ESTERLY

AD10

SOFTWARE REFERENCE MANUAL

ADI

APPLIED DYNAMICS INTERNATIONAL
3800 STONE SCHOOL ROAD / ANN ARBOR, MI 48104 / PH. 313-973-1300 / TLX. 230238

TABLE OF CONTENTS PAGE

TERMS AND SYNTAX CONVENTIONS

ADX	NAME OF THE AD-10 EXECUTIVE PROGRAM
DEC	DIGITAL EQUIPMENT CORPORATION, MAKERS OF THE PDP-11 COMPUTER
RSX-11	DEC'S REALTIME OPERATING SYSTEM EXECUTIVE FOR THE PDP-11
UIC	THE RSX-11 USER IDENTIFICATION CODE (REF. CHAPTER 3 OF THE RSX-11 OPERATOR'S PROCEDURES MANUAL)
MCR>	PROMPT GENERATED BY THE RSX-11 MONITOR CONSOLE ROUTINE
ADX>	PROMPT GENERATED BY THE AD-10 EXECUTIVE
INFILE, OUTFILE	TERMS USED TO INDICATE WHERE THE INPUT SPECIFIED BY AN ADX COMMAND STRING IS TO COME FROM, OR WHERE THE OUTPUT GENERATED IS TO GO.
@	INDIRECT FILE SPECIFIER (REF. CHAPTER 5 OF THE RSX-11 OPERATOR'S PROCEDURES MANUAL)
/	ADX COMMAND STRING SWITCH DELIMITER
:	ADX COMMAND STRING SWITCH VALUE DELIMITER
< >	ANGLE BRACKETS ARE USED TO ENCLOSE THE NAME OF A SYNTACTIC ELEMENT OR CLASS OF ELEMENTS IN COMMAND STRING EXAMPLES
[ ]	BRACKETS ARE USED TO ENCLOSE OPTIONAL SYNTACTIC ELEMENTS IN COMMAND STRING EXAMPLES.
CR	CARRIAGE RETURN (RETURN)
CONTROL-Z CTRL-Z	CHARACTER GENERATED BY SIMULTANEOUSLY DEPRESSING THE "CONTROL" AND THE "Z" KEYS. USED TO EXIT FROM ADX (AND RETURN TO MCR) WITHOUT AFFECTING THE AD-10.
CONTROL-U CTRL-U	"CONTROL" AND "U" KEYS. USED TO DELETE A LINE.
RUBOUT	USED TO DELETE A CHARACTER.
CONTROL-S CTRL-S	"CONTROL" AND "S" KEYS. USED TO STOP THE OUTPUT WHEN THE DISPLAY IS SCROLLING.
CONTROL-Q CTRL-Q	"CONTROL" AND "Q" KEYS. USED TO RESTART THE OUTPUT AFTER THE DISPLAY SCROLLING HAS BEEN STOPPED.

THE AD-10 EXECUTIVE

THE AD-10 EXECUTIVE (ADX) IS A SOFTWARE TOOL WHICH RUNS UNDER DEC'S RSX-11 OPERATING SYSTEM ON THE PDP-11 COMPUTER AND ALLOWS THE USER TO CONTROL AND MONITOR THE OPERATION OF THE AD-10. ADX ALLOWS THE USER TO START AND STOP THE AD-10, AS WELL AS TO STEP IT FOR A SPECIFIED NUMBER OF INSTRUCTION CYCLES. IT PERMITS SPECIFIC AD-10 REGISTERS, PROGRAM MEMORY LOCATIONS, AND DATA MEMORY LOCATIONS TO BE READ OR WRITTEN AS REQUIRED. USING ADX, SPECIFIED SECTIONS OF PROGRAM MEMORY OR DATA MEMORY CAN BE SAVED IN PDP-11 COMPUTER SYSTEM FILES AND CAN LATER BE RELOADED, AS CAN LOAD MODULES CREATED BY THE PDP-11/AD-10 CROSS ASSEMBLER. IN ADDITION, ADX DEBUGGING COMMANDS ALLOW THE USER TO INSERT BREAKPOINTS IN AD-10 PROGRAMS, TO LOG OR INSERT INTERNAL MULTIBUS DATA AT SPECIFIED POINTS IN AD-10 PROGRAMS, AND TO TRACE THE AD-10 PROGRAM FLOW. THE USER COMMANDS MAY COME FROM THE TERMINAL OR FROM AN INDIRECT COMMAND FILE, AND THEY USE STANDARD RSX-11 COMMAND STRING SYNTAX CONVENTIONS.

THE ADX COMMANDS ARE ORGANIZED INTO SIX BASIC CATEGORIES:

1. INFORMATIONAL COMMANDS (IDENTIFY,FRACTION,FLOATING)

   THESE COMMANDS HAVE NO EFFECT UPON THE AD-10 SYSTEM. THEY MERELY PROVIDE THE USER WITH THE SPECIFIED INFORMATION.

2. AD-10 START/STOP COMMANDS (CONTINUE,RUN,STEP,HALT,EXIT)

   THIS GROUP OF COMMANDS EITHER STARTS OR STOPS THE AD-10 PROGRAM.

3. AD-10 WRITE COMMANDS (INIT,LOAD,SET,CLEAR,MODIFY,ZERO)

   THESE COMMANDS DO SOMETHING "TO" THE AD-10, CHANGING THE CONTENTS OF AD-10 REGISTERS, PROGRAM MEMORY LOCATIONS, OR DATA MEMORY LOCATIONS.

4. AD-10 READ COMMANDS (SAVE,DISPLAY,DUMP,HISTORY,TEST)

   THESE COMMANDS GET DATA "FROM" THE AD-10, ENABLING THE USER TO LOOK AT (AND/OR SAVE IN A FILE) THE CURRENT VALUES OF AD-10 REGISTERS, PROGRAM MEMORY LOCATIONS, AND DATA MEMORY LOCATIONS.

5. AD-10 DEBUGGING COMMANDS (AT,BREAK,RESTORE,TRACE,UNTRACE)

   THIS GROUP OF COMMANDS ENABLES THE USER TO DYNAMICALLY CHECKOUT THE AD-10 PROGRAM. THEY PROVIDE THE USER WITH INFORMATION OR CONTROL AT THE SPECIFIED POINTS WITHIN THE AD-10 PROGRAM.

6. AD-10 CONSOLE COMMANDS (ATTACH,DETACH,CONSOLE)

   THIS GROUP OF COMMANDS ALLOWS THE USER TO CONTROL THE ACCESS AND USE OF THE VARIOUS CONSOLES. THE USER CAN RESERVE AND FREE CONSOLES FOR HIS EXCLUSIVE USE AND SWITCH FROM ONE CONSOLE TO ANOTHER.

RSX-11 OVERVIEW

RSX-11 IS DEC'S REALTIME, MULTIPROGRAMMING OPERATING SYSTEM EXECUTIVE FOR THE PDP-11 SERIES OF COMPUTERS. THE RSX-11 EXECUTIVE ALONE USES ABOUT 8K WORDS OF MEMORY AND REQUIRES THAT THERE BE AT LEAST ONE FILE-STRUCTURED DEVICE IN THE SYSTEM. IT PROVIDES THE NECESSARY CONTROL FOR SHARING SYSTEM RESOURCES AMONG ANY NUMBER OF USER-PREPARED "TASKS" (PROGRAMS). THESE USER TASKS ARE USUALLY CREATED AS FOLLOWS:

1. THE USER WRITES A SOURCE PROGRAM AND PUTS IT INTO A FILE ON A SYSTEM FILE-STRUCTURED DEVICE USING THE LINE TEXT EDITOR (EDI).

2. THE APPROPRIATE TRANSLATOR ROUTINE IS THEN USED TO COMPILE (FOR) OR ASSEMBLE (MAC) THE SOURCE PROGRAM, CREATING AN OBJECT FILE OF HIS PROGRAM.

3. THIS OBJECT FILE, ALONG WITH ANY OTHERS WHICH MAY BE REQUIRED FOR THIS TASK AND WHICH WERE CREATED SEPARATELY, ARE THEN SUBMITTED TO THE TASK BUILDER ROUTINE (TKB) WHICH LINKS THEM TOGETHER AND CREATES A "TASK IMAGE FILE".

4. THIS TASK IMAGE FILE MAY THEN BE "INSTALLED" (INS) INTO THE SYSTEM (MEANING ESSENTIALLY THAT THE TASK'S NAME, SIZE, AND LOCATION ARE MADE KNOWN TO THE SYSTEM, BUT THAT THE TASK IS STILL DORMANT).

5. ONCE INSTALLED, THE TASK MAY BE EXECUTED BY A USER COMMAND (RUN) OR BY ANOTHER TASK.

THE COMMANDS TO PERFORM THESE FUNCTIONS ARE GIVEN BY THE USER TO THE MONITOR CONSOLE ROUTINE (MCR) WITHIN THE RSX-11 EXECUTIVE. THE SYSTEM TASKS (EDITOR,MACRO-11 ASSEMBLER,FORTRAN-IV COMPILER,TASK BUILDER, PERIPHERAL INTERCHANGE PROGRAM, ETC.) ARE INVOKED BY TYPING, IN RESPONSE TO THE "MCR>" PROMPT, THEIR THREE CHARACTER TASK NAME (EDI,MAC,FOR,TKB,PIP,ETC.) FOLLOWED BY A CARRIAGE RETURN. USER PROGRAMS ARE GENERALLY STARTED BY THE "RUN <TASKNAME>" COMMAND.

NOTICE THAT ADX IS CONSTRUCTED TO APPEAR AS ANOTHER RSX-11 SYSTEM TASK.

SUCCESFUL OPERATION OF AN AD-10 SYSTEM IS GREATLY ENHANCED BY THE USER'S UNDERSTANDING OF THE RSX-11 OPERATING SYSTEM AND ITS CAPABILITIES. THE USER SHOULD REFER TO DEC'S RSX-11 OPERATOR'S PROCEDURES MANUAL FOR A MORE THOROUGH DESCRIPTION OF THE OPERATING SYSTEM AND THE MCR COMMANDS. THE RSX-11 UTILITIES PROCEDURES MANUAL CONTAINS DESCRIPTIONS OF THE SYSTEM UTILITY PROGRAMS, AND THE PDP-11 FORTRAN REFERENCE MANUAL AND THE IAS/RSX-11 MACRO-11 REFERENCE MANUAL CONTAIN DESCRIPTIONS OF THESE PROGRAMS.

ADX OPERATING PROCEDURES

ADX IS SUPPLIED AS AN INSTALLED TASK ON THE RSX-11 SYSTEM DEVICE. IT IS NECESSARY THAT THE HYBRID DRIVER (HY) BE LOADED IN THE RSX-11 SYSTEM FOR ADX TO COMMUNICATE WITH THE AD-10. ADX IS LOADED AND RUN AS ARE OTHER RSX-11 SYSTEM PROGRAMS, BY TYPING (IN RESPONSE TO THE MCR PROMPT) :

```
MCR>ADX [<COMMAND STRING>] <CARRIAGE RETURN>
```

THE COMMAND STRING IS OPTIONAL HERE. IF THE COMMAND STRING IS NOT ENTERED WITH THE MCR COMMAND, ADX WILL RESPOND WITH ITS OWN PROMPT :

```
MCR>ADX <CR>
ADX>
```

THE GENERAL FORMAT OF THE ADX COMMAND STRING IS DEFINED AS FOLLOWS :

```
        @<FILE SPECIFICATION>
OR,     <COMMAND> [<SWITCHES> AND/OR <OTHER PARAMETERS>]
```

THE FIRST FORM INDICATES THAT THE COMMAND STRING(S) WILL COME FROM THE SPECIFIED COMMAND FILE. THE SECOND FORM CONSISTS OF THE APPROPRIATE ADX COMMAND AND ITS RELATED SWITCHES AND/OR OPTIONAL PARAMETERS, AS DEFINED IN THE FOLLOWING PAGES OF THIS USER'S GUIDE. IF THE INDIRECT COMMAND FILE FORMAT IS USED ON THE SAME LINE AS THE MCR PROMPT, CONTROL WILL RETURN TO MCR AFTER PROCESSING THE ADX COMMANDS IN THE FILE :

```
MCR>ADX @SY:FILE.CMD;3 <CR>
MCR>
```

IF THE INDIRECT COMMAND FILE IS SPECIFIED AFTER THE ADX PROMPT, CONTROL WILL REMAIN WITH ADX FOLLOWING COMMAND FILE PROCESSING :

```
MCR>ADX <CR>
ADX>@FILE <CR>
ADX>
```

NOTICE IN THIS CASE THAT THE EQUIVALENT COMMAND FILE HAS BEEN SPECIFIED, USING DEFAULT COMMAND FILE VALUES FOR THE DEVICE ("SY:"), THE UIC NUMBER (THE DEFAULT IS THE CURRENT UIC NUMBER), THE FILE EXTENSION (".CMD"), AND THE MOST RECENT VERSION NUMBER (ASSUMING HERE THAT ";3" IS THE MOST RECENT VERSION).

IF THE USER ISSUES AN ADX COMMAND WHICH REQUIRES ADDITIONAL SPECIFIERS (SWITCHES AND/OR FILENAMES AND/OR OTHER PARAMETERS), ADX WILL PROMPT FOR THE REQUIRED SPECIFIERS :

```
MCR>ADX <CR>
ADX>DISPLAY <CR>
DIS>[OUTFILE=] [/SW[[,]...[[,]/SW]]] <CR>
```

TO EXIT FROM THIS INTERNAL COMMAND PROMPT MODE, SIMPLY TYPE A NULL LINE :

```
ADX>DISPLAY <CR>
DIS><CR>
ADX>
```

TO EXIT FROM ADX WITHOUT AFFECTING THE AD-10, SIMPLY TYPE <CONTROL-Z> :

```
ADX><CONTROL-Z>
MCR>
```

AT
--

PROTOTYPE:

AT /PM:PROC:ADDR [,OUTFILE/SW] [=INFILE/SW]
-

DESCRIPTION:

AN "AT-POINT" IS A POINT IN PROGRAM EXECUTION AT WHICH THE USER WISHES TO LOG AND/OR INSERT MULTIBUS DATA. "AT" SETS AN "AT-POINT" FOR PROCESSOR "PROC" AT PROGRAM MEMORY ADDRESS "ADDR". WHEN A "RUN" OR "CONTINUE" COMMAND IS ISSUED WITH "AT-POINTS" SET, THE AD-10 IS SINGLE STEPPED, AND ADX MONITORS THE SPECIFIED PROCESSOR'S PROGRAM COUNTER. WHEN THE PROCESSOR'S PROGRAM COUNTER REACHES THE "AT-POINT" ADDRESS, DATA IS LOGGED FROM THE DATA MULTIBUS TO THE "OUTFILE" (IF SPECIFIED) AND PLACED ON THE DATA MULTIBUS FROM THE "INFILE" (IF SPECIFIED). EITHER THE "OUTFILE" OR THE "INFILE" (OR BOTH) MUST BE SPECIFIED. THE DEFAULT EXTENSION IS .DAT FOR BOTH "OUTFILE" AND "INFILE".

SWITCHES:

/DR - FILE IS DIRECT ACCESS (DEFAULT)
/LO - FILE IS FORMATTED SEQUENTIAL (FOR OUTFILE ONLY)
/RE - DATA VALUES ARE REALS (DEFAULT)
/IN - DATA VALUES ARE DECIMAL INTEGERS
/OC - DATA VALUES ARE OCTAL INTEGERS
/FI - SPECIFIES DATA ON "FIRST" MULTIBUS TRANSACTION (DEFAULT)
/SE - SPECIFIES DATA ON "SECOND" MULTIBUS TRANSACTION
/DO - SPECIFIES DATA ON "DOUBLE" MULTIBUS TRANSACTIONS
(I.E., "FIRST" AND "SECOND" TRANSACTIONS)
/TR - TRACE AT POINTS
(THE FOLLOWING SWITCHES MAY BE USED IN PLACE OF /PM:PROC)
/MAP - PROGRAM MEMORY [/PM:1]
/DEP - PROGRAM MEMORY [/PM:2]
/ARP - PROGRAM MEMORY [/PM:3]
/NIP - PROGRAM MEMORY [/PM:4]
/COP - PROGRAM MEMORY [/PM:7]

EXAMPLE:

```
ADX>AT/PM:3:4,TI:/LO
ADX>BREAK/ARP:5
ADX>INIT
ADX>CONTINUE
0.12497
*** BREAKPOINT AT ARP:  5 ***
ADX>RESTORE
*** BREAKPOINT AT ARP:  5 RESTORED
ADX>RESTORE
*** BREAKPOINT AT ARP:  4 RESTORED
ADX>
```

NOTE:

A COMBINED TOTAL OF TEN "AT-POINTS" AND/OR "BREAKPOINTS" CAN BE SET. IF AN "AT-POINT" HAS BOTH AN "INFILE" AND AN "OUTFILE", THEN IT COUNTS AS TWO "AT-POINTS". ADX WILL NOT PERMIT BOTH AN "AT-POINT" AND A "BREAK-POINT" AT THE SAME LOCATION. EXECUTION CAN BE PREMATURELY HALTED BY CONDITIONS SET IN THE AD-10 HALT MASK REGISTER. (I.E. HLT0, HLT1, ...)

ATTACH

PROTOTYPE: ATTACH [CONSOLE #]

DESCRIPTION: GIVES USER EXCLUSIVE ACCESS TO SPECIFIED CONSOLE. IF NO CONSOLE # IS GIVEN, THE COMMAND DEFAULTS TO 0.

SWITCHES: (NONE)

EXAMPLE:
```
ADX>ATT 0
ADX>
```

BREAK

PROTOTYPE: BREAK /PM:PROC:ADDR [,...[,/PM:PROC:ADDR]]

DESCRIPTION: A BREAKPOINT SIGNIFIES A POINT IN PROGRAM EXECUTION AT WHICH THE USER WISHES TO HALT. "BREAK" SETS A BREAKPOINT FOR PROCESSOR "PROC" AT PROGRAM MEMORY ADDRESS "ADDR". WHEN A "RUN" OR "CONTINUE" COMMAND IS ISSUED WITH BREAKPOINTS SET, THE AD-10 IS SINGLE STEPPED AND ADX MONITORS THE SPECIFIED PROCESSOR'S PROGRAM COUNTER. WHEN THE PROCESSOR'S PROGRAM COUNTER REACHES THE SPECIFIED BREAKPOINT ADDRESS, THE AD-10 IS HALTED AND CONTROL IS RETURNED TO THE USER. NOTE THAT /PM:PROC CAN BE ABBREVIATED AS SHOWN BELOW:

```
/MAP - PROGRAM MEMORY [/PM:1]
/DEP - PROGRAM MEMORY [/PM:2]
/ARP - PROGRAM MEMORY [/PM:3]
/NIP - PROGRAM MEMORY [/PM:4]
/COP - PROGRAM MEMORY [/PM:7]
```

EXAMPLE:
```
ADX>HALT
ADX>BREAK/PM:7:3
ADX>; THAT SET THE BREAKPOINT AT LOCATION 3 IN THE COP ...
ADX>CONTINUE

*** BREAKPOINT AT COP:  3 ***
ADX>RESTORE

*** BREAKPOINT AT COP:  3 RESTORED
ADX>
```

NOTE: A COMBINED TOTAL OF TEN "AT-POINTS" AND/OR "BREAKPOINTS" CAN BE SET. IF AN "AT-POINT" HAS BOTH AN "INFILE" AND AN "OUTFILE", THEN IT COUNTS AS TWO "AT-POINTS". ADX WILL NOT PERMIT BOTH AN "AT-POINT" AND A "BREAK-POINT'" AT THE SAME LOCATION. EXECUTION CAN BE PREMATURELY HALTED BY CONDITIONS SET IN THE AD-10 HALT MASK REGISTER. (I.E. HLT0,HLT1, ...)

CLEAR

PROTOTYPE:

```
CLEAR [/SW:BIT:BIT:...[[,]/SW:BIT:BIT:...]]
--
```

DESCRIPTION:

CLEARS THE SPECIFIED BITS IN THE HIC-11 REGISTER AS SPECIFIED BY "/SW". IF THE SAME SWITCH IS USED MORE THAN ONCE IN THE COMMAND LINE, A "," MUST SEPARATE EACH SUCCESSIVE REFERENCE OF THAT SWITCH.

SWITCHES:

/RIC - REMOTE INTERFACE CONTROL REGISTER

- ENB - INTERRUPT ENABLE BIT (BIT 6)

/CSR - CONTROL STATUS REGISTER

- ENB - INTERRUPT ENABLE BIT (BIT 6)

/TCR - TEST CONTROL REGISTER

- TST - TEST/RUN MODE BIT (BIT 4)

/HMR - HALT MASK REGISTER

- HL0 - HALT 0 (BIT 1)
- HL1 - HALT 1 (BIT 2)
- TCC - TEST CYCLE COMPLETE (BIT 4)
- RCZ - RUN COUNT ZERO (BIT 7)
- CER - ADDRESS CONTENTION ERROR (BIT 10)
- DER - DATA CONTENTION ERROR (BIT 11)
- TER - TIMING ERROR: DATA MEMORY (BIT 12)
- PER - PARITY ERROR: DATA MEMORY (BIT 13)
- AER - ARITHMETIC ERROR (BIT 14)
- ERR - ERROR ("OR" OF ALL ERRORS) (BIT 15)

/IMR - INTERRUPT MASK REGISTER

- (SAME BITS AS /HMR ABOVE)

EXAMPLE:

```
ADX>DISPLAY/CSR

CSR HAS VALUE    100
ADX>CLEAR/CSR:ENB
ADX>; THAT CLEARED THE ENB BIT IN THE CSR
ADX>DISPLAY/CSR

CSR HAS VALUE      0
ADX>
```

CONSOLE

PROTOTYPE: CONSOLE [CONSOLE #]

DESCRIPTION: ALLOWS USER TO SWITCH FROM ONE CONSOLE TO ANOTHER CONSOLE. CONSOLE # DEFAULTS TO 0 WHEN NOT SPECIFIED. THE CONSOLE SELECTED MUST BE ATTACHED.

SWITCHES: (NONE)

EXAMPLE:

```
ADX>ATT 0
ADX>ATT 1
ADX>CONSOLE 0
   .
   .      WORK WITH CONSOLE 0
   .
ADX>CONSOLE 1
   .
   .      WORK WITH CONSOLE 1
   .
ADX>
```

CONTINUE

PROTOTYPE: CONTINUE

DESCRIPTION: STARTS THE AD-10 FROM ITS CURRENT STATE.

SWITCHES: (NONE)

EXAMPLE:

```
ADX>INIT
ADX>; THAT INITIALIZED THE AD-10
ADX>CONTINUE
ADX>; THAT STARTED THE AD10
ADX>HALT
ADX>; THAT STOPPED THE AD-10
ADX>CONTINUE
ADX>; THAT RESTARTED THE AD-10 FROM THE HALT POINT
ADX>
```

DETACH

PROTOTYPE: DETACH [CONSOLE #]

DESCRIPTION: FREES THE SPECIFIED CONSOLE FOR GENERAL USE. CONSOLE # DEFAULTS TO 0 IF NOT GIVEN IN COMMAND.

SWITCHES: (NONE)

EXAMPLE:
```
ADX>DE 0
ADX>
```

DISPLAY

PROTOTYPE: DISPLAY [OUTFILE=] [/SW[[,]...[[,]/SW]]]

DESCRIPTION: DISPLAYS THE REGISTER(S) AND/OR MEMORY LOCATION(S) SPECIFIED BY "/SW" TO THE "OUTFILE" (OR "TI:" BY DEFAULT). MULTIPLE REFERENCES TO THE SAME SWITCH MUST BE SEPARATED BY A ",". THE DEFAULT EXTENSION IS .LST FOR OUTFILE.

SWITCHES:
```
/TCR - TEST CONTROL REGISTER
/TSH - TEST/SHUTDOWN/HISTORY COUNTERS
/TBS - TEST BLOCK ADDRESS REGISTERS
/TAS - TEST ADDRESS REGISTERS
/TDS - TEST DATA REGISTERS
/SCS - SHUTDOWN/RESTART CONDITION REGISTERS
/SDS - SHUTDOWN/RESTART DATA REGISTERS
/CSR - CONTROL STATUS REGISTERS
/EHS - ERROR HALT STATUS REGISTER
/HMR - HALT MASK REGISTER
/IMR - INTERRUPT MASK REGISTER
/RCR - RUN COUNT REGISTER
/BAR - BLOCK ADDRESS REGISTER
/RIC - REMOTE INTERFACE CONTROL REGISTER
/PCS - PROGRAM COUNTERS
/PSS - PROCESSOR STATUS REGISTERS
/HBS - HISTORY BLOCK ADDRESS REGISTERS
/HAS - HISTORY ADDRESS REGISTERS
/HDS - HISTORY DATA REGISTERS
/PM:PROC - PROGRAM MEMORY
(THE FOLLOWING 5 SWITCHES CAN BE USED IN PLACE OF /PM:PROC)
/MAP - PROGRAM MEMORY [/PM:1]
/DEP - PROGRAM MEMORY [/PM:2]
/ARP - PROGRAM MEMORY [/PM:3]
/NIP - PROGRAM MEMORY [/PM:4]
/COP - PROGRAM MEMORY [/PM:7]
/DM:PAGE - DATA MEMORY
/BW  - BUS WINDOW
```

```
/GR  - COP'S GENERAL REGISTERS
/IR  - MAP'S INDEX REGISTERS
/XR  - DEP'S X REGISTERS
/TR  - ARP'S TEMPORARY REGISTERS
/RR  - ARP'S RESULT REGISTER
/FR  - NIP'S "F" REGISTERS
/DR  - NIP'S "D" REGISTER
/HI  - EQUIVALENT TO "/HA/HB/HD"
/TE  - EQUIVALENT TO "/TA/TB/TD"
/DP  - DATA MEMORY PAGES
/DL  - WORDS/PAGE LIMIT
/PF  - PROGRAM MEMORY FIELDS
/PL  - PROGRAM MEMORY WORD LIMIT
/AC  - NUMBER OF AD-10 CONSOLES
```

NOTE: SPECIFIC REGISTERS OR LOCATIONS CAN BE SPECIFIED BY APPENDING ":N" OR ":N:M" TO A SWITCH FOR ONE OR A RANGE OF REGISTERS OR LOCATIONS. ONLY THE FIRST TWO CHARACTERS OF THE SWITCH ARE REQUIRED.

EXAMPLE:

```
ADX>; DISPLAY ARP PM LOCATIONS 0-7
ADX>DISPLAY/ARP::7

ARP :     0 **        0  36000      0       0       0
ARP :     1 **        0 104000      0       0       0
ARP :     2 **        0 110000 100400       0       0
ARP :     3 **      104      0      0       0       0
ARP :     4 **        0      0      0       0   40377
ARP :     5 **        0 140204  40400  140377   40604
ARP :     6 **        0      0      0       0       0
ARP :     7 **        0      0      0       0       0
ADX>
```

DUMP

PROTOTYPE: DUMP [OUTFILE]

DESCRIPTION: DUMPS ALL HIC-11 REGISTERS TO "OUTFILE" ("TI:" BY DEFAULT). THE DEFAULT EXTENSION IS .LST FOR OUTFILE.

SWITCHES: (NONE)

EXAMPLE:

```
ADX>DUMP

RIC:      0
CSR: 100000
EHS:  10042
HMR:     40
IMR:      0
RCR:  77373
PCS:      0      0      0      0      0      0     41
PSS:      0      0      0      0      0      0   7776
HBS:      0 100021  20000  20000  20000  20000  20000  20000
      20000 120035  20000 120021  20000  20000  20000  20000
HAS:      0    252      0      0      0      0      0      0
          0    212      0    251      0      0      0      0
HDS:      0 177777      0      0      0      0      0      0
          0      0      0 177777      0      0      0      0
TCR: 100000
TSH:     60
TBS: 172073   1062  63146  10020  20012      1   2026  30021
      50105 120052  62567 101040  10005  43114     12  24145
TAS: 110166  31442  62106      0 120042 110001  10022  10420
       2005 121030  33163  20002    445  43045   2052  64403
TDS:   3115  70643    156    421  20040     20  10120  10423
      10004    252  70525  21040  20001  62166 100312  10027
SCS:      0      0      0      0      0      0      0      0
          0      0      0      0      0      0      0      0
SDS: 177777      0      0      0      0      0      0      0
          0      0      0      0      0      0      0      0
BAR:   3420
ADX>
```

EXIT

PROTOTYPE: EXIT

DESCRIPTION: EXITS FROM THE AD-10 EXECUTIVE AFTER HALTING THE AD-10. NOTE: TO EXIT FROM THE EXECUTIVE WITHOUT HALTING THE AD-10, SIMPLY TYPE <CONTROL Z>.

SWITCHES: (NONE)

EXAMPLE:

```
ADX>EXIT

MCR>
```

```
                FLOATING
                --------

PROTOTYPE:      FLOATING [OCTAL SCALED FRACTION]
---------       --
DESCRIPTION:    THE 16 BIT OCTAL REPRESENTATION OF A SCALED FRACTION
-----------     IS CONVERTED AND PRINTED AS A FLOATING POINT NUMBER.

SWITCHES:       (NONE)
--------
EXAMPLE:        ADX>FLOATING 10000
-------         FLOATING: 0.12500
                ADX>FLOATING 170000
                FLOATING:-0.125
                ADX>FLOATING 40065
                FLOATING: 0.50162
                ADX>


                FRACTION
                --------

PROTOTYPE:      FRACTION [FLOATING POINT NUMBER]
---------       --

DESCRIPTION:    THE FLOATING POINT REPRESENTATION OF A SCALED FRACTION
-----------     IS CONVERTED AND PRINTED AS A 16 BIT OCTAL SCALED
                FRACTION.

SWITCHES:       (NONE)
--------

EXAMPLE:        ADX>FRACTION -.5
-------            OCTAL: 140000
                ADX>FRACTION .5
                   OCTAL: 40000
                ADX>FRACTION 0.50162
                   OCTAL: 40065
                ADX>


                HALT
                ----

PROTOTYPE:      HALT
---------       -
DESCRIPTION:    HALTS THE AD-10.
-----------
SWITCHES:       (NONE)
--------
EXAMPLE:        ADX>HALT
-------         ADX>CONTINUE
                ADX>; THAT RESUMES EXECUTION FROM THE HALT POINT
                ADX>
```

HISTORY

PROTOTYPE:
HISTORY [OUTFILE]
--

DESCRIPTION:
DISPLAYS THE DATA IN THE HISTORY BUFFER REGISTERS TO "OUTFILE" ("TI:" BY DEFAULT) IN AN EASILY READABLE FORMAT, DISPLAYING THE ADDRESS AND DATA BUS VALUES FOR EACH REGISTER, AS WELL AS WHAT HAPPENED: WHETHER A DM PARITY ERROR OCCURRED, IF THIS WAS A READ OR WRITE, THE STATUS OF THE MULTIBUS CONTROL LINES, AND WHETHER ANY OTHER ERRORS OCCURRED. THE DEFAULT EXTENSION IS .LST FOR OUTFILE.

SWITCHES:
(NONE)

EXAMPLE:
```
ADX>HISTORY

                                    DRCCCAPTDC
                      DATA BUS      1/000EEEEE
R# ADDR BUS   OCTAL FLOATING        6W210RRRRR
 0 0000000       0  0.00000        ..........
 1 0010652  177777 -0.00003        1W........
 2 0000000       0  0.00000        ...R......
 3 0000000       0  0.00000        ...R......
 4 0000000       0  0.00000        ...R...E..
 5 0000000       0  0.00000        ...R......
 6 0000000       0  0.00000        ...R......
 7 0000000       0  0.00000        ...R......
 8 0000000       0  0.00000        ...R......
 9 0016612       0  0.00000        .W.R......
10 0000000       0  0.00000        ...R......
11 0010651  177777 -0.00003        .W.R......
12 0000000       0  0.00000        ...R......
13 0000000       0  0.00000        ...R......
14 0000000       0  0.00000        ...R......
15 0000000       0  0.00000        ...R......
ADX>
```

IDENTIFY

PROTOTYPE: IDENTIFY

DESCRIPTION: DISPLAYS THE CURRENT VERSION NUMBER OF THE EXECUTIVE.

SWITCHES: (NONE)

EXAMPLE:

```
MCR>ADX
ADX>IDENTIFY
*** AD-10 EXECUTIVE HERE (26-SEP-78) ***
ADX>
```

INIT

PROTOTYPE: INIT

DESCRIPTION: INITIALIZES THE AD-10 (I.E., SETS THE "INT" BIT IN THE HIC CSR REGISTER).

SWITCHES: (NONE)

EXAMPLE:

```
ADX>; INITIALIZE EVERYTHING FOR A STARTUP ...
ADX>INIT
ADX>; NOW START THE AD-10 ...
ADX>CONTINUE
ADX>
```

LOAD

PROTOTYPE:

```
LOAD [LOADFILE[/SW] [,...[,LOADFILE[/SW]]]]
-
```

DESCRIPTION:

THE SPECIFIED FILES ARE LOADED INTO SPECIFIED LOCATIONS IN AD-10 PROGRAM AND/OR DATA MEMORY. THERE ARE BASICALLY TWO TYPES OF FILES WHICH MAY BE LOADED INTO THE AD-10: LOAD MODULES AND DIRECT ACCESS FUNCTION DATA FILES. LOAD MODULES ARE CREATED BY THE CROSS-ASSEMBLER OR BY THE "SAVE" COMMAND AND ARE LOADED INTO AD-10 PROGRAM MEMORY. DIRECT ACCESS FUNCTION DATA FILES CAN BE CREATED ON THE HOST COMPUTER AND ARE LOADED INTO THE AD-10 MEMORY.

SWITCHES:

/MO - THE FILE IS A LOAD MODULE GENERATED BY THE ASSEMBLER OR BY THE "SAVE" COMMAND (DEFAULT FILE TYPE). THE LOAD MODULE CONTAINS ALL NECESSARY INFORMATION FOR THE LOADER ROUTINE REGARDING WHERE IT IS TO BE LOADED AND THE DATA'S FORMAT. NO OTHER SWITCHES ARE NECESSARY WITH "/MO". DEFAULT EXTENSION IS .MOD.

/AL:PAGE:WORD - SPECIFIES A DIRECT ACCESS DATA FILE TO BE LOADED INTO DATA MEMORY IN THE "ALIGNED" MODE STARTING AT THE SPECIFIED ADDRESS (I.E., THE FIRST DATA VALUE IS LOADED AT ADDRESS "PAGE:WORD", THE SECOND DATA VALUE IS LOADED AT ADDRESS "PAGE+1:WORD", THE THIRD DATA VALUE IS LOADED AT ADDRESS "PAGE:WORD+1, ...ETC.). EACH RECORD OF THE FILE CONTAINS A TWO WORD DATA VALUE WHICH IS CONVERTED TO SCALED FRACTION FORMAT PRIOR TO LOADING UNLESS OTHERWISE SPECIFIED BY A "/IN" OR "/RI" SWITCH. DEFAULT EXT IS .DAT

/UN:PAGE:WORD - SAME AS /AL EXCEPT THAT THE DATA IS LOADED IN THE "UNALIGNED" MODE (I.E., SUCCESSIVE DATA VALUES ARE LOADED INTO SUCCESSIVE MEMORY LOCATIONS STARTING AT "PAGE:WORD" AS FOLLOWS: "PAGE:WORD","PAGE:WORD+1", "PAGE:WORD+2",...ETC.). DEFAULT EXTENSION IS .DAT

/IN - USED WITH "/AL" OR "/UN" TO INDICATE THAT EACH RECORD OF THE FILE CONTAINS A ONE WORD DATA VALUE WHICH IS TO BE LOADED DIRECTLY INTO THE AD-10.

/RI - USED WITH "/AL" OR "/UN" TO INDICATE THAT EACH RECORD OF THE FILE CONTAINS A TWO WORD REAL VALUE WHICH IS TO BE CONVERTED TO INTEGER PRIOR TO LOADING.

/RS - USED WITH "/AL" OR "/UN" TO INDICATE THAT EACH RECORD OF THE FILE CONTAINS A TWO WORD REAL VALUE WHICH IS TO BE CONVERTED TO SCALED FRACTION FORMAT PRIOR TO LOADING (DEFAULT).

EXAMPLE:

```
ADX>; BRING IN ARP LOAD MODULE ...
ADX>LOAD ARP.SAV/MO
ADX>; LOAD DATA MEMORY
ADX>LOAD A.DAT;1/AL:0:0, B.DAT;3/UN:2:0/RI, C.DAT/UN:3:0/IN
ADX>
```

MODIFY

PROTOTYPE:
MODIFY [/SW[[,]...[[,]/SW]] [=INFILE]]

DESCRIPTION:
THE REGISTERS AND/OR MEMORY LOCATIONS DENOTED BY THE SPECIFIED SWITCHES ARE DISPLAYED ONE AT A TIME, ALLOWING THE USER TO MODIFY THE CURRENT VALUE OR LEAVE IT UNCHANGED BY SIMPLY TYPING A CARRIAGE RETURN. INPUT VALUES CAN COME FROM THE USER AT "TI:" (BY DEFAULT) OR FROM THE "INFILE" (IF SPECIFIED). THE DEFAULT EXTENSION IS .LST FOR INFILE.

SWITCHES:
THE SAME AS FOR THE "DISPLAY" COMMAND.

EXAMPLE:

```
ADX>MODIFY/COP::3
COP :     0 **    6427  137673
           / 0,0
COP :     1 **   42412   43530
           /
COP :     2 **   73435  157473
           / 0,0
COP :     3 **    7421  155126
           /
ADX>DIS/PM:7::2
COP :     0 **       0       0       0       0       0
COP :     1 **   42412   43530       0       0       0
COP :     2 **       0       0       0       0       0
ADX>
```

RESTORE

PROTOTYPE:
RESTORE [/PM:PROC:ADDR [,...[,/PM:PROC:ADDR]]]

DESCRIPTION:
RESTORE DELETES THE SPECIFIED LIST OF "BREAKPOINTS" AND/OR "AT-POINTS". IF NO LIST IS SPECIFIED, THE LAST "BREAKPOINT" OR "AT-POINT" WHICH THE USER HAS SET IS DELETED. NOTE THAT /PM:PROC CAN BE ABBREVIATED THE SAME AS WAS SHOWN FOR THE "AT" AND "BREAK" COMMANDS.

EXAMPLE:

```
ADX>; INSERT AN AT POINT IN ARP PROGRAM
ADX>AT/PM:3:3,TI:/LO
ADX>; INSERT A BREAKPOINT IN COP PROGRAM
ADX>BREAK/COP:5
ADX>; INITIALIZE THE AD-10
ADX>INIT
ADX>; START THE AD-10
ADX>CONTINUE
0.72535
*** BREAKPOINT AT COP:  5 ***
ADX>; RESTORE THE BREAK AND AT POINTS
ADX>RESTORE/PM:3:3,/COP:5
*** BREAKPOINT AT ARP:  3 RESTORED
*** BREAKPOINT AT COP:  5 RESTORED
ADX>
```

```
                RUN
                ---

PROTOTYPE:      RUN [LOADFILE[/SW] [,...[,LOADFILE[/SW]]]]
---------       -

DESCRIPTION:    THE AD-10 IS INITIALIZED (SEE "INIT"), THE LOADFILES ARE
-----------     LOADED (IF SPECIFIED, SEE "LOAD"), AND THE AD-10
                IS STARTED.

SWITCHES:       THE SAME AS FOR THE "LOAD" COMMAND.
--------

EXAMPLE:        ADX>; INITIALIZE THINGS
-------         ADX>INIT
                ADX>;LOAD THE PROGRAMS AND DATA, AND BEGIN EXECUTION
                ADX>RUN AD10.SAV/MO, A./AL:0:0, B.;2/UN:2:0/RI, C.DAT/UN:3:0/IN
                ADX>


                SAVE
                ----

PROTOTYPE:      SAVE OUTFILE[/DA][=/SW[[,]...[[,]/SW]]]
---------       --

DESCRIPTION:    THE SPECIFIED DATA AND/OR PROGRAM MEMORY LOCATIONS ARE
-----------     WRITTEN TO THE "OUTFILE" IN AD-10 ASSEMBLER LOAD MODULE
                FORMAT (WITH A DEFAULT EXT OF .MOD).  "OUTFILE" CAN THEN
                BE LOADED USING THE "LOAD" COMMAND (I.E. LOAD OUTFILE/MO).
                IF /DA IS SPECIFIED, THE LOCATIONS ARE "DISASSEMBLED" AND
                WRITTEN TO THE "OUTFILE" IN AD-10 ASSEMBLER SOURCE FORMAT
                (WITH A DEFAULT FILE EXT OF .ASM).

SWITCHES:       /PM:PROC - SAVE THE PROGRAM MEMORY OF PROCESSOR "PROC".
--------        /DM:PAGE - SAVE THE SPECIFIED "PAGE" OF DATA MEMORY.
                (THE FOLLOWING 5 SWITCHES MAY BE USED IN PLACE OF /PM:PROC)
                /MAP     - PROGRAM MEMORY [/PM:1]
                /DEP     - PROGRAM MEMORY [/PM:2]
                /ARP     - PROGRAM MEMORY [/PM:3]
                /NIP     - PROGRAM MEMORY [/PM:4]
                /COP     - PROGRAM MEMORY [/PM:7]
                /DA      - DISASSEMBLE (PRODUCE AD-10 ASSEMBLER SOURCE)

                NOTE: A LOCATION OR RANGE OF LOCATIONS CAN BE SPECIFIED
                BY APPENDING ":N" OR ":N:M" TO THE SWITCH SPECIFICATION.

EXAMPLE:        ADX>HALT
-------         ADX>; SAVE LOCATIONS 0-17 OF COP PM IN FILE COP.SAV
                ADX>SAVE COP.SAV=/COP::17
                ADX>; SAVE PAGES 2,3,4 OF DATA MEMORY IN FILE DM24UN.SAV
                ADX>SAVE DM24UN.SAV=/DM:2:4
                ADX>
```

SET

PROTOTYPE:
SET [/SW:BIT:BIT:...[[,]/SW:BIT:BIT:...]]

DESCRIPTION:
SETS THE SPECIFIED BITS IN THE HIC-11 REGISTER DENOTED BY THE "/SW". IF THE SAME SWITCH IS USED MORE THAN ONCE IN THE COMMAND LINE, A "," MUST SEPARATE EACH SUCCESSIVE REFERENCE OF THAT SWITCH.

SWITCHES:
THE SAME AS FOR THE "CLEAR" COMMAND.

EXAMPLE:

```
ADX>DISPLAY/RIC

RIC HAS VALUE      0
ADX>SET/RIC:ENB
ADX>; THAT SET THE INTERRUPT ENABLE BIT IN THE RIC (BIT 6)
ADX>DISPLAY/RIC

RIC HAS VALUE    100
ADX>
```

STEP

PROTOTYPE:
STEP [[#]N]

DESCRIPTION:
THE AD-10 IS STEPPED N+1 INSTRUCTION CYCLES (WHERE 0<=N<=177777 (OCTAL), BUT ONLY 0<=N<=32767 (DECIMAL)). IF "N" IS NOT SPECIFIED, A SINGLE STEP IS PERFORMED (I.E., N=0 IS ASSUMED). "N" MAY BE SPECIFIED IN EITHER DECIMAL INTEGER (DEFAULT) OR OCTAL (BY PRECEEDING N WITH "#").

SWITCHES:
(NONE)

EXAMPLE:

```
ADX>INIT
ADX>DIS/PCS:7

PCS(  7) HAS VALUE      0
ADX>STEP #12
ADX>DIS/PCS:7

PCS(  7) HAS VALUE     13
ADX>
```

TEST

PROTOTYPE: TEST [OUTFILE]

DESCRIPTION: DISPLAYS THE DATA IN THE TEST BUFFER REGISTERS TO "OUTFILE" ("TI:" BY DEFAULT) IN THE SAME FORMAT AS "HISTORY". THE DEFAULT EXTENSION IS .LST FOR OUTFILE.

SWITCHES: (NONE)

EXAMPLE:

```
ADX>HALT
ADX>TEST
                                          DRCCCAPTDC
                        DATA BUS          1/000EEEEE
R# ADDR BUS       OCTAL FLOATING          6W210RRRRR
 0 1035566         3115  0.04922           .W.RM..E..
 1 0431042        70643  0.88779           ......EE..
 2 1463106          156  0.00336           ...R.EE..E
 3 0010000          421  0.00833           ....M.....
 4 0005042        20040  0.25098           1..R..E...
 5 0000401           20  0.00049           .......E..
 6 1013022        10120  0.12744           .......E..
 7 0010420        10423  0.13339           ...RM..E..
 8 0042405        10004  0.12512           ....M....E
 9 0025030          252  0.00519           .W.R..E...
10 1273563        70525  0.88541           ...R..EE.E
11 0420002        21040  0.26660           .W....E...
12 0002445        20001  0.25003           ....M.....
13 1446045        62166  0.78485           .....E...E
14 0005052       100312 -0.99384           .........E
15 2062403        10027  0.12570           ...R.EE.E.
ADX>
```

TRACE

PROTOTYPE: TRACE [OUTFILE]

DESCRIPTION: TRACE THE EXECUTION OF THE AD-10 WITH A PRINTED LOG TO "OUTFILE" (OR "TI:" BY DEFAULT), DISPLAYING THE BUS TRANSACTIONS FOR EACH HALF OF THE INSTRUCTION CYCLE, AND THE PROCESSOR PROGRAM COUNTERS AND STATUS WORDS FOR EACH INSTRUCTION CYCLE. EXECUTION CAN BE PREMATURELY HALTED BY CONDITIONS SET IN THE AD-10 HALT MASK REGISTER (I.E. HLT0, HLT1, ...). THE DEFAULT EXTENSION IS .LST FOR OUTFILE.

SWITCHES: (NONE)

EXAMPLE:

```
ADX>HALT
ADX>BREAK/PM:7:5
ADX>TRACE
ADX>INIT
ADX>CONTINUE
```

```
                                  DRCCCAPTDC     COP      MAP      DEP      ARP      NIP
                    DATA BUS      1/000EEEEE     F-PC     F-PC     F-PC     F-PC     F-PC
 T# ADDR BUS   OCTAL FLOATING     6W210RRRRR     S-PS     S-PS     S-PS     S-PS     S-PS

  F 0000000       0  0.00000     ..........         1        0        0        0        0
  S 0000000       0  0.00000     ..........      1400   140400      400      400      400

  F 0043530       0  0.00000     .W........         2        0        0        0        0
  S 0000000       0  0.00000     ..........    141400   140400     1400      400      400

  F 0000000       0  0.00000     .....E....         3        0        1        0        0
  S 0000000  157473 -0.25601     ..........     41400   140400   141406      400      400

  F 0000000       0  0.00000     ..........         4        0        1        0        0
  S 0000000       0  0.00000     ..........     41400   140400    41405      400      400

  F 0266537       0  0.00000     .W........         5      537        1        0        0
  S 0000000       0  0.00000     ..........    141400   140400    41404      400      400

   *** BREAKPOINT AT 7:  5 ***
  ADX>
```

UNTRACE

PROTOTYPE: UNTRACE

DESCRIPTION: TURNS OFF TRACE MODE.

SWITCHES: (NONE)

EXAMPLE:

```
ADX>; AD-10 CURRENTLY IN TRACE MODE ...
ADX>HALT
ADX>UNTRACE
ADX>CONTINUE
ADX>; WILL RESUME EXECUTION WITHOUT TRACE ...
ADX>
```

ZERO

PROTOTYPE: ZERO [/SW[[,]...[[,]/SW]]] [=MASK]

DESCRIPTION: ZEROS AD-10 PROGRAM MEMORYS, DATA MEMORY, AND PROGRAM COUNTERS. IF THE OCTAL MASK IS SPECIFIED, THE LOCATIONS WILL BE LOADED WITH THIS VALUE INSTEAD OF ZERO.

SWITCHES: THE SAME AS FOR THE "DISPLAY" COMMAND, EXCEPT THAT ":PAGE" IS NOT REQUIRED FOLLOWING "/DM" (I.E., "/DM" WILL ZERO ALL DATA MEMORY), AND ":PROC" IS NOT REQUIRED FOLLOWING "/PM" (I.E., "/PM" WILL ZERO ALL PROGRAM MEMORY).

EXAMPLE:

```
ADX>HALT
ADX>ZERO/PM:3:0:2
ADX>DISPLAY/PM:3:0:2,/RR

ARP :     0 **       0      0      0      0      0
ARP :     1 **       0      0      0      0      0
ARP :     2 **       0      0      0      0      0

RR HAS VALUE      0
ADX>ZERO/RR=177777
ADX>DISPLAY/RR

RR HAS VALUE 177777
ADX>
```

@
-

PROTOTYPE: @INFILE
-

DESCRIPTION: SPECIFIES AN INDIRECT COMMAND FILE (DEFAULT EXT IS .CMD).

SWITCHES: (NONE)

EXAMPLE:
```
MCR>ADX
ADX>@COMFIL
ADX>; THAT EXECUTED THE ADX COMMANDS IN
ADX>; THE FILE COMFIL.CMD
ADX>
```

;
-

PROTOTYPE: ; [COMMENT]
-

DESCRIPTION: SPECIFIES A COMMENT, WHICH IS NOT ECHOED TO THE TERMINAL WHEN ENCOUNTERED IN A COMMAND FILE. ADX IGNORES THE LINE.

SWITCHES: (NONE)

EXAMPLE:
```
ADX>; THIS IS A NON-ECHO COMMENT
ADX>
```

!
-

PROTOTYPE: ! [COMMENT]
-

DESCRIPTION: SPECIFIES A COMMENT, WHICH IS ECHOED TO THE TERMINAL WHEN ENCOUNTERED (E.G.,AS IN A COMMAND FILE), BUT IGNORED BY ADX. THE "!" MUST BE FOLLOWED BY AT LEAST ONE BLANK.

SWITCHES: (NONE)

EXAMPLE:
```
ADX>! THIS IS AN ECHO COMMENT (! BECOMES A BLANK)
  THIS IS AN ECHO COMMENT (! BECOMES A BLANK)
ADX>
```

SUMMARY OF ADX COMMANDS

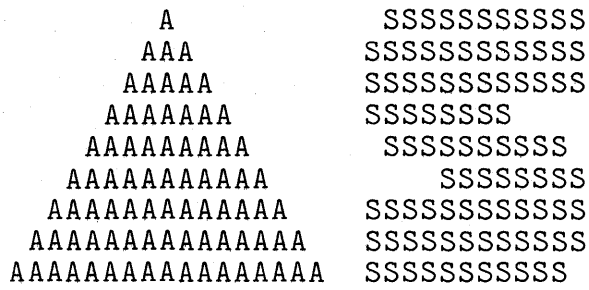
```
AT        /PM:PROC:ADDR [,OUTFILE/SW] [=INFILE/SW]
-
ATTACH    [CONSOLE #]
---
BREAK     /PM:PROC:ADDR [,...[,/PM:PROC:ADDR]]
-
CLEAR     [/SW:BIT:BIT:...[[,]/SW:BIT:BIT:...]]
--
CONSOLE   [CONSOLE #]
-----
CONTINUE
-
DETACH    [CONSOLE #]
--
* DISPLAY  [OUTFILE=] [/SW[[,]...[[,]/SW]]]
-
DUMP      [OUTFILE]
--
EXIT
--
* FLOATING [OCTAL SCALED FRACTION]
--
* FRACTION [FLOATING POINT NUMBER]
--
HALT
-
HISTORY   [OUTFILE]
--
IDENTIFY
--
INIT
-
LOAD      [LOADFILE[/SW] [,...[,LOADFILE[/SW]]]]
-
* MODIFY   [/SW[[,]...[[,]/SW]] [=INFILE]]
-
RESTORE   [/PM:PROC:ADDR [,...[,/PM:PROC:ADDR]]]
--
RUN       [LOADFILE[/SW] [,...[,LOADFILE[/SW]]]]
-
* SAVE     OUTFILE[/DA][=/SW[[,]...[[,]/SW]]]
--
SET       [/SW:BIT:BIT:...[[,]/SW:BIT:BIT:...]]
--
STEP      [[#]N]
--
```

SUMMARY OF ADX COMMANDS (CONT.)

	TEST	[OUTFILE]
	TRACE	[OUTFILE]
	UNTRACE	
*	ZERO	[/SW[[,]...[[,]/SW]]] [=MASK]
	@	INDIRECT FILE SPECIFIER
	;	COMMENT SPECIFIER (NON-ECHO)
	!	COMMENT SPECIFIER (ECHO)

(ONLY THE UNDERLINED PORTIONS OF THE COMMANDS ARE REQUIRED).

* NOTE : IF THE STARRED COMMANDS ARE NOT FOLLOWED BY A COMMAND STRING, ADX WILL PROMPT THE USER FOR A COMMAND STRING OF THE FORM SHOWN. TO EXIT FROM THIS INTERNAL COMMAND PROMPT MODE, SIMPLY TYPE A NULL LINE. THE "SAVE" COMMAND REQUIRES THE "OUTFILE" TO ALSO BE SPECIFIED TO GET THE INTERNAL COMMAND PROMPT.



THE
AD-10 / PDP-11
CROSS-ASSEMBLER
USER'S MANUAL

TABLE OF CONTENTS PAGE

TERMS AND CONVENTIONS

ASM	NAME OF THE AD-10 / PDP-11 CROSS-ASSEMBLER PROGRAM
DEC	DIGITAL EQUIPMENT CORPORATION, MAKERS OF THE PDP-11 COMPUTER
RSX-11	DEC'S REAL-TIME OPERATING SYSTEM EXECUTIVE FOR THE PDP-11
MCR>	PROMPT GENERATED BY THE RSX-11 MONITOR CONSOLE ROUTINE
ASM>	PROMPT GENERATED BY THE ASSEMBLER
ADX>	PROMPT GENERATED BY THE AD-10 EXECUTIVE PROGRAM
@	INDIRECT FILE SPECIFIER (REF. CHAPTER 6 OF THE RSX-11M OPERATOR'S PROCEDURES MANUAL)
/	ASM COMMAND STRING SWITCH DELIMITER
:	ASM COMMAND STRING SWITCH VALUE DELIMITER, ALSO PAGE:WORD DELIMITER (E.G., '0:4095')
< >	ANGLE BRACKETS ARE USED TO ENCLOSE THE NAME OF A SYNTACTIC ELEMENT OR CLASS OF ELEMENTS
[ ]	BRACKETS ARE USED TO ENCLOSE OPTIONAL SYNTACTIC ELEMENTS IN COMMAND STRING EXAMPLES.
CR	CARRIAGE RETURN (RETURN)
CONTROL-Z CTRL-Z	EOF CHARACTER GENERATED BY SIMULTANEOUSLY DEPRESSING THE "CONTROL" AND THE "Z" KEYS. USED TO EXIT FROM ASM AND RETURN TO MCR.
CONTROL-U CTRL-U	"CONTROL" AND "U" KEYS. USED TO DELETE A LINE.
RUBOUT	USED TO DELETE A CHARACTER.
CONTROL-S CTRL-S	"CONTROL" AND "S" KEYS. USED TO TEMPORARILY STOP THE OUTPUT TO THE TERMINAL WHEN THE DISPLAY IS SCROLLING.
CONTROL-Q CTRL-Q	"CONTROL" AND "Q" KEYS. USED TO RESTART THE OUTPUT AFTER CONTROL-S HAS BEEN TYPED.
!	COMMENT DELIMITER
<SPACE>	SEPARATOR BETWEEN SYNTACTIC ELEMENTS IN A STATEMENT
<TAB>	SEPARATOR BETWEEN SYNTACTIC ELEMENTS IN A STATEMENT
,	SEPARATOR BETWEEN OPERANDS
;	SEPARATOR BETWEEN OPCODE/OPERAND GROUPS
#	FIRST CHARACTER IN VARIABLE SYMBOL NAME
$	FIRST CHARACTER IN PREDEFINED SYMBOL NAME
' '	TITLE DELIMITERS, ALSO VALUE DELIMITERS
TOKEN	A TERM REFERRING TO A SYNTACTIC ELEMENT WITHIN AN ASSEMBLER STATEMENT (E.G., A SYMBOL, ARITHMETIC OPERATOR, OPCODE, ETC.)

THE CROSS-ASSEMBLER PROGRAM

THE AD-10/PDP-11 CROSS-ASSEMBLER PROGRAM (OR "ASSEMBLER") PROCESSES THE AD-10 ASSEMBLY LANGUAGE SOURCE STATEMENTS FROM A SOURCE FILE ON THE PDP-11 HOST COMPUTER. THIS PROCESSING INVOLVES THE TRANSLATION OF THE SOURCE STATEMENTS INTO AD-10 MACHINE LANGUAGE, THE ASSIGNMENT OF STORAGE LOCATIONS (AD-10 PROGRAM MEMORY, DATA MEMORY, OR REGISTERS), AND THE PERFORMANCE OF AUXILIARY FUNCTIONS AS DESIGNATED BY THE PROGRAMMER. THE OUTPUT FROM THE ASSEMBLER CONSISTS OF AN OBJECT FILE AND A LISTING FILE. THE OBJECT FILE CONTAINS THE AD-10 MACHINE LANGUAGE TRANSLATION OF THE SOURCE PROGRAM, IN A FORM WHICH MAY THEN BE LOADED INTO THE AD-10 AND RUN USING THE AD-10 EXECUTIVE PROGRAM (ADX). THE LISTING FILE CONSISTS OF A LISTING OF THE SOURCE STATEMENTS WITH THE GENERATED OBJECT CODE AND ANY ERRORS DETECTED AND AN OPTIONAL MULTIBUS TIMING DIAGRAM. THE PROGRAMMER CAN CONTROL THE FORMAT AND CONTENT OF THE LISTING VIA THE APPROPRIATE ASSEMBLER DIRECTIVES. POTENTIAL OR ACTUAL ERRORS INVOLVING ASSEMBLY LANGUAGE SYNTAX OR USAGE ARE DETECTED BY THE ASSEMBLER AND ARE INCLUDED IN THE LISTING. THE ASSEMBLER ALSO CHECKS FOR ERRORS IN THE TIMING RELATIONSHIPS OF THE VARIOUS AD-10 PROCESSORS' MULTIBUS ACCESSES AND INDICATES POTENTIAL PROBLEMS ON THE LISTING OR ON THE TIMING DIAGRAM.

THE ASSEMBLY LANGUAGE

THE AD-10 ASSEMBLY LANGUAGE CONSISTS OF A COLLECTION OF MNEMONIC SYMBOLS WHICH REPRESENT :

1. AD-10 MACHINE LANGUAGE INSTRUCTIONS
2. AD-10 ASSEMBLER DIRECTIVES

ALL VALID AD-10 MACHINE LANGUAGE INSTRUCTIONS HAVE CORRESPONDING MNEMONIC SYMBOLS WHICH GENERATE THE APPROPRIATE MACHINE LANGUAGE CODE FOR EACH AD-10 PROCESSOR. REFER TO THE APPROPRIATE AD-10 PROCESSOR MANUAL FOR DESCRIPTIONS OF THE INSTRUCTIONS.

ASSEMBLER DIRECTIVES (OR "PSEUDO-OP'S") SPECIFY AUXILIARY FUNCTIONS WHICH THE PROGRAMMER REQUESTS THE ASSEMBLER TO PERFORM. THEY GENERALLY RESULT IN NO MACHINE LANGUAGE CODE BEING GENERATED. THEY ALLOW THE PROGRAMMER TO REPRESENT NUMERIC DATA AS DECIMAL, OCTAL, BINARY, HEXADECIMAL, OR SCALED FRACTIONS, TO ASSIGN THESE NUMERIC VALUES TO SYMBOLIC NAMES, TO DEFINE DATA AND DATA STORAGE LOCATIONS, TO IDENTIFY THE PROCESSOR TO WHICH A SECTION OF CODE BELONGS AND THE LOCATION OF THAT CODE IN ITS PROGRAM MEMORY, TO CONTROL THE LISTING'S FORMAT, TO INCLUDE ANOTHER SOURCE FILE IN THE CURRENT ASSEMBLY, AND SEVERAL OTHER RELATED FUNCTIONS.

NUMERIC CONSTANTS

THE FOLLOWING METHODS FOR SPECIFYING NUMERIC CONSTANTS ARE AVAILABLE IN THE ASSEMBLER :

N	WHERE, "N" IS A NUMBER IN THE CURRENT RADIX (AS SET BY THE .OCTAL OR .DECIMAL DIRECTIVES)
O'N'	OCTAL VALUE "N" (E.G., O'177777')
B'N'	BINARY VALUE "N" (E.G., B'01010101')
S'N'	SCALED FRACTION VALUE "N" (E.G., S'-0.0325')
D'N'	DECIMAL VALUE "N" (E.G., D'198')

NUMERIC CONSTANTS ARE STORED AS 32-BIT TWO'S COMPLEMENT INTEGER VALUES. THE ASSEMBLER USES 32-BIT TWO'S COMPLEMENT ARITHMETIC FOR ITS EXPRESSION EVALUATION. HOWEVER, WHEN CALCULATING THE NUMERIC VALUE TO BE USED IN AN IMMEDIATE INSTRUCTION (E.G., LFI X), ONLY THE LOW-ORDER 16 BITS FROM THE EXPRESSION EVALUATION WILL EVENTUALLY BE USED. ALSO, IN AN ADDRESS CALCULATION, THE EXPRESSION FOR THE PAGE ADDRESS MUST EVALUATE TO A NUMBER IN THE RANGE 0-63 DECIMAL, WHILE THE EXPRESSION REPRESENTING THE WORD ADDRESS MUST EVALUATE TO A NUMBER IN THE RANGE 0-4095 DECIMAL (18-BIT ADDRESSING). SEE PAGE 8 FOR A DESCRIPTION OF ADDRESS SPECIFICATION EXPRESSIONS. NOTE FURTHER THAT SCALED FRACTION CONSTANTS SHOULD NOT BE USED IN ARITHMETIC EXPRESSIONS, AS THE EXPRESSION MAY NOT BE EVALUATED CORRECTLY BY THE ASSEMBLER.

SYMBOLS

A SYMBOL IS THE NAME WHICH IS ASSOCIATED WITH A NUMERIC VALUE. THIS VALUE MAY SIMPLY REPRESENT A NUMBER OR A BIT-PATTERN, AS IN :

```
WRITE   .EQU    B'011'
DAC02   .EQU    2
```

THE SYMBOLS "WRITE" AND "DAC02" ARE EXPLICITLY DEFINED HERE TO HAVE THE VALUES 3 AND 2, RESPECTIVELY. SYMBOLS MAY ALSO BE ASSOCIATED WITH MEMORY LOCATIONS, AS IN :

```
        .COP            ! COP CODE
        .ORG 100        ! LOCATION = 100
DSTART  PAUSE 1         ! WAIT TWO CYCLES
```

HERE, THE SYMBOL "DSTART" HAS THE VALUE 100 (OCTAL). AS USED IN THESE EXAMPLES, A SYMBOL IS ALSO A "LABEL", SINCE IT APPEARS IN THE LABEL FIELD OF THE STATEMENT (I.E., IT STARTS IN COLUMN 1). THESE SAME SYMBOLS, HOWEVER, MAY THEN BE USED IN SUBSEQUENT REFERENCES TO THE NUMERIC VALUES WHICH THESE SYMBOLS REPRESENT :

```
        .COP            ! COP CODE
        .ORG 200        ! LOCATION = 200
        PFI WRITE,DAC02 ! UPDATE DAC #2
        JMP DSTART      ! GOTO LOCATION 100
```

SYMBOLS MAY CONSIST OF 1-6 ALPHANUMERIC CHARACTERS (OR "#" OR "$" AS DESCRIBED BELOW). SYMBOLS LONGER THAN 6 CHARACTERS WILL BE TRUNCATED WITHOUT WARNING! THE SYMBOL MUST BEGIN WITH AN ALPHABETIC CHARACTER (A-Z) (OR "#" OR "$" IN SPECIAL CASES). THERE ARE THREE TYPES OF SYMBOLS : STANDARD, VARIABLE, AND PREDEFINED.

STANDARD SYMBOLS — CAN BE DEFINED ONLY ONCE, AND REPRESENT ONE FIXED VALUE THROUGHOUT AN ASSEMBLY.

1ST CHARACTER	A-Z
2ND - 6TH CHARACTERS	A-Z, 0-9
EXAMPLES :	ARP00,X0,K12345,REG12,SIN60

VARIABLE SYMBOLS — CAN BE DEFINED MORE THAN ONCE DURING AN ASSEMBLY, AND MAY BE USED TO REPRESENT DIFFERENT VALUES AT DIFFERENT POINTS WITHIN THE PROGRAM.

1ST CHARACTER	#
2ND - 6TH CHARACTERS	A-Z, 0-9
EXAMPLES :	#AA,#XPTR,#FVAL,#TEMP0

PREDEFINED SYMBOLS — ARE RECOGNIZED BY THE ASSEMBLER AS HAVING CERTAIN PREDEFINED VALUES WHICH MAY NOT BE CHANGED (I.E., THE USER CANNOT REDEFINE THESE SYMBOLS!). THE PREDEFINED SYMBOLS ARE:

PREDEFINED PROCESSOR SYMBOLS

```
$MAP    .EQU    1
$DEP    .EQU    2
$ARP    .EQU    3
$COP    .EQU    7
```

PREDEFINED ARP REGISTER SYMBOLS

A, B, C, D, E, R, S, L, K

OTHER PREDEFINED SYMBOLS

ALL PROCESSOR INSTRUCTIONS (E.G., LFI, MOV0, NOP, PAUSE, START, ETC.), AND ALSO THE SYMBOLS : EQ, NE, GT, GE, LT, LE, DF, MC, NDF, AND, OR, '*'.

NOTE : THE ASSEMBLER WILL FLAG ALL ILLEGAL USES OF PREDEFINED SYMBOLS AS SYNTAX ERRORS.

ASSEMBLER EXPRESSIONS

THERE ARE A NUMBER OF ELEMENTS WHICH CAN MAKE UP AN ASSEMBLER EXPRESSION. THESE ELEMENTS OF AN ASSEMBLER EXPRESSION ARE AS FOLLOWS :

1. A <FACTOR> IS THE BASIC ELEMENT OF AN ASSEMBLER EXPRESSION, CONSISTING OF :

   A) A <NUMERIC CONSTANT> (E.G., 123, D'189', O'100377'),

   B) A <SYMBOL> , WHICH HAS A NUMERIC VALUE ASSOCIATED WITH IT (E.G., #OUT0, $ARP, X),

   C) THE ASTERISK CHARACTER ('*'), WHICH HAS A NUMERIC VALUE EQUAL TO THAT OF THE LOCATION COUNTER,

   D) OR AN <ADDRESS CONSTANT> WHICH CAN BE ENCLOSED IN PARENTHESES (E.G., (PAGE0:WORD0+2), OR (X+Y)).

2. A <TERM> CONSISTS OF THE FOLLOWING :

   A) A SINGLE <FACTOR> (E.G., #OUT0),

   B) ANOTHER <TERM> MULTIPLIED BY A <FACTOR> (E.G., #OUT0*123),

   C) OR ANOTHER <TERM> DIVIDED BY A <FACTOR> (E.G., (#OUT0*123)/456).

3. AN <EXPRESSION> IS COMPOSED OF <TERMS> AND OTHER <EXPRESSIONS> AS FOLLOWS :

   A) A SINGLE <TERM> (E.G., X*Y/(Z*2)),

   B) ANOTHER <EXPRESSION> PLUS A <TERM> (E.G., (X*Y/(Z*2))+(BB/AA)),

   C) ANOTHER <EXPRESSION> MINUS A <TERM> (E.G., ((X*Y/(Z*2))+(BB/AA))-1),

   D) OR A <TERM> PRECEDED BY EITHER A PLUS SIGN ('+') OR A MINUS SIGN ('-'), WHERE THESE BINARY OPERATORS INDICATE EITHER A POSITIVE OR A NEGATIVE VALUE OF THE ENTIRE <EXPRESSION> (E.G., +(X*Y/(Z*2)), OR -(X*Y/(Z*2))).

ASSEMBLER EXPRESSIONS (CONT.)

4. AN <ADDRESS CONSTANT> CONSISTS OF :

   A) AN <EXPRESSION> (SEE ITEM #3 ON PREVIOUS PAGE),

   B) OR A DATA MEMORY <ADDRESS> , AS SPECIFIED BELOW.

5. AN <ADDRESS> SPECIFIES A DATA MEMORY LOCATION AS FOLLOWS :

   A) UNALIGNED DATA MEMORY ADDRESSES ARE REPRESENTED BY AN <EXPRESSION> REPRESENTING A PAGE ADDRESS, FOLLOWED BY A COLON (':'), FOLLOWED BY ANOTHER <EXPRESSION> REPRESENTING THE WORD ADDRESS WITHIN THE PAGE (E.G., (PAG0+2):(4095-(Y*2))).

   B) ALIGNED DATA MEMORY ADDRESSES ARE REPRESENTED BY AN <EXPRESSION> REPRESENTING A PAGE ADDRESS, FOLLOWED BY TWO COLONS ('::'), FOLLOWED BY ANOTHER <EXPRESSION> REPRESENTING THE WORD ADDRESS WITHIN THE PAGE (E.G., (PAG2+INDX)::(BASE+N)).

EXPRESSION EVALUATION

ARITHMETIC OPERATIONS WITHIN EXPRESSIONS ARE EVALUATED IN A SIMILAR MANNER TO THOSE IN FORTRAN EXPRESSIONS, WITH MULTIPLICATION AND DIVISION OPERATIONS BEING EVALUATED BEFORE ADDITION AND SUBTRACTION, UNLESS PARENTHESES SPECIFICALLY INDICATE A DIFFERENT ORDER. NO EXPONENTIATION IS PERMITTED. THE PERMISSIBLE OPERATORS ARE, IN ORDER OF PRECEDENCE :

1. * AND /
2. + AND -

FOR EXAMPLE, EACH OF THE FOLLOWING EXPRESSIONS IS EQUIVALENT :

```
**X/Y*Z-AA+BB/CC

(((((*)*X)/Y)*Z)-AA)+(BB/CC)
```

HERE THE FIRST ASTERISK IN THE EXPRESSION REPRESENTS THE CURRENT VALUE OF THE LOCATION COUNTER. ALL ARITHMETIC OPERATIONS ARE PERFORMED USING TWO'S COMPLEMENT 32-BIT INTEGER ARITHMETIC. THAT IS, AN EXPRESSION IS EVALUATED BY OPERATING ON THE 32-BIT BINARY VALUES WHICH REPRESENT THE VARIOUS FACTORS IN AN EXPRESSION. ONLY THE LOW-ORDER 16 BITS WILL EVENTUALLY BE USED FOR THE VALUE IN AN IMMEDIATE INSTRUCTION. ALSO NOTE THAT SCALED FRACTION CONSTANTS WILL NOT BE INTERPRETED AS SCALED FRACTIONS IN ARITHMETIC OPERATIONS.

GENERAL ASSEMBLER STATEMENT FORMAT

THE GENERAL FORM OF AN ASSEMBLY LANGUAGE STATEMENT IS AS FOLLOWS, WHERE THE BRACKETS INDICATE OPTIONAL ITEMS :

```
[LABEL]  OPCODE  [OPERANDS]  [[;OPCODE  [OPERANDS]]...]  [!COMMENTS]
```

THE FOLLOWING CONVENTIONS MUST BE FOLLOWED WHEN WRITING AN AD-10 ASSEMBLY LANGUAGE STATEMENT :

1. THE LABEL FIELD MUST BEGIN IN COLUMN 1.
2. THE OTHER ITEMS MAY BEGIN IN COLUMNS 2-120. THE ASSEMBLER WILL ACCEPT STATEMENTS CONTAINING UP TO 120 CHARACTERS, BUT ONLY THE FIRST 72 CHARACTERS OF A LINE WILL BE PRINTED ON THE LISTING.
3. IF NECESSARY, AN ASSEMBLER STATEMENT CAN BE CONTINUED ON SUCCESSIVE LINES, USING COLUMNS 2-120.
3. OPCODE/OPERAND SETS MUST BE SEPARATED BY ";".
4. OPERANDS MUST BE SEPARATED BY ",".
5. COMMENTS MUST BE PRECEDED BY "!".
6. SPACES OR TABS MAY BE INSERTED BETWEEN ITEMS AS DESIRED, BUT MAY NOT APPEAR WITHIN A LABEL, AN OPCODE, OR AN OPERAND.

EXAMPLES OF ASSEMBLER STATEMENTS ARE :

```
ARP10   MOV0 S,R; MOV1 R,TEMP0  ! PUT BUS DATA INTO TEMP REG.
BEGIN   START $DEP,$ARP
        PFI     WRITE,DAC02
        PAUSE 3         ! WAIT THREE INSTRUCTION CYCLES
TWO5    .EQU    40      ! 2**5 = 32
        .PAGE
```

(NOTE: THESE STATEMENTS DO NOT CONSTITUTE A MEANINGFUL AD-10 PROGRAM.)

LABELS

A LABEL CAN BE ANY STANDARD SYMBOL OR VARIABLE SYMBOL AS DEFINED IN THE "SYMBOLS" SECTION OF THIS MANUAL. THE LABEL MUST BEGIN IN COLUMN ONE (1) OF THE ASSEMBLER STATEMENT. SHOULD THE PROGRAMMER INADVERTENTLY USE ONE OF THE PREDEFINED SYMBOLS AS A LABEL, THE ASSEMBLER WILL FLAG THAT STATEMENT AS A SYNTAX ERROR.

OPCODES

AN OPCODE IS THE MNEMONIC REPRESENTATION OF AN AD-10 INSTRUCTION OR AN ASSEMBLER DIRECTIVE. THE OPCODE ENTRY IS MANDATORY (UNLESS THE LINE IS SIMPLY A COMMENT). THERE MUST BE AT LEAST ONE SPACE BETWEEN THE LABEL (IF PRESENT) AND THE OPCODE. SOME EXAMPLES OF OPCODES ARE :

```
LFI
PAUSE
MOVO
RUIF
START
.EQU
.UNDEFINE
```

OPERANDS

AN OPERAND IS AN ASSEMBLER EXPRESSION WHICH IDENTIFIES AND DESCRIBES THE DATA TO BE ACTED UPON BY THE OPCODE PORTION OF AN INSTRUCTION. IN THE CASE OF THE ARITHMETIC PROCESSOR IT ALSO INDICATES PART OF THE OPERATION TO BE PERFORMED. DEPENDING UPON THE NEEDS OF THE PARTICULAR INSTRUCTION, ONE OPERAND, SEVERAL OPERANDS, OR NONE MAY BE REQUIRED. WHEN MORE THAN ONE OPERAND IS USED, THEY MUST BE SEPARATED BY COMMAS (","). BLANKS MAY BE USED BETWEEN OPERANDS IN A LIST, BUT NOT WITHIN AN INDIVIDUAL OPERAND. EXAMPLES OF OPERANDS ARE :

```
S,R               (AS IN:   MOVO  S,R)
1                 (AS IN:   A .EQU 1)
$ARP,$MAP,$DEP    (AS IN:   START $ARP,$MAP,$DEP)
```

COMMENTS

COMMENTS PROVIDE DESCRIPTIVE INFORMATION ABOUT THE PROGRAM WHICH THE PROGRAMMER WISHES TO INCLUDE IN THE LISTING. THEY HELP TO DOCUMENT THE PROGRAM. THE COMMENT MUST BEGIN WITH AN EXCLAMATION POINT CHARACTER ("!"). THE ASSEMBLER SIMPLY IGNORES ALL CHARACTERS TO THE RIGHT OF THE EXCLAMATION POINT. THE COMMENT MAY START IN ANY COLUMN, AND IT MAY CONSIST OF ANY LEGAL ASCII CHARACTERS. AN EXAMPLE OF A COMMENT IS :

```
! ********** THIS IS A COMMENT **********
```

MICROCODING

IN ALL AD-10 PROCESSORS IT IS POSSIBLE TO MICROCODE MORE THAN ONE OPERATION IN PARALLEL. SINCE A SINGLE OPCODE FOR EACH COMBINATION OF OPERATIONS WOULD BE OVERWHELMING, THE SEPARATE OPERATIONS TO BE MICROCODED MAY ALL BE SPECIFIED ON THE SAME LINE, THUS CLARIFYING THE OPERATION. TO MICROCODE AN INSTRUCTION SEVERAL OPCODE/OPERAND GROUPS ARE COMBINED (UP TO A MAXIMUM OF SIX IN THE ARP). THESE SUB-INSTRUCTIONS MUST BE SEPARATED BY A SEMICOLON (";"). THE ASSEMBLER WILL COMBINE THE SUB-INSTRUCTIONS INTO ONE MICROCODED INSTRUCTION, IF THE COMBINATION IS LEGAL. SOME EXAMPLES OF LEGAL MICROCODED INSTRUCTIONS ARE :

```
MOV0 T2,B; MOV1 T3,L; MOV2 T4,D; MOV3 R,TEMP0; PAUSE 1

RAID AAA,BBB ; PAUSE 3
```

SOME GENERAL COMMENTS REGARDING MICROCODING :

1. THE SUB-INSTRUCTIONS WHICH CAN BE LEGALLY COMBINED ARE DEFINED FOR EACH PROCESSOR IN THE AD-10 REFERENCE MANUAL.

2. FOR SOME PROCESSORS THE ORDER IN WHICH THE SUB-INSTRUCTIONS MAY BE MICROCODED IS FIXED (E.G., IN THE ARP, THE "MOVE" INSTRUCTIONS MUST BE CODED IN ORDER : "MOV0" FIRST, THEN "MOV1", THEN "MOV2", AND THEN "MOV3"). IN GENERAL, IF A "PAUSE" SUB-INSTRUCTION IS MICROCODED, IT MUST TERMINATE THE INSTRUCTION. REFER TO THE AD-10 REFERENCE MANUAL FOR FURTHER DETAILS.

3. THE LAST SUB-INSTRUCTION IN A MICROCODED INSTRUCTION MUST NOT TERMINATE WITH A ";" UNLESS IT IS TO BE CONTINUED ON THE NEXT LINE. FOR EXAMPLE :

```
MOV0 S,R; MOV1 R,T0;
MOV2 S,R; MOV3 R,T1; PAUSE 3
```

   THE ABOVE STATEMENTS ARE EQUIVALENT TO THE SINGLE STATEMENT :

```
MOV0 S,R; MOV1 R,T0; MOV2 S,R; MOV3 R,T1; PAUSE 3
```

4. A COMMENT CANNOT BE MICROCODED WITHIN A SUB-INSTRUCTION. FOR EXAMPLE, THE FOLLOWING WILL RESULT IN EVERYTHING AFTER THE FIRST "!" BEING TREATED AS A COMMENT :

```
MOV0 S,R ! GET DATA ; MOV1 R,T ! SAVE IT
```

   THE ABOVE STATEMENT IS EQUIVALENT TO :

```
MOV0 S,R
```

PROGRAM SECTION DIRECTIVES

THE AD-10 CROSS-ASSEMBLER ACCEPTS INSTRUCTIONS FOR ALL AD-10 PROCESSORS AND DATA VALUES FOR AD-10 DATA MEMORY FROM THE SAME SOURCE FILE ON THE PDP-11 SYSTEM. THIS GROUP OF PROGRAM SECTION DIRECTIVES ENABLES THE PROGRAMMER TO IDENTIFY FOR THE ASSEMBLER THE PROGRAM MEMORY SECTION OR DATA MEMORY SECTION INTO WHICH THE INSTRUCTIONS OR DATA FOLLOWING THE DIRECTIVE SHOULD BE PUT. THE DEFAULT PROGRAM SECTION IS ".DAT".

.ARP	ARP PROGRAM SECTION
.COP	COP PROGRAM SECTION
.DEP	DEP PROGRAM SECTION
.MAP	MAP PROGRAM SECTION
.DAT	DATA MEMORY SECTION

CONTROL DIRECTIVES

THIS GROUP OF ASSEMBLER DIRECTIVES PERFORMS A VARIETY OF ASSEMBLY CONTROL FUNCTIONS.

.ORG N	SETS THE LOCATION COUNTER FOR THE CURRENT PROCESSOR OR FOR DATA MEMORY TO THE VALUE "N" (DEFAULT ORIGIN IS 0).
.INCLUDE <FILESPEC>	COPIES THE SPECIFIED SOURCE FILE INTO THE ASSEMBLER SOURCE STREAM (NORMALLY USED FOR MACROFILES).
.OCTAL	TELLS ASSEMBLER TO INTERPRET ALL NUMERIC CONSTANTS AS OCTAL VALUES (THIS IS THE DEFAULT SETTING).
.DECIMAL	TELLS ASSEMBLER TO INTERPRET ALL NUMERIC CONSTANTS AS DECIMAL VALUES.
.END	IDENTIFIES THE END OF THE ASSEMBLY (MUST TERMINATE THE USER'S PROGRAM).

DATA DIRECTIVES

THESE ASSEMBLER DIRECTIVES ASSIGN VALUES TO SYMBOLS, DEFINE CONSTANTS, AND RESERVE STORAGE SPACE.

LABEL	.EQU	<EXPRESSION>	ASSIGNS THE VALUE OF THE EXPRESSION TO THE SPECIFIED LABEL/SYMBOL.
[LABEL]	.WORD	<EXPRESSION LIST>	ASSIGNS THE VALUES OF THE EXPRESSIONS TO LOCATIONS IN THE CURRENT PROGRAM OR DATA MEMORY SECTION, STARTING WITH THE CURRENT LOCATION.
LABEL	.DEFINE	<EXPRESSION LIST>	ASSIGNS A LIST OF EXPRESSIONS TO AN 'ARRAY' OF SYMBOLS (THE ASSEMBLER APPENDS 0,1,2,... AS REQUIRED TO THE LABEL, AND DOES THE EQUIVALENT OF MULTIPLE ".EQU'S"). THE NUMBER OF CHARACTERS IN THE LABEL MUST REMAIN AT 6 OR LESS INCLUDING THE APPENDED DECIMAL DIGITS.
LABEL	.DEFAULT	<EXPRESSION LIST>	ASSIGNS THE DEFAULT VALUES IN THE LIST TO THE GROUP OF SYMBOLS IF THEY ARE CURRENTLY UNDEFINED (SIMILAR TO .DEFINE).
LABEL	.UNDEFINE		MARKS ALL ".DEFINE'D" SYMBOLS WHICH BEGIN WITH THE SPECIFIED SYMBOL'S NAME AS UNDEFINED.
[LABEL]	.BLKWD N		RESERVES STORAGE SPACE FOR A BLOCK OF DATA OR PROGRAM MEMORY LOCATIONS BY INCREMENTING THE CURRENT LOCATION COUNTER IN THE CURRENT PROGRAM SECTION BY "N" LOCATIONS.

LISTING DIRECTIVES

THESE ASSEMBLER DIRECTIVES CONTROL THE FORMAT OF THE ASSEMBLY LISTING.

.TITLE 'XXX...XXX'	SPECIFIES THE TITLE, ENCLOSED IN QUOTES, WHICH WILL APPEAR AT THE TOP OF SUCCESSIVE PAGES OF THE LISTING. ".TITLE" ALSO DOES A FORM FEED PRIOR TO PRINTING THE TITLE.
.PAGE	FORCES THE ASSEMBLER TO BEGIN A NEW PAGE IN THE LISTING.
.SPACE N	FORCES THE ASSEMBLER TO INSERT "N" BLANK LINES IN THE LISTING. IF THIS FORCES THE LISTING TO A NEW PAGE, NO FURTHER BLANK LINES ARE INSERTED.
.PRON	INCREMENTS THE PRINT LEVEL COUNTER (PLC). IF THE PLC IS >= 0, THE FOLLOWING SOURCE LINES ARE PRINTED, OTHERWISE THE PRINTING REMAINS DISABLED.
.PROFF	DECREMENTS THE PRINT LEVEL COUNTER.

MACROFILES

A MACROFILE IS AN AD-10 ASSEMBLY LANGUAGE APPLICATION ROUTINE IN SOURCE FORM WHICH CAN BE INCLUDED IN A USER APPLICATION PROGRAM WITH USER SPECIFIED INPUT/OUTPUT PARAMETERS OR ARGUMENTS. A MACROFILE IS SIMILIAR TO A SUBROUTINE IN A HIGH LEVEL LANGUAGE, WITH THE EXCEPTION THAT EACH "CALL" TO A MACROFILE INCLUDES ANOTHER COPY OF THE MACROFILE CODE, WITH THE USER SPECIFIED ARGUMENTS, IN THE USER PROGRAM.

THE AD-10 MACROFILE LIBRARY CONTAINS ROUTINES WHICH SUPPORT ALL PHASES OF MULTIVARIABLE FUNCTION GENERATION APPLICATIONS, INCLUDING DATA INPUT AND OUTPUT, DATA TRANSFERS WITHIN THE AD-10, BINARY AND SHIFT SEARCH SCHEMES (USED TO DETERMINE THE LOCATION OF INPUT VARIABLES IN THE DOMAIN OF THE FUNCTION), POINTER CALCULATIONS, AND LINEAR INTERPOLATION FOR 1,2,3,4, AND 5 VARIABLES. IN ADDITION, A NUMBER OF SUPPORT ROUTINES ARE INCLUDED TO PERFORM SUCH CALCULATIONS AS SIN'S AND COS'S, FORWARD AND INVERSE RESOLUTION, "SGN" FUNCTION, ETC... ROUTINES WHICH PERFORM GENERAL CALCULATIONS SUCH AS THESE ARE CONSTANTLY BEING ADDED TO THE MACROFILE LIBRARY AS THEY PROVE USEFUL IN USER APPLICATIONS. THE CONVENTIONS USED IN WRITING MACROFILES AND IN PASSING ARGUMENTS TO MACROFILES ARE VERY SIMPLE, THUS USERS CAN EASILY WRITE THEIR OWN SPECIAL PURPOSE MACROFILES TO AUGMENT THOSE PROVIDED IN THE LIBRARY.

REFER TO THE MACROFILE LIBRARY USER'S MANUAL (MFLIB) FOR A COMPLETE DESCRIPTION OF ALL AVAILABLE MACROFILE ROUTINES AND HOW TO USE THEM.

USING MACROFILES

A MACROFILE IS AN AD-10 ASSEMBLY LANGUAGE APPLICATION PROGRAM. IT CONTAINS ITS OWN COP CONTROL PROGRAM AS WELL AS PROGRAMS FOR ALL AD-10 PROCESSORS REQUIRED TO PERFORM THE DESIRED TASK. ARGUMENTS ARE PASSED TO AND FROM MACROFILES USING SYMBOLS WHICH BEGIN WITH A "#". THESE "#" SYMBOLS STAND FOR EITHER A TEMPORARY REGISTER NUMBER, A CONSTANT, OR A MEMORY ADDRESS. THE ONLY DIFFERENCE BETWEEN A "#" SYMBOL AND AN ORDINARY SYMBOL IS THAT THE AD-10 ASSEMBLER ALLOWS A SYMBOL WHICH BEGINS WITH "#" TO BE DEFINED MORE THAN ONCE. THIS ALLOWS THE USER TO CALL THE SAME MACROFILE MORE THAN ONCE AND TO CHANGE THE ARGUMENTS AS NECESSARY. IF A MACROFILE ARGUMENT DOES NOT CHANGE FROM ONE CALL TO THE NEXT IT IS NOT NECESSARY TO DEFINE THAT ARGUMENT MORE THAN ONCE. HOWEVER, BE AWARE THAT IN SOME CASES THE SAME SYMBOLIC ARGUMENT IS USED BY SEVERAL MACROFILES.

SINCE BOTH THE AD-10 PROCESSORS AND THE DATA MEMORY REQUIRE PIPELINED PROGRAMMING TO REALIZE FULL SPEED EFFICIENT OPERATION, MOST MACROFILES PERFORM THE SAME TASK FOR SEVERAL SETS OF INPUTS. BECAUSE OF THIS, THE NAMING CONVENTION FOR MACROFILE ARGUMENTS IS TO END EACH ARGUMENT WITH A NUMBER TO IDENTIFY EACH ARGUMENT SET.

FOR EXAMPLE, SUPPOSE "#IN" IS THE INPUT AND "#OUT" IS THE OUTPUT OF A MACROFILE CALLED "COMPUTE", AND THE CALCULATIONS ARE PERFORMED FOR 3 SETS OF ARGUMENTS. THE FOLLOWING STATEMENTS WOULD BE REQUIRED TO DEFINE THE 3 SETS OF ARGUMENTS AND TO "CALL" THE MACROFILE:

```
#IN0    .EQU    <VALUE1>
#IN1    .EQU    <VALUE2>
#IN2    .EQU    <VALUE3>
#OUT0   .EQU    <VALUE4>
#OUT1   .EQU    <VALUE5>
#OUT2   .EQU    <VALUE6>
        .INCLUDE  COMPUTE   ! "CALLS" MACROFILE
```

THE AD-10 ASSEMBLER'S ".DEFINE" DIRECTIVE DOES THE EQUIVALENT OF MULTIPLE ".EQU" SYMBOL DEFINITIONS AND ALLOWS THE ARGUMENTS TO MACROFILES TO BE DEFINED MORE SIMPLY AS FOLLOWS :

```
#IN     .DEFINE <VALUE1>,<VALUE2>,<VALUE3>
#OUT    .DEFINE <VALUE4>,<VALUE5>,<VALUE6>
        .INCLUDE  COMPUTE  ! "CALLS" MACROFILE
```

SOME MACROFILES MUST DEFINE THEIR OWN INTERNAL SYMBOLS FOR ADDRESS CALCULATIONS OR FOR TEMPORARY STORAGE LOCATIONS; WHENEVER A MACROFILE DOES DEFINE A SYMBOL INTERNALLY, THE SYMBOL ALWAYS BEGINS WITH "##", THUS INTERNAL SYMBOLS SHOULD NEVER CONFLICT WITH USER SYMBOLS OR OTHER MACROFILE ARGUMENTS.

USING MACROFILES (CONT.)

THE GENERAL FORMAT FOR MACROFILES IN AD-10 ASSEMBLY LANGUAGE NOTATION IS AS FOLLOWS:

```
 .PROFF     ! PRECEDE  WITH A ".PRON" TO PRINT CODE
 .PROFF     ! PRECEDE WITH ANOTHER ".PRON" TO PRINT DESCRIPTION
 .PAGE      ! STARTING AT THE TOP OF THE NEXT PAGE
!
!  DESCRIPTION OF MACROFILE
!
 .PRON
 .COP
!************************************
!  COP CONTROL PROGRAM
!************************************
!  PROGRAMS FOR ANY OTHER PROCESSORS
!************************************
 .COP
 .PRON      ! END OF MACROFILE
```

THE USER MUST SPECIFY ONE .PRON IN THE PROGRAM PRIOR TO INCLUDING A MACROFILE FOR THE MACROFILE CODE TO BE PRINTED IN THE PROGRAM LISTING. A DETAILED DESCRIPTION OF THE MACROFILE AND ITS ARGUMENTS CAN ALSO BE PRINTED BY USING A SECOND ".PRON", HOWEVER THIS IS NOT RECOMMENDED SINCE SOME OF THE DESCRIPTIONS ARE QUITE LONG AND THE SAME INFORMATION IS CONTAINED IN THIS MANUAL. NOTICE THAT MACROFILES END WITH A ".COP" DIRECTIVE, THUS A MACROFILE CAN BE FOLLOWED WITH COP CODE WITHOUT ISSUING ANOTHER ".COP" DIRECTIVE.

THERE ARE A FEW RULES WHICH MUST BE FOLLOWED WHEN INCLUDING MACROFILES TO AVOID CONFLICTS AND ERRONEOUS RESULTS AT RUNTIME :

1) UPON ENTRY TO A MACROFILE ALL AD-10 PROCESSORS MUST BE STOPPED AND MUST NOT BE IN THE MIDDLE OF A "PAUSE" INSTRUCTION. (NOTE: PROCESSOR(S) NOT USED BY A MACROFILE COULD POSSIBLY BE PROGRAMMED TO PERFORM SOME INTERNAL OPERATIONS IN PARALLEL WITH THE MACROFILE, BUT THIS IS NOT RECOMMENDED.)

2) ALSO UPON ENTRY A READ FROM MEMORY AND/OR THE IOCC MUST NOT BE IN PROGRESS, AS THE DATA MIGHT CONFLICT WITH DATA THE MACROFILE PUTS ON THE MULTIBUS.

3) THE USER SHOULD TAKE CARE PRIOR TO AND/OR FOLLOWING ANY MACROFILE WHICH ACCESSES DATA MEMORY TO AVOID A MEMORY PAGE CONFLICT. IF IN DOUBT, A "PAUSE 2" INSTRUCTION PRIOR TO AND/OR FOLLOWING SUCH A MACROFILE WILL AVOID ANY POSSIBILITY OF A MEMORY PAGE CONFLICT (FOR THE WORST CASE SITUATION).

4) ALL UNUSED MACROFILE ARGUMENTS MUST BE DEFINED SO AS NOT TO CONFLICT WITH THE USED ARGUMENTS. FOR EXAMPLE, IF A TRANSFER MACROFILE IS USED TO TRANSFER 6 VALUES TO MEMORY, WHEN IT HAS THE CAPABILITY TO TRANSFER 8 VALUES, THE 2 UNUSED MEMORY ADDRESSES MUST BE DEFINED SUCH THAT THEY DO NOT CAUSE A MEMORY ACCESS ERROR. THE INDIVIDUAL MACROFILE DESCRIPTIONS SUGGEST RECOMMENDED DEFINITIONS FOR UNUSED ARGUMENTS.

OPERATING PROCEDURES

ASM IS SUPPLIED AS AN INSTALLED TASK ON THE RSX-11 SYSTEM DEVICE. ASM IS THEN LOADED AND RUN AS ARE OTHER RSX-11 SYSTEM PROGRAMS, BY TYPING (IN RESPONSE TO THE MCR PROMPT) :

```
        MCR>ASM [<COMMAND STRING>] <CR>
OR,     MCR>RUN $ASM <CR>
OR,     MCR>RUN ...ASM <CR>
```

THE COMMAND STRING IS OPTIONAL HERE. IF THE COMMAND STRING IS NOT ENTERED WITH THE MCR COMMAND, ASM WILL RESPOND WITH ITS OWN PROMPT :

```
MCR>ASM <CR>
ASM><COMMAND STRING> <CR>
```

THE GENERAL FORMAT OF THE ASM COMMAND STRING IS DEFINED AS FOLLOWS :

```
        @<FILE SPECIFICATION>
OR,     [<OBJECT>],[<LISTING>]=<SOURCE>[/<SWITCHES>]
```

THE FIRST FORM INDICATES THAT THE COMMAND STRING(S) WILL COME FROM THE SPECIFIED COMMAND FILE. THE SECOND FORM CONSISTS OF THE APPROPRIATE ASM COMMAND STRING WITH THE OPTIONAL SWITCHES. IF THE INDIRECT COMMAND FILE FORMAT IS USED ON THE SAME LINE AS THE MCR PROMPT, CONTROL WILL RETURN TO MCR AFTER PROCESSING THE ASM COMMANDS IN THE FILE :

```
MCR>ASM @DK0:FILE.CMD;3 <CR>
MCR>
```

IF THE INDIRECT COMMAND FILE IS SPECIFIED AFTER THE ASM PROMPT, CONTROL WILL REMAIN WITH ASM FOLLOWING COMMAND FILE PROCESSING :

```
MCR>ASM <CR>
ASM>@FILE <CR>
ASM>
```

THE AVAILABLE SWITCHES ARE AS FOLLOWS (DEFAULTS ARE UNDERLINED) :

/LI:SRC:SYM:CRF:TIM --- ---	IDENTIFIES WHAT ITEMS ARE TO BE INCLUDED IN THE LISTING (SOURCE, SYMBOL TABLE, CROSS REFERENCE, AND/OR TIMING DIAGRAM).
/NL:SRC:SYM:CRF:TIM --- ---	IDENTIFIES WHAT ITEMS ARE NOT TO BE INCLUDED IN THE LISTING
/SY:N	SETS THE SIZE OF THE SYMBOL TABLE TO "N" (DECIMAL) (DEFAULT IS 500 SYMBOLS). THE MAXIMUM SIZE IS 1000 SYMBOLS, WITHOUT REBUILDING ASM.
/CR:N	SETS THE SIZE OF THE CROSS-REFERENCE TABLE TO "N" (DECIMAL) (DEFAULT IS 1000).
/NI	IGNORE ALL ".INCLUDE'S" IN THE SOURCE CODE.

/ID	PRINTS THE ASSEMBLER VERSION NUMBER ON THE CONSOLE TERMINAL.
/PG:N	SETS THE LISTING SIZE TO "N" LINES PER PAGE (DEFAULT IS 58).

THE DEFAULT SWITCH SETTINGS ARE AS FOLLOWS :

```
/LI:SRC:SYM /NL:CRF:TIM /SY:500 /CR:1000 /PG:58
```

THE ASSEMBLER USES A FILE FOR THE SYMBOL TABLE AND ONE FOR THE CROSS-REFERENCE TABLE. EACH SYMBOL REQUIRES 8 WORDS OF FILESPACE, AND EACH CROSS-REFERENCE REQUIRES 2 WORDS OF FILESPACE. THE SWITCHES MAY BE PUT ANYWHERE IN THE COMMAND LINE, BUT A GOOD PRACTICE TO FOLLOW IS TO PUT /LI, /NL, AND /PG ON THE LISTING FILE; TO PUT /SY, /CR, AND /NI ON THE SOURCE FILE; AND TO USE /ID AS A SINGLE COMMAND TO ASM.

ASM WILL APPEND THE FOLLOWING DEFAULT EXTENSIONS TO THE FILENAMES, IF NO EXTENSION IS SPECIFIED BY THE USER :

FILE	DEFAULT EXTENSION
OBJECT FILE	.MOD
LISTING FILE	.LST
SOURCE FILE	.ASM

AN EXAMPLE SEQUENCE OF COMMANDS TO ASSEMBLE AND RUN AN AD-10 PROGRAM IS AS FOLLOWS :

```
MCR>ASM <CR>
ASM>/ID <CR>
AD-10 ASSEMBLER --- UL107

ASM>TEST,TEST/LI:TIM=TEST/SY:700/NI <CR>
ASM><CONTROL-Z>
MCR>ADX <CR>
ADX>; NOW THE AD-10 EXECUTIVE CAN BE USED TO RUN THE <CR>
ADX>; OBJECT PROGRAM <CR>
ADX>RUN TEST/MO <CR>
```

THIS EXAMPLE ASSUMES THAT THE SOURCE CODE IS IN A FILE "TEST.ASM". THE OBJECT CODE WILL BE PUT INTO A FILE CALLED "TEST.MOD" AND THE LISTING WILL BE PUT INTO ANOTHER FILE CALLED "TEST.LST". THE LISTING WILL INCLUDE THE SOURCE CODE, THE SYMBOL TABLE, AND A BUS TIMING DIAGRAM. THE SIZE OF THE SYMBOL TABLE HAS BEEN INCREASED TO 700 SYMBOLS TO ALLOW FOR A LARGER NUMBER OF SYMBOLS, AND ALL ".INCLUDE" STATEMENTS IN THE SOURCE CODE WILL BE IGNORED. THE /NL SWITCH IS NORMALLY USED ONLY FOR DEBUGGING THE PROGRAM'S MULTIBUS TIMING (MACRO-FILES ARE ASSUMED ERROR-FREE ...). THE OBJECT PROGRAM IS THEN LOADED INTO THE AD-10 AND RUN VIA THE AD-10 EXECUTIVE (ADX).

COMMAND LINE ERROR MESSAGES

ASM -- I/O ERROR	BAD INDIRECT FILE	
ASM -- SYNTAX ERROR	COMMAND LINE SYNTAX PROBLEM	
ASM -- INPUT FILE ????	MISSING INPUT FILESPEC	
ASM -- INVALID LIST OPTION	/LI:XXX	"XXX" IS BAD
ASM -- INVALID NOLIST OPTION	/NL:XXX	"XXX" IS BAD
ASM -- INVALID SYMBOL TABLE SIZE	/SY:N	"N" IS BAD
ASM -- INVALID CROSS-REFERENCE SIZE	/CR:N	"N" IS BAD
ASM -- INVALID PAGE SIZE	/PG:N	"N" IS BAD

THE FOLLOWING MESSAGES SHOULD NEVER APPEAR. THEY INDICATE THAT SOMETHING IS WRONG WITH THE ASSEMBLER OR WITH THE FILE SYSTEM :

ASM -- STACK OVERFLOW - PASS 7

ASM -- TOKENS DO NOT MATCH <T7> <T1> <STMT>

LISTING ERROR MESSAGES

ALL OF THE APPROPRIATE ERROR MESSAGES WILL BE PRINTED ON THE FIRST PAGE OF THE ASSEMBLY LISTING. IF NONE ARE PRINTED, NO ERRORS WERE DETECTED. THE STATEMENT(S) IN THE LISTING WITH ERRORS WILL ALSO BE FLAGGED WITH AN "E" NEXT TO THE STATEMENT NUMBER ON THE LISTING.

```
101 **  STACK OVERFLOW -- CHECKING ABORTED
102     TOO MANY INCLUDES
103     ILLEGAL CHARACTER IN OCTAL FIELD
104     NO PRODUCTION APPLICABLE (SYNTAX)
105     ILLEGAL CHARACTER
106     STRING NOT TERMINATED BY '
107     ILLEGAL CONSTANT
108     CONSTANT NOT TERMINATED BY '
109     ILLEGAL SYMBOL PAIR (SYNTAX)

201     SYMBOL TOO LONG TO DEFINE
202     DUPLICATE SYMBOL
203 *   SYMBOL TABLE OVERFLOW / OH DARN !
204 **  STACK OVERFLOW - PASS 2
205 **  CROSS REFERENCE TABLE OVERFLOW

301 **  STACK OVERFLOW - PASS 3
302     INVALID MICRO-INSTRUCTION IN COP
303     COP INSTRUCTION HAS INVALID OPERAND(S)
304     INVALID MICRO-INSTRUCTION IN DEP
305     DEP INSTRUCTION HAS INVALID OPERAND(S)
306     INVALID MICRO-INSTRUCTION IN ARP
307     INVALID IA STATEMENT IN ARP
308     INVALID ARP EXPRESSION
309     ARP MOV HAS INVALID OPERAND(S)
310     EXPRESSION LIST TOO LONG
311     SYMBOL NOT DEFINED
312     PAGE ADDRESS OUT OF RANGE
313     WORD ADDRESS OUT OF RANGE
314     MAP STATEMENT HAS INVALID OPERAND(S)
315     INVALID PAUSE COUNT
316     INVALID SCRATCH REGISTER

401     ILLEGAL PROCESSOR FOR TIMING
402     ILLEGAL NON-LINEAR COP CODE
403     PROCESSORS ACTIVE AT JUMP
404 **  STACK OVERFLOW - PASS 4

501 **  STACK OVERFLOW - PASS 5
502     INSTRUCTION NOT USED IN TIMING
```

* TOO MANY SYMBOLS : INCREASE SYMBOL TABLE SIZE (RE-ASSEMBLE USING "/SY:N" SWITCH, WITH 500<N<=1000)

** THESE ERROR MESSAGES SHOULD NEVER BE SEEN !

TIMING DIAGRAM ERROR MESSAGES

THE FOLLOWING MNEMONIC ERROR MESSAGES ARE PRINTED IN THE 'COMMENTS' COLUMN OF THE MULTIBUS TIMING DIAGRAM, AND INDICATE POSSIBLE PROGRAMMING ERRORS. IN SOME INSTANCES (SEE THE FIRST COP INSTRUCTION IN THE EXAMPLE PROGRAM AT THE END OF THIS MANUAL) A TIMING ERROR WON'T REALLY EXIST BECAUSE OF THE WAY IN WHICH A PARTICULAR INSTRUCTION HAS BEEN USED :

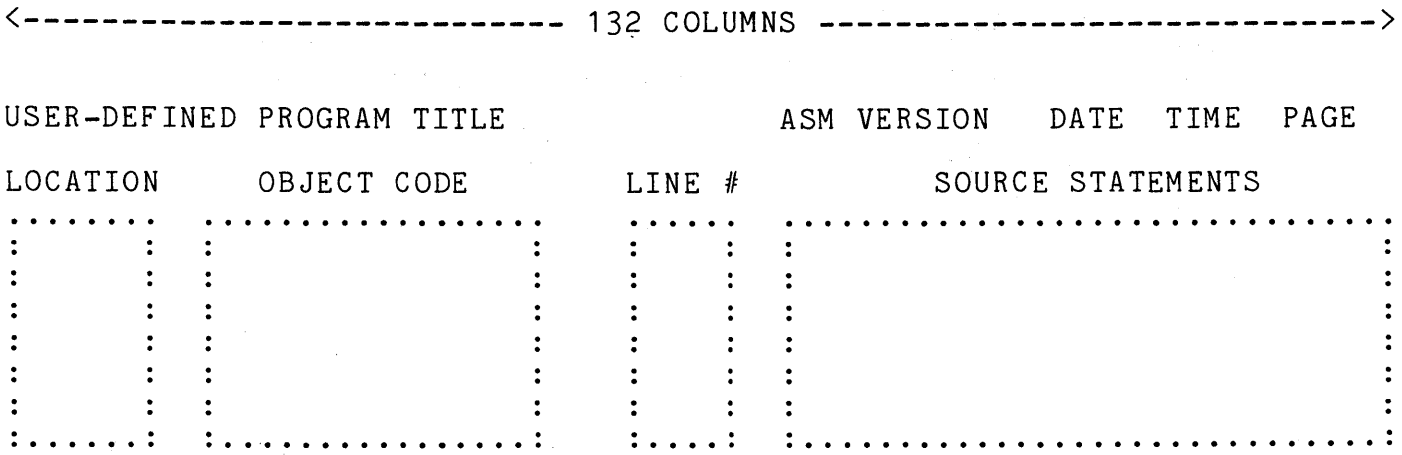
DCF	DATA CONTENTION DURING FIRST MULTIBUS CYCLE
DCS	DATA CONTENTION DURING SECOND MULTIBUS CYCLE
ACF	ADDRESS CONTENTION DURING FIRST MULTIBUS CYCLE
ACS	ADDRESS CONTENTION DURING SECOND MULTIBUS CYCLE
NSOF	NO SOURCE DURING FIRST MULTIBUS CYCLE
NSOS	NO SOURCE DURING SECOND MULTIBUS CYCLE
NSIF	NO SINK DURING FIRST MULTIBUS CYCLE
NSIS	NO SINK DURING SECOND MULTIBUS CYCLE

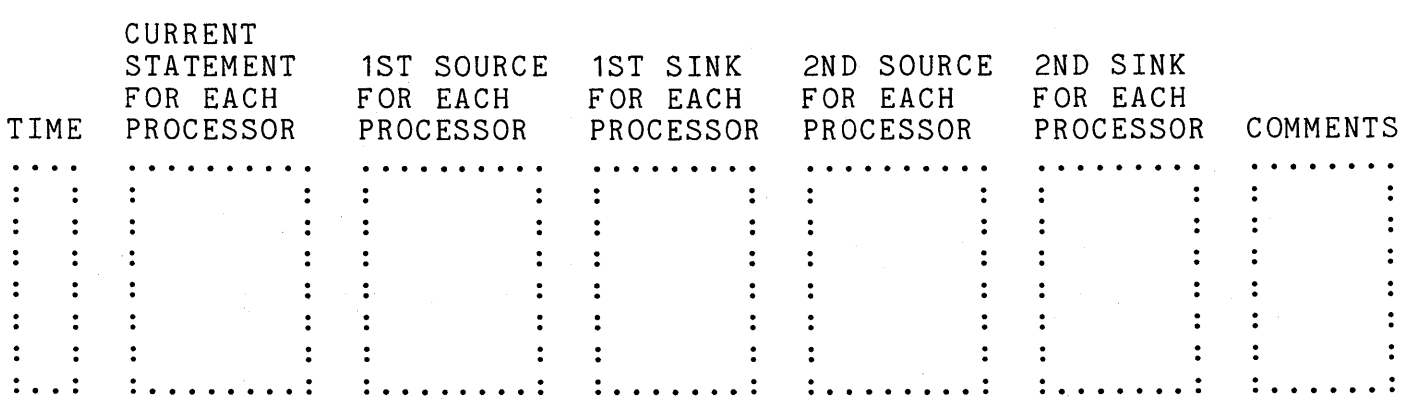
NOTE : THE TERM "SOURCE" REFERS TO AN INSTRUCTION WHICH PUTS A DATA VALUE OR AN ADDRESS VALUE ONTO THE MULTIBUS.

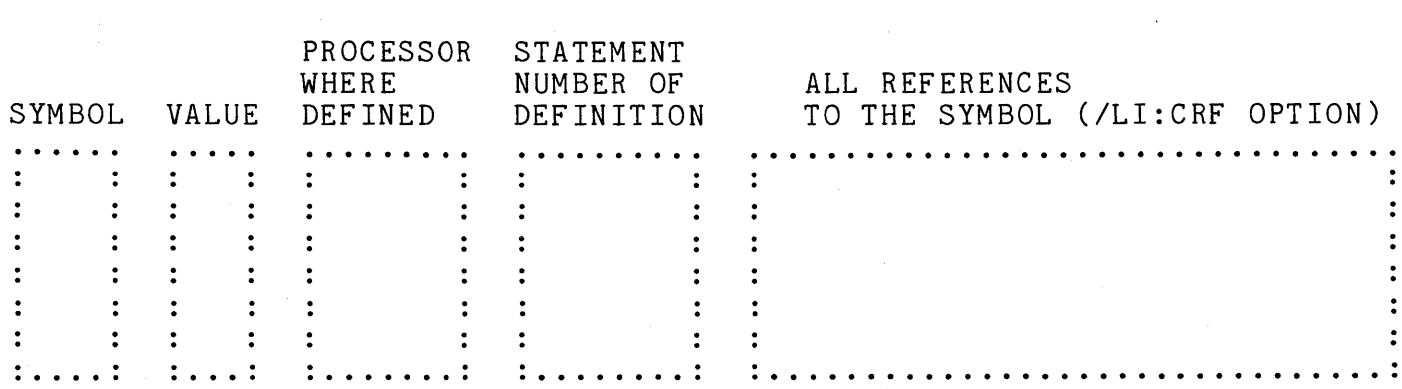
THE TERM "SINK" REFERS TO AN INSTRUCTION WHICH TAKES A DATA VALUE OR ADDRESS VALUE OFF OF THE MULTIBUS.

LISTING AND TIMING DIAGRAM FORMATS

THE FOLLOWING DIAGRAMS INDICATE THE FORMAT OF THE 132-COLUMN ASSEMBLER LISTING, INCLUDING THE SOURCE AND OBJECT LISTING, THE MULTIBUS TIMING DIAGRAMS (BOTH DATA AND ADDRESS), AND THE SYMBOL TABLE AND CROSS-REFERENCES. THE "/LI:TIM" SWITCH MUST BE SPECIFIED TO GET THE TIMING INFORMATION, AND THE "/LI:CRF" SWITCH MUST BE SPECIFIED TO GET THE CROSS-REFERENCE INFORMATION.







OBJECT MODULE FORMAT

THE OBJECT MODULE IS IN A FORMAT WHICH ALLOWS IT TO BE LOADED INTO THE AD-10 BY THE AD-10 EXECUTIVE PROGRAM (ADX). THE FORMAT IS AS FOLLOWS :

1. A HEADER, FOLLOWED BY
2. THE CODE (INSTRUCTIONS OR DATA), FOLLOWED BY
3. OTHER HEADER/CODE MODULES (IF MORE THAN ONE PROGRAM SECTION WAS SPECIFIED IN THE ASSEMBLY), FOLLOWED BY
4. < END>.

THE HEADER FORMAT TAKES TWO FORMS :

1. FOR A PROCESSOR PROGRAM SECTION THE HEADER IS :

   <PROCESSOR> <STARTING LOCATION> <NUMBER OF LOCATIONS>

   WHERE, <PROCESSOR> = < ARP>, < COP>, < DEP>, < MAP>
   <STARTING LOCATION> = 0-1777
   <NUMBER OF LOCATIONS> = 0-2000

2. FOR A DATA MEMORY SECTION THE HEADER IS :

   < DAT> <PAGE> <STARTING LOCATION> <NUMBER OF LOCATIONS>

   WHERE, <PAGE> = 0-77
   <STARTING LOCATION> = 0-7777
   <NUMBER OF LOCATIONS> = 0-10000

THE FORTRAN FORMAT WHICH IS USED TO WRITE BOTH HEADERS IS AS FOLLOWS :

```
FORMAT (A4,4X,O6,1X,O6,1X,O6)
```

THE OCTAL CODE FOLLOWS THE HEADER, ORGANIZED AS 10 DATA VALUES PER LINE AS FOLLOWS :

<#1> <#2> <#3> <#4> <#5> <#6> <#7> <#8> <#9> <#10>
<#11> <#12> ... <#N>

NOTE : EACH DATA VALUE WILL REPRESENT ONE 16-BIT FIELD OF A PROCESSOR
---- INSTRUCTION, OR ONE 16-BIT DATA MEMORY WORD.

THE FORTRAN FORMAT WHICH IS USED TO WRITE THE DATA VALUES IS :

```
FORMAT (10(1X,O6))
```

A SAMPLE PROGRAM OBJECT MODULE IS AS FOLLOWS :

```
 COP          0     21
     22     210      6  10401       0  40040      22     400       0  40020
      0   10401      0 130002       0  40040       0   10401       0  10401
      0   10401      0  40020      10  10401       0   30005       0  10400
      0  130000      0  20000
 ARP          0     10
      0  130200      0      0       0      0  110000       0       0       0
      0       0      0      0       0    106       0       0       0       0
      0       0      0      0   60400      0       0       0       0       0
      0       0      0      0       0      0       0       0       0       0
 END
```

AN EXAMPLE ASSEMBLER PROGRAM

THE EXAMPLE AD-10 PROGRAM ON THE FOLLOWING PAGES SHOULD HELP TO CLARIFY THE INFORMATION WHICH HAS BEEN PRESENTED IN THIS MANUAL. AT THE END OF THE DISCUSSION IS PRESENTED THE SEQUENCE OF COMMANDS TO ASM WHICH WERE USED TO ASSEMBLE THE SOURCE FILE FOR THIS PROGRAM, THE LISTING OF WHICH IS INCLUDED. FOLLOWING THE SOURCE LISTING IS THE OUTPUT OF THE ASSEMBLER. FOLLOWING THE LISTING OUTPUT OF THE ASSEMBLER IS A COPY OF THE OBJECT MODULE WHICH WAS GENERATED FOR THIS PROGRAM.

PROGRAM DESCRIPTION

THIS AD-10 PROGRAM (SINE.ASM) READS AN INPUT VALUE FROM ADC CHANNEL 0 AND USES THIS VALUE TO DETERMINE THE DATA MEMORY LOCATIONS TO USE FOR DOING A TABLE-LOOKUP IN A TABLE OF FUNCTION DATA VALUES FOR THE FUNCTION SIN(X). THE TABLE OF DATA VALUES MAY BE CREATED BY THE FOLLOWING FORTRAN PROGRAM :

```
C  SIN.FTN
C
C  PROGRAM TO CREATE A FUNCTION DATA FILE
C  (SIN.DAT) FOR THE FUNCTION  Y=SIN(X)
C
        CALL ASSIGN (1,'SIN.DAT',7)
        DEFINE FILE 1 (513,2,U,IREC)
C
        DO 100 I=1,513
          Y= SIN ( ((I-257)/256.) * 3.14159 )
          WRITE (1'I) Y
100     CONTINUE
C
        CALL CLOSE(1)
        END
```

THIS FORTRAN PROGRAM CREATES A DATA FILE (SIN.DAT) WHICH CONTAINS 513 FUNCTION DATA VALUES FROM SIN(-PI) TO SIN(+PI), AND WHICH MAY BE LOADED INTO THE AD-10 DATA MEMORY VIA ADX. THE AD-10 PROGRAM WILL CALCULATE A BREAKPOINT VALUE TO USE AS AN INDEX INTO THIS TABLE BASED UPON THE VALUE IT READS FROM THE ADC (-1.0 TO +1.0). IT WILL THEN INTERPOLATE BETWEEN FUNCTION VALUES TO CALCULATE THE OUTPUT VALUE FOR THE CURRENT ADC INPUT. THIS FUNCTION VALUE IS THEN SENT TO DAC CHANNEL 8 (OCTAL 010). THE DAC WILL BE UPDATED TO A NEW VALUE EVERY 12.5 MICROSECONDS, AS THIS IS THE TIME REQUIRED FOR A COMPLETE PROGRAM LOOP. THUS, WHEN A (-1.0 TO +1.0) RAMP FUNCTION IS INPUT TO THE ADC, A SINEWAVE FUNCTION WILL BE GENERATED AT THE DAC'S OUTPUT.

THE FOLLOWING PAGE CONTAINS A BRIEF DISCUSSION OF THIS PROGRAM.

AS YOU STUDY THIS EXAMPLE, THERE ARE SEVERAL COMMENTS WHICH CAN BE MADE REGARDING THIS PROGRAM AND AD-10 PROGRAMMING TECHNIQUES IN GENERAL. THE PROGRAM BEGINS WITH A HEADER, BRIEFLY DESCRIBING THE ALGORITHM. FOLLOWING THE HEADER IS THE SYMBOL DEFINITION SECTION. NOTICE THAT ALL NUMERIC CONSTANTS WILL BE INTERPRETED AS DECIMAL NUMBERS. BECAUSE NO PROGRAM SECTION DIRECTIVE IS GIVEN, THESE SYMBOLS WILL APPEAR IN THE "DAT" PROGRAM SECTION IN THE SYMBOL TABLE LISTING, SINCE ".DAT" IS THE DEFAULT PROGRAM SECTION. SINCE THE COP IS THE PROCESSOR WHICH USUALLY CONTROLS THE OTHER PROCESSORS, IT IS A GOOD IDEA TO ALWAYS MAKE ".COP" THE FIRST PROGRAM SECTION AFTER THE DATA MEMORY SECTION. EXCEPT FOR PROVIDING CONSISTENCY FROM PROGRAM TO PROGRAM, THE ORDER OF THE PROGRAM SECTIONS IS NOT IMPORTANT (BE AWARE THAT UPON RETURNING FROM A MACROFILE THE PROGRAM SECTION WILL BE ".COP").

NOTICE THE USE OF COMMENTS IN THIS PROGRAM. NOT ONLY DO THEY DESCRIBE WHAT THE INSTRUCTIONS ARE DOING, BUT THEY ALSO INCLUDE PRECALCULATED TIMING INFORMATION. YOU MUST CALCULATE THIS INFORMATION IN ORDER TO INSURE THAT THE PROCESSORS WILL COMMUNICATE WITH EACH OTHER AT THE CORRECT TIMES, SO INCLUDE IT IN THE SOURCE CODE AS COMMENTS. IF YOU HAVE MISCALCULATED THE TIMING SOMEWHERE, A COMPARISON OF THESE EXPECTED TIMES (WHICH ARE INSTRUCTION CYCLES, UNITS OF 100 NSEC) WITH THE TIMES ON THE ADDRESS OR DATA MULTIBUS TIMING DIAGRAMS WILL QUICKLY LOCATE THE ERROR.

NOTICE HOW THE PROCESSORS INTERACT IN THIS PROGRAM. AFTER THE INITIAL 10 MICROSECOND DELAY (6.4 MICROSEC FROM THE PAUSE INSTRUCTION, AND 3.6 MICROSEC FROM THE RFR INSTRUCTION AND ITS DELAY), THE COP TELLS THE IOCC TO PUT THE ADC VALUE ONTO THE MULTIBUS. THE DATA IS NOT PRESENT ON THE BUS FOR 5 MORE INSTRUCTION CYCLES, AT WHICH TIME THE ARP SINKS IT TO USE IN ITS CALCULATION OF THE FUNCTION VALUE INDEX. NOTICE HOW THE DEP ALSO SENDS CONSTANTS TO THE ARP AT THE APPROPRIATE TIMES FOR USE IN THE ARP'S INDEX CALCULATION, AND SEE HOW THE ARP RETURNS THE INDEX VALUE TO THE DEP SO THAT IT MAY BE PUT INTO THE MAP/DEP "I" REGISTER AND USED BY THE MAP TO READ THE APPROPRIATE FUNCTION VALUE. AFTER THE 5 CYCLE DELAY TO READ THE PAIR OF VALUES FROM DATA MEMORY, THE ARP WILL SINK THE VALUES FROM THE MULTIBUS. THE ARP WILL THEN SOURCE THE FUNCTION VALUE ONTO THE MULTIBUS AT THE APPROPRIATE TIME SO THAT IT WILL BE THERE WHEN THE COP TELLS THE IOCC TO WRITE THE VALUE ON THE MULTIBUS TO DAC CHANNEL 010.

THE ASSEMBLER'S DATA MULTIBUS TIMING DIAGRAM FOR THIS PROGRAM SHOWS AN "NSOF", OR "NO SOURCE DURING FIRST MULTIBUS CYCLE", ERROR OCCURRING ON THE INITIAL COP STATEMENT (PFI 3,ADC). THIS, HOWEVER, IS NOT REALLY AN ERROR CONDITION HERE, BECAUSE THIS PFI INSTRUCTION IS NOT INTENDED TO TELL THE IOCC TO TAKE THE DATA OFF OF THE MULTIBUS, BUT MERELY TO SEND THE COMMAND WHICH INITIATES ADC CONVERSION.

THE FOLLOWING COMMANDS WERE USED TO CREATE THE SINE FUNCTION DATA (THE FORTRAN PROGRAM WAS IN FILE SIN.FTN), ASSEMBLE SINE.ASM, AND LOAD AND RUN THE PROGRAM :

```
MCR>FOR SIN,SIN=SIN <CR>
MCR>TKB SIN,SIN/SH=SIN <CR>
MCR>RUN SIN <CR>
TT0  --  STOP
MCR>ASM SINE,SINE/LI:TIM:CRF/PG:60=SINE <CR>
MCR>ADX LOAD SIN.DAT/AL:0:0 <CR>
MCR>ADX RUN SINE/MO <CR>
MCR>
```

```
!  AD-10 SINEWAVE PROGRAM (SINE.ASM)
!
!  1) READS "X" FROM AN ADC CHANNEL.
!  2) PERFORMS A SHIFT SEARCH AND DELTA CALCULATION ON "X".
!  3) INTERPOLATES FOR A ONE VARIABLE FUNCTION SIN(X).
!  4) SETS DAC = SIN(X)
!  5) REPEATS STEPS 1) THROUGH 4) EVERY 12.5 MICRO-SECONDS.
!
        .DECIMAL        ! CONSTANTS WILL BE DECIMAL NUMBERS.
ADC     .EQU    0       ! USE ADC CHANNEL 0
DAC     .EQU    8       ! USE DAC CHANNEL 8
NBPS    .EQU    513     ! NUMBER OF BREAKPOINTS FOR SINE FUNCTION.
T       .DEFINE 0,1,2   ! ARP TEMPORARY REGISTERS.
X       .EQU    3       ! ARP REGISTER FOR INPUT VARIABLE "X".
I       .EQU    0       ! MAP/DEP BREAKPOINT INDEX REGISTER.
SIN     .EQU    0::0    ! DATA MEMORY ADDRESS OF SINE FUNCTION DATA.
!
        .COP
BEGIN   PFI     3,ADC                   !0      INITIATE ADC CONVERSION.
        PAUSE   63                      !1      WAIT 10 MICRO-SECONDS
        RFR                             !65     FOR CONVERSION TO COMPLETE.
        GIF     2,ADC                   !101    READ CONVERTED VALUE.
        NOP                             !102
        START   $DEP                    !103
        START   $ARP                    !104
        PAUSE   3                       !105
        START   $MAP  ;  STOP $DEP      !109
        NOP                             !110
        STOP    $MAP                    !111
        PAUSE   5                       !112
        PFI     3,DAC                   !118    SET DAC = SIN(X)
        STOP    $ARP                    !119
        HLT     0                       !120    HALT (MAYBE).
        LPC     $DEP,0                  !121    RESET PROGRAM COUNTERS.
        LPC     $MAP,0                  !122
        LPC     $ARP,0                  !123
        JMP     BEGIN                   !124    GO DO IT AGAIN...
!
        .DEP
        LSI     32768/(NBPS-1)                  !104    SEND CONSTANTS FOR
        LSI     (NBPS-1)/2                      !105    SHIFT SEARCH AND DELTA
        LSI     32768/((NBPS-1)/2) ; PAUSE 2    !106    CALCULATION IN ARP.
        SIS     I                               !109    STORE BP INDEX.
!
        .MAP
        RAID    SIN,I ; PAUSE 1                 !110    READ FUNCTION VALUES.
!
        .ARP
                MOV0        ; MOV1          ; MOV2 S,B,R ; MOV3 R,T0     !105
                MOV0 S,A,R  ; MOV1 R,X      ; MOV2 S,C,D,R;MOV3 R,T1     !106
 FA (A-B)*C+D; MOV0         ; MOV1          ; MOV2 S,R   ; MOV3 R,T2     !107
                MOV0 T1,B   ; MOV1 T2,C     ; MOV2       ; MOV3 R,A,L    !108
 IA (A-B)*C  ; MOV0         ; MOV1          ; MOV2       ; MOV3          !109
                MOV0 T1,C   ; MOV1 X,A      ; MOV2       ; MOV3 R,B      !110
 IA (A-B)*C  ; MOV0         ; MOV1          ; MOV2       ; MOV3          !111
                MOV0        ; MOV1          ; MOV3 R,C   ; PAUSE 2       !112
                MOV0 S,B,D  ; MOV1          ; MOV2 S,A   ; MOV3          !115
 FA (A-B)*C+D; MOV0         ; MOV1          ; MOV2       ; MOV3          !116
                MOV0        ; MOV1          ; MOV2       ; MOV3          !117
                MOV0        ; MOV1 R,L      ; MOV2       ; PAUSE 1       !118
        .END
```

```
LOCATION OBJECT CODE                                                LINE# SOURCE STATEMENT

                                                                      1 !  AD-10 SINEWAVE PROGRAM (SINE.ASM)
                                                                      2 !
                                                                      3 !  1) READS "X" FROM AN ADC CHANNEL.
                                                                      4 !  2) PERFORMS A SHIFT SEARCH AND DELTA CALCULATION ON "X".
                                                                      5 !  3) INTERPOLATES FOR A ONE VARIABLE FUNCTION SIN(X).
                                                                      6 !  4) SETS DAC = SIN(X)
                                                                      7 !  5) REPEATS STEPS 1) THROUGH 4) EVERY 12.5 MICRO-SECONDS.
                                                                      8 !
                                                                      9         .DECIMAL        ! CONSTANTS WILL BE DECIMAL NUMBERS.
0000000                                                              10 ADC     .EQU    0       ! USE ADC CHANNEL 0
0000010                                                              11 DAC     .EQU    8       ! USE DAC CHANNEL 8
0001001                                                              12 NBPS    .EQU    513     ! NUMBER OF BREAKPOINTS FOR SINE FUNCTION.
                                                                     13 T       .DEFINE 0,1,2   ! ARP TEMPORARY REGISTERS.
0000003                                                              14 X       .EQU    3       ! ARP REGISTER FOR INPUT VARIABLE "X".
0000000                                                              15 I       .EQU    0       ! MAP/DEP BREAKPOINT INDEX REGISTER.
0000000                                                              16 SIN     .EQU    0::0    ! DATA MEMORY ADDRESS OF SINE FUNCTION DATA.
                                                                     17 !
                                                                     18         .COP
0000000  000000 100600                                               19 BEGIN   PFI     3,ADC                !0      INITIATE ADC CONVERSION.
0000001  000000 000077                                               20         PAUSE   63                   !1      WAIT 10 MICRO-SECONDS
0000002  000000 011400                                               21         RFR                          !65     FOR CONVERSION TO COMPLETE.
0000003  000000 106400                                               22         GIF     2,ADC                !101    READ CONVERTED VALUE.
0000004  000000 000000                                               23         NOP                          !102
0000005  000000 040010                                               24         START   $DEP                 !103
0000006  000000 040040                                               25         START   $ARP                 !104
0000007  000000 000003                                               26         PAUSE   3                    !105
0000010  000000 040006                                               27         START   $MAP  ;  STOP $DEP   !109
0000011  000000 000000                                               28         NOP                          !110
0000012  000000 040001                                               29         STOP    $MAP                 !111
0000013  000000 000005                                               30         PAUSE   5                    !112
0000014  000000 100610                                               31         PFI     3,DAC                !118    SET DAC = SIN(X)
0000015  000000 040020                                               32         STOP    $ARP                 !119
0000016  000000 010400                                               33         HLT     0                    !120    HALT (MAYBE).
0000017  000000 120000                                               34         LPC     $DEP,0               !121    RESET PROGRAM COUNTERS.
0000020  000000 110000                                               35         LPC     $MAP,0               !122
0000021  000000 130000                                               36         LPC     $ARP,0               !123
0000022  000000 020000                                               37         JMP     BEGIN                !124    GO DO IT AGAIN...
                                                                     38 !
                                                                     39         .DEP
0000000  140001 000100                                               40         LSI     32768/(NBPS-1)               !104   SEND CONSTANTS FOR
0000001  140001 000400                                               41         LSI     (NBPS-1)/2                   !105   SHIFT SEARCH AND DELTA
0000002  142001 000200                                               42         LSI     32768/((NBPS-1)/2) ; PAUSE 2 !106   CALCULATION IN ARP.
0000003  000000 003400                                               43         SIS     I                            !109   STORE BP INDEX.
                                                                     44 !
                                                                     45         .MAP
0000000  000071 000000 000000                                        46         RAID    SIN,I ; PAUSE 1              !110   READ FUNCTION VALUES.
                                                                     47 !
                                                                     48         .ARP
0000000  000000 000000 000000 110200 040200                          49                 MOV0       ; MOV1       ; MOV2 S,B,R ; MOV3 R,T0     !105
0000001  000000 120200 040203 106200 040201                          50                 MOV0 S,A,R ; MOV1 R,X   ; MOV2 S,C,D,R;MOV3 R,T1     !106
0000002  000236 000000 000000 100200 040202                          51  FA (A-B)*C+D; MOV0       ; MOV1       ; MOV2 S,R   ; MOV3 R,T2     !107
0000003  000000 150001 144002 000000 060400                          52                 MOV0 T1,B  ; MOV1 T2,C  ; MOV2       ; MOV3 R,A,L    !108
0000004  003036 000000 000000 000000 000000                          53  IA (A-B)*C  ; MOV0       ; MOV1       ; MOV2       ; MOV3          !109
0000005  000000 144001 160003 000000 050000                          54                 MOV0 T1,C  ; MOV1 X,A   ; MOV2       ; MOV3 R,B      !110
0000006  003036 000000 000000 000000 000000                          55  IA (A-B)*C  ; MOV0       ; MOV1       ; MOV2       ; MOV3          !111
0000007  020000 000000 000000 000000 044000                          56                 MOV0       ; MOV1       ; MOV3 R,C   ; PAUSE 2       !112
```

```
LOCATION OBJECT CODE

0000010  000000 112000 000000 120000 000000
0000011  000236 000000 000000 000000 000000
0000012  000000 000000 000000 000000 000000
0000013  010000 000000 040400 000000 000000
```

```
LINE# SOURCE STATEMENT

   57                MOVO S,B,D ; MOV1       ; MOV2 S,A   ; MOV3         !115
   58  FA (A-B)*C+D; MOVO       ; MOV1       ; MOV2       ; MOV3         !116
   59                MOVO       ; MOV1       ; MOV2       ; MOV3         !117
   60                MOVO       ; MOV1 R,L   ; MOV2       ; PAUSE 1      !118
   61          .END
```

TIME	CURRENT STMT COP	CURRENT STMT MAP	CURRENT STMT DEP	CURRENT STMT ARP	FIRST SOURCE COP	FIRST SOURCE MAP	FIRST SOURCE DEP	FIRST SOURCE ARP	FIRST SINK COP	FIRST SINK MAP	FIRST SINK DEP	FIRST SINK ARP	SECOND SOURCE COP	SECOND SOURCE MAP	SECOND SOURCE DEP	SECOND SOURCE ARP	SECOND SINK COP	SECOND SINK MAP	SECOND SINK DEP	SECOND SINK ARP	COMMENT(S)
0	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	20	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	0	0	0	0	NSOF
65	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
100	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
101	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
102	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
103	25	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
104	26	0	41	49	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	49	
105	26	0	42	50	22	0	0	0	0	0	0	50	0	0	41	0	0	0	0	50	
106	26	0	0	51	0	0	0	0	0	0	0	0	0	0	42	0	0	0	0	51	
107	26	0	0	52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
108	27	0	43	53	0	0	0	0	0	0	0	0	0	0	0	52	0	0	43	0	
109	28	46	0	54	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
110	29	0	0	55	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
111	30	0	0	56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
112	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
113	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
114	30	0	0	57	0	46	0	0	0	0	0	57	0	46	0	0	0	0	0	57	
115	30	0	0	58	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
116	30	0	0	59	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
117	31	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
118	32	0	0	0	0	0	0	60	31	0	0	0	0	0	0	0	0	0	0	0	
119	33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
120	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
121	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
122	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
123	37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

	CURRENT STMT				FIRST SOURCE				FIRST SINK				SECOND SOURCE				SECOND SINK				
TIME	COP	MAP	DEP	ARP	COP	MAP	DEP	ARP	COP	MAP	DEP	ARP	COP	MAP	DEP	ARP	COP	MAP	DEP	ARP	COMMENT(S)
0	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	20	0	0	0	19	0	0	0	19	0	0	0	0	0	0	0	0	0	0	0	
65	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
100	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
101	23	0	0	0	22	0	0	0	22	0	0	0	0	0	0	0	0	0	0	0	
102	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
103	25	0	40	0	24	0	0	0	0	0	24	0	0	0	0	0	0	0	0	0	
104	26	0	41	49	25	0	0	0	0	0	0	25	0	0	0	0	0	0	0	0	
105	26	0	42	50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
106	26	0	0	51	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
107	26	0	0	52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
108	27	0	43	53	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
109	28	46	0	54	27	0	0	0	0	27	27	0	0	0	0	0	0	0	0	0	
110	29	0	0	55	0	46	0	0	0	46	0	0	0	46	0	0	0	46	0	0	
111	30	0	0	56	29	0	0	0	0	29	0	0	0	0	0	0	0	0	0	0	
112	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
113	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
114	30	0	0	57	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
115	30	0	0	58	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
116	30	0	0	59	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
117	31	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
118	32	0	0	0	31	0	0	0	31	0	0	0	0	0	0	0	0	0	0	0	
119	33	0	0	0	32	0	0	0	0	0	0	32	0	0	0	0	0	0	0	0	
120	34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
121	35	0	0	0	34	0	0	0	0	0	34	0	0	0	0	0	0	0	0	0	
122	36	0	0	0	35	0	0	0	0	35	0	0	0	0	0	0	0	0	0	0	
123	37	0	0	0	36	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	

SYMBOL	VALUE	PROC	STMT	REFERENCES		
ADC	0000000	DAT	10	19	22	
BEGIN	0000000	COP	19	37		
DAC	0000010	DAT	11	31		
I	0000000	DAT	15	43	46	
NBPS	0001001	DAT	12	40	41	42
SIN	0000000	DAT	16	46		
T0	0000000	DAT	13	49		
T1	0000001	DAT	13	50	52	54
T2	0000002	DAT	13	51	52	
X	0000003	DAT	14	50	54	

THIS IS A LISTING OF THE LOAD MODULE (SINE.MOD) FOR THIS PROGRAM :

```
COP          0      23
     0 100600       0     77       0  11400       0 106400       0       0
     0  40010       0  40040       0      3       0  40006       0       0
     0  40001       0      5       0 100610       0  40020       0   10400
     0 120000       0 110000       0 130000       0  20000
MAP          0       1
    71       0       0
DEP          0       4
140001     100  140001     400  142001     200       0   3400
ARP          0      14
     0       0       0 110200   40200       0  120200  40203  106200  40201
   236       0       0 100200   40202       0  150001 144002       0  60400
  3036       0       0      0       0       0  144001 160003       0  50000
  3036       0       0      0       0   20000       0      0       0  44000
     0  112000       0 120000       0     236       0      0       0      0
     0       0       0      0       0   10000       0  40400       0      0
END
```

```
DDDDDDDD     IIIII          A            GGGGG
DDDDDDDDDD   IIIII        AAA         GGGGGGGGG
DDDDDDDDDDD   III        AAAAA       GGGGGGGGGGG
DDDDDDDDDDDD  III       AAAAAAA      GGGGG
DDDDDDDDDDDD  III      AAAAAAAAA     GGGGG GGGGG       THE
DDDDDDDDDDDD  III     AAAAAAAAAAA    GGGGG GGGGG       AD-10
DDDDDDDDDDD   III    AAAAAAAAAAAAA   GGGGGGGGGG        DIAGNOSTIC
DDDDDDDDDD   IIIII  AAAAAAAAAAAAAAA   GGGGGGGGG        PACKAGE
DDDDDDDD     IIIII AAAAAAAAAAAAAAAAA    GGGGG
```

TABLE OF CONTENTS PAGE

TABLE OF CONTENTS

INTRODUCTION

THE AD-10 DIAGNOSTIC PACKAGE CONSISTS OF A GROUP OF PROGRAMS DESIGNED TO BE USED TO CHECK FOR (OR TO VERIFY) AD-10 COMPUTER SYSTEM HARDWARE PROBLEMS. THE DIAGNOSTIC PROGRAMS ARE WRITTEN IN PDP-11 MACRO ASSEMBLY LANGUAGE AND ARE DESIGNED TO BE PART OF DEC'S XXDP DIAGNOSTIC PACKAGE. THE DIAGNOSTIC ROUTINES ARE ORGANIZED INTO THE FOLLOWING CATEGORIES :

A. AD-10 DATA MEMORY DIAGNOSTICS (ME01,ME02,ME03,ME05,ME23,ME30).

B. AD-10 PROCESSOR DIAGNOSTICS (ARP: AR06,AR14,AR15;
   COP: CO11,CO21,CO22,CO24,CO25,
   CO26;
   DEP: DE08,DE17,DE18,DE19,DE20;
   MAP: MA07,MA09,MA10,MA31).

C. IOCC DIAGNOSTICS (IO27,IO28).

D. HIC DIAGNOSTICS (HI12,HI13,HI32)

XXDP OVERVIEW

DEC'S XXDP ("XX DIAGNOSTIC PACKAGE") IS THEIR COLLECTION OF DIAGNOSTIC ROUTINES ON FILE-ORIENTED MEDIA. THE "XX" IS REPLACED BY DEC'S CODE FOR THE PARTICULAR DEVICE (E.G., "RK" FOR RK05 DISK DRIVE). XXDP PROVIDES A COMPACT MEANS FOR STORING THE NUMEROUS DIAGNOSTIC PROGRAMS. IT ALLOWS THE USER TO LOAD AND RUN A DIAGNOSTIC PROGRAM UNDER KEYBOARD CONTROL VIA THE XXDP MONITOR. THIS MONITOR ROUTINE ALSO PROVIDES THE MEANS FOR UPDATING AND MODIFYING PROGRAMS, AND ALLOWS THE USER TO "CHAIN" A SERIES OF PROGRAMS TOGETHER, SO THAT WHEN ONE IS FINISHED THE NEXT ONE WILL BEGIN EXECUTION. THE BASIC XXDP MONITOR COMMANDS ARE :

F	SETS CONSOLE FILL COUNT (FOR LA30 TERMINAL)
D	PRINTS DIRECTORY ON CONSOLE
D/F	PRINTS SHORT DIRECTORY ON CONSOLE
D/L	PRINTS DIRECTORY ON LINE PRINTER
D/L/F	PRINTS SHORT DIRECTORY ON LINE PRINTER
R COPY	RUNS COPY PROGRAM (TO MAKE A COPY OF THE XXDP DISK)
R FILE	RUNS ANY OTHER PROGRAM ON THE DISK (THE PROGRAM IS IN THE SPECIFIED FILE)
L FILE	LOADS ANY PROGRAM ON THE DISK (FROM THE SPECIFIED FILE)
S	STARTS THE LOADED PROGRAM
C FILE	RUNS A CHAIN OF PROGRAMS (THE CHAIN PARAMETERS ARE SPECIFIED IN THE FILE)
C FILE/QV	RUNS A CHAIN IN "QUICK VERIFY" MODE

WHEN THE XXDP MONITOR IS BOOTED , IT WILL INDICATE A RESTART ADDRESS FOR USE AFTER ERROR (OR USER) HALTS.

THE XXDP PACKAGE CONTAINS TWO UPDATE PROGRAMS CALLED UPD1.BIN (4K) AND UPD2.BIN (8K). THESE PROGRAMS ARE USED TO ADD, DELETE, RENAME, OR PATCH PROGRAMS ON THE XXDP PACKAGE AND TO PROVIDE FILE MAINTENANCE SERVICES. UPD1.BIN IS A SUBSET OF UPD2.BIN .

REFER TO DEC'S XXDP MANUAL FOR FURTHER INFORMATION .

NOTE : IF AN EARLIER VERSION OF DEC'S XXDP WAS USED TO CREATE THE AD-10 DIAGNOSTIC PACKAGE, THE MONITOR WILL NOT ACCEPT AS MANY COMMANDS AS INDICATED ABOVE. THE AVAILABLE COMMANDS WILL BE :

/D	PRINTS FULL DIRECTORY ON TERMINAL
/D/F	PRINTS SHORT DIRECTORY ON TERMINAL
/D/L	PRINTS FULL DIRECTORY ON LINE PRINTER
/R	TO RUN DISK COPY PROGRAM
FILE	TO RUN A PROGRAM IN THE SPECIFIED FILE

PROGRAM NAMING CONVENTIONS

THE CONVENTIONS USED IN NAMING THE AD-10 DIAGNOSTICS ARE SOMEWHAT DIFFERENT FROM THE DEC CONVENTIONS. AD-10 DIAGNOSTIC PROGRAM NAMES ARE FORMED BY A TWO CHARACTER TEST-TYPE DESIGNATION, FOLLOWED BY A TWO CHARACTER NUMERIC SEQUENCE NUMBER (E.G., "AR06"). THIS FOUR CHARACTER CORE IS FOLLOWED BY A SINGLE CHARACTER VERSION INDICATION (E.G., "A"). THE PROGRAM FILENAMES ALL HAVE THE XXDP STANDARD ".BIC" EXTENSION. THEREFORE, AN OPERATIONAL VERSION OF "AR06" WILL BE FOUND IN THE XXDP DIRECTORY AS "AR06A.BIC" . THE TEST-TYPE DESIGNATIONS ARE AS FOLLOWS :

AR	ARP (ARITHMETIC PROCESSOR)
CO	COP (CONTROL PROCESSOR)
DE	DEP (DECISION PROCESSOR)
MA	MAP (MEMORY ADDRESS PROCESSOR)
HI	HIC (HOST INTERFACE CONTROLLER)
IO	IOCC (I/O CHANNEL CONTROLLER)
ME	AD-10 DATA MEMORY

SWITCH SETTINGS

SWR BIT	"SET" CONDITION
15 (100000)	HALT ON ERROR (NOT RSX-11 VERSION)
14 (40000)	SCOPE LOOP ON ERROR (NOT ALL DIAGNOSTICS *)
13 (20000)	SUPPRESS ERROR MESSAGE PRINTOUT
12 (10000)	-
11 (4000)	TERMINATE AFTER CURRENT PROGRAM LOOP
10 (2000)	LONG ERROR PRINTOUT (DUMPS HISTORY REGISTERS)
9 (1000)	QUICK EXIT FROM LONG LOOP PROGRAMS
8 (400)	-
7 (200)	USE INTERNAL DAC/ADC TABLE FOR IOCC TESTS

* A SCOPE LOOP ON ERROR IS ONLY ALLOWED IN THOSE DIAGNOSTICS WHICH DO NOT PERMIT THE AD-10 TO RUN FREELY. THE FOLLOWING TESTS DO PERMIT A SCOPE LOOP ON ERROR :

ME01
ME02
ME03
ME05
AR06
MA07
DE08
CO11

RUNNING A PROGRAM

1. BOOT THE XXDP DEVICE TO GET THE XXDP MONITOR UP AND RUNNING .

2. GET A DIRECTORY, IF NECESSARY, TO FIND OUT WHAT THE PROGRAM NAMES ARE :

   ./D (OR "/D/F", OR "/D/L", OR "/D/L/F")
   (THE "." IS THE MONITOR'S PROMPT CHARACTER)

3. SET THE PDP-11 CONSOLE SWITCHES TO THE APPROPRIATE SWITCH SETTINGS (SEE PRECEDING TABLE).

4. SELECT THE DESIRED DIAGNOSTIC PROGRAM (E.G., "DE17A.BIC" FOR A TEST OF THE DEP'S X AND I REGISTERS), AND TYPE "R XXXXX" TO RUN THE PROGRAM, WHERE "XXXXX" IS THE PROGRAM NAME AND VERSION, BUT NOT THE EXTENSION (NOTE THAT FOR THE EARLIER VERSION OF XXDP JUST "XXXXX" IS SUFFICIENT) :

   E.G., .R DE17A <CR> (OR, .DE17A <CR>)

```
AD10 DEP INDEX REGISTER TEST
THE FOLLOWING AD-10 PROCESSORS ARE PRESENT
MAP     000001
DEP     000002
ARP     000003
COP     000007
PRESS CONTINUE WHEN READY
```

   PRESS THE CONTINUE SWITCH ON THE PDP-11 TO START THE TEST. THE PROGRAM WILL PRINT THE "PRESS CONTINUE WHEN READY" MESSAGE AGAIN WHEN THE TEST IS COMPLETE. IF BIT 13 OF THE SWITCH REGISTER IS NOT SET AND NO ERROR MESSAGES WERE PRINTED, NO AD-10 ERRORS WERE DETECTED.

5. IF AN ERROR HALT OCCURS, THE MONITOR MAY BE RESTARTED AT THE ADDRESS SPECIFIED WHEN IT BOOTED, THE AD-10 DIAGNOSTIC PROGRAM CAN CONTINUE (IF THE USER MANUALLY PRESSES THE CONTINUE SWITCH ON THE PDP-11), OR THE DIAGNOSTIC CAN BE RESTARTED AT LOCATION 200.

SIGNIFICANT CORE LOCATIONS

THE USER MAY WANT TO EXAMINE AND/OR CHANGE THE TEST-RELATED CONSTANTS IN THESE SPECIFIED LOCATIONS :

PAT=220		MEMORY TEST PATTERN
NPAT=222		ERROR COUNT FOR THIS ITERATION
ITR=224		NUMBER OF ITERATIONS TO RUN THIS TEST
BACK=226		MEMORY BACKGROUND PATTERN
BRD=230	*	BOARD NUMBER TO BE TESTED
MOD=232	*	STARTING MODULE NUMBER FOR THIS TEST
WIN=236	*	STARTING WINDOW NUMBER FOR THIS TEST
WRD=236	*	STARTING WORD NUMBER FOR THIS TEST
WINC=240	*	NUMBER OF WINDOWS TO BE TESTED
WRDC=242	*	NUMBER OF WORDS TO BE TESTED
DEV=1112		OUTPUT DEVICE CSR LOCATION (CONSOLE TERMINAL IS 177564, LINE PRINTER IS 177514)

* NOTE : THESE LOCATIONS APPLY TO MEMORY TESTS ONLY.

ERROR REPORTING

ERROR MESSAGES ARE PRINTED BY EACH TEST, ACCORDING TO THE CONSOLE SWITCH SETTINGS. SEE THE TEST DESCRIPTIONS FOR FURTHER INFORMATION. IF SWITCH 10 IS SET, THE AD-10 HISTORY REGISTERS WILL BE DUMPED ALONG WITH ANY ERROR MESSAGE, IN THE FOLLOWING FORMAT :

HB00	HB01	HB02	HB03
HB04	HB05	HB06	HB07
HB10	HB11	HB12	HB13
HB14	HB15	HB16	HB17
HA00	HA01	HA02	HA03
HA04	HA05	HA06	HA07
HA10	HA11	HA12	HA13
HA14	HA15	HA16	HA17
HD00	HD01	HD02	HD03
HD04	HD05	HD06	HD07
HD10	HD11	HD12	HD13
HD14	HD15	HD16	HD17

EXECUTION TIMES

MOST TESTS WILL RUN ANYWHERE FROM 1 TO 3 MINUTES, BUT DE19, DE20, ME23, AND CO26 WILL TAKE AT LEAST 5 MINUTES PER PASS.

ME01

DESCRIPTION : AD-10 DATA MEMORY 17-BIT ROTATE TEST.

THIS MEMORY TEST FILLS 4K OF DATA MEMORY AT A TIME, AND READS BACK THE SAME 4K FOR VERIFICATION. CONSECUTIVE LOCATIONS ARE LOADED WITH A PATTERN WHICH ROTATES AS A 17-BIT WORD AS FOLLOWS :

AD-10 LOCATION	PATTERN	PDP-11 CARRY BIT
0	11111111111111111	0
1	01111111111111111	1
2	10111111111111111	1
3	11011111111111111	1
4	11101111111111111	1
(ETC.)		
13	11111111111110111	1
14	11111111111111011	1
15	11111111111111101	1
16	11111111111111110	1
17	11111111111111111	0
(ETC.)		

ERROR MESSAGE : AD10 MEMORY ERROR

WBA	CURRENT WORD BLOCK ADDRESS CONTENTS
LOC	LOCATION IN ERROR
SENT	EXPECTED DATA PATTERN
REC	RECEIVED DATA PATTERN

DESCRIPTION : AD-10 DATA MEMORY ALTERNATING PATTERN TEST.

THIS MEMORY TEST FILLS 4K OF DATA MEMORY AT A TIME, THEN READS BACK THE SAME 4K FOR VERIFICATION. CONSECUTIVE LOCATIONS ARE LOADED WITH A 16 BIT ALTERNATING PATTERN AS FOLLOWS :

LOCATION	PATTERN
0	1010101010101010
1	0101010101010101
2	1010101010101010
3	0101010101010101
(ETC.)	

THE STARTING PATTERN IS COMPLEMENTED AFTER EACH ITERATION SO THAT ALL BITS IN EACH LOCATION WILL BE CHECKED.

ERROR MESSAGE : AD10 MEMORY ERROR

WBA	CURRENT WORD BLOCK ADDRESS CONTENTS
LOC	LOCATION IN ERROR
SENT	EXPECTED DATA PATTERN
REC	RECEIVED DATA PATTERN

ME03

DESCRIPTION : AD-10 DATA MEMORY MARCHING PATTERN TEST.

THIS IS A BASIC TEST OF A MEMORY TO PROVIDE REASONABLE ASSURANCE THAT IT IS FUNCTIONAL, I.E., THAT THE ADDRESSING IS OPERATIONAL AND THAT EACH LOCATION CAN BE READ AND WRITTEN TO THE ALL-ZERO STATE. FIRST, A 256 WORD "WINDOW" IS FILLED WITH A "BACKGROUND" PATTERN. THEN, STARTING AT THE SPECIFIED STARTING WINDOW ADDRESS AND PROCEDING SEQUENTIALLY, THE BACKGROUND PATTERN IS READ AND THE SPECIFIED PATTERN IS WRITTEN. THIS PROCEDURE CONTINUES TO THE LAST LOCATION, AT WHICH POINT THE PATTERN IS COMPLEMENTED AND THE ADDRESS IS SEQUENTIALLY REDUCED UNTIL THE FIRST LOCATION IS REACHED. THE BACKGROUND PATTERN IS THEN COMPLEMENTED, AND THE SEQUENCE IS REPEATED. THE TEST WILL BE DONE FOR UP TO 16 WINDOWS OF 256 WORDS (4K). THIS TEST BY NO MEANS CHECKS EVERYTHING (OR ALL INTERACTIONS) BUT DOES PROVIDE REASONABLE ASSURANCE THAT NO DEFECTIVE ELEMENTS ARE PRESENT.

ERROR MESSAGE : AD10 MEMORY ERROR

WBA	CURRENT WORD BLOCK ADDRESS CONTENTS
LOC	MEMORY LOCATION IN ERROR
SENT	EXPECTED DATA PATTERN
REC	RECEIVED DATA PATTERN

ME05

DESCRIPTION : AD-10 DATA MEMORY GALLOPING PATTERN TEST.

THIS TEST CHECKS ALL POSSIBLE ADDRESS TRANSITIONS. ALL LOCATIONS WITHIN THE STARTING 256 WORD WINDOW ARE WRITTEN TO THE BACKGROUND PATTERN. A TEST PATTERN IS THEN WRITTEN TO A "LOAD WORD". THE SEQUENCE "READ THE LOAD WORD, READ ANOTHER WORD" IS PERFORMED UNTIL THE ENTIRE WINDOW HAS BEEN CHECKED. THE LOAD WORD IS THEN RESET, A NEW LOAD WORD IS CHOSEN, AND THE SEQUENCE IS REPEATED. THIS PROCESS CONTINUES UNTIL ALL 256 WORDS HAVE BEEN LOAD WORDS. THE BACKGROUND PATTERN IS THEN COMPLEMENTED AND THE ENTIRE PROCEDURE IS REPEATED FOR THIS WINDOW. WHEN FINISHED WITH THIS WINDOW, REPEAT FOR EACH WINDOW .

ERROR MESSAGE : AD10 MEMORY ERROR

WBA	CURRENT WORD BLOCK ADDRESS CONTENTS
LOC	LOCATION ADDRESS WITHIN THE WINDOW
SENT	EXPECTED DATA PATTERN
REC	RECEIVED DATA PATTERN

ME23

DESCRIPTION : AD-10 DATA MEMORY PARITY TEST.

THIS TEST CHECKS EACH WORD OF MEMORY FOR CORRECT PARITY GENERATION. A DECREMENTING DATA PATTERN IS USED.

ERROR MESSAGE : DATA ERROR

ADDR	CURRENT MEMORY ADDRESS WITHIN THE PAGE
OFFSET	OCTAL OFFSET *
SENT	EXPECTED DATA PATTERN
REC	RECEIVED DATA PATTERN

* OFFSET FROM THE START OF THE HISTORY DATA REGISTER TO THE CURRENT DATA VALUE. THE FOLLOWING TABLE CORRELATES THE OFFSET WITH THE PAGE NUMBER :

IF DATA IS EVEN	IF DATA IS ODD
22 --> 0	22 --> 1
20 --> 1	20 --> 0
16 --> 2	16 --> 3
14 --> 3	14 --> 2
12 --> 4	12 --> 5
10 --> 5	10 --> 4
6 --> 6	6 --> 7
4 --> 7	4 --> 6

ERROR MESSAGE : PARITY ERROR

ADDR	(AS ABOVE)
OFFSET	(AS ABOVE)
FLAG	CURRENT PARITY BIT-PATTERN
MBUS PARITY	CURRENT MULTIBUS ADDRESS VALUE (BIT 15 IS THE PARITY BIT) (BIT 13 IS THE PARITY ERROR BIT)

ME30

DESCRIPTION : PROCESSOR TEMPORARY REGISTER MEMORY TEST.

FOR EACH PROCESSOR PRESENT IN THE AD-10, THE TEMPORARY REGISTER STORAGE IS TESTED USING AN INCREMENTING MEMORY TEST. THIS CHECKS DATA READ/WRITE THROUGH THE HIC FOR ALL FIELD 5 MEMORY LOCATIONS. THE REGISTERS TESTED ARE :

COP	GENERAL REGISTERS
ARP	T REGISTERS
MAP	I REGISTER
DEP	X REGISTERS

ERROR MESSAGE : AD10 MEMORY ERROR

WBA	WORD BLOCK ADDRESS (PROCESSOR)
LOC	LOCATION WITHIN THE WINDOW
SENT	EXPECTED DATA PATTERN
REC	RECEIVED DATA PATTERN

AR06

DESCRIPTION : ARP PROGRAM MEMORY INCREMENTING PATTERN TEST.

THIS PROGRAM PERFORMS AN INCREMENTING PATTERN TEST ON THE FIVE FIELDS OF THE ARP PROGRAM MEMORY AS FOLLOWS (FOR EACH FIELD) :

ARP LOCATION	PATTERN
0	0000000000000000
1	0000000000000001
2	0000000000000010
3	0000000000000011
4	0000000000000100
(ETC.)	
1020	0000001111111100
1021	0000001111111101
1022	0000001111111110
1023	0000001111111111

AFTER EACH ITERATION THE STARTING PATTERN IS INCREMENTED BY 200 (OCTAL), SO THAT ALL BITS IN EACH LOCATION WILL BE THOROUGHLY CHECKED AFTER 1000 ITERATIONS.

ERROR MESSAGE : AD10 MEMORY ERROR

WBA	CURRENT WORD BLOCK ADDRESS CONTENTS
LOC	LOCATION WITHIN THE FIELD
SENT	EXPECTED DATA PATTERN
REC	RECEIVED DATA PATTERN

AR14

DESCRIPTION :

ARP DATA PATH TEST.

THIS TEST EXERCISES THE FOUR ARP DATA PATHS WITH A ROTATING DATA PATTERN. THE DATA PATHS ARE :

1. S(B) -> A S=A P=S Q=P R=Q R -> L (=A)

2. S(B) -> B S=B P=S Q=P R=Q R -> L (=B)

3. S(B) -> C S=1 P=S*C Q=P R=Q R -> L (=C)

4. S(B) -> D S=0 P=S Q=P+E R=Q R -> L (=D)

ERROR MESSAGE : ARP PATH ERROR

PATH	1 = REGISTER A PATH
	2 = REGISTER B PATH
	3 = REGISTER C PATH
	4 = REGISTER D PATH
ARITH	0 = FRACTIONAL ARITHMETIC
	1000 = FRACTIONAL * 2
	2000 = FRACTIONAL / 2
	3000 = INTEGER
SENT	EXPECTED DATA PATTERN
REC	RECEIVED DATA PATTERN

AR15

DESCRIPTION :

ARP ARITHMETIC INSTRUCTION TEST.

THIS TEST CHECKS EACH OF THE FOUR SETS OF ARITHMETIC INSTRUCTIONS IN THE ARP, WHILE THE OTHER THREE ARE NOT IN USE. THIS CHECK IS DONE WITH A STATIC SET OF DATA CONSTANTS. THE TESTS ARE :

1. EXERCISE S, WITH: P=S, Q=P, R=Q
   WHERE, S=1,0,B,-B,A,A+1,A+B,A-B

2. EXERCISE P, WITH: S=1, Q=P, R=Q
   WHERE, P=0,S,-S,S*C,-S*C,
   S*ABS(C),-S*ABS(C)

3. EXERCISE Q, WITH: S=1, P=S, R=Q
   WHERE, Q=P,P+E,P-E

ERROR MESSAGE : ARP INST ERROR

XXXX	(NOT USED)
INST	OCTAL CODE FOR ARP ARITHMETIC INSTRUCTION
SENT	EXPECTED DATA PATTERN
REC	RECEIVED DATA PATTERN

CO11

DESCRIPTION :

COP PROGRAM MEMORY INCREMENTING PATTERN TEST.

THIS PROGRAM PERFORMS AN INCREMENTING PATTERN TEST ON THE 1024 WORD, 2 FIELD COP PROGRAM MEMORY AS FOLLOWS (FOR EACH FIELD) :

COP LOCATION	PATTERN
0	0000000000000000
1	0000000000000001
2	0000000000000010
3	0000000000000011
4	0000000000000100
(ETC.)	
1020	0000001111111100
1021	0000001111111101
1022	0000001111111110
1023	0000001111111111

AFTER EACH ITERATION THE STARTING PATTERN IS INCREMENTED BY 200 (OCTAL), SO THAT ALL BITS IN EACH LOCATION WILL BE THOROUGHLY CHECKED AFTER 1000 (OCTAL) ITERATIONS.

ERROR MESSAGE : AD10 MEMORY ERROR

WBA	CURRENT WORD BLOCK ADDRESS CONTENTS
LOC	LOCATION WITHIN THE FIELD
SENT	EXPECTED DATA PATTERN
REC	RECEIVED DATA PATTERN

CO21

DESCRIPTION :

COP LOAD IMMEDIATE INSTRUCTION TEST.

THIS TEST EXERCISES THE LOAD FIRST, LOAD SECOND, AND LOAD DOUBLE INSTRUCTIONS USING A ROTATING DATA PATTERN. THE COP PLACES THE CURRENT DATA PATTERN ON THE AD-10 MULTIBUS USING "LOAD IMM" INSTRUCTIONS. THE PDP-11 CHECKS FOR THIS DATA IN THE AD-10 HISTORY REGISTERS.

ERROR MESSAGE : INST ERROR

TYPE	1 = LOAD DOUBLE 2 = LOAD DOUBLE 3 = LOAD SECOND 4 = LOAD FIRST
XXXX	(NOT USED)
SENT	EXPECTED DATA PATTERN
REC	RECEIVED DATA PATTERN

C022

DESCRIPTION :

COP CONDITION BIT AND CONDITIONAL JUMP TEST.

THIS PROGRAM TESTS THE SETTING AND CLEARING OF THE CBIT, AND THE PROPER DETECTION OF THE CBIT WITH A CONDITIONAL JUMP. A COP PROGRAM LOOP CONSISTING OF CBIT SET/CLEAR AND CONDITIONAL JUMPS IS EXECUTED. IF AN ERROR OCCURS, THE COP PLACES PREDETERMINED DATA ONTO THE AD-10 MULTIBUS, WHICH IS DETECTED BY THE PDP-11 IN THE HISTORY REGISTERS.

TESTS:
1. CBIT SET/CLEAR
2. BUS CONDITIONAL CBIT SET/CLEAR
3. JUMP AND CONDITIONAL JUMP

ERROR MESSAGE : INST ERROR

TYPE	DATA CHECK INDEX
TEST	TEST TABLE INDEX
DATA	WORKING TEST DATA PATTERN
ERROR FLAG	EXPECTED DATA PATTERN

ERROR MESSAGE : JMP 0 ERROR, RC NOT -1 RC = XXXX

(WHERE, XXXX IS THE CURRENT RUN COUNTER VALUE)

THIS ERROR MEANS THAT THE COP "JMP" INSTRUCTION FAILED.

C024

DESCRIPTION :

COP GENERAL REGISTER TEST.

ERROR MESSAGE : INST ERROR

TYPE	TEST TABLE INDEX
REG	REGISTER NUMBER
SENT	EXPECTED DATA PATTERN
REC	RECEIVED DATA PATTERN

C025

DESCRIPTION : COP HALT0 AND HALT1 TEST.

C025 TESTS THE VARIOUS COMBINATIONS OF THE HALT MASK IN COMBINATION WITH THE HALT0 AND HALT1 INSTRUCTIONS. THE PAUSE INSTRUCTION IS ALSO CHECKED.

ERROR MESSAGE : HALT ERROR

TYPE	-
RC	RUN COUNTER VALUE
EPC	EXPECTED PROGRAM COUNTER
PC	RECEIVED PROGRAM COUNTER

ERROR MESSAGE : PAUSE ERROR

EXRC	EXPECTED RUN COUNTER VALUE
RC	ACTUAL RUN COUNTER VALUE
PAUSE	PAUSE COUNT FOR THIS RUN
XXXX	(NOT USED)

C026

DESCRIPTION : COP PROGRAM COUNTER AND RUN TEST.

1. TESTS THE LOADING AND READING OF PROCESSOR PC'S (INCLUDING THE COP) THROUGH THE HIC.
2. TESTS THE LOADING OF PROCESSOR PC'S THROUGH COP PROGRAMMING.
3. CHECKS PROPER PC INCREMENTING WHEN THE PROCESSOR IS RUNNING.

ERROR MESSAGE : HIC PC LOAD ERROR (TEST 1 MESSAGE)

PRO	PROCESSOR NUMBER
EPC	EXPECTED PC
PC	ACTUAL PC
XXXX	(NOT USED)

ERROR MESSAGE : COP PC LOAD ERROR (TEST 2 MESSAGE)

PRO	PROCESSOR NUMBER
EPC	EXPECTED PC
PC	ACTUAL PC
XXXX	(NOT USED)

ERROR MESSAGE : PC RUN ERROR (TEST 3 MESSAGE)

PRO	PROCESSOR NUMBER
EPC	EXPECTED PC
PC	ACTUAL PC
XXXX	(NOT USED)

DE08

DESCRIPTION : DEP PROGRAM MEMORY INCREMENTING PATTERN TEST.

THIS PROGRAM PERFORMS AN INCREMENTING PATTERN TEST ON THE 1024 WORD, 2 FIELD DEP PROGRAM MEMORY AS FOLLOWS (FOR EACH FIELD) :

DEP LOCATION	PATTERN
0	0000000000000000
1	0000000000000001
2	0000000000000010
3	0000000000000011
4	0000000000000100
(ETC.)	
1020	0000001111111100
1021	0000001111111101
1022	0000001111111110
1023	0000001111111111

AFTER EACH ITERATION THE STARTING PATTERN IS INCREMENTED BY 200 (OCTAL), SO THAT ALL BITS IN EACH LOCATION WILL BE THOROUGHLY CHECKED AFTER 1000 (OCTAL) ITERATIONS.

ERROR MESSAGE : AD10 MEMORY ERROR

WBA	CURRENT WORD BLOCK ADDRESS CONTENTS
LOC	LOCATION WITHIN THE FIELD
SENT	EXPECTED DATA PATTERN
REC	RECEIVED DATA PATTERN

DE17

DESCRIPTION :

DEP X AND I REGISTER TEST.

THIS TEST LOADS ALL DEP X AND I REGISTER LOCATIONS (USING AD10 PROGRAMMING) WITH A DATA PATTERN. THE PATTERN IS ROTATED AND CHECKED. THIS CHECKS THESE INSTRUCTIONS :

LIF
LIS
LXF
LXS
SIF
SIS
SXF
SXS
SSI

ALSO TESTS THE DATA HOLDING ABILITY OF THE X AND I REGISTER LOCATIONS.

ERROR MESSAGE : INST ERROR

TYPE	0 = I REGISTER, LOAD 1ST
	24 = I REGISTER, LOAD 2ND
	50 = X REGISTER, LOAD 1ST
	74 = X REGISTER, LOAD 2ND
REG	REGISTER NUMBER
SENT	EXPECTED DATA PATTERN
REC	RECEIVED DATA PATTERN

DE18

DESCRIPTION :

DEP SPECIAL INSTRUCTION TEST.

DE18 TESTS THE DEP SPECIAL INSTRUCTIONS FOR A SINGLE DATA POINT. INSTRUCTIONS TESTED ARE :

LFI
LSI
LDI

LFI 100000
LSI 100000

ERROR MESSAGE : INST ERROR

TYPE	0 = LOAD 2ND, LOAD 1ST
	1 = LOAD DOUBLE
	2 = LOAD 2**15 (100000)
XXXX	(NOT USED)
SENT	EXPECTED DATA PATTERN
REC	RECEIVED DATA PATTERN

DE19

DESCRIPTION : DEP REGISTER TO REGISTER INSTRUCTION TEST.

DE19 TESTS THE X & I INTRA-REGISTER TRANSFER INSTRUCTIONS WITH A ROTATING DATA PATTERN, USING COP AND DEP PROGRAMMING.

ERROR MESSAGE : INST ERROR

TYPE	0 = I TO I TRANSFER
	20 = I TO X TRANSFER
	40 = X TO X TRANSFER
	60 = X TO I TRANSFER
REGS	LOW BYTE = SOURCE REGISTER
	HIGH BYTE = DESTINATION REGISTER
SENT	EXPECTED DATA PATTERN
REC	RECEIVED DATA PATTERN

DE20

DESCRIPTION : DEP COMPARE AND COMPARE & MODIFY INSTRUCTION TEST.

THIS TEST CHECKS THE COMPARE INSTRUCTION AND THE COMPARE AND MODIFY INSTRUCTION FOR ALL REGISTERS AND FOR VARIOUS DATA VALUES. THE CONDITION INSTRUCTION IS ALSO CHECKED, USING 2**15 -> DM(1).

ERROR MESSAGE : INST ERROR

TYPE	0 = CMM TEST, BOTH 1ST AND 2ND
	1 = CMP TEST, BOTH 1ST AND 2ND
REG/DATA	LOW BYTE = INDEXED REGISTER
	HIGH BYTE = X REGISTER USED IN COMPARISON
SENT	EXPECTED DATA PATTERN
REC	RECEIVED DATA PATTERN

MA07

DESCRIPTION : MAP PROGRAM MEMORY INCREMENTING PATTERN TEST.

MA07 PERFORMS AN INCREMENTING PATTERN TEST ON THE 1024 WORD, 3 FIELD MAP PROGRAM MEMORY AS FOLLOWS (FOR EACH FIELD) :

MAP LOCATION	PATTERN
0	0000000000000000
1	0000000000000001
2	0000000000000010
3	0000000000000011
4	0000000000000100
(ETC.)	
1020	0000001111111100
1021	0000001111111101
1022	0000001111111110
1023	0000001111111111

AFTER EACH ITERATION THE STARTING PATTERN IS INCREMENTED BY 200 (OCTAL), SO THAT ALL BITS IN EACH LOCATION WILL BE THOROUGHLY CHECKED AFTER 1000 (OCTAL) ITERATIONS.

ERROR MESSAGE : AD10 MEMORY ERROR

WBA	CURRENT WORD BLOCK ADDRESS
LOC	LOCATION WITHIN THE FIELD
SENT	EXPECTED DATA PATTERN
REC	RECEIVED DATA PATTERN

MA09

DESCRIPTION : MAP NON-INDEXED INSTRUCTION TEST.

MA09 CHECKS ALL NON-INDEXED OPCODES WITH WAIT COUNTS. A ROTATING PATTERN ADDRESS IS EXERCISED WITH EACH OPCODE, AND WAIT COUNTS 0-7 ARE EXERCISED WITH EACH OPCODE.

ERROR MESSAGE : MAP INST ERROR

SENT1	EXPECTED PATTERN ON HI ADDRESS MULTIBUS
ADDR1	ACTUAL PATTERN ON HI ADDRESS MULTIBUS
SENT2	EXPECTED PATTERN ON LOW ADDRESS MULTIBUS
ADDR2	ACTUAL PATTERN ON LOW ADDRESS MULTIBUS

MA10

DESCRIPTION : MAP INDEXED INSTRUCTION TEST.

MA10 CHECKS ALL INDEXED OPCODES WITH WAIT COUNTS. A ROTATING PATTERN ADDRESS IS EXERCISED WITH EACH OPCODE, AS ARE WAIT COUNTS 0-7. EACH INDEX REGISTER IS USED AND CONTAINS ITS OWN ADDRESS.

ERROR MESSAGE : MAP INST ERROR

SENT1	EXPECTED PATTERN ON HI ADDRESS MULTIBUS
ADDR1	ACTUAL PATTERN ON HI ADDRESS MULTIBUS
SENT2	EXPECTED PATTERN ON LOW ADDRESS MULTIBUS
ADDR2	ACTUAL PATTERN ON LOW ADDRESS MULTIBUS

MA31

DESCRIPTION : MAP SINGLE INDEX REGISTER INSTRUCTION TEST.

MA31 CHECKS ALL INDEXED OPCODES WITH WAIT COUNTS 0-7 AND A ROTATING 19-BIT ADDRESS PATTERN. A SINGLE INDEX REGISTER IS USED. THE REGISTER'S CONTENTS ARE POWERS OF 2 (1'S BIT WALK).

ERROR MESSAGE : MAP INST ERROR

SENT1	EXPECTED PATTERN ON HI ADDRESS MULTIBUS
ADDR1	ACTUAL PATTERN ON HI ADDRESS MULTIBUS
SENT2	EXPECTED PATTERN ON LOW ADDRESS MULTIBUS
ADDR2	ACTUAL PATTERN ON LOW ADDRESS MULTIBUS

I027

DESCRIPTION: IOCC BUFFER AND ADC/DAC LOOP TEST.

1. TESTS IOCC LOAD BUFFER AND READ BUFFER INSTRUCTIONS :
   A. PFB, PIBL, PIBH, GIB
   B. A POWERS OF 2 BIT PATTERN TESTS THE BUFFER BIT INTEGRITY.

2. IOCC DAC/ADC LOOP TEST
   A. TESTS PFI, GIF.
   B. A POWERS OF 2 BIT PATTERN IS SENT TO A DAC AND READ FROM AN ADC.

THIS TEST IS CHAINABLE ONLY IF THE DAC/ADC TABLE IS ASSEMBLED INTO THE TEST CORRECTLY. THERE IS NO KEYBOARD INPUT IN CHAIN MODE.

ADC TOLERANCE IS 20 (OCTAL) BY DEFAULT (1 LSB). THIS VALUE IS AT SYMBOL 999$ IN THIS TEST (SEE THE LISTING FOR THE PDP-11 LOCATION).

ERROR MESSAGE : ADC READ ERROR

TYPE	0 = 1ST READ IN HISTORY REGISTER 1 = 2ND READ IN HISTORY REGISTER
DAC/ADC	THE OCTAL DAC/ADC PAIR ADDRESSES (BYTES)
SENT	EXPECTED DATA PATTERN
REC	RECEIVED DATA PATTERN

ERROR MESSAGE : BUFFER ERROR

TYPE	-
XXXX	(NOT USED)
SENT	EXPECTED DATA PATTERN
REC	RECEIVED DATA PATTERN

IO28

DESCRIPTION : IOCC ADC/DAC LOOP TEST.

EACH DAC/ADC PAIR IS LOOP TESTED FOR EVERY POSSIBLE LEGAL BIT COMBINATION.

THIS TEST IS CHAINABLE ONLY IF THE DAC/ADC TABLE IS ASSEMBLED INTO THE TEST. THERE IS NO KEYBOARD INPUT IN CHAIN MODE.

ADC TOLERANCE IS 20 (OCTAL) BY DEFAULT (1 LSB). THIS VALUE IS AT SYMBOL 999$ IN THIS TEST (SEE THE LISTING FOR THE PDP-11 LOCATION).

ERROR MESSAGE : ADC READ ERROR

TYPE	-
DAC/ADC	THE OCTAL DAC/ADC PAIR ADDRESSES (BYTES)
SENT	EXPECTED DATA PATTERN
REC	RECEIVED DATA PATTERN

ERROR MESSAGE : BUFFER ERROR

TYPE	-
XXXX	(NOT USED)
SENT	EXPECTED DATA PATTERN
REC	RECEIVED DATA PATTERN

HI12

DESCRIPTION : HIC AND COP BIT WALK TEST.

A BIT WALK OF 1'S AND 0'S IS PERFORMED ON ALL PERMISSIBLE HIC AND COP REGISTERS. (READ/WRITE BITS ONLY).

ERROR MESSAGE : AD10 HIC REG ERROR

ADDR	HIC ADDRESS AT WHICH ERROR OCCURRED
MASKED	ORIGINAL PATTERN BEFORE MASK
SENT	EXPECTED DATA PATTERN
REC	RECEIVED DATA PATTERN

HI13

DESCRIPTION : HIC PROCESSOR BOARD PROGRAM COUNTER EXERCISER AND HISTORY REGISTER/TEST REGISTER TESTS.

EACH PROCESSOR BOARD PROGRAM COUNTER HAS ALL VALUES (0-1777) WRITTEN TO IT AND VERIFIED. ALL PROCESSORS MUST BE PRESENT FOR THIS TEST. ALSO, THE TEST REGISTER IS LOADED, THE AD10 IS RUN FOR EIGHT INSTRUCTIONS, AND THE HISTORY REGISTER'S CONTENTS CHECKED AGAINST THE ORIGINAL TEST REGISTER'S CONTENTS.

ERROR MESSAGE : AD10 HIC REG ERROR

TEST	TEST REGISTER DATA
HIST	HISTORY REGISTER DATA
OFFSET	THE OFFSET INTO THE HISTORY REGISTER DATA FOR THIS COMPARISON
XXXX	(NOT USED)

HI32

DESCRIPTION : HIC SHUTDOWN REGISTER TEST.

MAP AND COP PROGRAMS ARE USED TO FILL THE MULTIBUS PIPELINE. THE DATA COMES FROM EIGHT PAGES OF DATA MEMORY AND THE COP GENERAL REGISTERS. WHEN THE AD-10 IS STOPPED, THE SHUTDOWN REGISTERS ARE LOADED WITH THE PIPE-LINE DATA. THESE REGISTERS ARE THEN READ AND CHECKED. EACH DATA VALUE IS INCREMENTED UNTIL ALL BIT PATTERNS ARE TESTED IN EACH SHUTDOWN REGISTER.

ERROR MESSAGE : SHUT DOWN ERROR

REG ADDR	HIC ADDRESS OF SHUTDOWN REGISTER
XXXX	(NOT USED)
EXPECTED	EXPECTED DATA PATTERN
RECEIVED	RECEIVED DATA PATTERN

APPLIED DYNAMICS INTERNATIONAL
3800 STONE SCHOOL ROAD
ANN ARBOR, MICHIGAN 48104
313-973-1300

```
HHHHH  HHHHH  LLLL      IIIII  BBBBBBBBBB
HHHHH  HHHHH  LLLL      IIIII  BBBBBBBBBBB
HHHHH  HHHHH  LLLL       III   BBBBBBBBBB
HHHHHHHHHHHH  LLLL       III   BBBBBBBBB
HHHHHHHHHHHH  LLLL       III   BBBBBBB
HHHHHHHHHHHH  LLLL       III   BBBBBBBBB
HHHHH  HHHHH  LLLL       III   BBBBBBBBBB
HHHHH  HHHHH  LLLLLLLLL IIIII  BBBBBBBBBB
HHHHH  HHHHH  LLLLLLLLL IIIII  BBBBBBBBB
```

THE HIC-11
COMMUNICATION
SUBROUTINE LIBRARY
DE1478

TABLE OF CONTENTS PAGE

INTRODUCTION

THE HIC-11 COMMUNICATION SUBROUTINE LIBRARY (HIC.OLB) PROVIDES THE MEANS FOR USER PROGRAMS WRITTEN IN FORTRAN OR MACRO-11 TO COMMUNICATE WITH THE AD-10 COMPUTER. THE AD-10 EXECUTIVE (ADX) USES THESE HIC LIBRARY ROUTINES TO ACCESS AND CONTROL THE AD-10. THE HIC LIBRARY IS AN OBJECT LIBRARY AND NEEDS TO BE LINKED TO THE USER'S OBJECT PROGRAM AT TASK BUILD TIME. IT CONSISTS OF A GROUP OF FORTRAN SUBROUTINES, FROM WHICH THE TASK BUILDER WILL SELECT THE ONES WHICH ARE CALLED BY THE USER'S PROGRAM AND WILL INCLUDE THEM IN THE RESULTING RSX-11 TASK. THESE SUBROUTINES ENABLE THE USER'S PROGRAM TO DO THE FOLLOWING:

1. ATTACH AD-10 CONSOLES FOR EXCLUSIVE USE (MANDATORY FOR ACCESS TO THE AD-10), SWITCH FROM ONE CONSOLE TO ANOTHER, AND DETACH CONSOLES (ATT10,CON10,DET10),

2. INITIALIZE THE AD-10 AND HIC REGISTERS (INTHIC,INIT10),

3. START THE AD-10 (RUN10),

4. STOP THE AD-10 (HLT10),

5. PUT THE AD-10 INTO TEST MODE (TEST10),

6. DETERMINE IF THE AD-10 IS RUNNING (BUSY10),

7. READ FROM OR WRITE TO HIC REGISTERS (RHICR,WHICR,RHICRS, WHICRS),

8. READ FROM OR WRITE TO AD-10 DATA MEMORY (RPM,WPM,RDMS,WDMS),

9. READ FROM OR WRITE TO AD-10 PROGRAM MEMORY FOR EACH AD-10 PROCESSOR (RPM,WPM,RPMS,WPMS),

10. READ FROM OR WRITE TO THE 256-WORD BUS WINDOW (RBW,WBW, RBWS,WBWS),

11. READ FROM OR WRITE TO INDIVIDUAL HIC AND AD-10 REGISTERS (RTCR,WTCR,RTSH,WTSH,RRIC,WRIC,RCSR,WCSR,REHS,WEHS,RHMR, WHMR,RIMR,WIMR,RRCR,WRCR,RBAR,WBAR,RRR,WRR),

12. READ FROM OR WRITE TO EITHER A SINGLE REGISTER IN A REGISTER GROUP, OR TO THE ENTIRE REGISTER GROUP (RTB,WTB,RTBS,WTBS, RTA,WTA,RTAS,WTAS,RTD,WTD,RTDS,WTDS,RPC,WPC,RPCS,WPCS,RSC, WSC,RSCS,WSCS,RSD,WSD,RSDS,WSDS,RHB,WHB,RHBS,WHBS,RHA,WHA, RHAS,WHAS,RHD,WHD,RHDS,WHDS,RGR,WGR,RGRS,WGRS,RIR,WIR,RIRS, WIRS,RXR,WXR,RXRS,WXRS,RTR,WTR,RTRS,WTRS).

USING HIC LIBRARY SUBROUTINES

THE PROGRAM PREPARATION STEPS INVOLVED IN USING THE HIC LIBRARY ROUTINES TO COMMUNICATE WITH OR CONTROL THE AD-10 ARE AS FOLLOWS :

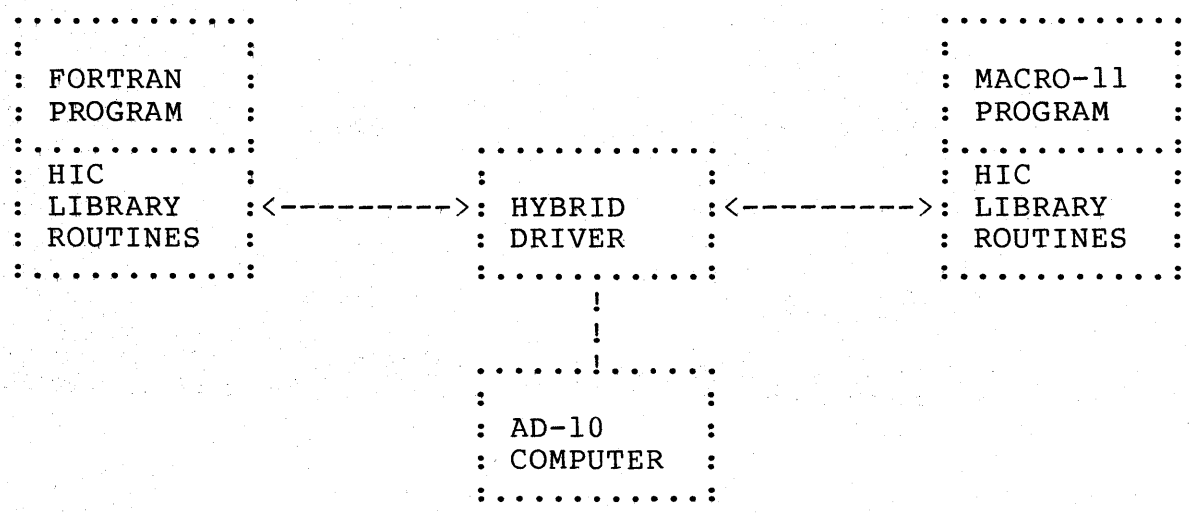
1. DEFINE THE PROBLEM AND DO THE "PAPER CODING" FOR THE SOURCE LANGUAGE BEING USED (FORTRAN OR MACRO-11).

2. USE THE TEXT EDITOR (EDI) TO CREATE A SOURCE FILE.

3. COMPILE (FORTRAN) OR ASSEMBLE (MACRO-11) THE SOURCE FILE TO CREATE AN OBJECT FILE.

4. TASK BUILD (TKB) THE OBJECT FILE ALONG WITH THE HIC-11 COMMUNICATION SUBROUTINE LIBRARY ([1,1]HIC.OLB), TO CREATE A TASK IMAGE FILE WHICH CAN BE RUN UNDER RSX-11M.

THE HIC LIBRARY SUBROUTINES INTERFACE TO THE THE AD-10 THROUGH THE HYBRID DRIVER (HY), WHICH MUST BE LOADED IN THE RSX-11M SYSTEM PRIOR TO RUNNING THE ANY TASK WHICH CALLS THESE SUBROUTINES. TO ALLOW THE HYBRID DRIVER TO ACCESS THE AD-10 INTERFACE "WNDWS=1" MUST BE SPECIFIED AS AN OPTION TO THE TASKBUILDER. PRIOR TO ANY COMMUNICATION WITH THE AD-10, THE USER TASK MUST SUCESSIFULLY "ATTACH" THE AD-10 CONSOLE OF INTEREST BY CALLING THE HIC LIBRARY ROUTINE "ATT10". THE FOLLOWING DIAGRAM INDICATES THE GENERAL TASK STRUCTURE:



THE FOLLOWING EXAMPLE SHOWS THE BASIC CONSOLE COMMANDS FOR TASKBUILDING AND RUNNING A FORTRAN OR MACRO-11 PROGRAM WHICH CALLS HIC LIBRARY SUBROUTINES :

```
MCR>TKB <CR>
TKB>PROG,PROG/SH=PROG,[1,1]HIC/LB <CR>
TKB>/ <CR>
ENTER OPTIONS:
TKB>WNDWS=1
TKB>// <CR>

MCR>RUN PROG <CR>
```

NOTE : "<CR>" INDICATES CARRIAGE RETURN (RETURN).
---- THROUGHOUT THIS MANUAL ANGLE BRACKETS ("<" AND ">") ARE USED TO ENCLOSE THE NAME OF A SYNTACTIC ELEMENT OR CLASS OF ELEMENTS.

ARGUMENTS AND CALLING CONVENTIONS

ALL ARGUMENTS FOR THE HIC LIBRARY SUBROUTINES ARE TYPE INTEGER*2 , EXCEPT FOR THE FUNCTION SUBPROGRAM BUSY10(I), WHERE THE SINGLE ARGUMENT IS TYPE LOGICAL*1 . THE ARGUMENT "IE" IN MOST ROUTINES CONTAINS THE ERROR CODE UPON RETURNING FROM THE ROUTINE, WHERE IE=0 INDICATES SUCCESS AND IE=1 INDICATES AN ERROR CONDITION.

WHEN CALLING THE SUBROUTINES FROM FORTRAN, SIMPLY FOLLOW THE STANDARD FORTRAN PROCEDURES :

```
CALL <HIC LIBRARY SUBROUTINE NAME> (<ARGUMENT LIST>)
```

SAMPLE FORTRAN CALLS ARE INCLUDED IN THE INDIVIDUAL ROUTINE DESCRIPTIONS.

THE ONLY EXCEPTION IS THE FUNCTION SUBPROGRAM BUSY10(I), WHICH RETURNS A LOGICAL VALUE (.TRUE. OR .FALSE., DEPENDING UPON WHETHER OR NOT THE AD-10 IS CURRENTLY RUNNING). A SAMPLE "CALL" TO THIS FUNCTION SUBPROGRAM WOULD BE AS FOLLOWS :

```
      LOGICAL*1 BUSY10,I
      IF (BUSY10(I)) GOTO 100
      ...
100   ...
```

WHEN CALLING A HIC LIBRARY SUBROUTINE FROM MACRO-11, THE USER MUST DO THE FOLLOWING :

1. CREATE A TABLE CONTAINING THE ADDRESSES OF THE ARGUMENTS :

```
TABLE:  .BYTE   0, <COUNT OF THE NUMBER OF ARGUMENTS>
        .WORD   <ADDRESS OF ARGUMENT #1>
        .WORD   <ADDRESS OF ARGUMENT #2>
        ...
        .WORD   <ADDRESS OF ARGUMENT #N>
```

2. THEN, THE CALL TO THE SUBROUTINE IS MADE AS FOLLOWS :

```
        <SAVE REGISTERS R0-R5>
        ;
        MOV     #TABLE,R5
        JSR     PC, <HIC LIBRARY SUBROUTINE NAME>
        ;
        <RESTORE REGISTERS R0-R5>
```

NOTE : UPON RETURNING FROM FUNCTION SUBPROGRAM BUSY10, REGISTER R0 WILL CONTAIN THE LOGICAL RESULT (AS WILL THE ARGUMENT, WHICH STILL MUST BE SPECIFIED) :

```
R0 = 0          IMPLIES .FALSE.
R0 NONZERO      IMPLIES .TRUE.
```

ROUTINE NAME(S): ATT10, DET10, CON10

CALL SEQUENCE(S): CALL ATT10(IC,IL,IEFN,IE)
CALL CON10(IC,IE)
CALL DET10(IE)

DESCRIPTION: ATTACH AD-10 CONSOLE FOR EXCLUSIVE USE (AND SELECT CONSOLE)
SELECT AD-10 CONSOLE (IN A MULTI-CONSOLE SYSTEM)
DETACH AD-10 CONSOLE CURRENTLY SELECTED

INPUT VARIABLES: IC = CONSOLE #
IL = LOGICAL UNIT #
IEFN = EVENT FLAG #

OUTPUT VARIABLES: IE = ERROR WORD
= 0 NORMALLY
= 1 FOR ILLEGAL ARGUMENT

ROUTINE NAME(S): HLT10, INTHIC, INIT10, RUN10, TEST10

CALL SEQUENCE(S): CALL HLT10(IE)
CALL INTHIC(IE)
CALL INIT10(IE)
CALL RUN10(IE)
CALL TEST10(IE)

DESCRIPTION: HALT THE AD-10
INITIALIZE ALL HIC REGISTERS WHICH ALLOW WRITE ACCESS
INITIALIZE THE AD-10
RUN THE AD-10
PUT THE AD-10 IN TEST MODE

INPUT VARIABLES: NONE

OUTPUT VARIABLES: IE = 0 NORMALLY
= 1 IF AD-10 POWER IS DOWN

```
ROUTINE NAME(S):  RHICR, WHICR, RHICRS, WHICRS
---------------

CALL SEQUENCE(S): CALL RHICR(REG,IDATA1,IE)
----------------  CALL WHICR(REG,IDATA2,IE)

                  CALL RHICRS(REG,IARY1,ICNT,IE)
                  CALL WHICRS(REG,IARY2,ICNT,IE)

DESCRIPTION:      READ FROM OR WRITE TO HIC REGISTER(S)
-----------

INPUT VARIABLES:  REG    = HIC REGISTER NUMBER (0-255)
---------------   IDATA2 = DATA WORD TO WRITE
                  IARY2  = ARRAY OF DATA WORDS TO WRITE
                  ICNT   = NUMBER OF REGISTERS TO READ OR WRITE
                           (STARTING WITH "REG") (1-256)

OUTPUT VARIABLES: IDATA1 = DATA WORD TO READ
----------------  IARY1  = ARRAY OF DATA WORDS READ
                  IE     = ERROR WORD
                         = 0  NORMALLY
                         = 1  FOR ILLEGAL ARGUMENT VALUE


ROUTINE NAME(S):  BUSY10(I)
---------------

CALL SEQUENCE(S): LOGICAL*1 BUSY10,I,BUSY
----------------  BUSY=BUSY10(I)

DESCRIPTION:      FUNCTION SUBPROGRAM WHICH RETURNS :
-----------

                  .TRUE.        IF AD-10 IS BUSY (RUN BIT IN CSR IS SET)
                  .FALSE.       IF AD-10 IS NOT BUSY
                                (RUNBIT IN CSR IS NOT SET)

INPUT VARIABLES:  NONE
---------------

OUTPUT VARIABLES: I = .TRUE. OR .FALSE. (THE SAME AS THE FUNCTION VALUE)
----------------
```

ROUTINE NAME(S): RDM, WDM, RDMS, WDMS

CALL SEQUENCE(S):
```
CALL RDM(PAGADR,WRDADR,IDATA1,IE)
CALL WDM(PAGADR,WRDADR,IDATA2,IE)

CALL RDMS(PAGADR,WRDADR,IARY1,ICNT,IE)
CALL WDMS(PAGADR,WRDADR,IARY2,ICNT,IE)
```

DESCRIPTION: READ OR WRITE A SINGLE DATA MEMORY LOCATION OR A GROUP OF LOCATIONS

INPUT VARIABLES:
```
PAGADR = MEMORY PAGE ADDRESS (0-63)
WRDADR = MEMORY WORD ADDRESS (0-4095)
IDATA2 = DATA WORD TO WRITE
IARY2  = ARRAY OF DATA WORDS TO WRITE
ICNT   = NUMBER OF DATA MEMORY WORDS TO READ OR WRITE
         (STARTING WITH THE SPECIFIED ADDRESS)
```

OUTPUT VARIABLES:
```
IDATA1 = DATA WORD READ
IARY1  = ARRAY OF DATA WORDS READ
IE     = ERROR WORD
       = 0  NORMALLY
       = 1  FOR ILLEGAL ARGUMENT(S)
```

ROUTINE NAME(S): RPM, WPM, RPMS, WPMS

CALL SEQUENCE(S):
```
CALL RPM(PROC,FLDNUM,WRDADR,IDATA1,IE)
CALL WPM(PROC,FLDNUM,WRDADR,IDATA2,IE)

CALL RPMS(PROC,FLDNUM,WRDADR,IARY1,ICNT,IE)
CALL WPMS(PROC,FLDNUM,WRDADR,IARY2,ICNT,IE)
```

DESCRIPTION: READ FROM OR WRITE TO AD-10 PROGRAM MEMORY

INPUT VARIABLES:
```
PROC   = AD-10 PROCESSOR NUMBER  (1-7)
FLDNUM = FIELD NUMBER IN PROCESSOR MEMORY WORD
         (FROM 0 TO THE MAXIMUM FOR THAT PROCESSOR,
         OR FIELD 5 FOR REGISTER ACCESS)
WRDADR = PROGRAM MEMORY WORD ADDRESS (0-1023)
IDATA2 = DATA WORD TO WRITE
IARY2  = ARRAY OF DATA WORDS TO WRITE
ICNT   = NUMBER OF WORDS TO READ OR WRITE TO
         FIELD "FLDNUM" OF PROGRAM MEMORY WORDS
         (STARTING WITH "WRDADR") (1-1024)
```

OUTPUT VARIABLES:
```
IDATA1 = DATA WORD READ
IARY1  = ARRAY OF DATA WORDS READ
IE     = ERROR WORD
       = 0  NORMALLY
       = 1  FOR ILLEGAL ARGUMENT(S)
```

ROUTINE NAME(S): RBW, WBW, RBWS, WBWS

CALL SEQUENCE(S): CALL RBW(IADDR,IDATA1,IE)
CALL WBW(IADDR,IDATA2,IE)

CALL RBWS(IADDR,IARY1,ICNT,IE)
CALL WBWS(IADDR,IARY2,ICNT,IE)

DESCRIPTION: READ FROM OR WRITE TO BUS WINDOW LOCATION(S)

INPUT VARIABLES:
- IADDR = BUS WINDOW LOCATION (0-255)
- IDATA2 = DATA WORD TO WRITE
- IARY2 = ARRAY OF DATA WORDS TO WRITE
- ICNT = NUMBER OF BUS WINDOW LOCATIONS TO READ OR WRITE (STARTING WITH LOCATION "IADDR") (1-256)

OUTPUT VARIABLES:
- IDATA1 = DATA WORD READ
- IARY1 = ARRAY OF DATA WORDS READ
- IE = ERROR WORD
  - = 0 NORMALLY
  - = 1 FOR ILLEGAL ARGUMENT(S)

ROUTINE NAME(S): R***, W***

CALL SEQUENCE(S): R***(IDATA1)
W***(IDATA2)

DESCRIPTION: READ FROM OR WRITE TO REGISTER : ***

WHERE, *** CAN BE ANY OF THE FOLLOWING MNEMONICS:

MNEMONIC	REGISTER
TCR	TEST CONTROL REGISTER
TSH	TEST/SHUTDOWN/HISTORY COUNTERS (READ ONLY)
RIC	REMOTE INTERFACE CONTROL REGISTER
CSR	CONTROL STATUS REGISTER
EHS	ERROR AND HALT STATUS REGISTER
HMR	HALT MASK REGISTER
IMR	INTERRUPT MASK REGISTER
RCR	RUN COUNT REGISTER
BAR	BUS WINDOW BLOCK ADDRESS REGISTER
RR	ARP "R" (RESULT) REGISTER
DR	NIP "D" REGISTER

INPUT VARIABLES: IDATA2 = DATA WORD TO WRITE

OUTPUT VARIABLES: IDATA1 = DATA WORD READ

ROUTINE NAME(S): R**, W**, R**S, W**S

CALL SEQUENCE(S): CALL R**(N,IDATA1,IE)
CALL W**(N,IDATA2,IE)

CALL R**S(IARY1)
CALL W**S(IARY2)

DESCRIPTION: READ FROM OR WRITE TO THE REGISTERS IN REGISTER GROUP : **

WHERE, ** CAN BE ANY OF THE FOLLOWING MNEMONICS:

MNEMONIC	REGISTER GROUP	REGISTER NUMBERS
TB	TEST BLOCK ADDRESS	(0-15)
TA	TEST ADDRESS AND CONTROL	(0-15)
TD	TEST DATA	(0-15)
PC	PROCESSOR PROGRAM COUNTERS	(1-7)
PS	PROCESSOR STATUS WORDS	(1-7)
SC	SHUTDOWN/RESTART CONDITIONS	(0-15)
SD	SHUTDOWN/RESTART DATA	(0-15)
HB	HISTORY BLOCK ADDRESS	(0-15)
HA	HISTORY ADDRESS/ERROR	(0-15)
HD	HISTORY DATA	(0-15)

INPUT VARIABLES:
N = REGISTER NUMBER IN GROUP
(ALL REGISTERS ARE NUMBERED STARTING WITH 0 EXCEPT FOR PC'S AND PS'S WHICH ARE NUMBERED FROM 1 TO 7)
IDATA2 = DATA WORD TO WRITE
IARY2 = ARRAY OF DATA WORDS TO WRITE

OUTPUT VARIABLES:
IDATA1 = DATA WORD READ
IARY1 = ARRAY OF DATA WORDS READ
IE = ERROR WORD
= 0 NORMALLY
= 1 FOR REGISTER NUMBER OUT OF RANGE

ROUTINE NAME(S): R**, W**, R**S, W**S

CALL SEQUENCE(S):
```
CALL R**(N,IDATA1,IE)
CALL W**(N,IDATA2,IE)

CALL R**S(N,IARY1,ICNT,IE)
CALL W**S(N,IARY2,ICNT,IE)
```

DESCRIPTION: READ FROM OR WRITE TO THE REGISTERS IN REGISTER GROUP : **

WHERE, ** CAN BE ANY OF THE FOLLOWING MNEMONICS:

MNEMONIC	REGISTER GROUP	REGISTER #
GR	COP GENERAL REGISTER	(0-127)
IR	MAP/DEP "I" REGISTER	(0-127)
XR	DEP "X" REGISTER	(0-127)
TR	ARP "T" REGISTER	(0-127)

INPUT VARIABLES:
```
N      = TEMPORARY REGISTER ADDRESS (0-127)
IDATA2 = DATA WORD TO WRITE
ICNT   = THE NUMBER OF TEMPORARY REGISTERS
IARY2  = DESIRED TEMPORARY REGISTER VALUES
```

OUTPUT VARIABLES:
```
IARY1 = AN ARRAY WHICH WILL RECEIVE THE
        TEMPORARY REGISTER VALUES
IDATA1 = DATA WORD READ
IE     = ERROR WORD
       = 0  NORMALLY
       = 1  FOR ILLEGAL ARGUMENT(S)
```

ROUTINE NAME(S): RFR, WFR, RFRS, WFRS

CALL SEQUENCE(S):
```
CALL RFR(FLDNUM,REG,IDATA1,IE)
CALL WFR(FLDNUM,REG,IDATA2,IE)

CALL RFRS(FLDNUM,REG,IARY1,ICNT,IE)
CALL WFRS(FLDNUM,REG,IARY2,ICNT,IE)
```

DESCRIPTION: READ FROM OR WRITE TO THE 48 BIT NIP REGISTERS

INPUT VARIABLES:
```
FLDNUM = 1 (LOW), 2 (MIDDLE), OR 3 (HIGH 16 BIT WORD)
REG    = REGISTER NUMBER (0-1023 DECIMAL)
IDATA2 = 16 BIT DATA WORD TO WRITE
IARY2  = ARRAY OF 16 BIT DATA WORDS TO WRITE
ICNT   = NUMBER OF 16 BIT DATA WORDS TO READ OR WRITE
         STARTING WITH REGISTER "REG" (1-1024)
```

OUTPUT VARIABLES:
```
IDATA1 = 16 BIT DATA WORD READ
IARY1  = ARRAY OF 16 BIT DATA WORDS READ
IE     = ERROR WORD
       = 0  NORMALLY
       = 1  FOR ILLEGAL ARGUMENT(S)
```

```
MMMM     MMMM  FFFFFFFFFF  LLLL        IIII  BBBBBBBB
MMMMM   MMMMM  FFFFFFFFFF  LLLL        IIII  BBBBBBBBBB
MMMMMMMMMMMMM  FFFFFFFFFF  LLLL        IIII  BBBB  BBBBB
MMMMMMMMMMMMM  FFFF        LLLL        IIII  BBBB  BBBBB
MMMMMMMMMMMMM  FFFFFFFF    LLLL        IIII  BBBBBBBBB      THE
MMMM MM MMMM   FFFFFFFF    LLLL        IIII  BBBBBBBBB      AD-10
MMMM    MMMM   FFFFFFFF    LLLL        IIII  BBBB  BBBBB    MACROFILE
MMMM    MMMM   FFFF        LLLLLLLLLL  IIII  BBBB  BBBBB    LIBRARY
MMMM    MMMM   FFFF        LLLLLLLLLL  IIII  BBBBBBBBBB     USER'S MANUAL
MMMM    MMMM   FFFF        LLLLLLLLLL  IIII  BBBBBBBB
```

AD-10 DOCUMENTATION UPDATES

MANUAL NAME: AD 10 SOFTWARE REFERENCE MANUAL CHAPTER 5 MFLIB

DATE: 3-FEB-78

UPDATES:

1) THE EXAMPLE AD-10 PROGRAM "BENCH.ASM" HAS AN ERROR ON THE BOTTOM OF PAGE 57. THE SYMBOL "UPDATE" SHOULD BE DEFINED TO HAVE A VALUE OF 1 INSTEAD OF 3, AS FOLLOWS:

```
UPDATE  .EQU  1
```

2) THE FORTRAN PROGRAM "FUNDAT.FTN" ASSOCIATED WITH THE EXAMPLE AD-10 PROGRAM HAS SEVERAL ERRORS. THE ARGUMENT TO THE "SIN" FUNCTION AND "COS" FUNCTION ON PAGES 64 AND 65 SHOULD INCLUDE A FACTOR OF "PI" (3.14159) AS FOLLOWS:

```
FS = SIN(A2(I)*3.14159)
```

AND THE NUMBER OF RECORDS IN THE FILES "F14.DAT" AND "F15.DAT" DEFINED ON PAGES 63 AND 64 SHOULD BE 252 (NOT 336), THUS THE FILE DEFINITIONS SHOULD READ:

```
DEFINE FILE 1(252,2,U,NREC)
```

3) THE TRANSFER MACROFILES WHICH ACCESS THE IOCC ALLOW THE USER TO SPECIFY THE I/O OPCODE. THESE OPCODES ARE DEFINED IN SECTION 4.2.5.3.2 OF THE AD-10 REFERENCE MANUAL AND ARE REFERRED TO AS "I/O DEVICE CONTROL BITS".

4) THE DEFINITION OF THE TERM "FUNCTION DATA" ON PAGE 1 ALSO ATTEMPTS TO DESCRIBE HOW THE FUNCTION DATA ARRAYS MUST BE ORDERED IN SEQUENTIAL PDP-11 DATA FILES. THE FOLLOWING SHOULD FURTHER CLARIFY FUNCTION DATA ORDERING:

FUNCTION DATA MUST BE ORDERED INTO A LINEAR ARRAY IN THE PDP-11, PRIOR TO LOADING INTO THE AD-10. THE ORDER OF INDEXING IN THIS ARRAY IS DETERMINED BY THE ORDER OF LISTING OF THE VARIABLES IN THE FUNCTIONAL NOTATION. THUS FOR F1(X,Y,Z) THE DATA ARRAY WOULD BEGIN WITH: F1(X1,Y1,Z1), F1(X2,Y1,Z1), F1(X3,Y1,Z1),... AND AFTER INDEXING "X" FOR ALL BREAKPOINTS, "Y" IS INCREMENTED AND "X" IS AGAIN INDEXED THROUGH ALL BREAKPOINTS AS FOLLOWS: F1(X1,Y2,Z1), F1(X2,Y2,Z1), F1(X3,Y2,Z1), ... AFTER ALL VALUES OF "Y" HAVE BEEN INDEXED, "Z" IS THEN INCREMENTED AND THE PROCESS IS REPEATED. IT IS ASSUMED THAT IF THE SAME SET OF VARIABLES OCCUR IN MORE THAN ONE FUNCTION, THESE WILL BE LISTED IN THE SAME ORDER IN ALL FUNCTIONS, SUCH AS F1(X,Y,Z), F2(X,Y,Z), F3(X,Y,Z), ETC..., SO THAT THE SAME POINTER CALCULATION CAN BE USED FOR ALL SUCH FUNCTIONS.

5) THE MANUAL SHOULD CONTAIN A SECTION ON CALLING THE MACROFILES AS SUBROUTINES. IN GENERAL, ANY MACROFILE CAN BE CALLED AS A SUBROUTINE AS LONG AS ALL "IMMEDIATE DATA" AND "IMMEDIATE ADDRESSES" WHICH ARE BUILT INTO THE AD-10 INSTRUCTIONS DO NOT NEED TO CHANGE FROM ONE CALL TO THE NEXT. THE PROCEDURE IS AS FOLLOWS:

A) SWAP INPUT QUANTITIES INTO THE REGISTERS WHICH WERE ASSEMBLED AS THE INPUT REGISTERS FOR THE MACROFILE.

B) LOAD PROGRAM COUNTERS FOR ALL PROCESSORS USED IN THE MACROFILE TO THE STARTING LOCATIONS FOR THESE PROCESSORS IN THE MACROFILE SUBROUTINE.

```
        LPC   $ARP,SUBARP
        LPC   $DEP,SUBDEP
        LPC   $MAP,SUBMAP
```

C) SETUP THE RETURN ADDRESS FOR THE COP IN A GENERAL REGISTER AND JUMP TO THE SUBROUTINE:

```
        LFI   RETADR
        SGRF  RETREG
        JMP   SUBADR
RETADR  LPC   $ARP,NEXTA   ! RESTORE PC'S FOR
        LPC   $DEP,NEXTD   ! PROCESSORS USED IN
        LPC   $MAP,NEXTM   ! THE SUBROUTINE.
```

D) FOLLOW THE SUBROUTINE WITH A COP PROGRAM TO RETURN TO THE ADDRESS SETUP IN STEP C), I.E.

```
        .COP
SUBADR  .EQU *
        .ARP
SUBARP  .EQU *
        .DEP
SUBDEP  .EQU *
        .MAP
SUBMAP  .EQU *
        .INCLUDE  MACROFILE
        LGRF   RETREG ! LOAD RETURN ADDR TO MULTIBUS
        JPM           ! JUMP TO THE ADDR ON MULTIBUS
```

E) SWAP RESULTS FROM OUTPUT REGISTERS WHICH WERE ASSEMBLED AS THE OUTPUT REGISTERS FOR THE MACROFILE TO THE DESIRED LOCATIONS.

PREFACE

A MACROFILE IS AN AD-10 ASSEMBLY LANGUAGE APPLICATION ROUTINE IN SOURCE FORM WHICH CAN BE INCLUDED IN A USER APPLICATION PROGRAM WITH USER-SPECIFIED INPUT/OUTPUT PARAMETERS OR ARGUMENTS. A MACROFILE IS SIMILAR TO A SUBROUTINE IN A HIGH LEVEL LANGUAGE, WITH THE EXCEPTION THAT EACH "CALL" TO A MACROFILE INCLUDES ANOTHER COPY OF THE MACROFILE CODE, WITH THE USER-SPECIFIED ARGUMENTS IN THE USER PROGRAM.

THE AD-10 MACROFILE LIBRARY CONTAINS ROUTINES WHICH SUPPORT ALL PHASES OF MULTIVARIABLE FUNCTION GENERATION APPLICATIONS, INCLUDING DATA INPUT AND OUTPUT, DATA TRANSFERS WITHIN THE AD-10, BINARY AND SHIFT SEARCH SCHEMES (I.E. TO DETERMINE THE LOCATION OF INPUT VARIABLES IN THE DOMAIN OF THE FUNCTION), FUNCTION DATA POINTER CALCULATIONS, AND LINEAR INTERPOLATION FOR 1,2,3,4, AND 5 VARIABLE FUNCTIONS. ALSO SEVERAL SUPPORT ROUTINES ARE INCLUDED TO PERFORM CALCULATIONS OF SIN'S AND COS'S, FORWARD AND INVERSE RESOLUTION, "SGN" FUNCTION, ETC... ROUTINES WHICH PERFORM GENERAL CALCULATIONS SUCH AS THESE ARE CONSTANTLY BEING ADDED TO THE MACROFILE LIBRARY AS THEY PROVE USEFUL IN USER APPLICATIONS. THE CONVENTIONS USED IN WRITING MACROFILES AND IN PASSING ARGUMENTS TO MACROFILES ARE VERY SIMPLE, THUS USERS CAN EASILY WRITE THEIR OWN SPECIAL PURPOSE MACROFILES TO AUGMENT THOSE PROVIDED IN THE LIBRARY.

TABLE OF CONTENTS

TERMS USED IN THIS MANUAL

MULTIVARIANT FUNCTION	A CONTINUOUS FUNCTION OF ONE OR MORE VARIABLES WHICH WILL BE DEFINED AT DISCRETE VALUES OF THE VARIABLE(S).
FUNCTION DATA	AN ARRAY OF DISCRETE VALUES OF A FUNCTION ORDERED SUCH THAT THE VALUES FOR THE FIRST VARIABLE VARY MOST RAPIDLY AND THE VALUES FOR THE LAST VARIABLE VARY MOST SLOWLY.
FUNCTION DATA PAIR	TWO ADJACENT FUNCTION DATA VALUES BETWEEN WHICH A LINEAR INTERPOLATION WILL BE PERFORMED.
FUNCTION DATA POINTER *	A POINTER TO THE FIRST FUNCTION DATA PAIR TO BE USED IN THE INTERPOLATION ALGORITHM. THIS POINTER IS A FUNCTION OF THE BREAKPOINT INDICES FOR ALL VARIABLES IN THE VARIABLE SET.
BREAKPOINTS	THE DISCRETE VALUES OF AN INPUT VARIABLE AT WHICH A FUNCTION OF THAT VARIABLE IS DEFINED.
BREAKPOINT TABLE	AN ARRAY OF BREAKPOINTS IN ASCENDING ORDER FOR A PARTICULAR VARIABLE.
BREAKPOINT INTERVAL	THE INTERVAL BETWEEN THE TWO ADJACENT BREAKPOINTS BETWEEN WHICH THE VALUE OF THE INPUT VARIABLE FALLS.
BREAKPOINT INDEX *	THE POINTER TO THE FIRST BREAKPOINT TABLE ENTRY WHICH IS LESS THAN OR EQUAL TO THE CURRENT VALUE OF THE INPUT VARIABLE (I.E. THE LOWER BREAKPOINT OF THE BREAKPOINT INTERVAL).
DELTA	THE FRACTIONAL VALUE (BETWEEN 0 AND 1.0) CORRESPONDING TO THE POSITION OF A VARIABLE IN A BREAKPOINT INTERVAL. $$DELTA = \frac{V - B(I)}{B(I+1) - B(I)}$$
VARIABLE SET	A COLLECTION OF VARIABLES WHICH WILL BE USED AS THE ARGUMENT LIST FOR ONE OR MORE FUNCTIONS.

* NOTE: THESE POINTERS AND/OR INDEX VALUES BEGIN WITH THE VALUE 0.

MACROFILE CALL SEQUENCE AND CONVENTIONS

A MACROFILE IS AN AD-10 ASSEMBLY LANGUAGE APPLICATION ROUTINE IN SOURCE FORM WHICH CAN BE INCLUDED IN A USER APPLICATION PROGRAM WITH USER-SPECIFIED INPUT/OUTPUT PARAMETERS OR ARGUMENTS. ARGUMENTS ARE PASSED TO AND FROM MACROFILES USING SYMBOLS WHICH BEGIN WITH A "#". THESE "#" SYMBOLS STAND FOR EITHER A TEMPORARY REGISTER NUMBER, A CONSTANT, OR A MEMORY ADDRESS. THE ONLY DIFFERENCE BETWEEN A "#" SYMBOL AND AN ORDINARY SYMBOL IS THAT THE AD-10 ASSEMBLER ALLOWS A SYMBOL WHICH BEGINS WITH "#" TO BE DEFINED MORE THAN ONCE. THIS ALLOWS THE USER TO CALL THE SAME MACROFILE MORE THAN ONCE AND TO CHANGE THE ARGUMENTS AS NECESSARY. IF A MACROFILE ARGUMENT DOES NOT CHANGE FROM ONE CALL TO THE NEXT IT IS NOT NECESSARY TO DEFINE THAT ARGUMENT MORE THAN ONCE. HOWEVER, BE AWARE THAT IN SOME CASES THE SAME SYMBOLIC ARGUMENT IS USED BY SEVERAL MACROFILES.

SINCE THE AD-10 PROCESSORS AND DATA MEMORY BOTH REQUIRE PIPELINED PROGRAMMING TO REALIZE FULL SPEED AND EFFICIENT OPERATION, MOST MACROFILES PERFORM THE SAME TASK FOR SEVERAL SETS OF INPUTS. BECAUSE OF THIS, THE NAMING CONVENTION FOR MACROFILE ARGUMENTS IS TO END EACH ARGUMENT WITH A NUMBER TO IDENTIFY EACH ARGUMENT SET.

FOR EXAMPLE, SUPPOSE "#IN" IS THE INPUT AND "#OUT" IS THE OUTPUT OF A MACROFILE CALLED "COMPUTE", AND THE CALCULATIONS ARE PERFORMED FOR 3 SETS OF ARGUMENTS. THE FOLLOWING STATEMENTS WOULD BE REQUIRED TO DEFINE THE 3 SETS OF ARGUMENTS AND TO "CALL" THE MACROFILE:

```
#IN0    .EQU    <VALUE1>
#IN1    .EQU    <VALUE2>
#IN2    .EQU    <VALUE3>
#OUT0   .EQU    <VALUE4>
#OUT1   .EQU    <VALUE5>
#OUT2   .EQU    <VALUE6>
        .INCLUDE  COMPUTE   ! "CALLS" MACROFILE
```

THE AD-10 ASSEMBLER HAS A ".DEFINE" DIRECTIVE WHICH DOES THE EQUIVALENT OF MULTIPLE ".EQU" SYMBOL DEFINITIONS AND ALLOWS THE ARGUMENTS TO MACROFILES TO BE DEFINED MORE SIMPLY AS FOLLOWS:

```
#IN     .DEFINE <VALUE1>,<VALUE2>,<VALUE3>
#OUT    .DEFINE <VALUE4>,<VALUE5>,<VALUE6>
        .INCLUDE  COMPUTE  ! "CALLS" MACROFILE
```

SOME MACROFILES MUST DEFINE THEIR OWN INTERNAL SYMBOLS FOR ADDRESS CALCULATIONS OR FOR TEMPORARY STORAGE LOCATIONS. WHENEVER A MACROFILE DOES DEFINE A SYMBOL INTERNALLY, THE SYMBOL ALWAYS BEGINS WITH "##", THUS INTERNAL SYMBOLS SHOULD NEVER CONFLICT WITH USER SYMBOLS OR OTHER MACROFILE ARGUMENTS.

THE GENERAL FORMAT FOR MACROFILES IN AD-10 ASSEMBLY LANGUAGE NOTATION IS AS FOLLOWS:

```
 .PROFF     ! PRECEDE WITH A ".PRON" TO PRINT CODE
 .PROFF     ! PRECEDE WITH ANOTHER ".PRON" TO PRINT DESCRIPTION
 .PAGE      ! STARTING AT THE TOP OF THE NEXT PAGE
!
!  DESCRIPTION OF MACROFILE
!
 .PRON
 .COP
!**********************************
!
!  COP CONTROL PROGRAM
!
!**********************************
!
!  PROGRAMS FOR ANY OTHER PROCESSORS
!
!**********************************
 .COP
 .PRON      ! END OF MACROFILE*******
```

THE USER MUST SPECIFY ONE .PRON IN THE PROGRAM PRIOR TO INCLUDING A MACROFILE FOR THE MACROFILE CODE TO BE PRINTED IN THE PROGRAM LISTING. A DETAILED DESCRIPTION OF THE MACROFILE AND ITS ARGUMENTS CAN ALSO BE PRINTED BY USING A SECOND ".PRON", HOWEVER THIS IS NOT RECOMMENDED SINCE SOME OF THE DESCRIPTIONS ARE QUITE LONG AND THE SAME INFORMATION IS CONTAINED IN THIS MANUAL. NOTICE THAT MACROFILES END WITH A ".COP" DIRECTIVE, THUS A MACROFILE CAN BE FOLLOWED WITH COP CODE WITHOUT ISSUING ANOTHER ".COP" DIRECTIVE.

THERE ARE A FEW RULES WHICH MUST BE FOLLOWED WHEN INCLUDING MACROFILES TO AVOID CONFLICTS AND ERRONEOUS RESULTS AT RUNTIME.

1) UPON ENTRY TO A MACROFILE ALL AD-10 PROCESSORS MUST BE STOPPED AND MUST NOT BE IN THE MIDDLE OF A "PAUSE" INSTRUCTION. (NOTE: PROCESSOR(S) NOT USED BY A MACROFILE COULD POSSIBLY BE PROGRAMMED TO PERFORM SOME INTERNAL OPERATIONS IN PARALLEL WITH A MACROFILE, HOWEVER THIS IS NOT RECOMMENDED).

2) ALSO UPON ENTRY A READ FROM MEMORY AND/OR THE IOCC MUST NOT BE IN PROGRESS AS THE DATA MIGHT CONFLICT WITH DATA THE MACROFILE PUTS ON THE MULTIBUS.

3) THE USER SHOULD TAKE CARE PRIOR AND/OR FOLLOWING ANY MACROFILE WHICH ACCESSES DATA MEMORY TO AVOID A MEMORY PAGE CONFLICT. IF IN DOUBT, A "PAUSE 2" INSTRUCTION PRIOR TO AND/OR FOLLOWING SUCH A MACROFILE WILL AVOID ANY POSSIBILITY OF A MEMORY PAGE CONFLICT (FOR THE WORST CASE SITUATION).

4) ALL UNUSED MACROFILE ARGUMENTS MUST BE DEFINED SO AS NOT TO CONFLICT WITH THE USED ARGUMENTS. FOR EXAMPLE, IF A TRANSFER MACROFILE IS USED TO TRANSFER 6 VALUES TO MEMORY, WHEN IT HAS THE CAPABILITY TO TRANSFER 8 VALUES, THE 2 UNUSED MEMORY ADDRESSES MUST BE DEFINED SUCH THAT THEY DO NOT CAUSE A MEMORY ACCESS ERROR. THE INDIVIDUAL MACROFILE DESCRIPTIONS SUGGEST RECOMMENDED DEFINITIONS FOR UNUSED ARGUMENTS.

MVFG MACROFILES

THE MVFG MACROFILES ARE A SET OF ROUTINES WHICH SUPPORT ALL PHASES OF MULTIVARIANT FUNCTION GENERATION ON THE AD-10. THESE MACROFILES ALLOW FOR EITHER EQUALLY OR NON-EQUALLY SPACED BREAKPOINTS AND CAN INTERPOLATE FOR FUNCTIONS OF FROM 1 TO 5 VARIABLES. AN OUTLINE OF THE OVERALL FUNCTION GENERATION PROBLEM IS PROBABLY THE EASIEST WAY TO ILLUSTRATE HOW THE APPLICATION OF MULTIVARIANT FUNCTION GENERATION IS SUBDIVIDED AND ROUTINES ASSIGNED TO THE VARIOUS AREAS.

THE MVFG MACROFILES WERE WRITTEN WITH THE RUN TIME FUNCTION GENERATION PROBLEM DIVIDED INTO THE FOLLOWING THREE TASKS:

1) LOCATE EACH VARIABLE IN ITS BREAKPOINT TABLE (EITHER EQUALLY OR NON-EQUALLY SPACED BREAKPOINTS) AND COMPUTE THE CORRESPONDING "DELTA" QUANTITY.

2) COMPUTE THE FUNCTION DATA POINTER FOR EACH VARIABLE SET OF WHICH A FUNCTION IS TO BE GENERATED.

3) INTERPOLATE FOR THE VALUE OF EACH FUNCTION USING THE FUNCTION DATA PAIRS POINTED TO BY OFFSETS FROM THE FUNCTION DATA POINTER FROM 2) AND THE DELTA QUANTITIES FROM 1). ONLY STEP 3) NEED BE REPEATED TO GENERATE MULTIPLE FUNCTIONS OF THE SAME VARIABLE SET.

THE ABOVE STEPS ARE SUPPORTED BY THE THE FOLLOWING MACROFILES:

STEP	MVFG SUBROUTINE(S)
1)	BD.6 , SD.6
2)	PT2.3, PT3.3, PT4.3, PT5.3
3)	FI1.3, FI2.3, FI3.3, FI4.3, FI5.3

AS AN AID TO UNDERSTANDING HOW TO USE THESE ROUTINES, IT WILL BE HELPFUL TO FIRST DISCUSS THE SEARCH SCHEMES, DELTA CALCULATION, FUNCTION DATA INDEXING, AND INTERPOLATION ALGORITHM USED BY THE MVFG MACROFILES.

BREAKPOINT INDEX AND DELTA CALCULATIONS

THERE ARE TWO TECHNIQUES SUPPORTED BY THE MACROFILES TO DETERMINE THE BREAKPOINT INDEX AND DELTA VALUE ASSOCIATED WITH EACH INPUT VARIABLE: A BINARY SEARCH AND DELTA CALCULATION MACROFILE (BD.6) AND A SHIFT SEARCH AND DELTA CALCULATION MACROFILE (SD.6).

THE BINARY SEARCH TECHNIQUE REQUIRES THAT A TABLE OF BREAKPOINTS BE STORED IN AD-10 DATA MEMORY. THE BINARY SEARCH MACROFILE ALLOWS UP TO 33 BREAKPOINTS TO BE SPECIFIED (32 IN THE TABLE PLUS AN ASSUMED +1.0 UPPER BREAKPOINT). IF FEWER THAN 33 BREAKPOINTS ARE DESIRED, THE LOWER END OF THE BREAKPOINT TABLE IS SIMPLY PADDED WITH -1.0 ENTRIES. THE MAIN REASON FOR THE CHOICE OF THE BINARY SEARCH IS THAT IT IS EFFICIENT (ONLY 5 COMPARISONS ARE REQUIRED TO DETERMINE THE BREAKPOINT INDEX FOR 33 BREAKPOINTS) AND THE EXECUTION TIME OF THE MACROFILE IS FIXED, WHICH IS IMPORTANT IF A UNIFORM TIME FRAME IS TO BE MAINTAINED. ASSOCIATED WITH EACH BREAKPOINT TABLE THE USER MUST PRECOMPUTE TWO ARRAYS OF "S" AND "G" DATA VALUES WHICH ARE USED IN THE CALCULATION OF THE DELTA QUANTITY "DV":

$$DV = \frac{V-B(I)}{B(I+1)-B(I)}$$

THIS CALCULATION IS PERFORMED IN TWO STEPS AS FOLLOWS:

```
R  = (V-B(I))*S(I)
DV = 2*R*G(I)

WHERE  S(I) = INTEGER(.5/(B(I+1)-B(I))) + 1
       G(I) = .5/(S(I)*(B(I+1)-B(I)))
```

THE SHIFT SEARCH SCHEME ALLOWS (2**N)+1 BREAKPOINTS (N=2,15) WHICH ARE EQUALLY SPACED OVER THE RANGE FROM -1.0 TO +1.0 . THE BASIC TECHNIQUE USED BY THE MACROFILE IS TO EXTRACT THE BREAKPOINT INDEX "I" FROM THE HIGH ORDER BITS OF THE INPUT VARIABLE, USE "I" TO COMPUTE THE CORRESPONDING BREAKPOINT VALUE, AND THEN COMPUTE THE DELTA VALUE USING THE INPUT VARIABLE, THE BREAKPOINT VALUE, AND THE FIXED SPACING BETWEEN BREAKPOINTS. IN SUMMARY, THE SHIFT SEARCH AND DELTA CALCULATION ARE PERFORMED AS FOLLOWS:

```
LET THE NUMBER OF BREAKPOINTS = (2**N)+1

I    = (V-2**(-N)) * 2**(-16+N)   SCALED FRACTION CALCULATION

B(I) = I * 2**(16-N)              INTEGER CALCULATION

DV   = (V-B(I)) * 2**(N-1)        INTEGER CALCULATION
```

THE SHIFT SEARCH AND DELTA CALCULATION HAVE THE ADVANTAGE OF BEING ABOUT 3 TIMES FASTER THAN THE 33 BREAKPOINT BINARY SEARCH TECHNIQUE REGARDLESS OF HOW MANY BREAKPOINTS THE USER WISHES TO USE. THE EXECUTION TIME OF THE SHIFT SEARCH MACROFILE IS ALSO FIXED FOR ANY NUMBER OF BREAKPOINTS.

FUNCTION DATA INDEXING

THE FOLLOWING TABLE SHOWS HOW FUNCTION DATA POINTERS ARE COMPUTED BASED ON THE NUMBER OF VARIABLES IN THE VARIABLE SET, THE LENGTH OF THE BREAKPOINT TABLE FOR EACH VARIABLE, AND THE CURRENT BREAKPOINT INDICES FOR EACH VARIABLE. IT ALSO ILLUSTRATES HOW THE ADDRESSES OF FUNCTION DATA PAIRS ARE COMPUTED BASED ON OFFSETS FROM THE FUNCTION DATA POINTER.

M	NUMBER OF VARIABLES IN THE VARIABLE SET.
IK	BREAKPOINT INDEX FOR THE K'TH VARIABLE IN THE VARIABLE SET.
NIK	NUMBER OF BREAKPOINTS FOR THE K'TH VARIABLE IN THE VARIABLE SET.
ISETM	FUNCTION DATA POINTER FOR A SET OF "M" VARIABLES (COMPUTED BY THE "PT..." MACROFILES).
BASE	BASE ADDRESS OF THE FUNCTION DATA.
OPAIRI	OFFSET OF FUNCTION DATA PAIR "I" FROM THE POINTER "ISETM". THE FUNCTION DATA PAIRS ARE ADDRESSED FOR INTERPOLATION BY A MAP "RAID OPAIRI,I" INSTRUCTION; THE FIXED PART OF THE ADDRESS, "OPAIRI", IS BUILT INTO THE MAP INSTRUCTION AND THE VARIABLE PART OF THE ADDRESS, THE FUNCTION DATA POINTER "ISETM", IS COMPUTED AT RUN TIME AND IS STORED IN A MAP "I" INDEX REGISTER.

M	"FUNCTION DATA POINTER" AND "FUNCTION DATA PAIRS"
1	ISET0 = I0
	OPAIR0 = BASE
2	ISET1 = I0 + I1*NI0
	OPAIR0 = BASE
	OPAIR1 = BASE + NI0
3	ISET2 = I0 + I1*NI0 + I2*NI0*NI1
	OPAIR0 = BASE
	OPAIR1 = BASE + NI0
	OPAIR2 = BASE + NI0*NI1
	OPAIR3 = BASE + NI0*NI1 + NI0
4	ISET4 = I0 + I1*NI0 + I2*NI0*NI1 + I3*NI0*NI1*NI2
	OPAIR0 = BASE
	OPAIR1 = BASE + NI0
	OPAIR2 = BASE + NI0*NI1
	OPAIR3 = BASE + NI0*NI1 + NI0
	OPAIR4 = BASE + NI0*NI1*NI2
	OPAIR5 = BASE + NI0*NI1*NI2 + NI0
	OPAIR6 = BASE + NI0*NI1*NI2 + NI0*NI1
	OPAIR7 = BASE + NI0*NI1*NI2 + NI0*NI1 + NI0

THE USER SHOULD BE AWARE OF A FEW TRICKS WHICH ALLOW MORE FLEXIBLE USE OF THE "PT..." AND "FI..." MACROFILES. NOTICE THAT THE EQUATION FOR THE FUNCTION DATA POINTER HAS AN ADDED TERM EACH TIME THE NUMBER OF VARIABLES INCREASES BY ONE. BY SETTING THE APPROPRIATE "NI" VARIABLE(S) EQUAL TO 0, IT IS POSSIBLE FOR EXAMPLE TO USE THE PT4.3 MACROFILE TO GENERATE A FUNCTION DATA POINTER FOR A 3 OR 2 VARIABLE SET. THIS ALLOWS THE USER TO MAKE FULL USE OF THE CAPABILITY OF THE PT4.3 MACROFILE EVEN IF THE APPLICATION DOES NOT REQUIRE 3 POINTERS FOR SETS OF 4 VARIABLES.

THE SAME TECHNIQUE ALSO CAN BE USED WITH THE "FI..." INTERPOLATION MACROFILES. NOTICE THAT THE ADDRESSES OF THE FUNCTION DATA PAIRS ARE SUCH THAT IF THE NUMBER OF BREAKPOINTS (I.E. THE "NI" VARIABLE) FOR THE LAST VARIABLE WERE SET TO 0, THEN FOR EXAMPLE IN THE CASE OF M=4 THE LAST 4 PAIR ADDRESSES WOULD BE THE SAME AS FOR THE FIRST 4 PAIRS. THIS ALLOWS FOR EXAMPLE USING THE FI4.3 MACROFILE TO INTERPOLATE FOR A FUNCTION OF 3, 2 OR EVEN 1 VARIABLE BY SETTING THE APPROPRIATE "NI" ARGUMENT(S) EQUAL 0. IN THE CASE OF THE 4 VARIABLE INTERPOLATION IF NI2=0, THEN THE RESULTING FUNCTION VALUE WILL BE THE LINEAR INTERPOLATION FOR A 3 VARIABLE FUNCTION OF THE FIRST 3 VARIABLES. IF NI1 AND NI2 ARE BOTH SET TO 0, THEN THE RESULT WILL BE A 2 VARIABLE FUNCTION OF THE FIRST 2 VARIABLES. NOTE THAT THE CORRESPONDING DELTA VALUES FOR THE UNUSED VARIABLES WILL NOT AFFECT THE INTERPOLATION, AND THUS CAN HAVE ANY VALUE. THE REASON THIS TECHNIQUE WORKS, IN GENERAL TERMS, IS THAT THE FINAL INTERPOLATIONS ARE BETWEEN THE SAME TWO FUNCTION VALUES. REFER TO THE FOLLOWING SECTION DESCRIBING THE FUNCTION INTERPOLATION ALGORITHM FOR MORE DETAILS.

INTERPOLATION ALGORITHM

THE ALGORITHM USES A SCHEME WHICH REDUCES THE INTERPOLATION FOR FUNCTIONS OF ANY NUMBER OF VARIABLES TO A SEQUENCE OF CALCULATIONS OF THE FORM:

```
F0 + (F1 - F0)*DELTA
```

WHERE [F0,F1] IS A "FUNCTION DATA PAIR", AS REFERRED TO IN THE PREVIOUS DISCUSSION OF FUNCTION DATA INDEXING. TO DESCRIBE THE ALGORITHM, THE FOLLOWING TERMS WILL BE USED:

```
   X     REPRESENTS:  THE CURRENT VALUE OF THE
                      INDEPENDENT VARIABLE.

  X(I)   REPRESENTS:  THE I'TH BREAKPOINT VALUE FOR
                      FOR THE VARIABLE X.

F(000)   REPRESENTS:  F(X(I),Y(J),Z(K))
F(111)   REPRESENTS:  F(X(I+1),Y(J+1),Z(K+1))
F(101)   REPRESENTS:  F(X(I+1),Y(J),Z(K+1))

                          X-X(I)
  D(X)   REPRESENTS:  -------------
                       X(I+1)-X(I)
```

INTERPOLATION FOR A FUNCTION OF ONE VARIABLE IS GIVEN BY THE FOLLOWING:

```
F(X) = F(0) + (F(1) - F(0)) * D(X)
```

INTERPOLATION FOR A FUNCTION OF 2 VARIABLES CAN BE REPRESENTED BY USING AN INTERMEDIATE FUNCTION "H" AS FOLLOWS:

```
H(0) = F(00) + (F(10) - F(00)) * D(X)
H(1) = F(01) + (F(11) - F(01)) * D(X)

F(X,Y) = H(0) + (H(1) - H(0)) * D(Y)
```

THE GENERAL FORM OF THE INTERPOLATION FOR A 3 VARIABLE FUNCTION IS:

```
G(JK) = F(0JK) + (F(1JK) - F(0JK)) * D(X) ; J=0,1 ; K=0,1

 H(I) = G(0I) + (G(1I) - G(0I)) * D(Y)    ; I=0,1

F(X,Y,Z) = H(0) + (H(1) - H(0)) * D(Z)
```

OR IN MORE EXPANDED FORM:

```
G(00) = F(000) + (F(100) - F(000)) * D(X)
G(10) = F(010) + (F(110) - F(010)) * D(X)
G(01) = F(001) + (F(101) - F(001)) * D(X)
G(11) = F(011) + (F(111) - F(011)) * D(X)

 H(0) = G(00) + (G(10) - G(00)) * D(Y)
 H(1) = G(01) + (G(11) - G(01)) * D(Y)

F(X,Y,Z) = H(0) + (H(1) - H(0)) * D(Z)
```

THE EXTENSION OF THE INTERPOLATION TECHNIQUE ILLUSTRATED TO A FUNCTION OF AN ARBITRARY NUMBER OF VARIABLES SHOULD BE QUITE CLEAR. AMONG THE ADVANTAGES OF THIS INTERPOLATION ALGORITHM ARE:

1) THE BREAKPOINT INDICES AND DELTA QUANTITIES ARE COMPUTED JUST ONCE FOR EACH INPUT VARIABLE AND USED FOR ANY FUNCTION OF THAT VARIABLE.

2) THE FORMULA ALWAYS USES PAIRS OF FUNCTION VALUES WHICH ARE ADJACENT IN THE FUNCTION DATA ARRAY. NOTE: ADJACENT FUNCTION VALUES ARE STORED IN DIFFERENT PAGES OF AD-10 DATA MEMORY SO A PAIR CAN BE ACCESSED AT FULL MULTIBUS SPEED USING A MAP "RAID" INSTRUCTION.

3) COMPARED TO OTHER ALGORITHMS, THE OVERALL ARITHMETIC OPERATION COUNT IS SMALL AND THE MULTIPLICATION COUNT IS MINIMIZED FOR A WIDE VARIETY OF FUNCTION MIXES.

4) TEMPORARY STORAGE CAN BE EFFICIENTLY ORGANIZED.

5) THE ALGORITHM IS IDEALLY SUITED TO GENERALIZATION FOR THE GENERATION OF FUNCTIONS OF AN ARBITRARY NUMBER OF VARIABLES.

TRANSFER MACROFILES

THE TRANSFER MACROFILES TRANSFER DATA BETWEEN PROCESSORS AND DATA MEMORY, BETWEEN THE COP AND THE OTHER PROCESSORS, AND FROM THE EXTERNAL IOCC TO THE AD-10. THERE IS ALSO A GROUP OF "LOAD" MACROFILES WHICH TRANSFER IMMEDIATE DATA (I.E. DATA IMBEDDED IN A PROGRAM MEMORY INSTRUCTION) TO THE VARIOUS PROCESSORS AND MEMORY. THE NAMING CONVENTIONS FOR THESE MACROFILES ARE AS FOLLOWS:

TR<SOURCE><DESTINATION>.8
LOAD<DESTINATION>.8

<SOURCE> OR <DESTINATION>	MEANING
A	ARP TEMPRORARY REGISTER
C	COP GENERAL REGISTER
X	DEP "X" REGISTER
I	MAP/DEP "I" REGISTER
E	EXTERNAL IOCC
M	DATA MEMORY

THUS, MACROFILE "TRMA.8" TRANSFERS 8 DATA VALUES FROM MEMORY TO ARP TEMPORARY REGISTERS, AND MACROFILE "LOADI.8" TRANSFERS 8 IMMEDIATE DATA VALUES TO MAP/DEP "I" REGISTERS.

GENERAL APPLICATIONS MACROFILES

THE AD-10 IS NOT A GENERAL PURPOSE DIGITAL COMPUTER, HOWEVER THERE ARE A NUMBER A GENERAL CALCULATIONS INVOLVED IN SIMULATION APPLICATIONS BESIDES MULTIVARIANT FUNCTION GENERATION, WHICH THE AD-10 CAN PERFORM VERY EFFECTIVELY. IN GENERAL, THE AD-10 CAN PERFORM MOST OF THE ALGEBRAIC CALCULATIONS WHICH IN THE PAST WERE ASSIGNED TO AN ANALOG COMPUTER IN A HYBRID SIMULATION, ASSUMING THE CALCULATIONS ARE PROPERLY SCALED (I.E. NORMALIZED AS THEY WOULD ALSO NEED TO BE FOR AN ANALOG COMPUTER IMPLEMENTATION). THE MACROFILE LIBRARY CONTAINS SEVERAL MACROFILES WHICH PERFORM GENERAL CALCULATIONS, SUCH AS: COMPUTING THE "SGN" FUNCTION, PERFORMING VECTOR COORDINATE TRANSFORMATIONS, CONVERTING FROM RECTANGULAR TO THE POLAR COORDINATE SYSTEM, ETC... AD-10 USERS MAY FIND IN THEIR OWN APPLICATIONS BLOCKS OF GENERAL CALCULATIONS WHICH CAN BE IMPLEMENTED EFFECTIVELY AS MACROFILES. THIS AREA OF THE MACROFILE LIBRARY IS CONSTANTLY GROWING AS EACH NEW AD-10 APPLICATION PROBLEM GENERATES NEW IDEAS FOR GENERAL CALCULATIONS WHICH CAN BE PERFORMED ON THE AD-10 AND WHICH ARE SUITABLE TO IMPLEMENT IN MODULAR ROUTINES.

DETAILED DESCRIPTIONS OF MACROFILES

THE FOLLOWING SECTION OF THIS MANUAL CONTAINS A DETAILED DESCRIPTION OF EACH MACROFILE IN THE AD-10 MACROFILE LIBRARY. THE FIRST PAGE OF EACH DESCRIPTION BRIEFLY EXPLAINS WHAT THE MACROFILE COMPUTES, DEFINES THE REQUIRED ARGUMENTS, INDICATES THE INSTRUCTION COUNTS FOR THE VARIOUS AD-10 PROCESSORS AND THE EXECUTION TIME, AND IN SOME CASES WARNS THE USER OF POSSIBLE ERROR CONDITIONS. IF FURTHER INFORMATION IS NECESSARY TO USE THE MACROFILE OR TO EXPLAIN MORE FULLY WHAT CALCULATIONS ARE PERFORMED, THEN THE DESCRIPTION IS CONTINUED ON SUCCESSIVE PAGES. MOST OF THE INFORMATION IN THIS SECTION OF THE MANUAL WAS EXTRACTED FROM THE COMMENT HEADER AT THE BEGINNING OF THE SOURCE CODE OF EACH MACROFILE.

BD.6 MACROFILE

DESCRIPTION: BINARY SEARCH AND DELTA COMPUTATION

THIS MACROFILE ACCEPTS 6 INPUT VARIABLES IN DEP "X" REGISTERS AND COMPUTES THE CORRESPONDING BREAKPOINT TABLE INDICES, WHICH ARE RETURNED IN THE MAP/DEP "I" REGISTERS WITH THE SAME NUMBERS AS THE INPUT VARIABLE "X" REGISTERS. IT ALSO COMPUTES THE DELTA VALUE ASSOCIATED WITH EACH INPUT VARIABLE AND RETURNS THE RESULT IN THE SPECIFIED ARP "T" REGISTERS. UP TO 33 BREAKPOINTS CAN BE SPECIFIED FOR EACH INPUT VARIABLE (+1 CAN SERVE AS THE UNDERSTOOD 33'RD BREAKPOINT). THE CALCULATIONS PERFORMED BY THIS MACROFILE ARE AS FOLOWS:

1) BINARY SEARCH FOR I'S SUCH THAT:
   BJ(I) <= VJ < BJ(I+1) ; J=0,5

2) DELTA COMPUTATION USING I'S FROM 1) FOR J=0,5 :
   RJ = (VI - BJ(I))*SJ(I)
   #DVJ = 2*RJ*GJ(I) = [VI-BJ(I)] / [BJ(I+1)-BJ(I)]

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
MAP,DEP	#V0,#V1,#V2,#V3,#V4,#V5	INPUT VARIABLES AND OUTPUT INDICES (REG. NUMBERS)
COP	#NV0,#NV1,#NV2,#NV3,#NV4,#NV5	BREAKPOINT TABLE LENGTHS (NOT COUNTING ASSUMED +1)
ARP	#DV0,#DV1,#DV2,#DV3,#DV4,#DV5	DELTA VALUES (REG. NUMBERS)
DAT	#ORG	ORIGIN OF BREAKPOINT DATA BLOCK
ARP	T0,T1,T2	SCRATCH REGISTERS

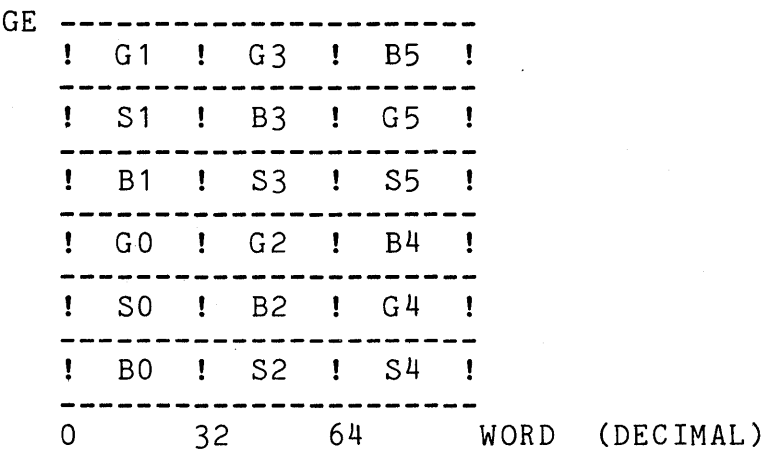
	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	13	22	43	49

EXECUTION TIME: 6.0 MICRO-SEC.

POSSIBLE ERRORS: -AN ADDRESS ERROR COULD OCCUR DUE TO A MEMORY ACCESS DURING THE INSTRUCTION PRIOR TO THIS MACROFILE.

REQUIRED DATA:

THE "B" BREAKPOINT DATA ARRAYS AND THE ASSOCIATED "S" AND "G" DATA VALUES MUST BE LOCATED IN DATA MEMORY AS SHOWN IN THE FOLLOWING SPECIALLY FORMATTED BLOCK WHICH IS 96 WORDS WIDE BY 6 PAGES HIGH. THE BASE ADDRESS OF THIS DATA BLOCK (#ORG) IS ONE OF THE USER SPECIFIED INPUTS TO THIS MACROFILE.



"B" BREAKPOINT DATA:

BJ(I) ; I=0,N SHOULD BE RIGHT JUSTIFIED WITH LOW ORDER UNUSED BREAKPOINTS SET TO -1. THE BREAKPOINT TABLE LENGTH IS N+1 AND SHOULD NOT COUNT THE ASSUMED UPPER +1 BREAKPOINT.

"S" AND "G" DATA VALUES:

SJ(I) = INTEGER(.5/(BJ(I+1)-BJ(I)) + 1 ; J=0,5

GJ(I) = .5/(SJ(I)*(BJ(I+1)-BJ(I)) ; J=0,5

(NOTE: IT IS NOT NECESSARY TO CALCULATE AND STORE AN "S" OR "G" VALUE FOR THE HIGHEST BREAKPOINT AND "S" AND "G" VALUES FOR UNUSED BREAKPOINTS SHOULD BE SET TO 0)

SYMBOLS DEFINED AND USED IN MACRO FILE:

USED IN CODE FOR	SYMBOLS	MEANING
MAP	##B0, ##B1, ##B2, ##B3, ##B4, ##B5	BREAKPOINT TABLES
MAP	##S0, ##S1, ##S2, ##S3, ##S4, ##S5 ##G0, ##G1, ##G2, ##G3, ##G4, ##G5	DELTA CALCULATION SCALING CONSTANTS
MAP	##PAG,##WRD	"PAGE" AND "WORD" CORRESPONDING TO #ORG

SPECIAL PROGRAMMING CONSIDERATIONS:

IF THE USER DOES NOT NEED TO USE ALL 6 INPUTS FOR THIS MACROFILE, THERE ARE SOME RECOMMENDED ARGUMENT AND DATA DEFINITIONS:

1) THE "S" AND "G" TABLES FOR THE UNUSED INPUTS TO THIS MACROFILE SHOULD BE LOADED WITH ALL 0'S TO AVOID THE POSSIBILITY OF AN ARITHMETIC OVERFLOW IN THE ARP DURING THE DELTA CALCULATION.

2) THE "B" ARRAYS FOR UNUSED INPUTS SHOULD BE LOADED WITH ALL -1.0'S (SCALED FRACTIONS) AND THE CORRESPONDING "#NV" INPUT SET TO 1 (INTEGER). THIS WILL RESULT IN A BREAKPOINT INDEX VALUE OF 0 FOLLOWING THE BINARY SEARCH.

   AN ALTERNATIVE IS TO SIMPLY SET THE CORRESPONDING "#NV" INPUTS TO 32 (WITHOUT LOADING ANYTHING IN THE UNUSED "B" ARRAYS). THIS WILL RESULT IN A BREAKPOINT INDEX REGISTER VALUE IN THE RANGE 0 TO 31.

THE REASON FOR THESE RECOMMENDATIONS IS TO AVOID ADDRESS CONFLICTS WHICH COULD OCCUR BETWEEN USED AND UNUSED PARTS OF THE MACROFILE.

SD.6 MACROFILE

DESCRIPTION: SHIFT SEARCH AND DELTA COMPUTATION FOR:

THIS MACROFILE ACCEPTS 6 INPUT VARIABLES IN DEP "X" REGISTERS AND COMPUTES THE CORRESPONDING BREAKPOINT INDICES, WHICH ARE RETURNED IN THE MAP/DEP "I" REGISTERS WITH THE SAME NUMBERS AS THE INPUT "X" REGISTERS. THIS ROUTINE ALLOWS (2**N)+1 EQUALLY SPACED BREAKPOINTS (FOR N=2,15) OVER THE RANGE -1.0 TO +1.0. IT ALSO COMPUTES THE DELTA VALUE ASSOCIATED WITH EACH INPUT VARIABLE AND RETURNS THE RESULT IN THE SPECIFIED ARP "T" REGISTERS.

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
DEP	#V0,#V1,#V2,#V3,#V4,#V5	INPUT VARIABLES AND OUTPUT INDICES (REG. NUMBERS)
COP	#NBPS	NUMBER OF BREAKPOINTS (MUST SATISIFY: #NBPS = (2**N)+1 FOR 2 <= N <= 15)
ARP	#DV0,#DV1,#DV2,#DV3,#DV4,#DV5	DELTA VALUES (REG. NUMBER)
ARP	T0,T1,T2	

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	6	22	13	0

EXECUTION TIME: 2.3 MICRO-SEC.

SPECIAL PROGRAMMING CONSIDERATIONS:

IF THE USER WISHES TO USE THIS MACROFILE, BUT THE INPUT VARIABLES ARE NOT DEFINED OVER THE ENTIRE RANGE -1.0 TO +1.0, THERE ARE SEVERAL POSSIBLE WAYS THAT THIS MACROFILE CAN STILL BE USED:

1) SUCH VARIABLES CAN BE TRANSLATED IN THE NEGATIVE DIRECTION SO THAT THE SMALLEST VALUE THE VARIABLES TAKE ON ARE TRANSLATED TO -1.0 . THE OUTPUT BREAKPOINT INDICES FROM THE MACROFILE WILL HAVE A VALUE OF 0 WHEN THE CORRESPONDING INPUT VARIABLES TAKE ON THEIR SMALLEST VALUE.

2) THE RESULTING INDICES FROM THIS MACROFILE CAN BE TRANSLATED SUCH THAT THE INDICES CORRESPONDING TO THE SMALLEST VALUE OF THE INPUT VARIABLE ARE EQUAL TO 0. THIS CAN BE DONE IN THE DEP PROCESSOR USING THE "CMM" INSTRUCTION, I.E.

```
.DEP
LFI  S'-1.0'     !-1.0 WILL FORCE CMM TO ADD K TO I REG.
CMM  IREG,-64    !ADD -64 TO THE CONTENTS OF IREG
```

3) THE BASE ADDRESS(ES) OF THE FUNCTION(S) ASSOCIATED WITH THE INPUT VARIABLE CAN BE "FUDGED" (I.E. TRANSLATED IN THE NEGATIVE DIRECTION) TO ACCOUNT FOR THE INDEX VALUE NOT STARTING AT 0. THIS TECHNIQUE WILL ONLY WORK IF THE RESULTING TRANSLATED BASE ADDRESS IS A POSITIVE NUMBER. REFER TO THE SECTION OF THIS MANUAL ON "FUNCTION DATA INDEXING" FOR A DETAILED DESCRIPTION OF THE INDEX CALCULATIONS.

4) THE USER CAN STORE FUNCTION VALUES FOR THE ENTIRE -1.0 TO +1.0 RANGE OF THE INPUT VARIABLE, EVEN THOUGH THE INPUT VARIABLE WILL NOT TRAVERSE THIS ENTIRE RANGE. THIS IS THE EASIEST SOLUTION, ASSUMING MEMORY REQUIREMENTS ARE NOT TIGHT.

CALCULATIONS PERFORMED:
---

1) I = (V - 2**(-N))*((2**(N-16)) + 2**(N-16)

THIS CALCULATION, DONE IN THE FRACTIONAL ARITHMETIC MODE, USES THE ROUNDOFF CHARACTERISTICS OF THE ARP AND DETERMINES THE BREAKPOINT INDEX "I" FROM THE HIGH ORDER BITS OF THE INPUT VARIABLE "V". THE CALCULATION RESULTS IN:

"I"	FOR "V" IN THE RANGE:				
0	-1.0	<=	V	<	-1.0+2.0/(2**N)
1	-1.0+2.0/(2**N)	<=	V	<	-1.0+4.0/(2**N)
2	-1.0+4.0/(2**N)	<=	V	<	-1.0+6.0/(2**N)
3	-1.0+6.0/(2**N)	<=	V	<	-1.0+8.0/(2**N)
.	.		.		.
.	.		.		.
.	.		.		.
2**N	1.0-2.0/(2**N)	<=	V	<	1.0

(NOTE: I IS COMPUTED AS A SCALED FRACTION BUT IS VIEWED AS AN INTEGER)

2) B(I) = (I - 2**(N-16)) * 2**(16-N)

THIS CALCULATION, WHICH IS DONE IN THE INTEGER ARITHMETIC MODE, USES THE INDEX VALUE "I" TO COMPUTE THE CORRESPONDING BREAKPOINT VALUE B(I). IN THIS CALCULATION "I" IS VIEWED AS A SCALED FRACTION. NOTE, THE INTEGER VALUE 2**(16-N) IS EQUIVALENT TO THE SCALED FRACTION 2**(1-N).

3) DV = ((V - B(I)) * 2**(N-1)

THIS CALCULATION, WHICH IS DONE IN THE INTEGER ARITHMETIC MODE, USES THE INPUT VARIABLE "V" AND THE BREAKPOINT VALUE "B(I)" TO COMPUTE THE DELTA VALUE "DV". NOTE THAT:

B(I) <= V AND 0 <= DV < 1.0

AND ALSO NOTE THAT THE INTEGER VALUE 2**(N-1) IS EQUIVALENT TO THE SCALED FRACTION 2**(N-16).

PT2.3 MACROFILE

DESCRIPTION: COMPUTE FUNCTION DATA POINTERS FOR 2 VARIABLE FUNCTIONS.

POINTER CALCULATIONS:

```
#IS0 = #I0 + #NI0*#J0
#IS1 = #I1 + #NI1*#J1
#IS2 = #I2 + #NI2*#J2
```

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
DEP	#I0,#J0,#I1,#J1,#I2,#J2	BREAKPOINT TABLE INDICES (REG. NUMBERS)
DEP	#NI0,#NI1,#NI2	BREAKPOINT TABLE LENGTHS
DEP	#IS0,#IS1,#IS2	FUNCTION DATA INDICES FOR EACH VARIABLE SET (REG. NUMBERS)
ARP	T0,T1	SCRATCH REGISTERS

	AD-10 PROCESSOR COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	4	8	8	0

EXECUTION TIME: 1.1 MICRO-SEC.

PT3.3 MACROFILE

DESCRIPTION: COMPUTE FUNCTION DATA POINTERS FOR 3 VARIABLE FUNCTIONS.

POINTER CALCULATIONS:

```
#IS0 = #I0 + #NI0*#J0 + #NI0*#NJ0*#K0
#IS1 = #I1 + #NI1*#J1 + #NI1*#NJ1*#K1
#IS2 = #I2 + #NI2*#J2 + #NI2*#NJ2*#K2
```

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
DEP	#I0,#J0,#K0,#I1,#J1,#K1,#I2,#J2,#K2	BREAKPOINT TABLE INDICES (REG. NUMBERS)
DEP	#NI0,#NJ0,#NI1,#NJ1,#NI2,#NJ2	BREAKPOINT TABLE LENGTHS
DEP	#IS0,#IS1,#IS2	FUNCTION DATA INDICES FOR EACH VARIABLE SET (REG. NUMBERS)
ARP	T0,T1	SCRATCH REGISTERS

INSTRUCTION COUNTS:

AD-10 PROCESSOR COP	ARP	DEP	MAP
10	11	11	0

EXECUTION TIME: 1.4 MICRO-SEC.

PT4.3 MACROFILE

DESCRIPTION: COMPUTE FUNCTION DATA POINTERS FOR 4 VARIABLE FUNCTIONS.

POINTER CALCULATIONS:

```
#IS0 = #I0 + #NI0*#J0 + #NI0*#NJ0*#K0 + #NI0*#NJ0*#NK0*#L0
#IS1 = #I1 + #NI1*#J1 + #NI1*#NJ1*#K1 + #NI1*#NJ1*#NK1*#L1
#IS2 = #I2 + #NI2*#J2 + #NI2*#NJ2*#K2 + #NI2*#NJ2*#NK2*#L2
```

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
DEP	#I0,#J0,#K0,#L0 #I1,#J1,#J1,#L1 #I2,#J2,#K2,#L2	BREAKPOINT TABLE INDICES (REG. NUMBERS)
DEP	#NI0,#NJ0,#NK0 #NI1,#NJ1,#NK1 #NI2,#NJ2,#NK2	BREAKPOINT TABLE LENGTHS
DEP	#IS0,#IS1,#IS2	FUNCTION DATA INDICES FOR EACH VARIABLE SET (REG. NUMBERS)
ARP	T0,T1	SCRATCH REGISTERS

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	13	14	14	0

EXECUTION TIME: 1.7 MICRO-SEC.

PT5.3 MACROFILE

DESCRIPTION: COMPUTE FUNCTION DATA POINTERS FOR 5 VARIABLE FUNCTIONS.

POINTER CALCULATIONS:

```
#IS0 = #I0 + #NI0*#J0 + #NI0*#NJ0*#K0 + #NI0*#NJ0*#NK0*#L0 +
                                        #NI0*#NJ0*#NK0*#NL0*#M0
#IS1 = #I1 + #NI1*#J1 + #NI1*#NJ1*#K1 + #NI1*#NJ1*#NK1*#L1 +
                                        #NI1*#NJ1*#NK1*#NL1*#M1
#IS2 = #I2 + #NI2*#J2 + #NI2*#NJ2*#K2 + #NI2*#NJ2*#NK2*#L2 +
                                        #NI2*#NJ2*#NK2*#NL2*#M2
```

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
DEP	#I0,#J0,#K0,#L0,#M0 #I1,#J1,#J1,#L1,#M1 #I2,#J2,#K2,#L2,#M2	BREAKPOINT TABLE INDICES (REG. NUMBERS)
DEP	#NI0,#NJ0,#NK0,#NL0 #NI1,#NJ1,#NK1,#NL1 #NI2,#NJ2,#NK2,#NL2	BREAKPOINT TABLE LENGTHS
DEP	#IS0,#IS1,#IS2	FUNCTION DATA INDICES FOR EACH VARIABLE SET (REG. NUMBERS)
ARP	T0,T1	SCRATCH REGISTERS

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	16	17	17	0

EXECUTION TIME: 2.0 MICRO-SEC.

FI1.3 MACROFILE

DESCRIPTION: INTERPOLATION FOR 1 VARIABLE FUNCTIONS.

F0(X0) , F1(X1) , F2(X2)

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
ARP	#DX0,#DX1,#DX2	DELTA VALUES (REG. NUMBERS)
ARP	#F0,#F1,#F2	FUNCTION VALUES (REG. NUMBERS)
MAP	#AF0,#AF1,#AF2	FUNCTION DATA ADDRESS ORIGINS
MAP	#IF0,#IF1,#IF2	FUNCTION DATA INDICES (REG. NUMBERS)

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	3	6	0	3

EXECUTION TIME: 1.1 MICRO-SEC.

POSSIBLE ERRORS:
- -ADDRESS ERROR DUE TO ILLEGAL FUNCTION DATA ORIGINS AND/OR INDICES.
- -ADDRESS ERROR DUE TO A CONFLICTING MEMORY REFERENCE DURING ONE OF THE 3 INSTRUCTIONS PRIOR TO THIS MACRO FILE.

SUGGESTED DEFINITIONS OF UNUSED ARGUMENTS:

IF THE USER DOES NOT NEED TO USE ALL 3 ARGUMENT SETS, THE FOLLOWING GUIDELINES SHOULD BE FOLLOWED TO AVOID MEMORY PAGE ADDRESS CONFLICTS:

1) THE UNUSED "#AF" FUNCTION DATA ADDRESS ORIGINS SHOULD BE DEFINED TO ADDRESS WORD 0 OF A DIFFERENT PAGE FROM THOSE PAGES OCCUPIED BY THE FUNCTION DATA FOR THE USED INPUTS.

2) THE UNUSED "#IF" FUNCTION DATA INDEX REGISTERS SHOULD CONTAIN 0 (OR A KNOWN REASONABLE VALUE WHICH WHEN ADDED TO THE CORRESPONDING "#AF" ADDRESS AT RUN TIME DOES NOT CAUSE AN ADDRESS CONFLICT).

3) DEFINE BREAKPOINT TABLE LENGTHS FOR UNUSED INPUTS TO BE 0.

FI2.3 MACROFILE

DESCRIPTION: INTERPOLATION FOR 2 VARIABLE FUNCTIONS.

F0(X0,Y0) , F1(X1,Y1) , F2(X2,Y2)

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
ARP	#DX0,#DY0,#DX1,#DY1,#DX2,#DY2	DELTA VALUES (REG. NUMBERS)
ARP	#F0,#F1,#F2	FUNCTION VALUES (REG. NUMBERS)
MAP	#AF0,#AF1,#AF2	FUNCTION DATA ADDRESS ORIGINS
MAP	#NX0,#NX1,#NX2	BREAKPOINT TABLE LENGTHS
MAP	#IF0,#IF1,#IF2	FUNCTION DATA INDICES (REG. NUMBERS)
ARP	T0 - T2	SCRATCH REGISTERS

INSTRUCTION COUNTS:

AD-10 PROCESSOR COP	ARP	DEP	MAP
7	16	0	6

EXECUTION TIME: 2.1 MICRO-SEC.

POSSIBLE ERRORS:
- -ADDRESS ERROR DUE TO ILLEGAL FUNCTION DATA ORIGINS AND/OR INDICES.
- -ADDRESS ERROR DUE TO A CONFLICTING MEMORY REFERENCE DURING ONE OF THE 3 INSTRUCTIONS PRIOR TO THIS MACRO FILE.

SUGGESTED DEFINITIONS OF UNUSED ARGUMENTS:

1) THE UNUSED "#AF" FUNCTION DATA ADDRESS ORIGINS SHOULD BE DEFINED TO ADDRESS WORD 0 OF A DIFFERENT PAGE FROM THOSE PAGES OCCUPIED BY THE FUNCTION DATA FOR THE USED INPUTS.

2) THE UNUSED "#IF" FUNCTION DATA INDEX REGISTERS SHOULD CONTAIN 0 (OR A KNOWN REASONABLE VALUE WHICH WHEN ADDED TO THE CORRESPONDING "#AF" ADDRESS AT RUN TIME DOES NOT CAUSE AN ADDRESS CONFLICT).

3) DEFINE BREAKPOINT TABLE LENGTHS FOR UNUSED INPUTS TO BE 0.

FI3.3 MACROFILE

DESCRIPTION: INTERPOLATION FOR 3 VARIABLE FUNCTIONS.

F0(X0,Y0,Z0) , F1(X1,Y1,Z1) , F2(X2,Y2,Z2)

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
ARP	#DX0,#DY0,#DZ0,#DX1,#DY1,#DZ1, #DX2,#DY2,#DZ2	DELTA VALUES (REG. NUMBERS)
ARP	#F0,#F1,#F2	FUNCTION VALUES (REG. NUMBERS)
MAP	#AF0,#AF1,#AF2	FUNCTION DATA ADDRESS ORIGINS
MAP	#NX0,#NY0,#NX1,#NY1,#NX2,#NY2	BREAKPOINT TABLE LENGTHS
MAP	#IF0,#IF1,#IF2	FUNCTION DATA INDICES (REG. NUMBERS)
ARP	T0 - T5	SCRATCH REGISTERS

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	7	31	0	12

EXECUTION TIME: 3.6 MICRO-SEC.

POSSIBLE ERRORS:
- -ADDRESS ERROR DUE TO ILLEGAL FUNCTION DATA ORIGINS AND/OR INDICES.
- -ADDRESS ERROR DUE TO A CONFLICTING MEMORY REFERENCE DURING ONE OF THE 3 INSTRUCTIONS PRIOR TO THIS MACRO FILE.

SUGGESTED DEFINITIONS OF UNUSED ARGUMENTS:

1) THE UNUSED "#AF" FUNCTION DATA ADDRESS ORIGINS SHOULD BE DEFINED TO ADDRESS WORD 0 OF A DIFFERENT PAGE FROM THOSE PAGES OCCUPIED BY THE FUNCTION DATA FOR THE USED INPUTS.

2) THE UNUSED "#IF" FUNCTION DATA INDEX REGISTERS SHOULD CONTAIN 0 (OR A KNOWN REASONABLE VALUE WHICH WHEN ADDED TO THE CORRESPONDING "#AF" ADDRESS AT RUN TIME DOES NOT CAUSE AN ADDRESS CONFLICT).

3) DEFINE BREAKPOINT TABLE LENGTHS FOR UNUSED INPUTS TO BE 0.

FI4.3 MACROFILE

DESCRIPTION: INTERPOLATION FOR 4 VARIABLE FUNCTIONS.

F0(V0,W0,X0,Y0)
F1(V1,W1,X1,Y1)
F2(V2,W2,X2,Y2)

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
ARP	#DV0,#DW0,#DX0,#DY0 #DV1,#DW1,#DX1,#DY1 #DV2,#DW2,#DX2,#DY2	DELTA VALUES (REG. NUMBERS)
ARP	#F0,#F1,#F2	FUNCTION VALUES (REG. NUMBERS)
MAP	#AF0,#AF1,#AF2	FUNCTION DATA ADDRESS ORIGINS
MAP	#NV0,#NW0,#NX0 #NV1,#NW1,#NX1 #NV2,#NW2,#NX2	BREAKPOINT TABLE LENGTHS
MAP	#IF0,#IF1,#IF2	FUNCTION DATA INDICES (REG. NUMBERS)
ARP	T0 - T8	SCRATCH REGISTERS

AD-10 PROCESSOR

	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	5	58	0	24

EXECUTION TIME: 6.3 MICRO-SEC.

POSSIBLE ERRORS:
- -ADDRESS ERROR DUE TO ILLEGAL FUNCTION DATA ORIGINS AND/OR INDICES.
- -ADDRESS ERROR DUE TO A CONFLICTING MEMORY REFERENCE DURING ONE OF THE 3 INSTRUCTIONS PRIOR TO THIS MACRO FILE.

NOTE:
1) THE UNUSED "#AF" FUNCTION DATA ADDRESS ORIGINS SHOULD BE DEFINED TO ADDRESS WORD 0 OF A DIFFERENT PAGE FROM THOSE PAGES OCCUPIED BY THE FUNCTION DATA FOR THE USED INPUTS.
2) THE UNUSED "#IF" FUNCTION DATA INDEX REGISTERS SHOULD CONTAIN 0 (OR A KNOWN REASONABLE VALUE WHICH WHEN ADDED TO THE CORRESPONDING "#AF" ADDRESS AT RUN TIME DOES NOT CAUSE AN ADDRESS CONFLICT).
3) DEFINE BREAKPOINT TABLE LENGTHS FOR UNUSED INPUTS TO BE 0.

FI5.3 MACROFILE

DESCRIPTION: INTERPOLATION FOR 5 VARIABLE FUNCTIONS.

F0(V0,W0,X0,Y0,Z0)
F1(V1,W1,X1,Y1,Z1)
F2(V2,W2,X2,Y2,Z2)

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
ARP	#DV0,#DW0,#DX0,#DY0,#DZ0 #DV1,#DW1,#DX1,#DY1,#DZ1 #DV2,#DW2,#DX2,#DY2,#DZ2	DELTA VALUES (REG. NUMBERS)
ARP	#F0,#F1,#F2	FUNCTION VALUES (REG. NUMBERS)
MAP	#AF0,#AF1,#AF2	FUNCTION DATA ADDRESS ORIGINS
MAP	#NV0,#NW0,#NX0,#NY0 #NV1,#NW1,#NX1,#NY1 #NV2,#NW2,#NX2,#NY2	BREAKPOINT TABLE LENGTHS
MAP	#IF0,#IF1,#IF2	FUNCTION DATA INDICES (REG. NUMBERS)
ARP	T0 - T11	SCRATCH REGISTERS

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	5	109	0	48

EXECUTION TIME: 11.4 MICRO-SEC.

POSSIBLE ERRORS:
- -ADDRESS ERROR DUE TO ILLEGAL FUNCTION DATA ORIGINS AND/OR INDICES.
- -ADDRESS ERROR DUE TO A CONFLICTING MEMORY REFERENCE DURING ONE OF THE 3 INSTRUCTIONS PRIOR TO THIS MACRO FILE.

NOTE: 1) THE UNUSED "#AF" FUNCTION DATA ADDRESS ORIGINS SHOULD BE DEFINED TO ADDRESS WORD 0 OF A DIFFERENT PAGE FROM THOSE PAGES OCCUPIED BY THE FUNCTION DATA FOR THE USED INPUTS.

2) THE UNUSED "#IF" FUNCTION DATA INDEX REGISTERS SHOULD CONTAIN 0 (OR A KNOWN REASONABLE VALUE WHICH WHEN ADDED TO THE CORRESPONDING "#AF" ADDRESS AT RUN TIME DOES NOT CAUSE AN ADDRESS CONFLICT).

3) DEFINE BREAKPOINT TABLE LENGTHS FOR UNUSED INPUTS TO BE 0.

TRMA.8 MACROFILE

DESCRIPTION: TRANSFER 8 DATA VALUES FROM MEMORY TO ARP "T" REGISTERS

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
ARP	#ATR0,#ATR1,...,#ATR7	ARP "T" REGISTER NUMBERS
MAP	#DM0,#DM1,...,#DM7	MEMORY ADDRESSES (TO BE ACCESSED IN SUCCESSIVE INSTRUCTIONS)

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	3	9	0	8

EXECUTION TIME: 1.4 MICRO-SEC.

POSSIBLE ERRORS: -ADDRESS ERROR DUE TO A CONFLICTING MEMORY REFERENCE DURING ONE OF THE 3 INSTRUCTIONS PRIOR TO THIS MACRO FILE.

TRMC.8 MACROFILE

DESCRIPTION: TRANSFER 8 DATA VALUES FROM MEMORY TO COP REGISTERS

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
COP	#CGR0,#CGR1,...,#CGR7	COP GENERAL REGISTER NUMBERS
MAP	#DM0,#DM1,...,#DM7	MEMORY ADDRESSES (TO BE ACCESSED IN SUCCESSIVE INSTRUCTIONS)

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	10	0	0	8

EXECUTION TIME: 1.4 MICRO-SEC.

POSSIBLE ERRORS: -ADDRESS ERROR DUE TO A CONFLICTING MEMORY REFERENCE DURING ONE OF THE 3 INSTRUCTIONS PRIOR TO THIS MACRO FILE.

TRMX.8 MACROFILE

DESCRIPTION: TRANSFER 8 DATA VALUES FROM MEMORY TO DEP "X" REGISTERS

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
DEP	#DXR0,#DXR1,...,#DXR7	DEP "X" REGISTER NUMBERS
MAP	#DM0,#DM1,...,#DM7	MEMORY ADDRESSES (TO BE ACCESSED IN SUCCESSIVE INSTRUCTIONS)

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	3	0	9	8

EXECUTION TIME: 1.4 MICRO-SEC.

POSSIBLE ERRORS: -ADDRESS ERROR DUE TO A CONFLICTING MEMORY REFERENCE DURING ONE OF THE 3 INSTRUCTIONS PRIOR TO THIS MACRO FILE.

TRMI.8 MACROFILE

DESCRIPTION: TRANSFER 8 DATA VALUES FROM MEMORY TO DEP "I" REGISTERS

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
DEP	#DIR0,#DIR1,...,#DIR7	DEP "I" REGISTER NUMBERS
MAP	#DM0,#DM1,...,#DM7	MEMORY ADDRESSES (TO BE ACCESSED IN SUCCESSIVE INSTRUCTIONS)

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	3	0	9	8

EXECUTION TIME: 1.4 MICRO-SEC.

POSSIBLE ERRORS: -ADDRESS ERROR DUE TO A CONFLICTING MEMORY REFERENCE DURING ONE OF THE 3 INSTRUCTIONS PRIOR TO THIS MACRO FILE.

TRME.8 MACROFILE

DESCRIPTION: TRANSFER 8 DATA VALUES FROM MEMORY TO EXTERNAL IOCC

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
COP	#IOC0,#IOC1,...,#IOC7	IOCC CHANNEL NUMBERS
MAP	#DM0,#DM1,...,#DM7	MEMORY ADDRESSES (EACH SUCCESSIVE 4 TO BE ACCESSED IN SUCCESSIVE INSTRUCTIONS)
COP	#OP	IO OPCODE

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	12	0	0	8

EXECUTION TIME: 1.8 MICRO-SEC.

POSSIBLE ERRORS: -ADDRESS ERROR DUE TO A CONFLICTING MEMORY REFERENCE DURING ONE OF THE 3 INSTRUCTIONS PRIOR TO THIS MACRO FILE.

TRAM.8 MACROFILE

DESCRIPTION: TRANSFER 8 DATA VALUES FROM ARP "T" REGISTERS TO MEMORY

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
ARP	#ATR0,#ATR1,...,#ATR7	ARP "T" REGISTER NUMBERS
MAP	#DM0,#DM1,...,#DM7	MEMORY ADDRESSES (TO BE ACCESSED IN SUCCESSIVE INSTRUCTIONS)

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	3	8	0	8

EXECUTION TIME: 1.0 MICRO-SEC.

POSSIBLE ERRORS:

-ADDRESS ERROR DUE TO A CONFLICTING MEMORY REFERENCE DURING ONE OF THE 3 INSTRUCTIONS PRIOR TO THIS MACRO FILE.

-ADDRESS ERROR DUE TO A CONFLICTING MEMORY REFERENCE DURING ONE OF THE 3 INSTRUCTIONS FOLLOWING THIS MACRO FILE.

TRCM.8 MACROFILE

DESCRIPTION: TRANSFER 8 DATA VALUES FROM COP REGISTERS TO MEMORY

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
COP	#CGR0,#CGR1,...,#CGR7	COP GENERAL REGISTER NUMBERS
MAP	#DM0,#DM1,...,#DM7	MEMORY ADDRESSES (TO BE ACCESSED IN SUCCESSIVE INSTRUCTIONS)

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	10	0	0	8

EXECUTION TIME: 1.0 MICRO-SEC.

POSSIBLE ERRORS:

-ADDRESS ERROR DUE TO A CONFLICTING MEMORY REFERENCE DURING ONE OF THE 3 INSTRUCTIONS PRIOR TO THIS MACRO FILE.

-ADDRESS ERROR DUE TO A CONFLICTING MEMORY REFERENCE DURING ONE OF THE 3 INSTRUCTIONS FOLLOWING THIS MACRO FILE.

TRXM.8 MACROFILE

DESCRIPTION: TRANSFER 8 DATA VALUES FROM DEP "X" REGISTERS TO MEMORY

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
DEP	#DXR0,#DXR1,...,#DXR7	DEP "X" REGISTER NUMBERS
MAP	#DM0,#DM1,...,#DM7	MEMORY ADDRESSES (TO BE ACCESSED IN SUCCESSIVE INSTRUCTIONS)

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	3	0	8	8

EXECUTION TIME: 1.0 MICRO-SEC.

POSSIBLE ERRORS:

-ADDRESS ERROR DUE TO A CONFLICTING MEMORY REFERENCE DURING ONE OF THE 3 INSTRUCTIONS PRIOR TO THIS MACRO FILE.

-ADDRESS ERROR DUE TO A CONFLICTING MEMORY REFERENCE DURING ONE OF THE 3 INSTRUCTIONS FOLLOWING THIS MACRO FILE.

TRIM.8 MACROFILE

DESCRIPTION: TRANSFER 8 DATA VALUES FROM DEP "I" REGISTERS TO MEMORY

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
DEP	#DIR0,#DIR1,...,#DIR7	DEP "I" REGISTER NUMBERS
MAP	#DM0,#DM1,...,#DM7	MEMORY ADDRESSES (TO BE ACCESSED IN SUCCESSIVE INSTRUCTIONS)

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	3	0	8	8

EXECUTION TIME: 1.0 MICRO-SEC.

POSSIBLE ERRORS:

-ADDRESS ERROR DUE TO A CONFLICTING MEMORY REFERENCE DURING ONE OF THE 3 INSTRUCTIONS PRIOR TO THIS MACRO FILE.

-ADDRESS ERROR DUE TO A CONFLICTING MEMORY REFERENCE DURING ONE OF THE 3 INSTRUCTIONS FOLLOWING THIS MACRO FILE.

TREM.8 MACROFILE

DESCRIPTION: TRANSFER 8 DATA VALUES FROM EXTERNAL IOCC TO MEMORY

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
COP	#IOC0,#IOC1,...,#IOC7	IOCC CHANNEL NUMBERS
MAP	#DM0,#DM1,...,#DM7	MEMORY ADDRESSES (EACH SUCCESSIVE 4 TO BE ACCESSED IN SUCCESSIVE INSTRUCTIONS)
COP	#OP	IO OPCODE

INSTRUCTION COUNTS:

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
	12	0	0	8

EXECUTION TIME: 1.8 MICRO-SEC.

POSSIBLE ERRORS: -ADDRESS ERROR DUE TO A CONFLICTING MEMORY REFERENCE DURING ONE OF THE 3 INSTRUCTIONS FOLLOWING THIS MACRO FILE.

TRCA.8 MACROFILE

DESCRIPTION: TRANSFER 8 DATA VALUES FROM COP REG'S TO ARP "T" REG'S

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
ARP	#ATR0,#ATR1,...,#ATR7	ARP "T" REGISTER NUMBERS
COP	#CGR0,#CGR1,...,#CGR7	COP GENERAL REGISTER NUMBERS

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	9	8	0	0

EXECUTION TIME: .9 MICRO-SEC.

TRCX.8 MACROFILE

DESCRIPTION: TRANSFER 8 DATA VALUES FROM COP REG'S TO DEP "X" REG'S

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
DEP	#DXR0,#DXR1,...,#DXR7	DEP "X" REGISTER NUMBERS
COP	#CGR0,#CGR1,...,#CGR7	COP GENERAL REGISTER NUMBERS

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	9	0	8	0

EXECUTION TIME: .9 MICRO-SEC.

TRCI.8 MACROFILE

DESCRIPTION: TRANSFER 8 DATA VALUES FROM COP REG'S TO DEP "I" REG'S

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
DEP	#DIR0,#DIR1,...,#DIR7	DEP "I" REGISTER NUMBERS
COP	#CGR0,#CGR1,...,#CGR7	COP GENERAL REGISTER NUMBERS

	AD-10 PROCESSOR COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	9	0	8	0

EXECUTION TIME: .9 MICRO-SEC.

TRCE.8 MACROFILE

DESCRIPTION: TRANSFER 8 DATA VALUES FROM COP REG'S TO EXTERNAL IOCC

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
COP	#CGR0,#CGR1,...,#CGR7	COP GENERAL REGISTER NUMBERS
COP	#IOC0,#IOC1,...,#IOC7	EXTERNAL IOCC CHANNEL NUMBERS
COP	#OP	IO OPCODE

	AD-10 PROCESSOR COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	8	0	0	0

EXECUTION TIME: .8 MICRO-SEC.

TRAC.8 MACROFILE

DESCRIPTION: TRANSFER 8 DATA VALUES FROM ARP "T" REG'S TO COP REG'S

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
ARP	#ATR0,#ATR1,...,#ATR7	ARP "T" REGISTER NUMBERS
COP	#CGR0,#CGR1,...,#CGR7	COP GENERAL REGISTER NUMBERS

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	10	8	0	0

EXECUTION TIME: 1.0 MICRO-SEC.

TRXC.8 MACROFILE

DESCRIPTION: TRANSFER 8 DATA VALUES FROM DEP "X" REG'S TO COP REG'S

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
DEP	#DXR0,#DXR1,...,#DXR7	DEP "X" REGISTER NUMBERS
COP	#CGR0,#CGR1,...,#CGR7	COP GENERAL REGISTER NUMBERS

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	10	0	8	0

EXECUTION TIME: 1.0 MICRO-SEC.

TRIC.8 MACROFILE

DESCRIPTION: TRANSFER 8 DATA VALUES FROM DEP "I" REG'S TO COP REG'S

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
DEP	#DIR0,#DIR1,...,#DIR7	DEP "I" REGISTER NUMBERS
COP	#CGR0,#CGR1,...,#CGR7	COP GENERAL REGISTER NUMBERS

	AD-10 PROCESSOR COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	10	0	8	0

EXECUTION TIME: 1.0 MICRO-SEC.

TREC.8 MACROFILE

DESCRIPTION: TRANSFER 8 DATA VALUES FROM EXTERNAL IOCC TO COP REG'S

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
COP	#CGR0,#CGR1,...,#CGR7	COP GENERAL REGISTER NUMBERS
COP	#IOC0,#IOC1,...,#IOC7	EXTERNAL IOCC CHANNEL NUMBERS
COP	#OP	IO OPCODE

	AD-10 PROCESSOR COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	13	0	0	0

EXECUTION TIME: 1.3 MICRO-SEC.

TREXM.8 MACROFILE

DESCRIPTION: TRANSFER 8 DATA VALUES FROM IOCC TO DEP "X" REG'S AND MEMORY

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
DEP	#DXR0,#DXR1,...,#DXR7	DEP "X" REGISTER NUMBERS
MAP	#DM0,#DM1,...,#DM7	MEMORY ADDRESSES
COP	#IOC0,#IOC1,...,#IOC7	EXTERNAL IOCC CHANNEL NUMBERS
COP	#OP	IO OPCODE

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	10	0	8	9

EXECUTION TIME: 1.8 MICRO-SEC.

POSSIBLE ERRORS: -ADDRESS ERROR DUE TO A CONFLICTING MEMORY REFERENCE DURING ONE OF THE 3 INSTRUCTIONS FOLLOWING THIS MACRO FILE.

TRAEM.8 MACROFILE

DESCRIPTION: TRANSFER 8 DATA VALUES FROM ARP "T" REG'S TO EXTERNAL IOCC AND MEMORY

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
ARP	#ATR0,#ATR1,...,#ATR7	ARP "T" REGISTER NUMBERS
COP	#IOC0,#IOC1,...,#IOC7	EXTERNAL IOCC CHANNEL NUMBERS
MAP	#DM0,#DM1,...,#DM7	MEMORY ADDRESSES
COP	#OP	IO OPCODE

	AD-10 PROCESSOR COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	10	8	0	8

EXECUTION TIME: 1.0 MICRO-SEC.

POSSIBLE ERRORS:

-ADDRESS ERROR DUE TO A CONFLICTING MEMORY REFERENCE DURING ONE OF THE 3 INSTRUCTIONS PRIOR TO THIS MACRO FILE.

-ADDRESS ERROR DUE TO A CONFLICTING MEMORY REFERENCE DURING ONE OF THE 3 INSTRUCTIONS FOLLOWING THIS MACRO FILE.

LOADA.8 MACROFILE

DESCRIPTION: LOAD IMMEDIATE DATA INTO ARP "T" REGISTERS

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
ARP	#ATR0,#ATR1,...,#ATR7	ARP "T" REGISTER NUMBERS
DAT	#IDAT0,#IDAT1,...,#IDAT7	IMMEDIATE DATA

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	10	9	0	0

EXECUTION TIME: 1.0 MICRO-SEC.

LOADC.8 MACROFILE

DESCRIPTION: LOAD IMMEDIATE DATA INTO COP GENERAL REGISTERS

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
COP	#CGR0,#CGR1,...,#CGR7	COP "X" REGISTER NUMBERS
DAT	#IDAT0,#IDAT1,...,#IDAT7	IMMEDIATE DATA

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	10	0	8	0

EXECUTION TIME: 1.0 MICRO-SEC.

LOADX.8 MACROFILE

DESCRIPTION: LOAD IMMEDIATE DATA INTO DEP "X" REGISTERS

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
DEP	#DXR0,#DXR1,...,#DXR7	DEP "X" REGISTER NUMBERS
DAT	#IDAT0,#IDAT1,...,#IDAT7	IMMEDIATE DATA

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	10	0	9	0

EXECUTION TIME: 1.0 MICRO-SEC.

LOADI.8 MACROFILE

DESCRIPTION: LOAD IMMEDIATE DATA INTO DEP "I" REGISTERS

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
DEP	#DIR0,#DIR1,...,#DIR7	DEP "I" REGISTER NUMBERS
DAT	#IDAT0,#IDAT1,...,#IDAT7	IMMEDIATE DATA

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	10	0	9	0

EXECUTION TIME: 1.0 MICRO-SEC.

LOADM.8 MACROFILE

DESCRIPTION: LOAD IMMEDIATE DATA INTO MEMORY

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
MAP	#DM0,#DM1,...,#DM7	MEMORY ADDRESSES (TO BE ACCESSED IN SUCCESSIVE INSTRUCTIONS)
DAT	#IDAT0,#IDAT1,...,#IDAT7	IMMEDIATE DATA

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	10	0	0	9

EXECUTION TIME: 1.0 MICRO-SEC.

POSSIBLE ERRORS:

-ADDRESS ERROR DUE TO A CONFLICTING MEMORY REFERENCE DURING ONE OF THE 3 INSTRUCTIONS PRIOR TO THIS MACRO FILE.

-ADDRESS ERROR DUE TO A CONFLICTING MEMORY REFERENCE DURING ONE OF THE 3 INSTRUCTIONS FOLLOWING THIS MACRO FILE.

SGN.2 MACROFILE

DESCRIPTION: COMPUTES "SGN" FUNCTION

```
#SGNX  = SGN(X) = 1  (INTEGER) ; X >= 0
                = -1 (INTEGER) ; X < 0

#SGNY  = SGN(Y) = (AS ABOVE)
```

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
ARP	#X,#Y	INPUT ARGUMENTS (REG. NUMBERS)
ARP	#SGNX,#SGNY	RESULTS (REG. NUMBERS)
ARP	T0,T1	SCRATCH REGISTERS

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	5	7	0	0

EXECUTION TIME: .8 MICRO-SEC.

CTR.3 MACROFILE

DESCRIPTION: PERFORMS COORDINATE TRANSFORMATIONS FOR 3 VECTORS.

```
XPI =  XI*COS(AI) + YI*SIN(AI)  ;  I=0,2
YPI = -XI*SIN(AI) + YI*COS(AI)  ;  I=0,2

-1.0 <= AI < 1.0 ; UNIT AI = 180 DEGREES
```

NOTE: THIS MACROFILE CAN BE USED TO ROTATE 3 INDEPENDENT VECTORS EACH THROUGH A SEPARATE ANGLE. IT CAN ALSO BE USED TO ROTATE ONE VECTOR THROUGH 3 SEPARATE ANGLES BY SPECIFING THE INPUT TO THE SECOND ROTATION AS THE OUTPUT OF THE FIRST AND THE INPUT TO THE THIRD ROTATION AS THE OUTPUT OF THE SECOND TRANSFORMATION (SEE EXAMPLE ON NEXT PAGE).

USER DEFINED MACRO ARGUMENTS:

USED IN CODE FOR	SYMBOLS	MEANING
ARP	#X0,#X1,#X2,#Y0,#Y1,#Y2	INPUT COMPONENTS (REG. NUMBERS)
ARP	#A0,#A1,#A2	ROTATION ANGLES (REG. NUMBERS)
ARP	#XP0,#XP1,#XP2,#YP0,#YP1,#YP2	OUTPUT COMPONENTS (REG. NUMBERS)
ARP	#SIN0,#SIN1,#SIN2 #COS0,#COS1,#COS2	OUTPUT SIN'S AND COS'S (REG NUMBERS)
MAP	#SIN,#COS	ORIGINS OF SIN AND COS FUNCTION DATA TABLES. EACH TABLE HAS 513 DATA VALUES FOR EQUAL SPACED BREAKPOINTS OVER THE RANGE: -180 TO +180 DEG.
ARP	T0,T1,T2	SCRATCH REGISTERS
DEP	I0,I1,I2	SCRATCH REGISTERS

INSTRUCTION COUNTS:

AD-10 PROCESSOR			
COP	ARP	DEP	MAP
13	31	3	6

EXECUTION TIME: 3.2 MICRO-SEC.

EXAMPLE:

SUPPOSE WE HAVE A VECTOR WITH 3 ORTHOGONAL COMPONENTS [X,Y,Z] AND WE WISH TO PERFORM A 3 ANGLE COORDINATE TRANSFORMATION THROUGH THE ANGLES AXY, AXZ, AND AYZ (I.E. ANGLES IN THE X-Y, X-Z, AND Y-Z PLANES). THE ARGUMENTS TO THIS MACROFILE WOULD BE DEFINED AS FOLLOWS:

```
#X   .DEFINE X   ,TX0,TY0
#Y   .DEFINE Y   ,Z   ,TX1
#A   .DEFINE AXY,AXZ,AYZ
```

IF THE 3 VECTOR COMPONENTS OF THE RESULT ARE TO BE [XX,YY,ZZ], THEN THE OUTPUTS OF THE MACROFILE WOULD BE DEFINED AS FOLLOWS:

```
#XP .DEFINE TX0,TX1,YY
#YP .DEFINE TY0,XX ,ZZ
```

THE EQUATIONS SOLVED WITH THESE ARGUMENT DEFINITIONS ARE:

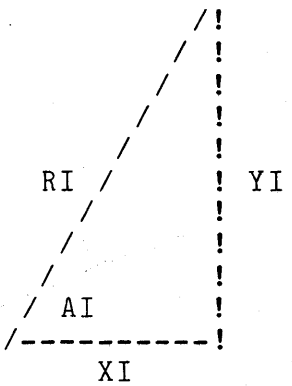
```
TX0 = #XP0 = Y*SIN(AXY) + X*COS(AXY)
TY0 = #YP0 = Y*COS(AXY) - X*SIN(AXY)

TX1 = #XP1 = Z*SIN(AXZ) + TX0*COS(AXZ)
XX  = #YP1 = Z*COS(AXZ) - TX0*SIN(AXZ)

YY  = #XP2 = TX1*SIN(AYZ) + TY0*COS(AYZ)
ZZ  = #YP2 = TX1*COS(AYZ) - TY0*SIN(AYZ)
```

IRS.3 MACROFILE

DESCRIPTION: INVERSE RESOLUTION



INPUTS: XI,YI ; I=0,2

OUTPUTS:
- AI (RADIANS/PI) ; I=0,2
- SIN(AI),COS(AI) ; I=0,2
- RI ; I=0,2
- HRI = RI/2 ; I=0,2

USER DEFINED MACRO ARGUMENTS

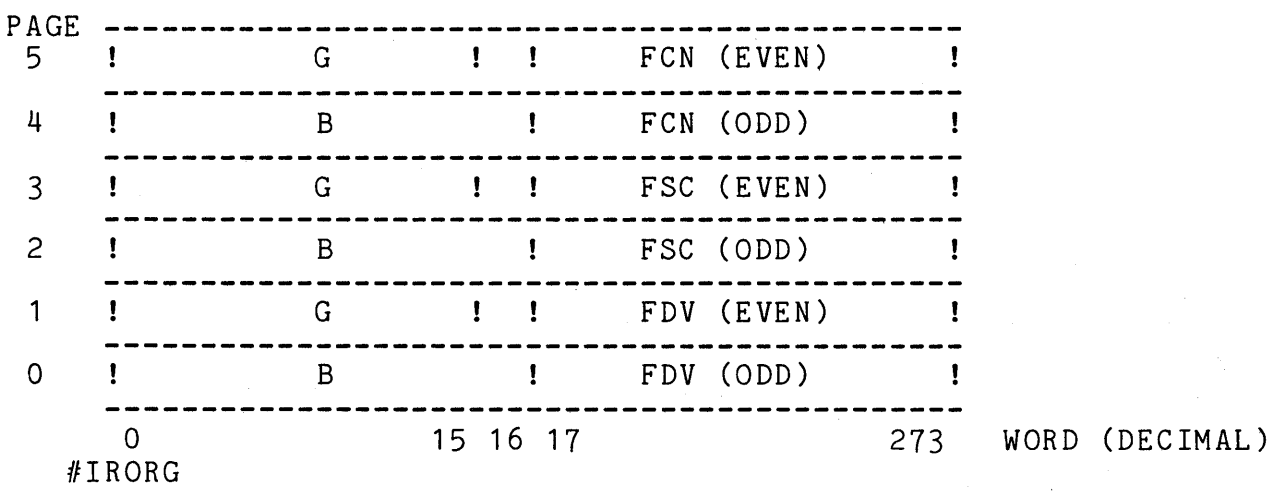
USED IN CODE FOR	SYMBOLS	MEANING
ARP	#X0,#Y0,#X1,#Y1,#X2,#Y2	INPUTS
MAP	#IRORG	ORIGIN OF FUNCTION DATA TABLES
ARP	#A0,#A1,#A2	OUTPUT ANGLES (UNIT A = 180 DEG.)
ARP	#SIN0,#SIN1,#SIN2 #COS0,#COS1,#COS2	OUTPUT SIN'S AND COS'S
ARP	#R0,#R1,#R2	OUTPUT VECTORS
ARP	#HR0,#HR1,#HR2	OUTPUT VECTORS DIVIDED BY 2
ARP	T0-T4	SCRATCH REGISTERS
MAP/DEP	IX0-IX11	SCRATCH "I" AND "X" REGISTERS

NOTE: THE #A, #SIN, #COS, #R, AND #HR OUTPUT REGISTERS ARE USED INTERNALLY BY THIS MACROFILE FOR INTERMEDIATE RESULTS, THUS THEY MUST BE UNIQUE REGISTERS. FOR EXAMPLE, #A0 MUST NOT BE THE SAME ARP REGISTER AS #SIN0.

	AD-10 PROCESSOR			
	COP	ARP	DEP	MAP
INSTRUCTION COUNTS:	12	102	32	33

EXECUTION TIME: 11.5 MICRO-SEC.

MAP OF FUNCTION DATA REQUIRED BY THIS MACROFILE:



THE DATA ARRAYS ARE DEFINED AS FOLLOWS:

```
B(0)      = -1.0
B(I)      =  1.0-2**(1-I)  ;  I=1,16

G(I)      =  2**I          ;  I=0,14
G(15)     =  (2**15)-1

FDV(V)    =  1.0/(V-1.0)   ;  V= -1.0 TO 0

FSC(SC)   =  SC/SQRT(1.0-2*!SC!+2*SC**2)  ;  SC= -1.0 TO 1.0

FCN(CN)   =  ARCTAN((1/CN)+1)  ;  CN = -1.0 TO 0.0
             ARCTAN((1/CN)-1)  ;  CN =  0.0 TO 1.0
```

(NOTE: "FDV", "FSC", AND "FCN" REQUIRE 513 EQUALLY SPACED VALUES OVER THE RANGE SPECIFIED).

A FORTRAN PROGRAM WHICH GENERATES ALL DATA ARRAYS REQUIRED BY THIS MACROFILE IS SUPPLIED WITH THE MACROFILE LIBRARY. THE FOLLOWING IS AN EXAMPLE "ADX" COMMAND FILE TO LOAD THIS DATA FOR #IRORG=0 :

```
LOAD B.DAT/UN:0:0/IN
LOAD G.DAT/UN:1:0/IN
LOAD B.DAT/UN:2:0/IN
LOAD G.DAT/UN:3:0/IN
LOAD B.DAT/UN:4:0/IN
LOAD G.DAT/UN:5:0/IN
LOAD FDV.DAT/AL:0:17/RS
LOAD FSC.DAT/AL:2:17/RS
LOAD FCN.DAT/AL:4:17/RS
```

CALCULATIONS PERFORMED:

1) COMPUTE

$$SN = \frac{Y}{!X! + !Y!} \quad ; \quad CN = \frac{X}{!X! + !Y!}$$

A) RE-SCALE RANGE OF DENOMINATOR OF DIVIDE

$U = -!X!-!Y!+1 \qquad (-1.0 <= U < 1.0)$

B) PERFORM BINARY SEARCH ON U USING TABLE "B(I)" TO GET "I" SUCH THAT:

$B(I) <= U < B(I+1) \quad ; \quad I=0,15$

$B(0) = -1.0$
$B(I) = 1.0-2**(I-1) \quad ; \quad I=1,16$

$G(I) = 2**I \quad ; \quad I=0,14$
$G(15) = 2**15-1$

C) SCALE DENOMINATOR OF DIVIDE BY SHIFTING "U" LEFT SO THAT THE MOST SIGNIFICANT DIGIT IS NEXT TO THE DECIMAL POINT:

$V = [U - B(I+1)] * G(I)$
$= -[!X! + !Y!] * G(I) + 1$
$(-1.0 <= V <= 0)$

THIS PRESERVES 1 LSB ACCURACY IN THE DIVISION CALCULATION.

D) PERFORM SHIFT SEARCH AND DELTA CALCULATION ON V AND PERFORM LINEAR INTERPOLATION ON DIVIDE FUNCTION:

$FDV(V) = 1 / (V-1)$

E) MULTIPLY "X" AND "Y" NUMBERATORS TIMES THE DIVIDE FUNCTION:

$WY = Y*FDV(V) \quad ; \quad WX = XX*FDV(V)$

NOTE: IF $!X!+!Y!=0$, THEN $XX = 2**(-15)$
ELSE $XX = X$

F) FINISH SCALING OF DIVISIONS:

$SN = -WY*G(I) \quad ; \quad CN = -WX*G(I)$

CALCULATIONS PERFORMED (CONT.):

2) COMPUTE SIN(A), COS(A), AND THE ANGLE "A" AS FUNCTIONS OF "SN", "CN", AND "Y".

A) PERFORM SHIFT SEARCH AND DELTA CALCULATION ON "SN" AND"CN".

B) COMPUTE THE SIGN OF "Y", I.E. SGN(Y).

C) PERFORM THE LINEAR INTERPOLATION ON FUNCTIONS "FSC" AND "FCN" WHICH ARE DEFINED AS FOLLOWS:

$$FSC(SC) = \frac{SC}{SQRT(1-2*!SC!+2*SC**2)}$$

FCN(CN) = ARCTAN((1/CN)-1) ; CN >= 0
= ARCTAN((1/CN)+1) ; CN <= 0

SIN(A) = FSC(SN)

COS(A) = FSC(CN)

A = SGN(Y)*FCN(CN)

3) COMPUTE: R = X*COS(A) + Y*SIN(A)
HR = [X*COS(A) + Y*XIN(A)]/2

(NOTE: THE CALCULATION OF "R" WILL OVERANGE AND RETURN R=1.0-2**(-15) IF X AND Y ARE SUCH THAT: SQRT(X**2 + Y**2) > 1.0)

RANGES OF MAJOR VARIABLES FOR THE CALCULATIONS PERFORMED:

I	!X!+!Y!	U = -!X!-!Y!+1	B(I)	G(I)
0	1 TO 2	-1 TO 0	-1.0	1
1	2**(-1) TO 1	0 TO 1-2**(-1)	0	2
2	2**(-2) TO 2**(-1)	1-2**(-1) TO 1-2**(-2)	1-2**(-1)	2**2
3	2**(-3) TO 2**(-2)	1-2**(-2) TO 1-2**(-3)	1-2**(-2)	2**3
4	2**(-4) TO 2**(-3)	1-2**(-3) TO 1-2**(-4)	1-2**(-3)	2**4
5	2**(-5) TO 2**(-4)	1-2**(-4) TO 1-2**(-5)	1-2**(-4)	2**5
6	2**(-6) TO 2**(-5)	1-2**(-5) TO 1-2**(-6)	1-2**(-5)	2**6
7	2**(-7) TO 2**(-6)	1-2**(-6) TO 1-2**(-7)	1-2**(-6)	2**7
8	2**(-8) TO 2**(-7)	1-2**(-7) TO 1-2**(-8)	1-2**(-7)	2**8
9	2**(-9) TO 2**(-8)	1-2**(-8) TO 1-2**(-9)	1-2**(-8)	2**9
10	2**(-10) TO 2**(-9)	1-2**(-9) TO 1-2**(-10)	1-2**(-9)	2**10
11	2**(-11) TO 2**(-10)	1-2**(-10) TO 1-2**(-11)	1-2**(-10)	2**11
12	2**(-12) TO 2**(-11)	1-2**(-11) TO 1-2**(-12)	1-2**(-11)	2**12
13	2**(-13) TO 2**(-12)	1-2**(-12) TO 1-2**(-13)	1-2**(-12)	2**13
14	2**(-14) TO 2**(-13)	1-2**(-13) TO 1-2**(-14)	1-2**(-13)	2**14
15	2**(-15) TO 2**(-14)	1-2**(-14) TO 1.0	1-2**(-14)	32767
16			1-2**(-15)	

NOTE: IF !X!+!Y! = 0 THEN -!X!-!Y!+1 WILL OVERANGE AND ROUND DOWN TO 1-2**(-15), THEREFORE X=Y=0 IS EQUIVALENT TO EITHER X OR Y = 1 LSB.

SYMBOLS DEFINED AND USED IN MACROFILE:

USED IN CODE FOR	SYMBOLS	MEANING
ARP	##MX0,##MS1,##MX2	!X! VALUES
ARP	##MY0,##MY1,##MY2	!Y! VALUES
ARP	##XX0,##XX1,##XX2	"XX" VALUES
ARP	##G0,##G1,##G2	"G(I)" VALUES
ARP	##U0,##U1,##U2	"U" VALUES
ARP	##V0,##V1,##V2	"V" VALUES
ARP	##DV0,##DV1,##DV2	DELTA "V" VALUES
ARP	##SGN0,##SGN1,##SGN2	"SGN" VALUES
ARP	##CN0,##CN1,##CN2	"CN" VALUES
ARP	##SN0,##SN1,##SN2	"SN" VALUES
ARP	##DCN0,##DCN1,##DCN2	DELTA "CN" VALUES
ARP	##DSN0,##DSN1,##DSN2	DELTA "SN" VALUES
ARP	##FCN0,##FCN1,##FCN2	"FCN" VALUES
MAP/DEP	##IXU0,##IXU1,##IXU2 ##IV0,##IV1,##IV2 ##ICN0,##ICN1,##ICN2 ##ISN0,##ISN1,##ISN2	"U", "V", "CN", AND "SN" VALUES IN THE "X" REGISTERS AND INDEX VALUES IN THE "I" REGISTERS.
MAP	##PAG,##WRD	"PAGE" AND "WORD" CORRESPONDING TO #IRORG.

EXAMPLE AD-10 MVFG PROGRAM USING MACROFILES

THE FOLLOWING EXAMPLE ILLUSTRATES HOW THE AD-10 MACROFILES ARE USED TO PROGRAM A TYPICAL MULTIVARIABLE FUNCTION GENERATION APPLICATION. FIRST, THE GENERAL PROCEDURE WHICH SHOULD BE FOLLOWED TO PROGRAM MOST MVFG PROBLEMS IS OUTLINED BELOW:

1) LIST VARIABLES WHICH WILL BE USED AS ARGUMENTS OF FUNCTIONS TO BE GENERATED. ALL FUNCTIONS OF A PARTICULAR VARIABLE MUST BE DEFINED AT THE SAME SET OF BREAKPOINTS FOR THAT VARIABLE. IF THIS CONDITION IS NOT POSSIBLE TO MEET FOR A PARTICULAR VARIABLE, THEN CONSIDER EACH REFERENCE OF THAT VARIABLE WHICH REQUIRES A DIFFERENT BREAKPOINT SET AS A SEPARATE VARIABLE.

2) DETERMINE THE BREAKPOINTS FOR EACH VARIABLE AND DECIDE WHICH SEARCH SCHEME WILL BE USED FOR EACH. THE BINARY SEARCH MACROFILE (BD.6) ALLOWS FROM 2 TO 33 BREAKPOINTS TO BE SPACED AT THE USERS DISCRETION; THE SHIFT SEARCH SCHEME ALLOWS (2**N)+1 BREAKPOINTS (N=2,15) WHICH ARE EQUALLY SPACED FROM -1.0 TO +1.0 .

3) LIST ALL UNIQUE VARIABLE SETS WHICH WILL BE USED AS FUNCTION ARGUMENTS. EACH VARIABLE SET OF 2 OR MORE VARIABLES WILL REQUIRE A FUNCTION DATA POINTER CALCULATION USING ONE OF THE "PT?.3" MACROFILES.

4) LAYOUT AD-10 DATA MEMORY MAP DEFINING WHERE IN THE MEMORY THE VARIOUS ARRAYS OF FUNCTION DATA AND BREAKPOINT DATA TABLES ARE TO BE STORED.

5) WRITE AD-10 PROGRAM USING THE MVFG MACROFILES. THE GENERAL FORMAT FOR A FUNCTION GENERATION PROGRAM IS AS FOLLOWS:
   - [A] TRANSFER INPUT VARIABLES TO "DEP" USING THE APPROPRIATE "TR.." MACROFILES.
   - [B] PERFORM BINARY OR SHIFT SEARCH AND DELTA CALCULATION ON ALL INPUT VARIABLES USING APPROPRIATE MACROFILES.
   - [C] PERFORM FUNCTION DATA POINTER CALCULATION USING THE APPROPRIATE "PT..." MACROFILES.
   - [D] PERFORM FUNCTION INTERPOLATIONS USING THE "FI..." MACROFILES.
   - [E] TRANSFER RESULTS FROM ARP TO DESIRED DESTINATION USING THE APPROPRIATE "TR..." MACROFILE.

6) ORGANIZE THE FUNCTION DATA SUCH THAT IT IS CONSISTENT WITH THE NUMBER OF BREAKPOINTS FOR EACH VARIABLE IN THE CORRESPONDING VARIABLE SET FOR THAT FUNCTION. ORDER THE FUNCTION DATA IN A LINEAR ARRAY SUCH THAT THE VALUES FOR CHANGES IN THE FIRST VARIABLE ARE ENTERED FIRST AND VALUES FOR CHANGES IN THE LAST VARIABLE ARE ENTERED LAST. NOTE THAT THIS IS CONSISTENT WITH THE NORMAL INTERNAL STORAGE OF MULTI-DIMENSIONED ARRAYS IN FORTRAN.

7) PLACE FUNCTION DATA AND BREAKPOINT DATA INTO DIRECT ACCESS FILES FOR LOADING INTO THE AD-10 DATA MEMORY USING "ADX".

EXAMPLE (CONT.)

THE SPECIFIC FUNCTION GENERATION PROBLEM TO BE SOLVED IN THIS EXAMPLE IS THE AD-10 PORTION OF A HYBRID SIMULATION OF THE LONGITUDINAL FLIGHT EQUATIONS FOR AN AIRCRAFT. THE AD-10 ACCEPTS THREE INPUTS FROM THE ANALOG COMPUTER: A (ANGLE OF ATTACK), H (ALTITUDE), AND V (VELOCITY). THE AD-10 THEN COMPUTES M (MACH NUMBER) AS A FUNCTION OF "H" AND "V" AND THEN GENERATES FIVE AERODYNAMIC COEFFICIENTS WHICH ARE FUNCTIONS OF "A", "M", AND "H" ; THESE COEFFICIENTS CONSIST OF THREE FUNCTIONS OF 3 VARIABLES AND 2 FUNCTIONS OF 2 VARIABLES. THE AD-10 WILL ALSO BE CALLED UPON TO GENERATE THE SINE AND COSINE OF THE ANGLE "A" AND FINALLY WILL OUTPUT THE 7 INTERPOLATED FUNCTION VALUES ON DAC'S TO THE ANALOG COMPUTER.

THE PROBLEM CAN BE SUMMARIZED AS FOLLOWS:

INPUTS: A,H,V

COMPUTES: M=2*V*F6(H)

F11(A1,M,H), F12(A1,M,H), F13(A1,M,H), F14(M,H), F15(M,H), F8(A2), F9(A2)

NOTE: A1 AND A2 ARE THE SAME VARIABLE (A), BUT A DIFFERENT SET OF BREAKPOINTS IS USED FOR EACH.

OUTPUTS: F11, F12, F13, F14, F15, F8, F9

BREAKPOINTS: ALL VARIABLES WILL HAVE UNEQUALLY SPACED BREAKPOINTS.

A1 HAS 16 BREAKPOINTS
A2 HAS 33 BREAKPOINTS
M HAS 21 BREAKPOINTS
H HAS 12 BREAKPOINTS

DATA FILES: A FORTRAN PROGRAM WAS USED TO GENERATE THE FUNCTION DATA USING ANALYTIC APPROXIMATIONS OF TYPICAL AREODYNAMIC FUNCTIONS, AND IT ALSO GENERATES THE BREAKPOINT DATA FILES. A LISTING OF THIS PROGRAM IS INCLUDED IN THIS EXAMPLE.

EXAMPLE (CONT.)
AD-10 PROGRAM USING MACROFILES

THE FOLLOWING IS A LISTING OF THE AD-10 SOURCE PROGRAM FOR THIS EXAMPLE PROBLEM. THIS PROGRAM IS HEAVILY COMMENTED FOR THE SAKE OF CLEARLY PRESENTING THIS EXAMPLE (COMMENT LINES BEGIN WITH A "!"). THE TOTAL NUMBER OF ASSEMBLER INSTRUCTIONS IS ROUGHLY 150 LINES, OF WHICH ONLY ABOUT 30 ARE ACTUAL AD-10 MACHINE INSTRUCTIONS AND THE REST CONSIST OF SYMBOL DEFINITIONS, ASSEMBLER DIRECTIVES, AND MACROFILE CALLS (I.E. ".INCLUDE" DIRECTIVES). WHEN THIS PROGRAM WAS ASSEMBLED WITH THE MACROFILES INCLUDED IN THE SOURCE CODE LINE COUNT, THE ASSEMBLER PRODUCED APPROXIMATELY 1500 LINES OF OUTPUT IN THE LISTING. THUS, FOR THIS EXAMPLE THE USER ONLY HAD TO WRITE 10 PERCENT OF THE TOTAL SOURCE CODE ASSEMBLED, AND MOST OF THAT CONSISTED OF SYMBOL DEFINITIONS. MOST TYPICAL MVFG PROBLEMS CAN BE PROGRAMMED ON THE AD-10 IN A SIMILIAR FASHION USING MACROFILES. THIS DRAMATICALLY ILLUSTRATES THE POWER OF THE AD-10 MACROFILE LIBRARY.

```
!*************************************************************************
!
!  AD-10 FUNCTION GENERATION BENCHMARK PROBLEM (BENCH.ASM)
!
!  INPUTS:     A,H,V
!
!  COMPUTES:   M=2*V*F6(H)
!
!              F11(A1,M,H), F12(A1,M,H), F13(A1,M,H), F14(M,H), F15(M,H),
!              F8(A2), F9(A2)
!
!              NOTE:  A1 AND A2 ARE THE SAME VARIABLE (A), BUT A
!              DIFFERENT SET OF BREAKPOINTS IS USED FOR EACH.
!
!  OUTPUTS:    F11, F12, F13, F14, F15, F8, F8
!
!*************************************************************************
!
        .PRON           !PRINT MACROFILE CODE
        .DECIMAL
!
!  DEFINE ARP TEMPORARY REGISTERS
!
        .ARP
T       .DEFINE 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
!
!  DEFINE ARP REGISTERS FOR DELTA QUANTITIES
!
DELA1   .EQU    16
DELA2   .EQU    17
DELM    .EQU    18
DELH    .EQU    19
```

EXAMPLE (CONT.)
AD-10 PROGRAM USING MACROFILES

```
!
!  DEFINE ARP REGISTERS FOR FUNCTION VALUES
!
F11     .EQU    20
F12     .EQU    21
F13     .EQU    22
F14     .EQU    23
F15     .EQU    24
F6      .EQU    25
F8      .EQU    26
F9      .EQU    27
!
!  DEFINE  MAP/DEP REGISTERS FOR INPUT VARIABLES
!
        .DEP
A1DEP   .EQU    0
A2DEP   .EQU    1
MDEP    .EQU    2
HDEP    .EQU    3
VDEP    .EQU    4
!
!  DEFINE MAP FUNCTION DATA INDEX REGISTERS FOR EACH VARIABLE SET
!
        .MAP
IAMH    .EQU    5       ! INDEX FOR A1,M,H SET
IMH     .EQU    6       ! INDEX FOR M,H SET
!
!  DEFINE NUMBER OF BREAKPOINTS FOR EACH VARIABLE
!
NBPA1   .EQU    16
NBPA2   .EQU    33
NBPM    .EQU    21
NBPH    .EQU    12
!
!  DEFINE MAPPING BETWEEN INPUT/OUTPUT VARIABLES AND I/O PORTS
!
        .COP
ADC     .DEFINE 0,1,2,3,4
DAC     .DEFINE 5,6,7,8,9,10,11,12,13,14,15
AIN     .EQU    ADC0
HIN     .EQU    ADC1
VIN     .EQU    ADC2
F11OUT  .EQU    DAC0
F12OUT  .EQU    DAC1
F13OUT  .EQU    DAC2
F14OUT  .EQU    DAC3
F15OUT  .EQU    DAC4
F8OUT   .EQU    DAC5
F9OUT   .EQU    DAC6
!
!  DEFINE I/O COMMANDS
!
SET     .EQU    0
INIT    .EQU    1
READ    .EQU    2
WRITE   .EQU    2
UPDATE  .EQU    3
```

EXAMPLE (CONT.)
AD-10 PROGRAM USING MACROFILES

```
        .PAGE
!
!  AD-10 DATA MEMORY MAP FOR THIS PROGRAM
!
!PAGE
!    ------------------------------------------------------------------
! 5 !  F13 (ODD)   !  F6  (ODD)   !                !       !         !
!   !--------------!--------------!                !       !         !
! 4 !  F13 (EVEN)  !  F6  (EVEN)  !                !       !         !
!   !--------------!--------------!-------------!  !       !         !
! 3 !  F12 (ODD)   !  F15 (ODD)   !  F9 (ODD)   !  ! I/O   !   BP    !
!   !--------------!--------------!-------------!  !       !         !
! 2 !  F12 (EVEN)  !  F15 (EVEN)  !  F9 (EVEN)  !  ! DATA  !  DATA   !
!   !--------------!--------------!-------------!  !       !         !
! 1 !  F11 (ODD)   !  F14 (ODD)   !  F8 (ODD)   !  !BUFFER! TABLES  !
!   !--------------!--------------!-------------!  !       !         !
! 0 !  F11 (EVEN)  !  F14 (EVEN)  !  F8 (EVEN)  !  !       !         !
!   !--------------!--------------!-------------!--!-------!---------!
!    0              2016           2142          3397    4000 (DECIMAL)
!    0              3740           4136          7635    7640 (OCTAL)
!                                 WORD ADDRESS
!
!  DEFINE MEMORY ADDRESSES FOR INPUT/OUTPUT QUANTITIES
!
        .MAP
AMEM    .EQU    0:3999      ! PAGE:WORD  (UNALIGNED ADDRESS)
MMEM    .EQU    3:3999
HMEM    .EQU    1:3999
VMEM    .EQU    2:3999
F11MEM  .EQU    0:3998
F12MEM  .EQU    1:3998
F13MEM  .EQU    2:3998
F14MEM  .EQU    3:3998
F15MEM  .EQU    4:3998
F6MEM   .EQU    1:3997
F8MEM   .EQU    5:3998
F9MEM   .EQU    0:3997
DRAIN   .EQU    63:4095     ! A PLACE TO DUMP THINGS
FAUCET  .EQU    DRAIN       ! A PLACE TO READ GARBAGE DATA
!
!  DEFINE FUNCTION DATA BASE ADDRESSES
!
AF11    .EQU    0::0        ! PAGE::WORD  (ALIGNED ADDRESS)
AF12    .EQU    2::0
AF13    .EQU    4::0
AF14    .EQU    0::2016
AF15    .EQU    2::2016
AF6     .EQU    4::2016
AF8     .EQU    0::2142
AF9     .EQU    2::2142
!
!  DEFINE ORIGIN OF BREAKPOINT DATA TABLES
!
BPORG   .EQU    0:4000
        .PAGE
!
```

EXAMPLE (CONT.)
AD-10 PROGRAM USING MACROFILES

```
!*************************************************************************
!
!  INITIALIZATION CODE (ONLY EXECUTED ONCE)
!
!  LOAD IOCC GROUP CODES 0 AND 1 IN COP GENERAL REGISTERS 0 AND 1
!
#CGR    .DEFINE 0,1,1,1,1,1,1,1
#IDAT   .DEFINE 0,1,1,1,1,1,1,1
        .INCLUDE DK1:[1,1]LOADC.8
!
!  SETUP A GROUP OF 3 ADC CHANNELS
!
#IOC    .DEFINE AIN,HIN,VIN,VIN,VIN,VIN,VIN,VIN
#CGR    .DEFINE 0,0,0,0,0,0,0,0
#OP     .EQU    0          ! SET GROUP CODE INTO I/O CHANNEL
        .INCLUDE DK1:[1,1]TRCE.8
!
!  SETUP A GROUP OF 7 DAC CHANNELS
!
#IOC    .DEFINE F11OUT,F12OUT,F13OUT,F14OUT,F15OUT,F8OUT,F9OUT,F9OUT
#CGR    .DEFINE 1,1,1,1,1,1,1,1
        .INCLUDE DK1:[1,1]TRCE.8
        .PAGE
!
!*************************************************************************
!
!  START OF FUNCTION GENERATION LOOP
!
!
!  DEFINE PROGRAM COUNTERS AT BEGINNING OF PROGRAM LOOP
!
        .ARP
ARP00   .EQU    *               !INITIAL ARP PC
        .DEP
DEP00   .EQU    *               !INITIAL DEP PC
        .MAP
MAP00   .EQU    *               !INITIAL MAP PC
!
COP00   PFI     INIT,0          ! INITIATE A CONVERSION (10 MS)
        RFR                     ! AND FORCE A MEMORY REFRESH
        PAUSE   63              ! WAIT FOR ADC'S TO CONVERT
!
!  READ CONVERTED VALUES AND TRANSFER THEM INTO MEMORY
!
#IOC    .DEFINE AIN,HIN,VIN,VIN,VIN,VIN,VIN,VIN
#DM     .DEFINE AMEM,HMEM,VMEM,DRAIN,DRAIN,DRAIN,DRAIN,DRAIN
#OP     .EQU    READ
        .INCLUDE DK1:[1,1]TREM.8
!
        HLT     0               ! HALT (MAYBE) TO LOOK AT OR MODIFY INPUTS
!
!  TRANSFER VALUES OF INPUT VARIABLES FROM MEMORY TO DEP "X" REGISTERS
!
#DXR    .DEFINE A1DEP,HDEP,VDEP,127,127,A2DEP,127,127
#DM     .DEFINE AMEM,HMEM,VMEM,FAUCET,FAUCET,AMEM,FAUCET,FAUCET
        .INCLUDE DK1:[1,1]TRMX.8
```

EXAMPLE (CONT.)
AD-10 PROGRAM USING MACROFILES

```
!
!  PERFORM BINARY SEARCH AND DELTA COMPUTATION ON H
!
#V      .DEFINE 127,127,HDEP,127,127,127
#NV     .DEFINE 32,32,NBPH-1,32,32,32          !UNUSED #NV'S SET TO 32
#DV     .DEFINE 127,127,DELH,127,127,127
#ORG    .EQU    BPORG
        .INCLUDE DK1:[1,1]BD.6
!
!  COMPUTE MACH NUMBER  M = 2 * V * F6(H)
!
        .COP
        START   $MAP
        PAUSE   3
        START   $ARP,$DEP
        PAUSE   4
        STOP    $ARP,$MAP,$DEP
!
        .ARP
        MOV0 S,B,D    ; MOV2 S,A ; MOV3 DELH,C
        FA (A-B)*C+D                           !INTERPOLATE FOR F6(H)
        MOV0 S,A      ; MOV3 R,C,F6
        FASL (A)*C    ; MOV3 F6,L              !COMPUTE M=2*V*F6(H)
        MOV3 R,L      ;  PAUSE 1               !DUMP M TO MULTIBUS
!
        .MAP
        RAID    AF6,HDEP ; PAUSE 7
        WUS     F6MEM                  !WRITE F6 SENT FROM ARP TO MEMORY
        WUS     MMEM     ;  PAUSE 1    !WRITE M SENT FROM ARP TO MEMORY
        .DEP
        PAUSE   0
        LXF     VDEP   ;  PAUSE 3      !SEND OUT V TO ARP
        SXS     MDEP                   !GRAB M SENT FROM ARP
!
!  DO BINARY SEARCH AND DELTA COMPUTATION FOR A1,M,H, AND A2 THIS TIME
!
#V      .DEFINE A1DEP,MDEP,HDEP,A2DEP,127,127
#NV     .DEFINE NBPA1-1,NBPM-1,NBPH-1,NBPA2-1,32,32  !UNUSED #NV'S = 32
#DV     .DEFINE DELA1,DELM,DELH,DELA2,127,127
#ORG    .EQU    BPORG
        .INCLUDE DK1:[1,1]BD.6
!
!  COMPUTE FUNCTION DATA POINTERS FOR THE TWO VARIABLE SETS:
!       IAMH  FOR  "A1,M,H"  SET
!       IMH   FOR  "M,H"  SET
!
#I      .DEFINE A1DEP,MDEP,127
#J      .DEFINE MDEP,HDEP,127
#K      .DEFINE HDEP,127,127
#NI     .DEFINE NBPA1,NBPM,0
#NJ     .DEFINE NBPM,0,0     ! SET #NJ1=0 TO GENERATE 2 VAR. POINTER
#IS     .DEFINE IAMH,IMH,127
        .INCLUDE DK1:[1,1]PT3.3
```

EXAMPLE (CONT.)
AD-10 PROGRAM USING MACROFILES

```
!
!  INTERPOLATE FOR 3 VARIABLE FUNCTIONS:
!  F11(A1,M,H), F12(A1,M,H), F13(A1,M,H)
!
#DX     .DEFINE DELA1,DELA1,DELA1
#DY     .DEFINE DELM,DELM,DELM
#DZ     .DEFINE DELH,DELH,DELH
#F      .DEFINE F11,F12,F13
#AF     .DEFINE AF11,AF12,AF13
#NX     .DEFINE NBPA1,NBPA1,NBPA1
#NY     .DEFINE NBPM,NBPM,NBPM
#IF     .DEFINE IAMH,IAMH,IAMH
        .INCLUDE DK1:[1,1]FI3.3
!
!  DO INTERPOLATION FOR 2 VARIABLE FUNCTIONS:
!  F14(M,H), F15(M,H)
!
#DX     .DEFINE DELM,DELM,127
#DY     .DEFINE DELH,DELH,127
#F      .DEFINE F14,F15,127
#NX     .DEFINE NBPM,NBPM,0
#AF     .DEFINE AF14,AF15,AF6     ! DUMMY UP ADDR AND INDEX FOR UNUSED
#IF     .DEFINE IMH,IMH,HDEP      ! INPUT TO AVOID ADDR ERRORS
        .INCLUDE DK1:[1,1]FI2.3
!
!  INTERPOLATE FOR 1 VARIABLE FUNCTIONS:
!  F8(A2), F9(A2)
!
#DX     .DEFINE DELA2,DELA2,127
#F      .DEFINE F8,F9,127
#AF     .DEFINE AF8,AF9,AF6       ! DUMMY UP ADDR AND INDEX FOR UNUSED
#IF     .DEFINE A2DEP,A2DEP,HDEP  ! INPUT TO AVOID ADDR ERRORS
        .INCLUDE DK1:[1,1]FI1.3
!
!  TRANSFER RESULTS FROM ARP REGISTERS TO DAC'S AND MEMORY
!
#ATR    .DEFINE F11,F12,F13,F14,F15,F8,F9,F9
#IOC    .DEFINE F11OUT,F12OUT,F13OUT,F14OUT,F15OUT,F8OUT,F9OUT,F9OUT
#DM     .DEFINE F11MEM,F12MEM,F13MEM,F14MEM,F15MEM,F8MEM,F9MEM,DRAIN
#OP     .EQU    WRITE
        .INCLUDE DK1:[1,1]TRAEM.8
!
        PFI     UPDATE,1          ! UPDATE GROUP 1 DAC'S SIMULTANEOUSLY
        LPC     $DEP,DEP00        ! RESET PC'S
        LPC     $ARP,ARP00
        LPC     $MAP,MAP00
        HLT     1                 ! HALT (MAYBE) TO LOOK AT RESULTS
        JMP     COP00             !  GO DO IT AGAIN
        .END
```

EXAMPLE (CONT.)
FORTRAN PROGRAM TO GENERATE DATA FILES

```
C  FUNDAT.FTN
C
      REAL CLAMH(21,12),S(12),Q(21,12)
      REAL M(21),CLAM(21),H(12),A1(16),A2(33)
C
      REAL BH(33),SH(32),GH(32)
      REAL BA1(33),SA1(32),GA1(32)
      REAL BA2(33),SA2(32),GA2(32)
      REAL BM(33),SM(32),GM(32)
C
      DATA RHO0,A0/0.002377,1116.4/
      DATA M/.0,.0,.2,.4,.6,.7,.8,.85,.9,.95,1.,1.05,1.10,1.15,1.2
     1,1.3,1.4,1.55,1.7,1.85,2.0/
      DATA CLAM/2.5,2.5,2.51,2.54,2.64,2.73,2.88,3.0,3.19,3.4,3.59
     1,3.6,3.48,3.36,3.25,3.05,2.87,2.625,2.41,2.21,2.02/
      DATA H/1.,1.,.8617,.7385,.6292,.5328,.4481,.3741,.3099,.2462
     1,.1936,.1522/
      DATA A1/-.5,-.2,-.15,-.1,-.05,0.,.05,.1,.15,.2,.25,.3,.35,.4
     1,.45,.5/
      DATA S/1.,1.,1.0176,1.0363,1.056,1.0768,1.0989,1.1225,1.1476
     1,1.1533,1.1533,1.1533/
      DATA F11MAX,F12MAX,F13MAX,F14MAX,F15MAX,F6MAX
     1/0.7265,155.67,45.324,2.518,453.24,2.0/
      DATA BH/22*-1.,0.,.1,.2,.3,.4,.5,.6,.7,.8,.9,1.0/
      APROX1=1.0-(2.**(-15))
      DO 10 I=1,33
      A2(I)=(I-17)/32.
10    CONTINUE
      DO 50 J=1,21
      DO 50 K=1,12
      CLAMH(J,K) = CLAM(J) * (1. - .1*H(K)*M(J)*M(J))
      Q(J,K) = .5*RHO0*A0*A0*H(K)*(M(J)/S(K))*(M(J)/S(K))
50    CONTINUE
      CALL ASSIGN(6,'TI:',3)
C
C  GENERATE DATA FILE FOR F11  (LIFT FUNCTION)
      CALL ASSIGN(1,'F11.DAT;1',9)
      DEFINE FILE 1(4032,2,U,NREC)
      NREC=1
      DO 100 K=1,12
      DO 100 J=1,21
      DO 100 I=1,16
      CL = A1(I)*CLAMH(J,K)
      IF (CL.LT.1.1) F1 = CL
      IF (CL.GE.1.1) F1 = CL - 2.*(CL-1.1)*(CL-1.1)
      IF (CL.LE.-1.1) F1 = CL + 2.*(CL+1.1)*(CL+1.1)
      F11 = .5*RHO0*A0*H(K)*(M(J)/S(K))*F1
      CALL MINMAX(NREC,RMIN,RMAX,F11)
      FS = F11/F11MAX
      IF (FS.GT.APROX1) FS=APROX1
      IF (FS.LT.-1.0) FS=-1.0
      WRITE(1'NREC) FS
100   CONTINUE
      CALL CLOSE(1)
      NFUN=11
```

```
      WRITE(6,9100) NFUN,RMIN,RMAX
9100  FORMAT(1X,'F',I2,'   MIN = 'G14.7,'  MAX = ',G14.7)
C
C  GENERATE DATA FILE FOR F12  (DRAG FUNCTION)
      CALL ASSIGN(1,'F12.DAT;1',9)
      DEFINE FILE 1(4032,2,U,NREC)
      NREC=1
      DO 200 K=1,12
      DO 200 J=1,21
      DO 200 I=1,16
      CL = A1(I)*CLAMH(J,K)
      F2 = (.007 + .05*CL*CL)*CLAM(J)
      F12 = Q(J,K)*F2
      CALL MINMAX(NREC,RMIN,RMAX,F12)
      FS = F12/F12MAX
      IF (FS.GT.APROX1) FS=APROX1
      IF (FS.LT.-1.0) FS=-1.0
      WRITE(1'NREC) FS
200   CONTINUE
      CALL CLOSE(1)
      NFUN=12
      WRITE(6,9100) NFUN,RMIN,RMAX
C
C  GENERATE DATA FILE FOR F13  (PITCHING MOMENT FUNCTION)
      CALL ASSIGN(1,'F13.DAT;1',9)
      DEFINE FILE 1(4032,2,U,NREC)
      NREC=1
      DO 300 K=1,12
      DO 300 J=1,21
      DO 300 I=1,16
      CL = A1(I)*CLAMH(J,K)
      F3 = -.1*(1.+.5*M(J))*CL
      F13 = Q(J,K)*F3
      CALL MINMAX(NREC,RMIN,RMAX,F13)
      FS = F13/F13MAX
      IF (FS.GT.APROX1) FS=APROX1
      IF (FS.LT.-1.0) FS=-1.0
      WRITE(1'NREC) FS
300   CONTINUE
      CALL CLOSE(1)
      NFUN=13
      WRITE(6,9100) NFUN,RMIN,RMAX
C
C  GENERATE DATA FILE FOR F14  (PITCH DAMPING FUNCTION)
      CALL ASSIGN(1,'F14.DAT;1',9)
      DEFINE FILE 1(336,2,U,NREC)
      NREC=1
      DO 400 K=1,12
      DO 400 J=1,21
      F14 = -(RHOO*AO/4.)*H(K)*(M(J)/S(K))*CLAM(J)
      CALL MINMAX(NREC,RMIN,RMAX,F14)
      FS = F14/F14MAX
      IF (FS.GT.APROX1) FS=APROX1
      IF (FS.LT.-1.0) FS=-1.0
      WRITE(1'NREC) FS
400   CONTINUE
```

EXAMPLE (CONT.)
FORTRAN PROGRAM TO GENERATE DATA FILES

```
      CALL CLOSE(1)
      NFUN=14
      WRITE(6,9100) NFUN,RMIN,RMAX
C
C  GENERATE DATA FILE FOR F15 (PITCH CONTROL FUNCTION)
      CALL ASSIGN(1,'F15.DAT;1',9)
      DEFINE FILE 1(336,2,U,NREC)
      NREC=1
      DO 500 K=1,12
      DO 500 J=1,21
      F15 = -.15*Q(J,K)*CLAMH(J,K)
      CALL MINMAX(NREC,RMIN,RMAX,F15)
      FS = F15/F15MAX
      IF (FS.GT.APROX1) FS=APROX1
      IF (FS.LT.-1.0) FS=-1.0
      WRITE(1'NREC) FS
500   CONTINUE
      CALL CLOSE(1)
      NFUN=15
      WRITE(6,9100) NFUN,RMIN,RMAX
C
C  GENERATE DATA FILE FOR F6  (INVERSE SPEED OF SOUND RATIO)
      CALL ASSIGN(1,'F6.DAT;1',8)
      DEFINE FILE 1(12,2,U,NREC)
      NREC=1
      DO 600 K=1,12
      F6 = .8957*S(K)
      CALL MINMAX(NREC,RMIN,RMAX,F6)
      FS = F6/F6MAX
      WRITE(1'NREC) FS
600   CONTINUE
      CALL CLOSE(1)
      NFUN=6
      WRITE(6,9100) NFUN,RMIN,RMAX
C
C  GENERATE DATA FILE FOR F8 ( COS(ALPHA) )
      CALL ASSIGN(1,'F8.DAT;1',8)
      DEFINE FILE 1(33,2,U,NREC)
      NREC=1
      DO 800 I=1,33
      FS = COS(A2(I))
      CALL MINMAX(NREC,RMIN,RMAX,FS)
      IF (FS.GT.APROX1) FS=APROX1
      IF (FS.LT.-1.0) FS=-1.0
      WRITE(1'NREC) FS
800   CONTINUE
      CALL CLOSE(1)
      NFUN=8
      WRITE(6,9100) NFUN,RMIN,RMAX
```

EXAMPLE (CONT.)

FORTRAN PROGRAM TO GENERATE DATA FILES

```
C
C  GENERATE DATA FILE FOR F9  ( SIN(ALPHA) )
      CALL ASSIGN(1,'F9.DAT;1',8)
      DEFINE FILE 1(33,2,U,NREC)
      NREC=1
      DO 900 I=1,33
      FS = SIN(A2(I))
      CALL MINMAX(NREC,RMIN,RMAX,FS)
      IF (FS.GT.APROX1) FS=APROX1
      IF (FS.LT.-1.0) FS=-1.0
      WRITE(1'NREC) FS
900   CONTINUE
      CALL CLOSE(1)
      NFUN=9
      WRITE(6,9100) NFUN,RMIN,RMAX
C
C  NOW FOR THE BREAKPOINT DATA FILES
      DO 1000 I=1,33
      BA1(I) = -1.0
      IF (I.GT.17) BA1(I)=2.*A1(I-17)
      BA2(I)=2.*A2(I)
      BM(I) = -1.0
      IF (I.GT.13) BM(I)=M(I-12)/2.0
1000  CONTINUE
C
      CALL ASSIGN(1,'BH.DAT;1',8)
      CALL ASSIGN(2,'SH.DAT;1',8)
      CALL ASSIGN(3,'GH.DAT;1',8)
      CALL BSGOUT(BH,12)
      CALL ASSIGN(1,'BA1.DAT;1',9)
      CALL ASSIGN(2,'SA1.DAT;1',9)
      CALL ASSIGN(3,'GA1.DAT;1',9)
      CALL BSGOUT(BA1,16)
      CALL ASSIGN(1,'BA2.DAT;1',9)
      CALL ASSIGN(2,'SA2.DAT;1',9)
      CALL ASSIGN(3,'GA2.DAT;1',9)
      CALL BSGOUT(BA2,33)
      CALL ASSIGN(1,'BM.DAT;1',8)
      CALL ASSIGN(2,'SM.DAT;1',8)
      CALL ASSIGN(3,'GM.DAT;1',8)
      CALL BSGOUT(BM,21)
      END
```

EXAMPLE (CONT.)
FORTRAN PROGRAM TO GENERATE DATA FILES

```
      SUBROUTINE MINMAX(NREC,RMIN,RMAX,FS)
      IF (NREC.EQ.1) RMIN=FS
      IF (NREC.EQ.1) RMAX=FS
      IF (FS.LT.RMIN) RMIN=FS
      IF (FS.GT.RMAX) RMAX=FS
      RETURN
      END



      SUBROUTINE BSGOUT(B,NBPS)
      DIMENSION B(33),S(32),G(32)
      DEFINE FILE 1(32,2,U,NREC)
      DEFINE FILE 2(32,2,U,NREC)
      DEFINE FILE 3(32,2,U,NREC)
      DO 100 I=1,32
      S(I)=0
100   G(I)=0
      ISTART=34-NBPS
      DO 200 I=ISTART,32
      DIFF = B(I+1)-B(I)
      S(I) = INT(.5/DIFF)+1
200   G(I) = .5/(S(I)*DIFF)
      DO 300 I=1,32
      WRITE(1'I)B(I)
      WRITE(2'I)S(I)
300   WRITE(3'I)G(I)
      CALL CLOSE(1)
      CALL CLOSE(2)
      CALL CLOSE(3)
      RETURN
      END
```

EXAMPLE (CONT.)
ADX COMMAND FILE TO LOAD AND RUN PROBLEM

```
;  BENCH.CMD
;
;  LOAD AD-10 PROGRAM:
;
LOAD BENCH.MOD
;
;  LOAD FUNCTION DATA FILES:
;
LOAD F11.DAT/AL:0:0/RS
LOAD F12.DAT/AL:2:0/RS
LOAD F13.DAT/AL:4:0/RS
LOAD F14.DAT/AL:0:3740/RS
LOAD F15.DAT/AL:2:3740/RS
LOAD F6.DAT/AL:4:3740/RS
LOAD F8.DAT/AL:0:4136/RS
LOAD F9.DAT/AL:2:4136/RS
;
;  LOAD BREAKPOINT DATA FILES:
;
LOAD BA1.DAT/UN:0:7640/RS, SA1.DAT/UN:1:7640/RI, GA1.DAT/UN:2:7640/RS
LOAD BM.DAT/UN:3:7640/RS,  SM.DAT/UN:4:7640/RI,  GM.DAT/UN:5:7640/RS
LOAD SH.DAT/UN:0:7700/RI,  BH.DAT/UN:1:7700/RS,  GH.DAT/UN:2:7700/RS
LOAD SA2.DAT/UN:3:7700/RI, BA2.DAT/UN:4:7700/RS, GA2.DAT/UN:5:7700/RS
;
;  INITIALIZE AND START THE AD-10:
;
INIT
CONTINUE
```

SUMMARY OF MACROFILE
INPUTS, OUTPUTS, INSTRUCTION COUNTS, AND EXECUTION TIMES

TITLE	INPUTS	OUTPUTS	INSTRUCTIONS COP	ARP	DEP	MAP	EXECUTION TIME (MICRO-SEC)
BD.6	#V0-#V5 #NV0-#NV5 #ORG T0-T2	#V0-#V5 #DV0-DV5	13	22	43	49	6.0
SD.6	#V0-#V5 #NBPS T0-T2	#V0-#V5 #DV0-#DV5	6	22	13	0	2.3
PT2.3	#I0,#J0 #I1,#J1 #I2,#J2 #NI0-#NI2 T0-T2	#IS0-#IS2	4	8	8	0	1.1
PT3.3	#I0,#J0,#K0 #I1,#J1,#K1 #I2,#J2,#K2 T0-T2	#IS0-#IS2	10	11	11	0	1.4
PT4.3	#I0,#J0,#K0,#L0 #I1,#J1,#K1,#L1 #I2,#J2,#K2,#L2 #NI0,#NJ0,#NK0 #NI1,#NJ1,#NK1 #NI2,#NJ2,#NK2 T0,T1	#IS0-#IS2	13	14	14	0	1.7
PT5.3	#I0,#J0,#K0,#L0,#M0 #I1,#J1,#K1,#L1,#M1 #I2,#J2,#K2,#L2,#M2 #NI0,#NJ0,#NK0,#NL0 #NI1,#NJ1,#NK1,#NL1 #NI2,#NJ2,#NK2,#NL2 T0-T1	#IS0-#IS2	16	17	17	0	2.0
FI1.3	#DX0-#DX2 #AF0-#AF2 #IF0-#IF2	#F0-#F2	3	6	0	3	1.1
FI2.3	#DX0,#DY0 #DX1,#DY1 #DX2,#DY2 #NX0-#NX2 #AF0-#AF2 #IF0-#IF2 T0-T2	#F0-#F2	5	16	0	6	2.1

TITLE	INPUTS	OUTPUTS	INSTRUCTIONS COP	ARP	DEP	MAP	EXECUTION TIME (MICRO-SEC)
FI3.3	#DX0,#DY0,#DZ0 #DX1,#DY1,#DZ1 #DX2,#DY2,#DZ2 #NX0,#NY0 #NX1,#NY1 #NX2,#NY2 #AF0-#AF2 #IF0-#IF2 T0-T5	#F0-#F2	5	31	0	12	3.6
FI4.3	#DV0,#DW0,#DX0,#DY0 #DV1,#DW1,#DX1,#DY1 #DV2,#DW2,#DX2,#DY2 #NV0,#NW0,#NX0 #NV1,#NW1,#NX1 #NV2,#NW2,#NX2 #AF0-#AF2 #IF0-#IF2 T0-T8	#F0-#F2	5	58	0	24	6.3
FI5.3	#DV0,#DW0,#DX0,#DY0,#DZ0 #DV1,#DW1,#DX1,#DY1,#DZ1 #DV2,#DW2,#DX2,#DY2,#DZ2 #NV0,#NW0,#NX0,#NY0 #NV1,#NW1,#NX1,#NY1 #NV2,#NW2,#NX2,#NY2 #AF0-#AF2 #IF0-#IF2 T0-T11	#F0-#F2	5	109	0	48	11.4
TRMA.8	#DM0-#DM7	#ATR0-#ATR7	3	9	0	8	1.4
TRMC.8	#DM0-#DM7	#CGR0-#CGR7	10	0	0	8	1.4
TRMX.8	#DM0-#DM7	#DXR0-#DXR7	3	0	9	8	1.4
TRMI.8	#DM0-#DM7	#DIR0-#DIR7	3	0	9	8	1.4
TRME.8	#DM0-#DM7 #OP	#IOC0-#IOC7	12	0	0	8	1.8
TRAM.8	#ATR0-#ATR7	#DM0-#DM7	3	8	0	8	1.0
TRCM.8	#CGR0-#CGR7	#DM0-#DM7	10	0	0	8	1.0
TRXM.8	#DXR0-#DXR7	#DM0-#DM7	3	0	8	8	1.0
TRIM.8	#DIR0-#DIR7	#DM0-#DM7	3	0	8	8	1.0
TREM.8	#IOC0-#IOC7 #OP	#DM0-#DM7	12	0	0	8	1.8

TITLE	INPUTS	OUTPUTS	INSTRUCTIONS COP	ARP	DEP	MAP	EXECUTION TIME (MICRO-SEC)
TRCA.8	#CGR0-#CGR7	#ATR0-#ATR7	9	8	0	0	.9
TRCX.8	#CGR0-#CGR7	#DXR0-#DXR7	9	0	8	0	.9
TRCI.8	#CGR0-#CGR7	#DIR0-#DIR7	9	0	8	0	.9
TRCE.8	#CGR0-#CGR7 #OP	#IOC0-#IOC7	8	0	0	0	.8
TRAC.8	#ATR0-#ATR7	#CGR0-#CGR7	10	8	0	0	1.0
TRXC.8	#DXR0-#DXR7	#CGR0-#CGR7	10	0	8	0	1.0
TRIC.8	#DIR0-#DIR7	#CGR0-#CGR7	10	0	8	0	1.0
TREC.8	#CGR0-#CGR7 #OP	#IOC0-#IOC7	13	0	0	0	1.3
TREXM.8	#IOC0-#IOC7 #OP	#DXR0-#DXR7 #DM0-#DM7	10	0	8	9	1.8
TRAEM.8	#ATR0-#ATR7 #OP	#IOC0-#IOC7 #DM0-#DM7	10	8	0	8	1.0
LOADA.8	#IDAT0-#IDAT7	#ATR0-#ATR7	10	9	0	0	1.0
LOADC.8	#IDAT0-#IDAT7	#CGR0-#CGR7	10	0	8	0	1.0
LOADX.8	#IDAT0-#IDAT7	#DXR0-#DXR7	10	0	9	0	1.0
LOADI.8	#IDAT0-#IDAT7	#DIR0-#DIR7	10	0	9	0	1.0
LOADM.8	#IDAT0-#IDAT7	#DM0-#DM7	10	0	0	9	1.0
SGN.2	#X,#Y	#SGNX,#SGNY	5	7	0	0	.8
CTR.3	#X0,#Y0,#A0 #X1,#Y1,#A1 #X2,#Y2,#A2 #SIN,#COS T0,T1,T2 I0,I1,I2	#XP0,#YP0 #SIN0,#COS0 #XP1,#YP1 #SIN1,#COS1 #XP2,#YP2 #SIN2,#COS2	13	31	3	6	3.2
IRS.3	#X0,#Y0 #X1,#Y1 #X2,#Y2	#A0 #SIN0,#COS0 #R0,#HR0 #A1 #SIN1,#COS1 #R1,#HR1 #A2 #SIN2,#COS2 #R2,#HR2	12	102	32	33	11.5