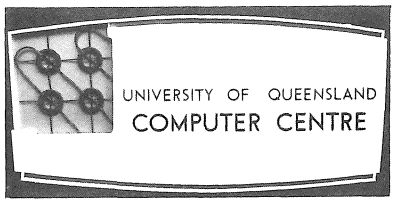


COMPUTER CENTRE BULLETIN

Vol. 5 No. 10
2 October 1972

Editor:
Sarah Barry

CHRISTMAS SHUT DOWN

The Computer Centre will shut down on 22 December 1972 and remain closed during the intervening days between Christmas and New Year. No further work will be processed after 11.00 p.m. on Thursday 21 December. However, the Centre will remain open until 1 p.m. on Friday 22 December to allow staff to finalize end of year accounting and users to collect any work or card punching. The Centre will reopen at 9 a.m. on Tuesday 2 January 1973.

EXTENSION OF HOURS OF OPERATIONS

On Monday October the Centre will introduce extended hours of timesharing operation on a temporary basis until Christmas.

The new schedule of operations will be as follows

0600-0800	:	system maintenance
0800-midnight	:	timesharing services
midnight-0600	:	third shift batch work, dedicated mode, end of day accounting and file backup. (The Centre will be closed during this shift).

This extension has been made possible by the Centre staff volunteering to work earlier hours during the summer months.

During this period a revised schedule of operations for 1973 will be announced.

AUThORIZATION TO USE REMOTE TERMINALS

[WN-104]

In the past, any user authorized to work on a remote terminal (other than the public terminals) could access his project from any terminal connected to the system. This has created some

problems.

In order to provide departments and organizations with adequate control over the use of their own terminals and still allow users the same flexibility of access they currently enjoy, the Centre will introduce the following changes.

(a) Submission of a completed 'Authority to Use Remote Terminal' form establishes the nominated project as a remote terminal project with a logged out file storage limit of 37.5 K words. Only 15 such projects per terminal can be established. Access will initially be restricted to the terminal for which authorization is given.

(b) If a user wishes to access his remote terminal project(s) from another terminal, he should complete an 'Authority to use Alternate Terminal' form (obtainable from the Centre), have it signed by the head of the department or organization whose terminal he wishes to use, and return it to the Centre.

This merely allows use of the terminal. It does not include the project as one of the 15 originating from the terminal.

These changes were implemented as from Monday 18 September. From that date the system allowed access only in respect of authorizations held by the Centre.

It should be noted that departments that allow their terminal to be used by others are entitled to collect an additional 60 cents per hour connect time ($1.20 per hour for external organizations) direct from the user to help defray maintenance charges.

MULTIPLE SIGNATORIES FOR PROJECTS

In response to a number of requests, the Centre's policy of allowing only one signature to authorize batch runs on a given project has been relaxed. Either of two signatures per project will now be allowed.

The second signature for any project should be registered with the Centre at the Enquiries counter.

EXTENDED COMMAND SET

[WN-103]

On Wednesday 9 August a new version of the monitor was implemented on the PDP-10 system. As well as correcting a number of errors and implementing some additional internal facilities (e.g. expanded hardware error reporting), this monitor provides an extended command set. These new commands are an extension of the existing command set - all previous commands continue to work as they previously did.

The following description outlines the new facilities and commands now available on the system.

1 PROGRAM FILES

In the PDP-10 system a program file can exist in any one (or several) of the five different states described below.

(a) Source Language File

This is a source language program (e.g. Fortran, Cobol, Macro, Algol etc) that exists as a text file, generally on disk or on punched cards.

(b) Relocatable File

A relocatable binary program file is produced on disk as the result of a compilation of a source language file by the appropriate processor. Relocatable files can be complete programs, individual routines or subroutines, or libraries of subroutines.

(c) Absolute File

An absolute binary (or core image) program file is the result of loading one or more relocatable binary files with the linking loader and then transferring a copy of the incore program to disk.

The loader loads a program together with its required subprograms into core store, relocating each one, adjusting addresses as required, and resolving linkages between independently compiled subprograms. This produces, in memory, the required program ready to commence execution. The absolute file is simply a copy on disk of the program as it finally resides in core ready to run (hence 'core image' file).

(d) Incore (dormant) File

This is a copy of the absolute program file in core but not executing. It may be ready to commence execution, or may have ceased execution due to some external interrupt.

(e) File in Execution

This is a program executing in core.

Figure 1 shows the transitions that can be made between these different states, and indicates the commands required for each transition. It should be noted that it is not yet possible to go from relocatable binary files directly to absolute files. To generate an absolute file two steps are involved.

(i) From relocatable files use the Load command to produce an incore dormant program file

(ii) Produce a copy of the incore file onto disk with the Save command.

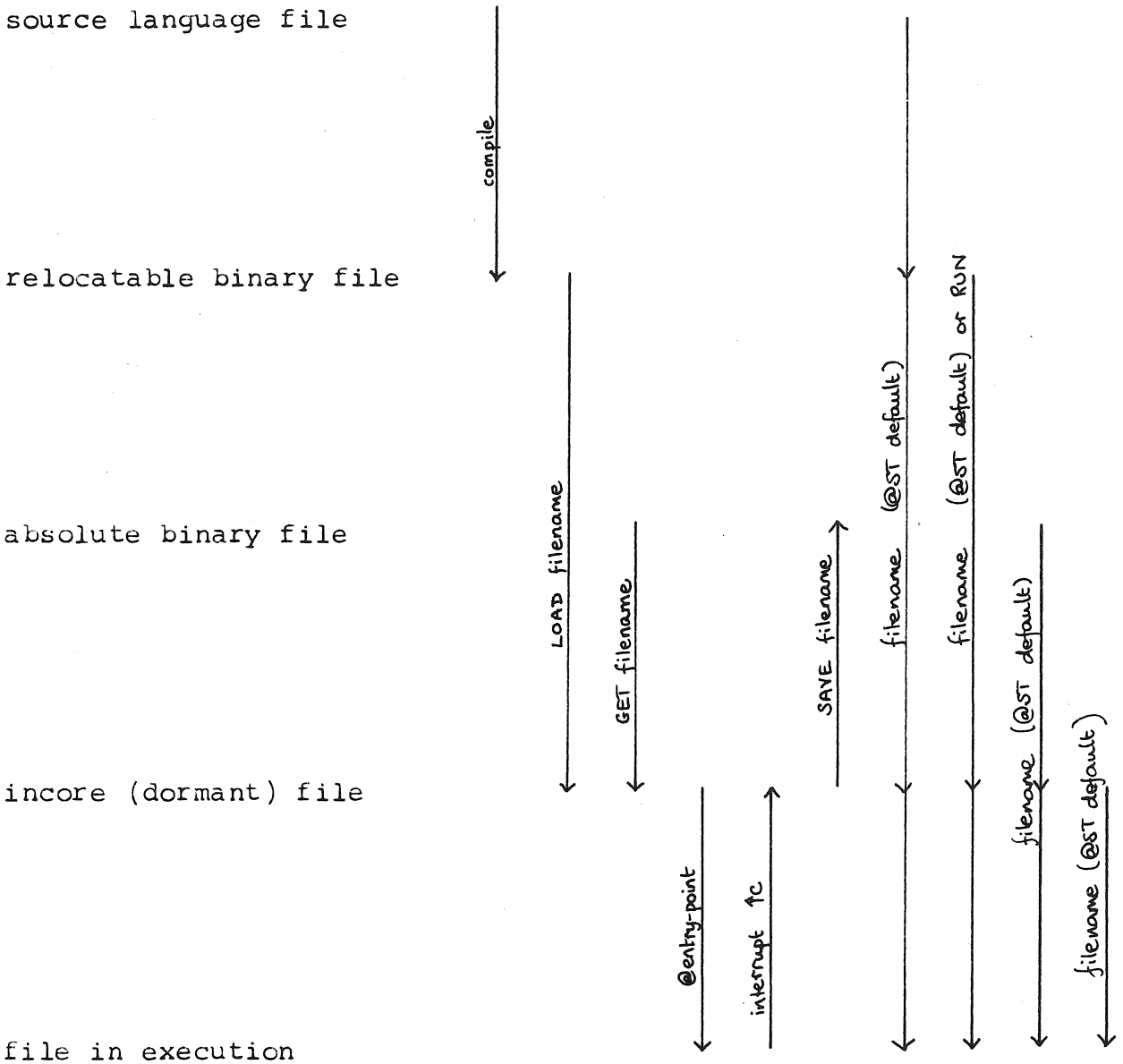

source language file
relocatable binary file
absolute binary file
incore (dormant) file
file in execution
compile
LOAD filename
GET filename
@entry-point
interrupt ↑C
SAVE filename
filename (@ST default)
filename (@ST default) or RUN
filename (@ST default)
filename (@ST default)

Figure 1

2 RUN CONTROL COMMANDS

2.1 LOAD

```
        MAP
LOAD(NOMAP,DDT) filename-1(LIB), ..., filename-n(LIB)
        SYMBOL
```

Load performs the same functions as the Run command except that the files loaded are not actually executed. That is, it loads the relocatable binary file(s) named in the command (or from the loadlist) and places them in core ready for execution.

2.2 SAVE

```
SAVE(S) filename {core}
```

The Save command writes out an image of the user's current core area onto the disk with the filename given. This core image file can then be reloaded quickly into memory at any later time with a Get command.

There are in fact three types of user programs with corresponding core image files. These are: one segment non-reentrant progams, two segment non-reentrant programs, and reentrant programs.

The last two types are discussed in section 5 on 'Segments and Reentrant Programs'. The one segment non-reentrant program is by far the most generally used type of program. When saved, it results in a single core image file on disk with the filename given and the processor program name of /SAV.

The core argument is the amount of core in which the program is to be run, specified in 1K blocks. If omitted the amount of core actually required by the program is assumed.

The S option is used for saving reentrant programs as explained in section 5.

2.3 GET

```
GET filename {core}
```

Get retrieves an absolute binary file from the disk, placing it in core ready for execution.

filename is the name of the file. core is the amount of core to be assigned if this is different from the minimum core needed to load the program, or from the core argument of the Save command that saved the file.

2.4 @entry-point

```
ST
RES
REE
@PC
DDT
CST
CPC
octal-address
```

This set of commands will begin execution of the incore program at a number of different entry points. Many of the entry points are locations discussed in the job data area (see section 5).

2.4.1 @ST

Begins execution of a previously loaded program at its normal starting address.

2.4.2 @RES

@RES starts the incore program executing at the normal start address+1. Many of the system programs use this facility when their arguments have been decoded.

2.4.3 @REE

The @REE command causes execution of the program to commence (or recommence) at an alternate entry point called the 'reentry address' which must have been specified by the user or his program.

2.4.4 @PC

@PC continues the program execution at the saved program counter address stored by a ↑C instruction.

2.4.5 @DDT

For programs loaded with DDT, this command commences execution of the program at the DDT start address. This allows the user to give DDT commands and follow the execution of the program (see the DDT manual MNT-12).

2.4.6 @CST

Same as @ST except that the user's console is left in monitor mode.

2.4.7 @CPC

Same as @PC except that the user's console is left in monitor mode.

Further monitor commands can be entered from the console after @CPC or @CST have been given. Note that these commands should not be used when the user program (which is continuing to run) is also requesting input from the console.

2.4.8 @octal-address

This command begins execution of a previously loaded program. The octal address specified is the address at which execution is to begin.

2.5 E

E {location}

E examines a core location in the user's area (high or low segment).

The location is the address specified; the contents being typed out in half-word octal mode. The address is required the first time the E or D command is used. If the address is not specified, the contents of the location following the previously specified E address, or the location of the previous D address are typed out.

2.6 D

D lh rh {location}

D deposits information in the user's core area (high or low segment).

lh is the octal value to be deposited in the left half of the location. rh is the octal value to be deposited in the right half of the location. location is the address of the location into which the information is to be deposited. If the address is omitted, the data is deposited in the location following the last D address or in the location of the last E address.

2.7 Example

```
.LOAD ABC/REL↵
LOADING

LOADER 1K CORE

EXIT
↑C

.SAVE ABC↵

JOB SAVED
↑C

.GET ABC↵

JOB SETUP
↑C

.E 137↵
000137/ 000002 000030      .@ST↵

;begins execution

EXIT
↑C

.DIR(F) ABC/ALL↵

DIRECTORY      PROJECT 46      13:24      12-AUG-72
FILENAME       KWDS KEPT       ACCESS     CREATED

ABC    /REL    0.25 K          F N        16-MAR-72
ABC    /SAV    0.38            F N        22-AUG-72
```

```
TOTAL SIZE     0.63

EXIT
↑C

.
```

3 FACILITY ALLOCATION COMMANDS

The monitor allocates devices and core memory to users upon request, and protects these allocated facilities from interference by other users. The monitor maintains a pool of available facilities from which a user can draw.

A user should never abandon a timesharing console without returning his allocated facilities to the monitor pool. Until a user returns his allocated facilities to the pool no other users may utilize them.

All devices controlled by the system have associated with them a physical name, consisting of three letters (and zero to three numerals to specify the unit number). A logical device name may also be assigned by the user. This logical name (one to six alphanumeric characters) is used synonymously with a physical device name in all references to the device. In writing a program, the user may arbitrarily select device names that he assigns to the most convenient physical devices at runtime. All references to devices in the monitor pool are made by physical names or by assigned logical names. When a device is assigned to a job, it is removed from the monitor's pool of available facilities. The device is returned to the pool when the user deassigns it or ends his job.

3.1 ASSIGN

```
ASSIGN $dev{=log-dev}
```

Assign allocates an I/O device to the user's job for the duration of the job, or until a Deassign command is given. Assign may be abrreviated to AS.

dev is the device DSK or TTY. This argument is required. log-dev is the logical name assigned by the user (optional).

This is a useful command in allocating logical device numbers to devices for Fortan programs. For example,

AS $DSK=6 will send output to the disk (as filename FORØ6) instead of the job output device,

AS $TTY=1Ø will receive/send data via the teletype instead of the disk.

3.2 DEASSIGN

DEASSIGN $dev

Deassign returns one or more devices currently assigned to the user's job to the monitor pool of available devices. Deassign may be abbreviated to DEAS.

If dev is not specified, all devices assigned to the user's job are deassigned. If the argument is specified, it can be either the logical or physical device name.

3.3 EOF

EOF $dev

Eof terminates any input or output currently in progress on the device.

dev is the logical or physical name of the device on which I/O is to be terminated. If no argument is specified, I/O is terminated on all devices assigned to the job.

3.4 CORE

CORE core

Core is used to modify the amount of core assigned to the user's job.

If core is Ø the low and high segments disappear from the job's virtual addressing space. If core is greater than Ø then this argument specifies the total number of 1K blocks of core to be assigned to the job from this point on. If this argument is omitted, the monitor types out the same response as when an error occurs, but does not change the core assignment.

The response is
m+n/p CORE

VIR. CORE LEFT=v

where

m = number of 1K blocks in low segment
n = number of 1K blocks in high segment
p = maximum K per job (max phys user core)
v = number of K unassigned in core and swapping device

3.5 RESOURCES

RESOURCES

Resources will print out all the available devices in the system device pool (except teletypes) and the number of free blocks on the public disk packs. Resources may be abbreviated to RES.

4 ADMINISTRATION COMMANDS

4.1 PJOB

PJOB

The monitor responds by typing the job number to which the user's console is attached. If the console is not attached to a job, the monitor responds with 'LOGIN PLEASE'. PJOB may be abbreviated to PJ.

4.2 SYSTAT

SYSTAT {option|job-no}

Systat types out the status of the system: the system name, time of day, date, uptime, percent null time. It also gives:

Status of each job: job number, project number, TTY number, program name being run, size of low segment, state of program (RN=runnable, TT=TTY input wait, ↑C=monitor command mode) and run time.

Status of high segments being used: name, directory name, size, number of users in core or no disk.

Status of each assigned device: name, job number, how assigned (AS=ASSIGN command, INIT=INIT UUO).

The arguments are optional; if omitted a full systat will be given.

The options specify various phases of the systat.

B gives busy devices
D dormant segments
J job status
N non-job status
S short job status

job-no gives the job status relevant to a particular job.

Systat may be abbreviated to SYS.

4.3 HELP

HELP name

The Help command prints out helpful documentation on various Computer Centre facilities.

If a name is specified, Help will look for, and print out the information about the facility named in the command. For example, HELP MANUAL will print out the current status of the Centre manuals.

HELP ALL will give a list of all currently available information. Only the first six characters of the argument are looked at. They must consist of the characters A-Z, 0-9.

Help is initially available for the following:

NEWS the latest weekly newsletter
MANUAL the current status of the manuals
LOGIN how to login into the system

5 SEGMENTS AND REENTRANT PROGRAMS

In a non-reentrant system, each user has a separate copy of a program even though a large part of it is the same as for other users. In a reentrant system hardware allows a user area to be divided into two logical segments which may occupy non-contiguous areas in core. The monitor allows one of the segments of each user area to be the same as one or more other users, so that only one copy of a shared segment need exist no matter how many users are using it.

5.1 Segments

A segment is a continuous region of the user's core area maintained in core or on the swapping device. A program or user job is composed of one or two segments. A segment may contain instructions and/or data. The monitor determines the allocation and movement of segments in core and on the swapping device.

5.2 Sharable Segments

A sharable segment is a segment which is the same for many users. The monitor keeps only one copy in core and/or on the swapping device, no matter how many users are using it. A non-sharable segment is a segment which is different for each user in core and/or on the swapping device.

The PDP-10's hardware permits a user program to be composed of one or two segments at any point in time. The required low segment starts at user location 0. The optional high segment starts at user location 400000 or at the end of the low segment, whichever address is greater. The low segment contains the user's accumulators, job data area, instructions and/or data, I/O buffers, and DDT symbols. A user's core image is composed of a low segment, in multiples of 1K (1K=1024_{10} words), and a high segment also in multiples of 1K. A high segment may be sharable or non-sharable, whereas a low segment is always non-sharable.

5.3 Reentrant Programs

A reentrant program is always composed of two segments - a low segment which usually contains just data, and a high (sharable) segment which usually contains instructions and constants. The low segment is sometimes referred to as the impure segment. The sharable high segment, if write-protected, is referred to as the pure segment.

A one segment non-reentrant program is composed of a single low segment containing instructions and data. A two segment non-reentrant program is composed of a low segment and a non-sharable high segment.

The Save command supplies standard processor program names to the segments it creates. A one segment non-reentrant program has a processor program name /SAV. For two segment programs the low segment has the processor program /LOW. The high segment is name /HGH or /SHR depending on whether or not it is a sharable segment.

The S option in the Save command is used to make the high segment sharable when it is loaded with the Get command. To indicate this sharability, the high segment is written with processor program /SHR instead of /HGH. A subsequent Get will cause the high segment to be sharable. Because an error message is not given if the program does not have a high segment, a user can use this command to save without having to know which are sharable.

5.4 User's Core Storage

A user's core storage consists of blocks of memory whose sizes are an integral multiple of 1024_{10} (2000_8) words. There are two methods available to the user for loading his core area. The simplest way is to load a core image stored on disk (see Get). The other method is to use the relocatable binary loader to link-load binary files. The user may then write the core image on disk for future use (see Load and Save).

5.5 Job Data Area

The job data area provides storage for specific information of interest to both the monitor and the user. The first 140 (octal) locations of the user's core area are always allocated to the job data area. Locations in this area have been given mnemonic assignments whose first three characters are .JB (or JOB). Therefore, all mnemonics referred to with a .JB (or JOB) prefix refer to locations in the job data area. It is planned eventually to replace the JOB prefix by .JB, but in the meantime either prefix is acceptable.

Table 1

Job Data Area Locations

name	octal location	description
.JBUUO	40	User's location 40_8. Used for processing user UUO's (001 through 137). Operation code and effective address are stored here.
.JB41	41	User's location 41_8. Contains the beginning address of the user's programmed operator service routine (usually a JSR or PUSHJ).
.JBERR	42	Left half: unused at the present. Right half: accumulated error count from one

program to the next. Programs should be written to look at the right half only.

.JBREL 44 Left half: Ø
Right half: the highest relative core location available to the user (i.e. the contents of the memory protection register when this is running).

.JBDDT 74 Contains the starting address of DDT. If contents are Ø, DDT has not been loaded.

.JBHRL 115 Left half: first relative free location in the high segment (relative to the high segment origin so it is the same as the high segment length). Set by the loader and subsequent Gets, even if there is no file to initialize the low segment. The left half is a relative quantity because the high segment can appear at different user origins at the same time. The Save command uses this quantity to know how much to write from the high segment.
Right half: highest legal user address in the high segment. Set by the monitor every time the user starts to run. The word is ⩾ 401777 unless there is no high segment, in which case it will be zero. The proper way to test if a high segment exists is to test this word for a non-zero value.

.JBSYM 116 Contains a pointer to the symbol table created by the linking loader.
Left half: negative count of the length of the symbol table.
Right half: lowest register used.

.JBUSY 117 Contains a pointer to the undefined symbol table created by the linking loader.

.JBSA 120 Left half: first free location in the low segment (set by the loader).
Right half: starting address of the user's program.

.JBFF 121 Left half: Ø
Right half: address of the first free location following the low segment.

.JBREN 124 Left half: unused at present
Right half: the reentry starting address. Set by the user or by the linking loader and used by @REE command as an alternative entry point.

.JBOPC	130	The previous contents of the user's program counter are stored here by the monitor upon execution of an @DDT, @REE, @ST, or @CST command.
.JBCHN	131	Left half: Ø address of first location after first Fortran IV loaded program. Right half: address of first location after first Fortran IV block data.
.JBCOR	133	Left half: highest location in low segment loaded with non-zero data. No low file written on Save if less than 140. Set by the loader. Right half: user argument on last Save or Get command. Set by the monitor.
.JBVER	137	Left half: first digit represents who last edited the program, next three digits give major version number, last two digits gives minor version number (alpha). Right half: incremental edit number. The number is never converted to decimal. After a Get command, an E command can be used to find the version number.
.JBDA	140	The value of this symbol is the first location available to the user.

Note that only those JOBDAT locations of significant importance to the user are given in this table. JOBDAT locations not listed included those which are used by the monitor and those which are unused at the present time. User programs should not refer to any locations not listed above since such locations are subject to change without notice.

Some locations in the job data area, such as .JBSA and .JBDDT, are set by the user's program for use by the monitor. Others, such as .JBREL, are set by the monitor for use by the user's program. In particular, the right half of .JBREL contains the highest legal address set by the monitor whenever the user's core allocation changes.

JOBDAT exists in binary form in the System Library for loading with user programs that refer to job data area locations symbolically. User macro programs must reference locations by means of the assigned mnemonics, which are declared as EXTERNAL references to the assembler JOBDAT is loaded automatically, if needed, during the loader's library search for undefined global references, and the values assigned to the mnemonics.

BATCH

[WN-104]

A new version of batch is to be released. The only changes in this version are concerned with improving QUIT and EOJ procedures and permitting the use of magnetic tape for input. A program has been written for stacking of batch jobs on magnetic tape using the GE-225. It is intended that results should be identical w th the present system, although if it is found that a submitted deck contains illegal characters, a log of the job indicating the illegal characters may be returned with the job. This log will have a file identification including the JOBID and .JOB card images. Illegal cards will be listed, together with their number and a following line will indicate the location of the illegal character(s). The end of the job will be indicated by a summary that includes the number of cards, the number of bad cards and the number of bad characters.

PDP-10 FORTRAN

[WN-104]

1 ERROR

In certain circumstances the 640th character of an input data file is ignored. This occurs only if the first line of the file is skipped through the use of a format statement beginning with one or more slashes. This error may be avoided by changing

```
FORMAT(/ . . .
```

to

```
FORMAT (1X/ . . .
```

when the first line is to be skipped.

[WN-105]

2 FORTRAN MANUAL MNT-5

2.1 Page 6A-13

The paragraph beginning 'It is inadvisable . . .' should be amended to read:

> It is inadvisable to create a file in the first instance or enlarge a file further ahead than the next sequential record using direct access writes. A file to be directly accessed should initially be created by

sequentially writing the full number of blank records required, releasing the channel and then defining the file appropriate. Note that the record size for the sequential write and the define file must be the same.

2.2 Page 6A-14

The second form of END and ERROR

READ (u'r, f, END=n, ERR=m) list

will not be successful in detecting the end of the file, since at present, if r, the record number, is < 1 or > the last record number, the result is a fatal 'ERROR IN JOB n ILL MEM REF'.

BMD ERRORS CORRECTED

[WN-104]

1 BMD02D

Two errors in BMD02D have been corrected. They were a failure to read data from a disk file, and incorrect interpretation of boolean condition cards.

2 BMD02R

An error that resulted in the partial correlation for variables not in the equation not being output has been corrected. The error occurred only in certain cases when labels were provided by the user, and did not affect computation.

SCIENTIFIC SUBROUTINE PACKAGE

[WN-102]

A number of double precision routines use a function DFLOAT to convert from integer to double precision. This operation is handled directly by our compiler and in due course these routines may be modified to eliminate the use of a library call for this purpose.

In the meantime, if the undefined global DFLOAT occurs, it can be eliminated by incorporating the trivial function listed below in your source program.

```
DOUBLE PRECISION FUNCTION DFLOAT(I)
DFLOAT=I
RETURN
END
```

NEW VERSIONS OF SYSTEMS PROGRAMS

[WN-104]

1 ACCOUNTING PROGRAMS

New versions of the following system programs were implemented on Thursday 24 August; LOGIN, ACCOUN, LIMIT, IDENT, QUIT, LIST, PASSWORD and FINISH. Several of the accounting files have been divided into a number of smaller files in order to reduce the delays of multiple access. At the same time some smaller changes were introduced into ACCOUN and FINISH.

(a) ACCOUN

A job number may now be accepted as an argument. For example,

```
ACCOUNT(FULL) 20
```

(b) FINISH

If the terminal job has given an IDENT command and no corresponding QUIT, FINISH warns the user and gives him a chance to abort the logout. The corresponding patch to EOJ requires additional changes to the monitor and batch, and will be implemented shortly.

[WN-105]

2 BASIC VERSION 17A

The Centre is contemplating replacing the present system version of Basic with version 17A. However, there are one or two problems associated with this:

(a) file differences

The new version uses a different format for its program files that involves setting bit 35 of words that contain line numbers. Such files cannot be directly listed but it would be a simple matter to make them listable if this were an important requirement. Note that files produced by the present Basic are acceptable as input, as are any files created by Editor.

(b) increased size

Version 17A may use up to 14K of storage (2K low segment and 12K high segment) while the current version only takes up 10K of storage (3K low segment 7K high segment).

Version 17A offers greatly improved facilities, for example a formatted output capability, a much extended file capability and a means of linking between programs. It is felt that the advantages of 17A outweigh the disadvantages of file compatibility problems amd increased size. While the Centre will be contacting major user groups for their opinions; comments on this proposal will be welcomed from any interested user. Please contact Mr de Voil on extension 8168.

3 AID VERSION 17

Version 17 of AID has been put on the system. This version corrects reported errors (for example an error in the ARG function).

4 NEW TYPE 4 PROGRAM RELEASES

The following programs have all been released as type 4 programs. There is temporary documentation available in the clients room for reference only.

(a) SIM-11

Released on the SIMUL directory. This is version 12 of the PDP-11 simulator, which allows PDP-11 programs etc to be developed on the PDP-10 without the aid of a PDP-11 processor. Documentation reference is Decus 10-28.

(b) PALX-11

Released on the LANG directory. Documentation on the PAL-11 assembly language is in Decus 10-31.

(c) GIST

Gist is a General Interpretive String Translator for the PDP-10. It is released on the LANG directory. Documentation is in Decus 6/10-39.

(d) MEDFOR

Medfor is a fast Fortran compiler, developed by the Medical School computing facility of the University of Pensylvania. It is released on the LANG directory. See the documentation for a list of error messages as well as discussion on running Medfor and a cost comparison between Medfor and Fortran.

MULTICHOICE TEST MARKING SCHEME - EVAL

D. Woodrow

1 INTRODUCTION

The implementation of the Radford recommendations in Queensland secondary schools has resulted in an increase in the number of short multichoice tests being used. The final examination has been replaced to a large extent by continuous assessment, which in science, includes a large percentage of muliple choice items.

Teachers who have endeavoured to construct original items, appreciate the amount of time required to structure a valid and reliable test without the added time for marking and item analysis. The 'tried and true' examination questions reappear on most tests. Multichoice testing provides an opportunity for effective item analysis; however the time required to go through the whole procedure is seldom available, and the validity and reliability of the reoccuring items remains a matter of speculation.

Progressive assessment also highlights an additional problem - that of combining test results to give final grades. The simple

---

The author is science master at St Peter's Lutheran College in Brisbane, and Vice President of the Science Teachers Association. In addition he is District Moderator for Science and is thus closely concerned with the effects resulting from the Radford implementation.

adding of raw scores to give final results (if means and standard deviations were not considered) could lead, in many cases to incorrect ranking and grading.

2 REQUIREMENTS

The above situation required a readily available, economical and simple method of:

(a) marking multichoice tests

(b) standardizing scores, and

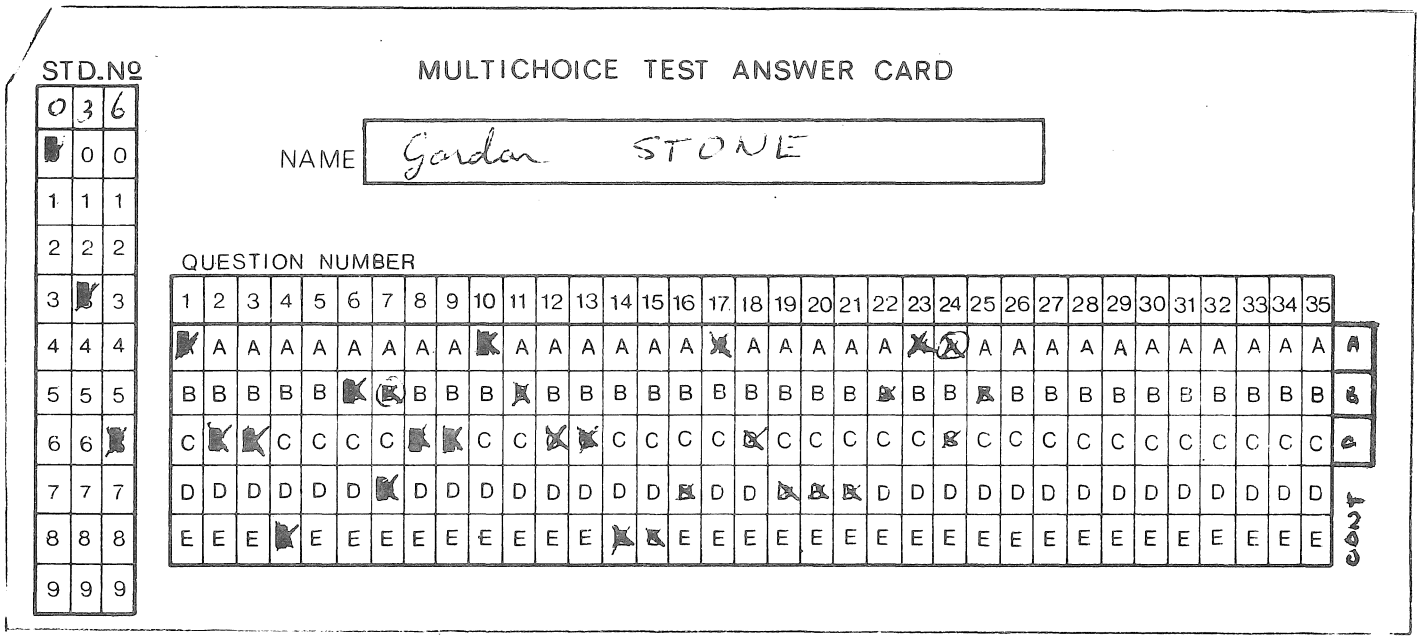
(c) item analysis.

The idea of computer analysis of tests had been previously investigated using optical scanning or punched cards, but systems proved uneconomical and too difficult to organize.

3 SYSTEM

The following system, involving the use of student punched 'porta punch' or 'minitran' type cards, has been developed and tested by several Brisbane schools and tertiary educational establishments. It is now available for use through the University of Queensland Computer Centre.

3.1 Cards



Each card takes 35 items with up to 5 responses (A to E). It also includes space for the student's name and number. Continuation marks enable a further card to be used making it possible to process a test of up to 70 items. This is the current limit although the program will be modified to handle as many items per student as required.

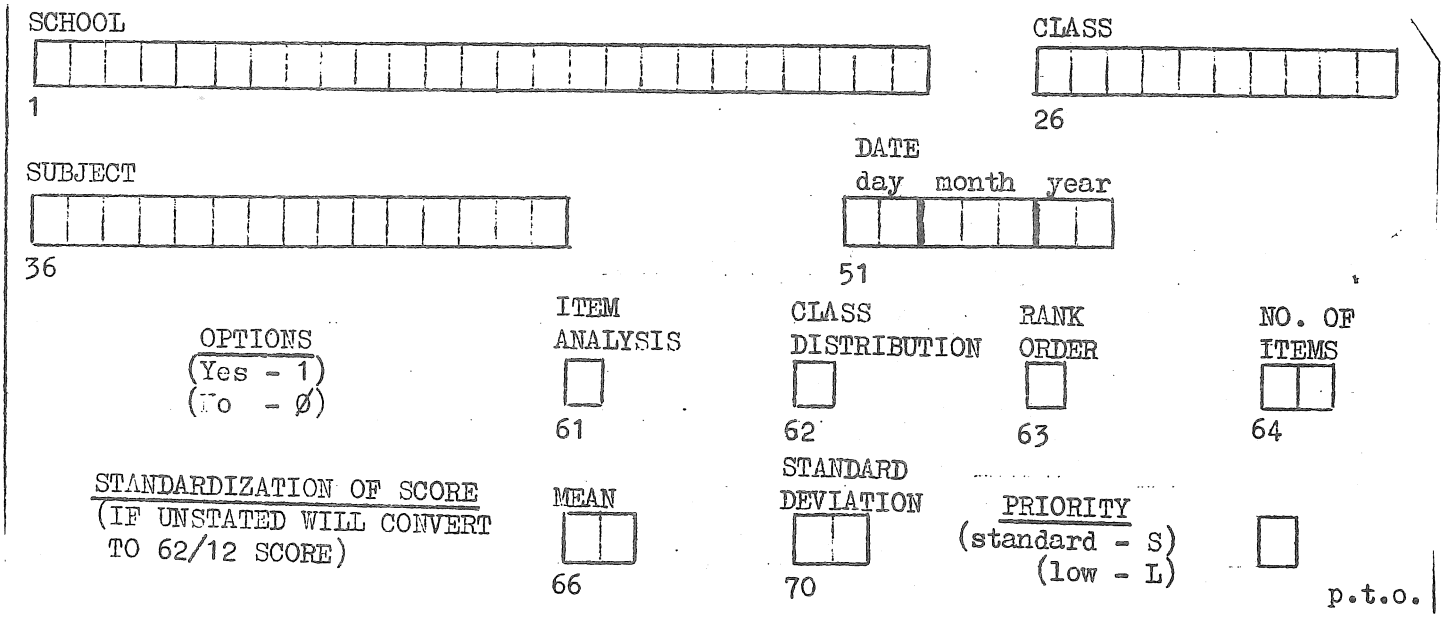
3.2 Recording Answers

The students mark their responses directly onto the cards, in pencil or biro during the test. Corrections can be made as the response is not finally recorded until the chad is punched out.

After the test is completed the chads are punched out using a bent paper clip (Figure 1). This takes about two minutes.

3.3 Processing

The cards are then collected, checked and a 'correct response' card is punched and put on the front of the deck together with a form containing information such as 'name of school', 'test title', etc (Figure 2).



Complete details of 'instructions to students', 'methods of processing', etc will be given in a booklet to be produced.

Currently, processing is being done on the University's PDP-10 computer.

4 OUTPUT

The output is on standard computer stationery.

4.1 Marking

The test is marked and the raw score and standard score listed (Figure 3). (We have been using the old Junior mean of 62 and standard deviation of 12).

4.2 Options

The following three options, as well as marking, are available if requested.

(a) Item Analysis

This gives an analysis of each item, listing the frequency and percentage distribution of each response, the item difficulty and discrimination.

An explanation of these terms is given at the end of the printout for those unfamiliar with educational statistics (Figure 4).

This analysis enables a teacher to identify quickly items that were too easy or too difficult, and items that did not discriminate well. 'Non distracting' distractors can be replaced and the valid items restructured. It is surprising how some 'tried and true' items turn out.

Sections of the work requiring extra attention can be identified easily.

(b) Class Statistics

This lists the range of scores, frequency, and cumulative percent, and prints out a histogram of score frequency (Figure 5).

Class statistics give an indication of overall performance and a comparison with other groups on the quality of student performance and/or test instruments.

(c) Rank Order

This lists the place in grade, and raw and standard scores (Figure 6).

Teachers are able to use this placing for internal moderation procedures and it is useful for schools that still use place in grade on school reports to parents.

The main value of the system, I feel, is in the area of item analysis - this must result in the improvement of the quality of testing.

4.3 Option Use

Either the marking or rank order output sheet can be displayed on notice boards. As there are no names, only student numbers recorded, students do not appear to object to this. The sheets are printed in double line spacing for readability.

5 PROGRAM LIMITATIONS

The program has a limit of 400 students at the moment (to reduce cost) but can easily be altered for larger groups. The current limitation of 70 items per test will shortly be removed.

6 COST

6.1 Cards

These will be available from the publications officers of the Science Teachers and Mathematics Teachers Associations for approximately $6.50 per thousand.

6.2 Instruction Manuals

These are being prepared. For those interested in investigating the system initially, a manual and 200 cards will be available for approximately $2.

6.3 Processing Cost

This depends on the size of the job. For 100 students and a 35 item test with all options, the cost was approximately $2.60 at standard priority on external rates. Low priority rates (20% less) are available and processing will usually be done overnight. Processing time for the above test was 16.4 seconds.

7 PROCESSING ARRANGEMENTS AND ENQUIRIES

Anyone wishing to enquire about using this package can do so by contacting any of the following people concerned:

Dave WOODROW
St Peter's Lutheran College (phone 70 7141)

John ELKINS
Schonell Educational Research Centre (phone 71 3611)

Eddie BRAY
Brisbane State High School (phone 44 4161)

Graeme COOTE
Kelvin Grove Teacher's College (phone 56 9311)

Alternatively, if they have their own account at the Centre they can access the package with the command

177.EVAL/SAV

in front of their data deck.

8 ACKNOWLEGEMENTS

The author would like to express his appreciation to the following people: John ELKINS, Graeme COOTE, Eddie BRAY (for voluntary assistance given in writing the programs) and to the teachers who tried out the system and offered advice on the requirements.

ANALYSIS OF A MULTIPLE CHOICE TEST

SCHOOL: COORPAROO EVENING CLASSES TEST: BIOLOGY TERM 2

CLASS: SENIOR DATE: 7-AUG-72

NO. OF ITEMS = 60

STUDENT	RAW SCORE	BLANK	ILLEGAL	STD.SCORE
3 1 0	31	0	0	59.
3 1 1	24	0	0	48.
3 1 2	29	0	0	56.
3 1 3	25	0	0	50.
3 1 4	46	0	0	81.
3 1 5	33	1	0	62.
3 1 6	46	0	0	81.
3 1 7	23	0	0	47.
3 1 8	23	0	0	47.
3 1 9	42	1	0	75.
3 2 0	28	3	0	54.
3 2 1	37	0	0	68.
3 2 2	42	0	0	75.
3 2 3	39	0	0	71.
3 2 4	38	0	0	69.
3 2 5	25	0	0	50.
3 2 6	21	4	0	44.
3 2 7	34	0	0	63.
3 2 8	35	0	0	65.
3 2 9	41	1	0	74.
3 3 0	29	0	0	56.
3 3 1	26	0	0	51.
3 3 2	46	0	0	81.

NO. OF STUDENTS = 23 CLASS MEAN = 33.2 REQUIRED MEAN = 62.7

CLASS STD. DEVIATION = 8.0 REQUIRED STD. DEVIATION = 12.5

FIGURE 3

ITEM ANALYSIS

		DISTRIBUTION OF RESPONSES						ITEM CHARACTERISTICS	
ITEM NO.	A	B	C	D	E	BLANK	ILLEGAL	DIFFICULTY	DISCRIMINATION
1 FREQ	2	36	1	49	2	0	0	2,54	0,36
PCT	2	40	1	54	2	0	0		
2 FREQ	4	78	1	4	2	1	0	0,87	0,35
PCT	4	86	1	4	2	1	0		
3 FREQ	1	2	51	12	22	2	0	0,57	0,24
PCT	1	2	56	13	24	2	0		
PCT	10	8	14	57	7	1	0		
35 FREQ	7	17	9	35	22	0	0	0,39	0,35
PCT	7	18	10	38	24	0	0		

DIFFICULTY: THE DIFFICULTY IS CALCULATED FROM THE PERCENTAGE OF STUDENTS WHO GET THE QUESTION CORRECT. (THE LOWER THE DECIMAL- THE HARDER THE QUESTION) COEFFICIENTS OF ABOVE 0,80 WOULD INDICATE EASY QUESTIONS AND COEFFICENTS OF BELOW 0,20 WOULD INDICATE DIFFICULT ITEMS. A DIFFICULTY COEFFICIENT OF AROUND 0,50 SHOULD GIVE MAXIMUM RELIABILITY.

DISTRACTORS SCORING LESS THAN 10 PERCENT OF RESPONSES ARE PROBABLY TOO WEAK.

DISCRIMINATION: THE DISCRIMINATION INDEX FOR EACH QUESTION MEASURES THE EXTENT TO WHICH THE STUDENTS WITH HIGH TOTAL SCORES GAVE THE CORRECT RESPONSE MORE OFTEN THAN THOSE WITH LOW SCORES. AN INDEX OF 0,4 OR MORE INDICATES A QUESTION WHICH CORRELATES WELL WITH THE OTHERS. IF THE INDEX IS LOW (BELOW 0,10 OR NEGATIVE) THE QUESTION SHOULD BE EXAMINED CAREFULLY.

FIGURE 4

CLASS STATISTICS
****

SCORE FREQUENCY DISTRIBUTION

SCORE	FREQ.	CUM. PCT.	HISTOGRAM
5	1	1.	*
6	0	1.	
7	0	1.	
8	1	2.	*
9	0	2.	
10	0	2.	
11	2	4.	**
12	2	7.	**
13	3	10.	***
14	1	11.	*
15	2	13.	**
16	3	17.	***
17	0	17.	
18	2	19.	**
19	7	27.	*******
20	5	32.	*****
21	8	41.	********
22	4	46.	****
23	8	54.	********
24	7	62.	*******
25	2	64.	**
26	5	70.	*****
27	6	77.	******
28	6	83.	******
29	3	87.	***
30	2	89.	**
31	3	92.	***
32	6	99.	******
33	0	99.	
34	1	100.	*

CLASS MEAN =22.7 NO. OF STUDENTS = 90

FIGURE 5.

RANK ORDER
****

STUDENT	PLACE	RAW SCORE	STD. SCORE
3 1 4	1	46	81.
3 1 6	1	46	81.
3 3 2	1	46	81.
3 1 9	4	42	75.
3 2 2	4	42	75.
3 2 9	6	41	74.
3 2 3	7	39	71.
3 2 4	8	38	69.
3 2 1	9	37	66.
3 2 8	10	35	65.
3 2 7	11	34	63.
3 1 5	12	33	62.
3 1 0	13	31	59.
3 1 2	14	29	56.
3 3 0	14	29	56.
3 2 0	16	28	54.
3 3 1	17	26	51.
3 1 3	18	25	50.
3 2 5	18	25	50.
3 1 1	20	24	48.
3 1 7	21	23	47.
3 1 8	21	23	47.
3 2 6	23	21	44.

NO. OF STUDENTS = 23 CLASS MEAN = 33.2 REQUIRED MEAN =62.7

CLASS STD. DEVIATION = 8.0 REQUIRED STD. DEVIATION =12.0

FIGURE 6