FAIRCHILD
A Schlumberger Company

USER GUIDE

F9450

FAIRCHILD
A Schlumberger Company

Fairchild F9450 User Guide

February 1986

Microcontroller Division

Table of Contents

General Description

This manual is an introduction and guide to the Fairchild F9450, a 16-bit high-performance control processor.

The F9450 is the center of a family of high-speed devices intended for commercial and military applications requiring sophisticated real-time processing. The F9450 instruction set is optimized for this purpose, and implements the complete MIL-STD-1750A Instruction Set Architecture (ISA) on a single chip.

The F9450 executes instructions at extremely high speed: for example, 0.25 microseconds for an ADD, 1.85 microseconds for an integer multiply. Its overall instruction throughput is 1.5 MIPS. It is also remarkably small, contained on a 0.6-inch-width 64-pin DIP package. The chip, which drives standard TTL I/O components, uses I^3L bipolar technology. The advantages of I^3L are high immunity to radiation effects, an operating temperature range of −55°C to +125°C, and operating speeds up to 20 MHz.

The F9450 possesses, on chip, complete floating-point capabilities without the use of a math coprocessor. Other on-chip facilities that make the F9450 particularly well suited for real-time applications are a 16-leve! prioritized interrupt structure, comprehensive fault handling, and two user-programmable timers. Multiprocessing is supported by flexible bus arbitration and process synchronization.

The basic F9450 can be augmented with two subsidiary microcontrollers that implement optional enhancements to the MIL-STD-1750A environment. The Fairchild F9451 Memory Management Unit (MMU) provides memory expansion capabilities allowing up to 16M of random-access memory. The Fairchild F9452 Block Protect Unit (BPU) permits 1K pages of memory to be selectively write-protected. These two controllers can be configured to operate either separately or together in connection with an F9450.

Fairchild furnishes programming software and other development aids for the F9450. The basic software component is the MACRO-50 macro assembler, a cross-assembler currently available for VAX/VMS and under development for the IBM PC. Early in 1986, Fairchild will release a full Kernighan and Ritchie standard C compiler for the F9450 that runs on the VAX/VMS, and a DoD-validated Ada compiler is tentatively planned for release in 1986. As to hardware aids, Fairchild manufactures the SBC-50 single-board computer, an evaluation and development board centered around the F9450. The U. S. government and other vendors also provide a wealth of design aids, including development systems, compilers, and real-time operating executives.

MIL-STD-1750A ISA Advantages

The F9450 microprocessor implements the standard government-mandated 1750A Instruction Set Architecture. The military and its suppliers have a significant commitment to the 1750A technology. As a result, the availability of design aids is guaranteed, and an extensive set of utilities for 1750A processors has evolved under the direction of the federal government. A considerable body of software with full documentation is available free of charge from the USAF Language Control Facility.

The 1750A standard was developed to fulfill a very broad set of military requirements for embedded control computers. The flexibility, standardization, and power of 1750A and the small size and high performance of the Fairchild F9450 make it suitable for both military and commercial controller applications.

About This Book

This User Guide describes the Fairchild F9450 microprocessor and associated products. The terms F9450 and 1750A are to a large extent interchangeable, inasmuch as the F9450 is a total embodiment of the standard. Throughout this book, the term 1750A refers to the government standard and the environment it defines, while F9450 refers specifically to the Fairchild product that implements that standard. MIL-STD-1750A can be obtained without charge from the U. S. Department of Defense, Washington, D. C. 20360. Where conflicts arise between this book and the standard, the standard should be assumed to be correct.

Chapters Two through Seven present engineering information suitable for system designers, organized into a logical progression of topics. Chapters Eight and Nine describe the optional F9451 Memory Management Unit and the F9452 Block Protect Unit, respectively. Chapter Ten discusses development tools—both hardware and software—available as of this writing from Fairchild. Four appendices furnish details regarding the F9450 instruction set, instruction execution times, and pinouts.

Table 2.2 **Fault Register Bit Assignments**

Bit	Fault	M	U	W
0	CPU Memory Protect Error			
	Data Cycle	X		
	Instruction Fetch		X	
1	Non-CPU Memory Protect Error			X
2	Memory Parity Error			
	Data Cycle			X
	Instruction Fetch		X	
3	Spare (always 0)			
4	Spare (always 0)			
5	Illegal I/O Address	X		
6	Spare (always 0)			
7	System Fault 0			X
8	Illegal Memory Address			
	Data Cycle	X		
	Instruction Fetch		X	
9	Illegal Instruction		X	
10	Privileged Instruction	X		
11	Address State Error		X	
12	Spare (always 0)			
13	BITE (Built-In Test) or			
	System Fault 1 Unclassified			X
14	Spare (always 0)			
15	System Fault 1			X

Table 2.3 **Flags in SW Register**

Flag	Bit	Purpose
C	0	Carry
P	1	Positive
Z	2	Zero
N	3	Negative

The reserved field (SW bits 4–7) is always zero-filled. Programmers should not attempt to use these bits for any purpose.

The Access Key/Processor State (AK/PS) bits serve two purposes:

- Determine the legal/illegal criteria for privileged instructions. A privileged instruction is executed with PS = 0 only. An attempt to execute a privileged instruction with PS not equal to 0 causes a Major Error, sets bit 10 in the Fault Register, and causes an instruction abort.
- Define the Access Key that is used in systems with an MMU to match an Access Lock.

Address State (AS) defines a page register group in the F9451 MMU. For configurations without an MMU, an Address State fault is generated (bit 11 in the Fault Register is set) for any operations attempting to modify the AS field to a nonzero value. This condition is also tested during interrupt processing. An interrupt service status word (where the new AS does not equal 0) is aborted if the MMU bit of the System Configuration Register is not set. To implement the AS bits as an extension of the address field in configurations without an MMU, the processor must be initialized upon reset with the System Configuration Register indicating the presence of an MMU.

All usable bits of the status word can be modified under program control or from the console.

System Configuration Register (SCR)

The first five bits of the SCR define to the F9450 internal control circuitry the configuration of the external system connected to the CPU. The remaining eleven bits are unused. MIL-STD-1750A does *not* define this register, and as a result there are no programming instructions for reading or manipulating it.

In order to load the SCR, external hardware must respond to an I/O Read request at I/O address 8410_{16} during either Reset initialization or the execution of a breakpoint (BPT) instruction. If any of the following configuration options is active, the responding external device at 8410_{16} must furnish a 1-bit in the appropriate position. *Table 2.4* shows the meaning of each bit.

Table 2.4 SCR Bits

Bit	Meaning
0	Memory Management Unit Present = 1 if an F9451 MMU is connected in the system. This must be set if the application ever requires AS not equal to 0. (Use of the AS bits to extend address field requires that this bit be set.)
1	Block Protect Unit Present = 1 if an F9452 BPU is connected in the system.
2	Console Present = 1 if a console is connected in the system.
3	Coprocesssor = 1 if a coprocessor is connected in the system. If this bit is a 0, the "Built-In Function" (BIF) instruction is illegal.
4	Interrupt Mode. This bit selects the interrupt mode for the Power Down and User 1 through User 5 signals: 1 is level-sensitive, 0 is edge-sensitive (low to high).

Timer A and Timer B

The F9450 has two 16-bit timers that are loaded, started, read, and halted under software control. Timer A "ticks" at 100 kHz and Timer B at 10 kHz. Both timers count up. When Timer A and Timer B reach their terminal counts (roll over from 0FFFF hex to zero), they set the corresponding bit in the PIR, thus generating an interrupt.

Although implemented in the F9450 as registers, the timers appear within the 1750A instruction set as external devices and are thus managed through I/O instructions.

The recommended method for starting a timer is:

1. Determine what time interval you need in terms of 10 kHz or 100 kHz.
2. Load the complement of the interval into R0.
3. Issue an XIO to load the selected timer with the value in R0 and start it running. These instructions are:

XIO R0,OTA	To load and start Timer A
XIO R0,OTB	To load and start Timer B

Both timers halt when the CPU enters console mode and continue when program execution resumes.

48-bit object beginning in R15 concatenates to R0 and R1, in that order. The register address is the register where the object begins; in the preceding example, instructions referring to the 48-bit object would give its location as R15.

Pending Interrupt Register (PIR)

Mask Register (MK)

The F9450 has sixteen levels of on-chip prioritized interrupts, of which:

- Nine are external (PIR0, PIR2, PIR8, PIR10-PIR15) with:
  - Two level-sensitive (I/O Levels 1 and 2).
  - Seven level- OR edge-sensitive (User 1 – User 5 and Power Down). Level or edge sensitivity is set via the interrupt mode bit in the SCR.
  - Seven are internal (PIR1, PIR3-PIR7, PIR9).

An interrupt signals the F9450 for service by latching its associated bit in the PIR. The highest-level interrupt corresponds to the most significant bit of the PIR (bit 0), with priorities descending to bit 15. *Table 2.1* maps interrupts to PIR bit positions. The programmer can elect to ignore one or more types of interrupts by loading a mask into the MK, subject to the limitations noted in *Table 2.1*. A 0-bit in the MK masks the PIR interrupt associated with that bit position.

The F9450 interrupt servicing routine automatically saves the MK register before activating the interrupt handler. Chapter Four discusses interrupts in more detail.

Except for I/O Levels 1 and 2, all external interrupts have hysteresis circuitry for noise immunity.

Fault Register (FT)

The Fault Register serves as a source of additional information when a Machine Error interrupt occurs. In effect, the Machine Error interrupt is notification of an exception condition, and the FT shows what happened. Three levels of severity are recorded in the FT: major (M), unrecoverable (U), and unclassified (W). *Table 2.2.* shows the FT bit assignment and severity level for each type of fault. See Chapter 5 for a more comprehensive description of this register and its role.

Table 2.1 Pending Interrupt and Mask Registers

Interrupt	PIR/MK bit	Source
Power Down[1]	0	External
Machine Error[2]	1	Internal
User 0	2	External
Floating-Point Overflow	3	Internal
Fixed-Point Overflow	4	Internal
Executive Call[1]	5	Internal
Floating-Point Underflow	6	Internal
Timer A	7	Internal
User 1	8	External
Timer B	9	Internal
User 2	10	External
User 3	11	External
I/O Level 1	12	External[3]
User 4	13	External[3]
I/O Level 2	14	External
User 5	15	External

Instruction Counter (IC)

The Instruction Counter Register holds the address of the next instruction to be executed. During interrupt servicing and subroutine calls, the content of this register is saved in memory so that it can later be fetched and restored to the IC Register, thereby resuming execution where it left off. The F9450 interrupt servicing routine automatically saves the IC register before activating an interrupt handler.

Status Word Register (SW)

The Status Word Register contains flags and control information. Its configuration is:

0 3	4 7	8 11	12 15
Flags	Reserved	AK(PS)	AS

The flags reflect the results of the most recent arithmetic operation as shown in *Table 2.3.*

Note

[1] Cannot be masked via the MK register or disabled with the DSBL instruction.

[2] Cannot be disabled with DSBL.

[3] Level-sensitive only.

Register Set

Introduction

The F9450 contains the 24 programmer-accessible registers shown in *Figure 2-1*. R0 through R15 are 16-bit general-purpose registers. The other registers are for dedicated:

- Pending Interrupt Register (PIR).
- Mask Register (MK).
- Fault Register (FT).
- Instruction Counter (IC).
- Status Word Register (SW).
- System Configuration Register (SCR).
- Two Timers (A and B).

The PIR, MK and FT are described in detail in connection with interrupts in Chapter Four. The others are described in this chapter.

In addition, the F9450 has six temporary registers (A1, A2, D01, D02, Q1, and Q2) internal to the operation of the F9450 and accessible only from the console. These are excluded from the programmers' model.

Register bits are numbered 0–15, with 0 the most significant bit.

General Registers

All F9450 general registers can be used as 16-bit accumulators. In addition, certain registers have attributes that enable them to be used for special applications, as follows:

- R1 through R15 can be index registers. R0 cannot.
- R12 through R15 can be base registers for instructions using Base Relative addressing.
- R15 is the implicit stack pointer for the PSHM and POPM (push and pop multiple) instructions.
- Register pair R0, R1 act as the accumulator for double precision and floating point.
- R2 is the accumulator for single precision, and single precision multiply and divide in Base Relative mode use R3 as well.

Data objects larger than sixteen bits occupy adjacent registers in left-to-right order with the most significant bits on the left. Thus, 32-bit objects take two adjacent registers and 48-bit objects, three registers. The register set can wrap, with R15 wrapping to R0. For example, a

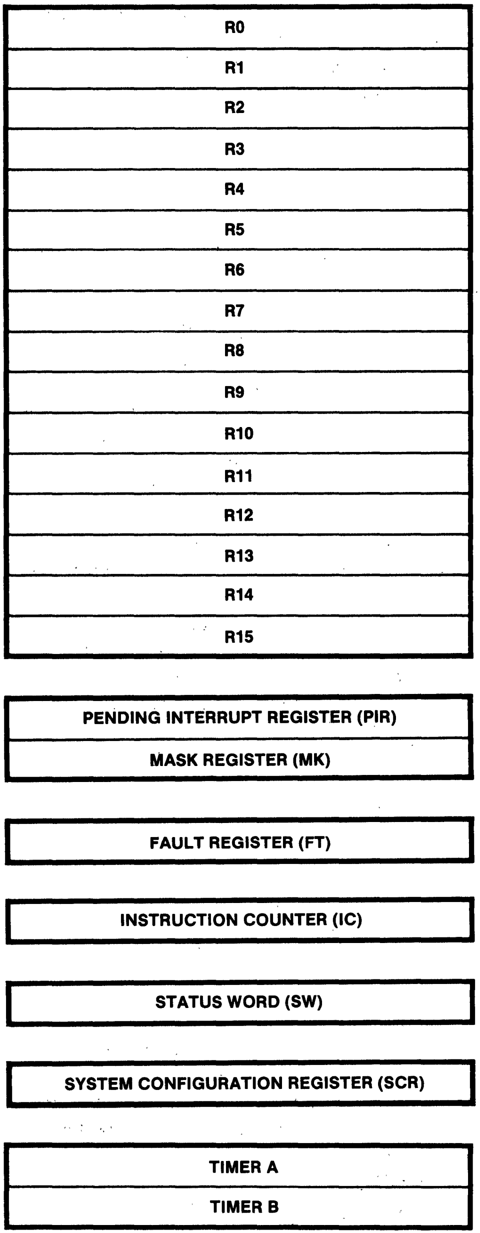


Figure 2.1 F9450 Programmer's Register Model

writing the appropriate values into ADDR. Note that the instruction does not save old pointers prior to the jump. This is an efficient method for branching to a different address space when no return is desired.

Stack Management

The F9450 supports stacks in user memory for two purposes: temporary storage of register contents, and calling and returning from subroutines.

A stack is a LIFO (last-in, first-out) data structure that grows downward in memory as items are added, and shrinks upward as they are removed. A stack pointer is a general register assigned by the programmer to keep track of the stack. This register always points to the item most recently pushed onto the stack, or to the start of the stack (its highest memory address plus one) if the stack contains nothing. When an item, always a 16-bit word, is pushed onto the stack, the stack pointer decrements, then furnishes its contents as the write address. When an item is popped off the stack, the item is copied into an instruction-designated register from the address indicated by the stack pointer, and then the stack pointer increments. A stack is always in the active address space indicated by the AS bits in the SW register.

Push/pop multiple

One type of stack operation is the saving and retrieving of multiple general registers, accomplished with the PSHM and POPM instructions respectively. Because these instructions use R15 as the implied stack pointer, the programmer must ensure that R15 points to a valid stack area at least large enough to hold the number of register words to be pushed/popped. The instructions loop internally to copy the specified registers, repeating the steps described in the preceding paragraph.

The format of the push instruction, which copies registers onto the stack, is:

```
PSHM RA, RB
```

The range of registers RA through RB may be specified in either order, and the instruction wraps around the register set according to the following rules:

- When RA ≤ RB, the contents of RB through RA move to the stack.
- When RA > RB, the order of copying is RB through R0, then R15 through RA.

In both cases, R15 decrements by one for each register pushed and ends up pointing to the last item, which is a copy of RA. The effect of this scheme is that successive increasing addresses on the stack correspond to successive increasing register numbers, except that there is a discontinuity between R15 and R0 when the sequence wraps.

The format of the POPM instruction, which removes multiple entries from the stack and copies them into a series of registers, is:

```
POPM RA, RB
```

The following rules apply:

- If RA ≤ RB, registers RA through RB are loaded sequentially from the stack.
- When RA > RB, registers RA through R14, then R0 through RB are copied from the stack in order.

Note that R15 is not overlaid in the second case. It is valid to copy R15 to the stack with the PSHM instruction; the POPM algorithm later retrieves the pushed copy of this register, but it effectively disregards it and fetches the next (R0) value without corrupting the stack pointer.

R15 increments by one for each word popped from the stack. After the instruction completes execution, R15 points to the new "top" of the stack, which contains the item most recently pushed onto the stack before the push/pop sequence occurred.

Subroutine calls/returns

The second stack mechanism involves calling and returning from subroutines, accomplished by the SJS and URS instructions, respectively. These are considerably simpler operations than those just described. They are also more flexible, in that any register R0–R15 can be used as the stack pointer. In fact, a complex program might use several stacks and assign a different register to each. The programmer selects the desired stack by coding the appropriate register name in stack-affecting instructions.

The SJS instruction is a mnemonic for "Stack IC and jump to subroutine." It is similar to the CALL instruction in many other assembly languages. It takes two forms:

```
SJS RA,LABEL

SJS RA,LABEL,RX
```

The first is a direct jump, the second indexed-direct. RA is the stack pointer and indicates a memory address one word above the location where the current content of the IC register is to be saved. In a direct call, the stack pointer RA decrements, the IC register is pushed onto the stack at the address indicated by RA, and the address of LABEL (the entry point of the subroutine) is loaded into the IC. Execution continues from that point.

The indexed-direct call is slightly more complex. In this case, RX holds an offset that, when added to LABEL, results in a derived address that is the entry point of the subroutine. This permits a form of vectoring. An unusual feature of the 1750A instruction set allows the programmer to use the same register as both stack pointer and index register. The rationale is that one could push the return address into the word immediately preceding the entry point of the subroutine. Because the derived address is calculated before decrementing the stack pointer, the result is a correctly-positioned return address and a correctly-computed entry point. Subroutines called with this technique are necessarily non-reentrant, since any call overlays the return address of the previous caller.

The URS ("Unstack IC and return to subroutine") instruction is similar to the RET instruction in many other assembly languages. Its form is:

```
URS RA
```

in which RA is the stack pointer. It is assumed to point to the valid return address for the currently-executing subroutine.

Floating-Point Arithmetic

The F9450 carries out floating-point arithmetic on-chip and possesses a rich set of mathematical instructions for doing so. There is a choice of two formats: single-precision and extended-precision. These formats are depicted in *Figures 3.1* and *3.2*.

All floating-point arithmetic operations occur in concatenated registers, since the float types occupy two or three words. The register address of a floating-point object is the name of the register in which the object originates, i.e. its high-order or leftmost register.

Both float types are normalized, meaning that the most significant 1-bit of the mantissa appears in the most significant numeric bit of the field, with the exponent adjusted appropriately. The processor performs normalization automatically. The single-precision mantissa consists of 23 numeric bits and a sign; extended-precision is 39 numeric bits and a sign. The exponent field of both types is an 8-bit sign-extended integer expressing a power of two in the range -128 through $+127$. This provides a range of absolute numeric values from $|2.4652 \times 10^{-39}|$ to $|1.4272 \times 10^{38}|$ for both types. Single-precision expresses approximately nine decimal digits of numeric accuracy, while extended-precision expresses approximately seventeen.

FAIRCHILD

A Schlumberger Company

Data Organization and Addressing

This chapter discusses data formats, memory addressing, stack management, and other matters related to the movement of data objects in F9450 systems. It does not cover systems implementing the F9451 Memory Management Unit; Chapter Eight treats that subject.

Data Types

The F9450 instruction set provides for operations on six types of data:

- Bit
- Byte (8 bits)
- Word (16 bits)
- Double word (32 bits)
- Single-precision floating point (32 bits)
- Extended-precision floating point (48 bits)

The basic unit of data is a 16-bit word. All registers and memory locations are this size, and all data bus transactions take sixteen bits. Therefore, bits and bytes are submultiples of the F9450 16-bit data unit, and double words and floating point numbers are multiples of it.

A word is synonymous with an integer or a single-precision fixed-point number and occupies one memory location. A double word corresponds to a long integer or a double-precision fixed-point number; it consists of two concatenated words. All F9450 data formats are in two's complement form, and all words and word-multiple types have a sign in the most significant bit position, where 0 is positive and 1 is negative. The exception to this rule is when two bytes are packed into a word; character data are unsigned. In a double word, the sign appears in the high-order bit of the most significant word.

The number of a bit position is inverse to its significance. Bit 0 is the MSB, and bit 15 is the LSB.

The F9450 performs on-chip floating point and has an extensive instruction set in support of floating-point arithmetic. Two formats are available: single-precision and extended-precision. *Figures 3.1* and *3.2* show these formats. Floating-point numbers are represented by a fractional two's complement mantissa (24 bits for single-precision and 40 bits for extended-precision) and an 8-bit two's complement exponent. Mantissas are normalized. F9450 floating-point arithmetic is discussed in greater detail later in this chapter.

Figure 3.1 Single-Precision Floating-Point Format

S_m	Mantissa	S_e	Expon
0	1 23	24	25 31

Figure 3.2 Extended-Precision Floating-Point Format

S_m	Mantissa $_{MS}$	S_e	Expon	Mantissa L_s
0	1 23	24	25 31	32 47

Addressing Modes and Instruction Formats

The F9450 uses ten addressing modes. Because the smallest addressable unit of memory is a 16-bit word, a 16-bit pointer allows direct addressing of 64K words. This can be expanded to 2M words, as described later in this chapter under Address Space Implementation. There is no restriction on the location of double-word operands in memory, i.e. a double word may be aligned on an odd-numbered memory address.

Instruction formats for the following addressing modes are given in *Table 3.1.*

Register Direct (R)

The instruction directly specifies the register(s) containing and/or receiving the data object.

Memory Direct (D)

The instruction specifies the memory address of the operand.

Memory Direct—Indexed (DX)

The memory address of the required operand is derived by adding the contents of an index register to the instruction's address field. Registers R1 through R15 may serve as index registers.

Table 3.1 F9450 Addressing Modes and Instruction Formats

Mode	Format	Derived Operand (DO)		Derived Address (DA)	
		Single-Precision	Floating-Point and Double Precision	Single-Precision	Floating-Point and Double Precision
1. **Register Direct**	0 7 8 11 12 15 O.C. | RA | RB				
"R"		(RB)	(RB, RB + 1)	RB	RB, RB + 1
2. **Memory Direct**	0 7 8 11 12 15 16 31 O.C. | RA | RX | A				
"D"	RX = 0 (Non-Indexed)	[A]	[A, A + 1]	A	A, A + 1
"DX"	RX ≠ 0 (Indexed)	[A + (RX)]	[A + (RX), A + 1 + (RX)]	A + (RX)	A + (RX), A + 1 + (RX)
3. **Memory Indirect**	0 7 8 11 12 15 16 31 O.C. | RA | RX | A				
"I"	RX = 0 (Non-Indexed)	[|A|]	[|A|, |A| + 1]	[A]	[A], [A] + 1
"IX"	RX ≠ 0 (Indexed)	[|A + (RX)|]	[|A + (RX)|, |A + (RX)| + 1]	[A + (RX)]	[A + (RX)], [A + (RX)] + 1
4. **Immediate Long**					
a. **Not Indexable**	0 7 8 11 12 15 16 31 O.C. | RA | OCX | I				
"IM"		I			
b. **Indexable**	0 7 8 11 12 15 16 31 O.C. | RA | RX | I				
"IM"	RX = 0 (Non-Indexed)	I			
"IMX"	RX ≠ 0 (Indexed)	I + (RX)			
5. **Immediate Short**					
a. **Positive**	0 7 8 11 12 15 O.C. | RA | I				
"ISP"		+ (I + 1)			
b. **Negative**	0 7 8 11 12 15 O.C. | RA | I				
"ISN"		- (I + 1)			
6. **IC Relative** [1]	0 7 8 15 O.C. | DU				
"ICR"				DU + (IC - 1)	
7. **Base Relative**[2]					
a. **Not Indexable**[3]	0 5 6 7 8 15 O.C. | BR | DU				
"B"	BR = BR + 12	[DU + (BR)]	[DU + (BR), DU + 1 + (BR)]	DU + (BR)	DU + (BR), DU + 1 + (BR)
b. **Indexable**	0 5 6 7 8 11 12 15 O.C. | BR | OCX | RX				
"B"	RX = 0 (Non-Indexed)	[(BR)]	[(BR), (RR) + 1]	(BR)	(BR), (BR) + 1
"BX"	RX ≠ 0 (Indexed)	[(BR) + (RX)]	[(BR) + (RX), (BR) + 1 + (RX)]	(BR) + (RX)	(BR) + (RX), (BR) + 1 + (RX)

Notes:

1. $-128 \leq DU \leq 127$.
2. Base registers: BR = R12, R13, R14, and R15.
3. $0 \leq DU \leq 255$.
4. Extended-precision floating-point instructions require addressing of three operands located at DA, DA + 1, and DA + 2.

Any attempt to modify the AS bits without setting this configuration bit causes an Address State Fault (bit 11 in the Fault Register). Once the SCR is initialized, the AS bits can be toggled without generation of an Address State Fault. Thereafter, any of the three following methods may be used to change the AS bits.

Method A: BEX n instruction
This is the preferred way to change the AS bits. The BEX (BRANCH TO EXECUTIVE) instruction generates a special kind of interrupt (Executive Call, PIR bit 5) described in detail in Chapter Four. Briefly, the vector entry for BEX contains IC settings for up to sixteen executive routines, as *Figure 3.3* shows. The 'n' field of the BEX instruction specifies which IC entry to use. On executing a BEX, the F9450 saves the current contents of the MK, SW, and IC registers in the area of page 0 pointed to by the linkage pointer. The service pointer indicates the location of the new MK and SW. The BEX interrupt processor goes to the nth IC location to obtain the address of the desired routine.

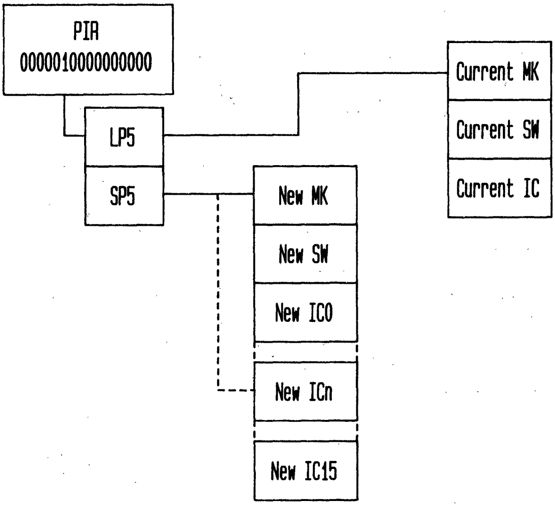


Figure 3.3 BEX n

The service linkage values replace the contents of the MK, SW, and IC registers. The programmer can specify the bit patterns of these new register contents. If the AS field of the new SW register contains a valid "page segment," that segment is automatically and immediately selected. The new IC value should contain the offset of the routine being entered within the new address space.

Note that interrupts are automatically disabled when the BEX instruction is executed, and the programmer must explicitly re-enable them after entering the new routine.

The best way to return is via the LS (Load Status) instruction, which restores the MK, SW, and IC contents saved when the BEX instruction executed.

Method B: XIO RA,WSW
This instruction (EXECUTE INPUT/OUTPUT, WRITE STATUS WORD) copies the contents of a general register into the SW. As soon as it executes, the value in the AS field becomes the currently-active address space, superseding that which prevailed at the start of the instruction's execution cycle.

This method has several problems. The basic purpose of this particular instruction is to set/clear the flags and AK(PS) bits in the SW register, not to change the AS bits. As a result, it offers no provision for saving and replacing the current processor status; if a return is desired at some future point, the programmer must explicitly change the processor status prior to issuing the instruction. Secondly and more importantly, it allows no control over the IC, which remains the same plus one. Therefore, the program continues execution in a new memory segment, but at the same offset (IC) as in the previous segment. For example, suppose the active address space is 0000 and the IC contains 34FAH when the XIO instruction loads 0001 into the AS bits of the SW register. The next instruction to execute will be at 34FBH, but in address space 0001. This method requires that the programmer have non-dynamic memory organization.

Method C: LST ADDR
Designed to be used for an unconditional jump, the LOAD STATUS instruction loads a new MK, SW, and IC from the specified memory location ADDR, and begins execution based on these new pointers. In Method A above, this instruction was recommended as the best way to accomplish a return. It can also jump unconditionally to a different address space and IC simply by

Memory Indirect (I)

The instruction-specified memory address contains the address of the operand.

Memory Indirect with Pre-Indexing (IX)

The contents of an index register plus the instruction's address field specify a memory address that contains the address of the operand. Registers R1 through R15 may be used for pre-indexing.

Immediate Long (IM)

One method of Immediate Long addressing allows indexing and one does not. If the specified index register field (RX) contains a reference to any register except R0, the content of the index register is added to the immediate field to form the operand. Otherwise, the RX field contains all zeros, and the immediate field is the operand without modification. This is because R0 cannot be used as an index register.

Immediate Short (IS)

The required 4-bit operand is contained within the instruction. Immediate Short Positive (ISP) treats the immediate operand as a positive integer between 1 and 16. Immediate Short Negative (ISN) treats the immediate operand as a negative integer between 1 and 16.

Instruction Counter Relative (ICR)

16-bit branch instructions use this mode. The content of the instruction counter minus one (i.e., the address of the current instruction) is added to the sign extended 8-bit displacement field of the instruction. The sum points to the memory address to which control will transfer. This mode allows addressing within a region of −128 to +127 words, relative to the address of the current instruction.

Base Relative (B)

The contents of the specified base register are added to the 8-bit displacement field of the instruction. The displacement field is a positive number between 0 and 255. The sum points to the memory address of the operand. This mode allows addressing within a memory region of 256 words, beginning at the address pointed to by the base register. Registers R12 through R15 can function as base registers.

Base Relative—Indexed (BX)

The sum of the contents of an index register and a base register is the address of the required operand. Registers R1 through R15 can act as index registers, and R12 through R15 as base registers.

Address Space Implementation

In compliance with MIL-STD-1750A, the F9450 uses sixteen address lines and a Data/Instruction (D/I) line to differentiate between data and instruction addresses. Sixteen address lines provide 64K words of address space; the D/I line has the effect of adding another address line, thus providing 128K addressable words, 64K of data and 64K of instructions.

In addition to these address lines, the 1750A standard calls for four Address State (AS) lines to implement extended addressing. The Fairchild F9451 Memory Management Unit utilizes the AS lines. In the absence of an MMU, these four lines can be used directly as address lines. This makes a total of 21 address lines, effectively increasing the F9450's addressable range to 2M words: 1M of data and 1M of instructions.

The 1750A architecture stipulates that the AS bits select memory "pages" or "segments," and must be explicitly changed in the AS field of the SW Register to enter a different address space.

If the application does not require conformance with the 1750A specification, there are three ways to extend the addressing range of an F9450 to a maximum of 2M words without utilizing an F9451 MMU. In all cases, note that: A) Fairchild does not certify these non-standard uses of the circuitry; B) most interpretations of 1750A do not permit extended address space without an MMU, and; C) the methods described here require that the CPU be initialized as though attached to an MMU.

On initialization after RESET, the processor performs a complete test of all major functions and initializes its internal registers in accordance with available information. (See Self-Test and Initialization in Chapter Four.) Since all extensions to the normal addressing range rely on the use of the AS bits in the SW register, the F9450 must be initialized as though attached to an MMU, by setting bit 0 in the SCR as described in Chapter Two.

Initialization and Exception Processing

Introduction

This chapter covers a number of F9450 operations that come under the general heading of exceptions: that is, deviations from normal processing, such as errors and interrupts. It discusses how the F9450 responds to exceptions and the implications for system designers and programmers. Exceptions in general come to the attention of the processor via the Pending Interrupt Register (PIR).

Covered here also is self-test and initialization. Whereas this is not an interrupt signaled via the PIR, it does cause the F9450 to abort current activity and is thus in the class of an exception.

Self-Test and Initialization

The self-test and initialization sequence, whose flowchart is shown in *Figure 4.1*, begins by assertion of $\overline{\text{RESET}}$ signal (pin 5). This is normally high, so $\overline{\text{RESET}}$ is signaled by driving it low. This forces the processor into an S5 (idle) state on the next rising edge of the CPU clock, regardless of all other inputs. The processor remains in the S5 state until $\overline{\text{RESET}}$ again goes high, at which time the self-test sequence begins.

The self-test functions are as follows:

1. Reads and writes all registers in the register file.
2. Verifies ALU functions.
3. Checks multiplication hardware by performing a multiply.
4. Checks division hardware by performing a divide.
5. Checks ALU shifter in both directions by multiply and divide.
6. Verifies ROM constants can be addressed and used.
7. Verifies that the IC and MAR can be addressed and incremented.

As *Figure 4.1* shows, the FT initially contains all zeros. If the self-test completes successfully, the processor sets the NML PWRUP discrete output on pin 6. Otherwise, FT bit 13 is set any time the self-test detects a failure. Note that any external fault occurring prior to assertion of normal power-up causes a fault to be pending on the completion of self-test. From normal power-up to the end of initialization, the number of clock cycles can be determined from the figure. This includes additional clock cycles spent initializing the F9451 MMU and/or the F9452 BPU, providing they are part of the system.

Any faults that occur during self-test, such as $\overline{\text{EXT ADR ER}}$, are indicated by the appropriate FT bits on completion of the self-test and cause a machine error interrupt.

On successful completion of the self-test, the initialization procedure reads the system configuration from I/O address 8410 into the SCR to ascertain the presence or absence of a memory management unit, block protect unit, console, and coprocessor. It also establishes the interrupt mode and initializes the system to the values given in *Table 4.1*.

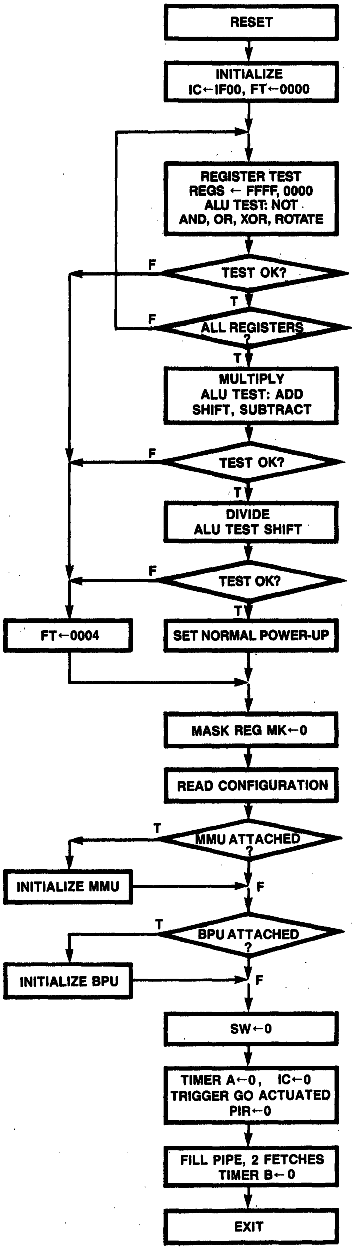
Table 4.1 System Initialization Values

CPU	
Instruction Counter (IC)	All zeros
Status Word (SW)	All zeros
Fault register (FT)	All zeros
Pending Interrupt Register (PIR)	All zeros
Interrupt Mask (MK)	All zeros
Interrupts	Disabled
DMA Enable	Disabled
Timers A and B	All zeros, counting up
Trigger Go Reset ($\overline{\text{TRIGO RST}}$)	Pulsed
Normal Power Up (NML PWRUP)	High
MMU	
Page Registers	All zeros
AL Field	All zeros
W Field	All zeros
E Field	All zeros
PPA Field	Logical to physical
BPU	
CPU Write Protect Registers	All zeros
DMA Write Protect Registers	All zeros
Global Memory Protect	Enabled

Fault and Error Handling

The extensive fault handling that real-time applications demand involves the FT, interrupt priority processing, and the abort scheme.

Initialization and Exception Processing



I/O ADDRESS	DATA	REMARKS	CLOCK CYCLES	
1F00	0000	IOW	12	
1F01	FFFF			
02	0000			
03	0000			
04	FFFF			
05	0000			
06	0000			
07	FFFF	IOW	294	
08	0000			
09	0000			
•	•			
1F30	000			
1F31	0000	IOW	40	
1F32	0000	IOW		
1F33	000F	IOW	63	
1F34	0000	IOW		
			3	
			16	
8410		IOR		
5200	0000		5	
5100	0000			
5201	0001			
5101	0001			
5202	0002			
5102	0002	IOW	3334	IF MMU PRESENT
•	•			
52FF	00FF		5	
51FF	00FF			
5000-507F	0000	IOW	1158	IF BPU PRESENT
			20	

Figure 4.1 Self-Test and Initialization Sequence

The heart of F9450 fault handling is the FT register. The FT records unrecoverable, major, and unclassified errors from violations internal and external to the CPU. The meaning of each FT bit is shown in *Table 4.2*. Whenever a bit is set in the FT, PIR bit 1 is also set, generating a machine error interrupt that cannot be disabled.

Four levels of severity exist for faults and errors:

- Unrecoverable Errors—Errors with fatal implications for program execution. This class is recorded in the FT and generates a machine error interrupt (PIR level 1) that cannot be disabled. The five unrecoverable errors are:
  - Illegal instruction
  - Memory Protect error on an instruction fetch
  - Parity error on an instruction fetch
  - External address error on an instruction fetch
  - Address state fault
- Major Errors—Errors without fatal implications for program execution, but which are nonetheless intolerable. This class is recorded in the FT and results in a machine error interrupt (PIR level 1) after the instruction has completed. The three types of major errors are:
  - Privileged instruction violation
  - Memory protect or parity error on a CPU-initiated memory data bus cycle
  - External address error on a CPU-initiated data bus (memory or I/O) read cycle
- Unclassified Errors—These errors, which are in the nature of warnings, are recorded in the FT and generate a machine error interrupt (PIR level 1).
- Arithmetic Errors—Errors resulting from arithmetic exceptions (overflow and underflow) are not recorded in the FT, but rather in the PIR to generate low-priority interrupts.

There are three methods for clearing the FT register after intercepting a fault:

- The XIO instruction CLIR (CLear Interrupt Request), which reinitializes both the FT and the PIR.
- The XIO instruction RCFR (Read and Clear Fault Register). This instruction copies the content of the FT to a designated general register, clears the FT, and resets PIR bit 1.
- The console command EXAMINE AND CLEAR FAULT REGISTER.

Table 4.2 Fault Register Bit Assignments

Bit	Fault	M	U	W
0	CPU Memory Protect Error			
	Data Cycle	X		
	Instruction Fetch		X	
1	Non-CPU Memory Protect Error			X
2	Parity Error			
	Data Cycle			X
	Instruction Fetch		X	
3	Spare (always 0)			
4	Spare (always 0)			
5	Illegal I/O Address	X		
6	Spare (always 0)			
7	System Fault 0			X
8	Illegal Memory Address			
	Data Cycle	X		
	Instruction Fetch		X	
9	Illegal Instruction		X	
10	Privileged Instruction	X		
11	Address State ErrorT		X	
12	Spare (always 0)			
13	BITE (Built-In Test) or System Fault 1 Unclassified			X
14	Spare (always 0)			
15	System Fault 1T			X

M = Major error
U = Unrecoverable error
W = Unclassified error (warning)

Instruction Abort

An instruction aborts because of any of three major errors:

- Memory protect error ($\overline{\text{MEM PRT ER}}$)
- Parity error ($\overline{\text{PAR ER}}$)

- External address error (EXT ADR ER) on a CPU-initiated data bus (memory or I/O) read cycle.

An aborted instruction completes execution but inhibits modification of internal registers. Erroneous information may, however, be written into memory or I/O locations during completion of an aborted instruction cycle. Outputs UNRCV ER (pin 8) and MAJ ER (pin 31) may be OR-wired together to form the ABORT input for the F9451 MMU (pin 13) and/or the F9452 BPU (pin 11), if these peripherals are part of the system. This external ABORT signal may be connected to the Power Down Interrupt (PWRDN INT) input (pin 33) to ensure proper handling of major and unrecoverable error conditions, even if PIR bit 1 is masked.

Three conditions cause an effective NOP; i.e. the current instruction is not executed and the next instruction is fetched:

- Illegal instruction.
- Address state fault.
- Privileged instruction violation

These conditions set the appropriate bits in the FT and initiate a machine error interrupt.

Under certain circumstances, an instruction cannot be fetched:

- Memory protect error.
- Parity error.
- External address error.

All three conditions cause a machine error interrupt. Because of the F9450 pipeline architecture, faults that occur during an instruction fetch result in an internal machine error interrupt *before* the instruction can be executed, providing PIR bit 1 is not masked. The user should make provisions to handle these faults in the interrupt service routine.

Interrupts

The F9450 has sixteen levels of on-chip prioritized interrupts, of which:

- Nine are external (PIR0, PIR2, PIR8, PIR10-PIR15) with:
  - Two level-sensitive (I/O Levels 1 and 2).
  - Seven level- OR edge-sensitive (User 1 – User 5 and Power Down). Level or edge is set via the interrupt mode bit in the SCR.
- Seven are internal (PIR1, PIR3-PIR7, PIR9).

An interrupt signals the F9450 for service by latching its associated bit in the PIR. The highest-level interrupt corresponds to the most significant bit of the PIR (bit 0), with priorities descending to bit 15. *Table 4.3* maps interrupts to PIR bit positions.

The programmer can elect to ignore one or more types of interrupts by loading a mask into the MK, subject to the limitations noted in the table. Each 0-bit in the MK masks the PIR interrupt associated with that bit position. The F9450 interrupt servicing routine automatically saves the MK register before activating the interrupt handler, as described below.

Except for I/O Levels 1 and 2, all external interrupts have hysteresis circuitry for noise immunity.

Table 4.3 Pending Interrupt and Mask Registers

Interrupt	PIR/MK bit	Source
Power Down[1]	0	External
Machine Error[2]	1	Internal
User 0	2	External
Floating-Point Overflow	3	Internal
Fixed-Point Overflow	4	Internal
Executive Call[1]	5	Internal
Floating-Point Underflow	6	Internal
Timer A	7	Internal
User 1	8	External
Timer B	9	Internal
User 2	10	External
User 3	11	External
I/O Level 1	12	External[3]
User 4	13	External[3]
I/O Level 2	14	External
User 5	15	External

Note

1. Cannot be masked via the MK register or disabled with the DSBL instruction.
2. Cannot be disabled with DSBL.
3. Level-sensitive only.

When an interrupt occurs during execution of an instruction, that instruction completes. If two or more interrupts occur during an instruction cycle, the one with the highest priority (and that is not masked) gains control upon completion of the current instruction and invokes its handler. The F9450 proceeds as follows in response to an interrupt, and *Figure 4.2* depicts this sequence:

1. Further interrupts are disabled.
2. The AS bits in the SW register are ignored, coercing the processor to memory page 0.
3. The F9450 internally stores the current MK, SW, and IC (the 'old processor status').
4. The processor fetches Service Pointer 'n' (where 'n' corresponds to the PIR bit number). From the three-word area pointed to by the service pointer, the F9450 retrieves and loads the new MK, SW, and IC.
5. The processor fetches Linkage Pointer 'n' from the interrupt vector. It stores the old MK, SW, and IC contents starting at the address pointed to by the linkage pointer. This address is within the address space given by the AS bits of the new SW.
6. Execution then commences at the point indicated by the IC within the new page segment. This is the entry into a handler appropriate for the interrupt being serviced.

The F9450 interrupt vector must begin at address 20_{16} and extend through $3F_{16}$ within page segment 0. The linkage and service pointers indicate three-word areas. The service pointers must point to areas within page 0 containing images of the MK, SW, and IC registers. The linkage pointers indicate uninitialized areas within the target page segments. It is the programmer's responsibility to:

- Initialize the interrupt vector pointers.
- Initialize the areas pointed to by the service pointers so that the registers are properly set up to enter the appropriate handler.

NOTE: The area indicated by Service Pointer 5 (Executive Call) differs from other areas, and is discussed below.

The F9450 acknowledges an interrupt by resetting the acknowledged bit in the PIR and executing an I/O cycle, during which it sends the complement of the interrupt bit pattern to I/O device address 1000_{16}. For example, to acknowledge interrupt level 11, the CPU writes to I/O address 1000_{16} data equal to FFEF (bit 11 = 0, all other bits = 1). Interrupt requests must be removed within two machine cycles after their acknowledge cycle (IOW cycle). If an additional delay in removing the interrupt request is required, the delay is extended by inserting wait states.

After invoking a handler, the disabling of interrupts remains in effect. It is the programmer's responsibility to re-enable interrupts via the ENBL instruction. This allows the processor to act on any interrupts not masked out by the MK register.

The PIR can be loaded via the privileged XIO instruction SPI to generate a simulated interrupt. If the bit set by SPI is not masked via the MK register and interrupts are enabled, the indicated interrupt occurs at the end of the next instruction cycle.

Executive Call

The BEX (Branch to Executive) instruction provides a means to make controlled, protected calls to any of sixteen executive routines or program segments, which may be in other address spaces. The form of the instruction is:

BEX n

where 'n' is a number in the range 0–15 specifying which of the possible sixteen entry points is desired. This instruction sets bit 5 in the PIR (which cannot be masked or disabled), thereby generating a special interrupt referred to as an Executive Call.

What makes BEX special is its service vector. When BEX executes, the F9450 saves the old MK, SW, and IC in the same manner as for any other interrupt; i.e. in the three-word area within the new address space pointed to by Linkage Pointer 5. The service pointer, however, points to an 18-word programmer-initialized area within page 0, structured as follows:

New MK
New SW
New IC0
New IC1
. . .
New IC15

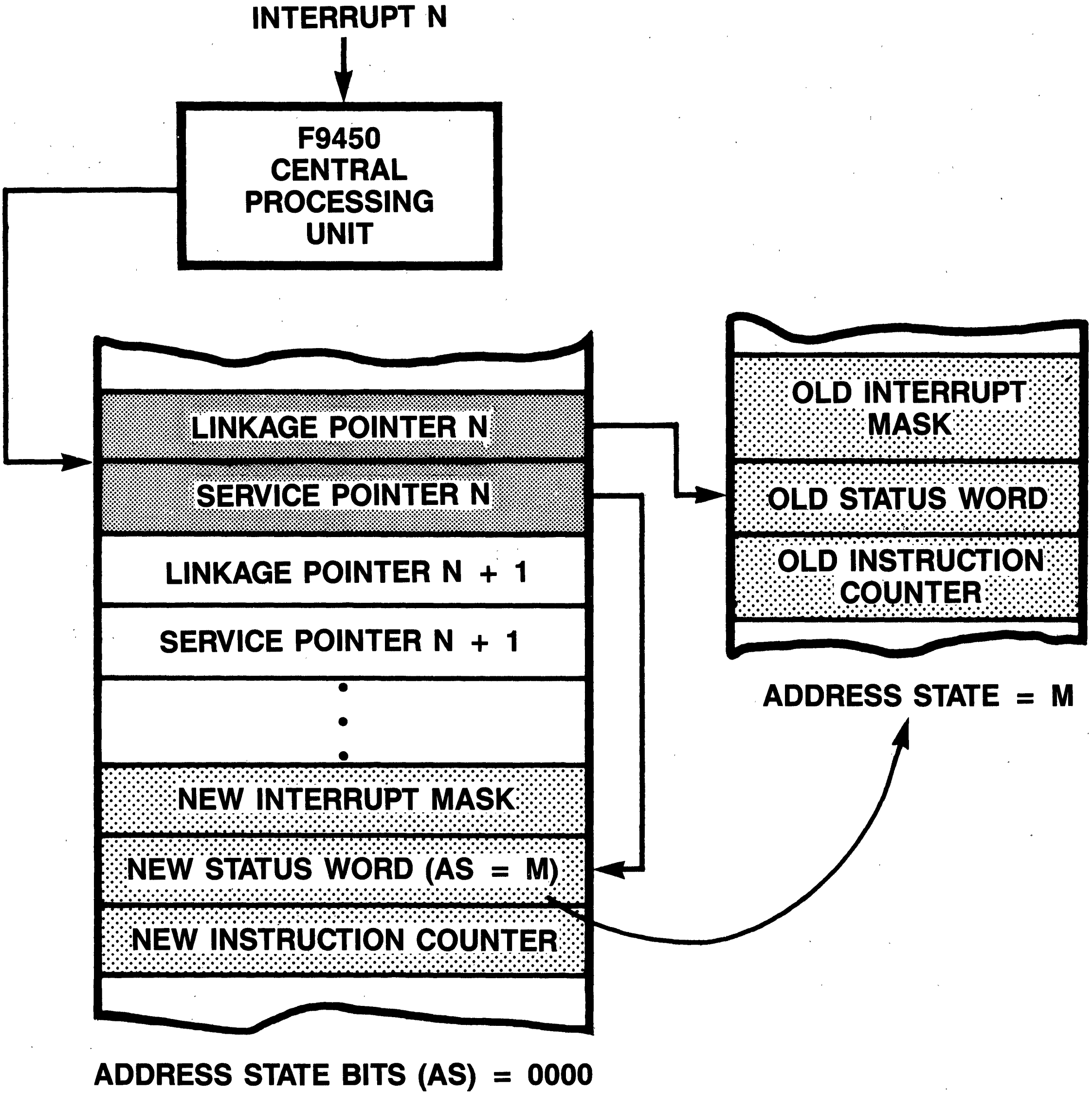


Figure 4.2 Interrupt Linkage

In processing an Executive Call, the F9450 saves the old processor status in the new address space and loads the new MK and SW as described above. It then goes to the nth IC entry (where n is the constant specified in the BEX instruction) to obtain the new Instruction Counter. This value is the entry point of the desired code segment within the address space defined by the AS bits of the new SW. The next instruction to be executed is the first in the new routine.

The LST (Load STatus) instruction furnishes the means to return from a routine called with BEX. It can also be used to make an unconditional jump within an address space or to another address space. This privileged instruction has two forms:

LST ADDR

LST ADDR, RX

(Note: the examples given here are for direct addressing mode; there is also an LSTI for indirect addressing.) The first form uses the given address. The second derives an address by adding ADDR to the contents of an index register.

In either case, LST goes to the three-word derived address and loads the MK, SW, and IC registers from three sequential memory words. The next instruction to execute will be the one indicated by the new IC, within the address space given by the AS bits of the newly-loaded SW.

Note that the address for LST must be consistent. If BEX was used to call the routine, Linkage Pointer 5 (in the interrupt vector in page 0) must point to the same address that LST refers to in retrieving the old processor status, assuming that the programmer wishes to resume at the instruction following the invoking BEX. If not—that is, the code segment being exited is not a subroutine, and control is now moving on to another code segment—LST should load from a different three-word area containing the appropriate images for the new processor status.

Context Switches

A context switch occurs when one interrupt vector structure replaces another, thereby dramatically altering the context in which the system operates by changing the way it responds to interrupts and executive calls. This is accomplished by a simple MOV that overlays the present interrupt vector with the new one. There are a few factors that the programmer must consider when planning a context switch.

The first is the format of the interrupt vector. The only ironclad rules are that the sixteen pairs of linkage and service pointers must begin at address 20_{16} in page 0, and must point to three-word areas (except for the Executive Call vector) within the same memory page. In order to keep the context switch as uncomplicated as possible, it makes sense to organize the elements of the interrupt table in a contiguous 146-word memory area as shown in *Table 4.4.*, so that the entire image of an interrupt structure can be moved with a single MOV instruction.

Table 4.4 Recommended Interrupt Vector Map

Address (hex)	Content
20–3F	Sixteen prioritized pairs of linkage and service pointers (mandatory location)
40–6F	Sixteen linkage vectors: Three-word uninitialized save areas for old MK, SW, IC.
70–9F	Sixteen service vectors: Three-word initialized areas containing new MK, SW, IC.
A0–B1	Executive Call: new MK and SW, sixteen new IC values.

The second consideration is that a disruptive interrupt can occur during the context switch. MOV is the only F9450 instruction that suspends operation upon interrupt; all other instructions complete before the processor acts on the interrupt. This is a potentially disastrous situation, since an interrupt could occur while its service vector is in transition. The result would be a branch Off Yonder Somewhere, with a loss of program integrity.

To protect against this contingency, disable interrupts just before altering the interrupt vector, and re-enable them immediately afterward. Because DSBL and ENBL are privileged instructions, this suggests that an executive routine might be the best approach to handling context switches, invoked either through BEX or via a simulated interrupt using one of the USER vectors.

The third consideration is self-evident, but easy to overlook; there must be a way to recover the previous processor context when a switch-back is appropriate. After disabling interrupts and before activating the new vector, the present vector should be transferred to a 146-byte save area allocated in page 0 using the MOV instruction. It can later be fetched from the save area and restored.

For the sake of consistency, it is advisable to have different executive routines to activate Vector 1 and Vector 2, but accessible through the same call in either vector. For example, Vector 1 might use BEX 12 to activate Vector 2, while Vector 2 uses BEX 12 to activate Vector 1. Where three or more contexts exist, a more complex approach will need to be developed.

In summary, the following is a suggested method for performing a context switch:

1. Set aside 146-word memory areas containing initialized images of each contiguous interrupt vector. These areas can later be used to save their assigned vectors when being switched out.
2. Disable interrupts.
3. Copy the current vector to its save area.
4. Copy the new vector from its save area into the reserved location starting at 20_{16}.
5. Re-enable interrupts.
6. Branch to the entry point of the new context.

Privileged Instructions

The 1750A ISA identifies certain instructions as privileged; i.e., as having the ability to corrupt data areas and I/O devices used by other subprograms. The F9450 provides a safeguard mechanism to prevent privileged instructions from executing under inopportune conditions.

For a privileged instruction to execute, the AK(PS) bits in the SW register must all be set to zeros. An attempt to execute a privileged instruction when any 1-bits are present in the AK(PS) field causes a Machine Error (PIR 1), sets bit 10 in the FT register, and causes the instruction to abort.

There are 47 privileged instructions in the F9450 instruction set: LST, VIO, and the 45 instructions covered by XIO.

Arithmetic Exceptions

The 1750A ISA provides limited facilities for detecting arithmetic exceptions. Three types of exception signals are furnished:

- Fixed-point overflow.
- Floating-point overflow.
- Floating-point underflow.

These exceptions latch PIR bits 4, 3, and 6 respectively, thereby generating an interrupt.

A fixed-point overflow occurs when an arithmetic operation on integers produces a result greater than $7FFF_{16}$ or less than 8000_{16} in single-precision (16-bit) format, or when the result is greater than $7FFF\ FFFF_{16}$ or less than $8000\ 0000_{16}$ in double-precision (32-bit) format. In the event of a fixed-point overflow, the instruction completes and leaves the 16 or 32 least-significant bits of the result in the affected register(s). Division by zero produces a fixed-point overflow, but clears the affected register(s).

Floating-point overflow and underflow occur when the exponent exceeds the numeric capacity of the 8-bit signed exponent field with a value greater than $7F_{16}$ or less than 80_{16}. Overflow returns the largest possible absolute value to the affected registers and sets the sign to that of the result. *Underflow* sets the affected registers to zero. Floating-point division by zero produces an *overflow*.

Console Operation

The F9450 provides a feature unusual among microprocessors: the ability to interface directly to an external console. This console enables the operator to examine and change the contents of CPU registers, memory, and I/O subsystems.

The F9450 conducts all console transactions via three I/O addresses:

Address:	*Console function:*
8400_{16}	Command input
8401_{16}	Data input (from switches)
0400_{16}	Data output (to a display)

Entering Console Mode

There are two ways to enter console mode:

1. By driving the CONREQ input low. When this happens, the CPU completes the current instruction and then executes an I/O cycle to read the console command from I/O address 8400_{16}. The console command can be any of those shown in *Table 5.1*.
2. By executing the BPT (breakpoint) instruction. Upon encountering a BPT, the CPU reads the system configuration bits from I/O address 8410_{16} . If the console bit is zero, the CPU treats the BPT instruction as a NOP; otherwise, it enters console mode and waits for a command to be entered at 8400_{16}.

While in console mode, the F9450 executes each command and then goes into a loop, waiting for assertion of the CONREQ signal.

Sequence of Console Operations

This section generally describes the events depicted in *Figure 5.1*, Console Handshake Sequence.

1. The user activates the Console Request (CONREQ) input (pin 2), which has a lower priority than interrupts. It is sampled on entry to the last micro-cycle of each instruction.
2. The F9450, on completing the instruction and recognizing the console request, enters console mode.
3. Timers A and B are automatically halted. They resume counting from the same point when the CPU exits from console mode.
4. The CPU reads I/O address 8400_{16}. At this time the user should deactivate the CONREQ signal and place the console command on the bus.
5. The CPU, on decoding the command, proceeds through the requisite steps to execute it. This usually includes additional I/O cycles that either read data entered from the console at I/O address 8401_{16}, or that write output data to the console via I/O address 0400_{16}.
6. After executing a console command, the CPU enters an idling loop and waits for CONREQ to be reasserted.

Notes on Console Use

MIL-STD-1750A describes the console as optional and defines data transfer to the console as byte operations. Since the F9450 uses full 16-bit word transfers, Fairchild selected the console I/O addresses from the available spare addresses defined by the Military Standard.

During console commands that generate a bus cycle, the state of the D/I line is 1 (Data cycle). An illegal console command is treated as a NOP instruction and the CPU waits for another console request.

Since one of the main purposes of the console is to permit programmers to observe their software in action and debug it, single-stepping is an important aspect of console operation. In order to perform the single step operation, the user should provide the CONTINUE command, and then activate the CONREQ signal immediately on recognizing the second fetch from instruction space in the system memory. The user *cannot* single step the processor by maintaining an active CONREQ signal and repeatedly responding to the I/O read at 8400_{16} with the CONTINUE command. If this is done, the CPU recognizes CONREQ before executing the instruction and never exits from Console Mode.

If an external fault (EXT ADR ER, PRT ER, or PAR ER) occurs during console operation, the register file is protected in the same manner as if an instruction were in process when the fault occurred. Because alteration of registers is inhibited following an external fault, the user cannot modify any of the sixteen general purpose registers or the six temporary registers until the processor is taken out of console mode as described in the next section.

Table 5.1 Console Command Formats

0 — 7	8	9[1]	10 — 15
CONSOLE CODE	X	X	REGISTER ADDRESS

Console Code and Command (Hex)	
74	: DISABLE (1)[2, 3]
60	: EXAMINE REGISTER (1)
61	: DEPOSIT REGISTER (2)
62	: EXAMINE AND CLEAR FAULT REGISTER (FT) (1)[3]
63	: DEPOSIT STATUS WORD (SW) (2)[3]
66	: EXAMINE MEMORY (2)[3]
67	: DEPOSIT MEMORY (2)[3]
6A	: EXAMINE NEXT MEMORY (1)[3]
6B	: DEPOSIT NEXT MEMORY (2)[3]
75	: CONTINUE (1)[3]
6C	: EXAMINE XIO (2)[3]
6D	: DEPOSIT XIO (2)[3]
6E	: EXAMINE NEXT XIO (1)[3]
6F	: DEPOSIT NEXT XIO (2)[3]

Binary	Hex	Reg.
0 0 0 0 0 0	(00)	R0
0 1 0 0 0 0	(10)	R1
0 0 0 0 0 1	(01)	R2
0 1 0 0 0 1	(11)	R3
0 0 0 0 1 0	(02)	R4
0 1 0 0 1 0	(12)	R5
0 0 0 0 1 1	(03)	R6
0 1 0 0 1 1	(13)	R7
0 0 0 1 0 0	(04)	R8
0 1 0 1 0 0	(14)	R9
0 0 0 1 0 1	(05)	R10
0 1 0 1 0 1	(15)	R11
0 0 0 1 1 0	(06)	R12
0 1 0 1 1 0	(16)	R13
0 0 0 1 1 1	(07)	R14
0 1 0 1 1 1	(17)	R15
0 0 1 0 0 0	(08)	A2
0 1 1 0 0 0	(18)	A1
0 0 1 0 0 1	(09)	Q2
0 1 1 0 0 1	(19)	Q1
0 0 1 0 1 0	(0A)	DO0
0 1 1 0 1 0	(1A)	DO1
0 0 1 0 1 1	(0B)	PIR
0 1 1 0 1 1	(1B)	MK
0 0 1 1 0 0	(0C)	FT[4]
0 0 1 1 0 1	(0D)	SW[5]
0 0 1 1 1 0	(0E)	TA
0 1 1 1 1 0	(1E)	TB
1 0 0 1 0 1	(25)	IC[5]
1 0 1 1 X X	(2C–2F)	IR

Notes:

1. Bits 8 and 9 are always "Don't Care."
2. The (1) and (2) designations indicate a 1- or 2-word-long console command:
   (1) indicates command only
   (2) indicates command and data
3. For commands that do not require a register address, bits 10 through 15 are "Don't Care."
4. In executing the DEPOSIT REGISTER command (61 Hex), only bits 9, 10, 11, and 13 of the Data Word can be deposited into the FT register. These four bits are user-specified with the inverted polarity. The remaining 12 bits are Don't Care; e.g., to set bits 10, 11, 13 to a one and bit 9 to a zero, Data Word XXXX XXXX X100 X0XX must be supplied from I/O location 8401.
5. Only the EXAMINE REGISTER command applies to the Status Word (SW) and Instruction Counter (IC) registers. To deposit (load) the IC with a 16-bit value, the EXAMINE MEMORY console command is used. The address specified in this command (the second word) is automatically loaded into the IC.

The DEPOSIT MEMORY command moves input data from I/O address 8401_{16} into the memory location indicated by the IC register. When DEPOSIT NEXT executes, the F9450 increments the IC and stores the next item of user-provided data into the memory location pointed to by the new value of the IC.

The EXAMINE XIO command reads an I/O address from A1, which is an F9450 internal scratchpad register, and fetches the value found at that I/O address. This command expects the I/O address to be preloaded in register A1 by the DEPOSIT REGISTER console command.

When executing the EXAMINE NEXT XIO and DEPOSIT NEXT XIO console commands, the F9450 increments register A1 and performs the appropriate transfer with the I/O location pointed to by the new value of A1. Although the most significant bit (MSB) of the I/O address normally indicates direction (read or write), this is not true for the four console commands EXAMINE XIO, EXAMINE NEXT XIO, DEPOSIT XIO, and DEPOSIT NEXT XIO.

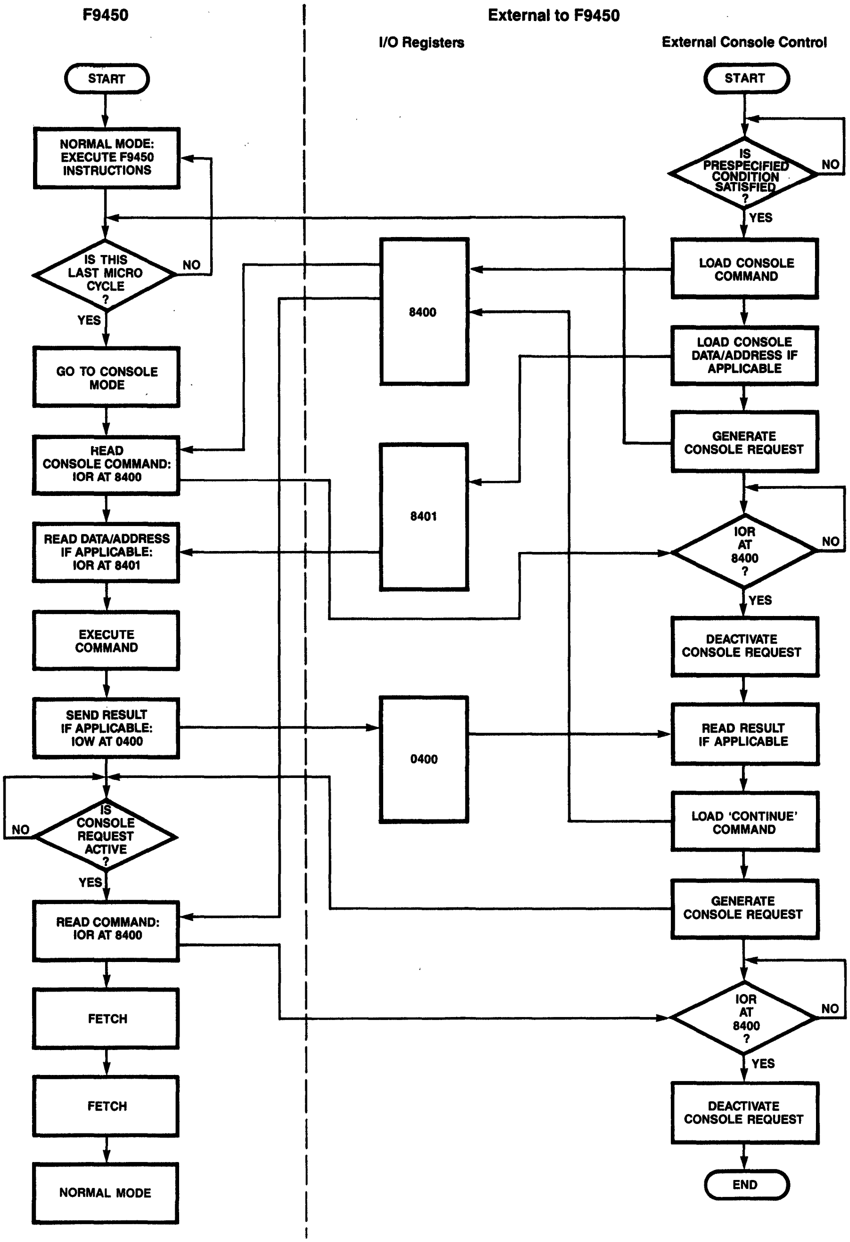
Figure 5.1 is a flow chart that maps the handshake sequence between the F9450 console port and the processor's console control logic. This external logic first sets up the I/O communications registers (8400_{16} and 8401_{16}) and then generates a $\overline{\text{CONREQ}}$ input to the CPU when a pre-specified condition is satisfied. The flow chart depicts a sequence of two console commands: a general console command followed by a CONTINUE console command.

Exiting Console Mode

There are two ways to exit from console mode:

1. If an interrupt is pending when the CPU exits a DISABLE command, the CPU services the interrupt and then returns to the normal mode of operation, fetching and executing instructions from the system memory. If an interrupt is not pending, the CPU remains in an intermediate state, alternately checking for pending interrupts or a console request.
2. By having the CPU execute a console CONTINUE command.

Figure 5.1 F9450 Console Handshake Sequence


F9450
External to F9450
I/O Registers
External Console Control
START
NORMAL MODE: EXECUTE F9450 INSTRUCTIONS
IS THIS LAST MICRO CYCLE ?
NO
YES
GO TO CONSOLE MODE
HEAD CONSOLE COMMAND: IOR AT 8400
READ DATA/ADDRESS IF APPLICABLE: IOR AT 8401
EXECUTE COMMAND
SEND RESULT IF APPLICABLE: IOW AT 0400
IS CONSOLE REQUEST ACTIVE ?
NO
YES
READ COMMAND: IOR AT 8400
FETCH
FETCH
NORMAL MODE
8400
8401
0400
START
IS PRESPECIFIED CONDITION SATISFIED ?
NO
YES
LOAD CONSOLE COMMAND
LOAD CONSOLE DATA/ADDRESS IF APPLICABLE
GENERATE CONSOLE REQUEST
IOR AT 8400 ?
NO
YES
DEACTIVATE CONSOLE REQUEST
READ RESULT IF APPLICABLE
LOAD 'CONTINUE' COMMAND
GENERATE CONSOLE REQUEST
IOR AT 8400 ?
NO
YES
DEACTIVATE CONSOLE REQUEST
END

F9450 Processor Characteristics

This chapter, describing the operating characteristics of the F9450 microprocessor, is most useful to engineers who design systems that incorporate the F9450.

Processor Internal Organization

The F9450 microprocesor architecture is organized into five functional sections. *Figure 6.1* illustrates the relationships of these components, which are:

- Data processor
- Microprogrammed control
- Address processor
- Interrupt and fault processor
- Timing unit

Data Processor

The 16-bit wide data processor section performs all data manipulations in the CPU. It is organized into nine functional blocks:

- 17-bit Arithmetic/Logic Unit (ALU)
- Shifter
- 16 general-purpose registers (R0-R15)
- Six temporary (internal) registers
- Memory Data Register (MDR)
- Two timers
- Constants ROM
- Status Register (SR)
- Pre-shifter and mask

Microprogrammed Control

The F9450 is governed by a microprogrammed control section with two levels of pipelining. The microcode resides in an on-chip ROM.

As the processor runs, it fetches the next instruction from the memory location indicated by the IC register and loads it into the CPU's internal instruction register. Microcode then decodes the instruction and feeds the mapping PLA, which generates the pointers necessary for both execution and the effective address routines that reside in the microcontrol store. The microcontrol store writes three output fields to the microregister. Two of these—Address Field and Branch Field—determine the subsequent microaddress. The third field controls the operation of all CPU components.

Address Processor

The address processor consists of an Instruction Counter (IC) and a Memory Address Register (MAR) that determine the addresses for instructions and data operands, respectively. Also included in the Address Processor is an independent incrementer that performs IC and MAR updates in parallel with data processor operation.

Interrupt and Fault Processor

All faults and interrupts, whether generated internally or externally, are handled by the Interrupt and Fault Processor as described in detail in Chapter Four. This CPU section consists of the Pending Interrupt Register (PIR), Mask Register (MK), Fault Register (FT), interrupt enabling logic, and a priority encoder, as well as abort condition detection and activation logic.

Timing Unit

The Timing Unit generates the internal and external strobes required for internal CPU operation and the bus transactions. A basic machine cycle can comprise three, four, or five CPU clock cycles (states), as follows:

- A 3-state cycle (S0, S4, S5) for pure internal ALU operations.
- A 4-state cycle (S0, S1, S2, S3) for minimum length bus cycles.
- A 5-state cycle (S0, S1, S2, S3, S3A or S0, S4, S5, S5A, S5B) applies to cycles that use the result of the current ALU operation to determine the next address in the microprogrammed control store.

Every timing cycle starts with state S0, in which the timing unit receives the control information needed to initiate a bus cycle or a short ALU cycle.

Signal Descriptions

Table 6.1 details the F9450 signal descriptions. These signals are arranged in the table by function rather than by pin number.

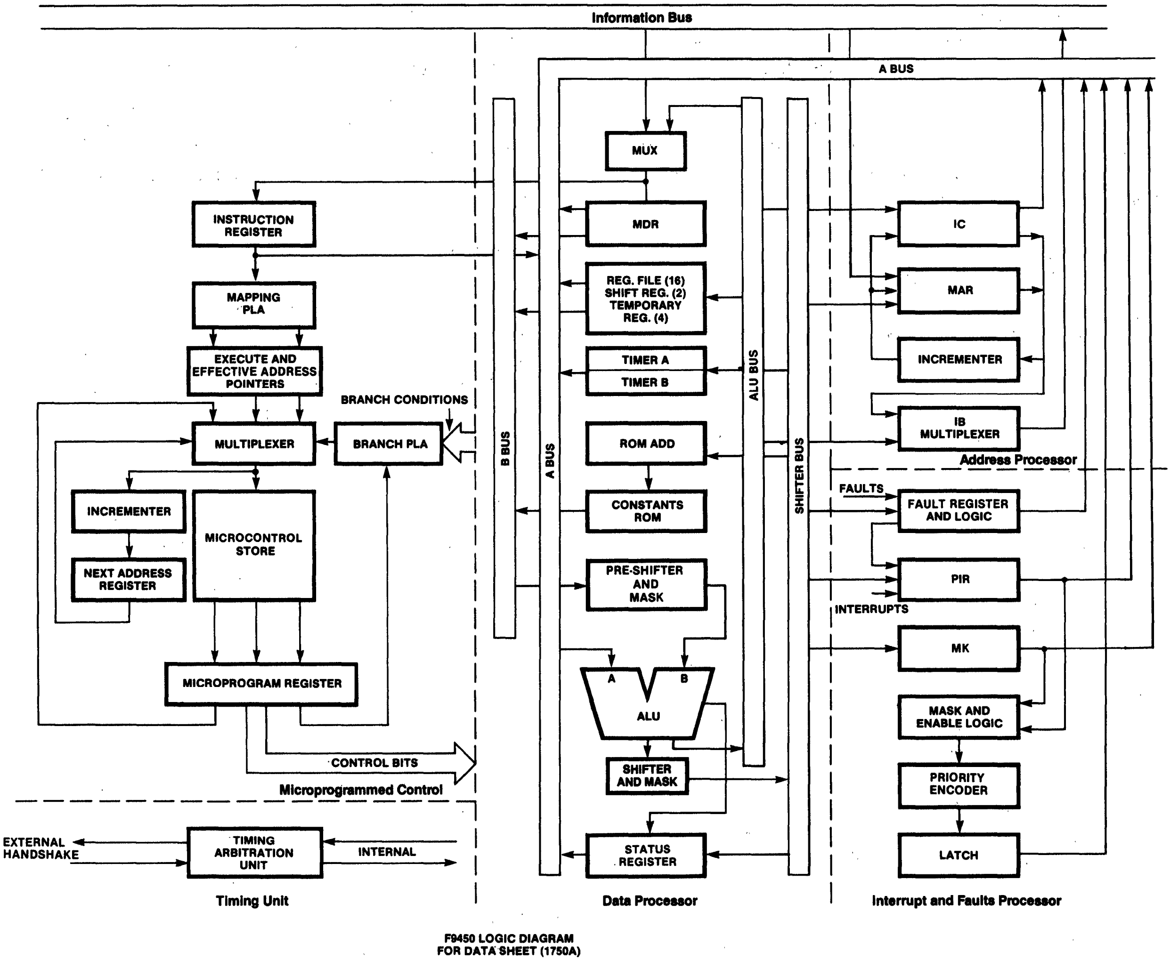


Figure 6.1 F9450 Block Diagram

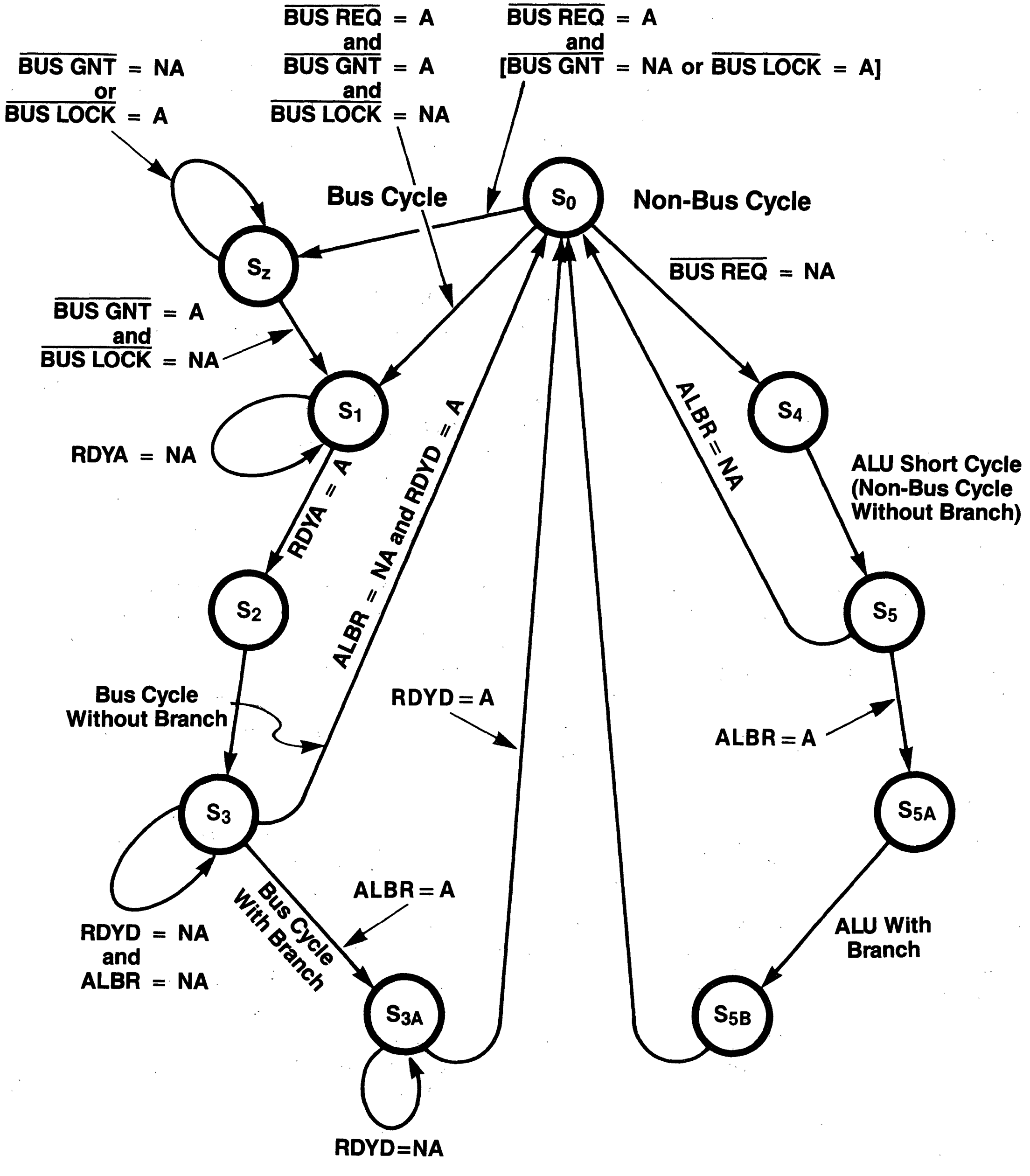

$\overline{\text{BUS GNT}}$ = NA
or
$\overline{\text{BUS LOCK}}$ = A
$\overline{\text{BUS REQ}}$ = A
and
$\overline{\text{BUS GNT}}$ = A
and
$\overline{\text{BUS LOCK}}$ = NA
$\overline{\text{BUS REQ}}$ = A
and
[$\overline{\text{BUS GNT}}$ = NA or $\overline{\text{BUS LOCK}}$ = A]
Bus Cycle
S0
Non-Bus Cycle
Sz
$\overline{\text{BUS REQ}}$ = NA
$\overline{\text{BUS GNT}}$ = A
and
$\overline{\text{BUS LOCK}}$ = NA
S1
RDYA = NA
RDYA = A
ALBR = NA and RDYD = A
ALBR = NA
S4
ALU Short Cycle
(Non-Bus Cycle
Without Branch)
S2
S5
Bus Cycle
Without Branch
RDYD = A
ALBR = A
S3
S5A
ALBR = A
Bus Cycle
With Branch
RDYD = NA
and
ALBR = NA
ALU With
Branch
S3A
S5B
RDYD=NA

Figure 6.2 F9450 Timing Generator State Diagram

Table 6.1 F9450 Signal Descriptions

Mnemonic	Pin No.	Clock	Description
CLOCKS			
CPU CLK	51	CPU Clock	Input clock signal (0–20 MHz).
TIMER CLK	17	Timer Clock	Input signal (100 kHz) that provides the clock for Timer A and Timer B.
EXTERNAL REQUESTS			
$\overline{\text{RESET}}$	5	Reset	An active low input that initializes the MPU.
$\overline{\text{CONREQ}}$	2	Console	An active low input that initiates the CPU Request console operations after the current instruction. This signal is syncronized internally with NRCK. If $\overline{\text{CONREQ}}$ is latched on entering the last machine cycle of an instruction, the F9450 enters console mode immediately after completing that instruction. Note that pending interrupts always take precedence over the console. The $\overline{\text{CONREQ}}$ should be held active until the CPU executes an I/O read at address 8400_{16} to engage console mode.
INTERRUPTS			
PWRDN INT	33	Power Down Interrupt	An input signal that is active on the positive-going edge or the high level, according to the interrupt mode bit in the configuration register. This is the highest-priority interrupt.
USR$_0$INT– USR$_5$INT	34–39	User Interrupt	Input signals that are active on the positive-going edge or high level, according to the interrupt mode bit in the configuration register.
IOL$_1$INT– IOL$_2$INT	40,41	I/O Level Interrupts	Active high inputs that can be used to expand the number of user interrupts.
FAULTS			
$\overline{\text{MEM PRT ER}}$	26	Memory Protect Error	An active low input generated by the MMU or BPU, or both, and sampled by the $\overline{\text{BUS BUSY}}$ signal into the Fault Register (bit 0 if CPU bus cycle, bit 1 if non-CPU bus cycle).
$\overline{\text{PAR ER}}$	27	Parity Error	An active low input sampled by the $\overline{\text{BUS BUSY}}$ (on ALL Bus Cycles) signal into bit 2 of the Fault Register.
$\overline{\text{EXT ADR ER}}$		External Address Error	An active-low input sampled by the $\overline{\text{BUS BUSY}}$ signal into Fault Register bit 5 or 8, depending on the cycle (memory or I/O).
SYSFLT$_0$ SYSFLT$_1$	29, 30	System Fault 0 System Fault 1	Asynchronous, positive-edge-sensitive inputs to the F9450 that set bit 7 (SYSFLT$_0$) or bits 13 and 15 (SYSFLT$_1$) in the Fault Register. These inputs are protected from system noise through hysteresis circuitry.

Table 6.1 F9450 Signal Descriptions (Continued)

Mnemonic	Pin No.	Clock	Description
INFORMATION BUS			
IB_0-IB_{15}	9–18 20–25	Information Bus	An active-high bidirectional time-multiplexed address/data bus that is three-state during bus cycles not assigned to this CPU; IB_0 is the most significant bit.
STATUS BUS			
AK_0–AK_3	47–50	Access Key	Active-high outputs used to match the Access Lock in the MMU for memory accesses (a mismatch is one of several possible situations that cause the MMU to pull the $\overline{\text{PRT ER}}$ signal low). These signals are three-state during bus cycles not assigned to this CPU.
AS_0–AS_3	42–45	Address State	Active-high outputs that select the page register group in the MMU; three-state during bus cycles not assigned to this CPU. These outputs together with D/$\overline{\text{I}}$ can be used to expand the F9450's direct addressing range to 2M words (See Chapter 3).
ERROR CONTROL			
UNRCV ER	8	Unrecoverable Error	An active-high output that indicates the occurrence of an error classified as unrecoverable. This signal goes active on the machine cycle following the error.
MAJ ER	31	Major Error	An active-high output indicating the occurence of an error classified as major. This signal goes active on the machine cycle following the error.
DISCRETE CONTROL			
DMA EN	3	Direct Memory Access Enable	An active high output indicating that DMA is enabled. It is disabled when the CPU initializes ($\overline{\text{RESET}}$) and is enabled under program control.
NML PWRUP	6	Normal Power Up	An active high output set when the CPU successfully completes the built-in test in the initialization sequence.
SNEW	63	Start New	An active high output that indicates a new instruction will execute in the next cycle; useful for tracing instructions.
$\overline{\text{TRIGO RST}}$	4	Trigger Go Reset	An active low discrete output. This signal can be pulsed low under program control. Also goes low on the completion of self-test.

Table 6.1 F9450 Signal Descriptions (Continued)

Mnemonic	Pin No.	Clock	Description
BUS CONTROL			
D/$\overline{\text{I}}$	58	Data or Instruction	An output signal indicating if the current bus cycle access is for data (high) or instruction (low); three-state during bus cycles not assigned to this CPU. This line indicates instruction space any time the F9450 is fetching an instruction from memory. This is true for all words of an instruction as defined in the MIL-STD-1750A; i.e. an immediate operand is defined as a part of the instruction, whereas an indirect address is considered to be data. This line can be used as an additional memory address bit for systems that require separate data and program memory.
R/$\overline{\text{W}}$	57	Read or Write	An output signal indicating the direction of data flow with respect to the current bus master: a high indicates a read or input operation, a low indicates a write or output operation. This signal is three-state during bus cycles not assigned to this CPU.
M/$\overline{\text{IO}}$	59	Memory or I/O	Output signal indicating whether the current bus cycle is Memory (high) or I/O output (low). This signal is three-state during bus cycles not assigned to this CPU.
STRBA	52	Address Strobe	An active high output that latches the Memory or I/O address in an external latch at the high-to-low transition of the strobe. The signal is three-state during bus cycles not assigned to this CPU.
RDYA	55	Address Ready	An active high input available to extend the address phase of a bus cycle. When RDYA is not active, wait states are inserted to accommodate slower memory or I/O devices.
$\overline{\text{STRBD}}$	53	Data Strobe	An active low output that strobes data in memory and XIO cycles. This signal is three-state during bus cycles not assigned to this CPU.
RDYD	56	Data Ready	An active high input that extends the data phase of a bus cycle. When RDYD is not active, wait states are inserted to accommodate slower memory or I/O devices.
BUS ARBITRATION			
$\overline{\text{BUS REQ}}$	54	Bus Request	An active low output indicating that the CPU requires the bus; becomes inactive when the CPU has acquired the bus and started the bus cycle.
$\overline{\text{BUS GNT}}$	61	Bus Grant	An active low input from an external arbiter indicating that the CPU currently has the highest priority bus request. If the bus is not locked, the CPU may begin a bus cycle, commencing with the next CPU pclock cycle. If the CPU locks the bus, $\overline{\text{BUS GNT}}$ is ignored.

Table 6.1 F9450 Signal Descriptions (Continued)

Mnemonic	Pin No.	Clock	Description
$\overline{\text{BUS BUSY}}$	60	Bus Busy	An active low bidirectional signal that establishes the beginning and end of a bus cycle. The trailing edge (low-to-high transition) is used for sampling bits into the Fault Register. It is three-state in bus cycles not assigned to this CPU; however, the CPU monitors this line for latching non-CPU bus-cycle faults into the Fault Register. This signal toggles on every bus cycle.
$\overline{\text{BUS LOCK}}$	62	Bus Lock	An active low, bidirectional signal that locks the bus for successive bus cycles. The CPU only locks the bus during "read-modify-write" instructions: INCM, DECM, SB (all addressing modes except R); RB (all addressing modes except R); TSB (locks bus for 5 machine cycles); SRM; STUB; and STLB. During non-locked bus cycles, the $\overline{\text{BUS LOCK}}$ signal mimics the $\overline{\text{BUS BUSY}}$ signal, and as an input it operates the same as $\overline{\text{BUS GNT}}$. It is three-state during bus cycles not assigned to this CPU.
POWER*			
V_{CC}	64	Power Supply	+2V, 1.1 Amp power supply.
GND	1,32	Ground	0V reference. These pins should be tied together as close to the chip as possible.
I_{INJ1}, I_{INJ2}	19 46	Injector Current	Current source provide bias for the injection logic. These pins should be tied together as close to the chip as possible.

Bus Transactions

The design of the F9450 permits it to operate both as a standalone processor and in a multi-processor environment where two or more CPU's—F9450 or otherwise—work in tandem with a bus arbiter resolving conflicts. The characteristics of its bus transactions reflect this design flexibility.

Bus arbitration with multiprocessors is discussed is more detail later in this chapter.

F9450 bus transactions are four or five states long, with each state being equivalent to one CPU clock period. Memory and I/O cycles have identical timing requirements and are distinguished by the status of the $M/\overline{IO}$ line, as *Table 6.1* indicates.

As *Figure 6.3* illustrates, the F9450 notifies the external arbiter that it needs the bus by activating $\overline{\text{BUS REQ}}$ during the S0 state. At the end of state S0, it samples the $\overline{\text{BUS GNT}}$ and $\overline{\text{BUS LOCK}}$ inputs. If $\overline{\text{BUS GNT}}$ is active and $\overline{\text{BUS LOCK}}$ inactive, the F9450 enters state S1. Otherwise, the CPU enters high-impedance state Sz and waits for $\overline{\text{BUS GNT}}$ to go active and $\overline{\text{BUS LOCK}}$ to go inactive. Because there is no arbiter in a single-processor system, the $\overline{\text{BUS GNT}}$ pin should be wired low and $\overline{\text{BUS LOCK}}$ permanently pulled high by a resistor to V_{cc} to simulate bus mastership for a standalone F9450.

Entering S1, the F9450 makes the $\overline{\text{BUS REQ}}$ signal inactive. This allows bus contenders to bid early for the next bus cycle. The F9450 then activates the $\overline{\text{BUS BUSY}}$, $\overline{\text{BUS LOCK}}$, and status signals, and sends the

*See Chapter Seven for detailed power specifications

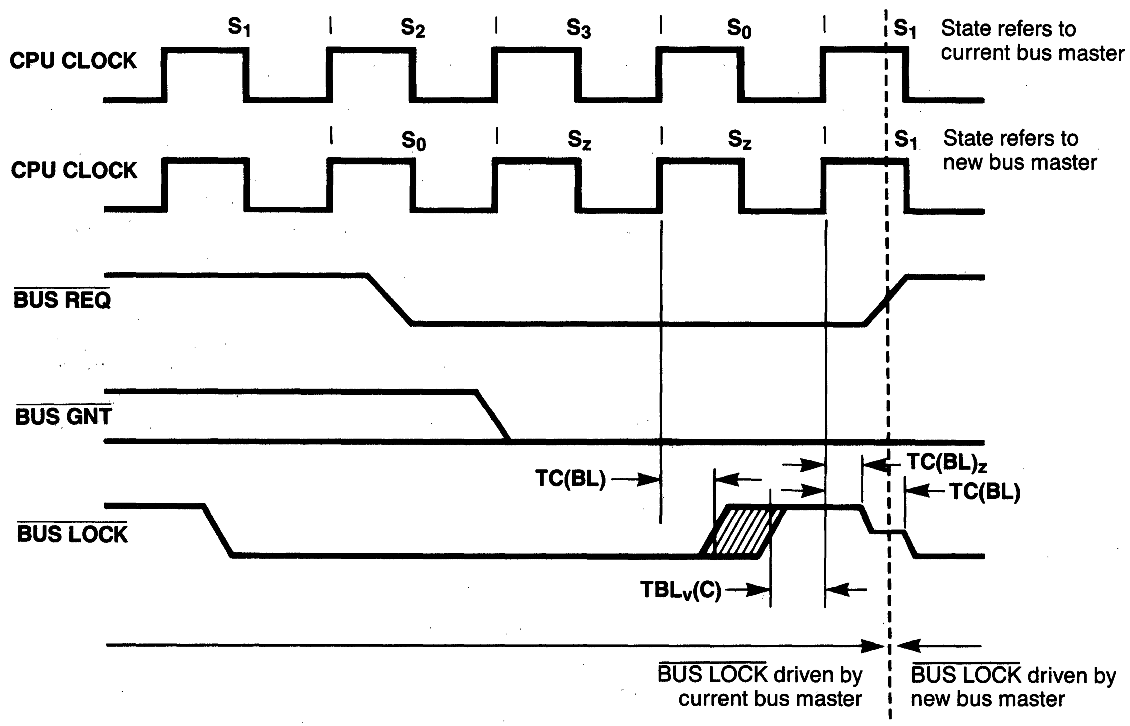

S_1
S_2
S_3
S_0
S_1
State refers to current bus master
CPU CLOCK
S_0
S_z
S_z
S_1
State refers to new bus master
CPU CLOCK
BUS REQ
BUS GNT
BUS LOCK
TC(BL)
$TC(BL)_z$
TC(BL)
$TBL_V(C)$
BUS LOCK driven by current bus master
BUS LOCK driven by new bus master

Figure 6.3 Bus Access Signal Requirements

address after a delay measured from the start of the S1 state. At the end of S1, the CPU samples the RDYA input to determine if the address is ready. If not (indicated by RDYA low), the CPU stays in the S1 state, extending the address phase on the bus. Otherwise, or when RDYA goes high, it proceeds to state S2.

Once in state S2, the CPU drives the STRBA output low. This signal can be used to latch the address in an external device. For read cycles, the F9450 also activates the $\overline{\text{STRBD}}$ output, where it prepares to receive data by turning the address/data bus around. For write cycles, the CPU starts driving the bus with the write data immediately after the address.

The $\overline{\text{STRBD}}$ signal is always activated during S3. It furnishes a reasonable time for set-up after the falling edge and a reasonable hold time to write data prior to the rising edge. The F9450 samples the RDYD signal at the end of S3 and terminates the bus cycle when RDYD is high; otherwise, it remains in the S3 state until RDYD clears.

At the end of the bus cycle, all CPU outputs are three-state. Note that the STRBA, $\overline{\text{STRBD}}$, and IB_0–IB_{15} signals for the Write cycles are actively driven through S0 of the next cycle.

All XIO/VIO output commands and internal input commands (e.g. move contents of a special-purpose register to a general register) echo back externally in the form of an I/O Write cycle. The address is the command itself, and the write data is the result of the execution phase, if applicable. The system must provide the RDYA and RDYD signals in these cycles.

Table 6.2 gives typical memory subsystem access times required by the F9450 at various operating frequencies. The access time includes address latches, address decoder delays, and system memory chip enable access.

Table 6.2 Typical Memory Subsystem Access Time Requirements

CPU Clock (MHz)	System Memory Address Access Time (ns)				
	No Wait	1 Wait	2 Waits	3 Waits	4 Waits
20	90	140	190	240	290
18	107	162	218	273	329
15	140	207	274	340	407
10	240	340	440	540	640

Device Timing Characteristics

Figures 6.4 through *6.14* furnish information concerning device timing characteristics related to F9450 I/O and control signals, and *Figure 6.15* shows typical schematics for switching time test circuits.

Throughout these figures, the abbreviated symbol convention used for timing parameters is $TAb(C)_d$,in which:

- Timing symbols all begin with the letter "T". The mnemonic in the position represented by "A" indicates the signal node beginning the interval.
- The mnemonic in the position represented by "b" defines the direction of signal transition at the beginning node, pif such definition is necessary; the new state of the signal may be low (l), high (h), 3-state (z), don't care (x), or valid (v).
- The mnemonic in the position represented by "(C)", which always appears in parentheses, indicates the signal node ending the interval.
- The mnemonic in the position represented by "d" is the same as "b", but refers to the state of the signal at the node indicated by the mnemonic in position "(C)".
- The mnemonics in the positions represented by "b" and "d" are not used for reference to the CPU CLK signal, which is assumed to be on the rising edge.

For example, $TF_1(BB)_h$ is the setup time from a valid fault ($\overline{\text{EXT ADR ER}}$, $\overline{\text{MEM PRT ER}}$, $\overline{\text{PAR ER}}$) input to $\overline{\text{BUS BUSY}}$ high.

External Coprocessors

The Built-In Function (BIF) is an escape code in the F9450 instruction set that provides for user-defined instructions to control up to four external, loosely-coupled coprocessors. MIL-STD-1750A (notice 1) permits implementors to establish their own formats for BIF; this section discusses the Fairchild format, which is supported both by the F9450 chip and the MACRO-50 assembler.

Fairchild implements BIF as a three-word sequence in which:

- The first eight bits contain the value $4F_{16}$ in compliance with MIL-STD-1750A.
- Bit 9 is always set to indicate a three-word command sequence.
- Bits 10 and 11 contain a digit 0–3 indicating the coprocessor being addressed.
- Bits 12–15 specify an index register by number (zero means not indexed). The index register contains an offset to the data address given in the third word of the sequence.
- Word 1 (bits 16–31) contains the coprocessor machine-language instruction to be executed.
- Word 2 (bits 32–47) points to the address from which the coprocessor is to fetch data.

In MACRO-50, BIF occupies three lines of source code. It has two possible formats, depending on whether the indexed or non-indexed mode is used. For non-indexed, the instruction format is:

```
BIF <n>
<coprocessor instruction>
<data address>
```

where <n> is the coprocessor selector (0–3) and the coprocessor instruction is specified as a numeric value. The form for indexed is:

```
BIFI <n>, Rx
<coprocessor instruction>
<data address>
```

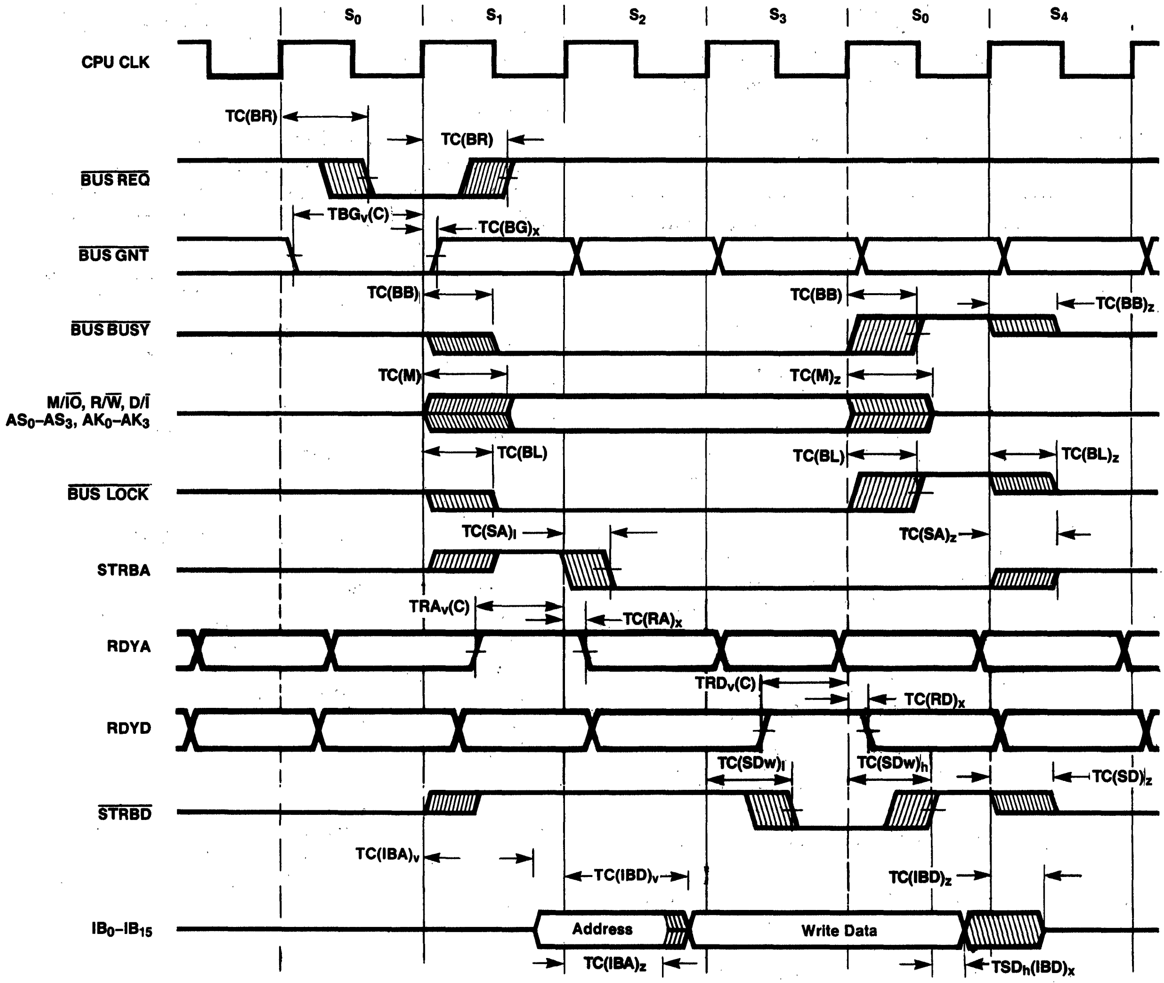


Figure 6.4 Minimum Write Bus Cycle Timing Diagram

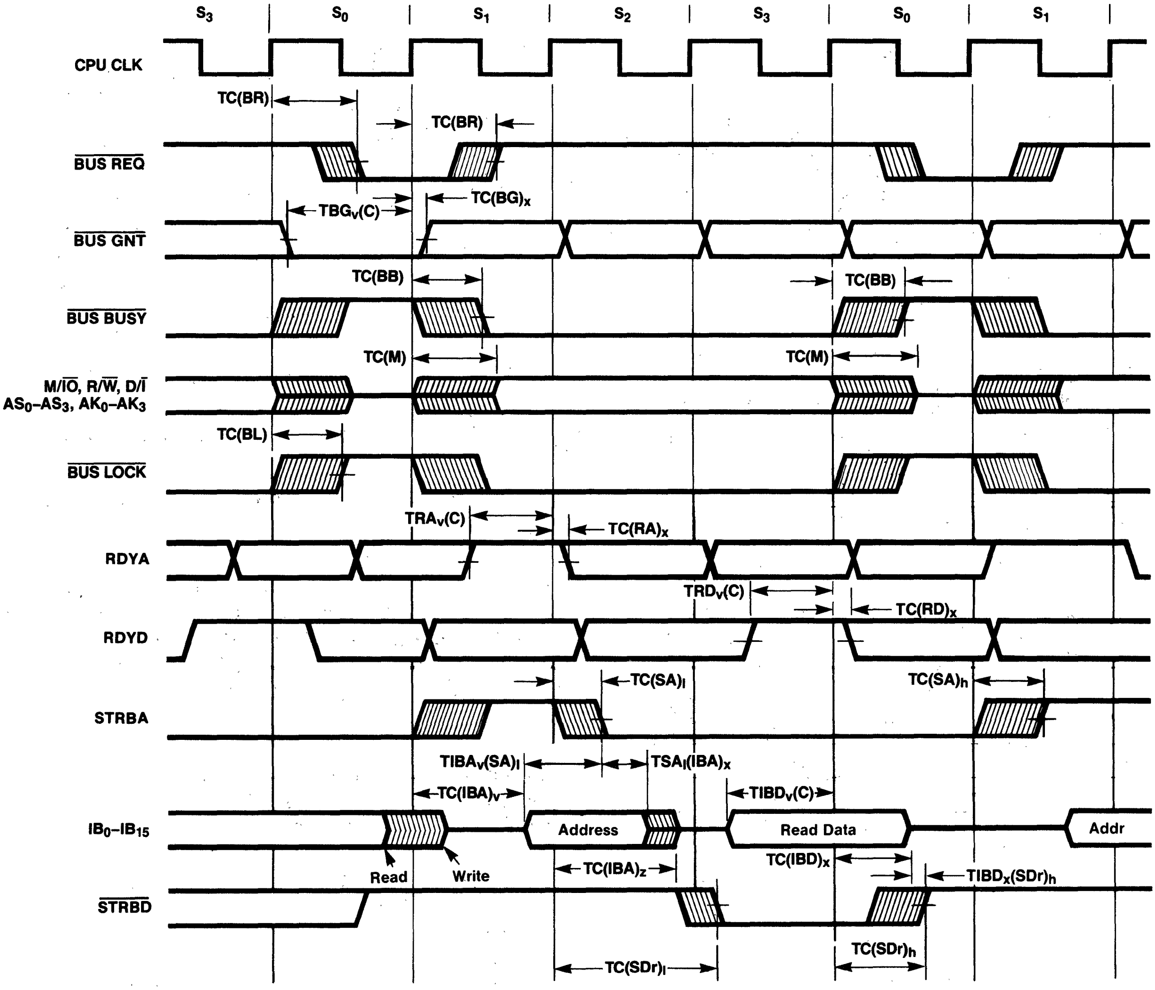

S_3
S_0
S_1
S_2
S_3
S_0
S_1
CPU CLK
TC(BR)
TC(BR)
$\overline{\text{BUS REQ}}$
$\text{TBG}_v\text{(C)}$
TC(BG)_x
$\overline{\text{BUS GNT}}$
TC(BB)
TC(BB)
$\overline{\text{BUS BUSY}}$
TC(M)
TC(M)
$\text{M}/\overline{\text{IO}}, \text{R}/\overline{\text{W}}, \text{D}/\overline{\text{I}}$
$\text{AS}_0\text{–AS}_3, \text{AK}_0\text{–AK}_3$
TC(BL)
$\overline{\text{BUS LOCK}}$
$\text{TRA}_v\text{(C)}$
TC(RA)_x
RDYA
$\text{TRD}_v\text{(C)}$
TC(RD)_x
RDYD
TC(SA)_l
TC(SA)_h
STRBA
$\text{TIBA}_v\text{(SA)}_l$
$\text{TSA}_l\text{(IBA)}_x$
TC(IBA)_v
$\text{TIBD}_v\text{(C)}$
$\text{IB}_0\text{–IB}_{15}$
Address
Read Data
Addr
Read
Write
TC(IBA)_z
TC(IBD)_x
$\text{TIBD}_x\text{(SDr)}_h$
$\overline{\text{STRBD}}$
TC(SDr)_h
TC(SDr)_l

Figure 6.5 Minimum Read Bus Cycle Timing Diagram

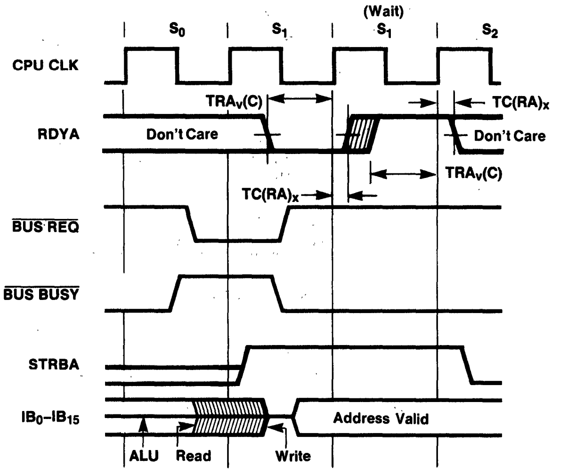


Figure 6.6 RDYA Signal Timing Diagram

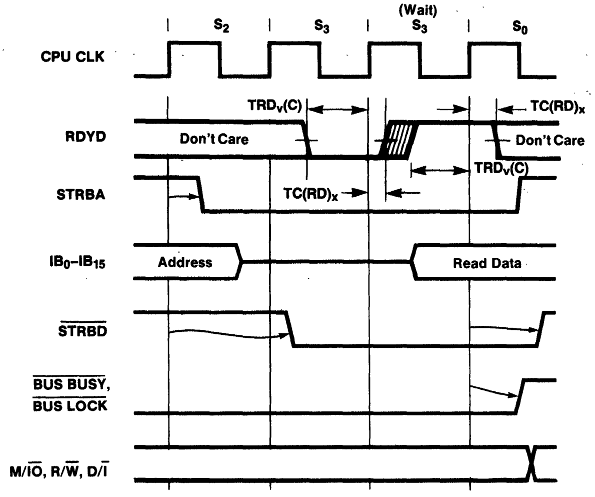


Figure 6.7 RDYD Signal—Read Bus Cycle Timing Diagram

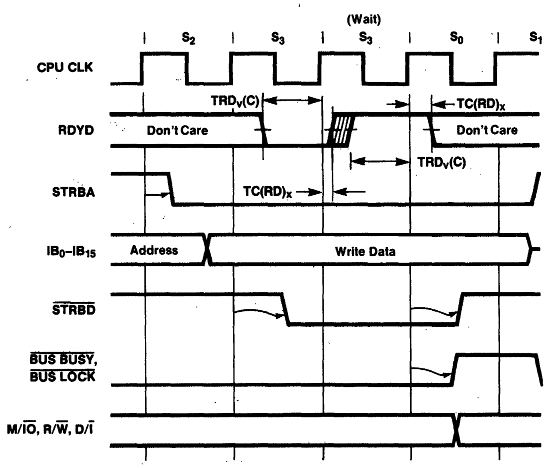
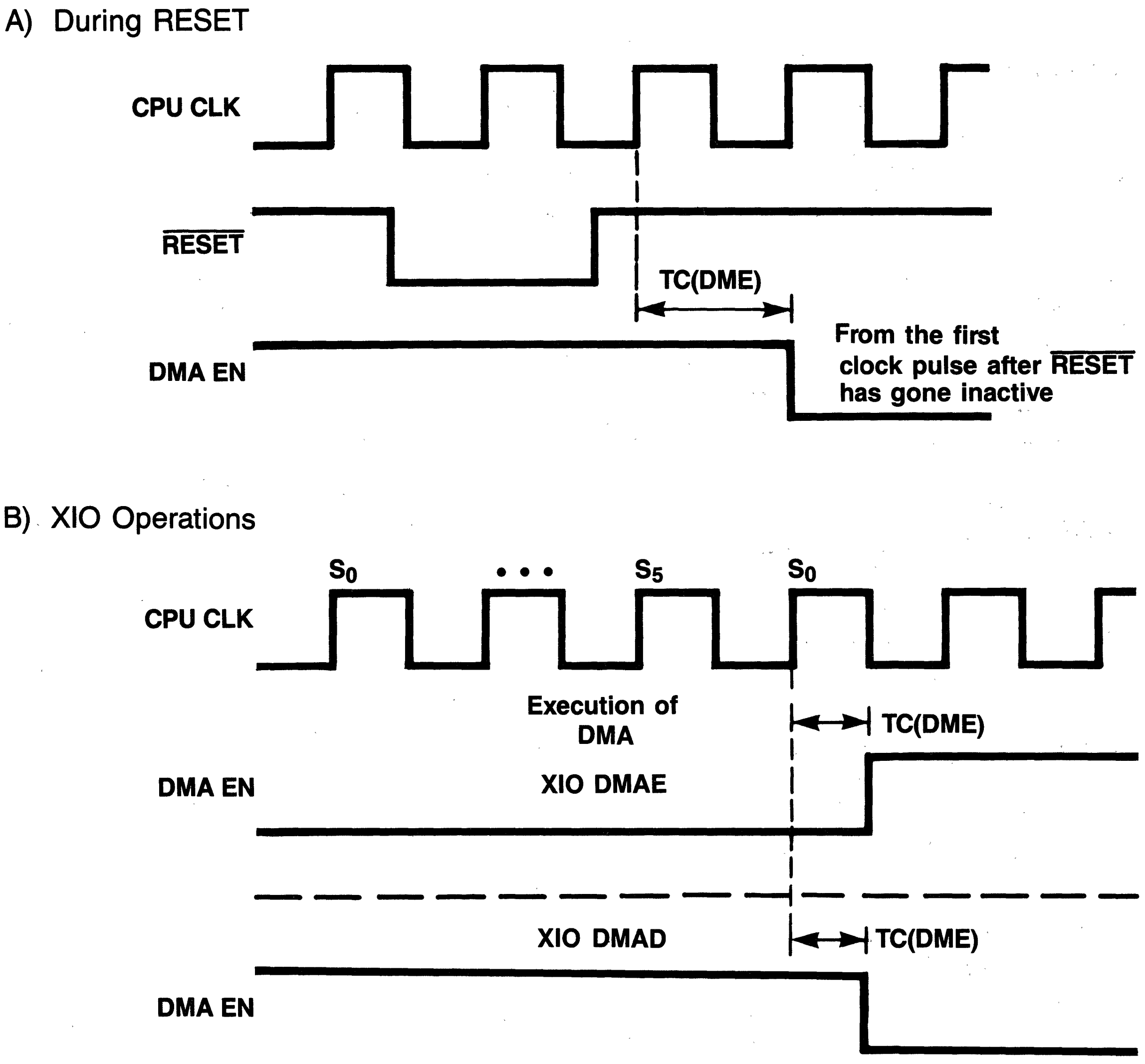


Figure 6.8 RDYD Signal—Write Bus Cycle Timing Diagram

Figure 6.16 illustrates two typical configurations incorporating a coprocessor, which is a bus contender under control of the bus arbiter. If the system includes an MMU, an additional latch is added to provide an Address State (AS) and Access Key (AK) for the coprocessor.

In executing a BIF, the F9450 sends the instruction and derived data address directly to the coprocessor via two cycles of the instruction bus.

The method by which the coprocessor returns its result to the F9450 is highly application-dependent and subject to a number of design trade-offs. Perhaps the simplest method is to have the coprocessor write its result to a fixed memory location, from which the F9450 can subsequently fetch it. This solution only works, of course, if the F9450 gives the coprocessor enough time to perform the computation and write the result. A more failsafe method is to have steering logic associated with the coprocessor initiate a user interrupt after the result has been saved; the F9450 can then run a handler that copies the result into the appropriate program workspace and sets a flag, thereby giving assurance that the value is available when needed.



A) During RESET

B) XIO Operations

Figure 6.12 DMA EN Discrete Timing Diagram

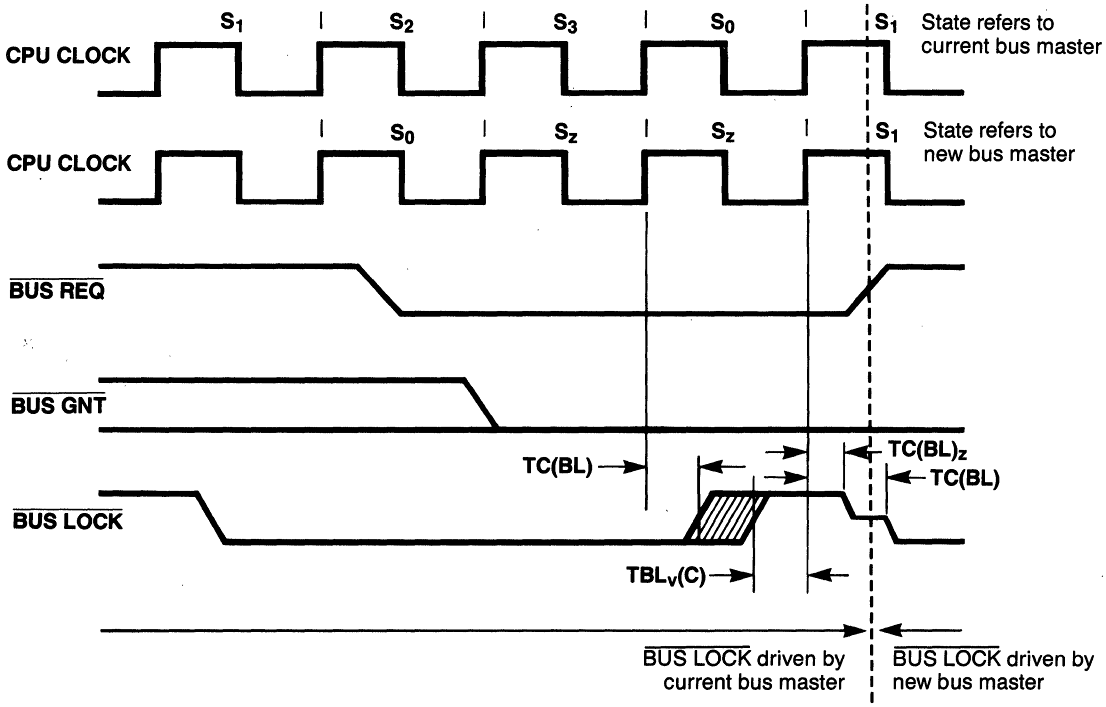

S1
S2
S3
S0
S1
CPU CLOCK
State refers to current bus master
S0
Sz
Sz
S1
CPU CLOCK
State refers to new bus master
BUS REQ
BUS GNT
TC(BL)
TC(BL)z
TC(BL)
BUS LOCK
TBLv(C)
BUS LOCK driven by current bus master
BUS LOCK driven by new bus master

Figure 6.9 Signal Requirements for Bus Access

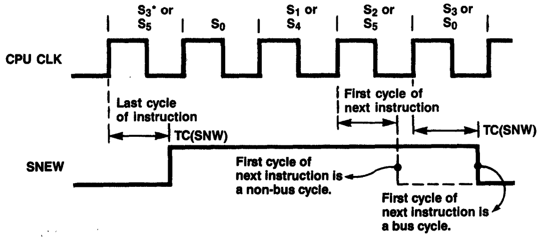

S3* or S5
S0
S1 or S4
S2 or S5
S3 or S0
CPU CLK
Last cycle of instruction
First cycle of next instruction
TC(SNW)
TC(SNW)
SNEW
First cycle of next instruction is a non-bus cycle.
First cycle of next instruction is a bus cycle.

Figure 6.10 SNEW Discrete Timing Diagram

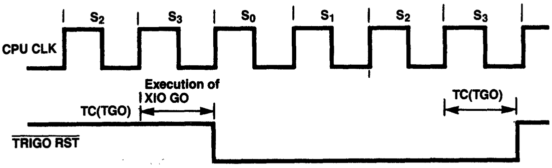


Figure 6.11 $\overline{\text{TRIGO RST}}$ Discrete Timing Diagram

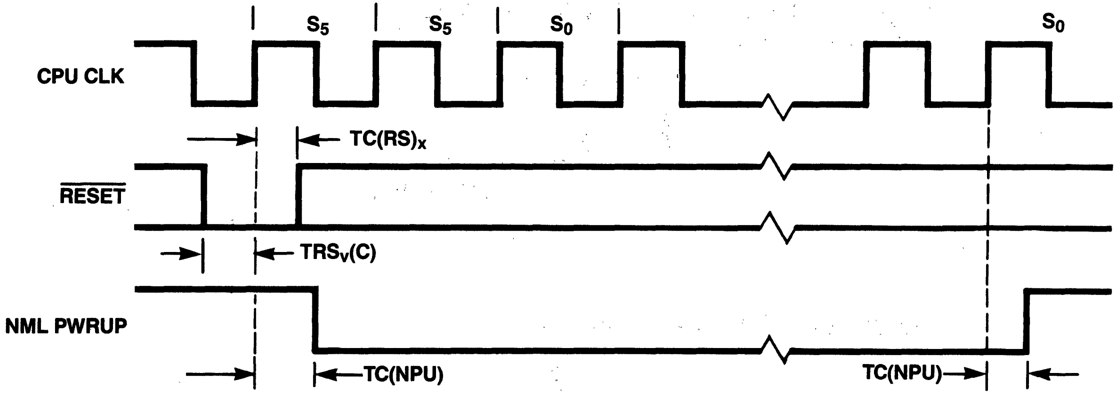


Figure 6.13 Normal Power Up Discrete Timing Diagram

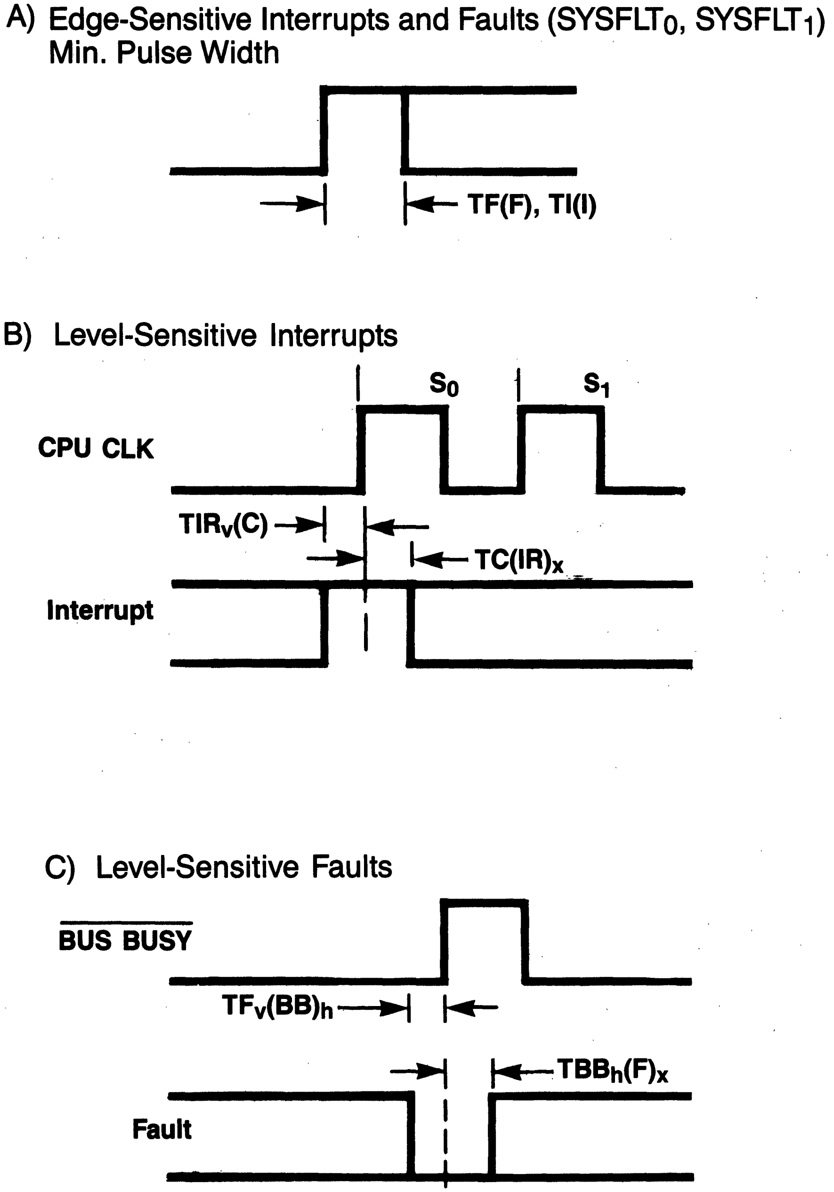
Bus Arbitration with Multiprocessors

The F9450 employs a simple bus arbitration scheme involving three bus control signals:

- BUS REQ—Bus request generated by any master requiring control of the bus.
- BUS GNT—Bus grant issued by the arbiter to the requestor with the highest priority.
- BUS LOCK—Generated by the master currently in control of the bus, to indicate that the bus is unavailable to other contenders.

This bus arbitration scheme considers three levels of priority as follows:

1. CPU/DMA-controller arbitration
   In a single CPU/DMA-controller configuration, depicted by *Figure 6.17*, the DMA device has the higher priority while enabled by the CPU. The F9450 has access to the bus only while the DMA controller does not require access.
2. Dual CPU's
   In a dual-CPU configuration, shown in *Figure 6.18*, one CPU has a higher priority than the other, and the lower-priority processor only obtains access to the bus while the higher-priority device does not need it.
3. Multiple processors with arbitrated priority
   Figure 6.19 illustrates a system with three bus masters, of which two are F9450 CPU's synchronized by a common system clock and one is an asynchronous DMA controller (this system could be expanded to include up to seven F9450's operating in tandem). Each CPU has an assigned priority. Because DMA controllers are generally asynchronous, and since the DMA REQ signal is the highest-priority bus request, the DMA controller must either keep its request active as long as it is using the bus or, after acquiring control of the bus, must assert the BUS LOCK signal *before* relinquishing the request. In this design, the DMA controller always has precedence over any CPU, as long as one of the CPU's has its DMA EN signal active.

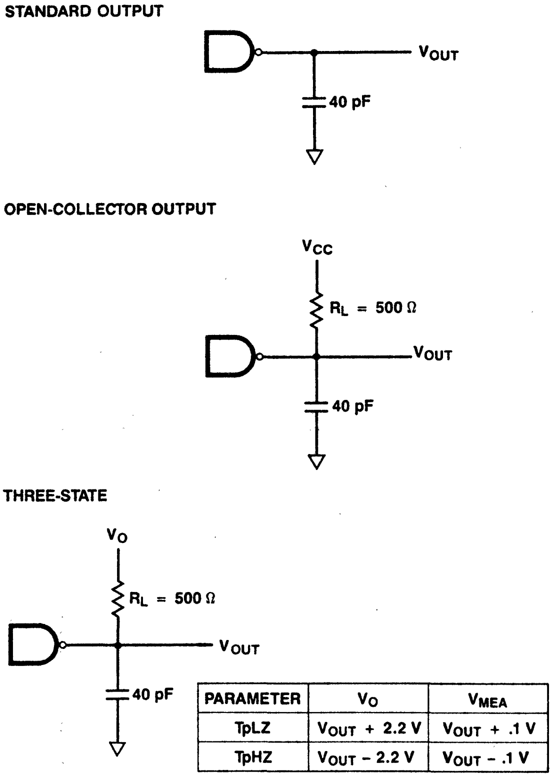


A) Edge-Sensitive Interrupts and Faults (SYSFLT$_0$, SYSFLT$_1$)
Min. Pulse Width

B) Level-Sensitive Interrupts

C) Level-Sensitive Faults

Figure 6.14 External Faults and Interrupts Timing Diagram



PARAMETER	V_O	V_{MEA}
TpLZ	V_{OUT} + 2.2 V	V_{OUT} + .1 V
TpHZ	V_{OUT} - 2.2 V	V_{OUT} - .1 V

Figure 6.15 Switching Time Test Circuits

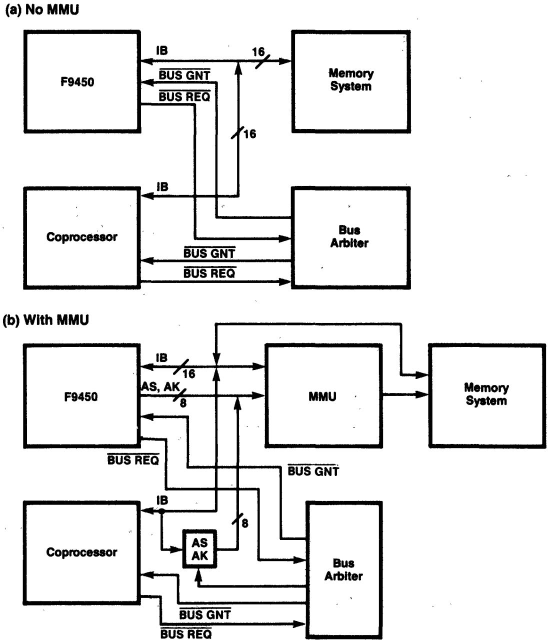


Figure 6.16 F9450/Coprocesor Configurations

Memory Expansion and Protection

The F9451 Memory Management Unit (MMU) and F9452 Block Protect Unit (BPU) facilitate system memory expansion and provide comprehensive memory protection.

The MMU does logical-to-physical address translation for a system consisting of up to 16M words of memory (1M words for MIL-STD-1750A applications). It also supplies the logic that allows execute protection in the instruction space and write protection in the data space.

The BPU provides supplementary protection capability by allowing separate write protection in 1KW blocks for both CPU and DMA access. This protection is based on a 20-bit physical address for the data space.

Chapters Eight and Nine discuss these devices.

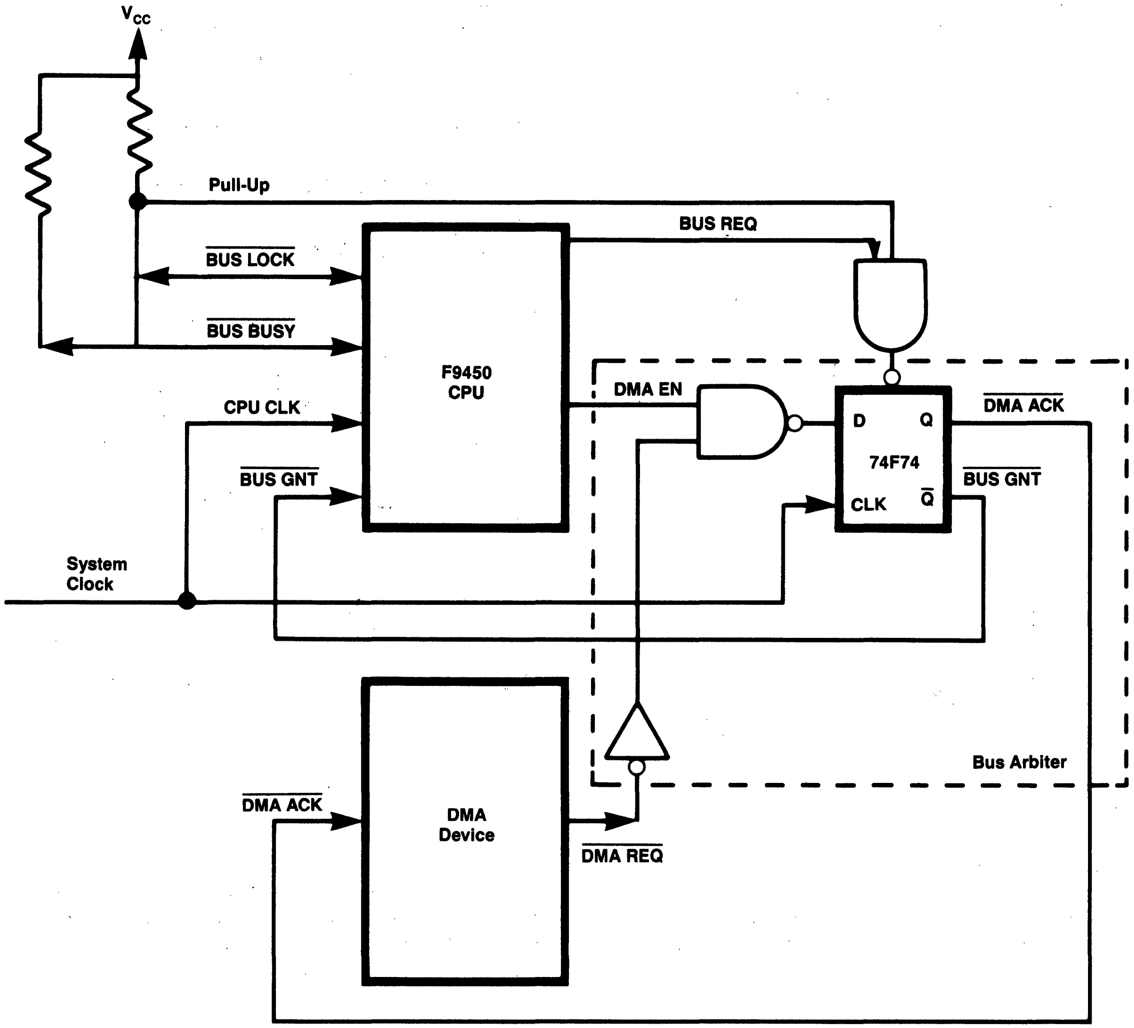

V_{CC}
Pull-Up
BUS REQ
$\overline{\text{BUS LOCK}}$
$\overline{\text{BUS BUSY}}$
F9450
CPU
CPU CLK
$\overline{\text{BUS GNT}}$
DMA EN
D
Q
74F74
CLK
$\overline{\text{Q}}$
$\overline{\text{DMA ACK}}$
$\overline{\text{BUS GNT}}$
System
Clock
Bus Arbiter
$\overline{\text{DMA ACK}}$
DMA
Device
$\overline{\text{DMA REQ}}$

Figure 6.17 Single CPU/DMA Controller System with External Arbiter

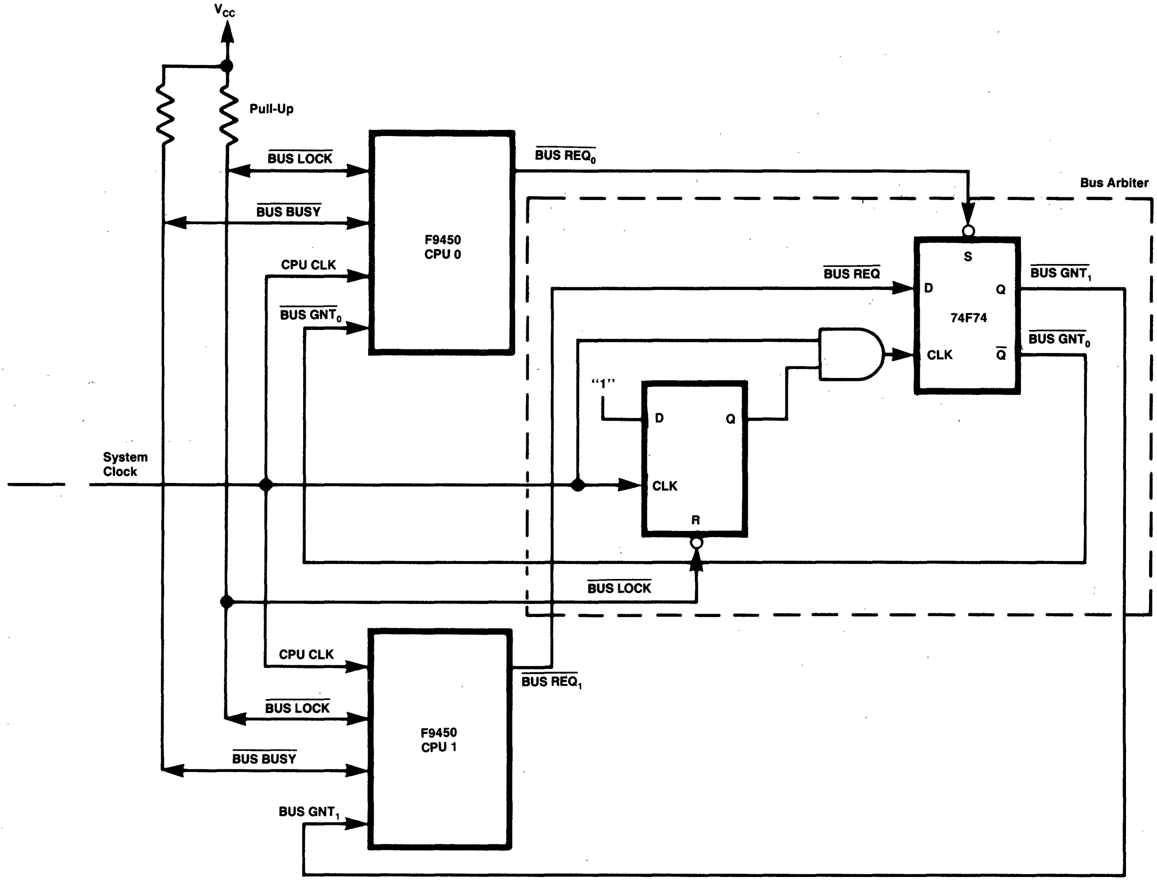

V_{CC}
Pull-Up
$\overline{\text{BUS LOCK}}$
$\overline{\text{BUS BUSY}}$
CPU CLK
$\overline{\text{BUS GNT}_0}$
F9450
CPU 0
$\overline{\text{BUS REQ}_0}$
Bus Arbiter
S
$\overline{\text{BUS REQ}}$
D
Q
74F74
CLK
$\overline{\text{Q}}$
$\overline{\text{BUS GNT}_1}$
$\overline{\text{BUS GNT}_0}$
"1"
D
Q
CLK
R
System
Clock
$\overline{\text{BUS LOCK}}$
CPU CLK
$\overline{\text{BUS REQ}_1}$
$\overline{\text{BUS LOCK}}$
$\overline{\text{BUS BUSY}}$
F9450
CPU 1
BUS GNT_1

Figure 6.18 Dual CPU with External Arbiter

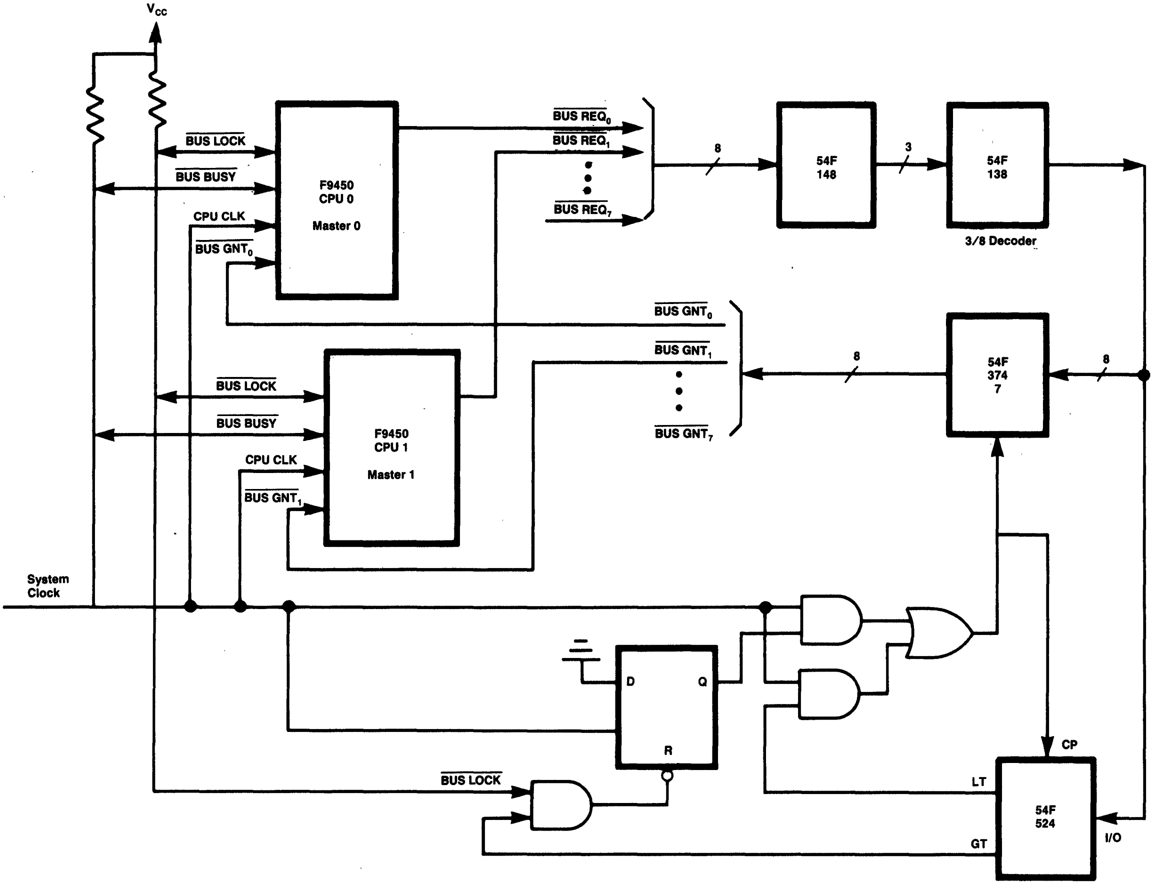


Figure 6.19 Multiple Processors with Prioritized Arbitration

FAIRCHILD

A Schlumberger Company

F9450 Environment

In July of 1984, the U. S. Department of Defense validated the Fairchild F9450 for being in total compliance with MIL-STD-1750A and all other applicable military standards. This chapter describes the F9450's environmental characteristics for designers utilizing it.

The information presented in this chapter is complete and current as of the time this book was written. It should be recognized, however, that while the contents of the book are static, the part itself is undergoing a continuous process of refinement and improvement. Consequently, this information is subject to change without notice. When basing detailed engineering decisions on the data given here, it is advisable to verify the values in question. This can be done by contacting your Fairchild representative.

Power Requirements

The collector power supply (V_{CC}) is an input to pin 64 on the F9450. *See Table 6.1* for specifics.

The chip is grounded to 0 V reference via pins 1 and 32. In order to avoid power imbalance within the chip's internal circuitry, these pins should be tied together as close to the chip as possible and with conductors of equal length.

Similarly, the injector pins I_{INJ1} and I_{INJ2} (pins 19 and 46 respectively) should be tied together symmetrically under the chip socket to prevent current hogging.

The F9450 requires 1.1 amp current supply at I_{INJ}, with a tolerance of ±5%. The maximum allowable current level at I_{INJ} is 2.0 amps. Note that the chip may fail to operate at such a high current level, but will resume operation upon return to 1.1 amp ±5%. A sustained current in excess of 2.0 amps may impair the reliability of the device. The maximum allowable voltage at I_{INJ} is 2.0 V, and any voltage in excess of that level will destroy the part. As a result, input to pins 19 and 46 should be clamped at 2.0 volts and 2.0 amps.

Table 7.1 describes other DC characteristics of the F9450.

Absolute Minimum/Maximum Ratings

The absolute minimum and maximum ratings of the F9450 are as follows:

Storage temperature	−65°C, +150°C
Operating temperature under bias	−55°C, +125°C
V_{CC} pin potential to ground pin	−0.5V, +6.0V
Input voltage	−0.5V, +5.5V
Input current	−20mA, +5 mA
Output voltage (output high)	−0.5V, +5.5V
Output current (dc output low)	+20mA
Injector current (I_{INJ})	2.0A

These are stress ratings only, and do not imply functional operation at these ratings or under any other extreme conditions. Exposure to the absolute rating conditions for extended periods may impair the reliability of the device, and exceeding these ratings may cause permanent damage.

Table 7.2 furnishes recommended operating ranges for the F9450.

Packaging

The F9450 comes in two standard packages, one for DIP mounting, the other a surface-mount gull wing configuration. *Figures 7.1* and *7.2* furnish details of the standard packages, and *Table 7.3* provides operating ranges for the case configurations.

The F9450 is available with 100 percent screening and quality conformance inspection required by MIL-STD-883.

Table 7.1 F9450 DC Characteristics

Symbol	Characteristic	Min	Typ	Max	Unit	Test Conditions
V_{IH}	Input High Level	2.0			V	Guaranteed Input High
V_{IL}	Input Low Level			0.8	V	Guaranteed Input Low
V_{T+}	Positive-Going Threshold Voltage USR_0 INT–USR_5 INT, $SYSFLT_0$, $SYSFLT_1$, PWRDN INT	1.5	1.8	2.0	V	V_{CC} = 5.0 V I_{INJ} = Typ.
V_{T-}	Negative-Going Threshold Voltage USR_0 INT–USR_5 INT, $SYSFLT_0$, $SYSFLT_1$, PWRDN INT	0.6	0.95	1.1	V	V_{CC} = 5.0 V I_{INJ} = Typ.
$(V_{T+})–(V_{T-})$	Hysteresis, USR_0 INT–USR_5 INT, $SYSFLT_0$–$SYSFLT_1$, PWRDN INT	0.4	0.8		V	V_{CC} = 5.0 V I_{INJ} = Typ.
V_{CD}	Input Clamp Voltage		–0.9	–1.5	V	I_{IN} = –18 mA V_{CC} = Min. I_{INJ} = Typ.
V_{OH}	Output High Voltage	2.4	3.2		V	I_{OH} = –0.4 mA V_{CC} = Min. I_{INJ} = Typ.
V_{OL}	Output Low Voltage		0.25	0.5	V	I_{OL} = 8.0 mA V_{CC} = Min. I_{INJ} = Typ.
I_{IH}	Input High Current Except IB_0–IB_{15}, $\overline{\text{BUS BUSY}}$, $\overline{\text{BUS LOCK}}$			40	µA	V_{IN} = 2.7 V I_{INJ} = Typ.
I_{IH}	Input High Current IB_0–IB_{15}, $\overline{\text{BUS BUSY}}$, $\overline{\text{BUS LOCK}}$			100	µA	V_{IN} = 2.7 V I_{INJ} = Typ.
I_{IH}	Input High Current All Inputs			1.0	mA	V_{IN} = V_{CC} Max. I_{INJ} = Typ.
I_{IL}	Input Low Current		–200	–400	µA	V_{IN} = 0.4 V V_{CC} = Max. I_{INJ} = Typ.
I_{OZH}	Output 3-State Current IB_0–IB_{15}, $\overline{\text{BUS BUSY}}$, $\overline{\text{BUS LOCK}}$			140	µA	V_{IN} = 2.4 V V_{CC} = Max. I_{INJ} = Typ.
I_{OZH}	Output 3-State Current AK_0–AK_3, AS_0–AS_3, $R/\overline{W}$, $M/\overline{IO}$, $D/\overline{I}$, STRBA, $\overline{\text{STRBD}}$			100	µA	V_{IN} = 2.4 V V_{CC} = Max. I_{INJ} = Typ.

Table 7.1 F9450 DC Characteristics (Continued)

Symbol	Characteristic	Min	Typ	Max	Unit	Test Conditions
I_{OZL}	Output 3-State Current IB$_0$–IB$_{15}$, $\overline{\text{BUS BUSY}}$, $\overline{\text{BUS LOCK}}$			-500	μA	V_{OUT} = 0.5 V V_{CC} = Max. I_{INJ} = Typ.
I_{OZL}	Output 3-State Current AK$_0$–AK$_3$, AS$_0$–AS$_3$, R/W, M/IO, D/I, STRBA, $\overline{\text{STRBD}}$			-100	μA	V_{OUT} = 0.5 V V_{CC} = Max. I_{INJ} = Typ.
I_{OSH}	Output Short Circuit*	-15		-100	mA	V_{OUT} = 0 V V_{CC} = Max. I_{INJ} = Typ.
I_{CC}	Power Supply Current		230	350	mA	V_{CC} = Max. I_{INJ} = Typ.
V_{INJ}	Injector Voltage	1.2	1.4	1.6	V	I_{INJ} = Max.

* Not more than one output shorted at one time.

Table 7.2 Recommended Operating Ranges

Part Number	Supply Voltage (V_{CC}) (Volts)			Injector Current (I_{INJ}) (Amps)			
	Min	Typ	Max	Min	Typ	Max	Case Temperature (°C)
F9450DC	4.75	5.0	5.25	1.0	1.1	1.2	0 to +85
F9450DM	4.75	5.0	5.25	1.0	1.1	1.2	-55 to +125
F9450GC	4.75	5.0	5.25	1.0	1.1	1.2	0 to +85
F9450GM	4.75	5.0	5.25	1.0	1.1	1.2	-55 to +125

Table 7.3 Case Operating Ranges

Order Code	Case Temperature (T_C) Operating Range (°C)	Package Type
F9450DC	0 to +85	Ceramic DIP
F9450GC	0 to +85	Gull Wing Surface Mount
F9450-15DC	0 to +85	Hermetic DIP
F9450-20DC	0 to +85	Hermetic DIP
F9450-15GC	0 to +85	Hermetic Gull Wing Surface Mount
F9450-20GC	0 to +85	Hermetic Gull Wing Surface Mount
F9450-15DM	-55 to +125	Hermetic DIP
F9450-20DM	-55 to +125	Hermetic DIP
F9450-15GM	-55 to +125	Hermetic Gull Wing Surface Mount
F9450-20GM	-55 to +125	Hermetic Gull Wing Surface Mount

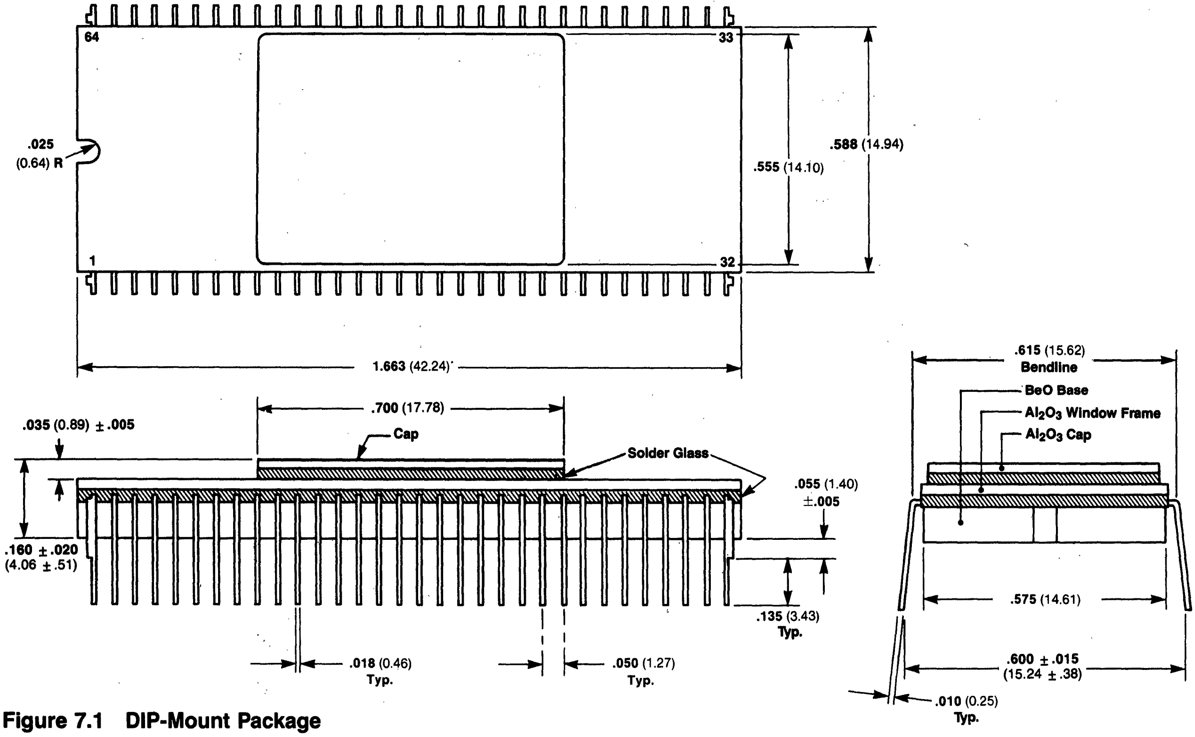


Figure 7.1 DIP-Mount Package

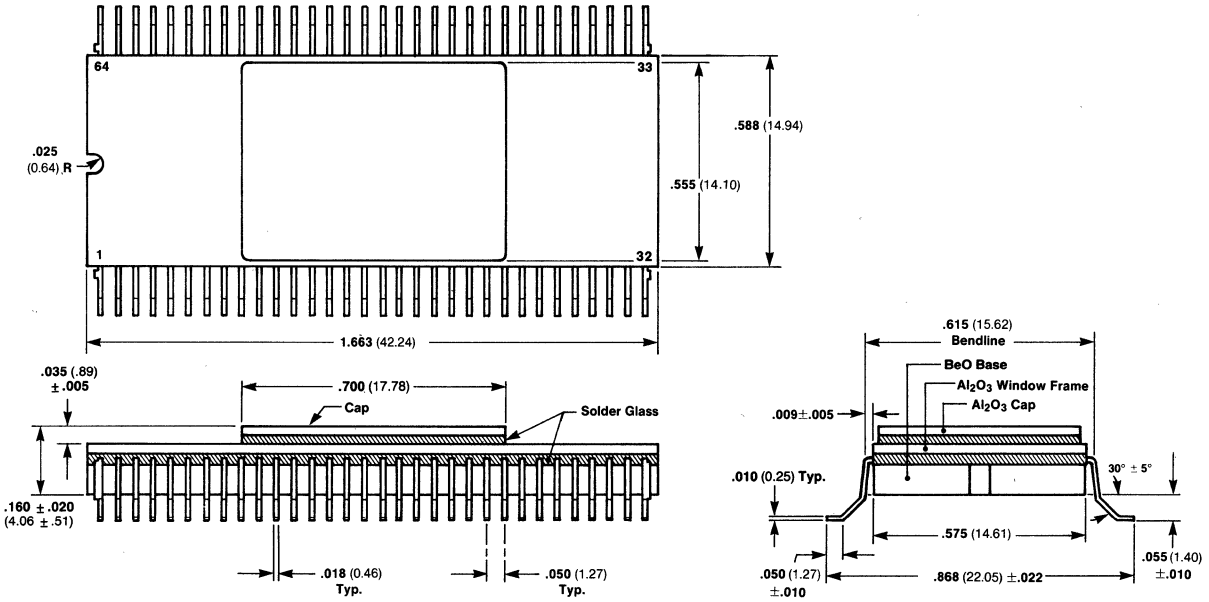


Figure 7.2 Gull-Wing Package

FAIRCHILD
A Schlumberger Company

F9451 Memory Management Unit

MIL-STD-1750A describes an optional scheme to extend memory addressing beyond 64K words. The F9451 Memory Management Unit (MMU) is a monolithic implementation of this option. It allows addressing of up to 1M word of memory with access lock/access key comparison, execute, and write protection complying with the Military Standard. In addition, applications not requiring adherence to the standard can use the F9451 MMU to address up to 16M words by utilizing 1750A-reserved bits in the physical page address.

Designed to work in conjunction with the F9450 CPU, the F9451 performs such memory management functions as logical-to-physical address translation, protection of logical space in 4K word granularity, on-chip address cache, and two translation maps (instructions and data).

Like the F9450 CPU and the F9452 Block Protect Unit, the F9451 is fabricated using the high-performance Isoplanar Integrated Injection Logic (I^3L) technology.

Functional Description

Figure 8.1 shows the block diagram of the F9451, and *Figure 8.2* illustrates the relationship of an MMU and an F9450 CPU.

During memory cycles, the F9451 generates the addresses and controls for the memory-map RAMs, which in turn generate the eight most significant bits of the physical address. This address is stable from the assertion of the ADD VAL signal until the end of the cycle. The F9451 reads back five bits of protective information (Access Lock and Execute/Write protect bits) in order to detect addressing violations.

To avoid map RAM access on every memory cycle, the F9451 employs on-chip cache mechanisms. It stores the AL (Access Lock) and E/W (Execute/Write) bits from the last map RAM access into cache latches for use in following cycles, in case of a cache hit. There are two cache latches, one each for instruction and data. The F9451 determines cache hit/miss conditions by comparing the Address State (AS) bits and the four most significant logical address bits used in the last map RAM access with the current AS and logical address. Separate latches for instructions and data are used to store the logical address bits used for the last map RAM access.

During execution of I/O instructions that manipulate the page registers, data transfers directly to and from the map RAMs through the 54F245 bidirectional buffers shown in *Figure 8.2.* The F9451 generates the address for the map RAMs and the controls for both the RAMs and the buffers.

The CPU/MMU configuration in *Figure 8.2* provides a 1M word memory space, with access lock/access key, execute, and write protection. The MMU indicates the validity of the eight extended address bits (concatenated with the twelve least significant bits of the address to constitute the 20-bit physical address) by activating ADD VAL, pin 12. It also holds the RDYA input to the CPU low until the extended address bits are valid.

Any protection violation is reported to the CPU to initiate the fault handling process by pulling the $\overline{\text{MEM PRT ER}}$ pin low. The two 54F245 octal buffers provide a data path between the map SRAMs and the CPU.

The CPU/MMU/BPU configuration in *Figure 8.3* provides a 1M word addressing space with 4K page protection in logical space for lock/key, write, and execute protection, as well as 1K page write protection in the physical space. In this case, the RDYA and $\overline{\text{MEM PRT ER}}$ inputs to the CPU are wired-OR between the MMU and the BPU.

Page Registers

Page registers reside in external 256 × 9-bit static RAMs, labeled in *Figure 8.2* as Fairchild F93479 SRAMs. The page register organization, depicted in *Figure 8.4*, is as follows:

- Physical page address (PPA) in the low byte (bits 8–15).
- Three 1750A-reserved bits in the lower section of the upper byte (Bits 5–7).*
- Execute/Write protect (E/W) in bit 4.
- Access Lock (AL) in the upper nibble of the high byte (bits 0–3).

*For other than Mil-Std applications, the reserved bits may be used for additional address bits in the PPA to increase the address range of the system.

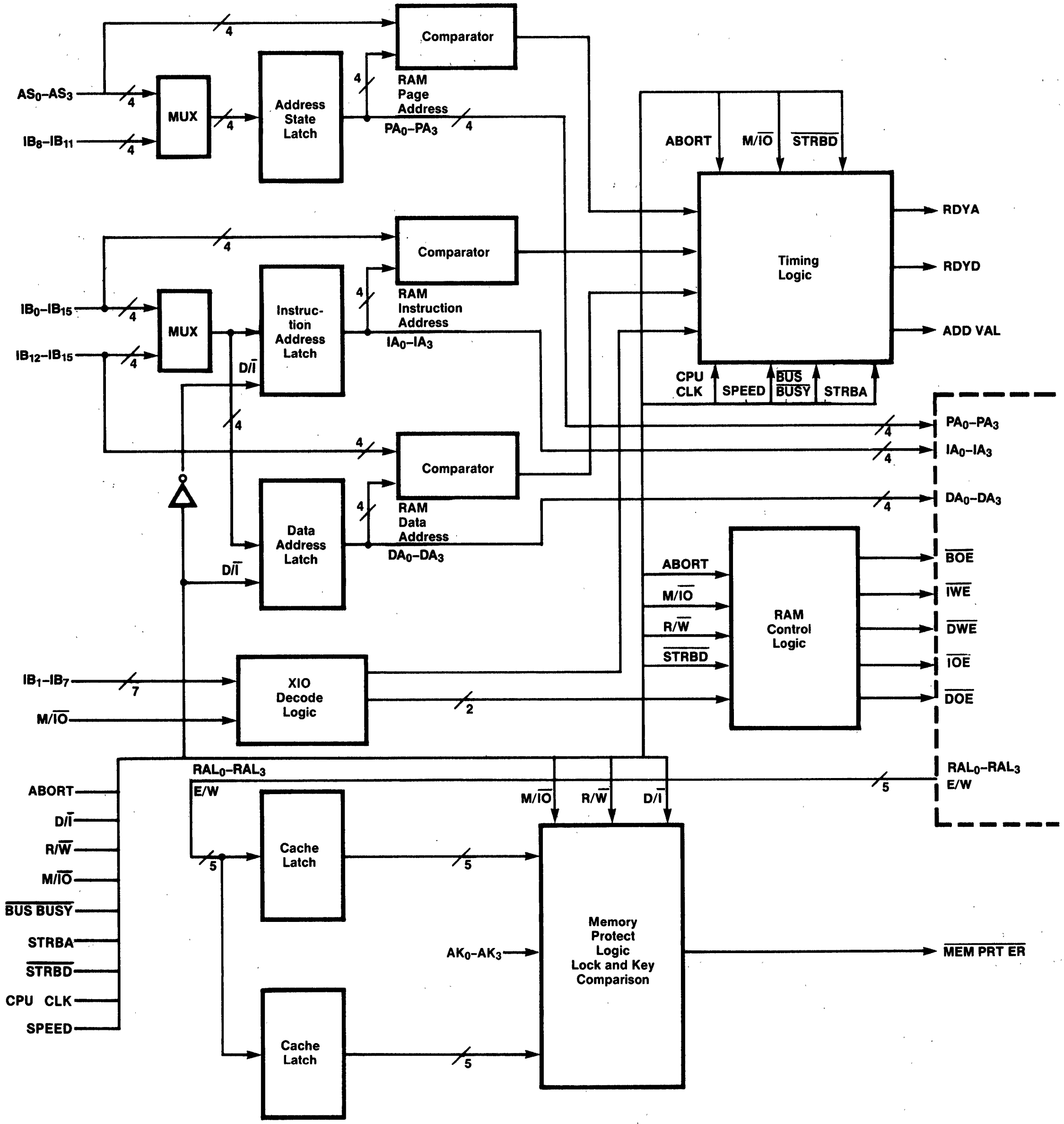


Figure 8.1 F9451 MMU Block Diagram

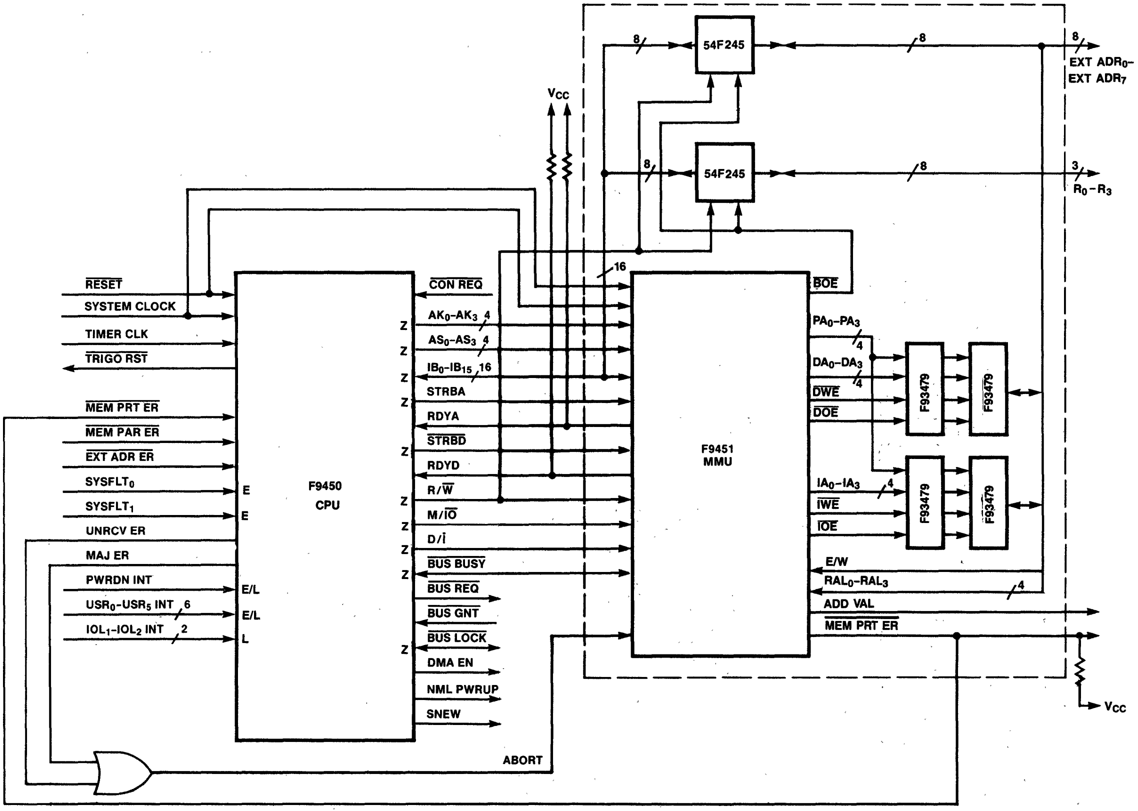


Figure 8.2 CPU/MMU Configuration

Page Register Manipulation

The F9450 CPU directly controls its associated MMU. Page register manipulation in the F9451 occurs as a result of the F9450 executing one of the instructions shown in *Table 8.1*, which are specific to the MMU. These instructions belong to the XIO class and are thus privileged.

Address Translation and Protection Mechanism

Figure 8.5 shows the mapping structure of the F9451 MMU, which is an exact implementation of memory paging as described in MIL-STD-1750A. *Figure 8.6* furnishes details for interconnection of the mapping components.

Table 8.1 Page Register Manipulation Instructions

Mnemonic	I/O command	Effect
WIPR	51XY	Write Instruction Page Register
WOPR	52XY	Write Operand Page Register
RIPR	D1XY	Read Instruction Page Register
ROPR	D2XY	Read Operand Page Register

X is the page register group
Y is the register within that group

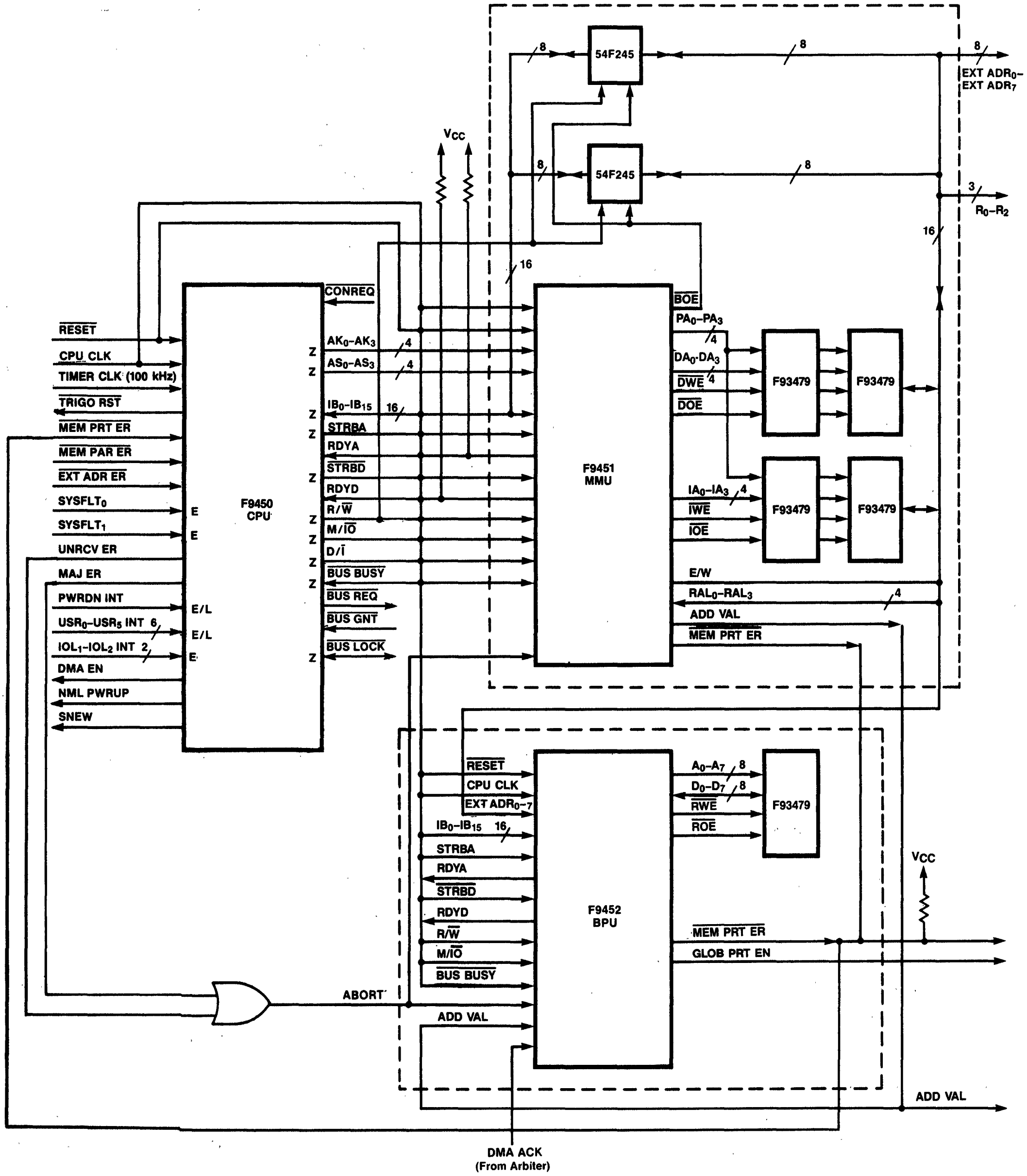


Figure 8.3 CPU/MMU/BPU Configuration

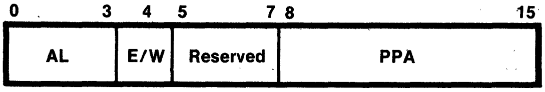


Figure 8.4 Page Register Format

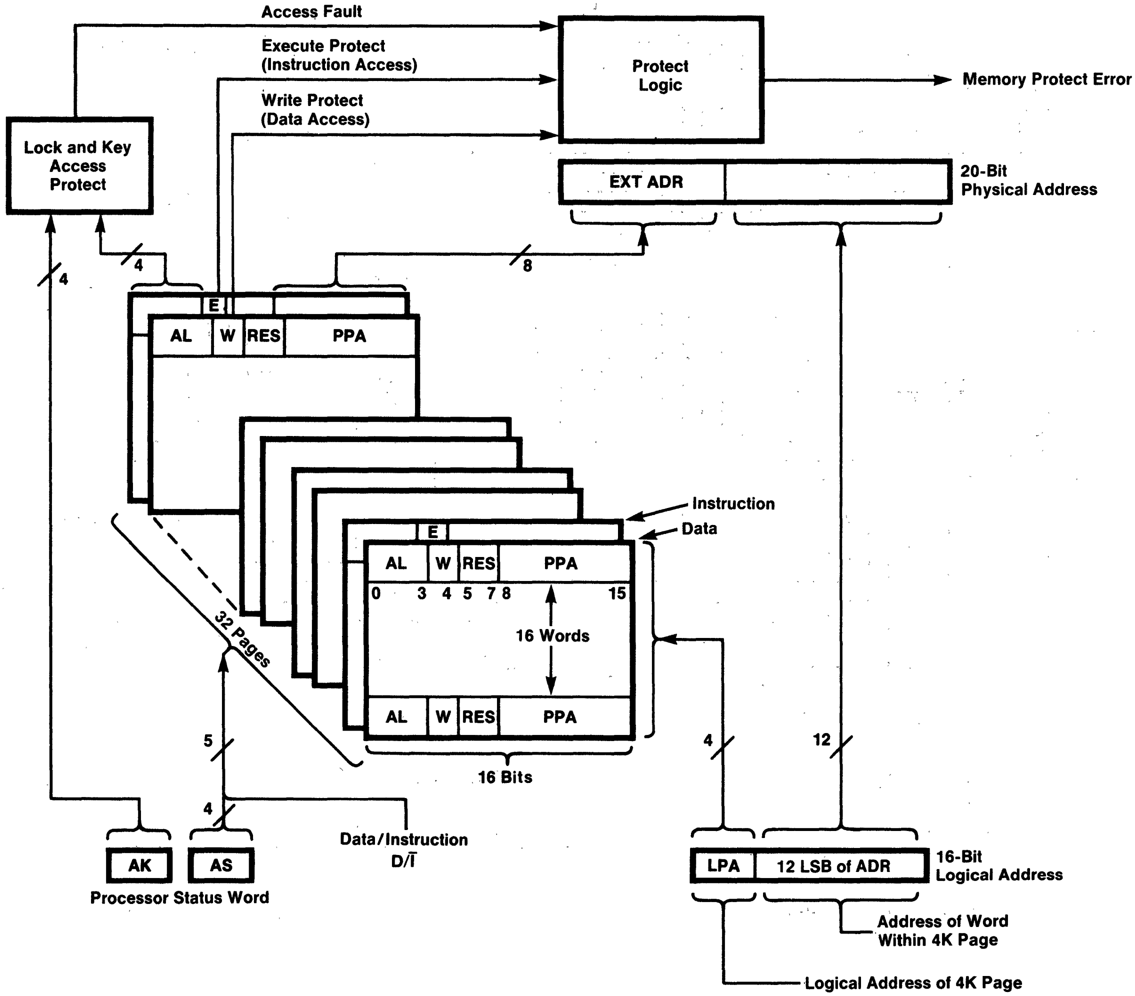


Figure 8.5 MMU Mapping Structure

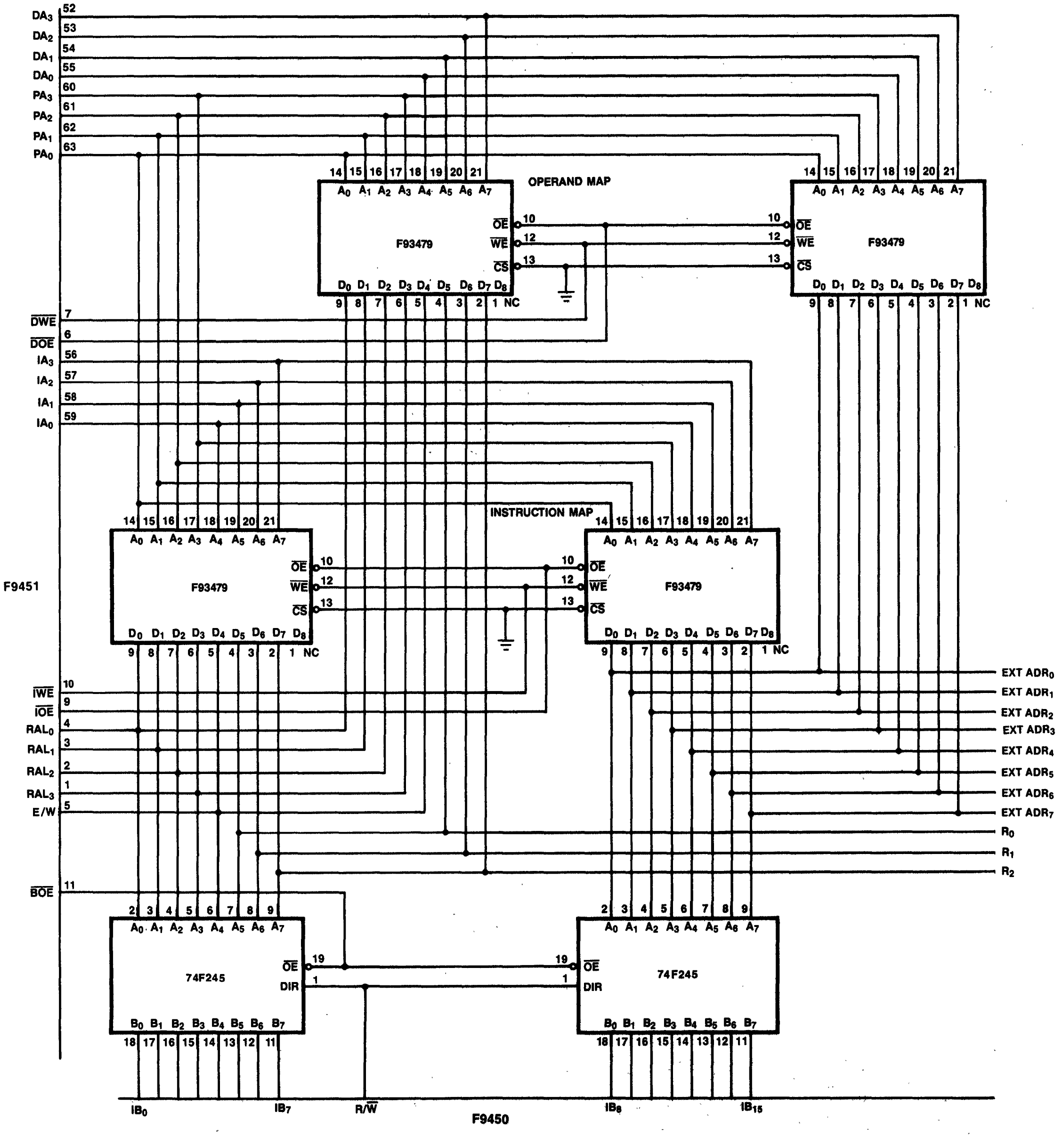


Figure 8.6 F9451 Map Interface

The memory is mapped in 4K word pages. The mapping mechanism incorporates 512 page registers that are organized into 32 groups: 16 for instruction space and 16 for data. Each group contains 16 page registers, accommodating a total of 256 registers apiece.

Any of the 32 groups is selected by the combination of the four AS bits from the F9450 SW register and the Data/Instruction (D/I) bit. The four most significant bits of the logical address identify the page register within the selected group. The eight bits of the physical page address (PPA) within the page register are then concatenated with the 12 least significant bits of the logical address, to become 20 bits of physical address.

Applications not requiring conformance with MIL-STD-1750A can use the page register's three reserved bits (5–7) as additional address bits, thus expanding addressable memory up to 8M words of data and 8M of instruction space.

The F9451 MMU employs two means to check for protection violations. It compares the Access Key (AK) from the CPU to the Access Lock (AL) field in the page register. It also examines the write protect (W) bit in the page register if data space is addressed, or the execute protect (E) bit in the page register if instruction space is addressed. If a violation occurs, it asserts the Memory Protect Error signal. Page registers are read or written by means of XIO instructions.

The access key (AK) furnished by the CPU consists of four bits, in which:

- Lock $0F_{16}$ is an "unlocked" lock and allows all keys.
- Key 00_{16} is the "master" key and is acceptable to all access lock codes.
- Keys 01_{16} through $0E_{16}$ are acceptable only to their own matched locks or the "unlocked" lock $0F_{16}$. For example, if the AK from the CPU equals $0A_{16}$, the Access Lock in the page register must be either $0A_{16}$ or $0F_{16}$, or else the MMU asserts the Memory Protect Error output.
- Key $0F_{16}$ is acceptable only to the lock $0F_{16}$.

Table 8.2 maps the relationships of MMU locks to CPU keys.

Table 8.2 Correlation of Locks and Keys

Lock	Key	Lock	Key
00	00	08	00,08
01	00,01	09	00,09
02	00,02	0A	00,0A
03	00,03	0B	00,0B
04	00,04	0C	00,0C
05	00,05	0D	00,0D
06	00,06	0E	00,0E
07	00,07	0F	All keys

Wait States

Table 8.3 shows the number of wait states introduced by the F9451 MMU and F9452 BPU, and the setting of their SPEED pins as a function of input clock frequency.

Signal Descriptions

Table 8.4 describes all the F9451 MMU input and output signals.

Timing Characteristics

Figures 8.7 through *8.11* depict the timing characteristics of the F9451 MMU, and *Figure 8.12* provides schematics of switching time test circuits.

Note the following signal directions and conditions:

- The signals CPU CLK, $\overline{\text{BUS BUSY}}$, IB_0-IB_{15}, and STATUS come from the CPU.
- The STATUS signals for the MMU include $M/\overline{IO}$, $R/\overline{W}$, D/I, AS_0–AS_3, and AK_0–AK_3.
  Timing for the CPU signals referred to here is defined in the F9450 timing characteristics (Chapter 6).

The abbreviated symbol convention used for timing parameters in this book is TAb(C)d, where:

- Timing symbols all begin with the letter "T". The mnemonic in the position represented by "A" indicates the signal node beginning the interval.

Table 8.3 MMU/BPU Wait States

Clock Rate	15–20 MHz		12–15 MHz		10–12 MHz		8–10 MHz	
MMU Speed (pin 8)	1		1		1		0	
BPU Speed 1,2 (pins9,10)	11		10		01		01	
Cycle	S1	S3	S1	S3	S1	S3	S1	S3
I/O								
To MMU	4	2	4	2	4	2	1	0
To other devices	2	0	2	0	2	0	1	0
Memory Read								
MMU/BPU: MMU Hit	2	0	2	0	2	0	1	0
MMU/BPU: MMU Miss	4	0	4	0	4	0	2	0
Memory Write								
MMU BPU								
Hit Hit	4	0	3	0	2	0	1	0
Hit Miss	5	0	4	0	3	0	2	0
Miss Hit	4	0	4	0	4	0	2	0
Miss Miss	5	0	4	0	4	0	2	0
MMU Only								
Hit	2	0	2	0	2	0	1	0
Miss	4	0	4	0	4	0	2	0

- The mnemonic in the position represented by "b" defines the direction of signal transition at the beginning node, if such definition is necessary; the new state of the signal may be low (l), high (h), three-state (z), don't care (x), or valid (v).
- The mnemonic in the position represented by "(C)", which always appears in parentheses, indicates the signal node ending the interval.
- The mnemonic in the position represented by "d" is the same as "b", but refers to the state of the signal at the node indicated by the mnemonic in position "(C)".
- The mnemonics in the positions represented by "b" and "d" are not used for reference to the CPU CLK signal, as it is assumed to be active on the rising edge.

For example, TSA(RA)$_1$ is the propagation delay time from the STRBA to the RDYA low.

Absolute Minimum/Maximum Ratings

The following are the absolute minimum and maximum ratings of the F9451 MMU:

Storage Temperature	−65°C, +150°C
Ambient Temperature Under Bias	−55°C, +125°C
V_{CC} Pin Potential to Ground Pin	−0.5V, +6.0V
Input Voltage (dc)	−0.5V, +5.5V
Input Current (dc)	−20mA, +5mA
Output Voltage (output "High" state)	−0.5V, +5.5V
Output Current (dc) (output "low" state) +20mA	
Injector Current (I_{INJ})	+500mA
Injector Voltage (V_{INJ})	−0.5V, +2.0V

These are stress ratings only, and functional operation at or beyond these or any other ratings in this book is neither implied nor recommended. Exposure to the absolute rating conditions for extended period of time may affect device reliability, and exposure to stresses greater than those listed may cause permanent damage to the device.

Recommended Operating Ranges

Table 8.5 lists the recommended operating ranges for the Fairchild F9451 MMU.

Power Requirements

Table 8.6 describes the F9451's DC power and signal requirements.

Table 8.4 **F9451 Memory Management Unit Signal Descriptions**

Mnemonic	Pin No.	Name	Description
CPU CLK	49	CPU Clock	System clock input to the F9451; a low to high transition causes a change of state in the F9451. The frequency range is 0–20 MHz. No data is lost when the clock frequency is 0.
$\overline{\text{RESET}}$	17	Reset	An active low system input used to initialize the MMU at power up. Proper initialization requires that the $\overline{\text{RESET}}$ signal be asserted for at least four clock cycles. A low indicates reset, and a high, normal operation.
BUS LINES			
AK_0–AK_3	14,15,	Access Key	Four inputs from the CPU (or any other bus master) that convey the memory access key. The MMU compares them with the lock from the selected page register and asserts the $\overline{\text{MEM PTR ER}}$ signal in case of a mismatch.
AS_0–AS_3	20–23	Address State Bus	Four inputs used by the F9451 to select the page group in the MMU memory map.
IB_0–IB_{15}	24–31 33–40	Information Bus	A 16-bit input from the main Address/Data bus on which the F9451 receives address or I/O commands from the CPU during the address phase of a bus cycle.
TIMING and STATUS			
STRBA	50	Address Strobe	An active-high input indicating the address phase of a memory or I/O cycle. A high-to-low transition latches the address.
RDYA	42	Address Ready	An open-collector output used by the F9451 to extend the address phase of a memory or I/O cycle. The F9451 pulls RDYA low prior to the beginning of a bus cycle and releases it after a variable number of S1 states. Other devices can insert additional S1 states with a wired-OR connection pbetween their open-collector RDYA outputs and this pin.
$\overline{\text{STRBD}}$	51	Data Strobe	An active-low input representing the Data phase of a bus cycle. A low-to-high transition signals termination of the $\overline{\text{IOE}}$ or $\overline{\text{IWE}}$ signals.
RDYD	43	Data Ready	An active-high three-state output to the CPU, used to extend the data phase of I/O cycles when the CPU is writing to map RAMs. This signal inserts additional S3 states; it is not enabled during memory cycles or non-MMU I/O cycles. A high input means ready, a low signifies another S3 state.
R/$\overline{\text{W}}$	44	Read/Write	An input that specifies the direction of data transfer during memory or I/O cycles. A high moves data from an I/O device or memory to the CPU or other bus master; a low moves data from the CPU or bus master to the I/O device or memory.
M/$\overline{\text{IO}}$	46	Memory-I/O	An input from the CPU to specify the cycle type. A high indicates memory cycle, a low indicates I/O.

8

Table 8.4 F9451 Memory Management Unit Signal Descriptions (Continued)

Mnemonic	Pin No.	Name	Description
D/$\overline{\text{I}}$	45	Data/Instruction	An input from the CPU that specifies the memory cycle type. A high means data reference, while a low signifies a CPU instruction fetch. This pin can be used as an additional address bit in non-MIL-STD segmented memory applications. This pin will always be high for I/O operations.
$\overline{\text{BUS BUSY}}$	47	Bus Busy	An active-low input from the CPU or any other bus master. A low-to-high transition marks the end of the current bus cycle; a high indicates the bus is idle or being rearbitrated.
MMU MAP RAM INTERFACE			
PA_0–PA_3	60–63	Page Address Bus	The four outputs from the MMU to the four MSB address inputs of the 256 × 9-bit MMU map SRAMS: directly related to AS_0-AS_3.
DA_0–DA_3	52–55	Data Address Bus	The four outputs used to address the four LSB address inputs of the two map SRAMs used during data references.
IA_0–IA_3	56–59	Instruction Address Bus	Four outputs from the F9451 used as the four LSB address inputs to the two MMU map SRAMs used for instruction references.
$\overline{\text{DWE}}$ $\overline{\text{IWE}}$	7 10	Data Write, Instruction Write Enable	The two outputs used to control the writing of the MMU map SRAMs by the CPU or any other bus master. A low means write, a high indicates read.
$\overline{\text{DOE}}$ $\overline{\text{IOE}}$	6 9	Data Output, Instruction Output Enable	The two outputs used to control the output-enables of the two groups of MMU map SRAMs for data references and instruction fetches, respectively. Active also during I/O cycles that read the contents of the RAMs to the CPU. A low signifies enable, a high means disable.
RAL_0–RAL_3	1–4	RAM Access Lock	Four inputs to the MMU from the MMU map SRAMs to convey the lock loaded in the map for the current AS and the page of memory the user is addressing. This signal is compared to the access key bus to generate the $\overline{\text{MEM PRT ER}}$ signal, if a violation has occurred.
E/$\overline{\text{W}}$	5	Execute/ Write Protect	An input from the MMU map SRAMs that signifies whether the CPU is allowed to fetch an instruction from the addressed area of memory, or is allowed to alter the memory. A high means a protect, a low means an access.
MISCELLANEOUS CONTROL SIGNALS			
$\overline{\text{BOE}}$	11	Buffer Output Enable	An active-low output used to control a 16-bit bidirectional bus transceiver between the information bus and the map RAMs. This buffer is enabled during I/O operations to the MMU map RAMs. The direction input of this buffer is driven directly from the R/W signal. A low indicates enable, a high, disable (normal).

Table 8.4 F9451 Memory Management Unit Signal Descriptions (Continued)

Mnemonic	Pin No.	Name	Description
ABORT	13	Aborted Cycle	An active-high input from the system (logical-OR of MAJ ER, UNRCV ER) used by the F9451 to inhibit alteration of any of its registers during aborted cycles initiated by the CPU. This forces the RDYA signal high after the current S1 state. A high = abort cycle, a low = normal operation.
$\overline{\text{MEM PRT ER}}$	41	Memory Protect Error	An active-low open-collector output to the CPU, memory, and other circuits signifying that the user's Access Key signal did not match the Access Lock signal. It can also indicate that the user was attempting to execute an instruction from an execution-protected area of memory, or attempting to write into a write-protected area of memory. This signal is stable prior to the high-to-low transition of the $\overline{\text{STRBD}}$ signal. A low signals a violation, and a high that the memory reference is valid. This signal is latched into the CPU Fault Register on the low-to-high transtion of the $\overline{\text{BUS BUSY}}$ signal.
ADD VAL	12	Address Valid	An active-high output to the F9452 BPU and other circuits. A low-to-high transition signifies that the Extended Address signal from the MMU map SRAMs is stable; a high-to-low transition follows the STRBA signal.
SPEED	8	Speed Select	An active-high input used to indicate the speed range of the CPU clock. A high signifies a 10 to 20 MHz clock, and a low indicates a clock of less than 10 MHz. This input should be tied to ground or through a 1K ohm resister to V_{CC}.
POWER			
V_{CC}	64	Power Supply	+5 V (DC) power supply. This powers the F9451 TTL I/O circuits.
GND	16,48	Ground	0 V reference. These pins should be tied together as close to the chip as possible.
I_{INJ}	32	Injector Current	Current source to provide bias for the Injection Logic. Typical value = 100mA.

Table 8.5 F9451 Recommended Operating Ranges

	Supply Voltage (V_{CC})			Injector Current			
Part Number	Min	Typ (V)	Max	Min	Typ (mA)	Max	Case Temperature (°C)
F9451DC	4.75	5.0	5.25	90	100	120	0 to +85
F9451DM	4.50	5.0	5.50	90	100	120	−55 to +125
F9451GC	4.75	5.0	5.25	90	100	120	0 to +85
F9451GM	4.50	5.0	5.50	90	100	120	−55 to +125

Table 8.6 F9451 DC Characteristics

Symbol	Characteristic	Min	TYP	Max	Units	Test Condition
V_{IH}	Input High Level	2.0			V	Guaranteed Input High
V_{IL}	Input Low Level			0.8	V	Guaranteed Input Low
V_{CD}	Input Clamp Voltage		−0.9	−1.5	V	$I_{IN} = -18mA$ V_{CC} = Min. I_{INJ} = Min.
I_{IH}	Input High Current			40	μA	$V_{IN} = 2.7V$ V_{CC} = Max I_{INJ} = Min.
I_{IH}	Input High Current			1	mA	V_{IN} = VCC V_{CC} = Max. I_{INJ} = Min.
I_{IL}	Input Low Current		−200	−400	μA	$V_{IN} = 0.4V$ V_{CC} = Max. I_{INJ} = Min.
V_{OH}	Output High Voltage Except MEM PRT ER, RDYA	2.4	3.2		V	$I_{OH} = -0.4mA$ V_{CC} = MIN. I_{INJ} = Min.
V_{OL}	Output Low Voltage		0.2	0.5	V	$I_{OL} = 8.0mA$ V_{CC} = Min. I_{INJ} = Min.
I_{OSH}	Output Short-Circuit Current*	−15		−100	mA	$V_{out} = 0V$
I_{OZH}	Output 3-State Current RDYA			100	μA	$V_{OUT} = 2.4V$ V_{CC} = Max. I_{INJ} = Min.
I_{OZL}	Output 3-State Current RDYA			−100	μA	$V_{OUT} = 0.5V$ V_{CC} = Max. I_{INJ} = Min.
I_{LOH}	O.C. Output Leakage MEM PRT ER, RDYA			1.0	mA	$V_{OH} = V_{CC}$ V_{CC} = Min. I_{INJ} = Min.
I_{CC}	Power Supply Current		120		mA	V_{CC} = Max. I_{INJ} = Min.
V_{INJ}	Injector Voltage		1.2		V	V_{CC} = Max. I_{INJ} = Min.

*Not more than one output shorted at one time.

Figure 8.7 F9451 MMU Minimum Timing (Page Boundary Not Crossed, Cache Hit)

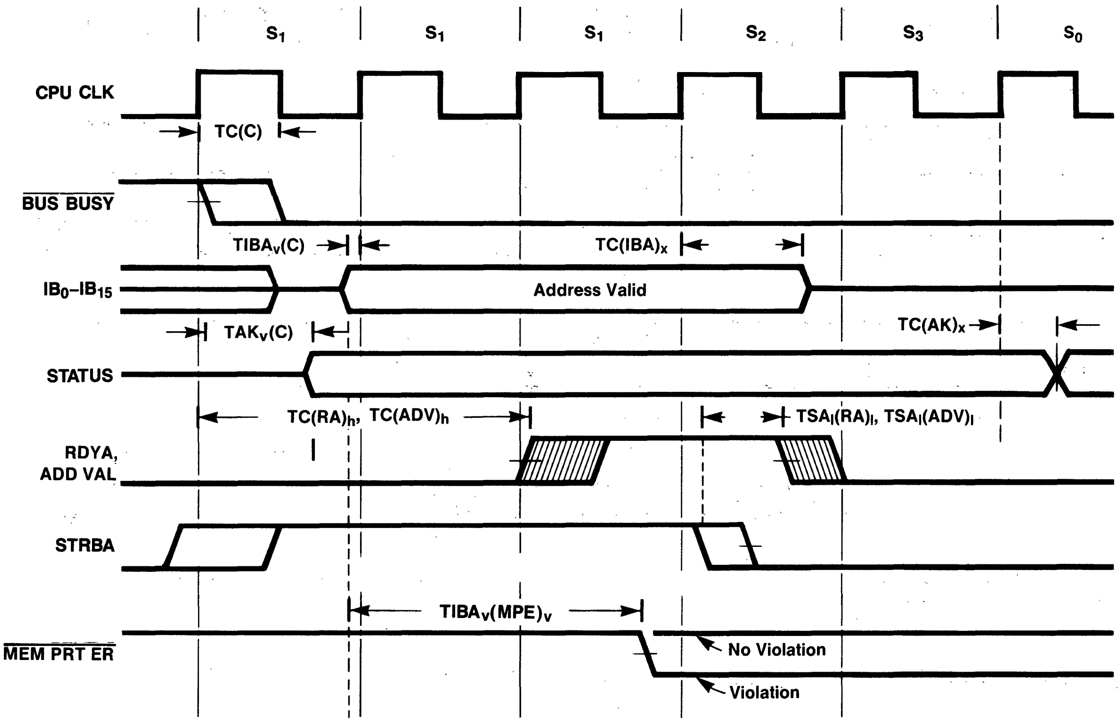


Table 8.7 Case Operating Ranges

Order Code	Case Temperature (T_C) Operating Range (°C)	Package Type
F9451DC	0 to +85	Hermetic DIP
F9451GC	0 to +85	Hermetic Gull Wing Surface Mount
F9451DM	−55 to +125	Hermetic DIP
F9451GM	−55 to +125	Hermetic Gull Wing Surface Mount

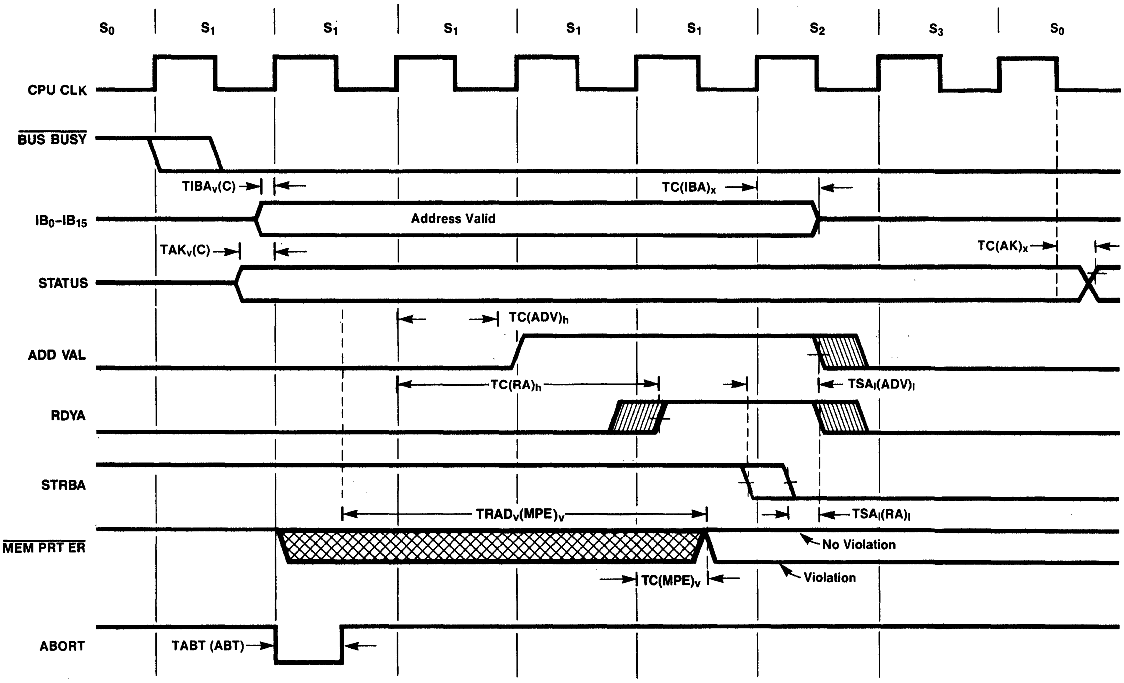


Figure 8.8 F9451 MMU Timing (Page Boundary Crossed, Cache Miss)

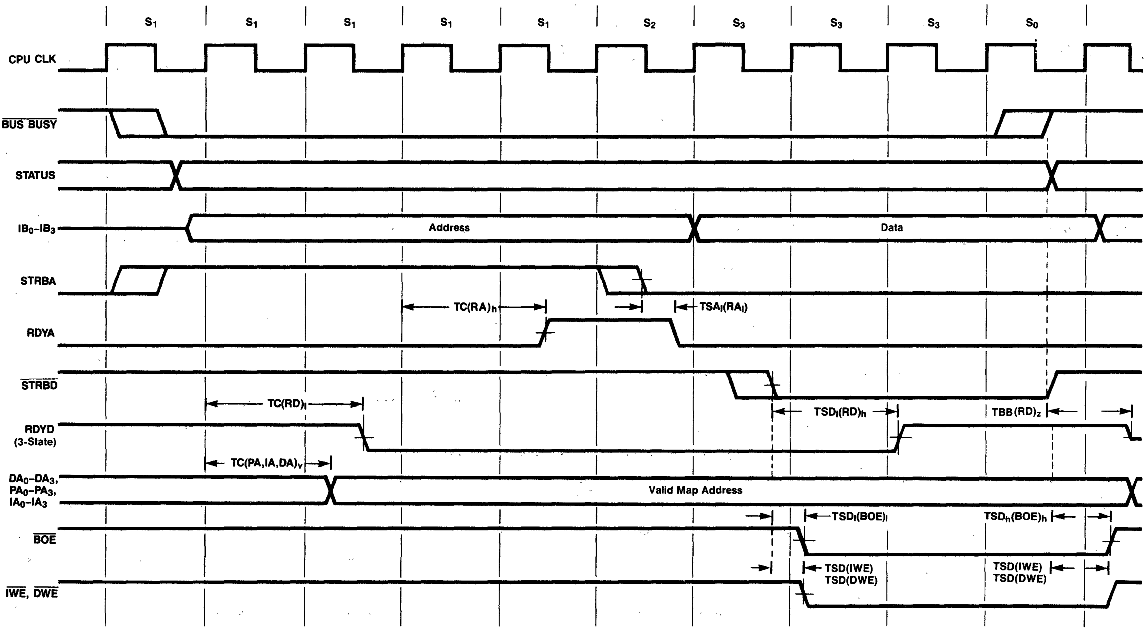


Figure 8.9 F9451 Write Timing for XIO, WIPR, and WOPR

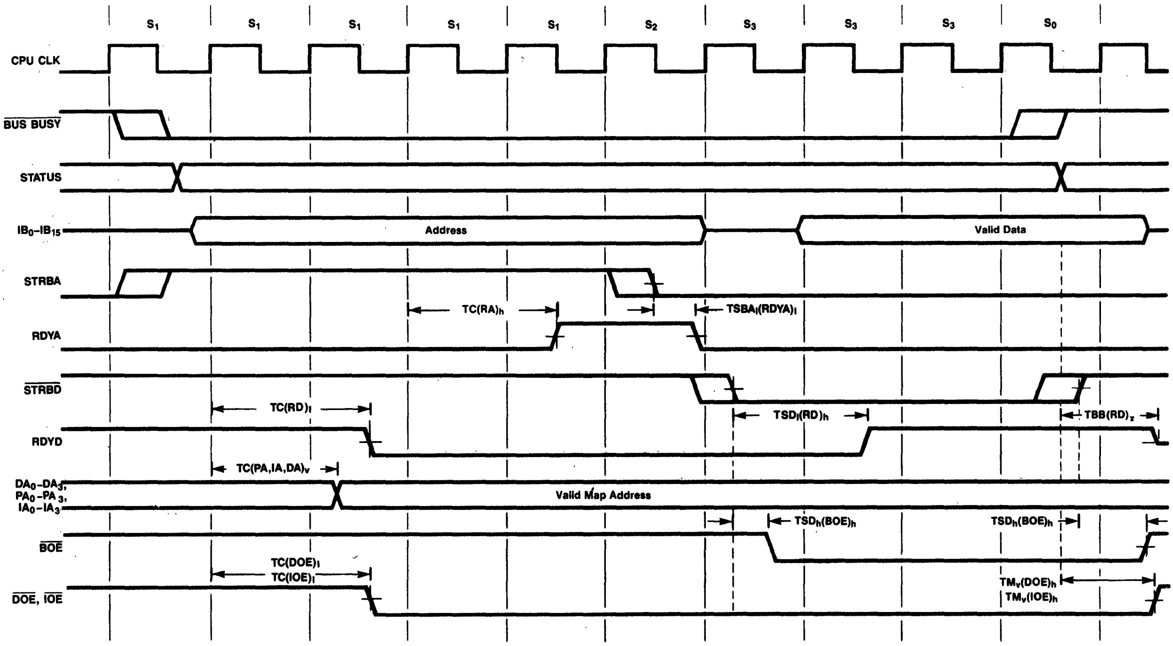


Figure 8.10 F9451 Read Timing for XIO, RIPR, and ROPR

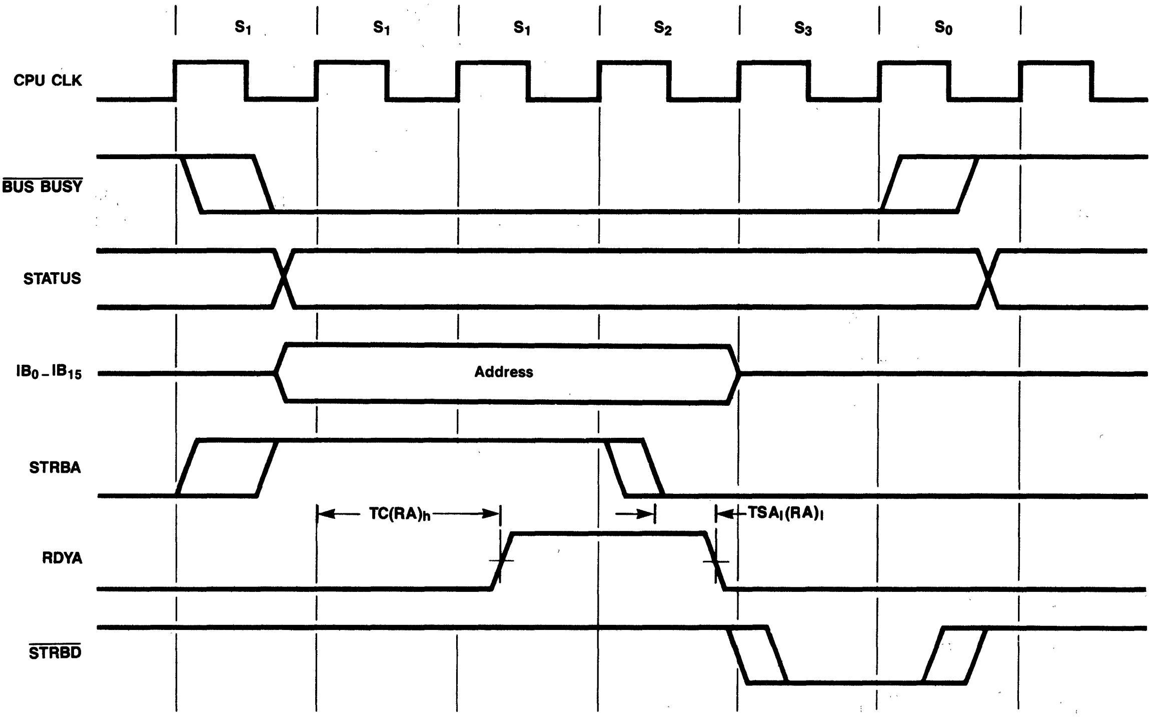
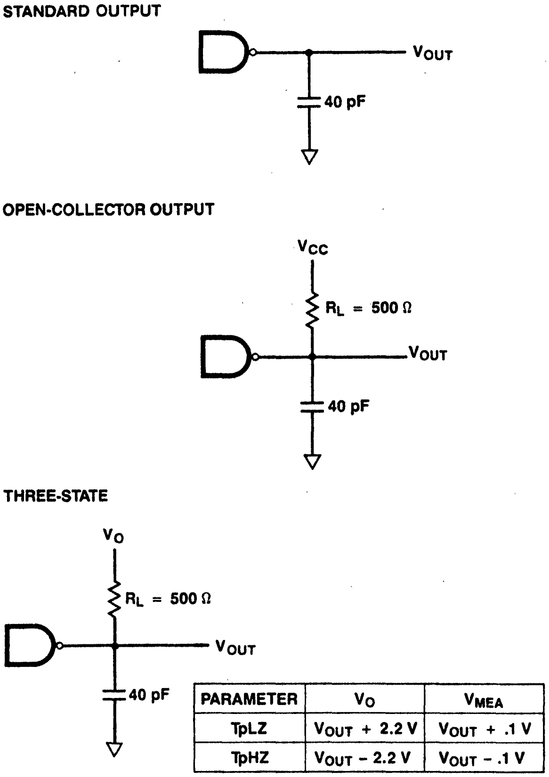


Figure 8.11 F9451 I/O Timing for XIO to Other Devices



PARAMETER	V_O	V_{MEA}
TpLZ	V_{OUT} + 2.2 V	V_{OUT} + .1 V
TpHZ	V_{OUT} - 2.2 V	V_{OUT} - .1 V

Figure 8.12 Switching Time Test Circuits

Package Information

The F9451 comes in two standard packages, one for DIP mounting, the other a surface-mount gull wing configuration. *Figures 8.12* and *8.13* furnish details of the standard packages, and *Table 8.7* provides operating ranges for the case configurations.

The F9451 is available with 100 percent screening and quality conformance inspection required by MIL-STD-883.

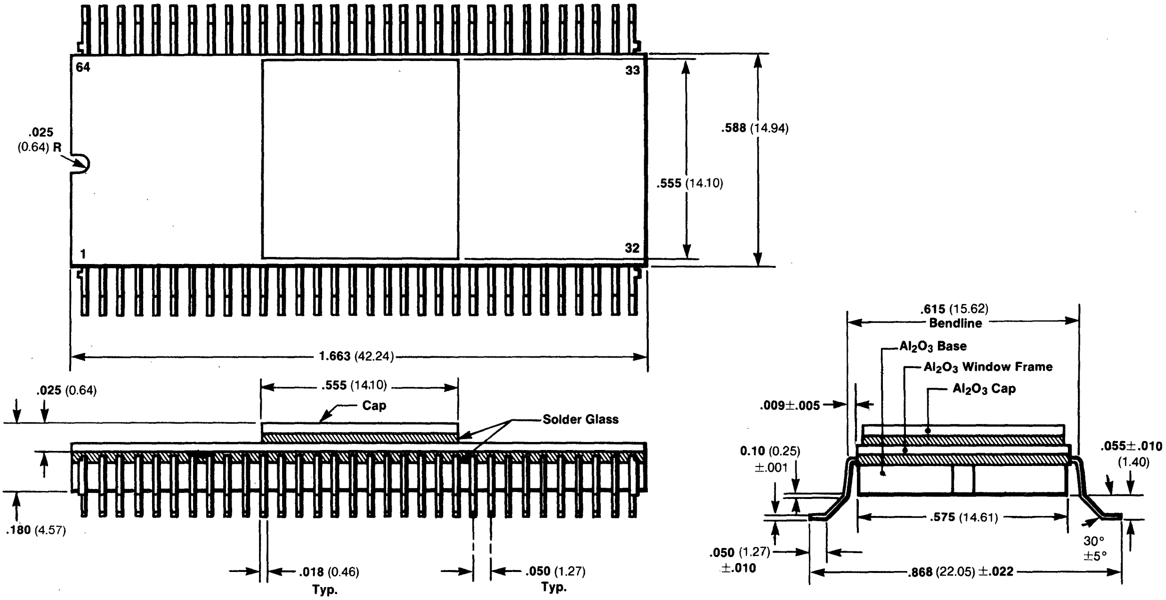


Figure 8.13 DIP-Mount Package

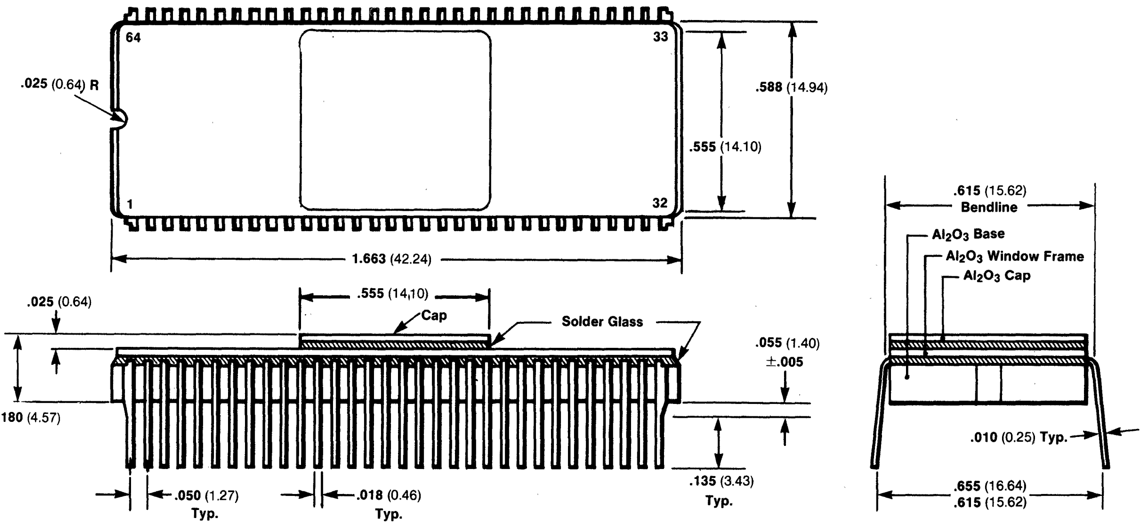


Figure 8.14 Gull-Wing Package

F9452 Block Protect Unit

Description

The Fairchild F9452 Block Protect Unit (BPU) provides write protection in 1K page granularity for up to 1M word of physical memory for both CPU and DMA access. The BPU works with the F9450 CPU, either alone or in concert with the F9451 Memory Management Unit (MMU) to implement the optional memory block protection functions described in MIL-STD-1750A, sections 4.5.4 and 5.1.

A complete BPU consists of an F9452 BPU chip and one F93479 or equivalent static RAM. The SRAM stores CPU and DMA protection tables separately. Global write protection is provided from reset until memory writes are enabled.

Like the other members of the F9450 family, the F9452 is fabricated using Fairchild's high-performance I^3L technology for reliable operation over the entire military temperature range. It is driven by a clock at up to 20 MHz, and uses TTL-compatible inputs and outputs.

Functional Description

The BPU is composed of the F9452 and one F93479 or equivalent SRAM serving as a look-up table. Each bit in the look-up table represents a 1K word page in physical memory. *Figure 9.1* depicts the organization of the BPU with the 256 × 9 SRAM.

The F9452 employs an on-chip cache mechanism to speed access operations during memory cycles. In the cache register, it stores the last 16-bit write protect word (representing 16K words) read from the look-up table RAM. The BPU uses this information in subsequent memory write cycles to detect protection violations in case of a cache hit. It evaluates cache hit/miss situations by comparing the DMA ACK and EXT ADD_0 – EXT ADD_5 bits used to address the look-up table during the last access with the current values of those bits. A cache hit saves one cycle of look-up table access.

The BPU communicates with the look-up table SRAM in bytes. On a cache miss, it reads/writes the first byte of the 16-bit protection word during the address phase of the cycle, and the second byte during the data phase. These bytes can be either the high or low bytes of a 16-bit protection word, depending on the physical memory address. During I/O instructions that read the look-up table, the BPU assembles the two bytes in the cache register and transfers them to the CPU. During instructions that write into the look-up table, the BPU first loads data into the cache register and then writes to the SRAM in two byte transfers.

Block Protect Mechanism

The BPU divides the physical memory into 1K word pages, with each page to be write-protected represented by a 1-bit in the look-up table. *Figure 9.2* shows the BPU look-up table structure. The structure consists of 128 16-bit registers: 64 for CPU access, and 64 for DMA controller access (DMA ACK = 0 or 1, respectively.)

The F9452 offers write protection for two different configuration options:

- **MMU Absent**
  In the CPU/BPU configuration shown in *Figure 9.3*, the IB_0 and IB_1 bits select one of four protection words each for CPU and DMA access. The 16 bits within each word are addressed by IB_2 through IB_5, making the maximum block-protected memory space 64K words.
- **MMU present**
  In a CPU/MMU/BPU configuration, as illustrated by *Figure 9.4*, the MMU generates extended address bits 0 through 5, which select one of 64 protection words each for CPU or DMA controller access. Each word contains 16 bits; one of these bits is selected via extended address bits 6 and 7 and information bus bits 4 and 5. The BPU ignores IB_0 through IB_3. Thus, the maximum block-protected memory space is up to 1M words, with separate protection for CPU and DMA access.

The four most significant bits of the extended address inputs should be connected to ground or, for segmented memory systems, to AS_0 through AS_3 of the CPU.

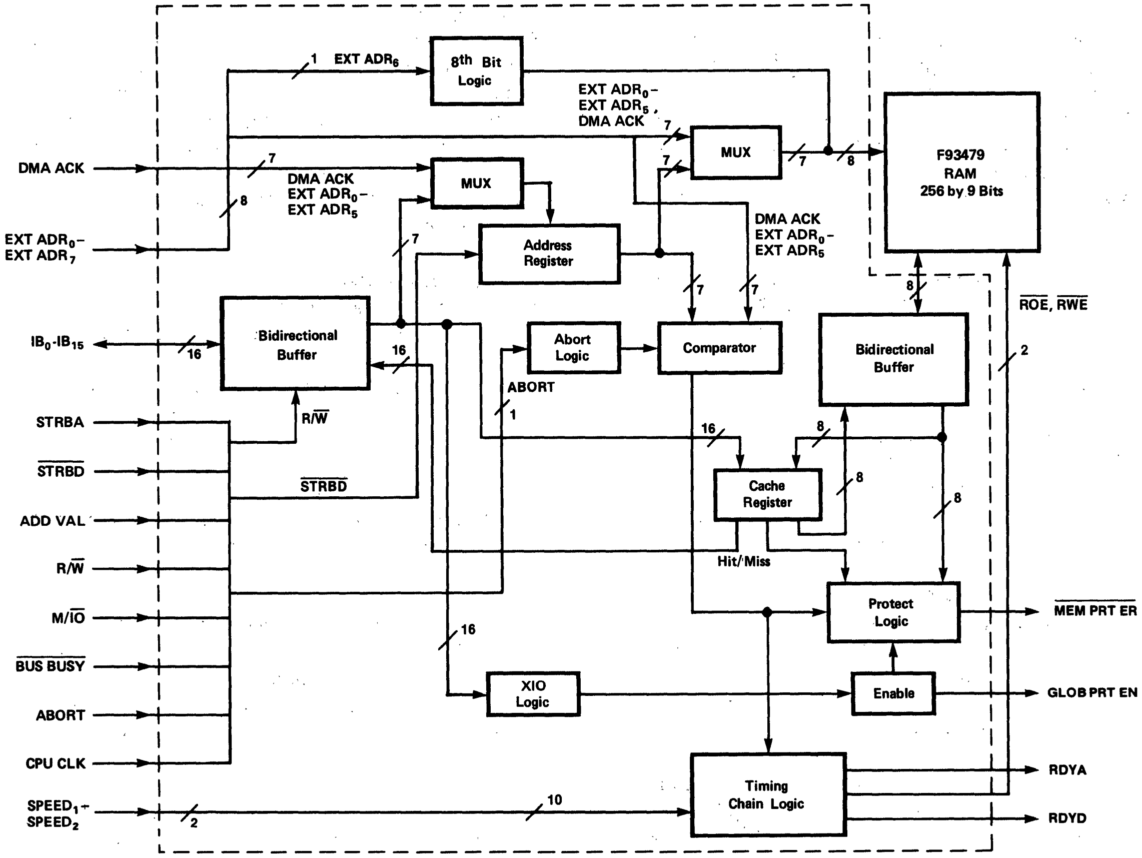


Figure 9.1 F9452 BPU Block Diagram

Basic System Configurations

The basic system configurations using the F9450 CPU and F9452 BPU are shown in *Figures 9.3* and *9.4*. In a CPU/BPU configuration, the BPU provides write protection for 1K word pages. The protection is available for the CPU memory space and the DMA memory space, as determined by the DMA ACK input. As long as the BPU is not enabled, the GLOB PRT EN output provides global memory write protection. The BPU drives the RDYA input high when it is ready and reports a write protect error by driving the $\overline{\text{MEM PRT ER}}$ output low.

The CPU/MMU/BPU configuration provides up to 1M of addressable memory with 4K-page protection in logical space for lock/key, write, and execute protection, as well as 1K word write protection in the physical space. In this case, the RDYA and $\overline{\text{MEM PRT ER}}$ inputs to the CPU are wired-OR between the MMU and BPU.

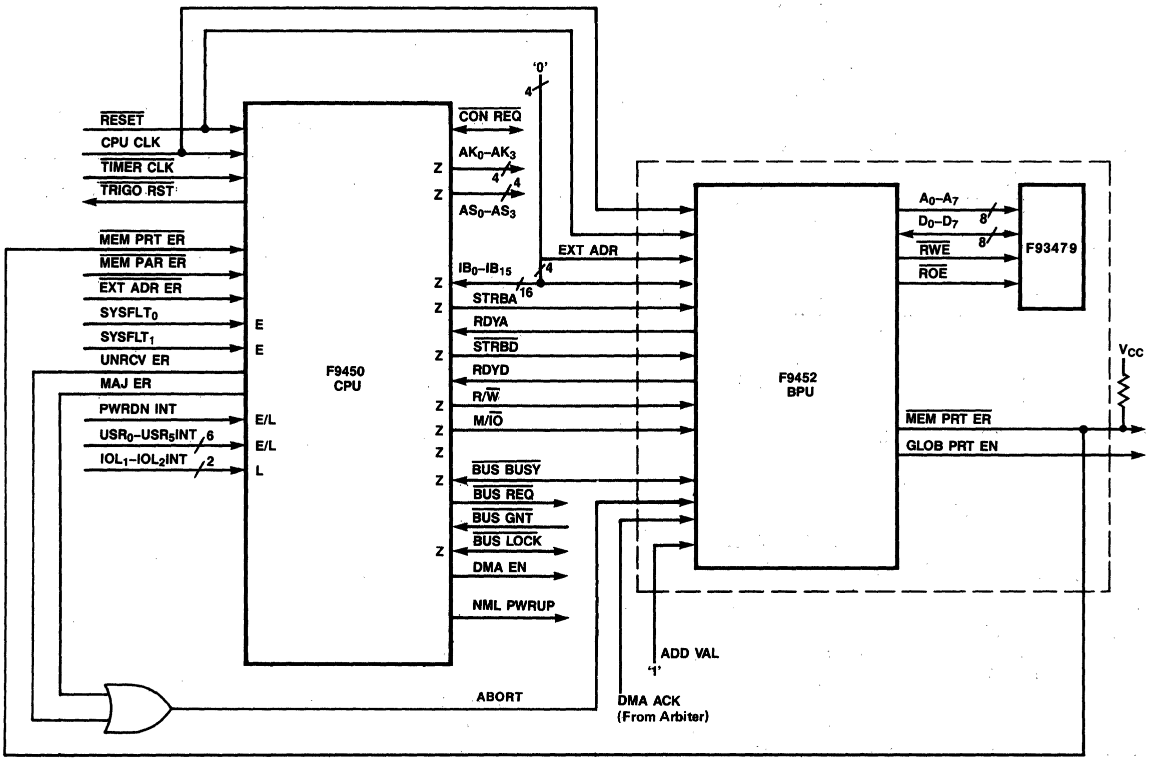


Figure 9.3 CPU/BPU Configuration

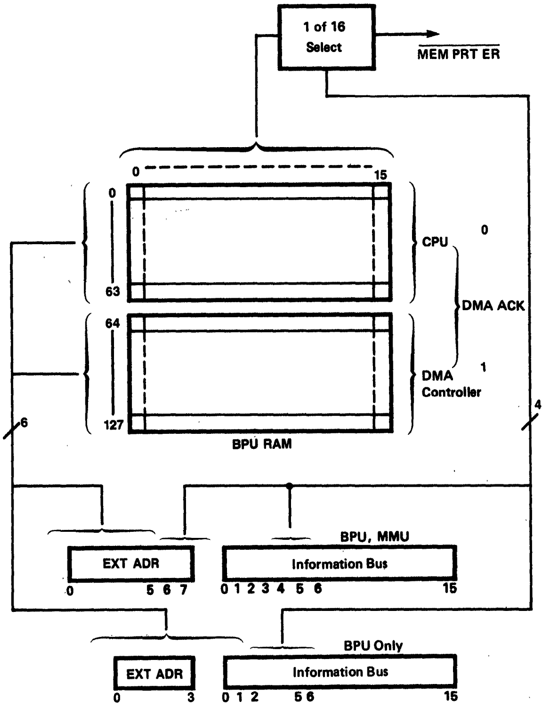


Figure 9.2 F9452 BPU Look-Up Table Structure

Instructions for Map Manipulation

The F9450 CPU instructions for manipulating the RAM map are show in *Table 9.1*. *Figure 9.5* illustrates the BPU map interface.

Table 9.1 F9450 Map Manipulation Instructions

MMU Control	I/O Command	Manipulation
LMP	50XX	Load Memory Protect RAM
RMP	D0XX	Read Memory Protect RAM
MPEN	4003	Memory Protect Enable

XX is the look-up table address, 00_{16}–FF_{16}

Wait States

Table 9.2 lists the number of wait states introduced by the F9452 BPU and the F9451 MMU.

Table 9.2 MMU/BPU Wait States

Clock Rate	15–20 MHz		12–15 MHz		10–12 MHz		8–10 MHz	
MMU Speed (pin 8)	1		1		1		0	
BPU Speed 1,2 (pins 9,10)	11		10		01		01	
Cycle	S1	S3	S1	S3	S1	S3	S1	S3
I/O								
To BPU	2	8	2	6	2	6	1	6
To Other Devices	2	0	2	0	2	0	1	0
Memory Read								
MMU/BPU: MMU Hit	2	0	2	0	2	0	1	0
MMU/BPU: MMU Miss	4	0	4	0	4	0	2	0
BPU	1	0	1	0	1	0	1	0
Memory Write								
MMU BPU								
Hit Hit	4	0	3	0	2	0	1	0
Hit Miss	5	0	4	0	3	0	2	0
Miss Hit	4	0	4	0	4	0	2	0
Miss Miss	5	0	4	0	4	0	2	0
BPU Only								
Hit	2	0	2	0	2	0	1	0
Miss	3	0	2	0	2	0	1	0

Signal Descriptions

Table 9.3 describes F9452 input and output signals.

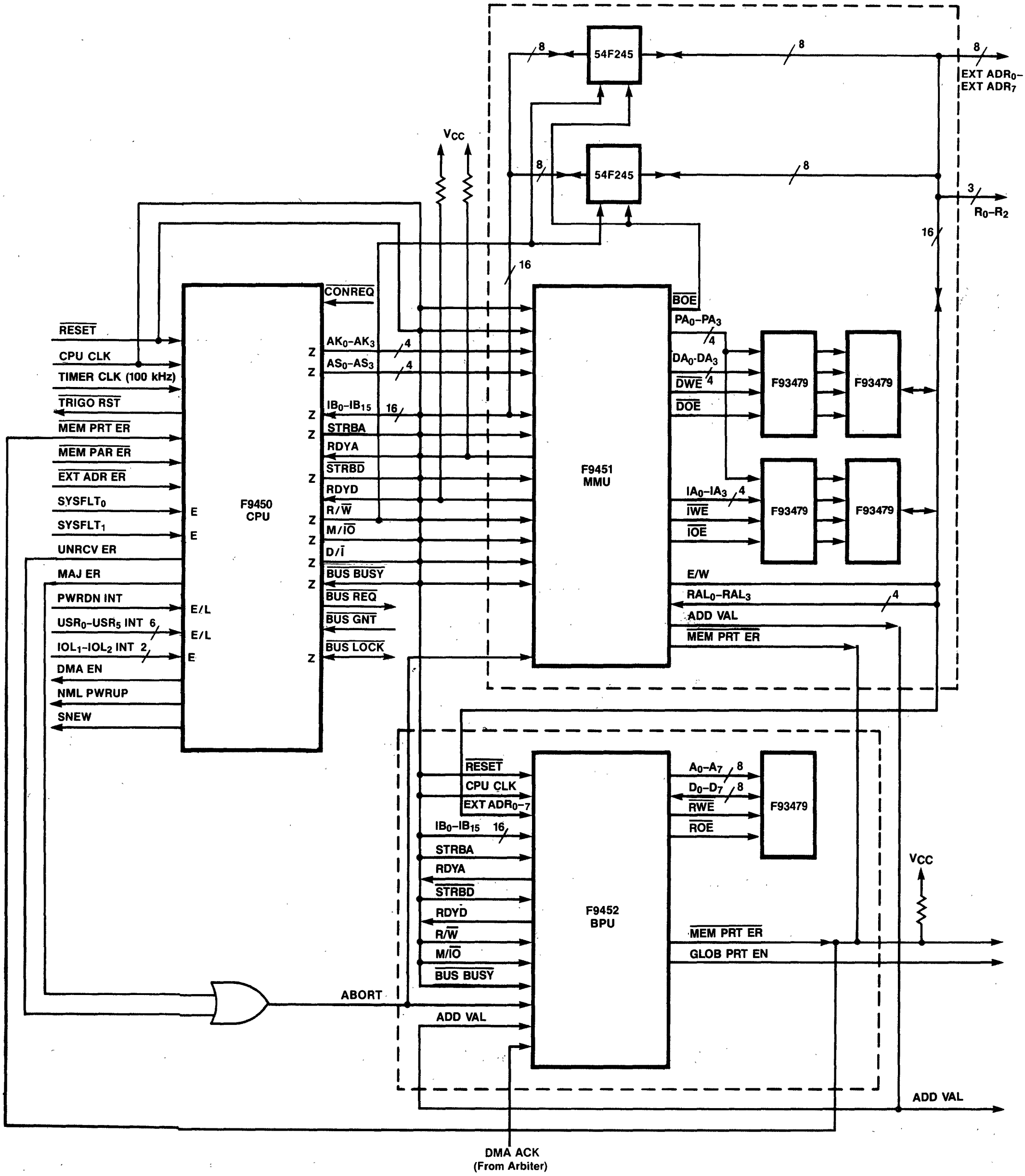


Figure 9.4 CPU/MMU/BPU Configuration

Table 9.3 BPU Signal Descriptions

Mnemonic	Pin No.	Name	Description
CPU CLK	49	CPU Clock	An active-high single-phase clock input; a low-to-high transition causes a change of state in the F9452. Frequency range is 0 to rated clock speed.
$\overline{\text{RESET}}$	17	Reset	An active-low system input used to initialize the internal circuitry and to disable the memory protection mechanism. Asserts the GLOB PRT EN signal.
BUS LINES			
IB_0–IB_{15}	24–31 33–40	Information Bus	An active-high bidirectional 16-bit bus used for communication between the various system components.
EXT ADD_0– EXT ADD_7	14,15 18–23	Extended Address	Active-high 8-bit extended address input from the MMU subsystem. If an MMU is not used in the system, these lines must be provided from some other source (See Basic System Configurations elsewhere in this chapter).
TIMING AND STATUS			
STRBA	50	Address Strobe	An active-high input used to latch the memory or I/O address from the information bus on a high-to-low transition.
RDYA	42	Address Ready	An active-high open-collector used by the BPU to insert additional S1 states.
$\overline{\text{STRBD}}$	51	Data Strobe	An active-low input used as a strobe for the data phase of the information bus.
RDYD	43	Data Ready	An active-high three-state output to the CPU used to extend the data phase of I/O cycles when the CPU is writing to the look-up tables. This inserts additional S3 states; it is not enabled during memory cycles or non-BPU I/O cycles. A high input = ready, a low = another S3 state.*
R/$\overline{\text{W}}$	44	Read/Write	An input from the CPU or DMA controller that specifies the direction of data transfer during memory or I/O cycles. A high moves data from the I/O device or memory to the CPU or DMA controller; a low moves data from the CPU or DMA controller to the BPU.
M/$\overline{\text{IO}}$	46	Memory-I/O	An input from the CPU that specifies memory or I/O cycle type: High = Memory, Low = I/O.
$\overline{\text{BUS BUSY}}$	47	Bus Busy	An active-low input from the CPU or any other bus master that indicates a bus cycle is in progress.
RAM INTERFACE			
A_0–A_7	56–63	RAM Address	An 8-bit active-high bus output of address for the look-up table RAM.
D_0–D_7	1–8	RAM Data	An 8-bit active-high bidirectional bus carrying data between the BPU and the look-up table RAM.

Table 9.3 BPU Signal Descriptions (Continued)

Mnemonic	Pin No.	Name	Description
$\overline{\text{RWE}}$	54	RAM Write Enable	An active-low output that provides a write enable signal to the look-up table RAM.
$\overline{\text{ROE}}$	53	RAM Output Enable	An active-low output that controls the output enable of the look-up table RAM: low = enable, high = disable.
GENERAL-PURPOSE CONTROLS			
$\overline{\text{MEM PRT ER}}$	41	Memory Protect Error	An active-low open collector output indicating that a memory write was attempted to a protected location.
GLOB PRT EN	55	Global Memory Protect Enable	An active-high output indicating that memory is to be globally write-protected. It is set by the $\overline{\text{RESET}}$ signal, and reset by executing the MPEN instruction.
ABORT	11	Aborted Cycle	An active-high input indicating that the cycle is aborted (logical-OR of MAJ ER,UNRCV ER in the F9450 CPU). Writing to the look-up table RAM is inhibited for the current cycle. The ABORT signal also forces a cache miss. The signal is latched and is cleared by either the next memory write or $\overline{\text{RESET}}$ signal.
DMA ACK	13	Direct Memory Access Acknowledgement	An active-high input from the arbiter used to select the appropriate bank in the memory look-up table; High = DMA, Low = CPU.
ADD VAL	12	Address Valid	An active-high output from the MMU that is used to indicate that the extended address is valid.
SPEED_1 SPEED_2	9,10	Clock Speed Data	Active-high inputs used to control the number of wait states introduced by the BPU as a function of the clock frequency range (see the "wait states" section). These inputs must be tied to ground or through a 1K ohm resister to V_{CC}.
POWER			
V_{CC}	64	Power Supply	+ 5 V (dc) power supply.
GND	16,48	Ground	0 V reference. These pins should be tied together as close to the chip as possible.
I_{INJ}	32	Injector Current	Current source to provide bias for the injection logic.

* At some time during execution of XIO instructions that transfer data to the BPU, the RDYD signal is driven high (see the "Timing Characteristics" section). The designer should ensure that no other device connected to RDYD drives the signal low at these times.

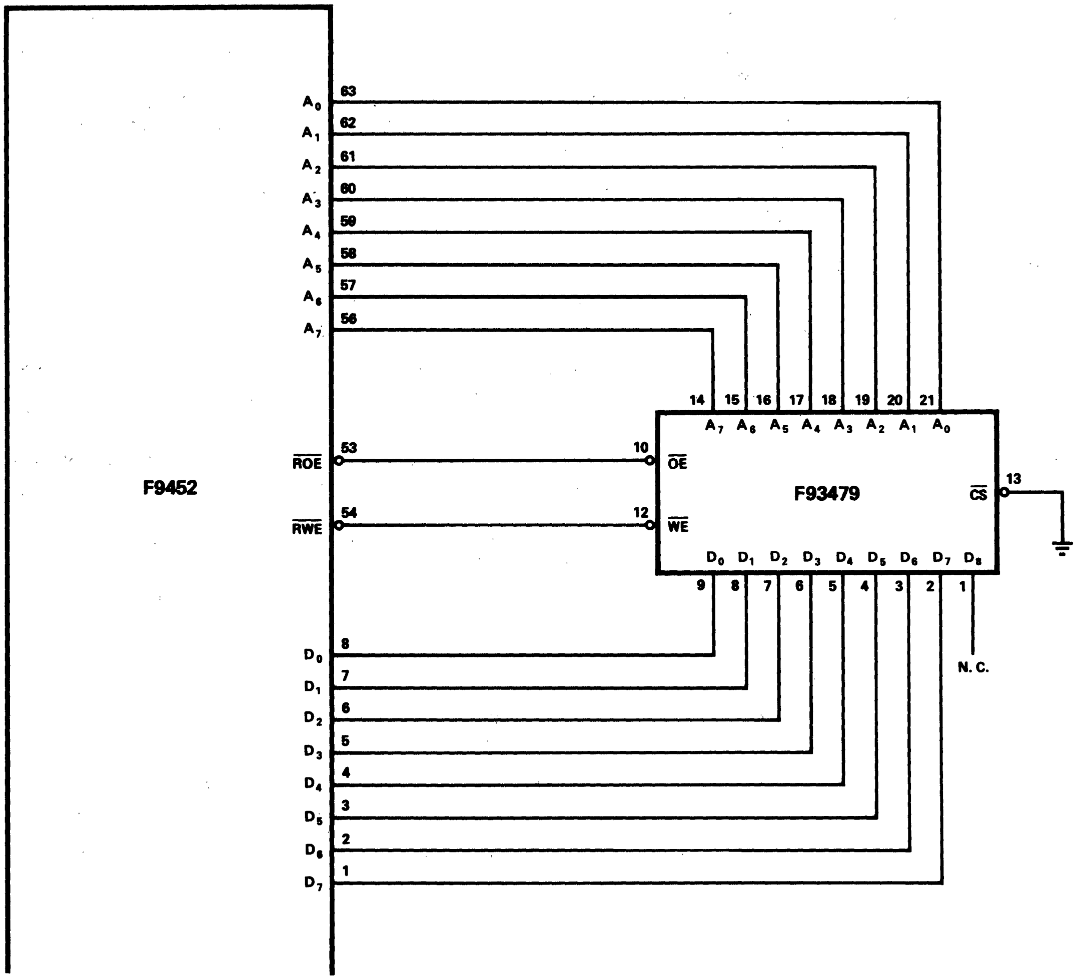


Figure 9.5 BPU Map SRAM Interface

Timing Characteristics

Figures 9.6 through *9.12* illustrate the F9452 timing characteristics, and *Figure 9.13* furnishes switching time test circuits.

Note the following signal directions and conditions:

- The signals $\overline{\text{BUS BUSY}}$, IB_0–IB_{15}, and STATUS come from the CPU.
- The STATUS signal for the BPU includes $M/\overline{IO}$ and $R/\overline{W}$.
- Timing for the CPU signals referred to here is defined in Chapter Six, *Figures 6.4* through *6.15*.

The abbreviated symbol convention used for timing parameters in this book is TAb(C)d, where:

- Timing symbols all begin with the letter "T". The mnemonic in the position represented by "A" indicates the signal node beginning the interval.
- The mnemonic in the position represented by "b" defines the direction of signal transition at the beginning node, if such definition is necessary; the new state of the signal may be low(l), high(h), 3-state(z), don't care(x), or valid(v).
- The mnemonic in the position represented by "(C)", which always appears in parentheses, indicates the signal node ending the interval.
- The mnemonic in the position represented by "d" is the same as "b", but refers to the state of the signal at the node indicated by the mnemonic in position "(C)".
- The mnemonics in the positions represented by "b" and "d" are not used for reference to the CPU CLK signal, as it is assumed to be active on the rising edge.

For example, TADV(SA)l is the setup time from ADD VAL to STRBA low.

Absolute Minimum/Maximum Ratings

The absolute Minimum and maximum ratings of the Fairchild F9452 are as follows:

Storage Temperature	−65°C, +150°C
Ambient Temperature Under Bias	−55°C, +125°C
V_{CC} Pin Potential to Ground Pin	−0.5V, +6.0V
Input Voltage (dc)	−0.5V, +5.5V
Input Current (dc)	−20mA, +5mA
Output Voltage (output High)	−0.5V, +5.5V
Output Current (dc) (Output Low)	+20mA
Injector Current(I_{INJ})	+500mA
Injector Voltage (V_{INJ})	−0.5V, +2.0V

These are stress ratings only, and functional operation at or beyond these or any other ratings for this product is neither implied nor recommended. Exposure to the absolute rating conditions for prolonged times may affect device reliability, and exposure to stresses greater than those listed may cause permanent damage to the device.

Recommended Operating Ranges

Table 9.4 lists the recommended operating ranges for the BPU.

DC Characteristics

Table 9.5 describes the dc characteristics of the F9452 BPU.

Table 9.4 F9452 Recommended Operating Ranges

Part Number	Supply Voltage (V_{CC}) Min	Typ (V)	Max	Injector Current (I_{INJ}) Min	Typ (mA)	Max	Case Temperature (°C)
F9452DC	4.75	5.0	5.25	190	200	230	0 to +85
F9452DM	4.50	5.0	5.50	190	200	230	−55 to +125
F9452GC	4.75	5.0	5.25	190	200	230	0 to +85
F9452GM	4.5	5.0	5.5	190	200	230	−55 to +125

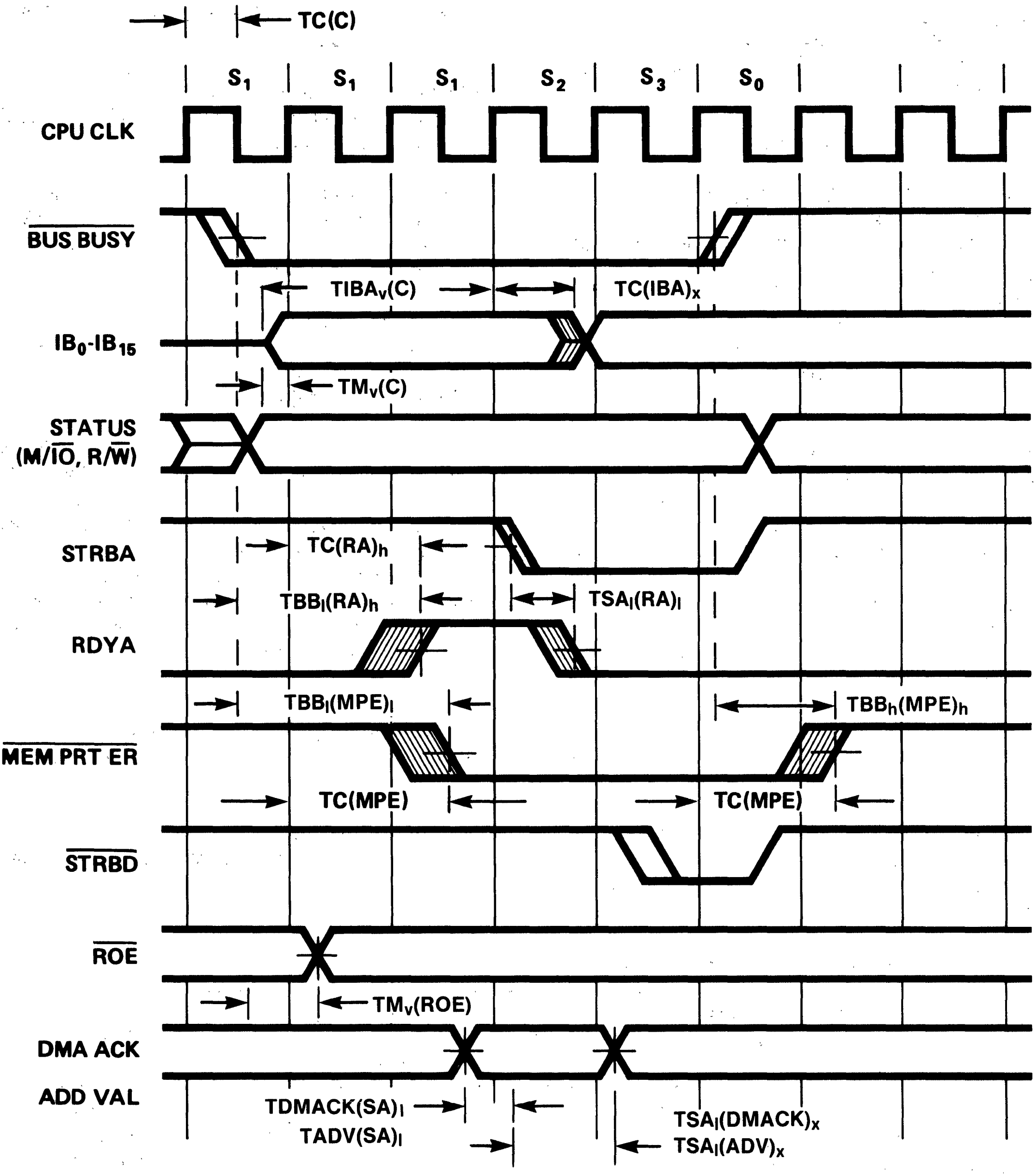


Figure 9.6 F9452 Memory Write Cycle Timing with Memory Protect Error and Internal Cache Hit

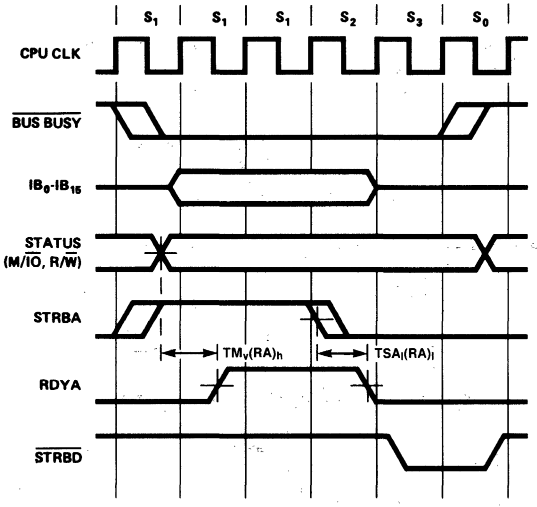


Figure 9.8 F9452 Memory Read Cycle Timing

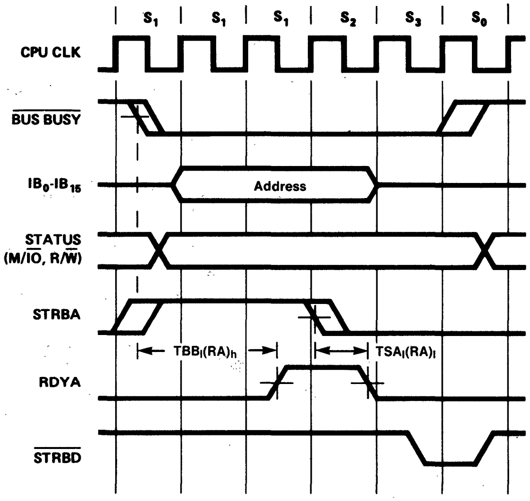


Figure 9.9 F9452 Timing for XIO to Other Devices

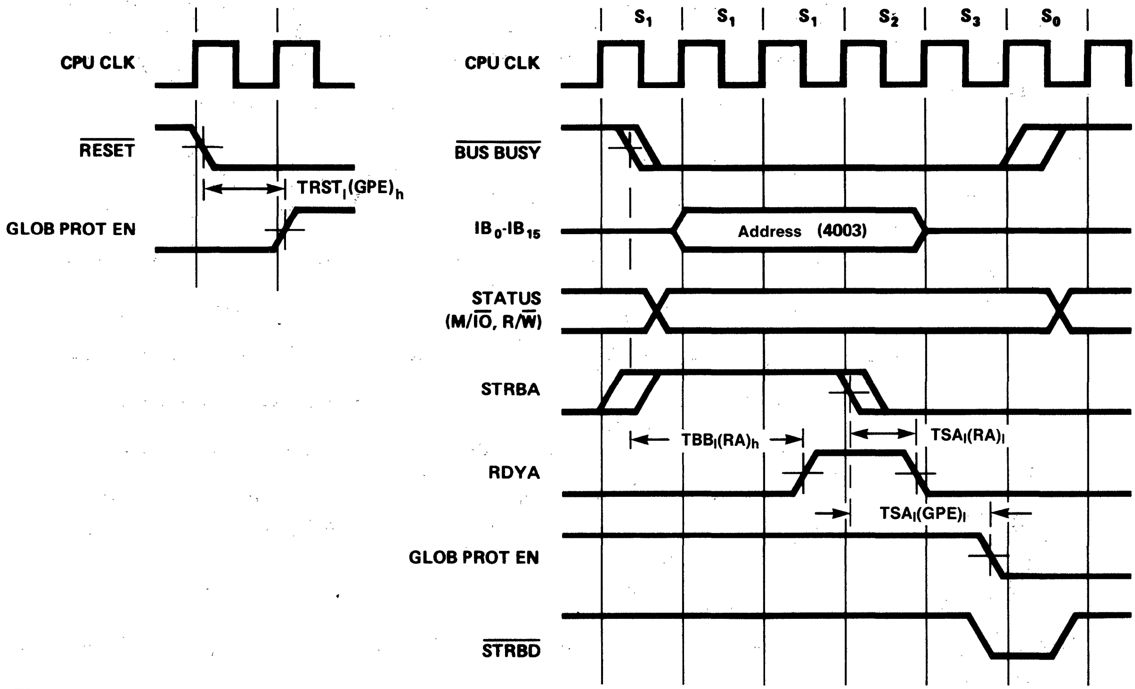


Figure 9.10 F9452 Timing for XIO MPEN

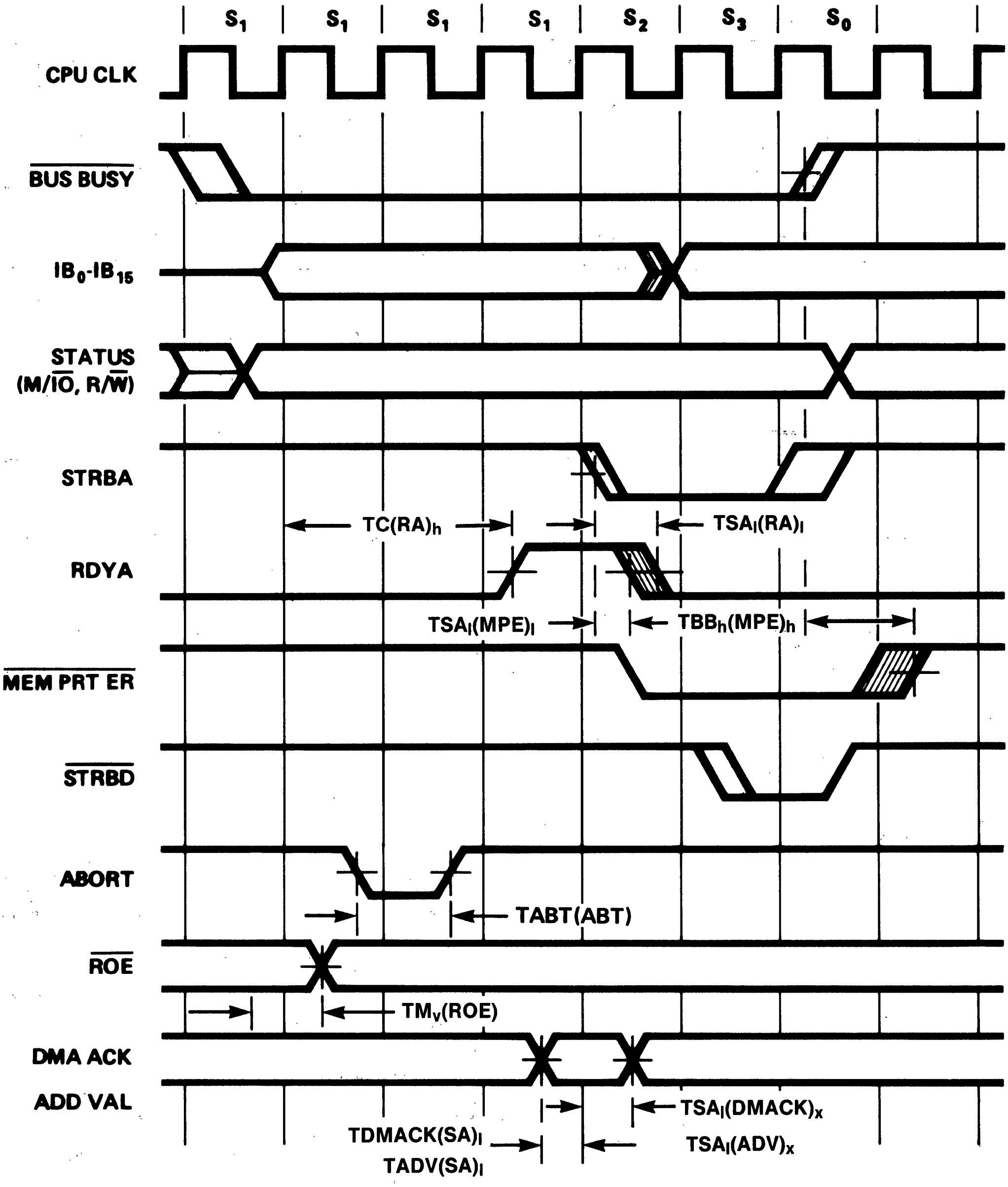


Figure 9.7 F9452 Memory Write Cycle Timing with Memory Protect Error and Internal Cache Miss

Table 9.5 F9452 DC Characteristics (Continued)

Symbol	Characteristic	Min.	Typ.	Max.	Units	Test Condition
I_{LOH}	O.C. Output Leakage $\overline{\text{MEM PRT ER}}$, RDYA			1.0	mA	$V_{OH} = V_{CC}$ V_{CC} = Min. I_{INJ} = Min.
I_{CC}	Power Supply Current		120		mA	V_{CC} = Max. I_{INJ} = Min.
V_{INJ}	Injector Voltage		1.2		V	V_{CC} = Max. I_{INJ} = Min.

* Not more than one output shorted at one time.

Figure 9.11 F9452 Timing for XIO Read Memory Protect (RMP)

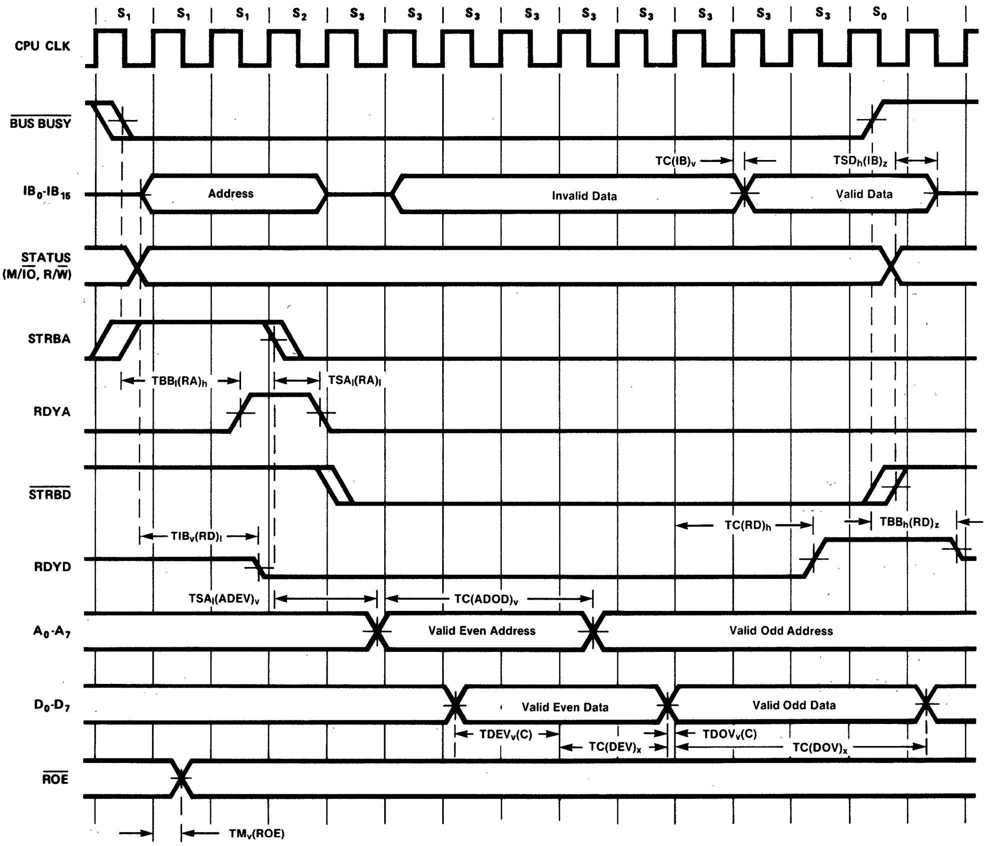


Table 9.5 F9452 DC Characteristics

Symbol	Characteristic	Min.	Typ.	Max.	Units	Test Condition
V_{IH}	Input High Level	2.0			V	Guaranteed Input High
V_{IL}	Input Low Level			0.8	V	Guaranteed Input Low
V_{CD}	Input Clamp Voltage		−0.9	−1.5	V	$I_{IN} = -18mA$ V_{CC} = Min. I_{INJ} = Min.
I_{IH}	Input High Current (Except D_0–D_7, IB_0–IB_{15})			40	μA	$V_{IN} = 2.7V$ V_{CC} = Max. I_{INJ} = Min.
I_{IH}	Input High Current (D_0–D_7,IB_0–IB_{15})			140	μA	$V_{IN} = 2.7V$ V_{CC} = Max. I_{INJ} = Min.
I_{IH}	Input High Current			1	mA	$V_{IN} = V_{CC}$ V_{CC} = Max. I_{INJ} = Min.
I_{IL}	Input Low Current		−200	−400	μA	$V_{IN} = 0.4V$ V_{CC} = Max. I_{INJ} = Min.
V_{OH}	Output High Voltage Except $\overline{\text{MEM PRT ER}}$, RDYA	2.4	3.2		V	$I_{OH} = -0.4mA$ V_{CC} = Min. I_{INJ} = Min.
V_{OL}	Output Low Voltage		0.2	0.5	V	$I_{OL} = 8.0mA$ V_{CC} = Min. I_{INJ} = Min.
I_{OSH}	Output Short-Circuit* Current	−15		−100	mA	$V_{OUT} = 0V$ V_{CC} = Max. I_{INJ} = Min.
I_{OZH}	Output 3-State Current RDYD, D_0–D_7,IB_0–IB_{15}			100	μA	$V_{OUT} = 2.4V$ V_{CC} = Max. I_{INJ} = Min.
I_{OZL}	Output 3-State Current D_0–D_7, IB_0–IB_{15}			−500	μA	$V_{OUT} = 0.5V$ V_{CC} = Max. I_{INJ} = Min.
I_{OZL}	Output 3-State Current RDYD			−100	μA	$V_{OUT} = 0.5V$ V_{CC} = Max. I_{INJ} = Min.

Package Information

The F9452 comes in two standard packages, one for DIP mounting, the other a surface-mount gull wing configuration. *Figures 9.14* and *9.15* furnish details of the standard packages, and Table 9.6 provides operating ranges for the case configurations.

The F9452 is available with 100 percent screening and quality conformance inspection required by MIL-STD-883.

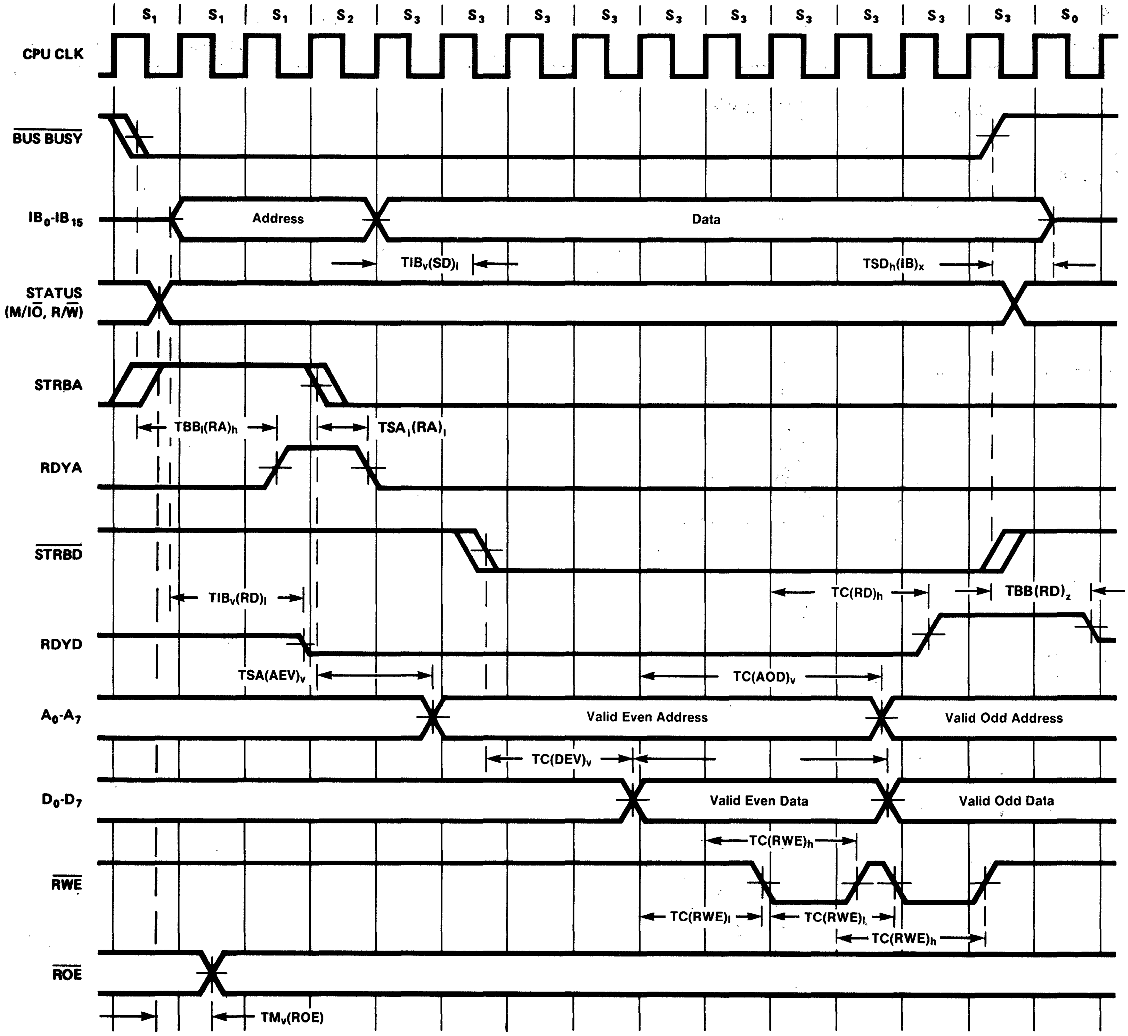
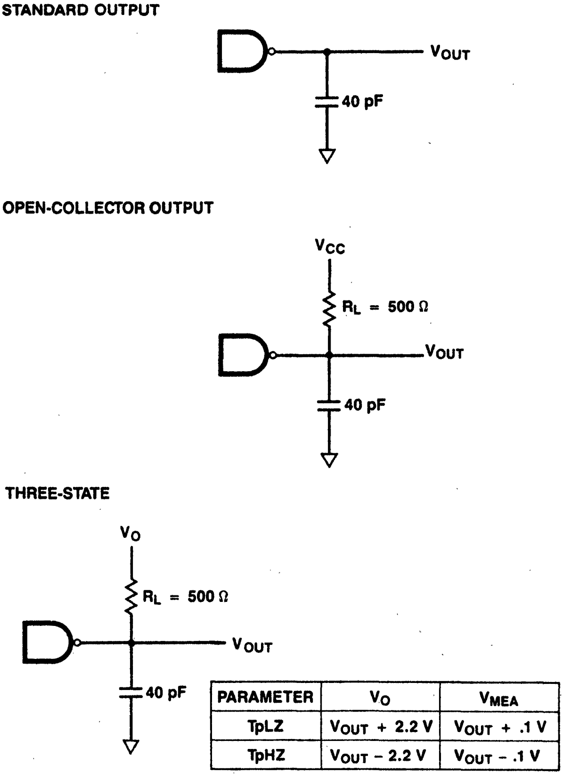


Figure 9.12 F9452 Timing for XIO Load Memory Protect (LMP)

Table 9.6 Case Operating Ranges

Order Code	Case Temperature (T_C) Operating Range (°C)	Package Type
F9452DC	0 to +85	Hermetic DIP
F9452GC	0 to +85	Hermetic Gull Wing Surface Mount
F9452DM	−55 to +125	Hermetic DIP
F9452GM	−55 to +125	Hermetic Gull Wing Surface Mount

Figure 9.13 Switching Time Test Circuits



PARAMETER	V_O	V_{MEA}
TpLZ	V_{OUT} + 2.2 V	V_{OUT} + .1 V
TpHZ	V_{OUT} − 2.2 V	V_{OUT} − .1 V

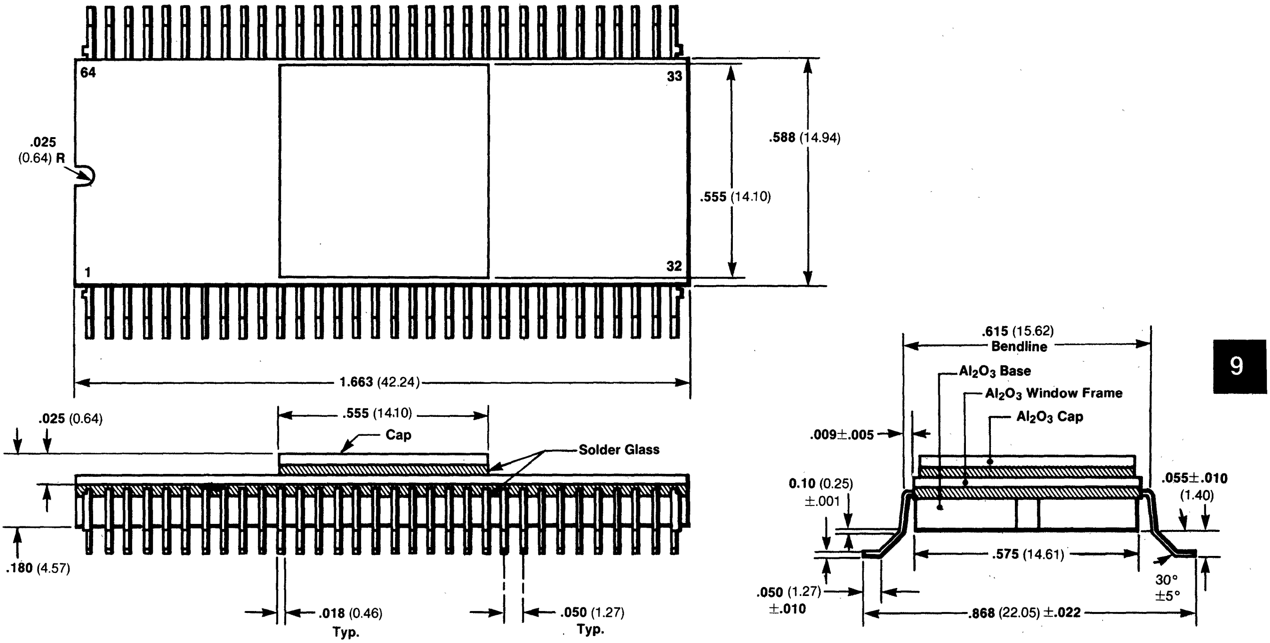


Figure 9.14 DIP-Mount Package

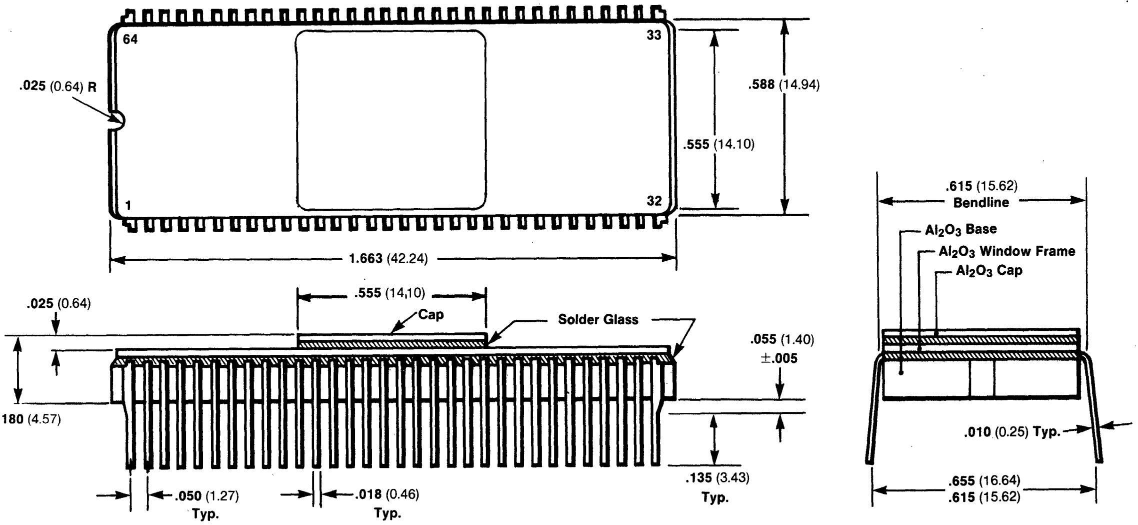


Figure 9.15 Gull-Wing Package

FAIRCHILD

A Schlumberger Company

F9450 Development Tools

An important factor in the success of any processor, including the F9450 family, is the availability of tools for designing the device in products. Recognizing this, Fairchild provides high-quality development tools to support the F9450. This chapter describes the tools available from Fairchild as of late 1985.

Fairchild continues to investigate support products useful to F9450 customers. For example, the introduction of a DoD-validated Ada compiler for the F9450 is contemplated for 1986; other programming languages are also under consideration.

Because MIL-STD-1750A is a government-mandated technology, numerous vendors have added to the government's offerings of 1750A development tools applicable to the F9450. This chapter makes no attempt to give an exhaustive list of those tools or to represent them as to quality and applicability. It does, however, identify sources of which Fairchild is now aware.

SBC-50 Single Board Computer

The Fairchild SBC-50 is a single board computer based on the F9450 microprocessor, designed to function as an evaluation and prototyping tool. The SBC-50 operates either as a serial-port attachment to a general-purpose host computer such as an IBM PC or a DEC VAX, or as a stand-alone system to which the user connects a terminal. The basic SBC-50 includes an F9450 CPU, 64K words of RAM, EPROM-resident systems software, and dual RS232 asynchronous ports capable of operating at up to 19.2K bps. External interfaces provide for memory and I/O expansion, as well as the addition of an optional F9451 MMU and/or F9452 BPU. The SBC-50 system bus is compatible with IEEE 796 (Multibus).

The systems software furnished on-board includes two EPROM-based programs. The first, SBC50TEST, is a self-test program that reports on the operating status of the board via a single-digit hex display included with the board, and also by means of signals sent to the host through one of the serial ports. This diagnostic program runs automatically when the SBC-50's manual RESET button is operated, and on command from the host.

The other resident program is SBC-50 MONITOR, which lets the user interact with the board. It communicates with the host or terminal in downloading user-written programs to the SBC-50, running them, setting breakpoints, stepping and tracing through machine code, examining registers and memory locations, and other debugging tasks.

These resident programs occupy two of the SBC-50's four 28-pin EPROM sockets. The other two sockets are available for insertion of user-developed firmware.

The SBC-50 has two on-board manual switches. One mentioned earlier is a RESET, which causes the SBC-50 to terminate the current activity, reinitialize, and perform self-test. The other switch, when depressed, generates a power-down interrupt to the F9450.

Six connectors allow for customization of the SBC-50 through attachment of external devices. These connectors and their purposes are as follows:

- P1: An 86-pin edge connector for IEEE 796 bus signals.
- P2: A 60-pin edge connector for extended Multibus signals.
- J1: A 60-pin header connector for interfacing with the F9451 MMU and/or F9452 BPU (available on a separate option board).
- J2: A 44-pin header connector for the iSBX bus, providing I/O expansion through multimodule plug-on boards.
- J3: A 26-pin header connector for two RS-232 serial ports.
- J4: A 60-pin header connector for F9450 Native Bus signals, through which external high-speed devices such coprocessors, other SBC-50's, etc., can gain direct access to the system bus.

The SBC-50's high-speed on-board RAM can be addressed at any clock speed of which the F9450 is capable, without wait states. The SBC-50 CPU can address on-board memory, memory on the Native Bus, and memory modules connected with the Multibus. During a memory operation initiated either by the F9450 or by a Native Bus master, a 20-bit physical memory address is developed on the SBC-50. This address consists of an 8-bit block address and a 12-bit word address. The block address selects a 4K word page, and the word address defines the specific 16-bit word within that block.

By assembling components around an SBC-50 board, the user can readily simulate proposed designs for testing and software development.

Operating conditions:

Power:	+ 5V DC ± 5% @ 11.0A max. + 12V DC ± 5% @ 25mA max. − 12V DC ± 5% @ 25mA max.
Clock frequency:	DC to maximum clock speed for the F9450 over the specified temperature and supply range.
Temperature:	20°C to 55°C.
Humidity:	0 to 90% non-condensing.

SBC-50 dimensions:

Height:	10.30 inches (26.3 cm)
Width:	12.00 inches (30.5 cm)
Depth:	00.65 inches (1.9 cm)
Weight:	1 lb., 4 oz.

MACRO-50 Macro Assembler

Fairchild's MACRO-50 is an assembly-language development system for the F9450. It supports all 1750A mnemonics including the optional instructions implemented in the F9450 family. A cross-assembler, it runs on the DEC VAX/VMS Version 3.2 or later. An IBM PC version is scheduled for release early in 1986.

The assembler possesses full macro definition and expansion capabilities, and allows named operations with parameter-passing in the manner of high-level languages. Conditional assembly is supported, and the program assembles subprogram modules into object form for later inclusion in libraries.

MACRO-50 furnishes a rich set of pseudo-ops for memory allocation and initialization, listing control, symbol table manipulation, interprogram communications, and repetitive or conditional assembly. In addition, programmers can define operation codes and new instruction mnemonics.

MACRO-50 comes with the RELOAD-50 linker and several utilities. RELOAD-50 accommodates both absolute and relocatable code, linkage of separate subprogram modules, and library searches. The LIBEDIT utility program manages libraries through simple interactive commands. Two object code convertors enable programmers to change MACRO-50 object modules to and from the .LDM format compatible with MSS, the government-supplied 1750A simulator/debugger.

After assembling and linking a program with MACRO-50, the executable result can be downloaded to the target F9450 system or "burned" onto a PROM destined for the target system.

Fairchild C Compiler

The Fairchild C Compiler is a full implementation of the Kernighan and Ritchie standard, hosted on the VAX/VMS Version 3.2 or later. The C language generates optimized code comparable in execution speed with assembly language code, while greatly increasing programmer productivity through the power of a high-level language.

The compiler accepts source code written in C and generates a file of optimized F9450 assembly language. This output flows into the MACRO-50 Macro Assembler (included with the C Compiler system) for assembly and linkage. The executable module can then be downloaded to a target F9450 machine or written to a PROM. The C programmer controls whether the code is relocatable or absolute.

The Fairchild C Compiler supports all the standard data types including bit fields. Types float and double correspond to the native F9450 single-precision and extended-precision floating point, and all floating-point operations are carried out using F9450 on-chip capabilities. Although the basic 1750A data element is a 16-bit word, the Fairchild C compiler packs character arrays with two bytes per word to conserve memory and builds in appropriate routines to extract the desired upper or lower word half when indexing the array. The compiler permanently reserves six F9450 general registers for register variables, thereby ensuring highly efficient loop control.

The C Compiler encourages modular programming and the accumulation of subprogram libraries. Included are macros for interfacing modules written in C and in

F9450 assembly language, as well as all the MACRO-50 utilities.

Other Sources

As a service to readers, Fairchild lists the following companies as additional sources of development tools known as of this writing. No representation is made or implied concerning the 1750A products they sell. If any 1750A vendors are omitted here, we apologize; the omission is not intentional.

The most authoritative source of information concerning 1750A-related products and vendors is the Defense Department's Language Control Facility, operated in cooperation with Softech, Inc. The LCF sponsors a 1750A Users' Group that holds periodic meetings and publishes a bulletin. The LCF also makes public-domain software tools and other design aids available for a nominal fee. Their address is:

Language Control Facility
ASD/ADOL
Wright-Patterson AFB, Ohio 45433

Following is a list of suppliers known to Fairchild and organized by product categories. Contact information and updates to the list can be obtained from the LCF.

Hardware development systems:	Tektronix Hewlett-Packard
Operating systems:	Hunter and Ready Proprietary Software Systems
JOVIAL:	LCF Software Engineering Associates Proprietary Software Systems Softech Advanced Computer Techniques
FORTRAN:	Advanced Computer Techniques
Assemblers:	Fairchild LCF Advanced Computer Techniques Proprietary Software Systems Intermetrics McDonnell-Douglas
C compiler:	Fairchild
Ada:	Boeing Softech Westinghouse
Simulators/debuggers:	LCF McDonnell-Douglas Proprietary Software Systems

FAIRCHILD
A Schlumberger Company

Appendix A
F9450 Instruction Set

Introduction

This appendix describes each instruction implemented on the Fairchild F9450 microprocessor. The F9450 instruction set fully conforms to MIL-STD-1750A notice 1, *MILITARY STANDARD SIXTEEN BIT COMPUTER INSTRUCTION SET ARCHITECTURE*, 21 May 1982.

Fairchild's MACRO-50 Assembler utilizes the 1750A mnemonics. For those readers employing an assembler that utilizes IEEE mnemonics, the descriptions give the corresponding IEEE instruction. In addition, Appendix B furnishes a table for translating from IEEE to 1750A.

Mnemonics as Related to Addressing Modes

The 1750A instruction set is highly complex, non-orthogonal, and especially suited for real-time control applications. In general, the following convention is used to "build up" mnemonics for applicable addressing modes (where 'm' is the basic memory-direct mnemonic):

Mnemonic	Addressing mode
m	Memory direct
mX	Memory direct indexed
mI	Memory indirect
mIX	Memory indirect indexed
mIM	Immediate long
mIMX	Immediate long indexed
mISP	Immediate short positive
mISN	Immediate short negative
mICR	Instruction-counter relative
mB	Base relative
mBX	Base relative indexed
mR	Register direct
mS	Special

Register Transfer Notation

In the instruction descriptions that comprise the bulk of this appendix, the following abbreviations and symbols are used to define the applicable register transfer language:

CPU Registers

R0, R1, ..., R15	The sixteen 16-bit general registers
IC	Instruction Counter Register
SW	Status Word Register
CS	Condition Status; the flags field within the F9450 SW register
LP	Linkage Pointer
SP	Stack Pointer; implicitly R15 for the PSHM (Push Multiple) and POPM (Pop Multiple) instructions.
SVP	Service Pointer
MK	Interrupt Mask Register
PI	Pending Interrupt Register
RA, RB	An unspecified general register

Data Quantities

MSH, LSH	Most Significant Half, Least Significant Half (bytes within a word)
MSB, LSB	Most Significant Bit, Lease Significant Bit
S.P., D.P.	Single Precision, Double Precision
Ft.P., E.F.P.	Floating Point, Extended Floating Point
MO	Floating point derived operand mantissa (fractional part)
EO	Floating point eight-bit 2's complement derived operand characteristic (exponent)
MA	Floating point register accumulator mantissa
EA	Floating point register accumulator exponent
RQ, MP, MQ, RR	Entities used for register level transfer description clarification. These registers are not part of the general register set. Q = quotient, P = product, R = remainder.

Miscellaneous

(X)	Contents of Register X
(X,X + 1)	Contents of concatenated Registers X and X + 1

Appendix A
F9450 Instruction Set

[X]	Contents of memory address X
[X,X + 1]	Contents of sequential memory locations X and X + 1
OVM	Floating point mantissa overflow
Exit	Termination of the present register transfer operation, except for setting of the CS bits
DA	Derived Address
DO	Derived Operand
N, M, n	An integer number
DSPL	Displacement
X_n	If X is a CPU register or a data quantity, then n specifies a bit position in X. If X is not a CPU register or a data quantity, the number X is to the base of n. If X is a number and n = 16, then X is a 2's complement hexadecimal number.
X^i	If X is a CPU register or a memory address, then i specifies the state of X. This notation is used in the register transfer descriptions to refer to the contents of a CPU register or a memory address at different times (states) of the execution of the instruction. If X is not a CPU register or a memory address, the number X is raised to the ith power.

Symbols

←	Unilateral transfer
→	Bilateral transfer
:	Comparison
x	Indicates a "don't care" bit when used in a binary number
>	Greater than
<	Less than
≥	Greater than or equal to
≤	Less than or equal to
↑	Logical AND
v	Logical OR
⊕	Exclusive OR
–	Logical NOT
|X|	Absolute value of X

Operation Codes

Tables A-1 and A-2 are matrices showing the F9450 op-codes and extended op-codes, respectively.

Instruction Set

The balance of this appendix decribes in detail all the instructions implemented on the F9450. At the end of the appendix is an index to the instruction set.

Appendix A
F9450 Instruction Set

Table A-1 F9450 Operation Code Matrix

	Load Store	Integer Arithmetic	Floating Point	Logical Compare	Opcode Extensions	Bit	Shift	Jump	Load	Store	Add	Sub	Mult	Divide	Logical	Compare
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	LB BR12	AB BR12	FAB BR12	ORB BR12	BRX BR12*	SB	SLL	JC	L	ST	A	S	MS	DV	OR	C
1	LB BR13	AB BR13	FAB BR13	ORB BR13	BRX BR13*	SBR	SRL	JCI	LR	STC	AR	SR	MSR	DVR	ORR	CR
2	LB BR14	AB BR14	FAB BR14	ORB BR14	BRX BR14*	SBI	SRA	JS	LISP	STCI	AISP	SISP	MISP	DISP	AND	CISP
3	LB BR15	AB BR15	FAB BR15	ORB BR15	BRX BR15*	RB	SLC	SOJ	LISN	MOV	INCM	DECM	MISN	DISN	ANDR	CISN
4	DLB BR12	SBB BR12	FSB BR12	ANDBBR12		RBR		BR	LI	STI	ABS	NEG	M	D	XOR	CBL
5	DLB BR13	SBB BR13	FSB BR13	ANDBBR13		RBI	DSLL	BEZ	LIM		DABS	DNEG	MR	DR	XORR	
6	DLB BR14	SBB BR14	FSB BR14	ANDBBR14		TB	DSRL	BLT	DL	DST	DA	DS	DM	DD	N	DC
7	DLB BR15	SBB BR15	FSB BR15	ANDBBR15		TBR	DSRA	BEX	DLR	SRM	DAR	DSR	DMR	DDR	NR	DCR
8	STB BR12	MB BR12	FMB BR12	CB BR12	XIO*#	TBI	DSLC	BLE	DLI	DSTI	FA	FS	FM	FD	FIX	FC
9	STB BR13	MB BR13	FMB BR13	CB BR13	VIO*#	TSB		BGT	LM	STM	FAR	FSR	FMR	FDR	FLT	FCR
A	STB BR14	MB BR14	FMB BR14	CB BR14	IMML*#	SVBR	SLR	BNZ	EFL	EFST	EFA	EFS	EFM	EFD	EFIX	EFC
B	STB BR15	MB BR15	FMB BR15	CB BR15			SAR	BGE	LUB	STUB	EFAR	EFSR	EFMR	EFDR	EFLT	EFCR
C	DSTB BR12	DB BR12	FDB BR12	FCB BR12		RVBR	SCR	LSTI#	LLB	STLB	FABS	FNEG			XBR	
D	DSTB BR13	DB BR13	FDB BR13	FCB BR13			DSLR	LST#	LUBI	SUBI					XWR	
E	DSTB BR14	DB BR14	FDB BR14	FCB BR14		TVBR	DSAR	SJS	LLBI	SLBI						
F	DSTB BR15	DB BR15	FDB BR15	FCB BR15	BIF**		DSCR	URS	POPM	PSHM						NOP/ BPT

*These order types represent instructions which have "extended" operation codes and are fully described in the instruction specifications and in Table A-2.
#Privileged instructions
**User Defined Built-In Function Opcode.

Appendix A
F9450 Instruction Set

Table A-2 Extended Operation Codes
Opcode Extension (see below for location)

* MSH	** Format	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
40	BRX BR12	LBX	DLBX	STBX	DSTX	ABX	SBBX	MBX	DBX	FABX	FSBX	FMBX	FDBX	CBX	FCBX	ANDX	ORBX
41	BRX BR13	LBX	DLBX	STBX	DSTX	ABX	SBBX	MBX	DBX	FABX	FSBX	FMBX	FDBX	CBX	FCBX	ANDX	ORBX
42	BRX BR14	LBX	DLBX	STBX	DSTX	ABX	SBBX	MBX	DBX	FABX	FSBX	FMBX	FDBX	CBX	FCBX	ANDX	ORBX
43	BRX BR15	LBX	DLBX	STBX	DSTX	ABX	SBBX	MBX	DBX	FABX	FSBX	FMBX	FDBX	CBX	FCBX	ANDX	ORBX
4A	IMML		AIM	SIM	MIM	MSIM	DIM	DVIM	ANDM	ORIM	XORM	CIM	NIM				

*MSH (Most Significant Half)

**Base Relative Indexed Format

← MSH →			
6	2	4	4
Opcode	BR	Op. Ex.	RX

**Immediate Opcode Extension Format

8	4	4		16
Opcode	RA	Op. Ed.		Operand

These Extended Operation Codes do not include those for Built-In Function (BIF). The Extended Operation Codes for these instructions are within each instruction's definition.

Notice

Timing characteristics for the devices discussed in this book are typical and should not be regarded as absolute inasmuch as revision levels of the products differ slightly in their timing characteristics. For exact timing information, see the current product Data Sheet, or contact the Fairchild Microcontroller Division, your Fairchild Product Specialist, or Sales Office.

Integer Arithmetic/Logic

A, AB, ABX, AIM, AISP, AR

Single Precision Integer Add

DESCRIPTION: THE DERIVED OPERAND (DO) is added to the contents of the RA register. The result (a 2's complement sum) is stored in register RA. The Condition Status (CS) is set based on the result in register RA and carry. A fixed point overflow occurs if both operands are of the same sign and the sum is of opposite sign.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode		Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
A (ADD	RA,ADDR addr,Ra)	D	8 · 4 · 4 A0 · RA · RX	16 ADDR	13,02,01
A (ADD	RA,ADDR,RX addr(Rx),Ra)	DX			13,02,01
AB (ADD	BR,DSPL addr(Rx),Ra)	B	4 · 2 · 2 · 8 1 · 0 · BR′ · DSPL	12 ≤ BR ≤ 15 BR′ = BR − 12 RA = R2	12,01,01
ABX (ADD	BR,RX dsp1(Br),R2)	BX	4 · 2 · 2 · 4 · 4 4 · 0 · BR′ · 4 · RX	12 ≤ BR ≤ 15 BR′ = BR − 12 RA = R2	12,01,01
AIM (ADD	RA,DATA #data,Ra)	IM	8 · 4 · 4 4A · RA · 1	16 DATA	12,02,00
AISP (ADD4	RA,N #data,Ra)	ISP	8 · 4 · 4 A2 · RA · N−1	1 ≤ N ≤ 16	08,01,00
AR (ADD	RA,RB Rb,Ra)	R	8 · 4 · 4 A1 · RA · RB		05,01,00

Register Transfer Description:

$(RA)^2 \leftarrow (RA)^1 + DO;$

$PI_4 \leftarrow 1$, IF $(RA_0)^1 = DO_0$ and $(RA_0)^1 \neq (RA_0)^2$

$(CS) \leftarrow 0010$ if carry $= 0$ and $(RA) = 0;$
$(CS) \leftarrow 0001$ if carry $= 0$ and $(RA) < 0;$
$(CS) \leftarrow 0100$ if carry $= 0$ and $(RA) > 0;$
$(CS) \leftarrow 1010$ if carry $= 1$ and $(RA) = 0;$
$(CS) \leftarrow 1001$ if carry $= 1$ and $(RA) < 0;$
$(CS) \leftarrow 1100$ if carry $= 1$ and $(RA) > 0;$

Registers Affected:
RA, CS, PI

DA, DAR

Double Precision Integer Add

DESCRIPTION: The double precision Derived Operand (DO) is added to the contents of registers RA and RA+1. The result (a 2's complement 32-bit sum) is stored in registers RA and RA+1. The MSH is in RA. The condition status (CS) is set based on the double precision results in RA and RA+1, and carry. A fixed point overflow occurs if both operands are of the same sign and the sum is of opposite sign.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
			8	4	4	16	
DA (ADDL	RA,ADDR addr,Ra)	D	A6	RA	RX	ADDR	24,02,01
DA (ADDL	RA,ADDR,RX addr(Rx),Ra)	DX					24,02,01
			8	4	4		
DAR (ADDL	RA,RB Rb,Ra)	R	A7	RA	RB		18,01,00

Register Transfer Description:

$(RA,RA+1)^2 \leftarrow (RA,RA+1)^1 + DO;$

$PI_4 \leftarrow 1$, if $(RA_0)^1 = DO_0$ and $(RA_0)^1 \neq (RA_0)^2$

$(CS) \leftarrow 0010$ if carry $= 0$ and $(RA,RA+1) = 0;$
$(CS) \leftarrow 0001$ if carry $= 0$ and $(RA,RA+1) < 0;$
$(CS) \leftarrow 0100$ if carry $= 0$ and $(RA,RA+1) > 0;$
$(CS) \leftarrow 1010$ if carry $= 1$ and $(RA,RA+1) = 0;$
$(CS) \leftarrow 1001$ if carry $= 1$ and $(RA,RA+1) < 0;$
$(CS) \leftarrow 1100$ if carry $= 1$ and $(RA,RA+1) > 0;$

Registers Affected:

RA, RA+1, CS, PI

Single Precision Integer Subtract

DESCRIPTION: The Derived Operand (DO) is subtracted from the contents of the RA register. The result, a 2's complement difference,is stored in RA. The condition status (CS) is set based on the result in register RA and carry. A fixed point overflow occurs if both operands are of opposite signs and the derived operand is the same as the sign of the difference.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode		Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
S (SUB	RA,ADDR addr,Ra)	D	8 4 4 B0 RA RX	16 ADDR	13,02,01
S (SUB	RA,ADDR,RX addr(Rx),Ra)	DX			13,02,01
SBB (SUB	BR,DSPL dspl(Br),R2)	B	4 2 2 8 1 1 BR′ DSPL	$12 \leq BR \leq 15$ BR′ = BR − 12 RA = R2	12,01,01
SBBX (SUB	BR,RX (Rx)(Br),R2)	BX	4 2 2 4 4 4 0 BR′ 5 RX	$12 \leq BR \leq 15$ BR′ = BR − 12 RA = R2	12,01,01
SIM (SUB	RA,DATA #data,Ra)	IM	8 4 4 4A RA 2	16 DATA	12,02,00
SISP (SUB4	RA,N #data,Ra)	ISP	8 4 4 B2 RA N−1	$1 \leq N \leq 16$	08,01,00
SR (ADD	RA,RB Rb,Ra)	R	8 4 4 B1 RA RB		05,01,00

A2

Register Transfer Description:
$(RA)^2 \leftarrow (RA)^1 - DO$, i.e., $(RA) - DO$ means $\{(RA) + DO\} + 1$;

$PI_4 \leftarrow 1$, if $(RA_0)^1 \neq DO_0$ AND $(RA_0)^2 = DO_0$

(CS) ← 0010 if carry = 0 and (RA) = 0;
(CS) ← 0001 if carry = 0 and (RA) < 0;
(CS) ← 0100 if carry = 0 and (RA) > 0;
(CS) ← 1010 if carry = 1 and (RA) = 0;
(CS) ← 1001 if carry = 1 and (RA) < 0;
(CS) ← 1100 if carry = 1 and (RA) > 0;

Registers Affected:
RA, CS, PI

Integer Arithmetic/Logic

DS, DSR

Double Precision Integer Subtract

DESCRIPTION: The double precision Derived Operand, DO, is subtracted from the contents of registers RA and RA+1. The result, a 2's complement 32-bit difference, is stored in registers RA and RA+1. The MSH is RA. The condition status (CS) is set based on the double precision results in RA and RA+1, and carry. A fixed point overflow occurs if both operands are of opposite sign and the derived operand is the same as the sign of the difference.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Fetch Bus Cycles
DS (SUBL	RA,ADDR addr,Ra)	D	8	4	4	16	24,02,02
DS (SUBL	RA,ADDR,RX addr(Rx),Ra)	DX	B6	RA	RX	ADDR	24,02,02
			8	4	4		
DSR (SUBL	RA,RB Rb,Ra)	R	B7	RA	RB		18,01,00

Register Transfer Description:

$(RA,RA,+1)^2 \leftarrow (RA,RA+1)^1 - DO$, i.e., $(RA,RA+1) - DO$ means $\{(RA,RA+1) + \overline{DO}\} + 1$;

$PI_4 \leftarrow 1$, if $(RA_0)^1 \neq DO_0$ and $(RA_0)^2 = DO_0$;

$(CS) \leftarrow 0010$ if carry $= 0$ and $(RA,RA+1) = 0$;
$(CS) \leftarrow 0001$ if carry $= 0$ and $(RA,RA+1) < 0$;
$(CS) \leftarrow 0100$ if carry $= 0$ and $(RA,RA+1) > 0$;
$(CS) \leftarrow 1010$ if carry $= 1$ and $(RA,RA+1) = 0$;
$(CS) \leftarrow 1001$ if carry $= 1$ and $(RA,RA+1) < 0$;
$(CS) \leftarrow 1100$ if carry $= 1$ and $(RA,RA+1) > 0$;

Registers Affected:

RA, RA+1, CS, PI

NEG

Single Precision Negate Register

DESCRIPTION: The negative (i.e., the 2's complement) of the Derived Operand, DO (i.e., the contents of register RB), is stored into register RA. The condition status, CS, is set based on the result in register RA.

Note: The negative of zero is zero.

The negative of a number with a 1 in the sign bit and all other bits zero is the same word, and causes fixed point overflow to occur.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode			Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	
NEG RA,RB (NEG Rb,Ra)	R	B4	RA	RB	05,01,00

Register Transfer Description:

$(RA) \leftarrow -DO$;

$PI_4 \leftarrow 1$, exit, if $DO = 8000_{16}$;

$(CS) \leftarrow 0010$ if $(RA) = 0$;
$(CS) \leftarrow 0001$ if $(RA) < 0$;
$(CS) \leftarrow 0100$ if $(RA) > 0$;

Registers Affected:
RA, CS, PI

DNEG

Double Precision Negate Register

DESCRIPTION: The negative (i.e., the 2's complement) of the Derived Operand, DO (i.e., the contents of register RB and RB+1), is stored into register RA and RA+1 such that register RA contains the MSH of the result. The condition status, CS, is set based on the result in register RA and RA+1.

Note: The negative of zero is zero.

The negative of a number with a 1 in the sign bit and all other bits zero is the same word, and causes fixed point overflow to occur.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode			Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	
DNEG RA,RB (NEGL Rb,Ra)	R	B5	RA	RB	18,01,00

Register Transfer Description:

$(RA,RA+1) \leftarrow -\ DO$;

$PI_4 \leftarrow 1$, exit, if $DO = 8000\ 0000_{16}$;

$(CS) \leftarrow 0010$ if $(RA,RA+1) = 0$;
$(CS) \leftarrow 0001$ if $(RA,RA+1) < 0$;
$(CS) \leftarrow 0100$ if $(RA,RA+1) > 0$;

Registers Affected:

RA, RA+1, CS, PI

DM, DMR

Double Precision Integer Multiply

DESCRIPTION: The double precision Derived Operand, DO, a 32-bit 2's complement number, is multiplied by the contents of registers RA and RA+1, a 32-bit 2's complement number, with the MSH in RA. The LSH of the product is retained in RA and RA+1 as a 32-bit 2's complement number. The MSH is lost. The Condition Status, CS, is set based on the double precision result in registers RA and RA+1. A fixed point overflow occurs if (1) both operands are of the same sign and the MSH of the product is not zero, or the sign bit of the LSH is not zero, or (2) if the operands are of opposite sign and the MSH of the product is not FFFF $FFFF_{16}$, or the sign bit of the LSH is not one. A fixed point overflow does not occur if either of the operands is zero.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	16	
DM RA,ADDR (MULLS addr,Ra)	D	C6	RA	RX	ADDR	132,02,02
DM RA,ADDR,RX (MULLS addr(Rx),Ra)	DX					132,02,02
		8	4	4		
DMR RA,RB (MULLS Rb,Ra)	R	C7	RA	RB		126,01,00

Register Transfer Description:

$(RQ,RQ+1,RQ+2,RQ+3) \leftarrow (RA,RA+1)^1 \times DO;$

$(RA,RA,+1)^2 \leftarrow (RQ+2,RQ+3);$

$PI_4 \leftarrow 1$, if $\{(RA_0)^1 = DO_0$ and $\{(RQ,RQ+1) \neq 0$ or $(RQ+2_0) = 1\}\}$ or
$\{(RA_0)^1 \neq DO_0$ and $\{(RQ,RQ+1) \neq$ FFFF $FFFF_{16}$ or $(RQ+2_0) = 0\}$ and $\{(RA)^1 \neq 0$ and $DO \neq 0\}\};$

$(CS) \leftarrow 0010$ if $(RA,RA+1) = 0;$
$(CS) \leftarrow 0001$ if $(RA,RA+1) < 0;$
$(CS) \leftarrow 0100$ if $(RA,RA+1) > 0;$

Registers Affected:
RA, RA+1, CS, PI

Single Precision Integer Divide with 16-Bit Dividend

DESCRIPTION: The contents of register RA are divided by the Derived Operand, DO, a single precision, 2's complement number. The result is stored in registers RA and RA+1 such that RA stores the single precision integer quotient and RA+1 stores the remainder. The Condition Status, CS, is set based on the result in RA. A fixed point overflow occurs if the divisor, DO, is zero, or if the dividend is 8000_{16} and the divisor is $FFFF_{16}$.

Note: The sign of the non-zero remainder is the same as the sign of the dividend.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
			8	4	4		
DISN (DIV4S	RA,N #data,Ra)	ISN	D3	RA	N−1	1 ≤ N ≤ 16	98,01,00
			8	4	4		
DISP (DIV4S	RA,N #data,Ra)	ISP	D2	RA	N−1	1 ≤ N ≤ 16	98,01,00
			8	4	4	16	
DV (DIVS	RA,ADDR addr,Ra)	D	DO	RA	RX	ADDR	103,02,01
DV (DIVS	RA,ADDR,RX addr(Rx),Ra)	DX					103,02,01
			8	4	4	16	
DVIM (MULS	RA,DATA #data,Ra)	IM	4A	RA	6	DATA	102,02,00
			8	4	4		
DVR (DIVS	RA,RB Rb,Ra)	R	D1	RA	RB		98,01,00

Register Transfer Description:

$(RA,RA+1) \leftarrow (RA) / DO$;

$PI_4 \leftarrow 1$, if $DO = 0$ or $\{RA = 8000_{16}$ and $DO = FFFF_{16}\}$;

$(CS) \leftarrow 0010$ if $(RA) = 0$;
$(CS) \leftarrow 0001$ if $(RA) < 0$;
$(CS) \leftarrow 0100$ if $(RA) > 0$;

Registers Affected:

RA, RA+1, CS, PI

MISN, MISP, MS, MSIM, MSR

Single Precision Integer Multiply with 16-Bit Product

DESCRIPTION: The Derived Operand, DO, is multiplied by the contents of register RA. The LSH of the result, a 16-bit, 2's complement integer, is stored in register RA. The Condition Status, CS, is set based on the result in register RA. A fixed point overflow occurs if (1) both operands are of the same sign and the MSH of the product is not zero, or the sign bit of the LSH is not zero, or (2) if the operands are of opposite sign and the MSH of the product is not $FFFF_{16}$ or the sign bit of the LSH is not one. A fixed point overflow does not occur if either of the operands is zero.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4		
MISN RA,N (MUL4S #data,Ra	ISN	C3	RA	N−1	$1 \leq N \leq 16$	42,01,00
		8	4	4		
MISP RA,N (MUL4 #data,Ra)	ISP	C2	RA	N−1	$1 \leq N \leq 16$	42,01,00
		8	4	4	16	
MS RA,ADDR (MUL4S #data,Ra)	D	C0	RA	RX	ADDR	47,02,01
MS RA,ADDR,RX (MULS addr,(Rx),Ra)	DX					47,02,01
		8	4	4	16	
MSIM RA,DATA (MULS #data,Ra)	IM	4A	RA	4	DATA	46,02,00
		8	4	4		
MSR RA,RB (MULS Rb,Ra)	R	C1	RA	RB		39,01,00

Register Transfer Description:

$(RQ, RQ+1) \leftarrow (RA)^1 \times DO;$

$(RA)^2 \leftarrow (RQ+1);$

$PI_4 \leftarrow 1, \text{ if } \{(RA_0)^1 = DO_0 \text{ and } \{(RQ) \neq 0 \text{ OR } (RQ+1_0) = 1\}\} \text{ or }$
$\{(RA_0)^1 \neq DO_0 \text{ and } \{(RQ) \neq FFFF_{16} \text{ or } (RQ+1_0) = 0\} \text{ and }$
$\{(RA)^1 \neq 0 \text{ and } DO \neq 0\}\};$

$(CS) \leftarrow 0010 \text{ if } (RA) = 0;$
$(CS) \leftarrow 0001 \text{ if } (RA) < 0;$
$(CS) \leftarrow 0100 \text{ if } (RA) > 0;$

Registers Affected:
D, CS, PI

Integer Arithmetic/Logic

M, MB, MBX, MIM, MR

Single Precision Integer Multiply with 32-Bit Product

DESCRIPTION: The Derived Operand, DO, is multiplied by the contents of register RA. The result, a 32-bit, 2's complement integer, is stored in registers RA and RA+1 with the MSH of the product in register RA. The Condition Status, CS, is set based on the result in registers RA and RA+1.

SPECIAL CASE: DO = (RA) = 8000 (the largest negative number), then DO × (RA) = 4000 0000.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode						Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
M (MUL	RA,ADDRR addr,Ra)	D	8 C4	4 RA	4 RX			16 ADDR	45,02,01
M (MUL	RA,ADDR,RX addr(Rx),Ra)	DX							45,02,01
MB (MUL	BR,DSPL dsp1(Br),R2)	B	4 1	2 2	2 BR′	8 DSPL		12≤ BR≤ 15 BR′ = BR − 12 RA = R2	44,01,01
MBX (MUL	BR,RX (Rx)(Br),R2)	BX	4 4	2 0	2 BR′	4 6	4 RX	12≤ N≤ 15 BR′ = BR − 12 RA = R2	44,01,01
MIM (MUL	RA,DATA #data Ra)	IM	8 4A	4 RA	4 3			16 DATA	44,02,00
MR (MUL	RA,RB Rb,Ra)	R	8 C5	4 RA	4 RB				37,01,00

Register Transfer Description:
(RA,RA+1) ← (RA) × DO;

(CS) ← 0010 if (RA,RA+1) = 0;
(CS) ← 0001 if (RA,RA+1) < 0;
(CS) ← 0100 if (RA,RA+1) > 0;

Registers Affected:
RA, RA+1, CS

D, DB, DBX, DIM, DR

Single Precision Integer Divide with 32-Bit Dividend

DESCRIPTION: The contents of registers RA and RA+1, a double precision 2's complement number, are divided by the Derived Operand, DO, a single precision, 2's complement number. RA contains the MSH of the 32-bit dividend. The result is stored in registers RA and RA+1 such that RA stores the single precision integer quotient and RA+1 stores the remainder. The Condition Status, CS, is set based on the result in RA. A fixed point overflow occurs if the divisor equals zero or if a positive quotient exceeds $7FFF_{16}$ or a negative quotient is less than 8000_{16}.

Note: The sign of the non-zero remainder is the same as that of the dividend.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode		Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
D (DIV	RA,ADDR addr,Ra)	D	8 4 4 D4 | RA | RX	16 ADDR	102,02,01
D (DIV	RA,ADDR,RX addr(Rx),Ra)	DX			102,02,01
DB (DIV	BR,DSPL dsp1(Br),R2)	B	4 2 2 8 1 | 3 | BR′ | DSPL	$12 \leq BR \leq 15$ BR′ = BR − 12 RA = R2	101,01,01
DBX (DIV	BR,RX (Rx)(Br),R2)	BX	4 2 2 4 4 4 | 0 | BR′ | 7 | RX	$12 \leq BR \leq 15$ BR′ = BR − 12 RA = R2	101,01,01
DIM (DIV	RA,DATA #data,Ra)	IM	8 4 4 4A | RA | 5	16 DATA	101,02,00
DR (DIV	RA,RB Rb,Ra)	R	8 4 4 D5 | RA | RB		97,01,00

Register Transfer Description:

(RQ, RQ+1, RR) ← (RA, RA+1) / DO;

PI_4 ← 1, if DO = 0 or (RQ, RQ+1) > 0000 $7FFF_{16}$ or (RQ, RQ+1) < FFFF 8000_{16}

(RQ) ← (RQ+1)

(RA+1) ← (RR)

(CS) ← 0010 if (RA) = 0;
(CS) ← 0001 if (RA) < 0;
(CS) ← 0100 if (RA) > 0;

Registers Affected:
RA, RA+1, CS, PI

Double Precision Integer Divide

DESCRIPTION: The contents of registers RA and RA+1, a double precision 2's complement number, are divided by the Derived Operand, DO, a double precision 2's complement number. RA contains the MSH of the 32-bit dividend. The quotient part of the integer result is stored in registers RA and RA+1 (with MSH in RA) and the remainder is lost. The Condition Status, CS, is set based on the results in registers RA and RA+1. A fixed point overflow occurs if the divisor, DO, is zero, or if the dividend is 8000 0000_{16} and the divisor is FFFF $FFFF_{16}$.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	16	
DD RA,ADDRR (DIVLS addr,Ra)	D	D6	RA	RX	ADDR	245,02,02
DD RA,ADDR,RX (DIVLS addr(Rx),Ra)	DX					245,02,02
		8	4	4		
DDR RA,RB (DIVLS Rb,Ra)	R	D7	RA	RB		239,01,00

Register Transfer Description:
(RA,RA+1) ← (RA, RA+1) / DO;

PI_4 ← 1, if DO = 0 or {RA, RA+1 = 8000 0000_{16} and DO = FFFF $FFFF_{16}$};

(CS) ← 0010 if (RA,RA+1) = 0;
(CS) ← 0001 if (RA,RA+1) < 0;
(CS) ← 0100 if (RA,RA+1) > 0;

Registers Affected:
RA, RA+1, CS, PI

INCM

Increment Memory by a Positive Integer

DESCRIPTION: The contents of the memory location specified by the Derived Address, DA, is incremented by N, where N is an integer, $1 \leq N \leq 16$. This instruction adds a positive constant to memory. The condition status, CS, is set based on the results of the addition and carry. A fixed point overflow occurs if the operand in memory is positive and the result is negative. The memory location specified is updated to contain the result of the addition process even if a fixed point overflow occurs.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	16	
INCM N, ADDR (ADD4 #data,addr)	D	A3	N−1	RX	ADDR	17,02,02
INCM N,ADDR,RX (ADD4 #data,addr(Rx))	DX					17,02,02

Register Transfer Description:

$[DA]^2 \leftarrow [DA]^1 + N$, where $1 < N < 16$;

$PI_4 \leftarrow 1$, if $[DA]^2 < 0 < [DA]^1$;

$(CS) \leftarrow 0010$ if carry $= 0$ and $[DA] = 0$;
$(CS) \leftarrow 0001$ if carry $= 0$ and $[DA] < 0$;
$(CS) \leftarrow 0100$ if carry $= 0$ and $[DA] > 0$;
$(CS) \leftarrow 1010$ if carry $= 1$ and $[DA] = 0$;
$(CS) \leftarrow 1001$ if carry $= 1$ and $[DA] < 0$;
$(CS) \leftarrow 1100$ if carry $= 1$ and $[DA] > 0$;

Registers Affected:
CS, PI

DECM

Decrement Memory by a Positive Integer

DESCRIPTION: The contents of the memory location specified by the Derived Address, DA, are decremented by N, where N is an integer, 1≤N≤16. This is the equivalent of a "subtract-from-memory instruction." The condition status, CS, is set based on the results of the subtraction and carry. A fixed point overflow occurs if the operand in memory is negative and the result is positive. The memory location specified is updated to contain the result of the subtraction process even if a fixed point overflow occurs.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	16	
DECM N,ADDR (SUB4 #data,addr)	D	B3	N−1	RX	ADDR	17,02,02
DECM N,ADDR,RX (SUB4 #data,addr(Rx))	DX					17,02,02

Register Transfer Description:

$[DA]^2 \leftarrow [DA]^1 - N$, where $1 \le N \le 16$;

$PI_4 \leftarrow 1$, if $[DA0]^1 < 0 < [DA0]^2$;

$(CS) \leftarrow 0010$ if carry $= 0$ and $[DA] = 0$;
$(CS) \leftarrow 0001$ if carry $= 0$ and $[DA] < 0$;
$(CS) \leftarrow 0100$ if carry $= 0$ and $[DA] > 0$;
$(CS) \leftarrow 1010$ if carry $= 1$ and $[DA] = 0$;
$(CS) \leftarrow 1001$ if carry $= 1$ and $[DA] < 0$;
$(CS) \leftarrow 1100$ if carry $= 1$ and $[DA] > 0$;

Registers Affected:

CS, PI

DABS

Double Precision Absolute Value of Register

DESCRIPTION: If the sign bit of the double precision Derived Operand, DO (i.e., the sign bit of register (RB, RB+1)), is a one, its negative or 2's complement is stored into register RA and RA+1, such that register RA contains the MSH of the result. However, if the sign bit of DO is a zero, it is stored, unchanged, into RA and RA+1. The condition status, CS, is set based on the result in register RA and RA+1.

Note: RA may equal RB.

The absolute value of a number with a 1 in the sign bit and all other bits zero is the same word, and causes fixed point overflow to occur.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode			Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	
DABS RA,RB	R	A5	RA	RB	13,01,00 (POS. #) 21,01,00 (NEG. #)

Register Transfer Description:
(RA,RA+1) ← |DO|;

PI_4 ← 1, exit, if DO = $8000\ 0000_{16}$;

(CS) ← 0001 if (RA,RA+1) = $8000\ 0000_{16}$;
(CS) ← 0010 if (RA,RA+1) = 0;
(CS) ← 0100 if (RA,RA+1) > 0;

Registers Affected:
RA, RA+1, CS, PI

ABS

Single Precision Absolute Value of Register

DESCRIPTION: If the sign bit of the Derived Operand, DO (i.e., the sign bit of register RB), is a one, its negative or 2's complement is stored into register RA. However, if the sign bit of DO is a zero, it is stored, unchanged, into RA. The condition status, CS, is set based on the result in register RA.

Note: RA may equal RB.

The absolute value of a number with a 1 in the sign bit and all other bits zero is the same word, and causes fixed point overflow to occur.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode			Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	
ABS RA,RB (ABS Rb,Ra)	R	A4	RA	RB	05,01,00 (POS. #) 10,01,00 (NEG. #)

Register Transfer Description:
$(RA) \leftarrow |DO|;$

$PI_4 \leftarrow 1$, exit, if $DO = 8000_{16}$;

$(CS) \leftarrow 0001$ if $(RA) = 8000_{16}$;
$(CS) \leftarrow 0010$ if $(RA) = 0$;
$(CS) \leftarrow 0100$ if $(RA) > 0$;

Registers Affected:
RA, CS, PI

Logical AND

DESCRIPTION: The Derived Operand, DO, is bit-by-bit ANDed with the contents of register RA. The result is stored in register RA. The condition status, CS, is set based on the result in register RA.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode		Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
AND RA,ADDR (AND addr,Ra)	D	8: E2, 4: RA, 4: RX	16: ADDR	12,02,01
AND RA,ADDR,RX (AND addr(Rx),Ra)	DX			12,02,01
ANDB BR,DSPL (AND dspl(Br),R2)	B	4: 3, 2: 1, 2: BR′, 8: DSPL	12 ≤ BR ≤ 15 BR′ = BR − 12 RA = R2	11,01,01
ANDX BR,RX (AND (Rx)(Br),R2)	BX	4: 4, 2: 0, 2: BR′, 4: E, 4: RX	12 ≤ BR ≤ 15 BR′ = BR − 12 RA = R2	11,01,01
ANDM RA,DATA (AND #data,RA)	IM	8: 4A, 4: RA, 4: 7	16: DATA	11,02,00
ANDR RA,RB (AND Rb,Ra)	R	8: E3, 4: RA, 4: RB		04,01,00

Register Transfer Description:
(RA) ← (RA) ↑ DO;

(CS) ← 0010 if (RA) = 0;
(CS) ← 0001 if (RA) < 0;
(CS) ← 0100 if (RA) > 0;

Registers Affected:
RA, CS

N, NIM, NR

Logical NAND

DESCRIPTION: The Derived Operand, DO, is bit-by-bit logically NANDed with the contents of register RA. The result is stored in RA.

Note: The logical NOT of a register can be attained with a NR instruction with RA = RB.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	16	
N RA,ADDR (NAND addr,Ra)	D	E6	RA	RX	ADDR	15,02,01
N RA,ADDR,RX (NAND addr(Rx),Ra)	DX					15,02,01
		8	4	4	16	
NIM RA,DATA (NAND #data,Ra)	IM	4A	RA	B	DATA	14,02,00
		8	4	4		
NR RA,RB (NAND Rb,Ra)	R	E7	RA	RB		07,01,00

Register Transfer Description:
(RA) ← (RA) ↑ DO;

(CS) ← 0010 if (RA) = 0;
(CS) ← 0001 if (RA) < 0;
(CS) ← 0100 if (RA) > 0;

Registers Affected:
RA, CS

XOR, XORM, XORR

Exclusive Logical OR

DESCRIPTION: The Derived Operand, DO, is bit-by-bit exclusively ORed with the contents of RA. The result is stored in RA. The condition status, CS, is set based on the result in RA.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
			8	4	4	16	
XOR (XOR	RA,ADDR addr,Ra)	D	E4	RA	RX	ADDR	12,02,01
XOR (XOR	RA,ADDR,RX addr(Rx),Ra)	DX					12,02,01
			8	4	4	16	
XORM (XOR	RA,DATA #data,Ra)	IM	4A	RA	9	DATA	11,02,00
			8	4	4		
XORR (XOR	RA,RB Rb,Ra)	R	E5	RA	RB		04,01,00

Register Transfer Description:

(RA) ← (RA) ⊕ DO;

(CS) ← 0010 if (RA) = 0;
(CS) ← 0001 if (RA) < 0;
(CS) ← 0100 if (RA) > 0;

Registers Affected:

RA, CS

ORR, ORB, ORBX, OR, ORIM

Inclusive Logical OR

DESCRIPTION: The Derived Operand, DO, is bit-by-bit inclusively ORed with the contents of RA. The result is stored in register RA. The condition status, CS, is set based on the result in register RA.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode		Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
OR (OR	RA,ADDR addr,Ra)	D	8 4 4 E0 RA RX	16 ADDR	12,02,01
OR (OR	RA,ADDR,RX addr(Rx),Ra)	DX			12,02,01
ORB (OR	BR,DSPL dspl(Br),R2)	B	4 2 2 8 3 0 BR' DSPL	12 ≤ BR ≤ 15 BR' = BR − 12 RA = R2	11,01,01
ORBX (OR	BR,RX (Rx)(Br),R2)	BX	4 2 2 4 4 4 0 BR' F RX	12 ≤ BR ≤ 15 BR' = BR − 12 RA = R2	11,01,01
ORIM (OR	RA,DATA #data,Ra)	IM	8 4 4 4A RA 8	16 DATA	11,02,00
ORR (OR	RA,RB Rb,Ra)	R	8 4 4 E1 RA RB		04,01,00

Register Transfer Description:
(RA) ← (RA) v DO;

(CS) ← 0010 if (RA) = 0;
(CS) ← 0001 if (RA) < 0;
(CS) ← 0100 if (RA) > 0;

Registers Affected:
RA, CS

Single Precision Compare

DESCRIPTION: The single precision Derived Operand, DO, is compared to the contents of RA. Then, the Condition Status, CS, is set based on whether the contents of RA is less than, equal to, or greater than the DO. The contents of RA are unchanged.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode		Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
C (CMP	RA,ADDR Ra,addr)	D	8 4 4 F0 RA RX	16 ADDR	16,02,01
C (CMP	RA,ADDR,RX Ra,addr(Rx))	DX			16,02,01
CB (CMP	BR,DSPL R2,dspl(Br))	B	4 2 2 8 3 2 BR′ DSPL	12 ≤ BR ≤ 15 BR′ = BR − 12 RA = R2	15,01,01
CBX (CMP	BR,RX R2,(Rx)(Br))	BX	4 2 2 4 4 4 0 BR′ C RX	12 ≤ BR ≤ 15 BR′ = BR − 12 RA = R2	15,01,01
CIM (CMP	RA,DATA Ra,#data)	IM	8 4 4 4A RA A	16 DATA	15,02,00
CISP (CMP4	RA,N Ra,#data)	ISP	8 4 4 F2 RA N−1	1 ≤ N ≤ 16	11,01,00
CISN (CMP4	RA,N Ra,#data)	ISN	8 4 4 F3 RA N−1	1 ≤ N ≤ 16	11,01,00
CR (CMP	RA,RB Ra,Rb)	R	8 4 4 F1 RA RB		08,01,00

Register Transfer Description:
(RA) : DO;

(CS) ← 0010 if (RA) = DO;
(CS) ← 0001 if (RA) < DO;
(CS) ← 0100 if (RA) > DO;

Registers Affected:
CS

DC,DCR

Double Precision Compare

DESCRIPTION: The double precision Derived Operand, DO, is compared to the contents of registers RA and RA+1 where RA contains the MSH of a double precision word. Then, the Condition Status, CS, is set based on whether the contents of RA, RA+1 is less than, equal to, or greater than the DO. The contents of RA and RA+1 are unchanged.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
DC (CMPL	RA,ADDR Ra,addr)	D	8 F6	4 RA	4 RX	16 ADDR	27,02,02
DC (CMPL	RA,ADDR,RX Ra,addr(Rx))	DX					27,02,02
DCR (CMPL	RA,RB Ra,Rb)	R	8 F7	4 RA	4 RB		21,01,00

Register Transfer Description:
(RA,RA+1) : DO;

(CS) ← 0010 if (RA,RA+1) = DO;
(CS) ← 0001 if (RA,RA+1) < DO;
(CS) ← 0100 if (RA,RA+1) > DO

Registers Affected:
CS

Compare Between Limits

DESCRIPTION: The contents of register RA are compared to two (2) different sixteen bit derived operands, DO1 and DO2. The derived operands, DO1 and DO2 are located at DA and DA+1, respectively, and their values are defined such that DO1 ≤ DO2. The CS is set based on the results. If the values for DO1 and DO2 are defined incorrectly (that is, DO1 > DO2), then CS is set to 1000.

Mil Std Mnemonic (IEEE Mnemonic	Addr Mode	Format/Opcode	Clock Cycles, Instruction Fetch Bus Cycles, Operand Fetch Bus Cycles	
CBL RA,ADDR (CMPRNG Ra,addr)	D		(RA<DO1) (DO1<RA<DO2) (RA>DO2)	30,02,02 43,02,02 40,02,02
CBL RA,ADDR,RX (CMPRNG Ra,addr(Rx))	DX		(RA<DO1) (DO1<RA<DO2) (RA>DO2)	30,02,02 43,02,02 40,02,02

8	4	4	16
F4	RA	RX	ADDR

Register Transfer Description:
(CS) ← 1000 if DO1 > DO2, exit;
(CS) ← 0001 if (RA) < DO1;
(CS) ← 0010 if DO1 ≤ (RA) ≤ DO2;
(CS) ← 0100 if (RA) > DO2

Registers Affected:
CS

Floating Point
FA, FAB, FABX, FAR

Floating Point Add

DESCRIPTION: The floating point Derived Operand, DO, is floating point added to the contents of registers RA and RA+1. The result is stored in registers RA and RA+1. The process of this operation is as follows: the mantissa of the number with the smaller algebraic exponent is shifted right and the exponent incremented by one for each bit shifted until the exponents are equal. The mantissas are then added. If the sum overflows the 24-bit mantissa, then the sum is shifted right one position, the sign bit restored, and the exponent incremented by one. If the exponent exceeds $7F_{16}$ as a result of this incrementation, overflow occurs and the operation is terminated. If the sum does not result in exponent overflow, the result is normalized. If in the normalization process the exponent is decremented below 80_{16}, then underflow occurs and a zero is inserted for the result.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode		Clock Cycles*, Instruction Fetch Bus Cycles, Operand Bus Cycles
FA RA,ADDRR (ADDF addr,Ra)	D	8 4 4 16 A8 RA RX ADDR		68,02,02
FA RA,ADDR,RX (ADDF addr(Rx),Ra)	DX			68,02,02
FAB BR,DSPL (ADDF dspl(Br),R0)	B	4 2 2 8 2 0 BR′ DSPL	$12 \leq BR \leq 15$ $BR' = BR - 12$ $RA = R0$	67,01,02
FABX BR,RX (ADDF (Rx)(Br),R0)	BX	4 2 2 4 4 4 0 BR′ 8 RX	$12 \leq BR \leq 15$ $BR' = BR - 12$ $RA = R0$	67,01,02
FAR RA,RB (ADDF (Rb,Ra))	R	8 4 4 A9 RA RB		62,01,00

* Clock cycles assume no shifts in exponent adjust and no shifts in normalization

Register Transfer Description:

$N = EA - EO;$

$EA \leftarrow EO$, if $MA = 0$;

$MO \leftarrow MO$ Shifted Right Arithmetic n positions, if $n > 0$ and $MA \neq 0$;

$MA \leftarrow MA$ Shifted Right Arithmetic $-n$ positions, $EA \leftarrow EO$, if $n < 0$ and $MO \neq 0$;

$MA \leftarrow MA + MO;$

$MA \leftarrow MA$ Shifted Right Arithmetic 1 position, $MA_0 \leftarrow \overline{MA_0}$, $EA \leftarrow EA+1$, if $OVM = 1$;

$PI_3 \leftarrow 1$, $EA \leftarrow 7F_{16}$, $MA \leftarrow 7FFF\ FF_{16}$, exit, if $EA > 7F_{16}$ and $MA_0 = 0$;

$PI_3 \leftarrow 1$, $EA \leftarrow 7F_{16}$, $MA \leftarrow 8000\ 00_{16}$, exit, if $EA > 7F_{16}$ and $MA_0 = 1$;

EA, MA ← normalized EA, MA;

PI_6 ← 1, EA ← 0, MA ← 0, if EA < 80_{16};

(CS) ← 0010 if (RA, RA+1) = 0;
(CS) ← 0001 if (RA, RA+1) < 0;
(CS) ← 0100 if (RA, RA+1) > 0;

Registers Affected:
RA, RA+1, CS, PI

Floating Point

EFA, EFAR

Extended Precision Floating Point Add

DESCRIPTION: The extended precision floating point Derived Operand, DO, is extended floating point added to the contents of register RA, RA+1, and RA+2. The result is stored in register RA, RA+1, and RA+2. The process of this operation is as follows: the mantissa of the number with the smaller algebraic exponent is shifted right and the exponent is incremented by one for each bit shifted. When the exponents are equal, the mantissas are added. If the sum overflows the 39-bit mantissa, then the sum is shifted right one position, the sign bit restored, and the exponent is incremented by one. If the exponent exceeds $7F_{16}$ as a result of this incrementation, overflow occurs and the operation is terminated. If the sum does not result in exponent overflow, the result is normalized. If in the normalization process the exponent is decremented below 80_{16}, then underflow occurs and a zero is inserted for the result.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles*, Instruction Fetch Bus Cycles, Operand Bus Cycles
EFA RA,ADDRR (ADDX addr,Ra)	D	8 AA	4 RA	4 RX	16 ADDR	78,02,03
EFA RA,ADDR,RX (ADDX addr(Rx),Ra)	DX					78,02,03
EFAR RA,RB (ADDX Rb,Ra)	R	8 AB	4 RA	4 RB		71,01,00

* Clock cycles assume no shifts in exponent adjust and no shifts in normalization

Register Transfer Description:

$n = EA - EO;$

$EA \leftarrow EO$, if $MA = 0;$

$MO \leftarrow MO$ Shifted Right Arithmetic n positions, if $n > 0$ and $MA \neq 0;$

$MA \leftarrow MA$ Shifted Right Arithmetic $-n$ positions, $EA \leftarrow EO$, if $n < 0$ and $MO \neq 0;$

$MA \leftarrow MA + MO;$

$MA \leftarrow MA$ Shifted Right Arithmetic 1 position, $MA_0, \leftarrow \overline{MA_0}$, $EA \leftarrow EA+1$, if $OVM = 1;$

$PI_3 \leftarrow 1$, $EA \leftarrow 7F_{16}$, $MA \leftarrow 7FFF\ FF\ FFFF_{16}$, exit, if $EA > 7F_{16}$ and $MA_0 = 0;$

$PI_3 \leftarrow 1$, $EA \leftarrow 7F_{16}$, $MA \leftarrow 8000\ 00\ 0000_{16}$, exit, if $EA > 7F_{16}$ and $MA_0 = 1;$

$EA, MA \leftarrow$ normalized $EA, MA;$

$PI_6 \leftarrow 1$, $EA \leftarrow 0$, $MA \leftarrow 0$, if $EA < 80_{16};$

$(CS) \leftarrow 0010$ if $(RA, RA+1, RA+2) = 0;$
$(CS) \leftarrow 0001$ if $(RA, RA+1, RA+2) < 0;$
$(CS) \leftarrow 0100$ if $(RA, RA+1, RA+2) > 0;$

Registers Affected:
RA, RA+1, RA+2, CS, PI

FABS

Floating Point Absolute Value of Register

DESCRIPTION: If the sign bit of the mantissa of the Derived Operand, DO (i.e. the contents of registers RB and RB + 1) is a one, its floating point negative is stored in registers RA and RA + 1. The negative of DO is computed by taking the 2's complement of the mantissa and leaving the exponent unchanged. Exceptions to this are negative powers of two: -1.0×20, -1.0×21,.... The absolute value of these are: 0.5×2^1, 0.5×2^2,...; in other words, the DO mantissa is shifted logically right one position and the exponent incremented. A floating point overflow shall occur if DO is the smallest negative number, -1.0×2^{127}. If the sign bit of DO is a zero, it is stored unchanged into RA and RA + 1. The condition status, CS, is set based on the result in register RA and RA + 1.

Note: RA may equal RB.

DO is assumed to be a normalized number or floating point zero.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode			Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	
FABS RA,RB (ABSF Rb,Ra)	R	AC	RA	RB	16,01,00 (POS.#) 62,01,00 (NEG.#)

Register Transfer Description:

$EA \leftarrow EA+1$, $MA \leftarrow 4000\ 00_{16}$, if $MO = 8000\ 00_{16}$;

$PI_3 \leftarrow 1$, $EA \leftarrow 7F_{16}$, $MA \leftarrow 7FFF\ FF_{16}$, exit, if $EA > 7F_{16}$;

$EA \leftarrow EO$, $MA \leftarrow -MO$, if $MO < 0$, $MO \neq 8000\ 00_{16}$;

$EA \leftarrow EO$, $MA \leftarrow MO$, if $MO > 0$;

$(CS) \leftarrow 0010$ if $(RA, RA+1) = 0$;
$(CS) \leftarrow 0001$ if $(RA, RA+1) < 0$;
$(CS) \leftarrow 0100$ if $(RA, RA+1) > 0$;

Registers Affected:
RA, RA + 1, CS, PI

FS, FSB, FSBX, FSR

Floating Point Subtract

DESCRIPTION: The floating point Derived Operand, DO, is floating point subtracted from the contents of registers RA and RA+1. The result is stored in registers RA and RA+1. The process of this operation is as follows: the mantissa of the number with the smaller algebraic exponent is shifted right and the exponent incremented by one for each bit shifted until the exponents are equal. The mantissa of the DO is then subtracted from (RA,RA+1). If the difference overflows the 24-bit mantissa, then it is shifted right one position, the sign bit restored and the exponent incremented by one. If the exponent exceeds $7F_{16}$ as a result of this incrementation, overflow occurs and the operation is terminated. If the sum does not result in exponent overflow, the result is normalized. If during the normalization process the exponent is decremented below 80_{16}, then underflow occurs and a zero is inserted for the result.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode		Clock Cycles*, Instruction Fetch Bus Cycles, Operand Bus Cycles
FS (SUBF	RA,ADDRR addr,Ra)	D	8 4 4 B8 RA RX	16 ADDR	68,02,02
FS (SUBF	RA,ADDR,RX addr(Rx),Ra)	DX			68,02,02
FSB (SUBF	BR,DSPL dspl(Br),RO)	B	4 2 2 8 2 1 BR' DSPL	12 ≤ BR ≤ 15 BR' = BR − 12 RA = R0	67,01,02
FSBX (SUBF	BR,RX (Rx)(Br),RO)	BX	4 2 2 4 4 4 0 BR' 9 RX	12 ≤ BR ≤ 15 BR' = BR − 12 RA = R0	67,01,02
FSR (SUBF	RA,RB Rb,Ra)	R	8 4 4 B9 RA RB		62,01,00

Register Transfer Description:

n = EA − EO;

EA ← EO, if MA = 0;

MO ← MO Shifted Right Arithmetic n positions, if $n > 0$ and $MA \neq 0$;

MA ← MA Shifted Right Arithmetic −n positions, EA ← EO, if $n < 0$ and $MO \neq 0$;

MA ← MA − MO;

MA ← MA Shifted Right Arithmetic 1 position, $MA_0 \leftarrow \overline{MA_0}$, EA ← EA+1, if OVM = 1;

$PI_3 \leftarrow 1$, EA ← $7F_{16}$, MA ← 7FFF FF_{16}, exit, if EA > $7F_{16}$ and $MA_0 = 0$;

$PI_3 \leftarrow 1$, EA ← $7F_{16}$, MA ← 8000 00_{16}, exit, if EA > $7F_{16}$ and $MA_0 = 1$;

EA, MA ← normalized EA, MA;

$PI_6 \leftarrow 1$, EA ← 0, MA ← 0, if EA < 80_{16};

(CS) ← 0010 if (RA, RA+1) = 0;
(CS) ← 0001 if (RA, RA+1) < 0;
(CS) ← 0100 if (RA, RA+1) > 0;

Registers Affected:
RA, RA+1, CS, PI

Extended Precision Floating Point Subtract

DESCRIPTION: The extended precision floating point Derived Operand, DO, is extended floating point subtracted from the contents of register RA, RA+1, and RA+2. The result is stored in register RA, RA+1, and RA+2. The process of this operation is as follows: the mantissa of the number with the smaller algebraic exponent is shifted right and the exponent is incremented by one for each bit shifted. When the exponents are equal, the mantissas are subtracted. If the difference overflows the 39-bit mantissa, then the difference is shifted right one position, the sign bit restored, and the exponent is incremented. If the exponent exceeds $7F_{16}$ as a result of this incrementation, overflow occurs and the operation is terminated. If the difference does not result in exponent overflow, the result is normalized. If during in the normalization process the exponent is decremented below 80_{16}, then underflow occurs and a zero is inserted for the result.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles*, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	16	
EFS RA,ADDR (SUBX addr,Ra)	D	BA	RA	RX	ADDR	78,02,03
EFS RA,ADDR,RX (SUBX addr(Rx),Ra)	DX					78,02,03
		8	4	4		
EFSR RA,RB (SUBX Rb,Ra)	R	BB	RA	RB		71,01,00

* Clock cycles assume no shifts for exponent and no shifts for normalization

Register Transfer Description:

n = EA − EO;

EA ← EO, if MA = 0;

MO ← MO Shifted Right Arithmetic n positions, if $n > 0$ and $MA \neq 0$;

MA ← MA Shifted Right Arithmetic −n positions, EA ← EO, if $n < 0$ and $MO \neq 0$;

MA ← MA − MO; MA ← MA Shifted Right Arithmetic 1 position, $MA_0 \leftarrow \overline{MA_0}$, EA ← EA+1, if OVM = 1;

$PI_3 \leftarrow 1$, EA ← $7F_{16}$, MA ← 7FFF FF $FFFF_{16}$, exit, if EA > $7F_{16}$ and $MA_0 = 0$;

$PI_3 \leftarrow 1$, EA ← $7F_{16}$, MA ← 8000 00 0000_{16}, exit, if EA > $7F_{16}$ and $MA_0 = 1$;

EA, MA ← normalized EA, MA;

$PI_6 \leftarrow 1$, EA ← 0, MA ← 0, if EA < 80_{16};

(CS) ← 0010 if (RA, RA+1, RA+2) = 0;
(CS) ← 0001 if (RA, RA+1, RA+2) < 0;
(CS) ← 0100 if (RA, RA+1, RA+2) > 0;

Registers Affected:
RA, RA+1, RA+2, CS, PI

FNEG

Floating Point Negate Register

DESCRIPTION: The 24-bit mantissa of the Derived Operand, DO, i.e. the floating point number in registers RB and RB+1, is 2's complemented. The exponent remains unchanged. The result, the negative of the original number, is stored in RA and RA+1. The 2's complement of a floating point zero is a floating point zero. Exceptions to this are all powers of two: -1.0×2^n and $0.5\times 2^{n-1}$; i.e. when the mantissa is either $8000\ 00_{16}$ or $4000\ 00_{16}$. The negation of 0.5×2^n is $-1.0\times 2^{n-1}$; i.e. the mantissa is shifted left one position and the exponent decremented by one. Conversely, the negation of -1.0×2^n is $0.5\times 2^{n+1}$; i.e. the mantissa is shifted right one position and the exponent is incremented by one. A floating point overflow occurs for the negation of the smallest negative number, -1.0×2^{127}. A floating point underflow occurs for the negation of the smallest positive number, 0.5×2^{128}, and causes the result to be zero. The conditon status, CS, is set based on the result in registers RA and RA+1.

Note: RA may equal RB.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode			Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	
FNEG RA,RB (NEGF Rb,Ra)	R	BC	RA	RB	56,01,00

Register Transfer Description:

$PI_3 \leftarrow 1$, $EA \leftarrow 7F_{16}$, $MO \leftarrow 7FFF\ FF_{16}$, exit, if $DO = 8000\ 007F_{16}$;

$PI_6 \leftarrow 1$, $EA \leftarrow 0$, $MA \leftarrow 0$, exit, if $DO = 4000\ 0080_{16}$;

$EA \leftarrow EO+1$, $MA \leftarrow 4000\ 00_{16}$, if $MO = 8000\ 00_{16}$;

$EA \leftarrow EO-1$, $MA \leftarrow 8000\ 00_{16}$, if $MO = 4000\ 00_{16}$;

$EA \leftarrow EO$, $MA \leftarrow -MO$, if $MO \neq 8000\ 00_{16}$ or $4000\ 00_{16}$;

(CS) ← 0010 if (RA, RA+1) = 0;
(CS) ← 0001 if (RA, RA+1) < 0;
(CS) ← 0100 if (RA, RA+1) > 0;

Registers Affected:
RA, RA+1, CS, PI

Floating Point Multiply

DESCRIPTION: The floating point Derived Operand, DO, is floating point multiplied by the contents of registers RA and RA+1. The result is stored in registers RA and RA+1. The process of the operation is as follows: the exponents of the operands are added. If the sum exceeds $7F_{16}$, a floating point overflow occurs. If the sum is less than 80_{16} then underflow occurs and the result set to zero. The operand mantissas are multiplied and the result normalized and stored in RA and RA+1. An exceptional case is when both operands are negative powers of two: $(-1.0 \times 2^n) \times (-1.0 \times 2^m)$; the result is a $0.5 \times 2^{n+m+1}$. If $n+m = 7F_{16}$, this shall yield an exponent overflow, floating point overflow occurs. Also, it is possible that the normalization process may yield an exponent underflow; if this occurs, then the result is forced to zero. The condition status, CS, is set based on the result in RA and RA+1.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode		Clock Cycles*, Instruction Fetch Bus Cycles, Operand Bus Cycles
FM RA,ADDRR (MULF addr,Ra)	D	8 4 4 C8 | RA | RX	16 ADDR	126,02,02
FM RA,ADDR,RX (MULF addr(Rx),Ra)	DX			126,02,02
FMB BR,DSPL (MULF dspl(Br),RO)	B	4 2 2 8 2 | 2 | BR' | DSPL	$12 \leq BR \leq 15$ BR' = BR − 12 RA = R0	125,01,02
FMBX BR,RX (MULF (Rx)(Br),RO)	BX	4 2 2 4 4 4 | 0 | BR' | A | RX	$12 \leq BR \leq 15$ BR' = BR − 12 RA = R0	125,01,02
FMR RA,RB (MULF Rb,Ra)	R	8 4 4 C9 | RA | RB		120,01,00

* Clock cycles assume no shifts for exponent and no shifts for normalization

Register Transfer Description:

n = EA + EO;

$PI_3 \leftarrow 1$, $EA \leftarrow 7F_{16}$, $MA \leftarrow 7FFF\ FF_{16}$, exit, if $n > 7F_{16}$ and $MA_0 = MO_0$;

$PI_3 \leftarrow 1$, $EA \leftarrow 7F_{16}$, $MA \leftarrow 8000\ 00_{16}$, exit, if $n > 7F_{16}$ and $MA_0 \neq MO_0$;

$PI_6 \leftarrow 1$, $EA \leftarrow 0$, $MA \leftarrow 0$, exit, if $n < 80_{16}$;

$MP \leftarrow MA \times MO$; (integer multiply)

$MP \leftarrow MP$ shift left 1 position;

$n \leftarrow n + 1$, $MP_{0-23} \leftarrow 4000\ 00_{16}$, if $MP_{0-23} = 8000\ 00_{16}$;

$PI_3 \leftarrow 1$, $EA \leftarrow 7F_{16}$, $MA \leftarrow 7FFF\ FF_{16}$, exit, if $n > 7F_{16}$ and $MP_0 = 0$;

$PI_3 \leftarrow 1$, $EA \leftarrow 7F_{16}$, $MA \leftarrow 8000\ 00_{16}$, exit, if $n > 7F_{16}$ and $MP_0 = 1$;

n, MP ← normalized n, MP;

$PI_6 \leftarrow 1$, $EA \leftarrow 0$, $MA \leftarrow 0$, exit, if $n < 80_{16}$;

EA ← n;

$MA \leftarrow MP_{0-23}$;

(CS) ← 0010 if (RA, RA+1) = 0;
(CS) ← 0001 if (RA, RA+1) < 0;
(CS) ← 0100 if (RA, RA+1) > 0;

Registers Affected:
RA, RA+1, CS, PI

Extended Precision Floating Point Multiply

DESCRIPTION: The extended precision floating Derived Operand, DO, is extended floating point multiplied by the contents of registers RA, RA+1 and RA+2. The result is stored in reigsters RA, RA+1 and RA+2. The process of the operation is as follows: the exponent of the operands are added. If the sum exceeds $7F_{16}$, a floating point overflow occurs. If the sum is less than 80_{16}, then underflow occurs and the result set to zero. The operand mantissas are multiplied and the result normalized and stored in RA, RA+1 and RA+2. The condition status, CS, is set based on the result in RA, RA+1 and RA+2.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles*, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	16	
EFM RA,ADDR (MULX addr,Ra)	D	CA	RA	RX	ADDR	258,02,03
EFM RA,ADDR,RX (MULX addr(Rx),Ra)	DX					258,02,03
		8	4	4		
EFMR RA,RB (MULX Rb,Ra)	R	CB	RA	RB		251,01,00

* Clock cycles assume no shifts for exponent and no shifts for normalization

Register Transfer Description:

$n = EA + EO;$

$PI_3 \leftarrow 1$, $EA \leftarrow 7F_{16}$, $MA \leftarrow 7FFF\ FF\ FFFF_{16}$, exit, if $n > 7F_{16}$ and $MA_0 = MO_0$;

$PI_3 \leftarrow 1$, $EA \leftarrow 7F_{16}$, $MA \leftarrow 8000\ 00\ 0000_{16}$, exit, if $n > 7F_{16}$ and $MA_0 \neq MO_0$;

$PI_6 \leftarrow 1$, $EA \leftarrow 0$, $MA \leftarrow 0$, exit, if $n < 80_{16}$;

$MP \leftarrow MA \times MO$; (integer multiply)

$MP \leftarrow MP$ shift left 1 position;

$n \leftarrow n + 1$, $MP_{0-39} \leftarrow 4000\ 00\ 0000_{16}$, if $MP_{0-39} = 8000\ 00\ 0000_{16}$;

$PI_3 \leftarrow 1$, $EA \leftarrow 7F_{16}$, $MA \leftarrow 7FFF\ FF\ FFFF_{16}$, exit, if $n > 7F_{16}$ and $MP_0 = 0$;

$PI_3 \leftarrow 1$, $EA \leftarrow 7F_{16}$, $MA \leftarrow 8000\ 00\ 0000_{16}$, exit, if $n > 7F_{16}$ and $MP_0 = 1$;

$n, MP \leftarrow$ normalized n, MP;

$PI_6 \leftarrow 1$, $EA \leftarrow 0$, $MA \leftarrow 0$, if $n < 80_{16}$;

$EA \leftarrow n$;

$MA \leftarrow MP_{0-39}$;

$(CS) \leftarrow 0010$ if $(RA, RA+1, RA+2) = 0$;
$(CS) \leftarrow 0001$ if $(RA, RA+1, RA+2) < 0$;
$(CS) \leftarrow 0100$ if $(RA, RA+1, RA+2) > 0$;

Registers Affected:
RA, RA+1, RA+2, CS, PI

Floating Point

FD, FDB, FDBX, FDR

Floating Point Divide

DESCRIPTION: The floating point number in registers RA and RA+1 is divided by the floating point Derived Operand, DO. The result is stored in register RA and RA+1. A floating point overflow occurs if the exponent result exceeds $7F_{16}$ at any point in the calculation process. Underflow occurs if the exponent result is less than 80_{16} at any point in the process. If underflow occurs, then the quotient is forced to zero. A divide by zero yields a floating point overflow.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode		Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
FD RA,ADDRR (DIVF addr,Ra)	D	8 4 4 16 D8 RA RX ADDR		240,02,02
FD RA,ADDR,RX (DIVF addr(Rx),Ra)	DX			240,02,02
FDB BR,DSPL (DIVF dspl(Br),RO)	B	4 2 2 8 2 3 BR' DSPL	$12 \leq BR \leq 15$ $BR' = BR - 12$ $RA = R0$	239,01,02
FDBX BR,RX (DIVF (Rx)(Br),RO)	BX	4 2 2 4 4 4 0 BR' B RX	$12 \leq BR \leq 15$ $BR' = BR - 12$ $RA = R0$	239,01,02
FDR RA,RB (DIVF Rb,Ra)	R	8 4 4 D9 RA RB		234,01,00

Register Transfer Description:

$n = EA - EO;$

$n \leftarrow 0$, if $MA = 0$

$PI_3 \leftarrow 1$, $EA \leftarrow 7F_{16}$, $MA \leftarrow 7FFF\ FF_{16}$, exit, if $MA_0 = MO_0$;
and $\{n > 7F_{16}$ or $DO = 0\}$;

$PI_3 \leftarrow 1$, $EA \leftarrow 7F_{16}$, $MA \leftarrow 8000\ 00_{16}$, exit, if $MA_0 \neq MO_0$;
and $\{n > 7F_{16}$ or $DO = 0\}$;

$PI_6 \leftarrow 1$, $EA \leftarrow 0$, $MA \leftarrow 0$, exit, if $n < 80_{16}$;

$MQ \leftarrow MA / MO$;

$MQ \leftarrow MQ$ Shift Right Arithmetic 1 position, $n \leftarrow n + 1$ if $|MQ| \geq 1.0$;

$PI_3 \leftarrow 1$, $EA \leftarrow 7F_{16}$, $MA \leftarrow 7FFF\ FF_{16}$, exit, if $n > 7F_{16}$ and $MQ_0 = 0$;

```
$PI_3 \leftarrow 1$, $EA \leftarrow 7F_{16}$, $MA \leftarrow 8000\ 00_{16}$, exit, if $n > 7F_{16}$ and $MQ_0 = 1$;

$EA \leftarrow n$;

$MA \leftarrow MP_{0-23}$;

$(CS) \leftarrow 0010$ if $(RA, RA+1) = 0$;
$(CS) \leftarrow 0001$ if $(RA, RA+1) < 0$;
$(CS) \leftarrow 0100$ if $(RA, RA+1) > 0$;
```

Registers Affected:
RA, RA+1, CS, PI

EFD, EFDR

Extended Precision Floating Point Divide

DESCRIPTION: The contents of registers RA, RA+1, and RA+2 are extended precision floating point divided by the extended precision floating point Derived Operand, DO. The result is stored in register RA, RA+1, RA+2. A floating point overflow occurs if the exponent result exceeds $7F_{16}$ at any point in the calculation process. Underflow occurs if the exponent result is less than 80_{16} at any point in the process. If underflow occurs, then the quotient is forced to zero. A divide by zero yields a floating point overflow.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
			8	4	4	16	
EFD	RA,ADDRR	D	DA	RA	RX	ADDR	487,02,03
(DIVX	addr,Ra)						
EFD	RA,ADDR,RX	DX					487,02,03
(DIVX	addr(Rx),Ra)						
			8	4	4		
EFDR	RA,RB	R	DB	RA	RB		480,01,00
(DIVX	Rb,Ra)						

Register Transfer Description:

$n = EA - EO;$

$n \leftarrow 0$, if $MA = 0$

$PI_3 \leftarrow 1$, $EA \leftarrow 7F_{16}$, $MA \leftarrow 7FFF\ FF\ FFFF_{16}$, exit, if $MA_0 = MO_0$;
and $\{n > 7F_{16}$ or $DO = 0\}$;

$PI_3 \leftarrow 1$, $EA \leftarrow 7F_{16}$, $MA \leftarrow 8000\ 00\ 0000_{16}$, exit, if $MA_0 \neq MO_0$;
and $\{n > 7F_{16}$ or $DO = 0\}$;

$PI_6 \leftarrow 1$, $EA \leftarrow 0$, $MA \leftarrow 0$, exit, if $n < 80_{16}$;

$MQ \leftarrow MA / MO$;

$MQ \leftarrow MQ$ shift Right Arithmetic 1 position, $n \leftarrow n + 1$ if $|MQ| \geq 1.0$;

$PI_3 \leftarrow 1$, $EA \leftarrow 7F_{16}$, $MA \leftarrow 7FFF\ FF\ FFFF_{16}$, exit, if $n > 7F_{16}$ and $MQ_0 = 0$;

$PI_3 \leftarrow 1$, $EA \leftarrow 7F_{16}$, $MA \leftarrow 8000\ 00\ 0000_{16}$, exit, if $n > 7F_{16}$ and $MQ_0 = 1$;

$EA \leftarrow n$;

$MA \leftarrow MQ_{0-39}$;

$(CS) \leftarrow 0010$ if $(RA, RA+1, RA+2) = 0$;
$(CS) \leftarrow 0001$ if $(RA, RA+1, RA+2) < 0$;
$(CS) \leftarrow 0100$ if $(RA, RA+1, RA+2) > 0$;

Registers Affected:
RA, RA+1, CS, PI

Floating Point Compare

DESCRIPTION: The floating point number in registers RA and RA+1 is compared to the floating point Derived Operand, DO. Then, the Condition Status, CS, is set based on whether the contents of RA, RA+1 is less than, equal to, or greater than the DO. The contents of RA and RA+1 are unchanged.

Note: This instruction does not cause an overflow to occur.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode		Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
FC (CMPF	RA,ADDR RA,addr)	D	8 4 4 F8 RA RX	16 ADDR	58,02,02
FC (CMPF	RA,ADDR,RX Ra(Rx),addr)	DX			58,02,02
FCB (CMPF	BR,DSPL RO(Br),dspl)	B	4 2 2 8 3 3 BR' DSPL	12 ≤ BR ≤ 15 BR' = BR − 12 RA = R0	57,01,02
FCBX (CMPF	BR,RX (RO)(Br),Rx)	BX	4 2 2 4 4 4 0 BR' D RX	12 ≤ BR ≤ 15 BR' = BR − 12 RA = R0	57,01,02
FCR (CMPF	RA,RB Ra,Rb)	R	8 4 4 F9 RA RB		52,01,00

Register Transfer Description:
(RA, RA+1) : DO;

(CS) ← 0010 if (RA, RA+1) = DO;
(CS) ← 0001 if (RA, RA+1) < DO;
(CS) ← 0100 if (RA, RA+1) > DO;

Registers Affected:
CS

EFC, EFCR

Extended Precision Floating Point Compare

DESCRIPTION: The extended precision floating Derived Operand, DO, is compared to the contents of registers RA, RA+1, and RA+2 where RA contains the most significant 16 bits of the extended precision floating point word. The condition status, CS, is set based on whether the contents of RA, RA+1, and RA+2 are less than, equal to or greater than the DO. The contents of RA, RA+1, and RA+2 are unchanged.

Note: This instruction does not cause overflow to occur.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	16	
EFC RA,ADDRR (CMPX Ra,addr)	D	FA	RA	RX	ADDR	65,02,03
EFC RA,ADDR,RX (CMPX Ra,addr(Rx))	DX					65,02,03
		8	4	4		
EFCR RA,RB (CMPX Ra,Rb)	R	FB	RA	RB		52,01,00

Register Transfer Description:
(RA, RA+1, RA+2) : DO;

(CS) ← 0010 if (RA, RA+1,RA+2) = DO;
(CS) ← 0001 if (RA, RA+1,RA+2) < DO;
(CS) ← 0100 if (RA, RA+1,RA+2) > DO;

Registers Affected:
CS

FIX

Convert Floating Point to 16-Bit Integer

DESCRIPTION: The integer portion of the floating point Derived Operand, DO (i.e., the contents of registers RB and RB + 1) is stored into register RA. If the actual value of the DO floating point exponent is greater than $0F_{16}$, then RA remains unchanged and a fixed point overflow occurs. The condition status, CS, is set based on the result in RA.

Note: The algorithm truncates toward zero.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode			Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	
FIX RA,RB (INTGR Rb,Ra)	R	E8	RA	RB	10,01,00

Register Transfer Description:

$PI_4 \leftarrow 1$, exit, if EO > $0F_{16}$;

(RA) : Integer portion of DO;

(CS) ← 0010 if (RA) = 0 ;
(CS) ← 0001 if (RA) < 0 ;
(CS) ← 0100 if (RA) > 0 ;

Registers Affected:
RA, CS, PI

FLT

Convert 16-Bit Integer to Floating Point

DESCRIPTION: The integer Derived Operand, DO (i.e., the contents of register RB), is converted to Single Precision floating point format and stored in register RA and RA+1. The condition status, CS, is set based on the results in RA and RA+1. The operation process is as follows: The exponent is initially considered to be $0F_{16}$. The integer value in RB is normalized, i.e., the number is left shifted and the exponent decremented for each shift until the sign bit and the next MSB are unequal, and the exponent and mantissa stored in the proper fields of RA and RA+1.

Note: RA may equal RB.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode			Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	
FLT RA,RB (FLOAT Rb,Ra)	R	E9	RA	RB	16,01,00

Register Transfer Description:

EA ← 0, MA ← 0, exit, if (RB) = 0;

EA ← $0F_{16}$;

MA ← (RB);

EA, MA ← normalize EA, MA;

(CS) ← 0010 if (RA, RA+1) = 0 ;
(CS) ← 0001 if (RA, RA+1) < 0 ;
(CS) ← 0100 if (RA, RA+1) > 0 ;

Registers Affected:

RA, RA+1, CS

Convert Extended Precision Floating Point to 32-Bit Integer

DESCRIPTION: The integer portion of the floating point Derived Operand, DO (i.e., the contents of registers RB, RB + 1, and RB + 2) is stored into register RA and RA + 1. If the actual value of the DO floating point exponent is greater than $1F_{16}$ then RA and RA + 1 remain unchanged and a fixed point overflow occurs. The condition status, CS, is set based on the result in RA and RA + 1.

Note: The algorithm truncates toward zero.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode	Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
EFIX RA,RB (INTGRX Rb,Ra)	R	8 4 4 EA RA RB	21,01,00

Register Transfer Description:

$PI_4 \leftarrow 1$, exit, if $EO > 1F_{16}$;

(RA, RA + 1) ← Integer portion of DO;

$(CS) \leftarrow 0010$ if $(RA, RA+1) = 0$;
$(CS) \leftarrow 0001$ if $(RA, RA+1) < 0$;
$(CS) \leftarrow 0100$ if $(RA, RA+1) > 0$;

Registers Affected:
RA, RA + 1, CS, PI

EFLT

Convert 32 Bit Integer to Extended Precision Floating Point

DESCRIPTION: The double precision integer Derived Operand, DO (i.e., the contents of register RB and RB+1), is converted to Extended Precision floating point format and stored in register RA, RA+1, and RA+2. The condition status, CS, is set based on the result in RA, RA+1, and RA+2. The operation process is as follows: The exponent is initially considered to be $1F_{16}$. The integer value in RB, RB+1 is normalized, i.e., the number is left shifted and the exponent decremented for each shift until the sign bit and the next MSB are unequal, and the exponent and mantissa stored in the proper field of RA, RA+1, and RA+2.

Note: RA may equal RB.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode			Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	
EFLT RA,RB (FLOATL Rb,Ra)	R	EB	RA	RB	25,01,00

Register Transfer Description:

EA ← 0, MA ← 0, exit, if (RB,RB+1) = 0;

EA ← $1F_{16}$, MA ← (RB,RB+1);

EA, MA ← normalize EA, MA;

(CS) ← 0010 if (RA, RA+1, RA+2) = 0 ;
(CS) ← 0001 if (RA, RA+1, RA+2) < 0 ;
(CS) ← 0100 if (RA, RA+1, RA+2) > 0 ;

Registers Affected:

RA, RA+1, RA+2, CS

Bit Operations

SB, SBI, SBR

Set Bit

DESCRIPTION: Bit number N of the Derived Operand, DO, is set to one. The MSB is designated bit number zero and the LSB is designated bit number fifteen.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
			8	4	4	16	
SB (SETI	N,ADDR #data,addr)	D	50	N	RX	ADDR	16,02,02
SB (SETI	N,ADDR,RX #data,addr(Rx))	DX					16,02,02
			8	4	4	16	
SBI (SETI	N,ADDR #data,@addr)	I	52	N	RX	ADDR	20,02,03
SBI (SETI	N,ADDR,RX #data,addr(Rx)@)	IX					20,02,03
			8	4	4		
SBR (SETI	N,RB #data,Rb)	R	51	N	RB		07,01,00

Register Transfer Description:
$DO_N \leftarrow 1;$

Registers Affected:
RB

Reset Bit

DESCRIPTION: Bit number N of the Derived Operand, DO, is set to zero. The MSB is designated bit number zero and the LSB is designated bit number fifteen.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
			8	4	4	16	
RB (CLRI	N,ADDR #data,addr)	D	53	N	RX	ADDR	16,02,02
RB (CLRI	N,ADDR,RX #data,addr(Rx))	DX					16,02,02
			8	4	4	16	
RBI (CLRI	N,ADDR #data,@addr)	I	55	N	RX	ADDR	20,02,03
RBI (CLRI	N,ADDR,RX #data,addr(Rx)@)	IX					20,02,03
			8	4	4		
RBR (CLRI	N,RB #data,Rb)	R	54	N	RB		07,02,00

Register Transfer Description:
$DO_N \leftarrow 0$;

Registers Affected:
RB

Bit Operations

TB, TBI, TBR

Test Bit

DESCRIPTION: Bit number N ($0 \leq N \leq 15$) of the Derived Operand, DO, is tested. Then the Condition Status, CS, is set to indicate non-zero if bit number N of the DO contains a one. Otherwise CS is set to indicate zero. The MSB of the DO is designated bit number zero and the LSB of the DO is designated bit number fifteen.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
			8	4	4	16	
TB (TESTI	N,ADDR #data,addr)	D	56	N	RX	ADDR	15,02,01
TB (TESTI	N,ADDR,RX #data,addr(Rx))	DX					15,02,01
			8	4	4	16	
TBI (TESTI	N,ADDR #data,@addr)	I	58	N	RX	ADDR	19,02,02
TBI (TESTI	N,ADDR,RX #data,addr(Rx)@)	IX					19,02,02
			8	4	4		
TBR (TESTI	N,RB #data,Rb)	R	57	N	RB		07,02,00

Register Transfer Description:

(CS) ← 0010 if $DO_N = 0$ and $0 \leq N \leq 15$;
(CS) ← 0001 if $DO_N = 1$ and $N = 0$;
(CS) ← 0100 if $DO_N = 1$ and $1 \leq N \leq 5$;

Registers Affected:
CS

TSB

Test and Set Bit

DESCRIPTION: Bit number N (0≤N≤15) of the Derived Operand, DO, is tested and set to one. The CS is set according to the test.

Test: External memory accesses shall be inhibited until this instruction is complete.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
TSB (TESTSETI	N,ADDR #data,addr)	D	8	4	4	16	23,02,03
TSB (TESTSETI	N,ADDR,RX #data,addr(RX))	DX	59	N	RX	ADDR	23,02,03

Register Transfer Description:

$(CS) \leftarrow 0010$ and $(DO_N) \leftarrow 1$ if $DO_N = 0$ and $0 \leq N \leq 15$;
$(CS) \leftarrow 0001$ if $(DO_N) = 1$ and $N = 0$;
$(CS) \leftarrow 0100$ if $(DO_N) = 1$ and $1 \leq N \leq 15$;

Registers Affected:
CS

SVBR

Set Variable Bit in Register

DESCRIPTION: Bit number N(0≤N≤15) of the register RB is set to one where the least significant four bits of the contents of register RA is N. Bits $(RA)_{0-11}$ have no effect on the operation. If RA = RB, then the count is determined first and then the appropriate bit is changed.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode	Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
SVBR RA,RB (SETI Ra,Rb)	R	8 4 4 59 N RX	07,01,00

Register Transfer Description:
$(RB)_N \leftarrow 1$ where $N = (RA)_{12-15}$

Registers Affected:
RB

Reset Variable Bit in Register

DESCRIPTION: Bit number N($0 \leq N \leq 15$) of the register RB is set to zero where the least significant four bits of the contents of register RA is N. Bits $(RA)_{0-11}$ have no effect on the operation. If RA = RB, then the count is determined first and then the appropriate bit is changed.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode			Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	
RVBR RA,RB (CLRI Ra,Rb)	R	5C	RA	RB	07,01,00

Register Transfer Description:
$(RB)_N \leftarrow 0$ where $N = (RA)_{12-15}$;

Registers Affected:
RB

TVBR

Test Variable Bit in Register

DESCRIPTION: Bit number N($0 \leq N \leq 15$) of the register RB is tested where the least significant four bits of the contents of register RA is N. The Condition Status, CS, is then set to indicate non-zero if bit number N of register RB is a one. Otherwise, CS is set to indicate zero.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode			Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	
TVBR RA,RB (TESTI Ra,Rb)	R	5E	RA	RB	07,01,00

Register Transfer Description:

$N = (RA)_{12-15}$

$(CS) \leftarrow 0010$ if $(RB_N) = 0$ and $0 \leq N \leq 15$;
$(CS) \leftarrow 0001$ if $(RB_N) = 1$ and $N = 0$;
$(CS) \leftarrow 0100$ if $(RB_N) = 1$ and $1 \leq N \leq 15$;

Registers Affected:

CS

Shift Left Logical

DESCRIPTION: The contents of the Derived Address, DA (i.e., the contents of register RB) are shifted left logically N positions. The shifted result is stored in RB. The logical shift left operation is as follows: zeros enter the least significant bit position (bit 15) and bits shifted out of the sign bit position (bit 0) are lost. The condition status, CS, is set based on the result in register RB.

Note: N − 1 = 0 represents a shift of one positon.

N − 1 = 15 represents a shift of sixteen positions.

Example:

	0			15
RB Before Shift	sabc	defg	hijk	lmnp
RB After Shift (N = 4)	defg	hijk	lmnp	0000

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4		
SLL RB,N (SHL #data,Rb)	R	60	N − 1	RB	1 ≤ N ≤ 16	3N + 4,01,00

Register Transfer Description:
(RB) ← (RB) Shifted left logically by N positions;

(CS) ← 0010 if (RB) = 0 ;
(CS) ← 0001 if (RB) < 0 ;
(CS) ← 0100 if (RB) > 0 ;

Registers Affected:
RB, CS

SRL

Shift Right Logical

DESCRIPTION: The contents of the Derived Address, DA (i.e., the contents of register RB), are shifted right logically N positions. The shifted result is stored in RB. The logical shift right operation is as follows: zeros enter the sign bit position (bit 0) and bits shifted out of the least significant bit position (bit 15) are lost. The condition status, CS, is set based on the result in register RB.

Note: N − 1 = 0 represents a shift of one position.

N − 1 = 15 represents a shift of sixteen positions.

Example:

	0			15
RB Before Shift	sabc	defg	hijk	lmnp
RB After Shift (N=4)	0000	sabc	defg	hijk

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
			8	4	4		
SRL (SHR	RB,N #data,Rb)	R	61	N−1	RB	1 ≤ N ≤ 16	3N+4,01,00

Register Transfer Description:
(RB) ← (RB) Shifted right logically by N positions;

(CS) ← 0010 if (RB) = 0 ;
(CS) ← 0001 if (RB) < 0 ;
(CS) ← 0100 if (RB) > 0 ;

Registers Affected:
RB, CS

SRA

Shift Right Arithmetic

DESCRIPTION: The contents of the Derived Address, DA (i.e., the contents of register RB), are shifted right arithmetically N positions. The shifted result is stored in RB. The arithmetic right shift operation is as follows: the sign bit, which is not changed, is copied into the next positon for each position shifted and bits shifted out of the least significant bit position (bit 15) are lost. The condition status, CS, is set based on the result in register RB.

Note: N − 1 = 0 represents a shift of one positon.

N − 1 = 15 represents a shift of sixteen positions.

Example:

	0			15
RB Before Shift	sabc	defg	hijk	lmnp
RB After Shift (N = 4)	ssss	sabc	defg	hijk

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4		
SRA RB,N (SHRA #data,Rb)	R	62	N − 1	RB	1 ≤ N ≤ 16	3N + 4,01,00

Register Transfer Description:
(RB) : (RB) Shifted right arithmetically by N positions;

(CS) ← 0010 if (RB) = 0 ;
(CS) ← 0001 if (RB) < 0 ;
(CS) ← 0100 if (RB) > 0 ;

Registers Affected:
RB, CS

SLC

Shift Left Cyclic

DESCRIPTION: The contents of the Derived Address, DA (i.e., the contents of register RB), are shifted left cyclically N positions. The shifted result is stored in RB. The cyclic left shift operation is as follows: bits shifted out of the sign bit position (bit 0) enter the least significant bit position (bit 15) and, consequently, no bits are lost. The conditions status, CS, is set based on the result in RB.

Note: N − 1 = 0 represents a shift of one position.

N − 1 = 15 represents a shift of sixteen positions.

Example:

	0			15
RB Before Shift	sabc	defg	hijk	lmnp
RB After Shift (N = 4)	defg	hijk	lmnp	sabc

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
			8	4	4		
SLC (ROL	RB,N #data,Rb)	R	63	N − 1	RB	1 ≤ N ≤ 16	3N + 4,01,00

Register Transfer Description:
(RB) ← (RB) Shifted left cyclically by N positions;

(CS) ← 0010 if (RB) = 0 ;
(CS) ← 0001 if (RB) < 0 ;
(CS) ← 0100 if (RB) > 0 ;

Registers Affected:
RB, CS

DSLL

Double Shift Left Logical

DESCRIPTION: The concatenated contents of the Derived Address, DA, and DA+1 (i.e. the concatenated contents of RB and RB+1), are shifted left logically N positions. The shifted results are stored in RB and RB+1. The double left shift logical operation is as follows: zeros enter the least significant bit position of RB+1, bits shifted out of the sign bit position of RB+1 enter the least significant bit of RB and bits shifted out of the sign bit position of RB are lost. The condition status, CS, is set based on the result in registers RB and RB+1.

Note: N−1 = 0 represents a shift of one position.

N−1 = 15 represents a shift of sixteen positions.

Example:

RB, RB+1 Before Shift

0	RB		15	0	RB+1		15
$s_1$abc	defg	hijk	lmnp	$s_2$qrs	tuvw	xyzz	zzzz

RB, RB+1 After Shift (N=4)

0	RB		15	0	RB+1		15
defg	hijk	lmnp	$s_2$qrs	tuvw	xyzz	zzzz	0000

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4		
DSLL RB,N (SHLL #data,Rb)	R	65	N−1	RB	$1 \leq N \leq 16$	6N+10,01,00

A5

Register Transfer Description:

(RB,RB+1) ← (RB,RB+1) Shifted left logically by N positions;

(CS) ← 0010 if (RB,RB+1) = 0 ;
(CS) ← 0001 if (RB,RB+1) < 0 ;
(CS) ← 0100 if (RB,RB+1) > 0 ;

Registers Affected:

RB, RB+1, CS

DSRL

Double Shift Right Logical

DESCRIPTION: The concatenated contents of the Derived Address, DA, and DA + 1 (i.e. the concatenated contents of RB and RB + 1), are shifted right logically N positions. The shifted results are stored in RB and RB + 1. The double logical right shift operation is as follows: zeros enter the sign bit positon of RB, bits shifted out of the least significant bit position of RB enter the sign bit position of RB + 1 and bits shifted out of the least significant bit position of RB + 1 are lost. The condition status, CS, is set based on the result in registers RB and RB + 1.

Note: N − 1 = 0 represents a shift of one position.

N − 1 = 15 represents a shift of sixteen positions.

Example:

RB, RB + 1 Before Shift

0	RB		15	0	RB + 1		15
$s_1$abc	defg	hijk	lmnp	$s_2$qrs	tuvw	xyzz	zzzz

RB, RB + 1 After Shift (N = 4)

0	RB		15	0	RB + 1		15
0000	$s_1$abc	defg	hijk	lmnp	$s_2$qrs	tuvw	xyzz

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4		
DSRL RB,N (SHRL #data,Rb)	R	66	N − 1	RB	1 ≤ N ≤ 16	6N + 10,01,00

Register Transfer Description:

(RB,RB + 1) ← (RB,RB + 1) Shifted right logically by N positions;

(CS) ← 0010 if (RB,RB + 1) = 0 ;
(CS) ← 0001 if (RB,RB + 1) < 0 ;
(CS) ← 0100 if (RB,RB + 1) > 0 ;

Registers Affected:

RB, RB + 1, CS

DSRA

Double Shift Right Arithmetic

DESCRIPTION: The concatenated contents of the Derived Address, DA, and DA + 1 (i.e. the concatenated contents of RB and RB + 1), are shifted right arithmetically N positions. The shifted results are stored in RB and RB + 1. The double right shift arithmetic operation is as follows: the sign bit of RB, which is not changed, is copied into the next position for each position shifted, bits shifted out of the least significant position of RB enter the sign bit position of RB + 1, and bits shifted out of the least significant bit position of RB + 1 are lost. The condition status, CS, is set based on the result in register RB and RB + 1.

Note: N − 1 = 0 represents a shift of one position.

N − 1 = 15 represents a shift of sixteen positions.

Example:

RB, RB + 1 Before Shift

0	RB		15	0	RB+1		15
s_1abc	defg	hijk	lmnp	s_2qrs	tuvw	xyzz	zzzz

RB, RB + 1 After Shift (N = 4)

0	RB		15	0	RB+1		15
s1s1s1s1	s_1abc	defg	hijk	lmnp	s_2qrs	tuvw	xyzz

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4		
DSRA RB,N (SHRAL #data,Rb)	R	67	N − 1	RB	$1 \leq N \leq 16$	6N + 10,01,00

Register Transfer Description:

(RB,RB + 1) ← (RB,RB + 1) Shifted right arithmetically by N positions;

(CS) ← 0010 if (RB,RB + 1) = 0 ;
(CS) ← 0001 if (RB,RB + 1) < 0 ;
(CS) ← 0100 if (RB,RB + 1) > 0 ;

Registers Affected:

RB, RB + 1, CS

DSLC

Double Shift Left Cyclic

DESCRIPTION: The concatenated contents of the Derived Address, DA, and DA + 1 (i.e. the concatenated contents of RB and RB + 1), are shifted left cyclically N positions. The shifted results are stored in RB and RB + 1. The double left shift cyclic operation is as follows: bits shifted out of the sign bit position of RB enter the least significant bit position of RB + 1, bits shifted out of the sign bit position of RB + 1 enter the least significant bit position of RB, and consequently no bits are lost. The condition status, CS, is set based on the result in RB and RB + 1.

Note: N − 1 = 0 represents a shift of one position.

N − 1 = 15 represents a shift of sixteen positions.

Example:

RB, RB + 1 Before Shift

0	RB		15
$s_1$abc	defg	hijk	lmnp

0	RB + 1		15
$s_2$qrs	tuvw	xyzz	zzzz

RB, RB + 1 After Shift (N = 4)

0	RB		15
defg	hijk	lmnp	$s_2$qrs

0	RB + 1		15
tuvw	xyzz	zzzz	$s_1$abc

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode		Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
DSLC RB,N (ROLL #data,Rb)	R		1 ≤ N ≤ 16	9N + 10,01,00

8	4	4
68	N − 1	RB

Register Transfer Description:

(RB,RB + 1) ← (RB,RB + 1) Shifted left cyclically by N positions;

(CS) ← 0010 if (RB,RB + 1) = 0 ;
(CS) ← 0001 if (RB,RB + 1) < 0 ;
(CS) ← 0100 if (RB,RB + 1) > 0 ;

Registers Affected:

RB, RB + 1, CS

SLR

Shift Logical, Count in Register

DESCRIPTION: The contents of register RA are shifted logically N positions, where N is the contents of register RB. If N is positive $((RB_0)=0)$, then the shift direction is left; if N is negative (2's complement notation, $(RB_0)=1$), then the shift direction is right. The condition status, CS, is set based on the result in RA.

Note: N = 0 represents a shift of zero positions.

If $|N|>16$, the fixed point overflow occurs, no shifting takes place, and this instruction is treated as a NOP. (See NOP)

The contents of RB remain unchanged, unless RA = RB; in this event the contents are shifted N positions.

(See "Description" of the logical shift instructions, SLL and SRL for the definition of shift operations.)

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4		
SLR RA,RB (SHLR Rb,Ra)	R	6A	RA	RB	if $((RB_0)=0)$ $\lvert(RB)\rvert \le 16$	5N+33,01,00s
					if $((RB_0)=1)$	3N+18,01,00
					if N=0	11,01,00

Register Transfer Description:
$PI_4 \leftarrow 1$, exit, if $|N| > 16$;

(RA) ← (RA) Shifted left logically by (RB) positions.
 if $0 < (RB) \le 16$;

(RA) ← (RA) Shifted right logically by $-(RB)$ positions.
 if $0 > (RB) \ge -16$;

(CS) ← 0010 if (RA) = 0 ;
(CS) ← 0001 if (RA) < 0 ;
(CS) ← 0100 if (RA) > 0 ;

Registers Affected:
RA, RB, CS, PI

SAR

Shift Arithmetic, Count in Register

DESCRIPTION: The contents of register RA are shifted arithmetically N positions, where N is the contents of register RB. If N is positive ((RB_0)=0), then the shift direction is left; if N is negative (2's complement notation, (RB_0)=1), then the shift direction is right. The condition status, CS, is set based on the result in RA.

Note: N = 0 represents a shift of zero positions.

If |N|>16, a fixed point overflow occurs, no shifting takes place, and this instruction is treated as a NOP. (See NOP)

The contents of RB remain unchanged, unless RA = RB; in this event, the contents are shifted N positions.

(See "Description" of the arithmetic shift instruction, SRA for definition of the right shift operation. Left shift causes "zeros" to be shifted into low order position of result.)

Fixed point overflow occurs if the sign bit changes during a left shift.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode			Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
SAR RA,RB (SHLRA Rb,Ra)	R	8 | 4 | 4 6B | RA | RB	|(RB)| ≤ 16	if ((RB_0)=0)	5N+24,01,00
				if ((RB_0)=1)	3N+18,01,00
				if N=0	11,01,00

Register Transfer Description:

$PI_4 \leftarrow 1$, exit, if $|N| > 16$;

(RA) ← (RA) Shifted left arithmetically (RB) positions.
if $16 \geq (RB) \geq 0$;

(RA) ← (RA) Shifted right arithmetically −(RB) positions.
if $0 > (RB) \geq -16$;

$PI_4 \leftarrow 1$, if (RA_0) changes during the shift.

(CS) ← 0010 if (RA) = 0 ;
(CS) ← 0001 if (RA) < 0 ;
(CS) ← 0100 if (RA) > 0 ;

Registers Affected:
RA, RB, CS, PI

SCR

Shift Cyclic, Count in Register

DESCRIPTION: The contents of register RA are shifted cyclically N positions, where N is the contents of register RB. If N is positive $((RB_0)=0)$, then the shift direction is left; if N is negative (2's complement notation, $((RB_0)=1)$, then the shift direction is right. The condition status, CS, is set based on the result in RA.

Note: N = 0 represents a shift of zero positions.

If $|N| > 16$, the fixed point overflow occurs, no shifting takes place, and this instruction is treated as a NOP. (See NOP)

(See "Description" of the cyclic shift instruction, SLC, for definition of shift operations.)

The contents of RB remain unchanged, unless RA = RB; in this event the contents are shifted N positions.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode		Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
SCR RA,RB (ROLR Rb,Ra)	R	8 4 4 6C RA RB	if $((RB_0)=0)$	3N+21,01,00
			$\lvert(RB)\rvert \le 16$	
			if $((RB_0)=1)$	3N+18,01,00
			if N=0	11,01,00

Register Transfer Description:

$PI_4 \leftarrow 1$, exit, if $|N| > 16$

(RA) ← (RA) Shifted left cyclically by (RB) positions.
if $0 < (RB) \le 16$;

(RA) ← (RA) Shifted right cyclically by −(RB) positions.
if $0 > (RB) \ge -16$;

(CS) ← 0010 if (RA) = 0 ;
(CS) ← 0001 if (RA) < 0 ;
(CS) ← 0100 if (RA) > 0 ;

Registers Affected:
RA, RB, CS, PI

Double Shift Logical, Count in Register

DESCRIPTION: The concatenated contents of register RA and RA+1 are shifted logically N positions where register RB contains the count, N. If the count is positive $((RB_0)=0)$, then the shift direction is left. If the count is negative (2's complement notation, $(RB_0)=1$), then the shift direction is right. The condition status, CS, is set based on the result in RA and RA+1.

Note: N = 0 represents a shift of zero positions.

If $|N|>32$, a fixed point overflow occurs, no shifting takes place, and this instruction is treated as a NOP. (See NOP)

(See "Description" of the double shift logical instructions, DSRL and DSLL, for definition of shift operations.)

The contents of RB remain unchanged, unless RA = RB; in this event, the contents are shifted N positions.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4		
DSLR RA,RB (SHLRL Rb,Ra)	R	6D	RA	RB	if $((RB_0)=0)$ $\lvert(RB)\rvert \leq 32$	8N+36,01,00
					if $((RB_0)=1)$	6N+24,01,00
					if N=0	11,01,00

Register Transfer Description:
$PI_4 \leftarrow 1$, exit, if $|N| > 32$;

$(RA,RA+1) \leftarrow (RA,RA+1)$ Shifted left logically by (RB) positions.
if $32 \geq (RB) \geq 0$;

$(RA,RA+1) \leftarrow (RA,RA+1)$ Shifted right logically by $-(RB)$ positions.
if $0 > (RB) \geq -32$;

$(CS) \leftarrow 0010$ if $(RA,RA+1) = 0$;
$(CS) \leftarrow 0001$ if $(RA,RA+1) < 0$;
$(CS) \leftarrow 0100$ if $(RA,RA+1) > 0$;

Registers Affected:
RA, RA+1, RB, CS, PI

Double Shift Arithmetic, Count in Register

DESCRIPTION: The concatenated contents of register RA and RA+1 are shifted arithmetically N positions where register RB contains the count, N. If the count is positive $((RB_0)=0)$, then the shift direction is left. If the count is negative (2's complement notation, $(RB_0)=1$), then the shift direction is right. The condition status, CS, is set based on the result in RA and RA+1.

Note: N = 0 represents a shift of zero positions.

If |N|>32, a fixed point overflow occurs, no shifting occurs, and this instruction is treated as a NOP. (See NOP)

The contents of RB remain unchanged, unless RA = RB; in this event the contents are shifted N positions.

(See "Description" of the double shift arithmetic instruction, DSRA, for the definition of the right shift operation. Left shift causes "zeros" to be shifted into low order position of result.)

Fixed point overflow occurs if the sign bit is changed during a left shift.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode					Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4			
DSAR RA,RB (SHLRAL Rb,Ra)	R	6E	RA	RB	$|(RB)| \leq 32$	if $((RB_0)=0)$	8N+27,01,00
						if $((RB_0)=1)$	6N+24,01,00
						if N=0	11,01,00

Register Transfer Description:

$PI_4 \leftarrow 1$, exit, if $|N| > 32$;

$(RA,RA+1) \leftarrow (RA,RA+1)$ Shifted left arithmetically by (RB) positions.
if $32 \geq (RB) > 0$;

$(RA,RA+1) \leftarrow (RA,RA+1)$ Shifted right arithmetically by $-(RB)$ positions.
if $0 > (RB) \geq -32$;

$PI_4 \leftarrow 1$, if (RA_0) changes during the shift;

$(CS) \leftarrow 0010$ if $(RA,RA+1) = 0$;
$(CS) \leftarrow 0001$ if $(RA,RA+1) < 0$;
$(CS) \leftarrow 0100$ if $(RA,RA+1) > 0$;

Registers Affected:

RA, RA+1, RB, CS, PI

DSCR

Double Shift Cyclic, Count in Register

DESCRIPTION: The concatenated contents of register RA and RA+1 are shifted cyclically N positions, where register RB contains the count, N. If the count is positive $((RB_0)=0)$, the shift direction is left. If the count is negative (2's complement notation, $(RB_0)=1$), the shift direction is right. The condition status, CS, is set based on the result in RA and RA+1.

Note: N = 0 represents a shift of zero positions.

If |N|>32, the fixed point overflow occurs, no shifting takes place, and this instruction is treated as a NOP. (See NOP)

(See "Description" of the double shift cyclic instructions, DSLC, Tfor the definition of shift operations.)

The contents of RB remain unchanged, unless RA = RB; in this event Tthe contents are shifted N positions.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4		
DSCR RA,RB (ROLRL Rb,Ra)	R	6F	RA	RB	if $((RB_0)=0)$ $\lvert(RB)\rvert \leq 32$	9N+24,01,00
					if ((RB)=1)	9N+24,01,00
					if N=0	11,01,00

Register Transfer Description:
$PI_4 \leftarrow 1$, exit, if $|N| > 32$;

(RA,RA+1) ← (RA,RA+1) Shifted left cyclically by (RB) positions.
if $32 \geq (RB) > 0$;

(RA,RA+1) ← (RA,RA+1) Shifted right cyclically by −(RB) positions.
if $0 > (RB) \geq -32$;

(CS) ← 0010 if (RA,RA+1) = 0 ;
(CS) ← 0001 if (RA,RA+1) < 0 ;
(CS) ← 0100 if (RA,RA+1) > 0 ;

Registers Affected:
RA, RA+1, RB, CS, PI

Load/Store/Exchange

L, LB, LBX, LI, LIM, LISN, LISP, LR

Single Precision Load

DESCRIPTION: The single precision Derived Operand, DO, is loaded into the register RA. The Condition Status, CS, is set based on the result in register RA.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode		Clock Cycles*, Instruction Fetch Bus Cycles, Operand Bus Cycles
L (LD	RA,ADDR addr,Ra)	D	8 · 4 · 4 80 · RA · RX	16 ADDR	12,02,01
L (LD	RA,ADDR,RX addr(Rx),Ra)	DX			12,02,01
LB (LD	BR,DSPL dspl(Br), R2)	B	4 · 2 · 2 · 8 0 · 0 · BR′ · DSPL	$12 \leq BR \leq 15$ $BR' = BR - 12$ $RA = R2$	11,01,01
LBX (LD	BR,RX (Rx)(Br),R2)	BX	4 · 2 · 2 · 4 · 4 4 · 0 · BR′ · 0 · RX	$12 \leq BR \leq 15$ $BR' = BR - 12$ $RA = R2$	11,01,01
LI (LD	RA,ADDR @addr,Ra)	I	8 · 4 · 4 84 · RA · RX	16 ADDR	16,02,02
LI (LD	RA,ADDR,RX Addr(Rx)@,Ra)	IX			16,02,02
LIM (LD	RA,DATA #data,Ra)	IM	8 · 4 · 4 85 · RA · RX	16 DATA	11,02,00
LIM (LD	RA,DATA,RX #data(Rx),Ra)	IMX			14,02,00
LISN (LD4	RA,N #data,Ra)	ISN	8 · 4 · 4 83 · RA · N−1	$1 \leq N \leq 16$	07,01,00
LISP (LD4	RA,N #data,Ra)	ISP	8 · 4 · 4 82 · RA · N−1	$1 \leq N \leq 16$	07,01,00
LR (MOV	RA,RB Rb,Ra)	R	8 · 4 · 4 81 · RA · RB		04,01,00

Register Transfer Description:
(RA) ← DO;

(CS) ← 0010 if (RA) = 0 ;
(CS) ← 0001 if (RA) < 0 ;
(CS) ← 0100 if (RA) > 0 ;

Registers Affected:
RA, CS

Double Precision Load

DESCRIPTION: The double precision Derived Operand, DO, is loaded into the register RA and RA+1 such that the MSH of DO is in RA. The Condition Status, CS, is set based on the result in RA and RA+1.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode						Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
DL (LDL DL (LDL	RA,ADDR addr,Ra) RA,ADDR,RX addr(Rx),Ra)	D DX	8 86	4 RA	4 RX			16 ADDR	22,02,02 22,02,02
DLB (LDL	BR,DSPL dspl(Br),R0)	B	4 0	2 1	2 BR′	8 DSPL		12 ≤ BR ≤ 15 BR′ = BR − 12 RA = R0	21,01,02
DLBX (LDL	BR,RX (Rx)(Br),R0)	BX	4 4	2 0	2 BR′	4 1	4 RX	12 ≤ BR ≤ 15 BR′ = BR − 12 RA = R0	21,01,02
DLI (LDL DLI (LDL	RA,ADDR @addr,Ra) RA,ADDR,RX addr(Rx)@,Ra)	I IX	8 88	4 RA	4 RX			16 ADDR	26,02,03 26,02,03
DLR (MOVL	RA,RB Rb,Ra)	R	8 87	4 RA	4 RB				16,01,00

Register Transfer Description:
(RA,RA+1) ← DO;

(CS) ← 0010 if (RA,RA+1) = 0 ; (Double fixed point zero);
(CS) ← 0001 if (RA,RA+1) < 0 ;
(CS) ← 0100 if (RA,RA+1) > 0 ;

Registers Affected:
RA, RA+1, CS

EFL

Extended Precision Floating Point Load

DESCRIPTION: The extended precision floating point Derived Operand, DO, is loaded into registers RA, RA+1 and RA+2 such that the most significant 16-bits of the word are loaded into register RA. The condition status, CS, is set based on the results in registers RA, RA+1 and RA+2.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
			8	4	4	16	
EFL (LDX	RA,ADDR addr,Ra)	D	8A	RA	RX	ADDR	26,02,03
EFL (LDX	RA,ADDR,RX addr(Rx),Ra)	DX					26,02,03

Register Transfer Description:
(RA, RA+1, RA+2) ← DO;

(CS) ← 0010 if (RA, RA+1, RA+2) = 0 ;
(CS) ← 0001 if (RA, RA+1, RA+2) < 0 ;
(CS) ← 0100 if (RA, RA+1, RA+2) > 0 ;

Registers Affected:
RA, RA+1, RA+2, CS

LUB, LUBI

Load From Upper Byte

DESCRIPTION: The MSH (upper byte) of the Derived Operand, DO, is loaded into the LSH (lower byte) of register RA. The MSH (upper byte) of RA is unaffected. The condition status, CS, is set based on the result in RA.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
LUB	RA,ADDR	D	8	4	4	16	15,02,01
(LDUB	addr,Ra)		8B	RA	RX	ADDR	
LUB	RA,ADDR,RX	DX					15,02,01
(LDUB	addr(Rx),Ra)						
LUBI	RA,ADDR	I	8	4	4	16	19,02,02
(LDUB	@addr,Ra)		8D	RA	RX	ADDR	
LUBI	RA,ADDR,RX	IX					19,02,02
(LDUB	addr(Rx)@,Ra)						

Register Transfer Description:

$(RA)_{8-15} \leftarrow DO_{0-7}$;

$(CS) \leftarrow 0010$ if $(RA) = 0$;
$(CS) \leftarrow 0001$ if $(RA) < 0$;
$(CS) \leftarrow 0100$ if $(RA) > 0$;

Registers Affected:

RA, CS

Load/Store/Exchange

LLB, LLBI

Load From Lower Byte

DESCRIPTION: The LSH (lower byte) of the Derived Operand, DO, is loaded into the LSH (lower byte) of register RA. The MSH (upper byte) of RA is unaffected. The condition status, CS, is set based on the result in RA.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
LLB RA,ADDR (LDLB addr,Ra)	D	8	4	4	16	12,02,01
LLB RA,ADDR,RX (LDLB addr(Rx),Ra)	DX	8C	RA	RX	ADDR	12,02,01
LLBI RA,ADDR (LDLB @addr,Ra)	I	8	4	4	16	16,02,02
LLBI RA,ADDR,RX (LDLB addr(Rx)@,Ra)	IX	8E	RA	RX	ADDR	16,02,02

Register Transfer Description:

$(RA)_{8-15} \leftarrow DO_{8-15}$;

$(CS) \leftarrow 0010$ if $(RA) = 0$;
$(CS) \leftarrow 0001$ if $(RA) < 0$;
$(CS) \leftarrow 0100$ if $(RA) > 0$;

Registers Affected:

RA, CS

Load/Store/Exchange

ST, STB, STBX, STI

Single Precision Store

DESCRIPTION: The contents of the register RA are stored into the Derived Address, DA.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode		Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
ST (ST	RA,ADDR Ra,addr)	D	8 4 4 90 RA RX	16 ADDR	12,02,01
ST (ST	RA,ADDR,RX Ra,addr(Rx))	DX			12,02,01
STB (ST	BR,DSPL R2,dspl(Br))	B	4 2 2 8 0 2 BR′ DSPL	12 ≤ BR ≤ 15 BR′ = BR − 12 RA = R2	11,01,01
STBX (ST	BR,RX R2,(RX)(B))	BX	4 2 2 4 4 4 0 BR′ 2 RX	12 ≤ BR ≤ 15 BR′ = BR − 12 RA = R2	11,01,01
STI (ST	RA,ADDR Ra,@addr)	I	8 4 4 94 RA RX	16 ADDR	16,02,02
STI (ST	RA,ADDR,RX Ra,addr(Rx)@)	IX			16,02,02

Register Transfer Description:

[DA] ← (RA);

Registers Affected:

None

STC, STCI, STZ, STZI

Store A Non-Negative Constant
DESCRIPTION: The constant N, where N is an integer(0 ≤ N ≤ 15) is stored at the Derived Address, DA. For the special case of storing zero into memory the mnemonics

STZ ADDR, RX for direct addressing
and STZI ADDR, RX for indirect addressing

may be used. In this special case, the N field equals 0.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	16	
STC N,ADDR (MOV4 #data,addr)	D	91	N	RX	ADDR	12,02,01
STC N,ADDR,RX (MOV4 #data,addr(Rx))	DX					12,02,01
		8	4	4	16	
STCI N,ADDR (MOV4 #data,@addr)	I	92	N	RX	ADDR	16,02,02
STCI N,ADDR,RX (MOV4 #data,addr(Rx)@)	IX					16,02,02

Register Transfer Description:
[DA] ← N, where 0 ≤ N ≤ 15;

Registers Affected:
None

Double Precision Store

DESCRIPTION: The contents of registers RA and RA+1 are stored at the Derived Address, DA, and DA+1, respectively.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode		Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
DST RA,ADDR (STL Ra,addr)	D	8 4 4 96 RA RX	16 ADDR	16,02,02
DST RA,ADDR,RX (STL Ra,addr(Rx))	DX			16,02,02
DSTB BR,DSPL (STL R0,dspl(Br))	B	4 2 2 8 0 3 BR′ DSPL	12 ≤ BR ≤ 15 BR′ = BR − 12 RA = R0	15,01,02
DSTI RA,ADDR (STL Ra,@addr)	I	8 4 4 98 RA RX	16 ADDR	20,02,03
DSTI RA,ADDR,RX (STL Ra,addr(Rx)@)	IX			20,02,03
DSTX BR,RX (STL R0,(Rx)(Br))	BX	4 2 2 4 4 4 0 BR′ 3 RX	12 ≤ BR ≤ 15 BR′ = BR − 12 RA = R0	15,01,02

Register Transfer Description:
[DA,DA+1] ← (RA,RA+1);

Registers Affected:
None

SRM

Store Register Through Mask

DESCRIPTION: The contents of register RA are stored into the Derived Address, DA, through the mask in register RA+1. For each position in the mask that is a one, the corresponding bit of register RA is stored into the corresponding bit of the DA. For each position in the mask that is a zero no change is made to the corresponding bit stored in the DA.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
			8	4	4	16	
SRM (STMI	RA,ADDR Ra, addr)	D	97	RA	RX	ADDR	25,02,02
SRM (STMI	RA,ADDR,RX Ra,addr(Rx))	DX					25,02,02

Register Transfer Description:

$[DA] \leftarrow \{[DA] \uparrow (\overline{RA+1})\} \vee \{[RA] \uparrow [RA+1]\};$

$(RA+1) = MASK, (RA) = DATA;$

or, equivalently,

$(RQ) \leftarrow [DA];$

$(RQ)_i \leftarrow (RA)_i$ if $(RA+1)_i = 1$ for $i = 0, 1, \ldots, 15;$

$[DA] \leftarrow (RQ);$

Registers Affected:

None

EFST

Extended Precision Floating Point Store

DESCRIPTION: The contents of registers RA, RA+1, RA+2 are stored at the Derived Address, DA, DA+1, and DA+2.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
EFST RA,ADDR (STX Ra, addr)	D	8	4	4	16	20,02,03
EFST RA,ADDR,RX (STX Ra,addr(Rx))	DX	9A	RA	RX	ADDR	20,02,03

Register Transfer Description:

[DA, DA+1, DA+2] ← (RA, RA+1, RA+2);

Registers Affected:

None

Load/Store/Exchange

STUB, SUB

Store into Upper Byte

DESCRIPTION: The LSH (lower byte) of register RA is stored into the MSH (upper byte) of the Derived Address, DA. The LSH (lower byte) of the DA is unchanged.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
STUB	RA,ADDR	D	8	4	4	16	16,02,02
(STUB	Ra,addr)		9B	RA	RX	ADDR	
STUB	RA,ADDR,RX	DX					16,02,02
(STUB	Ra,addr(Rx))						
SUBI	RA,ADDRR	I	8	4	4	16	20,02,03
(STUB	Ra,@addr)		9D	RA	RX	ADDR	
SUBI	RA,ADDR,RX	IX					20,02,03
(STUB	Ra,addr(Rx)@)						

Register Transfer Description:
$[DA]_{0-7} \leftarrow (RA)_{8-15}$;

Registers Affected:
None

SLBI, STLB

Store into Lower Byte

DESCRIPTION: The LSH (lower byte) of register RA is stored into the LSH (lower byte) of the Derived Address, DA. The MSH (upper byte) of the DA is unchanged.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
STLB (STLB	RA,ADDR Ra,addr)	D	8	4	4	16	16,02,02
STLB (STLB	RA,ADDR,RX Ra,addr(Rx))	DX	9C	RA	RX	ADDR	16,02,02
SLBI (STLB	RA,ADDR Ra,@addr)	I	8	4	4	16	20,02,03
SLBI (STLB	RA,ADDR,RX Ra,addr(Rx)@)	IX	9E	RA	RX	ADDR	20,02,03

Register Transfer Description:
$[DA]_{8-15} \leftarrow (RA)_{8-15};$

Registers Affected:
None

XBR

Exchange Bytes in Register

DESCRIPTION: The upper byte of register RA is exchanged with the lower byte of register RA. The CS is set based on the result in register RA.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode			Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	
XBR RA (XCHB Ra)	S	EC	RA	0	07,01,00

Register Transfer Description:

$(RA)_{0-7} \leftrightarrow (RA)_{8-15};$

$(CS) \leftarrow 0010$ if $(RA) = 0$;
$(CS) \leftarrow 0001$ if $(RA) < 0$;
$(CS) \leftarrow 0100$ if $(RA) > 0$;

Registers Affected:
RA, CS

XWR

Exchange Words in Registers

DESCRIPTION: The contents of register RA are exchanged with the contents of register RB. The CS is set based on the result in register RA.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode			Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	
XWR RA,RB (XCH Rb,Ra)	R	ED	RA	RB	10,01,00

Register Transfer Description:

(RA) ↔ (RB);

(CS) ← 0010 if (RA) = 0 ;
(CS) ← 0001 if (RA) < 0 ;
(CS) ← 0100 if (RA) > 0 ;

Registers Affected:

RA, RB, CS

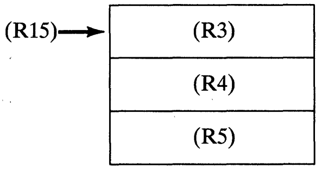
Push Multiple Registers onto the Stack

DESCRIPTION: For RA < RB, the contents of RB through RA are pushed onto a stack in memory using R15 as the stack pointer. As each register contents are pushed onto the memory stack, R15 is decremented by one word for each word pushed. On completion, R15 points to the last item on the stack, the contents of RA.

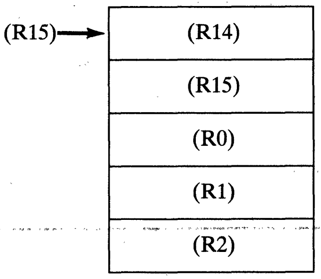
For RA > RB, the contents of RB through R0, and then the contents of R15 through RA, are pushed onto the stack. On completion, R15 points to the last item on the stack, the contents of RA.

In both cases, successive increasing addresses on the stack correspond to successive increasing register addresses, with a point discontinuity between R15 and R0 in the latter case.

Example: PSHM R3,R5 results in



PSHM R14,R2 results in



Register Transfer Description:

```
if RA ≤ RB then
for i = 0 thru RB − RA do
  begin
  (R15) ← (R15) − 1;
  [(R15)] ← (RB) − i);
  end;
else
  begin
  for i = 0 thru RB do
  begin
  (R15) ← (R15) − 1;
  [(R15)] ← (RB − i);
  end;
for i = 0 thru 15 − RA do
  begin
  (R15) ← (R15) − 1;
  [(R15)] ← (R15 − i);
  end;
end;
```

Registers Affected:
R15

POPM

Pop Multiple Registers Off the Stack

DESCRIPTION: For RA ≤ RB, registers RA through RB are loaded sequentially from a stack in memory using R15 as the stack pointer.

For RA > RB, registers RA through R14 and then R0 through RB are loaded sequentially from the stack.

In both cases,

a. as each word is popped from the stack, R15 is incremented by one;

b. if R15 is included in the transfer, then it is effectively ignored;

c. on completion, R15 points to top word of the stack remaining.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode			Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	
POPM RA,RB (POPM @R15,Ra,Rb)	S	8F	RA	RB	16N*+4,01,N*

*N = Number of registers manipulated.

Register Transfer Description:

```
if RA ≤ RB then
for i = 0 thru RB − RA do
   begin
   if RA + i ≠ 15 then (RA + i) ← [(R15)];
   (R15) ← (R15) + 1;
   end;
else
   begin
   for i = 0 thru 15-RA do
   begin
   if RA + i ≠ 15 then (RA + i) ← [(R15)];
   (R15) ← (R15) + 1;
   end;
for i = 0 thru RB do
   begin
   (i) ← [(R15)];
   (R15) ← (R15) + 1;
   end;
end;
```

Registers Affected:
RA through R14, R0 through RB, R15

Multiple Load/Store

LM

Load Multiple Registers

DESCRIPTION: The contents of the Derived Address, DA, are loaded into register R0, then the contents of the DA+1 are loaded into register R1,..., finally, the contents of DA+N are loaded into RN. Effectively, this instruction allows the transfer of (N + 1) words from memory to the register file.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	16	
LM N,ADDR (LDM #data,addr,R0)	D	89	N	RX	ADDR	8N*+8,02,N*
LM N,ADDR,RX (LDM #data,addr(Rx),R0)	DX	0 ≤ N ≤ 15				8N*+8,02,N*

*N = Number of words manipulated.

Register Transfer Description:

(R0) ← [DA];

(R1) ← [DA+1];

(R2) ← [DA+2];

.

.

.

(RN) ← [DA+N];

Registers Affected:

R0 through RN

Store Multiple Registers

DESCRIPTION: The contents of register R0 are stored into the Derived Address, DA; then the contents of R1 are stored into DA+1;...; finally the contents of RN are stored into DA+N where N is an integer, $0 \leq N \leq 15$. Effectively, this instruction allows the transfer of (N+1) words from the register file to memory.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode	Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
STM N,ADDR (STM #data,R0,addr)	D		9N*+8,02,N*
STM N,ADDR,RX (STM #data,R0,addr(Rx))	DX		9N*+8,02,N*

8	4	4	16
99	N	RX	ADDR

*N = Number of words manipulated.

Register Transfer Description:

$[DA] \leftarrow (R0);$

$[DA+1] \leftarrow (R1);$

$[DA+2] \leftarrow (R2);$

.

.

.

$[DA+N] \leftarrow (RN)\ 0 \leq N \leq 15;$

Registers Affected:
None

JC, JCI

Jump On Condition

DESCRIPTION: This is a conditional jump instruction wherein the instruction sequence jumps to the Derived Address, DA, if a logical one results from the following operation:

(1) The 4-bit C field is bit-by-bit ANDed with the 4-bit condition status, CS

(2) The resulting 4-bits are ORed together

(3) or if C = 7 or C = F.

Otherwise, the next sequential instruction is executed.

Condition Code

C_2	C_{16}	Condition	Mil Std Mnemonic			IEEE Mnemonic
0000	0	NOP	–	–	–	NOP
0001	1	less than (zero)	LT	LZ	M	BLT,BN
0010	2	equal to (zero)	EQ	EZ	–	BE,BZ
0011	3	less than or equal to (zero)	LE	LEZ	NP	BLE,BNP
0100	4	greater than (zero)	GT	GZ	P	BGT,BP
0101	5	not equal to (zero)	NE	NZ	–	BNE,BNZ
0110	6	greater than or equal to (zero)	GE	GEZ	NM	BGE,BNN
0111	7	unconditional	–	–	–	BR,RET*
1000	8	carry	CY	–	–	BC
1001	9	carry or less than (LT)	–	–	–	BCLT,BCN
1010	A	carry or equal (EQ)	–	–	–	BCE,BCZ
1011	B	carry or less than or equal to (zero) (LE)	–	–	–	BCLE,BCNP
1100	C	carry or greater than (GT)	–	–	–	BCGT,BCP
1101	D	carry or not equal to	–	–	–	BCNE,BCNZ
1110	E	carry or greater than or equal to (zero) (GE)	–	–	–	BCGE,BCNN
1111	F	unconditional	–	–	–	BR,RET*

* RET is equivalent to JC 7,0,RX or JC 15,0,RX

JC, JCI

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
			8	4	4	16	
JC	C,LABEL	D	70	C	RX	LABEL	WO/JMP 09,02,00
(**	addr)						W/JMP 17,03,00
JC	C,LABEL,RX	DX					WO/JMP 09,02,00
(**	addr(Rx))						W/JMP 17,03,00
			8	4	4	16	
JCI	C,ADDR	I	71	C	RX	ADDR	WO/JMP 13,02,00
(**	@addr)						W/JMP 21,03,00
JCI	C,ADDR,RX	IX					WO/JMP 13,02,00
(**	Addr(Rx)@)						W/JMP 21,03,00

Register Transfer Description:

(IC) ← DA if C = 7, or
if C = F, or
if $(C_0 \uparrow CS_0) \vee (C_1 \uparrow CS_1) \vee (C_2 \uparrow CS_2) \vee (C_3 \uparrow CS_3) = 1$;

Registers Affected:

IC (if jump is executed)

** See previous page for IEEE menmonic based on value of C

Program Control

JS

Jump To Subroutine

DESCRIPTION: The value of the instruction counter (the address of the next sequential instruction) is stored into register RA. Then, the IC is set to the derived address, DA, thus effecting the jump. This sets up the return from subroutine to the address stored in the register RA, i.e., an indexed unconditional jump from location zero using RA as the index register shall transfer control to the instruction following the JS instruction.

Note: If RA = RX, then the derived address, DA, is calculated before the IC is stored in RA.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
JS	RA,LABEL	D	8	4	4	16	12,03,00
(CALL	Ra,addr)		72	RA	RX	LABEL	
JS	RA,LABEL,RX	DX					12,03,00
(CALL	Ra,addr(Rx))						

Register Transfer Description:
(RA) ← (IC);

(IC) ← DA;

Registers Affected:
RA, IC

Subtract One And Jump

DESCRIPTION: The 16-bit contents of register RA are decremented by one. Then if the content of register RA is zero, the next sequential instruction is executed. If the content of register RA is non-zero, then a jump to the Derived Address, DA, occurs.

Note: If RA = RX, then the derived address, DA, is calculated before RA is decremented.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
SOJ	RA,LABEL	D	8	4	4	16	WO/JMP 13,02,00
(DECBNZ	Ra,addr)		73	RA	RX	LABEL	W/JMP 17,03,00
SOJ	RA,LABEL,RX	DX					WO/JMP 13,02,00
(DECBNZ	Ra,addr(Rx))						W/JMP 17,03,00

Register Transfer Description:

(RA) ← (RA) − 1;

(IC) ← DA if (RA) ≠ 0;

(CS) ← 0010 if (RA) = 0;
(CS) ← 0001 if (RA) < 0;
(CS) ← 0100 if (RA) > 0;

Registers Affected:
RA, CS, IC (if the jump is executed)

BR

Branch Unconditionally

DESCRIPTION: A program branch is made to LABEL, i.e., the Derived Address, DA.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode		Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
			8	8	
BR (BR	LABEL %addr)	ICR	74	D*	14,02,00

$* -128 \leq D \leq 127$

Register Transfer Description:

(IC) ← DA;

Registers Affected:

IC

BEZ

Branch if Equal To (Zero)

DESCRIPTION: A program branch is made to LABEL, i.e., the Derived Address, DA, if the condition status, CS, indicates that the previous result set the CS is equal to (zero). Otherwise, the next sequential instruction is executed.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode		Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
			8	8	
BEZ (BE	LABEL %addr)	ICR	75	D*	WO/BRANCH 04,01,00 W/BRANCH 15,03,00

$* -128 \le D \le 127$

Register Transfer Description:
(IC) ← DA if (CS) = X010;

Registers Affected:
IC (if the jump is executed)

BLT

Branch if Less Than (Zero)

DESCRIPTION: A program branch is made to LABEL, i.e., the Derived Address, DA, if the condition status, CS, indicates that the previous result which set the CS is less than (zero). Otherwise, the next sequential instruction is executed.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode		Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	8	
BLT LABEL (BN %addr) (BLT %addr)	ICR	76	D*	WO/BRANCH 04,01,00 W/BRANCH 15,03,00

* $-128 \leq D \leq 127$

Register Transfer Description:
(IC) ← DA if (CS) = X001;

Registers Affected:
IC (if the jump is executed)

BLE

Branch if Less Than or Equal to (Zero)

DESCRIPTION: A program branch is made to LABEL, i.e., the Derived Address, DA, if the condition status, CS, indicates that the previous result which set the CS is less than or equal to (zero). Otherwise, the next sequential instruction is executed.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode		Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	8	
BLE LABEL (BLE %addr) (BNP %addr)	ICR	78	D*	WO/BRANCH 04,01,00 W/BRANCH 15,03,00

$* -128 \leq D \leq 127$

Register Transfer Description:
(IC) ← DA if (CS) = X010 or (CS) = X001;

Registers Affected:
IC (if the jump is executed)

BGT

Branch if Greater Than (Zero)

DESCRIPTION: A program branch is made to LABEL, i.e., the Derived Address, DA, if the condition status, CS, indicates that the previous result which set the CS is greater than (zero). Otherwise, the next sequential instruction is executed.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode		Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	8	
BGT LABEL (BGT %addr) (BP %addr)	ICR	79	D*	WO/BRANCH 04,01,00 W/BRANCH 15,03,00

$* -128 \leq D \leq 127$

Register Transfer Description:
(IC) ← DA if (CS) = X100;

Registers Affected:
IC (if the jump is executed)

BNZ

Branch if Not Equal To (Zero)

DESCRIPTION: A program branch is made to LABEL, i.e., the Derived Address, DA, if the condition status, CS, indicates that the previous result which set the CS is not equal to (zero). Otherwise, the next sequential instruction is executed.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode 8	 8	Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
BNZ LABEL (BNE %addr) (BNZ %addr)	ICR	7A	D*	WO/BRANCH 04,01,00 W/BRANCH 15,03,00

$* -128 \leq D \leq 127$

Register Transfer Description:
(IC) ← DA if (CS) = X100 or (CS) = X001;

Registers Affected:
IC (if the jump is executed)

BGE

Branch if Greater Than Or Equal To (Zero)

DESCRIPTION: A program branch is made to LABEL, i.e., the Derived Address, DA, if the condition status, CS, indicates that the previous result which set the CS is greater than or equal to (zero). Otherwise, the next sequential instruction is executed.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode	Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
BGE LABEL (BGE %addr) (BNN %addr)	ICR	8 | 8 7B | D* * − 128 ≤ D ≤ 127	WO/BRANCH 04,01,00 W/BRANCH 15,03,00

Register Transfer Description:
(IC) ← DA if (CS) = X100 or (CS) = X010;

Registers Affected:
IC (if the jump is executed)

BEX

Branch To Executive

DESCRIPTION: This instruction provides a means to jump to a routine in another address state, AS. It is typically used to make controlled, protected calls to an executive. The 4-bit literal N selects one of 16 executive entry points to be used. Execution of this instruction causes an interrupt to occur using the EXEC call interrupt vector (interrupt 5). The new IC is loaded from the Nth location following the SW in the new processor state. The linkage pointer (LP), service pointer (SVP), and the new processor state (new MK, new SW, and new IC) are fetched from address state zero. The current processor state (old MK, old SW, and old IC) are stored in the address state specified by the new SW AS field. Interrupts are disabled when BEX is executed. The EXEC call interrupt cannot be masked or disabled. Arguments associated with the BEX instruction are passed by software convention. The processor lock and key function is ignored when this instruction is executed. An attempt to branch into an execute protected area of memory shall result in FT_0 being set to 1.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode			Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles	
		8	4	4		
BEX N (BRK #data)	S	77	0000	N	MMU no MMU	87,02,08 92,02,08

Register Transfer Description:

(RQ,RQ+1,RQ+2) ← (MK,SW,IC);

(SVP) ← [$2B_{16}$], where AS = 0;

PI_5 ← 1;

(MK,SW,IC) ← [(SVP), (SVP)+1, (SVP)+2+N)], where AS = 0;

(LP) ← [$2A_{16}$], where AS = 0;

[(LP),(LP)+1,(LP)+2] ← (RQ,RQ+1,RQ+2), where AS = SW_{12-15};

Registers Affected:

MK, SW, IC, PI

LST, LSTI

Load Status

DESCRIPTION: The contents of the Derived Address, DA, and DA+1, and DA+2 are loaded into the Interrupt Mask register, Status Word register and Instruction Counter, respectively. This is a privileged instruction.

Note: This instruction is an unconditional jump and is typically used to exit from an interrupt routine. DA, DA+1, and DA+2, in this typical case, contain the Interrupt Mask, Status Word, and Instruction Counter values for the interrupted program and the execution of LST causes the program to return to its status prior to being interrupted.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles	
LST ADDR (LD addr,STATUS	D	8	4	4	16	W/ MMU	42,03,03
		7D	0000	RX	ADDR	W/O MMU	47,03,03
LST ADDR,RX (LD addr(Rx),STATUS)	DX					W/ MMU	42,03,03
						W/O MMU	47,03,03
LSTI ADDR (LD @addr,STATUS	I	8	4	4	16	W/ MMU	46,03,04
		7C	0000	RX	ADDR	W/O MMU	51,03,04
LSTI ADDR,RX (LD addr(Rx)@,STATUS)	IX					W/ MMU	46,03,04
						W/O MMU	51,03,04

Register Transfer Description:
(MK, SW, IC) ← [DA, DA+1, DA+2];

Registers Affected:
MK, SW, IC

Stack IC And Jump To Subroutine

DESCRIPTION: The contents of register RA are decremented by one. The address of the instruction following the SJS instruction is stored into the memory location pointed to by RA. Program control is then transferred to the instruction at the Derived Address, DA. RA is the stack pointer and can be selected by the programmer as any one of the 16 general registers.

Note: If RA = RX, then the derived address, DA, is calculated before RA is decremented.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
SJS RA,LABEL (CALL @ – RA,addr)	D	8	4	4	16	22,03,01
SJS RA,LABEL,RX (CALL @ – Ra,addr(Rx))	DX	7E	RA	RX	LABEL	22,03,01

Register Transfer Description:

(RA) ← (RA) – 1;

[(RA)] ← (IC);

(IC) ← DA;

Registers Affected:

IC, RA

URS

Unstack IC And Return From Subroutine

DESCRIPTION: The contents of the memory location pointed to by register RA is loaded into the instruction counter, IC. RA is then incremented by one. Any one of the 16 general registers may be designated as the stack pointer. This instruction is the subroutine return for SJS, Stack and Jump to Subroutine.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode			Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	
URS RA (RET @Ra+)	S	7F	RA	0	15,03,01

Register Transfer Description:

(IC) ← [(RA)];

(RA) ← (RA) + 1;

Registers Affected:

RA, IC

NOP

No Operation

DESCRIPTION: No operation is performed.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode			Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	
NOP (NOP)	S	FF	0	0	09,01,00

Register Transfer Description:
None

Registers Affected:
None

BPT

Break Point

DESCRIPTION: This instruction is typically used for halting the processor during maintenance and diagnostic procedures when the maintenance console is connected to the system. If the console is not connected, this instruction is treated as a NOP. Restarting the processor after a BPT can only be done by: the maintenance console or the power on sequence.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode			Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	
BPT (HALT)	S	FF	F	F	*27,01,01

*Assume no console

Register Transfer Description:
None

Registers Affected:
None

Built-In-Function

DESCRIPTION: This instruction invokes special operations defined by the user. Note that this instruction may use one or more additional words immediately following it, the number and interpretation of which are determined by the Op. Ex.

The F9450 implements *B*uilt-*In*-*F*unction as a three (3) word instruction. The MSH of word 0 contains 4F per 1750A (notice 1), and the format of the remaining 40 bits is as follows:

BIT#	DESCRIPTION
8	*Indicates immediate(0)/direct(1) (refering to coprocessor command)
9	*Indicates 2 word(0)/3 word (1) command (must be (1), if (0), bit 9 of FT is set)
10,11	*Indicates coprocessor number (00=0,01=1,10=2,11=3) (The F9450 will support 4 coprocessors)
12,13,14,15	*Indicates Index Register number (0 thru 16, 0=no index register)
16 → 31	coprocessor command
32 → 47	coprocessor data address

* = Bits in Op. Ex.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode							Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
BIF Op. Ex.,Addrl,Addr2	S	8	1	1	2	4	16	16	34,03,02
( )	(D)	4F	0	1	XX	0→F	Addrl	Addr2	
[BIF]	(DX)								34,03,02
	(I)								*38,03,03
[BIFI]		4F	1	1	XX	0→F	Addrl	Addr2	
	(IX)								*38,03,03

*These operations have two I/O's

BIF

Register Transfer Description:

```
  if $IR_9 = 0$, set FT9,exit
  Temp ← 0800
  $Temp_{6,7} \leftarrow IR_{10,11}$
IMMEDIATE no index reg.
  $IR_{16\rightarrow31} \rightarrow [Temp]$
DIRECT no index reg.
  $[IR_{16\rightarrow31}] \rightarrow [Temp]$
IMMEDIATE with index
  $Rx + IR_{16\rightarrow31} \rightarrow [Temp]$
DIRECT with index
  $[Rx + IR_{16\rightarrow31}] \rightarrow [Temp]$
  $Temp = Temp + 1$
  $IR_{32\rightarrow47} \rightarrow Temp$
```

Registers Affected:
User Defined

Execute Input/Output

DESCRIPTION: The input/output instruction transfers data between an external/internal device and the register RA. The Derived Operand, DO, specifies the operation to be performed or the device to be addressed. The immediate operand field may be viewed as an operation code extension field. Note that if indexing is specified, then the input/output operation or device address is formed by summing the contents of the register RX and the immediate field. This is a privileged instruction.

The mandatory and optional input/output immediate command fields are listed below.

Mil Std Mnemonic (IEEE Mnemonic)		Addr Mode	Format/Opcode	Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
XIO (XIO	RA,CMD Ra,#data)	IM		Input 30,02,01 Output 26,02,01
XIO (XIO	RA,CMD,RX RA,#data(Rx))	IMX		Input 33,02,01 Output 29,02,01

8	4	4	16
48	RA	RX	CMD

Mandatory XIO Command Fields and Mnemonics

2000 SMK

Set Interrupt Mask: This command outputs the 16-bit contents of the register RA to the interrupt mask register. A "1" in the corresponding bit position allows the interrupt to occur and a "0" prevents the interrupt from occurring except for those interrupts that are defined such that they cannot be masked.

IM 25,02,01
IMX 28,02,01
(vio) D 65,02,02
(vio) DX 68,02,02

2001 CLIR

Clear Interrupt Request: All interrupts are cleared (i.e., the pending interrupt register is cleared to all zeros) and the contents of the fault register are reset to zero.

IM 28,02,01
IMX 31,02,01
(vio) D 68,02,02
(vio) DX 71,02,02

2002 ENBL

Enable Interrupts: This command enables all interrupts which are not masked out. The enable operation takes place after execution of the next instruction.

IM 26,02,01
IMX 29,02,01
(vio) D 66,02,02
(vio) DX 69,02,02

2003 DSBL

Disable Interrupts: This command disables all interrupts (except those that are defined such that they cannot be disabled) at the beginning of the execution of the DSBL instruction.

IM 26,02,01
IMX 29,02,01
(vio) D 66,02,02
(vio) DX 69,02,02

2004 RPI

Reset Pending Interrupt: The individual interrupt bit to be reset shall be designated in register RA as a right justified four bit code. (0_{16} represents interrupt number 0,F_{16} represents interrupt number 15). If interrupt 1_{16} is to be cleared, then the contents of the fault register shall also be set to zero.

IM 36,02,01
IMX 39,02,01
(vio) D 76,02,02
(vio) DX 79,02,02

2005 SPI

Set Pending Interrupt Register: This command ORs the 16-bit contents of RA with the pending interrupt register. If there is a one in the corresponding bit position of the interrupt mask (same bit set in both the PI and the MK), and the interrupts are enabled, then an interrupt shall occur after execution of the next instruction. If PI_5 is set to 1, then N is assumed to be 0 (see set bit instructions).

IM 25,02,01
IMX 28,02,01
(vio) D 65,02,02
(vio) DX 68,02,02

200E WSW

Write Status Word: This command transfers the contents of RA to the status word.

W/MMU IM 33,02,01
WO/MMU IM 38,02,01
W/MMU IMX 36,02,01
WO/MMU IMX 41,02,01
W/MMU (vio) D 73,02,02
WO/MMU (vio) D 78,02,02
W/MMU (vio) DX 76,02,02
WO/MMU (vio) DX 81,02,02

A000 RMK

Read Interrupt Mask: The current interrupt mask is transferred into register RA. The interrupt mask is not altered.

IM 31,02,01
IMX 34,02,01
(vio) D 78,02,02
(vio) DX 81,02,02

A004 RPIR

Read Pending Interrupt Register: This command transfers the contents of the pending interrupt register into RA. The pending interrupt register is not altered.

IM 31,02,01
IMX 34,02,01
(vio) D 78,02,02
(vio) DX 81,02,02

A00E RSW

Read Status Word: This command transfers the 16-bit status word into register RA. The status word remains unchanged.

IM 31,02,01
IMX 34,02,01
(vio) D 78,02,02
(vio) DX 81,02,02

A00F RCFR

Read and Clear Fault Register: This command inputs the 16-bit fault register to register RA. The contents of the fault register are reset to zero. Bit 1 in the pending interrupt register is reset to zero.

IM 34,02,01
IMX 37,02,01
(vio) D 81,02,02
(vio) DX 84,02,02

Optional XIO Command Fields And Mnemonics Implemented Directly On The F9450

200A RNS

Reset Normal Power Up Discrete: This command resets the normal power up discrete bit.

IM 26,02,01
IMX 29,02,01
(vio) D 66,02,02
(vio) DX 69,02,02

4003 MPEN

Memory Protect Enable: This command allows the memory protect RAM to control memory protection.

IM 26,02,01
IMX 29,02,01
(vio) D 66,02,02
(vio) DX 69,02,02

4006 DMAE

Direct Memory Access Enable: This command enables direct memory access (DMA).

IM 26,02,01
IMX 29,02,01
(vio) D 66,02,02
(vio) DX 69,02,02

4007 DMAD

Direct Memory Access Disable: This command disables DMA.

IM 26,02,01
IMX 29,02,01
(vio) D 66,02,02
(vio) DX 69,02,02

4008 TAS

Timer A, Start: This command starts timer A from its current state. The timer is incremented every 10 microseconds.

IM 26,02,01
IMX 29,02,01
(vio) D 66,02,02
(vio) DX 69,02,02

4009 TAH

Timer A, Halt: This command halts timer A at its current state.

IM 26,02,01
IMX 29,02,01
(vio) D 66,02,02
(vio) DX 69,02,02

C00E ITB

Input Timer B: This command inputs the 16-bit contents of timer B into register RA. Bit fifteen is the least significant bit and represents a time increment of one hundred microseconds.

IM 34,02,01
IMX 37,02,01
(vio) D 84,02,02
(vio) DX 87,02,02

Optional XIO Command Fields And Mnemonics Implemented Directly On The F9451 And F9452

D0XX RMP

Read Memory Protect RAM (D000 + RAM Address): This command inputs the appropriate memory protect word into register RA. A "1" in a bit provides write protection and a "0" in a bit permits writing to the corresponding 1024 word physical memory block. The RAM words MSB (bit 0) represents the lowest number block and the RAM word LSB (bit 15) represents the highest block (i.e., bit 0 represents locations 0 throught 1023 and bit 15 represents locations 15360 through 16383 for word zero). Each word represents consecutive 16K blocks of physical memory. The RAM words of 0 through 63 apply to processor write protect and words 64 through 127 apply to DMA write protect.

IM 31,02,01
IMX 34,02,01
(vio) D 69,02,02
(vio) DX 72,02,02

D1XY RIPR

Read Instruction Page Register: This command transfers the 16-bit contents of the page register Y of the instruction set of group X to register RA.

IM 31,02,01
IMX 34,02,01
(vio) D 69,02,02
(vio) DX 72,02,02

D2XY ROPR

Read Operand Page Register: This command transfers the 16-bit contents of page register Y of the operand set of group X to register RA.

IM 31,02,01
IMX 34,02,01
(vio) D 69,02,02
(vio) DX 72,02,02

Optional XIO Command Fields And Mnemonics Implementable Externally To The F9450

0YXX PO

Programmed Output: This command outputs 16-bits of data from RA to a programmed I/O port. Y may be from 0 through 3.

IM 26,02,01
IMX 29,02,01
(vio) D 66,02,02
(vio) DX 69,02,02

2008 OD

Output Discretes: This command outputs the 16-bit contents of the register RA to the discrete output buffer. A "1" indicates an "on" condition and a "0" indicates an "off" condition.

IM 26,02,01
IMX 29,02,01
(vio) D 66,02,02
(vio) DX 69,02,02

4000 CO

Console Output: The 16-bit contents (2 bytes) of register RA are output to the console. The eight most significant bits (byte) are sent first. If no console is present, then this command is treated as a NOP.

IM 26,02,01
IMX 29,02,01
(vio) D 66,02,02
(vio) DX 69,02,02

4001 CLC

Clear Console: This command clears the console interface.

IM 26,02,01
IMX 29,02,01
(vio) D 66,02,02
(vio) DX 69,02,02

4004 ESUR

Enable Start Up ROM: This command enables the start up ROM (i.e.,the ROM overlays main memory).

IM 26,02,01
IMX 29,02,01
(vio) D 66,02,02
(vio) DX 69,02,02

400A OTA

Output Timer A: The contents of register RA are loaded (ie., jam transferred) into timer A and the timer automatically starts operation by incrementing from the loaded timer in steps of ten microseconds. Bit fifteen is the least significant bit and shall represent ten microseconds.

IM 28,02,01
IMX 31,02,01
(vio) D 68,02,02
(vio) DX 71,02,02

400B GO

Trigger Go Indicator: This command restarts a counter which is connected to a discrete output. The period of time from restart to time-out shall be determined by the system requirements. When the Go timer is started, the discrete output shall go high and remain high for TBD milliseconds, at which time the output shall go low unless another Go is executed. The Go discrete output signal may be used as a software fault indicator.

IM 26,02,01
IMX 29,02,01
(vio) D 66,02,02
(vio) DX 69,02,02

400C TBS

Timer B, Start: This command starts timer B from its current state. The timer is incremented every 100 microseconds.

IM 26,02,01
IMX 29,02,01
(vio) D 66,02,02
(vio) DX 69,02,02

400D TBH

Timer B, Halt: This command halts timer B at its current state.

IM 26,02,01
IMX 29,02,01
(vio) D 66,02,02
(vio) DX XXXX2,02

400E OTB

Output Timer B: The contents of register RA are loaded (i.e., jam transferred) into timer B and the timer automatically starts operation by incrementing from the loaded timer in steps of one hundred microseconds. Bit fifteen is the least significant bit and shall represent one hundred microseconds.

IM 28,02,01
IMX 31,02,01
(vio) D 68,02,02
(vio) DX 71,02,02

50XX LMP

Load Memory Protect RAM (5000 + RAM address): This command outputs the 16-bit contents of register RA to the memory protect RAM. A "1" in a bit provides write protection and a "0" in a bit permits writing the corresponding 1024 word physical memory block. The RAM word MSB (bit 0) represents the lowest number block and the RAM word LSB (bit 15) represents the highest block (i.e., bit 0 represents locations 0 through 1023 and bit 15 represents locations 15360 through 16383 for word zero). Each word represents consecutive 16K blocks of physical memory. The RAM words of 0 through 63 apply to processor write protect and words 64 through 127 apply to DMA write protect.

IM 26,02,01
IMX 29,02,01
(vio) D 66,02,02
(vio) DX 69,02,02

51XY WIPR

Write Instruction Page Register: This command transfers the contents of register RA to page register Y of the instruction set of group X.

IM 26,02,01
IMX 29,02,01
(vio) D 66,02,02
(vio) DX 69,02,02

52XY WOPR

Write Operand Page Register: This command transfers the contents of register RA to page register Y of the operand set of group X.

IM 26,02,01
IMX 29,02,01
(vio) D 66,02,02
(vio) DX 69,02,02

C00A ITA

Input Timer A: This command inputs the 16-bit contents of timer A into register RA. Bit fifteen is the least significant bit and represents a time increment of ten microseconds.

IM 34,02,01
IMX 37,02,01
(vio) D 84,02,02
(vio) DX 87,02,02

4005 DSUR *Disable Start Up ROM:* This command disables the start up ROM.

IM 26,02,01
IMX 29,02,01
(vio) D 66,02,02
(vio) DX 69,02,02

8YXX PI *Programmed Input:* This command inputs 16-bits of data into RA from the programmed I/O port. Y may be from 0 through 3.

IM 31,02,01
IMX 34,02,01
(vio) D 65,02,02
(vio) DX 72,02,02

A001 RIC1 *Read Input/Output Interrupt Code, Level 1:* This command inputs the contents of the level 1 IOIC register into register RA. The channel number is right justified.

IM 31,02,01
IMX 34,02,01
(vio) D 65,02,02
(vio) DX 72,02,02

A002 RIC2 *Read Input/Output Interrupt Code, Level 2:* This command inputs the contents of level 2 IOIC register into register RA. The channel number is right justified.

IM 31,02,01
IMX 34,02,01
(vio) D 65,02,02
(vio) DX 72,02,02

A008 RDOR *Read Discrete Output Register:* This command inputs the 16-bit discrete output buffer into register RA.

IM 31,02,01
IMX 34,02,01
(vio) D 65,02,02
(vio) DX 72,02,02

A009 RDI

Read Discrete Input: This command inputs the 16-bit discrete input word into register RA. A "1" indicates an "on" condition and a "0" indicates an "off" condition.

IM 31,02,01
IMX 34,02,01
(vio) D 65,02,02
(vio) DX 72,02,02

A00B TPIO

Test Programmed Output: This command inputs the 16-bit contents of the programmed output buffer into register RA. This command may be used to test the PIO channel by means of a wrap around test.

IM 31,02,01
IMX 34,02,01
(vio) D 65,02,02
(vio) DX 72,02,02

A00D RMFS

Read Memory Fault Status: This command transfers the 16-bit contents of the memory fault status register to RA. The fields within the memory fault status register shall delineate memory related fault types and shall provide the page register designators associated with the designated fault.

IM 31,02,01
IMX 34,02,01
(vio) D 65,02,02
(vio) DX 72,02,02

C000 CI

Console Input: This command inputs the 16-bits (2 bytes) from the console into register RA. The eight most significant bits of RA shall represent the first byte.

IM 31,02,01
IMX 34,02,01
(vio) D 65,02,02
(vio) DX 72,02,02

C001 RCS

Read Console Status: This command inputs the console interface status into register RA. The status is right justified.

IM 31,02,01
IMX 34,02,01
(vio) D 65,02,02
(vio) DX 72,02,02

Register Transfer Description:
Varies depending on the command field.

Registers Affected:
Varies depending on the command field.

Input/Output

VIO

Vectored Input/Output

DESCRIPTION: The vectored input/output instruction performs the I/O operation as specified by the input/output vector table starting at the derived address, DA, as shown below:

DA	CMD
DA+1	Vector Select
DA+2	Data
. . .	. . .

} one data word for each bit set in the vector select

The input/output operation or device address is specified by the sum of the CMD and the product of the bit number of the bit set in the vector select times the contents of RA. This device address is then interpreted as specified by the XIO instruction (see paragraph 5.1) with the exception that I/O data is transferred to or from DA + 2 + i rather than RA (where i starts at zero and is incremented after each transfer). This is a privileged instruction. If an illegal XIO command is encountered as part of a VIO chain, the following actions occur:

a. The illegal I/O command bit of the fault register (FT_5) is set to a one.

b. The VIO chain is terminated, and the illegal XIO is treated as a NOP. This termination shall not affect execution of preceding XIO commands which are part of the VIO chain being executed.

Mil Std Mnemonic (IEEE Mnemonic)	Addr Mode	Format/Opcode				Clock Cycles, Instruction Fetch Bus Cycles, Operand Bus Cycles
		8	4	4	16	
VIO RA,ADDR (XIOM Ra,addr)	D	49	RA	RX	ADDR	Input 65,02,00 Output 66,02,00
VIO RA,ADDR,RX (XIOM RA,addr(Rx))	DX					Input 72,02,00 Output 69,02,00

Register Transfer Description:

Step 1. n ← 0 and i ← 0;

Step 2. if $[DA+1]_n = 1$, then I/O command = [DA] + n × (RA);

Step 3. $FT_5 \leftarrow 1$, exit, if XIO = illegal command;

Step 4. if $[DA+1]_n = 1$, then I/O data = [DA + 2 + i];

Step 5. if $[DA+1]_1 = 1$, then i ← i + 1;

Step 6. n ← n + 1, exit, if n = 16;

Step 7. go to step 2;

Registers Affected:

None

Index to Instructions

Key: SP = Single Precision 16-bit
DP = Double Precision 32-bit
FP = Single Precision Floating Point
XP = Extended Precision Floating Point

Mnemonic	Addressing Mode	Function	Page
STC,STCI,STZ,STZI	D,DX,I,IX	Store a non-negative constant	A6-9
STLB,SLBI	I,IX,D,DX	Store into lower byte	A6-14
STM	D,DX	Store multiple registers	A7-6
STUB,SUB	D,DX,I,IX	Store into upper byte	A6-13
XBR	S	Exchange bytes in register	A6-15
XWR	R	Exchange words in registers	A6-16
Program Control			
BEX	S	Branch to executive	A8-14
BEZ	ICR	Branch if equal to	A8-8
BGE	ICR	Branch if greater than or equal to	A8-13
BGT	ICR	Branch if greater than	A8-11
BIF	S	Built-in function (for coprocessor)	A8-20
BLE	ICR	Branch if less than or equal to	A8-10
BLT	ICR	Branch if less than	A8-9
BNZ	ICR	Branch if not equal to	A8-12
BPT	S	Breakpoint	A8-19
BR	ICR	Branch unconditionally	A8-7
JC,JCI	D,DX,I,IX	Jump on condition	A8-3
JS	D,DX	Jump TO subroutine	A8-5
LST,LSTI	D,DX,I,IX	Load status	A8-15
NOP	S	No operation	A8-18
SJS	D,DX	Stack IC and jump to subroutine	A8-16
SOJ	D,DX	Subtract one and jump	A8-6
URS	S	Unstack IC and return from subroutine	A8-17
Input/Output			
VIO	D,DX	Vectored input/output	A9-13
XIO	IM,IMX	Execute input/output	A9-3
Command fields for XIO:			
CI		Console Input	A9-12
CLC		Clear console	A9-10
CLIR		Clear interrupt request	A9-3
CO		Console Output	A9-10
DMAD		DMA disable	A9-6
DMAE		DMA enable	A9-6
DSBL		Disable interrupts	A9-4

Index to Instructions

Key: SP = Single Precision 16-bit
DP = Double Precision 32-bit
FP = Single Precision Floating Point
XP = Extended Precision Floating Point

Index to Instructions

NOTICE

Timing characteristics for the devices discussed in this book are typical and should not be regarded as absolute inasmuch as revision levels of the products differ slightly in their timing characteristics. For exact timing information, see the current product Data Sheet, or contact the Fairchild Microcontroller Division, your Fairchild Product Specialist, or Sales Office.

FAIRCHILD

A Schlumberger Company

IEEE 1750A Cross-Reference

This appendix cross-references IEEE mnemonics to their corresponding 1750A instructions. It is included as a convenience to developers working with an assembler that utilizes IEEE mnemonics.

	IEEE	Addressing	MIL-STD	
Mnemonic	Operand	Mode	Mnemonic	Operand
ABS	Rb,Ra	R	ABS	RA,RB
ABSF	Rb,Ra	R	FABS	RA,RB
ABSL	Rb,Ra	R	DABS	RA,RB
ADD	#data,Ra	IM	AIM	RA,DATA
ADD	(Rx)(Br),R2	BX	ABX	BR,RX
ADD	Rb,Ra	R	AR	RA,RB
ADD	addr(Rx),Ra	DX	A	RA,ADDR,RX
ADD	dspl(Br),R2	B	AB	BR,DSPL
ADD4	#data,Ra	ISP	AISP	RA,N
ADD4	#data,addr(Rx)	DX	INCM	N,ADDR,RX
ADDF	(Rx)(Br),R0	BX	FABX	BR,RX
ADDF	Rb,Ra	R	FAR	RA,RB
ADDF	addr(Rx),Ra	DX	FA	RA,ADDR,RX
ADDF	dspl(Br),R0	B	FAB	BR,DSPL
ADDL	Rb,Ra	R	DAR	RA,RB
ADDL	addr(Rx),Ra	DX	DA	RA,ADDR,RX
ADDX	Rb,Ra	R	EFAR	RA,RB
ADDX	addr(Rx),Ra	DX	EFA	RA,ADDR,RX
AND	#data,Ra	IM	ANDM	RA,DATA
AND	(Rx)(Br),R2	BX	ANDX	BR,RX
AND	Rb,Ra	R	ANDR	RA,RB
AND	addr(Rx),Ra	DX	AND	RA,ADDR,RX
AND	dspl(Br),R2	B	ANDB	BR,DSPL
BC	addr(Rx)	DX	JC	8,LABEL,RX
BC	addr(Rx)@	IX	JCI	8,LABEL,RX
BCE	addr(Rx)	DX	JC	10,LABEL,RX
BCE	addr(Rx)@	IX	JCI	10,LABEL,RX
BCGE	addr(Rx)	DX	JC	14,LABEL,RX
BCGE	addr(Rx)@	IX	JCI	14,LABEL,RX
BCGT	addr(Rx)	DX	JC	12,LABEL,RX
BCGT	addr(Rx)@	IX	JCI	12,LABEL,RX
BCLE	addr(Rx)	DX	JC	11,LABEL,RX
BCLE	addr(Rx)@	IX	JCI	11,LABEL,RX
BCLT	addr(Rx)	DX	JC	9,LABEL,RX
BCLT	addr(Rx)@	IX	JCI	9,LABEL,RX
BCN	addr(Rx)	DX	JC	9,LABEL,RX
BCN	addr(Rx)@	IX	JCI	9,LABEL,RX
BCNE	addr(Rx)	DX	JC	13,LABEL,RX
BCNE	addr(Rx)@	IX	JCI	13,LABEL,RX
BCNN	addr(Rx)	DX	JC	14,LABEL,RX
BCNN	addr(Rx)@	IX	JCI	14,LABEL,RX
BCNP	addr(Rx)	DX	JC	11,LABEL,RX
BCNP	addr(Rx)@	IX	JCI	11,LABEL,RX
BCNZ	addr(Rx)	DX	JC	13,LABEL,RX
BCNZ	addr(Rx)@	IX	JCI	13,LABEL,RX
BCP	addr(Rx)	DX	JC	12,LABEL,RX
BCP	addr(Rx)@	IX	JCI	12,LABEL,RX
BCZ	addr(Rx)	DX	JC	10,LABEL,RX
BCZ	addr(Rx)@	IX	JCI	10,LABEL,RX
BE	%addr	ICR	BEZ	LABEL
BE	addr(Rx)	DX	JC	2,LABEL,RX
BE	addr(Rx)@	IX	JCI	2,LABEL,RX
BGE	%addr	ICR	BGE	LABEL
BGE	addr(Rx)	DX	JC	6,LABEL,RX
BGE	addr(Rx)@	IX	JCI	6,LABEL,RX
BGT	%addr	ICR	BGT	LABEL
BGT	addr(Rx)	DX	JC	4,LABEL,RX
BGT	addr(Rx)@	IX	JCI	4,LABEL,RX
BLE	%addr	ICR	BLE	LABEL
BLE	addr(Rx)	DX	JC	3,LABEL,RX
BLE	addr(Rx)@	IX	JCI	3,LABEL,RX
BLT	%addr	ICR	BLT	LABEL
BLT	addr(Rx)	DX	JC	1,LABEL,RX
BLT	addr(Rx)@	IX	JCI	1,LABEL,RX
BN	%addr	ICR	BLT	LABEL
BN	addr(Rx)	DX	JC	1,LABEL,RX
BN	addr(Rx)@	IX	JCI	1,LABEL,RX
BNE	%addr	ICR	BNZ	LABEL
BNE	addr(Rx)	DX	JC	5,LABEL,RX
BNE	addr(Rx)@	IX	JCI	5,LABEL,RX
BNN	%addr	ICR	BGE	LABEL
BNN	addr(Rx)	DX	JC	6,LABEL,RX
BNN	addr(Rx)@	IX	JCI	6,LABEL,RX
BNP	%addr	ICR	BLE	LABEL
BNP	addr(Rx)	DX	JC	3,LABEL,RX
BNP	addr(Rx)@	IX	JCI	3,LABEL,RX
BNZ	%addr	ICR	BNZ	LABEL
BNZ	addr(Rx)	DX	JC	5,LABEL,RX
BNZ	addr(Rx)@	IX	JCI	5,LABEL,RX
BP	%addr	ICR	BGT	LABEL
BP	addr(Rx)	DX	JC	4,LABEL,RX
BP	addr(Rx)@	IX	JCI	4,LABEL,RX
BR	%addr	ICR	BR	LABEL
BR	addr(Rx)	DX	JC	15,LABEL,RX
BR	addr(Rx)@	IX	JCI	15,LABEL,RX
BRK	#data	S	BEX	N
BZ	%addr	ICR	BEZ	LABEL
BZ	addr(Rx)	DX	JC	2,LABEL,RX
BZ	addr(Rx)@	IX	JCI	2,LABEL,RX
CALL	@-Ra,addr(Rx)	DX	SJS	RA,LABEL,RX
CALL	Ra,addr(Rx)	DX	JS	RA,LABEL,RX
CLRI	#data,Rb	R	RBR	N,RB
CLRI	#data,addr(Rx)	DX	RB	N,ADDR,RX
CLRI	#data,addr(Rx)@	IX	RBI	N,ADDR,RX
CLRI	Ra,Rb	R	RVBR	RA,RB
CMP	R2,(Rx)(Br)	BX	CBX	BR,RX
CMP	R2,dspl(Br)	B	CB	BR,DSPL
CMP	Ra,#data	IM	CIM	RA,DATA
CMP	Ra,Rb	R	CR	RA,RB
CMP	Ra,addr(Rx)	DX	C	RA,ADDR,RX
CMP4	Ra,#data	ISN	CISN	RA,N
CMP4	Ra,#data	ISP	CISP	RA,N
CMPF	R0,(Rx)(Br)	BX	FCBX	BR,RX
CMPF	R0,dspl(Br)	B	FCB	BR,DSPL
CMPF	Ra,Rb	R	FCR	RA,RB
CMPF	Ra,addr(Rx)	DX	FC	RA,ADDR,RX
CMPL	Ra,Rb	R	DCR	RA,RB
CMPL	Ra,addr(Rx)	DX	DC	RA,ADDR,RX
CMPRNG	Ra,addr(Rx)	DX	CBL	RA,ADDR,RX
CMPX	Ra,Rb	R	EFCR	RA,RB
CMPX	Ra,addr(Rx)	DX	EFC	RA,ADDR,RX
DECBNZ	Ra,addr(Rx)	DX	SOJ	RA,LABEL,RX
DIV	#data,Ra	IM	DIM	RA,DATA
DIV	(Rx)(Br),R2	BX	DBX	BR,RX
DIV	Rb,Ra	R	DR	RA,RB
DIV	addr(Rx),Ra	DX	D	RA,ADDR,RX
DIV	dspl(Br),R2	B	DB	BR,DSPL
DIV4S	#data,Ra	ISN	DISN	RA,N
DIV4S	#data,Ra	ISP	DISP	RA,N
DIVF	(Rx)(Br),R0	BX	FDBX	BR,RX
DIVF	Rb,Ra	R	FDR	RA,RB
DIVF	addr(Rx),Ra	DX	FD	RA,ADDR,RX
DIVF	dspl(Br),R0	B	FDB	BR,DSPL
DIVLS	Rb,Ra	R	DDR	RA,RB

IEEE 1750A Cross-Reference

	IEEE	Addressing	MIL-STD	
Mnemonic	Operand	Mode	Mnemonic	Operand
DIVLS	addr(Rx),Ra	DX	DD	RA,ADDR,RX
DIVS	#data,Ra	IM	DVIM	RA,DATA
DIVS	Rb,Ra	R	DVR	RA,RB
DIVS	addr(Rx),Ra	DX	DV	RA,ADDR,RX
DIVX	Rb,Ra	R	EFDR	RA,RB
DIVX	addr(Rx),Ra	DX	EFD	RA,ADDR,RX
FLOAT	Rb,Ra	R	FLT	RA,RB
FLOATL	Rb,Ra	R	EFLT	RA,RB
HALT		S	BPT	
INTGR	Rb,Ra	R	FIX	RA,RB
INTGRX	Rb,Ra	R	EFIX	RA,RB
LD	#data(Rx),Ra	IMX	LIM	RA,DATA,RX
LD	#data,Ra	IM	LIM	RA,DATA
LD	(Rx)(Br),R2	BX	LBX	BR,RX
LD	addr(Rx),Ra	DX	L	RA,ADDR,RX
LD	addr(Rx),STATUS	DX	LST	ADDR,RX
LD	addr(Rx)@,Ra	IX	LI	RA,ADDR,RX
LD	addr(Rx)@,STATUS	IX	LSTI	ADDR,RX
LD	dspl(Br),R2	B	LB	BR,DSPL
LD4	#data,Ra	ISN	LISN	RA,N
LD4	#data,Ra	ISP	LISP	RA,N
LDL	(Rx)(Br),R0	BX	DLBX	BR,RX
LDL	addr(Rx),Ra	DX	DL	RA,ADDR,RX
LDL	addr(Rx)@,Ra	IX	DLI	RA,ADDR,RX
LDL	dspl(Br),R0	B	DLB	BR,DSPL
LDLB	addr(Rx)Ra	DX	LLB	RA,ADDR,RX
LDLB	addr(Rx)@,Ra	IX	LLBI	RA,ADDR,RX
LDM	#data,addr(Rx),R0	DX	LM	N,ADDR,RX
LDUB	addr(Rx),Ra	DX	LUB	RA,ADDR,RX
LDUB	addr(Rx)@,Ra	IX	LUBI	RA,ADDR,RX
LDX	addr(Rx),Ra	DX	EFL	RA,ADDR,RX
MOV	#data,addr(Rx)	DX	STC	N,ADDR,RX
MOV	#data,addr(Rx)@	IX	STCI	N,ADDR,RX
MOV	Rb,Ra	R	LR	RA,RB
MOVBK	@Rb,@Ra	S	MOV	RA,RB
MOVL	Rb,Ra	R	DLR	RA,RB
MUL	#data,Ra	IM	MIM	RA,DATA
MUL	(Rx)(Br),R2	BX	MBX	BR,RX
MUL	Rb,Ra	R	MR	RA,RB
MUL	addr(Rx),Ra	DX	M	RA,ADDR,RX
MUL	dspl(Br),R2	B	MB	BR,DSPL
MUL4S	#data,Ra	ISN	MISN	RA,N
MUL4S	#data,Ra	ISP	MISP	RA,N
MULF	(Rx)(Br),R0	BX	FMBX	BR,RX
MULF	Rb,Ra	R	FMR	RA,RB
MULF	addr(Rx),Ra	DX	FM	RA,ADDR,RX
MULF	dspl(Br),R0	B	FMB	BR,DSPL
MULLS	Rb,Ra	R	DMR	RA,RB
MULLS	addr(Rx),Ra	DX	DM	RA,ADDR,RX
MULS	#data,Ra	IM	MSIM	RA,DATA
MULS	Rb,Ra	R	MSR	RA,RB
MULS	addr(Rx),Ra	DX	MS	RA,ADDR,RX
MULX	Rb,Ra	R	EFMR	RA,RB
MULX	addr(Rx),Ra	DX	EFM	RA,ADDR,RX
NAND	#data,Ra	IM	NIM	RA,DATA
NAND	Rb,Ra	R	NR	RA,RB
NAND	addr(Rx),Ra	DX	N	RA,ADDR,RX
NEG	Rb,Ra	R	NEG	RA,RB
NEGF	Rb,Ra	R	FNEG	RA,RB
NEGL	Rb,Ra	R	DNEG	RA,RB
NOP		S	NOP	
OR	#data,Ra	IM	ORIM	RA,DATA
OR	(Rx)(Br),R2	BX	ORBX	BR,RX
OR	Rb,Ra	R	ORR	RA,RB
OR	addr(Rx),Ra	DX	OR	RA,ADDR,RX
OR	dspl(Br),R2	B	ORB	BR,DSPL

	IEEE	Addressing	MIL-STD	
Mnemonic	Operand	Mode	Mnemonic	Operand
POPM	@R15,Ra,Rb	S	POPM	RA,RB
PUSHM	Ra,Rb,@R15	S	PSHM	RA,RB
RET	@Ra+	S	URS	RA
RET	Rx	DX	JC	15,0,RX
RET	addr(Rx)@	IX	JCI	15,LABEL,RX
ROL	#data,Rb	R	SLC	RB,N
ROLL	#data,Rb	R	DSLC	RB,N
ROLR	Rb,Ra	R	SCR	RA,RB
ROLRL	Rb,RA	R	DSCR	RA,RB
SETI	#data,Rb	R	SBR	N,RB
SETI	#data,addr(Rx)	DX	SB	N,ADDR,RX
SETI	#data,addr(Rx)@	IX	SBI	N,ADDR,RX
SETI	Ra,Rb	R	SVBR	RA,RB
SHL	#data,Rb	R	SLL	RB,N
SHLL	#data,Rb	R	DSLL	RB,N
SHLR	Rb,Ra	R	SLR	RA,RB
SHLRA	Rb,Ra	R	SAR	RA,RB
SHLRAL	Rb,Ra	R	DSAR	RA,RB
SHLRL	Rb,Ra	R	DSLR	RA,RB
SHR	#data,Rb	R	SRL	RB,N
SHRA	#data,Rb	R	SRA	RB,N
SHRAL	#data,Rb	R	DSRA	RB,N
SHRL	#data,Rb	R	DSRL	RB,N
ST	R2,(Rx)(Br)	BX	STBX	BR,RX
ST	R2,dspl(Br)	B	STB	BR,DSPL
ST	Ra,addr(Rx)	DX	ST	RA,ADDR,RX
ST	Ra,addr(Rx)@	IX	STI	RA,ADDR,RX
STL	R0,(Rx)(Br)	BX	DSTX	BR,RX
STL	R0,dspl(Br)	B	DSTB	BR,DSPL
STL	Ra,addr(Rx)	DX	DST	RA,ADDR,RX
STL	Ra,addr(Rx)@	IX	DSTI	RA,ADDR,RX
STLB	Ra,addr(Rx)	DX	STLB	RA,ADDR,RX
STLB	Ra,addr(Rx)@	IX	SLBI	RA,ADDR,RX
STM	#data,R0,addr(Rx)	DX	STM	N,ADDR,RX
STMI	Ra,addr(Rx)	DX	SRM	RA,ADDR,RX
STUB	Ra,addr(Rx)	DX	STUB	RA,ADDR,RX
STUB	Ra,addr(Rx)@	IX	SUBI	RA,ADDR,RX
STX	Ra,addr(Rx)	DX	EFST	RA,ADDR,RX
SUB	#data,Ra	IM	SIM	RA,DATA
SUB	(Rx)(Br),R2	BX	SBBX	BR,RX
SUB	Rb,Ra	R	SR	RA,RB
SUB	addr(Rx),Ra	DX	S	RA,ADDR,RX
SUB	dspl(Br),R2	B	SBB	BR,DSPL
SUB4	#data,Ra	ISP	SISP	RA,N
SUB4	#data,addr(Rx)	DX	DECM	N,ADDR,RX
SUBF	(Rx)(Br),R0	BX	FSBX	BR,RX
SUBF	Rb,Ra	R	FSR	RA,RB
SUBF	addr(Rx),Ra	DX	FS	RA,ADDR,RX
SUBF	dspl(Br),R0	B	FSB	BR,DSPL
SUBL	Rb,RA	R	DSR	RA,RB
SUBL	addr(Rx),Ra	DX	DS	RA,ADDR,RX
SUBX	Rb,Ra	R	EFSR	RA,RB
SUBX	addr(Rx),Ra	DX	EFS	RA,ADDR,RX
TESTI	#data,Rb	R	TBR	N,RB
TESTI	#data,addr(Rx)	DX	TB	N,ADDR,RX
TESTI	#data,addr(Rx)@	IX	TBI	N,ADDR,RX
TESTI	Ra,Rb	R	TVBR	RA,RB
TESTSETI	#data,addr(Rx)	DX	TSB	N,ADDR,RX
XCH	Rb,Ra	R	XWR	RA,RB
XCHB	Ra	S	XBR	RA
XIO	Ra,#data	IM	XIO	RA,CMD
XIO	Ra,#data(Rx)	IMX	XIO	RA,CMD,RX
XIOM	Ra,addr(Rx)	DX	VIO	RA,ADDR,RX
XOR	#data,Ra	IM	XORM	RA,DATA
XOR	Rb,Ra	R	XORR	RA,RB
XOR	addr(Rx),Ra	DX	XOR	RA,ADDR,RX

FAIRCHILD

A Schlumberger Company

F9450 Instruction Execution Times

The tables in this appendix furnish execution times in microseconds for a core instruction set of the F9450. All times are at 20 MHz with no wait states, running on the "D-step" chip.

Note that execution times increase in inverse linear relationship to the clock speed. For example, if an instruction executes in 2.0 usec at 20 MHz, it will take 3.0 usec at 15 MHz and 4.0 usec at 10 MHz, assuming no wait states are inserted.

Blanks in the tables indicate that the addressing mode is not available for the type of operation.

NOTICE

Timing characteristics for the devices discussed in this book are typical and should not be regarded as absolute inasmuch as revision levels of the products differ slightly in their timing characteristics. For exact timing information, see the current product Data Sheet, or contact the Fairchild Microcontroller Division, your Fairchild Product Specialist, or Sales Office.

Table C.1 Single Precision

	Load/ store	Add/ sub	Multiply	Divide	Compare	Set/reset bit
Register	0.2	0.25	1.85	4.85	0.4	0.35
Direct	0.6	0.65	2.25	5.1	0.8	0.8
Direct indexed	0.6	0.65	2.25	5.1	0.8	0.8
Indirect	0.8					1.0
Immediate	0.55	0.6	2.2	5.1	0.75	
Immediate short	0.35	0.4			0.55	
Base relative	0.55	0.6	2.2	5.05	0.75	
Base relative indexed	0.55	0.6	2.2	5.05	0.75	

F9450 Instruction Execution Times

Table C.2 Double Precision

	Load	Store	Add/sub	Multiply	Divide	Compare
Register	0.8		0.9	6.3	11.95	1.05
Direct	1.1	0.8	1.2	6.6	12.25	1.35
Direct indexed	1.1	0.8	1.2	6.6	12.25	1.35
Indirect	1.3	1.0				
Base relative	1.05	0.75				
Base relative indexed	1.05	0.75				

Table C.3 Single-Precision Floating Point

	Add/ sub*	Multiply*	Divide*	Compare
Register	3.1	6.0	11.7	2.6
Direct	3.4	6.3	12.0	2.9
Direct indexed	3.4	6.3	12.0	2.99
Base relative	3.35	6.25	11.95	2.85
Base relative indexed	3.35	6.25	11.95	2.85

Table C.4 Extended-Precision Floating Point

	Load	Store	Add/sub*	Multiply*	Divide*	Compare*
Register			3.55	12.55	24.0	2.6
Direct	1.3	1.0	3.9	12.9	24.35	3.25
Direct indexed	1.3	1.0	3.9	12.9	24.35	3.25

*Includes no shifts in exponent adjustment or in normalization

FAIRCHILD

A Schlumberger Company

Signal Functions and Connection Diagrams

This appendix furnishes diagrams illustrating the signal functions and pin assignments for the members of the F9450 family.

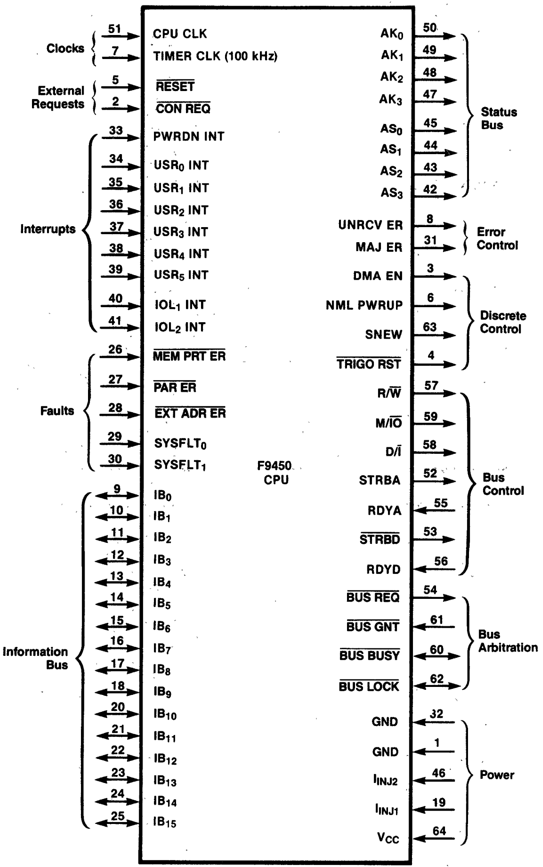


Figure D-1 F9450 CPU Signal Functions

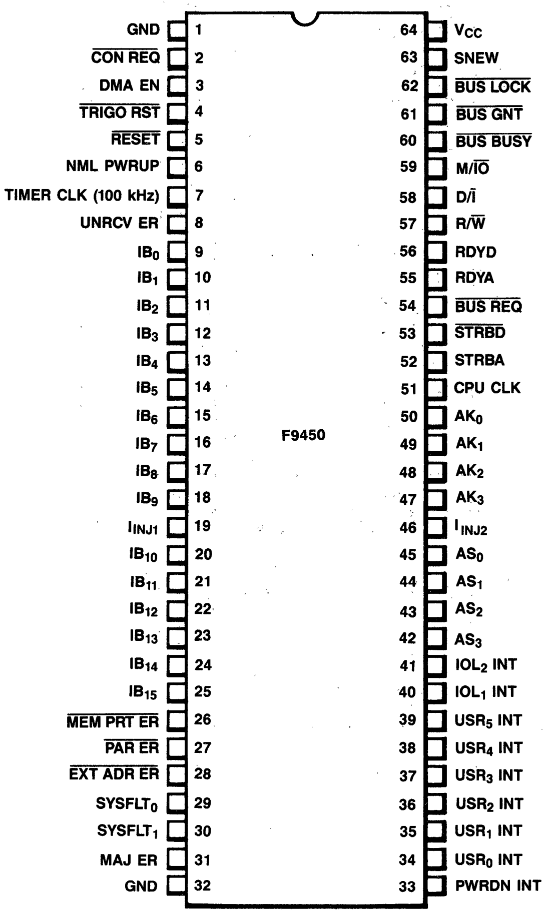


Figure D-2 F9450 CPU Connection Diagram

Signal Functions and Connection Diagrams

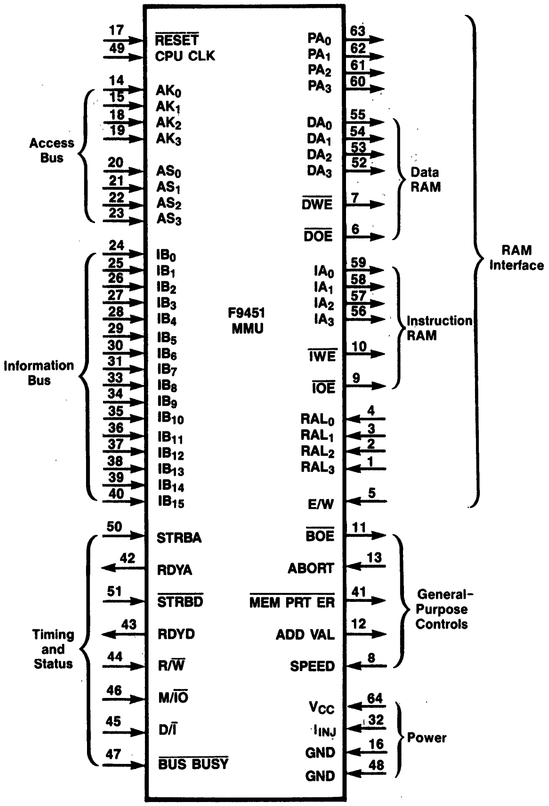


Figure D-3 F9451 MMU Signal Functions

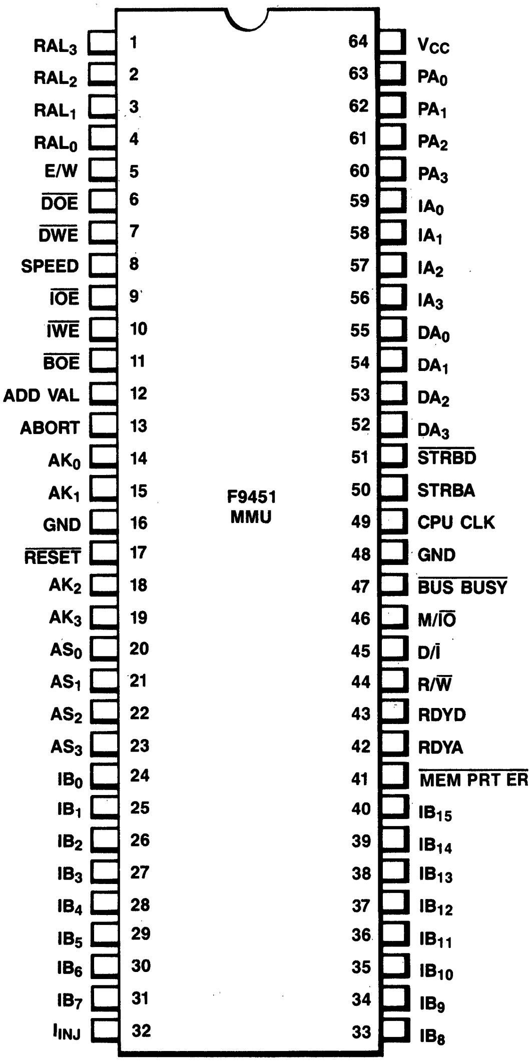


Figure D-4 F9451 MMU Connection Diagram

Signal Functions and Connection Diagrams

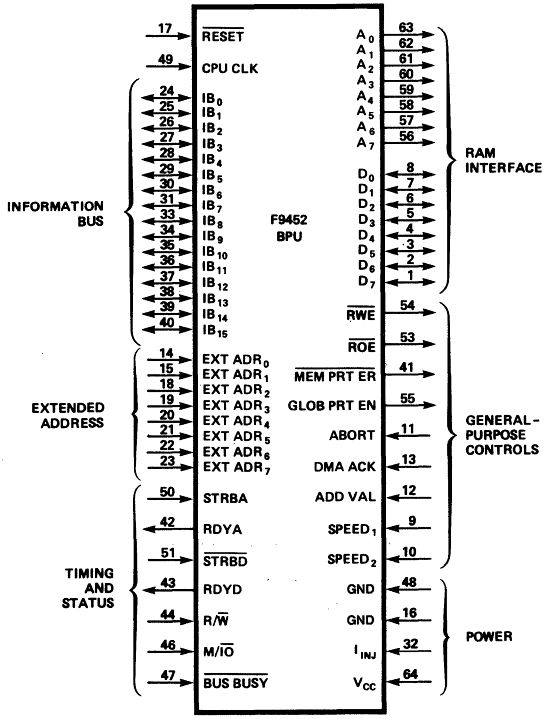


Figure D-5 F9452 BPU Signal Functions

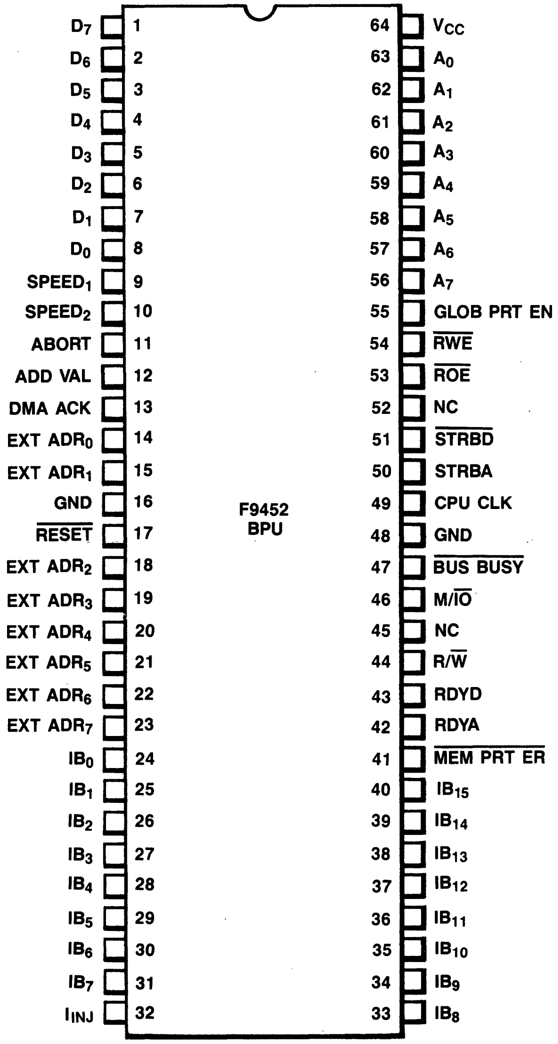


Figure D-6 F9452 BPU Connection Diagram

FAIRCHILD

A Schlumberger Company

Index

Index

Index

NOTES

NOTES

NOTES

FAIRCHILD

A Schlumberger Company

Fairchild reserves the right to make changes in circuitry or specifications at any time without notice. Manufactured under one or more of the following U.S. Patents: 2,981,877, 3,015,048, 3,064,167, 3,108,359 3,117,260; other patents pending. Fairchild cannot assume responsibility for use of any circuitry described other than circuitry embodied in a Fairchild product. No other patent licenses are implied. Printed in U.S.A.

8900504-001A 20M

FAIRCHILD A Schlumberger Company

F9450

86