J-11

PROGRAMMER'S REFERENCE

Rev. 2.04 (January, 1982)

C O M P A N Y C O N F I D E N T I A L

REVISION HISTORY

REV	DATE	REASON
2.04	1/82	Minor corrections
2.03	7/81	Revised CPU Error Register and stack trap description; minor corrections
2.02	12/80	Added floating point instructions
2.01	7/80	Added Table of Contents; revised I/D space, CPU Error Register, and Memory System Error Register; reorganized chapters 3, 4, and 5
2.00	1/80	Added PS protection chart, console single step
1.04	12/79	Revised Cache Control Register and Memory System Error Register; added Hit/Miss Register; cleanup
1.03	8/79	Added comparison to 11/44; eliminated stack limit; title change
1.02	7/79	Eliminated instruction complete; cleanup
1.01	6/79	Added I/O bus time error bit in CPU Error Register; added CPU abort error bit in Memory System Error Register; added disable interrupt bit in Cache Control Register; fixed up original
1.00	5/79	Preliminary

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

1.0 INTRODUCTION

1.1 Scope

This document specifies the programmer-visible functions of the J-11, a high performance MOS CPU chip set for the PDP-11 family. The J-11 implements the important 11/44 and 11/70 features (see sections 10 and 11 for summary difference lists) and achieves 11/70 performance in most applications.

1.2 Method

The J-11 is intended to replace both the 11/44 and the 11/70. It will run RT-11, RSX-11M, RSX-11M+, RSTS/E, DSM-11, UNIX, and KSOS. The 11/44 and 11/70 are not entirely compatible. When a choice between conflicting implementations is necessary, the J-11 follows the 11/44 rather than the 11/70. The only exceptions are features which impact potential software coverage (e.g., dual register set) or which unduly complicate the MOS implementation.

1.3 Applicable Documents

J-11 Chip System Specification
PDP-11/70 Processor Handbook
J-11 Control Chip Specification
J-11 Data Chip Specification
J-11 Microprogrammer's Reference

2.0 INSTRUCTION SET

The J-11 instruction set consists of the following:

- 11/70 Base Instruction Set including the Extended Instruction Set (EIS) plus the MTPS, MFPS, MFPT, TSTSET, WRTLCK, and CSM instructions. Appendix 1 contains the complete list of J-11 base instructions.

- Floating Point (FP11) Instruction Set compatible with the FP11A/C/E floating point processors. Appendix 2 contains the complete list of J-11 floating point instructions.

- Commercial Instruction Set (CIS) compatible with DEC Standard 168. Appendix 3 contains the complete list of J-11 CIS instructions.

3.0 ARCHITECTURAL FEATURES

3.1 General Registers

These include:

- Two sets of six working registers (R0-R5)

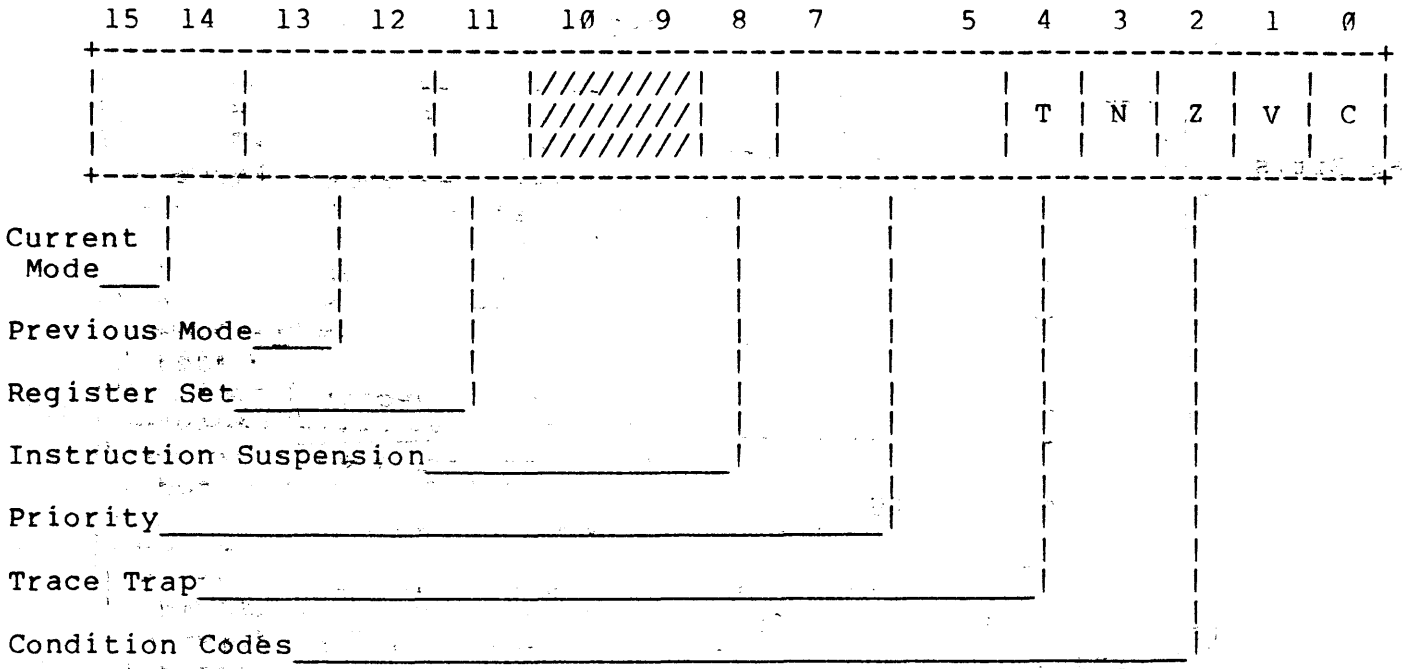
- Kernel/supervisor/user stack pointers (R6)

- Program counter (R7).

This is fully compatible with the 11/70. The 11/44 lacks a second general register set.

3.2 Processor Status Word - PS (17777776)

The Processor Status Word (PS) contains information on the status of the processor.



BIT	NAME	FUNCTION
15:14	Current Mode (RW, protected)	Current processor mode: ØØ = kernel Ø1 = supervisor 1Ø = illegal (traps) 11 = user.
13:12	Previous Mode (RW, protected)	Previous processor mode, same encoding as current mode.
11	Register Set (RW, protected)	General register set select: Ø = register set Ø 1 = register set 1.
8	Instruction Suspension (RW)	Set to indicate that a CIS instruction was suspended to service an interrupt.
7:5	Priority (RW, protected)	Processor interrupt priority level.
4	Trace Trap (RW, protected)	Set to force a trace trap.
3:Ø	Condition Codes (RW)	Processor condition codes.

For the protection on the PS under various conditions, see Table 3-1. The PS is initialized at power up (depends on power up options) and is cleared at console start. The RESET instruction does not affect the PS.

Table 3-1
PS PROTECTION

	RTI, RTT			TRAPS & INTERRUPTS		
PS Bit(s)	User	Super	Kernel	User	Super	Kernel
Condition Codes PS <3:0>	loaded from stack	loaded from stack	loaded from stack	loaded from vector	loaded from vector	loaded from vector
Trap Bit PS <4>	loaded from stack	loaded from stack	loaded from stack	loaded from vector	loaded from vector	loaded from vector
Processor Priority PS <7:5>	un-changed	un-changed	loaded from stack	loaded from vector	loaded from vector	loaded from vector
CIS Suspend Bit PS <8>	loaded from stack	loaded from stack	loaded from stack	loaded from vector	loaded from vector	loaded from vector
Register Select PS <11>	ORed from stack	ORed from stack	loaded from stack	loaded from vector	loaded from vector	loaded from vector
Previous Mode PS <13:12>	ORed from stack	ORed from stack	loaded from stack	copied from PS <15:14>	copied from PS <15:14>	copied from PS <15:14>
Current Mode PS <15:14>	ORed from stack	ORed from stack	loaded from stack	loaded from vector	loaded from vector	loaded from vector

Table 3-1 (continued)
PS PROTECTION

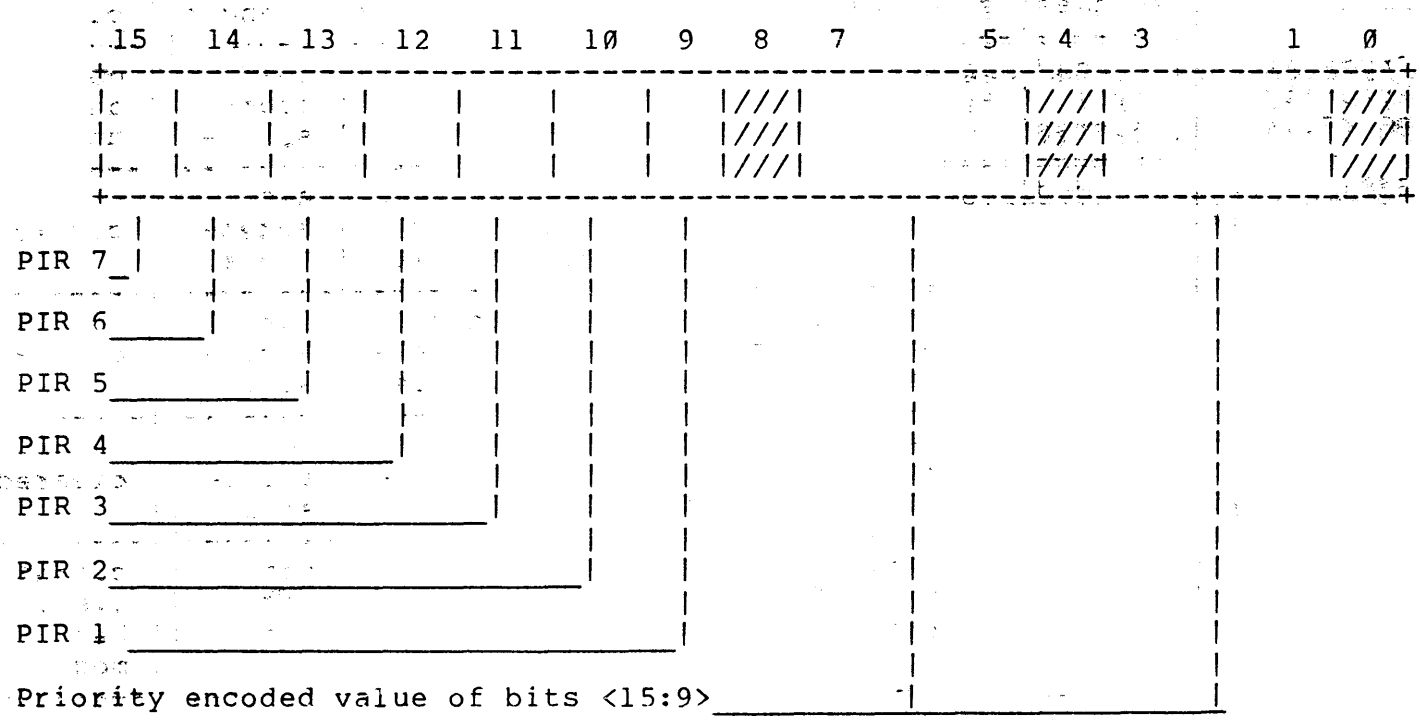
PS Bit(s)	EXPLICIT PS ACCESS			MTPS			POWER UP
	User	Super	Kernel	User	Super	Kernel	
Condition Codes PS <3:0>	loaded from source	loaded from source	loaded from source	loaded from source	loaded from source	loaded from source	cleared
Trap Bit PS <4>	unchanged	unchanged	unchanged	unchanged	unchanged	unchanged	cleared
Processor Priority PS <7:5>	loaded from source	loaded from source	loaded from source	unchanged	unchanged	loaded from source	depends on power up option
CIS Suspend Bit PS <8>	loaded from source	loaded from source	loaded from source	not accessible	not accessible	not accessible	cleared
Register Select PS <11>	loaded from source	loaded from source	loaded from source	not accessible	not accessible	not accessible	cleared
Previous Mode PS <13:12>	loaded from source	loaded from source	loaded from source	not accessible	not accessible	not accessible	cleared
Current Mode PS <15:14>	loaded from source	loaded from source	loaded from source	not accessible	not accessible	not accessible	cleared i.e., kernel mode

3.3 Program Interrupt Request Register - PIRQ (17777772)

The Program Interrupt Request Register (PIRQ) implements a software interrupt facility.

A request is queued by setting one of the bits <15:9> which corresponds to a program interrupt request at levels 7-1. Bits <7:5> and <3:1> are set by hardware to the encoded value of the highest pending request set.

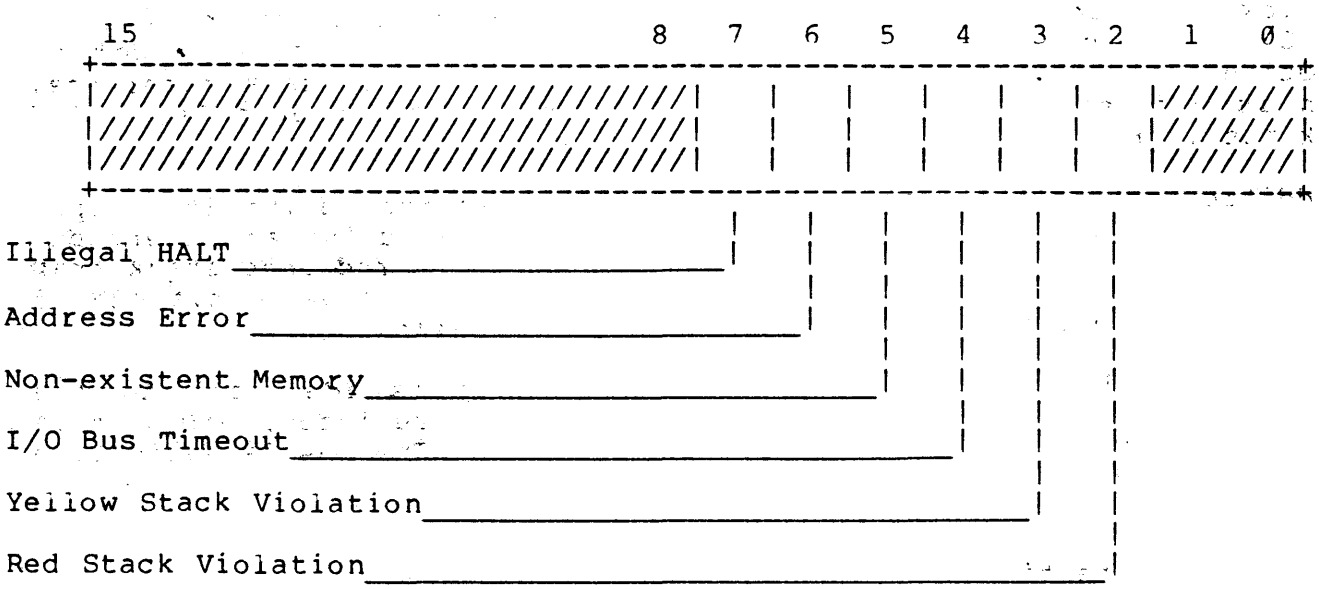
When the program interrupt request is granted, the processor traps through location 240. It is the interrupt service routine's responsibility to clear the appropriate bit in PIRQ before exiting.



PIRQ bits <15:9> are read/write; bits <7:5,3:1> are read only; the remaining bits always read as zeroes. PIRQ is cleared at power up, by a console start, and by a RESET instruction.

3.4 CPU Error Register (17777766)

This register identifies the source of any abort or trap that caused a trap through location 4.



BIT	NAME	FUNCTION
7	Illegal HALT (RO)	Set when execution of a HALT instruction is attempted in user or supervisor mode.
6	Address Error (RO)	Set when word access to an odd byte address or an instruction fetch from an internal register is attempted.
5	Non-existent Memory (RO)	Set when a reference to main memory times out.
4	I/O Bus Timeout (RO)	Set when a reference to the I/O page times out.
3	Yellow Stack Trap (RO)	Set on a yellow stack trap.
2	Red Stack Trap (RO)	Set on a red stack trap.

The CPU Error Register is cleared by any write reference. It is also cleared at power up or by console start. It is unaffected by a RESET instruction.

NOTE: This register is identical to the 11/70. The 11/44 includes several additional transient status bits. Note that the definition of address trap has been expanded to include instruction fetches from an internal register, and that the definition of stack trap has also been changed.

3.5 Stack Protection

The J-11 checks kernel stack references against a fixed limit of 400(8). If the virtual address of a kernel stack reference is less than 400(8), a yellow stack trap occurs at the end of the current instruction (except for CIS instructions, which abort at the start of instruction execution). A stack trap can occur only on a kernel stack reference, which is defined as a kernel mode 4 or 5 reference through R6, a CIS stack push, or a JSR, trap, or interrupt stack push.

In addition, the J-11 checks for kernel stack aborts during interrupt, trap, or abort sequences. If a kernel stack push during an interrupt, trap, or abort causes an abort, the J-11 initiates a red zone stack trap by creating an emergency stack at locations 2 and 0, setting bit <2> of the CPU Error Register, and vectoring through location 4.

NOTE: The J-11 treatment of yellow stack trap is identical to the 11/44. The 11/70 includes a stack limit register, and a more inclusive definition of a stack reference. The J-11's definition of a red stack trap is unique.

3.6 Kernel Protection

In order to protect the kernel operating system against interference, the J-11 incorporates a number of protection mechanisms:

- In kernel mode, HALT, RESET, and SPL execute as specified. In supervisor or user mode, HALT causes a trap through location 4, while RESET and SPL are treated as NOPs.

- In kernel mode, RTI and RTT can alter PS <15:11> and PS <7:5> freely. In supervisor or user mode, RTI and RTT can only set PS <15:11> and cannot alter PS <7:5>.

- In kernel mode, MTPS can alter PS <7:5>. In supervisor or user mode, MTPS cannot alter PS <7:5>.

- All trap and interrupt vector references are classified as kernel data space references, irrespective of the memory management mode at the time of the trap or interrupt.

- Kernel stack references are checked for stack overflow. Supervisor and user stack references are not checked.

These protection mechanisms are fully compatible with the 11/44 and 11/70.

4.0 MEMORY MANAGEMENT

The J-11 implements 11/44-11/70 compatible memory management. This features:

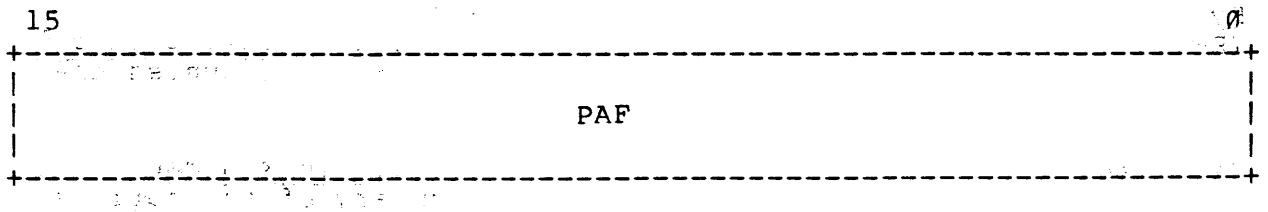
- 22 bit physical address translation.
- Instruction and data (I/D) address spaces.
- Kernel, supervisor, and user (K/S/U) processor modes.

NOTE: No I/O map is supplied with the J-11 chip set. It is coupled with the UNIBUS adapter module, if any.

The visible memory management state consists of 48 Page Address Registers (PARs), 48 Page Descriptor Registers (PDRs), and four Memory Management Registers (MMR0-3).

4.1 Page Address Registers - PARs

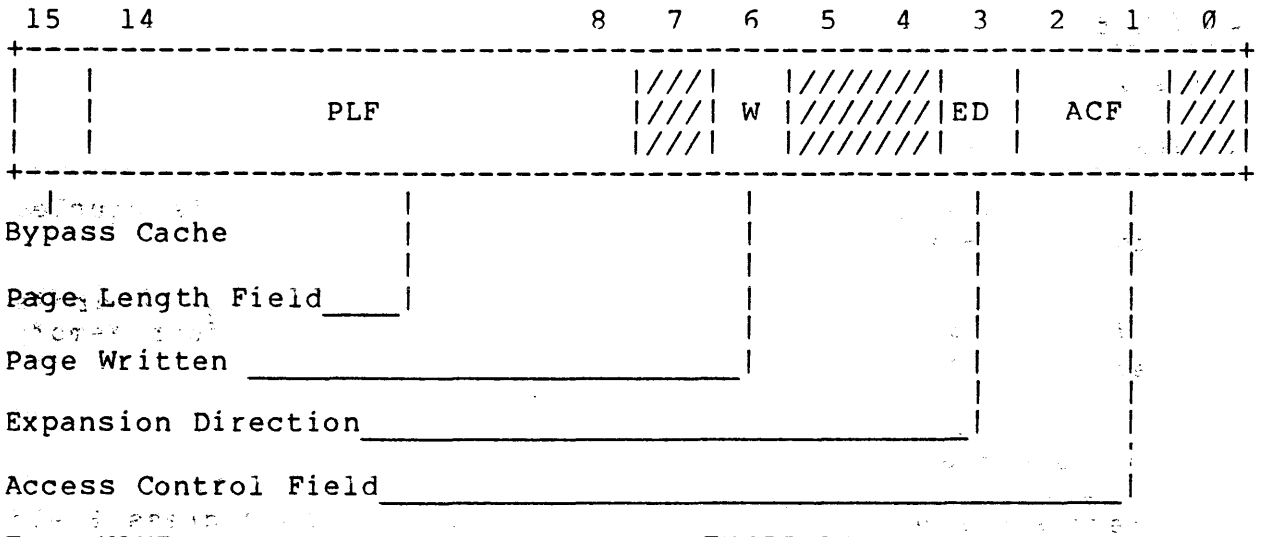
The Page Address Registers (PARs) contain the 16-bit Page Address Field (PAF).



All bits are read/write. These registers are not affected by console start or a RESET instruction. Their state at power up is UNDEFINED.

4.2 Page Descriptor Registers - PDRs

The Page Descriptor Registers (PDRs) contain information relative to page expansion, page length, and access control.



BIT	NAME	FUNCTION
15	Bypass Cache (RW)	This bit implements a conditional cache bypass mechanism. If set, references to the selected virtual page will bypass the cache.
14:8	Page Length Field (RW)	This field specifies the block number which defines the boundary of the current page. The block number of the virtual address is compared against the Page Length Field to detect length errors. An error occurs when expanding upwards if the block number is greater than the Page Length Field, and when expanding downwards if the block number is less than the Page Length Field.
6	Page Written (RO)	This bit indicates whether or not this page has been modified (i.e. written into) since either the PAR or PDR was loaded (1 is affirmative). It is useful in applications which involve disk swapping and memory overlays. It is used to determine which pages have been modified and hence must be saved in their new form and which pages have not been modified and can simply be overlaid. This bit is reset to Ø whenever either the PDR or the associated PAR is written into.

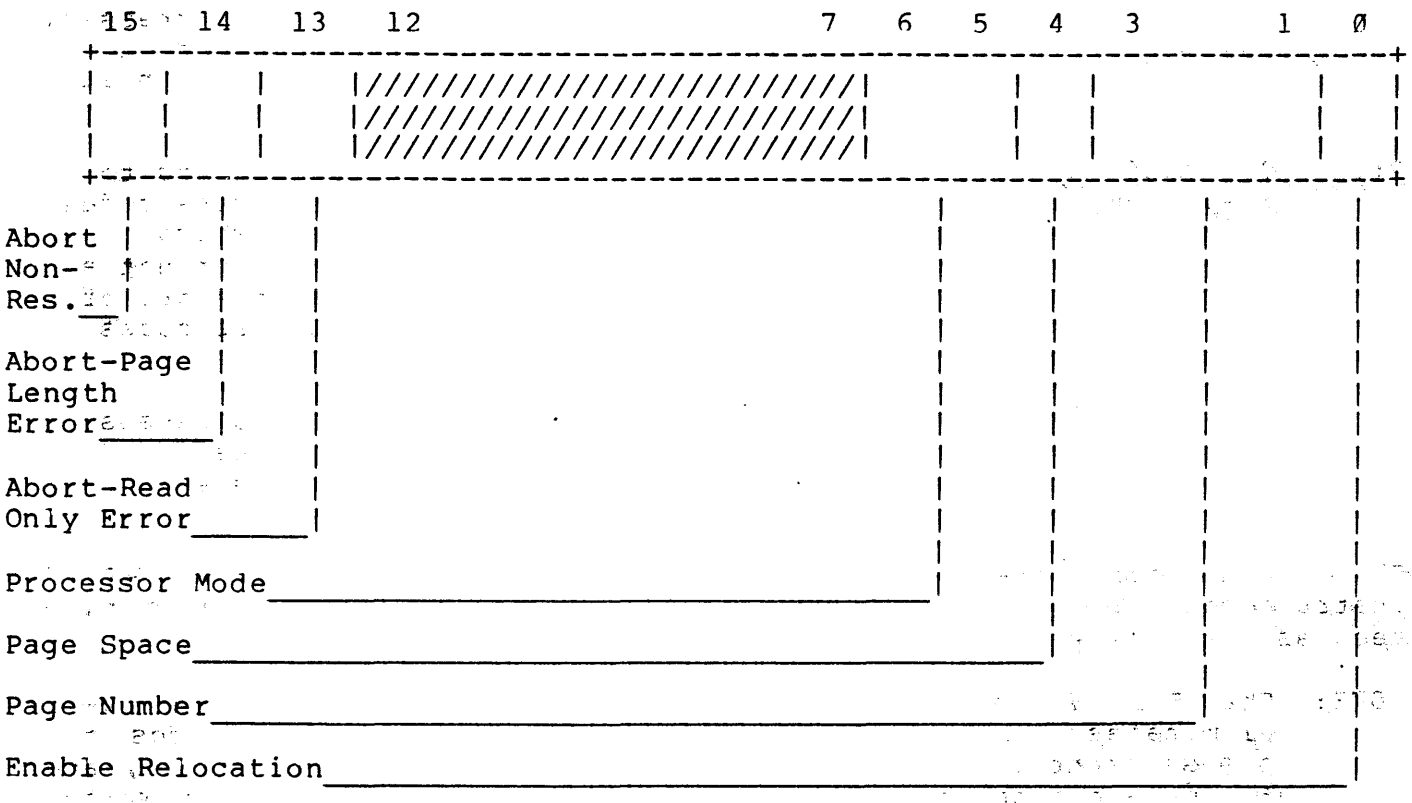
3	Expansion Direction (RW)	This bit specifies in which direction the page expands. If ED=Ø the page expands upwards from block number Ø to include blocks with higher addresses; if ED=1, the page expands downwards from block number 127 to include blocks with lower addresses. Upward expansion is usually used for program space while downward expansion is used for stack space.
2:1	Access Control Field (RW)	This field contains the access rights to to this particular page. The access codes or "keys" specify the manner in which a page may be accessed and whether or not a given access should result in an abort of the current operation. The access codes are: ØØ Non-resident - abort all accesses Ø1 Read only - abort on writes 1Ø Not used - abort all accesses 11 Read/write

These registers are not affected by console start or a RESET instruction. Their state at power up is UNDEFINED. All unused bits read as zero and cannot be written.

NOTE: The J-11 PDR's are identical to the 11/44 PDR's. The J-11 eliminates the 11/7Ø's "A" (any access) status bit, adds the bypass cache bit, and only supports 11/7Ø access modes Ø, 2, and 5. In addition, the J-11 sets the W (page written) bit on writes which cause aborts or modify internal registers, while the 11/44 and 11/7Ø do not.

4.3 Memory Management Register Ø - MMRØ (17777572)

MMRØ contains error flags, the page number whose reference caused the abort, and various other status flags.



BIT	NAME	FUNCTION
15	Abort - Non Resident (RW)	Bit 15 is set by attempting to access a page with an Access Control Field key equal to Ø or 2. It is also set by attempting to use memory relocation with a processor mode (PS<15:14>) of 2.
14	Abort - Page Length (RW)	Bit 14 is set by attempting to access a location in a page with a block number (virtual address bits <12:6>) that is outside the area authorized by the Page Length Field of the Page Descriptor Register for that page.
13	Abort - Read Only (RW)	Bit 13 is set by attempting to write in a "Read Only" page. "Read Only" pages have access keys of 1.

Note that bits <15:13> can be set by an explicit write; however such an action does not cause an abort. Whether set explicitly or by an abort, bits <15:13> cause memory management to freeze the contents of MMRØ<15:13,6:1>, MMR1, and MMR2. The status registers remain frozen until MMRØ<15:13> are cleared by an explicit write or any initialization sequence.

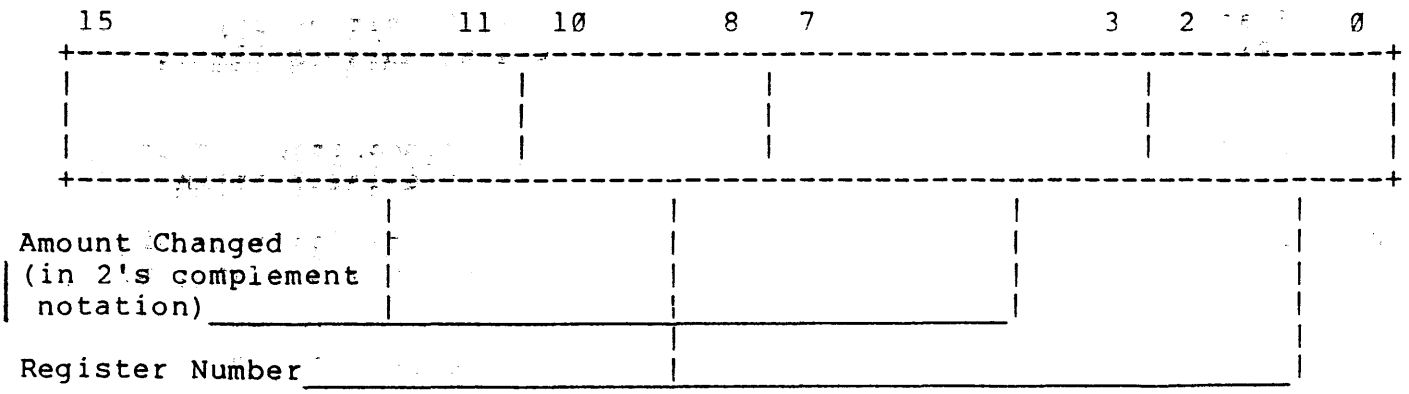
6:5	Processor Mode (RO)	Bits <6:5> indicate the processor mode (kernel/supervisor/user/illegal) associated with the page causing the abort (kernel = ØØ, supervisor = Ø1, user = 11, illegal = 1Ø). If the illegal mode is specified, an abort is generated and bit <15> is set.
4	Page Space (RO)	Bit 4 indicates the address space (I or D) associated with the page causing the abort (Ø = I space, 1 = D space).
3:1	Page Number (RO)	Bits <3:1> contain the page number of the page causing the abort.
Ø	Enable Relocation (RW)	Bit Ø enables relocation. When it is set to 1, all addresses are relocated. When bit Ø is set to Ø, memory management is inoperative and addresses are not relocated.

MMRØ<15:13,Ø> is cleared at power up, by a console start, and by a RESET instruction. MMRØ<6:1> is UNDEFINED at power up.

NOTE: The J-11 eliminates the 11/44-11/7Ø maintenance mode feature, and the 11/7Ø memory management trap and instruction complete features. The J-11 and 11/44 update MMRØ<6:1> on references to internal processor registers; the 11/7Ø does not. The 11/44 sets only MMRØ<15> on an abort due to the illegal processor mode; the 11/7Ø sets MMRØ<15:14>; the J-11 sets MMRØ<15>, but the state of MMRØ<14:13> is unpredictable.

4.4 Memory Management Register 1 - MMR1 (177777574)

MMR1 records any auto increment or decrement of the general purpose registers. This register supplies necessary information needed to recover from a memory management abort.



MMR1 is read only. Its state at power up is UNDEFINED.

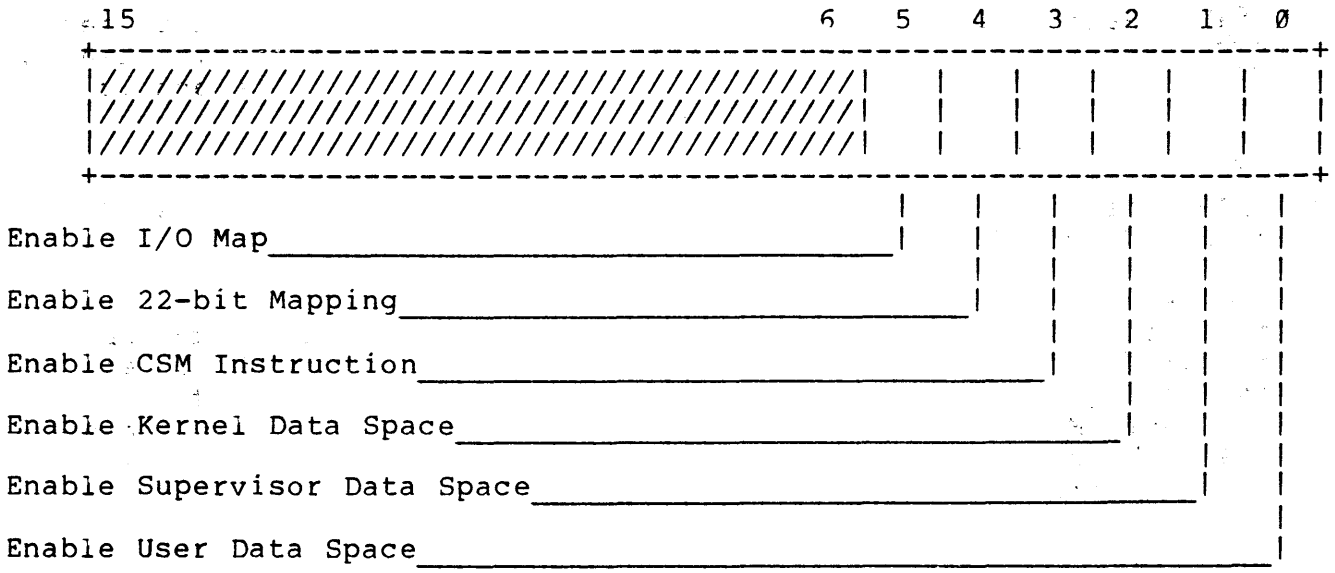
4.5 Memory Management Register 2 - MMR2 (17777576)

MMR2 is loaded with the virtual address at the beginning of each instruction fetch. MMR2 is read only. Its state at power up is UNDEFINED.

NOTE: The 11/70 also loads MMR2 with the vector during an interrupt or trap.

4.6 Memory Management Register 3 - MMR3 (17772516)

MMR3 enables or disables D space, 22-bit mapping, the CSM instruction, and the I/O map (when applicable).



BIT	NAME	FUNCTION
5	Enable I/O Map (RW)	This bit enables the I/O map on an external UNIBUS adapter, if any.
4	Enable 22-bit Mapping (RW)	This bit enables 22-bit memory addressing (the default is 18-bit addressing).
3	Enable CSM Instruction (RW)	This bit enables recognition of the Call Supervisor Mode instruction.
2:0	Enable Data Space (RW)	These bits enable Data Space mapping for kernel, supervisor, and user mode, respectively.

MMR3 is cleared at power up, by a console start, and by a RESET instruction.

NOTE: No I/O map is supplied with the J-11. It is coupled with the UNIBUS adapter module, if any.

4.7 I and D Space

When the data space feature is enabled, the J-11 classifies memory references into instruction (I) and data (D) space references and uses the corresponding mapping registers. In general, the following are classified as I space references:

- instruction fetches
- immediate operands (mode 27)
- absolute addresses (mode 37)
- index words
- inline operands (CIS instructions)
- first references in modes 17, 47, and 57

and all other references are classified as D space. However, MTPI, MFPI, MTPD, and MFPD behave differently than normal instructions. In particular, MFPI (if PS<15:12> = 1111), MTPD, and MFPD always force the last memory reference to D space; while MFPI (if PS<15:12> ≠ 1111) and MTPI always force the last memory reference to I space. Table 4-1 provides an exact description of the interaction of I and D space with the addressing modes.

Table 4-1
I AND D SPACE OPERATION
(first/second/third memory references)

Address Mode and Reg Select	Normal Instruction	MTPI, MFPI (PS<15:12> ≠ 1111)	MTPD,MFPD, MFPI (PS<15:12> = 1111)
00 - 07	na	na	na
10 - 16	D	I	D
17	I	I	D
20 - 26	D	I	D
27	I	I	D
30 - 36	D/D	D/I	D/D
37	I/D	I/I	I/D
40 - 46	D	I	D
47	I	I	D
50 - 56	D/D	D/I	D/D
57	I/D	I/I	I/D
60 - 67	I/D	I/I	I/D
70 - 77	I/D/D	I/D/I	I/D/D

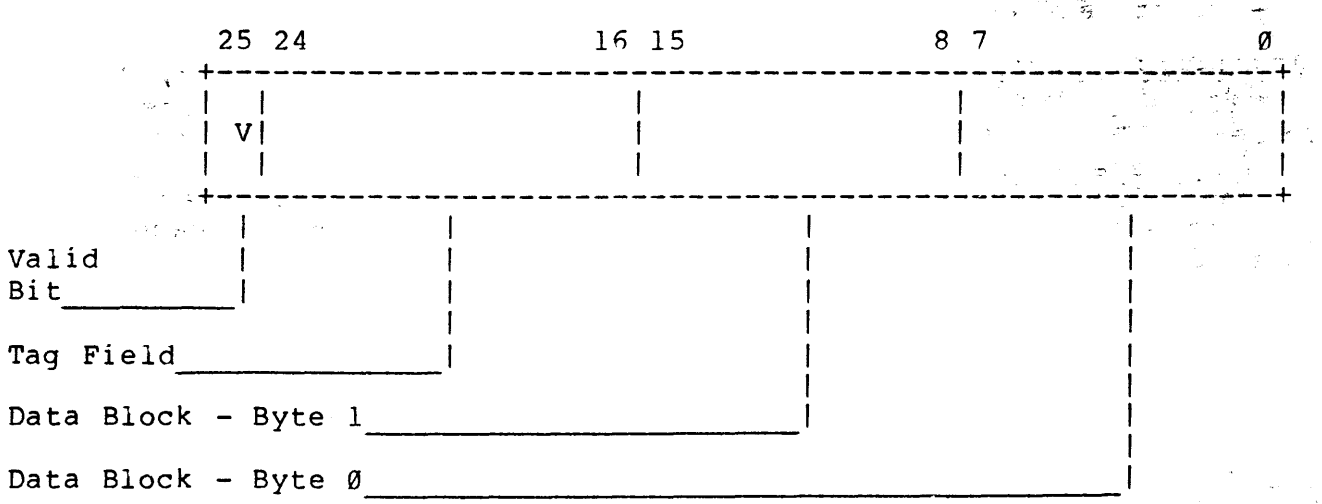
5.0 MEMORY SYSTEM

The following highlights the J-11 memory system:

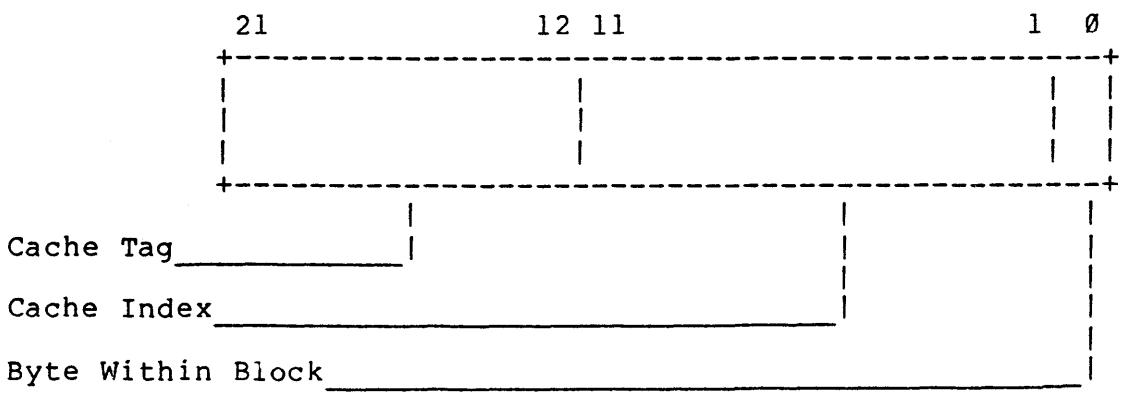
- It can contain a cache.
- It incorporates an instruction stream buffer which implements a prefetch/predecode scheme.

5.1 Cache

The J-11 supports a physical cache subsystem. Many different cache organizations are possible. The example used here is an 8 KB direct map cache with a block size of two bytes. The organization of each cache entry (exclusive of parity or other protection mechanism) is:

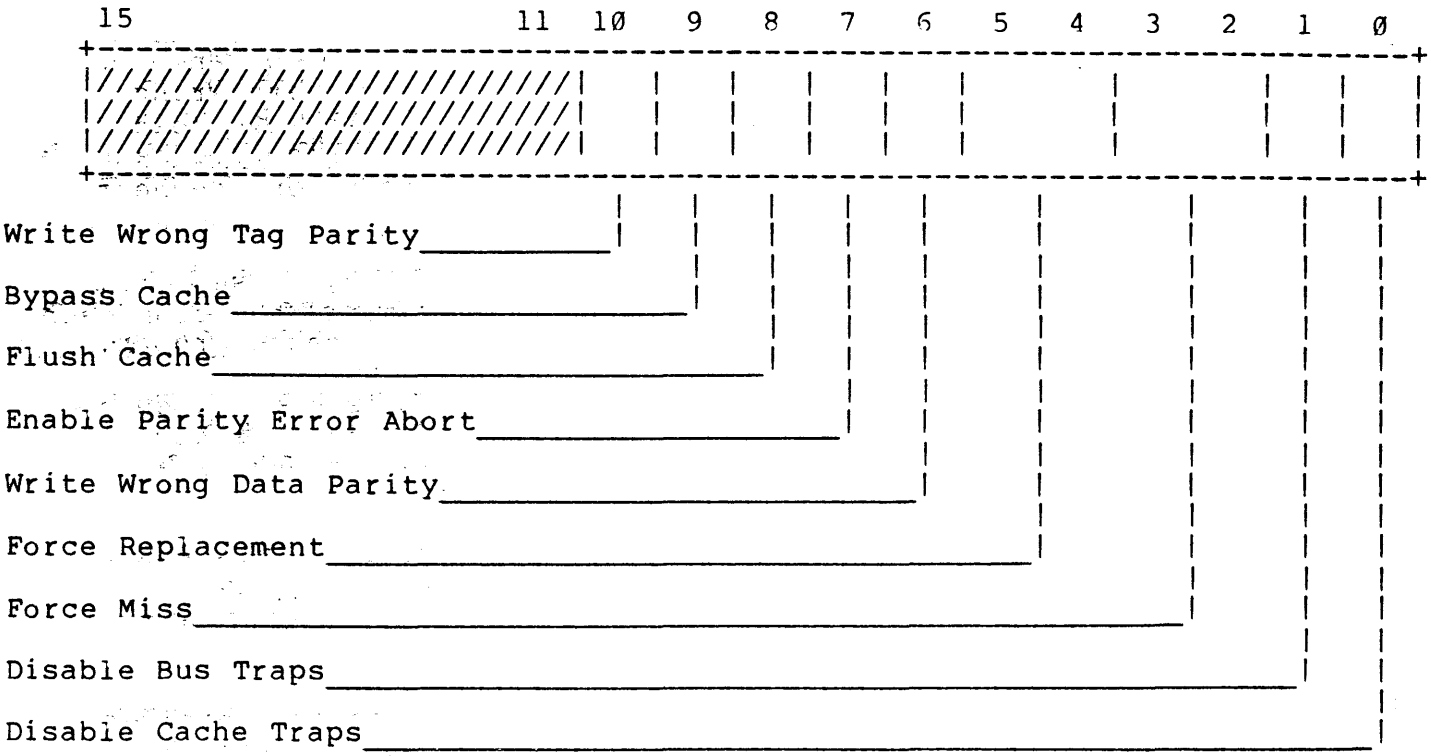


The physical address is logically subdivided as follows:



5.1.1 Cache Control Register (17777746)

The Cache Control Register controls the operation of the cache. Of its features, only bypass and force miss are architecturally part of the J-11 chip set. Tag parity, data parity, and cache flush, if implemented, are the responsibility of the control logic around the chip set.



BIT	NAME	FUNCTION
10	Write Wrong Tag Parity (RW)	This bit, when set, causes the cache tags to be written with wrong parity on all update cycles. This will cause a cache tag parity error to occur on the next access to that location.
9	Bypass Cache (RW)	This bit, when set, forces all CPU memory references to go directly to main memory. Read or write hits will result in invalidation of accessed locations in the cache.
8	Flush Cache (WO)	Setting this bit causes the entire contents of the cache to be declared invalid. Writing a "0" into this bit will have no effect.

7	Enable Parity Error Abort (RW)	This bit controls the response of the to a parity error. When set a cache parity error will cause a force miss and an abort to occur. When clear this bit inhibits the abort and enables an interrupt to parity error vector 114. All cache parity errors result in force misses.
6	Write Wrong Data Parity (RW)	This bit, when set, causes high and low parity bytes to be written with wrong parity on all update cycles (CPU read misses and write hits). This will cause a cache parity error to occur on the next access to that location.
5:4	Force Replacement (RW)	In a set associative cache, these bits, when set, force data replacement from main memory within one or both cache groups.
3:2	Force Miss (RW)	These bits, when either is set, force all CPU memory references to go directly to main memory. The cache tag and data stores are not changed.
1	Disable Bus Traps (RW)	In a system with separate I/O and memory busses, this bit, when set, disables recognition of parity errors on the I/O bus.
Ø	Disable Cache Traps (RW)	This bit disables cache parity interrupts. When set, no interrupt to location 114 will occur when a parity error is encountered.

If the control logic around the chip set implements cache data parity, then words read from the cache will be checked for parity. A parity error in the accessed word causes the following CPU responses:

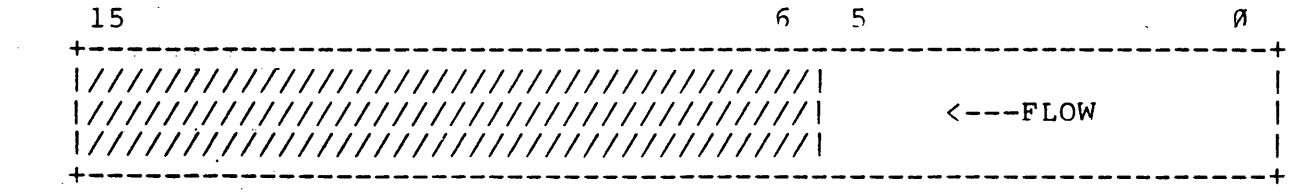
Bit 7	Bit Ø	Action
Ø	Ø	Interrupt to 114 and force miss.
Ø	1	Force miss only.
1	X	Abort and force miss.

This register is cleared on power up or by a console start. It is unaffected by a RESET instruction.

NOTE: The organization of this register reflects the operating system groups' requests. It differs in some details from both the 11/44 and the 11/70.

5.1.2 Hit/Miss Register (17777752)

This register indicates whether the six most recent CPU memory references resulted in cache hits or cache misses:



Bits enter from the right (at bit <Ø>) and are shifted leftward. A one indicates a cache hit, a zero indicates a cache miss.

The Hit/Miss Register is read only. Its value at power up is UNDEFINED. The Hit/Miss Register is not affected by console start or a RESET instruction.

NOTE: The Hit/Miss Register is compatible with both the 11/44 and the 11/70.

5.1.3 Cache Multi-Processor Hooks

The following multi-processor cache "hooks" exist in the J-11:

- Conditional cache bypass - selected virtual pages can be made to bypass the cache. Bit <15> in the PDRs sets this condition.

- Unconditional cache bypass - all CPU references can be made to bypass the cache. Bit <9> in the Cache Control Register sets this condition.

- Flush cache - all valid bits in the cache are cleared.

- Lock instructions (ASRB, TSTSET, WRTLCK) - these instructions guarantee a cache bypass reference.

5.1.4 Cache Response Matrix

The cache response matrix is:

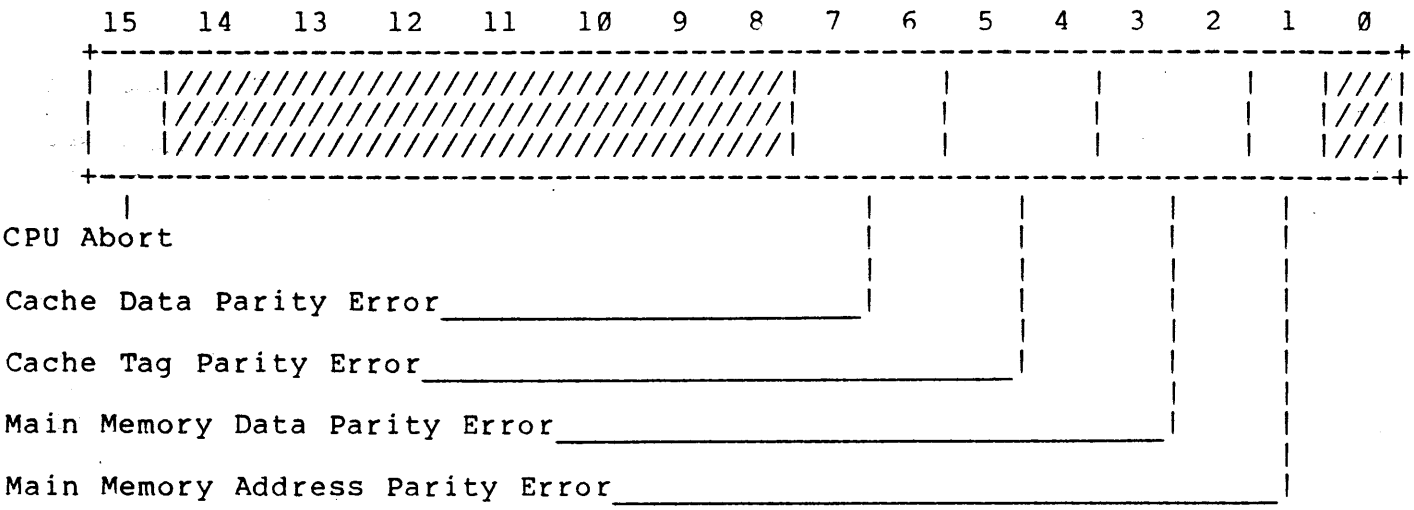
	CPU		DMA	
	Hit	Miss	Hit	Miss
Read	Read cached data	Read memory & allocate cache	Read memory	Read memory
Write	Write thru cache to memory	Write memory	Invalidate cache & write mem	Write memory
Read bypass	Invalidate cache & read mem	Read memory	na	na
Write bypass	Invalidate cache & write mem	Write memory	na	na
Read forced miss	Read memory	Read memory	na	na
Write forced miss	Write memory	Write memory	na	na

5.2 I-Stream Buffer

The J-11 gets much of its performance from a prefetching mechanism. Basically, sequential instruction stream words are prefetched under microcode control. The J-11 Data Chip Specification details the precise prefetch mechanism.

5.3 Memory System Error Register (17777744)

The Memory System Error register details the memory system failure mode. The Memory System Error Register is not part of the J-11 chip set; its implementation is the responsibility of the control logic surrounding the chip set. If implemented, it would have the following format, or a subset thereof:



BIT	NAME	FUNCTION
15	CPU Abort (RO)	Set if any of bits <3:1> are set, or if any of bits <7:4> are set and Cache Control Register bit <7> is also set.
7:6	Cache Data Parity Error (RO)	One or both set if there is a cache data parity error.
5:4	Cache Tag Parity Error (RO)	One or both set if there is a cache tag parity error.
3:2	Main Memory Data Parity Error (RO)	One or both set if there is a main memory data parity error.
1	Main Memory Address Parity Error (RO)	Set if there is a main memory address parity error.

This register is cleared by any write reference. It is also cleared on power up or by console start. It is unaffected by a RESET instruction.

NOTE: Due to hardware dependencies, this register differs from the 11/44 Cache Memory Error Register and the 11/70 Memory System Error Register.

6.Ø FLOATING POINT INSTRUCTIONS

The floating point instruction set (FP-11) in the J-11 is completely software compatible with the FP11-A used on the PDP-11/34, the FP11-F on the PDP-11/44, the FP11-E on the PDP-11/6Ø, and the FP11-C on the PDP-11/7Ø. Both single and double precision floating point capability are available together with other features including floating-to-integer and integer-to-floating conversion.

The floating point instruction set is implemented either in microcode residing in the base Control chip, or in a separate coprocessor. The coprocessor acts as a floating point accelerator (FPA) and provides approximately five times the performance of the microcode implementation.

6.1 Floating Point Status Register - FPS

This register provides mode and interrupt control for floating point instructions and records conditions resulting from the execution of the previous instruction. Three bits of the FPS register control the modes of operation:

Single/Double: Floating point numbers can be either single or double precision.

Long/Short: Integer numbers can be 16 bits or 32 bits.

Chop/Round: The result of a floating point operation can be either chopped or rounded. The term "chop" is used instead of "truncate" in order to avoid confusion with truncation of series used in approximations for function subroutines.

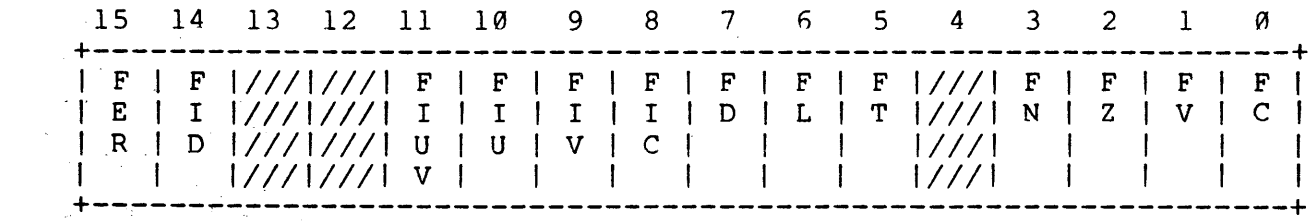
The FPS register contains an error flag and four condition codes (5 bits): carry, overflow, zero, and negative, which are analogous to the processor status condition codes.

The FP-11 recognizes six floating point exceptions:

Detection of the presence of the undefined variable in memory
Floating overflow
Floating underflow
Failure of floating to integer conversion
Attempt to divide by Ø
Illegal floating opcode.

For the first four of these exceptions, bits in the FPS register are available to individually enable and disable interrupts. An interrupt on the occurrence of either of the last two exceptions can be disabled only by setting a bit which disables interrupts on all six of the exceptions, as a group.

Of the thirteen FPS bits, five are set by the FP-11 as part of the output of a floating point instruction: the error flag and condition codes. Any of the mode and interrupt control bits may be set by the user; the LDFPS instruction is available for this purpose. The FPS register is formatted as follows:



BIT	NAME	FUNCTION
15	Floating Error (FER)	The FER bit is set by the FP-11 if 1. Division by zero occurs 2. Illegal opcode occurs 3. Any one of the remaining occurs and the corresponding interrupt is enabled. Note that the above action is independent of whether the FID bit is set or clear. Note also that the FP-11 never resets the FER bit. Once the FER bit is set by the FP-11, it can be cleared only by an LDFPS instruction (note the RESET instruction does not clear the FER bit). This means that the FER bit is up to date only if the most recent floating point instruction produced a floating point exception.
14	Interrupt Disable (FID)	If the FID bit is set, all floating point interrupts are disabled. NOTES 1. The FID bit is primarily a maintenance feature. It should normally be clear. In particular, it must be clear if one wishes to assure that storage of -0 by the FP-11 is always accompanied by an interrupt. 2. Throughout the rest of this chapter, it is assumed that the FID bit is clear in all discussions involving overflow, underflow, occurrence of -0, and integer conversion errors.

13		Reserved for future use.
12		Reserved for future use.
11	Interrupt on Undefined Variable (FIUV)	An interrupt occurs if FIUV is set and a -Ø is obtained from memory as an operand of ADD, SUB, MUL, DIV, CMP, MOD, NEG, ABS, TST, or any LOAD instruction. The interrupt occurs before execution on all instructions. When FIUV is reset, -Ø can be loaded and used in any FP-11 operation. Note that the interrupt is not activated by the presence of -Ø in an AC operand of an arithmetic instruction; in particular, trap on -Ø never occurs in mode Ø. A result of -Ø will not be stored without the simultaneous occurrence of an interrupt.
1Ø	Interrupt on Underflow (FIU)	When the FIU bit is set, floating underflow will cause an interrupt. The fractional part of the result of the operation causing the interrupt will be correct. The biased exponent will be too large by 400_8, except for the special case of Ø, which is correct. An exception is discussed later in the detailed description of the LDEXP instruction. If the FIU bit is reset and if underflow occurs, no interrupt occurs and the result is set to exact Ø.
9	Interrupt on Overflow (FIV)	When the FIV bit is set, floating overflow will cause an interrupt. The fractional part of the result of the operation causing the overflow will be correct. The biased exponent will be too small by 400_8. If the FIV is reset and overflow occurs, there is no interrupt. The FP-11 returns exact Ø.

8	Interrupt on Integer Conversion Error (FIC)	When the FIC bit is set and a conversion to integer instruction fails, an interrupt will occur. If the interrupt occurs, the destination is set to Ø, and all other registers are left untouched. If the FIC bit is reset, the result of the operation will be the same as detailed above, but no interrupt will occur. The conversion instruction fails if it generates an integer with more bits than can fit in the short or long integer word specified by the FL bit (bit 6).
7	Floating Double Precision Mode (FD)	The FD bit determines the precision that is used for floating point calculations. When set, double precision is assumed; when reset, single precision is used.
6	Floating Long Integer Mode (FL)	The FL bit is active in conversion between integer and floating point format. When set, the integer format assumed is double precision 2's complement (i.e., 32 bits). When reset, the integer format is assumed to be single precision 2's complement (i.e., 16 bits).
5	Floating Chop Mode (FT)	When the FT bit is set, the result of any arithmetic operation is chopped (or truncated). When reset, the result is rounded.
4		Reserved for future use.
3	Floating Negative operation was	FN is set if the result of the last (FN) negative, otherwise it is reset.
2	Floating Zero (FZ)	FZ is set if the result of the last operation was Ø, otherwise it is reset.
1	Floating Overflow (FV)	FV is set if the last operation resulted in an exponent overflow, otherwise it is reset.
Ø	Floating Carry (FC)	FC is set if the last operation resulted in a carry of the most significant bit. This can only occur in floating or double to integer conversions.

6.2 Floating Exception Code and Address Registers - FEC, FEA

One interrupt vector is assigned to take care of all floating point exceptions (location 244_8). The six possible errors are coded in the 4-bit floating exception code (FEC) register as follows:

2 Floating opcode error
4 Floating divide by 0
6 Floating to integer conversion error
8. Floating overflow
10. Floating underflow
12. Floating undefined variable.

The address of the instruction producing the exception is stored in the floating exception address (FEA) register.

The FEC and FEA registers are updated only when one of the following occurs:

1. Divide by 0

2. Illegal opcode

3. Any of the other four exceptions with the corresponding interrupt enabled.

This implies that when and only when the FER bit is set by the FP-11 are the FEC and FEA registers updated.

NOTES

1. If one of the last four exceptions occurs with the corresponding interrupt disabled, the FEC and FEA are not updated.

2. Inhibition of interrupts by the FID bit does not inhibit updating of the FEC and FEA, if an exception occurs.

3. The FEC and FEA do not get updated if no exception occurs. This means that the STST (store status) instruction will return current information only if the most recent floating point instruction produced an exception.

4. Unlike the FPS register, no instructions are provided for storage into the FEC and FEA registers.

6.3 Accuracy

General comments on the accuracy of the FP-11 are presented here. An instruction or operation is regarded as "exact" if the result is identical to an infinite precision calculation involving the same operands. The a priori accuracy of the operands is thus ignored. All arithmetic instructions treat an operand whose biased exponent is Ø as an exact Ø (unless FIUV is enabled and the operand is -Ø, in which case an interrupt occurs). For all arithmetic operations, except DIV, a Ø operand implies that the instruction is exact. The same statement holds for DIV if the Ø operand is the dividend. But if it is the divisor, division is undefined, and an interrupt occurs.

For nonvanishing floating point operands, the fractional part is binary normalized. It contains 24 bits or 56 bits for floating mode and double mode, respectively. For ADD, SUB, MUL, and DIV, two guard bits are necessary and sufficient for the general case to guarantee return of a chopped or rounded result identical to the corresponding infinite precision operation chopped or rounded to the specified word length. Thus, with two guard bits, a chopped result has an error bound of one least significant bit (LSB); a rounded result has an error bound of 1/2 LSB. These error bounds are realized by the J-11 on all instructions. Both the FP11-A and the FP11-E have an error bound greater than 1/2 LSB for ADD and SUB.

In the rest of this specification, an arithmetic result is called exact if no nonvanishing bits would be lost by chopping. The first bit lost in chopping is referred to as the "rounding" bit. The value of a rounded result is related to the chopped result as follows.

1. If the rounding bit is 1, the rounded result is the chopped result incremented by an LSB.

2. If the rounding bit is Ø, the rounded and chopped results are identical.

It follows that:

1. If the result is exact,
   rounded value = chopped value = exact value

2. If the result is not exact, its magnitude

   a. is always decreased by chopping
   b. is decreased by rounding if the rounding bit is Ø
   c. is increased by rounding if the rounding bit is 1.

Occurrence of floating point overflow and underflow is an error condition: the result of the calculation cannot be correctly stored because the exponent is too large to fit into the eight bits reserved for it. However, the internal hardware has produced the correct answer. For the case of underflow, replacement of the correct answer by Ø is a reasonable resolution of the problem for many applications. This is done by the J-11 if the underflow interrupt is disabled. The error incurred by this action is an absolute rather than a relative error; it is bounded (in absolute value) by 2**(-128). There is no such simple resolution for the case of overflow. The action taken, if the overflow interrupt is disabled, is described under FIV (bit 9).

The FIV and FIU bits (of the floating point status word) provide the user with an opportunity to implement his own correction of an overflow or underflow condition. If such a condition occurs and the corresponding interrupt is enabled, the microcode stores the fractional part and the low eight bits of the biased exponent. The interrupt will take place and the user can identify the cause by examination of the FV (floating overflow) bit of the FEC (floating exception) register. The reader can readily verify that (for the standard arithmetic operations ADD, SUB, MUL, and DIV) the biased exponent returned by the instruction bears the following relation to the correct exponent generated by the microcode.

1. On overflow, it is too small by 400_8.

2. On underflow, if the biased exponent is Ø, it is correct. If it is not Ø, it is too large by 400_8.

Thus, with the interrupt enable, enough information is available to determine the correct answer. The user may, for example, rescale his variables (via STEXP and LDEXP) to continue a calculation. Note that the accuracy of the fractional part is unaffected by the occurrence of underflow or overflow.

7.0 TRAPS AND INTERRUPTS

In both traps and interrupts, the currently executing program is interrupted and a new program, the starting address of which is specified by the trap or interrupt vector, is executed. The hardware process for traps and interrupts through a vector V is identical:

```
PS --> temp 1                        !save PS, PC in temporaries
PC --> temp 2
Ø --> PS <15:14>                     !force kernel mode
M[V] --> PC                          !fetch PC from vector, data space
M[V+2] --> PS                        !fetch PS from vector, data space
temp1<15:14> --> PS<13:12>           !set previous mode
SP-2 --> SP                          !selected by new PS
temp1 --> M[SP]                      !push old PS on stack, data space
SP-2 --> SP
temp2 --> M[SP]                      !push old PC on stack, data space
                                     !go execute next instruction
```

Note that if an abort occurs during either the vector fetch or the stack pushes, the PS and PC are restored to their original values prior to recognition of the abort.

The priority order for traps and interrupts is as follows:

address error
memory management violation
timeout/non-existent memory
parity error
trace (T-bit) trap
yellow stack trap
power fail
floating point trap
PIRQ 7
interrupt level 7
PIRQ 6
interrupt level 6
PIRQ 5
interrupt level 5
PIRQ 4
interrupt level 4
PIRQ 3
PIRQ 2
PIRQ 1
Halt line

8.0 GENERAL PERFORMANCE GOALS

The overall performance goals of the J-11 are:

- J-11 base instruction performance equivalent to the 11/70.
- J-11 floating point performance equal to half of the 11/44. With an optional floating point accelerator, the performance will be boosted to 11/70 speeds.
- J-11 CIS performance equal to the 11/44.

9.0 CONSOLE

The J-11 contains console microcode. This will enable a user to access most of the J-11 state, run diagnostics, and monitor the system. The J-11 console replaces the "lights and switches" programmer's console with microcode that interprets ASCII characters to perform equivalent panel functions.

The J-11 console microcode provides the minimum functionality needed to control the chip set. A more elaborate console protocol can be implemented using an external console processor. The console processor would then simulate an external console in order to gain access to the console microcode and the chip set.

Appendix 4 details the operation of the console.

10.0 11/44 HARDWARE DIFFERENCES

The J-11 is designed to replace the 11/44 in existing and future applications; however, it does not contain the following PDP-11/44 hardware features:

- Cache data and maintenance registers (17777750, 17777754)
- Switch register (17777570).

The J-11 does contain additional functionality not present in the 11/44:

- Dual general register set
- SPL, MTPS, MFPS, TSTSET, WRTLCK instructions.

The following list summarizes the hardware differences between the 11/44 and the J-11:

Address	Function	Differences
17 777 776	PS	Added register set select bit<11>
17 777 772	PIRQ	No difference.
17 777 766	CPU Error	Unibus monitoring bits unimplemented.
17 777 754	Cache Data	Unimplemented.
17 777 752	Hit/Miss	No difference.
17 777 750	Maintenance	Unimplemented.
17 777 746	Cache Control	Hardware specific changes (see section 5.1.1).
17 777 744	Memory Error	Hardware specific changes (see section 5.3).
17 777 676 to 17 777 660	User Data PAR	No difference.
17 777 656 to 17 777 640	User Instruction PAR	No difference.
17 777 636 to 17 777 620	User Data PDR	No difference.

17 777 616 to 17 777 600	User Instruction PDR	No difference.
17 777 576	MMR2	No difference.
17 777 574	MMR1	No difference.
17 777 572	MMR0	Eliminated maintenance mode.
17 777 570	Switch Register	Unimplemented.
17 772 516	MMR3	No difference.
17 772 376 to 17 772 360	Kernel Data PAR	No difference.
17 772 356 to 17 772 340	Kernel Instruction PAR	No difference.
17 772 336 to 17 772 320	Kernel Data PDR	No difference.
17 772 316 to 17 772 300	Kernel Instruction PDR	No difference.
17 772 276 to 17 772 260	Supervisor Data PAR	No difference.
17 772 256 to 17 772 240	Supervisor Instruction PAR	No difference.
17 772 236 to 17 772 220	Supervisor Data PDR	No difference.
17 772 216 to 17 772 200	Supervisor Instruction PDR	No difference.

11.0 11/70 HARDWARE DIFFERENCES

The J-11 is designed to replace the PDP-11/70 in existing and future applications; however it does not contain the following PDP-11/70 hardware features:

- Stack Limit Register (17777774)
- Micro Break Register (17777770)
- System ID Register (17777764)
- System Size Registers (17777760, 17777762)
- Maintenance Register (17777750)
- Physical Error Address Registers (17777740, 17777742)
- Switch Register (17777570).

The J-11 does contain additional functionality not present in the 11/70:

- MTPS, MFPS, MFPT, CSM, TSTSET, WRTLCK instructions
- CIS instructions
- Bypass cache bit in PDRs.

The following list summarizes the hardware differences between the 11/70 and the J-11:

Address	Function	Differences
17 777 776	PS	Added suspended instruction bit <8>.
17 777 774	Stack Limit	Unimplemented.
17 777 772	PIRQ	No difference.
17 777 770	Micro Break	Unimplemented.
17 777 766	CPU Error	No difference.
17 777 764	System ID	Unimplemented.
17 777 762	System Size	Unimplemented.
17 777 760	System Size	Unimplemented.
17 777 752	Hit/Miss	No difference.

17 777 750Ø	Maintenance	Unimplemented.
17 777 746	Cache Control	Hardware specific changes (see section 5.1.1).
17 777 744	Memory Error	Hardware specific changes (see section 5.3).
17 777 742	High Error Address	Unimplemented.
17 777 740	Low Error Address	Unimplemented.
17 777 676 to 17 777 660	User Data PAR	No difference.
17 777 656 to 17 777 640	User Instruction PAR	No difference.
17 777 636 to 17 777 620	User Data PDR	Added bypass cache, eliminated access flags and access modes other than Ø, 2, and 6.
17 777 616 to 17 777 600	User Instruction PDR	Added bypass cache, eliminated access flags and access modes other than Ø, 2, and 6.
17 777 576	MMR2	No difference.
17 777 574	MMR1	No difference.
17 777 572	MMRØ	Eliminated traps, maintenance mode, and instruction complete.
17 777 570	Switch Register	Unimplemented.
17 772 516	MMR3	Added CSM enable bit <3>.
17 772 376 to 17 772 360	Kernel Data PAR	No difference.
17 772 356 to 17 772 340	Kernel Instruction PAR	No difference.

17 772 336 to 17 772 320	Kernel Data PDR	Added bypass cache, eliminated access flag and access modes other than 0, 2, and 6.
17 772 316 to 17 772 300	Kernel Instruction PDR	Added bypass cache, eliminated access flag and access modes other than 0, 2, and 6.
17 772 276 to 17 772 260	Supervisor Data PAR	No difference.
17 772 256 to 17 772 240	Supervisor Instruction PAR	No difference.
17 772 236 to 17 772 220	Supervisor Data PDR	Added bypass cache, eliminated access flag and access modes other than 0, 2, and 6.
17 772 216 to 17 772 200	Supervisor Instruction PDR	Added bypass cache, eliminated access flag and access modes other than 0, 2, and 6.

Appendix 1 - J-11 Base Instruction Set

Double Operand Instructions	ADD	BISB	MOV
	ASH	BIT	MOVB
	ASHC	BITB	MUL
	BIC	CMP	SUB
	BICB	CMPB	XOR
	BIS	DIV	
Single Operand Instructions	ADC	DEC	ROR
	ADCB	DECB	RORB
	ASL	INC	SBC
	ASLB	INCB	SBCB
	ASR	MFPS	SWAB
	ASRB	MTPS	SXT
	CLR	NEG	TST
	CLRB	NEGB	TSTB
	COM	ROL	
	COMB	ROLB	
Branch Instructions	BCC/BHIS	BHI	BNE
	BCS/BLO	BLE	BPL
	BEQ	BLOS	BR
	BGE	BLT	BVC
	BGT	BMI	BVS
Jump and Subroutine Instructions	CSM	JSR	RTS
	JMP	MARK	SOB
Trap and Interrupt Instructions	BPT	IOT	RTT
	EMT	RTI	TRAP
Miscellaneous Instructions	HALT	MTPD	TSTSET
	MFPD	MTPI	WAIT
	MFPI	RESET	WRTLCK
	MFPT	SPL	
Condition Code Operators	CCC	CLZ	SEN
	CLC	NOP	SEV
	CLN	SCC	SEZ
	CLV	SEC	

Appendix 2 - J-11 Floating Point Instruction Set

Floating Point Instructions

ABSD	LDCLF	STCDI
ABSF	LDD	STCDL
ADDD	LDEXP	STCFD
ADDF	LDF	STCFI
CFCC	LDFPS	STCFL
CLRD	MODD	STD
CLRF	MODF	STEXP
CMPD	MULD	STF
CMPF	MULF	STFPS
DIVD	NEGD	STST
DIVF	NEGF	SUBD
LDCDF	SETD	SUBF
LDCFD	SETF	TSTD
LDCID	SETI	TSTF
LDCIF	SETL	
LDCLD	STCDF	

Appendix 3 - J-11 Commercial Instruction Set

Character String Instructions	CMPC	MOVC	SCANC
	CMPCI	MOVCI	SCANCI
	LOCC	MOVRC	SKPC
	LOCCI	MOVRCI	SKPCI
	MATC	MOVTC	SPANC
	MATCI	MOVTCI	SPANCI
Numeric String Instructions	ADDN	CMPN	
	ADDNI	CMPNI	
	ASHN	SUBN	
	ASHNI	SUBNI	
Packed String Instructions	ADDP	CMPP	MULP
	ADDPI	CMPPI	MULPI
	ASHP	DIVP	SUBP
	ASHPI	DIVPI	SUBPI
Convert Instructions	CVTLN	CVTNL	CVTPL
	CVTLP	CVTNP	CVTPN
Load Descriptor Instructions	L2DØ	L3DØ	
	L2D1	L3D1	
	L2D2	L3D2	
	L2D3	L3D3	
	L2D4	L3D4	
	L2D5	L3D5	
	L2D6	L3D6	
	L2D7	L3D7	

Appendix 4 - Console Commands

4.1 INTRODUCTION

The console microcode (console ODT) is a portion of the processor microcode that allows the processor to respond to commands and information entered via the terminal. The terminal addresses are 17777560_8 through 17777566_8. They are generated in microcode and cannot be changed. Console ODT is very useful as an aid in running and debugging programs. Communication between the user and processor is via a stream of ASCII characters interpreted by the processor as console commands. These commands are a subset of ODT-11.

4.2 TERMINAL INTERFACE

The minimum hardware requirements for a serial line interface permitting a terminal to communicate with console ODT are contained in the following paragraphs. The intent is to describe the minimum hardware required; this is a subset of the hardware needed to operate system software. For system software/hardware requirements refer to the DLV11 hardware specification.

4.2.1 Receiver Control and Status Register (RCSR)

The RCSR (Figure 4-1) must exist at address 17777560_8 for character input to console ODT. Console ODT does not execute output bus cycles to this address; therefore, the RCSR only needs to respond to input bus cycles. However, system software causes output cycles in order to affect certain bits, such as Interrupt Enable (bit 6), which console ODT does not use.

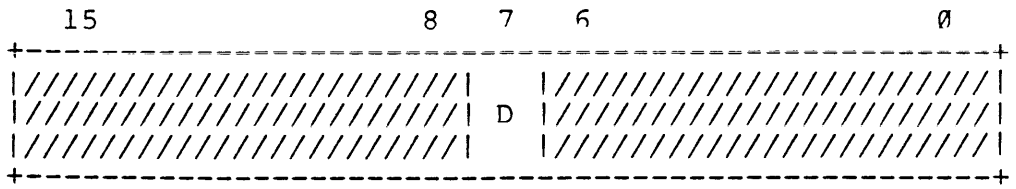


Figure 4-1 Receiver Status Register

Bit	Description
<7>	Done flag. After a character is assembled and exists in the receiver buffer register (RBUF), the Done flag must be set to a 1. When an input cycle is performed to the RBUF (to pick up the character), the Done flag must be cleared by hardware. The system initialization signal must also clear this bit.

Bit	Description
<6:Ø> <15:8>	Unused. These bits are don't cares and can be in any state since console ODT does not use them. In DIGITAL interfaces, these bits may be defined.

4.2.2 Receiver Buffer Register (RBUF)

The RBUF (Figure 4-2) must exist at address 17777562_8 for character input to console ODT. This register only needs to respond to input bus cycles since console ODT does not execute output bus cycles to this address. System software interfaces similarly, but DIGITAL diagnostics may cause an output cycle and not operate properly.

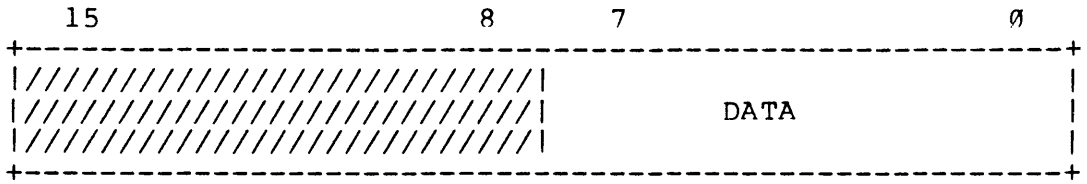


Figure 4-2 Receiver Buffer Register

Bit	Description
<7:Ø>	ASCII character. These eight bits are read by the processor and interpreted as a console ODT command. When bit 7 of RCSR is a 1, the processor does a input cycle to the RBUF. After the input cycle, the hardware must clear bit 7 of RCSR to Ø.
<15:8>	Unused. These bits are don't cares and can be in any state since console ODT does not use them. In DIGITAL interfaces, these bits may be defined.

4.2.3 Transmitter Control and Status Register (XCSR)

The XCSR (Figure 4-3) must exist at address 17777564_8 for character output from console ODT. ODT does not execute output bus cycles to this address; therefore, the XCSR only needs to respond to input bus cycles. However, system software causes output cycles to affect certain bits, such as Interrupt Enable, which console ODT does not use.

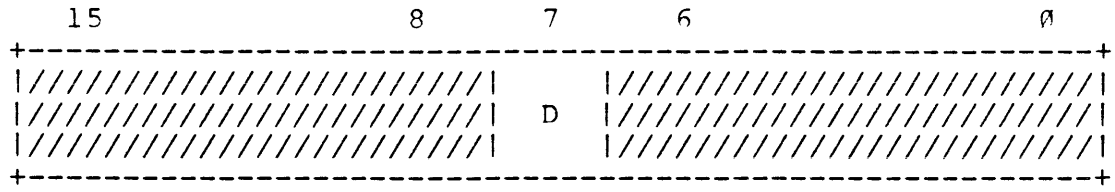


Figure 4-3 Transmitter Control and Status Register

Bit	Description
<7>	Done flag. In the idle state, this bit is a 1 indicating that the hardware is ready to print a character. After an output cycle to the transmitter buffer register by the processor (i.e., a character loaded), this bit must be cleared to 0 by the hardware. After the character is printed, the hardware sets this bit to 1. Power up and the system bus initialization signal must also set this bit to a 1.
<6:0> <15:8>	Unused. These bits are don't cares and can be in any state since console ODT does not use them. In DIGITAL interfaces, these bits may be defined.

4.2.4 Transmitter Buffer Register (XBUF)

The XBUF (Figure 4-4) must exist at address 17777566_8 for character output from console ODT. This register only needs to respond to output bus cycles since console ODT does not execute input bus cycles to this address. System software interfaces similarly but DIGITAL diagnostic may cause an input cycle and not operate properly.

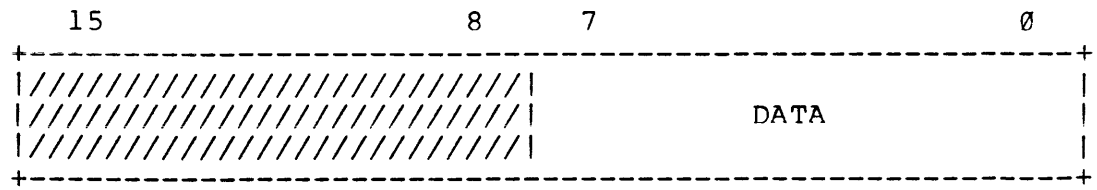


Figure 4-4 Transmitter Buffer Register

Bit	Description
<7:0>	ASCII character. These eight bits are written by the processor with the ASCII character to be printed. When bit 7 of XCSR is a 1, the processor does an output cycle to the XBUF. After the output cycle the hardware must clear bit 7 of XCSR to 0.
<15:8>	Unused. These bits are don't cares and can be in any state since console ODT does not use them. In DIGITAL interfaces, these bits may be defined.

4.3 CONSOLE ODT OPERATION

The processor's microcode operates the serial line interface in half-duplex mode. Program I/O techniques are used rather than interrupts. When the console ODT microcode is printing characters using the transmit side of the interface, the microcode is not monitoring the receive side for incoming characters. Any characters coming in at this time are lost. The interface may post overrun errors, but the microcode does not check for error bits in the interface. Therefore users should not type ahead to ODT because those characters are not recognized. In addition, if another processor is at the other end of the interface, it must obey half-duplex operation. No input characters should be sent until console ODT has finished outputting.

4.3.1 Console ODT Entry Conditions

ODT is entered under the following conditions:

1. Execution of a HALT instruction in kernel mode.
2. Assertion of the HALT signal on the system bus. The signal must be asserted long enough so that it is seen at the end of a macroinstruction by the service state in the processor.
3. At power up, if the appropriate power up option is selected.

4.3.2 Console ODT Input Sequence

Upon entry to console ODT, the RBUF register is read and the character present in the buffer is ignored. This is done so that erroneous characters or user program characters are not interpreted by console ODT as a command, especially when a program is halted.

The input sequence for console ODT is as follows.

1. Output <CR><LF> to terminal.
2. Output contents of PC (R7) in six digits to terminal.
3. Read and ignore character in RBUF.
4. Output <CR><LF> to terminal.
5. Output the prompt character, @, to terminal.
6. Enter a wait loop for terminal input. The Done flag, bit <7> in RCSR, is tested. If it is Ø, the test continues.
7. If RCSR bit <7> is a 1, then low byte of RBUF is read.

4.3.3 Console ODT Output Sequence

The output sequence for ODT is as follows.

1. Test XCSR bit <7> (Done flag) and if a Ø, continue testing.

2. If XCSR bit <7> is a 1, write character to low byte of XBUF (high byte is ignored by interface).

4.4 CONSOLE ODT COMMAND SET

The console ODT command set, listed in Table 4-1, is described in the following paragraphs. The commands are a subset of ODT-11 and use the same command character. Only specific characters are recognized as valid inputs; other inputs invoke a "?" response.

Table 4-1 Console ODT Commands

Command	Symbol	Use
Slash	/	Prints the contents of a specified location.
Carriage Return	<CR>	Closes an open location.
Line Feed	<LF>	Closes an open location and then opens the next contiguous location.
Internal Register Designator	$ or R	Opens a specific processor register.
Processor Status Word Designator	S	Opens the PS - must follow an $ or R command.
Go	G	Starts program execution.
Proceed	P	Resumes execution of a program.
Binary Dump	Control-Shift-S	Manufacturing use only.

The parity bit (bit <7>) on all input characters is ignored (i.e., not stripped) by console ODT. If the input character is echoed, the state of the parity bit is copied to the output buffer (XBUF). Output characters internally generated (e.g., <CR>) by ODT have the parity bit equal to Ø. All commands are echoed except for ASCII codes in the range $Ø\text{-}17_8$. Where applicable, upper- and lowercase of command characters are recognized.

The word "location," as used in the following sections, refers to a memory location, an I/O device register, an internal processor register, or the processor status word (PS).

NOTE

In the examples the response from the processor is underlined, while the user's entry is not.

4.4.1 / (ASCII 057) Slash

This command is used to open a memory location, I/O device register, internal processor register, or processor status word and must be preceded by other characters which specify a location. In response to /, console ODT prints the contents of the location (i.e., six characters) and then a space (ASCII 40). After printing is complete, console ODT waits for either new data for that location or a valid close command.

Example: @001000/012525<SPACE>

where:

@	= console ODT prompt character.
001000	= octal location desired by the user (leading 0s are not required).
/	= command to open and print contents of location.
012525	= contents of octal location 1000.
<SPACE>	= space character generated by console ODT.

4.4.2 <CR> (ASCII 015) Carriage Return

This command is used to close an open location. If a location's contents are to be changed, the user should precede the <CR> with the new data. If no change is desired, <CR> closes the location without altering its contents.

Example: @R1/004321<SPACE> <CR> <CR><LF>
@

Processor register R1 was opened and no change was desired so the user issued<CR>. In response to the <CR>, console ODT printed <CR><LF>@.

Example: @R1/004321<SPACE> 1234 <CR> <CR><LF>
@

In this case the user desired to change R1, so new data, 1234, was entered before issuing the <CR>. Console ODT deposited the new data in the open location and then printed <CR><LF>@.

Console ODT does not directly echo the <CR> entered by the user but instead prints a <CR>, followed by a <LF>, and @.

4.4.3 <LF> (ASCII Ø12) Line Feed

This command is used to close an open location and then open the next contiguous location. Memory locations and processor registers are incremented by 2 and 1 respectively. If the PS is open when a <LF> is issued, it is closed and a <CR><LF>@ is printed; no new location is opened. If the open location's contents are to be changed, the new data should precede the <LF>. If no data is entered, the location is closed without being altered.

Example: @R2/123456<SPACE> <LF> <CR><LF>
R3/Ø54321<SPACE>

In this case, the user entered <LF> with no data preceding it. In response, console ODT closed R2 and then opened R3. When a user has the last register, R7, open, and issues <LF>, console ODT opens the beginning register, RØ.

Example: @R7/ØØØØØØ<SPACE> <LF> <CR><LF>
RØ/123456<SPACE>

Unlike with most other commands, console ODT does not echo the <LF>. Instead it prints <CR>, then <LF>, so that terminal printers operate properly. In order to make this easier to decode, console ODT does not echo ASCII characters in the range Ø - 17_8.

4.4.4 $ (ASCII Ø44) or R (ASCII 122) Internal Register Designator

Either character when followed by a register number, Ø to 7, or PS designator, S, will open that specific processor register.

The $ character is recognized to be compatible with ODT-11. The R character was introduced for the convenience of one key stroke and because it is representative of what it does.

Example: @$Ø/ØØØ123<SPACE>

or

@R7/ØØØ123<SPACE> <LF>
RØ/Ø54321<SPACE>

If more than one character is typed (digit or S) after the R or $, console ODT uses the last character as the register designator.

4.4.5 S (ASCII 123) Processor Status Word

This designator is for opening the PS (processor status word) and may be employed only after the user has entered an R or $ register designator.

Example: @RS/100377<SPACE> 0 <CR> <CR><LF>

Note the trace bit (bit <4>) of the PS cannot be modified by the user. This is done so that PDP-11 program debug utilities (e.g., ODT-11), which use the T bit for single-stepping, are not accidentally harmed by the user.

If the user issues a <LF> while the PS is open, the PS is closed and ODT prints <CR><LF>@. No new location is opened in this case.

4.4.6 G (ASCII 107) Go

This command is used to start program execution at a location entered immediately before the G. This function is equivalent to the LOAD ADDRESS and START switch sequence on other PDP-11 consoles.

Example: @200G<NULL><NULL>

The console ODT sequence for a G, after echoing the command character, is as follows.

1. Print two nulls (ASCII 0) so the bus initialize that follows does not flush the G character from the double-buffered UART chip in the serial line interface.

2. Load R7 (PC) with the entered data. If no data is entered, 0 is used. (In the above example, R7 is set to 200, and that is where program execution begins.)

3. The PS, MMR0<15:13,0>, MMR3, PIRQ, CPU Error Register, Memory System Error Register, Cache Control Register, and Floating Point Status Register are cleared to zero.

4. The cache, if present, is flushed.

5. The system bus is initialized by the processor.

6. The service state is entered by the processor. If there is anything to be serviced, it is processed. If the bus HALT signal is asserted, the processor reenters the console ODT state. This feature is used to initialize a system without starting a program (R7 is altered).

4.4.7 P (ASCII 120) Proceed

This command is used to resume execution of a program and corresponds to the CONTINUE switch on other PDP-11 consoles. No programmer-visible machine state is altered using this command.

Example: @P

Program execution resumes at the address pointed to by R7. After the P is echoed, the processor immediately enters the state to fetch the next instruction. After the instruction is executed, outstanding interrupts, if any, are serviced. If the HALT bus signal is asserted, it is recognized at the end of the instruction, and the processor enters the console ODT state. Upon entry, the content of the PC (R7) is printed. In this fashion, the user can single-instruction step through a program and obtain a PC "trace" on the terminal.

4.4.8 Control-Shift-S (ASCII 023) Binary Dump

This command is used for manufacturing test purposes and is not a normal user command. It is described here to explain the processor's response if accidentally invoked. It is intended to more efficiently display a portion of memory compared to using the "/" and <LF> commands. The protocol is as follows.

1. After a prompt character, console ODT receives a control-shift-S command and echoes it.

2. The host system at the other end of the serial line must send two 8-bit bytes which console ODT interprets as a starting address. These two bytes are not echoed.

   The first byte specifies starting address <15:08> and the second byte specifies starting address <07:00>. Address bits <21:16> are always forced to be 0; the dump command is restricted to the first 32K words of address space.

3. After the second address byte has been received, console ODT outputs ten bytes to the serial line starting at the address previously specified. When the output is finished, console ODT prints <CR><LF>@.

   If a user accidentally enters this command, it is recommended, in order to exit from the command, that two @ characters (ASCII 100) be entered as a starting address. After the binary dump, an @ prompt character is printed.

4.5 ADDRESS SPECIFICATION

All I/O addresses (17 760 000 to 17 777 777) must be entered by users with all 22 bits specified. For example, if a user desires to open the RCSR of the console serial interface he must enter 17777560, not 177560, or 777560.

4.5.1 General Registers

Accessing the general register sets is accomplished in the following way. Whenever R0-R5 are referenced in console ODT, they access the general register set specified by the PS register set bit (PS<11>). If a program operating in general register set zero (PS<11> = 0) is halted and a general register is opened, register set zero is accessed. Similarily, if a program is operating in register set one, "R0-R5" accesses register set one.

If a specific register set is desired, PS<11> must be set by the user to the appropriate value, and then the "R0"-"R5" commands can be used. If an operating program has been halted, the original value of PS<11> must be restored in order to continue execution.

Example: PS = 000000

@R4/052525<SPACE> <CR> <CR><LF>

R4 in register set zero has been opened.

```
@RS/000000<SPACE> 4000 <CR> <CR><LF>
@R4/177777<SPACE> <CR> <CR><LF>
@RS/004000<SPACE> 0 <CR> <CR><LF>
@P
```

In this case, R4 in register set one was desired. The PS was opened, and PS<11> was set to 1 (register set one). Then R4 was examined and closed. The original value of PS<11> was restored, and then the program was continued using the P command.

4.5.2 Stack Pointers

Accessing kernel, supervisor, and user stack pointer registers is accomplished in the following way. Whenever R6 is referenced in console ODT, it accesses the stack pointer specified by the PS current mode bits (PS<15:14>). If a program operating in kernel mode (PS<15:14> = 00) is halted and R6 is opened, the kernel stack pointer is accessed. Similarly, if a program is operating in supervisor or user mode, "R6" accesses the supervisor or user stack pointers.

If a specific stack pointer is desired, PS<15:14> must be set by the user to the appropriate value and then the "R6" command can be used. If an operating program has been halted, the original value of PS<15:14> must be restored in order to continue execution.

Example: PS = 140000

```
@R6/123456<SPACE> <CR> <CR><LF>
```

The user mode stack pointer has been opened.

```
@RS/140000<SPACE> 0 <CR> <CR><LF>
@R6/123456<SPACE> <CR> <CR><LF>
@RS/000000<SPACE> 140000<CR> <CR><LF>
@P
```

In this case, the kernel mode stack pointer was desired. The PS was opened, and PS<15:14> were set to 00 (kernel mode). Then R6 was examined and closed. The original value of PS<15:14> was restored, and then the program was continued using the P command.

4.5.3 Floating Point Accumulators

The floating point accumulators cannot be accessed from console ODT. Only floating point instructions can access these registers.

4.6 ENTERING OCTAL DIGITS

When the user is specifying an address, console ODT will use the last eight octal digits if more than eight have been entered. When the user is specifying data, console ODT will use the last six octal digits if more than six have been entered. The user need not enter leading 0s for either address or data; console ODT forces 0s as the default. If an odd address is entered, console ODT responds to the error by printing ?<CR><LF>@.

4.7 ODT TIMEOUT

If the user specifies a nonexistent address or causes a parity error, console ODT responds to the error by printing ?<CR><LF>@.

4.8 INVALID CHARACTERS

Console ODT will recognize upper- and lowercase characters as commands. Any character that console ODT does not recognize during a particular sequence is echoed (except for ASCII characters in the range $\emptyset$ - 17_8), and console ODT prints ?<CR><LF>@. Console ODT has several internal states, each of which has its own set of valid input characters. When in a particular state, only commands specific to that state are valid. This is done to lower the probability of a user unintentionally destroying a program by pressing the wrong key.