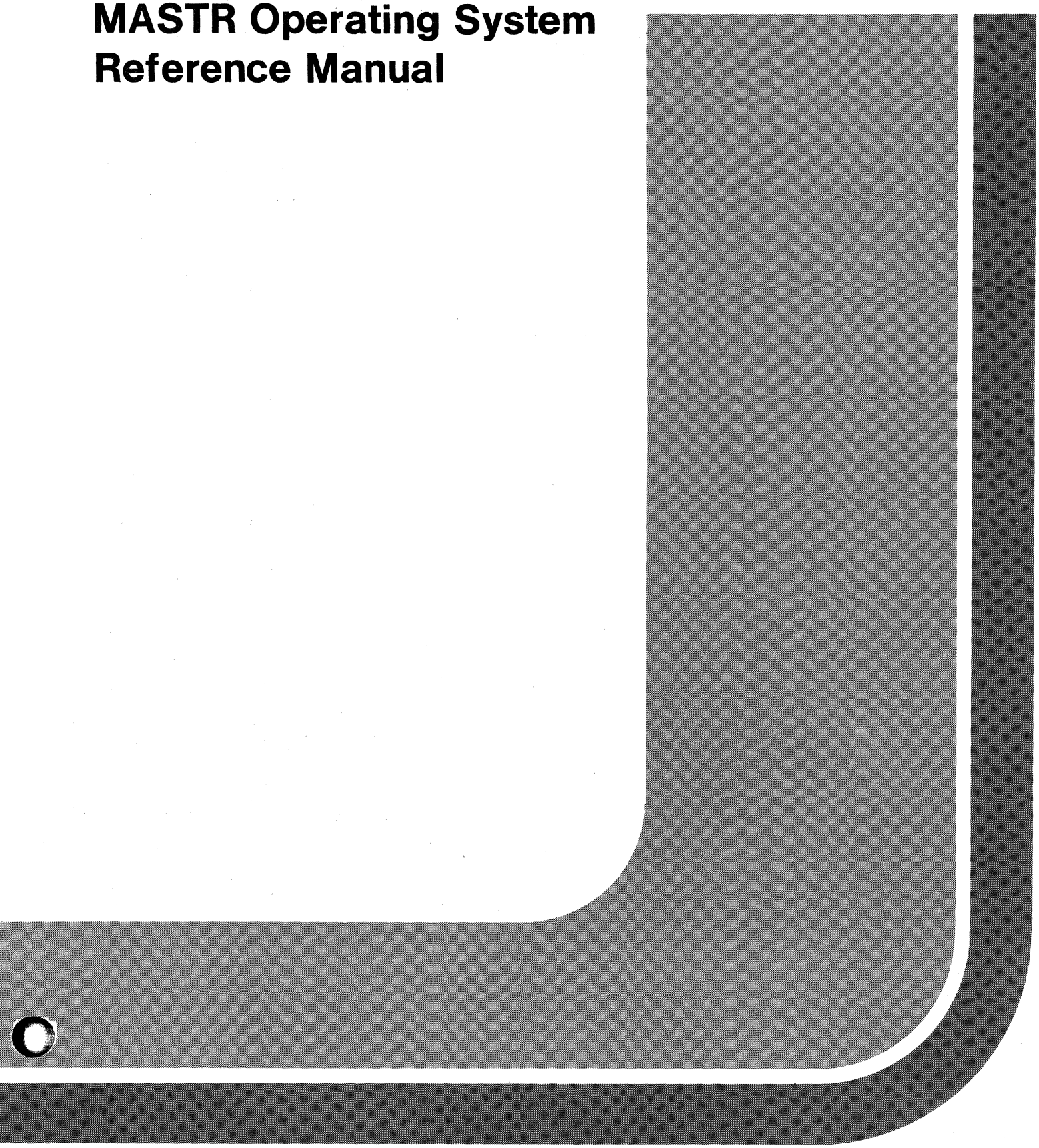
FAIRCHILD

Test Systems Group

MASTR Operating System Reference Manual



Publication Number 57518702

August 1979 Edition

This publication is subject to replacement by a later edition. This publication is also subject to technical updating by a Publications Bulletin. The issuance of publication improvements by either a later edition or a bulletin is decided partly on the basis of the degree of change required. To determine whether a later edition or bulletins exist for a particular publication, consult your local branch office, or FTSG Publications Department, 1725 Technology Drive, San Jose, CA 95110 (TWX 910-338-0558).

To request copies of this publication, contact:

Fairchild
Test Systems Group
Customer Services
M/S 36-07/57
1725 Technology Drive
San Jose, California 95110
(408) 998-0123 TWX 910-338-0558

A reader's feedback form is provided at the back of this publication. Help us to help you by providing comments. If the form has been removed, please send comments to FTSG Publications Department.

MASTR Operating System Reference Manual

Publication Number 57518702

August 1979 Edition

Preface

This manual provides reference data for executing the Sentry assembly language under the MASTR operating system (Rel. 2.1). The manual describes the conventions, system tables and system as used by the assembler.

Use of this manual assumes a previous understanding of the Sentry assembly language and its application to the Sentry test systems. Users who intend to implement assembler programs to augment FACTOR language capabilities must also have a detailed knowledge of the Sentry operating system characteristics.

The role of this manual is to provide the Sentry system programmer with the details necessary to implement user-written assembly language overlays under the MASTR operating system. Appendix C provides supplementary information on the conversion of TOPSY/DOPSY user overlays to MASTR.

The following manuals are suggested for reference and supplemental reading:

Description	Publication Number
FST-2 Computer Manual	57000002
Sentry VII Users Manual	57000013
MASTR Command Language Reference Manual	57518701
MASTR FACTOR Manual	57518700

Caution

The description of the MASTR system in this manual is provided for reference purposes only. Any alteration of the operating system may cause disastrous effects, including damage or destruction of the tester hardware. The direct use of system tables is inadvisable because the table may be changed without notice. Use of any routines noted as reserved for system use only may result in undefined operations. Any modification of the system software is performed at the sole risk of the user. Fairchild Test Systems Group will provide no support for modified software systems. Fairchild Test Systems Group reserves all rights to the software described in this manual. Contents and descriptions are subject to change without notice.

Contents

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

CONTENTS

APPENDICES

TABLES

FIGURES

FIGURES

1

Introduction

1.1 OVERVIEW

The information within this manual describes the interface between user written overlays and the MASTR operating system. The information pertains to the interface only; details of assembly language commands and their use are defined in the FST-1 Assembler Manual (publication number 67094951) and the FST-2 Computer Manual (publication number 57000002).

An overlay is an assembly language program that is loaded into memory to perform a particular function. This manual describes the assembly language under MASTR as a reference for user written overlays. It contains detailed descriptions of the conventions, system tables, and system subroutines available with MASTR software.

Two types of user-written assembly language programs or files may be executed under MASTR. The first type of assembly language program is one which is called by a FACTOR test program. These are called Assembly Language Linkage files or ALLINK files.

The second type of assembly language program is one which is called by an operator command while no test station is active. These files are called user overlays. They are not to be confused with the system overlays, DATALOG for example, which are called into memory as the system demands. To be executed, system overlays must be resident on the disk or in memory.

1.2 MANUAL CONTENTS

This manual is divided into six sections, each one addressing a different facet of MASTR software.

1.2.1 Introduction

Section 1 is a general introduction to MASTR organization and gives a brief summary of the reference material that will be encountered.

1.2.2 System Global Data

Section 2 includes a listing of system global data. The system global data tables are divided into five categories:

Global Constants Tables. These tables contain the constants frequently used by the system and overlays. They are located at an absolute memory location in order to be referenced via an EQU to the address.

System Variables Tables. This table contain the system configuration. Variables are referenced via an EQU.

Global Variables Tables. System related items which are global in nature and not unique for each station. They are referenced directly via an EQU.

Tester Variables Tables. Contains data unique to each of up to four test heads. They are referenced via index register 2.

Current Station Variables Tables. Contains data unique to the station which is currently on line. They are referenced via index register 1.

Each listing contains the following information for the data words:

- Location in memory
- Octal representation of the data
- Symbolic label
- Assembler opcode
- Operand
- Comment field

The listings are followed by individual descriptions of the 24-bit data words contained in the table. Bit position information is provided where applicable.

1.2.3 System Subroutines

Section 3 provides information about the system subroutines, beginning with a listing of the system subroutine transfer vector. It describes those system procedures that are available to users by means of the CALL directive. Each subroutine is described along with the calling sequence needed to activate it under MASTR.

1.2.4 Input/Output Control System ($IOCS)

Section 4 describes the Input/Output Control System ($IOCS). MASTR software requires that all I/O functions must be accomplished through $IOCS to preserve foreground/back-ground configuration. This section provides the programmer with information concerning principles of operation, handling of devices, $IOCS functions, and the general calling sequence format. The I/O Assignment Table (IOATAB) is provided along with a detailed explanation of each data word. I/O operations available with MASTR are individually discussed, and the calling sequence for each is provided.

1.2.5 MASTR System Files

Section 5 contains information about the manner in which MASTR maintains system files. Physical and logical formats are described for disk, magnetic tape, Integrator and memory file storage. The system memory map (figure 4-12) shows the MASTR file organization in memory.

1.2.6 ALLINK Programs

Section 6 describes linking of Assembly Language overlays (ALLINK programs) under MASTR. It shows MASTR format for ALLINK program headers and the loading and calling procedures from foreground and background. Scheduling background tasks from foreground is explained. A sample ALLINK program is provided. Figure 6-1 gives a working example of the MASTR ALLINK file.

2

System Global Data

This section describes system globals and their use in assembly language programs.

2.1 GENERAL DESCRIPTION AND USAGE

There are three types of globals available: constants, variables, and subroutine or label addresses. All globals are located in the base page at absolute addresses. By defining the data name and EQUing to the global address, the user may access any global data.

2.1.1 Global Constant Usage

BITFLD at 102B	Binary constants with one bit set
NBTFLD at 132B	Binary constants with one bit not set
RTMSK at 162B	Binary constants with more than one bit set from bit 0
LFMSK at 211B	Binary constants with more than one bit set from bit 23
OCTFLD at 237B	Octal numbers
DECFLD at 237B	Decimal numbers
DBFLD at 404B	Binary constants with two or more bits set

The complete list of available constants is provided in figure 2-1.

Examples of usage:

```
BITFLD    EQU    102B
RTMSK     EQU    162B

B10       EQU    BITFLD+10          2000B
B23       EQU    BITFLD+23      40000000B
MINUS1    EQU    RTMSK+22       77777777B

          LDA    B10
          AND    MINUS1
```

2.1.2 Global Variable Usage

GLOVAR at 420B

Variables listed under GLOVAR are used to transfer data between different programs. Each data item has a particular meaning and must be used according to the individual definition. Detailed descriptions of variables are provided in figure 2-3.

Example of usage:

```
GLOVAR    EQU     420B
REVN      EQU     GLOVAR+23
          .
          .
          .
          LDA     REVN
```

2.1.3 System Global Variable Usage

SYSVAR at 64B

Variables listed under SYSVAR are used by MASTR to define system initialization conditions. These conditions in general are: default device for file loading and dumping; miscellaneous configuration flags; line printer type; miscellaneous disk addressing information for directory, file area, and working storage; and starting address of monitor after bootstrap.

Example of usage:

```
SYSVAR    EQU     64B
SELP      EQU     SYSVAR+2
          .
          .
          .
          LDA     SELP
```

2.1.4 Global Subroutine Usage

SYXVEC at 640B

The starting address of the transfer vectors is located at 640B. There are 63 global subroutines. Detailed descriptions of global subroutines are provided in section 3. The transfer vector is shown in figure 3-1.

The transfer vectors follow the SYXVEC address.

Example of usage:

```
SYXVEC      EQU     640B
FFIX        EQU     SYXVEC+57
$IOCS       EQU     SYXVEC+5
              .
              .
              .
            BSM*    FFIX
            BSM*    $IOCS
```

2.2 GLOBAL CONSTANTS

A listing of global constants appears in figure 2-1.

```
                        PAGE
                    * BIT FIELD DEFINITIONS
                    * BX IMPLIES BIT X IS SET
                    * NBX IMPLIES NOT BIT X IS SET
                    * BX$Y IMPLIES BIT X THRU Y ARE SET
                    *
                    * SINGLE BIT FIELDS
                    *
00101 00000001              DATA  1B          FOR PMU CONVERGENCE TABLE
      00000102      BITFLD  EQU   *
00102 00000001      B0      DATA  1B
00103 00000002      B1      DATA  2B
00104 00000004      B2      DATA  4B
00105 00000010      B3      DATA  10B
00106 00000020      B4      DATA  20B
00107 00000040      B5      DATA  40B
00110 00000100      B6      DATA  100B
00111 00000200      B7      DATA  200B
00112 00000400      B8      DATA  400B
00113 00001000      B9      DATA  1000B
00114 00002000      B10     DATA  2000B
00115 00004000      B11     DATA  4000B
00116 00010000      B12     DATA  10000B
00117 00020000      B13     DATA  20000B
00120 00040000      B14     DATA  40000B
00121 00100000      B15     DATA  100000B
00122 00200000      B16     DATA  200000B
00123 00400000      B17     DATA  400000B
00124 01000000      B18     DATA  1000000B
00125 02000000      B19     DATA  2000000B
00126 04000000      B20     DATA  4000000B
00127 10000000      B21     DATA  10000000B
00130 20000000      B22     DATA  20000000B
00131 40000000      B23     DATA  40000000B
                    *
                    * NOT BIT-TABLE (INVERSE OF BIT)
                    *
      00000132      NBTFLD  EQU   *
00132 77777776      NB0     DATA  777777768
00133 77777775      NB1     DATA  777777758
00134 77777773      NB2     DATA  777777738
00135 77777767      NB3     DATA  777777678
00136 77777757      NB4     DATA  777777578
00137 77777737      NB5     DATA  777777378
00140 77777677      NB6     DATA  777776778
00141 77777577      NB7     DATA  777775778
00142 77777377      NB8     DATA  777773778
00143 77776777      NB9     DATA  777767778
00144 77775777      NB10    DATA  777757778
00145 77773777      NB11    DATA  777737778
00146 77767777      NB12    DATA  777677778
00147 77757777      NB13    DATA  777577778
00150 77737777      NB14    DATA  777377778
00151 77677777      NB15    DATA  776777778
00152 77577777      NB16    DATA  775777778
00153 77377777      NB17    DATA  773777778
00154 76777777      NB18    DATA  767777778
00155 75777777      NB19    DATA  757777778
00156 73777777      NB20    DATA  737777778
00157 67777777      NB21    DATA  677777778
00160 57777777      NB22    DATA  577777778
00161 37777777      NB23    DATA  377777778
```

Figure 2-1 Global Constants Listing

```
                                PAGE
                          *
                          * ASSORTED MASKS
                          *
      00000162  RTMSK     EQU   *           N
                          *
00162 00000003  B1$0       DATA  3B          0
00163 00000007  B2$0       DATA  7B          1
00164 00000017  B3$0       DATA  17B         2
00165 00000037  B4$0       DATA  37B         3
00166 00000077  B5$0       DATA  77B         4
00167 00000177  B6$0       DATA  177B        5
00170 00000377  B7$0       DATA  377B        6
00171 00000777  B8$0       DATA  777B        7
00172 00001777  B9$0       DATA  1777B       8
00173 00003777  B10$0      DATA  3777B       9
00174 00007777  B11$0      DATA  7777B       10
00175 00017777  B12$0      DATA  17777B      11
00176 00037777  B13$0      DATA  37777B      12
00177 00077777  B14$0      DATA  77777B      13
00200 00177777  B15$0      DATA  177777B     14
00201 00377777  B16$0      DATA  377777B     15
00202 00777777  B17$0      DATA  777777B     16
00203 01777777  B18$0      DATA  1777777B    17
00204 03777777  B19$0      DATA  3777777B    18
00205 07777777  B20$0      DATA  7777777B    19
00206 17777777  B21$0      DATA  17777777B   20
00207 37777777  B22$0      DATA  37777777B   21
00210 77777777  B23$0      DATA  77777777B   22
                          *
      00000211  LFMSK     EQU   *           N
                          *                 *
00211 77777776  B23$1      DATA  77777776B   0
00212 77777774  B23$2      DATA  77777774B   1
00213 77777770  B23$3      DATA  77777770B   2
00214 77777760  B23$4      DATA  77777760B   3
00215 77777740  B23$5      DATA  77777740B   4
00216 77777700  B23$6      DATA  77777700B   5
00217 77777600  B23$7      DATA  77777600B   6
00220 77777400  B23$8      DATA  77777400B   7
00221 77777000  B23$9      DATA  77777000B   8
00222 77776000  B23$10     DATA  77776000B   9
00223 77774000  B23$11     DATA  77774000B   10
00224 77770000  B23$12     DATA  77770000B   11
00225 77760000  B23$13     DATA  77760000B   12
00226 77740000  B23$14     DATA  77740000B   13
00227 77700000  B23$15     DATA  77700000B   14
00230 77600000  B23$16     DATA  77600000B   15
00231 77400000  B23$17     DATA  77400000B   16
00232 77000000  B23$18     DATA  77000000B   17
00233 76000000  B23$19     DATA  76000000B   18
00234 74000000  B23$20     DATA  74000000B   19
00235 70000000  B23$21     DATA  70000000B   20
00236 60000000  B23$22     DATA  60000000B   21
```

Figure 2-1 Global Constants Listings (Continued)

```
                        PAGE
                *
                *  DECIMAL/OCTAL NUMBER
                *
      00000237  DECFLD   EQU   *          DECIMAL FIELD
      00000237  OCTFLD   EQU   *          OCTAL FIELD
                *
00237 00000000  D0        DATA  0          +0
00240 00000001  D1        DATA  1          +1
00241 00000002  D2        DATA  2          +2
00242 00000003  D3        DATA  3          +3
00243 00000004  D4        DATA  4          +4
00244 00000005  D5        DATA  5          +5
00245 00000006  D6        DATA  6          +6
00246 00000007  D7        DATA  7          +7
00247 00000010  D8        DATA  8          +8
00250 00000011  D9        DATA  9          +9
00251 00000012  D10       DATA  10         +10
00252 00000013  D11       DATA  11         +11
00253 00000014  D12       DATA  12         +12
00254 00000015  D13       DATA  13         +13
00255 00000016  D14       DATA  14         +14
00256 00000017  D15       DATA  15         +15
00257 00000020  D16       DATA  16         +16
00260 00000021  D17       DATA  17         +17
00261 00000022  D18       DATA  18         +18
00262 00000023            DATA  19         +19
00263 00000024  D20       DATA  20         +20
00264 00000025  D21       DATA  21         +21
00265 00000026  D22       DATA  22         +22
00266 00000027  D23       DATA  23         +23
00267 00000030            DATA  24         +24
00270 00000031  D25       DATA  25         +25
00271 00000032            DATA  26         +26
00272 00000033            DATA  27         +27
00273 00000034  D28       DATA  28         +28
00274 00000035            DATA  29         +29
00275 00000036            DATA  30         +30
00276 00000037  D31       DATA  31         +31
00277 00000040  D32       DATA  32         +32
00300 00000041  D33       DATA  33         +33
00301 00000042  D34       DATA  34         +34
00302 00000043            DATA  35         +35
00303 00000044            DATA  36         +36
00304 00000045  D37       DATA  37         +37
00305 00000046            DATA  38         +38
00306 00000047            DATA  39         +39
00307 00000050  D40       DATA  40         +40
00310 00000051  D41       DATA  41         +41
00311 00000052  D42       DATA  42         +42
00312 00000053  D43       DATA  43         +43
00313 00000054  D44       DATA  44         +44
00314 00000055  D45       DATA  45         +45
00315 00000056            DATA  46         +46
00316 00000057  D47       DATA  47         +47
00317 00000060  D48       DATA  48         +48
00320 00000061  D49       DATA  49         +49
00321 00000062            DATA  50         +50
00322 00000063            DATA  51         +51
00323 00000064            DATA  52         +52
00324 00000065            DATA  53         +53
00325 00000066            DATA  54         +54
00326 00000067            DATA  55         +55
```

Figure 2-1 Global Constants Listings (Continued)

```
00327 00000070           DATA  56          +56
00330 00000071           DATA  57          +57
00331 00000072           DATA  58          +58
00332 00000073           DATA  59          +59
00333 00000074  D60      DATA  60          +60
00334 00000075           DATA  61          +61
00335 00000076  D62      DATA  62          +62
00336 00000077  D63      DATA  63          +63
00337 00000100  D64      DATA  64          +64
00340 00000101  D65      DATA  65          +65
00341 00000102           DATA  66          +66
00342 00000103  D67      DATA  67          +67
00343 00000104           DATA  68          +68
00344 00000105           DATA  69          +69
00345 00000106           DATA  70          +70
00346 00000107           DATA  71          +71
00347 00000110           DATA  72          +72
00350 00000111  D73      DATA  73          +73
00351 00000112  D74      DATA  74          +74
00352 00000113           DATA  75          +75
00353 00000114           DATA  76          +76
00354 00000115           DATA  77          +77
00355 00000116           DATA  78          +78
00356 00000117           DATA  79          +79
00357 00000120  D80      DATA  80          +80
00360 00000121           DATA  81          +81
00361 00000122           DATA  82          +82
00362 00000123           DATA  83          +83
00363 00000124           DATA  84          +84
00364 00000125  D85      DATA  85          +85
00365 00000126           DATA  86          +86
00366 00000127           DATA  87          +87
00367 00000130           DATA  88          +88
00370 00000131           DATA  89          +89
00371 00000132           DATA  90          +90
00372 00000133           DATA  91          +91
00373 00000134  D92      DATA  92          +92
00374 00000135           DATA  93          +93
00375 00000136           DATA  94          +94
00376 00000137           DATA  95          +95
00377 00000140           DATA  96          +96
00400 00000141           DATA  97          +97
00401 00000142           DATA  98          +98
00402 00000143           DATA  99          +99
00403 00000144           DATA  100         +100
                 *
      00000113  D512     EQU   BITFLD+9
      00000210  DM1      EQU   RTMSK+22
      00000211  DM2      EQU   LFMSK+0
      00000212  DM4      EQU   LFMSK+1


                         PAGE
                 *
                 *
                 * DOUBLE BIT FIELDS
                 *
      00000404  DBFLD    EQU   *             N
00404 00006000  B11$10   DATA  00006000B     0
00405 00014000  B12$11   DATA  00014000B     1
00406 37000000  B22$18   DATA  37000000B     2
00407 00000360  B7$4     DATA  00000360B     3
00410 00000300  B7$6     DATA  00000300B     4
      00000255  B3$1     EQU   D14
00411 00001700  D960     DATA  960         5
```

Figure 2-1 Global Constants Listings (Continued)

2.3 SYSTEM GLOBAL VARIABLES

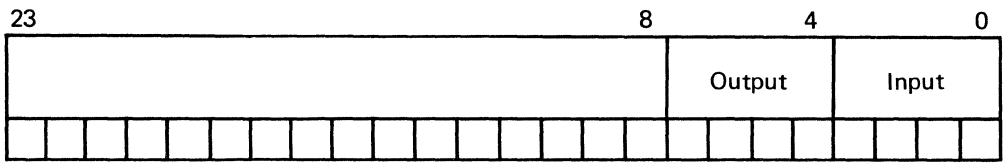
A listing of system variables appears in figure 2-2.

```
                        PAGE
      00000064          ORG   64B
                *
                *  GLOBAL VARIABLE FOR USE BY THE SYSTEM
                *
      00000064  SYSVAR    EQU   *          N
00064 00000210  DFDV      DATA  210B       0  DEFAULT DEVICE FOR FILE LOAD/DUMP
00065 00000000  M1INIT    DATA  0          1  SYSTEM INITIALIZED FLAG
00066 00000002  SFLP      DATA  2          2  LP TYPE, DEFAULT=PRINTRONIX
00067 00000000  M1WSWC    DATA  0          3  # OF WORD IN WS
00070 00000000  M1WSSC    DATA  0          4  # OF SECTORS IN WS
00071 00000000  M1WSDA    DATA  0          5  START SECTOR OF WS
00072 00000000  M1FDDA    DATA  0          6  START SECTOR OF DIRECTORY
00073 00000000  M1FDA     DATA  0          7  START SECTOR OF FILE AREA
00074 00000000            DATA  0          8   SPARE
00075 00001753            DATA  SYSINT     9 ENTRY POINT
00076 00000000            DATA  0          10  SPARE
                *
      00000100            ORG   100B
00100 01002107            BRU   MPRO       RESTART FROM 100B
```

Figure 2-2 System Global Variables Listing

2.3.1 DFDV (SYSVAR + 0)

Contains the default device code for file loading and dumping. For a disk-based system it is set to disk by $MASTR. Otherwise, it is set to magnetic tape unit 1.



output	5	disk system
	8	memory system
input	5	disk system
	8	memory system

2.3.2 M1INIT (SYSVAR + 1)

Contains the following system configuration flags:

Bit	Function
23	0 first execution 1 it has been executed before (set by system initializer $MASTR and never cleared until it is recreated)
19	Flag to subroutine PUTA 0 address output in decimal 1 address output in octal
18	0 1V/1mV 1 2V/2mV
17	0 Sentry V, VII 1 Sentry VIII
16	0 4 range PMU 1 6 range PMU

2.3.3 SELP (SYSVAR + 2)

Contains the line printer type indicator:

0	Data Products/Printronix 80 column
1	Data Products 132 column
2	Centronics/Printronix

If the system is called from DOPSY, SELP is set to the type used by DOPSY. MASTR default is 2.

2.3.4 M1WSWC (SYSVAR + 3)

Contains the number of words available in disk working storage and is set when the system is loaded from DOPSY.

2.3.5 M1WSSC (SYSVAR + 4)

Contains the number of sectors available in disk working storage.

2.3.6 M1WSDA (SYSVAR + 5)

Contains the starting address of disk working storage in binary sector format.

2.3.7 M1FDDA (SYSVAR + 6)

Contains the starting address of disk directory in binary sector format.

2.3.8 M1FDA (SYSVAR + 7)

Contains the starting address of disk file area in binary sector format.

2.3.9 SYSINT (SYSVAR + 9)

Contains the starting address of monitor upon bootstrap.

2.4 GLOBAL VARIABLES

A listing of global variables appears in figure 2-3.

```
                        PAGE
      00000420          ORG   420B
                *
                * GLOBAL VARIABLES FOR USE BY OVERLAYS
                *
      00000220  NGLOV   EQU   144        # OF WORDS IN GLOVAR
                *
      00000420  GLOVAR  EQU   *          N
                *
00420 00000000  ATPA    DATA  0          0  STAT1 TEST PLAN ATTACHED TO STATION
00421 00000000          DATA  0          1  STAT2 + = MACTAB POINTER FOR TP
00422 00000000          DATA  0          2  STAT3 0 = NONE ATTACHED
00423 00000000          DATA  0          3  STAT4 - = BEING EDITED
00424 21211721  RELDAT  TEXT  '11/12/78' 4,5  RELEASE DATE (SYSREL HAS RELEASE #)
00425 22172730
00426 00000000  SITEQN  DATA  0          6  STATION ON LINE (0-6)
00427 00000000  APMREV  DATA  0          REV OF APM SOFTWARE
00430 00000222  NTVT    DATA  TVTL       8  # OF VARIABLES/STATION IN TVT TABLE
00431 00000000  TVT     DATA  0          9  TVT TABLE ADDRESS
00432 00000062  NSVT    DATA  SVTL       10 # OF VARIABLES IN SVT TABLE
00433 00000000  SVT     DATA  0          11 SVT TABLE ADDRESS
00434 00000010  NMAC    DATA  MACEL      12 # OF WORDS/ENTRY IN MACTAB
00435 00000000  FWMAC   DATA  0          13 MACTAB ADDRESS
00436 00000000  LWMAC   DATA  0          14 LAST USED ADDRESS+1 MACTAB
00437 00000000  STAVKT  DATA  0          15 DEFAULT VKT FOR ALL STATIONS
00440 00000000  PIDPMF  DATA  0          16 PID ENTRY ADDRESS IN IOATAB
00441 00000000  PODPMF  DATA  0          17 POD ENTRY ADDRESS IN IOATAB
00442 00000000  DRPMF   DATA  0          18 DISC DIRECT ENTRY ADDRSS IN IOATAB
00443 00000000  FWALT   DATA  0          19 FIRST ADDRESS OF ALTER BUFFER
00444 00000000  LWALT   DATA  0          20 LAST USED ADDRESS +1 OF ALTBUF
00445 00000000  FWIOA   DATA  0          21 FIRST ADDRESS OF IOATAB
00446 00000015  NIOA    DATA  IOAEL      22 # OF WORDS/ENTRY IN IOATAB
00447 00000000  CURSYS  DATA  0          23 CURRENT SYSTEM, 0=BG, 1=FG
00450 00000000  FGBGFL  DATA  0          24 1=BG WAIT FOR FG,2=FG WAIT FOR BG
00451 00000001  REVN    DATA  REV        25 CURRENT REV NUMBER
00452 77777777  JOB     DATA  -1         26 CURRENT JOB NUMBER
00453 00000022  TPHL    DATA  THL        27 TEST PLAN HEADER LENGTH
00454 00000024  OPHL    DATA  OHL        28 OVERLAY HEADER LENGTH
00455 00000000  DATE    DATA  0,0        29 CURRENT DATE
00456 00000000
00457 00000000  TIME    DATA  0          31 CURRENT TIME IN SECONDS
00460 00000000  PGPMF   DATA  0          32 PAGE PMF POINTER
00461 00000000  LWCPU   DATA  0          33 CPU LAST AVAILABLE WORD
00462 00000000  LWSYS   DATA  0          34 LAST SYSTEM RESERVED WORD +1
00463 00000000  LWAM    DATA  0          35 LAST AVAILABLE WORD TO TP, OVERLAY
00464 00000000  FWAM    DATA  0          36 FIRST AVAILABLE WORD TO TP, OVERLAY
00465 00000000  ADJFLG  DATA  0          37 ADJMEM WAITING FOR MEMBSY
00466 00000000  THDACT  DATA  0          38 FG ON/OFF FLAG
00467 00000000  PIDFLG  DATA  0          39 COMMAND ALREADY IN BUFFER FLAG
00470 00000000  ECHFLG  DATA  0          40 ECHO FLAG FOR PROCESS
00471 00000000  COMIMG  DATA  0          41 COMMAND IMAGE FROM PROCESS
00472 00000000  CMDPMF  DATA  0,CMDBUF,18 42 PMF FOR SYSTEM COMMAND
00473 00001503
00474 00000022
00475 00000000  OCTAL   DATA  0          45 OCTAL VALUE OF INTSCN
00476 00000000  OFLERR  DATA  0          46 DECIMAL APPEARED IN INTSCN
00477 00000000  LDFLG   DATA  0          47 LOAD IS BUSY FLAG
00500 00000000  MANTISSA DATA 0          48 INTEGER VALUE FROM NUMBER
00501 00000000  CMDV    DATA  0          49 DEVICE CODE FROM PROCESS
00502 00000000  NAMEM1  DATA  0          50 1ST STRING FROM PROCESS
00503 00000000  NAMEM2  DATA  0          51
00504 00000000  BINUM   DATA  0          52 BINARY VALUE FROM INTSCN/NUMBER
00505 00000000  BINC    DATA  0          53 BINARY COUNT FORM INTSCN/NUMBER
```

Figure 2-3 Global Variables Listings

```
00506 00000000  COLFLG   DATA  0          54 COLUMN FORMAT FLAG FOR PUTE
00507 00002261  RSTIO    DATA  RSTIOF     55 RESET PENDING SCHEDULER FLAG
00510 00000000  ACTFIO   DATA  0          56 FGIO IS ACTIVE OR PENDING
00511 00000000  MEMBSY   DATA  0          57 MEMORY BUSY
00512 00000000  NAMEM5   DATA  0          58 3RD NAME FROM PROCESS
00513 00000000  NAMEM6   DATA  0          59 3RD NAME FROM PROCESS
00514 00000000  NAMEM3   DATA  0          60 2ND STRING FROM PROCESS
00515 00000000  NAMEM4   DATA  0          61
00516 00000000  ONUMB1   DATA  0          62 OCTAL NUMBER 1 FROM PROCESS
00517 00000000  ONUMB2   DATA  0          63 OCTAL NUMBER 2 FROM PROCESS
00520 00000000  NUMB1    DATA  0          64 1ST DECIMAL # FROM PROCESS IN F.P.
00521 00000000  NUMB2    DATA  0          65 2ND DECIMAL # FROM PROCESS IN F.P.
00522 00000000  STATC    DATA  0          66 VALUE OF STAT N FROM PROCESS
00523 00000000  SPNUM1   DATA  0          67 SPECIAL # FROM PROCESS
00524 00000000  SPNUM2   DATA  0          68 SPECIAL # FROM PROCESS
00525 00000000  SPNUM3   DATA  0          69 SPECIAL # FROM PROCESS
00526 00000000  SPNUM4   DATA  0          70 SPECIAL # FROM PROCESS
00527 00000000  SPNUM5   DATA  0          71 SPECIAL # FROM PROCESS
00530 00000000  SPNUM6   DATA  0          72 SPECIAL # FROM PROCESS
00531 00000000  BINARY   DATA  0          73 BINARY VALUE FROM PROCESS
00532 00000000  INUMB1   DATA  0          74 1ST INTEGER FROM PROCESS
00533 00000000  INUMB2   DATA  0          75 2ND INTEGER FROM PROCESS
00534 00000000  BFLERR   DATA  0          76 NOT BINARY FLAG FROM INTSCN
00535 00000000  SPOPT    DATA  0          77 SPECIAL OPTION FLAG FROM PROCESS
00536 00000000  BINCNT   DATA  0          78 BINARY DIGIT COUNT FROM PROCESS
00537 00002263  AWATF    DATA  WATSPD     79 ADDRESS OF WAIT FLAG SCHEDULER
00540 00002264  ATHDF    DATA  THDFLG     80 ADDRESS OF THDFLAG SCHEDULER
00541 00000000  NUMFLG   DATA  0,0        81 NUMBER APPEARED FLAG FROM IDTSCN
00542 00000000
00543 00000000  NAME1    DATA  0          83 1ST NAME FROM IDTSCN
00544 00000000  NAME2    DATA  0          84 2ND NAME FROM IDTSCN
00545 00000000           DATA  0          85  SPARE
00546 00000000  RAIDRR   DATA  0          86 RAID BREAKEE RR
00547 00001561  DBUGSA   DATA  DBUGS      87 ADDR OF DEBUG ADDR HALT ROUTINE
00550 00000000  DFSTAT   DATA  0          88 DEFAULT STATION ID
00551 00000000  SMAFLG   DATA  0          89 MA STATION CONTROL
00552 00000000  SPDA     DATA  0          90 PD BUFFER BUSY FLAG
00553 00017725  RAIDER   DATA  RAIDEM     91 ADDR OF RAID PG 0 LOGIC
00554 00000100  RAIDBK   DATA  100B       92 SAVE RAID'S RR HERE
00555 00002301  AFGBGF   DATA  FGBGSC     93 FGBGH SCHEDULER FLAG
00556 22162100  SYSREL   DATA  '2.1'      94 SYSTEM REL # IN ASCII
00557 00000000  MASTAT   DATA  0          95 MA STATION FOR CR REQUEST
00560 00000000  CLIOID   DATA  0,0        96 CLIO NAME1,2
00561 00000000
00562 00000000  LOTNUM   DATA  0,0,0      98 LOT # FOR CLIO
00563 00000000
00564 00000000
00565 00000000  DEVNUM   DATA  0,0        101 DEVICE # FOR CLIO
00566 00000000
00567 00000000  CATGRY   DATA  0,0,0      103 CATEGORY FOR CLIO
00570 00000000
00571 00000000
00572 00000000  RSTTSC   DATA  0          106 STSC REG IMAGE
00573 00000000  BGID     DATA  0,0        107 BACKGROUND ID
00574 00000000
```

Figure 2-3 Global Variables Listing (Continued)

2.4.1 ATPA (GLOVAR + 0, 1, 2, 3)

Contains the address of MACTAB for the test program attached to the station.

ATPA + 0 station 1
ATPA + 1 station 2
ATPA + 2 station 3
ATPA + 3 station 4

Contains zero if a test program is not attached.

2.4.2 RELDAT (GLOVAR + 4,5)

Contains the release date of the operating system.

2.4.3 SITEQQ (GLOVAR + 6)

Contains the station number currently online.

0 station 1
1 station 2
2 station 3
3 station 4

2.4.4 APMREV (GLOVAR + 7)

Contains the APM F8 operating system revision number in TRASCII.

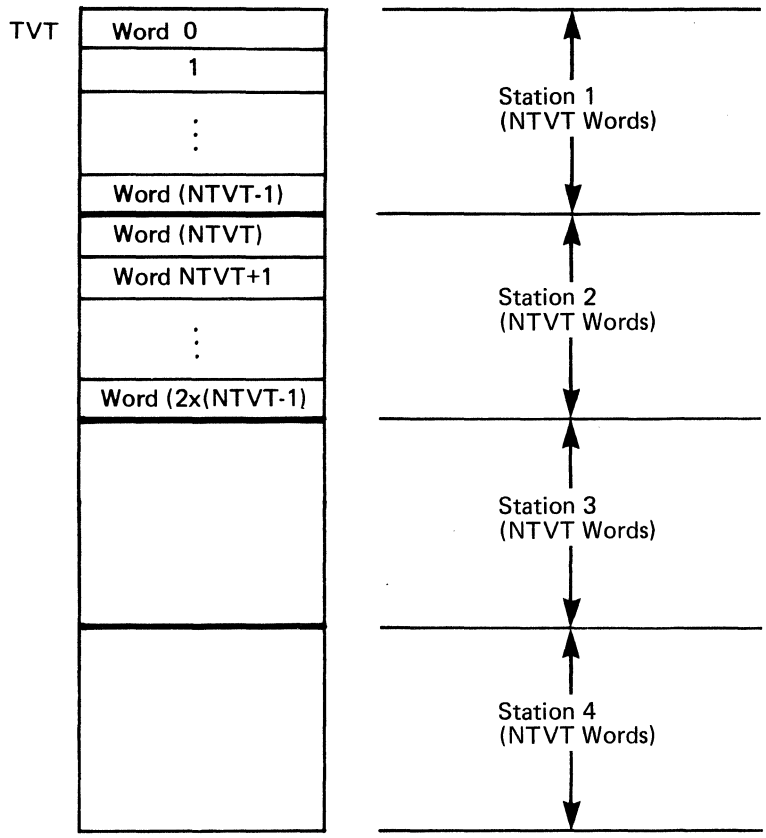
2.4.5 NTVT (GLOVAR + 8)

Contains the number of variables available in the test head variable table TVT for each station. See the description of each variable in section 2.5.

TVT table size is NTVT X 4 stations.

2.4.6 TVT (GLOVAR + 9)

Contains the starting address of TVT table. Each station is allotted the same number of variables in the test head variable table.



2.4.7 NSVT (GLOVAR + 10)

Contains the number of variables in the current station variable table SVT. See the description of each variable in section 2.6.

2.4.8 SVT (GLOVAR + 11)

Contains the starting address of SVT table.

2.4.9 NMAC (GLOVAR + 12)

Contains the number of variables per entry in MACTAB.

2.4.10 FWMAC (GLOVAR + 13)

Contains the starting address of MACTAB.

2.4.11 LWMAC (GLOVAR + 14)

Contains the last-used entry address +1 of MACTAB. Points to the first available entry. If MACTAB is empty, LWMAC = FWMAC. When the table is full, LWMAC = LWSYS.

2.4.12 STAVKT (GLOVAR + 15)

Contains the IOATAB pointer of VKT initially assigned to all stations. If there are two VKTs in the system then VKT1 is used.

2.4.13 PIDPMF (GLOVAR + 16)

Contains the address of system PID entry in IOATAB from which the last command was entered.

2.4.14 PODPMF (GLOVAR + 17)

Contains the address of system POD entry in IOTAB from which the last command was entered.

2.4.15 DRPMF (GLOVAR + 18)

Contains the address of disk directory entry in IOATAB.

2.4.16 FWALT (GLOVAR + 19)

Contains the first word address of ALTER buffer.

2.4.17 LWALT (GLOVAR + 20)

Contains the last-used address of ALTER buffer +1. When the buffer is empty LWALT = FWALT. When the buffer is full, LWALT = TVT.

2.4.18 FWIOA (GLOVAR + 21)

Contains the first word address of I/O assignment table (IOATAB).

2.4.19 NIOA (GLOVAR + 22)

Contains the number of words per entry in IOATAB.

2.4.20 CURSYS (GLOVAR + 23)

Contains the indicator showing whether current operation is in foreground or background.

1 foreground
0 background

2.4.21 FGBGFL (GLOVAR + 24)

A flag to indicate that background is waiting for a foreground breakpoint to execute a memory move or vice versa.

1 Background wait for foreground
2 foreground wait for background

2.4.22 REVN (GLOVAR + 25)

Contains the revision level of the system.

2.4.23 JOB (GLOVAR + 26)

Contains the current job number.

2.4.24 TPHL (GLOVAR + 27)

Contains the test program header size.

2.4.25 OPHL (GLOVAR + 28)

Contains the overlay header size.

2.4.26 DATE (GLOVAR + 29, 30)

Contains the current date entered by the command: DATE. Any eight-character presentation of date is stored.

2.4.27 TIME (GLOVAR + 31)

Contains the current time in seconds initialized by the command: TIME. It is updated every time a clock pulse occurs.

2.4.28 PGPMF (GLOVAR + 32)

Contains the address of IOATAB where the entry is reserved for the test program paging.

2.4.29 LWCPU (GLOVAR + 33)

Contains the highest CPU memory address.

2.4.30 LWSYS (GLOVAR + 34)

Contains the highest address reserved by the system +1.

2.4.31 LWAM (GLOVAR + 35)

Contains the last available memory address for overlays, test programs, and other memory files. The area above this is used for stack. If there is no stack, LWAM equals LWCPU.

2.4.32 FWAM (GLOVAR + 36)

Contains the next available memory address for overlays, test programs, and other files which may be resident in memory . When there are no files in memory, FWAM equals LWSYS.

2.4.33 ADJFLG (GLOVAR + 37)

Contains the flag indicating that the ADJMEM subroutine is waiting for DMA to memory to complete before doing a memory move. Reserved for system use.

2.4.34 THDACT (GLOVAR + 38)

Contains the test head active flag. It is set to 1 when THD is entered and cleared to 0 on exit. Entering the background to wait for tester activity does not affect the flag. This flag prohibits entering THD reentrantly. An overlay called from the keyboard that uses the tester may use the flag to control test head use.

Example:

```
--                          program entry
LDA         THDACT          TEST HEAD AVAILABLE
BZ          *+3             YES
BSM*        MONINT          NO, WAIT
BRU         *-3             RETRY
LDA         D1
STA         THDACT          PREVENT OTHER TEST HEAD USE
.
.
.
.
CLA
STA         THDACT
--                          program exit
```

2.4.35 PIDFLG (GLOVAR + 39)

Reserved for use by the monitor to indicate that a keyboard command is in a buffer and ready to be processed.

2.4.36 ECHFLG (GLOVAR + 40)

Contains a flag to PROCESS.

2,0 Do not echo command.
-1 Echo command if input is not VKT.
2 Process scans disk for command.
1 Noise words are acceptable. Do not scan disk.

2.4.37 COMIMG (GLOVAR + 41)

Contains the data formed by the PROCESS routine using the monitor command table. Any key words appearing in the monitor command table are picked up, and the bit configuration provided for the key words are stored in COMIMG before any overlay is called.

2.4.38 CMDPMF (GLOVAR + 42, 43, 44)

The PMF header for the system command. Any record read by the monitor is stored in the buffer pointed to by CMDPMF.

2.4.39 OCTAL (GLOVAR + 45)

Contains octal value obtained by the INTSCN routine.

2.4.40 OFLERR (GLOVAR + 46)

A flag to indicate that a decimal digit 8 or 9 appeared during scanning a number in INTSCN. It is a nonzero when the above condition occurs.

2.4.41 LDFLG (GLOVAR + 47)

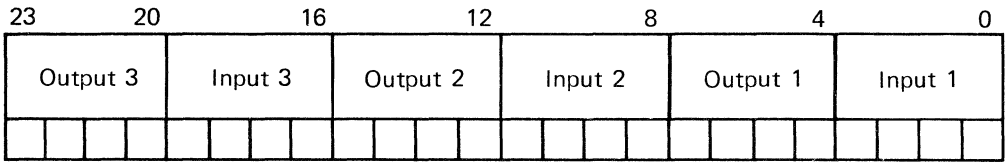
Set when LOAD is called. Overlays may not call LOAD unless this flag is clear.

2.4.42 MANTISSA (GLOVAR + 48)

Contains the integer value obtained by the NUMBER subroutine.

2.4.43 CMDV (GLOVAR + 49)

Contains the device codes obtained from a command by calling the PROCESS routine. Up to three input and three output device codes are saved.



Device Code	Input	Output
0	PID	POD
1	TTK	TTP
2	MTR1	MTW1
3	MTR2	MTW2
4	CR	LP
5	DIF	DOF
6	CLI	CLO
7	VK2	VP2
8	MIF	MOF

If no device is specified in a command it contains 0.

2.4.44 NAMEM1, NAMEM2 (GLOVAR +50, GLOVAR +51)
NAMEM3, NAMEM4 (GLOVAR +60, GLOVAR +61)
NAMEM5, NAMEM6 (GLOVAR +58, GLOVAR +59)

These cells are used to store string names appearing in a command during the execution of PROCESS routine. Up to six characters of the first string are stored in NAMEM2 left justified. The second string is stored in NAMEM3 and NAMEM4 and the third string in NAMEM5, NAMEM6.

2.4.45 BINUM (GLOVAR + 52)

Contains the binary value obtained by INTSCN/NUMBER.

2.4.46 BINC (GLOVAR + 53)

Contains the binary digit count in BINUM.

2.4.47 COLFLG (GLOVAR + 54)

The THD sets this flag to control the output of engineering numbers by PUTE. Reserved for system use.

2.4.48 RSTIO (GLOVAR + 55)

Reset interrupt sets this flag if foreground I/O is in progress. Reserved for system use.

2.4.49 ACTFIO (GLOVAR + 56)

This flag is set when foreground I/O is in progress. Reserved for system use.

2.4.50 MEMBSY (GLOVAR + 57)

This flag is set when DMA to or from memory is in progress. It prevents memory moves. Reserved for system use.

2.4.51 ONUMB1, ONUMB2 (GLOVAR + 62, GLOVAR + 63)

Contain octal form of numbers appearing in a command. The numbers are stored by the PROCESS routine. A one indicates that no or only one number appeared. Values are always absolute (signs are ignored).

2.4.52 NUMB1, NUMB2 (GLOVAR + 64, GLOVAR + 65)

Used to store numbers that appear in a command during the execution of PROCESS routine. The first number to appear in the command is stored in the NUMB1 and the second in NUMB2 in signed floating point format. A -1 in a cell indicates that no or only one number appeared.

2.4.53 STATC (GLOVAR + 66)

Contains the decoded station identification appearing in a command. The decoded form is 1, 2, 3 or 4. It contains -1 when a station ID is not entered in the command.

2.4.54 SPNUM1, SPNUM2, SPNUM3, SPNUM4, SPNUM5, SPNUM6 (GLOVAR + 67, 68, 69, 70, 71, 72)

Used to store numbers that appear in a command in the identifier number batched form during execution of PROCESS routine. A number is stored to a specific cell as directed by the key number supplied by the user. A -1 in any cell implies no number.

2.4.55 BINARY (GLOVAR + 73)

Contains the binary value obtained by the subroutine PROCESS. The binary number must have the format nnnn, where n is either 1 or 0.

2.4.56 INUMB1, INUMB2 (GLOVAR + 74, 75)

Contain values equivalent to NUMB1 and NUMB2 in integer format. The first number that appears is placed in INUMB1 and the second in INUMB2 by the subroutine PROCESS. These cells are initialized to -1. The values are always absolute (signs are ignored).

2.4.57 BFLERR (GLOVAR + 76)

A flag to indicate that the number scanned contains a digit greater than 1 and cannot be a binary number. It is set by INTSCN and checked by NUMBER routine.

2.4.58 SPOPT (GLOVAR + 77)

Contains special character flag bits set by PROCESS. If a special character is sensed during command scan, the defined bit is set in SPOPT. It is initialized to zero.

Bit	Special Character
14	a comma (,) is sensed
0	a minus sign (-) is sensed
1	a plus sign (+) is sensed

2.4.59 BINCNT (GLOVAR + 78)

Contains the number of digits sensed for the binary number during number scan. The number is placed in the global BINARY.

2.4.60 AWATF (GLOVAR + 79)

Contains the address of scheduler wait flag. The flag must be set to 1 by the tester interrupt service if the wait condition is completed. A0M* AWATF

2.4.61 ATHDF (GLOVAR + 80)

Contains the address of scheduler flag for tester start. The flag must be set to 1 by the tester interrupt service if the start interrupt occurs. A0M* ATHDF

2.4.62 NUMFLG (GLOVAR + 81, 82)

A flag that indicates that one or more digits has appeared in an identifier. The first word contains the character-count (position of the first digit in the buffer) and the second word contains the starting character-count of the identifier. NUMFLG + 1 is set for any identifier by $PARSE. NUMFLG is set by IDTSCN only if a digit is sensed in the identifier. These cells are used by PROCESS to obtain the special numbers in the form of XXXnnn where XXX is an identifier and nnn is a number.

2.4.63 NAME1, NAME2 (GLOVAR + 83, 84)

Contain the names or strings scanned by the routine IDTSCN. The maximum of eight characters is packed in TRASCII left justified. These cells are also used by SEARCH routine as input names to be searched in the table.

2.4.64 RAIDRR (GLOVAR + 86)

Reserved for system use by RAID to store a relocation register.

2.4.65 DBUGSA (GLOVAR + 87)

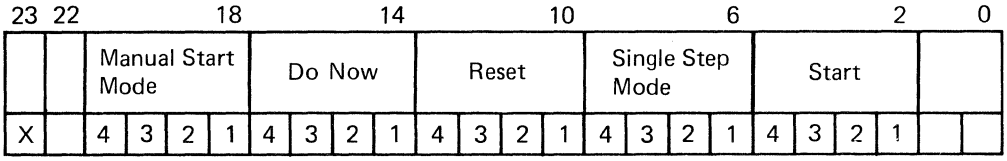
Contains the address of DEBUG address halt entry point in the monitor. BSM* DBUGSA is placed in the user's program.

2.4.66 DFSTAT (GLOVAR + 88)

Contains the station identification entered by: SET STATn. It is a binary value of 1 through 4. It is used as the default station identification for processing commands that require STATn when a station number is not entered in the command.

2.4.67 SMAFLG (GLOVAR + 89)

Contains the station control information for test head driver issued by the user through manual analysis command.



START	Request for test start from MA (START or carriage return after MANUAL or STEP). Cleared by test head driver when accepted.
SINGLE STEP MODE	Request to single step (STEP). Cleared by MA upon MANUAL, CONTINUE, or STEP OFF.
RESET	Request to clear before test start (RESET) cleared by test head driver when accepted.
DO NOW	Request to call MA foreground immediately. (MEAS, READ, WRITE, DISP without 'ON'). Cleared by test head driver when accepted. START bit is also set.
MANUAL START MODE	Request to start test with carriage return. Used by MA only. Cleared by CONTINUE or MANUAL OFF.
B23	When there is at least one request of START or DO NOW present.

2.4.68 RAIDER (GLOVAR + 91)

Reserved for system use by RAID.

2.4.69 RAIDBK (GLOVAR + 92)

Reserved for RAID during breakpoint execution.

2.4.70 SPDA (GLOVAR + 90)

A flag to indicate that parameter distribution overlay buffer area is busy when SPDA = 1. SPDA = 0 when it is not busy.

2.4.71 AFGBGF (GLOVAR + 93)

Reserved for system use.

2.4.72 SYSREL (GLOVAR + 94)

Contains the system release level in TRASCII that is printed by DIRECT and TE. It is patched by the system generation procedure.

2.4.73 MASTAT (GLOVAR + 95)

Contains the station ID used to start testing by entering a carriage return. It is set by MA upon MANUAL or STEP request. A 0 indicates no carriage return requests. Bit 23 set to 1 indicates VK2. Bits 2 to 0 specify station number (1 to 4).

2.4.74 CLIOID (GLOVAR + 96, 97)

Contains the six-character file name entered in OPEN or USE command.

2.4.75 LOTNUM (GLOVAR + 98, 99, 100)

Contains the lot number used by the Integrator system. Before calling $IOCS to open a file in the Integrator system to output tester data, these cells must be set.

2.4.76 DEVNUM (GLOVAR + 101, 102)

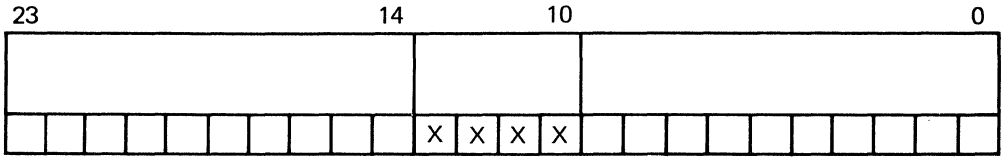
Contains the device number used by the Integrator system; usage is the same as for LITNUM.

2.4.77 CATGRY (GLOVAR + 103, 104, 105)

Contains the category number used by the Integrator system. Usage is the same as for LOTNUM.

2.4.78 RSTTSC (GLOVAR + 106)

Contains the image of STST register when the reset interrupt occurred.



Bit	Function
10	Reset on station 1
11	Reset on station 2
12	Reset on station 3
13	Reset on station 4

2.4.79 BGID (GLOVAR + 107, 108)

Can be set by overlays if a message END of XXXX is to be ouput upon overlay termination. BGID contains the program name XXXX. If it is zero, the message is not output.

2.5 TESTER VARIABLES

A listing of tester variables appears in figure 2-4.

```
                PAGE
          *
          *
          * TEST HEAD VARIABLE TABLE EQU'S (TVT)
          *
          *         ACCESSED BY INDEX TP(X2)
          *
00000222  TVTL      EQU   146
          *
00001110  TVTLT     EQU   TVTL+TVTL+TVTL+TVTL  4 STATIONS
          *
00000000  TSWITCH   EQU   0          GLOBAL VARIABLE SWITCH
00000001  TVALUE    EQU   1          GLOBAL VARIABLE VALUE
00000002  TSN       EQU   2          GLOBAL VARIABLE SN
00000003  TTT       EQU   3          GLOBAL VARIABLE TT
00000004  TDATAL    EQU   4          DATALOG REQUEST
00000005  TRTD      EQU   5          ROUND TRIP DELAY
          *         EQU   6               SPARE
          *         EQU   7               SPARE
          *         EQU   8               SPARE
          *         EQU   9               SPARE
          *         EQU   10              SPARE
00000013  TGLOB1    EQU   11         GLOBAL VARIABLE GLOB1
          *TGLOB2   EQU   12         GLOBAL VARIABLE GLOB2
          *TGLOB3   EQU   13         GLOBAL VARIABLE GLOB3
          *TGLOB4   EQU   14         GLOBAL VARIABLE GLOB4
          *TGLOB5   EQU   15         GLOBAL VARIABLE GLOB5
          *TGLOB6   EQU   16         GLOBAL VARIABLE GLOB6
          *TGLOB7   EQU   17         GLOBAL VARIABLE GLOB7
          *TGLOB8   EQU   18         GLOBAL VARIABLE GLOB8
          *TGLOB9   EQU   19         GLOBAL VARIABLE GLOB9
          *TGLO10   EQU   20         GLOBAL VARIABLE GLOB10
          *TGLO11   EQU   21         GLOBAL VARIABLE GLOB11
          *TGLO12   EQU   22         GLOBAL VARIABLE GLOB12
          *TGLO13   EQU   23         GLOBAL VARIABLE GLOB13
          *TGLO14   EQU   24         GLOBAL VARIABLE GLOB14
          *TGLO15   EQU   25         GLOBAL VARIABLE GLOB15
          *TGLO16   EQU   26         GLOBAL VARIABLE GLOB16
          *TGLO17   EQU   27         GLOBAL VARIABLE GLOB17
          *TGLO18   EQU   28         GLOBAL VARIABLE GLOB18
          *TGLO19   EQU   29         GLOBAL VARIABLE GLOB19
          *TGLO20   EQU   30         GLOBAL VARIABLE GLOB20
          *TGLO21   EQU   31         GLOBAL VARIABLE GLOB21
          *TGLO22   EQU   32         GLOBAL VARIABLE GLOB22
          *TGLO23   EQU   33         GLOBAL VARIABLE GLOB23
          *TGLO24   EQU   34         GLOBAL VARIABLE GLOB24
          *TGLO25   EQU   35         GLOBAL VARIABLE GLOB25
          *TGLO26   EQU   36         GLOBAL VARIABLE GLOB26
          *TGLO27   EQU   37         GLOBAL VARIABLE GLOB27
          *TGLO28   EQU   38         GLOBAL VARIABLE GLOB28
          *TGLO29   EQU   39         GLOBAL VARIABLE GLOB29
          *TGLO30   EQU   40         GLOBAL VARIABLE GLOB30
          *TGLO31   EQU   41         GLOBAL VARIABLE GLOB31
          *TGLO32   EQU   42         GLOBAL VARIABLE GLOB32
          *TGLO33   EQU   43         GLOBAL VARIABLE GLOB33
          *TGLO34   EQU   44         GLOBAL VARIABLE GLOB34
          *TGLO35   EQU   45         GLOBAL VARIABLE GLOB35
          *TGLO36   EQU   46         GLOBAL VARIABLE GLOB36
          *TGLO37   EQU   47         GLOBAL VARIABLE GLOB37
          *TGLO38   EQU   48         GLOBAL VARIABLE GLOB38
          *TGLO39   EQU   49         GLOBAL VARIABLE GLOB39
00000062  TGLO40    EQU   50         GLOBAL VARIABLE GLOB40
          *         EQU   51              SPARE
```

Figure 2-4 Tester Variables Listing

```
           *         EQU   52            SPARE
           *         EQU   53            SPARE
           *         EQU   54            SPARE
           *         EQU   55            SPARE
           *         EQU   56            SPARE
           *         EQU   57            SPARE
           *         EQU   58            SPARE
           *         EQU   59            SPARE
           *         EQU   60            SPARE
           *
00000075   TINDEX    EQU   61        BINNIG INDEX
00000076   TCPC      EQU   62        PROCESS CONTROL,PAUSE,SYNC
00000077   TDCDLY    EQU   63        DC TIME DELAY
00000100   TODLY     EQU   64        LM TIME OUT DELAY
00000101   TOVER     EQU   65        MA OVERRIDE
           *         EQU   66             SPARE
00000103   TDLO      EQU   67        DL OPTION & DEVICE
00000104   TDLF      EQU   68        DATALOG FREQUENCY COUNT
00000105   TDLS      EQU   69        DATALOG SKIP CONTROL
00000106   TDLR      EQU   70        DATALOG REQUEST
00000107   TDLC      EQU   71        DATALOG CONTROL & STATUS
00000110   TLMFC     EQU   72        DATALOG ADDITIONAL FAIL COUNT
00000111   TPDF      EQU   73        PD FREQUENCY COUNT
00000112   TPDS      EQU   74        PD SKIP CONTROL
00000113   TPDR      EQU   75        PD REQUEST
00000114   TDFR      EQU   76        DCF REQUEST
00000115   TMACTL    EQU   77        MA CONTROL
00000116   TPPO      EQU   78        MA PPM REQUEST
00000117   TSYNC     EQU   79        MA SYNC COUNT
           *         EQU   80             SPARE
00000121   TMOD      EQU   81        PPM,LM MODULE NUMBER
           *
00000122   TAPMP1    EQU   82        APM PROCEDURES #1
00000123   TAPMP2    EQU   83        APM PROCEDURES #2
00000124   TAPMF1    EQU   84        APM FILE NAME #1
00000125   TAPMF2    EQU   85        APM FILE NAME #2
           *
           *         EQU   86             SPARE
           *         EQU   87             SPARE
           *         EQU   88             SPARE
           *         EQU   89             SPARE
           *         EQU   90             SPARE
           *         EQU   91             SPARE
           *         EQU   92             SPARE
00000135   TPDD      EQU   93        DIST COUNT
00000135   TVTLL     EQU   93        END OF LOCAL CLEAR (// CLEAR STAT)
00000136   TSTEP     EQU   94        PROGRAM STEP COUNT
00000137   TPAUSE    EQU   95        PROGRAM PAUSE COUNT
00000140   TIP       EQU   96        INSTRUCTION POINTER
00000141   TBINT     EQU   97        BIN INITIALIZED
00000142   TBINS     EQU   98        BIN STATUS
00000143   TMPIN     EQU   99        MAX PIN (DEFAULT = 60)
           *         EQU   100            SPARE
           *         EQU   101            SPARE
           *         EQU   102            SPARE
           *         EQU   103            SPARE
           *         EQU   104            SPARE
00000150   TVTLG     EQU   104       END OF GLOBAL CLEAR (// LOAD STAT)
00000151   TPID      EQU   105       PID ADDRESS IN IOATAB
00000152   TTTK      EQU   106       TTK  ADDRESS IN IOATAB
00000153   TMTR1     EQU   107       MTR1 ADDRESS IN IOATAB
00000154   TMTR2     EQU   108       MTR2 ADDRESS IN IOATAB
00000155   TCR       EQU   109       CR   ADDRESS IN IOATAB
```

Figure 2-4 Tester Variables Listing (Continued)

```
00000156  TDIF      EQU   110        DIF  ADDRESS IN IOATAB
          *         EQU   111             SPARE
00000160  TVK2      EQU   112        VK2 ADDRESS IN IOATAB
00000161  TMIF      EQU   113        MIF ADDRESS IN IOATAB
00000162  TPOD      EQU   114        POD  ADDRESS IN IOATAB
00000163  TTTP      EQU   115        TTP  ADDRESS IN IOATAB
00000164  TMTW1     EQU   116        MTW1 ADDRESS IN IOATAB
00000165  TMTW2     EQU   117        MTW2 ADDRESS IN IOATAB
00000166  TLP       EQU   118        LP   ADDRESS IN IOATAB
00000167  TDOF      EQU   119        DOF  ADDRESS IN IOATAB
00000170  TCLO      EQU   120        CLO  ADDRESS IN IOATAB
00000171  TVP2      EQU   121        VP2 ADDRESS IN IOATAB
00000172  TMOF      EQU   122        MOF ADDRESS IN IOATAB
00000173  TOPT      EQU   123        TESTER OPTION CONTROL
00000174  TATTA     EQU   124        ATTACH FLAG
00000175  TJOB      EQU   125        STATION'S JOB NUMBER
00000176  TMSTK     EQU   126        MAX STACK SIZE USED
00000177  TOMSTK    EQU   127        MAX STACK SIZE SPECIFIED
00000200  TRTDS     EQU   128        SAVE ROUND TRIP DELAY SO IT WONT BE CLEARED
          *         EQU   129             SPARE
00000202  TTITLE    EQU   130 THRU 145 STATION TITLE
```

Figure 2-4 Tester Variables Listing (Continued)

2.5.1 TSWITCH (TVT + 0)

Contains the value in floating point, either programmed as the SWITCH variable or set by the command: SWITCH.

2.5.2 TVALUE (TVT + 1)

Contains the measurement value in floating point. It is set during execution of MEASURE VALUE, PIN, VARIABLE, and MACRO MEASURE PIN.

2.5.3 TSN (TVT + 2)

Contains the serial number in floating point either programmed as the SN variable or set by the command: SN.

2.5.4 TTT (TVT + 3)

Contains the test type programmed as the TT variable in floating point.

2.5.5 TDATAL (TVT + 4)

Contains the floating point representation of the datalog request flag TDLR. It can be accessed by the test program to override an operator requested option. The program may not turn on a request not specified by the operator. Any combination of bits may be entered.

FCT Count	4010B
FCT IFM	4020B
EOT	200B
LOG	400B
MEAS	1000B
DCT	2000B
FCT	4000B
TRIP	10000B

2.5.6 TRTD (TVT + 5)

Contains the floating point value of the round trip delay. It is added to the strobe values during program execution. It may be changed by a FACTOR program or displayed by the operator with the command: READ RTD.

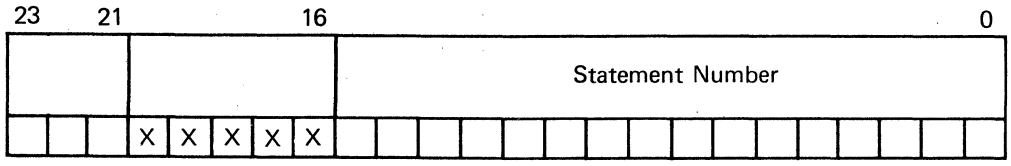
2.5.7 TGLOB1 - TGLO40 (TVT + 11 - 50)

Defined and used by the FACTOR programs and by ALLINK programs, and may be set or displayed by the command: GLOB.

2.5.8 TINDEX (TVT + 61)

Contains the count of number of times BIN specification has been updated.

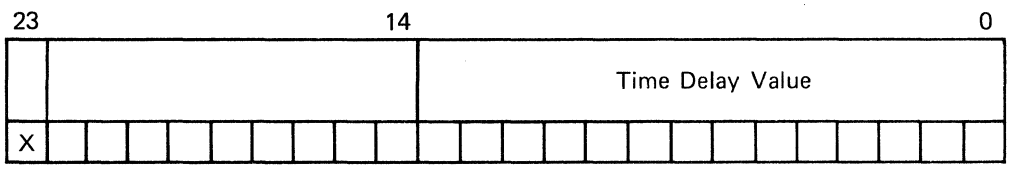
2.5.9 TCPC (Command Processor Control) (TVT + 62)



Bit	Description
20	Pause on statement number requested
19	Pause on fail requested
18	Sync on statement number requested
17	Sync on ADDR
16	Sync on COUNT (TSYNC contains COUNT value)
15 to 0	Statement number

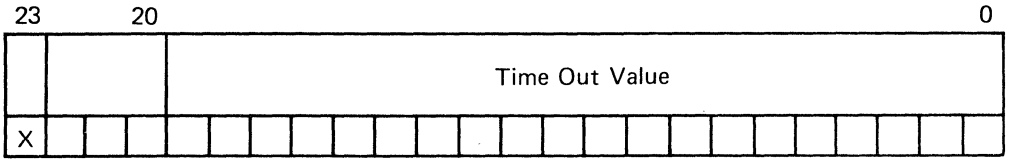
Set by manual analysis and used by THD.

2.5.10 TDCDLY (DC time DeLaY) (TVT + 63)



Bit	Description
23	Modify active (do not use programmed value)
13 to 0	DC time delay value

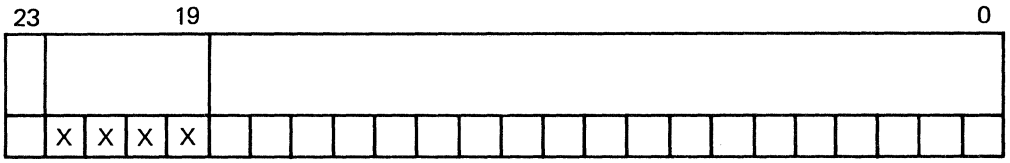
2.5.11 TODLY (Time Out)(TVT + 64)



Bit	Description
23	Modify active (do not use programmed value)
19 to 0	Time out value

TDCDLY and TODLY are set by MA when MODIFY commands are entered. These cells are cleared when MODIFY OFF, CLEAR, or LOAD command is entered.

2.5.12 TOVER (OVERride) (TVT + 65)



Bit	Description
22	Override on TRIP
21	Override on FCT ALL
20	Override on DCT
19	Override on RESET

2.5.13 TDLO (TVT + 67)

Contains the DATALOG device code.

Bit	Description
22	Top of form on line printer
3 to 0	Device Code 0 Station POD 1 TTP 2 MTW1 3 MTW2 4 LP 5 DOF 6 CLO 7 TTP2

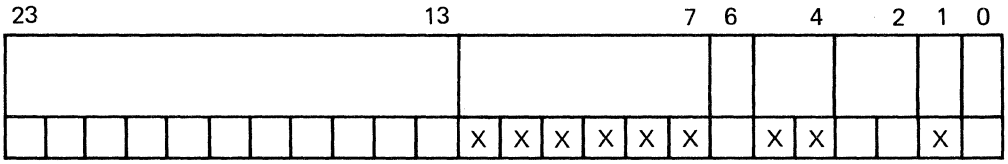
2.5.14 TDLF (TVT + 68)

Contains the DATALOG frequency request. Set by FRQn option to DATALOG command. If any log option is specified, this cell is cleared or set to the value specified by FRQn. TDLS is cleared to cause the first device to log.

2.5.15 TDLS (TVT + 69)

Running variable for the DATALOG frequency. It is initialized to zero and DATALOG overlay is called only when this is zero. At the end of a test, it is reset to TDLF or decremented to control datalogging.

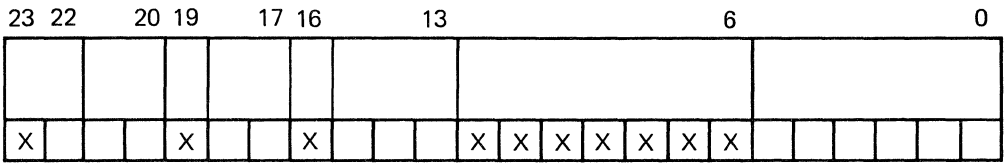
2.5.16 TDLR (DataLog Request) (TVT + 70)



Bit	Description	
12	DPS trip log requested	TRIP
11	Functional fail log requested	FCT
10	Measurement fail log requested	DCT
9	All measurement log requested	MEAS
8	MEASURE LOG statement log requested	LOG
7	EOT log requested	EOT
6	Reserved for future use	
5	Ignore fail mode (B11 also set)	IFM
4	Count requested (B11 also set)	COUNT
3	Spare	
2	Spare	
1	DATALOG OFF requested	

Set by DATALOG in background based on the parameters entered in DATALOG command.

2.5.17 TDLC (DataLog Control and status) (TVT + 71)



Bit	Description
23	Datalog active (set when TDLS = 0)
19	Time-out fail (bit 11 also set)
16	Reserved for Sentry VII development
12	Log trip fail
11	Log functional fail
10	Log measurement fail
9	Log all measurements
8	Log only measure long
7	Log EOT
6	Reserved for future use

2.5.18 TLMFC (TVT + 72)

Contains the additional fail request on a local memory load. It can be set by entering the additional number in the DATALOG command. When it is not zero, the specified number plus one or all fails, whichever is less, is datalogged.

2.5.19 TPDF (TVT + 73)

Contains the DIST frequency request.

2.5.20 TPDS (TVT + 74)

Similar to TDLS and used to keep track of skipping DIST calls.

2.5.21 TPDR (TVT + 75)

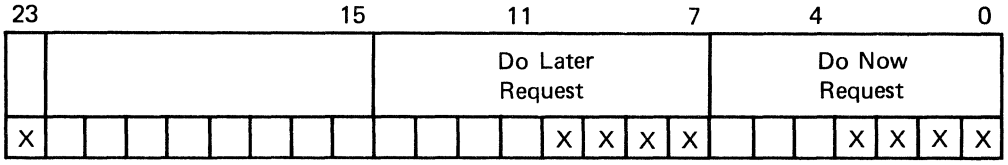
A DIST request flag. When it is not zero, DIST is considered active and DIST overlay is called for data collection.

2.5.22 TDFR (TVT + 76)

A DCF request flag. When it is not zero, DCF is considered active, and the DCF overlay is called for data collection.

2.5.23 TMACTL (TVT + 77)

Used to indicate manual analysis requests to foreground from background.



Bit 23 — Requests test head driver to enter MA foreground. Turned on by MA upon request. Turned off when there is no request for the next PAUSE and by the command CLEAR.

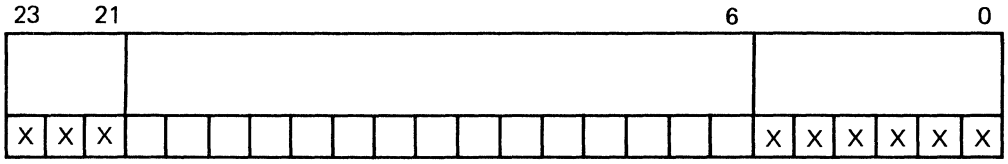
DO LATER REQUEST — Do the function at next PAUSE

Bit	Description
10	WRITE
9	DISPLAY
8	READ
7	MEASURE

DO NOW REQUEST — Do the function immediately

Bit	Description
3	WRITE
2	DISPLAY
1	READ
0	MEASURE

2.5.24 TPPO (TVT + 78)



Bit	Description
23	LOOP or STOP ON
22	LOOP
21	STOP
5 to 0	PPM address

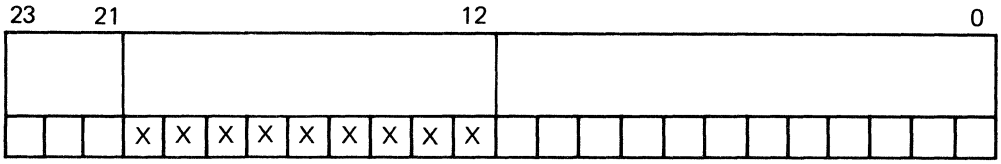
Contains MA request for LOOP/STOP ON PPM memory address.

2.5.25 TSYNC (TVT + 79)

Contains 24-bit count for MA SYNC ON COUNT request.

2.5.26 TMADSP (TVT + 80)

Used to indicate the Manual Analysis DISPLAY requests. DISPLAY ALL sets bits 14 to 20. DISPLAY TIME sets bits 12, 13, 19, and 20.



Bit	Description
20	TG
19	STRB
18	CLK
17	PWR
16	EIR
15	FCT
14	PMU
13	SCRAM (ETM)
12	TVn (ETM)

2.5.27 TMOD (TVT + 81)

Contains the current module number loaded to the PPM memory.

2.5.28 TAPMP1, TAPMP2 (TVT + 82, 83)

Contains the APM procedure download control. Each procedure executed by APM is assigned a VECTOR NUMBER by the FACTOR compiler. As each procedure is downloaded to APM, a bit is set in the appropriate control word designating that this procedure is now loaded. A maximum of 48 procedures may be sent to APM from any FACTOR program.

TAPMP1	Bits 23 to 0 represent procedures 47 to 24.
TAPMP2	Bits 23 to 0 represent procedures 23 to 0.

2.5.29 TAPMF1, TAPMF2 (TVT + 85, 86)

Contains the APM file download control. Each APM procedure file is assigned a unique file number by the FACTOR compiler. As each procedure file is processed by the APM test head driver, a bit is set in the appropriate control word designating that the file is complete. (It will not be processed again.) A maximum of 48 procedure files may be processed.

TAPMF1	Bits 23 to 0 represent APM procedure files 47 to 24, respectively.
TAPMF2	Bits 23 to 0 represent APM procedure files 23 to 0, respectively.

2.5.30 TPDD (TVT + 93)

Set by DIST to indicate that histogram data is in memory.

2.5.31 TVTLL (TVT + 93)

The TVT buffer from 0 through TVTLL is cleared by the CLEAR STATn command.

2.5.32 TSTEP (TVT + 94)

Contains the actual number of manual halts which occurred during a test. Used to determine if the test is a re-execution due to another station execution.

2.5.33 TPAUSE (TVT + 95)

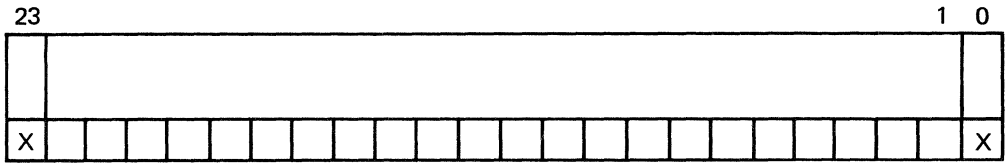
Contains the actual number of PAUSES that occurred during a test. Used to determine if the test is a re-execution due to another station execution.

2.5.34 TIP (TVT + 96)

Contains the current instruction number by counting every code as one instruction. It is not the statement number. The header is included; therefore, at the beginning of the test, it is initialized to the length of the header.

2.5.35 TBINT (TVT + 97)

Used to indicate that the binning test is initialized or the specification has been updated.



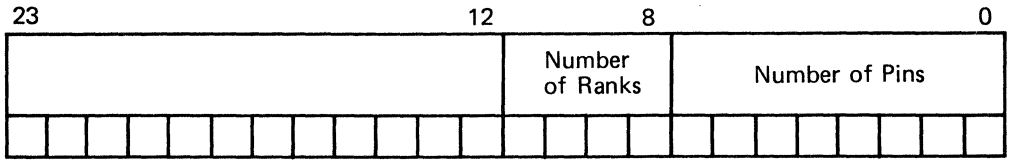
Bit	Description
23	Specification has been updated by UPDATE INDEX
0	Binning test is initialized

2.5.36 TBINS (TVT + 98)

Used to indicate binning is active for the station. Bit 23 set to one indicates that binning is active.

2.5.37 TMPIN (TVT + 99)

Used to save the maximum pin number set by the test program with SET MPIN. If none is set, the default is 60 for a Sentry VII and 120 for a Sentry VIII.



Bit	Description
11 to 8	Number of ranks allowed
7 to 0	Number of pins allowed

2.5.38 TVTLG (TVT + 104)

The TVT buffer from 0 through TVTLG is cleared by the LOAD STATn command.

2.5.39 TPID, TTTK, TMTR1, TMTR2, TCR, TDIF, TVK2, TMIF (TVT + 105 - 113)
TPOD, TTTD, TMTW1, TMTW2, TLD, TDOF, TCLO, TVP2, TMOF (TVT + 114 - 112)

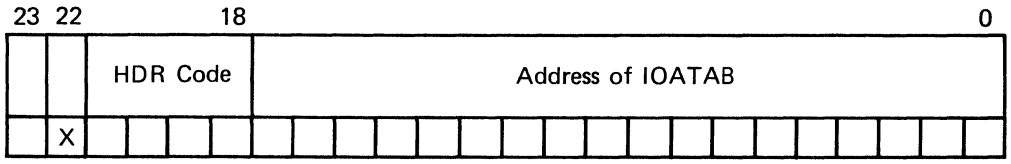
These cells are used to do I/O from FACTOR and datalog output. When the cells have bit 22 set to 1, usage of these devices requires an OPEN command and datalog OUTPUT is in binary format. Bits 17 to 0 contain the address of IOATAB if the device has been opened.

If bit 22 is set and bits 17 to 0 are zero at the time of usage, then a terminal error is issued.

If bit 22 is zero and bits 17 to 0 are zero, an OPEN call to $IOCS must be issued by the user and the address of IOATAB returned in X6 must be placed in the cell so that the device need not be opened for a following usage.

These cells can be located by calling GTTDV subroutine (SYXVEC + 33) and properly updated by calling FGOH and FGIO (SYXVEC + 72, 70) for doing I/O.

TPID/TPOD

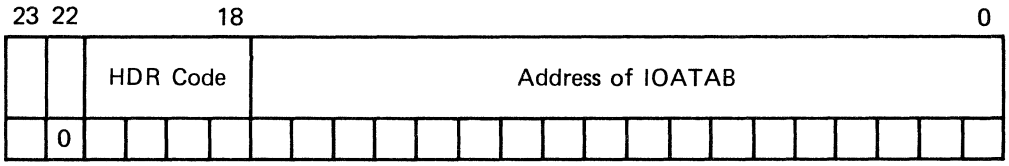


Bit 22 = 1 For USE [MTR1/MTR2/MTW1/MTW2/DIF/DOF/CLO] STATn.

Bit 22 = 0 For no USE command (when using system PID/POD) or for USE [TTK/TTP/LP/CR/PID/POD/VK2/VP2] STATn.

It is reset to system PID/POD upon CLEAR or LOAD STATn by closing the file previously used.

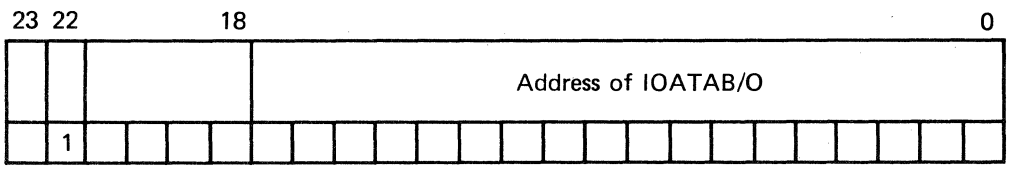
TTTK/TTTP/TCR/TLP/TTTK2/TTTP2/TVK2/TVP2



It is cleared to zero upon CLEAR or LOAD STATn

TMTR1/TMTR2/TDIF/TMTW2/TDOR/TCLO/TMIF/TMOF

The address is cleared to zero upon CLOSE command. These cells are not cleared (closed) upon CLEAR or LOAD STATn unless the device is opened by USE command.



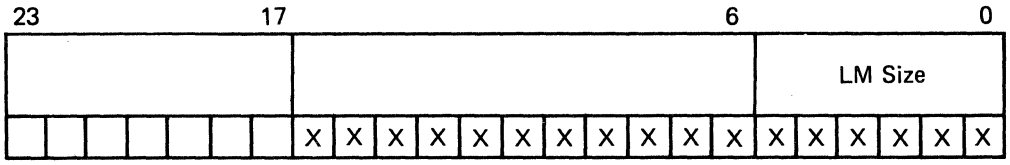
HDR CODE indicates which data type was last output to this device.

Bits 21 to 18	Last header issued code
	0 None
	1 Trip
	2 Measure/Measure; Log/Measurement Fail
	3 Spare
	4 FCT
	5 Spare
	6 Spare
	7 PPM
	8 PPM, DATA EXTENSION
	9 FACTOR WRITE/FACTOR PAUSE

The command OPEN normally sets binary devices, and the USE command sets TPID/TPOD. When the USE command is entered for a binary device (e.g. USE CLO ' name'), the corresponding cell (TCLO) is checked to see if the device is open. The same check is done at OPEN command so that the same device can be opened only once by a station.

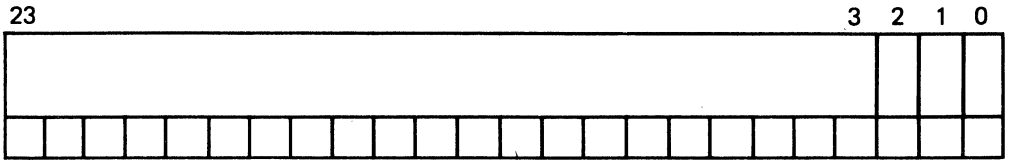
2.5.40 TOPT (TVT + 123)

Used to save hardware option. It is set during system initialization (after boot from magnetic tape or entered from DOPSY) and is never cleared.



Bit	Definition	Obtained From
16	Reserved for Sentry VII development	Possible M3 Change to S-20 (DATA FROM LMTRAQ source file.)
15	6 Range PMU	
14	Sentry VIII	Same as before
13	ETM	Bit 3 SAMC
12	28 volt swing	Bit 18 mode register
11	Low voltage	Bit 17 mode register
10	New REF/MUX module	Bit 15 mode register
9	2V/2mV Option	RVS range bit
8	SPM	Bit 8 SAMC
7	PPM	Bit 7 SAMC
6	10 MHz head	Bit 6 SAMC
5 to 0	LM size (number of thousands of words)	

2.5.41 TATTA (TVT + 124)



The overlays Binning, DCF, and PD are flagged that the LOAD STATn has taken place. LOAD sets the flag equal to 7. When the bit is set, the overlay prohibits new data for the current test program being added to data for the previous test plan. Before these overlays can be recalled any old data must be deleted. The overlays reset their respective bit when this occurs.

Bit	Description
0	Binning
1	DC failure analysis
2	Parameter distribution

2.5.42 TJOB (TVT + 125)

Contains the job name of the test program attached to the station. This allows the user to change stations for background use. During test program paging and EXEC assembly language overlays, the station job is searched, rather than the current job.

2.5.43 TMSTK (TVT + 126)

The maximum size that the run time stack reaches during execution. If the stack is forced beyond 500 words a message is printed at end of test: STATION n REQUIRED x WORDS OF STACK.

2.5.44 TOMSTK (TVT + 127)

Contains the stack size required by the test program.

2.5.45 TRTDS (TVT + 128)

Saves the round trip delay. TRTDS restores TRTD by the CLEAR and LOAD STATn commands.

2.5.46 TTITLE (TVT + 130 - 145)

Contains the TRASCII station title entered by the user. It prints whenever the station header outputs.

2.6 CURRENT STATION VARIABLES (SVT)

A listing of current station variables appear in figure 2-5.

```
                PAGE
          *
          * CURRENT STATION VARIABLE TABLE SVT
          *
          *         ADDRESSED BY INDEX SP(X1)
          *
00000062  SVTL      EQU   50
          *
          *
00000000  SITE      EQU   0          ON-LINE SITE
00000001  STHC      EQU   1          TEST HEAD CONTROL
00000002  SPIN      EQU   2          CURRENT PMU PIN NUMBER
00000003  SMSR      EQU   3          CURRENT PMU MEASUREMENT
00000004  SMF       EQU   4          MEASURE FLAG FOR DL
00000005  SEIR      EQU   5          EIR IMAGE
00000006  STEF      EQU   6          TERMINAL ERROR FLAG
00000007  SVOFFS    EQU   7          VOFFSET VALUE
00000010  SLML      EQU   8          LOCAL MEMORY LOCATION
00000011  STRIP     EQU   9          DPS TRIP STATUS
00000012  STPP      EQU   10         TEST PLAN ADDRESS
00000013  SMSRH     EQU   11         SAVE MEASUREMENT IN REG FORMAT FOR MA
00000014  SINC      EQU   12         INC ENABLE FLAG
00000015  SFVAL     EQU   13         FORCING VALUE FOR MACRO
00000016  SPG       EQU   14         LOCAL MEMORY PAGE SIZE
00000017  SPMOD     EQU   15         PROGRAM MODE (PPM/SPM,ETC)
00000020  SDLAF     EQU   16         DL ADDITIONAL FAIL FLAG
00000021  SIFC      EQU   17         IF COUNT FOR DL BY COUNT
00000022  SIFV      EQU   18         IF COUNT FOR MA SYNC, SET IF
00000023  SMR       EQU   19         MODE OF MEASUREMENT -SET PMU SENSE
00000024  SFR       EQU   20         MODE OF FORCE - SET PMU FORCE
00000025  SSAMC     EQU   21         CLAMP BIT (28 VOLT SWING) FOR SAMC, ANY WRITE
          *                          TO SAMC SHOULD OR IN THIS GLOBAL, EXCEPT
          *                          ANALYSIS
00000026  SQ        EQU   22         ORIGINAL VALUE OF Q REG
00000027  SQL       EQU   23         ORIGINAL VALUE OF QL
00000030  SLIM0     EQU   24         LIMIT 0 OR LIMIT IF ONLY ONE
00000031  SLIM1     EQU   25         LIMIT 1 WHEN THERE ARE 2 LIMITS
00000032  SDCT0E    EQU   26         ENABLE DCT0 FLAG WORD
00000033  SDCT0     EQU   27         ENABLE DCT0 VALUE
00000034  SDCT1E    EQU   28         ENABLE DCT1 FLAG WORD
00000035  SDCT1     EQU   29         ENABLE DCT1 VALUE
00000036  SILOE     EQU   30
00000037  SILO      EQU   31         ENABLED LIMIT VALUE
00000040  SIHIE     EQU   32         ENABLE IHI
00000041  SIHI      EQU   33
00000042  SVLOE     EQU   34         ENABLE VLO
00000043  SVLO      EQU   35
00000044  SVHIE     EQU   36         ENABLE VHI
00000045  SVHI      EQU   37
          *
00000046  S488CT    EQU   38         488 BUS CONTROL WORD
00000047  SAPMCT    EQU   39         APM CONTROL WORD
```

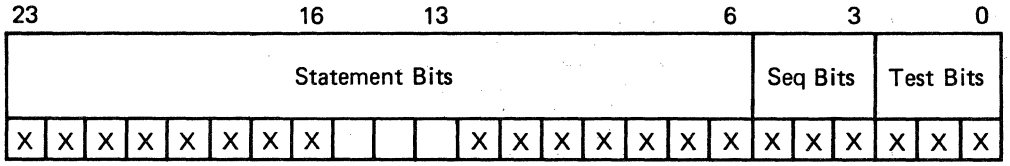
Figure 2-5 Current Station Variables Listing

2.6.1 SITE (SVT + 0)

Current station number (0 to 3). Set to minus one on entry to the operating system. It is used when a station comes on line to determine if the start is for the same station as the previous start.

0 = STAT1, 1 = STAT2, 2 = STAT3, 3 = STAT4

2.6.2 STHC (Test Head driver Control) (SVT + 1)



Bit	Description
23	One or more of the following bits are set
22	Tester busy complete
21	Time out interrupt - FCT fail
20	Instruction number compare interrupt
19	Fail occurred (bits 21, 16,12, 11, or 10 are set)
18	Pause statement executed
17	DC interrupt occurred (DC measure or DCT)
16	Reserved for Sentry VII development
15 to 13	Spare
12	TRIP - DPS TRIP FAIL interrupt
11	Functional fail interrupt
10	DC fail - measurement fail
9	Measurement executed
8	Measure, LOG executed
7	EOT
6	Reserved for D/L expansion
5 to 3	SEQ bits
2 to 0	TEST bits
5,2	FC fail
4,1	DC fail
3,0	Trip fail

Statement bits are cleared at the end of each statement execution.

SEQ bits are cleared at the end of each sequence and by the CLEAR FAIL statement. They are displayed to the EIR lights at the end of each sequence.

TEST bits are displayed at the end of each test and cleared by the CLEAR FAIL statement.

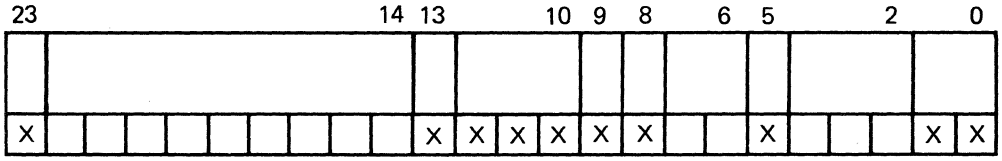
2.6.3 SPIN (SVT + 2)

Contains the pin number that the PMU is connected to following an interpretive CPMU PIN, or the pin measured following an interpretive measure or a macro, or a DMA measure. This is logged even though the PMU may be disconnected, as after measure node, or connected to a rest pin, as after a macro.

2.6.4 SMSR (SVT + 3)

Contains the floating point value measured following an interpretive measure or a DMA measure which is logged. The voltage offset value programmed is added to the measurement value in SMSR.

2.6.5 SMF (SVT + 4)



Bit	Description
23	For MEASURE VARIABLE
13	Mode of measurement 0 = current, 1 = voltage
12 to 10	Range of SMSR
9	If two limits enabled it contains pass/fail for SLIM0 = 1 SLIM0 failed = 0 SLIM0 passed
8	LT/GT for SLIM1 LT = 0, GT = 1
5	LT/GT for SLIM0 LT = 0, GT = 1
1, 0	Number of limits (0,1,2

2.6.6 SEIR (SVT + 5)

Contains EIR register information to be displayed. SEIR is written to the EIR register at the sequence halt (MANUAL, PAUSE, etc.) at the end of each test and at the terminal error. Any program writing to the EIR should put the value in SEIR.

Bit	Description
14	End of test
13	Functional test pass
12	Functional test fail
11	DC/TRIP test pass
10	DC/TRIP test fail
9 to 0	User written information or binning gates if used.

At a terminal error, bits 11 and 10 are on and bits 9 to 0 contain the terminal error number.

2.6.7 STEF (SVT + 6)

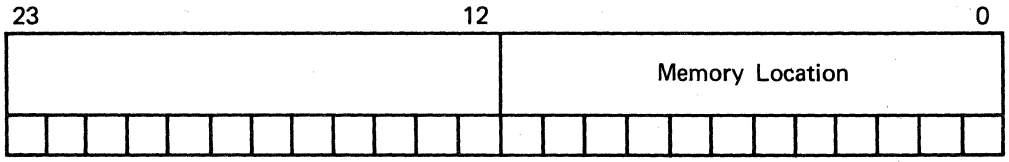
Contains the terminal error number. See the terminal error list for currently available error numbers.

2.6.8 SVOFFS (SVT + 7)

Voltage offset value is stored here in floating point format. All are zero if no offset is programmed.

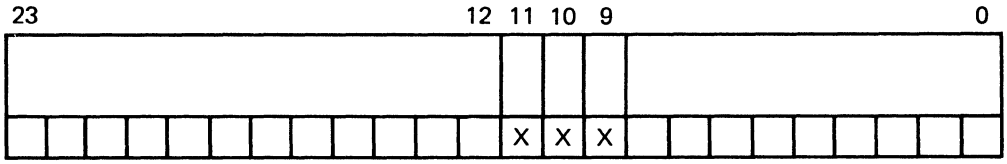
2.6.9 SLML (SVT + 8)

Contains the delayed memory address needed by the DATALOG overlay in order to obtain the functional data from the local memory.



2.6.10 STRIP (SVT + 9)

Contains the failed DPS number needed by the DATALOG overlay. The interrupt service for TRIP FAIL sets this information.



Bit	Failure
11	DPS3 TRIP
10	DPS2 TRIP
1	DPS1 TRIP

2.6.11 STPP (SVT + 10)

Contains the pointer to the absolute address of the test program for the station currently online. It points to one of the four ATPA cells when a test station is online.

2.6.12 SMSRH (SVT + 11)

Contains the results of a measurement following any MEASURE statement except MEASURE variable. The value does not have the programmed voltage offset added back as SMSR and TVALUE do. Bits 0 to 10 contain the measurement; bits 11 to 14 contain the mode and range from the PSL.

2.6.13 SINC (SVT + 12)

Flags test head driver to set the INC interrupt enable following a statement. After a station start request following a pause on statement number the IND is at the value which causes an interrupt. This interrupt is thrown away. After the IND is bumped by the next statement, SINC indicates that the INC interrupt enable should be turned back on.

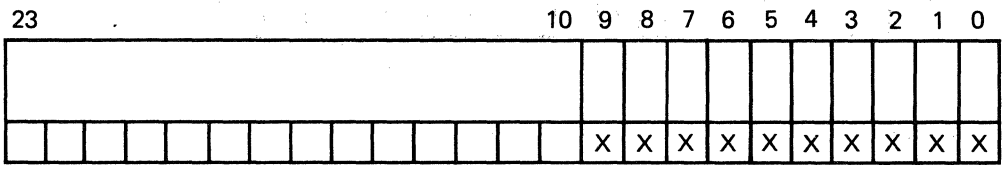
2.6.14 SFVAL (SVT + 13)

Contains the value of the PPS register at the time of the last SET TEST number. It is used by DC MACRO processing and analysis.

2.6.15 SPG (SVT + 14)

An indicator that tells if the currently running program contained a SET PAGE statement. If there is no SET PAGE, it is zero. Otherwise, it contains the local memory size specified by SET PAGE.

2.6.16 SPMOD (SVT + 15)



Bit	Description
9	2V/2mV program
8	SPM program
7	PDMA (SET PERIODi is DMA)
6	ETM program
5	APM program
4	488 Bus program
3	6 range PMU program
2	Sentry VIII program
1	Set PPM ON programmed; do not reset bit 0
0	REXEC executed or SET PPM ON, D/L treats a functional fail as a PPM fail

2.6.17 SDLAF (SVT + 16)

SDLAF is non-zero when additional fails are being logged. This controls the datalogger output.

2.6.18 SIFC (SVT + 17)

IGNORE FAIL register value used to save last location during additional datalog fails.

2.6.19 SIFV (SVT + 18)

IGNORE FAIL value set by SET IFAIL instruction and sync by count or ADDRESS.

2.6.20 SMR (SVT + 19)

The sense mode and range of the PMU set by the SET PMU SENSE statement is stored here.

Bit	Description
14,12,11	range
0	AUTO

2.6.21 SFR (SVT + 20)

The force mode of the PMU set by the SET PMU FORCE statement is stored here.

Bit	Description	
13	mode	1 = voltage
14,12,11	range	
0,1	AUTO	

2.6.22 SSAMC (SVT + 21)

Contains the value of bit 0 of SAMC for 28 volt swing.

Bit 0 = 1	Indicates that the reference voltage supplies are clamped at -22 volts.
Bit 0 = 0	Indicates that the reference voltage supplies are clamped at -16 volts. This occurs when a period is less than 200 ns.

2.6.23 SQ (SVT + 22)

Contains the original value of Q from the SET Q statement.

2.6.24 SQL (SVT + 23)

Contains the original value of QL from the SET Q statement.

2.6.25 SLIM0 (SVT + 24)

Contains the ENABLE DCT0 limit if two limits have been enabled for a measurement, or it contains either the DCT0, DCT1, or SET DCT limit if only one is enabled. The voltage offset has been added if it was programmed. This is the value printed by the datalogger. The value is in floating point.

2.6.26 SLIM1 (SVT + 25)

Contains the ENABLE DCT1 limit if two limits have been enabled for a measurement. The voltage offset has been added if it was programmed. This is the value printed by the datalogger. The value is in floating point.

2.6.27 SDCT0E (SVT + 26)

Flag word for ENABLE DCT0

Bit	Description
0	never enabled or disabled
-1	disabled
0	1 enabled
13	LT/GT flag 0 = LT, 1 = GT (DCT0/DCT1)

2.6.28 SDCT0 (SVT + 27)

Value of ENABLE DCT0 in floating point. The voltage offset is not added.

2.6.29 SDCT1E (SVT + 28)

Flag word for ENABLE DCT1. See SDCT0E.

2.6.30 SDCT1 (SVT + 29)

Value of ENABLE DCT1 in floating point. The voltage offset is not added.

2.6.31 SILOE (SVT + 30)

Flag word for ENABLE ILO.

Bit	Description
0	never enabled or disabled
-1	disabled
1	enabled

2.6.32 SILO (SVT + 31)

Value of ENABLE ILO in floating point.

2.6.33 SIHIE (SVT + 32)

Flag word for ENABLE IHI. See SILOE.

2.6.34 SIHI (SVT + 33)

Value of ENABLE IHI in floating point.

2.6.35 SVLOE (SVT + 34)

Flag word for ENABLE VLO. See SILOE.

2.6.36 SVLO (SVT + 35)

Value of ENABLE VLO in floating point. The voltage offset is added to the value.

2.6.37 SVHIE (SVT + 36)

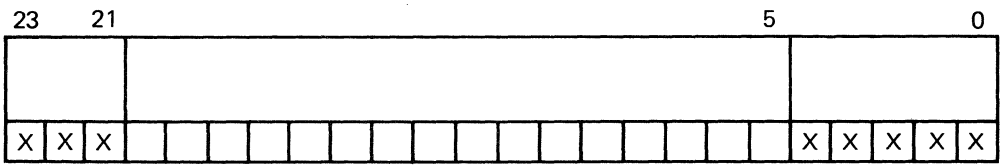
Flag word for ENABLE VHI. See SILOE.

2.6.38 SVHI (SVT + 37)

Value of ENABLE VHI in floating point. The voltage offset is added to the value.

2.6.39 S488CT (SVT + 38)

Contains the control word for execution of the FACTOR 488 statements.

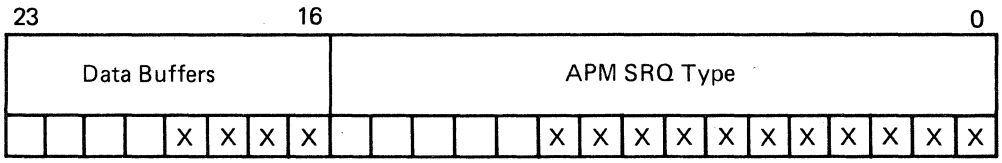


Bit	Description
23	IEEE 488 Bus SRQ interrupt pending
22	APM high speed sync (APS) interrupt pending
21	APM SRQ pending
4	400 opcode active
3	ON BRANCH active
2	APM checked and initialized
1	APM high speed sync branching disabled (ON APS)
0	IEEE 488 bus SRQ branching disabled (ON SRQ/ON PPR/ON APF)

Bits 23 and 22 are set directly by their respective interrupt service routines.

2.6.40 SAPMCT (SVT + 39)

APM control word for execution of APM programs. This word contains a history of all SRQs sent from APM. As APM requests are serviced, the corresponding bits in this control word are reset. Bits 0 to 15 represent the SRQ types sent by APM. Bit 0 represents SRQ type 0, bit 1 SRQ 1, etc.



Bit	Description
19	Data in APM error buffer
18	Data in APM data log buffer
17	Data in APM pass/fail test buffer
16	Data in APM measurement result buffer
10	APM system terminal error
9	Error SRQ
8	Information for data logger
7	Immediate statement complete
6	Abortive procedure end (APM fail)
5	Normal procedure end (APM pass)
4	APM at a pause
3	Completed data record
2	System operation complete
1	Trigger
0	Read data to send from data buffer

2.6.41 SPPM (SVT + 42)

Reserved for Sentry VII development.

2.6.42 SPERN (SVT + 43)

Reserved for Sentry VII development.

2.6.43 SPERV (SVT + 44)

Reserved for Sentry VII development.

3

System Subroutines

System subroutines are procedures that are available under MASTR for use by the operating system or user-written overlays. System subroutine definitions are provided unless the subroutine is reserved for system use. The reserved subroutines must not be CALLed by user overlays. Using the reserved subroutines may leave the operating system in an undefined state. Transfer vectors used to access the system subroutines are listed in figure 3-1.

```
                        PAGE
                *
      00000640          ORG   GLOVAR+NGLOV
                *
                *
                * SYSTEM SUBROUTINE TRANSFER VECTOR (BSM* (SYXVEC+N))
                *
                *
      00000640  SYXVEC  EQU   *         N
00640 00002536          TV    WAIT      0  TESTER BUSY WAIT IN FOREGROUND
00641 00006735          TV    OUTOPN    1  OPEN OUTPUT  FOR ALPHA PRINT
00642 00006642          TV    OUTCLS    2  CLOSE OUTPUT ALPHA PRINT
00643 00001726          TV    NUMERR    3  NUMBER ERROR          (BRU*)
00644 00001741          TV    COMERR    4  COMMAND SYNTAX ERROR  (BRU*)
00645 00010322          TV    $IOCS     5  I/O DRIVER
00646 00006651          TV    MSGIN     6  INPUT A MESSAGE FROM SYSTEM PID
00647 00006666          TV    MSGOUT    7  OUTPUT A MESSAGE ON SYSTEM POD
00650 00006702          TV    UMSGW     8  OUTPUT A MESSAGE ON SYSTEM POD W/O CR/LF
00651 00005663          TV    ADJMEM    9  ADJUST MEMORY USAGE
00652 00005655          TV    SCNFIL   10  SCAN FILES IN MEMORY
00653 00003276          TV    GTSTAT   11  DECODE STATION ID FROM COMMAND OR DEFAULT
00654 00017671          TV    CONV     12  CONVERT BINARY TO DECIMAL
00655 00005674          TV    PUTD     13  PUT DECIMAL NUMBER IN BUFFER
00656 00017625          TV    PUTC     14  PUT CHARACTER IN BUFFER
00657 00006343          TV    MOVEDN   15  MOVE A MEMORY BLOCK ASCENDING ORDER
00660 00006352          TV    MOVEUP   16  MOVE A MEMORY BLOCK DESCENDING ORDER
00661 00005666          TV    PUTE     17  PUT ENG # IN BUFFER
00662 00005671          TV    PUTO     18  PUT OCTAL # IN BUFFER
00663 00005660          TV    PROCESS  19  PROCESS A COMMAND
00664 00005724          TV    ALTER    20  ALTER BUFFER SCAN/PROCESS
00665 00005727          TV    $PARSE   21  DEFINE NEXT INPUT RECORD FIELD
00666 00005732          TV    IDTSCN   22  SCAN IDENTIFIER
00667 00005735          TV    NUMBER   23  SCAN A NUMBER (SET IN F.P. FORM)
00670 00005740          TV    INTSCN   24  SCAN INTEGER NUMBER
00671 00017467          TV    SEARCH   25  SEARCH TABLE
00672 00017514          TV    MPZERO   26  CLEAR CORE (X6,X7)
00673 00017523          TV    GETC     27  GET A CHARACTER FROM BUFFER
00674 00017564          TV    READW    28  GET A WORD FROM BUFFER
00675 00017577          TV    WRITEW   29  PUT WORD IN BUFFER
00676 00005425          TV    IERMSG   30  TERMINAL ERROR IN THD
00677 00005710          TV    DUMP     31  DUMP TP,OVLY,MOD
00700 00003307          TV    PUTIME   32  PUT DATE, TIME IN BUFFER
00701 00004360          TV    GTTDV    33  GET IO DEVICE ADDR IN TVT
00702 00005721          TV    HEADER   34  OUTPUT HEADER (BINN,PD,DCF,MA)
00703 00003106          TV    SPIOER   35  $IOCS ERROR CHECK AND MSG
00704 00003510          TV    FGOVC    36  CALL OVERLAY FOREGROUND
00705 00003474          TV    ALLEX    37  ASSEMBLY LANGUAGE LINKAGE EXECUTION
00706 00002341          TV    COMMND   38  RETURN TO MONITOR (BRU* )
00707 00017607          TV    PUTW     39  PLACE 4 TASCII CHARACTERS IN BUFFER
00710 00003442          TV    TWAIT    40  WAIT ON TESTER ACTIVITY
00711 00003403          TV    FGBGRT   41  SCHEDULE BACKGROUND FROM FOREGROUND
00712 00003637          TV    MONINT   42  ENTER SCHEDULER(SAVE HARDWARE)
00713 00005452          TV    SCALE    43  SCALE F.P. TO TESTER VALUES
00714 00003457          TV    FGWAIT   44  FOREGROUND WAIT FOR BACKGROUND
00715 00005343          TV    ERRCNV   45  $IOCS ERROR CODE DECODER, MESSAGE OUTPUT
00716 00005472          TV    FSUB     46  FLOATING POINT SUBTRACT (A-E -> A)
00717 00005500          TV    FAND     47  FLOATING POINT AND (A AND E -> A)
00720 00005506          TV    FEOR     48  FLOATING POINT EOR (A EOR E -> A)
00721 00005522          TV    FLOG     49  FLOATING POINT LOG (LOG A -> A)
00722 00005475          TV    FADD     50  FLOATING POINT ADD (A+E -> A)
00723 00005467          TV    FDIV     51  FLOATING POINT DIVIDE (A/E -> A)
00724 00005530          TV    FFIXS    52  FIX FLOATING POINT INTO A,E
00725 00005503          TV    FOR      53  FLOATING POINT OR (A OR E -> A)
```

Figure 3-1 System Subroutine Transfer Vector Listing

```
00726 00005511          TV    FNOT      54 FLOATING POINT NOT (A NOT F -> A)
00727 00005525          TV    FEXP      55 FLOATING POINT EXPONENT (A -> A)
00730 00005464          TV    FMUL      56 FLOATING POINT MULTIPLY (A * F -> A)
00731 00005514          TV    FFIX      57 FIX FLOATING POINT INTO A (A -> A)
00732 00005517          TV    FFLT      58 FLOAT INTEGER IN A (A -> A)
00733 00005533          TV    FFLTS     59 FLOAT NUMBER IN A,E (A,E -> A)
00734 00017712          TV    FCAM      60 FLOATING POINT COMPARE (A,E)
00735 00005705          TV    LOAD      61 LOAD A FILE INTO MEMORY
00736 00003052          TV    DELFIL    62 DELETE FILE BY NAME CHANGE
00737 00004420          TV    RELOV     63 CALL OVLY FOR RELEASE
00740 00004554          TV    ATTA      64 ATTACH OVLY TO STATN
00741 00004571          TV    DTTA      65 DETTACH OVLY FROM STATN
00742 00004663          TV    PAGETP    66 PAGE TEST PLAN
00743 00004604          TV    FGBGWT    67 FG/BG WAITING
00744 00004631          TV    FGBGH     68 FG/BG HALT FOR MEM MOVE
00745 00005436          TV    FINDVL    69 FIND VARIABLE ADDR IN STACK
00746 00006364          TV    FGIO      70 FOREGROUND IO
00747 00005441          TV    DMASTR    71 START TESTER DMA AND WAIT FOR DONE
00750 00006532          TV    FGOH      72 FOREGROUND HEADER OUT
00751 00005444          TV    ENBTST    73 ENABLE LOCAL MEMORY TEST
00752 00005447          TV    WWAIT     74 WAIT FOR LM TEST IN FG
00753 00005713          TV    ADRXLA    75 DISC ADDR TRANSLATE
00754 00005456          TV    INTERP    76 CALL INTERPRETER
00755 00005461          TV    ENTBSY    77 ENABLE TESTER BUSY COMP INTERRUPT
00756 00003611          TV    RSOVC     78 CALL OVLY FOR FG RESET
00757 00004025          TV    STALL     79 STALL BG PROCESS
00760 00005743          TV    UPDATE    80 CREATE/DELETE A FILE
00761 00005751          TV    PUTENG    81 PUT ENG VALUES IN BUFFER
00762 00005754          TV    PUTA      82 PUT OCTAL/DECIMAL ADDR IN BUFFER
00763 00003757          TV    BGCHK     83 CHECK IF BG ACTIVITY ON
00764 00000000          TV    0         84
00765 00005757          TV    CALLMOD   85 CALL OVLY MODULE
00766 00005677          TV    PUTB      86 PUT BINARY # IN BUFFER
00767 00005702          TV    PUTH      87 PUT HEX # IN BUFFER
00770 00006061          TV    SAVENV    88 SAVE ENVIRONMENT
00771 00006115          TV    RSVENV    89 RESTORE ENVIRONMENT
                  *     TV    IBIO      90 IBUS DRIVER - RESERVED
                  *
```

Figure 3-1 System Subroutine Transfer Vector Listing (Continued)

3.1 WAIT

Reference Location	SYXVEC + 0
Description	This is a wait routine called by the foreground processing when the test station becomes busy. It allows background processing to continue. Control returns to foreground only after the contents of address AWATF (GLOVAR + 79) are set to 1 by the interrupt service when the busy condition is completed.
Input Parameters	None
Output Parameters	None
Return	BSM address +1
Calling Sequence	BSM* WAIT
Registers Used	All registers and state switches are saved and restored.
Allowed Usage	Foreground only

3.2 OUTOPN

Reference Location	SYXVEC + 1
Description	This subroutine opens an output device for a string record file. It can be used to output data summary.
Input Parameters	CMDV (GLOVAR +49) must contain the device code. If CMDV is zero, the system POD is used. NAMEM1, NAMEM2 (GLOVAR +50, 51) must contain the file name if the device is a CLO or DOF.
Output Parameter	X6 contains the pointer to IOATAB for this device.
Return	BSM address +1 error return BSM address +2 normal return
Calling Sequence	BSM* OUTOPN
Routine Used	$IOCS
Registers Used	X1, X6, A, and E are not restored.
Allowed Usage	Background only

3.3 OUTCLS

Reference Location	SYXVEC +2
Description	This subroutine closes an output device.
Input Parameter	X6 pointer to IOATAB.
Output Parameters	None
Return	BSM address +1 error return BSM address +2 normal return
Calling Sequence	LDX* X6,IOATAB ptr. BSM* OUTCLS
Routine Used	$IOCS
Registers Used	X1, A, E and X6 are not restored.
Allowed Usage	Background only

3.4 NUMERR

Reference Location	SYXVEC + 3
Description	This handles number errors for all programs and subroutines. Because of its general nature, error reporting is at a minimum.
Input Parameters	None
Output Parameters	None
Return	Control is not returned to the caller. An error message, ERROR IN NUMB, is output to POD by the monitor, and control is passed to the monitor command scan routine.
Calling Sequence	BRU* NUMERR
Routine Used	MSGOUT to print out the error message by the monitor.
Registers Used	Not applicable since there is no return to caller.

Note

If the user intends to conduct any kind of recovery from a number error (for example, requesting a number again from an input device), this routine should not be used. Instead, BRU to another routine in the user's program for recovery.

Allowed Usage	Background only

3.5 COMERR

Reference Location	SYXVEC + 4
Description	Processes a command decoding error (decoding done in the subroutine called PROCESS) for all programs and subroutines. Because of its general nature, error reporting is at a minimum.
Input Parameter	E contains one-word text of error.
Output Parameters	None
Return	Control is not returned to the caller, an error message, ERROR IN XXXX, is output to POD by the monitor, and control is passed onto the monitor command scan routine. XXXX is the contents of E on entry.
Calling Sequence	BRU* COMERR
Routine Used	MSGOUT prints out the error message by the monitor.
Registers Used	Nonessential to the caller, as this is an error terminating subroutine.

Note

If the user intends to conduct any kind of recovery from a command error on further diagnosing the error, this subroutine should not be used. Instead, BRU to user's own error handling routine for this error processing.

Allowed Usage	Background only

3.6 $IOCS

Reference Location: SYXVEC + 5

Description: See section 4

3.7 MSGIN

Reference Location: SYXVEC + 6

Description: Reads a record from the system PID.

Input Parameters:

X1 Starting address of input buffer (buffer must have 18 words)
A Prompting character in TRASCII right justified or zero

Output Parameter: A $IOCS error code for error return

Return:

BSM address +1 error return
BSM address +2 normal return

Calling Sequence

```
INBUF   BSS     18
        LDX     X1,INBUF
        LDA     D0 or CLA
        BSM*    MSGIN
        BRU     ERROR
        --                      normal return
```

Registers Used: A, E, X1, and X6 are not restored.

Allowed Usage: Foreground/background

3.8 MSGOUT

Reference Location: SYXVEC +7

Description: Writes a record to the system VKT.

Input Parameters:

X1 Starting address or record buffer
X2 Number of words to be output

Output Parameter: A $IOCS error code for error return

Return:

BSM address +1 error return
BSM address +2 normal return

Calling Sequence

```
OUTDCB   DATA   0,OUTBUF,5
OUTBUF   TEXT   'XXXXXX WORDS LEFT'

         LDX    X2,5
         LDX    X1,OUTBUF
         BSM*   MSGOUT
         BRU    ERROR
         --                        error return
         --                        normal return
```

Registers Used — A, E, X1, and X6 are not restored.

Allowed Usage — Foreground/background

3.9 UMSGW

Reference Location — SYXVEC + 8

Description — Writes a record without carriage return and line-feed to system POD.

Input Parameters
- X1 Starting address of record buffer
- X2 Number of words to be output

Output Parameter — A $IOCS error code for error return

Return
- BSM address +1 error return
- BSM address +2 normal return

Calling Sequence

```
OUTBUF   TEXT   'PIN  = '

         LDX    X2,5
         LDX    X1,OUTBUF
         BSM*   UMSGW
         BRU    ERROR
         --                        normal return
```

Registers Used — A, E, X1, and X6 are not restored.

Allowed Usage — Foreground/background

3.10 ADJMEM

Reference Location

SYXVEC + 9

Description

This routine handles adjustment of dynamic memory allocation area. Its functions are releasing a file by name, or releasing all test programs or all overlays; expanding a test program or overlay, or repacking a test program or overlay. Physically, the data in the impacted memory area is moved up or down and the memory activity table is updated for the change.

Input Parameters

BSM address + 1
Function code to be performed

0 Release a file specified in X5
1 Release all test programs
2 Release all overlays
3 Release all test program overlays
4 Expand the test program or overlay specified in X5
5 Repack the test program or overlay specified in X5
6 Bump page TP or release inactive programs
7 Release all modules
8 Release all files previously marked for release
9 Make room for fixed overlay

For functions 0, 4 and 5, X5 must contain the address of the file entry in MACTAB.

For function 4, the A register must contain the expansion size in words.

For function 5, the A register must contain the new test program or overlay size in words.

For functions 6 and 9, the A register must contain the required size in words.

Output Parameters

For function 9, X5 points to MACTAB entry.

Return

For functions 4, 5, 6, and 9, BSM address + 2 cannot expand due to lack of memory space.

BSM address + 3 for expansion complete
BSM address + 2 for all others

Calling Sequence

For functions 0, 1, 2 and 3

```
LDX     X1,function code
BSM*    ADJMEM
DATA    0/1/2/3
--                          normal return
```

For functions 4 and 5

```
LDX     X5,MACTAB address
LDA     size
BSM*    ADJMEM
DATA    4/5
--                          error return
--                          normal return
```

For functions 6 and 9

```
LDA     size
BSM*    ADJMEM
DATA    6/9
--                          error return
--                          normal return
```

Routines Used

MOVEDN	to pack the memory
MOVEUP	to expand the memory
FGBGH	to wait for foreground or background activity completion

Registers Used

A and E registers are not restored.

Allowed Usage

Background only. Foreground may call this routine on special condition, but this feature is reserved for the operating system.

3.11 SCNFIL

Reference Location

SYXVEC + 10

Description

This routine scans MACTAB to find the requested file, or output test program or overlay names on POD.

The function performed depends on the code supplied in X1 register.

Input Parameters

X1 Function code

0	Search for any file with the name in A and E
1	Search for a test program with the name in A and E
2	Search for an overlay with the name in A and E
3	Search for a system job overlay with overlay code in A
4	List all test programs in memory
5	List all overlays in memory
6	List all files in memory
7	List one file with X5 pointing to MACTAB
8	LIST JOB

For functions 0, 1 and 2, A and E registers must contain maximum six-character file name left justified.

For function 3, the overlay code (13 to 40B) must be in A register.

For functions 0, 1, 2, and 3, the BSM address +1 must contain the job name. (If zero, the current job is used).

For function 7, X5 must point to the file entry in MACTAB.

Output Parameters

For functions 0, 1, 2, and 3 upon normal return:

X5 address of the file entry in MACTAB
X7 starting address of the file in memory

None for all other functions

Return

For functions 0, 1, 2, and 3:
BSM address +2 not found return
BSM address +3 normal return

For all others:
BSM address +1

Calling Sequence

For functions 0, 1, 2, and 3

```
        LDX     X1,function code
        BSM*    SCNFIL
JOB     DATA    0               Job number
        BRU     ERROR           not found return
        --                      found return
```

Routines Used

$IOCS to output names for functions 7, 4, 5, and 6
PUTD to place values in output buffer

Registers Used

A and E are not restored for functions 0, 1, 2, and 3. No register is restored for functions 4, 5, 6, and 7.

Allowed Usage

Functions 0, 1, 2, and 3 are allowed for foreground and background.

Functions 4, 5, 6, and 7 are allowed for background only.

3.12 GTSTAT

Reference Location

SYXVEC + 11

Description

This subroutine decodes the station identification entered in an operator command into internal usage format.

Input Parameters

STATC (GLOVAR +66) Station identification 1 through 4 which is normally set by the PROCESS routine. If STATC = 0, then DFSTAT (GLOVAR +88) is used as default station.

Output Parameters

A and X6	Logical station number (0 through 3)
X2	Starting address of tester variable table, VKT for that station. On error return, E = 'STAT'.

Return

BSM address +1	station ID not entered in command or no default has been set up by SET STAT command.
BSM address +2	normal return

Calling Sequence

```
BSM*    GTSTAT
BRU     ERROR       NO STATION RETURN
--                  normal return
```

Registers Used

A, E, X2, and X6 are not restored.

Allowed Usage

Foreground/background

3.13 CONV

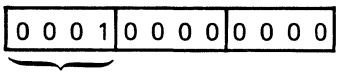
Reference Location

SYXVEC + 12

Description

This subroutine is used to convert a positive number in A register into its decimal equivalent. The decimal digit is represented in a certain number of bits specified by the caller (the digit width). The digit width is specified in the E register.

For example, before calling the subroutine,
A = 144 (octal)
E = 4 [i.e., 4 bits for each decimal digit (BCD)]
Upon exiting from CONV,
A = 0
but,
E =



One Decimal Digit

1 0 0 (BCD)

Input Parameters

A positive octal number
E digit width in bits

Output Parameters

A 0
E converted number in decimal representation

Return

BSM address + 1

Calling Sequence

```
LDA     OCTNUM
LDE     DIGWID
BSM*    CONV
```

Registers Used

A, E, and X1

Allowed Usage

Foreground/background

3.14 PUTD

Reference Location

SYXVEC + 13

Description

This subroutine converts a positive number in the A register into its TRASCII coded decimal equivalent and packs it into a user-specified buffer.

Input Parameters

A positive number to be converted and packed
X1 number of digits wanted (field width)
(Each digit is represented by a TRASCII coded character).
X7 pointer to the three-word DCB
word 0 character count
word 1 buffer address
word 2 buffer size in words

State switch 7 set — do not suppress leading zeros
State switch 7 not set — suppress leading zeros

Output Parameters

A pointer to the next available location in the PMF buffer.

If the number overflows its field (larger than the field width specified), a back slash is stored into the buffer before an exit of the routine.

Word 0 of the DCB points to the next available character in the buffer.

Return

BSM address + 1

Calling Sequence

```
LDX     X1,3
LDA     NUMBER
LDX     X7,PMFDCB
BSM*    PUTD
```

Routine Used

PUTC

Register Used

X1 is not restored.

Allowed Usage

Foreground/background

3.15 PUTC

Reference Location

SYXVEC + 14

Description

This subroutine is used to pack a character in the lower six bits in A register into the buffer specified by the caller in X7.

Input Parameters

A a TRASCII character in the lower six bits
X7 pointer to a three-word DCB
- word 0 character count
- word 1 buffer address
- word 2 buffer size in words

Output Parameters

A 0
0, X7 incremented to next character location in buffer

Return

BSM address +1 0, X7 beyond buffer size
BSM address +2 normal return

Calling Sequence

```
LDA     TRASCII
LDX     X7,PMFDCB
BSM*    PUTC
--                      error return
--                      normal return
```

Register Used

A is not restored.

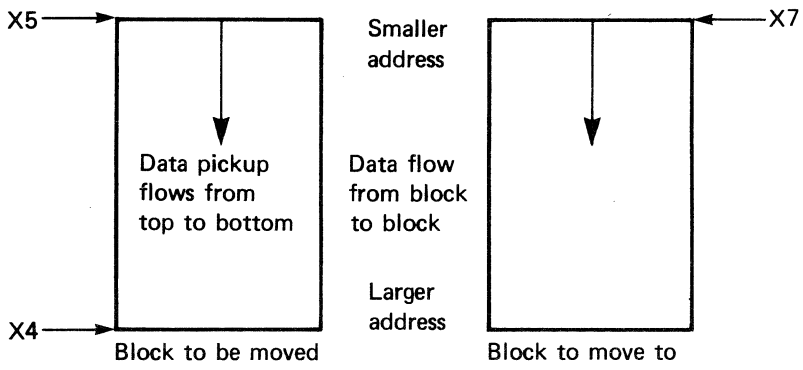
Allowed Usage

Foreground/background

3.16 MOVEDN

Reference Location — SYXVEC + 15

Description — This subroutine moves data blocks from one memory location to another. Data is moved from a block in ascending order, or from a smaller to a larger address in the block.



Input Parameters
- X5 starting address of block to be moved
- X4 ending address of block to be moved
- X7 starting address of block to move into

Output Parameter — X7 last word moved address + 1

Return — BSM address + 1

Calling Sequence

```
LDX     X5,BLKTOP
LDX     X4,BLKBOT
LDX     X7,INTOP
BSM*    MOVEDN
```

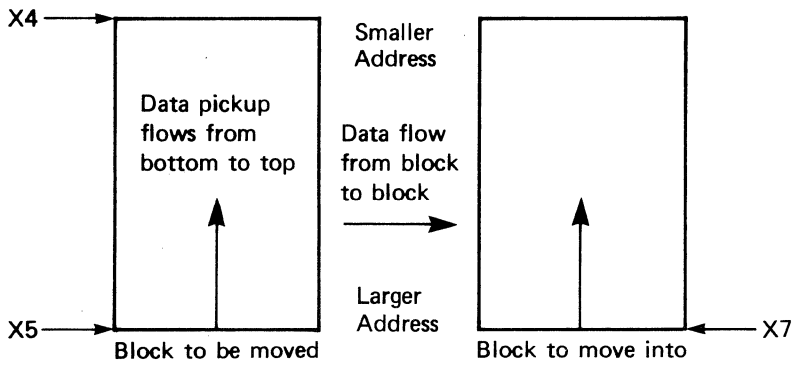
Registers Used — X4, 5, and 7

Allowed Usage — Foreground/background

3.17 MOVEUP

Reference Location: SYXVEC + 16

Description: This subroutine moves data blocks from one memory location to another. Data is moved from a block in descending order, or from a larger to a smaller address in the block.



Input Parameters:

X4	ending address of block to be moved
X5	starting address of block to be moved
X7	starting address of block to move into

Output Parameters: None

Return: BSM address + 1

Calling Sequence:

```
LDX     X5,BLKBOT
LDX     X4,BLKTOP
LDX     X7,INTOP
BSM*    MOVEUP
```

Registers Used: X4, 5, and 7

Allowed Usage: Foreground/background

3.18 PUTE

Reference Location: SYSVEC + 17

Description: This subroutine converts a floating point number to printing format, a four-digit integer, or scientific format and places it in the buffer.

Input Parameters:

A a floating point number
X7 starting address of three-word DCB (used by PUTC, PUTD)

Output Parameters: Converted number in the buffer. If the floating point number is an integer and the magnitude is between -999 and +9999, the format is -nnn to nnnn.

For all other numbers, the format is ±n.nnnE±nn. The decimal point moves left or right in order to make the characteristic a multiple of three.

0 and X7 are adjusted to one character beyond the last used. The output requires 10 columns. If an integer is output the remaining spaces are cleared.

Return: BSM address + 1

Calling Sequence:

```
LDA     NUMBER
LDX     X7,DCB
BSM*    PUTE
```

Routines Used: PUTD, PUTC, CONV, FFIXS

Registers Used: A, E, X1, SW7 are not restored.

Allowed Usage: Foreground/background

3.19 PUTO

Reference Location: SYXVEC +18

Description: This subroutine converts a binary number into octal TRASCII and places it in the buffer, right justified.

Input Parameters:

A positive binary number
X1 number of digits desired
X7 PMF pointer

State switch 7 set	do not suppress leading zeros
State switch 7 not set	suppress leading zeros

Output Parameters — 0 and X7 are adjusted to point to the next character position

Return — BSM address +1

Calling Sequence

```
        DCB     DATA    0, BUF, 20
        BUF     BSS     20

                LDA     NUMBER
                LDX     X1,8            8 digits
                LDX     X7,DCB
                BSM*    PUTO
```

Routine Used — PUTC

Registers Used — A and X1 are not restored

Allowed Usage — Foreground/background

3.20 PROCESS

Reference Location — SYXVEC + 19

Description — This subroutine is used to scan a command statement (represented in a string record) to create a coded 24-bit word representation of the statement. A statement is made up of various fields separated by spaces or other special characters. The codes are supplied by the caller in his keyword table which contains identifiers and corresponding codes for all identifiers in the command statement to be decoded.

PROCESS scans the statement fields, using the table to select the code corresponding to matching identifiers or special options and builds the coded representation from the codes.

The fields of a statement are identified as one of the following:

- Identifier - Starts with an alpha character and up to four alphanumeric characters.
- Name - A string of up to six characters, enclosed by single quotes ('). Quotation marks are not counted as part of the six character name.

- Number - A field with only numeric characters (±nnn), numeric characters with a decimal point (±nn.n), numeric characters followed by B to specify an octal number, numeric characters with exponential notation (±nnE±n), numeric characters followed by * to specify a binary number.
- Identifier Number - Identifier up to four characters is matched with a number (FRQnn). There is no space or special character between the identifier and the number.
- Special Character - A comma (,) is considered as a special option identifier.

Input Parameters

X5 The address of three-word DCB
- word 0 Current word/character count
- word 1 Buffer address word
- word 2 Buffer size

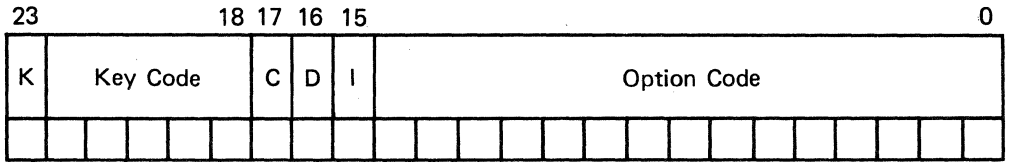
X7 Starting address of the key word table

X6 Ending address (last word of the table +1)

Rules and Restriction of Input:

1. Each entry in the key word table must be two words.

Word 1	four character identifier
Word 2	code for the identifier
Word 2	format



Bit 23	Key word flag
Bit 22 to 18	Key code

All identifiers that belong to the key word contain the same code as the key word.

Bit 17 Complete flag

This flag provides the capability to require a modifier. The statement is complete only if this bit is set. It is set in the code of the key word if no modifier is required; otherwise, it is set in the code of a required option.

At least one identifier must set this bit or PROCESS takes the error return. For example, if either ON or OFF must be specified, the bit is off in the key word code and set in the codes for ON and OFF.

Bit 16 Duplicate flag

This flag provides the capability to disallow the use of two or more options together. If this bit is set by a modifier that is in a statement, PROCESS takes the error return if another modifier is scanned that also has this bit set. For example, to cause an error if both OFF and ON are entered in a statement, set the duplicate bit in the codes for OFF and ON.

Bits 14 to 0 Options

These bits are defined by the user to identify the optional identifiers By checking these bits the user can determine which modifiers were entered.

Bit 15 Identifier number concatenated from flag (see 6 below).

2. The first field in the statement must be a key word. Any other occurrences of key words are ignored.

3. A maximum of two number fields in a statement are saved.

4. A maximum of two string names in a statement are saved.

5. A maximum of six identifier-number fields are allowed in a statement.

6. Codes provided for the identifiers that appear in a statement in the identifier-number form must be a value in the range 1 to 6 in bits 5 to 0 and bit 15 is set to 1. These may not be key words and the value is not ORed into the final code. The value 1-6 is used to store the associated number. Bits 23 to 18 should conform to the key code.

7. No more than three input device and three output device mnemonics may appear in a statement.

8. Normally, noise words are not allowed in a command. ECHFLG (GLOVAR + 40) is used to control this condition.

9. If the STATn is allowed in the command, it must be entered in the table with the code, XX100000B; -f universal, XX is zero; otherwise, XX should conform to the key code.

10. Only one binary number is saved.

Output Parameters

A Final coded representation of the statement upon error return

E The graphics of the error

'PARM' Invalid parameter usage
'COMM' Duplicate key word or missing parameters
'NUMB' Number error

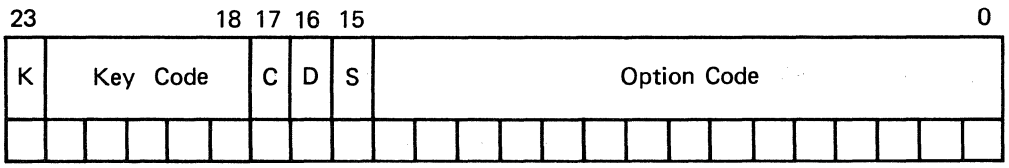
X5 MACTAB pointer if an overlay is loaded

X7 Start address of file if overlays are loaded

Description of Output:

1. The final code is formed by ORing the codes of identifiers appeared in the statement except identifier-number codes.

   Final code format:



Bit	Description
23	Key word flag (from the identifier in the first field)
22 to 18	From the key identifier and subset of key identifier, if any
17	Complete statement
16	Duplicate flag
15	One or more string names have appeared in the statement. The first four characters of the first name are stored in NAMEM1 global and the last two characters in NAMEM2 global. The second name is stored in NAMEM3 and NAMEM4. Only the first six characters of each name are saved.
14 to 0	Codes provided for optional identifiers are ORed together

2. If one or more numbers have appeared in the statement, decimal values are saved in NUMB1 and NUMB2 in floating point format in the order entered (twos complement if negative). Numbers are also saved in ONUMB1, ONUMB2 in octal form. Fixed decimal representations of numbers are saved in INUMB1, INUMB2. If a digit 8 or 9 has appeared in a number field, global OFLERR is set to 1. NUMB1, NUMB2, INUMB1, INUMB2, ONUMB1, ONUMB2 and BINARY are initialized to -1. A binary number is saved in BINARY with its digit count in BINCNT.

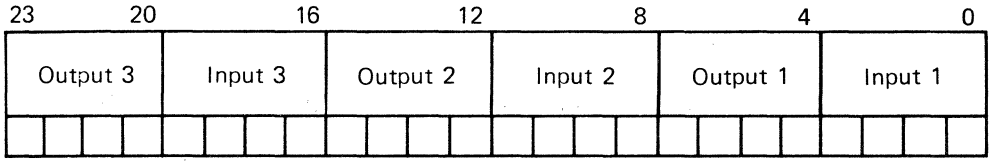
   Examples:

Input	INUMB1	INUMB2	ONUMB1	ONUMB2	OFLERR
0101*	5	-1	101	-1	0
101	145B	-1	101	-1	0
101B	101B	-1	101	-1	0
81	121B	-1	101	-1	1
10B 10	10B	12B	10	10	0
20 0101*	24B	5B	20	101	0
0101* 20	5B	24B	101	20	0

Input	BINARY	BINCNT	NUMB1		NUMB2	
0101*	5	4	20720000B	(FP5)	-1	
101	-1	0	21745000B	(FP101)	-1	
101B	-1	0	21701000B	(FP101B)	-1	
81	-1	0	21721000B	(FP81)	-1	
10B 10	-1	0	21100000B	(FP8)	21120000B	(FP10)
20 0101*	5	4	21320000B	(FP20)	20720000B	(FP5)
0101* 20	5	4	20720000B	(FP5)	21320000B	(FP20)

3. If one or more identifier-number fields have appeared in the statement, numbers are stored in globals SPNUM1 through SPNUM6 according to the code provided for the identifier. SPNUM1 through SPNUM6 are -1 if no identifier number is entered.

4. If I/O device mnemonics appear in the statement, then a pre-defined device code is stored for each device mnemonic in the same order as they appear in the statement stored in global CMDV. The device codes are in the following format:



Code	Input	Output
1	TTK	TTP
2	MTR1	MTW1
3	MTR2	MTW2
4	CR	LP
5	DIF	DOF
6	CLI	CLO
7	VK2	VP2
8	MIF	MOF

5. If a STATn appears in the statement, the station identification number is stored in global STATC in binary, (STAT1 then STATC = binary 1. STATC is initialized to -1).

6. If a comma appears, bit 14 of SPOPT is set to 1. If a plus sign appears, bit 1 is set to 1. If a minus sign has appears, bit 0 is set to 1.

Return

BSM address +1 error return with E = error code
BSM address +2 normal return with A = final code

Calling Sequence

```
LDX     X5,DCB addr
LDX     X7,table start addr
LDX     X6,table end addr
BSM*    PROCESS
BRU     ERROR                   error return
--                              normal return
```

Example:

```
TABSTR      EQU     *
            DATA    'COPY', 41400000B   KEY IDENTIFIER
            DATA    'ALL',  01000100B   SUB-OPTION IDENTIFIER
            DATA    'OFF',  01000200B   SUB-OPTION IDENTIFIER
            DATA    'FRQ',  01100001B   IDENT-NUMBER FIELD
TABEND      EQU     *

PMFDCB      DATA    0                   CURRENT WORD COUNT
            DATA    BUF                 BUFFER ADDRESS
            DATA    20                  BUFFER SIZE
BUF         BSS     20


            LDX     X5,PMFDCB
            LDX     X6,TABEND
            LDX     X7,TABSTR
            BSM*    PROCESS
            BRU     ERROR
            STA     COMING              SAVE A AT NORMAL RETURN
```

Result:

Command Entered			Code Formed	
COPY			41400000B	
COPY	ALL		41400100B	
COPY	OFF		41400200B	
COPY	ALL	FRQ6	41400100B	SPNUM1 = 6

3.21 ALTER

Reference Location: SYXVEC + 20

Description: This routine scans the ALTER buffer to clear all entries for a station, replace or make new entry, find a particular entry, and list all entries for a station.

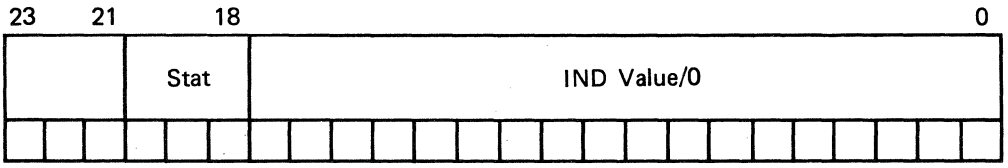
Input Parameters: BSM address +1 contains option number

Option number

0 Clear all entries for a station and pack
1 Replace if an entry for the instruction exists; otherwise, make a new entry
2 Find an entry for the instruction
3 List all entries for a station
4 Clear one entry for the instruction

A Station number in bits 20 through 18 right justified for options 0 and 3

Station number in bits 20 through 18 IND value in bits 17 through 0 for options 1, 2 and 4



STAT 0 = STAT1
1 = STAT2

Output Parameters:

E Floating point value for option 1
A Floating point value for option 2
None for all other options

Return:

BSM address +3 normal return
BSM address +2 not found for option 2
alter buffer full for option 1

Routines Used: $PARSE, GETC, IDTSCN, SEARCH

Registers Used: X1, X5, X6, X7, A, and E registers
X1 is restored

Allowed usage — System use only

Routines Used — PUTE, PUTC, $IOCS for option 3 only

Calling Sequence

Options 0 and 3

```
LDA     station number
SL      18
BSM*    ALTER
DATA    0 (or 3)
NOP     *
```

Option 1

```
LDA     station number
SL      18
OR      IND value
LDE     floating pt. val.
BSM*    ALTER
DATA    1
BRU     ERROR           OVERFLOW
--                      normal return
```

Option 2

```
LDA     station number
SL      18
OR      IND value
BSM*    ALTER
DATA    2
BRU                     NO ALTER VALUE
--                      YES ALTER HERE
```

Registers Used — A, X6, and E are not restored for options 0, 1, 2 and 4. A, E, X1, X2, X3, X6 and X7 are not used and not restored for option 3.

Allowed Usage — Options 0, 1, 2 and 4 are allowed for foreground and background for system use only.

Option 3 is allowed for background for system use only.

3.22 $PARSE

Reference Location: SYXVEC + 21

Description: This subroutine is used to scan and identify a field in the caller's buffer as follows:

1. Identifier - a string of alphanumeric characters with an alpha leading character. It is terminated by either a maximum of eight characters or by a special character (a space is treated as a special character). It is packed (left justified) into two global variables NAME1 and NAME2. However, upon exiting from this subroutine, X6 contains the location of NAME1 in memory.

2. Number - a string of numeric characters (TRASCII coded) is converted into a floating point number and stored in the E register upon exiting from this subroutine. When a '-' sign is encountered, a 15B is returned in A. The caller must make provision for this in processing negative numbers.

3. Special character - If the character is other than a space or a period, which are 0B and 16B respectively in TRASCII code, it is stored in the A register before exiting from this subroutine. A space is ignored when scanning for the start of a field, but also terminates the field when processing it. A period is taken as a decimal point when processing a number field.

Input Parameters:

X7	address of the three-word DCB	
	word 1	character count
	word 2	buffer address
	word 3	buffer size in words

Output Parameters: The A register can be one of the following codes:

1.	0	error in a number scan
2.	1 thru 77B	the TRASCII code of a special character encountered
3.	100B	an identifier is encountered and packed into NAME1 and NAME2
	X6	location of NAME1 in memory
4.	101B	a number is encountered and converted into a floating point equivalent (stored in E register) and an octal equivalent (stored in OCTAL i.e. $10=8_{10}$).
5.	177B	end of record - end of buffer

Return — BSM address + 1

Calling Sequence

```
LDX     X7,PMFDCB
BSM*    $PARSE
```

Routines Used — GETC, IDTSCN, NUMBER

Registers Used — A, E, X6 are not restored.

Allowed Usage — Background only.

3.23 IDTSCN

Reference Location — SYXVEC + 22

Description — This subroutine is used to scan the caller's buffer for an identifier or a string field and packs it (left justified) into global variables NAME1 and NAME2.

Prerequisites:

1. The caller must have located the first character of either an identifier or a string field.

2. The caller should distinguish between packing an identifier (setting the OV, overflow indicator) or a string input (resetting the OV).

Explanation:

When this subroutine is packing a string input, the subroutine checks for whichever of the following terminators comes first:

- a prime mark -'
- an end-of-record mark -177B
- exceeds eight characters in the string, at which time the subroutine continues scanning the buffer for either of the above, but there is no more packing into NAME1 and NAME2.

When this subroutine is packing an identifier, the subroutine stops scanning when it detects any special character.

In all cases, only the first eight characters maximum are packed (left justified) into NAME1 and NAME2.

Input Parameters

OV set for identifier scan
reset for string scan
A first character
X7 pointer to three-word D

Output Parameters

X6 address of NAME1 in memory.
The globals NAME1 and NAME2 contain eight characters of the name.

Return

BSM address +1

Calling Sequence

```
LDA     CHARS
LDX     X7,PMFDCB
SST     OV              for identifier scan
BSM*    IDTSCN
LDA     CHARS
LDX     X7,PMFDCB
RST     OV              for string scan
BSM*    IDTSCN
```

Routine Used

GETC

Registers Used

A, E, X7, and X6

Allowed Usage

Background only

3.24 NUMBER

Reference Location

SYXVEC + 23

Description

This subroutine scans a number in one of the following forms in the input buffer:

1. Octal integer nnn, nnnB
2. Decimal number nn.nn, nn
3. Exponential number nnE±nn
4. Binary number nnnnnnnnnn*

Input Parameter

X7 starting address of the PMF control block 0, X7 must contain the character pointer pointing to the beginning of the number field (the most significant digit or - sign).

Output Parameters

A = 0	number syntax error
A = 100B	number has been converted
E	number in floating point
OFLERR	(GLOVAR + 46) = non-zero if a digit 8 or 9 has appeared in the field
OCTAL	(GLOVAR + 45) contains octal representation of the number

	0, X7	is updated to point to the next character (number terminating character)
	BINC	(GLOVAR + 52) contains binary representation of the number
	BINCNT	(GLOVAR +53) contains the number of digits appeared in the representation of the number
	MANTISSA	(GLOVAR +48) contains fixed decimal form of the number

Return — BSM address + 1

Calling Sequence

```
LDX     X7,PMFDCB
BSM*    NUMBER
```

Routines Used — GETC, INTSCN

Registers Used — A, E, X6 are not restored

Allowed Usage — Background only

3.25 INTSCN

Reference Location — SYXVEC + 24

Description — This subroutine is used to scan the number field in the buffer for an integer and to convert it into an octal number and a decimal number. The two conversions are done simultaneously and both values are available to the caller upon exiting from this subroutine.

Input Parameters

X7	pointer to three-word DCB
X6	pointer to the location which is to store the result of the decimal conversion

Output Parameters

A	last character obtained from the buffer (the first non-numeric character obtained in the scanning of the number)
E	digit count of the number obtained
X6	pointer to the location containing the decimal equivalent
0,X7	has been updated just past the number field
OCTAL	(global variable) contains the octal equivalent

Return — BSM address + 1

Calling Sequence

```
LDA     X7,PMFDCB
LDX     X6,DECNUM
BSM*    INTSCN
```

Routine Used: GETC

Registers Used: A, E, X6 are not restored

Allowed Usage: Background only

3.26 SEARCH

Reference Location: SYXVEC + 25

Description: This subroutine is used to search through a table supplied by the caller for a name specified in NAME1 and NAME2. NAME1 and NAME2 are normally set up by the $PARSE subroutine before this subroutine is called.

Input Parameters:

X6	ending address +1 of the table to be searched
X7	starting address of the table to be searched
A	number of words per entry in the table supplied. It is used as a bias to step through the table from item to item. The minimum value is two.

Output Parameter: X7 address of the item found in the table.

Return:

BSM address + 1	item found in table
BSM address + 2	item not found in table

Calling Sequence

```
        LDX     X6,TABEND
        LDX     X7,TABSTR
        LDA     ITMSIZ
        BSM*    SEARCH
         .
         .
         .
         .
         .
TABSTR  EQU     *
        BSS     200B
TABEND  EQU     *
ITMSIZ  EQU     4B
```

Registers Used: X6, X7, A, E are not restored

Indicator Used — OV (overflow) is undefined on return.

Allowed Usage — Foreground/background

3.27 MPZERO

Reference Location — SYXVEC + 26

Description — This subroutine is used to clear memory (move zeros into memory locations) within specified limits.

Input Parameters —
X6 ending address of the memory to be cleared (the larger of the limits).
X7 starting address of the memory to be cleared (the smaller of the limits).

Output Parameter — A = 0

Return — BSM address + 1

Calling Sequence

```
        LDX     X6,BUFEND
        LDX     X7,BUFSTR
        BSM*    MPZERO      for external
                            programs/subroutines (indirect
                            calling through system trans-
                            fer vector)

BUFSTR  EQU     *
        BSS     200B
BUFEND  EQU     *-1
```

Registers Used — X6, X7, A, E are not restored

Allowed Usage — Foreground/background

3.28 GETC

Reference Location — SYXVEC + 27

Description — This subroutine gets a character from the buffer specified by the caller and decodes it into one of the following:

- alpha
- numeric
- special character

Input Parameters

X7 pointer to three-word DCB
- word 1 — character count
- word 2 — buffer address
- word 3 — buffer size in words

Output Parameters

A character from buffer

Return

BSM address + 1	character is alpha
BSM address + 2	character is numeric
BSM address + 3	character is special, including the 177B (end of buffer)

Note that $, the dollar sign, is regarded as an alpha character. The character returned is in TRASCII code (right justified).

Calling Sequence

```
LDX     X7,PMFDCB
BSM*    GETC
```

Routine Used

READW

Register Used

A

Allowed Usage

Foreground/background

3.29 READW

Reference Location

SYXVEC + 28

Description

This subroutine is used to read a word from the specified buffer in memory.

Input Parameters

X7 pointer three-word DCB
- word 0 — character count
- word 1 — buffer address
- word 2 — buffer size in words

A relative address of the word in the buffer

Output Parameters

A the content of the word (if the location of the word is still within the buffer limit)

X7 unchanged

Return

BSM address +1	if location of word is out of bounds
BSM address +2	if still in bounds

Calling Sequence

```
LDX     X7,PMFDCB
LDA     0,X7          get character count
SR      2             get the relative word address
BSM*    READW
```

Register Used — A is not restored.

Allowed Usage — Foreground/background

3.30 WRITEW

Reference Location — SYXVEC + 29

Description — This subroutine is used to write a value from the E register into a caller-specified buffer in memory.

Input Parameters

X7 pointer to the three-word DCB
A relative address of the word in the buffer to receive the value
E value to be written into the buffer

Output Parameters

X7 = A absolute address of the location in the buffer that was just written into (if it does not exceed the boundary of the buffer itself)

Return

BSM address +1, if write out of bounds
BSM address + 2, if write within bounds

Calling Sequence

```
LDX     X7,PMFDCB
LDA     0,X7          get character count
SR      2             get relative address of word
LDE     VALUE
BSM*    WRITEW
--                    error return
--                    normal return
```

Register Used — A is not restored.

Allowed Usage — Foreground/background

3.31 IERMSG

Reference Location	SYXVEC + 30
Description	Routine in test head driver (THD) to handle terminal errors
Input Parameter	A terminal error number
Output Parameters	None
Return	No return. THD completes testing and continues to next station
Calling Sequence	

```
LDA     TENUM       terminal number
BSM*    IERMSG
```

Registers Used	Not applicable
Allowed Usage	Foreground only

3.32 DUMP

Reference Location	SYXVEC + 31
Description	This routine dumps an overlay, a test plan, or a module to a storage medium, magnetic tape, disk or Integrator. The file to be dumped must be in memory.
Input Parameters	X5 points to MACTAB CMDV (bits 7 to 4) contains the output device code. If these bits are zero, the DFDV device is used.
Output Parameters	A error code for error return A If positive, IOCS error code A If negative, PARM if illegal device NAME if the file is partial or not in memory
Return	BSM address + 1 error return BSM address + 2 normal return
Routine Used	$IOCS

Calling Sequence

```
BSM*    DUMP
--                          error return
--                          normal return
```

Registers Used — A, E, X1, X6 are not restored.

Allowed Usage — Background only

3.33 PUTIME

Reference Location — SYXVEC + 32

Description — This routine converts the current time into HH:MM format and places it in the buffer following the date.

Input Parameters

DATE	(GLOVAR + 29, 30) contains the current date
TIME	(GLOVAR + 31) contains the current time in seconds
X7	starting address of three-word DCB (used by PUTD, PUTC)

Output Parameters — Date and time are placed in the buffer

Format:

MM/DD/YY HH:MM

where MM/DD/YY is an eight-character date entered by the command: DATE. It is output in the format entered in the command.

HH	hours in 01 through 23
MM	minutes in 00 through 59
YY	2 spaces

0, X7 is adjusted to one character beyond the last used.

Return — BSM address + 1

Calling Sequence

```
LDX     X7,DCB
BSM*    PUTIME
```

Routine Used — PUTD, PUTC

Registers Used — A, E, X1 are not restored.

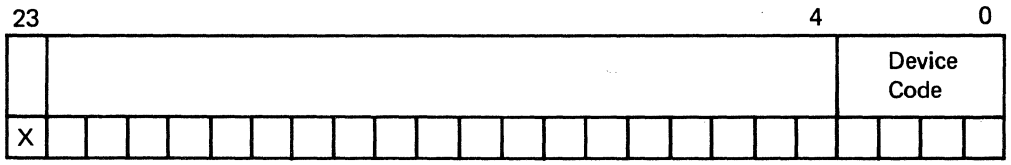
Allowed Usage — Foreground/background

3.34 GTTDV

Reference Location: SYXVEC + 33

Description: This subroutine calculates the address of the device entry in TVT table for the station and obtains the IOATAB pointer to be used for IO for the station.

Input Parameters: A device code in the following format:



Device Code	Output	Input
0	POD	PID of the station
1	TTP	TTK
2	MTW1	MTR1
3	MTW2	MTR2
4	LP	CR
5	DOF	DIF
6	CLO	-
7	VK2	VP2
8	MIF	MOF

Only one device, either input or output, can be requested at one time.

Bit 23 1 for input, 0 for output
X2 address of TVT table for the station

Output Parameters:

X7 address of the device entry in TVT table for the station

A contents of the device entry (see the description of TPID, TMTRI, or TDOF.)

Return:

BSM address + 1 error return (invalid device)
BSM address + 2 normal return

Calling Sequence:

```
LDA     device
BSM*    GTTDV
```

Registers Used	A, E, X7 are not restored.
Allowed Usage	Foreground/background

3.35 HEADER

Reference Location	SYXVEC + 34
Description:	This subroutine is used to output a standard header for the data accumulation overlays, the station number, test plan, name and serial number.
Input Parameters	A TSN (serial number) E station number (0 to 3) X6 IOATAB pointer for the output device. Output device must have been opened prior to calling this routine.
Output Parameters	A header line is output to the device. TOF is issued if the device is a line printer.
Return	BSM address + 1
Calling Sequence	

```
LDA     TSN,TP
LDE     station
LDX*    X6,IOPT
BSM*    HEADER
```

Routine Used	$IOCS
Registers Used	A, E, and X7 are not restored. State switch 7 is used.
Allowed Usage	Foreground

3.36 SPIOER

Reference Location	SYXVEC + 35
Description	This subroutine checks error code returned from $IOCS. If the device is busy, it returns to BSM address-2 for retry; otherwise, it outputs a message on the VKT and returns to the monitor.
Input Parameter	A $IOCS error code
Output Parameters	None
Return	BSM address-2 if the device is busy (A = 7). Otherwise, no return.

Calling Sequence

```
LDX     X1,DCB
BSM*    $IOCS
BSM*    SPIOER          $IOCS error return point
```

Routine Used — ERRCNV

Register Used — A is not restored.

Allowed Usage — Background only

3.37 FGOVC

Reference Location — SYXVEC + 36

Description — This routine calls system overlays at foreground entry point. Reserved for operating system use only. Not reentrant.

Input Parameter — A assigned system overlay code

Output Parameters — None

Return

BSM address + 1 overlay not found
BSM address + 2 normal return

Calling Sequence

```
LDA     overlay code
BSM*    FGOVC
--                      error return
--                      normal return
```

Routine Used — None

Allowed Usage — Foreground only

3.38 ALLEX

Reference Location — SYXVEC + 37

Description — This subroutine is used to execute foreground portion of ALLINK overlay. If the called overlay is not in memory, it is loaded before executing it from the foreground entry point.

Input Parameters — A and E register must contain the six-character name of the overlay left justified.

Output Parameters — None

Return

BSM address +1 error return

There is no room in memory to load, or overlay cannot be found on disk

BSM address +2 execution complete

Calling Sequence

```
LDA     name1
LDE     name2
BSM*    ALLEX
BRU     ERROR       error return
--                  normal return
```

Routine Used

SCNFIL

Registers Used

All registers and state switches are saved and restored.

Allowed Usage

Foreground only

3.39 COMMND

Reference Location

SYXVEC + 38

Description

Allows return to the monitor after completing a process. COMMND can be used for a temporary return while waiting to be called by other programs (DEBUG waiting for an address halt to occur; return from background after scheduled by foreground). It is not normal termination of a foreground or background task and may not be used to indicate the completion of a process.

Input Parameters

None

Output Parameters

None

Return

No return

Calling Sequence

```
BRU*    COMMND
```

Registers Used

Not applicable

Allowed Usage

Foreground/background

3.40 PUTW

Reference Location: SYXVEC + 39

Description: This subroutine is used to pack a word in A register into the buffer specified by the caller. The word should be of four TRASCII characters.

Input Parameters:

A	four TRASCII characters
X7	starting address of the three-word DCB

Output Parameters: None

Return:

BSM address + 1	for buffer overflow
BSM address + 2	for normal return

Calling Sequence

```
LDA     CHARS
LDX     X7,PMFDCB
BSM*    PUTW
--                          error return
--                          normal return
```

Routine Used: PUTC

Registers Used: X0, X1, and A are not restored.

Allowed Usage: Foreground/background

3.41 TWAIT

Reference Location: SYXVEC + 40

Description: This subroutine waits for the foreground activity to complete before returning control to the caller.

Input Parameters: None

Output Parameters: None

Return: BSM address +1

Calling Sequence

```
BSM*    TWAIT
```

Registers Used: All registers and state switches are saved and restored.

Allowed Usage: Background only

3.42 FGBGRT

Reference Location: SYXVEC + 41

Description: This routine schedules the background part of a program from the foreground.

Input Parameter: X7 address of the background process to be activated

Output Parameters: None

Return: BSM address +1

Calling Sequence

```
            LDX     X7,BGADR
            BSM*    FGBGRT
             .
             .
             .
BGADR       EQU     *               start of background process
```

Registers Used: All registers are restored.

Allowed Usage: Foreground only

Note

When the background is entered due to scheduling through FGBGRT, all registers are undefined. Refer to paragraph 6.4 for additional information.

3.43 MONINT

Reference Location: SYXVEC + 42

Description: This subroutine allows entering the scheduler loop once, so that the higher priority functions such as testing can regain control. Control returns to the calling program if there is no higher priority function waiting or at the next break of the higher priority function process. It is recommended to call this routine during a long calculation or if formatting of data is involved.

Input Parameters: None

Output Parameters: None

Return: BSM address +1

Calling Sequence

```
BSM*    MONINT
```

Registers Used — All registers and state switches are saved and restored.

Allowed Usage — Background only

3.44 SCALE

Reference Location — SYXVEC + 43

Description — For use by THD only

3.45 FGWAIT

Reference Location — SYXVEC + 44

Description — This routine allows entering the scheduler once from foreground to allow the background to process. It is used by the program that requires handshaking between the foreground and background.

Input Parameters — None. The program should maintain a flag to indicate that background has completed the process and is ready to accept next data.

Output Parameters — None

Return — BSM address + 1

Calling Sequence

```
LDA     flag        BACKGROUND BUSY?
BZ      *+3         NO, READY NOW
BSM*    FGWAIT      YES, WAIT
BRU     *-3
```

Registers Used — All registers and state switches are saved and restored.

Allowed Usage — Foreground only

3.46 ERRCNV

Reference Location: SYXVEC + 45

Description: This subroutine is used to output an $IOCS error message to the VKT.

Error code 7 should be detected prior to calling ERRCNV, since it is simply a busy condition.

Input Parameter: A register error code

Output Parameters:

Code	Message
0	NONE
1	END OF FILE INPUT
2	DEVICE NOT AVAILABLE
3	INVALID FUNCTION
4	ERROR IN FILE - DATA
5	FILE NOT FOUND
6	I/O ERROR
7	DEVICE BUSY
8	INVALID I/O TABLE
9	DATA OVERFLOW
10	NO WRITE RING
11	CLIO ERROR
12	I/O TABLE OVERFLOW
13	EXCESS WORDS IN READ
14	WS NOT AVAILABLE
15	MACTAB OVERFLOW
16	DUPLICATE FILE
777	INTEGRATOR ERROR

Return:

BSM address +1 error return
BSM address +2 normal return

Calling Sequence

```
        BSM*    $IOCS
        BRU     ERROR

ERROR   BSM*    ERRCNV
        NOP     *               Ignore Error
```

Registers Used: A, E, X1, and X6 are not restored.

Allowed Usage: Background/foreground

3.47 FSUB

Reference Location: SYXVEC + 46

Description: This routine subtracts a signed floating point value in E from a signed floating point value in A and returns the difference in A in floating point.

Input Parameters:
A floating point number to be subtracted from (minuend)
E floating point number to subtract (subtrahend)

Output Parameter: A signed difference in floating point (remainder)

Return: BSM address + 1

Calling Sequence:

```
LDA     VAL1
LDE     VAL2            (VAL1 - VAL2)
BSM*    FSUB
```

Registers Used: A and E are not restored.

Allowed Usage: Foreground/background

3.48 FAND

Reference Location: SYXVEC + 47

Description: This routine applies logical AND operation to A and E and returns the result in A. A and E are fixed before logical AND is applied, hence, the values must be integers in floating point format, (no decimal fractions in A or E).

Input Parameters: A and E contain integers in floating point format.

Output Parameter: A contains the result in floating point format.

Return: BSM address + 1

Calling Sequence:

```
LDA     VAL1
LDE     VAL2
BSM*    FAND
```

Routines Used: FFIX, FFLT

Registers Used: A and E are not restored.

Allowed Usage: Foreground/background

3.49 FEOR

Reference Location: SYXVEC + 48

Description: This routine applies exclusive OR to A and E and returns the result in A. A and E are fixed before exclusive OR is applied, hence, the values must be integers in floating point format.

Input Parameters: A and E contain integers in floating point format.

Output Parameter: A contains the result in floating point format.

Return: BSM address + 1

Calling Sequence:

```
LDA     VAL1
LDE     VAL2
BSM*    FEOR
```

Routines Used: FFIX, FFLT

Registers Used: A and E are not restored.

Allowed Usage: Foreground/background

3.50 FLOG

Reference Location: SYXVEC + 49

Description: This routine converts the floating point value in A into base 2 logarithm and returns the result in A.

Input Parameter: A contains a positive floating point value.

Output Parameters: A contains base 2 logarithm value in floating point format. OV indicator is set if the input is negative or zero.

Return: BSM address + 1

Calling Sequence:

```
LDA     VAL
BSM*    FLOG
```

Registers Used: A and E are not restored.

Allowed Usage: Foreground/background

3.51 FADD

Reference Location: SYXVEC + 50

Description: This routine adds signed floating point values in A and E and returns the sum in A in floating point.

Input Parameters: A and E floating point values.

Output Parameter: A the sum.

Overflow indicator is set if the overflow condition occurs.

Return: BSM address + 1

Calling Sequence

```
LDA     VAL1
LDE     VAL2
BSM*    FADD
```

Registers Used: A and E are not restored.

Allowed Usage: Foreground/background

3.52 FDIV

Reference Location: SYXVEC + 51

Description: This routine divides the floating point value in A by the floating point value in E and returns the quotient in A in floating point.

Input Parameters: A the dividend in floating point.
E the divisor in floating point.

Output Parameters: A the signed quotient in floating point.
Overflow indicator is set for overflow or underflow.

Return: BSM address +1

Calling Sequence

```
LDA     VAL1
LDE     VAL2            (VAL1/VAL2)
BSM*    FDIV
```

Registers Used: A and E are not restored.

Allowed Usage: Foreground/background

3.53 FFIXS

Reference Location: SYXVEC + 52

Description: This subroutine converts a floating point number in the A register into a signed octal integer returned in the A register and the power of 10 multiplier returned in the E register.

Input Parameter:
- A the floating point number to be fixed (converted to an octal integer).

Output Parameters:
- A the signed octal integer equivalent of the floating point number
- E the power of 10 multiplier

Return: BSM address + 1

Calling Sequence:

```
LDA     FPNUM
BSM*    FFIXS
```

Registers Used: A and E are not restored.

Allowed Usage: Foreground/background

3.54 FOR

Reference Location: SYXVEC + 53

Description: This routine applies logical OR to A and E and returns the result in A. A and E are fixed before OR is applied, hence, the values must be integers in floating point format.

Input Parameters: A and E integers in floating point format.

Output Parameters: A the result in floating point format.

Return: BSM address +1

Routine Used: FFIX, FFLT

Calling Sequence:

```
LDA     VAL1
LDE     VAL2
BSM*    FOR
```

Registers Used: A and E are not restored.

Allowed Usage: Foreground/background

3.55 FNOT

Reference Location: SYXVEC + 54

Description: This routine applies logical negation to A. A is fixed before negation is applied, hence, the value must be an integer in floating point format.

Input Parameter: A integer in floating point format.

Output Parameter: A negated value in floating point format.

Return: BSM address +1

Routines Used: FFIX, FFLT

Calling Sequence:

```
LDA     VAL
BSM*    FNOT
```

Registers Used: A and E are not restored.

Allowed Usage: Foreground/background

Note

A floating point number may be negated by use of the machine language TCA instruction.

3.56 FEXP

Reference Location: SYXVEC + 55

Description: This routine calculates the value 2^n where n is given in the A register in floating point format.

Input Parameter: A signed exponent in floating point.

Output Parameter: A result in floating point.

Return: BSM address +1

Calling Sequence:

```
LDA     EXP
BSM*    FEXP            CALCULATE2**EXP
```

Registers Used: A and E are not restored.

Allowed Usage: Foreground/background

3.57 FMUL

Reference Location

SYXVEC + 56

Description

This subroutine is called for multiplying two floating point numbers together. The numbers are all 24 bits. The first number is loaded in the A register upon entering this subroutine and the second number is in the E register. If the result of this multiplication is too large (the characteristic is greater than 177 octal), then the overflow indicator (OV) is set upon exiting from the subroutine. If the product is too small (the characteristic is less than 101 octal), the end result is represented by zeros.

The sign of the product follows the algebraic convention (+ x + = +, + x - = -, - x - = +).

Input Parameters

A value (floating point) of first number
E value (floating point) of second number

Output Parameters

A signed product in floating point
Overflow indicator is set if the end result is too big

Return

BSM address + 1

Calling Sequence

```
LDA     FPNUM1
LDE     FPNUM2
BSM*    FMUL
```

Registers Used

A and E registers are not restored.

Allowed Usage

Foreground/background

3.58 FFIX

Reference Location

SYXVEC + 57

Description

This subroutine is used to convert a floating point number in the A register into an octal integer returned in A register.

Input Parameter

A the floating point number to be fixed (i.e., to be converted to an octal integer).

Output Parameters

A the octal integer equivalent.

If the floating point number is so small that its characteristic is less than 101B, then it is truncated as a zero octal integer.

Return — BSM address + 1

Calling Sequence

```
LDA     FPNUM
BSM*    FFIX
```

Registers Used — A and E are not restored.

Allowed Usage — Foreground/background

3.59 FFLT

Reference Location — SYXVEC + 58

Description — This subroutine converts an octal number in the A register into a floating point number that is stored in the A register on exit.

Input Parameter — A signed octal integer

Output Parameter — A signed floating point number

Return — BSM address + 1

Calling Sequence

```
LDA     OCTNUM
BSM*    FFLT
```

Registers Used — A and E are not restored.

Allowed Usage — Foreground/background

3.60 FFLTS

Reference Location — SYXVEC + 59

Description — This subroutine converts an octal number in the A register and its power of 10 in the E register into a floating point number that is stored in the A register upon exiting.

Both positive and negative numbers are processed by this subroutine.

Input Parameters — A signed octal integer
E degree of power of 10

Output Parameters

A signed floating point number

If there is an underflow (number is too small), then A equals 0.

If there is an overflow (number is too large), the overflow indicator is set when exiting from this subroutine and the A register does not contain this floating point equivalent.

Return

BSM address + 1

Calling Sequence

```
LDA     OCTNUM
LDE     PWR10
BSM*    FFLTS
```

Registers Used

A and E are not restored.

Allowed Usage

Foreground/background

3.61 FCAM

Reference Location

SYXVEC + 60

Description

This routine compares the floating point value in A to the floating point value in E and returns indicators GT, EQ, LT, and BE.

Input Parameters

A and E registers contain floating point values.

Output Parameters

Indicators are set as follows:

GT set if A > E
E EQ set if A = E
LT set if A < E
BE set if any bit in A set corresponds to the bit set in E.

Return

BSM address + 1

Calling Sequence

```
LDA     VAL1
LDE     VAL2
BSM*    FCAM
```

Register Used

E register is not restored.

Allowed Usage

Foreground/background

Note

This routine maintains compatibility with other programs that use FCAM. The machine language instruction CAM may be used with floating point numbers.

3.62 LOAD

Reference Location: SYXVEC + 61

Description: This routine loads files into memory from the specified IO device and makes entries into MACTAB. It can be called from either foreground or background. If the loading device is a disk and the file is a test program, the file is not loaded into memory unless there is enough room to load completely. However, the entry is still made in MACTAB.

Input Parameters: A and E registers contain six-character file names. BSM address + 1 contains the expected file type.

14B to 40B	system overlay file codes
76B	test program
75B	module test program
74B	string or any file without a header
77B	ALLINK overlay
71B	ALLINK overlay that must remain fixed in
- 1	any one of above type
- 2	75 or 76 and KEEP

BSM address + 2 contains the loading device code.

0	default loading device
2	magnetic tape unit 1
3	magnetic tape unit 2
5	disk
6	CLI

BSM address + 3 contains file expansion size in words.

Output Parameters:

X7	start address of the file loaded in memory
X5	MACTAB pointer where the entry is made
A	error code for error return

A error code from $IOCS if positive
error message if negative:

'PARM'	Illegal loading device code
'SIZE'	Insufficient room in MACTAB
→ 'TYPE'	Wrong file type
'NAME'	File not found

Return:

BSM address + 4 error return
BSM address + 5 normal return

Calling Sequence

```
DLD     file name
BSM*    LOAD
DATA    file type
DATA    device code
DATA    expansion size
                        if -1 or 0, use word 6
                        if  0, use this word
DATA    job number
--                      error return
--                      normal return
```

Registers Used — None restored

Allowed Usage — Foreground/background

3.63 DELFIL

Reference Location — SYXVEC + 62

Description — For system use only.

3.64 RELOV

Reference Location — SYXVEC + 63

Description — For system use only.

3.65 ATTA

Reference Location — SYXVEC + 64

Description — This routine attaches a file to a station or makes it busy. Before the overlay starts collecting data for a station or starts interactive process with a station, ATTA should be called. The attached programs/stations are indicated in the NAME command output. Calling ATTA prevents autoreleasing in a disk-based system. See DTTA for detaching.

Input Parameters

X5 MACTAB pointer of the file to be attached (at background and foreground entry points X5 is set to MACTAB for the overlay). If MACTAB pointer is unknown, call SCNFIL subroutine to locate the pointer.

A logical station ID (0, 1, 2, or 3) to attach program to a station makes overlay busy.

Output Parameters — None

Return — BSM address + 1

Calling Sequence

```
LDX*    X5,SAVX5      SAVX5 contains MACTAB
LDA     STAT          pointer
BSM*    ATTA
```

Register Used — A is not restored.

Allowed Usage — Foreground/background

3.66 DTTA

Reference Location — SYXVEC + 65

Description — This routine detaches a file from a station. See ATTA for attaching.

Input Parameters

X5 MACTAB pointer of the file to be detached (at background and foreground entry points, X5 is already set).

A logical station ID (0, 1, 2, 3) for station related file; 4 to clear busy.

Output Parameters — None

Return — BSM address + 1

Calling Sequence

```
LDX*    X5,SAVX5
LDA     STAT
BSM*    DTTA
```

Register Used — A is not restored.

Allowed Usage — Foreground/background

3.67 PAGTP

Reference Location: SYXVEC + 66

Description: This routine reads a test program into memory in pages during testing. If the test program is currently in memory, the required block is read into the same memory area. If it is not in memory (bumped) due to other programs being released or loaded, it allocates the new test program area by releasing inactive programs and by bumping other test programs before it loads the test program.

Input Parameters:
A current instruction number
X5 the pointer to MACTAB for the test program

Output Parameters: X5 points to MACTAB for test program

Return:
BSM address + 1 normal return
No return if error.

Terminal errors:

12 not enough memory space to page
44 I/O error during paging

Calling Sequence:

```
LXA     X5          X5 = address of MACTAB
LDA     TIP,TP      instruction pointer
BSM*    PAGETP
--                  normal return
```

Routines Used: $IOCS, ADJMEM

Register Used: E is not restored.

Allowed Usage: Foreground only

3.68 FGBGWT

Reference Location: SYXVEC + 67

Description: Reserved for system use.

3.69 FGBGH

Reference Location: SYXVEC + 68

Description: Reserved for system use.

3.70 FINDVL

Reference Location	SYXVEC + 69 Reserved for system use only.
Description	This routine converts a block or variable number of a variable, array, or global into the absolute location of the variable. The address is in the A register on exit. Internal cells are altered for use by ARITH which calls the routine when a statement references a variable. BNO contains the block number, VNO contains the variable number, and VLOC contains the absolute address.
Input Parameter	A contains the block or variable number. Bits 12 through 10 contain the block number, and bits 9 to 0 contain the variable number.
Output Parameter	A contains address of the variable.
Return	BSM address + 1
Calling Sequence	BSM* FINDVL
Routines Used	None
Registers Used	X7 and E registers are not restored.
Allowed Usage	Foreground or background at a pause

3.71 FGIO

Reference Location	SYXVEC + 70
Description	This routine does I/O from foreground. Prior to calling routine, the device must have been opened and set up by calling FGOH. This routine is used to output the data to the device used by DATALOG or FACTOR. If the MON button is depressed, output is suppressed.
Input Parameters	A IOATAB address returned from FGOH call. X1 address of IOCS DCB
Output Parameters	None

Return

BSM address + 1 end of file on input
BSM address + 2 normal return

Error condition:

The control branches to the terminal error processor if an error condition occurs. No return is made to the caller in this case.

Terminal Error

40 I/O error
45 Device is not opened

Calling Sequence

```
LDA     I/O             data from FGOH (BSM + 1)
LDX     X1,DCB
BSM*    FGIO
NOP     *               EOF return
--                      normal return
```

Routines Used

$IOCS, ERRCNV

Registers Used

A and E are not restored,
X1 (SP) is restored to the SVT pointer.

Allowed Usage

Foreground only

3.72 DMASTR

Reference Location

SYXVEC + 71

Description

This routine starts DMA and gives control to the background. All interrupt bits are cleared, saving any enable bits, then the trap (bit 4), fail (bit 6), and reset (bit 12) interrupt enable bits are set in the status register. Bits 11 and 10 of the mode registers are cleared and bit 9 is set to start DMA. WAIT is called to give background control until an interrupt takes place.

Input Parameters

None

Output Parameters

None

Return

BSM address + 1

If tester reset is pressed while background has control, there is no return to the caller.

Calling Sequence

```
BSM*    DMASTR
```

Routines Used — DWAIT is called, and calls WAIT.

Registers Used — The A register is not restored.

Allowed Usage — Foreground

3.73 FGOH

Reference Location — SYXVEC + 72

Description — This subroutine keeps track of I/O device usage by stations. If the device is a disk, magnetic tape, or communication link, the device should open by command: OPEN. It opens all other devices if it has not been used before. It keeps track of the type of output issued by using the header code. When it is the first use of the current I/O since the beginning of the test, the standard station header is output before it returns to the caller.

Input Parameters

A device code in bits 3 to 0.

0	PID/POD of the current station
1	TTK/TTP
2	MTR1/MTW1
3	MTR2/MTW2
4	CR/LP
5	DIF/DOF
6	CLO (CLI is illegal)
7	VP2/VK2
8	MIF/MOF

Bit 23 must be set to 1 for input device.

E current line header code in bits 3 to 0.

0	override output of station header
1	datalog trip
2	datalog measure/DCT
4	datalog FCT
7	datalog PPM memory fail
8	datalog PPM data extension
9	FACTOR write/FACTOR pause
13 to 15	Reserved for ALLINK generated headers

X2 TVT address of the station

Output Parameters

BSM address + 1 contains the device pointer required by FGIO. It is the contents of the device variable in station global table such as TPOD or TLP (see the description in TVT table description).

Bit 22 set to 1 indicates that binary formatting is required for this device.

Return

BSM address + 2

The current output is not the same type as the last one or the output requires binary format.

BSM address + 3

The current output is not the same type as the last one and needs a line header (Datalog FCT after DCT output).

BSM address + 4

The current output is the first output to this device since the beginning of the test, hence, the station header has been output.

Error condition:

If the device is a disk, magnetic tape, or communication link, and it has not been opened by the command OPEN or SET, then the control branches to terminal error 42.

If the device code is invalid, the control branches to terminal error 43.

Calling Sequence

```
        LDA     device code
        LDE     header code
        BSM*    FGOH
IOPT    DATA    0           Device pointer put in here
        --                  No header change return
        --                  Header change return
        --                  First output return
```

Routines Used

GTTDV, FGOPEN, HEADER

Registers Used

A and E are not restored.
X1 (SP) is restored to SVT pointer.

Allowed Usage

Foreground only

3.74 ENBTST

Reference Location: SYXVEC + 73

Description: This routine initiates a functional test by writing the contents of the A register on entry to SAMA. The user should enable the clock timeout interrupt and write the clock register before entry if a timeout is required (MATCH or EXT). This routine then enables the FCT interrupt (set bit 6 SR), and enables the tester busy interrupt (set bit 16, reset bit 17 SR). Then SAMA is written. Generally, WWAIT should be called following the call to ENBTST to pass control to the background.

Input Parameters: A must contain the data to be written to SAMA. Bit 0 must be set to begin testing.

Output Parameters: None

Return: BSM address +1

Calling Sequence:

```
LDA     SAMA data
BSM*    ENBTST
BSM*    WWAIT
```

Routine Used: ENTBSY is called to enable the tester busy interrupt.

Registers Used: A and E registers are not restored.

Allowed Usage: Foreground

3.75 WWAIT

Reference Location: SYXVEC + 74

Description: This routine turns control over to the background until the tester busy flag set by ENBTST or ENTBSY is reset.

Input Parameters: None

Output Parameters: None

Return

BSM address +1

If WWAIT is called by a foreground program and tester reset is pressed, there is no return to the caller. If WWAIT is called by a background program, reset causes control to be returned to the caller. In this case, bit 10 to 13 of RSTTSC (GLOVAR + 106) indicates that reset was pressed. The program should check for reset following a call to WWAIT. In most cases when the station is reset, the background program should exit.

Calling Sequence

```
LDA     SAMA data
BSM*    ENBTST
BSM*    WWAIT
```

Routines Used

DWAIT is called and calls WAIT.

Registers Used

All registers are restored.

Allowed Usage

Foreground/background overlays using the tester

3.76 ADRXLA

Reference Location

SYXVEC + 75

Description

This routine translates the relative address of a word in a file into an absolute address of the word in a memory buffer. If the word is not currently in memory, the current buffer contents are written back to the file (if its contents have been modified) and a new portion of the file is read into the buffer.

The current contents of the buffer can be written back to the file by calling with a relative address of -1.

Input Parameters

X6 I/O assignment table pointer
X7 pointer to three-word DCB for memory buffer
  word 0 0
  word 1 buffer address
  word 2 buffer word count

A relative address of word in file (or -1 to force the current buffer contents to be written back to file)

Output Parameters

A address or word in memory buffer (relative to current relocation register)
X6,X7 unchanged

Return

BSM address +1 on error
A = 0 address error
A = non zero $IOCS error code
BSM address +2 for normal return

Calling Sequence

```
LDX     X6,I/O assignment table pointer
LDX     X7,buffer DCB
LDA     word relative address in file
BSM*    ADRXLA
BRU     ERROR                   error return
--                              normal return
```

Routines Used

$IOCS

Registers Used

A, X6, X7

Allowed Usage

Foreground/background

3.77 INTERP

Reference Location

SYXVEC + 76

Description

This call accesses the THD interpreter for processing a data code in the 500 or 600 series. Any tester function desired that directly corresponds to a FACTOR code may be done by a call to INTERP. Care must be taken to return any system cells used to their previous state. For example, if a measurement is done by a call to INTERP, STHC records the measurement and any pass/fail information. This triggers the datalogger and lights the pass/fail lamps unless STHC is restored to its previous state before an exit of the user overlay. The IND register should be restored by an overlay calling INTERP.

Input Parameters

X1 pointer to SVT table
X2 pointer to TVT table
A opcode in acceptable form

Return

BSM address +1 IND has not been incremented
BSM address +2 IND has been incremented

Calling Sequence

```
LDA     opcode
BSM*    INTERP
NOP     0
--                              return
```

Routines Used	Any interpreter routines
Registers Used	Depends on opcode
Allowed Usage	Foreground

3.78 ENTBSY

Reference Location	SYXVEC + 77
Description	This routine enables the tester busy complete interrupt and turns off the tester busy interrupt bit (SR bit 16 on, bit 17 off). A flag is set to indicate that the interrupt is enabled. The flag is reset by the tester busy interrupt routine. Either before or after the call, some activity must be done to generate a tester busy interrupt. This routine then enables the interrupt. Generally WWAIT is called, following the call to ENTBSY, to wait for completion in the background. If the interrupt has already occurred WWAIT does not pass control to the background, since this would cause a hang condition.
Input Parameters	None
Output Parameters	None
Return	BSM address +1
Calling Sequence	Initiate activity which generates interrupt BSM* ENTBSY BSM* WWAIT
Routines Used	None
Registers Used	The A register is restored.
Allowed Usage	Foreground

3.79 RSOVC

Reference Location	SYXVEC + 78
Description	This routine calls foreground overlay at the reset entry point if an overlay was running at the time the reset was pressed. Reserved for system use.
Input Parameters	None
Output Parameters	None

Return	BSM address + 1
Calling Sequence	
	BSM* RSOVC
Registers Used	All registers and state switches are restored.
Allowed Usage	Foreground only

3.80 STALL

Reference Location	SYXVEC + 79
Description	Reserved for system use.

3.81 UPDATE

Reference Location	SYXVEC + 80
Description	This subroutine creates, assigns, or deletes memory or disk files. Before a file is created, the remaining data in the block is written out to working storage if a block buffer has been specified and the current block pointer in IOATAB is non-zero. The working storage file is closed after the file is created. Word 10 of the IOATAB entry specifies the number of words used. If word 10 of the IOATAB entry is zero, assigning a space rather than creating a file is assumed. The A register must contain the assigning size. The second option changes the file name on disk or in memory so that the file becomes obsolete. The first character of the obsolete file name is changed to an ampersand.
Input Parameters	BSM address + 1 contains the option code. 0 create 1 delete (name change) BSM address + 2 and 3 contain the file name left justified.

For create,

BSM address + 4 contains the file type. Word must never be zero. Bits 23 through 6 are reserved for future expansion.

14B to 30B	system overlays
71B	ALLINK/Overlay
72B	DATA
73B	OBJECT
74B	string
75B	module test
76B	test plans
77B	ALLINK/overlay

If the device is a disk, the file type is mapped to disk file type as follows:

14B to 30B, 77B	coreimage
72B,75B, 76B	data
73B	OBJECT
74B	string

BSM address + 3, for delete option only, contains the device code.

10B	MIF
5B	DIF

X6	IOATAB pointer of WS (create only)
A	number of words to assign for zero word used

Output Parameters

A Contains the $1OCS error code for error return.

2	the device is not the memory or disk or the file is not a working storage
9	insufficient block size or file overflow
8	invalid I/O assignment
14	MACTAB overflow for memory file
16	duplicate file

Return

BSM address + 4	error return
BSM address + 5	normal return

Calling Sequence

For create:

```
        LDX*    X6, IOAPT
        BSM*    UPDATE
        DATA    0
        DATA    name1,name2
        DATA    filetype
        BRU     ERROR
        --                          normal return
        .
        .
ERROR   BSM*    ERRCNV
        NOP     0
```

For delete:

```
        BSM*    UPDATE
        DATA    1
        DATA    name1,name2
        DATA    device code (5B/10B)
        BRU     ERROR
        --                          normal return
        .
        .
ERROR   BSM*    ERCNV
        NOP     0
```

Routines Used — $IOCS, ADJMEM, SCNFIL, STALL

Registers Used — A and E registers are not restored.

Allowed Usage — Foreground/background

3.82 PUTENG

Reference Location — SYXVEC + 81

Description — This subroutine converts a floating point number to printing format, a four-digit integer, engineering format, or scientific format and places it in the buffer.

Input Parameters

A	a floating point number
X7	starting address of three-word DCB (used by PUTC, PUTD)
E	For engineering format, use letter. For scientific format, use 0.

Word 0 of the DCB points to the next available character.

Output Parameters

Converted number in the buffer.

For E = 0:

If the floating point number is an integer and the magnitude is between - 999 and 9999, the format is is - nnn to nnnn.

For all other numbers, the format = ±n.nnnE±nn. The decimal point moves left or right in order to make the characteristic multiple of three.

For E = letter:

If the floating point number is an integer and the magnitude is between -999 and +9999 the format is -nnn±ml to nnnn±ml. Where m is the magnitude and l is the letter entered in the E register.

For all other numbers, the format is ±n.nnn±ml. The decimal moves to the left or right so that the magnitude represents a multiple of three.

The magnitude is represented as follows:

T	Tera	E+12	M	Milli	E-3
G	Giga	E+9	U	Micro	E-6
M	Mega	E+6	N	Nano	E-9
K	Kilo	E+3	P	Pico	E-12
			F	Femto	E-15

Return

BSM address +1

Calling Sequence

```
LDA     FPNUM
LDX     X7, DCB
BSM*    PUTENG
```

Routine Used

PUTD, PUTC, CONV, FFIXS

Registers Used

A, E, X1, SW7 are not restored.

Allowed Usage

Foreground/background

3.83 PUTA

Reference Location: SYXVEC + 82

Description: This routine converts a binary number to either octal or decimal and places it in the output buffer. Either PUTO or PUTD is used. The routine outputs decimal if the system is configured for decimal output by bit 19 of M1NIT set to zero. Leading zeros are suppresed. If M1NIT is set to 1 the output is octal. Leading zeros are not suppressed for octal output. This routine is used throughout the system for outputting local memory addresses.

Input Parameters:

A	positive binary
X1	number of digits
X7	PMF pointer

Output Parameters: 0 and X7 are adjusted to point to the next character position.

Return: BSM address + 1

Calling Sequence:

```
LDX     X1,length
LDA     binary number
LDX     X7,DCB
BSM*    PUTA
```

Routines Used: PUTO or PUTD

Registers Used: A and X1 are not restored.

Allowed Usage: Foreground/background

3.84 BGCHK

Reference Location: SYXVEC + 83

Description: This subroutine checks for background activity. There are two options. The first checks for no activity, by testing from foreground. The other option checks for more than one background task. It allows testing from a background task to see if there is any other background task active.

Input Parameters:

A	0	Check for no background only
	1	Check for one background
	2	Check for no overlay
	3	Check for one overlay only

Output Parameters	None
Return	BSM address + 1 BSM address + 2
Calling Sequence	LDA check code BSM* BGCHK
Routine Used	None
Registers Used	A and E are used and not restored.
Allowed Usage	Foreground/background

3.85 CALLMOD

Reference Location	SYXVEC + 85
Description	This subroutine provides a means to call another overlay module from an overlay module. When the overlay calls a module using this routine, CPU control is transferred to the module at the module entry point after the relocation register is set to the address specified by the calling module. It returns to the calling module when the called module returns control to CALLMOD. The sequence that occurs when the call to CALLMOD is executed starts at the module entry point in the module. First, the parameter following the call is retrieved and the return address is incremented. The address field of the parameter is used as an index to the module linkage table. The true BSM or BRU instruction is extracted from the linkage table and stored in the program execution path to be executed later.
Input Parameters	A register contains the starting address of the called module in absolute address. All other registers and switches are transferred to the calling module as they are.
Output Parameters	All registers and switches returned from the called module are transferred back to the calling module.
Return	Defined by modules.

Calling Sequence

```
        LDA     module address
        BSM*    CALLMOD
```

Routines Used — None

Registers Used — A is undefined on return.

Allowed Usage — Foreground/background

Rules for module programs callable by CALLMOD:

- The standard overlay header must be maintained.
- It is always called at the module entry point.
- Header word 8 is used to store the relocation register of the calling program.
- Header word 9 is used to store the return address of the calling program.
- Return to the calling module must be done through the module entry point.
- Header word 9 may be incremented to control the return address.

3.86 PUTB

Reference Location — SYXVEC + 86

Description — This subroutine converts a binary number to binary TRASCII and places it right justified in the buffer.

Input Parameters — A positive binary number
X1 number of digits desired
X7 DCB pointer

Output Parameters — 0 and X7 are adjusted to point to the next character position.

Return — BSM address + 1

Calling Sequence

```
DCB     DATA    0,BUF,20
BUF     BSS     20
        LDA     binary number
        LDX     X1,8
        LDX     X7,DCB
        BSM*    PUTB
```

Routines Used — A and X1 are not restored.

Allowed Usage — Foreground/Background

3.87 PUTH

Reference Location: SYXVEC + 87

Description: This subroutine converts a binary number into hexadecimal TRASCII and places it in the buffer right justified.

Input Parameters:

A positive binary number
X1 number of digits desired
X7 DCB pointer

SW7 set do not suppress leading zeros
SW7 not set suppress leading zeros

Output Parameters: 0 and X7 are adjusted to point to the next character position.

Return: BSM address + 1

Calling Sequence

```
        DCB     DATA    0,BUF,20
        BUF     BSS     20
                LDA     binary number
                LDX     X1,8
                LDX     X7,DCB
                BSM*    PUTH
```

Routine Used: PUTC

Registers Used: A and X1 are not restored.

Allowed Usage: Foreground/Background

3.88 SAVENV

Reference Location: SYXVEC + 88

Description: Reserved for interrupt system use.

3.89 USVENV

Reference Location: SYXVEC + 89

Description: Reserved for interrupt system use.

4

I/O Control System ($IOCS)

4.1 $IOCS

All I/O functions must be accomplished through $IOCS to preserve foreground/background configuration. $IOCS is closely tied to the task scheduler and is a reentrant routine (it can be called any number of times at any point by any number of tasks).

4.1.1 $IOCS Operation

$IOCS operation is based on the following principles:

- Devices are available to any program on a first-come-first-served basis, except for the keyboard (TTK), which can be overridden by the TTP.
- Devices on the same channel can be active only one at a time. One more request can be accepted while the channel is busy and is waiting for the channel release by the first request.
- Device protection on a channel is on a function basis. This means that a channel is busy while processing a function such as reading a record or rewinding. It does not prohibit other channels from being activated.
- One program may use any number of devices or may use the same device for two or more different purposes. For example, read a file from the disk and write another file on the disk.
- I/O completion is defined as having one of the following conditions fulfilled:

  In an input operation, a terminating character is received.
  In an output operation, the end of buffer is reached.
  A terminating interrupt from a device is obtained.

- The character set used internally by the system is in TRASCII code. $IOCS makes conversion just prior to sending it out or after reading it in if the external data mode is different from the internal mode in normal mode. (See ASCII control mode)

TTK/VK2	ASCII TRASCII
TTP, LP, VP2	TRASCII ASCII
CR	BCD TRASCII
MTR1, MTR2	No conversion
MTW1, MTW2	No conversion
DIF, DOF, MIF, MOF	No conversion
CLI (binary read)	No conversion
CLI (alpha read)	ASCII TRASCII
CLO (binary write)	No conversion
CLO (alpha write)	TRASCII ASCII

- All read or write requests are initiated by OPEN and terminated by CLOSE.
- Data on all devices are read or written sequentially forward from the beginning of the file. Random access on a disk file must be controlled by the user, using the I/O assignment table provided.
- All I/O operations are considered WAIT FOR COMPLETION unless otherwise specified.
- A maximum of two requests on the same channel can be entered. If the third request is the same file as the one pending (same device, same program, same file), the last request overrides the one pending unless otherwise specified.
- If a function requested is meaningless to the device, it is treated as NOP (return as if completed); for example, TOP-OF-FORM on magnetic tape.
- Read from TTK always echoes the character to TTP in normal mode.
- ASCII control mode for VKT:

Read — Data characters are echoed to screen and are packed into buffer in TRASCII (four characters per word). As soon as a control character is sensed, it is placed in the last word of buffer + 1 right justified and control is returned to the caller.

Write — Output is specified by the character count. Characters are stored in ASCII (three characters/word) left justified.

4.1.2 Devices Handled by $IOCS

Device	Mnemonic	Code	Transfer Mode
VKT keyboard/ printer	TTK/TTP	1	Character by character interrupt
Magnetic tape unit 1	MTR1/MTW1	2	DMA
Magnetic tape unit 2	MTR2/MTW2	3	DMA
Card reader	CR	4	DMA
Line printer	LP	4	Block transfer through hardware buffer
Disk file	DIF/DOF	5	DMA
Com link to Integrator	CLI/CLO	6	Character by character link to interrupt
VKT2	VK2/VP2	7	Character by character interrupt
Memory file	MIF/MOF	8	Word by word move

4.1.3 Functions Performed by $IOCS

Code	Function
0	Open (request to reserve the usage of a device)
1	Read/write a record
2	Kill input (TTK/CLI only)/clear VKT screen (foreground and background)
3	Top of form on line printer
4	Unformatted write (write TTP without CR/LF)/file transmit (create to CLO)
5	Skip file on magnetic tape/file end to CLI
6	Rewind magnetic tape/file end (process) to CLO
7	Write EOF on magnetic tape/file end (purge) to CLO
10	Status check on magnetic tape and disk
11	Read without transfer on magnetic tape and disk for verification/ file end (hold) to CLO
12	Close (release the usage of a device)
13	File request to CLI/magnetic tape record skip
14	Control message to CLO/file page from DIF/MIF
15	Disconnect CLI/CLO/VKT screen transmit
16	File transmit (ADD)
17	Reserved

4.1.4 Error Detected by $IOCS

An error code is returned in the A register upon error exit from $IOCS.

Code	Description
1	An end of file has been reached on a read operation
2	An invalid device code is used for the function or device not available
3	An invalid function code is in the DCT
4	Parity error on a read magnetic tape/CLI/DIF read error on card reader
5	File not found on DIF/DOF/CLI
6	Undefined magnetic tape error/unrecoverable CL error
7	Device busy (both active and pending occupied)
8	Invalid I/O assignment table address is used
9	Insufficient block size or file overflow for DOF/MOF
10	No write ring on magnetic tape
11	CLIO error
12	I/O table overflow
13	Excess word count on read
14	Disk working storage already in use
15	MACTAB overflow for MOF
777	Integrator/RTE error nn

4.1.5 Definition of End-of-File

While reading a file, $IOCS recognizes an end of file when the following condition occurs:

Device	Condition
TTK/VK2	// characters in column 1 and 2 and no other character in the record
MTR1/MTR2	Sensing a file mark
CR	// characters in column 1 and 2 and no other character on the card
DIF	Number of words so far reaches the file size specified in the directory
CLI	Sensing a file end message
MIF	Number of words so far reaches the file size specified in MACTAB

4.1.6 Definition of End of Record

While reading a record, $IOCS recognizes an end of record when the following condition occurs.

Device	Condition
TTK/VK2	Sensing a carriage return character
MTR1/MTR2	Sensing a record mark
CR	Sensing a completion interrupt
DIF, MIF	
not blocked	Sensing a completion interrupt
blocked-source	Sensing a 77B character
blocked-variable	Number of words obtained = record size
size-data	Specified in the first record word (Bit 7 to 0)
blocked-fixed	Number of words obtained = record size
size-data	Specified in the directory
CLI	Sensing ETX during message type 3 transfer

4.1.7 General Calling Sequence

All read and write functions must be initiated by an OPEN call and terminated by a CLOSE call. If the OPEN is successful, a pointer to I/O assignment table is returned to the calling program. This pointer must be set to X6 for all read or write in that device.

Three or four other parameters must be passed to $IOCS. The location of the first parameter must be specified in index register X1 and all other parameters must be in consecutive memory locations.

Parameter 1	Contains input/output indication, function, device code, and other control information particular to the function
Parameter 2	Contains the address of the buffer
Parameter 3	Contains the buffer size (number of words to be transferred)
Parameter 4	Contains the job number (OPEN call only)
Parameter 5, 6	Contains the file name (OPEN call only)

Example of general calling sequence:

```
OPNDCB    DATA    XXXXXXXXB     OPEN KEY WORD
          DATA    BLKBUF        BLOCK BUFFER ADDR
          DATA    48            BUFFER SIZE
          DATA    0             CURRENT SIZE
          DATA    0,0           FILE NAME
*
RDCB      DATA    XXXXXXXXB     READ KEY WORD
          DATA    INBUF         RECORD BUFFER ADDR
          DATA    20            RECORD SIZE
*
RCLOSE    DATA    52000000B     CLOSE KEY WORD
          .
          .
          .
          LDX     X1,OPNDCB
          BSM*    $IOCS         OPEN CALL
          BRU     OPERR         ERROR RETURN
          STX     X6,IOPTR      SAVE IOATAB PTR
          .
          .
          .
          LDX*    X6,IOPTR      GET IOATAB PTR
          LDX     X1,RDCB
          BSM*    $IOCS         READ CALL
          BRU     RDERR
          .
          .
          .
          LDX*    X6,IOPTR      GET IOATAB PTR
          LDX     X1,RCLOSE
          BSM*    $IOCS
          BRU     ERROR
```

4.1.8 Mechanism

$IOCS is divided into three parts: initializing I/O request, driving I/O device, and terminating I/O request.

1. Initialization of I/O request. This occurs at the time of OPEN request and the majority of work is internal housekeeping. Unless the device is a communication link or disk, the hardware is not involved at this time. $IOCS assigns the device to the requesting program by entering necessary information into the I/O assignment table and returning the entry pointer to the program to be used for read/write operation.

2. Driving I/O device. This part of $IOCS is divided into three sections: I/O initialization, interrupt service, and I/O completion. Though functions performed in these sections differ depending on the device type, the following functions are common to all devices:

   - Decoding of keywords
   - Stacking the request as pending if the device is not available
   - Setting up the device into a ready state by sending out control functions (for example, an ESCAPE code to clear the VKT interface circuit before reading or writing to it)
   - Turning on the interrupt enable flip-flop on a selected device (PON) and turning on the CPU interrupt system (IEN) after setting up the device for the operation

3. Termination of I/O Request. Completion of I/O function is always detected in an interrupt service. When this condition occurs, a scheduler flag for the channel is set so that the control returns to the end of channel process. If another request is pending for the channel, it is started immediately before returning to the current request.

4.1.9 I/O Device Formats

4.1.9.1 VKT DRIVER

A character is transferred between the CPU and an I/O device through the accumulator bus. Each character transfer is accompanied by an interrupt. The process continues until, on output, the buffer has been exhausted, or on input, the buffer has been filled or a terminating character (carriage return) has been received. On output, a line feed and a carriage return, characters (12B, 15B) are sent out at the end of the buffer transfer.

4.1.9.2 LINE PRINTER

Data characters are written to the printer hardware buffer until the hardware buffer is full. The size of the buffer differs depending on the printer model.

Buffer Size	Model	Columns
20 characters	Data Products	80
24 characters	Data Products	132
132 characters	Centronics/Printronix	132

If the output message exceeds the length of the respective line printer hardware buffer, it can be printed as a multiple hardware buffer dump. An interrupt occurs at the end of a hardware buffer dump (after printing one buffer of characters).

4.1.9.3 MAGNETIC TAPE

A block of data is transferred between the CPU memory buffer and the tape in DMA mode. Between 6 and 16 thousand words can be transferred in one read or write operation. At the end of one transfer, an interrupt occurs. The status of the device can be obtained through the accumulator bus. The record gap is automatically written after each block write. At the end of a file, a file mark is sent as a file terminator. Up to 10 retries are made on parity error.

4.1.9.4 CARD READER

Data characters are transferred in BCD format from the card reader to the CPU memory buffer in DMA mode. One to 20 words can be transferred at one read. At the end of one transfer, an interrupt occurs. The card reader status can be obtained through the accumulator bus.

4.1.9.5 DISK

A block of data is transferred between the CPU memory buffer and disk in DMA mode. A block must be in multiples of 48 words; the maximum is 64 thousand words. At the end of one transfer an interrupt occurs. The disk status can be read through the accumulator bus. Up to 10 retries are made on parity error. Verification is applied immediately after each write operation.

$IOCS provides functions similar to INREC/OUTREC if the user provides a block buffer that is different from the record buffer.

On input — $IOCS reads one or more sectors into the block buffer and transfers to one record at a time at each read request.

On output — At each write request, $IOCS places a record into the block buffer and transfers the block when it becomes full or at CLOSE request.

4.1.9.6 COMMUNICATION LINK

Each record is preceded and followed by line protocol. Each character, once the record is initiated, is accompanied by an interrupt. A character is transferred between the CPU and the I/O channel through the accumulator bus. The process continues until, on output, the buffer has been exhausted, or on input, the buffer has been filled or a terminating character (ETX) has been received. There is a 112-character buffer to accumulate a note message which could be received at any point in time.

4.2 I/O ASSIGNMENT TABLE (IOATAB)

The I/O assignment table consists of 20 entries. Each entry (figure 4-1) contains 13 words, for a total of 260 words. The first six words are reserved for use by system. Word formats are described in table 4-1.

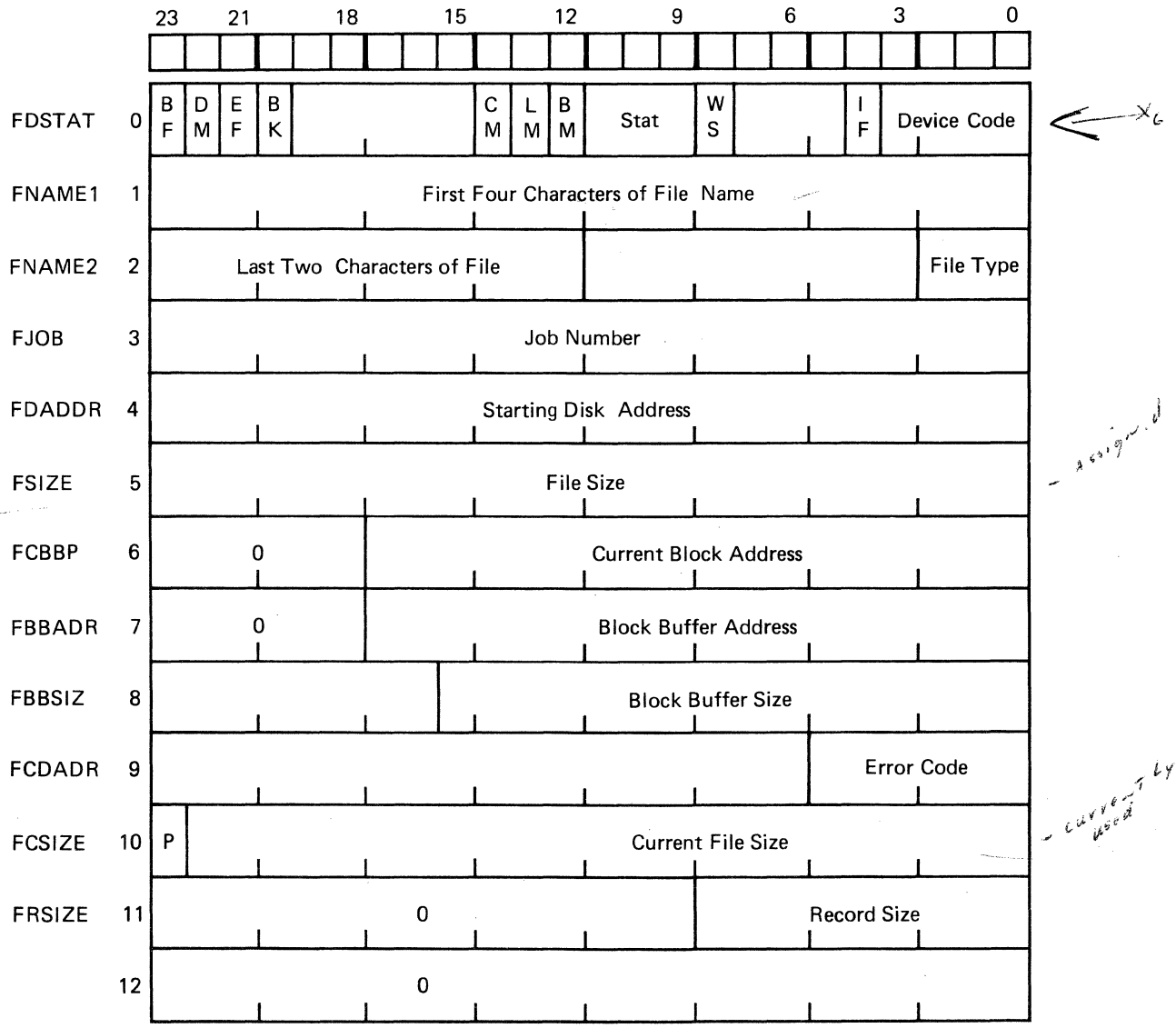


Figure 4-1 I/O Assignment Table Entry

Table 4-1 Word Formats

Word 0: FDSTAT	0 Entry not used. Entry is used when set to 1.
BS (Bit 23)	I/O is active or pending. Turned on when a request with an immediate return is accepted and cleared when that I/O is completed.
DM (Bit 22)	Binary data format; 0 = ASCII.
EF (Bit 21)	EOF has occurred on CLI. It signals CLOSE function MSG TYPE 5 instead of KILL input.
BK (Bit 20)	Auto blocking by $IOCS requested for disk and memory files. Transfer one record at a time to or from the block buffer.
CM (Bit 14)	PID is a command file and the last command has not been completed. Cleared as soon as the command is completed.
LM (Bit 13)	Lock out a command from being entered from this PID.
BM (Bit 12)	Block has been modifed. Used by ADRXLA.
WS (Bit 8)	Working storage I/O.
IF (Bit 4)	I/O is input.
(Bits 3 to 0)	Device Code 1 VK1/VP1 2 MTR1/MTW1 3 MTR2/MTW2 4 CR/LP 5 DIF/DOF 6 CLI/CLO 7 VK2/VP2 8 MIF/MOF
Word 1: FNAME1	First four characters of the file name in TRASCII/ MACTAB address for MIF/MOF.

Table 4-1 Word Formats (Continued)

Word 2: FNAME2	
Bit 23 to 12	Last two characters of the file name in TRASCII.
Bit 2 to 0	File type 000 Source 001 Fixed size data file 010 Object 011 Coreimage 101 Variable length data file
Word 3: FJOB	Job number in TRASCII.
Word 4: FDADDR	Starting disk address of a disk file in binary sector/starting memory address of a memory file.
Word 5: FSIZE	Number of words used for input. Number of words available for output until it is closed.
Word 6: FCBBP	Current block address (0 through FBBSIZ) relative to the first word of the block. Used by auto blocking and close. Word count for data files; character count for source files.
Word 7: FBBADR	Block buffer address in the user's program.
Word 8: FBBSIZ	Block buffer size in words; must be in multiples of 48 words for disk files.
Word 9: FCDADR	Error code detected during immediate return I/O; otherwise, 0.
Word 10: FCSIZE	Number of words already read or written; updated at the end of each read or write. If auto blocking is used, it corresponds with the block data rather than the record data so far processed.
P (Bit 23)	When auto blocking with a partial buffer, bit 23 is also set.
Word 11: FRSIZ Bit 9 to 0	Record size for fixed-length data files. At open time, set to the size in the disk directory or 18 words. At close, written back to disk directory.

4.3 OPEN CALL TO $IOCS

Purpose

To initialize read or write operation.

Description

$IOCS checks for the availability of the device and assigns an entry in I/O usage assignment table. Depending on the device type, the following additional function is performed:

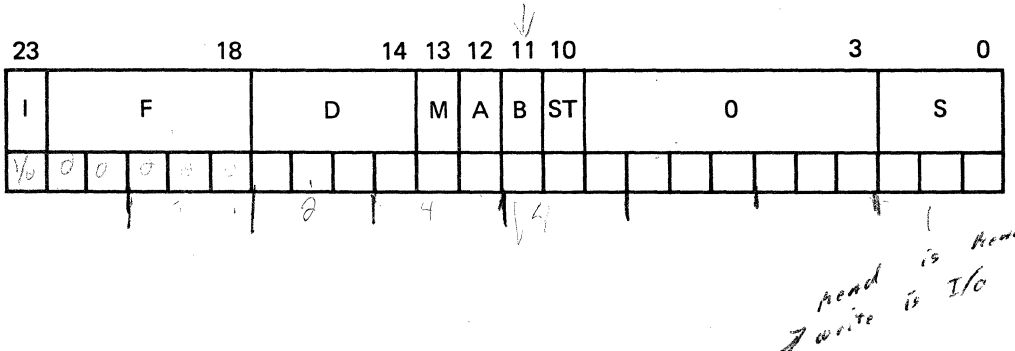
Device	Function Performed	Error Condition
TTK/TTP	Status check	Not available
LP	Status check	Not available
CR	Status check	Not available
MTR	Status check	Not available
MTW	Status check	Not available or write ring off
DIF/DOF	Directory search	Not available or file not found
CLI	Open at the Integrator	Not available or file not found
CLO	Open at the Integrator	Not available or duplicate file
VK2/VP2	Status check	Not available
MIF/MOF	MACTAB search	File not found

Entry Parameters

X1 address of six-word DCB

Globals LOTNUM (three words), DEVNUM (two words), CATGRY (three words), contains file identification if the device is a CLI or CLO and the station identification in DCB word 0 is not zero. These variables are ignored for all other cases.

DCB Word 0



I 0 = output, 1 = input
F function code = 0
D device code
0 device specified in global CMDV (process routine sets it). If CMDV = 0, the system PID/POD.

D device code (continued)
1 TTK/TTP
2 MTR1/MTW1
3 MTR2/MTW2
4 CR/LP
5 DIF/DOF
6 CLI/CLO
7 VK2/VP2
8 MIF/MOF

M Data transfer mode
0 Alpha
1 Binary

B 1 Auto block requested
0 No blocking

A 1 Append open (CLO only); otherwise 0

ST Strobed download for communication link
(ST bit must also be set on each read)

S Station identification
0 Non-station related background function
1 Station 1
2 Station 2
3 Station 3
4 Station 4

DCB Word 1 Block buffer address or 0.

DCB Word 2 Block buffer size in multiples of 48 words or 0.

The block buffer supplied in open is used only if the device is DIF/DOF, MIF/MOF, or MTR/MTW and bit 11 of DCB word 0 = 1.

Input Read a block from disk if the block buffer is empty. Transfer one record from the block buffer to the record buffer.

Output Move a record from the record buffer to the block buffer. Write a block to the disk if the block is full.

The record buffer is supplied at the time of read or write call to $IOCS. Automatic blocking is provided to file type 000 (source), 001 (fixed size data), and 101 (variable size). For all other devices and file types records are transferred directly between the record buffer and the device one record at a time.

If bit 11 of DCB word 0 is zero and the device is a disk, it is assumed that the user takes care of blocking in his own program.

DCB Word 3	Job number in TRASCII 0 = current job in global, JOB -1 = system job It is used for disk files only.
DCB Word 4, 5	Six-character file name for DIF, DOF, CLI, CLO, MIF, MOF.
DCB Word 4	-1 implies the file to be opened is the disk or memory working storage. Bits 2 to 0 of DCB word 5 must contain the file type if the file is a DOF/MOF working storage. 0 string 1 fixed size data 2 OBJ 3 coreimage 5 variable size data

Note

If the device is a system PID or POD, then no function is performed upon open call. The system PID or POD is opened at the time of processing SET COMMAND. It can be used by the user but the user is not able to physically open or close it.

Calling Sequence

```
ROPEN   DATA    40XXXXXXB
        DATA    RBLBUF          BLOCK BUFFER ADDRESS
        DATA    n               BLOCK BUFFER SIZE
        DATA    0,0,0

WOPEN   DATA    00XXXXXXB
        DATA    WBLBUF          BLOCK BUFFER ADDRESS
        DATA    n               BLOCK BUFFER SIZE
        DATA    0,0,0

        LDX     X1, ROPEN
        BSM*    $IOCS
        BRU     ROERR           -error return
        .                       -normal return
        .
        .
        LDX     X1,WOPEN
        BSM*    $IOCS
        BRU     WOERR           -error return
                                -normal return
```

Exit Parameter

For normal return:

X6 The address of I/O assignment table for this device.

For error return:

A register error code.

2 An invalid device code specified or device not available (no interface)
3 An invalid function code specified
4 Parity error during disk directory read or OPEN call to DP system
5 File not found for DIF, DOF, MIF, MOF
7 Device busy
9 Insufficient block size or memory space
10 No write ring on magnetic tape
11 CLIO error
12 I/O table overflow
14 Working storage already in use
15 MACTAB overflow for MOF

4.4 READ/WRITE RECORD

Purpose

To initiate a read or write a record

Description

If the device is a card reader or a magnetic tape, $IOCS initiates a DMA record transfer. If the device is a TTK,TTP,CLI, or CLO, $IOCS initiates record transfer in character-by-character interrupt mode. If the device is a disk, tape, or memory and the block buffer is provided, $IOCS transfers a record between the record buffer and the block buffer. If the block buffer is full for output, $IOCS moves the block to the disk; if the block is empty for input, $IOCS reads the block from the disk. If the block buffer is not provided, $IOCS initiates DMA transfer directly between disk and the user's read or write buffer. If the read/write is requested as ASCII-control-character mode, data characters (40B-137B) obtained are echoed and packed into buffer in TRASCII. A control character is placed right justified into the last word of buffer + 1 and $IOCS exits to the caller. A prompting character (may be used to echo the previous character read) is output before read, if provided. Trailing blanks are not output.

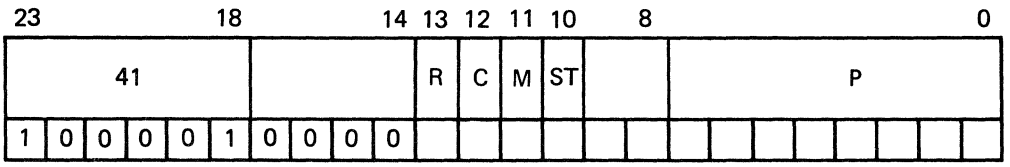
Entry Parameter

X6 The address of I/O assignment table obtained from OPEN.

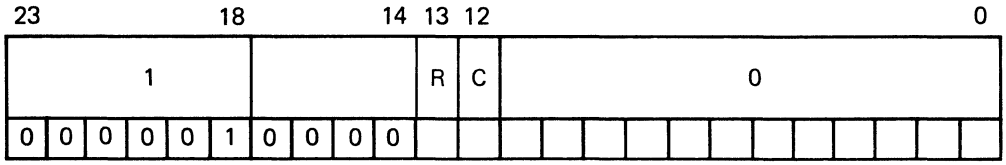
X1 The address of three-word DCB.

DCB Word 0

For read



For write



Where R = Return type

1 Immediate return after initiating I/O or placing in the pending list.

0 Wait for I/O completion.

When the request is activated or placed in the pending list, bit 23 of the first word of I/O assignment table for the request is set to 1, indicating it is busy. When the requested function is completed, it is cleared to zero.

C = 0 Normal mode
1 ASCII control character mode

M = 1 Monitor command read only
0 All others

ST Strobed download from communication link (ST bit must also be set on the open call)

P A prompting character in ASCII code

DCB Word 1 Starting address of record buffer

DCB Word 2 Record size (the number of words to be transferred), the character count for WRITE ASCII control mode

Calling Sequence

```
RDCB    DATA    410XXX00B
        DATA    INBUF
        DATA    n           RECORD SIZE

WDCB    DATA    01000000B
        DATA    OUTBUF
        DATA    n           NO. OF WORDS TO WRITE

        LDX*    X6,IOTPI
        LDX     X1,RDCB
        BSM*    $IOCS
```

```
BRU     RDERR       error return
--                  normal return

LDX*    X6,IOTPO
LDX     X1,WDCB
BSM*    $IOCS
BRU     WRERR       error return
--                  normal return
```

Exit Parameters

For error return:

A register error code

1	end of input file
2	device off-line
4	parity error
6	unrecoverable hardware error
7	device busy
8	invalid I/O assignment address
9	file overflow for DOF or MOF
13	excess word count on read

For error 13, the E register contains the actual number of words read. Error 13 takes normal return rather than error return.

4.5 TERMINATE I/O—CLEAR SCREEN

Purpose

To terminate currently active I/O on TTK/TTP

Description

$IOCS clears the channel if the device is TTK or CLI. If the device is VKT screen, the screen is cleared.

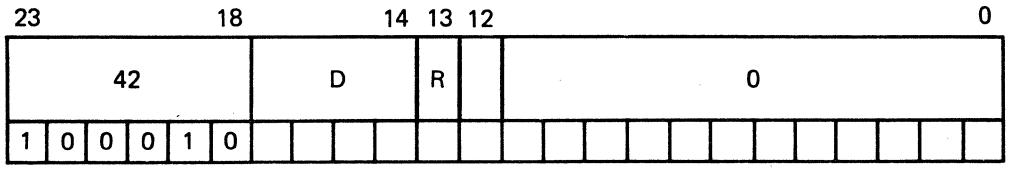
Entry Parameters

X6 The address of I/O assignment table or 0. If it is zero, the device must be specified in the parameter word 1.

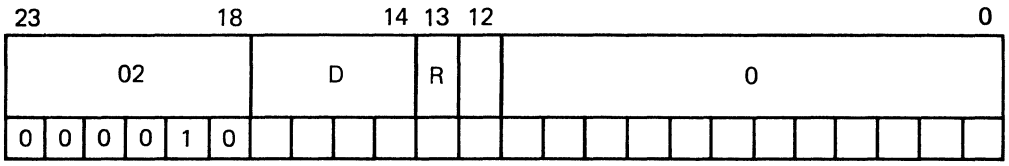
X1 The address of one word DCB

DCB word 0

For input



For output



Where

D device code
- 0 device specified by X6
- 1 TTK/TTP
- 6 CLI
- 7 VK2/VP2

R return type

(See description in paragraph 4.4 READ/WRITE RECORD)

Calling Sequence

```
KILLIN    DATA    42XX0000B
KILLOUT   DATA    02XX0000B

          LDX*    X6,IOTPI
          LDX     X1,KILLIN
          BSM*    $IOCS
          BRU     ERROR          error return
          --                     normal return
          LDX*    X6,IOTPO
          LDX     X1,KILLOUT
          BSM*    $IOCS
          BRU     ERROR          error return
          --                     normal return
```

Exit Parameters

For error return:

A register error code

- 2 device off-line
- 6 unrecoverable hardware error
- 7 device busy
- 8 invalid I/O assignment table address

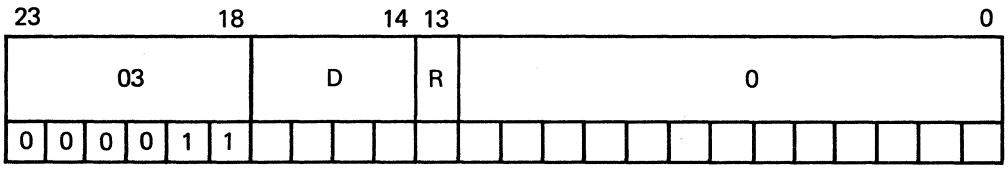
4.6 TOP OF FORM

Purpose — To advance to top-of-form on line printer

Description — This function is performed to the line printer only. If other devices are specified, it is a NOP.

Entry Parameter
- X6 Address of I/O assignment table
- X1 Address of one word DCB

DCB Word 0



Where

D Device code
- 0 Device specified by X6
- 4 LP

R Return type (See paragraph 4.4 READ/WRITE RECORD for description)

Calling Sequence:

```
TOF     DATA    03XXXXXXB
        LDX*    X6,IOTPO
        LDX     X1,TOF
        BSM*    $IOCS
        NOP     *           ignore error return
        --                  normal return
```

Exit Parameters — For error return:

A register error code

- 2 device off-line
- 7 device busy
- 8 invalid I/O assignment table address

4.7 UNFORMATTED ALPHA WRITE

Purpose

To initiate a write ASCII operation without trailing carriage return and line feed

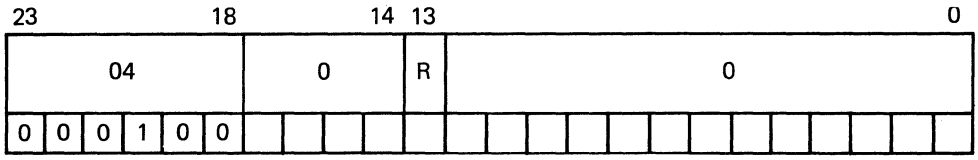
Description

This function is used to write a message and wait for a response in the same line; for example, PIN NUMBER = response entered here. Therefore, this function is provided to TTP only.

Entry Parameters

X6 The address of I/O assignment table
X1 The address of three-word DCB

DCB Word 0



Where

R Return type (see paragraph 4.4 READ/WRITE RECORD for description)

DCB Word 1

Starting address of the output buffer

DCB Word 2

Record size (the number of words to be output)

Calling Sequence

```
WNCRLF  DATA    040X0000B
        DATA    WRBUF       BUFFER ADDR
        DATA    n           RECORD SIZE

        LDX*    X6,IOTPO
        LDX     X1,WNCRLF
        BSM*    $IOCS
        BRU     ERROR       error return
        --                  normal return
```

Exit Parameter

For error return:

A register error code

7 device busy
8 invalid I/O assignment table address

4.8 SKIP A FILE MARK ON MAGNETIC TAPE

Purpose

To skip past an EOF mark on the tape.

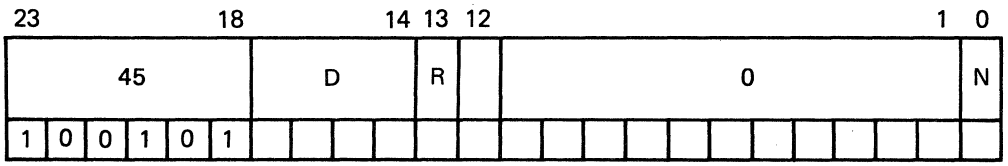
Description

The tape is moved until an EOF mark is skipped over.

Entry Parameters

X6 The address of I/O assignment table or zero if the device is specified in the DCB
X1 The address of one word DCB

DCB Word 0



Where

D Device code
- 0 device specified by X6
- 2 MTR1
- 3 MTR2

R Return type (see paragraph 4.4 READ/WRITE RECORD for description)

N
- 0 skip forward
- 1 skip backward

Calling Sequence:

```
SKIPF   DATA    45XX000XB

        LDX*    X6,IOTPI
        LDX     X1,SKIPF
        BSM*    $IOCS
        BRU     ERROR       error return
        --                  normal return
```

Exit Parameters

For error return:

A register error code

2 device off-line
7 device busy
8 invalid I/O assignment table address

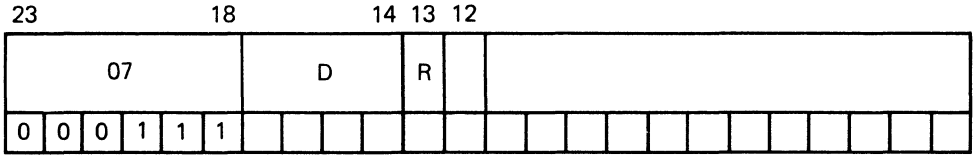
4.9 WRITE EOF MARK ON MAGNETIC TAPE

Purpose

To write a file terminator mark on a tape

Entry Parameter

X1 Address of one-word DCB
X6 Address of I/O assignment table or 0 for device override



Where

D device code

0 device specified by X6
2 MTW1
3 MTW2

R Return type (See paragraph 4.4 READ/WRITE RECORD for description)

Calling Sequence

```
EOFDCB  DATA    07XX0000B

        LDX     X1,EOFDCB
        BSM*    $IOCS
        BRU     ERROR       error return
        --                  normal return
```

Exit Parameter

None for normal return

For error return:

A register error code

2 device off-line
7 device busy
8 invalid I/O assignment table address
10 write ring

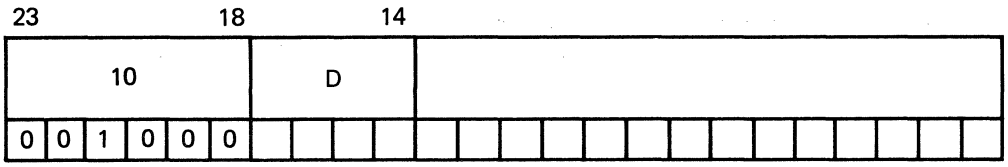
4.10 STATUS CHECK REQUEST

Purpose — To check the status of a device

Description — $IOCS issues a status read to the device and returns the status word.

Entry Parameter
- X1 Address of one-word DCB
- X6 Address of I/O assignment table for 0 for device override

DCB Word 0



Where:

D — Device code
- 0 device specified by X6
- 2 MTR1/MTW1
- 3 MTR2/MTW2
- 5 DIF/DOF
- 6 CLI/CLO

Calling Sequence

```
        STDCB   DATA    10XX0000B

                LDX     X1,STDCB
                BSM*    $IOCS
                BRU     ERROR       error return
                --                  normal return
```

Exit Parameter — For normal return:

A register status word

Status Word Description:

Bit	Magnetic Tape Status	Disk Status
0	Device ready	Disk not ready
1	Interrupt in process	Parity error
2	Interrupt enabled	Interrupt enabled
3	Interrupt pending	Interrupt not manually inhibited
4	Rewinding	Memory not manually inhibited
5	No write enable ring	Maintenance segment addressable
6	Memory protect Switch on	Data overflow/memory Protect on
7	BOT	Track address overflow
8	Low density	DCB error
9	Tape mark has passed	Interrupt in process
10	Data overflow	Segment not found
11	DCB error	Write not manually inhibited
12	Rewind ended	Write not inhibited
13	Word count record Length	DCU in error state
14	Word count record Length	
15	Longitudinal parity Error	
16	Vertical parity error	
17	EOT has passed	

For error return:

A register error code

7 device busy
8 invalid I/O assignment table address

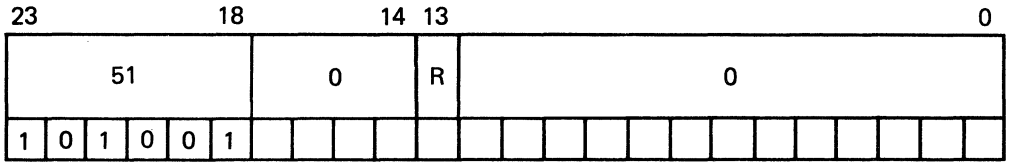
4.11 VERIFY/READ

Purpose — To read a record without data transfer to memory for verification

Description — This function is provided to magnetic tape and disk only.

Entry Parameter
- X1 Address of two-word DCB
- X6 Address of I/O assignment table

DCB Word 0



- R Return type (see paragraph 4.4 READ/WRITE RECORD for description)

DCB Word 1 — Buffer address

DCB Word 2 — Record word count

Calling Sequence

```
VERDCB  DATA    510X0000B
        DATA    RBUF
        DATA    512

        LDX     X1,VRDCB
        BSM*    $IOCS
        BRU     ERROR       error return
        --                  normal return
```

Exit Parameter: For error return:

A register error code

1 end of file
2 device not available
4 parity error
7 device busy
8 invalid I/O assignment table address

4.12 CLOSE A FILE

Purpose

To release the device usage

Description

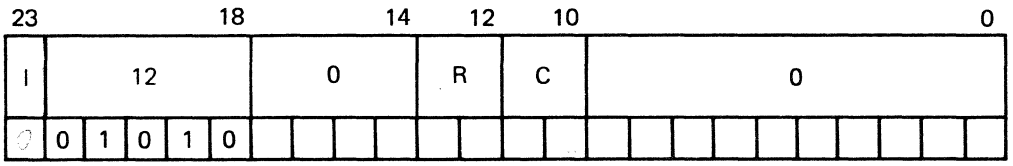
Causes the entry in the I/O assignment table to be released. When I is equal to zero, the following actions are also taken.

If the device is a MTW1 or MTW2, two end-of-file marks (one for file end, one for tape end) are written and the tape is repositioned backward so that the next file overrides the tape-end mark if written. If the device is a DOF or MOF, the block buffer, if any, is output to disk or memory and the directory is updated with the current size. If the device is a CLI or CLO, a CLOSE call is issued. If the device is a TTK, TTP, CR, LP, MTR1, MTR2, MIF, or DIF, no other function is performed.

Entry Parameters

X6 The address of I/O assignment table
X1 The address of one-word DCB

DCB Word 0



Where

I 1 input
0 output

R Return type (see paragraph 4.4 READ/WRITE RECORD description)

C CLO OPTION
0 hold CLO
1 purge CLO
2 process CLO

Calling Sequence

```
RCLOSE  DATA   52000000B
        LDX    X6,IOTPI
        LDX    X1,RCLOSE
        BSM*   $IOCS
        BRU    ERROR        error return
        --                  normal return
```

Exit Parameters

For error return:

A register error code

4 parity error
7 device busy
8 invalid I/O assignment table address

4.13 OPERATOR MESSAGE

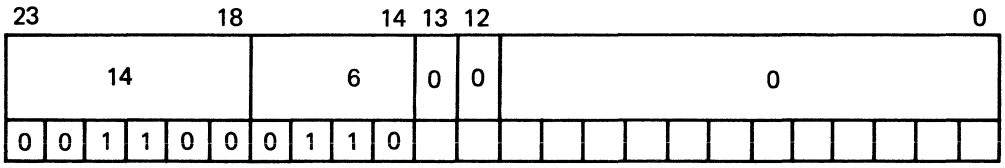
Purpose

To display a message at the Integrator VKT

Description

Allows transfer of a message from the local VKT to the Integrator VKT

DCB Word 0



DCB Word 1

Address of the message buffer

DCB Word 2

Number of words to output

Calling Sequence

```
OPMDCB  DATA    14300000B,MSAGBUF,n

        LDX     X1,OPMDCB
        BSM*    $IOCS                   error return
        --                              normal return
```

Exit Parameters

For error return:

A register error code

6 unrecoverable CLIO error
7 device busy

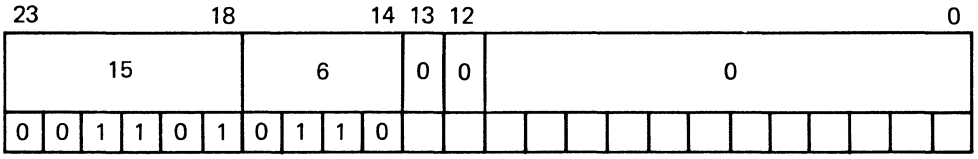
4.14 DISCONNECT CLIO

Purpose — To disconnect communication link

Description — Used to disconnect the linkage to Integrator. Upon next open, the line is connected again.

Entry Parameters — X1 Address of one-word DCB

DCB Word 0



Calling Sequence

```
HNGDCB   DATA    15300000B

         LDX     X1,HNGDCB
         BSM*    $IOCS
         --                      error return
         --                      normal return
```

Exit Parameters — For error return:

A register error code

6 unrecoverable CLIO error
7 device busy

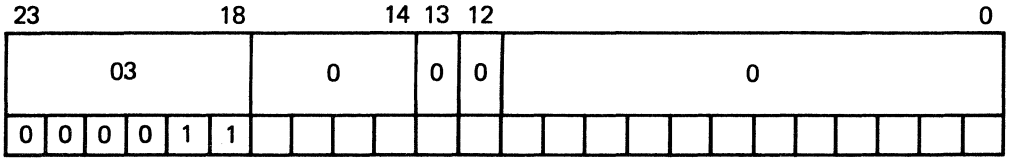
4.15 FILE TRANSMIT (ADD)

Purpose — To initiate up-load of data to be appended to the file at the Integrator

Description — This function is reserved for $IOCS internal use. It is internally generated by $IOCS initializer when an open output add is issued to $IOCS.

Entry Parameters — X6 Address of I/O assignment table
X1 Address of three-word DCB

DCB Word 0



DCB Word 1 — The address of the 10-word file identification

word	0,1	file name, file type
	2,3,4	lot number or zero
	5,6	device number or zero
	7,8,9	category or zero

DCB Word 2 — The number of words to output = 10

Calling Sequence

```
FTADCB   DATA    03000000B,CLIOID, 10

         LDX*    X6,CSX6
         LDX     X1,FTADCB
         BSM     $IOCS
         --                          error return
         --                          normal return
```

Exit Parameters — For error return:

A register error code

4 parity error
6 unrecoverable CLIO error
7 device busy

4.16 FILE TRANSMIT (CREATE)

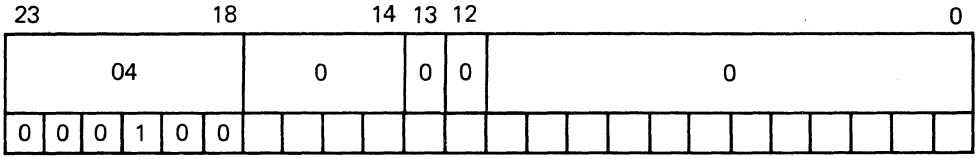
Purpose — To initiate up-load of data and create a file at the Integrator

Description — This function is reserved for $IOCS internal use. It is generated internally by $IOCS initializer when an open output is issued to $IOCS.

Entry Parameters

X6 Address of I/O assignment table
X1 Address of three-word DCB

DCB Word 0



DCB Word 1 — The address of the 10-word file identification

word	0,1	file name, file type
	2,3,4	lot number or zero
	5,6	device number or zero
	7,8,9	category or zero

DCB Word 2 — The number of words to output = 10

Calling Sequence

```
FTDCB   DATA    04000000B,CLIOID,10
        LDX*    X6,CSX6
        LDX     X1,FTDCB
        BSM     $IOCS
        --                          error return
        --                          normal return
```

Exit Parameters — For error return:

A register error code

4 parity error
5 duplicate file
6 unrecoverable CLIO error
7 device busy

4.17 FILE END INPUT

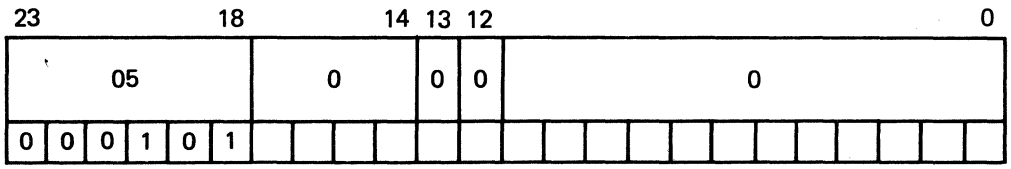
Purpose — To close input file due to EOF at the Integrator

Description — This function is reserved for $IOCS internal use. It is generated by $IOCS initializer when a close input is issued to $IOCS.

Entry Parameters

X6 Address of I/O assignment table
X1 Address of one-word DCB

DCB Word 0



Calling Sequence

```
FEIDCB    DATA    5000000B

          LDX*    X6,CSX6
          LDX     X1,FEIDCB
          BSM     $IOCS
          --                          error return
          --                          normal return
```

Exit Parameters

For error return:

A register error code

4 parity error
6 unrecoverable CLIO error
7 device busy

4.18 FILE END OUTPUT (PROCESS)

Purpose

To close an output file at the Integrator and allow immediate processing.

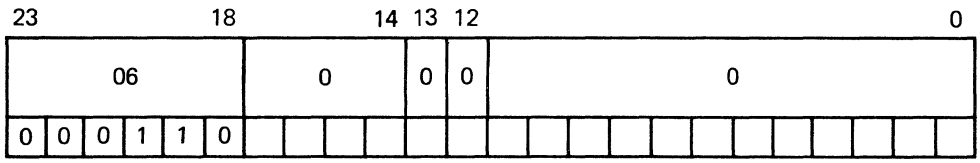
Description

This function is reserved for $IOCS internal use. It is internally generated by $IOCS initializer when a close output is issued to $IOCS.

Entry Parameters

X6 Address of I/O assignment table
X1 Address of one-word DCB

DCB Word 0



Calling Sequence

```
FEDCB   DATA    06000000B

        LDX*    X6,CSX6
        LDX     X1,FEDCB
        BSM     $IOCS
        --                      error return
        --                      normal return
```

Exit Parameters

For error return:

A register error code

4 parity error
6 unrecoverable CLIO error
7 device busy

4.19 FILE END OUTPUT (PURGE)

Purpose

To purge an output file at the Integrator

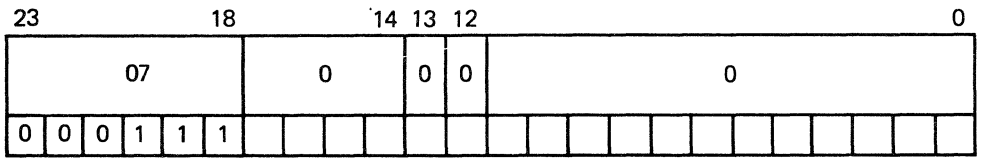
Description

This function is reserved for $IOCS internal use. It is generated by $IOCS initializer when a purge-open is issued to $IOCS.

Entry Parameters

X6 Address of I/O assignment table
X1 Address of three-word DCB

DCB Word 0



DCB Word 1 — The address of 10-word file identification

word	0,1	file name, file type
	2,3,4	lot number or zero
	5,6	device number or zero
	7,8,9	category or zero

DCB Word 2 — The number of words to output = 10

Calling Sequence

```
FEPDCB   DATA   07000000B,CLIOID,10

         LDX*   X6,CSX6
         LDX    X1,FEPDCB
         BSM    $IOCS
         --                          error return
         --                          normal return
```

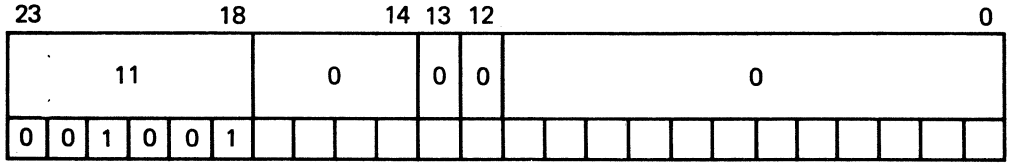
4.20 FILE END OUTPUT (HOLD)

Purpose — To close an output file and hold for later processing at the Integrator

Description — This function is reserved for $IOCS internal use. It is generated by $IOCS initializer when a close output and hold is issued to $IOCS.

Entry Parameters
- X6 Address of I/O assignment table
- X1 Address of one-word DCB

DCB Word 0



Calling Sequence

```
FEHDCB    DATA    11000000B

          LDX*    X6,CSX6
          LDX     X1,FEHDCB
          BSM     $IOCS
          --                        error return
          --                        normal return
```

Exit Parameters

For error return:

A register error code

4 parity error
5 file name error
6 unrecoverable CLIO error
7 device busy

4.21 FILE REQUEST

Purpose

To initiate download of a file from the Integrator

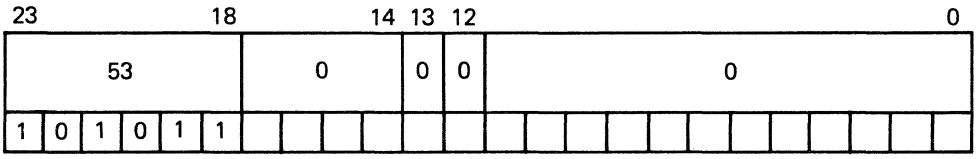
Description

This function is reserved for $IOCS internal use. It is internally generated by $IOCS initializer when an open input to CLI is issued to $IOCS.

Entry Parameters

X6 Address of I/O assignment table
X1 Address of three-word DCB

DCB Word 0



DCB Word 1

The address of the 10-word file identification

word 0,1 file name, file type
2,3,4 lot number or zero
5,6 device number or zero
7,8,9 category

DCB Word 2

The number of words to output = 10

Calling Sequence

```
FRDCB   DATA    53000000B,CLIOID,10

        LDX*    X6,CSX6
        LDX     X1, FRDCB
        BSM     $IOCS
        --                          error return
        --                          normal return
```

Exit Parameters

For error return:

A register error code

4 parity error
5 file not found
6 unrecoverable CLI error
7 device busy

4.22 VKT TRANSMIT

Purpose

To initiate data transmisssion from VKT screen to a buffer

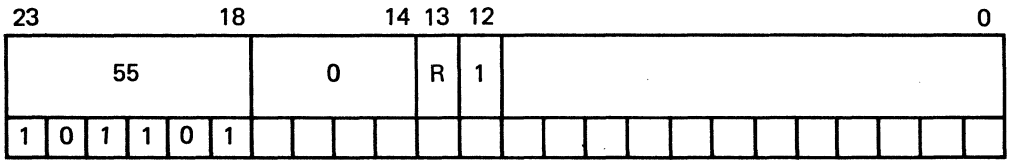
Description

It is a special form of read and only applies to TTP. VKT transmit allows reading of the data on the VKT screen from the home position to the last-written data position. The buffer must be large enough to store 74 characters by 27 lines in ASCII (666 words). All characters except EOT (4B) are transferred to the buffer without any conversion (three ASCII characters per word).

Entry Parameters

X6 Address of I/O assignment table obtained from open
X1 Address of three-word DCB.

DCB Word 0



Where

R Return type (see paragraph 4.4 READ/WRITE RECORD)

DCB Word 1 — Starting address of record buffer

DCB Word 2 — Record size (number of characters)

Calling Sequence

```
        DCB     DATA    55010000B,BUF,1998

                LDX     X1,DCB
                BSM*    $IOCS
                BRU     ERROR           error return
                --                      normal return
```

Exit Parameters — For error return only:

A register error code

7 device busy
8 invalid I/O assignment table address

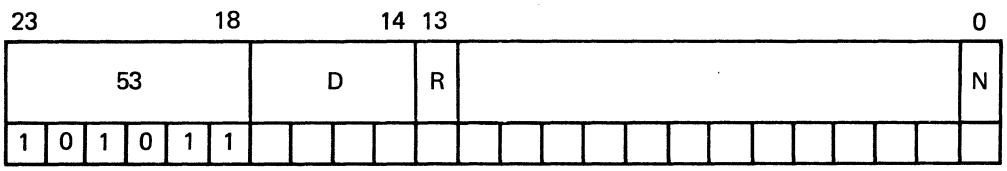
4.23 SKIP A RECORD ON MAGNETIC TAPE

Purpose — To skip a record forward or backward

Entry Parameters

X6 Address of I/O assignment table or zero if the device is specified in DCB
X1 Address of one-word DCB

DCB Word 0



Where

D device code
- 0 device specified by X6
- 2 MTR1
- 3 MTR2

R return type
- 0 return after completion
- 1 immediate return

N
- 0 skip forward
- 1 skip backward

Calling Sequence

```
SKIPR   DATA    53000000B

        LDX*    X6,IOTP
        LDX     X1,SKIPR
        BSM*    $IOCS
        BRU     ERROR       error return
        --                  normal return
```

Exit Parameters

For error return:

A register error code

7 device busy
8 invalid I/O assignment table

4.24 PAGE A BLOCK INTO MEMORY

Purpose

To refresh the block of data from the disk or memory file at the current position of the file. For operating system use only.

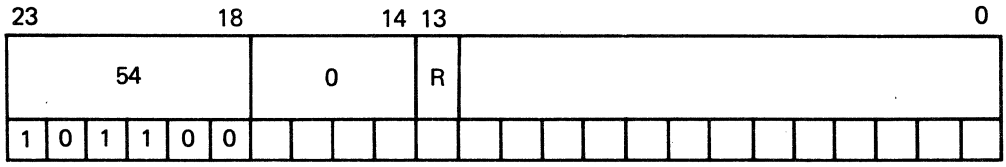
Description

This function can be used only if the block buffer was provided at open time. $IOCS reads in a block from the disk or memory starting at the sector/memory address which is one block before the current address specified in the IOATAB.

Entry Parameter

X6 Address of I/O assignment table
X1 Address of one-word DCB

DCB Word



R return type (see paragraph 4.4 READ/WRITE RECORD for description)

Calling Sequence

```
RPDCB   DATA    54000000B

        LDX*    X6,IOPT
        LDX     X1,RPDCB
        BSM*    $IOCS
        BRU     ERROR       error return
        --                  normal return
```

Exit Parameters

For error return:

A register error code

4 parity error
7 device busy
8 invalid I/O assignment table

4.25 REWIND MAGNETIC TAPE

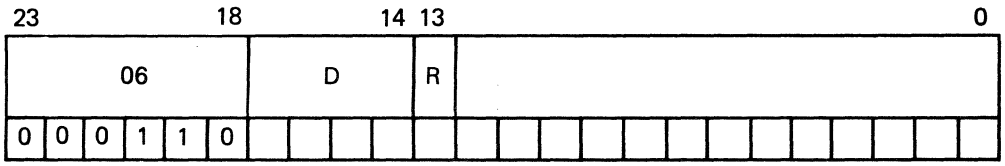
Purpose

To rewind a tape to BOT position

Entry Parameter

X6 Address of I/O assignment table or 0 for device override
X1 Address of one-word DCB

DCB Word



D device code
0 device specified by X6
2 MTW1
3 MTW2

R return type

Calling Sequence

```
        DCB     DATA    06XX0000B

                LDX     X1,DCB
                BSM*    $IOCS
                BRU     ERROR       error return
                --                  normal return
```

Exit Parameters

For error return:

A register error code

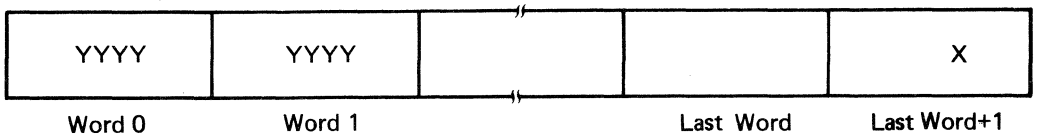
2 device off line
7 device busy
8 invalid I/O assignment table

4.26 MASTR $IOCS ASCII CONTROL MODE I/O

1. Read

   Uses the same DCB word 0 as normal read with bit 12 of DCB word 0 set to 1 (see paragraph 4.4). The record buffer must be defined as the maximum allowed input words plus one. All data characters, (40_8 and 137_8) are stored into the buffer in TRASCII (four characters per word). As soon as a control character appears, it is placed in ASCII right justified into the last word of the buffer and $IOCS returns to the caller.

   Caller's Buffer



   Y = data characters in TRASCII
   X = control characters in ASCII

   The number of characters obtained, excluding the control character, is returned in the A register.

2. Write

   Uses the same DCB word 0 as normal write with bit 12 of DCB word 0 set to 1 (see paragraph 4.4). Characters to be written must be stored in ASCII (three characters per word) left justified. DCB word 3 contains the number of characters to be written.

5

MASTR File Description

5.1 MASTR SYSTEM FILES

MASTR system files are created, saved, and transferred to and from the disk, magnetic tapes, the disk at the Integrator, and in memory. MASTR maintains all files in defined format. The physical formats are defined by the peripheral requirements and described in paragraph 5.2 through 5.5. The logical formats of files are defined by MASTR and described in paragraphs 5.6 through 5.10.

5.2 DISK FILES AND USAGE

5.2.1 Disk Specification

200 tracks in Burroughs disk
192 tracks in Alpha Data disk

80 sectors/track
48 words/sector

5.2.2 Disk Organization

The disk is physically divided into five sections (figure 5-1). The coreimage buffer is used by DOPSY; $ARR is the DOPSY monitor and is booted into memory when the MASTR command DOPSY is entered.

The disk files are identified in the disk directory. The information of each file is stored in a six-word directory entry (figure 5-2). There are 640 file entry spaces reserved on disk (80 sectors).

The file area contains the actual file data in consecutive order, as specified in the directory. The directory does not contain any pointer to a file; to locate the starting sector address of a file, the sum of assigned sector numbers of the preceding files must be added to the starting address of the file area.

The unused portion of the disk is called working storage, which can be used to assign a file space or to create a new file.

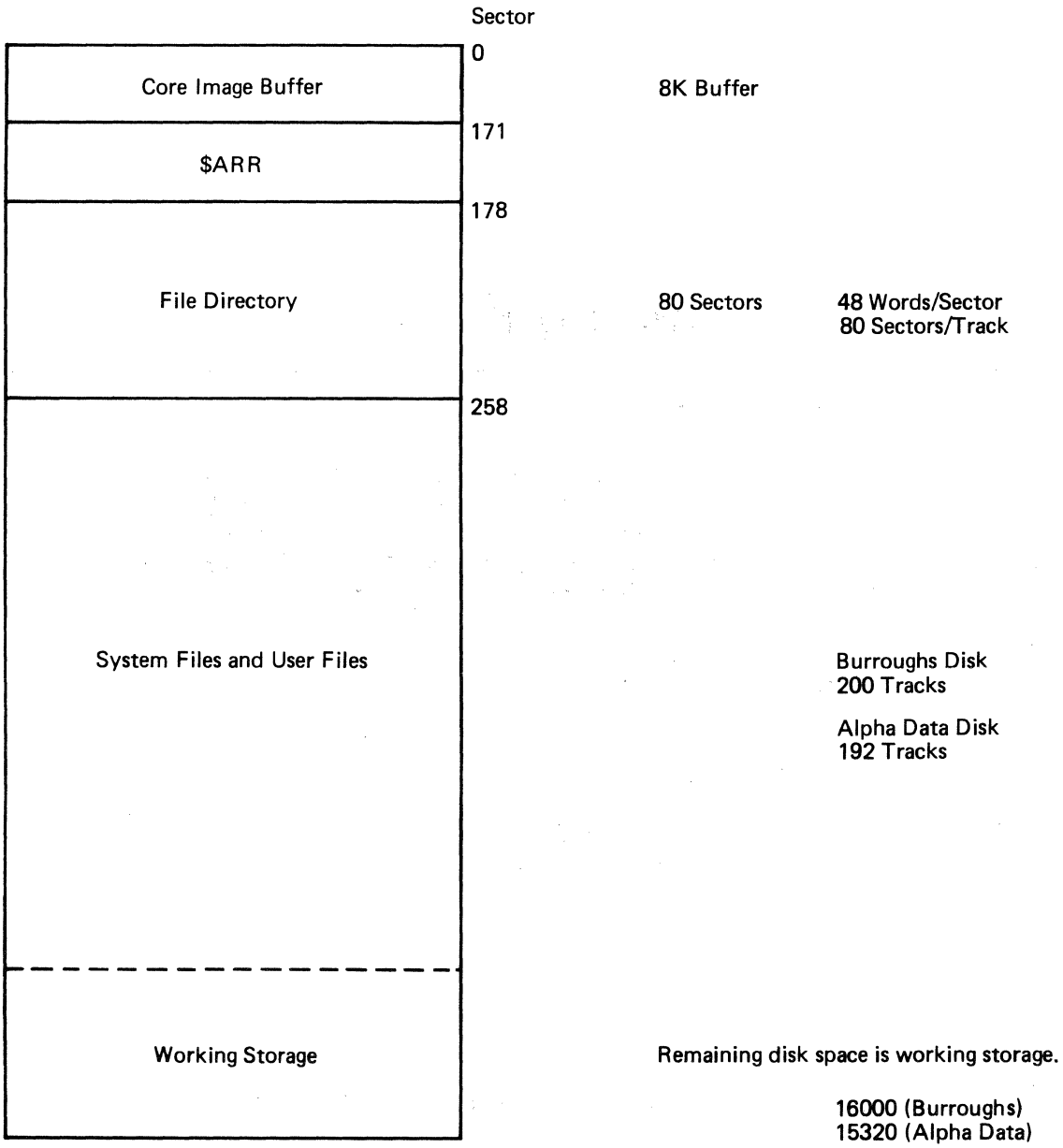
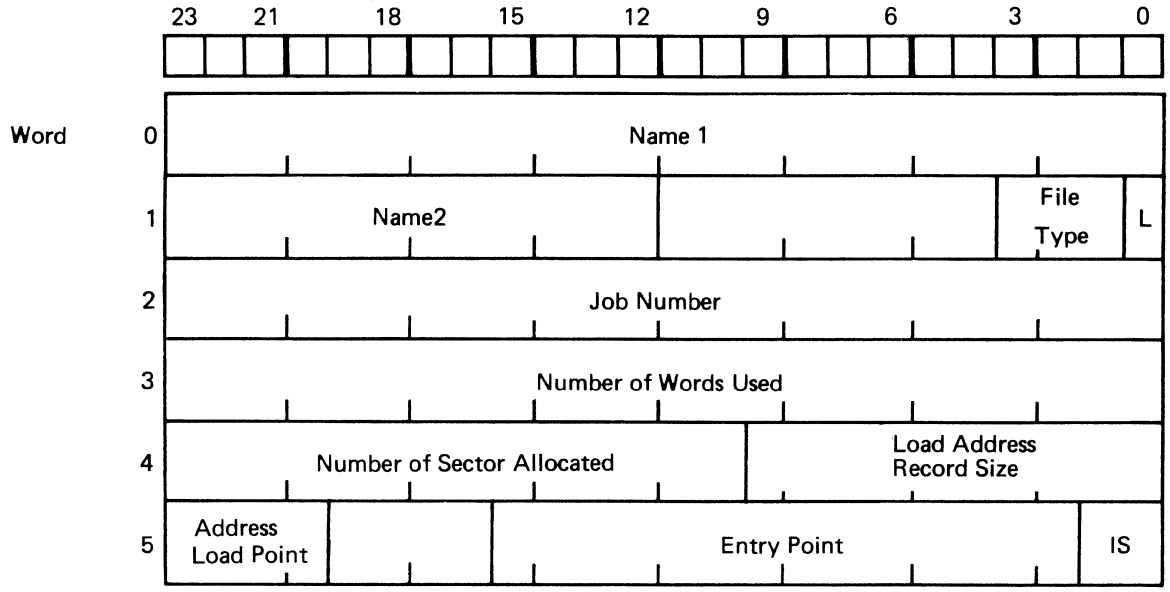


Figure 5-1 Disk Organization



Word	Bit	Description
0	23 to 0	First four characters of file name (TRASCII)
1	23 to 12	Last two characters of file name (TRASCII)
	11 to 4	Not used
	3 to 1	File type 000 string 001 data (fixed length) 010 object 011 coreimage 101 data (variable length)
	0	= 1 last entry in the directory
2	23 to 0	Four-character job number (TRASCI) of the file
3	23 to 0	Number of words used by the file (file size)
4	23 to 10	Number of sectors allocated for the file
	9 to 0	File type 011 only - first 10 bits of address (loading point of program)
	9 to 0	File type 001 only - record size
5 (File Type) 011 only	23 to 20	Last four bits of address (load point)
	19 to 16	Not used
	15 to 2	Entry point of program
	1 to 0	Interrupt sector count used

Figure 5-2 Disk Directory Entry (Six Words per File)

5.2.3 Disk File Format

On disk, five different file types are maintained: string files, variable data files, fixed data files, coreimage files and object files. DOPSY does not generate or recognize variable length data files and treats all data files as if they are fixed.

All files on disk are placed continguously without any physical record or file terminators. An end of a file is identified by the number of words in the file, which can be obtained in the directory entry. A record can be identified based on the file type.

A string file record for the disk is described in paragraph 5.6 and it is identical to ones on other media.

Coreimage file format is described in paragraph 5.9. In DOPSY, coreimage files are not transferable.

Five different record formats for object files are described in paragraph 5.10.

5.3 MAGNETIC TAPE FILES AND USAGE

5.3.1 Tape Organization

A physical record is data terminated by a record mark. Skipping a record forward or backward is done on this physical record. The size of one physical record is determined by the program which generates the record onto a tape.

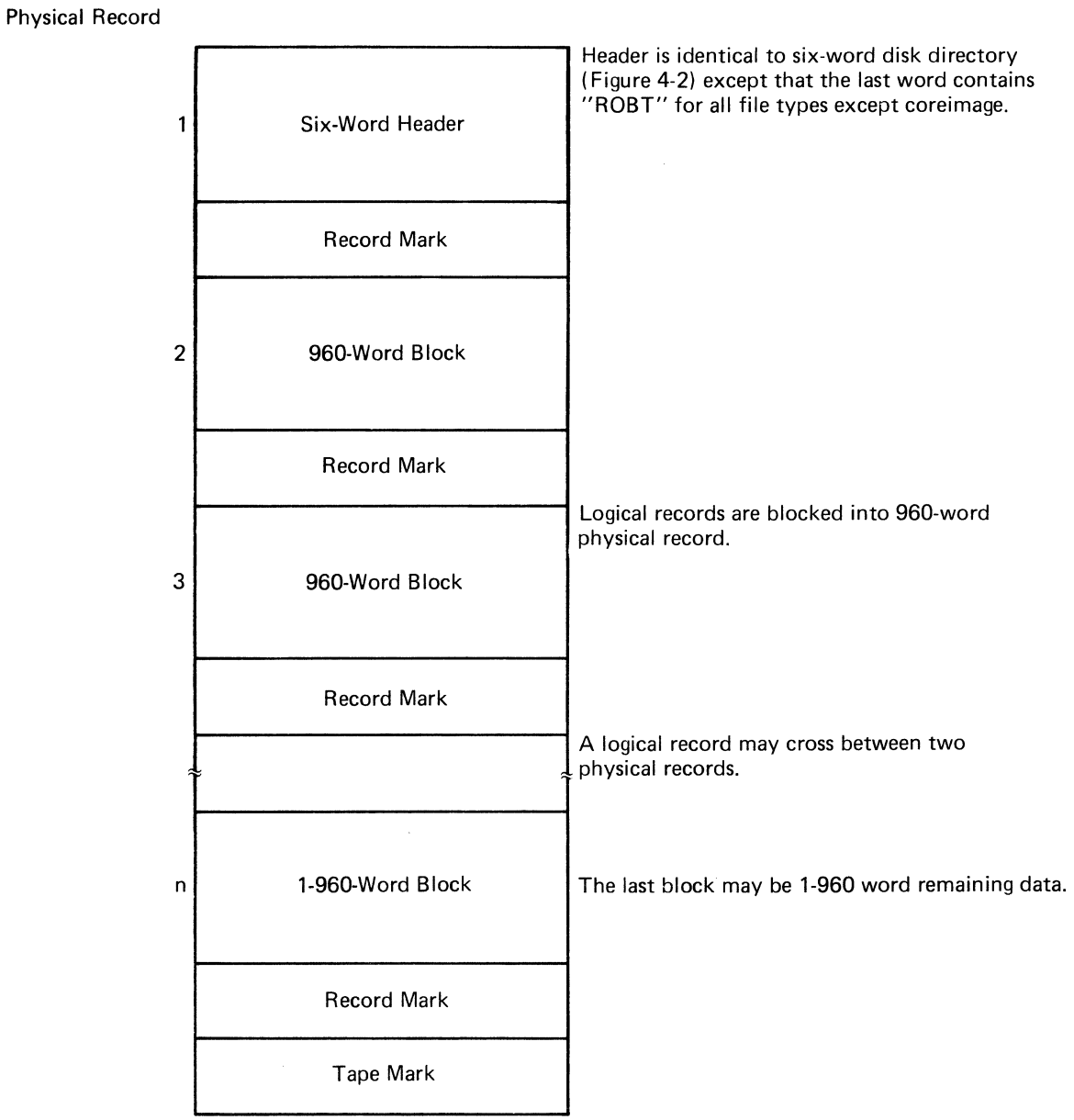
A logical record is defined based on the file type. Normally, there are several logical records in a physical record.

A file is a set of physical records terminated by a tape mark. Skipping a file forward or backward uses this tape mark for positioning. The tape mark is normally written at the file closing time. Datalogging to a magnetic tape or FACTOR writing to a magnetic tape always causes a tape mark to be written immediately after a physical record and repositioned back. If another record is written, the tape mark is erased over; otherwise, the tape mark is there even if the file is never closed.

An end of a logical tape (an end of valid data area) is identified by having a tape mark without a file. $IOCS always writes two tape marks at the file closing time and backs up one tape mark in case another file is written. Therefore, all closed tapes always have an end of a logical tape.

5.3.2 Blocked File Format Generated by DUMP and COPY Programs

File types handled by DUMP and COPY are identical to those on disk; string, variable and fixed size data, coreimage and object files. Record formats for these files are described in paragraphs 5.6 through 5.10 and shown in figure 5-3.



This format can be read by LOAD, COPY or DOPSY BMT programs.

Figure 5-3 A Blocked File

5.3.3 TDX-Generated Magnetic Tape

During TDX INIT, a terminator record is generated (figure 5-4).

TDX MAKE generates multiple directories (figure 5-5). Each directory contains a directory header (figure 5-6) followed by multiple files (figure 5-5).

TDX program causes COPY program to generate files as described in paragraph 5.3.2.

This tape format can be read by LOAD, MASTR TDX and DOPSY TDX.

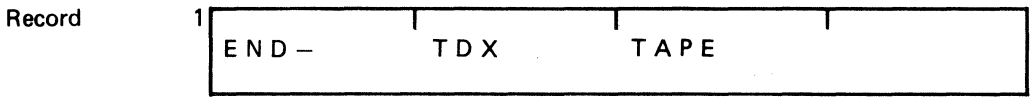


Figure 5-4 Terminator Format

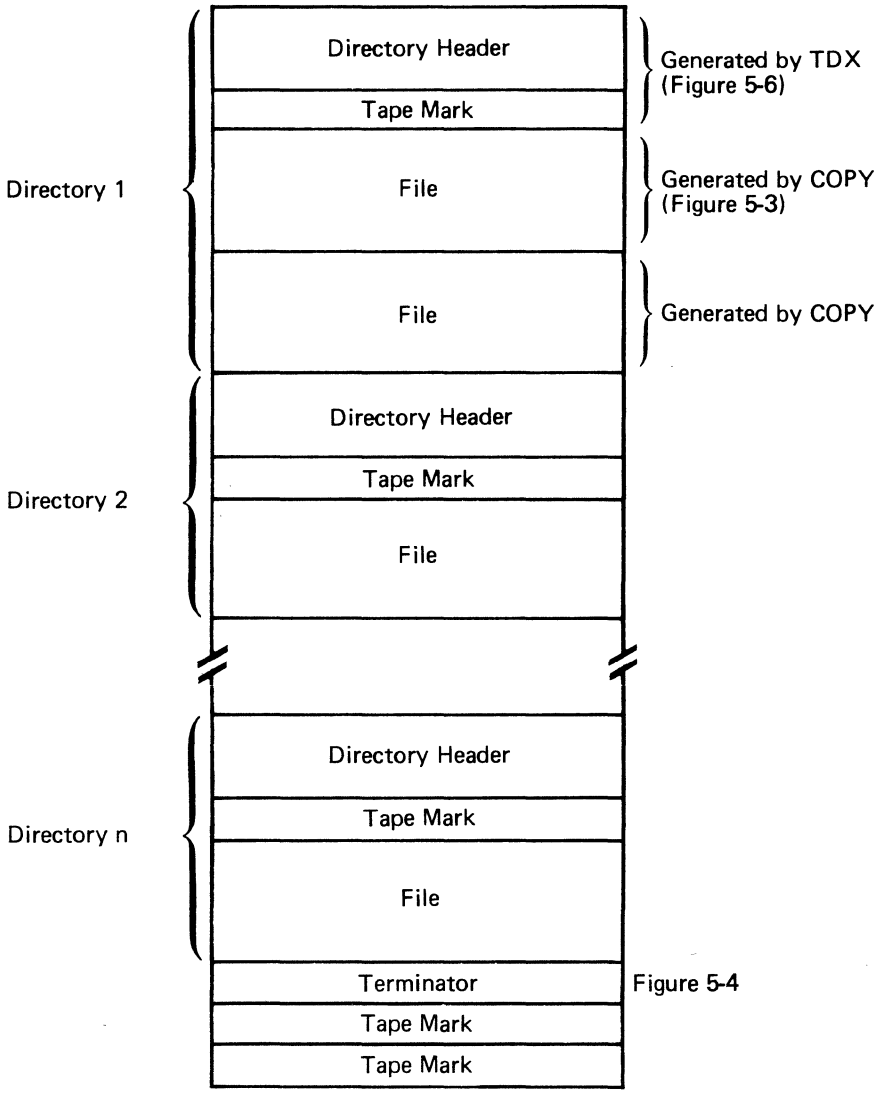


Figure 5-5 TDX Tape Format

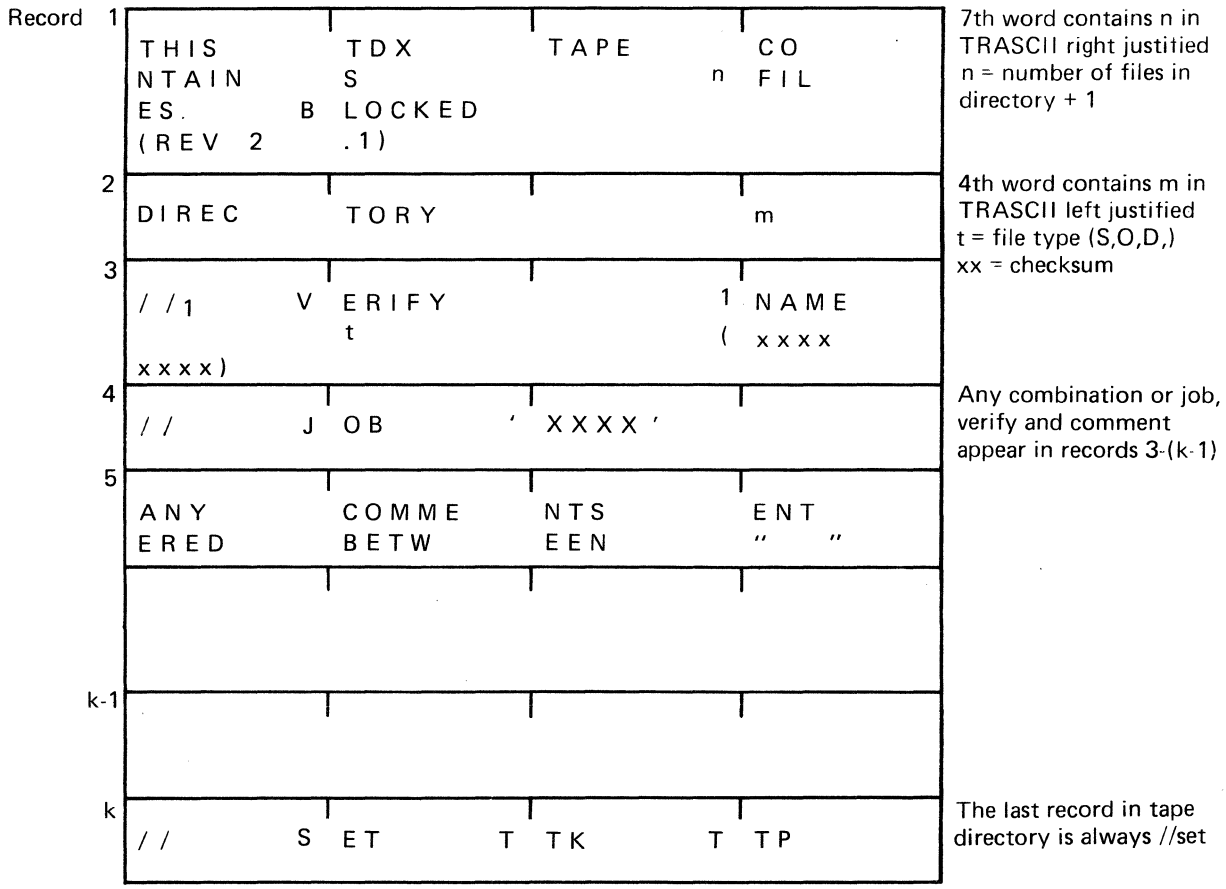


Figure 5-6 TDX Directory Header Format

5.3.4 MBUP Tape Format

The purpose of the MBUP tape is to save the contents of memory on magnetic tape (figure 5-7). MBUP transfers the bootstrap loader and the operating system from memory to magnetic tape. It then calls DUMP to output the remaining files.

This tape can be read by the MBUP overlay or LOAD program. If the first two files (the boot and the operating system) are skipped, the remaining files can be read by the COPY overlay or DOPSY BMT.

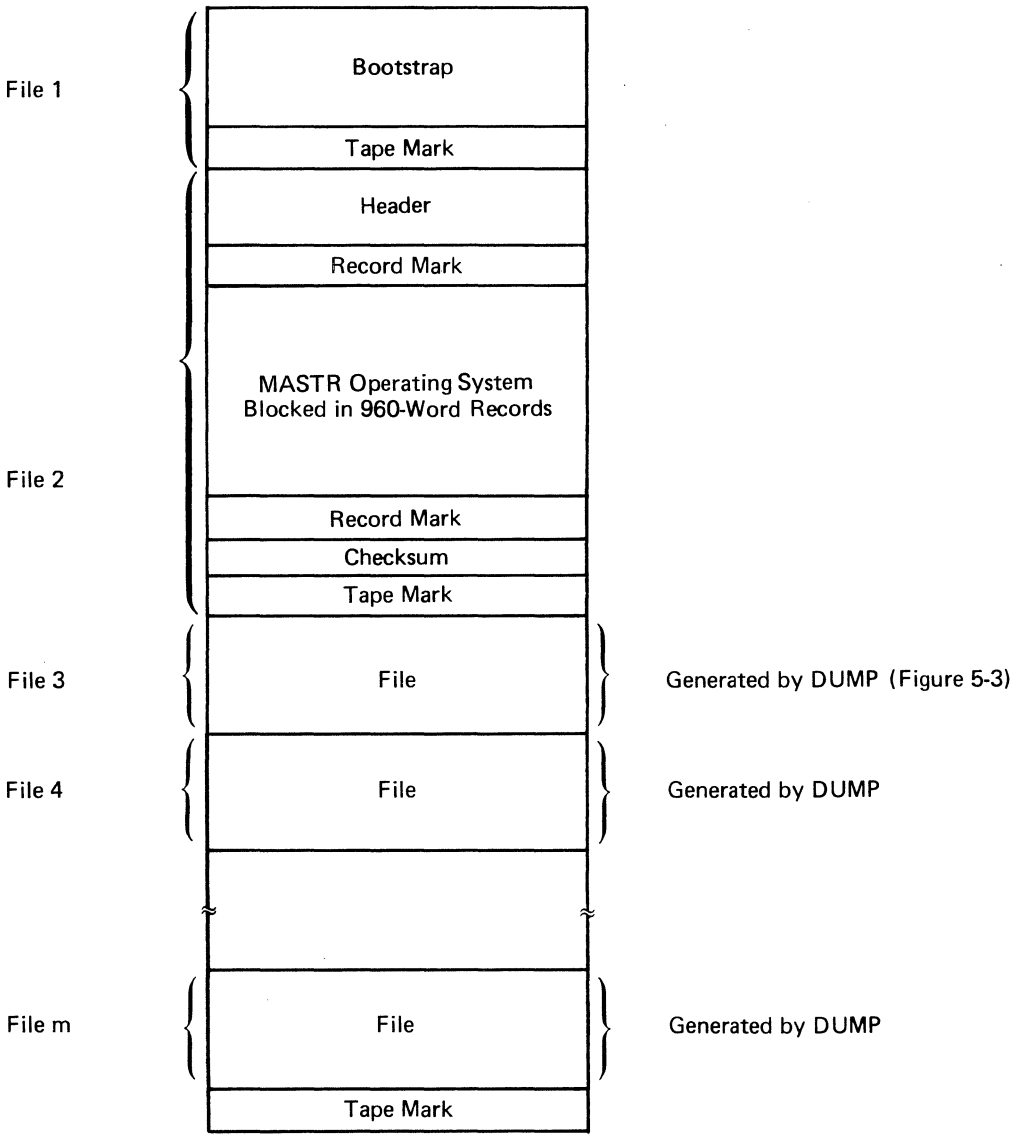
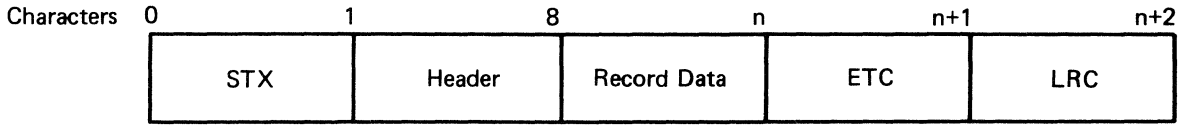


Figure 5-7 MBUP Tape Format

5.4 FILES TRANSFERRED TO THE INTEGRATOR

5.4.1 Record Description

The maximum size of a record is 124 characters for a string record and 40 words for a binary record. Either record is appended with an eight-character record header before transmitting to the Integrator. The header is stripped off by the driver when it is sent from the Integrator. A leading protocol character, STX, and trailing protocol characters, ETC and LRC, also are appended or stripped by the driver.



Character		Record Format
STX	0	Start transmission
Header	1	FST system code = ASCII 0
	2	Sub addr = Station 1, 2, 3, 4 in ASCII or ASCII 8 for background
	3	Destination code = ASCII 0
	4	Sub addr = same as word 2
	5	Character count for binary record (3 per word), 0 for string record
	6	Message type
	7	Spare
	8	Spare
Record Data	n	Data
ETC	n + 1	End of transmission
LRC	n + 2	Checksum

5.4.2 CLI/CLO Blocked Binary Files

Files transferred to and from the Integrator by programs LOAD or DUMP are in blocked binary format. All file types described in paragraphs 5.6 through 5.10 can be transferred in the format shown in figure 5-8.

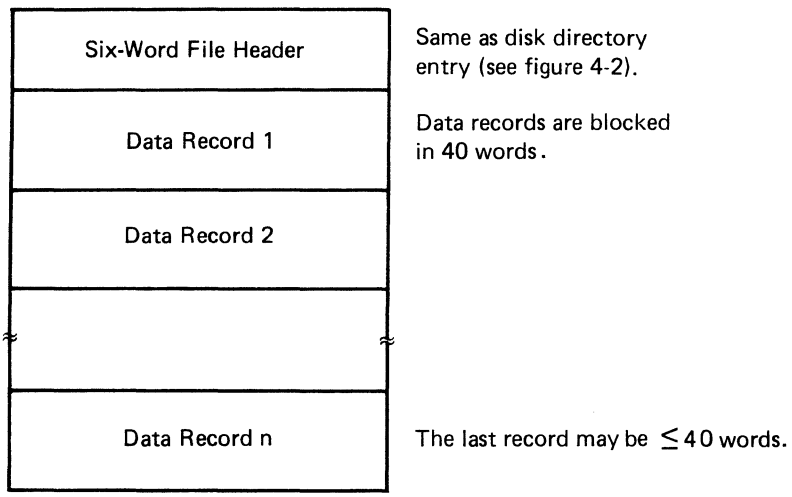


Figure 5-8 CLI/CLO Blocked File Format

5.4.3 CLI/CLO Unblocked String Files

Files transferred to and from the Integrator by the program COPY are unblocked and contain variable-length string records. Only the string files (paragraph 5.6) can be transferred in the format shown in figure 5-9.

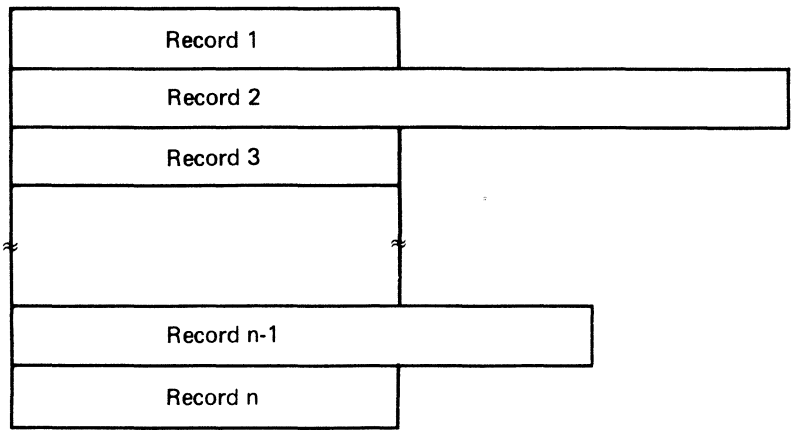


Figure 5-9 CLI/CLO Unblocked File Format

5.4.4 CLI/CLO Variable Length Data File

The files CLO or DATALOG to CLO are created by FACTOR WRITE statements and contain variable-length data records (See paragraph 5.7 and figure 5-10). The record size is limited to 40 words.

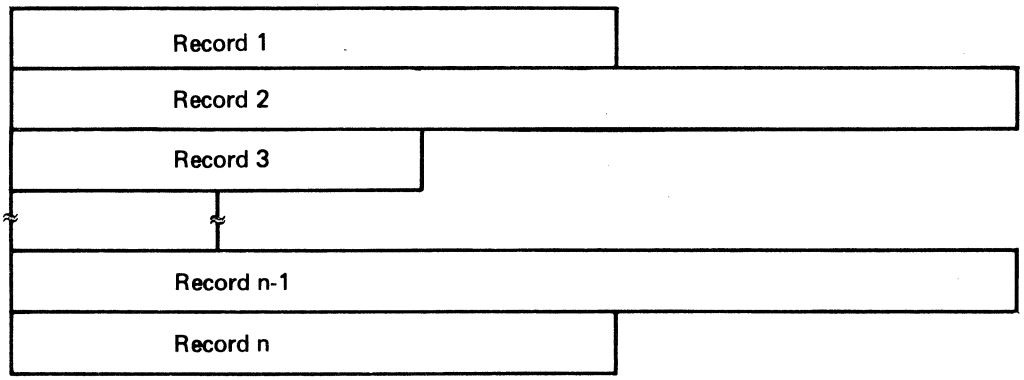


Figure 5-10 CLI/CLO Variable Length Data File Format

5.5 FILES STORED IN MEMORY

Since memory has no physical breaks, any file loaded or created in memory resides contiguously from the starting address to the end address.

MACTAB is a file directory (figure 5-11) that has information on the files currently in memory. There are many more file types in memory than on disk or magnetic tape. This is because once a file is loaded into memory, it can be executed or used to test a device. It is desirable to indicate allowed usage to minimize operator error. File types indicated for files in memory are subsets of file types provided to disk or magnetic tape files.

They are mapped as follows:

Type In Memory	Type On Disk	Description
OVLY	COREIMAGE	Assembly language program executable as an overlay
TP	DATA	FACTOR test program executable on a station or loadable by LMLOAD overlay
MOD	DATA	FACTOR module program callable or loadable by LMLOAD overlay.
S	STRING	String/source file
DATA	DATA	File written by a FACTOR program or by an overlay
OBJ	OBJECT	A file generated by Assembler
U Undefined	COREIMAGE	Coreimage file that is not an overlay or a working storage file that has not been assigned as a particular file type.

Formats for these files are described in sections 5.6 through 5.10. Internally, file types are used by a code which maps with the file type code supplied in the test program header (paragraph 5.8) or the overlay header (paragraph 5.9). Refer to the system memory map, figure 5-12.

8 words per entry, 32 entries in table
Total 256 words. The table is expandable by ASSIGN command.

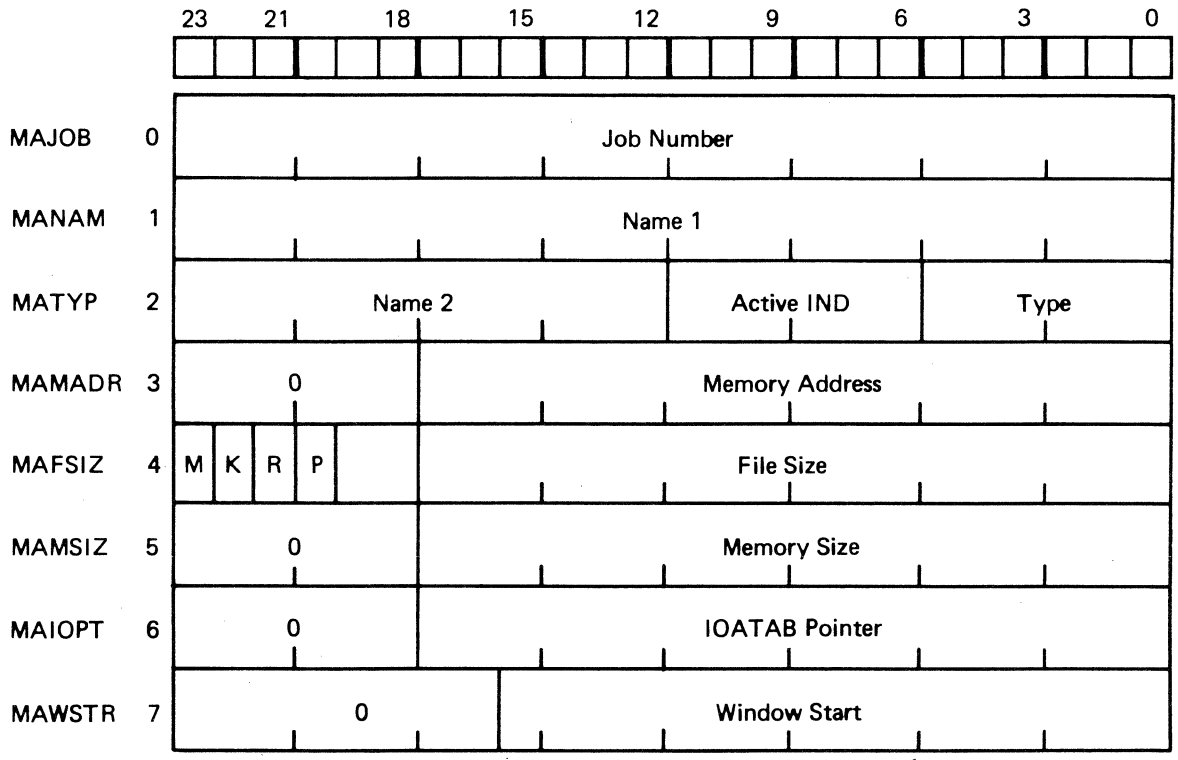


Figure 5-11 MACTAB (Memory Activity Table)

Job Number	**Job number of the file**
NAME1, NAME2	Six-character file name 0 = not used, NAME 1 = -1 = Working Storage
Active IND	File is active when non-zero Bit 6 file attached to STAT1 Bit 7 file attached to STAT2 Bit 8 file attached to STAT3 Bit 9 file attached to STAT4 Bit 10 overlay currently running Bit 11 file is open for I/O

Type	File type code from file header (octal) 13 to 40 System overlay 71 Overlay in background, expands, or generates memory files. 72 DATA 73 OBJ 74 STRING 75 Module test plan 76 Test plan 77 ALLINK overlay 0 Undefined
Memory Address	Starting address of the file in memory
M (Bit 23)	Indicates that this file is in memory NAME 1 = 0 and M = 0 occurs only for PAGE test plan which is bumped out of memory
K (Bit 22)	Keep indicator 1 The file was loaded with KEEP option or from non-disk media 0 The file was loaded from disk without KEEP option
R (Bit 21)	Pending to be released
P (Bit 20)	Partial indicator 0 complete file is in memory 1 partial test plan due to paging is in memory
File size	Number of words required to hold the file.
Memory size	Number of words in memory actually allocated to the file. If greater than file size, file has extra memory (expanded). If less, file is forced to page. If zero, file not in memory.
IOATAB pointer	Starting disk sector address for test programs loaded from disk/IOATAB address for files used as memory files
Window start	Instruction count at the first address of the file area for paging test programs.

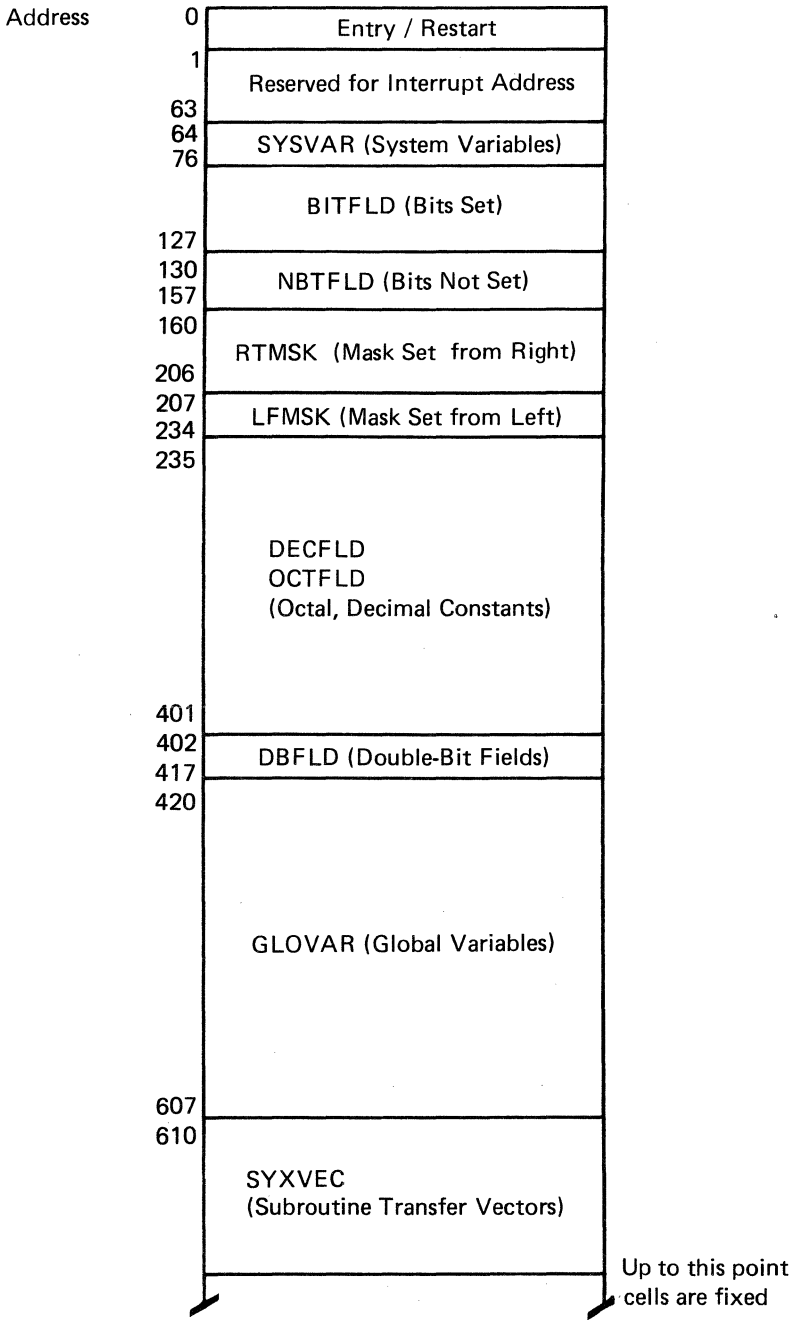


Figure 5-12 System Memory Map

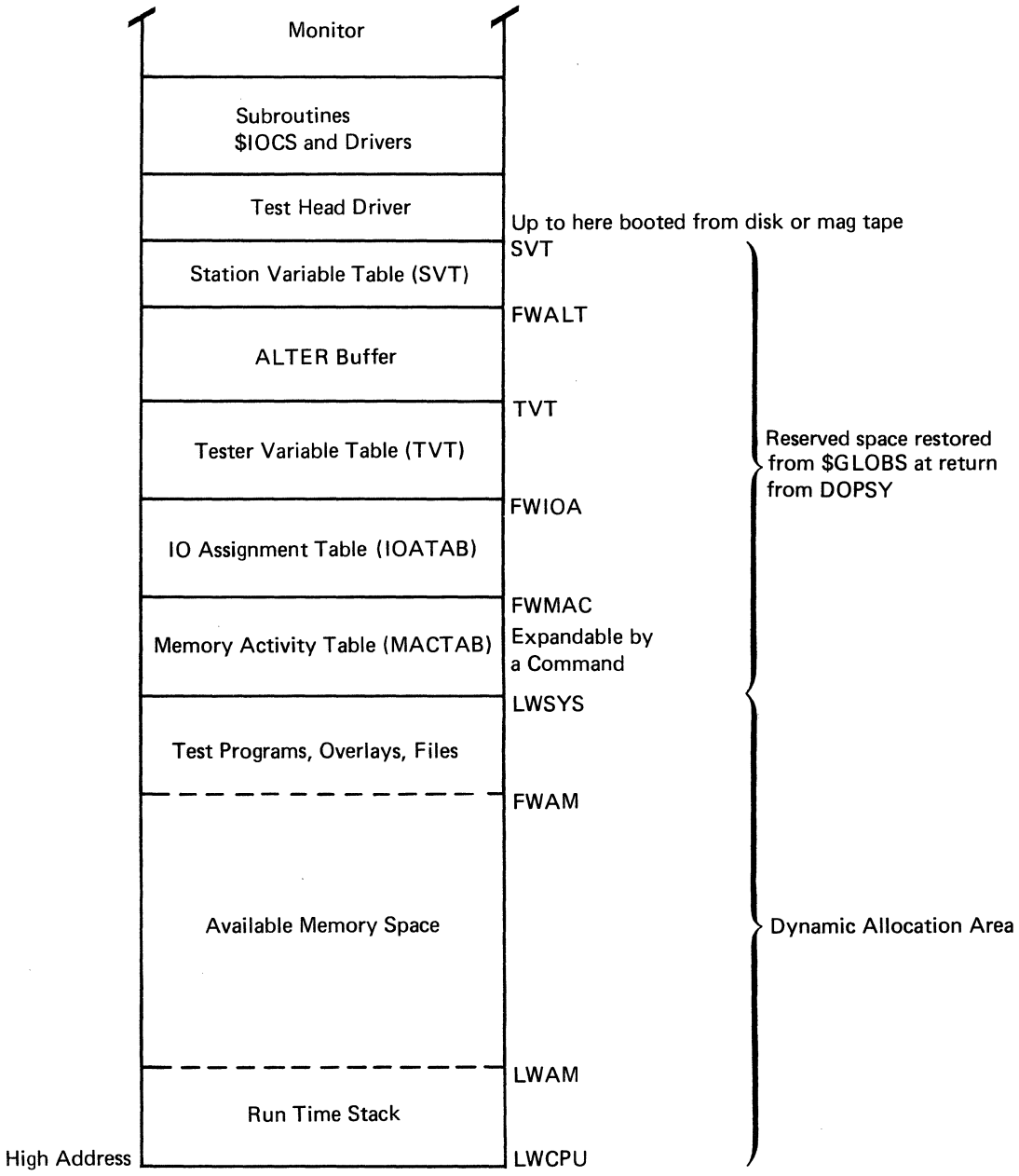


Figure 5-12 System Memory Map (Continued)

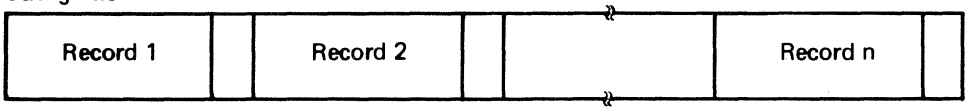
5.6 STRING FILES

String files may be an assembly language source, a FACTOR test program source, a command file, or any file containing TRASCII data.

A string record contains 0 to 132 TRASCII characters terminated by 77B (a backarrow character).

The terminator 77B is always right justified in the last word of a record. The record may contain up to three blank characters. All other trailing blanks are eliminated.

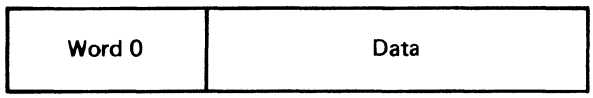
String File



5.7 VARIABLE LENGTH RECORD DATA FILES

Data files opened by the command OPEN are variable in length and are written by FACTOR WRITE statements, datalog in binary format, or by overlays. To be a variable record format can be specified during OPEN procedure and blocking or unblocking can be done using the word count in each record.

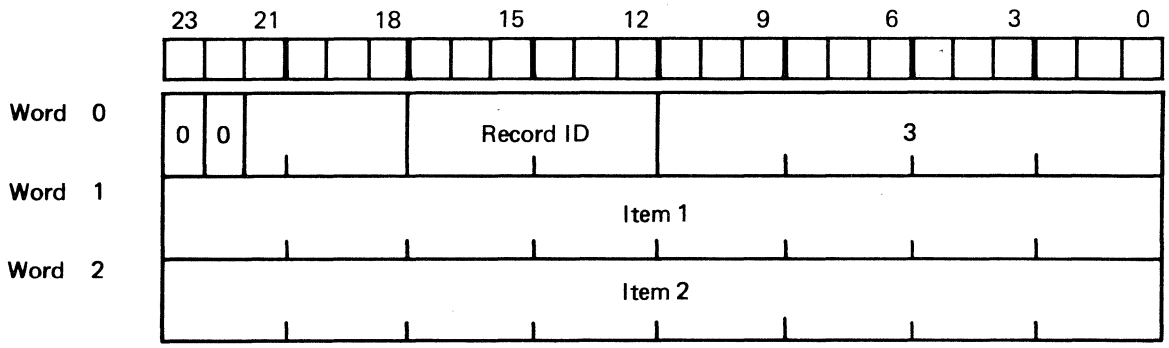
5.7.1 General Record Format



Word 0 contains the following information:

Bit	Description
0 to 11	Record length in words, 4096 words maximum
12 to 17	Record Type. Each datalog record carries a two-digit number that specifies the type of record. (See table 5-1) Bits 12 to 14 specify the particular record type with the class, for example, dc measurement, voltage mode or failure Bits 15 to 17 specify a record class, such as DCMEAS
18 to 21	Spare, not used.
22	1 User-defined record 0 System-defined record
23	1 Format record 0 DATA record

A data record from the datalogger with two items would appear as:



5.7.2 File Format for Variable Length Data

The disk, memory, or magnetic tape files generated by DATALOG or FACTOR WRITE statements contain a header record that is generated during OPEN output. Since it is skipped over during open input, the FACTOR test program does not see this record. If an overlay is written to read this file, it must handle this record. If an overlay generates a file to be read by a FACTOR program, this record must be generated. File formats are shown in figure 5-13.

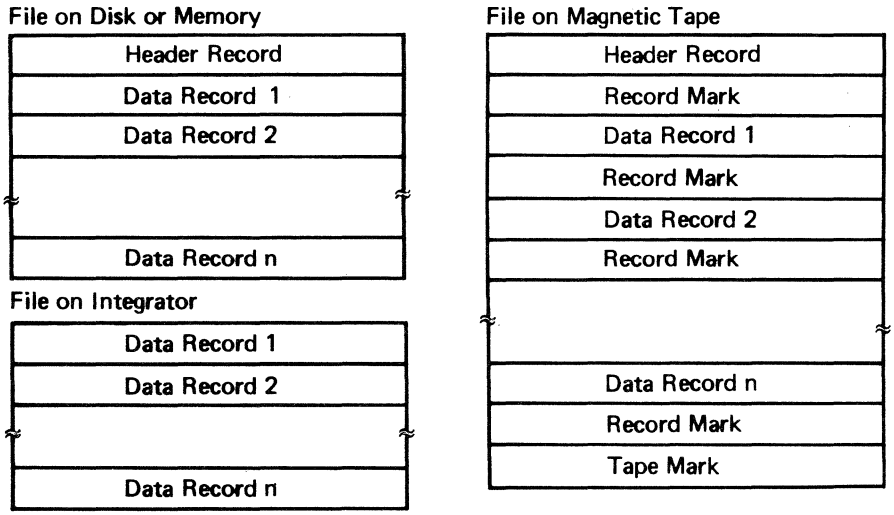


Figure 5-13 File Formats For Variable Length Data

5.7.3 Header Record Format

The header record (figure 5-14) is prepared at the time of opening an output by requesting log, device, and category information. This data is required at the Integrator in order to sort the data properly. They are sent to the Integrator as part of the file identification for opening the file, but are stored as the first record for all other media.

Each item is left justified in each position of words provided.

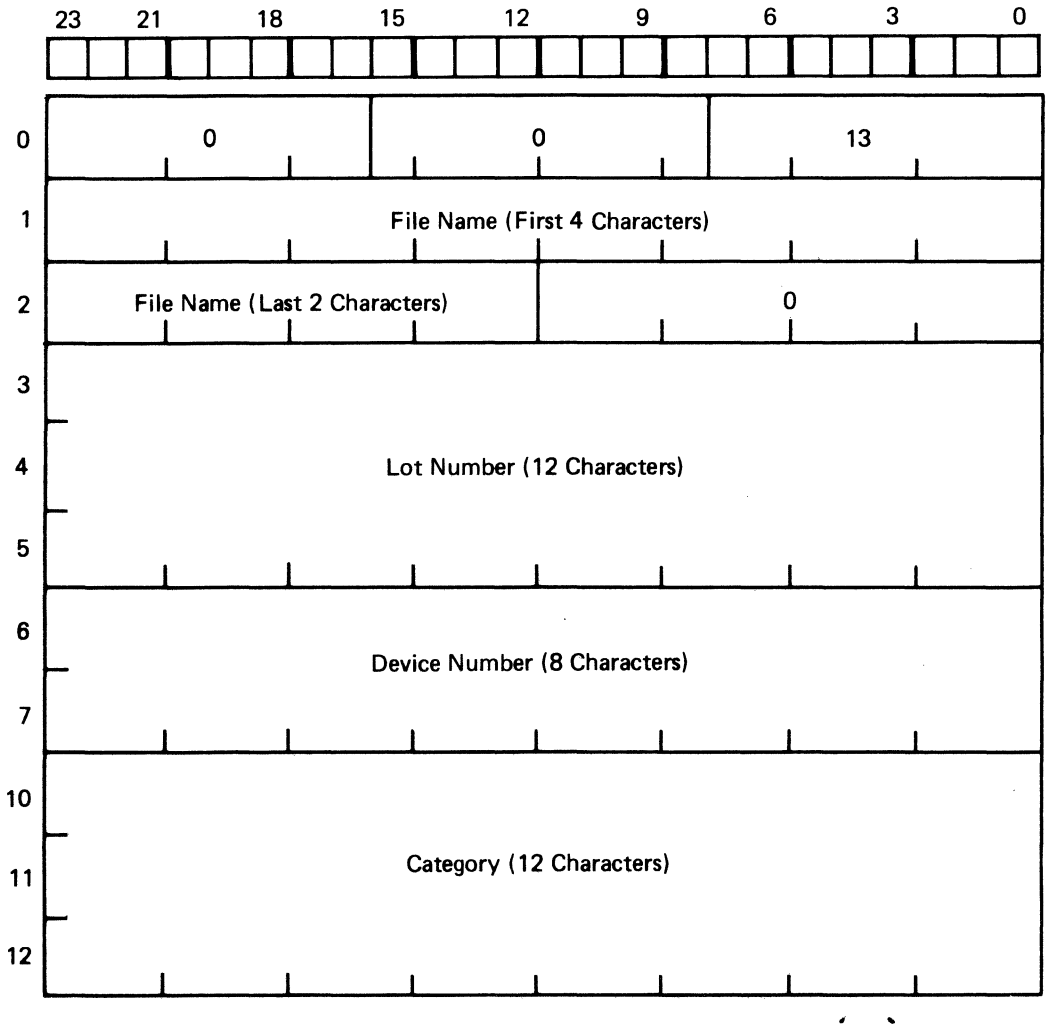


Figure 5-14 Header Record

5.7.4 Datalog Record Formats

Measurements are sent as FST 24-bit floating point numbers. Records with ASCII information are sent with four 6-bit TRASCII characters per word. Integer and central information varies depending on the record. See each record description for specific format (table 5-1).

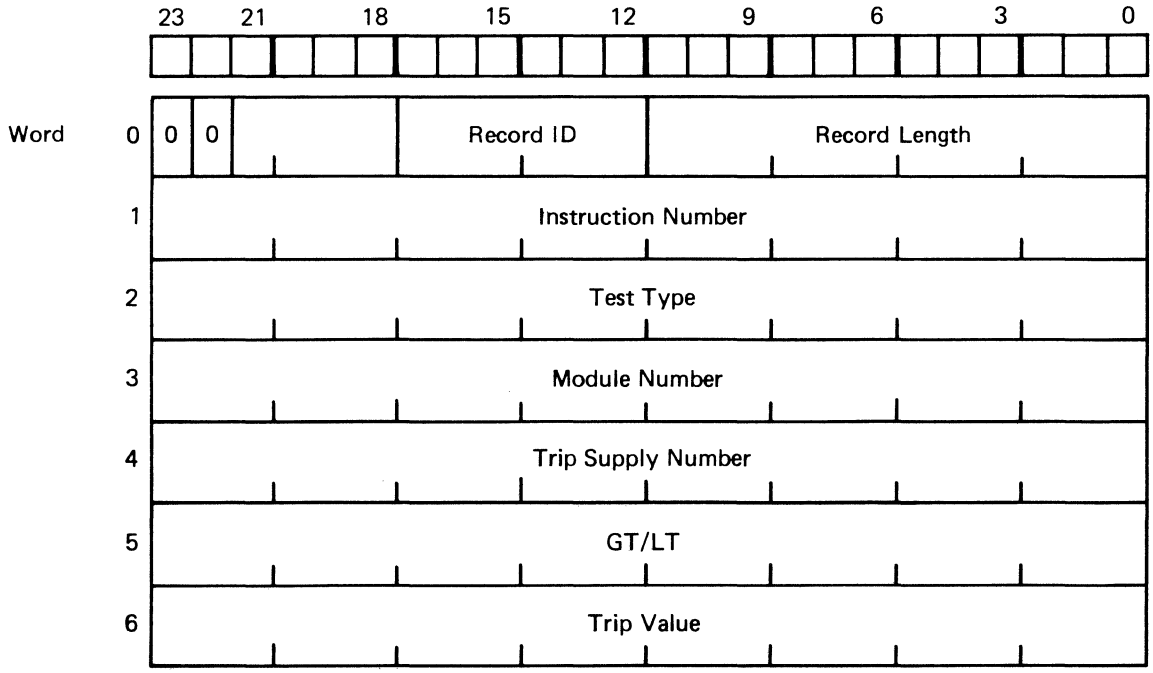
Table 5-1 Record ID For Datalog Records

Record ID Number	Record Is From
00	Open header record
01	DPS voltage trip
02	DPS current trip
10	Direct current fail - current force mode
11	Direct current fail - voltage force mode
12	Direct current pass - current force mode
13	Direct current pass - voltage force mode
14	Measure variable
20	Functional failure
21	PPM memory fail
22	PPM memory fail, data extension
23	Functional failure, messages
30	EOT record
31	Device header
40	Shmoo plot
50	Data I/O

5.7.4.1 DPS TRIP FAIL RECORD

The DPS trip fail record (figure 5-15) is generated when DATALOG TRIP is requested and a power supply trip occurs.

Each power supply trip is written as a separate record.

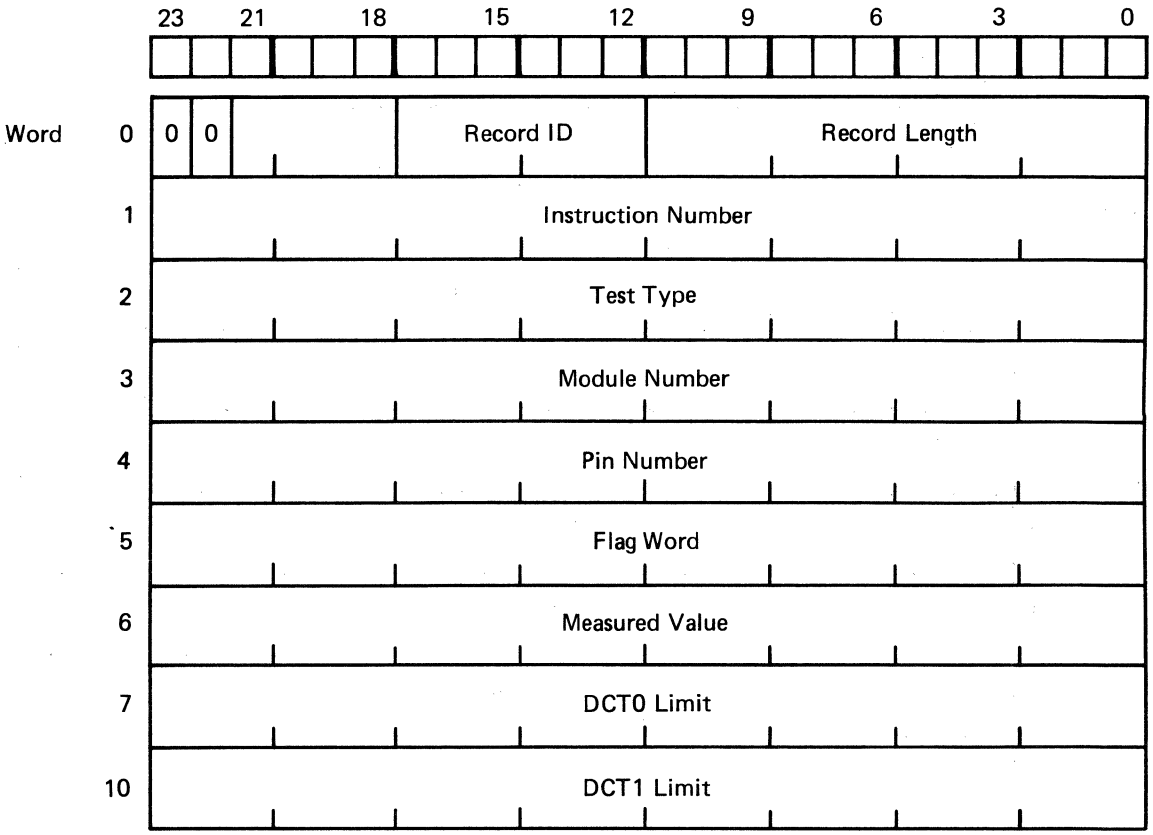


Word	Description
0	Record ID = 02 Current TRIP Record ID = 01 Voltage TRIP Record length = 7 FST words
1	Instruction number = 16-bit unsigned integer
2	Test type = 16-bit unsigned integer
3	Module number = 16-bit unsigned integer
4	Trip supply number = 16-bit unsigned integer
5	GT/LT = TRASCII LT or GT
6	Trip value = 24-bit floating point number

Figure 5-15 DPS Trip Fail Record Format

5.7.4.2 DIRECT CURRENT FAIL RECORD

The direct current fail record (figure 5-16) is generated when DATALOG DCT/ MEAS/LOG is requested and the measurement fails.

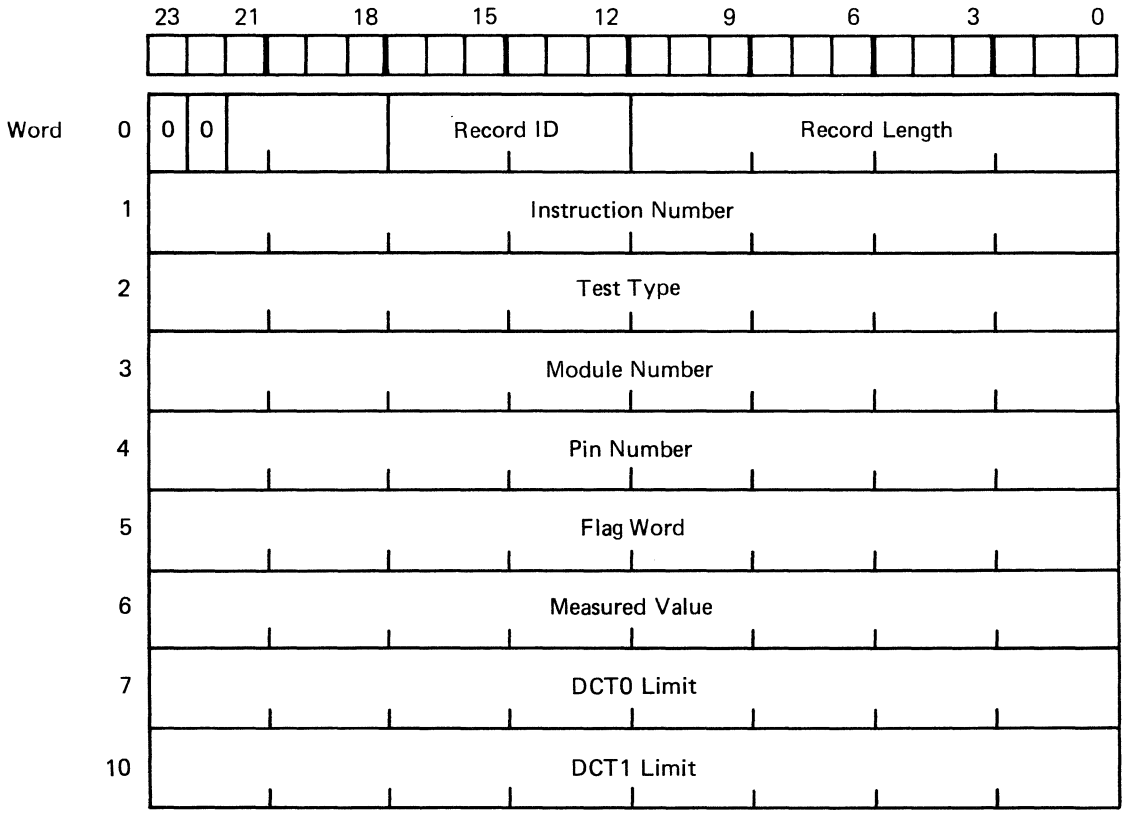


Word	Description
0	Record ID = 10 current force/voltage measure mode Record ID = 11 voltage force/current measure mode Record length = 9 FST words
1	Instruction number
2	Test type = 16-bit unsigned integer
3	Module number = 16-bit unsigned integer
4	Pin number
5	Flag word - ASCII formatting information - cell SMF of SUT table
6	Measured value = 24-bit floating point number
7	DCT0 = 24-bit floating point number
8	DCT1 = 24-bit floating number

Figure 5-16 Direct Current Fail Record Format

5.7.4.3 DIRECT CURRENT PASS RECORD

The dc PASS record (figure 5-17) is generated when the DATALOG MEASURE/LOG is requested and the dc measurement passes.



RECORD ID = 12 Current mode
13 Voltage mode
14 Measure variable

Remaining record fields are the same as a dc fail record, (figure 5-16)

Figure 5-17 Direct Current Pass Record Format

5.7.4.4 FUNCTIONAL FAILURE RECORD

The functional failure record (figure 5-18) is generated when DATALOG FCT is requested and a functional failure occurs.

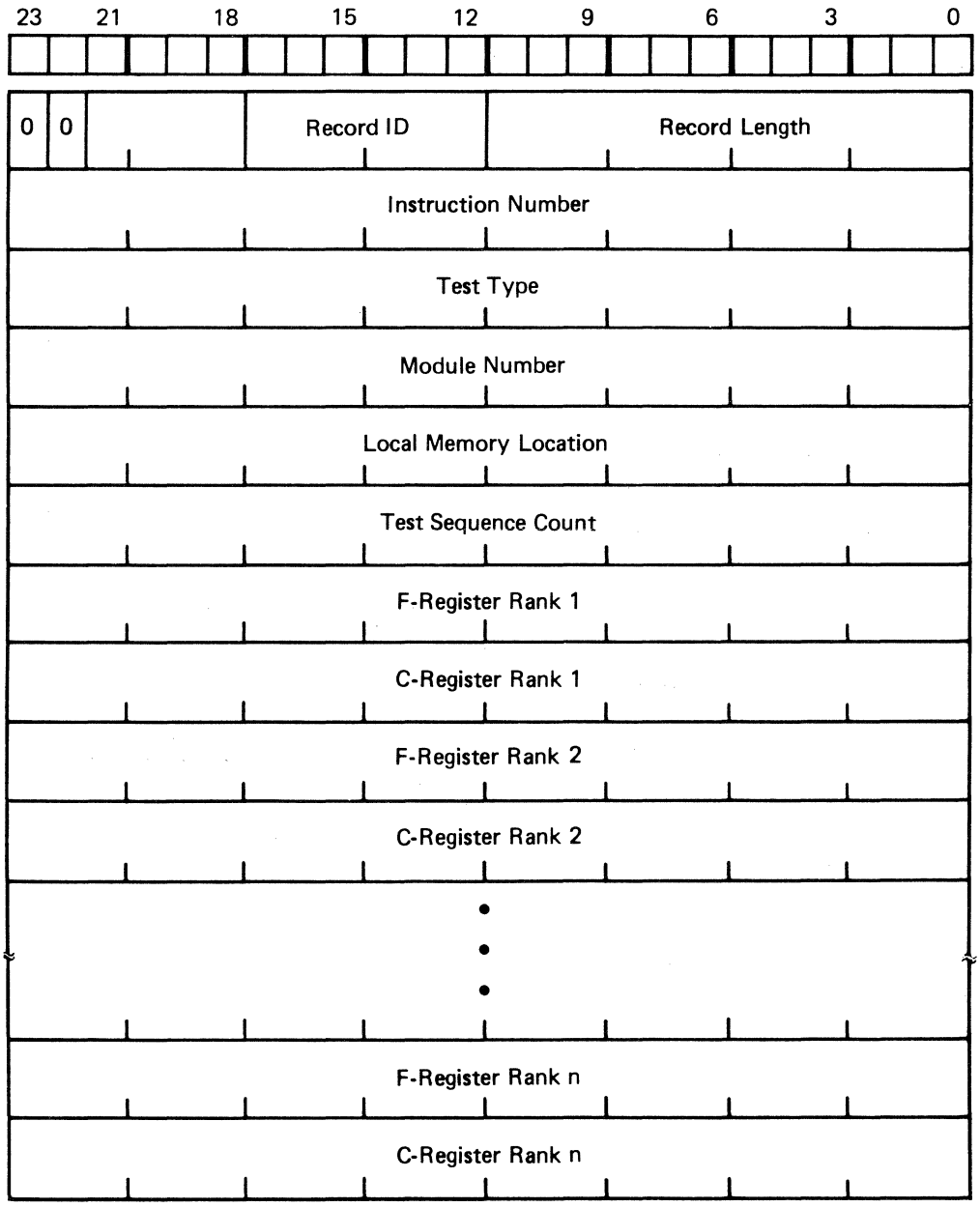


Figure 5-18 Functional Failure Record Format

Word	Description
0	Record ID = 20 Record length = 14 to 22 words (depends on number of ranks used)
1	Instruction number = 16-bit integer
2	Test type = 16-bit integer
3	Module number = 16-bit integer
4	Local memory location = 12-bit integer
5	Test sequence count = 24-bit integer

The C and F register data is contained in bits 0 to 14.

Figure 5-18 Functional Failure Record Format (Continued)

5.7.4.5 FUNCTIONAL FAILURE RECORD FAIL MESSAGE

The functional failure record fail message (figure 5-19) is generated for dc timeout, loop count or clock burst count failure.

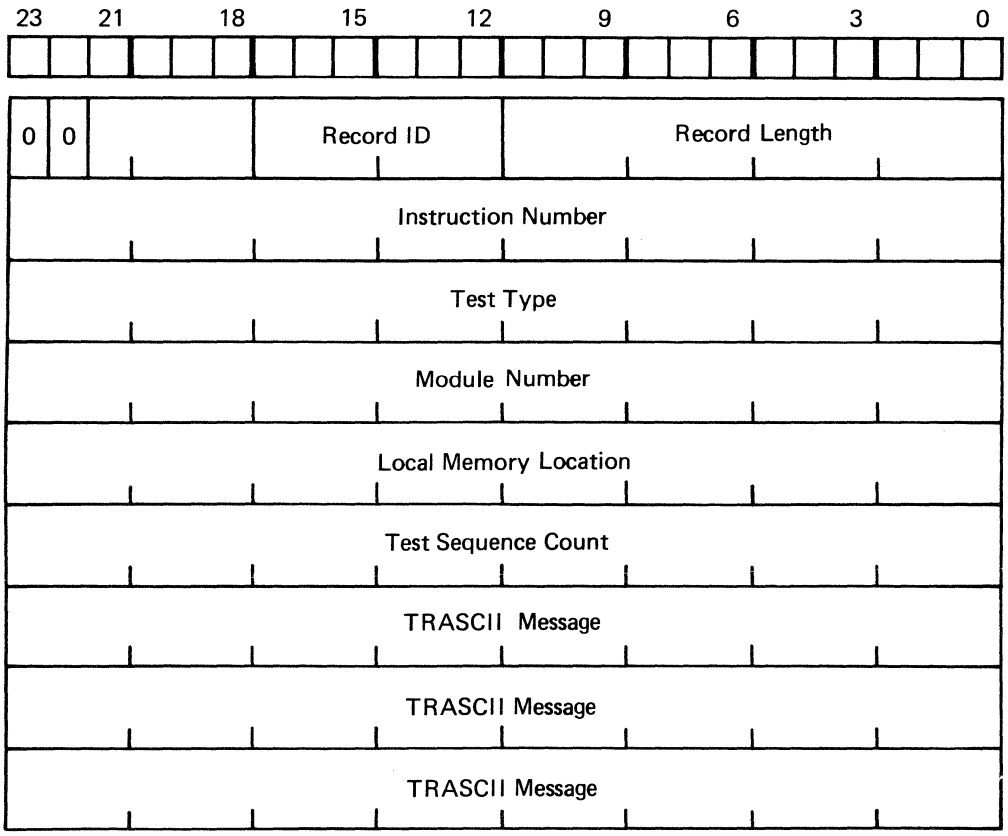


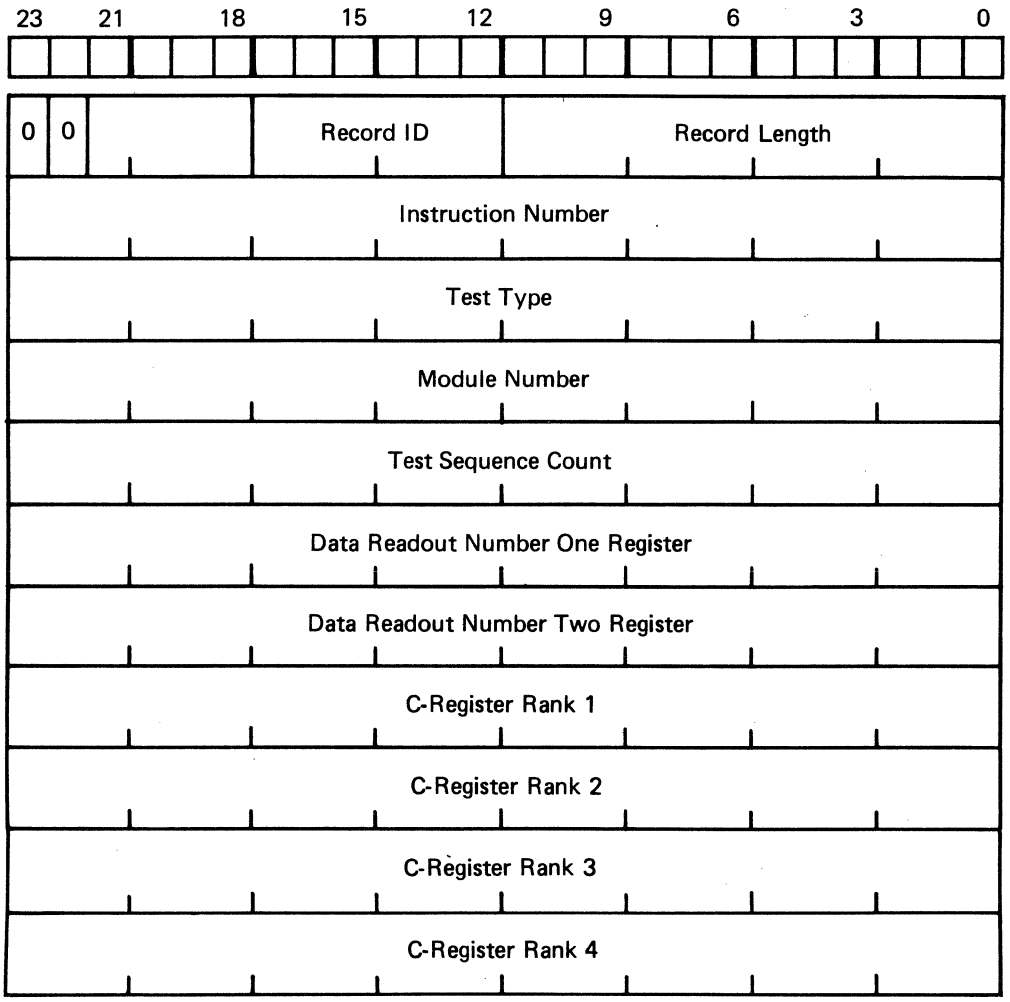
Figure 5-19 Functional Failure Record Fail Message Format

Word	Description
0	Record ID = 23 Record length = 9 FST words
1	Instruction number = 16-bit integer
2	Test type = 16-bit integer
3	Module number = 16-bit integer
4	Local memory location = 12-bit integer
5	Test sequence count = 24-bit integer
6 to 9	TRASCII messages are: T/O FC LOOP T/O, FC T/O, LOOP Each message is three words, with four 6-bit TRASCII characters per word.

Figure 5-19 Functional Failure Record--Fail Message Format (Continued)

5.7.4.6 PPM MEMORY FUNCTIONAL FAILURE RECORD

The PPM memory functional failure record (figure 5-20) is generated when a function fail in an MUT occurs.



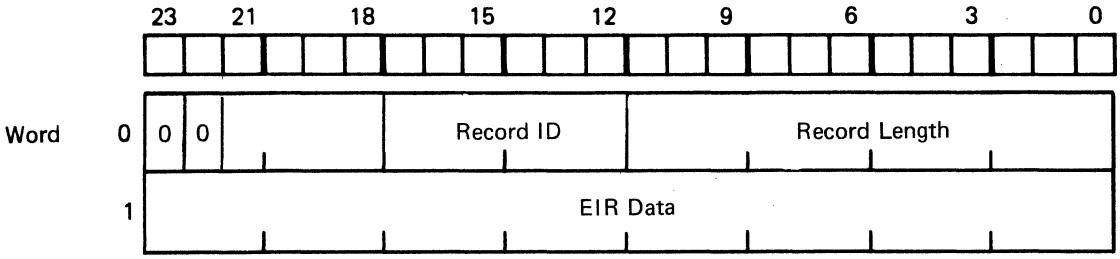
Word	Description
0	Record ID = 21 (nondata extension mode) Record ID = 22 (data extension mode) Record length = 11 words
4	Test sequence count = 24-bit integer
9 to 10	C - Register Ranks 3 and 4 contain information for data extension mode only.

Except for ID and test sequence count, all words are 16-bit integers

Figure 5-20 PPM Memory Functional Failure Record Format

5.7.4.7 EOT RECORD

The EOT record (figure 5-21) is generated when end-of-test point is reached and DATALOG EOT is requested.



Record length = 2
Record ID = 30
EIR data

Bit	Description
0 to 9	User defined value
10	DC test fail
11	DC test pass
12	Functional test fail
13	Functional test pass
14	End of test

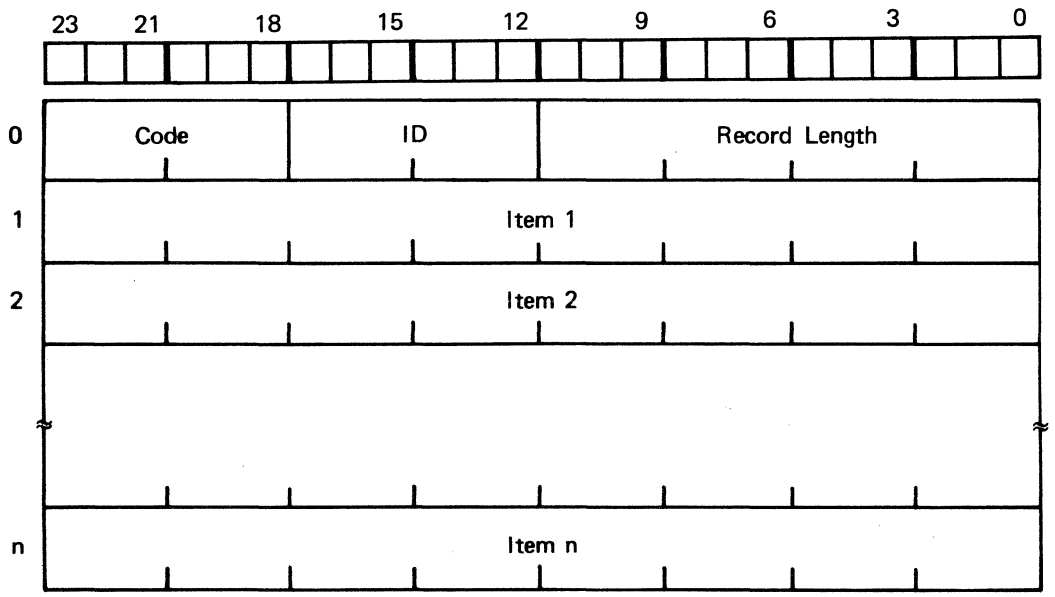
Figure 5-21 EOT Record Format

5.7.5 Writing From A FACTOR Program

A FACTOR program may cause data to be written by one of the following types of statements:

- WRITE "ident" ARRAY;
- WRITE "ident" variable; variable,...;
- WRITE "ident" 'TEXT';
- WRITE "ident" 'TEXT', V1, V2, V3, 'TEXT', V4, V5, V6;
- WRITE "ident" /n/ 'TEXT', variable;

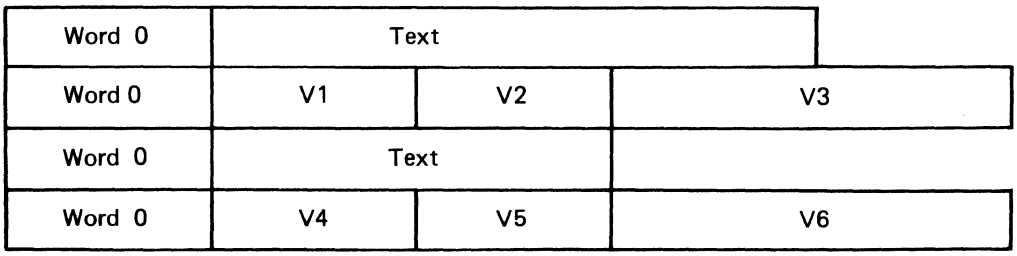
The first two statements above result in the following format:



Word 0:

Record length	number of elements in the array plus 1 (4,096 maximum) or number of variables plus 1 (33 maximum).
ID	TRASCII character in double quotes
Code	010000 (binary)
Word 1 to n	floating point numbers

The last three statements involve a combination of binary and TRASCII data and may cause more than one record to be generated.

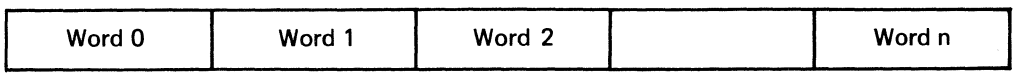


Word 0 has the same format as the first two FACTOR program statements.

Record length — number of words in text plus 1 or number of variables plus 1 (maximum 33 words).

5.8 FIXED LENGTH DATA FILES

Fixed length data files contain data in words rather than characters. The number of words per record can be obtained in the disk directory or file headers (figure 5-2). The record length of 18 words is used as default.



RECORD = n + 1 words

Test programs and modules are normally considered fixed-length data files. However, these files have additional defined format. The first 18 words are the test program header; each word is used according to definition. The FACTOR compiler generates this header for test programs and modules. The extended FACTOR compiler generates this header for macro modules. LMLOAD overlay generates this header for local memory data modules.

Test Program, Module Header

Word	0	Reserved
	1	First four TRASCII characters of the name
	2	Last two TRASCII characters of the name (left justified
	3	File type: 76B for test programs, 75B for modules
	4	Release revision number
	5	Program size (number of words used)
	6 to 16	Reserved
	17	Contains APM vector table pointer

5.9 COREIMAGE FILES

Coreimage files contain data in words. This data may be absolute or relocatable, depending on the data and the usage of the file. Programs run under DOPSY are absolute coreimage and overlays run under MASTR are relocatable coreimage. Normally absolute coreimage files are not transferrable and remain on disk. The format of a relocatable coreimage file is identical to the one of an absolute coreimage file except that it does not use interrupt sectors.

COREIMAGE FILES (FILE TYPE = 011)

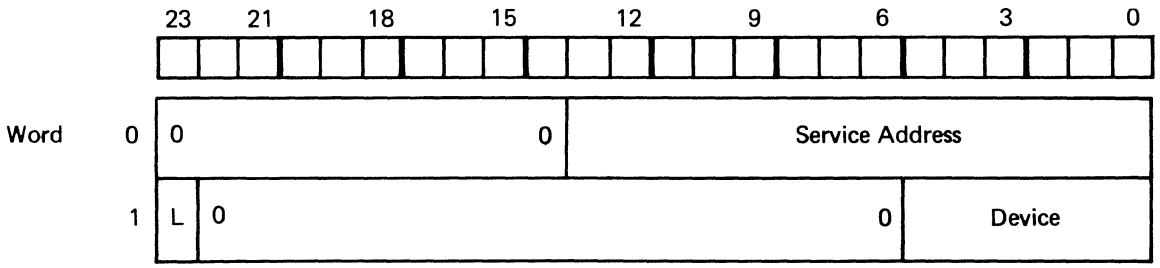
Coreimage Files (File Type = 011)

Interrupt Sectors

The first three sectors may contain interrupt addresses. The number of sectors used for interrupts is specified in the directory word 5 bits 1 to 0. Interrupt sectors contain two words of information per interrupt. (48 words by 3 sectors/2word = 72 interrupts per program maximum).

Program body is already mapped to memory (1 word of data = 1 word of memory when loaded).

INTERRUPT INFORMATION IN INTERRUPT SECTORS



Word	Bit	Description
0	13 to 0	Address of interrupt service routine
1	23	= 1 last interrupt entry
	5 to 0	Interrupt address of the device

5.9.1 Overlay Header Format

A MASTR overlay is a relocatable coreimage file beginning with a 22-word overlay header.

Words 22 to 27 contain information used by the VERIFY overlay to display REL and date.

Table 5-2 describes the contents of each of the words that comprise the header.

Table 5-2 Overlay Header Format

WORD 0	0
1	The first four characters of overlay name in TRASCII
2	The last two characters of overlay name in TRASCII
3	File type code
4	Release revision number in TRASCII
5	File length including header (words)
6	Expansion size (words) at load time
7	Reserved for system use
10	Module relocation value store
11	Module return address store
12	PZE for module entry point
13	BRU for module entry point
14	PZE for reset/kill entry
15	BRU for reset/kill process
16	PZE for special purpose entry
17	BRU for special purpose entry
20	PZE for release overlay entry
21	BRU for release overlay process

Table 5-2 Overlay Header Format (Continued)

WORD 22	PZE for normal background entry
23	BRU for normal background entry
24	PROC for foreground entry
25	BRU for foreground entry
26	Number of words in release notes
27 to 33	Release notes (date)

5.9.2 File Type Code List

Table 5-3 shows the file type code list.

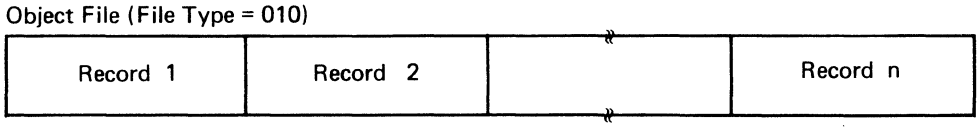
Table 5-3 File Type Code List In Memory

Code	Files
0 to 12B	Undefined
14B	
15B	
16B	
17B	
20B	Manual analysis
21B	
22B	
23B	Parameter distribution
24B	Datalogger
25B	
26B	DEBUG
27B	
30B	
31B	
32B	
33B	
34B	
35B	
36B	
71B	Overlay that takes long time to execute (EDIT), expands (COMPILE), or generates memory files (ASM, COPY)
72B	Data files
73B	Object files
74B	STRING files
75B	Module test programs
76B	Test programs
77B	General overlay

5.10 OBJECT FILES

Object files are generated by assembler and used by CREATE to generate coreimage files. An object file contains relocation directives as well as executable CPU instructions.

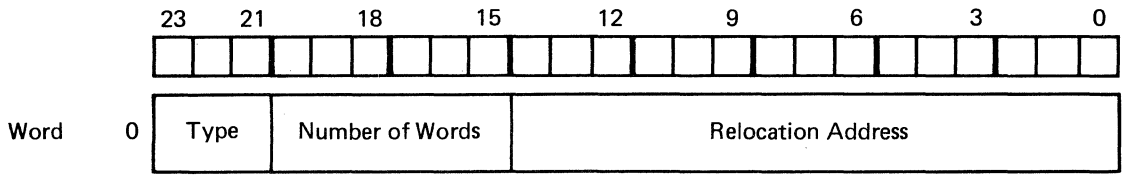
OBJECT FILE (FILE TYPE = 010)



Variable Size Record

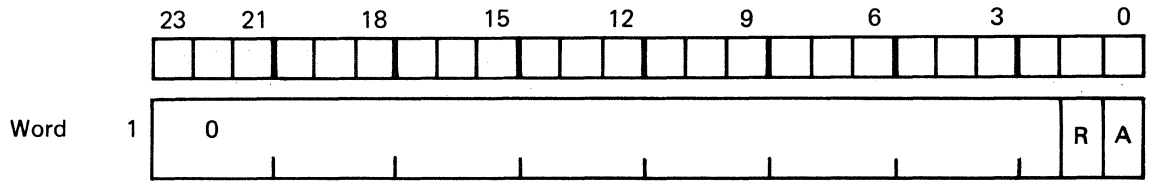
Record: Contains from 2 to 6 words, depending on the record type

The following format is used for word 0 in all records, except record type 4 and 5.



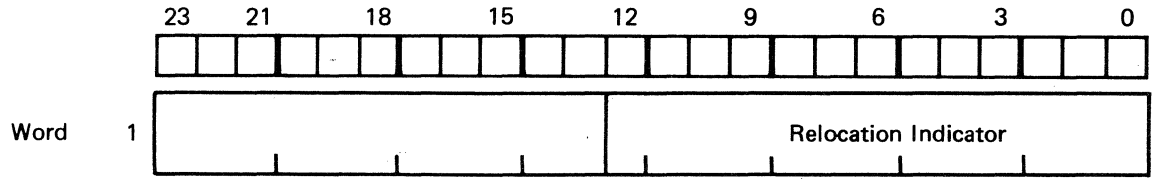
	Bit	Description
Word 0	23 to 21	Record Type 0 = START (first object record) 1 = DATA (Instruction record) 2 = PROC 3 = CALL 7 = END
	20 to 15	Number of data words in this record excluding word 0 and word 1
	14 to 0	Relocation address to which the data words are loaded.

Record Type = 0 (START)



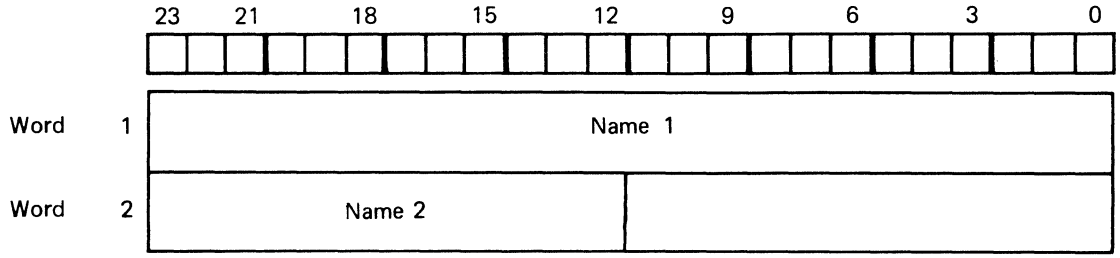
	Bit	Description
Word 1	0	0 No ABS directive (software relocatable) 1 Absolute program (ABS directive)
	1	0 No REL directive (not relocatable by hardware) 1 Hardware relocatable (REL directive)

Record Type = 1 (DATA/INSTRUCTION)



	Bit	Description
Word 1	23 to 0	Instruction in record is relocatable if corresponding bit is set.
Word 2	Word n	Contains instruction words.

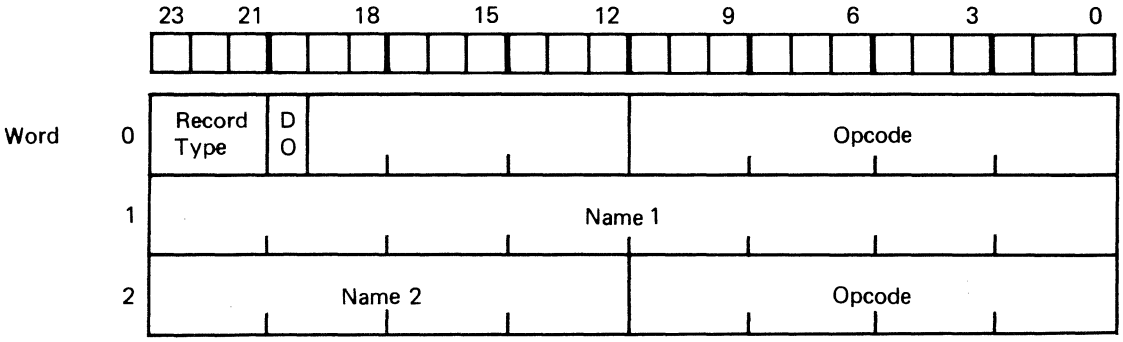
Record Type = 2 or 3 (PROC or CALL)



	Bit	Description
Word 1	23 to 0	First four character symbol name (TRASCII)
Word 2	23 to 12	Last two character symbol name (TRASCII)

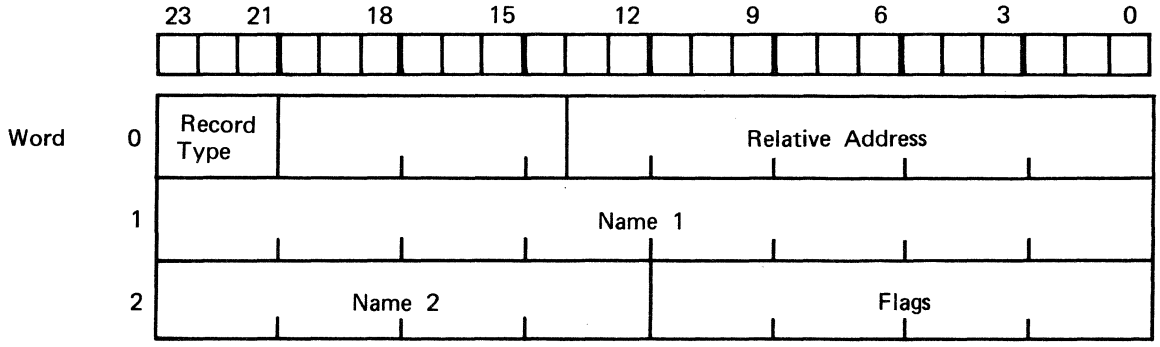
Record types 2 and 3 contain no other data.

Record Type = 4 (EXT)



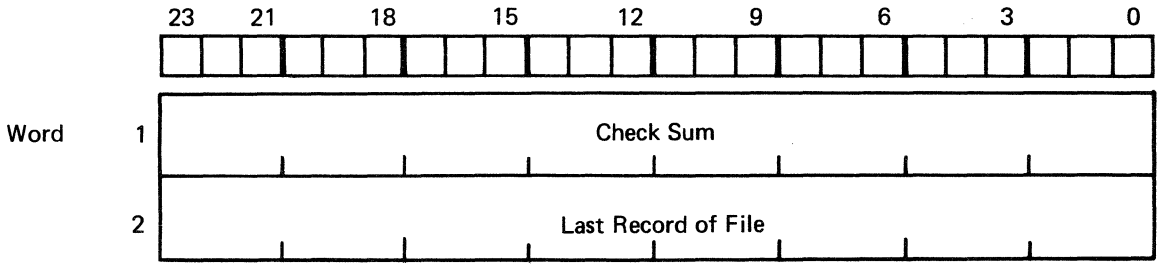
	Bit	Description
Word 0	23 to 21	Record type
	20	D/O data = 1, opcode = 0
	11 to 0	Right half of opcode
Word 1	23 to 0	First four characters symbol name (TRASCII)
Word 2	23 to 12	Last two characters symbol name (TRASCII)
	11 to 0	Left half of opcode

Record Type = 5 (ENT)



	Bit	Description
Word 0	23 to 21	Record type
	13 to 0	Relative address of symbol
Word 1	23 to 0	First four characters symbol name
Word 2	23 to 12	Last two characters symbol name
	11 to 0	Flags

Record Type = 7 (END)



	Bit	Description
Word 1	23 to 0	Checksum of the file excluding type 7 record. Bit 23 is always set to 1. Checksum is obtained from summing all words in the file and 40000000B is ORed at the end.
Word 2	23 to 0	Contains 77777777B to indicate last record of file.

6

Assembly Language Overlays

6.1 INTRODUCTION

Assembly language overlays (ALLINK programs) are user-written assembly language programs that are executed either through the EXEC statement of a FACTOR program or as a background function. Executing ALLINK programs through the EXEC statement initiates and performs data collections or manipulations by communicating between a FACTOR program and/or provides communications between a FACTOR program in foreground and associated background processing. Executing assembly language programs as a background function performs I/O operations, for example, outputting reports or receiving information from the keyboard.

ALLINK programs are written in Fairchild Assembly Language for the FST-2 computer. The instruction set and the assembler are described in the FST-2 Computer Manual, publication number 57000002.

6.1.1 Foreground/Background Processing

Foreground processing refers to Tester-related operations. Thus, the FACTOR program and the section of the ALLINK program executed through the EXEC statement are referred to as operating in the foreground.

Background processing refers to non-Tester-related operations which are relatively slow, such as I/O operation. During foreground processing, there is a certain amount of hardware idle time, such as relay switching and power supply settling time. During this idle time, background performs its operation. Thus, the slower background operations are performed without affecting test time. Obviously, I/O operations in foreground processing would decrease Tester throughput significantly.

6.1.2 Risks Involved in ALLINK Programs

ALLINK programs that perform functional or dc failure testing have a direct effect on the results from system routines such as datalog or DCF. Device failure, whether it occurs in a FACTOR test program or an ALLINK program, causes the interrupt service to set flags, which are the basis for decisions by system routines. Thus, if an ALLINK program does perform this type of testing, the results from system routine may be meaningless.

Since any ALLINK program executed by the system cannot be completely controlled by the system software, any undebugged ALLINK program inadvertently may alter the hardware and software and thus, alter or destroy the integrity of the system software. Therefore, it is highly recommended that ALLINK programs be debugged outside of production environments.

6.2. ALLINK PROGRAM DEFINITION

The system requires the ALLINK program to have a header. See paragraph 5.9.1 for a description of the overlay header format.

6.2.1 ALLINK Header

The header consists of 27B words at the beginning of the ALLINK program and contains the system information for loading and executing the overlay. The header must follow the format given in appendix A, and must have an OBJ 7 and REL directives in front of the header. There are three program entry points. The mode under which the ALLINK program is executed determines which of the entry points is used.

The foreground entry point is used to transfer control from a FACTOR test plan (through the EXEC statement) to the foreground section of the overlay.

The background entry point is used when the monitor receives a program-name command for the overlay.

The release entry point is used when the command: RELEASE is entered to remove the ALLINK program from memory.

The reset entry point is used when the tester is reset manually (reset pushbutton).

Example Entry Points:

```
RESET   PZE     0       RESET ENTRY POINT
        BRU*    RESET   RETURN IMMEDIATELY
        BSS     2
RELSE   PZE     0       RELEASE ENTRY POINT
        BRU*    RELSE   RETURN IMMEDIATELY
BGEP    PZE     0       BACKGROUND ENTRY POINT
        BRU     BGSTRT  GO TO BACKGROUND
                        PROCESS ROUTINE
FGEP    PZE     0       FOREGROUND ENTRY POINT
        BRU     FGSTRT  GO TO FOREGROUND
                        PROCESS ROUTINE
```

This example shows foreground, background, reset, and release entry points with no clean-up when the overlay is released (the release is always performed when requested). There is no foreground processing in this program.

```
RSENT   PZE     0
        BRU*    RSENT
        BSS     2
RELSE   PZE     0
        BRU     CLRUP       GO TO RELEASE
                            PROCESS ROUTINE
BGEP    PZE     0
        BRU     BGSTRT
FGEP    PZE     0
        BRU*    FGEP        NO FOREGROUND PROCESS
```

6.2.2 Creating Relocatable Coreimage

After an ALLINK program is assembled and an object file is created, a relocatable coreimage file can be created with the following command in DOPSY:

```
CREATE 'name' 'obj-file-name'
```

Rules and Restrictions:

- o REL directive is provided at the beginning of the SOURCE program or REL parameter is supplied in the ASM command.
- o The program must be originated at location 0. This will happen by default if an ORG directive is not included at the beginning of the program.

6.3 LOADING AND CALLING PROCEDURES

6.3.1 Loading Procedure

Before the ALLINK program can be called by either foreground or background, it can be loaded using the monitor LOAD command.

Example:

```
LOAD 'DBTS'
```

This command causes the ALLINK program DBTS to be loaded from disk.

Once loaded, the program can be called from either foreground or background. If the ALLINK program is called by a command or by the EXEC FACTOR statement, and if it has not been loaded into memory, the system tries to load automatically. If there is not enough room, a terminal error occurs at EXEC execution point.

6.3.2 Calling Procedure

An ALLINK program is called from the foreground by a FACTOR EXEC statement. EXEC statements must have the following format:

EXEC program-name (v_1,v_2,...v_n)

Where program-name is the name of the ALLINK program in relocatable coreimage form. A maximum of 63 parameters is allowed. Each parameter is evaluated at the time of EXEC and may be global variables, user-defined variables, array elements, formal parameters or arithmetic expressions.

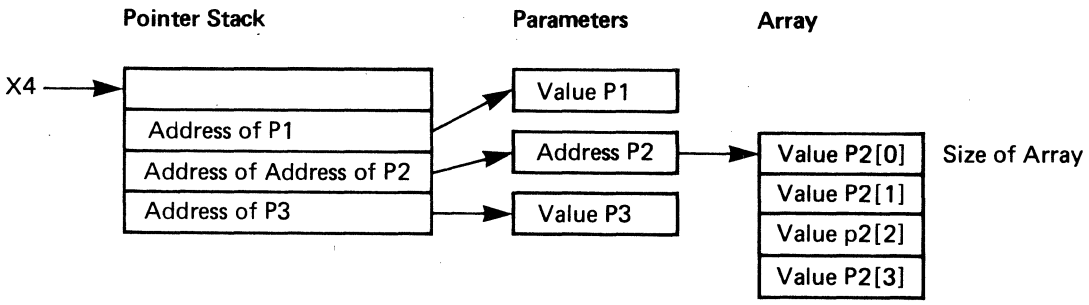
Upon entry to the ALLINK program, the index registers contain the following information:

X3 number of parameters passed
X4 address of 1st parameter -1
X6 station code -1 (i.e. code 0 to 3 = station 1 to 4)

The FACTOR statements:

```
DCL P2 [3];   REM NOTE P2 IS AN ARRAY;
.
.
.
EXEC XTEST (P1,P2,P3);
```

would cause pointers and variables to be mapped as shown:



To obtain and save P1 and P3 use the following instruction sequence:

```
LDA*    1,X4        GET 1ST PARAMETER
STA     TEMP1       SAVE IT
LDA*    3,X4        GET 3RD PARAMETER
STA     TEMP3       SAVE IT
```

To obtain element 3 of array P2, use:

```
LDA*    2,X4        GET ADDRESS OF ARRAY
LXA     X5          MOVE TO X5
LDA     3,X5        GET 3RD ELEMENT
```

If it is known whether a passed parameter is a variable or an array, then the determination can be made by checking bit 22 of the word in the pointer stack. For example, to check P2 for an array use the following instruction sequence:

```
LDA     2,X4        GET P2 POINTER
CAM     B22         IF B22 SET
BBC     ARRAY       THEN IT IS AN ARRAY
```

To obtain the number of parameters, the instruction

```
LAX     X3
```

is executed.

6.3.2.2 BACKGROUND CALL

The background section of ALLINK program can be entered by the command:

program-name (input) (output) optional parameters

where program-name is the name of the ALLINK overlay and the input and output devices are optionally assigned to the program. Absence of input/output dvices causes default to the PID/POD for any I/O operations. Allowable input/output mnemonics are shown in the Sentry VII Users Manual, publication 57000013.

Example:

EXAMPLE LP

This command would cause the ALLINK program (overlay) named EXAMPLE to be executed in the background mode using the line printer as its output device.

6.3.2.3 RELEASE CALL

This section of the ALLINK program can be entered by a command:

RELEASE 'program-name'

There are three possible release procedures:

- No release processing is needed. The ALLINK program should return to the system immediately as is shown in the first example entry point (see paragraph 6.2.1).
- A clean-up process is necessary before release in order to maintain system/user program integrity. The entry to this user clean-up process may be accomplished as is shown in the second example entry point (see paragraph 6.2.1). This type of process may be done, for example, to restore system words that are used by the ALLINK program but that are not automatically restored by the system.
- The ALLINK processing is incomplete and the program should not be released at the time the release command is entered. For example, data saved by the ALLINK program has not been completely analyzed.

The ALLINK program should display a message to the operator and return to the system at CALLed address +2.

Example:

```
CLRUP    EQU     *
         .
         .
         .
         AOM     RELSE          can not release yet
         BRU*    RELSE
```

6.3.2.4 OVERLAY ACTIVE/INACTIVE CALL

In addition to the command: RELEASE, any overlay can be released automatically by MASTR in a disk based system when the memory space is needed.

When this condition occurs, the system searches overlays and test programs in memory and tries to release inactive ones. It is the responsibility of the overlay to indicate if it is active or inactive. To indicating being active, the overlay must call ATTA subroutine (SYXVEC + 64) and DTTA ((SYXVEC + 65) when it is no longer needed for a station. An active indicator for each station is shown in NAME command output. During the automatic release process, the overlay is also called at RELEASE entry point.

Prior to calling the background part of an overlay, the system automatically indicates that it is active and clears it to inactive upon background return.

6.4 SCHEDULING A BACKGROUND TASK FROM FOREGROUND

Within the ALLINK program, the foreground section may call a routine in the background section in order to perform some background-type processing; for example, to print a message on data collected by foreground.

Note that while these sections are part of the same overlay, they perform as separate tasks. Foreground initiates this type of call through the scheduler. If the foreground and the background tasks share a common data area, the foreground task has to wait to allow the background task to finish using the data area.

6.4.1 Flagword

A flagword must be provided to indicate background or foreground activity. This word is a local variable; it is defined and used exclusively within the ALLINK program. The flagword is turned on by foreground processing and turned off at the end of the background processing so that the next time the foreground task comes in, it can tell if the background process is done. Refer to figure 6-1 for a typical ALLINK program.

6.4.2 Foreground Procedure

At the beginning of the foreground, entry is made from a test program by an EXEC statement.

1. Test the activity flag to see if the background task is completed. If the flag is ON, call the scheduler wait routine, FGWAIT (refer to SYXVEC + 44). This makes the foreground task wait one complete scheduler cycle in order to give the background time to finish.

2. At the return from FGWAIT, test the flag again.

3. If the flag is OFF, start the foreground task.

   Example:

```
FG      EQU     *       FOREGROUND START ADDRESS
        LDA     FLAG    BACKGROUND DONE?
        BZ      *+3     YES, START FG PROCESS
        BSM*    FGWAIT  NO, THEN WAIT
        BRU     *-3     CHECK FLAG AGAIN
        --              Start foreground
                        process here
```

At the end of foreground processing:

1. Set the activity flag ON.

2. Call the task scheduling routine FGBGRT (refer to SYXVEC + 41) with the background process address in index register 7.

3. Return to test program execution.

Example:

```
*FG PROCESSING
            LDA     D1
            STA     FLAG          SET ACTIVITY FLAG
            LDX     X7,BGAD1      GET ADDRESS OF
                                  TASK AND SET IN THE
            BSM*    FGBGRT        SCHEDULER
            BRU*    FGENTR        RETURN TO TEST PLAN THROUGH
*                                 THE FOREGROUND ENTRY POINT.

BGAD1       EQU     *             THIS MUST BE THE BACKGROUND
                                  PROCESSING START ADDRESS
```

6.4.3 Calling Background

When the background is called because foreground scheduled an activity:

1. Complete the necessary background processing, (I/O data processing, etc. I/O usage and restrictions are described in section 3).

2. At the end of processing, clear the activity flag.

3. Exit from background through the global transfer VECTOR COMMAND (refer to SYXVEC + 38).

Example:

```
BGAD1       EQU     *
            .
            .                     Any processing
            .
            .
            .
            .
            LDA     D0            Clear the flag
            STA     FLAG
            BRU*    COMMND        Return
```

6.5 PROCESSING A COMMAND THROUGH MONITOR FROM FOREGROUND

To force the monitor to process a command from the prepared buffer the following steps are taken.

1. At the background entry:

   - Save the global PIDPMF (GLOVAR + 16)
     It contains the address of PID entry in IOATAB
   - Save the global CMDPMF+1 (GLOVAR + 43)
     It contains the address of the command record buffer of the current PID.
   - Lock out the current PID from reading another command (bit 13 of the first word of IOATAB entry must be set to 1)

2. Just before returning to monitor through background entry point:

   - Clear the command lock out.
   - Place a required command in the record buffer using the address saved.
   - Set the global PIDFLG (GLOVAR + 39) to the saved PIMPMF (the address of the entry IOATAB).

```
 FST MASTR ASSEMBLER  REL   2.1          00:03     SOURCE= *EXM2
ARR       00000177    ARRCNT   00000146    B22       00000110
BGFIN     00000264    BGM1     00000237    BGMSG     00000232
BGSTAR    00000267    BUFIN    00000033    CLSE      00000314
COMFLG    00000105    COMMND   00000706    D1        00000106
D205      00000112    D4       00000107    EXMPLE    00000024
FFIX      00000731    FG1      00000160    FG2       00000205
FGBGRT    00000711    FGENT    00000147    FGEX      00000221
FGWAIT    00000714    FP9      00000111    FWORD     00000000
GLOVAR    00000420    IERMSG   00000676    LWORD     00000315
MOVEIT    00000301    MSG      00000030    MSGIN     00000646
MSGOUT    00000647    MSGS     00000113    OUTBUF    00000057
OVBGEP    00000022    PARMS    00000145    PARX      00000104
PODPMF    00000441    PRCHAR   00000026    RECBUF    00000024
RELSE     00000020    STARS    00000120    STNO      00000032
SVX6      00000103    SYXVEC   00000640    TABEND    00000144
TABLE     00000121    TE205    00000230    TOF       00000027
WRITR     00000307    X1       00000001    X2        00000002
X3        00000003    X4       00000004    X5        00000005
X6        00000006    X7       00000007    $IOCS     00000645
```

Figure 6-1 Sample ALLINK Program

```
                *    THIS IS A SAMPLE ALLINK PROGRAM NAMED 'EXMPLE'
                *    IT CAN BE LOADED USING THE SYSTEM COMMAND
                *
                *        // LOAD 'EXMPLE'
                *
                *    THE FOREGROUND SECTION IS CALLED BY THE FACTOR STATEMENT
                *
                *        EXEC EXMPLE (PARAMETER);
                *
                *    THE FOREGROUND SECTION CALLS THE BACKGROUND ROUTINE
                *    BGMSG WHICH OUTPUTS A MESSAGE BASED ON THE PARAMETERS
                *    PASSED TO IT BY FOREGROUND
                *
                *    THE BACKGROUND SECTION CAN BE CALLED WITH THE COMMAND
                *
                *        // EXMPLE
                *
                *    IT OUTPUTS A PROMPTING CHARACTER $, READ INPUT FROM PID
                *    AND OUTPUTS IT TO POD
                *
                *
                *
                          OBJ    7
                          REL                MAKE IT RUN-TIME RELOCATABLE
                *
                ********************************************
                *                                          *
                *         OVERLAY HEADER                   *
                *                                          *
                ********************************************
                *
00000 00000000  FWORD     DATA   0              0
00001 45705560            TEXT   'EXMPLE'       1,2    PROGRAM NAME
00002 54450000
00003 00000077            DATA   778            3      ALLINK CODE
00004 00000000            DATA   0              4
00005 00000315            DATA   LWORD-FWORD    5      PROGRAM SIZE
00006 00000012            BSS    10             6-15   NOT USED
00020 00000000  RELSE     PZE    0              16     RELEASE ENTRY POINT
00021 41040020            BRU*   RELSE          17     IMMEDIATE RETURN
00022 00000000  OVBGEP    PZE    0              18     BACKGROUND ENTRY POINT
00023 41000267            BRU    BGSTART        19     GO TO BACKGROUND START
00024 00000000  EXMPLE    PZE    0              20     FOREGROUND ENTRY POINT
00025 41000147            BRU    FGENT          21     GO TO FOREGROUND START
                *
```

Figure 6-1 Sample ALLINK Program (Continued)

```
                        PAGE
                    *
                    *  THE FOLLOWING EQU'S LINK TO SYSTEM GLOBALS AND
                    *  TRANSFER VECTORS
                    *
      00000640  SYXVEC   EQU   6408          LOCATION OF TRANSFER VECTOR
      00000645  $IOCS    EQU   SYXVEC+5       IOCS SUBROUTINE
      00000646  MSGIN    EQU   SYXVEC+6       READ FROM PID ROUTINE
      00000647  MSGOUT   EQU   SYXVEC+7       WRITE POD ROUTINE
      00000676  IERMSG   EQU   SYXVEC+30      TERMINAL ERRORS
      00000706  COMMND   EQU   SYXVEC+38      RETURN TO SYSTEM
      00000711  FGBGRT   EQU   SYXVEC+41
      00000714  FGWAIT   EQU   SYXVEC+44
      00000731  FFIX     EQU   SYXVEC+57
                    *
      00000420  GLOVAR   EQU   420B          LOCATION OF GLOBAL VARIABLES
      00000441  PODPMF   EQU   GLOVAR+17     ADDRESS OF POD PMF
                    *
                    *  IO CONTROL WORDS AND BUFFERS
                    *
      00000024  RECBUF   EQU   20            SIZE OF IO BUFFER
00026 00000004  PRCHAR   DATA  4B            PROMPTING CHARACTER $
00027 03000000  TOF      DATA  030000006     TOP OF FORM CODE FOR $IOCS
00030 63645760  MSG      DATA  'STOP',0
00031 00000000
00032 00000001  STNO     BSS   1
                    *
00033 00000024  BUFIN    BSS   RECBUF
00057 00000024  OUTBUF   BSS   RECBUF
                    *
                    *
                    *   PROGRAM DATA STORAGES AND CONSTANTS
                    *
      00000001  X1       EQU   1          INDEX REGISTER 1
      00000002  X2       EQU   2          INDEX REGISTER 2
      00000003  X3       EQU   3          INDEX REGISTER 3
      00000004  X4       EQU   4          INDEX REGISTER 4
      00000005  X5       EQU   5          INDEX REGISTER 5
      00000006  X6       EQU   6          INDEX REGISTER 6
      00000007  X7       EQU   7          INDEX REGISTER 7
                    *
00103 00000000  SVX6     DATA  0
00104 00000000  PARX     DATA  0
00105 00000000  COMFLG   DATA  0          FG/BG BUSY FLAG
00106 00000001  D1       DATA  1
00107 00000004  D4       DATA  4
00110 20000000  B22      DATA  20000000B
00111 21110000  FP9      DATA  21110000B
00112 00000315  D205     DATA  205
00113 00000000  MSGS     DATA  0
00114 21212121           DATA  '1111'
00115 22222222           DATA  '2222'
00116 23232323           DATA  '3333'
00117 24242424           DATA  '4444'
00120 12121212  STARS    DATA  '****'
                    *
                    *
00121 00000024  TABLE    BSS   20
      00000144  TABEND   EQU   *-1
00145 00000000  PARMS    DATA  0
00146 00000000  ARRCNT   DATA  0
                    *
```

Figure 6-1 Sample ALLINK Program (Continued)

```
                    PAGE
                *
                ************************************************
                *                                              *
                *   FOREGROUND SECTION CALLED BY EXEC STATEMENT  *
                *                                              *
                ************************************************
                *
      00000147  FGENT     EQU   *
00147 64000105            LDA   COMFLG     BACKGROUND STILL BUSY?
00150 42200153            BZ    *+3        NO, GO AHEAD AND SEND NEXT MSG
00151 12040714            BSM*  FGWAIT     YES, WAIT
00152 41000147            BRU   FGENT
                *
00153 56300145            STX   X3,PARMS
00154 45700121            LDX   X7,TABLE
00155 45600144            LDX   X6,TABEND
00156 07000604            CLA
00157 54000104            STA   PARX
00160 64000145  FG1       LDA   PARMS
00161 42200221            BZ    FGEX
00162 24400001            LDA   1,X4
00163 63000110            CAM   B22
00164 43040177            BBC   ARR
00165 24440001            LDA*  1,X4
00166 63000111            CAM   FP9        DON'T ALLOW 9
00167 43200230            BE    TE205
00170 76000104            AOM   PARX
00171 77000145            SOM   PARMS
00172 14700000            STA   0,X7       STORE PARAMETER
00173 11400001            ATX   X4,1
00174 11700001            ATX   X7,1
00175 43100160            BL    FG1
00176 41000221            BRU   FGEX       TABLE FULL
                *
00177 24440001  ARR       LDA*  1,X4
00200 11400001            ATX   X4,1
00201 77000145            SOM   PARMS          DECREMENT COUNTER
00202 07500000            LXA   X5         POINTER TO ARRAY
00203 25500000            LDE   0,X5       ARR SIZE
00204 55000146            STE   ARRCNT
00205 11500001  FG2       ATX   X5,1
00206 24500000            LDA   0,X5
00207 63000111            CAM   FP9        DONT ALLOW 9
00210 43200230            BE    TE205
00211 14700000            STA   0,X7
00212 76000104            AOM   PARX
00213 11700001            ATX   X7,1
00214 43600221            BGE   FGEX
00215 77000146            SOM   ARRCNT
00216 64000146            LDA   ARRCNT
00217 42200160            BZ    FG1        END OF ARRAY
00220 41000205            BRU   FG2
                *
      00000221  FGEX      EQU   *
00221 07600200            LAX   X6         SAVE STATION ID
00222 54000103            STA   SVX6
                *
00223 64000106            LDA   D1         TURN ON BACKGROUND BUSY FLAG
00224 54000105            STA   COMFLG
                *
00225 45700232            LDX   X7,BGMSG ADDRESS OF BACKGROUND ROUTINE
00226 12040711            BSM*  FGBGRT     AND SCHEDULE BACKGROUND
```

Figure 6-1 Sample ALLINK Program (Continued)

```
00227 41040024            BRU*  EXMPLE    RETURN TO TESTING
                  *
00230 64000112  TE205     LDA   D205
00231 12040676            BSM*  IFRMSG
```

```
                          PAGE
                  ************************************************
                  *                                              *
                  *   BACKGROUND ROUTINE SCHEDULED BY FOREGROUND  *
                  *                                              *
                  ************************************************
                  *
      00000232    BGMSG     EQU   *
00232 05640441            LDX*  X6,PODPMF SET X6 TO ADDRESS OF POD PMF
00233 45100027            LDX   X1,TOF
00234 12040645            BSM*  $IOCS     DO TOP OF FORM IF POD IS LP
00235 50000235            NOP   *         IGNORE ERROR
00236 45700121            LDX   X7,TABLE
00237 56700146  BGM1      STX   X7,ARRCNT
00240 64000104            LDA   PARX      DONE?
00241 42200264            BZ    BGFIN
00242 24700000            LDA   0,X7
00243 12040731            BSM*  FFIX
00244 42300247            BNZ   *+3
00245 63000107            CAM   D4
00246 43300251            BLE   *+3
00247 64000120            LDA   STARS
00250 41000253            BRU   *+3
00251 07500000            LXA   X5
00252 64500113            LDA   MSGS,X5   GET MESSAGE TO PRINT
00253 54000032            STA   STNO
00254 45100030            LDX   X1,MSG    START ADDRESS OF OUTPUT
00255 05200003            LDX   X2,3      NO. OF WORDS TO OUTPUT
00256 12040647            BSM*  MSGOUT
00257 50000257            NOP   *         IGNORE ERROR
                  *
00260 77000104            SOM   PARX      DECREMENT COUNT
00261 45740146            LDX*  X7,ARRCNT
00262 11700001            ATX   X7,1
00263 41000237            BRU   BGM1
                  *
      00000264    BGFIN     EQU   *
00264 07000604            CLA             CLEAR BACKGROUND BUSY FLAG
00265 54000105            STA   COMFLG
00266 01040706            BRU*  COMMND    RETURN TO SYSTEM
```

Figure 6-1 Sample ALLINK Program (Continued)

```
                    PAGE
                *
                ******************************************************
                *                                                    *
                *   BACKGROUND SECTION CALLED BY AN OPERATOR COMMAND  *
                *    // EXMPLE                                       *
                *                                                    *
                *    IT READS A RECORD FROM PID AND OUTPUTS IT TO POD *
                *       IT EXITS WHEN FOUR ASTERISKS ARE ENTERED     *
                *                                                    *
                ******************************************************
                *
      00000267  BGSTART  EQU   *
00267 45100033           LDX   X1,BUFIN  ADDRESS OF INPUT BUFFER
00270 64000026           LDA   PRCHAR    PROMPTING CHARACTER $
00271 12040646           BSM*  MSGIN     READ A RECORD
00272 50000272           NOP   *
                *
                *  NOW OUTPUT IT TO POD
                *
00273 05400012           LDX   X4,10
00274 05500000           LDX   X5,0
00275 05300002           LDX   X3,2
00276 64000033           LDA   BUFIN     THE RECORD = **** ?
00277 63000120           CAM   STARS
00300 43200314           BE    CLSE      YES, EXIT NOW
                *
      00000301  MOVEIT   EQU   *
00301 64500033           LDA   BUFIN,X5
00302 54300057           STA   OUTBUF,X3 MOVE IT TO OUTPUT BUFFER
00303 11500001           ATX   X5,1      WITH SPACING
00304 43400307           BG    WRITR
00305 11300002           ATX   X3,2
00306 41000301           BRU   MOVEIT
                *
      00000307  WRITR    EQU   *
00307 45100057           LDX   X1,OUTBUF ADDRESS OF OUTPUT BUFFER
00310 05200024           LDX   X2,20     NO. OF WORDS TO OUTPUT
00311 12040647           BSM*  MSGOUT
00312 50000312           NOP   *         IGNORE ERROR
00313 41000267           BRU   BGSTART
                *
      00000314  CLSE     EQU   *
00314 41040022           BRU*  OVBGEP    RETURN TO SYSTEM
                *
      00000315  LWORD    EQU   *
                         END
```

Figure 6-1 Sample ALLINK Program (Continued)

```
00000177    ARR       00164
00000146    ARRCNT    00204 00215 00216 00237 00261
00000110    B22       00163
00000264    BGFIN     00241
00000237    BGM1      00263
00000232    BGMSG     00225
00000267    BGSTAR    00023 00313
00000033    BUFIN     00267 00276 00301
00000314    CLSE      00300
00000105    COMFLG    00147 00224 00265
00000706    COMMND    00266
00000106    D1        00223
00000112    D205      00230
00000107    D4        00245
00000024    EXMPLE    00227
00000731    FFIX      00243
00000160    FG1       00175 00217
00000205    FG2       00220
00000711    FGBGRT    00226
00000147    FGENT     00025 00152
00000221    FGEX      00161 00176 00214
00000714    FGWAIT    00151
00000111    FP9       00166 00207
00000000    FWORD     00005
00000420    GLOVAR    00026
00000676    IERMSG    00231
00000315    LWORD     00005
00000301    MOVEIT    00306
00000030    MSG       00254
00000646    MSGIN     00271
00000647    MSGOUT    00256 00311
00000113    MSGS      00252
00000057    OUTBUF    00302 00307
00000022    OVBGEP    00314
00000145    PARMS     00153 00160 00171 00201
00000104    PARX      00157 00170 00212 00240 00260
00000441    PODPMF    00232
00000026    PRCHAR    00270
00000024    RECBUF    00033 00057
00000020    RELSE     00021
00000120    STARS     00247 00277
00000032    STNO      00253
00000103    SVX6      00222
00000640    SYXVEC    00026 00026 00026 00026 00026 00026 00026 00026
00000144    TABEND    00155
00000121    TABLE     00154 00236
00000230    TE205     00167 00210
00000027    TOF       00233
00000307    WRITR     00304
00000001    X1        00233 00254 00267 00307
00000002    X2        00255 00310
00000003    X3        00153 00275 00302 00305
00000004    X4        00162 00165 00173 00177 00200 00273
00000005    X5        00202 00203 00205 00206 00251 00252 00274 00301 00303
00000006    X6        00155 00221 00232
00000007    X7        00154 00172 00174 00211 00213 00225 00236 00237 00242
                      00261 00262
00000645    $IOCS     00234
```

Figure 6-1 Sample ALLINK Program (Continued)

A

Overlay Header Format File Type Code List

WORD	0	0
	1	The first four characters of overlay name in TRASCII
	2	The last two characters of overlay name in TRASCII
	3	File type code
	4	Release revision number in TRASCII
	5	File length including header (words)
	6	Expansion size (words) at load time
	7	Reserved for system use
	10	Module relocation value store
	11	Module return address store
	12	PZE for module entry point
	13	BRU for module entry point
	14	PZE for reset/kill entry
	15	BRU for reset/kill process
	16	PZE for special purpose entry
	17	BRU for special purpose entry
	20	PZE for release overlay entry
	21	BRU for release overlay entry
	22	PZE for normal background entry
	23	BRU for normal background entry
	24	PZE for foreground entry
	25	BRU for foreground entry
	26	Number of words in release notes
	27 to 33	Release notes (date)

FILE TYPE CODE LIST

in Header Location 3.

Code	File
0	Undefined
20B	Manual Analysis
21B	
22B	
23B	Parameter Distribution
24B	Datalogger
25B	
26B	DEBUG
27B	
30B	
31B	
32B	
33B	
34B	
35B	
36B	
71B	Overlay that takes a long time to execute (EDIT), expands (COMPILE), or generates memory files (ASM, COPY).
72B	Data files
73B	Object files
74B	STRING files
75B	Module test programs
76B	Test programs
77B	General overlay

B

Instruction Mnemonics

B.1 OPCODES SORTED BY ASCENDING ALPHA OPCODES

	Opcode	Mnemonic	Code Description	Cycles
		ABS	ABSOLUTE PROGRAM LOCATOR	
*	20000000	ADD	ADD	
*	26000000	AND	LOGICAL AND	2
*	36000000	AOM	ADD ONE TO MEMORY	4
	064034XX	ARD	ALTERNATE READ	1
	066134XX	ARDS	ALTERNATE READ STATUS	1
	06422400	ASPAC	ALTERNATE SPACE	1
	06401500	ART	ALTERNATE READ RECORD TAPE	1
*	11000000	ATX	ADD TO INDEX	2
*	07000000	AUG	AUGMENT	
*	06423400	AWRIT	ALTERNATE WRITE	1
*	00000000	BAH	BRANCH AFTER HALT	1
*	02000000	BAT	BRANCH ON A-REGISTER TEST	1
*	03040000	BBC	BRANCH BIT COMPARE	1
*	03200000	BE	BRANCH IF EQUAL	1
*	03400000	BG	BRANCH IF GREATER	1
*	03600000	BGE	BRANCH IF GREATER OR EQUAL	1
*	03100000	BL	BRANCH IF LESS	1
*	33000000	BLE	BRANCH IF LESS OR EQUAL	1
*	02100000	BN	BRANCH IF NEGATIVE	1
*	03500000	BNE	BRANCH NOT EQUAL	1
*	02500000	BNEZ	BRANCH IF NOT EQUAL TO ZERO	1
*	02300000	BNZ	BRANCH IF NEGATIVE OR ZERO	1
*	02040000	BO	BRANCH IF ODD	1
*	03000000	BOI	BRANCH ON INDICATOR	1
*	04000000	BOS	BRANCH ON STATE	1
*	04440000	BOV	BRANCH ON OVERFLOW	1
*	02400000	BP	BRANCH IF POSITIVE	1
*	02600000	BPZ	BRANCH IF POSITIVE OR ZERO	1
*	01000000	BRU	BRANCH UNCONDITIONAL	1

B.1 OPCODES SORTED BY ASCENDING ALPHA OPCODES (Continued)

	Opcode	Mnemonic	Code Description	Cycles
*	12000000	BSM	BRANCH STORAGE RETURN AT M	2
		BSS	BLOCK STORAGE SIZE	
*	02200000	BZ	BRANCH IF ZERO	1
	12000000	CALL	SUBROUTINE CALL	
*	23000000	CAM	COMPARE A WITH MEMORY	2
	07000604	CLA	CLEAR ACCUMULATOR	1
*	30000000	DADD	DOUBLE ADD	3
*		DATA	DATA DEFINITION	
*	35000000	DIV	DIVIDE	26
	35000000	DLD	DOUBLE LOAD	3
	07034000	DSA	DOUBLE SHIFT AROUND	3
	07036000	DSL	DOUBLE SHIFT LEFT	
	07016000	DSN	DOUBLE SHIFT NORMALIZED	
	07030000	DSR	DOUBLE SHIFT RIGHT	
*	33000000	DST	DOUBLE STORE	3
*	32000000	DSUB	DOUBLE SUBTRACT	3
	07014000	DTC	DOUBLE TWO'S COMPLEMENT	2
		END	PROGRAM TERMINATOR	
*	21000000	EOR	EXCLUSIVE OR	2
		EQU	EQUIVALENCE	
	060100XX	ETST	ERROR TEST	1
	07010000	EXC	EXCHANGE A AND E	1
	06051500	FSKIPB	SKIP FILE FORWARD (ADVANCE TO TAPE MARK)	1
	06041500	FSKIPF	SKIP FILE BACKWARE (ADVANCE TO TAPE MARK)	1
	07012400	IDA	INTERRUPT DISABLE	1
	07004400	IEN	INTERRUPT ENABLE	1
	07000600	LAR	LOAD A FROM RELOCATION REGISTER	1
	07000200	LAX	LOAD A FROM INDEX	1
*	24000000	LDA	LOAD A-REGISTER	2
*	25000000	LDE	LOAD E-REGISTER	2
	07032000	LDS	LOGICAL DOUBLE SHIFT	
*	05000000	LDX	LOAD INDEX	1
		LIST	PRODUCE ASSEMBLY LISTING	
	07000400	LRA	LOAD RELOCATION REGISTER FROM A	1
	07022000	LS	LOGICAL SHIFT A	
	07000000	LXA	LOAD INDEX FROM A	1
*	34000000	MUL	MULTIPLY	25
		NOLIST	NO ASSEMBLY LISTING	
	10000000	NOP	NO OPERATION	1
		OBJ	PRODUCE OBJECT PROGRAM	
*	27000000	OR	OR (INCLUSIVE)	2
		ORG	ORIGINATION CONTROL	

	Opcode	Mnemonic	Code Description	Cycles
		PAGE	PAGINATION CONTROL	
	060010XX	PCOMP	PRIORITY COMPLETE	1
	060110XX	POFF	PRIORITY OFF (INTERRUPT DISABLE)	1
	060130XX	PON	PRIORITY ON (INTERRUPT ENABLE)	1
	00000000	PROC	SUBROUTINE ENTRY POINT	
	00000000	PZE	POSITIVE ZERO (ENTRY PT)	
	064014XX	RD	READ	1
	066114XX	RDS	READ STATUS	1
	06501500	RDT	READ (MAGNETIC) TAPE	1
	06601400	RDTT	READ TELETYPE	1
	06611700	REWC	READ EXCESS WORD COUNT	1
	06000500	REWIND	REWIND TAPE	1
	06011500	RSKIPB	SKIP RECORD BACKWARD	1
	06001500	RSKIPF	SKIP RECORD FORWARD	1
	07006000	RSR	READ SWITCH REGISTER	1
	07012000	RST	RESET STATE	1
*	17000000	RUM	REPLACE UNDER MASK	2
	07024000	SA	SHIFT A AROUND LEFT	
	06461500	SKWR	SKIP AND WRITE	1
	07026000	SL	SHIFT A LEFT	
*	37000000	SOM	SUBTRACT ONE FROM MEMORY	4
	06420400	SPAC	SPACE	1
	06000000	SPU	SELECT PERIPHERAL UNIT (DETAILED SPU COMMANDS ARE LISTED IN APPENDIX D)	1
	07020000	SR	SHIFT A RIGHT	
	07004000	SST	SET STATE	1
*	14000000	STA	STORE-A-REGISTER	2
*	15000000	STE	STORE-E-REGISTER	2
	07000611	STM1	SET FST-1 MODE	1
	07000612	STM2	SET FST-2 MODE	1
	060000XX	STST	STATUS TEST	1
*	16000000	STX	STORE INDEX	2
*	22000000	SUB	SUBTRACT	2
	07002000	TCA	TWO's COMPLEMENT A	1
	06000400	TOF	TOP-OF-FORM	1
	064214XX	WRIT	WRITE	1
	06061500	WRITM	WRITE TAPE MARK	1

* B23=0 for Absolute Memory Reference of Non-REL Program
=1 for Relocatable Memory Reference with REL Program

B.2 OPCODES SORTED BY ASCENDING OCTAL OPCODES

	Opcode	Mnemonic	Code Description	Cycles
		ABS	ABSOLUTE PROGRAM LOCATOR	
	00000000	BSS	BLOCK STORAGE SIZE	
*	00000000	DATA	DATA DEFINITION	
	00000000	END	PROGRAM TERMINATOR	
		EQU	EQUIVALENCE	
		LIST	PRODUCE ASSEMBLY LISTING	
		NOLIST	NO ASSEMBLY LISTING	
		OBJ	SPECIFY OBJECT PROGRAM SIZE	
		ORG	ORIGINATION CONTROL	
		PAGE	PAGINATION CONTROL	
*	00000000	BAH	BRANCH AFTER HALT	1
	00000000	PROC	SUBROUTINE ENTRY POINT	
	00000000	PZE	POSITIVE ZERO (ENTRY PT)	
*	01000000	BRU	BRANCH UNCONDITIONAL	1
*	02000000	BAT	BRANCH ON A-REGISTER TEST	1
*	02040000	BO	BRANCH IF ODD	1
*	02100000	BN	BRANCH IF NEGATIVE	1
*	02200000	BZ	BRANCH IF ZERO	1
*	02300000	BNZ	BRANCH IF NEGATIVE OR ZERO	1
*	02400000	BP	BRANCH IF POSITIVE	1
*	02500000	BNEZ	BRANCH IF NOT EQUAL TO ZERO	1
*	02600000	BPZ	BRANCH IF POSITIVE OR ZERO	1
*	03000000	BOI	BRANCH ON INDICATOR	1
*	03040000	BBC	BRANCH BIT COMPARE	1
*	03100000	BL	BRANCH IF LESS	1
*	03200000	BE	BRANCH IF EQUAL	1
*	03300000	BLE	BRANCH IF LESS OR EQUAL	1
*	03400000	BG	BRANCH IF GREATER	1
*	04000000	BOS	BRANCH ON STATE	1
*	04440000	BOV	BRANCH ON OVERFLOW	1
	06000000	SPU	SELECT PERIPHERAL UNIT (DETAILED SPU COMMANDS ARE LISTED IN APPENDIX D)	
	060000XX	STST	STATUS TEST	1
	06000400	TOF	TOP-OF-FORM	1
	06000500	REWIND	REWIND TAPE	1

B.2 OPCODES SORTED BY ASCENDING OCTAL OPCODES (Continued)

Opcode	Mnemonic	Code Description	Cycles
060010XX	PCOMP	PRIORITY COMPLETE	1
06001500	RSKIPF	SKIP RECORD FORWARD	1
060100XX	ETST	ERROR TEST	1
060110XX	POFF	PRIORITY OFF (INTERRUPT DISABLE)	1
06011500	RSKIPB	SKIP RECORD BACKWARD	1
060130XX	PON	PRIORITY ON (INTERRUPT ENABLE)	1
06020400	FEED	CHARACTER (PAPER TAPE) FEED	1
06041500	FSKIPF	SKIP FILE FORWARD (ADVANCE TO TAPE MARK	1
06051500	FSKIPB	SKIP FILE BACKWARD (GO BACK TO TAPE MARK)	1
06061500	WRITM	WRITE TAPE MARK	1
064014XX	RD	READ	1
06401500	ART	ALTERNATE READ RECORD TAPE	1
064034XX	ARD	ALTERNATE READ	1
06420400	SPAC	SPACE	1
064214XX	WRIT	WRITE	1
06422400	ASPAC	ALTERNATE SPACE	1
06423400	AWRIT	ALTERNATE WRITE	1
06461500	SKWR	SKIP AND WRITE	1
06501500	RDT	READ (MAGNETIC) TAPE	1
06601400	RDTT	READ TELETYPE	1
066114XX	RDS	READ STATUS	1
06613400	ARDS	ALTERNATE READ STATUS	1
07000000	AUG	AUGMENT	
07000000	LXA	LOAD INDEX FROM A	1
07000200	LAX	LOAD A FROM INDEX	1
07000400	LRA	LOAD RELOCATION REGISTER FROM A	1
07000600	LAR	LOAD A FROM RELOCATION REGISTER	1
07000604	CLA	CLEAR ACCUMULATOR	1
07000611	STM1	SET FST-1 MODE	1
07000612	STM2	SET FST-2 MODE	1
07002000	TCA	TWO'S COMPLEMENT A	1
07004000	SST	SET STATE	1
07004400	IEN	INTERRUPT ENABLE	1
07006000	RSR	READ SWITCH REGISTER	1
07010000	EXC	EXCHANGE A AND E	1

B.2 OPCODES SORTED BY ASCENDING OCTAL OPCODES (Continued)

	Opcode	Mnemonic	Code Description	Cycles
	07012000	RST	RESET STATE	1
	07012400	IDA	INTERRUPT DISABLE	1
	07014000	DTC	DOUBLE SHIFT NORMALIZED	
	07020000	SR	SHIFT A RIGHT	
	07022000	LS	LOGICAL SHIFT A	
	07024000	SA	SHIFT A AROUND LEFT	
	07026000	SL	SHIFT A LEFT	
	07030000	DSR	DOUBLE SHIFT RIGHT	
	07032000	LDS	LOGICAL DOUBLE SHIFT	
	07034000	DSA	DOUBLE SHIFT AROUND	
	07036000	DSL	DOUBLE SHIFT LEFT	
*	10000000	NOP	NO OPERATION	1
*	11000000	ATX	ADD TO INDEX	2
*	12000000	BSM	BRANCH STORE RETURN AT M	2
	12000000	CALL	SUBROUTINE CALL	
*	14000000	STA	STORE A-REGISTER	2
*	15000000	STE	STORE E-REGISTER	2
*	16000000	STX	STORE INDEX	2
*	17000000	RUM	REPLACE UNDER MASK	2
*	20000000	ADD	ADD	2
*	21000000	EOR	EXCLUSIVE OR	2
*	22000000	SUB	SUBTRACT	2
*	23000000	CAM	COMPARE A WITH MEMORY	2
*	24000000	LDA	LOAD A-REGISTER	2
*	25000000	LDE	LOAD E-REGISTER	2
*	26000000	AND	LOGICAL AND	2
*	27000000	OR	OR (INCLUSIVE)	2
*	30000000	DADD	DOUBLE ADD	3
*	31000000	DLD	DOUBLE LOAD	3
*	32000000	DSUB	DOUBLE SUBTRACT	3
*	33000000	DST	DOUBLE STORE	3
*	34000000	MUL	MULTIPLY	25
*	35000000	DIV	DIVIDE	26
*	36000000	AOM	ADD ONE TO MEMORY	4
*	37000000	SOM	SUBTRACT ONE FROM MEMORY	4

* Bit 23 = 0 for Absolute Reference of non-REL Program
= 1 for Relocation Memory Reference with REL Directive

C

Conversion of TOPSY/DOPSY Assembly Language Programs to MASTR

C.1 INTRODUCTION

To run in MASTR, assembly language overlays must be modified to include a program header, to use the new global variables, and to follow the calling sequences as defined for IOCS and system subroutines.

The same overlay may be called from the FACTOR program by the EXEC statement and from a command from the background. There is no need to maintain two copies of the program as in TOPSY.

MASTR can load and execute an assembly language overlay at any location in memory. It is no longer necessary to create the coreimage files at specific locations to avoid overlap with other coreimage files.

Note

The object files are no longer supplied on the system DBUP tape.

C.2 PROCEDURE

This section defines a step by step procedure for the conversion of DOPSY/TOPSY assembly language programs to MASTR. The difficulty of the conversion depends to a great deal on the complexity of the program and the familiarity of the programmer doing the conversion with that program. If the program is well structured and documented, the conversion should be straight forward.

1. In the job 'HELP', there is a source file named '*AHDR'. Add this header to the program by inserting the existing program in the appropriate place for a foreground or background program. Note that a foreground entry is made when called from a FACTOR program by means of the EXEC statement. The background entry is made when called by a command.

2. Examine all EQU statements to find references to system globals. If any exist, then their usage must be examined and replaced with the equivalent in MASTR. Note that there is not always a direct replacement for a DOPSY/ TOPSY variable. System globals in MASTR will be located in SYSVAR (system variables), GLOVAR (global variables), TVT (test head variables), and SVT (current station variables).

3. Examines all EQU statements to fine uses of system routines that are called by a BSM* instruction. These calls must be replaced with the equivalent call to MASTR by the SYXVEC table. Note that some calling sequences are different, i.e., the floating point routines use the E-register instead of a parameter placed after the call.

4. Examine all CALL statements to find references to external routines.

   a. The routine that is called may now be a part of the MASTR routine library, so examine the routines in SYXVEC for an equivalent function.

   b. The routine that is called may be another part of the program that also must be converted. A linking CREATE program is available on MASTR Rel 2.0. When linking object files together to make a MASTR overlay, put the tag LASTW on the END instruction of the last module only and link it to the first module with an ENT and EXT instruction. Also change the fifth word to be "DATA LASTW".

5. Examine the program for any PROC directives with an interrupt location specified. Also examine the program for any SPU instruction to I/O devices. All routines that perform direct input/output to a peripheral must be changed to use IOCS. An overlay may read and write directly to the test registers providing the usage does not conflict with the test head driver.

6. Add an initialization section that:

   a. clears any flags that must be 0 upon entry - remember that the programs may not be reloaded from the disk each time. The routine MPZERO may be useful for this function.

   b. opens files for I/O and saves the pointers to IOATAB (X6), for use by subsequent calls to IOCS.

7. Define data control blocks (DCBs) in MASTR format for each I/O operation and change all calls to use IOCS.

8. Upon exit from the program, all files that are open must be closed (even upon exit due to an error). This is required so that the system IOATAB is not filled with files that are no longer used. The IOATAB pointer serves as a useful flag if coded as follows:

```
EXIT      -
          LDA     IOATP          Get File ID
          BZ      CLOSED         File Already Closed
          LXA     X6             Set Up ID for IOCS
          CLA
          STA     IOATP          Mark File Closed
          LDX     X1,CLOSE       Get Control Word
          BSM*    IOCS           Close the File
          BRU     ERROR          Error Return
CLOSED    -
```

9. Change program exit to return through the foreground (FGEP) or background (BGEP) as appropriate.

10. Check for conflicting uses of index registers and state switches with the new system routines, i.e., IOCS uses X1 and X6.

Index

Feedback Form

Please send us your comments. They will help us produce better publications for you. We are interested in comments on accuracy, completeness, organization, clarity, audience suitability, illustrations, and detail level. If you choose to identify yourself (lower left), we will acknowledge your comments. All comments are confidential, FTSG property.

Optional

Name _____

Title _____

Address _____

Phone _____

Telex _____

Publications Department Use Only

P/N 57518702

Rev _____ Date 8/79

Technical Reviewer _____ Date _____ Response Sent _____ Date _____

FAIRCHILD Test Systems Group

634700-1

STAPLE

STAPLE

No Postage
Necessary If
Mailed In The
United States

Business Reply Mail
First Class Permit No. 5699 San Jose, California

Postage Will Be Paid By

Fairchild Test Systems Group
Technical Publications
1725 Technology Drive
San Jose, Ca. 95110