACKINFO[snr].VISIBLEF; this is seen in the III.1 MCP as bit 40 of word 15 in the task group (relative to word 0, the group length).

SWAPPER STATISTICS (Type 10)

The Type 10 SYSTEMSTATUS call is used to obtain information pertaining to SWAPPER and swaptasks. It was previously documented in II.9 MCP note D2145, which is incorporated here with appropriate changes. The principal differences are in the LOCATION values, the dimensions of the SWAPSTAT array (SWAPCOUNTSIZE), and some field layouts in the first of the 3 words returned for each swaptask.

The displacement convention used to get to the swaptask information is as follows:

```
                 --------------------
                 :                  V
       ------------             -------------
       :Length of :             : Swaptask  :
       : SWAPSTAT :             :information:
       ------------             -------------
        (word 3)                  (word X)
```

WORD	FIELD	TYPE -CAT	NAME & MEANING
0		I	Group size word
1		I	The value of CLOCK when this call was made
2		I	The number of swaptasks currently in the system.
3		I	Length+1 of the SWAPSTAT array, or 1 if SWAPPER has not been used since Halt/Load.
4		I	SWAPSTAT array

The SWAPSTAT array contains a vector of 18 time accumulators, four miscellaneous accumulators, and a matrix of 18x18 transition counters. The vector and matrix are indexed by a composite SWAPPER state:

STATE	NAME & MEANING
0	NEWV A new swaptask, not yet set up.
1	INCOREV Able to run (not EXPRESS).
2	QUIESCINGV Swapout requested; waiting because of I/O in progress, dependent tasks in core, etc.
3	WRITEREQV Swapout is requested; pending SWAPPER attention.
4	WRITEV Swapout write is being considered or performed.
5	ONDISKV Task is on disk until demand (interactive).
6	READREQUV Swapin is requested; pending SWAPPER attention.
7	NEEDCOREV Needs central memory (not EXPRESS nor SLICED).
8	READINV Swapin read being set up or performed.
9	DONEV Swaptask is being terminated.
10	OUTOFDISKV Needs swapdisk for swapout (exception case).
11	NEEDPARENTV Waiting for antecedent task to be in core.
12	LINGERINGV Swapout requested; waiting to see if task will be resumed shortly.
13	CONSIDERINGV Special transition state.
14	INCOREEXPX Able to run as an EXPRESS task.
15	NEEDCOREEXPX Needs memory as an EXPRESS task.
16	NEEDCORESLIX Needs memory after being SLICED out.
17	NEEDCORESLIEXPX

STATE	NAME & MEANING
	Needs memory as an EXPRESS task SLICED to grow.

The indices 0 through 13 are the LOCATION values that control SWAPPER as a state processor; indices 14 through 17 are derived states.

The four miscellaneous accumulators in SWAPSTATE have defined indices 18 through 21:

WORD	FIELD	TYPE	NAME & MEANING
4+18		I	SWAPTIMESX Elapsed time accumulator for all swaptasks
4+19		I	SWAPQCOUNTX Number of messages removed from SWAPQ
4+20		I	SWAPREADX Number of SWAPPER reads of subspaces
4+21		I	SWAPWRITEX Number of SWAPPER writes of subspaces
4+22		I	Transition count matrix

Let A be the array parameter to SYSTEMSTATUS. Then, the following is the amount of time swaptasks have spent in state I, not including the current contribution of any task now in that state:

```
A[4+I]
```

The following is the number of transitions from state I to state J:

```
A[4+22+I+J*18]
```

The following is the total elapsed time of all swaptasks, in clocks:

```
A[4+18]+A[2]*A[1]
```

After the transition matrix follows a sequence of 3-word entries, one for each swaptask in the mix. For K=0,1...,A[2]-1, let X=3+A[3]+3*K. Then the data returned are:

WORD	FIELD	TYPE	NAME & MEANING
X	47:1		Task was time sliced the last time it was swapped out.
	46:1		Task is an EXPRESS task.
	45:1		D1 stack and code are in the subspace.
	31:5	I	LOCATION (state) index.
	23:4	I	Box number of swap task.
	15:16	I	Mix number of the swap task.
X+1	47:8	I	Contains the number of additional slots needed.
	39:10	I	Contains the current length of the subspace in slots.
	29:10	I	Contains the slot index+1 where the subspace begins in the swapspace, or zero if no memory is allocated for this task.
	19:20	I	Contains the swapdisk address for the subspace, or zero if no disk is allocated for this task. [19:10] is the disk row index. [9:10] is the slot index within that row (Disk slot 0,0 is never allocated; it holds the swapper parameters.)
X+2	47:5	I	Contains the composite state index (0 through 17) of the state last entered by this swaptask.
	42:43	I	Contains the time (CLOCK value) when the task entered that state.

The swaptask is in state I=A[X+2].[47:5]. The length of time it has been in this state is A[1]-A[X+2].[42:43]. This value may be added to A[4+I] to accumulate the total time swaptasks have spent in this state.

The organization of the SYSTEMSTATUS array is i'lustrated by the figure that follows.

S=18 is the number of composite swapstates.

T=S+4 is the number of accumulators before the transition matrix.

```
|-------|
|   0   | length of array = M
|-------|
|   1   | clock
|-------|
|   2   | number of swaptasks
|-------|
|   3   | length+1 of swapstat array = N (= S*(S+1)+5), or 1 if none
|-------|                  ----------------|-------------------------|
|  0+4  | time in state 0                  |                         |
|-------|                                  |                         |
|  1+4  | time in state 1                  |                         |
|-------|                                  |   time info             |
    .                                      |                         |
    .                                      |                         |
    .                                      |                         |  S
|-------|                                  |                         |  W
| S-1+4 | time in state S-1                |                         |  A
|-------|                  ----------------                          |  P
| 18+4  | total time in swap tasks - A[1]*A[2]                       |  S
|-------|                                                            |  T
| 19+4  | number of swapq messages                                   |  A
|-------|                                                            |  T
| 20+4  | swapper reads                                              |
|-------|                                                            |  A
| 21+4  | swapper writes                                             |  R
|-------|                  ----------------|                         |  R
| T+0+4 | transitions from state 0         |                         |  A
|-------|                                  |                         |  Y
| T+1+4 | transitions from state 1         |                         |
|-------|                                  |                         |
    .                          .           | transitions to state 0  |
    .                          .           |                         |
    .                          .           |                         |
|-------|                                  |                         |
|T+S-1+4| transitions from state S-1       |                         |
|-------|                  ----------------|                         |
|T+S+0+4| transitions from state 0         |                         |
|-------|                                  |                         |
|T+S+1+4| transitions from state 1         | transitions to state 1  |
|-------|                                  |                         |
    .                          .           .                         |
    .                          .           .                         |
    .                          .           .                         |
    .                                      .                         |
    .                                      .                         |
    .                                      .                         |
|-------|                  ----------------|                         |
|N-1-S+4| transitions from state 0         |                         |
|-------|                                  |                         |
    .                          .           |                         |
    .                          .           | transitions to state S-1|
    .                          .           |                         |
|-------|                                  |                         |
|N-1-1+4| transitions from state S-1       |                         |
|-------|                  ----------------|-------------------------|
|  N+3  |                        |
|-------|                        |
| N+3+1 |                        |  info for 1st swap task
|-------|                        |
| N+3+2 |                   -----|
|-------|                        |
|  N+6  |                        |
|-------|                        |
| N+6+1 |                        |  info for 2nd swap task
|-------|                        |
    .                            .
    .                            .
    .                            .
|-------|                   -----|
|  M-3  |                        |
|-------|                        |
|  M-2  |                        |  info for last swap task
|-------|                        |
|  M-1  |                        |
|-------|                   -----|
```

SWAPPER Parameter Information (Type 11)

Type 11 is new to the III.1 release; it returns SWAPPER parameters.

The structure of the information returned by the Type 11 SYSTEMSTATUS call is:

```
------------------------------------     ---------------------------
: GS : M : GS : SWAPPER Parameters : ... : GS : SWAPPER Parameters :
------------------------------------     ---------------------------
          \_________________________/      \_________________________/
                       :                                :
                  Lowest Box                       Highest Box
                    Number                           Number
```

Type 11 is not affected by the bitmask in V1; it returns data from all boxes in which SWAPPER is active. M is a bitmask of these boxes.

WORD	FIELD	TYPE	NAME & MEANING
0			Group-size word
1		I	SWAPCOREMAX in words
2		R	MINTIMESLICE in seconds
3		I	MAXSLICENR
4		R	RATIO
5		I	MAXCORE in words
6		I	EXPRESERVE in words
7		I	EXPMAXCORE in words
8		I	PRIORITYBIAS 0-99
9		I	UTILIZATIONBIAS 0-99
10		R	EXPRESSMAXTIME in seconds
11		R	IOBIAS
12		R	MEMORYBIAS

SUSPENDER Information (Type 14)

A SYSTEMSTATUS call of Type 14 will return SUSPENDER information. It is new to the III.1 release, and is subject to change with the needs of SYSTEM/SUSPENDER. All these items are accessible through other SYSTEMSTATUS calls (types 7, 2 and 1).

GETSPACE Statistics (Case 15)

A temporary Case 15 has been added to accumulate statistics about GETSPACE usage. It will be removed in a subsequent release.