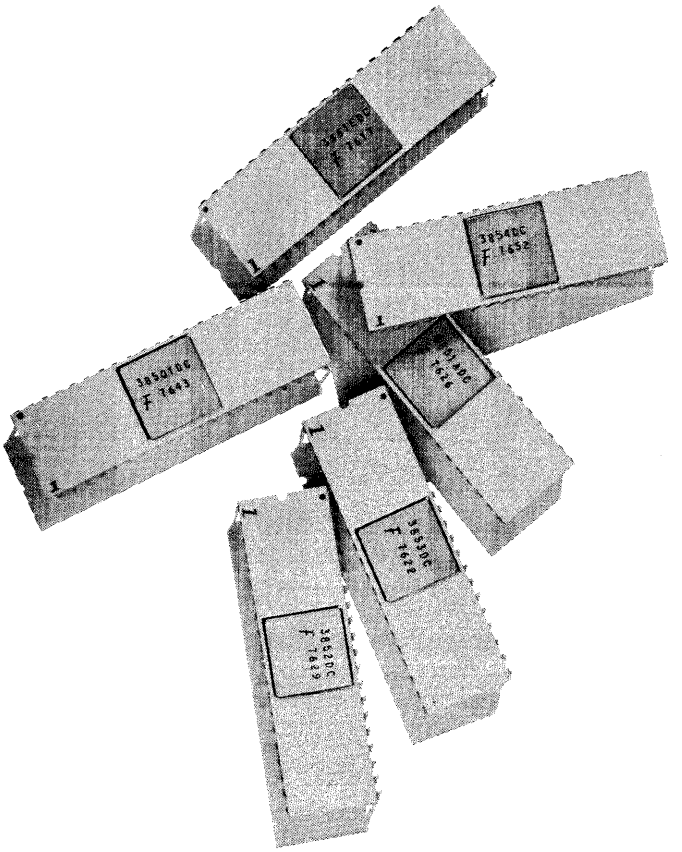
F8 USER'S GUIDE



FAIRCHILD

MOS Microcomputer Division
464 Ellis Street, Mountain View, California 94042

F8 USER'S GUIDE

67095665

R	S	DATE	PAGE
A	0	2/13/76	
B	0	5/1/76	
C		11/1/77	

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Continued)

LIST OF ILLUSTRATIONS (Continued)

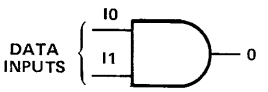
LIST OF TABLES

LIST OF TABLES (Continued)

1.0 The F8 Microcomputer System

Microcomputers represent a new concept in logic design: A single logic device capable of performing any logic operation.

The most elementary logic device receives one or more input signals and immediately generates output signals that may be defined as a function of the input signal via an appropriate truth table. Consider a simple AND gate, as illustrated in Figure 1-1.



DATA OUTPUT

I0	I1	0
0	0	0
0	1	0
1	0	0
1	1	1

Figure 1-1. A Very Elementary Logic Device

In Figure 1-1, both the input signals and the output signals may be described as data. A slightly more complex logic device will introduce signals which are not data, but rather provide control. Consider the latched dual input, four-bit multiplexer buffer illustrated in Figure 1-2.

The device illustrated in Figure 1-2 has taken two subtle steps toward becoming a microcomputer.

First, two control signals have been added.

Second, a four bit data unit has been defined.

Consider the two control signals; select (S) determines whether inputs A or inputs B will be outputs. Enable (E) determines when the selected input will be output. Until E goes true, the buffer disconnects itself from external logic by presenting a high impedance at its output pins.

Why is this logic device 4 bits wide? The answer, simply, is that the processing of parallel binary data

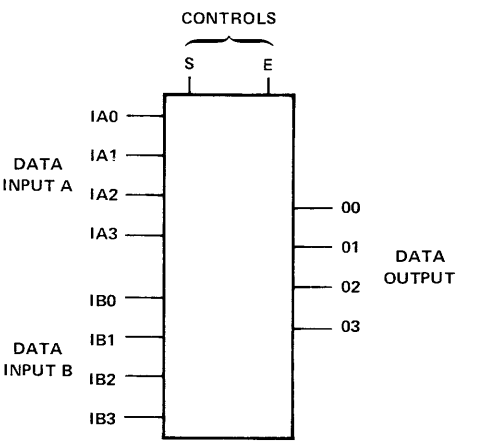


Figure 1-2. A Logic Device Using Control Signals

is a sufficiently common event for a 4-bit device to be justifiable in preference to four 1-bit devices; and a 4-bit device costs very little more than a 1-bit device, so using the 4-bit device will greatly reduce costs. This being the case, why not standardize on 8-bit devices? or 16-bit devices? The answer is that a logic designer must worry about the size of a DIP (Dual Inline Package). Economics dictate that any application must be implemented on as few PC cards as possible; therefore, using 40-pin DIPs, where a 20-pin DIP would do as well, wastes PC card space.

1.1 FROM LOGIC DEVICE TO MICROCOMPUTER

A relatively small conceptual step is all that is needed to go from the latched, dual input, buffer illustrated in Figure 1-2, to a microcomputer. Assume that semiconductor state of the art has reached the point where logic to implement every single device in the Fairchild TTL Data Book can be provided on a single chip. This device would have the same three types of signals as the latched, dual input buffer; namely, data input, data output and control. This is illustrated in Figure 1-3.

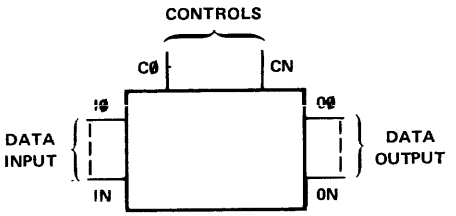


Figure 1-3. A Universal Logic Device

Resolving two aspects of the universal logic device illustrated in Figure 1-3 will provide the foundation for the design of any microcomputer. What will be the data width for the device and what will be the nature of the control signals?

Selecting a microcomputer's data width is about the same as selecting the data width for any parallel logic device. The selection is made by trading off the number of DIP pins available, against the way in which these pins can be used. This trade off has most frequently led to an 8-bit data width. In order to understand microcomputers in general, justifying this data width is not of any importance. It is sufficient to point out that while 4 bit and 16 bit microcomputers are available, 8 bit microcomputers are the most common.

There are, perhaps, ten thousand different logic devices listed in any complete catalog. Does this mean that fourteen control pins will be needed to provide sufficient selection options? Indeed, no. It would most certainly be impractical to faithfully reproduce ten thousand sets of logic on a single chip. A relatively small number of unique operations provides the basis for any logic device. By implementing this small number of unique operations on the single chip and by providing the means to correctly combine or sequence these elementary operations, a microcomputer can perform as though all ten thousand devices had been faithfully reproduced.

These are the basic functions out of which any logic device may be built:

1. Binary addition and subtraction. Subtraction can be (and usually is), replaced by the complement function, since most microromputers use twos complement binary arithmetic.
2. Boolean logic operators, including AND, OR and Exclusive OR. The NOT function is taken care of by having a complement.
3. Real time control. This is a more comprehensive equivalent of the enable strobe.

Binary arithmetic and Boolean Logic are described in "A Guide to Programming the F8 Microcomputer." Items 1 and 2 above are combined into a piece of logic called the "Arithmetic and Logic Unit" (or ALU).

Because the actual manipulative logic of the microcomputer is implemented as a set of basic functions, rather than as a compendium of well known logic devices, the whole concept of logic design changes when using a microcomputer. To be specific, when using a microcomputer, logic is broken down into a serial stream of basic operations. This serial stream of basic operations is created by programming the microcomputer in real time. Three pieces of logic must be added to the ALU in order to make this possible:

1. There must be a register which holds the control input. The control input is referred to as an instruction, and the register which holds it is referred to as the "Instruction Register."
2. There must be logic capable of decoding the instruction register's contents. This logic must then enable appropriate logic within the ALU, so that the operations specified by the control input occur.
3. There must be an additional data register capable of buffering data input to and output by the ALU. This register is named the "Accumulator."

Combining these three requirements, Figure 1-4 illustrates a group of logic which is often referred to as a "Central Processing Unit" (or CPU). The sequence of instructions which, taken together, cause the CPU to respond as required by an application, is referred to as a "Program."

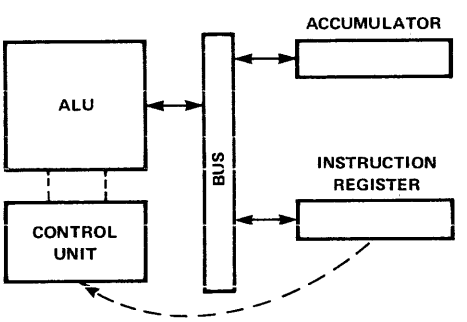


Figure 1-4. An Elementary Central Processing Unit

Since the Accumulator is a buffer for transient data, a status register will be added to the CPU. This register will record the condition of the most recent ALU operation. Conditions such as a zero or non-zero result, or a positive or negative result will be recorded.

Consider next, the source of the instruction code sequence and the source or destination of data. The instruction sequence, or program, is a definable and non-varying specification for any application; as such, it will usually become a Read Only Memory. This ROM is going to need an address register which, at any time, identifies the location from which the next instruction code will be fetched. This register is referred to as a Program Counter. Data, likewise, is going to need memory and an address register identifying the memory location which is the current data source or destination; however, data will require some Read/Write Memory, since some memory will have to serve as a data destination, as well as a data source. Adding program and data storage logic to a microcomputer system results in Figure 1-5.

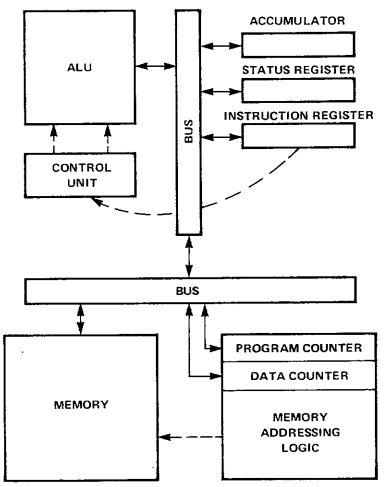


Figure 1-5. A Simple CPU with Memory

A basic necessity of any microcomputer system is a conduit for transmitting data to or from external logic. I/O ports serve this purpose. An I/O port is a bi-directional, 8 bit buffer, communicating on one side with external logic and on the other side with the microcomputer system. The I/O port also provides the third basic functions required of a microcomputer: the equivalent of enable logic. In a microcomputer system, the enable signal becomes an instruction which causes data to be output to, or input from an I/O port.

There is a further type of communication that occurs in microcomputer systems, but has no discrete logic parallel; it is the Interrupt. Observe that the microcomputer described thus far provides no means for external logic to initiate asynchronous communication. External logic is, in effect, at the mercy of the microcomputer. Unless the microcomputer program includes instructions to access an I/O port, external logic has no way of communicating with the microcomputer. This means that microcomputer program logic must be all encompassing enough to anticipate any external event which may occur. In many cases, such a program simply cannot exist. The interrupt is a signal which external logic transmits to the microcomputer in order to deflect the microcomputer program temporarily, to serve external logic's immediate needs. This concept is illustrated in Figure 1-6. "A Guide to Programming the Fairchild F8 Microcomputer" provides a thorough description of what interrupts are and how to use them.

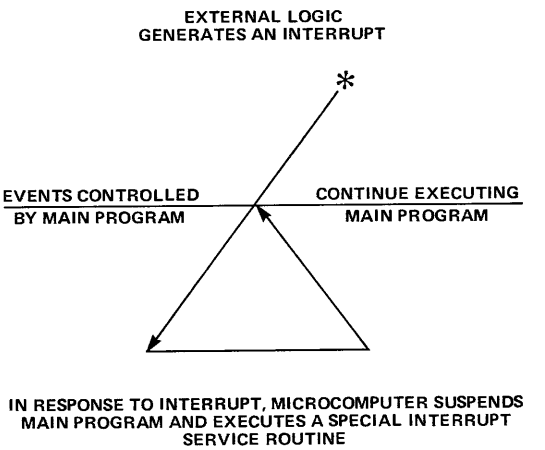
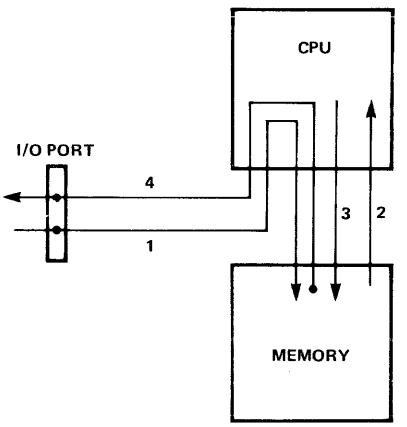


Figure 1-6. The Concept of an Interrupt

The F8 microcomputer makes additional use of interrupt logic to provide a real time capability via a programmable timer. The programmable timer is an 8 bit shift register which may be loaded with a value that specifies a fixed time period at the end of which an interrupt is generated. The programmable timer provides the microcomputer with the

ability to synchronize events within the system to real time outside the system.

At this point, a microcomputer system may be visualized as consisting of data manipulation logic within the CPU, plus large data buffers provided by memory. I/O ports constitute relatively small conduits through which data passes to or from external logic. Since memory buffers within the microcomputer system may be quite large, there will be an additional step required by logic that uses a microcomputer, an additional step which would not exist were discrete logic being used: raw data will be loaded into or out of memory buffers before or after being processed. This is illustrated in Figure 1-7.



1 DATA TO MEMORY
2 DATA TRANSMITTED AS INPUT TO CPU
3 DATA TO MEMORY AS OUTPUT FROM CPU
4 DATA OUTPUT FROM MEMORY

Figure 1-7. Memory as a Buffer between External Logic and the CPU

Using input and output instructions, the normal sequence of events required to output data from the microcomputer to external logic, is to load the data from memory into the accumulator, then transmit data from the accumulator to an I/O port. To input data, the data is first loaded from an I/O port into the accumulator, then it is moved from the accumulator to memory. If a large block of data is to be moved between memory and an external device, then time will be wasted performing the data movement in two steps. In addition, the CPU will be tied up implementing what is nothing more than simple data movement. This being the case, a separate path is provided between memory and external devices and this path is referred to as Direct Memory Access. Figure 1-8 now illustrates the complete logic of a microcomputer system. A great deal more could be said about the logic illustrated in Figure 1-8, and it has been said in the "Guide to Programming the F8 Microcomputer." This book assumes the reader understands all the concepts summarized thus far; if this assumption is incorrect, return to the "Guide to Programming the F8 Microcomputer."

1.2 TIMING IN MICROCOMPUTER SYSTEMS

Microcomputers are inherently synchronous devices. This is a necessity, since the complications of sequencing logic in innumerable different ways would become too complex to handle if logic were asynchronous. The entire microcomputer system is therefore driven by a clock and operations within the system occur within well defined instruction cycles. A register manipulation, such as doing a binary add of memory to the Accumulator and placing the result back into the Accumulator, takes one complete instruction cycle. The F8 instruction cycle is illustrated in Figure 1-9. It is the period between two trailing edges of a WRITE clock. Most of the F8 instruction cycles are short cycles (4Φ periods long), however long cycles (6Φ periods long) are used when data transfers between two circuits require a longer time than is available in a short cycle.

1.3 THE DEVICES OF THE F8 MICROCOMPUTER SYSTEM

Consider now how the total logic of a microcomputer as illustrated in Figure 1-8, is implemented on various devices of the F8 microcomputer system.

The distribution of logic among various devices of a microcomputer system is one of the most variable features of microcomputers in general. This results from the fact that by sheer coincidence, the requirements for a logic replacement device is quite similar in architecture to the traditional computer. The first microcomputers were developed as very small computers for use in intelligent terminals. The F8, on the other hand, was designed after the microcomputer industry had started to take a recognizable form and intelligent decisions could be made regarding the most economical implemen-

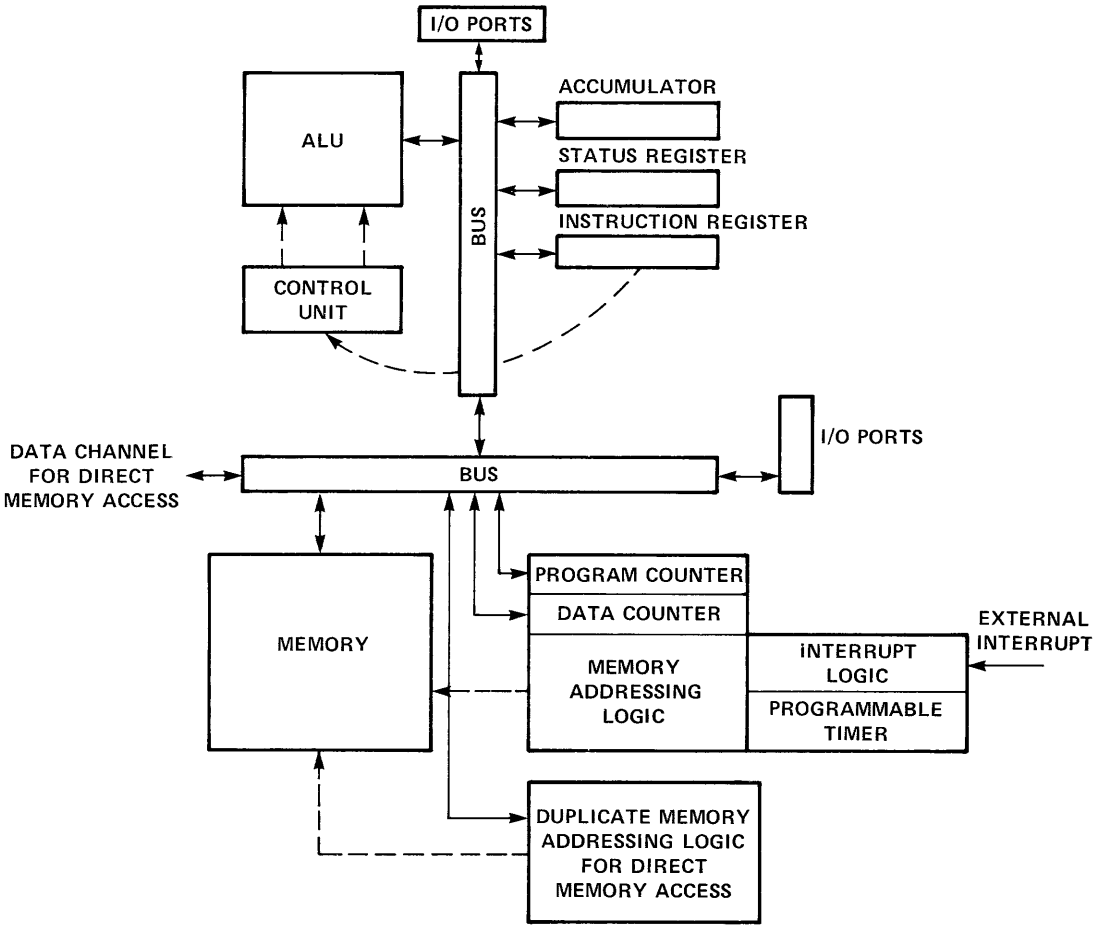


Figure 1-8. Functional Logic of a Complete Microcomputer System

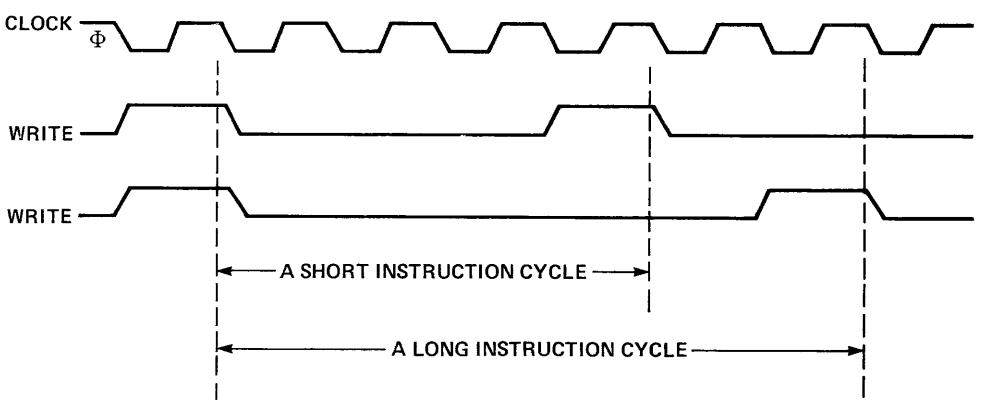


Figure 1-9. F8 Microcomputer Timing

tation of logic for a logic replacement product. For this reason, logic distribution differs significantly when the F8 is compared with predecessor microcomputers.

It would appear to be a truism that the most important features of any application is its complexity. Therefore it makes sense to implement logic on devices in terms of application complexity, rather than in terms of traditional computer function. The F8 microcomputer system is the only one that takes this approach. Refer to Figure 1-10. Two F8 devices, the 3850 CPU and the 3851 PSU, together implement all the basic necessities of a small microcomputer system. For the F8 to be implemented in this intelligent way, these three features of the F8 had to contradict old computer concepts:

1. A small amount of RAM is implemented within the CPU as a scratchpad memory.
2. Memory addressing logic is implemented in the memory devices and not in the CPU.
3. I/O ports are implemented in the CPU and memory devices, rather than requiring separate I/O devices.

A logic designer is unlikely to see anything especially revolutionary about the logic distribution illustrated in Figure 1-10. Indeed, the only revolutionary thing about it is that it does not conform to traditional data processing computer concepts; but then,

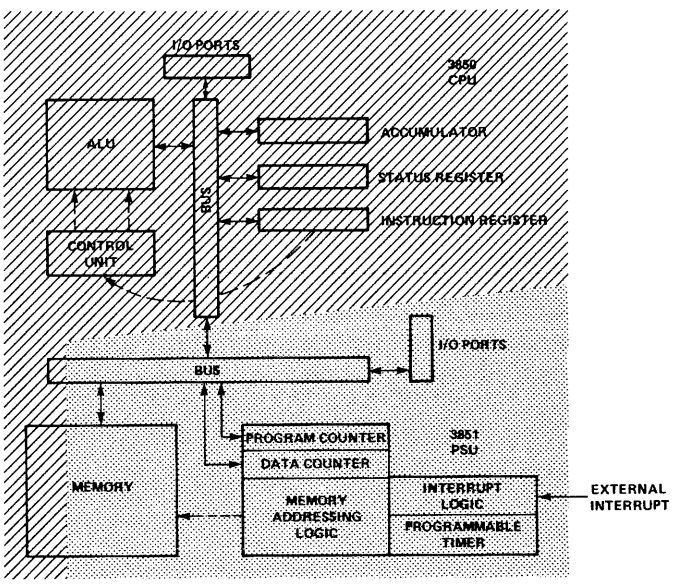


Figure 1-10. A Two-Device Configuration Consisting of a 3850 CPU and a 3851 PSU

traditional data processing computers were never built to replace discrete logic.

The 3850 CPU must be present in all F8 configurations, and as a minimum either a 3851 PSU or a memory interface device (3852 or 3853) with standard ROM/PROM. Any of the three memory oriented circuits may be used together or singly in the same system. In addition, when system needs dictate, multiple units of the same type may be used. For example, if 2K words of ROM, 64 bytes of RAM, and six I/O ports are required, one 3850 and two 3851s may comprise the system. If larger RAM or ROM is required the system may consist of one 3850, one 3853 and the appropriate amount of standard memory packages.

As soon as an application becomes complex enough to require large amounts of RAM, or when the custom marking process associated with the 3851 is too cumbersome, the 3852 Dynamic Memory Interface (DMI) or the 3853 Static Memory Interface (SMI) devices may be used to replace the 3851 PSU. Each of these devices interpret control signals output by the 3850 CPU in order to generate the standard address and control signals required by off-the-shelf dynamic and static memory devices. Figures 1-11 and 1-12 illustrate the logic which is provided by the 3852 DMI and 3853 SMI devices, respectively. A 4-chip system, which is the equivalent of the 2-chip system but which does not require the customized 3851 PSU, can be built of a 3850 CPU, 3853 SMI, 3861 PIO, and standard ROM/PROM.

Applications that require additional I/O and interrupt ability but that do not need the program storage of the 3851 Program Storage Unit (PSU) can use the 3861 Peripheral Input/Output (PIO). This device interprets the control signals of the 3850 CPU to drive two I/O ports of 8 bits each. Interrupt logic and a programmable timer are also on the device. Figure 1-13 illustrates the logic provided by the 3861 PIO device.

The final device of the F8 microcomputer system is the 3854 Direct Memory Access (DMA) device. Figure 1-14 illustrates the logic implemented on this device. The 3854 DMA device, working in conjunction with the 3852 DMI device, generates the control and address signals needed to implement data transfer between memory and external logic. This data transfer occurs in parallel with normal events occurring within the F8 microcomputer system.

Table 1-1 illustrates how F8 devices provide capabilities needed by various applications. Note the strong correlation between number of devices and complexity of application.

1.4 F8 BUS STRUCTURE

F8 system components are connected through an F8 system bus. This system bus is composed of the following elements:

- Data Bus (8 lines) DB0-DB7
- Control Lines (5 lines) ROMC0-ROMC4
- Clocks Φ, WRITE (2 lines), and
- Interrupt lines $\overline{\text{PRI IN}}$, $\overline{\text{PRI OUT}}$, $\overline{\text{INT REQ}}$ (3 lines)

DATA BUS (DB0-DB7)

These eight (8) bi-directional lines are used to transfer data, addresses, and input/output signals between various devices, including memories, in an F8 system. Within a system only one device may be driving signals on the Data Bus at any given time.

CONTROL LINES (ROMC0-ROMC4)

These five (5) lines transmit thirty-two different commands from the F8 CPU to the rest of the devices in the system. At the beginning of each cycle, the CPU decodes the instruction and a command is presented on these lines to indicate the operation to be performed during the current cycle by the rest of the system.

Φ, WRITE CLOCKS

These two (2) lines carry the necessary timing information required from the F8 CPU to the rest of the system. 'Φ' is the basic clock—normally 2 MHz—generated by the F8 CPU timing circuits. 'WRITE' clock is a signal having a pulse-width of one 'Φ' period, and having a period of either four 'Φ' clock cycles or six 'Φ' clock cycles. All the registers and flip-flops (except for the scratchpad, which is static) in the F8 devices are either updated or refreshed during the active state of the 'WRITE' clock.

INTERRUPT LINES (3)

$\overline{\text{INT REQ}}$ is a signal from the other F8 devices to the CPU that indicates that an interrupt is requested. $\overline{\text{PRIORITY IN}}$ and $\overline{\text{PRIORITY OUT}}$ are signals used to connect various interrupt circuits into a priority daisy chain.

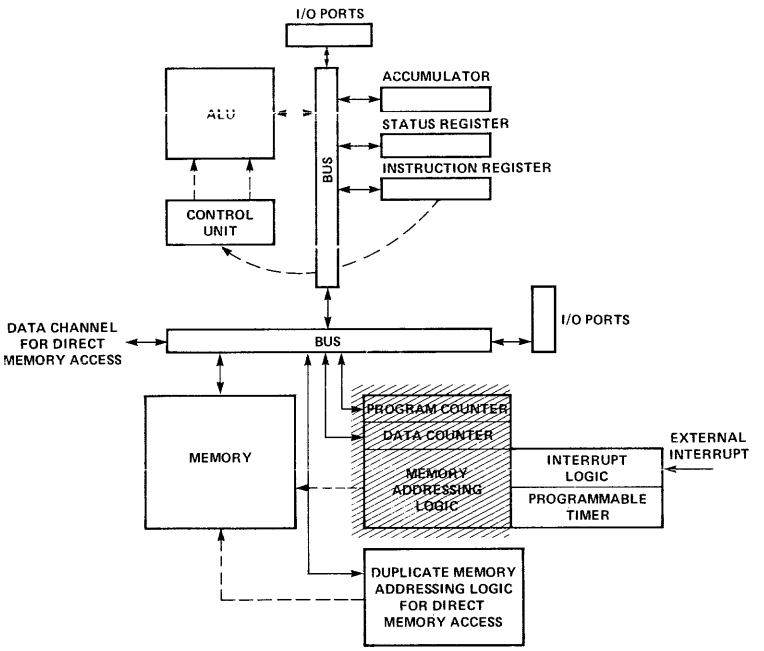


Figure 1-11. 3852 Dynamic Memory Interface Device Logic

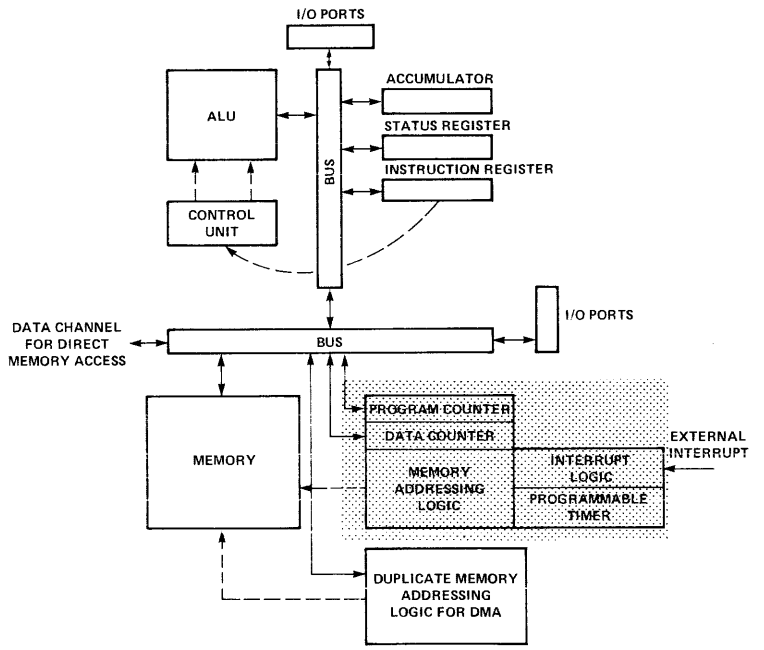


Figure 1-12. 3853 Static Memory Interface Device Logic

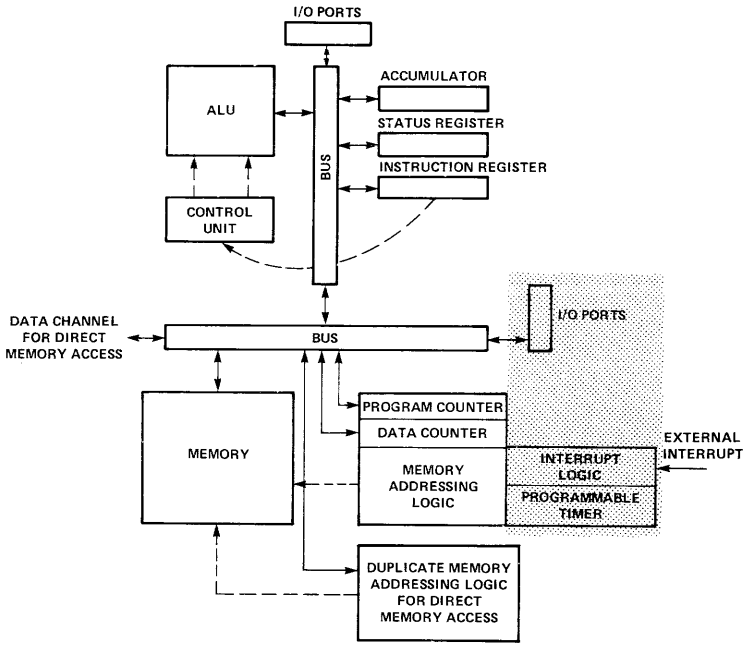


Figure 1-13. 3861 Peripheral Input/Output Device Logic

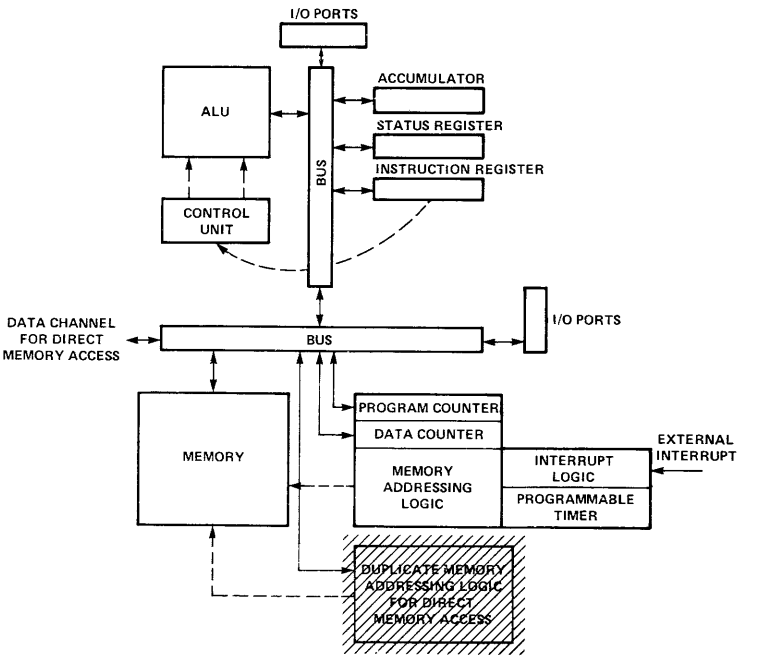


Figure 1-14. 3854 Direct Memory Access Device Logic

Table 1-1. Number of F8 Devices that can be in a Standard, One CPU F8 Configuration

DEVICE	LOGIC PROVIDED				NUMBER IN SYSTEM**	COMMENTS
	BYTES OF MEMORY	I/O* PORTS	INTERRUPTS	TIMER		
3850 CPU	64 RAM	2			1	RAM is independent of external memory
3851 PSU	1024 ROM	2	1	1	0 to 64	Maximum memory in system equals 64 X 1024 bytes
3852 DMI					0 or 1	3852 DMI must be present for 3854 DMA to be present. Either 3852 DMI or 3853 SMI will be present in systems with external ROM/RAM. Up to 64K bytes can be addressed. Maximum memory equals 65,536 minus PSU memory. RAM/ROM/PROM may be intermixed.
3853 SMI			1	1	0 or 1	
3861 PIO		2	1	1	0 to 64	Ports, interrupt, and timer same as those of 3851 PSU
3854 DMA					0 to 4	Provides logic for a DMA channel. If present, a 3852 DMI must also be present.

*The number of I/O ports listed refers to the number of I/O ports available to external devices and excludes internal buffers addressed as I/O ports.

**These numbers apply to single CPU configurations that use none of the special expansion techniques described in Section 7 through 10.

1.5 F8 I/O AND INTERRUPT

I/O ports and the interrupt structure are distributed throughout the F8 system.

Characteristics common to all I/O ports in F8 devices can be summarized as follows:

- Each has a unique 8-bit I/O address
- Each can be used for output or input on a bit basis
- Latches in the port hold output data
- Output data is "wire-ORed" with input data (an internal pull-up is provided)
- Input and output are TTL compatible.

Interrupt capability allows external stimulus to stop normal processing and to force processing to an interrupt service routine.

The interrupt structure is also distributed throughout the F8 system. The interrupt sequence begins with the transition of an input line (External Interrupt); the F8 system takes over from that point to assign priorities, to generate an interrupt vector address, and to direct program execution to that interrupt location. Each PSU, PIO, and SMI has its own independent interrupt structure; each has its own external interrupt input and interrupt vector. The CPU has the task of directing program execution to the vector address; the other F8 chips have the task of accepting the interrupt input and of generating the interrupt vector. Interrupts may also be generated by internal timers.

1.6 F8 INSTRUCTION SET

The instruction set of a microprocessor is the tool used to shape the microprocessor to fit a particular

application, just as NAND gates and flip-flops were the tools used in discrete TTL logic design. The designer must make a transition as he picks up the tools of a microprocessor design. With TTL logic, the design concentrated on state diagrams and on the conditions that cause transition between states. In microprocessor design, the state diagram is transformed into a flow chart, an outline of the serial stream of operations. Each operation is implemented with a set of instructions. States become equivalent to the contents of the microprocessor registers. One of the most significant registers of the microprocessor is the Program Counter (PC0). This register keeps track of the address of the next instruction to be executed. The NAND gate, which simultaneously tested a number of conditions to determine a state transition, is replaced by a serial sequence of conditional branch instructions, each of which alter the Program Counter contents if the specified condition is matched.

Figure 1-15 illustrates a simple problem solved in two ways—by TTL design and by microprocessor design. Two conditions, READY and ENABLE, are necessary before a signal START can activate a motor. The TTL design combines the two conditions in a NAND gate and clocks the NAND gate output through a flip-flop. The microprocessor design uses a flow chart in place of the logic symbols. The two input conditions, READY and ENABLE, are assumed connected to two different I/O ports. Compare instructions set the zero status flag if the input instruction loaded the desired bits into the accumulator. Conditional branch instructions in turn test the status flag. Only if both input signals are "1" does the program get down to the step that uses the output instruction to set the signal START to "1". This simple problem exemplifies the effectiveness of the microprocessor approach to system design.

The F8 instruction set is categorized into four groups:

- Input/Output
- Arithmetic/Logical: Accumulator group
  Immediate reference group
  Scratchpad reference group
  Memory reference group
- Address Register Control: Program counter group
  Conditional branch group
  Data counter group
  Q, K, H and J register group
- ISAR and status control

The following paragraphs are an overview of the instruction set; a more complete discussion of programming is found in the book "A Guide to Programming the F8 Microcomputer."

Just as the two chip system shown in Section 1-3 emphasized I/O capability and compact systems, so too does the F8 instruction set emphasize I/O, bit manipulation, fast powerful scratchpad manipulation, and short one byte instructions. The F8 logic design complements the power of the instruction set; for instance, branch instructions save bytes and time by using relative addressing. Low address I/O ports can be accessed by one byte instructions. Twelve scratchpad registers are directly addressable.

Input/Output Instructions — Within the input/output category are the instructions that bring data from the external world, via the I/O port, into the accumulator. Once into the accumulator, the arithmetic/logical instructions can begin manipulating and testing the data bits. The input/output instructions are given in Table 1-2. Input/output instructions use two modes of I/O port addressing:

Short Direct Addressing: One byte input/output instructions have the op code as the high order 4-bit and the I/O address as the low 4-bits. Ports 00 through 0F (HEX) may be addressed.

Long Direct Addressing: In these two byte instructions, the I/O port address is contained in the second byte of the instruction. Any port may be accessed.

Arithmetic/Logical Instructions — Within the arithmetic/logical category are the instructions that manipulate data bytes. Instructions within this category are given in Table 1-3. Functions available are:

ACC ← ACC + B (Binary Add)
ACC ← ACC + B (Decimal Add)
ACC ← ACC $\triangle$ B (Logical AND)
ACC ← ACC $\triangledown$ B (Logical OR)
ACC ← ACC + B (Exclusive OR)
Status Flags ← B - ACC (Two's Complement Subtract)
ACC ← $\overline{\text{ACC}}$ (One's Complement)
ACC ← (Logical Shifts) ACC

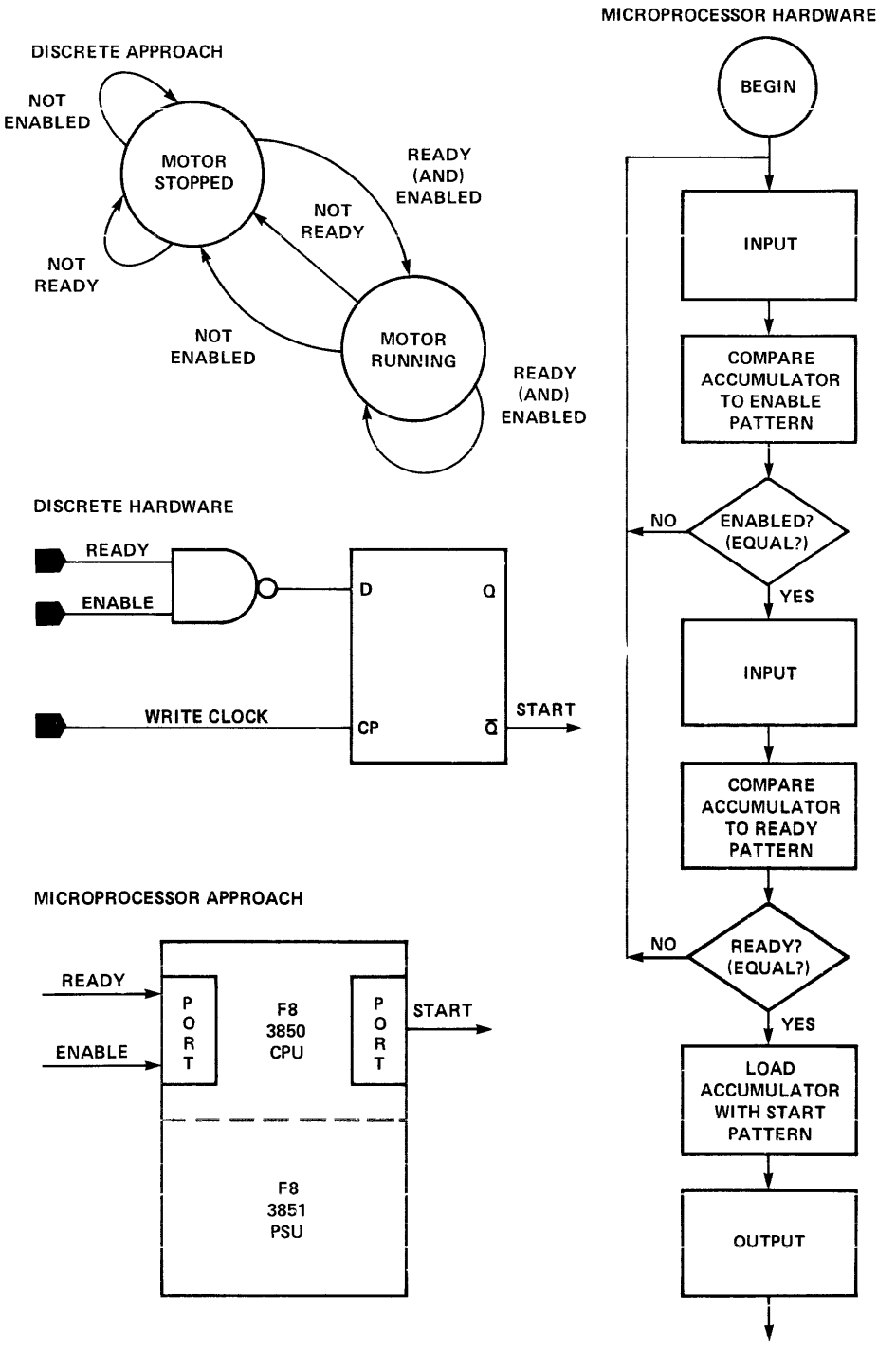


Figure 1-15. Discrete vs. Microprocessor Approach

The accumulator is involved in all of the above operations. Two variable operations, such as ADD, always use the accumulator as one of the operands. The result of the operation is always loaded back into the accumulator. The second operand for the arithmetic/logical functions can have three sources:

- Scratchpad memory within the 3850 CPU for the scratchpad reference group.
- Memory within a 3851 PSU, or standard RAM/ROM for the memory reference group.
- Immediate values stored with the instruction as its second byte for the immediate reference group.

Addressing modes for the arithmetic/logical instructions are:

Scratchpad Register Group –

Direct Register Addressing – This mode of addressing may be used to directly reference some scratchpad registers. By including the register number in the one-byte instruction, the first 12 of the 64 scratchpad registers may be referenced directly.
Indirect Register Addressing – All 64 scratchpad registers may be indirectly referenced, using the Indirect Scratchpad Address Register (ISAR) in the CPU. This 6-bit register acts as a pointer to the scratchpad memory. It is either incremented, decremented or left unchanged after accessing the scratchpad register.

Memory Reference Group –

Indirect Memory Addressing – A 16-bit indirect address register, the Data Counter, points to either data or constants in bulk memory. A group of one-byte instructions is provided to manipulate this area of memory. These instructions imply that the Data Counter is pointing to the desired memory byte. The Data Counter is self-incrementing, allowing for an entire data field to be scanned and manipulated without requiring special instructions to increment its content.

Immediate Reference Group –

Short Immediate Addressing – Instructions whose addressing mode is Short Immediate have the instruction op code as the first four bits and the operand as the last four bits. They are all one-byte instructions.
Long Immediate Addressing – In these two-byte instructions, the first instruction byte is the op code and the second byte is the 8-bit operand.

Address Register Control Instructions – The instructions of this category are given in Table 1-4.

These instructions manipulate the Program Counter (PC0), the stack register (PC1), the primary Data Counter (DC0), and the secondary Data Counter (DC1).

Normally the Program Counter increments after each instruction, addressing one instruction after another. Jumps and branches are used to alter this sequence; jumps load a 16-bit value into PC0 while branches add or subtract an offset to PC0.

Subroutines make use of instructions that save and that reload the contents of the Program Counter. A subroutine is a segment of programming that performs a specific task which is needed at several different places in a program. Each time a subroutine call is made, the processor executes the instruction stream of the subroutine. When execution of the subroutine is complete, control returns to the instruction immediately following the subroutine call. Special instructions are used to call a subroutine–push instructions. A push is like a jump except that the next address after the push instruction is saved in the Stack Register (PC1).

The POP instruction reloads the Program Counter (PC0) from the Stack Register (PC1). Thus a return to the main program after execution of a subroutine is accomplished.

Subroutines may be nested, which means that a call to a second subroutine may occur during the execution of a first subroutine. Nested subroutines require that more than one value from the Program Counter be saved. The F8 microprocessor dedicates pairs of stratchpad registers in the 3850 CPU which are used to save and then to reload the Stack Register (PC1) as well as the Data Counter (DC0). The interrelationship of the scratchpad registers and the address registers is shown in Figure 1-16.

Other single byte instructions of the address register control category link the K and Q scratchpad registers to the accumulator so that they may be stored elsewhere. Thus, a stack may be created. The PK and LR P0, Q instructions can be used to push or jump addresses dynamically created during the program's execution.

Conditional branch instructions also modify the Program Counter. The branch instructions test selected status flags and will either modify the program counter if the test is satisfied or will otherwise continue to the next sequential instruction.

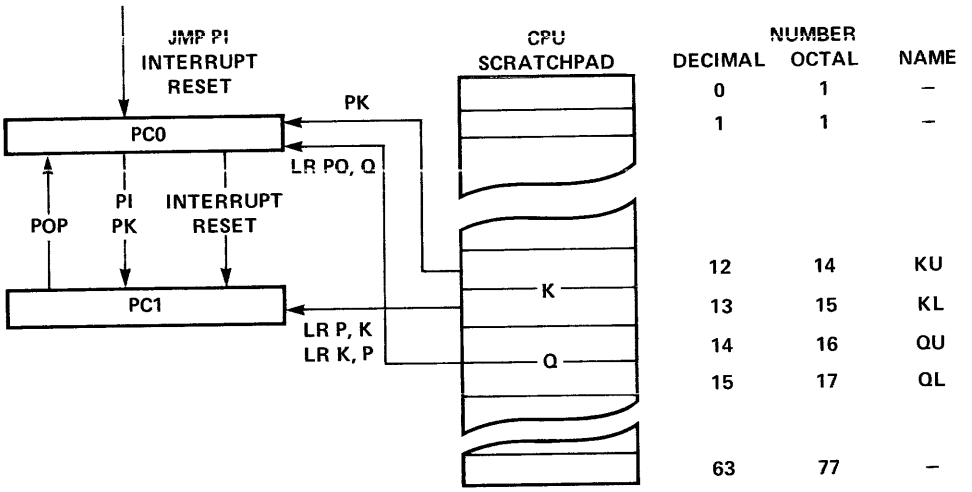


Figure 1-16. Instructions Linking Program Counter

Since branches are the decision making tools, they are frequently used. The F8 has a large number of branch instructions, providing ease of programming. A test can be made for either a single status flag being true or false or for a combination of flags, such as "branch if a result is positive OR produced a carry." The four status flags are overflow, zero, carry, and sign. Branch instructions can move the Program Counter forwards or backwards.

The primary Data Counter register (DC0) supplies the indirect address used by the memory referencing arithmetic/logical instructions. The Data Counter may be used to access data bytes or tables of data, such as a table of 256 entries that matches ASCII values to EBCDIC values. Sometimes the location of the data byte is fixed; in such a case the DCI instruction is used to load DC0 using the 2-byte immediate operand. At other times, such as table look-ups, the value loaded into DC0 is not fixed but rather is the result of a calculation made during program execution. In such an instance, the Data Counter may be loaded from the scratchpad registers. Table 1-4 gives the instructions that manipulate DC0. Some of these instructions link DC0 to dedicated pairs of scratchpad registers. The scratchpad registers and their linkage to DC0 are given in Figure 1-17.

The linkages of DC0 to the scratchpad are used for calculating values to be loaded into DC0; they are also used for loading and storing DC0. Notice that the H register pair is within the direct addressing range of the arithmetic/logical scratchpad instructions. The registers of the Q pair are linked by one-byte byte load and store instructions to the accumulator.

The ADC instruction adds the accumulator to the contents of the Data Counter. The Q register pair is linked to both the Data Counter and the program counter; this path can be used to transfer a value from DC0 to PC0 after a jump value is calculated in DC0 using the ADC instruction.

The memory interface devices have two data counters, DC0 and DC1. The secondary data counter, DC1, is useful for programs working with two blocks of memory.The H and Q register pairs come into use if three blocks are being handled, such as when going "A (+) B (into) C."

The addressing modes used in the program control category are:

Implied Addressing — The data for these one-byte instructions are implied by the actual instruction.

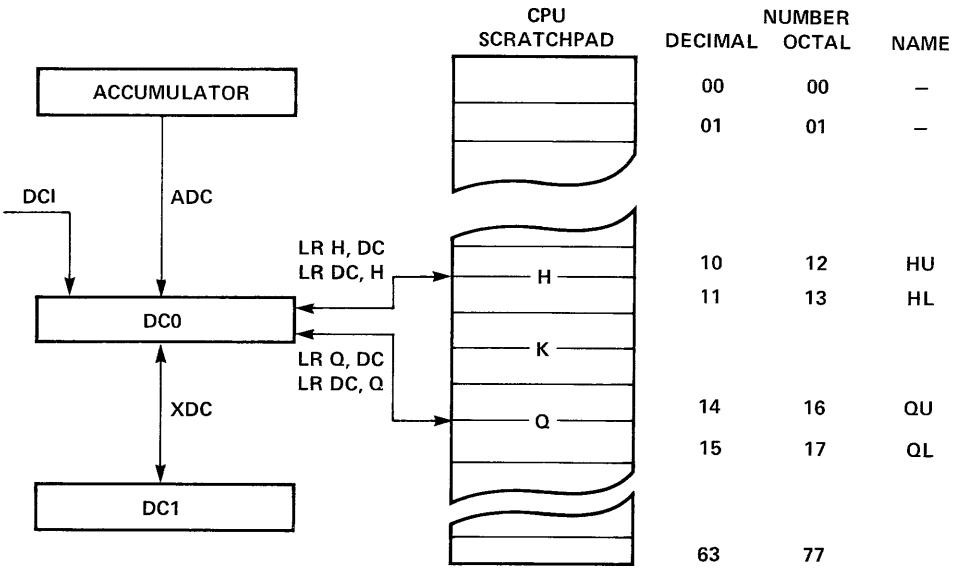


Figure 1-17. Instructions Linking Data Counter

For example, the POP instruction automatically implies that the content of the Program Counter will be set to the value contained in the Stack Register.

Relative Addressing – All F8 branch instructions use the relative addressing mode. Whenever a branch is taken, the Program Counter is updated by an 8-bit relative address contained in the second byte of instructions. A branch may extend 128 locations forward or 127 locations backward from the address of the branch instruction.

ISAR and Status Control Instructions – The ISAR and status control category of instructions manipulate the Indirect Scratchboard Address Register (ISAR) and the Status (W) register. The address for the scratchpad can be provided either directly (for the first 12 registers) or is provided by the 6-bit ISAR. The scratchpad is a powerful resource in the F8 system; convenient and flexible control of the ISAR lets the program utilize this power. Instructions of this category are given in Table 1-5. Figure 1-18 shows the register linkages. The ISAR can be loaded into two 3-bit segments using LISU and LISL instructions or all 6 bits can be loaded from the accumulator. It is handy to be able to load half of the ISAR independently of the other half; the operation "A (+) B (into) C," where A, B, and C are multi-byte arrays in scratchpad, is easily done by changing only the 3 high bits of ISAR. The ISAR can be tested by the BR7 instruction to test the lower 3 bits for "not 7." The ISAR is also linked to the accumulator for more general testing and address calculating.

The status register (W) is linked to a dedicated scratchpad register J (decimal 9). The linkage is used if it is necessary to save and restore the status, such as during interrupt servicing.

The addressing modes used in the ISAR and status control category are:

Short Immediate Addressing – Used by LISU and LISL instructions. The 3-bit operand is part of the 8-bit op code of the instruction.

Implied Addressing – The register and scratchpad addresses are implied by these one-byte instructions.

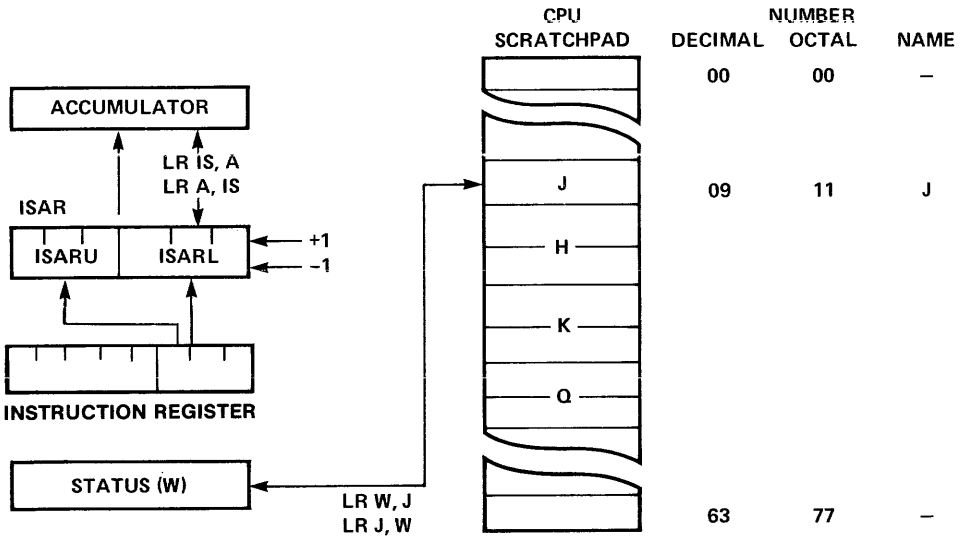


Figure 1-18. Instructions Linking ISAR, Status

Table 1-2. Input/Output Instructions

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	TIME
INPUT	IN	aa	ACC←(INPUT PORT aa)	26aa	2	4
INPUT SHORT	INS	a	ACC←(INPUT PORT a)	Aa	1	4**
OUTPUT*	OUT	aa	OUTPUT PORT aa←(ACC)	27aa	2	4
OUTPUT SHORT*	OUTS	a	OUTPUT PORT a←(ACC)	Ba	1	4**
DISABLE INTERRUPT	DI		RESET ICB	1A	1	2
ENABLE INTERRUPT*	EI		SET ICB	1B	1	2
NO-OPERATION	NOP		$PC_0 \leftarrow (PC_0) + 1$	2B	1	1

*Privileged instruction: no interrupt may occur between this instruction and whatever instruction follows.
**Execution time for CPU ports is 2 unit times.

Table 1-3. Arithmetic/Logical Instructions

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	TIME
	ACCUMULATOR GROUP INSTRUCTIONS					
ADD CARRY	LNK		ACC←(ACC) + CRY	19	1	1
CLEAR	CLR		ACC←H'00'	70	1	1
COMPLEMENT	COM		ACC←(ACC) ⊕ H'FF'	18	1	1
INCREMENT	INC		ACC←(ACC) + 1	1F	1	1
SHIFT LEFT ONE	SL	1	SHIFT LEFT 1	13	1	1
SHIFT LEFT FOUR	SL	4	SHIFT LEFT 4	15	1	1
SHIFT RIGHT ONE	SR	1	SHIFT RIGHT 1	12	1	1
SHIFT RIGHT FOUR	SR	4	SHIFT RIGHT 4	14	1	1
	IMMEDIATE REFERENCE INSTRUCTIONS					
ADD IMMEDIATE	AI	ii	ACC←(ACC) + H'ii'	24ii	2	2.5
AND IMMEDIATE	NI	ii	ACC←(ACC) △ H'ii'	21ii	2	2.5
OR IMMEDIATE	OI	ii	ACC←(ACC) ▽ H'ii'	22ii	2	2.5
EXCLUSIVE-OR IMMEDIATE	XI	ii	ACC←(ACC) ⊕ H'ii'	23ii	2	2.5
COMPARE IMMEDIATE	CI	ii	H'ii' + ($\overline{ACC}$) + 1	25ii	2	2.5
LOAD IMMEDIATE	LI	ii	ACC←H'ii'	20ii	2	2.5
LOAD IMMEDIATE SHORT	LIS	i	ACC←H'0i'	7i	1	1
	SCRATCHPAD REFERENCE INSTRUCTIONS (See Notes)					
ADD BINARY	AS	r	ACC←(ACC) + (r)	Cr	1	1
ADD DECIMAL	ASD	r	ACC←(ACC) + (r)	Dr	1	2
AND	NS	r	ACC←(ACC) △ (r)	Fr	1	1
EXCLUSIVE-OR	XS	r	ACC←(ACC) ⊕ (r)	Er	1	1
DECREMENT	DS	r	r←(r) + H'FF'	3r	1	1.5
LOAD	LR	A,r	ACC←(r)	4r	1	1
STORE	LR	r,A	r←(ACC)	5r	1	1
	MEMORY REFERENCE INSTRUCTIONS					
ADD BINARY	AM		ACC←(ACC) + [(DC_0)]	88	1	2.5
ADD DECIMAL	AMD		ACC←(ACC) + [(DC_0)]	89	1	2.5
AND	NM		ACC←(ACC) △ [(DC_0)]	8A	1	2.5
LOGICAL OR	OM		ACC←(ACC) ▽ [(DC_0)]	8B	1	2.5
EXCLUSIVE-OR	XM		ACC←(ACC) ⊕ [(DC_0)]	8C	1	2.5
COMPARE	CM		[(DC_0)] + ($\overline{ACC}$) + 1	8D	1	2.5
LOAD	LM		ACC←[(DC_0)]	16	1	2.5
STORE	ST		[(DC_0)]←(ACC)	17	1	2.5

In all Memory Reference Instructions, the Data Counter is incremented: $DC_0 \leftarrow DC_0 + 1$.

Table 1-4. Address Register Control Instructions

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	TIME
	PROGRAM COUNTER INSTRUCTIONS					
JUMP*	JMP	aaaa	$PC_0 \leftarrow H'aaaa'$	29aaaa	3	5.5
BRANCH RELATIVE	BR	aa	$PC_0 \leftarrow [(PC_0) + 1] + H'aa'$	90aa	2	3.5
LOAD PROGRAM COUNTER	LR	PQ,Q	$PC_0U \leftarrow (r14)$; $PC_0L \leftarrow (r15)$	0D	1	4
CALL TO SUBROUTINE IMMEDIATE*	PI	aaaa	$PC_1 \leftarrow (PC_0)$; $PC_0 \leftarrow H'aaaa'$	28aaaa	3	6.5
CALL TO SUBROUTINE*	PK		$PC_0U \leftarrow (r12)$; $PC_0L \leftarrow r13)$; $PC_1 \leftarrow PC_0)$	0C	1	4
LOAD STACK REGISTER	LR	P,K	$PC_1U \leftarrow (r12)$; $PC_1L \leftarrow (r13)$	09	1	4
STORE STACK REGISTER	LR	K,P	$r12 \leftarrow (PC_1U)$; $r13 \leftarrow (PC_1L)$	08	1	4
RETURN FROM SUBROUTINE**	POP		$PC_0 \leftarrow (PC_1)$	1C	1	2
	CONDITIONAL BRANCH INSTRUCTIONS*					
BRANCH IF POSITIVE	BP	aa	$PC_0 \leftarrow [(PC_0) + 1] + H'aa'$ if SIGN = 1	81aa	2	3.5
BRANCH IF CARRY	BC	aa	$PC_0 \leftarrow [(PC_0) + 1] + H'aa'$ if CRY = 1	82aa	2	3.5
BRANCH IF ZERO	BZ	aa	$PC_0 \leftarrow [(PC_0) + 1] + H'aa'$ if ZERO = 1	84aa	2	3.5
BRANCH IF TRUE TEST	BT	taa	$PC_0 \leftarrow [(PC_0) + 1] + H'aa'$ if any test bit is true (OR) t = TEST CONDITION 2^2 ZERO | 2^1 CRY | 2^0 SIGN	8taa	2	3.5
BRANCH IF NEGATIVE	BM	aa	$PC_0 \leftarrow [(PC_0) + 1] + H'aa'$ if SIGN = 0	91aa	2	3.5
BRANCH IF NO CARRY	BNC	aa	$PC_0 \leftarrow [(PC_0) + 1] + H'aa'$ if CARRY = 0	92aa	2	3.5
BRANCH IF NOT ZERO	BNZ	aa	$PC_0 \leftarrow [(PC_0) + 1] + H'aa'$ if ZERO = 0	94aa	2	3.5
BRANCH IF NO OVERFLOW	BNO	aa	$PC_0 \leftarrow [(PC_0) + 1] + H'aa'$ if OVF = 0	98aa	2	3.5
BRANCH IF FALSE TEST	BF	taa	$PC_0 \leftarrow [(PC_0) + 1] + H'aa'$ if all test bits are false (AND) t = TEST CONDITION 2^3 OVF | 2^2 ZERO | 2^1 CRY | 2^0 SIGN	9taa	2	3.5
BRANCH IF ISAR (LOWER) ≠7	BR7	aa	$PC_0 \leftarrow [(PC_0) + 1] + H'aa'$ IF ISARL ≠ 7 $PC_0 \leftarrow (PC_0) + 2$ if ISARL=7	8Faa —	2 —	2.5 2.0
	DATA COUNTER INSTRUCTIONS					
LOAD DC IMMEDIATE	DCI	aaaa	$DC_0 \leftarrow H'aaaa'$	24aaaa	3	6
LOAD DATA COUNTER	LR	DC,Q	$DCU \leftarrow (r14)$; $DCL \leftarrow (r15)$	0F	1	4
LOAD DATA COUNTER	LR	DC,H	$DCU \leftarrow (r10)$; $DCL \leftarrow (r11)$	10	1	4
STORE DATA COUNTER	LR	Q,DC	$r14 \leftarrow (DCU)$; $r15 \leftarrow (DCL)$	0E	1	4
STORE DATA COUNTER	LR	H,DC	$r10 \leftarrow (DCU)$; $r11 \leftarrow (DCL)$	11	1	4
ADD TO DATA COUNTER	ADC		$DC_0 \leftarrow (DC_0) + (ACC)$	8E	1	2.5
EXCHANGE DC	XDC		$DC_0 \rightleftharpoons DC_1$	2C	1	2

*Privileged instruction: no interrupt may occur between this instruction and whatever instruction follows.

**In all conditional branches except BR7, if the test condition is not met, then PC0←(PC0) + 2 and execution takes 3.0 unit times.

Table 1-4. Address Register Control Instructions (Continued)

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	TIME
	Q, K, H, AND J REGISTER INSTRUCTIONS					
LOAD	LR	A,KU	ACC←(r12)	00	1	1
LOAD	LR	A,KL	ACC←(r13)	01	1	1
LOAD	LR	A,QU	ACC←(r14)	02	1	1
LOAD	LR	A,QL	ACC←(r15)	03	1	1
LOAD	LR	A,HU	SEE LR A,r			
	LR	A,HL	INSTRUCTION,			
	LR	A,J	TABLE 1-3			
STORE	LR	KU,A	r12←(ACC)	04	1	1
STORE	LR	KL,A	r13←(ACC)	05	1	1
STORE	LR	QU,A	r14←(ACC)	06	1	1
STORE	LR	QL,A	r15←(ACC)	07	1	1
STORE	LR	HU,A	SEE LR r,A			
	LR	HL,A	INSTRUCTION,			
	LR	J,A	TABLE 1-3			

Table 1-5. ISAR and Status Control Instructions

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	TIME
LOAD ISAR LOWER	LISL	a	ISARL←a	01101a**	1	1
LOAD ISAR UPPER	LISU	a	ISARU←a	01100a**	1	1
LOAD ISAR	LR	IS,A	ISAR←(ACC)	0B	1	1
STORE ISAR	LR	A,IS	ACC←(ISAR)	0A	1	1
LOAD STATUS REGISTER*	LR	W,J	W←(r9)	1D	1	2
STORE STATUS REGISTER	LR	J,W	r9←(W)	1E	1	1
BRANCH IF ISAR (LOWER) ≠7	BR7		(SEE CONDITIONAL BRANCH INSTRUCTIONS)			

*Privileged instruction: no interrupt may occur between this instruction and whatever instruction follows.
**Three bit octal digit.

Notes: *(for Tables 1-2 to 1-5)*

Time is in units of 4Φ clock periods (short instruction cycle) and equals 2.0 μS at Φ = 2 MHz. Each lower case character represents a Hexadecimal digit. Lower case denotes variables specified by programmer.

Function Definitions

Φ is replaced by
( ) the contents of
(-) Binary "1"s complement of
+ Arithmetic Add (Binary or Decimal)
⊕ Logical "OR" exclusive
V Logical "AND"
Δ Logical "OR" inclusive
H'' Hexadecimal digit

Register Names

a Address variable
A Accumulator
DC_0 Data counter #0
DC_1 Data counter #1
DCL Least significant 8 bits of data counter #0
DCU Most significant 8 bits of data counter #0
H Scratchpad register #10 and #11
i,ii Immediate operand

Notes: (Continued)

ICB	Interrupt control bit
ISAR,IS	Indirect scratchpad address register
ISARL	Least significant 3 bits of ISAR
ISARU	Most significant 3 bits of ISAR
J	Scratchpad register #9
K	Registers #12 and #13
KL	Register #13
KU	Register #12
PC_0	Program counter
PC_0L	Least significant 8 bits of program counter
PC_0U	Most significant 8 bits of program counter
PC_1	Stack register
PC_1L	Least significant 8 bits of stack register
PC_1U	Most significant 8 bits of stack register
Q	Registers #14 and #15
QL	Register #15
QU	Register #14
r	Scratchpad register (any address through 11)
W	Status register

Scratchpad Addressing Modes (Machine Code Format)

r = C	(Hexadecimal), register addressed by ISAR (unmodified)
r = D	(Hexadecimal), register addressed by ISAR; ISARL incremented
r = E	(Hexadecimal), register addressed by ISAR; ISARL decremented
r = F	(Hexadecimal), undefined
r = 0 through B	(Hexadecimal), register 0 through 11 addressed directly from the instruction

Status Register

CRY	Carry flag
OVF	Overflow flag
SIGN	Sign of result flag; equals "1" if result positive
ZERO	Zero flag

2.0 The 3850 CPU

THE 3850 CPU

Section 1 has described the concept of a microprocessor, how microprocessors may be used to replace discrete logic, and the overall organization and instruction set of the F8 Microprocessor system. This section describes the 3850 CPU.

The 3850 is the Central Processing Unit for the F8 system; it has more than 70 instructions in its instruction set, and it operates on 8-bit units of information. 3850 features include:

- N-channel Isoplanar MOS technology
- 2 μS cycle time
- 64 byte RAM on the CPU chip
- Two bi-directional, 8-bit I/O ports, with output latches
- 8-bit arithmetic and logic unit, supporting both binary and decimal arithmetic
- Interrupt control logic
- Power-on reset logic
- Clock generation logic within the CPU chip, with the following three modes of clock generation:
  - RC network
  - Crystal
  - External clock
- Over 70 instructions
- +5V and +12V power supplies
- Low power dissipation – typically less than 330 mW

The 3850 CPU is functionally illustrated in Figure 2-1; the figure shows logic functions, registers, data paths and device pins (with signal names); control signals within the CPU are not shown.

2.1 DEVICE ORGANIZATION

The 3850 CPU is described with reference to Figure 2-1.

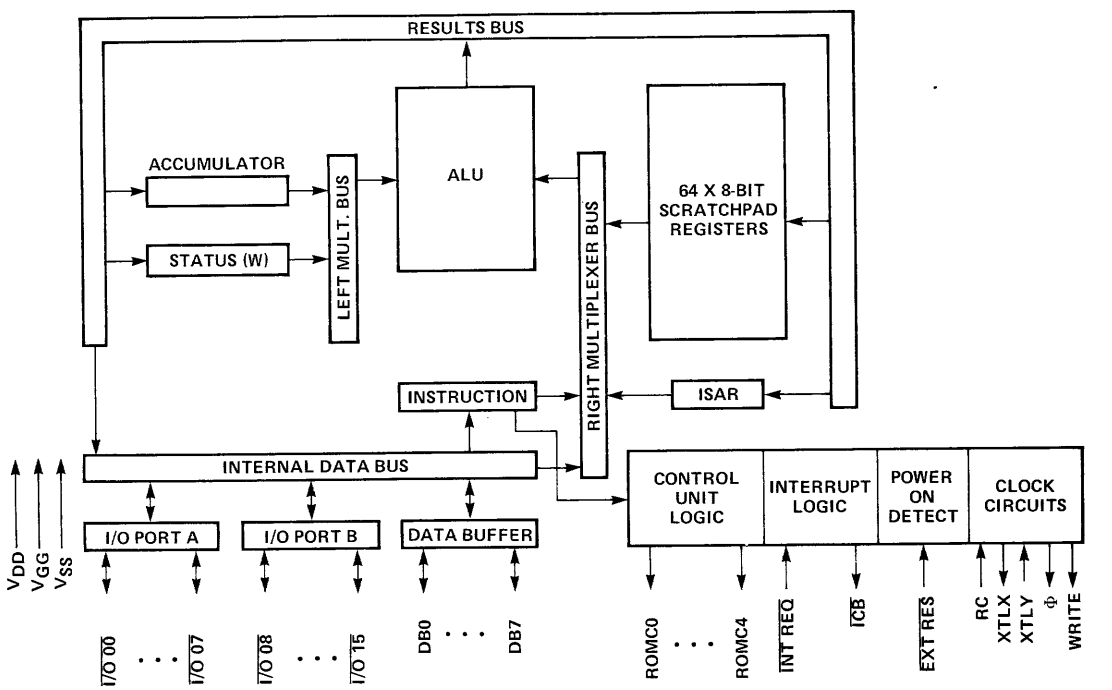


Figure 2-1. Logical Organization and Pins for the 3850 CPU

2.1.1 The Arithmetic and Logic Unit

The Arithmetic and Logic Unit (ALU) provides all data manipulating logic for the 3850 CPU; it contains logic which operates on a single, 8-bit source data word, or it combines two 8-bit words of source data, to generate a single, 8-bit result; additional information is reported in status flags, where appropriate.

Operations performed on two units of source data include addition, compare, and the Boolean operations (AND, OR, Exclusive-OR). The two sources are input to the ALU via the Left and Right Multiplexer Buses. The result is placed on the Result Bus.

Operations performed on a single 8-bit unit of source data include complement, increment, decrement, shift right, shift left and clear. The source may be input to the ALU via either the Left or the Right Multiplexer Bus. The result is placed on the Result Bus.

2.1.2 The Instruction Register

There are a number of registers within the CPU where data of various types may be stored.

The instruction register holds an 8-bit code which defines the operations to be performed by the CPU.

The contents of the instruction register are decoded by control unit logic, which generates signals to enable specific sequences of logic operations within the CPU chip; in response to the contents of the instruction register, the control unit also generates five signals, ROMC0 through ROMC4, which control operations throughout the microprocessor system.

2.1.3 The Accumulator

The Accumulator is a general purpose, 8-bit data register, which is the most common data source, and results destination for the ALU.

2.1.4 The Scratchpad and ISAR

The Scratchpad provides 64 8-bit registers which may be used as general purpose RAM memory.

The Indirect Scratchpad Address Register (ISAR) is a 6-bit register used to address the 64 scratchpad registers.

The first 16 scratchpad bytes can be identified by instructions without using the ISAR. The remaining scratchpad bytes are referenced via the ISAR; i.e., the ISAR is assumed to hold the address of the scratchpad byte which is to be referenced. Observe that the first 16 bytes of the scratchpad can also be referenced via the ISAR.

The ISAR should be visualized as holding two octal digits, HI and LO, as illustrated in Figure 2-2. This division of the ISAR is important, since a number of instructions increment or decrement the contents of the ISAR, when referencing scratchpad bytes via the ISAR. This makes it easy to reference a buffer consisting of contiguous scratchpad bytes. However, only the low order octal digit (LO) is incremented or decremented; thus ISAR is incremented from 0'27' to 0'20', not to 0'30'. Similarly, ISAR is decremented from 0'20' to 0'27', not to 0'17'. This feature of the ISAR is very useful in that it greatly simplifies many program sequences. (The notation 0'nn' is used to specify an octal number.)

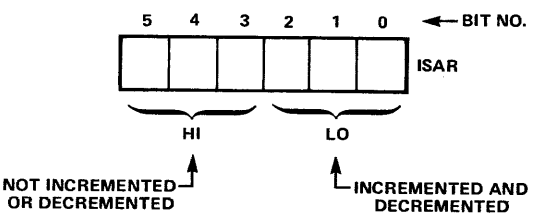


Figure 2-2. The ISAR Register

As illustrated in Figure 2-3, selected scratchpad registers are reserved for direct communication with other registers within the F8 system.

Scratchpad register 9 (0'11') is used as temporary storage for the CPU status register (also called the W register), which is described in Section 2.1.5.

Scratchpad registers 10 through 15 (0'12' through 0'17') communicate directly with data and program memory address registers which are maintained on the 3851, 3852 and 3853 chips. Figure 2-3 identifies the data transfers which may be implemented by executing a single F8 instruction. For example, the illustration:

W register of 3850 CPU ⟷ J

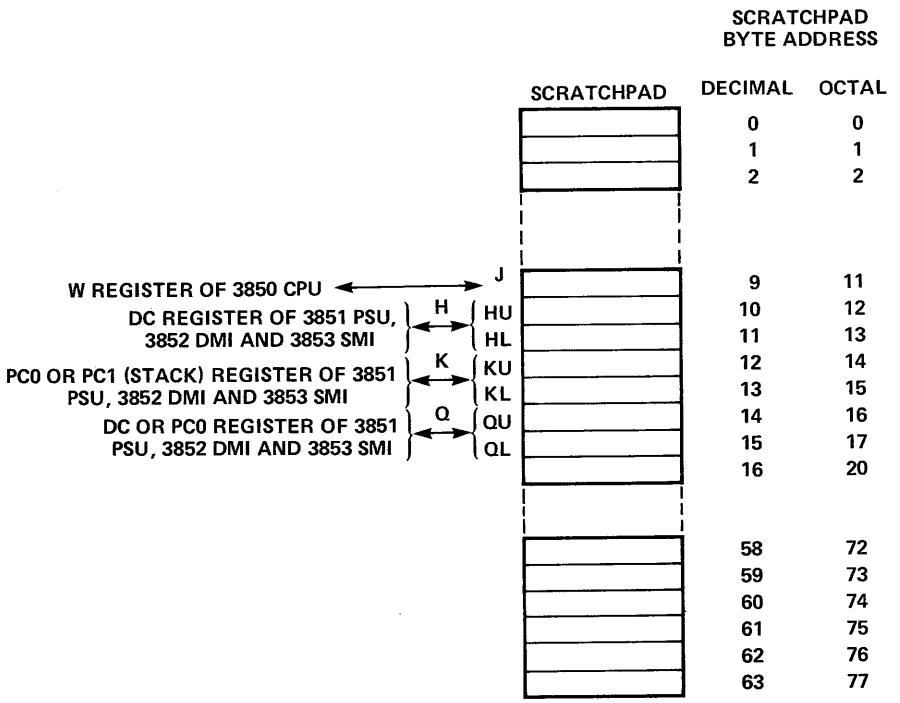
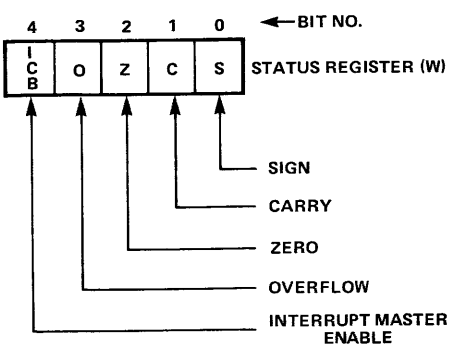


Figure 2-3. The 3850 CPU Scratchpad Registers

means that a single instruction can move the contents of the W (or status) register to scratchpad register 9 (also called the J register). Another single instruction can move data in the opposite direction.

2.1.5 The Status Register

The status register (also called the W register) holds five status flags as follows:



The way in which each status flag (or bit) is used is described next, and is summarized in Table 2-1.

Note that status flags are selectively modified following execution of different instructions. Table 2-7 defines the way in which individual F8 instructions modify status flags.

SIGN (S BIT)

When the results of an ALU operation are being interupreted as a signed binary number, the high order bit (bit 7) represents the sign of the number. At the conclusion of instructions that may modify the accumulator bit 7, the S bit is set to the complement of the accumulator bit 7.

CARRY (C BIT)

The C bit may be visualized as an extension of an 8-bit data unit, i.e., the ninth of a 9-bit data unit. When two bytes are added, and the sum is greater

than 255, then the carry out of the high order bit appears in the C bit. Here are some examples:

```
                C 76543210  ← Bit Number
Accumulator contents:  01100101
Value added:           01110110
Sum: 0                 11011011
```

There is no carry, so C is reset to 0.

```
                C 76543210  ← Bit Number
Accumulator contents:  10011101
Value added:           11010001
Sum: 1                 01101110
```

There is a carry, so C is set to 1.

ZERO (Z BIT)

The Z bit is set whenever an arithmetic or logical operation generates a zero result. The Z bit is reset to 0 when an arithmetic or logical operation could have generated a zero result, but did not.

OVERFLOW (0 BIT)

When the results of an ALU operation are being interpreted as a signed binary number, since the high order bit (bit 7) represents the sign of the number, some method must be provided for indicating carries out of the highest numeric bit (bit 6). This is done using the 0 bit. After arithmetic operations, the 0 bit is set to the Exclusive-OR of carries out of bits 6 and bits 7. The fact that this simplifies signed binary arithmetic is described in the Guide to Programming the F8. Here are some examples:

```
                76543210  ← Bit Number
Accumulator contents:  10110011
Value added:           01110001
Sum:                   00100100
                       1
```

There is a carry out of bit 6 and out of bit 7, so the 0 bit is reset to 0 ($1 \oplus 1 = 0$). The C bit is set to 1.

```
                76543210  ← Bit Number
Accumulator contents:  01100111
Value added:           00100100
Sum:                   10001011
```

There is a carry out of bit 6, but no carry out of bit 7; the 0 bit is set to 1 ($1 \oplus 0 = 1$). The C bit is reset to 0.

INTERRUPTS (ICB BIT)

External logic can alter program execution sequence within the CPU by interrupting ongoing operations, as described in Section 3.6, however, interrupts are allowed only when the ICB bit is set to 1; interrupts are disallowed when the ICB bit is reset to 0.

Table 2-1 A Summary of Status Bits

OVERFLOW	=	$CARRY_7 \oplus CARRY_6$
ZERO	=	$\overline{ALU_7} \wedge \overline{ALU_6} \wedge \overline{ALU_5} \wedge \overline{ALU_4} \wedge \overline{ALU_3} \wedge \overline{ALU_2} \wedge \overline{ALU_1} \wedge \overline{ALU_0}$
CARRY	=	$CARRY_7$
SIGN	=	$\overline{ALU_7}$

2.1.6 The Control Unit

The Control Unit decodes the contents of the Instruction Register, and generates two sets of control signals.

A set of internal control signals enable all logic sequences within the CPU chip. The F8 user has no control over these signals, and needs to know nothing about them.

Five control signals, named ROMC0 through ROMC4 are output by the control unit to identify operations which other chips of the F8 family must perform. These signals are described later in this section, and are summarized in Table 2-5.

2.1.7 Interrupt Logic

This logic handles interrupt requests, as described in Section 2.4.7.

2.1.8 Power On Detect

When the $\overline{EXT\ RES}$ (External Reset) signal is pulled low and then returned high, or when power is turned on, the "power on detect" logic sets the PC registers to 0, causing a program origined at memory location 0 to be executed. Also, the Interrupt Control status bit is set low, inhibiting interrupt acknowledgement. The system is locked in an idle state while $\overline{EXT\ RES}$ is held low.

2.1.9 Clock Circuits

Clock circuit logic is part of the 3850 CPU chip; it generates timing for the entire microcomputer system, as described in Section 2.3. WRITE is high for one Φ period at the end of each F8 instruction cycle and, hence, its falling edge defines the beginning of the next cycle. There are 4 or 6 Φ periods in an instruction cycle; the choice between short and long cycle is made by the CPU and is dependent on the instruction being executed.

2.1.10 The Data Bus

All data and address information is transferred between the 3850 CPU and other devices of the F8 devices via the eight data lines D0-D7.

Since the F8 is an 8-bit oriented microprocessor, having an 8-bit data bus is reasonable. All memory addresses, however, require 16 bits, which would imply transferring addresses in 8-bit nibbles; because of the unique architecture of the F8 system, this is not a consequential liability. Since memory addressing logic is located on the memory (and memory interface) devices, 16-bit memory addresses only need to be transferred across the 8-bit data bus in these three circumstances:

1. When a three-byte instruction specifies a memory address in the second and third bytes.
2. When data is being moved between DC or PC registers and associated scratchpad registers.
3. During the interrupt acknowledge sequence, when the interrupt vector is pushed onto PC0.

The F8 system's unique memory addressing logic is described in more detail in Section 2.4.

2.1.11 I/O Ports

The 16 address pins which most microprocessors require are used by the 3850 for two I/O ports. Data may be transferred, via these two I/O ports, between the 3850 CPU and logic external to the microprocessor system.

While other F8 devices provide additional I/O ports, the two I/O ports on the 3850 CPU execute data transfers twice as fast, since they do not use the external Data Bus.

Observe that the data path between the accumulator and the two CPU I/O ports is entirely within the 3850 CPU chip.

2.2 SIGNAL DESCRIPTIONS AND ELECTRICAL CHARACTERISTICS

Figure 2-4 illustrates the 3850 CPU device pins. Signal names agree with Figure 2-1, and are summarized in Table 2-2.

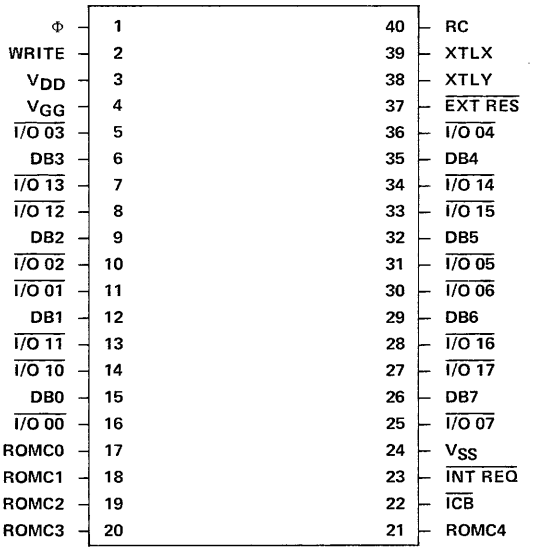


Figure 2-4. 3850 CPU Pin Assignments

Table 2-2. 3850 CPU Signals

PIN NAME	DESCRIPTION	TYPE
DB0-DB7	Data Bus Lines	Bi-directional (3-State)
Φ, WRITE	Clock Lines	Output
I/O 00-I/O 07	I/O Port Zero	Input/Output
I/O 10-I/O 17	I/O Port One	Input/Output
RC	RC Network Pin	Input
ROMC0-ROMC4	Control Lines	Output
EXT RES	External Reset	Input
INT REQ	Interrupt Request	Input
ICB	Interrupt Control Bit	Output
XTLX	Crystal Clock Line	Output
XTLY	External Clock Line	Input
V_{SS}, V_{DD}, V_{GG}	Power Lines	Input

2.2.1 Signal Descriptions

Φ and WRITE are clock outputs which drive all other devices in the F8 family. The relationship between Φ and WRITE is described in Section 2.3.

RC is connected to a Resistor/Capacitor network when using the RC mode for clock generation. It is grounded when using either the Crystal or External modes. See Section 2.3 for details.

XTLX and XTLY are used when generating the system clock in the Crystal mode. The XTLY pin is also used for operating in the External clock mode. See Section 2.3 for details.

ROMC0 through ROMC4 are control outputs which control logic operations for other devices in the F8 family. ROMC0 through ROMC4 assume a state early in each machine cycle and hold that state for the duration of the cycle. Table 2-5 summarizes the way in which CPU logic intents ROMC0 through ROMC4 to be interpreted.

DB0 through DB7 are bi-directional data bus lines which link the 3850 CPU with all other F8 chips in the system. These are multiplexed lines, used to transfer data and addresses, as described in Section 2.5.

$\overline{\text{I/O 00}}$ through $\overline{\text{I/O 17}}$ are Input/Output ports through which the CPU communicates with logic external to the microprocessor system. These signals are described in Section 3.4.3.

$\overline{\text{EXT RES}}$ may be used to externally reset the system. When this line is pulled low, a program, origined at memory address 0, is executed.

$\overline{\text{INT REQ}}$ is used to signal the CPU that an interrupt is being requested. The 3851 PSU and 3853 SMI devices contain logic to initiate interrupt requests by pulling $\overline{\text{INT REQ}}$ low. The CPU acknowledges interrupt requests by outputting appropriate ROMC signal sequences, as described in Section 3.6.

$\overline{\text{ICB}}$ indicates whether or not the CPU is currently ignoring the $\overline{\text{INT REQ}}$ line. If $\overline{\text{ICB}}$ is low, the CPU will respond to interrupt requests, if $\overline{\text{ICB}}$ is high, the CPU will ignore interrupt requests. This signal is described further in Section 3.6.

2.2.2 Electrical Specifications

Absolute maximum ratings (above which useful life may be impaired)

V_{GG}	+15V to -0.3V
V_{DD}	+7V to -0.3V
RC, XTLX and XTLY	+15V to -0.3V (RC with 5KΩ series resistor)
All other inputs	+7V to -0.3V
Storage temperature	-55°C to +150°C
Operating temperature	0°C to +70°C

Note: All voltages with respect to V_{SS}.

DC Characteristics: V_{SS} = 0V, V_{DD} = +5V ± 5%, V_{GG} = +12V ± 5%, T_A = 0°C to +70°C

SUPPLY CURRENTS

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
I_{DD}	V_{DD} Current		45	75	mA	f = 2 MHz, Outputs unloaded
I_{GG}	V_{GG} Current		12	30	mA	f - 2 MHz, Outputs unloaded

2.3 CLOCK CIRCUITS

A unique feature of the F8 CPU is that clock logic is an integral part of the 3850 CPU chip.

The 3850 CPU offers two alternate ways of generating a system clock; these are Crystal mode and External mode. The 3850-1 offers a third mode in addition; this is the RC mode.

2.3.1 Crystal Mode

Figure 2-5 shows the pin configuration for clock generation using the crystal mode. A crystal in the 1 to 2 MHz range is placed across the XTLX and XTLY pins, along with two capacitors (C_1 and C_2), to provide a highly precise clock frequency. The external crystal (and capacitors), together with internal circuitry, combine to form a parallel resonant crystal oscillator. C_1 and C_2 capacitors should be approximately 15 pf. The characteristics of the crystal used in this mode of clock generation can be summarized as follows:

Frequency: 1 to 2 MHz, typical AT cut
Mode of Oscillation: Fundamental
Operating Temperature Range: 0°C to +70°C
Drive Level: 10 mW
Frequency Tolerance: f_o = 1 or 2 MHz ± 1000 ppm @ C_L = 20 pf

2.3.2 External Mode

For F8 applications where synchronization with an external system clock is desired, the external clock mode may be used as shown in Figure 2-6. For example, a slave 3850 CPU may receive its timing from a master 3850 CPU, by having the master Φ

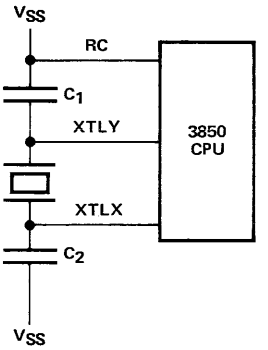


Figure 2-5. Crystal Mode Clock Generation

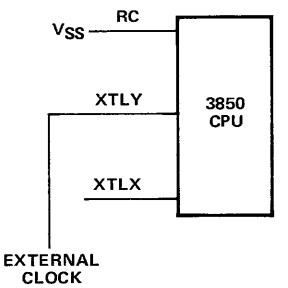


Figure 2-6. External Mode Clock Generation

output drive the slave XTLY input. See Section 14 for a more detailed description of multiple CPU applications.

Figure 2-8 illustrates the AC characteristics of the clock signal needed for external mode clock generation, plus the AC characteristics of the Φ and WRITE signals generated by the CPU.

2.3.3 RC Mode (3850-1 Only)

The RC mode, shown in Figure 2-7, is very inexpensive to implement. A resistor/capacitor network connected to the RC pin is the only external hardware required. The following formula can be used to compute the approximate clock period:

$$\text{Period} \simeq (1.2RC + 400\ \text{nS})$$

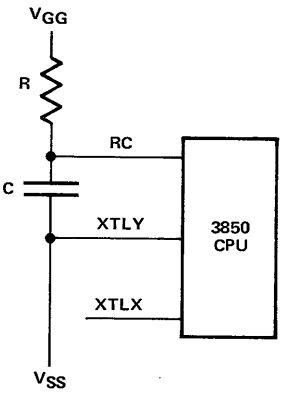


Figure 2-7. RC Mode Clock Generation

where: R is the resistance in KΩ (minimum resistance is 5 KΩ)
C is the capacitance in pf (including any parasitic capacitance)

2.3.4 Timing Signal Characteristics

In response to the clock inputs described in Sections 2.3.1, 2.3.2 and 2.3.3, the 3850 CPU outputs two timing signals, a clock signal Φ, and an instruction cycle control signal WRITE.

Referring to Figure 2-8, Φ is the signal used to synchronize the entire microprocessor system.

WRITE defines the duration of each machine cycle, as described in Section 2.4.

Table 2-4 provides values for the parameters shown in Figure 2-8.

2.4 INSTRUCTION EXECUTION

Section 1.1 describes the nature of an instruction, and how an instruction is executed using logic dispersed on devices of the F8 microcomputer chip set.

Specific timing and signal information is given in this section.

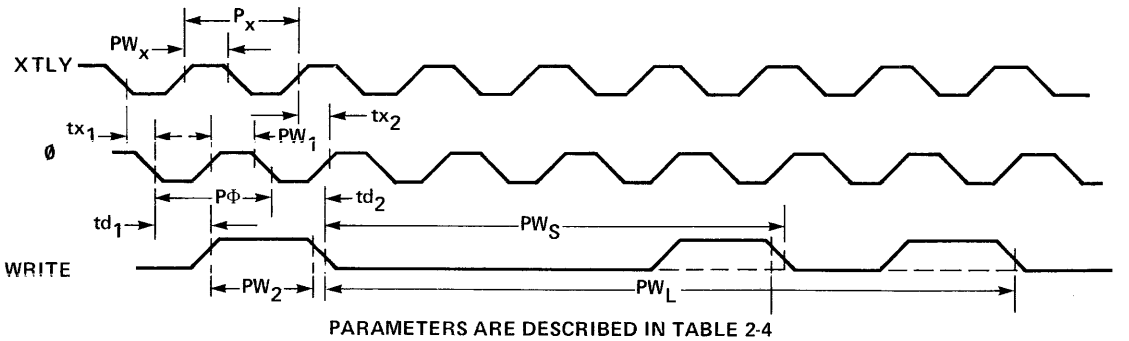


Figure 2-8. Timing Signal Specifications

3850 CPU logic controls instruction execution via the Φ and WRITE timing controls, plus the five ROMC control lines. Devices external to the 3850 CPU must respond directly to these signals.

2.4.1 The Instruction Cycle

All instructions are executed in cycles, which are timed by the trailing edge of WRITE.

There are two types of instruction cycle, the short cycle which is four Φ periods long, and the long cycle which is six Φ periods long. The long cycle is sometimes referred to as 1.5 cycles. Figure 2-8 (in Section 2.3.4) illustrates the short cycle (PW_S) and the long cycle (PW_l). Observe that WRITE high appears only at the end of an instruction cycle.

The simplest instructions of the F8 instruction set execute in one short cycle. The most complex instruction (PI) requires two short cycles plus three long cycles.

2.4.2 The ROMC Signals

CPU logic uses the five ROMC signals to identify operations which other devices must perform during any instruction cycle. The 32 possible ROMC states are described in Table 2-5. The state of the ROMC signals and the operation they identify last through one instruction cycle.

The general distribution of logic among devices of the F8 family, and general data movements associated with instruction execution are given in "An Introduction to Programming the Fairchild F8 Microcomputer." Using that book for background reference, the paragraphs below draw attention to certain F8 microprocessor system features which must be understood in order to appreciate how the ROMC states described in Table 2-5 are used.

Memory addressing logic is located on the 3851 PSU, the 3852 DMI, and the 3853 SMI devices; each of these devices contain registers to address programs (PC0 and PC1) or data (DC0 or DC1). The 3851 PSU does not have a DC1 register.

Unlike other microprocessors, the 3850 CPU does not output addresses at the start of memory access sequences; a simple command to access the memory location addressed by PC0 or DC0 is sufficient, since the device receiving the memory access command contains PC0 and DC0 registers. (PC1 and DC1 are buffer registers for PC0 and DC0.)

Moving memory addressing logic from the CPU to memory (and memory interface) devices simplifies CPU logic; however, it creates the potential for devices to compete when responding to memory access commands.

There will be as many PC0 and DC0 registers in a microcomputer system as there are PSU, DMI and SMI devices; which is to respond to a memory read or write command? This ambiguity is resolved by insuring that all PC0 registers, and all DC0 registers contain the same information, at all times. Every PSU, DMI and SMI device, on the other hand, has a unique address space, that is, a unique block of memory addresses within which it, alone, responds to memory access commands. For example, a 3851 PSU may have an address space of H'0000' through H'03FF'; a 3852 DMI may have an address space of H'0400' through H'07FF'. If a microcomputer system has these two memory devices, and no others,

Table 2-3. A Summary of 3850 CPU Signal DC Characteristics

SIGNAL	SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
Φ, WRITE	V_{OH}	Output High Voltage	4.4	V_{DD}	Volts	I_{OH} = -50 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1.6 mA
	V_{OH}	Output High Voltage	2.9		Volts	I_{OH} = -100 μA
XTLY	V_{IH}	Input High Voltage	4.5	V_{GG}	Volts	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_{IH}	Input High Current	5	50	μA	$V_{IN} = V_{DD}$
	I_{IL}	Input Low Current	-10	-120	μA	$V_{IN} = V_{SS}$
ROMC0 ⋮ ROMC4	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = -100 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1.6 mA
DB0 ⋮ DB7	V_{IH}	Input High Voltage	2.9	V_{DD}	Volts	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = -100 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1.6 mA
	I_{IH}	Input High Current		3	μA	V_{IN} = 7V 3-State mode
	I_{IL}	Input Low Current		-3	μA	$V_{IN} = V_{SS}$, 3-State mode
I/O 0 ⋮ I/O 17	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = -30 μA
	V_{OH}	Output High Voltage	2.9	V_{DD}	Volts	I_{OH} = -150 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1.6 mA
	V_{IH}	Input High Voltage (1)	2.9	V_{DD}	Volts	Internal pull-up to V_{DD}
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_{IL}	Input Low Current		-1.6	mA	V_{IN} = 0.4V (2)
$\overline{\text{EXT RES}}$	V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	Internal pull-up to V_{DD}
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_{IL}	Input Low Current	-0.1	-1.0	mA	$V_{IN} = V_{SS}$
$\overline{\text{INT REQ}}$	V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	Internal pull-up to V_{DD}
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_{IL}	Input Low Current	-0.1	-1.0	mA	$V_{IN} = V_{SS}$
$\overline{\text{ICB}}$	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	IOH = -10 μA
	V_{OH}	Output High Voltage	2.9	V_{DD}	Volts	I_{OH} = -100 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 100 μA

(1) Hysteresis input circuit provides additional 0.3V noise immunity while internal pull-up provides TTL compatability.

(2) Measured while F8 port is outputting a high level.

Note:

Positive current is defined as conventional current flowing into the pin referenced.

(3) Guaranteed but not tested.

Table 2-4. A Summary of 3850 CPU Signal AC Characteristics

AC Characteristics: V_{SS} = 0V, V_{DD} = +5V ± 5%, V_{GG} = +12V ± 5%, T_A = 0°C to +70°C

Symbols in this table are used by all figures in Section 2.

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
P_x*	External Input Period	0.5		10	μS	
PW_x*	External Pulse Width	200		P_x-200	nS	t_r, $t_f \leqslant$ 30 nS
tx_1	Ext. to Φ - to - Delay			250	nS	CL = 100 pf
tx_2	Ext. to Φ + to + Delay			250	nS	CL = 100 pf
PΦ	Φ Period	0.5		10	μS	
PW_1	Φ Pulse Width	180		PΦ-180	nS	t_r, t_f = 50 nS; C_L = 100 pf
td_1	Φ to WRITE + Delay		150	250	nS	C_L = 100 pf
td_2	Φ to WRITE - Delay		150	250	nS	C_L = 100 pf
PW_2	WRITE Pulse Width	PΦ-100		PΦ	nS	t_r, t_f = 50 nS typ; C_L = 100 pf
PW_S	WRITE Period; Short		4PΦ			
PW_L	WRITE Period; Long		6PΦ			
td_3	WRITE to ROMC Delay	80	300	550	nS	C_L = 100 pf
td_4*	WRITE to $\overline{ICB}$ Delay			350	nS	C_L = 50 pf
td_5	WRITE to $\overline{INT\ REQ}$ Delay			430 (2)	nS	C_L = 100 pf
t_{sx}*	$\overline{EXT\ RES}$ set-up time	1.0			μS	C_L = 20 pf
t_{su}*	I/O set-up time	300			nS	
t_h*	I/O hold time	50			nS	
t_o*	I/O Output Delay			2.5	μS	C_L = 50 pf
tdb_1*	WRITE to Data Bus Stable		0.6	1.3	μS	C_L = 100 pf
tdb_2	WRITE to Data Bus Stable	2PΦ		2PΦ+1.0	μS	C_L = 100 pf
tdb_3*	Data Bus Set-up	200			nS	
tdb_4*	Data Bus Set-up	500			nS	
tdb_5	Data Bus Set-up	500			nS	
tdb_6*	Data Bus Set-up	500			nS	

*The parameters which are starred in the table above represent those which are most frequently of importance when interfacing to an F8 system. These encompass I/O timing, external timing generation and possible external RAM timing. The remaining parameters are typically those that are only relevant between F8 chips and not normally of concern to the user.

(1) Input and output capacitance is 3 to 5 pf typical on all pins except V_{DD}, V_{GG}, and V_{SS}.

(2) If $\overline{INTREQ}$ is being supplied asynchronously, it can be pulled down at any time except during a fetch cycle that has been preceded by a non-priviledged instruction. In that case $\overline{INTREQ}$ must go down according to the requirements of td_5.

then the 3851 PSU will respond to memory access commands when the PC0 or DC0 registers (whichever are identified as the address source) contain a value between H'0000' and H'03FF'; the 3852 DMI will respond to addresses in the range H'0400' through H'07FF'. No device will respond to addresses beyond H'07FF', even though such addresses may exist in PC0 and/or DC0.

How PSU, DMI and SMI devices identify their address space is not germane to the current discussion, and is described in Sections 3.4, 4.4 and 5.4, respectively. The only important point to note, for the moment, is that each device compares its address space with the contents of PC0 and DC0, whichever is identified as the address source, and only responds to a memory access command if the contents of PC0 or DC0 is within the device's address space.

If all memory address registers, PC0, PC1, DC0 and DC1, are to contain the same information, then ROMC states that require any of these registers' contents to be modified must be acted upon by all devices containing any of these four registers.

If devices are not to compete when a ROMC state specifies that a memory access must be performed, then only a device whose address space includes the identified memory address must respond to the ROMC state.

As illustrated in Figure 2-9, the five ROMC signals which define the ROMC state are output early in the instruction cycle, and are maintained stable for the duration of the instruction cycle. In other words, only one ROMC state can be specified per instruction cycle. As this would imply, devices can only be called upon to perform one instruction execution related operation in one instruction cycle.

With reference to Table 2-5, each ROMC state is identified by individual signal line states (1 for high, 0 for low), and by a two-digit, hexadecimal code. The hexadecimal code is used to identify ROMC states in the rest of this manual. Also shown is the instruction cycle length (short or long) implied by each code, plus the way in which codes must be interpreted by the other F8 devices.

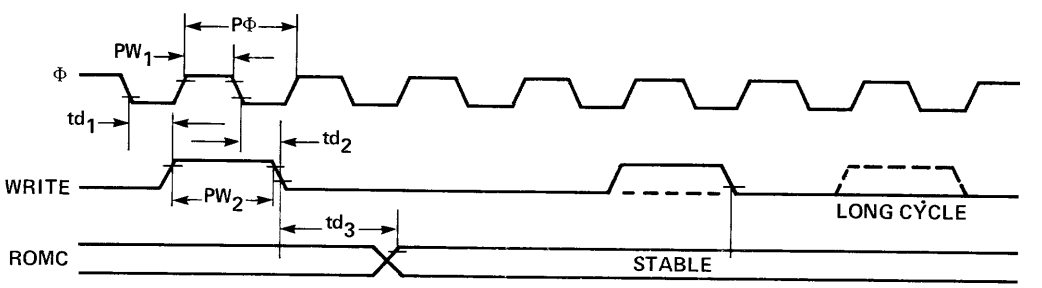
2.4.3 Instruction Execution Sequence

Every instruction's execution sequence ends with an instruction code being fetched from memory to identify the next instruction cycle. The instruction code is loaded into the CPU's instruction register, out of which it is decoded by the CPU's Control Unit Logic.

An instruction fetch is executed during the last instruction cycle of the previous instruction, as illustrated in Figure 2-10.

There are a group of F8 instructions that cause operations to occur entirely within the 3850 CPU. These instructions do not use the data bus, therefore can execute in one cycle. Since one cycle instructions do not use the data bus, no ROMC state needs to be generated for the one cycle instruction being executed; therefore, as illustrated in Figure 2-10, ROMC state 0 is specified, causing the next instruction's instruction fetch.

Multi-cycle instructions must end with a cycle that does not use the data bus; ROMC state 0 is specified at the beginning of this last instruction cycle, causing the next instruction to be fetched.



SYMBOLS ARE DEFINED BY TABLE 2-4

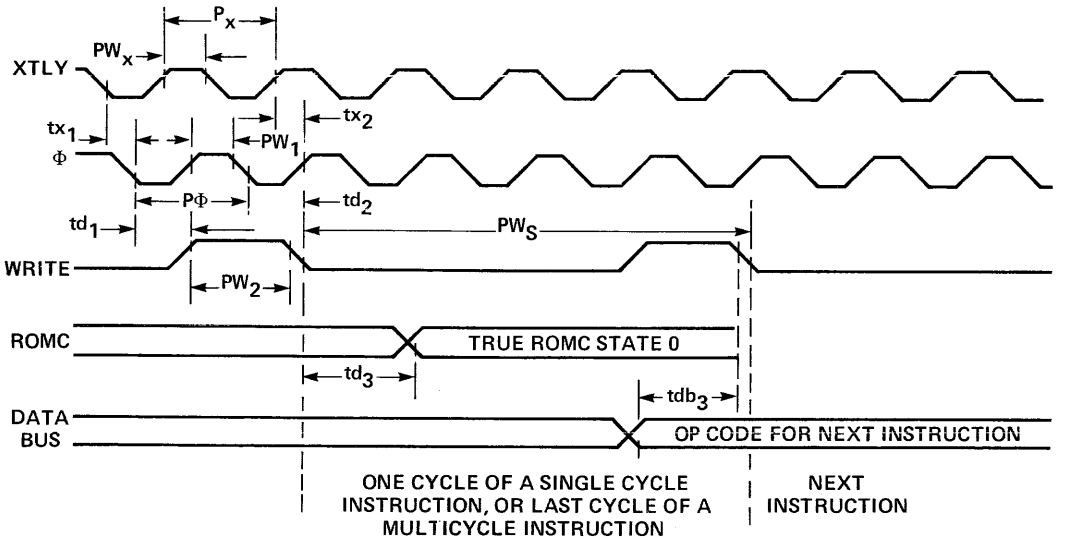
Figure 2-9. ROMC Signals Output by 3850 CPU

Table 2-5. ROMC Signals and What They Imply

ROMC 4 3 2 1 0	HEX	CYCLE LENGTH	FUNCTION
0 0 0 0 0	00	S,L	Instruction Fetch. The device whose address space includes the contents of the PC0 register must place on the data bus the op code addressed by PC0. Then all devices increment the contents of PC0.
0 0 0 0 1	01	L	The device whose address space includes the contents of the PC0 register must place on the data bus the contents of the memory location addressed by PC0. Then all devices add the 8-bit value on the data bus, as a signed binary number, to PC0.
0 0 0 1 0	02	L	The device whose DC0 addresses a memory word within the address space of that device must place on the data bus the contents of the memory location addressed by DC0. Then all devices increment DC0.
0 0 0 1 1	03	L,S	Similar to 00, except that it is used for Immediate Operand fetches (using PC0) instead of instruction fetches.
0 0 1 0 0	04	S	Copy the contents of PC1 into PC0.
0 0 1 0 1	05	L	Store the data bus contents into the memory location pointed to by DC0. Increment DC0.
0 0 1 1 0	06	L	Place the high order byte of DC0 on the data bus.
0 0 1 1 1	07	L	Place the high order byte of PC1 on the data bus.
0 1 0 0 0	08	L	All devices copy the contents of PC0 into PC1. The CPU outputs zero on the data bus in this ROMC state. Load the data bus into both halves of PC0 thus clearing the register.
0 1 0 0 1	09	L	The device whose address space includes the contents of the DC0 register must place the low order byte of DC0 onto the data bus.
0 1 0 1 0	0A	L	All devices add the 8-bit value on the data bus, treated as a signed binary number, to the Data Counter.
0 1 0 1 1	0B	L	The device whose address space includes the value in PC1 must place the low order byte of PC1 on the data bus.
0 1 1 0 0	0C	L	The device whose address space includes the contents of the PC0 register must place the contents of the memory word addressed by PC0 onto the data bus. Then all devices move the value which has just been placed on the data bus into the low order byre of PC0.
0 1 1 0 1	0D	S	All devices store in PC1 the current contents of PC0, incremented by 1. PC0 is unaltered.
0 1 1 1 0	0E	L	The device whose address space includes the contents of PC0 must place the contents of the word addressed by PC0 onto the data bus. The value on the data bus is then moved to the low order byte of DC0 by all devices.
0 1 1 1 1	0F	L	The interrupting device with highest priority must place the low order byte of the interrupt vector on the data bus. All devices must copy the contents of PC0 into PC1. All devices must move the contents of the data bus into the low order byte of PC0.
1 0 0 0 0	10	L	Inhibit any modification to the interrupt priority logic.
1 0 0 0 1	11	L	The device whose memory space includes the contents of PC0 must place the contents of the addressed memory word on the data bus. All devices must then move the contents of the data bus to the upper byte of DC0.

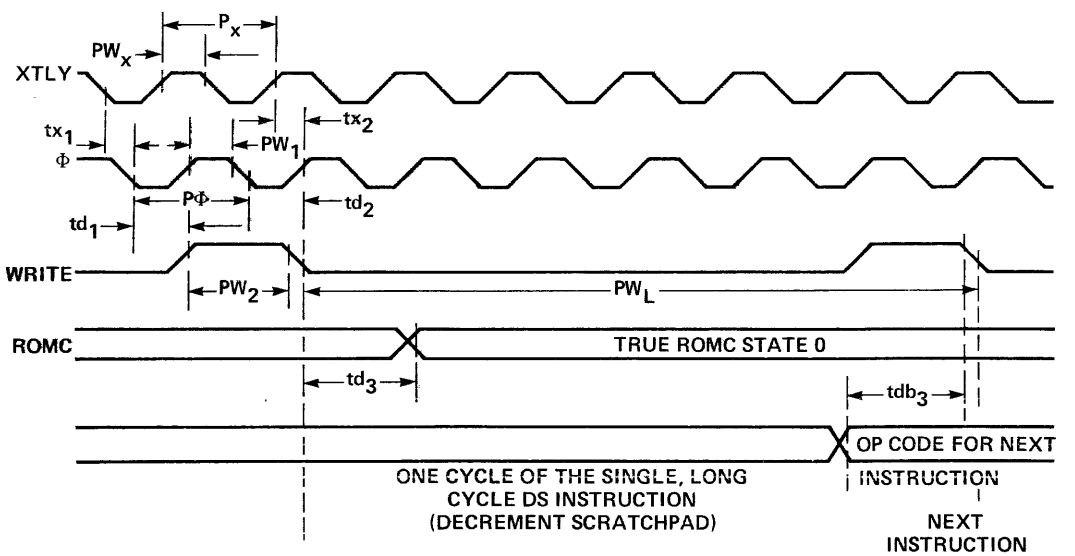
Table 2-5. ROMC Signals and What They Imply (Continued)

ROMC 4 3 2 1 0	HEX	CYCLE LENGTH	FUNCTION
1 0 0 1 0	12	L	All devices copy the contents of PC0 into PC1. All devices then move the contents of the data bus into the low order byte of PC0.
1 0 0 1 1	13	L	The interrupting device with highest priority must move the high order half of the interrupt vector onto the data bus. All devices must move the contents of the data bus into the high order byte of PC0. The interrupting device will reset its interrupt circuitry (so that it is no longer requesting CPU servicing and can respond to another interrupt).
1 0 1 0 0	14	L	All devices move the contents of the data bus into the high order byte of PC0.
1 0 1 0 1	15	L	All devices move the contents of the data bus into the high order byte of PC1.
1 0 1 1 0	16	L	All devices move the contents of the data bus into the high order byte of DC0.
1 0 1 1 1	17	L	All devices move the contents of the data bus into the low order byte of PC0.
1 1 0 0 0	18	L	All devices move the contents of the data bus into the low order byte of PC1.
1 1 0 0 1	19	L	All devices move the contents of the data bus into the low order byte of DC0.
1 1 0 1 0	1A	L	During the prior cycle an I/O port timer or interrupt control register was addressed, the device containing the addressed port must move the current contents of the data bus into the addressed port.
1 1 0 1 1	1B	L	During the prior cycle the data bus specified the address of an I/O port. The device containing the addressed I/O port must place the contents of the I/O port on the data bus. (Note that the contents of timer and interrupt control registers cannot be read back onto the data bus.)
1 1 1 0 0	1C	L or S	None.
1 1 1 0 1	1D	S	Devices with DC0 and DC1 registers must switch registers. Devices without a DC1 register perform no operation.
1 1 1 1 0	1E	L	The device whose address space includes the contents of PC0 must place the low order byte of PC0 onto the data bus.
1 1 1 1 1	1F	L	The device whose address space includes the contents of PC0 must place the high order byte of PC0 onto the data bus.



Symbols are defined in Table 2-4

Figure 2-10A. A Short Cycle Instruction Fetch



Symbols are defined in Table 2 4

Figure 2-10B. A Long Cycle Instruction Fetch (During DS Only)

Following an instruction fetch, CPU logic decodes the fetched instruction code and executes the specified instruction. There are five types of instruction cycle that can follow:

1. Operations may all be internal to the CPU. This will be the last, or the only cycle for an instruction, and will specify ROMC state 0, as illustrated in Figures 2-10A and 2-10B.
2. Data may be transferred between the 3850 CPU and memory devices, as described in Section 2.4.4.
3. Data may be transferred from one memory device to all memory devices. The CPU is neither the transmitter nor the receiver of data in this transfer. This instruction cycle is described in Section 2.4.5.
4. Data may be transferred to or from an I/O port, as described in Section 2.4.6.
5. An interrupt may be acknowledged, as described in Section 2.4.6.

Every F8 instruction is executed as one, or a sequence of standard instruction cycles. Timing for the standard instruction cycles is given in Figures 2-10, 2-11, 2-13 and 2-14.

Table 2-7 lists the instruction cycles, plus the ROMC state associated with each cycle, for every F8 instruction.

2.4.4 Referencing Memory

Memory may be referenced during an instruction cycle either to transfer the data from the CPU to a memory word, or to transfer data from a memory word to the CPU. A memory reference occurs as shown in Figure 2-11.

If data is being output by the CPU, then the delay before data output is stable will be tdb_1 when data comes from the accumulator; the instruction cycle will be long. The delay before data output is stable will be tdb_2 when data comes from the scratchpad; the instruction cycle in this case will also be long.

If data is being input to the CPU, then the delay before incoming data must be stable depends on the destination of the data, as described in Figure 2-11.

The type of data transfer will be identified by the ROMC state which is output at the beginning of the instruction cycle.

The instruction fetch may also be viewed as a memory reference operation where the destination is the Instruction register. Timing for this case is illustrated in Figure 2-10.

2.4.5 Memory-to-Memory Data Transfers

In response to appropriate ROMC states, data may be transferred from one memory device to all memory devices during one instruction cycle. For example, data may be transferred from a memory byte within (or controlled by) one memory device, to one byte of an address register (PC0 or DC0) within all memory devices.

ROMC states C, E and 11 specify operations of this type, and Figure 2-11 provides timing for the data transfer. In Figure 2-11, tdb_2 is the delay until data from memory or a memory address register is stable on the data bus.

2.4.6 Input/Output Interfacing

Programmed input/output (I/O) in the F8 microcomputer system is influenced by the design of the I/O port pins.

As illustrated in Figure 2-12, each I/O port pin is a "wire-AND" structure between an internal latch and an external signal, if any. The latch is always loaded directly from the accumulator.

Each F8 I/O pin may be set high or low, under program control. If a 1 (high) is presented at the latch, then gate (b) will turn on and gate (a) will turn off, so that P will be at V_{SS} (low). If a 0 (low) is presented at the latch, then gate (a) will turn on and gate (b) will turn off, so that P will be at V_{DD} (high).

When outputting data through an I/O port, the pin can be connected directly to a TTL gate input ("TTL Device Input" in Figure 2-12).

Data is input to the pin from a "TTL Device Output" in Figure 2-12.

In normal operation, high or low levels at P drive the external TTL device input transistor (d). If a low level is set at P, transistor (d) conducts current through the path J, I, P, and FET (b). This is transferred as a low level to the rest of the circuits in the TTL device, and results in a high or low

Table 2-6. Symbology used in Table 2-7

SYMBOL	INTERPRETATION
A	The Accumulator.
($\overline{A}$)	The complement of accumlator contents.
a	A single hexadecimal digit being interpreted as data.
aa	Two hexadecimal digits being interpreted as a single byte of data, or as the high order byte of 16 bits of data.
bb	Two hexadecimal digits being interpreted as the low order byte of 16 bits of data.
Binary	Binary arithmetic specified.
C	The carry status flag.
DB	F8 System Data Bus.
DC0	The primary data counter register.
DC0L	The low order byte of the primary data counter register.
DC0U	The high order byte of the primary data counter register.
DC1	The secondary data counter register.
Decimal	Decimal arithmetic specified.
e	A single octal digit being interpreted as data.
H	Scratchpad bytes 10 and 11.
ii	Two hexadecimal digits being interpreted as the high order byte of a 16-bit address, or as a simple byte address displacement.
ISAR	The six-bit scratchpad address register.
ISARL	The low order three bits of ISAR.
ISARU	The high order three bits of ISAR.
J	Scratchpad byte 9.
jj	Two hexadecimal digits being interpreted as the low order byte of a 16-bit address.
K	Scratchpad bytes 12 and 13.
KL	Scratchpad byte 13.
KU	Scratchpad byte 12.
O	The overflow status flag.
P	A single hexadecimal digit being interpreted as an I/O port address (0-15).
PP	Two hexadecimal digits being interpreted as an I/O port address (0-255).
PC0	The program counter register.
PC0L	The low order byte of the program counter register.
PC0U	The high order byte of the program counter register.
PC1	The static register.
PC1L	The low order byte of the stack register.
PC1U	The high order byte of the stack register.
Q	Scratchpad bytes 14 and 15.
QL	Scratchpad byte 15.
QU	Scratchpad byte 14.
r	Single hexadecimal digit interpreted as scratchpad address: r = 0 through B for locations 0 through B in scratchpad r = C for ISAR as address source with no change after access r = D for ISAR as address source with ISARL = ISARL + 1 after access.

Table 2-6. Symbology Used In Table 2-7 (Continued).

SYMBOL	INTERPRETATION
	r = E for ISAR as address source with ISARL = ISARL -1 after access r = F is not allowed
S	The sign status flag.
t	A single hexadecimal digit identifying a status condition which will be tested by a "Branch on Condition" instruction.
W	The status register.
Z	The zero status flag.
∧	The logical OR of 8-bit quantities on each side of this symbol is specified.
⊕	The logical Exclusive-OR of 8-bit quantities on each side of this symbol is specified.
←	The value to the right of this symbol is to be loaded into the location specified on the left of this symbol.
( )	The contents of the location within the brackets is specified.
(( ))	The contents of the memory word addressed by the contents of the location within the double brackets is specified.
+	The binary address of 8-bit quantities on each side of this symbol is specified.

Table 2-7. Instructions' Execution and Timing

OP CODE	OPERAND(S)	OBJECT CODE	CYCLE	ROMC STATE	TIMING	STATUS FLAGS				INTERRUPT	FUNCTION
						O	Z	C	S		
LR	A, KU	00	S	0	3S	–	–	–	–		A ← (r12)
LR	A, KL	01	S	0	3S	–	–	–	–		A ← (r13)
LR	A, QU	02	S	0	3S	–	–	–	–		A ← (r14)
LR	A, QL	03	S	0	3S	–	–	–	–		A ← (r15)
LR	KU, A	04	S	0	3S	–	–	–	–		r12 ← (A)
LR	KL, A	05	S	0	3S	–	–	–	–		r13 ← (A)
LR	QU, A	06	S	0	3S	–	–	–	–		r14 ← (A)
LR	QL, A	07	S	0	3S	–	–	–	–		r15 ← (A)
LR	K, P	08	L	7	5	–	–	–	–		r12 ← (PC1U)
			L	B	5	–	–	–	–		r13 ← (PC1L)
			S	0	3S	–	–	–	–		
LR	P, K	09	L	15	2	–	–	–	–		PC1U ← (r12)
			L	18	2	–	–	–	–		PC1L ← (r13)
			S	0	3S	–	–	–	–		
LR	A, IS	0A	S	0	3S	–	–	–	–		A ← (ISAR)
LR	IS, A	0B	S	0	3S	–	–	–	–		ISAR ← (A)
PK		0C	L	12	2	–	–	–	–		PC1 ← (PC0); PC0L ← (r13)
			L	14	2	–	–	–	–		PC0U ← (r12)
			S	0	3S	–	–	–	–	x	
LR	P0, Q	0D	L	17	2	–	–	–	–		PC0L ← (r15)
			L	14	2	–	–	–	–		PC0U ← (r14)
			S	0	3S	–	–	–	–		

Table 2-7. Instructions' Execution and Timing (Continued)

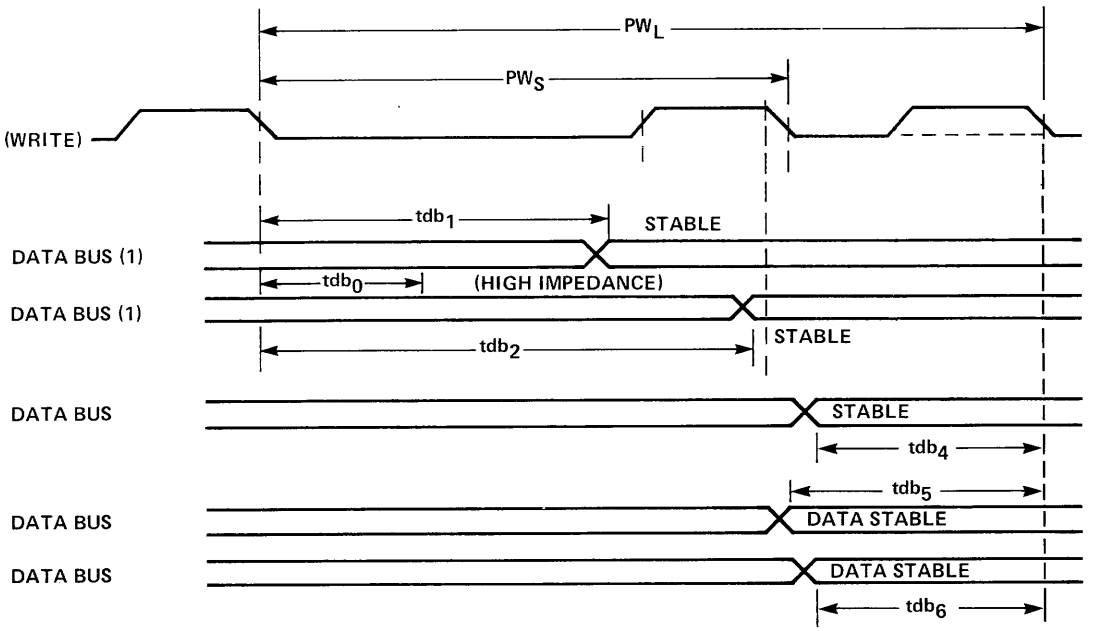
OP CODE	OPERAND(S)	OBJECT CODE	CYCLE	ROMC STATE	TIMING	STATUS FLAGS				INTERRUPT	FUNCTION
						O	Z	C	S		
LR	Q, DC	0E	L	6	5	–	–	–	–		r14 ← (DC0U)
			L	9	5	–	–	–	–		r15 ← (DC0L)
			S	0	3S	–	–	–	–		
LR	DC, Q	0F	L	16	2	–	–	–	–		DC0U ← (R14)
			L	19	2	–	–	–	–		DC0L ← (R15)
			S	0	3S	–	–	–	–		
LR	DC, H	10	L	16	2	–	–	–	–		DC0U ← (R10)
			L	19	2	–	–	–	–		DC0L ← (R11)
			S	0	3S	–	–	–	–		
LR	H, DC	11	L	6	5	–	–	–	–		r10 ← (DC0U)
			L	9	5	–	–	–	–		r11 ← (DC0L)
			S	0	3S	–	–	–	–		
SR	1	12	S	0	3S	0	1/0	0	1		Shift (A) right one bit position (zero fill)
SL	1	13	S	0	3S	0	1/0	0	1/0		Shift (A) left one bit position (zero fill)
SR	4	14	S	0	3S	0	1/0	0	1		Shift (A) right four bit positions (zero fill)
SL	4	15	S	0	3S	0	1/0	0	1/0		Shift (A) left four bit positions (zero fill)
LM		16	L	2	6	–	–	–	–		A ← ((DC0))
			S	0	3S	–	–	–	–		
ST		17	L	5	1	–	–	–	–		(DC) ← (A)
			S	0	3S	–	–	–	–		
COM		18	S	0	3S	0	1/0	0	1/0		A ← (A) ⊕ H'FF' Complement accumulator
LNK		19	S	0	3S	1/0	1/0	1/0	1/0		A ← (A) + (C)
DI		1A	S	1C	0	–	–	–	–	y	Clear ICB.
			S	0	3S	–	–	–	–		
EI		1B	S	1C	0	–	–	–	–		Set ICB
			S	0	3S	–	–	–	–	x	
POP		1C	S	4	0	–	–	–	–		PC0 ← (PC1)
			S	0	3S	–	–	–	–	x	
LR	W, J	1D	S	1C	0	1/0	1/0	1/0	1/0		W ← (r9)
			S	0	3S	–	–	–	–	x	
LR	J, W	1E	S	0	3S	–	–	–	–		r9 ← (W)
INC		1F	S	0	3S	1/0	1/0	1/0	1/0		A ← (A) + 1
LI	aa	20	L	3	6	–	–	–	–		A ← H'aa'
	└→	aa	S	0	3S	–	–	–	–		
NI	aa	21	L	3	4	0	1/0	0	1/0		A ← (A) ∧ H'aa'
	└→	aa	S	0	3S	–	–	–	–		
OI	aa	22	L	3	4	0	1/0	0	1/0		A ← (A) ∨ H'aa'
	└→	aa	S	0	3S	–	–	–	–		
XI	aa	23	L	3	4	0	1/0	0	1/0		A ← (A) ⊕ H'aa'
	└→	aa	S	0	3S	–	–	–	–		
AI	aa	24	L	3	4	1/0	1/0	1/0	1/0		A ← (A) + H'aa'
	└→	aa	S	0	3S	–	–	–	–		

Table 2-7. Instructions' Execution and Timing (Continued)

OP CODE	OPERAND(S)	OBJECT CODE	CYCLE	ROMC STATE	TIMING	STATUS FLAGS				INTERRUPT	FUNCTION
						O	Z	C	S		
CI	aa	25	L	3	4	—	—	—	—		Perform H'aa' + ($\overline{A}$)
	→ aa		S	0	3S	1/0	1/0	1/0	1/0		+ 1. Do not save result, but modify status flags to reflect result.
IN	PP	26	L	3	2	—	—	—	—		DB ← PP
	→ PP		L	1B	6	0	1/0	0	1/0		A ← (I/O Port PP)
			S	0	3S	—	—	—	—		
OUT	PP	27	L	3	2	—	—	—	—		DB ← PP
	→ pp		L	1A	1	—	—	—	—		I/O Port PP ← (A)
			S	0	3S	—	—	—	—	x	
PI	iijj	28	L	3	6	—	—	—	—		A ← H'ii'
	→ ii		S	D	0	—	—	—	—		PC1 ← (PC0) + 1
	→ jj		L	C	2	—	—	—	—		PC0L ← H'jj'
			L	14	1	—	—	—	—		PC0U ← (A)
			S	0	3S	—	—	—	—	x	
JMP	iijj	29	L	3	6	—	—	—	—		A ← H'ii'
	→ ii		L	C	2	—	—	—	—		PC0L ← H'jj'
	→ jj		L	14	1	—	—	—	—		PC0U ← (A)
			S	0	3S	—	—	—	—	x	
DCI	iijj	2A	L	11	2	—	—	—	—		DC0U ← ii
	→ ii		S	3	0	—	—	—	—		(increment PC0)
	→ jj		L	E	2	—	—	—	—		DC0L ← jj
			S	3	0	—	—	—	—		(increment PC0)
			S	0	3S	—	—	—	—		
NOP		2B	S	0	0	—	—	—	—		
XDC		2C	S	1D	0	—	—	—	—		DC0 ⇄ DC1
			S	0	0	—	—	—	—		
DS	r	3r	L	0	3L	1/0	1/0	1/0	1/0		r ← (r) + H'FF' Decrement scratchpad byte
LR	A, r	4r	S	0	3S	—	—	—	—		A ← (r)
LR	r, A	5r	S	0	3S	—	—	—	—		r ← (A)
LISU	e	6e	S	0	3S	—	—	—	—		ISARU ← 0'e'
LISL	e	68 + e	S	0	3S	—	—	—	—		ISARL ← 0'e'
LIS	a	7a	S	0	3S	—	—	—	—		A ← H'0a'
BT	e, ii	8e	S	1C	0	—	—	—	—		Test e ∧ W. register
	→ ii		S	3	0	—	—	—	—		Res = 0 so PC0 = (PC0) + 2
			S	0	3S	—	—	—	—		
			S	1C	0	—	—	—	—		Test e ∧ W. register
			L	1	2	—	—	—	—		Res ≠ 0 so PC0 = (PC0) + H'ii' + 1
			S	0	3S	—	—	—	—		
AM		88	L	2	4	1/0	1/0	1/0	1/0		A ← (A) + ((DC0)) Binary, DC0 ← (DC) + 1
			S	0	3S						
AMD		89	L	2	4	1/0	1/0	1/0	1/0		A ← (A) + ((DC0)) Decimal; DC0 ← (DC0) + 1
			S	0	3S						

Table 2-7. Instructions' Execution and Timing (Continued)

OP CODE	OPERAND(S)	OBJECT CODE	CYCLE	ROMC STATE	TIMING	STATUS FLAGS				INTERRUPT	FUNCTION
						O	Z	C	S		
NM		8A	L	2	4	0	1/0	0	1/0		A ← (A) ∧ ((DC0));
			S	0	3S						DC0 ← (DC0) + 1
OM		8B	L	2	4	0	1/0	0	1/0		A ← (A) ∨ ((DC0)),
			S	0	3S						DC0 ← (DC0) + 1
XM		8C	L	2	4	0	1/0	0	1/0		A ← (A) ⊕ ((DC0));
			S	0	3S						DC0 ← (DC0) +1
CM		8D	L	2	4	1/0	1/0	1/0	1/0		Set status flags on basis
			S	0	3S						of ((DC)) + ($\overline{A}$) + 1;
											DC0 ← (DC0) + 1
ADC		8E	L	A	1	–	–	–	–		DC ← (DC) + (A)
			S	0	3S	–	–	–	–		
BR7	ii	8F	S	3	0	–	–	–	–		PC0 ← (PC0) + 2
	↳	ii	S	0	3S	–	–	–	–		because (ISARL) = 7
			L	1	2	–	–	–	–		PC0 ← (PC0) + H'ii' + 1
			S	0	3S	–	–	–	–		because (ISARL) ≠ 7
BF	t, ii	9t	S	1C	0	–	–	–	–		Test t ∧ W. register
	↳	ii	L	1	2	–	–	–	–		Res = 0 so PC0 = (PC0)
			S	0	3S	–	–	–	–		+ H'ii' + 1
			S	1C	0	–	–	–	–		Test t ∧ W. register
			S	3	0	–	–	–	–		Res ≠ 0 so PC0 = (PC0)
			S	0	3S	–	–	–	–		+ 2
INS	0 or 1	A0, A1	S	1C	0	0	1/0	0	1/0		A ← (I/O Port 0 or 1)
			S	0	3S	–	–	–	–		
INS	4	A4	L	1C	0	0	1/0	0	1/0		DB ← Port address (4
	through	through	L	1B	6	–	–	–	–		through 15)
	15	AF	S	0	3S	–	–	–	–		A ← (Port4 through 15)
OUTS	0 or 1	B0, B1	S	1C	0	–	–	–	–		I/O Port 0 or 1 ← (A)
			S	0	3S	–	–	–	–		
OUTS	4	B4	L	1C	0	–	–	–	–		DB ← Port address (4
	through	through	L	1A	1	–	–	–	–		through 15)
	15	BF	S	0	3S	–	–	–	–	x	Port (4 through 15) ←
											(A)
AS	r	Cr	S	0	3S	1/0	1/0	1/0	1/0		A ← (A) + (r) Binary
ASD	r	Dr	S	1C	0	1/0	1/0	1/0	1/0		A ← (A) + (r) Decimal
			S	0	3S	–	–	–	–		
XS	r	Er	S	0	3S	0	1/0	0	1/0		A ← (A) ⊕ (r)
NS	r	Fr	S	0	3S	0	1/0	0	1/0		A ← (A) ∧ (r)
INTRPT		xx	L	1C	0	–	–	–	–		IDLE
			L	0F	2	–	–	–	–		PC0L ← Int. address
											(lower byte); PC1 ← PC0
			L	13	2	–	–	–	–	y	PC0U ← Int. address
											(upper byte)
			S	0	3S	–	–	–	–	x	
RESET		xx	S	1C	0	–	–	–	–		IDLE
			L	8	1	–	–	–	–	y	PC0 ← 0, PC1 ← PC0
			S	0	3S	–	–	–	–	x	



1. Timing for CPU outputting data onto the data bus.

   Delay tdb$_1$ is the delay when data is coming from the accumulator.

   Delay tdb$_2$ is the delay when data is coming from the scratchpad (or from a memory device).

   Delay tdb$_0$ is the delay for the CPU to stop driving the data bus.

2. There are four possible cases when inputting data to the CPU, via the data bus lines: they depend on the data path and the destination in the CPU, as follows:

   tdb$_3$; Destination – IR (instruction Fetch) – See Figure 2-10 for details.
   tdb$_4$; Destination – Accumulator (with ALU operation – AM)
   tdb$_5$; Destination – Scratchpad (LR K,P etc.)
   tdb$_6$; Destination – Accumulator (no ALU operation – LM)

   In each case a stable data hold time of 50 nS from the WRITE refrence point is required.

Symbols are defined in Table 2-4

Figure 2-11. Memory Reference Timing

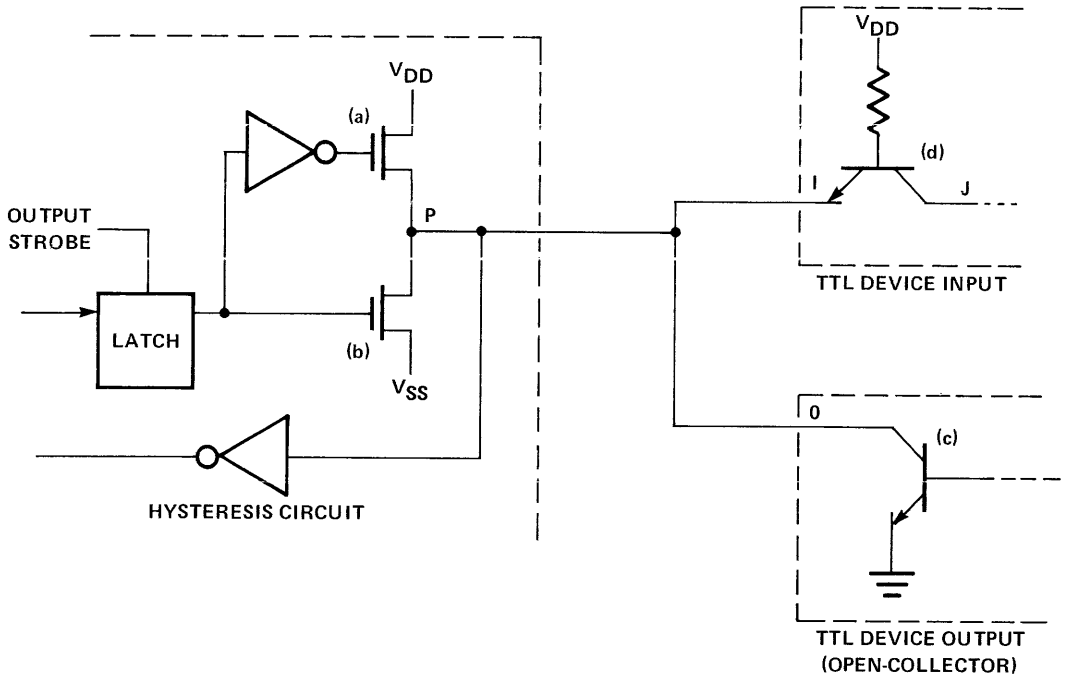


Figure 2-12. An F8 I/O Port Bit

level at the output of the device, depending on its characteristics. If the level at P is set high, transistor (d) does not conduct current, and a high level is transferred by (d).

When data is input to the I/O pin, high or low levels at 0 drive the hysteresis circuit in the port, and result in logic 1's or 0's being transferred to the accumulator.

Since the I/O pin and the TTL device output at 0 are wire-ANDed, it is possible for the state of one to affect the transfer of data out from the I/O pin or in from the TTL device output. For example, if the latch in the I/O port is set so that the pin is clamped low by (b), then the level at 0 cannot pull P high. Conversely, if P is clamped to a low level by (c), setting the latch for a high level has no effect.

It can be seen, then, that all I/O port bits should be set for a high level, before data input, to prevent incoming logic 0's from being "masked" by logic 1's present at the port from previous outputs.

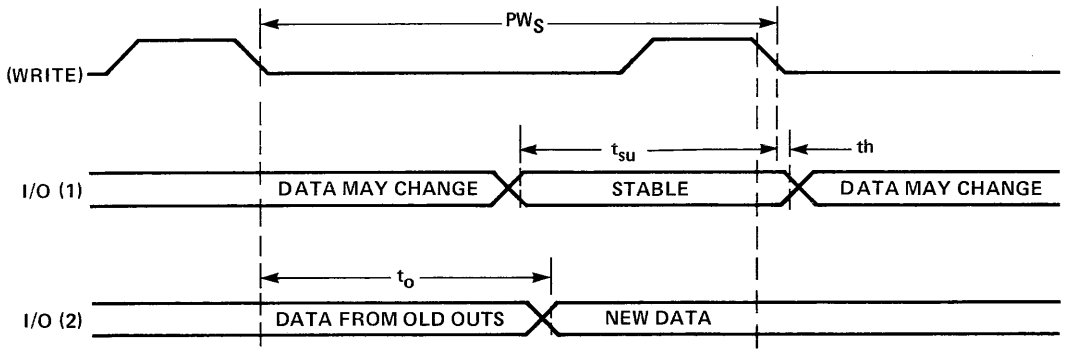
In some instances, the ability to mask bits of a port to logic 1 is useful. (Note that logic 1 becomes a 0V electrical level at the I/O pin; likewise logic 0 corresponds to a high electrical level.)

There are two types of programmed I/O operation that the F8 CPU may execute:

1. I/O via the two CPU ports (0 and 1),
2. I/O via ports on the other devices.

I/O operations that use the two CPU I/O ports execute in two instruction cycles. During the first cycle, the fetched instruction is decoded; the data bus is unused. During the first cycle, data is either sent from the accumulator to the I/O latch or enabled from the I/O pin to the accumulator depending on whether the instruction is an output or an input. At the falling edge of WRITE (marking the end of the first cycle and beginning of the second cycle) the data is strobed into either the latch (OUTS) or the accumulator (INS) respectively. The second cycle is then used by the CPU for its next instruction fetch. Figure 2-13 indicates I/O timing.

Observe that for the data input (INS) the set-up and hold times specified are with respect to the



(1) This represents the timing for data at the I/O pin during the execution of the INS instruction, i.e., the CPU is inputting.

(2) This represents the timing for data being output by the CPU at the I/O pin.

Symbols are defined in Table 2-4

Figure 2-13. Timing for Data Input or Output at I/O Port Pins

WRITE pulse occurring at the end of the first cycle in the two cycle instruction. For output data (OUTS) the delay is specified with respect to the falling edge of WRITE marking the beginning of the second cycle in the two cycle instruction.

I/O instructions that address I/O ports with an I/O port address greater than $0F_{16}$ occupy two bytes; the first byte specifies an IN or OUT instruction, while the second byte provides the I/O port address. Required timing at I/O port pins is given in the section of this manual that describes the device which contains the addressed I/O port.

2.4.7 Interrupts

This section describes timing associated with interrupts, as controlled by the CPU. The general concepts of interrupts has been described in Section 1.4 and use of interrupts, as it relates to other devices and device combinations, are described in other appropriate sections of this manual.

There are three CPU signals with interrupt processing; timing for all signals is illustrated in Figure 2-14.

An interrupt sequence is initiated by pulling either $\overline{\text{INT REQ}}$ or $\overline{\text{EXT RES}}$ low. In the case of $\overline{\text{INT REQ}}$, nothing will happen unless $\overline{\text{ICB}}$ is low. Also, nothing will happen until the next interruptable instruction comes to the end of execution. In the case of $\overline{\text{EXT RES}}$, execution of the interrupt routine will begin in the machine cycle immediately following that in which the signal goes low, providing the set-up time specified in Figure 2-14 has been met. $\overline{\text{EXT RES}}$ response logic ignores ICB.

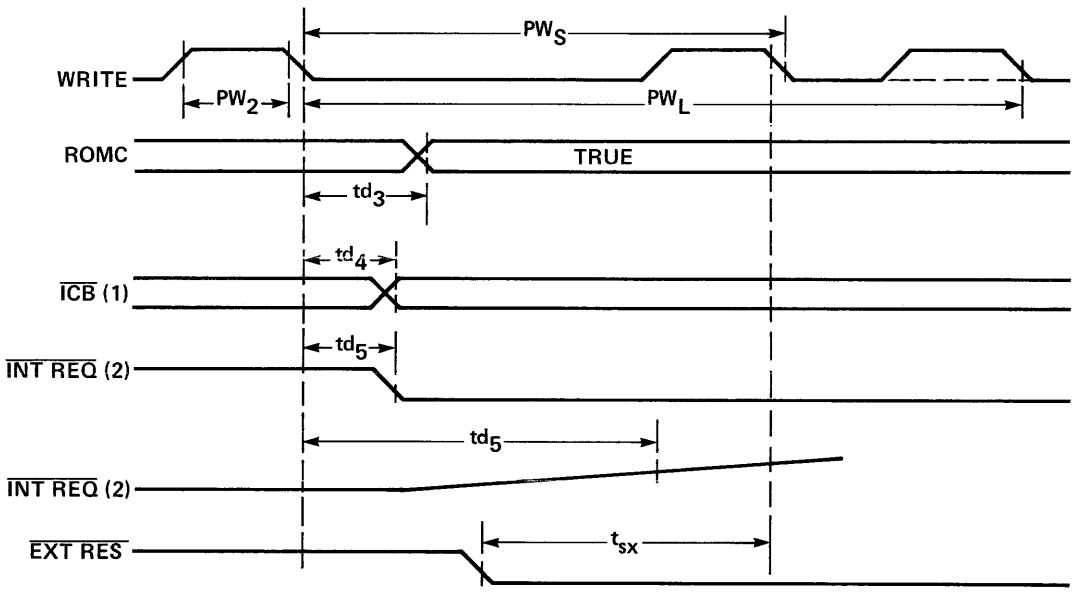
In response to $\overline{\text{INT REQ}}$ low, when the CPU acknowledges the interrupt, it forces $\overline{\text{ICB}}$ high, and initiates instruction cycles with ROMC states 1C, 0F, 13 and 00, in that order. This causes program execution to branch to the interrupting device's address vector.

In response to $\overline{\text{EXT RES}}$ low, when the CPU acknowledges the interrupt, it forces $\overline{\text{ICB}}$ high, then initiates instruction cycles with ROMC states 1C, 08 and 00, in that order. This causes program execution to branch to memory location 0.

The $\overline{\text{ICB}}$ signal is pulled low by the EI instruction, and is returned high by the DI instruction.

2.5 INSTRUCTION SET SUMMARY

The 3850 CPU instruction set is summarized in Table 2-7; this table and the accompanying text below does not attempt to teach a reader how to program the F8 microcomputer system, rather this section explains signals and timing associated with the execution of every instruction. The reader who wishes to learn how to write F8 programs should



(1) $\overline{ICB}$ will go from a 1 to a 0 following the execution of the EI instruction and will go from a 0 to 1 following either the execution of the DI instruction or the CPU's acknowledgement of an interrupt.

(2) This is an input to the CPU chip and is generated by a PSU or 3853 MI chip. The open drain outputs of these chips are all wire "ANDed" together on this line with the pull-up being located on the CPU chip. For a 0 to 1 transition the delay is measured to 2.0V.

Symbols are defined in Table 2-4

Figure 2-14. Interrupt Signals Timing

read "A Guide to Programming the F8 Microcomputer."

Referring to Table 2-7, columns should be interpreted as described next.

OP CODE

This is the instruction mnemonic which appears in the mnemonic field of an assembly language instruction, and identifies the instruction.

OPERAND(S)

If the instruction contains any information in the operand field of the assembly language source code, the information is shown in this column. Arrows identify the portion of object code which represent the operand field. Any portion of object code that does not represent the operand field must represent the mnemonic field. Table 2-6 explains symbology used in the operand field.

OBJECT CODE

This is the hexadecimal representation of the instruction's object code. The first byte of object code, or in some cases the first hexadecimal digit of object code, represents the Op Code. The operand is represented by the second and third bytes of object code, if present, or in some cases by the second hexadecimal digit of the first object code byte. Table 2-6 explains symbology used in the object code field.

CYCLE

This column identifies each instruction cycle for every instruction. Every cycle is listed on a separate

horizontal line, and is identified by the letter S for a short (4 clock period) cycle, or the letter L for a long (6 clock period) cycle. Thus the entry:

S

represents an instruction that executes in one short cycle. The entry:

S
L
S

represents an instruction that executes in three cycles; the first is a short cycle, the second is a long cycle, the third (and last) is a short cycle.

ROMC STATE

This is the state, as identified in Table 2-5, which is output by the 3850 CPU in the early stages of the instruction cycle.

TIMING

Timing for all instructions, except INS and OUTS accessing I/O ports 0 and 1, can be created out of Figures 2-10 and 2-11. For the exceptions, Figure 2-13 is required. The ROMC lines are always set after a delay of td_3, as shown in Figure 2-10. The only timing variations for each instruction cycle are data bus timing variations. Therefore data bus timing is defined using the delays tdb_1 through tdb_6. With the exception of tdb_3, these time delays are unambiguous, in that they are keyed to either the leading edge, or to the trailing edge of WRITE high, for either a long instruction cycle, or for a short instruction cycle, as illustrated in Figure 2-11. There are two cases for tdb_3, however, as illustrated in Figures 2-10A and 2-10B; these are identified in Table 2-7 as 3S for Figure 2-10A, and 3L for Figure 2-10B. tdb_1 through tdb_6 are otherwise identified by the numbers 1 through 6.

Cycles that do not use the data bus are identified by 0 in the timing column; Figure 2-9 illustrates timing in this case. In summary:

0 represents Figure 2-9
1 represents tdb_1 in Figure 2-11
2 represents tdb_2 in Figure 2-11
3S represents tdb_3 in Figure 2-10A
3L represents tdb_3 in Figure 2-10B
4 represents tdb_4 in Figure 2-11
5 represents tdb_5 in Figure 2-11
6 represents tdb_6 in Figure 2-11

STATUS FLAGS

Status flags are identified as follows:

0 – Overflow
Z – Zero
C – Carry
S – Sign

Within each column, symbology is used as follows:

– Status not effected
0 Status set to 0
1/0 Status set to either 1 or 0, depending on the results of the instruction's execution

INTERRUPT

An x in this column identifies an instruction that disallows interrupts at the end of the instruction's execution. A y identifies cycles in which the ICB bit is reset to 0 (cleared).

FUNCTION

The effect of each instruction cycle is described in this column using symbology given in Table 2-6.

Observe that instructions are described in Table 2-7 in order to ascending instruction (first byte) object code.

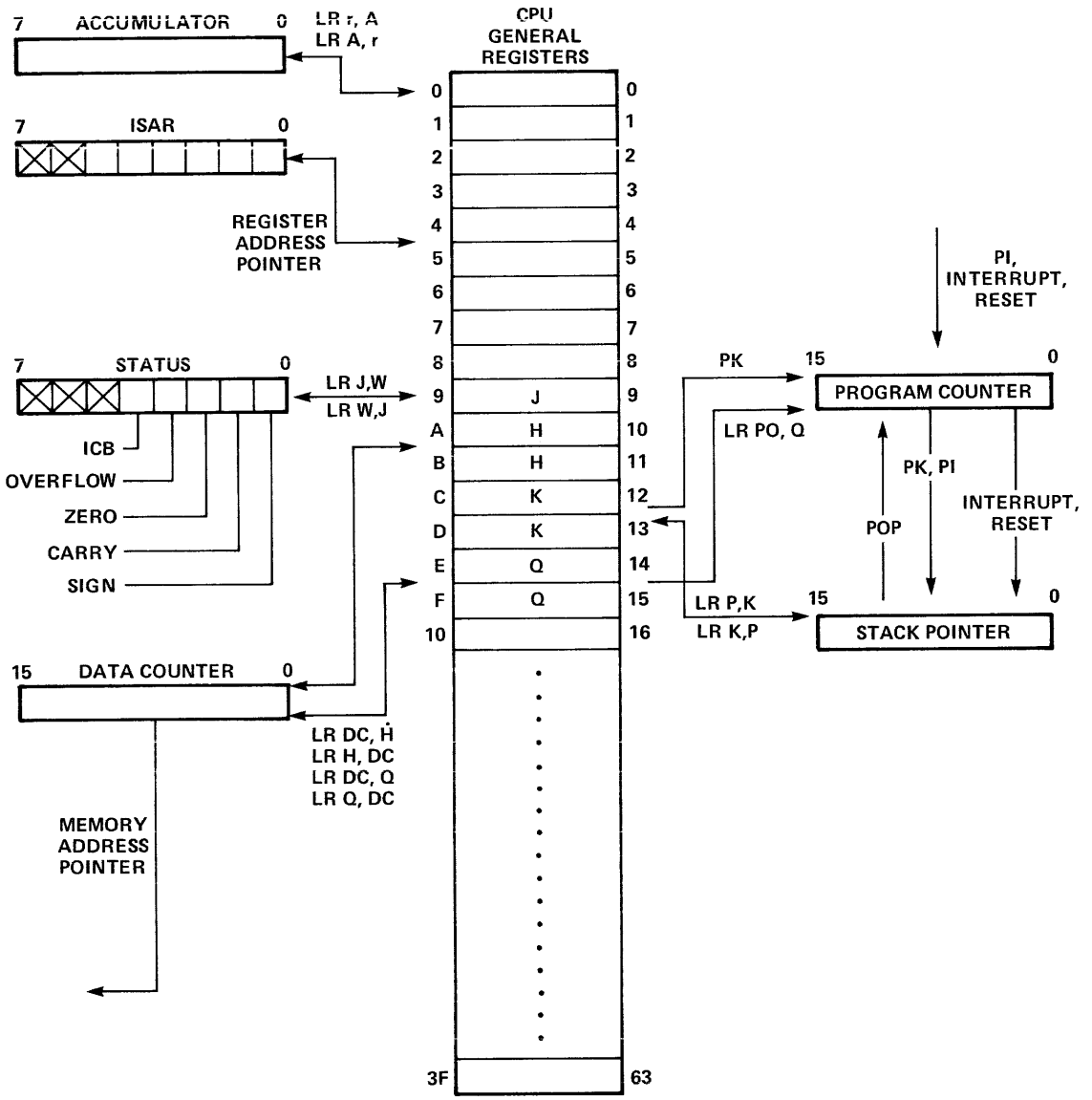


Figure 2-15. Instructions that Move Data between the Scratchpad and Registers

3.0 The 3851 Program Storage Unit (PSU)

THE 3851 PROGRAM STORAGE UNIT (PSU)

Those portions of a microcomputer system's logic which have been implemented on the 3851 PSU are described in Section 1. This section describes the 3851 PSU in detail.

The 3851 PSU is the principal program storage device for the F8 system, and to serve this purpose it provides 1024 bytes of ROM; programs and permanent data tables are specified as ROM masks, which are turned into custom 3851 ROM devices. Every 3851 PSU contains its own memory addressing logic.

In addition to providing 1024 bytes of ROM, plus memory addressing logic, the 3851 PSU has two 8-bit I/O ports, logic to handle an external interrupt, and a programmable timer.

+5V and +12V power supplies are required. The 3851 PSU is manufactured using N-channel, Isoplanar MOS technology, therefore power dissipation is very low, typically less than 275 mW.

The 3851 PSU is functionally illustrated in Figure 3-1, the figure shows logic functions, registers, data paths and device pins (with signal names); control signals within the PSU are not shown.

3.1 DEVICE ORGANIZATION

In order to understand 3851 PSU organization, it is important to recall that this device is not a simple Read-Only Memory unit.

Before the advent of LSI technology, any form of logic duplication within a computer system was a costly luxury that had to be avoided. Many microcomputer manufacturers have followed the separation of logic functions that became standard in the minicomputer industry, and have separated microcomputer logic, on LSI chips, along traditional minicomputer lines; with the F8 system, this is not the case. Every memory device within the F8 system contains its own memory addressing logic, along with associated address registers. The cost of duplicating memory addressing logic is trivial, at most costing a few pennies in any system; but as a result, a single 8-bit data bus provides all necessary communication between a 3851 PSU (or any other memory device), and a 3850 CPU. The 16 address lines via which the 3850 CPU would transmit memory addresses, if the 3851 PSU did not contain its own memory addressing logic, are used instead to implement two I/O ports, each of which is eight bits wide.

In order to support memory addressing logic, the 3851 PSU may be likened to a very simple CPU chip. Consider these two parallels:

1. The 3850 CPU has a powerful arithmetic and logic unit which performs a wide variety of data manipulations.

   The 3851 PSU has an elementary arithmetic unit which can increment and add 16-bit data units; for memory addressing logic, these two operations are sufficient.
2. The 3850 CPU has a control unit which decodes the 8-bit contents of the instruction register, and generates control signals within the CPU to enable data movement and ALU logic, as required; the control unit also generates five external control lines (ROMC0-ROMC4) which identify one of 32 possible operations which other devices within the F8 system are obliged to perform.

   The 3851 PSU also contains a primitive control unit; it decodes the five ROMC lines output by the CPU, as though they were a 5-bit instruction code. Just like the 3850 CPU, the 3851 PSU control unit generates internal signals to control data flow and arithmetic logic within the PSU. One control output, $\overline{DBDR}$, is generated to coincide with data being output by the PSU.

The fact that every memory device has its own memory addressing logic will not lead to logic contentions in a correctly implemented F8 system, as explained later in this section.

3.1.1 ROM Storage

Refer to Figure 3-1.

The 3851 PSU has 1024 bytes of read-only memory. This ROM array may contain object program code and/or tables of non-varying data. Every 3851 PSU is implemented using a custom mask which specifies

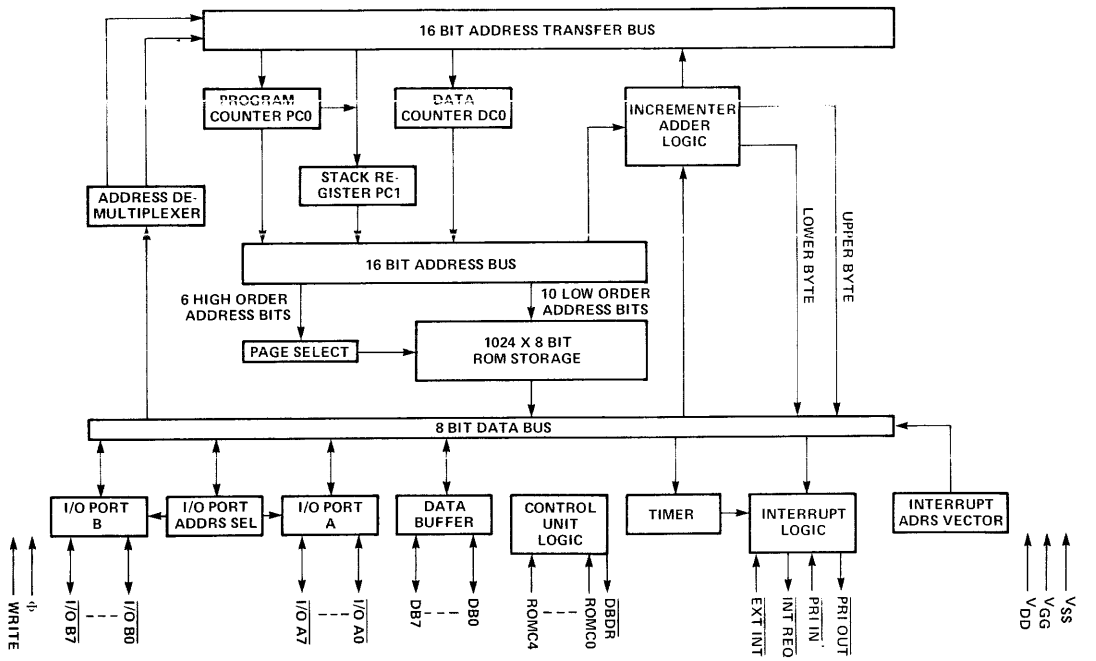


Figure 3-1. Logical Organization and Pins for the 3851 PSU

the state of every ROM bit, as well as certain address mask options which are external to the ROM array. Section 3.2.3 summarizes these address mask options.

3.1.2 The Program Counter (PC0) and Data Counter (DC0)

3851 PSU addressing logic consists primarily of two 16-bit registers: the program counter (PC0) and the data counter (DC0).

As explained in Section 1, the program counter will at all times address the memory word from which the next object program code must be fetched. The data counter addresses memory words containing individual data bytes, or bytes within data tables to be used as operands.

The provision of two address registers, PC0 and DC0, is a convenience to the 3850 CPU, and is not a necessary part of the memory addressing logic sequence within a 3851 PSU. The mechanism whereby an address is decoded by 3851 PSU logic is identical, whether the address originated in PC0 or in DC0.

Recall that PC0 always addresses the memory location out of which the next object program instruction byte will be read. If the instruction requires data (i.e., an operand) to be accessed, DC0 must address memory for this purpose; PC0 cannot be used to address data, since PC0 is saving the address of the next instruction code.

3.1.3 Page Select and Address Space

All memory addresses are 16-bits wide, whether the memory address originates in the program counter or the data counter. Address decode logic within the 3851 PSU separates the 16-bit address into two portions. The low-order 10 bits address one of the PSU's 1024 bytes of ROM storage. The high-order 6 bits constitute a page select.

Every 3851 PSU has a 6-bit page select register, which is a mask option that must be specified before the PSU ROM chip is created. If the high order six bits of the address coincide exactly with the page select mask, then an enable signal will be generated which causes PSU logic to respond to a memory access request. If the high order 6 bits of the address do not coincide exactly with the page

select, then no enabling signal is generated and the PSU will not respond to memory access requests.

The 6-bit page select register may be looked upon as identifying the memory addressing space of the individual 3851 PSU device. Each of the 64 page select options allowed by the 6-bit page select register identifies a single address space, consisting of 1024 contiguous memory addresses. Here are some examples:

Page Select Mask: 0 0 0 0 0 0

PSU Address Space:
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1
H'000' through H'03FF'

Page Select Mask: 0 0 1 0 1 1

PSU Address Space:
0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0
0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1
H'2C00' through H'2FFF'

Six high order address bits | Ten low order address bits

3.1.4 Addressing Consistency in Multiple Memory Devices

From Table 2-5 in Section 2, observe that those ROMC states which specify a memory access, call for only one memory device to respond to the memory access operation itself. On the other hand, every memory device responds to ROMC states that call for modification of program counter or data counter registers' contents. Consider two examples:

1. ROMC state 5 specifies that the data counter (DC0) register contents must be incremented. Every single memory device will simultaneously receive this ROMC state, and will simultaneously increment the contents of its DC0 register.
2. ROMC state 0 is the standard instruction fetch. Only the memory device whose address space includes the current contents of the program counter (PC0) registers will respond to this ROMC state by accessing memory and placing the contents of the addressed memory word on the 8-bit data bus. However, every memory device will increment the contents of its PC0 register, whether or not the PC0 register contents is within the memory space of the device.

Providing every memory device that is connected to the 8-bit data bus of a 3850 CPU is also connected to the ROMC control lines of the same CPU, address contentions can never arise; every memory device simultaneously receives the same ROMC state signals from the CPU; every memory device responds to ROMC states by identically modifying the contents of memory address registers, if such modifications are specified. Therefore every PC0 register, on every memory device, always contains identical information; the same will hold true for DC0 and PC1 registers. (There is one small exception which is discussed in Sections 4.1.1 and 5.1.1.)

Only one memory device, the one whose address space includes the specified memory address, will actually respond to any memory access request. To avoid addressing conflicts, it is only necessary to insure that the following three, self-evident conditions exist:

1. All memory devices must receive the same ROMC state signals from one CPU, and must contain identical PC and DC copies.
2. Page select masks must not be duplicated; that is, more than one memory device cannot have the same memory space.
3. The memory address contained in the specified register (PC0 or DC0) must be within the memory space of at least one memory device.

3.1.5 The Stack Register PC1

3851 PSU addressing logic contains a third 16-bit register, called the stack register. The stack register is labeled PC1 on Figure 3-1. The stack register is a buffer for the program counter PC0. The contents of the stack register are never used directly to address memory.

The following instructions access PC1, and are described in Table 2-7 of Section 2:

LR K,P	MOVE THE CONTENTS OF PC1 TO THE CPU SCRATCHPAD K REGISTERS
LR P,K	MOVE THE CONTENTS OF THE CPU K SCRATCHPAD REGISTERS TO PC1
PK	SAVE THE CONTENTS OF PC0 IN PC1 THEN MOVE THE CONTENTS OF CPU SCRATCHPAD REGISTERS 12 AND 13 TO PC0
PI	MOVE THE CONTENTS OF PC0 TO PC1 THEN LOAD THE HEXADECIMAL VALUE INTO PC0
POP	MOVE THE CONTENTS OF PC1 TO PC0

In addition, when an interrupt is acknowledged, the contents of PC0 is saved in PC1 as described in Section 3.4.7.

3.1.6 Incrementer Adder Logic

There are only two arithmetic operations that memory devices need to perform on the contents of memory address registers; they are:

1. Increment by 1 the 16-bit value stored in an address register.
2. Add an 8-bit value, treated as a signed binary number (subject to twos complement arithmetic) to the 16-bit value stored in an address register. If the 8-bit value is being treated as a signed binary number, then the high order bit of the 8-bit value is the sign bit; the sign bit must be propagated through the missing high order 8 bits as follows:

ADDITION EXAMPLE

```
  23A6  0 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0
+  7A   0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0
= 2420  0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0
```

Propagated positive sign bit

SUBTRACTION EXAMPLE

```
  23A6  0 0 1 0 0 0 1 1 1 0 1 0 0 1 1 0
-  7A   1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 0
= 232C  0 0 1 0 0 0 1 1 0 0 1 0 1 1 0 0
```

Propagated sign bit

Incrementer adder logic within the 3850 PSU may be likened to a primitive arithmetic and logic unit. The 3850 PSU control unit implements incrementer adder logic appropriately via control signals internal to PSU device logic. Table 2-5 defines all ROMC states, including those states in response to which PSU incrementer adder logic is used. Table 3-4 describes ROMC states from the PSU's point of view.

3.1.7 Interrupt Logic

This logic responds to an interrupt request signal which may originate internally from timer logic, or be input by an external device. Based on priority considerations, the interrupt request is passed on to the 3850 CPU, as described in Section 2.4.7.

3.1.8 Timer Logic

Every 3851 PSU has a polynomial shift register which may be used in conjunction with interrupt logic to generate real-time intervals.

Upon counting down to zero, the timer uses interrupt logic in order to signal that it has timed out.

The timer is programmable and is handled as though it were an I/O port. Using an OUT or OUTS instruction, a value may be loaded into the timer in order to determine the real-time period at the end of which a time-out interrupt will be generated. For detailed information on use of the timer, see Section 3.5.

3.1.9 The Data Bus

The 8-bit data bus is the main path for transfer of information between the 3850 CPU and other devices in the F8 microprocessor system. It is identified in Figure 3-1 by data lines DB0-DB7. This is the same data bus that has been described in Section 2.1.9.

3.1.10 I/O Ports

Every 3851 PSU has four, 8-bit I/O ports. Associated with the I/O ports is an I/O port address select register. This is a 6-bit register, the contents of which is a PSU mask option, which must be specified at the time the 3851 PSU is created.

Two of the four I/O ports, identified as I/O ports A and B in Figure 3-1, are used to transfer data to or

from external devices. A third I/O port is assigned to the programmable timer while the fourth port is used to access the interrupt control register.

The four I/O ports of any 3851 PSU are addressed via an 8-bit I/O port address, the high order 6 bits of which are specified by the I/O port address select as follows:

xxxxxx00 I/O Port A. See Section 3.4.3.
xxxxxx01 I/O Port B. See Section 3.4.3.
xxxxxx10 Interrupt logic control. See Section 3.6.
xxxxxx11 Programmable Timer. See Section 3.5.

xxxxxx represents a six binary digit, PSU mask option. For example, if the six binary digits are 000010, then the four I/O port addresses are H'08', H'09', H'0A' and H'0B'.

When a logic "1" is output to I/O port A or B, it places a 0 volt level on the output pin. This same inverted logic applies to input also. The I/O ports, timer, and interrupt control ports are not initialized during the power on reset.

3.2 SIGNAL DESCRIPTIONS, ELECTRICAL CHARACTERISTICS AND MASK OPTIONS

Figure 3-2 illustrates the 3851 PSU device pins. Signal names agree with Figure 3-1 and are summarized in Table 3-1.

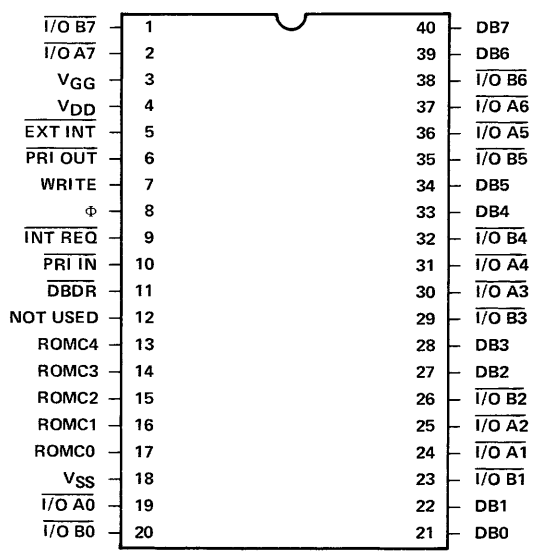


Figure 3-2. 3851 PSU Pin Assignments

Table 3-1. 3851 PSU Signals

PIN NAME	DESCRIPTION	TYPE
$\overline{\text{I/O A0-I/O A7}}$	I/O Port A	Input/Output
$\overline{\text{I/O B0-I/O B7}}$	I/O Port B	Input/Output
DB0-DB7	Data Bus	Bi-directional (3-State)
ROMC0-ROMC4	Control Lines	Input
Φ, WRITE	Clock Lines	Input
$\overline{\text{EXT INT}}$	External Interrupt	Input
$\overline{\text{PRI IN}}$	Priority In	Input
$\overline{\text{PRI OUT}}$	Priority Out	Output
$\overline{\text{INT REQ}}$	Interrupt Request	Output
$\overline{\text{DBDR}}$	Data Bus Drive	Output
V_{SS}, V_{DD}, V_{GG}	Power Supply Lines	Input

3.2.1 Signal Descriptions

Individual signals are described next. Signal characteristics are given in Table 3-2.

Φ and WRITE are the clock outputs from the 3850 CPU.

ROMC0 through ROMC4 are the control signals output by the 3850 CPU.

DB0 through DB7 are the bi-directional data bus lines which link the 3851 PSU with all other devices in the F8 system.

$\overline{\text{EXT INT}}$. A high to low transition on this signal is interpreted as an interrupt request from an external device.

$\overline{\text{PRI IN}}$. Unless this input signal is low, the 3851 PSU will not set $\overline{\text{INT REQ}}$ low in response to an interrupt.

$\overline{\text{PRI OUT}}$. This signal becomes $\overline{\text{PRI IN}}$ to the next device in the interrupt priority daisy chain. $\overline{\text{PRI OUT}}$ is output high unless $\overline{\text{PRI IN}}$ is entering the 3851 PSU low, and the 3851 PSU is not requesting an interrupt.

$\overline{\text{INT REQ}}$. This signal becomes the $\overline{\text{INT REQ}}$ input to the 3850 CPU. $\overline{\text{INT REQ}}$ must be output low in order to interrupt the 3850 CPU; this only occurs if $\overline{\text{PRI IN}}$ is low, and 3851 PSU interrupt control logic is requesting an interrupt.

$\overline{\text{I/O A0}}$ through $\overline{\text{I/O B7}}$ are Input/Output ports through which the 3851 PSU communicates with logic external to the microprocessor system.

$\overline{\text{DBDR}}$ is low when the 3851 PSU is outputting data on the data bus (DB0-DB7). For information on using $\overline{\text{DBDR}}$ see Section 3.4.1. $\overline{\text{DBDR}}$ is an open drain signal.

Table 3-2. A Summary of 3851 PSU Signal Characteristics

SIGNAL	SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
DATA BUS (DB0-DB7)	V_{IH}	Input High Voltage	2.9	V_{DD}	Volts	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = -100 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1.6 mA
	I_{IH}	Input High Current		1	μA	$V_{IN} = V_{DD}$, 3-State mode
	I_{OL}	Input Low Current		-1	μA	$V_{IN} = V_{SS}$, 3-State mode
CLOCK LINES (Φ, WRITE)	V_{IH}	Input High Voltage	4.0	V_{DD}	Volts	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_L	Leakage Current		3	μA	$V_{IN} = V_{DD}$
PRIORITY IN AND CONTROL LINES ($\overline{\text{PRI IN}}$, ROMC0-ROMC4)	V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_L	Leakage Current		3	μA	$V_{IN} = V_{DD}$
PRIORITY OUT ($\overline{\text{PRI OUT}}$)	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = -100 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 100 μA
INTERRUPT REQUEST ($\overline{\text{INT REQ}}$)	V_{OH}	Output High Voltage			Volts	Open Drain Output [1]
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1 mA
	I_L	Leakage Current		3	μA	$V_{IN} = V_{DD}$
DATA BUS DRIVE ($\overline{\text{DBDR}}$)	V_{OH}	Output High Voltage				External Pull-up
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 2 mA
	I_L	Leakage Current		3	μA	$V_{IN} = V_{DD}$
EXTERNAL INTERRUPT ($\overline{\text{EXT INT}}$)	V_{IH}	Input High Voltage	3.5		Volts	
	V_{IL}	Input Low Voltage		0.8	Volts	
	V_{IC}	Input Clamp Voltage		15	Volts	I_{IH} = 185 μA
	I_{IH}	Input High Current		10	μA	$V_{IN} = V_{DD}$
	I_{IL}	Input Low Current		-225	μA	V_{IN} = 2V
	I_{IL}	Input Low Current	-150	-500	μA	$V_{IN} = V_{SS}$
I/O PORT OPTION A (STANDARD PULL-UP)	V_{OH}	Output High Voltage	3.9(5)	V_{DD}	Volts	I_{OH} = -30 μA
	V_{OH}	Output High Voltage	2.9	V_{DD}	Volts	I_{OH} = -150 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1.6 mA
	V_{IH}	Input High Voltage	2.9(3)	V_{DD}	Volts	Internal Pull-up to V_{DD} [3]
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_L	Leakage Current		1	μA	$V_{IN} = V_{DD}$
	I_{IL}	Input Low Current		-1.6	mA	V_{IN} = 0.4V [4]
I/O PORT OPTION B (OPEN DRAIN)	V_{OH}	Output High Voltage				External Pull-up
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 2 mA
	V_{IH}	Input High Voltage	2.9(3)	V_{DD}	Volts	[3]
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_{IL}	Leakage Current		2	μA	V_{IN} = + 12V

Table 3-2. A Summary of 3851 PSU Signal Characteristics (Continued)

SIGNAL	SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
I/O PORT OPTION C (DRIVER PULL-UP)	V_{OH} V_{OL}	Output High Voltage Output Low Voltage	3.75 V_{SS}	V_{DD} 0.4	Volts Volts	I_{OH} = -1 mA I_{OL} = 1.6 mA

Notes:

1. Pull-up resistor to V_{DD} on CPU.
2. Positive current is defined as conventional current flowing into the pin referenced.
3. Hysteresis input circuit provides additional 0.3V noise immunity while internal/external pull-up provides TTL compatibility.
4. Measured while I/O port is outputting a high level.
5. Guaranteed but not tested.

3.2.2 Mask Options

The following mask options must be specified for every 3851 PSU:

1. The 1024 bytes of ROM storage. This will reflect programs and permanent data table stored in the PSU memory.
2. The 6-bit page select. This defines the PSU address space, as described in Section 3.1.3.
3. The 6-bit I/O port address select. This defines the four PSU I/O port addresses, as described in Section 3.1.10.
4. The 16-bit interrupt address vector, excluding bit 7. This address vector is described in Section 3.6.2, but for a complete understanding of its use, see the Guide to Programming the F8 Microcomputer.
5. The I/O port output option. The choices are between the standard Pull-up (Option A), the Open-Drain (Option B), and the Driver Pull-up (Option C). See Section 3.4.3 for further details.

3.2.3 Card Format Used to Define 3851 PSU Mask Options

Mask options are specified using a card file which may include the following types of card:

- Option card,
- Comment cards,
- 'X' cards (text format commands), and
- 'C' cards (ROM truth table data).

OPTION CARD FORMAT

The option card should always be the first card in the input data file. The format of the option card follows:

Column	1-20	26-30	35-36	40-42	45	50-53	58-60	63-65
	User	SL	ROM	IO	Port	Timer	HEX DEC —	HEX DEC —

User is the customer name

SL is a 5-digit SL number for the device assigned by FSC

ROM is the ROM number (0-63 decimal) Specifies ROM page

IO is the decimal number of the first I/O port of the group selected

Port is 1 for Standard I/O, 2 for Open Drain, 3 for Output Only } Refer to Section 3.4.3

Timer is the Timer/External Interrupt Address Vector (4 Hexadecimal digits) (Refer to Section 3.6.3)

Columns 58-60 specify the desired number base for the address field on the output listing.

Columns 63-65 specify the desired number base for the data fields on the output listing. Each defaults to DECIMAL when not specified. All other fields on the option card must be specified.

COMMENT CARD FORMAT

Each comment card must have an asterisk (*) in column 1. All other columns are ignored. A comment card may occur any time after the option card in the input file. Comment cards are optional.

TEXT FORMAT CARD FORMAT

The text format commands are used to describe the format of the ROM data cards which follow. Text format commands should have the character 'X' in

column 1 and should precede all ROM data cards. The valid text format commands are:

X SEQUENCE
indicates that the ROM data has sequence numbers in columns 77-79. This command causes F8 ROM to do sequence checking.

X BASE HEX HEX
DEC DEC
— —
specifies the number base of the ROM address input and the ROM data input respectively. If no X BASE card occurs, all fields are assumed to be decimal.

DATA CARD FORMAT

The data cards for F8 PSUs must have the character 'C' in column 1. The ROM truth table data card format is as follows:

Column	1	2-9	10-12	14-16	17-19	20-22	.....	77-79
	C	Add	Bytes	Data1	Data2	Data3	.....	Data 22

Add is the ROM address of the first data field on the card
Bytes is the number of bytes of data on the card (<23)
Datan specifies the data to be coded at ROM address (Add + n - 1) for 0 < n < = Bytes
Data22 is a sequence number if an X SEQUENCE card has occurred

EXAMPLE OF F8ROM INPUT DECK

```
SUPER ELECTRONICS 3000        0      4   1 0010
C  0 10  33 35181 27 24182 32177 11 91
C 10 10 143 54 97 93 43254 98111 92 98
C 20 10 114 21 92 99111 32 63 87116 90
C 30 10 119176 31180 14182 27114 35 97
C 40 10  64 92 98 66 92111 27 67 31145
C 50 10  23 83155 20 68 31145 16 89144
C 60 10  13 69 32245  9 77144  6 70 31
C 70 10 145  2 86161 94254247 80144 82
C 80 10 118176 22176 75224132216  0224
C 90 10 132219  1224132 11  2224132 19
```

3.2.4 Paper Tape and Cartridge Format Used to Define 3851 PSU Mask Options

Information concerning the use of paper tapes and cartridges as a medium for ordering 3851 PSU custom devices can be obtained by contacting Fairchild's MicroSystems Division Marketing.

3.2.5 Electrical Specifications

Absolute Maximum Ratings (Above which useful life may be impaired)

V_{GG}	+15V to -0.3V
V_{DD}	+7V to -0.3V
I/O Port Open Drain Option	+15V to -0.3V
External Interrupt Input	-600 μA to +225 μA
All other inputs & outputs	+7V to -0.3V
Storage Temperature	-55°C to +150°C
Operating Temperature	0°C to +70°C

Note: All voltages with respect to V_{SS}.

DC Characteristics: V_{SS} = 0V, V_{DD} = +5V ± 5%, V_{GG} = +12V ± 5%, T_A = 0°C to +70°C

SUPPLY CURRENTS

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
I_{DD}	V_{DD} Current		28	60	mA	f = 2 MHz, Outputs Unloaded
I_{GG}	V_{GG} Current		10	30	mA	f = 2 MHz, Outputs Unloaded

3.3 CLOCK TIMING

All timing within the 3851 PSU is controlled by Φ and WRITE, which are input from the 3850 CPU. For a description of these clock signals, and how they are generated, see Section 2.3.

The WRITE clock refreshes and updates 3851 PSU address registers, which are dynamic.

The Φ clock drives sequencing logic to precharge the ROM matrix. The Φ clock also drives the programmable timer.

3.4 INSTRUCTION EXECUTION

The 3851 PSU responds to signals which are output by the 3850 CPU in the course of implementing instruction cycles. Figure 2-9 illustrates timing for instruction cycles and ROMC signals being output by the CPU.

Table 3-4 summarizes the data bus response of the 3851 PSU to the ROMC states described in Table 2-5.

Table 3-3. A Summary of 3851 PSU Signal AC Characteristics

AC Characteristics: V_{SS} = 0V, V_{DD} = +5V ± 5%, V_{GG} = +12V ± 5%, T_A = 0°C to +70°C

Symbols in this table are used by all figures in Section 3.

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
$P\phi$	ϕ Period	0.5		10	μS	
PW_1	ϕ Pulse Width	180		$P\phi$-180	nS	t_r, t_f = 50 nS typ.
td_1	ϕ to WRITE + Delay			250	nS	C_L = 100 pf
td_2	ϕ to WRITE-Delay			250	nS	C_L = 100 pf
td_4	WRITE to DB Input Delay			$2P\phi$ + 1.0	μS	
PW_2	WRITE Pulse Width	$P\phi$-100		$P\phi$	nS	t_r, t_f = 50 nS typ.
PW_S	WRITE Period; Short		$4P\phi$			
PW_L	WRITE Period; Long		$6P\phi$			
td_3	WRITE to ROMC Delay			550	nS	
td_7	WRITE to DB Output Delay WRITE to $\overline{DBDR}$ — Delay	$2P\phi$ + 100 - td_2	$2P\phi$ + 200	$2P\phi$ + 850 - td_2	nS	C_L = 100 pf
td_8	WRITE to $\overline{DBDR}$ + Delay		200		nS	Open Drain
tr_1	WRITE to $\overline{INT\ REQ}$ — Delay			430	nS	C_L = 100 pf [1]
tr_2	WRITE to $\overline{INT\ REQ}$ + Delay			430	nS	C_L = 100 pf [3]
tpr_1	PRI IN to $\overline{INT\ REQ}$ — Delay		200		nS	C_L = 100 pf [2]
tpd_1	PRI IN to $\overline{PRI\ OUT}$ — Delay			300	nS	C_L = 50 pf
tpd_2	PRI IN to $\overline{PRI\ OUT}$ + Delay			300	nS	C_L = 50 pf
tpd_3	WRITE to $\overline{PRI\ OUT}$ + Delay			600	nS	C_L = 50 pf
tpd_4	WRITE to $\overline{PRI\ OUT}$ — Delay			600	nS	C_L = 50 pf
t_{sp}	WRITE to Output Stable			1.0 (3)	μS	C_L = 50 pf, Standard Pull-up
t_{od}	WRITE to Output Stable			1.0 (3)	μS	C_L = 50 pf, R_L = 12.5 KΩ to V_{DD} plus TTL load
t_{dp}	WRITE to Output Stable		200	400	nS	C_L = 50 pf, Driver Pull-up
t_{su}	I/O Setup Time	1.3			μS	
t_h	I/O Hold Time	0			nS	
t_{ex}	$\overline{EXT\ INT}$ Setup Time	400			nS	

Notes:

1. Assume Priority In was enabled ($\overline{PRI\ IN}$ = 0) in previous F8 cycle before interrupt is detected in the PSU.
2. PSU has interrupt pending before priority in is enabled.
3. Assume pin tied to $\overline{INT\ REQ}$ input of the 3850 CPU.
4. The parameters which are shaded in the table above represent those which are most frequently of importance when interfacing to an F8 system. Unshaded parameters are typically those that are relevant only between F8 chips and not normally of concern to the user.
5. Input and output capacitance is 3 to 5 pf typical on all pins except V_{DD}, V_{GG}, and V_{SS}.

Table 3-4. Data Bus Contents as Function of ROMC State for the 3851 PSU

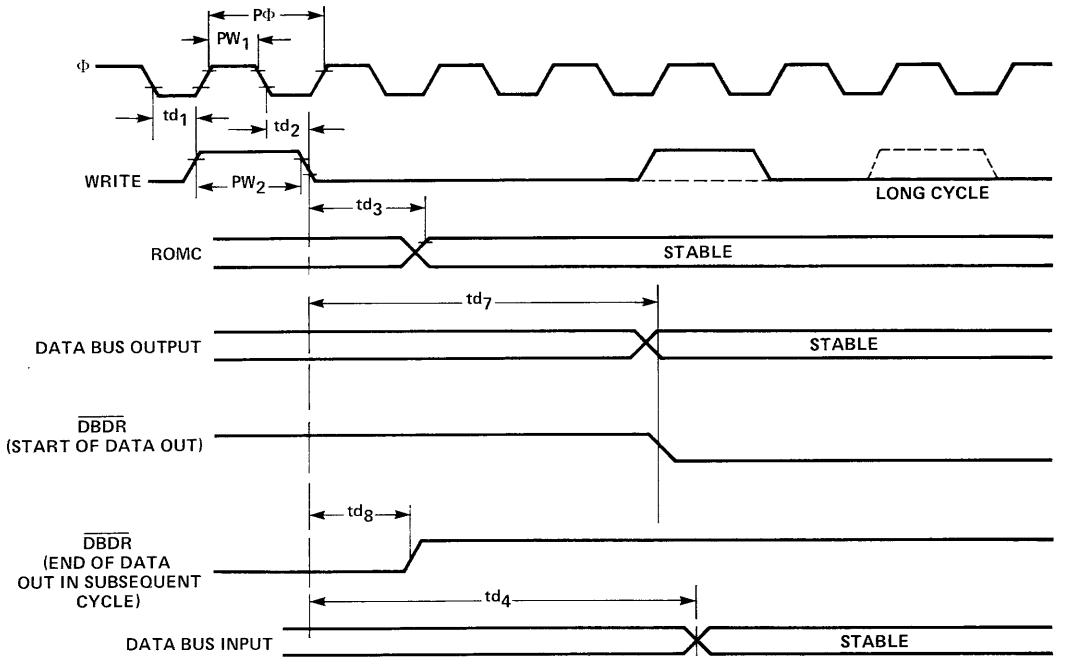
ROMC STATE (HEX)	IF 3851 PSU IS THE SOURCE*		IF 3850 CPU IS THE SOURCE, DESCRIPTION OF DATA
	DESCRIPTION OF DATA	ADDRESS**	
00	Instruction	PC0	
01	Offset for branch	PC0	
02	Operand	DC0	
03	Operand	PC0	
04			
05			Byte to be stored
06	Upper byte, DC0		
07	Upper byte, PC1		
08			= 00 for PC0
09	Lower byte, DC0		
0A			Offset for DC0
0B	Lower byte, PC1		
0C	Byte for PC0, Lower	PC0	
0D			
0E	Byte for DC0, Lower	PC0	
0F	Lower byte of Interrupt Vector if it is source of the interrupt		
10			
11	Byte for DC0, Upper	PC0	
12			Byte for PC0, Lower
13	Upper byte of Interrupt Vector if it is source of the interrupt		
14			Byte for PC0, Upper
15			Byte for PC1, Upper
16			Byte for DC0, Upper
17			Byte for PC0, Lower
18			Byte for PC1, Lower
19			Byte for DC0, Lower
1A			Byte for selected I/O Port
1B	Byte from I/O register, if selected		
1C			(Note 1)
1D			
1E	Lower byte, PC0		
1F	Upper byte, PC0		

*Will only drive data bus within the segment of Address space that belongs to the 3851 PSU.

**An entry in this column specifies the register from which a memory address was obtained.

Note:

1. During INS or OUTS instruction for port 0 or 1: I/O byte
   During INS or OUTS instruction for port 4-F: I/O address
   During all other instructions: 3850 doesn't drive



SYMBOLS ARE DEFINED IN TABLE 3-3

Figure 3-3. 3851 PSU Data Bus Timing

Table 3-5. Conversion of Timer Contents into Timer Counts

TIMER CONTENTS	TIMER COUNTS	TIMER CONTENTS	TIMER COUNTS	TIMER CONTENTS	TIMER COUNTS	TIMER CONTENTS	TIMER COUNTS
FE	254	68	237	B6	220	F5	203
FD	253	D1	236	6C	219	EA	202
FB	252	A3	235	D9	218	D4	201
F7	251	47	234	B2	217	A9	200
EE	250	8F	233	64	216	52	199
DC	249	1F	232	C8	215	A4	198
B8	248	3F	231	91	214	49	197
71	247	7E	230	23	213	92	196
E3	246	FC	229	46	212	25	195
C7	245	F9	228	8D	211	4A	194
8E	244	F3	227	1B	210	94	193
1D	243	E6	226	37	209	29	192
3B	242	CD	225	6F	208	53	191
76	241	9B	224	DF	207	A6	190
ED	240	36	223	BE	206	4D	189
DA	239	6D	222	7D	205	9A	188
B4	238	DB	221	FA	204	34	187

Table 3-5. Conversion of Timer Contents into Timer Counts (Continued)

TIMER CONTENTS	TIMER COUNTS	TIMER CONTENTS	TIMER COUNTS	TIMER CONTENTS	TIMER COUNTS	TIMER CONTENTS	TIMER COUNTS
69	186	86	139	4C	92	50	45
D3	185	0C	138	98	91	A0	44
A7	184	18	137	30	90	41	43
4F	183	31	136	61	89	83	42
9E	182	63	135	C2	88	06	41
3C	181	C6	134	84	87	0D	40
78	180	8C	133	08	86	1A	39
F0	179	19	132	10	85	35	38
E0	178	33	131	20	84	6B	37
C1	177	67	130	40	83	D7	36
82	176	CE	129	81	82	AF	35
04	175	9D	128	02	81	5E	34
09	174	3A	127	05	80	BD	33
12	173	74	126	0B	79	7B	32
24	172	E9	125	16	78	F6	31
48	171	D2	124	2C	77	EC	30
90	170	A5	123	59	76	D8	29
21	169	4B	122	B3	75	B0	28
42	168	96	121	66	74	60	27
85	167	2D	120	CC	73	C0	26
0A	166	5B	119	99	72	80	25
14	165	B7	118	32	71	00	24
28	164	6E	117	65	70	01	23
51	163	DD	116	CA	69	03	22
A2	162	BA	115	95	68	07	21
45	161	75	114	2B	67	0F	20
8B	160	EB	113	57	66	1E	19
17	159	D6	112	AE	65	3D	18
2E	158	AD	111	5C	64	7A	17
5D	157	5A	110	B9	63	F4	16
BB	156	B5	109	73	62	E8	15
77	155	6A	108	E7	61	DO	14
EF	154	D5	107	CF	60	A1	13
DE	153	AB	106	9F	59	43	12
BC	152	56	105	3E	58	87	11
79	151	AC	104	7C	57	0E	10
F2	150	58	103	F8	56	1C	9
E4	149	B1	102	F1	55	39	8
C9	148	62	101	E2	54	72	7
93	147	C4	100	C5	53	E5	6
27	146	88	99	8A	52	CB	5
4E	145	11	98	15	51	97	4
9C	144	22	97	2A	50	2F	3
38	143	44	96	55	49	5F	2
70	142	89	95	AA	48	BF	1
E1	141	13	94	54	47	7F	0
C3	140	26	93	A8	46		

3.4.1 Data Output by the PSU

Figure 3-3 provides timing when the 3851 PSU outputs data on the data bus. This timing applies whenever a 3851 PSU is the data source. With reference to Figure 2-11, note that the 3851 PSU places data on the data bus, even in the worst case, in time for the set-up required by any 3850 CPU destination.

Observe that $\overline{DBDR}$ is low while data output by the 3851 PSU is stable on the data bus. Thus $\overline{DBDR}$ low indicates that the data bus currently contains data flowing from a 3851 PSU. For systems with more than one 3851 PSU (as described in Section 11), the $\overline{DBDR}$ outputs may be wire-ORed, and the result may be used as a bus data flow direction indicator. $\overline{DBDR}$ may remain low until td_8 into the instruction cycle following the one in which $\overline{DBDR}$ was set low.

3.4.2 Data Input to the PSU

When the 3851 PSU receives data off the data bus, in the worst case the data must be added to a 16-bit number within the PSU's Adder/Incrementer. This worst case corresponds to data coming from the accumulator of the CPU for an ADC instruction, or from a memory device for a BR instruction. For this worst case, arriving data must allow sufficient time for 16-bit Adder logic. td_4 in Figure 3-3 identifies this worst case timing.

3.4.3 Input/Output Interfacing

The two PSU I/O ports with addresses xxxxxx00 and xxxxxx01 (xxxxxx is the six-bit I/O port address select) may be used to transmit data between the PSU and external devices. IN and INS instructions cause data at the I/O ports to be transmitted to the CPU; OUT and OUTS instructions cause data in the CPU's accumulator to be loaded into an I/O port. Each I/O pin has an output latch which holds the pin's DC data.

Data bus timing associated with the execution of I/O instructions does not differ from data bus timing associated with any other data transfer to, or from the PSU. However, timing at the I/O port itself depends on which port option is being used. Figures 3-4, 3-5 and 3-6 illustrate the three port options. Figure 3-7 illustrates timing for the three cases.

Figure 3-4 illustrates the standard pull-up configuration which is described in Section 2.4.6.

When the I/O port is configured as shown in Figure 3-5, the drain connection of FET (a) is "open", i.e., not connected to V_{DD} through a pull-up transistor. This option is most useful in applications where several signals (possibly several I/O port lines) are to be wire-ORed together. A common external pull-up, R_L, is used to establish the logic 1 levels. Another advantage of this option is that the output (point Y) may be tied through a pull-up resistor to a voltage higher than V_{DD} (clear up to V_{GG}) for interfacing to external circuits requiring a higher logic 1 level than V_{DD} would provide.

The process of inputting and outputting with this configuration can be described as follows:

If a high level is present at point X, (this would be coming from the port latch), FET (a) will conduct and pull point Y to a low level by current flow through R_L. This low level at Y will cause transistor (b) to turn on and present a low level to the input TTL circuit. On the other hand, if a low level is present at X, FET (a) will turn off and point Y will be pulled toward V_{DD} by R_L. This causes transistor (b) to turn off and present a high level to the internal TTL circuits.

When data is input, a high level at the base of transistor (c) causes (c) to conduct and pull point Y low through current flow through R_L. This transfers a high level to the internal I/O port logic through inverting action by the hysteresis circuit. If a low level is present at the base of (c), conduction stops and point Y is pulled toward V_{DD} by R_L. This is then transferred as a low level to the internal I/O port logic through the hysteresis circuit.

Figure 3-6 shows the I/O port driver pull-up option —here used to drive an LED indicator. This application is typical of a front-panel address or data display, where a row of LED indicators shows the logic state at each pin of an I/O port. In this case, a high level at X turns FET (b) on and (a) off, providing a path for current through resistor R from the base of transistor (c). This stops (c) from conducting and the LED does not light. If, however, a low level is present at X, (b) turns off and (a) turns on, providing a path for current from V_{DD} through (a) to R. This current through R turns on (c), which causes the LED to conduct and be lighted.

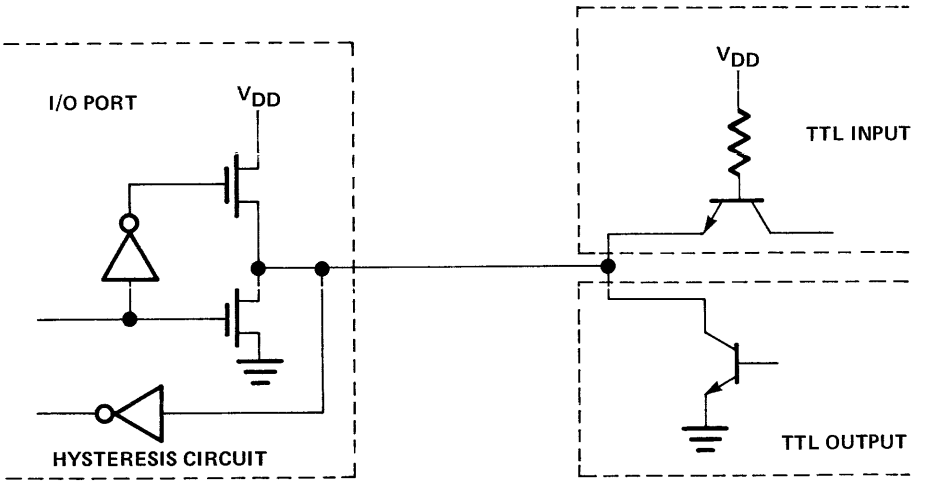


Figure 3-4. Standard Pull-up Configuration

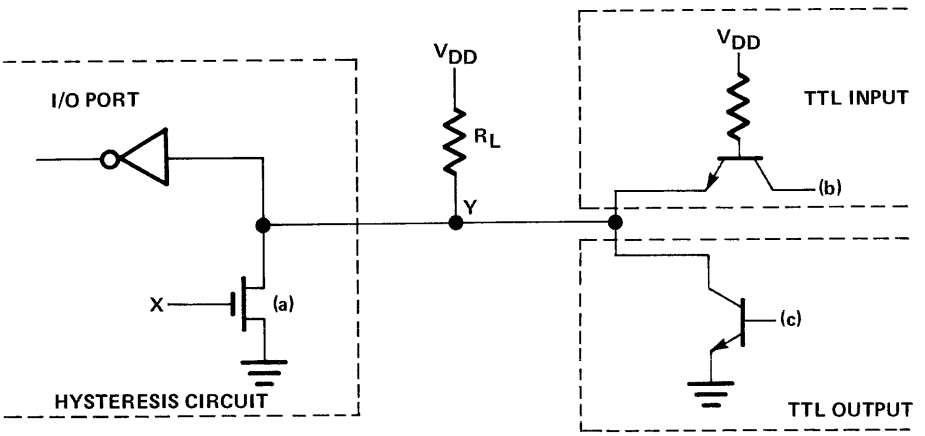


Figure 3-5. Open Drain Configuration

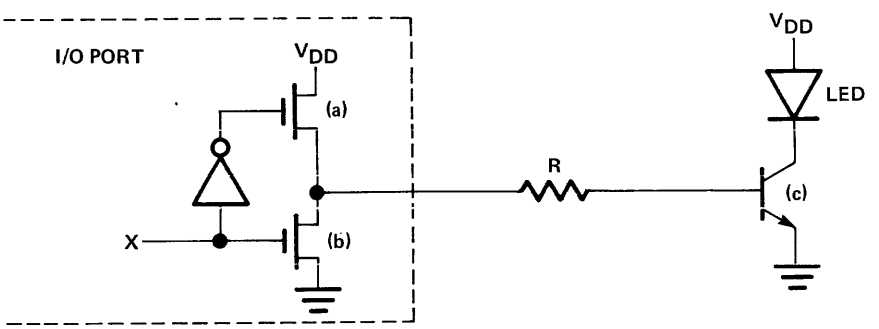
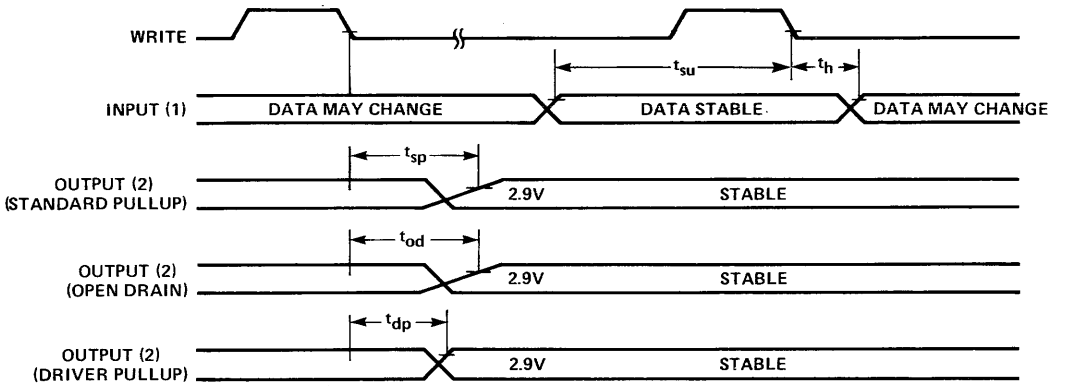


Figure 3-6. Driver Pull-up Configuration



SYMBOLS ARE DEFINED IN TABLE 3-3

1. The set-up and hold times specified are with respect to the end of the second long cycle during execution of the three cycle IN or INS instruction.
2. All delay times are specified with respect to the end of the second long cycle during execution of the three cycle OUT or OUTS instruction.

Figure 3-7. Timing at PSU I/O Ports

The three options for I/O port output configurations described above are provided to aid the designer in optimizing (minimizing) the system hardware for his particular application. The option is specified as a mask option by the designer as described in Section 3.2.3.

3.5 THE PROGRAMMABLE TIMER

The 3851 PSU has an 8-bit shift register, addressable as I/O port xxxxxx11, which may be used as a programmable timer. (xxxxxx is the 6-bit I/O port address select, a PSU mask option.) Figure 3-8 illustrates the shift register logic and the exclusive-OR feedback path.

Based on the logic illustrated in Figure 3-8, binary values in the range 0 through 254, when loaded into the timer, are converted into "timer counts," as shown in Table 3-5. In Table 3-5, "Timer Contents" is the actual binary value loaded into a timer, and "Timer Counts" is the corresponding number of time intervals the timer will take to time out. Data cannot be read out of the programmable timer I/O port.

As described in the Guide to Programming the F8 Microcomputer, an assembly language program specifies "timer counts," and the assembler converts timer counts into the binary value which must be loaded into the programmable timer; this is the value given under "Timer Contents" in Table 3-5. A logic designer who wishes to use a programmable timer, bypassing assembly language programming, must load the programmable timer with the value given under "Timer Contents" in Table 3-5, to time out after the number of intervals given under "Timer Counts" in Table 3-5.

It is also possible to write small subroutines that calculate time values one count faster or slower than a given value. Such subroutines would be used if programmed delays are required.

The OUT or OUTS instruction is used to load "timer counts" into the programmable timer. The contents of the programmable timer can not be read using an IN or INS instruction. The timer will time out after a time interval given by the product:

(period of clock Φ) X (timer counts) X 31

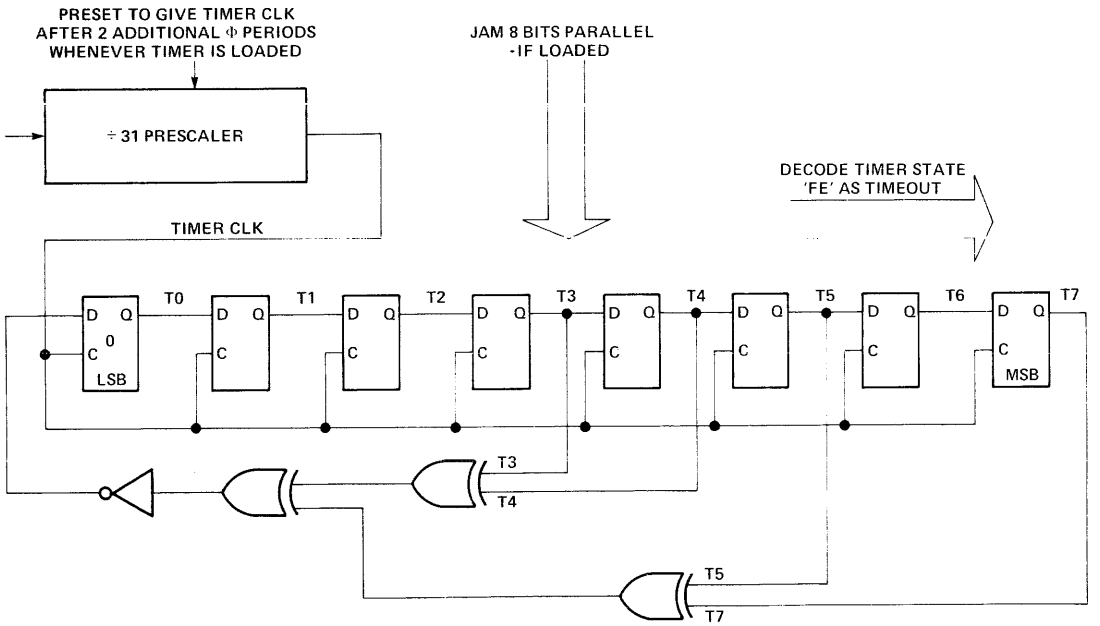


Figure 3-8. Timer Block Diagram

For example, a value of 200 (11001000, or H'C8') loaded into the programmable timer, becomes 215 timer counts. The timer will therefore time out in 3.33 milliseconds, if the period of clock signal Φ is 500 nanoseconds.

A value of 255 (H'FF') loaded into a programmable timer will stop the timer.

All timers run continuously, unless they have been stopped by loading H'FF' into the timer. Upon timing out, the timer transmits an interrupt request to the interrupt logic (described in Section 3.6). If proper interrupt logic conditions exist, the timer interrupt request is passed on to the CPU via $\overline{\text{INT REQ}}$.

After a programmable timer has timed out, it will again time out after 255 timer counts; therefore if the programmable timer is simply left running, it will time out ever 7905 Φ clock periods, or every 3.953 milliseconds for a 500 nanosecond clock.

Observe that if the timer is actually loaded with a zero value, then it will time out in 24 counts, whereas once it has timed out, it will next time out in 255 counts; in other words, a time out is not the same thing as counting down to zero.

Whenever the timer and timer interrupt are being set to time a new interval, always load the timer before enabling the timer interrupt. The act of loading the timer clears any pending timer interrupts. When the timer interrupt is enabled, any pending timer interrupt will be acknowledged and forwarded to the CPU. Since the timer runs continuously, unless stopped under program control, enabling the timer before loading a time count can cause errors. Prior time outs of the timer will be latched in the interrupt logic of the PSU, even while timer interrupts are disabled. When the timer is enabled, an immediate interrupt acknowledge will occur if, by chance, the continuous running timer happened to time out while timer interrupts were disabled.

On the other hand, if the timer is loaded just prior to enabling timer interrupts, loading the timer clears pending timer interrupts, Now a spurious interrupt request will not exist when the timer interrupt is enabled.

Figure 3-9 illustrates a possible signal sequence for a timer which is initially loaded with 200, then allowed to run continuously.

3.6 INTERRPUT LOGIC

The 3851 PSU interrupt system is illustrated conceptually in Figure 3-10. Figures 3-11 and 3-12 show the timer and external interrupt event sequences which occur during interrupt processing.

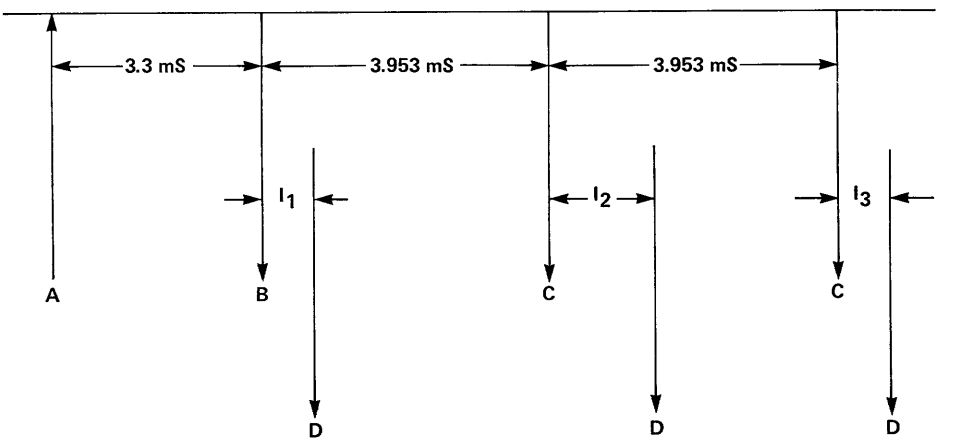
3.6.1 Interrupt Logic Organization

The Interrupt Control Register (I/O port) has the I/O port address xxxxxx10, where xxxxxx is the 6-bit I/O port address select. Data is loaded into this register (I/O port) using an OUT or OUTS instruction. Data cannot be read out of this register (I/O port). The contents of the Interrupt Control Register (I/O port) is interpreted as follows:

CONTENTS OF I/O PORT	INTERPRETATION
B'xxxxxx00'	Disable all interrupts
B'xxxxxx01'	Enable external interrupt, disable time interrupt
B'xxxxxx10'	Disable all interrupts
B'xxxxxx11'	Disable external interrupt, enable timer interrupt

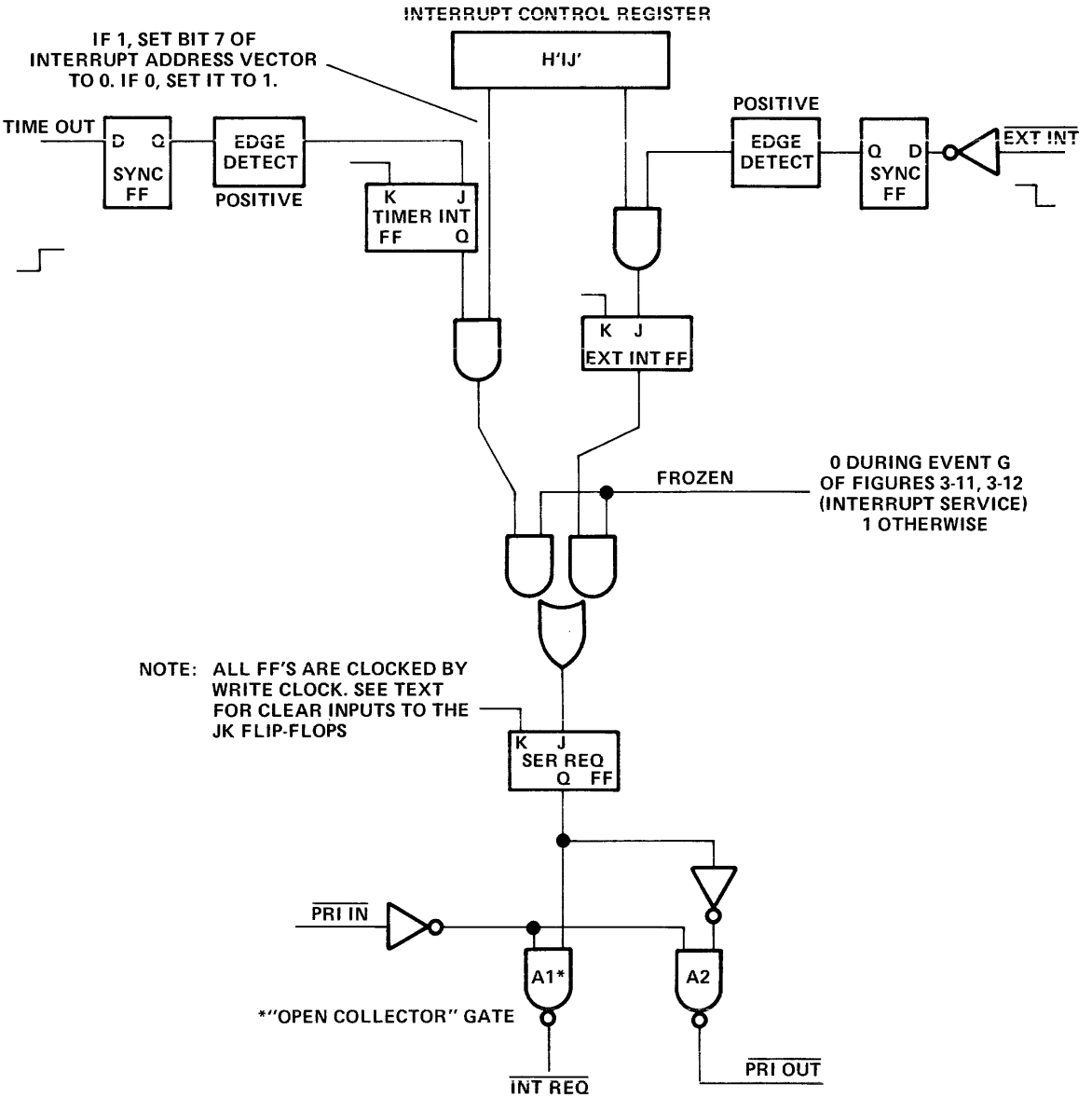
In the above I/O port contents definitions, x represents "don't care" binary digits.

Depending on the contents of the Interrupt Control Register (I/O port), a 3851 PSU's interrupt control logic can be accepting timer interrupts, or external interrupts, or neither, but never both.



A – 200 loaded into timer.
B – First time out.
C – Second, and subsequent time outs.
D – Interrupt Service Routines being entered by CPU.
I_1, I_2, I_3 – Intervals between time out interrupt request reacing interrupt logic, and service routines being entered by CPU.

Figure 3-9. Time Out and Interrupt Request Timing



NOTE: ALL FF'S ARE CLOCKED BY WRITE CLOCK. SEE TEXT FOR CLEAR INPUTS TO THE JK FLIP-FLOPS

Figure 3-10. Conceptual Illustration of 3851 PSU Interrupt Logic

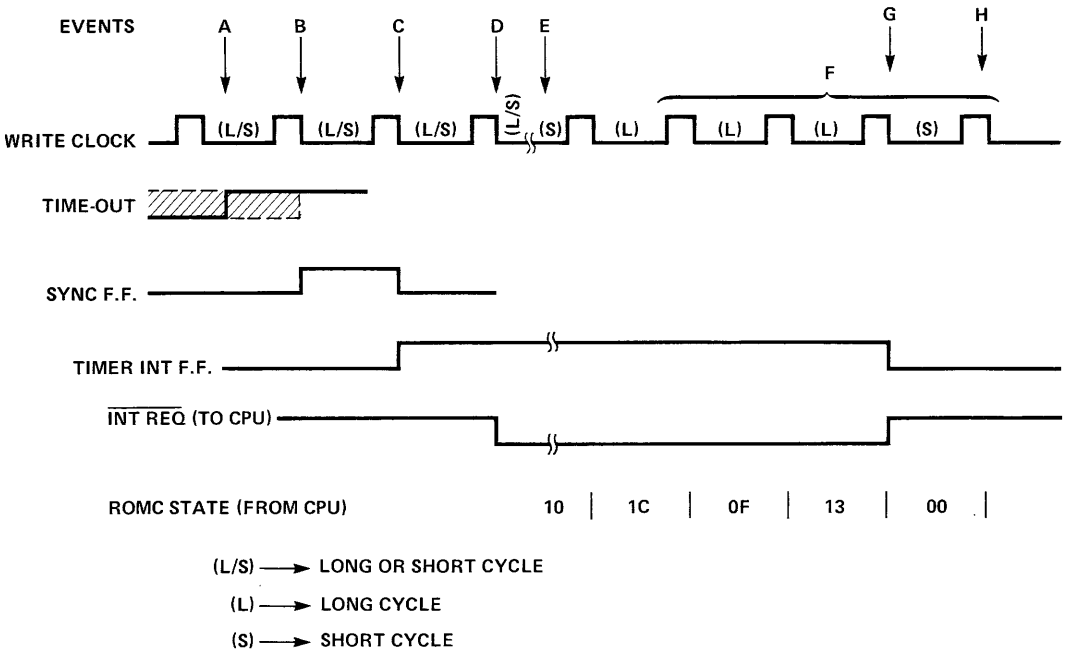


Figure 3-11. Timer Interrupt Sequence

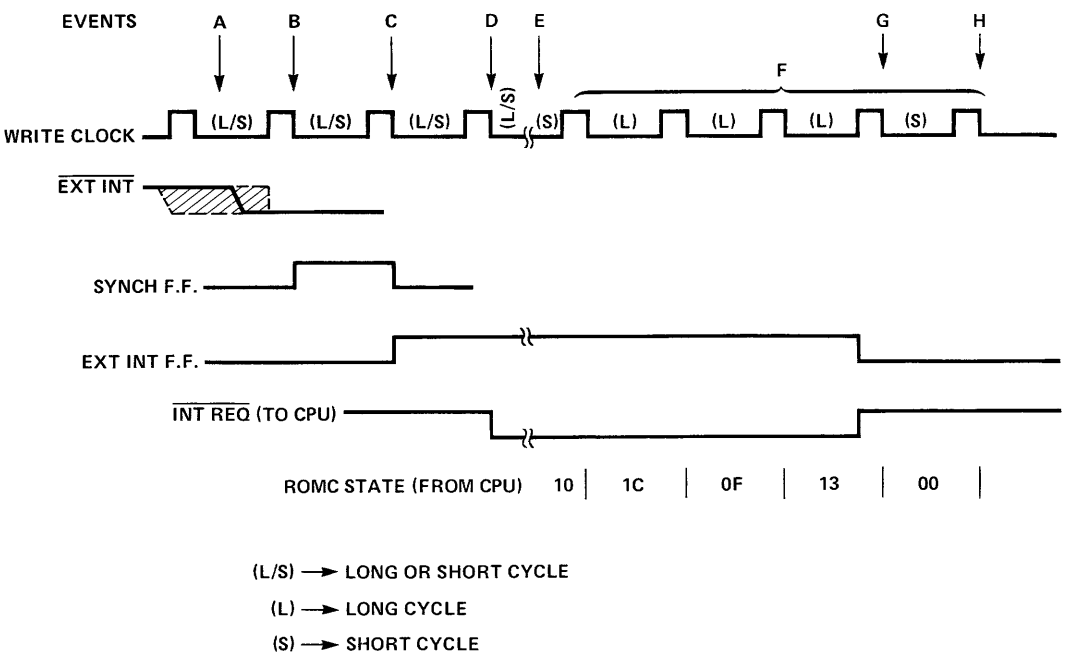


Figure 3-12. External Interrupt Sequence

Figure 3-10 is a conceptual logic diagram of the PSU's interrupt logic. Between the $\overline{\text{EXT INT}}$ input or the time-out input and the output $\overline{\text{INT REQ}}$, there are 3 flip-flops. $\overline{\text{EXT INT}}$ and the time-out interrupt input each have a synchronizing flip-flop and edge detect logic.

Each edge detect block is followed by its own INTERRUPT flip-flop which latches the true condition.

The outputs of the TIMER INTERRUPT flip-flop and the EXTERNAL INTERRUPT flip-flop are ORed to set the SERVICE REQUEST flip-flop, providing that an interrupt from some other PSU is not being acknowledged.

$\overline{\text{INT REQ}}$ is the NAND of PRI IN and SERVICE REQUEST.

$\overline{\text{INT REQ}}$ is an open drain signal. The $\overline{\text{INT REQ}}$ signal of several PSUs may be tied together so that any one can force the line to 0V if it is requesting interrupt service; a pull-up to V_{DD} is provided by the 3850 CPU to $\overline{\text{INT REQ}}$ input pin.

$\overline{\text{PRI IN}}$ is part of the interrupt priority chain. The chain begins by a strap to V_{SS}. Each device in the chain has a $\overline{\text{PRI IN}}$ input and a $\overline{\text{PRI OUT}}$ output. $\overline{\text{PRI OUT}}$ of the PSU will be active (0V) only if $\overline{\text{PRI IN}}$ is active (0V) and SERVICE REQUEST is inactive. This means that $\overline{\text{PRI OUT}}$ and $\overline{\text{INT REQ}}$ are always at opposite levels. $\overline{\text{PRI OUT}}$ becomes $\overline{\text{PRI IN}}$ for the next device in the interrupt priority daisy chain, if there is one. The function of the priority daisy chain is to insure that just one device at a time be requesting interrupt service.

The SERVICE REQUEST flip-flop cannot become set if another interrupt request is in the process of being acknowledged anywhere in the system. Rather, if an interrupt request has been latched into the TIMER INTERRUPT flip-flop, or the EXTERNAL INTERRUPT flip-flop, the PSU logic waits until after the process of acknowledging the other interrupt has been completed, before setting SERVICE REQUEST. This precaution is necessary to insure that the priority chain is not altered during acknowledgement; chaos would result if half of the interrupt vector came from one device and the second half from some other device.

The SERVICE REQUEST flip-flop is cleared after an interrupt from the PSU has been acknowledged. It is also cleared whenever the PSU's interrupt control register is accessed by an output instruction.

The conditions for setting the TIMER INTERRUPT flip-flop and the EXTERNAL INTERRUPT flip-flop differ slightly. External interrupts must be enabled before the EXTERNAL INTERRUPT flip-flop can be set by a negative going transition of $\overline{\text{EXT INT}}$. However, TIMER INTERRUPT will be set by a timer TIME OUT, independent of the timer interrupt enable bit. This means that the PSU can detect a time out interrupt that is requested while the PSU was checking for external interrupts.

The TIMER INTERRUPT flip-flop is cleared whenever the PSU's timer is loaded, or when its timer interrupt has been acknowledged. The EXTERNAL INTERRUPT flip-flop is cleared whenever the PSU's interrupt control register is accessed by an output instruction, or when its external interrupt has been acknowledged.

3.6.2 Interrupt Acknowledge Sequence

Upon receiving an interrupt request, whether from an external source via $\overline{\text{EXT INT}}$ or from the internal timer, the PSU and CPU go through an interrupt sequence which ultimately results in the execution of an interrupt service routine located at the memory address pointed to by the Interrupt Address Vector. Figures 3-11 and 3-12 illustrate the interrupt sequences for the two cases. Events occurring in these sequences are labeled with the letters A through H. Events are described as follows.

EVENT A –
The initial interrupt request arrives. The falling edge of $\overline{\text{EXT INT}}$ pin identifies an external interrupt. The rising edge of interval timer output indicates a time-out.

EVENT B –
The synchronizing flip-flop in the PSU control logic changes state.

EVENT C –
The timer interrupt, or external interrupt flip-flop goes true, indicating the local interrupt logic's acknowledgement of the interrupt. The timer interrupt flip-flop will always respond and save the time-out occurrance, whereas the external interrupt flip-flop will only be set at this time if the external interrupt mode is enabled within the local control logic.

EVENT D –
The $\overline{\text{INT REQ}}$ line is pulled low by the PSU, passing the request for servicing on to the CPU. The conditions that must be present for this to occur are:

The $\overline{\text{PRI IN}}$ pin must be low.

The proper enable state must exist in the local control logic for the type of interrupt (timer or external).

The system is not already into Event F due to servicing some other interrupt.

EVENT E –
The CPU now begins its response to the $\overline{\text{INT REQ}}$ line by outputting the unique ROMC state H'10'. This will only occur when the following conditions are satisfied:

The CPU is executing the last cycle of an instruction (beginning an instruction fetch).

The ICB is enabled ($\overline{\text{ICB}}$ = 0).

The current instruction fetch is not protected (see Table 2-6).

EVENT F –
The CPU generates the interrupt acknowledge sequence of ROMC states. See Table 2-6 under INTRPT for details.

EVENT G –
At this point the CPU begins fetching the first instruction of the interrupt service routine. In the PSU interrupt logic, the SERVICE REQUEST flip-flop and the appropriate INTERRUPT REQUEST flip-flop have been cleared.

EVENT H –
The CPU begins executing the first instruction of the interrupt service routine.

3.6.3 Interrupt Address Vector

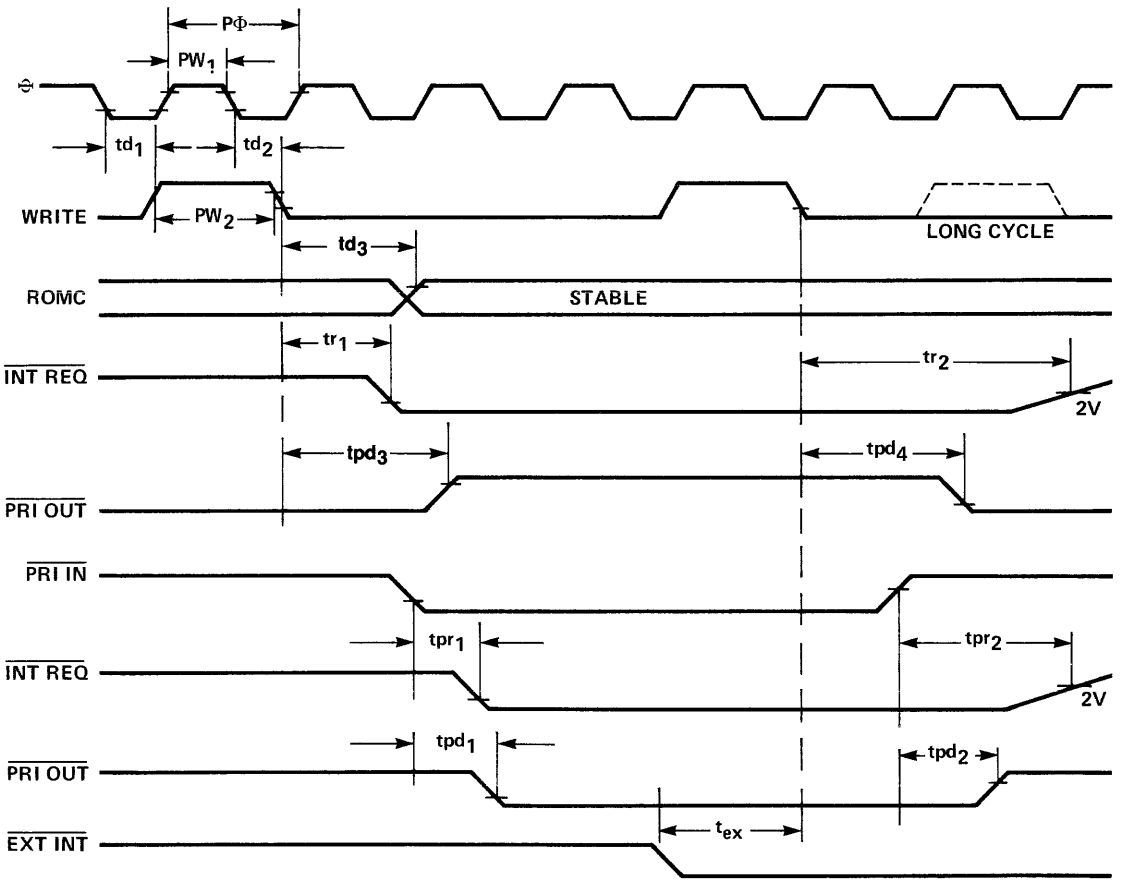
During the interrupt acknowledge, the interrupting PSU provides a 16-bit interrupt address vector. The CPU causes this vector to be loaded into PC0, so that program execution can branch to the routine that handles this particular interrupt. Fifteen bits of the interrupt vector are specified as a mask option. Bit 7 cannot be masked; it is set by the interrupt control logic to 0 if the timer interrupt is enabled or to a 1 if external interrupt is enabled. So that the interrupt vector looks like:

WWWW, XXXX, 0YYY, ZZZZ for timer interrupt
and WWWW, XXXX, 1YYY, ZZZZ for external interrupt

where W, X, Y and Z are the mask specified bits.

3.6.4 Interrupt Signals Timing

Timing for signals associated with the 3851 interrupt logic is shown in Figure 3-13.



NOTE: TIMING MEASUREMENTS ARE MADE AT VALID LOGIC LEVEL OF THE SIGNALS REFERENCED UNLESS OTHERWISE NOTED.

SYMBOLS ARE DEFINED IN TABLE 3-3

Figure 3-13. Interrupt Logic Signals' Timing

4.0 The 3852 Dynamic Memory Interface (DMI)

THE 3852 DYNAMIC MEMORY INTERFACE (DMI)

The 3852 DMI provides all interface logic needed to include up to 64K bytes of dynamic or static RAM memory in an F8 microcomputer system. In response to control signals output by the 3850 CPU, the 3852 DMI generates address and control signals needed by standard static and dynamic RAM devices.

The 3852 DMI, like all other F8 memory devices, contains its own memory address generation logic.

In addition to providing dynamic memory interface logic, the 3852 DMI automatically refreshes dynamic RAM; also, interface logic to support a 3854 Direct Memory Access controller is provided.

+5V and +12V power supplies are required. The 3852 DMI is manufactured using N-channel, Isoplanar MOS technology, therefore power dissipation is very low, typically less than 335 mW.

The 3852 DMI is functionally illustrated in Figure 4-1; the figure shows logic functions, registers, data paths and device pins (with signal names); control signals within the DMI are not shown.

4.1 DEVICE ORGANIZATION

Because of the many similarities between memory addressing logic, as implemented on the 3851 PSU and the 3852 DMI, this section should be read after reading Section 3.1.

4.1.1 Dynamic Memory Interface

These are the principal differences between 3852 DMI and 3851 PSU memory interface logic:

1. The 3852 DMI has no memory on the device itself. (The 3851 PSU includes 1024 bytes of ROM.)

2. The address space allocated to the 3852 DMI must be determined by logic external to the 3852 DMI. (The 3851 PSU address space is determined by a six bit, address selected mask option that is part of the device.) Details are given in Section 4.3.

3. The 3852 has a second Data Counter (DC1). DC1 may be used as a record counter.

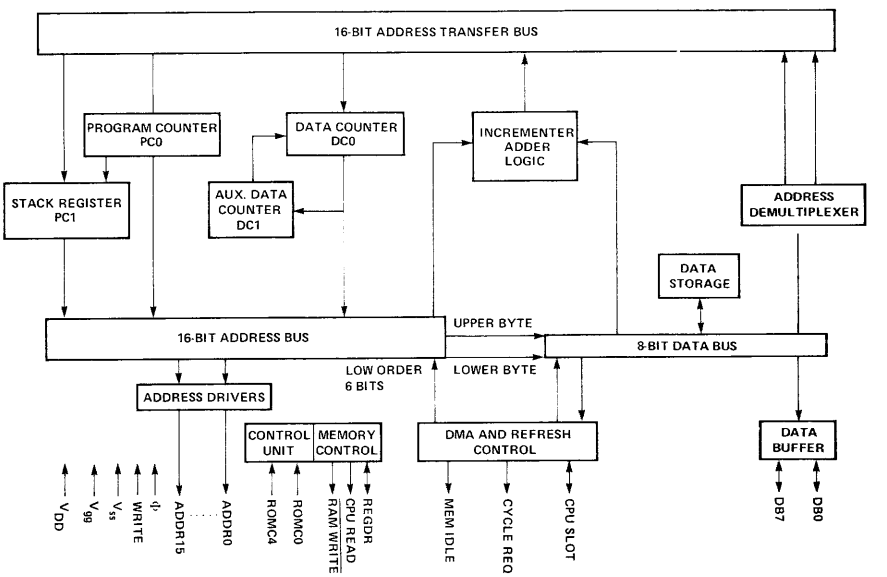


Figure 4-1. Logic Organization and Pins for the 3852 DMI

For descriptions of 3852 DMI logic which is identical to 3851 PSU logic, see Section 3 as follows:

The program counter (PC0) Section 3.1.2.
The data counter (DC0) Section 3.1.2.
The stack register (PC1) Section 3.1.5.
Incrementer adder logic Section 3.1.6.

The auxiliary data counter (DC1) is a back-up for DC0. The XDC instruction (ROMC state 1D) causes the contents of DC0 and DC1 counters to be exchanged.

The Control Unit and Memory Control logic, together control events within the 3852 DMI, and within memory controlled by the 3852 DMI.

The Control Unit decodes ROMC states and enables logic sequences within the 3852 DMI, as described for the 3851 PSU.

Memory Control logic generates appropriate timing and control signals needed by RAM to input or output data. Timing and control signals are generated in response to ROMC states, as decoded by the Control Unit.

The data bus DB0-DB7 is used to transfer data between the 3850 CPU and address registers within the 3852 DMI. Data being transferred between RAM and the 3850 CPU does not pass through the 3852 DMI, and does not use 3852 DMI data bus pins. RAM will have its own connection to the F8 data bus.

4.1.2 DMA and Refresh Control

Because of the organization of the F8 microcomputer system, there is a period within every instruction execution cycle when the CPU is not accessing memory.

DMA and Refresh Control logic generates timing and control signals that identify time periods when the CPU is not accessing memory; during these time periods memory is refreshed, or DMA data accesses occur, as described in Section 4.6.

4.1.3 I/O Ports

The 3852 DMI has four I/O port addresses reserved for its use. There are two versions of the 3852 DMI; one has I/O port addresses 0C, 0D, 0E and 0F for its four I/O ports; since these addresses are also used by the 3853 SMI, another version of the 3852 DMI uses I/O port address EC, ED, EE and EF. This allows an F8 microcomputer system to include both a 3852 DMI and a 3853 SMI.

I/O port addresses 0E and 0F (or EE and EF), though reserved for the 3852 DMI, are not used. Port 0C (or EC) is a general purpose, 8-bit data storage buffer which can be loaded with the OUT or OUTS instruction and read using the IN or INS instruction. Port 0D (or ED) is a control register which controls memory refresh and DMA operations, as described in Section 4.5.5.

4.2 SIGNAL DESCRIPTIONS AND ELECTRICAL CHARACTERISTICS

Figure 4-2 illustrates the 3852 DMI device pins. Signal names correspond to those given in Figure 4-1 and are summarized in Table 4-1.

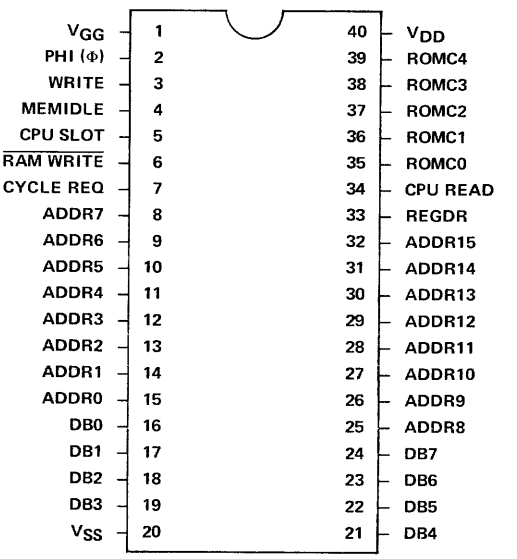


Figure 4-2. 3852 DMI Pin Assignments

4.2.1 Signal Descriptions

Individual signals are described next. Signal characteristics are given in Table 4-2.

Φ and WRITE are the clock outputs from the 3850 CPU.

ROMC0 through ROMC4 are the control signals output by the 3850 CPU.

Table 4-1. 3852 DMI Signal Summary

PIN NAME	DESCRIPTION	TYPE
DB0-DB7	Data Bus Lines	Bi-directional (3-State)
ADDR0-ADDR15	Address Lines	Output (3-State)
Φ, WRITE	Clock Lines	Input
MEMIDLE	DMA Timing Line	Output
CYCLE REQ	RAM Timing Line	Output
CPU SLOT	Timing Line	Input/Output
CPU READ	RAM Timing Line	Output
REGDR	Register Drive Line	Input/Output
$\overline{\text{RAM WRITE}}$	Write Line	Output (3-State)
ROMC0-ROMC4	Control Lines	Input
V_{SS}, V_{DD}, V_{GG}	Power Lines	Input

DB0 through DB7 are the bi-directional data bus lines which link the 3852 DMI with all other devices in the F8 system. Only data moving to or from 3852 DMI address registers and I/O ports use the 3852 DMI DB0-DB7 pins.

ADDR0 through ADDR15 are 16 address lines via which an address is transmitted to dynamic RAM. The address may come from the PC0 or DC0 registers.

CYCLE REQ. There may be either two or three memory access periods within one instruction cycle. CYCLE REQ identifies each memory access period by making a high to low transition at the start of the memory access period. CYCLE REQ does not identify events which are to occur during the memory access period. CYCL REQ is a divide-by-2 of Φ during all ROMC states except ROMC state 05 (store in memory); it can be used to generate the clock signals required by many dynamic RAMs.

MEM IDLE high identifies portions of an instruction execution cycle during which the F8 system is not accessing memory, to read, write or refresh. MEM IDLE high therefore identifies the portion of an instruction cycle which is available for DMA operations. The 3852 DMI can inhibit DMA by holding MEM IDLE constantly low. The address drivers and $\overline{\text{RAM WRITE}}$ driver are always in a high impedance state when MEM IDLE is high, so that a DMA device may drive the address lines at this time.

CPU SLOT high identifies portions of an instruction execution cycle during which the 3850 CPU is reading data out of RAM, or writing data into RAM. CPU SLOT is a bi-directional signal. If held low by external logic, it causes the address line drivers and $\overline{\text{RAM WRITE}}$ driver to be held in a high impedance state.

$\overline{\text{RAM WRITE}}$. When low, this signal specifies that data is to be written into a RAM location. When high, this signal is off; that is, $\overline{\text{RAM WRITE}}$ high does not necessarily specify a read operation.

CPU READ. When high, this signal specifies that data is to be read out of a RAM location. When low, this signal is off; that is, CPU READ low does not specify a write operation; that is done by $\overline{\text{RAM WRITE}}$ low.

REGDR. This signal functions both as an input and an output. As an input, it can be clamped low by an external open collector gate. This prevents the 3852 DMI from placing a byte out of its PC1 or DC0 registers onto the data bus. The DMI internally supplies a pull-up resistor. The signal, functioning as an output, can control data bus buffers. The DMI will internally clamp REGDR low except during those ROMC states during which the DMI is required to place a byte out of PC1 or DC0 registers or either of its two control registers (I/O ports) onto the data bus. Figure 4-3 shows the REGDR internal logic.

4.2.2 DC Electrical Specifications

Absolute Maximum Ratings (Above which useful life may be impaired).

V_{GG}	+15V to -0.3V
V_{DD}	+7V to -0.3V
All other inputs & outputs	+7V to -0.3V
Storage Temperature	−55°C to +150°C
Operating Temperature	0°C to +70°C

Note: All voltages with respect to V_{SS}.

DC Characteristics: V_{SS} = 0V, V_{DD} = +5V ± 5%, V_{GG} = +12V ± 5%, T_A = 0°C to +70°C

SUPPLY CURRENTS

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
I_{DD}	V_{DD} Current		35	70	mA	f = 2 MHz, Outputs unloaded
I_{GG}	V_{GG} Current		13	30	mA	f = 2 MHz, Outputs unloaded

4.3 THE 3852 DMI ADDRESS SPACE

Section 3.1.3 describes the concept of memory device address space, and explains how a 3851 PSU

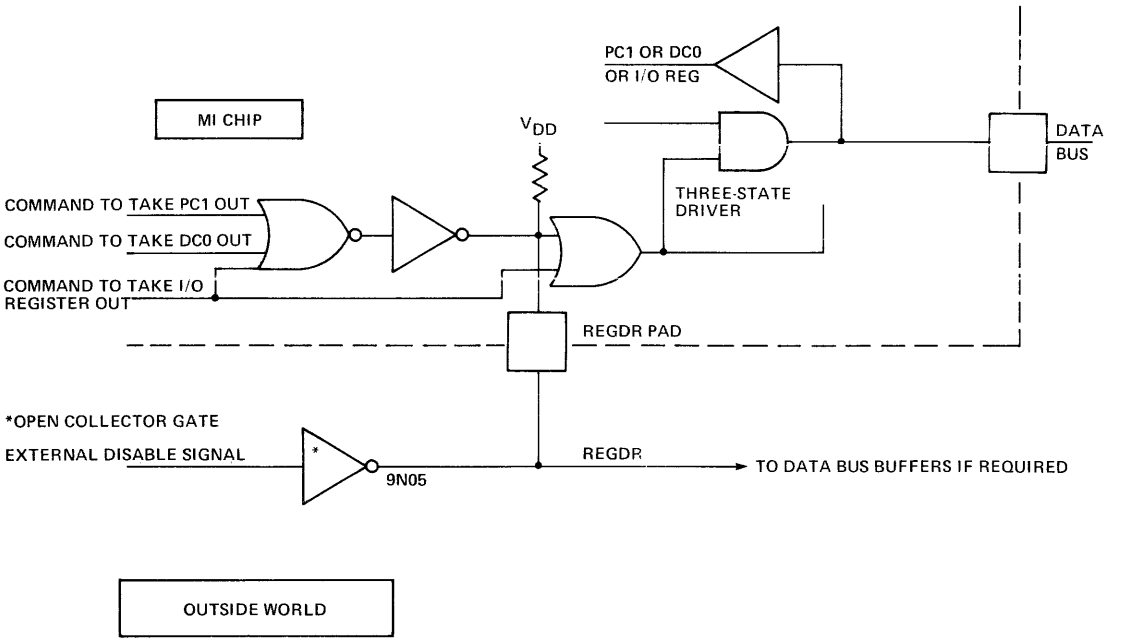


Figure 4-3. REGDR Controls Data Bus Drivers

device's address space is defined. Section 3.1.4 discusses the question of address contentions that can arise when each memory device contains duplicated memory address registers. The way in which address space is defined for a 3852 DMI is very different from the way in which it is defined for a 3851 PSU; there are also a few small differences in address contention resolution.

The 3852 DMI has two separate and distinct address spaces:

1. The attached dynamic RAM address space.
2. The DMI address registers' address space.

4.3.1 Dynamic RAM Address Space

Any dynamic RAM which is controlled by the 3852 DMI will have a PAGE SELECT input, which must be true if the memory is to respond to read or write control signal sequences.

PAGE SELECT true is created by logic external to the 3852 DMI, and defines the dynamic RAM address space. PAGE SELECT true can be generated in any way; there are no special rules.

For example, consider an F8 system with 1K bytes of ROM on a 3851 PSU and 4K bytes of dynamic RAM controlled by a 3852 DMI; address ranges will be as follows:

1K bytes of ROM	0000_{16} to $03FF_{16}$
4K bytes of RAM	0400_{16} to $13FF_{16}$

In binary format, the dynamic RAM address space is defined by:

	ADDR15			ADDR12	ADDR11	ADDR10										ADDR0
Minimum:	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Maximum:	0	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1

PAGE SELECT may be the OR of ADDR12, ADDR11 and ADDR10, which are shaded above.

Depending on the way in which dynamic RAM is being used, PAGE SELECT may be a simple memory enable signal, or it may be ANDed with CPU READ and $\overline{\text{RAM WRITE}}$, to generate local versions of these two signals which are locally true only.

Table 4-2. Summary of 3852 DMI Signal Characteristics

SIGNAL	SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
DATA BUS (DB0-DB7)	V_{IH}	Input High Voltage	2.9	V_{DD}	Volts	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = -100 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1.6 mA
	I_{IH}	Input High Current		3	μA	$V_{IN} = V_{DD}$, 3-State mode
	I_{IL}	Input Low Current		-3	μA	$V_{IN} = V_{SS}$, 3-State mode
ADDRESS LINES (ADDR0-ADDR15) AND $\overline{\text{RAM WRITE}}$	V_{OH}	Output High Voltage	4.0	V_{DD}	Volts	I_{OH} = -1 mA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 3.2 mA
	I_L	Leakage Current		3	μA	$V_{IN} = V_{DD}$, 3-State mode
	I_L	Leakage Current		-3	μA	$V_{IN} = V_{SS}$, 3-State mode
CLOCK (Φ, WRITE)	V_{IH}	Input High Voltage	4.0	V_{DD}	Volts	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_L	Leakage Current		3	μA	$V_{IN} = V_{DD}$
MEMIDLE, CYCLE REQ, CPU READ	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = -1 mA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 2 mA
CONTROL LINES (ROMC0-ROMC4)	V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_L	Leakage Current		3	μA	V_{IN} = 6V
REGDR, CPU SLOT	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = -300 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 2 mA
	V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	Internal Pull-up
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_{IL}	Input Low Current (REGDR)	-3.5	-14.0	mA	V_{IN} = 0.4V & Device outputting a logic "1"
	I_L	Leakage Current		3	μA	V_{IN} = 6V

Note: Positive current is defined as conventional current flowing into the pin referenced.

4.3.2 3852 DMI Address Registers' Address Space

As described in Table 2-5, certain ROMC states require the contents of the high order, or low order half of PC0, PC1 or DC0 to be placed on the data bus. If there is more than one memory device in an F8 system, only one device must respond to these ROMC states.

The 3851 PSU uses its address select mask to determine if it is to place address register contents on the data bus; for the 3851 PSU, therefore, the memory and address registers' address spaces must be identical.

The 3852 DMI address registers' address space is identified by the REGDR signal; if this signal is not clamped low, the 3852 will place data on the data bus in response to ROMC states that require data from PC0, PC1 or DC0 to be placed on the data bus. If REGDR is derived from the PAGE SELECT signal described in Section 4.3.1, then the RAM memory and the 3852 DMI address registers' address spaces will coincide.

On the other hand, it is a good idea to make the 3852 DMI address registers' address space cover all addresses that are not part of another memory device's address registers' address space. For

example, returning to the illustration in Section 4.3.1, the following address spaces would be desirable:

	ADDRESS SPACES	
	MEMORY	ADDRESS REGISTERS
3851 PSU	0000_{16}-$03FF_{16}$	0000_{16}-$03FF_{16}$*
3852 DMI	0400_{16}-$13FF_{16}$	0400_{16}-$FFFF_{16}$

*For the 3851 PSU, the two address spaces must be identical.

If the address space for the address registers covers all possible memory addresses, then instructions that read data out of address registers will always generate a valid response.

In the above illustration, if memory and address registers' address spaces coincided for the 3852, then in response to instructions that require data to be output from PC0, PC1 or DC0, no device would respond when the selected address register contains a value in excess of $13FF_{16}$; as a result, invalid values would be received by the 3850 CPU.

4.3.3 Address Contentions

When a 3852 DMI is present in an F8 system, memory addressing contentions are resolved as described in Section 3.1.4, with one exception: the 3852 DMI has a DC1 register and the 3851 PSU does not.

The XDC instruction (ROMC state 1D) causes the contents of the DC0 and DC1 registers to be exchanged; having no DC1 register, the 3851 PSU does not respond to this ROMC state, therefore 3851 PSU and 3852 DMI devices can have different values in their DC0 registers, and each value can be within the different address spaces of the two memory devices. An instruction that requires data to be output from DC0 may now cause two devices to simultaneously place different data on the data bus. This may be illustrated as follows:

	PSU	DMI
Before XDC:	DC0 = XXXX	XXXX
	DC1 =	YYYY
After XDC:	DC0 = XXXX	YYYY
	DC1 =	XXXX

If XXXX happens to be in a PSU's address space while YYYY is in the DMI address space, then address contentions will arise.

Even if XXXX is not in the PSU's address space, address contentions may arise due to the fact that memory reference instructions will increment different DC0 contents. Suppose two memory reference instructions are executed following one XDC, then another XDC is executed; this is what happens:

	PSU	DMI
After first XDC:	DC0 = XXXX	YYYY
	DC1 =	XXXX
After two memory	DC0 = XXXX+2	YYYY+2
reference instructions:	DC1 =	XXXX
After second XDC:	DC0 = XXXX+2	XXXX
	DC1 =	YYYY+2

An address contention may arise if DC0 contents approaches the boundary of the PSU address space. For example, if the address space boundary occurs at XXXX+1, the PSU and the DMI will both consider themselves selected.

The following coding instruction sequence shows how to use the XDC instruction without encountering address contentions. The example allows use of a second address value, YYYY, which is held in DC1, while using the H register to temporarily hold the first address value, XXXX. Address YYYY, which at the beginning of the example is held in DC1, must be in the DMI address space. The address XXXX may be in any address space.

INSTRUCTION	PSU	DMI	
	DC0	DC0	DC1
	XXXX	XXXX	YYYY
LR H,DC			
DCI ZZZZ	ZZZZ	ZZZZ	YYYY
XDC	ZZZZ	YYYY	ZZZZ
–			
–			
–			
Other Instructions			
–			
–			
–	ZZZZ+N	YYYY+N	ZZZZ
XDC	ZZZZ+N	ZZZZ	YYYY+N
LR DC,H	XXXX	XXXX	YYYY+N

For the above scheme to work, it is only necessary for ZZZZ through ZZZZ+N to be outside any PSU's address space.

If the value XXXX through XXXX+N is outside of any PSU's address space, then the DCI ZZZZ instruction may be omitted.

In many cases, it will not be necessary to restore the XXXX value; then the LR H,DC and LR DC,H instructions can also be omitted—letting a subsequent DC0 loading instruction synchronize the DC0's.

Before a value held in DC1 can be used, it must first have been loaded into DC1. The XDC instruction is used to load DC1. Consider the following instruction sequence:

INSTRUCTION		PSU DC0	DMI DC0	DC1
		XXXX	XXXX	WWWW
DCI	YYYY	YYYY	YYYY	
XDC		YYYY	WWWW	YYYY
DCI	ZZZZ	ZZZZ	ZZZZ	YYYY

YYYY lies in the address space of the DMI, ZZZZ lies anywhere, XXXX and WWWW are arbitrary initial values. The DCI instructions could just as well be LR DC,H or LR DC,Q.

The exchange of DC0 and DC1 becomes most powerful when a series of swaps are used to add two blocks of memory, or to move data from one block to a second. The XDC instruction can be used to do this so long as neither block is in a PSU's address space. Notice that the DC0 of the PSU is out of step throughout the example.

INSTRUCTION		PSU PC0	DMI DC0	DC1
		XXXX	XXXX	YYYY
DCI	ZZZZ	ZZZZ	ZZZZ	YYYY
LM		ZZZZ+1	ZZZZ+1	YYYY
XDC		ZZZZ+1	YYYY	ZZZZ+1
ST		ZZZZ+2	YYYY+1	ZZZZ+1
XDC		ZZZZ+2	ZZZZ+1	YYYY+1
Other Instructions				
LM		ZZZZ+ΔZ+ΔY-1	ZZZZ+ΔZ	YYYY+ΔY
XDC		ZZZZ+ΔZ+ΔY-1	YYYY+ΔY-1	ZZZZ+ΔZ
ST		ZZZZ+ΔZ+ΔY	YYYY+ΔY	ZZZZ+ΔZ
DCI	WWWW	WWWW	WWWW	ZZZZ+ΔZ

In the above example ZZZZ and YYYY both lie in the address of a DMI. The space spanned by ZZZZ to ZZZZ+ΔZ+ΔY must be outside of any PSU's address space.

4.4 TIMING

The 3852 DMI receives timing signals from the 3850 CPU, then outputs timing signals used by dynamic RAM, and a 3854 DMA device, if present.

4.4.1 Timing Signals Received by the 3852 DMI

The Φ and WRITE signals, generated by the 3850 CPU and described in Section 2.3, are the timing inputs to the 3852 DMI. Figure 2-9 provides timing for Φ, WRITE, and the ROMC state signals output by the 3850 CPU.

4.4.2 Timing Signals Output by a 3852 DMI

Figure 4-4 along with Table 4-3 summarizes the timing for signals output by a 3852 DMI.

Within an instruction cycle, there may be either two or three memory access periods, depending on whether the instruction cycle is long or short. A memory access period is equivalent to two Φ clock periods, and is identified by CYCLE REQ, which is a divide-by-two of Φ. Whether the instruction cycle is short, or long, depends on the source and destination of the data being transmitted during instruction execution.

During the first memory access period, the 3852 DMI outputs the contents of PC0 on the address lines of ADDR0-ADDR15.

In effect, 3852 DMI logic begins by assuming that a memory read is to occur, with the memory address provided by PC0.

While the contents of PC0 are being output on the address lines, the 3852 DMI control unit, in parallel, decodes the ROMC state which has been received from the 3850 CPU.

If the assumed logic proves to be correct, or if no memory access is to occur, then the second access period can be used for memory refresh or DMA.

If the instruction, once decoded by the CPU, specifies a memory read with another memory address, then the 3852 DMI wastes the first access period. The instruction cycle will always be long in this case. During the second access period, the required memory access is performed, while memory refresh occurs, or DMA is implemented in the third access period.

If a memory write instruction is decoded, then no access periods are available for memory refresh or DMA.

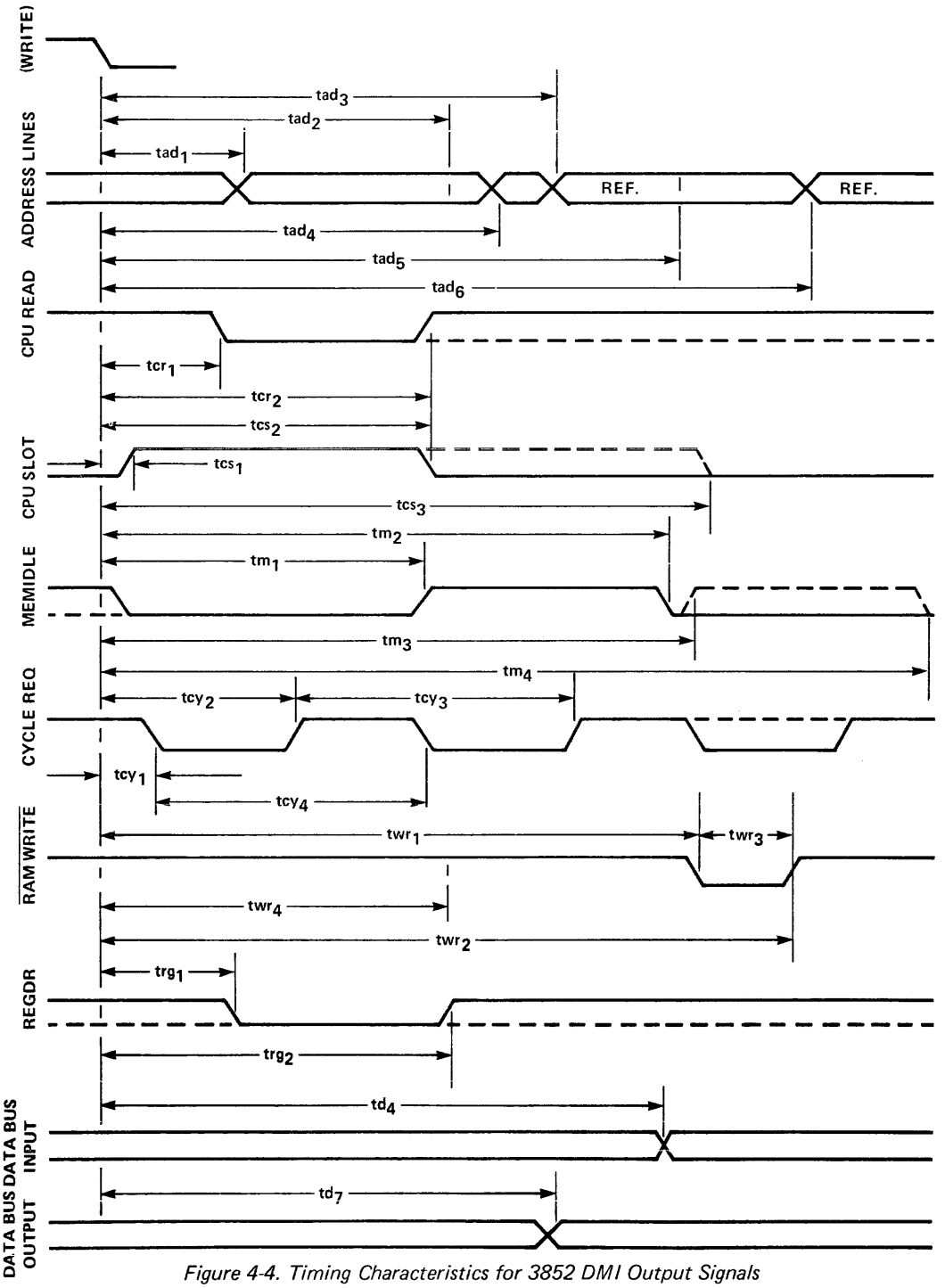


Figure 4-4. Timing Characteristics for 3852 DMI Output Signals

Table 4-3. 3852 DMI Output Signals Timing Summary

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNITS	NOTES
PΦ	Φ clock period	0.5		10	μS	Fig. 2-9
td_2	Φ to WRITE - Delay			250	nS	
tad_1	Address delay if PC0	50	300	500	nS	3
tad_2	Address delay to high Z (short cycle with DMA on)	tcs_2+50		tcs_2+200	nS	3
tad_3	Address delay to refresh (short cycle with REF on)	tcs_2+50		tcs_2+400	nS	3
tad_4	Address delay if DC	2PΦ+50-td_2		2PΦ+400-td_2	nS	3
tad_5	Address delay to high Z (long cycle with DMA on)	tcs_3+50		tcs_3+200	nS	3
tad_6	Address delay to refresh (long cycle with REF on)	tcs_3+50		tcs_3+400	nS	3
tcr_1	CPU READ - Delay	50	250	450	nS	1
tcr_2	CPU READ + Delay	2PΦ+50-td_2		2PΦ+400-td_2	nS	1
tcs_1	CPU SLOT + Delay	80-td_2		320-td_2	nS	1
tcs_2	CPU SLOT - Delay (PC0 access)	2PΦ+60-td_2		2PΦ+420-td_2	nS	1
tcs_3	CPU SLOT - Delay (DC access)	4PΦ+60-td_2		2PΦ+420-td_2	nS	1
tm_1	MEMIDLE + Delay (PC0 access)	2PΦ+50-td_2		4PΦ+400-td_2	nS	1
tm_2	MEMIDLE - Delay (PC0 access)	4PΦ+50-td_2		4PΦ+350-td_2	nS	1
tm_3	MEMIDLE + Delay (DC access)	4PΦ+50-td_2		4PΦ+400-td_2	nS	1
tm_4	MEMIDLE - Delay (DC access)	6PΦ+50-td_2		6PΦ+350-td_2	nS	1
tcy_1	WRITE to CYCLE REQ - Delay	80-td_2		400-td_2	nS	1, 4
tcy_2	WRITE to CYCLE REQ + Delay	PΦ+80-td_2		PΦ+400-td_2	nS	1, 4
tcy_3	CYCLE REQ + to + Edge Delay		2PΦ			1, 4
tcy_4	CYCLE REQ - to - Edge Delay		2PΦ			1, 4
twr_1	$\overline{\text{RAM WRITE}}$ - Delay	4PΦ+50-td_2		4PΦ+450-td_2	nS	3
twr_2	$\overline{\text{RAM WRITE}}$ + Delay	5PΦ+50-td_2		5PΦ+300-td_2	nS	3
twr_3	$\overline{\text{RAM WRITE}}$ Pulse Width	350		PΦ	nS	3
twr_4	$\overline{\text{RAM WRITE}}$ to High Z Delay	tcs_2+40		tcs_2+200	nS	3
trg_1	REGDR - Delay	70	300	500	nS	1
trg_2	REGDR + Delay	2PΦ+80-td_2		2PΦ+500-td_2	nS	1
td_4	WRITE to Data Bus Input Delay			2PΦ+1000	nS	
td_7	WRITE to Data Bus Output Delay	2PΦ+100-td_2		2PΦ+850-td_2		2

Notes:

1. C_L = 50 pf.

2. C_L = 100 pf.

3. C_L = 500 pf.

4. CYCLE REQ is a divide-by-2 of Φ for all instructions except the STORE instruction.

5. On a given chip, the timing for all signals will tend to track. For example, if CPU SLOT for a particular chip is fairly slow and its timing falls out near the MAX delay value specified, then the timing for all signals on that chip will tend to be out near the MAX delay values. Likewise for a fast chip whose signals fall near the MIN values. This is a result of the fact that processing parameters (which affect device speed) are quite uniform over small physical areas on the surface of a wafer.

6. Input and output capacitance is 3 to 5 pf typical on all pins except V_{DD}, V_{GG}, and V_{SS}.

Four variations of the instruction cycle result. The timing diagrams illustrating the four variations represent worst cases, and assume td_2 = 150 nS. These are the four variations:

1. The instruction fetch. The memory address originates in PC0, and the instruction cycle is short. Timing is shown in Figure 4-5.

2. An immediate operand fetch. The memory address originates in PC0, and the instruction cycle is long. Timing is shown in Figure 4-6.

3. A data fetch. A data byte is output from an address register, or the memory address originates in DC0, therefore the instruction cycle is long. Timing is shown in Figure 4-7.

4. A memory write. Data is written into the RAM memory location addressed by DC0. Timing is shown in Figure 4-8.

CPU SLOT and MEM IDLE identify the way in which a memory access period is being used. Figure 4-9 illustrates the relationship.

When the 3850 CPU is accessing memory, CPU SLOT is high; RAM WRITE and the address lines are driven at this time.

When memory is available for DMA access, CPU SLOT is low, and MEM IDLE is high.

When the 3852 DMI is refreshing dynamic memory, CPU SLOT and MEM IDLE are both low.

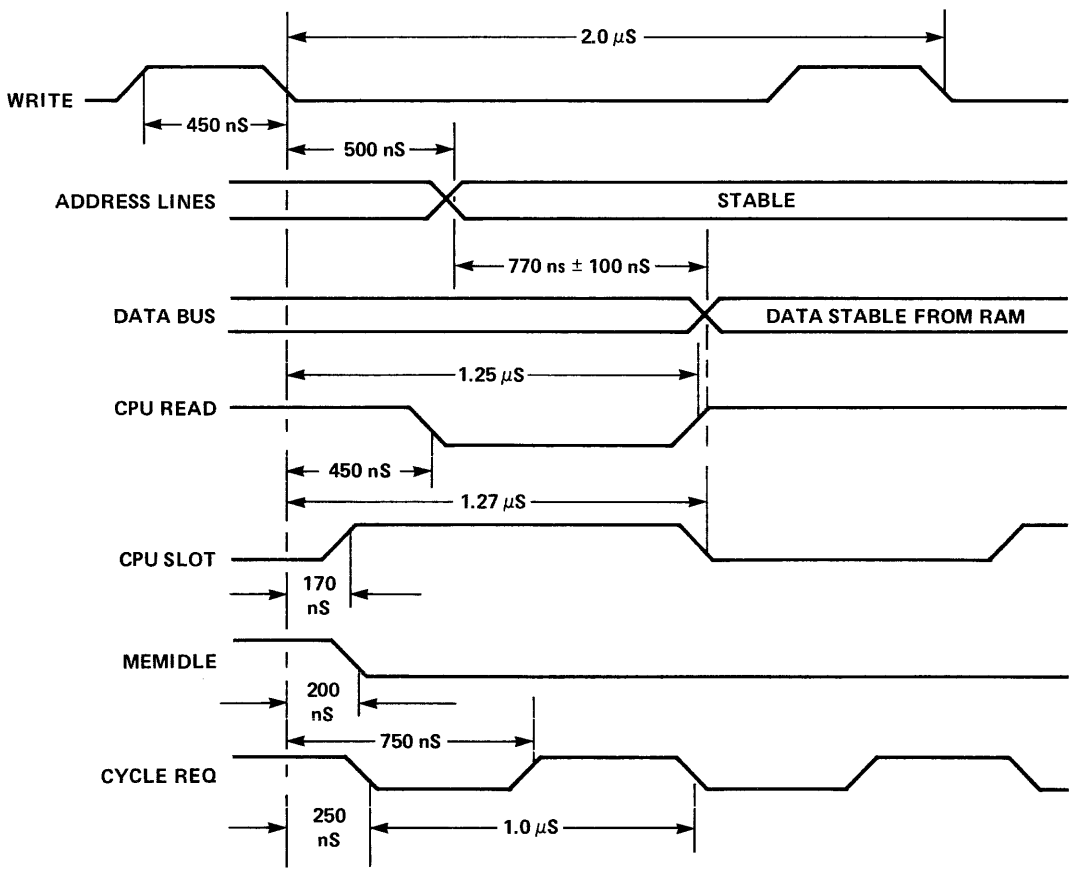


Figure 4-5. 3852 DMI Timing Signals Output during a Short Cycle Memory Read with Address from PC0

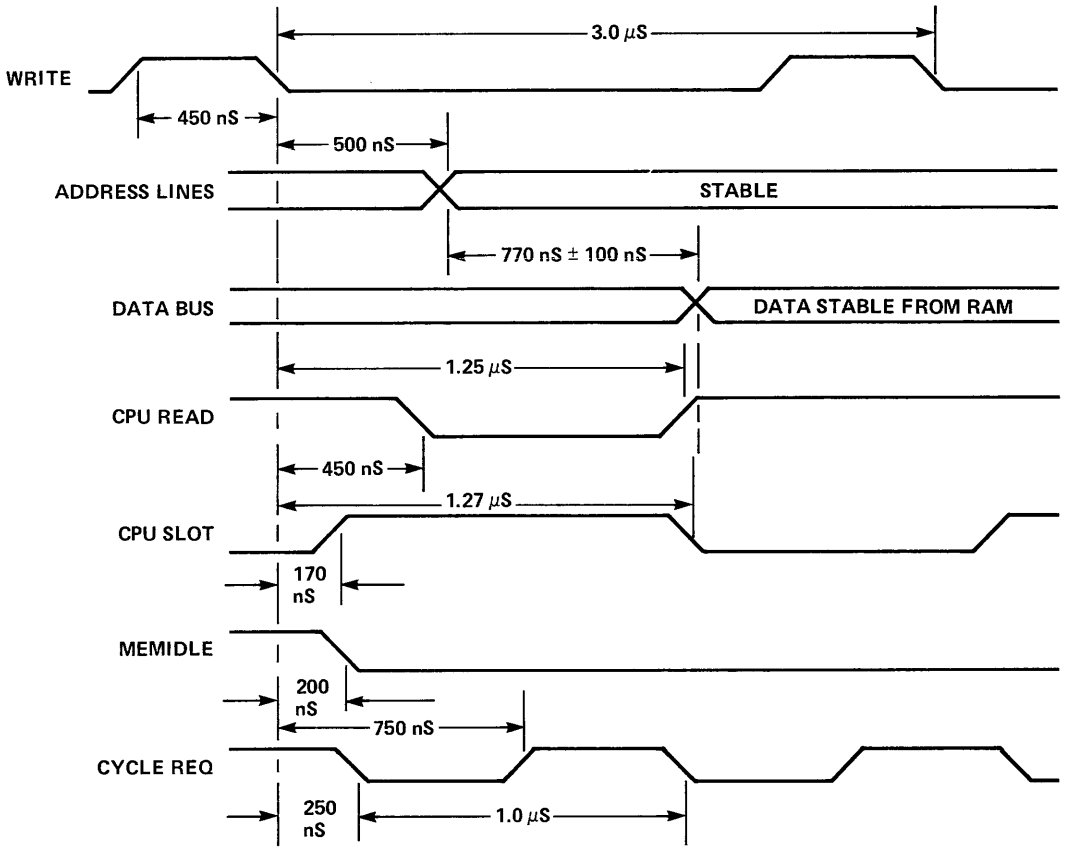


Figure 4-6. 3852 DMI Timing Signals Output during a Long Cycle Memory Read, with Address Out of Program Counter

3852 DMI logic is able to achieve two memory accesses within one instruction cycle by pursuing the logic sequence summarized in Table 4-4. Buffer/latches are placed on the F8 data bus lines between the RAM and the F8 system to hold the data fetched during the first access.

4.5 INSTRUCTION EXECUTION

Like other memory devices of the F8 family, the 3852 DMI responds to signals which are output by the 3850 CPU in the course of implementing instruction cycles.

Figure 4-9 and Table 4-6 summarize states for the three data transfer control signals (CPU READ, $\overline{\text{RAM WRITE}}$, REGDR), the contents of the data bus, and the source of memory addresses, for every ROMC state described in Table 2-5.

4.5.1 Data Output by RAM

Figures 4-5, 4-6 and 4-7 provide worst case timing when RAM, controlled by the 3852 DMI, outputs data onto the data bus. In these figures it is assumed that CPU SLOT is used to strobe the RAM data into the data bus latches. Refresh timing is not shown here (see Section 4.6).

CPU READ is output high by the 3852 DMI to enable transfer of data from the data bus buffers to the data bus. Recall that dynamic RAM have its own connection to the data bus via buffer/latches; data is not transferred via the 3852 DMI.

Observe that CPU READ high is similar to $\overline{\text{DBDR}}$ low—each is active when its respective data bus drivers are turned on.

Table 4-4. How the Memory Access Periods of an Instruction Cycle are Used

OPERATION PERFORMED DURING INSTRUCTION CYCLE	FIRST ACCESS	SECOND ACCESS	THIRD ACCESS
No memory access, or read from memory addressed by PC0. (See Figure 4-5.)	[PC0] → A0 - A15	Latch data on F8 data bus. Second memory access for DMA or refresh.	None
No memory access, or read from memory addressed by PC0. (See Figure 4-6.)	[PC0] → A0 - A15	Latch data on F8 data bus. Second memory access for DMA or refresh.	Third memory access not used.
Read data from memory addressed by register other than PC0, or read data from address register. (See Figure 4-7.)	[PC0] → A0 - A15	[Other register] → A0 - 15	Latch data on F8 data bus. Third memory access for DMA or refresh
Write data to memory. (See Figure 4-8.)	[PC0] → A0 - A15	[DC0] → A0 - A15	Access memory to write data. No DMA or refresh.
[ ] means "contents of register identified within square brackets.			

4.5.2 Data Output by the 3852 DMI

As described in Section 4.3, REGDR defines the address space of the address registers within the 3852 DMI.

If a ROMC state received by the 3852 DMI requires data to be output from an address register, then the 3852 DMI will become the selected data source if REGDR is allowed to go high.

Timing is identical to that for the 3851 PSU when outputting onto the data bus as indicated in Figure 3-3, td_7.

4.5.3 Data Input to RAM

Figure 4-7 provides worst case timing when data is written into RAM. Figure 2-11, tdb_1 provides timing for data output from the accumulator of the 3850 CPU. Data is transferred through tri-state buffers on the data bus and into RAM. (See Section 8 for system configuration examples.)

$\overline{\text{RAM WRITE}}$ is pulsed low by the 3852 DMI to enable the transfer of data off the data bus, into RAM. The tri-state buffers or multiplexers between data bus and RAM WRITE data lines are necessary if DMA sources are also allowed to write into RAM. See Section 4.3 for a discussion of RAM address space.

4.5.4 Data Input to the 3852 DMI

As described in Section 3.1.4, problems of addressing contention are posed by having duplicated address registers; one step in resolving this possible problem is to force every memory device to read data into its address registers whenever a ROMC state specifies any such operation. Address space concepts therefore do not apply when data is read into 3852 DMI address registers.

4.5.5 Input/Output

There are two versions of the 3852 DMI; each has four reserved I/O port addresses, implemented as follows:

PORT ADDRESSES		FUNCTION
STANDARD	OPTION	
0C	EC	General purpose latch
0D	ED	Memory/DMA control
0E	EE	Not implemented
0F	EF	Not implemented

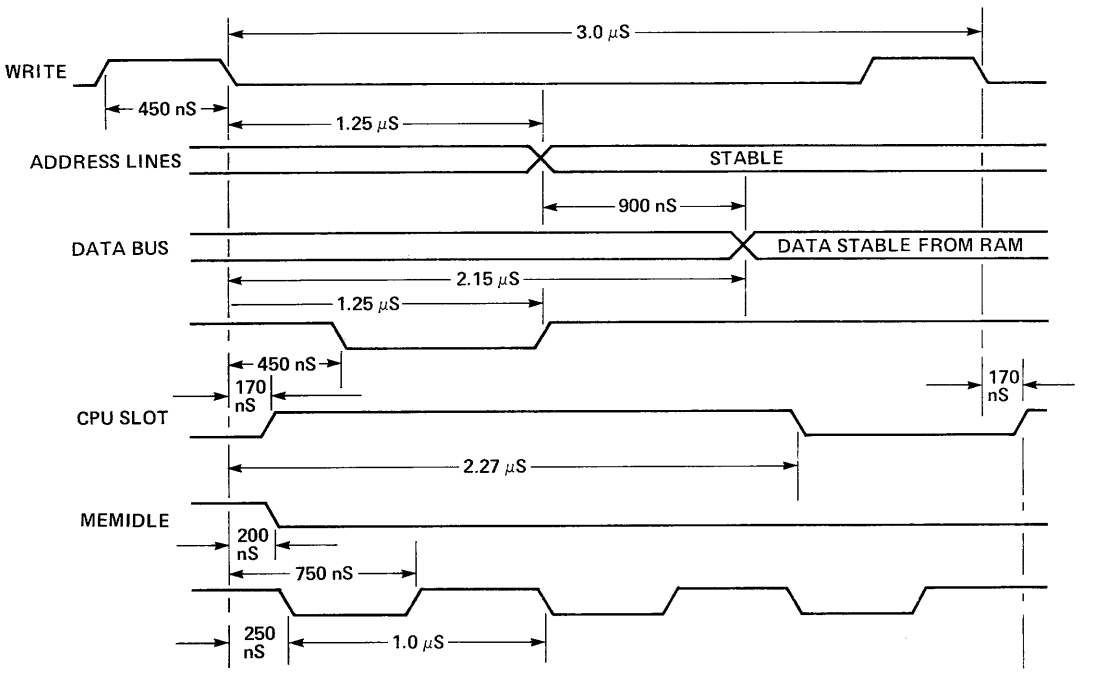


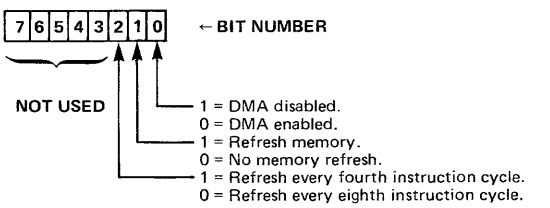
Figure 4-7. 3852 DMI Timing Signals Output during a Long Cycle Memory Read, with Address Out of Data Counter

Option port addresses are used in F8 systems that include both a 3852 DMI and a 3853 SMI; it is differentiated from the Standard by a SL31116 marking.

The implemented I/O ports are accessed via IN, INS, OUT and OUTS instructions, just like any other I/O port. However, the 3852 DMI I/O ports are internal latches, having no connection to I/O pins or external interface. REGDR, if not clamped low by an external device, will go high during IN or INS instructions that select either of the DMI ports. However, clamping REGDR low does not inhibit data bus driving during I/O as it did during the output of address registers.

I/O port 0C (or EC) is used as a general purpose, 8-bit data storage location.

I/O port 0D (or ED) controls memory refresh and DMA as follows:



4.5.6 System Initialization

An F8 system is initialized by power on, or $\overline{\text{EXT RESET}}$ being pulsed low at the CPU.

When an F8 system is initialized, DMA is turned off and memory refresh is on, with refresh every fourth cycle selected.

Contents of all other registers are indeterminate; reading the control port 0D (or ED) also gives indeterminate results, although the DMA/refresh state of the DMI has been initialized.

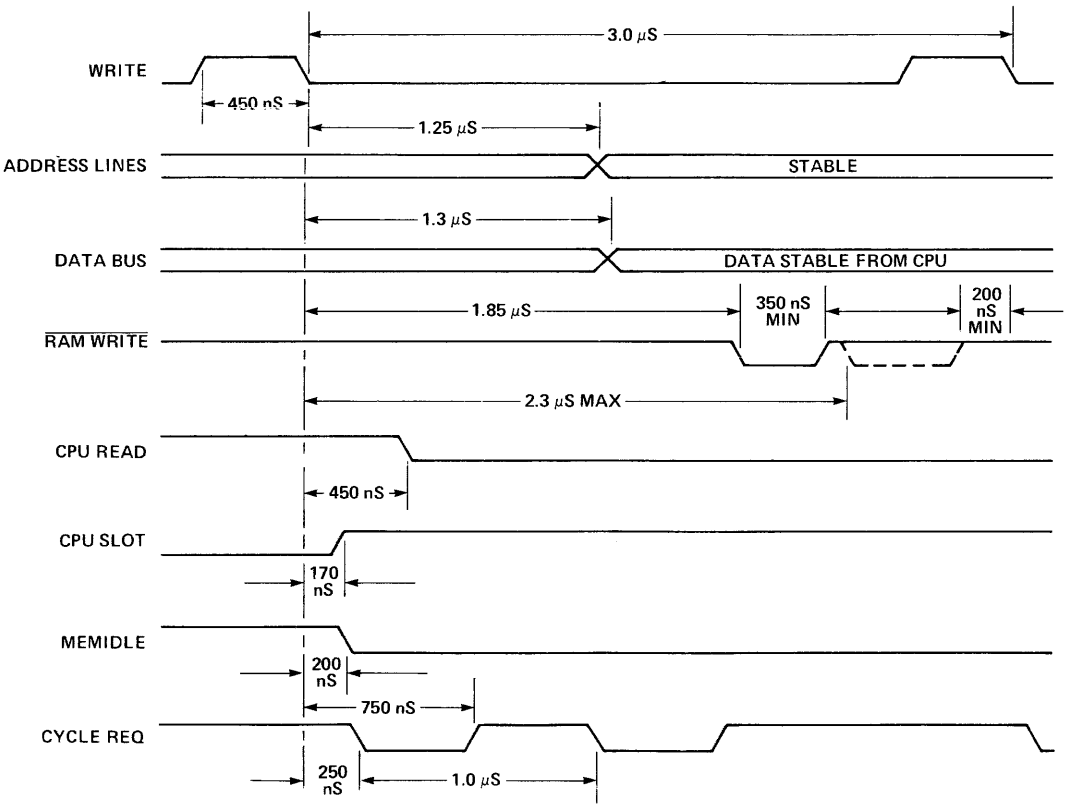


Figure 4-8. 3852 DMI Timing Signals Output during a Write to Memory

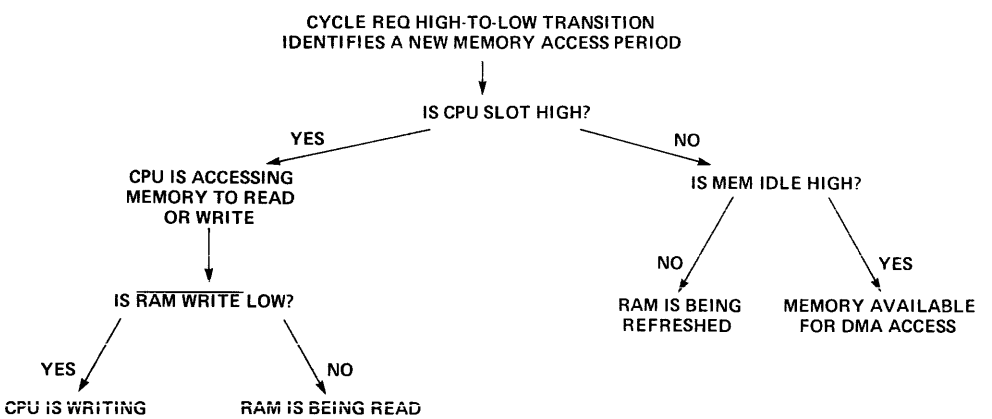
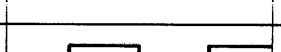
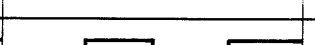
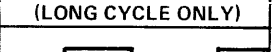
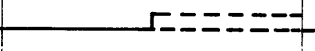
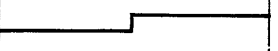


Figure 4-9. Interpretation of Signals Output by the 3852 DMI

Table 4-5. Symbols Used in Table 4-6

SYMBOL	MEANING
[ ]	Contents of register or memory location identified within brackets.
DMA/REF/PC0	If refresh occurs during this access, A0-A5 = Refresh address A6-A15 = Previous [PC0] If no refresh during this access, and DMA operation is disabled, A0-A15 = [PC0] If no refresh during this access, and DMA operation is enabled, A0-A15 is in high impedance state
FL/REF/PC0	If no refresh during second access, and DMA operation is enabled, A0-A15 is in high impedance state, but DMA cannot occur If refresh occurred during the second access, then A0-A15 maintained from previous access during this third access If no refresh during this access, and DMA operation is disabled, A0-A15 = [PC0]
DMA/REF/DC0	As DMA/REF/PC0, but substitute [DC0] for [PC0]
DMA/REF/PC1	As DMA/REF/PC0, but substitute [PC1] for [PC0]
[[ ]]	Contents of memory word addressed by contents of register
→	Source data on left of arrow, destination on right of arrow
[DC0] U	DC0, upper byte
[DC0] L	DC0, lower byte
[PC0] U	PC0, upper byte
[PC0] L	PC0, lower byte
[PC1] U	PC1, upper byte
[PC1] L	PC1, lower byte
IO	In place of PC0 in DMA/REF/PC0, or FL/REF/PC0, means: If Port 0C (or EC) selected, A0-A7 = 1111111 A8-A15 = Port C contents If Port 0D (or ED) selected, A0-A7 = Port D contents A8-A15 = 1111111 If neither Port 0C (or EC) or 0D (or ED) selected, A0-A15 = [PC0]

Table 4-6. 3852 DMI Responses to ROMC States

ROMC STATE	SIGNAL NAME	SIGNAL CONDITON OR INFORMATION CONTAINED		
		FIRST MEMORY ACCESS	SECOND MEMORY ACCESS	THIRD MEMORY ACCESS (LONG CYCLE ONLY)
	Φ			
	WRITE			
	CYCLE REQ			
00*	ADDR	[PC0]	DMA/REF/PC0	
	DATA BUS		INSTRUCTION CODE	
	CPU SLOT	1	0	
	MEM IDLE	0	1	
	CPU READ	0	1	
	REGDR	0	0	
01	ADDR	[PC0]	DMA/REF/PC0	FL/REF/PC0
	DATA BUS		OFFSET FOR BRANCH	
	CPU SLOT	1	0	0
	MEM IDLE	0	1	0
	CPU READ	0	1	1
	REGDR	0	0	0
02	ADDR	[PC0]	[DC0]	DMA/REF/DC0
	DATA BUS		INSTRUCTION OPERAND	
	CPU SLOT	1	1	0
	MEM IDLE	0	1	0
	CPU READ	0	1	1
	REGDR	0	0	0
03**	ADDR	[PC0]	DMA/REF/PC0	FL/REF/PC0
	DATA BUS		INSTRUCTION OPERAND	
	CPU SLOT	1	0	0
	MEM IDLE	0	0	1
	CPU READ	0	1	1
	REGDR	0	0	0
04			SEE 0D	
05	ADDR	[PC0]	[DC0]	[DC0]
	DATA BUS		BYTE FROM CPU, TO BE STORED IN RAM	
	CPU SLOT	1	1	1
	MEM IDLE	0	0	0
	CPU READ	0	0	0
	REGDR	0	0	0

*This is a long cycle for the DS (op code 30) instruction only.

**This instruction is short for BT (op code 8X), BF (op code 9X), and BR7 (op code 8F) if branch not taken, and for DCI (op code 2A) always.

Table 4-6. 3852 DMI Responses to ROMC States (Continued)

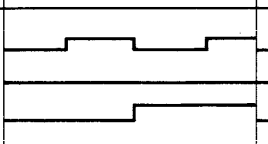
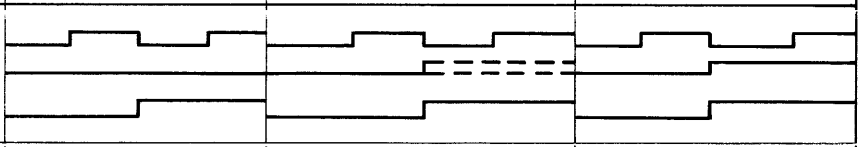
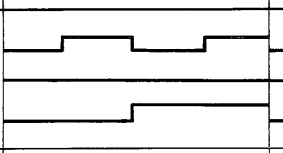
ROMC STATE	SIGNAL NAME	SIGNAL CONDITION OR INFORMATION CONTAINED		
		FIRST MEMORY ACCESS	SECOND MEMORY ACCESS	THIRD MEMORY ACCESS (LONG CYCLE ONLY)
	Φ WRITE CYCLE REQ			
06 07 09 0B	ADDR DATA BUS CPU SLOT MEM IDLE CPU READ REGDR	[PC0] 1 0 0 0	[DC0] FOR 06 & 09, [PC1] FOR 07 & 0B [DC0] u(06), [PC1] u(07), 1 0 0 1	DMA/REF/(DC0 OR PC1) [DC0] L(09), [PC1] L(0B) 0 1 0 1
08	ADDR DATA BUS CPU SLOT MEM IDLE CPU READ REGDR	[PC0] 1 0 0 0	DMA/REF/PC0 0 FROM CPU TO [PC0] 0 1 0 0	FL/REF/PC0 0 0 0 0
0A	ADDR DATA BUS CPU SLOT MEM IDLE CPU READ REGDR	[PC0] 1 0 0 0	DMA/REF/DC0 OFFSET FOR DC0, FROM CPU 0 1 0 0	FL/REF/DC0 0 0 0 0
0B			SEE 06	
0C	ADDR DATA BUS CPU SLOT MEM IDLE CPU READ REGDR	[PC0] 1 0 0 0	DMA/REF/PC0 [[PC0]]→[PC0], LOWER BYTE 0 1 1 0	FL/REF/PC0 0 0 1 0
04 0D 10 1D	ADDR DATA BUS CPU SLOT MEM IDLE CPU READ REGDR	[PC0] 1 0 0 0	DMA/REF/PC0 UNUSED 0 1 0 0	
0E 13	ADDR DATA BUS CPU SLOT MEM IDLE CPU READ REGDR	[PC0] 1 0 0 0	DMA/REF/PC0 [[PC0]]→[DC0], LOWER BYTE 0 1 1 0	FL/REF/PC0 0 0 1 0

Table 4-6. 3852 DMI Responses to ROMC States (Continued)

ROMC STATE	SIGNAL NAME	SIGNAL CONDITION OR INFORMATION CONTAINED		
		FIRST MEMORY ACCESS	SECOND MEMORY ACCESS	THIRD MEMORY ACCESS (LONG CYCLE ONLY)
	Φ			
	WRITE			
	CYCLE REQ			
0F	ADDR	[PC0]	DMA/REF/PC0	FL/REF/PC0
	DATA BUS		INTERRUPT VECTOR (0F LOWER BYTE, 13 UPPER BYTE)*	
	CPU SLOT	1	0	0
	MEM IDLE	0	1	0
	CPU READ	0	0	0
	REGDR	0	0	0
10			SEE 0D	
11	ADDR	[PC0]	DMA/REF/PC0	FL/REF/PC0
	DATA BUS		[[PC0]]→[DC0], UPPER BYTE	
	CPU SLOT	1	0	0
	MEM IDLE	0	1	0
	CPU READ	0	1	1
	REGDR	0	0	0
12	ADDR	[PC0]	DMA/REF/PC0	FL/REF/PC0
	DATA BUS		BYTE FROM CPU TO [PC0] LOWER	
	CPU SLOT	1	0	0
	MEM IDLE	0	1	0
	CPU READ	0	0	0
	REGDR	0	0	0
13			SEE 0F	
14	ADDR	[PC0]	DMA/REF/PC0	FL/REF/PC0
15	DATA BUS	[PC0]u, [PC1]u, [DC0]u, [PC0]L, [PC1]L, [DC0]L, BYTE FOR I/O PORT		
16	CPU SLOT	1	0	0
17	MEM IDLE	0	1	0
18	CPU READ	0	0	0
19	REGDR	0	0	0
1A				
1B	ADDR	[PC0]	DMA/REF/I0	FL/REF/I0
	DATA BUS		[SELECTED I0 PORT DATA]	
	CPU SLOT	1	0	0
	MEM IDLE	0	1	0
	CPU READ	0	0	0
	REGDR	0	1 (IF SELECTED)	1 (IF SELECTED)

*From interrupting device.

Table 4-6. 3852 DMI Responses to ROMC States (Continued)

ROMC STATE	SIGNAL NAME	SIGNAL CONDITION OR INFORMATION CONTAINED		
		FIRST MEMORY ACCESS	SECOND MEMORY ACCESS	THIRD MEMORY ACCESS (LONG CYCLE ONLY)
	Φ WRITE CYCLE REQ			
1C	ADDR	[PC0]	DMA/REF/PC0	FL/REF/PC0
	DATA BUS		*	
	CPU SLOT	1	0	0
	MEM IDLE	0	1	0
	CPU READ	0	0	0
	REGDR	0	0	0
1D			SEE 0D	
1E 1F	ADDR	[PC0]	[PC0]	DMA/REF/PC0
	DATA BUS		$[PC0]_L$ (1E), [PC0] u(1F)	
	CPU SLOT	1	1	0
	MEM IDLE	0	0	1
	CPU READ	0	0	0
	REGDR	0	1	1

*During INS or OUTS instruction for Port 0 or 1: I/O data byte. During INS, OUTS, instructions, other ports: I/O port address. Otherwise not used, and short cycle.

Note:

See Table 4-5 for a description of symbols used in this table.

4.6 MEMORY REFRESH AND DIRECT MEMORY ACCESS

These two topics are covered together, since in terms of 3852 DMI logic, they are similar operations.

As described in Section 4.4, CYCLE REQ identified 2 or 3 memory access periods within an instruction cycle.

Either the first or the second access period, as summarized in Table 4-4, is reserved for the instruction cycle being decoded. Let us refer to this as the "reserved" access period. If the ROMC state for the instruction cycle requires data to be read out of RAM, then the read occurs during the reserved access period. If the ROMC state for the instruction cycle requires data to be input to an address reigster, or if no data movement occurs on the data bus, then the reserved access period is not used for any memory access—it is, in effect, wasted.

One more memory access may occur within the instruction cycle; this occurs during either the second or third access period, as summarized in Table 4-4, while the data bus latches hold data accessed during the first period. We will refer to this as the "free" access period.

Some available free access periods must be used to refresh dynamic RAM. A refresh uses logic within the 3852 DMI. Therefore a refresh occurs in parallel to anything else that is going on.

If the free access period is not used to refresh dynamic RAM, it may be used by a 3854 DMA device to perform direct memory accesses. As described in Section 6, DMA uses a separate data channel to access memory, so DMA can occur in parallel with anything else that is going on within the F8 system.

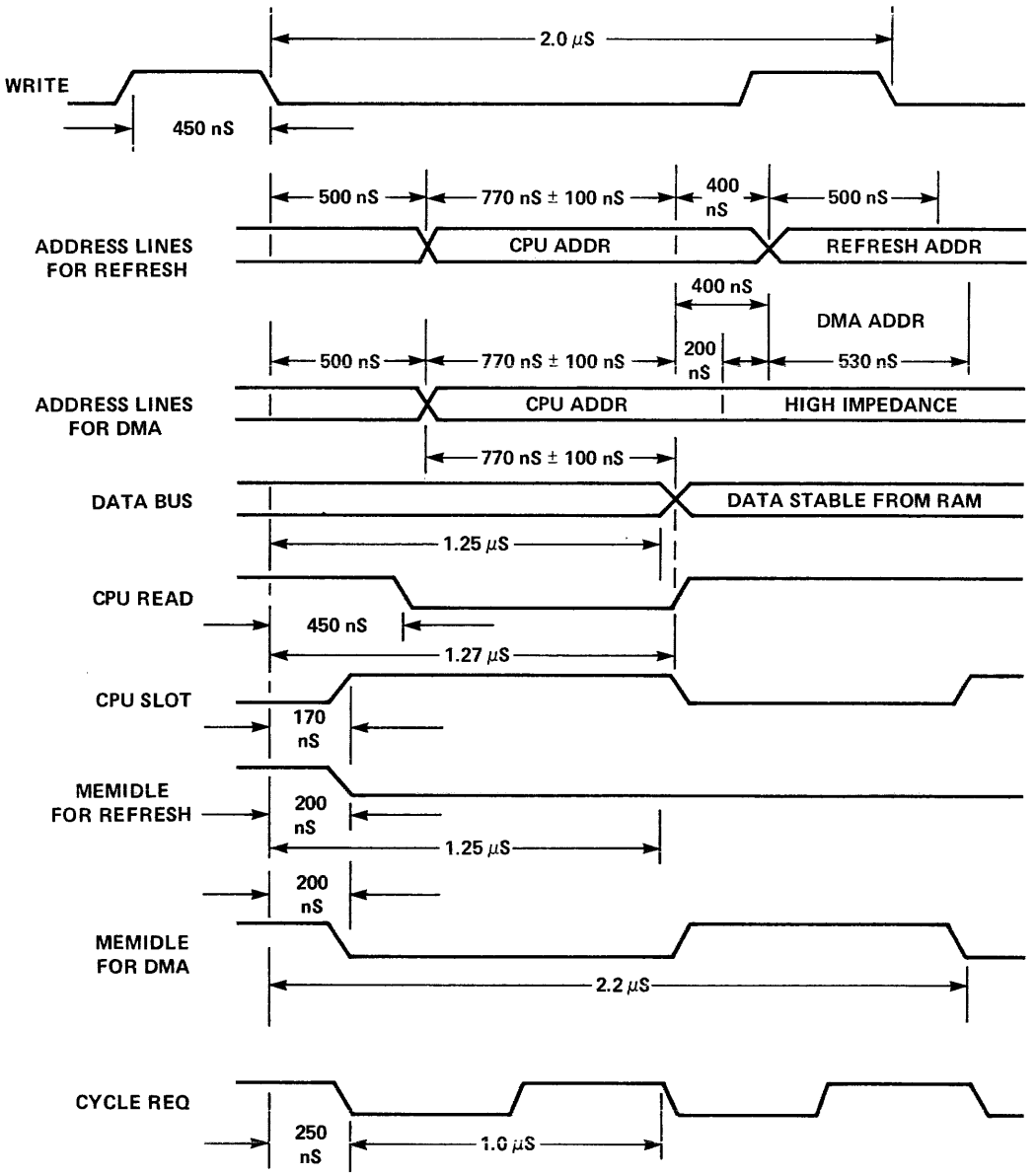


Figure 4-10. Timing for Memory Refresh and DMA during a Short Cycle Memory Read, with Address Out of Program Counter

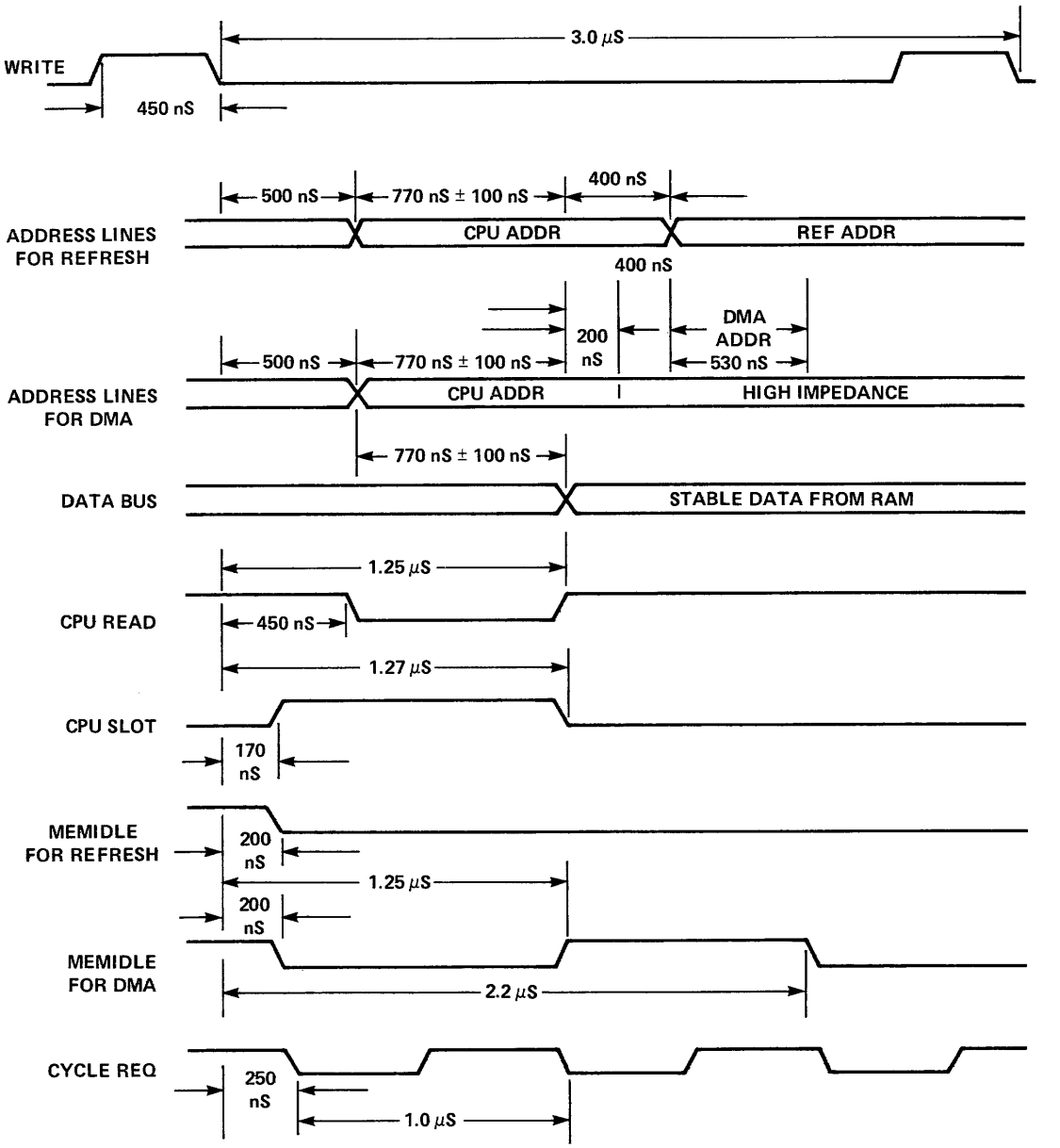


Figure 4-11. Timing for Memory Refresh and DMA during a Long Cycle Memory Read, with Address Out of Program Counter

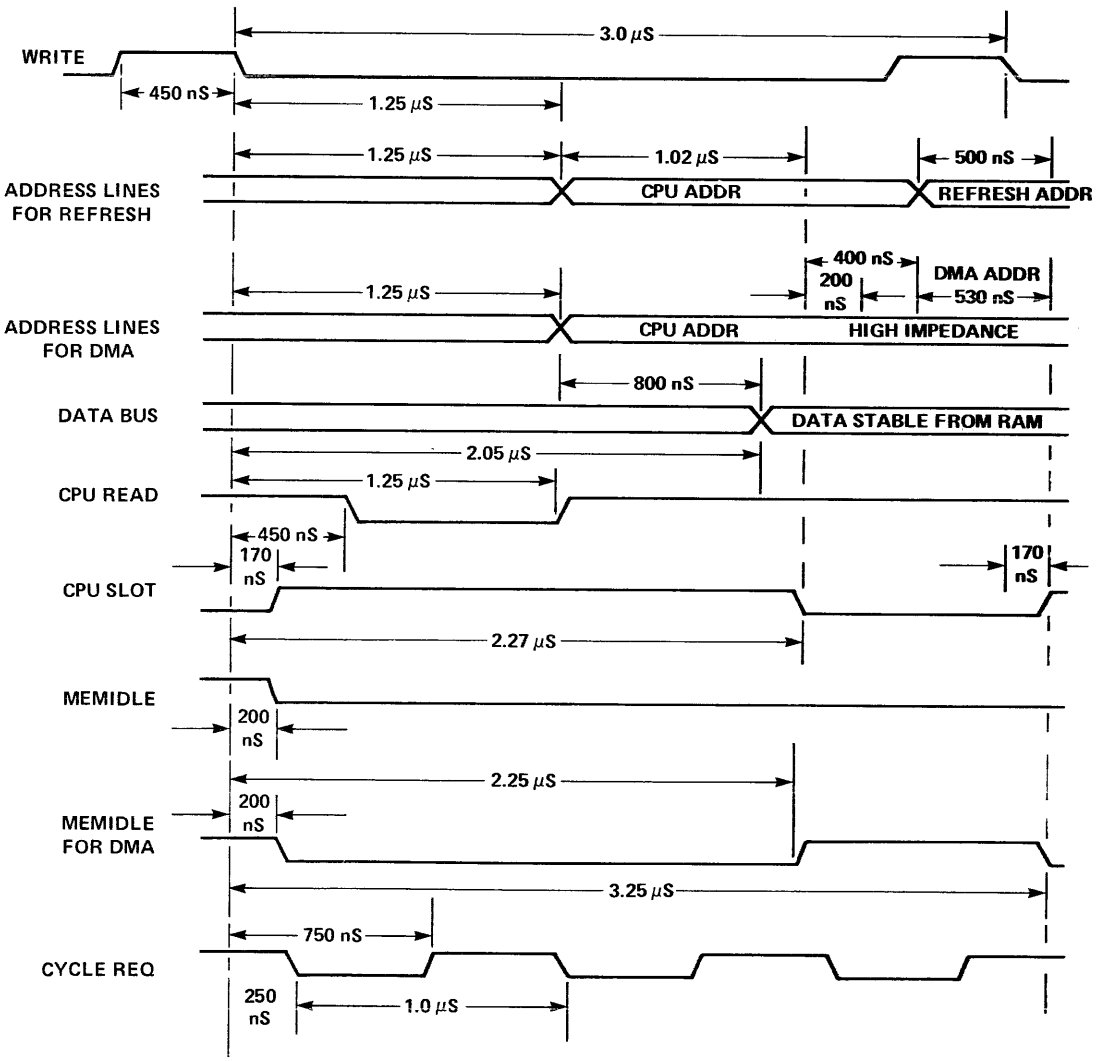


Figure 4-12. Timing for Memory Refresh and DMA during a Long Cycle Memory Read, with Address Out of Data Counter

Figures 4-10, 4-11 and 4-12 indicate worst cases timing for memory refresh and for DMA access. In an F8 write-to-memory cycle (ST), no refresh or DMA may take place, therefore the timing is as shown in Figure 4-8.

A complete memory refresh cycle will execute in F milliseconds, where F is given by:

$$F = (2^6)\ T \cdot R$$

where T is the instruction cycle time, either 2 μsec or 3 μsec
R is the refresh rate, either 4 (for one slot in 4) or 8 (for one slot in 8)

4.7 USING A 3852 DMI WITH STATIC MEMORY

The 3852 DMI may control static memory; refresh is, of course, no longer necessary. If I/O Port 0D (or ED) is set to 01, turning off DMA and memory refresh, then the 3852 DMI now has the characteristics of a 3853 SMI, as described in Section 5.

Static RAM may be accessed via DMA if the control port is set to 00 and the data bus is properly buffered as in any DMA system.

4.8 SUMMARY OF 3852 DMI SYSTEM RAM CHARACTERISTICS

From the worst case timing waveforms presented in Figures 4-5 through 4-8 and 4-10 through 4-12, the AC characteristics of static and dynamic RAMs suitable for use with the 3852 DMI can be derived. Three distinct cases arise:

1. Static RAM with no DMA or refresh. The timing characteristics recommended are the same as those recommended for the 3853 (see Section 5).
2. The 3852 DMI used with dynamic RAM and no DMA. Here the recommended RAM characteristics are as follows:

Access Time	500 nS max.
Address Set-up Time to WRITE	600 nS max
Data Set-up Time to WRITE	550 nS max.
WRITE Pulse-width	350 nS max.
Data and Address Hold Times	200 nS max.
Read Cycle Time	900 nS max.
WRITE Cycle Time	3 μS max.

3. A system using either static or dynamic RAM with a 3852 and DMA. The DMA access dominates the timing requirements, with resulting recommended RAM characteristics as follows:

Access Time	550 nS max.
Address/Data Stable Time (In most memory specifications, this is the CE width during READ or WRITE)	580 nS max.

5.0 The 3853 Static Memory Interface (SMI)

THE 3853 STATIC MEMORY INTERFACE (SMI)

The 3853 SMI provides all interface logic needed to include up to 65,536 bytes of static RAM memory in an F8 microcomputer system. In response to control signals output by the 3850 CPU, the 3853 SMI generates address and control signals needed by standard static RAM devices.

The memory addressing logic of the 3853 SMI is almost identical to the memory addressing logic of the 3852 DMI which was described in Section 4.0. However, static memory does not need to be refreshed; therefore, the 3852 DMI dynamic memory refresh logic is not implemented on the 3853 SMI. This means that the 3853 SMI cannot support direct memory access since memory refresh and DMA logic are interdependent.

Since the 3853 SMI has no memory refresh or DMA logic, it includes a programmable timer and interrupt control circuitry, as described for the 3851 PSU (with a few small differences).

The 3853 SMI may therefore be visualized as a hybrid of the 3852 DMI memory addressing logic, and the 3851 PSU programmable timer and interrupt control circuitry. This is functionally illustrated in Figure 5-1. The figure shows logic functions, registers, data paths and device pins with signal names; control signals within the SMI are not shown.

Because the 3853 SMI is a hybrid of 3852 DMI logic and 3851 PSU logic, repetitive description of logic functions are not given in this section. If you plan to use a 3853 SMI device, first read Sections 3 and 4, then in this section note the ways in which the 3853 SMI differs from appropriate logic descriptions, as given for the 3852 DMI and 3851 PSU.

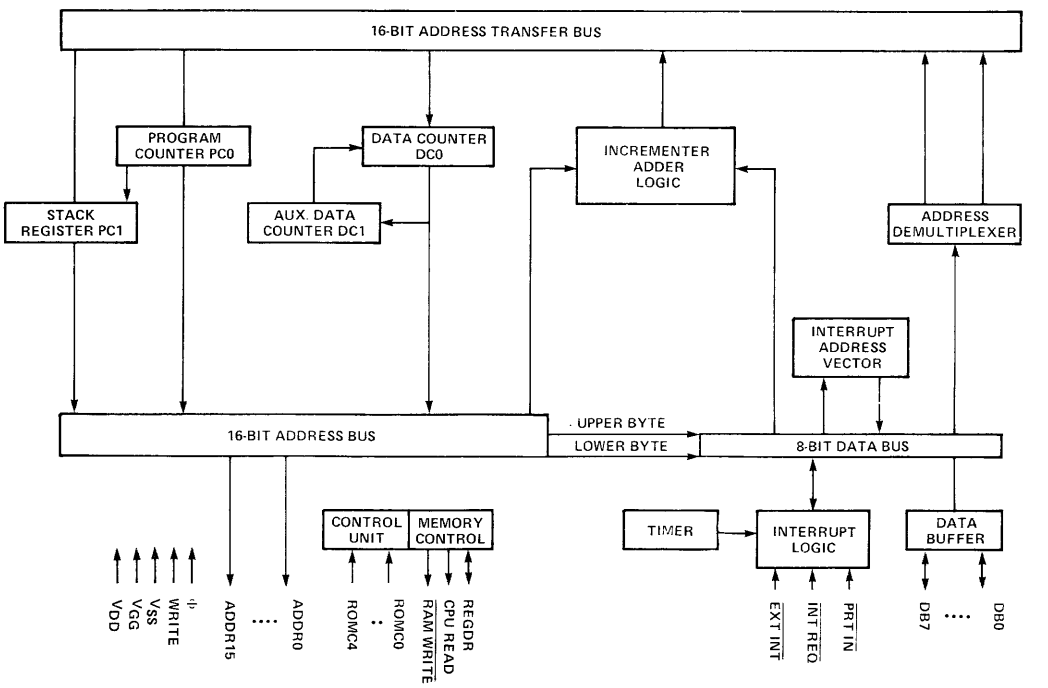


Figure 5-1. Logic Organization and Pins for the 3853 SMI

The 3853 SMI requires +5V and +12V power supplies; the chip is manufactured using N-channel, Isoplanar MOS technology, therefore power dissipation is very low, typically less than 335 mW.

5.1 DEVICE ORGANIZATION

3853 SMI device organization is described below in terms of differences between 3852 DMI and 3853 SMI memory addressing logic, and differences between 3851 PSU and 3853 SMI timer and interrupt processing logic.

5.1.1 Memory Addressing Logic

The only difference between 3853 SMI memory addressing logic, as compared to 3852 DMI memory addressing logic, is that the 3853 SMI has no memory refresh and DMA capabilities. There are two consequences of this omission.

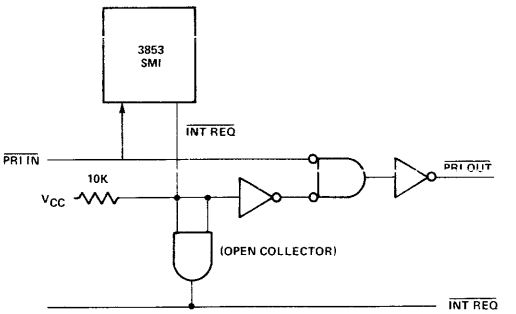
First, the three timing signals, CPU SLOT, CYCLE REQ and MEM IDLE are not generated, therefore they do not require device pins. These pins are used instead by interrupt logic.

Second, since the 3853 SMI does not have to access memory within a single memory access period, as identified by CPU SLOT, data bus timing is relaxed when using the 3853 SMI, as compared to the 3852 DMI. Figures 5-3 through 5-6 indicate the worst case timing for the four possible machine cycles. The implications of this relaxed data bus timing parameter are that slower static memories may be used with the 3853 SMI; but, on the other hand, memory controlled by a 3853 SMI cannot be accessed by a 3854 DMA. Another implication is that a latching type buffer is not needed between memory and the data bus.

5.1.2 Timer and Interrupt Logic

Programmable timer and interrupt handling logic on the 3853 SMI differ from similar logic on the 3851 PSU in two ways.

First, since only three device pins are available for use by interrupt logic, there is no priority out signal. This means that if a 3853 SMI is in an interrupt priority daisy chain, then it must be the last device in the daisy chain or else external logic must generate $\overline{\text{PRI OUT}}$ as follows:



Second, the 3853 SMI interrupt address vector consists of two programmable I/O ports. The interrupt address vector is set under program control, rather than being a mask option, as it is with the 3851 PSU. Even though the 3853 SMI interrupt address vector is programmable, bit 7 is still set to 0 for a timer interrupt, or to 1 for an external interrupt, as described for the 3851 PSU.

5.1.3 I/O Ports

The 3853 SMI has four I/O ports reserved for its use. Addresses 0C, 0D, 0E and 0F are reserved for the four 3853 SMI I/O ports.

I/O ports 0C and 0D are used for the interrupt address vector upper and lower bytes, respectively. They can be written into and read from using the I/O instructions.

I/O port 0E is the interrupt control port.

I/O port 0F is the programmable timer.

Use of the interrupt control I/O port with address 0E is identical to the 3851 PSU's interrupt control I/O port, as described in Section 3.6.1. Use of the programmable timer is also identical.

5.2 SIGNAL DESCRIPTIONS AND ELECTRICAL CHARACTERISTICS

Figure 5-2 illustrates the 3853 SMI device pins. Signal names agree with Figure 5-1 and are summarized in Table 5-1.

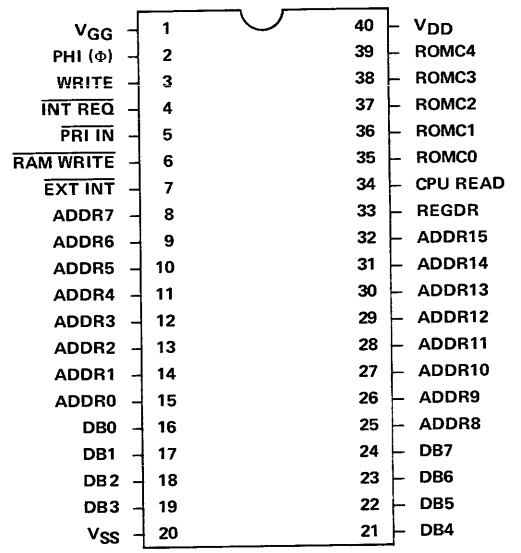


Figure 5-2. 3853 SMI Pin Assignments

Table 5-1. 3853 SMI Signal Summary

PIN NAME	DESCRIPTION	TYPE
DB0-DB7	Data Bus Lines	Bi-directional
ADDR0-ADDR15	Address Lines	Output
Φ, WRITE	Clock Lines	Input
$\overline{\text{INT REQ}}$	Interrupt Request	Output
$\overline{\text{PRI IN}}$	Priority In Line	Input
$\overline{\text{RAM WRITE}}$	Write Line	Output
$\overline{\text{EXT INT}}$	External Interrupt Line	Input
REGDR	Register Drive Line	Input/Output
CPU READ	CPU Read Line	Output
ROMC0-ROMC4	Control Lines	Input
V_{SS}, V_{DD}, V_{SS}	Power Supply Lines	Input

5.2.1 Signal Descriptions

Individual signals are described next. Signal characteristics are given in Table 4-2.

Φ and WRITE are the clock outputs from the 3850 CPU.

ROMC0 through ROMC4 are the control signals output by the 3850 CPU.

DB0 through DB7 are the bi-directional data bus lines which link the 3853 SMI with all other devices in the F8 system. Only data moving to or from 3853 SMI address and control registers use the 3853 SMI DB0-DB7 pins.

ADDR0 through ADDR15 are 16 address lines via which an address is transmitted to dynamic RAM. The address may come from the PC0 or DC0 registers.

$\overline{\text{RAM WRITE}}$. When low, this signal specifies that data is to be written into a RAM location. When high, this signal is off; that is, $\overline{\text{RAM WRITE}}$ high does not necessarily specify a read operation.

CPU READ. When high, this signal specifies that data is to be read out of a RAM location. When low, this signal is off; that is, CPU READ low does not specify a write operation; that is done by $\overline{\text{RAM WRITE}}$ low.

REGDR. This signal functions both as an input and an output. As an input, it can be clamped low by an external open collector gate. This prevents the 3853 SMI from placing a byte out of its PC1 or DC0 registers onto the data bus. The SMI internally supplies a pull-up resistor. The signal, functioning as an output, can control data bus buffers. The SMI will internally clamp REGDR low except during those ROMC states during which the SMI should drive PC1 or DC0 registers or either of its two control registers (I/O ports) onto the data bus. Figure 5-3 shows the REGDR internal logic.

$\overline{\text{EXT INT}}$. A high to low transition on this signal is interpreted as an interrupt request from an external device.

$\overline{\text{PRI IN}}$. Unless this input signal is low, the 3853 SMI will not set $\overline{\text{INT REQ}}$ low in response to an interrupt.

$\overline{\text{INT REQ}}$. This signal becomes the $\overline{\text{INT REQ}}$ input to the 3850 CPU. $\overline{\text{INT REQ}}$ must be output low in order to interrupt the 3850 CPU; this only occurs if $\overline{\text{PRI IN}}$ is low, and 3853 SMI interrupt control logic is requesting an interrupt.

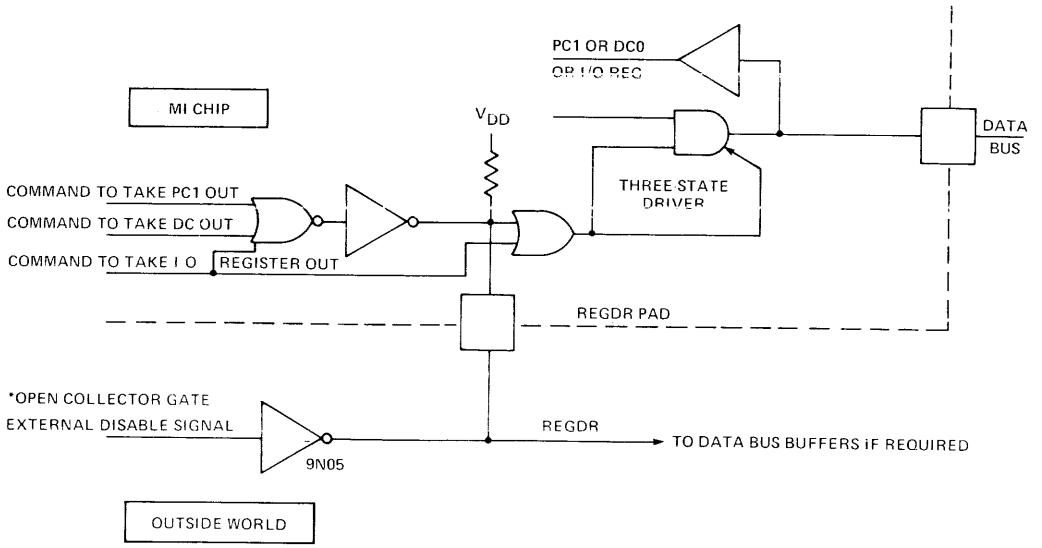


Figure 5-3. REGDR Controls Data Bus Drivers

5.2.2 Electrical Specifications

Electrical specifications are identical for the 3853 SMI and the 3852 DMI. See Section 4.2.2.

5.3 TIMING

Timing for the 3853 SMI is shown in Figure 5-4 and tabulated in Table 5-2. The timing of the interrupt signals, $\overline{\text{INT REQ}}$, $\overline{\text{PRI IN}}$, and $\overline{\text{EXT IN}}$ are identical to those of the 3851 PSU. The other signals have the same timing as the 3852 DMI signals, except for the address lines.

Figures 5-5 through 5-8 provide 3853 SMI worst case timing corresponding to the four possibilities illustrated for the 3852 DMI in Figures 4-5 through 4-8, respectively.

Based on the waveforms shown in Figures 5-5 through 5-8, the following RAM characteristics are recommended for use with the 3853 SMI:

Access Time	900 nS max.
Address Set-up Time to WRITE	600 nS max.
Data Set-up Time to WRITE	550 nS max.
WRITE Pulse Width	350 nS max.
Data and Address Hold Times	200 nS max.

The above numbers must also allow for any buffer delays which may be present on the data bus.

5.4 INSTRUCTION EXECUTION

The actions taken by the 3853 SMI during instruction execution are a function of the ROMC state. These actions are shown in Table 5-4. The actions are similar to those of the 3852 DMI; the primary difference is that the address lines of the 3853 are always driven.

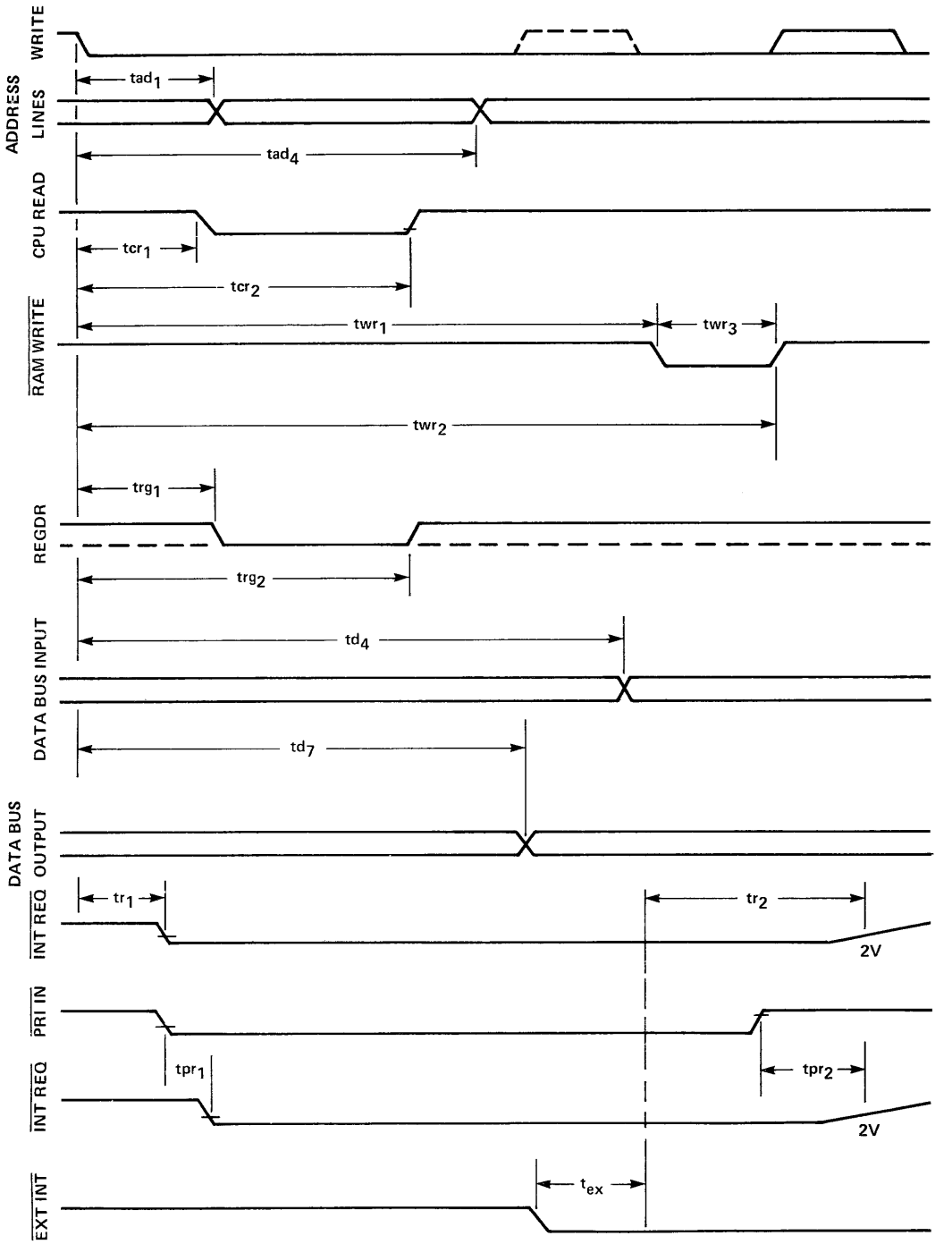


Figure 5-4. 3853 Signal Timing

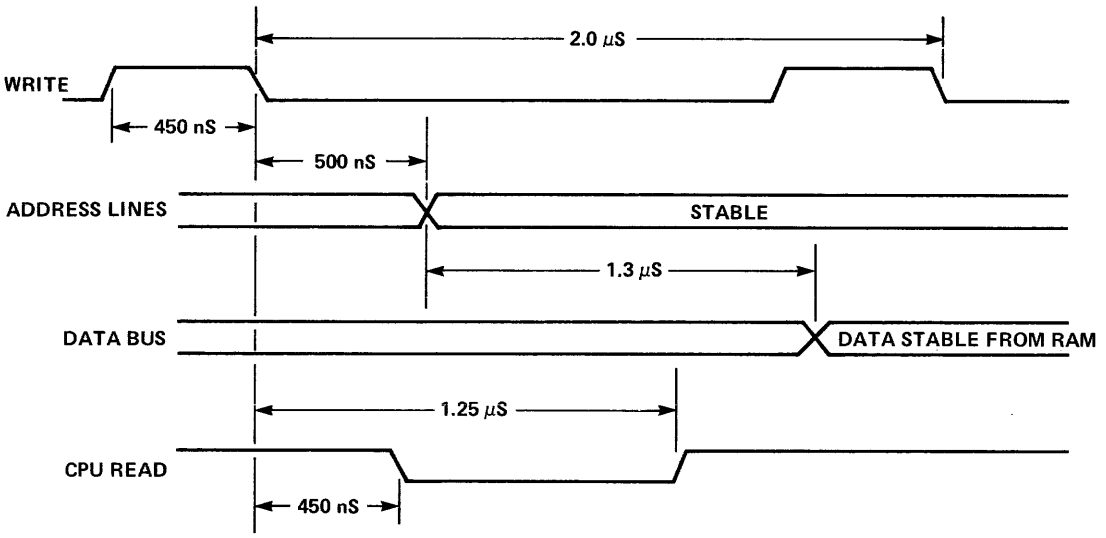


Figure 5-5. 3853 SMI Timing Signals Output during a Short Cycle Memory Read Using PC0

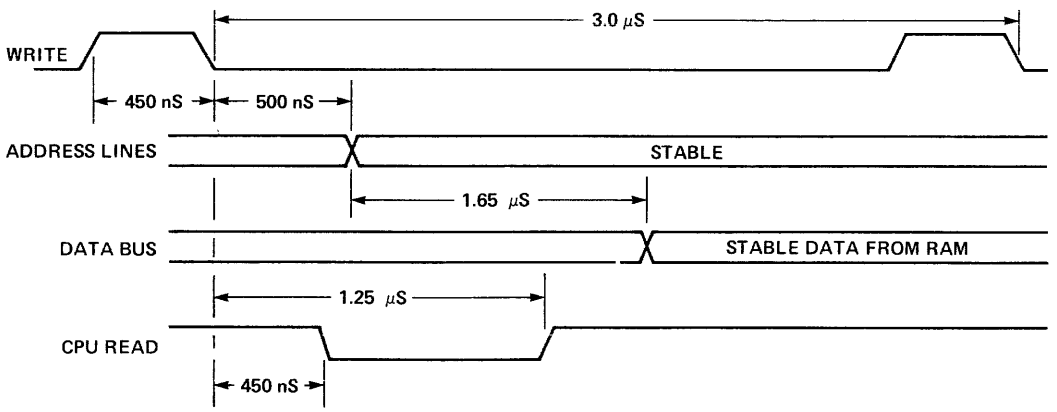


Figure 5-6. 3853 SMI Timing Signals Output during a Long Cycle Memory Read, with Address Out of Program Counter

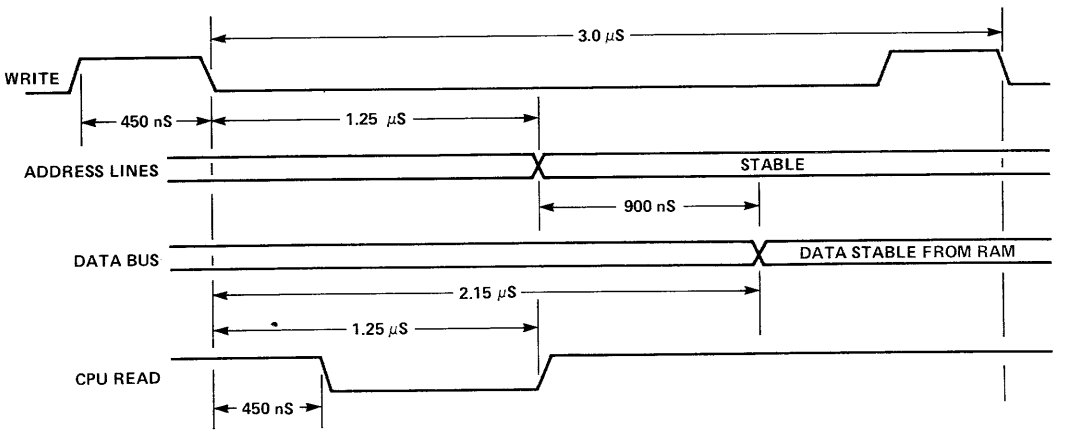


Figure 5-7. 3853 SMI Timing Signals Output during a Long Cycle Memory Read, with Address Out of Data Counter

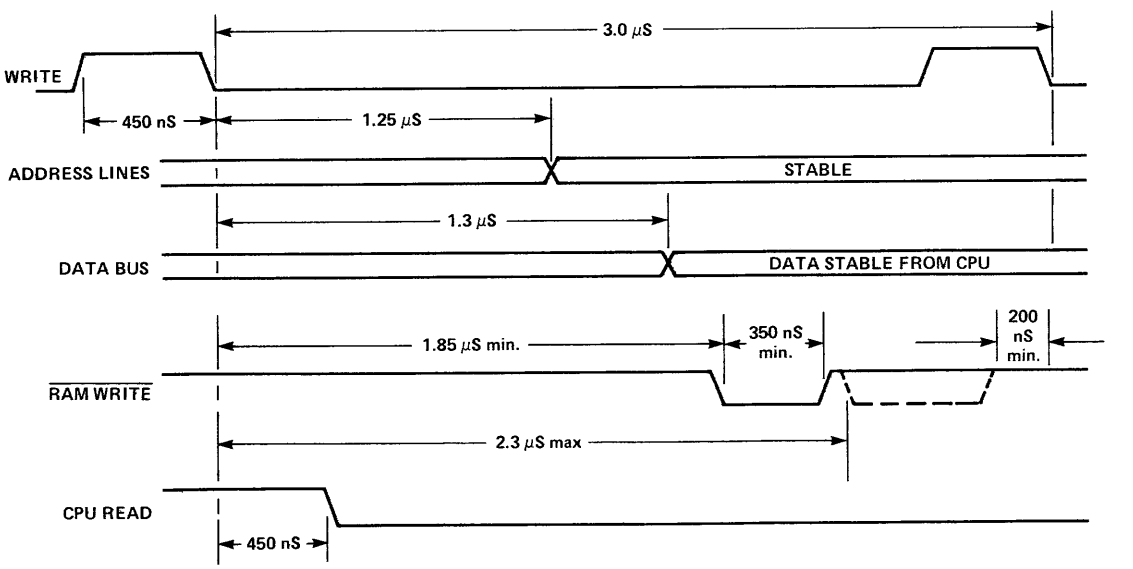


Figure 5-8. 3853 SMI Timing Signals Output during a Write to Memory

Table 5-2. 3853 SMI Output Signals Timing Summary

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNITS	NOTES
$P\Phi$	Φ clock period	0.5		10	μS	Fig. 2-9
td_2	Φ to WRITE - Delay			250	nS	2
tad_1	Address delay if PC0	50	300	500	nS	3
tad_4	Address delay if DC0	$2P\Phi+50-td_2$		$2P\Phi+400-td_2$	nS	3
tcr_1	CPU READ – Delay	50	250	450	nS	1
tcr_2	CPU READ + Delay	$2P\Phi+50-td_2$		$2P\Phi+400-td_2$	nS	1
twr_1	$\overline{\text{RAM WRITE}}$ – Delay	$4P\Phi+50-td_2$		$4P\Phi+450-td_2$	nS	3
twr_2	$\overline{\text{RAM WRITE}}$ + Delay	$5P\Phi+50-td_2$		$5P\Phi+300-td_2$	nS	3
twr_3	$\overline{\text{RAM WRITE}}$ Pulse	350		$P\Phi$	nS	3
trg_1	REGDR – Delay	70	300	500	nS	1
trg_2	REGDR + Delay	$2P\Phi+80-td_2$		$2P\Phi+500-td_2$	nS	1
td_4	WRITE to Data Bus Input Delay			$2P\Phi+1000$	nS	
td_7	WRITE to Data Bus Output Delay	$2P\Phi+100-td_2$		$2P\Phi+850-td_2$	nS	2
tr_1	WRITE to $\overline{\text{INT REQ}}$ – Delay			430	nS	2, 6
tpr_1	$\overline{\text{PRI IN}}$ to $\overline{\text{INT REQ}}$ – Delay		200	240	nS	2, 7
t_{ex}	$\overline{\text{EXT INT}}$ Set-up Time	400			nS	

Notes:

1. C_L = 50 pf.
2. C_L = 100 pf.
3. C_L = 500 pf.
4. On a given chip, the timing for all signals will tend to track. For example, if CPU SLOT for a particular chip is fairly slow and its timing falls out near the MAX delay value specified, then the timing for all signals on that chip will tend to be out near the MAX delay values. Likewise for a fast chip whose signals fall out near the MIN values. This is a result of the fact that processing parameters (which affect device speed) are quite uniform.
5. Input and output capacitance is 3 to 5 pf typical on all pins except V_{DD}, V_{GG}, and V_{SS}.
6. Assume Priority In was enabled ($\overline{\text{PRI IN}}$ = 0) in previous F8 cycle before interrupt is detected in the PSU.
7. PSU has interrupt pending before priority in is enabled.

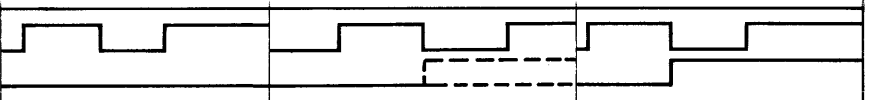
Table 5-3. 3853 SMI Responses to ROMC States

ROMC STATE	SIGNAL NAME	SIGNAL CONDITION OR INFORMATION CONTAINED		
		FIRST MEMORY ACCESS	SECOND MEMORY ACCESS	SECOND ACCESS CONT. (LONG CYCLE ONLY)
	Φ WRITE			
00*	ADDR DATA BUS CPU READ REGDR	[PC0] 0 0	[PC0] INSTRUCTION CODE 1 0	
01	ADDR DATA BUS CPU READ REGDR	[PC0] 0 0	[PC0] OFFSET FOR BRANCH 1 0	[PC0] OFFSET FOR BRANCH 1 0
02	ADDR DATA BUS CPU READ REGDR	[PC0] 0 0	[DC0] INSTRUCTION OPERAND 1 0	[DC0] INSTRUCTION OPERAND 1 0
03**	ADDR DATA BUS CPU READ REGDR	[PC0] 0 0	[PC0] INSTRUCTION OPERAND 1 0	[PC0] INSTRUCTION OPERAND 1 0
04			SEE 0D	
05	ADDR DATA BUS CPU READ REGDR	[PC0] 0 0	[DC0] BYTE FROM CPU, TO BE STORED IN RAM 0 0	[DC0] 0 0
06 07 09 0B	ADDR DATA BUS CPU READ REGDR	[PC0] 0 0	IF 06 OR 09: [DC0]; IF 07 or 08: [PC1] IF 06: [DC0]u; 07: [PC1]u; 09: [DC0]L; 0B: [PC1]L 0 1	 0 1
08	ADDR DATA BUS CPU READ REGDR	[PC0] 0 0	[PC0] 0 FROM CPU TO [PC0] 0 0	[PC0] 0 FROM CPU TO [PC0] 0 0
0A	ADDR DATA BUS CPU READ REGDR	[PC0] 0 0	[DC0] OFFSET FOR DC0, FROM CPU 0 0	[DC0] OFFSET FOR DC0, FROM CPU 0 0
0B			SEE 06	

*This is a long cycle for the DS (op code 30) instruction only.

**This instruction is short for BT (op code 8X), BF (op code 9X), and BR7 (op code 8F) if branch not taken, and for DC1 (op code 2A) always.

Table 5-3. 3853 SMI Responses to ROMC States (Continued)

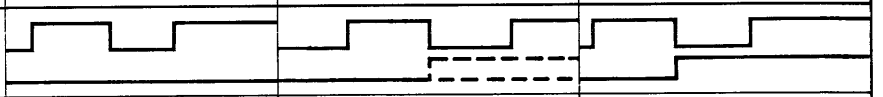
ROMC STATE	SIGNAL NAME	SIGNAL CONDITION OR INFORMATION CONTAINED		
		FIRST MEMORY ACCESS	SECOND MEMORY ACCESS	SECOND ACCESS CONT. (LONG CYCLE ONLY)
	Φ WRITE			
0C	ADDR DATA BUS CPU READ REGDR	[PC0] 0 0	[PC0] [[PC0]] → [PC0]L 1 0	[PC0] [[PC0]] → [PC0]L 1 0
04 0D 10* 1D	ADDR DATA BUS CPU READ REGDR	[PC0] 0 0	[PC0] UNUSED 0 0	
0E	ADDR DATA BUS CPU READ REGDR	[PC0] 0 0	[PC0] [[PC0]] → [DC0]L 1 0	[PC0] [[PC0]] → [DC0]L 1 0
0F 13	ADDR DATA BUS CPU READ REGDR	[PC0] 0 0	[PC0] ** INTERRUPT VECTOR 0 0***	[PC0] ** (0F-LOWER, 13 UPPER) 0 0***
10			SEE 0D	
11	ADDR DATA BUS CPU READ REGDR	[PC0] 0 0	[PC0] [[PC0]] → [DC0]u 1 0	[PC0] [[PC0]] → [DC0]u 1 0
12	ADDR DATA BUS CPU READ REGDR	[PC0] 0 0	[PC0] BYTE FROM CPU TO [PC0]L 0 0	[PC0] BYTE FROM CPU TO [PC0]L 0 0
13			SEE 0F	
14 15 16 17, 18, 19	ADDR DATA BUS CPU READ REGDR	[PC0] 0 0	[PC0] BYTE FOR [PC0]u, [PC1]u, [DC0]u, [PC0]L, [PC1]L, [DC0]L 0 0	[PC0] 0 0
1A	ADDR DATA BUS CPU READ REGDR	[PC0] 0 0	[PC0] BYTE FROM CPU FOR I/O PORT 0 0	[PC0] 0 0

*This is a long cycle for the DS (op code 30) instruction only.

**If interrupt source: Bit 0-7 = port D, bit 8-15 = 1's (0F or 1B) or selected I/O port: Bit 0-7 = 1's, bit 8-15 = port C (13 or 1B)

***REGDR = 1 if interrupt source or selected I/O port.

Table 5-3. 3853 SMI Responses to ROMC States (Continued)

ROMC STATE	SIGNAL NAME	SIGNAL CONDITION OR INFORMATION CONTAINED		
		FIRST MEMORY ACCESS	SECOND MEMORY ACCESS	SECOND ACCESS CONT. (LONG CYCLE ONLY)
	Φ			
	WRITE			
1B	ADDR	[PC0]	[PC0] *	[PC0] *
	DATA BUS		BYTE FOR CPU FROM SELECTED I/O PORT	
	CPU READ	0	0	0
	REGDR	0	0**	0**
1C	ADDR	[PC0]	[PC0]	[PC0]
	DATA BUS		***	***
	CPU READ	0	0	0
	REGDR	0	0	0
1E 1F	ADDR	[PC0]	[PC0]	[PC0]
	DATA BUS		[PC0] L (1E), [PC0] u (1F)	
	CPU READ	0	0	0
	REGDR	0	1	1

*If interrupt source: Bit 0-7 = port D, bit 8-15 = 1's (0F or 1B) or selected I/O port: Bit 0-7 = 1's, bit 8-15 = port C (13 or 1B)

**REGDR = 1 if interrupt source or selected I/O port.

***During INS or OUTS instruction for port 0 or 1: I/O data byte. During INS, OUTS instructions, other ports: I/O port address. Otherwise not used, and short cycle.

6.0 The 3854 Direct Memory Access Controller (DMA)

The 3854 DMA controller interprets timing signals generated by the DMI device in order to control the direct flow of data between memory and devices external to the F8 microcomputer system. DMA data transfers occur in parallel with any other operations, thus there is no reduction in program execution speed.

+5V and +12V power supplies are required. The 3854 DMA is manufactured using N-channel, Isoplanar MOS technology, therefore power dissipation is very low, typically less than 280 mW.

The 3854 DMA is functionally illustrated in Figure 6-1; the figure shows logic functions, registers, data paths and device pins (with signal names). Control signals within the DMA are not shown.

6.1 DEVICE ORGANIZATION

The 3854 DMA device makes use of time slots during which the CPU is not accessing memory. During these time slots, the 3854 DMA device generates data transfer control signals which enable data to be read out of read-write memory, or to be written into read-write memory. The 3852 DMI device outputs the MEM IDLE signal to identify time slots available for DMA access.

In addition to providing appropriate data transfer control signals, the 3854 DMA controller outputs the address of the memory location which is to be accessed.

6.1.1 I/O Ports

Every 3854 DMA controller has four 8-bit registers which are addressed as I/O ports.

Since there may be up to four DMA controllers in an F8 system, 16 I/O port addresses are reserved for the exclusive use of DMA controllers, as shown in Table 6-1.

The four I/O port address that will be used by any DMA are defined by the two signals (P1 and P2) which are input to the DMA controller and become bits 2 and 3 of the I/O port address. This may be illustrated as follows:

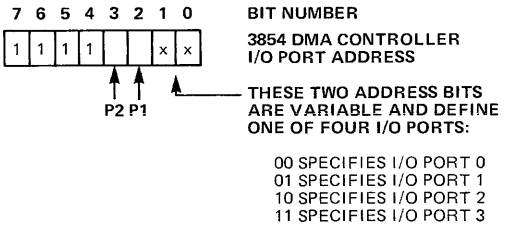


Table 6-1. Addresses of I/O Ports Used by 3854 DMA Devices

FUNCTION OF I/O PORT	FIRST 3854	SECOND 3854	THIRD 3854	FOURTH 3854
Address, L.O. Byte (PORT0)	F0	F4	F8	FC
Address, H.O. Byte (PORT1)	F1	F5	F9	FD
Count, L.O. Byte (PORT2)	F2	F6	FA	FE
Count, H.O. Four bits, and Control (PORT3)	F3	F7	FB	FF

6.1.2 DMA Options

The four I/O ports of a DMA device must be loaded with appropriate data to control the DMA operation. I/O ports are loaded using OUT instructions. The contents of I/O ports may be read at any time using IN instructions.

Before a DMA operation begins, the beginning address of the memory buffer from which data will be read, or to which data will be written, must be loaded into I/O ports 0 and 1. I/O ports 2 and 3 are used to define the length of the memory buffer which is to be accessed plus various DMA options and controls, as illustrated in Figure 6-2.

With reference to Figure 6-2, observe that 12 bits are set aside to define the memory buffer length (byte count), therefore memory buffers up to 4096 bytes in length may be written into or read via DMA. A byte count of 01 transfers one byte; a count of 00 transfers 4096 bytes.

Bit 7 of I/O port 3 may be used at any time to start or stop DMA operations. During normal initiation sequence this bit will be zero while I/O

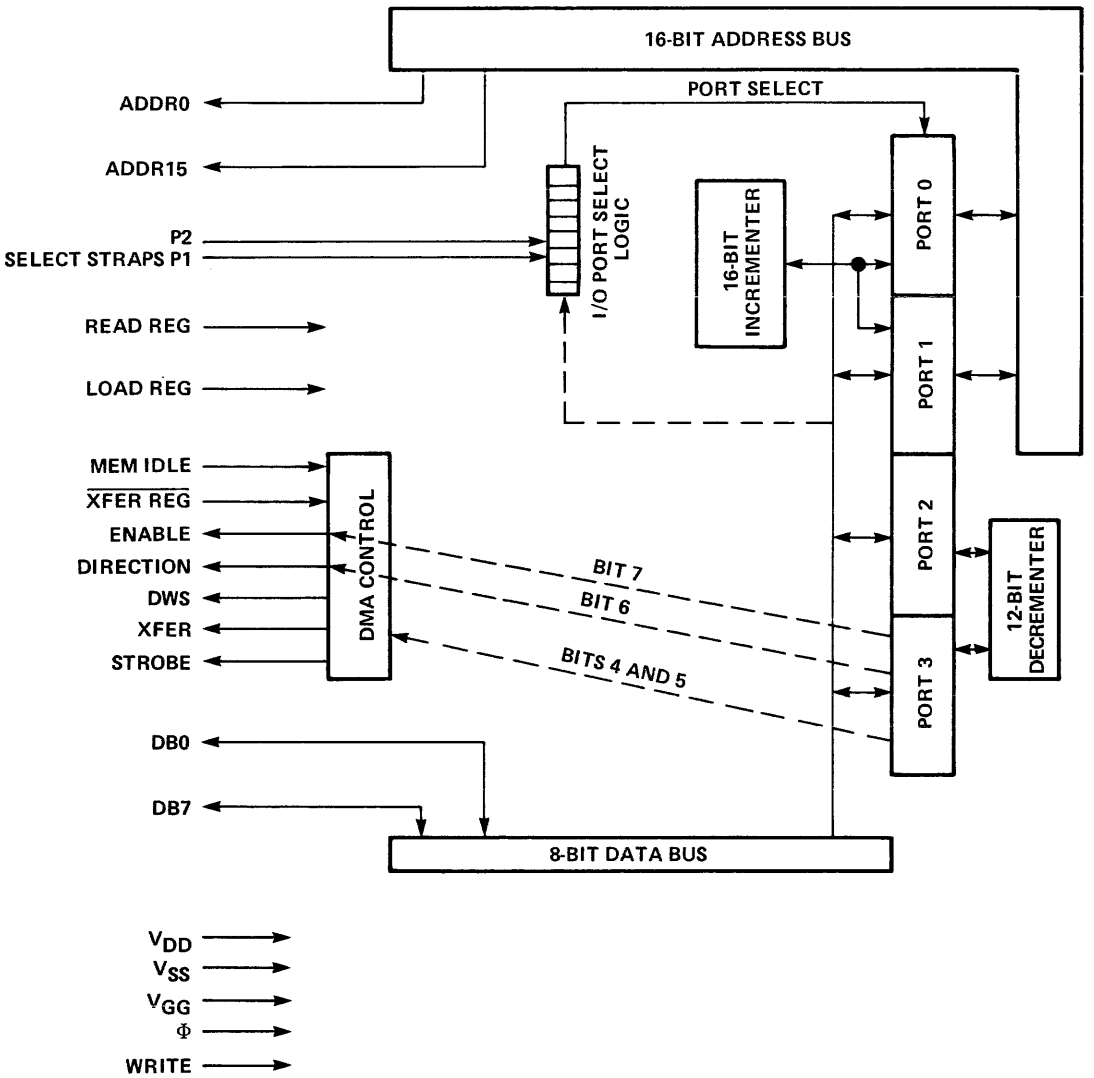


Figure 6-1. Logic Organization and Pins for the 3854 DMA Device

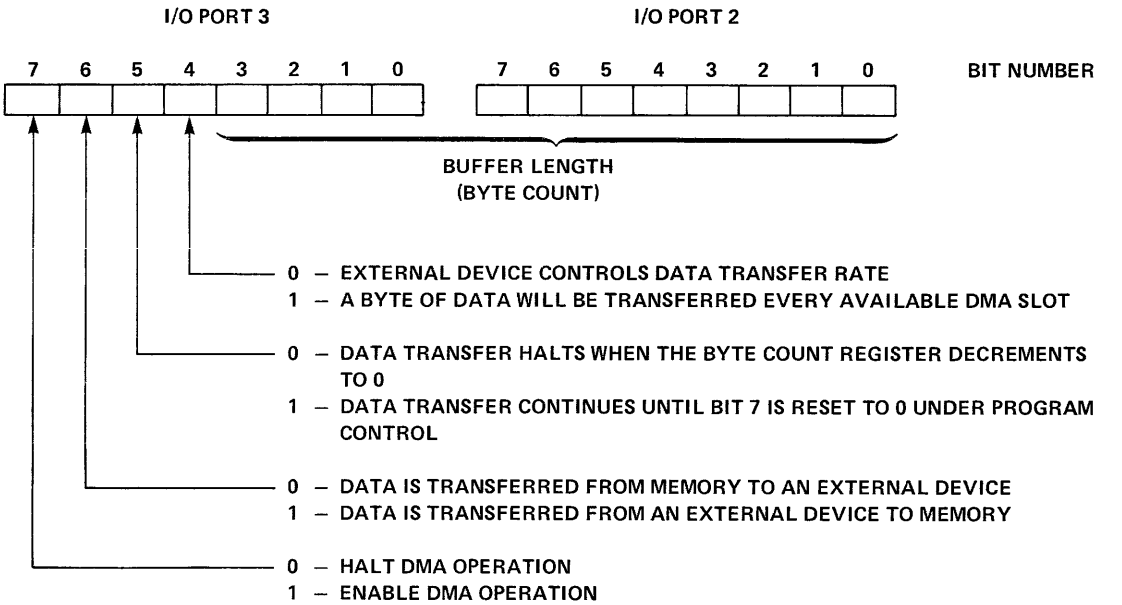


Figure 6-2. How PORT2 and PORT3 are Used to Control DMA Operations

ports 0, 1 and 2 are loaded with appropriate data. Then in order to initiate the DMA operation, I/O port 3 will be loaded with a data byte that includes a 1 in the high order bit. However, in the case of repeated block transfers, it may only be necessary to reload port3, as port 2 will hold zero and the contents of port 0 and 1 will be the address of the last byte previously transferred plus 1.

The direction of the DMA data transfer is determined by bit 6 of I/O port 3. If this bit is zero, data will be read out of memory by the external device. If this bit is one, data will be written into memory by the external device.

The rate of DMA data transfer is determined by bit 4 of I/O port 3. If this bit is zero, then the external device must provide a transfer request ($\overline{\text{XFER REQ}}$) signal whenever it is ready for a DMA data transfer; the actual data transfer will then occur during the next DMA slot, as identified by MEM IDLE high. In other words, the external device controls DMA transfer rate. If bit 4 of I/O port 3 is 1, then the 3854 DMA controller assumes that external logic is ready for a DMA transfer whenever MEM IDLE high identifies a DMA slot. In other words, the F8 system controls DMA transfer rate.

Each time a DMA data transfer occurs, logic within the 3854 DMA controller that is clocked by XFER increments the memory address in I/O ports 0 and 1 and decrements the buffer counter in I/O ports 2 and 3. If bit 5 of I/O port 3 is zero, then DMA transfer will automatically halt and clear bit 7—the enable bit—as soon as the buffer length is decremented to zero. If bit 5 of I/O port 3 is 1, however, the buffer length count is ignored and DMA data transfer will continue until halted by an OUT instruction setting bit 7 of I/O port 3 to zero. If continuous DMA data transfer is specified by bit 5 of I/O port 3 being set to 1, then the memory address in I/O port 0 and 1 will still be incremented and the buffer length decremented after each DMA access, even though the buffer length counter is ignored.

6.1.3 DMA Control Logic

This logic generates the control signals required to implement DMA data transfers, as defined by the options described in Section 6.1.2. For a discussion of these control signals see Section 6.2.

The LOAD REG and READ REG signal inputs to the DMA control logic requires special mention.

Most F8 support devices have a control unit which decodes the five ROMC signals output by the 3850 CPU. However, the 3854 DMA controller will only respond to ROMC states 1A and 1B, which are "write to I/O port" and "read from I/O port" controls, respectively. All other states constitute "No Operations." Therefore, instead of having a control unit, external logic is used to decode these ROMC state signals, creating READ REG in response to state 1B, and LOAD REG in response to state 1A.

6.1.4 Increment and Decrement Logic

This logic is used to increment the address in ports 0 and 1 and to decrement the buffer length in ports 2 and 3.

6.1.5 The Data and Address Busses

Note carefully that whereas the address bus is used to output the address of the memory location which will be accessed during the next DMA operation, 3854 DMA controller's connection to the data bus is used only to transfer data between 3854 DMA device I/O ports and the CPU. The data bus is not used to transfer data bytes during a DMA operation.

6.2 SIGNAL DESCRIPTIONS AND ELECTRICAL CHARACTERISTICS

Figure 6-3 illustrates the 3854 DMA device pins. Signal names agree with Figure 6-1 and are summarized in Table 6-2. Signal characteristics are given in Table 6-3. Figure 6-4 illustrates the way in which input timing and controls are combined to generate the output control signals ENABLE, DIRECTION, DWS, XFER and STROBE.

6.2.1 Signal Descriptions

Φ and WRITE are the clock outputs from the 3850 CPU. Φ is only used in the generation of STROBE. WRITE is only used for loading I/O ports and data bus monitoring for I/O match.

READ REG and LOAD REG are control signals that must be input to the 3854 DMA device in lieu of the five ROMC state signals. Since the 3854 DMA device only responds to ROMC states 1A and 1B, external logic must generate READ REG true for ROMC state 1B and LOAD REG true for ROMC state 1A, as follows:

READ REG = ROMC0·ROMC1·$\overline{\text{ROMC2}}$·ROMC3·ROMC4
LOAD REG = $\overline{\text{ROMC0}}$·ROMC1·$\overline{\text{ROMC2}}$·ROMC3·ROMC4

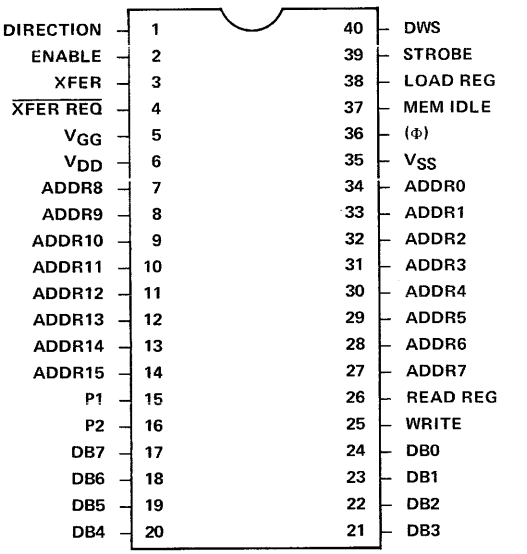


Figure 6-3. 3854 DMA Pin Assignments

Table 6-2. 3854 DMA Signals

PIN NAME	DESCRIPTION	TYPE
DB0-DB7	Data Bus Lines	Bi-directional (3-State)
ADDR0-ADDR15	Address Lines	Output (3-State)
Φ, WRITE	Clock Lines	Input
LOAD REG/READ REG	Registers Load/Read Line	Input
P1, P2	Port Address Select	Input
MEM IDLE	Memory Idle Line	Input
$\overline{\text{XFER REQ}}$	Transfer Request Line	Input
ENABLE, DIRECTION	Control Status Lines	Output
DWS, XFER	DMA Write Slot, Transfer	Output
STROBE	Output Strobe Line	Output
V_{SS}, V_{DD}, V_{GG}	Power Lines	Input

DB0 through DB7 are the bi-directional data bus lines which link the 3850 CPU with all other devices in the F8 system. Note, that only data being transferred to or from one of the four 3854 I/O ports uses the data bus pins. Data being transferred to or from memory under DMA control completely bypasses the 3854 DMA device.

P1 and P2 must be strapped externally to determine the addresses of the four 3854 DMA device I/O ports as illustrated in Section 6.1.1.

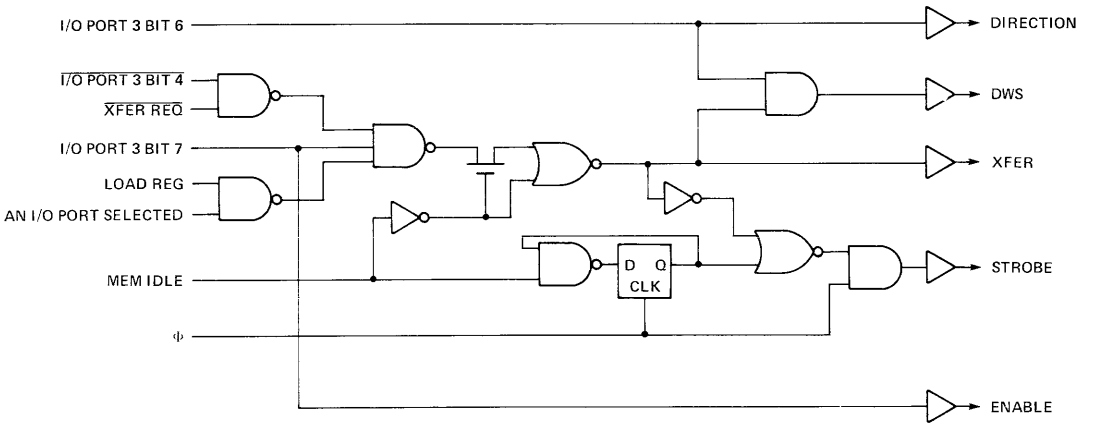


Figure 6-4. DMA Control Signals Output by the 3854 DMA Device

Table 6-3. Summary of 3854 DMA Signal Characteristics

ELECTRICAL SPECIFICATIONS

Absolute Maximum Ratings (Above which useful life may be impaired)

V_{GG}	+15V to -0.3V
V_{DD}	+7V to -0.3V
All other Inputs & Outputs	+7V to -0.3V
Storage Temperature	-55°C to +150°C
Operating Temperature	0°C to +70°C

Note: All voltages with respect to V_{SS}.

DC CHARACTERISTICS: V_{SS} = 0V, V_{DD} = +5V ± 5%, V_{GG} = +12V ± 5%, T_A = 0 to +70°C

SUPPLY CURRENTS

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
I_{DD}	V_{DD} Current		20	40	mA	f = 2 MHz, Outputs Unloaded
I_{GG}	V_{GG} Current		15	28	mA	f = 2 MHz, Outputs Unloaded

SIGNAL	SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
DATA BUS (DB0-DB7)	V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = -100 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1.6 mA
	I_{IH}	Input High Current		1	μA	V_{IN} = 6V, 3-State mode
	I_{IL}	Input Low Current		-1	μA	V_{IN} = V_{SS}, 3-State mode
ADDRESS LINES (ADDR0-ADDR15)	V_{OH}	Output High Voltage	4.0	V_{DD}	Volts	I_{OH} = -1 mA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 3.2 mA
	I_L	Leakage Current		1	μA	V_{IN} = 6V, 3-State mode
ENABLE, DIRECTION DWS (DMA WRITE SLOT), XFER, STROBE	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = -100 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 2 mA
	I_L	Leakage Current		1	μA	V_{IN} = 6V

Table 6-3. Summary of 3854 DMA Signal Characteristics (Continued)

SIGNAL	SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
MEM IDLE, $\overline{\text{XFER REQ}}$	V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_L	Leakage Current		1	μA	V_{IN} = 6V
LOAD REG, READ REG, P1, P2	V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_L	Leakage Current	0	1	μA	V_{IN} = 6V
WRITE, Φ	V_{IH}	Input High Voltage	4.0	V_{DD}	Volts	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_L	Leakage Current	0	1	μA	V_{IN} = 6V

Note:

Positive current is defined as conventional current flowing into the pin referenced.

ADDR0 through ADDR15 are the 16 address lines via which the address of the memory location to be accessed during the current DMA operation are output. This memory address originates in I/O ports 0 and 1 as illustrated in Figure 6-1. These lines are in a high impedance state when no DMA operation is taking place (XFER = 0).

MEM IDLE is a timing signal input to the 3854 DMA device from the 3852 DMI device. This signal is output high to identify time slots when memory is available for DMA access.

$\overline{\text{XFER REQ}}$ is a control signal which must be input to the 3854 DMA device by an external device which is controlling the DMA transfer rate (I/O port 3, bit 4 must be set to zero in this case). When low, this signal causes a byte of data to be transferred to or from memory during the next available DMA time slot. This signal is latched while MEM IDLE = 1; changes during a DMA time slot are therefore ignored.

DIRECTION is an output control signal which reflects the contents of I/O port 3, bit 6. When high, data is being written into memory. When low, data is being read from memory.

ENABLE is a control output which reflects the contents of I/O port 3, bit 7. When high, DMA data transfers may occur. When low, DMA is disabled.

XFER is a control output which identifies the time slots when a DMA data transfer is occurring. XFER is high whenever MEM IDLE is high and other conditions specify that a DMA data transfer is to occur during the next available time slot. These conditions are that a DMA transfer is specified either by bit 4 of I/O port 3 being set to 1, or by $\overline{\text{XFER REQ}}$ being low while DMA has been enabled and the currently executed instruction is not attempting to access the DMA device's I/O ports. ENABLE is provided by I/O port 3, bit 7. DMA data transfers are inhibited while an instruction is accessing the I/O ports of the 3854 DMA device since these instructions may be in the process of modifying the parameters that control the DMA operation. This inhibit is generated by ANDing the LOAD REG input with an internal I/O port selected signal.

DWS is a DMA write slot signal. It is the AND of XFER and DIRECTION, thus it is true during any DMA write to memory.

STROBE is a DMA transfer signal output that is a narrow pulsed and delayed version of XFER: it is used for strobing data and for generating RAM WRITE. STROBE is high only during the second occurrence of Φ clock high after MEM IDLE goes true, provided that XFER is also true.

6.3 TIMING

Figure 6-5 provides timing for signals input to and output from the 3854 device. Table 6-3 identifies the symbols used in Figure 6-5. Address bus timing is referenced to MEM IDLE. The description of 3852 DMI timing in Section 4.4 includes DMA address and data bus timing within the context of an F8 instruction cycle.

Section 13—which discusses use of DMA—gives the overview of signal relationships during the whole of a data transfer operation. The timing here is the detailed timing of transferring one byte and of other segments of operation.

Referring to Figure 6-5, address line timing assumes that control criteria for a DMA transfer have already been met when MEM IDLE goes to logic "1".

In the ENABLE-DIRECTION timing, TD_9 represents a time delay from WRITE. TD_9 is the time taken to clear the ENABLE bit (bit 7 of I/O port 3) when the buffer length count goes to zero.

6.4 DMA I/O OPERATION

DMA registers are loaded and read when the 3850 CPU executes I/O instructions that access the DMA registers. The I/O instructions use the DATA BUS to transmit the I/O address in one instruction cycle and to transfer data during the following instruction cycle. The appropriate control signal, LOAD REG or READ REG, will become active during this second cycle. The DMA will load one of its registers during a cycle with LOAD REG high if the I/O address, which had been on the data bus during the previous cycle, matched a DMA port address. The register is loaded and the address comparator is up-dated by the WRITE clock; these are the only function of WRITE in the 3854 DMA. Likewise a DMA chip will drive the contents of a selected register onto the DATA BUS only while READ REG is high, if there was a similar address match during the prior cycle. I/O address assignment is made using pins P1 and P2 as discussed in Section 6.1.1.

Table 6-4. 3854 DMA Device Signals Summary

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNITS	NOTES
PΦ	Φ Clock Period	0.5		10	μS	Note 1
PW_1	Φ Pulse Width	180		PΦ-180	nS	t_r, t_f = 50 nS typ.
td_1	Φ to WRITE + Delay	60		300	nS	Note 1
td_2	Φ to WRITE – Delay	60		250	nS	Note 1
PW_2	WRITE Pulse Width	PΦ-100		PΦ	nS	t_r, t_f = 50 nS typ.
td_3	WRITE to READ/LOAD REG Delay			600	nS	
td_4	DB Input Set-up Time			300	nS	
td_6	$\overline{\text{XFER REQ}}$ to MEM IDLE Set-up	200			nS	
td_7	MEM IDLE to ADDR True	50	200	500	nS	C_L = 500 pf
td_7'	MEM IDLE to ADDR 3-State	30		250	nS	C_L = 500 pf
td_8	READ REG to DB Output	40		300	nS	C_L = 100 pf
td_9	WRITE to ENABLE & DIRECTION + Delay			450	nS	C_L = 50 pf
td_9'	MEM IDLE to ENABLE - Delay			400	nS	C_L = 50 pf
td_{10}	MEM IDLE to XFER & DWS + Delay			300	nS	C_L = 50 pf
td_{10}	MEM IDLE to XFER & DWS – Delay			300	nS	C_L = 50 pf
td_{11}	Φ to STROBE + Delay	30		200	nS	C_L = 50 pf
td_{11}	Φ to STROBE – Delay	30		200	nS	C_L = 50 pf

Notes:

1. These specifications are those of Φ and WRITE as supplied by the 3850 CPU.
2. Input and output capacitance is 3 to 5 pf typical on all pins except V_{DD}, V_{GG}, and V_{SS}.

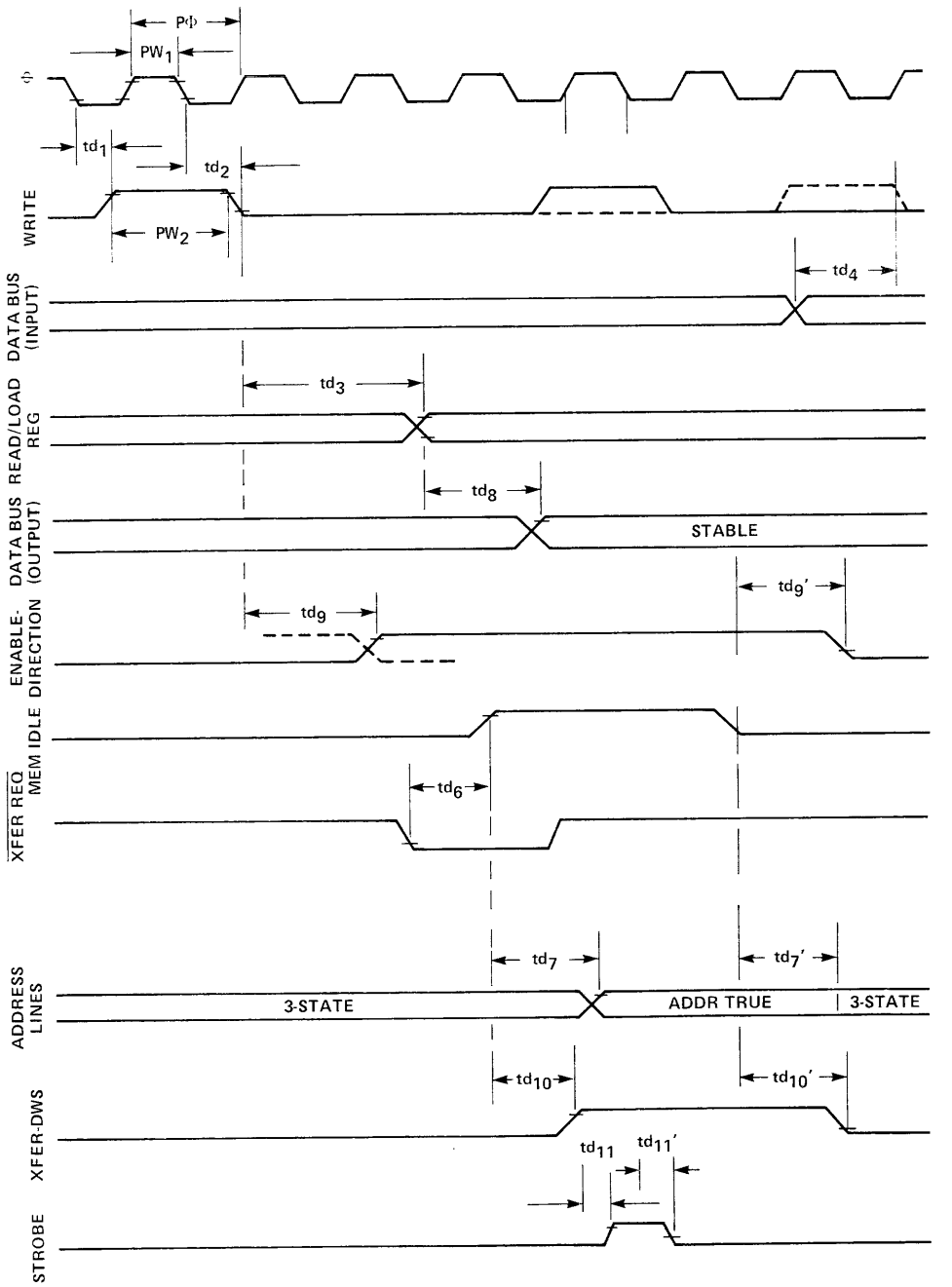


Figure 6-5. 3854 DMA Device Signals and Timing

7.0 The 3861 Peripheral Input/Output (PIO)

THE 3861 PERIPHERAL INPUT/OUTPUT (PIO)

The 3861 Peripheral Input/Output (PIO) provides two 8-bit I/O ports, logic to handle an external interrupt, and a programmable interval timer. An 8-bit wide bi-directional data bus transfers I/O data bytes between the CPU and PIO. The I/O ports of the PIO are configured in the standard pull-up option.

The PIO is used in systems that require the I/O capability and interrupt functions of the 3851 Program Storage Unit (PSU) but that do not need the ROM storage of the PSU. The PIO is pin compatible to the PSU. There are five versions of PIO available; each version has its own set of pre-assigned I/O port addresses and interrupt vectors (see Table 7-1).

+5V and +12V power supplies are required. The 3861 PIO is manufactured using N-channel, Isoplanar MOS technology, therefore power dissipation is very low, typically less than 250 mW.

The 3861 PIO is functionally illustrated in Figure 7-1; the figure shows logic functions, registers, data paths and device pins (with signal names); control signals within the PIO are not shown.

7.1 DEVICE ORGANIZATION

The device organization of the 3861 Peripheral Input/Output (PIO) is similar to the I/O portion of the 3851 Program Storage Unit. The PIO includes I/O logic, timer logic, interrupt logic, data bus logic, and control logic.

7.1.1 Interrupt Logic

This logic responds to an interrupt request signal which may originate internally from timer logic, or be input by an external device. Based on priority considerations, the interrupt request is passed on to the 3850 CPU, as described in Section 7.7. The

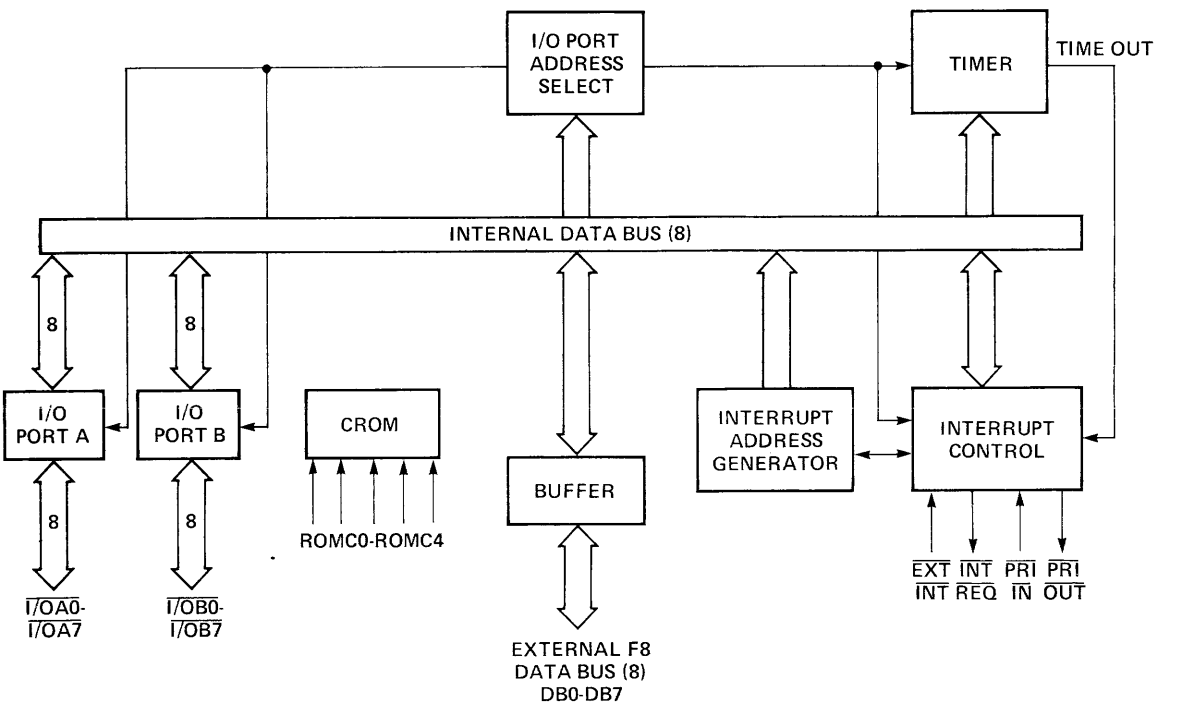


Figure 7-1. Logical Organization and Pins for the 3861 PIO

Table 7-1. 3861 Port and Address Assignments (HEX)

	VERSION	PORT ADDRESSES	INTERRUPT ADDRESS VECTOR	
			TIMER	EXTERNAL
3861	A	4-7	0600	0680
3861	B	8-B	0340	03C0
3861	C	20-23	0320	03A0
3861	D	24-27	0360	03E0
3861	E	4-7	0020	00A0

interrupt vector address provided by an interrupting 3861 PIO is fixed; interrupt vectors for the five versions of the 3861 PIO are given in Table 7-1.

7.1.2 Timer Logic

Every 3861 PIO has a polynomial shift register which may be used in conjunction with interrupt logic to generate real-time intervals.

Upon counting down to the time-out value, the timer uses interrupt logic in order to signal that it has timed out.

The timer is programmable and is handled as though it were an I/O port. Using an OUT or OUTS instruction, a value may be loaded into the timer in order to determine the real-time period at the end of which a time-out interrupt will be generated. For information on use of the timer, see Section 7.6.

7.1.3 The Data Bus

The 8-bit data bus is the main path for transfer of information between the 3850 CPU and other devices in the F8 microprocessor system. It is identified in Figure 7-1 by data lines DB0-DB7. This is the same data bus that has been described in Section 2.1.9.

7.1.4 I/O Ports

Every 3851 PIO has four I/O addresses assigned to it. The addresses for each of the five versions of 3861 PIO are given in Table 7-1. Table 7-2 gives the allocation of the 4 addresses on the PIO. The two lowest of the four addresses are assigned to the two I/O ports; these two ports are identified as I/O ports A and B in Figure 7-1. The ports are used to transfer data to or from external devices. The other two I/O addresses are assigned to two internal registers of the 3861 PIO that control interrupt logic and treats these two internal registers as if they were I/O ports, although in fact the registers do not have connection to external devices.

Use of the two I/O ports is explained in Section 7.4.3. Use of the interrupt logic and interval timer is the same as in the 3851 Program Storage Unit and is discussed in Sections 7.6 and 7.5, respectively.

Table 7-2. Allocation of Port Addresses on a 3861 PIO

ADDRESS	ASSIGNED TO
XXXX XX00	I/O Port A
XXXX XX01	I/O Port B
XXXX XX10	Interrupt Control Register
XXXX XX11	Programmable Timer

Where 'XXX XX' are the 6 bits that complete the particular set of four addresses given in Table 7-1.

7.2 SIGNAL DESCRIPTIONS, ELECTRICAL CHARACTERISTICS

Figure 7-2 illustrates the 3861 PIO device pins. Signal names agree with Figure 7-1 and are summarized in Table 7-3.

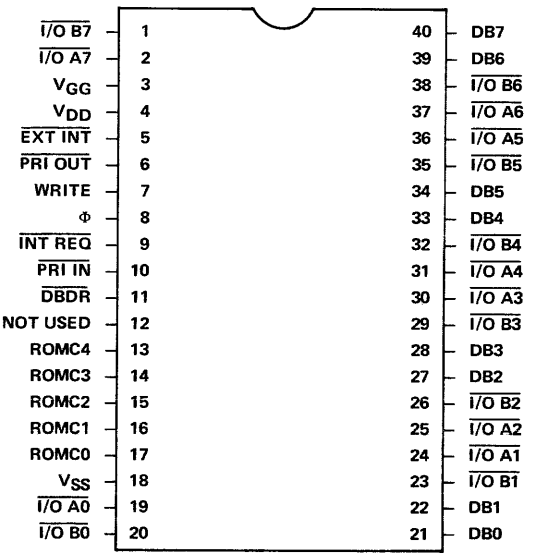


Figure 7-2. 3861 PIO Pin Assignments

7.2.1 Signal Descriptions

Individual signals are described next. Signal characteristics are given in 7.2.2.

Φ and WRITE are the clock outputs from the 3850 CPU.

ROMC0 through ROMC4 are the control signals output by the 3850 CPU.

Table 7-3. 3861 PIO Signals

PIN NAME	DESCRIPTION	TYPE
$\overline{\text{I/O A0}}$-$\overline{\text{I/O A7}}$	I/O Port A	Input/Output
$\overline{\text{I/O B0}}$-$\overline{\text{I/O B7}}$	I/O Port B	Input/Output
DB0-DB7	Data Bus	Bi-directional (3-State)
ROMC0-ROMC4	Control Lines	Input
Φ, WRITE	Clock Lines	Input
$\overline{\text{EXT INT}}$	External Interrupt	Input
$\overline{\text{PRI IN}}$	Priority In	Input
$\overline{\text{PRI OUT}}$	Priority Out	Output
$\overline{\text{INT REQ}}$	Interrupt Request	Output
$\overline{\text{DBDR}}$	Data Bus Drive	Output
V_{SS}, V_{DD}, V_{GG}	Power Supply Lines	Input

DB0 through DB7 are the bi-directional data bus lines which link the 3861 PIO with all other devices in the F8 system.

$\overline{\text{EXT INT}}$. A high to low transition on this signal is interpreted as an interrupt request from an external device.

$\overline{\text{PRI IN}}$. Unless this input signal is low, the 3861 PIO will not set $\overline{\text{INT REQ}}$ low in response to an interrupt.

$\overline{\text{PRI OUT}}$. This signal becomes $\overline{\text{PRI IN}}$ to the next device in the interrupt priority daisy chain. $\overline{\text{PRI OUT}}$ is output high unless $\overline{\text{PRI IN}}$ is entering the 3861 PIO low, and the 3861 PIO is not requesting an interrupt.

$\overline{\text{INT REQ}}$. This signal becomes the $\overline{\text{INT REQ}}$ input to the 3850 CPU. $\overline{\text{INT REQ}}$ must be output low in order to interrupt the 3850 CPU; this only occurs if $\overline{\text{PRI IN}}$ is low, and 3861 PIO interrupt control logic is requesting an interrupt.

$\overline{\text{I/O A0}}$ through $\overline{\text{I/O B7}}$ are input/output ports through which the 3861 PIO communicates with logic external to the microprocessor system.

$\overline{\text{DBDR}}$ is low when the 3861 PIO is outputting data on the data bus (DB0-DB7). For information on using $\overline{\text{DBDR}}$ see Section 7.4.1. $\overline{\text{DBDR}}$ is an open drain signal.

7.2.2 Electrical Specifications

Absolute Maximum Ratings (Above which useful life may be impaired)

V_{GG}	+15V to -0.3V
V_{DD}	+7V to -0.3V
External Interrupt Input	-600 μA to +225 μA
All other Inputs & Outputs	+7V to -0.3V
Storage Temperature	-55°C to +150°C
Operating Temperature	0°C to +70°C

SUPPLY CURRENTS

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
I_{DD}	V_{DD} Current		30	70	mA	f = 2 MHz, Outputs Unloaded
I_{GG}	V_{GG} Current		10	18	mA	f = 2 MHz, Outputs Unloaded

Supply currents measured with V_{DD} = +5V ± 5%, V_{GG} = +12V ± 5%, T_A = 0°C to +70°C. All other electrical specifications are in Table 7-4. All voltages measured with respect to V_{SS}.

7.3 CLOCK TIMING

All timing within the 3861 PIO is controlled by Φ and WRITE, which are input from the 3850 CPU. For a description of these clock signals, and how they are generated, see Section 2.3.

The WRITE clock refreshes and updates 3861 PIO registers, which are dynamic.

The Φ clock drives sequencing logic to precharge interrupt logic. The Φ clock also drives the programmable timer.

7.4 INSTRUCTION EXECUTION

The 3861 PIO responds to signals which are output by the 3850 CPU in the course of implementing instruction cycles. Figure 2-9 illustrates timing for instruction cycles and ROMC signals being output by the CPU.

Table 7-6 summarizes the response of the 3861 PIO to the ROMC states described in Table 2-5.

7.4.1 Data Output by the PIO

Figure 7-3 provides timing when the 3861 PIO outputs data on the data bus. This timing applies whenever a 3861 PIO is the data source. With reference to Figure 2-11, note that the 3861 PIO places data on the data bus, even in the worst case, in time for the set-up required by any 3850 CPU destination.

Table 7-4. A Summary of 3861 PIO Signal Characteristics

SIGNAL	SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
DATA BUS (DB0-DB7)	V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = -100 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1.6 mA
	I_{IH}	Input High Current		1	μA	V_{IN} = 6V, 3-State mode
	I_{OL}	Input Low Current		-1	μA	V_{IN} = V_{SS}, 3-State mode
CLOCK LINES (Φ, WRITE)	V_{IH}	Input High Voltage	4.0	V_{DD}	Volts	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_L	Leakage Current		1	μA	V_{IN} = 6V
PRIORITY IN AND CONTROL LINES ($\overline{\text{PRI IN}}$, ROMC0-ROMC4)	V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_L	Leakage Current		1	μA	V_{IN} = 6V
PRIORITY OUT ($\overline{\text{PRI OUT}}$)	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = -100 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 100 μA
INTERRUPT REQUEST ($\overline{\text{INT REQ}}$)	V_{OH}	Output High Voltage			Volts	Open Drain Output [1]
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1 mA
	I_L	Leakage Current		1	μA	V_{IN} = 6V
DATA BUS DRIVE ($\overline{\text{DBDR}}$)	V_{OH}	Output High Voltage				External Pull-up
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 2 mA
	I_L	Leakage Current		1	μA	V_{IN} = 6V
EXTERNAL INTERRUPT ($\overline{\text{EXT INT}}$)	V_{IH}	Input High Voltage	3.5		Volts	
	V_{IL}	Input Low Voltage		1.2	Volts	
	V_{IC}	Input Clamp Voltage		15	Volts	I_{IH} = 185 μA
	I_{IH}	Input High Current		10	μA	V_{IN} = V_{DD}
	I_{IL}	Input Low Current		-225	μA	V_{IN} = 2V
	I_{IL}	Input Low Current	-150	-500	μA	V_{IN} = V_{SS}
I/O PORT (STANDARD PULL-UP)	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = -30 μA
	V_{OH}	Output High Voltage	2.9	V_{DD}	Volts	I_{OH} = -100 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 2 mA
	V_{IH}	Input High Voltage	2.9	V_{DD}	Volts	Internal Pull-up to V_{DD} [3]
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_{IL}	Leakage Current		1	μA	V_{IN} = 6V
	I_L	Input Low Current		-1.6	mA	V_{IN} = 0.4V [4]

Notes:

1. Pull-up resistor to V_{DD} on CPU.
2. Positive current is defined as conventional current flowing into the pin referenced.
3. Hysteresis input circuit provides additional 0.3V noise immunity while internal/external pull-up provides TTL compatibility.
4. Measured while I/O port is outputting a high level.
5. V_{SS} = 0V, V_{DD} = +5V ± 5%, V_{GG} = +12V ± 5%, T_A = 0°C to +70°C.
6. Output device off.

Table 7-5. A Summary of 3861 PIO Signal AC Characteristics

AC Characteristics: V_{SS} = 0V, V_{CC} = +5V ± 5%, T_A = 0 C to +70°C

Symbols in this table are used by all figures in Section 7.

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
PΦ	Φ Period	0.5		10	μS	
PW_1	Φ Pulse Width	180		PΦ-180	nS	t_r, t_f = 50 nS typ.
td_1	Φ to WRITE + Delay	60		250	nS	C_L = 100 pf
td_2	Φ to WRITE – Delay	60		225	nS	C_L = 100 pf
td_4	WRITE to DB Input Delay			2PΦ+1.0	μS	
PW_2	WRITE Pulse Width	PΦ-100		PΦ	nS	t_r, t_f = 50 nS typ.
PW_S	WRITE Period; Short		4PΦ			
PW_L	WRITE Period; Long		6PΦ			
td_3	WRITE to ROMC Delay			550	nS	
td_7	WRITE to DB Output Delay WRITE to $\overline{DBDR}$ – Delay	2PΦ+100-td_2	2PΦ+200	2PΦ+850-td_2	nS	C_L = 100 pf
td_8	WRITE to $\overline{DBDR}$ + Delay		200		nS	Open Drain
tr_1	WRITE to $\overline{INT\ REQ}$ – Delay			430	nS	C_L = 100 pf [1]
tr_2	WRITE to $\overline{INT\ REQ}$ + Delay			430	nS	C_L = 100 pf [3]
tpr_1	$\overline{PRI\ IN}$ to $\overline{INT\ REQ}$ – Delay			240	nS	C_L = 100 pf [2]
tpr_2	$\overline{PRI\ IN}$ to $\overline{INT\ REQ}$ + Delay			240	nS	C_L = 100 pf
tpd_1	$\overline{PRI\ IN}$ to $\overline{PRI\ OUT}$ – Delay			300	nS	C_L = 50 pf
tpd_2	$\overline{PRI\ IN}$ to $\overline{PRI\ OUT}$ + Delay			365	nS	C_L = 50 pf
tpd_3	WRITE to $\overline{PRI\ OUT}$ + Delay			700	nS	C_L = 50 pf
tpd_4	WRITE to $\overline{PRI\ OUT}$ – Delay			640	nS	C_L = 50 pf
*t_{sp}	WRITE to Output Stable			2.5	μS	C_L = 50 pf, Standard Pull-up
*t_{su}	I/O Set-up Time	1.3			μS	
*t_h	I/O Hold Time	0			nS	
*t_{ex}	$\overline{EXT\ INT}$ Set-up Time	400			nS	

Notes:

1. Assume Priority In was enabled ($\overline{PRI\ IN}$ = 0) in previous F8 cycle before interrupt is detected in the PIO.
2. PSU has interrupt pending before priority in is enabled.
3. Assume pin tied to $\overline{INT\ REQ}$ input of the 3850 CPU.

*4. The parameters which are starred in the table above represent those which are most frequently of importance when interfacing to an F8 system. Other parameters are typically those that are relevant only between F8 chips and not normally of concern to the user.

5. Input and output capacitance is 3 to 5 pf typical on all pins except V_{DD}, V_{GG}, and V_{SS}.

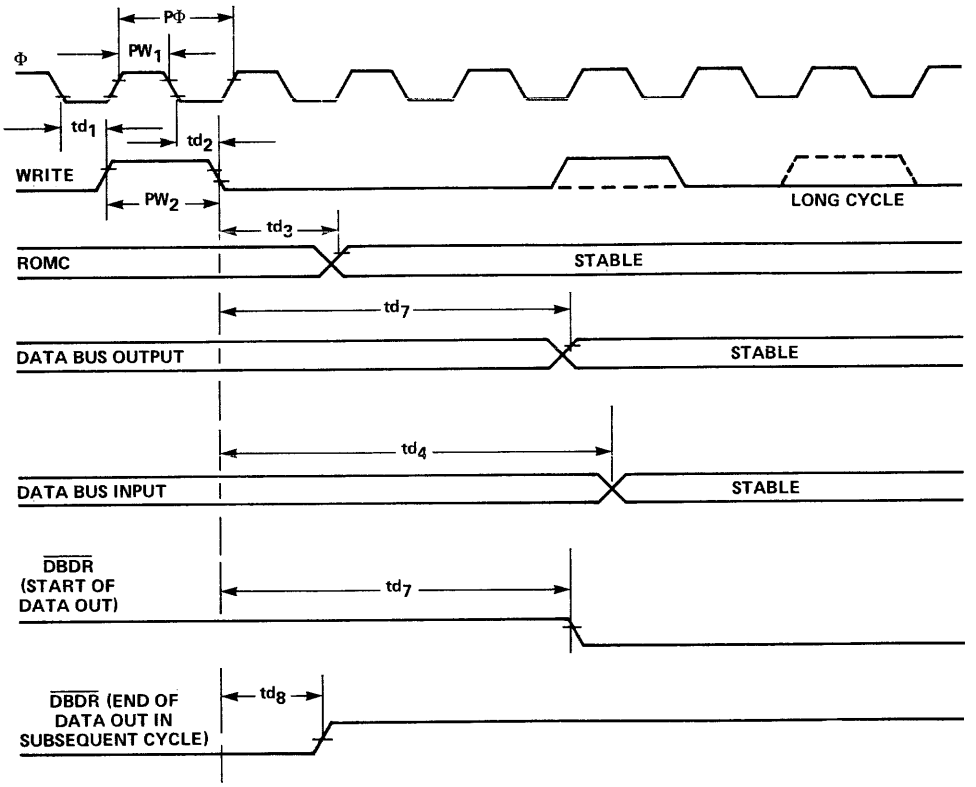


Figure 7-3. 3861 PIO Data Bus Timing

Observe that $\overline{DBDR}$ is low while data output by the 3861 PIO is stable on the data bus. Thus $\overline{DBDR}$ low indicates that the data bus currently contains data flowing from a 3861 PIO. For systems with more than one 3851 PSU or 3861 PIO (as described in Section 11), the $\overline{DBDR}$ outputs may be wire-ORed, and the result may be used as a bus data flow direction indicator. $\overline{DBDR}$ may remain low until td_8 into the instruction cycle following the one in which $\overline{DBDR}$ was set low.

7.4.2 Data Input to the PIO

The 3861 PIO inputs a byte from the data bus when commanded by an output instruction to load one of its two I/O ports or internal registers. Data bus timing requirements for input to the PIO are also given in Figure 7-3.

7.4.3 Input/Output Instruction

The 3861 Peripheral Input/Output (PIO) device executes the OUT instruction in the same manner as the OUTS, likewise for IN and INS. The difference between the long and short form instructions is only the source of the I/O address.

The F8 input/output instructions place the I/O port address on the data bus during one instruction cycle and then use the data bus in the following instruction cycle to do the actual I/O data movement. The ROMC lines that come from the 3850 CPU signal the 3861 PIO that an I/O data movement is occurring during the current instruction cycle. Thus the 3861 PIO needs to know whether the contents of the data bus during the instruction cycle just prior matched any of its four assigned I/O addresses wherever the ROMC lines indicate an I/O transfer. The address select logic answers this need. It constantly monitors the data bus for a match to any of the four addresses and holds the information of a match through the following cycle.

Input instructions that select a port cause the contents of the selected port to be placed on the

Table 7-6. PIO Functions vs. ROMC States

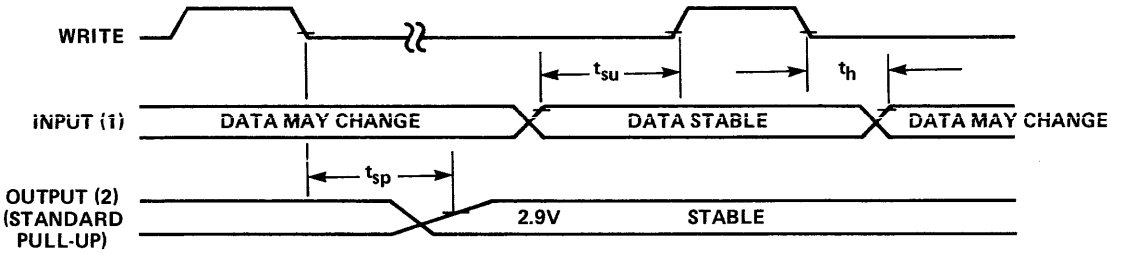
ROMC STATE BINARY	ROMC STATE HEX	3861 FUNCTIONS
00000	00	No-Op
00001	01	No-Op
00010	02	No-Op
00011	03	No-Op
00100	04	No-Op
00101	05	No-Op
00110	06	No-Op
00111	07	No-Op
01000	08	No-Op
01001	09	No-Op
01010	0A	No-Op
01011	0B	No-Op
01100	0C	No-Op
01101	0D	No-Op
01110	0E	No-Op
01111	0F	If this circuit is interrupting and is highest in the priority chain, move lower half of interrupt vector into the data bus.
10000	10	Place interrupt circuitry in an inhibit state that prevents altering the interrupt priority chain.
10001	11	No-Op
10010	12	No-Op
10011	13	If this circuit is interrupting and is highest in the priority chain, move upper half of interrupt vector into data bus and reset interrupt circuit.
10100	14	No-Op
10101	15	No-Op
10110	16	No-Op
10111	17	No-Op
11000	18	No-Op
11001	19	No-Op
11010	1A	If contents of data bus in prior cycle were an address of I/O ports on this device, move current contents of data bus into the appropriate port (I/OA, I/OB, Timer, or Control).
11011	1B	If contents of data bus in prior cycle were an address of I/O ports on this device, move contents of appropriate I/O port onto data bus (I/OA or I/OB).
11100	1C	No-Op
11101	1D	No-Op
11110	1E	No-Op
11111	1F	No-Op

data bus during the input cycle, as discussed in 7.4.1. Only the two I/O ports (lowest two addresses) respond to input instructions. Output instructions that select a port transfer the contents of the data bus to that port as discussed in 7.4.2. Outputs of the latches change at the end of the I/O transfer cycle.

Data bus timing for I/O transfers has been shown in Figure 7-3. Timing at the I/O port pins is shown in Figure 7-4.

7.5 INPUT/OUTPUT INTERFACING

The two PIO ports with the lowest I/O addresses may be used to transmit data between the PIO and external devices. IN and INS instructions cause data at the I/O ports to be transmitted to the CPU; OUT and OUTS instructions cause data in the CPU's accumulator to be loaded into an I/O port. Each I/O pin has an output latch which holds the data last output to that pin.



SYMBOLS USED ARE DEFINED IN TABLE 7-5

Notes:

1. Data from the I/O port is strobed into the accumulator of the CPU at the end of the second instruction cycle during execution of an IN or INS instruction.
2. During an OUT or OUTS instruction, data is strobed into the port latch at the end of the second instruction cycle; thus the cycle shown is the second cycle within the execution of the instruction.
3. Input and output capacitance of 3 to 5 pf typical on all pins except V_{DD}, V_{GG}, and V_{SS}.

Figure 7-4. Timing at PIO I/O Ports

The configuration of the port and an example of its connection to TTL logic is shown in Figure 7-5. Each I/O port pin is a "wire-AND" structure between an internal output data latch and the external signal. The latch is loaded from the data bus.

When outputting data through an I/O port, the pin can be connected directly to a TTL gate input ("TTL Device Input" in Figure 7-5). Each F8 I/O latch may be set high or low under program control. If a logic 1 is set into the latch, then gate (b) will turn on and gate (a) will turn off. Since gate (b) is a large low impedance device, node P will be at V_{SS}. If a logic 0 is set in the latch, gate (b) will turn off and gate (a) will turn on. Gate (a) is a small high impedance device and functions as a pull-up resistor; node P will then be pulled high in the absence of external pull-down devices.

Data is input to the pin from a "TTL Device Output" in Figure 7-5. When data is input to the I/O pin, high or low levels at P drive the hysteresis circuit in the port, and result in logic 1's or 0's being transferred to the accumulator.

Since the I/O pin and the TTL device output at P are wire-ANDed, it is possible for the state of one to affect the transfer of data out from the I/O pin or in from the TTL device output. For example, if the latch in the I/O port is set so that the pin is clamped low by (b), then the "TTL Device Output" cannot change node P. Conversely, if P is clamped to a low level by the TTL Device Output, setting the latch for a high level has no effect.

It can be seen, then, that in most instances I/O port bits should be set for a high level (logic 0), before data input, to prevent incoming logic 0's from being "masked" by logic 1's present at the port from previous outputs. However, the ability to mask bits of a port to logic 1 is useful during some input functions.

Note that the logic 1 of the F8 microprocessor becomes a 0 volt electrical level at the I/O pin for both input and output; likewise logic 0 corresponds to a high electrical level. Also note that the output latches of the I/O ports are not initialized by the system reset sequence.

7.6 PROGRAMMABLE TIMER

The 3861 PIO has an 8-bit shift register, addressable as an I/O port, which functions as a polynomial timer. The timer is loaded with a value of delay; it will count down this value of delay and after the programmed interval will generate an interrupt through the interrupt logic of the PIO.

The OUT or OUTS instruction is used to load the interval value into the programmable timer; the port

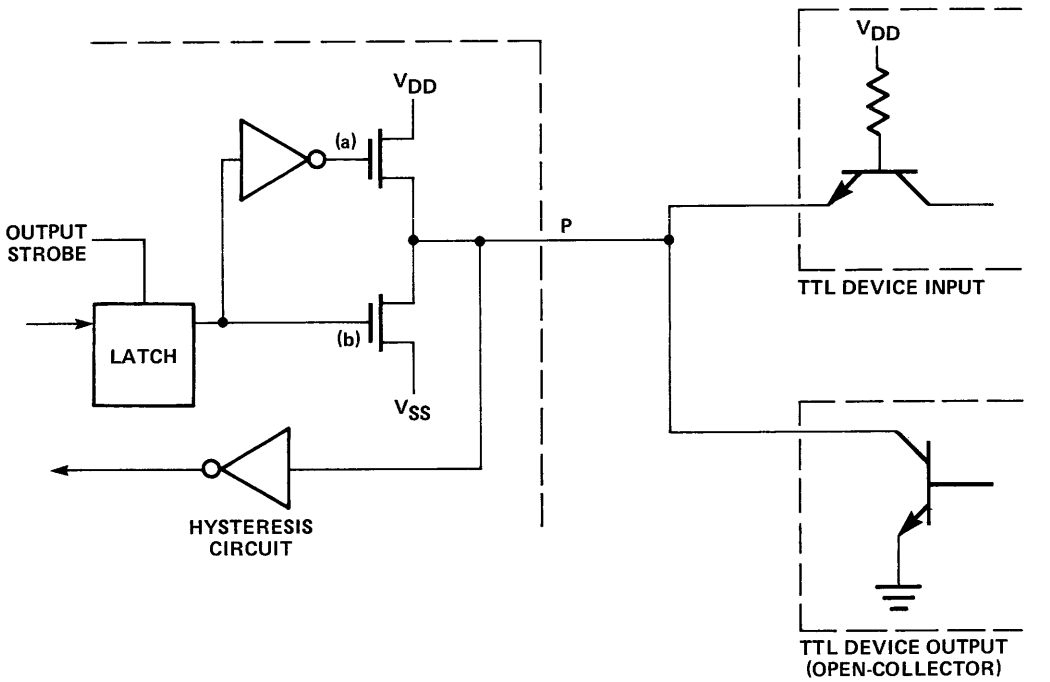


Figure 7-5. An F8 I/O Port Bit

number is H'07', H'0B', H'23', or H'27' as appropriate. The contents of the programmable timer cannot be read using an IN or INS instruction. The timer will time out after a time interval given by the product:

(period of Φ clock) * (timer counts) * 31

The timer will continue running after a time out; subsequent time outs will occur at intervals of 7905 Φ clock periods. The timer will not run if it is loaded with the value H'FF'.

A full discussion of the programmable timer will be found in Section 3.5. The discussion there includes a block diagram, a table of timer counts, and details of the timer interrupt process.

7.7 INTERRUPT LOGIC

The 3861 PIO has an interrupt logic block that is identical to that of the 3851 Program Storage Unit (PSU). The interrupt logic can be programmed to respond to either a transition of the external interrupt input or to the condition that the interval timer has timed out. Upon receiving an interrupt request, the interrupt logic will signal the 3850 CPU using the interrupt request line if the PIO's Priority In signal is active.

Masked programmed interrupt vector addresses provide a 16-bit address whenever an interrupt from the INTERRUPT CONTROL block is serviced; this address is pushed into the PC0 of all PSUs and MIs in the F8 system, forcing the system to execute the sequence of instructions located there. Fifteen bits of the interrupt vector address are fixed (see Table 7-1). Bit 7 of the interrupt vector is set by the INTERRUPT CONTROL block to 0 if the timer interrupt is enabled or 1 if external interrupt is enabled.

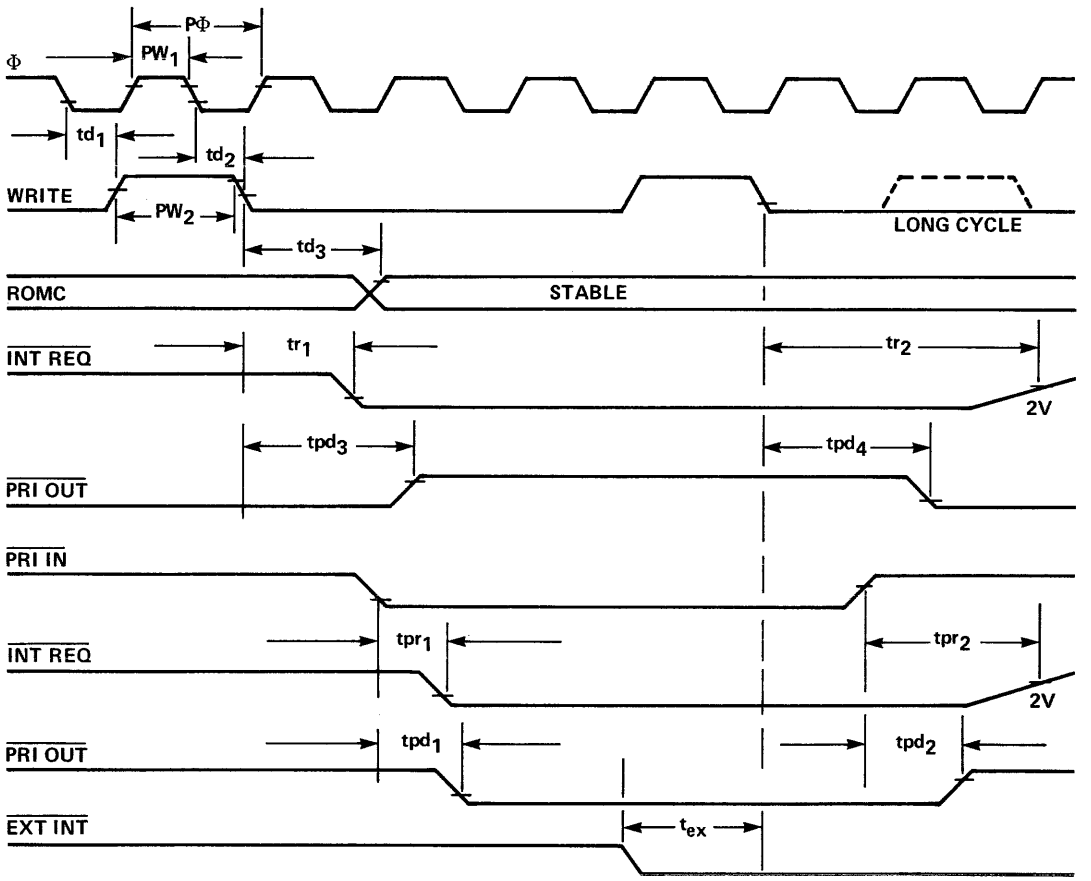
The interrupt logic block is programmed by output instructions to the interrupt control register (port H'06', H'0A', H'22', or H'26' as appropriate). Only the least significant two bits are used; their interpretation is as follows:

CONTENTS OF INTERRUPT CONTROL REGISTER	INTERPRETATION
B'xxxxxx00'	Disable all interrupts
B'xxxxxx01'	Enable external interrupt, disable timer interrupt
B'xxxxxx10'	Disable all interrupts
B'xxxxxx11'	Disable external interrupt, enable timer interrupt

In the above interrupt control register contents definitions, x represents "don't care" binary digits.

A complete discussion of the interrupt logic, of how it responds to interrupt inputs, and of the interrupt acknowledge sequences is found in the discussion of the PSU—Section 3.7.1.

Timing for signals associated with the 3861 PIO interrupt logic is shown in Figure 7-6.



SYMBOLS ARE DEFINED IN TABLE 7-5

Note:

Timing measurements are made at valid logic level to valid logic level of the signals referenced unless otherwise noted.

Figure 7-6. Interrupt Logic Signals's Timing

8.0 (Pending)

9.0 (Pending)

10.0 (Pending)

11.0 3850 CPU-3851 PSU Systems

Simple yet powerful microcomputer systems can be configured out of a 3850 CPU plus one or more 3851 PSU devices. There are some differences in the design considerations that apply to systems with one, or more than one 3851 PSU. Therefore this chapter begins by describing a simple system, including one 3851 PSU, then examines design techniques which only become meaningful in multiple 3851 PSU configurations.

11.1 SINGLE 3851 PSU CONFIGURATIONS

A simple F8 system consisting of one 3850 CPU and one 3851 PSU is illustrated in Figure 11-1. The most significant feature of this configuration is that it presents a very simple interface to external logic.

Apart from the standard external crystal, power and ground, the only signals and timing of concern to a logic designer are the four I/O ports, external interrupts and external reset.

The normal address space for the 3851 PSU will be 0000_{16}-$03FF_{16}$. The fact that the $\overline{\text{EXT RES}}$ input resets the program counter contents to 0 makes this address space assignment a logical mask choice.

The 3851 PSU I/O ports may be assigned any addresses other than 0 and 1, which are reserved for the CPU I/O ports. The four 3851 PSU I/O ports must have sequential addresses. The low order two bits of the first address in the sequence must be 00.

11.1.1 i/O in Single 3851 PSU Configurations

Data input or output via I/O ports must conform to timing specifications given in Figure 2-13 of Section 2 and Figure 3-7 of Section 3. In most cases, external logic treats I/O data asynchronously, and timing given in these figures is used only to calculate delays. Special applications may need to synchronize external logic to the F8 microcomputer system when accessing I/O ports; the WRITE control signal may be used for this purpose.

The logic interface at I/O port pins is described in Section 3.4.3.

Recall that input data is ORed with any information that is currently in an I/O port. For this reason, before data is input to an I/O port, program steps must clear the I/O port by writing 0 into it.

A more comprehensive I/O capability may be achieved by discriminating between data and status on input, plus data and control on output. Although an endless variety of combinations are possible, consider the I/O port pin assignments illustrated in Figure 11-2. Pure data is input and output through

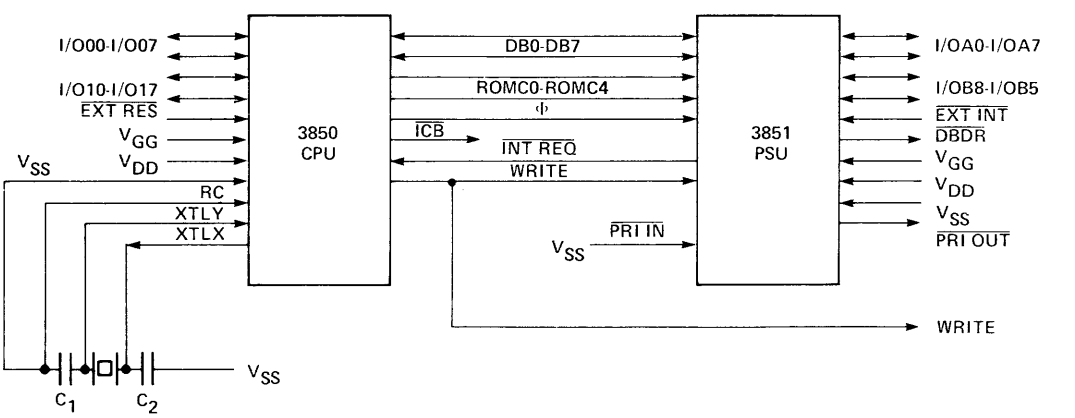


Figure 11-1. A Basic, Two-Device F8 Configuration

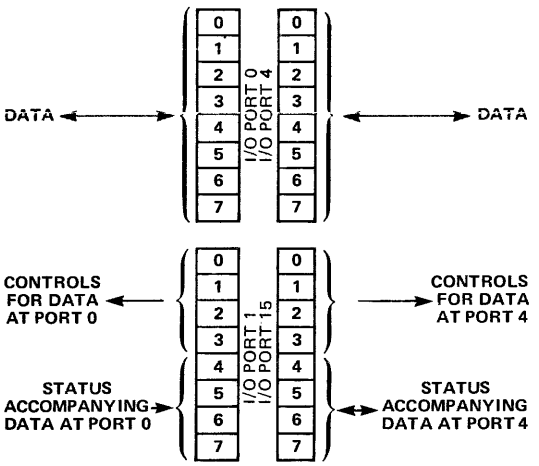


Figure 11-2. I/O Ports Divided into Two Subsystems for a Two-Device F8 Configuration

I/O ports 0 and 4. I/O ports 1 and 5 are assigned to provide control and status for the data channels assigned to I/O ports 0 and 4, respectively. Bits 0, 1, 2 and 3 become control signals identifying the way in which external logic must interpret accompanying data output; or alternatively these control outputs may be used as enable signals, either to precipitate data input, or for pure control functions that are not accompanied by a data transfer in either direction. Bits 4, 5, 6 and 7 are used by external logic to provide information describing the nature of data being input via associated I/O ports, or simply to describe external logic conditions.

Special instruction sequences are not needed when I/O ports are used as illustrated in Figure 11-2. IN (or INS) and OUT (or OUTS) instructions input or output data via any I/O port. Instruction sequences must interpret bit contents appropriately, to conform to any logic pattern or assignment of input and output bits that may arbitrarily be selected. For a port where some bits are used for input and other bits for output, output instructions to that port should have logic 0 in the bits of the output data byte that correspond to pins being used for input. Input instructions will access both the bits assigned to input and the current contents of the output latches of those bits assigned to output; the input instruction does not alter the contents of the output latches.

Apart from the fact that controls are output by the CPU whereas status is input to the CPU, the key conceptual difference between controls and status is that external logic can sense control signal level changes the moment they occur and can respond to them accordingly. Status input by external logic to an I/O port will not be sensed until an IN (or INS) instruction is executed by the 3850 CPU.

Another version of I/O port utilization is illustrated in Figure 11-3. In this case I/O ports are connected to unidirectional lines and thus each I/O port is dedicated to data input or data output. Observe that I/O port 1 is used for control and status information, but the control and status now applies to the remaining three ports. The assignment of I/O ports is, of course, completely arbitrary and in reality any I/O port could be assigned for any service.

Note that there is no reason why one I/O port must be used for both control and status information, nor is there any reason why any I/O ports must be reserved for data transfer. An entire I/O port may be used to output control signals while another entire I/O port may be used to input status. Moreover, either controls, or status, or both may be absent. There is no fundamental difference between data input and status input, or between data output and control output; differences result

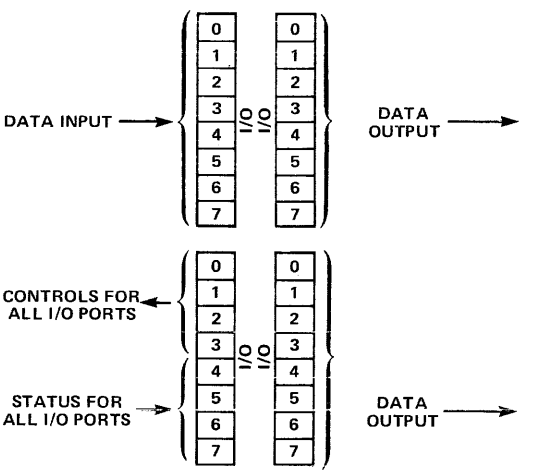


Figure 11-3. I/O Ports Divided into Three Unidirectional Data Ports and One Control/Status Port for a Two-Device F8 Configuration

entirely from external logic acting in concert with the program which is executed out of the 3851 PSU.

11.1.2 Connections to the Data Bus in Single 3851 PSU Configurations

$\overline{DBDR}$ is an active low signal which indicates that a PSU is driving the data bus and is used in F8 configurations where 3851 PSUs are competing with other logic for data bus access.

In a simple two-device system as illustrated in Figure 11-1, it is unlikely that external logic will access the data bus and therefore $\overline{DBDR}$ will remain unused. The only possible other use of $\overline{DBDR}$ in a two-chip system would be if the CPU and PSU were physically separated by a large distance such that the data bus lines had more than 100 pf loading on them. In this case $\overline{DBDR}$ would be used to enable buffers on these lines as discussed in Section 11.2.

For simple data transfer, external logic can connect to the I/O ports, so a direct connection to the data bus must be justified by special considerations that preclude the use of I/O ports. An external connection to the data bus is complicated by the fact that within a simple system as illustrated in Figure 11-1, memory addresses are never output on any bus in the normal course of events, since all memory addressing logic is internal to the 3851 PSU. External logic therefore cannot identify addresses in any way. The ROMC state signals are the only means available to external logic to identify events on the data bus, and only a limited level of identification is provided by the ROMC state signals.

11.1.3 Interrupt Processing in Single 3851 PSU Configurations

The $\overline{EXT\ RES}$ and $\overline{EXT\ INT}$ signals are available to interrupt the F8 system. $\overline{EXT\ RES}$, you will recall, resets the entire system, causing program execution to restart with the instruction stored at memory location 0. This signal is likely to be used by the mechanism which initializes program execution.

$\overline{EXT\ INT}$ is available for an external device, or external devices to request an interrupt. There are two features of interrupt processing which are worth examining in a simple two-device system: the separation of external interrupt requests from programmable timer interrupt requests, and the ability to handle multiple external interrupt requests.

Recall that 3851 PSU interrupt control logic requires a program to select either the programmable timer, or an external interrupt as enabled at any given time; but both interrupts cannot be enabled simultaneously. In small systems that are not using the programmable timer, there is no problem, since external interrupts will be enabled and the programmable timer interrupt request will be disabled. If the programmable timer is being used, then instructions must be executed to sequentially enable and disable programmable timer interrupts or external interrupts. It is conceivable that the programmable timer will time out while timer interrupts are disabled and external interrupts are enabled. The timer interrupt request will be maintained until acknowledged, while the timer starts counting down to time out again.

Therefore, if the timer times out while timer interrupts are disabled, program logic can still effectively use "greater than or equal to" timing logic.

If external interrupts are enabled in a single PSU F8 configuration, handling multiple external interrupts is very straightforward. Logic is illustrated in Figure 11-4.

Every external device will generate an external interrupt request signal which makes a negative transition (high to low) when active. These signals will then be combined via an AND gate to generate INTO shown in the Figure. At the same time these signals are encoded by some logic to produce signals INT1 through INT6.

The PSU in Figure 11-4 will have its programmable timer interrupt permanently disabled, and its external interrupt permanently enabled. Thus any external device requesting an interrupt will cause program execution to vector to the 3851 PSU's external interrupt address vector.

Three aspects of the scheme illustrated in Figure 11-4 need further discussion:

1. How does the CPU identify which external device is requesting an interrupt?

2. What will the interrupt response time be?

3. How are external device interrupt priorities determined?

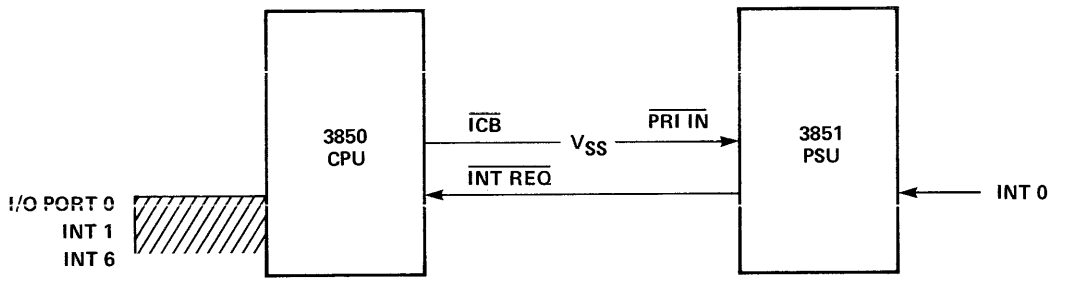


Figure 11-4. Handling Multiple Interrupts in a Two-Device F8 Configuration

In order to identify the source of the interrupt, external encoding logic will generate INT1 through INT6 high as follows:

EXTERNAL DEVICE NUMBER	INT1	INT2	INT3	INT4	INT5	INT6
0	0	0	0	0	0	0
1	1	0	0	0	0	0
2	0	1	0	0	0	0
3	1	1	0	0	0	0
4	0	0	1	0	0	0
5	1	0	1	0	0	0
6	0	1	1	0	0	0
etc.						

The input to I/O port 0 will be created as follows:

I/O Port 0 Pin:	0	1	2	3	4	5	6	7
Input:	0	0	INT1	INT2	INT3	INT4	INT5	INT6

Inputs to the 3850 CPU I/O port 0 may now be interpreted as follows:

I/O PORT 0 CONTENTS	INTERRUPT SOURCE
0	External Device 0
4	External Device 1
8	External Device 2
C	External Device 3
10	External Device 4
14	External Device 5
etc.	

Following any interrupt acknowledge, program execution branches to the external interrupt vector address which is a permanent feature of the 3851 PSU. Beginning at this address, the following instruction sequence reads the contents of I/O port 0 to determine which device was requesting an interrupt, then branches to the appropriate interrupt service routine as follows:

```
IAV     EQU  ?          EQUATE IAV TO THE EXTER-
                        NAL INTERRUPT ADDRESS
                        VECTOR

        ORG  IAV

*HERE PLACE INSTRUCTIONS TO SAVE CONTENTS OF
 ACCUMULATOR,

*ISAR, DATA COUNTER, STATUS, OR OTHER REGISTERS
 THAT NEED

*TO BE SAVED.

        LIS  0          CLEAR I/O PORT 0
        OUTS 0
        INS  0          INPUT INTERRUPTING DEVICE ID
        DCI  BTBASE     LOAD BRANCH TABLE BASE
                        ADDRESS INTO DC0
        ADC             ADD INTERRUPTING DEVICE
                        ID
        LR   Q,DC       MOVE DC0 CONTENTS TO
                        PC0 TO FORCE BRANCH
        LR   P0,Q
BTBASE  JMP  ADDR0      BEGINNING OF BRANCH
                        TABLE
        NOP             ADDR0 ETC. ARE LABELS OF
        JMP  ADDR1      ROUTINES SERVICING
                        INDIVIDUAL INTERRUPTS

        NOP

        JMP  ADDR2      SINCE EACH JMP INSTRUCTION
        NOP             OCCUPIES 3 BYTES, A NOP MUST
        JMP  ADDR3      FOLLOW TO USE UP THE
                        FOURTH BYTE

        NOP
        ETC.
```

It is quite conceivable that a simple two-device system may be used for switching logic in a configuration where a number of external devices are requesting interrupts. The method illustrated in Figure 11-4 allows 64 devices to request interrupts.

Interrupt response time is made up of the delay while the PSU generates an interrupt request to the CPU, plus the time taken by the CPU to respond to the interrupt request.

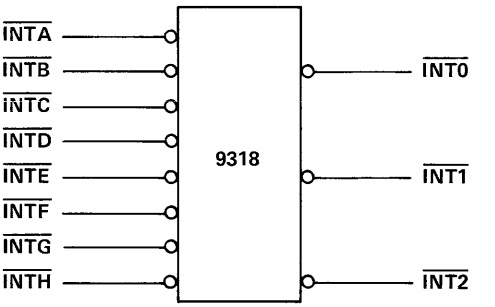
Figure 11-5 is a state diagram showing how to compute the delay between an interrupt request arriving at a PSU, and the start of instruction execution within the interrupt service routine.

The time taken by the CPU to respond to its interrupt request will depend upon programming considerations, but in simple systems we may conclude that nested interrupts are not allowed, that is, while one interrupt is being serviced all other interrupts are disabled. And further, we may assume that each interrupt is serviced by a relatively simple routine which performs some elementary logic sequence without a great deal of complication. This being the case, the following interrupt service routine would be sufficient to save registers and status for any background program before an interrupt is serviced:

```
        ORG     PTIAV
        LR      8,A             SAVE ACCUMULATOR CONTENTS
                                IN BYTE 8
        LR      A,IS            SAVE ISAR IN BYTE 7
        LR      7,A
        LR      J,W             SAVE STATUS IN BYTE 9
        LR      H,DC            SAVE DC IN BYTES 10 AND 11
*BYTES 7 THROUGH 11 OF THE SCRATCHPAD CANNOT BE
*USED BY THE FOLLOWING INTERRUPT SERVICE ROUTINE.
        LIS     0               CLEAR PORT 0
        OUTS    0
        INS     0
        DCI     BTBASE
        ETC.
```

It is possible that a simple two-device F8 system being used in a switching network has no background program, but rather sits in an elementary wait loop, simply responding to interrupt requests. In this case, following each interrupt request, interrupts will be disabled, a service routine will be executed, then interrupts will be re-enabled. This situation demands no protocol to save contents of registers and status before an interrupt service routine is executed, since when interrupts are enabled, nothing of importance is occurring within the microcomputer system.

External interrupts can have their priorities set by external logic using a simple MSI chip such as the 9318 Priority Encoder:



11.2 MULTIPLE PSU CONFIGURATIONS

It is possible for an F8 microcomputer system to consist of a 3850 CPU with up to 64 3851 PSUs.

As illustrated in Figure 11-6, a system bus must be generated, consisting of the data bus lines, the ROMC state lines, the timing signals Φ and WRITE, plus the interrupt request and data bus drive signals $\overline{\text{INT REQ}}$ and $\overline{\text{DBDR}}$. The latter two signals are shown in Figure 11-6 as separated from the system bus, but this is for clarity only.

The system bus does not need to be buffered if less than 100 pf of capacitance is present. If significantly more than 100 pf is present on the data bus lines, then some form of buffering will be required. One possibility is illustrated in Figure 11-7. Observe that control and clock lines are trivial to buffer since they are unidirectional. Note that so long as one of the PSUs has an address space of 0000_{16} through $03FF_{16}$, the address spaces assigned to other PSUs is flexible, providing that address spaces do not overlap. I/O port addresses, likewise, may be assigned at will, again providing they do not overlap.

11.2.1 I/O in Multiple 3851 PSU Configurations

When moving from single PSU configurations, as described in Section 11.1.1, to multiple PSU configurations, the only major new consideration is that the multiple PSU configuration is likely to be very rich in I/O ports. This being the case, there will be a tendency to wire I/O ports for single

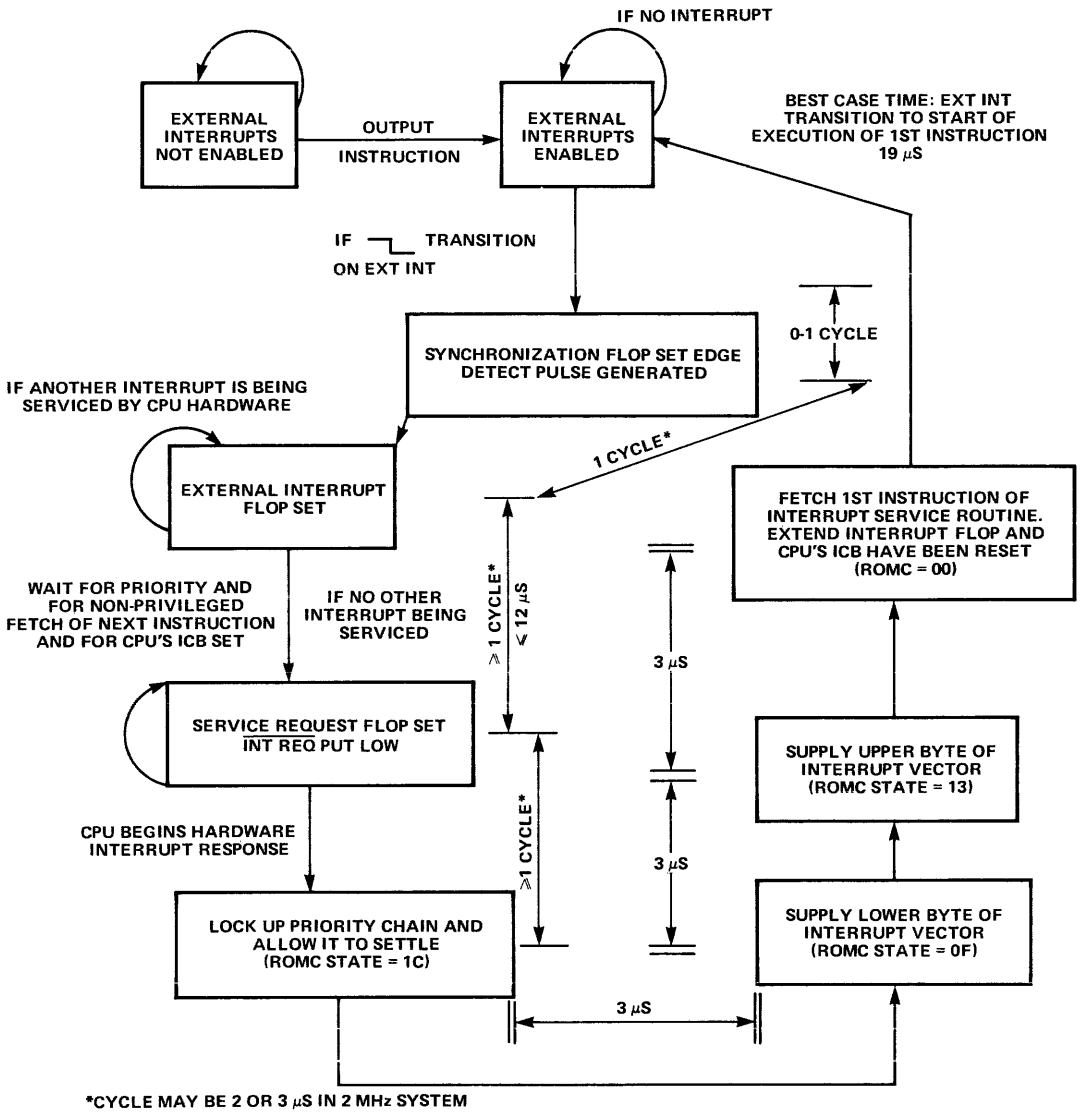


Figure 11-5. A Time State Diagram for the 3851 PSU External Interrupt Request Logic

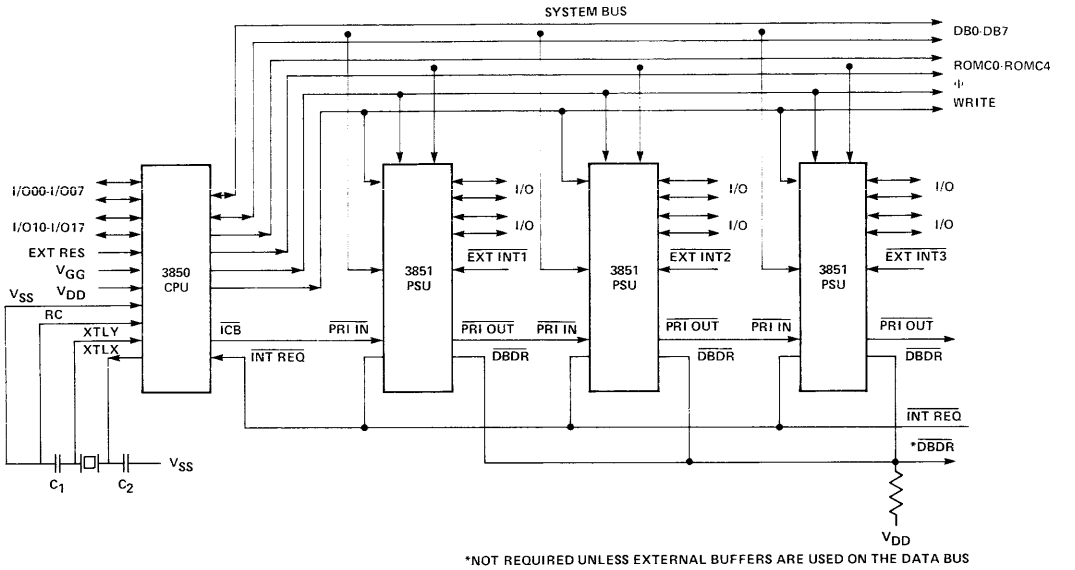


Figure 11-6. A Multi PSU F8 Configuration

direction data transfers as illustrated in Figure 11-4, since unidirectional I/O ports require simpler external logic.

11.2.2 Connections to the Data Bus in Multiple 3851 PSU Configurations

Notice that the data bus drive signals $\overline{DBDR}$, output by each 3851 PSU, may be wired-ORed to generate a system $\overline{DBDR}$. There is, however, no more likelihood of external logic directly connecting to the data bus in a multiple PSU configuration than there is in a single PSU configuration as discussed in Section 11.1.2. Thus the signal's principal use is to buffer the F8 data bus lines whenever a capacitance load significantly greater than 100 pf is present.

11.2.3 Interrupt Processing in Multiple 3851 PSU Configurations

If an F8 configuration that consists of one 3850 CPU plus a number of 3851 PSUs is implementing a very computational intensive application, then servicing a large number of interrupts with one CPU is inefficient. This is because more complex computations make greater use of scratchpad memory and programmable registers; therefore excessive protocol is needed following an interrupt acknowledge, in order to save transient data prior to executing an interrupt service routine. This is particularly true in light of the fact that without any RAM memory present, only 64 bytes of scratchpad RAM are available to the system. Handling numerous interrupts presents no problems providing interrupts are processed one at at time, and do not require extensive entry and exit protocol.

If a multiple PSU configuration is using one or more programmable timers, the most effective way of organizing priorities is to execute timer logic in the highest priority PSU, that is, the ones electrically closest to the CPU, and to leave these PSUs with the external interrupt permanently disabled while the programmable timer is permanently enabled. Lower priority PSUs will have external interrupts permanently enabled and programmable timers permanently disabled.

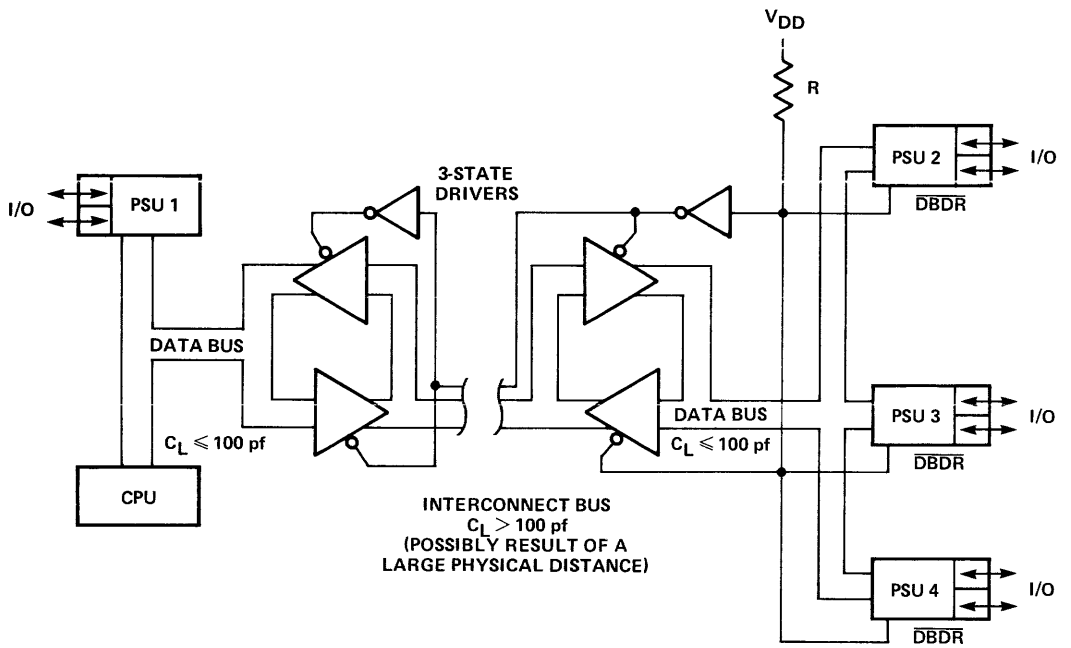


Figure 11-7. A Buffered System Bus

In a multiple PSU system (or one with multiple 3861 PIO devices) where the interrupts of some devices will never be used, the $\overline{\text{PRI IN}}$ and $\overline{\text{PRI OUT}}$ signals of those devices should be removed from the priority daisy chain and $\overline{\text{PRI IN}}$ should be tied to V_{DD} (+5V). Tieing $\overline{\text{PRI IN}}$ high will disable the interrupt circuitry of that device so that it will not interface. The interrupt control register of every device in the interrupt priority daisy chain must be loaded to the desired operation —disabled, external interrupts enabled, or timer interrupt enabled—before initially enabling the 3850 CPU for interrupt servicing. All must be loaded, as the reset sequence does not initialize the interrupt control register.

12.0 F8 Configurations that Include 3852 DMI and 3853 SMI Devices

F8 CONFIGURATIONS THAT INCLUDE 3852 DMI AND 3853 SMI DEVICES

Standard ROM and/or RAM memory may be included in an F8 configuration, controlled either by a 3852 DMI or a 3853 SMI device. Either device can address up to 65,536 bytes of memory, which is the maximum that can be addressed by one 3850 CPU. ROM and/or RAM, controlled by either a 3852 DMI or a 3853 SMI, may be mixed freely with 3851 PSUs, providing the total bytes of memory do not exceed 65,536.

If dynamic RAM devices are being used, or if direct memory access is required, the 3852 DMI should be used. Although the 3852 DMI can also control static RAM, the 3853 SMI is preferred, since it provides interrupt logic and a programmable interrupt address vector.

Simple memory interface logic is almost identical in the 3852 DMI and the 3853 SMI devices. Both devices identify data ready to be output from memory using CPU READ. $\overline{\text{RAM WRITE}}$ identifies a write-to-memory. The 3852 DMI uses CPU SLOT, in addition to CPU READ, to strobe data read from memory into bus latches; this is necessary since the 3852 DMI divides each machine cycle into multiple memory access cycles. The 3852 DMI also generates CYCLE REQ to serve as the dynamic memory clock signal, identifying the start of each memory access cycle.

The logic differences associated with using a 3852 DMI or a 3853 SMI are less significant than the logic differences associated with connecting small or large memories to a 3850 CPU. This section therefore begins by describing small configurations, then follows with a description of large configurations.

12.1 SMALL CPU-RAM CONFIGURATIONS

Small CPU-RAM configurations are likely to use the 3853 SMI device since small dynamic RAMs are not economical. The 3852 DMI is therefore not illustrated in this sub-section.

12.1.1 Small CPU-RAM Configurations without a PSU

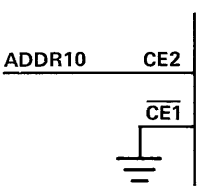
Figure 12-1 illustrates a very simple F8 configuration, consisting of a 3850 CPU, a 3853 SMI and 256 bytes of RAM provided by a single, 256 X 8-bit, 3539 memory device.

10K pull-ups are needed on the data bus lines in order to make the output one level from the RAM MOS compatible.

A 3539 memory is selected by $\overline{\text{CE1}} \cdot \text{CE2}$ true.

The two chip select signals, $\overline{\text{CE1}}$ and CE2, have been tied to ground and V_{DD}, respectively, so that the 256 bytes of RAM will respond to memory addresses 0000_{16} through $00FF_{16}$. This is, of course, unrealistic since the F8 system requires programs to be origined at memory location 0, therefore if ROM is present in a system, it will occupy the lowest memory address space. In real applications RAM address space would be defined by creating chip selects out of the higher eight address lines ADDR8 through ADDR15.

A system that includes one 3851 PSU, with address space 0000_{16}-$03FF_{16}$, and one 3539 RAM, with address space 0400_{16}-$04FF_{16}$ would generate 3539 RAM chip selects as follows:



The data lines DB0 through DB7 are bi-directional and would normally require a three state buffer interface with memory, but this is unnecessary with a 3539 RAM which has an output disable function, driven by the CPU READ output of the 3853 SMI; therefore the I/O pins of the RAM device may be connected directly to the data bus, without buffering.

If an F8 microcomputer system is operating with the maximum clock rate of 2 MHz, then the maximum access time for RAM is 900 nS and the $\overline{\text{RAM WRITE}}$ pulse width a minimum of 350 nS.

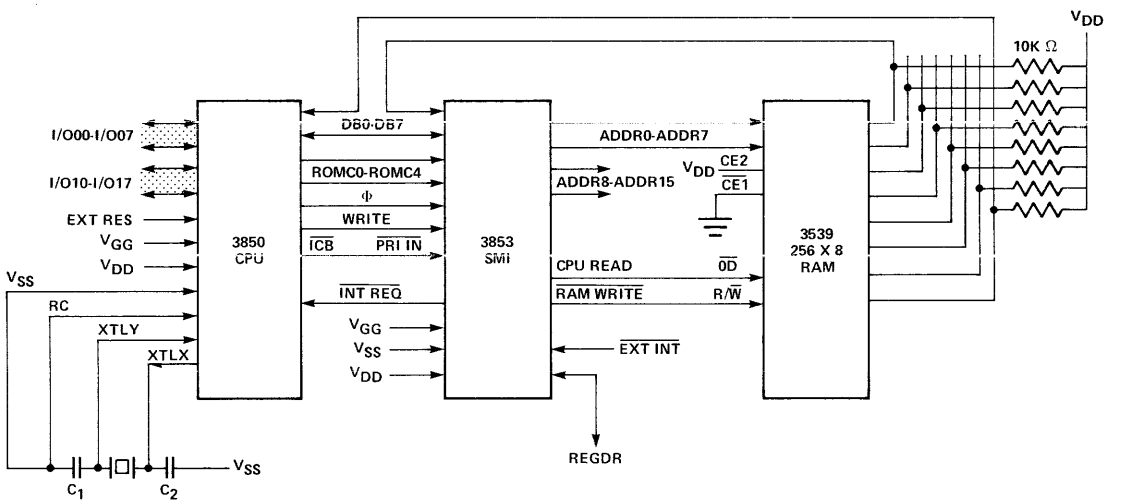


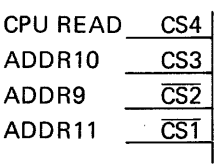
Figure 12-1. Interfacing RAM using a 3853 SMI

12.1.2 Small CPU-RAM Configurations with a PSU

Figure 12-2 illustrates a simple F8 configuration that includes a 3850 CPU, a 3851 PSU, plus a 3853 SMI connected to 1024 bytes of PROM and 256 bytes of RAM. Figure 12-2 is largely self-evident in that it represents an elementary extension of the logic illustrated in Figure 12-1. Memory address spaces have been defined as follows:

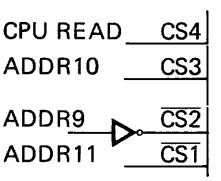
3851 PSU: 0000_{16}-$03FF_{16}$
Selected by device select mask.

93448 PROM A: 0400_{16}-$05FF_{16}$
Selected as follows:



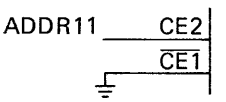
$\overline{CS1}\cdot\overline{CS2}\cdot CS3\cdot CS4$ = Select

93448 PROM B: 0600_{16}-$07FF_{16}$
Selected as follows:



$\overline{CS1}\cdot\overline{CS2}\cdot CS3\cdot CS4$ = Select

3539 RAM: 0800_{16}-08FF
Selected as follows:



$CE2\cdot\overline{CE1}$ = Select

12.1.3 Connecting the REGDR Input

As described in Sections 4 and 5, 3852 DMI and 3853 SMI devices each have two associated address spaces.

Memory that is controlled by one of these memory interface devices responds to a range of memory addresses which is identified by chip select logic. This address space applies to instructions which read data out of memory or write data into memory.

There are an additional set of instructions which read data out of memory address registers or write data into memory address registers. However, memory address registers are duplicated within the memory devices of an F8 system. Therefore each device that contains memory address registers must have a unique address space within which its address registers alone will respond to instructions that access address registers.

The address registers' address space for a 3851 PSU coincides exactly with the address space for

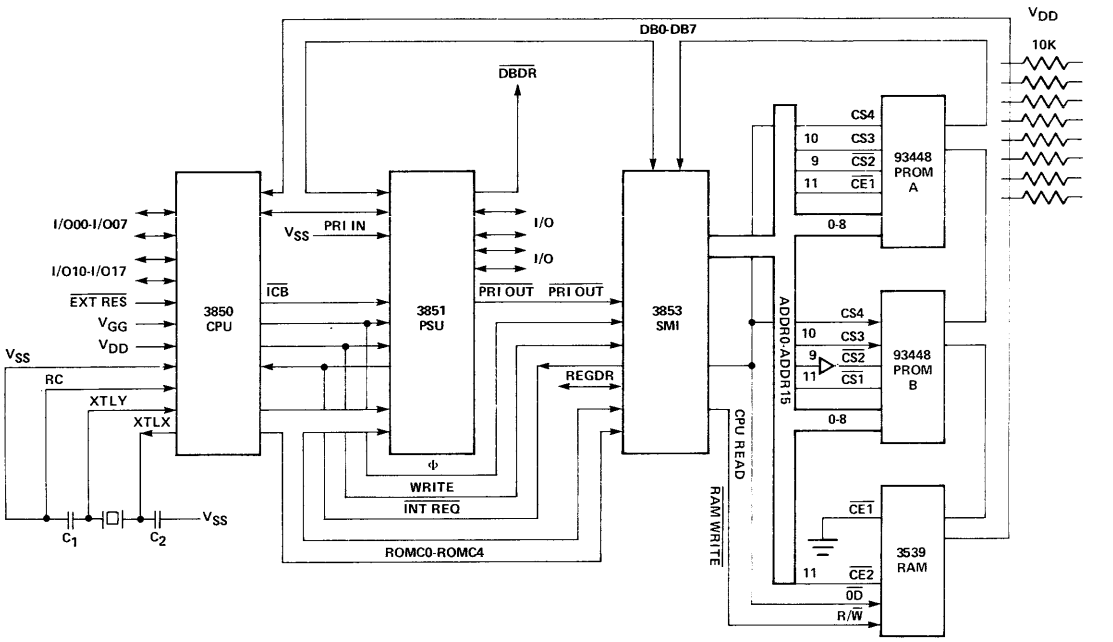


Figure 12-2. Interfacing ROM and RAM using a 3853 SMI

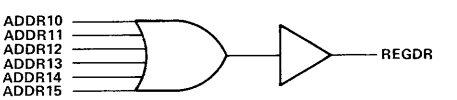
the PSU memory. This is a characteristic of the 3851 PSU and there is nothing that a logic designer can do about it.

The address registers' address space for the 3852 DMI and 3853 SMI devices are set by REGDR.

It is a good idea to create REGDR true for all memory addresses that are not part of a 3851 PSU's address space. This ensures that instructions which attempt to access address registers do not end up reading invalid data, as would happen if the current contents of the address register are outside the address space of any memory device.

Whenever a ROMC state specifies an address register access operation, the contents of the address register to be accessed is output on the address lines of the DMI or SMI device. Since the address register contents appears on the address lines, REGDR can be decoded from the address lines in order to determine whether the device is, or is not selected. Refer now to Figures 12-1 and 12-2. Figure 12-1 is not a realistic F8 configuration as illustrated, but it is certainly conceivable to have F8 configurations where all memory consists of ROM and/or RAM devices connected to a 3853 SMI device. In this case, REGDR would be allowed to go true during all address register access operations. This is accomplished by not connecting any external logic to the pin.

In Figure 12-2, the 3853 SMI device's desired address space is 0400_{16} through $FFFF_{16}$. REGDR could now be created as follows:



The REGDR signal itself must be an open collector gate signal, as illustrated in Figure 12-3.

In systems that include one or more 3851 PSUs, plus a 3852 DMI or a 3853 SMI, REGDR may be created by buffering $\overline{DBDR}$, as follows:

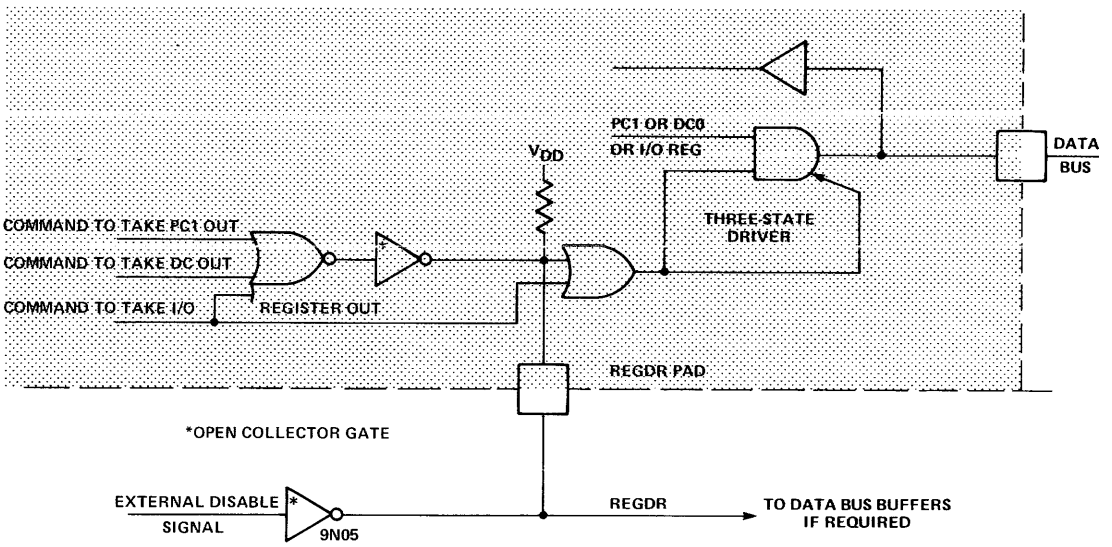
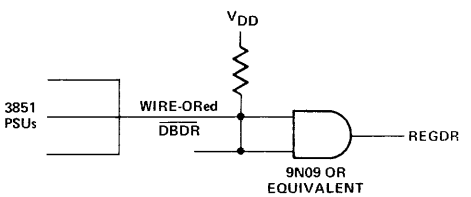


Figure 12-3. REGDR Controls Data Bus Drivers



12.2 LARGE F8 CONFIGURATIONS WITH MIXED MEMORY

Large F8 configurations are defined as configurations which include memory devices which are less than eight bits wide and require buffered data busses, or which include memories such as the 2102 which combine the output driver enable with chip select. Two such configurations are illustrated in Figures 12-4 and 12-5.

In Figure 12-4, an F8 configuration is illustrated with 2K bytes of ROM implemented on two 3851 PSUs and 8K bytes of RAM implemented using 64 2102 memory devices. Each 2102 memory device provides 1024 X 1 bits of memory. Eight 2102 memory devices are therefore required to implement 1K bytes of RAM.

Figure 12-5 illustrates an F8 configuration which includes 4K bytes of ROM implemented on four 3851 PSUs, plus 32K bytes of RAM configured using 64 4096 memory devices. Each 4096 memory device provides 4096 X 1 bits of memory.

The CMOS device labeled 340097 in Figures 12-4 and 12-5 buffer the data bus. Data coming from memory is buffered through the 340097 devices, which have an external control line via which they may be forced into a high output impedance state. This control line is, in turn, driven by the 3852 DMI or 3853 SMI CPU READ signal which is ANDed with an externally derived page select. This page select must be the sum of all memory chip selects. When the data bus line is to be driven by RAM, the DMI or SMI device raises the CPU READ line; and if page select is true, information for RAM is gated onto the data bus. In all other cases CPU READ holds the buffers in a high impedance output state.

The CMOS buffers labeled 34050 in Figures 12-4 and 12-5 will be needed only if the capacitance on each data bus line exceeds 100 pf total, where this total applies to all devices connected to the data bus, not just to the RAM devices controlled by the DMI or SMI. This buffer may be eliminated if the capacitance is less than 100 pf.

Memory address spaces in Figures 12-4 and 12-5 are defined in the usual way using the lower address lines as an address select and the higher

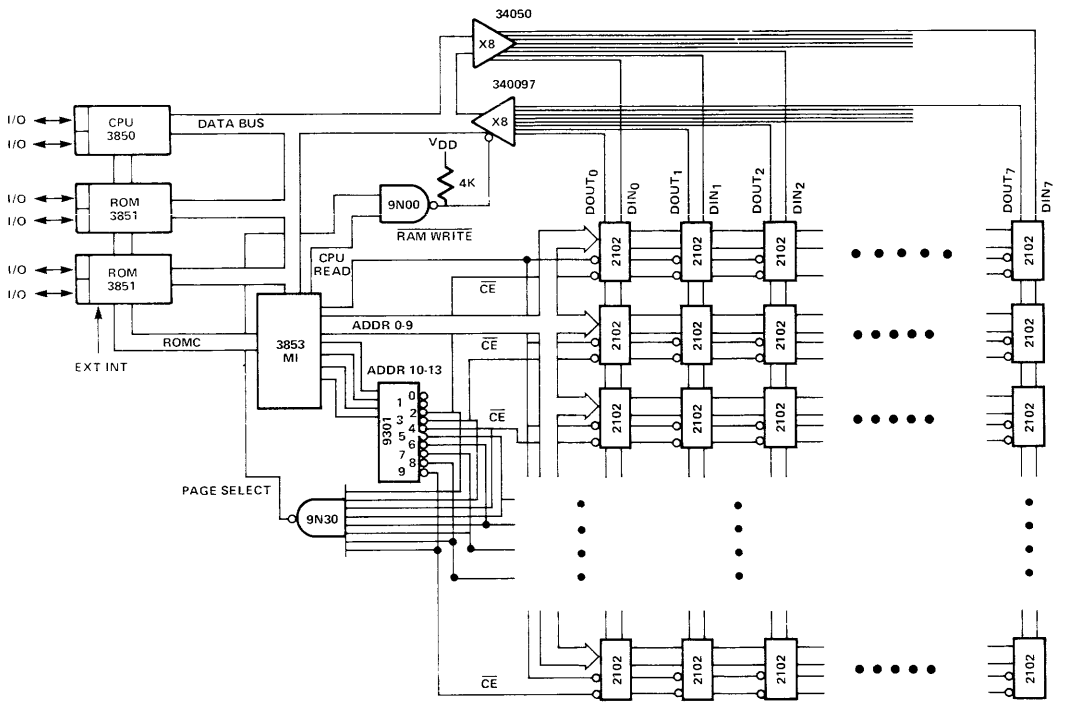


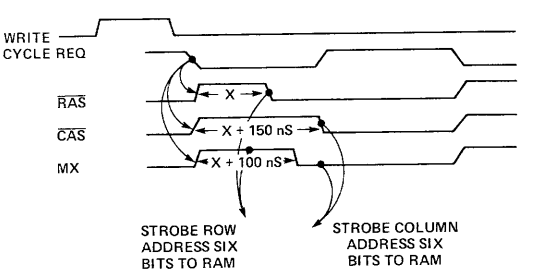
Figure 12-4. 2K ROM and 8K RAM Configuration using 2102 Memory Devices

address lines to generate chip selects with a 9301 1-of-10 decoder.

Memory interfaces are designed to drive a 500 pf load on each address line and the $\overline{\text{RAM WRITE}}$ line. Buffering of these lines is therefore not required.

Figure 12-5 illustrates how the timing signals output by a 3852 DMI device may be used with dynamic memories, to multiplex addresses, and to implement refresh cycles.

Since the 16 pin 4K RAMs are being used in this configuration it is necessary to multiplex the 12 address lines onto the six RAM address lines. This is done using three 9N51 (AOI) chips. CYCLE REQUEST is input to a one-shot (9602) to generate the control gating signals needed for the multiplexing. CYCLE REQUEST is also used to generate the $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ strobes for the RAM chips. The timing for these signals is as follows:



X is the address set-up time for the DMI, and has a worst case, maximum value of 400 nS.

During a normal CPU cycle when information is being fetched from RAM, the information must be

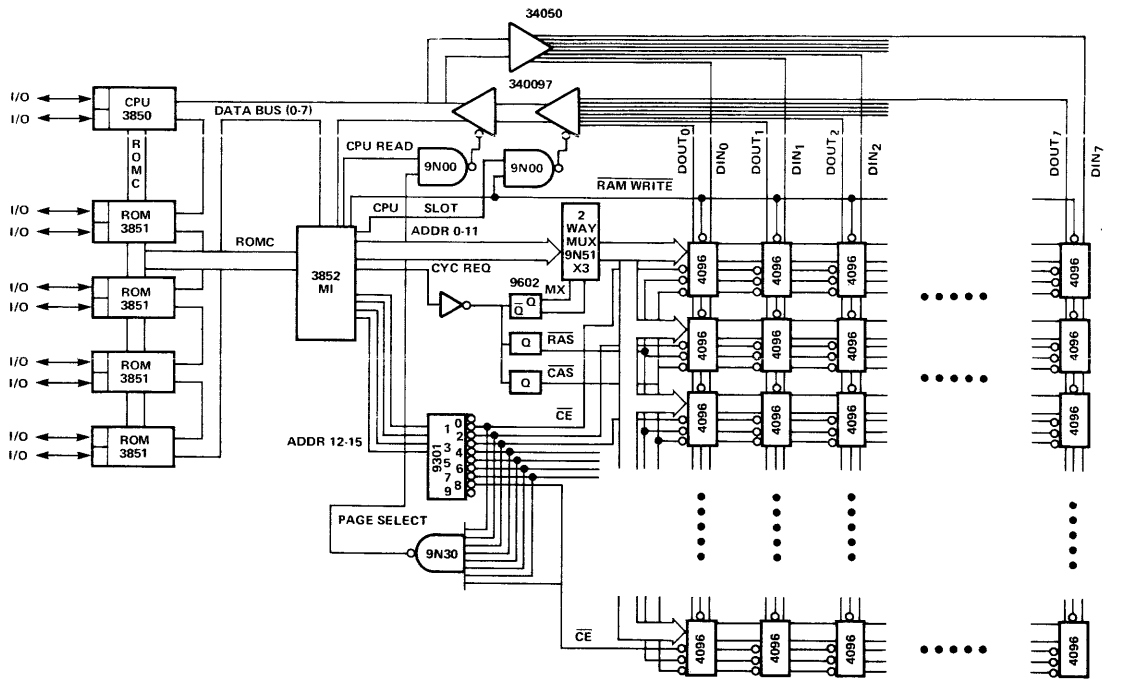


Figure 12-5. 4K ROM and 32K RAM Configuration using 4096 Devices

held on the data bus lines for the entire length of the machine cycle. To divide a machine cycle in half and use the second part for refreshing RAM, the information fetched during the first half of the cycle must be saved. This is accomplished using the CPU SLOT signal and two levels of CMOS buffering (340097's). During the first half of a cycle, CPU SLOT is true and information from RAM is passed onto the node between the two buffers. When the second half of the cycle begins (during which time RAM will be refreshed), CPU SLOT goes low, temporarily latching the RAM information onto the node between the two buffers. This allows the data to remain constant on the data bus lines for the entire length of the cycle. An ordinary latch may also be used instead of the second level CMOS buffer described here.

The system shown in Figure 12-5, operating with a maximum clock rate of 2 MHz, requires a maximum RAM access time of 500 nS.

12.3 INTERFACING VERY LARGE MEMORIES

A technique known as memory bank switching may be employed in an F8 system to expand total system memory beyond the normal 64K bytes. Figure 12-6 indicates one way of implementing such a system. In this figure, RAM is divided into blocks of memory; the size of each block is $\leq$ 63K (leaving a 1K hole for a PSU). All RAM blocks are driven in parallel by the DMI. Data lines into and out of each RAM block are then buffered onto the F8 data bus with control coming from the DMI (CPU READ and CPU SLOT). Each buffer is switched

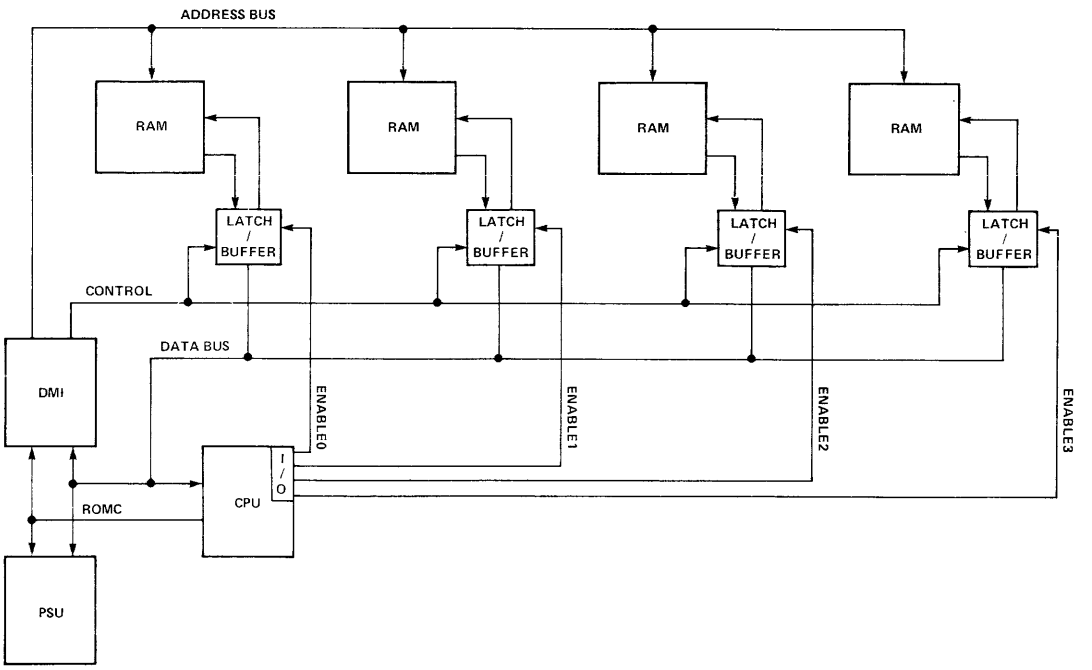


Figure 12-6. Memory Bank Switching

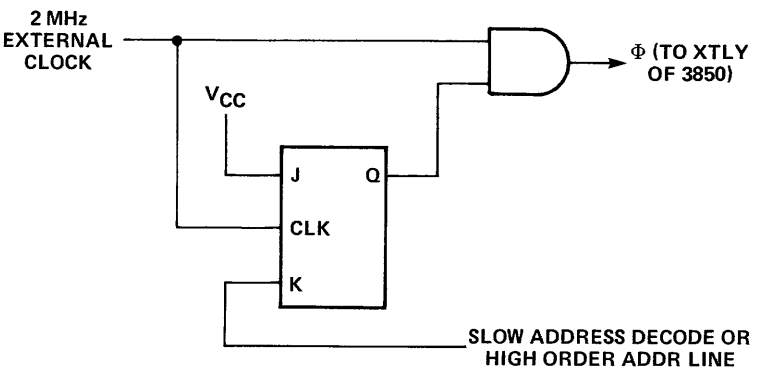


Figure 12-7. Modifying Clock Period under Hardware Control

into and out of the F8 data bus by an enable line. The enable line is generated by an I/O port bit in the CPU. At any given time only one of these enable lines will be true, allowing the CPU to switch in and out any memory bank it desires.

12.4 INTERFACING TO SLOW MEMORIES

The access times required of memories within an F8 system running with a 2 MHz clock are not particularly difficult to achieve using generally available, commercial memory devices. Nevertheless, lower cost, slower memories may be desirable in an F8 system where execution speed is not important.

The easiest way of incorporating slower memories into an F8 system is to use a slower system clock. Tables 12-1, 12-2 and 12-3 show how time periods may be extended for various critical memory signal parameters, as a function of the Φ clock period. In these three tables, PΦ represents clock period and ΔPΦ represents the increase in clock period. For example, if ΔPΦ is 50 nS so that PΦ is 550 nS, then the access time required of a memory interfaced to the 3852 would be 900 + 3 X (50) = 1050 nS instead of 900 nS.

Table 12-1. Required 3853 SMI Memory Characteristics

PARAMETER	PΦ = 500 nS + ΔPΦ
Access Time	900 nS + 3ΔPΦ
Address Set-up Time to WRITE	600 nS + 2ΔPΦ
Data Set-up Time to WRITE	550 nS + 4ΔPΦ
WRITE Pulse Width	350 nS + ΔPΦ
Data and Address Hold Times	350 nS + ΔPΦ

Table 12-2. Required 3852 DMI Memory Characteristics with No DMA

PARAMETER	PΦ = 500 nS + ΔPΦ
Access Time	500 nS + 2ΔPΦ
Address Set-up Time to WRITE	600 nS + 2ΔPΦ
Data Set-up Time to WRITE	550 nS + 4ΔPΦ
WRITE Pulse Width	350 nS + ΔPΦ
Data and Address Hold Times	200 nS + ΔPΦ
READ Cycle Time	900 nS + 2ΔPΦ
WRITE Cycle Time	3 μS + 6ΔPΦ

A simple scheme exists for doubling the clock period, under hardware control, in a system that includes both fast and slow memory. This scheme is illustrated in Figure 12-7. The input K may be generated in any way from the logic which creates chip selects for slow memory devices.

Table 12-3. Required 3852 DMI Memory Characteristics with DMA

PARAMETER	PΦ = 500 nS + ΔPΦ
Access	500 nS + 2ΔPΦ
Address/Data Stable Time	580 nS + 2ΔPΦ

12.5 INTERRUPT PROCESSING IN F8 CONFIGURATIONS THAT INCLUDE ROM AND RAM

The 3853 SMI device has an external interrupt request line and interrupt processing logic, but the 3852 DMI device has no interrupt capability. In a system that includes a 3852 DMI device, therefore, the presence of the DMI device in no way affects interrupt processing and information presented in Sections 11.1.3 and 11.2.3 apply.

The 3853 SMI's interrupt capability differs from the 3851 PSU interrupt logic in these ways:

1. The 3853 SMI has no priority output signal, therefore it must be the last device in the interrupt priority chain or external logic can be used to generate the priority output signal.

2. The 3853 SMI device's interrupt address vector is programmable. While this does not preclude any of the interrupt handling features described in Sections 11.1.3 and 11.2.3, it does make possible some interesting additional features.

The programmable interrupt address vector of the 3853 SMI allows program logic to select the interrupt service routine which will be executed following the next interrupt requested by the 3853 SMI. This is useful in a broad range of control applications where, depending upon circumstances, external logic must follow one of two (or more) policies.

For example, a class of controllers, known as "bang-bang controllers," measure a parameter which is input by external logic and depending upon the value of this parameter, either no output controls or one of two specific types of output control are generated. Figure 12-9 illustrates bang-bang control logic.

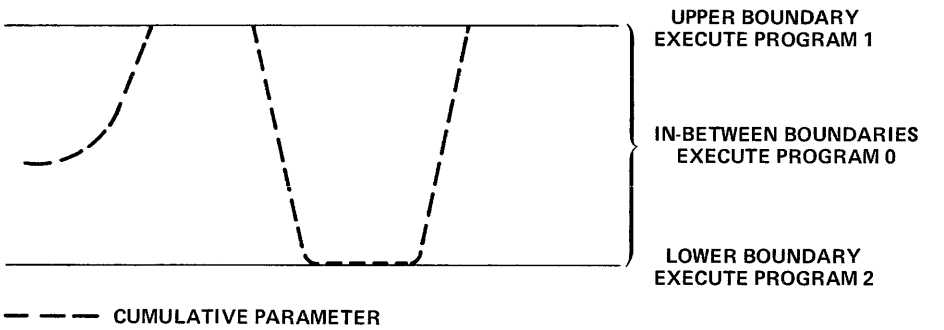


Figure 12-8. Bang-Bang Control Logic

An F8 microcomputer system being used to implement bang-bang control logic will receive the control parameter as an asynchronous input, the arrival of which is indicated by an interrupt request. Three different interrupt service routines will exist, one for each boundary and one for the condition between boundaries. Thus, whenever external logic requests an interrupt and transmits a new parameter value, F8 program logic immediately branches to the interrupt service routine which reflects the current control condition. Following an interrupt program logic determines whether the new parameter has caused a change to a new condition, in which case this change can be reflected by altering the interrupt address vector stored in the 3853 SMI.

13.0 Using Direct Memory Access

USING DIRECT MEMORY ACCESS

The use of direct memory access within an F8 system is one of the most versatile and rewarding challenges facing the logic designer. The key concept to understand is that direct memory access is a totally asynchronous event controlled entirely by signals output by the 3852 DMI and 3854 DMA devices. These signal sequences have been described in Sections 4 and 6, and providing timing is adhered to, external logic can respond directly to this set of well defined signals, completely disregarding the very existence of the 3850 CPU or any executing programs.

13.1 A SIMPLE DMA CONFIGURATION

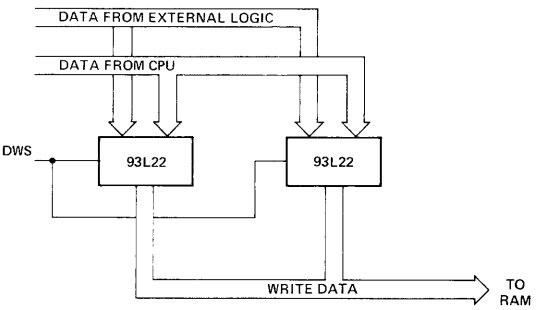
Consider a simple DMA configuration as illustrated in Figure 13-1. This figure shows a 3850 CPU with one 3851 PSU for program storage, plus a RAM array with a 3852 DMI and 3854 DMA providing RAM control and direct memory access. Only signals that are of specific interest to the DMA operation are illustrated in Figure 13-1. Φ and WRITE clock generation, power and ground, for example, are all omitted for clarity.

13.1.1 RAM Array Interface

A description of Figure 13-1 will begin at the RAM array. Observe that the terms "input" and "output" are used with respect to the CPU, or external devices; in other words, "output" constitutes a write-to-memory whereas "input" constitutes a read-from-memory.

Data being output to the RAM array must pass through a multiplexer which receives data either from the CPU or from external devices. Data destined for the F8 system is strobed into a latch by CPU SLOT. Data output by external devices is gated onto the WRITE data bus in response to DWS true. DWS, the AND of DIRECTION and XFER, is equivalent to DWS. DIRECTION and XFER are generated by the 3854 DMA in response to $\overline{\text{XFER REQ}}$ low. Recall that DIRECTION has its condition set under program control, therefore determines whether INPUT STROBE or OUTPUT STROBE will occur. In other words, $\overline{\text{XFER REQ}}$ (together with other internal conditions) cause either INPUT STROBE or an OUTPUT STROBE to be generated, the selection being made by DIRECTION, and the timing being dictated by XFER.

External logic need not concern itself with timing for data arriving at the Two In Mux since OUTPUT STROBE will guarantee that contentions do not arise. There are various ways in which the WRITE DATA bus may be implemented, one way is as follows:



The 83L22 devices are low power quad two-input multiplexers. Each is four bits wide. DWS is used to select between the two data sources.

Input data, that is, data being read from memory, need not be multiplexed. External logic can simply use an input strobe, formed by the AND of $\overline{\text{DIRECTION}}$ and $\overline{\text{XFER}}$, to read data off the input data bus. Observe that the input data bus can be connected to the main F8 data bus via a double buffer. This logic is illustrated in Figure 12-5 and has been described in the accompanying text of Section 12. An alternative to this scheme would be to replace the first buffer (nearest the RAM) with a 93L14 latch which would then be enabled by CPU SLOT also. Once again, external logic neither knows nor cares what the conditions are within the F8 system. As soon as external logic is ready to receive a byte of data, it simply sets $\overline{\text{XFER REQ}}$ low. On the following INPUT STROBE true, external logic can be sure that data on the input bus is the requested data byte.

There are no special requirements needed for connecting the address bus to the 3852 DMI and 3854 DMA devices. The MEM IDLE signal output by the 3852 DMI to the 3854 DMA device ensures that these two devices do not attempt to compete by placing overlapping addresses on the address bus. Therefore, at the RAM array, address logic and

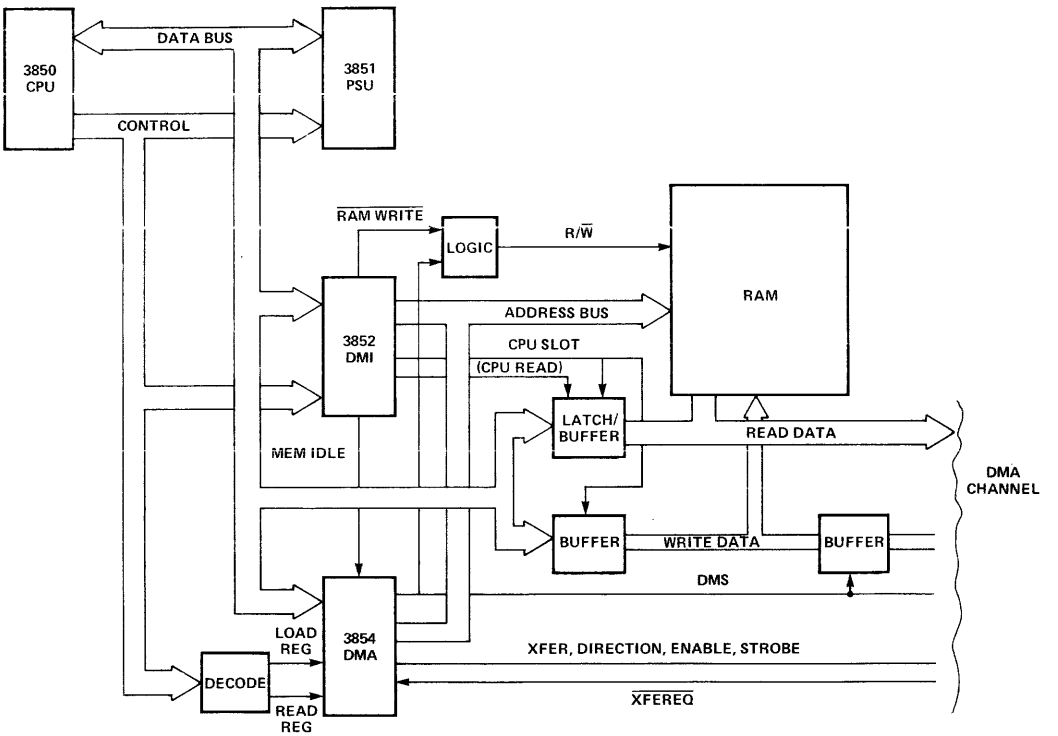
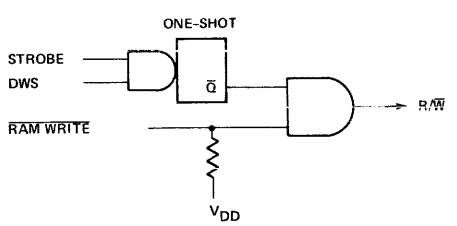


Figure 13-1. A Typical DMA Configuration

chip select logic will be completely standard and will depend upon the size and type of memory being used.

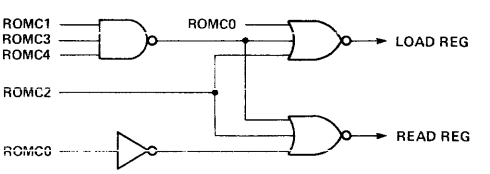
The R/$\overline{W}$ signal required by RAM can be generated by the AND of $\overline{\text{RAM WRITE}}$, output by the 3852 DMI and a combination of the DWS and STROBE outputs from the 3854 DMA. Recall that DWS identifies the write operation whereas STROBE identifies a time period during which the DMA address lines are STABLE (See Figure 6-5). The following is a simple scheme for implementing R/$\overline{W}$:



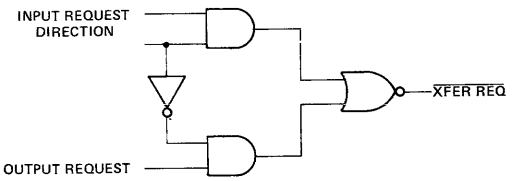
The one-shot is needed to adjust the DMA WRITE pulse width to the particular RAM requirements.

13.1.2 3854 DMA Device Signals

Consider now the 3854 DMA device. Logic which generates inputs to this device are very standard and have been described in Section 6. The following is one scheme whereby the LOAD REG and READ REG control inputs may be generated in response to ROMC states of 1A and 1B, respectively.



The principle input from external logic to the 3854 DMA device is $\overline{\text{XFER REQ}}$. Here is one way in which this signal might be generated out of an input request and an output request:



This logic may or may not be required for a given application depending on the amount of control the CPU had over the external logic.

13.1.3 DMA Timing during Transfer of One Byte

Figure 13-2 shows the state of various control lines and the data content of busses within the Figure 13-1 DMA system during execution of a typical machine cycle.

The cycle shown in Figure 13-2 is a long cycle; it is an access to memory via the DC0 registers as might occur during an LM instruction. At the same time a DMA transfer is taking place, from external logic, to RAM.

In Figure 13-2, the symbol [ ] indicates that the contents of the register identified within the brackets appears on the bus; the symbol [[ ]] indicates that the contents of the memory location pointed to by the contents of the stated register appears on the bus.

Signal delays have been ignored in Figure 13-2 so that the waveforms represent an ideal case.

As indicated at the top of Figure 13-2, the cycle is broken into three access periods. These are the significant aspects of each period:

ACCESS PERIOD I: During this period the PC0 address is forced out onto the address lines by the DMI, as is always the case, resulting in a PC0 access to RAM. Data appears on the READ DATA BUS lines as a result. At the end of this cycle, however, it is learned that this access was wasted because a DC0 access was needed by the instruction.

ACCESS PERIOD II: During period II the DMI places the DC0 contents on the address bus and accesses the RAM location pointed to by it. CPU SLOT remains high indicating that, because its first access was wasted, the CPU requires the use of this period to make its memory access. CPU READ also goes true during this period, enabling the data fetched from ROM onto the F8 data bus and back to the CPU. At the end of this access period, CPU SLOT goes false latching the fetched data onto the storage node between the CMOS buffers where it will remain for the rest of the cycle.

ACCESS PERIOD III: At the start of period III, the DMI drives the MEM IDLE signal high to the DMA indicating that a DMA access may now take place. At the same time the DMI places its address drivers in a high impedance state. The DMA chip now responds by driving the contents of its address registers onto the address bus and outputting a high signal on DWS to enable DMA data into the RAM via the WRITE DATA bus. Toward the end of this access period STROBE is output by the DMA chip which is used to generate the R/$\overline{\text{W}}$ signal into RAM. At the end of this access period the DMA places its address drivers back into a high impedance state and the DMI address drivers again take over at the start of the next cycle.

13.1.4 DMA Timing during a Block Transfer

A DMA transfer operation begins by output instructions loading the ports of the 3854 Direct Memory Access (DMA); the timing of these events is synchronized to the WRITE clock. The transfer proceeds as a function of XFER REQ and MEM IDLE; here the timing of byte transfers is synchronized to MEM IDLE. The transfer concludes either when the buffer length has been decremented to zero or by an output instruction that clears the DMA enable bit (bit 7 of DMA port 3); here the events will be synchronized by MEM IDLE in the first case or by WRITE in the second.

Figure 13-3 gives an overview of signal relations over an entire transfer process. It assumes that bit 4 of DMA port 3 is low so that the external device must make a transfer request before each data transfer. Both alternatives of transfer termination are shown. The timing again assumes the ideal case of no signal delay to provide clarity.

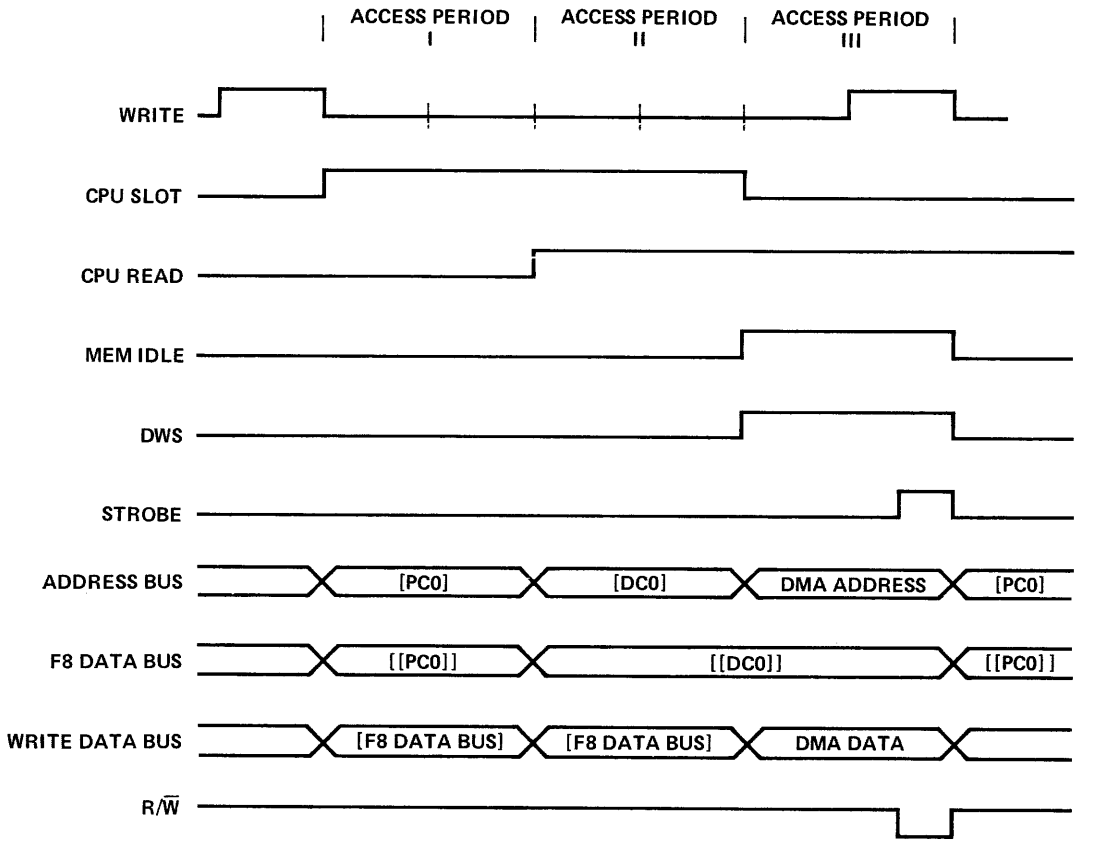


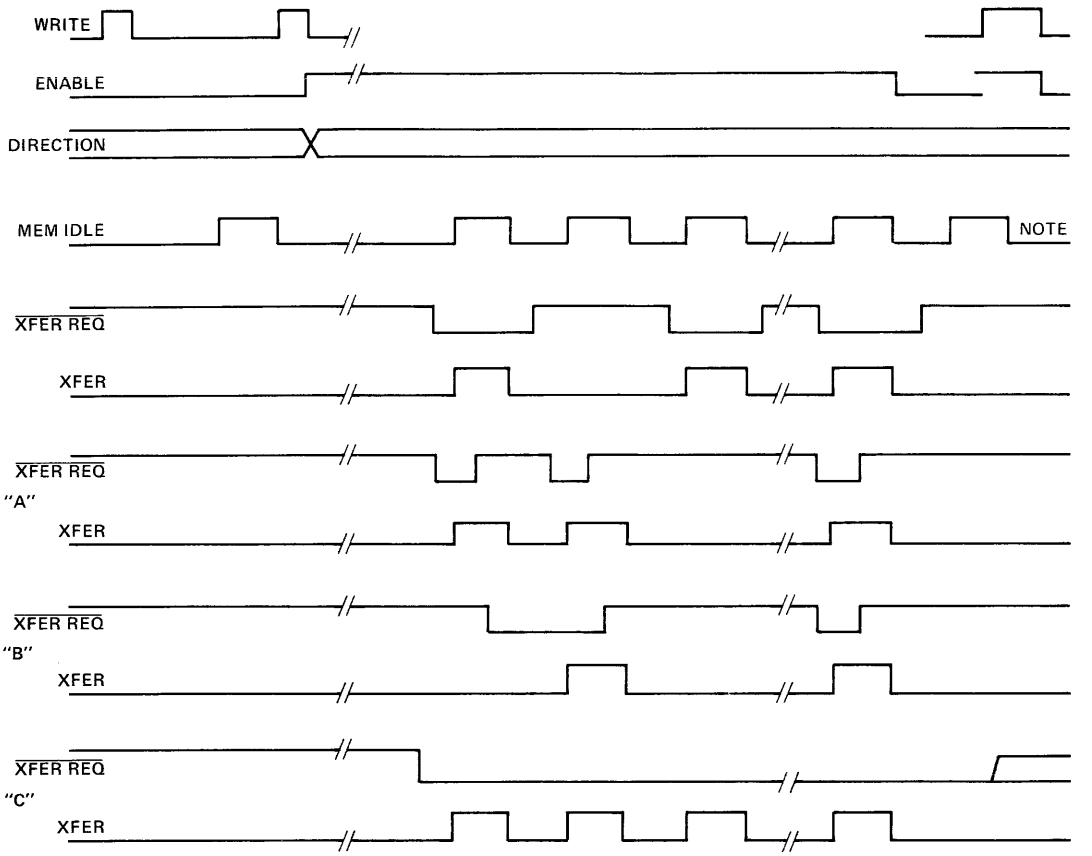
Figure 13-2. DMA Timing during a Typical Cycle

Several different examples of $\overline{\text{XFER REQ}}$ are detailed. These emphasize that $\overline{\text{XFER REQ}}$ can be completely asynchronous to MEM IDLE. The set-up time for $\overline{\text{XFER REQ}}$ that was given in Section 6 is only of importance if a data transfer must occur during a particular instance of MEM IDLE high. "A" of Figure 13-3 illustrates that $\overline{\text{XFER REQ}}$ is not required to remain low throughout the MEM IDLE high period. "B" of Figure 13-3 shows a case of $\overline{\text{XFER REQ}}$ going low to make a request while MEM IDLE is high; the request will be honored during the next subsequent MEM IDLE, provided $\overline{\text{XFER REQ}}$ is still low. "C" of Figure 13-3 is pointing out that if $\overline{\text{XFER REQ}}$ is held low, bytes will be transferred continuously at the maximum rate allowed by MEM IDLE.

$\overline{\text{XFER REQ}}$ can be conveniently implemented with a "handshaking" logic. The external device can asynchronously set $\overline{\text{XFER REQ}}$ to its active low state when the external device requires a byte. $\overline{\text{XFER REQ}}$ would stay low until the DMA honors the request. The rising edge of XFER would clear $\overline{\text{XFER REQ}}$ back to its inactive high state. This sequence is shown by "B" of Figure 13-3.

13.1.5 DMA and Refresh Rates

In the system shown in Figure 13-1, a DMA channel is established between the system RAM and an external peripheral device. In such a system it is frequently desirable to know what the bandwidth available for DMA is and what refresh rates will be maintained should dynamic RAM be used. Table 13-1 presents an analysis of DMA access and RAM refresh rates as a function of mode setting in the DMI control logic. As indicated, these rates are a strong function of the instruction sequence being executed by the CPU.



Note 1. MEM IDLE High to MEM IDLE High Interval varies as Function of Instruction – See Section 6.

Figure 13-3. DMA Block Transfer Timing

13.2 USING AN INTERRUPT TO IDENTIFY THE END OF A DMA TRANSFER

A DMA transfer may be terminated either under program control, by loading the DMA control I/O port with a 0 in the enable bit, or by the last byte of a fixed length buffer being accessed. (See Figure 6-2.) In the latter case, the executing program may be interrupted at the end of the DMA transfer, using the 3854 DMA device's enable output as an interrupt request to the 3851 PSU. This can be illustrated as follows:

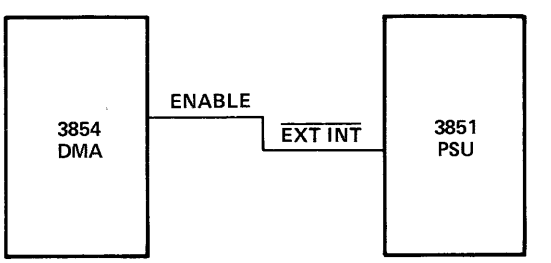


Table 13-1. DMA and Refresh Rates

MODE SETTING \ SITUATION		BEST CASE 1 slot every 2 μS. e.g., all short cycle instructions	TYPICAL CASE 1 slot every 2.5 μS. e.g., branch instruction (short-long-short)	WORST CASE 1 slot every 3.9 μS. e.g., ST/ST/ST/ST Branch
Use 1 of every 4 DMA/REF memory access slots for refresh	DMA	370K bytes/sec	300K bytes/sec	180K bytes/sec
	Refresh	64 locations in 0.51 mS	64 locations in 0.64 mS	64 locations in 1.0 mS
Use 1 of every 8 DMA/REF memory access slots for refresh	DMA	435K bytes/sec	350K bytes/sec	220K bytes/sec
	Refresh	64 locations in 1.02 mS	64 locations in 1.28 mS	64 locations in 2.0 mS
Refresh off	DMA	500K bytes/sec	400K bytes/sec	250K bytes/sec

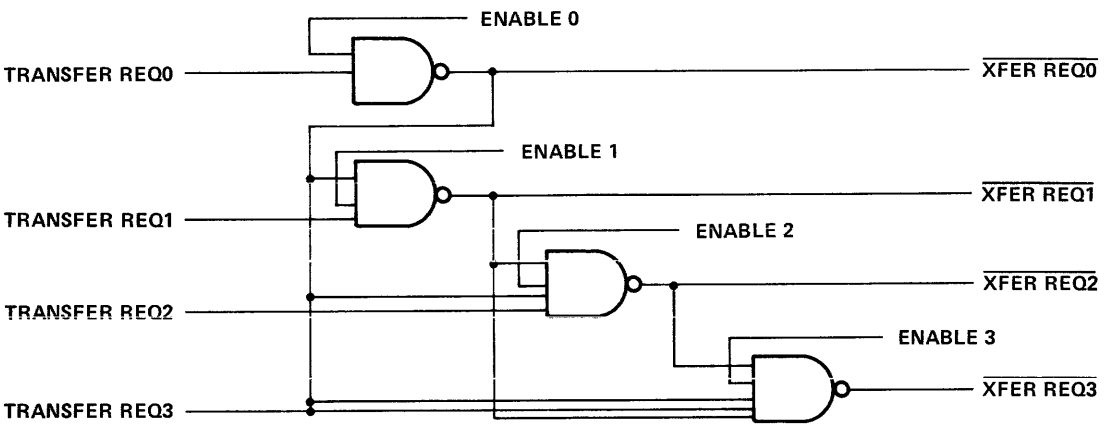
Each time the ENABLE line makes a high to low transition, which indicates the end of a DMA transfer, the $\overline{\text{EXT INT}}$ pin will sense this transition and cause an external interrupt to be generated within the F8 system.

13.3 INCLUDING MORE THAN ONE 3854 DMA DEVICE IN A CONFIGURATION

Up to four DMA devices may be present in an F8 configuration that includes just one 3850 CPU. Each DMA device provides one DMA channel.

These are the requirements for a configuration with more than one 3854 DMA device:

1. Each DMA device must have its own unique I/O port address, selected by strapping the P0 and P1 inputs appropriately.
2. One set of LOAD REG and READ REG signals may be generated to serve all of the DMA devices.
3. Each DMA device must have its own set of $\overline{\text{XFER REQ}}$, INPUT STROBE and OUTPUT STROBE logic.
4. Some means of selecting DMA device priority is needed. One method is to use $\overline{\text{XFER REQ}}$, as follows:



67095665

5. The DWS and STROBE signals of each 3854 DMA, along with the $\overline{\text{RAM WRITE}}$ signal from the 3852 DMI can be ANDed to create R/$\overline{\text{W}}$.
6. Each 3854 DMA device must be connected to the data bus and to the address bus of the F8 system.

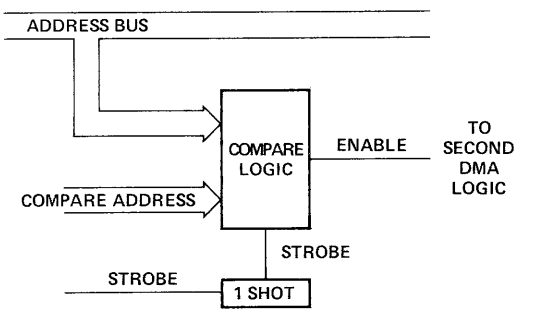
13.4 CATCHING DMA ON THE FLY

There are a number of applications where in order to save time, the processing of data being transferred into, or out of memory via DMA can begin before the DMA operation has completed. For example, it may be sufficient to ensure that the DMA operation is at least 25 bytes ahead of a subsequent operation which is going to access the same data buffer.

There are two ways of catching DMA on the fly. If the subsequent operation involves executing a program to operate on data in the DMA buffer, then the program that is to do the processing can simply read the contents of the DMA buffer address, out of the 3854 DMA device, and by comparing this address with some reference value, program logic may determine whether the DMA operation has proceeded far enough. The following is an instruction sequence that would provide the appropriate program logic:

```
        IN    F1      INPUT HIGH ORDER ADDRESS BYTE
        CI    HI      COMPARE IMMEDIATE WITH
                      REFERENCE VALUE
        BM    OUT     DMA FAR ENOUGH IF [F1] > HI
        BNZ   NMAT    DMA NOT FAR ENOUGH IF [F1] < HI
       *——    ——      TEST LOW BYTE IF [F1] = HI
        IN    F0      INPUT LOW ORDER ADDRESS BYTE
        CI    LO      COMPARE WITH REFERENCE VALUE
        BM    OUT     DMA NOT FAR ENOUGH IF [F0] ≤ LO
*PROGRAM STEPS FOR NO MATCH BEING HERE
NMAT    —
        —
        —
*PROGRAM STEPS FOR DMA HAVING PROGRESSED FAR
 ENOUGH BEGIN HERE
OUT     —
        —
        —
```

In a configuration that has more than one 3854 DMA device accessing the same memory, it is conceivable that the subsequent operation is a second access of the same data buffer. For example, one DMA operation may write data into a data buffer from external source (A) while a second DMA operation subsequently reads the same data (perhaps subject to some elementary amount of processing) to an external source (B). In this case, it is possible to catch data on the fly by comparing the contents of the address bus, using external logic. By only reading the contents of the address bus during the pulse from the 1 shot which is triggered by STROBE, external logic can guarantee that it is reading the address output by the 3854 DMA, rather than the address output by the 3852 DMI. In this way, external logic can isolate the DMA address and compare it using the following type of logic:



14.0 Multiprocessor Configurations and Applications

MULTIPROCESSOR CONFIGURATIONS AND APPLICATIONS

While the concept of connecting several computers together to solve a problem has been around for a number of years, the recent advent of MOS microprocessors has stimulated much new interest in this area. The unique architectural features of the F8 make it especially amenable to multiprocessor configurations. This section will attempt to present the designer with a glimpse of the potential applications of multiprocessor systems and how the F8 fits into such structures.

14.1 MULTIPROCESSOR CONFIGURATIONS

The concept of multiprocessor networks is well suited for MOS microprocessors. There is, first of all, an immutable economical rationale. The F8 devices are extremely cost-effective and they are able to perform many dedicated functions without supporting electronics or special I/O chips. Thus, they can be used as a universal standard component for literally any definable task. They are, in fact, the LSI version of the TTL chips. Secondly, the MOS microprocessor, such as F8, unlike its hard-wired predecessor, is capable of generalized data manipulation, information storage and retrieval, and message communications. These attributes allow two or more microprocessors to team together to perform tasks requiring cooperation by several microprocessors. Here is a simple example: Microprocessor A is, say, used as a controller for an in-plant telephone switchboard, and microprocessor B is used to control the temperature of the same. If these two microprocessors are linked together, we will have the added capability of interrogating or controlling the plant temperature via a telephone call. In this example, the added benefits of this coupled microprocessor, is derived from the communications link.

There is another form of microprocessor network in which the individual microprocessor shares a common memory. This configuration provides the most efficient information exchange among the constituent microprocessors of the network. Here is an example of this type of microprocessor network: Suppose that there are four microprocessors in a certain manufacturing company; microprocessor A handles the order entry; microprocessor B controls the manufacturing and inventory; microprocessor C processes the receivables and payables, and microprocessor D keeps track of shipment and returned goods. Since each microprocessor has only a portion of the total company data base, none is capable of compiling a comprehensive monthend P and L statement for the company. Now, if these microprocessors are arranged to share the same memory, then any one of the four microprocessors is capable of preparing a monthend statement since it has immediate access to all relevant data without human intervention. One can readily appreciate from the above example that a network of four low cost microprocessors is capable of performing tasks that have traditionally been done on a large scale computer costing some hundred thousands of dollars. Just as significant is the fact that this network concept provides a high degree of modularity which provides ease in hardware implementation, software partitioning and simplicity in system debugging.

14.1.1 Network with Communications Bus Link

Figure 14-1 below shows a group of microprocessors linked together via a common communications bus. This is the most inexpensive way of interconnecting a group of microprocessors together. Since microprocessors are capable of manipulating any data format, this network can be made to work with any suitable communications protocol. Figure 14-2 shows an example of this type of network using F8 microprocessors. In this case, microprocessor #1 is considered as the communications bus controller. The entire communications bus is made up of only two wires. One wire is used as a bi-directional data line while the other wire is used as a synchronizing line to strobe the data. Here, the F8's unique I/O structure is utilized to almost completely eliminate the hardware overhead that would accompany this communications scheme were it implemented with the more conventional structures of other processors. Figure 14-3 is a timing diagram showing the relationship between the data and the sync signals. The diagram is exaggerated to emphasize that the successive sync pulses do not have to conform to a fixed time interval, i.e., it can be asynchronous. The sync line is controlled by microprocessor #1, the bus controller. This line drives the INTERRUPT inputs of all the other microprocessors on the

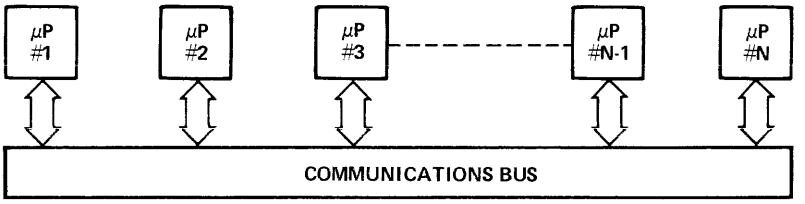


Figure 14-1. Communication Oriented Microprocessor Network using a Common Communication Bus

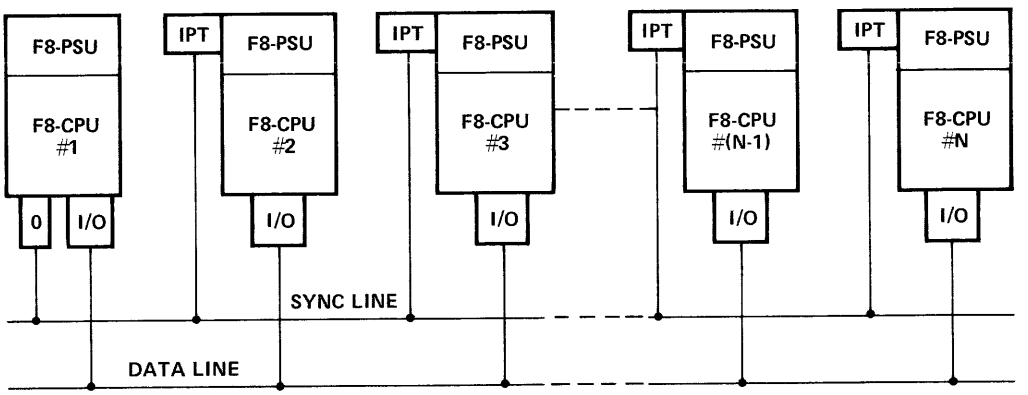


Figure 14-2. F8 Microprocessor Network using a Two-Wire Bus

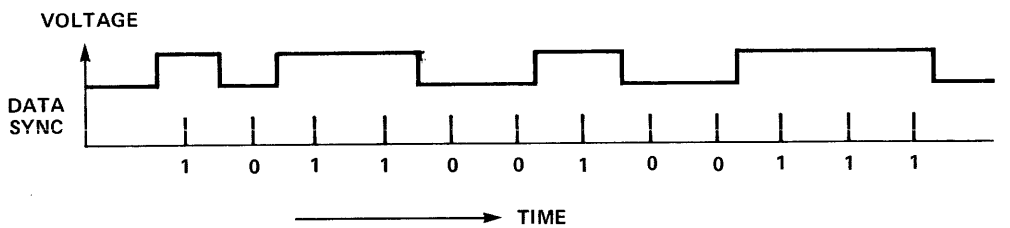


Figure 14-3. Timing Relationship between Data and Sync Signals

network. When microprocessor #1 broadcasts a message, all the other microprocessors have access to the message. This message should be formatted to take advantage of the best practices of communications discipline, i.e., it should have a header field with addresses of the transmitter and receiver, a control field, the data field, and finally the CRC check field. In fact, even the SDLC protocol can be employed if desired. Since each microprocessor on the network is capable of checking CRC's And capable of understanding the message format, reliable message transmission and reception is assured. Periodically, microprocessor #1 will permit one of the other microprocessors to send out a message on the bus. In this case, the sync timing is still provided by the bus controller. Since the message is broadcasted to all other microprocessors, it may address any member of the network. It will be a useful convention to specify a maximum time interval between two consecutive sync pulses. Thus, a period of silence from the bus controller will mark the beginning of a new message, and spurious pulses at power-on will be disregarded. F8 microprocessors have no difficulty in determining the time interval between interrupts because they are equipped with on-board timers. There are many possible enhancements on this basic system shown in Figure 14-2. Clearly, the data rate can be increased by increasing the number of data lines. Also, a more symetrically wired system, shown in Figure 14-4, permits the role of the bus controller to be dynamically reassigned. Thus, the failure of a bus controller will not deprive the rest of the network of their ability to communicate among themselves. The reliability of the network is therefore enhanced. This type of network is especially useful where the individual microprocessors are separated from each other by significant distances. For example, this two-wire network is ideally suited for controlling a large aircraft. Because of the physical simplicity of this network, it becomes practical to provide a duplicate system as a backup. To sum up, this type of "distributed" system has the following salient features:

1. Simplicity in communication
2. Localized intelligence
3. Modularity
4. High reliability

However, it is not an efficient network organization for multitask operations that must have access to a common storage.

14.1.2 Network with Shared Common Memory

Figure 14-5 shows the essence of a microprocessor network with a common memory. Each microprocessor may have its own private memory in addition to the common storage, therefore each microprocessor executes its own program without waiting for others. The data that have common interest can be written into and read from the common memory. The common memory is, therefore, a highly accessible medium of information exchange. This network configuration takes timely advantage of the recent improvements in the IC memory technology. Structurally, all microprocessors of the network are similarly wired to the common memory via the common address bus and the data

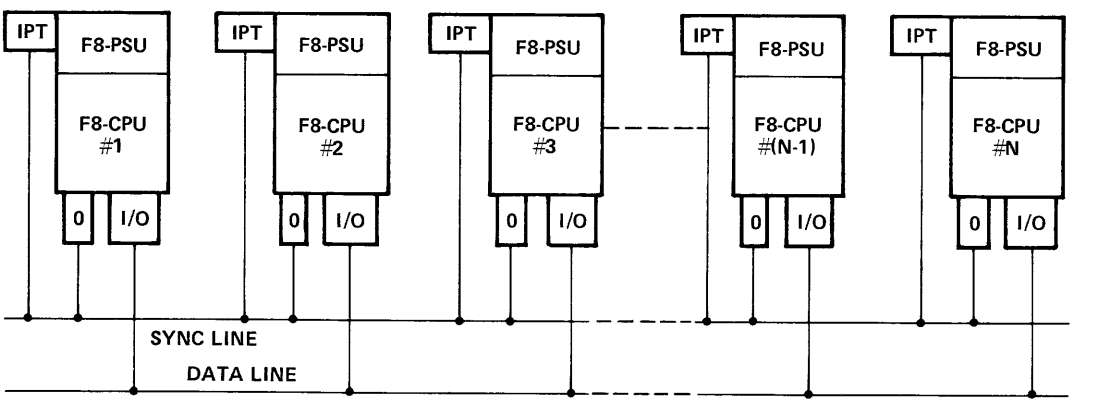


Figure 14-4. F8 Microprocessor Network using a Symetrically Connected Two-Wire System

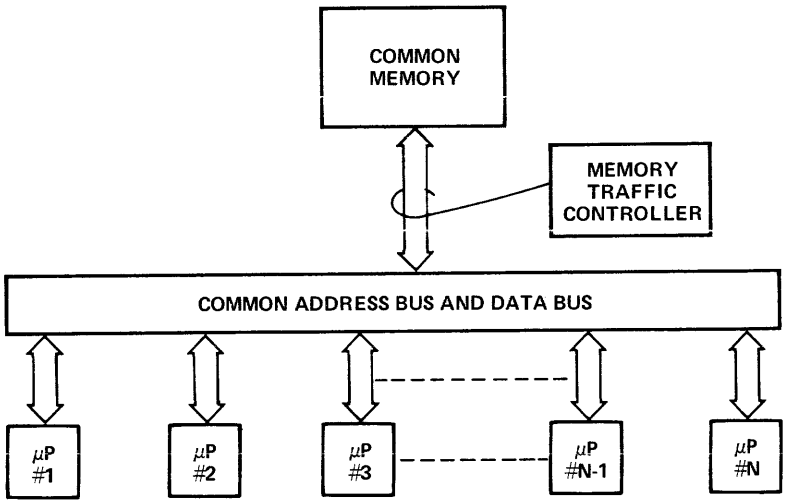


Figure 14-5. Microprocessor Network with Common Memory

bus. A memory traffic controller is needed to resolve the conflict due to the simultaneous requests for the use of the common memory by several microprocessors. This function can be readily incorporated into the microprocessor itself. Typically, the common address bus (Figure 14-5) has some 16 lines, and the common data bus has some 8 lines. Therefore, it does not have the simplicity of a two-wire system shown in Figure 14-2. Besides, long bus wires deteriorate the performance through the lengthening of the cycle time of the common memory. Except for the fact that this type of network is not particularly suited for long distance communications, it is an exceedingly powerful organization. Its most outstanding attribute lies in its ability to execute simultaneously all logically divisible subtasks, for subsequent correlation. This attribute is especially compatible with the low cost MOS microprocessors.

The F8 chip set permits several convenient methods of implementing this network concept. Figure 14-6 shows how several F8 microprocessors gain access to a common memory through the use of F8 memory interface chip (F8-DMI) and the F8 direct memory access chip (F8-DMA). The key to understanding this system is to recognize that modern RAM chips are much faster than the microprocessors. Typically, the cycle time of the available RAM chips ranges from 100-500 nanoseconds, while the shortest execution cycle of the available MOS microprocessors ranges from one to two microseconds. The memory interface chip (F8-DMI) takes advantage of the inherent bandwidth of the memory by dividing the F8 machine cycle into several memory access periods as discussed in previous sections. In Figure 14-6, the division of memory bandwidth between μP #0 and the DMA chain has been simplified for clarity. Actually, a certain amount of the memory's bandwidth will be lost to overhead by way of the MOS devices' circuit delays in switching from one channel to another.

Since the microprocessors associated with DMA channels, i.e., microprocessors #1, #2, • • •, #N, have their own private memories, their operations are continuous and independent of the assigned bandwidth to the common memory. However, each DMA has an assigned task. For example, microprocessor #1 may be used as a Floppy Diskette controller. In this case, DMA #1 must be able to accommodate a maximum data transfer rate of 250K bits per second or 31.25K bytes per second. Let us further assume that microprocessor #2 controls a duplex data link with a baud rate of 56,000. The corresponding bandwidth required by DMA #2 is (2 X 56,000) ÷ 8 = 14K bytes per second. Thus the bandwidth remaining for the rest of the DMA chain would be (if we assume a total DMA bandwidth of 400K bytes being available):

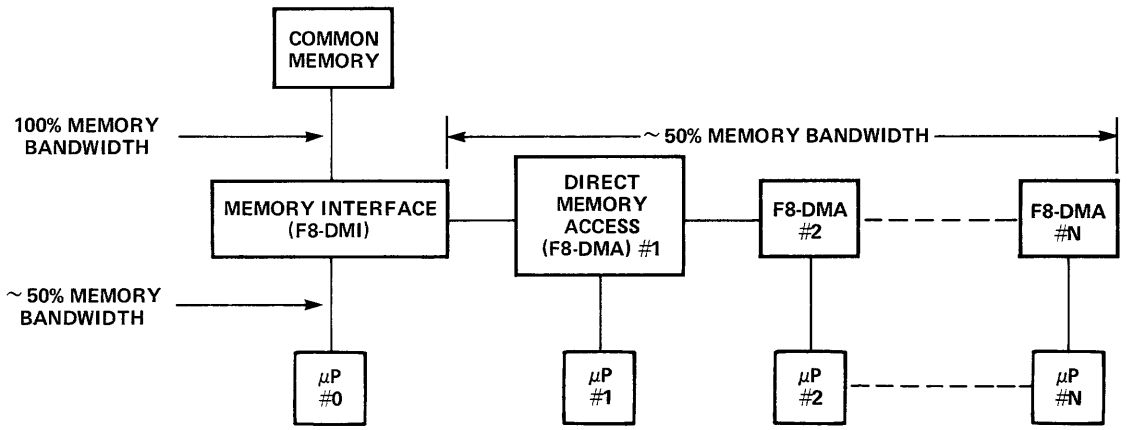


Figure 14-6. Multiprocessor Network using F8-DMI and F8-DMA Chips

400 - 31.25 - 14 = 354.75K bytes per second

This is certainly enough to accommodate a large number of other high speed devices. Table 13-1 in Section 13 analyzes DMA bandwidth availability in an F8 system.

Figure 14-7 presents an expanded picture of what μP #1 might look like and its interface into the network. We can see from this figure that while microprocessor #1 specifies the locations of the common memory for transferring data to or from the floppy diskette, the common memory cannot be specified as part of its own memory space. In other words, the DMA method of accessing the common memory in this configuration involves an intermediary step of data buffering. This step is quite desirable if a peripheral device is involved such as a floppy diskette or a CRT screen. However, it is extraneous if the data fetched from the common memory is to be executed as an instruction by the microprocessor. When the latter condition prevails, a network such as the one shown in Figure 14-8 is more suitable.

In Figure 14-8, all CPUs share the same common memory. The block designated as the memory interface provides an orderly means of utilizing the common memory by several contending CPUs. This is called a "one port" memory system, because only one CPU can use the common memory at one time. Therefore, the maximum speed is determined by the access time of the common memory.

Theoretically, the optimum CPU execution rate should be N times the memory access time—providing the perfect match between the memory and the CPU speeds. But, if the CPU speed is comparable to the memory speed, a multiport memory system should be constructed as shown in Figure 14-9.

This is a highly efficient system where any CPU can have access to any module of the common memory. The block designated as the memory interface is a gigantic cross-bar switch with built-in conflict resolver. While conceptually gratifying, this system is wasteful of hardware. Therefore, it is rarely implemented except in very large computing systems. A good compromise is realized in Figure 14-10, where every microprocessor is autonomous since each has its own private memory. This pre-empts the memory conflict that prevailed in the network shown in Figure 14-8. Nevertheless, each microprocessor may have access to a common memory through a simple pair of isolating buffers. These buffers are controlled by a system conflict resolver which permits only one microprocessor to use the common memory at any given time. Each microprocessor, upon gaining access to the common memory, treats it as a part of its own memory space. This system provides a simple but elegant synergism among a set of simultaneously operating microprocessors.

The system diagram presented in Figure 14-11 illustrates how some of the F8's characteristic

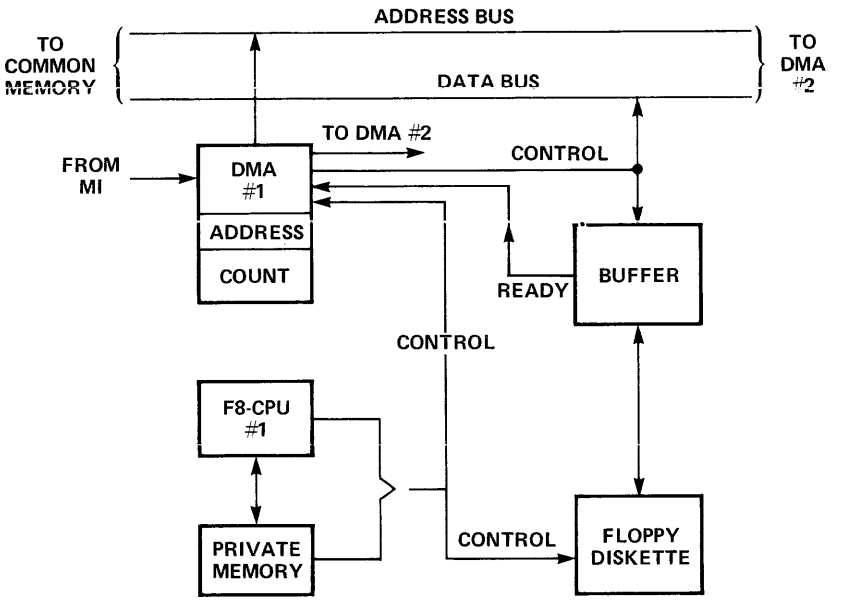


Figure 14-7. Detailed Block Diagram of Floppy Diskette Controller (DMA #1 and F8-CPU #1 of Figure 14-6)

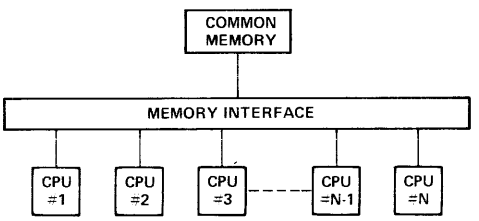


Figure 14-8. Network with Common Memory Space

features can be utilized in implementing a shared memory network. Here, each F8 microprocessor, in addition to having its own private (local) memory in the form of a PSU, is linked to the common memory address bus by a 3852 DMI. At any given time only one of the DMIs will have its address drivers enabled, with control coming from the I/O port of a coordinator CPU and being administered through the CPU SLOT input to the DMIs. Recall that CPU SLOT, when held low by external logic, causes the address drivers and $\overline{\text{RAM WRITE}}$ driver of the DMI to be placed in a high impedance state.

When the coordinator CPU decides that, for example, one of the other CPUs should relinquish control of the common memory, it activates the EXTERNAL INTERRUPT line of that CPU. In response to this the second processor system makes sure that it is not executing any code from the common memory address space (i.e., it jumps back into its own PSU), and the coordinator CPU can then change the active CPU SLOT to another processor system. Thus, in a system such as this we can actually establish two levels of processor communication. One level would involve the hand-shaking and prompting type of information that act as stimuli for a processor system to take some further action. This can be done using either the external interrupt structure of the network as is illustrated in the figure, or through I/O port bits linking the microprocessor systems. The key point is that this first level of communication would involve only very low speed information rates and be only very primitive in nature.

The second level of communication is then via the shared memory. This provides for an almost

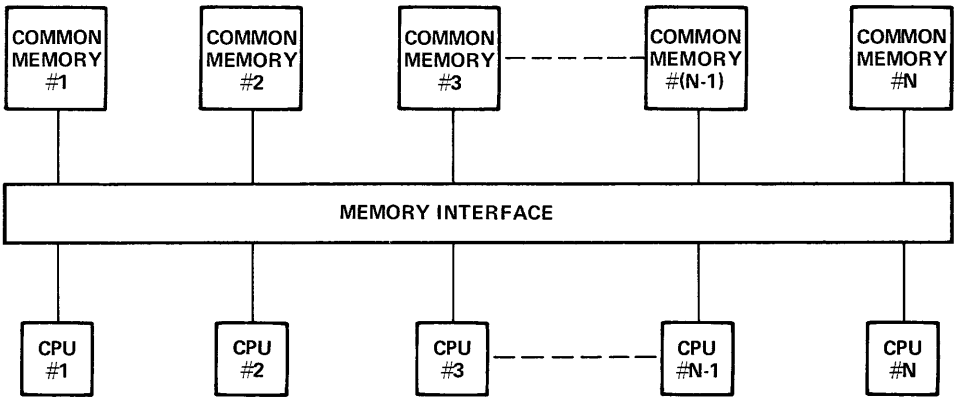


Figure 14-9. Network with Multiport Common Memory

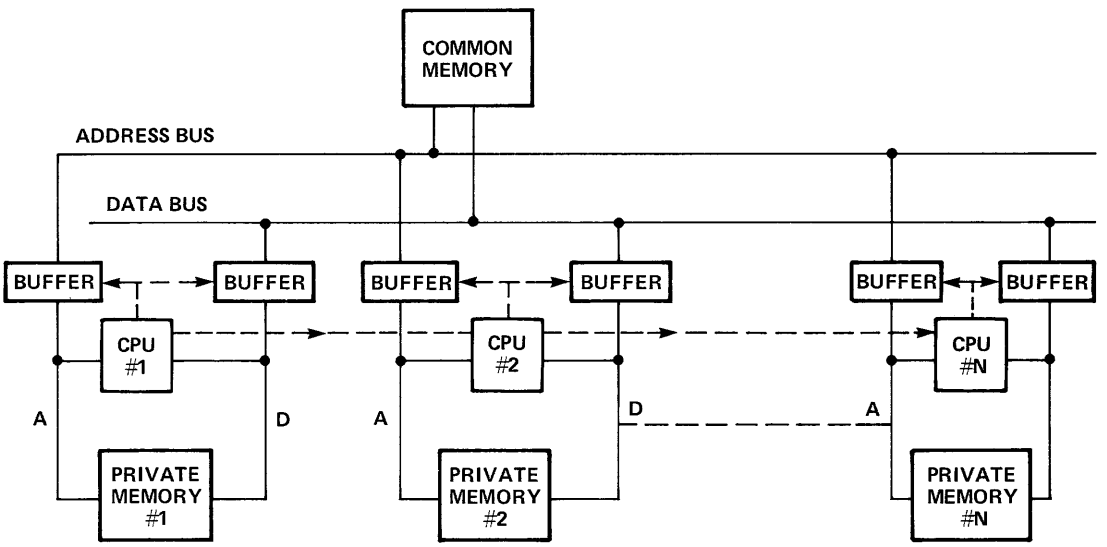


Figure 14-10. Network with Both Private and Common Memories in the Same Memory Space

instantaneous transfer of large amounts of data from one processor system to another. Information transmitted in this way can be coordinated through a simple software protocol. This protocol is known as the "mailbox" system. In this system, one microprocessor is designated as the "coordinator." Every other microprocessor on the network has two sets of memory locations (the mailboxes) in the common memory for message exchange with the coordinator. Let us say that microprocessor #1 is the coordinator for the network in Figure 14-11. We may then designate locations, 0-9, in the common memory as Mailbox 12, i.e., these ten memory locations are used for microprocessor #1, the coordinator, to pass information to microprocessor #2. Similarly, we may designate locations, 10-19, as Mailbox 21 that is used for microprocessor #2 to deposit messages intended for microprocessor #1. Figure 4-12 shows a possible mailbox assignment.

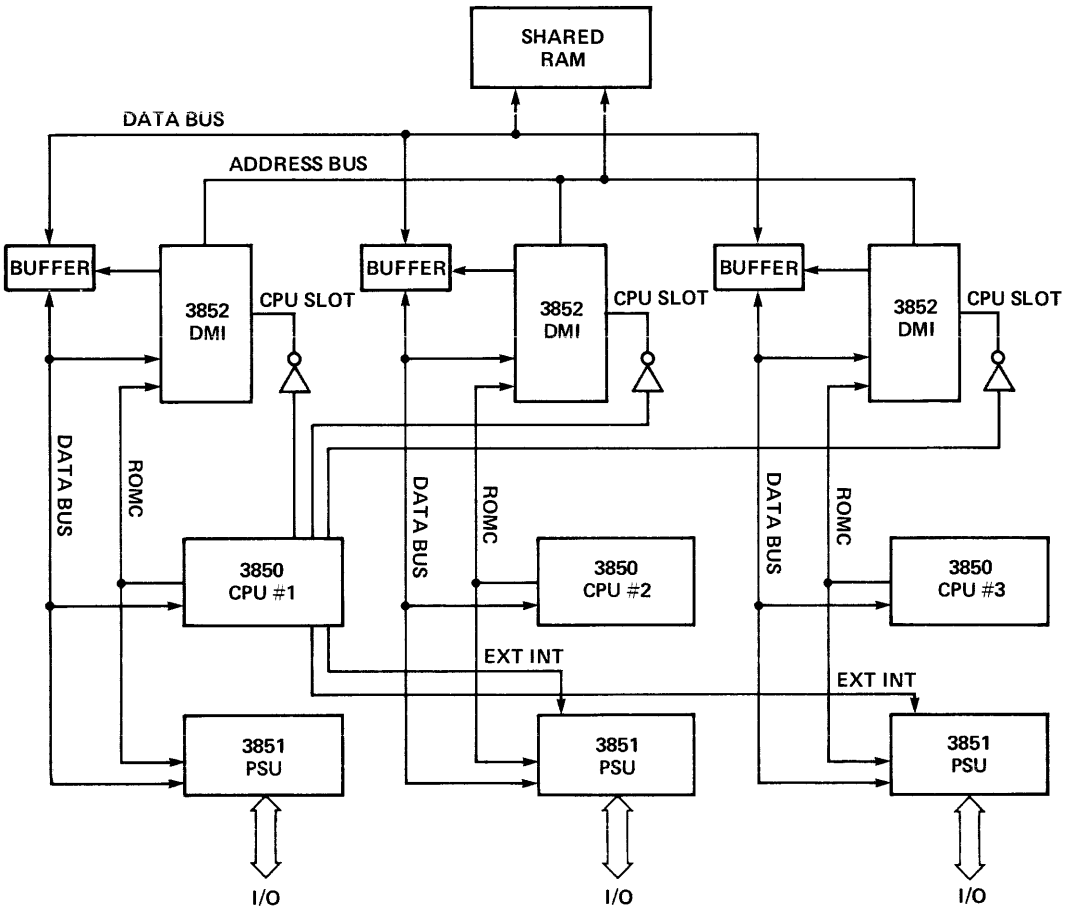


Figure 14-11. F8 Shared Memory System

Let us clarify how this mailbox system works through an example. In a certain application of the network shown in Figure 14-11, microprocessors #1, #2, and #3, are respectively the coordinator, the floppy diskette controller, and the data link controller. If the coordinator wishes to transmit a record currently stored in floppy diskette #8, track #4, and record #17, to a distant city, it will manipulate the mailboxes in the following way:

1. Deposit a message in Mailbox 12 to the effect that microprocessor #2 is to fetch the proper record (i.e., diskette #8, track #4, and record #17) to the common memory to the locations 1000-1127.

2. On periodic scan of its own mailbox, i.e., Mailbox 12, microprocessor #2 discovers the above message. It promptly executes the instruction

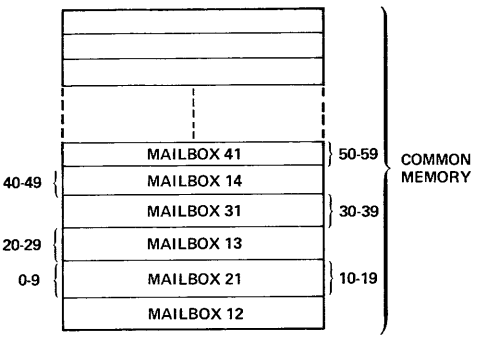


Figure 14-12. A Possible Mailbox Assignment

i.e., places the record in locations 1000-1127. It then leaves a message in Mailbox 21 stating that the operation is successfully concluded.

3. In one of the periodic scans of Mailbox 21, the coordinator becomes aware of the presence of the record in locations 1000-1127. It then issues an order to microprocessor #3, the data link controller, via Mailbox 13. In the order, the coordinator states that the record is presently in locations 1000-1127, and that it is to be sent to a specific coding scheme such as bi-sync or SDLC.

4. Microprocessor #3 having understood the message in Mailbox 13, transmits the data in locations 1000-1127 with the specified format. It then signals the completion of the data link operation by leaving an appropriate message in Mailbox 31 for the coordinator.

5. When the message in Mailbox 31 is read by the coordinator, it marks the end of the transfer.

This entire operation consumes only a few milliseconds. The significant point of the above example is that each microprocessor is almost totally independent with the exception of the simple "Mailbox" convention. This means that the program of each microprocessor can be independently developed and debugged without any regard to the program of the other microprocessors on the same network. This is an important development in computing technology especially in view of the high availability and low cost of the modern MOS microprocessors such as the F8.

So far in this section, we have discussed two practical types of microprocessor networks:

1. Communications oriented
2. Common memory oriented

Out of these two canonical forms, we can derive many hybrid networks using different combinations and hierarchies. It is safe to predict that many tasks having henceforth been the private domain of large computers, will succumb to the onslaught of microprocessor networks. There is even suggestion that large operating software be partitioned for multiprocessor implementation. To the microprocessor user, the possibility is unbounded.

14.2 A COMPARISON BETWEEN THE SINGLE MICROPROCESSOR SYSTEM AND THE MULTIPROCESSOR NETWORK

When MOS microprocessors first became available some four years ago, the users tended to use them as if they were expensive minicomputers. Subsequently, they built various peripheral controllers to supplement the single microprocessor. This tends to negate any cost advantage for using the microprocessor in the first place. In a single microprocessor system, the CPU has to assume a variety of different tasks. This gives rise to two undesirable consequences:

1. It has neither hardware nor software modularity
2. It tends to force the microprocessor architecture to mimic the minicomputer architecture.

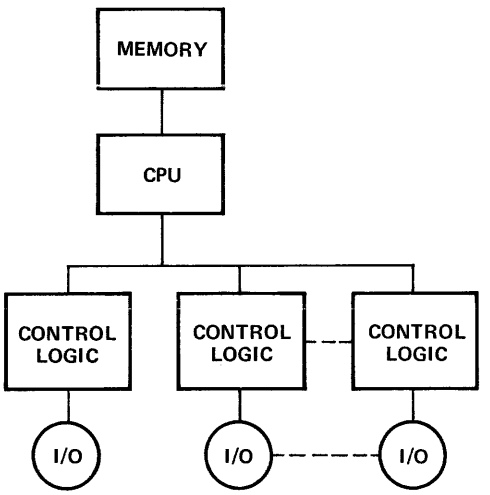
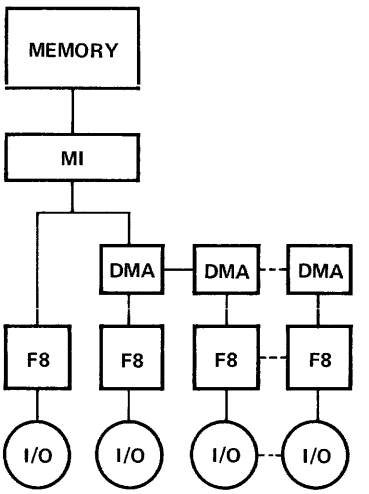
These trends run counter to the most important advantage offered by the LSI technology, namely, low cost and large volume.

The F8 microprocessor possesses two important features that gainfully exploit the LSI technology:

1. It takes minimum number of chips to configure a dedicated function
2. It permits the formation of microprocessor networks easily.

Table 14-1 gives a comparison between a single processor system versus a microprocessor network system.

Table 14-1. A Comparison between a Single Processor System and a Microprocessor Network

SINGLE MICROPROCESSOR SYSTEM	MULTI-F8 SYSTEM
	
1. difficult to expand the system in hardware	1. easy to expand the system in hardware (using the same basic chips)
2. difficult to extend the software to include more function	2. easy to extend the software because it considers each additional S.R.
3. single memory port	3. 2 memory ports
4. expensive minimal system	4. low cost minimal system
5. different control logic	5. same F8 parts
6. difficult to integrate and debug	6. easy to integrate and debug
7. demands high rate of interrupt response	7. does not require high rate of interrupt response
8. costly (many different parts)	8. cost effective by using the same and fewer F8 parts
9. no modularity	9. total modularity in hardware and software
10. incompatible with low cost MOS technology	10. compatible with low cost (highly reproducible) MOS technology

Appendix A. FAIR-BUG – 3851A PSU Specifications

A.1 GENERAL DESCRIPTION

A special Debug ROM 3851A PSU has been developed by Fairchild to provide the F8 user with a convenient and powerful programming debug facility which is used to aid in the development of F8 programs. This debugging program (FAIR-BUG) provides the user with an interactive system via a teletype terminal. The following capabilities are provided:

- Display or Alter Memory locations
- Display or Alter Scratchpad Registers
- Display or Alter Accumulator, ISAR, Status (W Register)
- Display or Alter PC0, DC0, DC1
- Load Formatted Paper Tape
- Punch Formatted Paper Tape
- Punch Paper Tape in PROM Format
- Entry from Keyboard or by Program Instruction
- I/O Subroutines available to user

A.2 ELECTRICAL SPECIFICATIONS

The DC and AC electrical characteristics of the 3851A PSU are, since this is just a standard PSU with a special program in it, identical to those described in the 3851 PSU Device Specification section.

A.3 FUNCTIONAL SPECIFICATIONS

The FAIR-BUG is assigned memory addresses '8000'-'83FF', with entry point being '8080'. Port assignments are as follows:

I/O Port A —	'04'
I/O Port B —	'05'
Local Int. Control —	'06'
Timer —	'07'

The interrupt address vector for the timer is '0020' and for external interrupt is '00A0'.

FAIR-BUG can be entered in several ways. Execution of program instructions such as PI '8080'; JMP '8080'; LR P0, Q, or PK (Q or K contains '8080') can be used to achieve entry from another software module. Another technique is to use hardware to decode the RESET state on the ROMC lines and force the high order bit on the data bus to a 1 at this time. This is used in the F8 Kit and allows the lowering of the $\overline{\text{EXT RES}}$ line to force the system into FAIR-BUG.

FAIR-BUG will save the state of the machine upon entry and will restore it upon return to the user's program. Register 8 and PC1 are destroyed by FAIR-BUG; however, under program control the user can save and restore these if necessary. The save area utilized by FAIR-BUG is scratchpad registers 3C to 3F and also the last 26 bytes of user RAM. FAIR-BUG tests locations 3E6-3FF and if there is RAM there, it uses these locations for its save area; if it finds that these locations are not RAM, it assumes that locations BE6-BFF of user memory are RAM and thus uses these locations for its save area. The interrupt is disabled by FAIR-BUG and may be re-enabled by the user if desired.

Two I/O ports (4 and 5) are available to the user when FAIR-BUG is not executing, as is the timer and external interrupt facilities (FAIR-BUG does not use either of these). During FAIR-BUG execution Port 4 is used for serial input/output and control functions while Port 5 is used for parallel input from a high speed paper tape reader. Assignments for Port 4 are: (1 = GND, 0 = +5V).

BIT	FUNCTION		
7	Serial input (0V = MARK)		
6	Character Ready Input (Parallel Device) (+5V = READY)		
5	—		
4	Device Ready Input (Parallel Device) (+5V = READY)		
3	Step Reader Output (Parallel Device) (0V = STEP)		
2-1	BIT 2	BIT 1	BAUD RATE
	0	0	110 Baud for Serial Input/Output
	0	1	300 Baud for Serial Input/Output
	1	0	Parallel Loader Input Baud Delay Counter in Memory Loc. 3FF (or BFF)
	1	1	Baud Delay Counter in Memory Loc. 3FF (or BFF)
0	Serial Output (0V = MARK)		

If Port 5 is not utilized by FAIR-BUG, then pins 3, 4, and 6 of Port 4 are also available to the user. If Port 5 is used for parallel input, the parallel input data should have logic 1 correspond to +5V electrical level.

Pins 1-2 of Port 4 are examined when FAIR-BUG is entered to initialize the baud delay counter and also whenever a load command is given to determine whether the input is bit serial on Port 4 or parallel on Port 5. If pin 2 is at ground so that the baud delay count is being obtained from memory location 3FF (or BFF), then the total delay count (time between bits for a bit serial I/O device) can be adjusted as follows:

bit time interval = (Count + 1) * 36 μS + 55
(Φ = 2 MHz)

Thus, 110 baud requires a count of 06

300 baud requires a count of A6

1200 baud requires a count of EA

A.4 FAIR-BUG COMMANDS

When FAIR-BUG is entered.a prompt character (?) is sent to the output device. The user then has the option of using any of the debug commands. After each debug execution the user is again prompted with (?). All data and input parameters are in hexidecimal notation. (C/R) following a command indicates a carriage return.

COMMAND TYPE	COMMAND	FUNCTION
Display	A (C/R)	Dispaly the contents of the accumulator
	D0 (C/R)	Display the contents of DC0
	D1 (C/R)	Display the contents of DC1
	I (C/R)	Display the contents of ISAR
	M XXXX (C/R)	Display Memory Location XXXX
	M XXXX-YYYY (C/R)	Display Memory Location XXXX to YYYY
	P0 (C/R)	Display the contents of PC0
	R XX (C/R)	Display the contents of Register XX
	R XX-YY (C/R)	Display the contents of Registers XX to YY
	S (C/R)	Display the contents of W Register, status
	W (C/R)	Display the contents of W Register, status
Change	C XX (C/R)(C/R)	Change the previously displayed memory location or register to XX
	C XX (C/R) YY (C/R) . . . (C/R)(C/R)	Change the sequential registers or memory locations to XX, YY . . .
	C XXXX (C/R)(C/R)	Change the previously displayed PC or DC to XXXX
Examine	E (C/R)	Display the last addressed register or memory location
Next	N (C/R)	Display the next register or memory location
Load	L (C/R)	Load formatted object paper tape. If (CK) prints then checksum error has occurred on block last read.
Punch	B XXXX-YYYY-Z	Binary Punch PROM format; XXXX is starting page address and YYYY is ending page address. Z is number of bytes per block; 0 = 256, 1 - 512. To punch 0 to BFF then enter B0-C00-0.
	F XXXX-YYYY (C/R)	Formatted punch for future load

COMMAND TYPE	COMMAND	FUNCTION
Go To	G (C/R)	Go to address of PC0
	G AAAA (C/R)	Change PC0 to address AAAA, then go to AAAA and continue execution
Delete Command	[	Delete command and start a new command input string

The following I/O Subroutines on the FAIR-BUG ROM are available to the user:

NAME	ENTRY ADDRESS	FUNCTION
TTY1	83AD	Input 1 byte from TTY type device (11 bits serial/character)
TTY0	83E5	Output 1 byte to TTY type device (11 bits serial/character)
TTCR	83D6	Output CR, LF & Null characters using TTY1 subroutine
PINP	8397	Input 1 byte from the parallel IP device (150 μS minimum delay between characters)
FOP1	80E9	Output 1 or 2 hexadecimal digits in ASCII format from a memory location
FOP2	80EB	Output 1 or 2 hexadecimal digits in ASCII format from Register QL
BYTE	837B	Input 2 ASCII characters from a parallel or serial IP device; then convert them to one hexadecimal byte

3508 Future Product Description. 8192-Bit Read-Only Memory

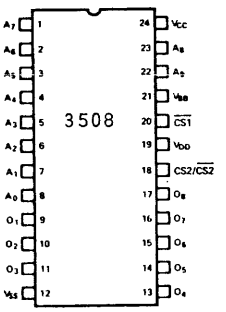
GENERAL DESCRIPTION

The 3508 is an 8192-bit static MOS read-only memory designed to be a high performance replacement for the iNTEL 2308. Internal organization is 1024 X 8 bits to allow for simple interface to F8 microprocessor systems. Ease of memory expansion is facilitated by two chip selects, one of which is mask programmed (CS2/$\overline{CS2}$), and OR-tie capability on the outputs.

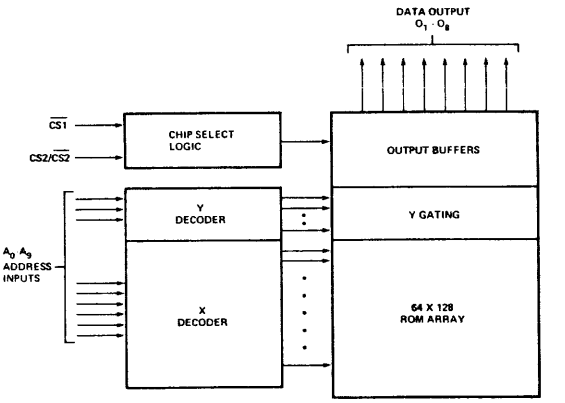
FEATURES

- TTL Compatible—all Inputs and Outputs
- Three-State Output—OR-tie Capability
- Fully Decoded—On Chip Address Decode
- Programmable Chip Select
- 500 μS Access Time
- Standard Power Supplies—+12V, +5V
- Completely Static Operation

PIN CONFIGURATION



BLOCK DIAGRAM



PIN NAMES

A_0 A_9	ADDRESS INPUTS
O_1 O_8	DATA OUTPUTS
$\overline{CS_1}$	CHIP SELECT INPUT
CS2/$\overline{CS2}$	PROGRAMMABLE CHIP SELECT INPUT

3516 Future Product Description. 16K-Bit MOS Read-Only Memory

GENERAL DESCRIPTION

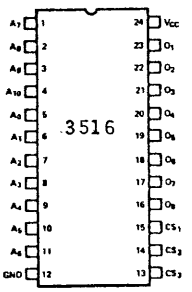
The 3516 is a 16,384-bit read-only memory circuit designed as a high performance replacement for the INTEL 2316. The internal organization of the 3516 is arranged as a 2K X 8 matrix to allow simple interface with eight bit processor applications. The static operation of the 3516 coupled with three programmable chip select inputs provide an easily expandable, high performance memory circuit with extremely simple interface requirements.

The 3516 read-only memory is fabricated with N-channel silicon gate technology to minimize chip size and optimize circuit performance. Ion Implantation allows TTL compatibility at both the inputs and the outputs.

FEATURES

- Single +5V power supply
- Completely static operation (no clocks required)
- Less than 600 nS access time
- Directly TTL compatible
- Three programmable chip select inputs
- Three-state outputs for OR-tie capability
- Input protection against static charge

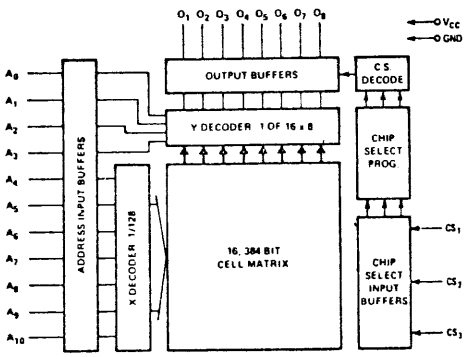
PIN CONFIGURATION



PIN NAMES

A_0-A_{10}	ADDRESS INPUTS
O_1-O_8	DATA OUTPUTS
CS_1, CS_2	PROGRAMMABLE CHIP SELECT INPUTS

BLOCK DIAGRAM



3539

256 × 8 STATIC RANDOM ACCESS MEMORY

FAIRCHILD ISOPLANAR SILICON GATE MOS INTEGRATED CIRCUIT

GENERAL DESCRIPTION – The 3539 is a 2048-bit Static Read/Write Random Access Memory. Organized as 256 8-bit words, the 3539 features a common I/O structure which allows packaging in a standard 22-pin ceramic DIP. This device uses a single +5 volt power supply and is TTL-compatible on inputs and outputs. The 3539 is manufactured using Fairchild's N-channel Isoplanar process.

- **256 x 8 WITH COMMON I/O BUS**
- **STANDARD 22-PIN DIP**
- **SINGLE +5 VOLT POWER SUPPLY**
- **COMPLETELY STATIC – NO CLOCKS OR REFRESH**
- **TOTALLY TTL-COMPATIBLE**
- **650 NS MAXIMUM ACCESS TIME**
- **<500 mW POWER DISSIPATION**
- **TWO SEPARATE CHIP SELECT INPUTS**
- **SEPARATE OUTPUT DISABLE FUNCTION**

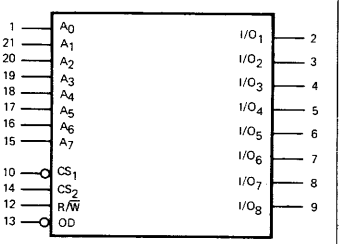
PIN NAMES

A_n	Address Inputs
$\overline{CS}_n$	Chip Select Inputs
$\overline{OD}$	Output Disable
$R/\overline{W}$	Read/Write Control Input
I/O_n	Data Buss Pins
V_{DD}	+5 V Power Supply
V_{SS}	0 V Power Supply

ABSOLUTE MAXIMUM RATINGS

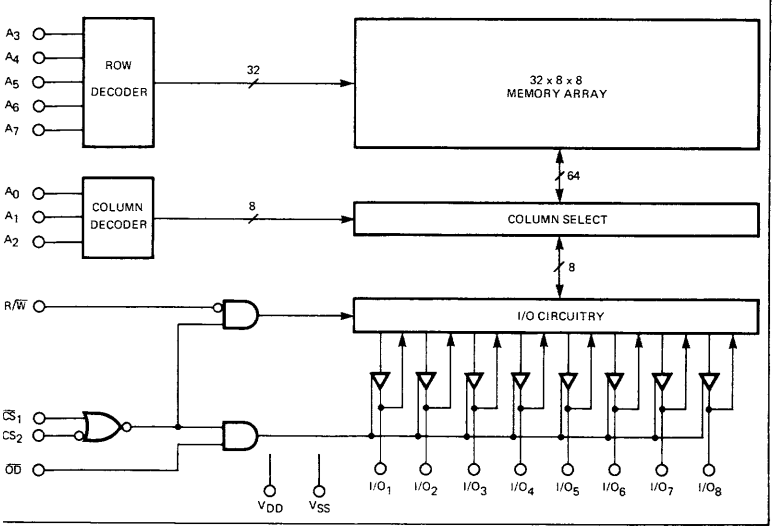
Any Pin with Respect to V_{SS}	−0.5 V to + 7.0 V
Storage Temperature	−55°C to + 150°C
Operating Temperature	0°C to + 70°C

LOGIC SYMBOL

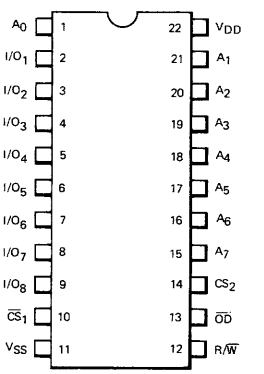


V_{DD} = Pin 22
V_{SS} = Pin 11

BLOCK DIAGRAM



CONNECTION DIAGRAM
(TOP VIEW)



464 ELLIS STREET, MOUNTAIN VIEW, CALIFORNIA 94042 (415) 962-5011/TWX 910-379-6435

FAIRCHILD SEMICONDUCTOR

 Printed in U.S.A. 2020-11-0008-075 8M

FUNCTIONAL DESCRIPTION: The 3539 uses a multiplexed Input/Output (I/O) structure, allowing the device to be packaged in a 22-pin DIP. The I/O network is controlled by the Read/Write (R/$\overline{W}$), Output Disable ($\overline{OD}$), and two Chip Select ($\overline{CS}_1$ and CS_2) inputs.

The I/O network is in a high impedance state and the R/$\overline{W}$ input disabled whenever $\overline{CS}_1$ is HIGH or CS_2 is LOW. When $\overline{CS}_1$ is LOW and CS_2 is HIGH the circuit will read or write, depending on the $\overline{OD}$ and R/$\overline{W}$ inputs.

When $\overline{OD}$ is HIGH, the eight I/O pins are in the Output mode, so that the R/$\overline{W}$ input should be HIGH to force the chip into a read mode. However, when R/$\overline{W}$ is HIGH, the $\overline{OD}$ can be used as a chip select input, turning the outputs off when it goes LOW.

When the R/$\overline{W}$ and $\overline{CS}_1$ are LOW and CS_2 is HIGH, the circuit is in the write mode. $\overline{OD}$ must be LOW to turn off the output structures. Data is then entered from the I/O pins. The $\overline{OD}$ is used to turn off the output structures independent of the Chip Selects, allowing input data to be entered at the I/O ports sooner than if the I/O were controlled by R/$\overline{W}$. Output data is not inverted by the 3539. The output buffers will each drive one standard TTL Load.

The eight address inputs specify which location of the memory array will be selected for the read or write operations. Each control, address and I/O input is directly TTL-compatible.

DC CHARACTERISTICS: $T_A = 0°C$ to $+70°C$; $V_{DD} = 5.0\ V \pm 5\%$

SYMBOL	PARAMETER	35391		35392		3539		UNITS	CONDITIONS
		MIN	MAX	MIN	MAX	MIN	MAX		
V_{IH}	Input HIGH Voltage	2.2		2.2		2.2		V	
V_{IL}	Input LOW Voltage		0.65		0.65		0.65	V	
V_{OH}	Output HIGH Voltage	2.2		2.2		2.2		V	$I_{OH} = -100\ \mu A$
V_{OL}	Output LOW Voltage		0.4		0.4		0.4	V	$I_{OL} = 1.6\ mA$
I_{BH}	Bus HIGH Current		10		10		10	μA	$V_{IN} = V_{DD}$, Chip Deselected
I_{BL}	Bus LOW Current		-10		-10		-10	μA	$V_{IN} = 0\ V$, Chip Deselected
I_{DD}	Power Supply Current		95		95		95	mA	$V_{DD} = 5.25\ V$
PD	Power Dissipation		500		500		500	mW	

AC CHARACTERISTICS: $T_A = 0°C$ to $+70°C$; $V_{DD} = 5.0\ V \pm 5\%$

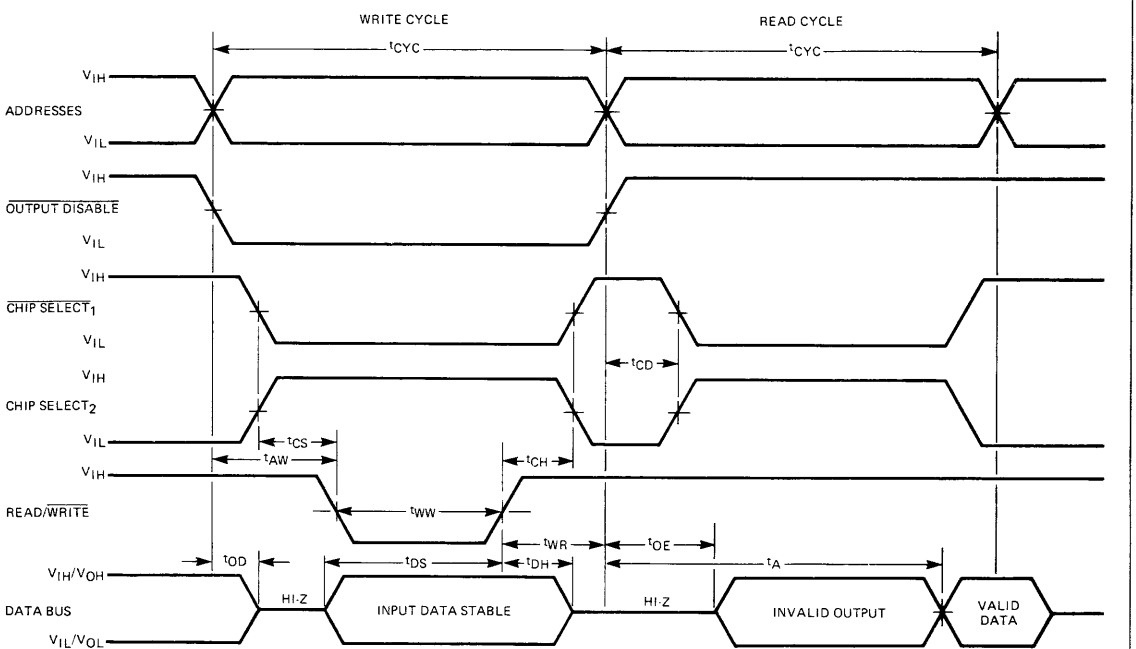
SYMBOL	PARAMETER	35391		35392		3539		UNITS	CONDITIONS
		MIN	MAX	MIN	MAX	MIN	MAX		
t_{CYC}	Read or Write Cycle Time	400		500		650		ns	
t_A	Read Access Time		400		500		650	ns	
t_{AW}	Address to Write Delay	150		200		300		ns	
t_{DS}	Data Set-Up Time	200		250		275		ns	
t_{DH}	Data Hold Time	25		25		50		ns	
t_{WR}	Write Recovery Time	75		75		100		ns	
t_{WW}	Write Pulse Width	175		225		250		ns	
t_{CS}	Chip select to write set-up time	100		125		200		ns	
t_{CD}	Chip Select Delay Time		100		100		100	ns	
t_{CH}	Chip select to write hold time	25		25		50		ns	
t_{OD}	Output Disable Time		150		150		150	ns	
t_{OE}	Output Enable Time		150		150		150	ns	

TRUTH TABLE

CONTROL INPUTS				3539 OPERATING MODE				I/O BUS MODE		
$\overline{CS_1}$	CS_2	$\overline{OD}$	$R/\overline{W}$	Selected	Deselected	Write	Read	Input	Output	HI-Z
H	X	X	X		•					•
X	L	X	X		•					•
L	H	L	L	•		•		•		
L	H	H	L	•		•			•	
L	H	L	H	•			•			•
L	H	H	H	•			•		•	

X = Irrelevent state
L = LOW($-V_{DD}$)
H = HIGH($-V_{SS}$)

3539 TIMING DIAGRAM

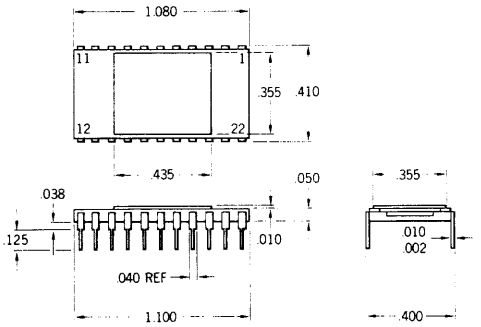


ORDERING INFORMATION

PART NUMBER	ACCESS TIME
3539DC	650 ns
35392DC	500 ns
35391DC	400 ns

PACKAGE OUTLINES

22-Pin Dual In-line (Metal Cap)



2102

1024×1 STATIC RANDOM ACCESS MEMORY

FAIRCHILD ISOPLANAR SILICON GATE MOS INTEGRATED CIRCUIT

GENERAL DESCRIPTION – The 2102 is a 1024-word by 1-bit Static Random Access Memory. It requires a single 5 V power supply, is fully TTL compatible on the inputs and the output and requires no clocking or refresh. The Chip Select ($\overline{CS}$) provides a 3-state output which allows the outputs to be wired-OR.

The 2102 is manufactured with the N-channel Isoplanar process. It is available in 16-pin ceramic Dual In-line Packages in either commercial, limited military or military temperature ranges.

- **FAST ACCESS (350 ns and 450 ns)**
- **SINGLE +5 V SUPPLY**
- **TTL COMPATIBLE ON INPUTS AND OUTPUT**
- **TOTALLY STATIC – NO CLOCKS OR REFRESH**
- **3-STATE OUTPUT**
- **FULLY EXPANDABLE**
- **FULLY DECODED**
- **16-PIN CERAMIC DUAL IN-LINE PACKAGE**

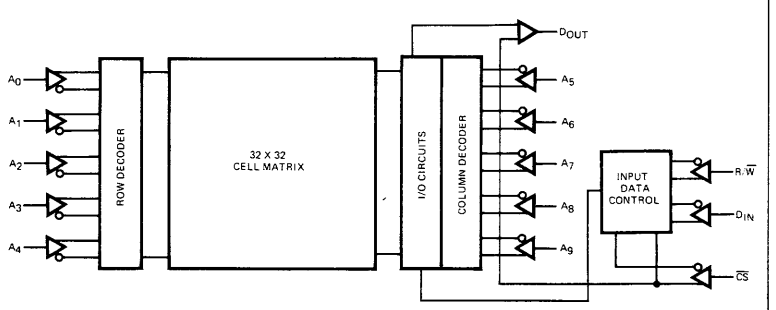
PIN NAMES

A_n	Address Inputs
D_{OUT}	Data Output
D_{IN}	Data Input
$R/\overline{W}$	Read/Write
$\overline{CS}$	Chip Select (active LOW)

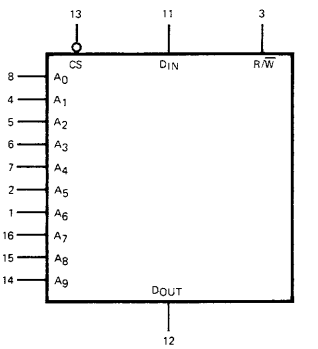
ABSOLUTE MAXIMUM RATINGS

Any Lead with Respect to V_{SS}	−0.5 V to +7.0 V
Storage Temperature	−55°C to +150°C
Operating Temperature DC	0°C to +70°C
DL	−55°C to +85°C
DM	−55°C to +125°C

BLOCK DIAGRAM

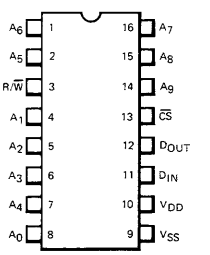


LOGIC SYMBOL



V_{SS} = Pin 9
V_{DD} = Pin 10

CONNECTION DIAGRAM
DIP (TOP VIEW)



TRUTH TABLE

$\overline{CS}$	$R/\overline{W}$	D_{IN}	D_{OUT}	Comments
H	X	X	*	Chip Deselected
L	L	H	H	Write "1" †
L	L	L	L	Write "0" †
L	H	X	D_n	Read †

X = Don't Care
* = Output High Impedance State
D_n = Data at Address Location
† = Chip Selected

464 ELLIS STREET, MOUNTAIN VIEW, CALIFORNIA 94042 (415) 962-5011/TWX 910-379-6435

FAIRCHILD SEMICONDUCTOR

 Printed in U.S.A. 2020-11-0002-055 15M

FUNCTIONAL DESCRIPTION – The 2102 is a 1024 x 1 static RAM. When the Chip Select ($\overline{CS}$) goes HIGH, the Read/Write ($R/\overline{W}$) input is disabled and the Data Output (D_{OUT}) is forced into a high impedance state. When Chip Select goes LOW, the Read / Write is enabled.

When $R/\overline{W}$ goes LOW, data from the Data Input (D_{IN}) is written at the location specified by the Address Inputs (A_n). The Data Output will be identical to the Data Input during a write command. When $R/\overline{W}$ goes HIGH, the contents of the addressed location will appear at D_{OUT}. D_{OUT} is not inverted from D_{IN} in the 2102. (See Truth Table)

DC REQUIREMENTS: DC: $T_A = 0°C$ to $+70°C$; DL: $T_A = -55°C$ to $+85°C$; DM: $T_A = -55°C$ to $+125°C$

SYMBOL	PARAMETER	DC		DL		DM		UNITS	CONDITIONS
		MIN	MAX	MIN	MAX	MIN	MAX		
V_{IH}	Input HIGH Voltage	2.2	V_{DD}	2.0	V_{DD}	2.0	V_{DD}	V	
V_{IL}	Input LOW Voltage	–0.5	0.65	–0.5	0.8	–0.5	0.8	V	

DC CHARACTERISTICS: $V_{SS} = 0$ V; DC: $V_{DD} = 5.0$ V ± 5%, $T_A = 0°C$ to $+70°C$; DL: $V_{DD} = 5.0$ V ± 10%, $T_A = -55°C$ to $+85°C$; DM: $V_{DD} = 5.0$ V ± 10%, $T_A = -55°C$ to $+125°C$

SYMBOL	PARAMETER	DC		DL		DM		UNITS	CONDITIONS
		MIN	MAX	MIN	MAX	MIN	MAX		
V_{OH}	Output HIGH Voltage	2.2		2.2		2.2		V	$I_{OH} = -100\ \mu A$
V_{OL}	Output LOW Voltage		0.4		0.4		0.4	V	$I_{OL} = 2.1$ mA
I_{IN}	Input Leakage Current		10		10		10	μA	$V_{IN} = V_{DD}$
I_{OUT}	Output Leakage Current	–10	10	–10	10		10	μA	$V_{OUT} = 0$ V to V_{DD}, $\overline{CS} = V_{IH}$
I_{DD}	Power Supply Current		50		70		70	mA	

AC REQUIREMENTS: DC: $T_A = 0°C$ to $+70°C$; DL: $T_A = -55°C$ to $+85°C$; DM: $T_A = -55°C$ to $+125°C$

SYMBOL	PARAMETER	2102F DC, DL, DM	21021 DC, DL, DM	21022 DC, DL, DM	2102 DC	UNITS	CONDITIONS
		MIN	MIN	MIN	MIN		
t_{CYC}	Read or Write Cycle Time	350	450	650	1000	ns	$V_{SS} = 0$ V $V_{DD} = +5.0$ V ± 5% For DL, DM: $V_{DD} = 5.0$ V ± 10%
t_{AW}	Address to Write Time	100	170	200	200	ns	
t_{WP}	Write Pulse Width	170	200	350	550	ns	
t_{WR}	Write Recovery Time	50	50	50	50	ns	
t_{DS}	Data Set-up Time	170	200	350	550	ns	
t_{DH}	Data Hold Time	50	50	50	50	ns	
t_{CW}	Chip Enable to Write Time	200	250	400	600	ns	
t_{WC}	Write to Chip Enable time	50	50	50	50	ns	

AC CHARACTERISTICS: DC: $T_A = 0°C$ to $+70°C$; DL: $T_A = -55°C$ to $+85°C$; DM: $T_A = -55°C$ to $+125°C$

SYMBOL	PARAMETER	2102F DC, DL, DM		21021 DC, DL, DM		21022 DC, DL, DM		2102 DC		UNITS	CONDITIONS
		MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX		
t_A	Read Access Time		350		450		650		1000	ns	$V_{SS} = 0$.V $V_{DD} = +5.0$ V ± 5%, For DL, DM: $V_{DD} = 5.0$ V ± 10%,
t_{CO}	Chip Enable to Output Time		180		200		400		500	ns	
t_{OH1}	Data Valid After Address	50		50		50		50		ns	
t_{OH2}	Previous Data Valid After Chip Deselect	0		0		0		0		ns	
C_{IN}	Input Capacitance		5		5		5		5		$V_{IN} = 0$ V, $V_{SS} = 0$ V, f = 1 MHz, $T_A = 25°C$
C_{OUT}	Output Capacitance		10		10		10		10		

WAVEFORMS

READ CYCLE TIMING

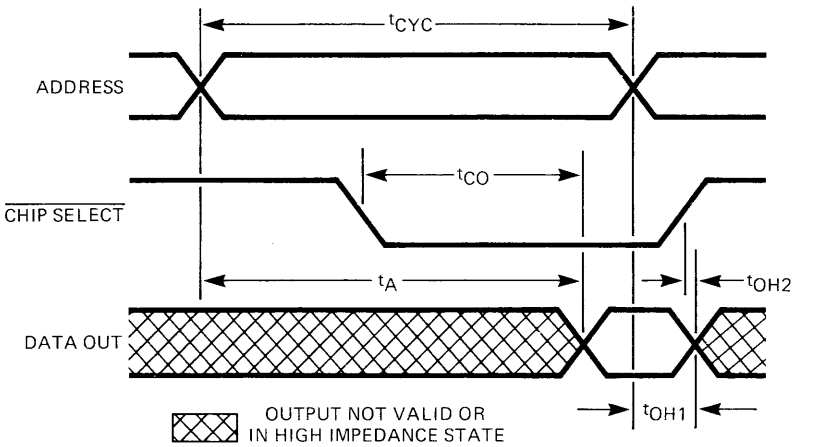


Fig. 1

WRITE CYCLE TIMING

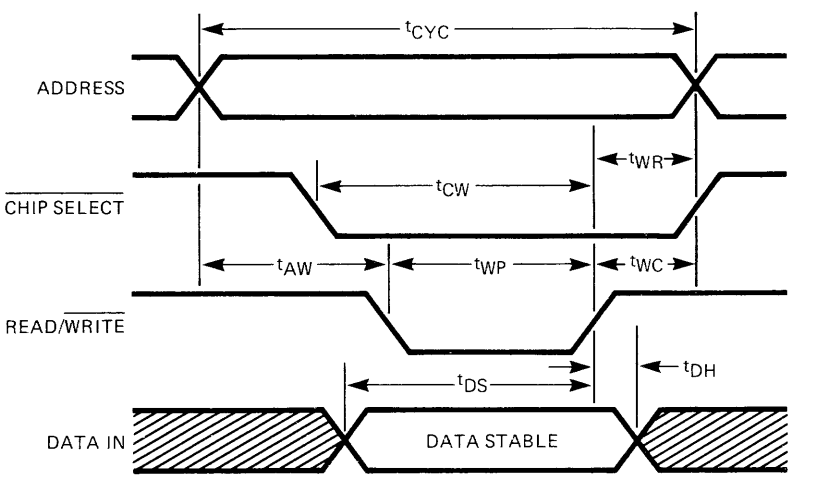


Fig. 2

AC CONDITIONS:

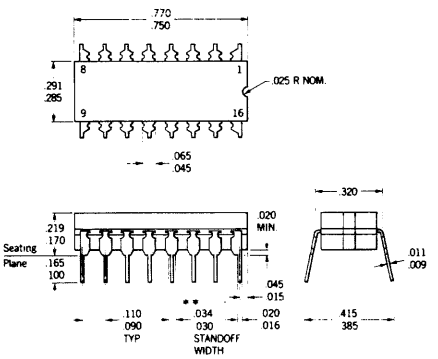
Input Levels: V_{IL} MAX to V_{IH} MIN
Input Rise and Fall Times: 10 ns
Timing Measurement Reference Level: 1.5 V
Output Load: 1 TTL Gate + 100 pF

ORDERING INFORMATION

ACCESS TIME ns	TEMPERATURE RANGE		
	Commercial 0°C to +70°C	Limited Military −55°C to +85°C	Full Military −55°C to +125°C
350	2102FDC	2102FDL	2102FDM
450	21021DC	21021DL	21021DM
650	21022DC	21022DL	21022DM
1000	2102DC		

PHYSICAL DIMENSIONS

6D 16-Pin Ceramic* Dual In-Line Package



NOTES:

All dimensions in inches

Leads are tin-plated kovar

*Package material varies depending on the product line

Leads are intended for insertion in hole rows on .300" centers

They are purposely shipped with "positive" misalignment to facilitate insertion

Board drilling dimensions should equal your practice for .020 inch diameter lead

**The .034-.030 dimension does not apply to the corner leads

Package weight is 2.2 grams

4096

4096×1 DYNAMIC RANDOM ACCESS MEMORY

FAIRCHILD ISOPLANAR SILICON GATE MOS

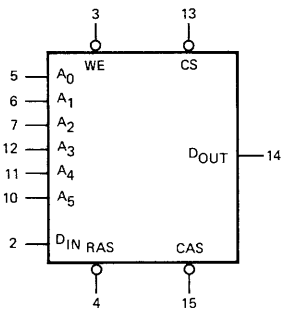
GENERAL DESCRIPTION – The 4096DC is a 4096-bit dynamic Random Access Memory organized as 4096 one-bit words. This device is designed utilizing the single transistor dynamic memory cell.

A unique address multiplexing and latching technique permits the packaging of the 4096DC in a standard 16-pin ceramic Dual In-line Package. The use of this package allows construction of highly dense memory systems utilizing widely available automated testing and insertion equipment.

The 4096DC features direct TTL compatibility, on-chip address, data input and data output latches, TTL-level clocks with extremely low capacitance and a range of access times from 200 ns (4096-2DC) to 350 ns (4096-5DC). The 4096DC is manufactured using the n-channel Isoplanar process.

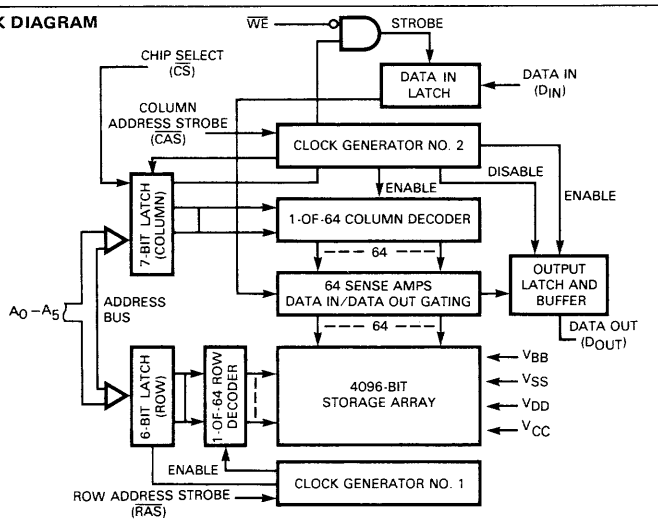
- **ALL INPUTS TTL-COMPATIBLE, INCLUDING CLOCKS**
- **ON-CHIP LATCHES FOR ADDRESSES, CHIP SELECT, DATA INPUT**
- **THREE-STATE TTL-COMPATIBLE OUTPUT**
- **CHIP SELECT DECODING DOES NOT ADD TO ACCESS TIME**
- **READ OR WRITE CYCLES: 4096-2: 300 ns, 4096-3: 360 ns, 4096-4: 420 ns, 4096-5: 500 ns**
- **ACTIVE POWER: 4096-2: <431 mW, 4096-3: <378 mW, 4096-4: <341 mW, 4096-5: <315 mW**
- **STANDBY POWER: <25 mW**
- **STANDARD 16-PIN CERAMIC PACKAGE**

LOGIC SYMBOL

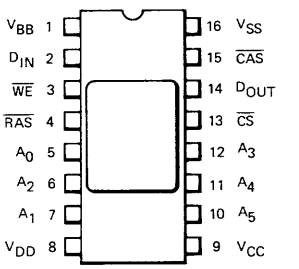


V_{SS} = Pin 16
V_{CC} = Pin 9
V_{DD} = Pin 8
V_{BB} = Pin 1

BLOCK DIAGRAM



CONNECTION DIAGRAM DIP (TOP VIEW)



464 ELLIS STREET, MOUNTAIN VIEW, CALIFORNIA 94042 (415) 962-5011/TWX 910-379-6435

FAIRCHILD SEMICONDUCTOR

PIN NAMES

An	Address Inputs	D_{OUT}	Data Output
D_{IN}	Data Input	V_{CC}	+5 V Power Supply
$\overline{CS}$	Chip Select Input	V_{SS}	0 V Power Supply
$\overline{WE}$	Write Enable Input	V_{BB}	−5 V Power Supply
$\overline{RAS}$	Row Address Strobe (Clock) Input	V_{DD}	+12 V Power Supply
$\overline{CAS}$	Column Address Strobe (Clock) Input		

ABSOLUTE MAXIMUM RATINGS (Note 1)

Voltage of any pin relative to V_{BB}	−0.5 V to +25.0 V
Operating Temperature	0°C to 70°C
Storage Temperature (Ambient)	−55°C to 150°C

ADDRESSING – The 12 address bits required to decode 1 of 4096 cell locations are multiplexed onto the 6 address pins and latched into the on-chip row and column address latches. The Row Address Strobe ($\overline{RAS}$) latches the 6 row address bits onto the chip. The Column Address Strobe ($\overline{CAS}$) latches the 6 column address bits plus Chip Select ($\overline{CS}$) onto the chip. Since the Chip Select signal is not required until well into the cycle, its decoding time does not add to the system access or cycle time.

DATA INPUT/OUTPUT – Data to be written into a selected cell is latched into an on-chip register by a combination of $\overline{WE}$ and $\overline{CAS}$. The last of these signals making its negative transition is the strobe for the Data In register. This permits several options in the write timing. In a write cycle, if the $\overline{WE}$ input is activated prior to $\overline{CAS}$, the Data In is strobed by $\overline{CAS}$ and the set-up and hold times are referenced to this signal. If the cycle is to be a read-write cycle or read-modify-write cycle, then the $\overline{WE}$ input will not go to a logic 0 until after the access time has elapsed. But now, because $\overline{CAS}$ is ready at a logic 0, the Data In is strobed in by $\overline{WE}$ and the set-up hold times are referenced to $\overline{WE}$.

At the beginning of a memory cycle the state of the Data Out Latch and buffer depend on the previous memory cycle. If during the previous cycle the chip was unselected, the output buffer will be in its open-circuit condition. If the previous cycle was a read, read-write, or read-modify-write cycle and the chip was selected, then the output latch and buffer will contain the data read from the selected cell. This output data is the same polarity (not inverted) as the input data. If the previous cycle was a write cycle ($\overline{WE}$ active low before access time) and the chip was selected, then the output latch and buffer will contain a logic 1. Regardless of the state of the output it will remain valid until $\overline{CAS}$ goes negative. At that time the output will unconditionally go to its open-circuit state. It will remain open circuit until after an access time has elapsed. At access time the output will assume the proper state for the type of cycle performed. If the chip is unselected, it will not accept a WRITE command and the output will remain in the open-circuit state.

INPUT/OUTPUT LEVELS – All inputs, including the two address strobes, will interface directly with TTL. The high impedance, low capacitance input characteristics simplify input driver selection by allowing use of standard logic elements rather than specially designed driver elements. Even though the inputs may be driven directly by TTL gates, pull-up or termination resistors are normally required in a system to prevent ringing of the input signals due to line inductance and reflections. In high speed memory systems, transmission line techniques must be employed on the signal lines to achieve optimum system speeds. Series rather than parallel terminations may be employed at some degradation of system speed.

The three-state output buffer is a low impedance to V_{CC} for a logic 1 and a low impedance to V_{SS}, for a logic 0. The resistance to V_{CC} is 500 ohms maximum and 150 ohms typically. The resistance to V_{SS} is 200 ohms maximum and 100 ohms typically. The separate V_{CC} pin allows the output buffer to be powered from the supply voltage of the logic to which chips are interfaced. During battery standby operation, the V_{CC} pin may be unpowered without affecting the 4096 refresh operation. This allows all system logic except the $\overline{RAS}$ timing circuitry and the refresh address logic to be turned off during battery standby to conserve power.

REFRESH – Refresh of the cell matrix is accomplished by performing a memory cycle at each of the 64 row addresses every 2 milliseconds or less. Any read cycle refreshes the selected row, regardless of the state of the Chip Select. A write, read-write, or read-modify-write cycle also refreshes the selected row but the chip should be unselected to prevent writing data into the selected cell.

POWER DISSIPATION/STANDBY MODE – Most of the circuitry used in the 4096 is dynamic and draws power only as the result of an address strobe edge. Because the power is not drawn during the whole time the strobe is active, the dynamic power is a function of operating frequency. Typically, the power is 120mW at a 1 μS cycle time for the 4096DC with a worst case power of less than 341 mW at a 420 ns cycle time. To reduce the overall system power the Row Address Strobe ($\overline{RAS}$) must be decoded and supplied to only the selected chips. The $\overline{CAS}$ must be supplied to all chips (to turn off the unselected outputs). But those chips that did not receive a $\overline{RAS}$ will not dissipate any power on the $\overline{CAS}$ edges, except for that required to turn off the output. If the $\overline{RAS}$ is decoded and supplied to the selected chips, then the Chip Select input of all chips can be at a logic 0. The chips that receive a $\overline{CAS}$ but no $\overline{RAS}$ will be unselected (output open-circuited) regardless of the Chip Select input.

RECOMMENDED DC OPERATING CONDITIONS ($0°C \leq T_A \leq 70°C$)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	NOTES
V_{DD}	Supply Voltage	11.4	12.0	12.6	V	2
V_{CC}	Supply Voltage	4.5	5.0	V_{DD}	V	2
V_{SS}	Supply Voltage	0	0	0	V	2,12
V_{BB}	Supply Voltage	−5.5	−5.0	−4.5	V	2
V_{IH1}	Input HIGH Voltage Address Input	2.4	5.0	V_{GG}	V	2,14
V_{IL}	Input LOW Voltage, All Inputs	−1.0	0	0.6	V	2,14
V_{IH2}	Input HIGH Voltage, $\overline{RAS}$, $\overline{CAS}$, $\overline{CS}$, $\overline{WE}$	2.7	5.0	V_{GG}	V	2,14

DC ELECTRICAL CHARACTERISTICS ($0°C \leq T_A \leq 70°C$) (V_{DD} = 12.0 V ± 5%, V_{CC} = 5.0 V ± 10%, V_{SS} = 0 V, V_{BB} = −5.0 V ± 10%)

SYMBOL	PARAMETER	PART NUMBER								UNITS	NOTES
		4096-2		4096-3		4096-4		4096-5			
		MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX		
I_{DD1}	Average V_{DD} Power Supply Current		35		30		27		25	mA	16
I_{CC}	V_{CC} Power Supply Current									mA	9
I_{BB}	Average V_{BB} Power Supply Current		75		75		75		75	µA	
I_{DD2}	Standby V_{DD} Power Supply Current		2		2		2		2	mA	
I_{IN}	Input Leakage Current (Any Input)		10		10		10		10	µA	10
I_{OUT}	Output Leakage Current		10		10		10		10	µA	11
V_{OH}	Output HIGH Voltage at I_{OUT} = −5 mA	2.4		2.4		2.4		2.4		V	
V_{OL}	Output LOW Voltage at I_{OUT} = 2 mA		0.4		0.4		0.4		0.4	V	
C_{IN1}	Input Capacitance ($A_0 - A_5$)		10		10		10		10	pF	
C_{IN2}	Input Capacitance ($\overline{RAS}$, $\overline{CAS}$, D_{IN}, $\overline{WE}$, $\overline{CS}$)		7		7		7		7	pF	
C_{OUT}	Output Capacitance (D_{OUT})		8		8		8		8	pF	

RECOMMENDED AC OPERATING CONDITIONS ($0°C \leq T_A \leq 70°C$) (V_{DD} = 12.0 V ± 5%; V_{CC} = 5.0 V ± 10%, V_{SS} = 0 V, V_{BB} = −5.0 V ± 10%) (Note 17)

SYMBOL	PARAMETER	PART NUMBER								UNITS	NOTES
		4096-2		4096-3		4096-4		4096-5			
		MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX		
t_{RC}	Random Read or Write Cycle Time	300		365		425		500		ns	3
t_{RAC}	Access Time from ROW Address Strobe		200		250		300		350	ns	3, 15
t_{CAC}	Access Time from Column Address Strobe		120		150		175		200	ns	4
t_{OFF}	Output Buffer Turn-Off Delay	0	70	0	80	0	90	0	100	ns	4
t_{RP}	ROW Address Strobe Precharge Time	100		115		125		150		ns	
t_{RCL}	ROW to Column Strobe Lead Time	80		100		125		150		ns	3
t_{CPW}	Column Address Strobe Pulse Width	120		150		175		200		ns	
t_{AS}	Address Set-Up Time	0		0		0		0		ns	3, 4
t_{AH}	Address Hold Time	50		60		70		80		ns	3, 4
t_{CH}	Chip Select Hold Time	70		80		90		100		ns	
t_{RCS}	Read Command Set-Up Time	0		0		0		0		ns	4
t_{RCH}	Read Command Hold Time	30		35		40		45		ns	5
t_{WCH}	Write Command Hold Time	90		110		140		150		ns	4, 6
t_{WP}	Write Command Pulse Width	120		150		175		200		ns	
t_{CRL}	Column to ROW Strobe Lead Time	-20		-20		−20	+20	-20		ns	7
t_{CWL}	Write Command to Column Strobe Lead Time	120		150		175		200		ns	13
t_{DS}	Data In Set-Up Time	0		0		0		0		ns	13
t_{DH}	Data In Hold Time	90		110		130		150		ns	13
t_{RFSH}	Refresh Period		2		2		2		2	ms	
t_{MOD}	Modify Time		10		10		10		10	µs	8

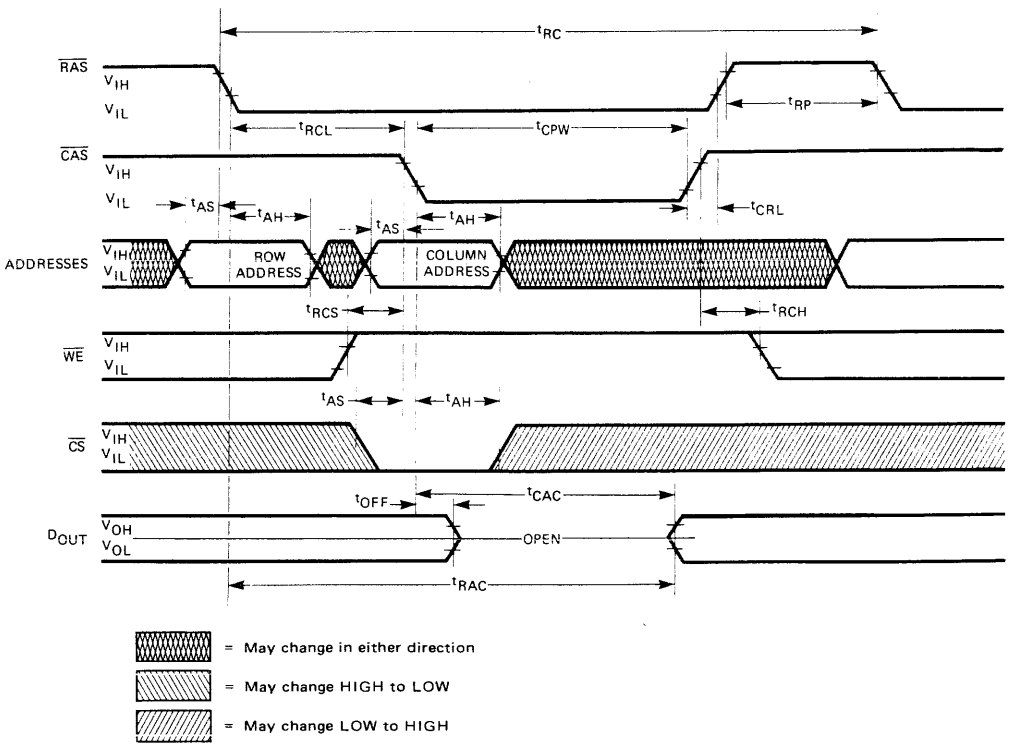
Notes on following page.

NOTES:

1. Stresses greater than those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.
2. All voltages referenced to V_{SS}.
3. Referenced to $\overline{RAS}$ leading edge.
4. Referenced to $\overline{CAS}$ leading edge.
5. Referenced to $\overline{CAS}$ trailing edge.
6. Write Command Hold Time is important only when performing normal random write cycles. During read-write or read-modify-write cycles, the Write Command Pulse Width is the limiting parameter.
7. Referenced to the $\overline{RAS}$ trailing edge.
8. Referenced to access time.
9. Depends upon output loading. The V_{CC} supply is connected only to the output buffer.
10. All device pins at 0 volts except V_{BB} at −5 volts and pin under test which is at +10 volts.
11. Output disabled by chip select input.
12. Output voltage will swing from V_{SS} to V_{CC} independent of differential between V_{SS} and V_{CC}.
13. These parameters are referenced to the $\overline{CAS}$ leading edge in random write cycle operation and to the $\overline{WE}$ leading edge in read-write or read-modify-write cycles.
14. Input voltages greater than TTL levels (0 to 5 V) require device operation at reduced speed.
15. Assumes t_{RCL} minimum.
16. Current is proportional to speed with maximum current measured at fastest cycle rate.
17. AC measurements assume ≈10 ns rise and fall times.

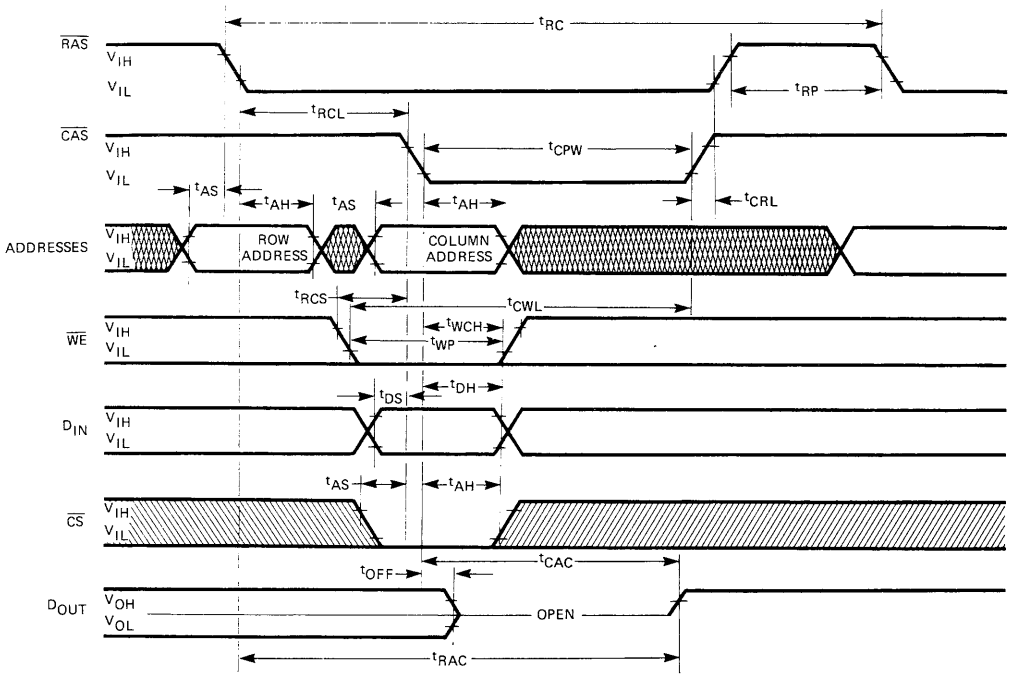
TIMING DIAGRAMS

READ CYCLE

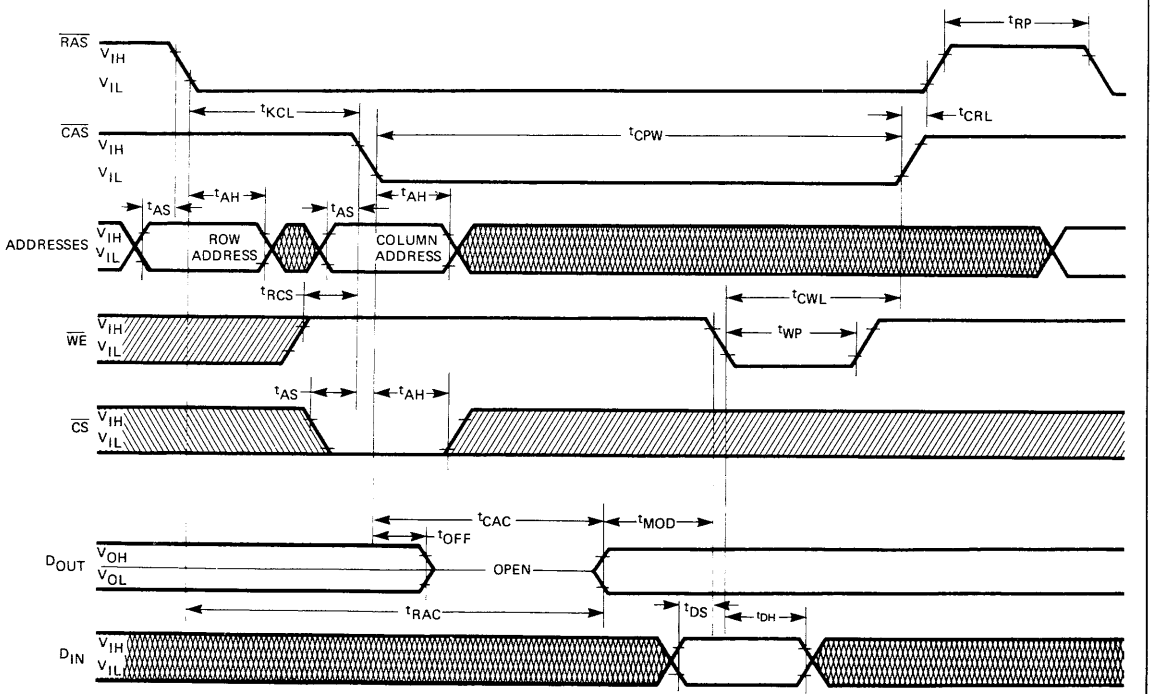


TIMING DIAGRAMS (Cont'd)

WRITE CYCLE



READ-MODIFY-WRITE CYCLE*

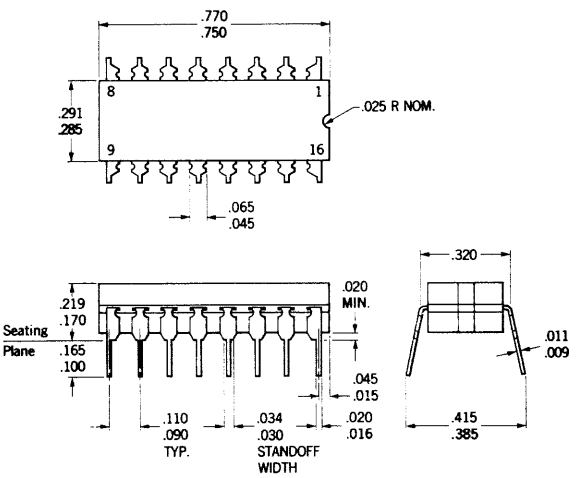


*Read-modify-write cycle time = $t_{RCL} + t_{CAC} + t_{MOD} + t_{CWL} + t_{CRL} + t_{RP} + 3\,t_f + t_r$.

ORDERING INFORMATION

Part Number	Access Time
40962DC	200 ns
40963DC	250 ns
40964DC	300 ns
40965DC	350 ns

PACKAGE OUTLINES
16-Pin Dual In-line



NOTES:
1. Pins are tin-plated kovar
2. Pins are intended for insertion in hole rows on .300" centers
3. They are purposely shipped with "positive" misalignment to facilitate insertion
4. Board-drilling dimensions should equal your practice for .020 inch diameter lead
5. Hermetically sealed alumina package
6. Cavity size is .130 x .230
7. The .034-.030 dimension does not apply to the corner leads
8. Package weight is 2.2 grams

93436 • 93446
ISOPLANAR SCHOTTKY TTL MEMORY
512×4-BIT PROGRAMMABLE READ ONLY MEMORY

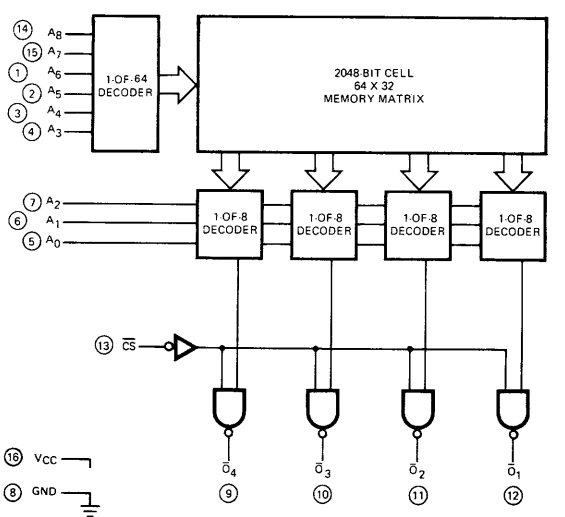
DESCRIPTION — The Fairchild 93436 and 93446 are fully decoded high speed 2048-bit field programmable ROMs organized 512 words by four bits per word. The devices are identical except for the output stage. The 93436 has uncommitted collector outputs, while the 93446 has 3-state outputs. In either case, the outputs are off when the $\overline{CS}$ input is in the HIGH state. The 93436 and 93446 are supplied with all bits stored as logic "1"s and can be programmed to logic "0"s by following the field programming procedure.

- FULL MIL AND COMMERCIAL RANGES
- FIELD PROGRAMMABLE
- ORGANIZATION — 512 WORDS x FOUR BITS
- UNCOMMITTED COLLECTORS — 93436
- 3-STATE OUTPUTS — 93446
- FULLY DECODED — ON-CHIP ADDRESS DECODER AND BUFFER
- CHIP SELECT INPUT PROVIDES EASY MEMORY EXPANSION
- WIRED-OR CAPABILITY
- STANDARD 16-PIN DUAL IN-LINE PACKAGE
- NICHROME FUSE LINKS
- REPLACES TWO 256 x 4 PROMS — DOUBLE DENSITY WITH SAME SPACE AND POWER

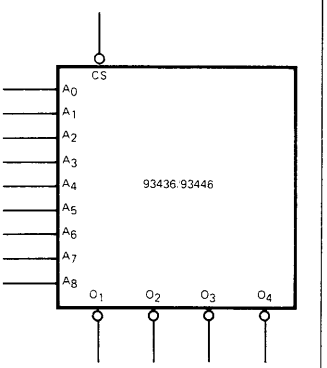
PIN NAMES

$A_0 - A_8$	Address Inputs
$\overline{CS}$	Chip Select Input
$\overline{O_1} - \overline{O_4}$	Data Outputs

LOGIC DIAGRAM

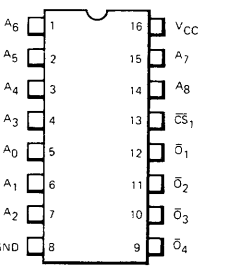


LOGIC SYMBOL



V_{CC} = Pin 16
GND = Pin 8

CONNECTION DIAGRAM
DIP(TOP VIEW)



NOTE:
The Flatpak version has the same pinouts(Connection Diagram) as the Dual In-Line Package.

464 ELLIS STREET, MOUNTAIN VIEW, CALIFORNIA 94042 (415) 962-5011/TWX 910-379-6435

FAIRCHILD SEMICONDUCTOR

 Printed in U.S.A. 2029-11-0043-065 7.5M

Manufactured under one or more of the following U.S. Patents: 2981877, 3015048, 3025589, 3064167, 3108359, 3117260; other patents pending.

FUNCTIONAL DESCRIPTION – The 93436 and 93446 are bipolar field Programmable Read Only Memories (PROMs) organized 512 words by four bits per word. Open collector outputs are provided on the 93436 for use in wired-OR systems. The 93446 has 3-state outputs which provide active pull ups when enabled and high output impedance when disabled. Chip Select for both devices is active LOW; i.e., a HIGH (logic "1") on the $\overline{CS}$ pin will disable all outputs.

The read function is identical to that of a conventional bipolar ROM. That is, a binary address is applied to the A_0 through A_8 inputs, the chip is selected, and data is valid at the outputs after t_{AA} nanoseconds.

Programming (selectively opening nichrome fuse links) is accomplished by following the sequence outlined below.

PROGRAMMING – The 93436 and 93446 are manufactured with all bits in the logic "1" state. Any desired bit (output) can be programmed to a logic "0" state by following the procedure shown below. One may build a programmer to satisfy the specifications or buy any of the commercially available programmers which meet these specifications.

PROGRAMMING SPECIFICATIONS

PARAMETER	SYMBOL	MIN	RECOMMENDED VALUE	MAX	UNITS	COMMENTS
Address Input	V_{IH}	2.4	5.0	5.0	V	Do not leave inputs open
	V_{IL}	0	0	0.4	V	
Chip Select	$\overline{CS}$	2.4	5.0	5.0	V	
Programming Voltage Pulse	V_{OP}	19	20	21	V	Applied to output to be programmed
Programming Pulse Width	t_{pw}	0.18		50	ms	All bits can be programmed in ⩽ 2.0 seconds
Duty Cycle Programming Pulse			20	*	%	*Maximum duty cycle to maintain $T_C < 85°C$
Programming Pulse Rise Time	t_r	1.0		10	μs	
Power Supply Voltage	V_{CC}	4.75	5.0	5.25	V	
Case Temperature	t_c		25	85	°C	
Programming Pulse Current	I_{OP}			100	mA	If pulse generator is used, set current limit to this max value

PROGRAMMING SEQUENCE – The Fairchild 93436/93446 may be programmed using the following method.

1. Apply the proper power, V_{CC} = 5.0 V, GND = 0 V.
2. Select the word to be programmed by applying the appropriate voltages to the address pins A_0 through A_8.
3. Enable the chip for programming by application of a "HIGH" (logic "1") to Chip Select ($\overline{CS}$), pin 13.
4. Apply the 20 V programming pulse to the output associated with the bit to be programmed. The other outputs may be left open or tied to any logic "1" (output HIGH)i.e., 2.4 V to 4.0 V. Note that only one output may be programmed at a time.
5. To verify the logic "0" in the bit just programmed, remove the programming pulse from the output and sense it after applying a logic "0" to Chip Select ($\overline{CS}$).
6. The above procedure is then repeated to program other bits on the chip.

BOARD PROGRAMMING– To program a single PROM out of a group of "OR" tied memories the following procedure is required:

1. Connect the outputs of a TTL Decoder (supplied by V_{CC} = +12.8 V, V_{EE} = +7.8 V) to the $\overline{CS}$ pin of the memories on the board.
2. Address the decoder such that the particular decoder output connected to the $\overline{CS}$ pin of the memory to be programmed will be "LOW" at +7.8 V. All the other decoder outputs will be "HIGH" at +10.8 V.
3. Apply the 20 V programming pulse to one group of "OR" tied outputs selected for programming; only the addressed bit in the +7.8 V selected memory will program; all other memories remain deselected (those with $\overline{CS}$ = +10.8 V).
4. To verify the logic "0" in the bit just programmed remove programming pulse and sense the "OR" tie after lowering the decoder supplies to the conventional V_{CC} = +5.0 V, V_{EE} = 0 V.

ABSOLUTE MAXIMUM RATINGS

Storage Temperature	−65°C to +150°C
Temperature (Ambient) Under Bias	−55°C to +125°C
V_{CC} Pin Potential to Ground	−0.5 V to +7.0 V
Input Voltage	−0.5 V to +5.5 V
Current Into Output Terminal	100 mA
Output Voltages	−0.5 V to 4.0 V

GUARANTEED OPERATING RANGES

PART NUMBER	SUPPLY VOLTAGE (V_{CC})			AMBIENT TEMPERATURE
	MIN	TYP	MAX	
93436XC, 93446XC	4.75 V	5.0 V	5.25 V	0°C to +75°C
93436XM, 93446XM	4.50 V	5.0 V	5.50 V	−55°C to +125°C

X = package type; F for Flatpak, D for Ceramic DIP, P for Plastic DIP. See Package Information on this data sheet.

DC CHARACTERISTICS: Over guaranteed operating ranges unless otherwise noted.

SYMBOL	CHARACTERISTIC	LIMITS MIN	LIMITS TYP (Note 1)	LIMITS MAX	UNITS	CONDITIONS
I_{CEX}	Output Leakage Current (93436 only)			50	µA	V_{CC} = MAX, V_{CEX} = 4.0 V, 0°C to +75°C Address any HIGH Output
I_{CEX}	Output Leakage Current (93436 only)			100	µA	V_{CC} = MAX, V_{CEX} = 4.0 V, −55°C to +125°C Address any HIGH Output
V_{OL}	Output LOW Voltage		0.30	0.45	V	V_{CC} = MIN, I_{OL} = 16 mA, A_0 = +10.8 V A_1 through A_8 = HIGH
V_{OH}	Output HIGH Voltage (93446 only)	2.4			V	V_{CC} = MIN, I_{OH} = −2.0 mA
I_{off}	Output Leakage Current for HIGH Impedance State (93446 only)			50 −50	µA µA	V_{OH} = 2.4 V V_{OL} = 0.4 V 0°C to +75°C
I_{off}	Output Leakage Current for HIGH Impedance State (93446 only)			100 −50	µA µA	V_{OH} = 2.4 V V_{OL} = 0.4 V −55°C to +125°C
V_{IH}	Input HIGH Voltage	2.0			V	Guaranteed Input HIGH Voltage for All Inputs
V_{IL}	Input LOW Voltage			0.8	V	Guaranteed Input LOW Voltage for All Inputs
I_F	Input LOW Current I_{FA} (Address Inputs) I_{FCS} (Chip Select Inputs)		 −160 −160	 −250 −250	 µA µA	V_{CC} = MAX, V_F = 0.45 V
I_R	Input HIGH Current I_{RA} (Address Inputs) I_{RCS} (Chip Select Input)			 40 40	 µA µA	V_{CC} = MAX, V_R = 2.4 V
I_{CC}	Power Supply Current		95	130	mA	V_{CC} = MAX, Outputs open Inputs Grounded and Chip Selected
C_O	Output Capacitance		7		pF	V_{CC} = 5.0 V, V_O = 4.0 V, f = 1.0 MHz
C_{IN}	Input Capacitance		4		pF	V_{CC} = 5.0 V, V_O = 4.0 V, f = 1.0 MHz
V_C	Input Clamp Diode Voltage			−1.2	V	V_{CC} = MIN, I_A = −18 mA

AC CHARACTERISTICS: T_A = 0°C to +75°C, V_{CC} = 5.0 V ± 5%.

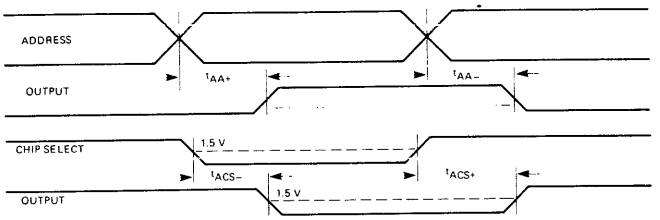
SYMBOL	CHARACTERISTIC	LIMITS MIN	LIMITS TYP (Note 1)	LIMITS MAX	UNITS	CONDITIONS
t_{AA-}	Address to Output Access Time		30	50	ns	See Figure 1A and 1B
t_{AA+}			30	50	ns	
t_{ACS-}	Chip Select Access Time		15	25	ns	
t_{ACS+}			15	25	ns	

AC CHARACTERISTICS: T_A = −55°C to +125°C, V_{CC} = 5.0 V ± 10%.

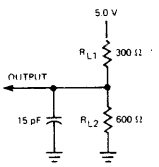
SYMBOL	CHARACTERISTIC	LIMITS MIN	LIMITS TYP (Note 1)	LIMITS MAX	UNITS	CONDITIONS
t_{AA-}	Address to Output Access Time		30	60	ns	See Figure 1A and 1B
t_{AA+}			30	60	ns	
t_{ACS-}	Chip Select Access Time		15	30	ns	
t_{ACS+}			15	30	ns	

Note 1: Typical limits are at V_{CC} = 5.0 V, +25°C and max loading.

SWITCHING WAVEFORMS

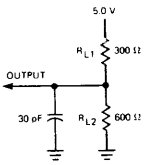


SWITCHING TEST OUTPUT LOAD



15 mA Load
Applies to 93436 Only

Fig. 1A



15 mA Load
Applies to 93446 Only

Fig. 1B

PROM PROGRAMMING CIRCUIT

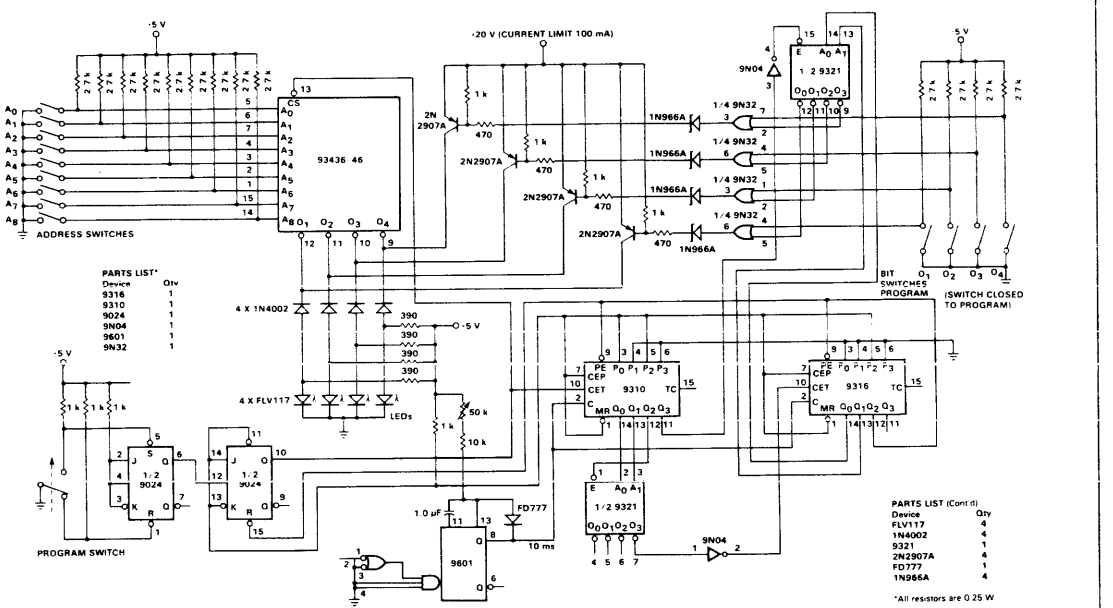
This circuit will sequentially program all four bits of a given word address of the 93436 or 93446. Selection of the bit patterns to be programmed is made by the address bit switches.

Until the program switch is depressed, the contents of the 93436 or 93446 at the address set in the address switch register is displayed on the four FLV117 LEDs. If the content is a Logic "1" or the chip is deselected, the LED is turned on with current supplied by the 390 Ω resistors. If the content of the PROM is a Logic "0" and the PROM is enabled, the output is Logic "0" turning the LEDs off. The 1N4002s isolate the LEDs from the 20.V programming pulse.

The 9601 is a one shot continuous 10 ms oscillator. One half of a 9024 is used as a switch debouncer while the other half is the "run" flip-flop. When the program is initiated by depressing the program switch, the first half of the 9024 (switch debouncer) is set and clocks the other half of the 9024 ("run" flip-flop) to the "run" state. This enables the counters to operate and disables the 93436. The counter is preset to 5 on the 9310 and 4 on the 9316. The counter provides the proper duty cycle and program timing.

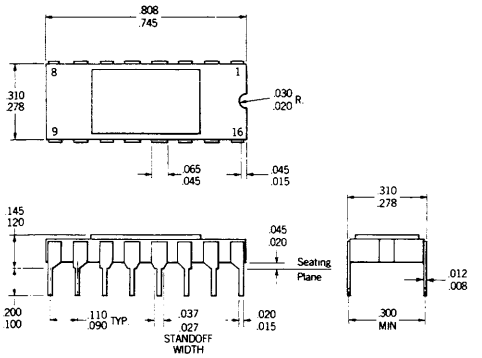
To avoid overlap problems between the programming pulse, the chip enable and the scan, the 9316 advances when the 9310 goes from state 3 to state 4. When the last bit has been programmed, the counter presets itself and resets the "run" flip-flop. The programming sequence is now complete.

The bit to be programmed is decoded by one-half of the 9321 and wired-OR with the bit switch. The OR gate is a high voltage driver supplying the drive to the programming transistors.



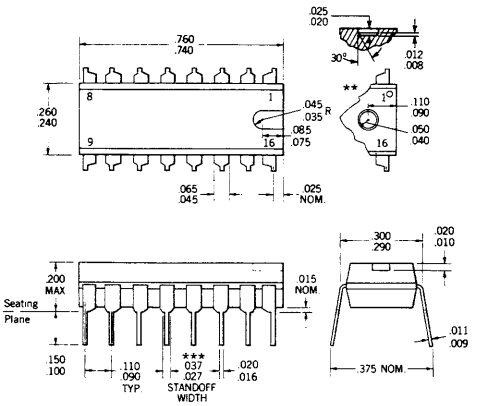
16-PIN CERAMIC DUAL IN-LINE PACKAGE (D)

(METAL CAP)



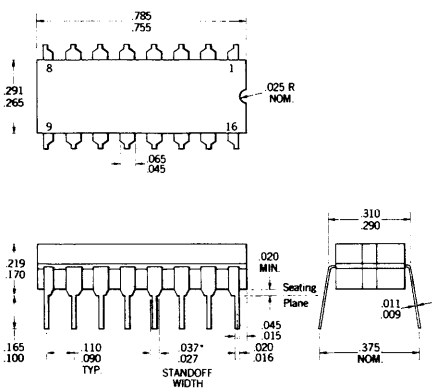
NOTES:
All dimensions in inches
Leads are intended for insertion in hole rows on .300" centers
Board-drilling dimensions should equal your practice for .020 inch diameter lead
Leads are gold-plated kovar
Package weight is 1.3 grams

16-PIN PLASTIC DUAL IN-LINE (P) *



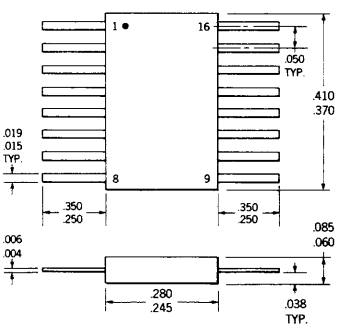
NOTES:
All dimensions in inches
Leads are tin-plated kovar
*Package material varies depending on the product line
Leads are intended for insertion in hole rows on .300" centers
They are purposely shipped with "positive" misalignment to facilitate insertion
Board-drilling dimensions should equal your practice for .020 inch diameter lead
**Notch or ejector hole varies depending on the product line
***The .037/.027 dimension does not apply to the corner leads
Package weight is 0.9 gram

16-PIN CERAMIC DUAL IN-LINE PACKAGE (D)



NOTES:
All dimensions in inches
Leads are tin-plated kovar
Leads are intended for insertion in hole rows on .300" centers
Board-drilling dimensions should equal your practice for .020 inch diameter lead
*The .037 / .027 dimension does not apply to the corner leads
Package weight is 2.2 grams

16-PIN BeO FLATPAK (F)



NOTES:
All dimensions in inches
Leads are gold-plated kovar
Package weight is 0.4 gram

ORDERING INFORMATION

ORDER CODE	PACKAGE	TEMPERATURE
93346DC, 93446DC	Ceramic DIP**	Commercial
93436DM, 93446DM	Ceramic DIP**	Military
93436PC, 93446PC	Plastic DIP	Commercial
93436FM, 93446FM	Flatpak	Military

*Commercial = 0°C to +75°C
Military = −55°C to +125°C

**Available in both Ceramic DIP package options

93438/93448
ISOPLANAR SCHOTTKY TTL MEMORY
512×8-BIT PROGRAMMABLE READ ONLY MEMORY

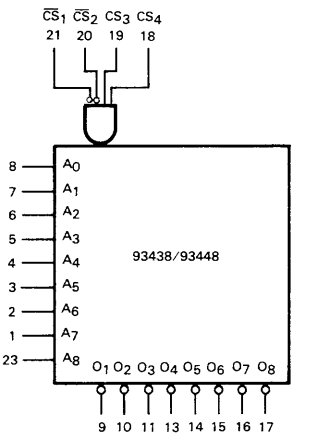
DESCRIPTION – The 93438 and 93448 are fully decoded 4096-bit field programmable ROMs organized 512 words by eight bits per word. The devices are identical except for the output stage. The 93438 has uncommitted collector outputs, while the 93448 has 3-state outputs. Either device is enabled when $\overline{CS}_1$ and $\overline{CS}_2$ are LOW and CS_3 and CS_4 are HIGH. The 93438/48 is supplied with all bits stored as logic "1's" and may be programmed to logic "0's" by following the field programming procedure.

- FULL MIL AND COMMERCIAL RANGES
- FIELD PROGRAMMABLE
- ORGANIZATION – 512 WORDS X EIGHT BITS
- UNCOMMITTED COLLECTORS – 93438
- 3-STATE OUTPUTS – 93448
- FULLY DECODED – ON-CHIP ADDRESS DECODER AND BUFFER
- CHIP SELECT INPUTS PROVIDE EASY MEMORY EXPANSION
- WIRED-OR CAPABILITY
- STANDARD 24-PIN DUAL IN-LINE PACKAGE
- NICHROME FUSE LINKS
- REPLACES TWO 256 X 8 PROMs – DOUBLE DENSITY WITH SAME SPACE AND POWER

PIN NAMES

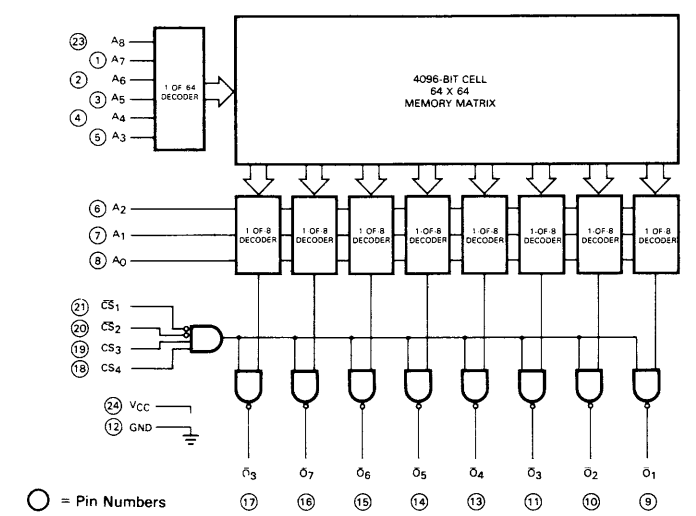
$A_0 - A_8$	Address Inputs
$\overline{CS}_1$, $\overline{CS}_2$, CS_3, CS_4	Chip Select Inputs
$\overline{O}_1 - \overline{O}_8$	Data Outputs

LOGIC SYMBOL



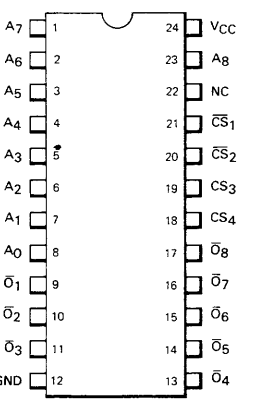
V_{CC} = Pin 24
GND = Pin 12

LOGIC DIAGRAM



○ = Pin Numbers

CONNECTION DIAGRAM
DIP (TOP VIEW)



FAIRCHILD SEMICONDUCTOR

464 ELLIS STREET, MOUNTAIN VIEW, CALIFORNIA 94042 (415) 962-5011/TWX 910-379-6435

 Printed in U.S.A. 2029-11-0035-065 7.5M

ABSOLUTE MAXIMUM RATINGS

Storage Temperature	−65°C to +150°C
Temperature (Ambient) Under Bias	−55°C to +125°C
V_{CC}	−0.5 V to +7.0 V
Input Voltage	−0.5 V to +5.5 V
Current into Output Terminal	100 mA
Output Voltages	−0.5 V to 4.0 V

GUARANTEED OPERATING RANGES

PART NUMBERS	SUPPLY VOLTAGE (V_{CC})			AMBIENT TEMPERATURE
	MIN	TYP	MAX	
93438DC, 93448DC	4.75 V	5.0 V	5.25 V	0°C to +75°C
93438DM, 93448DM	4.50 V	5.0 V	5.50 V	−55°C to +125°C

FUNCTIONAL DESCRIPTION – The 93438 and 93448 are bipolar field Programmable Read Only Memories (PROMs) organized 512 words by eight bits per word. Open collector outputs are provided on the 93438 for use in wired-OR systems. The 93448 has 3-state outputs which provide active pull-ups when enabled and high output impedance when disabled. Chip Select for both devices follows the logic equation: $\overline{CS}_1 \bullet \overline{CS}_2 \bullet CS_3 \bullet CS_4 = CS$; i.e., if $\overline{CS}_1$ and $\overline{CS}_2$ are both active LOW and CS_3 and CS_4 are both active HIGH, all eight outputs are enabled; for any other condition all eight outputs are disabled.

The read function is identical to that of a conventional bipolar ROM. That is, a binary address is applied to the A_0 through A_8 inputs, the chip is selected, and data is valid at the outputs after t_{AA} nanoseconds.

Programming (selectively opening nichrome fuse links) is accomplished by following the sequence outlined below.

PROGRAMMING – The 93438 and 93448 are manufactured with all bits in the logic "1" state. Any desired bit (output) can be programmed to a logic "0" state by following the procedure shown below. One may build a programmer to satisfy the specifications or purchase any of the commercially available programmers which meet these specifications.

PROGRAMMING SPECIFICATIONS

PARAMETER	SYMBOL	MIN	RECOMMENDED VALUE	MAX	UNITS	COMMENTS	
Address Input	V_{IH}	2.4	5.0	5.0	V	Do not leave inputs open	
	V_{IL}	0	0	0.4	V		
Chip Select	$\overline{CS}_1, \overline{CS}_2$	2.4	5.0	5.0	V	Pin 20 or 21 or both	Either or both
	CS_3, CS_4	0	0	0.4	V	Pin 18 or 19 or both	
Programming Voltage Pulse	V_{OP}	19	20	21	V	Applied to output to be programmed	
Programming Pulse Width	t_{pw}	0.18		50	ms	All bits can be programmed in ⩽ 4.1 sec.	
Duty Cycle Programming Pulse			20	*	%	*Maximum duty cycle to maintain $T_C < 85°C$	
Programming Pulse Rise Time	t_r	1.0		10	μs		
Power Supply Voltage	V_{CC}	4.75	5.0	5.25	V		
Case Temperature	t_c		25	85	°C		
Programming Pulse Current	I_{OP}			100	mA	If pulse generator is used, set current limit to this max value.	

PROGRAMMING SEQUENCE – The Fairchild 93438/93448 may be programmed using the following method.

1. Apply the proper power, V_{CC} = 5.0 V, GND = 0 V.
2. Select the word to be programmed by applying the appropriate voltages to the address pins A_0 through A_8.
3. Enable the chip for programming by application of a HIGH (logic "1") to Chip Select $\overline{CS}_1$, (Pin 21) or $\overline{CS}_2$, (Pin 20), or by application of a LOW (logic "0") to Chip Select CS_3, (Pin 19) or CS_4, (Pin 18).
4. Apply the 20 V programming pulse to the output associated with the bit to be programmed. The other outputs may be left open or tied to any logic "1" (output HIGH), i.e., 2.4 V to 4.0 V. Note that only one output may be programmed at a time.
5. To verify the logic "0" in the bit just programmed, remove the programming pulse from the output and sense it after applying a logic "0" to Chip Selects $\overline{CS}_1$ and $\overline{CS}_2$ and a logic "1" to Chip Selects CS_3 and CS_4.
6. The above procedure is then repeated to program other bits on the chip.

BOARD PROGRAMMING

CASE A: For systems using Pins $\overline{CS}_1$, $\overline{CS}_2$ refer to Figure 1.

1. Memories 1 through 4 are OR-tied and connected to the programmer as shown.
2. Connect CS_3 (Pin 19) and CS_4 (Pin 18) of all the memories to a HIGH (logic "1") or leave unconnected.
3. Connect $\overline{CS}_2$ (Pin 20) of all memories to ground.
4. Connect outputs of the TTL Decoder to $\overline{CS}_1$s (Pins 21) of the four memories on the board.
5. To program a bit in one of the four memories, connect the decoder supply voltages to V_{CC} = +12.6 V and V_{EE} = +7.6 V and select an Address A_0, A_1 (HIGH = +10.6 V; LOW = +7.6 V).
6. Raise the programming voltage to 20 V; the memory whose $\overline{CS}_1$ is LOW at +7.8 V (Memory 4 in Figure 1) will program, all others with $\overline{CS}_1$ HIGH at +10.6 V will remain deselected.
7. To verify the logic "0" in the bit just programmed remove the programming pulse and sense the OR-tie after simultaneously lowering the decoder supplies to V_{CC} = 5.0 V, V_{EE} = 0 V and shifting the decoder address A_0, A_1 down to its normal levels (HIGH = 3.0 V; LOW = 0 V).
8. Repeat procedure for other bits following the normal programming sequence.
9. To select a different memory on the board change decoder address A_0, A_1.

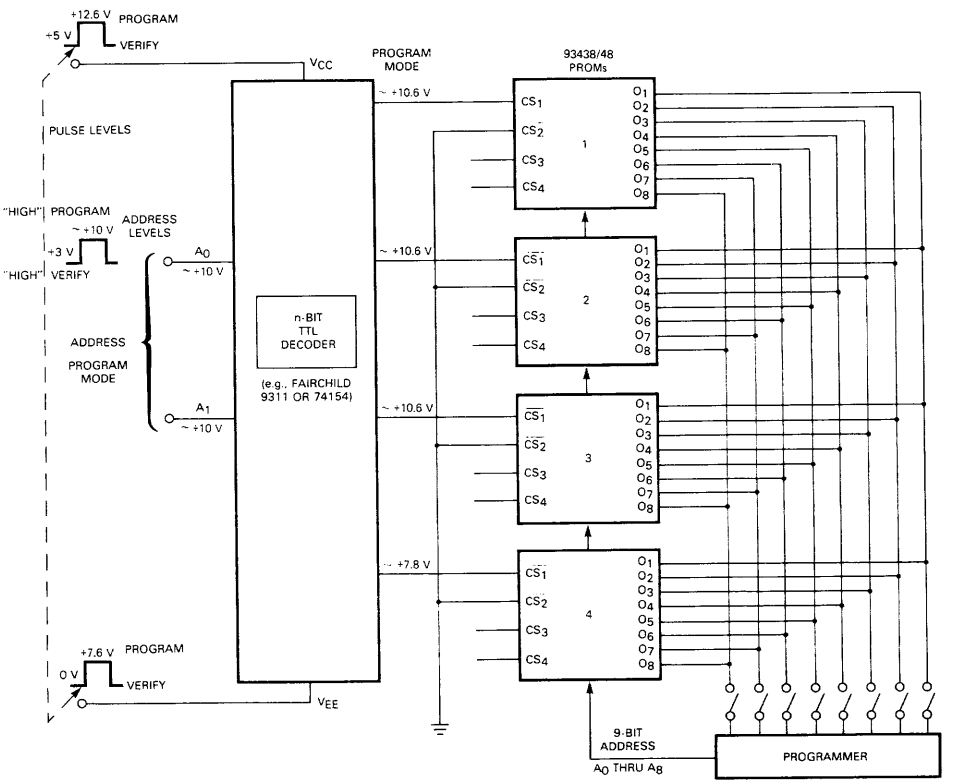


Fig. 1 Case A

CASE B: For systems using CS_3, CS_4 refer to Figure 2.

1. Memories 1 through 4 are OR-tied and connected to the programmer as shown.
2. Connect $\overline{CS}_1$ (Pin 21) and $\overline{CS}_2$ (Pin 20) of all memories to ground.
3. Connect CS_3 (Pin 19) of all memories to outputs of the "High Level Decoder" (supplied by V_{CC} = +12.6 V, V_{EE} = +7.6 V).
4. Connect CS_4 (Pin 18) of all memories via positive AND Gates (7408) and Inverters (9016) to the outputs of the "Low Level Decoder" (supplied by V_{CC} = +5.0 V; V_{EE} = 0 V).
5. The input addresses to both decoders, A_0, A_1 and $A_0{}^1$, $A_1{}^1$, are identical in the "logic" sense but differ in level by ~ 7.6 V.
6. The level of the latch pulse into the common terminal of the AND Gates determines whether the selected memory is being programmed or verified.
7. To program a bit in one of the four memories select the desired address A_0, A_1 of the "High Level Decoder" and select the "program mode" by applying a LOW level (0 V) to the latch input.
8. Raise the programming voltage to 20 V; the memory whose CS_3 is "LOW" at +7.8 V (Memory 4 in Figure 2) will program; all others with CS_3 HIGH at +10.6 V will remain deselected.
9. To verify the logic "0" in the bit just programmed remove the programming pulse and sense the OR-tie after raising the latch input level to a HIGH (+3.0 V).
10. Repeat the procedure for other bits following the normal programming sequence.
11. To select a different memory on the board change the "High Level Decoder" address A_0, A_1.

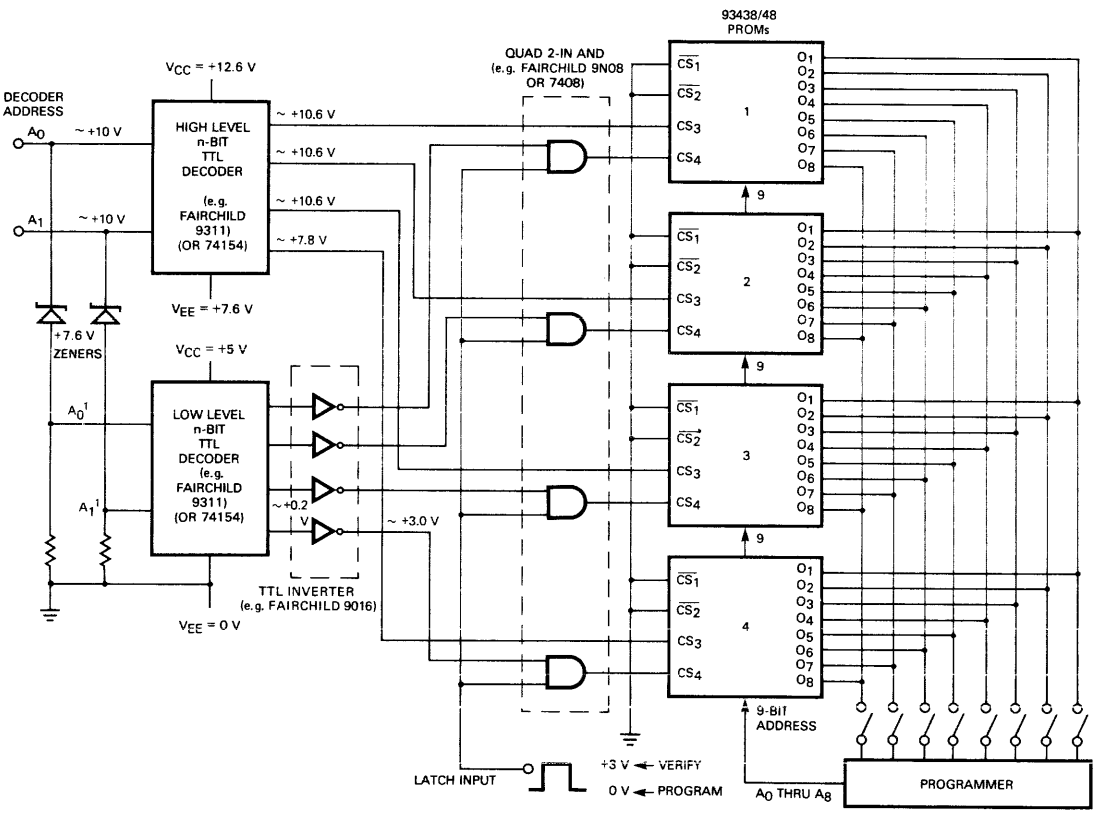


Fig. 2 Case B

DC CHARACTERISTICS: Over guaranteed operating ranges unless otherwise note.

SYMBOL	CHARACTERISTIC	LIMITS MIN	LIMITS TYP (Note 1)	LIMITS MAX	UNITS	CONDITIONS
I_{CEX}	Output Leakage Current (93438 only)			50	μA	V_{CC} = MAX, V_{CEX} = 4.0 V, 0°C to +75°C Address any HIGH Output
I_{CEX}	Output Leakage Current (93438 only)			100	μA	V_{CC} = MAX, V_{CEX} = 4.0 V, −55°C to +125°C Address any HIGH Output
V_{OL}	Output LOW Voltage		0.30	0.45	V	V_{CC} = MIN, I_{OL} = 16 mA A_0 = +10.8 V, $A_1 - A_8$ = HIGH
V_{OH}	Output HIGH Voltage (93448 only)	2.4			V	V_{CC} = MIN, I_{OH} = −2.0 mA
I_{off}	Output Leakage Current for HIGH Impedance State (93448 only)			50 −50	μA μA	V_{OH} = 2.4 V V_{OL} = 0.4 V 0°C to +75°C
I_{off}	Output Leakage Current for HIGH Impedance State (93448 only)			100 −50	μA μA	V_{OH} = 2.4 V V_{OL} = 0.4 V −55°C to +125°C
V_{IH}	Input HIGH Voltage	2.0			V	Guaranteed Input HIGH Voltage for All Inputs
V_{IL}	Input LOW Voltage			0.8	V	Guaranteed Input LOW Voltage for All Inputs
I_F	Input LOW Current I_{FA} (Address Inputs) I_{FCS} (Chip Select Inputs)		 −160 −160	 −250 −250	 μA μA	V_{CC} = MAX, V_F = 0.45 V
I_R	Input HIGH Current I_{RA} (Address Inputs) I_{RCS} (Chip Select Input)			 40 40	 μA μA	V_{CC} = MAX, V_R = 2.4 V
I_{CC}	Power Supply Current		130	175	mA	V_{CC} = MAX, Outputs Open Inputs Grounded and Chip Selected
C_O	Output Capacitance		7		pF	V_{CC} = 5.0 V, V_O = 4.0 V, f = 1.0 MHz
C_{IN}	Input Capacitance		4		pF	V_{CC} = 5.0 V, V_O = 4.0 V, f = 1.0 MHz
V_C	Input Clamp Diode Voltage			−1.2	V	V_{CC} = MIN, I_A = −18 mA

AC CHARACTERISTICS: T_A = 0°C to +75°C, V_{CC} = 5.0 V ±5%

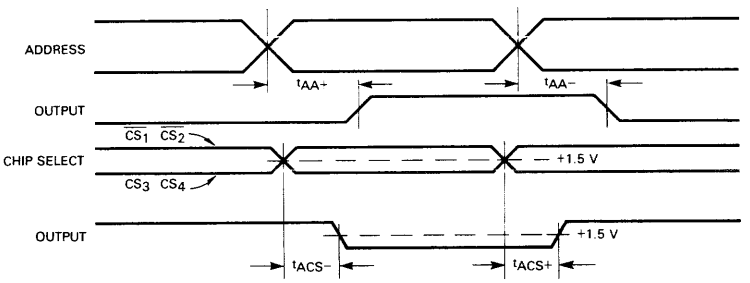
SYMBOL	CHARACTERISTIC	LIMITS MIN	LIMITS TYP (Note 1)	LIMITS MAX	UNITS	CONDITIONS
t_{AA-} t_{AA+}	Address to Output Access Time		35 35	55 55	ns ns	See Figure 3A and 3B
t_{ACS-} t_{ACS+}	Chip Select Access Time		15 15	25 25	ns ns	

AC CHARACTERISTICS: T_A = −55°C to +125°C, V_{CC} = 5.0 V ±10%

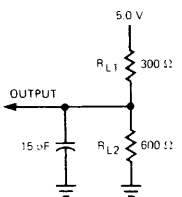
SYMBOL	CHARACTERISTIC	LIMITS MIN	LIMITS TYP (Note 1)	LIMITS MAX	UNITS	CONDITIONS
t_{AA-} t_{AA+}	Address to Output Access Time		35 35	70 70	ns ns	See Figure 3A and 3B
t_{ACS-} t_{ACS+}	Chip Select Access Time		15 15	30 30	ns ns	

Note (1): Typical limits are at V_{CC} = 5.0 V, +25°C and max loading.

SWITCHING WAVEFORMS

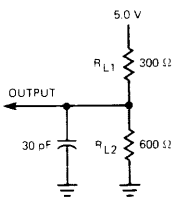


SWITCHING TEST OUTPUT LOAD



15 mA Load
Applies to 93438 Only

Fig. 3A



15 mA Load
Applies to 93448 Only

Fig. 3B

PROM PROGRAMMING CIRCUIT

This circuit will sequentially program all eight bits of a given word address of the 93438 or 93448. Selection of the bit patterns to be programmed is made by the address bit switches.

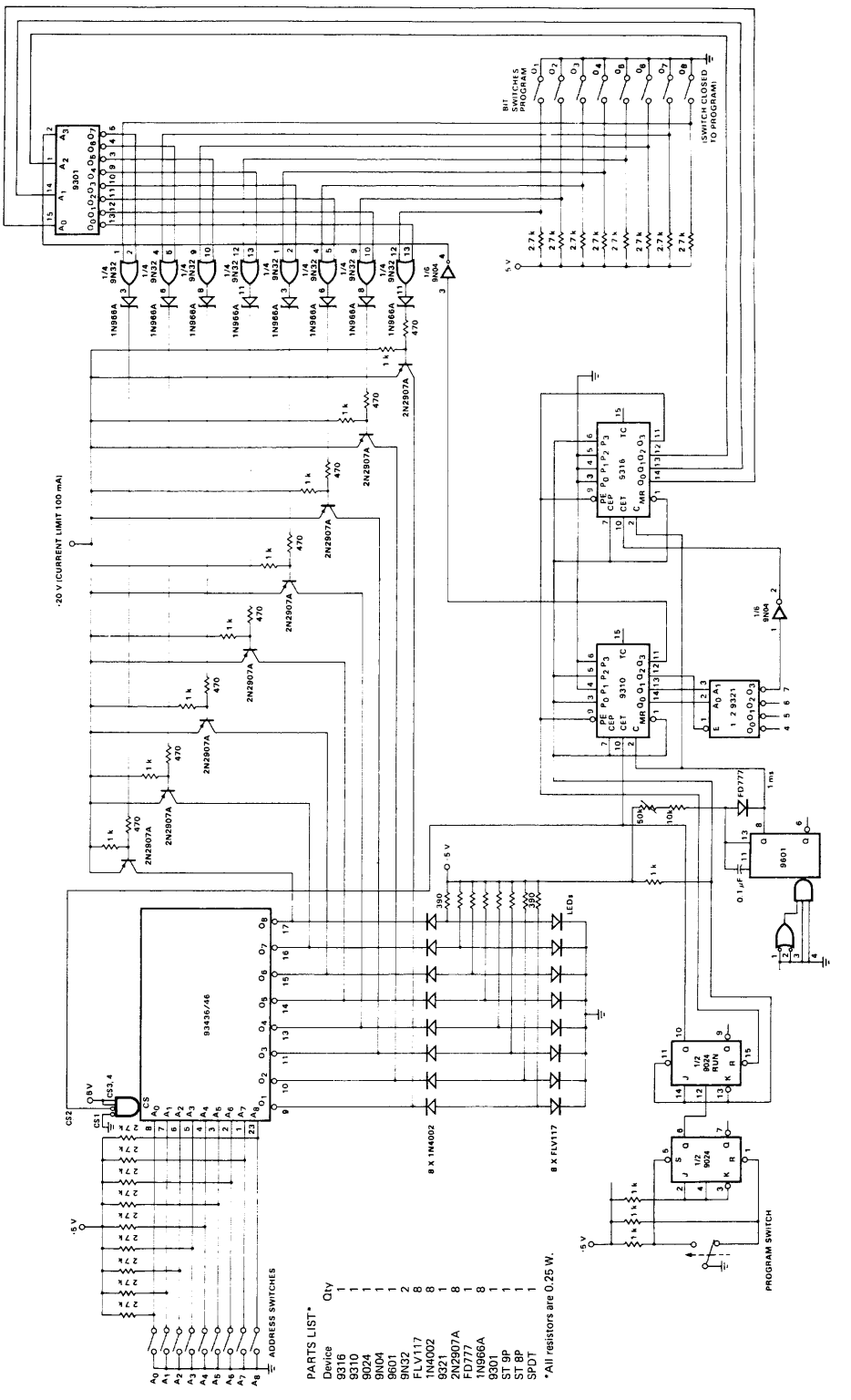
Until the program switch is depressed, the contents of the 93438 or 93448 at the address set in the address switch register is displayed on the eight FLV117 LEDs. If the content is a Logic "1" or the chip is deselected, the LED is turned on with current supplied by the 390 Ω resistors. If the content of the PROM is a Logic "0" and the PROM is enabled, the output is Logic "0" turning the LEDs off. The 1N4002s isolate the LEDs from the 20 V programming pulse.

The 9601 is a one-shot continuous 1.0 ms oscillator. One-half of a 9024 is used as a switch debouncer while the other half is the "run" flip-flop. When the program is initiated by depressing the program switch, the first half of the 9024 (switch debouncer) is set and clocks the other half of the 9024 ("run" flip-flop) to the "run" state. This enables the counters to operate and disables the 93438. The counter is preset to 5 on the 9310 and 8 on the 9316. The counter provides the proper duty cycle and program timing.

To avoid overlap problems between the programming pulse, the chip enable and the scan, the 9316 advances when the 9310 goes from state 3 to state 4. When the last bit has been programmed, the counter presets itself and resets the "run" flip-flop. The programming sequence is now complete.

The bit to be programmed is decoded by one-half of the 9301 and wired-OR with the bit switch. The OR gate is a high voltage driver supplying the drive to the programming transistors.

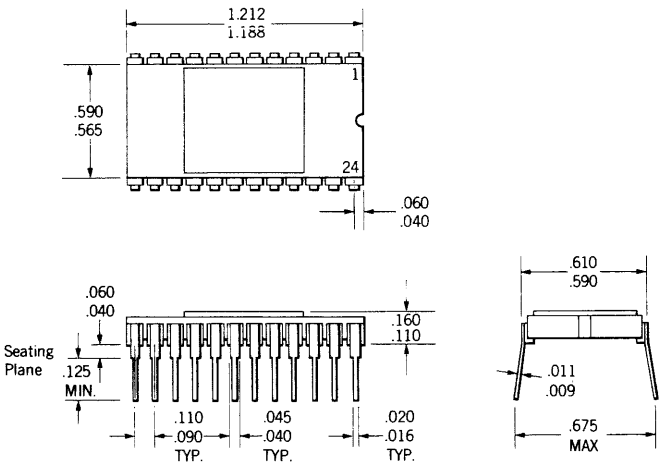
PROM PROGRAMMING CIRCUIT



ORDER INFORMATION – Specify 93438DC or 93438DM for open collector outputs, 93448DC or 93448DM for 3-state outputs, where "D" is the Ceramic Dual In-Line package, "C" is the commercial/industrial temperature range 0°C to +75°C and "M" is the military temperature range –55°C to +125°C.

PACKAGE INFORMATION

24-PIN CERAMIC DUAL IN-LINE PACKAGE
(METAL CAP)



NOTE
All dimensions in inches
Leads are gold-plated kovar
Package weight 4.0 grams