Tektronix®
COMMITTED TO EXCELLENCE

S-3200 Series
Automated Test Systems

TEKTEST III, Version 4 Overview and UPDATE for the System Programmer's Reference Guide

Tektronix, Inc.
P.O. Box 500
Beaverton, Oregon 97077

PLEASE CHECK FOR CHANGE INFORMATION
AT THE REAR OF THIS MANUAL.

061-2592-00 SEMICONDUCTOR TEST SYSTEMS FIRST PRINTING—JULY 1981

SOFTWARE LICENSE

Software supplied by Tektronix, Inc., as a component of a system or as a separate item is furnished under a license for use on a single system and can be copied (with the inclusion of copyright notice) only for use on that single system.

CONTENTS

TEKTEST III, Version 4.0

INTRODUCTION

TEKTEST III Version 4 was developed to significantly enhance the existing TEKTEST III Version 3. To do this, Version 4

- Provides space for future system growth and development.
- Increases size and speed for running test programs while decreasing system dependency.
- Eliminates some existing extended core problems.
- Maintains compatibility with the existing user interface.

VERSION 4 ENHANCEMENT HIGHLIGHTS

Larger Test Programs

The Version 4 operating system allows larger test programs than Version 3. Programs run from the TSCU (Test Station Control Unit) can now be a full 20K, an increase of at least 5K. Also, with Version 4, the 20K partitions are not reduced by **lun** assignments or mass storage devices.

Faster Speed

At bootup, Version 4 configures the system in the most optimal fashion for the 11/34 or 11/35 floating-point hardware. On an 11/34 with Version 4, floating-point intensive programs have run two times faster than an 11/34 with Version 3. This is because Version 3 on an 11/34 uses software to emulate the 11/35 floating-point instructions while Version 4's 11/34 uses its own floating-point processor. An 11/34 with Version 4 and floating-point hardware will also execute floating-point intensive programs 18% faster than Version 3's 11/35 and floating-point hardware. Thus, with Version 4, users will finally be able to see the increased capability of the 11/34 over the 11/35.

Enhanced TCM and REDUCE

The net maximum size of a program in TCM is 7K-8K. This is a 40% increase over the 4K-5K available with Version 3, where a 12K maximum is reduced by mass storage devices and **lun** assignments.

The maximum size program run through REDUCE in background Ø can now be 22K, up from the 13K typically available in Version 3. The additional Version 4 background partitions can be a maximum 24K, up from 20K in Version 3. (These partition sizes are affected by buffer requirements for some types of **luns**.)

Version 4 has a default of four different mass storage **lun** assignments available to the user running in foreground. If the user needs more than the default, there is a new command in LOG that changes the amount of memory set aside for **lun** assignment overhead so that more **luns** can be assigned.

Extended Core

Version 4 does not physically fragment programs in memory. This alleviates problems created in Version 3 when DMAs are attempted across fragmented memory, and allows all TEKTEST subroutines to work in extended core.

SYSTEM INSTALLATION REQUIREMENTS

Additional Hardware

To run Version 4, the user's CPU must contain 42K of memory and have memory management hardware. An additional requirement is floating-point hardware for both the 11/34 and 11/35.

Files Retranslation

Special attention has been paid to maintaining compatibility with this system and the existing customer software base. However, all **.EDT** files will require translation. This is a necessary and unavoidable result of the previously mentioned space increases. Also, all programs using **PRMSUB.FNC, R2942.FNC** or **WAVE.FNC** must be retranslated using the new releases of these **.FNC** files. Current releases of all other **.FNC** files are permitted in other cases.

VERSION 4 SYSTEM MODIFICATIONS IN BRIEF

System Memory Allocation Modifications

1. Additional error conditions may be encountered during bootup:

TEKTST III VERS 4. 0X REQUIRES MEM MNGT.	Check for memory management on system.
FAILS MIN MEMORY REQUIREMENT. WILL NOW RETRY WITH MIN SYSSIZ.	Check for minimum memory.

2. TEKBUG outputs system size for booted system:

   SYSTEM SIZE = xxK.

3. If there is insufficient memory allocated to build the monitor:

NO MORE HIGH CORE MEMORY LEFT INCREASE SYSTEM SIZE. DISK BLK3 LOC 50 REBOOT	The system size must be increased to allow more space.

Additions to TMON

Routine to set the TESTID and test station. User prompt:

TESTID AND STATION:

reply format:

AAA, X.

AAA = TESTID. X = station (when an '*' is used for the station, all stations are set). The IDENT is also set. Error messages:

ILLEGAL STATION NUMBER
INVALID CHARACTER IN TESTID.

Additions to LOG

1. LPO command is used to set the line printer width and number of lines per page. The following are the prompts:

   ENTER PRINT LINE WIDTH:
   ENTER PRINT LINES PER PAGE:

   Error messages:

   ENTER NUMBERS ONLY
   OUT OF RANGE (9<X<133).

   The current values are printed out as follows:

   PRINT LINES PER PAGE ARE nnn.
   PRINT LINE WIDTH IS nnn.

2. SIZE command used before a system reboot allows the user to change the monitor size. The prompt is:

   ENTER SYSTEM SIZE.

   Error messages:

   OUT OF RANGE (17.<X<27.)
   ENTER NUMBERS ONLY.

   The current values are printed out as follows:

   SYSTEM SIZE IS nnK
   NOW REBOOT THE SYSTEM.

3. MEMORY command gives the following information:

   MEMORY USED nnnnn. WORDS
   MEMORY FREE nnnnn. WORDS. } in REDUCE

MEMORY USED FOR ASSIGNING LUNS nnnn. WORDS
MEMORY FREE FOR NEW ASSIGNMENTS nnnn. WORDS } in LOG

4. MAP command gives two additional messages:

   STATNS OVER BG0 MUST BE AT LEAST 20K

   and

   +BG0 SIZE = xxK.

Additional and Modified EMTs

WRDMOV (EMT 200). Used to send a word from user space to kernel space.

WRDUSR (EMT 201). Moves a word from kernel space to user's R1.

FGCLIM (EMT 55 modified). Foreground execution only. Used by the system to relocate portions of the PRINT and FORMATTER programs to high core at bootup. Also used to obtain space in the kernel for user-written MONEXs.

FGLMEM (EMT 202). Foreground low address limit. Foreground execution only.

FLPCON (EMT 203). Allows connection of a user-written floating-point error handler to the monitor.

SYSBUF (EMT 206). Coordinates access to the low core user's buffer to facilitate inter-partition data transfers.

MTW18 (EMT 212). Allows specification of an 18-bit address for writing to magtape.

MTEW18 (EMT 213). Same as MTW18 except that it writes with an extended inter-record gap.

MTR18 (EMT 214). Allows specification of an 18-bit address for reading from magtape.

SYSTEM PROGRAMMER'S REFERENCE GUIDE UPDATE

INSTRUCTIONS

The update pages following this instruction sheet are intended to replace pages in the existing *System Programmer's Reference Guide* and to provide reference for TEKTEST III, Version 4.

The table below itemizes additions, deletions and replacements, as required.

Updates to the System Programmer's Reference Guide (070-3551-00)

Add:

Following old page 2-28, add 2-29, 2-30

(Version 3) Remove Page(s):	**(Version 4) Replace with Page(s):**
i through viii	i through viii
3-1 through 3-4	3-1 through 3-4
3-55, 3-56	3-55, 3-56
3-123	3-123 through 3-129
4-1 through 4-8	4-1 through 4-4, 4-4.1, 4-4.2, 4-5 through 4-8
6-1 through 6-6	6-1 through 6-6
7-1, 7-2	7-1, 7-2, 7-2.1
7-5, 7-6	7-5, 7-6
7-17, 7-18	7-17, 7-18
8-1 through 8-4	8-1 through 8-4
8-7, 8-8	8-7, 8-8
9-1, 9-2	9-1, 9-2
Section 10 tab divider and entire Section 10	New Section 10 tab divider, 10-1

Fielding Interrupts from User Space

It is not possible under Version 4 to have test programs field interrupts directly as in earlier versions. Since test programs running in foreground now run in user space, they must field interrupts from kernel space. The WAVFORM package, or user-defined hardware with interrupt capabilities must follow a prescribed protocol in order to service their own interrupts. The overhead is two instructions before and two instructions after the interrupt service routine. The basic mechanism for handling these types of interrupts consists of:

1. The interrupt service routine (ISR) address must be given to the monitor.

2. The interrupt must vector to WAVINT, a routine within the monitor, which will pass control to the ISR.

3. The test program resume address must be made known to the monitor.

4. The interrupt is enabled. (Option 1: the CPU can be returned to BG during interrupt processing.)

5. Interrupt handling begins. The ISR shall jump to USRSTP, a routine within the monitor, after each interrupt.

6. Interrupt handling finishes. The last interrupt shall cause the interrupt to be disabled from the ISR at level 7. (If option 1 was used, after all interrupts are finished, FG must be restarted from the ISR.)

7. The ISR shall jump to a routine within the monitor, which uses the test program resume address to start the test program running.

The specific procedure for the above steps is defined below. The code should be written as a standard TEKTST.FNC file. See the *System Programmer's Reference Guide*. The ISR should run at level 7.

[1] Set up for ISR.

```
MOV     PC,R1           ;GET PIC ADDR OF ISR
ADD     #ISR-.,R1       ;...
MOV     R1,R3           ;SAVE IT
MOV     R1,R0           ;FIND APR WHICH MAPS ISR
ASH     #-12.,R0        ;...
BIC     #177761,R0      ;...
ADD     #PAR0,R0        ;GET ADDR OF PAR IN R0
                        ;PAR0=177640
BIC     #160000,R1      ;GET PAGE ADDR FIELD
ASH     #-6,R1          ;...
ADD     (R0),R1         ;PAGE ADDR FOR KRNL APR 1
MOV     R1,@#WAVBAS     ;TELL MNTR PAGE ADDR FOR
                        ;KRNL APR 1 (WAVBAS=1070)
BIC     #177700,R3      ;OFFSET FROM BEGINNING OF
                        ;APR TO ISR
ADD     #20000,R3       ;ADDR OR ISR IN APR 1
MOV     R3,@#WAVOFS     ;TELL MNTR OF ISR ADDR
                        ;WAVOFS=1066
```

[2]

```
MOV    @#WAVSRV,@#HRDWAR        ;SETUP INTERRUPT VECTOR
                                ;WAVSRV=1064
                                ;HRDWAR=USER DEFINED HRDWR
                                ;VECTOR ADDRESS.
```

[3]

```
MOV    PC,R0                    ;MAKE PIC
ADD    #RESUME-.,R0             ;
MOV    R0,@#USWSPA              ;SAVE TEST PROGRAM RESUME
                                ;ADDR. USWSPA=1072.
```

[4] Enable the interrupt (if option 1, the following becomes step 4).

```
BIS    #340,@#PSW               ;LEVEL 7
       ENABLE THE INTERRUPT
       MOV    @#IPSTOP,PC       ;GIVE BG THE CPU. IPSTOP=342
```

(End of option 1 addition.)

[5] Interrupt handling begins.

```
ISR:    .                       ;START OF INTERRUPT HANDLER
        .                       ;RUNS ON KERNEL APR 1
        .                       ;AT LEVEL 7
        .
       MOV    @#USTOPS,PC      ;RETURN FROM INTERRUPT
                               ;USTOPS=1074.
```

[6] Interrupt handling finishes. Upon receipt of the last interrupt, the user shall disable the interrupt (if option 1, the following becomes step 6).

```
MOV    #030340, @#IPINT         ;IPSTRT PRIORITY RETURN LEVEL
                                ;IPINT=1062
MOV    @#IPSTRT,-(SP)           ;RESTART FG (IPSTRT=340)
JSR    R5,@(SP)+                ;...
MOV    #140000,@#IPINT          ;LEVEL 0 FOR OTHER USERS
                                ;THIS WORD TO PSW ON RETURN
                                ;FROM IPSTRT.
```

(End of option 1 addition)

[7]

```
MOV    @#WAVSPC,PC              ;RESUME THE TEST PROGRAM
                                ;WAVSPC=1076
```

INTRODUCTION

This manual describes the data structures and programming conventions used by the TEKTEST III, Version 4 operating system and how user-written programs, subroutines and functions can be added to the system. A knowledge of PDP-11 assembly language and programming concepts is assumed.

CONTENTS

kc

SECTION 4: FORMATTED I/O (IOTs)

SECTION 5: FOREGROUND/BACKGROUND COMMUNICATION (TRAPs)

SECTION 6: SYSTEM SUBROUTINES

SECTION 7: FILE AND DATA STRUCTURES

SECTION 8: MAINTENANCE AND DEBUGGING AIDS

EMT SUMMARY

EMT Number	Name	Function
0	**INPUT**	Keyboard input (ASCII) (line level)
1	**OUTPUT**	Terminal screen output (ASCII) (line level)
2	**PRLINE**	Paper tape input (ASCII) (line level)
3	**PPLINE**	Paper tape output (line level)
4	**LPLINE**	Line printer output (ASCII) (line level)
5	**CRLINE**	Card reader input (IBM029/ASCII) (line level)
6	**CRBIN**	Card reader input (Binary packed) (line level)
7	**CRPACK**	Card reader input (Packed Card code) (line level)
10	**CRIMAG**	Card reader input (Binary image) (line level)
11	**CHARI**	Terminal character input
12	**CHARO**	Terminal character output
13	**PRCHAR**	Paper tape input
14	**PPCHAR**	Paper tape output
15	**READ**	Disk input
16	**WRITE**	Disk output
17	**MTREAD**	Magnetic tape input
20	**MTWRITE**	Magnetic tape output
21	**MTWIRG**	MT output (Ext IRG)
22	**WRTCHK**	Disk write check
23	**ALOCPR**	Paper tape reader allocation
24	**ALOCPP**	Paper tape punch allocation
25	**ALOCLP**	Line printer allocation
26	**ALOCCR**	Card reader allocation
27	**ALOCDK**	Disk drive n allocation
30	**ALOCMT**	Magnetic tape allocation
31	**PALODK**	Permanent DK n allocation
32	**RLSPR**	Paper tape reader release
33	**RLSPP**	Paper tape punch release
34	**RLSLP**	Line printer release
35	**RLSCR**	Card reader release
36	**RLSDK**	Disk drive n
37	**RLSMT**	Magnetic tape release
40	**PRLSDK**	Permanent DK n release

EMT Number	Name	Function
41	**GETERR**	Return abort status
42	**FILE**	Disk file management
43	**CLRKB**	Clear keyboard input queue
44	**CLRTTY**	Clear terminal screen output queue
45	**MTSPEC**	Magnetic tape special functions
46	**TSTKB**	Test keyboard input queue
47	**GETSR**	Read switch register
50	**GETPAR**	Get terminal PSA number
51	**GETUID**	Get user's ident
52	**SETUID**	Set user's ident
53	**GETDATE**	Get date/time
54	**SETDATE**	Set date/time
55	**FGCLIM**	Move MONEX to kernel
56	**HIMEM**	Background high address limit
57	**SWUSER**	Switch user
60	**EXIT**	Exit to monitor
61	**ABOADDR**	CTRL/C return address
62	**RETURN**	Return exit
63	**CTRLCH**	Manual interrupt flag
64	**RREAD**	Disk input
65	**RWRITE**	Disk output
66	**RFILE**	Disk file management
67	**RUNOVR**	Disk input and transfer control
70	**GTSTID**	Get test ID
71	**STSTID**	Set test ID
72	**SFGFLG**	Set foreground control flag
73	**STRTBG**	Start background
74	**STRTFG**	Start foreground
75	**SUSPND**	Suspend user
76		Reserved for Tektronix, Inc.
77	**RUNFG**	Load foreground program
100	**RLDTST**	Reload test program
101	**EMTRTN**	EXIT – no condition codes, leave ΛC
102	**LPCHAR**	LP character output
103	**SCLOSA**	Set close for all users
104	**CCLOSA**	Clear close for all users

EMT Number	Name	Function
105	**ASGNKB**	Assign terminal to FG test station
106	**RLSKB**	Release terminal from FG
107	**ATTACH**	Attach assigned terminal
110	**DETACH**	Detach assigned terminal
111	**GRABKB**	Unconditionally attach terminal
112	**CONSOLE**	Assign terminal to console
113	**RELEASE**	Release console status
114	**CONNUM**	Return terminal number of console
115	**GKBNUM**	Return terminal number
116	**GETSTS**	Get STSBLK address
117	**SETSTS**	Set STSBLK address
120	**RINPUT**	Reentrant line KB in
121	**ROUTPUT**	Reentrant line TTY out
122	**RPRLINE**	Reentrant line PR
123	**RPPLINE**	Reentrant line PP
124	**RLPLINE**	Reentrant line LP
125	**RCRLINE**	Reentrant line CR (029/ASCII)
126	**RCRBIN**	Reentrant line CR (Binary)
127	**RMTREAD**	Reentrant MT read
130	**RMTWRITE**	Reentrant MT write
131	**RMTWIRG**	Reentrant MT write with EIRG
132	**EMTRT1**	Exit and pass condition codes, leave AC
133	**GETDK**	Get user's drive number
134	**SETDK**	Set user's drive number
135	**DKBLK**	Get address of user's drive block
136	**GFGDK**	Get FG drive number
137	**SFGDK**	Set FG drive number
140	**FGDKBLK**	Get address of FG drive block
141	**RDKFILE**	Reentrant file management any drive
142	**DKFILE**	File management any drive
143	**RDKSFILE**	Reentrant file management system drive
144	**DKSFILE**	File management system drive
145	**OFFLINE**	Set drive offline
146	**ONLINE**	Set up drive
147	**DKSTAT**	Check RK05 hardware status
150	**DKSNUM**	Get system drive number

EMT Number	Name	Function
151	**DKSBLK**	Get address of system drive block
152	**DKBITS**	Return the disk allocation bits
153	**DKRUNFG**	Run foreground any drive
154	**DKSRUNFG**	Run foreground system drive
155	**DKFMT**	Write format
156	**RDKFMT**	PIC write format
157	**DKZERO**	Write format and zero
160	**RWRTCHK**	Disk write check
161	**FREE**	Return largest free block
162	**TRANID**	Set FG default ident and drive
163	**KBSTUFF**	Stuff character in KB input queue
164	**TTSTUFF**	Stuff character in KB output queue
165	**READ18**	Disk read to 18 bit address
166		Reserved for Tektronix, Inc.
167	**LOADMTL**	Load MTL file for IP3455
170		Reserved for users
171		Reserved for users
172		Reserved for users
173		Reserved for users
174		Reserved for users
175		Reserved for users
176		Reserved for users
177		Reserved for users
200	**WRDMOV**	Move word to kernel space
201	**WRDUSR**	Get word from kernel space
202	**FGLMEM**	Foreground test low address limit
203	**FLPCON**	Connect floating point error handler
204		Reserved for Tektronix, Inc.
205		Reserved for Tektronix, Inc.
206	**SYSBUF**	System Buffer Communication Control
207-377		Reserved for Tektronix, Inc.

EMT 55 ; FGCLIM ; MOVE MONEX TO KERNEL

This instruction must be executed only in the foreground.

This EMT is used by the system to relocate portions of the PRINT and FORMATTER programs to high core at bootup. It is also used to obtain space in the kernel for user-written MONEXs. When a MONEX is being run, a request to the monitor must be made for space.

CAUTION

Any violation of this rule may result in the MONEX writing over portions of the monitor.

This instruction does not modify the general registers. It conditionally clears and sets the condition code bits in the processor status word.

Calling sequence:

– R0 contains the address of the lowest requested memory space –

```
EMT    55
```

– return here –

Return conditions:

Condition	Code N	Z	V	C
No error	0	0	0	0
Illegal call from Background	0	0	0	1
Address overlaps monitor	0	0	1	1

EMT 56 ; HIMEM ; BACKGROUND HIGH ADDRESS LIMIT

You use this instruction in the background to request additional or release excess memory space for data storage while the program is running. R0 contains the address of the highest requested memory space.

If this request causes an overlap of memory allocation with a foreground test program, the background program is suspended from operation until the foreground releases memory space sufficient to meet this EMT's request. } BGØ Only

Ordinarily, the general registers are not modified by this instruction. The condition code bits in the processor status word are conditionally cleared and set as shown below.

	Code			
Condition	N	Z	V	C
No errors	0	0	0	0
Address out of bounds	0	0	0	1

Requesting more memory space than is available on the system causes the address-out-of-bounds error condition. When this error occurs, the maximum memory address is returned in general register R0.

Calling sequence:

```
— R0 contains the requested address —

        EMT     56

— return here —
```

EMT 200 ; WRDMOV ; MOVE WORD TO KERNEL SPACE

The WRDMOV EMT is used to send a word from user space to kernel space. EMT 200 duplicates the function of TRAP 021 and can be used generally.

Calling sequence:

– R0 contains the destination address in kernel (monitor) space –

– R1 contains the data to be sent –

```
EMT    200
```

There are no errors for EMT 200.

EMT 201 ; WRDUSR ; GET WORD FROM KERNEL SPACE

The WRDUSR EMT gets a word from kernel space and places it in the user's R1. This instruction duplicates the function of TRAP 022 and can be used generally.

Calling sequence:

– R0 contains the source address in kernel (monitor) space –

```
EMT    201
```

– R1 contains the data from the source address –

There are no errors for EMT 201.

EMT 202 ; FGLMEM ; FOREGROUND TEST LOW ADDRESS LIMIT

Foreground low address limit, FGLMEM, must be executed only in the foreground. It is used by the system and not recommended for general use. Some system programs require use of BG0's memory partition (TCM, IP, LOG). Before this area can be used, the foreground program must issue this instruction to take over BG0's memory. R0 contains the address of the lowest requested memory space. If the request for space causes an overlap with BG0 memory area (see EMT HIMEM, SYSTEM REF), the BG0 program is suspended and swapped out to the disk swap buffer. When the foreground program is done using BG0's memory, it again issues the instruction, this time with R0 containing a high virtual address. The BG0 program is then read back into memory and its suspension removed.

Calling sequence:

```
— R0 contains the requested address —

        EMT     202
```

Return conditions:

Condition	Code N	Z	V	C
No error	0	0	0	0
Illegal call from Background	0	0	0	1
Address out of Bounds	0	0	1	1

EMT 203 ; FLPCON ; CONNECT FLOATING POINT ERROR HANDLER

This EMT instruction allows the user to connect a floating point error handler to the monitor.

Calling sequence:

```
— R0 contains the address of the user-written FP error handler —

        EMT     203
```

The monitor floating point interrupt handler passes the floating point error on to a user handler by putting the error register on the user stack as follows:

```
— R16 is the user stack pointer —
    R16+4 = old PC
    R16+2 = FEC register contents
```

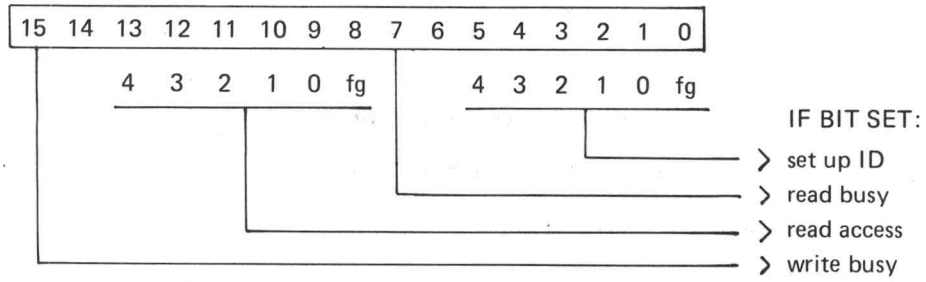
All registers are the same prior to interrupt. All condition code bits will set if not connected and an interrupt occurs. The PSW will be unchanged if an interrupt comes when connected.

EMT 206 ; SYSBUF ; SYSTEM BUFFER COMMUNICATION CONTROL

This is used to coordinate access to the low core user's buffer (maximum size 16 words). This could facilitate inter-partition data transfers. For example, the 16-word space could hold a file description block. The buffer is limited in size due to low core total space restriction.

- **System Elements**

Buffer semaphore, USEMA location 1100



Set up ID indicates who is in control of the buffer.
Ready busy indicates a read is in progress.
Read access indicates when partitions have read since a write.
Write busy indicates a write is in progress.

Buffer pointer, USRBUF location 356.

Low core buffer (location is system dependent).

```
.word 0          ; length of data stored in bytes
.blkw 16.        ; data storage area
```

All 17 words are transferred to and from user.

- **User Elements**

R0 indicates type of access.

bit	7 = 0	read
	= 1	write
bit	15 = 0	use system
	= 1	override system

R1 contains the address of the buffer in user's space to or from which the data is transferred.

Errors are indicated by C bit = 1. The semaphore, USEMA, is returned in the user's R0.

EMT 212 ; MTW18 ; WRITE BUFFER TO MAGTAPE

EMT 213 ; MTEW18 ; WRITE BUFFER TO MAGTAPE WITH EXTENDED IRG

EMT 214 ; MTR18 ; READ BUFFER FROM MAGTAPE

These EMTs allow the user to specify an 18-bit memory address for magtape I/O.

Calling sequence:

– R0 contains the high order base address, bits 16 and 17 –

– R1 contains the low order base address, bits 0 through 15 –

– R2 contains the word count –

EMT {212 / 213 / 214}

Error conditions:

See error conditions for EMTs 17, 20 and 21.

UNUSED EMTs

EMTs 166 and 207 through 377 are reserved for use by Tektronix, Inc.

EMTs 170 through 177 are reserved for users.

NETWORKING OPTION EMTs

The following EMTs are available to the user for use with the networking option. For descriptions, see the *TEKTEST III, Version 4, Networking Option* manual.

EMT 204 ; ININET ; INITIALIZE NETWORK PARTITION

EMT 205 ; NETWRK ; NETWORK REQUEST

The following EMTs are used with the networking option, but are not available to the user.

EMT 207 ; DLNIO ; DEVICE DRIVER FOR DL-11 INTERFACE

EMT 210 ; DMCNIO ; DEVICE DRIVER FOR DMC-11 INTERFACE

EMT 211 ; TTLSTF ; TERMINAL LINE TRANSFER

DATA BLOCK STRUCTURES

The IOT instruction set permits device-independent I/O with data types frequently used in a test program. File access is primarily sequential with provisions for random access on mass storage devices. The data structure set up by either LOG or REDUCE is illustrated in Figures 4-1 through 4-4.

Figure 4-1 shows the relationships between the three main data structures used. (A fourth structure, the PSA, contains a pointer to the master data block, the STSBLK. This pointer is fetched by EMT 116; the pointer is returned in R3. The pointer is set by moving the address to R3 and executing EMT 117.) The STSBLK is the master data block. It contains pointers to the other two types of data blocks, the DEVBLKs and LUNBLKs.

The LUNBLKs define each LUN. LUNs are numbers in the range 0-15 decimal and define the device and data type. To conserve space, the LUNBLKs contain pointers to the third data type, the DEVBLKs. Each LUN in a LUNBLK is a pointer; LUN 0 has word 0, LUN 1, word 1, etc. If a LUN is not assigned, the pointer in the associated word is 0. Many LUNs can point to the same DEVBLK. LUNBLKs are located sequentially in core, one LUNBLK per station. (Background has 1 LUNBLK.) Station 1 LUNBLK is lowest in core; Station 4 is highest in core.

The STSBLK contains information necessary to ASSIGN and CLOSE LUNs as well as the operation of the system. CURDEV, the first word of the STSBLK, points to the current DEVBLK. This must be set for most of the IOTs to work. One IOT, 0, will set this word. CURLUN points to the current LUNBLK. This is set by IP3260. The next three words are used primarily for assigning and releasing devices. DEVBOT points to the first word of the lowest DEVBLK. DEVTOP points to the first word above the highest DEVBLK. LUNBOT points to the base of all LUNBLKs. PARITY is a counter which keeps track of the number of parity errors in accessing mass storage devices. FREPTR points to available memory (in the same manner as DEVTOP/DEVBOT), including the resident test program. LUNNUM is the number of LUNBLKs (equal to the station count in foreground, except there is always at least one LUNBLK in foreground for TCM).

LOKOUT is a counter to prevent ↑C or STOP from interrupting a process which must be completed. To prevent being aborted, increment LOKOUT. Decrement it when an abort is again safe. LOKOUT = 0 implies the program may be aborted. The Instruction Processor checks an internal abort flag after processes that may have locked out STOPs so that the interrupt is not lost. STABLK (0 in background) points to the foreground data logging status block base. These blocks are described in Figure 4-6 and are used primarily by IP3260.

The DEVBLK is the data structure that defines the I/O for a given LUN. Figures 4-3 and 4-4 diagram the data structure for BG and FG, respectively. The byte count and bit maps are used for ASSIGNing and RELEASEing the DEVBLKs. The device code is the first word of data used for the I/O operations. The codes for each of the devices are given in Figures 4-3 and 4-4. The unit is the device unit (as in DK1, MT0, . . .). The system switch is a bit map that describes various attributes of the device. A description is given in Figure 4-5.

For nonbuffered devices, there are no additional words in the DEVBLK. For buffered devices, there exists an additional set of status words.

The four-word file name is the RAD50-packed file name. It is all 0 for the card reader and non-TEKTEST-structured mag tapes. The next three words are pointers for mass storage. All are absolute addresses. Last Block is the first block not in the file (Start Block + Length). Start Block is the absolute address of the first block of the file. Current Block is the absolute address of the current block in core. The next three pointers point to the in-core buffer. In foreground, this buffer is located in kernel space and can only be accessed through the IOTs. The foreground DEVBLK and its DEVBLK buffers do not use any of the space available to test programs. In background, the DEVBLK buffer is located in user space and may be directly accessed by user programs. Current Pointer points to the next word/byte to be read. End Pointer points to the first address out of the buffer. Begin Pointer points to the DMA count. All of these pointers are absolute addresses. Initial condition of the Current Pointer (CURPTR) in DEVBLK is as follows.

Input LUN – CURPTR is same as End Pointer.

Output LUN – CURPTR is same as Begin Pointer.

The next set of words is the System Status words. They are words used by Tektronix-supplied programs and functions. The number and use of the words are file-type dependent (see Figure 4-7). The next set of words is the User Status words. Their number is specified by the ASSIGN command and their use determined by customer-supplied subroutines and functions. Both sets of status words are optional.

The following pages describe each IOT in detail. Unless otherwise noted, the first word of the STSBLK is used to obtain a pointer to the data set (DEVBLK) describing the device. For some operations (mainly output operations), CURDEV = 0 results in a NOP. All registers are saved and restored, except when data is passed back in a register. All condition codes are set to 0 except where noted.

Figure 4-8 shows a special case of the DEVBLK (see Figure 4-3). It is the format used for input LOG files showing in detail the 14 system status words in background.

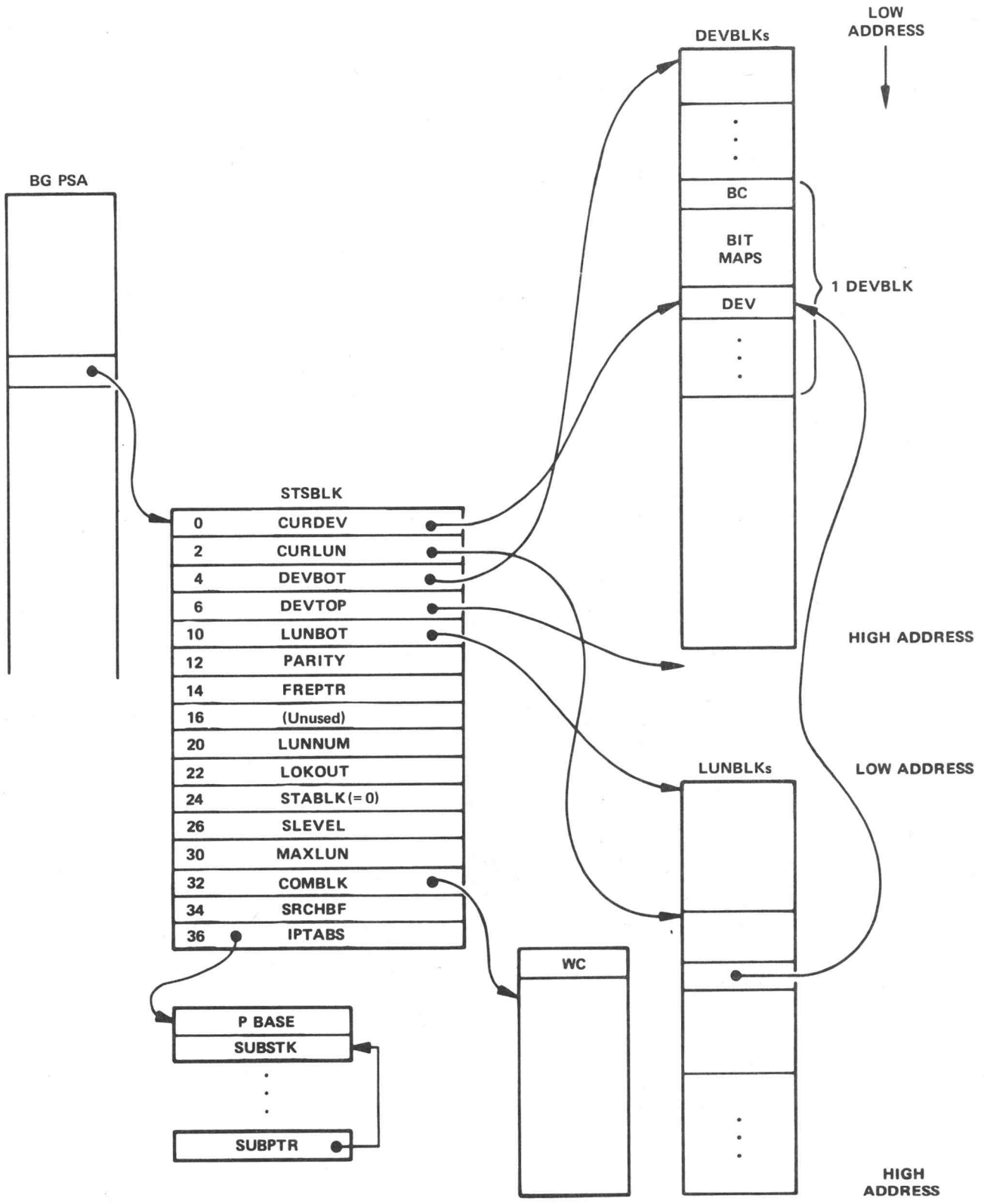


Figure 4-1. Background IOT Data Structures

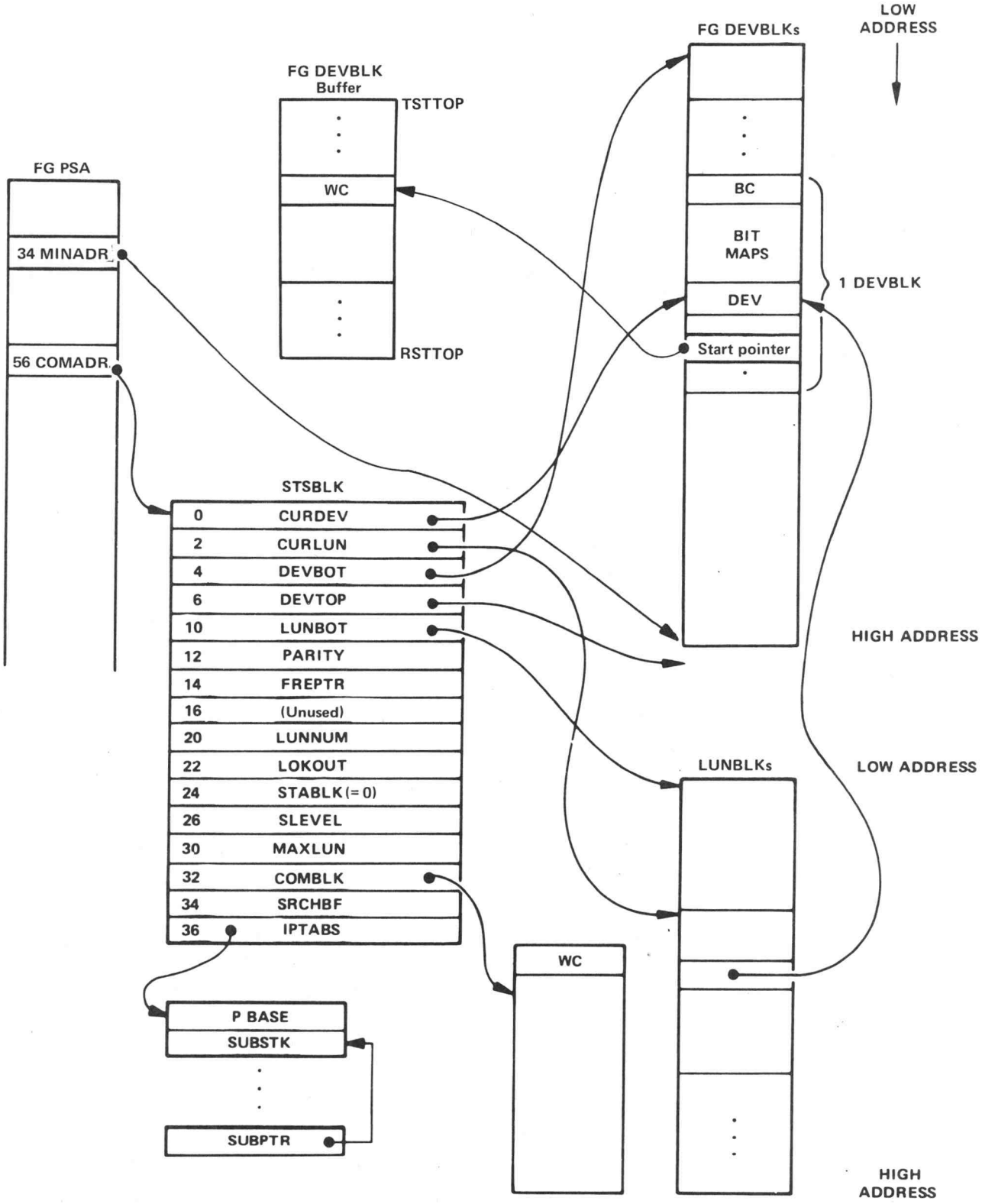


Figure 4-2. Foreground IOT Data Structures

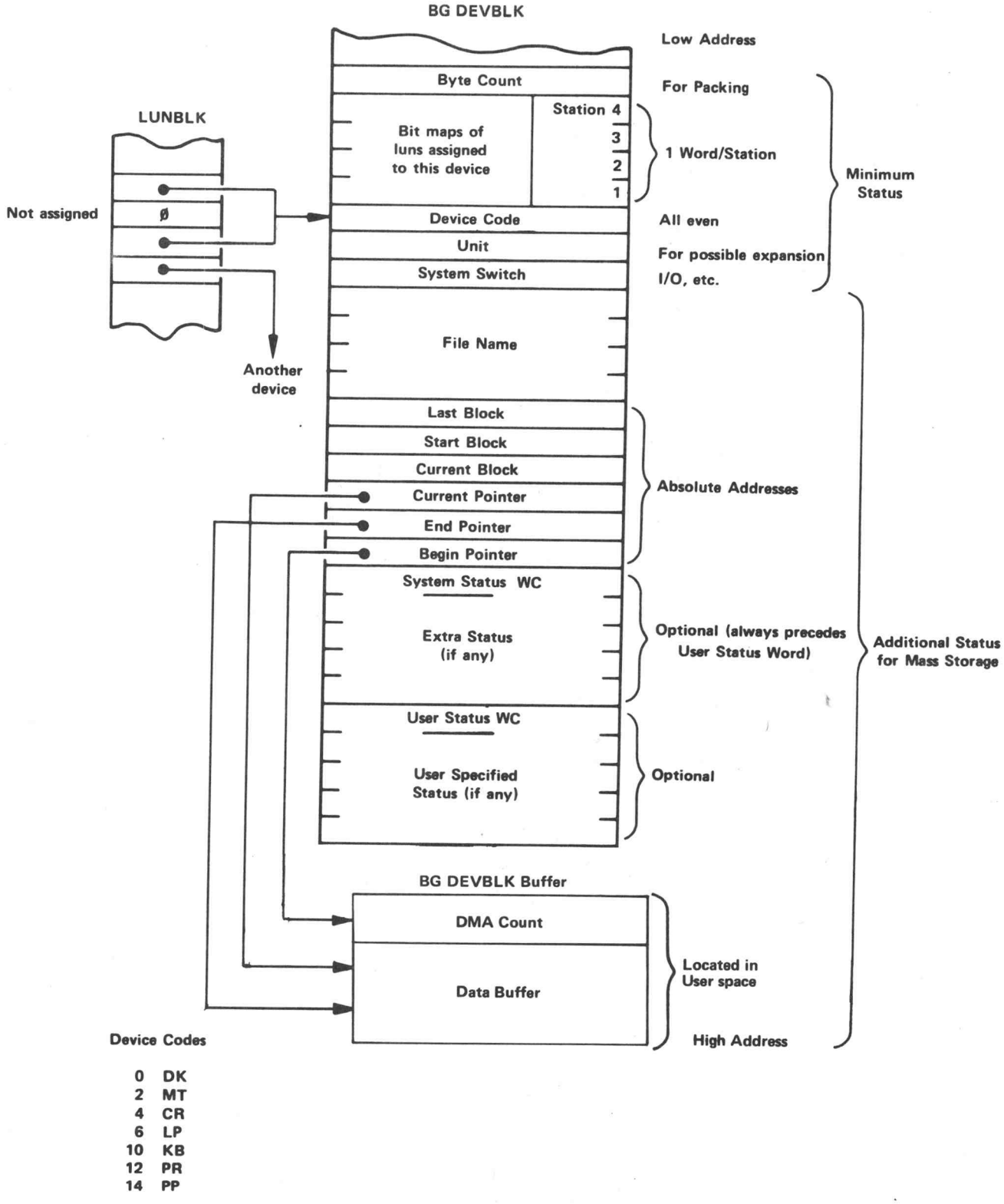


Figure 4-3. Background DEVBLK

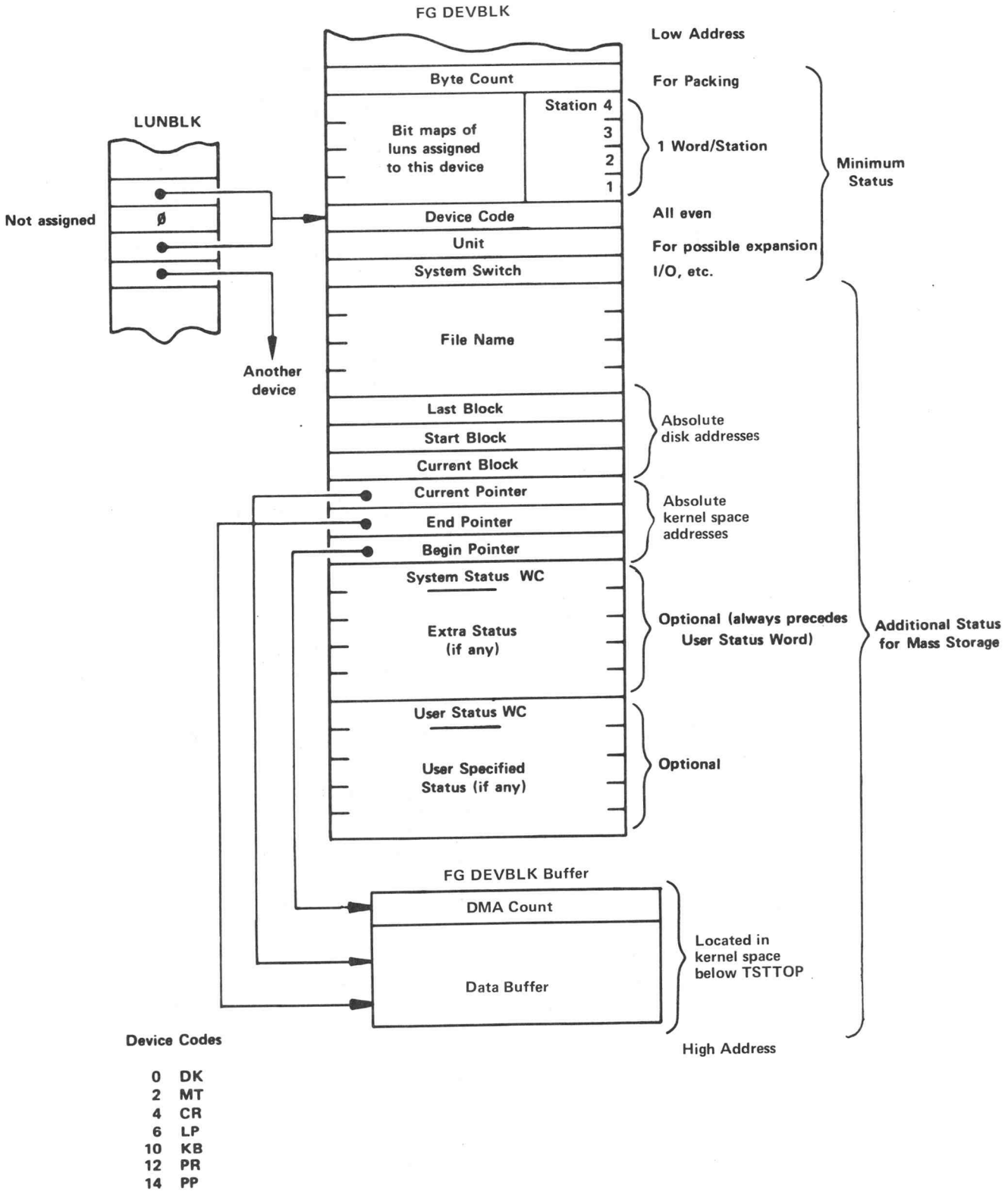


Figure 4-4. Foreground DEVBLK

SYSTEM SWITCH WORD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Buffer data not correct	Reserved	Reserved	Reserved	Not allocated	User status words	Reset	No rewind on MT	EOF	ASCII	Binary	System status words	TEKTEST file structured	Card image	Output	Input

1 – Has this attribute
0 – Does not have this attribute

BIT ASSIGNMENTS FOR LOG TYPES S-3260/S-3030

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X	X	Log Marker	Log Array	Log Iarray	Log Par	Log Register	Log Force	Log Error

1 – Log this type
0 – Do not log
X – Reserved

DEFAULTS:
SAMPLE RATE = 1
LOG ALL DATA

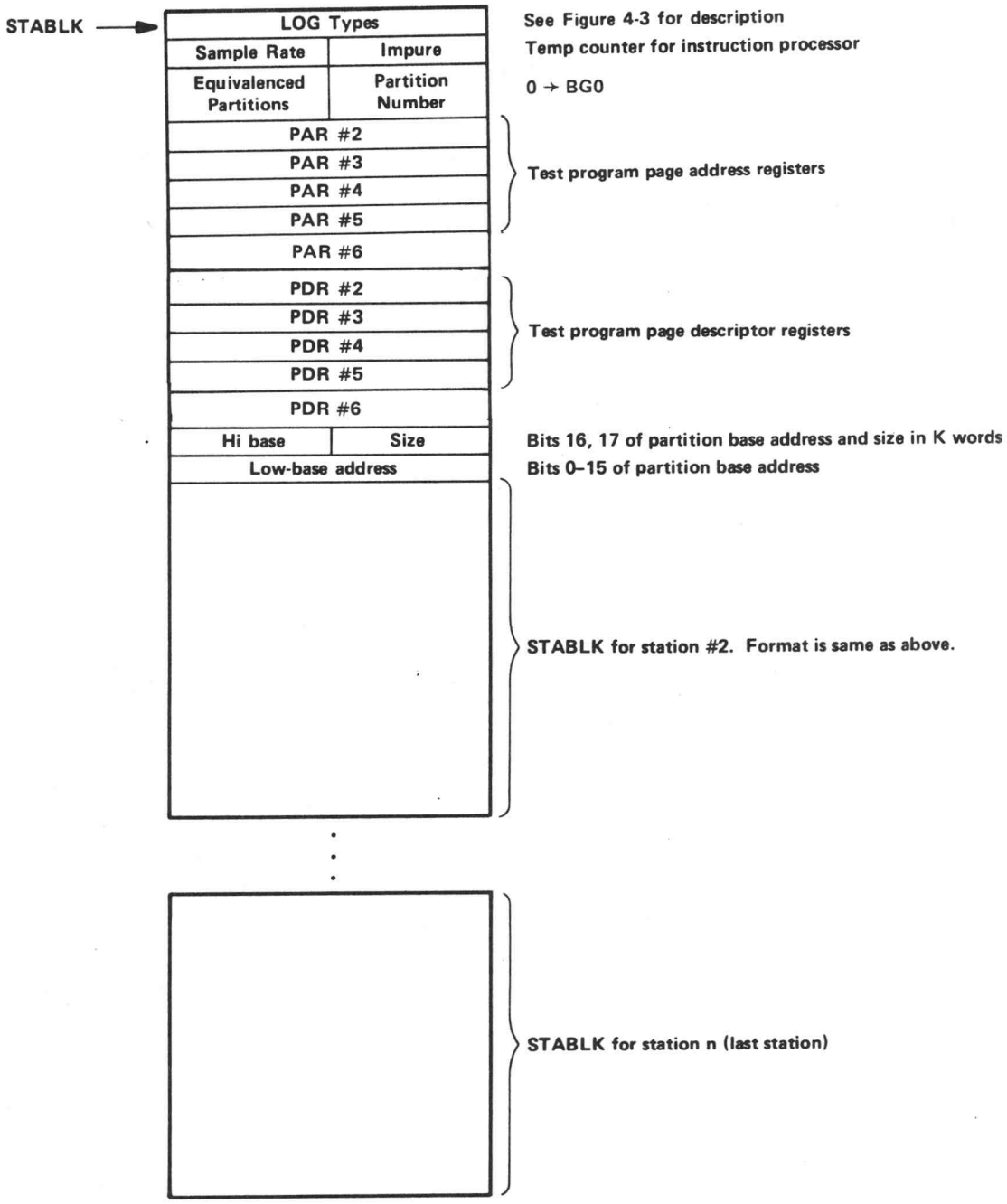
S-3455

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X												
				ERROR MASK RAM	BUFFER RAM	LOG BINARY (LOGFUNCTION)	LOGWAFER	LOG MARKER	LOGGEN	LOG ERRORS (INCLUDE PINS)	LOGVALUE	UNSATISFACTORY PAR	SATISFACTORY PAR	FAIL PAR	PASS PAR

Figure 4-5. Bit Assignments

23411-03

DATA LOGGING STATION STATUS BLOCK

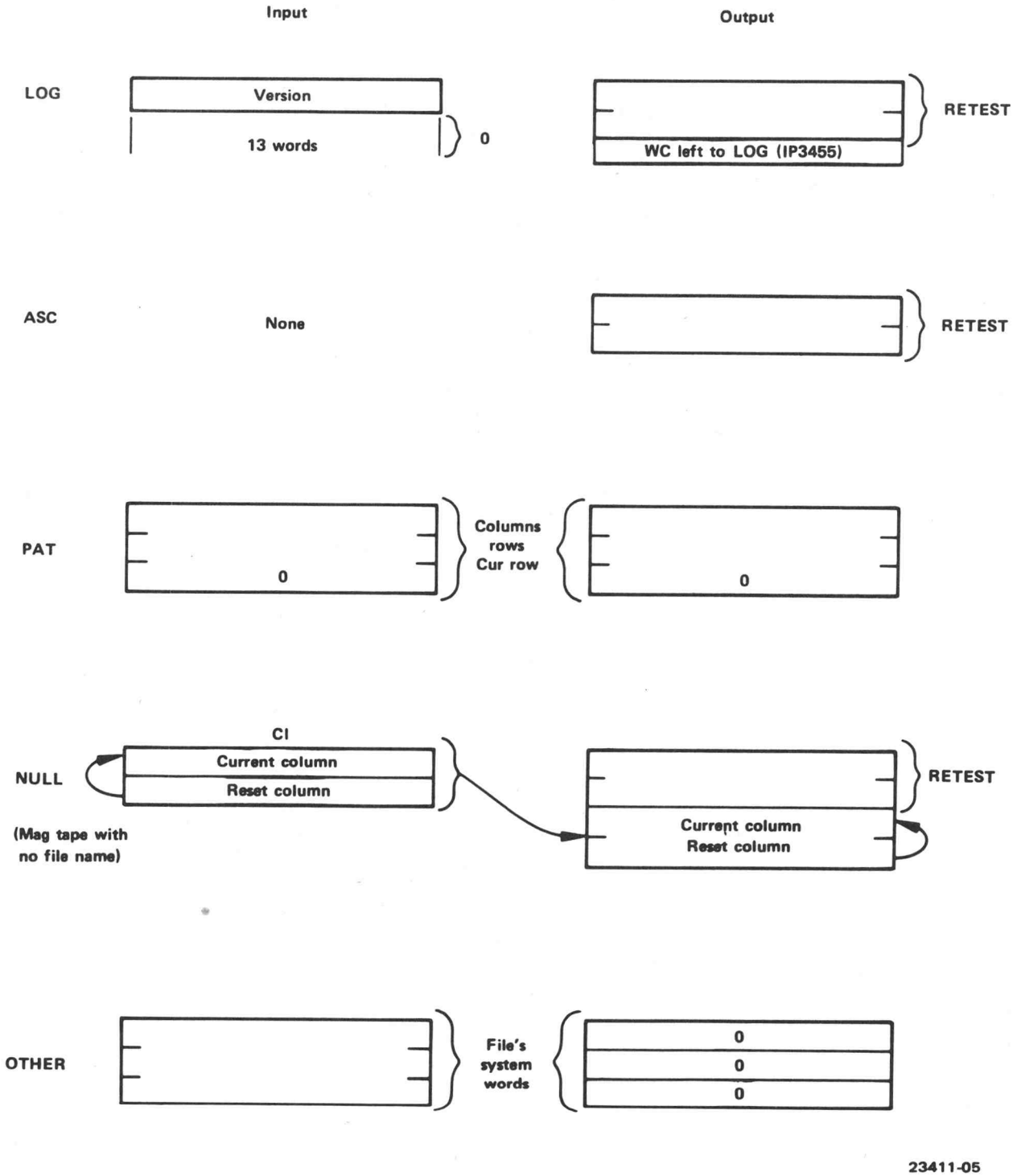


NOTE

On an S-3455 only the first two words of each STABLK exist. On these systems the block for station #2 is at STABLK + 4.

Figure 4-6

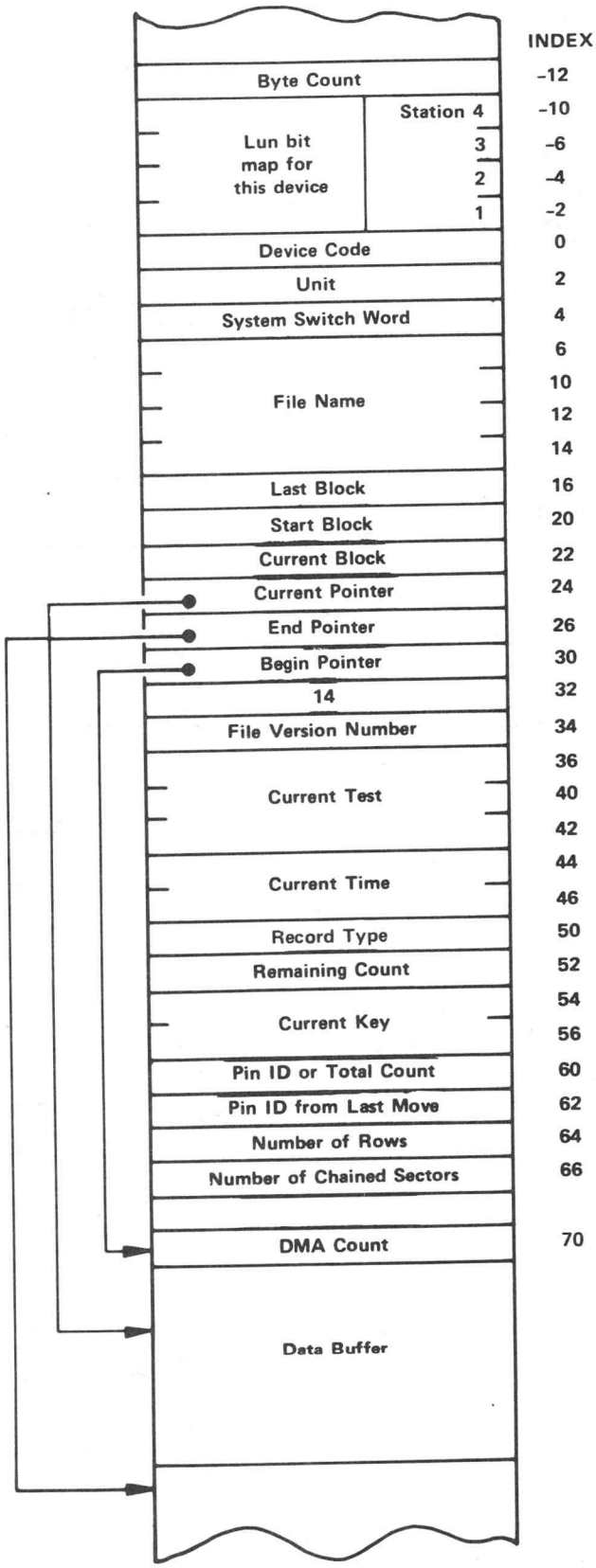
SYSTEM STATUS WORDS FOR VARIOUS FILE TYPES



23411-05

Figure 4-7

DEVBLK FOR INPUT LOG FILES
IN BACKGROUND



23411-06

Figure 4-8

Floating Point Subroutine (FIX)

Routine to fix floating point number pointed to by the first item on the R5 stack. The absolute value of the floating point number is returned as a 16-bit unsigned integer in R1.

This routine returns a rounded integer. Only R1 is modified.

Calling Sequence:

```
– R5    Points to argument stack –
JSR     PC, @ #110
– R1    Unsigned fixed integer –
```

NOTE

This subroutine requires floating point hardware.

Unsigned Bit Subroutine (FLOAT)

Routine to float 16-bit unsigned integer in R1 and return with the first item on the R5 stack pointing to the floating point number.

R1 is modified.

Calling Sequence:

```
– R1    Unsigned integer –
 JSR    PC, @ #114
– R5    Points to floating point number –
```

NOTE

This subroutine requires floating point hardware.

Integer Part Subroutine (IPART)

Routine extracts integer part of floating point number on R6 stack and returns a floating point 'integer' on the R6 stack. The original number is lost.

R0 and R1 are modified.

Calling Sequence:

(SP)	**Floating point number**
2(SP)	**Floating point number**
JSR	**PC, @ #120**

NOTE
This subroutine requires floating point hardware.

Floating Point Reference Number Subroutine (CMPTBL)

Routine to compare a floating point reference number against an ordered table (smallest to largest) of floating point numbers, and return the number of table entries less than the reference number.

Calling Sequence:

```
– R0    points to floating point reference number –
– R1    points to floating point table –
– R2    contains the number of table entries –
 JSR    PC, @ #124
```

Return Conditions

R2 has been decremented for each table entry that is lower than reference value. R2 = 0 implies over-range. R0 and R1 are modified.

NOTE
This subroutine requires floating point hardware.

Binary to BCD Subroutine (BINBCD, DBLBCD)

BINBCD changes binary numbers into BCD numbers. The BCD numbers are packed four characters per word. When the routine is called, R0 contains the binary number to be formatted. The routine returns with the low order four BCD digits in R0 and the binary residual in R1. All other registers are unchanged.

Calling Sequence:

```
JSR  PC, @ #130
```

DBLBCD is the same as BINBCD except that a double precision binary number is expected in R0–R1.

Calling Sequence:

```
JSR  PC, @ #134
```

R1 = 2	if set up to measure current from driver
R1 = 4	if set up to measure voltage, differential
R1 = 6	if set up to measure time
R1 = 8	if set up to measure time from reference
R1 = 10	if set up to measure voltage, single ended

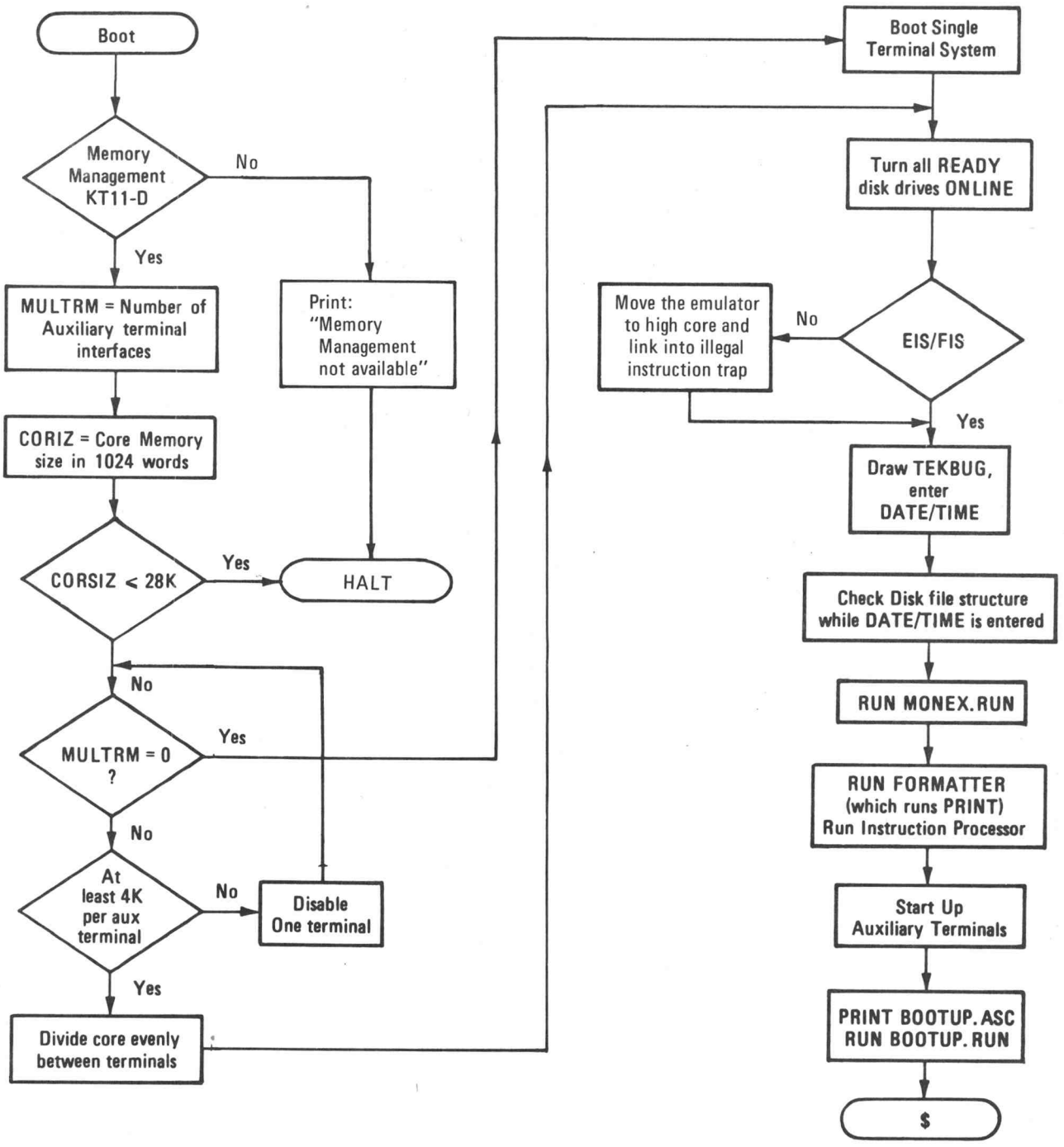
(R5) →	Pointer to maximum value (floating point) (next word)

```
JSR     PC, @ FMTSS     ;000152
BCS     ERROR           ;ILLEGAL RANGE
```

Returns with:

R0 = Formatted range

TEKTEST III BOOTUP PROCEDURE

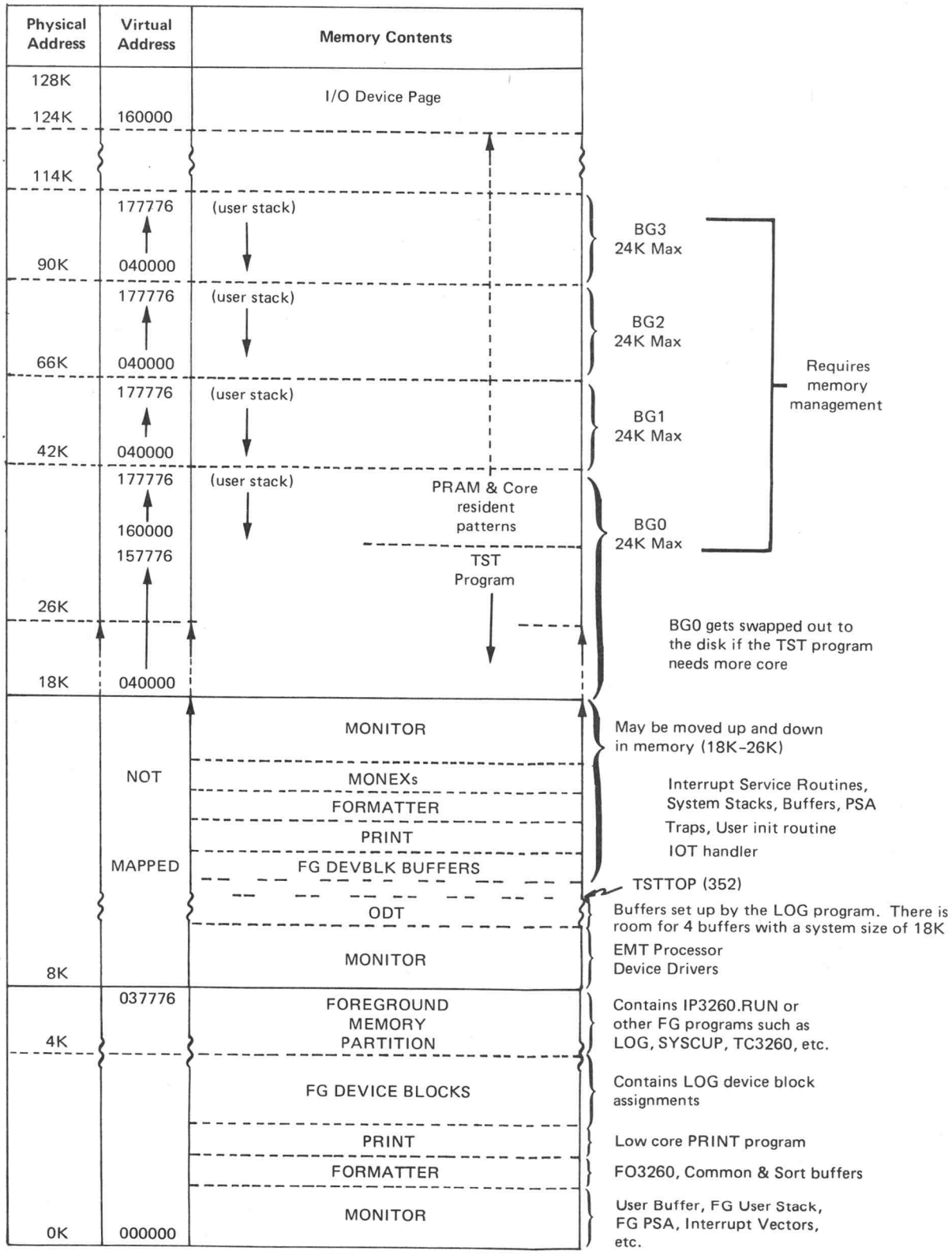


23411-08

TEKTEST III, VERSION 4.0 MEMORY MAP

This release of TEKTEST requires that a system has memory management hardware. Both a foreground test program and background users run in user space, with only a small portion of the monitor mapped into the user virtual address space. The rest of the monitor resides in kernel space. All interrupts are handled by routines mapped into the kernel.

Foreground DEVBLK buffers are mapped into kernel space only. These are only accessible to the user via IOTs. The system is set up to allow four DEVBLKs to be assigned in the foreground. To increase this number, or to allow space for large MONEXs, the entire portion of the high part of the kernel space monitor must be moved upwards. To change the system size, a word on the disk must be changed and the system rebooted. Disk block 3, location 50 stores the system size in "K" words (octal). Allowable values range from 18K to 26K, and the system size is displayed each time the system is booted.


Physical Address
Virtual Address
Memory Contents
128K
124K
160000
I/O Device Page
114K
177776
(user stack)
90K
040000
BG3
24K Max
177776
(user stack)
66K
040000
BG2
24K Max
177776
(user stack)
42K
040000
BG1
24K Max
177776
(user stack)
PRAM & Core resident patterns
160000
157776
TST Program
BG0
24K Max
Requires memory management
26K
BG0 gets swapped out to the disk if the TST program needs more core
18K
040000
MONITOR
May be moved up and down in memory (18K-26K)
NOT
MONEXs
FORMATTER
Interrupt Service Routines, System Stacks, Buffers, PSA
PRINT
Traps, User init routine
MAPPED
FG DEVBLK BUFFERS
IOT handler
TSTTOP (352)
ODT
Buffers set up by the LOG program. There is room for 4 buffers with a system size of 18K
MONITOR
EMT Processor
Device Drivers
8K
037776
FOREGROUND MEMORY PARTITION
Contains IP3260.RUN or other FG programs such as LOG, SYSCUP, TC3260, etc.
4K
FG DEVICE BLOCKS
Contains LOG device block assignments
PRINT
Low core PRINT program
FORMATTER
FO3260, Common & Sort buffers
MONITOR
User Buffer, FG User Stack, FG PSA, Interrupt Vectors, etc.
0K
000000

Address	Name/Contents	Description
200	LP	LP11/LS11 Line Printer interrupt vector
202	000340	
204	"0X	PEDIT pattern characters 00, 01
206	"1C	10, 11
210	DISABL	Non-zero → don't disable test station at end of test. Used by handler/prober auto restart.
212	DISLGT	Contains the value to be sent to BSTATN at the end of the test to turn on TSCU lights (NOT a pointer).
* 214	MC3TBL	Pointer to MC3 range table
* 216	MC3CNT	MC3 loop count — function of line frequency
220	DK	RK11/RK05 disk pack interrupt vector
222	000340	
224	MT	TM11/TU10 mag tape interrupt vector
226	000340	
230	CR	CR11/CD11 card reader interrupt vector
232	000340	
234	ACTSTA	Current active station number
236	MULTRM	Number of auxiliary terminals (background partitions) 0 → single terminal system.
240	BGSWAP	Non-zero → BG0 has been swapped onto disk.
* 242	PRGNAM	Current test program number (HEX)
244	000246	Floating-point unit trap
246	000002	RTI
250	KT11D	Memory management trap
252	000340	
254	VERDIC	Verdict pointer (used by RUNSEQ subroutine)
256	EXTCORE	Sign bit → memory management Upper byte → total memory on system in K Lower byte → memory free
260		Unused
262	TSTVER	TST program version number (see page 7-36)
264	PSA0	Pointer to PSA0
266	PSA1	Pointer to PSA1 (0 → does not exist)
270	PSA2	Pointer to PSA2 (0 → does not exist)
272	PSA3	Pointer to PSA3 (0 → does not exist)

*Different for TEKTEST IV (S-3455). See the table of differences.

Address	Name/Contents	Description
274	WAVCOM	Waveform communication flag
276	ITVTAB*	Pointer to 1340 interrupt table for MOD 80
300	KBD1	Keyboard 1 interrupt vector
302	000340	
304	TTY1	TTY 1 interrupt vector
306	000340	
310	KBD2	Keyboard 2 interrupt vector
312	000340	
314	TTY2	TTY 2 interrupt vector
316	000340	
320	KBD3	Keyboard 3 interrupt vector
322	000340	
324	TTY3	TTY 3 interrupt vector
326	000340	
*330	MATRX2	Bus address of 4 x 64 matrix (SM1)
332	INITBL	Pointer to hardware initialize routine (in formatter) for instruction processor (JSR PC, @ INITBL).
334	EMTBLE	Pointer to EMT dispatch table
*336	COMMON	Pointer to common storage area
340	FGSTRT	Pointer to subroutine used to start up a foreground program from an interrupt (JSR R5, @ FGSTRT).
342	FGEXIT	Pointer to foreground exit routine (JMP @ FGEXIT)
344	FGACTV	0 → foreground not active
346	STATE	Current system operating state: Negative → idle Zero → foreground Positive → background
350	FGSTOP	0 → foreground clear to go active (used internal to monitor) (see EMT72, page 3-68)
352	TSTTOP	Test program top memory address
354	IPPNTR	Pointer to table of pointers within IP3260/TC3260 or IP3030/TC3030. (see page 6-9)
356	USRBUF	Pointer to 17-word user buffer in low core
360	HSTBAS	Pointer to sort buffer base
362	INITHW	Pointer to subroutine used to initialize all 1340 data couplers (JSR PC, @ INITHW)
364	DLYPTR	Pointer to delay routine in instruction processor (used to process a TRAP 000)

*Different for TEKTEST IV (S-3455). See the table of differences.

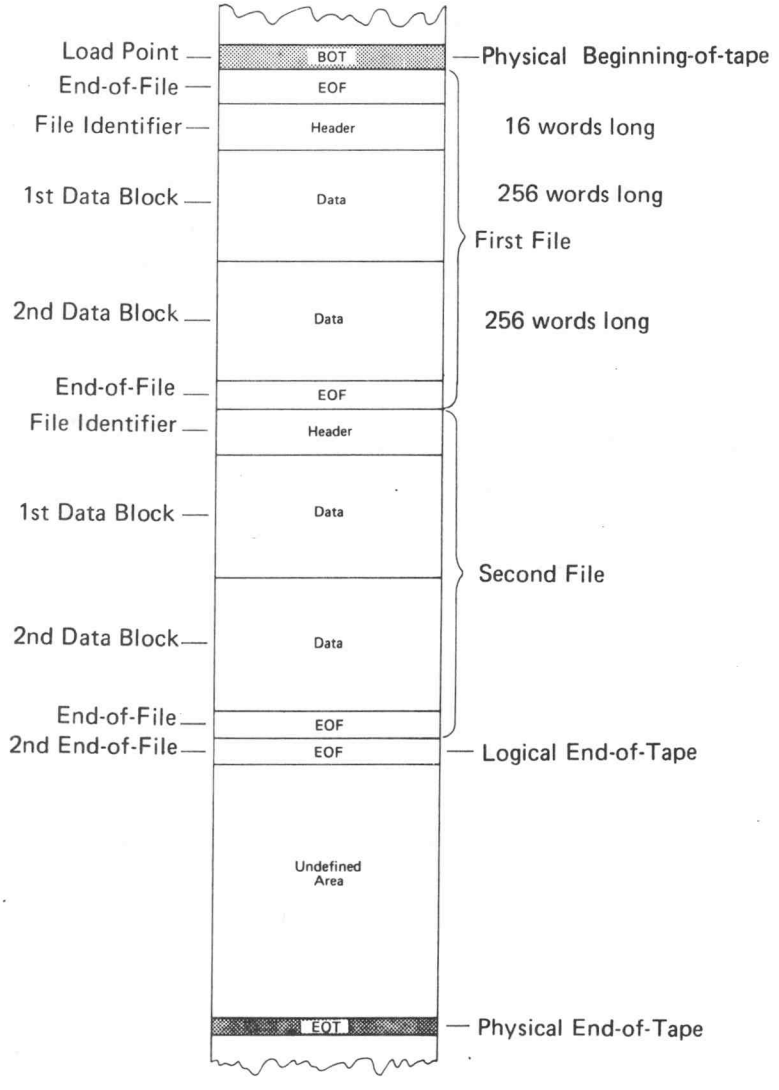
Word	Contents	Comments
024		Number of disk blocks available for program storage between monitor and directory.'.
026		First free block past monitor.
030		Number of disk blocks available after directory and free list.
032		First free block after free list.
034		Number of directory blocks.
036		First directory block number.
040		Number of free list blocks.
042		First free list block number.
044-046	**RAD50/TMON /**	System Executive.
050		Length of TMON in blocks.
052		First disk block of TMON.
054-056	**RAD50/LOAD /**	LOAD definition.
060		Length of LOAD in blocks.
062		First disk block of LOAD.
064-066	**RAD50/SMR /**	SMR definition.
070		Length of SMR in blocks.
062		First disk block of SMR.
074-076	**RAD50/TEKBUG /**	TEKBUG definition.
100		Length of TEKBUG in blocks.
102		First disk block of TEKBUG.
104	**000000**	End of system directory

Physical Structure

The data is blocked into 256 16-bit words. There are always a whole number of blocks in the file, even on 9-track magnetic tape, which will support smaller blocks. Block boundaries may occur anywhere. All files have a 16-word file descriptor entry on 9-track magnetic tape.

This entry is contained in a separate block at the beginning of the file. On the system disk, this entry is in the disk directory with the other files.

Magnetic Tape Physical Structure (inter-record gaps not shown)



When binary 16-bit words are stored on the tape, the low order 8 bits are stored first, followed by the high order 8 bits.

TWO TERMINAL ODT FOR MEMORY MANAGEMENT

VERSION 2.01

ODT is a monitor extension which runs in high memory and allows the user to monitor from terminal #1 a program which is running on terminal #0. It can also be used to monitor a program running in foreground. However, this requires some special conditions which must be created by the user manually.

ODT must be loaded in high memory from terminal #0 by running the program **ODT.RUN:SYS**. When run on terminal #0, this will result in the running of the foreground program **ODTFG.RUN:SYS**, which in turn loads into high memory the overlay **HIODT.OVR:SYS**. This overlay then becomes a part of the monitor until such time as the user tells ODT to return high memory to the monitor for other use.

To activate ODT, the user runs the program **ODT.RUN:SYS** on terminal #1 and, if ODT has already been loaded, it will prompt with an asterisk (*). ODT may be temporarily suspended on terminal #1 at any time by entering a control-S. This will return terminal #1 to TMON and will leave terminal #0 suspended. ODT may be reactivated by again running **ODT.RUN:SYS** on terminal #1. Terminal #0 will be in a STOP condition and may be restarted by using the CONTINUE, GO or SINGLE STEP commands. ODT may be removed from high memory by entering a control-C to it on terminal #1.

In the following list of commands, <CR> represents the return key, and <LF> represents the line feed key. Addresses are represented by the string (LOCN), where the N identifies the specific address. Various conditions may be monitored for and when any one of them is found to exist, the program will be halted and the reason for the halt will be given by a letter-digit code indicating the type of halt and which one of that type. In addition, the current PC, the location of the next instruction and the value of the next instruction will be displayed.

Command Summary:

G	Go	X	Open RAD50	\	Open byte
B	Break point	P	Protect	<	Reopen
S	Single step	T	Trap	F	Find
K	Mode	C	Continue	A	Base Address
L	List	M	Monitor	+	Hash total
/	Open word				

Protect a register or memory location against alteration.

Pn; <LOC><CR>	Set protector Pn for location <LOC>.
Pn; RM<CR>	Set protector Pn for register m.
Pn <CR>	Clear protector Pn.
PL	List the settings of all protector values.
PC	Clear all protectors.

There may be up to 10 locations protected at the same time. The n may be any digit from 0 to 9. Protected locations are checked after each instruction and if one has been altered, the program will be halted.

BREAK POINT (stop when the PC reaches a predetermined value).

Bn; <LOC1>-<LOC2><CR>	Set break-point group Bn for locations <LOC1> through <LOC2>.
Bn<CR>	Clear break points previously assigned to break-point group Bn.
BC	Clear all break points in all groups.
BL	List the range for each break-point group.

There may be up to 10 break-point groups at the same time. The n may be any digit from 0 to 9. The PC is checked before each instruction and if it falls within the range of one of the break-point groups, the program will be halted.

TRAP ON INSTRUCTION VALUE

Tn; <INST>(MASK)<CR>	Halt if the next instruction ANDed with MASK is equal to (INST). If the value of the MASK is omitted, a value of 177777 is assumed.
Tn (CR)	Clear this instruction trap value.
TC	Clear all instruction traps.
TL	List the values of the instruction traps.

There may be up to 10 instruction traps at the same time. The n may be any digit from 0 to 9.

PROTECTING A RANGE OF MEMORY AGAINST ALTERATION

+n; <LOC1>-<LOC2><CR>	Hash total locations <LOC1> through <LOC2> after each instruction and halt if there is a change in the total.
+n CR	Stop hash check for this entry.
+C	Stop all hash total checks.
+L	List all hash total ranges.

There may be up to 10 hash ranges at the same time. The n may be any digit from 0 to 9.

DEBUG MODE SWITCH

KY	Switches debug mode to kernel.
KN	Switches debug mode to user.
KU	Allows debugging of both kernel and user. When in single step mode, a K or U is printed to indicate the current mode.
<LF>	Displays the current debug mode and the present PC.

NOTE

When running ODT, any attempt to CLOSE a file that was previously opened by LOG, will hang the system. After entering ODT, if LOG files are still open, the operator is given warning of this possible error condition with the message:

"WARNING: BUFFERED DATA BLOCK(S) ABOVE ODT"

PC RELATED COMMANDS

G;<LOC><CR>	Force the value <LOC> into the PC and start execution at this point.
S	Execute the next instruction and halt. When the halt occurs, the value of the PC will be given along with the value of the next instruction.
C	Resume execution of the program.
MY<CR>	Continuously record the last 32 PC values.
Mn<CR>	Stop recording PC values.
L	List the last 32 recorded PC values. The most recent will be at the bottom.

BASE REGISTER VALUES (setting and reference to)

An; <LOC><CR>	Set the value <LOC> into base register n.
An <CR>	Set base register n to 0.
AL	List the base register values.
AC	Set all base registers to 0.

There are 10 base registers, A0 through A9. Each register is initially set to 0. Any value may be placed in any register. Any register may be referenced at any time.

nnn,Am	Any entry that calls for an octal value may be entered in this format,which is interpreted to read nnn plus the contents of BASE REGISTER m.

SMR (SYSTEM MAINTENANCE ROUTINE)

SMR is a program used to examine and modify locations on a disk pack. It recognizes the directory file structure on a TEKTEST operating system, and can access files on a TEKTEST disk by reference to a file name. It can also be used with non-TEKTEST disks, although directory references are meaningless in that case.

SMR is entered from the Executive as follows:

$SMR
→

1. To examine a **RUN** file, enter:

   GET, <filename>

   Examine and modify the program as if the program was in core and you were using ODT.

2. To examine a non-run file, enter:

   GET, <filename>

   Examine and modify the file as if it were loaded into core at address 0.

   - or -

   Specify the relative block number with a **<number>R** command and examine and modify the block as if it were loaded into core at address 0.

3. To examine a block on the disk, enter:

   LOCATE, <number>

   Examine and modify the block as if it were loaded into core at address 0.

4. To examine/modify the **file directory**, enter:

   FIND, <filename>

   Examine and modify the entry as if it were loaded into core at address 0.

MONITOR EXTENSIONS

EMT 170 through 177 are reserved for users to write monitor extensions. The monitor extensions are incorporated into the Executive at bootup with the use of a MONEX.RUN program.

The MONEX.RUN program is run by the bootup routine before the Formatter, PRINT program or instruction processor are loaded. MONEX.RUN should be linked with a T:40 000 (foreground program) and should contain code:

1. to relocate the user-written EMT to high core;
2. to enter a pointer to the EMT in the EMT dispatch table;
3. for the EMT itself.

See the example on the following page of a MONEX program used to create an EMT 177.

If the system displays the message:

```
NO MORE HIGH CORE MEMORY LEFT
INCREASE SYSTEM SIZE.  DISK BLK 3 LOC 50
REBOOT
```

the system has probably tried to write over the monitor as a result of running a MONEX. The combination of MONEX + FORMATTER + PRINT is too large to fit in available space. To increase available space, the system size must be changed and the system rebooted. Using SMR (Section 8), increase the system size word on disk block 3, location 50 and reboot. The system size is displayed, in decimal K words, under the version number. Allowable values range from 18K to 26K. (Also see Memory Map, Section 7.)

Example:

```
        .TITLE    MONEX
        .IDENT    /X02.20/

;       EMT DEFINITIONS:

        LOMEM  = 55                    ;CLAIM FOREGROUND HI CORE
        EXIT   = 60                    ;RETURN TO TMON

;       LOW CORE POINTERS:

        PSA0   = 264                   ;POINTER TO PSA0
        EMTTBL = 334                   ;POINTER TO EMT TABLE
        TSTTOP = 352                   ;FIRST UNUSED HIGH CORE LOCATION

;       FOREGROUND PROGRAM TO MOVE THE NEW EMT TO HIGH CORE

START:  MOV    @#TSTTOP,R0             ;POINTER TO FIRST FREE HIGH CORE LOCATION
        SUB    #EMTEND-EMTBEG,R0       ;SUBTRACT SIZE OF EMT
        MOV    R0,@#TSTTOP             ;NEW TOP OF HI CORE
        EMT    LOMEM
        MOV    R0,R1
        MOV    #<EMTEND-EMTBEG>/2,R2   ;NUMBER OF WORDS IN EMT
        MOV    #EMTBEG,R3
10$:    MOV    (R3)+,(R0)+             ;MOVE THE EMT
        SOB    R2,10$
        MOV    @#EMTTBL,R0             ;UPDATE THE EMT DISPATCH TABLE
        MOV    R1,177+177(R0)          ;ENTER ADDRESS OF NEW EMT 177
        EMT    EXIT                    ;RETURN TO TEKBUG
                                       ;AND CONTINUE BOOT

EMTBEG: .
        .
        .                              ;THE EMT ITSELF
                                       ;MUST BE PIC
EMTEND:
        .END   START
```

WRITING CUSTOM INTERFACE INITIALIZATION ROUTINES

The system is capable of transferring control to a user-written routine during initialization. This allows initializing hardware that is not defined by Tektronix as standard optional equipment. It is also useful with hardware that requires a sequence of operations for initialization. All standard and optional equipment is initialized by the system, when a foreground program is loaded into memory, through the TEKTEST language via the INITIALIZE command, or when a test program is started or stopped while running under either TCM or IP3260.

At boot-up, the FORMATTER looks for a file named FOINIT.FNC and, if found, the file becomes a part of the FORMATTER. Control is transferred to FOINIT.FNC whenever the system initializes. If the file is not found, the function is not enabled and standard initialization continues.

A system programmer should write the FOINIT.FNC routine as defined in the section on writing functions and subroutines. However, five differences from a standard .FNC routine format should be observed:

1. The FOINIT.FNC routine should not contain any subprogram entry definitions or header words.

2. Control is transferred to the first location of the routine.

3. Exit from the routine is through: RTS PC.

4. Only the I/O page, the first 4K of memory (low core), and the routine itself are accessible.

5. EMTs, such as EMT 0 or EMT 1, will not work when used from within the routine.

This function gives the user the ability to initialize hardware with addresses in the I/O page (160000 through 177776). Monitor services should not generally be used.