J-11

DATA CHIP SPECIFICATION

21-17677-00

Rev. 3.03 (July 1, 1982)

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

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may occur in this document.

This specification does not describe any program or product which is currently available from Digital Equipment Corporation. Nor does Digital Equipment Corporation commit to implement this specification in any program or product. Digital Equipment Corporation makes no commitment that this document accurately describes any product it might ever make.

RESERVED ISSUES

The following issues are reserved for the implementation effort and have no impact on contractual status or delivery. When these issues are resolved, the chip specifications will be updated accordingly.

1. Data Chip - EU register file codes. Harris may assign extra register select codes to ensure that all codes actually select a register.

   Freeze point: Data chip logic design review
   Status: Closed December, 1980

2. Data Chip - MMU register file codes. Harris may reassign register select codes to minimize internal logic.

   Freeze point: End June, 1980
   Status: Closed December, 1980

3. Data Chip - Opcodes. Within guidelines to be provided by DEC by end April 1980, Harris may assign currently unassigned opcodes to minimize internal logic.

   Freeze point: End June, 1980
   Status: Closed December, 1980

4. Control Chip - AIO codes. DEC may reassign AIO codes to minimize external logic.

   Freeze point: End April, 1980
   Status: Closed - no change to specification

5. Data Chip - NOP status flags. This is a don't care. NOP must have an opcode of 177 and must not alter the EU register file.

   Freeze point: End June, 1980
   Status: Closed December, 1980

6. Control and Data Chips - AC characteristics. By mutual agreement, Harris and DEC can adjust all non-critical parameters to minimize internal and external logic, within the following limits:

Control Chip AC Constraints

tpd + tps (MPRDC-L valid delay + setup) $\leq$ 75 ns

tseld + tsels (MCSEL-L active delay + setup) $\leq$ 75 ns

tmibs + tmibd (MIB-H valid delay + setup) $\leq$ 75 ns

Interchip AC Constraints

tscs + tsid (MSCTL-L setup + inactive delay) $\leq$ 50 ns

tmibs + tmibd (Data Chip + Control Chip) $\leq$ 75 ns

tpd + tprds (MPRDC-L valid delay + setup) $\leq$ 75 ns

MKPB-L, MSOV/JA-L must be valid by T100 and held to T150

Freeze point: End of pass 3 of Control Chip
Status: Open

7. Data Chip - I/O multiplexor. Within the functional and timing constraints of the Data Chip Specification, Harris can implement the I/O multiplexor as it chooses to minimize layout area and/or logic.

   Freeze point: Data Chip logic design review
   Status: Closed December, 1980

8. Data Chip - UMULS, SMULS, DIVS microinstructions. These microinstructions are shown as implemented for all operand lengths. Should a subset implementation save logic in the Data Chip, only the following lengths need be implemented:

   UMULS - longword
   SMULS - word
   DIVS - word, longword

   In particular, support of byte length operands need not be implemented.

   Freeze point: Data chip logic design review
   Status: Closed - byte length not implemented

9. Data Chip - Crystal input parameters. The crystal input AC parameters and test conditions will be specified when DEC selects an input crystal for the chip set.

   Freeze point: End April, 1980
   Status: Open, extended until December, 1982

10. Control and Data Chips - DC characteristics. The TTL input parameters (V_{IHT} and V_{ILT}) are goals and are subject to change as simulation results become available. Harris and DEC can adjust these parameters by mutual agreement.

    Freeze point: End April, 1980
    Status: Open, extended until December, 1982

11. Data Chip - Clock outputs. Input test conditions are tbs.

    Freeze point: End June, 1980
    Status: Open, extended until December, 1982

12. Data Chip - MDAL turnaround. The test specification is for an isolated chip. In a real system, the Data Chip will be fighting with another driver. By mutual agreement, Harris and DEC can adjust the test specification on MDAL turnaround to more accurately reflect actual operating conditions.

    Freeze point: End April, 1980
    Status: Closed - no change to test specification

REVISION HISTORY

REV	DATE	REASON
3.03	7/1/82	Incorporated spcification changes 1b, 63, 105, 111-112, 114, 118-121, 124-125, 128-130 of Rev 13 of the specification change list dated June 22, 1982.
3.02	2/9/81	Incorporated specification changes 1b, 30, 34, 36-57, 59-62, 64-81, and 83-104 of Rev 9 of the Specification Change list dated February 9, 1981.
3.01	8/4/80	Incorporated specification changes 1, 2, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 23, 28, 31, 32, 33, 35, 40, 41, 42, 43, and 44 of Rev 5 of the Specification Change list dated Aug 4, 1980.
3.00	4/16/80	Incorporated specification number.
2.00	1/80	Major revision. Reorganized to reflect proposed layout. Removed virtual cache structure.
1.01	10/79	Changed microcode format and prefetch mechanism to reflect virtual cache structure. Moved PS to dual ported register file. Defined shifting functions. Further enhanced English explanations. General clean up.
1.0	7/79	Preliminary.

TABLE OF CONTENTS

1.0 INTRODUCTION

1.1 Scope

This document specifies the design of a MOS/LSI chip which is part of a chip set that implements a high performance PDP-11 processor with Memory Management, FP-11 Floating Point, and CIS Commercial Instructions. This specification describes the internal organization of the Data chip. It does not describe the operation of the PDP-11 or other chips in the chip set. For further information,. the applicable documents should be consulted.

1.2 Applicable Documents

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

1.3 Data Chip Description

The Data chip is composed of three major pieces: an Execution Unit (EU), a Memory Management Unit (MMU), and a Prefetch Mechanism. The Execution Unit contains the PDP-11 general purpose registers, eight scratch registers, the ALU, a shifter/swapper and shift register, and necessary word/byte multiplexors. The Memory Management Unit contains the relocation registers, FP-11 floating point accumulators, the relocation logic, and the error status registers. The Prefetch Mechanism contains a virtual program counter (VPC), a relocated version of the PC, a prefetch buffer, and various status flags. The Data chip performs all arithmetic and logic functions, handles all data and address transfers including relocation, and operates most of the signals used for interchip communications and system timing.

The Data chip is designed to work with a cached or cacheless memory system. A cached memory system will yield the highest performance J-11 system; a cacheless memory system will result in the lowest cost J-11 system.

The Data chip receives microinstructions from the Control chips over the bidirectional Microinstruction Bus (MIB-H).

Note: The equations in this specification are meant to be interpreted as logic, irrespective of polarity, and not as levels.

1.4 Internal Busses

The Data chip is organized around the following internal busses:

- The A-bus (A<31:0>) interconnects the A-port of the EU register file, the A-input of the ALU, the shift register (SR), and the I/O multiplexor. Data flows both into and out of the EU register file on the A-bus; hence, the A-bus is bidirectional. Most microinstructions place EU register file data on the A-bus during the first part of the microcycle. This supplies the A-input to the ALU or, for address microinstructions, a virtual address to the I/O multiplexor. The output of the I/O multiplexor is driven onto the A-bus during the second part of the microcycle. This supplies the write data to the EU register file.

- The B-bus (B<31:0>) interconnects the B-port of the EU register file, the decimal adjust logic, the literal generation logic, the sign extension logic, and the B-input of the ALU. The ALU B-input is always the B-bus destination; hence, the B-bus is unidirectional. Most microinstructions place EU register file data on the B-bus during the first part of the microcycle. Decimal adjust microinstructions place a decimal constant on the B-bus; literal microinstructions, a byte wide literal.

- The S-bus (S<31:0>) connects the output of the ALU to the input of the shifter/swapper.

- The F-bus (F<31:0>) connects the output of the shifter/swapper to the input of the I/O multiplexor.

- The MMU data bus (M<21:0>) connects the I/O multiplexor to the MMU. The MMU register file, the adder input latch, the address adjust logic, the special MMU registers, the virtual program counter (VPC), the physical program counter (PPC), the output latch, and other MMU logic functions all connect to the M-bus. The M-bus is used for internal data transfers between the MMU and EU register files, for address relocation operations, and for explicit MMU register access.

- The MMU decode bus (D<7:0>) drives the MMU register file decoder. The D-bus is driven directly from the microinstruction during microcode access, from the explicit address logic during explicit access, and from the microinstruction and the incoming virtual address during relocation access.

- The fast A-bus (FA<15:13>) is a high-speed connection between A<15:13> and D<2:0>. This time-critical path is needed to meet the access budget of the MMU register file during data stream relocation. The bus data is always the high-order three bits of a virtual address.

1.5 Major States

At its maximum operation frequency (20 Mhz), the Data chip can potentially enter a new state every 25 nsec. The major states in a microcycle are as follows:

State	Time	Applicable Cycles	Action
T0	0 nsec	all	Precharge EU and MMU register files
			Clock MDMR-L synchronizer
T25	25 nsec	all	Start EU register file access
		all	Assert MALE-L
T75	75 nsec	all	Start ALU operation
		all	Strobe MPRDC-L
		all	Assert MSTRB-L
		read	Assert MBUFCTL-L
		relocation or expl access	Start MMU register file access
		all	Drive MBS-H<1:0> from cache bypass status and CCR<3> + CCR<2>
		all	Drive MMAP-L with DMA grant derived from output of MDMR-L synchronzier
		all	Drive ALU status onto MDAL<21:16>
T125	125 nsec	relocation	Start relocation add/compare
T150	150 nsec	read	Latch selected input latch from MDAL-H<15:0>
		all	Deassert MALE-L
		read	Deassert MBUFCTL-L
		mem I/O	Strobe MMISS-L
T175	175 nsec	prefetch	Latch PPC
		other relocation or output	Latch output latch

If the microcycle is not extended, T175 coincides with T-25. If the microcycle is extended, operations proceed as follows:

State	Time	Applicable Cycles	Action
T200	200 ns	extended non-writes	Assert MBUFCTL-L
Tx25	225, 325, ... ns	all extended	Clock MCONT-L synchronizer flip-flop
T250	250 ns	all extended	Assert MSCTL-L
Tx50	250, 350, ... ns	all extended	
Tx75	275, 375, ... ns	all extended	Clock MCONT-L holding flip-flop
Tx00	300, 400, ... ns	all extended	

All microcycles conclude as follows:

State	Time	Applicable Cycles	Action
T-100	300, 400, ... ns	all extended	Deassert MSCTL-L
T-50	350, 450, ... ns	extended non-writes	Deassert MBUFCTL-L
T-25	175 ns if not extended, else 375, 475, ... ns	all	Latch MIR from MIB-H<21:0> Write EU register file Drive MDAL-H<21:0> from PPC or output latch Drive MBS-H<1:0> from PCS<3:2> or bank select latches Drive MMAP-L from MMR3<5> Drive MABORT-L from PCS<0> or abort latch
TEND	200 ns if not extended, else 400, 500, ... ns	all	Deassert MSTR8-L

2.0 EXECUTION UNIT (EU)

The Execution Unit contains the main register file, special B-bus logic, the data path proper, branch/jump logic, the interface to the external MDAL bus, the Microinstruction Register, and the state sequencer.

2.1 Register File

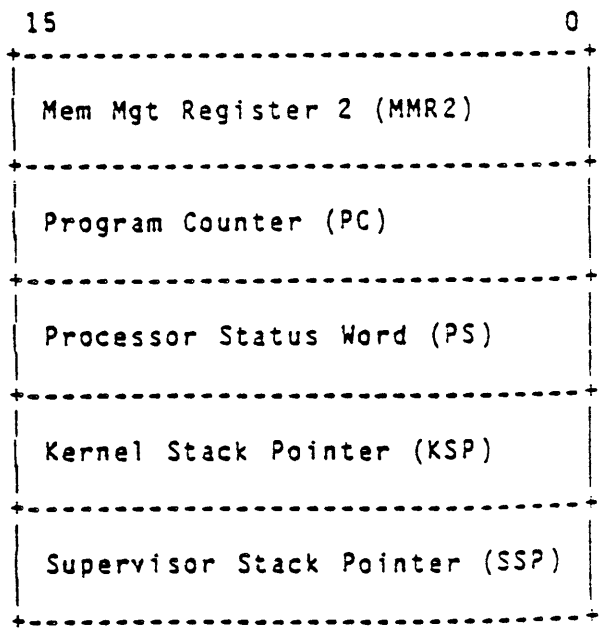
2.1.1 Physical Organization

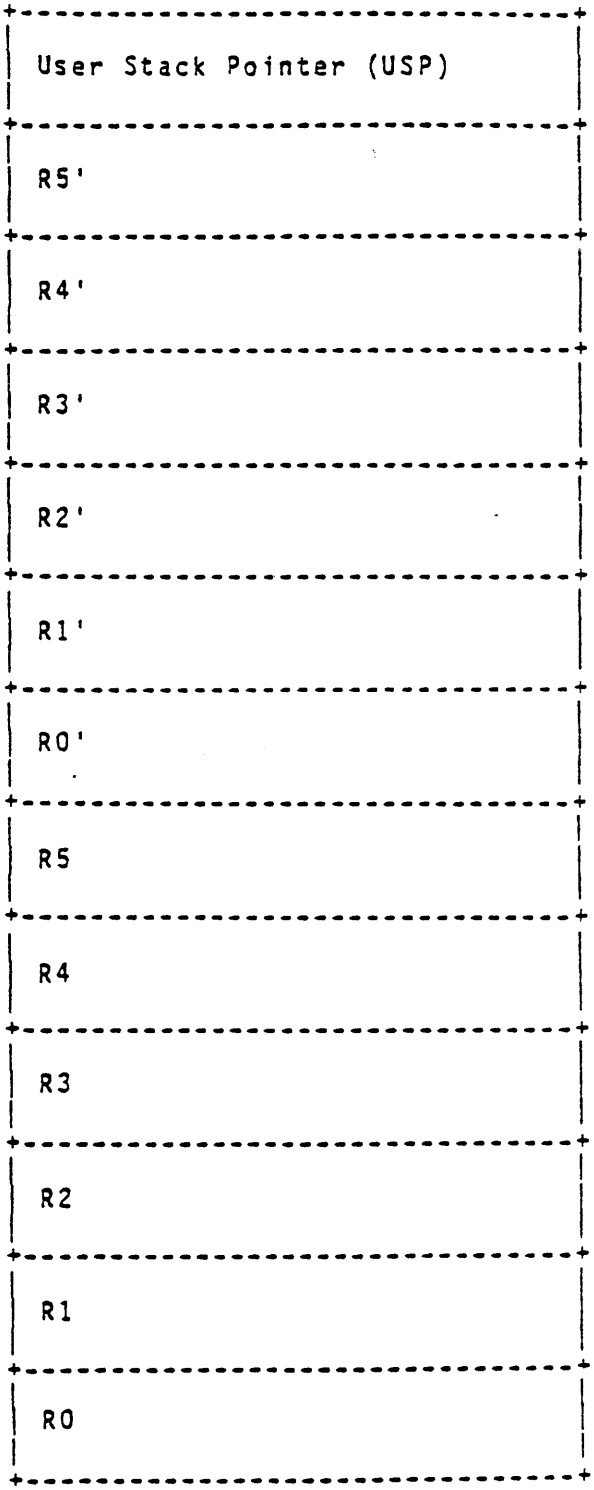
The EU register file is organized in two sections. The first section is 18 registers deep by 16 bits wide. The second section is 9 registers deep by 32 bits wide. The 18 by 16 section contains two sets of PDP-11 general registers (R0-R5, R0'-R5'), three stack pointers (KSP, SSP, USP), the Processor Status Register (PS), the Program Counter (PC), and Memory Management Register 2 (MMR2). The 9 by 32 section contains the Instruction Register (IR) and its field extractor, and eight scratch registers for microcode use.

The registers in the EU file (except for MMR2 and the IR) are dual ported. In any microcycle, an EU register can be accessed through both the A-port (which is read/write) and the B-port (which is read only). The A-port connects to the A-bus and thence to the data path, the B-port to the B-bus. The A-port of the register file is also connected to the I/O multiplexor which interfaces the EU to the external MDAL bus. Thus the contents of the EU registers can be modified from or written out to the external system bus.

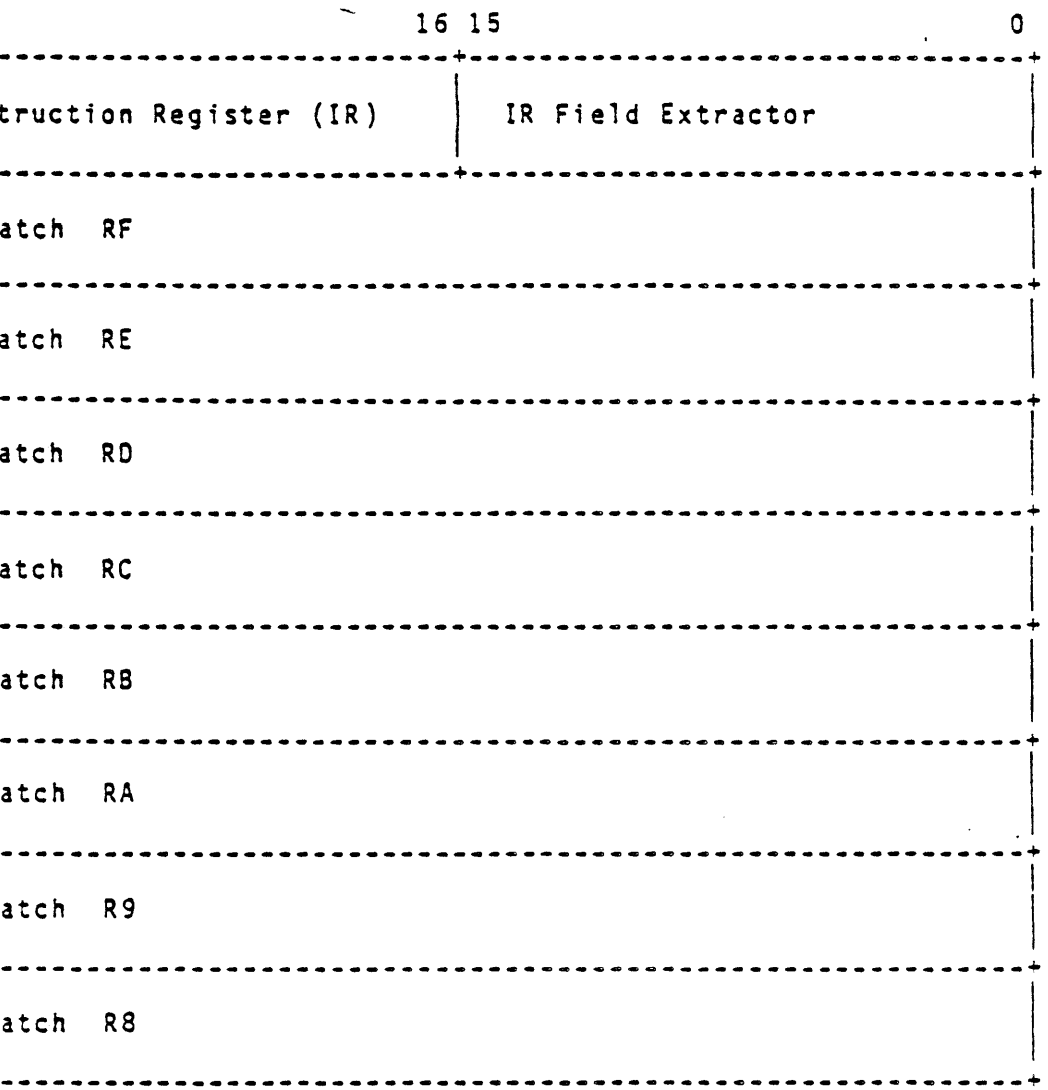
Note that MMR2 is read only and can be accessed only through the B-port.

The 18 by 16 register area is organized as follows:




User Stack Pointer (USP)
R5'
R4'
R3'
R2'
R1'
R0'
R5
R4
R3
R2
R1
R0

The 9 by 32 register area is organized as follows:



The Instruction Register (IR) connects to bits<31:16> of the A-port and, through the field extractor, to bits<15:0> of the B-port.

2.1.1.1 32-bit Scratch Registers

The scratch registers are accessible only from the microcode. Their state at power up is UNDEFINED.

2.1.1.2 PDP-11 General Registers (Rn, Rn')

There are two sets of general registers in the register file. The register set information in the PS (bit<11>) defines which set is selected. PS<11> = 0 selects R0 - R5; PS<11> = 1 selects R0' - R5'. The microcode can access only the selected register set.

The state of the PDP-11 general registers at power up is UNDEFINED.

2.1.1.3 Stack Pointers (SP)

There are three stack pointer registers in the register file to support the three processor modes (kernel, supervisor, user). The User Stack Pointer (USP) is used for stack references in user mode. The Supervisor Stack Pointer (SSP) is used for stack references in supervisor mode. The Kernel Stack Pointer (KSP) is used for kernel mode stack operations. The current mode field in the PS (bits<15:14>) selects the stack pointer to be used for the execution of all microinstructions except MOVPM. This microinstruction uses the previous mode field of the PS (bits<13:12>) to select the stack pointer. The microcode can access only the selected stack pointer.

If the mode field used to access the stack pointer specifies the illegal mode (= 10), the USP is selected.

The state of the stack pointers at power up is UNDEFINED.

2.1.1.4 Program Counter (PC)

The Program Counter is useable as a general purpose register in macro code (R7). It can also be incremented by 2 independently of the EU data path. This operation automatically happens during an RDI microinstruction or when a predecode (MPRDC-L) signal is received from the Control chip.

The state of the program counter at power up is UNDEFINED.

2.1.1.5 Memory Management Register 2 (MMR2)

MMR2 is a read-only register used to recover from memory management errors. If MMR0<15:13> = 000 and if either the predecode signal (MPRDC-L) is received or the current microinstruction is RDF, the Data chip loads the unincremented contents of the PC into MMR2.

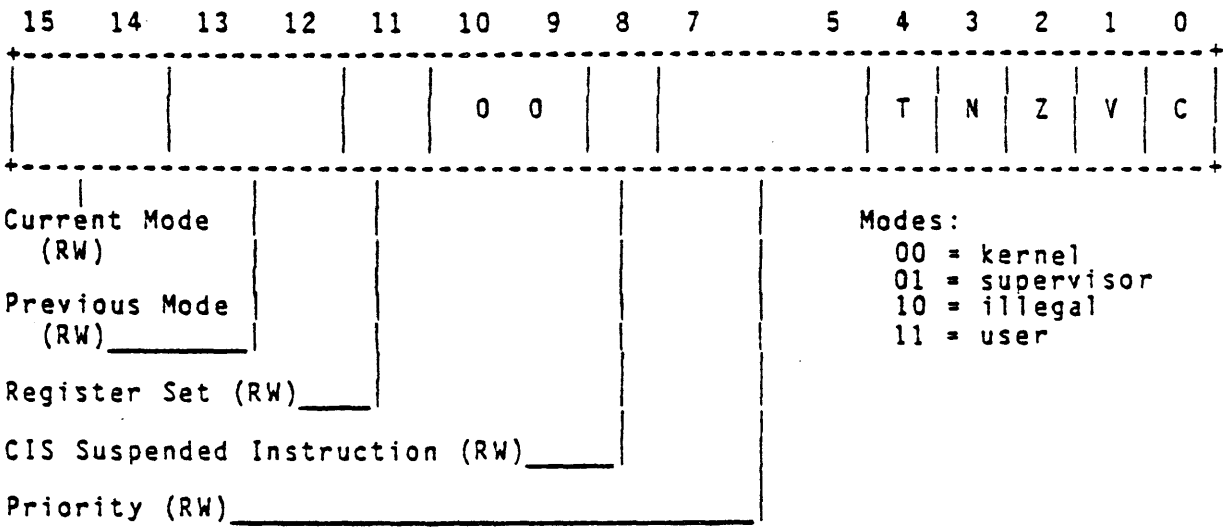
MMR2 can be read as an explicit memory location (17777576/7). It can also be read by microcode as an entry in the EU register file.

The state of MMR2 at power up is UNDEFINED.

2.1.1.6 Processor Status Register (PS)

2.1.1.6.1 Register Format

The PS has dedicated fields which are independently accessed in complex ways:



The PS can be accessed as an explicit location (17777776/7). It can also be accessed by microcode as an entry in the EU register file. The current mode (PS<15:14>) and previous mode (PS<13:12>) fields, as well as the register set select (PS<11>), are used by the EU and MMU register decoders in register access and relocation operations. Furthermore, the PS condition codes (N, Z, V, C) can be set by the microcode and reflect the results of an ALU operation. If the PS is written both explicitly (i.e., as a memory location) and implicitly (i.e., update condition codes) in the same microcycle, the PS condition codes (bits<3:0>) are set from the memory data, rather than from the EU status flags. Explicit and microcode PS references do not always completely overwrite the register. Table 2-1 summarizes the protection of the PS in various modes:

Table 2-1
PS PROTECTION

Access Method	Protection Mechanism	Protection
vector read	PS protection logic	PS<13:12> set from PS<15:14>
RTI/RTT read	PS protection logic	In supervisor or user mode, PS<15:11> are ORed with new value; PS<7:5> can not be altered

explicit write	PS protection logic	PS<4> can not be altered
MTPS	microcode	In supervisor or user mode, PS<7:5> cannot be altered; PS<15:8,4> can not be altered
console write	microcode	PS<4> can not be altered
power up	microcode	PS set to predefined value

2.1.1.6.2 PS Protection Logic

The PS protection logic handles protection of the PS. There are three distinct functions:

- Vector-to-PS protection
- RTI/RTT-to-PS protection
- Explicit write protection

Vector-to-PS protection is activated by a RD or RMW microinstruction with control bit<10> = 0. RTI/RTT-to-PS protection is activated by a RD or RMW microinstruction with control bit<9> = 0. In both cases, the destination register (Ra) should be PS. Explicit write protection is activated by an external write to the explicit PS address (17777776/7).

If vector-to-PS protection is specified, bits<15:14,11,8:0> of the input data are directly transferred to Ra. Bits<13:12> of the input, however, are suppressed, and the old contents of Ra<15:14> are transferred to Ra<13:12>. The microcode must guarantee that Ra=PS to protect the PS.

If RTI/RTT-to-PS protection is selected, the action taken depends on the current memory management mode. If the current mode is kernel, bits <15:11,8:0> of the input data are directly transferred to Ra. If the current mode is supervisor or user, only bits<8,4:0> of the input are directly transferred to Ra. Of the remaining bits, Ra<7:5> are not modified. Input bits<15:11> are first ORed with the old value of Ra<15:11> and then are written into Ra. The microcode must guarantee that Ra=PS to protect the PS.

Finally, on an explicit write to PS, bits<15:11,8:5,3:0> of the input are directly transferred to PS. PS<4> is not modified. Register Ra is also written with the same data. An explicit write may write only a byte of the PS.

A summary of the PS protection logic is given in Table 2-2.

Table 2-2
PS PROTECTION LOGIC

PS Write Data	Vector-to-PS	RTI/RTT User/Super	RTI/RTT Kernel	Explicit Write
Cond. Codes <3:0>	input<3:0>	input<3:0>	input<3:0>	input<3:0>
Trap <4>	input<4>	input<4>	input<4>	PS<4>
Priority <7:5>	input<7:5>	Ra<7:5>	input<7:5>	input<7:5>
Instr. Suspen. <8>	input<8>	input<8>	input<8>	input<8>
Register Set <11>	input<11>	input<11> OR Ra<11>	input<11>	input<11>
Previous Mode <13:12>	Ra<15:14>	input<13:12> OR Ra<13:12>	input<13:12>	input<13:12>
Current mode <15:14>	input<15:14>	input<15:14> OR Ra<15:14>	input<15:14>	input<15:14>

2.1.1.7 PDP-11 Instruction Register (IR)

2.1.1.7.1 Instruction Register

The Instruction Register (IR) is automatically loaded from the prefetch buffer at the end of a microcycle in which the predecode signal (MPRDC-L) is asserted. It can be accessed through the B-port using the IR field extractor (see below), or through the A-port as a high-word scratch register. The extractor is activated by any microinstruction which specifies the IR as the B-port operand. Its action is controlled by the microinstruction. AOBC and SOBC extract a subfield of the IR. All other microinstructions access the entire IR.

The IR serves several purposes in the Data chip. Bits<7:0> contain the PC offset for all macro-branch instructions. Bits<5:0> hold data for the SOB and MARK macroinstructions. The PC offset, SOB offset, and MARK data are accessed through the IR field extractor.

2.1.1.7.2 Shadow Instruction Register (SIR)

The Shadow Instruction Register copies bits<15,10:6,2:0> from the prefetch buffer on a predecode. It is also loadable by microcode with an OUTR microinstruction with control code 207 from register Ra. The SIR serves several purposes in the Data chip. Bits<8:6> and <2:0> are used to indirectly address PDP-11 general registers in the EU register file. Bits<7:6> and <2:0> are used to indirectly address floating point accumulators in the MMU register file. Bits<15,10:8> are used to interrogate the condition codes in the PS for the PDP-11 macro-branch instructions.

2.1.1.7.3 IR Field Extractor

The IR field extractor connects the IR to the B-port of the EU register file. The field extractor is activated by any microinstruction which specifies the IR as the B-port operand. The action of the field extractor is controlled by the microinstruction. Table 2-3 summarizes the actions performed by the IR field extractor.

Table 2-3
FIELD EXTRACTOR ACTION

MACRO CODE	MICROINSTRUCTION	B-BUS RESULT
BR, BNE, BEQ, BGE, BLT, BGT, BLE, BPL, BMI, BHI, BLOS, BVC, BVS, BCC, BCS	AOBC	B<8:1> <-- IR<7:0> B<0> <-- 0 B<15:9> <-- IR<7>
MARK, SOB	SOBC	B<6:1> <-- IR<5:0> B<0> <-- 0 B<15:7> <-- 0
all others	all others	B<15:0> <-- IR<15:0>

2.1.2 Decode and Access

All the registers in the EU can be accessed from the microcode. In addition, MMR2 and PS can be accessed explicitly as memory locations.

2.1.2.1 Microcode Access

The microcode can always access the 32-bit scratch registers, PC, MMR2, and PS, but can only access the selected stack pointers and general register set. Selection of the stack pointer is determined by the current processor mode (PS<15:14>, except for MOVPM); of the general register set, by the register set select (PS<11>).

Except for MMR2 and the IR, the EU registers are dual ported. Access to the EU registers through the A-port and the B-port is controlled by bits<4:0> and bits<9:5>, respectively, of the microinstruction. The codes in these register select fields provide for direct addressing of all the selected EU registers and indirect addressing of the selected general register set, selected stack pointer, and PC by the two register designator fields of the PDP-11 instruction. Register select information is latched and held constant for the duration of the cycle to prevent incoming information for the next microcycle from altering register selection. The register select codes are given in Table 2-4. Note that A- and B-port select codes are not equivalent. Note also that the IR field extractor is single ported since it is not accessible on the A-port.

When sixteen-bit registers are accessed on the B-bus, B-bus bits<31:16> are forced to zero; when read out on the A-bus, A-bus bits<31:16> contain the IR.

Table 2-4
EU REGISTER ACCESS CODES

Access code	selects Rb<31:16>	Rb<15:0>	Ra<31:16>	Ra<15:0>	Notes
0	0	PS	IR	PS	
1	0	IR field extractor	IR	PS	
2	0	MMR2	IR	PS	
3	0	Rs OR 1	IR	Rs OR 1	1
4	RE	RE	RE	RE	
5	RF	RF	RF	RF	
6	0	Rd	IR	Rd	2
7	0	Rs	IR	Rs	3
10	R8	R8	R8	R8	4
11	R9	R9	R9	R9	4
12	RA	RA	RA	RA	4
13	RB	RB	RB	RB	4
14	RC	RC	RC	RC	4
15	RD	RD	RD	RD	4
16	RE	RE	RE	RE	4
17	RF	RF	RF	RF	4
20	0	R0	IR	R0	
21	0	R1	IR	R1	
22	0	R2	IR	R2	
23	0	R3	IR	R3	
24	0	R4	IR	R4	
25	0	R5	IR	R5	
26	0	R6	IR	R6	
27	0	R7	IR	R7	
30	R8	R8	R8	R8	
31	R9	R9	R9	R9	
32	RA	RA	RA	RA	
33	RB	RB	RB	RB	
34	RC	RC	RC	RC	
35	RD	RD	RD	RD	
36	RE	RE	RE	RE	
37	RF	RF	RF	RF	

Note 1 - Rs OR 1 - select Rn, n = (SIR<8:6> OR 1).

Note 2 - Rd - select Rn, n = SIR<2:0>.

Note 3 - Rs - select Rn, n = SIR<8:6>.

Note 4 - Affects operation of ALU, I/O mux, write amplifiers, otherwise same as 30-37.

2.1.2.2 Explicit Access

MMR2 and PS can be explicitly accessed, byte or word, as memory locations. Table 2-5 lists the explicit addresses of these registers:

Table 2-5
EXPLICITLY ADDRESSABLE EU REGISTERS

Name	Address	Special Requirements
PS	17777776/7	PS<4> not modified by explicit write
MMR2	17777576/7	Read only

Selection of these registers is detected by the MMU address adjust and detect logic.

2.1.3 Register Read Operations

The EU register file places data on both the A-bus and the B-bus during a microinstruction. Some microinstructions, however, substitute other information on the B-bus or modify the B-bus data:

- Execution of a literal microinstruction places a literal generated from bits<12:5> of the microinstruction on the B-bus (see section 2.2.1).
- Execution of a decimal adjust microinstruction places data from the decimal adjust logic on the B-bus (see section 2.2.2).
- Execution of a MOVS microinstruction sign extends bit<7> of the B-bus through bits<31:8> (see section 2.2.3).

2.1.4 Register Write Operations

The EU register file is written by four sets of write amplifiers (one set per byte) through the A-port only. Since file writing is only permitted for certain microinstructions, and even then sometimes on a per word or per byte basis only, the write amplifier sets can selectively write any of the four bytes of the destination register. When some of the write amplifer sets are disabled, the associated byte or bytes in the destination register are not altered.

Operation of the write amplifier sets is controlled by the microinstruction decode and, for references to the 32-bit scratch registers, by the A-port register select code. The microinstruction decode detects if the EU data path output is to be written back to the A-port, the operand length, and any special conditions which control the action of the write amplifier sets. The A-port select code controls which half of a 32-bit register is overwritten. This is illustrated below:

Table 2-6
OPERATION OF WRITE AMPLIFIERS

Microinstruction Length Field	A-port Select Code <4:0>	Microinstruction	Bits Written
byte	00XXX + 1XXXX	-(SWAB.B + LLSW.B)	<7:0>
	00XXX + 1XXXX	SWAB.B + LLSW.B	<15:8>
	01XXX	-(SWAB.B + LLSW.B)	<23:16>
	01XXX	SWAB.B + LLSW.B	<31:24>
word	00XXX + 1XXXX	any	<15:0>
	01XXX	any	<31:16>
longword	any	any	<31:0>

On both MMU and bus read aborts, the original contents of the registers are preserved.

Note that if MPRDC-L is asserted, the write amplifiers for bits<31:16> are automatically enabled to write the IR, irrespective of the microinstruction.

2.2 Special B-bus Logic

Normally, the EU data path inputs are the A-port and B-port outputs of the EU register file. However, certain microinstructions place different information on or modify the B-bus input.

2.2.1 Literal Generation Logic

During execution of any literal microinstruction, this logic disconnects the B-port of the EU register file from the B-bus and generates a 32-bit operand for the B-bus from the 8-bit literal field in the microinstruction. The 8-bit literal (bits<12:5> of the microinstruction) is copied to bits<7:0> of the B-bus. Bits<31:8> of the B-bus are zeroed.

2.2.2 Decimal Adjust Logic

A decimal adjustment is required before and after a BCD operation because only binary arithmetic can be performed in the EU data path. Prior to any BCD addition, 6 is added to each of the nibbles in one of the operands. This is accomplished with a PDAD microinstruction. PDAD disconnects the B-port of the EU register file from the B-input of the EU data path, inserts the constant 66666666 (hexidecimal) on the B-input, and adds it to the A-port operand, forming the preadjusted result. Prior to any BCD subtraction, the subtrahend is one's complemented. This is accomplished with a normal COM microinstruction.

After preadjustment, the BCD operation is performed as a binary addition and the carry-out from each nibble is saved. A post-adjustment (invoked by a DADP microinstruction) is then executed to recover the final BCD answer. Specifically, a BCD correction constant is formed - nibble by nibble - on the B-input and is added (with a carry-in to the least significant bit of the addition operation) to the intermediate BCD result yielding the corrected BCD result. Each nibble BCD correction constant is either a 9 or F (hexidecimal) and is selected by the state of its saved carry-out from the binary add. If there was a carry-out, the constant F is used. Conversely, no carry-out results in the constant 9. The following example demonstrates the BCD capabilities of the Data chip:

Example: BCD Addition Capability

Operand A:	0	0	0	1	9	3	4	6
Operand B:	0	0	1	2	4	6	8	0

NOTE: All numbers are hexidecimal nibbles.

Step 1: Preadjust Operand (PDAD)

	0	0	0	1	9	3	4	6
	6	6	6	6	6	6	6	6
	6	6	6	7	F	9	A	C

Preadjusted result

Step 2: Binary Addition and Save Carries (Add)

	6	6	6	7	F	9	A	C
	0	0	1	2	4	6	8	0
	6	6	7	A	4	0	2	C
	0	0	0	0	1	1	1	0

Intermediate result

Saved carry out

Step 3: Post-adjustment (DADP)

```
  6 6 7 A 4 0 2 C
  9 9 9 9 F F F 9
                1   Carry into the LSB
  0 0 1 4 4 0 2 6   Final result
```

2.2.3 Sign Extension Logic

The sign extension logic modifies the data on the B-bus by sign extending the low order byte to 32 bits. The MOVS microinstruction sign extends the low order byte.

2.3 EU Data Path Components

The EU data path performs the arithmetic and logical operations in the Data chip. It consists of an arithmetic and logic unit (ALU), a combination bit shifter and byte swapper, and a shift extension register. The inputs to the ALU are the A-bus (A<31:0>) and B-bus (B<31:0>), which are usually driven by the A-port and B-port, respectively, of the EU register file. The output of the ALU connects to the shifter/swapper over the S-bus (S<31:0>). The output of the shifter/swapper connects to the I/O multiplexor over the F-bus (F<31:0>).

The operation of the EU data path is determined by the current microinstruction.

2.3.1 EU Data Path Functions

2.3.1.1 Operations

The EU data path is capable of the following operations:

- logical operations on two operands
- binary addition, with optional carry in
- binary subtraction, with optional borrow in
- left shift one bit, with optional carry in
- right shift one bit, with optional carry in
- byte swap within words
- byte shift left or right one bit followed by byte swap (low order word only)
- conditional add/subtract/negate
- combined conditional add/subtract and shift
- decimal adjustment

For further details, see section 5.0, Microinstructions.

2.3.1.2 Data Flow

Data flow within the EU data path depends on the microinstruction being executed, the length (byte/word/longword) of the operation being performed, and, for operations on the 32-bit scratch registers, the A-port and B-port register select codes. For the microinstructions which do not have a B-port register select field, the EU data flow behaves as if the B-port code is 00XXX + 1XXXX. This is summarized in Table 2-7.

Table 2-7
EU DATA FLOW

Length field	B-port code <9:5>	A-port code <4:0>	Micro-instruction	Effective operands	Store in
byte	00XXX+1XXXX	00XXX+1XXXX	-(SWAB.B+LLSW.B)	<7:0>	<7:0>
			SWAB.B+LLSW.B	<7:0>	<15:8>
	00XXX+1XXXX	01XXX	-(SWAB.B+LLSW.B)	<7:0>	<23:16>
			SWAB.B+LLSW.B	<7:0>	<31:24>
	01XXX	00XXX+1XXXX	-(SWAB.B+LLSW.B)	<23:16>	<7:0>
			SWAB.B+LLSW.B	<23:16>	<15:8>
	01XXX	01XXX	-(SWAB.B+LLSW.B)	<23:16>	<23:16>
			SWAB.B+LLSW.B	<23:16>	<31:24>
word	00XXX+1XXXX	00XXX+1XXXX	any	<15:0>	<15:0>
	00XXX+1XXXX	01XXX	any	<15:0>	<31:16>
	01XXX	00XXX+1XXXX	any	<31:16>	<15:0>
	01XXX	01XXX	any	<31:16>	<31:16>
longword	any	any	any	<31:0>	<31:0>

2.3.1.3 Status Flags

AN, AV, AZ, and AC are the local EU data path status flags. They are updated at the conclusion of an EU data path operation. Depending on instruction type, they may be set or cleared unconditionally, or affected conditionally to give additional information about the operation just completed. In the latter case, they generally have the following meaning:

AN=1: result is negative
=0: otherwise

AZ=1: result is zero
=0: otherwise

AV=1: operation results in an arithmetic overflow
=0: otherwise

AC=1: non-subtract operation results in a carry;
subtract-type operation does not result in a carry
=0: otherwise

The derivation of the status flags depends on the microinstruction being executed, the length of the operation being performed, and, for operations on the 32-bit scratch registers, the A-port register select code. This is detailed in Appendix B (Condition Code Equations) and summarized in Table 2-8.

Table 2-8
STATUS FLAGS DERIVATION

Microinstruction length field	A-port select code <4:0>	Microinstruction	Normally derive AN,AZ from A	Normally derive AV,AC from Cin/out
byte	00XXX + 1XXXX	-(SWAB.B + LLSW.B)	<7:0>	<7>
		SWAB.B + LLSW.B	<15:8>	na
	01XXX	-(SWAB.B + LLSW.B)	<23:16>	<7>
		SWAB.B + LLSW.B	<31:24>	na
word	00XXX + 1XXXX	-(SWAB.W + LLSW.W)	<15:0>	<15>
		SWAB.W + LLSW.W	<15:8>	na
	01XXX	-(SWAB.W + LLSW.W)	<31:16>	<15>
		SWAB.W + LLSW.W	<31:24>	na
longword	any	any	<31:0>	<31>

Note that the AV status flag can reflect multi-bit left shift operations, and the AZ flag can be used for multi-word zero testing. Section 2.3.1.4 details multi-bit shifting and multi-word zero testing.

The four local status flags are conditionally written under microprogram control into the PS condition codes at the end of the microcycle.

2.3.1.4 Extended Status Testing

The EU data path supports multi-word zero testing. The multi-word microinstructions ADDC, SUBC, and NEGC affect the local AZ status flag as follows:

```
AZ = 0      if result not equal to 0
   = Z      otherwise
```

That is, AZ is cleared if the result is non-zero; otherwise it is copied from the PS Z flag.

The EU data path also supports multi-bit shift overflow testing. The data path includes an extra flag called the sticky V flip-flop. This flip-flop is cleared by predecode (MPRDC-L asserted). It is set by any ASL microinstruction which sets AV. It is also an OR input into AV. Thus, once AV is set by an ASL microinstruction, it remains set during subsequent ASL microinstructions until predecode is recognized. If predecode is recognized during an ASL microinstruction, AV is evaluated with the sticky overflow flip-flop (SVFF) before it is cleared.

The EU data path also supports double-precision divide. The data path includes two extra flags called the extended carry flag (XC) and the extended rotate flag (XR). During an XLDIVS microinstruction, XC captures the carry out of the ALU, and XR the carry out of the shifter. During an XHDIVS, XC supplies the carry into the ALU, and XR the carry into the shifter.

2.3.2 Arithmetic-Logical Unit (ALU)

The ALU performs the arithmetic, logical, and selected left shift operations in the EU data path. The inputs to the ALU are the A-bus and B-bus; its output is the S-bus. The ALU function is determined by decoding the current microinstruction.

The input operands are fed to the P and G logic to generate the propagate and generate terms for the desired ALU function. The propagate and generate terms are used by the carry logic to derive the carry terms. The carry and propagate terms are XOR'd to yield the result. (Some of the carry terms are also used to control the decimal adjust logic, section 2.2.2.) The low order eight bits of this result are fed to a multiplexor which operates as follows:

microinstruction	multiplexor action
-(ASR.B + ROR.B + LSR.B)	S<7:0> <-- ALU result<7:0>
ASR.B	S<7> <-- ALU result<7> S<6:0> <-- ALU result<7:1> carry out <-- ALU result<0>
ROR.B	S<7> <-- C S<6:0> <-- ALU result<7:1> carry out <-- ALU result<0>
LSR.B	S<7> <-- 0 S<6:0> <-- ALU result<7:1> carry out <-- ALU result<0>

The output of the ALU and this multiplexor connect to the shifter/swapper over the S-bus.

2.3.3 Bit Shifter and Byte Swapper

The shifter/swapper receives its input from the S-bus (S<31:0>). It can shift data left or right one bit, and can be used in conjunction with a special shift register (SR) to perform relatively high speed multiplies, divides, and multi-word shifts. It can also swap the bytes within the low word and high word outputs from the ALU. The function performed by the shifter/swapper is determined by the current microinstruction. Note that a byte write to an odd address automatically invokes the byte swap function. Ra<15:8> will get written with the data, not Ra<7:0>.

The inputs to the shifter/swapper consist of the 32-bit ALU result (S<31:0>) and a shift-in signal; the outputs consist of a 32-bit result (F<31:0>) and a shift-out signal. The shifter works with 8, 16, or 32 bit values. This is accomplished by selecting where the shift-out and shift-in are connected to the shifter and SR. Table 2-9 details the functions performed by the shifter/swapper.

Note that the ROL.B and ASL.B microinstructions use an ALU shift to perform the left shift, while the ROR.B, ASR.B, and LSR.B microinstructions use an ALU shift to perform the right shift. This enables a bit shift and byte swap operation during the same microinstruction.

2.3.4 Shift Register (SR)

The shift register is a special purpose register used in conjunction with the shifter/swapper. It is addressable via an INPR or OUTR microinstruction with control code 206. This is encoded within the single port select codes shown in Table 3-3. The SR register accelerates the PDP-11 Extended Instruction Set (EIS). In particular, this hardware enables the inner multiply and divide loops to execute at the rate of one microcycle per bit.

The SR is a bidirectional shift register. It can provide the shift-in data to the shifter/swapper and can collect the shift-out data from the shifter/swapper. Furthermore, the EU data path can examine the low order bit in the SR and conditionally execute based on the result.

The SR connects to the A-bus. When written via any length OUTR, all 32 bits are modified according to the following table.

OUTR.x	[SR,Ra.H]	SR <-- 0
OUTR.B	[SR,Ra.x]	undefined
OUTR.L	[SR,Ra.L]	SR <-- Ra
OUTR.W	[SR,Ra.L]	SR<31:16> <-- 0 SR<15:0> <-- Ra<15:0>

When read via INPR, all 32 bits are read out, although the microinstruction length field controls how many bits are written into the destination (Ra) register.

In summary, the SR is a parallel loaded and read register which is shifted in parallel with the shifter/swapper, can be tested, and is 32 bits wide. Table 2-9 enumerates the operation of the shifter/swapper and the SR.

Table 2-9
DATA CHIP SHIFT/SWAP CAPABILITIES

[ALU shifts]

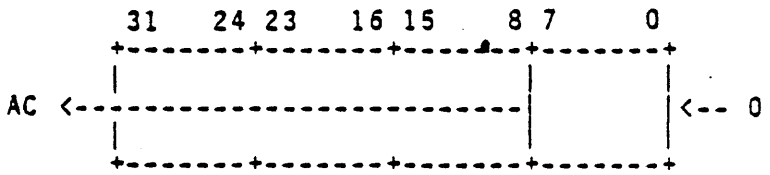
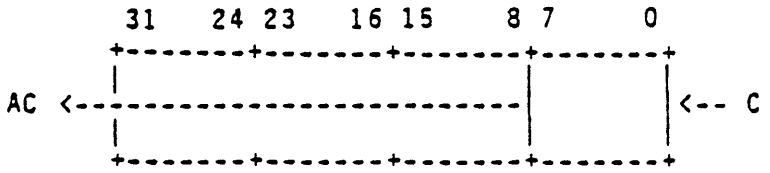
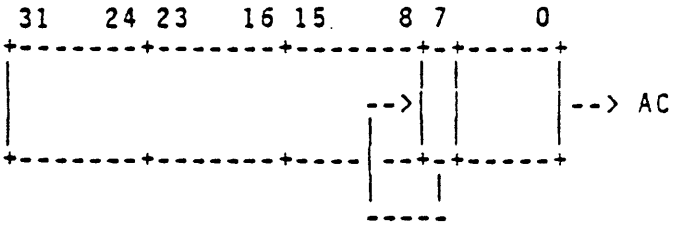
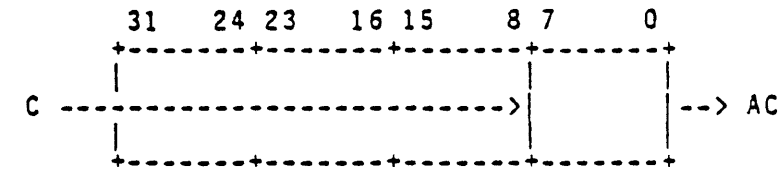
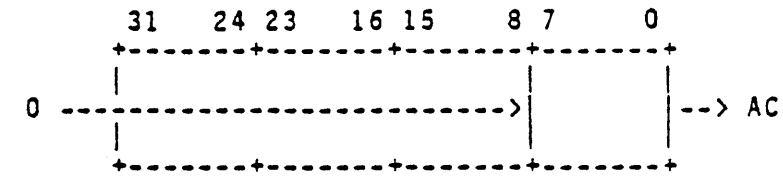
Microinstruction	ALU Shift Function
ASL.B	
ROL.B	
ASR.B	
ROR.B	
LSR.B	

Table 2-9 (continued)
DATA CHIP SHIFT/SWAP CAPABILITIES

[simple shifter/swapper functions]

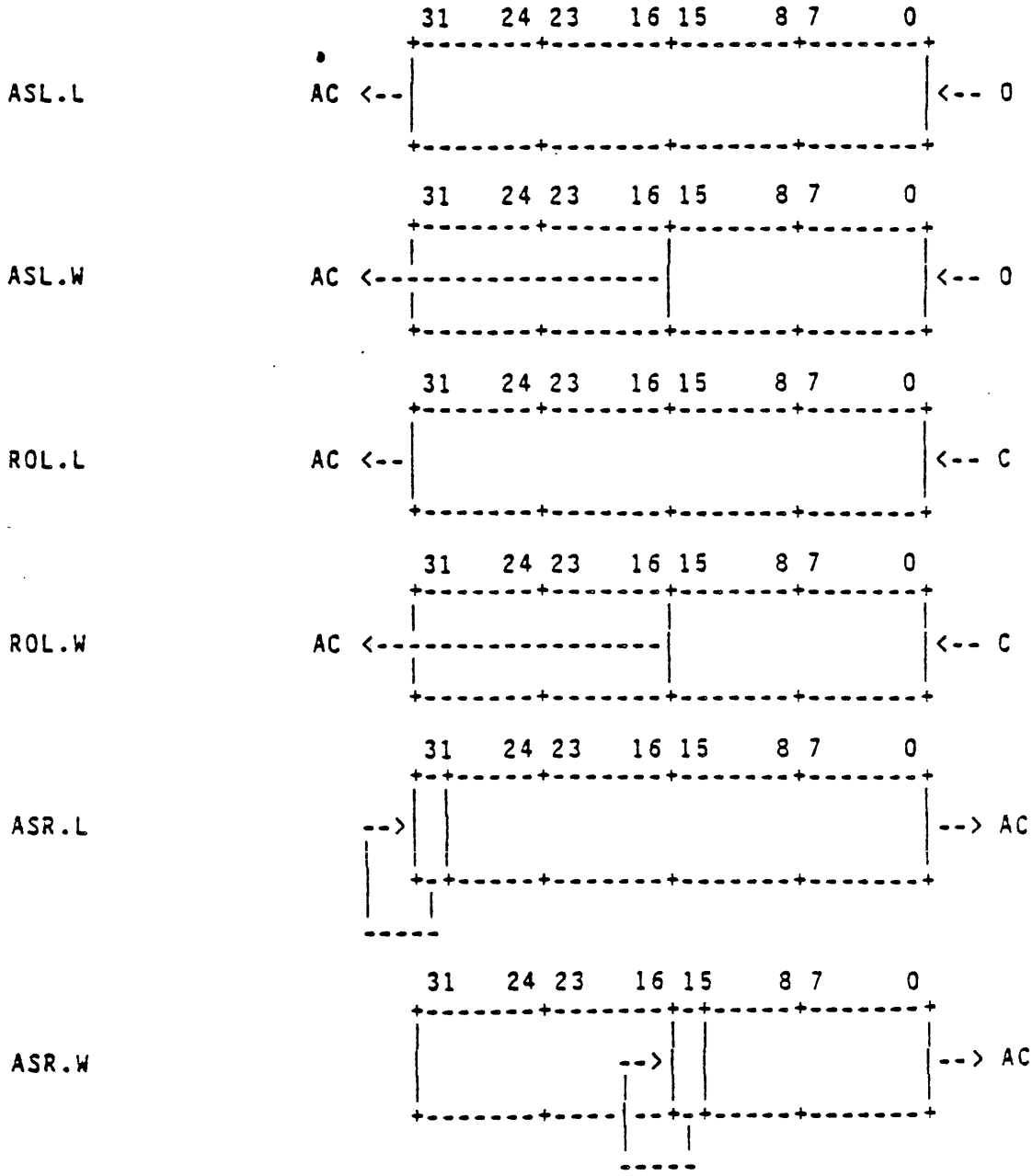


Table 2-9 (continued)
DATA CHIP SHIFT/SWAP CAPABILITIES

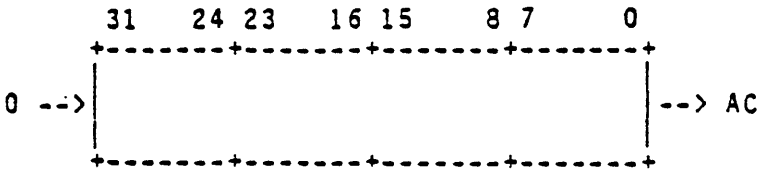
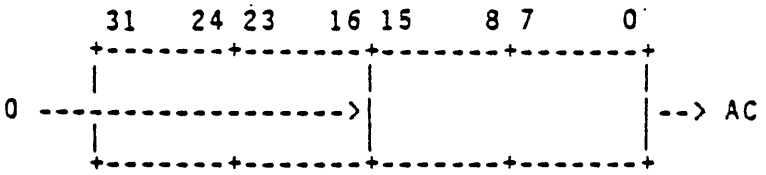
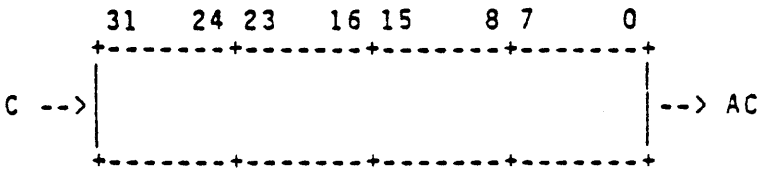
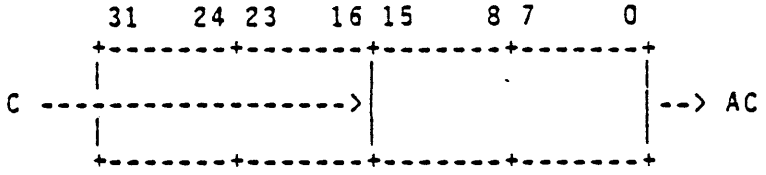
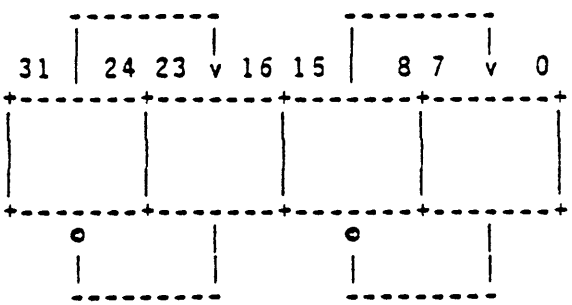
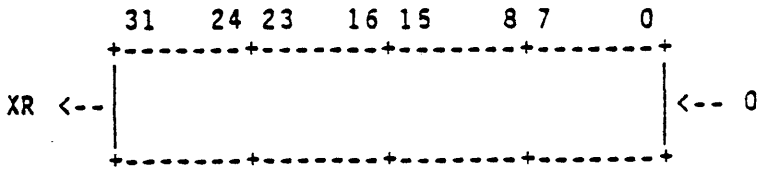
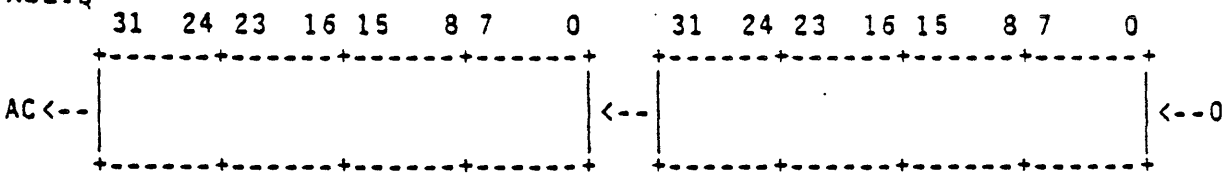
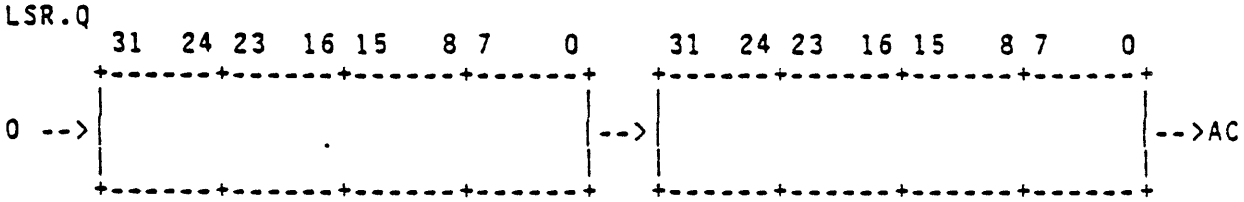
Microinstruction	Shifter Function
LSR.L	
LSR.W	
ROR.L	
ROR.W	
SWAB.(L)(W)(B)	
XLDIVS	

Table 2-9 (continued)
DATA CHIP SHIFT/SWAP CAPABILITIES

[quadword shifts]

Micro-instruction	Shifter Function	SR Function
ASL.Q		
LSR.Q		

[multiply/divide]

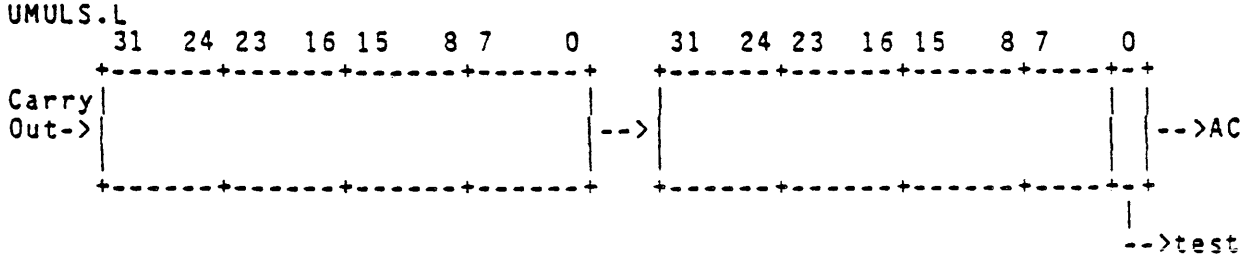
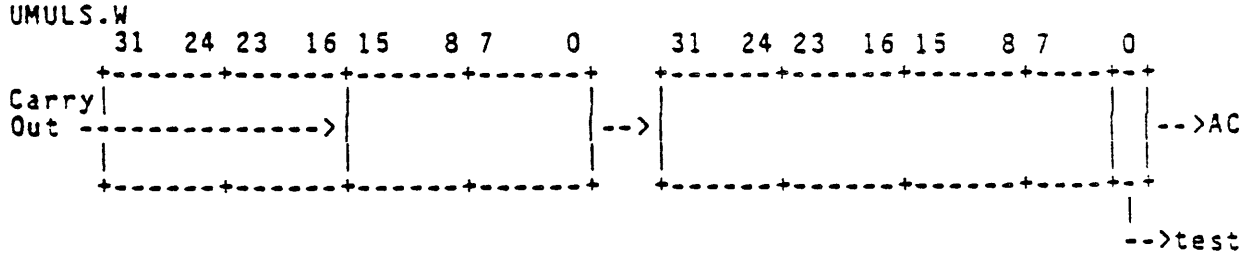
Micro-instruction	Shifter Function	SR Function
UMULS.L		
UMULS.W		

Table 2-9 (continued)
DATA CHIP SHIFT/SWAP CAPABILITIES

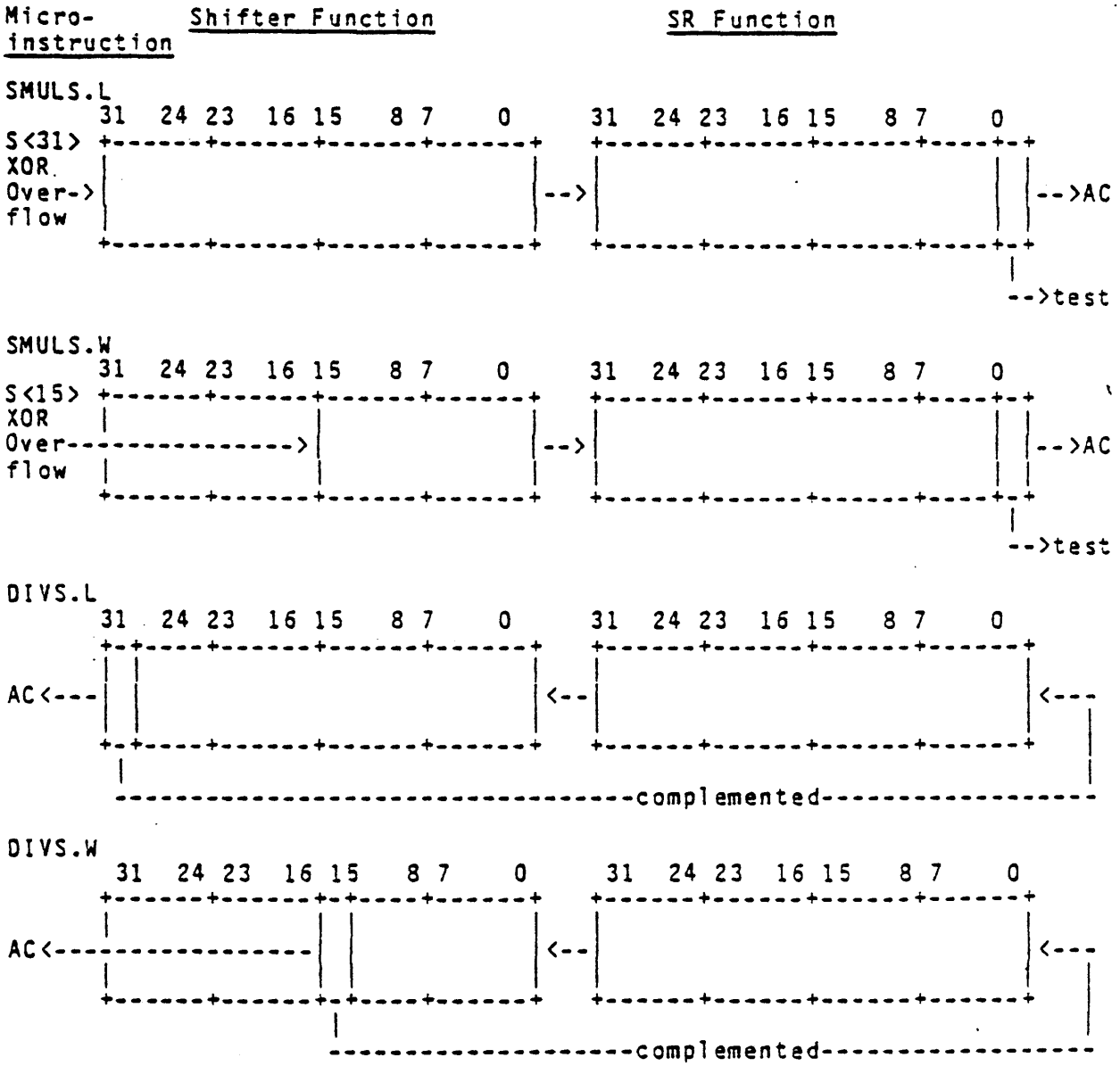
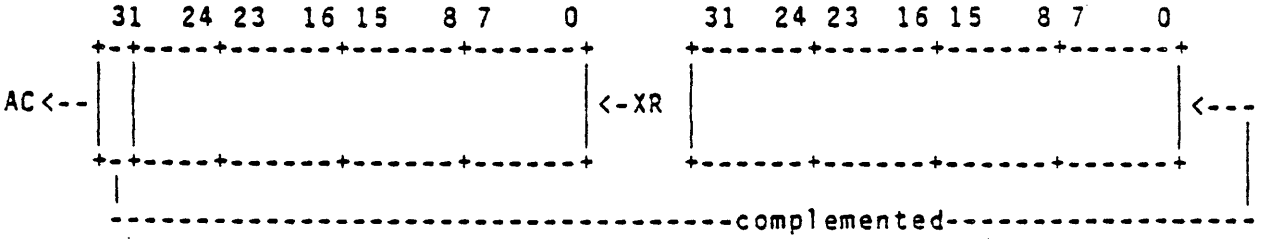


Table 2-9 (continued)
DATA CHIP SHIFT/SWAP CAPABILITIES

Micro-instruction	Shifter Function	SR Function
XHDIVS		

2.4 Branch/Jump Logic

This logic implements both the macro-branch microinstructions (AOBC, SOBC) and the micro-jump microinstructions.

2.4.1 Macro-branch Logic

The PS condition codes (N, Z, V, C) govern execution of PDP-11 conditional branches. Conditional branches are implemented using two microinstructions, Add On Branch Condition (AOBC) and Subtract On Branch Condition (SOBC).

AOBC conditionally performs an add operation based on conditions specified by bits<15,10:8> of the PDP-11 instruction in the SIR (see Table 2-10). If the stated branch condition is satisfied, the add operation is performed. Otherwise, the operands are left unchanged.

SOBC conditionally performs a subtract operation based on the local status flags set during the previous microinstruction (see Table 2-10). If the local status flags satisfy the branch condition, -AZ, the subtraction is performed. Otherwise, the operands are left unchanged.

In both AOBC and SOBC, the suppression of the arithmetic operation is implemented by inhibiting the write-back to the A-port. That is, the operation always takes place, and suppresion of the output is realized by disabling the write amplifiers. If the write-back is not suppressed (branch taken) and the destination register is the PC, the Data chip clears the prefetch buffer valid flag (PV) and asserts the kill prefetch buffer signal (MKPB-L).

Table 2-10
AOBC BRANCH CONDITIONS

SIR Bits: <15>	<10>	<9>	<8>	SPECIFIED BRANCH CONDITION	CORRESPONDING PDP-11 INSTR.
0	0	0	0	don't care	none
0	0	0	1	unconditional	BR
0	0	1	0	-Z	BNE
0	0	1	1	Z	BEQ
0	1	0	0	-(N XOR V)	BGE
0	1	0	1	N XOR V	BLT
0	1	1	0	-(Z + (N XOR V))	BGT
0	1	1	1	Z + (N XOR V)	BLE
1	0	0	0	-N	BPL
1	0	0	1	N	BMI
1	0	1	0	-(Z + C)	BHI
1	0	1	1	Z + C	BLOS
1	1	0	0	-V	BVC
1	1	0	1	V	BVS
1	1	1	0	-C	BCC
1	1	1	1	C	BCS

SOBC BRANCH CONDITION					
X	X	X	X	-AZ	SOB

2.4.2 Conditional Jump Testing Logic

The conditional jump logic tests various status flags in the Data chip (see Table 2-11). Both the local EU status flags (AN, AV, AZ, AC) and the processor status flags (N, V, Z, C) can be tested. It is also possible to test if the current mode (PS<15:14>) is kernel mode. If the stated condition is satisfied, the Data chip asserts the Jump Allow signal (MSOV/JA-L). This signals the Control chip to execute the jump. Note that the execution of a conditional jump is effectively delayed one microcycle, during which the microcode can specify a NOP instruction or, if possible, perform some useful operation.

Table 2-11
CONDITIONAL JUMP INSTRUCTIONS

Microinstruction	Condition Tested
JC	C = 1
JV	V = 1
JZ	Z = 1
JN	N = 1
JAC	AC = 1
JAV	AV = 1
JAZ	AZ = 1
JAN	AN = 1
JKM	PS<15:14> = 0

2.5 EU/MDAL Interface

The interface between the EU and the external MDAL bus consists of an I/O multiplexor, two input latches, and an output latch.

2.5.1 I/O Multiplexor

The I/O multiplexor provides a complex set of interconnects among the A-bus (A<31:0>), the output of the shifter/swapper (F<31:0>), the MMU data bus (M<15:0>), the input latches (IL[1:0]<15:0>), and the output latch (OL<15:0>). All connections are word width (16 bits); multiple paths may be enabled simultaneously to implement 32-bit transfers. The specific paths within the I/O multiplexor are as follows:

1- F<31:16> --> A<31:16>
2- F<15:0> --> A<31:16>
3- M<15:0> --> A<31:16>
4- IL[1:0]<15:0> --> A<31:16>
5- F<15:0> --> A<15:0>
6- F<31:16> --> A<15:0>
7- M<15:0> --> A<15:0>
8- IL[1:0]<15:0> --> A<15:0>
9- M<7:0,15:8> --> A<15:0>
10- IL[1:0]<15:8> --> A<7:0>
11- A<15:0> --> M<15:0>
12- F<31:16> --> M<15:0>
13- F<15:0> --> M<15:0>
14- F<7:0,15:8> --> M<15:0>

Note that:

- Paths 1,2,3,4 are mutually exclusive.
- Paths 5,6,7,8 are mutually exclusive.
- Paths 9,10 are mutually exclusive.
- Paths 11,12,13,14 are mutually exclusive.
- Paths 5 to 8 and 9 to 10 are mutually exclusive.
- Simultaneous use of paths 2 and 6 is not supported.
- The upper 16 bits of both the A-bus and the B-bus can not be used during a predecode microcycle.
- Other than F<31:0> --> A<31:0>, only 16 bits of the A bus can be driven with data at one time. The other 16 bits are zeroes.

2.5.2 Input Latch (IL[1:0]<15:0>)

The dual input latches (IL[1:0]<15:0>) latch incoming data from MDAL-H<15:0>. The latches are used on an alternating basis for instruction stream and data stream information. Operation of these latches is explained in detail in section 4.

2.5.3 Output Latch (OL<21:0>)

The output latch (OL<21:0>) latches outgoing data for MDAL-H<21:0>. It is loaded from the MMU data path with physical address information, and from the EU data path with data and status information.

2.6 Microinstruction Register (MIR<21:0>)

The Microinstruction Register (MIR<21:0>) latches the current microinstruction. It is loaded from MIB-H<21:0> at T-25. Microinstruction decoding is done locally throughout the Data chip as required.

Note that the Microinstruction Register is cleared by the assertion of MINIT-L.

2.7 State Sequencer

The state sequencer produces the internal control and timing signals used by the Data chip, as well as the external signals used by the Control chips and the System Interface. Inputs to the state sequencer include MCLK, MIR<21:5>, MINIT-L, MABORT-L, MBS-H<1:0>, MMISS-L, MMAP-L, MDMR-L, and MCONT-L. Its outputs include MALE-L, MBUFCTL-L, MSCTL-L, and MSTRB-L.

The Data chip enters a stretched microcycle if the following equation is true at T150:

```
STALL = STIO + MMAP-L + MABORT-L.DEMAND +
        READ.(MMISS-L + MBS-H<1> + MBS-H<0> + BSFF<1> + BSFF<0>) .
        (-MABORT-L)
```

where

STIO = current microinstruction is a stretched I/O microcycle (RDG, RDINTR, or write)

MMAP-L = I/O map enable signal (DMA grant)

MABORT-L = abort signal

DEMAND = current microinstruction is a demand bus operation (i.e., bus write, data stream read, demand instruction stream read, or interrupt vector read)

READ = current microinstruction is a read (AIO code = 1100 or 10xx, i.e., RD, RMW, RDF, RDI, or Operate Prefetch (see section 4.2.1))

MMISS-L = cache miss signal

MBS-H<1> = bank select signal (cache bypass)

MBS-H<0> = bank select signal (cache force miss)

BSFF<1:0>= bank select flip-flops

If the wait state is entered, the current microcycle is extended beyond the normal four clock periods to a minimum of eight clock periods and increments of two clock periods. Starting at Tx25, the Data chip samples MCONT-L through a synchronizer flip-flop. If not asserted, the Data chip waits two clock periods and then resamples (next Tx25). If asserted, the Data chip deasserts MSCTL-L at the next Tx00 and resumes normal operation with the next Tx75 as T-25.

3.0 MEMORY MANAGEMENT UNIT (MMU)

The memory management unit is responsible for the relocation and protection of virtual addresses. It contains the MMU register file, the relocation data path, and various special registers and supporting logic.

3.1. MMU Register File

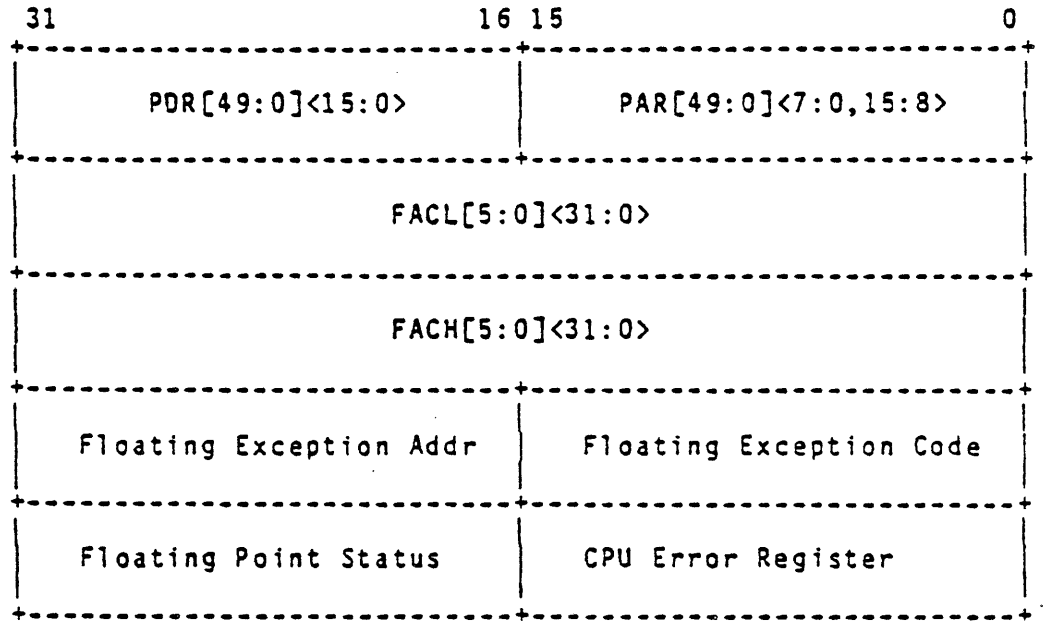
The MMU register file is physically organized 64 registers deep by 32 bits wide. It contains the memory management relocation registers, the floating point accumulators, the floating point exception and status registers, and the CPU Error Register.

The contents of an MMU register must be first moved to an EU register before any arithmetic or logical operation can be performed on it. External data cannot be read directly into an MMU register; instead, it must be read into an EU register and then moved to the destination register. An MMU register can be written directly to the external MDAL bus.

Internal data chip transfers, which are defined as data moves between MMU and EU registers, are only 16 bits wide; thus two separate microcycles are necessary to transfer an entire MMU register. The microcode can only address 16-bit words in the MMU register file; therefore, the register file appears to the microcode as 128 registers deep by 16 bits wide.

3.1.1 Physical Organization

The MMU register file consists of 50 memory management register pairs (PDRs and PARs), 6 floating point accumulator pairs (FACHs and FACLs), 3 floating point status registers, and the CPU Error Register.



3.1.1.1 Floating Point Registers

The floating point accumulators (FACH and FACL), Floating Point Exception Address Register (FEA), Floating Point Exception Code Register (FEC), and Floating Point Status Register (FPS) are used by the microcode to implement the floating point instruction set. The use and contents of these registers are the exclusive responsibility of the microcode. Their state at power up is UNDEFINED.

3.1.1.2 CPU Error Register

The CPU Error Register is used by the microcode to report CPU error conditions. Its use and contents are the exclusive responsibility of the microcode. The CPU Error Register can be read and written both explicitly as a memory location and by microcode. If written explicitly, all bits in the register are cleared. Its state at power up is UNDEFINED.

3.1.1.3 Relocation Registers

The memory management relocation registers consist of 50 Page Address Register (PAR)/Page Descriptor Register (PDR) pairs-- eight each for kernel instruction space, kernel data space, supervisor instruction space, supervisor data space, user instruction space, user data space, plus one PAR/PDR pair for use by the console microcode and one PAR/PDR pair for use when memory management is disabled:

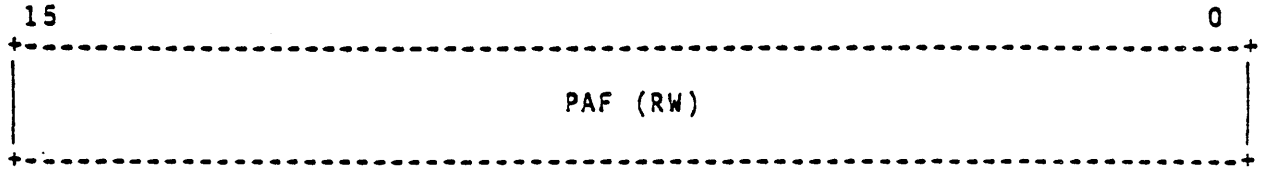
- 8 Kernel Mode Instruction Space Page Description Registers (KIPDR)
- 8 Kernel Mode Data Space Page Description Registers (KDPDR)
- 8 Kernel Mode Instruction Space Page Address Registers (KIPAR)
- 8 Kernel Mode Data Space Page Address Registers (KDPAR)
- 8 Supervisor Mode Instruction Space Page Description Registers (SIPDR)
- 8 Supervisor Mode Data Space Page Description Registers (SDPDR)
- 8 Supervisor Mode Instruction Space Page Address Registers (SIPAR)
- 8 Supervisor Mode Data Space Page Address Registers (SDPAR)
- 8 User Mode Instruction Space Page Description Registers (UIPDR)
- 8 User Mode Data Space Page Description Registers (UDPDR)
- 8 User Mode Instruction Space Page Address Registers (UIPAR)
- 8 User Mode Data Space Page Address Registers (UDPAR)
- 1 Console Mode Page Address Register (CPAR)
- 1 Console Mode Page Description Register (CPDR)
- 1 Non-Relocation Page Address Register (NPAR)
- 1 Non-Relocation Page Description Register (NPDR)

When the relocation registers are accessed for an address relocation operation, a 32-bit register pair is accessed (this permits the PAR and PDR to be available together for the relocation operation). When the relocation registers are accessed by the microcode or as explicit memory locations, only the register that was specifically addressed is accessed.

The formats of the PAR/PDR registers are depicted below.

3.1.1.3.1 Page Address Registers (PARs)

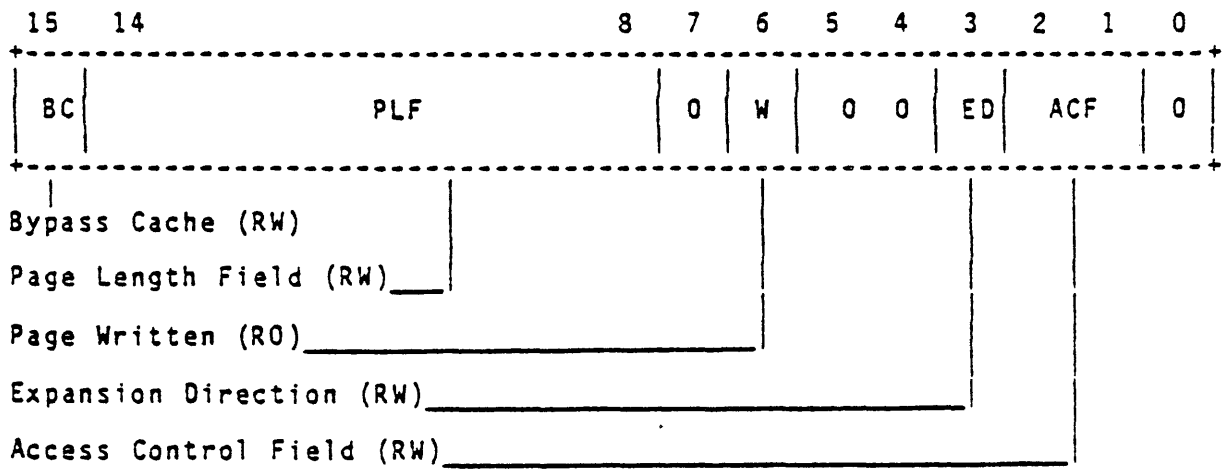
The Page Address Registers (PARs) contain a 16-bit displacement which is added to bits<12:6> of the virtual PC or the virtual address received from the Execution Unit to create part of the relocated physical address. All bits of these registers are implemented.



The state of these registers at power up is UNDEFINED.

3.1.1.3.2. Page Descriptor Registers (PDRs)

The Page Descriptor Registers (PDRs) contain information relative to page expansion, page length, and access control. Bits<7,5:4,0> are not writeable and always read as zeroes.



Bits	Name	Function
15	Bypass Cache (RW)	This bit implements a conditional cache bypass mechanism. If the PDR accessed during a relocation operation has this bit set, the time-multiplexed signal MBS-H<1> is asserted during the subsequent I/O cycle.

14:8 Page Length Field (RW)

This field specifies the block number which defines the page boundary. 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 is set when this PDR and its corresponding PAR are accessed during a relocation operation during an AW, AWI, or AWD microinstruction, regardless of any MMU or system abort. This bit is cleared to 0 whenever either the PDR or its associated PAR is modified (written into) by an explicit write and no abort or by a microcode write.

3 Expansion Direction (RW)

This bit specifies in which direction the page expands. If ED=0 the page expands upwards from block number 0 to include blocks with higher addresses; if ED=1, the page expands downwards from block number 127 to include blocks with lower addresses.

2:1 Access Control Field (RW)

This field contains the access code for this particular page. The access code specifies 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. Implemented codes are:

00 Non-resident - abort all accesses
01 Read only - abort on writes
10 Not used - abort all accesses
11 Read/write access

The state of these registers at power up is UNDEFINED.

3.1.2 MMU Register Access

All the registers in the MMU register file can be accessed by the microcode using INPR and OUTR microinstructions. In addition, the PAR/PDR registers (except for CPAR/CPDR and NPAR/NPDR) and the CPU Error Register can be accessed as explicit memory locations. Finally, PAR/PDR register pairs can be accessed for input to the MMU data path during relocation operations. In order to access registers in the MMU file, an eight-bit register access code (D<7:0>) is required:

bit <7> = multiplexor control (0 = no select, 1 = select bits<31:16>)

bit <6> = multiplexor control (0 = no select, 1 = select bits<15:0>)

bits <5:4> = processor mode select

bit <3> = instruction/data (I/D) space select

bits <2:0> = 1-of-8 select

The eight-bit access code is derived from the INPR/OUTR microinstruction during microcode access, from the explicit address logic during explicit access, and from the microinstruction and incoming virtual address during relocation access. Table 3-1 summarizes the derivation of the access code in different access modes.

Table 3-1
DERIVATION OF
ACCESS CODES

Access Bits	Code Name	Type of Access	Taken From
<7:6>	mux	microcode direct	spr<31:16>/spr<15:0> sel
		microcode indirect	spr<31:16>/spr<15:0> sel
		explicit	spr<31:16>/spr<15:0> sel
		relocation	spr<31:16>/spr<15:0> sel
<5:4>	mode	microcode direct	bits<10:9> of microinstruction
		microcode indirect	bits<10:9> of microinstruction
		explicit	PA<8>'-PA<6>
		relocation	forced to illegal mode if console or 16-bit relocation; otherwise selected mode (kernel/supervisor/ user)
<3>	I/D	microcode direct	bit<8> of microinstruction
		microcode indirect	bit<8> of microinstruction
		explicit	PA<4>
		relocation	forced to I space if console or 16-bit relocation; otherwise selected space (I/D)
<2:0>	1-of-8	microcode direct	bits<7:5> of microinstruction
		microcode indirect	(MIR<5>.SIR<2>'(MIR<5>.SIR<1> or MIR<6>.SIR<7> or MIR<7>.SIR<7>)' (MIR<5>.SIR<0> or MIR<6>.SIR<6> or MIR<7>)
		explicit	(PA<3> + CPU Error sel)'PA<2:1>
		relocation	VPC<15:13> if I stream relocation or VA<15:13> if D stream relocation or forced to 6 if 16-bit relocation or forced to 7 if console relocation

3.1.2.1 Microcode Access

All the registers in the MMU register file can be accessed directly from the microcode with an INPR or OUTR microinstruction. Table 3-2 lists the microinstruction codes for directly accessing the registers in the MMU file. Note that the control field codes in the table are precisely the eight-bit access codes required to access the register file.

In addition, the floating point accumulators can be indirectly accessed through the Shadow Instruction Register (SIR). Table 3-3 lists the microinstruction codes for indirectly accessing the floating point accumulators, and for accessing the special Data chip registers not in the EU or MMU files.

Table 3-2
INPR/OUTR CODES AND
EXPLICIT ADDRESSES FOR
MMU REGISTER FILE

Register	Explicit Address	INPR/OUTR Code <12:5>	Register	Explicit Address	INPR/OUTR Code <12:5>
KIPDR 0	17772300	000	KIPAR 0	17772340	100
KIPDR 1	17772302	001	KIPAR 1	17772342	101
KIPDR 2	17772304	002	KIPAR 2	17772344	102
KIPDR 3	17772306	003	KIPAR 3	17772346	103
KIPDR 4	17772310	004	KIPAR 4	17772350	104
KIPDR 5	17772312	005	KIPAR 5	17772352	105
KIPDR 6	17772314	006	KIPAR 6	17772354	106
KIPDR 7	17772316	007	KIPAR 7	17772356	107
KDPDR 0	17772320	010	KDPAR 0	17772360	110
KDPDR 1	17772322	011	KDPAR 1	17772362	111
KDPDR 2	17772324	012	KDPAR 2	17772364	112
KDPDR 3	17772326	013	KDPAR 3	17772366	113
KDPDR 4	17772330	014	KDPAR 4	17772370	114
KDPDR 5	17772332	015	KDPAR 5	17772372	115
KDPDR 6	17772334	016	KDPAR 6	17772374	116
KDPDR 7	17772336	017	KDPAR 7	17772376	117
SIPDR 0	17772200	020	SIPAR 0	17772240	120
SIPDR 1	17772202	021	SIPAR 1	17772242	121
SIPDR 2	17772204	022	SIPAR 2	17772244	122
SIPDR 3	17772206	023	SIPAR 3	17772246	123
SIPDR 4	17772210	024	SIPAR 4	17772250	124
SIPDR 5	17772212	025	SIPAR 5	17772252	125
SIPDR 6	17772214	026	SIPAR 6	17772254	126
SIPDR 7	17772216	027	SIPAR 7	17772256	127
SDPDR 0	17772220	030	SDPAR 0	17772260	130
SDPDR 1	17772222	031	SDPAR 1	17772262	131
SDPDR 2	17772224	032	SDPAR 2	17772264	132
SDPDR 3	17772226	033	SDPAR 3	17772266	133
SDPDR 4	17772230	034	SDPAR 4	17772270	134
SDPDR 5	17772232	035	SDPAR 5	17772272	135
SDPDR 6	17772234	036	SDPAR 6	17772274	136
SDPDR 7	17772236	037	SDPAR 7	17772276	137

Table 3-2 (continued)
INPR/OUTR CODES AND
EXPLICIT ADDRESSES FOR
MMU REGISTER FILE

Register	Explicit Address	INPR/OUTR Code <12:5>	Register	Explicit Address	INPR/OUTR Code <12:5>
FACL0<31:16>	na	040	FACL0<15:0>	na	140
FACL1<31:16>	na	041	FACL1<15:0>	na	141
FACL2<31:16>	na	042	FACL2<15:0>	na	142
FACL3<31:16>	na	043	FACL3<15:0>	na	143
FACL4<31:16>	na	044	FACL4<15:0>	na	144
FACL5<31:16>	na	045	FACL5<15:0>	na	145
NPDR	na	046	NPAR	na	146
CPDR	na	047	CPAR	na	147
FACH0<31:16>	na	050	FACH0<15:0>	na	150
FACH1<31:16>	na	051	FACH1<15:0>	na	151
FACH2<31:16>	na	052	FACH2<15:0>	na	152
FACH3<31:16>	na	053	FACH3<15:0>	na	153
FACH4<31:16>	na	054	FACH4<15:0>	na	154
FACH5<31:16>	na	055	FACH5<15:0>	na	155
FEA	na	056	FEC	na	156
FPS	na	057	CPU Error	17777766	157
UIPDR 0	17777600	060	UIPAR 0	17777640	160
UIPDR 1	17777602	061	UIPAR 1	17777642	161
UIPDR 2	17777604	062	UIPAR 2	17777644	162
UIPDR 3	17777606	063	UIPAR 3	17777646	163
UIPDR 4	17777610	064	UIPAR 4	17777650	164
UIPDR 5	17777612	065	UIPAR 5	17777652	165
UIPDR 6	17777614	066	UIPAR 6	17777654	166
UIPDR 7	17777616	067	UIPAR 7	17777656	167
UDPDR 0	17777620	070	UDPAR 0	17777660	170
UDPDR 1	17777622	071	UDPAR 1	17777662	171
UDPDR 2	17777624	072	UDPAR 2	17777664	172
UDPDR 3	17777626	073	UDPAR 3	17777666	173
UDPDR 4	17777630	074	UDPAR 4	17777670	174
UDPDR 5	17777632	075	UDPAR 5	17777672	175
UDPDR 6	17777634	076	UDPAR 6	17777674	176
UDPDR 7	17777636	077	UDPAR 7	17777676	177

Table 3-3
INPR/OUTR CODES FOR
INDIRECT FACS AND
SPECIAL REGISTERS

Register	Explicit Address	INPR/OUTR Code <12:5>
FACL[n]<31:16>, n = SIR<2:0>	na	241
FACL[n]<15:0> , n = SIR<2:0>	na	341
FACH[n]<31:16>, n = SIR<2:0>	na	251
FACH[n]<15:0> , n = SIR<2:0>	na	351
FACL[n]<31:16>, n = SIR<7:6>	na	242
FACL[n]<15:0> , n = SIR<7:6>	na	342
FACH[n]<31:16>, n = SIR<7:6>	na	252
FACH[n]<15:0> , n = SIR<7:6>	na	352
FACL[n]<31:16>, n = (SIR<7:6> OR 1)	na	244
FACL[n]<15:0> , n = (SIR<7:6> OR 1)	na	344
FACH[n]<31:16>, n = (SIR<7:6> OR 1)	na	254
FACH[n]<15:0> , n = (SIR<7:6> OR 1)	na	354
MMR0	17777572	200
MMR1	17777574	201
MMR3	17772516	202
PIRQ	17777772	203
Cache Control Register	17777746	204
Hit/Miss Register	17777752	205
Shift Register (SR)	na	206
Shadow Instruction Register (SIR)	na	207

3.1.2.2 PAR/PDR and CPU Error Register Explicit Access

Each PAR and PDR in the MMU register file (except for CPAR/CPDR and NPAR/NPDR), plus the CPU Error Register, has a unique address in the I/O page. Table 3-2 lists these addresses. During any relocation operation, the explicit address logic checks for a reference to an internal processor register. If it detects a reference to a PAR or PDR, or to the CPU Error Register, it supplies the appropriate 8-bit register access code to the MMU register file decoder and multiplexor. Read byte or word microinstructions then cause the contents of the accessed register to be sent through the I/O multiplexor to an EU register. Write byte or word microinstructions cause the contents of an EU register to be written into the addressed word or byte of the selected MMU register (the data is also written to MDAL-H). Note that any write (explicit or microcode) to a PAR or PDR clears the "W" bit in the selected PDR, while an explicit write to the CPU Error Register clears the entire register.

Once a reference to a PAR or PDR has been detected, the 8-bit register access code can be derived algorithmically from the physical address, as follows:

Table 3-4
ACCESS CODES FOR
EXPLICIT ACCESS

Register Access Code Bits	Meaning	Derived From
<7>	mux control	-PA<5>
<6>	mux control	PA<5>
<5>	mode select	PA<8>
<4>	mode select	-PA<6>
<3>	I/D select	PA<4>
<2:0>	1-of-8	(PA<3> OR CPU Error Register select)'PA<2:1>

Note that the special MMU registers do not conform to this algorithm.

3.1.2.3 PAR/PDR Relocation Access

The relocation registers are addressed as PAR/PDR pairs during address relocation operations. Instruction stream relocation occurs as the result of a prefetch microinstruction (RDI, RDF, Operate Prefetch (see section 4.2.1), RSYNC); data stream relocation as the result of an address class microinstruction. The 8-bit register access code consists of the following:

Table 3-5
ACCESS CODES FOR
RELOCATION ACCESS

Register Access Code Bits	Meaning	Derived From
<7:6>	mux control	forced to 0
<5:4>	mode select	forced to illegal mode if console or 16-bit mode relocation; otherwise selected mode
<3>	I/D select	forced to I space if console or 16-bit mode relocation; otherwise selected space
<2:0>	1-of-8	<15:13> of virtual address if data stream relocation <15:13> of virtual program counter if instruction stream relocation forced to 6 if 16-bit relocation forced to 7 if console relocation

With the exception of the PDR W bit, PAR/PDR contents cannot be altered by relocation references. The "W" bit is set when data is written into the memory page specified by a PAR/PDR pair (see section 3.1.1.3.2).

3.1.2.3.1 Mode Selection

In a relocation access, the mode select in the 8-bit MMU register access code is derived from the current microinstruction. For instruction stream relocation operations, the current mode (PS<15:14>) is always used. For data stream relocation operations, the mode is specified by control field bits<11:10> of the address microinstruction. These indirectly or directly select a PDP-11 processor mode (kernel/supervisor/user) or console mode:

Bits<11:10>	Access Option	PDP-11 Mode
00	Kernel	00
01	Previous	@PS<13:12>
10	Console	-
11	Current	@PS<15:14>

The kernel option directly defines the mode. The console option is used only by the console microcode and selects the console PAR/PDR registers. The previous and current options obtain the mode select from PS<13:12> and PS<15:14>, respectively. Note that if console mode is specified, 22-bit mapping is enabled and MMU aborts are suppressed for that relocation cycle, irrespective of the state of MMR0 and MMR3.

3.1.2.3.2 I/D Space Selection

In a relocation access, the I/D space select in the 8-bit MMU register access code is derived from the current microinstruction. For instruction stream relocation operations, the space select is always instruction (I) space. For data stream relocation operations, instruction space is selected unless control bits<6:5> of the address microinstruction specify data space, memory management is enabled, and MMR3 indicates that data space is enabled in the selected PDP-11 processor mode.

3.1.2.3.3 Virtual Address Selection

In a relocation access, the 1-of-8 select in the 8-bit MMU register access code is derived from bits<15:13> of the input virtual address. For instruction stream relocation operations, this is bits<15:13> of the prefetch mechanism's virtual PC (VPC<15:13>). For data stream relocation operations, this is bits<15:13> of the A-port operand of the microinstruction (brought over on FA<15:13>).

3.1.3 Register Decoder and Multiplexor

Access to the MMU register file is controlled by a register select decoder and an input/output multiplexor.

3.1.3.1 Register Select Decoder

The MMU register select decoder uses six bits of information (D<5:0>) to select a 32-bit register or register pair: mode select, I/D select, 1-of-8 select. See Table 3-1 for derivation of access codes.

3.1.3.2 Register File Multiplexor

The register file multiplexor uses two bits of information (D<7:6>) to decide whether to connect bits<31:16> or bits<15:0> of the selected MMU register to the MMU data bus. An input of 10 connects bits<31:16>, of 01 connects bits<15:0>, and of 00 or 11 connects neither. Note that during a relocation operation, the selected PAR/PDR pair is read out directly to the MMU data path as a 32-bit quantity.

3.1.4 Console PAR/PDR Registers (CPAR/CPDR)

These registers are used to form physical addresses for console access. They do not have an explicit address. They can be selected by an address microinstruction which specifies console mode. The format of the CPAR is identical to every other PAR. Note that specification of the console PAR/PDR enables 22-bit mapping and suppresses MMU aborts for that relocation cycle, irrespective of the state of MMR0 and MMR3.

CPDR can be used as a scratch register by the microcode. A one in bit 15 of the CPDR will not force a cache bypass. (See Sec. 3.1.1.3.2.)

3.1.5 Non-Relocation PAR/PDR Registers (NPAR/NPDR)

These registers are used to form physical addresses in 16-bit mapping mode (MMR0<0> = 0 and not console mode). They do not have an explicit address. They are selected by a relocation operation in 16-bit mode. NPAR has the following format:

NPAR<15:10> = 0

NPAR<9:7> = 1-of-8 select (FA<15:13> or VPC<15:13>)

NPAR<6:0> = 0

NPDR bits <15,9:8> are hardwired to zeroes and must not be written to with ones. The other bits are read/write bits, which can be used by the microcode.

NPAR is read-only and is restricted from being written by the microcode.

3.2 MMU Data Path

The principal function of the Memory Management Unit is to map 16-bit virtual addresses into physical addresses. Memory management takes as inputs a 16-bit virtual address (VA), a 2-bit memory management mode, a 1-bit instruction/data (I/D) space select, and the memory management control bits from MMR0 and MMR3. It produces as outputs a 22-bit physical address, two bits of bank select status, three bits of abort status, one bit of cache bypass status, and other information. Relocation is initiated by the following microinstructions:

- Instruction stream prefetch (RDI, RDF, Operate Prefetch (see section 4.2.1)) or resynchronization (RSYNC).

Inputs:	virtual address	<-- VPC
	mode	<-- PS<15:14>
	space	<-- I
Outputs:	PPC	<-- physical address
	PCS<1>	<-- bypass status
	PCS<3:2>	<-- bank select status
	PCS<0>	<-- OR of abort status
	MMR0<15:13>	<-- abort status if demand cycle.

- Data stream relocation (AR, ARI, ARD, AW, AWI, AWD).

Inputs:	virtual address	<-- Ra register
	mode	<-- specified by microinstruction
	space	<-- specified by microinstruction and I/D space logic
Outputs:	output latch	<-- physical address
	bypass latch	<-- bypass status
	bank select latches	<-- bank select status
	abort latch	<-- OR of abort status
	MMR0<15:13>	<-- abort status.

Mapping of virtual addresses is performed in one of three modes: 16-bit, 18-bit, or 22-bit mapping. Except in console access, MMR0<0> and MMR3<4,2:0> control address relocation. MMR0<0> enables relocation. When MMR0<0> = 0, 16-bit mapping is performed and MMU aborts are suppressed. When MMR0<0> = 1, MMR3<4> selects between 18-bit and 22-bit mapping, and MMR3<2:0> enables data (D) space in each of the processor modes. Console access forces 22-bit mapping and suppresses MMU aborts.

16-bit mapping calculates a physical address as follows:

```
PA<5:0>   = VA<5:0>
PA<15:6>  = VA<12:6> PLUS NPAR<9:0>
PA<21:16> = 111111 if PA<15:13> = 111, 000000 otherwise
```

18-bit mapping calculates a physical address as follows:

```
PA<5:0>   = VA<5:0>
PA<17:6>  = VA<12:6> PLUS PAR<11:0>
PA<21:18> = 1111 if PA<17:13> = 11111, 0000 otherwise
```

22-bit mapping calculates a physical address as follows:

PA<5:0> = VA<5:0>
PA<21:6> = VA<12:6> PLUS PAR<15:0>

The Memory Management data path consists of the following components:

- Adder input multiplexor/latch
- Address adder
- Page length comparator
- Abort logic
- Address error logic
- Address adjust and detect logic.

3.2.1 Adder Input Multiplexor/Latch

The adder input multiplexor/latch selects and latches the virtual address to be relocated. In instruction stream relocation operations, the virtual PC (VPC) is selected for input; in data stream operations, the EU register specified by the microinstruction Ra field is selected for input. The latch is an input to both the address adder and the page length comparator.

3.2.2 Address Adder

The adder produces the high order 16 bits of the relocated address from the sum of the 16-bit PAF of the selected PAR register and bits<12:6> of the virtual address latched in the input latch. The output of the adder is fed to the address adjust and detect logic.

3.2.3 Page Length Comparator

The 7-bit comparator determines if the allocated page length is exceeded during a relocation operation by comparing the desired block number (BN, bits<12:6> of the latched virtual address) with the PLF (PDR<14:8>) for the page being accessed (see Table 3-6 below). With ED=0 (expansion is permitted in an upward direction), the comparison is performed by subtracting the PLF+1 from the block number with any resulting carry representing a page length error. With ED=1 (lower expansion), the BN+1 is instead subtracted from the PLF with any resulting carry representing a page length error. The comparator results are fed to the memory management abort logic. Note that only the carry portion of the comparator is mechanized since the actual difference is not used.

Table 3-6
PAGE LENGTH ERROR DERIVATION

Upward Expansion	Downward Expansion
ERROR => PLF < BN	ERROR => BN < PLF
=> (PLF+1) =< BN	=> (BN+1) =< PLF
=> BN-(PLF+1) compare resulting in an ALU carry	=> PLF-(BN+1) compare resulting in an ALU carry

NOTE: The above is actually a 1's complement subtraction of BN-PLF or PLF-BN.

3.2.4 Memory Management Abort Logic

A memory management abort occurs if memory management is enabled (MMR0<0> = 1), the access mode is not console, and one or more of the following conditions occurs:

a. Attempted access in the illegal processor mode (PS<15:14> or PS<13:12>, as selected, = 10)

b. Attempted access of a non-resident page (PDR<2:1> = 00 or 10)

c. Page length error (detected by the page length comparator)

d. Write protect violation (PDR<2:1> = 01 during write or read-modify-write operations)

If a memory management abort occurs, the Data chip asserts the abort signal (MABORT-L) at T-25. In addition, if, during a demand bus operation, MMR0<15:13> = 000, the abort status is recorded in MMR0<15:13>:

- Non-existent mode or non-resident page: MMR0<15> = 1
- Page length error: MMR0<14> = 1
- Write protect violation: MMR0<13> = 1

If, on the other hand, a memory management abort occurs during an instruction stream request bus operation, the abort outputs are ORed together and latched in the prefetch control status latch (PCS<0>):

- Non-existent mode/non-resident page OR page length error: PCS<0> = 1

(A write protect violation cannot occur.)

3.2.5 Address Error Logic

This logic detects an address error. If the cycle is an instruction stream read, then an odd value in the physical PC or a reference to an internal or board register causes an address error. Since BS<0> is or'd into PPC<0>, this logic only has to detect for PPC<0> = 1. If the cycle is an address microinstruction with control bit<8> = 0, an odd value in the relocated physical address causes an address error during the next microcycle.

The Data chip responds to address errors by asserting signal MABORT-L. Timing is the same as for a memory management abort. The memory management status registers are unaffected.

3.2.6 Address Adjust and Detect Logic

The address adjust and detect logic receives bits<21:6> of the physical address from the address adder, and bits<5:0> directly from the input latch. It adjusts the physical address according to the mapping mode (16-bit, 18-bit, or 22-bit), as described in section 3.2. It then classifies the address into one of four categories:

- An internal Data chip register is a register in the Data chip which responds to a memory address. These are: the PS and MMR2 (see section 2.1), the CPU Error Register (see section 3.1.2.2), the PARs and PDRs except for CPAR/CPDR and NPAR/NPDR (see section 3.1.2.3), MMR0, MMR1, MMR3, PIRQ, and the Hit/Miss Register (see section 3.3).

- A system board register is a register which is partially or completely implemented in the external System Interface. The list of system board registers is given in Table 3-7.

Table 3-7
BOARD REGISTERS

Name	Address	Function
Maintenance Register	17777750/1	system maintenance
Cache Control Register	17777746/7	cache control
Memory System Error Register	17777744/5	memory system status
High Error Address Register	17777742/3	latches physical address <21:16> on error
Low Error Address Register	17777740/1	latches physical address <15:0> on error

Note that the Cache Control Register is classified as a system board register, not as an internal Data chip register.

- An I/O address is any location in the I/O page (addresses 17760000 - 17777777 octal) which is not an internal Data chip register or a system board register.

- A memory address is any location not in the I/O page.

If the address references a register in the MMU register file, the address detect logic generates the 8-bit register access code to the MMU register file decoder and multiplexor. If the address references an internal Data chip register not in the MMU file, this logic generates an appropriate select. In any case, this logic generates a two bit bank select as follows:

type	BS<1>	BS<0>
memory address	0	0
system board register	0	1
I/O address	1	0
internal Data chip register	1	1

3.2.7 Usage of Results

The results of a relocation operation are stored as follows:

MMU output	Microinstruction is RDI, RDF, RSYNC, Operate Prefetch (see section 4.2.1)	AR, ARI, ARD, AW, AWI, AWD
physical address	PPC<21:0> + BS<0>	OL<21:0>
bank select status	PCS<3:2>	bank select latches
bypass status	PCS<1>	bypass latch
OR of abort status	PCS<0>	abort latch
abort status	MMR0<15:13> if demand	MMR0<15:13>

At T-25 of an I/O cycle, these results are driven to the outside world:

signal	Previous microinstruction was not (address class or RMW)	Previous microinstruction was address class or RMW
MDAL-H<21:0>	PPC<21:0>	OL<21:0>
MBS-H<1:0>	PCS<3:2>	bank select latches
MMAP-L	MMR3<5>	MMR3<5>
MABORT-L	PCS<0> + PPC<0>	abort latch + OL<0>.word operation

At T75 of an I/O cycle, further information is driven out:

signal	Previous microinstruction was not (address class or RMW)	Previous microinstruction was address class or RMW
MDAL-H<21:0>	set up for input	set up for input or output
MBS-H<1>	PCS<1> + CCR<9>	bypass latch + CCR<9> + (bit<12> of RMW = 0)
MBS-H<0>	CCR<3> + CCR<2>	CCR<3> + CCR<2>
MMAP-L	DMA grant	DMA grant
MABORT-L	sustained	sustained

3.3 MMU Special Registers

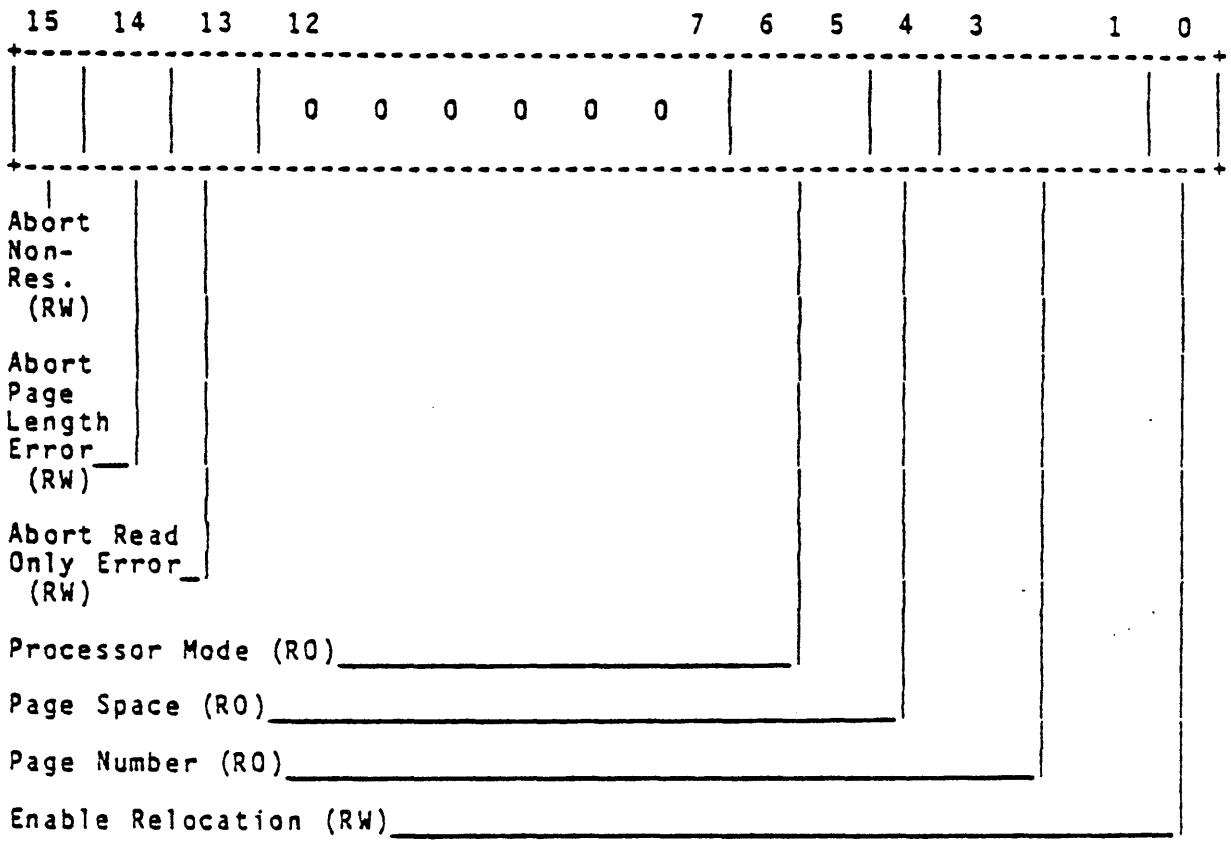
This section describes the explicitly addressable registers in the MMU which require special supporting logic. These are the MMU status registers (MMR0, MMR1, MMR3), the Program Interrupt Request Register (PIRQ), the Cache Control Register (CCR), and the Hit/Miss Register. Each has some unique hardware associated with it. All special purpose registers can be read with an explicit access by byte or word, and some can be written by byte or word via the same mechanism. On explicit writes to these registers, the data bits come out in the correct order on the MDAL-H. In addition to this, they can change independently of the microcode via special hardware. Table 3-8 lists the special MMU registers.

Table 3-8
SPECIAL MMU REGISTERS

Register	I/O Address	Special Supporting Hardware
MMR0	17777572/3	Derive MMU control and record MMU status
MMR1	17777574/5	Record register changes
MMR3	17772516/7	Additional MMU control
PIRQ	17777772/3	Priority encode bits<15:9> to bits<7:5> and bits<3:1>
Cache Control	17777746/7	Cache control logic
Hit/Miss	17777752/3	Record cache hit/miss info

3.3.1 MMR0

MMR0 contains abort flags and other status flags. The format is as follows:



Bits<15:13> are the abort flags. They are set by the memory management abort logic during demand bus operations (see section 3.2.4). Bits<15:13> can also be explicitly written (this does not cause an abort). If bits<15:13> are non-zero, the Memory Management Unit freezes the contents of MMR0<15:13,6:1>, MMR1, and MMR2, although MMR0<15:13> may still be explicitly written.

Bits	Name	Function
15	Abort - Non-Resident (RW)	Bit<15> is set by attempting to access a page with an Access Control Field code of 00 or 10. It is also set by a relocation operation in the illegal mode (PS<15:14> or PS<13:12>, as selected, = 10).

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 an access code of 01.

Bits<6:1> are altered only if:

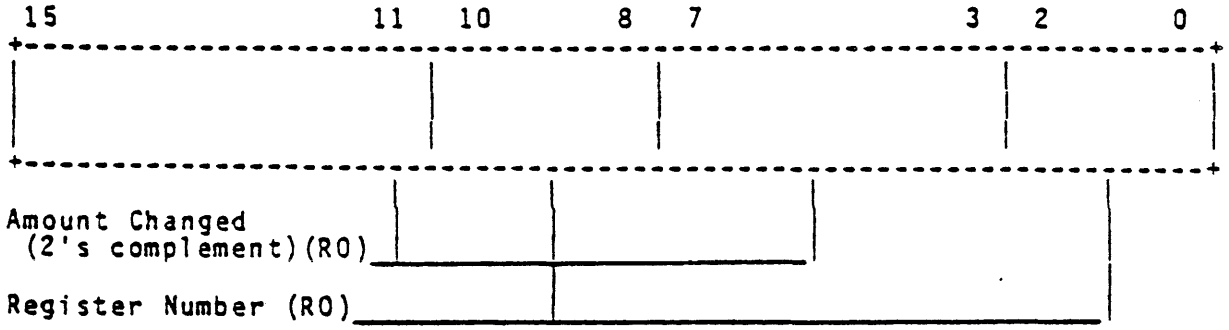
- the microcycle is a demand bus cycle;
- MMR0<15:13> = 000;
- MMR0<0> = 1;
- address microinstruction bit<7> = 1;
- address microinstruction bits<11:10> <> 10.

6:5	Processor Mode (R0)	Bits<6:5> capture the CPU mode (kernel/supervisor/user/illegal) of the current relocation operation: kernel = 00, supervisor = 01, user = 11, illegal Mode = 10.
4	Page Space (R0)	Bit<4> indicates the address space (I or D) of the current relocation operation: 0 = I Space, 1 = D Space.
3:1	Page Number (R0)	Bits<3:1> contain the page number of the current relocation operation.
0	Enable Relocation (RW)	Bit<0> is the "Enable Relocation" bit. When it is set to 1, all addresses are relocated normally by the Memory Management Unit. When bit 0 is set to 0, the MMU is effectively disabled; relocation is performed using the NPAR/NPDR register pair, and MMU aborts are suppressed. Note that if console access is selected for relocation, 22-bit mapping is enabled and MMU aborts are suppressed for that relocation cycle.

MMR0 is UNDEFINED at power up.

3.3.2 MMR1

MMR1 automatically records any autoincrement or autodecrement of the general registers. This register supplies information needed to recover from a Memory Management abort by allowing programmed register back up.



If MMR0<15:13> are non-zero, the contents of MMR1 are frozen. If MMR0<15:13> = 000, then receipt of signal MPRDC-L (start of macroinstruction) or an RDF microinstruction clears the register. Thereafter, selection of the half (bits<7:0> vs. bits<15:8>) of MMR1 which is updated, when it is not frozen, toggles between the two halves, starting with the low half (bits<7:0>). Therefore, source operand register changes will always be recorded in bits<7:0>; however, destination operand register changes may be recorded in either half of MMR1 (depending on the mode of the source operand and the instruction type).

MMR1 allows only single word or single byte operand register changes to be recorded. The recording action is activated by an address relocation microinstruction with control bit<12> = 0. Depending on the microinstruction.opcode and length field, the following (binary) constants are produced:

Microinstruction	Bit<14> of Address Microinstruction	Constant
ARI,AWI	1	00001
ARI,AWI	0	00010
ARD,AWD	1	11111
ARD,AWD	0	11110

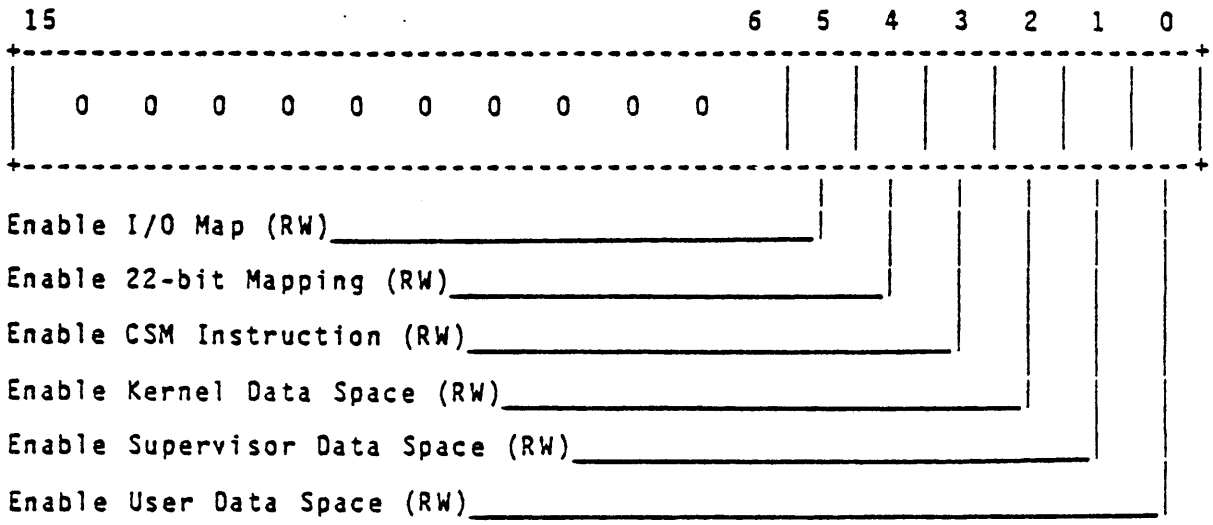
MMR1 cannot be used in the MARK instruction or the floating point or commercial instruction sets due to the limited number of constants. It is the responsibility of the microcode to guarantee that these macroinstructions do not modify the general registers until all operand references are finished.

MMR1 is UNDEFINED at power up.

3.3.3 MMR3

3.3.3.1 Register Format

MMR3 enables D space, selects between 22-bit and 18-bit mapping, enables the call to supervisor macroinstruction (CSM), and enables the I/O map (when applicable).



MMR3 is UNDEFINED at power up.

3.3.3.2 I/D Space Logic

The I/D space logic controls the selection of instruction versus data space during data stream relocation operations. All instruction stream relocation operations select instruction space. A data stream relocation operation selects data space only if all of the following conditions are met:

- The microinstruction specifies data space relocation (address class microinstruction with control bits<6:5> as follows):

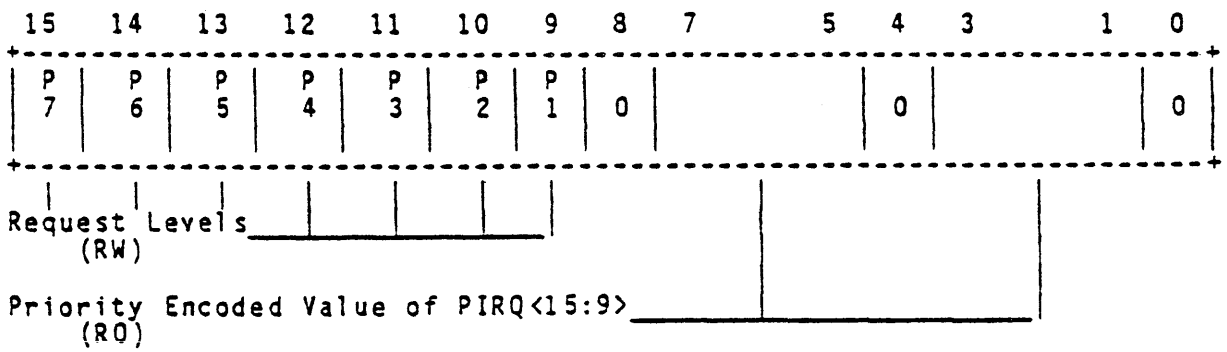
  bits<6:5> = 00, never (always instruction space)
  = 01 and PS<15:12> = 1111
  = 10, always
  = 11, unless Ra resolves to PC.

- Memory management is enabled (MMR0<0> = 1).

- Data space is enabled for the selected memory management mode.

3.3.4 Programmed Interrupt Request Register (PIRQ)

This register exists solely for explicit PIRQ register reads. The Control chips maintain a copy of this register which is used for interrupt generation.



Bits<15:9> are explicitly readable and writeable while bits<7:5> and bits<3:1> are only readable. Both <7:5> and <3:1> contain the three bit priority encoded value of bits<15:9>. The Data chip hardware priority encodes bits<15:9> and forces the result into both <7:5> and <3:1>, as shown in Table 3-9.

Table 3-9
PIRQ PRIORITY ENCODING

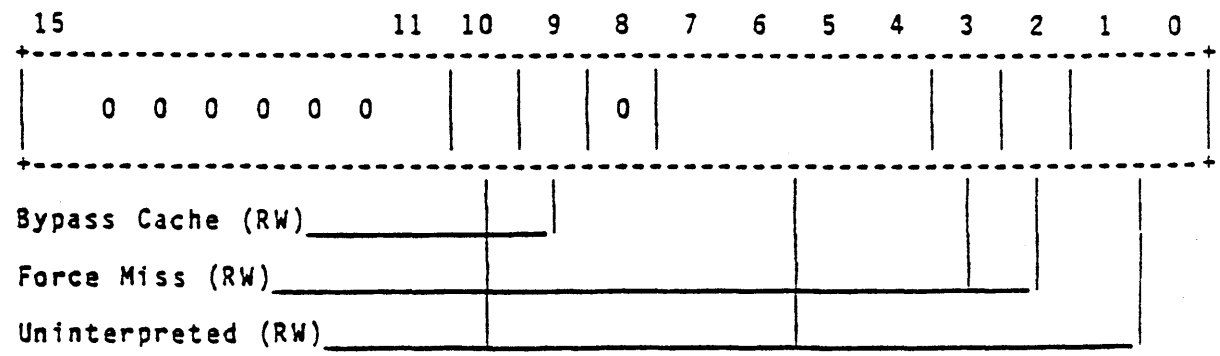
PIRQ<15:9>	PIRQ<7:5> and <3:1>
1XXXXXX	111
01XXXXX	110
001XXXX	101
0001XXX	100
00001XX	011
000001X	010
0000001	001

PIRQ is UNDEFINED at power up.

3.3.5 Cache Control Register (CCR)

3.3.5.1 Register Format

The Data chip implements bits<10:9,7:0> of the Cache Control Register as read/write and bits<15:11,8> as read-only 0. Bits<15:10,8:4,1:0>, however, are not interpreted in the Data chip and exist only for supplying data during Cache Control Register reads. Refer to the Programmer's Reference for the system definition of these bits. Bit<9> is used in deriving the time-multiplexed cache bypass signal (MBS-H<1>). Bits<3:2> drive the time-multiplexed force miss signal (MBS-H<0>). Any write to the CCR will assert the kill prefetch buffer signal (MKPB-L) at T75.



The Cache Control Register is UNDEFINED at power up.

3.3.5.2 Cache Control Logic

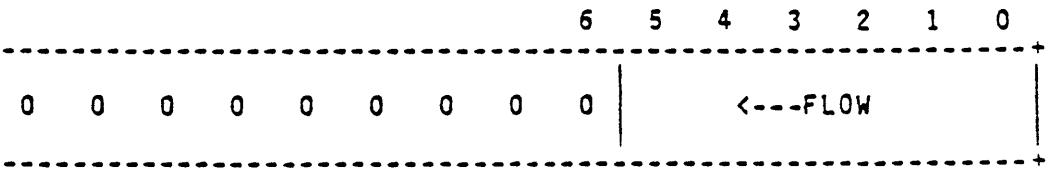
The cache control logic determines whether an I/O cycle will bypass the cache. In an instruction stream read, if the cache bypass bit is set (CCR<9> = 1), or if the PDR used to establish the PPC specified a cache bypass (PDR<15> = 1, latched in PCS<1>), then the cache control logic asserts MBS-H<1> in the second half of the I/O cycle. In a data stream read or write, if the cache bypass bit is set (CCR<9> = 1), or if the PDR used to establish the physical address specified a cache bypass (PDR<15> = 1, latched in the bypass latch), or if the read-modify-write microinstruction specifies a read lock, then the cache control logic asserts MBS-H<1> during the second half of the I/O cycle.

The cache control logic also determines whether an I/O cycle will force a cache miss. If either of the cache force miss bits is set (CCR<3> = 1 or CCR<2> = 1), then the cache control logic asserts signal MBS-H<0> during the second half of the I/O cycle.

Finally, on any write to the CCR, MKPB-L is asserted.

3.3.6 Hit/Miss Register

The Hit/Miss Register records the status of the MMISS-L signal at T150 in the six most recent read (all external I/O microinstructions including RDINTR and RDG), write to memory (i.e., only bus writes not GP writes), or Operate Prefetch (see section 4.2.1) microcycles. A hit is recorded as a 1, a miss as a zero. The Hit/Miss Register acts as a shift register; hit/miss data enters at bit <0> and is shifted leftward on successive memory references.



The Hit/Miss Register is read-only and is UNDEFINED at power up.

3.4 Stack Limit Logic

The stack limit logic protects against illegal kernel stack references. An illegal stack reference is defined as any stack reference at a virtual address less than 400(8). This logic is activated when an address microinstruction has control bit<9> = 0, R6 is selected, and the selected mode is kernel mode. If an illegal kernel stack reference is detected, the stack limit logic asserts the Stack Overflow signal (MSOV/JA-L) during the next microcycle for interpretation by the Control chips.

4.0 PREFETCH MECHANISM

The J-11 gets much of its performance from its instruction stream prefetch and parallel decode (predecode) mechanisms. The principal benefit is that instruction stream memory references are overlapped with internal operations, and the need for explicit instruction fetch and decode cycles is minimized.

The Data chip components of the prefetch mechanism consist of the following:

- VPC<15:0> (Virtual PC). This is a copy of the PC in the prefetch mechanism. During steady state, the VPC contains an incremented value of the PC. Whenever the PC is written (byte or word), the VPC is also written with the word value. The VPC can be independently incremented (by two).

- PPC<21:0> (Physical PC). The PPC is the physical address of the next word to be prefetched. PPC<0> is used to detect instruction stream address errors. BS<0> is or'd into PPC<0> in order to abort on I-stream reads from an internal or board register.

- PCS<3:0> (PPC Status). PCS<0> contains the OR of the volatile relocation abort information associated with the PPC (abort non-resident and page length error). PCS<1> contains bit<15> of the PDR accessed in the generation of PPC. PCS<3:2> contain the bank select status associated with the PPC. Once PCS<0> is set, PCS, PPC, and VPC are frozen until a RSYNC microinstruction is executed.

- IL[1:0]<15:0> (Input Latches). These latches are connected in parallel and are used on an alternating basis as the prefetch buffer and the flow-through input latch. The latch selected by PBFF is called the prefetch buffer; the deselected latch is the data latch. On a data stream read or RDI demand, the data latch is used as flow-through holding latch. On a RDF, Operate Prefetch (see section 4.2.1), or RDI request, PBFF is toggled; the old prefetch buffer supplies data to the A-bus, and the prefetch data is stored in the new prefetch buffer.

- PBFF (Prefetch Buffer Flip-Flop). This flip-flop selects which input buffer is being used as the PB. When clear, IL[0] is selected as the PB.

- PBA<5:1> (PB Address). This register contains bits<5:1> of the physical address of the data in the PB. It is used to detect overwrites of the PB data.

- PV (PB Valid). When set, PV indicates that the PB contains valid data.

4.1 Prefetch Operation

4.1.1 Normal Operating State

The prefetch mechanism is in steady state when the four level deep prefetch pipeline is full. In this state, the IR contains the macroinstruction being executed, the PB contains the data from the memory location pointed at by the PC, the PPC addresses the next instruction stream word, the VPC points to one word past the PPC, and PV is set.

Pipeline Level	State	
1	IR	<-- M[Relocated(PC at last predecode)]
2	PB	<-- M[Relocated(PC)]
	PV	<-- 1
3	PPC	<-- Relocated(PC PLUS 2)
	PCS<0>	<-- 0
4	VPC	<-- PC PLUS 4

The normal state of the prefetch mechanism can be disturbed in several ways. Some of these prefetch faults are detected by the hardware; others must be detected in the microcode.

4.1.2 Hardware Detected Prefetch Faults

The following conditions invalidate the prefetch buffer and are detected by the Data Chip hardware:

- Explicit writes to PS
- Explicit writes to the MMU registers
- Instruction stream overwrites
- Writes to PC

Explicit writes to the PS or the MMU registers (PARs, PDRs, MMR0, MMR3, CCR) occur during write operations to the I/O page. This is detected by the explicit address logic.

Instruction stream overwrites occur during a write operation when the operand's address is equal to the PB data address (PA<5:1> = PBA<5:1>).

Writes to the PC occur during operate, literal, address, or I/O microinstructions.

Whenever the Data chip detects one of the above conditions, it invalidates the prefetch buffer by clearing PV. It also asserts signal MKPB-L during the microcycle.

PV is set on any prefetch (unless RDI and PV=0) that does not result in an abort. This is accomplished by clocking PV at T50 during prefetch cycles and at Tx75 during all stretched prefetch cycles with the following equation:

PV = -(MABORT-L + (RDI . -PV)).

Note that in this case invalidation of the prefetch buffer does not cause MKPB-L to be asserted.

4.1.3 Microcode Detected Prefetch Faults

The microcode must assure that the following prefetch mechanism restrictions are met:

- A microinstruction which overwrites the PC must not start a relocation operation (can not be an Operate Prefetch).
- A microinstruction which overwrites the PC must not attempt a decode operation in the next microcycle.
- A microinstruction which overwrites the PC must not issue further RDI microinstructions without resynchronizing.

The situations in which difficulties are likely to occur are these:

- Conditional branches
- Unconditional branches and jumps
- Source modes -(PC) and @-(PC)
- Destination modes PC, (PC)+, -(PC), and @-(PC)
- Operations on the register pair SP'PC
- Traps and interrupts and related instructions

It is the responsibility of the microcode to resynchronize the instruction stream when these conditions occur.

4.2 Prefetch Microinstructions

There are five microinstructions which control the prefetch mechanism:

- Operate Prefetch
- RDI (Read Instruction)
- RDF (Read Demand)
- RSYNC (Resynchronization)

4.2.1 Operate Prefetch

An Operate Prefetch is an operate microinstruction with control bit<12>=0 and PV=1. It initiates a prefetch operation and is principally used to refill the prefetch buffer when predecode is issued.

If PV is 0, no prefetch operation takes place. If PV is 1, during the first half of the microcycle a request bus cycle is started using the PPC as the physical address, and the VPC is routed to the memory management unit (MMU) in order to generate the next PPC and PCS. If PCS<0> (relocation error) or PPC<0> (address error) is set at the beginning of the cycle, MABORT-L is asserted at the start of the microcycle to indicate that an error occurred when the PPC was established.

At the end of the cycle, the data is latched into the PB, and PV is set from MABORT-L. This conveniently sets PV to reflect the validity of the PB. That is, MABORT-L conveys whether a relocation, address, or memory system error occurred during the operation. Furthermore, if no abort occurred, the output of the MMU is latched into the PPC and PCS, thus establishing the new prefetch values, and VPC is incremented by two. BS<0> (new PCS<2>) is or'd into PPC<0> in order to force aborts on I-stream fetches from internal or board registers. In summary:

First Half		Second Half	
MDAL	<-- PPC	PB	<-- MDAL
MABORT-L	<-- PPC<0> + PCS<0>	PV	<-- -MABORT-L
MMU	<-- VPC	*! VPC	<-- VPC PLUS 2
MBS<1:0>	<-- PCS<3:2>	* PPC	<-- MMU
		** PCS<3:0>	<-- MMU status
		# MBS-H<1>	<-- PCS<1> + CCR<9>

* - Not executed if MABORT-L asserted at end of microcycle.
** - Not executed if MABORT-L asserted by system interface.
! - Not executed if MMU status non-zero (MMU abort) at end of microcycle.
- PCS<1> is the original value before it has been changed by MMU status.

4.2.2 Read Instruction Stream (RDI)

This microinstruction reads instruction stream operands for mode 27, 37, 6x, and 7x macroinstructions. Two sequences are possible during the execution of the RDI. A request bus cycle occurs when the contents of the PB are valid prior to the start of the cycle, and a demand cycle occurs if the contents of the PB are not valid. The PV flag is used to distinguish between the two situations. Regardless of how RDI is interpreted, the PC (R7) is always incremented by two.

A request RDI behaves similarly to the Operate Prefetch; however, it also includes writing the prefetch buffer into the EU register file. This enables the previous prefetched data to be written into the file while new data is fetched to the alternate input latch.

A demand RDI, on the other hand, transfers the bus data directly into the register file. Since it is a demand cycle, any abort condition alters the state of the machine. In summary:

RDI request, i.e., PV = 1:

First Half		Second Half	
			Ra <-- PB
			new PB <-- MDAL
MDAL	<-- PPC		PV <-- -MABORT-L
MABORT-L	<-- PPC<0> + PCS<0>	*!	VPC <-- VPC PLUS 2
MMU	<-- VPC	*	PPC <-- MMU
MBS<1:0>	<-- PCS<3:2>	**	PCS<3:0> <-- MMU status
		#	MBS-H<1> <-- PCS<1> + CCR<9>
			PC <-- PC PLUS 2

RDI demand, i.e., PV = 0:

First Half		Second Half	
MDAL	<-- PPC		
MABORT-L	<-- PPC<0> + PCS<0>		Ra <-- MDAL
MMU	<-- VPC	*!	VPC <-- VPC PLUS 2
MBS<1:0>	<-- PCS<3:2>	*	PPC <-- MMU
		**	PCS<3:0> <-- MMU status
		#	MBS-H<1> <-- PCS<1> + CCR<9>
			PC <-- PC PLUS 2
			MMR0<15:13> <-- MMU status
			PV unchanged

* - Not executed if MABORT-L asserted at end of microcycle.
** - Not executed if MABORT-L asserted by system interface.
! - Not executed if MMU status non-zero (MMU abort) at end of microcycle.
- PCS<1> is the original value before it has been changed by MMU status.

4.2.3 Read Demand Prefetch (RDF)

RDF forces a demand prefetch bus cycle. It typically follows a RSYNC microinstruction in order to refill the pipeline. During the first half of the microcycle, a demand bus cycle is started using the PPC as the physical address, and the VPC is routed to the memory management unit (MMU) in order to generate the next PPC and PCS. If PCS<0> (relocation error) or PPC<0> (address error) is set at the beginning of the cycle, MABORT-L is asserted at the start of the microcycle to indicate that an error occurred when the PPC was established.

At the end of the cycle, MMR2 is updated with the PC (provided MMR0<15:13> = 000), the data is latched into the PB, and PV is set from MABORT-L. This conveniently sets PV to reflect the validity of the PB. That is, MABORT-L will convey whether a relocation, address, or memory system error occurred during the operation. Furthermore, if no abort occurred, the output of the MMU is latched into the PPC and PCS, thus establishing the new prefetch values, and VPC is incremented by two. BS<0> (new PCS<2>) is or'd into PPC<0> in order to force aborts on I-stream fetches from internal or board registers. In summary:

First Half		Second Half	
MDAL	<-- PPC §Demand cycle†	PB	<-- MDAL
MABORT-L	<-- PPC<0> + PCS<0>	PV	<-- -MABORT-L
MMU	<-- VPC	*! VPC	<-- VPC PLUS 2
MBS<1:0>	<-- PCS<3:2>	* PPC	<-- MMU
		** PCS<3:0>	<-- MMU status
		# MBS-H<1>	<-- PCS<1> + CCR<9>
		MMR0<15:13>	<-- MMU status
		MMR1	<-- 0
		MMR2	<-- PC

* - Not executed if MABORT-L asserted at end of microcycle.
** - Not executed if MABORT-L asserted by system interface.
! - Not executed if MMU status non-zero (MMU abort) at end of microcycle.
- PCS<1> is the original value before it has been changed by MMU status.

4.2.4 Resynchronization (RSYNC)

RSYNC re-establishes PPC and PCS after a disturbance to the prefetch pipeline. Two sequences are possible during the execution of the RSYNC. A prefetch request occurs if control bit<12> = 0 and PV = 1. Otherwise just a resynchronization occurs (relocation with no bus cycle).

During the first half of the cycle, VPC is routed to the MMU. At the end of the cycle, the output of the MMU is latched into the PPC and PCS, and VPC is incremented by two. BS<0> (new PCS<2>) is or'd into PPC<0> in order to force aborts on I-stream fetches from internal or board registers. In summary:

CF<12>=1 OR PV=0:

First Half			Second Half	
MMU	<-- VPC	!	VPC	<-- VPC PLUS 2
			PPC	<-- MMU
			PCS<3:0>	<-- MMU status

! Not executed if MMU status non-zero (MMU abort) at end of microcycle.

CF<12>=0 AND PV=1:

First Half			Second Half	
MDAL	<-- PPC		PB	<-- MDAL
MABORT-L	<-- PPC<0> + PCS<0>		PV	<-- -MABORT-L
MMU	<-- VPC	*!	VPC	<-- VPC PLUS 2
MBS<1:0>	<-- PCS<3:2>	*	PPC	<-- MMU
		**	PCS<3:0>	<-- MMU status
		#	MBS-H<1>	<-- PCS<1> + CCR<9>

* - Not executed if MABORT-L asserted at end of microcycle.
** - Not executed if MABORT-L asserted by system interface.
! - Not executed if MMU status non-zero (MMU abort) at end of microcycle.
- PCS<1> is the original value before it has been changed by MMU status.

4.2.5 Resynchronization Sequences

Two different microcode sequences are necessary for resynchronizing and refilling the prefetch pipeline. The first both resynchronizes and refills the pipeline and must be used whenever the VPC, PPC, and PB are invalidated. This occurs after most prefetch faults. The resynchronization and refill sequence is:

```
MOV     R7,R7     ; re-establish the VPC
RSYNC             ; generate new PPC
RDF               ; get next I-stream word (demand)
NOP     PF,ID1    ; decode and prefetch
```

The second sequence is only used when the PB is invalid, but the VPC and PPC are correct. This occurs if the microcode has explicitly resynchronized in the course of executing a macroinstruction. The refill sequence in this case is:

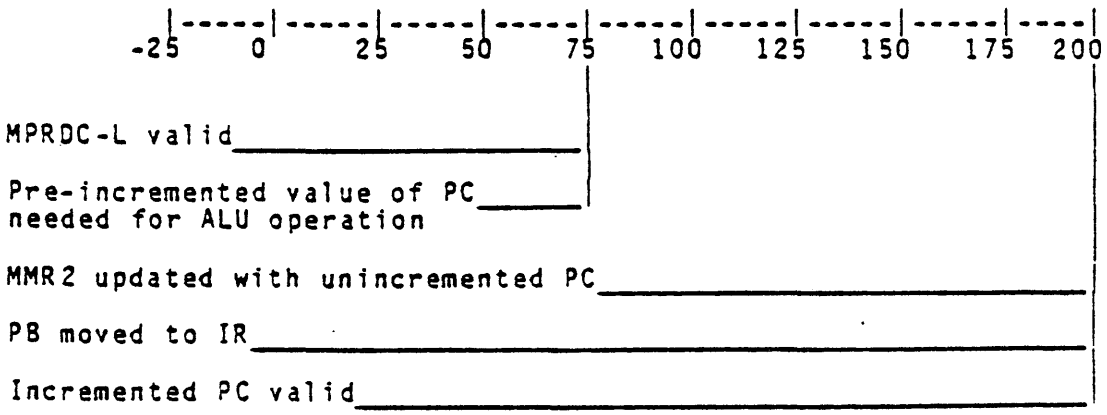
```
RDF               ; get next I-stream word (demand)
NOP     PF,ID1    ; decode and prefetch
```

4.3 Parallel Decode Operation

The Data chip recognizes when parallel decode (predecode) is occuring by examining MPRDC-L at T75. This initiates the following actions:

```
MMR2 <-- PC (if MMR0<15:13> = 000)
PC   <-- PC PLUS 2
IR   <-- PB
```

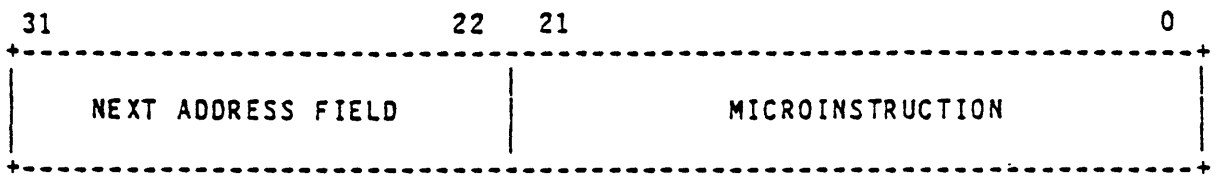
The timing of the predecode actions is critical. Specifically, the pre-incremented value of the PC must be used to update MMR2 and as input to the EU data path for an R7 register access. Furthermore, the PC must be incremented by two before the beginning of the next microinstruction. The following timing diagram illustrates the predecode sequence:



Note that the microinstruction which invokes predecode cannot specify longword or upper word operands.

5.0 MICROINSTRUCTIONS

Data chip operations are controlled by externally generated microinstructions. A Control chip outputs a microinstruction every microcycle on MIB-H<21:0>. The microinstruction is strobed from the MIB at T-25. All micro-sequencing is performed by the Control chips. There are ten (10) bits in each microword (Next Address Field) which specifically control the sequencing aspect of the Control chips. These bits are an integral part of the microinstruction format but always remain invisible to the Data chip; they never leave the Control chips. The following represents the complete microinstruction format:



The meaning of the next address field is covered in the Control Chip Specification. The Data Chip Specification is only concerned with bits<21:0>.

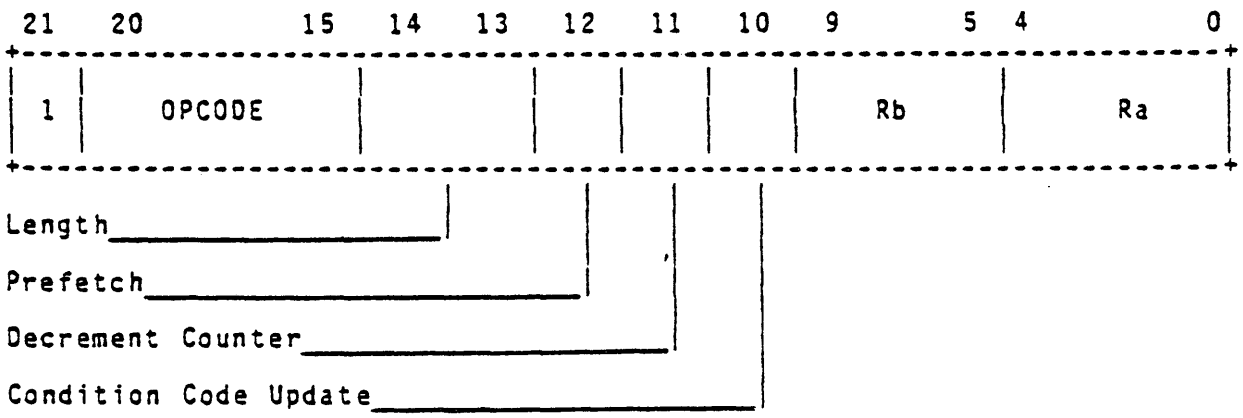
There are six basic classes of microinstructions: operate, literal, address, internal input/output, external input/output, and jump. This section summarizes the format and operation of all microinstructions. Note that opcodes are given in octal.

5.1 Operate Microinstructions

The microinstructions of this class operate on the registers in the EU register file. There are two basic types: double operand microinstructions which operate on the contents of the registers specified by the Ra and Rb fields, and which may write the result back into Ra; and single operand microinstructions which operate on the contents of register Rb and may write the result into Ra.

Most of the operate microinstructions can operate on a byte (8-bit), word (16-bit), or longword (32-bit) basis. With the exception of SWAB.B(*) and LLSW.B(*), byte microinstructions effectively operate on the lower byte (bits<7:0>) or mid byte (bits<23:16>) of the Ra/Rb operands, depending on the Rb select code. Word microinstructions effectively operate on either the low word (bits<15:0>) or the high word (bits<31:16>) of the Ra/Rb operands, depending on the Rb select code. Longword microinstructions operate on all 32 bits of the Ra/Rb operands. The local status flags are mechanized to reflect the state of byte, word, or longword operations. See sections 2.3.1.2 and 2.3.1.3 for details of data flow and status flag generation.

5.1.1 Format



5.1.1.1 Operand Length

In operate (as well as literal and input/output) microinstructions, bits<14:13> specify the operand length, which can be 8 bits, 16 bits, or 32 bits. These lengths are referred to as byte, word, or longword and are symbolically represented by concatenating the following symbols to a microinstruction mnemonic:

Symbol	Action
B	Byte operand
W	Word operand
L	Longword operand

In summary:

Bits<14:13> =
- 00: word operand (preferred representation)
- 01: longword operand
- 10: word operand
- 11: byte operand

NOTE: Note that a Control chip override of microinstruction bits<14:13> dynamically alters a byte-oriented microinstruction to the corresponding word-oriented version before transmission to the Data chip for execution.

5.1.1.2 PS Condition Code Update

In an operate microinstruction, bit<10> controls the updating of the PS condition codes (PS<3:0>). When bit<10> = 0, the local status flags (AN, AZ, AV, and AC) are copied to the PS condition codes (N, Z, V, C) at the end of the microcycle. This is symbolically indicated by concatenating the symbol "*" to a microinstruction mnemonic.

Bit<10> =
- 0: update PS condition codes
- 1: no effect

5.1.1.3 Counter Control

In an operate microinstruction, bit<11> is used to decrement the loop counter in the Control chip. The counter provides a mechanism for fast microcode sequencing. Refer to the Control chip for specific details on how microinstruction flow is affected by the counter.

Bit<11> =
- 0: decrement counter
- 1: no effect

5.1.1.4 Prefetch Control Field

In an operate microinstruction, bit<12> controls the prefetching of the next instruction stream word:

Bit<12> = 0: prefetch
1: no effect

5.1.2 Special Assembly Syntax

The condition code and length control fields can create six variations of the same instruction, as is shown by the ADD.(B)(W)(L)(*) example below:

Opcode <21:10>	Mnemonic	Instruction
14476	ADD.B	ADD byte, leave PS condition codes alone
14474	ADD.B*	ADD byte, update PS condition codes
14416	ADD.W	ADD word, leave PS condition codes alone
14414	ADD.W*	ADD word, update PS condition codes
14436	ADD.L	ADD longword, leave PS condition codes alone
14434	ADD.L*	ADD longword, update PS condition codes

5.1.3 Mnemonics

Table 5-1 is a list of the operate microinstructions and their functional descriptions including the resulting local status flags. Table 5-2 defines the local status flag symbols used in Table 5-1. Some of these microinstructions are identical to PDP-11 instructions in operation and condition code setting. The notes following Table 5-1 explain the non-PDP-11 microinstructions.

Table 5-1
OPERATE MICROINSTRUCTIONS

Opcode <21:15>	Mnemonic	Instruction	Operation	Flags AAAA NZVC	Notes
[arithmetic group]					
144	ADD.(B)(W)(L)(*)	Add	Ra=Ra+Rb	acnx	
160	ADDC.(B)(W)(L)(*)	Add with Carry	Ra=Ra+Rb+C	adnx	1
136	INC.(B)(W)(L)(*)	Increment	Ra=Rb+1	ach-	
164	ADC.(B)(W)(L)(*)	Add Carry	Ra=Rb+C	acjx	
141	SUB.(B)(W)(L)(*)	Subtract	Ra=Ra-Rb	acqv	
165	SUBC.(B)(W)(L)(*)	Subtract With Carry	Ra=Ra-Rb-C	adqv	1
145	DEC.(B)(W)(L)(*)	Decrement	Ra=Rb-1	ack-	
161	SBC.(B)(W)(L)(*)	Subtract Carry	Ra=Rb-C	acmw	
172	AOBC.(B)(W)(L)(*)	Add On Branch Condition	Ra=Ra+Rb if cond met	acnx	2
140	SOBC.(B)(W)(L)(*)	Subtract On Branch Condition	Ra=Ra-Rb if AZ=0	acqv	2
133	NEG.(B)(W)(L)(*)	Negate (2's complement)	Ra=-Rb	acht	
137	NEGC.(B)(W)(L)(*)	Negate with Carry	Ra=-Rb-C	adiu	1
103	CNEG.(B)(W)(L)(*)	Conditional Negate	Ra=-Rb if C=1	acHT	3
113	ACNEG.(B)(W)(L)(*)	Conditional Negate	Ra=-Rb if AC=1	acHT	3
107	NNEG.(B)(W)(L)(*)	Conditional Negate	Ra=-Rb if N=1	acHT	3
117	ANNEG.(B)(W)(L)(*)	Conditional Negate	Ra=-Rb if AN=1	acHT	3
143	CMP.(B)(W)(L)(*)	Compare	Rb-Ra	acpv	
[logical group]					
156	MOV.(B)(W)(L)(*)	Move	Ra=Rb	ac0-	
112	MOVPM.(B)(W)(L)(*)	Move Using Prev Mode	Ra=Rb	ac0-	4
102	MOVS.(B)(W)(L)(*)	Move Sign Extend	Ra=sext[Rb]	ac0-	5
104	CLR.(B)(W)(L)(*)	Clear	Ra=0	0100	
147	COM.(B)(W)(L)(*)	Complement	Ra=NOT Rb	ac01	
106	SXT.(B)(W)(L)(*)	Sign Extend	Ra=0 if N=0 Ra=-1 if N=1	ac0-	
151	AND.(B)(W)(L)(*)	And	Ra=Ra AND Rb	ac0-	
155	BIC.(B)(W)(L)(*)	Bit Clear	Ra=Ra AND NOT Rb	ac0-	
157	BIS.(B)(W)(L)(*)	Bit Set	Ra=Ra OR Rb	ac0-	
154	XOR.(B)(W)(L)(*)	Exclusive-Or	Ra=Ra XOR Rb	ac0-	
150	BIT.(B)(W)(L)(*)	Bit Test	Ra AND Rb	ac0-	
166	TST.(B)(W)(L)(*)	Test	Rb	ac00	
167	MTST(*)	Multiply Test	Ra	ac0y	6
146	SWAB.(B)(W)(L)(*)	Swap Bytes	Ra=Rb	be00	7
177	NOP(*)	No Operation	none	0100	

Table 5-1 (continued)
OPERATE MICROINSTRUCTIONS

Opcode <21:15>	Mnemonic	Instruction	Operation	Flags AAAA NZVC	Notes
[shift group]					
134	ASR.(B)(W)(L)(*)	Arith Shift Right	Ra=Rb/2	acfr	
124	ROR.(B)(W)(L)(*)	Rotate Right	Ra=Rb/2	acfr	
120	LSR.(B)(W)(L)(*)	Logical Shift Right	Ra=Rb/2	acfr	8
130	LSR.Q(*)	Logical Shift Right Quadword	Ra'SR=(Rb'SR)/2	acfz	9
123	ASL.(B)(W)(L)(*)	Arith Shift Left	Ra=Rb*2	acgs	
121	ROL.(B)(W)(L)(*)	Rotate Left	Ra=Rb*2+C	acfs	
131	ASL.Q(*)	Arith Shift Left Quadword	Ra'SR=(Rb'SR)*2	acfs	10
[multiply/divide group]					
110	UMULS.(W)(L)(*)	Unsigned Multiply Step	(see note)	acfz	11
114	SMULS.(W)(L)(*)	Signed Multiply Step	(see note)	acfz	11
105	DIVS.(W)(L)(*)	Divide Step	(see note)	acfx	11
111	XLDIVS(*)	Extended Low Divide Step	(see note)	acQ!	11
115	XHDIVS(*)	Extended High Divide Step	(see note)	acfx	11
[decimal group]					
174	DADP.(B)(W)(L)(*)	Decimal Post-Adjust	(see note)	acnx	12
170	PDAD.(B)(W)(L)(*)	Decimal Pre-Adjust	(see note)	acnx	12
[prefetch group]					
162	RSYNC	Resynchronize	(see note)	0100	13

NOTES

1. ADDC, SUBC, and NEGC perform the same arithmetic operations as ADD, SUB, and NEG respectively, except that the PS C flag is added to or subtracted from the result, as appropriate. Accordingly, they can be used for multi-word operations. The zero test is propagated by allowing these special instructions to only clear the local AZ flag if the result is non-zero. If the result is zero, the AZ flag is copied from the PS Z flag. All other flags are affected as for the ADD, SUB, and NEG microinstructions.

2. AOBC and SOBC are specified in section 2.4.1.

3. CNEG, ACNEG, NNEG, ANNEG negate Rb and store it in Ra if the specified condition is met; otherwise they store the unmodified Rb into Ra.

4. MOVPM is a specialized form of the MOV microinstruction. If R6 (Stack Pointer) is designated as a source or destination, MOVPM uses the previous mode field (PS<13:12>) to select the stack pointer. All other microinstructions use the current mode field (PS<15:14>).

5. MOVS is a specialized form of the MOV microinstruction. It sign extends the low order byte (bit<7>) of the B-bus operand through the upper byte(s).

6. MTST sets the AC flag if bits<31:15> of the result are not all zeroes or all ones. The other flags are set as for a TST microinstruction.

7. SWAB.B causes the contents of the even byte of the selected word of Rb (bits<23:16> or <7:0>) to be written into the odd byte of the selected word of Ra (bits<31:24> or <15:8>) without altering the remaining contents of Ra. SWAB.L swaps the bytes within both words of the 32-bit Rb operand.

8. LSR is similar to ASR except that the high order bit of the result is filled with a zero.

9. LSR.Q is similar to LSR.L, except that the Rb operand and the shift register are shifted together. The bit shifted out of the Rb operand is shifted into the high order bit of the SR, and the low order bit of the SR is shifted into the local AC flag.

10. ASL.Q is similar to ASL.L, except that the Rb operand and the shift register are shifted together. The low order bit of the Rb operand is filled with the high order bit shifted out of the SR, and the low order bit of the SR is filled with zero.

11. UMULS, SMULS, DIVS, XLDIVS, and XHDIVS are specified in Appendix A.

12. DADP and PDAD are specified in section 2.2.2.

13. RSYNC is specified in section 4.2.

Table 5-2
STATUS FLAGS KEY

!	copy the old ALU status flag (i.e., ALU status unchanged)
-	copy the old PS flag
1	set unconditionally
0	cleared unconditionally
a	set if MSB of result = 1; cleared otherwise
b	if longword opcode, set if bit<31> = 1; cleared otherwise if word opcode, set if MSB of low byte of result = 1; cleared otherwise if byte opcode, set if MSB of high byte of result = 1; cleared otherwise
c	set if result = 0; cleared otherwise
d	cleared if result not 0; copied from PS Z flag otherwise
e	if longword opcode, set if 32-bit result = 0; cleared otherwise if word opcode, set if low byte of result = 0; cleared otherwise if byte opcode, set if high byte of result = 0; cleared otherwise
f	the exclusive-or of the result's AN-bit and AC-bit
g	the OR of the exclusive-or of the result's AN-bit and AC-bit with sticky V flop
h	if longword opcode, set if result is 20000000000; cleared otherwise if word opcode, set if result is 100000; cleared otherwise if byte opcode, set if result is 200; cleared otherwise

H	if condition true then
	(if longword opcode, set if result is 20000000000; cleared otherwise
	if word opcode, set if result is 100000; cleared otherwise
	if byte opcode, set if result is 200; cleared otherwise)
	if condition false then
	(cleared)

i	if longword opcode, set if result is 20000000000 and C was 0; cleared otherwise if word opcode, set if result is 100000 and C was 0; cleared otherwise if byte opcode, set if result is 200 and C was 0; cleared otherwise
j	if longword opcode, set if result is 20000000000 and C was 1; cleared otherwise if word opcode, set if result is 100000 and C was 1; cleared otherwise if byte opcode, set if result is 200 and C was 1; cleared otherwise

Table 5-2 (continued)
STATUS FLAGS KEY

k if longword opcode, set if Rb was 20000000000; cleared otherwise
if word opcode, set if Rb was 100000; cleared otherwise
if byte opcode, set if Rb was 200; cleared otherwise

m if longword opcode, set if Rb was 20000000000 and C was 1; cleared otherwise
if word opcode, set if Rb was 100000 and C was 1; cleared otherwise
if byte opcode, set if Rb was 200 and C was 1; cleared otherwise

n set if operands' signs are same and result's sign is different; cleared otherwise

p set if operands' signs are different and the result's and Ra's signs are same; cleared otherwise

q set if operands' signs are different and the result's and Rb's signs are same; cleared otherwise

Q if AC = 1 then
(set if operands' signs are same and result's sign is different; cleared otherwise)
if AC = 0 then
(set if operands' signs are different and the result's and Rb's signs are same; cleared otherwise)

r old LSB of Rb

s old MSB of Rb

t cleared if result = 0; set otherwise

T if condition true then
(cleared if result = 0; set otherwise)
if condition false then
(cleared)

u cleared if result = 0 and C was 0; set otherwise

v cleared if carry out from MSB of result is 1; set otherwise

w set if Rb was 0 and C was 1; cleared otherwise

x set if carry out from MSB of result is 1; cleared otherwise

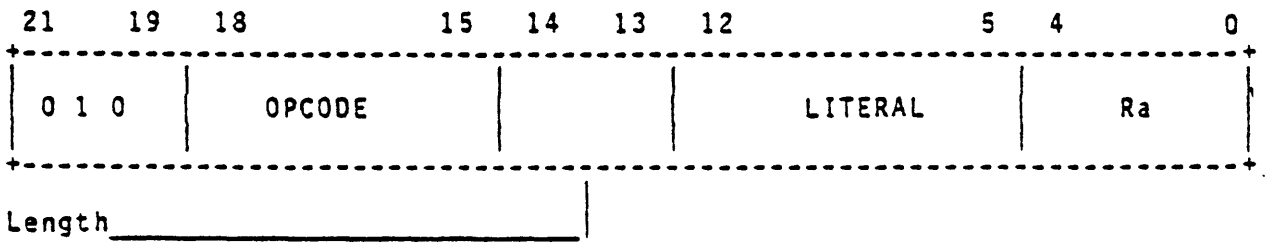
y set if Ra < -2**15 or Ra > 2**15-1; cleared otherwise

z old LSB of SR

5.2 Literal Microinstructions

The literal microinstructions perform arithmetic and logical functions using the contents of register Ra as one operand and a zero-extended 8-bit literal from the microinstruction as the other. The result is stored back in register Ra. Since literal microinstructions may be byte, word, or longword operations, a 16-bit or 32-bit operand must be created from the 8-bit literal information. All literal microinstructions place the literal field on bits<7:0> of the B-bus and insert zeroes in bits<31:8>. Data flow then depends on the length field and the Ra select code (see sections 2.3.1.2 and 2.3.1.3).

5.2.1 Format



The length field is the same as for operate microinstructions.

5.2.2 Mnemonics

Table 5-3 is a list of the mnemonics, operations, and local status flag output of the literal microinstructions.

Table 5-3
LITERAL MICROINSTRUCTIONS

Opcode	Mnemonic	Instruction	Operation	Flags A N	A Z	A V	A C
056	LLD.(B)(W)(L)	Load Literal	Ra=Literal	a	c	0	-
046	LLSW.(B)(W)(L)	Load Literal Swapped	Ra=Literal Swapped	b	e	0	0
047	LLCM.(B)(W)(L)	Load Literal Complement	Ra=NOT Literal	a	c	0	1
052	LCNTR.(B)(W)(L)	Load Counter	Ra=Literal	a	c	0	-
053	LMSTK.(B)(W)(L)	Load Stack	Ra=Literal	a	c	0	-
043	LCMP.(B)(W)(L)	Compare Literal	Literal-Ra	a	c	p	v
044	LADD.(B)(W)(L)	Add Literal	Ra=Ra+Literal	a	c	n	x
041	LSUB.(B)(W)(L)	Subtract Literal	Ra=Ra-Literal	a	c	q	v
057	LBIS.(B)(W)(L)	Bit Set Literal	Ra=Ra OR Literal	a	c	0	-
051	LAND.(B)(W)(L)	And Literal	Ra=Ra AND Literal	a	c	0	-
055	LBIC.(B)(W)(L)	Bit Clear Literal	Ra=Ra AND NOT Literal	a	c	0	-
050	LBIT.(B)(W)(L)	Bit Test Literal	Ra AND Literal	a	c	0	-
054	LXOR.(B)(W)(L)	Exclusive-OR Literal	Ra=Ra XOR Literal	a	c	0	-

Status Flags:
- \- copy the old PS flag
- 1 set unconditionally
- 0 cleared unconditionally
- a set if MSB of result = 1; cleared otherwise

b if longword opcode, set if bit<31> = 1; cleared otherwise
if word opcode, set if MSB of low byte of result = 1; cleared otherwise
if byte opcode, set if MSB of high byte of result = 1; cleared otherwise

c set if result = 0; cleared otherwise

e if longword opcode, set if 32-bit result = 0, cleared otherwise
if word opcode, set if low byte of result = 0; cleared otherwise
if byte opcode, set if high byte of result = 0; cleared otherwise

n set if operands' signs are the same and the result's sign is different; cleared otherwise

p set if operands' signs are different and the result's and Ra's signs are same; cleared otherwise

q set if operands' signs are different and the result's and Rb's signs are same; cleared otherwise

v cleared if carry out from MSB of result is 1; set otherwise

x set if carry out from MSB of result is 1; cleared otherwise

5.2.2.1 Load Counter (LCNTR)

As far as the Data chip is concerned, this literal microinstruction is identical to LLD. The selected Control chip decodes this microinstruction and then loads bits<12:5> of the microinstruction (together with eight high-order zero bits) into the Control chip counter. Refer to the Control Chip Specification for more details on this feature.

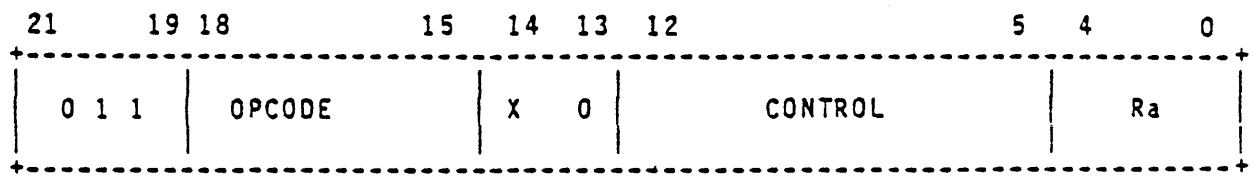
5.2.2.2 Load Microstack (LMSTK)

As far as the Data chip is concerned, this literal microinstruction is identical to LLD. The selected Control chip decodes this microinstruction and then pushes bits<14:5> of the microinstruction onto the microsubroutine stack. Refer to the Control Chip Specification for more details on this feature.

5.3 Address Microinstructions

The address microinstructions (except for ARG and AWG) initiate a relocation cycle prior to an external data stream I/O cycle. The virtual address specified by the Ra operand is relocated, and the resulting physical address is latched in the MDAL output latch. In addition, these microinstructions can increment or decrement the Ra operand by 1 or 2.

5.3.1 Format



The length field is always x0 (word length). If overridden with a byte to word override, the length field becomes 00 (still word length).

5.3.2 Mnemonics

Table 5-4 details the address microinstruction mnemonics and operations.

Table 5-4
ADDRESS MICROINSTRUCTIONS

Opcode	Mnemonic	Instruction	Operation
073	AR	Address read	Virt addr = Ra
070	ARI.(B)(W)	Address read and increment	Virt addr = Ra Ra = Ra + 1 + d
071	ARD.(B)(W)	Address read and decrement	Virt addr = Ra Ra = Ra - 1 - d
077	AW	Address write	Virt addr = Ra
074	AWI.(B)(W)	Address write and increment	Virt addr = Ra Ra = Ra + 1 + d
075	AWD.(B)(W)	Address write and decrement	Virt addr = Ra Ra = Ra - 1 - d
063	ARG	Address read general purpose	Final addr = 37400 @ bits<12:5>
067	AWG	Address write general purpose	Final addr = 37400 @ bits<12:5>

Where d = -bit<14>

Table 5-5 lists the control field encodings for all address microinstructions except ARG and AWG, which use the control field as the final address. Note that the control field only applies to the current microinstruction.

Table 5-5
ADDRESS CONTROL FIELD ENCODINGS

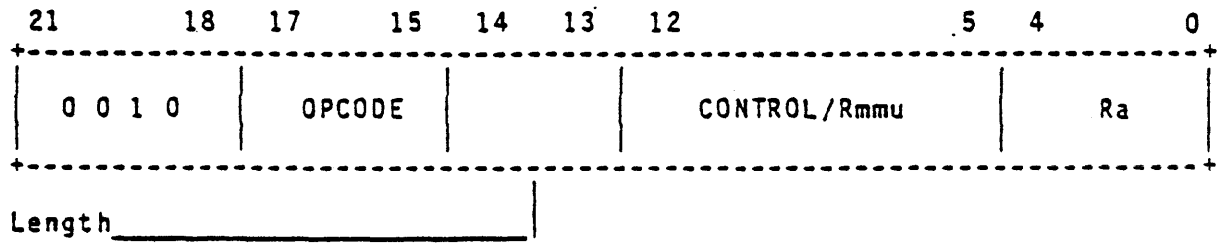
CF<6:5>	00: instruction space unconditionally 01: instruction space unless PS<15:12> = 1111, then data space 10: data space unconditionally 11: data space unless Ra resolves to PC, then instruction space
CF<7> =	0: force 16-bit mapping (i.e., calculate address as though MMR0<0> = 0) 1: no effect
CF<8> =	0: arm address trap 1: no effect
CF<9> =	0: activate stack overflow logic 1: no effect
CF<11:10> =	00: use kernel mode 01: use previous mode 10: use console mode 11: use current mode
CF<12> =	0: update MMR1 (only for ARI, ARD, AWI, AWD) 1: no effect

After an address microinstruction, the local status flags are unchanged.

5.4 Internal Input/Output Microinstructions

The internal I/O microinstructions are used to transfer data and control information within the chip set. No addressing and/or relocation is necessary.

5.4.1 Format



The length field is the same as for the operate microinstructions. Note that a length of longword is not meaningful except for transfers to and from the Shift Register (SR). In INPR and OUTR, bits<12:5> act as the Rmmu register select. In OUTS and OUTC, they act as a control field (CF).

5.4.2 Mnemonics

Table 5-6 lists the internal I/O microinstructions.

Table 5-6
INTERNAL I/O MICROINSTRUCTIONS

Opcode	Mnemonic	Instruction
021	INPR.(W)(L)	Read data from MMU register
025	OUTR.(W)(L)	Write data to MMU register
027	OUTS	Write Control chip status
026	OUTC	Write Control chip control

After an INPR or OUTR, the local status flags are set as follows:

AC = 0

AV = 0

AZ = 1 if operand was zero
= 0 otherwise

AN = 1 if most significant bit of operand was 1
= 0 otherwise

After an OUTS or OUTC, the local status flags are AC=0, AV=0, AZ=1, and AN=0.

5.4.2.1 Read Data from MMU Register (INPR)

This microinstruction is used to move data from the MMU register selected by the Rmmu field to the EU register selected by the Ra field. The data is also written to the external MDAL-H bus. There is no control field. INPR defines an MMU register as any internal Data chip register which is not in the EU register file. This allows the MMU registers to be directly accessible with microcode. The register select codes are shown in Tables 3-2 and 3-3.

5.4.2.2 Write Data to MMU Register (OUTR)

This microinstruction moves data from the EU register selected by the Ra field to the MMU register selected by the Rmmu field. There is no control field. OUTR defines an MMU register as any of the Data chip registers which are not physically part of the EU register file. This allows the MMU registers to be directly accessible with microcode. Tables 3-2 and 3-3 list the MMU register file select codes.

5.4.2.3 Write Control Chip Status (OUTS)

This microinstruction sends status to, or routes status within, the Control chip(s). The control field is used to signal the Control chip as is listed in Table 5-9.

Table 5-9
OUTS CONTROL FIELD ENCODINGS

CF<5>	=	0:	load PIR from interrupt service information
		1:	no effect
CF<6>	=	0:	load PIR from abort service information
		1:	no effect
CF<7>	=	0:	load PIR from FPS<7:5>, CPFF, and MDAL-H<11:0>
		1:	no effect
CF<8>	=	0:	push MDAL-H<9:0> onto top of Control chip stack
		1:	no effect
CF<9>	=	0:	latch Control chip PIR from MDAL-H<15:0>
		1:	no effect
CF<10>	=	0:	latch Control chip counter from MDAL-H<15:0>
		1:	no effect
CF<11>	=	0:	latch Control chip copy of PS<7:4> from MDAL-H<7:4>
		1:	no effect
CF<12>	=	0:	load FPS<7:5> from MDAL-H<7:5>
		1:	no effect

5.4.2.4 Write Control Chip Control (OUTC)

This microinstruction alters the internal state of the Control chip(s). The control field is used to affect the Control chip as is listed in Table 5-10.

Table 5-10
OUTC CONTROL FIELD ENCODINGS

CF<5>	=	data for CF<6>, CF<7> options
CF<6>	=	0: load single-step flip-flop from CF<5> 1: no effect
CF<7>	=	0: load coprocessor flip-flop from CF<5> 1: no effect
CF<8>		unused
CF<9>	=	0: copy internal PS<4> to intermediate T-bit flip-flop 1: no effect
CF<10>	=	0: clear stack overflow flip-flop 1: no effect
CF<11>	=	0: clear intermediate T-bit flip-flop 1: no effect
CF<12>	=	0: clear parity error flip-flop 1: no effect

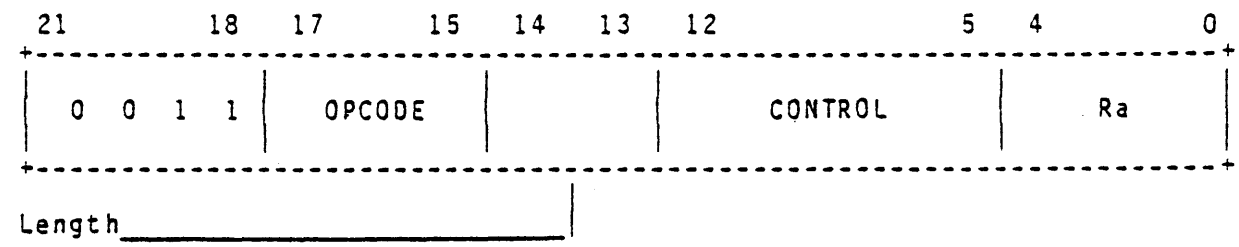
Note: Unused bits may be either 1 or 0 and must not have any effect on microinstruction operation either way.

5.5 External Input/Output Microinstructions

These microinstructions provide the microcode interface to the external bus and hence allow microcode access to main memory and peripheral devices. External I/O microinstructions supply a physical address (PA) or general purpose code (GP) on MDAL-H<21:0>. Instruction stream references obtain the physical address from the physical PC (PPC), data stream and general purpose references from the MDAL output latch. The Ra field within the microinstruction specifies the register destination for input data. Note that there are no explicit write data or write general purpose microinstructions. Instead, in the cycle following an address write microinstruction (AW, AWI, AWD, AWG) or a read-modify-write microinstruction (RMW), the data present at the I/O multiplexor (i.e., the output of the EU data path for word or byte operate and literal microinstructions) is automatically written out to MDAL-H or (during an explicit access) to the specified internal register, provided that the address write or read-modify-write did not result in an abort. Longword microinstructions after AW, AWI, AWD, AWG, or RMW are illegal and a microcode restriction.

For the timing of external I/O cycles, see the timing diagrams.

5.5.1 Format



The length specification is the same as for operate microinstructions. Note that a length of longword is not meaningful.

5.5.2 Mnemonics

Table 5-11 lists the mnemonics and operations of the external I/O microinstructions.

Table 5-11
EXTERNAL I/O MICROINSTRUCTIONS

Opcode	Mnemonic	Instruction
032	RD.(B)(W)	Read data
033	RMW.(B)(W)	Read-modify-write data
036	RDG.(B)(W)	Read general purpose
037	RDINTR	Read interrupt vector
030	RDI	Read instruction stream
031	RDF	Read demand prefetch

After an external I/O microinstruction, the local status flags are set as follows:

AC = 0

AV = 0

AZ = 1 if operand was zero
= 0 otherwise

AN = 1 if most significant bit of operand was 1
= 0 otherwise

5.5.2.1 Read Data (RD)

This microinstruction reads external data into the EU register specified by Ra. The Data chip first drives the physical address in the output latch. It then samples first for cache data and subsequently, in the case of a cache miss, cache bypass, or I/O page reference, for external bus data. The control field is used to activate various options, as shown in Table 5-12.

Table 5-12
RD CONTROL FIELD ENCODINGS

CF<8:5>	unused
CF<9> =	0: activate RTI/RTT-to-PS protection 1: no effect
CF<10> =	0: activate vector-to-PS protection 1: no effect
CF<12:11>	unused

Note: Unused bits may be either 1 or 0 and must have no effect on microinstruction operation either way.

5.5.2.2 Read-Modify-Write Data (RMW)

This microinstruction initiates a read-modify-write bus operation. The read operation is identical to a RD microinstruction as described above. During the microcycle following the RMW microinstruction, the physical address latched in the output latch is driven out onto MDAL-H, followed by the output of the EU data path.

The control field encodings for RMW are shown in Table 5-13.

Table 5-13
RMW CONTROL FIELD ENCODINGS

CF<8:5>	unused
CF<9> =	0: activate RTI/RTT-to-PS protection 1: no effect
CF<10> =	0: activate vector-to-PS protection 1: no effect
CF<11>	unused
CF<12> =	0: read locked 1: read not locked

Note: Unused bits may be either 1 or 0 and must have no effect on microinstruction operation either way.

5.5.2.3 Read General Purpose (RDG)

This microinstruction reads data from the System Interface into the EU register specified by Ra. The Data chip first drives the GP code in the output latch (OL). It then samples for data from MDAL-H. The control field is not used. The RDG.B microinstruction byte swaps the data based on OL<0> (which is usually set from GP address bit<0> by an ARG microinstruction). OL<0> = 1 means byte swap the data; OL<0> = 0 means no byte swap.

5.5.2.4 Read Interrupt Vector (RDINTR)

This microinstruction reads an interrupt vector from the System Interface into the EU register specified by Ra. The Data chip first drives the interrupt priority level in the output latch. It then samples MDAL-H for a vector. The control field is not used.

5.5.2.5 Read Instruction Stream (RDI)

This microinstruction reads the sequential instruction stream word specified by the physical PC (PPC). If the prefetch valid flag is set (instruction read request), the input latch selected as the prefetch buffer is moved to Ra, the incoming data is stored in the deselected input latch, and the prefetch buffer select is switched. If the prefetch valid flag is clear, the incoming data is stored directly in Ra. The control field bits are not used. For a complete description of RDI and its function in the prefetch mechanism, see section 4.

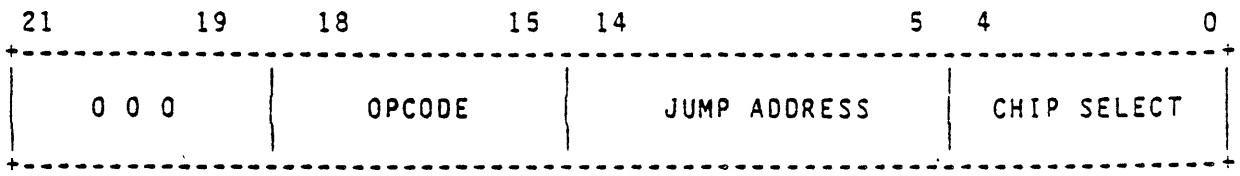
5.5.2.6 Read Demand Prefetch (RDF)

This microinstruction forces a demand prefetch of the sequential instruction stream word specified by the physical PC (PPC). The input data is stored in the prefetch buffer. The control field bits are not used; the register specified by Ra is not written into. For a complete description of RDF and its function in the prefetch mechanism, see section 4.

5.6 Jumps

The jump microinstructions provide for conditional jumps within a Control chip, or unconditional jumps between Control chips. The Data chip's role in conditional jump processing is to test for the specified jump condition and to assert the Jump Allow signal (MSOV/JA-L) if the specified condition is met. The Data chip treats JMP as a no operation.

5.6.1 Format



5.6.2 Mnemonics

Table 5-14 lists the mnemonics and test conditions for the jump microinstructions.

Table 5-14
JUMP MICROINSTRUCTIONS

Opcode	Mnemonic	Test Condition
000	JMP	Jump always (interchip)
017	JAN	Jump if AN flag = 1
011	JAZ	Jump if AZ flag = 1
013	JAC	Jump if AC flag = 1
015	JAV	Jump if AV flag = 1
007	JN	Jump if PS N flag = 1
001	JZ	Jump if PS Z flag = 1
003	JC	Jump if PS C flag = 1
005	JV	Jump if PS V flag = 1
016	JKM	Jump if PS<15:14> = 00 (kernel mode)

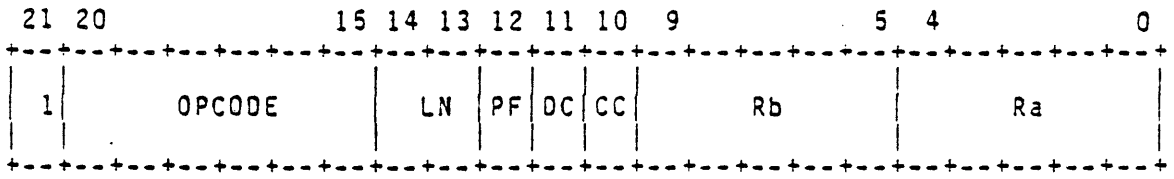
The local status flags are unchanged.

5.6.3 Pipelined Jumping

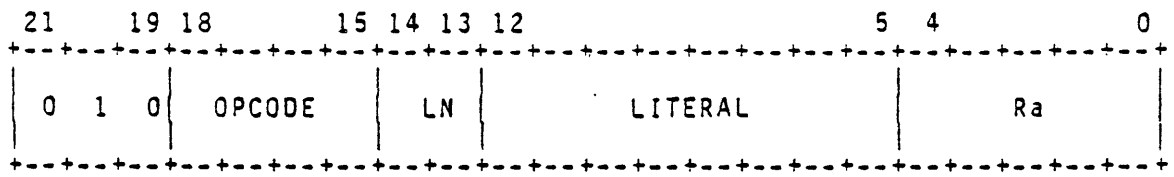
Due to the overlapped nature of the data paths, conditional jump microinstructions are pipelined. A conditional jump microinstruction whose jump condition is satisfied will not modify the microflow until the following microinstruction has been executed. This may necessitate padding the microcode with a NOP immediately after the conditional jump microinstruction. Unconditional jump (JMP) is not pipelined and takes effect immediately.

5.7 Microinstruction Format Summary

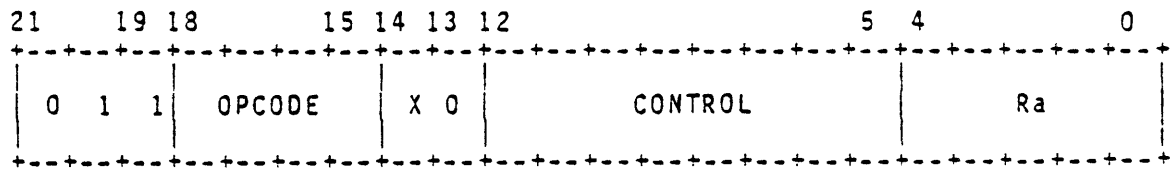
MICROINSTRUCTION FORMAT



Operate Microinstruction

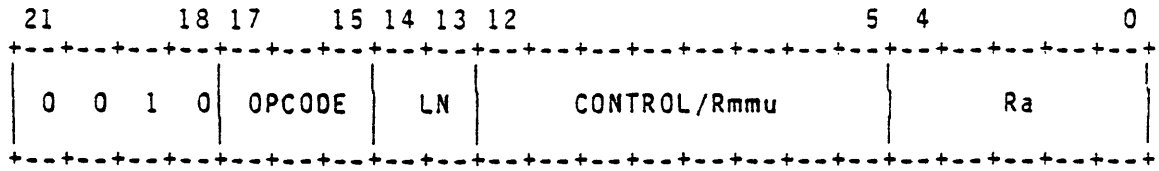


Literal Microinstruction

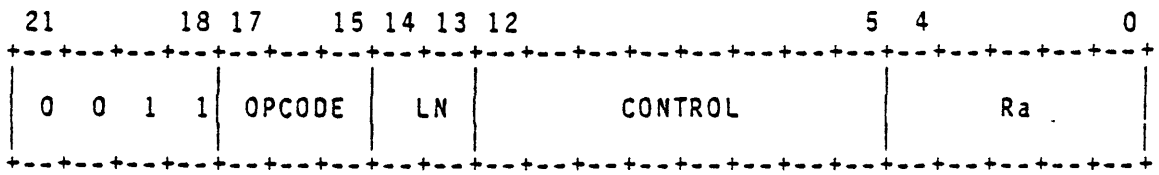


Address Microinstruction

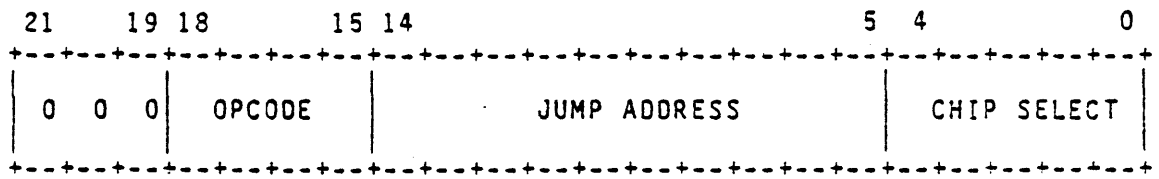
MICROINSTRUCTION FORMAT (CON'T)



Internal Input/Output Microinstruction



External Input/Output Microinstruction



Jump Microinstruction

6.0 INTERFACE

6.1 Inputs

6.1.1 Micro Data and Address Lines (MDAL-H<15:0>)

The Data chip reads data from the MDAL-H<15:0> lines. The Data chip strobes input data at T150 and also, during extended microcycles, when MDV-L is asserted.

6.1.2 Microinstruction Bus (MIB-H<21:0>)

The Data chip receives microinstructions from a Control chip on MIB-H<21:0>. The Data chip strobes MIB-H<21:0> at T-25.

6.1.3 Crystal Inputs (XTALI, XTALO)

A crystal (nominally 20Mhz) connects between XTALI and XTALO. This piezoelectric element is used in the generation of the system clock.

6.1.4 Data Valid (MDV-L)

During the extended portion of a non-write stretched microcycle (MBUFCTL-L asserted after T200), this signal controls the input latch which is waiting for data. If MDV-L is high, the latch is open; if MDV-L is low, the latch is closed. The System Interface asserts MDV-L during extended read cycles to indicate that valid data is present on the MDAL-H.

6.1.5 Cache Miss (MMISS-L)

This signal is asserted by the Cache Subsystem or the System Interface to indicate that a cache miss occurred during the current microcycle. It is sampled at T150.

6.1.6 Continue (MCONT-L)

The signal is driven by the System Interface. During extended microcycles, assertion of MCONT-L informs the Data chip that the System Interface is finished using the MDAL-H bus and that the chip set can start the next microcycle.

6.1.7 DMA Request (MDMR-L)

This signal is used by the System Interface to gain control of the MDAL bus. It is sampled by the Data chip at T0. The Data chip indicates the grant of the request by asserting the time-multiplexed signal MMAP-L. It then extends the current microcycle, asserts signal MSCTL-L, and enters wait state. The System Interface terminates wait state by asserting signal MCONT-L.

6.1.8 Abort (MABORT-L)

This signal is driven by the System Interface to indicate that an abort (non-existent memory, bus timeout, cache or main memory parity error) occurred during the current microcycle. The System Interface must assure that MABORT-L is synchronized with respect to MCONT-L and is not asserted before T200.

6.1.9 Predecode (MPRDC-L)

This signal indicates that the current microcycle is the last microcycle in the current macroinstruction. If MPRDC-L is asserted at T75 in a microcycle, the Data chip copies the contents of the prefetch buffer into the Instruction Register.

6.1.10 Initialize (MINIT-L)

This input is used by the power up logic in order to synchronize the chip set and the System Interface. Assertion of this input causes all bits in the Microinstruction Register (MIR) to be cleared, effectively setting up a JMP 0 for the first microcycle. When the Data chip recognizes MINIT-L, it asserts and then synchronously deasserts MSCTL-L to synchronize the Control chip(s) to the Data chip. Deassertion of MSCTL-L indicates that the next chip state is T-25. For details on initialization timing, see the timing diagrams.

6.1.11 Test Mode (TEST1-L)

Assertion of TEST1-L disables the output of MCLK and MCLK2. This permits external logic to drive the Data chip clock circuitry through MCLK. TEST1-L is internally pulled to the high (inactive) state.

6.1.12 Chip Disable (TEST2-L)

Assertion of TEST2-L causes the Data chip to disable all outputs. This input is internally pulled to the high (inactive) state.

6.2 Outputs

6.2.1 Micro Data and Address Lines (MDAL-H<21:0>)

These outputs transmit addresses and data from the Data chip to the Control chip(s), Cache Subsystem, and System Interface. During an I/O cycle, the Data chip drives the physical address (or GP code) onto MDAL-H<21:0> by T25. During a write sequence, the Data chip drives the output data onto MDAL-H<15:0> at T175 until the next T-25 and drives zeroes into MDAL<21:16>. During all cycles, the Data chip drives XC, XR, and the local status flags (AN, AZ, AV, AC) onto MDAL<21:16> at T75, and during non-I/O cycles, the low 16 bits of the A-port register onto MDAL-H<15:0> at T75.

6.2.2 Microinstruction Bus (MIB-H<21:0>)

The Data chip sustains the data on MIB-H<21:0> with a small sustainer device.

6.2.3 Clock (MCLK)

This signal is the source of the chip set clock. It nominally consists of a MOS level, single phase, 20Mhz squarewave.

6.2.4 Phase Clock (MCLK2)

This signal is an extra clock with the same output characteristics as MCLK for use by the System Interface. It nominally consists of a MOS level, single phase, 20Mhz squarewave.

6.2.5 Address Latch Enable (MALE-L)

This signal is asserted every microcycle at T25. It provides timing and control information for external logic. Assertion of MALE-L can be used to latch the physical address or GP code on MDAL-H<21:0> during any I/O cycle. MALE-L is deasserted at T150 every microcycle and can be used to latch the cache data.

6.2.6 MDAL Buffer Control (MBUFCTL-L)

During all read cycles, including instruction stream and data stream references, GP reads, and interrupt vector reads, this signal is asserted at T75 and deasserted at T150. It can be used to control the output enable on the cache data array. It is also asserted at T200 and deasserted at T-50 during all stretched non-write microcycles.

6.2.7 Stretch Control (MSCTL-L)

During extended microcycles, the Data chip asserts MSCTL-L to indicate the start of wait state, and deasserts MSCTL-L to mark the end. It is asserted at T250 and deasserted at T-100. Either edge may be used to strobe write data on writes.

During initialization, the Data chip asserts and then synchronously deasserts MSCTL-L to synchronize the Control chip(s) to the Data chip.

6.2.8 Stack Overflow and Jump Allow (MSOV/JA-L)

This time-multiplexed signal informs the Control chip that a stack violation has occurred or that the conditional jump test just executed has been satisfied. A stack violation is signaled during I/O microcycles; jump allow is signaled during conditional jump microinstructions. During other microcycles, this signal is set to a specified value at T75 and left unchanged for the rest of the microcycle.

6.2.9 Abort (MABORT-L)

This signal reports that an abort condition occurred during a memory operation. Both memory management errors (access violation, length error, etc.) and bus errors (parity, NXM, etc.) for demand and request cycles are reported by this line. The Data chip drives this signal high or low at T-25 and then sustains this state: if high, with a small sustainer device; if low, with a driver capable of sinking a TTL load. This signal may be driven low later in the microcycle by the System Interface. This low level is held by the Data chip until T-25 at which time the next microcycle's MMU abort state is asserted. During initialization, MABORT-L will be driven low by the Data chip.

6.2.10 Kill Prefetch Buffer (MKPB-L)

This signal is asserted by the Data chip when a condition which invalidates the prefetch buffer has been detected.

6.2.11 Strobe (MSTRB-L)

This signal is asserted every microcycle at T75 and deasserted at TEND (the next microcycle's T0). It is a general purpose strobe.

6.2.12 Bank Select (MBS-H<1:0>)

These signals are time-multiplexed. From T-25 to T75 during a read or write microcycle, they provide encoded information about the type of physical address present on MDAL-H<21:0>:

MBS<1>	MBS<0>	Address Class
0	0	Memory reference
0	1	System board register reference
1	0	External I/O reference
1	1	Internal Register reference

Physical addresses which are less than 17760000 (octal) are memory references. Addresses which fall in the I/O page (17760000 - 17777777) and do not access a CPU register (internal or board level) are external I/O references. Addresses which are in the I/O page and access Data chip internal registers (except for CCR) are internal register references. Lastly, addresses which access CPU registers which are partially or completely implemented in the System Interface are system board register references.

From T75 to T-25 during a read or write microcycle, MBS-H<1:0> convey miscellaneous status:

MBS-H<1> = cache bypass (if data stream, CCR<9>=1 + PDR<15>=1 + RMW<12>=0; if instruction stream, CCR<9>=1 + PCS<1>=1)

MBS-H<0> = force miss (CCR<3> + CCR<2>)

Note that these signals are stable until T-25 with the old data.

6.2.13 I/O Map Enable (MMAP-L)

This signal is time-multiplexed. If asserted at T50, it indicates that the I/O map is enabled (MMR3<5> = 1). If asserted at T150, it indicates that a DMA grant will occur during the current microcycle.

6.3 Signal Summary

Signal Name	Signal Type	Pin Numbers	Applicable DC Tests											
			VIH	VIL	VIHT	VILT	II	IIL	IOH	IOL	IOHT	IOSH	IOSL	IOZ	ICCSB
MDAL-H<21:16>	TTL IO								y	y				y	
MDAL-H<15:0>	TTL IO	tbs			y	y				y	y			y	
MIB-H<21:0>	MOS IO		y	y								y	y		
XTALI	tbs I		tbs												
XTALO	tbs O		tbs												
MDV-L	TTL I				y	y	y								
MMISS-L	TTL I				y	y	y								
MCONT-L	TTL I				y	y	y								
MDMR-L	TTL I				y	y	y								
MABORT-L	TTL IO	(1)				y				y	y	y		y	
MPRDC-L	TTL I				y	y	y								
MINIT-L	TTL I				y	y	y								
TEST1-L	MOS I		y	y				y							
TEST2-L	MOS I		y	y				y							
MCLK	MOS O								y	y				y	
MCLK2	MOS O								y	y				y	
MALE-L	TTL O									y	y			y	
MBUFCTL-L	TTL O									y	y			y	
MSCTL-L	TTL O									y	y			y	
MSOV/JA-L	MOS O								y	y				y	
MKPB-L	MOS O								y	y				y	
MSTRB-L	TTL O									y	y			y	
MBS-H<1:0>	TTL O									y	y			y	
MMAP-L	TTL O									y	y			y	
power	na														y
ground	na		na												

(1) MABORT-L must be driven with an open collector driver. The data chip has a pull-up device which supplies IOSH.

7.0 DC CHARACTERISTICS

Absolute Maximum Rating

Storage Temperature Range	-65 C to +150 C
Active Temperature Range	-55 C to +125 C
Supply Voltage	+7.0V
Input or Output Voltage Applied	V_{ss}-0.3V to V_{cc}+0.3V

Electrical Characteristics

Specified Temperature Range	0 C to +70 C
Specified Voltage Range	+4.75V to +5.25V
Test Conditions	Temperature = +70 C V_{ss} = 0V V_{cc} = +4.75V (except as noted)

Symbol	Parameter	Min.	Max.	Units	Test Condition
V_{IH}	High level MOS input	70% V_{CC}		V	
V_{IL}	Low level MOS input		30% V_{CC}	V	
V_{IHT}	High level TTL input	2.2		V	
V_{ILT}	Low level TTL input		0.6	V	
I_I	Input leakage current non-TEST inputs (note 1)	-10	10	uA	$0V \leq V_I \leq V_{cc}$
I_{ILL}	Input current TEST inputs (note 1)	.1	5	mA	V_I=0V
I_{OH}	Output current at high level	-2.0		mA	$V_O = V_{cc}$-0.4
I_{OL}	Output current at low level	2.0		mA	V_O=0.4V
I_{OHT}	Output current at high TTL level	-2.0		mA	V_O=2.4V

I_{OSH}	High level sustainer current (note 1)	-0.2	-0.6	mA	$V_O = V_{CC} - 1.0V$
I_{OSL}	Low level sustainer current (note 1)	0.2	0.6	mA	$V_O = 1.0V$
I_{OZ}	Output leakage current (notes 1,2)	-10	10	uA	$0V \leq V_O \leq V_{CC}$
I_{CCSB}	Static power supply current (notes 1,3)		tbs	uA	
C_{IN}	Input capacitance (note 4)		5	pF	
C_{IO}	Input/output capacitance (note 4)		15	pF	
C_{OUT}	Output capacitance (note 4)		15	pF	

Note 1 - tested at V_{CC} = 5.25V.

Note 2 - only applies in the high impedence condition.

Note 3 - with TEST1-L, TEST2-L, and all outputs open circuit, all other inputs equal to V_{CC}.

Note 4 - sampled and guaranteed, but not tested. Does not apply to TEST1-L or TEST2-L.

8.0 AC CHARACTERISTICS

Unless marked with an R (required), all AC characteristics are goals and are subject to further refinement.

Test Conditions

Temperature = +70 C
V_{ss} = 0V
V_{cc} = +4.75V (except as noted)
For output test loads, see Test Circuits

Timing Requirements

Symbol	Parameter	Min	Max	Units	Test Conditions
t_{initw}	MINIT-L pulse width	tbs		ns	
t_{initls}	MINIT-L setup	tbs		ns	
t_{mibs}	MIB-H setup	25		ns	
t_{mibh}	MIB-H hold	tbs		ns	
t_{prds}	MPRDC-L setup	25		ns	
t_{prdh}	MPRDC-L hold	tbs		ns	
t_{ds}	MDAL-H<15:0> setup with respect to T150 (R)	20		ns	
t_{dh}	MDAL-H<15:0> hold with respect to T150 (R)	0		ns	
t_{dvds}	MDAL-H<15:0> setup with respect to MDV-L	25		ns	
t_{dvdh}	MDAL-H<15:0> hold with respect to MDV-L	25		ns	
t_{dvw}	MDV-L pulse width	25		ns	
t_{dvf}	MDV-L fall time		15	ns	
t_{hms}	MMISS-L setup (R)	20		ns	
t_{hmh}	MMISS-L hold (R)	0		ns	
t_{abs}	MABORT-L drive	20		ns	
t_{abd}	MABORT-L delay	0		ns	

t_{abid}	MABORT-L turn-off delay	0	ns
t_{abw}	MABORT-L width	40 ns + 2 x t_{cycle}	
t_{cnts}	MCONT-L setup (note 1)	20	ns
t_{cnth}	MCONT-L hold (note 1)	20	ns
t_{dmrs}	MDMR-L setup (note 1)	20	ns
t_{dmrh}	MDMR-L hold (note 1)	20	ns

Note 1 - setup and hold requirements are only to guarantee recognition at next sample point.

Timing Responses

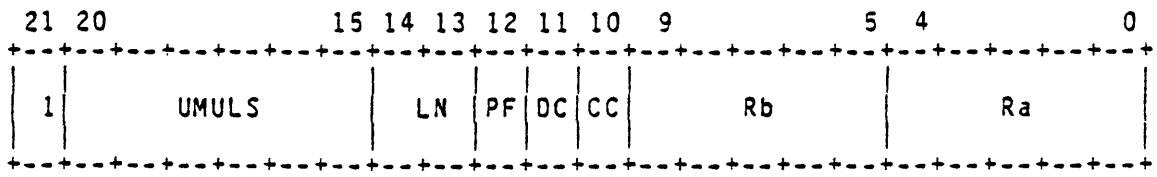
Symbol	Parameter	Min	Max	Units	Test Conditions
t_{cycle}	MCLK cycle time (R)	50		ns	test circuit 3 (note 2)
t_{clkh}	MCLK high width	18		ns	test circuit 3 (note 2)
t_{clkl}	MCLK low width	18		ns	test circuit 3 (note 2)
t_r	MCLK rise time		7	ns	test circuit 3 (note 2)
t_f	MCLK fall time		7	ns	test circuit 3 (note 2)
t_{pcyc}	MCLK2 cycle time	50		ns	test circuit 2 (note 2)
t_{pclkh}	MCLK2 high width	tbs		ns	test circuit 2 (note 2)
t_{pclkl}	MCLK2 low width	tbs		ns	test circuit 2 (note 2)
t_{pr}	MCLK2 rise time		tbs	ns	test circuit 2 (note 2)
t_{pf}	MCLK2 fall time		tbs	ns	test circuit 2 (note 2)
t_{pclkd}	MCLK2 valid delay		tbs	ns	test circuit 2 (note 2)
t_{sd}	strobe active delay (R)	0	25	ns	test circuit 2
t_{sid}	strobe inactive delay (R)	0	25	ns	test circuit 2
t_{dis}	MDAL-H output disable (R)		25	ns	test circuit 1

t_{dald}	MDAL-H valid delay (R)		25	ns	test circuit 2
t_{dalh}	MDAL-H valid hold	0		ns	test circuit 2
t_{dist}	transition to high impedance following assertion of TEST2-L		tbs	ns	functional test
t_{ent}	transition from high impedance following deassertion of TEST2-L		tbs	ns	functional test

Note 2 - input conditions are tbs.

Appendix A - Multiply/Divide Microinstructions

UMULS.(W)(L)(*) - Unsigned Multiply Step



Operation:

```
IF SR<0> = 1
    THEN S <-- Ra + Rb
    ELSE S <-- Ra
F<msb-1:0> <-- S<msb:1>
F<msb> <-- arithmetic carry out of ALU
AC <-- SR<0>
SR<30:0> <-- SR<31:1>
SR<31> <-- S<0>
```

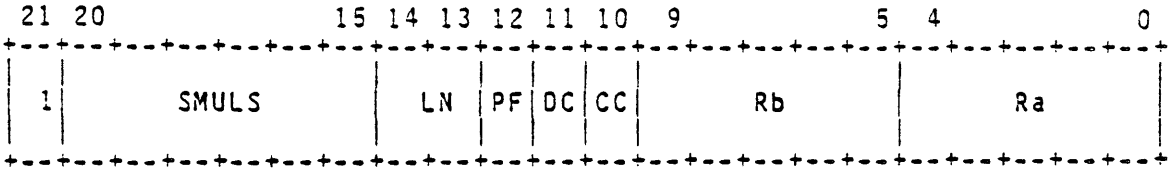
where msb is bit<31> for longword and bit<15> for word operations, S is the output of the ALU and the input to the shifter, F is the output of the shifter, and SR is the Shift Register.

Description: The lsb of the Shift Register (SR) is tested for a 0 or 1. If SR<0> = 0, then register Ra is passed through the ALU unmodified. If SR<0> = 1, then registers Rb and Ra are added together. In both cases the output of the ALU is rotated right together with the SR. The shift into the msb of the 32-bit shifter is the arithmetic carry out of the ALU. The shifted out bit of the SR becomes the new AC flag.

Status flags:

```
AN <-- msb of F
AZ <-- 0, if F <> 0
   <-- 1, if F = 0
AV <-- the exclusive-or of the result's AN-bit and AC-bit
AC <-- old lsb of the SR
```

SMULS.(W)(L)(*) - Signed Multiply Step



Operation:

```
IF SR<0> = 1
    THEN S <-- Ra + Rb
    ELSE S <-- Ra
F<msb-1:0> <-- S<msb:1>
F<msb> <-- S<msb> XOR (ALU arithmetic overflow)
AC <-- SR<0>
SR<30:0> <-- SR<31:1>
SR<31> <-- S<0>
```

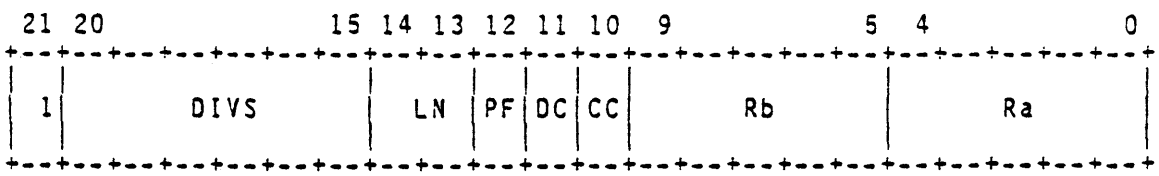
where msb is bit<31> for longword and bit<15> for word operations, S is the output of the ALU and the input to the shifter, F is the output of the shifter, and SR is the Shift Register.

Description: The lsb of the Shift Register (SR) is tested for a 0 or 1. If SR<0> = 0, then register Ra is passed through the ALU unmodified. If SR<0> = 1, then registers Rb and Ra are added together. In both cases the output of the ALU is rotated right together with the SR. The shift into the msb of the 32-bit shifter is the exclusive-or of the ALU's output's sign and the arithmetic overflow out of the ALU (arithmetic overflow is the exclusive-or the carry in and carry out of the msb). The shifted out bit of the SR becomes the new AC flag.

Status flags:

```
AN <-- msb of F
AZ <-- 0, if F <> 0
   <-- 1, if F = 0
AV <-- the exclusive-or of the result's AN-bit and AC-bit
AC <-- old lsb of the SR
```

DIVS.(W)(L)(*) - Divide Step



Operation:

```
IF AC = 1
    THEN S <-- Ra + Rb
    ELSE S <-- Ra - Rb
F<msb:1> <-- S<msb-1:0>
F<0> <-- SR<31>
SR<31:1> <-- SR<30:0>
SR<0> <-- -S<msb>
```

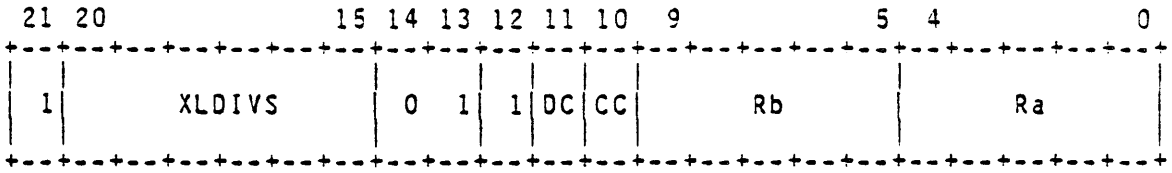
where msb is bit<31> for longword and bit<15> for word operations, S is the output of the ALU, F is the output of the shifter, and SR is the Shift Register.

Description: The AC flag is tested for a 0 or 1. If AC = 0, then register Rb is subtracted from register Ra. If AC = 1, then registers Rb and Ra are added together. The output of the ALU is rotated left with the msb of the Shift Register (SR) as the carry in. The SR is rotated left with the complement of the msb of the ALU result as the carry in. The msb of the ALU result becomes the new AC flag.

Status flags:

```
AN <-- msb of F
AZ <-- 0, if F <> 0
   <-- 1, if F = 0
AV <-- the exclusive-or of the result's AN-bit and AC-bit
AC <-- msb of S
```

XLDIVS(*) - Low Half of Extended Divide Step



Operation:

```
IF AC = 1
    THEN S <-- Ra + Rb; XC <-- arithmetic carry out
    ELSE S <-- Ra - Rb; XC <-- - arithmetic carry out
F<31:1> <-- S<30:0>
F<0> <-- 0
XR <-- S<31>
AC <-- AC
```

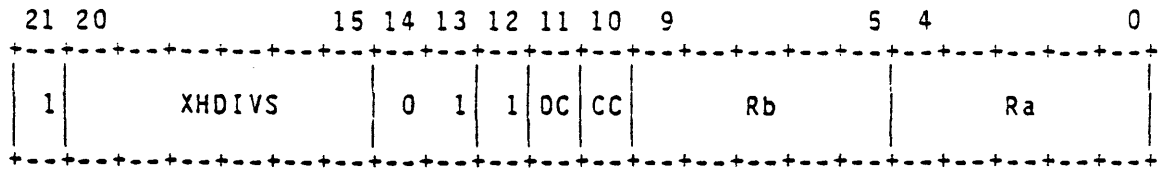
where S is the output of the ALU, F is the output of the shifter, XC is the extended precision carry flag, and XR is the extended precision rotate flag.

Description: The AC flag is tested for a 0 or 1. If AC = 0, then register Rb is subtracted from register Ra and the XC is loaded with the borrow out of the ALU. If AC = 1, then registers Rb and Ra are added together and the XC is loaded with the arithmetic carry out of the ALU. The output of the ALU is shifted left with a 0 as the carry in. The XR is loaded with S<31>. The AC flag is unchanged.

Status flags:

```
AN <-- F<31>
AZ <-- 0, if F <> 0
   <-- 1, if F = 0
AV <-- if AC = 1 then
       (set if operands' signs are same and result's sign
       is different; cleared otherwise)
       if AC = 0 then
       (set if operands' signs are different and result's
       and Rb's signs are same; cleared otherwise)
AC <-- unchanged
```

XHDIVS(*) - High Half of Extended Divide Step



Operation:

```
IF AC = 1
    THEN S <-- Ra + Rb + XC
    ELSE S <-- Ra - Rb - XC
F<31:1> <-- S<30:0>
F<0> <-- XR
SR<31:1> <-- SR<30:0>
SR<0> <-- -S<31>
AC <-- S<31>
```

where S is the output of the ALU, F is the output of the shifter, SR is the Shift Register, XC is the extended precision carry flag, and XR is the extended precision rotate flag.

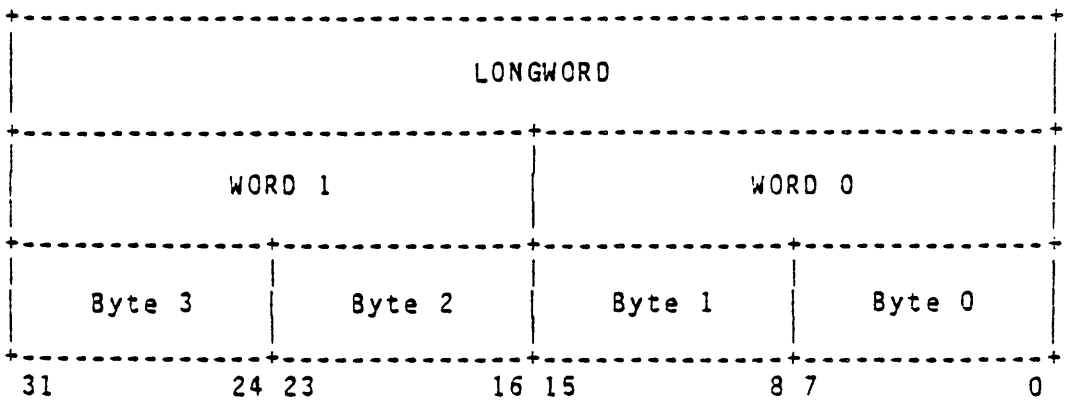
Description: The AC flag is tested for a 0 or 1. If AC = 0, then register Rb is subtracted with the borrow flag XC from register Ra. If AC = 1, then registers Rb and Ra are added together with the carry in flag XC. The output of the ALU is shifted left with the XR as the carry in. The Shift Register (SR) is rotated left with the complement of S<31> as the carry in. S<31> becomes the new AC flag.

Status flags:

```
AN <-- F<31>
AZ <-- 0, if F <> 0
   <-- 1, if F = 0
AV <-- the exclusive-or of the result's AN-bit and AC-bit
AC <-- S<31>
```

Appendix B - Condition Code Equations

Data Format:



Length Field (MIR<14:13>):

11 - byte
01 - longword
X0 - word

Upper Sixteen Bit Register Select:

MIR<4:0> = 01XXX

Symbols

. := logical AND
+ := logical inclusive OR
XOR := logical exclusive OR
- := logical inversion

Precedence Order

Highest () - . XOR + Lowest

CAPITALIZED variables are microinstruction names. Names which do not contain an operand length specifier (B/W/L) reference the generic microinstruction.

```
;; Decode equations

; byte swaps

swap    = SWAB + LLSW
```

```
;; Length equations

; longword select: longword

longword = -mir<14> . mir<13>

; word select: word, word0, word1

word   = -mir<13>
word0  = word . (mir<4> + -mir<3>)
word1  = word . -mir<4> . mir<3>

; byte select: byte, byte0, byte1, byte2, byte3

byte   = mir<14> . mir<13>
byte0  = byte . -swap . (mir<4> + -mir<3>)
byte1  = byte .  swap . (mir<4> + -mir<3>)
byte2  = byte . -swap . -mir<4> . mir<3>
byte3  = byte .  swap . -mir<4> . mir<3>


;; Intermediate Status latch equations

; buffered carries - ci_n refers to the carry into bit_n
;                    co_n refers to the carry out of bit_n

ci7(1) = ci7 latched at T150
co7(1) = co7 latched at T150
ci15(1)= ci15 latched at T150
co15(1)= co15 latched at T150
ci31(1)= ci31 latched at T150
co31(1)= co31 latched at T150


; buffered right shift carry out of the SR: sr<0>(1)

sr<0>(1)= sr<0> latched at T0

; buffered version of LSB of the s bus: s<0>(1)

s<0>(1)= s<0> latched at T150


;; AN, AZ, AV, AC latch

an     = ani latched at T0
az     = azi latched at T0
av     = avi latched at T0
ac     = aci latched at T0

;; ANI equations

ani    = a<31> . (longword + word1 . -swap + byte3) +
         a<23> . (word1 . swap + byte2) +
```

```
          a<15> . (word0 . -swap + byte1) +
          a<7> . (word0 . swap + byte0)

;; AZI equations

; zero byte conditions: zb0, zb1, zb2, zb3

zb0     = -a<7> . -a<6> . -a<5> . -a<4> . -a<3> . -a<2> . -a<1> . -a<0>
zb1     = -a<15> . -a<14> . -a<13> . -a<12> .
          -a<11> . -a<10> . -a<9> . -a<8>
zb2     = -a<23> . -a<22> . -a<21> . -a<20> .
          -a<19> . -a<18> . -a<17> . -a<16>
zb3     = -a<31> . -a<30> . -a<29> . -a<28> .
          -a<27> . -a<26> . -a<25> . -a<24>

; extend Z flag microinstruction: ez

ez      = SUBC + ADDC + NEGC

; propagate Z (PS<2>) flag -- sticky Z operations: pz

pz      = -ez + z

azi     = pz . § zb0 . zb1 . zb2 . zb3 . longword +
                 zb0 . zb1 . word0 . -swap + zb2 . zb3 . word1 . -swap +
                 zb0 . word0 . swap + zb2 . word1 . swap +
                 zb0 . byte0 + zb1 . byte1 + zb2 . byte2 + zb3 . byte3 †


;; ACI equations

; carry out of the ALU's most significant bit (MSB): msbco

msbco   = co31(1) . longword +
          co15(1) . word +
          co7(1) . byte

; carry into the ALU's MSB: msbci

msbci   = ci31(1) . longword +
          ci15(1) . word +
          ci7(1) . byte


; most significant S bus bit: mssb

mssb    = s<31> . longword +
          s<15> . word +
          s<7> . byte

; buffered versions of the most significant S bus bit (mssb): mssb(1)

mssb(1) = mssb latched at T150
```

```
; shift left microinstructions: shl1

shl1    = ASL + ROL + ASLQ + XHDIVS + DIVS

; shift right microinstructions: shr1, shr2

shr1    = ASR + ROR + LSR
shr2    = LSRQ + SMULS + UMULS

; conditional subtract microinstruction execution conditions: csube

csube   = DIVS . ac

; subtract operations: subt

subt    = NEG + NEGC + CMP + COM + SUB + DEC + SBC + LSUB +
          LCMP + LLCM + SOBC + ACNEG + ANNEG + CNEG + NNEG + csube

; microinstructions which copy C (PS<0>) into AC: ec

ec      = INC + DEC + XOR + BIT + MOV + SXT + BIC + BIS + AND +
          MOVPM + MOVS + LXOR + LBIT + LLD + LBIC + LBIS + LAND +
          LCNTR + LMSTK

; microinstuctions which copy AC into AC: eac

eac     = XLDIVS

; other special case AC microinstructions: spec1

spec1   = MTST

; special case conditions: sign0, sign1

sign0   = -a<31> . -a<30> . -a<29> . -a<28> . -a<27> . -a<26> . -a<25> .
          -a<24> . -a<23> . -a<22> . -a<21> . -a<20> . -a<19> . -a<18> .
          -a<17> . -a<16> . -a<15>

sign1   = a<31> . a<30> . a<29> . a<28> . a<27> . a<26> . a<25> .
          a<24> . a<23> . a<22> . a<21> . a<20> . a<19> . a<18> .
          a<17> . a<16> . a<15>

aci     = -shl1 . -shr1 . -shr2 . -ec . -eac . -spec1 .
                                          (subt xor msbco(1)) +
           shl1                          . mssb(1)          +
                 shr1                    . s<0>(1)          +
                       shr2              . sr<0>(1)         +
                             ec          . c                +
                                   eac   . ac               +
                                   spec1 . -(sign0 + sign1)

;; AVI equations

; sticky left shift microinstructions: sshl
```

```
sshl   = ASL

; sticky V flip-flop: svff

svff:  set = -prdc . sshl . (ani xor mssb(1))
       clear = prdc

; buffered sticky V flip-flop: svff(1)

svff(1)= svff latched at T150

avi    = -shll . -shrl . -shr2 . (msbco(1) xor msbci(1)) +
          shll                 . (ani xor mssb(1))       +
                   shrl        . (ani xor s<0>(1))       +
                          shr2 . (ani xor sr<0>(1))      +
                                  sshl . svff(1)
```

J-11 MICROINSTRUCTION OPCODES

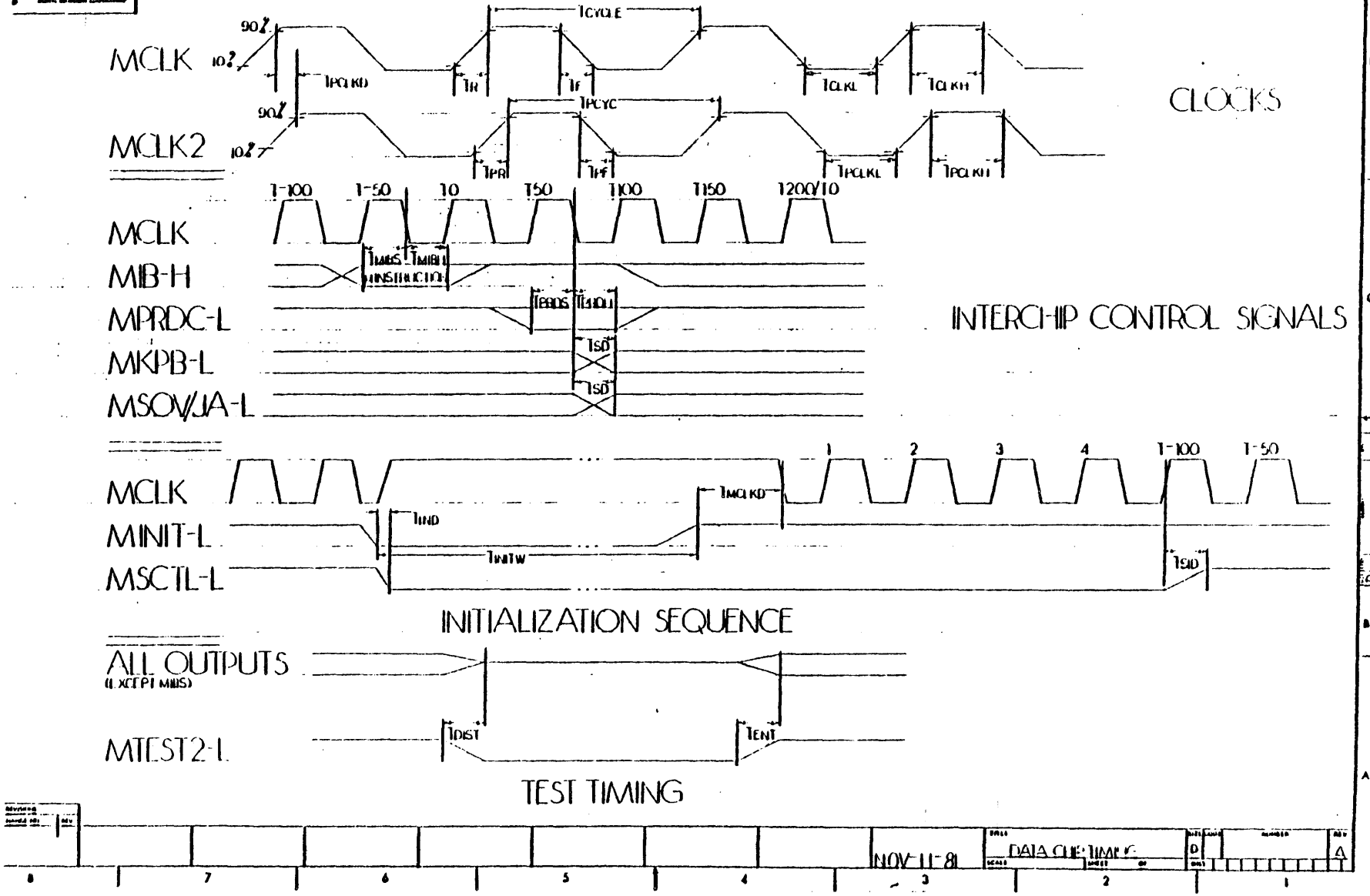
Octal	Binary	Mnemonic	K0	K1	K2	K3	L0	L1	L2	SUBT	EC
000	0000000	JMP									
001	0000001	JZ									
002	0000010										
003	0000011	JC									
004	0000100										
005	0000101	JV									
006	0000110										
007	0000111	JN									
010	0001000										
011	0001001	JAZ									
012	0001010										
013	0001011	JAC									
014	0001100										
015	0001101	JAV									
016	0001110	JKM									
017	0001111	JAN									
020	0010000										
021	0010001	INPR									
022	0010010										
023	0010011										
024	0010100										
025	0010101	OUTR	1	1							
026	0010110	OUTC									
027	0010111	OUTS									
030	0011000	RDI									
031	0011001	RDF									
032	0011010	RD									
033	0011011	RMW									
034	0011100										
035	0011101										
036	0011110	RDG									
037	0011111	RDINTR									

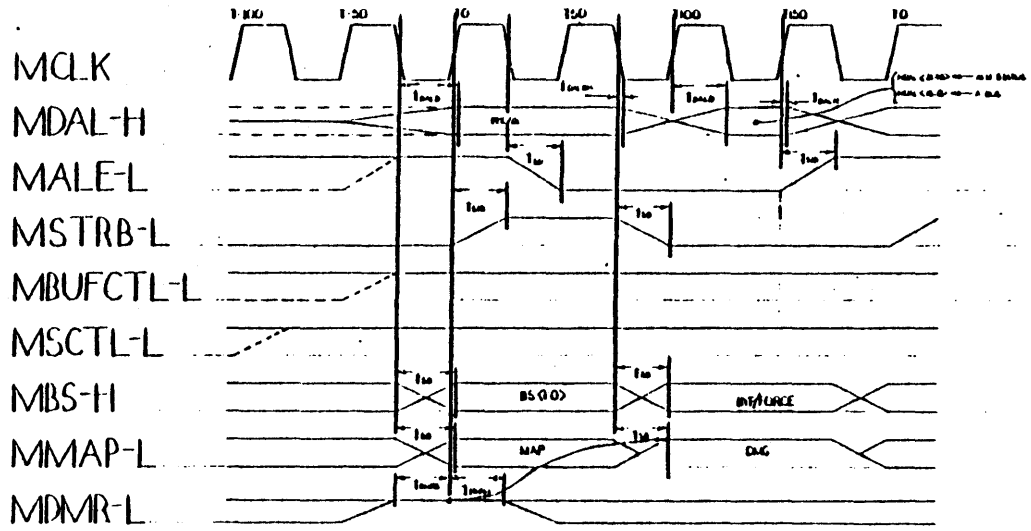
Octal	Binary	Mnemonic	K0	K1	K2	K3	L0	L1	L2	SUBT	EC
040	0100000		X			X		X		X	
041	0100001	LSUB	1			1		1		1	
042	0100010										
043	0100011	LCMP	1			1			1	1	
044	0100100	LADD		1	1		1				
045	0100101			X		X	X		X	X	X
046	0100110	LLSW	1		1						
047	0100111	LLCM		1		1				1	
050	0101000	LBIT	1								1
051	0101001	LAND	1								1
052	0101010	LCNTR	1		1						1
053	0101011	LMSTK	1		1						1
054	0101100	LXOR		1	1						1
055	0101101	LBIC		1							1
056	0101110	LLD	1		1						1
057	0101111	LBIS	1	1	1						1
060	0110000										
061	0110001										
062	0110010										
063	0110011	ARG									
064	0110100										
065	0110101										
066	0110110										
067	0110111	AWG									
070	0111000	ARI		1	1		1				
071	0111001	ARD	1			1		1			
072	0111010										
073	0111011	AR									
074	0111100	AWI		1	1		1				
075	0111101	AWD	1			1		1			
076	0111110										
077	0111111	AW									

Octal	Binary	Mnemonic	K0	K1	K2	K3
100	1000000					
101	1000001					
102	1000010	MOVS	1		1	
103	1000011	CNEG	-C	C	-C	C
104	1000100	CLR				
105	1000101	DIVS	-AC	AC	AC	-AC
106	1000110	SXT	N	N	N	N
107	1000111	NNEG	-N	N	-N	N
110	1001000	UMULS	-SR<0>	1	SR<0>	
111	1001001	XLDIVS	-AC	AC	AC	-AC
112	1001010	MOVPM	1		1	
113	1001011	ACNEG	-AC	AC	-AC	AC
114	1001100	SMULS	-SR<0>	1	SR<0>	
115	1001101	XHDIVS	-AC	AC	AC	-AC
116	1001110					
117	1001111	ANNEG	-AN	AN	-AN	AN
120	1010000	LSR				
121	1010001	ROL				
122	1010010					
123	1010011	ASL				
124	1010100	ROR	1		1	
125	1010101					
126	1010110					
127	1010111					
130	1011000	LSR.Q	1		1	
131	1011001	ASL.Q	1		1	
132	1011010					
133	1011011	NEG		1		1
134	1011100	ASR	1		1	
135	1011101					
136	1011110	INC	1		1	
137	1011111	NEGC		1		1

Octal	Binary	Mnemonic	K0	K1	K2	K3	L0	L1	L2	SUBT	EC	EZ
140	1100000	SOBC	1			1		1		1		
141	1100001	SUB	1			1		1		1		
142	1100010											
143	1100011	CMP	1			1			1	1		
144	1100100	ADD		1	1		1					
145	1100101	DEC		1		1	1		1	1	1	
146	1100110	SWAB	1		1							
147	1100111	COM		1		1				1		
150	1101000	BIT	1								1	
151	1101001	AND	1								1	
152	1101010		X		X						X	
153	1101011		X		X						X	
154	1101100	XOR		1	1						1	
155	1101101	BIC		1							1	
156	1101110	MOV	1		1						1	
157	1101111	BIS	1	1	1						1	
160	1110000	ADDC		1	1		1					1
161	1110001	SBC		1		1	1		1	1		
162	1110010	RSYNC										
163	1110011	CRSYNC										
164	1110100	ADC	1		1							
165	1110101	SUBC	1			1		1				1
166	1110110	TST	1		1							
167	1110111	MTST			1	1						
170	1111000	PDAD		1	1		1					
171	1111001											
172	1111010	AOBC		1	1		1					
173	1111011											
174	1111100	DADP		1	1		1					
175	1111101											
176	1111110											
177	1111111	NOP										

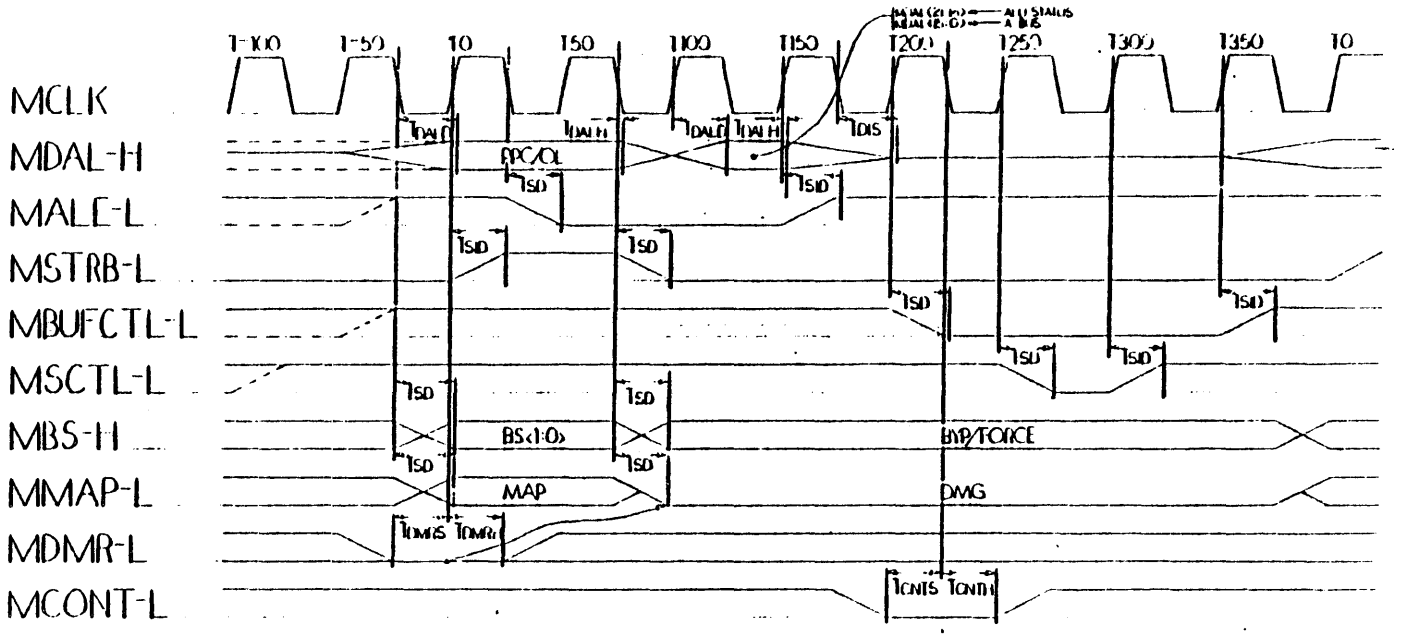
Octal	Binary	Mnemonic	L0	L1	L2	SUBT	EC	EZ	SHLL	SHR1	SHR2
100	1000000										
101	1000001										
102	1000010	MOVS					1				
103	1000011	CNEG									
104	1000100	CLR									
105	1000101	DIVS	AC	-AC		AC			1		
106	1000110	SXT				1					
107	1000111	NNEG									
110	1001000	UMULS	SR<0>								1
111	1001001	XLDIVS	AC	-AC							
112	1001010	MOVPM					1				
113	1001011	ACNEG				1					
114	1001100	SMULS	SR<0>								1
115	1001101	XHDIVS	AC	-AC					1		
116	1001110										
117	1001111	ANNEG				1					
120	1010000	LSR								1	
121	1010001	ROL	1		1				1		
122	1010010										
123	1010011	ASL	1		1				1		
124	1010100	ROR								1	
125	1010101										
126	1010110										
127	1010111										
130	1011000	LSR.Q									1
131	1011001	ASL.Q							1		
132	1011010										
133	1011011	NEG				1					
134	1011100	ASR								1	
135	1011101										
136	1011110	INC					1				
137	1011111	NEGC				1		1			


MCLK
MCLK2
TCYCLE
TPCLKD
TR
TF
TCLKL
TCLKH
TPCYC
TPR
TPF
TPCLKL
TPCLKH
CLOCKS
T-100
T-50
T0
T50
T100
T150
T200/T0
MCLK
MIB-H
TMIBS
TMIBH
INSTRUCTION
MPRDC-L
TRDS
TRDH
MKPB-L
TSD
MSOV/JA-L
INTERCHIP CONTROL SIGNALS
1
2
3
4
T-100
T-50
MCLK
MINIT-L
MSCTL-L
TIND
TMCLKD
TINITW
TSID
INITIALIZATION SEQUENCE
ALL OUTPUTS
(EXCEPT MBIS)
MTEST2-L
TDIST
TENT
TEST TIMING
NOV 11 81
DATA CUE TIMING
D
A



NON-STRETCHED IO TIMING

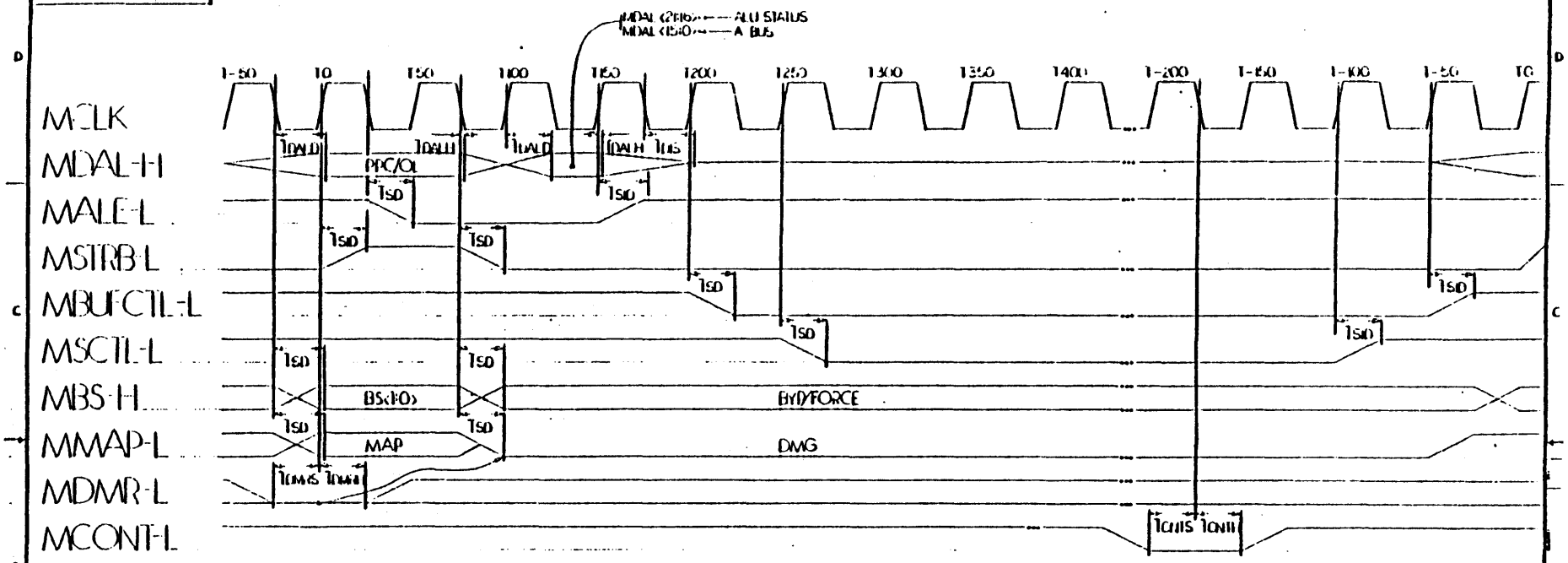
NOV-11-81 DATA CHIP TIMING



400NS NON_IO TIMING

NOV-11-81

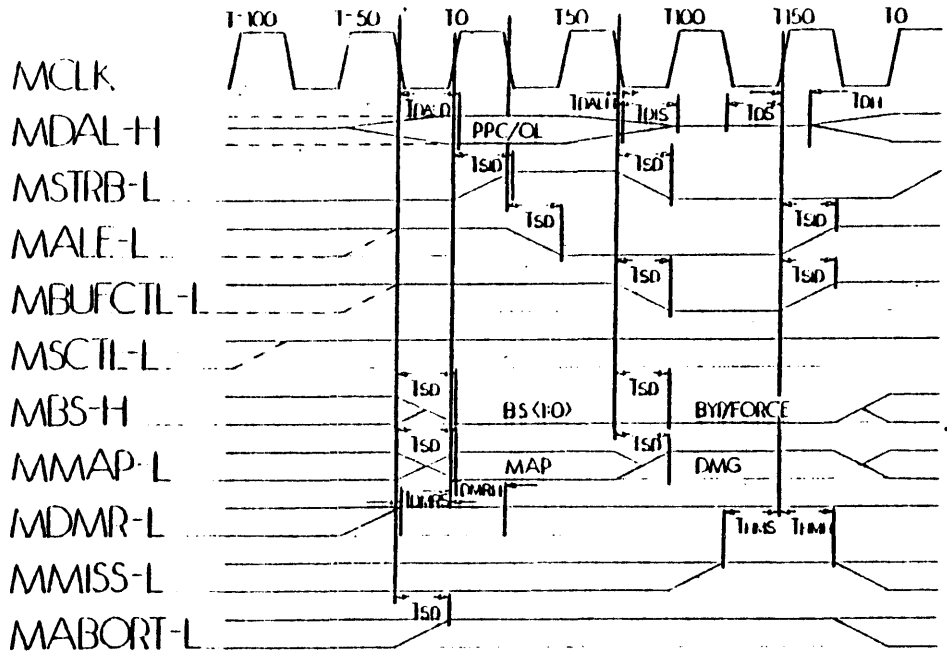
DATA CHIP TIMING



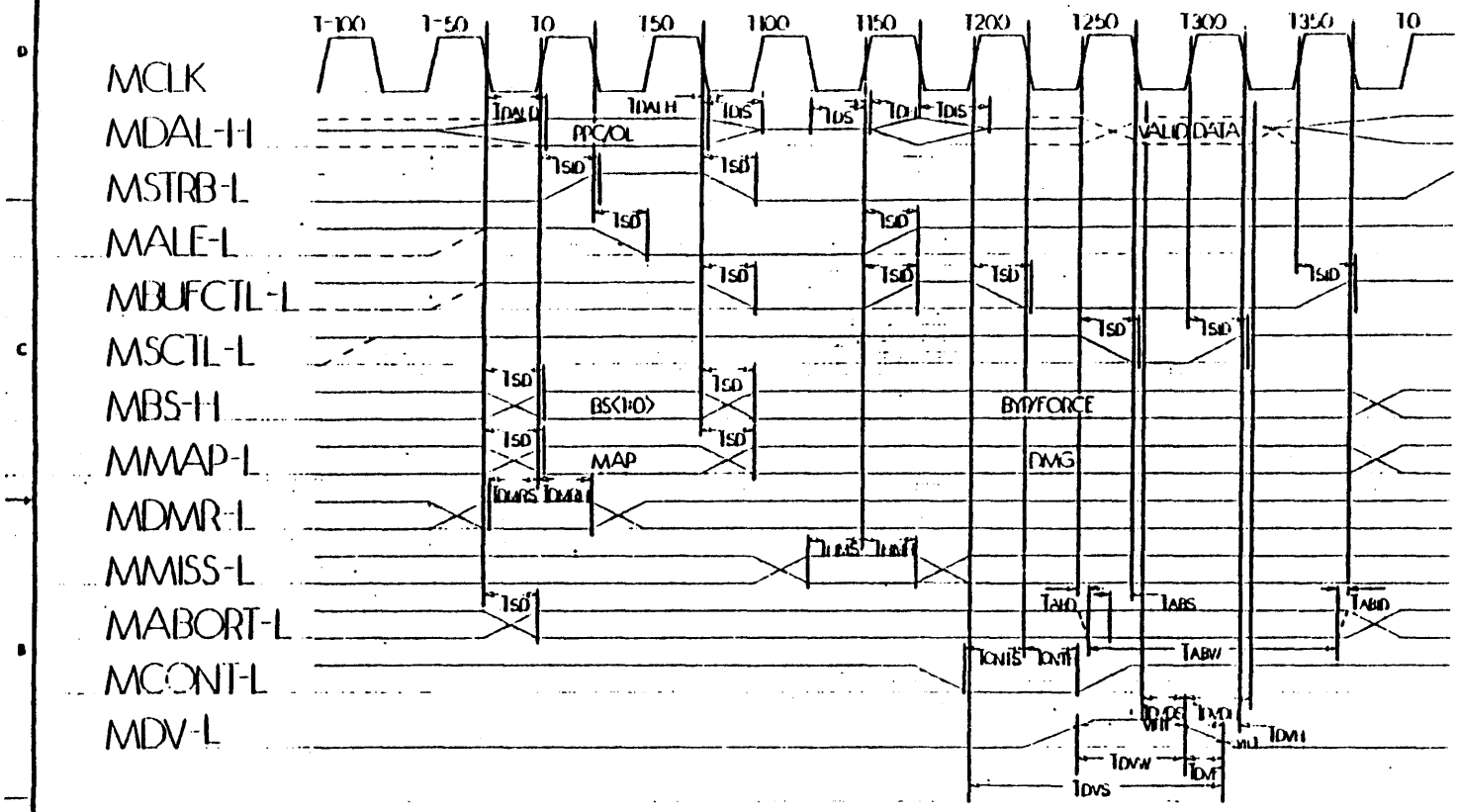
STRETCHED NON-IO (GENERAL CASE)

NOTE: NON IO INCLUDES ALL INTERNAL TO MICROINSTRUCTIONS

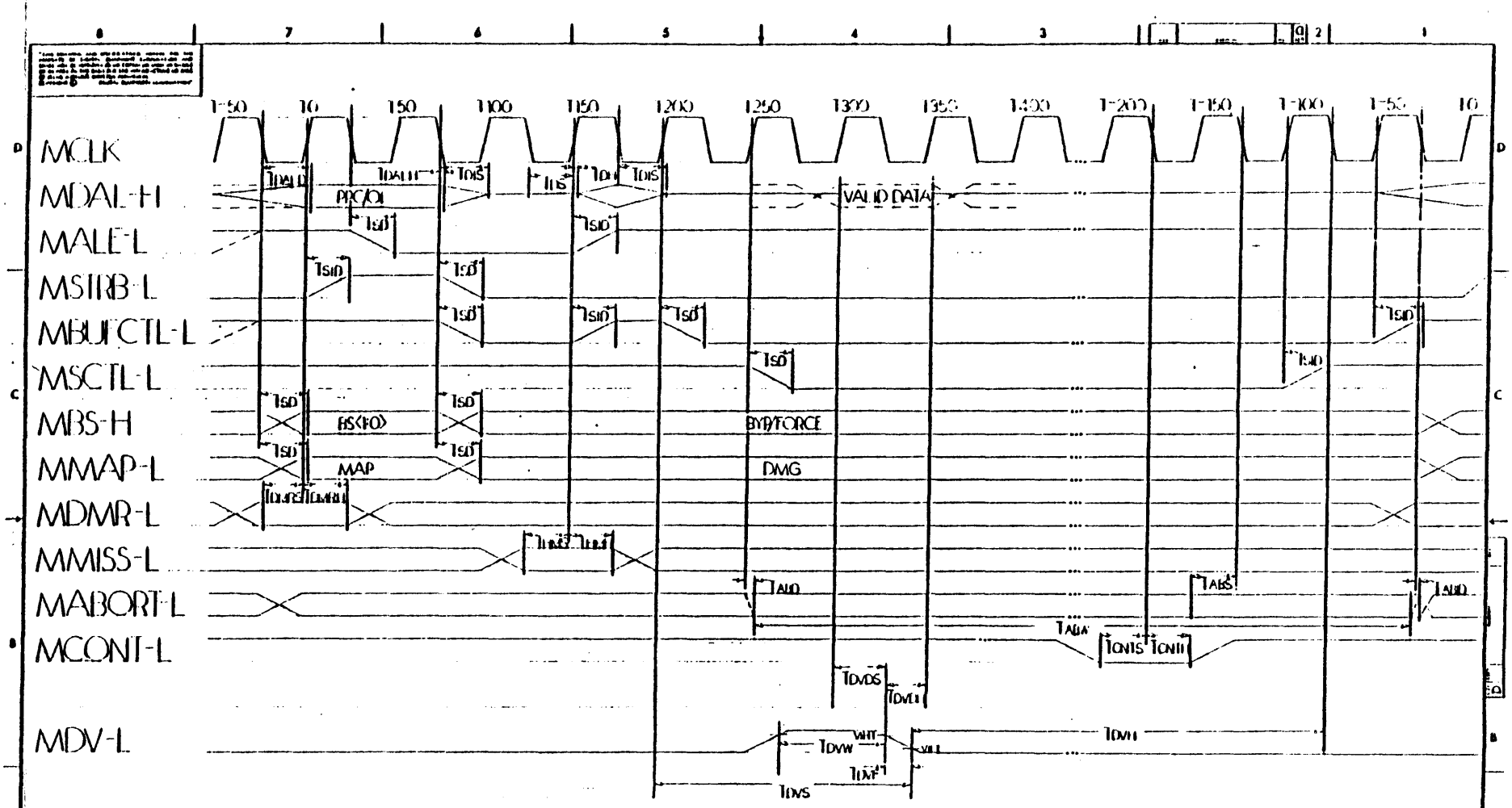
NOV-11-81



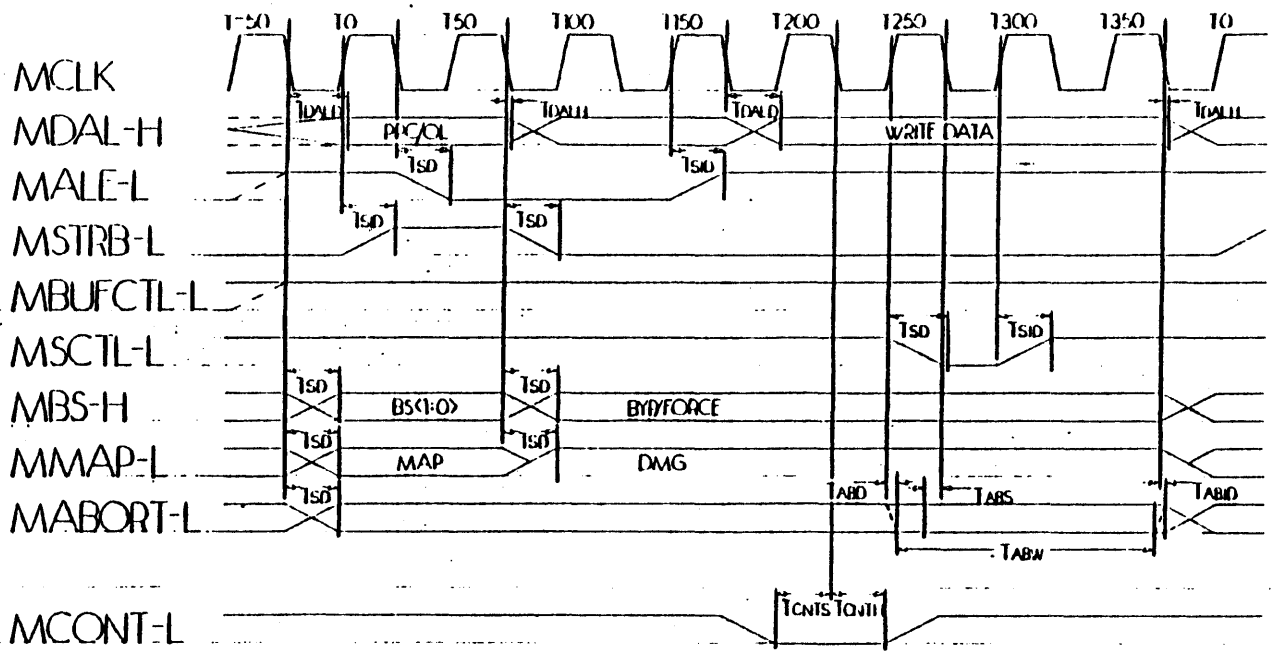
NON-STRETCH READ TIMING



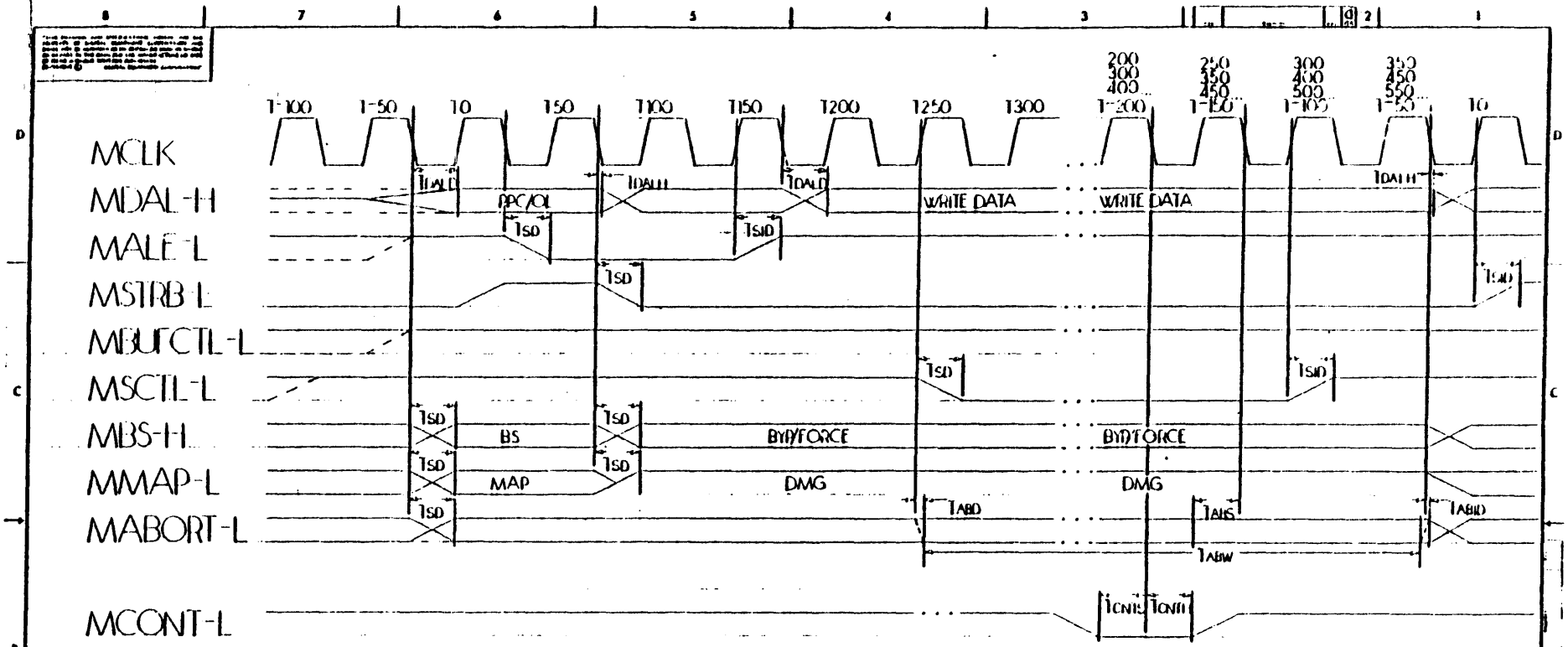
400 NS READ TIMING



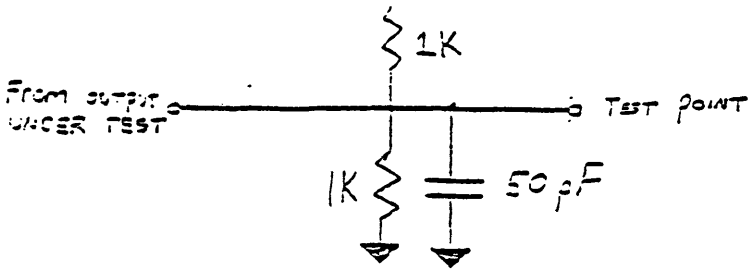
STRETCHED READS (GENERAL CASE)



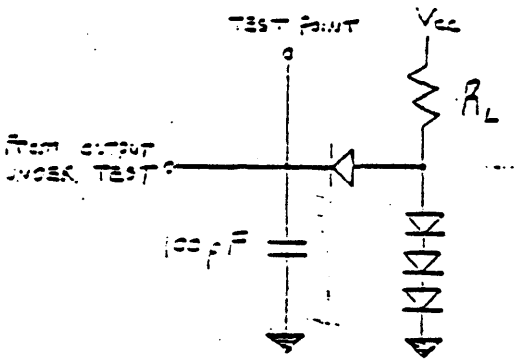
400 NS WRITE TIMING



WRITE TIMING (GENERAL CASE)



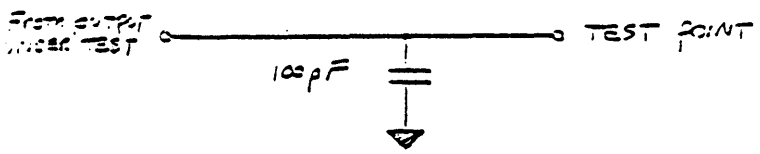
TEST CIRCUIT 1 - TRISTATE DISABLE.



ALL DIODES ARE EITHER 1N916 OR 1N3064.

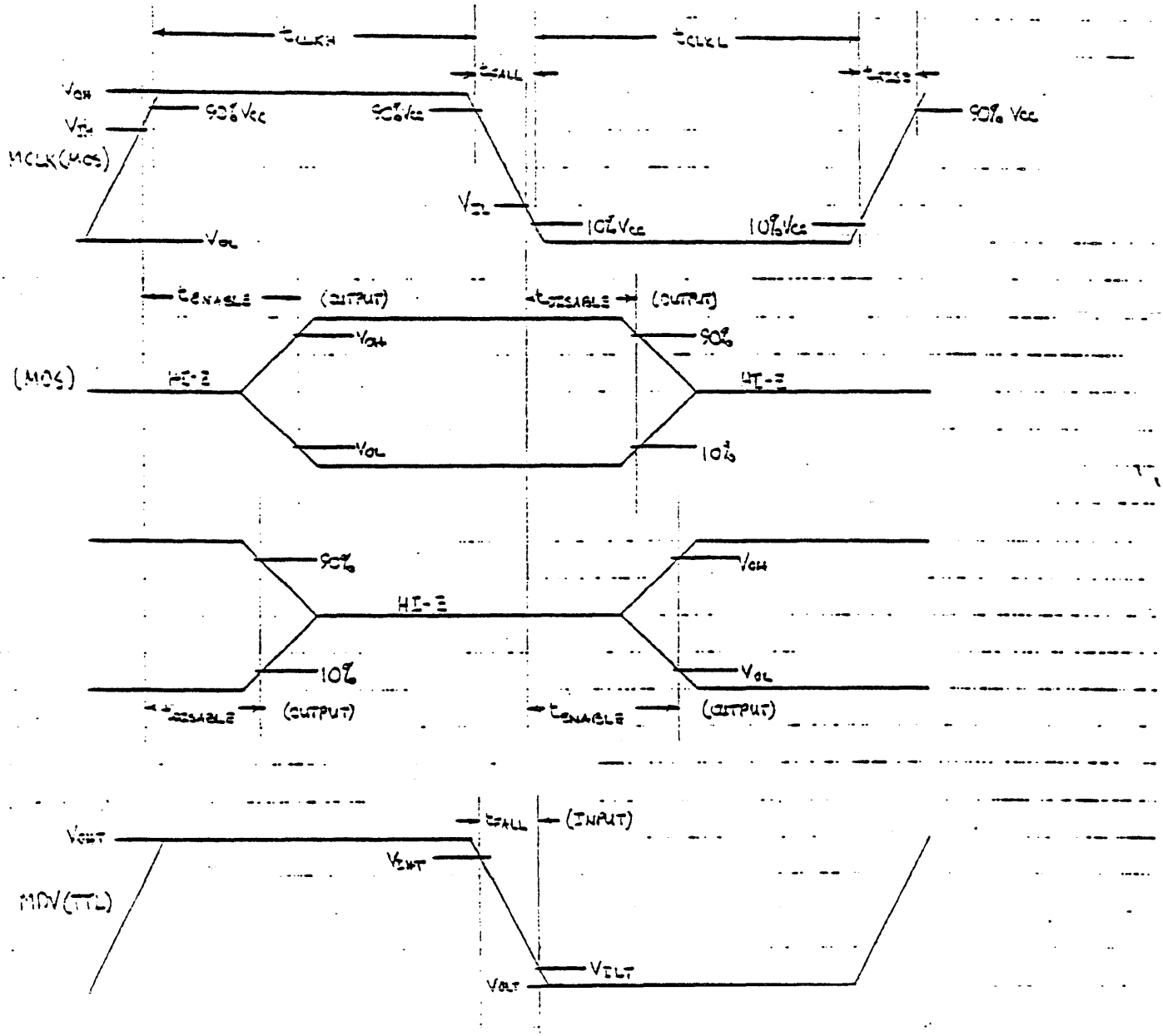
R_L is SELECTED to PROVIDE I_{OL} of 2ma @ .4 VOLTS.

TEST CIRCUIT 2 - TTL OUTPUT



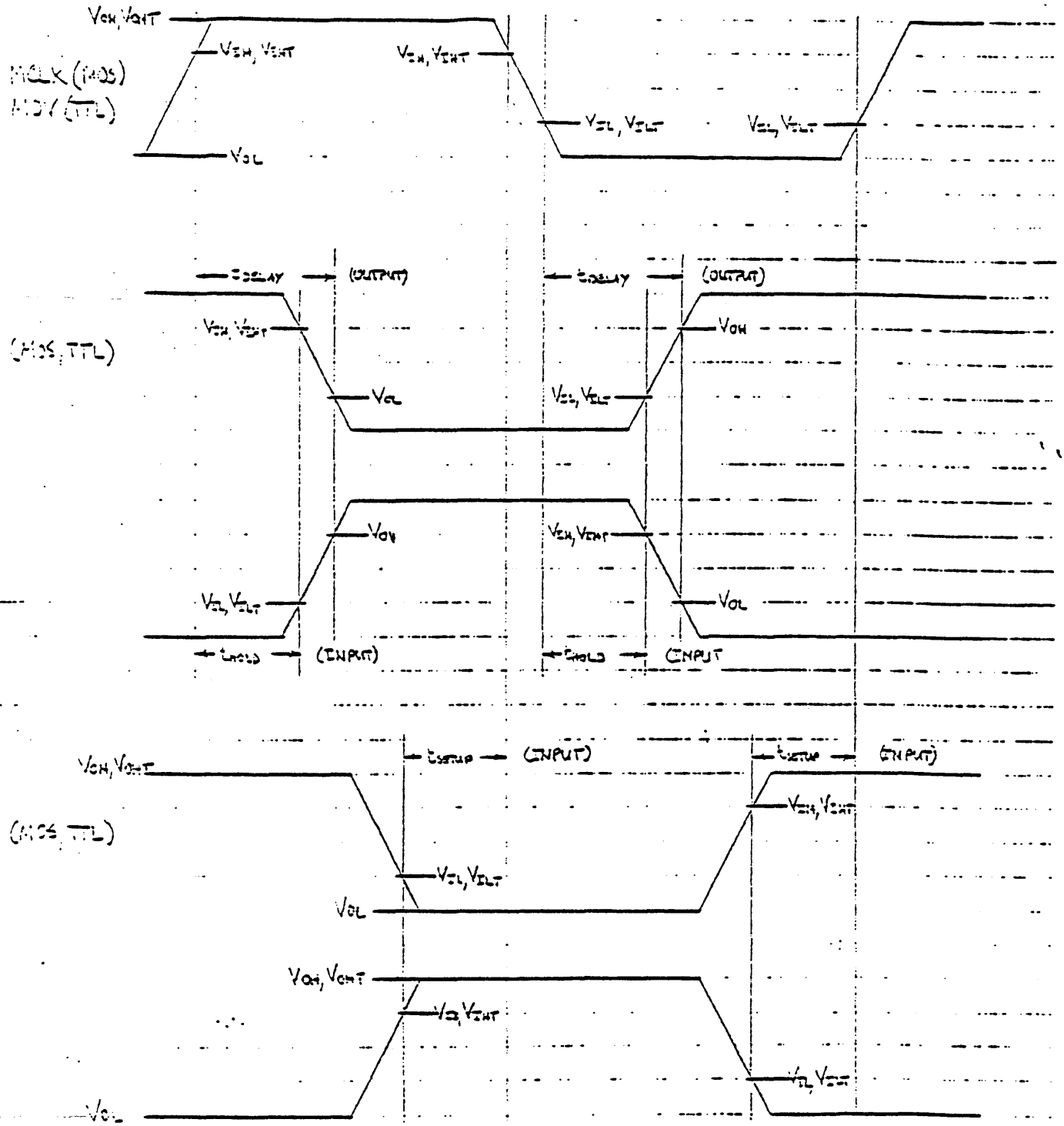
TEST CIRCUIT 3 - MOS OUTPUT

AC TEST CIRCUITS



$V_{OH} = V_{cc} - 0.4v$

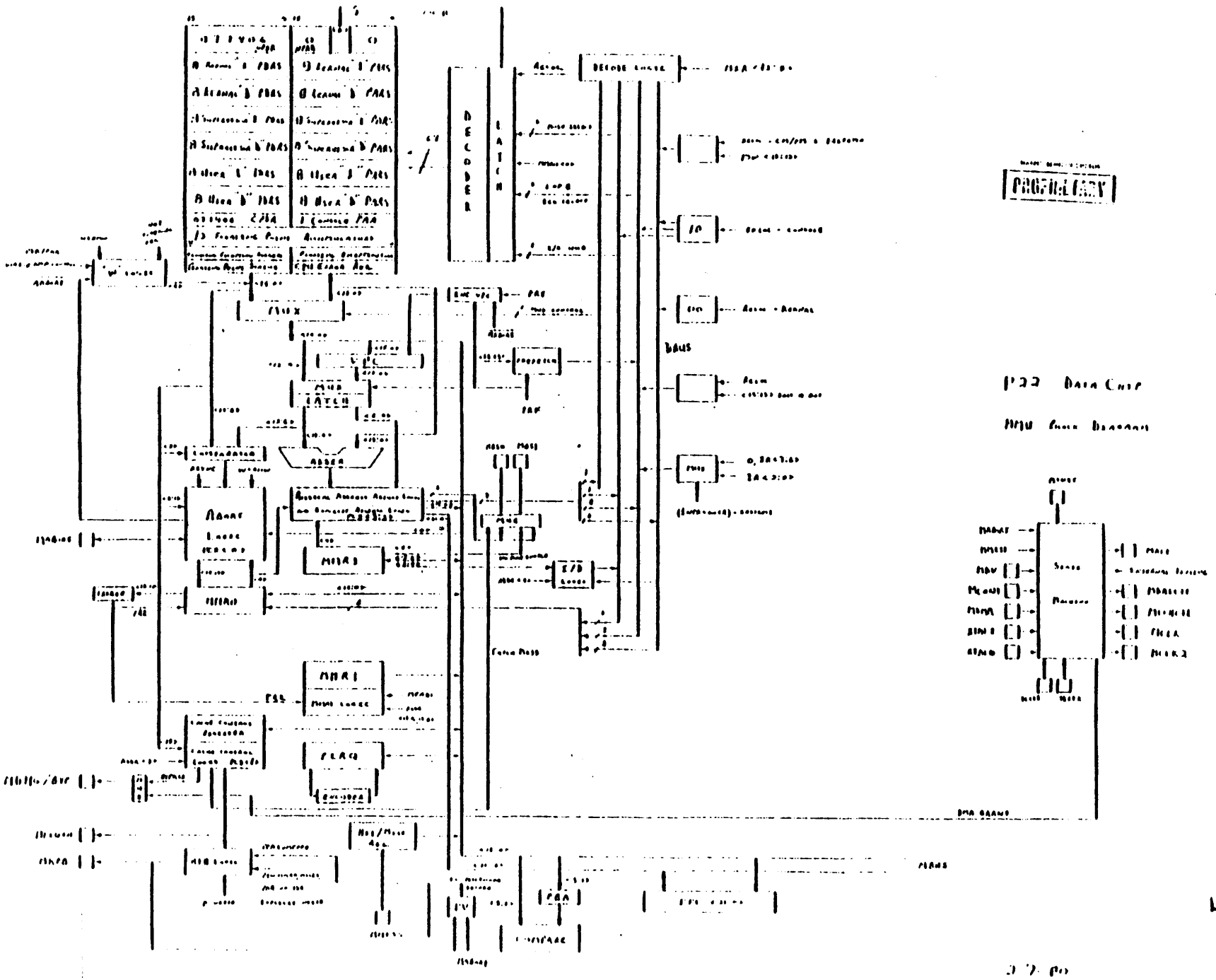
$V_{OL} = 0.4v$

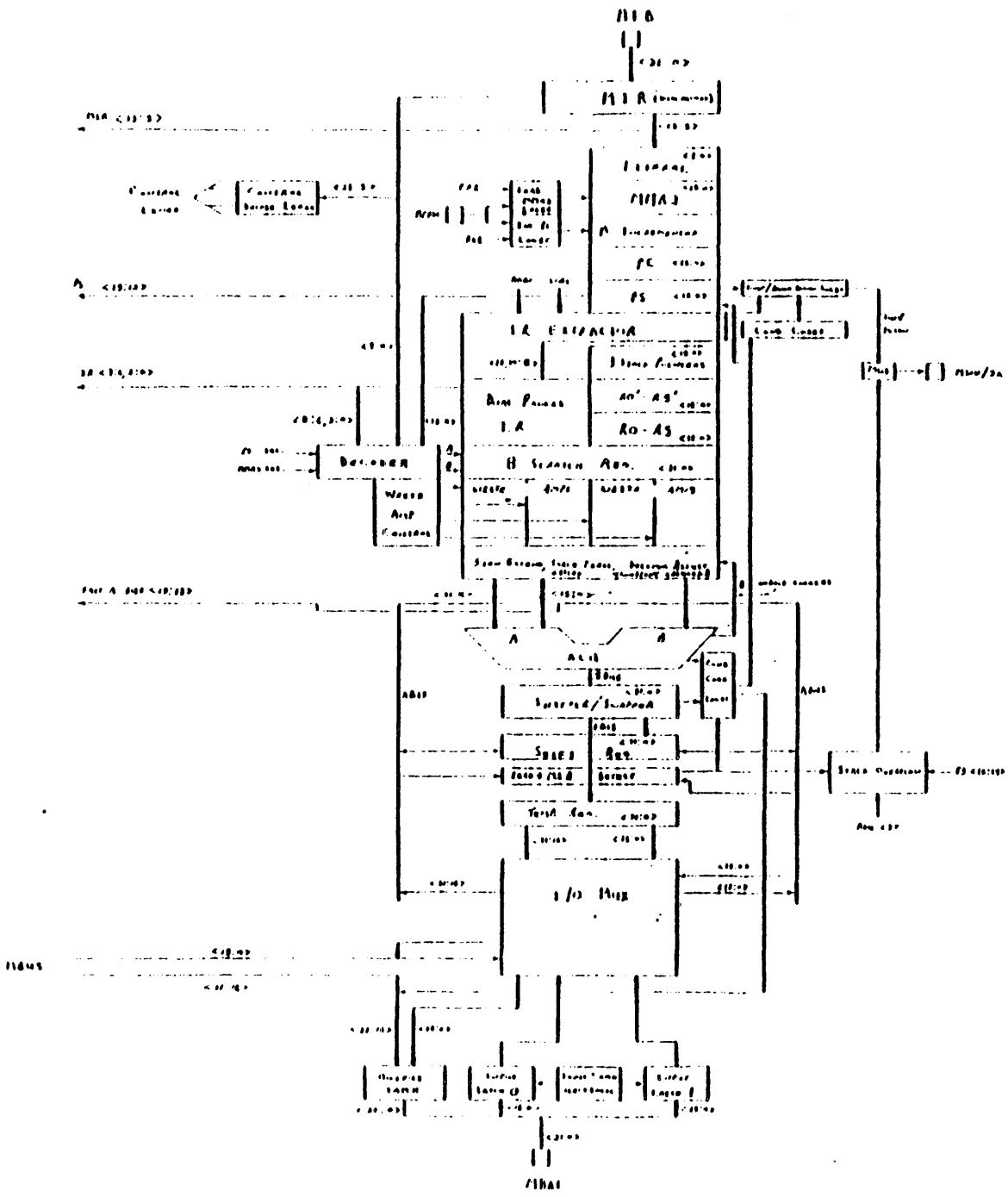


$V_{OH} = V_{CC} - 0.4v$

$V_{OL} = 0.4v$

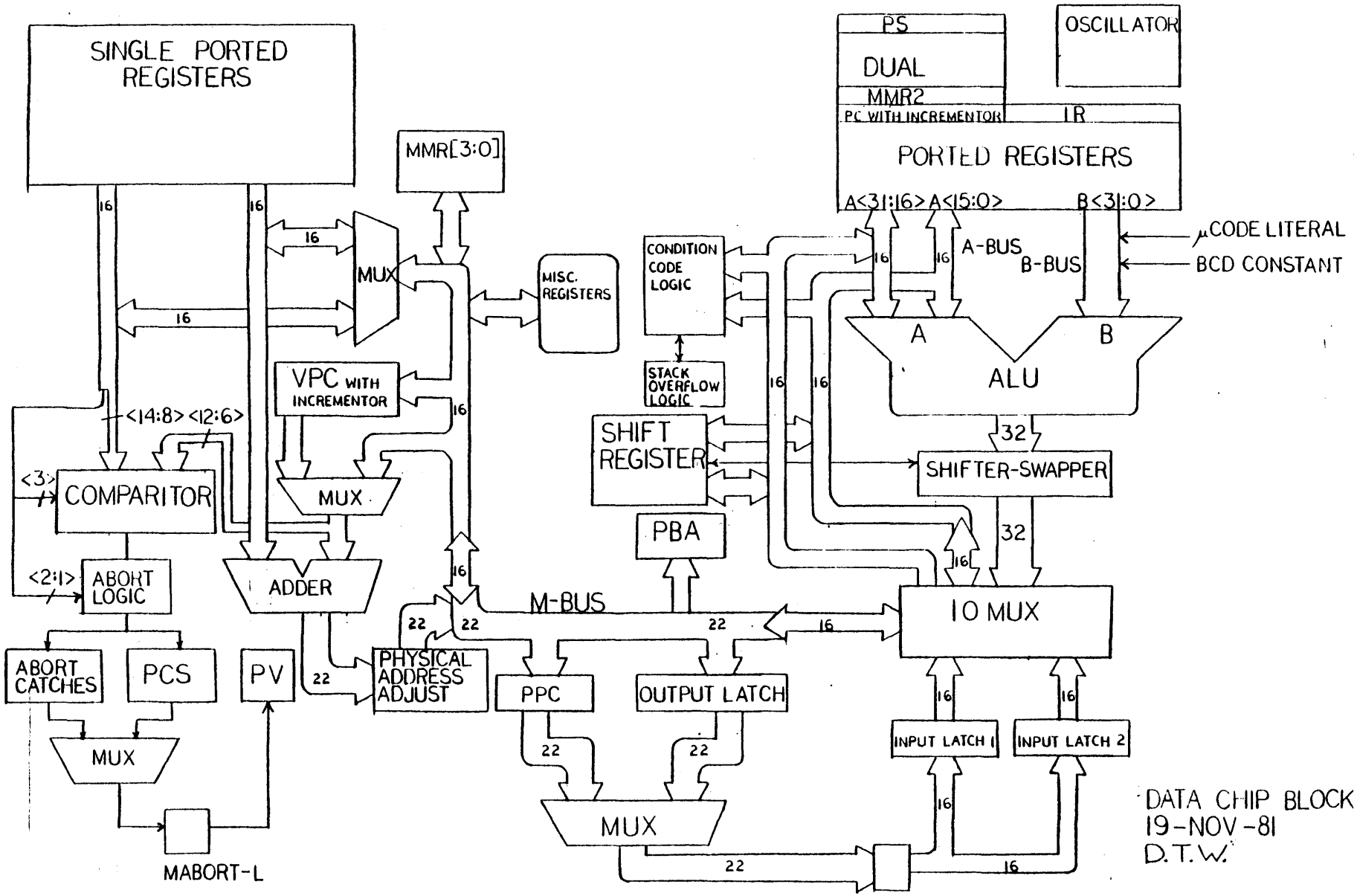
1/25/91


DECODER
LATCH
BUS



P22 Data Chip

F 11 Block Diagram



DATA CHIP BLOCK DIA.
19-NOV-81
D.T.W.