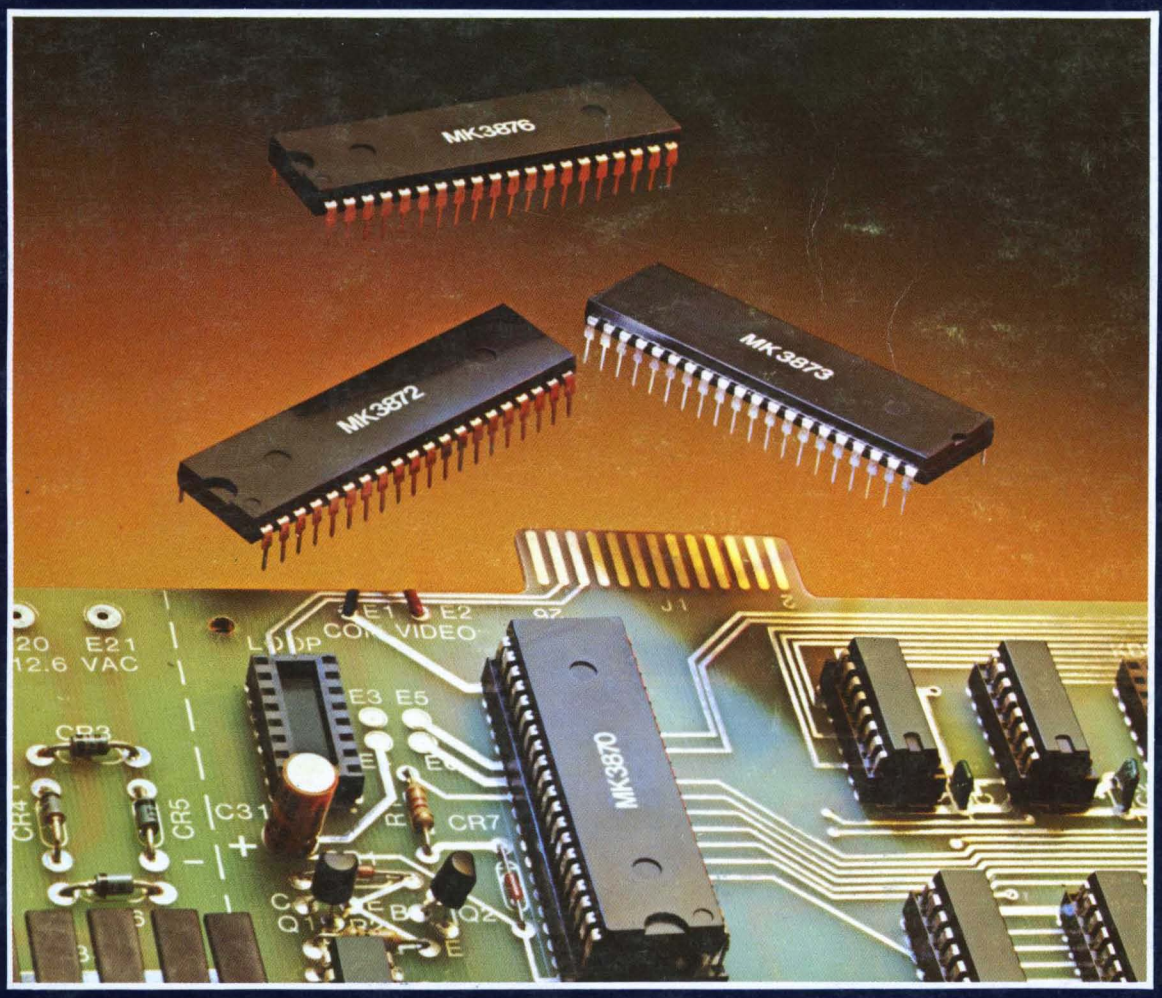
MOSTEK MICROCOMPUTER 3870/F8 DATA BOOK



MICROCOMPUTER 3870/F8 DATA BOOK INDEX

3870 DATA SHEETS

MOSTEK®

F8 MICROCOMPUTER DEVICES

Single-Chip Microcomputer MK3870

FEATURES

- ☐ Software compatible with 3870/F8 family
- ☐ 2048 X 8 mask programmable ROM
- ☐ 64 byte scratchpad RAM
- ☐ 32 bits (4 ports) TTL compatible I/O
- ☐ Programmable binary timer
  - Interval timer mode
  - Pulse width measurement mode
  - Event counter mode
- ☐ External interrupt
- ☐ Crystal, LC, RC, or external time base
- ☐ Low power (275 mW typ.)
- ☐ Single +5 volt ± 10% power supply
- ☐ Pinout compatible with 3870 family

GENERAL DESCRIPTION

The MK3870 is a complete 8-bit microcomputer on a single MOS integrated circuit. The 3870 can execute the F8 instruction set of more than 70 commands, allowing expansion into multi-chip configurations with software compatibility. The device features 2048 bytes of ROM, 64 bytes of scratchpad RAM, a programmable binary timer, 32 bits of I/O, and a single +5 volt power supply requirement.

Utilizing ion-implanted, N-channel silicon-gate technology and advanced circuit design techniques, the single-chip 3870 offers maximum cost effectiveness in a wide range of control and logic replacement applications.

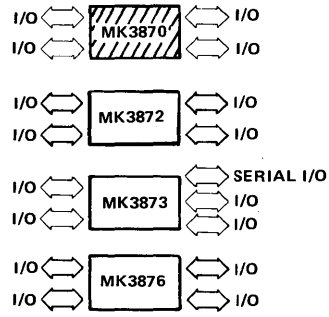
FUNCTIONAL PIN DESCRIPTION

$\overline{P0\text{-}0}$–$\overline{P0\text{-}7}$, $\overline{P1\text{-}0}$–$\overline{P1\text{-}7}$, $\overline{P4\text{-}0}$–$\overline{P4\text{-}7}$, and $\overline{P5\text{-}0}$–$\overline{P5\text{-}7}$ are 32 lines which can be individually used as either TTL compatible inputs or as latch outputs.

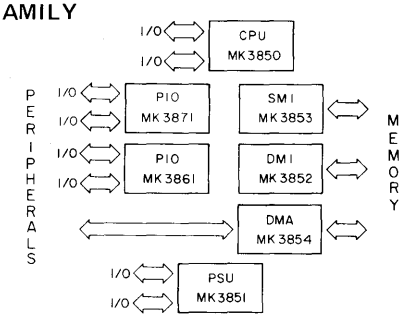
$\overline{STROBE}$ is a ready strobe associated with I/O Port 4. This pin which is normally high provides a single low pulse after valid data is present on the P4-0–P4-7 pins during an output instruction.

$\overline{RESET}$ may be used to externally reset the 3870. When pulled low the 3870 will reset. When then

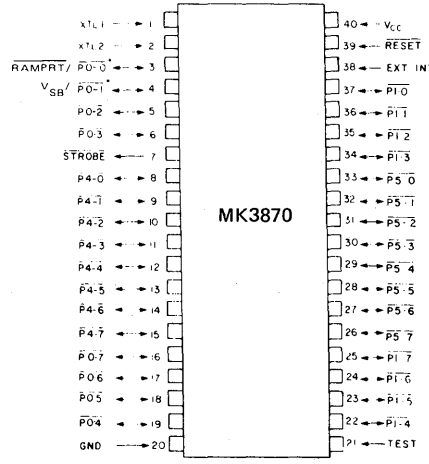
SINGLE CHIP 3870 MICROCOMPUTER FAMILY



F8 FAMILY



PIN CONNECTIONS



PIN NAME	DESCRIPTION	TYPE
$\overline{P0\text{-}0}$ – $\overline{P0\text{-}7}$	I/O Port 0	Bidirectional
$\overline{P1\text{-}0}$ – $\overline{P1\text{-}7}$	I/O Port 1	Bidirectional
$\overline{P4\text{-}0}$ – $\overline{P4\text{-}7}$	I/O Port 4	Bidirectional
$\overline{P5\text{-}0}$ – $\overline{P5\text{-}7}$	I/O Port 5	Bidirectional
$\overline{STROBE}$	Ready Strobe	Output
EXT INT	External Interrupt	Input
$\overline{RESET}$	External Reset	Input
TEST	Test Line	Input
XTL 1, XTL 2	Time Base	Input
V_{CC}, GND	Power Supply Lines	Input

allowed to go high the 3870 will begin program execution at program location H '000'.

EXT INT is the external interrupt input. Its active state is software programmable. This input is also used in conjunction with the timer for pulse width measurement and event counting.

XTL 1 and XTL 2 are the time base inputs to which a crystal (1 to 4 MHz), LC network, RC network, or an external single-phase clock may be connected.

TEST is an input, used only in testing the 3870. For normal circuit functionality this pin is left unconnected or may be grounded.

VCC is the power supply input (+5V ± 10%).

3870 ARCHITECTURE

This section describes the basic functional elements of the 3870 as shown in the block diagram of Figure 1. A programming model is shown in Figure 2.

Main Control Logic

The Instruction Register (IR) receives the operation code (OP code) of the instruction to be executed from the program ROM via the data bus. During all OP code fetches eight bits are latched into the IR. Some instructions are completely specified by the upper 4 bits of the OP code. In those instructions the lower 4 bits are an immediate register address or an immediate 4 bit operand. Once latched into the IR the main control logic decodes the instruction and provides the necessary control gating signals to all circuit elements.

ROM Address Registers

There are four 11 bit registers associated with the 2K x 8 ROM. These are the Program Counter (P0), the Stack Register (P), the Data Counter (DC) and the Auxiliary Data Counter (DC1). The Program Counter is used to address instructions or immediate operands. P is used to save the contents of P0 during an interrupt or subroutine call. Thus P contains the return address at which processing is to resume upon completion of the subroutine or the interrupt routine.

The Data Counter (DC) is used to address data tables. This register is auto-incrementing. Of the two data counters only DC can access the ROM. However, the XDC instruction allows DC and DC1 to be exchanged.

Associated with the address registers is an 11 bit Adder/Incrementer. This logic element is used to increment P0 or DC when required and is also used to add displacements to P0 on relative branches or to add the data bus contents to DC in the ADC (Add Data Counter) instruction.

2048 X 8 ROM

The microcomputer program and data constants are stored in the program ROM. When a ROM access is required, the appropriate address register (P0 or DC) is gated onto the ROM address bus and the ROM output is gated onto the main data bus. The first byte in the ROM is location zero.

Scratchpad and IS

The scratchpad provides 64 8-bit registers which may be used as general purpose RAM memory. The Indirect Scratchpad Address Register (IS) is a 6 bit register used to address the 64 registers. All 64 registers may be accessed using IS. In addition the lower order 12 registers may also be directly addressed.

IS can be visualized as holding two octal digits. This division of IS is important since a number of instructions increment or decrement only the least significant 3 bits of IS when referencing scratchpad bytes via IS. This makes it easy to reference a buffer consisting of contiguous scratchpad bytes. For example, when the low order octal digit is incremented or decremented IS is incremented from octal 27 (0 '27') to 0 '20) or is decremented from 0 '20' to 0 '27'. This feature of the IS is very useful in many program sequences. All six bits of IS may be loaded at one time or either half may be loaded independently.

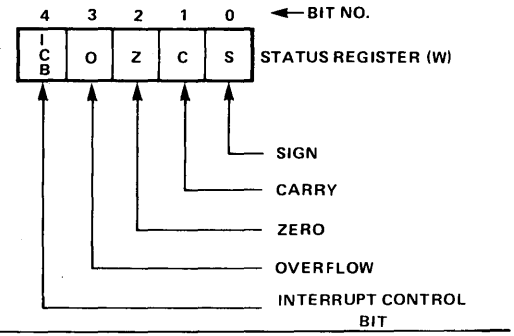
Scratchpad registers 9 through 15 (decimal) are given mnemonic names (J, H, K, and Q) because of special linkages between these registers and other registers such as the Stack Register. These special linkages facilitate the implementation of multi-level interrupts and subroutine nesting. For example, the instruction LR K,P stores the lower eight bits of the Stack Register into register 13 (K lower or KL) and stores the upper three bits of P into register 12 (K upper or KU).

Arithmetic and Logic Unit (ALU)

After receiving commands from the main control logic, the ALU performs the required arithmetic or logic operations (using the data presented on the two input busses) and provides the result on the result bus. The arithmetic operations that can be performed in the ALU are binary add, decimal adjust, add with carry, decrement, and increment. The logic operations that can be performed are AND, OR, EXCLUSIVE OR, 1's complement, shift right, and shift left. Besides providing the result on the result bus, the ALU also provides four signals representing the status of the result. These signals, stored in the Status Register (W), represent CARRY, OVERFLOW, SIGN, and ZERO condition of the result of the operation.

Accumulator(A)

The Accumulator (A) is the principal register for data manipulation within the 3870. The A serves as one input to the ALU for arithmetic or logical operations. The result of ALU operations are stored in the A.



Summary of Status Bits

OVERFLOW = CARRY $_7 \oplus$ CARRY $_6$

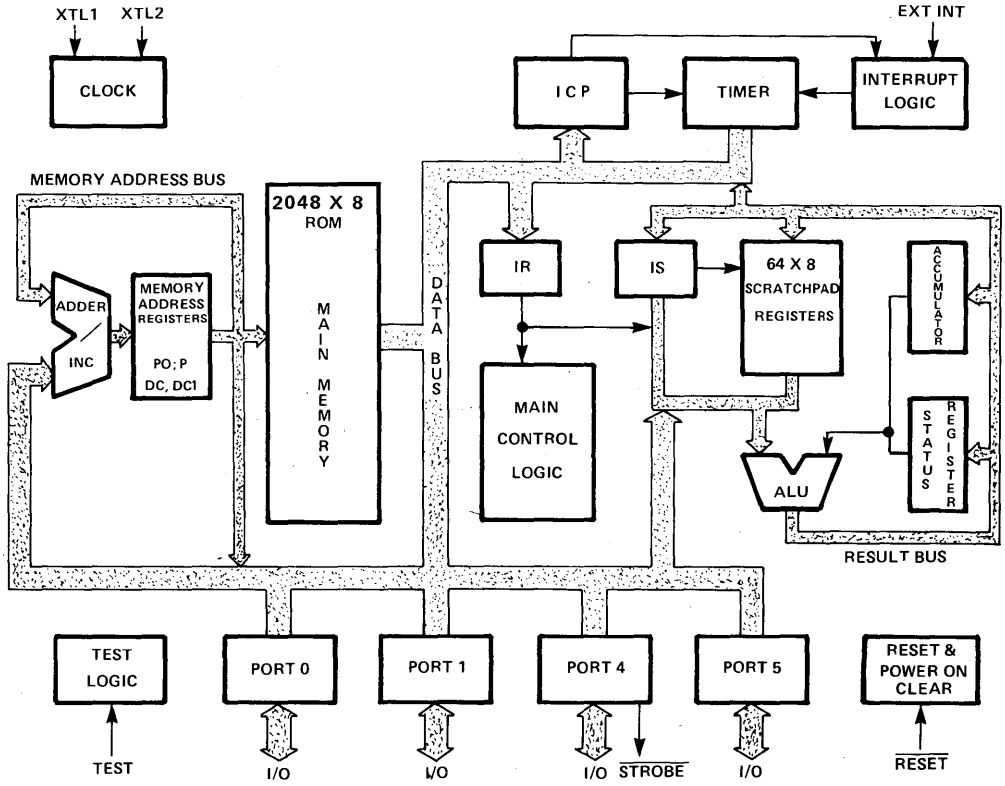
ZERO = $\overline{ALU_7} \wedge \overline{ALU_6} \wedge \overline{ALU_5} \wedge \overline{ALU_4} \wedge \overline{ALU_3} \wedge \overline{ALU_2} \wedge \overline{ALU_1} \wedge \overline{ALU_0}$

CARRY = CARRY$_7$

SIGN = $\overline{ALU_7}$

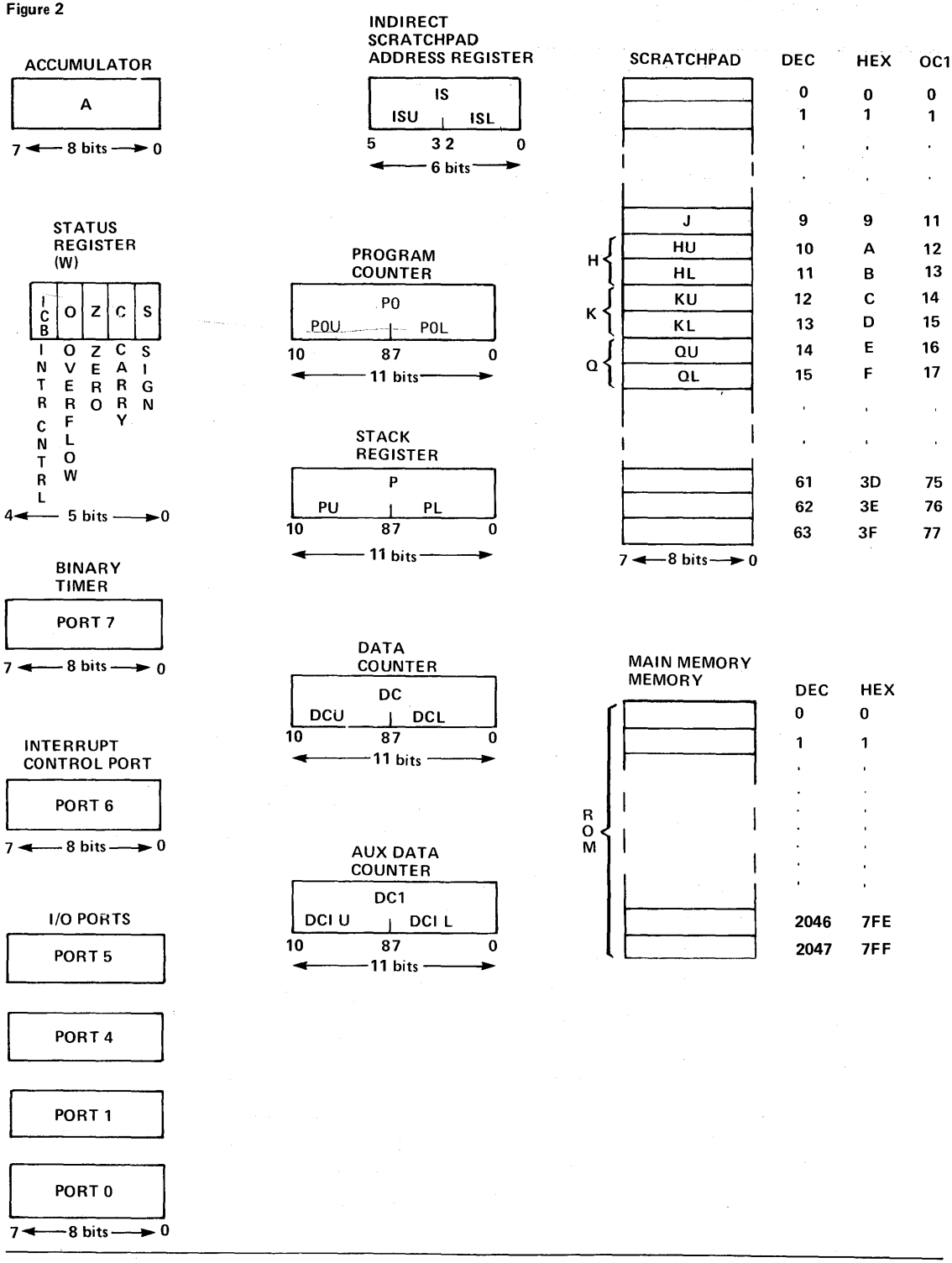
MK3870 BLOCK DIAGRAM

Figure 1



3870 PROGRAMMABLE REGISTERS, PORTS AND MEMORY MAP

Figure 2



The Status Register(W)

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

Interrupt Control Bit (ICB)

The ICB may be used to allow or disallow interrupts in the 3870. This bit is not the same as the two interrupt enable bits in the Interrupt Control Port (ICP). If the ICB is set and the 3870 interrupt logic communicates an interrupt request to the CPU section, the interrupt will be acknowledged and processed upon completion of the first non-privileged instruction. If the ICB is cleared an interrupt request will not be acknowledged or processed until the ICB is set.

I/O Ports

The 3870 provides four complete bidirectional Input/Output ports. These are ports 0, 1, 4, and 5. In addition, the Interrupt Control Port is addressed as port 6 and the binary timer is addressed as port 7. An output instruction (OUT or OUTS) causes the contents of A to be latched into the addressed port. An input instruction (IN or INS) transfers the contents of the port to A (port 6 is an exception which is described later). The schematic of an I/O pin and available output drive options are shown in Figure 3.

An output ready strobe is associated with port 4. This flag may be used to signal a peripheral device that the 3870 has just completed an output of new data to port 4. The strobe provides a single low pulse shortly after the output operation is completely finished, so either edge may be used to signal the peripheral. $\overline{\text{STROBE}}$ may also be used as an input strobe simply by doing a dummy output of H '00' strobe to port 4 after completing the input operation.

Timer and Interrupt Control Port

The Timer is an 8-bit binary down counter which is software programmable to operate in one of three modes: the Interval Timer Mode, the Pulse Width Measurement Mode, or the Event Counter Mode. As shown in Figure 4, associated with the Timer are an 8-bit register called the Interrupt Control Port, a programmable prescaler, and an 8-bit modulo-N register. A functional logic diagram is shown in Figure 5.

The desired timer mode, prescale value, starting and stopping the timer, active level of the EXT INT pin, and local enabling or disabling of interrupts are selected by outputting the proper bit configuration from the Accumulator to the Interrupt Control Port (port 6) with an OUT or OUTS instruction. Bits within the Interrupt Control Port are defined as follows:

Interrupt Control Port (Port 6)

Bit 0 - External Interrupt Enable
Bit 1 - Timer Interrupt Enable
Bit 2 - EXT INT Active Level
Bit 3 - Start/Stop Timer
Bit 4 - Pulse Width/Interval Timer
Bit 5 - ÷ 2 Prescale
Bit 6 - ÷ 5 Prescale
Bit 7 - ÷ 20 Prescale

A special situation exists when reading the Interrupt Control Port (with an IN or INS instruction). The Accumulator is <u>not</u> loaded with the content of the ICP; instead, Accumulator bits 0 through 6 are loaded with 0's while bit 7 is loaded with the logic level being applied to the EXT INT pin, thus allowing the status of EXT INT to be determined without the necessity of servicing an external interrupt request. When reading the Interrupt Control Port (Port 6) bit 7 of the Accumulator is loaded with the actual logic level being applied to the EXT INT pin, regardless of the status of ICP bit 2 (the EXT INT Active Level bit); that is, if EXT INT is at +5V bit 7 of the Accumulator is set to a logic 1, but if EXT INT is at GND then Accumulator bit 7 is reset to logic 0. This capability is useful in establishing a high speed polled handshake procedure or for using EXT INT as an extra input pin if external interrupts are not required and the Timer is used only in the Interval Timer Mode. However, if it is desirable to read the contents of the ICP then one of the 64 scratchpad registers or one byte of RAM may be used to save a copy of whatever is written to the ICP.

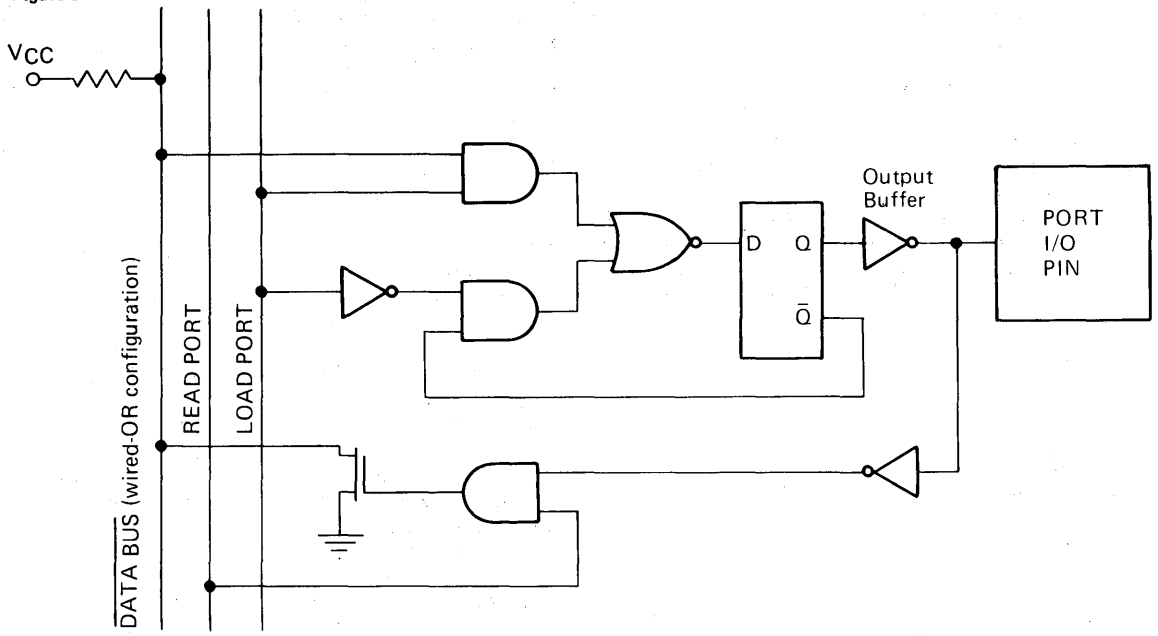
The rate at which the timer is clocked in the Interval Timer Mode is determined by the frequency of an internal Φ clock and by the division value selected for the prescaler. (The internal Φ clock operates at one-half the external time base frequency). If ICP bit 5 is set and bits 6 and 7 are cleared, the prescaler divides Φ by 2. Likewise, if bit 6 or 7 is individually set the prescaler divides Φ by 5 or 20 respectively. Combinations of bits 5, 6 and 7 may also be selected. For example, if bits 5 and 7 are set while 6 is cleared the prescaler will divide by 40. Thus possible prescaler values are ÷2, ÷5, ÷10, ÷20, ÷40, ÷100, and ÷200.

Any of three conditions will cause the prescaler to be reset: whenever the timer is stopped by clearing ICP bit 3, execution of an output instruction to Port 7, (the timer is assigned port address 7), or on the trailing edge transition of the EXT INT pin when in the Pulse Width Measurement Mode. These last two conditions are explained in more detail below.

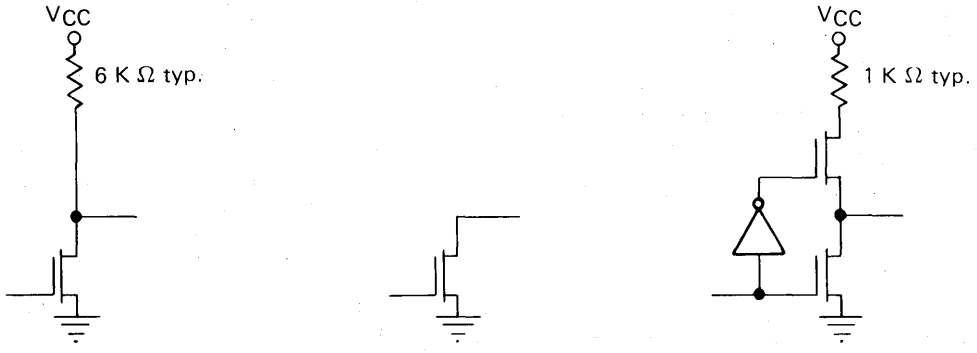
An OUT or OUTS instruction to Port 7 will load the content of the Accumulator to both the Timer and the 8-bit modulo-N register, reset the prescaler, and

I/O PIN CONCEPTUAL DIAGRAM WITH OUTPUT BUFFER OPTIONS

Figure 3



OUTPUT BUFFER OPTIONS



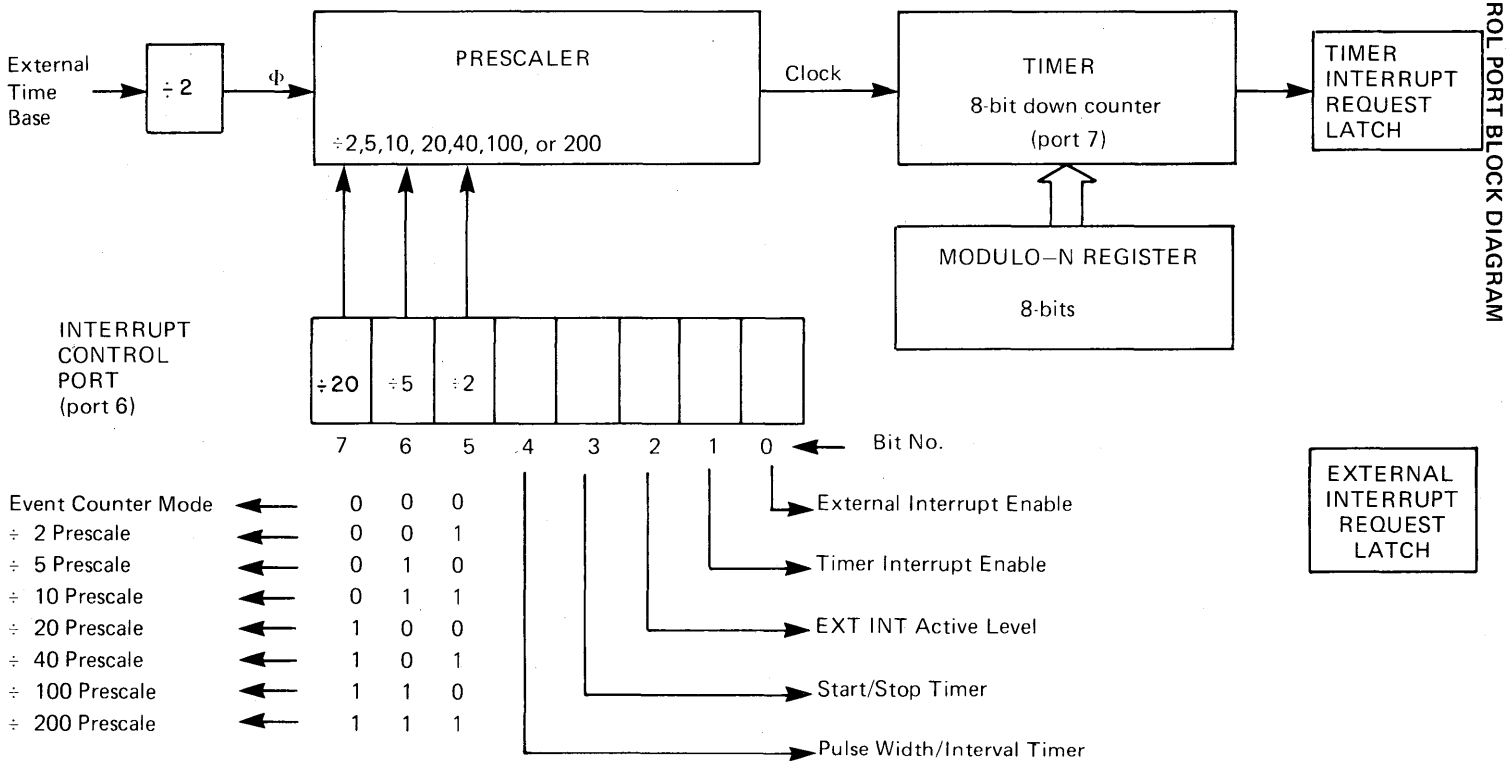
Ports 0 and 1 are Standard Output type only.

Ports 4 and 5 may both be any of the three output options (programmable bit by bit).

The $\overline{\text{STROBE}}$ output is always configured similar to a Direct Drive Output except that it is capable of driving 3 TTL loads.

$\overline{\text{RESET}}$ and EXT INT may have standard 6KΩ (typical) pull-up or may have no pull-up. These two inputs have Schmidt trigger inputs with a minimum of 0.2 volts of hysteresis.

TIMER & CONTROL PORT BLOCK DIAGRAM
Figure 4



Note: See Figure 5 for a more detailed functional diagram.

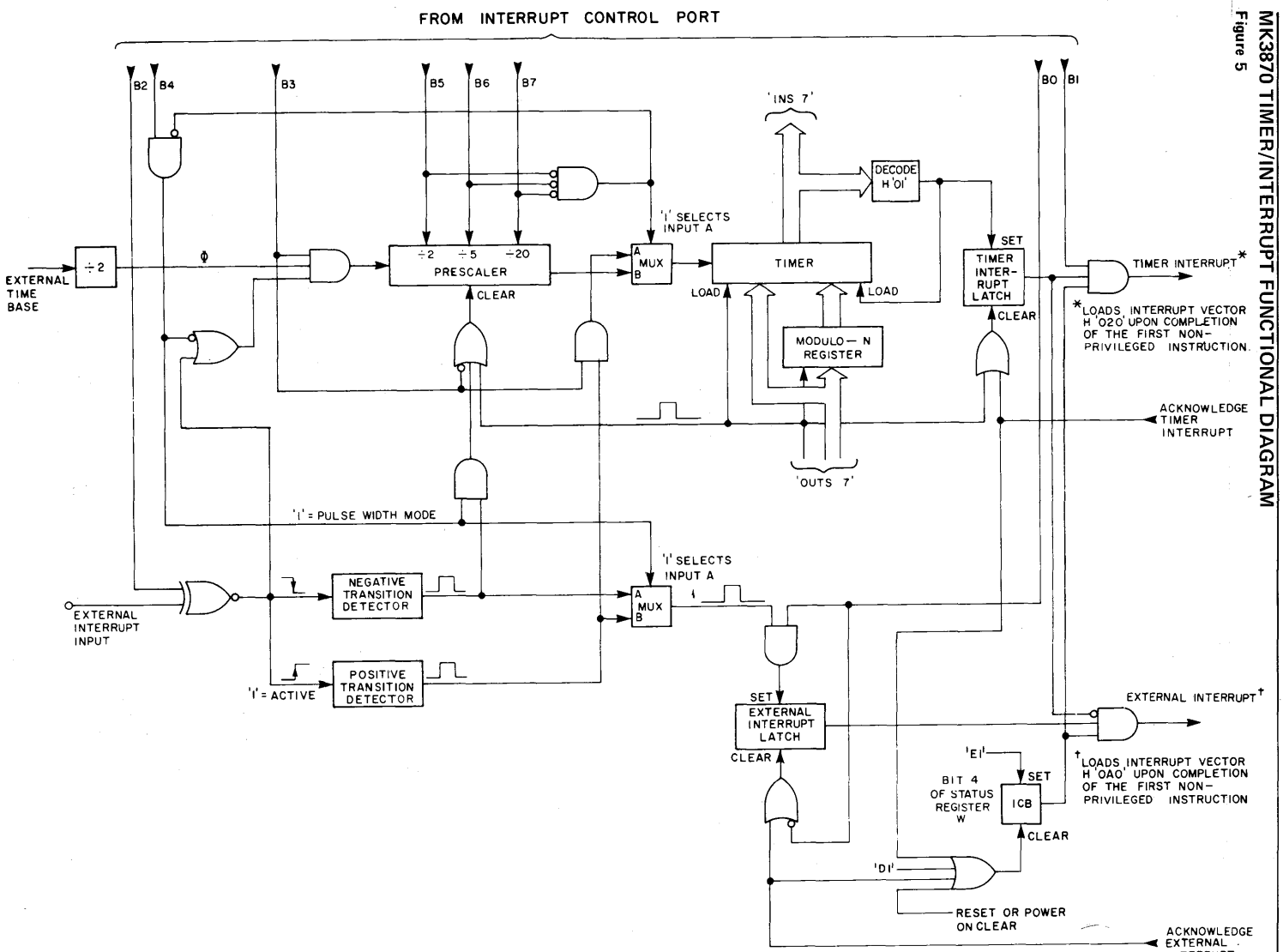


Figure 5 MK3870 TIMER/INTERRUPT FUNCTIONAL DIAGRAM

clear any previously stored timer interrupt request. As previously noted, the Timer is an 8-bit down counter which is clocked by the prescaler in the Interval Timer Mode and in the Pulse Width Measurement Mode. The prescaler is not used in the Event Counter Mode. The Modulo-N register is a buffer whose function is to save the value which was most recently outputted to Port 7. The modulo-N register is used in all three timer modes.

Interval Timer Mode

When ICP bit 4 is cleared (logic 0) and at least one prescale bit is set the Timer operates in the Interval Timer Mode. When bit 3 of the ICP is set the Timer will start counting down from the modulo-N value. After counting down to H'01', the Timer returns to the modulo-N value at the next count. On the transition from H'01' to H 'N' the Timer sets a timer interrupt request latch. Note that the interrupt request latch is set by the transition to H 'N' and not be the presence of H 'N' in the Timer, thus allowing a full 256 counts if the modulo-N register is preset to H '00'. If bit 1 of the ICP is set, the interrupt request is passed on to the CPU section of the 3870. However, if bit 1 of the ICP is a logic 0 the interrupt request is not passed on to the CPU section but the interrupt request latch remains set. If ICP bit 1 is subsequently set, the interrupt request will then be passed on to the CPU section. (Recall from the discussion of the Status Register's Interrupt Control Bit that the interrupt request will be acknowledged by the CPU section only if ICB is set). Only two events can reset the timer interrupt request latch; when the timer interrupt request latch is acknowledged by the CPU section, or when a new load of the modulo-N register is performed.

Consider an example in which the modulo-N register is loaded with H '64' (decimal 100). The timer interrupt request latch will be set at the 100th count following the timer start and the timer interrupt request latch will repeatedly be set on precise 100 count intervals. If the prescaler is set at ÷40 the timer interrupt request latch will be set every 4000 Φ clock periods. For a 2MHz Φ clock (4MHz time base frequency) this will produce 2 millisecond intervals.

The range of possible intervals is from 2 to 51,200 Φ clock periods (1μs to 25.6ms for a 2MHzΦ clock). However, approximately 50 Φ periods is a practical minimum because the time between setting the interrupt request latch and the execution of the first instruction of the interrupt service routine is at least 29 Φ periods (the response time is dependent upon how many privileged instructions are encountered when the request occurs); 29 is based on the timer interrupt occuring at the beginning of a non-privileged short instruction. To establish time intervals greater than 51,200 Φ clock periods is a simple matter of using the timer interrupt service routine to count the number of interrupts, saving the result in one or more of the scratchpad registers until the desired interval is achieved. With this technique virtually any time interval, or several time intervals, may be generated.

The Timer may be read at any time and in any mode using an input instruction (IN 7 or INS 7) and may take place "on the fly" without interfering with normal timer operation. Also, the Timer may be stopped at any time by clearing bit 3 of the ICP. The Timer will hold its current contents indefinitely and will resume counting when bit 3 is again set. Recall however that the prescaler is reset whenever the Timer is stopped; thus a series of starting and stopping will result in a cumulative truncation error.

A summary of other timer errors is given in the timing section of this specification. For a free running timer in the Interval Timer Mode the time interval between any two interrupt requests may be in error by ± 6 Φ clock periods although the cumulative error over many intervals is zero. The prescaler and Timer generate precise intervals for setting the timer interrupt request latch but the time out may occur at any time withi n a machine cycle. (There are two types of machine cycles; short cycles which consist of 4 Φ clock periods and long cycles which consist of 6 Φ clock periods. In the multi-chip F8 family there is a signal called the WRITE clock which corresponds to a machine cycle). Interrupt requests are synchronized with the internal WRITE clock thus giving rise to the possible ± 6 Φ error. Additional errors may arise due to the interrupt request occuring while a privileged instruction or multicycle instruction is being executed. Nevertheless, for most applications all of the above errors are negligible, especially if the desired time interval is greater than 1 ms.

Pulse Width Measurement Mode

When ICP bit 4 is set (logic 1) and at least one prescale bit is set the Timer operates in the Pulse Width Measurement Mode. This mode is used for accurately measuring the duration of a pulse applied to the EXT INT pin. The Timer is stopped and the prescaler is reset whenever EXT INT is at its inactive level. The active level of EXT INT is defined by ICP bit 2; if cleared, EXT INT is active low; if set, EXT INT is active high. If ICP bit 3 is set, the prescaler and Timer will start counting when EXT INT transitions to the active level. When EXT INT returns to the inactive level the Timer then stops, the prescaler resets, and if ICP bit 0 is set an external interrupt request latch is set. (Unlike timer interrupts, external interrupts are not latched if the ICP Interrupt Enable bit is not set).

As in the Interval Timer Mode, the Timer may be read at any time, may be stopped at any time by clearing ICP bit 3, the prescaler and ICP bit 1 function as previously described, and the Timer still functions as an 8-bit binary down counter with the timer interrupt request latch being set on the Timer's transition from H '01' to H 'N'. Note that the EXT INT pin has nothing to do with loading the Timer; its action is that of automatically starting and stopping the Timer and of generating external interrupts. Pulse widths longer than the prescale value times the modulo-N value are easily measured by using the timer interrupt service routine to store the number of timer interrupts in one or more scratchpad registers.

As for accuracy, the actual pulse duration is typically slightly longer than the measured value because the status of the prescaler is not readable and is reset when the Timer is stopped. Thus for maximum accuracy it is advisable to use a small division setting for the prescaler.

Event Counter Mode

When ICP bit 4 is cleared and all prescale bits (ICP bits 5, 6, and 7) are cleared the Timer operates in the Event Counter Mode. This mode is used for counting pulses applied to the EXT INT pin. If ICP bit 3 is set the Timer will decrement on each transition from the inactive level to the active level of the EXT INT pin. The prescaler is not used in this mode; but as in the other two timer modes, the Timer may be read at any time, may be stopped at any time by clearing ICP bit 3, ICP bit 1 functions previously described, and the timer interrupt request latch is set on the Timer's transition from H '01' to H 'N'.

Normally ICP bit 0 should be kept cleared in the Event Counter Mode; otherwise, external interrupts will be generated on the transition from the inactive level to the active level of the EXT INT pin.

For the Event Counter Mode the minimum pulse width required on EXT INT is 2 Φ clock periods and the minimum inactive time is 2 Φ clock periods; therefore, the maximum repetition rate is 500 KHz.

Timer Emulation

For total software compatibility when expanding into a multi-chip configuration the MK3871 Peripheral Input/Output circuit should be used rather than the older MK3861 PIO. The MK3871 has the same improved Timer (binary count, readable, and three modes of operation rather than one) and ready strobe output as are on the MK3870.

External Interrupts

When the timer is in the Interval Timer Mode the EXT INT pin is available for non-timer related interrupts. If ICP bit 0 is set an external interrupt request latch is set when there is a transition from the inactive level to the active level of EXT INT. (EXT INT is an edge-triggered input). The interrupt request is latched until either acknowledged by the CPU section or until ICP bit 0 is cleared (unlike timer interrupt requests which remain latched even when ICP bit 1 is cleared). External interrupts are handled in the same fashion when the Timer is in the Pulse Width Measurement Mode or in the Event Counter Mode, except that only in the Pulse Width Measurement Mode the external interrupt request latch is set on the trailing edge of EXT INT; that is, on the transition from the active level to the inactive level.

Interrupt Handling

When either a timer or an external interrupt request is communicated to the CPU section of the 3870, it will be acknowledged and processed at the completion of the first non-privileged instruction if the Interrupt Control Bit of the Status Register is set. If the Interrupt Control Bit is not set, the interrupt request will continue until either the Interrupt Control Bit is set and the CPU section acknowledges the interrupt or until the interrupt request is cleared as previously described.

If there is both a timer interrupt request and an external interrupt request when the CPU section starts to process the requests, the timer interrupt is handled first.

When an interrupt is allowed the CPU section will request that the interrupting element pass its interrupt vector address to the Program Counter via the data bus. The vector address for a timer interrupt is H '020'. The vector address for external interrupts is H '0A0'. After the vector address is passed to the Program Counter, the CPU section sends an acknowledge signal to the appropriate interrupt request latch which clears that latch. The exection of the interrupt service routine will then commence. The return address of the original program is automatically saved in the Stack Register, P.

The Interrupt Control Bit of W (Status Register) is automatically reset when an interrupt request is acknowledged. It is then the programmer's responsibility to determine when ICB will again be set (by executing an EI instruction). This action prevents an interrupt service routine from being interrupted unless the programmer so desires.

Figure 6 details the interrupt sequence which occurs whether the interrupt request is from an external source via EXT INT or from the 3870's internal timer. Events are labeled with the letters A through G and are described below.

Event A

An interrupt request must satisfy a hold time requirement as specified in the AC Characteristics in order to guarantee that it is valid on the rising edge of the WRITE clock.

Event B

Event B represents the instruction being executed when the interrupt occurs. The last cycle of B is normally the instruction fetch for the next cycle. However, if B is not a privileged instruction and the CPU's Interrupt Control Bit is set, then the last cycle becomes a "freeze" cycle rather than a fetch. At the end of the freeze cycle the interrupt request latches are inhibited from altering the interrupt daisy-chain so that sufficient time will be allowed for the daisy-chain to settle. (If B is a privileged instruciton, the instruction fetch is not replaced by a freeze cycle; instead, the fetch is performed and the next instruction is executed. Although unlikely to be encountered, a series of privileged instructions will be sequentially executed without interrupt. One more instruction, called a 'protected' instruction, will always be executed after the last privileged instruction. The last cycle of the protected instruction then performs the freeze.)

The dashed lines on EXT INT illustrate the last opportunity for EXT INT to cause the last cycle of a non-protected instruction to become a freeze cycle.

The freeze cycle is a short cycle (4 Φ clock periods) in all cases except where B is the Decrement Scratchpad instruction, in which case the freeze cycle is a long cycle (6 Φ clock periods).

INT REQ goes low on the next negative edge of WRITE if both PRI IN is low and the appropriate interrupt enable bit of the Interrupt Control Part is set. Both INT REQ and WRITE are internal signals.

Event C

A NO-OP long cycle to allow time for the internal priority chain to settle.

Event D

The program counter (P0) is pushed to the stack register (P) in order to save the return address. The interrupt circuitry places the lower 8 bits of the interrupt vector address onto the data bus. This is always a long cycle.

Event E

A long cycle in which the interrupt circuitry places the upper 8 bits of the interrupt vector address onto the data bus.

Event E

A long cycle in which the interrupt circuitry places the upper 8 bits of the interrupt vector address onto the data bus.

Event F

A short cycle in which the interrupting interrupt request latch is cleared. Also, the CPU's Interrupt Control Bit is cleared, thus disabling interrupts until an EI instruction is performed. The fetch of the next instruction from the interrupt address.

Event G

Begin execution of the first instruction of the interrupt service routine.

Summary Of Interrupt Sequence

For the MK3870 the interrupt response time is defined as the time elapsed between the occurence of EXT INT going active (or the Timer transitioning to H'N') and the beginning of execution of the first instruction of the interrupt service routine. The interrupt response time is a variable depedent upon what the microprocessor is doing when the interrupt request occurs. As shown in Figure 5, the minimum interrupt response time is 3 long cycles plus 2 short cycles plus one WRITE clock pulse width plus a setup time of EXT INT prior to the leading edge of the WRITE pulse – a total of 27 Φ clock periods plus the setup time. At a 2 MHz Φ this is 14.25 μs. Although the maximum could theoretically be infinite, a practical maximum is 35 μs (based on the interrupt request occurring near the beginning of a PI and LR K, P sequence).

Power-On Clear

The intent of the Power-On-Reset circuitry on the 3870 is to automatically reset the device following a typical power-up situation, thus saving external reset circuitry in many applications. This circuitry is not guaranteed to sense a "Brown Out" (low voltage) condition nor is it guaranteed to operate under all possible power-on situations.

Three conditions are required before the 3870 will leave the reset state and begin operation. Refer to Figure 7 as an aid to the following descriptions. The On-Chip Vcc detector senses a minimum value of Vcc before it will allow the 3870 to operate. The threshold of this detector is set by analog circuitry because a stable voltage reference is not available with n-channel MOS processing. Processing variations will cause this threshold to vary from a low of 3.0 volts to a high of 4.3 volts with 3.5 volts being typical.

The 3870 uses a substrate bias as a technique to provide improved performance verses power consumption relative to conventional grounded substrate approaches. This bias generator may start operating as low as Vcc = 3 volts on some devices while others may require Vcc = 4 volts in order to get adequate substrate bias. Until the substrate reaches the proper bias, the 3870 will not be released from the reset state. The final condition required is that the clocks of the 3870 must be functioning. Typically the clocks will start to function at Vcc equal to 3 to 3.5 volts but since the part is tested at 4.5 volts MOSTEK cannot guarantee any operation below 4.5 volts. The output of the delay circuit in Figure 7 will stay low until the clocks start to function. If the input to the delay circuit is high, typically after 100 cycles of the WRITE clock (800 cycles of the external clock) the output of the delay circuit will go high allowing the 3870 to begin execution.

If Vcc falls to ground for at least a few hundred nanoseconds the output of the delay circuit will go low immediately and the 3870 will reset.

The internal logic may detect a valid Vcc, bias and clocks at Vcc = 3.5 volts and allow the 3870 to start executing after the time delay. With a slowly rising power supply the part may start running before Vcc is above 4.5 volts which is below the guaranteed voltage range. When power-on-clear is required with a slowly rising power supply, an external capacitor must be used on the $\overline{\text{RESET}}$ pin to hold it below 0.8 volts until Vcc is stable above 4.5 volts. (Note: The option to disconnect the internal pull-up resistor on $\overline{\text{RESET}}$ is available which allows the use of a larger external pull-up resistor and a small capacitor on $\overline{\text{RESET}}$.)

In many applications, it is desirable if the unit does an automatic power-on-clear, but not mandatory. The unit will have a RESET push button and if the unit does not power-up correctly or malfuctions because of some disturbance on the Vcc line, the operator will simply press RESET and restore normal operation. It is for these applications that the internal power-on-clear circuitry was designed.

In some applications it is required that the microcomputer continue to run properly without operator intervention after brown-outs, power line disturbances, electrical noise, computer malfunction due to a programming bug or any other disturbance except a catastrophic failure of some component.

Once concept used to keep computers running is that of the "WATCHDOG TIMER". The computer is programmed to periodically reset the watchdog timer during the normal execution of its program (this is easily done in the 3870 as its normal application is in some control function which is typically periodic). As long as the computer continues to execute its program the watchdog timer is continually reset and never times out. Should the computer stop executing its program for whatever reason, the watchdog timer will time out producing a RESET pulse to the CPU re-starting execution. This is a very positive way to assure that the computer is doing its job, i.e., executing the program. It is important that the software driving the watchdog timer test as many functional blocks (timer, ALU, scratchpad RAM, and Ports) of the 3870 as possible before reseting the watchdog timer. This is because operation of the 3870 with an out of spec power supply may allow some of the functions to operate correctly while other functions are not operable.

MOSTEK can guarantee correct operation of the 3870 only while the Vcc voltage remains within its specified limits. If proper operation of the 3870 must be guaranteed after a disturbance on the Vcc line, then an external circuit must be used to monitor the Vcc line and produce a $\overline{\text{RESET}}$ to the 3870 whenever Vcc is out of the specified limits.

A related characteristic to power-on-clear is the Startup time of the basic timing element. The LC, and RC, oscillators begin to function almost immediately once Vcc is high enough to allow the on-board oscillator to operate (Vcc = 3.5). Operation with a crystal is partly mechanical and some start time is required to get the mass of the crystal into vibrational motion. This time is basically dependent on the frequency (mass) of the crystal. 4 MHz crystals typically require about 2-3 mSec to start while 1 MHz crystals require 60-70 mSec to start oscillating. Of course, this time may vary greatly from crystal to crystal and is also a function of the power supply rise time characteristic, however, the high frequency crystals start faster and are definately recommended (i.e., 3-4 MHz).

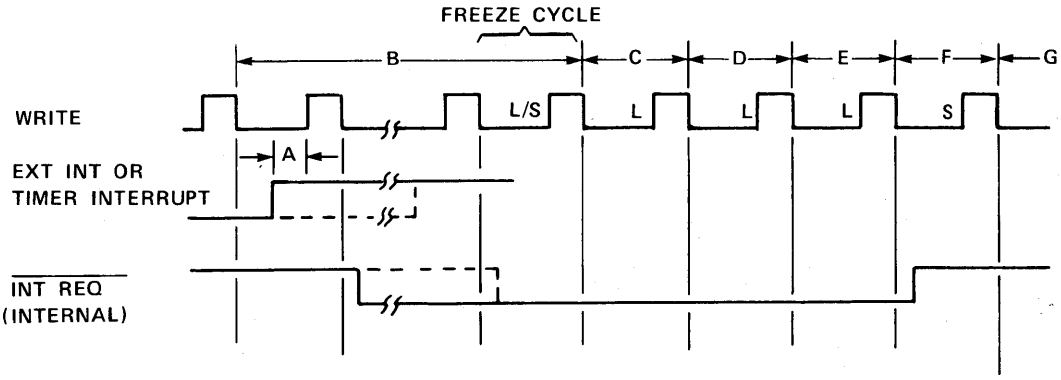
The condition of the port pins during the power-on-clear sequence is often asked. The port pins or the STROBE line cannot be specified until Vcc reaches 4.5V and the 3870 enters the RESET state. Before this, the port pins may stay at Vss, may track Vcc as it rises, or they may track Vcc part way up then return to Vss (Ports 4 & 5 will go to Vcc once the clocks are running and the 3870 has sufficient Vcc to properly operate the internal control logic and I/O ports).

External Reset

When $\overline{\text{RESET}}$ is taken low the content of the Program Counter is pushed to the Stack Register and then the Program Counter and the ICB bit of the W Status Register are cleared. The original Stack Register content is lost. Ports 4, 5, 6 and 7 are loaded

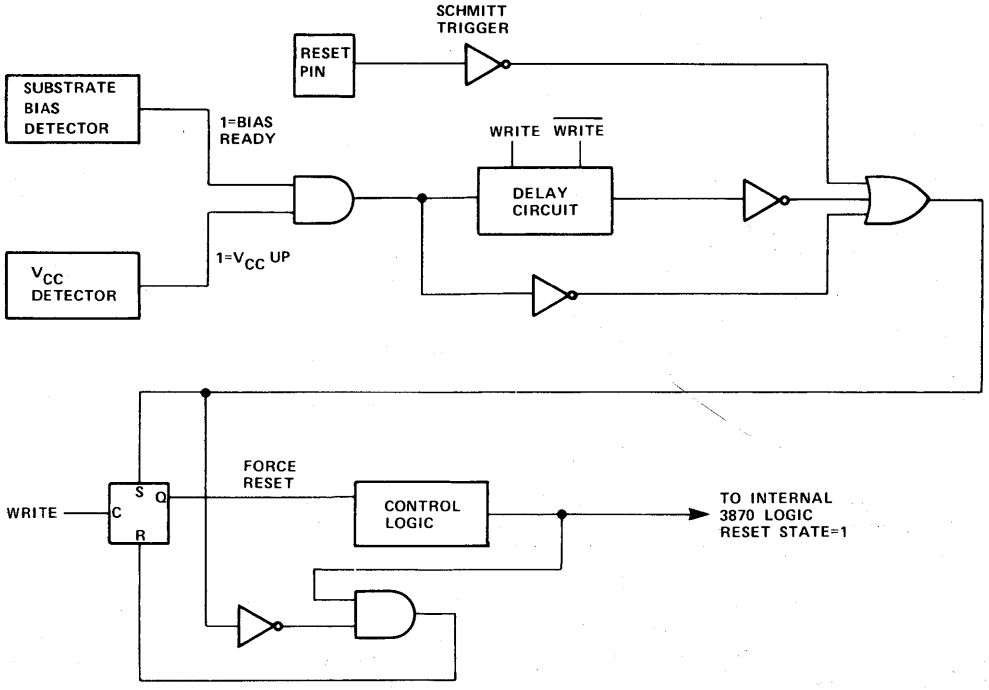
INTERRUPT SEQUENCE

Figure 6



POWER ON CLEAR BLOCK DIAGRAM

Figure 7



with H '00'. The contents of all other registers and ports are unchanged or undefined. When RESET is taken high the first program instruction is fetched from ROM location H'000'. When an external reset of the 3870 occurs, P0 is pushed into P and the old contents of P are lost. It must be noted that an external reset is recognized at the start of a machine cycle and not necessarily at the end of an instruction. Thus if the 3870 is executing a multi-cycle instruction, that instruction is not completed and the contents of P upon reset may not necessarily be the address of the instruction that would have been executed next. It may, for example, point to an immediate operand if the reset occurred during the second cycle of a LI or CI instruction. Additionally, several instructions (JMP, PI, PK, LR P0, Q) as well as the interrupt acknowledge sequence modify P0 in parts. That is, they alter P0 by first loading one part then the other and the entire operation takes more than one cycle. Should reset occur during this modification process the value pushed into P will be part of the old P0 (the as yet unmodified part) and part of the new P0 (already modified part). Thus care should be taken (perhaps by external gating) to insure that reset does not occur at an undesirable time if any significance is to be given to the contents of P after a reset occurs.

Vcc Decoupling

The 3870 family devices have dynamic circuitry internally which requires a good high frequency decoupling capacitor to surpress noise on the Vcc line. A .01 μF or .1 μF ceramic capacitor should be placed between Vcc and ground, located physically close to the 3870 device. This will reduce noise generated by the 3870 to about 70-100mVolts on the Vcc line.

Test Logic

Special test logic is implemented to allow access to the internal main data bus for test purposes.

In normal operation the TEST pin is unconnected or is connected to GND. When TEST is placed at a TTL level (2.0V to 2.6V) port 4 becomes an output of the internal data bus and port 5 becomes a wired-OR input to the internal data bus. The data appearing on the port 4 pins is logically true whereas input data forced on port 5 must be logically false. When TEST is placed at a high level (6.0V to 7.0V), the ports act as above and additionally the 2K x 8 program ROM is prevented from driving the data bus. In this mode operands and instructions may be forced externally through port 5 instead of being accessed from the program ROM. When TEST is in either the TTL state or the high state, $\overline{\text{STROBE}}$ ceases its normal function and becomes a machine cycle clock (identical to the F8 multi-chip system WRITE clock except inverted).

Timing complexities render the capabilities associated with the TEST pin impractical for use in a user's application, but these capabilities are thoroughly sufficient to provide a rapid method for thoroughly testing the 3870.

3870 Clocks

The time base for the 3870 may originate from one of four sources.

The four configurations are shown in Figure 8. There is an internal 26pF capacitor between XTL 1 and GND and an internal 26pF capacitor between XTL 2 and GND. Thus external capacitors are not necessarily required. In all external clock modes the external time base frequently is divided by two to form the internal Φ clock.

Crystal Selection

The use of a crystal as the time base is highly recommended as the frequency stability and reproducability from system to system is unsurpassed. The 3870 has an internal divide by two to allow the user of inexpensive and widely available TV Color Burst Crystals (3.58MHz). The following crystal parameters and vendors are suggested for 3870 applications:

Parameters

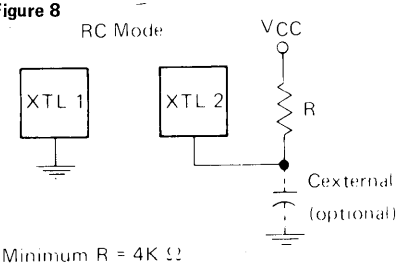
a) Parallel Resonance, Fundamental Mode AT-Cut, HC-33/μ holder
b) Frequency Tolerance measured with 18pF load (0.1% accuracy). Drive level 10mW.
c) Shunt Capacitance (Co) = 7pF max.
d) Series Resistance (Rs)

f = 1MHz	Rs = 550 ohms max.
f = 2MHz	Rs = 300 ohms max.
f = 3MHz	Rs = 100 ohms max.
f = 3.58MHz	Rs = 100 ohms max.
f = 4MHz	Rs = 100 ohms max.

CLOCK CONFIGURATION

Figure 8

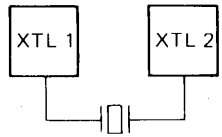
RC Mode



Minimum R = 4K Ω

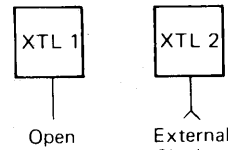
C = 26.5 pF ± 2.6pF + Cexternal

Crystal Mode

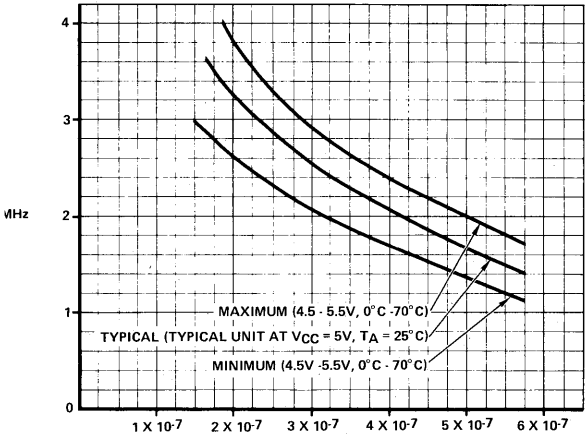


AT Cut 1 – 4 MHz

External Mode



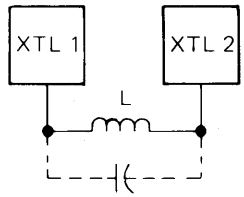
FREQUENCY VRS RC



UNIT TO UNIT VARIATION = ± 12%
VARIATION FROM 4.5 to 5.5V
REFERENCED TO 5V = +7% -4%
VARIATION FROM 0°C TO 70°C
REFERENCED TO 25°C = +6% -9%

TOTAL VARIATION NOT CONSIDERING
VARIATION IN EXTERNAL COMPONENTS = ± 25%

LC Mode



Cexternal (optional)

Minimum L = 0.1 mH
Minimum Q = 40

Maximum Cexternal = 30pF

C = 13pF ± 1.3pF + Cexternal

$$f \quad \frac{1}{2\pi\sqrt{LC}}$$

Suggested Crystal Vendors

a) Electro-Dynamics
5625 Foxridge Drive
Mission, Kansas 66201
913-262-2500

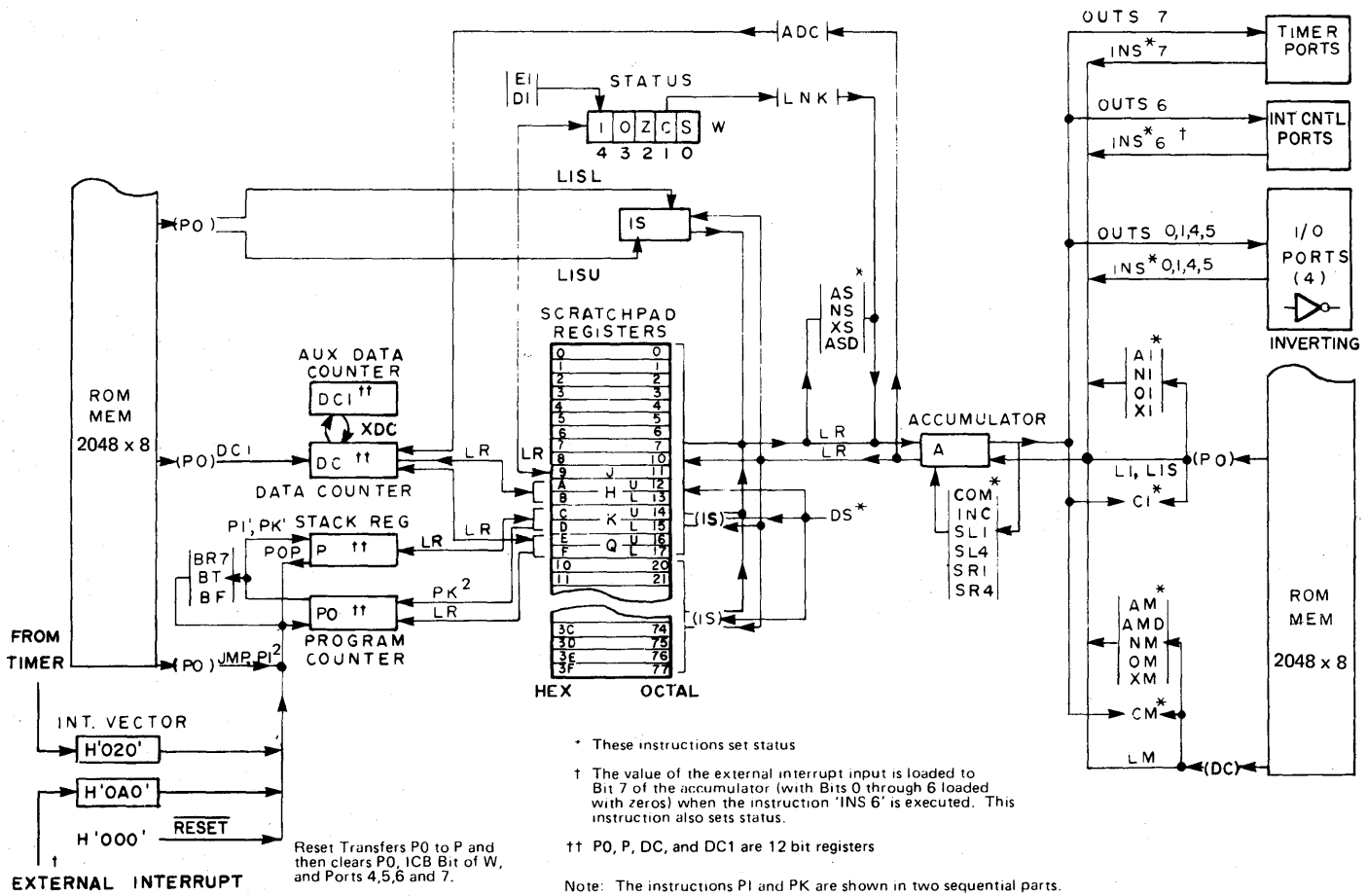
b) CRYSTEK
1000 Crystal Drive
Ft. Myers, Florida 33901
813-936-2109

c) W.T. Liggett Corp.
1500 Worcester Rd.
Section 30
Framingham, MA 01701
617-620-1150

d) Erie Frequency Control
453 Lincoln Street
Carlisle, Penn 17013
717-249-2232

e) Electronic Crystals Corp.
1153 Southwest Blvd.
Kansas City, Kansas 66103
913-262-1274

f) M-TRON Industries
P.O. Box 630
100 Douglas Avenue
Yankton, South Dakota
605-665-9321



Reset Transfers P0 to P and then clears P0, ICB Bit of W, and Ports 4,5,6 and 7.

* These instructions set status

† The value of the external interrupt input is loaded to Bit 7 of the accumulator (with Bits 0 through 6 loaded with zeros) when the instruction 'INS 6' is executed. This instruction also sets status.

†† P0, P, DC, and DC1 are 12 bit registers

Note: The instructions PI and PK are shown in two sequential parts. (PI1, PI2 and PK1, PK2).

MK3870 PROGRAMMING MODEL
Figure 9

INSTRUCTION EXECUTION

This section details the timing and execution of the 3870 instruction set. The 3870 executes the entire F8 instruction set with exact F8 timing. Refer to Figure 11 for a 3870 Programming Model.

F8 INSTRUCTION SET

ACCUMULATOR GROUP INSTRUCTIONS

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	CYCLES LONG	μS (2MHzΦ)	OVR	STATUS BITS ZERO	STATUS BITS CRY	STATUS BITS SIGN
Add Carry	LNK		A←(A) + CRY	19	1	1		2	1/0	1/0	1/0	1/0
Add Immediate	AI	ii	A←(A) + H'ii'	24ii	2	1	1	5	1/0	1/0	1/0	1/0
And Immediate	NI	ii	A←(A)∧H'ii'	21ii	2	1	1	5	0	1/0	0	1/0
Clear	CLR		A←H'00'	70	1	1		2	—	—	—	—
Compare Immediate	CI	ii	H'ii' + (A) + 1	25ii	2	1	1	5	1/0	1/0	1/0	1/0
Complement	COM		A←(A) + H'FF'	18	1	1		2	0	1/0	0	1/0
Exclusive or Immediate	XI	ii	A←(A) + H'ii'	23ii	2	1	1	5	0	1/0	0	1/0
Increment	INC		A←(A) + 1	1F	1	1		2	1/0	1/0	1/0	1/0
Load Immediate	LI	ii	A←H'ii'	20ii	2	1	1	5	—	—	—	—
Load Immediate Short	LIS	i	A←H'0i'	7i	1	1		2				
OR Immediate	OI	ii	A←(A) V H'ii'	22ii	2	1	1	5	0	1/0	0	1/0
Shift Left One	SL	1	Shift Left 1	13	1	1		2	0	1/0	0	1/0
Shift Left Four	SL	4	Shift Left 4	15	1	1		2	0	1/0	0	1/0
Shift Right One	SR	1	Shift Right 1	12	1	1		2	0	1/0	0	1
Shift Right Four	SR	4	Shift Right 4	14	1	1		2	0	1/0	0	1

BRANCH INSTRUCTIONS In all conditional branches P0←(P0) + 2 if the test condition is not met. Execution is complete in 3 short cycles.

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	CYCLES LONG	μS (2MHzΦ)	OVR	STATUS BITS ZERO	STATUS BITS CRY	STATUS BITS SIGN
Branch on Carry	BC	aa	P0←(P0) + 1 + H'aa' if CRY	82aa	2	2	1	7	—	—	—	—
Branch on Positive	BP	aa	P0←(P0) + 1 + H'aa' if SIGN 1	81aa	2	2	1	7	—	—	—	—
Branch on Zero	BZ	aa	P0←(P0) + 1 + H'aa' if Zero 1	84aa	2	2	1	7	—	—	—	—
Branch on True	BT	taa	P0←(P0) + 1 + H'aa' if any test is true	8taa	2	2	1	7	—	—	—	—
Branch If Negative	8M	aa	P0←(P0) + 1 + H'aa' if SIGN 0	91aa	2	2	1	7	—	—	—	—
Branch if No Carry	BNC	aa	P0←(P0) + 1 + H'aa' if CARRY 0	92aa	2	2	1	7	—	—	—	—
Branch if No Overflow	BNO	aa	P0←(P0) + 1 + H'aa' if OVR 0	98aa	2	2	1	7		—	—	—
Branch if Not Zero	BNZ	aa	P0←(P0) + 1 + H'aa' if ZERO 0	94aa	2	2	1	7	—	—	—	—
Branch if False Test	BF	taa	P0←(P0) + 1 + H'aa' if all false test bits	9taa	2	2	1	7	—	—	—	—
Branch if ISAR (Lower) /7	BR7	aa	P0←(P0) + 1 + H'aa' if ISARL /7	8Faa	2	2	1	5	—	—	—	—
			P0←(P0) + 2 if ISARL =		2	2		4	—	—	—	—
Branch Relative	BR	aa	P0←(P0) + 1 + H'aa'	90aa	2	2	1	7	—	—	—	—
Jump*	JMP	aaaa	P0←H'aaaa'	29aaaa	3	1	3	11	—	—	—	—

TEST CONDITION

2^2	2^1	2^0
ZERO	CRY	SIGN

TEST CONDITION

2^4	2^3	2^2	2^1	2^0
OVF	ZERO	CRY	SIGN	

*Privileged instruction, Accumulator contents altered during execution JMP

MEMORY REFERENCE INSTRUCTIONS In all Memory Reference Instructions, the Data Counter is incremented DC ←(DC)+1

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	CYCLES LONG	μS (2MHz↓)	OVR	STATUS BITS ZERO	STATUS BITS CRY	STATUS BITS SIGN
Add Binary	AM		A←(A) + [(DC)]	88	1	1	1	5	1/0	1/0	1/0	1/0
Add Decimal	AMD		A←(A) + [(DC)] + BCD Adjust	89	1	1	1	5	1/0	1/0	1/0	1/0
AND	NM		A←(A) ∧ [(DC)]	8A	1	1	1	5	0	1/0	0	1/0
Compare	CM		[(DC)] + $\overline{(A)}$ + 1	8D	1	1	1	5	1/0	1/0	1/0	1/0
Exclusive OR	XM		A←(A)⊕[(DC)]	8C	1	1	1	5	0	1/0	0	1/0
Load	LM		A←[(DC)]	16	1	1	1	5	—	—	—	—
Logical OR	OM		A←(A) ∨ [(DC)]	8B	1	1	1	5	0	1/0	0	1/0
Store	ST		A→[(DC)]	17	1	1	1	5	—	—	—	—

ADDRESS REGISTER GROUP INSTRUCTIONS

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	CYCLES LONG	μS (2MHz↓)	OVR	STATUS BITS ZERO	STATUS BITS CRY	STATUS BITS SIGN
Add to Data Counter	ADC		DC←(DC) + (A)	8E	1	1	1	5	—	—	—	—
Call to Subroutine*	PK		P0U←(r12); P0L←(r13), P←(P0)	0C	1	1	2	8	—	—	—	—
Call to Subroutine Immediate*	PI	aaaa	P←(P0), P0←H'aaaa'	28aaaa	3	2	3	13	—	—	—	—
Exchange DC	XDC		(DC)↔(DC1)	2C	1	2		4	—	—	—	—
Load Data Counter	LR	DC,0	DCU←(r14), DCL←(r15)	0F	1	1	2	8	—	—	—	—
Load Data Counter	LR	DC,H	DCU←(r10), DCL←(r11)	10	1	1	2	8	—	—	—	—
Load DC Immediate	DCI	aaaa	DC←H'aaaa'	2Aaaaa	3	3	2	12	—	—	—	—
Load Program Counter	LR	P0,0	P0U←(r14), P0L←(r15)	0D	1	1	2	8	—	—	—	—
Load Stack Register	LR	P,K	PU←(r12), PL←(r13)	09	1	1	2	8	—	—	—	—
Return from Subroutine*	POP		P0→P	1C	1	2		4	—	—	—	—
Store Data Counter	LR	0,DC	r14←(DCU), r15←(DCL)	0E	1	1	2	8	—	—	—	—
Store Data Counter	LR	H,DC	r10←(DCU), r11←(DCL)	11	1	1	2	8	—	—	—	—
Store Stack Register	LR	K,P	r12←(PU); r13←(PL)	08	1	1	2	8	—	—	—	—

SCRATCHPAD REGISTER INSTRUCTIONS (Refer to Scratchpad Addressing Modes)

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	CYCLES LONG	μS (2MHz↓)	OVR	STATUS BITS ZERO	STATUS BITS CRY	STATUS BITS SIGN
Add Binary	AS	r	A←(A) + (r)	Cr	1	1		2	1/0	1/0	1/0	1/0
Add Decimal	ASD	r	A←(A) + (r)	Dr	1	2		4	1/0	1/0	1/0	1/0
Decrement	DS	r	r←(r) + H'FF'	3r	1		1	3	1/0	1/0	1/0	1/0
Load	LR	A,r	A←(r)	4r	1	1		2	—	—	—	—
Load	LR	A,KU	A←(r12)	00	1	1		2	—	—	—	—
Load	LR	A,KL	A←(r13)	01	1	1		2	—	—	—	—
Load	LR	A,QU	A←(r14)	02	1	1		2	—	—	—	—
Load	LR	A,QL	A←(r15)	03	1	1		2	—	—	—	—
Load	LR	r,A	r←(A)	5r	1	1		2	—	—	—	—
Load	LR	KU,A	r12←(A)	04	1	1		2	—	—	—	—
Load	LR	KL,A	r13←(A)	05	1	1		2	—	—	—	—
Load	LR	QU,A	r14←(A)	06	1	1		2	—	—	—	—
Load	LR	QL,A	r15←(A)	07	1	1		2	—	—	—	—
And	NS	r	A←(A) ∧ (r)	Fr	1	1		2	0	1/0	0	1/0
Exclusive Or	XS	r	A←(A) ⊕ (r)	Er	1	1		2	0	1/0	0	1/0

*Privileged instruction, Accumulator contents altered during execution of PI instruction.

MISCELLANEOUS INSTRUCTIONS

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	CYCLES LONG	μS (2MHzΦ)	STATUS BITS OVR	STATUS BITS ZERO	STATUS BITS CRY	STATUS BITS SIGN
Disable Interrupt	DI		RESET ICB	1A	1	1		2	–	–	–	–
Enable Interrupt *	EI		SET ICB	1B	1	1		2	–	–	–	–
Input	IN	04,05,06,07	A←(Input Port aa)	26aa	2	1	2	8	0	1/0	0	1/0
Input Short	INS	0, 1	A←(Input Port 0 or 1)	A0,A1	1	2		4	0	1/0	0	1/0
Input Short	INS	4,5,6,7	A←(Input Port a)	Aa	1	1	2	8	0	1/0	0	1/0
Load ISAR	LR	IS,A	IS←(A)	0B	1	1		2	–	–	–	–
Load ISAR Lower	LISL	bbb	ISL←bbb	6(0bbb)**	1	1		2	–	–	–	–
Load ISAR Upper	LISU	bbb	ISU←bbb	6(1bbb)**	1	1		2	–	–	–	–
Load Status Register *	LR	W,J	W←(r9)	1D	1	2		4	1/0	1/0	1/0	1/0
No Operation	NOP		P0←(P0) + 1	2B	1	1		2	–	–	–	–
Output *	OUT	04,05,06,07	Output Port aa←(A)	27aa	2	1	2	8	–	–	–	–
Output Short	OUTS	0, 1	Output Port 0 or 1←(A)	B0, B1	1	2		4	–	–	–	–
Output Short	OUTS	4,5,6,7	Output Port a←(A)	Ba	1	1	2	8	–	–	–	–
Store ISAR	LR	A,IS	A←(IS)	0A	1	1		2	–	–	–	–
Store Status Reg	LR	J,W	r9←(W)	1E	1	1		2	–	–	–	–

*Privileged instruction

**b = 1 bit immediate operand

NOTES.

Lower case denotes variables specified by programmer

Function Definitions

←	is replaced by
( )	the contents of
(‾)	Binary "1's" complement of
+	Arithmetic Add (Binary or Decimal)
⊕	Logical "OR" exclusive
∧	Logical "AND"
∨	Logical "OR" inclusive
H' '	Hexadecimal digit
[( )]	Contents of memory specified by ( )
a	Address Variable (four bits)
A	Accumulator
b	One bit immediate operand
DC	Data Counter (Indirect Address Register)
DC1	Data Counter 1 (Auxiliary Data Counter)
DCL	Least significant 8 bits of Data Counter Addressed
DCU	Most significant 8 bits of Data Counter Addressed
H	Scratchpad Register 10 and 11
i	Immediate operand (four bits)
ICB	Interrupt Control Bit
IS	Indirect Scratchpad Address Register
ISL	Least Significant 3 bits of ISAR
ISU	Most Significant 3 bits of ISAR
J	Scratchpad Register 9
K	Registers 12 and 13
KL	Register 13
KU	Register 12
P0	Program Counter
P0L	Least Significant 8 bits of Program Counter
P0U	Most Significant 8 bits of Program Counter
P	Stack Register
PL	Least Significant 8 bits of Program Counter
PU	Most Significant 8 bits of Active Stack Register
Q	Registers 14 and 15
QL	Register 15
QU	Register 14
r	Scratchpad Register (any address 0 thru B) (See Below)
W	Status Register

Scratchpad Addressing Modes Using IS. (r ≠ 0 thru B)

r=H'C'	Register Addressed by IS is (Unmodified)
r=H'D'	Register Addressed by IS is Incremented
r=H'E'	Register Addressed by IS is Decremented
r=H'F'	Illegal OP Code.

Status Register

—	No change in condition
1/0	is set to "1" or "0" depending on conditions
CRY	Carry Flag
OVR	Overflow Flag
SIGN	Sign of Result Flag
ZERO	Zero Flag

ELECTRICAL SPECIFICATIONS

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias . . . . . 0°C to 70°C
Storage Temperature . . . . . −65°C to +150°C
Voltage on Any Pin With Respect To Grouns (except open drain pins) . . . . . −1.0V to +7V
Voltage On Open Drain Pins . . . . . −1.0V to +13.5V
Power Dissipation 1.5W
Power Dissipated by any one I/O pin[4] 60mW
Power Dissipated by all I/O pins[4] 600mW

A.C. CHARACTERISTICS – See Figure 12 and 13 for Timing Diagrams

T_A = 0°C to 70°C, V_{CC} = 5V ±10%, I/O POWER DISSIPATION ≤100mW

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	NOTES
XTL 1 XTL 2	t_O(INT)	Time Base Period, internal oscillator	250	1000	ns	4MHz - 1.0MHz
	t_o(EX)	Time base period, all external modes	250	1000	ns	4MHz-1MHz
	$t_{EX(H)}$	External Clock Pulse Width High	90	700	ns	
	$t_{EX(L)}$	External Clock Pulse Width Low	100	700	ns	
Φ	$t_Φ$	Internal Φ Clock Period	$2t_O$			
WRITE	tw	Internal WRITE Clock Period	$4t_Φ$ $6t_Φ$			Short Cycle Long Cycle
I/O	$t_{dI/O}$	Output delay from internal WRITE Clock	0	1000	ns	50pF plus one TTL load
	$t_{sI/O}$	Input Setup time to WRITE Clock	1000		ns	
	$t_{I/O\text{-}s}$	Output valid to STROBE Delay	$3t_Φ$ -1000	$3t_Φ$ +250		I/O load = 50pF + 1 TTL STROBE Load= 50pF + 3 TTL
$\overline{STROBE}$	tsl	$\overline{STROBE}$ Low Time	$8t_Φ$ -250	$12t_Φ$ +250	ns	
$\overline{RESET}$	t_{RH}	$\overline{RESET}$ Hold Time, Low	$6t_Φ$ +750		ns	
EXT INT	t_{EH}	EXT INT Hold Time,	$6t_Φ$ + 750		ns	To trigger interrupt
		Active and Inactive State	$2t_Φ$			To trigger timer

CAPACITANCE

T_A = 25°C, f=2MHz

SYMBOL	PARAMETER	MIN	MAX	UNIT	NOTES
C_{IN}	Input Capacitance: I/O Ports, $\overline{\text{RESET}}$, $\overline{\text{EXTINT}}$, $\overline{\text{RAMPRT}}$, TEST		7	pF	Unmeasured Pins Grounded
C_{XTL}	Input Capacitance: XTL1, XTL2	20.5	32.5	pF	

DC CHARACTERISTICS - See Figures 12-17 for typical curves.

T_A = 0° C to 70° C, V_{CC} = +5V ± 10%, I/O POWER DISSIPATION ≤ 100mW

SYMBOL	PARAMETER	MIN	MAX	UNIT	TEST CONDITIONS
I_{CC}	Power Supply Current		85	mA	Outputs Open
P_D	Power Dissipation		400	mW	Outputs Open
V_{IHEX}	External Clock Input High Level	2.4	5.8	V	
V_{ILHEX}	External Clock Input Low Current	−0.3	0.6	V	
I_{IHEX}	External Clock Input High Current		100	μA	$V_{IHEX} = V_{CC}$
I_{ILEX}	External Clock Input Low Current		100	μA	$V_{ILEX} = V_{SS}$
V_{IH}	Input High Level Ports, $\overline{\text{RESET}}$[1], EXT INT[1]	2.0	5.8	V	
V_{IHOD}	Open Drain Input High Level	2.0	13.2	V	
V_{IL}	Input Low Level Ports, $\overline{\text{RESET}}$[1], EXT INT[1]	−0.3	0.8	V	
I_{IL}	Input Low Current Ports, $\overline{\text{RESET}}$[2], EXT INT[2]		−1.6	mA	V_{IL}=0.4V
I_L	Leakage Current Open drain ports, $\overline{\text{RAMPRT}}$ $\overline{\text{RESET}}$[3], EXT INT[3]		+10 −5	μA	V_{IN}=13.2V V_{IN}=0.0V
I_{OH}	Output High Current Standard ports, $\overline{\text{RESET}}$[2] EXT INT[2]	−100 −30		μA μA	V_{OH}=2.4V V_{OH}=3.9V
I_{OHDD}	OUTPUT High Current Direct Drive Ports	−0.1 −1.5 	 −8.5	mA mA mA	V_{OH} = 2.4V V_{OH}=1.5V V_{OH}=.7V
I_{OL}	Output Low Current IO ports	1.8		mA	V_{OL}=0.4V
I_{OHS}	$\overline{\text{STROBE}}$ Output High Current	−300		μA	V_{OH}=2.4V

DC CHARACTERISTICS (Cont'd)

SYMBOL	PARAMETER	MIN	MAX	UNIT	NOTES
I_{OLS}	$\overline{\text{STROBE}}$ Output Low Current	5.0		mA	V_{OL} = 0.4V
V_{IHRPR}	$\overline{\text{RAMPRT}}$ Input High Level	1.9	5.8	V	Guaranteed .1V less than V_{IH} for $\overline{\text{RESET}}$
V_{ILRPR}	$\overline{\text{RAMPRT}}$ Input Low Level	−0.3	0.4	V	Guaranteed .1V less than V_{IL} for $\overline{\text{RESET}}$

* Stresses above 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 condition 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.

1. $\overline{\text{RESET}}$ and EXT INT have internal Schmit triggers giving minimum .2V hysteresis.
2. $\overline{\text{RESET}}$ or EXT INT programmed with standard pull-up
3. $\overline{\text{RESET}}$ or EXT INT programmed without standard pull-up
4. Power dissipation for I/O pins is calculated by $\Sigma(V_{CC} - V_{IL})(|I_{IL}|) + \Sigma(V_{CC} - V_{OH})(|I_{OH}|) + \Sigma(V_{OL})(I_{OL})$

TIMER AC CHARACTERISTICS

Definitions:

Error = Indicated time value - actual time value

tpsc = tΦ x Prescale Value

Interval Timer Mode:

Single interval error, free running (Note 3) ..... ±6tΦ
Cumulative interval error, free running (Note 3) 0
Error between two Timer reads (Note 2) ..... ±(tpsc + tΦ)
Start Timer to stop Timer error (Notes 1,4) ..... +tΦ to −(tpsc +tΦ)
Start Timer to read Timer error (Notes 1,2) ..... −5tΦ to −(tpsc + 7tΦ)
Start Timer to interrupt request error (Notes 1,3) ..... −2tΦ to −8tΦ
Load Timer to stop Timer error (Note 1) ..... +tΦ to −(tpsc + 2tΦ)
Load Timer to read Timer error (Notes 1,2) ..... −5tΦ to −(tpsc + 8tΦ)
Load Timer to interrupt request error (Notes 1,3) ..... −2t Φ to −9tΦ

Pulse Width Measurement Mode:

Measurement accuracy (Note 4) ..... +t Φ to −(tpsc +2t Φ)
Minimum pulse width of EXT INT pin 2tΦ

Event Counter Mode:

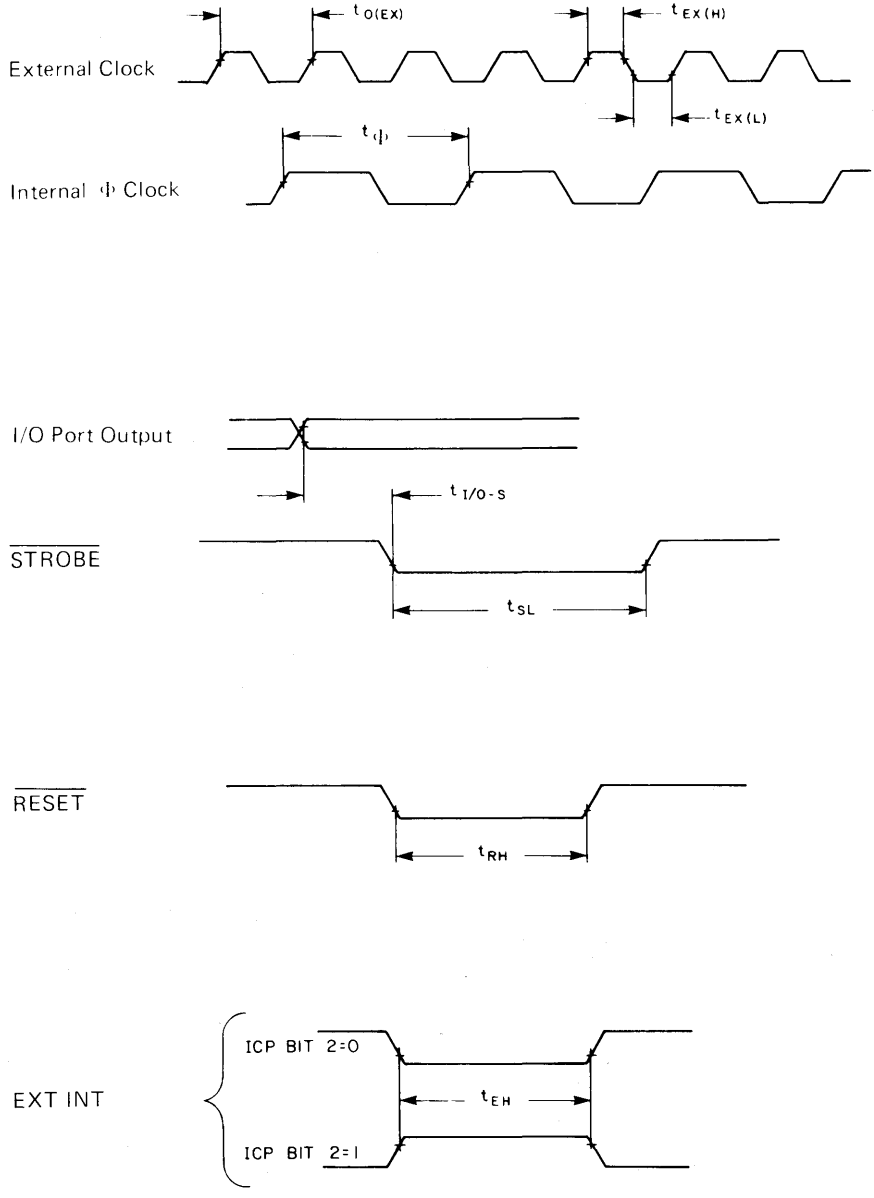
Minimum active time of EXT INT pin 2tΦ
Minimum inactive time of EXT INT pin 2tΦ

Notes:

1. All times which entail loading, starting, or stopping the Timer are referenced from the end of the last machine cycle of the OUT or OUTS instruction.
2. All times which entail reading the Timer are referenced from the end of the last machine cycle of the IN or INS instruction.
3. All times which entail the generation of an interrupt request are referenced from the start of the machine cycle in which the appropriate interrupt request latch is set. Additional time may elapse if the interrupt request occurs during a privileged or multicycle instruction.
4. Error may be cumulative if operation is repetitively performed.

AC TIMING DIAGRAM

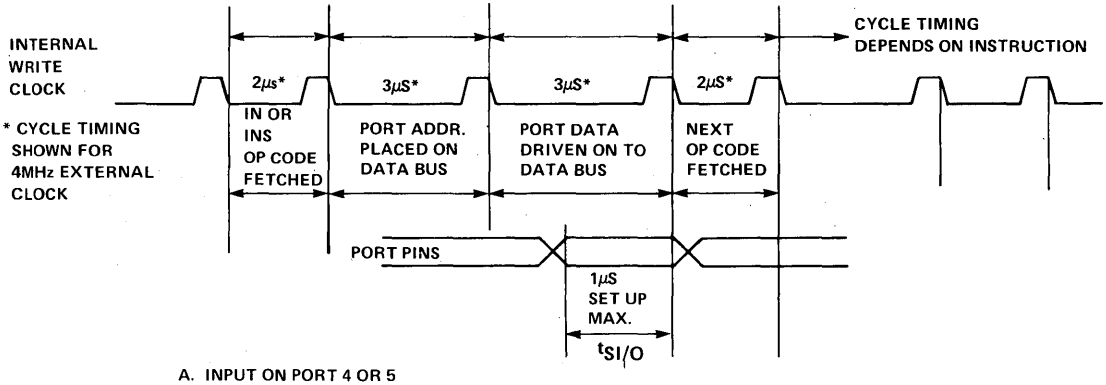
Figure 10



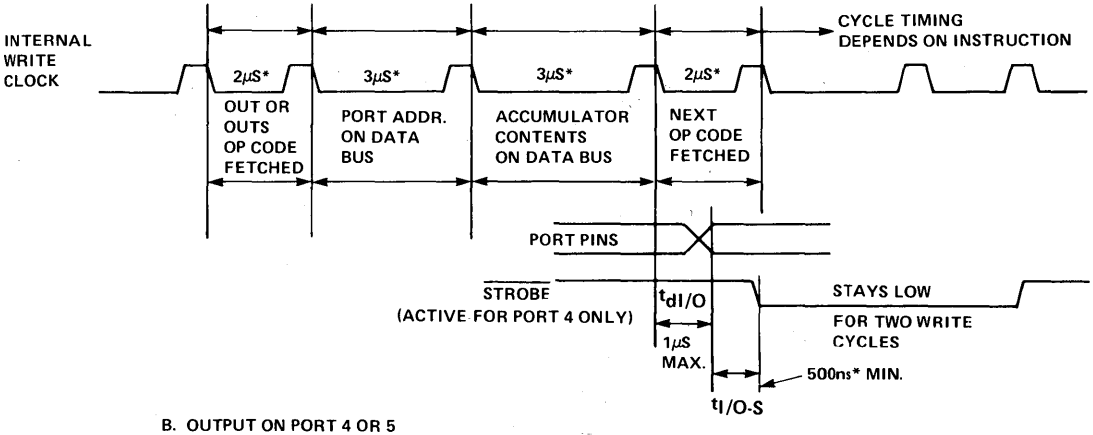
Note: All measurements are referenced to V_{IL} max., V_{IH} min., V_{OL} max., or V_{OH} min.

INPUT/OUTPUT AC TIMING

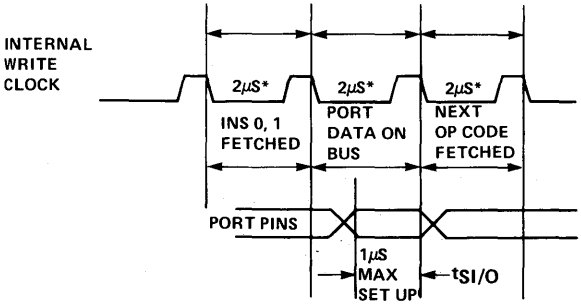
Figure 11



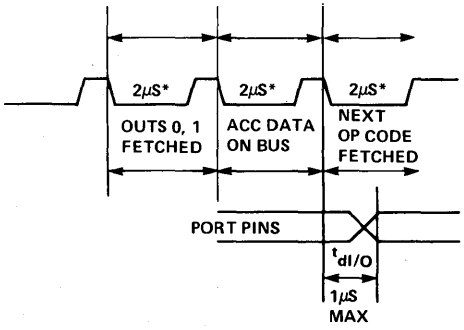
A. INPUT ON PORT 4 OR 5



B. OUTPUT ON PORT 4 OR 5

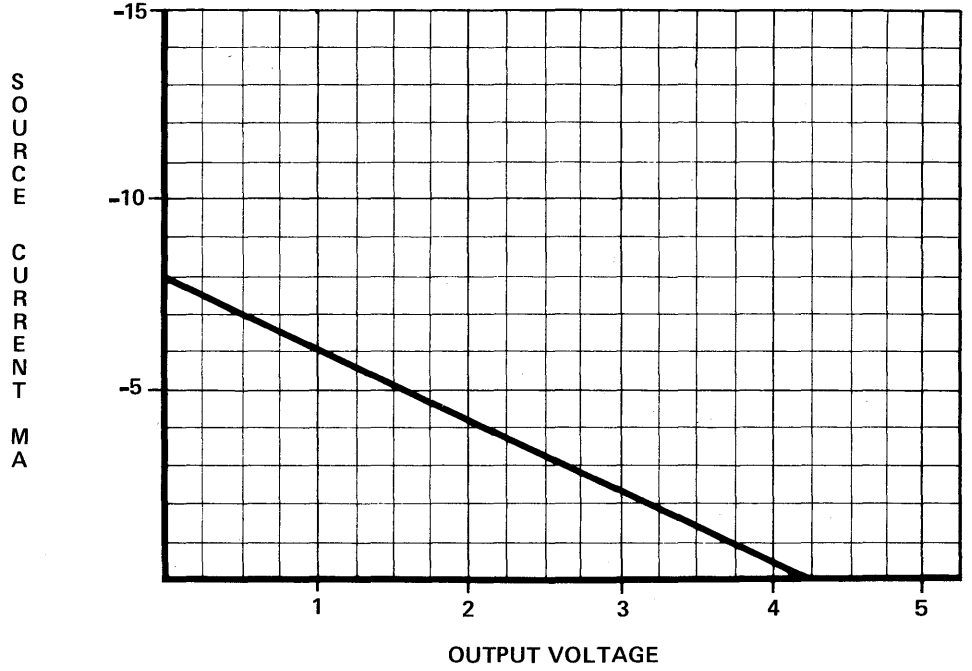


C. INPUT ON PORT 0 OR 1

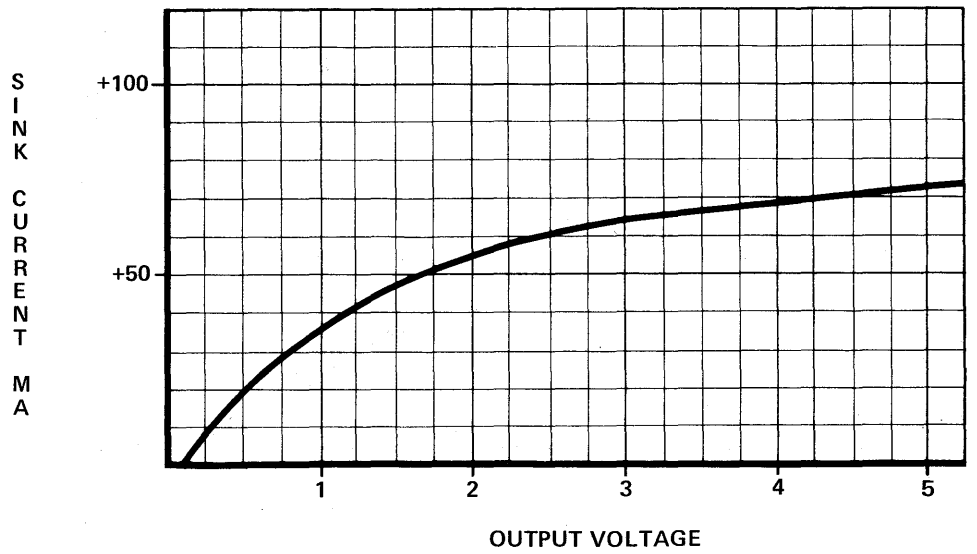


D. OUTPUT ON PORT 0, 1

$\overline{\text{STROBE}}$ SOURCE CAPABILITY
(TYPICAL AT V_{CC} = 5V, T_A = 25° C)
Figure 12

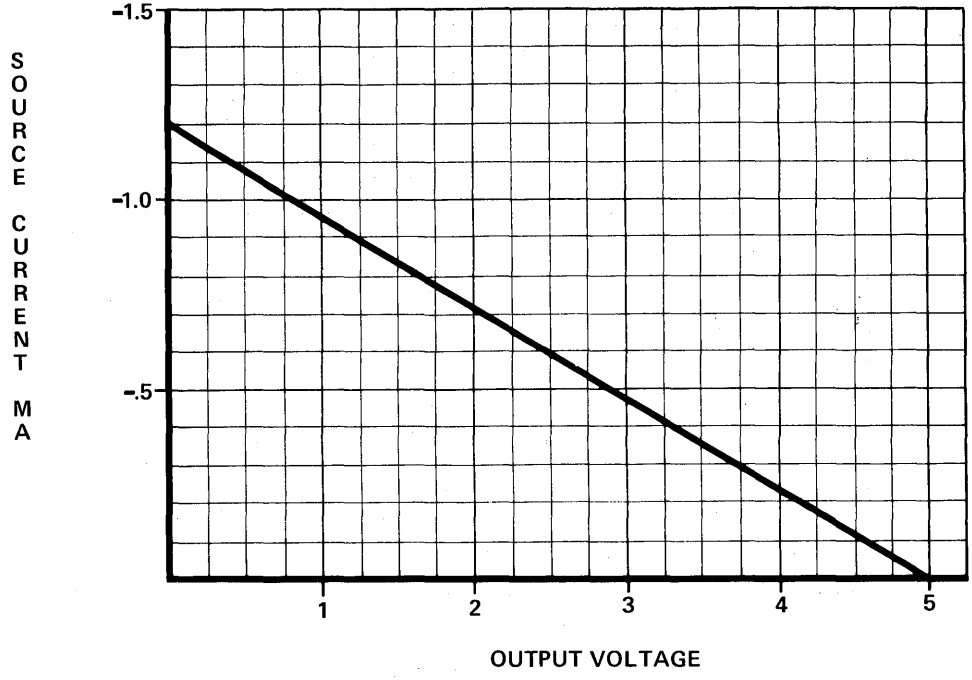


$\overline{\text{STROBE}}$ SINK CAPABILITY
(TYPICAL AT V_{CC} = 5V, T_A = 25° C)
Figure 13



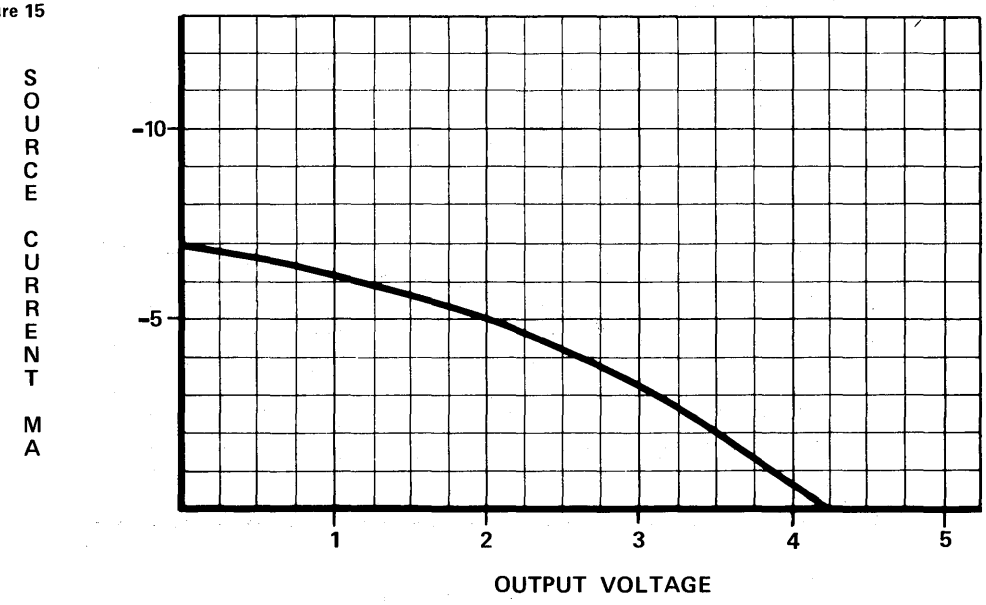
STANDARD I/O PORT SOURCE CAPABILITY
(TYPICAL AT V_{CC} = 5V, T_A = 25°C)

Figure 14



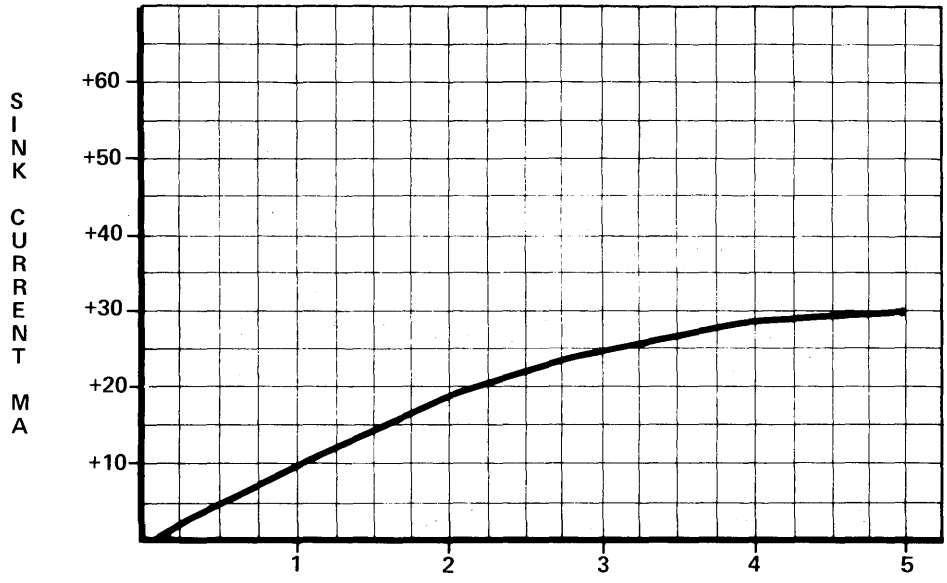
DIRECT DRIVE I/O PORT SOURCE CAPABILITY
(TYPICAL AT V_{CC} = 5V, T_A = 25°C)

Figure 15



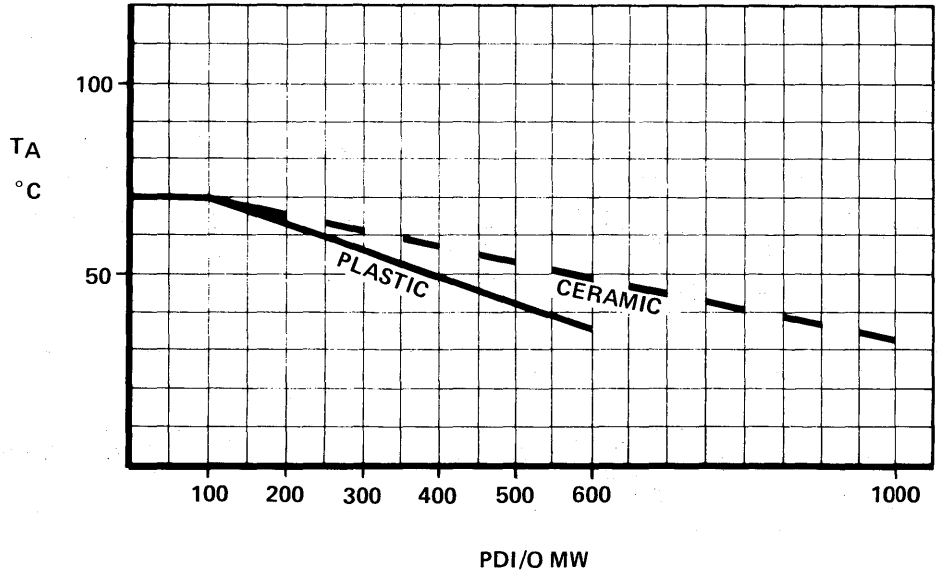
I/O PORT SINK CAPABILITY
(TYPICAL AT V_{CC} = 5V, T_A = 25°C)

Figure 16

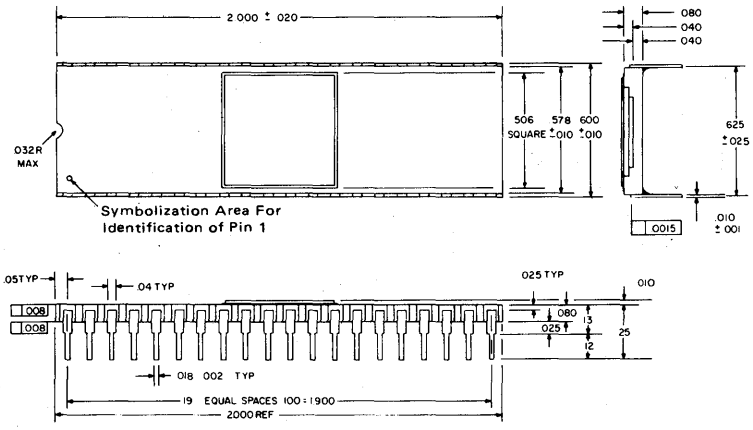


MAXIMUM OPERATING TEMPERATURE VS. I/O POWER DISIPATION

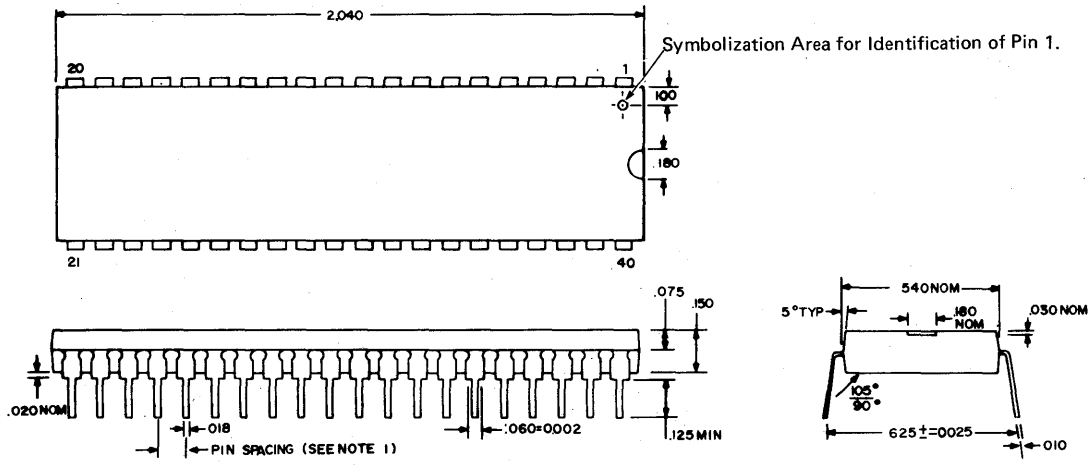
Figure 17



PACKAGE DESCRIPTION: 40-Pin Dual In-Line Ceramic Package



PACKAGE DESCRIPTION 40-Pin Dual-in-Line Plastic Package



ORDERING INFORMATION

PART NO.	PACKAGE TYPE	TEMPERATURE RANGE
MK3870N/14XXX	Plastic	0°C to +70°C
MK3870P/14XXX	Ceramic	0°C to +70°C

APPENDIX A

ORDERING INFORMATION

Custom MK3870 Option Specifications

The custom MK3870 program may be transmitted to Mostek in any of the following media, listed in order of preference:

1) PROMs from the EMU-70
2) Punched paper tape
3) AID-80F Flexible Disk
4) Card Deck (IBM 80 column cards)

The program may be specified in the following forms:

PROMS with correct object code in each location

OBJECT CODE produced by one of Mostek's assemblers.

XFOR-50/70 Fortran IV Cross Assembler, SDB-50/70 resident assembler (ASMB-50/70), AID-80F F8 Cross-Assembler (FZCASM)

OBJECT CODE produced by the dump command from any of Mostek's F8 development hardware (SDB-50/70, AID-80F).

DATA DECK FORMAT as described in the Data Deck section

A completed cover letter (See Fig. A-1) must be attached. The information should be properly packed and mailed prepaid and insured to:

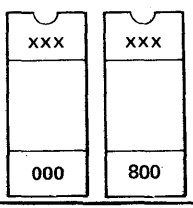
MOSTEK Corporation
Microcomputer Product Marketing
1215 West Crosby Road
Carrollton, Texas 75006

A second copy of the cover letter should be mailed separately to the above address.

PROMS

2708 type PROMs, programmed with the customer program (positive logic sense for addresses and data) may be submitted. The PROMs must be clearly marked to indicate which PROM corresponds to address space 000 7FF and which PROM corresponds to address space 800 FFF. See Fig. A-2 for marking. Include a three-letter customer ID on each PROM. After the PROMs are removed from the EMU-70, they must be placed in a conductive IC carriers and securely packed.

Figure A-2



XXX = Customer ID

Paper Tape

Punched paper tapes (1" wide, 8 level ASCII) will be accepted. The tape must contain the absolute object output from the above mentioned F8 assemblers. Paper object tapes in absolute format generated by the "D" (dump) command of DDT-2 or the dump command of the AID-80F (F8 debug option) are also acceptable if the entire memory space is dumped continuously. Tapes may also be punched using the DATA DECK FORMAT. They must contain 80 characters per record with a CR (carriage return) and LF (line feed) separating each record. The tape must be clearly labeled with customer name, and format used. Fan fold tape is preferred. Tape transparency should be limited to 60% transmissivity (40% opaque). Specifically, thin yellow or white tape is error prone on photo-electric readers and must not be used.

FLEXIBLE DISKS

FLEXIBLE DISKS (Floppy Disks) produced on the Mostek AID-80F development station may be submitted. The format must be the absolute object output from the assemblers, or an object dump using the memory dump command (F8 Debug Option). The disk must be clearly labeled with the format of the data (object, or object dump) and the customer's name.

Punched Card Deck

Standard 80 column punched cards must be used. They must be punched in IBM 029 code. The deck must contain two type of cards:

COMMENT CARDS
DATA CARDS

Comment Cards

Comment Cards must have an asterisk (*) in column 1. The remaining 79 columns may be any character. Comment Cards may be placed anywhere throughout the data deck.

Data Cards

These cards specify the actural ROM data. All fields are right justified.

COLUMN 1:	C (the letter C)
COLUMN 2-9:	ADDR
COLUMN 10-12:	BYTE
COLUMN 14-16:	DATA 1
COLUMN 17-19:	DATA 2
COLUMN 20-22:	DATA 3

3870 ORDERING INFORMATION

DATE _____ CUSTOMER PO NUMBER _____

CUSTOMER NAME _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

COUNTRY _____

PHONE _____ EXTENSION _____

CONTACT _____

CUSTOMER PART NUMBER _____

OPTIONS:

EXTERNAL INTERRUPT: Pull-Up ☐ No Pull-Up ☐

RESET: Pull-Up ☐ No Pull-Up ☐

PORT OPTIONS:

	STANDARD TTL	OPEN DRAIN	DRIVER PULL-UP
P4-0	☐	☐	☐
P4-1	☐	☐	☐
P4-2	☐	☐	☐
P4-3	☐	☐	☐
P4-4	☐	☐	☐
P4-5	☐	☐	☐
P4-6	☐	☐	☐
P4-7	☐	☐	☐
P5-0	☐	☐	☐
P5-1	☐	☐	☐
P5-2	☐	☐	☐
P5-3	☐	☐	☐
P5-4	☐	☐	☐
P5-5	☐	☐	☐
P5-6	☐	☐	☐
P5-7	☐	☐	☐

PATTERN MEDIA

☐ PROMS
(Customer can send in two extra PROM's, MOSTEK will program the customer's code on these PROM's for code verification in the Emulator-70.)

☐ PAPER TAPE (DATA DECK)

☐ PAPER TAPE (OBJECT)

☐ CARD DECK (DATA DECK)

☐ DISKETTE (OBJECT)

THESE ITEMS MAY AFFECT COST

BRANDING REQUIREMENT (If any, 10 Alpha-numeric digits allowed)

_____

_____

PROTOTYPE QUANTITY (10 pieces at no charge - higher quantity extra charge)

_____

WAIVE PROTOTYPES (Customer accepts liability for all work in process)

Yes_____ No _____

SIGNATURE_____

TITLE _____

COLUMN 76-78:	DATA 21
COLUMN 77-79:	DATA 22 or SEQUENCE NUMBER

ADDR is the address of the first byte of data (DATA 1) contained on that card. Successive data bytes read from that card will be placed in successively greater address locations. BYTE is the number of data bytes to be read from that card (1 to 22). If sequence numbers are used, the maximum number of bytes per card is 21. The base for ADDR and BYTE may be either decimal or hex but both must be the same. Data may be either in decimal or hex regardless of the base used for ADDR and BYTE. The base for sequence numbers (if they are used) is always decimal. The bases must be consistant throughout the deck. Data cards need not occur in order of increasing or decreasing addresses. Any unspecified address will be filled with zero. Any unpunched field will be read as a zero. If two data cards specify data for the same address, the one encountered second in the deck will override the first.

A portion of an example deck is shown.

```
*   3870 DATA DECK
*   MOSTEK CORP, EXAMPLE APPLICATION
*   ADDR/BYTE ARE IN DECIMAL
*   DATA IS IN HEX
C  0  8  20  FF  OB  54  34  56  71  B6
C  8  8  1B  28  03  F3  4C  25  2E  94
C  16  8  04  29  01  00
*   START OF SUBROUTINE ALPHA
C  1096  4  20  32  7C  53
C  1100  4  52  47  29  06
C  1104  1  07
```

Verification Media

All original pattern media (PROMs, paper tape, etc.) are filed for contractural purposes and are not returned. Two copies of computer listings printed during the creation of the custom mask pattern are returned. One copy may be kept by the customer. The other copy should be checked thoroughly, signed, and returned to Mostek. The signed listing constitutes the contractual agreement for creation of the customer mask. Though the computer listing serves as the actual verification media, Mostek will program 2708 PROMs programmed from the data file used to create the custom mask to aid in the verification process. If programmed PROMs are desired, two blank 2708 type PROMs must be provided by the customer.

PRELIMINARY

MOSTEK®

F8 MICROCOMPUTER DEVICES

Single-Chip Microcomputer MK3872

SINGLE CHIP μC-4K ROM
MK3872(P/N)

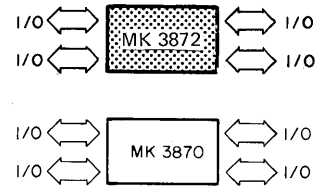
FEATURES

- ☐ Software compatible with F8 family
- ☐ 4032 x 8 mask programmable ROM
- ☐ 64 byte scratchpad RAM
- ☐ 64 additional bytes of executable RAM addressable by program counter or data counter
- ☐ Standby option for executable RAM including:
  - —Low standby power, less than 8.2mW
  - —Minimum 2.2V standby supply voltage
  - —No external components required to trickle charge battery
- ☐ 32 bits (4 ports) TTL Compatible I/O
- ☐ Programmable binary timer
  - —Internal timer mode
  - —Pulse width measurement mode
  - —Event counter mode
- ☐ External interrupt
- ☐ Crystal, LC, RC, external, or internal time base
- ☐ Low power (285mW typ.)
- ☐ Single +5 volt ± 10% power supply
- ☐ Same pinout as MK3870

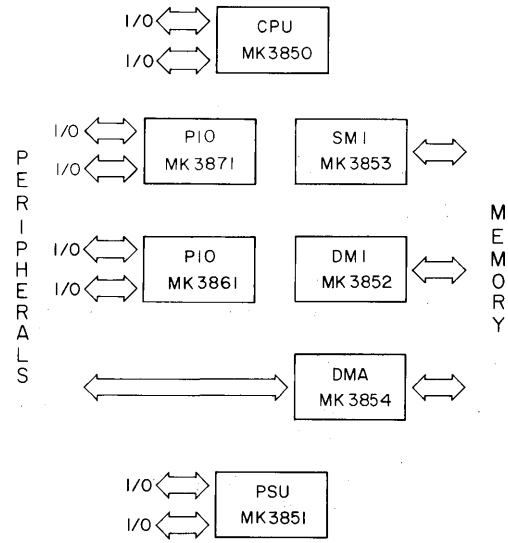
GENERAL DESCRIPTION

The MK3872 is a complete 8-bit microcomputer on a single MOS integrated circuit. The 3872 can execute the F8 instruction set of more than 70 commands, allowing expansion into multi-chip configurations with software compatibility. The device features 4032 bytes of ROM, 64 bytes of scratchpad RAM, 64 bytes of executable RAM, a programmable binary timer, 32 bits of I/O, and a single +5 volt power supply requirement. Utilizing ion-implanted, N-channel silicon gate technology and advanced circuit design techniques the single-chip 3872 offers maximum cost-effectiveness in a wide range of control and logic replacement applications. The 3872 is an expanded memory version of the 3870 single chip microcomputer. The 3872 is identical to the 3870 in the following areas: instruction set, architecture, AC and DC characteristics, and pinout. The only change is in the memory expansion along with the appropriate memory address registers.

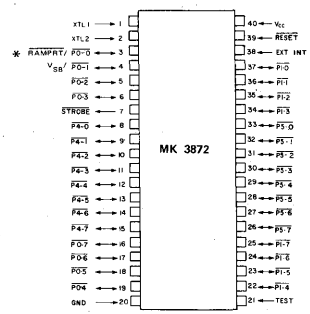
SINGLE CHIP 3870 MICROCOMPUTER FAMILY



F8 FAMILY SERIAL I/O



PIN CONNECTIONS



*PROGRAMMABLE (PORT PINS BECOME V_{SB} AND $\overline{RAMPRT}$ WITH STANDBY POWER OPTION)

PIN NAME	DESCRIPTION	TYPE
$\overline{P0\text{-}0} - \overline{P0\text{-}7}$	I/O Port 0	Bidirectional
$\overline{P1\text{-}0} - \overline{P1\text{-}7}$	I/O Port 1	Bidirectional
$\overline{P4\text{-}0} - \overline{P4\text{-}7}$	I/O Port 4	Bidirectional
$\overline{P5\text{-}0} - \overline{P5\text{-}7}$	I/O Port 5	Bidirectional
$\overline{STROBE}$	Ready Strobe	Output
EXT INT	External Interrupt	Input
$\overline{RESET}$	External Reset	Input
TEST	Test Line	Input
XTL 1, XTL 2	Time Base	Input
V_{CC}, GND	Power Supply Lines	Input
V_{SB}, $\overline{RAMPRT}$	Standby Power, RAM Protect	Input

FUNCTIONAL PIN DESCRIPTION

$\overline{P0\text{-}0}$–$\overline{P0\text{-}7}$, $\overline{P1\text{-}0}$–$\overline{P1\text{-}7}$, $\overline{P4\text{-}0}$–$\overline{P4\text{-}7}$, and $\overline{P5\text{-}0}$–$\overline{P5\text{-}7}$ are 32 lines which can be individually used as either TTL compatible inputs or as latched outputs.

$\overline{STROBE}$ is a ready strobe associated with I/O Port 4. This pin which is normally high provides a single low pulse after valid data is present on the P4-0–P4-7 pins during an output instruction.

$\overline{RESET}$ may be used to externally reset the 3872. When pulled low the 3872 will reset. When allowed to go high the 3872 will begin program execution at program location H '000'.

EXT INT is the external interrupt input. Its active state is software programmable. This input is also used in conjunction with the timer for pulse width measurement and event counting.

XTL 1 and XTL 2 are the time base inputs to which a crystal (1 to 4MHz), LC network, RC network, or an external single-phase clock may be connected. If timing is not critical, the 3872 will operate from its internal oscillator with no external components.

TEST is an input, used only in testing the 3872. For normal circuit functionality this pin is left unconnected or may be grounded.

V_{CC} is the power supply input (+5V±10%).

V_{SB} is the RAM standby power supply input if the standby option is selected (+5.5V to +2.2V).

$\overline{RAMPRT}$ is the RAM protect control when the RAM standby option is selected. When brought to a low level (near V_{SS}) the RAM is disabled and therefore protected against any alterations during loss of V_{CC}.

3870 ARCHITECTURE

This section describes the basic functional elements of the 3872 as shown in the block diagram of Figure 1. A programming model is shown in Figure 2.

Main Control Logic

The Instruction Register (IR) receives the operation code (OP code) of the instruction to be executed from the program ROM via the data bus. During all OP code fetches eight bits are latched into the IR. Some instructions are completely specified by the upper 4 bits of the OP code. In those instructions the lower 4 bits are an immediate register address or an immediate 4 bit operand. Once latched into the IR the main control logic decodes the instruction and provides the necessary control gating signals to all circuit elements.

ROM Address Registers

There are four 12 bit registers associated with the 4K x 8 ROM and 64 x 8 RAM. These are the Program Counter (P0), the Stack Register (P), the Data Counter (DC) and the Auxiliary Data Counter (DC1). The Program Counter is used to address instructions or immediate operands. P is used to save the contents of P0 during an interrupt or subroutine call. Thus, P contains the return address at which processing is to resume upon completion of the subroutine or the interrupt routine.

The Data Counter (DC) is used to address data tables. This register is auto-incrementing. Of the two data counters only DC can access the memory. However, the XDC instruction allows DC and DC1 to be exchanged.

Associated with the address registers is a 12 bit Adder/Incrementer. This logic element is used to increment P0 or DC when required and is also used to add displacements to P0 on relative branches or to add the data bus contents to DC in the ADC (add data counter) instruction.

4032 x 8 ROM

The microcomputer program and data constants are stored in the program ROM. When a ROM access is required, the appropriate address register (P0 or DC) is gated onto the ROM address bus and the ROM output is gated onto the main data bus. The first byte in ROM is location zero.

64 x 8 Executable RAM

The upper 64 bytes of the total 4096 byte memory of the 3872 is RAM memory. The first byte is at address 4032 decimal (FC0 hex). As with the ROM

memory the RAM memory may be accessed by the P0 and DC address registers. It may be written via the STORE (ST) instruction. It may be read via the LOAD (LM) instruction. Additionally instructions may be executed from the RAM. A mask programmable standby power option is available whereby the 64x8 RAM remains powered and protected so that its contents are saved during a loss of the normal circuit power supply.

Scratchpad and IS

The scratchpad provides 64 8-bit registers which may be used as general purpose RAM memory. The Indirect Scratchpad Address Register (IS) is a 6 bit register used to address the 64 registers. All 64 registers may be accessed using IS. In addition the lower order 12 registers may also be directly addressed.

IS can be visualized as holding two octal digits. This division of IS is important since a number of instructions increment or decrement only the least significant 3 bits of IS when referencing scratchpad bytes via IS. This makes it easy to reference a buffer consisting of contiguous scratchpad bytes. For example, When the low order octal digit is incremented or decremented IS is incremented from octal 27 (0 '27') to 0 '20' or is decremented from 0 '20' to 0 '27'. This feature of the IS is very useful in many program sequences. All six bits of IS may be loaded at one time or either half may be loaded independently.

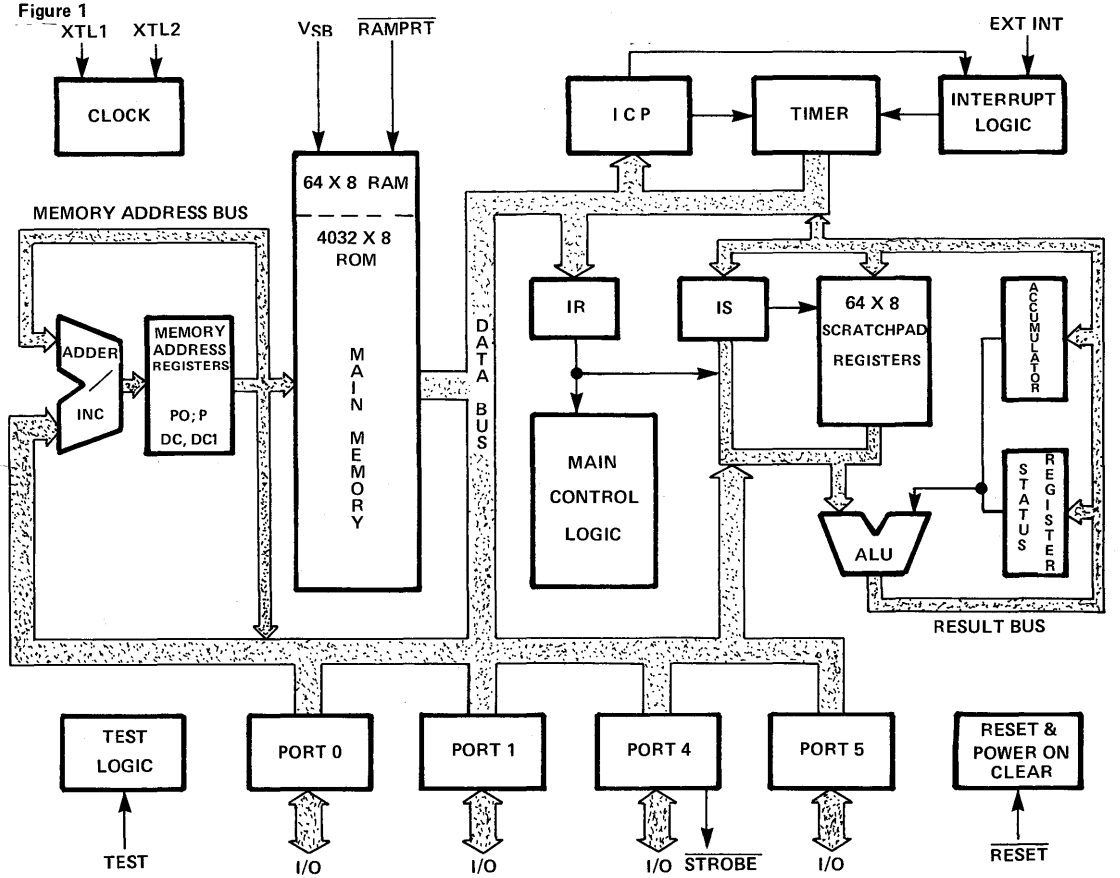
Scratchpad registers 9 through 15 (decimal) are given mnemonic names (J, H, K, and Q) because of special linkages between these registers and other registers such as the Stack Register. These special linkages facilitate the implementation of multi-level interrupts and subroutine nesting. For example, the instruction LRK, P stores the lower eight bits of the Stack Register into register 13 (K lower or KL) and stores the upper three bits of P into register 12 (K upper or KU) The scratchpad is not protected with the standby power option.

Arithmetic and Logic Unit (ALU)

After receiving commands from the main control

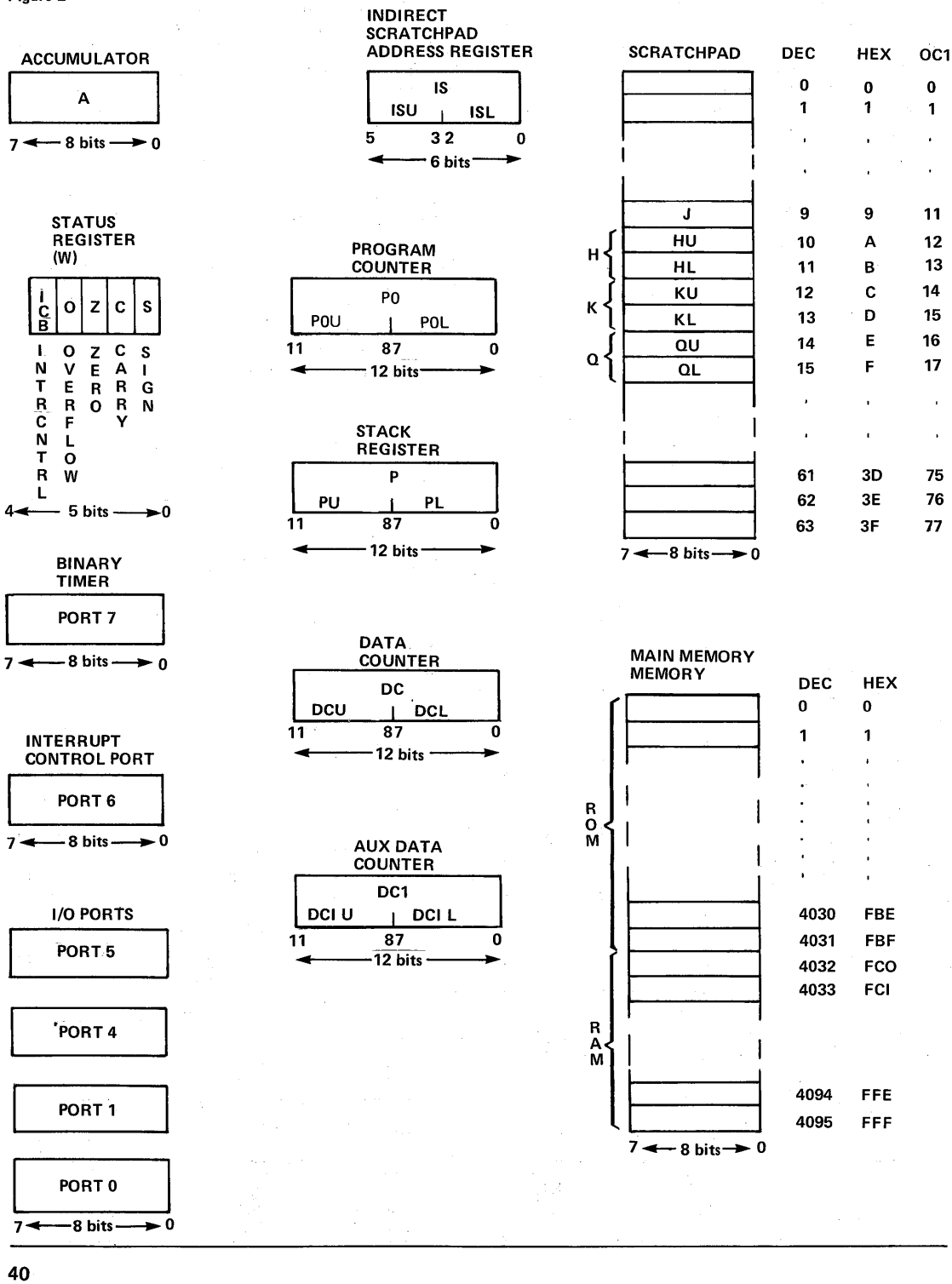
MK 3872 BLOCK DIAGRAM

Figure 1



3872 PROGRAMMABLE REGISTERS, PORTS AND MEMORY MAP

Figure 2



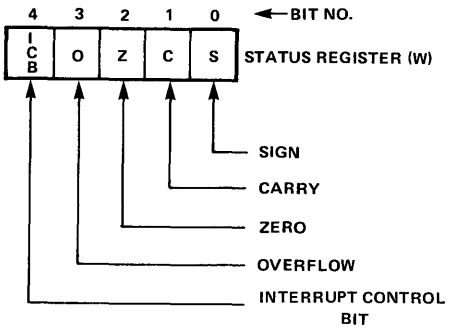
logic, the ALU performs the required arithmetic or logic operations (using the data presented on the two input busses) and provides the result on the result bus. The arithmetic operations that can be performed in the ALU are binary add, decimal adjust, add with carry, decrement, and increment. The logic operations that can be performed are AND, OR, EXCLUSIVE OR, 1's complement, shift right, and shift left. Besides providing the result on the result bus, the ALU also provides four signals representing the status of the result. These signals, stored in the Status Register (W), represent CARRY, OVERFLOW, SIGN, and ZERO condition of the result of the operation.

Accumulator (A)

The Accumulator (A) is the prinicpal register for data manipulation within the 3872. A serves as one input to the ALU for arithmetic or logical operations. The result of ALU operations are stored in A.

The Status Register (W)

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



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 = $\overline{ALU_7}$

Interrupt Control Bit (ICB)

The ICB may be used to allow or disallow interrupts in the 3872. This bit is not the same as the two interrupt enable bits in the Interrupt Control Port (ICP). If the ICB is set and the 3872 interrupt logic communicates an interrupt request to the CPU section, the interrupt will be acknowledged and processed upon completion of the first non-privileged instruction. If the ICB is cleared an interrupt request will not be acknowledged or processed until the ICB is set.

I/O Ports

The 3872 provides four complete bidirectional Input/Output ports. (When standby option is used, Port 0, bit 0 and 1 are not available). These are Ports 0, 1, 4, and 5. In addition, the Interrupt Control Port is addressed as Port 6 and the binary timer is addressed as Port 7. An output instruction (OUT or OUTS) causes the contents of A to be latched into the addressed port. An input instruction (IN or INS) transfers the contents of the port to A (port 6 is an exception which is described later). The I/O pins on the 3872 are logically inverted. The schematic of an I/O pin and available output drive options are shown in Figure 3.

An output ready strobe is associated with Port 4. This flag may be used to signal a peripheral device that the 3872 has just completed an output of new data to Port 4. The strobe provides a single low pulse shortly after the output operation is completely finished, so either edge may be used to signal the peripheral. $\overline{STROBE}$ may also be used as an input strobe simply by doing a dummy output of H '00' to Port 4 after completing the input operation.

Timer and Interrupt Control Port

The Timer is an 8-bit binary down counter which is software programmable to operate in one of three modes: the Interval Timer Mode, the Pulse Width Measurement Mode, or the Event Counter Mode. As shown in Figure 4, associated with the Timer are an 8-bit register called the Interrupt Control Port, a programmable prescaler, and an 8-bit modulo-N register. A functional logic diagram is shown in Figure 5.

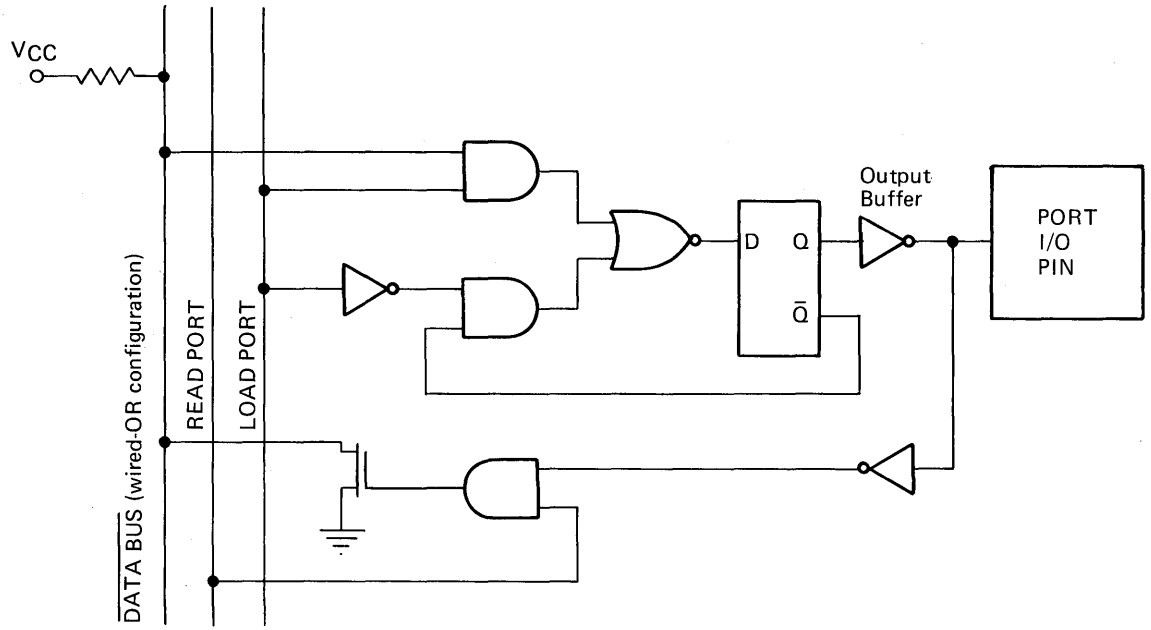
The desired timer mode, prescale value, starting and stopping the timer, active level of the EXT INT pin, and local enabling or disabling of interrupts are selected by outputting the proper bit configuration from the Accumulator to the Interrupt Control Port (Port 6) with an OUT or OUTS instruction. Bits within the Interrupt Control Port are defined as follows:

Interrupt Control Port (Port 6)

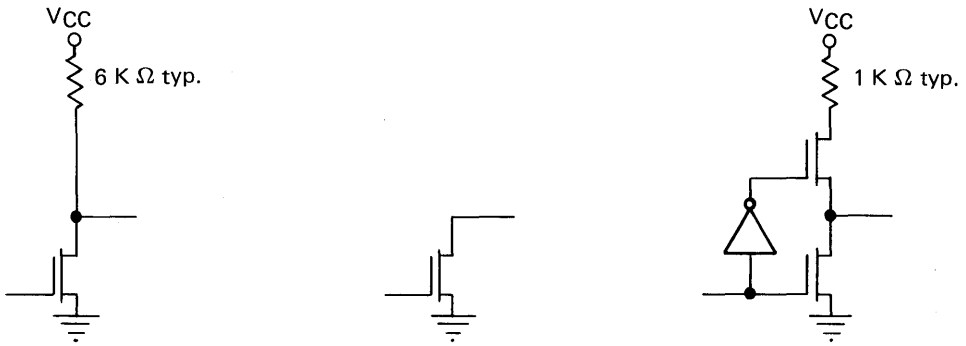
Bit 0 - External Interrupt Enable
Bit 1 - Timer Interrupt Enable
Bit 2 - EXT INT Active Level
Bit 3 - Start/Stop Timer
Bit 4 - Pulse Width/Interval Timer
Bit 5 - ÷ 2 Prescale
Bit 6 - ÷ 5 Prescale
Bit 7 - ÷ 20 Prescale

I/O PIN CONCEPTUAL DIAGRAM WITH OUTPUT BUFFER OPTIONS

Figure 3



OUTPUT BUFFER OPTIONS



Standard Output

Open Drain Output

Direct Drive Output

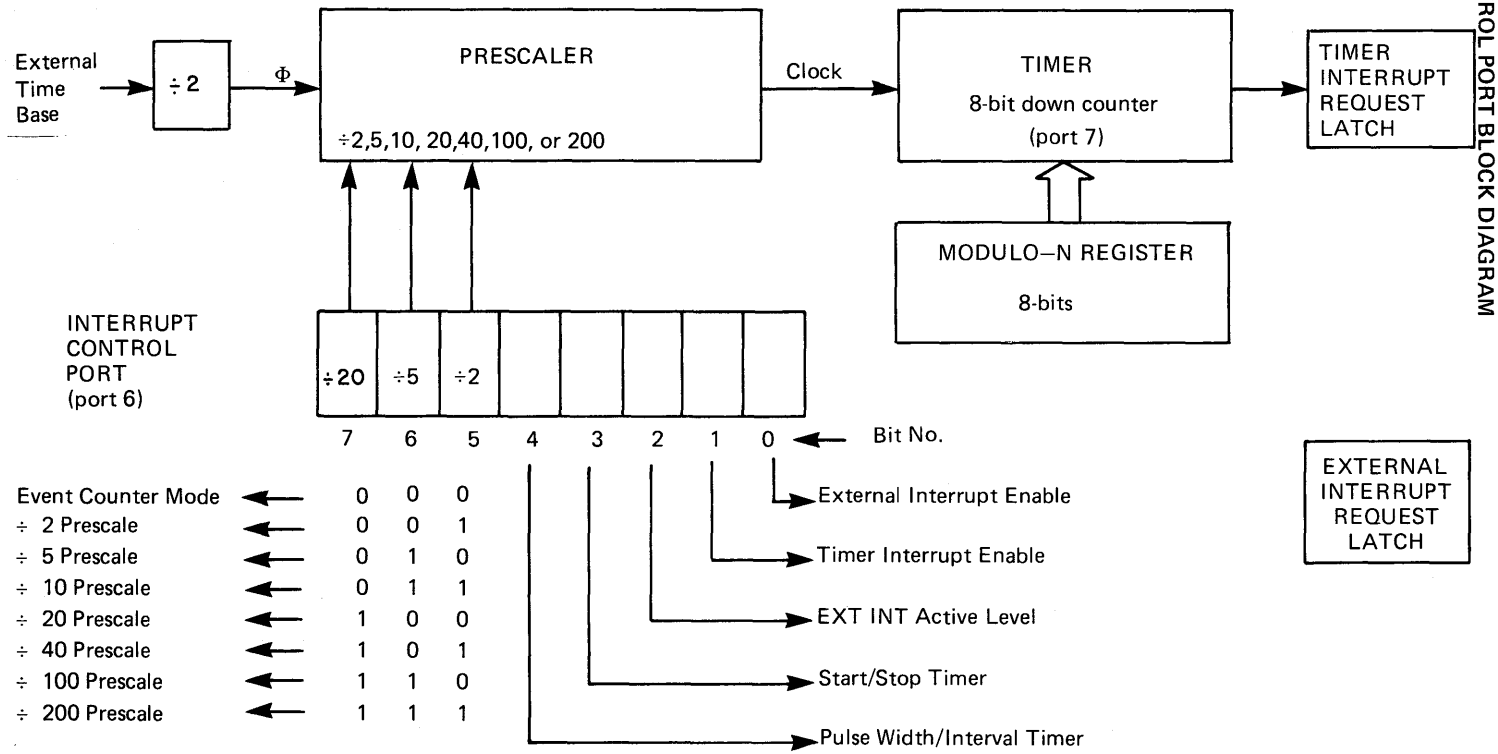
Ports 0 and 1 are Standard Output type only.

Ports 4 and 5 may both be any of the three output options (programmable bit by bit).

The $\overline{\text{STROBE}}$ output is always configured similar to a Direct Drive Output except that it is capable of driving 3 TTL loads.

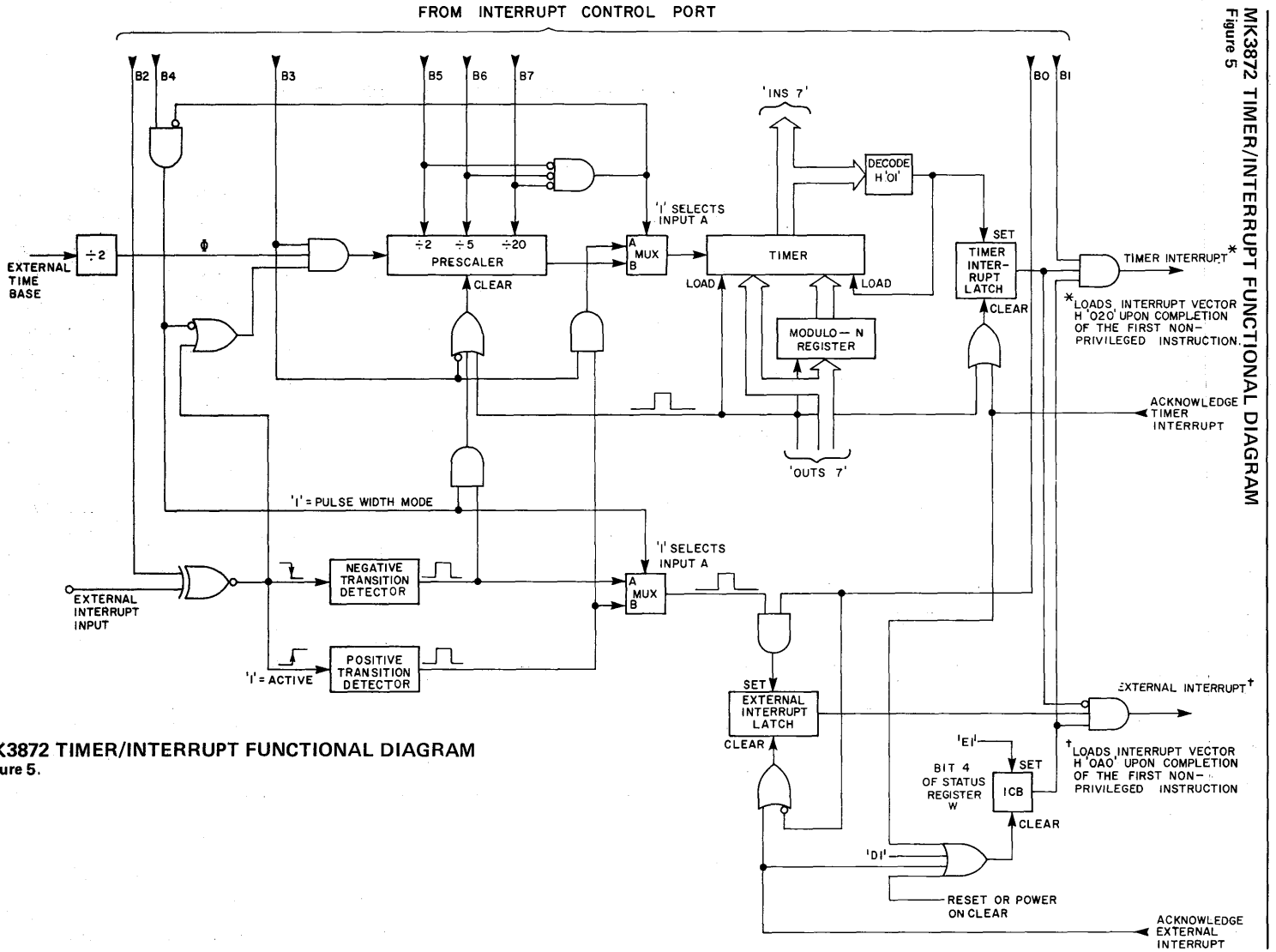
$\overline{\text{RESET}}$ and EXT INT may have standard 6KΩ (typical) pull-up or may have no pull-up.

TIMER & CONTROL PORT BLOCK DIAGRAM
Figure 4



Note: See Figure 5 for a more detailed functional diagram.

TIMER & CONTROL PORT BLOCK DIAGRAM
Figure 4



MK3872 TIMER/INTERRUPT FUNCTIONAL DIAGRAM
Figure 5.

A special situation exists when reading the Interrupt Control Port (with an IN or INS instruction). The Accumulator is not loaded with the content of the ICP; instead, Accumulator bits 0 through 6 are loaded with 0's while bit 7 is loaded with the logic level being applied to the EXT INT pin, thus allowing the status of EXT INT to be determined without the necessity of servicing an external interrupt request. When reading the Interrupt Control Port (Port 6) bit 7 of the Accumulator is loaded with the actual logic level being applied to the EXT INT pin, regardless of the status of ICP bit 2 (the EXT INT Active Level bit); that is, if EXT INT is at +5V bit 7 of the Accumulator is set to a logic 1, but if EXT INT is at GND then Accumulator bit 7 is reset to logic 0. This capability is useful in establishing a high speed polled handshake procedure or for using EXT INT as an extra input pin if external interrupts are not required and the Timer is used only in the Interval Timer Mode. However, if it is desirable to read the contents of the ICP then one of the 64 scratchpad registers or one byte of RAM may be used to save a copy of whatever is written to the ICP.

The rate at which the timer is clocked in the Interval Timer Mode is determined by the frequency of an internal Φ clock and by the division value selected for the prescaler. (The internal Φ clock operates at one-half the external time base frequency). If ICP bit 5 is set and bits 6 and 7 are cleared, the prescaler divides Φ by 2. Likewise, if bit 6 or 7 is individually set the prescaler divides Φ by 5 or 20 respectively. Combinations of bits 5, 6 and 7 may also be selected. For example, if bits 5 and 7 are set while 6 is cleared the prescaler will divide by 40. Thus possible prescaler values are ÷2, ÷5. ÷10, ÷20, ÷40, ÷100, and ÷200.

Any of three conditions will cause the prescaler to be reset: whenever the timer is stopped by clearing ICP bit 3, execution of an output instruction to Port 7, (the timer is assigned port address 7), or on the trailing edge transition of the EXT INT pin when in the Pulse Width Measurement Mode. These last two conditions are explained in more detail below.

An OUT or OUTS instruction to Port 7 will load the content of the Accumulator to both the Timer and the 8-bit modulo-N register, reset the prescaler, and clear any previously stored timer interrupt request. As previously noted, the Timer is an 8-bit down counter which is clocked by the prescaler in the Interval Timer Mode and in the Pulse Width Measurement Mode. The prescaler is not used in the Event Counter Mode. The Modulo-N register is a buffer whose function is to save the value which was most recently outputted to Port 7. The modulo-N register is used in all three timer modes.

Interval Timer Mode

When ICP bit 4 is cleared (logic 0) and at least one prescale bit is set the Timer operates in the Interval Timer Mode. When bit 3 of the ICP is set the Timer will start counting down from the modulo-N value. After counting down to H'01', the Timer returns to the modulo-N value at the next count. On the transition from H'01' to H 'N' the Timer sets a timer interrupt request latch. Note that the interrupt request latch is set by the transition to H 'N' and not be the presence of H 'N' in the Timer, thus allowing a full 256 counts if the modulo-N register is preset to H '00'. If bit 1 of the ICP is set, the interrupt request is passed on to the CPU section of the 3872. However, if bit 1 of the ICP is a logic 0 the interrupt request is not passed on to the CPU section but the interrupt request latch remains set. If ICP bit 1 is subsequently set, the interrupt request will then be passed on to the CPU section. (Recall from the discussion of the Status Register's Interrupt Control Bit that the interrupt request will be acknowledged by the CPU section only if ICB is set). Only two events can reset the timer interrupt request latch; when the timer interrupt request latch is acknowledged by the CPU section, or when a new load of the modulo-N register is performed.

Consider an example in which the modulo-N register is loaded with H '64' (decimal 100). The timer interrupt request latch will be set at the 100th count following the timer start and the timer interrupt request latch will repeatedly be set on precise 100 count intervals. If the prescaler is set at ÷40 the timer interrupt request latch will be set every 4000 Φ clock periods. For a 2MHz Φ clock (4MHz time base frequency) this will produce 2 millisecond intervals.

The range of possible intervals is from 2 to 51,200 Φ clock periods (1μs to 25.6ms for a 2MHzΦ clock). However, approximately 50 Φ periods is a practical minimum because the time between setting the interrupt request latch and the execution of the first instruction of the interrupt service routine is at least 29 Φ periods (the response time is dependent upon how many privileged instructions are encountered when the request occurs). To establish time intervals greater than 51,200 Φ clock periods is a simple matter of using the timer interrupt service routine to count the number of interrupts, saving the result in one or more of the scratchpad registers until the desired interval is achieved. With this technique virtually any time interval, or several time intervals, may be generated.

The Timer may be read at any time and in any mode using an input instruction (IN 7 or INS 7) and may

take place "on the fly" without interferring with normal timer operation. Also, the Timer may be stopped at any time by clearing bit 3 of the ICP. The timer will hold its current contents indefinitely and will resume counting when bit 3 is again set. Recall however that the prescaler is reset whenever the Timer is stopped; thus a series of starting and stopping will result in a cumulative truncation error.

A summary of other timer errors is given in the timing section of this specification. For a free running timer in the Interval Timer Mode the time interval between any two interrupt requests may be in error by ± 6 Φ clock periods although the cumulative error over many intervals is zero. The prescaler and Timer generate precise intervals for setting the timer interrupt request latch but the time out may occur at any time within a machine cycle. (There are two types of machine cycles: short cycles which consist of 4 Φ clock periods and long cycles which consist of 6 Φ clock periods. In the multi-chip F8 family there is a signal called the WRITE clock which corresponds to a machine cycle). Interrupt requests are synchronized with the internal WRITE clock thus giving rise to the possible ± 6 Φ error. Additional errors may arise due to the interrupt request occurring while a privileged instruction or multicycle instruction is being executed. Nevertheless, for most applications all of the above errors are negligible, especially if the desired time intervall is greater than 1ms.

Pulse Width Measurement Mode

When ICP bit 4 is set (logic 1) and at least one prescale bit is set the Timer operates in the Pulse Width Measurement Mode. This mode is used for accurately measuring the duration of a pulse applied to the EXT INT pin. The Timer is stopped and the prescaler is reset whenever EXT INT is at its inactive level. The active level of EXT INT is defined by ICP bit 2; if cleared, EXT INT is active low; if set, EXT INT is active high. If ICP bit 3 is set, the prescaler and Timer will start counting when EXT INT transitions to the active level. When EXT INT returns to the inactive level the Timer then stops, the prescaler resets, and if ICP bit 0 is set an external interrupt request latch is set. (Unlike timer interrupts, external interrupts are not latched if the ICP Interrupt Enable bit is not set).

As in the Interval Timer Mode, the Timer may be read at any time, may be stopped at any time by clearing ICP bit 3, the prescaler and ICP bit 1 function as previously described, and the Timer still functions as an 8-bit binary down counter with the timer interrupt request latch being set on the Timer's transition from H '01' to H 'N'. Note that the EXT INT pin has nothing to do with loading the Timer; its action is that of automatically starting and stopping the Timer and of generating external interrupts. Pulse widths longer than the prescale value times the modulo-N value are easily measured by using the timer interrupt service routine to store the number of timer interrupts in one or more scratchpad registers.

As for accuracy, the actual pulse duration is typically slightly longer than the measured value because the status of the prescaler is not readable and is reset when the Timer is stopped. Thus for maximum accuracy it is advisable to use a small division setting for the prescaler.

Event Counter Mode

When ICP bit 4 is cleared and all prescale bits (ICP bits 5, 6, and 7) are cleared the Timer operates in the Event Counter Mode. This mode is used for counting pulses applied to the EXT INT pin. If ICP bit 3 is set the Timer will decrement on each transition from the inactive level to the active level or the EXT INT pin. The prescaler is not used in this mode, but as in the other two timer modes, the timer may be read at any time, may be stopped at any time by clearing ICP bit 3, ICP bit 1 functions as previously described, and the timer interrupt request latch is set on the Timer's transition from H '01' to H 'N'.

Normally ICP bit 0 should be kept cleared in the Event Counter Mode; otherwise, external interrupts will be generated on the transition from the inactive level to the active level of the EXT INT pin.

For the Event Counter Mode the minimum pulse width required on EXT INT is 2 Φ clock periods and the minimum inactive time is 2 Φ clock periods; therefore, the maximum repetition rate is 500KHz.

Timer Emulation

For total software compatibility when expanding into a multi-chip configuration the MK3871 Peripheral Input/Output circuit should be used rather than the older MK3861 PIO. The MK3871 has the same improved Timer (binary count, readable, and three modes of operation rather than one) and ready strobe output as are on the MK3872.

External Interrupts

When the timer is in the Interval Timer Mode the EXT INT pin is available for non-timer related interrupts. If ICP bit 0 is set an external interrupt request latch is set when there is a transition from the inactive level to the active level of EXT INT. (EXT INT is an edge-triggered input). The interrupt request is latched until either acknowledged by the CPU section or until ICP bit 0 is cleared (unlike timer interrupt requests which remain latched even

when ICP bit 1 is cleared). External interrupts are handled in the same fashion when the Timer is in the Pulse Width Measurement Mode or in the Event Counter Mode, except that only in the Pulse Width Measurement Mode the external interrupt request latch is set on the trailing edge of EXT INT, that is, on the transition from the active level to the inactive level.

Interrupt Handling

When either a timer or an external interrupt request is communicated to the CPU section of the 3872, it will be acknowledged and processed at the completion of the first non-privileged instruction if the Interrupt Control Bit of the Status Register is set. If the Interrupt Control Bit is not set, the interrupt request will continue until either the Interrupt Control Bit is set and the CPU section acknowledges the interrupt or until the interrupt request is cleared as previously described.

If there is both a timer interrupt request and an external interrupt request when the CPU section starts to process the requests, the timer interrupt is handled first.

When an interrupt is allowed the CPU section will request that the interrupting element pass its interrupt vector address to the Program Counter via the data bus. The vector address for a timer interrupt is H '020'. The vector address for external interrupts is H '0A0'. After the vector address is passed to the Program Counter, the CPU section sends an acknowledge signal to the appropriate interrupt request latch which clears that latch. The execution of the interrupt service routine will then commence. The return address of the original program is automatically saved in the Stack Register, P.

The Interrupt Control Bit of W (Status Register) is automatically reset when an interrupt request is acknowledged. It is then the programmer's responsibility to determine when ICB will again be set (by executing an EI instruction). This action prevents an interrupt service routine from being interrupted unless the programmer so desires.

External Reset

When $\overline{\text{RESET}}$ is taken low the content of the Program Counter is pushed to the Stack Register and then the Program Counter and the ICB bit of the W Status Register are cleared. The original Stack Register content is lost. Ports 4, 5, 6 and 7 are loaded with H '00'. The contents of all other registers and ports are unchanged. When power is first applied all ports and registers are undefined until a reset is performed. When RESET is taken high the first program instruction is fetched from ROM location H '000'.

When an external reset of the 3872 occurs, P0 is pushed into P and the old contents of P are lost. It must be noted that an external reset is recognized at the start of a machine cycle and not necessarily at the end of an instruction. Thus if the 3872 is executing a multi-cycle instruction, that instruction is not completed and the contents of P upon reset may not necessarily be the address of the instruction that would have been executed next. It may, for example, point to an immediate operand if the reset occurred during the second cycle of a LI or CI instruction. Additionally, several instructions (JMP, PI, PK, LR P0, Q) as well as the interrupt acknowledge sequence modify P0 in parts. That is, they alter P0 by first loading one part then the other and the entire operation takes more than one cycle. Should reset occur during this modification process the value pushed into P will be part of the old P0 (the as yet unmodified part) and part of the new P0 (already modified part). Thus care should be taken (perhaps by external gating) to insure that reset does not occur at an undesirable time if any significance is to be given to the contents of P after a reset occurs.

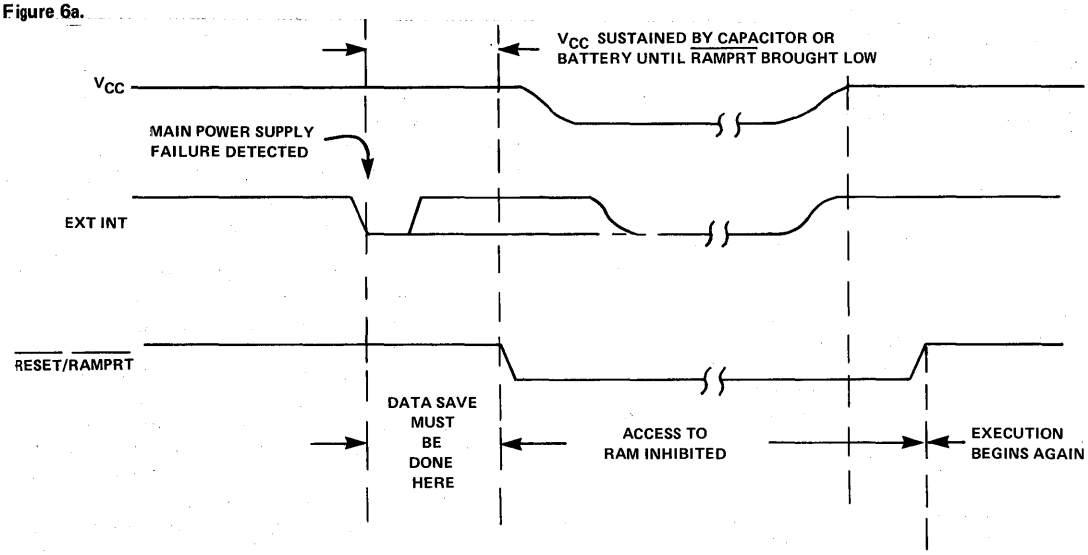
Test Logic

Special test logic is implemented to allow access to the internal main data bus for test purposes.

In normal operation the TEST pin is unconnected or is connected to GND. When TEST is placed at a TTL level (2.0V to 2.6V) Port 4 becomes an output of the internal data bus and Port 5 becomes a wired-OR input to the internal data bus. The data appearing on the Port 4 pins is logically true whereas input data forced on Port 5 must be logically false. When TEST is placed at high level (6.0V to 7.0V), the ports act as above and additionally the 2K x 8 program ROM is prevented from driving the data bus. In this mode operands and instructions may be forced externally through Port 5 instead of being accessed from the program ROM. When TEST is in either the TTL state or the high state, $\overline{\text{STROBE}}$ ceases its normal function and becomes a machine cycle clock (identical to the F8 multi-chip system WRITE clock except inverted).

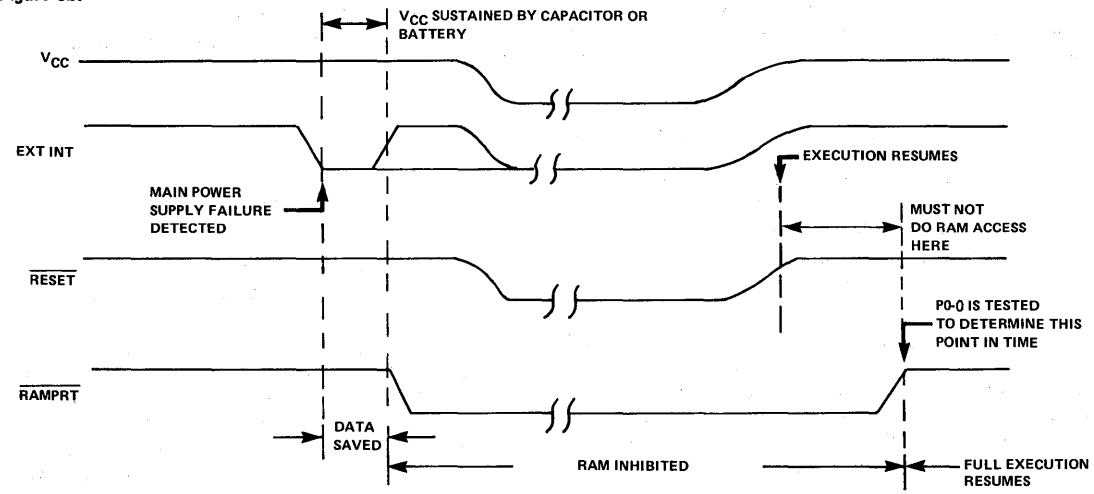
RESET TIED TO RAMPRT, $V_{SB} \geq 2.2$ VOLTS

Figure 6a.



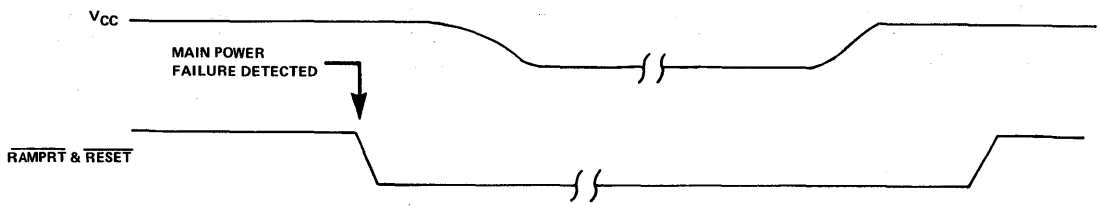
RAMPRT INDEPENDENT OF RESET, $V_{SB} \geq 2.2$ VOLTS

Figure 6b.



NO SAVE ROUTINE REQUIRED, $V_{SB} \geq 2.2$ VOLTS

Figure 6c.



Timing complexities render the capabilities associated with the TEST pin impractical for use in a user's application, but these capabilities are thoroughly sufficient to enable a rapid method for thoroughly testing the 3872.

STANDBY POWER OPTION

If the standby power option has not been selected Port 0-bit 0 and 1 are readable and writeable and the RAM Protect ($\overline{\text{RAMPRT}}$) signal is not operational. If the power down option is selected, Port 0 -bit 0 and bit 1 are readable only. The standby power source (V_{SB}) is connected to Pin 4 and $\overline{\text{RAMPRT}}$ control to Pin 3. It is recommended that Nickel-Cadinium batteries (typical voltage of series cells = 2.5V) be used for standby power, since the MK3872 can automatically trickle charge the two Ni-Cad's. If more than two cells in series are used, the charging circuit must be provided outside the MK3872. Whenever $\overline{\text{RAMPRT}}$ is brought low, the standby RAM (64x8 bit words in PO/DC address space, 4032 to 4095_{10} or FC0 to FFF_{16}) is disabled from being read or written. Also the RAM itself is switched from V_{CC} power to the V_{SB} power. Three modes of powering down are recommended. In the first mode, $\overline{\text{RESET}}$ and the $\overline{\text{RAMPRT}}$ pins are tied together. If data is to be saved in RAM, the processor must be interrupted early enough to save all necessary data before the V_{CC} falls below the minimum level. After the save is done, the $\overline{\text{RESET}}$ and $\overline{\text{RAMPRT}}$ can fall. This prevents any further access of the RAM; V_{CC} may now fall. As the power comes up, the $\overline{\text{RESET}}$/$\overline{\text{RAMPRT}}$ signal should be held low until V_{CC} is above the minimum level.

In the second mode of operation, the $\overline{\text{RESET}}$ pin is not tied to $\overline{\text{RAMPRT}}$. When these pins are brought high, the 3872 will begin execution at location 000. On power up a normal execution may begin but the program must monitor the Port 0-0 pin (Pin 3) and wait until the Port 0-0 $\overline{\text{RAMPRT}}$ pin is high before attempting any access of the RAM. With this approach, the RAM is not switched to standby power each time the $\overline{\text{RESET}}$ goes low.

If a special save data routine is not needed then the $\overline{\text{EXT INTERRUPT}}$ need not be used and the only requirement to save the RAM data is that $\overline{\text{RAMPRT}}$ be low before V_{CC} drops below 4.5V. For example if a few key variables are to be stored in RAM and it is desired that these be saved during a loss of power, two copies of each variable are kept with an associated flag, thus no interrupt and save routine is necessary. The method of updating a variable is as follows:

- Clear Flag Word 1
- Update Variable (Copy 1)
- Set Flag Word 1
- Clear Flag Word 2
- Update Variable (Copy 2)
- Set Flag Word 2

Now execution may terminate at any time, even during the update of a variable or flag word, causing that byte in RAM to be bad data. There is always a good data byte which contains either the most recent or next most recent value of the variable. Any copy of the variable where the flag word is "set" is a good data byte. While this method significantly encumbers the data storage process, it eliminates the need for a power fail interrupt which both reduces external circuitry and leaves the external interrupt pin completely free for other use.

3872 Clocks

The time base for the 3872 may originate from one of five sources. There are four external modes and one internal mode.

If both XTL 1 and XTL 2 are grounded, the 3872 will activate its internal oscillator.

The four external configurations are shown in Figure 7. There is an internal 20pF capacitor between XTL 1 and GND and an internal 20pF capacitor between XTL 2 and GND. Thus external capacitors are not neccessarily required. In all external clock modes the external time base frequently is divided by two to form the internal Φ clock.

Crystal Selection

The use of a crystal as the time base is highly recommended as the frequency stability and reproducability from system to system is unsurpassed. The 3872 has an internal divide by two to allow the user of inexpensive and widely available TV Color Burst Cyrstals (3.58MHz). The following crystal parameters and vendors are suggested for 3872 applications:

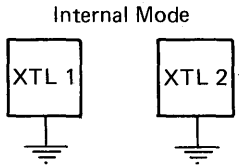
Parameters

a) Parallel Resonance, Fundamental Mode AT-Cut, HC-33/μ holder
b) Frequency Tolerance measured with 18pF load (0.1% accuracy). Drive level 10mW.
c) Shunt Capacitance (Co) = 7pF max.
d) Series Resistance (Rs)

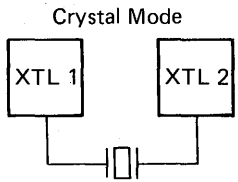
f = 1MHz	Rs	=	550 ohms	max.
f = 2MHz	Rs	=	300 ohms	max.
f = 3MHz	Rs	=	100 ohms	max.
f = 3.58MHz	Rs	=	100 ohms	max.
f = 4MHz	Rs	=	100 ohms	max.

CLOCK CONFIGURATIONS

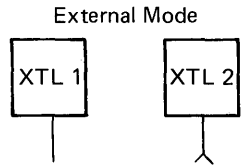
Figure 7

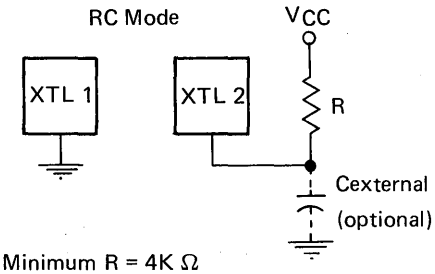


f = 1.7 –4 MHz



AT Cut 1 – 4 MHz



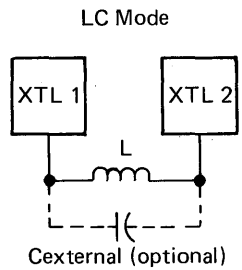


Minimum R = 4K Ω

C = 20.5 pF ± 2.5pF + Cexternal

$$f_{MIN} \cong \frac{1}{1.1\ RC + 65\ ns}$$

$$f_{MAX} \cong \frac{1}{1.0\ RC + 15\ ns}$$



Minimum L = 0.1 mH
Minimum Q = 40

Maximum Cexternal = 30pF

C = 10pF ± 1.3pF + Cexternal

$$f \cong \frac{1}{2\pi\sqrt{LC}}$$

Suggested Crystal Vendors

a) Electro-Dynamics
5625 Foxridge Drive
Mission, Kansas 66201
913-262-2500

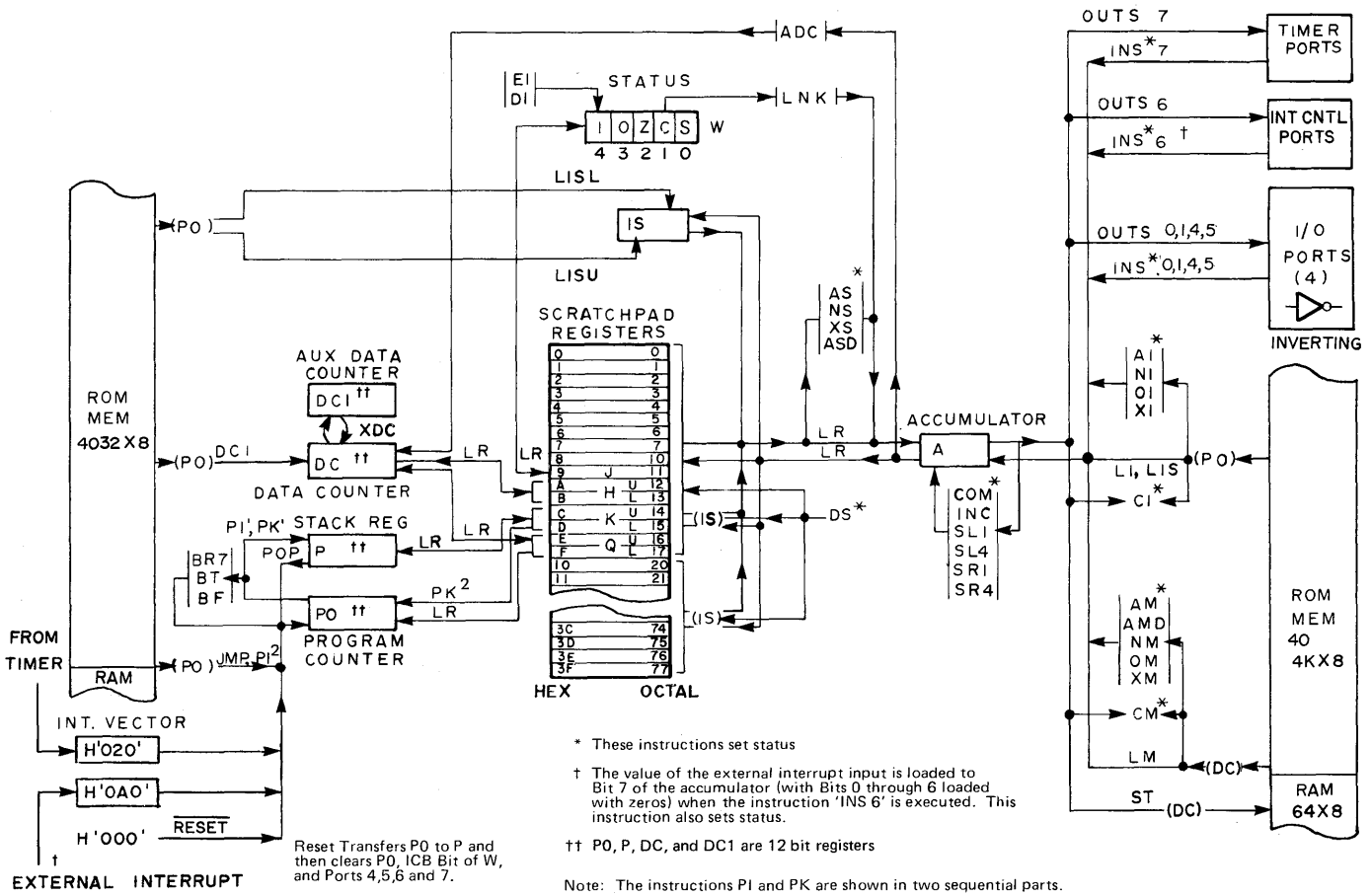
b) CRYSTEK
1000 Crystal Drive
Ft. Myers, Florida 33901
813-936-2109

c) W.T. Liggett Corp.
1500 Worcester Rd.
Section 30
Framingham, MA 01701
617-620-1150

d) Erie Frequency Control
453 Lincoln Street
Carlisle, Penn 17013
717-249-2232

e) Electronic Crystals Corp.
1153 Southwest Blvd.
Kansas City, Kansas 66103
913-262-1274

f) M-TRON Industries
P.O. Box 630
100 Douglas Avenue
Yankton, South Dakota
605-665-9321



* These instructions set status

† The value of the external interrupt input is loaded to Bit 7 of the accumulator (with Bits 0 through 6 loaded with zeros) when the instruction 'INS 6' is executed. This instruction also sets status.

†† PO, P, DC, and DC1 are 12 bit registers

Reset Transfers P0 to P and then clears P0, ICB Bit of W, and Ports 4,5,6 and 7.

Note: The instructions PI and PK are shown in two sequential parts. (PI1, PI2 and PK1, PK2).

MK3872 PROGRAMMING MODEL
Figure 8

INSTRUCTION EXECUTION

This section details the timing and execution of the 3872 instruction set. The 3872 executes the entire F8 instruction set with exact F8 timing.

F8 INSTRUCTION SET

ACCUMULATOR GROUP INSTRUCTIONS

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	CYCLES LONG	μS (2MHzΦ)	STATUS BITS OVR	ZERO	CRY	SIGN
Add Carry	LNK		A←(A) + CRY	19	1	1		2	1/0	1/0	1/0	1/0
Add Immediate	AI	ii	A←(A) + H 'ii'	24ii	2	1	1	5	1/0	1/0	1/0	1/0
And Immediate	NI	ii	A←(A)∧H 'ii'	21ii	2	1	1	5	0	1/0	0	1/0
Clear	CLR		A←H'00'	70	1	1		2	—	—	—	—
Compare Immediate	CI	ii	H'ii'+ (A) + 1	25ii	2	1	1	5	1/0	1/0	1/0	1/0
Complement	COM		A←(A) + H'FF'	18	1	1		2	0	1/0	0	1/0
Exclusive or Immediate	XI	ii	A←(A) + H'ii'	23ii	2	1	1	5	0	1/0	0	1/0
Increment	INC		A←(A) + 1	1F	1	1		2	1/0	1/0	1/0	1/0
Load Immediate	LI	ii	A←H'ii'	20ii	2	1	1	5	—	—	—	—
Load Immediate Short	LIS	i	A←H' 0i'	7i	1	1		2				
OR Immediate	OI	ii	A←(A) V H 'ii'	22ii	2	1	1	5	0	1/0	0	1/0
Shift Left One	SL	1	Shift Left 1	13	1	1		2	0	1/0	0	1/0
Shift Left Four	SL	4	Shift Left 4	15	1	1		2	0	1/0	0	1/0
Shift Right One	SR	1	Shift Right 1	12	1	1		2	0	1/0	0	1
Shift Right Four	SR	4	Shift Right 4	14	1	1		2	0	1/0	0	1

BRANCH INSTRUCTIONS In all conditional branches P0←(P0) + 2 if the test condition is not met. Execution is complete in 3 short cycles.

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	CYCLES LONG	μS (2MHzΦ)	STATUS BITS OVR	ZERO	CRY	SIGN
Branch on Carry	BC	aa	P0←(P0) +1+ H'aa' if CRY = 1	82aa	2	2	1	7	—	—	—	—
Branch on Positive	BP	aa	P0←(P0) + 1 + H'aa ' if SIGN = 1	81aa	2	2	1	7	—	—	—	—
Branch on Zero	BZ	aa	P0←(P0) + 1 + H'aa' if Zero = 1	84aa	2	2	1	7	—	—	—	—
Branch on True	BT	taa	P0←(P0) + 1 + H'aa ' if any test is true	8taa	2	2	1	7	—	—	—	—
Branch If Negative	8M	aa	P0←(P0)+1+ H'aa ' if SIGN =0	91aa	2	2	1	7	—	—	—	—
Branch if No Carry	BNC	aa	P0←(P0) +1+ H'aa ' if CARRY = 0	92aa	2	2	1	7	—	—	—	—
Branch if No Overflow	BNO	aa	P0←(P0) +1+H'aa ' if OVR = 0	98aa	2	2	1	7	—	—	—	—
Branch if Not Zero	BNZ	aa	P0←(P0)+1+ H'aa ' if ZERO = 0	94aa	2	2	1	7	—	—	—	—
Branch if False Test	BF	taa	P0←(P0) +1+ H'aa ' if all false test bits	9taa	2	2	1	7	—	—	—	—
Branch if ISAR (Lower) ≠7	BR7	aa	P0←(P0)+1+ H'aa ' if ISARL ≠7	87aa	2	2	1	5	—	—	—	—
			P0←(P0)+2 if ISARL =		2	2		4	—	—	—	—
Branch Relative	BR	aa	P0←(P0)+1+ H'aa '	90aa	2	2	1	7	—	—	—	—
Jump*	JMP	aaaa	P0←H'aaaa'	29aaaa	3	1	3	11	—	—	—	—

t=TEST CONDITION

2^2	2^1	2^0
ZERO	CRY	SIGN

t=TEST CONDITION

2^3	2^2	2^1	2^0
OVF	ZERO	CRY	SIGN

*Privileged instruction, Accumulator contents altered during execution JMP instruction.

MEMORY REFERENCE INSTRUCTIONS In all Memory Reference Instructions, the Data Counter is incremented DC ←(DC)+1

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	CYCLES LONG	μS (2MHzΦ)	OVR	STATUS BITS ZERO	STATUS BITS CRY	STATUS BITS SIGN
Add Binary	AM		A←(A) + [(DC)]	88	1	1	1	5	1/0	1/0	1/0	1/0
Add Decimal	AMD		A←(A) + [(DC)]* BCD Adjust	89	1	1	1	5	1/0	1/0	1/0	1/0
AND	NM		A←(A) ∧ [(DC)]	8A	1	1	1	5	0	1/0	0	1/0
Compare	CM		[(DC)] + $\overline{(A)}$ + 1	8D	1	1	1	5	1/0	1/0	1/0	1/0
Exclusive OR	XM		A←(A)⊕[(DC)]	8C	1	1	1	5	0	1/0	0	1/0
Load	LM		A←[(DC)]	16	1	1	1	5	—	—	—	—
Logical OR	OM		A←(A) ∨ [(DC)]	8B	1	1	1	5	0	1/0	0	1/0
Store	ST		A→[(DC)]	17	1	1	1	5	—	—	—	—

ADDRESS REGISTER GROUP INSTRUCTIONS

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	CYCLES LONG	μS (2MHzΦ)	OVR	STATUS BITS ZERO	STATUS BITS CRY	STATUS BITS SIGN
Add to Data Counter	ADC		DC←(DC) + (A)	8E	1	1	1	5	—	—	—	—
Call to Subroutine*	PK		P0U←(r12); P0L←(r13), P←(P0)	0C	1	1	2	8	—	—	—	—
Call to Subroutine Immediate*	PI	aaaa	P←(P0), P0←H'aaaa	28aaaa	3	2	3	13	—	—	—	—
Exchange DC	XDC		(DC)⇄(DC1)	2C	1	2		4	—	—	—	—
Load Data Counter	LR	DC,Q	DCU←(r14); DCL←(r15)	0F	1	1	2	8	—	—	—	—
Load Data Counter	LR	DC'H	DCU←(r10); DCL←(r11)	10	1	1	2	8	—	—	—	—
Load DC Immediate	DCI	aaaa	DC H'aaaa'	2Aaaaa	3	3	2	12	—	—	—	—
Load Program Counter	LR	P0,Q	P0U←(r14); P0L←(r15)	0D	1	1	2	8	—	—	—	—
Load Stack Register	LR	P,K	PU←(r12); PL←(r13)	09	1	1	2	8	—	—	—	—
Return from Subroutine*	POP		P0←(P)	1C	1	2		4	—	—	—	—
Store Data Counter	LR	Q,DC	r14←(DCU); r15←(DCL)	0E	1	1	2	8	—	—	—	—
Store Data Counter	LR	H,DC	r10←(DCU); r11←(DCL)	11	1	1	2	8	—	—	—	—
Store Stack Register	LR	K,P	r12←(PU); r13←(PL)	08	1	1	2	8	—	—	—	—

SCRATCHPAD REGISTER INSTRUCTIONS (Refer to Scratchpad Addressing Modes)

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	CYCLES LONG	μS (2MHzΦ)	OVR	STATUS BITS ZERO	STATUS BITS CRY	STATUS BITS SIGN
Add Binary	AS	r	A←(A)+ (r)	Cr	1	1		2	1/0	1/0	1/0	1/0
Add Decimal	ASD	r	A←(A) + (r)	Dr	1	2		4	1/0	1/0	1/0	1/0
Decrement	DS	r	r←(r) + H'FF'	3r	1			3	1/0	1/0	1/0	1/0
Load	LR	A,r	A←(r)	4r	1	1		2	—	—	—	—
Load	LR	A, KU	A←(r12)	00	1	1		2	—	—	—	—
Load	LR	A, KL	A←(r13)	01	1	1		2	—	—	—	—
Load	LR	A, QU	A←(r14)	02	1	1		2	—	—	—	—
Load	LR	A, QL	A←(r15)	03	1	1		2	—	—	—	—
Load	LR	r, A	r←(A)	5r	1	1		2	—	—	—	—
Load	LR	KU, A	r12←(A)	04	1	1		2	—	—	—	—
Load	LR	KL, A	r13←(A)	05	1	1		2	—	—	—	—
Load	LR	QU, A	r14←(A)	06	1	1		2	—	—	—	—
Load	LR	QL,A	r15←(A)	07	1	1		2	—	—	—	—
And	NS	r	A←(A) ∧ (r)	Fr	1	1		2	0	1/0	0	1/0
Exclusive Or	XS	r	A←(A) + (r)	Er	1	1		2	0	1/0	0	1/0

*Privileged instruction, Accumulator contents altered during execution of PI instruction.

MISCELLANEOUS INSTRUCTIONS

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	CYCLES LONG	μS (2MHzΦ)	STATUS BITS OVR	ZERO	CRY	SIGN
Disable Interrupt	DI		RESET ICB	1A	1	1		2	–	–	–	–
Enable Interrupt *	EI		SET ICB	1B	1	1		2	–	–	–	–
Input	IN	04,05,06,07	A←(Input Port aa)	26aa	2	1	2	8	0	1/0	0	1/0
Input Short	INS	0, 1	A←(Input Port 0 or 1)	A0,A1	1	2		4	0	1/0	0	1/0
Input Short	INS	4,5,6,7	A←(Input Port a)	Aa	1	1	2	8	0	1/0	0	1/0
Load ISAR	LR	IS,A	IS←(A)	0B	1	1		2	–	–	–	–
Load ISAR Lower	LISL	bbb	ISL←bbb	6(0bbb)**	1	1		2	–	–	–	–
Load ISAR Upper	LISU	bbb	ISU←bbb	6(1bbb)**	1	1		2	–	–	–	–
Load Status Register *	LR	W,J	W←(r9)	1D	1	2		4	1/0	1/0	1/0	1/0
No Operation	NOP		P0←(P0) + 1	2B	1	1		2	–	–	–	–
Output *	OUT	04,05,06,07	Output Port aa←(A)	27aa	2	1	2	8	–	–	–	–
Output Short	OUTS	0, 1	Output Port 0 or 1←(A)	B0, B1	1	2		4	–	–	–	–
Output Short*	OUTS	4,5,6,7	Output Port a←(A)	Ba	1	1	2	8	–	–	–	–
Store ISAR	LR	A,IS	A←(IS)	0A	1	1		2	–	–	–	–
Store Status Reg	LR	J,W	r9←(W)	1E	1	1		2	–	–	–	–

*Privileged instruction

**b = 1 bit immediate operand

NOTES.

Lower case denotes variables specified by programmer

Function Definitions

←	is replaced by
(_)	the contents of
( )	Binary "1's" complement of
+	Arithmetic Add (Binary or Decimal)
⊕	Logical "OR" exclusive
∧	Logical "AND"
∨	Logical "OR" inclusive
H' '	Hexadecimal digit
[( )]	Contents of memory specified by ( )
a	Address Variable (four bits)
A	Accumulator
b	One bit immediate operand
DC	Data Counter (Indirect Address Register)
DC1	Data Counter 1 (Auxiliary Data Counter)
DCL	Least significant 8 bits of Data Counter Addressed
DCU	Most significant 8 bits of Data Counter Addressed
H	Scratchpad Register 10 and 11
i	Immediate operand (four bits)
ICB	Interrupt Control Bit
IS	Indirect Scratchpad Address Register
ISL	Least Significant 3 bits of ISAR
ISU	Most Significant 3 bits of ISAR
J	Scratchpad Register 9
K	Registers 12 and 13
KL	Register 13
KU	Register 12
P0	Program Counter
P0L	Least Significant 8 bits of Program Counter
P0U	Most Significant 8 bits of Program Counter
P	Stack Register
PL	Least Significant 8 bits of Program Counter
PU	Most Significant 8 bits of Active Stack Register
Q	Registers 14 and 15
QL	Register 15
QU	Register 14
r	Scratchpad Register (any address 0 thru B) (See Below)
W	Status Register

Scratchpad Addressing Modes Using IS. ($r \neq 0$ thru B)

r=H'C'	Register Addressed by IS is (Unmodified)
r=H'D'	Register Addressed by IS is Incremented
r=H'E'	Register Addressed by IS is Decremented
r=H'F'	Illegal OP Code.

Status Register

—	No change in condition
1/0	is set to "1" or"0" depending on conditions
CRY	Carry Flag
OVR	Overflow Flag
SIGN	Sign of Result Flag
ZERO	Zero Flag

ELECTRICAL SPECIFICATIONS

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias . . . . . 0°C to 70°C
Storage Temperature . . . . . −65°C to +150°C
Voltage on Any Pin With Respect To Grouns (except open drain pins) . . . . . −1.0V to +7V
Voltage On Open Drain Pins . . . . . −1.0V to +13.5V
Power Dissipation 1.5W

A.C. CHARACTERISTICS — See Figure 9 and 10 for Timing Diagrams

T_A = 0°C to 70°C, V_{CC} = 5V ±10%, I/O POWER DISSIPATION ≤100mW

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	NOTES
	t_O(INT)	Time Base Period, internal oscillator	250	1000	ns	4MHz - 1.0MHz
XTL 1 XTL 2	t_O(EX)	Time base period, all external modes	250	1000	ns	4MHz-1MHz
	$t_{EX(H)}$	External Clock Pulse Width High	90	700	ns	
	$t_{EX(L)}$	External Clock Pulse Width Low	100	700	ns	
Φ	$t_{cΦ}$	Internal Φ Clock Period	$2t_O$			
WRITE	tw	Internal WRITE Clock Period	$4t_Φ$ $6t_Φ$			Short Cycle Long Cycle
I/O	$t_{dI/O}$	Output delay from internal WRITE Clock	0	1000	ns	50pF plus one TTL load
	$t_{sI/O}$	Input Setup time to WRITE Clock	1000		ns	
STROBE	$t_{I/O-s}$	Output valid to STROBE Delay	$3tΦ$ -1000	$3tΦ$ +250		I/O load = 50pF + 1 TTL STROBE Load= 50pF + 3 TTL
	tsl	STROBE Low Time	$8tΦ$ -250	$12tΦ$ +250	ns	
RESET	t_{RH}	RESET Hold Time, Low	$6tΦ$ +750		ns	
EXT INT	t_{EH}	EXT INT Hold Time, Active and Inactive State	$6tΦ$ + 750		ns	To trigger interrupt
			$2tΦ$			To trigger timer

CAPACITANCE

T_A = 25°C, f=2MHz

SYMBOL	PARAMETER	MIN	MAX	UNIT	NOTES
C_{IN}	Input Capacitance: I/O Ports, $\overline{RESET}$, $\overline{EXTINT}$, $\overline{RAMPRT}$, TEST		7	pF	Unmeasured Pins Grounded
C_{XTL}	Input Capacitance: XTL1, XTL2	18	23	pF	

DC CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = +5V ± 10%, I/O POWER DISSIPATION ⩽100mW

SYMBOL	PARAMETER	MIN	MAX	UNIT	TEST CONDITIONS
I_{CC}	Power Supply Current		93	mA	Outputs Open
P_D	Power Dissipation		440	mW	Outputs Open
V_{IHEX}	External Clock Input High Level	2.4	5.8	V	
V_{ILHEX}	External Clock Input Low Current	−0.3	0.6	V	
I_{IHEX}	External Clock Input High Current		100	μA	$V_{IHEX} = V_{CC}$
I_{ILEX}	External Clock Input Low Current		-100	μA	$V_{ILEX} = V_{SS}$
V_{IH}	Input High Level Ports, $\overline{RESET}$[1], EXT INT[1]	2.0	5.8	V	
V_{IHOD}	Open Drain Input High Level	2.0	13.2	V	
V_{IL}	Input Low Level Ports, $\overline{RESET}$[1], EXT INT[1]	−0.3	0.8		
I_{IL}	Input Low Current Ports, $\overline{RESET}$[2], EXT INT[2]	-1.6		mA	V_{IL}=0.4V
I_L	Leakage Current Open drain ports, $\overline{RAMPRT}$ $\overline{RESET}$[3], EXT INT[3]		+10 −5	μA	V_{IN}=13.2V V_{IN}=0.0V
I_{OH}	Output High Current Standard ports, $\overline{RESET}$[2] EXT INT[2]	−100 −30		μA μA	V_{OH}=2.4V V_{OH}=3.9V
I_{OHDD}	OUTPUT High Current Direct Drive Ports	−0.1 −1.5 	 −8.5	mA mA mA	V_{OH} = 2.4V V_{OH}=1.5V V_{OH}=.7V
I_{OL}	Output Low Current IO ports	1.8		mA	V_{OL}=0.4V
I_{OHS}	$\overline{STROBE}$ Output High Current	−300		μA	V_{OH}=2.4V
I_{OLS}	$\overline{STROBE}$ Output Low Current	5.0		mA	V_{OL} = 0.4V
V_{IHRPR}	$\overline{RAMPRT}$ Input High Level	1.9	5.8	V	Guaranteed .1V less than V_{IH} for $\overline{RESET}$
V_{ILRPR}	$\overline{RAMPRT}$ Input Low Level	−0.3	0.4	V	Guaranteed .1V less than V_{IL} for $\overline{RESET}$

DC CHARACTERISTICS (Cont'd)

SYMBOL	PARAMETER	MIN	MAX	UNIT	NOTES
V_{SB}	Standby V_{CC} for RAM	2.2	5.5	V	
I_{SB}	Standby current		6 3.7	mA mA	V_{SB}= 5.5 V_{SB}=2.2
I_{CHARGE}	Trickle charge available on V_{SB} with V_{CC}=4.5 to 5.5	-4 -4.5	-12 -15	mA mA	V_{SB}=2.8V V_{SB}=2.2V
P_{DIO}	Power dissipated by I/O Pins[4]		600 60	mW mW	All Pins any one pin

* Stresses above 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 condition 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.

1. $\overline{\text{RESET}}$ and EXT INT have internal Schmit triggers giving minimum .2V hysteresis.
2. $\overline{\text{RESET}}$ or EXT INT programmed with standard pull-up
3. $\overline{\text{RESET}}$ or EXT INT programmed without standard pull-up
4. Power dissipation for I/O pins is calculated by $\Sigma(V_{cc} - V_{IL})(|I_{IL}|) + \Sigma(V_{CC} - V_{OH})(|I_{OH}|) + \Sigma(V_{OL})(I_{OL})$

TIMER AC CHARACTERISTICS

Definitions:

Error = Indicated time value - actual time value

tpsc = tΦ x Prescale Value

Interval Timer Mode:

Single interval error, free running (Note 3) ..... ±6tΦ
Cumulative interval error, free running (Note 3) 0
Error between two Timer reads (Note 2) ..... ±(tpsc + tΦ)
Start Timer to stop Timer error (Notes 1,4) ..... +tΦ to −(tpsc +tΦ)
Start Timer to read Timer error (Notes 1,2) ..... −5tΦ to −(tpsc + 7tΦ)
Start Timer to interrupt request error (Notes 1,3) ..... −2tΦ to −8tΦ
Load Timer to stop Timer error (Note 1) ..... +tΦ to −(tpsc + 2tΦ)
Load Timer to read Timer error (Notes 1,2) ..... −5tΦ to −(tpsc + 8tΦ)
Load Timer to interrupt request error (Notes 1,3) ..... −2tΦ to −9tΦ

Pulse Width Measurement Mode:

Measurement accuracy (Note 4) ..... +tΦ to −(tpsc +2tΦ)
Minimum pulse width of EXT INT pin 2tΦ

Event Counter Mode:

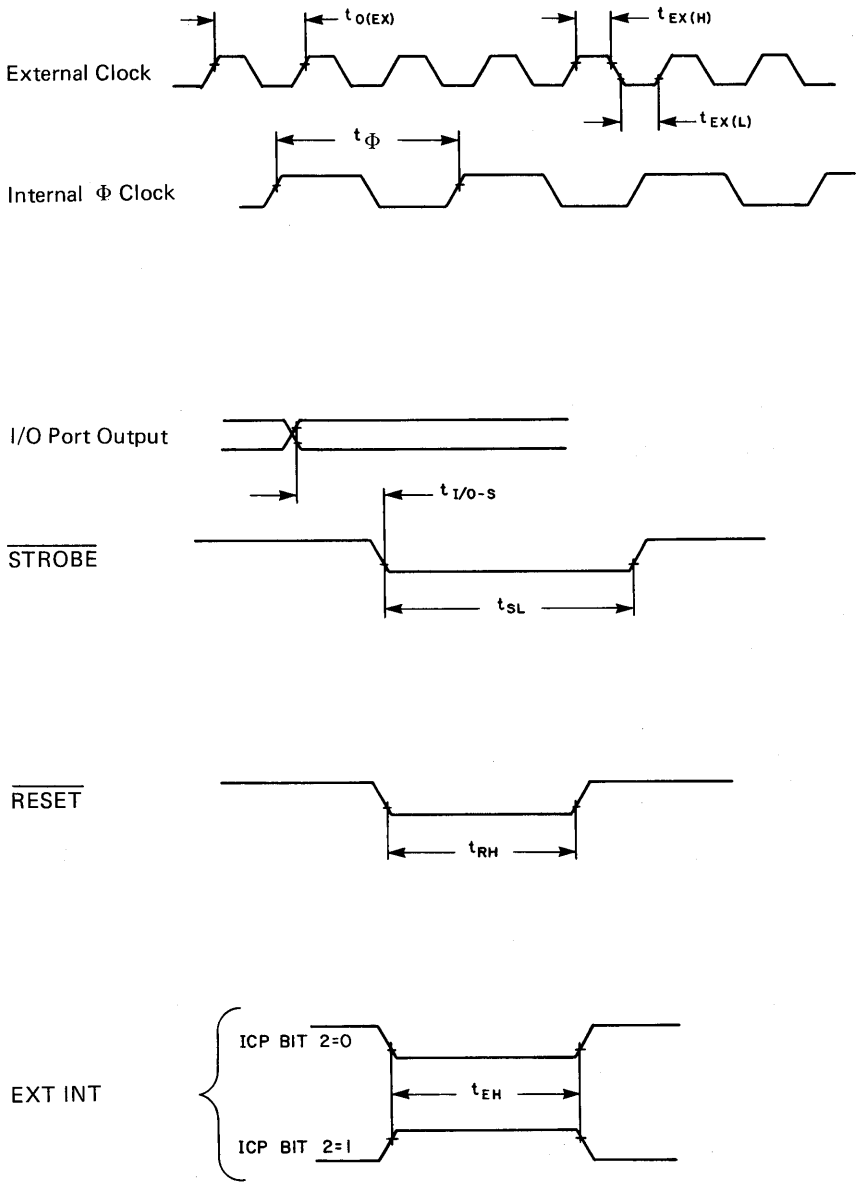
Minimum active time of EXT INT pin 2tΦ
Minimum inactive time of EXT INT pin 2tΦ

Notes:

1. All times which entail loading, starting, or stopping the Timer are referenced from the end of the last machine cycle of the OUT or OUTS instruction.
2. All times which entail reading the Timer are referenced from the end of the last machine cycle of the IN or INS instruction.
3. All times which entail the generation of an interrupt request are referenced from the start of the machine cycle in which the appropriate interrupt request latch is set. Additional time may elapse if the interrupt request occurs during a privileged or multicycle instruction.
4. Error may be cumulative if operation is repetitively performed.

AC TIMING DIAGRAM

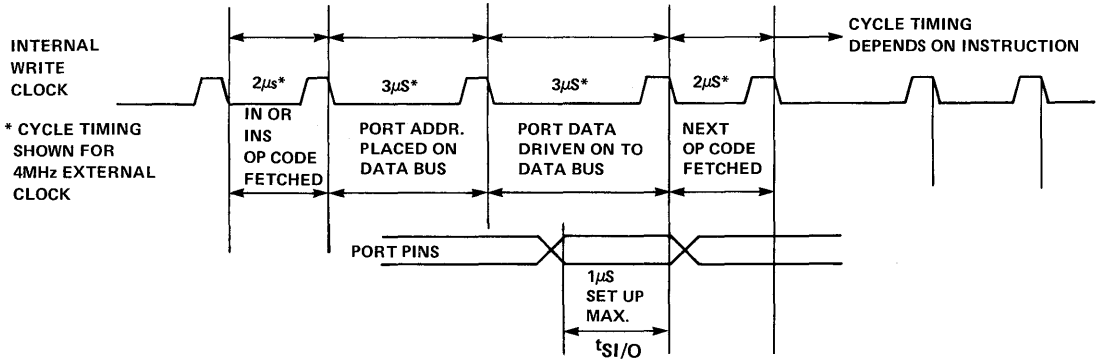
Figure 9



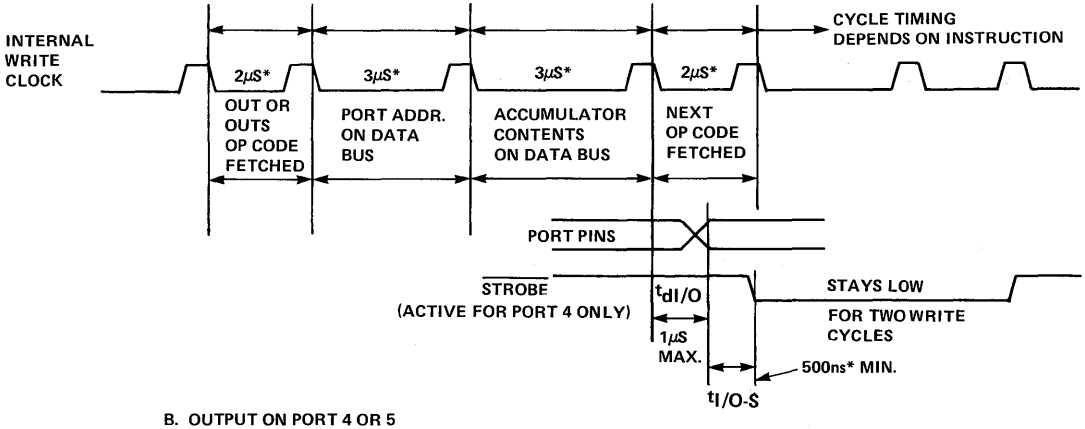
Note: All measurements are referenced to V_{IL} max., V_{IH} min., V_{OL} max., or V_{OH} min.

INPUT/OUTPUT AC TIMING

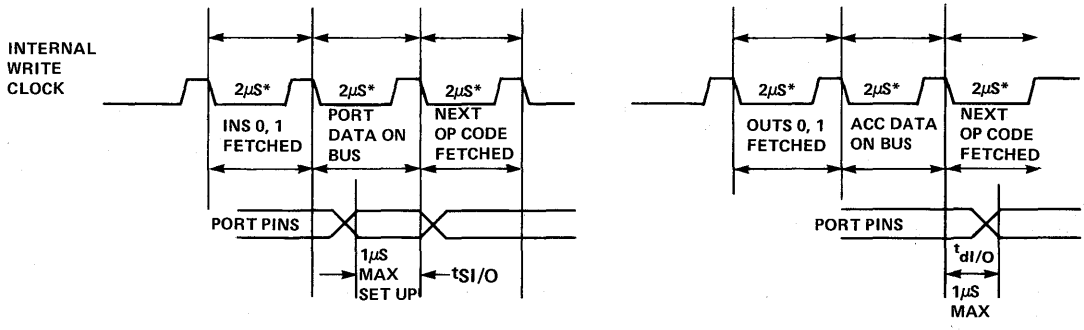
Figure 10



A. INPUT ON PORT 4 OR 5



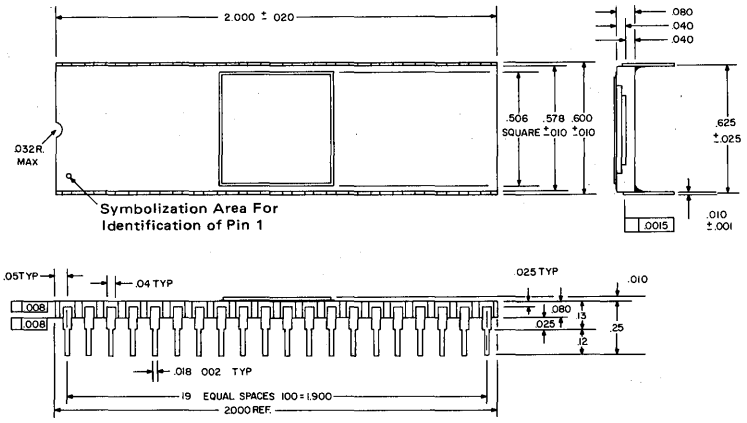
B. OUTPUT ON PORT 4 OR 5



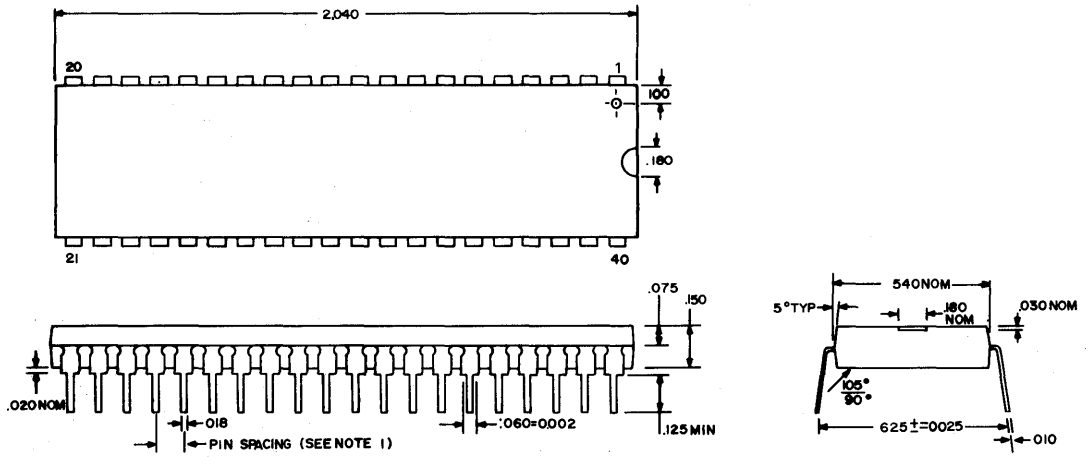
C. INPUT ON PORT 0 OR 1

D. OUTPUT ON PORT 0, 1

PACKAGE DESCRIPTION: 40-Pin Dual In-Line Ceramic Package



PACKAGE DESCRIPTION 40-Pin Dual-in-Line Plastic Package



ORDERING INFORMATION

PART NO.	PACKAGE TYPE	TEMPERATURE RANGE
MK3872(N)/16XXX	Plastic	0°C to +70°C
MK3872(P)/16XXX	Ceramic	0°C to +70°C

APPENDIX A

ORDERING INFORMATION

CUSTOM MK3872 OPTION SPECIFICATIONS

The custom MK3872 program may be transmitted to MOSTEK in any of the following media, listed in order of preference:

1) PROMs from the EMU-72
2) Punched paper tape
3) AID-80F Flexible Disk
4) Silent 700 cassette
5) Card Deck (IBM 80 column cards)

The program may be specified in the following forms:

PROMS with correct object code in each location

OBJECT CODE produced by one of MOSTEK's assemblers:

XFOR-50/70 Fortran IV Cross Assembler, SDB-50/70 resident assembler (ASMB-50/70), AID-80F F8 Cross-Assembler (FZCASM)

OBJECT CODE produced by the dump command from any of MOSTEK's F8 development hardware (SDB-50/70, AID-80F).

DATA DECK FORMAT as described in the Data Deck section

A completed cover letter (See Fig. A-1) must be attached. The information should be properly packed and mailed prepaid and insured to:

MOSTEK Corporation
Microcomputer Product Marketing
1215 West Crosby Road
Carrollton, Texas 75006

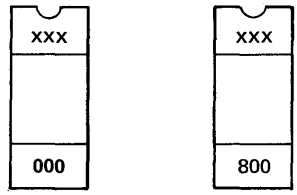
A second copy of the cover letter should be mailed separately to the above address.

PROMS

2716 type PROMs, programmed with the customer program (positive logic sense for addresses and data) may be submitted. The PROMs must be clearly marked to indicate which PROM corresponds to address space 000-7FF and which PROM corresponds to address space 800-FFF. See Fig. A-2 for marking. Include a three-letter customer ID on each PROM. After PROMs are removed from the EMU-72, they must be placed in conductive IC carriers and securely packed.

Figure A-2

XXX = Customer ID



PAPER TAPE

Punched paper tapes (1" wide, 8 level ASCII) will be accepted. The tape must contain the absolute object output from the above mentioned F8 assemblers Paper object tapes in absolute format generated by the "D" (dump) command of DDT-2 or the dump command of the AID-80F (F8 debug option) are also acceptable if the entire memory space is dumped continuously. Tapes may also be punched using the DATA DECK FORMAT. They must contain 80 characters per record with a CR (carriage return) and LF (line feed) separating each record. The tape must be clearly labeled with customer name, and format used. Fan fold tape is preferred. Tape transparency should be limited to 60% transmissivity (40% opaque). Specifically, thin yellow or white tape is error prone on photo-electric readers and must not be used.

FLEXIBLE DISKS

FLEXIBLE DISKS (Floppy Disks) produced on the MOSTEK AID-80F development station may be submitted. The format must be the absolute object output from the assemblers, or an object dump using the memory dump command (F8 Debug Option). The disk must be clearly labeled with the format of the data (object, or object dump) and the customer's name.

CASSETTE TAPES

Cassette tapes produced on a Silent 700 terminal using the MOSTEK SDB-50/70 or AID-80F. Silent 700 drivers are also acceptable. The format must be the absolute object output of the assemblers described above or that produced by the dump command of the SDB-50/70 or AID-80F (F8 Debug Option). The cassette must be clearly labeled with the data format used (object, or object dump) with the customer's name included. The dump must be of full 4096 bytes of memory starting at address zero on one contigous tape.

Figure A-1

DATE _____ CUSTOMER PO NUMBER _____

CUSTOMER NAME _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

COUNTRY _____

PHONE _____ EXTENSION _____

CONTACT MR. MS. _____

CUSTOMER PART # _____

OPTIONS:

EXTERNAL INTERRUPT: Pull-Up _____ No Pull-Up _____

RESET: Pull-Up _____ No Pull-Up _____

STANDBY POWER OPTION: Yes: _____ No _____

PORT OPTIONS:

	STANDARD TTL	OPEN DRAIN	DRIVER PULL-UP
P4-0 - - - - -	_____	_____	_____
P4-1 - - - - -	_____	_____	_____
P4-2 - - - - -	_____	_____	_____
P4-3 - - - - -	_____	_____	_____
P4-4 - - - - -	_____	_____	_____
P4-5 - - - - -	_____	_____	_____
P4-6 - - - - -	_____	_____	_____
P4-7 - - - - -	_____	_____	_____
P5-0 - - - - -	_____	_____	_____
P5-1 - - - - -	_____	_____	_____
P5-2 - - - - -	_____	_____	_____
P5-3 - - - - -	_____	_____	_____
P5-4 - - - - -	_____	_____	_____
P5-5 - - - - -	_____	_____	_____
P5-6 - - - - -	_____	_____	_____
P5-7 - - - - -	_____	_____	_____

PATTERN MEDIA:

PROMS _____ PAPER TAPE (OBJECT) _____

SILENT 700 CASSETTE (OBJECT) _____

Figure A-1 (Cont'd)

PAPER TAPE (DATA DECK) _____

CARD DECK (DATA DECK) _____ DISKETTE (OBJECT) _____

PREFERRED BASE ON VERIFICATION LISTING: HEX _____ DECIMAL _____

THESE ITEMS MAY AFFECT COST

BRANDING REQUIREMENT (If Any, 10 Alpha-numeric digits allowed)

_____

_____

PROTOTYPE QUANTITY (10 pieces normal - higher quantity extra charge)

_____

WAIVE PROTOTYPES (Customer accepts liability for all work in progress)

Yes _____ No _____

_____

Customer Signature

SIGNATURE _____

TITLE _____

PUNCHED CARD DECK

Standard 80 column punched cards must be used. They must be punched in IBM 029 code. The deck must contain two ty] e of cards:

COMMENT CARDS
DATA CARDS

COMMENT CARDS

Comment Cards must have an asterisk (*) in column 1. The remaining 79 columns may be any character. Comment Cards may be placed anywhere throughout the data deck.

DATA CARDS

These cards specify the actual ROM data. All fields are right justified.

COLUMN 1:	C (the letter C)
COLUMN 2-9:	ADDR
COLUMN 10-12:	BYTE
COLUMN 14-16:	DATA 1
COLUMN 17-19:	DATA 2
COLUMN 20-22:	DATA 3
.	
.	
.	
COLUMN 76-78:	DATA 21
COLUMN 77-79:	DATA 22 or SEQUENCE NUMBER

ADDR is the address of the first byte of data (DATA 1) contained on that card. Successive data bytes read from that card will be placed in successively greater address locations. BYTE is the number of data bytes to be read from that card (1 to 22). If sequence numbers are used, the maximum number of bytes per card is 21. The base for ADDR and BYTE may be either decimal or hex but both must be the same. Data may be either in decimal or hex regardless of the base used for ADDR and BYTE. The base for sequence numbers (if they are used) is always decimal. The bases must be consistant throughout the deck. Data cards need not occur in order of increasing or decreasing addresses. Any unspecified address will be filled with zero. Any unpunched field will be read as a zero. If two data cards specify data for the same address, the one encountered second in the deck will override the first.

A portion of an example deck is shown.

```
*  3872 DATA DECK
*  MOSTEK CORP, EXAMPLE APPLICATION
*  ADDR/BYTE ARE IN DECIMAL
*  DATA IS IN HEX
C  0  8  20  FF  OB  54  34  56  71  B6
C  8  8  1B  28  03  F3  4C  25  2E  94
C 16  8  04  29  01  00

*  START OF SUBROUTINE ALPHA

C  1096   4  20  32  7C  53
C  1100   4  52  47  29  06
C  1104   1  07
```

VERIFICATION MEDIA

All original pattern media (PROMs, paper tape, etc.) are filed for contractural purposes and are not returned. Two copies of computer listings printed during the creation of the custom mask pattern are returned. One copy may be kept by the customer. The other copy should be checked thoroughly, signed, and returned to MOSTEK. The signed listing constitutes the contractual agreement for creation of the custom mask. Though the computer listing serves as the actual verification media, MOSTEK will program 2716 PROMs programmed from the data file used to create the custom mask to aid in the verification process. If programmed PROMs are desired, two blank 2716 type PROMs must be provided by the customer.

PRELIMINARY

MOSTEK®

F8 MICROCOMPUTER DEVICES

Single-Chip Microcomputer MK3876

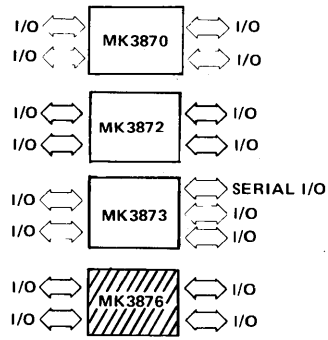
FEATURES

- ☐ Software compatible with F8 family
- ☐ 2048X 8 mask programmable ROM
- ☐ 64 byte scratchpad RAM
- ☐ 64 additional bytes of executable RAM addressable by program counter or data counter
- ☐ Standby option for executable RAM including:
  - —Low standby power, less than 8.2mW
  - —Minimum 2.2V standby supply voltage
  - —No external components required to trickle charge battery
- ☐ 32 bits (4 ports) TTL Compatible I/O
- ☐ Programmable binary timer
  - —Internal timer mode
  - —Pulse width measurement mode
  - —Event counter mode
- ☐ External interrupt
- ☐ Crystal, LC, RC, or external time base
- ☐ Low power (285mW typ.)
- ☐ Single +5 volt ± 10% power supply
- ☐ Same pinout as MK3870

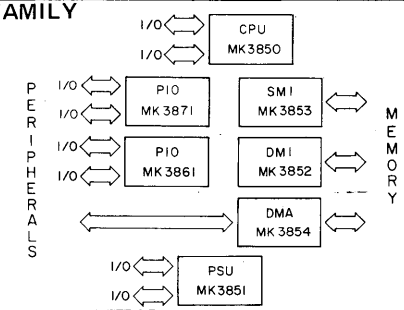
GENERAL DESCRIPTION

The MK3876 is a complete 8-bit microcomputer on a single MOS integrated circuit. The 3876 can execute the F8 instruction set of more than 70 commands, allowing expansion into multi-chip configurations with software compatibility. The device features 2048 bytes of ROM, 64 bytes of scratchpad RAM, 64 bytes of executable RAM, a programmable binary timer, 32 bits of I/O, and a single +5 volt power supply requirement. Utilizing ion-implanted, N-channel silicon gate technology and advanced circuit design techniques the single-chip 3876 offers maximum cost-effectiveness in a wide range of control and logic replacement applications. The 3876 is an expanded memory version of the 3870 single chip microcomputer. The 3876 is identical to the 3870 in the following areas: instruction set, architecture, AC and DC characteristics, and pinout. The only change is in the memory expansion along with the appropriate memory address registers.

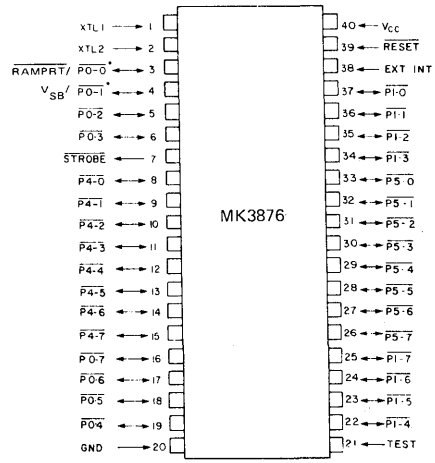
SINGLE CHIP 3870 MICROCOMPUTER FAMILY



F8 FAMILY



PIN CONNECTIONS



*PROGRAMMABLE (PORT PINS BECOME V_{SB} AND $\overline{RAMPRT}$ WITH STANDBY POWER OPTION)

PIN NAME	DESCRIPTION	TYPE
$\overline{P0\text{-}0}$ – $\overline{P0\text{-}7}$	I/O Port 0	Bidirectional
$\overline{P1\text{-}0}$ – $\overline{P1\text{-}7}$	I/O Port 1	Bidirectional
$\overline{P4\text{-}0}$ – $\overline{P4\text{-}7}$	I/O Port 4	Bidirectional
$\overline{P5\text{-}0}$ – $\overline{P5\text{-}7}$	I/O Port 5	Bidirectional
$\overline{STROBE}$	Ready Strobe	Output
EXT INT	External Interrupt	Input
$\overline{RESET}$	External Reset	Input
TEST	Test Line	Input
XTL 1, XTL 2	Time Base	Input
V_{CC}, GND	Power Supply Lines	Input
V_{SB}, $\overline{RAMPRT}$	Standby Power, RAM Protect	Input

FUNCTIONAL PIN DESCRIPTION

$\overline{P0\text{-}0}$–$\overline{P0\text{-}7}$, $\overline{P1\text{-}0}$–$\overline{P1\text{-}7}$, $\overline{P4\text{-}0}$–$\overline{P4\text{-}7}$, and $\overline{P5\text{-}0}$–$\overline{P5\text{-}7}$ are 32 lines which can be individually used as either TTL compatible inputs or as latched outputs.

$\overline{STROBE}$ is a ready strobe associated with I/O Port 4. This pin which is normally high provides a single low pulse after valid data is present on the P4-0–P4-7 pins during an output instruction.

$\overline{RESET}$ may be used to externally reset the 3876. When pulled low the 3876 will reset. When allowed to go high the 3876 will begin program execution at program location H '000'.

EXT INT is the external interrupt input. Its active state is software programmable. This input is also used in conjunction with the timer for pulse width measurement and event counting.

XTL 1 and XTL 2 are the time base inputs to which a crystal (1 to 4MHz), LC network, RC network, or an external single-phase clock may be connected.

TEST is an input, used only in testing the 3876. For normal circuit functionality this pin is left unconnected or may be grounded.

V_{CC} is the power supply input (+5V±10%).

V_{SB} is the RAM standby power supply input if the standby option is selected (+5.5V to +2.2V).

$\overline{RAMPRT}$ is the RAM protect control when the RAM standby option is selected. When brought to a low level (near V_{SS}) the RAM is disabled and therefore protected against any alterations during loss of V_{CC}.

3870 ARCHITECTURE

This section describes the basic functional elements of the 3876 as shown in the block diagram of Figure 1. A programming model is shown in Figure 2.

Main Control Logic

The Instruction Register (IR) receives the operation code (OP code) of the instruction to be executed from the program ROM via the data bus. During all OP code fetches eight bits are latched into the IR. Some instructions are completely specified by the upper 4 bits of the OP code. In those instructions the lower 4 bits are an immediate register address or an immediate 4 bit operand. Once latched into the IR the main control logic decodes the instruction and provides the necessary control gating signals to all circuit elements.

ROM Address Registers

There are four 12 bit registers associated with the 4K x 8 ROM and 64 x 8 RAM. These are the Program Counter (P0), the Stack Register (P), the Data Counter (DC) and the Auxiliary Data Counter (DC1). The Program Counter is used to address instructions or immediate operands. P is used to save the contents of P0 during an interrupt or subroutine call. Thus, P contains the return address at which processing is to resume upon completion of the subroutine or the interrupt routine.

The Data Counter (DC) is used to address data tables. This register is auto-incrementing. Of the two data counters only DC can access the memory. However, the XDC instruction allows DC and DC1 to be exchanged.

Associated with the address registers is a 12 bit Adder/Incrementer. This logic element is used to increment P0 or DC when required and is also used to add displacements to P0 on relative branches or to add the accumulator contents to DC in the ADC (add data counter) instruction.

2048 x 8 ROM

The microcomputer program and data constants are stored in the program ROM. When a ROM access is required, the appropriate address register (P0 or DC) is gated onto the ROM address bus and the ROM output is gated onto the main data bus. The first byte in ROM is location zero.

64 x 8 Executable RAM

The 64 bytes of Executable RAM begins at address 4032 decimal (FC0 hex). As with the ROM memory the RAM memory may be accessed by the P0 and DC address registers. It may be written via the STORE (ST) instruction. It may be read via the LOAD (LM) instruction. Additionally instructions may be executed from the RAM. A mask programmable standby power option is available whereby the 64x8 RAM remains powered and protected so that its contents are saved during a loss of the normal circuit power supply.

Scratchpad and IS

The scratchpad provides 64 8-bit registers which may be used as general purpose RAM memory. The Indirect Scratchpad Address Register (IS) is a 6 bit register used to address the 64 registers. All 64 registers may be accessed using IS. In addition the lower order 12 registers may also be directly addressed.

IS can be visualized as holding two octal digits. This division of IS is important since a number of instructions increment or decrement only the least significant 3 bits of IS when referencing scratchpad bytes via IS. This makes it easy to reference a buffer consisting of contiguous scratchpad bytes. For example, When the low order octal digit is incremented or decremented IS is incremented from octal 27 (0 '27') to 0 '20' or is decremented from 0 '20' to 0 '27'. This feature of the IS is very useful in many program sequences. All six bits of IS may be loaded at one time or either half may be loaded independently.

Scratchpad registers 9 through 15 (decimal) are given mnemonic names (J, H, K, and Q) because of special linkages between these registers and other registers such as the Stack Register. These special linkages facilitate the implementation of multi-level interrupts and subroutine nesting. For example, the instruction LR K,P stores the lower eight bits of the Stack Register into register 13 (K lower or KL) and stores the upper three bits of P into register 12 (K upper or KU) The scratchpad is not protected with the standby power option.

Arithmetic and Logic Unit (ALU)

After receiving commands from the main control

MK3876 BLOCK DIAGRAM
Figure 1

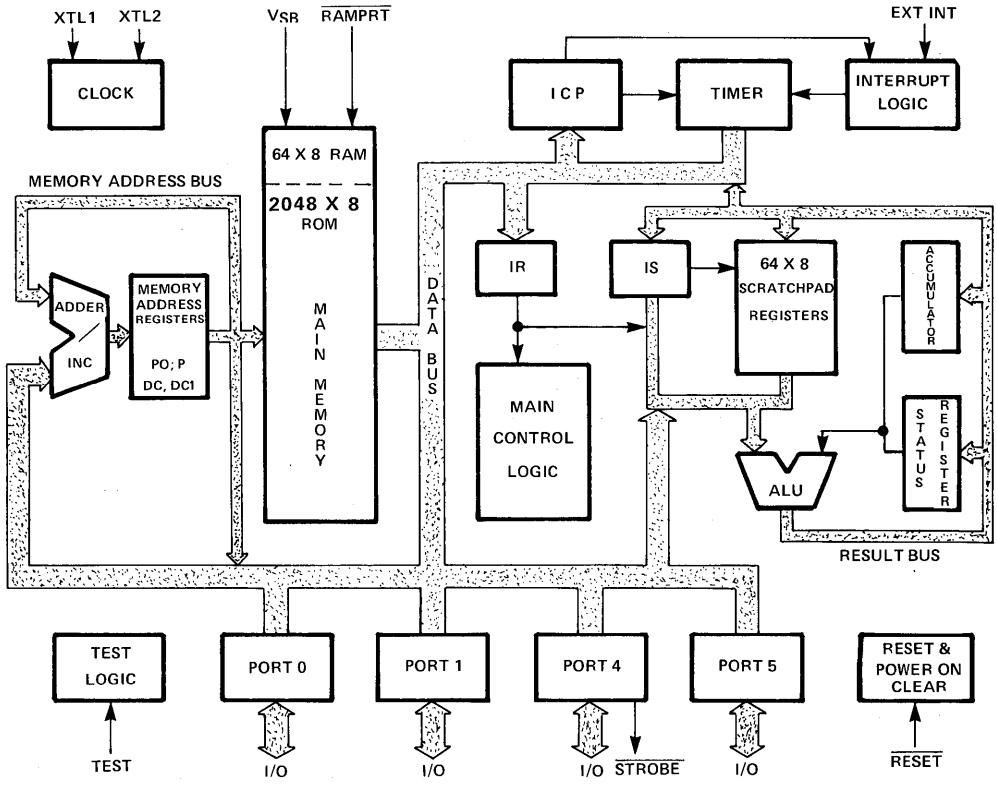
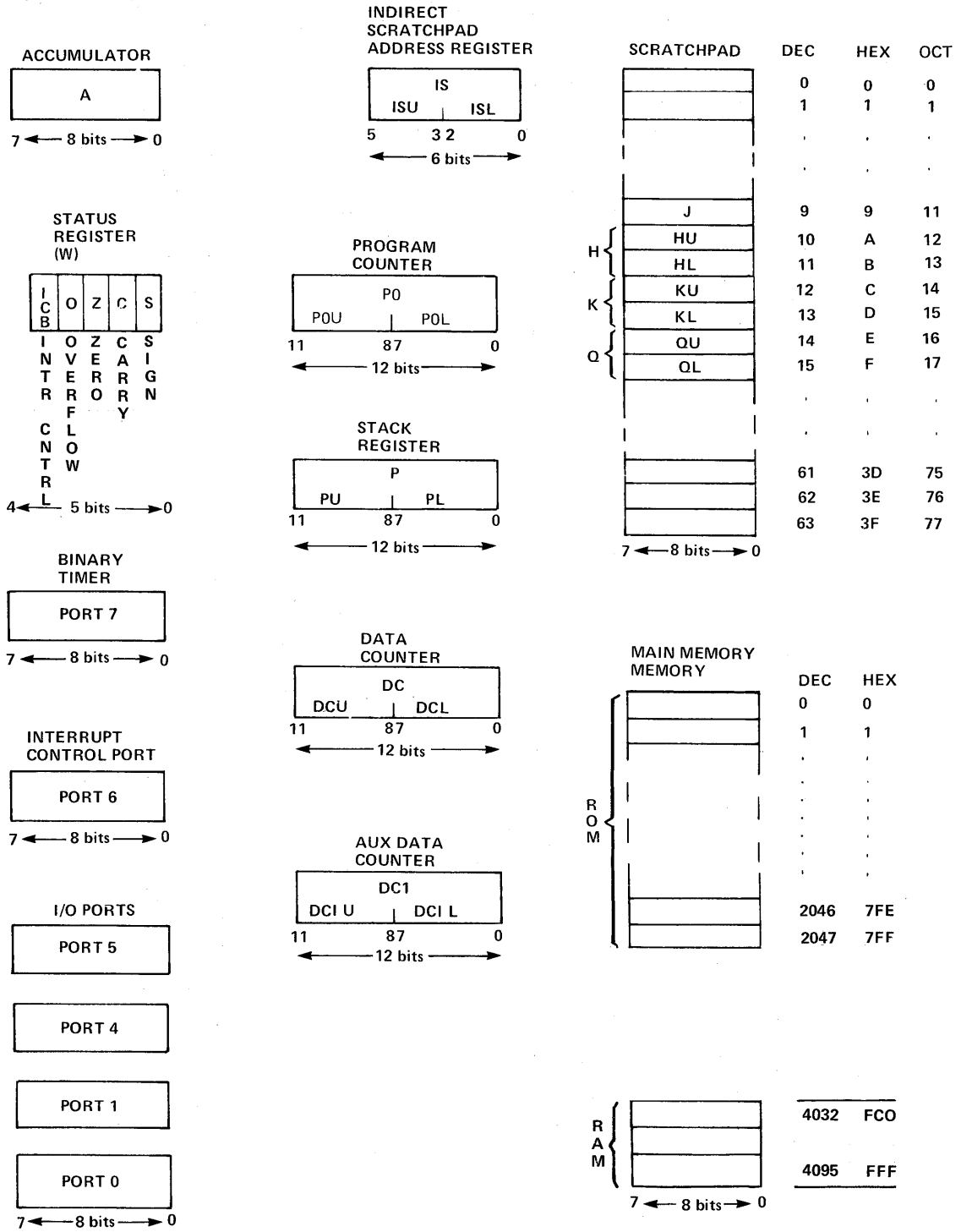


Figure 2


ACCUMULATOR
A
7 ← 8 bits → 0
INDIRECT SCRATCHPAD ADDRESS REGISTER
IS
ISU
ISL
5 3 2 0
6 bits
SCRATCHPAD
DEC HEX OCT
0 0 0
1 1 1
J 9 9 11
H HU 10 A 12
HL 11 B 13
K KU 12 C 14
KL 13 D 15
Q QU 14 E 16
QL 15 F 17
61 3D 75
62 3E 76
63 3F 77
7 ← 8 bits → 0
STATUS REGISTER (W)
ICB O Z C S
INTR CNTRL
OVERFLOW
ZERO
CARRY
SIGN
4 ← 5 bits → 0
PROGRAM COUNTER
P0
P0U
P0L
11 8 7 0
12 bits
STACK REGISTER
P
PU
PL
11 8 7 0
12 bits
BINARY TIMER
PORT 7
7 ← 8 bits → 0
DATA COUNTER
DC
DCU
DCL
11 8 7 0
12 bits
MAIN MEMORY MEMORY
DEC HEX
0 0
1 1
ROM
2046 7FE
2047 7FF
INTERRUPT CONTROL PORT
PORT 6
7 ← 8 bits → 0
AUX DATA COUNTER
DC1
DCI U
DCI L
11 8 7 0
12 bits
I/O PORTS
PORT 5
PORT 4
PORT 1
PORT 0
7 ← 8 bits → 0
RAM
4032 FC0
4095 FFF
7 ← 8 bits → 0

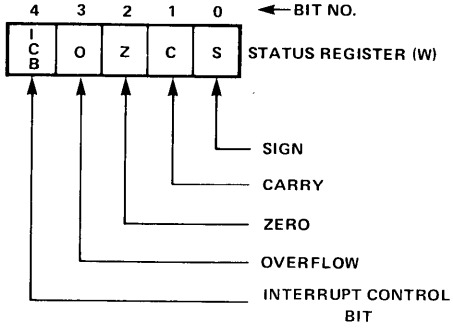
logic, the ALU performs the required arithmetic or logic operations (using the data presented on the two input busses) and provides the result on the result bus. The arithmetic operations that can be performed in the ALU are binary add, decimal adjust, add with carry, decrement, and increment. The logic operations that can be performed are AND, OR, EXCLUSIVE OR, 1's complement, shift right, and shift left. Besides providing the result on the result bus, the ALU also provides four signals representing the status of the result. These signals, stored in the Status Register (W), represent CARRY, OVERFLOW, SIGN, and ZERO condition of the result of the operation.

Accumulator (A)

The Accumulator (A) is the prinicpal register for data manipulation within the 3876 A serves as one input to the ALU for arithmetic or logical operations. The result of ALU operations are stored in A.

The Status Register (W)

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



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 = $\overline{ALU_7}$

Interrupt Control Bit (ICB)

The ICB may be used to allow or disallow interrupts in the 3876 This bit is not the same as the two interrupt enable bits in the Interrupt Control Port (ICP). If the ICB is set and the 3876 interrupt logic communicates an interrupt request to the CPU section, the interrupt will be acknowledged and processed upon completion of the first non-privileged instruction. If the ICB is cleared an interrupt request will not be acknowledged or processed until the ICB is set.

I/O Ports

The 3876 provides four complete bidirectional Input/Output ports. (When standby option is used, Port 0, bit 0 and 1 are not available). These are Ports 0, 1, 4, and 5. In addition, the Interrupt Control Port is addressed as Port 6 and the binary timer is addressed as Port 7. An output instruction (OUT or OUTS) causes the contents of A to be latched into the addressed port. An input instruction (IN or INS) transfers the contents of the port to A (port 6 is an exception which is described later). The I/O pins on the 3876 are logically inverted. The schematic of an I/O pin and available output drive options are shown in Figure 3.

An output ready strobe is associated with Port 4. This flag may be used to signal a peripheral device that the 3876 has just completed an output of new data to Port 4. The strobe provides a single low pulse shortly after the output operation is completely finished, so either edge may be used to signal the peripheral. $\overline{STROBE}$ may also be used as an input strobe simply by doing a dummy output of H '00' to Port 4 after completing the input operation.

Timer and Interrupt Control Port

The Timer is an 8-bit binary down counter which is software programmable to operate in one of three modes: the Interval Timer Mode, the Pulse Width Measurement Mode, or the Event Counter Mode. As shown in Figure 4, associated with the Timer are an 8-bit register called the Interrupt Control Port, a programmable prescaler, and an 8-bit modulo-N register. A functional logic diagram is shown in Figure 5.

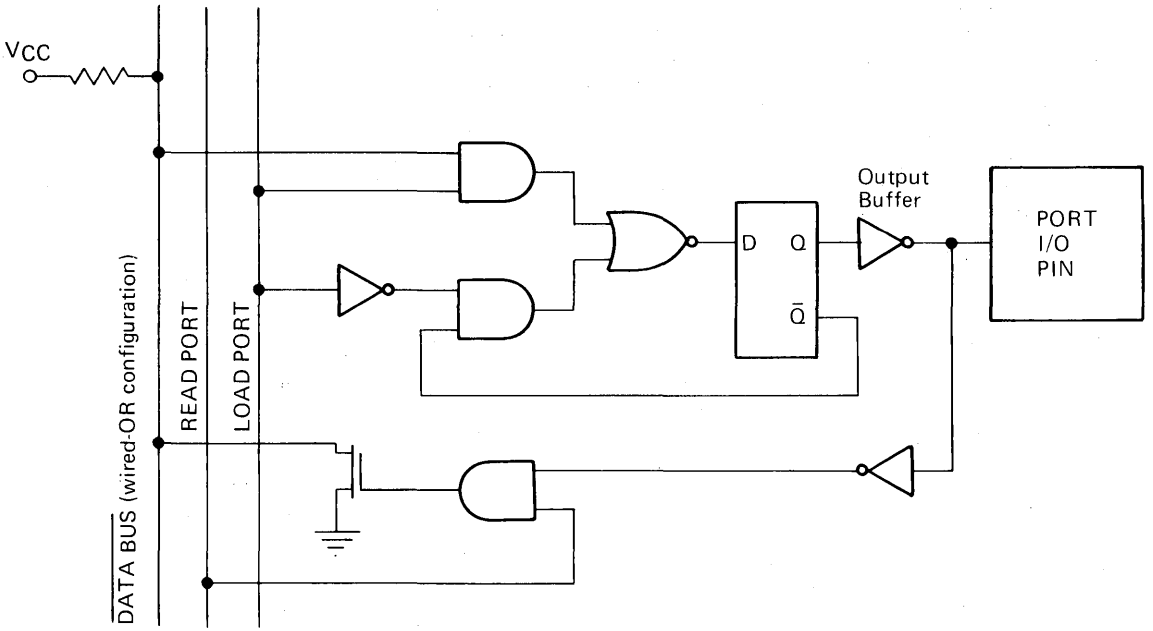
The desired timer mode, prescale value, starting and stopping the timer, active level of the EXT INT pin, and local enabling or disabling of interrupts are selected by outputting the proper bit configuration from the Accumulator to the Interrupt Control Port (Port 6) with an OUT or OUTS instruction. Bits within the Interrupt Control Port are defined as follows:

Interrupt Control Port (Port 6)

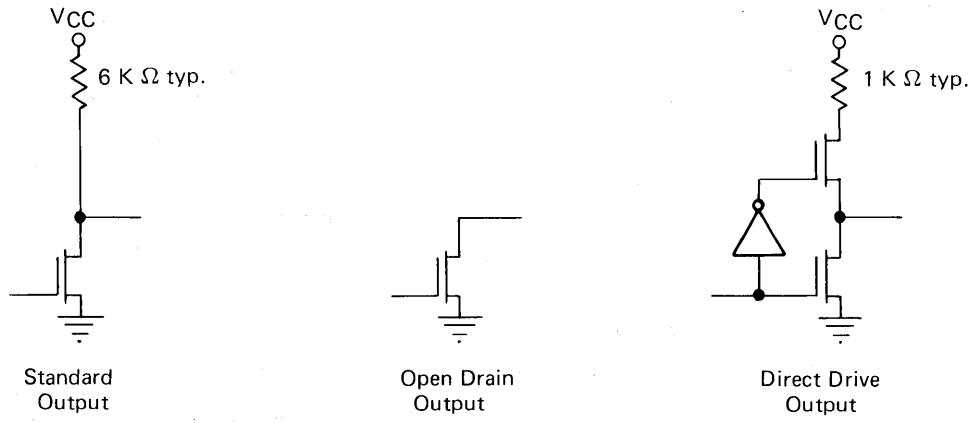
Bit 0 - External Interrupt Enable
Bit 1 - Timer Interrupt Enable
Bit 2 - EXT INT Active Level
Bit 3 - Start/Stop Timer
Bit 4 - Pulse Width/Interval Timer
Bit 5 - ÷ 2 Prescale
Bit 6 - ÷ 5 Prescale
Bit 7 - ÷ 20 Prescale

I/O PIN CONCEPTUAL DIAGRAM WITH OUTPUT BUFFER OPTIONS

Figure 3



OUTPUT BUFFER OPTIONS



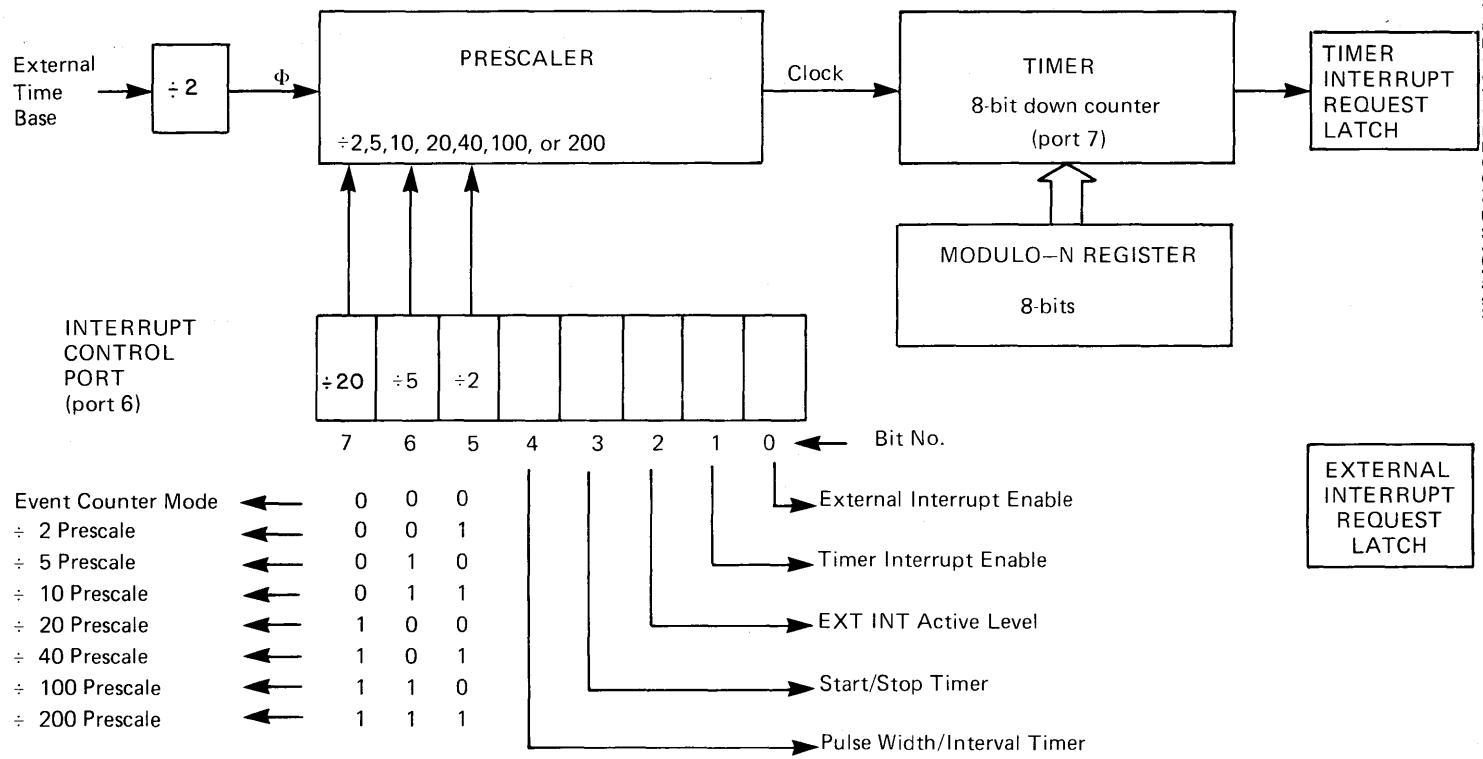
Ports 0 and 1 are Standard Output type only.

Ports 4 and 5 may both be any of the three output options (programmable bit by bit).

The $\overline{\text{STROBE}}$ output is always configured similar to a Direct Drive Output except that it is capable of driving 3 TTL loads.

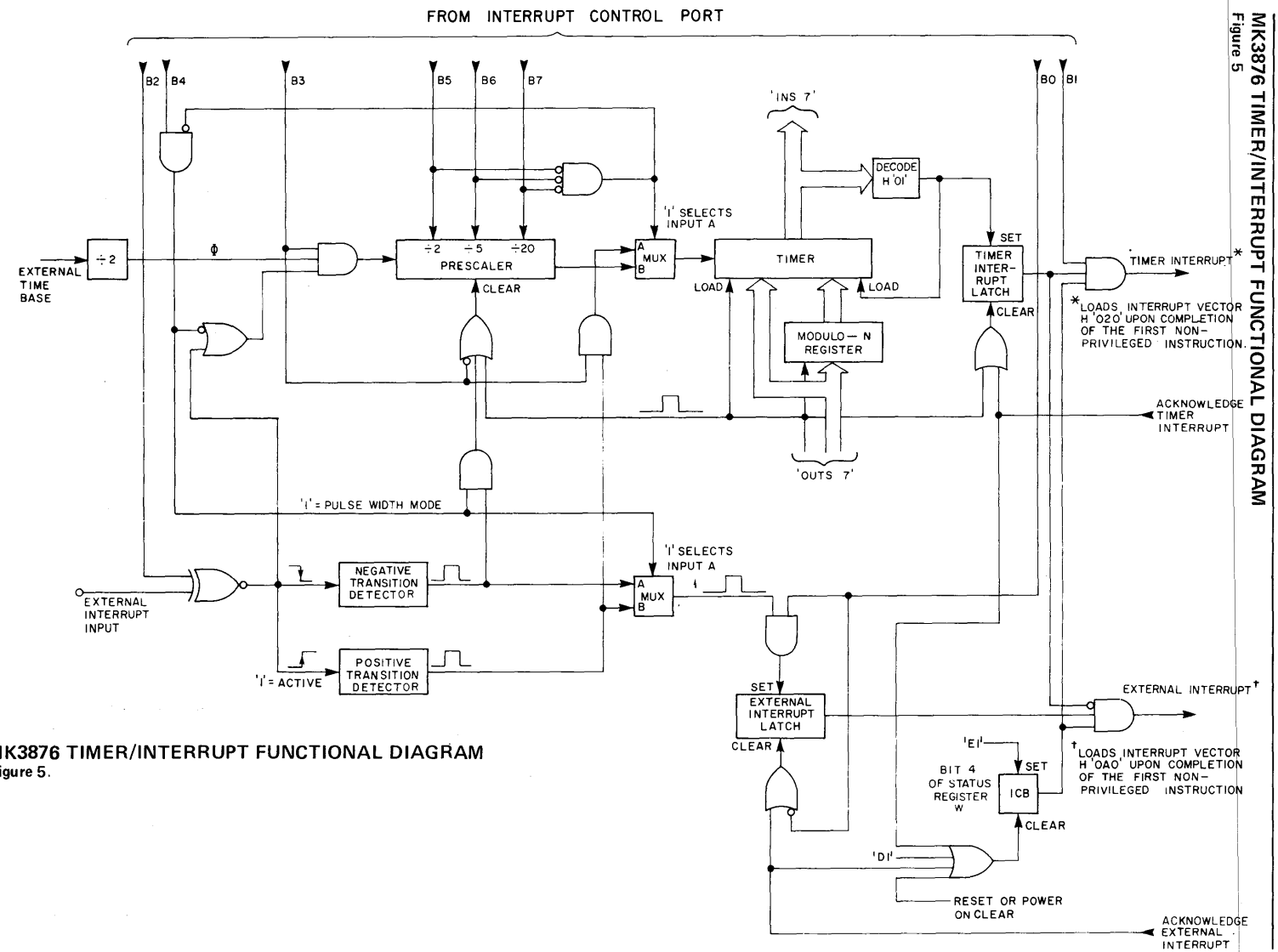
$\overline{\text{RESET}}$ and EXT INT may have standard 6KΩ (typical) pull-up or may have no pull-up. These two inputs have Schmidt trigger inputs with a minimum of 0.2 volts of hysteresis.

TIMER & CONTROL PORT BLOCK DIAGRAM
Figure 4



Note: See Figure 5 for a more detailed functional diagram.

TIMER & CONTROL PORT BLOCK DIAGRAM
Figure 4



MK3876 TIMER/INTERRUPT FUNCTIONAL DIAGRAM
Figure 5.

A special situation exists when reading the Interrupt Control Port (with an IN or INS instruction). The Accumulator is not loaded with the content of the ICP; instead, Accumulator bits 0 through 6 are loaded with 0's while bit 7 is loaded with the logic level being applied to the EXT INT pin, thus allowing the status of EXT INT to be determined without the necessity of servicing an external interrupt request. When reading the Interrupt Control Port (Port 6) bit 7 of the Accumulator is loaded with the actual logic level being applied to the EXT INT pin, regardless of the status of ICP bit 2 (the EXT INT Active Level bit); that is, if EXT INT is at +5V bit 7 of the Accumulator is set to a logic 1, but if EXT INT is at GND then Accumulator bit 7 is reset to logic 0. This capability is useful in establishing a high speed polled handshake procedure or for using EXT INT as an extra input pin if external interrupts are not required and the Timer is used only in the Interval Timer Mode. However, if it is desirable to read the contents of the ICP then one of the 64 scratchpad registers or one byte of RAM may be used to save a copy of whatever is written to the ICP.

The rate at which the timer is clocked in the Interval Timer Mode is determined by the frequency of an internal Φ clock and by the division value selected for the prescaler. (The internal Φ clock operates at one-half the external time base frequency). If ICP bit 5 is set and bits 6 and 7 are cleared, the prescaler divides Φ by 2. Likewise, if bit 6 or 7 is individually set the prescaler divides Φ by 5 or 20 respectively. Combinations of bits 5, 6 and 7 may also be selected. For example, if bits 5 and 7 are set while 6 is cleared the prescaler will divide by 40. Thus possible prescaler values are ÷2, ÷5. ÷10, ÷20, ÷40, ÷100, and ÷200.

Any of three conditions will cause the prescaler to be reset: whenever the timer is stopped by clearing ICP bit 3, execution of an output instruction to Port 7, (the timer is assigned port address 7), or on the trailing edge transition of the EXT INT pin when in the Pulse Width Measurement Mode. These last two conditions are explained in more detail below.

An OUT or OUTS instruction to Port 7 will load the content of the Accumulator to both the Timer and the 8-bit modulo-N register, reset the prescaler, and clear any previously stored timer interrupt request. As previously noted, the Timer is an 8-bit down counter which is clocked by the prescaler in the Interval Timer Mode and in the Pulse Width Measurement Mode. The prescaler is not used in the Event Counter Mode. The Modulo-N register is a buffer whose function is to save the value which was most recently outputted to Port 7. The modulo-N register is used in all three timer modes.

Interval Timer Mode

When ICP bit 4 is cleared (logic 0) and at least one prescale bit is set the Timer operates in the Interval Timer Mode. When bit 3 of the ICP is set the Timer will start counting down from the modulo-N value. After counting down to H'01', the Timer returns to the modulo-N value at the next count. On the transition from H'01' to H 'N' the Timer sets a timer interrupt request latch. Note that the interrupt request latch is set by the transition to H 'N' and not be the presence of H 'N' in the Timer, thus allowing a full 256 counts if the modulo-N register is preset to H '00'. If bit 1 of the ICP is set, the interrupt request is passed on to the CPU section of the 3872. However, if bit 1 of the ICP is a logic 0 the interrupt request is not passed on to the CPU section but the interrupt request latch remains set. If ICP bit 1 is subsequently set, the interrupt request will then be passed on to the CPU section. (Recall from the discussion of the Status Register's Interrupt Control Bit that the interrupt request will be acknowledged by the CPU section only if ICB is set). Only two events can reset the timer interrupt request latch; when the timer interrupt request latch is acknowledged by the CPU section, or when a new load of the modulo-N register is performed.

Consider an example in which the modulo-N register is loaded with H '64' (decimal 100). The timer interrupt request latch will be set at the 100th count following the timer start and the timer interrupt request latch will repeatedly be set on precise 100 count intervals. If the prescaler is set at ÷40 the timer interrupt request latch will be set every 4000 Φ clock periods. For a 2MHz Φ clock (4MHz time base frequency) this will produce 2 millisecond intervals.

The range of possible intervals is from 2 to 51,200 Φ clock periods (1μs to 25.6ms for a 2MHzΦ clock). However, approximately 50 Φ periods is a practical minimum because the time between setting the interrupt request latch and the execution of the first instruction of the interrupt service routine is at least 29 Φ periods (the response time is dependent upon how many privileged instructions are encountered when the request occurs); 29 is based on the timer interrupt occuring at the beginning of a non-privileged short instruction. To establish time intervals greater than 51,200 Φ clock periods is a simple matter of using the timer interrupt service routine to count the number of interrupts, saving the result in one or more of the scratchpad registers until the desired interval is achieved. With this technique virtually any time interval, or several time intervals, may be generated.

The Timer may be read at any time and in any mode using an input instruction (IN 7 or INS 7) and may

take place "on the fly" without interferring with normal timer operation. Also, the Timer may be stopped at any time by clearing bit 3 of the ICP. The timer will hold its current contents indefinitely and will resume counting when bit 3 is again set. Recall however that the prescaler is reset whenever the Timer is stopped; thus a series of starting and stopping will result in a cumulative truncation error.

A summary of other timer errors is given in the timing section of this specification. For a free running timer in the Interval Timer Mode the time interval between any two interrupt requests may be in error by ± 6 Φ clock periods although the cumulative error over many intervals is zero. The prescaler and Timer generate precise intervals for setting the timer interrupt request latch but the time out may occur at any time within a machine cycle. (There are two types of machine cycles: short cycles which consist of 4 Φ clock periods and long cycles which consist of 6 Φ clock periods. In the multi-chip F8 family there is a signal called the WRITE clock which corresponds to a machine cycle). Interrupt requests are synchronized with the internal WRITE clock thus giving rise to the possible ± 6 Φ error. Additional errors may arise due to the interrupt request occurring while a privileged instruction or multicycle instruction is being executed. Nevertheless, for most applications all of the above errors are negligible, especially if the desired time intervall is greater than 1ms.

Pulse Width Measurement Mode

When ICP bit 4 is set (logic 1) and at least one prescale bit is set the Timer operates in the Pulse Width Measurement Mode. This mode is used for accurately measuring the duration of a pulse applied to the EXT INT pin. The Timer is stopped and the prescaler is reset whenever EXT INT is at its inactive level. The active level of EXT INT is defined by ICP bit 2; if cleared, EXT INT is active low; if set, EXT INT is active high. If ICP bit 3 is set, the prescaler and Timer will start counting when EXT INT transitions to the active level. When EXT INT returns to the inactive level the Timer then stops, the prescaler resets, and if ICP bit 0 is set an external interrupt request latch is set. (Unlike timer interrupts, external interrupts are not latched if the ICP Interrupt Enable bit is not set).

As in the Interval Timer Mode, the Timer may be read at any time, may be stopped at any time by clearing ICP bit 3, the prescaler and ICP bit 1 function as previously described, and the Timer still functions as an 8-bit binary down counter with the timer interrupt request latch being set on the Timer's transition from H '01' to H 'N'. Note that the EXT INT pin has nothing to do with loading the Timer; its action is that of automatically starting and stopping the Timer and of generating external interrupts. Pulse widths longer than the prescale value times the modulo-N value are easily measured by using the timer interrupt service routine to store the number of timer interrupts in one or more scratchpad registers.

As for accuracy, the actual pulse duration is typically slightly longer than the measured value because the status of the prescaler is not readable and is reset when the Timer is stopped. Thus for maximum accuracy it is advisable to use a small division setting for the prescaler.

Event Counter Mode

When ICP bit 4 is cleared and all prescale bits (ICP bits 5, 6, and 7) are cleared the Timer operates in the Event Counter Mode. This mode is used for counting pulses applied to the EXT INT pin. If ICP bit 3 is set the Timer will decrement on each transition from the inactive level to the active level or the EXT INT pin. The prescaler is not used in this mode, but as in the other two timer modes, the timer may be read at any time, may be stopped at any time by clearing ICP bit 3, ICP bit 1 functions as previously described, and the timer interrupt request latch is set on the Timer's transition from H '01' to H 'N'.

Normally ICP bit 0 should be kept cleared in the Event Counter Mode; otherwise, external interrupts will be generated on the transition from the inactive level to the active level of the EXT INT pin.

For the Event Counter Mode the minimum pulse width required on EXT INT is 2 Φ clock periods and the minimum inactive time is 2 Φ clock periods; therefore, the maximum repetition rate is 500KHz.

Timer Emulation

For total software compatibility when expanding into a multi-chip configuration the MK3871 Peripheral Input/Output circuit should be used rather than the older MK3861 PIO. The MK3871 has the same improved Timer (binary count, readable, and three modes of operation rather than one) and ready strobe output as are on the MK3876.

External Interrupts

When the timer is in the Interval Timer Mode the EXT INT pin is available for non-timer related interrupts. If ICP bit 0 is set an external interrupt request latch is set when there is a transition from the inactive level to the active level of EXT INT. (EXT INT is an edge-triggered input). The interrupt request is latched until either acknowledged by the CPU section or until ICP bit 0 is cleared (unlike timer interrupt requests which remain latched even

when ICP bit 1 is cleared). External interrupts are handled in the same fashion when the Timer is in the Pulse Width Measurement Mode or in the Event Counter Mode, except that only in the Pulse Width Measurement Mode the external interrupt request latch is set on the trailing edge of EXT INT, that is, on the transition from the active level to the inactive level.

Interrupt Handling

When either a timer or an external interrupt request is communicated to the CPU section of the 3876 it will be acknowledged and processed at the completion of the first non-privileged instruction if the Interrupt Control Bit of the Status Register is set. If the Interrupt Control Bit is not set, the interrupt request will continue until either the Interrupt Control Bit is set and the CPU section acknowledges the interrupt or until the interrupt request is cleared as previously described.

If there is both a timer interrupt request and an external interrupt request when the CPU section starts to process the requests, the timer interrupt is handled first.

When an interrupt is allowed the CPU section will request that the interrupting element pass its interrupt vector address to the Program Counter via the data bus. The vector address for a timer interrupt is H '020'. The vector address for external interrupts is H '0A0'. After the vector address is passed to the Program Counter, the CPU section sends an acknowledge signal to the appropriate interrupt request latch which clears that latch. The execution of the interrupt service routine will then commence. The return address of the original program is automatically saved in the Stack Register, P.

The Interrupt Control Bit of W (Status Register) is automatically reset when an interrupt request is acknowledged. It is then the programmer's responsibility to determine when ICB will again be set (by executing an EI instruction). This action prevents an interrupt service routine from being interrupted unless the programmer so desires.

Figure 6 details the interrupt sequence which occurs whether the interrupt request is from an external source via EXT INT or from the 3876's internal timer. Events are labeled with the letters A through G and are described below.

Event A

An interrupt request must satisfy a hold time requirement as specified in the AC Charactertics in order to guarantee that it is valid on the rising edge of the WRITE clock.

Event B

Event B represents the instruction being executed when the interrupt occurs. The last cycle of B is normally the instruction fetch for the next cycle. However, if B is not a privileged instruction and the CPU's Interrupt Control Bit is set, then the last cycle becomes a "freeze" cycle rather than a fetch. At the end of the freeze cycle the interrupt request latches are inhibited from altering the interrupt daisy-chain so that sufficient time will be allowed for the daisy-chain to settle. (If B is a privileged instruction, the instruction fetch is not replaced by a freeze cycle; instead, the fetch is performed and the next instruction is executed. Although unlikely to be encountered, a series of privileged instructions will be sequentially executed without Interrupt. One more instruction, called a 'protected' instruction, will always be executed after the last privileged instruction. The last cycle of the protected instruction then performs the freeze.)

The dashed lines on EXT INT illustrate the last opportunity for EXT INT to cause the last cycle of a non-protected instruction to become a freeze cycle.

The freeze cycle is a short cycle (4 Φ clock periods) in all cases except where B is the Decrement Scratchpad instruction, in which case the freeze cycle is a long cycle (6 Φ clock periods).

INT REQ goes low on the next negative edge of WRITE if the appropriate interrupt enable bit of the Interrupt Control Port is set. Both INT REQ and WRITE are internal signals.

Event C

A NO-OP long cycle to allow time for the PRI IN/PRI OUT chain to settle.

Event D

The program counter (PO) is pushed to the stack register (P) in order to save the return address. The interrupt circuitry places the lower 8 bits of the interrupt vector address onto the data bus. This is always a long cycle.

Event E

A long cycle in which the interrupt circuitry places the upper 8 bits of the interrupt vector address onto the data bus.

Event F

A short cycle in which the interrupting interrupt request latch is cleared. Also, the CPU's Interrupt Con-

Event F (Cont'd)

trol Bit is cleared, thus disabling interrupts until an EI instruction is performed. The fetch of the next instruction from the interrupt address.

Event G

Begin execution of the first instruction of the interrupt service routine.

Summary Of Interrupt Sequence

For the MK3876 the interrupt response time is defined as the time elapsed between the occurence of EXT INT going active (or the Timer transitioning to H'N') and the beginning of execution of the first instruction of the interrupt service routine. The interrupt response time is a variable dependent upon what the microprocessor is doing when the interrupt request occurs. As shown in Figure 5, the minimum interrupt response time is 3 long cycles plus 2 short cycles plus one WRITE clock pulse width plus a setup time of EXT INT prior to the leading edge of the WRITE pulse – a total of 27 Φ clock periods plus the setup time. At a 2 MHz Φ this is 14.25 μs. Although the maximum could theoretically be infinite, a practical maximum is 35 μs (based on the interrupt request occurring near the beginning of a PI and LR K, P sequence).

Power-On Clear

The intent of the Power-On-Reset circuitry on the 3876 is to automatically reset the device following a typical power-up situation, thus saving external reset circuitry in many applications. This circuitry is not guaranteed to sense a "Brown Out" (low voltage) condition nor is it guaranteed to operate under all possible power-on situations.

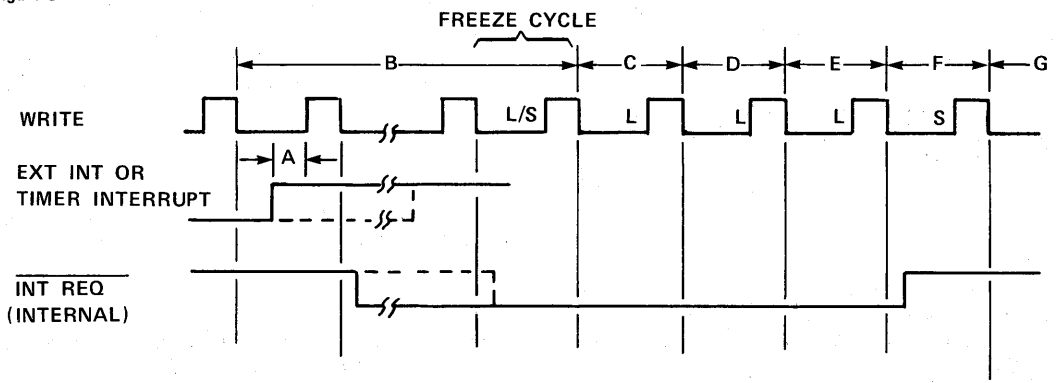
Three conditions are required before the 3876 will leave the reset state and begin operation. Refer to Figure 7 as an aid to the following descriptions. The on chip Vcc detector senses a minimum value of Vcc before it will allow the 3876 to operate. The threshold of this detector is set by analog circuitry because a stable voltage reference is not available with n-channel MOS processing. Processing variations will cause this threshold to vary from a low of 3.0 volts to a high of 4.3 volts with 3.5 volts being typical.

The 3876 uses a substrate bias as a technique to provide improved performance versus power consumption relative to conventional grounded substrate approaches. This bias generator may start operating as low as Vcc = 3 volts in order to get adequate substrate bias. Until the substrate reaches the proper bias, the 3876 will not be released from the reset state. The final condition required is that the clocks of the 3876 must be functioning. Typically the clocks will start to function at Vcc equal to 3 to 3.5 volts but since the part is tested at 4.5 volts MOSTEK cannot guarantee any operation below 4.5 volts. The output of the delay circuit in Figure 7 will stay low until the clocks start to function. If the input to the delay circuit is high, typically after 100 cycles of the WRITE clock (800 cycles of the external clock) the output of the delay circuit will go high allowing the 3876 to begin execution.

If Vcc falls to ground for at least a few hundred nanoseconds, the output of the delay circuit will go low immediately and the 3876 will reset.

The internal logic may detect a valid Vcc, bias and clocks at Vcc = 3.5 volts and allow the 3876 to start executing after the time delay. With a slowly rising power supply the part may start running before Vcc is above 4.5 volts, which is below the guaranteed voltage range. When power-on-clear is required with a slowly rising power supply, an external capacitor must be used on the $\overline{\text{RESET}}$ pin to hold it below 0.8

INTERRUPT SEQUENCE
Figure 6



Power-On Clear (Cont'd)

volts until Vcc is stable above 4.5 volts. (Note: The option to disconnect the internal pull-up resistor on RESET is available which allows the use of a larger external pull-up resistor and a small capacitor on RESET.)

In many applications, it is desirable if the unit does an automatic power-on-clear, but not mandatory. The unit will have a RESET push button and if the unit does not power-up correctly or malfunctions because of some disturbance on the Vcc line, the operator will simply press RESET and restore normal operation. It is for these applications that the internal power-on-clear circuitry was designed.

In some applications it is requried that the microcomputer continue to run properly without operator intervention after brown-outs, power line disturbances, electrical noise, computer malfunction due to a programming bug or any other disturbance except a catastrophic failure of some component.

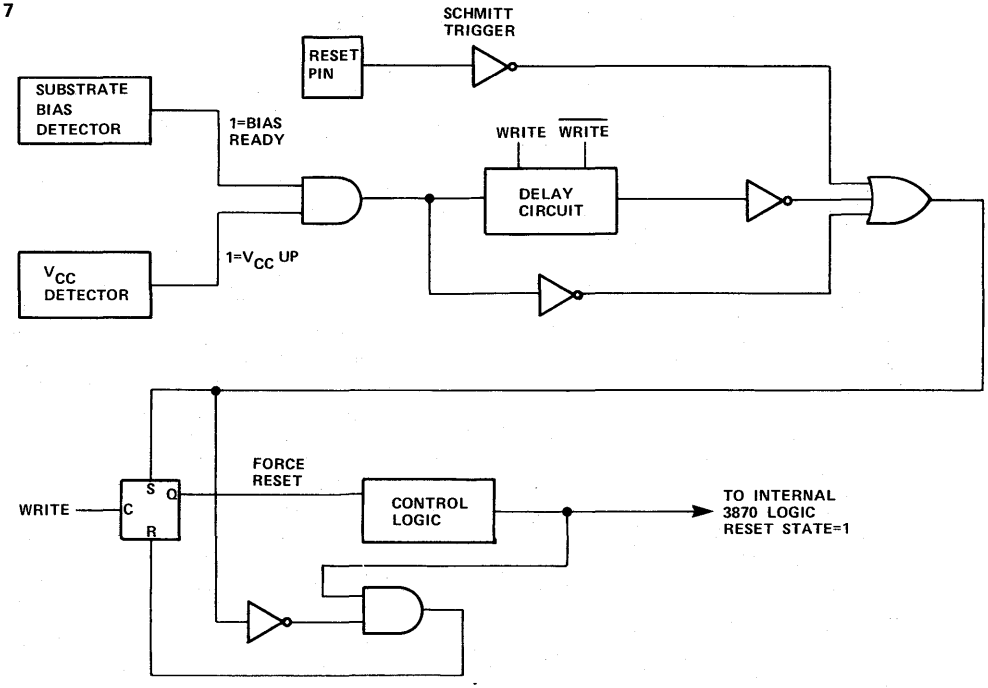
One concept used to keep computers running is that of the "WATCHDOG TIMER". The computer is programmed to periodically reset the watchdog timer during the normal execution of its program (this is easily done in the 3876 as its normal application is in some control function which is typically periodic). As long as the computer continues to execute its program the watchdog timer is continually reset and never times out. Should the computer stop executing its program for whatever reason, the watchdog timer will time out producing a RESET pulse to the CPU restarting execution. This is a very positive way to assure that the computer is doing its job, i.e., executing the program. It is important that the software driving the watchdog timer test as many functional blocks (timer, ALU, scratchpad RAM, and Ports) of the 3876 as possible before reseting the watchdog timer. This is because operation of the 3876 with an out of spec power supply may allow some of the functions to operate correctly while other functions are not operable.

MOSTEK can guarantee correct operation of the 3876 only while the Vcc voltage remains within its specified limits. If proper operation of the 3876 must be guaranteed after a disturbance on the Vcc line, then an external circuit must be used to monitor the Vcc line and produce a $\overline{\text{RESET}}$ to the 3876 whenever Vcc is out of the specified limits.

A related characteristic to power-on-clear is the Start-up time of the basic timing element. The LC and RC oscillators begin to function almost immediately once Vcc is high enough to allow the on-board

POWER ON CLEAR BLOCK DIAGRAM

Figure 7



Power-On Clear (Cont'd)

oscillator to operate (Vcc = 3.5V). Operation with a crystal is partly mechanical and some start time is required to get the mass of the crystal into vibrational motion. This time is basically dependent on the frequency (mass) of the crystal. 4 MHz crystals typically require about 2-3 mSec to start while 1 MHz crystals require 60-70 mSect to start oscillating. Of course, this time may vary greatly from crystal to crystal and is also a function of the power supply rise time characteristic, however, the higher frequency crystals start faster and are definitely recommended (i.e., 3-4 MHz).

The condition of the port pins during the power-on-clear sequence is often asked, The port pins or the STROBE line cannot be specified until Vcc reaches 4.5V and the 3876 enters the RESET state. Before this, the port pins may stay at Vss, may track Vcc as it rises, or they may track Vcc part way up then return to Vss (Ports 4 and 5 will go to Vcc once the clocks are running and the 3870 has sufficient Vcc to properly operate the internal control logic and I/O ports, Ports 0 and 1 must be controlled by the program).

External Reset

When $\overline{\text{RESET}}$ is taken low the content of the Program Counter is pushed to the Stack Register and then the Program Counter and the ICB bit of the W Status Register are cleared. The original Stack Register content is lost. Ports 4, 5, 6 and 7 are loaded with H '00'. The contents of all other registers and ports are unchanged. When power is first applied all ports and registers are undefined until a reset is performed. When $\overline{\text{RESET}}$ is taken high the first program instruction is fetched from ROM location H '000'. When an external reset of the 3876 occurs, P0 is pushed into P and the old contents of P are lost. It must be noted that an external reset is recognized at the start of a machine cycle and not necessarily at the end of an instruction. Thus if the 3876 is executing a multi-cycle instruction, that instruction is not completed and the contents of P upon reset may not necessarily be the address of the instruction that would have been executed next. It may, for example, point to an immediate operand if the reset occurred during the second cycle of a LI or CI instruction. Additionally, several instructions (JMP, PI, PK, LR P0, Q) as well as the interrupt acknowledge sequence modify P0 in parts. That is, they alter P0 by first loading one part then the other and the entire operation takes more than one cycle. Should reset occur during this modification process the value pushed into P will be part of the old P0 (the as yet unmodified part) and part of the new P0 (already modified part). Thus care should be taken (perhaps by external gating) to insure that reset does not occur at an undesirable time if any significance is to be given to the contents of P after a reset occurs.

V_{CC} Decoupling

The 3870 family devices have dynamic circuitry internally which requires a good high frequency decoupling capacitor to surpress noise on the Vcc line. A .01 μF or .1 μF ceramic capacitor should be placed between Vcc and ground, located physically close to the 3870 device. This will reduce noise generated by the 3870 to about 70-100 mVolts on the Vcc line.

Test Logic

Special test logic is implemented to allow access to the internal main data bus for test purposes.

In normal operation the TEST pin is unconnected or is connected to GND. When TEST is placed at a TTL level (2.0V to 2.6V) Port 4 becomes an output of the internal data bus and Port 5 becomes a wired-OR input to the internal data bus. The data appearing on the Port 4 pins is logically true whereas input data forced on Port 5 must be logically false. When TEST is placed at high level (6.0V to 7.0V), the ports act as above and additionally the 2K x 8 program ROM is prevented from driving the data bus. In this mode operands and instructions may be forced externally through Port 5 instead of being accessed from the program ROM. When TEST is in either the TTL state or the high state, $\overline{\text{STROBE}}$ ceases its normal function and becomes a machine cycle clock (identical to the F8 multi-chip system WRITE clock except inverted).

Timing complexities render the capabilities associated with the TEST pin impractical for use in a user's application, but these capabilities are thoroughly sufficient to provide a rapid method for thoroughly testing the 3876.

STANDBY POWER OPTION

If the standby power option has not been selected Port 0-bit 0 and 1 are readable and writeable and the RAM Protect ($\overline{\text{RAMPRT}}$) signal is not operational. If the power down option is selected, Port 0 -bit 0 and bit 1 are readable only. The standby power source (V_{SB}) is connected to Pin 4 and $\overline{\text{RAMPRT}}$ control to Pin 3. It is recommended that Nickel-Cadinium batteries (typical voltage of series cells = 2.5V) be used for standby power, since the MK3876 can automatically trickle charge the two Ni-Cad's. If more than two cells in series are used, the charging circuit must be provided outside the MK3876. Whenever $\overline{\text{RAMPRT}}$ is brought low, the standby RAM (64x8 bit words in P0/DC address space, 4032 to 4095_{10} or FC0 to FFF_{16}) is disabled from being read or written. Also the RAM itself is switched from V_{CC} power to the V_{SB} power. Three modes of powering down are recommended, see Figure 8. In the first

STANBY POWER OPTION (Cont'd)

mode, $\overline{\text{RESET}}$ and the $\overline{\text{RAMPRT}}$ pins are tied together. If data is to be saved in RAM, the processor must be interrupted early enough to save all necessary data before the V_{CC} falls below the minimum level. After the save is done, the $\overline{\text{RESET}}$ and $\overline{\text{RAMPRT}}$ can fall. This prevents any further access of the RAM; V_{CC} may now fall. As the power comes up, the $\overline{\text{RESET}}$/$\overline{\text{RAMPRT}}$ signal should be held low until V_{CC} is above the minimum level.

In the second mode of operation, the $\overline{\text{RESET}}$ pin is not tied to $\overline{\text{RAMPRT}}$. When these pins are brought high, the 3876 will begin execution at location 000. On power up a normal execution may begin but the program must monitor the Port 0-0 pin (Pin 3) and wait until the Port 0-0 $\overline{\text{RAMPRT}}$ pin is high before attempting any access of the RAM. With this approach, the RAM is not switched to standby power each time the $\overline{\text{RESET}}$ goes low.

If a special save data routine is not needed then the $\overline{\text{EXT INTERRUPT}}$ need not be used and the only requirement to save the RAM data is that $\overline{\text{RAMPRT}}$ be low before V_{CC} drops below 4.5V. For example if a few key variables are to be stored in RAM and it is desired that these be saved during a loss of power, two copies of each variable are kept with an associated flag, thus no interrupt and save routine is necessary. The method of updating a variable is as follows:

- Clear Flag Word 1
- Update Variable (Copy 1)
- Set Flag Word 1
- Clear Flag Word 2
- Update Variable (Copy 2)
- Set Flag Word 2

Now execution may terminate at any time, even during the update of a variable or flag word, causing that byte in RAM to be bad data. There is always a good data byte which contains either the most recent or next most recent value of the variable. Any copy of the variable where the flag word is "set" is a good data byte. While this method significantly encumbers the data storage process, it eliminates the need for a power fail interrupt which both reduces external circuitry and leaves the external interrupt pin completely free for other use.

Figure 9 represents the internal circuitry which can be connected to pins 3 and 4 to provide this Standby Mode. If the Standby Mode is selected, switches A1 and A2 are masked in the position shown, thereby disconnecting the normal port circuitry from pins 3 and 4. Switches B1 and B2 are masked in the position shown to allow pin 3 to become the control ($\overline{\text{RAMPRT}}$) and pin 4 the power (Vsb) for the Standby Mode. If the Standby Mode is not selected all switches are masked opposite of the positions shown and pins 3 and 4 become normal 3870 type ports.

$\overline{\text{RAMPRT}}$ is an input signal used to control access to the Standby RAM. If $\overline{\text{RAMPRT}}$ is high, access to the 64 Byte Standby RAM is permitted by the CPU via the Program Counter (PO) of the Data Counter (DC). The Standby RAM current is supplied by the series pass transistor and a 4 to 12 mA current can be supplied out of pin 4 (Vsb) to trickle charge two Ni-Cad cells (nominal 2.5 volts). The resistors shown simulate device impedances that limit the current available at pin 4 so that the battery is not overcharged. If $\overline{\text{RAMPRT}}$ is low, the Control Logic turns off the pass transistor and the Standby RAM is maintained by a current supplied by the battery connected to pin 4. When $\overline{\text{RAMPRT}}$ is low, the CPU cannot access the Standby RAM thereby protecting its contents as Vcc fails.

The Standby RAM can be maintained by a capacitor, however, a resistor and a diode will be required in order to charge the capacitor to Vcc. Internal voltage drops will not allow Vsb to go above 3 volts (typically) without this external resistor.

3876 Clocks

The time base for the 3876 may originate from one of four sources.

The four configurations are shown in Figure 10. There is an internal 26pF capacitor between XTL 1 and GND and an internal 26pF capacitor between XTL 2 and GND, thus external capacitors are not neccessarily required. In all external clock modes the external time base frequently is divided by two to form the internal Φ clock.

Crystal Selection

The use of a crystal as the time base is highly recommended as the frequency stability and reproducability from system to system is unsurpassed. The 3876 has an internal divide by two to allow the user of inexpensive and widely available TV Color Burst Cyrstals (3.58MHz). The following crystal parameters and vendors are suggested for 3876 applications:

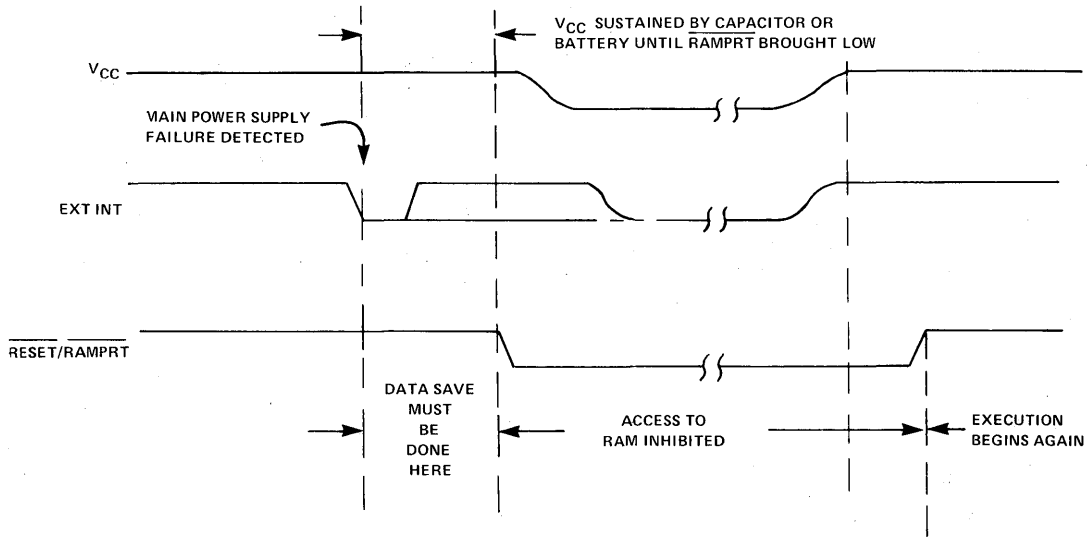
Parameters

a) Parallel Resonance, Fundamental Mode AT-Cut, HC-33/μ holder
b) Frequency Tolerance measured with 18pF load (0.1% accuracy). Drive level 10mW.
c) Shunt Capacitance (Co) = 7pF max.
d) Series Resistance (Rs)

f = 1MHz	Rs	=	550 ohms	max.
f = 2MHz	Rs	=	300 ohms	max.
f = 3MHz	Rs	=	100 ohms	max.
f = 3.58MHz	Rs	=	100 ohms	max.
f = 4MHz	Rs	=	100 ohms	max.

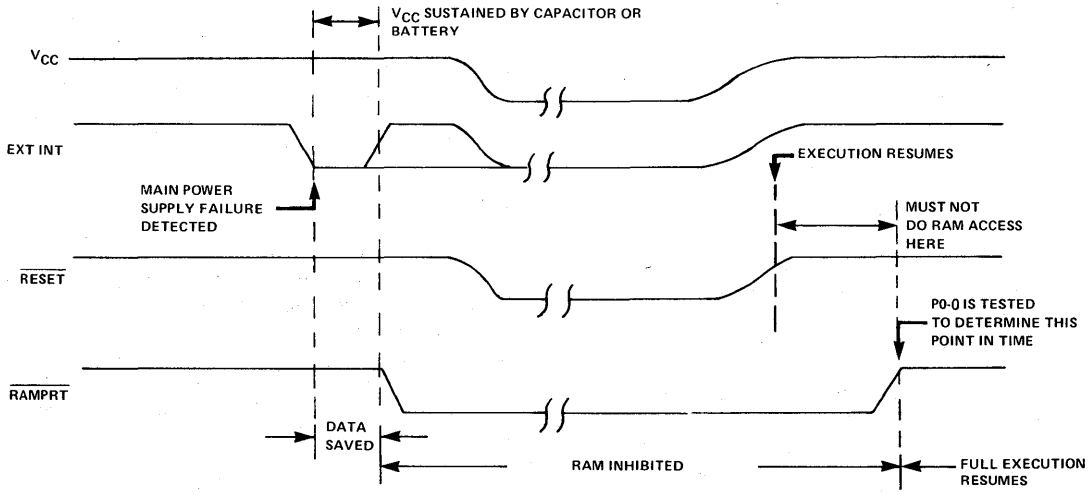
RESET TIED TO RAMPRT, $V_{SB} \geq 2.2$ VOLTS

Figure 8a



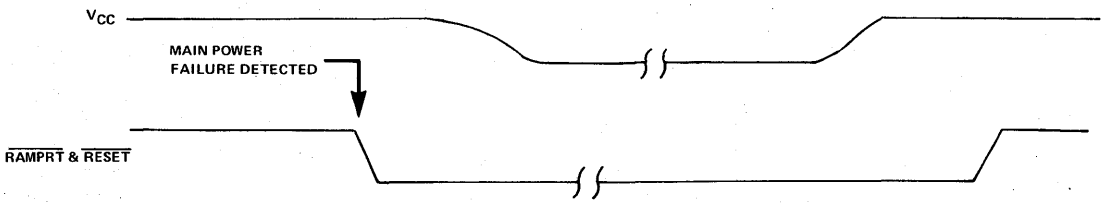
RAMPRT INDEPENDENT OF RESET, $V_{SB} \geq 2.2$ VOLTS

Figure 8b

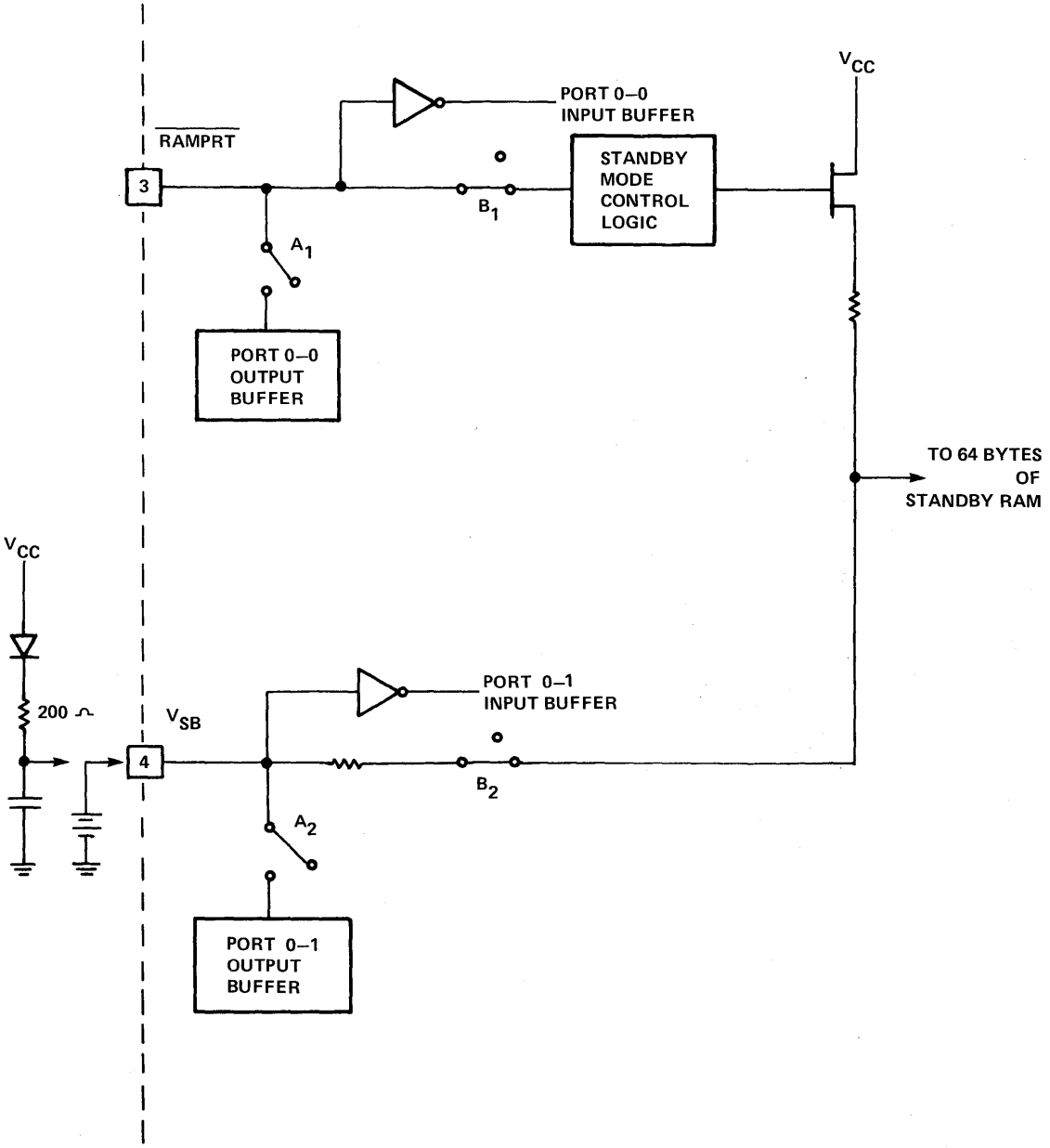


NO SAVE ROUTINE REQUIRED, $V_{SB} \geq 2.2$ VOLTS

Figure 8c

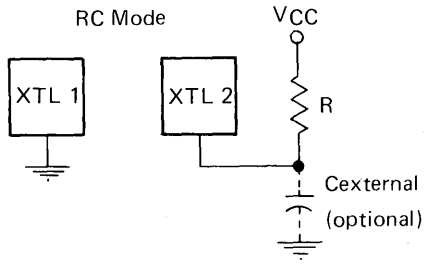


MK3876 STANDBY MODE CIRCUITRY
Figure 9



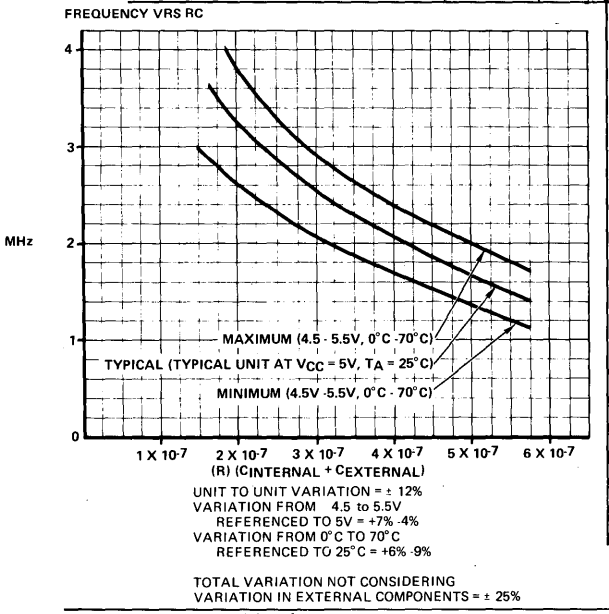
CLOCK CONFIGURATIONS

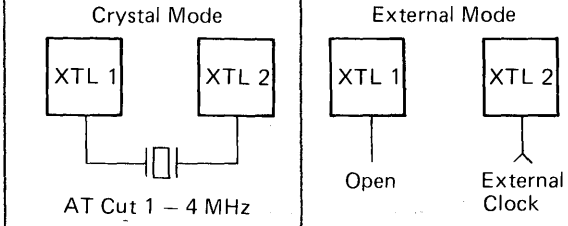
Figure 10

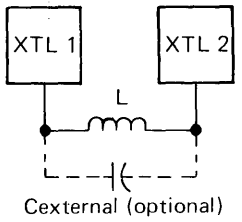


Minimum R = 4KΩ

C = 26.5pF ± 2.6pF + $C_{EXTERNAL}$







Minimum L = 0.1 mH

Minimum Q = 40

Maximum Cexternal = 30pF

C = 13pF ± 1.3pF + Cexternal

$$f \simeq \frac{1}{2\pi\sqrt{LC}}$$

Suggested Crystal Vendors

a) Electro-Dynamics
5625 Foxridge Drive
Mission, Kansas 66201
913-262-2500

b) CRYSTEK
1000 Crystal Drive
Ft. Myers, Florida 33901
813-936-2109

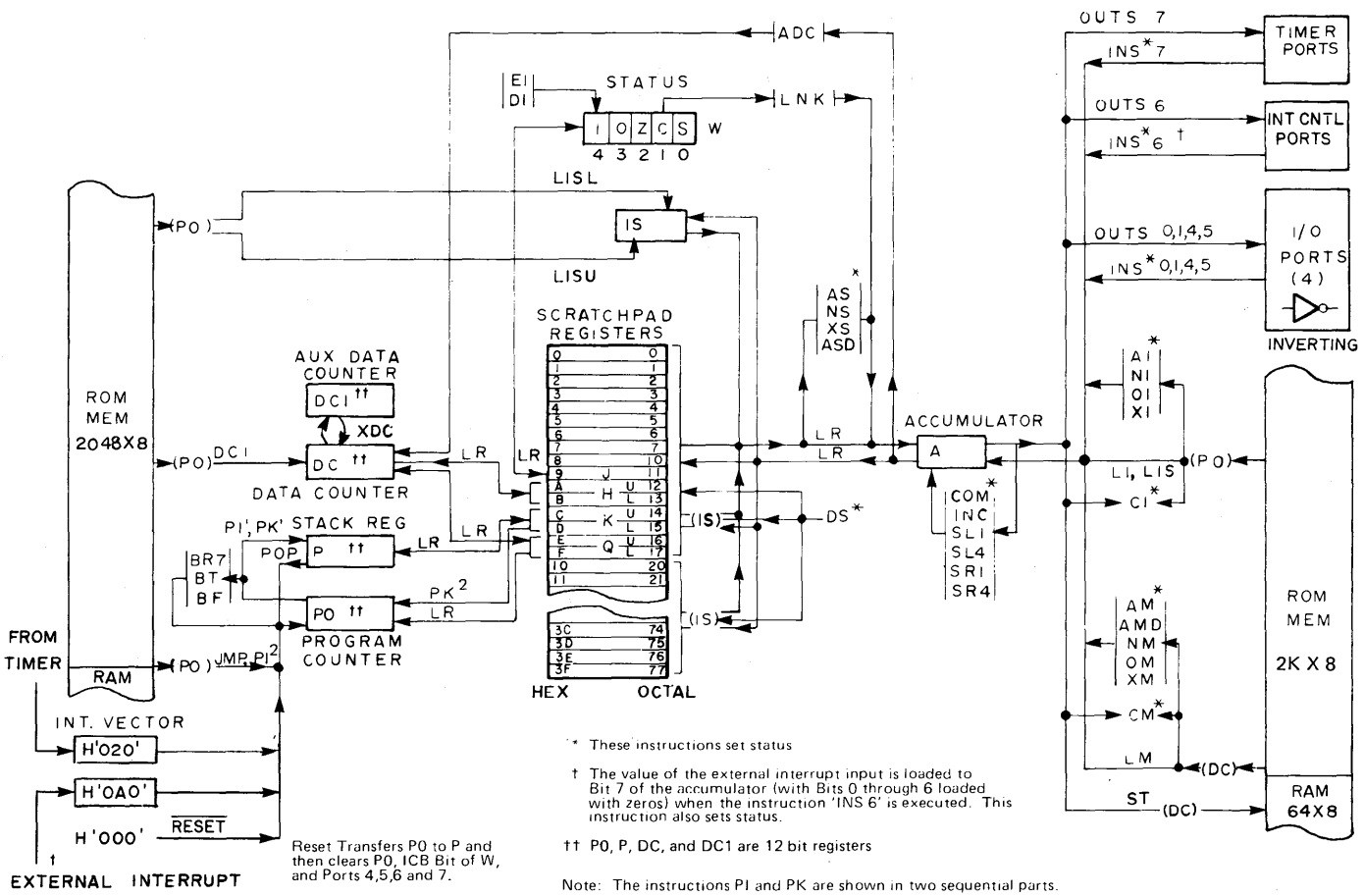
c) W.T. Liggett Corp.
1500 Worcester Rd.
Section 30
Framingham, MA 01701
617-620-1150

d) Erie Frequency Control
453 Lincoln Street
Carlisle, Penn 17013
717-249-2232

e) Electronic Crystals Corp.
1153 Southwest Blvd.
Kansas City, Kansas 66103
913-262-1274

f) M-TRON Industries
P.O. Box 630
100 Douglas Avenue
Yankton, South Dakota
605-665-9321

MK3876 PROGRAMMING MODEL
Figure 11



* These instructions set status

† The value of the external interrupt input is loaded to Bit 7 of the accumulator (with Bits 0 through 6 loaded with zeros) when the instruction 'INS 6' is executed. This instruction also sets status.

†† P0, P, DC, and DC1 are 12 bit registers

Note: The instructions PI and PK are shown in two sequential parts. (PI1, PI2 and PK1, PK2).

Reset Transfers P0 to P and then clears P0, ICB Bit of W, and Ports 4,5,6 and 7.

INSTRUCTION EXECUTION

This section details the timing and execution of the 3876 instruction set. The 3876 executes the entire F8 instruction set with exact F8 timing. Refer to Figure 11 for a 3876 Programming Model.

F8 INSTRUCTION SET

ACCUMULATOR GROUP INSTRUCTIONS

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	CYCLES LONG	μS (2MHzΦ)	STATUS BITS OVR	ZERO	CRY	SIGN
Add Carry	LNK		A←(A) + CRY	19	1	1		2	1/0	1/0	1/0	1/0
Add Immediate	AI	ii	A←(A) + H'ii'	24ii	2	1	1	5	1/0	1/0	1/0	1/0
And Immediate	NI	ii	A←(A)∧H'ii'	21ii	2	1	1	5	0	1/0	0	1/0
Clear	CLR		A←H'00'	70	1	1		2	—	—	—	—
Compare Immediate	CI	ii	H'ii'+ (A) + 1	25ii	2	1	1	5	1/0	1/0	1/0	1/0
Complement	COM		A←(A) + H'FF'	18	1	1		2	0	1/0	0	1/0
Exclusive or Immediate	XI	ii	A←(A) + H'ii'	23ii	2	1	1	5	0	1/0	0	1/0
Increment	INC		A←(A) + 1	1F	1	1		2	1/0	1/0	1/0	1/0
Load Immediate	LI	ii	A←H'ii'	20ii	2	1	1	5	—	—	—	—
Load Immediate Short	LIS	i	A←H' 0i'	7i	1	1		2				
OR Immediate	OI	ii	A←(A) V H 'ii'	22ii	2	1	1	5	0	1/0	0	1/0
Shift Left One	SL	1	Shift Left 1	13	1	1		2	0	1/0	0	1/0
Shift Left Four	SL	4	Shift Left 4	15	1	1		2	0	1/0	0	1/0
Shift Right One	SR	1	Shift Right 1	12	1	1		2	0	1/0	0	1
Shift Right Four	SR	4	Shift Right 4	14	1	1		2	0	1/0	0	1

BRANCH INSTRUCTIONS In all conditional branches P0←(P0) + 2 if the test condition is not met. Execution is complete in 3 short cycles.

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	CYCLES LONG	μS (2MHzΦ)	STATUS BITS OVR	ZERO	CRY	SIGN
Branch on Carry	BC	aa	P0←(P0) + 1 + H'aa' if CRY 1	82aa	2	2	1	7	—	—	—	—
Branch on Positive	BP	aa	P0←(P0) + 1 + H'aa' if SIGN 1	81aa	2	2	1	7	—	—	—	—
Branch on Zero	BZ	aa	P0←(P0) + 1 + H'aa' if Zero 1	84aa	2	2	1	7	—	—	—	—
Branch on True	BT	taa	P0←(P0) + 1 + H'aa' if any test is true	8taa	2	2	1	7	—	—	—	—
Branch If Negative	8M	aa	P0←(P0) + 1 + H'aa' if SIGN 0	91aa	2	2	1	7	—	—	—	—
Branch if No Carry	BNC	aa	P0←(P0) + 1 + H'aa' if CARRY 0	92aa	2	2	1	7	—	—	—	—
Branch if No Overflow	BNO	aa	P0←(P0) + 1 + H'aa' if OVR 0	98aa	2	2	1	7	—	—	—	—
Branch if Not Zero	BNZ	aa	P0←(P0) + 1 + H'aa' if ZERO 0	94aa	2	2	1	7	—	—	—	—
Branch if False Test	BF	taa	P0←(P0) + 1 + H'aa' if all false test bits	9taa	2	2	1	7	—	—	—	—
Branch if ISAR (Lower) //	BR7	aa	P0←(P0) + 1 + H'aa' if ISARL //	8Faa	2	2	1	5	—	—	—	—
			P0←(P0) + 2 if ISARL =		2	2		4	—	—	—	—
Branch Relative	BR	aa	P0←(P0) + 1 + H'aa'	90aa	2	2	1	7	—	—	—	—
Jump*	JMP	aaaa	P0←H'aaaa'	29aaaa	3	1	3	11	—	—	—	—

t TEST CONDITION

2^2	2^1	2^0
ZERO	CRY	SIGN

t TEST CONDITION

2^3	2^2	2^1	2^0
OVF	ZERO	CRY	SIGN

*Privileged instruction, Accumulator contents altered during execution JMP

MEMORY REFERENCE INSTRUCTIONS In all Memory Reference Instructions, the Data Counter is incremented DC ←(DC)+1

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	CYCLES LONG	μS (2MHz↓)	OVR	STATUS BITS ZERO	STATUS BITS CRY	STATUS BITS SIGN
Add Binary	AM		A←(A) + [(DC)]	88	1	1	1	5	1/0	1/0	1/0	1/0
Add Decimal	AMD		A←(A) + [(DC)] • BCD Adjust	89	1	1	1	5	1/0	1/0	1/0	1/0
AND	NM		A←(A) ∧ [(DC)]	8A	1	1	1	5	0	1/0	0	1/0
Compare	CM		[(DC)] + $\overline{(A)}$ + 1	8D	1	1	1	5	1/0	1/0	1/0	1/0
Exclusive OR	XM		A←(A)⊕[(DC)]	8C	1	1	1	5	0	1/0	0	1/0
Load	LM		A←[(DC)]	16	1	1	1	5	—	—	—	—
Logical OR	OM		A←(A) ∨ [(DC)]	8B	1	1	1	5	0	1/0	0	1/0
Store	ST		A→[(DC)]	17	1	1	1	5	—	—	—	—

ADDRESS REGISTER GROUP INSTRUCTIONS

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	CYCLES LONG	μS (2MHz↓)	OVR	STATUS BITS ZERO	STATUS BITS CRY	STATUS BITS SIGN
Add to Data Counter	ADC		DC←(DC) + (A)	8E	1	1	1	5	–	–	–	–
Call to Subroutine*	PK		P0U←(r12); P0L←(r13), P←(P0)	0C	1	1	2	8	–	–	–	–
Call to Subroutine Immediate*	PI	aaaa	P←(P0), P0←H'aaaa	28aaaa	3	2	3	13	–	–	–	–
Exchange DC	XDC		(DC)⇄(DC1)	2C	1	2		4	–	–	–	–
Load Data Counter	LR	DC,O	DCU←(r14), DCL←(r15)	0F	1	1	2	8	–	–	–	–
Load Data Counter	LR	DC'H	DCU←(r10), DCL←(r11)	10	1	1	2	8	–	–	–	–
Load DC Immediate	DCI	aaaa	DC←H'aaaa'	2Aaaaa	3	3	2	12	–	–	–	–
Load Program Counter	LR	P0,Q	P0U←(r14), P0L←(r15)	0D	1	1	2	8	–	–	–	–
Load Stack Register	LR	P,K	PU←(r12), PL←(r13)	09	1	1	2	8	–	–	–	–
Return from Subroutine*	POP		P←(P0)	1C	1	2		4	–	–	–	–
Store Data Counter	LR	Q,DC	r14←(DCU), r15←(DCL)	0E	1	1	2	8	–	–	–	–
Store Data Counter	LR	H,DC	r10←(DCU), r11←(DCL)	11	1	1	2	8	–	–	–	–
Store Stack Register	LR	K,P	r12←(PU); r13←(PL)	08	1	1	2	8	–	–	–	–

SCRATCHPAD REGISTER INSTRUCTIONS (Refer to Scratchpad Addressing Modes)

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	CYCLES LONG	μS (2MHz↓)	OVR	STATUS BITS ZERO	STATUS BITS CRY	STATUS BITS SIGN
Add Binary	AS	r	A←(A)+ (r)	Cr	1	1		2	1/0	1/0	1/0	1/0
Add Decimal	ASD	r	A←(A) + (r)	Dr	1	2		4	1/0	1/0	1/0	1/0
Decrement	DS	r	r←(r) + H'FF'	3r	1		1	3	1/0	1/0	1/0	1/0
Load	LR	A,r	A←(r)	4r	1	1		2	–	–	–	–
Load	LR	A, KU	A←(r12)	00	1	1		2	–	–	–	–
Load	LR	A, KL	A←(r13)	01	1	1		2	–	–	–	–
Load	LR	A, QU	A←(r14)	02	1	1		2	–	–	–	–
Load	LR	A, QL	A←(r15)	03	1	1		2	–	–	–	–
Load	LR	r, A	r←(A)	5r	1	1		2	–	–	–	–
Load	LR	KU, A	r12←(A)	04	1	1		2	–	–	–	–
Load	LR	KL, A	r13←(A)	05	1	1		2	–	–	–	–
Load	LR	QU, A	r14←(A)	06	1	1		2	–	–	–	–
Load	LR	QL,A	r15←(A)	07	1	1		2	–	–	–	–
And	NS	r	A←(A) ∧ (r)	Fr	1	1		2	0	1/0	0	1/0
Exclusive Or	XS	r	A←(A) + (r)	Er	1	1		2	0	1/0	0	1/0

*Privileged instruction, Accumulator contents altered during execution of PI instruction.

MISCELLANEOUS INSTRUCTIONS

OPERATION	MNEMONIC OP CODE	OPERAND	FUNCTION	MACHINE CODE	BYTES	CYCLES SHORT	CYCLES LONG	μS (2MHzΦ)	STATUS BITS OVR	ZERO	CRY	SIGN
Disable Interrupt	DI		RESET ICB	1A	1	1		2	–	–	–	–
Enable Interrupt *	EI		SET ICB	1B	1	1		2	–	–	–	–
Input	IN	04,05,06,07	A←(Input Port aa)	26aa	2	1	2	8	0	1/0	0	1/0
Input Short	INS	0, 1	A←(Input Port 0 or 1)	A0,A1	1	2		4	0	1/0	0	1/0
Input Short	INS	4,5,6,7	A←(Input Port a)	Aa	1	1	2	8	0	1/0	0	1/0
Load ISAR	LR	IS,A	IS←(A)	0B	1	1		2	–	–	–	–
Load ISAR Lower	LISL	bbb	ISL←bbb	6(0bbb)**	1	1		2	–	–	–	–
Load ISAR Upper	LISU	bbb	ISU←bbb	6(1bbb)**	1	1		2	–	–	–	–
Load Status Register *	LR	W,J	W←(r9)	1D	1	2		4	1/0	1/0	1/0	1/0
No Operation	NOP		P0←(P0) + 1	2B	1	1		2	–	–	–	–
Output *	OUT	04,05,06,07	Output Port aa←(A)	27aa	2	1	2	8	–	–	–	–
Output Short	OUTS	0, 1	Output Port 0 or 1←(A)	B0, B1	1	2		4	–	–	–	–
Output Short	OUTS	4,5,6,7	Output Port a←(A)	Ba	1	1	2	8	–	–	–	–
Store ISAR	LR	A,IS	A←(IS)	0A	1	1		2	–	–	–	–
Store Status Reg	LR	J,W	r9←(W)	1E	1	1		2	–	–	–	–

*Privileged instruction

**b = 1 bit immediate operand

NOTES.

Lower case denotes variables specified by programmer

Function Definitions

←	is replaced by
( )	the contents of
$\overline{( )}$	Binary "1's" complement of
+	Arithmetic Add (Binary or Decimal)
⊕	Logical "OR" exclusive
∧	Logical "AND"
∨	Logical "OR" inclusive
H' '	Hexadecimal digit
[( )]	Contents of memory specified by ( )
a	Address Variable (four bits)
A	Accumulator
b	One bit immediate operand
DC	Data Counter (Indirect Address Register)
DC1	Data Counter 1 (Auxiliary Data Counter)
DCL	Least significant 8 bits of Data Counter Addressed
DCU	Most significant 8 bits of Data Counter Addressed
H	Scratchpad Register 10 and 11
i	Immediate operand (four bits)
ICB	Interrupt Control Bit
IS	Indirect Scratchpad Address Register
ISL	Least Significant 3 bits of ISAR
ISU	Most Significant 3 bits of ISAR
J	Scratchpad Register 9
K	Registers 12 and 13
KL	Register 13
KU	Register 12
P0	Program Counter
P0L	Least Significant 8 bits of Program Counter
P0U	Most Significant 8 bits of Program Counter
P	Stack Register
PL	Least Significant 8 bits of Program Counter
PU	Most Significant 8 bits of Active Stack Register
Q	Registers 14 and 15
QL	Register 15
QU	Register 14
r	Scratchpad Register (any address 0 thru B) (See Below)
W	Status Register

Scratchpad Addressing Modes Using IS. ($r \neq 0$ thru B)

r=H'C'	Register Addressed by IS is (Unmodified)
r=H'D'	Register Addressed by IS is Incremented
r=H'E'	Register Addressed by IS is Decremented
r=H'F'	Illegal OP Code.

Status Register

—	No change in condition
1/0	is set to "1" or "0" depending on conditions
CRY	Carry Flag
OVR	Overflow Flag
SIGN	Sign of Result Flag
ZERO	Zero Flag

ELECTRICAL SPECIFICATIONS
ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias . . . . . 0°C to 70°C
Storage Temperature . . . . . −65°C to +150°C
Voltage on Any Pin With Respect To Ground (except open drain pins) . . . . . −1.0V to +7V
Voltage On Open Drain Pins . . . . . −1.0V to +13.5V
Power Dissipation 1.5W
Power Dissipated by any one I/O pin[4] 60mW
Power Dissipated by all I/O pins[4] 600mW

A.C. CHARACTERISTICS – See Figure 12 and 13 for Timing Diagrams

T_A = 0°C to 70°C, V_{CC} = 5V ± 10%, I/O POWER DISSIPATION ≤ 100mW

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	NOTES
XTL 1 XTL 2	$t_0(EX)$ $t_{EX(H)}$ $t_{EX(L)}$	Time base period, all external modes External Clock Pulse Width High External Clock Pulse Width Low	250 90 100	1000 700 700	ns ns ns	4MHz-1MHz
Φ	$t_Φ$	Internal Φ Clock Period	$2t_0$			
WRITE	tw	Internal WRITE Clock Period	$4t_Φ$ $6t_Φ$			Short Cycle Long Cycle
I/O	$t_{dI/O}$	Output delay from internal WRITE Clock	0	1000	ns	50pF plus one TTL load
	$t_{sI/O}$	Input Setup time to WRITE Clock	1000		ns	
$\overline{STROBE}$	$t_{I/O-s}$	Output valid to STROBE Delay	$3t_Φ$ -1000	$3t_Φ$ +250		I/O load = 50pF + 1 TTL STROBE Load= 50pF + 3 TTL
	tsl	$\overline{STROBE}$ Low Time	$8t_Φ$ -250	$12t_Φ$ +250	ns	
$\overline{RESET}$	t_{RH}	$\overline{RESET}$ Hold Time, Low	$6t_Φ$ +750		ns	
EXT INT	t_{EH}	EXT INT Hold Time, Active and Inactive State	$6t_Φ$ + 750		ns	To trigger interrupt
			$2t_Φ$			To trigger timer

TIMER AC CHARACTERISTICS

Definitions:

Error = Indicated time value - actual time value

tpsc = tΦ x Prescale Value

Interval Timer Mode:

Single interval error, free running (Note 3) ..... ±6tΦ
Cumulative interval error, free running (Note 3) 0
Error between two Timer reads (Note 2) ..... ±(tpsc + tΦ)
Start Timer to stop Timer error (Notes 1,4) ..... +tΦ to −(tpsc +tΦ)
Start Timer to read Timer error (Notes 1,2) ..... −5tΦ to −(tpsc + 7tΦ)
Start Timer to interrupt request error (Notes 1,3) ..... −2tΦ to −8tΦ
Load Timer to stop Timer error (Note 1) ..... +tΦ to −(tpsc + 2tΦ)
Load Timer to read Timer error (Notes 1,2) ..... −5tΦ to −(tpsc + 8tΦ)
Load Timer to interrupt request error (Notes 1,3) ..... −2t Φ to −9tΦ

Pulse Width Measurement Mode:

Measurement accuracy (Note 4) ..... +t Φ to −(tpsc +2t Φ)
Minimum pulse width of EXT INT pin 2tΦ

Event Counter Mode:

Minimum active time of EXT INT pin 2tΦ
Minimum inactive time of EXT INT pin 2tΦ

Notes:

1. All times which entail loading, starting, or stopping the Timer are referenced from the end of the last machine cycle of the OUT or OUTS instruction.
2. All times which entail reading the Timer are referenced from the end of the last machine cycle of the IN or INS instruction.
3. All times which entail the generation of an interrupt request are referenced from the start of the machine cycle in which the appropriate interrupt request latch is set. Additional time may elapse if the interrupt request occurs during a privileged or multicycle instruction.
4. Error may be cumulative if operation is repetitively performed.

CAPACITANCE

T_A = 25°C, f=2MHz

SYMBOL	PARAMETER	MIN	MAX	UNIT	NOTES
C_{IN}	Input Capacitance: I/O Ports, $\overline{RESET}$, $\overline{EXTINT}$, $\overline{RAMPRT}$, TEST		7	pF	Unmeasured Pins Grounded
C_{XTL}	Input Capacitance: XTL1, XTL2	20.5	32.5	pF	

DC CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = +5V ± 10%, I/O POWER DISSIPATION ⩽100mW

SYMBOL	PARAMETER	MIN	MAX	UNIT	TEST CONDITIONS
I_{CC}	Power Supply Current		93	mA	Outputs Open
P_D	Power Dissipation		440	mW	Outputs Open

DC CHARACTERISTICS (Cont'd)

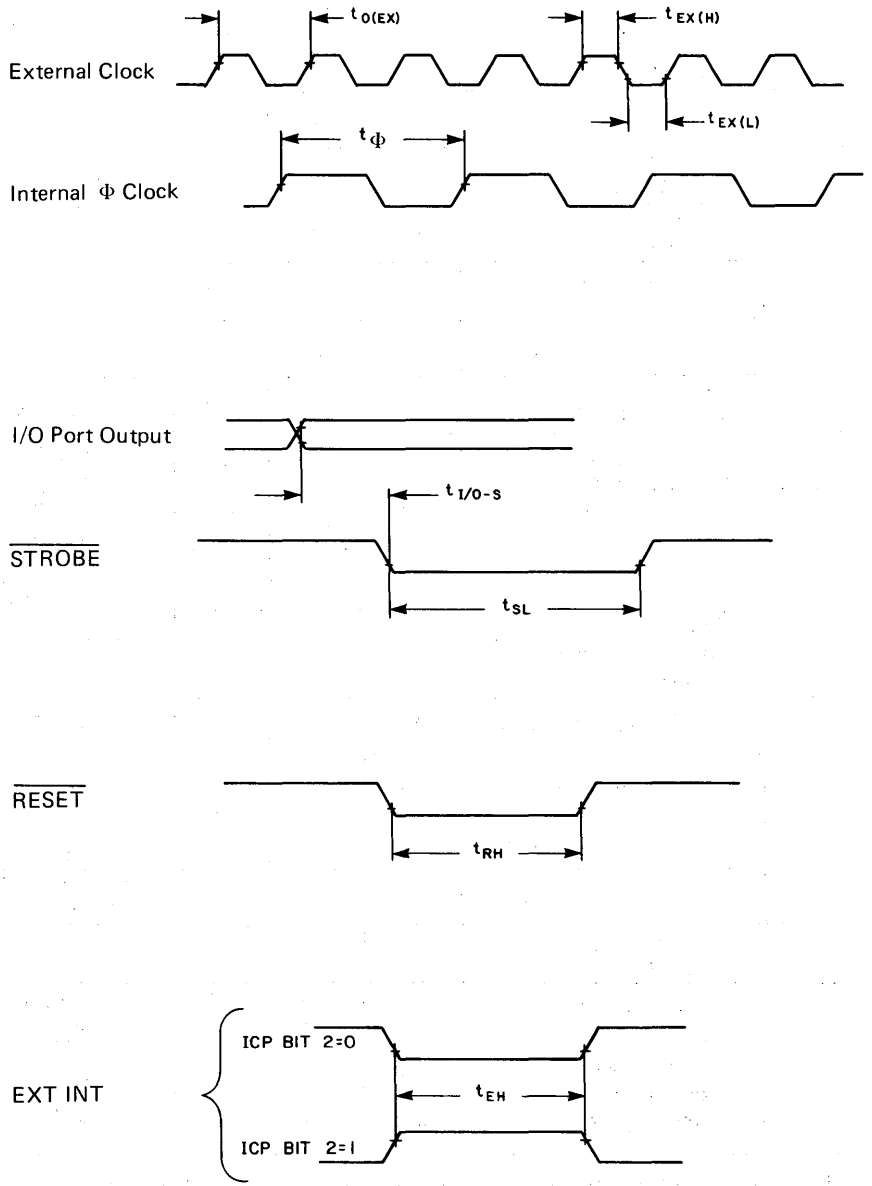
SYMBOL	PARAMETER	MIN	MAX	UNIT	NOTES
V_{IHEX}	External Clock Input High Level	2.4	5.8	V	
V_{ILHEX}	External Clock Input Low Current	−0.3	0.6	V	
I_{IHEX}	External Clock Input High Current		100	µA	$V_{IHEX} = V_{CC}$
I_{ILEX}	External Clock Input Low Current		-100	µA	$V_{ILEX} = V_{SS}$
V_{IH}	Input High Level Ports, $\overline{RESET}$[1], EXT INT[1]	2.0	5.8	V	
V_{IHOD}	Open Drain Input High Level	2.0	13.2	V	
V_{IL}	Input Low Level Ports, $\overline{RESET}$[1], EXT INT[1]	−0.3	0.8		
I_{IL}	Input Low Current Ports, $\overline{RESET}$[2], EXT INT[2]		−1.6	mA	V_{IL}=0.4V
I_L	Leakage Current Open drain ports, RAMPRT $\overline{RESET}$[3], EXT INT[3]		+10 −5	µA	V_{IN}=13.2V V_{IN}=0.0V
I_{OH}	Output High Current Standard ports, $\overline{RESET}$[2] EXT INT[2]	−100 −30		µA µA	V_{OH}=2.4V V_{OH}=3.9V
I_{OHDD}	OUTPUT High Current Direct Drive Ports	−0.1 −1.5 	 −8.5	mA mA mA	V_{OH} = 2.4V V_{OH}=1.5V V_{OH}=.7V
I_{OL}	Output Low Current IO ports	1.8		mA	V_{OL}=0.4V
I_{OHS}	$\overline{STROBE}$ Output High Current	−300		µA	V_{OH}=2.4V
I_{OLS}	$\overline{STROBE}$ Output Low Current	5.0		mA	V_{OL} = 0.4V
V_{IHRPR}	$\overline{RAMPRT}$ Input High Level	1.9	5.8	V	Guaranteed .1V less than V_{IH} for $\overline{RESET}$
V_{ILRPR}	$\overline{RAMPRT}$ Input Low Level	−0.3	0.4	V	Guaranteed .1V less than V_{IL} for $\overline{RESET}$
V_{SB}	Standby V_{CC} for RAM	2.2	5.5	V	
I_{SB}	Standby current		6 3.7	mA mA	V_{SB} = 5.5V V_{SB} = 2.2V
I_{CHARGE}	Trickle charge available on V_{SB} with V_{CC}=4.5 to 5.5	-4 -4.5	-12 -15	mA mA	V_{SB}=2.8V $\overline{RAMPRT}$ high V_{SB}=2.2V

* Stresses above 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 condition 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.

1. $\overline{RESET}$ and EXT INT have internal Schmit triggers giving minimum .2V hysteresis.
2. $\overline{RESET}$ or EXT INT programmed with standard pull-up
3. $\overline{RESET}$ or EXT INT programmed without standard pull-up
4. Power dissipation for I/O pins is calculated by $\Sigma(V_{CC} - V_{IL})(|I_{IL}|) + \Sigma(V_{CC} - V_{OH})(|I_{OH}|) + \Sigma(V_{OL})(I_{OL})$

AC TIMING DIAGRAM

Figure 12

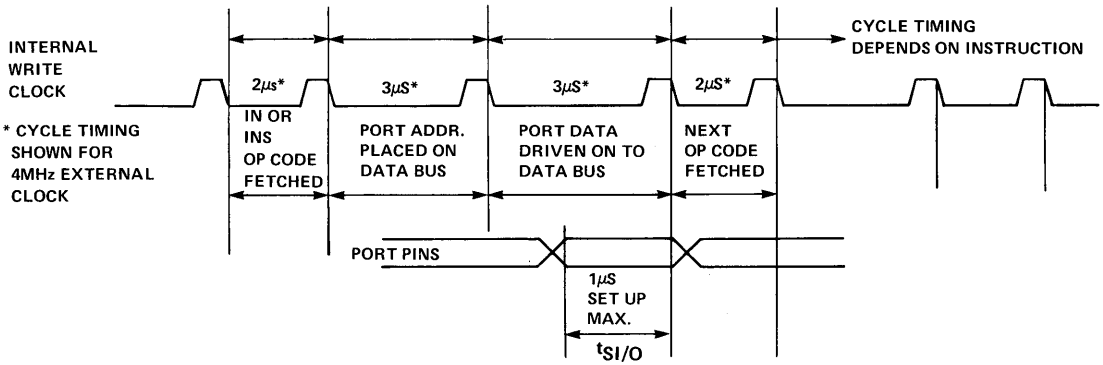


Note: All measurements are referenced to V_{IL} max., V_{IH} min., V_{OL} max., or V_{OH} min.

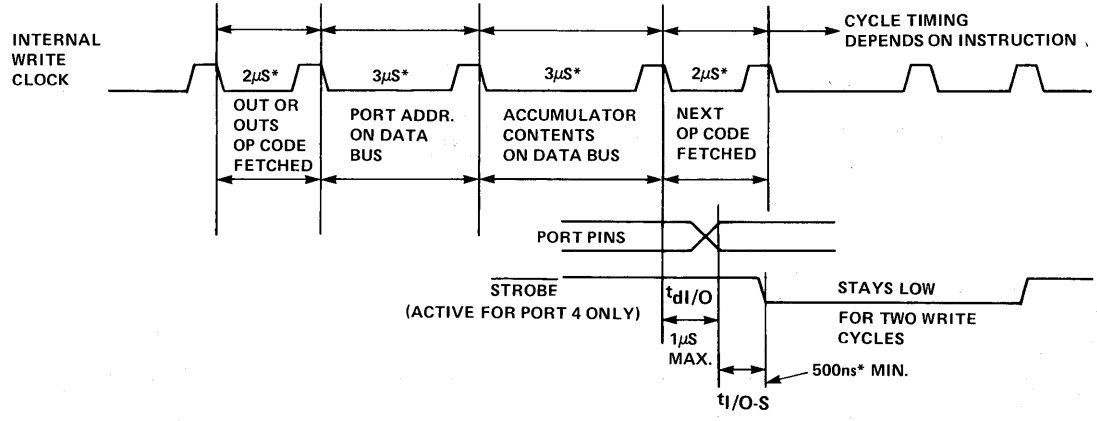
INPUT/OUTPUT AC TIMING

Figure 13

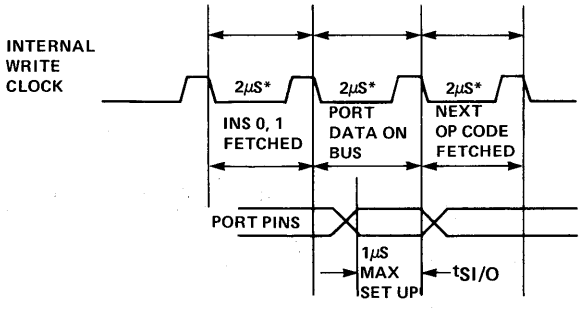
A. INPUT ON PORT 4 OR 5



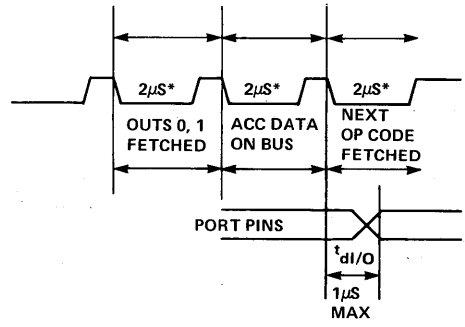
B. OUTPUT ON PORT 4 OR 5



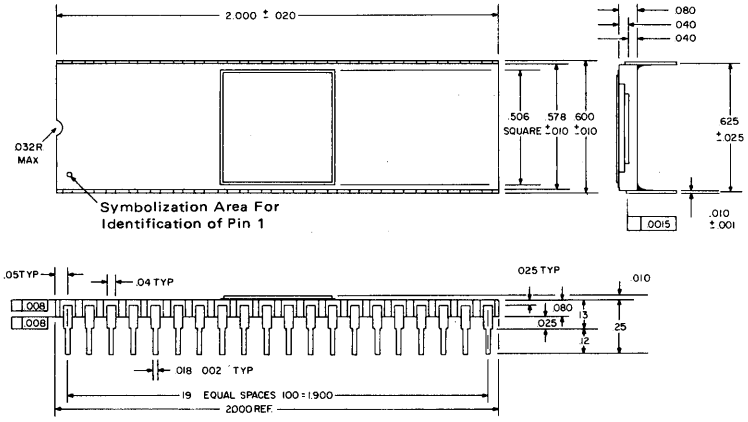
C. INPUT ON PORT 0 OR 1



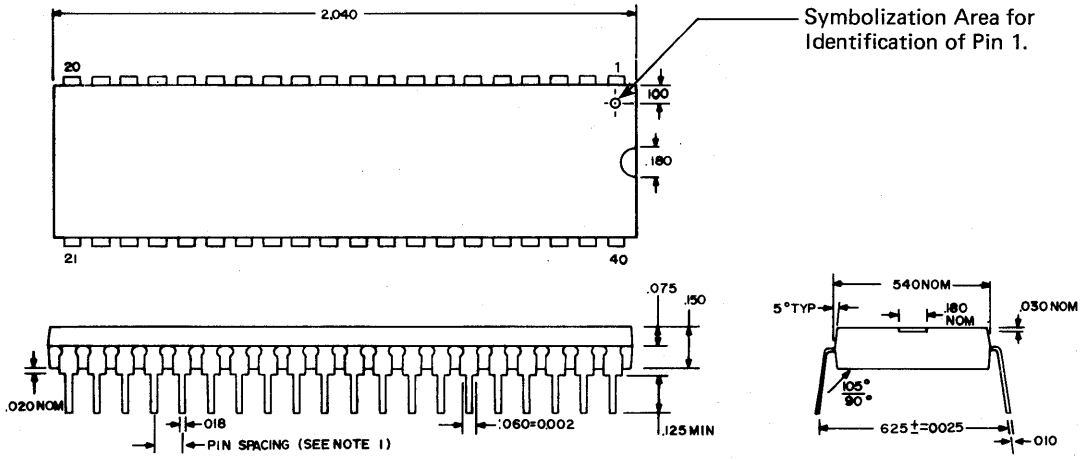
D. OUTPUT ON PORT 0, 1



PACKAGE DESCRIPTION: 40-Pin Dual In-Line Ceramic Package



PACKAGE DESCRIPTION 40-Pin Dual-in-Line Plastic Package



ORDERING INFORMATION

PART NO.	PACKAGE TYPE	TEMPERATURE RANGE
MK3876(N)/17XXX	Plastic	0°C to +70°C
MK3876(P)/17XXX	Ceramic	0°C to +70°C

APPENDIX A

ORDERING INFORMATION

CUSTOM MK 3876 OPTION SPECIFICATIONS

The custom MK3876 program may be transmitted to MOSTEK in any of the following media, listed in order of preference:

1) PROMs from the EMU-72
2) Punched paper tape
3) AID-80F Flexible Disk
4) Card Deck (IBM 80 column cards)

The program may be specified in the following forms:

PROMS with correct object code in each location

OBJECT CODE produced by one of MOSTEK's assemblers:

XFOR-50/70 Fortran IV Cross Assembler, SDB-50/70 resident assembler (ASMB-50/70), AID-80F F8 Cross-Assembler (FZCASM)

OBJECT CODE produced by the dump command from any of MOSTEK's F8 development hardware (SDB-50/70, AID-80F).

DATA DECK FORMAT as described in the Data Deck section

A completed cover letter (See page 31) must be attached. The information should be properly packed and mailed prepaid and insured to:

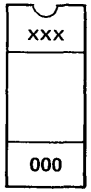
MOSTEK Corporation
Microcomputer Product Marketing
1215 West Crosby Road
Carrollton, Texas 75006

A second copy of the cover letter should be mailed separately to the above address.

PROMS

A 2716 type PROM, (5 volt only) programmed with the customer program (positive logic sense for addresses and data) may be submitted. See Fig. A-2 for marking. Include a three-letter customer ID on each PROM. After the PROM is removed from the EMU-72, it must be placed in a conductive IC carrier and securely packed.

Figure A-1



XXX = Customer ID

PAPER TAPE

Punched paper tapes (1" wide, 8 level ASCII) will be accepted. The tape must contain the absolute object output from the above mentioned F8 assemblers Paper. Object tapes in absolute format generated by the "D" (dump) command of DDT-2 or the dump command of the AID-80F (F8 debug option) are also acceptable if the entire memory space is dumped continuously. Tapes may also be punched using the DATA DECK FORMAT. They must contain 80 characters per record with a CR (carriage return) and LF (line feed) separating each record. The tape must be clearly labeled with customer name, and format used. Fan fold tape is preferred. Tape transparency should be limited to 60% transmissivity (40% opaque). Specifically, thin yellow or white tape is error prone on photo-electric readers and must not be used.

FLEXIBLE DISKS

FLEXIBLE DISKS (Floppy Disks) produced on the MOSTEK AID-80F development station may be submitted. The format must be the absolute object output from the assemblers, or an object dump using the memory dump command (F8 Debug Option). The disk must be clearly labeled with the format of the data (object, or object dump) and the customer's name.

PUNCHED CARD DECK

Standard 80 column punched cards must be used. They must be punched in IBM 029 code. The deck must contain two ty] e of cards:

COMMENT CARDS
DATA CARDS

3876 ORDERING INFORMATION

DATE _____

CUSTOMER NAME _____ CUSTOMER PO NUMBER _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

COUNTRY _____

PHONE _____ EXTENSION _____

CONTACT _____

CUSTOMER PART NUMBER _____

OPTIONS:

EXTERNAL INTERRUPT:	Pull-Up ☐	No Pull-Up ☐
RESET:	Pull-Up ☐	No Pull-Up ☐
STANDBY OPTION:	Yes ☐	No ☐

(Standby Power Option available only on the 3872 and 3876)

PORT OPTIONS:	STANDARD TTL	DRAIN OPEN	DRIVER PULL-UP
P4-0---	☐	☐	☐
P4-1---	☐	☐	☐
P4-2---	☐	☐	☐
P4-3---	☐	☐	☐
P4-4---	☐	☐	☐
P4-5---	☐	☐	☐
P4-6---	☐	☐	☐
P4-7---	☐	☐	☐
P5-0---	☐	☐	☐
P5-1---	☐	☐	☐
P5-2---	☐	☐	☐
P5-3---	☐	☐	☐
P5-4---	☐	☐	☐
P5-5---	☐	☐	☐
P5-6---	☐	☐	☐
P5-7---	☐	☐	☐

PATTERN MEDIA:

☐ PROMS
(Customer can send in two extra PROM'S, MOSTEK will program the customer's code on these PROM'S for code verification in the Emulator-70.)

☐ PAPER TAPE (DATA DECK)

☐ PAPER TAPE (OBJECT)

☐ CARD DECK (DATA DECK)

☐ DISKETTE (OBJECT)

THESE ITEMS MAY AFFECT COST

BRANDING REQUIREMENT (If any, 10 Alpha-numeric digits allowed)

_____

_____

PROTOTYPE QUANTITY (10 pieces at no charge - higher quantity extra charge)

_____

WAIVE PROTOTYPES (Customer accepts liability for all work in process)

Yes_____ No_____

SIGNATURE _____

TITLE_____

COMMENT CARDS

Comment Cards must have an asterisk (*) in column 1. The remaining 79 columns may be any character. Comment Cards may be placed anywhere throughout the data deck.

DATA CARDS

These cards specify the actual ROM data. All fields are right justified.

COLUMN 1:	C (the letter C)
COLUMN 2-9:	ADDR
COLUMN 10-12:	BYTE
COLUMN 14-16:	DATA 1
COLUMN 17-19:	DATA 2
COLUMN 20-22:	DATA 3
.	
.	
.	
COLUMN 76-78:	DATA 21
COLUMN 77-79:	DATA 22 or SEQUENCE NUMBER

ADDR is the address of the first byte of data (DATA 1) contained on that card. Successive data bytes read from that card will be placed in successively greater address locations. BYTE is the number of data bytes to be read from that card (1 to 22).

If sequence numbers are used, the maximum number of bytes per card is 21. The base for ADDR and BYTE may be either decimal or hex but both must be the same. Data may be either in decimal or hex regardless of the base used for ADDR and BYTE. The base for sequence numbers (if they are used) is always decimal. The bases must be consistant throughout the deck. Data cards need not occur in order of increasing or decreasing addresses. Any unspecified address will be filled with zero. Any unpunched field will be read as a zero. If two data cards specify data for the same address, the one encountered second in the deck will override the first.

A portion of an example deck is shown.

```
*  3876 DATA DECK
*  MOSTEK CORP, EXAMPLE APPLICATION
*  ADDR/BYTE ARE IN DECIMAL
*  DATA IS IN HEX
C  0    8  20  FF  OB  54  34  56  71  B6
C  8    8  1B  28  03  F3  4C  25  2E  94
C  16   8  04  29  01  00

*  START OF SUBROUTINE ALPHA

C  1096   4  20  32  7C  53
C  1100   4  52  47  29  06
C  1104   1  07
```

VERIFICATION MEDIA

All original pattern media (PROMs, paper tape, etc.) are filed for contractural purposes and are not returned. Two copies of computer listings printed during the creation of the custom mask pattern are returned. One copy may be kept by the customer. The other copy should be checked thoroughly, signed, and returned to MOSTEK. The signed listing constitutes the contractual agreement for creation of the custom mask. Though the computer listing serves as the actual verification media, MOSTEK will program 2716 PROMs programmed from the data file used to create the custom mask to aid in the verification process. If programmed PROMs are desired, two blank 2716 type PROMs must be provided by the customer.

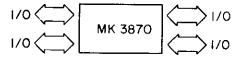
F8 DATA SHEETS

F8 Central Processing Unit MK 3850

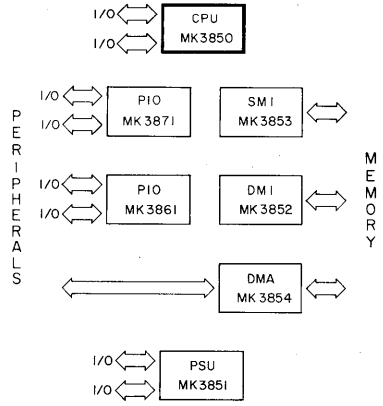
FEATURES

- ☐ N-channel Isoplanar MOS technology
- ☐ 2 μs cycle time
- ☐ 64 byte RAM on the CPU chip
- ☐ Two bi-directional, 8-bit I/O ports
- ☐ 8-bit arithmetic and logic unit, supporting both binary and decimal arithmetic
- ☐ Interrupt control logic
- ☐ Both external and crystal clock generating modes
- ☐ Over 70 instructions
- ☐ Low power dissipation—typically less than 330mW

SINGLE CHIP MK3870



F8 FAMILY

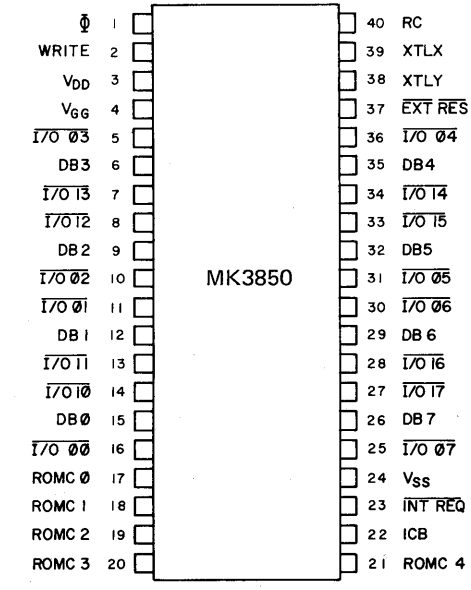


GENERAL DESCRIPTION

The MK3850 is the Central Processing Unit (CPU) for the F8 Microprocessor family. It is used in conjunction with other F8 family devices to configure the optimal microprocessor system for the amount of RAM, ROM/PROM, and I/O required in the users application. A minimum system may be configured with as few as two devices (CPU & PSU), while larger systems may have up to 64K bytes of memory, 128 I/O ports, direct memory acccess, and even multiple processors. Single chip microcomputer systems are also possible using the MK3870

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
$\overline{\text{EXT RES}}$	External Reset	Input
$\overline{\text{INT REQ}}$	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

PIN CONNECTIONS



FUNCTIONAL PIN DEFINITION

Φ and WRITE are clock outputs which drive all other devices in the F8 family.

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.

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.

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.

$\overline{I/O\ 00}$ through $\overline{I/O\ 07}$ and $\overline{I/O\ 10}$ through $\overline{I/O\ 17}$ are Input/Output port bits through which the CPU communicates with logic external to the microprocessor system.

$\overline{EXT\ RES}$ may be used to externally reset the system. When this line is pulled low, the program counter is set to address H '0000'.

$\overline{INT\ REQ}$ 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 $\overline{INT\ REQ}$ low. The CPU acknowledges interrupt requests by outputting appropriate ROMC signal sequences.

$\overline{ICB}$ indicates whether or not the CPU is currently ignoring the $\overline{INT\ REQ}$ line. If $\overline{ICB}$ is low, the CPU will respond to interrupt requests, if $\overline{ICB}$ is high, the CPU will ignore interrupt requests.

RC is not used and should be connected to V_{SS} for normal operation.

V_{SS} = OV
V_{DD} = +5V ± 5% @ 80mA max.
V_{GG} = +12V ± 5% @ 25mA max.

CPU ORGANIZATION

This section describes the basic functional elements of the MK3850 CPU. These elements are shown on the Functional Block Diagram of the CPU in Figure 3.

Instruction Register (IR)

The Instruction Register stores the instruction operation code during the instruction execution sequence. The OP Code is loaded into the Instruction Register from the data bus at the end of the execution sequence for the previous instruction. The last operation associated with each instruction is therefore the fetch of the OP code for the next instruction to be executed (unless an interrupt initiates the interrupt service sequence). The newly fetched OP code is latched into the Instruction Register at the start of the next machine cycle (as defined by the 1-0 transistion of the WRITE clock).

Most OP codes are either 4 or 8 bits long. For those instructions where the OP code may be completely specified using the upper 4 bits of the machine instruction, the lower 4 bits are used to specify an operand. This operand may specify a Scratch Pad Register, Port, or a 4-bit Immediate Constant. For this reason, the lower 4 bits of the instruction register are bussed to both the Scratch Pad Register Select logic and the Right Multiplexer Bus.

Control Unit

The Control Unit for the CPU consists of the Control ROM (CROM) and the State Counter. The CROM is responsible for generating all system timing and control signals required for controlling data flow within the F8 CPU and other F8 circuits.

The inputs to the CROM logic are the 8 bits from the instruction register, 4 bits from the State Counter, three internal status signals ("ALU RESULT = 0", "ISARL = 7", and the status of the Interrupt Control Bit (ICB) and two external conditions (INT REQ and Reset).

The IR inputs to the control logic identify which instruction is being executed, while the State Counter inputs define the machine cycle within the instruction execution sequence. The status of the ICB together with $\overline{INT\ REQ}$ are used to determine whether the interrupt sequence is to be initiated in lieu of fetching a new instruction. The reset input initiates the restart sequence. The remaining two internal signals are used to make branching decisions.

The outputs generated by the control logic fall into three groups.

- External Commands
- Next State Outputs
- Internal Commands

External commands are coded into the 5 system control lines (ROMC0 - ROMC4). Descriptions of these commands are shown in Table 2.

The next state outputs are 4 signals representing the next state of the State Counter. These signals are decoded during the present machine cycle and are strobed into the State Counter at the start of the next cycle. At that time these signals become the present state inputs to the CROM from the State Counter and new next State Outputs are generated.

The internal commands control data flow within the F8 CPU circuit. These commands include selecting the ALU operation to be performed, gating the proper input onto the Left and Right Multiplexer Busses, gating the Result Bus into the proper register or onto the Data Bus, selecting the proper Scratchpad Address input (either the ISAR or the lower 4 bits of the IR), and providing a signal to the timing circuits to force either a long or a short cycle.

Arithmetic And Logic Unit (ALU)

The 8-bit parallel ALU is the heart of the CPU. After receiving commands from the control circuits on the CPU circuit, the ALU performs the required arithmetic or logic operations (using the data presented on the two input busses) and provides

the result on the Result Bus. The arithmetic operations that can be performed in the ALU are binary add, decimal adjust, add with carry, decrement, and increment. The logic operations that can be performed are "AND", "OR", "EXCLUSIVE OR", and "1's COMPLEMENT". Associated with the left input port to the ALU is a shifter, a complementer, and a low order carry (C_0). The shifter can shift the left Multiplexer Bus to the left or to the right by 1 or 4 bits. The complementer can perform the 1's complement of the left Multiplexer Bus before providing it as an input to the ALU. C_0 participates whenever the ALU performs the add with carry operation. Normally it is a zero, but may be forced to a 1 or may take the state of the carry bit in the W register. Besides providing the result on the Result Bus, the ALU also provides four signals representing the status of the result. These signals, stored in the Status (W) register, represent carry, overflow, sign and zero condition of the result of the operation. The Zero condition is also used by the control circuits during execution of the branch instructions. In addition to performing arithmetic or logic operations, the ALU sometimes acts simply as a passage way to allow the contents of the various internal registers to be placed on the Result Bus so that they may be transferred to another register. For example, when the W register is stored in the Scratchpad, it first passes unaltered through the ALU on to the Result Bus, then into the Scratchpad register.

The Accumulator

The Accumulator is the principle register for data manipulations within the CPU. Using the ALU, the 8-bit contents of the Accumulator may be complemented, incremented, or shifted left or right. Its contents may also be logically or arithmetically combined with the contents of the Scratchpad or memory locations, with the result replacing the original contents of the Accumulator.

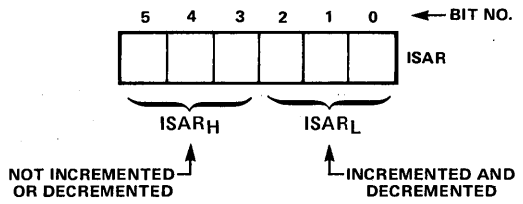
The Scratchpad And ISAR

The Scratchpad consists of 64 8-bit RAM data registers (H'00' thru H'3F') which are available to the programmer for the high speed access and manipulation of data. For most control/logic replacement this will provide all the data storage required.

All of the 64 Scratchpad registers are indirectly accessable through the use of the 6-bit Scratchpad address register, ISAR. In this way, any scratchpad register may be loaded to/from or added to the accumulator (binary or BCD); logically 'ANDED' or 'exclusive OR'ED' with the Accumulator; or decremented directly without disturbing the Accumulator. The contents of the least significant 3-bits of ISAR may be selectively auto-incremented, auto-decremented, or left unchanged (at the programmer's option) whenever the Scratchpad is accessed using ISAR (see Figure 1).

ISAR itself may be loaded either to/from the lower 6-bits of the accumulator, or loaded in 3-bit halves using the single byte immediate instructions LISU n and LISL n. The ability to independently modify the upper and lower halves of ISAR plus the auto-increment/auto-decrement options, can be used very effectively by the programmer to minimize the size of his programs.

FIGURE 1 – THE ISAR REGISTER



Additional saving may be further achieved by utilizing another key feature of the Scratchpad which permits the direct access of registers H'0' through H'B'. These registers should be reserved by the programmer for those variables most frequently accessed.

Scratchpad registers H'9' through H'F' (0 11' through O'17') have special significance since they have linkages directly with the status word (W), the Data Counter (DC), Stack Register (P) and Program Counter (P0) as shown in the F8 Programming Model (Figure 7). These linkages are implemented using single byte F8 instructions such as:

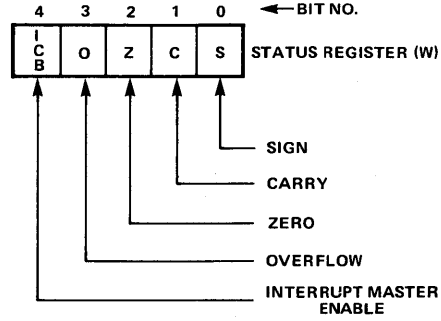
LR K,P

which transfers the 16-bit contents of the Stack Register (P) into the 'K' register pair (Scratchpad registers H'C' and H'D'). The contents of the accumulator are undisturbed by the execution by these instructions.

The Status Register

The status register (also called the W register) holds five status flags as shown in figure 2.

FIGURE 2 – THE STATUS REGISTER



Note that status flags are selectively modified following execution of different instructions. Table 4 defines the way in which individual F8 instructions modify status flags.

Sign (S BIT)

When the results of an ALU operation are being interpreted 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 7 6 5 4 3 2 1 0 ←Bit Number
Accumulator contents:   0 1 1 0 0 1 0 1
Value added:            0 1 1 1 0 1 1 0
Sum:            0       1 1 0 1 1 0 1 1
```

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

```
                C 7 6 5 4 3 2 1 0 ←Bit Number
Accumulator contents:   1 0 0 1 1 1 0 1
Value added:            1 1 0 1 0 0 0 1
Sum:            1       0 1 1 0 1 1 1 0
```

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 (O 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 O bit. After arithmetic operations, the O bit is set to the Exclusive-OR of carries out of bits 6 and bits 7. This simplifies signed binary arithmetic and is described in the Guide to Programming the F8. Here are some examples:

```
                  7 6 5 4 3 2 1 0 ←Bit Number
Accumulator contents:   1 0 1 1 0 0 1 1
Value Added:            0 1 1 1 0 0 0 1
Sum:                  1 0 0 1 0 0 1 0 0
                    1 ←┘
```

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

```
                  7 6 5 4 3 2 1 0 ←Bit Number
Accumulator contents:   0 1 1 0 0 1 1 1
Value Added:            0 0 1 0 0 1 0 0
Sum:                    1 0 0 0 1 0 1 1
                          ←┘
```

There is a carry out of bit 6, but no carry out of bit 7; the O 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, however interrupts are allowed only when the ICB bit is set to 1.

TABLE 1 – 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	=	$\overline{ALU_7}$

External Reset

When the EXT RES (External Reset) signal is pulled low and then returned high, the Program Counter (P0) is set to 0, causing the program origined at memory location 0 to be executed. The Interrupt Control status bit is also set low, inhibiting interrupt acknowledgement. The system is locked in an idle state while EXT RES is held low.

Timing Circuit

The timing circuit generates all the timing signals for the entire microcomputer. The two primary timing signals are Φ and WRITE. The Instruction Execution Sequence for each instruction is timed with these signals. The falling edge of WRITE marks the beginning of a new machine cycle, while Φ is used to time the length of the individual machine cycles.

A machine cycle is either 4 or 6 Φ periods long, with all instructions requiring between 1 and 5 machine cycles to complete their execution sequence.

The Data Bus

The Data Bus is used for transfering all address and data information between F8 System components. This includes Port Addresses, Memory Addresses, Read/ Write Memory Data, and Input/Output Port Data. Memory Address transfers are accomplished using two successive 8 bit transfers to complete the 16-bit Memory Address. The three conditions requiring Memory Address transfers are:

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 P0 registers and associated scratchpad registers.
3. During the interrupt acknowledge sequence, when the interrupt vector is loaded into P0.

I/O Ports

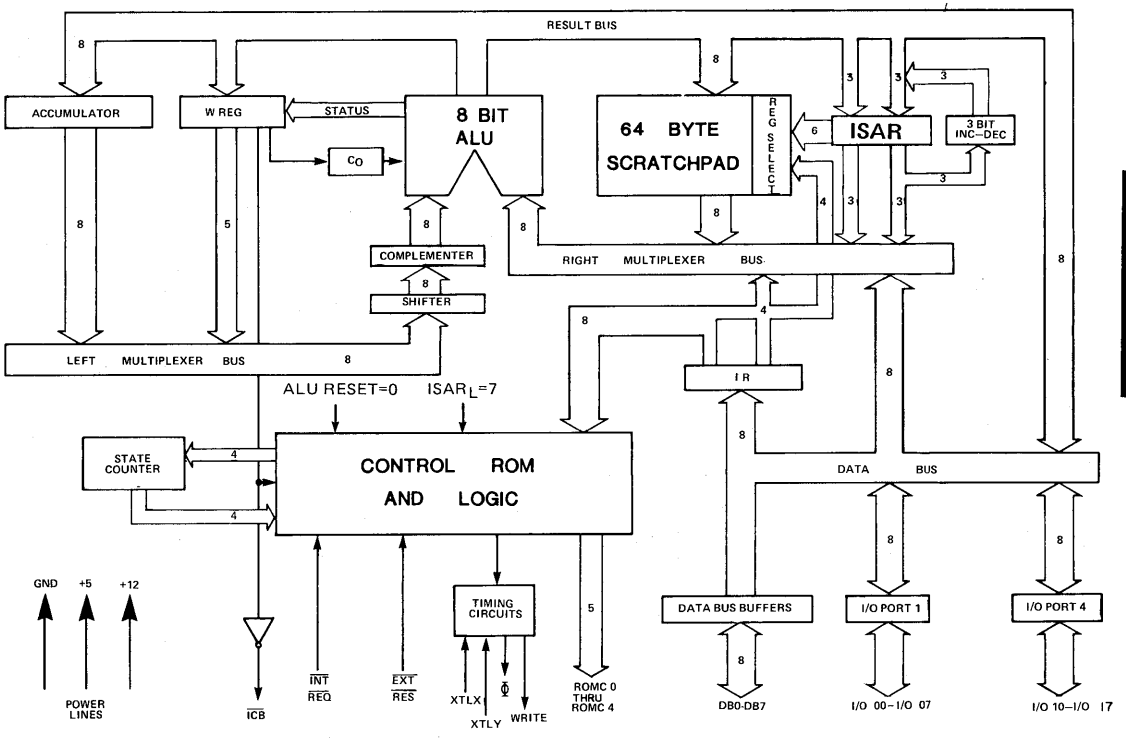
The 16 address pins which most microprocessors require are used by the 3850 for two I/O ports. Data may be transferred, via these two I/O ports, between the 3850 CPU and logic external to the microprocessor system.

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.

FIGURE 3 – MK 3850 CPU FUNCTIONAL DIAGRAM



INSTRUCTION EXECUTION SEQUENCE

All instructions are composed of long machine cycles (six Φ periods) and/or short machine cycles (four Φ periods). The long cycle is sometimes referred to as 1.5 cycles. Figure 8 illustrates the short cycle (PW_S) and the long cycle (PW_L). Observe that WRITE high appears at the end of each machine cycle.

The simplest instructions of the F8 instruction set execute in one short cycle while the most complex instruction (PI) requires two short cycles plus three long cycles. Every instruction's execution sequence ends with the next instruction OP code being fetched from memory. The OP code is loaded into the CPU's instruction register where it is decoded by the CPU's Control Unit.

The only instructions which may be executed in a single cycle are those which do not require the use of the Data Bus. This permits the Data Bus to be used to fetch the next instruction OP code simultaneously with the performance of the operation indicated by the current OP code. ROMC state 0 is used to specify the machine cycle during which a fetch is occurring, and therefore is used for all one cycle instructions.

Other instructions require more than one cycle to execute and use different ROMC states to specify the operation to be performed during each of the required cycles. The last cycle of each instruction, however, will always be the ROMC state 0 in order that the next OP code may be fetched.

The ROMC control signals are brought externally to the CPU itself in order to coordinate those operations which affect the memory referencing registers located on F8 devices other than CPU. Among these registers are the Program Counter, Stack Register and Data Counter. Most of the ROMC control states indicate those operations involving the contents of these registers, as shown in Table 2.

There are four different devices in the F8 Microprocessor family which contain the set of previously mentioned system registers (Program Counter, Stack Register, and Data Counter). These are the MK3853 SMI, MK3852 DMI, MK3851 PSU, and MK3871 PIO. Every F8 microprocessor system must contain at least one of these devices in addition to the MK3850 CPU. For those systems incorporating more than one of these devices, the resultant duplication of the Program Counter, Stack Register, and Data Counter is completely transparent to the user. This is accomplished since each device in the system receives the ROMC signals from the CPU and thus remains synchronized with all other devices.

INTERRUPTS

The Interrupt service sequence is initiated as the result of some other F8 device pulling the interrupt request (INT REQ) input to the CPU to V_{SS}. The interrupt service sequence begins during the last machine cycle of the first non-priviledged instruction to be executed after the interrupt request occurs. This is accomplished by modifying the ROMC state of the last machine cycle (which normally must be state 0 for the next OP code fetch) from state 0 to state 10 (Hex). Those instructions whose last machine cycle (ROMC state 0) is protected from being preempted by an interrupt request (and hence modified to ROMC state 10) are called PRIVILEGED instructions. These instructions are distinguished by the presence of an 'X' in the 'Interrupt' column of the instruction summary table (Table 4). The remainder of the interrupt service sequence requires three long and one short machine cycles as specified in Table 4. During this time, the high and low bytes of the Vector address from the interrupting device are transferred (via the Data Bus) into the Program Counter(s) and the Interrupt Control Bit (Bit 4 of the Status Register) is cleared to zero.

The response time for acknowledging an interrupt request can vary from 26 to 29 Φ periods if it is assumed that the CPU is executing a sequence of short cycle, non-privileged instructions during the time the interrupt request occurs (the minimum Φ period is 500 nS). The response time is defined as the duration from the 1-0 transition of INT REQ/ to the beginning of the execution sequence of the instruction stored at the Vector Address location in memory.

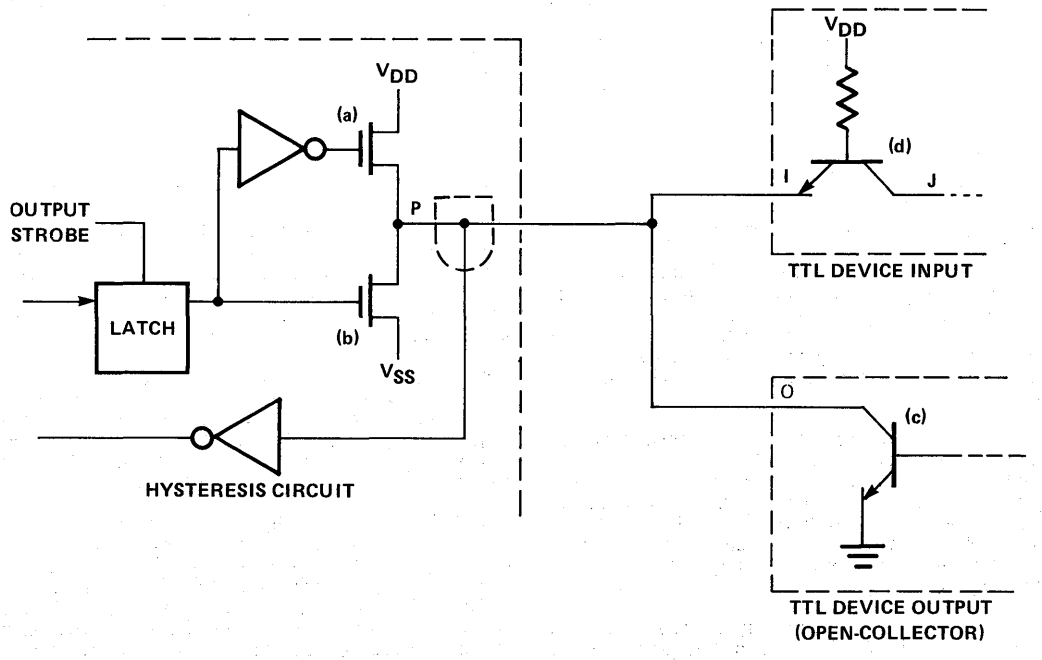
INPUT/OUTPUT INTERFACING

As illustrated in Figure 4, each I/O port pin is a "wire-AND" structure between an internal latch and any external signal. 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 4).

FIGURE 4 – F8 I/O PORT BIT



Data is input to the pin from a "TTL Device Output" in Figure 4.

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 a low level to the TTL device. If the level at P is set high, transistor (d) does not conduct. This is a high level to the TTL device.

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

A port input should only be driven by devices which are incapable of sourcing more than 2 mA when pulled to V_{SS}. Ideally only open collector T^2L or open drain CMOS logic devices should be used to drive an I/O Port bit. This will prevent damage to the I/O Port output buffers should they be pulling to V_{SS} while the external device is holding the port bit to a high level through an excessively low impedance. This condition can not be avoided with software since the damage may occur when a port bit "Powers Up" to a V_{SS} level.

Since the I/O pin and the TTL device output at O 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 O 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.

(Note: 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 operations 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 and 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 9 indicates I/O timing.

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

CLOCK CIRCUITS

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

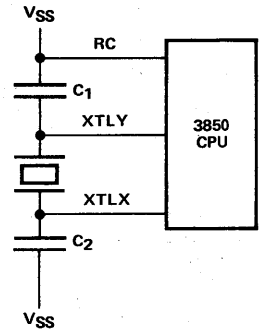
The 3850 CPU offers two alternate ways of generating a system clock; these are Crystal mode and External mode.

Crystal Mode

Figure 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=20pF

FIGURE 5 – CRYSTAL CONTROLLED CLOCK



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 6. For example, a slave 3850 CPU may receive its timing from a master 3850 CPU, by having the master Φ output drive the slave XTLY input.

FIGURE 6 – EXTERNAL CLOCK

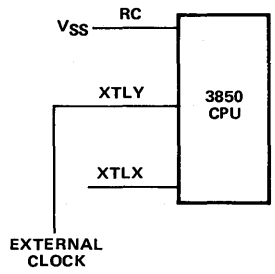


Figure 8 illustrates the AC characteristics of the clock signal needed for external mode clock generation, plus the AC characteristics of the Φ and WRITE signals generated by the CPU.

INSTRUCTION SET SUMMARY

The instruction set is summarized in Table 4. This table and the accompanying text explains the control signals and timing associated with the execution of every instruction.

The columns in Table 4 should be interpreted as follows:

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 3 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 3 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 is a short cycle.

ROMC STATE

This is the state, as identified in Table 2 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 12, 13 & 14. For the exceptions, Figure 9 is required. The ROMC lines are always set after a delay of td_3, as shown in Figure 12. 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 14. There are two cases for tdb_3, however, as illustrated in Figures 12 and 13; these are identified in Table 4 as 3S for Figure 12, and 3L for Figure 13. Delays tdb_1 through tdg_6 are identified by the numbers 1 through 6.

Cycles that do not use the data bus are identified by 0 in the timing column; Figure 10 illustrates timing in this case. In summary:

0	represents Figure 10
1	represents tdb_1 in Figure 14
2	represents tdb_2 in Figure 14
3S	represents tdb_3 in Figure 12
3L	represents tdb_3 in Figure 13
4	represents tdb_4 in Figure 14
5	represents tdb_5 in Figure 14
6	represents tdb_6 in Figure 14

STATUS FLAGS

Status flags are identified as follows:

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

Within each column, symbology is used as follows:

–	Status not effected
0	Status set to 0
I/O	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 3.

Observe that instructions are described in Table 4 in order of ascending instruction (first byte) object code.

TABLE 2 – ROMC CONTROL STATES

ROMC (Hexadecimal)	OPERATION PERFORMED	COMMENT
00	DB ← ((P0)) ; P0←P0 +1	0P CODE, FETCH
01	DB ← ((P0)) ; P0←P0 +DB	BRANCH OFFSET FETCH
02	DB ←((DC)); DC←DC+1	
03	DB ←((P0)) ; P0←P0+1	IMMEDIATE OPERAND FETCH
04	P0 ←P	
05	((DC))←DB ; DC←DC+1	MK3851 :DC← DC+1 ONLY
06	DB←DCU	
07	DB←P U	
08	P←P0 ; DB←H'00'; P0L, P0H← DB	EXTERNAL RESET
09	DB ←DCL	
0A	DC←DC+DB	
0B	DB←PL	
0C	DB←((P0)) ; DCL←DB	
0D	P←P0+1	
0E	DB←((P0)) ; DCL←DB	
0F	P←P0 ; DB←IAL ; P0L← DB	LOWER BYTE OF ADDRESS VECTOR
10	FREEZE INTERRUPT STATUS	PREVENT ADDRESS VECTOR CONFLICTS
11	DB←((P0)) ; DCU← DB	
12	P0L←DB ; P←P0	
13	DB←IAU; P0U← DB	UPPER BYTE OF ADDRESS VECTOR
14	P0U←DB	
15	PU←DB	
16	DCU←DB	
17	P0L ← DB	
18	PL←DB	
19	DCL←DB	
1A	((pp))←DB or ((p))←DB	
1B	DB←(pp)) or DB←((p))	
1C	NO OPERATION	
1D	DC ⇄ DC1	MK3851 : NO OPERATION
1E	DB←P0L	
1F	DB←P0U	

Definitions
- DB - Data Bus
- P0 - Program Counter
- DC - Data Counter
- P - Stack Register
- pp - Two hex digits (long I/O port address)
- p - One hex digit (short I/O port address)
- IA - Interrupt address vector
- L - Lower byte suffix
- U - Upper byte suffix
- ( ) - Contents of
- ← - transfer to
- ⇄ - exchange

TABLE 3 – SYMBOLOGY USED IN TABLES 2 and 4

SYMBOL	INTERPRETATION
( )	Contents of
A	The Accumulator contents.
a or H'a'	A single hexadecimal digit being interpreted as data.
aa or H'aa'	Two hexadecimal digits being interpreted as a single byte of data, or as the high order byte of 16 bits of data.
bb or H'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.
DC	The primary data counter register.
DCL	The low order byte of the primary data counter register.
DCU	The high order byte of the primary data counter register.
DC1	The auxiliary data counter register.
Decimal	Decimal arithmetic specified.
e or O'e'	A single octal digit being interpreted as data.
H	Scratchpad registers H'a' and H'b' contents.
ii or H'ii'	Two hexadecimal digits being interpreted as the high order byte of a 16-bit address, or as a simple byte address displacement.
IS	The six-bit scratchpad address register.
ISL	The low order three bits of ISAR.
ISU	The high order three bits of ISAR.
J	Scratchpad register H'9' contents.
jj or H'jj'	Two hexadecimal digits being interpreted as the low order byte of a 16-bit address.
K	Scratchpad registers H'c' and H'd' contents.
KL	Scratchpad register H'd' contents.
KU	Scratchpad register H'c' contents.
O	The overflow status flag.
p or H'p'	A single hexadecimal digit being interpreted as an I/O port address (short).
pp or H'pp'	Two hexadecimal digits being interpreted as an I/O port address (long).
P0	The program counter contents.
P0L	The low order byte of the program counter
P0U	The high order byte of the program counter
P	The stack register contents.
PL	The low order byte of the stack register
PU	The high order byte of the stack register
Q	Scratchpad registers H'e' and H'f'
QL	Scratchpad register H'f'
QU	Scratchpad register H'e'
r or H'r'	Single hexadecimal digit interpreted as scratchpad address: r = 0 through B for locations 0 through B in scratchpad. r = C or IS as address source with no change after access. r = D for IS as address source with ISL = ISL + 1 after access. r = E for IS as address source with ISL = ISL−1 after access. r = F is not allowed.

SYMBOLOGY USED IN TABLES 2 and 4 (continued)

SYMBOL	INTERPRETATION
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.
←	Transfer to
⇄	Exchange

TABLE 4 – 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 ← (PU)
			L	B	5	–	–	–	–		r13 ← (PL)
			S	0	3S	–	–	–	–		
LR	P, K	09	L	15	2	–	–	–	–		PU ← (r12)
			L	18	2	–	–	–	–		PL ← (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	–	–	–	–		P ← (P0)
											POL ← (r13)
			L	14	2	–	–	–	–		POU ← (r12)
			S	0	3S	–	–	–	–	x	
LR	P0, Q	0D	L	17	2	–	–	–	–		POL ← (r15)
			L	14	2	–	–	–	–		POU ← (r14)
			S	0	3S	–	–	–	–		

TABLE 4 – 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 ← (DCU)
			L	9	5	–	–	–	–		r15 ← (DCL)
			S	0	3S	–	–	–	–		
LR	DC, Q	0F	L	16	2	–	–	–	–		DCU ← (R14)
			L	19	2	–	–	–	–		DCL ← (R15)
			S	0	3S	–	–	–	–		
LR	DC, H	10	L	16	2	–	–	–	–		DCU ← (R10)
			L	19	2	–	–	–	–		DCL ← (R11)
			S	0	3S	–	–	–	–		
LR	H, DC	11	L	6	5	–	–	–	–		r10 ← (DCU)
			L	9	5	–	–	–	–		r11 ← (DCL)
			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 ← ((DC))
			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	–	–	–	–		PO ← (P)
			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) v 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 4 – 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; P0 ← P0+1
	→	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'jj'
	→	ii	S	D	0	–	–	–	–		P ← (P0) + 1
	→	jj	L	C	2	–	–	–	–		P0L ← H'jj'
			L	14	1	–	–	–	–		P0U ← (A)
			S	0	3S	–	–	–	–	x	
JMP	iijj	29	L	3	6	–	–	–	–		A ← H'ii'
	→	ii	L	C	2	–	–	–	–		P0L ← H'jj'
	→	jj	L	14	1	–	–	–	–		P0U ← (A)
			S	0	3S	–	–	–	–	x	
DCI	iijj	2A	L	11	2	–	–	–	–		DCU ← ii
	→	ii	S	3	0	–	–	–	–		(increment P0)
	→	jj	L	E	2	–	–	–	–		DCL ← jj
			S	3	0	–	–	–	–		(increment P0)
			S	0	3S	–	–	–	–		
NOP		2B	S	0	3S	–	–	–	–		P0 ← (P0) + 1
XDC		2C	S	1D	0	–	–	–	–		DC0 ⇆ DC_1
			S	0	3S	–	–	–	–		
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 P0 = (P0) + 2
			S	0	3S	–	–	–	–		
			S	1C	0	–	–	–	–		Test e ∧ W. register
			L	1	2	–	–	–	–		Res ≠ 0 so P0 = (P0) + H'ii' + 1
			S	0	3S	–	–	–	–		
AM		88	L	2	4	1/0	1/0	1/0	1/0		A ← (A) + ((DC)) Binary, DC ← (DC) + 1
			S	0	3S						
AMD		89	L	2	4	1/0	1/0	1/0	1/0		A ← (A)+((DC)) Decimal; DC ← (DC) + 1
			S	0	3S						

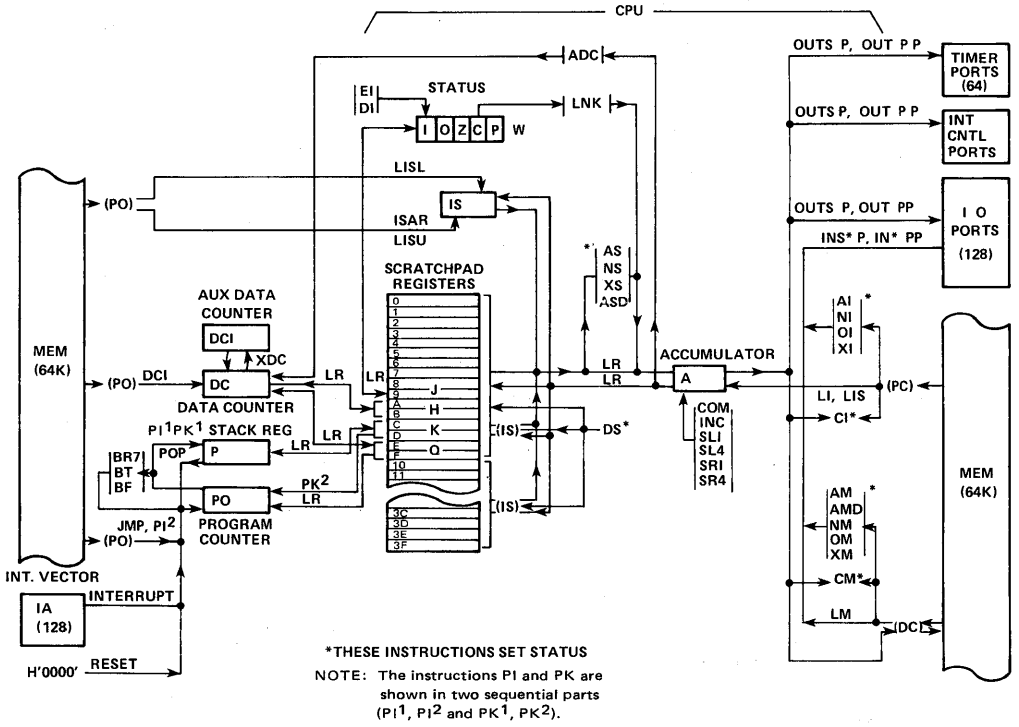
TABLE 4 – INSTRUCTIONS' EXECUTION AND TIMING (continued)

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) ∧ ((DC));
			S	0	3S						DC ← (DC) + 1
OM		8B	L	2	4	0	1/0	0	1/0		A ← (A) V ((DC)),
			S	0	3S						DC ← (DC) + 1
XM		8C	L	2	4	0	1/0	0	1/0		A ← (A) ⊕ ((DC));
			S	0	3S						DC ← (DC) + 1
CM		8D	L	2	4	1/0	1/0	1/0	1/0		Set status flags on basis
			S	0	3S						of ((DC)) + (A) + 1;
											DC ← (DC) + 1
ADC		8E	L	A	1	–	–	–	–		DC ← (DC) + A
			S	0	3S	–	–	–	–		
BR7	ii	8F	S	3	0	–	–	–	–		P0 ← (P0) + 2
	→	ii	S	0	3S	–	–	–	–		because (ISARL) = 7
			L	1	2	–	–	–	–		P0 ← (P0) + 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 P0 = (P0)
			S	0	3S	–	–	–	–		+ H'ii' + 1
			S	1C	0	–	–	–	–		Test t ∧ W. register
			S	3	0	–	–	–	–		Res ≠ 0 so P0 = (P0)
			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	2	A2	L	1C	0	0	1/0	0	1/0		DB ← Port address (2
	through	through	L	1B	6	–	–	–	–		through 15)
	15	AF	S	0	3S	–	–	–	–		A ← (Port 2 through 15)
OUTS	0 or 1	B0, B1	S	1C	0	–	–	–	–		I/O Port 0 or 1 ← (A)
			S	0	3S	–	–	–	–		
OUTS	2	B2	L	1C	0	–	–	–	–		DB ← Port address (2
	through	through	L	1A	1	–	–	–	–		through 15)
	15	BF	S	0	3S	–	–	–	–	x	Port (2 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	–	–	–	–		P0L ← Int. address
											(lower byte); PC1 ← P0
			L	13	2	–	–	–	–	y	P0U ← Int. address
											(upper byte)
			S	0	3S	–	–	–	–	x	
RESET		xx	S	1C	0	–	–	–	–		IDLE
			L	8	1	–	–	–	–	y	P ← P0, P0 ← 0
			S	0	3S	–	–	–	–	x	

PROGRAMMING MODEL

Figure 7 shows a Programming Model of the F8 Microcomputer system. This diagram is intended to depict the various data transfers and manipulations which are facilitated by the instruction set of the F8. Every F8 system configuration will contain the basic functional elements shown in this diagram, with the exception of the Auxiliary Data Counter (DC1). The Auxiliary Data Counter is available only in those systems incorporating the MK3852 Dynamic Memory Interface, the MK3853 Static Memory Interface, or the MK3870 single chip F8 Microcomputer.

FIGURE 7 – F8 PROGRAMMING MODEL



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}.

SUPPLY CURRENTS

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
I_{DD}	V_{DD} Current		30	80	mA	f = 2 MHz, Outputs unloaded f - 2 MHz
I_{GG}	V_{GG} Current		15	25	mA	Outputs unloaded

TABLE 5 – AC CHARACTERISTICS

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

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	20		110	ns	
tx_2	Ext. to Φ + to + Delay	20		125	μs	
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	60	150	250	ns	C_L = 100 pF
td_2	Φ to WRITE – Delay	60	150	225	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			410	ns	C_L = 50 pF
td_5	WRITE to $\overline{INT\ REQ}$ – Delay			430	ns	C_L = 100 pF
td_6	WRITE to $\overline{INT\ REQ}$ + Delay			1.65	μs	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			1.5	μs	C_L = 50 pF
tdb_0*	WRITE to data bus High Impedance		250	500	ns	
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	300			ns	
tdb_5	Data Bus Set-up	500			ns	
tdb_6*	Data Bus Set-up	300			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.

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

TABLE 6 – DC CHARACTERISTICS

(V_{SS} = 0V, V_{DD} = +5V ± 5%; V_{GG} = +12V ± 5%)

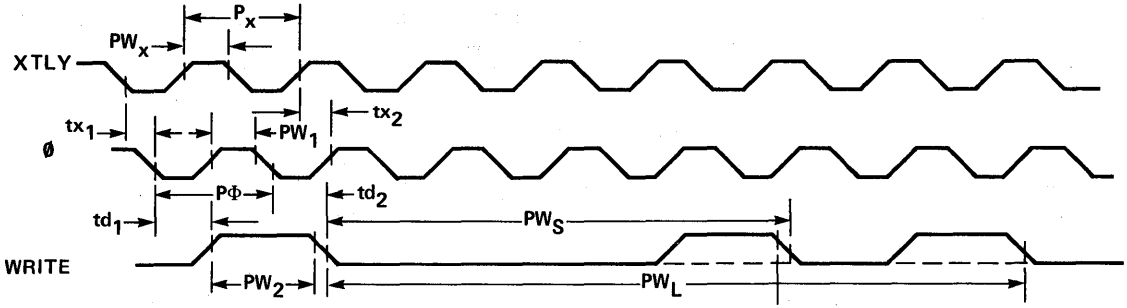
SIGNAL	SYMBOL	PARAMETER	MIN.	MAX.	UNITS	TEST CONDITIONS
Φ, WRITE	V_{OH}	Output High Voltage	4.4	V_{DD}	Volts	I_{OH} = −10 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1.6 mA
	V_{OH}	Output High Voltage	2.9		Volts	I_{OH} = −100 μA
XTLY	V_{IH}	Input High Voltage	4.5	V_{GG}	Volts	
	V_{IL}	Input 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	−80	μA	V_{IN} = V_{SS}
ROMC0	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
ROMC4						
DB0	V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
DB7	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} = 7V 3-State mode
	I_{IL}	Input Low Current		−1	μA	V_{IN} = V_{SS}, 3-State mode
I/O 0	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
I/O 17	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_L	Leakage Current		1	μA	V_{IN} = V_{DD}
	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		−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		−1.0	mA	V_{IN} = V_{SS}
$\overline{\text{ICB}}$	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} = 100μ A

(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.

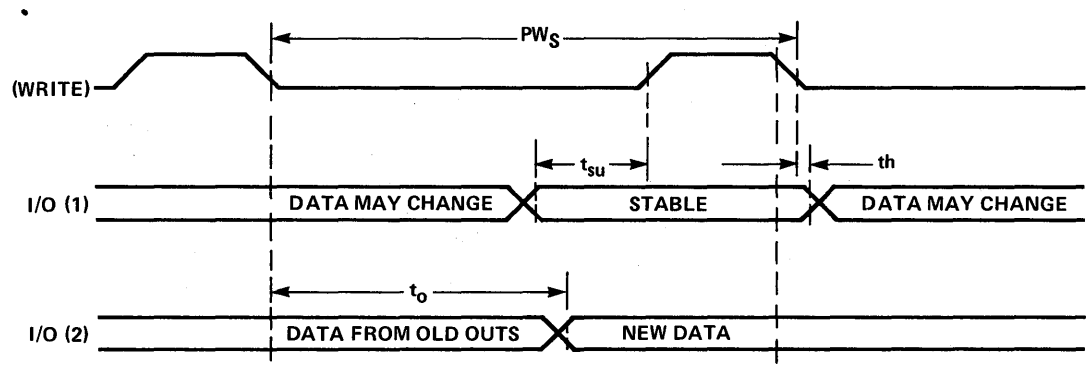
NOTE: Positive current is defined as conventional current flowing into the pin referenced.

FIGURE 8 – TIMING SIGNAL SPECIFICATIONS



Parameters are described in Table 5

FIGURE 9 – TIMING FOR DATA INPUT OR OUTPUT AT I/O PORT PINS

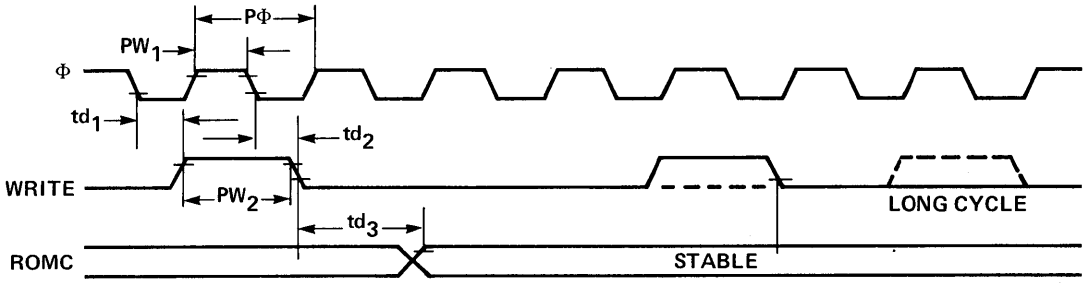


(1) This represents the timing for data at the I/O pin during the execution of the INS instruction, i.e., the CPU is inputting.

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

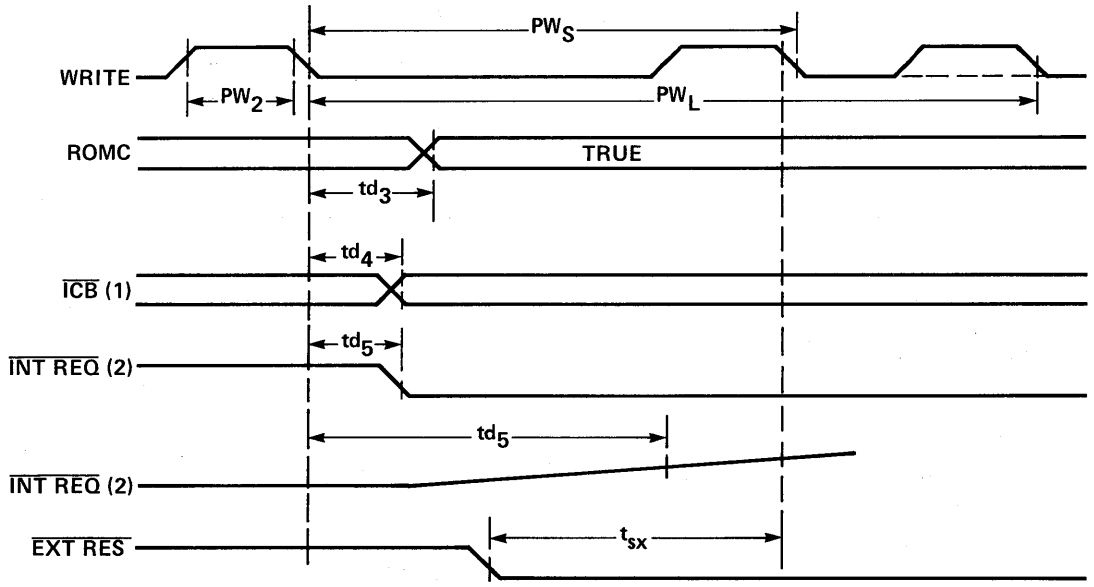
Symbols are defined in Table 5

FIGURE 10 – ROMC SIGNALS OUTPUT BY 3850 CPU



Symbols are defined in Table 5

FIGURE 11 – INTERRUPT SIGNALS TIMING

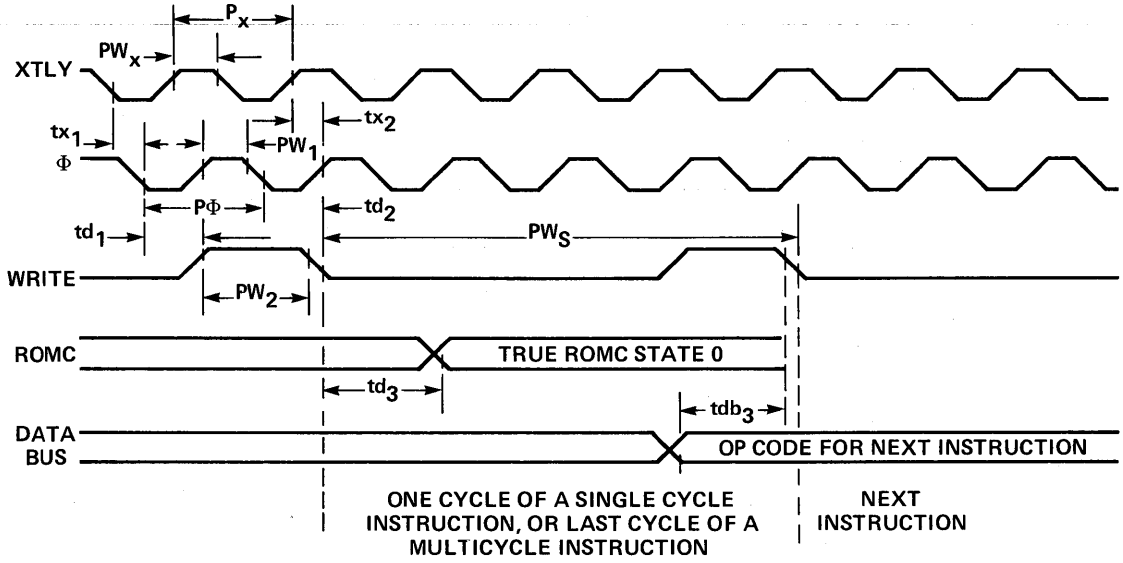


(1) ICB will go from a 1 to a 0 following the execution of the EI instruction and will go from a 0 to 1 following either the execution of the DI instruction or the CPU's acknowledgement of an interrupt.

(2) This is an input ot 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.

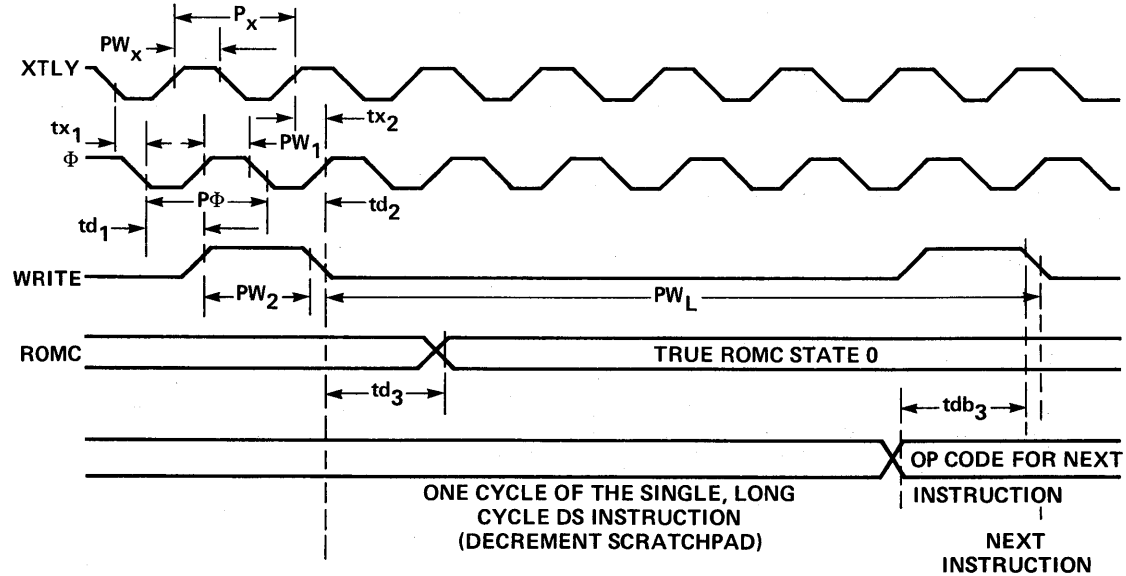
Symbols are defined in Table 5

FIGURE 12 – A SHORT CYCLE INSTRUCTION FETCH



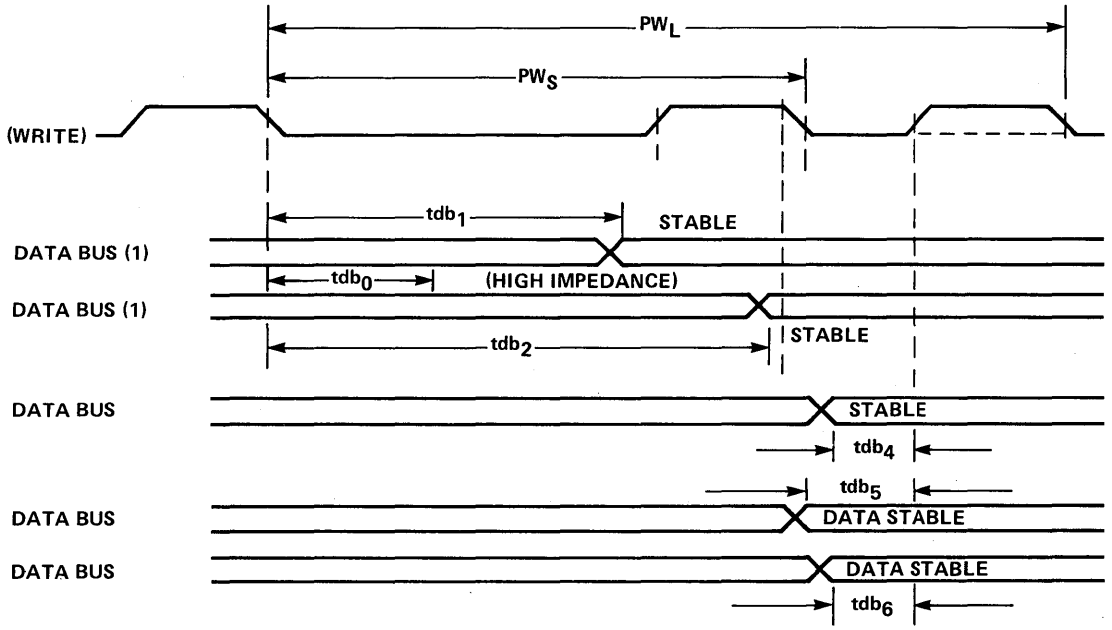
Symbols are defined in Table 5

FIGURE 13 – A LONG CYCLE INSTRUCTION FETCH (DURING DS ONLY)



Symbols are defined in Table 5

FIGURE 14 – MEMORY REFERENCE TIMING



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 imputting 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 reference point is required.

Symbols are defined in Table 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
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}.

SUPPLY CURRENTS (MK3850N-3, MK3850P-3)

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
I_{DD}	V_{DD} Current		30	80	mA	f = 2 MHz, Outputs unloaded
I_{GG}	V_{GG} Current		15	25	mA	Outputs unloaded

SUPPLY CURRENTS (MK3850N-13, MK3850P-13)

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
I_{DD}	V_{DD} Current		35	90	mA	f = 2 MHz, Outputs unloaded
I_{GG}	V_{GG} Current		20	33	mA	Outputs unloaded

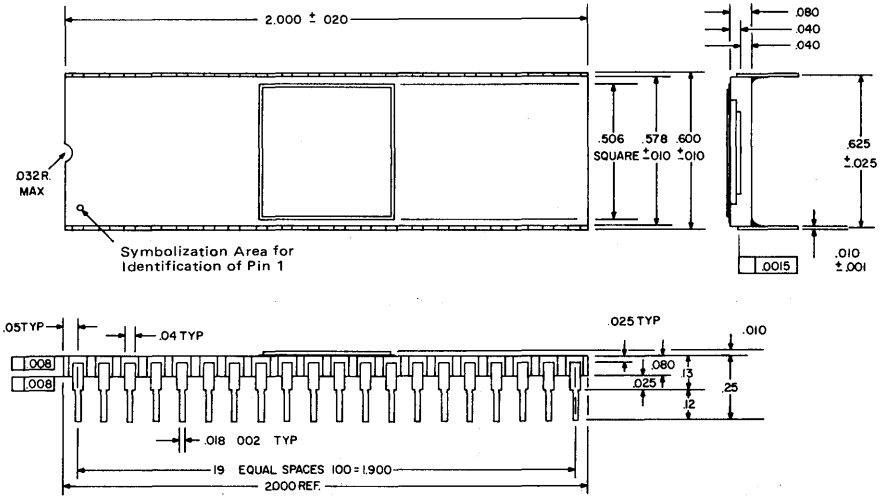
SUPPLY CURRENTS (MK3850P-23)

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNITS	TEST CONDITIONS
I_{DD}	V_{DD} Current		40	100	mA	f = 2 MHz, Outputs unloaded
I_{GG}	V_{GG} Current		25	40	mA	Outputs unloaded

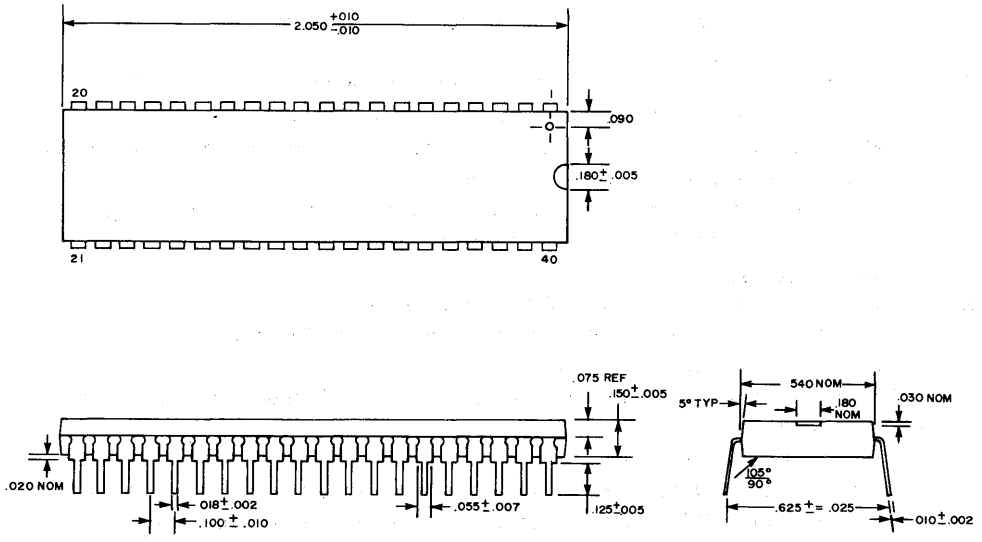
ORDER INFORMATION

PART NO.	PACKAGE TYPE	TEMPERATURE RANGE (T_A)	COMMENTS
MK3850N-3	Plastic	0°C to +70°C	
MK3850P-3	Ceramic	0°C to +70°C	
MK3850N-13	Plastic	−40°C to +85°C	
MK3850P-13	Ceramic	−40°C to +85°C	
MK3850P-23	Ceramic	−55°C to +125°C	

PACKAGE DESCRIPTION — 40-Pin Dual-In-Line Ceramic Package



PACKAGE DESCRIPTION — 40-Pin Dual-in-Line Plastic Package



INSTRUCTIONS FOR COMPLETION OF MASK PROGRAMMED PART FORM:

1. List customer name.
2. List customer address.
3. List customer city, state, and zip code.
4. List customer phone number and extension.
5. List a contact within the customer's company that can be called for reply to engineering questions.
6. List the responsible Disbributor should the order be placed through a Distributor.
7. List the ROM/PSU/3870 part number for example 3851/12XXX, 34XXX, or MK 3870/141XXX.
8. List the package type (plastic or ceramic) required by the customer for the production order (NOTE - prototypes will be Dallas assembled in ceramic).
9. List the customer part number.
10. List any special branding requirements desired by the customer (NOTE - usually the MOSTEK exclusive part will suffice for customer branding requirements).
11. List the customer specification number and indicate whether the customer intends to send a specification to MOSTEK for file. Should you circle NO this denotes that parts will be tested to the standard MOSTEK data sheet.
12. Should the customer request his specification to be on file with MOSTEK, please indicate the date that the customer spec was sent to MOSTEK.
13. Circle the pattern media that the customer wishes to use to transmit code to MOSTEK.
14. Indicate the verification media requested by the customer from MOSTEK. (NOTE - the listing is usually sufficient).
15. Check the port option requested by the customer (make reference to note #1).
16. Indicate the date that the customer's pattern was sent to MOSTEK.
17. Indicate whether the customer requires prototypes (NOTE - standard quantity of prototype Dallas assembled in ceramic is 10).
18. Indicate whether the customer requires pattern verification. Check YES or WAIVER.
19. Indicate whether the customer requires prototype verification. Indicate by checking YES or WAIVER.
20. Make any comments concerning waivers if stated above.
21. The customer purchase order to MOSTEK direct or to his Distributor.
22. List the date of the customer order.
23. List the Distributor purchase order number to MOSTEK should the order be placed through a Distributor.
24. Indicate the production quantity and price.
25. Indicate the delivery dates requested or committed to the customer; both prototypes and production. (NOTE - standard commitment is six weeks to prototype after verification of listing and twelve weeks from prototype verification to production).
26. Date this form was completed and forwarded to MOSTEK.
27. Name of Representative completing this form.

Customer Name _____

Address _____

City _____ State _____ Zip _____

Phone (____) _____ Extension _____

Contact _____

Distributor _____

ROM Generic Type _____

Package Type _____

Customer Part Number _____

Branding Requirement _____

Customer Specification: _____ ☐ Yes

_____ ☐ No Parts to be tested to standard Data Sheet

_____

Date customer spec sent to MOSTEK _____

PATTERN MEDIA

- ☐ EMU-70
- ☐ PROM
- ☐ Paper Object Tape
- ☐ Silent 700 Cassette
- ☐ Card Deck
- ☐ Tape of Card Deck

VERIFICATION MEDIA

- ☐ Listing
- ☐ Other

PORT OPTION (Note 1)

- ☐ Standard TTL
- ☐ Open Drain
- ☐ Driver Pullup

(Note 2) _____

Date Pattern Data Sent to MOSTEK _____

Does Customer Require Prototypes ☐ Yes ☐ No

Pattern Verification Required by Customer ☐ Yes ☐ Waived

Prototype Verification Required by Customer ☐ Yes ☐ Waived

COMMENTS: (Waiver Explanation) _____

_____

Customer Order Number _____

Date of Customer Order _____

Distributor Order Number to MOSTEK _____

Order Quantity and Price _____

Delivery Requested/Committed Prototypes _____

Production _____

Date Form Completed _____

Name of Representative Completing Form _____

Program Storage Unit MK3851

FEATURES

- ☐ 1024 x 8 ROM storage
- ☐ Two 8-bit I/O Ports
- ☐ Programmable timer
- ☐ External/timer interrupt circuitry
- ☐ Low power dissipation < 275mW typical

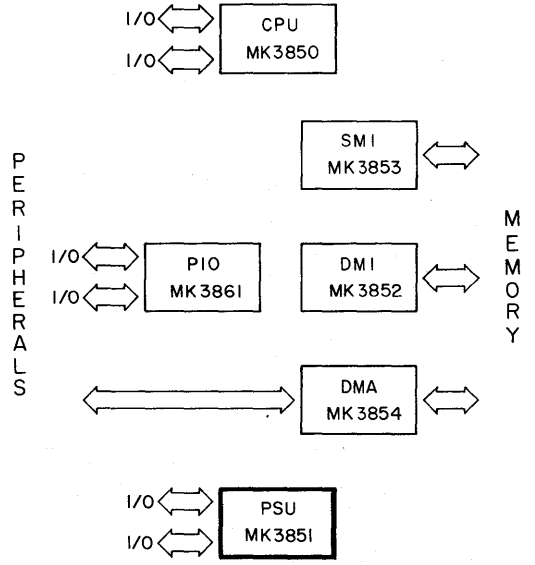
GENERAL DESCRIPTION

The MK 3851 program storage unit (PSU) provides 1024 bytes of read only memory (ROM) for the F8 system. Additionally each PSU provides two 8-bit I/O ports, a programmable timer and vectored timer and external interrupts. The PSU contains three 16-bit address registers and a 16-bit incrementer/adder. On command from the F8 CPU the MK 3851 accesses its internal memory using one of these three registers and increments or adds displacement to the register if required.

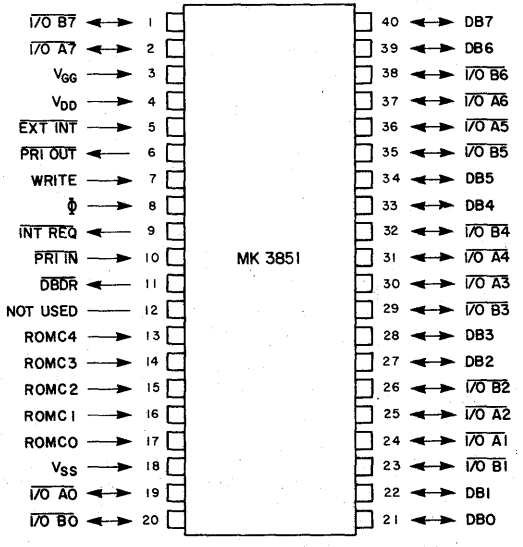
The MK 3851 PSU is manufactured using N-channel Isoplanar MOS technology. Power dissipation is very low, typically less than 275mW.

PIN NAME	DESCRIPTION	TYPE
$\overline{\text{I/O A0-I/O A7}}$	I/O Port A	Bi-directional
$\overline{\text{I/O B0-I/O B7}}$	I/O Port B	Bi-directional
DB0-DB7	Data Bus	Bi-directional, tri-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

F8 FAMILY



PIN CONNECTIONS



FUNCTIONAL DIAGRAM

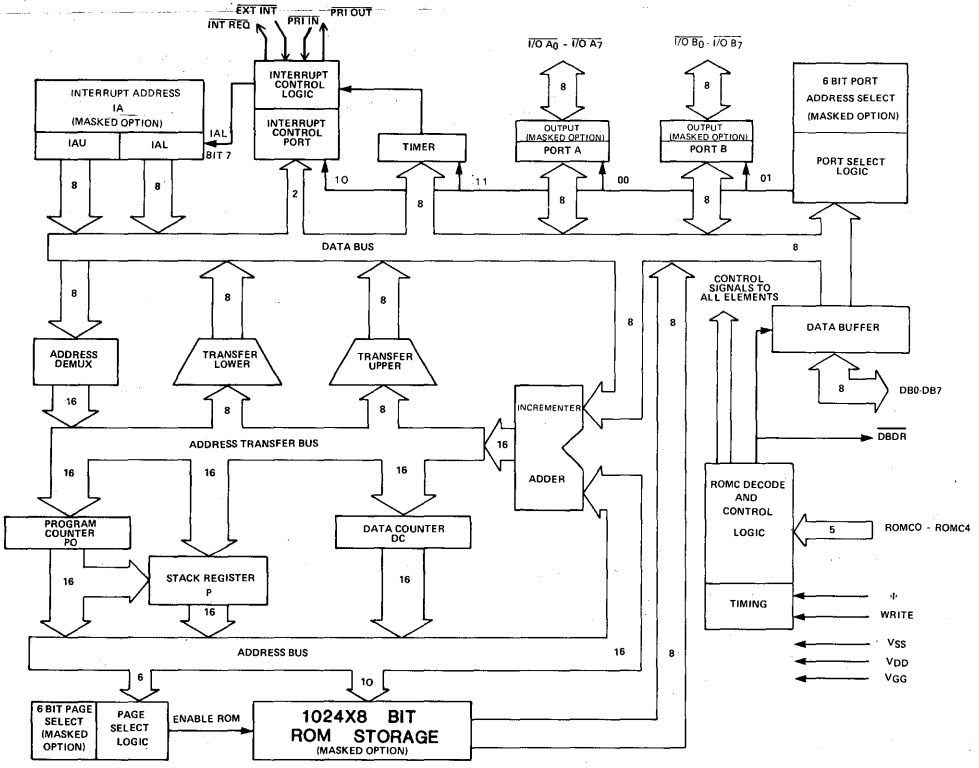


Figure 1

FUNCTIONAL PIN DESCRIPTION

Φ and WRITE are clock inputs generated by the MK 3850 CPU.

ROMC0 through ROMC4 are control inputs generated by the MK 3850 CPU.

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

$\overline{\text{INT REQ}}$. This signal is connected to the $\overline{\text{INT REQ}}$ input on the 3850 CPU. $\overline{\text{INT REQ}}$ is output low to interrupt the MK 3850 CPU. This occurs only if $\overline{\text{PRI IN}}$ is low, and MK 3851 PSU interrupt control logic is requesting an interrupt.

$\overline{\text{EXT INT}}$. A high to low transition on this signal is recognized by the MK 3851 as an interrupt request from an external device.

$\overline{\text{PRI IN}}$. This input must be low to allow the MK 3851 PSU to set $\overline{\text{INT REQ}}$ low in response to an interrupt.

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

$\overline{\text{I/O A0}}$ through $\overline{\text{I/O A7}}$ and $\overline{\text{I/O B0}}$ through $\overline{\text{I/O B7}}$ are two Input/Output bi-directional ports through which the MK 3851 PSU communicates with logic external to the microprocessor system.

$\overline{\text{DBDR}}$ is low when the MK 3851 PSU is outputting data on the data bus (DB0-DB7). $\overline{\text{DBDR}}$ is an open drain signal.

DEVICE ORGANIZATION

This section describes the operation of the basic functional elements of the MK 3851 PSU. These elements are shown on the PSU functional block diagram. (Fig.1)

ROM STORAGE

The MK 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 MK 3851 PSU is implemented using a custom mask which specifies the state of every ROM bit, as well as certain address mask options which are external to the ROM array.

THE PROGRAM COUNTER (P0) AND DATA COUNTER (DC)

The MK 3851 PSU addressing logic consists primarily of two 16-bit registers: the program counter (P0) and the data counter (DC).

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 mechanism whereby an address is decoded by the MK 3851 PSU logic is identical, whether the address originated in P0 or in DC.

Recall that P0 always addresses the memory location out of which the next object program instruction byte will be read. If the instruction requires data (an operand) other than an immediate operand to be accessed, DC must address memory. P0 cannot be used to address a non-immediate operand since P0 is saving the address of the next instruction code.

THE STACK REGISTER

The MK 3851 PSU addressing logic contains a third 16-bit register, called the stack register. The stack register is labeled P on Figure 1. The stack register is a buffer for the program counter P0. The contents of the stack register are never used directly to address memory.

The following instructions access P:

LR K,P	MOVE THE CONTENTS OF P TO THE CPU SCRATCHPAD K REGISTERS
LR P,K	MOVE THE CONTENTS OF THE CPU K SCRATCHPAD REGISTERS TO P
PK	SAVE THE CONTENTS OF P0 IN P THEN MOVE THE CONTENTS OF CPU SCRATCHPAD REGISTERS 12 AND 13 TO P0
PI H'aaaa'	MOVE THE CONTENTS OF P0 TO P THEN LOAD THE HEXADECIMAL VALUE INTO P0
POP	MOVE THE CONTENTS OF P TO P0

In addition, when an interrupt is acknowledged, the contents of P0 are saved in P.

PAGE SELECT LOGIC

All memory addresses are 16-bits wide, whether the memory address originates in the program counter or the data counter. Addressing logic within the MK 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 MK 3851 PSU has a 6-bit page select register, which is a mask option that must be specified when the PSU ROM chip is ordered. If the high order six bits of the address match the page select mask, 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 match the page select, 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 MK 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. Following are two 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 H'0000'
0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 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 H'2C00'
0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 through
H'2FFF'

Six high order address bits — Ten low order address bits

INCREMENTER ADDER LOGIC

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

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.

The incrementer adder logic performs these two functions in the MK 3851 PSU.

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 MK 3851 CPU.

TIMER LOGIC

Every MK 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 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.

THE DATA BUS

The 8-bit data bus is the main path for transfer of information between the MK 3850 CPU and other devices in the F8 microprocessor system. It is identified in Figure 1 by data lines DB0-DB7.

I/O PORTS

Every MK 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, that must be specified at the time the MK 3851 PSU is ordered.

Two of the four I/O ports, identified as I/O ports A and B in Figure 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 the Interrupt Control Port.

The four I/O ports of any MK 3851 PSU are addressed by an 8-bit I/O port address. The high order 6 bits are specified by the I/O port address select code with the remaining 2 bits identifying the particular I/O port as following:

XXXXXX00 I/O Port A
XXXXXX01 I/O Port B
XXXXXX10 Interrupt control
XXXXXX11 Programmable Timer

XXXXXX represents a six bit PSU mask option. For example, if the six are 000010, 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 negative true logic also applies to input. The I/O ports, timer, and interrupt control ports are not initialized during the power on reset.

MASK OPTIONS

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

1. The 1024 bytes of ROM storage. This will reflect programs and permanent data tables stored in the PSU memory.
2. The 6-bit page select. This defines the PSU address space
3. The 6-bit I/O port address select. This defines the four PSU I/O port addresses.
4. The 16-bit interrupt address vector, excluding bit 7.
5. The I/O port output option. The choices are the standard Pull-up (Option A), the Open-Drain (Option B) and the Driver Pull-up (Option C)

OPERATIONAL DESCRIPTION

CLOCK TIMING

All timing within the MK 3851 PSU is controlled by Φ and WRITE, which are input from the MK 3850 CPU.

The WRITE clock refreshes and updates MK 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.

INSTRUCTION EXECUTION

The MK 3851 PSU responds to signals which are output by the MK 3850 CPU in the course of executing instruction cycles.

Table 1 summarizes the response of the MK 3851 PSU to the ROMC states.

MEMORY ADDRESSING

Those ROMC states which specify a memory access call for only one memory device to respond to the memory access operation. However, every memory device responds to ROMC states that call for modification of program counter or data counter register contents. Consider two examples:

1. ROMC state 5 specifies that the data counter (DC) register contents must be incremented. Every memory device will simultaneously receive this ROMC state, and will simultaneously increment the contents of its DC 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 (P0) 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 P0 register, whether or not the P0 register contents are within the memory space of the device.

When all memory devices connected to the 8-bit data bus of a MK 3850 CPU are also connected to the ROMC control lines of the same CPU, the memory devices simultaneously receive the same ROMC state signals from the CPU and respond to ROMC states by identically modifying the contents of memory address registers. Therefore the P0 register on all memory devices contain identical information. The same holds true for DC and P registers.

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

1. Memory devices must receive the ROMC state signals from one CPU.
2. Page select masks must not be duplicated. (More than one memory device cannot have the same memory space).
3. The memory address contained in the specified register (P0 or DC) must be within the memory space of a memory device.

DATA OUTPUT BY THE PSU

Figure 10 shows the timing when the MK 3851 PSU outputs data on the data bus. This timing applies whenever a MK 3851 PSU is the data source. The MK 3851 PSU always places data on the data bus in time for the set-up required by an MK 3850 CPU destination.

ROMC STATES

ROMC (Hexadecimal)	OPERATION PERFORMED	COMMENT
00	DB ← ((P0)) ; P0←P0 +1	0P CODE, FETCH
01	DB ← ((P0)) ; P0←P0 +DB	BRANCH OFFSET FETCH
02	DB ←((DC)); DC←DC+1	
03	DB ←((P0)) ; P0←P0+1	IMMEDIATE OPERAND FETCH
04	P0 ←P	
05	((DC))←DB ; DC←DC+1	MK 3851:DC← DC+1 ONLY
06	DB←DCU	
07	DB←P U	
08	P←P0 ; DB←H'00'; P0L, P0H← DB	EXTERNAL RESET
09	DB ←DCL	
0A	DC←DC+DB	
0B	DB←PL	
0C	DB←((P0)) ; DCL←DB	
0D	P←P0+1	
0E	DB←((P0)) ; DCL←DB	
0F	P←P0 ; DB←IAL ; P0L← DB	LOWER BYTE OF ADDRESS VECTOR
10	FREEZE INTERRUPT STATUS	PREVENT ADDRESS VECTOR CONFLICTS
11	DB←((P0)) ; DCU←DB	
12	P0L←DB ; P←P0	
13	DB←IAU; P0U←DB	UPPER BYTE OF ADDRESS VECTOR
14	P0U←DB	
15	PU←DB	
16	DCU←DB	
17	P0L ←DB	
18	PL←DB	
19	DCL←DB	
1A	((pp))←DB or ((p))←DB	
1B	DB←(pp)) or DB←((p))	
1C	NO OPERATION	
1D	DC ⇄ DC1	MK 3851: NO OPERATION
1E	DB←P0L	
1F	DB←P0U	

Definitions

DB	-	Data Bus	IA	-	Interrupt address vector
P0	-	Program Counter	L	-	Lower byte suffix
DC	-	Data Counter	U	-	Upper byte suffix
P	-	Stack Register	( )	-	Contents of
pp	-	Two hex digits (long I/O port address)	←	-	transfer to
p	-	One hex digit (short I/O port address)	⇄	-	exchange

Table 1

Observe that $\overline{DBDR}$ is low while data output by the MK 3851 PSU is stable on the data bus. Thus $\overline{DBDR}$ low indicates that the data bus currently contains data flowing from a MK 3851 PSU. For systems with more than one MK 3851 PSU 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 td8 into the instruction cycle following the one in which $\overline{DBDR}$ was set low.

DATA INPUT TO THE PSU

The worst case timing for the MK 3851 PSU receiving data from the data bus is when the data must be added to a 16 bit number within the PSU's Incrementer Adder. This worst case corresponds to data coming from the accumulator in 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.td4 in Figure 10 identifies this worst case timing.

INPUT/OUTPUT INTERFACING

The two PSU I/O ports with addresses xxxxxx00 and xxxxxx01 (xxxxxx is the 6-bit I/O port address select) may be used to transmit data between the PSU and external devices. IN and INS instructions

STANDARD PULL–UP CONFIGURATION

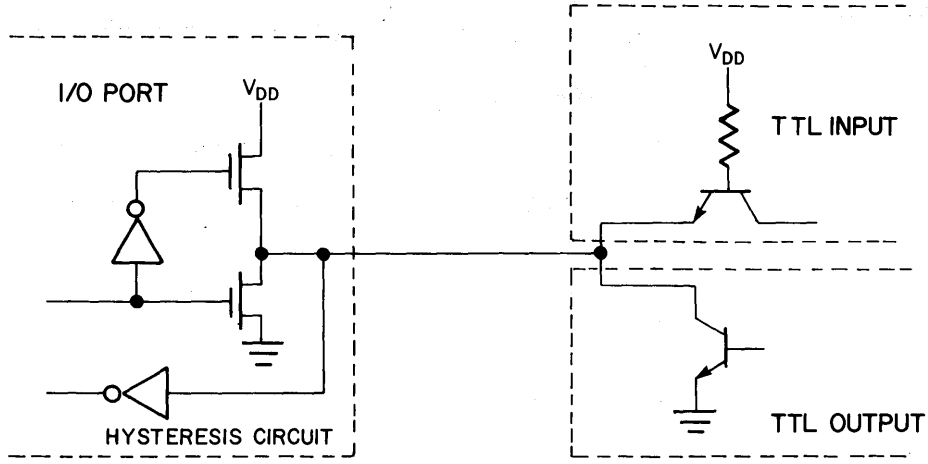


Figure 2

OPEN DRAIN CONFIGURATION

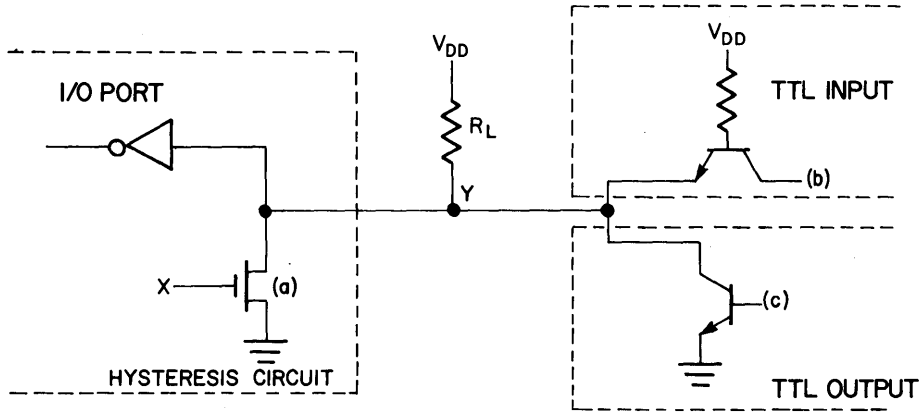


Figure 3

DRIVER PULL–UP CONFIGURATION

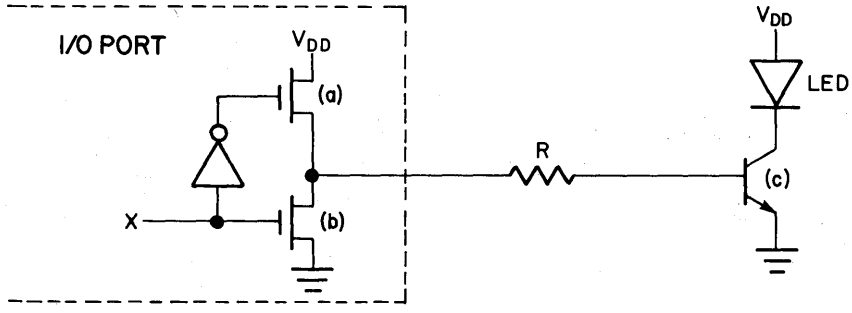


Figure 4

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 latch.

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 depends on which port option is being used. Figures 2,3 and 4 illustrate the three port options. Figure 11 illustrates timing for the three cases. Figure 2 illustrates the standard pull-up configuration.

When the I/O port is configured as shown in Figure 3 the drain connection of FET (a) is "open", (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 2 high output 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} (up to V_{GG}) for interfacing to external circuits requiring a higher 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. 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 it to conduct and pull point Y low. This transfers a high level to the internal I/O port logic through the inverting 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 4 shows the I/O port driver pull-up option shown driving a LED indicator. This application is typical of a front-panel address or data display, where a row of LED indicators shows the logic state 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. However, if 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), causing the LED to conduct and be lit.

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.

THE PROGRAMMABLE TIMER

The MK 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 5 illustrates the shift register logic and the exclusive-OR feedback path.

CONVERSION OF TIMER COUNTS INTO TIMER CONTENTS

	0	1	2	3	4	5	6	7	8	9
0	7F	BF	5F	2F	97	CB	E5	72	39	1C
1	0E	87	43	A1	D0	E8	F4	7A	3D	1E
2	0F	07	03	01	00	80	C0	60	B0	D8
3	EC	F6	7B	BD	5E	AF	D7	6B	35	1A
4	0D	06	83	41	A0	50	A8	54	AA	55
5	2A	15	8A	C5	E2	F1	F8	7C	3E	9F
6	CF	E7	73	B9	5C	AE	57	2B	95	CA
7	65	32	99	CC	66	B3	59	2C	16	0B
8	05	02	81	40	20	10	08	84	C2	61
9	30	98	4C	26	13	89	44	22	11	88
10	C4	62	B1	58	AC	56	AB	D5	6A	B5
11	5A	AD	D6	EB	75	BA	DD	6E	B7	5B
12	2D	96	4B	A5	D2	E9	74	3A	9D	CE
13	67	33	19	8C	C6	63	31	18	0C	86
14	C3	E1	70	38	9C	4E	27	93	C9	E4
15	F2	79	BC	DE	EF	77	BB	5D	2E	17
16	8B	45	A2	51	28	14	0A	85	42	21
17	90	48	24	12	09	04	82	C1	E0	F0
18	78	3C	9E	4F	A7	D3	69	34	9A	4D
19	A6	53	29	94	4A	25	92	49	A4	52
20	A9	D4	EA	F5	FA	7D	BE	DF	6F	37
21	1B	8D	46	23	91	C8	64	B2	D9	6C
22	B6	DB	6D	36	9B	CD	E6	F3	F9	FC
23	7E	3F	1F	8F	47	A3	D1	68	B4	DA
24	ED	76	3B	1D	8E	C7	E3	71	B8	DC
25	EE	F7	FB	FD	FE	FF halts timer				

Each timer count = 15.5 μs at 2MHz

Table 2

Based on the logic illustrated in Figure 5 binary values in the range 0 through 254 when loaded into the timer, are converted into "timer counts", as shown in table 2. Table 2 contains the actual (HEX) value loaded into the timer, and the column/row is the corresponding decimal number of time intervals the timer will take to time out. Data cannot be read out of the programmable timer I/O port.

Either 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

For example, a value of 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 H'FF' loaded into the programmable timer will stop the timer. This is because the timer shift

TIMER LOGIC DIAGRAM

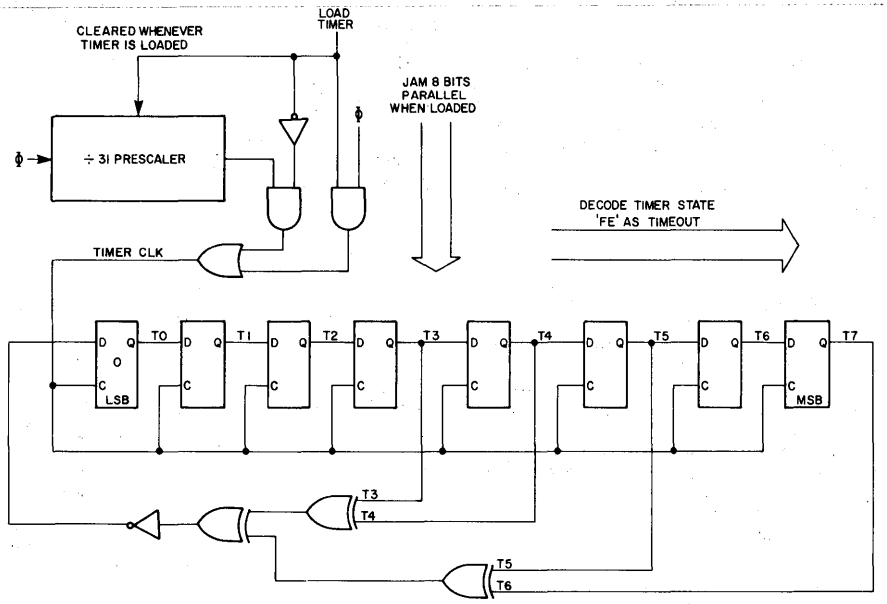


Figure 5

register feedback gates will always present a logic 1 to the D input of the LSB flip-flop (fig. 5). Therefore the timer will retain a value to H'FF' and a H'FE' will never be decoded to cause a time out.

The timer runs continuously unless it has been stopped by loading H'FF' into it. Upon timing out, the timer transmits an interrupt request to the interrupt logic. If proper interrupt logic conditions exist, the timer interrupt request is passed on to the CPU via INT REQ.

After the programmable timer has timed out it will again time out after 255 time counts. Therefore if the programmable timer is simply left running, it will time out every 7905 Φclock periods, or every 3.9525 milliseconds for a 500 nanosecond clock.

Whenever the timer and timer interrupt are being set to time a new interval, the timer should be loaded 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 a spurious interrupt. Time outs of the timer are 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 the continuous running timer timed out while timer interrupts were disabled.

If the timer is loaded just prior to enabling timer interrupts a spurious interrupt request will not exist when the timer interrupt is enabled.

Figure 6 illustrates a possible sequence for a timer which is initially loaded with H'C8' then allowed to run continuously.

INTERRUPT LOGIC ORGANIZATION

The Interrupt Control 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 from this port. The contents of the Interrupt Control Port are interpreted as follows:

CONTENTS OF INTERRUPT CONTROL PORT	FUNCTION
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 I/O port contents definitions x represents "don't care" bits.

Depending on the contents of the Interrupt Control Port, a MK 3851 PSU's interrupt control logic can be accepting timer interrupts, or external interrupts, or neither, but never both.

Figure 7 is a conceptual logic diagram of the PSU's interrupt logic. Between the EXT INT input or the time-out input and the output INT REQ, there are 4 flip-flops. EXT INT and the time-out interrupt input each have 2 synchronizing flip-flops to detect the active edge.

TIME OUT AND INTERRUPT REQUEST

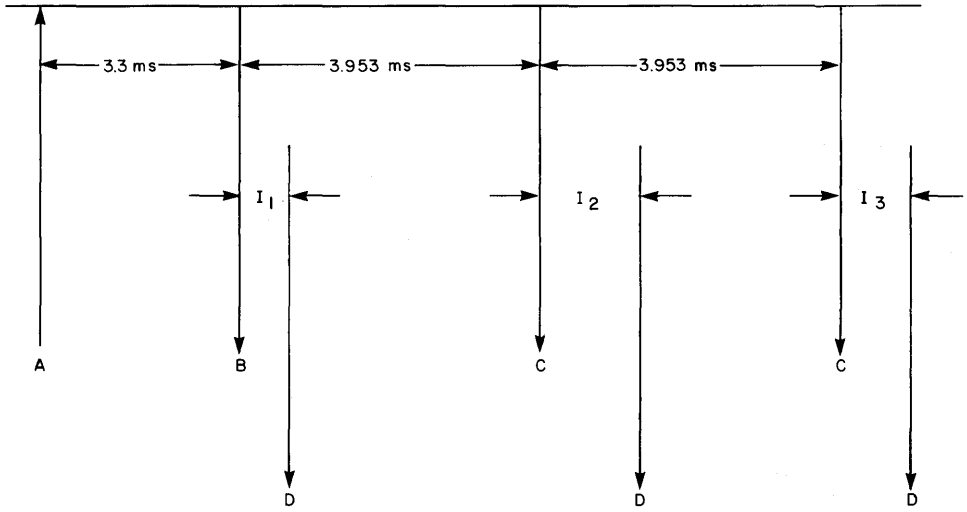


Figure 6

Each edge detect circuit 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 by the CPU.

$\overline{\text{INT REQ}}$ 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 PSU's 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 MK 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 $\overline{\text{PRI OUT}}$ output. $\overline{\text{PRI OUT}}$ of the PSU will be true (0V) only if $\overline{\text{PRI IN}}$ is true (0V) and SERVICE REQUEST is false. This means that when $\overline{\text{PRI IN}}$ is true (0V), $\overline{\text{PRI OUT}}$ and $\overline{\text{INT REQ}}$ are always at opposite levels. $\overline{\text{PRI OUT}}$ is connected to $\overline{\text{PRI IN}}$ on 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 be set if another interrupt request is in the process of being acknowledged anywhere in the system. 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.

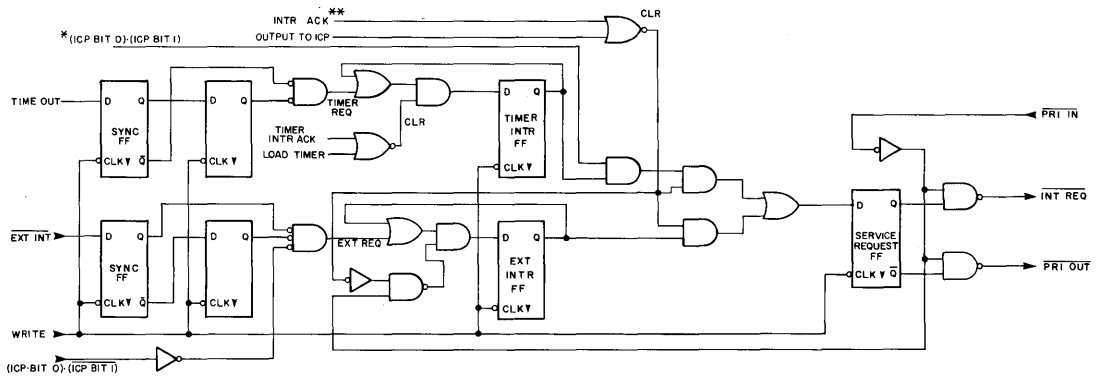
The SERVICE REQUEST flip-flop is cleared after an interrupt from the PSU has been acknowledged. It is also cleared whenever the PSU's interrupt control register is accessed by an output instruction.

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 Interrupt Control Port bit 1. This means that the PSU can detect a time out interrupt that occurred while the external interrupt was enabled in the PSU.

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.

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



* ICP-Interrupt Control Port
**1 During Event G in Fig 8

Figure 7

sequence which results in the execution of an interrupt service routine located at the memory address pointed to by the Interrupt Address Vector. Figures 8 and 9 illustrate the interrupt sequence for the two cases. Events occuring 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 occurrence, 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' inhibiting modification of interrupt priority logic. 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 (ICB = 0).

The current instruction fetch is not protected (not a privileged instruction).

EVENT F

The CPU generates the interrupt acknowledge sequence of ROMC states as follows:

ROMC STATE	
10	Inhibit modification of interrupt priority logic.
IC	No function
0F	Put lower byte of interrupt address vector on data bus
13	Put upper byte of interrupt address vector on data bus
00	Fetch instruction from memory (first instruction of interrupt service routine)

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 are cleared.

EVENT H

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

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 P0, 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. The interrupt vector is of the form:

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.

INTERRUPT SIGNALS TIMING

Timing for signals associated with the MK 3851 interrupt logic is shown in Figure 12.

TIMER INTERRUPT SEQUENCE

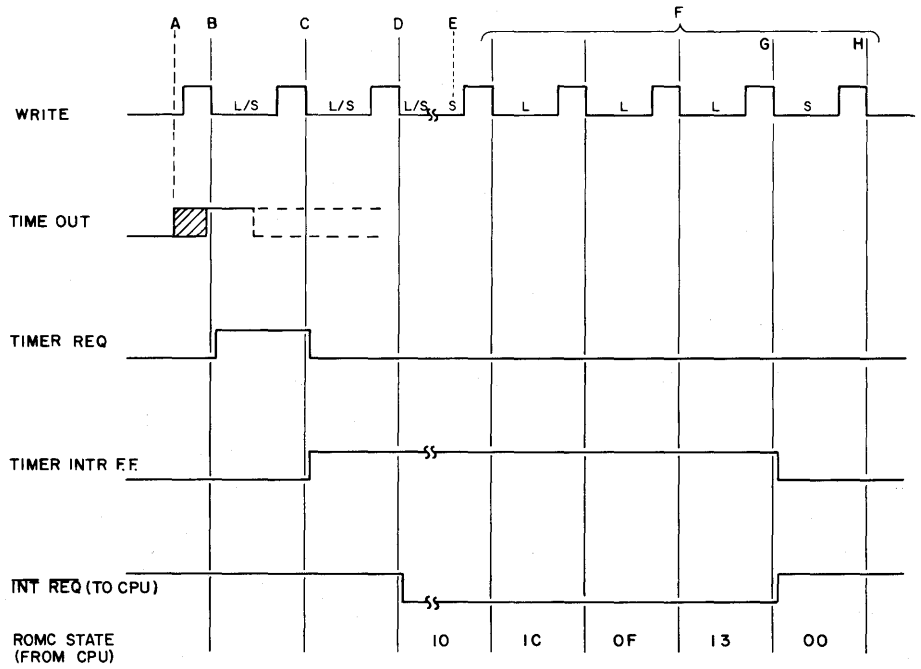


Figure 8

EXTERNAL INTERRUPT SEQUENCE

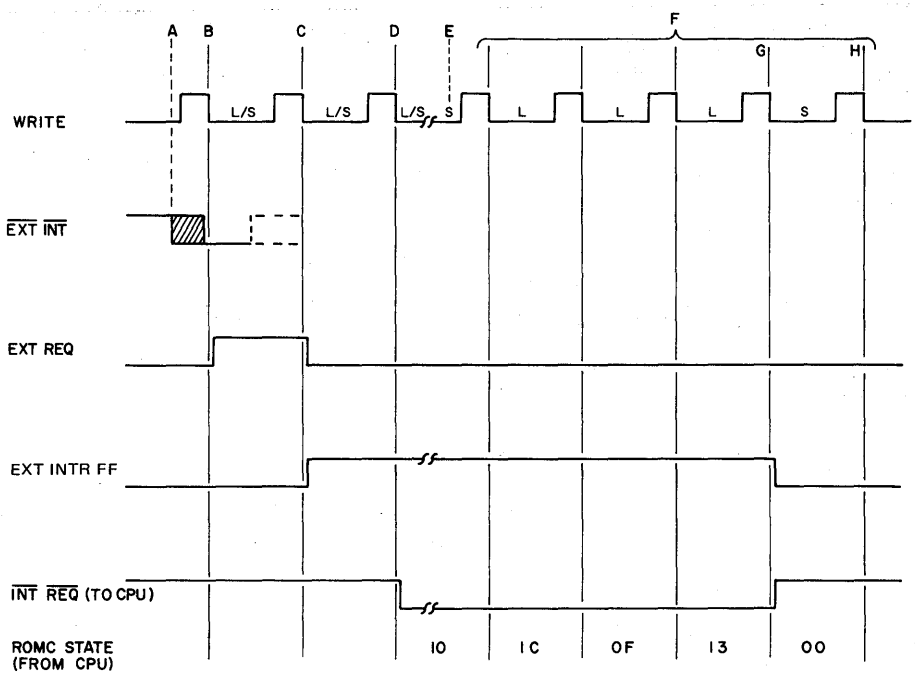


Figure 9

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 ..... −1mA 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		30	70	mA	f = 2MHz, Outputs Unloaded
I_{GG}	V_{GG} Current		10	18	mA	f = 2MHz, Outputs Unloaded

DC SIGNAL CHARACTERISTICS

V_{DD} = +5V ± 5%, V_{GG} = + 12V ± 5%, V_{SS} = 0V T_A = 0-70°C

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} = 1.6mA
	I_{IH}	Input High Current		1	μA	V_{IN} = V_{DD}, tri-state mode
	I_{OL}	Input Low Current		−1	μA	V_{IN} = V_{SS}, tri-state mode
CLOCK LINES (ΦWRITE)	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} = V_{DD}
PRIORITY IN AND CONTROL LINES (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} = V_{DD}
PRIORITY OUT (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} = 1mA
	I_L	Leakage Current		1	μA	V_{IN} = V_{DD}
DATA BUS DRIVE ($\overline{\text{DBDR}}$)	V_{OH}	Output High Voltage				Open Drain Output
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1.6mA
	I_L	Leakage Current		1	μA	V_{IN} = V_{DD}
EXTERNAL INTERRUPT ($\overline{\text{EXT INT}}$)	V_{IH}	Input High Voltage	3.5	15	Volts	
	V_{IL}	Input Low Voltage	−V_{SS}	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	−250	−750	μA	V_{IN} = V_{SS}
I/O PORT OPTION A	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = −30 μA
(STANDARD PULL-UP)	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} = 1.6mA
	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_L	Input Low Current		−1.6	mA	V_{IN} = 0.4V [4]
I/O PORT OPTION B (OPEN DRAIN)	V_{OH}	Output High Voltage				Open Drain Output
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1.6mA
	V_{IH}	Input High Voltage	2.9	V_{DD}	Volts	[3]
	V_{IL}	Input Low Voltage	V_{SS}	0.8	Volts	
	I_{IL}	Leakage Current		2	μA	V_{IN} = V_{DD}
I/O PORT OPTION (DRIVER PULL–UP)	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = −850 μA
	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1.6mA

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 typically 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.

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 timing diagrams.

Symbol	Parameters	Min	Typ	Max.	Units	Test Conditions/ Comments
P Φ	ΦPeriod	0.5		10	μs	
PW_1	ΦPulse Width	180		P Φ–180	ns	t_r, t_f = 50 ns typ
td_1	Φto WRITE + Delay	0		250	ns	C_L = 100pF
td_2	Φto WRITE – Delay	0		225	ns	C_L = 100pF
td_4	WRITE to DB Input Delay			2PΦ+1.0	μs	
PW_2	WRITE Pulse Width	P Φ–100		P Φ	ns	t_r,t_f = 50 ns typ.
PW_S	WRITE Period; Short		4P Φ			
PW_L	WRITE Period; Long		6P Φ			
td_3	WRITE to ROMC Delay			500	ns	
td_6	WRITE to DB Output Delay	2PΦ+100–td_2	2PΦ+200	2PΦ+800–td_2	ns	C_L = 100pF
td_7	WRITE to $\overline{DBDR}$ –Delay	2PΦ+100–td_2	2PΦ+200	2PΦ+800–td_2	ns	C_L = 100pF
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 = 100pF [3]
tpr_1	PRI IN to $\overline{INT\ REQ}$ – Delay			240	ns	C_L = 100pF [2]
tpr_2	PRI IN to $\overline{INT\ REQ}$ + Delay			430	ns	C_L = 100pF
tpd_1	PRI IN to $\overline{PRI\ OUT}$ –Delay			300	ns	C_L = 50pF
tpd_2	PRI IN to $\overline{PRI\ OUT}$ + Delay			365	ns	C_L = 50pF
tpd_3	WRITE to $\overline{PRI\ OUT}$ + Delay			700	ns	C_L = 50pF
tpd_4	WRITE to $\overline{PRI\ OUT}$ – Delay			640	ns	C_L = 50pF
t_{sp}	WRITE to Output Stable			1.7	μs	C_L = 50pF, Standard Pull-up
t_{od}	WRITE to Output Stable			1.7	μs	C_L = 50pF R_L = 12.5K Ω Open Drain
t_{dp}	WRITE to Output Stable		200	400	ns	C_L = 50pF, Driver Pull-up
t_{su}	I/O Setup Time	1.3			μs	
t_h	I/O Hold Time	0			μs	
t_{ex}	$\overline{EXT\ INT}$ Setup Time	400			ns	

Table 4

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 5pF typical on all pins except V_{DD}, V_{GG}, and V_{SS}.

PSU DATA BUS TIMING

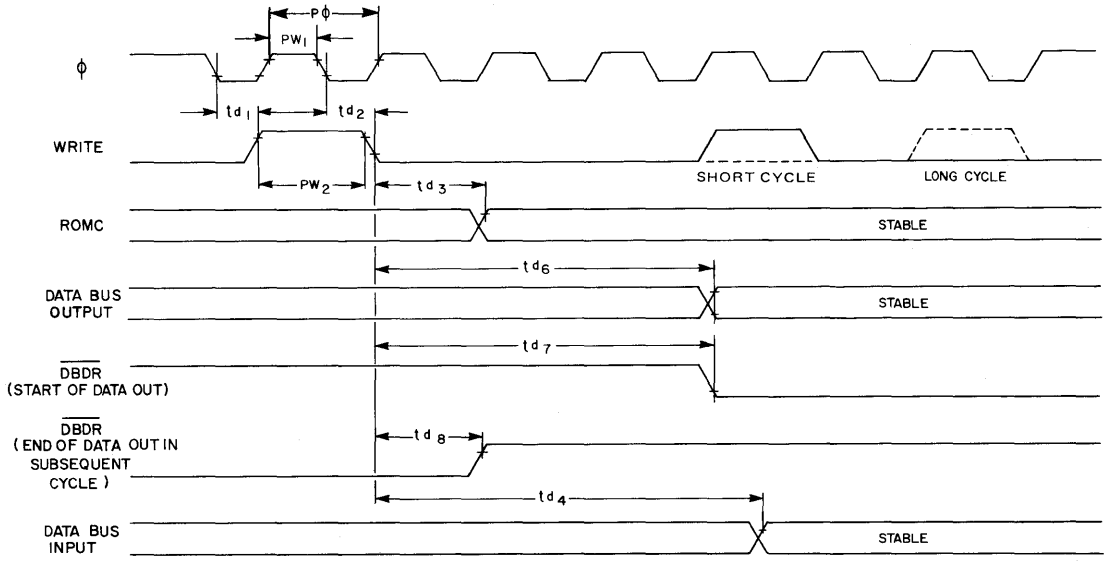
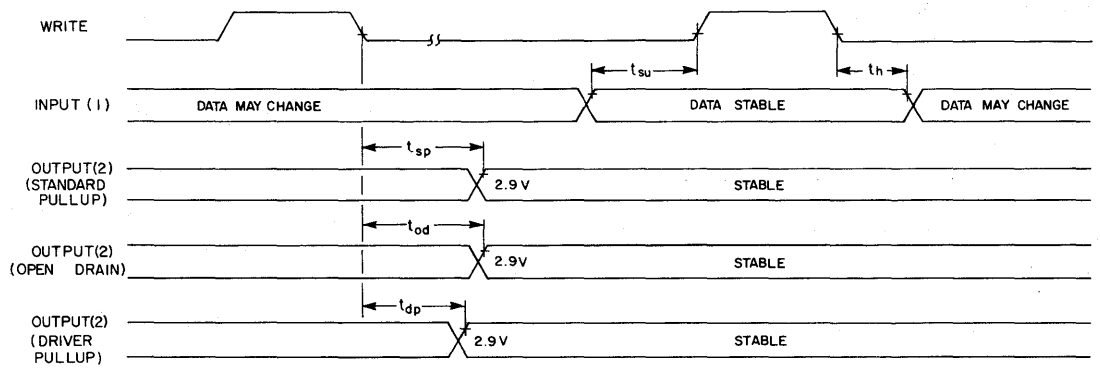


Figure 10

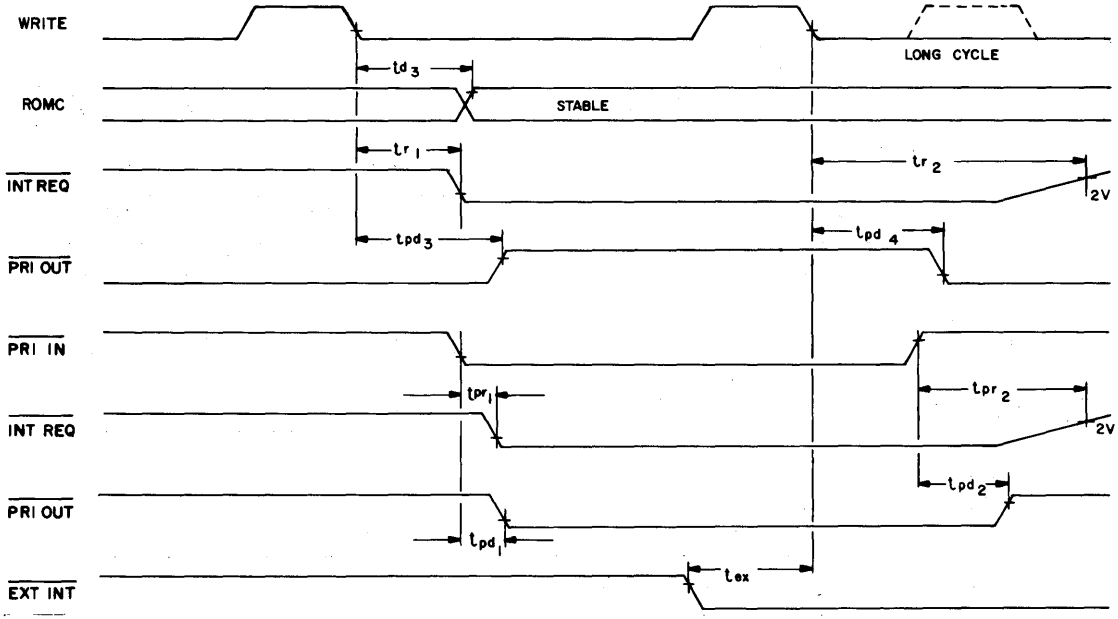
TIMING AT PSU I/O PORTS



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 11

INTERRUPT LOGIC SIGNALS TIMING



NOTE: Timing measurements are made at valid logic level of the signals referenced unless otherwise noted.

Figure 12

ORDER INFORMATION

PACKAGE SPECIFICATION

MK 3851N/12XXX	Plastic
MK 3851P/12XXX	Ceramic

The 12XXX number is assigned by MOSTEK when an MK 3851 is ordered. All mask options must also be specified as described in the next section.

OPTION SPECIFICATION

CARD FORMAT USED TO DEFINE MK 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 must 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 MOSTEK (Leave Blank)

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

IO is the decimal number (n) of the lowest of the four I/O port addresses selected where: $n = 4a, 1 \leqslant a \leqslant 63$

Port is 1 for Standard I/O
2 for Open Drain
3 for Driver Pull-up (Output Only)

Timer is the Timer/External Interrupt Address Vector (4 Hexadecimal digits)

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 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	Data 1	Data 2	Data 3. .	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) Same number base as address.
Data n specifies the data to be coded at ROM address (Add + n – 1) for 0 < n < = Bytes
Data 22 is a sequence number if an X SEQUENCE card has occurred

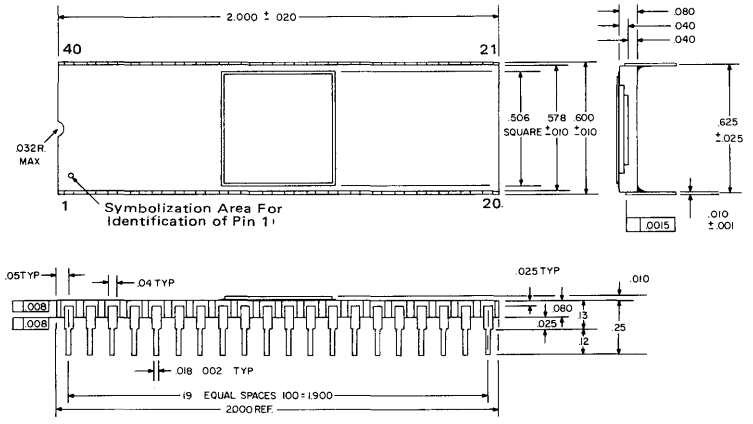
NOTE: All numeric fields must be right justified.

OTHER INPUT METHODS

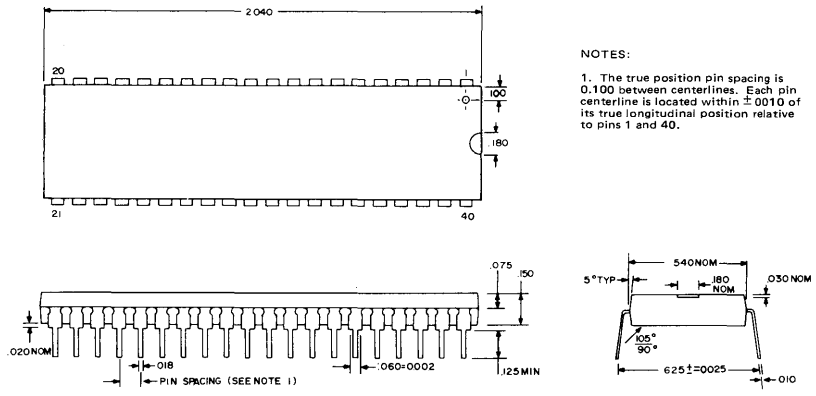
For information concerning other methods of input contact a MOSTEK representative.

PACKAGE DESCRIPTION

40-Pin Dual In-Line Ceramic



40-Pin Dual In-Line Plastic



NOTES:

1. The true position pin spacing is 0.100 between centerlines. Each pin centerline is located within ± 0.010 of its true longitudinal position relative to pins 1 and 40.

MOSTEK®

F8 MICROCOMPUTER DEVICES

Dynamic Memory Interface MK 3852

FEATURES

- ☐ Provides interface for 64K of dyanmic or static RAM
- ☐ Interfaces with MK3854 for DMA channel
- ☐ Provides automatic refresh for dynamic RAMs.
- ☐ Low Power Dissipation Typically Less Than 335mW

GENERAL DESCRIPTION

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 MK3852 DMI is manufactured using N-channel Isoplanar MOS technology.

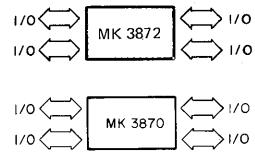
FUNCTIONAL PIN DESCRIPTION

Φ and WRITE are clock outputs from the 3850 CPU.

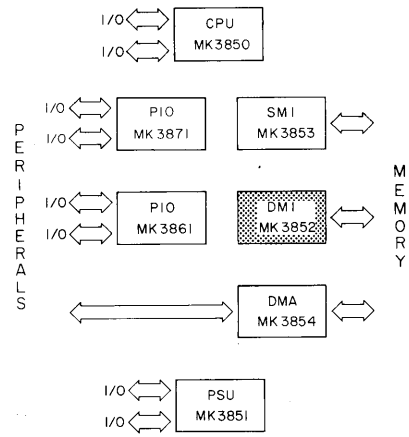
ROMC0 through ROMC4 are the control signals output by the 3850 CPU.

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
RAM WRITE	Write Line	Output (3-State)
ROMC0-ROMC4	Control Lines	Input
V_{SS}, V_{DD}, V_{GG}	Power Lines	Input

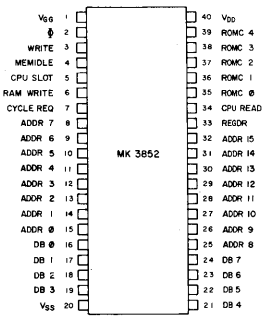
SINGLE CHIP 3870 MICROCOMPUTER FAMILY



F8 FAMILY



PIN CONNECTIONS



ADDR0 through ADDR15 are 16 address lines via which an address is transmitted to dyanmic RAM. The address may come from P0 or DC registers.

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.

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 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.

RAM WRITE. When low, this signal specifies that data is to be written into RAM locations. When high, this signal is off; that is, 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 RAM WRITE low.

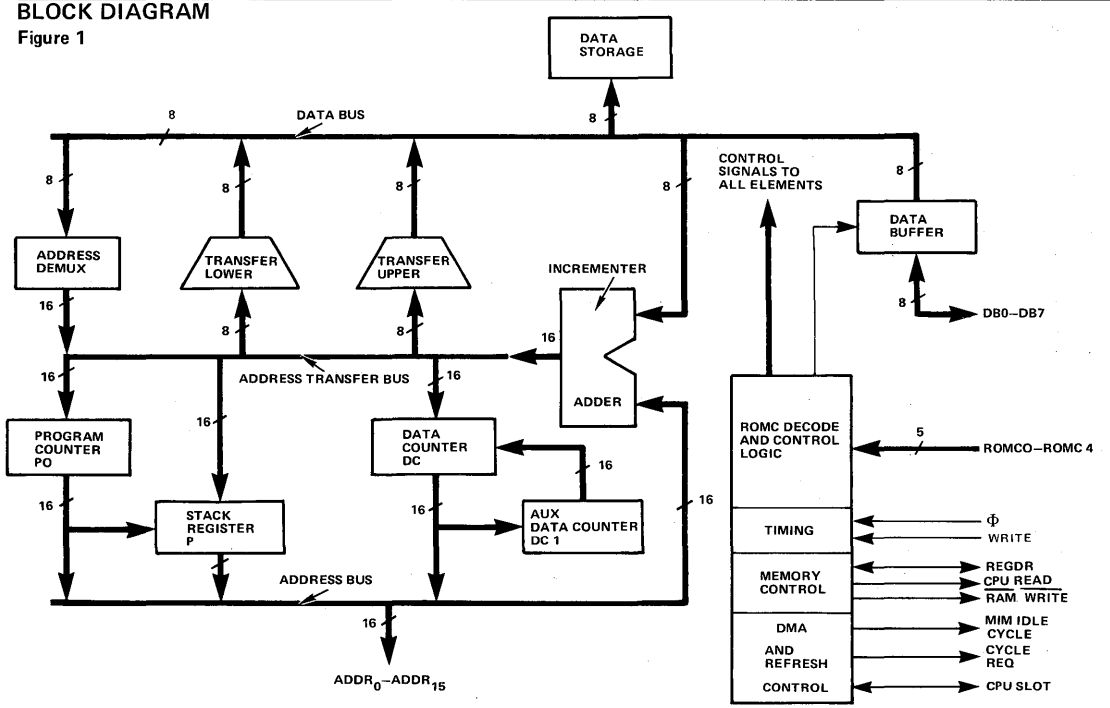
REGDR. This signal functions both as an input and an output. As an input, it can be clamped low by an external open collector gate. This prevents the 3852 DMI from placing a byte out of its P or DC registers onto the data bus. The DMI internally supplies a pull-up resistor. The signal, functioning as 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 P or DC registers or either of its two control registers (I/O ports) onto the data bus.

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.

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 RAM WRITE driver to be held in a high impedance state.

BLOCK DIAGRAM

Figure 1



DEVICE ORGANIZATION

This section describes the basic functional elements of the MK3852 DMI. These elements are shown in the DMI functional block diagram (figure 1).

PROGRAM COUNTER (PO) AND DATA COUNTER (DC AND DC1)

The MK3852 DMI addressing logic consists of 3 16-bit registers, the Program Counter (PO) and the Data Counters (DC and DC1).

The Program Counter will at all times address the memory word from which the next object program code must be fetched. The Data Counter (DC) addresses memory words containing individual data bytes or bytes within data tables to be used as operands.

It is important to note that the 3852 DMI has an auxiliary Data Counter (DC1). The contents of DC can be saved in DC1 by using the instruction XDC (exchange data counters). This instruction puts the contents of DC into DC1 and the contents of DC1 into DC. DC⇆DC1.

PO will always address the memory location out of which the next object program instruction byte will be read. If the instruction requires data (an operand) other than an immediate to be accessed DC must address memory. PO cannot be used to address a NON-immediate operand since PO is saving the address of the next instruction code.

THE STACK REGISTER P

The MK3852 DMI addressing logic contains a fourth 16-bit register called the stack register (P). The stack register is a buffer for the program counter PO. The contents of the stack register are never used directly to address memory.

The following instructions access P

LR K, P	Move the contents of P to the CPU scratchpad K registers
LR P,K	Move the contents of the CPU K scratchpad registers to P
PK	Save the contents of PO in P then move the contents of CPU scratchpad registers 12 and 13 to PO
PI H'aaaa'	Move the contents of PO to P then load the hexadecimal value into PO
POP	Move the contents of P to PO

In addition, when an interrupt is acknowledged, the contents of PO are saved in P.

MEMORY CONTROLS

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.

INCREMENTER ADDER LOGIC

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

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 complemented arithmetic) to the 16-bit value stored in address register.

The incrementer adder logic performs these two functions in the MK3852 DMI.

THE DATA BUS

The 8-bit data bus is the main path for transfer of information between the MK3850 CPU and other devices in the F8 microprocessor system.

ADDRESSABLE 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.

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.

OPERATIONAL DESCRIPTION

CLOCK TIMING

All timing within the MK3852 DMI is controlled by Φ and WRITE, which are input from the MK3850 CPU. Each machine cycle will contain either 4 Φ clock periods (short cycle) or 6 Φ clock periods (long cycle).

The WRITE clock refreshes and updates the MK3852 DMI. A machine cycle begins with the fall of the WRITE clock and the system control lines become stable shortly after the start of the cycle.

INSTRUCTION EXECUTION

The MK3852 DMI responds to signals which are output by the MK3850 CPU in the course of executing instruction cycles.

Table 1 summarizes the response of the MK3852 DMI to the ROMC states.

Table 1

ROMC STATES ROMC (Hexadecimal)	OPERATION PERFORMED	COMMENT
00	DB←((P0)); P0←P0 + 1	OP CODE, FETCH
01	DB←((P0)); P0←P0 + DB	BRANCH OFFSET FETCH
02	DB←((DC)); DC←DC + 1	
03	DB←((P0)); P0←P0 + 1	IMMEDIATE OPERAND FETCH
04	P0←P	
05	((DC))←DB; DC←DC + 1	
06	DB←DCU	
07	DB←PU	
08	P←P0; DB←H'00'; P0L, P0H←DB	EXTERNAL RESET
09	DB←DCL	
0A	DC←DC + DB	
0B	DB←PL	
0C	DB←((P0)); DCL←DB	
0D	P←P0 + 1	
0E	DB←((P0)); DCL←DB	
0F	NO OPERATION	
10	NO OPERATION	
11	DB←((P0)); DCU←DB	
12	P0L←DB; P←P0	
13	NO OPERATION	
14	P0U←DB	
15	PU←DB	
16	DCU←DB	
17	P0L←DB	
18	PL←DB	
19	DCL←DB	
1A	((pp))←DB or ((p))←DB	
1B	DB←(pp) or DB←((p))	
1C	NO OPERATION	
1D	DC⇆DC1	
1E	DB←P0L	
1F	DB←P0U	

Definitions:

DB - Data Bus
P0 - Program Counter
DC - Data Counter
DC1 - Aux Data Counter
P - Stack Register
pp - Two hex digits (long I/O port address)
p - One hex digit (short I/O port address)
1A - Interrupt address vector
L - Lower byte suffix
U - Upper byte suffix
( ) - Contents of
← - transfer to
⇆ - exchange

MEMORY ADDRESSING

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	ADDR13	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	1	1	1	1	1	1	1	1	1	1	1

PAGE SELECT may be the OR of ADDR 12, ADDR11 and ADDR10, which are shown 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.

3852 DMI ADDRESS REGISTERS' ADDRESS SPACE

As described in Table 1, certain ROMC states require the contents of the high order, or low order half of P0, P, or DC 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 P0, P or DC to be placed on the data bus. If REGDR is derived from the PAGE SELECT signal, 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, 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 P0, P, or DC, 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.

ADDRESS CONTENTIONS

When a 3852 DMI is present in an F8 system, memory addressing contentions are resolved as described in Memory Addressing, 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 DC 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 DC may now cause two devices to simultaneously place different data on the data bus. This may be illustrated as follows:

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

An address contention may arise if DC 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 DC 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 DC0	DMI DC0	DMI 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 DC loading instruction synchronize the DC'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 DC	DMI DC	P
		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 DC 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 DC of the PSU is out of step throughout the example.

INSTRUCTION		PSU P0	DMI DC	DMI 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.

TIMING 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 P0 on the address lines of ADDR0-ADDR15.

In effect, 3852 DMI logic beings by assuming that a memory read is to occur, with the memory address provided by P0.

While the contents of P0 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.

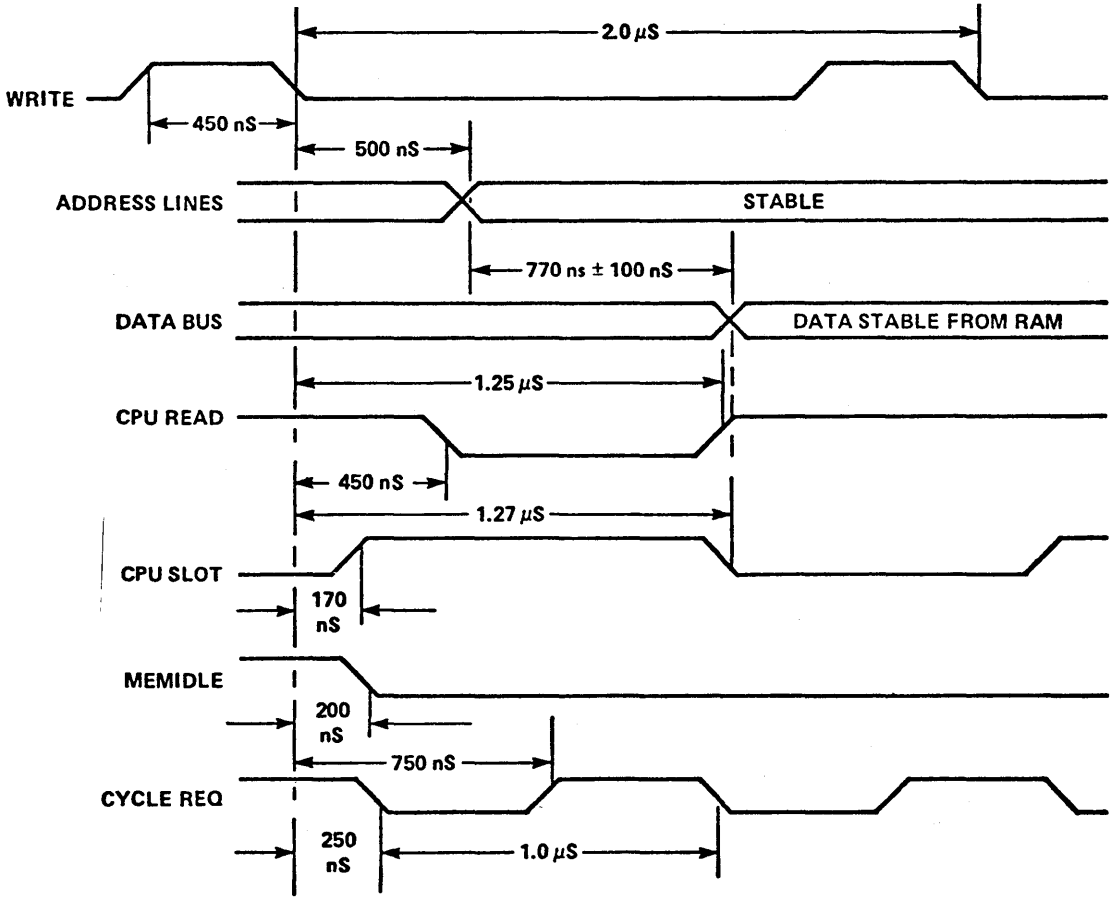
If the assumed logic proves to be correct, or if no memory access is to occur, then the second access period can be used for memory refresh or DMA.

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.

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

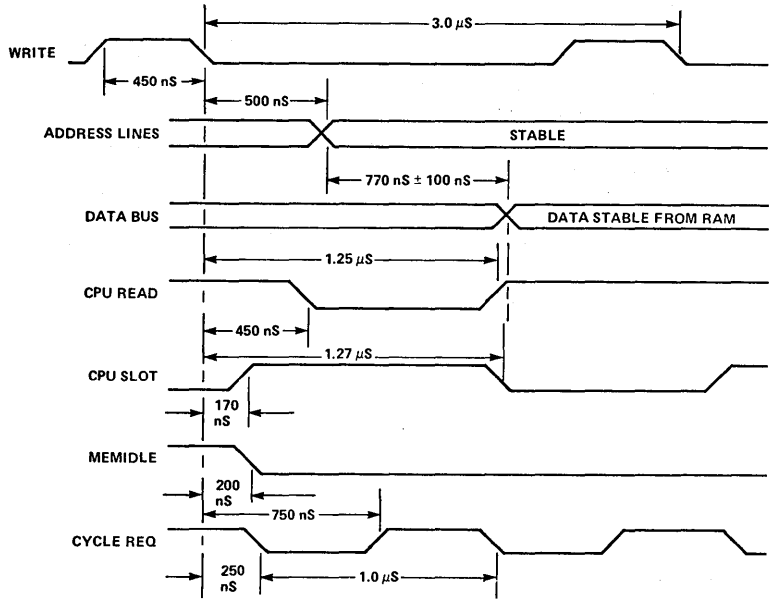
3852 DMI TIMING SIGNALS OUTPUT DURING A SHORT CYCLE MEMORY READ WITH ADDRESS FROM PO
Figure 2



1. The instruction fetch. The memory address originates in P0 and the instruction cycle is short. Timing is shown in Figure 2.

3852 DMI TIMING SIGNALS OUTPUT DURING A LONG CYCLE MEMORY READ, WITH ADDRESS OUT OF PROGRAM COUNTER

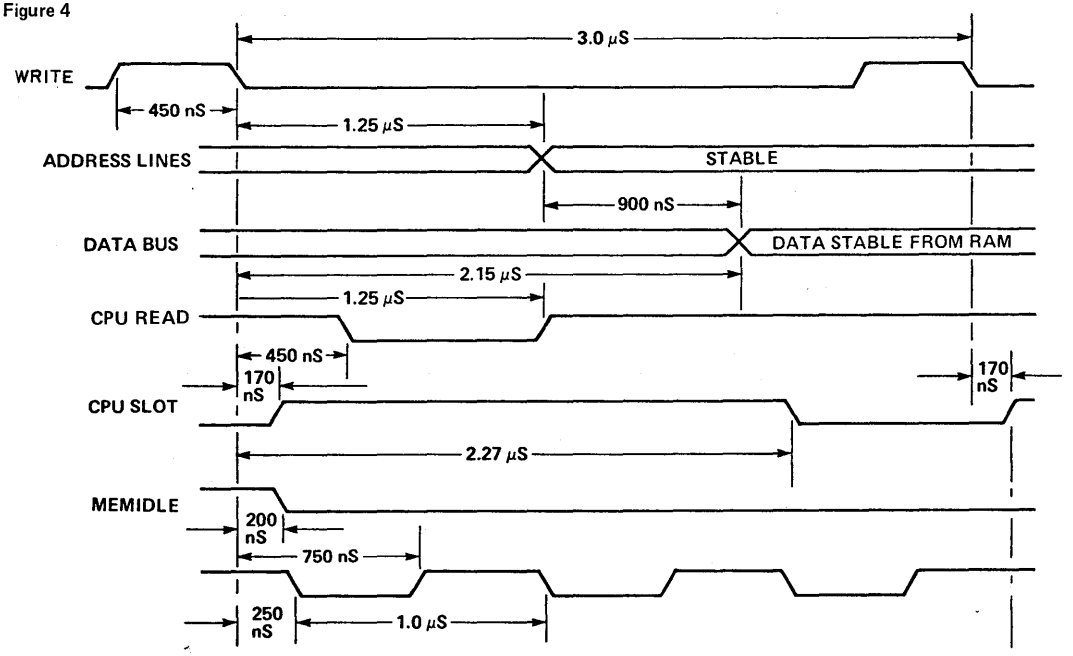
Figure 3



2. An immediate operand fetch. The memory address originates in P0, and the instruction cycle is long. Timing is shown in Figure 3.

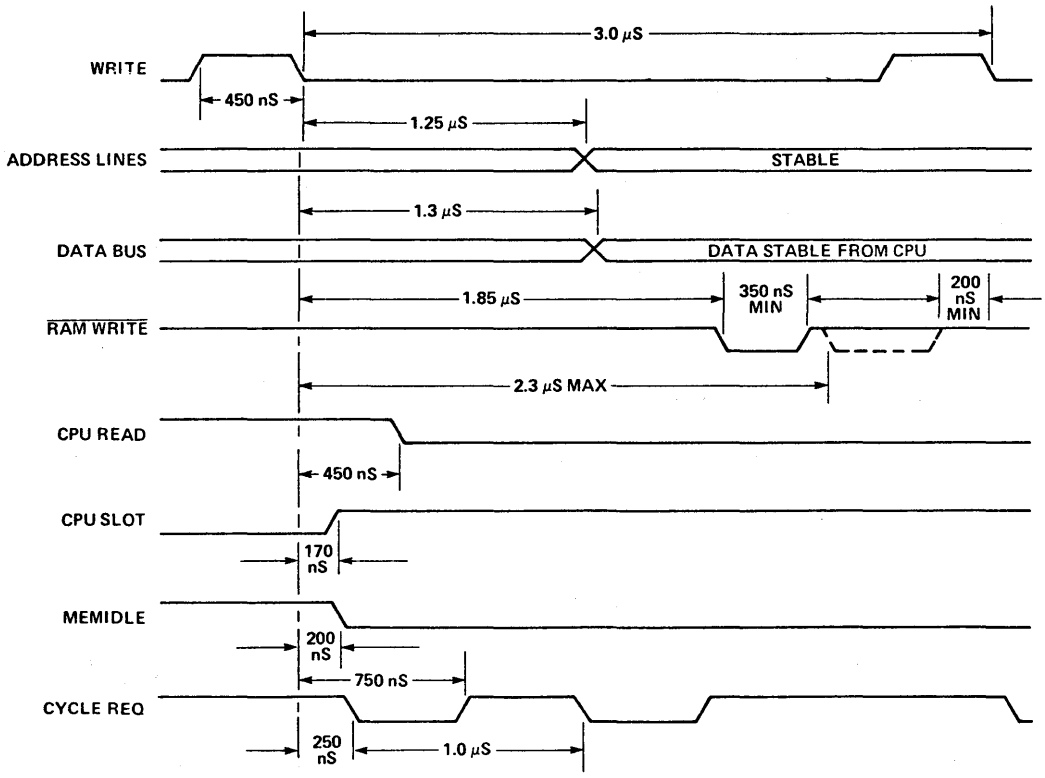
3852 DMI TIMING SIGNALS OUTPUT DURING A LONG CYCLE MEMORY READ, WITH ADDRESS OUT OF DATA COUNTER

Figure 4



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

3852 DMI TIMING SIGNALS OUTPUT DURING A WRITE TO MEMORY
Figure 5



4. A memory write. Data is written into the RAM memory location addressed by DC. Timing is shown in Figure 5.

CPU SLOT and MEM IDLE identify the way in which a memory access period is being used. Figures 6 and 7 illustrate the relationship.

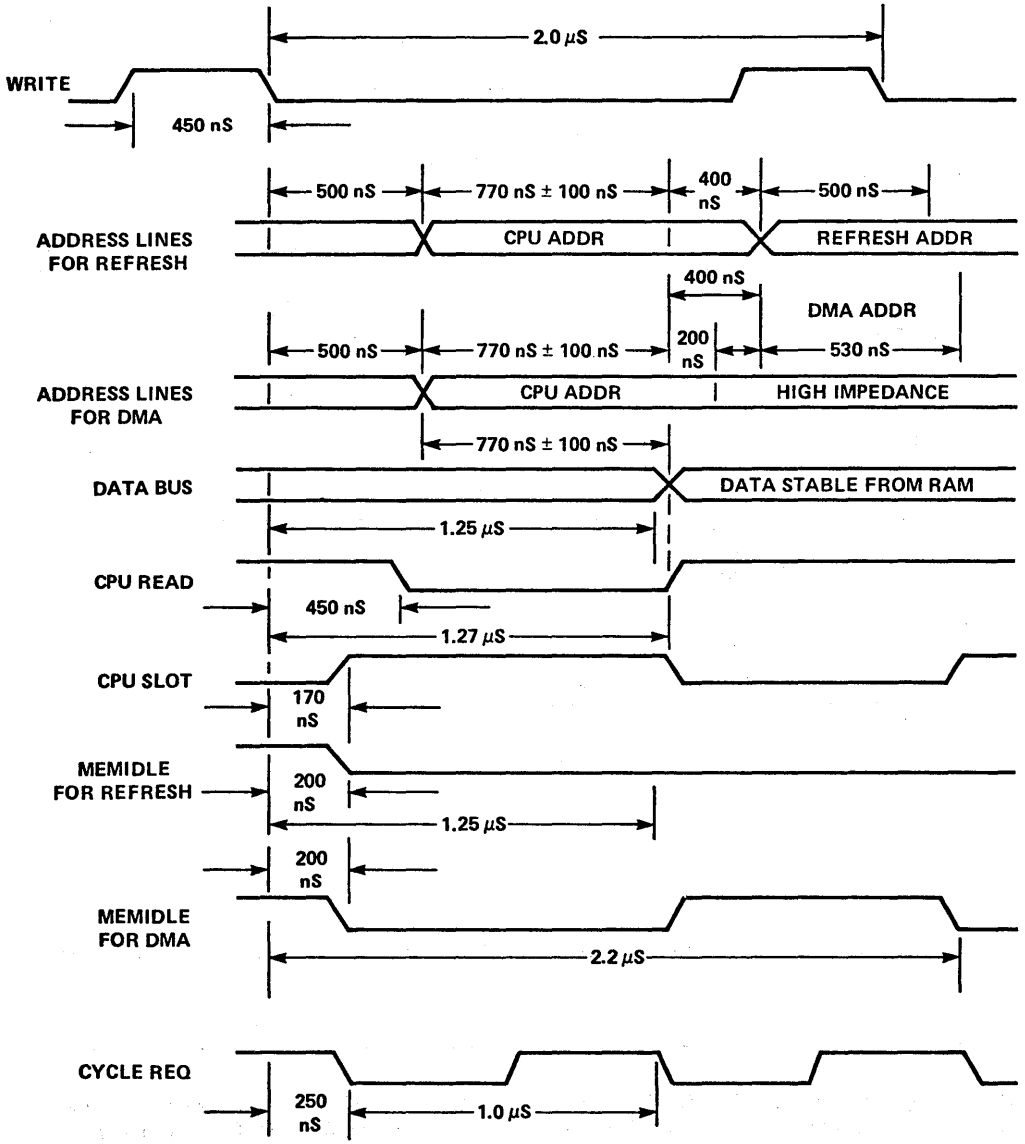
When the 3850 CPU is accessing memory, CPU SLOT is high; RAM WRITE and the address lines are driven at this time.

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.

3852 DMI logic is able to achieve two memory accesses within one instruction cycle by pursuing the logic sequence summarized in Table 2. 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.

TIMING FOR MEMORY REFRESH AND DMA DURING A SHORT CYCLE MEMORY READ, WITH ADDRESS OUT OF PROGRAM COUNTER
Figure 6



TIMING FOR MEMORY REFRESH AND DMA DURING A LONG CYCLE MEMORY READ, WITH ADDRESS OUT OF PROGRAM COUNTER

Figure 7

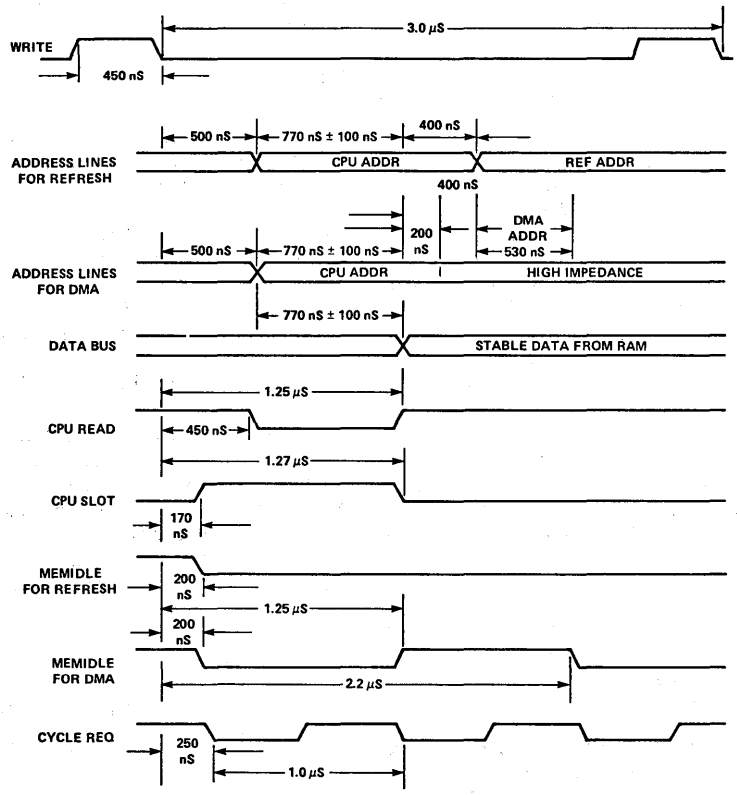


Table 2

OPERATION PERFORMED DURING INSTRUCTION CYCLE	FIRST ACCESS	SECOND ACCESS	THIRD ACCESS
No memory access, or read from memory addressed by P0. (See Figure 2.)	[P0]→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 P0. (See Figure 3.)	[P0]→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 P0, or read data from address register. (See Figure 4.)	[P0]→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 5.)	[P0]→A0 - A15	[DC]→A0 - A15	Access memory to write data. No DMA or refresh.
[ ] means "contents of register identified within square brackets.			

MEMORY REFRESH AND DIRECT MEMORY ACCESS

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

CYCLE REQ identified 2 or 3 memory access periods witnin an instruction cycle.

Either the first or the second access period, as summarized in Table 2, 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 register, 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 2, 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. The 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.

DATA OUTPUT BY RAM

Figures 2, 3, and 4 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.

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.

DATA OUTPUT BY THE 3852 DMI

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.

DATA INPUT TO RAM

Figure 5 provides worst case timing when data is written into RAM. Data is transferred through tri-state buffers on the data bus and into RAM.

$\overline{\text{RAM WRITE}}$ 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.

DATA INPUT TO THE 3852 DMI

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 onto 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.

INPUT/OUTPUT

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

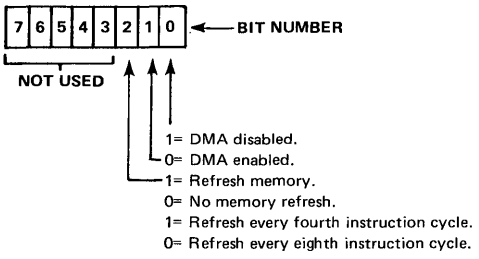
PORT ADDRESSES		FUNCTION
STANDARD	OPTION	
0C	EC	General purpose latch
0D	ED	Memory/DMA control
0E	EE	Not implemented
0F	EF	Not implemented

Option port addresses are used in F8 systems that include both a 3852 DMI and a 3853 SMI.

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:



SYSTEM INITIALIZATION

An F8 system is initialized by power on, or EXT $\overline{\text{RESET}}$ 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.

ELECTRICAL SPECIFICATIONS

ABSOLUTE MAXIMUM RATING (All voltages with respect to V_{SS})*

V_{GG}	+15V to −0.3V
V_{DD}	+7V to −0.3V
All other inputs and outputs	+7V to −0.3V
Operating temperature, T_A (Ambient)	0°C to +70°C
Storage temperature - Ambient (Ceramic)	−65°C to +150°C
Storage temperature - Ambient (Plastic)	−55°C to +125°C

*Stresses above 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 condition 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.

RECOMMENDED DC OPERATING CONDITIONS

($0°C \leq T_A \leq 70°C$)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{DD}	Supply	4.75	5.0	5.25	Volts	
V_{GG}	Voltage	11.4	12.0	12.6	Volts	
V_{SS}		0	0	0	Volts	

DC ELECTRICAL CHARACTERISTICS

($0°C \leq T_A \leq 70°C$) V_{DD} = +5V ± 5%; V_{GG} = +12V ± 5%; V_{SS} = 0V

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

DATA BUS (DB0-DB7)

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	
V_{IL}	Input Low Voltage	V_{SS}	.8	Volts	
V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	$I_{OH} = -100\mu A$
V_{OL}	Output Low Voltage	V_{SS}	.4	Volts	I_{OL} = 1.6mA
I_{IH}	Input High Current		1	μA	$V_{IN} = V_{DD}$, three-state mode
I_{IL}	Input Low Current		−1	μA	$V_{IN} = V_{SS}$, three-state mode
C_I	Capacitance		10	pF	Three-state mode

CONTROL LINES (ROMC0–ROMC4), AND CLOCK LINES (Φ, WRITE)

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	
V_{IL}	Input Low Voltage	V_{SS}	.8	Volts	
I_L	Leakage Current		1	μA	$V_{IN} = V_{DD}$
C_I	Capacitance		10	pF	

ADDRESS LINES (ADDR0-ADDR15) AND $\overline{\text{RAM WRITE}}$

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage	4.0	V_{DD}	Volts	$I_{OH} = -1mA$
V_{OL}	Output Low Voltage		.4	Volts	$I_{OL} = 3.2mA$
I_L	Leakage Current		1	μA	$V_{IN} = V_{DD}$

REGDR AND CPU SLOT

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	$I_{OH} = -300\mu A$
V_{OL}	Output Low Voltage	V_{SS}	.4	Volts	$I_{OL} = 2mA$
V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	Internal Pull-up to V_{DD}
V_{IL}	Input Low Voltage	V_{SS}	.8	Volts	
I_{IL}	Input Low Current	−3.5	−14.0	mA	V_{IN} = .4V and device outputting a logic "1"
I_L	Leakage Current		1	μA	$V_{IN} = V_{DD}$

CPU READ, MEMIDLE, AND CYCLE REQ

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH}=−1mA
V_{OL}	Output Low Voltage	V_{SS}	.4	Volts	$I_{OL} = 2mA$
I_L	Leakage Current		1	μA	$V_{IN} = V_{DD}$

AC ELECTRICAL CHARACTERISTICS

($0°C \leq T_A \leq 70°C$) (V_{DD} = +5V ± 5%; V_{GG} = +12V ± 5%; V_{SS} = 0V)

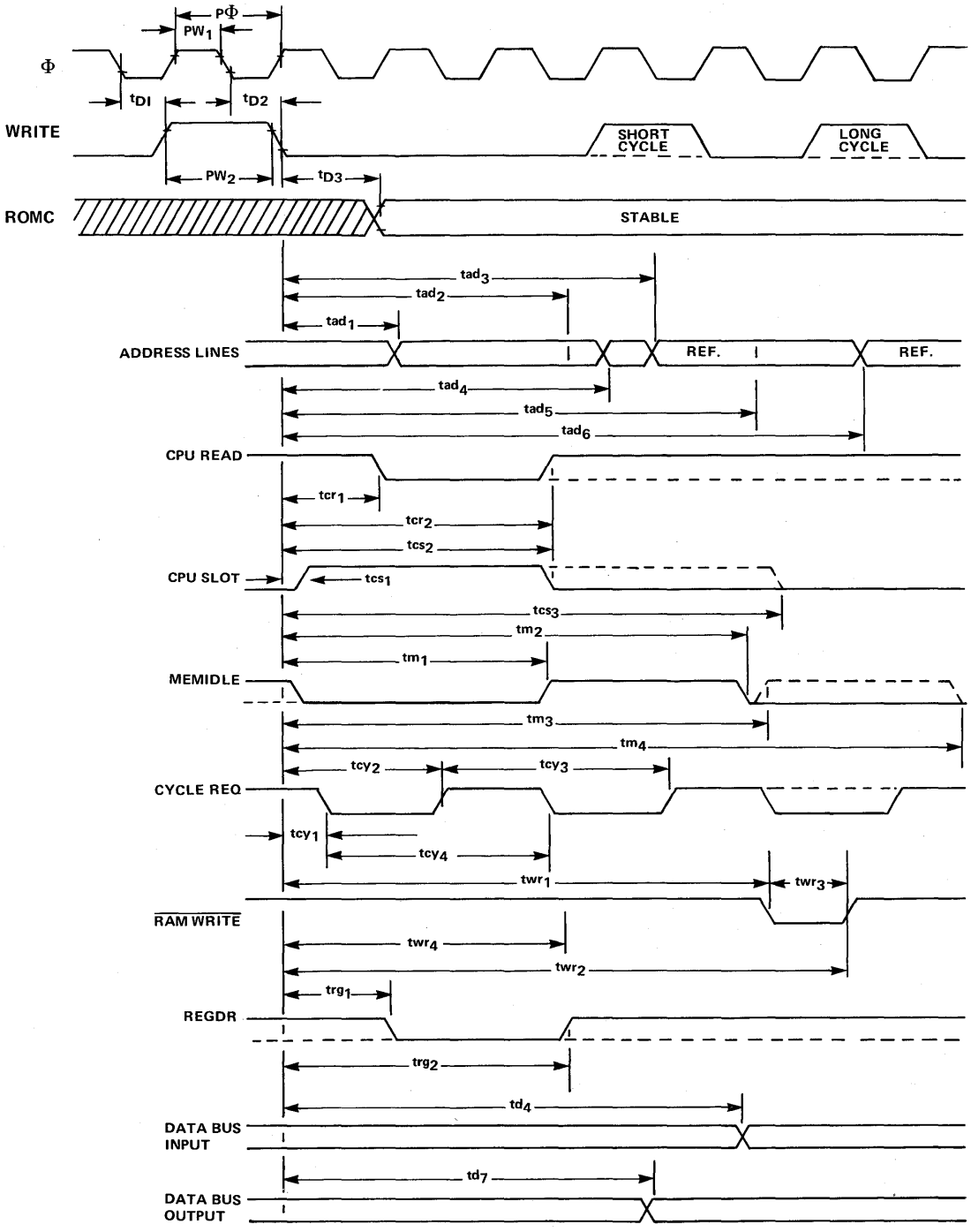
SYMBOL	PARAMETER	MIN	TYP	MAX	UNIT	TEST COND.
PΦ	Φ Clock Period	0.5		10	μS	
PW_1	Φ Pulse Width	180		PΦ–180	ns	
td_1	Φ to write + delay	0		300	ns	CL=100pF
td_2	Φ to write – delay	0		250	ns	CL=100pF
PW_2	Write Pulse Width	PΦ–100		PΦ	ns	
PW_S	Write Period; Short		4PΦ		ns	
PW_L	Write Period; Long		6PΦ		ns	
td_3	Write to ROMC Delay			750	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:

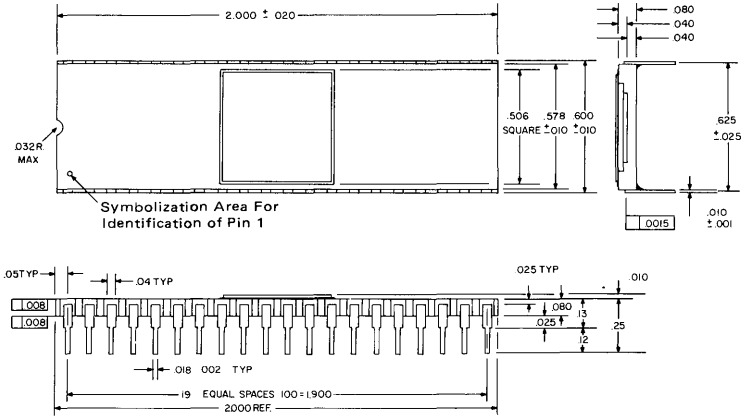
1. C_L = 50pF
2. C_L = 100pF.
3. C_L = 500pF.
4. CYCLE REQ is a divide-by-2 of Φ for all instructions except the STORE instruction.

TIMING DIAGRAM

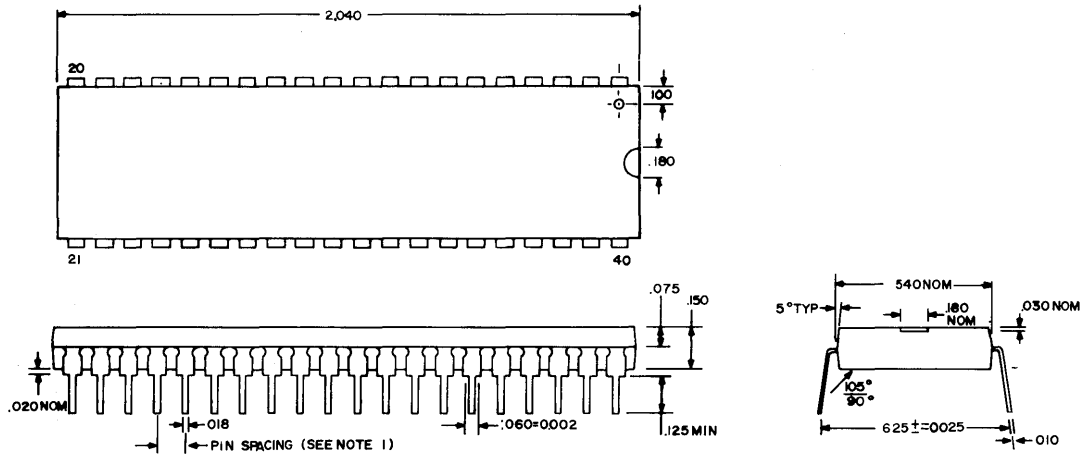
Figure 2



PACKAGE DESCRIPTION: 40-Pin Dual In-Line Ceramic Package



PACKAGE DESCRIPTION – 40-Pin Dual-in-Line Plastic Package



ORDERING INFORMATION

PART NUMBER	PACKAGE	TEMPERATURE RANGE
MK3852P	Ceramic	0°C to +70°C
MK3852N	Plastic	0°C to +70°C
MK3852P-10	Ceramic	-40°C to +85°C
MK3852N-10	Plastic	-40°C to +85°C

Static Memory Interface MK 3853

FEATURES

- ☐ Static Memory Interface to RAM, ROM or PROM
- ☐ Programmable Timer
- ☐ Programmable Interrupt Vectors for Timer and External Interrupts
- ☐ Low Power Dissipation Typically Less Than 335 mw

GENERAL DESCRIPTION

The MK 3853 Static Memory Interface (SMI) provides all necessary address lines and control signals to interface up to 65,536 bytes of Static RAM, ROM or PROM to an F8 microcomputer system. When quantities do not justify the mask charges for the MK 3851 PSU, or a fast turn around is of high importance, the MK 3853 SMI can be used to interface the F8 to EPROM or fusible-link bipolar PROMs. The 3853 SMI along with standard PROM can emulate the memory function of the 3851 PSU, while the 3861 provides the I/O ports, interrupt and timer features of the 3851 PSU. The 3853 is a high performance MOS/LSI circuit using N-channel Isoplanar technology.

FUNCTIONAL PIN DESCRIPTION

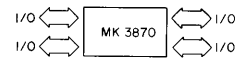
ADDR0-ADDR15 – The address bus provides the location of a memory read or write cycle.

DB0-DB7 – The Data Bus provides bi-directional communication between the 3850 F8 CPU and the 3853 SMI and all other F8 peripheral devices.

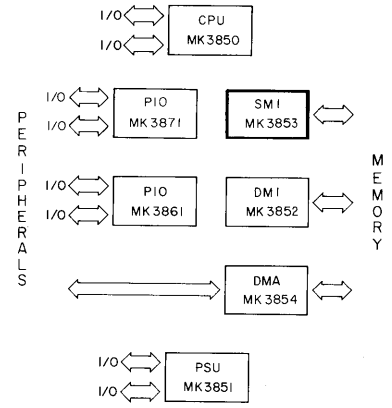
ROMC0-ROMC4 – These lines provide the 3853 SMI with control information from the 3850 F8 CPU.

PIN NAME	DESCRIPTION	TYPE
DB0-DB7	Data Bus Lines	Bi-directional, tri-state
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_{GG}, V_{DD}, V_{SS}	Power Supply Lines	Input

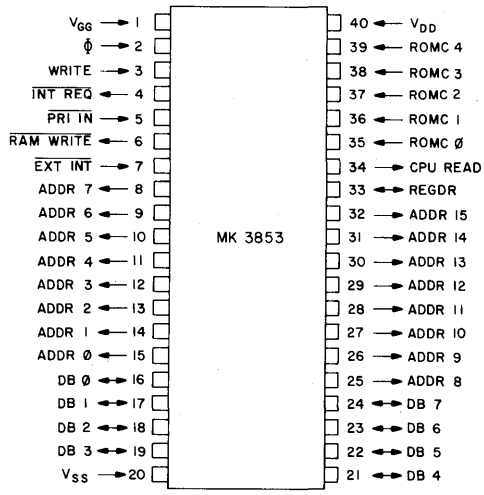
SINGLE CHIP MICROCOMPUTER



F8 FAMILY



PIN CONNECTIONS



BLOCK DIAGRAM

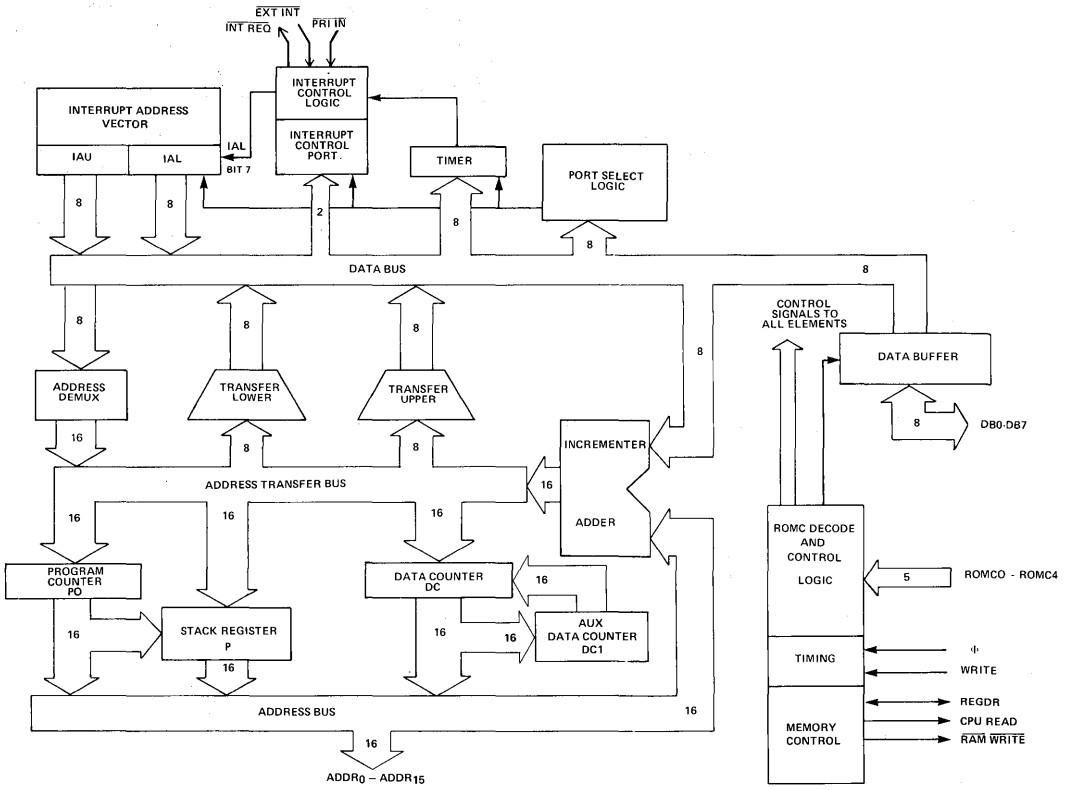


Figure 1

Φ– This is the system clock generated by the 3850 F8 CPU.

WRITE – This clock defines the machine cycle.

EXT INT – When an external circuit pulls this input "low", an external interrupt will be latched into the SMI if its interrupt control register has been set up to allow external interrupts. The SMI will then communicate this interrupt request to the CPU via INT REQ line.

PRI IN – This input signals the SMI that a higher priority peripheral has an interrupt request pending on the CPU. If the SMI has already requested an interrupt, the interrupt request will be maintained, but will not be serviced by the CPU until PRI IN is in the "low" state.

INT REQ – This is an open drain output that is wire ORed with the corresponding INT REQ outputs of all other peripherals to form the interrupt request input to the CPU.

RAM WRITE – This signal, when low, specifies that data from the data bus is to be written into a RAM location specified by the address bus.

CPU READ – This signal when high, specifies that data is to be read from the memory array interfaced to the SMI.

REGDR (OUTPUT/INPUT) – This signal functions both as an input and an output, to gate PO, DC, and I/O ports '0C' and '0D' onto the data bus at the proper time.

DEVICE ORGANIZATION

This section describes the basic functional elements of the MK 3853 SMI. These elements are shown in the SMI functional block diagram (figure 1).

PROGRAM COUNTER (PO) AND DATA COUNTERS (DC AND DC1)

The MK 3853 SMI addressing logic consists of 3 16-bit registers , the Program Counter (PO) and the Data Counters (DC and DC1)

The Program Counter will at all times address the memory word from which the next object program code must be fetched. The Data Counter (DC) addresses memory words containing individual data bytes or bytes within data tables to be used as operands.

It is important to note that the 3853 SMI has an auxiliary Data Counter (DC1). The contents of DC can be saved in DC1 by using the instruction XDC (exchange data counters). This instruction puts the contents of DC into DC1 and the contents of DC1 into DC. DC⇄DC1.

PO will always address the memory location out of which the next object program instruction byte will be read. If the instruction requires data (an operand) other than an immediate operand to be accessed, DC must address memory. PO cannot be used to address a NON-immediate operand since PO is saving the address of the next instruction code.

THE STACK REGISTER P

The MK 3853 SMI addressing logic contains a fourth 16-bit register called the stack register (P). The stack register is a buffer for the program counter PO. The contents of the stack register are never used directly to address memory.

The following instructions access P

LR K, P	Move the contents of P to the CPU scratchpad K registers
LR P,K	Move the contents of the CPU K scratchpad registers to P
PK	Save the contents of PO in P then move the contents of CPU scratchpad registers 12 and 13 to PO
PI H'aaaa'	Move the contents of PO to P then load the hexadecimal value into PO
POP	Move the contents of P to PO

In addition, when an interrupt is acknowledged, the contents of PO are saved in P.

MEMORY CONTROLS

The 3853 SMI provides three memory control outputs: RAM WRITE, CPU READ and REGDR.

RAM WRITE is used to control the read/write cycle of a static memory. RAM WRITE should be tied directly to the R/W line of the static memory.

CPU READ is a control signal that signifies that data is to be read out of a memory location. CPU READ and an externally generated address page select signal can be gated together to form a signal to enable the output of the memory array onto the F8 data bus.

REGDR is both an input and an output that is used to gate the program counter PO, data counter (DC), and I/O ports ('0C'H and '0D'H) onto the data bus at the proper time. If the 3851 PSU or 3852 DMI are not used in the system, then REGDR may be left open. If one or more 3851 PSU's are used in a system without the 3852 DMI, then the signal DBDR from all PSU's in the system should be tied together and gated through an open collector AND gate and tied to REGDR of the SMI. If the 3852 DMI and the 3853 SMI are used in a system without the 3851 PSU, then REGDR of the SMI should be left open and REGDR of the 3852 DMI should be tied low to prevent a data bus conflict when the PO and DC registers are output onto the data bus.

INCREMENTER ADDER LOGIC

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

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 complemented arithmetic) to the 16-bit value stored in address register.

The incrementer adder logic performs these two functions in the MK 3853 SMI.

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 MK 3850 CPU.

TIMER LOGIC

Every MK 3853 SMI 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 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.

THE DATA BUS

The 8-bit data bus is the main path for transfer of information between the MK 3850 CPU and other devices in the F8 microprocessor system.

ADDRESSABLE I/O PORTS

Every MK 3853 SMI has four, 8-bit I/O ports. Two of the I/O ports are used to store a programmable interrupt vector address. A third I/O port is assigned to a programmable timer while a fourth port is the Interrupt Control Port.

ROMC STATES

ROMC (Hexadecimal)	OPERATION PERFORMED	COMMENT
00	DB ← ((P0)) ; P0←P0 +1	0P CODE, FETCH
01	DB ← ((P0)) ; P0←P0 +DB	BRANCH OFFSET FETCH
02	DB ←((DC)); DC←DC+1	
03	DB ←((P0)) ; P0←P0+1	IMMEDIATE OPERAND FETCH
04	P0 ←P	
05	((DC))←DB ; DC←DC+1	
06	DB←DCU	
07	DB←P U	
08	P←P0 ; DB←H'00'; P0L, P0H← DB	EXTERNAL RESET
09	DB ←DCL	
0A	DC←DC+DB	
0B	DB←PL	
0C	DB←((P0)) ; DCL←DB	
0D	P←P0+1	
0E	DB←((P0)) ; DCL←DB	
0F	P←P0 ; DB←IAL ; P0L←DB	LOWER BYTE OF ADDRESS VECTOR
10	FREEZE INTERRUPT STATUS	PREVENT ADDRESS VECTOR CONFLICTS
11	DB←((P0)) ; DCU←DB	
12	P0L←DB ; P←P0	
13	DB←IAU; P0U←DB	UPPER BYTE OF ADDRESS VECTOR
14	P0U←DB	
15	PU←DB	
16	DCU←DB	
17	P0L ←DB	
18	PL←DB	
19	DCL←DB	
1A	((pp))←DB or ((p))←DB	
1B	DB←(pp)) or DB←((p))	
1C	NO OPERATION	
1D	DC ⇄ DC1	
1E	DB←P0L	
1F	DB←P0U	

Definitions

DB	Data Bus	IA	Interrupt address vector
P0	Program Counter	L	Lower byte suffix
DC	Data Counter	U	Upper byte suffix
DC1	Aux Data Counter	( )	Contents of
P	Stack Register	←	transfer to
pp	Two hex digits (long I/O port address)	⇄	exchange
p	One hex digit (short I/O port address)		

Table 1

The four I/O ports of the MK 3853 SMI have the following port addresses:

H'0C'	Programmable Interrupt Vector (upper byte)
H'0D'	Programmable Interrupt Vector (lower byte)
H'0E'	Interrupt Control Port
H'0F'	Programmable Timer

OPERATIONAL DESCRIPTION

CLOCK TIMING

All timing within the MK 3853 SMI is controlled by Φ and WRITE, which are input from the MK 3850 CPU. Each machine cycle will contain either 4 Φ clock periods (short cycle) or 6 Φ clock periods (long cycle).

The WRITE clock refreshes and updates the MK3853 SMI. A machine cycle begins with the fall of the WRITE clock and the system control lines become stable shortly after the start of the cycle.

INSTRUCTION EXECUTION

The MK 3853 SMI responds to signals which are output by the MK 3850 CPU in the course of executing instruction cycles.

Table 1 summarizes the response of the MK 3853 SMI to the ROMC states.

MEMORY ADDRESSING

Those ROMC states which specify a memory access call for only one memory device to respond to the memory access operation. However, every memory device responds to ROMC states that call for modification of program counter or data counter register contents. Consider two examples:

1. ROMC state 5 specifies that the data counter DC register contents must be incremented. Every memory device will simultaneously receive this ROMC state, and will simultaneously increment the contents of its DC 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 P0 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 P0 register, whether or not the P0 register contents are within the memory space of the device.

When all memory devices are connected to the 8-bit data bus of a MK 3850 CPU and are also connected to the ROMC control lines of the same CPU, the memory devices simultaneously receive the same ROMC state signals from the CPU and respond to ROMC states by identically modifying the contents of memory address registers. Therefore the P0 register on all memory devices contains identical information. The same holds true for DC and P registers.

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

1. Memory devices must receive the ROMC state signals from one CPU.

2. Memory array decoding must not overlap. (More than one memory device cannot have the same memory space).

3. The memory address contained in the specified register (P0 or DC) must be within the memory space of memory device.

TIMER LOGIC DIAGRAM

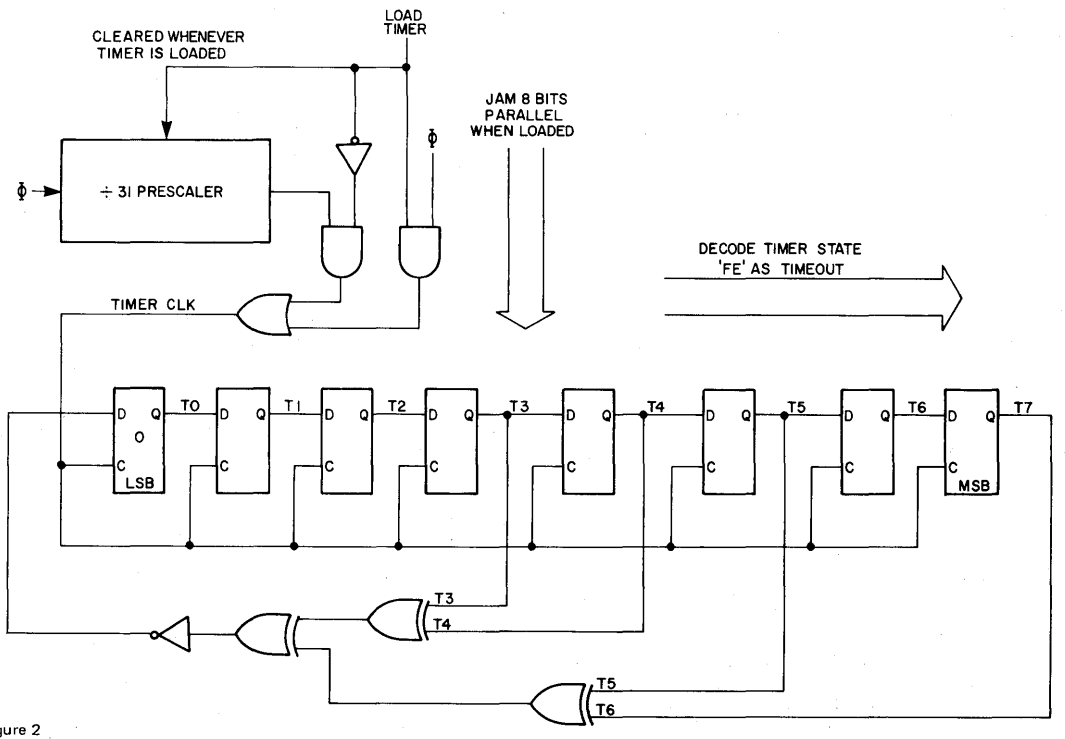


Figure 2

DATA INPUT TO THE SMI

The worst case timing for the MK 3853 SMI receiving data from the data bus is when the data must be added to a 16-bit number within the SMI's Incrementer Adder. This worst case corresponds to data coming from the accumulator in 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.

THE PROGRAMMABLE TIMER

The MK 3853 SMI has an 8-bit shift register, addressable as an I/O port, of which may be used as a programmable timer. Figure 2 illustrates the shift register logic and the exclusive OR feedback path.

Based on the logic illustrated in Figure 2, binary values in the range 0 through 254, when loaded into the timer, are converted into "timer counts", as shown in Table 2. Table 2 contains the actual (HEX) value loaded into the timer, and the column/row is the corresponding decimal number of time intervals the timer will take to time out. Data cannot be read out of the programmable timer I/O port.

Either the OUT or OUTS instruction is used to load "timer counts" into the programmable timer. 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 clock Φ) X (timer counts) X 31

For example, a value of 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 H'FF' loaded into the programmable timer will stop the timer. This is because the timer shift register feedback gates will always present a logic 1 to the D input of the LSB flip-flop (Fig. 2). Therefore, the timer will retain a value to H'FF' and a H'FE' will never be decoded to cause a time out.

The timer runs continuously unless it has been stopped by loading H'FF' into it. Upon timing out, the timer transmits an interrupt request to the interrupt logic. If proper interrupt logic conditions exist, the timer interrupt request is passed on to the CPU via INT REQ.

After the programmable timer has timed out it will again time out after 255 time counts. Therefore, if the programmable timer is simply left running, it will time out every 7905 Φ clock periods or every 3.9525 milliseconds for a 500 nanosecond clock.

Whenever the timer and timer interrupt are being set to time a new arrival, the timer should be loaded 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 a spurious interrupt. Time outs of the timer are latched in the interrupt logic of the SMI, even while timer interrupts are disabled. When the timer is enabled, an immediate interrupt acknowledge will occur if the continuous running timer timed out while timer interrupts were disabled.

If the timer is loaded just prior to enabling timer interrupts a spurious interrupt request will not exist when the timer interrupt is enabled.

Figure 3 illustrates a possible sequence for a timer which is initially loaded with H'C8' then allowed to run continuously.

CONVERSION OF TIMER COUNTS INTO TIMER CONTENTS

	0	1	2	3	4	5	6	7	8	9
0	7F	BF	5F	2F	97	CB	E5	72	39	1C
1	0E	87	43	A1	D0	E8	F4	7A	3D	1E
2	0F	07	03	01	00	80	C0	60	B0	D8
3	EC	F6	7B	BD	5E	AF	D7	6B	35	1A
4	0D	06	83	41	A0	50	A8	54	AA	55
5	2A	15	8A	C5	E2	F1	F8	7C	3E	9F
6	CF	E7	73	B9	5C	AE	57	2B	95	CA
7	65	32	99	CC	66	B3	59	2C	16	0B
8	05	02	81	40	20	10	08	84	C2	61
9	30	98	4C	26	13	89	44	22	11	88
10	C4	62	B1	58	AC	56	AB	D5	6A	B5
11	5A	AD	D6	EB	75	BA	DD	6E	B7	5B
12	2D	96	4B	A5	D2	E9	74	3A	9D	CE
13	67	33	19	8C	C6	63	31	18	0C	86
14	C3	E1	70	38	9C	4E	27	93	C9	E4
15	F2	79	BC	DE	EF	77	BB	5D	2E	17
16	8B	45	A2	51	28	14	0A	85	42	21
17	90	48	24	12	09	04	82	C1	E0	F0
18	78	3C	9E	4F	A7	D3	69	34	9A	4D
19	A6	53	29	94	4A	25	92	49	A4	52
20	A9	D4	EA	F5	FA	7D	BE	DF	6F	37
21	1B	8D	46	23	91	C8	64	B2	D9	6C
22	B6	DB	6D	36	9B	CD	E6	F3	F9	FC
23	7E	3F	1F	8F	47	A3	D1	68	B4	DA
24	ED	76	3B	1D	8E	C7	E3	71	B8	DC
25	EE	F7	FB	FD	FE	FF halts timer				

Each timer count = 15.5 μs at 2MHz

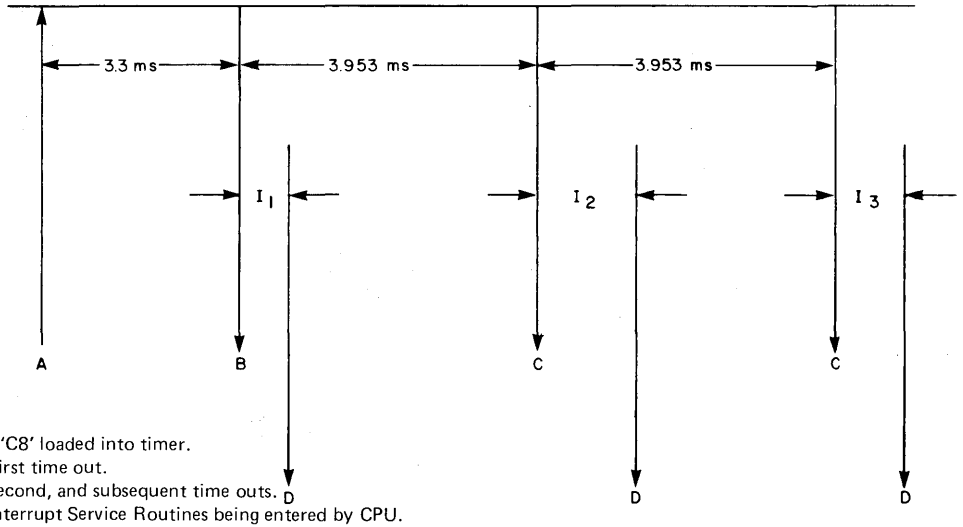
Table 2

INTERRUPT LOGIC ORGANIZATION

The interrupt Control Port has the I/O port address 'OE'H. Data is loaded into this register (I/O port) using an OUT or OUTS instruction. Data cannot be read from this port. The contents of the Interrupt Control Port are interpreted as follows:

CONTENTS OF INTERRUPT CONTROL PORT	FUNCTION
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

TIME OUT AND INTERRUPT REQUEST



A – H'C8' 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 reaching interrupt logic and service routines being entered by CPU. These time intervals depend on the number of privileged instructions encountered from the time $\overline{\text{INT REQ}}$ goes low. If none are encountered, 34P Φ is the minimum interval (17 μs for P Φ = 500ns)

Figure 3

In the preceding I/O port contents definitions x represents "don't care" bits.

Depending on the contents of the Interrupt Control Port, a MK 3853 SMI's interrupt control logic can be accepting timer interrupts, or external interrupts, or neither, but never both.

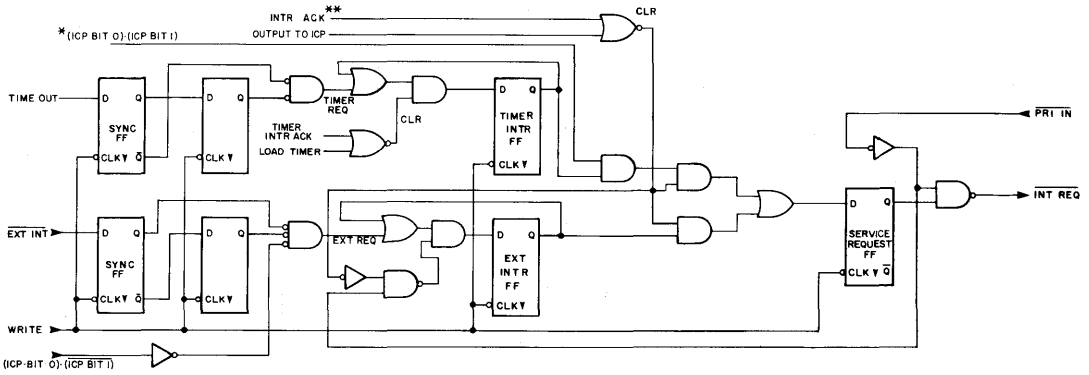
Figure 4 is a conceptual logic diagram of the SMI's interrupt logic. Between the $\overline{\text{EXT INT}}$ input or the time-out input and the output $\overline{\text{INT REQ}}$, there are 4 flip-flops. $\overline{\text{EXT INT}}$ and the time-out interrupt input each have 2 synchronizing flip-flops to detect the active edge.

Each edge detect circuit 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, provided that an interrupt from some other device is not being acknowledged by the CPU.

$\overline{\text{INT REQ}}$ is an open drain signal that is the NAND of PRI IN and SERVICE REQUEST. The $\overline{\text{INT REQ}}$ signal of several devices may be tied together so that any one can force the line to OV if it is requesting

INTERRUPT LOGIC



*ICP–INTERRUPT CONTROL PORT
**1 DURING EVENT G IN FIG 6

Figure 4

interrupt service. An internal pull-up to V_{DD} is provided by the MK 3850 CPU to the $\overline{\text{INT REQ}}$ input pin.

$\overline{\text{PRI IN}}$ is part of the interrupt priority chain. Each SMI has a $\overline{\text{PRI IN}}$ input but, it is important to note that the SMI does not have a $\overline{\text{PRI OUT}}$. This means that the SMI will be the last device in the daisy chain interrupt network. In a small system where only a CPU and a SMI are used, then $\overline{\text{PRI IN}}$ should be tied low. See Figure 5.

The SERVICE REQUEST flip-flop cannot be set if another interrupt request is in the process of being acknowledged anywhere in the system. If an interrupt request has been latched into the TIMER INTERRUPT flip-flop, or the EXTERNAL INTERRUPT flip-flop, the SMI logic waits until after the process of acknowledging the other interrupt before setting SERVICE REQUEST. This precaution is necessary to insure that the priority chain is not altered during acknowledgment. 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 SMI has been acknowledged. It is also cleared whenever the SMI 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 Interrupt Control Port bit 1. This means that the SMI can detect a time out interrupt that occurred while the external interrupt was enabled in the SMI.

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

INTERRUPT ACKNOWLEDGE SEQUENCE

Upon receiving an interrupt request, whether from an external source via $\overline{\text{EXT INT}}$ or from the internal timer, the SMI and CPU go through an interrupt sequence which results in the execution of an interrupt service routine located at the memory address pointed to by the Interrupt Address Vector. Figures 6 and 7 illustrate the interrupt sequence 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 identified an external interrupt. The rising edge of interval timer output indicates a timeout.

EVENT B

The synchronizing flip-flop in the SMI control logic changes state.

EVENT C

The timer interrupt, or external interrupt flip-flop goes true, indicating the local interrupt logic's acknowledgment of the interrupt. The timer interrupt flip-flop will always respond and save the time-out occurrence, 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 SMI, passing

INTERRUPT INTERCONNECTION

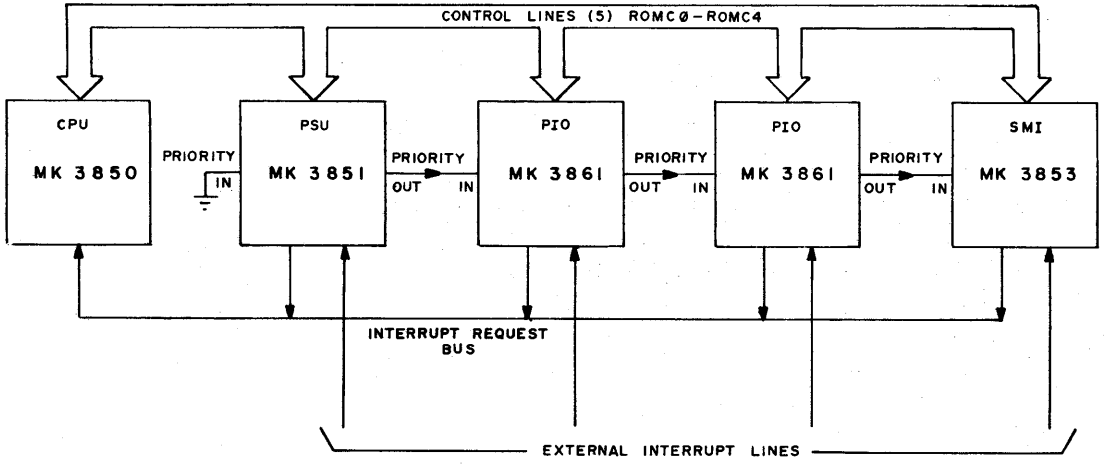


Figure 5

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 (time 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' inhibiting modification of interrupt priority logic. 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 (ICB = 0).

The current instruction fetch is not protected (not a privileged instruction).

EVENT F

The CPU generates the interrupt acknowledge sequence of ROMC states as follows:

ROMC STATE

10 Inhibit modification of interrupt priority logic.

IC No function

0F Put lower byte of interrupt address vector on data bus

13 Put upper byte of interrupt address vector on data bus

00 Fetch instruction from memory (first instruction of interrupt service routine)

EVENT G

At this point the CPU begins fetching the first instruction of the interrupt service routine. In the SMI interrupt logic, the SERVICE REQUEST flip-flop and the appropriate INTERRUPT REQUEST flip-flop are cleared.

EVENT H

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

INTERRUPT ADDRESS VECTOR

During the interrupt acknowledge, the interrupting SMI provides a 16-bit interrupt address vector. The CPU causes this vector to be loaded into P0, so that program execution can branch to the routine that handles this particular interrupt. Fifteen bits of the interrupt vector are programmable from I/O ports 'OC'H and 'OD'H. Bit 7 cannot be programmed. 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. The interrupt vector is of the form: WWWW, XXXX, 0YYY, ZZZZ for timer interrupt and WWWW, XXXX, 1YYY, ZZZZ for external interrupt

TIMER INTERRUPT SEQUENCE

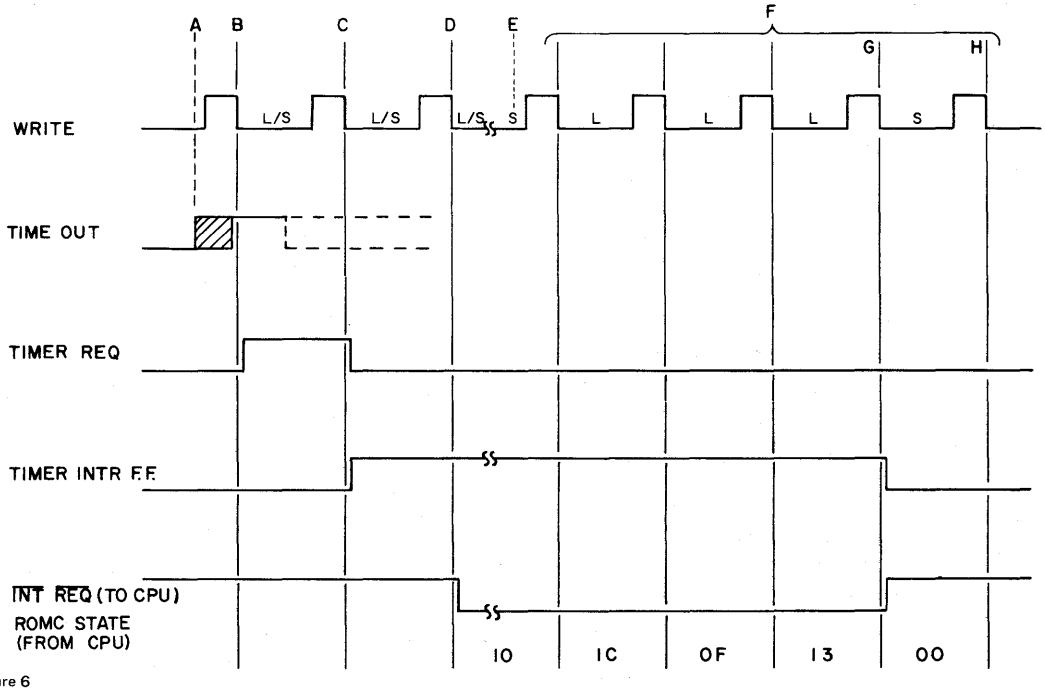


Figure 6

where W, X, Y and Z are the bits programmed by I/O ports '0C'H and '0D'H.

INTERRUPT SIGNALS TIMING

Timing for signals associated with the MK 3853 interrupt logic is shown in Figure 9.

EXTERNAL INTERRUPT SEQUENCE

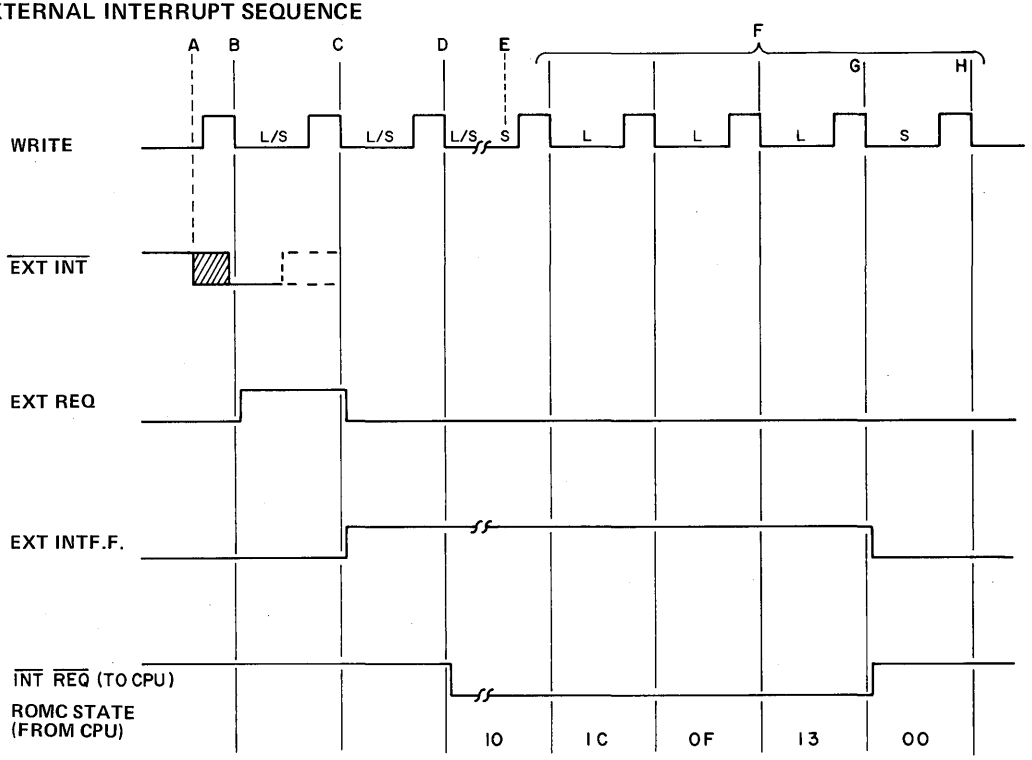


Figure 7

ELECTRICAL SPECIFICATIONS

ABSOLUTE MAXIMUM RATING (All voltages with respect to V_{SS}) *

V_{GG} ..... +15V to −0.3V
V_{DD} ..... +7V to −0.3V
All other inputs and outputs ..... +7V to −0.3V
Operating temperature, T_A (Ambient) ..... 0°C to +70°C
Storage temperature - Ambient (Ceramic) ..... −65°C to +150°C
Storage temperature - Ambient (Plastic) ..... −55°C to +125°C

*Stresses above 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 condition 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.

RECOMMENDED DC OPERATING CONDITIONS

(0°C ≤ T_A ≤ 70°C)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{DD}	Supply	4.75	5.0	5.25	Volts	
V_{GG}	Voltage	11.4	12.0	12.6	Volts	
V_{SS}		0	0	0	Volts	

DC ELECTRICAL CHARACTERISTICS

($0°C \leq T_A \leq 70°C$) ($V_{DD} = +5V \pm 5\%$; $V_{GG} = +12V \pm 5\%$; $V_{SS} = 0V$)

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

DATA BUS (DB0-DB7)

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	
V_{IL}	Input Low Voltage	V_{SS}	.8	Volts	
V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	$I_{OH} = -100\mu A$
V_{OL}	Output Low Voltage	V_{SS}	.4	Volts	$I_{OL} = 1.6mA$
I_{IH}	Input High Current		1	μA	$V_{IN} = V_{DD}$, three-state mode
I_{IL}	Input Low Current		−1	μA	$V_{IN} = V_{SS}$, three-state mode
C_I	Capacitance		10	pF	Three-state mode

PRIORITY IN ($\overline{\text{PRI IN}}$), CONTROL LINES (ROMCO–ROMC4) AND CLOCK LINES (Φ, WRITE)

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	
V_{IL}	Input Low Voltage	V_{SS}	.8	Volts	
I_L	Leakage Current		1	μA	$V_{IN} = V_{DD}$
C_I	Capacitance		10	pF	

ADDRESS LINES (ADDR0-ADDR15) and $\overline{\text{RAM WRITE}}$

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	$I_{OH} = -1mA$
V_{OL}	Output Low Voltage		.4	Volts	$I_{OL} = 3.2mA$
I_L	Leakage Current		1	μA	$V_{IN} = V_{DD}$

INTERRUPT REQUEST ($\overline{\text{INT REQ}}$)

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage				Open Drain Output [1]
V_{OL}	Output Low Voltage	V_{SS}	.4	Volts	$I_{OL} = 1mA$
I_L	Leakage Current		1	μA	$V_{IN} = V_{DD}$

EXTERNAL INTERRUPT ($\overline{\text{EXT INT}}$)

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	3.5	15	Volts	
V_{IL}	Input Low Voltage	V_{SS}	1.2	Volts	
V_{IC}	Input Clamp Voltage		15	Volts	$I_{IH} = 185\ \mu A$
I_{IH}	Input High Current		10	μA	$V_{IN} = V_{DD}$
I_{IL}	Input Low Current	−250	−750	μA	$V_{IN} = V_{SS}$

Notes:
1. Pull-up resister to V_{DD} on CPU.

REGDR

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	$I_{OH} = -300\ \mu A$
V_{OL}	Output Low Voltage	V_{SS}	.4	Volts	$I_{OL} = 2mA$
V_{IH}	Input High Voltage	3.5	V_{DD}	Volts	Internal Pull-up to V_{DD}
V_{IL}	Input Low Voltage	V_{SS}	.8	Volts	
I_{IL}	Input Low Current	−3.5	−14.0	mA	V_{IN} = .4V and Device outputting a logic "1"
I_L	Leakage Current		1	μA	$V_{IN} = V_{DD}$

CPU READ

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS;
V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	$I_{OH} = -1mA$
V_{OL}	Output Low Voltage	V_{SS}	.4	Volts	$I_{OL} = 2mA$
I_L	Leakage Current		1	μA	$V_{IN} = V_{DD}$

AC ELECTRICAL CHARACTERISTICS

($0°C \leq T_A \leq 70°C$) ($V_{DD} = +5V \pm 5\%$; $V_{GG} = +12V \pm 5\%$; $V_{SS} = 0V$)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST COND
$P\Phi$	Φ CLOCK Period	0.5		10	μS	
PW_1	Φ Pulse Width	180		PΦ−180	nS	
td_1	Φ to write + delay	0		300	nS	CL = 100pF
td_2	Φ to write − delay	0		250	nS	CL = 100pF
PW_2	Write Pulse Width	PΦ−100		PΦ	nS	
PW_S	Write Period; Short		4 PΦ		nS	
PW_L	Write Period; Long		6 PΦ		nS	
td_3	Write to ROMC Delay			750	nS	
td_4	Write to DB Input Delay			2PΦ+1.0	μS	
td_6	Write to DB Output Delay	$2P\Phi+100-td_2$	2PΦ+200	$2P\Phi+800-td_2$	nS	CL = 100pF
tad_1	Address delay if P0 (Instruction by immediate data)	50	300	500	nS	C_L = 500pF
tad_2	Address delay if DC (Operand fetch) or WRITE cycle	$2\Phi+50-td_2$		$2P\Phi+620-td_2$	nS	C_L = 500pF
tcr_1	CPU READ − Delay	50	250	450	nS	50pF
tcr_2	CPU READ + Delay	$2P\Phi+50-td_2$		$2P\Phi+400-td_2$	nS	50pF
tw_1	$\overline{\text{RAM WRITE}}$ − Delay	$4P\Phi+50-td_2$		$4P\Phi+450-td_2$	nS	500pF
tw_2	$\overline{\text{RAM WRITE}}$ + Delay	$5P\Phi+50-td_2$		$5P\Phi+300-td_2$	nS	500pF
tw_p	$\overline{\text{RAM WRITE}}$ Pulse Width	350		PΦ	nS	500pF
trg_1	WRITE to REGDR −Delay	70	300	500	nS	50pF
trg_2	WRITE to REGDR + Delay	$2P\Phi+80-td_2$		$2P\Phi+500-td_2$	nS	50pF

AC ELECTRICAL CHARACTERISTICS (Continued)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
tr_1	WRITE to $\overline{\text{INT REQ}}$ – Delay			430	nS	C_L = 100 pF [1]
tr_2	WRITE to $\overline{\text{INT REQ}}$ + Delay			1.65	μs	C_L = 100 pF [3]
tpr_1	$\overline{\text{PRI IN}}$ to $\overline{\text{INT REQ}}$ – Delay			240	nS	C_L = 100 pF [2]
tpr_2	$\overline{\text{PRI IN}}$ to $\overline{\text{INT REQ}}$ + Delay			1.5	μs	C_L = 100 pF
tex	$\overline{\text{EXT INT}}$ Setup Time	400			nS	

NOTES: 1. Assume PRIORITY IN was enabled ($\overline{\text{PRI IN}}$ = 0) in previous F8 cycle before interrupt is detected in the SMI.

2. SMI has interrupt pending before $\overline{\text{PRIORITY IN}}$ is enabled.

3. Assume pin tied to $\overline{\text{INT REQ}}$ input of 3850 CPU.

TIMING DIAGRAM

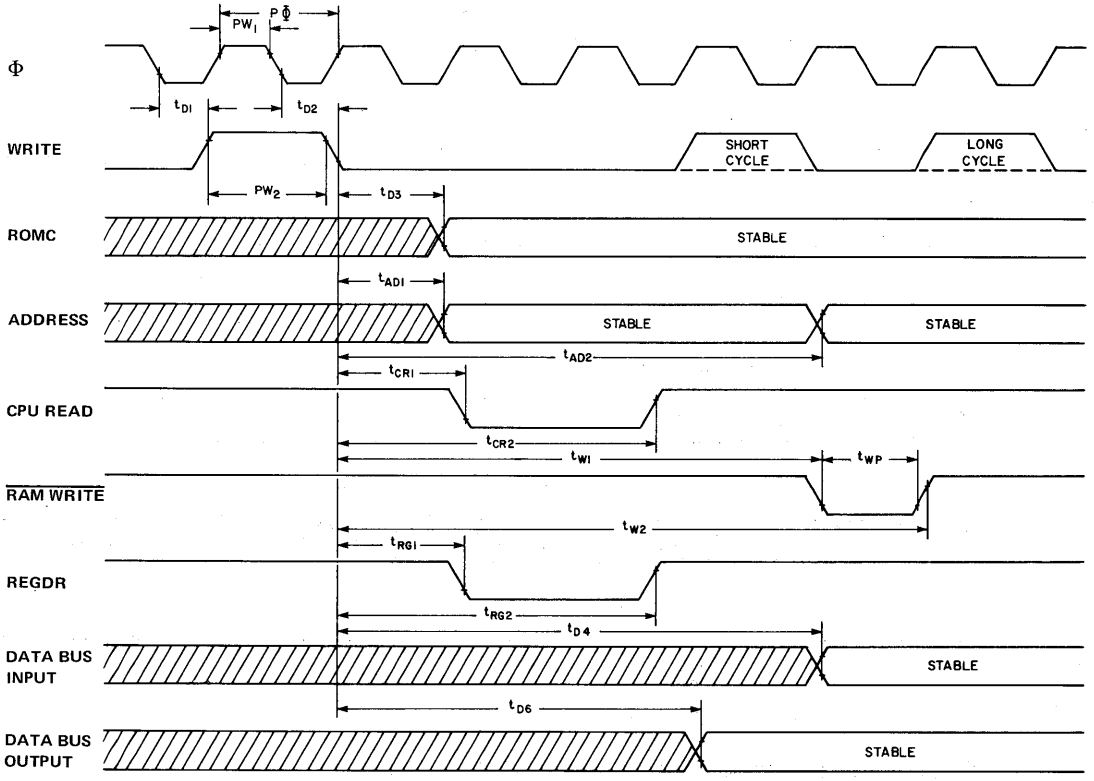


Figure 8

TIMING DIAGRAM

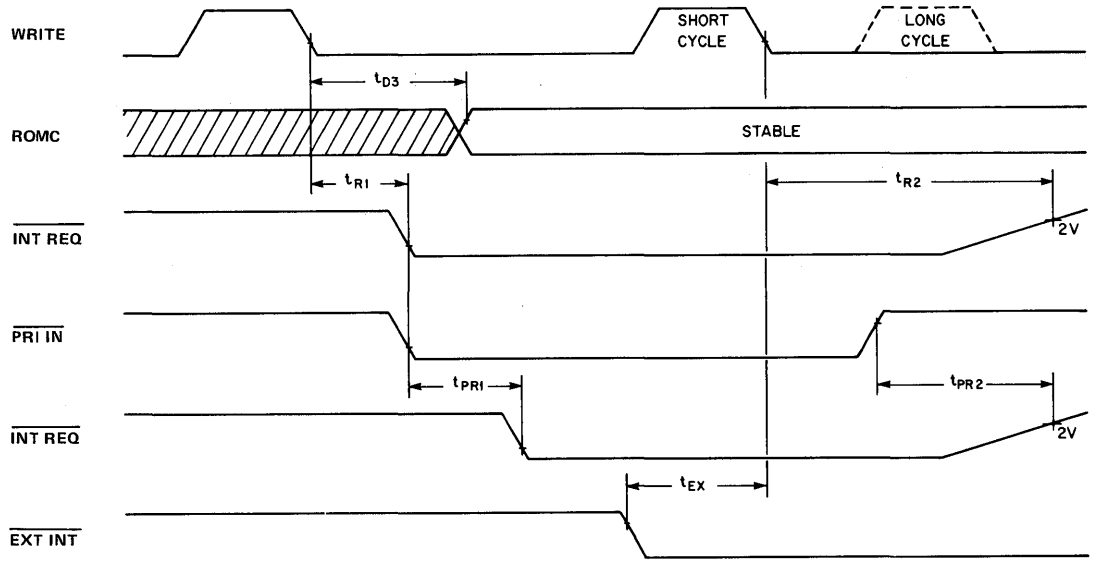
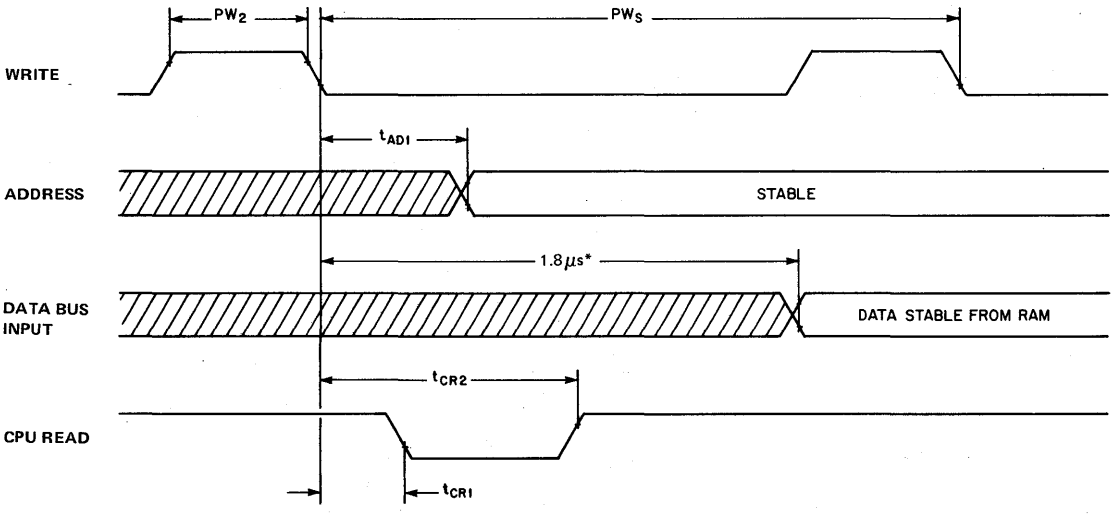


Figure 9

TIMING DIAGRAM

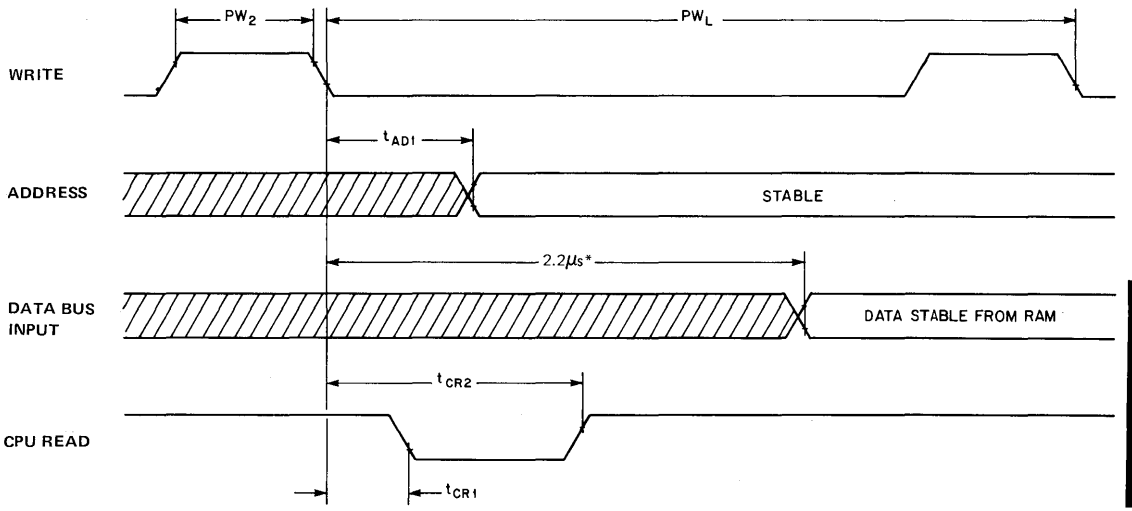


MK 3853 SMI TIMING SIGNALS OUTPUT DURING A SHORT CYCLE MEMORY READ USING P0

Figure 10

*NOTE: This is the time at which the CPU will strobe data in from the memory. (Φ = 2 MHz) Refer to MK3850 CPU data sheet for further information.

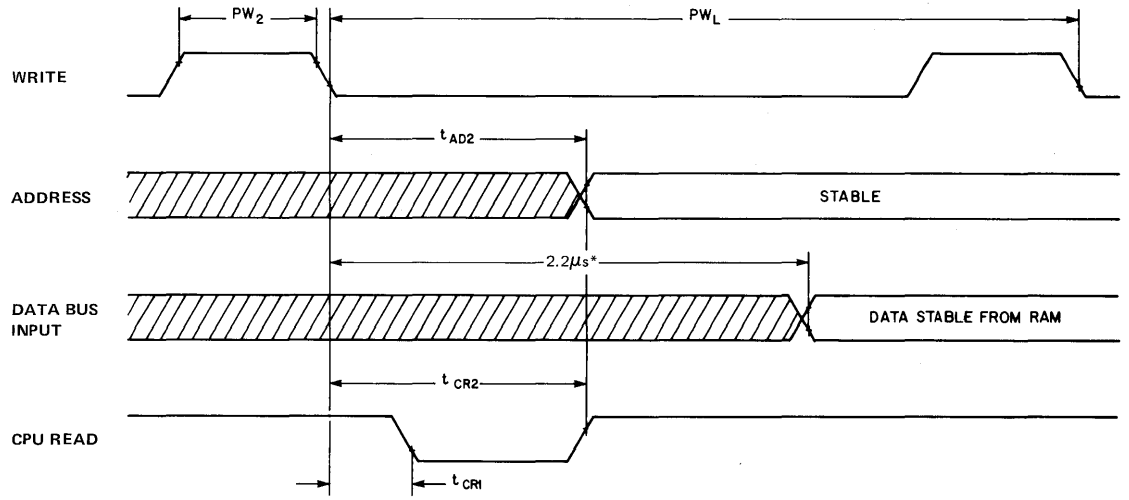
TIMING DIAGRAM



MK 3853 SMI TIMING SIGNALS OUTPUT DURING A LONG CYCLE MEMORY READ, WITH ADDRESS OUT OF PROGRAM COUNTER

Figure 11

TIMING DIAGRAM

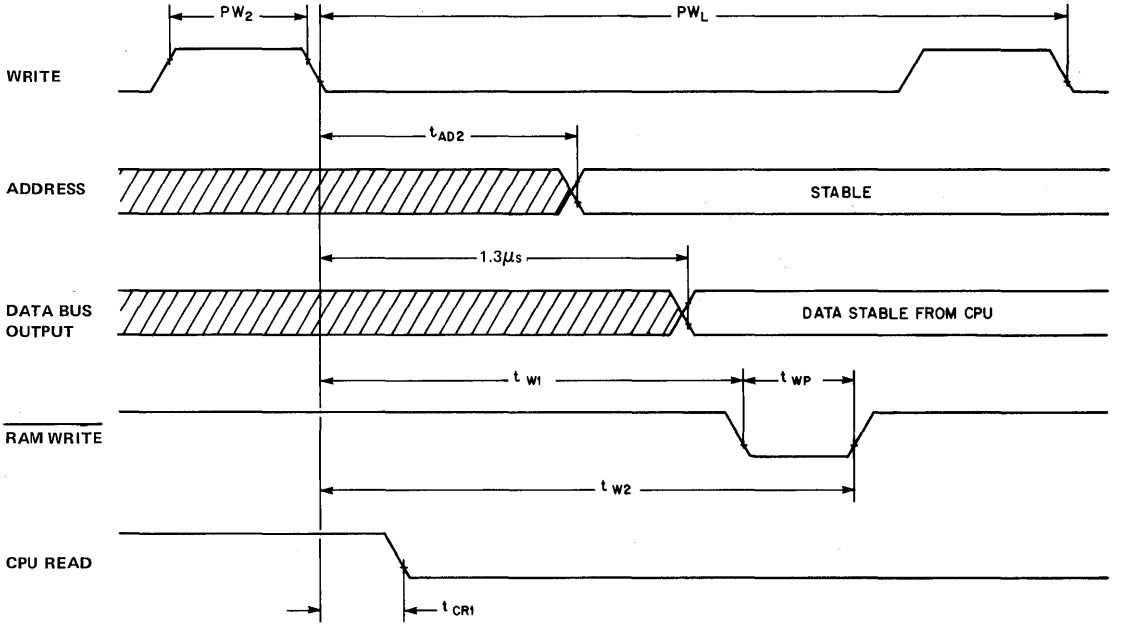


MK 3853 SMI TIMING SIGNALS OUTPUT DURING A LONG CYCLE MEMORY READ, WITH ADDRESS OUT OF DATA COUNTER

Figure 12

*NOTE: This is the time at which the CPU will strobe data in from the memory. (Φ = 2 MHz) Refer to MK3850 CPU data sheet for further information.

TIMING DIAGRAM



MK 3853 SMI TIMING SIGNALS OUTPUT DURING A WRITE TO MEMORY

Figure 13

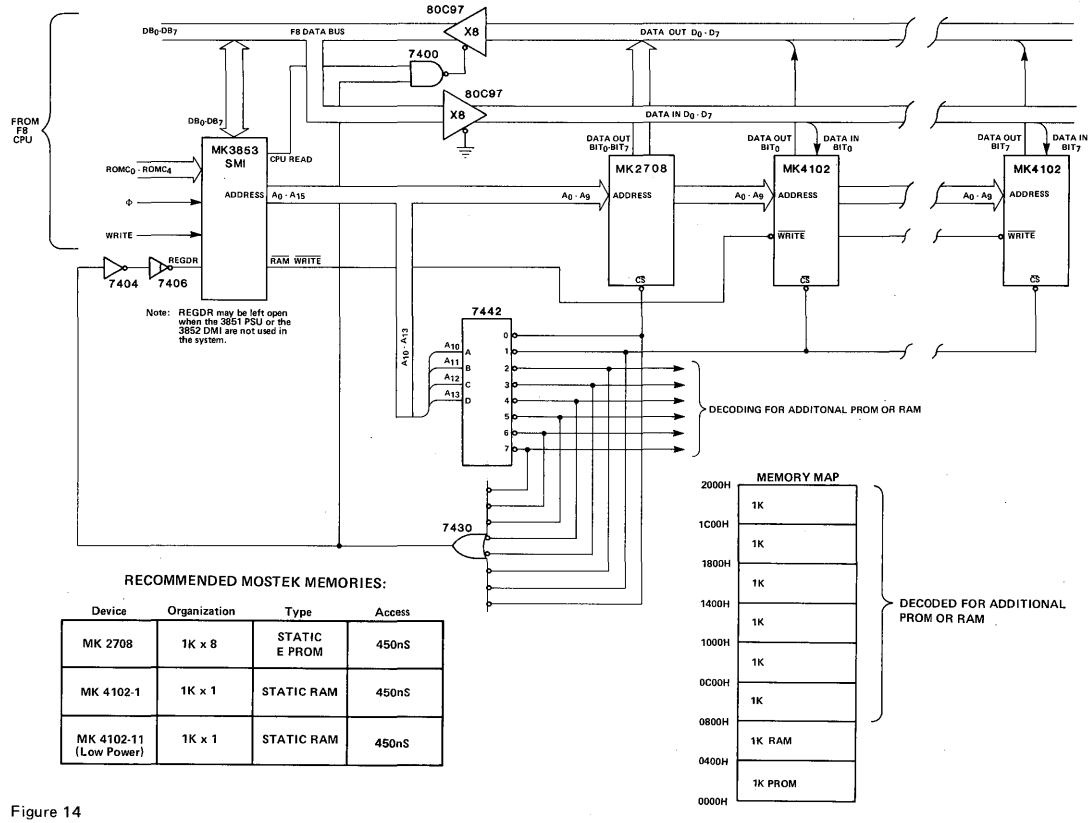
*NOTE: This is the time at which the CPU will output data to memory. (Φ= 2 MHz) Refer to MK3850 CPU data sheet for further information.

MK 3853 APPLICATION

Figure 9 shows a typical application for interfacing the MK 3853 SMI to static memories. This particular example shows a memory system using the MK 2708 1K x 8 EPROM and the MK 4102 1K x 1 Static RAM. Decoding is provided in 1K boundaries for up to 8K of memory. This should be more than adequate for most systems. However, if memory expansion is desired, decoders can be added to provide additional decoding.

The input to the memory array is isolated from the F8 data bus to avoid capacitive loading of the data bus and the output of the memory array is buffered with C/MOS drivers to meet the 3.5 V_{IH} requirement for the MK 3850 CPU.

MK 3853 APPLICATION



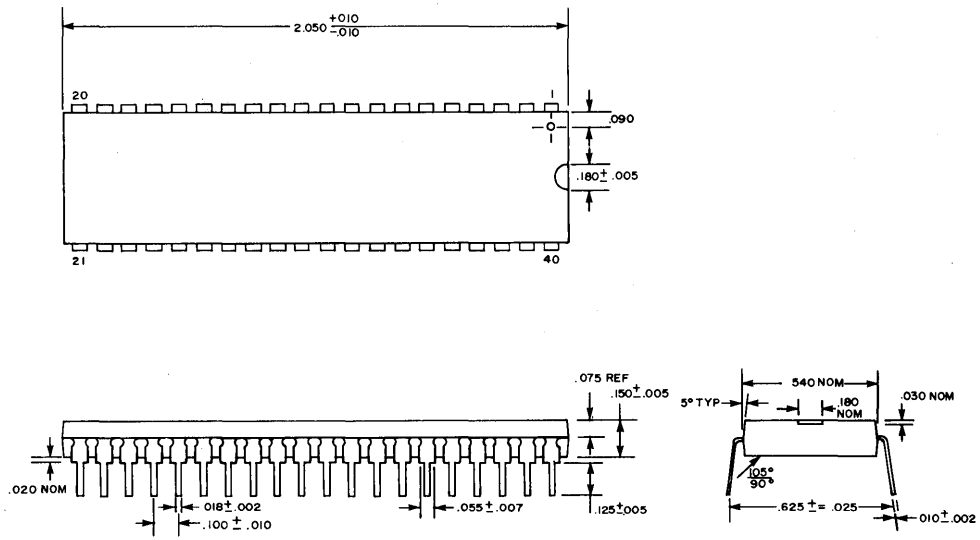
RECOMMENDED MOSTEK MEMORIES:

Device	Organization	Type	Access
MK 2708	1K x 8	STATIC E PROM	450nS
MK 4102-1	1K x 1	STATIC RAM	450nS
MK 4102-11 (Low Power)	1K x 1	STATIC RAM	450nS

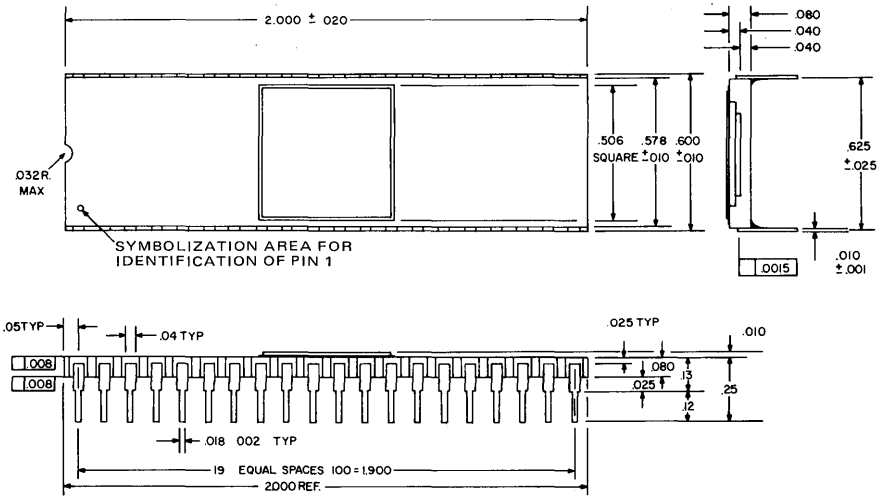
Figure 14

PACKAGE DESCRIPTION

40-lead plastic package



40-lead side-braze ceramic package



ORDERING INFORMATION

MK3853P	0°C To 70°C	Ceramic
MK3853N	0°C To 70°C	Plastic
MK3853P-10	–40°C To +85°C	Ceramic
MK3853N-10	–40°C To +85°C	Plastic
MK3853P-20	–55°C To +125°C	Ceramic
MK3853N-20	–55°C To +125°C	Plastic

F8 Direct Memory Access MK3854

FEATURES

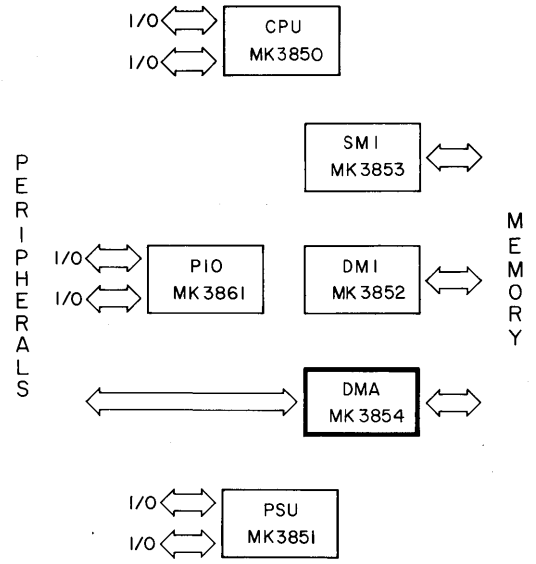
- ☐ 2 μsec cycle time
- ☐ Provides strobe for timing peripherals
- ☐ 16-bit address
- ☐ 12-bit byte count
- ☐ Control registers
- ☐ Port address selection
- ☐ +5V and +12V power supplies
- ☐ Low power dissipation—280mW

GENERAL DESCRIPTION

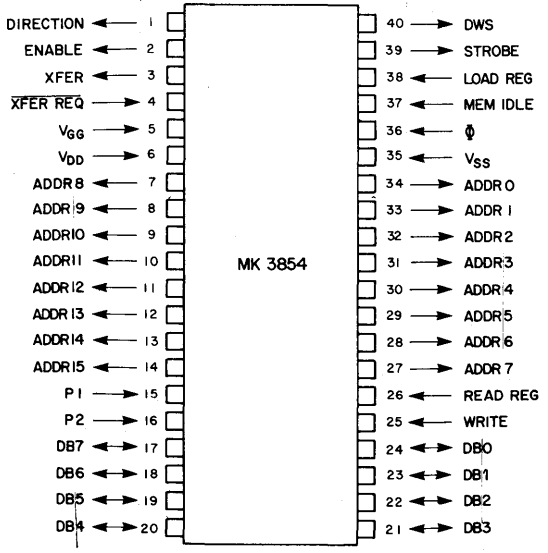
The MK 3854 Direct Memory Access (DMA) chip facilitates high speed data transfer between the main memory of an F8 system and peripherals. This transfer occurs without suspending normal operation of the processor, allowing DMA with no reduction of program execution speed. The MK 3854 DMA is manufactured using N-channel, Isoplanar MOS technology. Power dissipation is low, typically less than 280mW.

PIN NAME	DESCRIPTION	TYPE
DB0-DB7	Data bus lines	Bidirectional three state
ADDR0-ADDR15	Address lines	Output three 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
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

F8 FAMILY



PIN CONNECTIONS



MK 3854 DMA FUNCTIONAL DIAGRAM

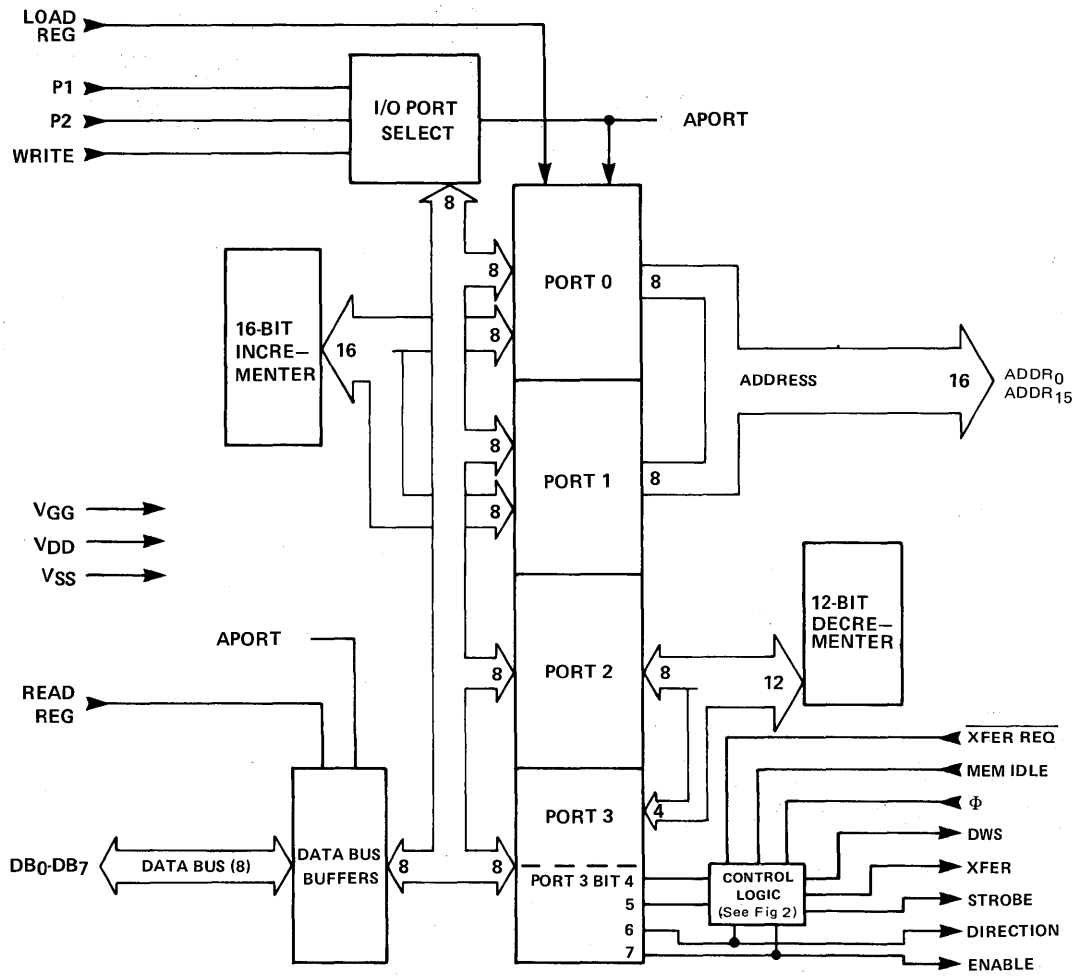


Figure 1

FUNCTIONAL PIN DESCRIPTION

Φand **WRITE** are clocks provided by the MK 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 MK 3854 DMA device in lieu of the five ROMC state signals. Since the MK 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

ADDR0 through ADDR15 are the 16 address lines which address the memory location to be accessed during the current DMA operation. This memory address originates in I/O ports 0 and 1 as illustrated in Figure 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 MK 3854 DMA device from the MK 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 MK 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.

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

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

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 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 MK 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.

DIRECTION is a control output 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.

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

STROBE is a DMA transfer signal output that is used for strobing data and for generating RAM WRITE. STROBE is high only during the second occurrence of Φclock high after MEM IDLE goes true, provided that XFER is also true.

DEVICE ORGANIZATION

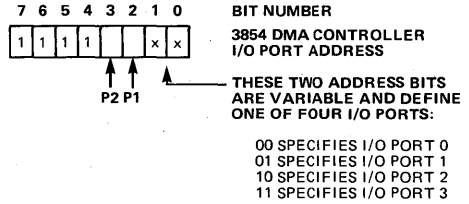
This section describes the operation of the basic functional elements of the MK 3854 DMA. These elements are shown on the DMA block diagram (fig. 1).

I/O PORTS

The MK 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 1.

The four I/O port address used by a 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:



MK 3854 DMA I/O PORT ADDRESSES

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

Table 1

The four I/O ports are not initialized during the power on reset.

DMA CONTROL LOGIC

This logic provides the control signals required to implement DMA data transfers. Figure 2 shows the detailed logic that generates these control signals.

LOAD REG/READ REG

The LOAD REG and READ REG signal inputs to the DMA require special mention.

Most F8 support devices have a control unit which decodes the five ROMC signals output by the MK 3850 CPU. However, the MK 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.

INCREMENT AND DECREMENT LOGIC

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

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, MK 3854 DMA controller's connection to the data bus is used only to transfer data between MK 3854 DMA device I/O ports and the CPU. The data bus is not used to transfer data bytes during a DMA operation.

DMA CONTROL SIGNALS

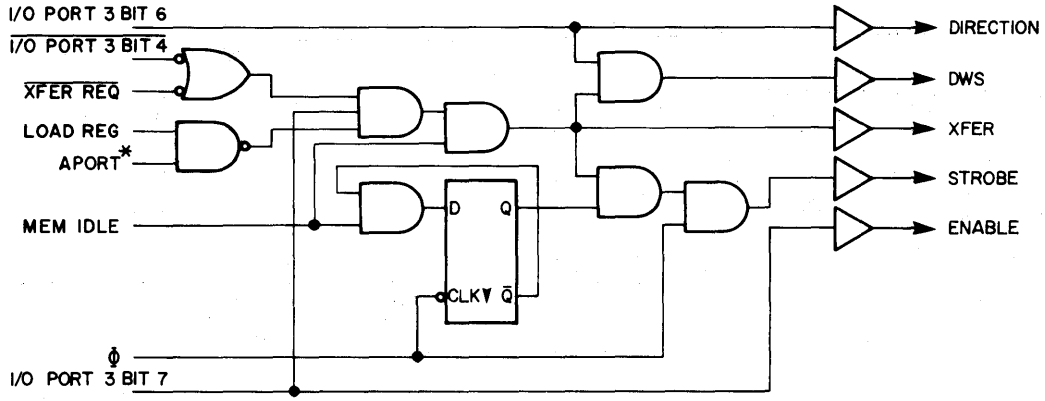


Figure 2

*AN I/O PORT WAS BEING SELECTED

OPERATIONAL DESCRIPTION

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

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

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 starts 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 3.

With reference to Figure 3, 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 ports 0, 1 and 2 are loaded with appropriate data. Then in order to initiate the DMA operations, 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 port 3, and 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 (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. The external device controls DMA transfer rate in this mode. If bit 4 of I/O port 3 is 1, the MK 3854 DMA controller assumes that external logic is ready for a DMA transfer whenever MEM IDLE high identifies a DMA slot. In this mode, the F8 system controls DMA transfer rate.

Each time a DMA data transfer occurs, logic within the MK 3854 DMA controller that is clocked by XFER increments the memory address in I/O ports 0 and 1 and decrements the byte 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 byte counter is decremented to zero. If bit 5 of I/O port 3 is 1, however, the byte 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 byte counter decremented after each DMA access even though the byte counter is ignored.

DMA registers are loaded and read when the MK 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 functions of WRITE in the MK 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.

USE OF PORT 2 AND PORT 3 AS DMA CONTROLS

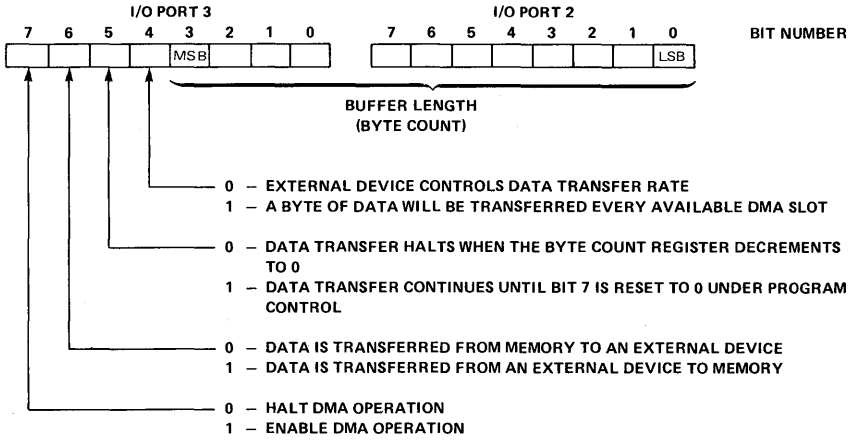


Figure 3

ELECTRICAL SPECIFICATIONS

ABSOLUTE MAXIMUM RATINGS (Above which useful life may be impaired)

V_{GG} ..... +15V to –0.3V
V_{DD} ..... +7 to –0.3V
All other Inputs and 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 = 2MHz, Outputs Unloaded
I_{GG}	V_{GG} Current		15	28	mA	f = 2MHz, Outputs Unloaded

DC 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} = 1.6mA
	I_{IH}	Input High Current		1	μA	V_{IN} = 6V, three-state mode
	I_{IL}	Input Low Current		–1	μA	V_{IN} = V_{SS} three-state mode
ADDRESS LINES	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = –1 mA
(ADDR0-ADDR15)	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 3.2 mA
	I_L	Leakage Current		1	μA	V_{IN} = 6V, three-state mode
ENABLE, DIRECTION	V_{OH}	Output High Voltage	3.9	V_{DD}	Volts	I_{OH} = –100 μA

Table 2

(continued)

(Table 2 continued)

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
DWS (DMA WRITE SLOT), XFER, STROBE	V_{OL}	Output Low Voltage	V_{SS}	0.4	Volts	I_{OL} = 1.6 mA
	I_L	Leakage Current		1	μA	V_{IN} = 6V
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		1	μA	V_{IN} = 6V
WRITE, Φ	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

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

Table 2

AC CHARACTERISTICS

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	NOTES
PΦ	Φ Clock Period	.5		10	μs	Note 1
PW1	Φ Pulse Width	180		PΦ −180	ns	t_r, t_f = 50ns typical
t_{d1}	Φ to WRITE↑ Delay	0		250	ns	Note 1
t_{d2}	Φ to WRITE↓ Delay	0		200	ns	Note 1
PW2	WRITE Pulse Width	P Φ−100		P Φ	ns	t_r, t_f = 50ns typical
t_1	WRITE↓ to CYCLE REQ↑	P Φ+100-t_{d2}		P Φ +300-t_{d2}	ns	Note 4
t_2	WRITE↓ to ENABLE & DIRECTION↑			450	ns	
t_3	MEM IDLE↓ to ENABLE↓			400	ns	
t_4	$\overline{\text{XFER REQ}}$ ↓ to MEM IDLE↑ Set-up	200			ns	
t_5	MEM IDLE↑ to ADDR Valid	50	200	300	ns	C_L = 500 pF
t_6	MEM IDLE↓ to ADDR Hi-Z	30		250	ns	C_L = 500 pF
t_7	MEM IDLE↑ to XFER & DWS↑	50		300	ns	C_L = 50 pF
t_8	MEM IDLE↓ to XFER & DWS↓	50		300	ns	C_L = 50 pF
t_9	MEM IDLE↑ to STROBE↑	600		$\frac{3P\Phi}{2}$ + 100	ns	C_L = 50 pF
t_{10}	STROBE Pulse Width	200		$\frac{P\Phi}{2}$ + 30	ns	C_L = 50 pF
t_{11}	DB Input Set-up Time	300			ns	
t_{12}	WRITE↓ to READ/LOAD REG↑			600	ns	
t_{13}	READ REG↑ to DB Valid	40		300	ns	C_L = 100 pF
t_{14}	WRITE↓ to MEM IDLE↑	2PΦ +50-t_{d2}		2PΦ+300-t_{d2}	ns	Short Cycle
t_{15}	WRITE↓ to MEM IDLE↓	4PΦ+50-t_{d2}		4PΦ+300-t_{d2}	ns	Short Cycle
t_{16}	XFER & DWS↓ to CYCLE REQ↑	0		400	ns	Note 3

Table 3

1. These specifications are those of Φ and WRITE as supplied by the MK 3850 CPU.
2. Input and Output capacitance is 3 to 5pF typical on all pins except V_{DD}, V_{GG}, and V_{SS}.
3. If the next Cycle Req↑ initiates a new read, XFER↓ can be used to clock DMA read data into the peripheral.
4. Cycle Req is output by the MK 3852 DMI to initiate a memory READ/WRITE cycle.

MK 3854 DMA DEVICE TIMING

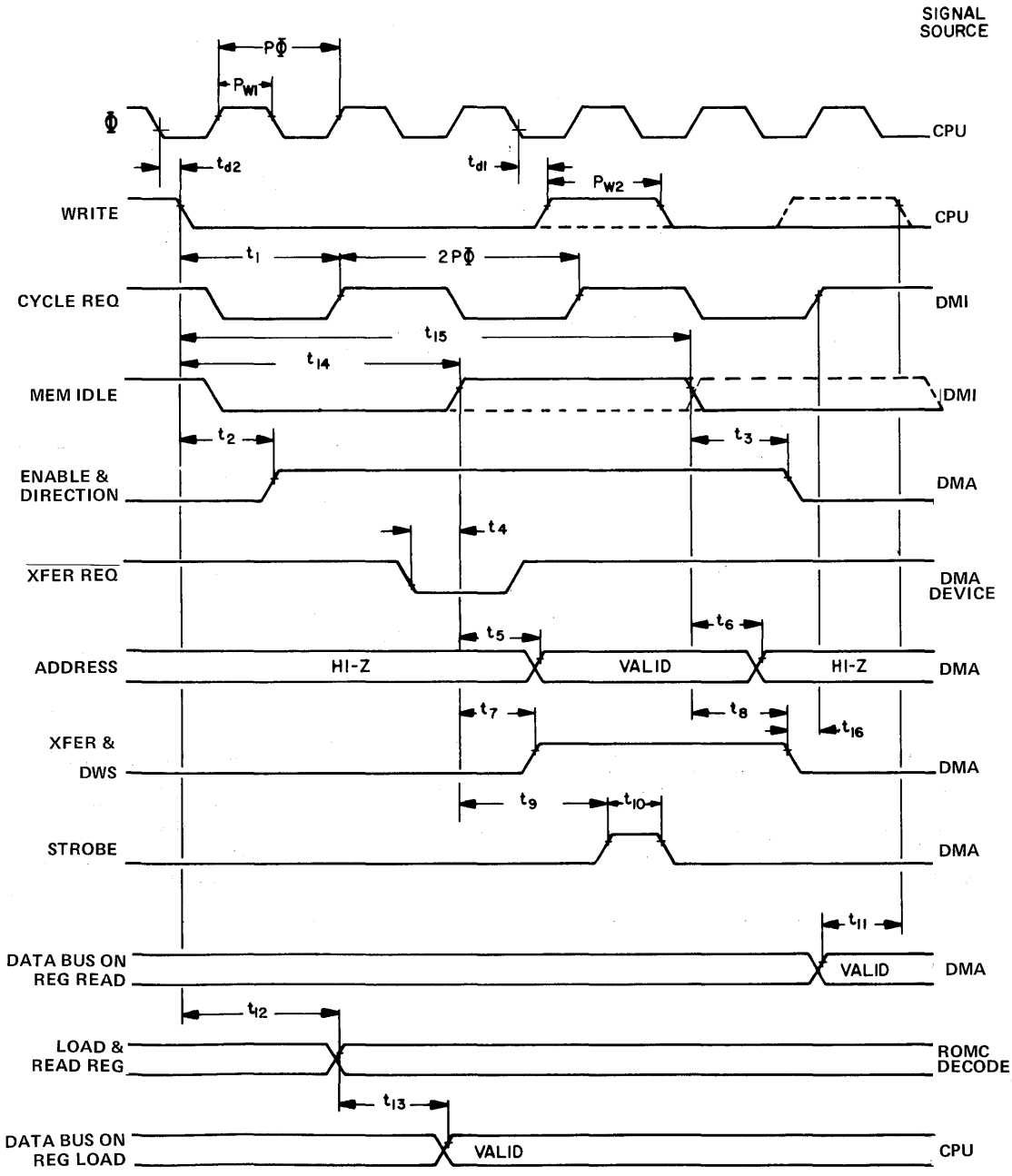
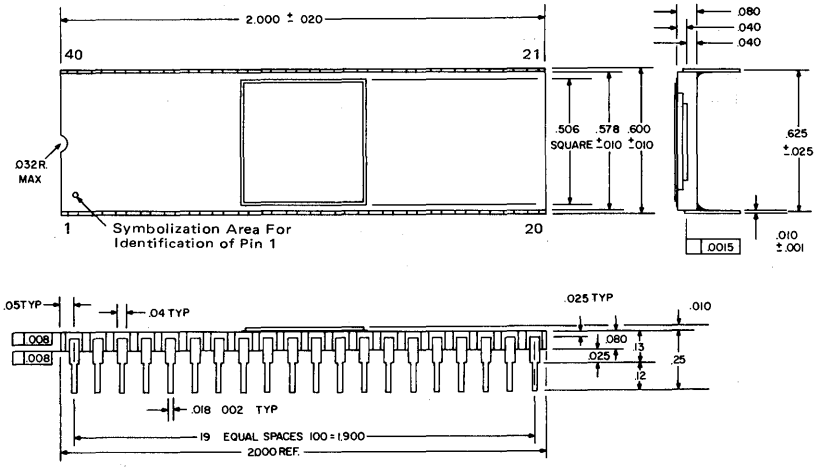
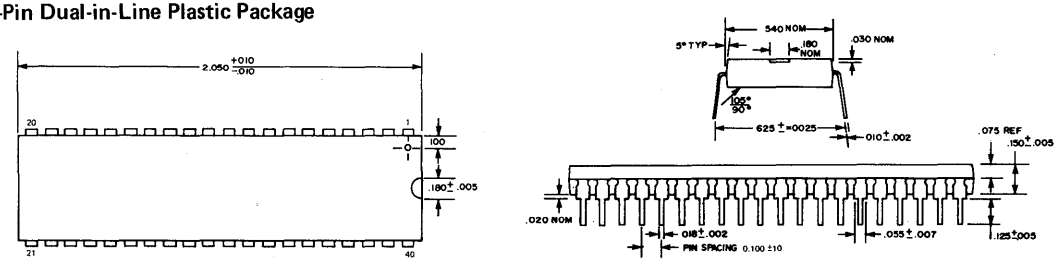


Figure 4

PACKAGE DESCRIPTION — 40-Pin Dual-in-Line Ceramic Package



40-Pin Dual-in-Line Plastic Package



ORDERING INFORMATION

PART NUMBER	PACKAGE	TEMPERATURE RANGE
MK3854(N)	Plastic	0°C to +70°C
MK3854(P)	Ceramic	0°C to +70°C
MK3854(N)-10	Plastic	-40°C to +85°C
MK3854(P)-10	Ceramic	-40°C to +85°C

MOSTEK®

F8 MICROCOMPUTER DEVICES

Peripheral Input/Output MK3861

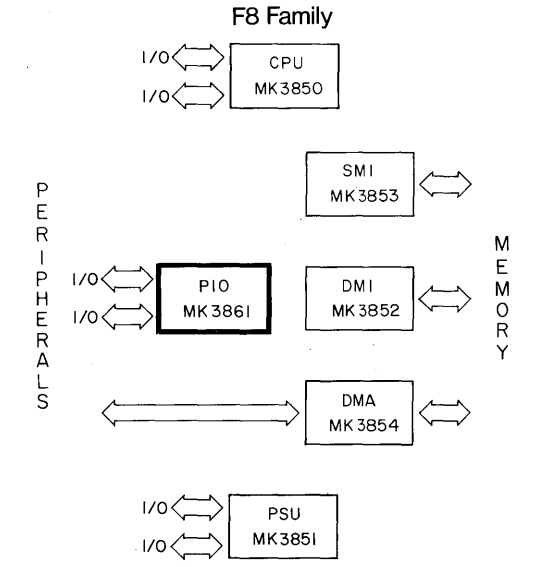
FEATURES

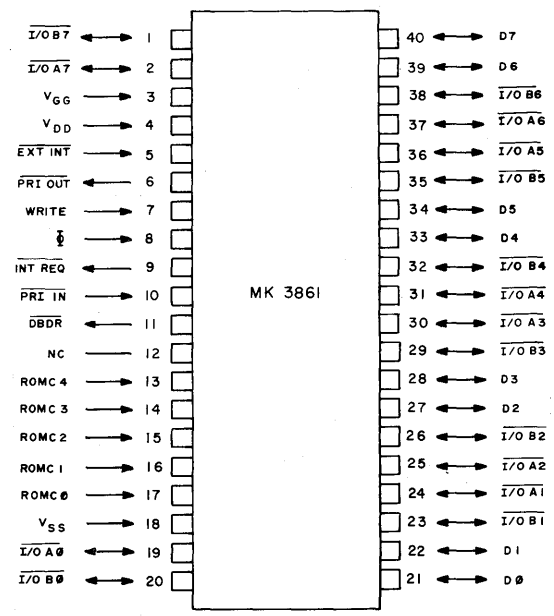
- ☐ Two 8-bit I/O ports
- ☐ Programmable timer
- ☐ External/timer interrupt control circuitry
- ☐ Low power dissipation-typically less than 200mW

GENERAL DESCRIPTION

Each 3861 Peripheral Input/Output Circuit (PIO) provides two 8-bit I/O ports, a programmable timer and a vectored timer or external interrupt for the F8 system. The timer, I/O ports and interrupt circuitry are identical to those of the MK 3851 PSU. The 3861 may be used to provide extra I/O, timer, and interrupt functions compatible with those of the 3851 PSU, or the 3861 may be used as the only I/O peripheral in non PSU systems. This circuit in conjunction with the 3853 and standard PROM is particularly useful in prototyping a PSU system. The 3853 MI circuit along with standard PROM can emulate the memory functions of the PSU while the 3861 provides the I/O, interrupt, and timer features of the PSU. The 3861 is manufactured using the same high performance N-channel Isoplanar technology as the F8 CPU.

PIN NAME	DESCRIPTION	TYPE
D0-D7	Data Bus Lines	Bi-directional, Tri-State
$\overline{\text{I/O A0}}$ - $\overline{\text{I/O A7}}$	I/O Port A	Bi-directional
$\overline{\text{I/O B0}}$ - $\overline{\text{I/O B7}}$	I/O Port B	Bi-directional
ROMC0-ROMC4	System 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 Lines	Input





FUNCTIONAL PIN DEFINITION

D0-D7 (BI-DIRECTIONAL, TRI-STATE)

DATA BUS: The Data Bus provides bi-directional communication between the F8 CPU and the 3861 and all other peripheral circuits for transfer of data. D0 is the least significant bit.

$\overline{\text{I/O A0}}$ – $\overline{\text{I/O A7}}$ and $\overline{\text{I/O B0}}$ – $\overline{\text{I/O B7}}$ (Bidirectional)

I/O PORTS: Two 8-bit I/0 ports are located on the 3861 PIO. These ports are referred to as Port A and Port B herein, but the actual port number is determined by the version of the 3861 that is selected. These ports have output latches to hold output data, and hysteresis circuits are provided to add input noise immunity. Bit 0 of each port is the least significant bit.

ROMC0 – ROMC4 (INPUT)

SYSTEM CONTROL LINES: These lines provide the 3861 with control information from the F8 CPU. The CPU sets up these lines early in each machine cycle, and the PIO executes that command during that cycle.

Φ (INPUT)

Φ (PHI) CLOCK: This is the high frequency F8 system clock. It is generated by the F8 CPU. Each machine cycle contains either 4 Φ periods (short cycle) or 6 Φ periods (long cycle).

WRITE (INPUT)

WRITE CLOCK: This clock defines the machine cycle. The cycle starts with the fall of the WRITE clock. The system control lines become stable shortly after the start of the cycle and the PIO decodes and executes the command communicated by the control lines. All ROMC commands are started and completed within one cycle of WRITE.

$\overline{\text{EXT INT}}$ (INPUT)

EXTERNAL INTERRUPT: When an external circuit pulls this input "low" an external interrupt request will be latched into the PIO if its interrupt control register has been set up to allow external interrupts. The PIO will subsequently communicate this interrupt request to the CPU via its $\overline{\text{INT REQ}}$ line.

$\overline{\text{PRI IN}}$ (INPUT)

PRIORITY IN: This input signals the PIO that a higher priority peripheral has an interrupt request impending on the CPU. If the PIO has already requested an interrupt, it will maintain that request, but it will not be serviced by the CPU until its $\overline{\text{PRI IN}}$ input is in the "low" state. If an interrupt is received, it will be latched into the PIO but it will not be serviced until $\overline{\text{PRI IN}}$ is in the "low" state.

$\overline{\text{PRI OUT}}$ (OUTPUT)

PRIORITY OUT: This output signals lower priority peripherals that the PIO either has an interrupt request impending on the CPU, or that a still higher priority peripheral has requested an interrupt.

$\overline{\text{INT REQ}}$ (OUTPUT)

INTERRUPT REQUEST: This open drain output is wired ANDed with the corresponding output on all other peripherals to form the interrupt request input to the CPU.

$\overline{\text{DBDR}}$ (OUTPUT)

DATA BUS DRIVE: This output goes "low" whenever the PIO is driving the Data Bus as an output. It may be used to control tri-state buffers in a buffered Data Bus system and to signal other peripherals that the PIO has "control" of the Data Bus at that time.

V_{SS} (INPUT)

V_{SS}: This is system ground (0V.) V_{DD} and V_{GG} are referenced to V_{SS}.

V_{DD} (INPUT)

V_{DD}: Power line; +5V ± 5%.

V_{GG} (INPUT)

V_{GG}: Power line; +12V ± 5%.

PIO ARCHITECTURE

Figure 3.0.1 shows the various functional blocks and registers. The 3861 uses the clock signals 'Φ' and 'WRITE', which are generated by the CPU to control timing functions within the circuit. It also uses the contents on five control lines (ROMC's) as various commands to be performed within each cycle. A control ROM within the PIO decodes the five control lines and provides control within the circuit.

ADDRESSABLE PORTS

The 3861 has four addressable ports. They are linked to the accumulator of the CPU by the I/O instructions. Each port is referenced by an 8-bit address. The upper six bits of the address refer to the circuit on which the ports are located while the lower two bits select one of the four ports; hence, the port addresses are referred to as X0,X1,X2 and X3, where X is a six-bit binary number determined by the particular version of the 3861 that is selected. Each port on the device may be writtten into using output instructions. The contents of the I/O ports may be read using input instructions. These instructions initiate the transfer to contents between ports and the accumulator on the CPU. In the PIO circuit, two ports are used as 8-bit I/O ports. The remaining two ports are the 8-bit timer and the local interrupt control port. Table 3.1.1 lists the addressable ports and their respective functions.

PIO FUNCTIONAL DIAGRAM

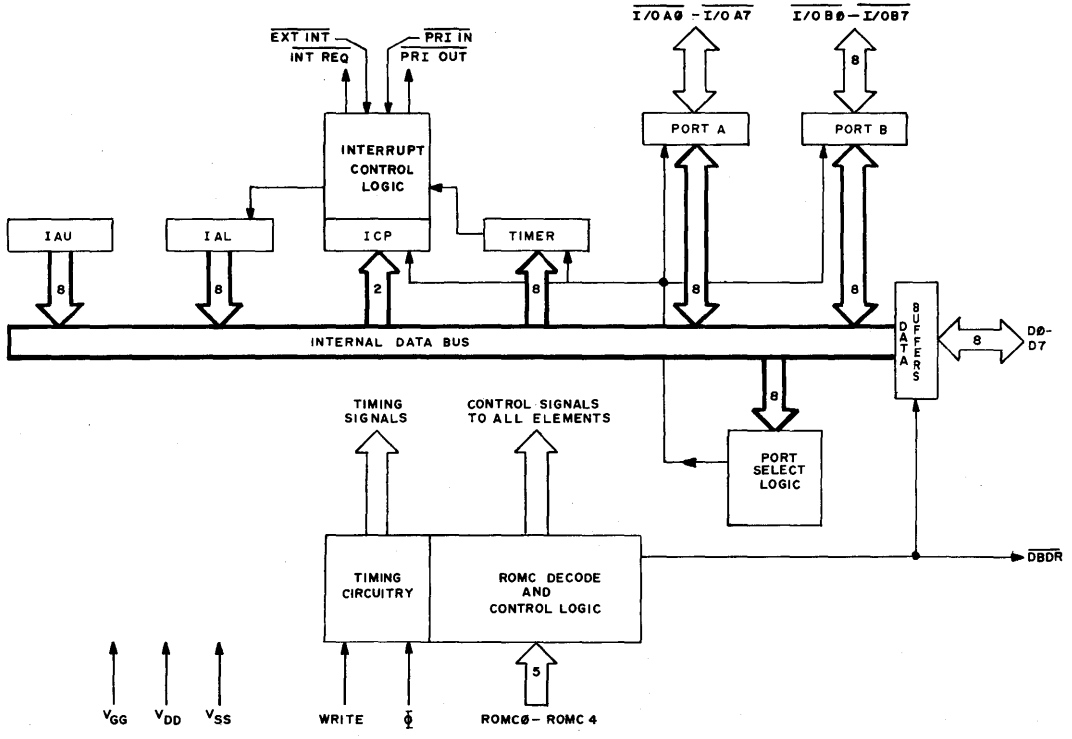


Figure 3.0.1

Table 3.1.1
3861 PIO PORT ASSIGNMENTS

PORT ADDRESS	
X00	PIO I/O Port A (READ–WRITE)
X01	PIO I/O Port B (READ-WRITE)
X10	PIO Local Interrupt Control (WRITE ONLY)
X11	PIO Timer (WRITE ONLY)

INPUT/OUTPUT PORTS

Each 3861 chip has two bidirectional 8-bit I/O ports. Each port's address, using binary notation, is XXXXXX00 or XXXXXX01, where the X binary digits are the chip's unique I/O port select code. Every 3861 used in a system must have a unique I/O port select code.

TIMER

The 3861 has a local timer to generate program initiated delays. To the programmer, the timer is an 8-bit register, addressable via F8 output instructions to the specified timer port address. Delay codes, calculated by the assembler, are loaded into the accumulator and then transferred to the timer (a polynomial shift register). An output instruction to the timer port number performs this function. After it is loaded, the timer counts down. A table of delay codes matched to delay times appears in the Appendix A.

The timer runs continuously. It signals the interrupt control circuitry after each timer cycle (3.953 ms in a 2MHz system). However, when an output instruction is executed with the timer port number as the operand, the timer is jammed with a specific count and the local interrupt control logic clears any stored timer interrupt. The timer then counts down from that count in a polynomial sequence (Appendix A) and generates an internal interrupt request when a count of H'FE' is reached. From that point, the timer continues to cycle every 3.953 milliseconds (for a 2MHz system) unless it is re-loaded as described above. If the interrupt is not set for timer interrupts, a timer initiated interrupt will be stored by the local interrupt control circuitry. When the local interrupt control logic is finally set to allow timer interrupts, the PIO will request interrupt service.

INTERRUPT INTERCONNECTION

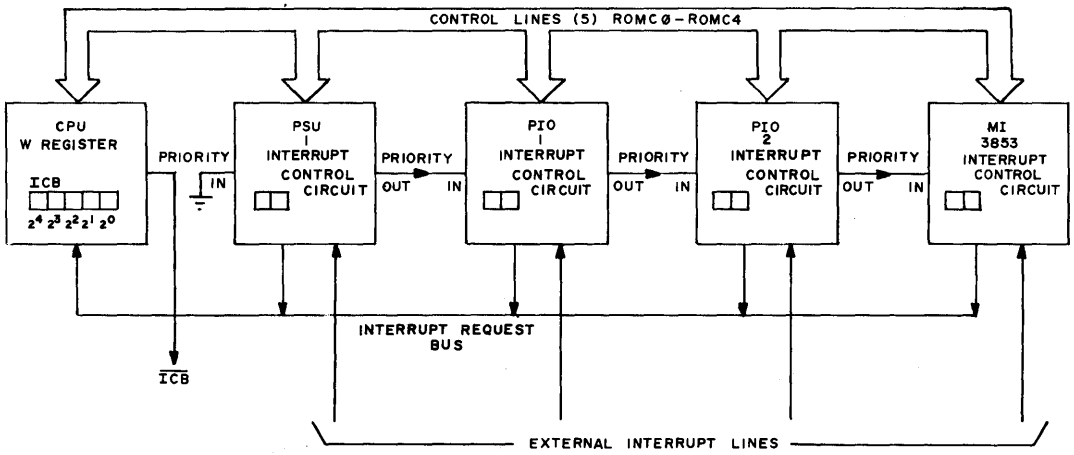


Figure 3.4.1

Time delays between 0 and 254 counts may be chosen. The timer is decremented once every 31 Φ clock cycles. Therefore, the counter may count as high as 7905 Φ clock cycles. (For a system at 2MHz, a clock cycle occurs every 500ns). Longer durations are achieved by counting multiple time interrupts. If the timer is loaded with all one's, it will stop counting.

INTERRUPT CONTROL PORT AND LOGIC

Figure 3.4.1 is a block diagram of the interrupt interconnection for a typical F8 system. The 3861 PIO, has either of two types of interrupts, internal or external. The internal interrupt may be generated by the programmable timer while the external interrupt is generated by external logic in the system. A local interrupt control cicuit containing two latches is included on each device. These latches are the Select Bit and the Interrupt Enable Bit.

Table 3.4.1

LOCAL INTERRUPT CONTROL BITS

2^1	2^0
Select Bit	Interrupt Enable Bit

These two bits have four possible states:

Select Bit	Interrupt Enable Bit	Function
0	0	No Interrupt
0	1	External Interrupt Enabled
1	0	No Interrupt
1	1	Timer (Internal) Interrupt Enabled

These control latches are loaded under program control using an output instruction. This loading clears the interrupt control logic, except for any pending timer interrupt. The operand for the OUT or OUTS instruction must be the predefined port number of the Interrupt Control Port (ICP). The two control bits allow each interrupt circuit to have independently controlled enable/disable capabilities. If enabled, the select bit may choose either internal (timer generated) interrupts or external interrupts.

Each PIO has a $\overline{\text{PRIORITY IN}}$ and a $\overline{\text{PRIORITY OUT}}$ line so that they may be daisy chained together in any order, to form a priority level of interrupts. When a PIO receives an interrupt (either timer or external) it pulls its $\overline{\text{PRI OUT}}$ output high, signaling all lower priority peripherals that it has a higher priority interrupt request impending on the CPU. Also when the PIO's $\overline{\text{PRI IN}}$ input is pulled high by a higher priority peripheral, signaling the PIO that there is a still higher priority interrupt request, it passes that signal along by pulling its $\overline{\text{PRI OUT}}$ high. When the CPU processes an interrupt request it commands the interrupting peripheral to place its interrupt vector address on the Data Bus. Only that peripheral whose $\overline{\text{PRI IN}}$ is low and who has an interrupt request impending will respond. Should there be another lower priority peripheral with an impending request, it will not respond at that time because its $\overline{\text{PRI IN}}$ input will be high.

To generate a timer interrupt, the timer must be set under program control. The PIO generates a timer interrupt request when the timer times out AND the interrupt control has been set (Select Bit = 1, Enable Bit = 1). The CPU will not process the request until 1, it is enabled to handle interrupts by setting the ICB bit in the status register, and 2, it has completed processing all higher priority interrupt requests. The timer may time out before ICB is set or the local interrupt control is enabled for internal interrupts; however, an interrupt will still be initiated after the required conditions have been met. Any pending timer interrupt is cleared whenver output instructions load the timer. The ICB is always cleared after the CPU has acknowledged an interrupt request.

The generation of an external interrupt request is also controlled by the local interrupt control circuit. If the Select Bit is set to zero and the Enable Bit is set to one, the control logic of the chip is responsive to the external interrupts. To guarantee an interrupt, the external interrupt line must drop from 1 (near V_{DD}) to 0 (near V_{SS}), and stay at zero for a minimum of two WRITE clock periods (4 μs for a 500 ns system clock). The ICB may or may not be set when this occurs. If it is not set, the request will be stored by the local interrupt control logic until the ICB is reenabled; however, the stored external interrupt request will be lost whenever the control bits are reloaded. However, loading the control bits does not clear a stored timer interrupt. The stored external interrupt request will be cleared after that interrupt is serviced.

Within each local interrupt control circuit there is a 16-bit interrupt address vector. This vector is the address to which the program counter will be set after an interrupt is acknowledged; hence, it is the address of the first executable instruction of the interrupt routine. The 3861 has an interrupt address which is particular to the version of the 3861 selected by the user. Fifteen bits are fixed. These are bits 0 through 6 and 8 through 15. Bit seven (2^7) is dependent upon the type of interrupt. This bit will be a 0 for internal timer generated interrupts and a 1 for external interrupts. When the interrupt logic sends an interrupt request signal and the CPU is enabled to service it, the normal state sequence of the CPU is interrupted at the end of an instruction. The CPU signals the interrupt circuits via the five control lines. The requesting local interrupt circuit sends a 16-bit interrupt address vector (from the interrupt address generator) onto the Data Bus in two consecutive bytes. The address is made available to the program counter via the address demultiplexer circuits. Simultaneously, the address is also made available to all other devices connected to the data bus. It is the address of the next instruction to be executed. The program counter (PC0) of each memory device is set with this new address while the stack register (PC1) is loaded with the previous contents of the program counter. The information in PC1 is lost. Thus, the next instruction to be executed is determined by the value of the interrupt address vector.

The Interrupt Control Bit (ICB) of the CPU (loaded in the W register) allows interrupts to be recognized. Clearing the ICB prevents acknowledgement of interrupts. The ICB is cleared during power on, external reset, and after an interrupt is acknowledged. The interrupt status of the PSU, PIO or MI devices are not affected by the execution of the DISABLE INTERRUPT (DI) instruction. At the conclusion of most instructions, the fetch logic checks the state of the Interrupt Request Line. If there is an interrupt, the next instruction fetch cycle is suspended and the system is forced into an interrupt sequence.

The CPU allows interrupts after all F8 instructions except the following:

(PK)	PUSH K
(PI)	PUSH IMMEDIATE
(POP)	POP
(JMP)	JUMP
(OUTS)	OUTPUT SHORT (Excluding OUTS 00 and 01)
(OUT)	OUTPUT
(EI)	SET ICB
(LR W, J)	LOAD THE STATUS REGISTER FROM SCRATCHPAD
POWER ON	

As a result, it is possible to perform one more instruction after the above CPU instructions without being interrupted.

DATA FLOW

Table 3.5.1 shows the function performed by the PIO for each ROMC command. Each function is entirely performed within one machine cycle (one cycle of the WRITE clock)

TABLE 3.5.1

The following ROMC states are decoded by the 3861 as indicated. All other ROMC states are decoded as "NO–OPERATION' (NO–OP).

Binary (ROMC 4 3 2 1 0)	Hex	3861 FUNCTION
0 1 1 1 1	0F	If this circuit is interrupting and no higher priority circuit is interrupting, move the lower half of the interrupt vector on to the Data Bus and signal Bus use with DBDR.
1 0 0 0 0	10	Place interrupt circuitry on an inhibit state that prevents altering the interrupt chain.

1 0 0 1 1 13 If this circuit is interrupting and no higher priority circuit is interrupting move the upper half of the interrupt vector on to the Data Bus and signal Bus use with DBDR. In any case, remove priority interrupt circuitry from inhibit state.

1 1 0 1 0 1A If contents of Data Bus in the previous were an address of an I/O port, the timer, or the Interrupt Control Port, move current contents of the Data Bus into that port. (Output Command).

1 1 0 1 1 1B If contents of Data Bus in the previous cycle was an I/0 port address, move the contents of that port on to the Data Bus and signal Bus use with DBDR (Input Command).

3861 PIO VERSIONS

Each version of the 3861 is denoted by a MK 90----- number. This ninety thousand series number should be used when ordering or specifying a 3861 to insure that the proper version is understood. Thus, the complete part designation of a particular version of the 3861 is:

3861
MK90--------

The presently available versions of the 3861 are listed in table 4.0.1.

AVAILABLE VERSIONS OF THE 3861
TABLE 4.0.1

VERSION	PORT SELECT CODE	PORT NUMBERS (DERIVED FROM THE PORT SELECT CODE; HEX)	PORT OUTPUT TYPE	INTERRUPT ADDRESS	
				TIMER	EXTERNAL
MK 90001	000001	04 thru 07	Standard T^2Compatible	0600	0680
MK 90002	000010	08 thru 0B	Standard T^2 Compatible	0340	03C0
MK 90003	001000	20 thru 23	Standard T^2 Compatible	0320	03A0
MK 90004	001001	24 thru 27	Standard T^2 Compatible	0360	03E0
MK 90005	000001	04 thru 07	Standard T^2 Compatible	0020	00A0

ELECTRICAL SPECIFICATIONS

ABSOLUTE MAXIMUM RATINGS*

V_{GG}, EXT INT ..... −.3V to +15V
V_{DD} ..... −.3V to +7V
I/O PORT OPEN DRAIN OPTION ..... −.3V to +15V
ALL OTHER INPUTS AND OUTPUTS ..... −.3V to +7V
STORAGE TEMPERATURE .....−55°C to +150°C
OPERATING TEMPERATURE .....0°C to 70°C

*All voltages are with respect to V_{SS}. Stresses above those listed may cause permanent damage to the device. Exposure to maximum rated stress for extended periods may impair the useful life of the device.

DC CHARACTERISTICS

V_{SS} = 0V, V_{DD} = 5V ± 5%, V_{GG} = 12V ± 5%
T_A = 0 to 70°C, unless otherwise noted.
Positive current is defined as conventional current flowing into the pin referenced.

SUPPLY CURRENTS

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
I_{DD}	V_{DD} Current		25	60	mA	f= 2MHz, Outputs unloaded
I_{GG}	V_{GG} Current		8	15	mA	f= 2MHz, Outputs unloaded

DATA BUS (DB0-DB7)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	3.5		V_{DD}	Volts	
V_{IL}	Input Low Voltage	V_{SS}		.8	Volts	
V_{OH}	Output High Voltage	3.9		V_{DD}	Volts	$I_{OH} = -100\mu A$
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	I_{OL} = 1.6mA [1]
I_{IH}	Input High Current	0		1	μA	V_{IN} = 6V, 3-State mode
I_{OL}	Input Low Current	0		−1	μA	$V_{IN} = V_{SS}$, 3-State mode
C_I	Input Capacitance			10	pF	3-State mode

CLOCK LINES (Φ WRITE)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	4.0		V_{DD}	Volts	
V_{IL}	Input Low Voltage	V_{SS}		.8	Volts	
I_L	Leakage Current			1	μA	V_{IN}= 6V
C_I	Input Capacitance			10	pF	

PRIORITY IN AND CONTROL ($\overline{\text{PRI IN}}$, ROMCO – ROMC4)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	3.5		V_{DD}	Volts	
V_{IL}	Input Low Voltage	V_{SS}		.8	Volts	
I_L	Leakage Current			1	μA	V_{IN} = 6V
C_I	Input Capacitance			10	pF	

PRIORITY OUT ($\overline{\text{PRI OUT}}$)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage	3.9		V_{DD}	Volts	$I_{OH} = -100\mu A$
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	I_{OL} = 100μA

INTERRUPT REQUEST ($\overline{\text{INT REQ}}$)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage				Volts	Open Drain Output [1]
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	I_{OL} = 1mA
I_L	Leakage Current			1	μA	V_{IN} = 6V, Output device off
C_I	Input Capacitance			10	pF	Output device off

DATA BUS DRIVE ($\overline{\text{DBDR}}$)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage					Open Drain Output
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	I_{OL} = 1mA
I_L	Leakage Current			1	μA	V_{IN} = 6V, Output device off
C_I	Input Capacitance			10	pF	Output device off

EXTERNAL INTERRUPT ($\overline{EXT\ INT}$)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	3.5			Volts	Internal pullup exsists
V_{IL}	Input Low Voltage			1.2	Volts	
V_{IC}	Input Clamp Voltage			15	Volts	$I_{IH} = 185\mu A$
I_{IL}	Input Low Current	−250		−750	μA	$V_{IN} = V_{SS}$
C_I	Input Capacitance			10	pF	

I/O PORT OPTION A (STANDARD PULLUP)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage	3.9		V_{DD}	Volts	$I_{OH} = -30\mu A$
V_{OH}	Output High Voltage	2.9		V_{DD}	Volts	$I_{OH} = -100\mu A$
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	I_{OL} = 2mA
V_{IH}	Input High Voltage	2.9		V_{DD}	Volts	Internal Pullup to V_{DD} [2]
V_{IL}	Input Low Voltage	V_{SS}		.8	Volts	
I_{IL}	Input Low Current			−1.2	μA	V_{IN} = .4V[3]
C_I	Input Capacitance			10	pF	

I/O PORT OPTION (OPEN DRAIN)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage					External Pullup
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	I_{OL} = 2mA
V_{IH}	Input High Voltage	2.9		V_{DD}	Volts	[2]
V_{IL}	Input Low Voltage	V_{SS}		.8	Volts	
I_L	Leakage Current			1	A	V_{IN} = 6V, Output device off
C_I	Input Capacitance			10	pF	

I/O PORT OPTION C (DRIVER PULLUP)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage	3.75		V_{DD}	Volts	I_{OH} = -1 mA
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	I_{OL} = 2 mA

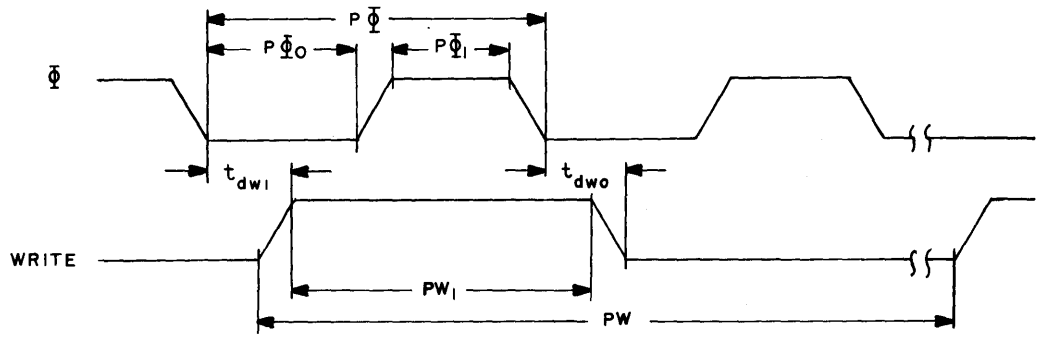
NOTES:

1. Pull up resistor to V_{DD} on CPU.
2. Hysteresis input circuit provides additional .3V noise immunity while internal/external pullup provides TTL compatibility.
3. Measured while I/O port is outputting a high level.

TIMING

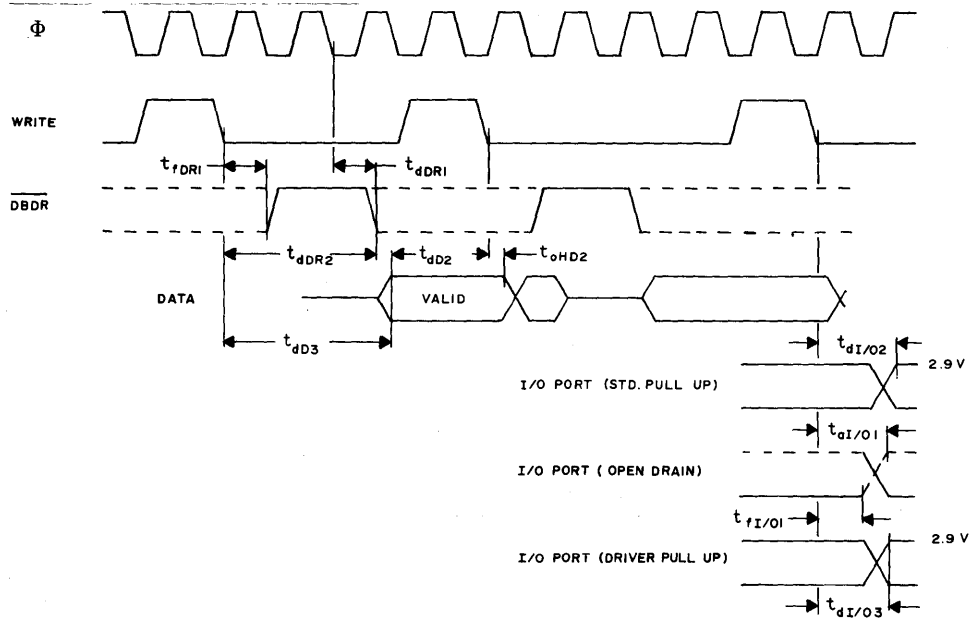
All timing specified at V_{SS} = 0V, V_{DD} = 5V ± 5%, V_{GG} = 12V ± 5%, T_A = 70°C to 0°C.

CLOCK TIMING



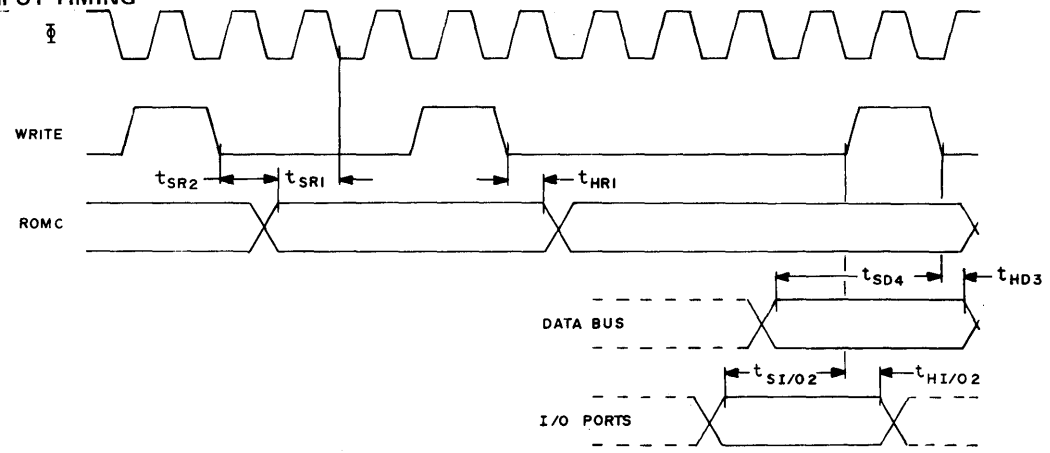
SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	CONDITIONS
PΦ	Clock Period	.5		10	μs	
$P\Phi_0$	Low time	180			ns	
$P\Phi_1$	High time	180			ns	
PW	WRITE Clock Period		4PΦ			Short cycle
PW_0	WRITE Clock Period		6PΦ			Long cycle
PW_1	WRITE Pulse Width	PΦ−100		PΦ		
t_{dw1}	Φ − to WRITE + delay			250	ns	
t_{dw0}	Φ − to WRITE − delay			225	ns	

OUTPUT TIMING



SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	CONDITIONS
t_{fDR1}	WRITE to DBDR floating			400	ns	
t_{dDR1}	Φ to DBDR 1-0		200	625	ns	C_L = 100pF, R_L = 12.5K
t_{dDR2}	WRITE to DBDR 1-0			2P Φ+ 625− tdw0	ns	C_L = 100pF, R_L = 12.5K
					ns	C_L = 100pF
t_{dD3}	WRITE to DATA VALID	2P Φ− tdW0	2P Φ− 400	2P Φ+ 700− t_{dW0}	ns	C_L = 100pF
t_{0HD2}	Guaranteed Data Hold Time After Fall of WRITE	30			ns	
$t_{dI/02}$	WRITE to I/0 Port Valid			1.5	μs	STD Pull up, C_L = 50pF
$t_{dI/03}$	WRITE to I/0 Port Valid			400	ns	Driver Pullup, C_L = 50pF
$t_{d\ I/01}$	WRITE to I/0 Port-Actively Pulled Down			400	ns	Open Drain R_L = 12.5K, C_L = 50pF
$t_{fI/01}$	WRITE to I/0 Port-Floating			375	ns	Open Drain

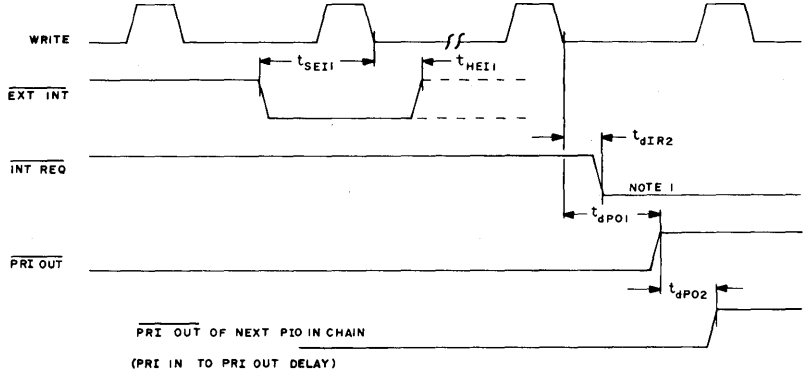
INPUT TIMING



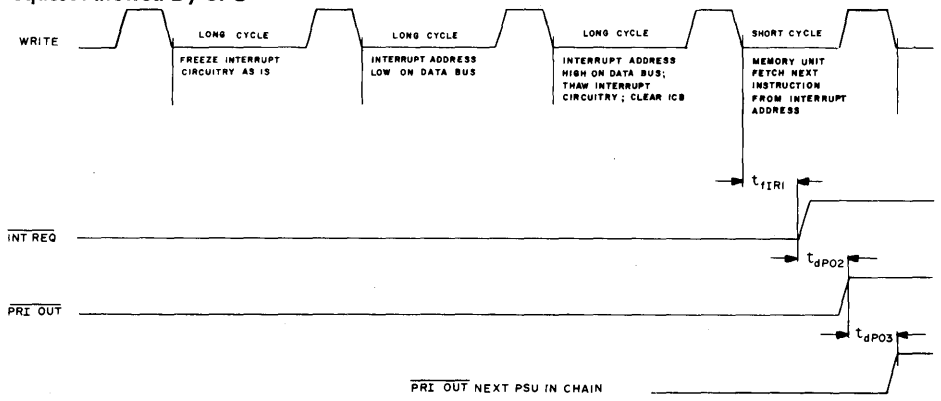
SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	CONDITIONS
t_{SR1}	ROMC Setup Time	225			ns	
t_{SR2}	ROMC Valid Measured From Fall of WRITE			550	ns	
t_{HR1}	ROMC Required Hold After Fall Of WRITE	20			ns	
t_{SD4}	Data Bus Set-up Time				ns	
t_{HD3}	Data Input Hold Time	20			ns	
$t_{SI/02}$	I/0 Input Set-up Time	1.3			ns	
$t_{HI/02}$	I/0 Input Hold Time	20			ns	

5.2.4 INTERRUPT TIMING

A. Request Made



B. Request Allowed By CPU



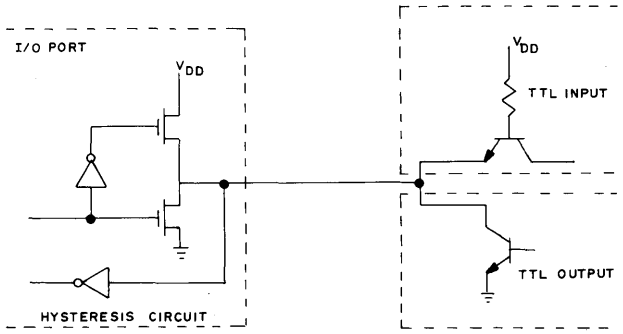
NOTES.

1. Assuming PRI IN is already low. If not, INT REQ 1–0 transition will be delayed 240ns max from the time PRI IN is enabled, and PRI OUT 0–1 transition will be delayed t_{dPO2} from the time PRI IN is enabled.

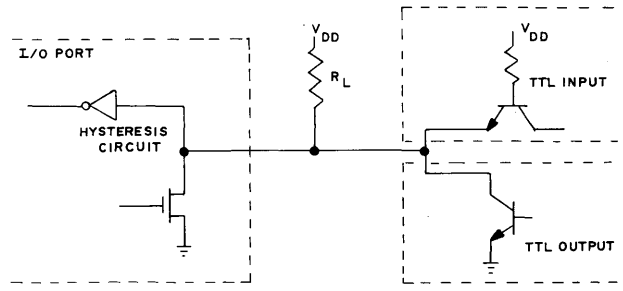
SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	CONDITIONS
t_{SEI1}	EXT INT Setup Time			1.3	ns	
t_{HEI}	EXT INT Hold Time	30			ns	
t_{dIR2}	WRITE to INT REQ Delay			430	ns	C_L = 100pF
t_{dPO1}	WRITE to PRI OUT Delay			640	ns	C_L = 50pF
t_{dPO2}	PRI IN to PRI OUT 0–1 Delay			300	ns	C_L = 50pF
t_{fIR1}	WRITE to INT REQ Float by PSU			640	ns	Open Drain Output
t_{dPO3}	PRI IN to PRI OUT 1–0 Delay			365	ns	C_L = 50pF

INTERFACING

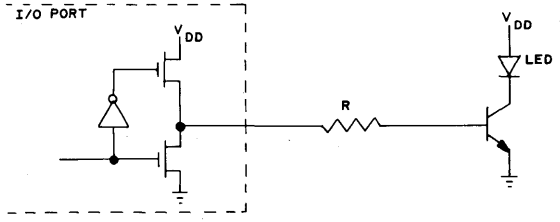
STANDARD CONFIGURATION



OPEN DRAIN CONFIGURATION



DRIVER PULL–UP CONFIGURATION



APPENDIX A-TIMER COUNTS

CONTENTS OF COUNTER	COUNTS TO INTERRUPT	CONTENTS OF COUNTER	COUNTS TO INTERRUPT	CONTENTS OF COUNTER	COUNTS TO INTERRUPT
FE	254	A4	198	70	142
FD	253	49	197	E1	141
FB	252	92	196	C3	140
F7	251	25	195	86	139
EE	250	4A	194	0C	138
DC	249	94	193	18	137
B8	248	29	192	31	136
71	247	53	191	63	135
E3	246	A6	190	C6	134
C7	245	4D	189	8C	133
8E	244	9A	188	19	132
1D	243	34	187	33	131
3B	242	69	186	67	130
76	241	D3	185	CE	129
ED	240	A7	184	9D	128
DA	239	4F	183	3A	127
B4	238	9E	182	74	126
68	237	3C	181	E9	125
D1	236	78	180	D2	124
A3	235	F0	179	A5	123
47	234	E0	178	4B	122
8F	233	C1	177	96	121
1F	232	82	176	2D	120
3F	231	04	175	5B	119
7E	230	09	174	B7	118
FC	229	12	173	6E	117
F9	228	24	172	DD	116
F3	227	48	171	BA	115
E6	226	90	170	75	114
CD	225	21	169	EB	113
9B	224	42	168	D6	112
36	223	85	167	AD	111
6D	222	0A	166	5A	110
DB	221	14	165	B5	109
B6	220	28	164	6A	108
6C	219	51	163	D5	107
D9	218	A2	162	AB	106
B2	217	45	161	56	105
64	216	8B	160	AC	104
C8	215	17	159	58	103
91	214	2E	158	B1	102
23	213	5D	157	62	101
46	212	BB	156	C4	100
8D	211	77	155	88	99
1B	210	EF	154	11	98
37	209	DE	153	22	97
6F	208	BC	152	44	96
DF	207	79	151	89	95
BE	206	F2	150	13	94
7D	205	E4	149	26	93
FA	204	C9	148	4C	92
F5	203	93	147	98	91
EA	202	27	146	30	90
D4	201	4E	145	61	89
A9	200	9C	144	C2	88
52	199	38	143	84	87

CONTENTS OF COUNTER	COUNTS TO INTERRUPT
08	86
10	85
20	84
40	83
81	82
02	81
05	80
0B	79
16	78
2C	77
59	76
B3	75
66	74
CC	73
99	72
32	71
65	70
CA	69
95	68
2B	67
57	66
AE	65
5C	64
B9	63
73	62
E7	61
CF	60
9F	59
3E	58
7C	57
F8	56
F1	55
E2	54
C5	53
8A	52
15	51
2A	50
55	49
AA	48
54	47
A8	46
50	45
A0	44
41	43
83	42
06	41
0D	40
1A	39
35	38
6B	37
D7	36
AF	35
5E	34
BD	33
7B	32
F6	31

CONTENTS OF COUNTER	COUNTS TO INTERRUPT
EC	30
D8	29
B0	28
60	27
C0	26
80	25
00	24
01	23
03	22
07	21
0F	20
1E	19
3D	18
7A	17
F4	16
E8	15
D0	14
A1	13
43	12
87	11
0E	10
1C	9
39	8
72	7
E5	6
CB	5
97	4
2F	3
5F	2
BF	1
7F	0

MOSTEK®

F8 MICROPROCESSOR DEVICES

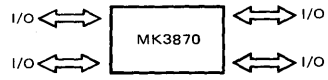
Peripheral Input/Output MK 3871

October 1977

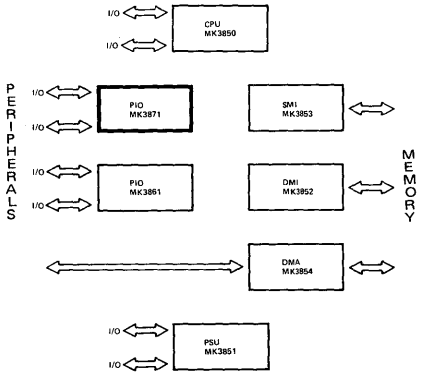
FEATURES

- ☐ Two 8-bit I/O ports
- ☐ Programmable binary timer
- ☐ External/timer interrupt control circuitry
- ☐ Low power dissipation — typically less than 200mW

SINGLE CHIP MK3870



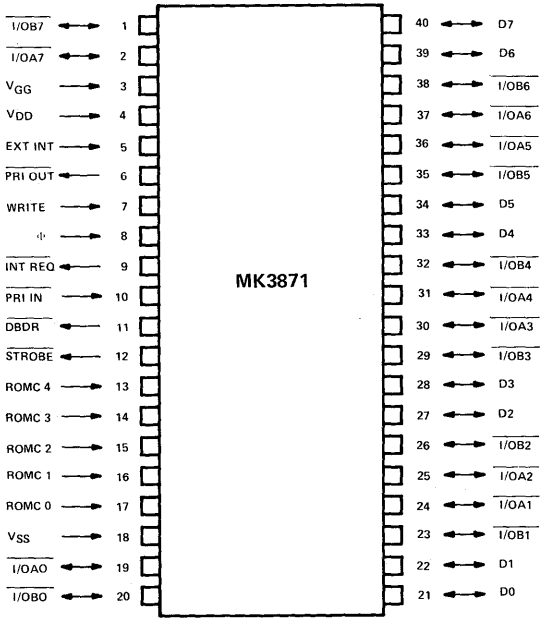
F8 FAMILY



GENERAL DESCRIPTION

The MK3871 Peripheral Input/Output Circuit (PIO) provides two 8-bit I/O ports and a programmable timer for an F8 multi-chip system (MK3850 family). The MK3871 has the same improved timer and ready strobe output as are on the MK3870 single-chip microcomputer. Thus, for software compatibility with the MK3870, the MK3871 PIO should be used in F8 multi-chip configurations rather than the MK3861 PIO. The MK3871 is manufactured using the same N-channel silicon-gate technology as the single chip MK3870 and the multi-chip F8 family.

PIN NAME	DESCRIPTION	TYPE
D0-D7	Data Bus Lines	Bi-Directional, Tri-State
$\overline{\text{I/O A0-I/O A7}}$	I/O Port A	Bi-Directional
$\overline{\text{I/O B0-I/O B7}}$	I/O Port B	Bi-Directional
ROMC 0-ROMC 4	System Control Lines	Input
Φ, WRITE	Clock Lines	Input
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 Lines	Input
$\overline{\text{STROBE}}$	Ready Strobe	Output



FUNCTIONAL PIN DEFINITION

D0 – D7 (BI-DIRECTIONAL, TRI-STATE)

DATA BUS: The Data Bus provides bi-directional communication between the F8 CPU and the 3871 and all other peripheral circuits for transfer of data. D0 is the least significant bit.

I/O A0 – I/O A7 and I/O B0 – I/O B7 (Bi-directional)

I/O PORTS: Two 8-bit I/O ports are located on the 3871 PIO. These ports are referred to as Port A and Port B herein, but the actual port number is determined by the version of the 3871 that is selected. These ports have output latches to hold output data.

ROMC 0 – ROMC 4 (INPUT)

SYSTEM CONTROL LINES: These lines provide the 3871 with control information from the F8 CPU. The CPU sets up these lines early in each machine cycle, and the PIO executes that command during that cycle.

Φ (INPUT)

Φ (PHI) CLOCK: This is the high frequency F8 system clock. It is generated by the F8 CPU. Each machine cycle contains either 4 Φ periods (short cycle) or 6 Φ periods (long cycle).

WRITE (INPUT)

WRITE CLOCK: This clock defines the machine cycle. The cycle starts with the fall of the WRITE clock. The system control lines become stable shortly after the start of the cycle and the PIO decodes and executes the command communicated by the control lines. All ROMC commands are started and completed within one cycle of WRITE.

EXT INT (INPUT)

EXTERNAL INTERRUPT: This is the external interrupt input. It may also be used in conjunction with the timer for pulse width measurement and event counting. Its active state is software programmable.

$\overline{\text{PRI IN}}$ (INPUT)

PRIORITY IN: This input signals the PIO that a higher priority peripheral has an interrupt request pending on the CPU. If an interrupt is received, it will be latched into the PIO but it will not be serviced until $\overline{\text{PRI IN}}$ is in the "low" state.

$\overline{\text{PRI OUT}}$ (OUTPUT)

PRIORITY OUT: This output signals lower priority peripherals that the PIO either has an interrupt request pending on the CPU, or that a still higher priority peripheral has requested an interrupt.

$\overline{\text{INT REQ}}$ (OUTPUT)

INTERRUPT REQUEST: This open drain output is wired OR ed with the corresponding output on all other peripherals to form the interrupt request input to the CPU.

$\overline{\text{DBDR}}$ (OUTPUT)

DATA BUS DRIVE: This output goes "low" whenever the PIO is driving the Data Bus as an output. It may be used to control tri-state buffers in a buffered Data Bus system and to signal other peripherals that the PIO has "control" of the Data Bus at that time.

V_{SS} (INPUT)

V_{SS}: This is system ground (0V.) V_{DD} and V_{GG} are referenced to V_{SS}.

V_{DD} (INPUT)

V_{DD}: Power line; +5V ± 5%.

V_{GG} (INPUT)

V_{GG}: Power line; +5V ± 5% or +12V ± 5%. With V_{GG} at +5V the Data Bus output levels are TTL compatible; however, for a CMOS or MOS higher output level V_{GG} may be connected to +12V.

$\overline{\text{STROBE}}$ (OUTPUT)

PORT A READY STROBE: This pin which is normally high provides a single low pulse after valid data is present on the $\overline{\text{I/O A0}}$ – $\overline{\text{I/O A7}}$ pins during an output instruction.

PIO FUNCTIONAL DIAGRAM

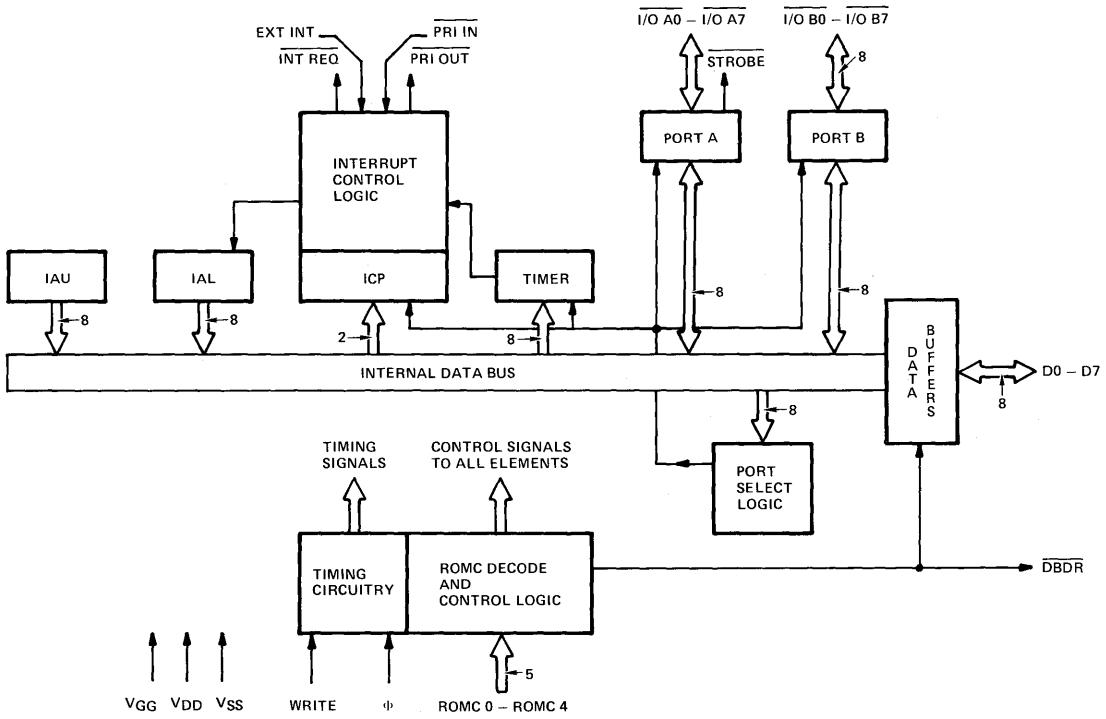


Figure 1.

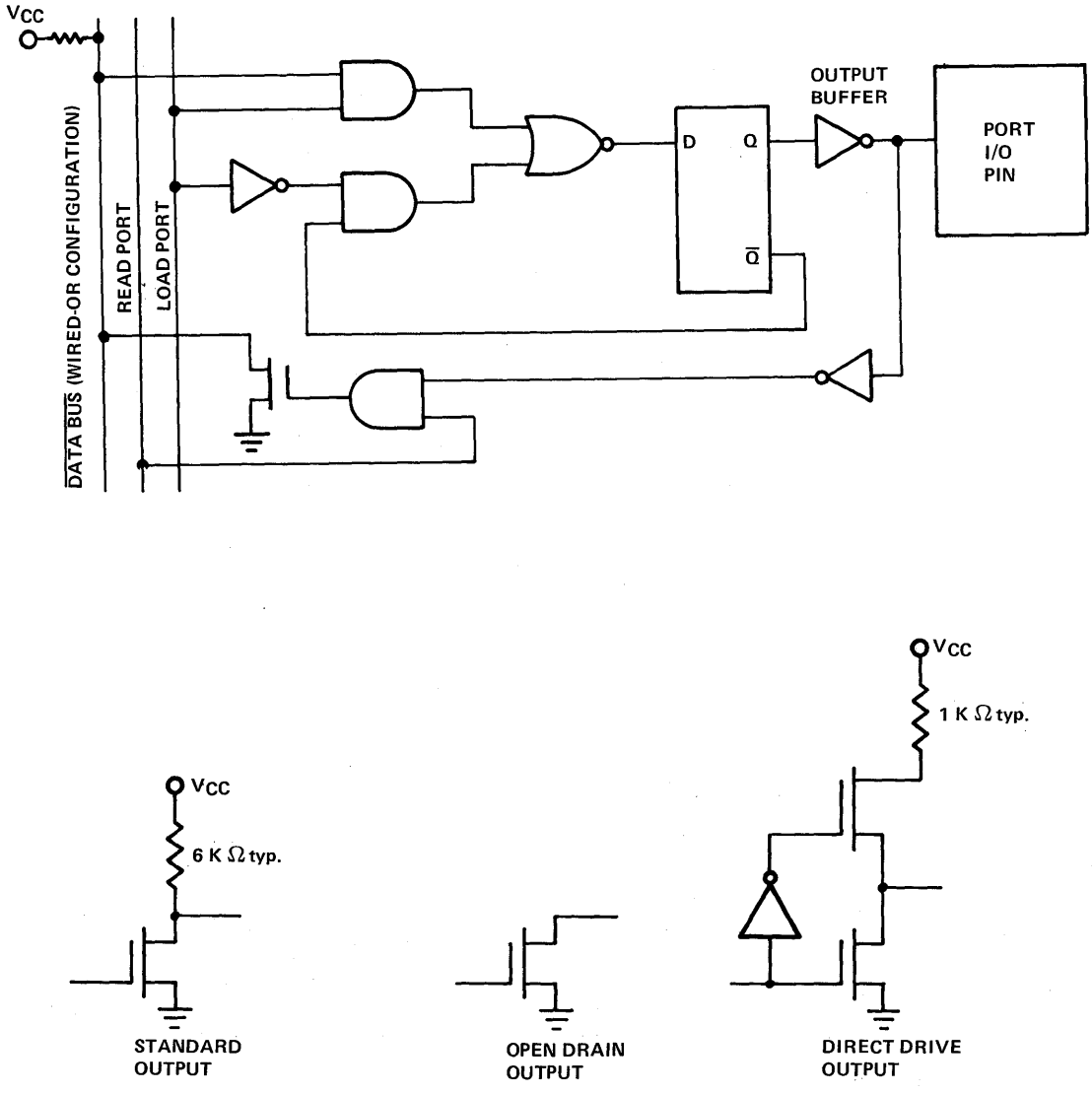
INPUT/OUTPUT PORTS

Each 3871 chip has two bi-directional 8-bit I/O ports. Using binary notation, Port A's address is XXXXXX00 and Port B's address is XXXXXX01, where the X binary digits are the chip's unique I/O port select code. If the port select code, for example, is chosen to be 000001, then Port A may be called Port 4 and Port B may be called Port 5. (The PIO port select code is not permitted to be all 0's since Ports 0 and 1 are reserved for the MK3850 CPU). In addition, the Interrupt Control Port is addressed as port XXXXXX10 and the binary timer is addressed as port XXXXXX11 (which become Ports 6 and 7 for the port select code example given above).

An output instruction (OUT or OUTS) causes the contents of the Accumulator to be latched into the addressed port. An input instruction (IN or INS) transfers the contents of the port to the Accumulator (the Interrupt Control Port is an exception which is described later). The I/O pins on the 3871 are logically inverted. The two I/O ports may both be any of the three output options shown in Figure 2.

An output ready strobe is associated with Port A. This strobe may be used to signal a peripheral device that the 3870 has just completed an output of new data to Port A. The strobe provides a single low pulse shortly after the output operation is completely finished. The $\overline{\text{STROBE}}$ output is always configured similar to a Standard Output (see Figure 2) except that it is capable of driving 3 TTL loads.

I/O PIN CONCEPTUAL DIAGRAM WITH OUTPUT BUFFER OPTIONS



Direct drive ports may be used only as outputs.

The $\overline{\text{STROBE}}$ output is always configured similar to a Standard Output except that it is capable of driving 3 TTL loads.

Figure 2.

TIMER & INTERRUPT CONTROL PORT BLOCK DIAGRAM

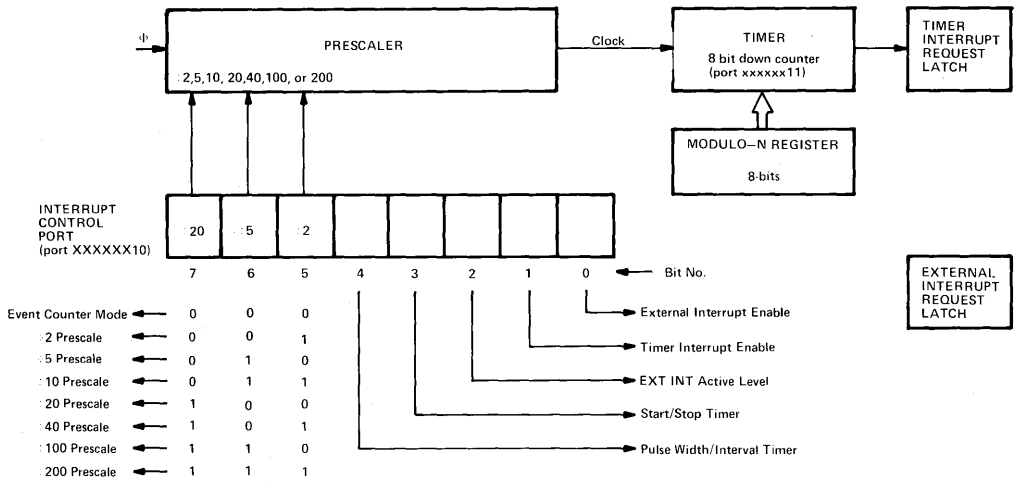


Figure 3.

TIMER

Timer and Interrupt Control Port

The Timer is an 8-bit binary down counter which is software programmable to operate in one of three modes: the Interval Timer Mode, the Pulse Width Measurement Mode, or the Event Counter Mode. As shown in Figure 3, associated with the Timer are an 8-bit register called the Interrupt Control Port, a programmable prescaler, and an 8-bit modulo-N register.

The desired timer mode, prescale value, starting and stopping the timer, active level of the EXT INT pin, and local enabling or disabling of interrupts are selected by outputting the proper bit configuration from the Accumulator to the Interrupt Control Port with an OUT or OUTS instruction. Bits within the Interrupt Control Port are defined as follows:

Interrupt Control Port (Port XXXXXX10)

Bit 0 – External Interrupt Enable
Bit 1 – Timer Interrupt Enable
Bit 2 – EXT INT Active Level
Bit 3 – Start/Stop Timer
Bit 4 – Pulse Width/Interval Timer
Bit 5 – ÷ 2 Prescale
Bit 6 – ÷ 5 Prescale
Bit 7 – ÷ 20 Prescale

A special situation exists when reading the Interrupt Control Port (with an IN or INS instruction). The Accumulator is not loaded with the content of the ICP; instead, Accumulator bits 0 through 6 are loaded with 0's while bit 7 is loaded with the logic level being applied to the EXT INT pin, thus allowing the status of EXT INT to be determined without the necessity of servicing an external interrupt request. This capability is useful in establishing a high speed polled handshake procedure or for using EXT INT as an extra input pin if external interrupts are not required and the Timer is used only in the Interval Timer Mode. However, if it is desirable to read the content of the ICP, then one of the 64 scratchpad registers may be used to save a copy of whatever is written to the ICP.

The rate at which the timer is clocked in the Interval Timer Mode is determined by the frequency of the Φ clock and by the division value selected for the prescaler. If ICP bit 5 is set and bits 6 and 7 are cleared, the prescaler divides Φ by 2. Likewise, if bit 6 or 7 is individually set the prescaler divides Φ by 5 or 20 respectively. Combinations of bits 5, 6 and 7 may also be selected. For example, if bits 5 and 7 are set while 6 is cleared the prescaler will divide by 40. Thus possible prescaler values are: ÷ 2, ÷ 5, ÷10, ÷20, ÷40, ÷100, and ÷200.

Any of three conditions will cause the prescaler to be reset: (1) Whenever the timer is stopped by

clearing ICP bit 3; (2) Execution of an output instruction to the timer (port address XXXXXX11); or (3) On the trailing edge transition of the EXT INT pin when in the Pulse Width Measurement Mode. These last two conditions are explained in more detail below.

An OUT or OUTS to Port XXXXXX11 will load the content of the Accumulator to both the Timer and the 8-bit modulo-N register, reset the prescaler, and clear any previously stored timer interrupt request. As previously noted, the Timer is an 8-bit down counter which is clocked by the prescaler in the Interval Timer Mode and in the Pulse Width Measurement Mode. The prescaler is not used in the Event Counter Mode. The modulo-N register is a buffer whose function is to save the value which was most recently outputted to port XXXXXX11. The modulo-N register is used in all three timer modes.

Interval Timer Mode

When ICP bit 4 is cleared (logic 0) and at least one prescale bit is set the Timer operates in the Interval Timer Mode. When bit 3 of the ICP is set the Timer will start counting down from the modulo-N value. After counting down to H'01', the Timer returns to the modulo-N value at the next count. On the transition from H'01' to H'N' the Timer sets a timer interrupt request latch. Note that the interrupt request latch is set by the transition to H'N' and not by the presence of H'N' in the Timer, thus allowing a full 256 counts if the modulo-N register is preset to H'00'. If bit 1 of the ICP is set, the interrupt request is passed on to the CPU via $\overline{\text{INT REQ}}$. However, if bit 1 of the ICP is a logic 0 the interrupt request is not passed on to the CPU, but the interrupt request latch remains set. If ICP bit 1 is subsequently set, the interrupt request will then be passed on to the CPU. Only two events can reset the timer interrupt request latch; when the timer interrupt request is acknowledged by the CPU, or when a new load of the modulo-N register is performed.

Consider an example in which the modulo-N register is loaded with H '64' (decimal 100). The timer interrupt request latch will be set at the 100th count following the timer start and the timer interrupt request latch will repeatedly be set on precise 100 count intervals. If the prescaler is set at ÷ 40, the timer interrupt request latch will be set every 4,000 Φ clock periods. For a 2 MHz Φ clock this will produce 2 millisecond intervals.

The range of possible intervals is from 2 to 51,200 Φ clock periods (1 μs to 25.6 ms for a 2 MHz Φ clock). However, approximately 50 Φ periods is a practical minimum because the time between setting the interrupt request latch and the execution of the first instruction of the interrupt service routine is at least 27 Φ periods (the response time is dependent upon how many privileged instructions are encountered when the request occurs). To establish time intervals greater than 51,200 Φ clock periods is a simple matter of using the timer interrupt service routine to count the number of interrupts, saving the result in one or more of the scratchpad registers until the desired interval is achieved. With this technique virtually any time interval or several time intervals may be generated.

The Timer may be read at any time and in any mode using an input instruction (IN or INS) and may take place "on the fly" without interfering with normal timer operation. Also, the Timer may be stopped at any time by clearing bit 3 of the ICP. The Timer will hold its current contents indefinitely and will resume counting when bit 3 is again set. Recall, however, that the prescaler is reset whenever the Timer is stopped; thus a series of starting and stopping will result in a cumulative truncation error.

A summary of other timer errors is given in the timing section of this specification. For a free running timer in the Interval Timer Mode, the time interval between any two interrupt requests may be in error by ±6 Φ clock periods although the cumulative error over many intervals is zero. The prescaler and Timer generate precise intervals for setting the timer interrupt request latch but the time out may occur at any time within a machine cycle. (There are two types of machine cycles; short cycles which consist of 4 Φ clock periods and long cycles which consist of 6 Φ clock periods. The WRITE clock corresponds to a machine cycle). Interrupt requests are synchronized with the WRITE clock, thus giving rise to the possible ±6 Φ error. Additional errors may arise due to the interrupt request occuring while a privileged instruction or multicycle instruction is being executed. Nevertheless, for most applications all of the above errors are negligible, especially if the desired time interval is greater than 1 ms.

Pulse Width Measurement Mode

When ICP bit 4 is set (logic 1) and at least one prescale bit is set, the Timer operates in the Pulse Width Measurement Mode. This mode is used for accurately measuring the duration of a pulse applied to the EXT INT pin. The Timer is stopped and the prescaler is reset whenever EXT INT is at its inactive level. The active level of EXT INT is defined by ICP bit 2; if cleared, EXT INT is active low; if set, EXT INT is active high. If ICP bit 3 is set, the prescaler and Timer will start counting when EXT INT

transitions to the active level. When EXT INT returns to the inactive level the Timer then stops, the prescaler resets, and if ICP bit 0 is set an external interrupt request latch is set. (Unlike timer interrupts, external interrupts are not latched if the ICP Interrupt Enable bit is not set).

As in the Interval Timer Mode, the Timer may be read at any time, may be stopped at any time by clearing ICP bit 3, the prescaler and ICP bit 1 function as previously described, and the Timer still functions as an 8-bit binary down counter with the timer interrupt request latch being set on the Timer's transition from H '01' to H 'N'. Note that the EXT INT pin has nothing to do with loading the Timer; its action is that of automatically starting and stopping the Timer and of generating external interrupts. Pulse widths longer than the prescale value times the molulo-N value are easily measured by using the timer interrupt service routine to store the number of timer interrupts in one or more scratchpad registers.

As for accuracy, the actual pulse duration is typically slightly longer than the measured value because the status of the prescaler is not readable and is reset when the Timer is stopped. Thus for maximum accuracy it is advisable to use a small division setting for the prescaler.

Event Counter Mode

When ICP bit 4 is cleared and all prescale bits (ICP bits 5, 6, and 7) are cleared, the Timer operates in the Event Counter Mode. This mode is used for counting pulses applied to the EXT INT pin. If ICP bit 3 is set, the Timer will decrement on each transition from the inactive level to the active level of the EXT INT pin. The prescaler is not used in this mode; but as in the other two timer modes, the Timer may be read at any time, may be stopped at any time by clearing ICP bit 3, ICP bit 1 functions as previously described, and the timer interrupt request latch is set on the Timer's transition from H '01' to H 'N'.

Normally ICP bit 0 should be kept cleared in the Event Counter Mode; otherwise, external interrupts will be generated on the transition from the inactive level to the active level of the EXT INT pin.

For the Event Counter Mode, the minimum pulse width required on EXT INT is 2 Φclock periods and the minimum inactive time is 2 Φ periods; therefore, the maximum repetition rate is 500 KHz.

EXTERNAL INTERRUPTS

When the timer is in the Interval Timer Mode the EXT INT pin is available for non-timer related interrupts. If ICP bit 0 is set, an external interrupt request latch is set when there is a transition from the inactive level to the active level of EXT INT. (EXT INT is an edge-triggered input). The interrupt request is latched until either acknowledged by the CPU section or until ICP bit 0 is cleared (unlike timer interrupt requests which remain latched even when ICP bit 1 is cleared). External interrupts are handled in the same fashion when the Timer is in the Pulse Width Measurement Mode or in the Event Counter Mode, except that only in the Pulse Width Measurement Mode the external interrupt request latch is set on the trailing edge of EXT INT; that is, on the transition from the active level to the inactive level.

INTERRUPT HANDLING

Figure 4 is a block diagram of the interrupt interconnection for a typical F8 system.

Each PIO has a $\overline{\text{PRIORITY IN}}$ and a $\overline{\text{PRIORITY OUT}}$ line so that they may be daisy chained together in any order, to form a priority level of interrupts. When a PIO receives an interrupt (either timer or external) it pulls its $\overline{\text{PRI OUT}}$ output high, signaling all lower priority peripherals that it has a higher priority interrupt request pending on the CPU. Also, when the PIO's $\overline{\text{PRI IN}}$ input is pulled high by a higher priority peripheral, signaling the PIO that there is a still higher priority interrupt request, it passes that signal along by pulling its $\overline{\text{PRI OUT}}$ high. When the CPU processes an interrupt request it commands the interrupting peripheral to place its interrupt vector address on the Data Bus. Only that peripheral whose $\overline{\text{PRI IN}}$ is low and which has an interrupt request pending will respond. Should there be another lower priority peripheral with a pending request, it will not respond at that time because its $\overline{\text{PRI IN}}$ input will be high.

If there is both a timer interrupt request and an external interrupt request when the CPU section starts to process the requests, the timer interrupt is handled first.

Within each local interrupt control circuit there is a 16-bit interrupt address vector. This vector is the address to which the program counter will be set after an interrupt is acknowledged; hence, it is the address of the first executable instruction of the interrupt routine. The 3871 has an interrupt address which is particular to the version of the 3871 selected by the user. Fifteen bits are fixed. These are bits 0

through 6 and 8 through 15. Bit seven (2^7) is dependent upon the type of interrupt. This bit will be a 0 for internal timer generated interrupts and a 1 for external interrupts. When the interrupt logic sends an interrupt request signal and the CPU is enabled to service it, the normal state sequence of the CPU is interrupted at the end of an instruction. The CPU signals the interrupt circuits via the five control lines. The requesting local interrupt circuit sends a 16-bit interrupt address vector (from the interrupt address generator) onto the Data Bus in two consecutive bytes. The address is made available to the program counter via the address demultiplexer circuits. Simultaneously, the address is also made available to all other devices connected to the data bus. It is the address of the next instruction to be executed. The program counter (PO) of each memory device is set with this new address while the stack register (P) is loaded with the previous contents of the program counter. The information in P is lost. Thus, the next instruction to be executed is determined by the value of the interrupt address vector.

The Interrupt Control Bit (ICB) of the CPU (loaded in the W register) allows interrupts to be recognized. Clearing the ICB prevents acknowledgement of interrupts. The ICB is cleared during power on, external reset, and after an interrupt is acknowledged. The interrupt status of the PSU, PIO or MI devices is not affected by the execution of the DISABLE INTERRUPT (DI) instruction. At the conclusion of most instructions, the fetch logic checks the state of the Interrupt Request Line. If there is an interrupt, the next instruction fetch cycle is suspended and the system is forced into an interrupt sequence.

The CPU allows interrupts after all F8 instructions except the following:

(PK)	PUSH K
(PI)	PUSH IMMEDIATE
(POP)	POP
(JMP)	JUMP
(OUTS)	OUTPUT SHORT (Excluding OUTS 00 and 01)
(OUT)	OUTPUT
(EI)	SET ICB
(LR W, J)	LOAD THE STATUS REGISTER FROM SCRATCHPAD

POWER ON

As a result, it is possible to perform one more instruction after the above CPU instructions without being interrupted.

INTERRUPT INTERCONNECTION

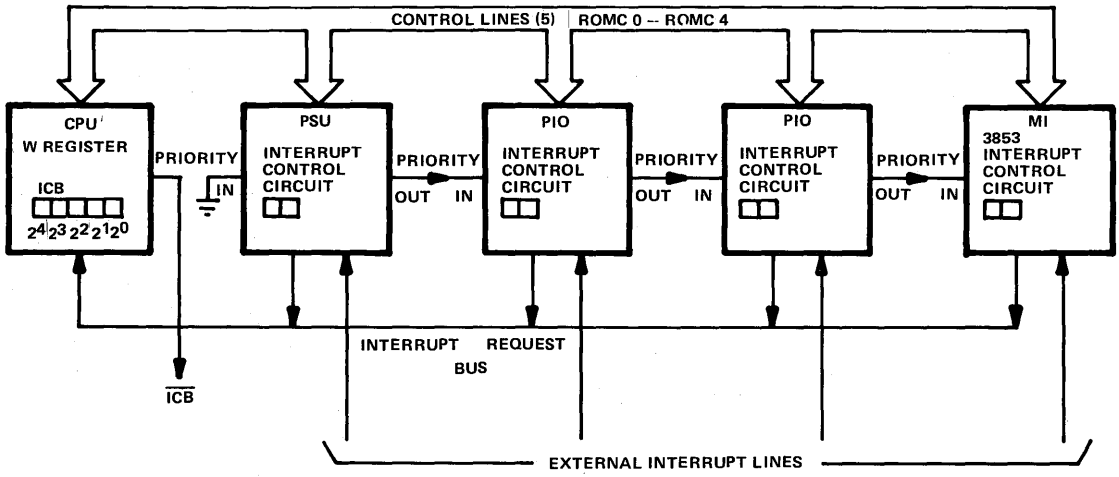


Figure 4.

INTERRUPT SEQUENCE

Figure 5 details the interrupt sequence which occurs whether the interrupt request is from an external source via EXT INT or from the PIO's internal timer. Events are labeled with the letters A through G and are described below.

Event A

An interrupt request must satisfy a setup time requirement as specified on page 19. If not satisfied, INT REQ will delay going low until the next negative edge of the WRITE clock.

Event B

Event B represents the instruction being executed when the interrupt occurs. The last cycle of B is normally the instruction fetch for the next cycle. However, if B is not a privileged instruction and the CPU's Interrupt Control Bit is set, then the last cycle becomes a "freeze" cycle rather than a fetch. At the end of the freeze cycle the interrupt request latches are inhibited from altering the interrupt daisy-chain so that sufficient time will be allowed for the daisy-chain to settle. (If B is a privileged instruction, the instruction fetch is not replaced by a freeze cycle; instead, the fetch is performed and the next instruction is executed. Although unlikely to be encountered, a series of privileged instructions will be sequentially executed without Interrupt. One more instruction, called a 'protected' instruction, will always be executed after the last privileged instruction. The last cycle of the protected instruction then performs the freeze.)

The dashed lines on EXT INT illustrate the last opportunity for EXT INT to cause the last cycle of a non-protected instruction to become a freeze cycle.

The freeze cycle is a short cycle (4 Φ clock periods) in all cases except where B is the Decrement Scratchpad instruction, in which case the freeze cycle is a long cycle (6 Φclock periods).

$\overline{\text{INT REQ}}$ goes low on the next negative edge of WRITE if both $\overline{\text{PRI IN}}$ is low and the appropriate interrupt enable bit of the Interrupt Control Port is set.

Event C

A NO-OP long cycle to allow time for the PRI IN/PRI OUT chain to settle. At a 2 MHz Φ clock rate a total of 7 PIO, PSU, or MI devices may be daisy-chained without the need for look-ahead logic.

INTERRUPT SEQUENCE

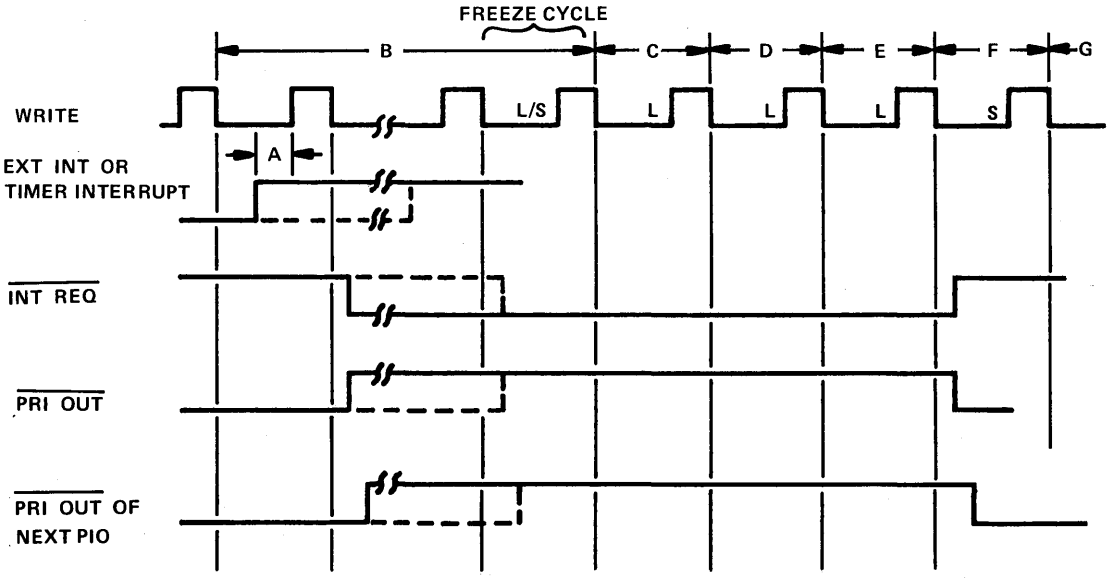


Figure 5.

Event D

In PSU circuits the program counter (PO) is pushed to the stack register (P) in order to save the return address. The interrupting PIO places the lower 8 bits of the interrupt vector address onto the data bus. This is always a long cycle.

Event E

A long cycle in which the PIO places the upper 8 bits of the interrupt vector address onto the data bus.

Event F

A short cycle in which the PIO's interrupting interrupt request latch is cleared. Also, the CPU's Interrupt Control Bit is cleared, thus disabling interrupts until an EI instruction is performed. Additionally, during EVENT F the $\overline{\text{PRI IN/PRI OUT}}$ daisy-chain freeze is removed since the interrupt vector address has been passed to the CPU. Another action is the fetch of the instruction from the interrupt address.

Event G

Begin execution of the first instruction of the interrupt service routine.

Summary Of Interrupt Sequence

For the MK3871 the interrupt response time is defined as the time elapsed between the occurrence of EXT INT going active (or the Timer transitioning to H'N') and the beginning of execution of the first instruction of the interrupt service routine. The interrupt response time is a variable dependent upon what the microprocessor is doing when the interrupt request occurs. As shown in Figure 5, the minimum interrupt response time is 3 long cycles plus 2 short cycles plus one WRITE clock pulse width plus a setup time of EXT INT prior to the leading edge of the WRITE pulse – a total of 27 Φ clock periods plus the setup time. At 2 MHz this is 14.25 μs. Although the maximum could theoretically be infinite, a practical maximum is 35 μs (based on the interrupt request occurring near the beginning of a PI and LR K, P sequence).

DATA FLOW

Table 1 shows the function performed by the PIO for each ROMC command. Each function is entirely performed within one machine cycle (one cycle of the WRITE clock).

TABLE 1

The following ROMC states are decoded by the 3871 as indicated. All other ROMC states are decoded as "NO-OPERATION" (NO-OP).

Binary (R R R R R / O O O O O / M M M M M / C C C C C / 4 3 2 1 0)	Hex	3871 Function
0 1 1 1 1	OF	If this circuit is interrupting and no higher priority circuit is interrupting, move the lower half of the interrupt vector on to the Data Bus and signal Bus use with $\overline{\text{DBDR}}$.
1 0 0 0 0	10	Place interrupt circuitry in an inhibit state that prevents altering the interrupt chain.
1 0 0 1 1	13	If this circuit is interrupting and no higher priority circuit is interrupting move the upper half of the interrupt vector on to the Data Bus and signal Bus use with $\overline{\text{DBDR}}$. In any case, remove priority interrupt circuitry from inhibit state.
1 1 0 1 1	1B	If contents of Data Bus in the previous cycle was an I/O port address, move the contents of that port on to the Data Bus and signal Bus use with $\overline{\text{DBDR}}$ (Input Command).
1 1 0 1 0	1A	If contents of Data Bus in the previous cycle were an address of an I/O port, the timer, or the Interrupt Control Port, move current contents of the Data Bus into that port. (Output Command).
0 1 0 0 0	08	Reset command. Load Port A, Port B, the Interrupt Control Port, and the timer with H'00'.

3871 PIO VERSIONS

Each version of the 3871 is denoted by a 90XXX number. The presently available versions of the 3871 are listed in Table 2.

AVAILABLE VERSIONS OF THE 3871
TABLE 2

VERSION	PORT SELECT CODE	PORT NUMBERS (DERIVED FROM THE PORT SELECT CODE; HEX)	PORT OUTPUT TYPE	INTERRUPT ADDRESS	
				TIMER	EXTERNAL
90070	000001	04 thru 07	Direct Drive	0020	00A0
90071	000001	04 thru 07	Standard	0020	00A0
90072	000001	04 thru 07	Open Drain	0020	00A0
90077	000010	08 thru 0B	Standard	4420	44A0

ELECTRICAL SPECIFICATIONS

ABSOLUTE MAXIMUM RATINGS

V_{GG} ..... −.3V to +15V
V_{DD} ..... −.3V to +7V
Open Drain Option Ports ..... −.3V to +13.2V
All Other Inputs and Outputs ..... −.3V to +7V
Storage Temperature ..... −55° C to +150° C
Operating Temperature ..... 0° C to 70° C

*All voltages are with respect to V_{SS}. Stresses above those listed may cause permanent damage to the device. Exposure to maximum rated stress for extended periods may impair the useful life of the device.

DC CHARACTERISTICS

V_{SS} = 0V, V_{DD} = 5V± 5%, V_{GG} = 12V± 5%
T_A = 0 to 70°C, unless otherwise noted.

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

SUPPLY CURRENTS

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
I_{DD}	V_{DD} Current		25	60	mA	f = 2 MHz, Outputs unloaded
I_{GG}	V_{GG} Current		3	8	mA	f = 2 MHz, Outputs unloaded

DATA BUS (DB0-DB7)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	2.0		V_{DD}	Volts	
V_{IL}	Input Low Voltage	V_{SS}		.8	Volts	
V_{OH}	Output High Voltage	3.9		V_{DD}	Volts	$I_{OH} = -100\ \mu A$
V_{OH}	Output High Voltage	2.4			Volts	$I_{OH} = -100\ \mu A$, $V_{GG} = 5V \pm 5\%$
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	$I_{OL} = 1.6$ mA [1]
I_{IH}	Input High Current	0		1	μA	$V_{IN} = 6V$, 3-State mode
I_{OL}	Input Low Current	0		−1	μA	$V_{IN} = V_{SS}$, 3-State mode
C_I	Input Capacitance			10	pF	3-State mode

CLOCK LINES (Φ, WRITE)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	2.0		V_{DD}	Volts	
V_{IL}	Input Low Voltage	V_{SS}		.8	Volts	
I_L	Leakage Current			±1	μA	$V_{IN} = V_{SS}$ to +6V
C_I	Input Capacitance			10	pF	

PRIORITY IN AND CONTROL ($\overline{\text{PRI IN}}$, ROMC0 – ROMC4)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	2.0		V_{DD}	Volts	
V_{IL}	Input Low Voltage	V_{SS}		.8	Volts	
I_L	Leakage Current			1	μA	$V_{IN} = V_{SS}$ to 6V
C_I	Input Capacitance			10	pF	

PRIORITY OUT ($\overline{\text{PRI OUT}}$)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage	3.9		V_{DD}	Volts	$I_{OH} = -100\ \mu A$
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	$I_{OL} = 1.8$ mA

INTERRUPT REQUEST ($\overline{\text{INT REQ}}$)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage				Volts	Open Drain Output [1]
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	I_{OL} = 1.8 mA
I_L	Leakage Current			1	μA	V_{IN} = 6V, Output device off
C_I	Input Capacitance			10	pF	Output device off

DATA BUS DRIVE ($\overline{\text{DBDR}}$)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage					Open Drain Output
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	I_{OL} = 1.8 mA
I_L	Leakage Current			1	μA	V_{IN} = 6V, Output device off
C_I	Input Capacitance			10	pF	Output device off

EXTERNAL INTERRUPT (EXT INT)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{IH}	Input High Voltage	2.0			Volts	Internal pullup exists
V_{IL}	Input Low Voltage			0.8	Volts	
I_{IL}	Input Low Current			−1.6	mA	V_{IN} = 0.4V
I_{IH}	Input High Current	−100			μA	V_{IN} = 2.4V
C_I	Input Capacitance			10	pF	

READY STROBE ($\overline{\text{STROBE}}$)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage	2.4		V_{DD}	Volts	I_{OH} = −300 μA
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	I_{OL} = 5.0 mA

I/O PORT (STANDARD OUTPUT OPTION)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage	3.9		V_{DD}	Volts	I_{OH} =-30μA
V_{OH}	Output High Voltage	2.4		V_{DD}	Volts	I_{OH} = −100μA
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	I_{OL} = 1.8 mA
V_{IH}	Input High Voltage	2.0		V_{DD}	Volts	Internal Pullup to V_{DD}
V_{IL}	Input Low Voltage	V_{SS}		.8	Volts	
I_{IL}	Input Low Current			−1.6	mA	V_{IN} = .4V[2]
C_I	Input Capacitance			10	pF	

I/O PORT (OPEN DRAIN OUTPUT OPTION)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage			13.2	Volts	External Pullup
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	I_{OL} = 1.8 mA
V_{IH}	Input High Voltage	2.0		13.2	Volts	
V_{IL}	Input Low Voltage	V_{SS}		.8	Volts	
I_L	Leakage Current			5	μA	V_{IN} = 13.2V, Output device off
C_I	Input Capacitance			10	pF	

I/O PORT (DIRECT DRIVE OUTPUT OPTION)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	TEST CONDITIONS
V_{OH}	Output High Voltage	1.5		V_{DD}	Volts	I_{OH} = −1.5 mA
V_{OL}	Output Low Voltage	V_{SS}		.4	Volts	I_{OL} = 1.8 mA
I_{OH}	Output High Current	−1.5	−4.0	−9.0	mA	V_{OH} = 0.7V to 1.5V

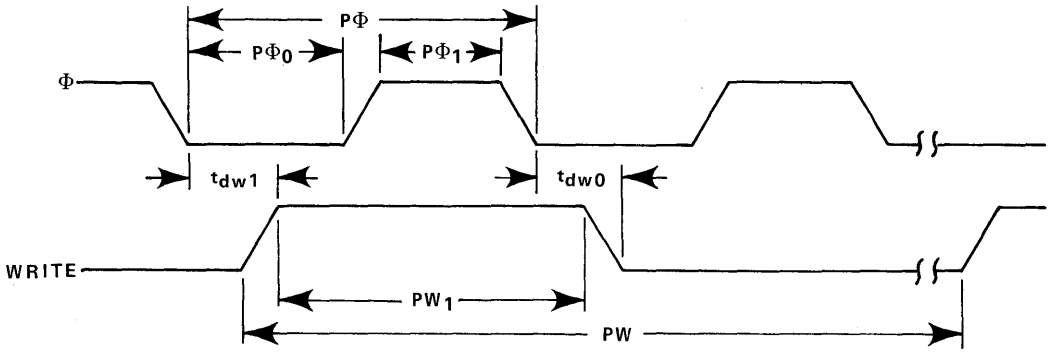
NOTES:

1. Pull up resistor to V_{DD} on CPU.
2. Measured while I/O port is outputting a high level.

TIMING

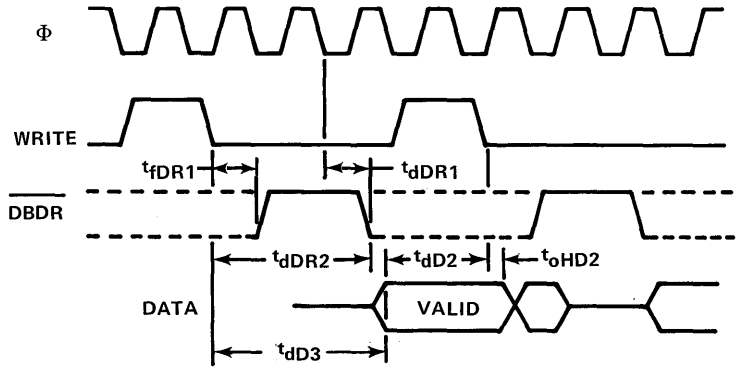
All timing specified at V_{SS} = 0V, V_{DD} = 5V ± 5%, V_{GG} = 12V ± 5%
T_A = 70°C to 0°C

CLOCK TIMING



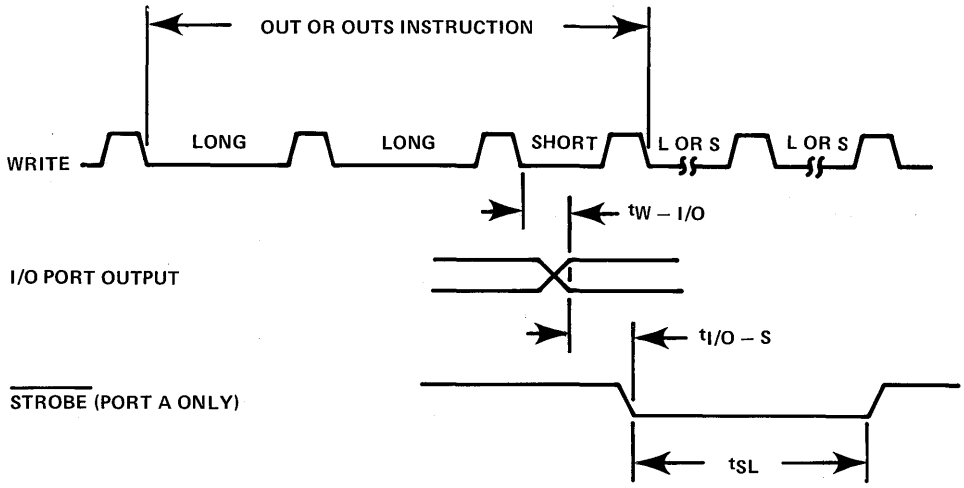
SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	CONDITIONS
PΦ	Clock Period	.5		10	μs	
PΦ	Low time	180			ns	
$P\Phi_1$	High time	180			ns	
PW	WRITE Clock Period		4PΦ			Short cycle
PW_0	WRITE Clock Period		6PΦ			Long cycle
PW_1	WRITE Pulse Width	PΦ−100		PΦ		
t_{dw1}	Φ — to WRITE + delay			250	ns	
t_{dw0}	Φ — to WRITE — delay			225	ns	

OUTPUT TIMING



SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	CONDITIONS
t_{fDR1}	WRITE to $\overline{DBDR}$ floating			400	ns	
t_{dDR1}	Φ to $\overline{DBDR}$ 1-0		200	625	ns	C_L = 100pF R_L = 12.5K
t_{dDR2}	WRITE to $\overline{DBDR}$ 1-0			2P Φ+ 625– tdw0	ns ns	C_L = 100pF R_L = 12.5K C_L = 100pF
t_{dD3}	WRITE to DATA VALID	2P Φ– tdW0	2P Φ– 400	2P Φ+ 700– t_{dW0}	 ns	 C_L = 100pf
t_{0HD2}	Guaranteed Data Hold Time After Fall of WRITE	30			ns	

OUTPUT TIMING (CONT'D)

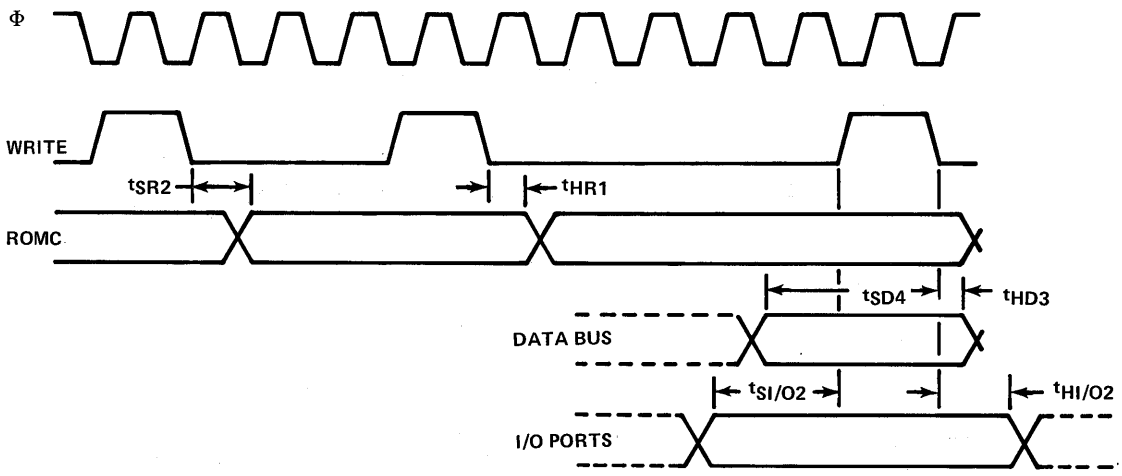


SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	COMMENTS
$\overline{\text{STROBE}}$	$t_{I/O-S}$	Port Output to $\overline{\text{STROBE}}$ Delay	3t Φ-1000	3t Φ + 250	ns	Note 1
	t_{SL}	$\overline{\text{STROBE}}$ Pulse Width, Low	8t Φ-250	12tΦ+250	ns	
I/O PORT	$t_{W-I/O}$	WRITE to I/O Port Output Valid		1000	ns	Note 2

NOTES:

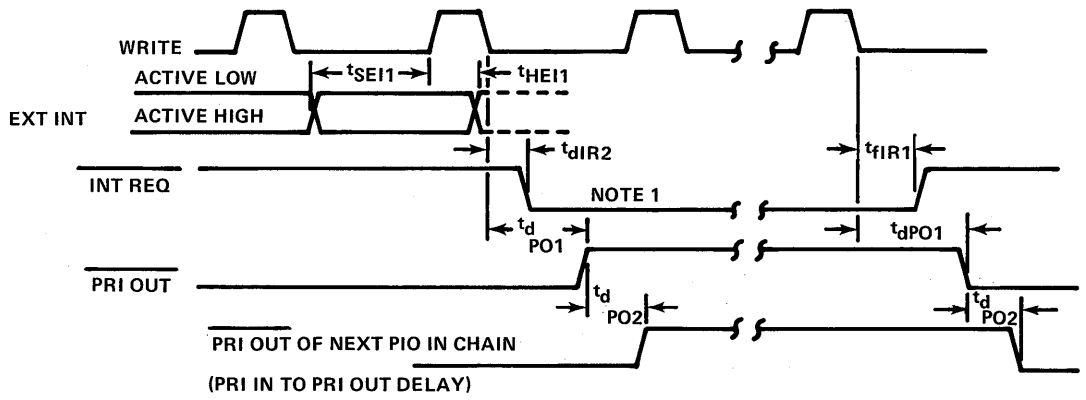
1. Load is 50 pF plus 3 standard TTL inputs.
2. Load is 50 pF plus 1 standard TTL input.

INPUT TIMING



SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	CONDITIONS
t_{SR2}	ROMC Valid Measured from Fall of WRITE			550	ns	
t_{HR1}	ROMC Required Hold After Fall of WRITE	20			ns	
t_{SD4}	Data Bus Set-Up Time				ns	
t_{HD3}	Data Input	20			ns	
$t_{SI/O2}$	I/O Input Set-Up Time	1.3			ns	
$t_{HI/O2}$	I/O Input Hold Time	20			ns	

INTERRUPT TIMING



NOTES:

1. Assuming $\overline{\text{PRI IN}}$ is already low. If not, $\overline{\text{INT REQ}}$ 1–0 transition will be delayed 240 ns max from the time $\overline{\text{PRI IN}}$ is enabled, and $\overline{\text{PRI OUT}}$ 0–1 transition will be delayed t_{dPO2} from the time $\overline{\text{PRI IN}}$ is enabled.

SYMBOL	PARAMETER	MIN	TYP	MAX	UNITS	CONDITIONS
t_{SEI1}	EXT INT Setup Time	750			ns	
t_{HEI}	EXT INT Hold Time	30			ns	
t_{dIR2}	WRITE to INT REQ Delay			430	ns	C_L = 100pF
t_{dPO1}	WRITE to PRI OUT Delay			640	ns	C_L = 50pF
t_{dPO2}	PRI IN to PRI OUT Delay			350	ns	C_L = 50pF
t_{fIR1}	WRITE to INT REQ Float by PIO			640	ns	Open Drain Output

TIMER CHARACTERISTICS

Definitions:

Error = Indicated time value – actual time value

tpsc = t Φ x Prescale Value

Interval Timer Mode:

Single interval error, free running (Note 3)	±6tΦ
Cumulative interval error, free running (Note 3)	0
Error between two Timer reads (Note 2)	±(tpsc + tΦ)
Start Timer to stop Timer error (Notes 1, 4)	+tΦ to –(tpsc + tΦ)
Start Timer to read Timer error (Notes 1, 2)	–5tΦ to –(tpsc + 7tΦ)
Start Timer to interrupt request error (Notes 1, 3)	–2tΦ to –8tΦ
Load Timer to stop Timer error (Note 1)	+tΦ to –(tpsc + 2tΦ)
Load Timer to read Timer error (Notes 1, 2)	–5tΦ to –(tpsc + 8tΦ)
Load Timer to interrupt request error (Notes 1, 3)	–2t Φ to –9tΦ

Pulse Width Measurement Mode:

Measurement accuracy (Note 4)	+t Φ to –(tpsc + 2t Φ)
Minimum pulse width of EXT INT pin	2tΦ

Event Counter Mode:

Minimum active time of EXT INT pin	2tΦ
Minimum inactive time of EXT INT pin	2tΦ

NOTES:

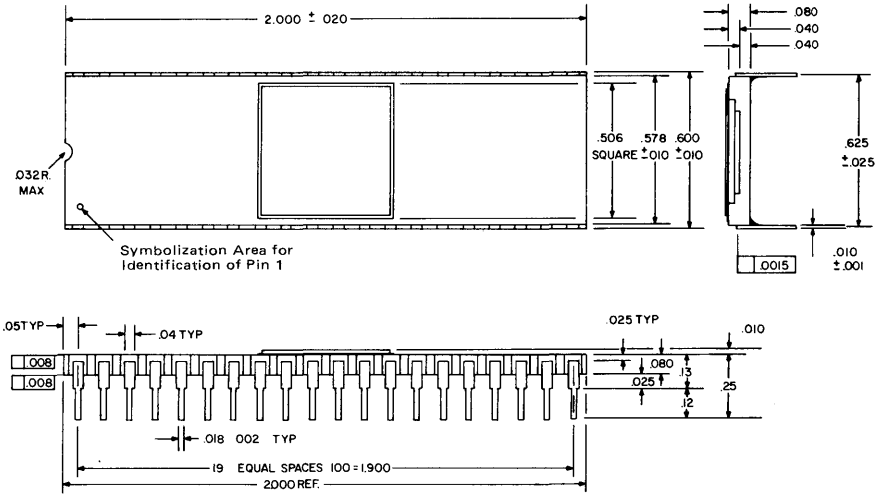
1. All times which entail loading, starting, or stopping the Timer, are referenced from the end of the last machine cycle of the OUT or OUTS instruction.

2. All times which entail reading the Timer are referenced from the end of the last machine cycle of the IN or INS instruction.

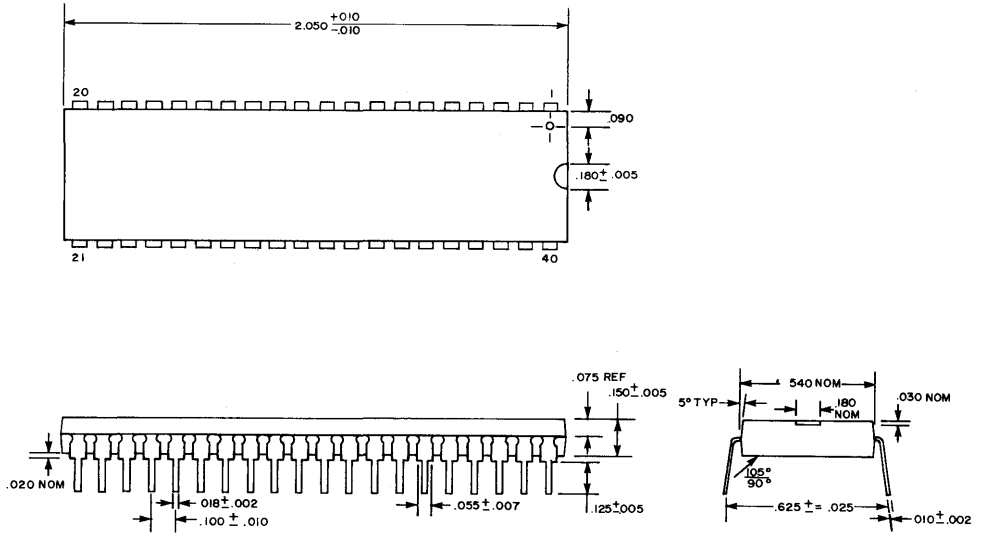
3. All times which entail the generation of an interrupt request are referenced from the start of the machine cycle in which the appropriate interrupt request latch is set. Additional time may elapse if the interrupt request occurs during a privileged or multicycle instruction.

4. Error may be cumulative if operation is repetitively performed. (That is, if the counter if used to total the width of several pulses the error associated with each pulse width measurement will accumulate in the total.)

PACKAGE DESCRIPTION – 40-Pin Dual-In-Line Ceramic Package



PACKAGE DESCRIPTION – 40-Pin Dual-In-Line Plastic Package



ORDERING INFORMATION

Part No.	Package Type
MK3871N/90XXX*	Plastic
MK3871P/90XXX*	Ceramic

*Refer to Table 2 on Page 11 for available 90XXX versions.

3870/F8 SYSTEM DOCUMENTATION

MOSTEK®

3870/F8 MICROCOMPUTER HARDWARE SUPPORT

Evaluation Kit (MCK 50/70)

FEATURES

The MOSTEK F8 Evaluation Kit is a basic F8 evaluation/development microcomputer with these features.

- ☐ 24 bits of I/O arranged in three 8 bit ports
- ☐ 1024 bits of Static Random Access Memory (MK4102)
- ☐ Full duplex TTY Interface (20mA loop)
- ☐ Crystal control clock
- ☐ Non-volatile operating system in MK3851 Program Storage Unit. Firmware called designer development tool L1 (DDT—1)

DESCRIPTION

The F8 Evaluation Kit comes with complete documentation including a detailed application note, programming guide, and a listing of the DDT—1 program.

Purchasers of the Mostek F8 Evaluation Kit will receive free the F8/ANSI Fortran IV Cross Assembler.

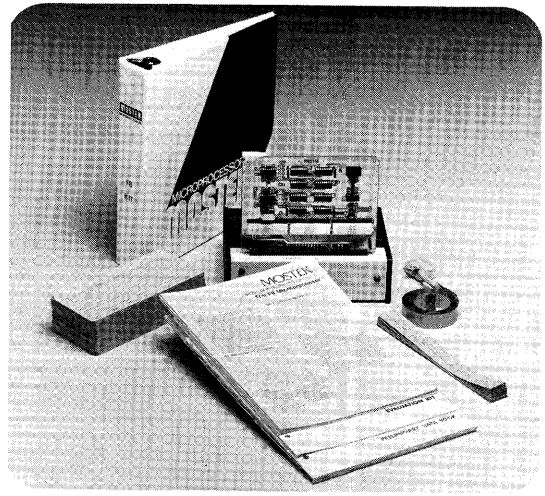
The Evaluation Kit may be ordered as an assembled and tested unit (MK79002), or as an unassembled kit (MK79001) containing all necessary components for assembly including a 72-pin edge connector. A power supply box (MK79003) that provides an edge card connector, all necessary power, switch selectable BAUD rate and a TTY cable is also available.

OPERATION

To operate, you simply attach a 110 or 300 BAUD ASCII terminal (such as a teletype or CRT monitor system) and +5 and +12V power supply. Using DDT—1, you can load, debug and modify your software in the 1K byte of RAM provided in the kit.

DDT—1 provides these features that can be accessed from the ASCII terminal to write and execute your own software.

- ☐ Load command-loads memory from paper tape
- ☐ Dump command-formats data and output to paper tape punch



Assembled F8 Evaluation Kit (79002) and Power Supply (79003)

- ☐ Type command-examines blocks of memory
- ☐ Memory Display and Modify command-examines and modifies memory one byte at a time
- ☐ Copy command-moves blocks of memory from one location to another
- ☐ Port commands-displays and modifies the 24 I/O lines
- ☐ Hexadecimal Arithmetic commands-performs hexadecimal arithmetic
- ☐ Execute command-executes programs at a specific location
- ☐ Breakpoint command-debugs users software

ORDERING INFORMATION

Unassembled Evaluation Kit. Order number MK-79001.

Assembled and Tested Evaluation Kit. Order number MK79002.

Power Supply for Evaluation Kit. Order number MK79003.

MOSTEK®

F8 | MICROCOMPUTER SUPPORT

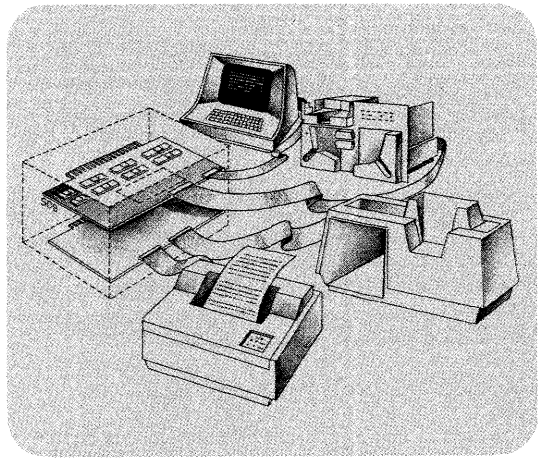
Software Development Board

SOFTWARE FEATURES

- ☐ 2K x 8 Operating System In ROM (DDT-2)
- ☐ 4K x 8 Resident Assembler In ROM
- ☐ Resident Text Editor Loadable In RAM

HARDWARE FEATURES

- ☐ 8K x 8 RAM Memory
- ☐ Four 8 bit I/O Ports
- ☐ Serial ASCII Interface (110-9600 Baud)
- ☐ Parallel Interface For High-Speed Reader/Punch
- ☐ Optional "Application Interface Module" (AIM)



GENERAL DESCRIPTION

The Software Development Board is a complete F8 Microprocessor System designed to aid in developing software for the F8. When combined with power supply, card cage and an ASCII terminal (such as a teletype), it will enable the user to develop the software for all types of F8 applications. This not only includes the ability to execute and debug user software, but also the ability to create and edit "source" listings (using the resident text editor) and assemble them into corresponding "object" code (using the resident assembler). Its other features include 8K x 8 of RAM (expandable with additional memory boards), a variable speed ASCII interface, and resident console and debugging routines. The SDB also includes an interface to an optional high speed paper tape reader/punch. Other peripherals such as a card reader and line printer may be added using an Auxiliary Interface Board.

USING THE SDB

The SDB may be used in two ways. First, as a standalone microcomputer, the SDB may be used to both generate (edit and assemble) and debug F8 Software using the 8K bytes of RAM and 32 bits of I/O available on the board. In many F8 applications, the SDB will thus provide all of the development capability the user will require. Other users, however, may prefer to emulate their application software in the circuit configuration required for their final system. This procedure can significantly reduce the development time for many types of applications. To support these users, an option is available for the SDB called AIM (Application Interface Module). With AIM, the user may apply all of the debug capabilities of the SDB operating system (DDT-2) directly to his final application configuration. As explained in the AIM descriptive literature, this is accomplished without any modifications to the hardware, software, or mechanical packaging of the users final system. The reader is referred to the AIM literature for further information on the use and operation of the SDB with the AIM option.

SPECIFICATIONS

Operating Temperature Range . . . 10°C to 40°C

Power Supply Requirements
+12V ± 5% @ 150mA
+5V ± 5% @ 1.2A
−12 5% @ 50mA

Board Size . . . 8.0"x 12.0" x 1.5"

Connector . . .100 pin edge connector (included)

DDT-2 COMMAND SUMMARY

The DDT-2 Operating system uses 10 basic commands:

.M s	Display and update memory at s
.M s,f	Tabulate memory block s,f
.P s	Display and update port s
.P s,f	Tabulate port block s,f
.E s	Execute program at s
.B s	Set breakpoint to exit program at s
.S s	Step single instruction at s in program
.L	Load tape into memory
.D s,f	Dump tape from memory block s,f
.C s,f,d	Copy memory block s,f to d

The s,f and d represent operands which may be hexadecimal constants, Literals (ASCII Equivalents), predefined mnemonics, or simple arithmetic expressions involving any combination of these. Allowable expressions are of the form ± n1 (=hhhh) ± n2 (=hhhh). . ., where the optional "=" may be used to display the four digit hexadecimal result. Expressions may be utilized in any of the DDT-2 commands, including a 'Dummy' command, 'H', which is provided to permit hexadecimal expression evaluation without performing any other operation.

MEMORY AND PORT COMMANDS (M,P)

The M and P commands provide the user with the means for sequentially accessing F8 I/O ports and memory. Both commands will accept either one or two operands (or operand expressions). With one operand, the contents of the memory or port locations indicated will be displayed and may be optionally modified. Typing carriage returns will automatically display the next successive locations which may also be modified. Typing a '^' will either display the previous location or, if contents of the current location are being changed, display the new contents of the current location. This process will continue until a 'period' is typed to return to the command mode. A 'period' may also be used to abort improperly entered commands. In the example on the adjacent page note the ease with which relative branch offsets may be calculated (at 4106).

With two operands, the M and P commands provide a compact listing of memory or I/O ports. The contents of the addresses specified (inclusively) by the two operands are typed sixteen bytes per line as shown on the adjacent page.

EXECUTE, BREAKPOINT, SINGLE STEP (E,B,S)

The E command is used to execute all programs, including design aids such as the Assembler and Text Editor. The B command may be used to set a Breakpoint to exit from a program at some predetermined location for debugging purposes. At the instant of Breakpoint exit, the contents of all system registers (Scratchpad, Status, Accumulator, etc.) are transferred to a designated 115 byte area of the SDB RAM where they may be examined or modified.

This portion of the SDB memory is called the 'Register Map'. It is also used to initialize system registers whenever execution is initiated (or resumed). Each register image in the Register Map may be accessed using the 'M' command followed by the predefined register mnemonic (or absolute address) of the storage location for that register (example :AC, :IS, :00, ..., :3F, etc). The E and B commands can thus be used together to initialize, execute and examine the results of individual program segments.

When a breakpoint is encountered, the address and accumulator are typed in the stepping format, and the user may continue stepping as above. The breakpoint is cleared automatically to prevent old breakpoints from cluttering up the program.

For a 'Trace' of the execution details of a routine, the programmer may use the S command to step one instruction at a time. With each step, the registers are loaded from the Register Map; the instruction executed; and the registers dumped back to the Register Map. After each step the Register Map may be examined or modified prior to executing the next instruction. The accumulator contents and the address of the next instruction to be executed are always typed after each Step. The programmer continues Stepping by typing carriage returns. In the example a short program has been loaded into memory locations 4100→4106 which will multiply Scratchpad Register R0 times R1 (MOD 256) and place the result in R2.

LOAD, DUMP, COPY (L,D,C)

The L and D commands load and dump object tapes thru the Object channel in standard F8 loader format. Checksums are used for error detection, and the addresses of questionable blocks are typed automatically while loading.

The C command will transfer the contents of the memory block specified by the first two operands to the memory block starting at the location specified by the third operand.

DDT-2 I/O CAPABILITIES

The SDB has 3 I/O channels, designated 'Console', 'Object', and 'Source', to which any suitable devices may be assigned. The Channel Assignment table is located in RAM where it may be updated using the M command. Where mnemonic designations have been predefined, they are automatically substituted for the Table Addresses and the dual byte contents of the table. The Table Addresses correspond to the I/O channels, with the Table Contents corresponding to the addresses of the peripheral driver routines. All the mnemonics used in the example are predefined in DDT-2 Firmware.

When a device is first assigned to a channel, the driver is automatically initialized as required. The user may write his own drivers, define mnemonics for them, and then use those mnemonics to assign them to channels as above. The user may also define mnemonics for any other addresses, such as starting points of programs or subroutines.

SAMPLE PROGRAM EXECUTION

```
. M :00
:00 00 3
:01 01 17
:02 00
. M 4100
4100 90 70
4101 37 42
4102 03 C1
4103 0E 52
4104 2A 30
4105 5F 94
4106 BE 4101-*=FFFB^
4106 FB
. S 4100
*4101 00
*4102 00
*4103 17
*4104 17
*4105 17
*4101 17
*4102 17
*4103 2E
*4104 2E
*4105 2E
*4101 2E
*4102 2E
*4103 45
*4104 45
*4105 45
*4107 45.
. M :00
:00 00
:01 17
:02 45
```

Set R0 = 3, R1 = 17 in Register Map

Ititialize Register Map

```
        LIS   H'0'    R2 = 0
        LR    2,A
Loop    LR    A,2     R2 = R2+R1
        AS    1
        LR    2,A
        DS    0       R0 = R0–1
        BNZ   Loop    (R0 = 0?)
        (Calculate Branch Offset)
```

Load Program
(Solves R0 x R1
MOD 256)

```
ACC =  00; Next Instruction at 4101
       00                    4102
       17                    4103
       17                    4104
       17                    4105
       17                    4101
       17                    4102
       2E                    4103
       2E                    4104
       2E                    4105
       2E                    4101
       2E                    4102
       45                    4103
       45                    4104
       45                    4105
       45                    4107
```

Execute
(Single Step)

R0 = 0
R1 = 17
R2 = 45 (= 3 x 17) MOD 256

Example Register Map

BLOCK MEMORY OPERATIONS

```
 M 4100,4146
4100   00 00 00 00   11 11 11 11   22 22 22 22   33 33 33 33
4110   44 44 44 44   55 55 55 55   66 66 66 66   77 77 77 77
4120   88 88 88 88   99 99 99 99   AA AA AA AA   BB BB BB BB
4130   CC CC CC CC   DD DD DD DD   EE EE EE EE   FF FF FF FF
4140   AE 45 33 28   07 66 CC
. C 4100,4117,4118
. M 4100,4146
4100   00 00 00 00   11 11 11 11   22 22 22 22   33 33 33 33
4110   44 44 44 44   55 55 55 55   00 00 00 00   11 11 11 11
4120   22 22 22 22   33 33 33 33   44 44 44 44   55 55 55 55
4130   CC CC CC CC   DD DD DD DD   EE EE EE EE   FF FF FF FF
4140   AE 45 33 28   07 66 CC
. D 4100,4146
. ********************
S4100
X00000000111111118
X2222222233333338
X4444444455555558
X00000000111111118
X2222222233333338
X4444444455555558
XCCCCCCCCDDDDDDD8
XEEEEEEEEFFFFFFF8
XAE45332807660CB29
********************
```

List Memory Block
4100 thru 4146

Copy Memory Block
4100 thru 4117 to location
4118 thru 412F and list.

Dump Memory Block 4100
thru 4146 in F8 Loader Format

RESIDENT ASSEMBLER

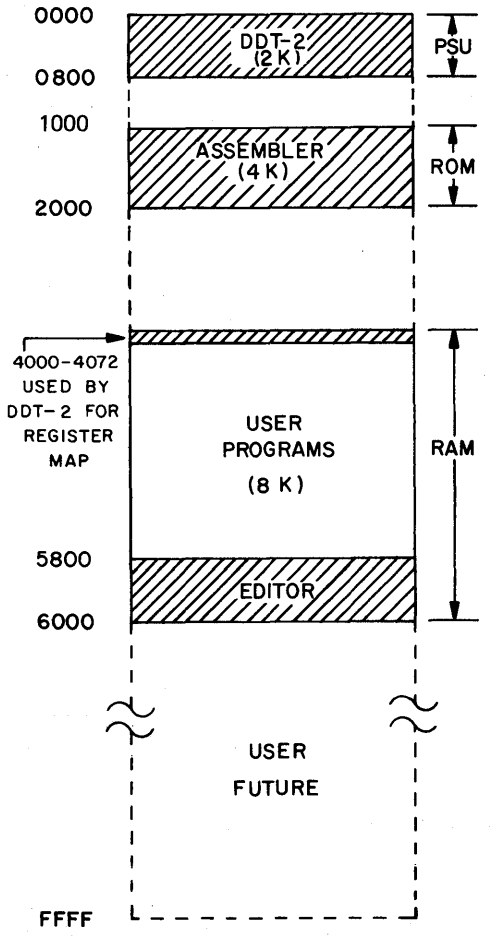
The Resident Assembler in the SDB is a program which translates F8 Assembly Languare Source Statements into Machine Language. The Machine Language produced by the Assembler (called an Object Module) is output in standard F8 Loader Format which may be loaded directly into RAM and Executed. Two Passes are required over the Source input for a complete assembly. The user also has the option of having an 'Assembled Source Listing' produced in addition to the Object Module. The Assembled Source Listing is printed (if desired) during Pass 2, while the Object Module is being buffered in memory. The Object Module is punched after an END statement is encountered or the object buffer has been filled. Buffering the Object Module eliminates the need for the third Pass required with many other Assemblers. This also permits the use of a single peripheral (such as a TTY) for outputting both the Assembled Source Listing and the Object Module without conflict. The only restriction on using the Assembler is that programs having more than 420 Labels must be assembled in sections:

All I/O for the Assembler is handled through the 'Console', 'Object', and 'Source' Channels provided by DDT–2. The Assembler receives Control characters (and responds) via the Console Channel, while the Source Channel is used for the Assembly Language input and the optional Assembled Source Listing output. The Object Channel is used to output the Object Module. All Channels are assigned to the serial ASCII Port when a teletype is the only available peripheral.

TEXT EDITOR

The Text Editor supplied with the SDB is in the form of a Paper Tape which may be loaded into RAM Memory (5800 thru 5FFF) and Executed. The various commands recognized by the Text Editor permit random access editing of ASCII characters strings (as would be stored on magnetic or paper tape). The data to be edited is read into memory where individual characters may be located by position or context. Approximately 5000 characters may be stored in the buffer area from 4100 thru 57FF. Character strings longer than this are edited in blocks with all loading and dumping of the buffer being performed automatically by the Text Editor. The Text Editor and Resident Assembler share the same buffer space and may be used alternately without reloading the Text Editor. While the primary application for the Text Editor is in the editing of Assembly Language Source Statements, it may be applied to any arbitrary ASCII character strings which are partitioned by 'Carriage Returns' into records of not more than 80 characters.

SDB MEMORY MAP



BLOCK DIAGRAM DESCRIPTION

Each of the major circuits shown on the block diagram is described below:

CPU – The Central Processing Unit for the SDB is the MK 3850. There are two eight bit I/O ports on the CPU. They are designated P0 and P1 and are available on the 100 pin edge connector. An eight bit bidirectional data bus is used for data transfer between the CPU and all other blocks in the system. The CPU generated control and timing signals for interface to the other blocks.

PSU – There are two 3851 Program Storage Units on the SDB. The ROM portion of these devices contain the DDT-2 operating system. They also provide four eight bit I/O ports (designated P10, P11, P14, and P15). Two of these ports (P11 and P14) are reserved for use by DDT–2, but the other two ports, as well as the timer and interrupt features or both PSUs, are available to the user.

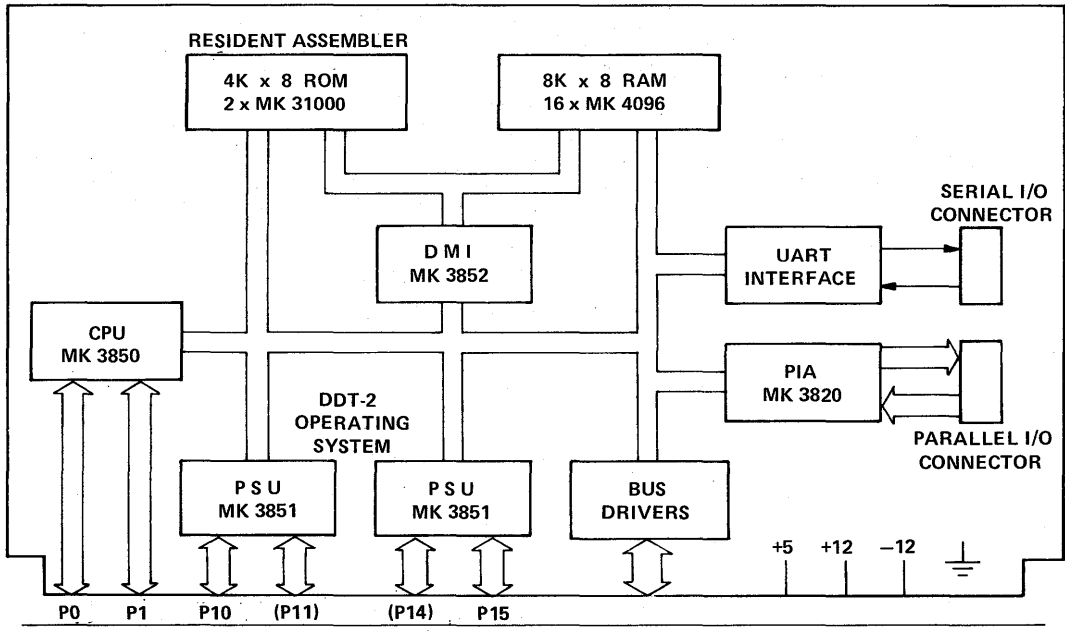
UART – The SDB uses a UART (Universal Asynchronous Receiver Transmitter) device to generate the variable speed serial ASCII interface. The BAUD rate is switch selectable from 110 to 9600 BAUD to be compatible with the various types of teletype and CRT terminals available.

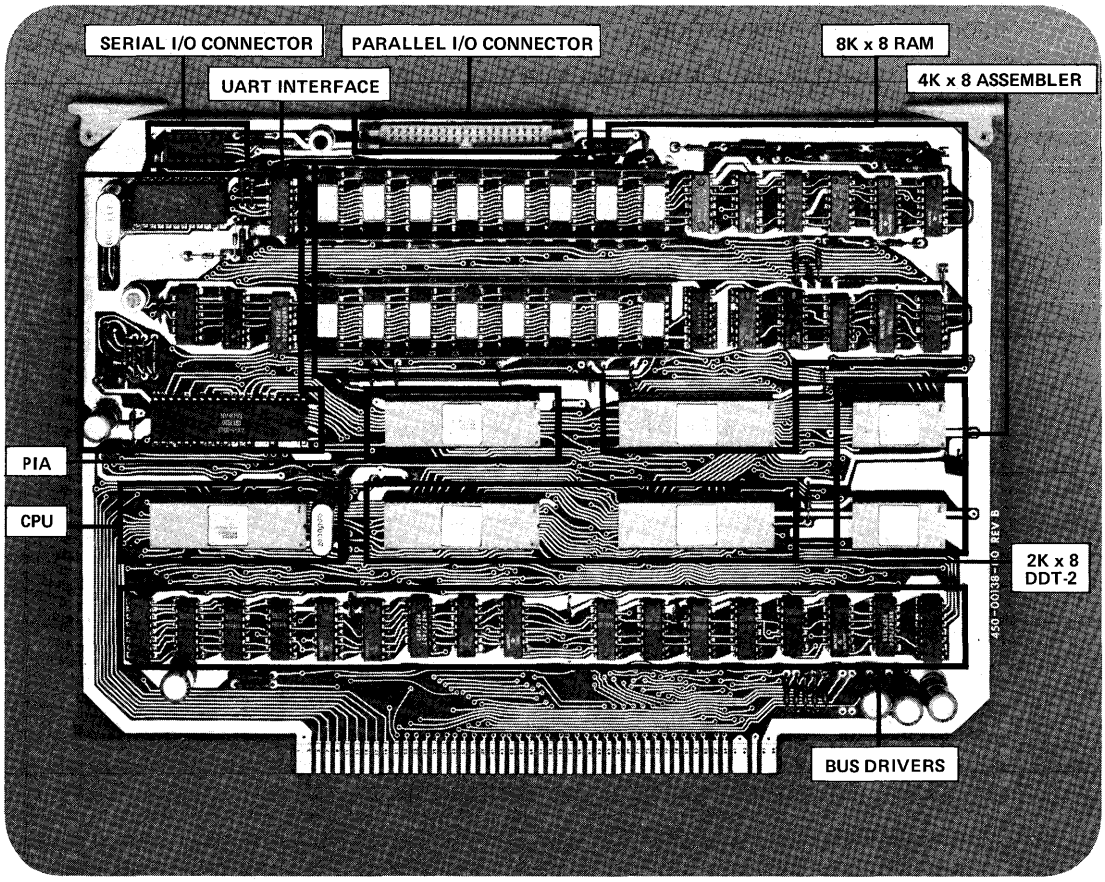
PIA – The MK 3820 Peripheral Interface Adapter provides the two eight bit I/O ports required for the optional high-speed reader/punch interface. This interface is available on the 40 pin 3M connector on the front edge of the SDB.

BUS DRIVERS – All F8 data bus, timing, and control signals are buffered and available on the 100 pin edge connector for expansion.

DMI – The MK 3852 Dynamic Memory Interface generates the timing and address signals for the sixteen MK 4096 (8K bytes) RAM and the two MK 31000 (4K bytes) ROMS.

FUNCTIONAL BLOCK DIAGRAM





ORDERING INFORMATION

PART NUMBER MK 79019

MOSTEK®

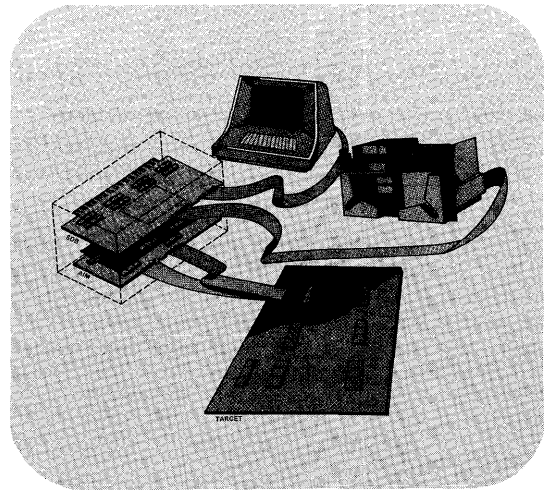
F8 MICROCOMPUTER HARDWARE SUPPORT

Application Interface Module (AIM-70)

October 1977

FEATURES

- ☐ Real time in-circuit emulation
- ☐ Breakpoint insertion
- ☐ Single step operation
- ☐ Direct interface with Mostek's Software Development Board (SDB-50/70)
- ☐ 3K bytes of RAM available during program development



GENERAL DESCRIPTION

AIM-70 (Application Interface Module) is a unique development aid for debugging MK3870 applications in the actual hardware and software configuration of the user's final system (referred to as the 'Target'.) To accomplish this, it is first necessary to emulate the Target ROM with RAM. This RAM must appear as ROM to the application, while retaining the ability to be loaded, debugged, and modified using peripherals independent of the Target. It is the purpose of AIM, used in conjunction with the Software Development Board (SDB-50/70) to provide these capabilities. With AIM-70, all of the peripheral and debugging capabilities of the SDB-50/70 may be applied directly to either the prototype or final production configuration of any MK3870 application; no modifications to the user's hardware, software, or mechanical packaging are required.

USING AIM

The pictorial diagram on the right shows how AIM-70 would typically be used during system development. Because the AIM-70 is an exact functional emulation of the MK3870, it may be directly inserted in the 3870 socket in the target system. Also, since the Target can be a production version of the user's application, product revisions and enhancements may be easily implemented.

As shown in the diagram, the AIM Board is usually mounted in a card cage with the Software Development Board (SDB). It is the purpose of the SDB to provide the user with the means for accessing and controlling the target system (via the AIM Board) during the program development phase. This provides access to all the development software and peripherals of the SDB without having to introduce any perturbations to the target system environment. AIM does not affect the peripheral expansion capabilities of the SDB.

SPECIFICATIONS

Operating Temperature Range. . . . . 0°C to 50°C

Power Supply Requirements
+5V ±5% @ 1.5A max.
+12V ± 5% @ 100mA max.

Board Size. . . . . 8.5" x 12.0" x 1.5"

Connectors/Cables:
40-Pin Ribbon Cable (24" long)

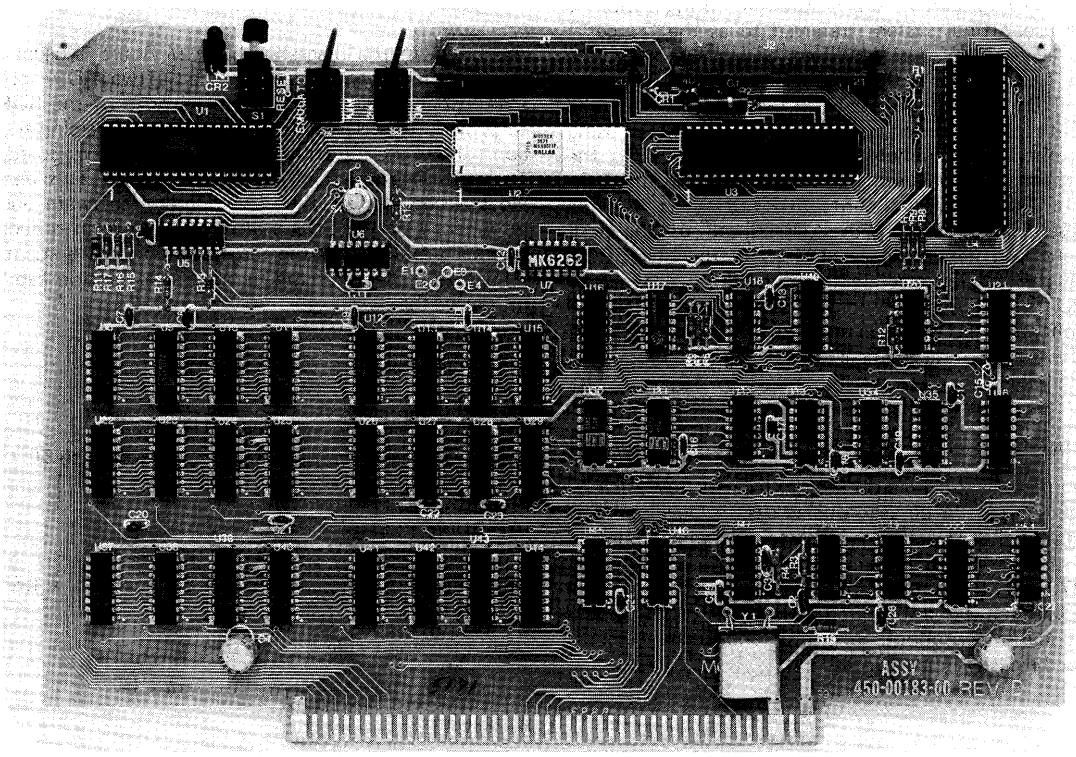
OPERATION DESCRIPTION

The hardware and software associated with AIM have been designed to retain the same command structure as the SDB. The only difference is that all operands (Memory Addresses or Port Addresses) which correspond to the 'Target' system must be preceded by the letter 'T'. The commands available with AIM are summarized below. Designations s, f and d stand for operands.

.M Ts	Display and update target memory at s
.M Ts, Tf	Tabulate target memory block s,f
.P Ts	Display and update target port s
.P Ts,Tf	Tabulate target port block s,f
.E Ts	Execute target program at s
.B Ts	Set breakpoint to exit target program at s
.S Ts	Begin single step execution at s in target program
.L T	Load formated tape into target memory
.D Ts,Tf	Dump formated tape from target memory block s,f
.C Ts,Tf,Td .C s,f, Td .C Ts,Tf,d	Copy memory block s,f in the SDB or target to the memory block location starting at address d in the SDB or target

Each of these SDB commands may be applied to any portion of the target system's port or memory map. This is accomplished by means of a 'handshaking' procedure between the CPU on the AIM and the CPU in the SDB. Handshaking is initiated whenever a target system breakpoint has been encountered, or the single step execution of a target instruction has been completed. Also, whenever handshaking is initiated, the contents of all target system registers (Scratchpad, Status, Accumulator, etc.) are transferred to a designated portion of the SDB memory map where they may be examined or modified. This portion of the SDB memory is called the 'Register Map' and is also used to initialize the target system register whenever execution is initiated (or resumed) in the target system.

AIM–70 PHOTO



BLOCK DIAGRAM DESCRIPTION

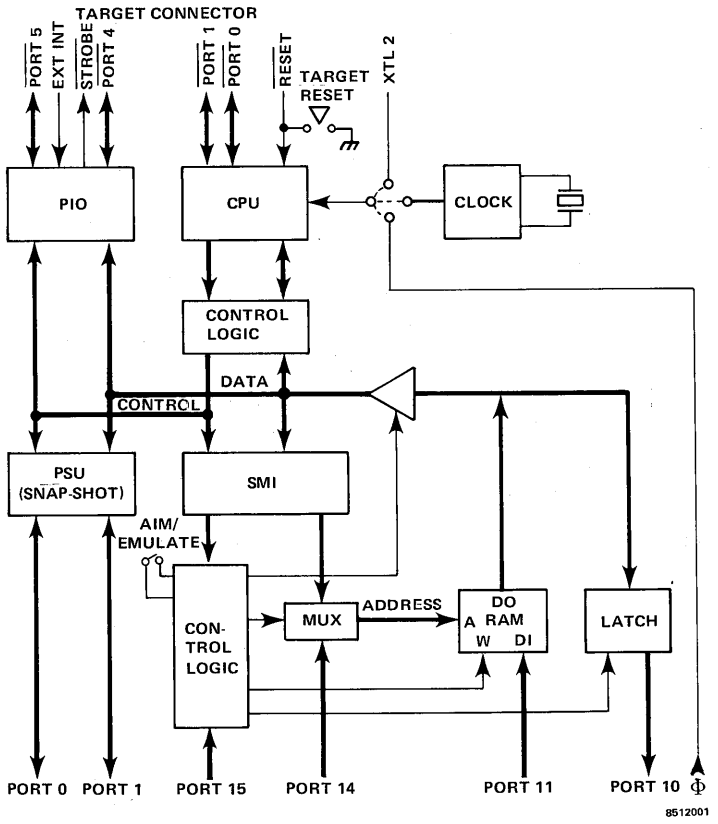
As shown in the block diagram, the AIM-70 contains all the functional elements necessary to emulate the MK3870. The portion of the handshaking software (called 'Snapshot') which resides in the target memory map is located in the PSU on the AIM Board. An MK3871 (PIO), is used to emulate the I/O timer, and interrupt features of the MK3870. Note that the AIM-70 contains 3K bytes of RAM memory-1K bytes more than required to emulate the MK3870's 2K bytes of ROM. The extra 1Kx8 of RAM is provided for use during program development for 'Patches' and to allow execution of the user's program prior to final optimization and code reduction. The AIM/EMULATE switch is provided to disable the extra 1K of memory and the upper 5 bits of the program and data counters. When the AIM/EMULATE switch is in the Emulate position, the AIM-70 is an exact RAM based equivalent of the MK3870. When the switch is in the AIM position, the expanded memory and handshaking are available for use during program development.

The AIM-70 clock may be from either the Target system, from an on-board crystal oscillator, or from the SDB-50/70 clock.

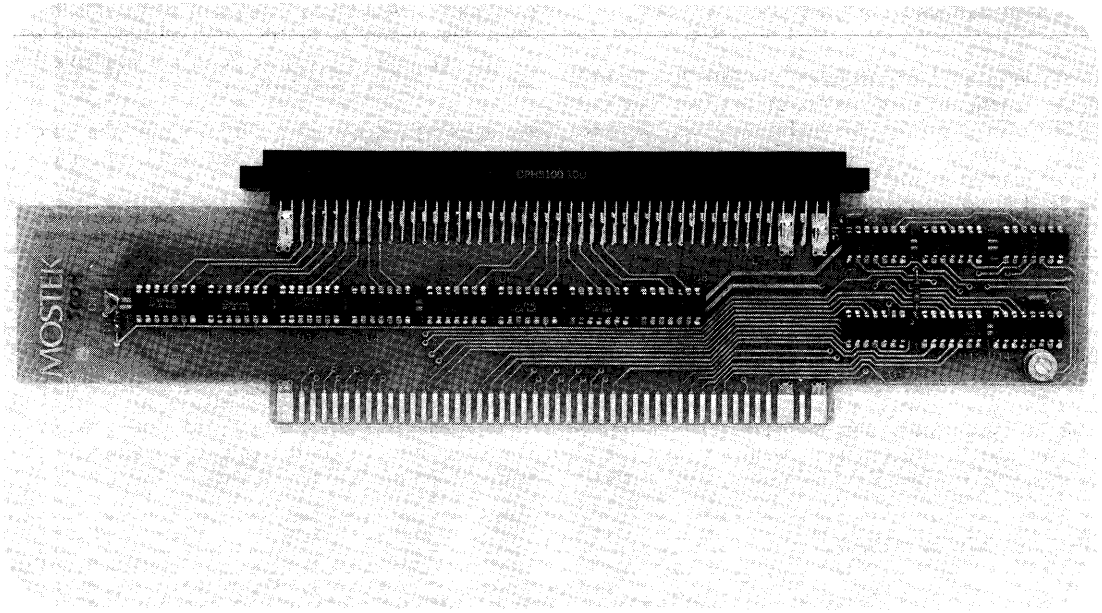
MULTI–3870 APPLICATIONS

For debugging applications incorporating more than one MK3870, multiple AIM–70s may be used in a single SDB/AIM development system. For these systems one AIM–70 plus an adaptor board is required for each MK3870 being emulated in the system. The adaptor board (designated AIM-70X) permits the use of a single SDB-50/70 for controlling up to seven AIM–70 boards. This adaptor board is physically inserted in the card cage between each AIM–70 and the SDB-50/70 bus.

AIM 70 BLOCK DIAGRAM



AIM–70X PHOTO



ORDER INFORMATION

Name	Description	Part No.	Price
AIM–70 Operations Manual	Contains a complete description of the use and operation of AIM–70 and AIM–70X for developing software for 3870 applications.	MK79549	$ 3.00
AIM–70	Includes the complete AIM–70 circuit board with the above described documentation.	MK79031	$ 750.00
AIM–70X	Includes the AIM–70X circuit board with the AIM–70 operations manual.	MK79053	$ 125.00
SDB-50/70	Includes the SDB-50/70 circuit board with complete documentation. The SDB-50/70 is used both with the AIM-70 and as a stand-alone microcomputer with resident firmware for F8 program assembly and text editing.	MK79019	$1295.00

*All prices are subject to change without notice and apply only within the U.S. and Canada

MOSTEK®

3870 MICROCOMPUTER SYSTEMS

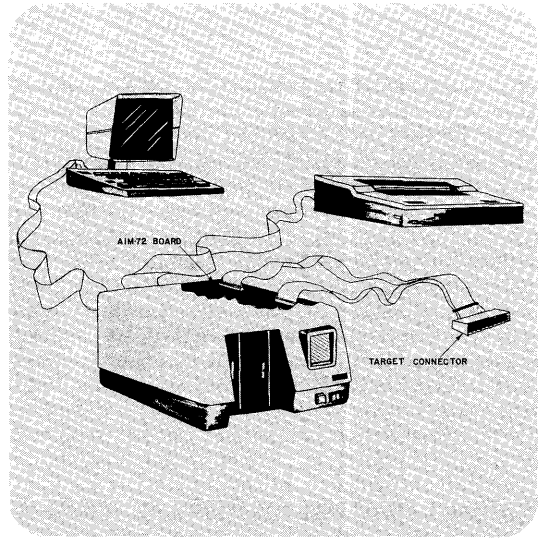
Application Interface Module (AIM-72)

FEATURES

- ☐ Real time in-circuit emulation of Mostek's 3870 family of microcomputers, including MK3870 MK3872 and MK3876
- ☐ Direct interface to Mostek's AID-80F Dual Floppy Disk Microcomputer with ZAIM-72 software supplied on floppy disk
- ☐ Direct interface with Mostek's SDB-50/70 (Software Development Board) with FAIM-72 software supplied on paper tape
- ☐ Standard features include:
  - Breakpoint insertion, memory display and modification, register display and modification, port display and modification, and single step
  - Execution intercept from user keyboard with the ESCAPE key
  - Debugging or emulation mode is selectable from the user's console
  - Debugging of 3870 and F8 programs up to 8K long can be done without a target system

GENERAL DESCRIPTION

AIM-72 (Application Interface Module) is a unique development aid for debugging 3870 Series Microcomputer applications in the actual hardware and software configuration of the user's final system (referred to as the 'Target'.) To accomplish this, it is first necessary to emulate the Target ROM with RAM. This RAM must appear as ROM to the application while retaining the ability to be loaded, debugged, and modified using peripherals independent of the Target. It is the purpose of AIM-72, used in conjunction with the AID-80F Disk Based Microcomputer or the SDB-50/70, to provide these capabilities. With AIM-72, all of the peripheral and debugging capabilities of the user's development system may be applied directly to either the prototype or final production configuration of any 3870, 3872 or 3876 application; no modifications to the user's hardware, software, or mechanical package are required.



USING AIM-72

The pictorial diagram above shows how AIM - 72 would typically be used during system development. Because the AIM-72 is an exact functional emulation of the 3870 family, it may be directly inserted into the 3870, 3872, or 3876 socket in the target system. Also, since the Target can be a production version of the user's application, product revisions and enhancements may be easily implemented. As shown in the diagram, the AIM board is usually mounted in the card cage of the user's development system. It is the purpose of the SDB to provide the user with the means for accessing and controlling the target system (via the AIM board) during the program development phase. This provides access to all the debugging software and peripherals of the development system without having to introduce any perturbations to the target system environment. AIM does not affect the peripheral expansion capabilities of the development system.

BLOCK DIAGRAM DESCRIPTION

As shown in the block diagram, the AIM-72 contains all the functional elements necessary to emulate 3870 Series Microcomputers. Target Ports are emulated with the CPU and PIO Ports. Target ROM

SPECIFICATIONS

Operating Temperature Range	0°C to 50°C
Power Supply Requirement	+5V ± 5% @ 1.5A max.
	+12 V ± 5% @ 100mA max.
	−12 V ± 5% @ 30mA max.
Board Size	8.5" x 12.0" x .75"
Connectors/Cables	40-Pin Ribbon Cable (24" long)

and RAM are emulated with the 8K x 8 RAM which can also be accessed directly by the control system via the bottom edge connector. System memory accesses are transparent to the Target system execution. Thus, there is no impact on target execution timing. The Target memory map can be controlled from the system allowing 2K, 4K or 8K Bytes of memory to be available in the Target System. Debug firmware in a PSU on the AIM-72 interfaces with the system to implement the breakpoint, single step and other functions. Trap control circuitry allows the use of a single byte breakpoint, providing complete flexibility when using break points in tight programming loops. Execution is at full speed, determined only by the user's crystal frequency - no speed reduction is introduced by the AIM's operating system. The AIM-72 clock may be implemented from the Target system, from an on-board crystal oscillator, or from the SDB-50/70 clock.

MULTI 3870 SERIES APPLICATIONS

Up to eight AIM-72 boards may be installed in one control system with each AIM-72 used to emulate a different 3870 Series Microcomputer. The debug functions on each AIM-72 may be enabled one at a time and each program developed until all Target programs are functional. Only one AIM-72 may be in the active debug mode at a time; other AIM-72's will be in the emulator mode.

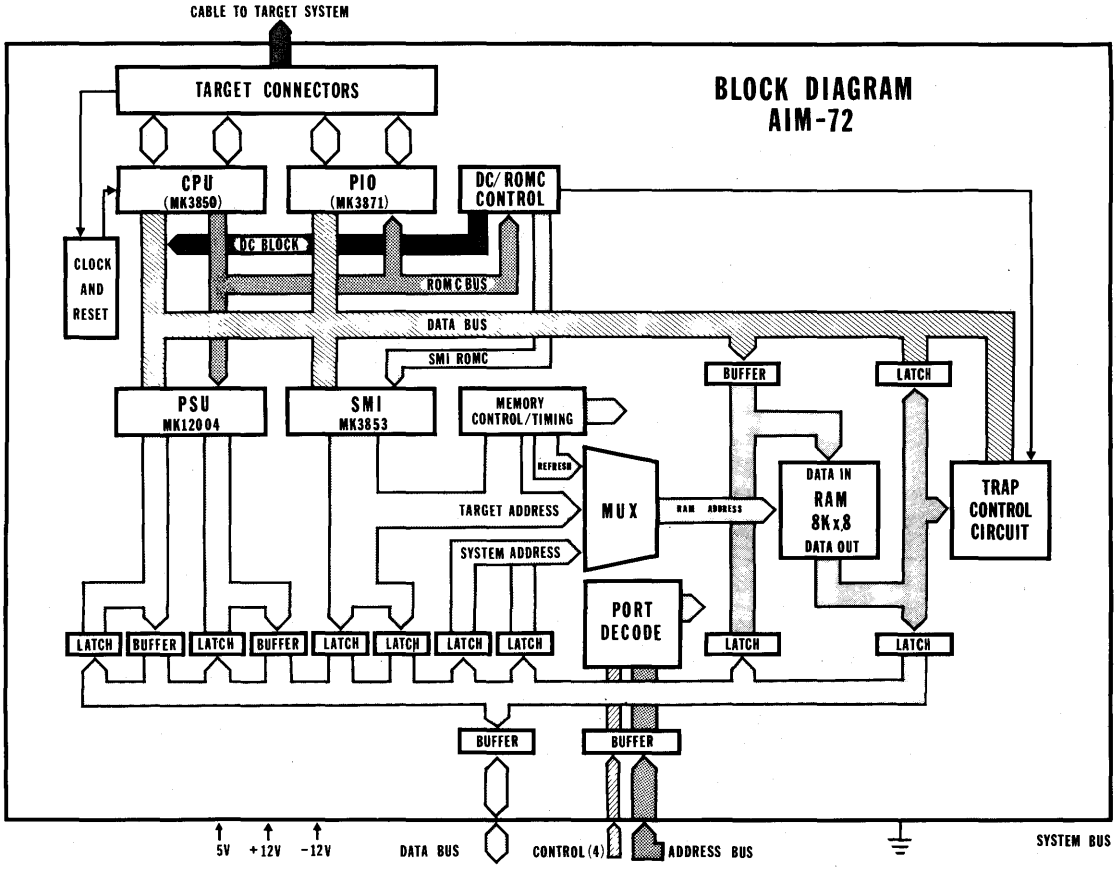
FAIM-72 SOFTWARE

FAIM-72 is the software designed to operate the AIM-72 board with the SDB-50/70 Software Development Board. It is supplied on a paper tape or cassette for loading into the SDB-50/70 memory.

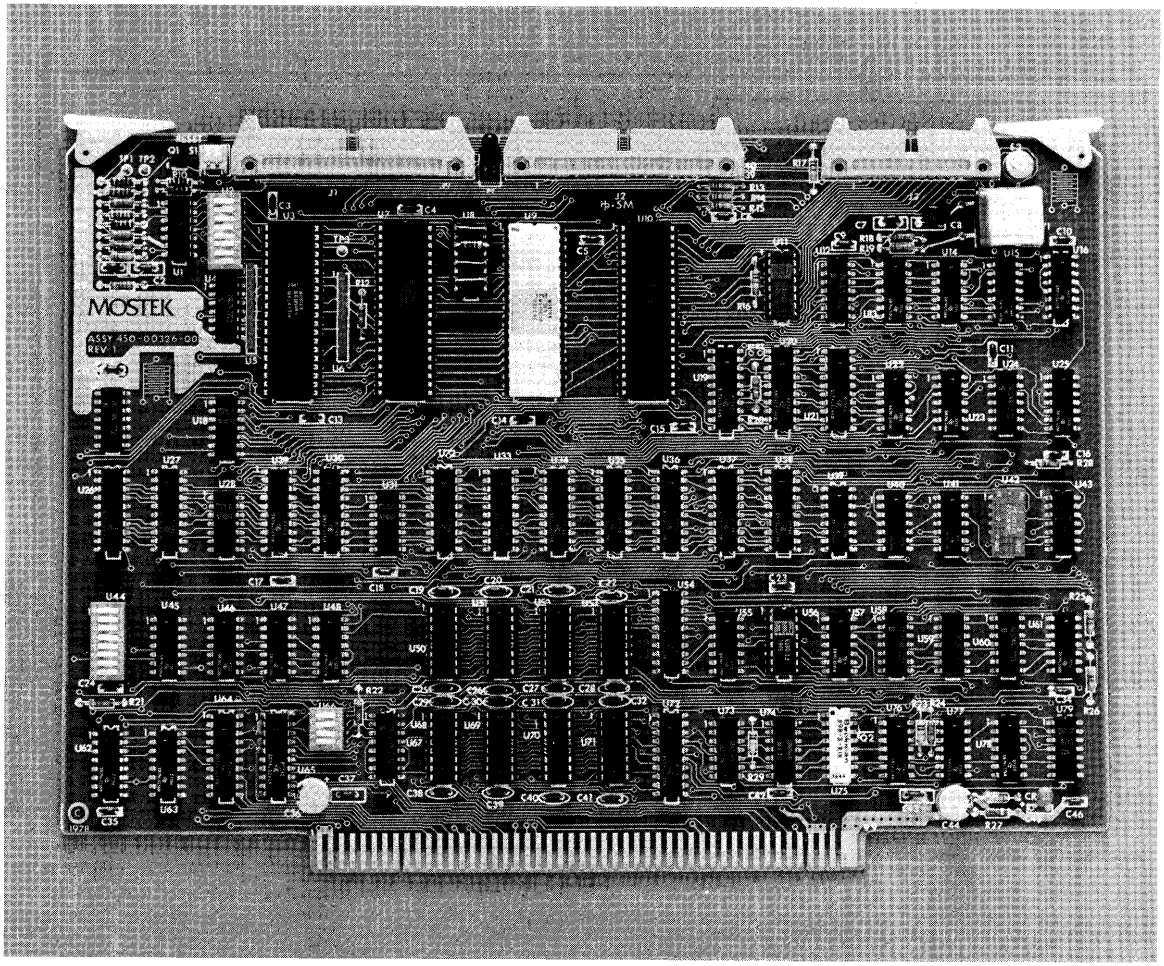
The hardware and software associated with AIM have been designed to retain the same command structure as the SDB. The only difference is that all operands (Memory Addresses or Port Addresses) which correspond to the 'Target' system must be preceded by the letter 'T'. The commands available with FAIM-72 are summarized. Designations s, f, and d stand for operands.

,B Ts	Set breakpoint to exit target program at address s
,C Ts, Tf, Td ,C s, f, Td ,C Ts, Tf, d	Copy memory block from address s thru address f in the SDB or target to the memory block starting at address d in the SDB or target
,D Ts, Tf	Dump formatted tape from target memory block from address s thru address f.
,E Ts	Execute target program at address s
,I	Re-initialize AIM-72
,L T	Load formatted tape into target memory
,M Ts	Display and update target memory at address s
,M Ts, Tf	Tabulate target memory block from address s thru address f
,P Ts	Display and update target port s
,P Ts, Tf	Tabulate target ports s thru f
,Q	Return to DDT-2
,S Ts	Begin single step execution at address s in target program

Each of these SDB commands may be applied to any portion of the target system's port or memory map. This is accomplished by means of a 'handshaking' procedure between the CPU on the AIM and the SDB. Handshaking is initiated automatically when the system is initialized or whenever single-step execution of a target instruction has been completed or when a breakpoint is encountered. Also, whenever handshaking is initiated, the contents of all target system registers (Scratchpad, Status, Accumulator, etc.) are transferred to a designated portion of the SDB memory map where they may be examined or modified. This portion of the SDB memory is called the 'Register Map' and is also used to initialize the target system register whenever execution is initiated (or resumed) in the target system.


CABLE TO TARGET SYSTEM
TARGET CONNECTORS
BLOCK DIAGRAM
AIM-72
CPU
(MK3850)
PIO
(MK3871)
DC/ROMC
CONTROL
CLOCK
AND
RESET
DC BLOCK
ROMC BUS
DATA BUS
SMI ROMC
BUFFER
LATCH
PSU
MK12004
SMI
MK3853
MEMORY
CONTROL/TIMING
REFRESH
MUX
TARGET ADDRESS
SYSTEM ADDRESS
RAM ADDRESS
DATA IN
RAM
8Kx8
DATA OUT
TRAP
CONTROL
CIRCUIT
PORT
DECODE
LATCH
BUFFER
LATCH
BUFFER
LATCH
LATCH
LATCH
LATCH
LATCH
LATCH
BUFFER
BUFFER
5V
+12V
-12V
DATA BUS
CONTROL (4)
ADDRESS BUS
SYSTEM BUS

AIM-72 PHOTO



ZAIM-72 SOFTWARE DESCRIPTION

ZAIM-72 is the software designed to operate the AIM-72 board on Mostek's AID-80F Dual Floppy Disk Microcomputer. It is supplied on a standard FLP-80DOS diskette. The software has the same command structure as other Mostek debuggers. The commands available with ZAIM-72 are summarized below. Designations s,f, and d stand for operands.

,A s,f	Assign data byte f to target memory location s.
,B s	Set a breakpoint at target memory location s. Up to 8 breakpoints can be set at once.
,C s, f, d	Copy the target memory block s to f to target memory starting at d.
,E s	Execute target program at location s.
,F s, f, d	Fill target memory locations s through f with data d.
,G s	Get binary file s and load it into Target memory.
,H	Hexadecimal arithmetic.
,I	Reinitialize target system.
,L s, f, d	Locate data d in target memory range s through f.
,M s	Display and update target memory at location s.
,M s, f, d	Tabulate target memory locations s through f. Option d specifies additional printout of ASCII characters or disassembly.
,O s	Set relative offset equal to s for all address operands.
,P s	Display and update target port number s.
,Q	Quit and return to FLP-80DOS Monitor.
,R s, f	Display target registers, Option s allows a heading to be printed and option f specifies the number of scratchpad registers to be displayed.
,S s, f	Single step starting at target location s for f number of steps.
,V s, f, d	Verify target memory block s through f against target memory block starting at location d.

Target system programs are developed using the Mostek AID-80F Cross Assembler for 3870/F8 Microcomputers (FZCASM-MK79079). Then ZAIM-72 is used to debug the completed program on the user's target system. The software features multiple breakpoints, single step, and in-line disassembly. Target system memory, ports, and registers may be displayed and updated.

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NO	PRICE
AIM-72 Operations Manual	Contains a complete description of the use and operation of AIM-72.	MK79577	$5.00
AIM-72	Includes the AIM-72 circuit board, the AIM-72 Operations Manual, the ZAIM-72 software on diskette, and the FAIM-72 software on paper tape for developing 3870 Series Applications.	MK79076	$1295.00
AID-80F	A complete dual floppy disk development system (less terminal and line printer). Order FZCASM to provide 3870 series assembly capability. Order AIM-72 to provide 3870 Series debug capability.	MK78125	$5995.00
FZCASM	AID-80F Cross Assembler for 3870/F8 Microcomputers. Provides disk-based assembly for 3870 assembly language programs on the Mostek AID-80F Microcomputer.	MK79079	$400.00
SDB-50/70	Includes the SDB-50/70 circuit board with complete documentation. The SDB-50/70 is used with the AIM-72 as a stand alone microcomputer with resident software for 3870 series program assembly and debug.	MK79019	$1295.00
FAIM-72	Optional cassette tape based AIM-72 software for use with Silent 700 terminal and SDB-50/70 development system.	MK79083	$50.00

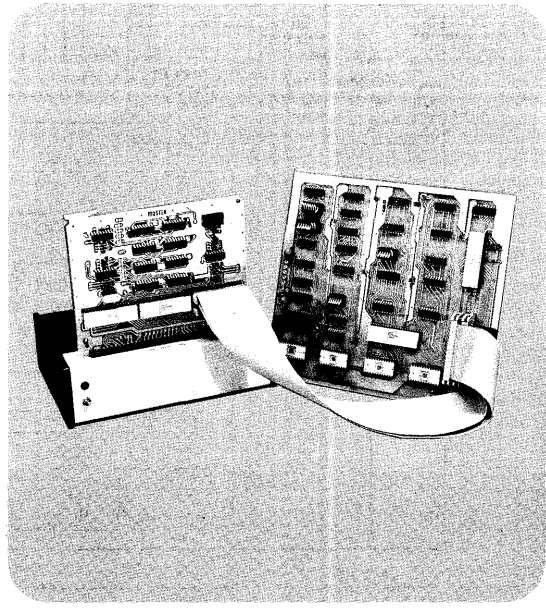
MOSTEK®

MICROPROCESSOR HARDWARE SUPPORT

F8 PSU Emulator (EMU–51)

FEATURES

- ☐ Completely emulates the MK 3851 Program Storage Unit (PSU)
- ☐ Utilizes either MK 3702/1702A or 2708 PROMS
- ☐ 2MHz operation
- ☐ 40-pin adapter cable for simple fast interconnect



The F8 PSU Emulator is a development aid for designing and field testing F8 microprocessor systems which utilize one or more MK 3851 Program Storage Units (PSU). The Emulator is electrically equivalent to the PSU but is field programmable instead of mask programmable. This enables a user to obtain final hardware verification of all PSU programming prior to ordering custom PSUs. Also, since the Emulator "plugs in" like a PSU (via a male, 40 pin connector on the end of an "umbilical cord"), prototype systems can be converted to final production status by simply unplugging the Emulator(s) and plugging in the corresponding custom PSU(s).

The MK 3851 is a 40-pin integrated circuit that provides 1K bytes of ROM, two 8-bit latched I/O ports, a software programmable timer, and interrupt circuitry for vectored addressing and priority control. Multiple MK 3851 PSU chips can be used in a single system.

USING THE EMULATOR

The Emulator performs all the functions of the PSU:

ROM
INPUT/OUTPUT PORTS
INTERRUPT VECTOR
TIMER

The ROM section of the Emulator uses either four 256 x 8 bit ultraviolet erasable PROMs or a single 1K x 8 bit ultraviolet erasable PROM to provide non-volatile storage of the users' program. The PROM(s) should be programmed using a PROM programmer and then installed on the Emulator. The six ROM address select switches can then be used to establish the location of the PROM in the system memory map.

The input/output ports, interrupt vector, and timer functions of the Emulator, are implemented using an MK 3851/12001, which is provided on the Emulator Board.

SPECIFICATIONS

Operating Temperature Range . . . 10°C to 40°C

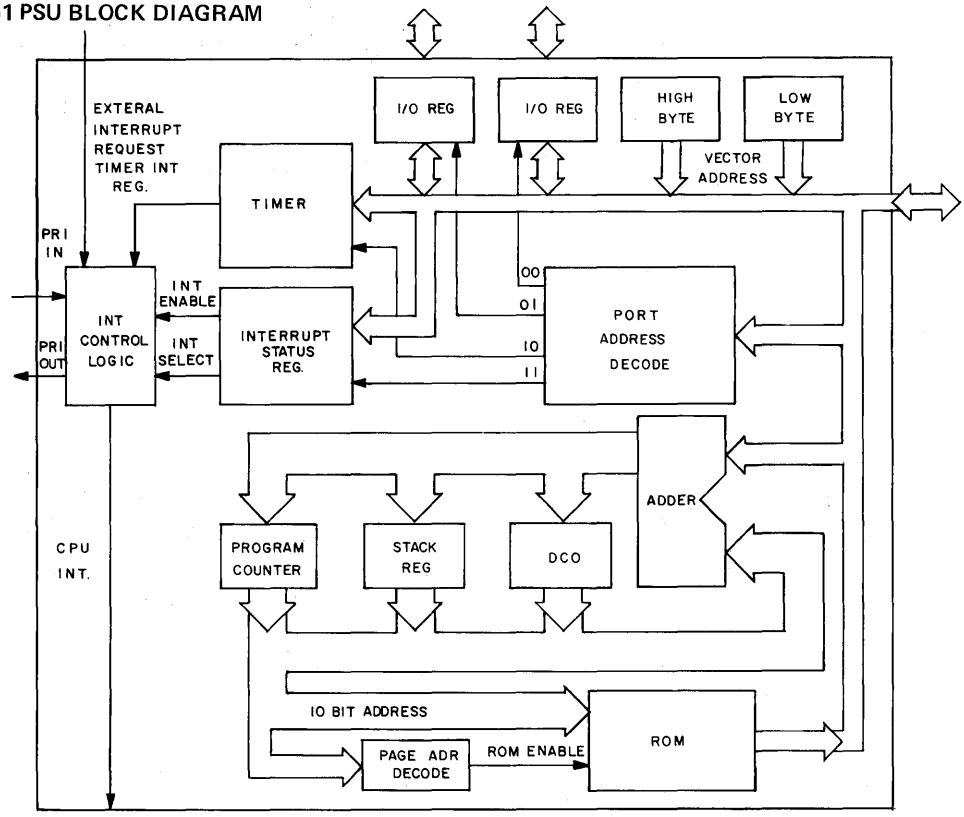
Power Supply Requirements (max.)

with 4, MK 3702s	with 1, 2708
+12V ± 5% @ 75mA	+12V ±5% @ 75mA
+5V ± 5% @ 500mA	+5V ±5% @ 350mA
–12V ± 5% @ 200mA	–12V ±5% – 100mA

Board Size . . . 8.2 in. x 9.19 in. x 1.0 in.

Connectors/Cables: (supplied with board)
- 5-Pin Power Connector
- 40-Pin Ribbon Cable (18 in. long)

MK 3851 PSU BLOCK DIAGRAM



EMULATOR BLOCK DIAGRAM DESCRIPTION

The Emulator block diagram shows how the PSU Emulator functions. Six ROM page select switches allow the user to place the 1K x 8 PROM memory at the desired location in the system memory map. Selection logic compares the most significant six bits of memory address from the Static Memory Interface (SMI) circuit with the six ROM page select switches, and causes the control logic to enable the PROM data output driver when the address is within range. Communication between the SMI and the CPU takes place on the Data Bus and the ROM Control Bus in the conventional manner. Note that either four 256 x 8 bit PROMS or a single 1024 x 8 bit PROM may be used to implement the PROM memory.

An MK 3851 PSU is used to provide timing and I/O port interface. The Data Bus is not connected directly to this PSU. Instead, port address translator logic modifies the contents of the Data Bus to allow the user to select the I/O port address desired.

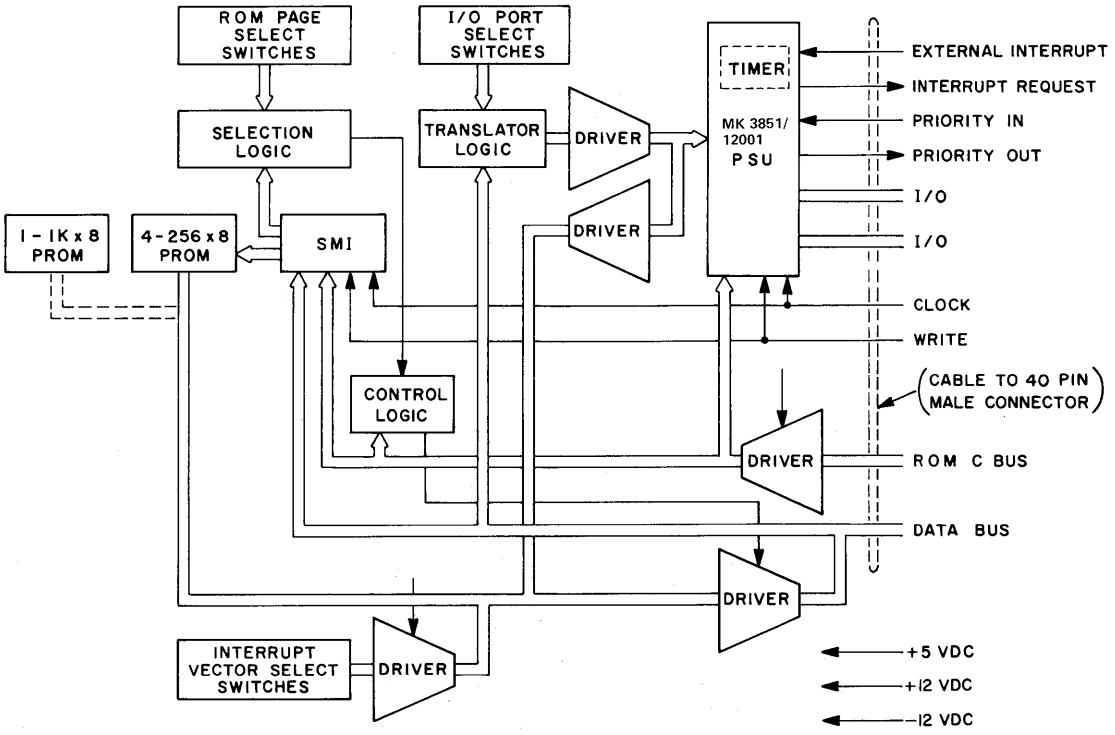
The I/O port addresses are determined by the position of the six I/O port select switches on the Emulator. The external interrupt line and the interrupt request line of the PSU on the Emulator provide interrupt control that allows the Emulator to perform in the system exactly as a production PSU.

The Interrupt Vector address on the Emulator is determined by the fifteen interrupt vector select switches, allowing the user to simulate the mask programmable vector address on the PSU. Since a PSU is used in the Emulator to provide the interrupt control logic, the interrupt control port status can still be modified normally.

The timer contained in the on board PSU circuit provides the timer function for the Emulator.

After the PROMs have been programmed and the switches have been correctly positioned the Emulator

EMULATOR BLOCK DIAGRAM



may be connected to the user's system by simply plugging the 40-pin connector into the corresponding PSU socket in the production/prototype circuit board.

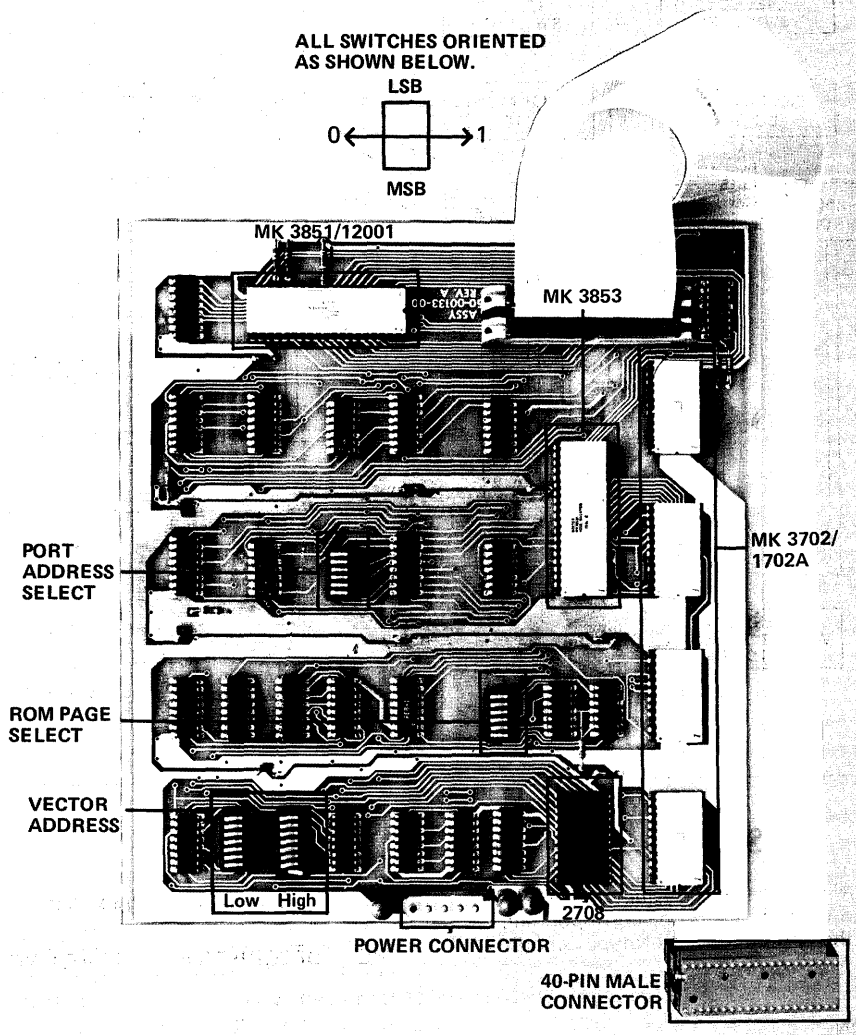
DOCUMENTATION

A complete set of documentation is provided with the Emulator to describe both the internal operation of the circuit board and the techniques for using it in system development. Also included are detailed instructions for ordering MK 3851 PSU's directly from the already verified data contained in the corresponding Emulator (i.e. the contents of the UV PROMs and the various switch positions).

ORDER INFORMATION

NAME	DESCRIPTION	PART NO.	PRICE*
EMU-51	PROM Emulator for the MK 3851. Includes power cable and 40-pin interface cable. PROMs not included.	MK 79018	435.00

*Prices are subject to change without notice and apply only in U.S. and Canada.


ALL SWITCHES ORIENTED
AS SHOWN BELOW.
LSB
0
1
MSB
MK 3851/12001
MK 3853
MK 3702/
1702A
PORT
ADDRESS
SELECT
ROM PAGE
SELECT
VECTOR
ADDRESS
Low High
270
POWER CONNECTOR
40-PIN MALE
CONNECTOR

MOSTEK®

MICROCOMPUTER HARDWARE SUPPORT

MK3870 Emulator (EMU-70)

November 1977

FEATURES

- ☐ Completely emulates the MK3870 single chip F8
- ☐ Utilizes MK2708 PROMs
- ☐ Connects directly to user's MK3870 socket
- ☐ Provides exact program verification

The MK3870 Emulator (EMU-70) is a development aid for designing and field testing F8 microprocessor systems which utilize the MK3870 single-chip F8.

The Emulator is electrically equivalent to the MK 3870 but is field programmable instead of mask programmable. This enables a user to obtain final software verification prior to ordering an MK3870. Also, since the Emulator "plugs in" like an MK3870 (via a male, 40-pin connector or a 40-conductor cable), prototype systems can be converted to final production status by simply unplugging the Emulator and plugging in the corresponding custom MK3870.

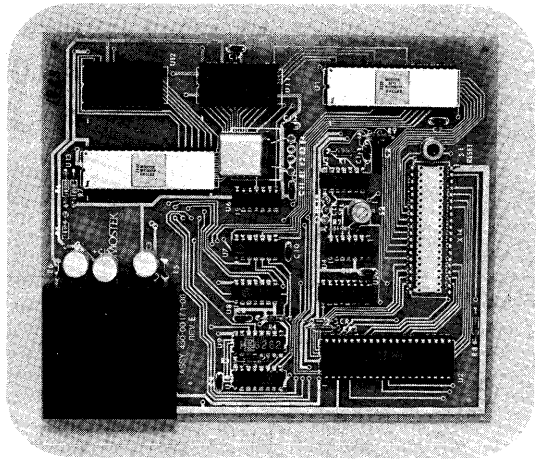
The MK3870 is a 5 volt only, 40-pin integrated circuit that provides 2K bytes of ROM, 64 bytes of RAM, four 8-bit latched I/O ports, a software programmable timer, and interrupt control circuitry.

EMU-70 DESCRIPTION

The Emulator performs all the functions of the MK3870:

CPU
ROM/RAM
INPUT/OUTPUT PORTS
VECTORED INTERRUPT
TIMER

The CPU functions, plus two I/O ports and scratch-pad RAM, are implemented using an MK3850 on the Emulator board.

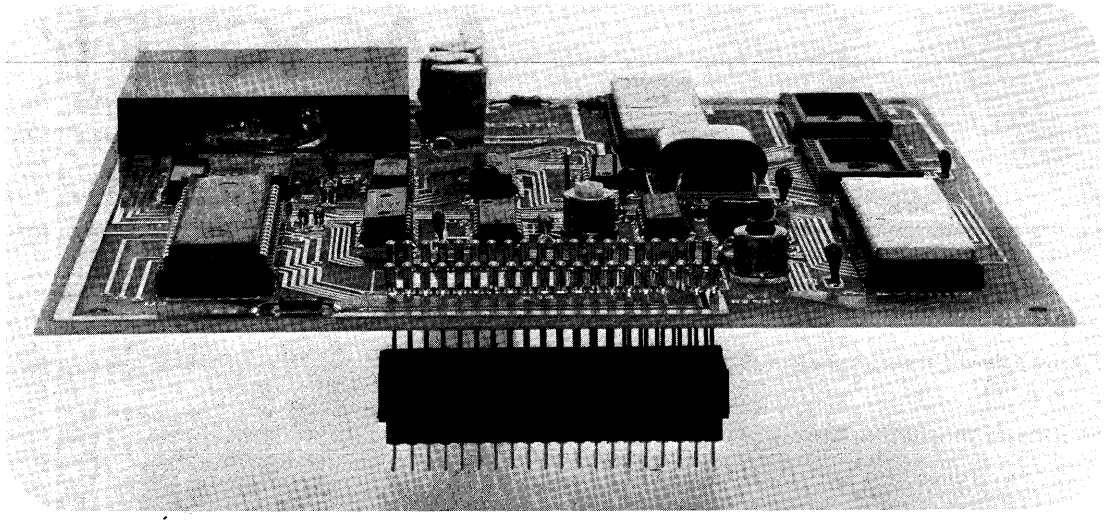


The ROM, Data Counter and Program Counter functions are implemented with an MK3853 SMI and two 1K x 8-bit UV Erasable PROMs to provide non-volatile storage of the user's program. The PROMs are programmed using a PROM programmer and then installed on the Emulator board.

Two I/O ports, interrupt and timer logic are implemented using an MK3871/90071 on the Emulator Board. The Emulator may be converted from standard TTL I/O ports to either open drain or direct drive I/O ports by ordering the appropriate PIO listed in the order information.

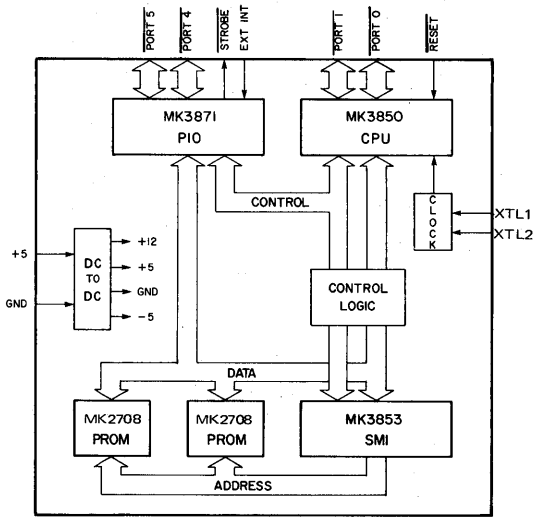
DOCUMENTATION

A complete set of documentation is provided with the Emulator to describe both the internal operation of the circuit board and the techniques for using it in system development. Also included are detailed instructions for ordering the MK3870 directly from the verified data contained in the Emulator (i.e. the contents of the UV PROMs).



EMU-70 showing 40-pin connector implementation.

FUNCTIONAL DIAGRAM



*All prices subject to change without notice, and apply only within the U.S. and Canada.

† Contact the factory for current pricing.

SPECIFICATIONS

Operating Temperature Range . . . . . .0°C to 50°C

Power Supply Requirements (max.)

$+5V_{DC} \pm 5\%$ @ 1.5A

Board Size . . . 6.0" x 7.0"

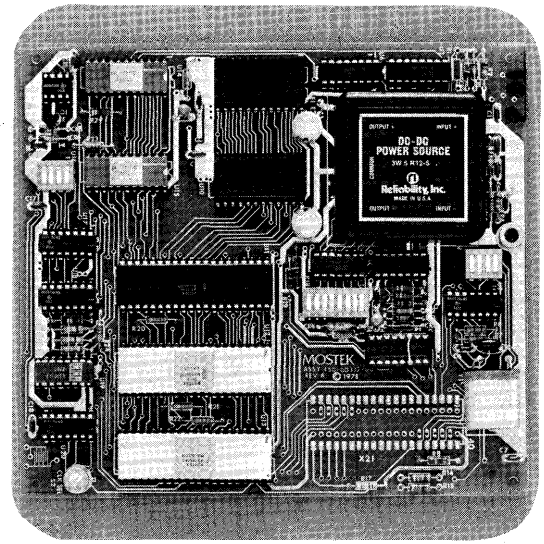
ORDER INFORMATION

NAME	DESCRIPTION	PART NO.	PRICE*
EMU-70 Operations Manual	Contains a detailed technical description with schematic diagrams.	MK79550	$1.50
EMU-70 (Less PROMs)	Circuit Board with documentation. Less PROMs.	MK79030	$200.
EMU-70 (With PROMs)	Circuit Board with documentation. Includes 2-MK2708 PROMs.	MK79032	†
XAID-706	Auxiliary 2ft interface cable for 'non-rigid' connection to the target system.	MK79050	$ 50.
Peripheral Input/Output	Direct Drive PIO	MK3871/ 90070	$12.35
Peripheral Input/Output	Open drain PIO	MK3871/ 90072	$12.35

3870 Series Microcomputer Emulator (EMU-72)

FEATURES

- ☐ Completely emulates 3870 Series single chip Microcomputers (MK3870, MK3872, and MK3876)
- ☐ Utilizes MK2716 PROMs
- ☐ Connects directly to user's 3870 Series socket
- ☐ Provides exact program verification
- ☐ The 3870 Series Microcomputer Emulator (EMU-72) is a development aid for designing and field testing microcomputer systems which utilize 3870 Series Microcomputers. The Emulator is electrically equivalent to a 3870 Series Microcomputer (3870, 3872, or 3876) but is field programmable instead of mask programmable. This enables a user to obtain final software verification prior to ordering the mask programmable 3870 Series Microcomputer. Also, since the Emulator "plugs into" a 3870 socket (via a male, 40-pin connector or a 40-conductor cable), prototype systems can be converted to final production status by simply unplugging the Emulator and plugging in the corresponding 3870 Series Chip.



DESCRIPTION

The Emulator performs all the functions of the 3870 Series Microcomputers:

CPU
ROM/RAM
INPUT/OUTPUT PORTS
VECTORED INTERRUPT
TIMER

The CPU functions, plus two I/O ports and scratchpad RAM, are implemented using an MK3850 on the Emulator board.

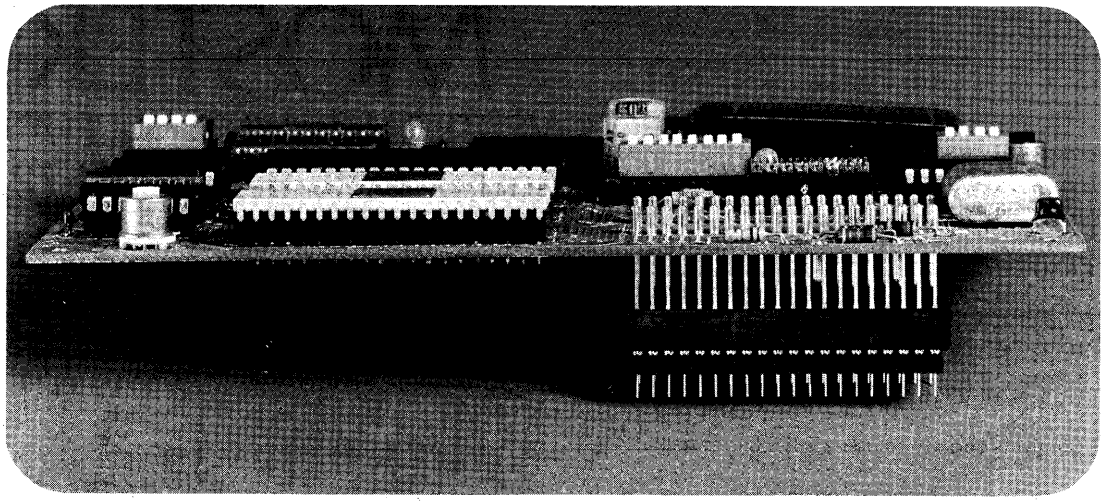
The ROM, Data Counter and Program Counter functions are implemented with an MK3853 SMI and two 2K x 8-bit UV Erasable PROMS to provide non-volatile storage of the user's program. The PROMs are programmed using a PROM programmer and then installed on the Emulator board.

The executable RAM of the MK3872 and MK3876 is emulated with two 5101 Static CMOS RAMs. Standby current and battery trickle charge current at the VSB terminal have been adjusted to emulate those functions on the MK3872 and MK3876.

Two I/O ports, interrupt logic, and timer logic are implemented using an MK3871 on the Emulator Board. The Emulator may be converted from standard TTL I/O ports to either open drain or direct drive I/O ports by ordering the appropriate PIO listed in the ordering information.

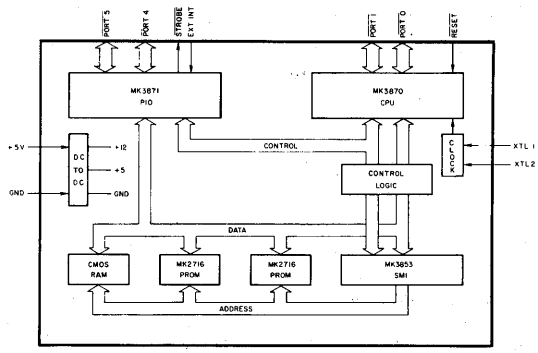
DOCUMENTATION

A complete set of documentation is provided with the Emulator to describe both the internal operation of the circuit board and the techniques for using it in system development.



EMU-72 showing 40-pin connector implementation

FUNCTIONAL DIAGRAM



* All prices subject to change without notice, and apply only within the U.S. and Canada.

SPECIFICATIONS

Operating Temperature Range. . . . . 0°C to 50°C
Power Supply Requirements. . .+5V ±5%@ 1.2A max. (700mA typ.)
Board Size. . . . . 6.25 x 7.0 in.

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NUMBER	PRICE*
EMU-72	3870 Series Microcomputer with Operations Manual. Includes a 40 pin adapter plug to interface to the user's 40 pin 3870 socket. Does not include 2716 PROMs.	MK79078	$375.00
EMU-72 Operations Manual	Contains a complete technical description of the operation and use of the EMU-72 Schematic diagram included.	MK79581	$ 5.00
XAID-706	Auxilary 2 foot interface cable for 'non-rigid' connection to the target system	MK79050	$ 50.00
Peripheral Input/Output	Direct Drive PIO	MK3871N/90070 MK3871P/90070	$ 12.35 $ 17.95
Peripheral Input/Output	Open Drain PIO	MK3871N/90072 MK3871P/90072	$ 12.35 $ 17.95

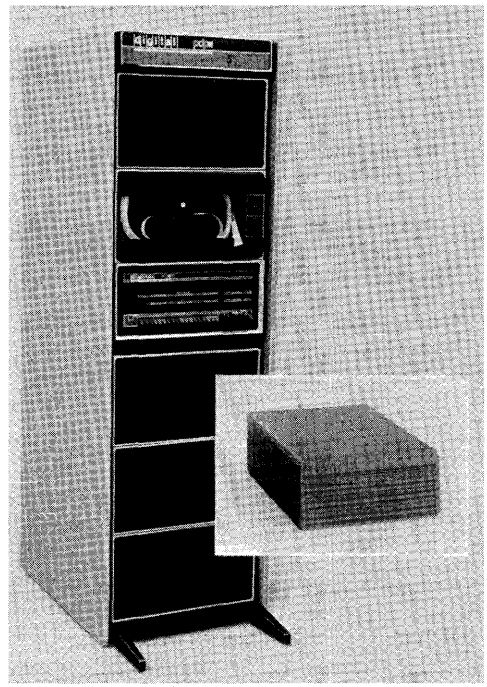
MOSTEK®

3870/F8 MICROCOMPUTER SOFTWARE SUPPORT

Fortran IV Cross Assembler (XFOR-50/70)

FEATURES

- ☐ ANSI-Fortran IV Source
- ☐ Executes on 16 bit word length machine
- ☐ Cross Assembler is machine independent for:
  - Character representation (ASCII or BCD)
  - Numerical representation (1's or 2's complement)
- ☐ I/O logical device assignments are user definable
- ☐ 2 pass assembly easily accomodated if no secondary storage available
- ☐ Memory required: 13K words (typical)
- ☐ Assembler directives
  - TITLE 'Set page title'
  - EJECT 'Page'
  - EQU 'Values'
  - ORG 'Beginning address'
  - PUNCH 'Create load tape F8 loader format'
  - PRINT 'Off and On enable for output listing
  - DC 'Define constants'
  - END



DESCRIPTION

The MOSTEK 3870/F8 Cross Assembler XFOR-50/70 is written in ANSI FORTRAN IV. It may be compiled and executed on any computer system which has at least a 16 bit word length for integer storage and 13K of memory for program storage. The Cross Assembler is independent of machine character representation (ASCII, BCD, etc.) and numerical representation (2's complement, 1's complement, etc.) Logical device assignments are set up in the source of the main program module, and may be easily changed to suit the installation. Also, if no secondary storage is available the main program may be changed to accommodate re-reading of the user input for the second pass of the assembly. Output is in F8 loader format.

ORDERING INFORMATION

The XFOR-50/70 is available directly from Mostek by filling out a copy of the Software Licensing Agreement printed on the back of this data sheet and returning it with the appropriate payment or Customer Purchase Order to:

MOSTEK CORPORATION
Microcomputer Systems Dept.
1215 West Crosby Road
Carrollton, Texas 75006

DESIGNATOR	DESCRIPTION	PART NO.	PRICE
XFOR-50/70	3870/F8 Cross Assembler written in ANSI Fortran IV is supplied as a source card deck with Operations Manual.	MK79012	$100.00

STANDARD SOFTWARE LICENSE AGREEMENT

All Mostek Corporation products are sold on condition that the Purchaser agrees to the following terms:

1. The Purchaser agrees not to sell, provide, give away, or otherwise make available to any unauthorized persons, all or any part of, the Mostek software products listed below; including, but not restricted to: object code, source code and program listings.

2. The Purchaser may at any time demonstrate the normal operation of the Mostek software product to any person.

3. All software designed, developed and generated independently of, and not based on, Mostek's software by purchaser shall become the sole property of purchaser and shall be excluded from the provisions of this Agreement. Mostek's software which is modified with the written permission of Mostek and which is modified to such an extent that Mostek agrees that it is not recognizable as Mostek's software shall become the sole property of purchaser.

4. Purchaser shall be notified by Mostek of all updates and modifications made by Mostek for a one-year period after purchase of said Mostek software product. Updated and/or modified software and manuals will be supplied at the current cataloged prices.

5. In no event will Mostek be held liable for any loss, expense or damage, of any kind whatsoever, direct or indirect, regardless of whether such arises out of the law of torts or contracts, or Mostek's negligence, including incidental damages, consequential damages and lost profits, arising out of or connected in any manner with any of Mostek's software products described below.

6. MOSTEK MAKES NO WARRANTIES OF ANY KIND, WHETHER STATUTORY, WRITTEN, ORAL, EXPRESSED OR IMPLIED (INCLUDING WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE AND MERCHANTABILITY AND WARRANTIES ARISING FROM COURSE OF DEALING OR USAGE OF TRADE) WITH RESPECT TO THE SOFTWARE DESCRIBED BELOW.

The Following Software Products Subject To This Agreement:

Order Number Description Price*

_____

_____

Ship To: _____ Bill To: _____

_____

Method of Shipment: _____ Customer P.O. Number: _____

Agreed To:

PURCHASER MOSTEK CORPORATION

By: _____ By: _____
Title: _____ Title: _____
Date: _____ Date: _____

*Prices Subject to change Without Notice

MOSTEK®

Z80 MICROCOMPUTER SYSTEMS

Microcomputer Development System (AID-80F)

INTRODUCTION

The Mostek AID-80F* is a complete state of the art disk based computer. Not only does it provide all the necessary tools for software development but it provides complete hardware/software debug through Mostek's AIM* series of in-circuit emulation cards for the Z80 as well as the 3870 family of single chip microcomputers. The AID-80F has at its heart the powerful OEM-80 (Single Board Computer), RAM-80 (RAM I/O add on board), and the FLP-80 (floppy controller board). Because these boards and software are available separately to OEM users, the AID-80F serves as an excellent test bed for developing systems applications.

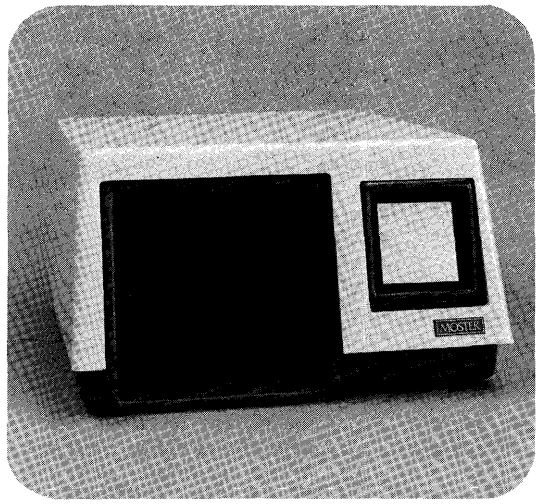
The disk based system eliminates the need for other mass storage media but provides ease of interface to any peripheral normally used with computers. The file based structure for storage and retrieval consolidates the data base and provides a reliable, portable media to speed and facilitate software development.

The FLP-80DOS Disk operating system is designed for maximum flexibility both in use and expansion to meet a multitude of end user or OEM needs.

Development System Features

The AID-80F is an excellent integration of both hardware and software development tools for use throughout the complete system design and development phase. The software development is begun by using the combination of Mostek's Text Editor with "roll in - roll out" virtual memory operation and the Mostek relocating assembler. Debug can then proceed inside the AID-80F domain using its resources as if they were in the final system. Using combinations of the Monitor, Designer's Debugging Tool, execution time breakpoints, and single step/multistep operation along with a formatted memory dump provides control for attacking those tough problems. The use of the Mostek AIM-80 option provides extended debug with versatile hardware breakpoints on memory or port locations, a buffered in-circuit emulation cable for extending the software debug into its own natural hardware environment, as well as 256x32 history memory to capture bus transactions in real time for later examination.

*Trademark of Mostek Corporation

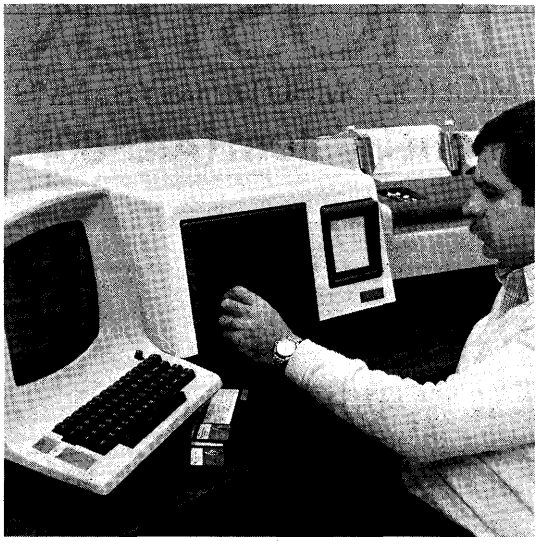


The relocatable and linking feature of the assembler enables the use of contemporary modular design techniques whereby major system alterations can be made in small tractable modules. Using either the Relocatabling Linking Loader or the Linker, the small modules can be combined to form a run time module without major reassembly of the entire program.

Packaged System Features

From a system standpoint, the AID-80F has been designed to be the basis of an end product small business/industrial computer. The flexibility provided in the FLP-80DOS operating system permits application programs to be as diverse as a high level language compiler to a supervisory control system in the industrial environment. Other hardware options are available, with even more to be added. Expansion of the disk drive units to a total of four single sided or double sided units provide up to two megabytes of storage. This computer uses the third generation Z80 processor supported with the power of a complete family of peripheral chips. Through use of its 158 instructions, including: 16-bit arithmetic, bit manipulation, advanced block moves and interrupt handling, almost any application from communication concentrators to general purpose accounting systems is made easy.

AID-80F AS A DEVELOPMENT SYSTEM



AID-80F AS A PACKAGED OEM SYSTEM

OEM Features

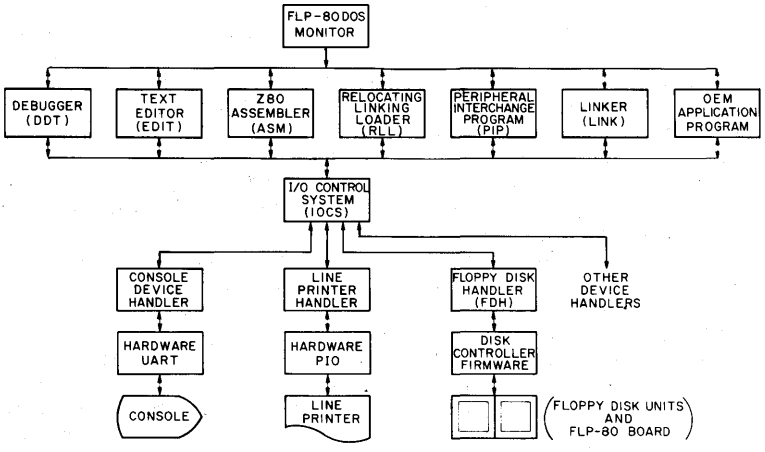
The hardware and software basis for the AID-80F is also available separately to the OEM purchaser. Through a software licensing agreement, all Mostek Software can be utilized on these OEM series of cards. A growing line of support cards and card cages, permits the user to configure a multitude of different systems.

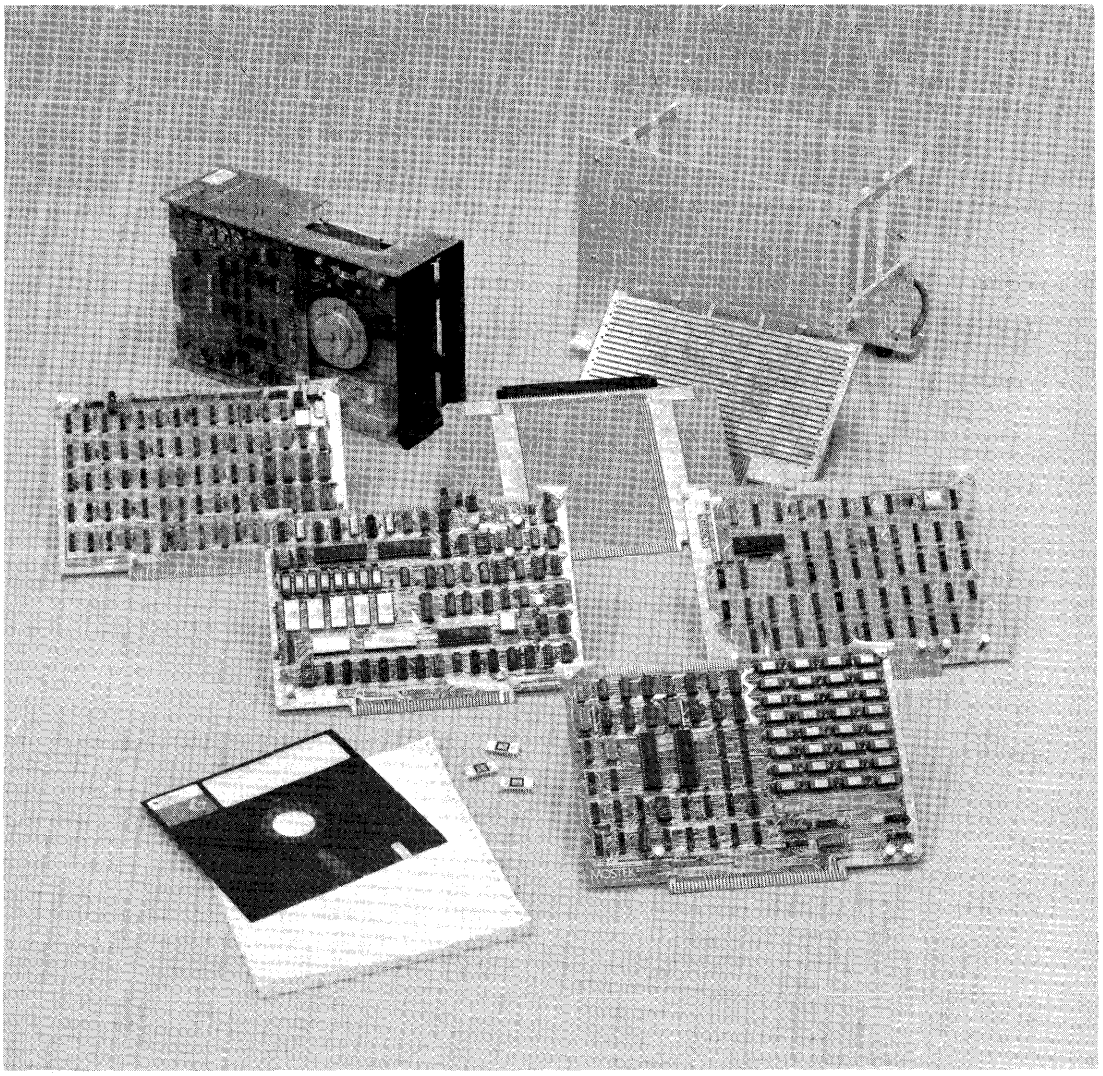
AID-80F RESIDENT SOFTWARE (FLP-80DOS)

A totally integrated package of resident software is offered in conjunction with the AID-80F consisting of:

Monitor
DDT-80 with extended debug through AIM-80
Text Editor
Z80 Relocating Assembler
Relocating Linking Loader
Peripheral Interchange Program
Linker
I/O Control System
Floppy Disk Handler
Device Driver Library
Batch Mode Operation

SOFTWARE BLOCK DIAGRAM





Monitor

The FLP-80DOS Monitor is the environment from which all activity in the system initiates. From the Monitor, any system routine such as PIP or a user generated problem is begun by simply entering the program name. FLP-80DOS I/O is done in terms of logical unit numbers, as is commonly done in FOR-TRAN. A set of logical units are preassigned to default I/O drivers upon power up or reset. From the console the user can reassign any logical unit to any new I/O device and can also display logical unit assignments. Executable file creation can be done by the Save command as well as printable absolute object files can be produced using the Dump command. Assembler generated absolute/relocatable files may be loaded using the Load command. For the relocatable feature, both load status (addresses, unresolved references) as well as global symbol table are available to any list device.

Text Editor

The Text Editor premits editing/creating of any source file independent of the language being written. The Editor is both line and string oriented to give maximum utility and user flexibility. The Editor through its virtual memory "roll in - roll out" techni-

que can edit a file whose length is limited only by maximum diskette storage. Included in the repertoire of 13 commands are macro commands to save time when encountering redundant editing tasks. The Editor is also capable of performing in one operation all the commands which will fit into an 80 column command buffer.

Summary of Editor Commands

Advance N	-Advance line pointer N line
Backup N	-backs up N lines
Change N /S_1/S_2	-change N occurrences of string 1 to string 2
Delete N	-Delete current line plus next N-1 lines of text
Exchange N	-Exchanges current line plus next N-1 lines with lines to be inserted while in insert mode.
Insert	-place Editor in insert mode. Text will be inserted after present line.
Line N	-Place line pointer on Line N.
MACRO 1 or MACRO 2	-Defines Macro 1 or Macro 2 by the following string of Text Editor commands.
Quit	-Stores off file under editing process and returns to Monitor environment.
Search N/S_1	-Search from existing pointer location until nth occurrence of string S1 is located and print it.
Top	-inserts records at top of file before first line.
Verify N	-Print current record to console plus next N-1 records while advancing pointer N records ahead.
Window N	-Prints current record plus next N-1 records to source output device while advancing pointer N records.
eXecute N	-Executes Macro 1 or Macro 2 as defined by Macro command.

Z80 Assembler

The Z80 Resident Assembler generates relocatable or absolute object code from source files independent of source medium. The assembler recognizes all 158 Z80 instructions as well as 20 powerful pseudo operators. The object code generated is industry standard absolute or relocatable format. With the relocating feature, large programs can easily be developed in smaller sections and linked using the System Relocating Linking Loader or Linker. Because the assembler utilizes the I/O Control System, object modules or list modules can be directed to disk files, paper tape, console, or line printer. Portability of output media eliminates the requirement for a complete set of peripherals at every software/hardware development system. The assembler run time options include sorted symbol table generation, no list, no object, pass 2 only, quit, cross reference table, and reset symbol table. The assembler is capable of handling 14 expression operators including logical, shift, multiplication, division, addition and subtraction operations. These permit complex expressions to be resolved at assembly time by the assembler rather than manually by the programmer. Comments can be placed anywhere but must be preceded by a semicolon. Error messages are integrated with listing file but can be directed to console device. In addition to the standard assembler psuedo operators are:

GLOBAL	- For global definition.
PSECT operator	- to generate relocatable or absolute modules
IF expression	- conditional assembly IF expression is true
INCLUDE dataset	- to include other datasets (files) as in-line code anywhere in source file.

Peripheral Interchange Program

PIP provides complete file maintenance activity for operations such as copy file from disk to disk, disk to peripheral, or any peripheral to any other peripheral supporting both file structured and character oriented devices. Key operations such as renaming, appending, and erasing files also exist along with status commands for diskette ID and vital statistics. PIP can search the diskette directories for any file or a file of a specific name, extension, and user number. The PIP operations are:

Append	-appends file 1 to file 2 without changing file 1.
Copy	-copies input files or data from an input device to an output file or device. The Copy command can be used for a variety of purposes such as listing files, concatenating individual files, or copying all the files or a single file from one disk unit (e.g. DK0) to a second disk unit (e.g. DK1).
Directory	-lists the directory of a specified disk unit (DK0, DK1 and etc.). The file name, extension, and use-number is listed for each file in the directory. The user can also request listing only files of a specified name, only files of a specified extension or only files

of a specified user number. The list device can be any device supported by the system as well as a file.

Erase -erases a single file or files from a diskette in a specified disk unit. The user has the option to erase all files, only files of a specified file name, only files of a specified extension or only files of a specified user number.

Format -takes completely unformatted soft sectored diskettes, formats to IBM 3740, and prepares to be system diskette. Operation is performed on diskette unit 1 and a unique 11 character name is assigned to that diskette.

Init -initializes maps in disk handler when a new diskette has been changed while in the PIP environment.

Rename -renames a file, its extension, and its user number to a file of name X, extension Y, and user Z.

Status -lists all vital statistics of a disk unit to any list device. These include number of allocated records, number of used records, and number of bad records.

Quit -returns to Monitor Environment.

DOS/Disk Handler

The heart of the FLP-80DOS software package is the Disk Operating System. Capable of supporting 4 double sided units, the system provides a file structured orientation timed and optimized for rapid storage and retrieval. Thorough error reporting exists from the DOS to enable an application programmer to quickly debug his program as well as extensive error recovery and bad sector allocation which insures data and file integrity. The DOS not only provides file reading and writing capability but special pointer manipulation, record deletions, record insertions, skip records both forward and backward as well as directory manipulation such as file creation, renaming, and erasure. The DOS is initiated by a calling vector which is a subset of the I/O control system vector or through the standard IOCS calling sequence to elect buffer allocation, blocking, and deblocking of data to a user selectable logical record type.

A unique dynamic allocation algorithm makes optimal use of disk storage space. Run time (Binary Files) are given first priority to large blocks of free space to eliminate any such overhead in operating system and overlay programs. The algorithm marks storage fragments as low priority and uses them only when diskette is nearing maximum capacity.

The DOS permits 7 files to be opened for operations at any one time, thus permitting complex application programs as well as multi-user operation of the DOS.

I/O Control System

The I/O Control System provides a central facility from which all calls to I/O can be structured. This permits a system applications program to dissolve any device dependence by utilizing the logical unit approach of large main frame computers. For example, a programmer may want to structure his utility to use logical unit No. 5 as his list device which normally in his system defaults to the line printer. He may, however, assign at run time a different device for logical unit No. 5. His application program remains unchanged.

Interface by a user to IOCS is done simply be entering a device mnemonic in a table and observing the calling sequence format. IOCS supplies a physical buffer of desired length, handles buffer allocation, blocking, deblocking, and provides a logical record structure as specified by the user.

DDT

The Designer's Debugging Tool consists of commands for facilitating an otherwise difficult debugging process. The AID-80F's rapid source changes through the editor and re-assemblies, followed by DDT operations close the loop on the debug cycle. The DDT commands include:

Memory -display, update, or tabulate memory

Port -display, update or tabulate I/O ports

Execute -execute user's program

Hexadecimal-performs 16 bit add/sub

Copy -copy one block to another

Breakpoint -set software trap in user code for interrupting execution in order to examine CPU registers

Register -display contents of user's registers

Offset -enter address adder for debug of relocatable modules

Fill -fill specified portion of memory with 8 bit byte

Verify -compare two blocks of memory

Walk -software single step/multistep

Quit -return to Monitor

Linker

The Linker program provides the capability of linking assembler generated absolute or relocatable object modules together to create a binary or run time file. This process is carried on independently of the Relocating Linking Loader to permit generation of programs which may require the total memory resources of the system. The linking process includes the library search option that if elected, will link in standard library object files (device drivers, math pack functions, IOCS features) on disk to resolve undefined global symbols. By selecting an option, a complete cross reference table will be generated and stored in a separate file, a list of undefined global symbols will be printed, and/or the global symbol table will be generated and stored in the same file as the cross reference symbol table.

Batch Mode Operation

In Batch Mode Operation, a command file is built on disk or assigned to a peripheral input device such as a card reader. The console input normally taken from the keyboard is taken from this batch device or batch file. While operating under direction from a batch file, the console output prompts the user as normal or the prompting can be directed to any other output device. The Batch file definable operation is especially useful to execute redundant procedures not requiring constant attention of the operator and to allow several programmers to use one system.

HARDWARE DESCRIPTION

OEM-80

The OEM-80, also available as a complete single board development system (SDB-80), provides the essential power of the system. While using the Z80 as the central processing unit, the OEM-80 is provided with other Z80 family peripheral chip support. Two Z80 PIO's give the system 4 completely programmable 8 bit parallel I/O ports with handshake from which the standard system peripherals are interfaced. Also, in the system is the Z80-CTC counter time circuit which has 3 free flexible channels to perform critical counting end event counter timing functions. Along with 16K of RAM, the OEM-80 provides 5 ROM/PROM sockets which can be utilized for 10/20K of ROM or 5/10K PROM. Three sockets contain the firmware portion of FLP-80DOS. The remaining sockets can be strapped for other ROM/PROM elements. The OEM-80 is particularly flexible for system expansion. Expansion of memory, (ROM, PROM, or RAM) is made easy by off board select logic or by the on board strapping flexibility.

RAM-80B

The RAM 80B adds additional memory with Mostek's MK4116 16K dynamic memory along with more I/O. These two fully programmable 8 bit I/O ports with handshake provide additional I/O expansion as system RAM memory needs grow.

FLP-80

Integral to the AID-80F system is the floppy controller. The FLP-80 is a complete IBM 3740 single density/double sided controller for up to 4 drives. The controller has 128 bytes of FIFO buffer resulting in a completely interruptable disk system.

AIM-80

The AIM-80 module provides extended debug for the AID-80F. In Z80 development, real time in-circuit emulation permits debug of the hardware and the software at the most intimate level. Hardware single step/multistep with register trace, execution intercept on memory access, port access, or external trigger provides the absolute control over any system no matter how complex. The "pushbutton intercept" enables the programmer to perform a controlled recovery for those extremely difficult to trace processor lock out loops. With the memory clock selectable history module, any past 256 events of data, address, or control bus operation are captured in real time and displayable.

The AIM-80 includes 8K bytes of ROM firmware introducing unique software including a mnemonic disassembler for inverse assembly of history module contents or single step/multistep operations. "In line" code disassembled to language mnemonics provides insight into execution results as if examining an assembler generated listing. Extra added capability is the ROM resident self test of OEM-80 or target RAM.

AIM-72

The AIM-72 module provides debug and in-circuit emulation capabilities for the 3870 series microcomputers (3870, 3872, 3873, and 3876) on the AID-80F. Multiple breakpoint capability and single step operation allows the designer complete control over the execution of the 3870 series microcomputer. Register and Port display and modification capability provides information needed to find system "bugs". All I/O is in the user's system connected to AIM-72 by a 40-pin interface cable. Program storage on the AIM-72 is in RAM so that the results of disk based Editing and Assembly can be quickly loaded to the AIM-72 memory for debug with the user's I/O devices.

Software supporting the AIM-72 in the AID-80F system is a F8/3870 Cross Assembler. This assembler produces either relocatable or absolute object code. All AID-80F editing and utility software is available to the user to speed the process of programming 3870 series of single chip microcomputers.

MECHANICAL SPECIFICATIONS

AID-80F Enclosure

Overall Dimensions - 20"w x 22"l x 12"h
Material - NORYL EN 185
Color Composition - White GE No. 8385; Blue GE No. 2283
Weight - 60 lbs.
Front Panel Dimensions - 3.75" x 3.75"
Read End Panel Dimensions - 4.25" x 4.62"
4.25" x 2.00"
Fan Capacity - 52 CFM
Card Cage Capacity-Six 8½"x12" (SDB) size boards
Card Connectors - 100 Pin 0.125" centers
Operating Temperature Range - +10° C to 35° C

Power Supply

Input - 115 VAC 60Hz
Outputs - +5VDC at 10 Amps Max.
−5VDC at 0.15 Amps Max.
+12 VDC at 3 Amps Max.
−12 VDC at 0.5 Amps Max.
+24 VDC at 3 Amps Max.

ORDERING INFORMATION

NAME	DESCRIPTION	PART NO.	PRICE
AID-80F	Complete enclosure with power supply, 2 Shugart single sided drives, 6 slot card cage, fan, and cabling including the following list of separately available items:	MK78125	$5,995.00
OEM-80	Z80 OEM Board with four 8 bit parallel I/O ports with handshake control, counter timer circuit, RS232 and TTY interface, 16K bytes dynamic memory, 5 PROM/ ROM sockets	MK78123	$995.00 (separately)
RAM-80B	RAM board with 16K bytes RAM and four 8 bit parallel I/O ports with handshake control.	MK78108	$945.00 (separately)
FLP-80	IBM-3740 Floppy controller for single density, single sided/double sided drives with 128 byte FIFO. Includes FLP-80DOS software.	MK78111	$2200.00 (separately)
FLP-80DOS	Complete software package* including relocatable Assembler, Text Editor, Peripheral Interchange Program, Monitor, Designer's Debugging Tool, Relocating Linking Loader, Linker, I/O Control System, Floppy Disk Handler, and other device drivers.		Included with MK78111
	OEM-80 Operations Manual	MK78544	$5.00
	RAM-80B Operations Manual	MK78545	$3.00
	FLP-80 Operations Manual	MK78560	$5.00
	FLP-80DOS Operations Manual	MK78557	$20.00
	Z80 Programming Manual	MK78515	$7.50

* The FLP-80DOS software package includes binary run time files of all system software described. By submitting a MOSTEK Standard Software Licensing Agreement, the source listings of the following programs are available: Relocatable Assembler, Text Editor, Peripheral Interchange Program, Monitor, Designer's Debugging Tool, Relocating Linking Loader, and Linker.

OPTIONAL HARDWARE FOR AID-80F

DESIGNATOR	DESCRIPTION	PART NO.	PRICE	AVAILABILITY
AIM-80	Extended Debug for Z80 with AIM-80X	MK78132	$1195.00	1st Qtr. 78
AIM-72	RAM based in-circuit emulation module for 3870 series of single chip micro-computers (to include DDT-70X software for AID-80F)	MK79076	$1295.00 (Budgetary)	3rd Qtr. 78
PPG-08	PROM programmer for MK2708	MK79033	$ 300.00	NOW
XAID-805	Cable for PPG-08	MK79041	$ 30.00	NOW
AID-103	Universal Wirewrap Card (SDB size 12" x 8.5")	MK79023	$ 100.00	NOW
XAID-104	Extender Card (SDB size 12" x 8.5")	MK79024	$ 100.00	NOW
FZCASM	Relocatable 3870/F8 cross assembler	———	———	2nd Qtr. 78

COMING

Fortran IV Resident Fortran Compiler

BASIC I Basic Interpreter for Business/Scientific Applications

BASIC II Basic Interpreter for Control applications, requires 6K, (Source & object)

PL/1S High Level Language - Subset of IBM PL/1

MOSTEK®

MICROCOMPUTER SYSTEMS

AID-80F Cross Assembler for 3870/F8 (FZCASM)

FEATURES

- ☐ Assembles all standard 3870/F8 family source statements
- ☐ Object output in industry standard hexadecimal format extended for relocatable and linkable programs
- ☐ Allows the following pseudo-ops:

ORG	- program origin
EQU	- equate label
DC	- define constant
DEFL	- define label
DEFM	- define message
DEFB	- define byte
DEFW	- define word
DEFS	- define storage
END	- end statement
IF	- conditional assembly
ENDIF	- end of conditional assembly
INCLUDE	- include another dataset within current assembly
NAME	- program name definition
PSECT	- program section definition
GLOBAL	- global symbol definition

- ☐ Supports the following assembler directive pseudo-ops:

EJECT	- eject a page of listing
TITLE	- place heading at top of each page of listing
LIST	- turn listing on
NLIST	- turn listing off

- ☐ Complete assembly in two passes with second pass repeatable
- ☐ Size of program to be assembled limited only by memory available for symbol table
- ☐ Supports conditional assembly, relocatable and linkable modules, symbol table and cross reference listings
- ☐ Supplied on a standard FLP-80DOS diskette for use with the MOSTEK AID-80F floppy disk based development system

DESCRIPTION

The purpose of the 3870/F8 Cross Assembler is to assemble source language programs for the MOSTEK 3870 Series and F8 microcomputers. The Cross Assembler is designed to run on the MOSTEK AID-80F Dual Disk Development System with the FLP-80DOS operating system. The Cross Assembler is supplied on flexible diskette. The Assembler reads F8 source mnemonics and pseudo-ops and outputs an assembly listing and object code. The assembly listing shows address, machine code, statement number, and source statement. The object code is in industry standard hexadecimal format modified for relocatable, linkable assemblies. A conversion utility (F8DUMP) is supplied to produce object code in F8 format for users of the MOSTEK SDB-50/70. The Assembler supports conditional assemblies, global symbols, relocatable programs, a printed symbol table and cross reference listing. It can assemble any length program limited only by a symbol table size of over 400 symbols. Expressions involving mathematical and logical operations are allowed. Conditional assembly allows the user to suspend assembly for a portion of the program depending upon the result of an expression. A

global symbol is catagorized as "internal" if it appears as a label in the program; otherwise it is an "external" symbol. The printed symbol table and cross reference listing show which symbols are internal and which are external. The Cross Assembler allows the user to select relocatable or non-relocatable assembly via the "PSECT" pseudo-op. Relocation records are placed in the object output for relocatable assemblies.

The Assembler can be run as a single pass assembler or as a learning tool. (In this mode, global symbols and forward references are not allowed).

In conjunction with the FLP-80DOS Text Editor and Linker, FZCASM provides the means for editing, assembling and linking F8 or 3870 family programs.

ORDERING INFORMATION

DESIGNATOR	DESCRIPTION	PART NUMBER	PRICE
FZCASM Cross Assembler	Includes the 3870/F8 Cross Assembler on a FLP-80 DOS system diskette, and the FZCASM Operations Manual.	MK79079	$400.00
FZCASM Operations Manual	Describes in detail the operation of the 3870/F8 Cross Assembler	MK78582	$10.00
AID-80F Data Sheet	Describes the MOSTEK AID-80F Dual Disk Development System	MK78568	NC
FLP-80DOS Data Sheet	Describes the operating system used on the AID-80F System	MK78556	NC
FLP-80DOS Operations Manual	Describes in detail the software and operating system used to run FZCASM on the AID-80F System	MK78557	$20.00
AIM-72 Data Sheet	Describes the MOSTEK AIM-72 Application Interface Module used to provide in-circuit emulation capability for 3870 series Microcomputers on the AID-80F.	MK79576	NC

STANDARD SOFTWARE LICENSE AGREEMENT

All Mostek Corporation products are sold on condition that the Purchaser agrees to the following terms:

1. The Purchaser agrees not to sell, provide, give away, or otherwise make available to any unauthorized persons, all or any part of, the Mostek software products listed below; including, but not restricted to: object code, source code and program listings.

2. The Purchaser may at any time demonstrate the normal operation of the Mostek software product to any person.

3. All software designed, developed and generated independently of, and not based on, Mostek's software by purchaser shall become the sole property of purchaser and shall be excluded from the provisions of this Agreement. Mostek's software which is modified with the written permission of Mostek and which is modified to such an extent that Mostek agrees that it is not recognizable as Mostek's software shall become the sole property of purchaser.

4. Purchaser shall be notified by Mostek of all updates and modifications made by Mostek for a one-year period after purchase of said Mostek software product. Updated and/or modified software and manuals will be supplied at the current cataloged prices.

5. In no event will Mostek be held liable for any loss, expense or damage, of any kind whatsoever, direct or indirect, regardless of whether such arises out of the law of torts or contracts, or Mostek's negligence, including incidental damages, consequential damages and lost profits, arising out of or connected in any manner with any of Mostek's software products described below.

6. MOSTEK MAKES NO WARRANTIES OF ANY KIND, WHETHER STATUTORY, WRITTEN, ORAL, EXPRESSED OR IMPLIED (INCLUDING WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE AND MERCHANTABILITY AND WARRANTIES ARISING FROM COURSE OF DEALING OR USAGE OF TRADE) WITH RESPECT TO THE SOFTWARE DESCRIBED BELOW.

The Following Software Products Subject To This Agreement:

Order Number	Description	Price*

Ship To: _____

Bill To: _____

Method of Shipment: _____

Customer P.O. Number: _____

Agreed To:

PURCHASER	MOSTEK CORPORATION
By: _____	By: _____
Title: _____	Title: _____
Date: _____	Date: _____

* Prices Subject To Change Without Notice

3870/F8 PERIPHERAL ACCESSORIES

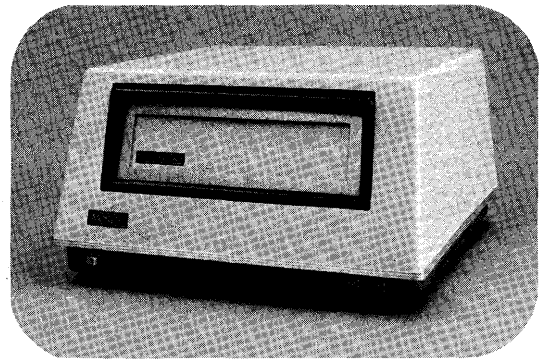
MOSTEK®

MICROCOMPUTER SYSTEMS HARDWARE

Aid Station (XAID-100)

FEATURES

- ☐ Card cage and power supply in one impact resistant enclosure.
- ☐ Forced air cooling.
- ☐ Accepts up to thirteen SDB size boards.
- ☐ Front and rear panels are removable.
- ☐ Hinged top—allows full accessibility to any board.
- ☐ Attractive styling—colors are white over blue.



DESCRIPTION

The MOSTEK Aid Station is intended as a "base" or starting point environment for system development applications. The thirteen card edge connectors are separated into one group of six and one group of seven. Wire wrap pins on the mother board allow for busing between the two .

Interface flexibility is assured as the single front panel and two rear panels can be drilled for mounting of controls, indicators, and connectors. (Additional blank panels are also available).

The Aid Station power supply is capable of supporting at least two complete systems simultaneously. For instance, one Z80 and one F8 system along with add on ROM/PROM or additional RAM. A typical user application could be a Z80 as system number 1 and the user's own prototype boards as system number 2.

As a further aid to system prototype fabrication, MOSTEK offers a wire wrap board (MK 79023) configured with power and ground. The MK 79023 accepts up to 120 equivalent 16-pin sockets.

Also, an extender card (MK79024) is available for troubleshooting individual boards.

SPECIFICATIONS

Aid Station Enclosure

Overall Dimensions — 20"W x 22"L x 12"H
Material — NORYL EN 185
Color Composition — White GE #8385; Blue GE #2283
Weight — 40 lb.
Front Panel Dimensions — 14.0" x 4.31"
Rear Panel Dimensions — 4.25" x 4.75"
4.25" x 2.125"
Fan Capacity — 50CFM
Card Cage accepts thirteen 8½" x 12.0" (SDB) size Boards.
Card Connectors — 100 Pin, 0.125" Centers
Operating Temperature Range — 0°C to 50°C

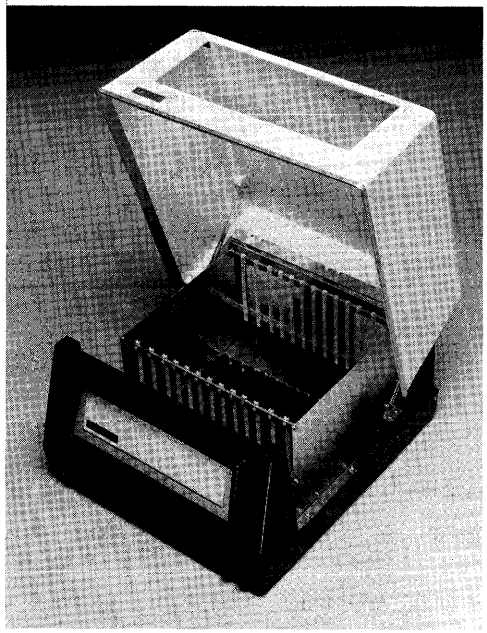
Power Supply

Input — 115VAC 60Hz
Outputs — +5VDC at 18 Amperes
+12VDC at 3.4 Amperes
−12VDC at 3.4 Amperes
Foldback current limiting.

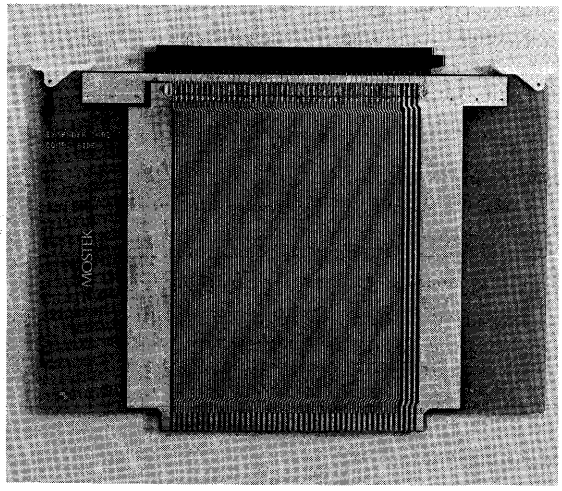
ORDER INFORMATION

NAME	DESCRIPTION	PART NO.	PRICE*
XAID-100	Enclosure, Card Cage, and Power Supply	MK79034	$850
XAID-103	Wire Wrap Card	MK79023	$100
XAID-104	Extender Card	MK79024	$100
XAID-105	Blank Panel Set	MK79065	$100

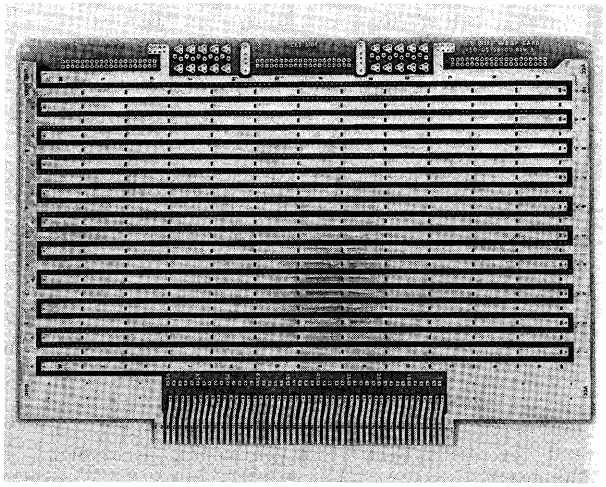
*Prices subject to change without notice and apply only within the U.S. and Canada.



XAID–100 Aid Station



XAID-103 Wire Wrap Card



XAID-104 Extender Card

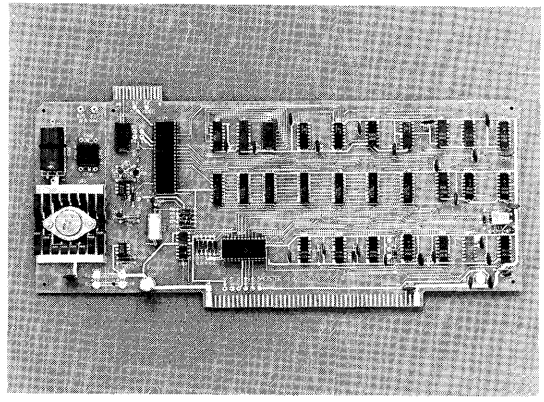
MOSTEK®

MICROCOMPUTER SYSTEMS

Video Adaptor Board (VAB-2)

FEATURES

- ☐ Complete video interface system on one board
- ☐ Single supply (+5VDC or 12.6VAC) operation
- ☐ On board rectifier and regulator for 12.6VAC operation
- ☐ 16 lines of 64 characters
- ☐ Full ASCII character set - 128 symbols including upper/lower case letters
- ☐ Full cursor controls: ↑ ↓ ← → home, screen clear, carriage return, erase to end of line/screen; plus direct X-Y addressing
- ☐ 8 bit ASCII or 5 bit Baudot operation



DESCRIPTION

The VAB-2 is a single board video terminal based on the MOSTEK MK3870 single chip microcomputer. It functions as an interface between a 20mA full duplex serial data loop, an ASCII encoded keyboard, and an EIA standard video monitor. The only other external component required is a 12.6 volt transformer.

The P.C. board 'form factor' facilitates installation within most standard keyboard housings. Alternatively, the 2 inch power supply section may be cut off the P.C. board allowing the board to be inserted into a standard 12" card rack (such as Mostek's XAID-100 MK79034) for system use.

SPECIFICATIONS

Operating Temperature 0°C – 50°C

Power Supply Requirements
5VDC±5% @ 0.75A max.
or
8 – 14 VAC rms @ 0.75A rms max.

Board size (with power supply) 14" x 6.5" x 1"
(without power supply) 12" x 6.5" x 1"

Video output 1.5Vp-p into 75Ω (EIA RS-170)

Current loop input/output 20mA nominal opto-isolated 240V max loop to ground

Keyboard inputs – standard TTL compatible

CUSTOMER SUPPLIED EQUIPMENT

Keyboard – Cherry B70-4753 or equivalent

Monitor – SC Electronics, Inc. 10M915 or equivalent

Transformer – Stancor P8384 or equivalent

MICROCOMPUTER BASED

The heart of the VAB-2 is the MK3870 single chip microcomputer. The MK3870 provides the following functions:

Serial data link interface
Control character decoding
Cursor positioning
Keyboard interface

ASCII OPERATION

In ASCII mode, the VAB-2 receives and transmits an 8 bit code (parity bit = 0 on transmit, ignored on receive). Two stop bits are transmitted by the VAB-2, but only one stop bit is required by the VAB-2 receiver. The VAB-2 works equally well with external systems transmitting one, two, or more stop bits. Available Baud rates for ASCII are 300 and 110.

See also Figure 1 – ASCII character set, and Table 1 – ASCII control characters.

BAUDOT OPERATION

In Baudot mode, the VAB-2 receives and transmits a 5 bit code (compatible with Model 15, Model 28, or similar Teletypes™). Two stop bits are transmitted, but only one stop bit is required by the VAB-2 receiver. The VAB-2 works equally well with external systems transmitting one, 1.5, or more stop bits. Available Baud rates for Baudot are 74.2 and 45.45. In Baudot mode, the only control codes available are carriage return and line feed. The Baudot "Letters" and "Figures" shift characters are generated automatically as required. Keys on the ASCII keyboard which generate codes having no equivalent Baudot code are ignored. ASCII code "Rubout" ($7F_{16}$ or 177_8) generates a "Letters" shift to facilitate synchronization of the distant end receiver.

ASCII CHARACTER SET

$\alpha\beta\gamma\delta\epsilon\theta\iota\lambda\mu\nu\pi\Sigma\phi\psi\omega\Omega$ 0 1 2 3 ^{0} ^{2} _ ÷ ≙ √ ∫ | ← ↑↓

! " # $ % & ' ( ) * + , – ; / 0 1 2 3 4 5 6 7 8 9 : ; < = > ?

@ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [\ ] ^

` a b c d e f g h i j k l m n o p q r s t u v w x y z { ¦ } ~ ▒

Figure 1

BAUDOT CHARACTER SET

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

– ? : * 3 $ & # 8 ( ) . , 9 0 1 4 ! 5 7 ; 2 / 6 "

Figure 2

FUNCTIONAL DIAGRAM

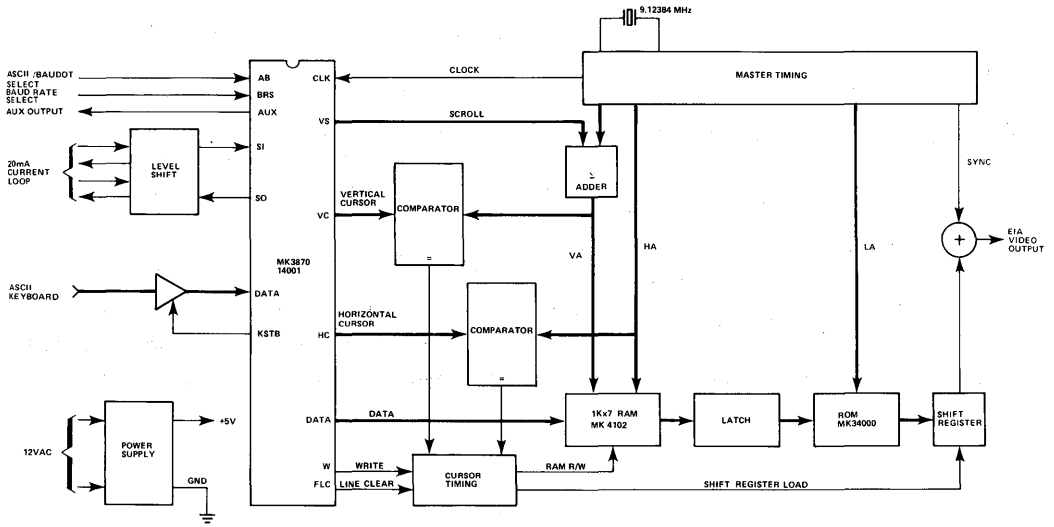


Figure 3

OCTAL	HEX	CNTL	FUNCTION	
004	04	D	HOM	Home — moves cursor to upper left corner of screen
005	05	E	EOL	Erase end of line — erases current line from right margin to current cursor position (1600mS max)
006	06	F	EOS	Erase end of screen — erases lines from bottom of screen to, but not including, current line (400mS max)
	1			
010	08	H	BS	Back space — move cursor left one column unless already in left most column
011	09	I	HT	Horizontal tab — moves cursor right one column unless already in right most column
012	0A	J	LF	Line feed — moves cursor down one line, scrolls screen up if already on bottom line
013	0B	K	VT	Vertical tab — moves cursor up one line, scrolls screen down if already on top line
014	0C	L	FF	Form feed — clears screen and homes cursor (400mS)
015	0D	M	CR	Carriage return — moves cursor to left margin
020	10	P	DS	Down shift sequence — causes character following DS to be interpreted as printable rather than control. Required for lower 32 symbols (Greek and math), but may be used with any characters.
021	11	Q	DC1	Device control — sets AUX bit
023	13	S	DC3	Device control — clears AUX bit
033	1B		ESC	Start cursor sequence — ESC + ΔV ΔH adds ΔV modulo 16 to vertical cursor address ΔH modulo 64 to horizontal cursor address ESC = ΔV ΔH sets vertical cursor address to ΔV modulo 16 horizontal cursor address to ΔH modulo 64
177	7F		DEL	Delete — moves cursor left one column, unless cursor was already on leftmost column; erases new position

TABLE 1. — ASCII CONTROL CHARACTERS

CHARACTER GENERATOR

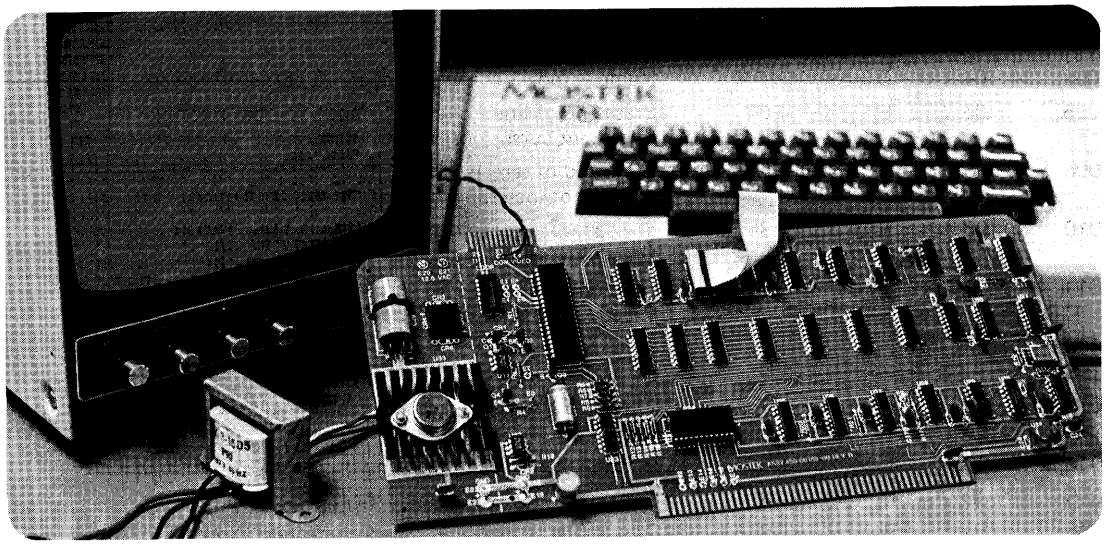
The VAB-2 is shipped with an MK34073 (2K x 8) character generator ROM, providing 128 displayable characters (see Figure 1 — ASCII character set). For custom applications, the MK34073 ROM may be removed and an MK2708 type PROM (1K x 8) installed, programmed with the user's custom font (external +12V and −5V or −12V supplies required for some PROMs). Alternatively, for high volume applications, a new ROM mask may be ordered. The MK34000 series can provide two complete 128 character sets per ROM. Provision is made for wiring the AUX bit to the ROM for program-selectable character font.

AUXILLARY BIT OUTPUT

A special output (AUX) is provided for custom control applications. AUX is capable of driving one TTL load, and is brought out to the P.C. edge connector. AUX is cleared upon power up and each time a DC3 character is recieved. AUX is set upon receipt of a DC1 character.

KEYBOARD

The VAB-2 interfaces directly with standard ASCII encoded keyboards. Although normally used with active high data and strobe keyboards, provision is made for active low keyboards.



CUSTOMER SELECTABLE OPTIONS

- ☐ 50/60 Hz (Strap option)
- ☐ 110/300 Baud ASCII (strap option)
- ☐ 74.2/45.45 Baud Baudot (strap option)
- ☐ MK34000 series ROM or MK2708 type PROM character generator (strap and population option; MK34073 standard)
- ☐ 5VDC or 12VAC operation (strap and population option; 12 VAC standard)
- ☐ Serial loop connector — 16 pin DIP socket or 26 pin edge connector
- ☐ Active high or active low keyboard input (population option; active high standard)
- ☐ Custom features and/or character generator for high volume OEM applications (one-time mask charge applicable)

ORDER INFORMATION

NAME	DESCRIPTION	PART NO.	PRICE
VAB-2 Operations Manual	Detailed description of the use and operation of VAB-2	MK79560	$ 1.50
VAB-2 Source Listing	Source Listing of the 3870 Firmware used in VAB-2	MK79561	$ 15.00
MK3870/ 14001 Firmware Package	Pre-programmed 3870 used with VAB-2 plus the Operations Manual and Source Listing described above	MK79056	$ 50.00
VAB-2	Assembled and tested VAB-2 Circuit Board plus the Operations Manual and Program Source Listing	MK79052	$195.00

*Prices are subject to change without notice and apply only to U.S. and Canada.

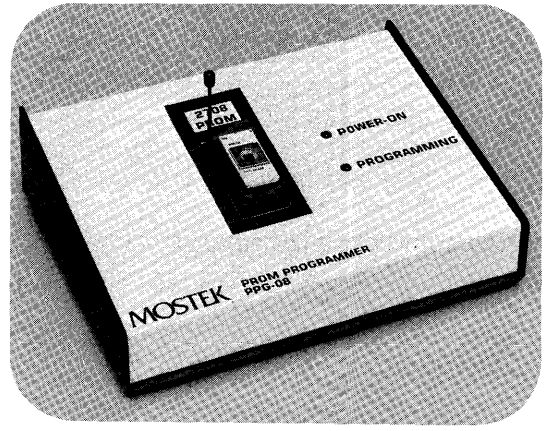
MOSTEK

MICROCOMPUTER SUPPORT

Prom Programmer (PPG-08)

FEATURES

- ☐ Programs, reads, and verifies MK 2708 PROMS
- ☐ Directly interfaces to SDB-50/70 and SDB-80
- ☐ Driver software included
- ☐ Zero insertion force socket
- ☐ Power and programming indicators



GENERAL DESCRIPTION

The MK 2708 PROM Programmer (PPG-08) is a peripheral which provides a low-cost means of programming MK 2708 UV erasable PROMs. The PPG-08 has a generalized computer interface (two 8-bit I/O ports) allowing it to be controlled by most types of host computers with user-generated driver software. It is directly compatible with MOSTEK's F8 Software Development Board (SDB-50/70) and Z80 Software Development Board (SDB-80). Driver software in paper-tape form and source listings for the SDB-50/70 and SDB-80 are included with the purchase of the PPG-08. A complete set of documentation is also provided with the PPG-08 which describes the internal operation and details user's operating procedures. Interface cables for the SDB-50/70 and SDB-80 may be purchased separately. Another optional accessory is a TI Silent 700 compatible cassette tape containing control software for the SDB-50/70 and SDB-80.

SPECIFICATIONS

Interface

40 pin control connector (.1" centers card edge)
12 pin power connector (.156" centers card edge)
All control signals are TTL compatible.

Power requirements

+12 VDC @ 250 mA typical
+ 5 VDC @ 100 mA typical
−12 VDC @ 50 mA typical

Operating Temperature . . . . . 0° to 50°C

Programming time (maximum) . . . . . 2.5 minutes

Physical Dimensions . . . . . 5" x 7" x 2"

ORDER INFORMATION

NAME	DESCRIPTION	PART NO.	PRICE*
PPG-08	MK 2708 PROM Programmer	MK 79033	$300
XAID-805	Cable for interface to SDB-80	MK 79041	$ 30
XAID-705	Cable for interface to SDB-50/70	MK 79046	$ 30
SWD-1	Driver software on TI Silent 700 compatible cassette tape for SDB-50/70 and SDB-80	MK 79051	$ 50

*Prices are subject to change without notice and apply only in U.S. and Canada.

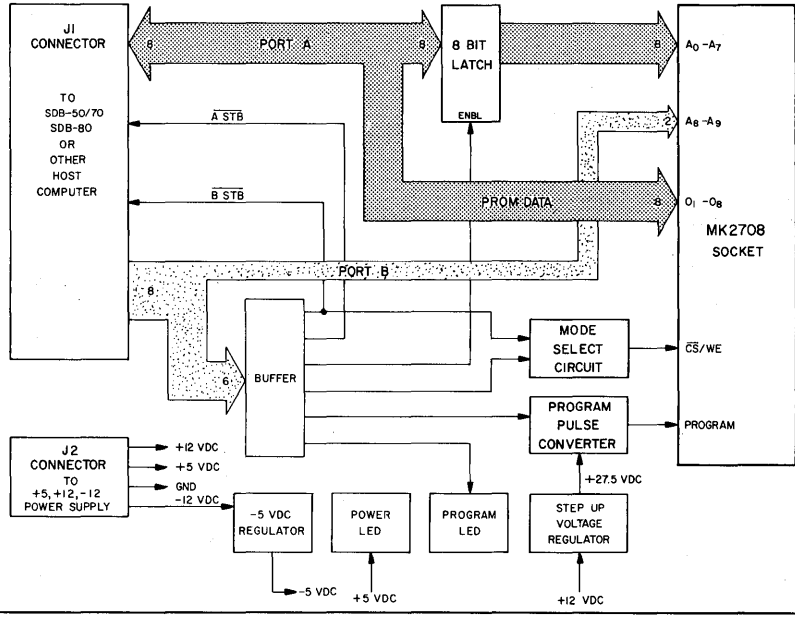
CONTROL CONNECTOR (J1) PIN-OUT

PIN #	Signal Name	Direction	Description
All Odd Pins (1-39)	GND		Logic Ground
J1-2	ASTB	Output	"LOW"when Port A (PA0-PA7) is in output mode
	PA0 - PA7	Bidirectional	PORT A (PA0-PA7) is used to output the lower 8 bits of PROM address to latch, output PROM data during programming and input PROM data during read sequence.
J1-24	$\overline{BSTB}$	Output	"LOW" when Port A (PA0-PA7) is in input mode.
J1-26	PB0/ADDR8	Input	PROM address bit 8
J1-28	PB1/ADDR9	Input	PROM address bit 9
J1-30	PB2/PAIN	Input	"HIGH" when Port A (PA0-PA7) is in input mode and PROM is in read mode.
J1-32	PB3/PROG MODE	Input	"HIGH" during program mode.
J1-34	PB4/PROG PULSE	Input	Programming Pulse
J1-36	PB5/PA OUT	Input	"HIGH" when Port A (PA0-PA7) is in output mode.
J1-38	PB6/CLK LATCH	Input	Clock to strobe address bits 0-7 into latch
J1-40	PB7/PROG LED	Input	Control line for programming indicator

POWER CONNECTOR (J2) PIN-OUT

J2-1,A	$+5V_{DC}$	J2-4, 5, D, E	$+12V_{DC}$
J2-2, 3, B, C	GND	J2-6, F	$-12V_{DC}$

BLOCK DIAGRAM



APPLICATION NOTES

MOSTEK®

F8 MICROCOMPUTER SUPPORT

Application Note

USING MOSTEK'S F8 IN A SCANNED KEYBOARD APPLICATION

Using Mostek's F8 In A Scanned Keyboard Application

BLOCK DIAGRAM OF
4x4 KEYBOARD MATRIX

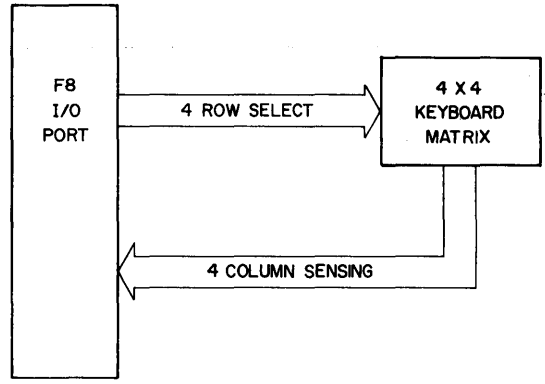


Figure 1

INTRODUCTION

Many microprocessor based systems require input from a keyboard of some type. The hardware required to encode a keyboard outside of the processor can be eliminated by using a keyboard scanning technique. With one F8 port, a 16 switch keyboard can be scanned (see fig. 1) using no external hardware. This is because of the bi-directional quality of the F8 ports.

THEORY OF OPERATION

When scanning the keyboard, one of the four row select bits is turned on supplying a ground return for one row of switches. The column data is then read back into the processor via the four column bits. These four bits will indicate the condition of all four switches in the selected row. Each of the four rows is selected, one at a time, continously providing current status of all 16 switches.

"BOUNCE" is a problem encountered when using mechanical switches (see fig. 2). In order to prevent multiple detection of the switch closure, the bounce must be filtered out. A conventional solution to the bounce problem was to use an R–C filter and attempt to eliminate it electrically. However, when using the F8 scanning technique the switch bounce can be filtered in software by taking multiple samples of the switch to verify switch depression and release.

Since the software must usually scan all switches continuously, a register (or half) can be used to maintain the status of each switch.

A common requirement for keyboards is "N–key rollover", meaning that if more than one switch is depressed at a time, all switch closures will be detected. This requirement can be met when using the scanning technique as described above. Since all switches are continuously scanned, the condition of each switch is always available to the processor.

SWITCH BOUNCE

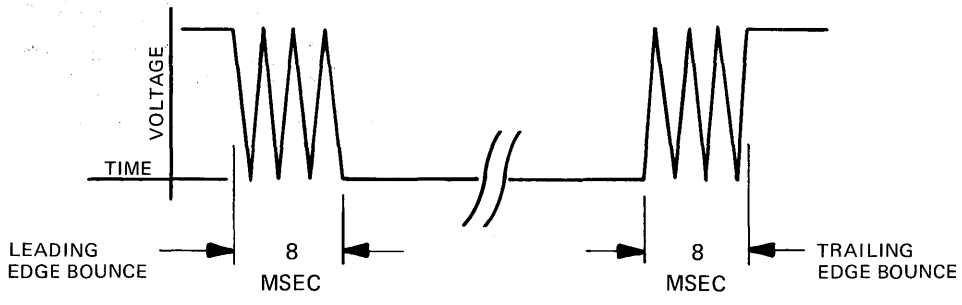


Figure 2

4 x 3 KEY MATRIX

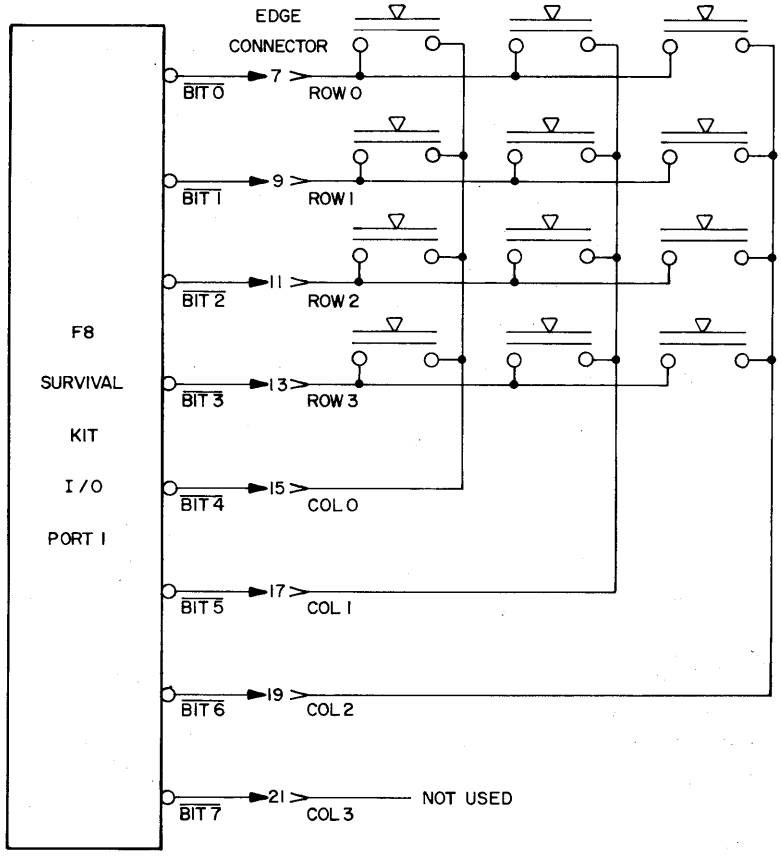


Figure 3

EXAMPLE HARDWARE DESIGN

The example in figure 3 shows a 4x3 matrix interfaced to an F8 port. This arrangement will provide N–key rollover input to the processor unless three keys are depressed simultaneously to form an L configuration. Then erroneous input could occur. If this presents a problem for a given application, one germanium diode (1N 270) should be added on the column pole of each switch (see fig. 4).

The operation of this keyboard (fig. 3) is simple. To sense the condition of row 0, a Hex '01' is written to port 1. Port 1 is then read back. The state of bits 4, 5 and 6 (COL 0, COL 1, COL 2) will be 1 if the respective switches in row 0 are closed and 0 if

FOR SOME APPLICATIONS DIODES ARE NECESSARY

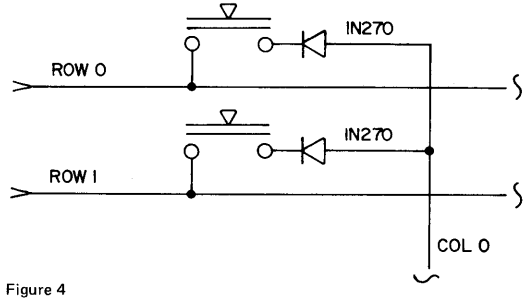


Figure 4

KEYBOARD SCAN ROUTINE (4 x 3 MATRIX)

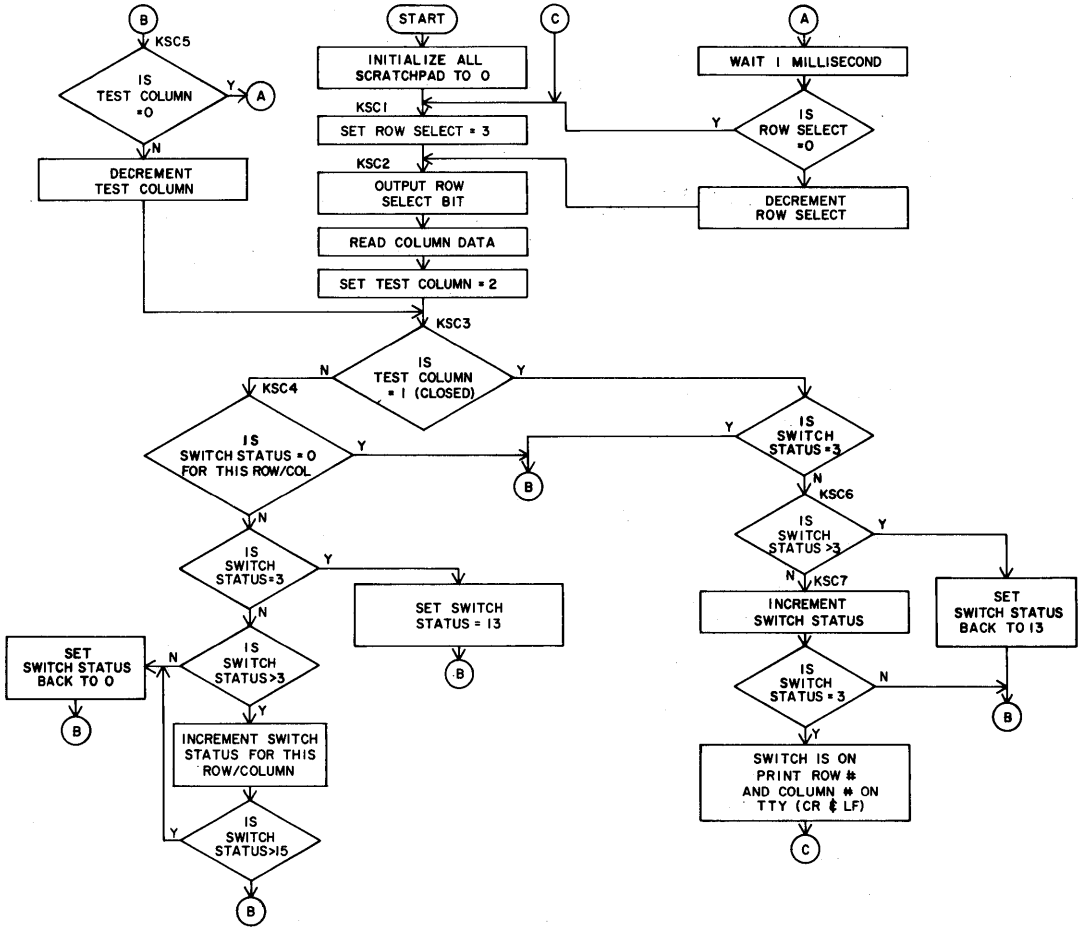


Figure 5

open. (Note: The F8 I/O ports contain internal pull-ups). The other three rows are read similarly.

EXAMPLE SOFTWARE FOR THE 4x3 MATRIX KEYBOARD

An example program was written to run on the F8 Survival Kit to demonstrate software switch sensing and debounce.

One scratchpad register is used to maintain current status for each switch. When a switch is inactive it maintains a status of 0. In order for the switch to be processed, three consecutive scans must occur in which the switch is sensed to be closed.

When a switch is first sensed closed, its status is incremented to 1. In succeeding scans its status is either incremented (if sensed closed) or reset to 0 (if sensed open) until the status reaches 3, thus requiring three consecutive scans with the switch closed.

The switch is then processed, which in the example means the column number and row number are printed on the TTY (terminal).

A status of 3 is maintained by the switch until the first time it is sensed open. At that time its status is set to 13. Then three consecutive scans with the switch open are required to get the switch back to inactive status (0). This is accomplished by incrementing the status (if sensed open) or resetting the status to 13 (if sensed closed) until it reaches 15. The status is then reset to 0. As long as bounce occurs, however, the status will be reset to 13.

The flowchart (fig. 5) shows the logic described above. Note that at the end of each row scan there is a one millisecond delay which effects an interscan delay of 4 milliseconds for each switch. This means that the switch must be on 'solid' for 8 milliseconds before being processed and off 'solid' 8 milliseconds before becoming inactive again; so the switch will only be processed one time per depression. This debounce time sets the max keyboard entry rate for a given switch at 1 entry/24 milliseconds.

Figure 6 shows the scratchpad register assignments used by the example program.

For an instruction by instruction description of the example program see the listing (fig. 7).

ALTERNATE DESIGN APPROACHES

When more than 16 switches are needed, an additional chip must be used. By adding a 4 to 16 decoder (see figure 8) to select 1 of 16 rows, up to 64 switches can be scanned.

Many off-the-shelf keyboards are available which have a 4x3 or 4x4 physical arrangement, but all switches have one common pole (on the P.C. Board). This type of keyboard can be scanned by using a 4 bit code to select one of up to 16 switches. The code is

SCRATCHPAD REGISTER ASSIGNMENTS

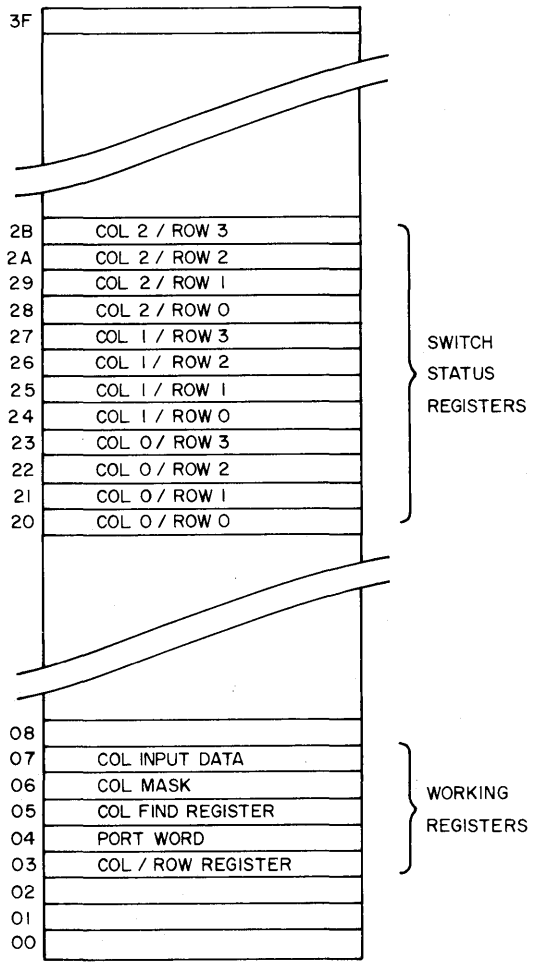


Figure 6

then decoded by a 4 to 16 decoder which supplies a ground return to the selected switch. The switch common line is then read to sense the condition of that switch (see figure 9).

If more ports can be assigned to the keyboard interface, other options may become advantageous. For example, with two ports 16 switches can be read without scanning. The basic requirements such as switch debounce and N-Key rollover will remain regardless of which option is taken. The best approach to a given design application will be determined by the system requirements and structure.

16 x 4 KEYBOARD

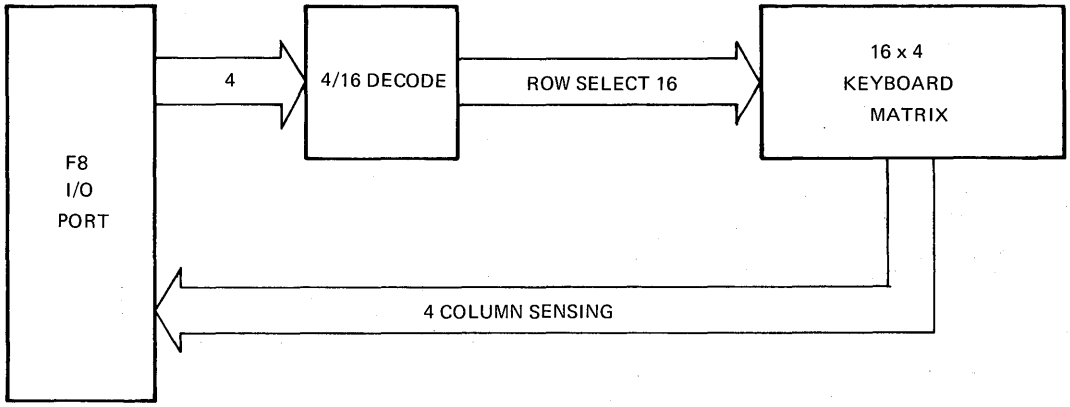


Figure 8

KEYBOARD WITH COMMON POLE

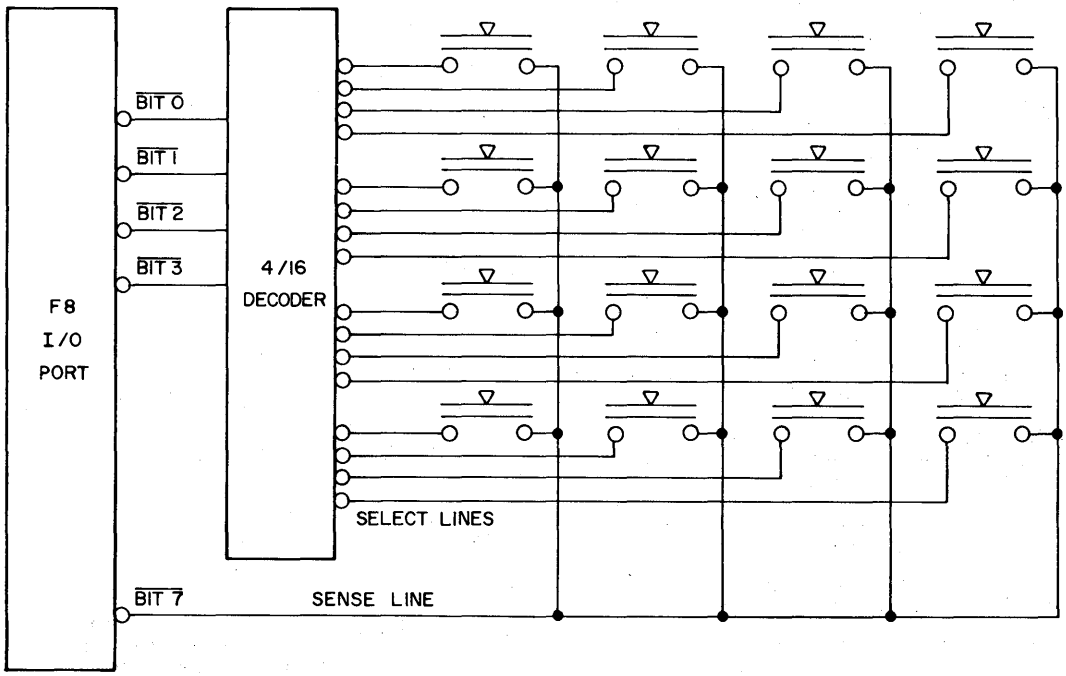


Figure 9

September 1976

MOSTEK

F8 MICROPROCESSOR SUPPORT

APPLICATION NOTE

USING MOSTEK'S F8 IN A SCANNED SEVEN-SEGMENT DISPLAY APPLICATION

Using Mostek's F8 In A Scanned Seven-Segment Display Application

by Dan Hammond

INTRODUCTION

Many microprocessor based devices require a numeric display as an integral part of the system. For reasons of cost and reliability, it is usually desirable to keep the chip count as low as possible with the microprocessor performing the control logic in software. Time multiplexed digit scanning is a common solution and works very well using a single F8 port for up to 8 digits.

THEORY OF OPERATION

An eight digit display can be scanned with one F8 port (fig. 1) by using half of the port for the BCD number and half for the digit select. When using the digit scanning technique an 'image' of the display must be maintained in memory, with a byte (or half byte) of memory containing the BCD number to be displayed in each of the eight digits. The following five steps show the basic control the software is required to execute:

Step 1 Output digit select and BCD number for this digit (from 'image')

Step 2 Turn on strobe

Step 3 Delay

Step 4 Turn off strobe

Step 5 Increment digit select, return to step 1

The scan rate should be fast enough to prevent the display from 'flickering'. It has been found that a 80 to 100Hz rate is sufficient for a stationary display. An approximate 100Hz rate is achieved in an eight digit display by making the delay in step 3, 1.25 milliseconds.

Maximum brilliance will be provided by leaving the strobe on for the whole delay time. This provides a 1/8 or 12.5% duty cycle. Reducing the strobe width will reduce the duty cycle and cause the display to be dimmer.

Interdigit blanking to prevent a blurring effect is accomplished by strobing the digit decoder after digit select/BCD number data is present on the port and removing the strobe before changing the data (see fig. 2)

EXAMPLE HARDWARE DESIGN

The example design in Figure 3 shows the hardware simplicity in an LED display scanning circuit interfaced to the F8. Bits 0-2 are used to select the digit, bit 3 as a strobe and bits 4-7 for the BCD number.

In this eight digit display the current required from the segment drivers and anode drivers is approximately 6-8 times what it would be for a static non-scanned display of equal brillance because only one digit is receiving current at a time (12.5% Duty Cycle).

NUMERIC DISPLAY BLOCK DIAGRAM

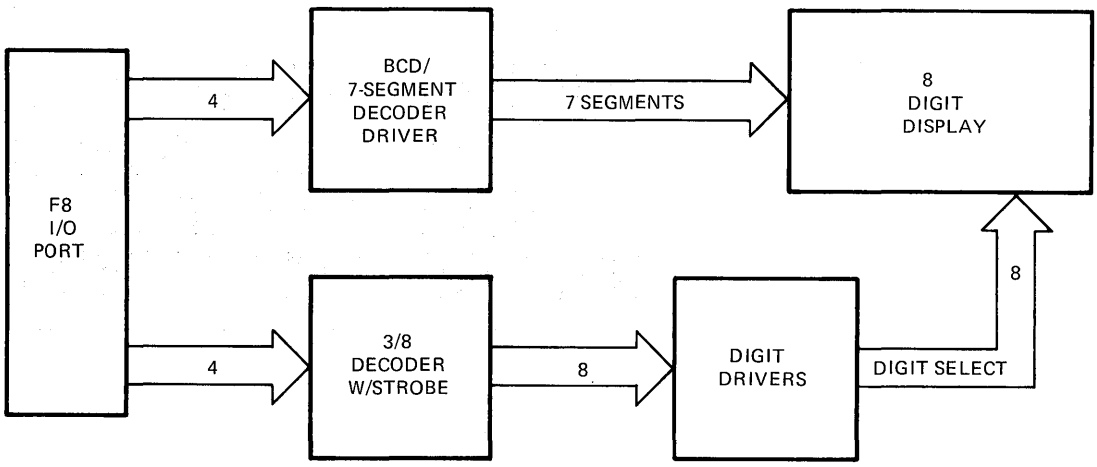


Figure 1

INTERDIGIT BLANKING

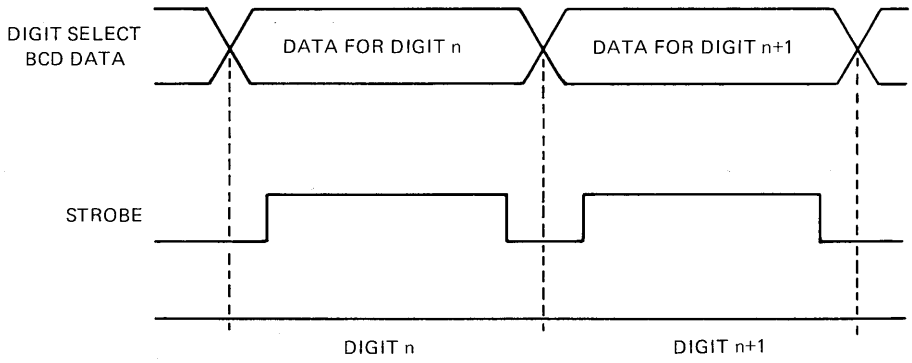


Figure 2

The SN7447 seven segment decoder/driver sinks 40mA per segment which will supply a maximum average current of 5mA per segment to each digit. This is an acceptable current level for many 7 segment LED displays such as the .43 inch HP7650. Since the anode transistors must drive seven segments, they will be required to source 280mA peak at a 12.5% duty cycle. Many discrete transistors (such as the 2N2907) and transistor arrays will handle this load.

7 SEGMENT DISPLAY INTERFACE

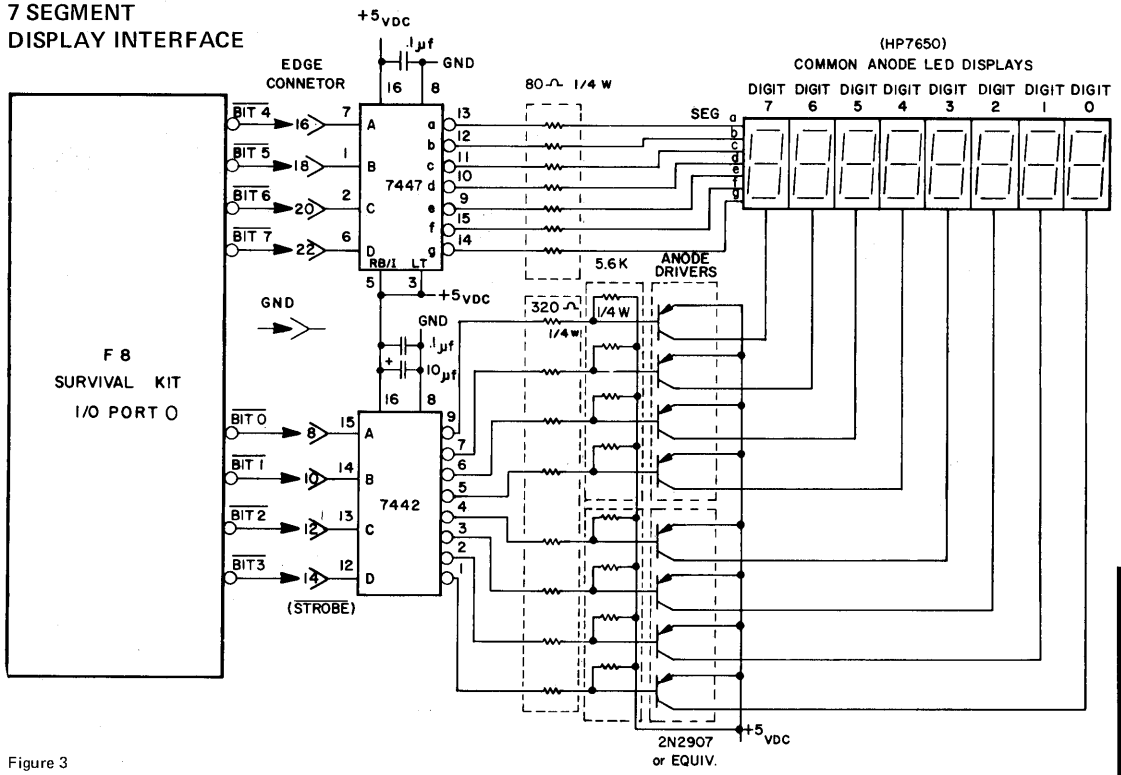


Figure 3

EXAMPLE CONTROL SOFTWARE

The 'MAIN PROGRAM' flow chart (Fig. 4) shows the initialization needed to start the scanning process. The main program must provide a means of entering numbers into the RAM image of the display in addition to the other processing required by the system.

As the BCD numbers are entered into the 'image' the interrupt service routine named 'SCAN' displays them. Note that the flow chart (Fig. 5) for 'SCAN' contains the five basic steps described in The Theory of Operations section (page 2).

The timer in the SMI (Static Memory Interface) chip provides periodic interrupts resulting in a continuous scan of the display. Therefore, only the SMI timer constant has to be changed in order to adjust the scan cycle delay. A key advantage to this interrupt scheme is that it effects a very minimal time burden on the processor. Specifically, every 1.5 milliseconds the interrupt routine takes less than .1 milliseconds to maintain the display scan, using less than 6% of F8 processor time. (It should be noted here that if a scanned keyboard is in the system, the timer interrupt service routine could also scan the keyboard and maintain its status).

In the example program the last eight F8 scratchpad registers are assigned to be used for the display image, register 0 for the display port image, register 1 for saving the display ISAR (Indirect Scratchpad Address Register), register 2 for saving the 'MAIN PROGRAM' ISAR, register 8 for saving the accumulator, and register J (9) for saving the status word (w), (See Fig. 6).

All six bits of ISAR are used to address the 'image' with the least significant three bits also defining the digit in which the addressed 'image' data is to be displayed. The instruction on line number 12 of figure 7 (LR A, I) loads the contents of the location in the scratchpad 'image' addressed by ISAR into the accumulator, then increments ISAR (preparing ISAR for the next interrupt).

MAIN PROGRAM

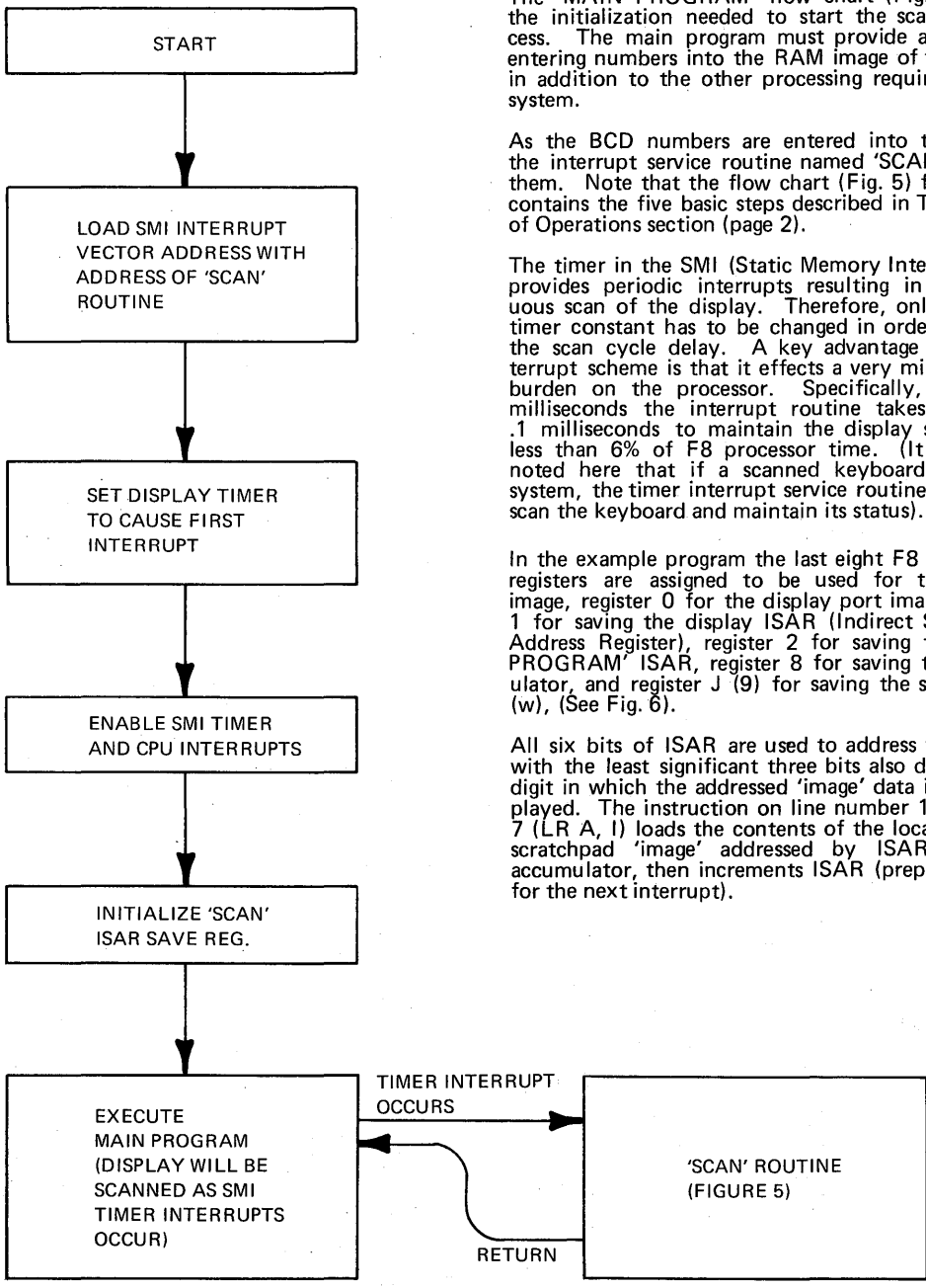


Figure 4

INTERRUPT SERVICE ROUTINE FOR NUMERIC DISPLAY

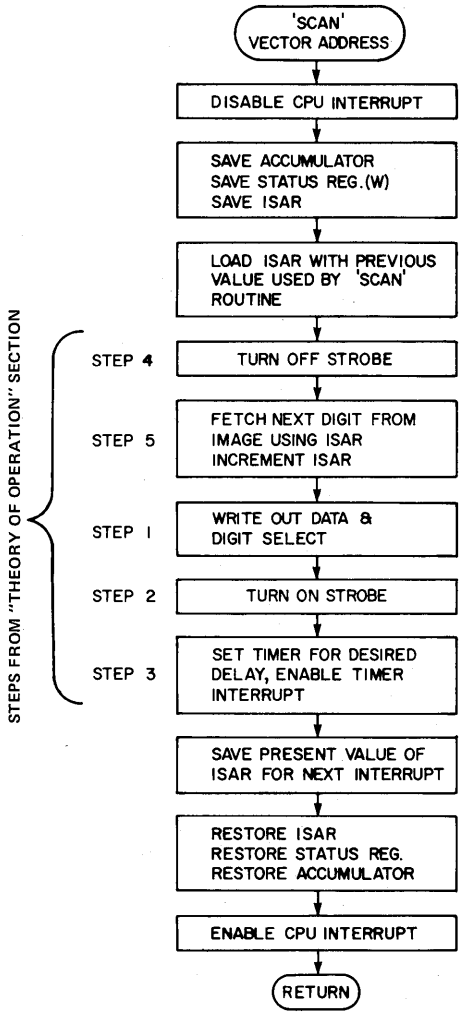


Figure 5

F8 SCRATCHPAD REGISTER USAGE MAP

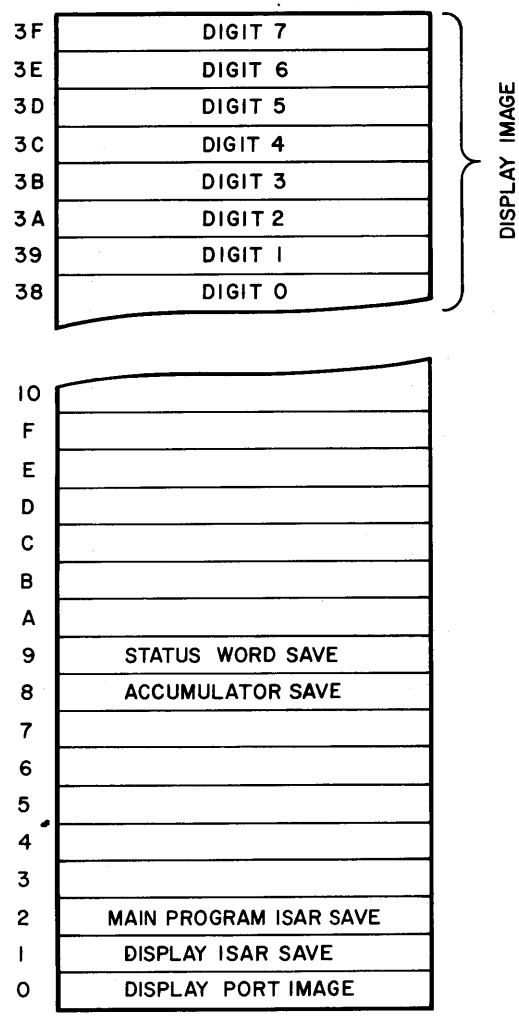


Figure 6

Output port H'F' is the timer constant register in the SMI chip (see line 1C in figure 7). Port H'E' is a register used to enable the timer interrupt in the SMI (line 1F). Note also that all outputs to the display port are 'OUTS 0' selecting port 0 (line E, line 17 & line 19).

The program listing (Fig. 7) contains comments that specify the purpose of each instruction.

SDB RESIDENT ASSEMBLER LISTING Figure 7

```
LINE # ADDRESS  OBJECT CODE        SOURCE CODE              COMMENTS
0000                        *                            INTERRUPT SERVICE ROUTINE
0001                        *                            FOR NUMERIC DISPLAY
0002                        *
0003                        *
0004                                ORG   H'700'
0005  0700      1A            SCAN  DI                   DISABLE CPU INTERRUPTS
0006  0701      58                  LR    8,A            SAVE ACCUMULATOR
0007  0702      1E                  LR    J,W            SAVE STATUS REG
0008  0703      0A                  LR    A,IS           LOAD ISAR INTO ACCUMLATOR
0009  0704      52                  LR    2,A            SAVE ISAR FROM MAIN PROGRAM
000A  0705      41                  LR    A,1            LOAD ACCUMULATOR WITH PREV ISAR
000B  0706      0B                  LR    IS,A           LOAD ISAR FOR SCAN
000C  0707      40                  LR    A,0            LOAD PREVIOUS DISPLAY PORT DATA
000D  0708      21 F7               NI    H'F7'          MASK OUT STROBE BIT
000E  070A      B0                  OUTS  0              TURN OFF STROBE
000F  070B      0A                  LR    A,IS           LOAD ISAR INTO ACCUMULATOR
0010  070C      21 07               NI    7              MASK OUT ISAR(U)
0011  070E      50                  LR    0,A            ISAR(L) TO R0 FOR DIGIT # SELECT
0012  070F      4D                  LR    A,I            GET BCD DATA USING ISAR,INC ISAR
0013  0710      15                  SL    4              MOVE IT TO MS HALF OF ACCUMULATO
0014  0711      C0                  AS    0              ADD DIGIT # TO BCD DATA
0015  0712      18                  COM                  INVERT DATA SINCE PORTS NEG TRUE
0016  0713      21 F7               NI    H'F7'          MASK OUT STROBE BIT
0017  0715      B0                  OUTS  0              WRITE NEW DATA OUT (NO STROBE)
0018  0716      22 08               OI    8              STROBE BIT ON
0019  0718      B0                  OUTS  0              TURN ON STROBE
001A  0719      50                  LR    0,A            SAVE DISPLAY PORT DATA
001B  071A      20 C4               LI    H'C4'          TIMER CONSTANT
001C  071C      BF                  OUTS  H'F'           WRITE TO SMI TIMER
001D  071D      73                  LIS   3              LOCAL INTERRUPT ENABLE BITS
001E  071E      BE                  OUTS  H'E'           ENABLE LOCAL INTERRUPTS
001F  071F      0A                  LR    A,IS           LOAD ISAR INTO ACCUMULATOR
0020  0720      51                  LR    1,A            SAVE DISPLAY SCAN ISAR
0021  0721      42                  LR    A,2            LOAD MAIN PROGRAM ISAR VALUE
0022  0722      0B                  LR    IS,A           RESTORE ISAR WITH IT
0023  0723      1D                  LR    W,J            RESTORE STATUS REG
0024  0724      48                  LR    A,8            RESTORE ACCUMULATOR
0025  0725      1B                  EI                   ENABLE CPU INTERRUPTS
0026  0726      1C                  POP                  RETURN TO MAIN PROGRAM
0027                                END
00

SCAN  0700
```

ALTERNATE DESIGN APPROACHES

There are several other approaches to a numeric display interface with the F8. For example, the BCD to seven segment conversion and 3/8 digit decoding could be done in software. This approach (Fig. 8) uses two ports.

If four ports are available, the display could also be driven statically, with each port controlling two digits. This approach (Fig. 9) would require one BCD to 7-segment decoder/driver (and 7 resistors) for each digit.

The best design approach depends on the application and the number of F8 ports available.

ALTERNATE SCANNING APPROACH

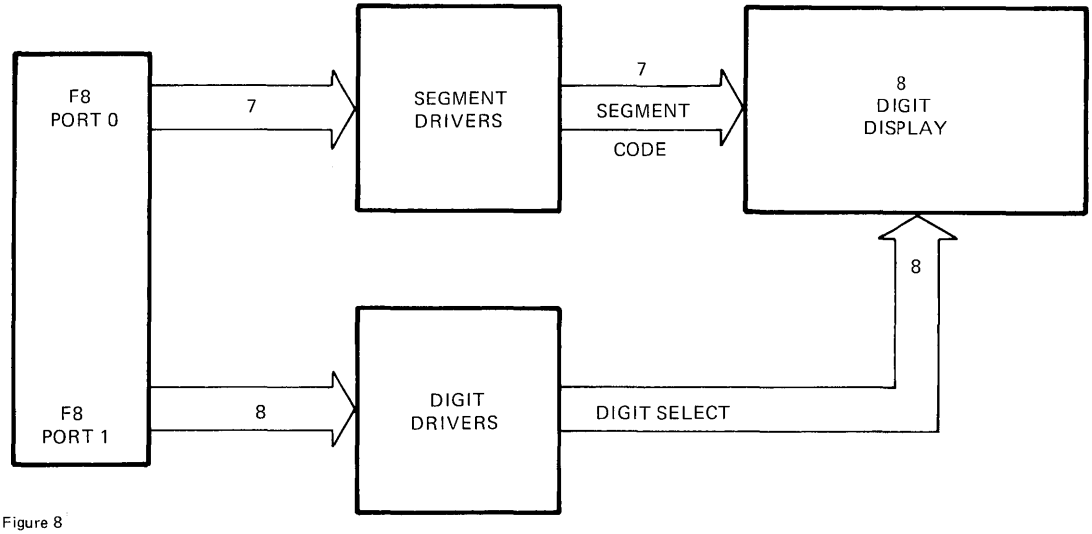


Figure 8

STATIC DISPLAY APPROACH

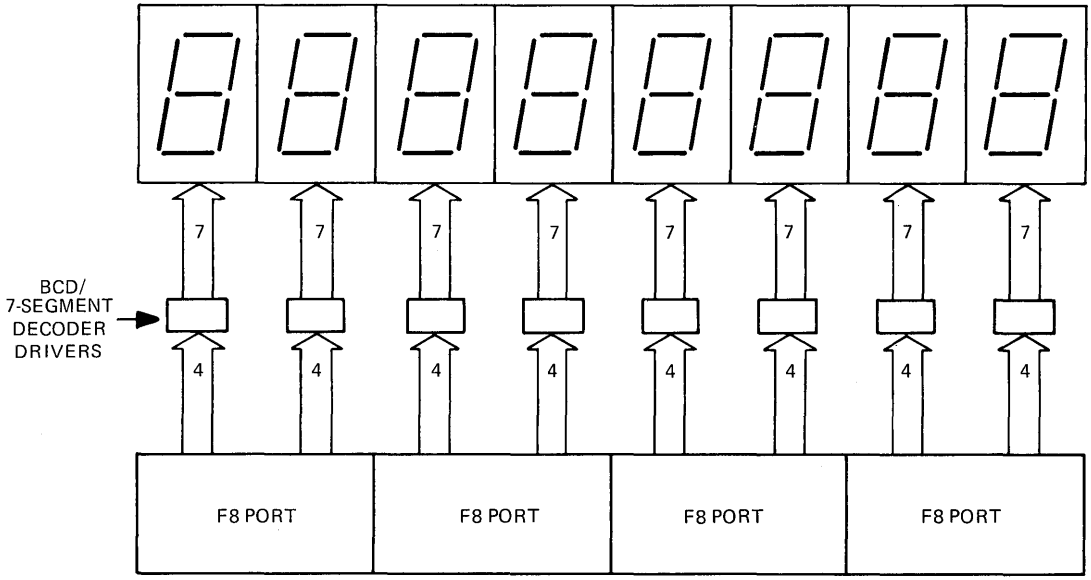


Figure 9

MOSTEK®

F8 MICROCOMPUTER SUPPORT

Application Note

EXPANDING MOSTEK'S F8 EXTERNAL INTERRUPT CAPABILITIES

Expanding Mostek's F8 External Interrupt Capabilities

by Jim Vittera

INTRODUCTION

One of the considerations involved in the design of any microprocessor based system is how to structure the interface between the peripherals (inputs or devices being controlled) and the CPU. The data line interface is usually dictated by the peripheral itself (e.g., a paper tape reader is eight bits of parallel data, a teletype is two lines of serial data, and a switch or front panel lamp usually only requires one line of data). The control lines of these peripherals however, can be handled in one of two basic ways by the system designer. The first method of handling these control lines, which is probably the most common, is to have the CPU periodically scan the control lines (connected to a I/O Port) to see if they require service. This is done by a small program which inputs the control lines through an I/O Port into the accumulator. They are then tested to determine if a line is active and the program flow diverted to service the active control line. The second method is to allow these control lines to interrupt the processor and divert program flow to service that peripheral. Servicing of these control inputs in a F8 based system is the topic of this application note with particular emphasis placed on implementing interrupt driven systems.

SCANNED VS INTERRUPT DRIVEN SYSTEMS

The basic difference between scanned and interrupt driven systems is that in a scanned system the peripherals are checked periodically to see if they need service. This periodic interval can be determined by the count down of a hardware timer (a software timer could be used, but the CPU would be tied up implementing a ripple counter—not a very effective use of the microprocessor). This technique is good for peripherals which can wait for service by the CPU (the maximum time would be the time between counter outputs), and good examples are any peripheral activated or observed by a human. For example a keyboard/display might be scanned at 1 ms intervals, as determined by the timer, which would be slow by microprocessor standards but exceedingly fast by human standards (after pressing a key or throwing a switch an extra 1 ms delay in service would not be noticeable).

FLOWCHART FOR SCANNING
N CONTROL LINES

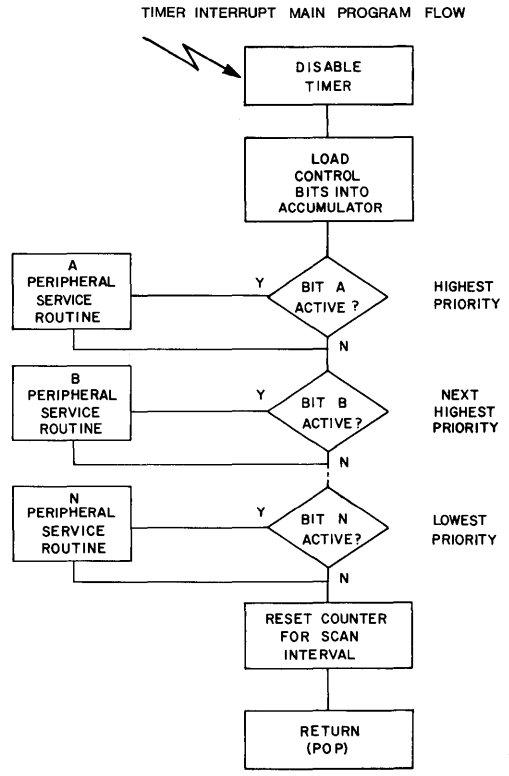


Figure 1

On the other hand many microprocessors are involved in the control of fast peripherals (Floppy Disk) or real time systems where quick response by the processor is required. In these situations, interrupt driven systems are mandatory, because the processor can be diverted from its present task to service the interrupting device in the order of tens of microseconds. Scanned systems are usually perferred by the system designer because they usually require less hardware, especially when implemented in a F8 System with its hardware timers. Figure 1 is a flow chart of a scanned system where the interval between scans is determined by the value preset into the timer. Note that priority is established by the order in which the control bits are tested and can be changed entirely by software.

HARDWARE VECTORED INTERRUPTS

The interrupt technique used by F8 Family devices capable of interrupting the CPU (PSU, PIO, or SMI) is to have the interrupting device provide to the CPU a Interrupt Vector unique to that interrupt. The CPU then loads this vector directly into the program

counter (saving the previous program counter in P) directing the CPU to the service routine for this interrupt. This technique provides a fast response to the interrupt because no time is consumed in polling to locate the interrupting device. In addition to providing automatic vectoring of the interrupts, the F8 devices provide automatic prioritizing of the interrupts. Priority is determined by the placement of the interrupting device in a daisy chain structure – a location closer to the CPU means higher priority – as shown in Figure 3. $\overline{ICB}$ is an output from the F8 CPU to indicate if interrupts have been enabled by the use of an EI instruction in the program being executed. $\overline{ICB}$ goes low when interrupts have been enabled, thereby enabling the daisy chain of interrupting devices. One or more of the three $\overline{EXT\ INT}$ inputs shown goes low signaling a request(s) for service by one or more of the peripherals. The device or devices that have $\overline{EXT\ INT}$ low now pull their $\overline{INT\ REQ}$ line low (assuming interrupts are not disabled at the local level) signaling the processor to begin an interrupt service sequence. The status of the $\overline{INT\ REQ}$ line is tested by the CPU at the end of every instruction which is not privileged. Privileged instructions cannot be interrupted so the CPU waits until the end of the next instruction (which is not privileged) to test the $\overline{INT\ REQ}$ line. When the CPU finds the $\overline{INT\ REQ}$ line low it begins the interrupt sequence by saving the Program Counter in P and using the ROM Control Lines to command the interrupting peripheral to transfer its vector address to the Program Counter. The 3851 is the highest priority device in Figure 3 and if its $\overline{EXT\ INT}$ line is low it sets its $\overline{PRI\ OUT}$ signal high thereby disabling all lower priority devices and outputs its vector address on the Data Bus. Should the PSU not be the interrupting device, it leaves its $\overline{PRI\ OUT}$ signal low passing the request to the second device in the chain (the PIO in this case). If the PIO is interrupting, it raises its $\overline{PRI\ OUT}$ line to a logic one and outputs its vector address. $\overline{PRI\ OUT}$ going high prevents all devices of lower priority from outputing their vector address even though they may be trying to interrupt. Twenty two cycles of the Φ clock are required to complete this interrupt vector fetch sequence. The next event that occurs is an instruction fetch from the location specified from the vector address. The SMI doesn't have a $\overline{PRI\ OUT}$ signal therefore it must be the lowest priority device in the system. The time required to get to an interrupt service routine can be calculated as shown in Figure 2 (at a 2 MHz Φ rate).

INTERRUPT VECTOR FETCH

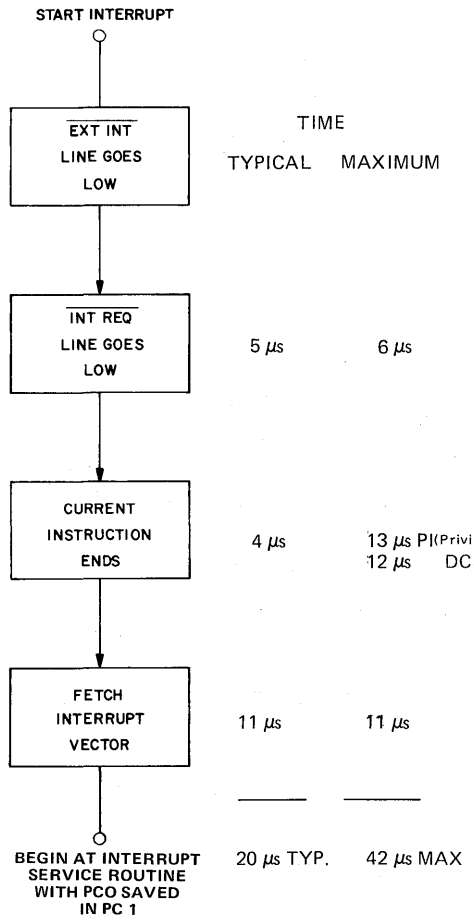


Figure 2

The time from an interrupt striking to the start of execution of its service routine is highly dependent on the instruction being executed at the time of the interrupt. The maximum number was based on a long privileged instruction such as PI followed by a long non-privileged instruction such as DCI. The typical instruction time is based on a 2 cycle instruction although many F8 instructions are one byte/one cycle instructions. The 6.0μs max number represents the propagation delay through the peripheral device from $\overline{EXT\ INT}$ to $\overline{INT\ REQ}$ (interrupts from the timer do not incur this delay). Once the $\overline{INT\ REQ}$ is recognized, 22 cycles are required to stack the program counter and fetch the interrupt vector. One technique that can be used to minimize the maximum delay that would be incurred upon an interrupt is to constrain the instructions that are executed when the interrupt is expected. A method that would reduce the maximum delay from the interrupt striking to the execution of the first instruction of the service routine would be to put the processor in a branch on self loop (BR*). This essentially provides a wait for interrupt situation with the CPU running in a loop waiting for the interrupt.

EXPANDING INTERRUPT INPUTS IN A MINIMUM SYSTEM

In a two-chip F8 microcomputer system (MK 3850 CPU and MK 3851 PSU) the system can be interrupted by either the timer in the PSU or the $\overline{EXT\ INT}$ line of the PSU. Thirty-two lines of bi-directional I/O are available and it may be desirable to have more than one input capable of interrupting

F8 SYSTEM INTERRUPT CONNECTION

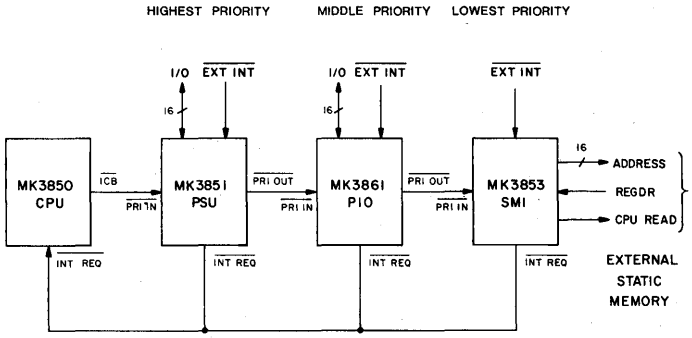
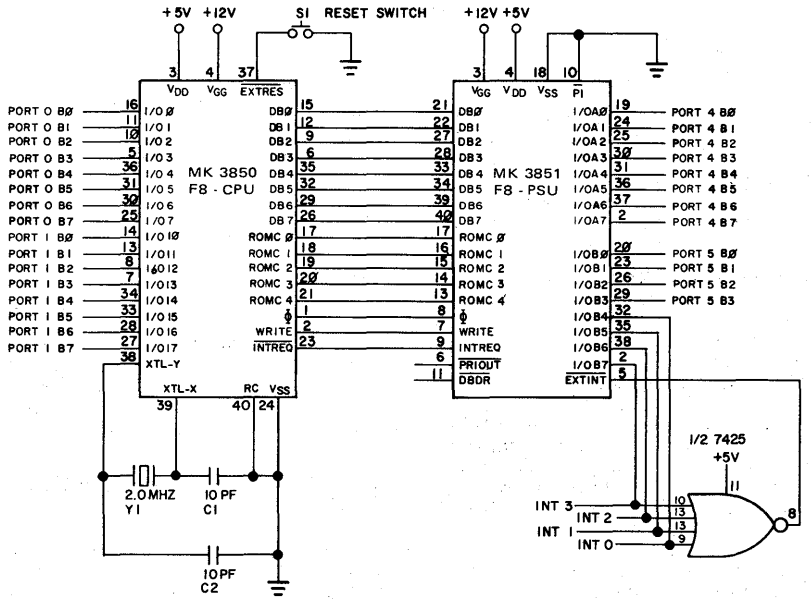


Figure 3

rupting the system. Figure 4 depicts this minimum F8 system, with four signals (INT0-INT3) capable of interrupting the system. The four external interrupting signals are defined active high and the presence of any one in the high state causes the output of the NOR gate to go low causing the interrupt. The interrupt service routine flowchart to locate the interrupting input is shown in Figure 5, with the actual program in Figure 6.

This service routine is entered with the interrupts automatically disabled at the CPU so that no further interrupts can occur until the interrupt is cleared by its service routine. The port containing the INT0-INT3 signals is loaded into the accumulator and tested to determine if bit 7 is low (a positive number). If bit 7 is low INT3 is active and the branch is taken to the service routine for INT3 (SERV3) (there is an inversion from the Port to the accumulator). If bit 7 is high a shift left one instruction is performed on the accumulator and it is again tested for bit 7 = 0 (bit 6 shifted). This process continues until all four of the interrupt lines have been tested. If an active interrupt bit has been found the proper service routine is branched to in order to service the active device and

EXPANSION OF INTERRUPT INPUTS IN A MINIMUM F8 SYSTEM



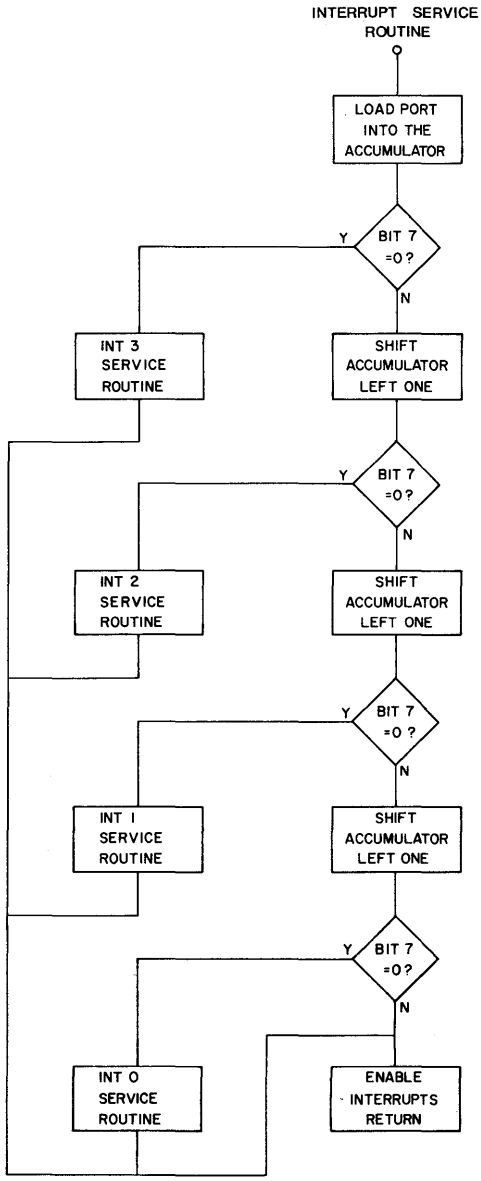
NOTE: MK 3870 Single Chip F8 will replace this two chip minimum system.

Figure 4

clear the interrupt. The routine ends by enabling the interrupts at the CPU and returning to the main program flow should no interrupt be found.

The additional time required to locate the active interrupt is a function of which interrupt is active due to the polling used. As shown in Figure 6, the additional delay to service interupts produced by polling varies from 15 μs for the highest priority device to 42 μs for the lowest priority device. To these times must be added the delays calculated earlier of 20 μs typical and 42 μs maximum which is required to get to the polling routine.

FLOWCHART OF INTERRUPT SERVICE ROUTINE



INTERRUPT SERVICE ROUTINE TO LOCATE INTERRUPTING DEVICE

CUMM μS	μS	8 INTSVC	INS	PORT	GET SIGNALS
INT 3	15	7	BP	SERV 3	INT 3 ACTIVE 7
		2	SL	1	NO, SHIFT LEFT
INT 2	24	7	BP	SERV 2	INT 2 ACTIVE 7
		2	SL	1	NO, SHIFT LEFT
INT 1	33	7	BP	SERV 1	INT 1 ACTIVE 7
		2	SL	1	NO, SHIFT LEFT
INT 0	42	7	BP	SERV 0	
		4	EI		ENABLE INTERRUPTS
		4	POP		RETURN

Figure 6

SINGLE CHIP MICROCOMPUTER

The MK 3870 Single Chip Microcomputer is the natural evolution of the F8 chip set. It will combine the functions of the 3850/3851 onto a single chip with the additions of another 1K bytes of ROM storage and an improved timer/interrupt structure. The techniques discussed in this application note apply also to the single chip F8 as it is software and hardware compatible with the multiple chip F8 family.

MOSTEK®

3870/F8 MICROCOMPUTER SUPPORT

Application Note

SUBROUTINE NESTING AND MULTIPLE INTERRUPT HANDLING

INTRODUCTION

The 3870 and F8 Microcomputer Families are quickly becoming recognized as a cost effective method of placing computing power into types of equipment which couldn't have justified computer control just a short time ago. The falling cost per computer function afforded by advances in Metal-Oxide Semiconductor-Large Scale Integration (MOS–LSI) has brought computer technology and techniques into areas where until now, mechanical controller's, random logic and relay logic predominated. The availability of a large number of Input/Output pins in the 3870 Microcomputer coupled with its minimum system configuration of just one device, makes it an ideal replacement for many previously used control devices. The purpose of this note is to discuss the use and implementation of subroutines and interrupts as they apply to programming an F8 based microcomputer system. The intent of this note is to discuss the use of subroutines and interrupts for the hardware designer who might not be totally familiar with the programming of a computer.

SUBROUTINES

A subroutine is a sequence of computer instructions or mnemonics which can be called or used in several portions of the computer program. The purpose of a subroutine is to reduce the total length of a computer program by consolidating in one portion of the program a sequence of instructions that are used in several different areas of the program. When this subroutine is required the program counter contents are replaced with the starting address of the subroutine. At the end of the subroutine the program is transferred back to the main body of the program.

Figure 1 depicts program flow when using subroutines. The main program calls a subroutine which causes the program counter to be loaded with the address of the subroutine. The last statement of the subroutine causes a return back to the main program flow by retrieving the saved program counter value, forcing a return to the main program flow. The subroutine is called again any place in the main program flow where the sequence of instructions contained in the subroutine is required. Every time the subroutine is called a savings in program length (and ROM size) equal to the length of the subroutine (minus three) is realized compared to a program which doesn't use subroutines. Many times a subroutine will call another subroutine resulting in what is referred to as nested or multi-level subroutines. Nested subroutines in an F8 system will be discussed in this note.

INTERRUPTS

Interrupts are used in a microcomputer system to make it responsive to the device it is controlling.

PROGRAM FLOW WHEN USING SUBROUTINES

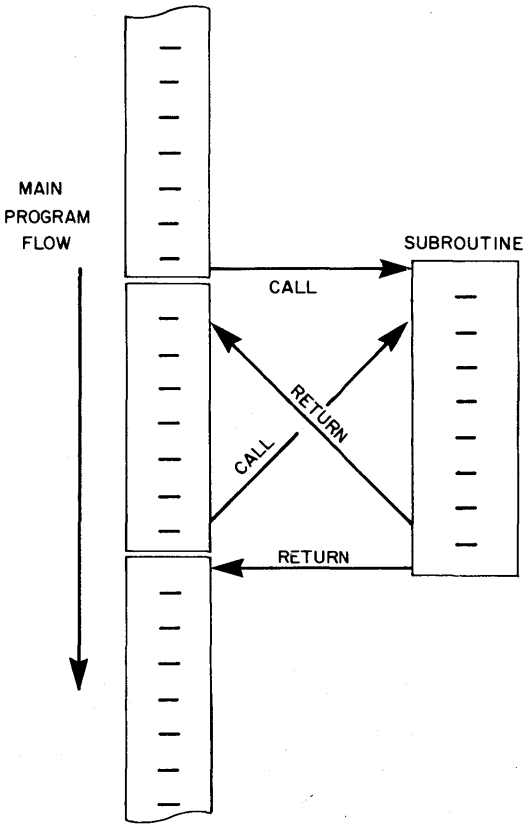


Figure 1

By interrupting the microcomputer the I/O device can signal its requirement for attention or service by the microcomputer. As in the case of the subroutine, the interrupt can divert the main program flow to a sequence of instructions called the Interrupt Service Routine (See Figure 2). This routine either inputs or outputs data to the device being controlled. At the end of this service routine the program counter value at the time of the system interrupt is retrieved from a temporary register and reloaded into the program counter to cause a return to the main

PROGRAM FLOW WHEN INTERRUPTED

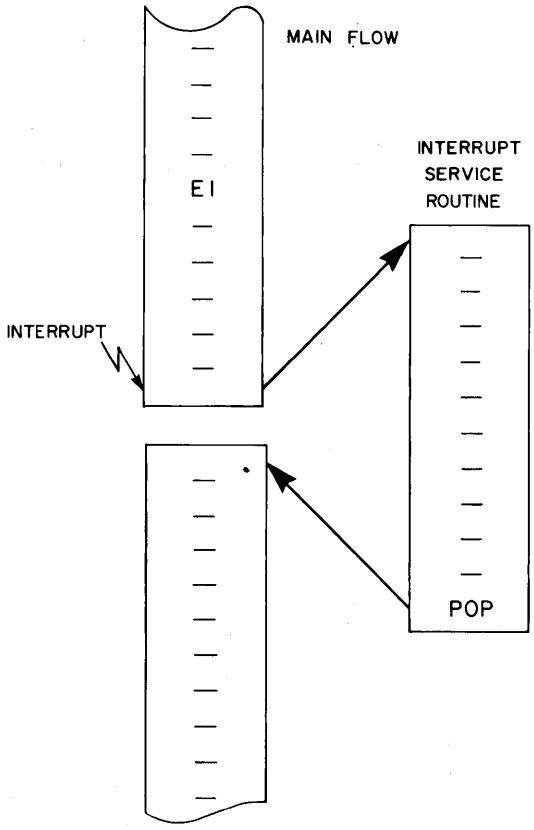


Figure 2

program flow. Interrupts, like subroutines, can be nested because an Interrupt Service Routine could be interrupted by a higher priority device or an Interrupt Service Routine may call a subroutine, which in either case causes nesting.

The F8 instructions which are used to transfer program flow to or from subroutines or interrupts are shown in Figure 3. The Program Counter (P0) holds the address of the next instruction to be executed by the microcomputer while the Stack Register (P)* is a temporary storage location for the Program Counter. In addition two pairs of registers in the Scratchpad have been designated K and Q with instructions that link them to P0 and P. The instructions that link and affect these registers are the following:

(a) Call to subroutine immediate –PI–an instruction which causes the next two bytes in the program to be loaded into the Program Counter (P0) in order to transfer control to a subroutine and saves the old program counter value (return address) in the Stack Register (P).

(b) Call to subroutine–PK–an instruction which causes the contents of the K register to be loaded into the Program Counter while the Program Counter is saved in the Stack Register.

(c) Return from subroutine–POP–an instruction used at the end of a subroutine or interrupt service routine to load the Stack Register back into the Program Counter to return program flow back to the main program. The previous value of the Program Counter is overwritten and lost.

(d) Load–LR P,K–a pair of instructions which
LR K,P
allows the transfer of the Stack Register (P) to the K register in the Scratchpad or vice versa. This switch is useful to save P in preparation for a subroutine or interrupt.

(e) Load – LR P0,Q which allows the transfer of the Program Counter (P0) to the Q register in the Scratchpad.

The following sections of this note will discuss the use of these instructions and registers as well as the general F8 architecture to handle Subroutines, Interrupts and the tradeoffs in doing so.

SUBROUTINES AND/OR INTERRUPTS UP TO TWO LEVELS

Many applications can be handled by two levels of subroutines and/or interrupts. Two levels means that only two return addresses need be saved, which can be handled easily by registers within the F8 for this purpose. The calling of subroutines is under control of the programmer and thus only the return addresses need be saved as other registers (such as the Data Counter) can either be saved by the calling or the called routines if the registers are needed by the subroutine. Interrupts are under control of the programmer only to the extent that they can be masked or enabled. Assuming interrupts are enabled, upon entry to an Interrupt Service Routine, it may not be known which registers in the CPU contain data which cannot be overwritten. In this case these registers should be stored in the scratchpad during the Interrupt Service Routine and be restored before exiting this routine. Examples of using the ISAR to store CPU registers in a push down stack are given in this note but in many cases the programmer will tailor the status saving routine for the specific circumstances of his sytem design (by using specific Scratchpad Registers to save CPU Registers).

Figure 4 shows the instructions usually used to call a subroutine (one level deep) in an F8 system. SUBA1 is the symbolic name of the two byte address of the subroutine and PI causes the return address (XXXX) to be saved in P. POP reverses the procedure at the end of the subroutine causing P0 to be reloaded with the address saved in P causing the program flow to return to the next instruction in the main flow (XXXX). Response to an interrupt from the main flow is similar to this example except that the interrupt causes a path similar to 1 to the Interrupt Service Routine with the address (vector) being supplied by the interrupting device and loaded into P0.

To call a second subroutine or to respond to an interrupt from SUBA1 the instructions in Figure 5 could be used. In this case PI SUBA1 transfers the

* Earlier versions of F8 literature labled the Program Counter PC0 and the Stack Register PC1.

F8 REGISTERS USED IN SUBROUTINES AND INTERRUPTS

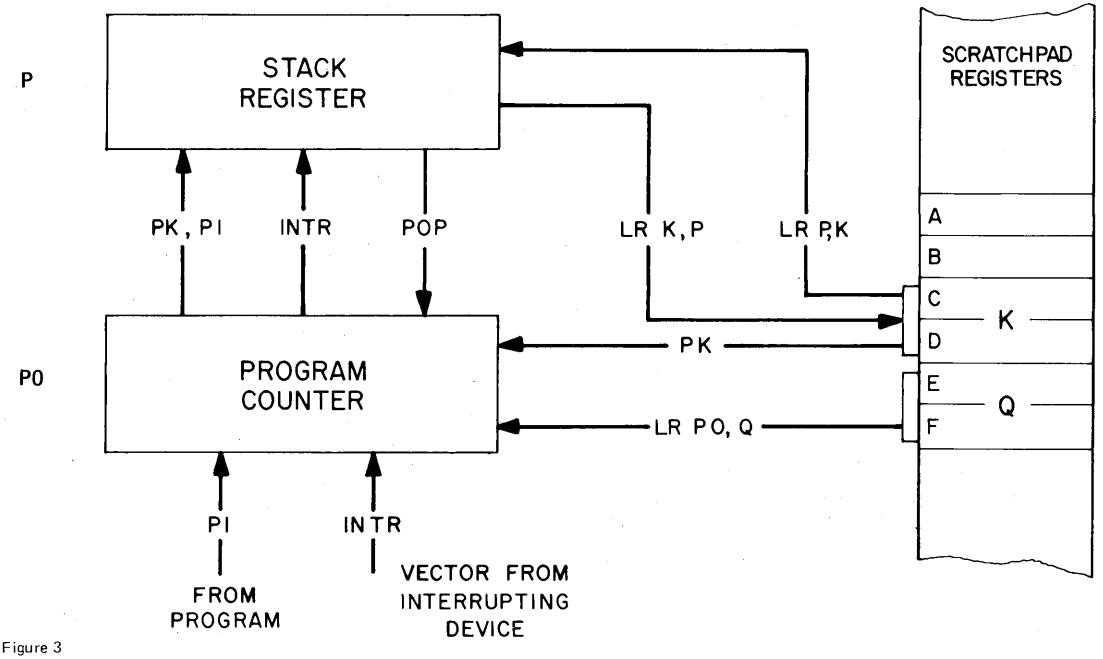


Figure 3

ONE LEVEL SUBROUTINES OR INTERRUPTS

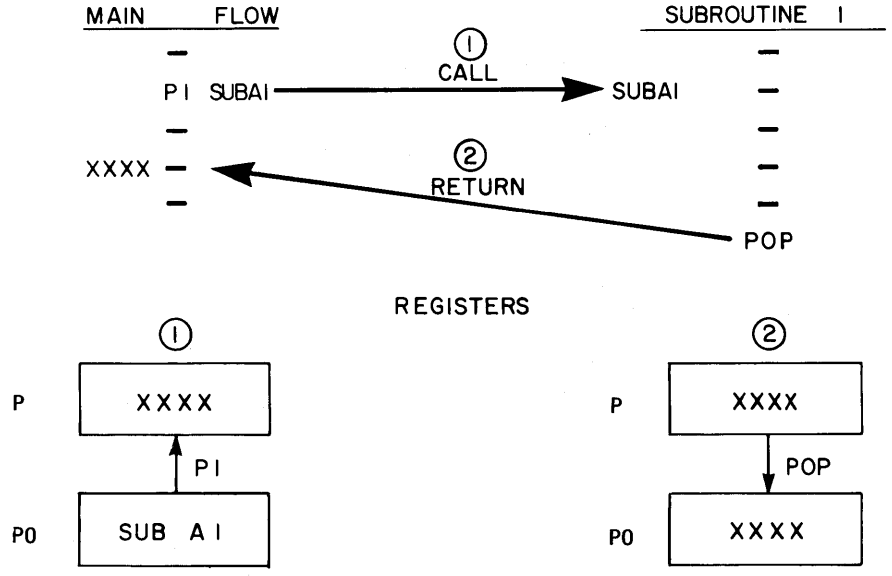


Figure 4

program flow to SUBA1 while saving the return address (XXXX) in the Stack Register (P). Subroutine 1 now transfers P to K in preparation for another subroutine or an interrupt (note that if an interrupt occurs during the PI SUBA1, LR K,P sequence it will not be serviced until after the LR K,P instruction because PI is privileged). Subroutine 2 is called by PI SUBA2 which saves YYYY in P which was just vacated. Program flow transfers to Subroutine 2 and the POP instruction reloads P0 with YYYY from P. At the end of Subroutine 1 the return address is now in K so a PK is used to load XXXX into P0, thereby returning to the main program. Note that LR K,P followed by POP could have been used in place of the PK instruction, but would be 1 byte longer.

Three levels of subroutines or interrupts can be handled by using the Q register to save a return address. Figure 6 shows programming with three levels of subroutines (three levels of interrupts would be handled in the same manner). The first subroutine is called from the main program and the return address is saved in the Stack Register P. At the beginning of SUB1, P is transferred to the K register in preparation for the second subroutine call or interrupt. This second call uses the just vacated P register for storage of the return address to SUB1. Upon entering SUB 2 both P and K contain valid return addresses so that interrupts must be disabled while the contents of K register are moved to the Q register and the contents of P are moved to the K register. Once P is clear, interrupts can be enabled by the use of the EI instruction, allowing a third level of subroutine nesting (as shown in Figure 6) or an interrupt. The return from SUB3 is accomplished by executing the POP instruction which loads the Program Counter with the value RTN2 from the Stack Register P. During the first portion of SUB2, P was moved to K so that a PK instruction will load the Program Counter with RTN1 from the K register. The return address for SUB1 (RTN) was moved to the Q register during the first portion of SUB2 and can be transferred to the Program Counter by the execution of LR P0,Q instruction.

MULTILEVEL INTERRUPTS OR SUBROUTINES

At a minimum when using the F8 in a system with greater than 3 levels of interrupts or subroutines a consistent method of placing return address into the scratchpad must be used to allow their recovery. In many cases it will be desirable to stack more registers than just the return addresses. Previous examples have shown 3 levels deep with the three return addresses in P, K, and Q registers. Any further nesting would destroy either P, K, or Q so the technique to be described is to move K into the scratchpad to make room for another level.

Figure 7 shows a generalized subroutine which automatically transfers P to K and then K into the scratchpad registers. The routines in Figure 7 assume that ISAR (Indirect Scratchpad Address Register) has been initialized at an odd value, probably H'3F'which

TWO LEVEL SUBROUTINES OR INTERRUPTS

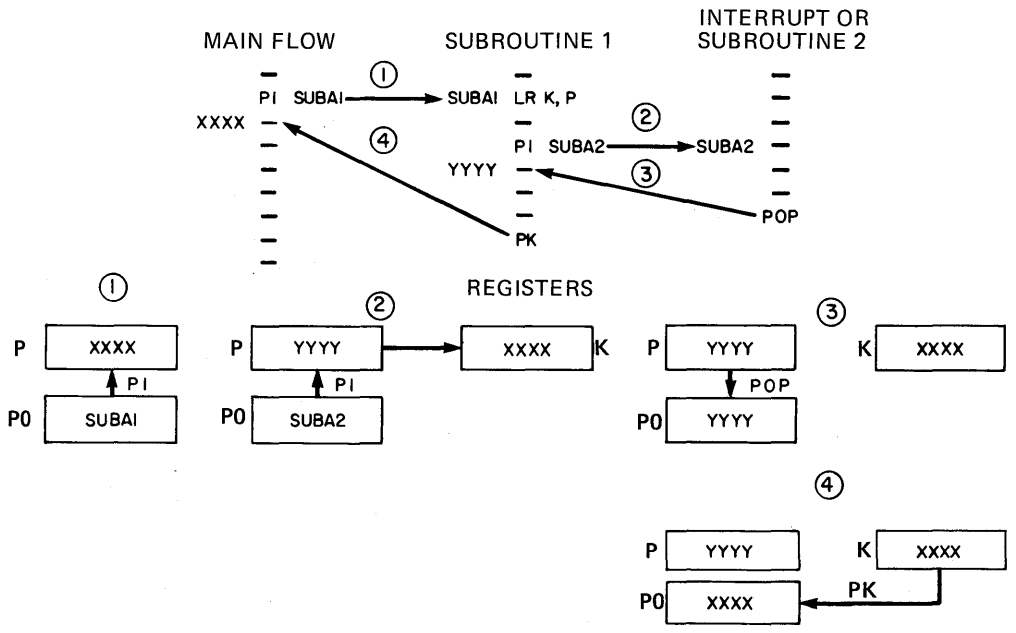


Figure 5

THREE LEVELS OF SUBROUTINES OR INTERRUPTS

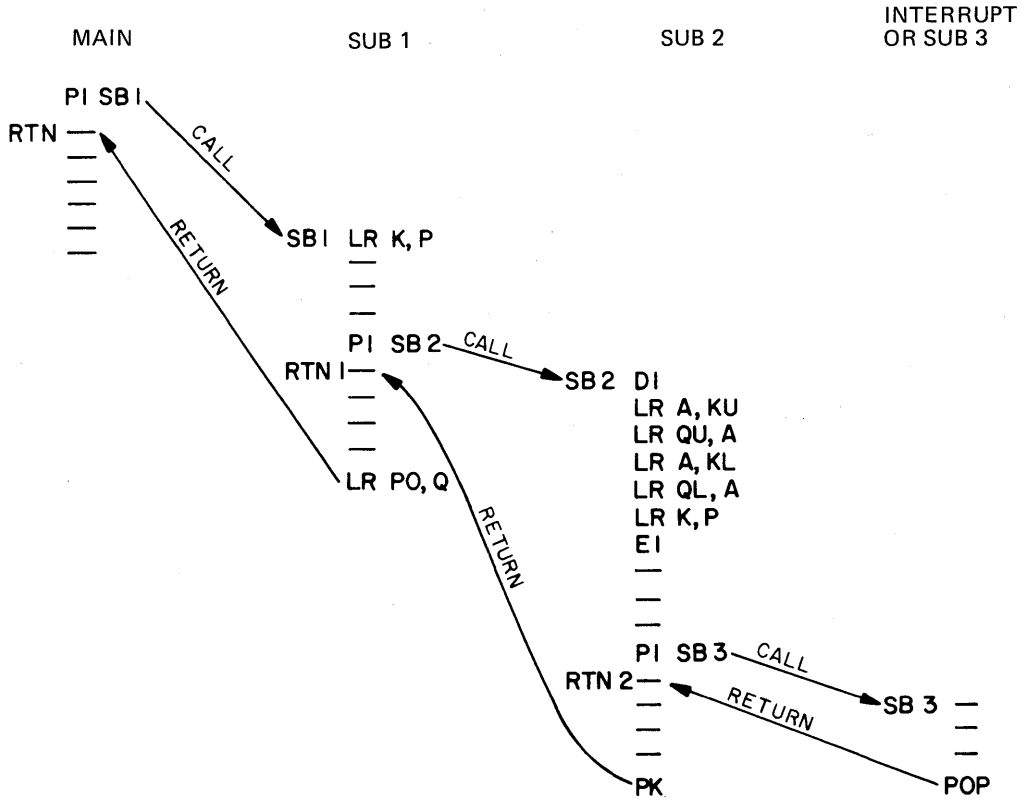


Figure 6

STACKING OF ACCUMULATOR AND STATUS REGISTERS

```
SUBX   DI
       LR K,P      }  K to
       PI PSHK     }  Stack
                   }
               ____}
               ____}  Body of
               ____}  Subroutine
                   }
       PI POPK     }  K from
                   }  Stack
       PK          }
```

TO STACK

```
PSHK   LR A, KL
       LR D, A     KL to stack
       LR A,KU
       LR S,A      KU to stack
       LR A,IS
       AI H'FF'    Decrement ISAR
       LR IS, A
       LR W,J
       POP         Return
```

FROM STACK

```
POPK   LR A, IS
       INC         Increment ISAR
       LR IS,A
       LR A,I
       LR KU,A     Recover KU
       LR A,S
       LR KL,A     Recover KL
       LR W,J
       POP         Return
```

Figure 7

is the top of the scratchpad registers. The odd starting value is required to insure that ISAR is not pointing to an 8 byte buffer boundary when the LR D, A instruction is executed. (The LR D,A instruction loads the accumulator from the scratchpad location pointed to by ISAR and then does a modulo 8 decrement of ISAR. This means that only the lower three bits of ISAR are decremented resulting in an 8 byte range for these auto decrementing and autoincrementing instructions which does not allow crossing of page boundarys. By initializing ISAR at an odd value, every time the LR D,A instruction is executed ISAR will be odd and therefore will not have to cross page boundaries which are even.) The decrement from even values is accomplished by loading ISAR into the accumulator and adding hexidecimal FF to it, which results in an 8 bit decrement of ISAR's contents. PSHK then moves the contents of K onto the stack and leaves ISAR pointing to the next empty location thus implementing a push-down stack. When in the body of the subroutine both P and K are clear, allowing a call to another subroutine of this format or the enabling of interrupts to allow interrupting out of this subroutine (return address would be held in P). POPK is called to recover the subroutine return address and place it in the K register. Note here that the 8 bit increment (INC) is done first to cross the page boundary and point to the last byte stored on the stack. ISAR is used to pull the return address off the stack and place it in the K register. The PK instruction reloads P0 and program flow is returned to the calling routine.

If interrupts are enabled during the body of the program they must be disabled during execution of POPK because this routine is using both P and K registers. The LR W,J instruction allows the user the option to control whether interrupts will be enabled or disabled after execution of PSHK or

SUBROUTINE OR INTERRUPTS NESTED 4 LEVELS DEEP

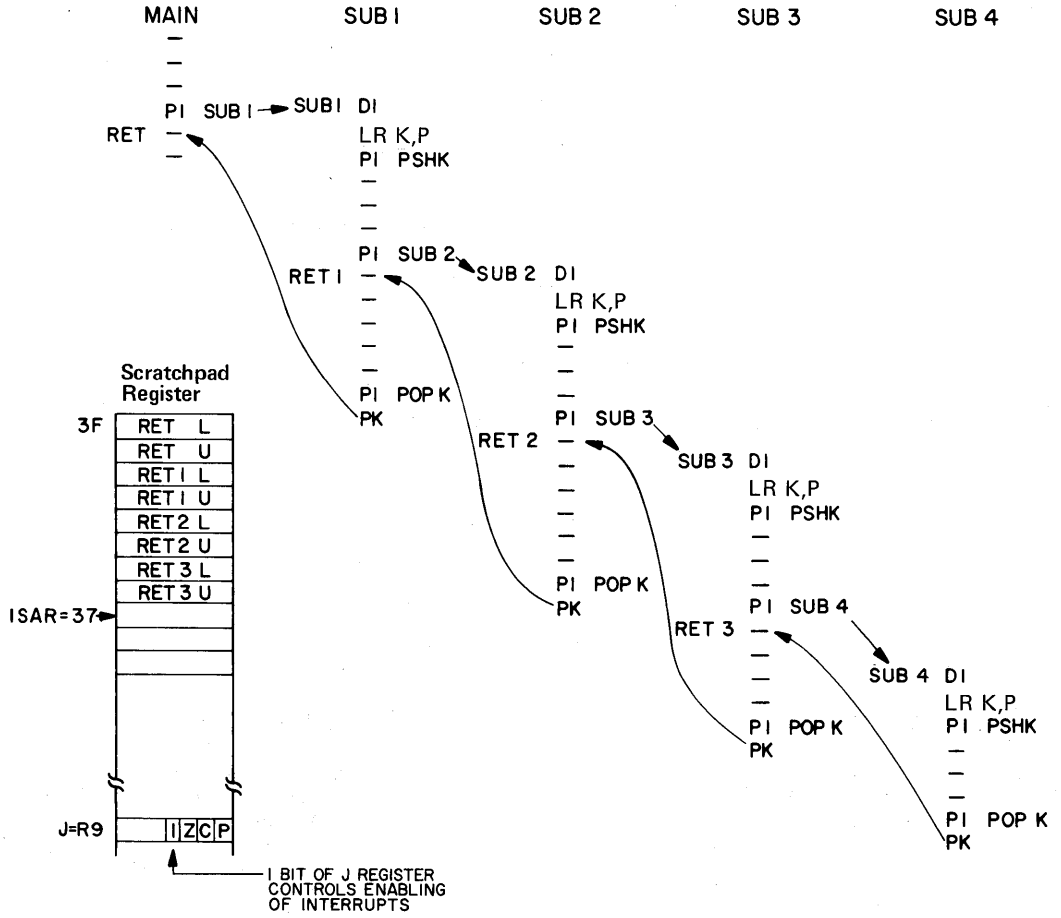


Figure 8

POPK by setting or clearing bit 4 in the J register. Thus if interrupts are desired during a subroutine the following instructions would be used to call SUBX.

MAIN PROGRAM

```
LR  A,IS
LR  0,A    SAVE ISAR IN R0
LI  H'09'
LR  IS,A   POINT ISAR TO J
LR  A,S    GET J INTO A
OI  H'10'  SET 1 BIT, INTERRUPTS ENABLE D
LR  S,A    A INTO J
LR  A,0    RESTORE ISAR
LR  IS,A
PI  SUBX   CALL SUBROUTINE
```

If interrupts are not desired during the nesting of subroutines the software can be simplified as follows: Assuming interrupts are disabled instructions DI and LR W,J can be deleted from the PSHK and POPK routines. Also the above calling sequences will not be required because bit 4 in the W register will already be clear.

Figure 8 shows the effects of PSHK used to save the return address on the stack. Note that the Stack Pointer (ISAR) is pointing to the next empty location on the stack and that bit 4 of the J register is controlling whether interrupts are enabled or not through the use of the LR W,J instruction in PSHK and POPK. Bit 4 of the J register is used to control whether interrupts are enabled or not in order to allow two different callers to use this subroutine. If one of the callers was in an interrupt driven portion of the program, bit 4 of the J register could be set to allow interrupts upon returning from the subroutine. If one of the other callers of the subroutine did not want interrupts enabled bit 4 of the J register could be cleared so that no interrupts would be recognized upon returning from the subroutine.

At the end of each subroutine, PI POPK followed by PK will be executed to unload the stack and return program flow back to the correct return address. Note also that interrupts will be allowed at any time except during the execution of the PSHK or POPK routines, their return address being stored in the P register. If the interrupting device service routine needs to call a subroutine, P will have to be pushed onto the stack using the methods previously described.

In many cases it may be desirable to save the major registers within the CPU whenever an interrupt is serviced. Saving registers on the stack frees up all of the computing power of CPU for use by the Interrupt Service Routine. Saving the registers upon the stack rather than in direct scratchpad locations makes the subroutine or interrupt service routine re-entrant (i.e., the routine calls itself without destroying scratch locations). The same philosophy as before can be used to save accumulator and status register on the stack (see Figure 9). Other registers within the machine could be saved using the technique; however, they must be pushed in pairs in order to leave ISAR pointing to an odd register location (since K, DC and DC1 are all 16 bit registers this should not be a limitation).

STACKING OF ACCUMULATOR AND STATUS REGISTERS

```
PSHAW   LR  D,A     Accm to stack
        LR  A,J
        LR  S,A     J Reg to stack
        LR  J,W     W reg to J reg
        LR  A,IS    ISAR to A
        AI  H'FF'   Decrement A
        LR  IS,A    A to ISAR

POPAW   LR  A, IS   ISAR to A
        INC         Increment A
        LR  IS,A    A to ISAR
        LR  W,J     J reg to W
        LR  A,I     J from stack
        LR  J,A     into J reg
        LR  A,S     Accm from stack
        POP         Return to caller
```

Figure 9

CONCLUSION

This application note has discussed a general method of handling subroutines and interrupts in an F8 system. Many applications for which the F8 is suited will have minimal subroutines or a minimum number of interrupts so that the internal P and K registers can be used to hold return addresses.

In the cases where deeply nested subroutines or multiple interrupts must be handled a push-down stack can be created in the scratchpad registers or external memory. Software routines were discussed to save return addresses in this stack as well as methods to save the general purpose registers. The user has the option in a F8 system to stack only what is necessary to accomplish his design goals in an optimum manner.

1215 W. Crosby Rd. • Carrollton, Texas 75006 • 214/242-0444
In Europe, contact: MOSTEK GmbH, TALSTR. 172,7024
Filderstadt-1, West Germany • Tele: (0711) 701096

PRINTED IN USA August 1978
Publication No. 79602