FAIRCHILD
A Schlumberger Company

F3850

Central Processing Unit (CPU)

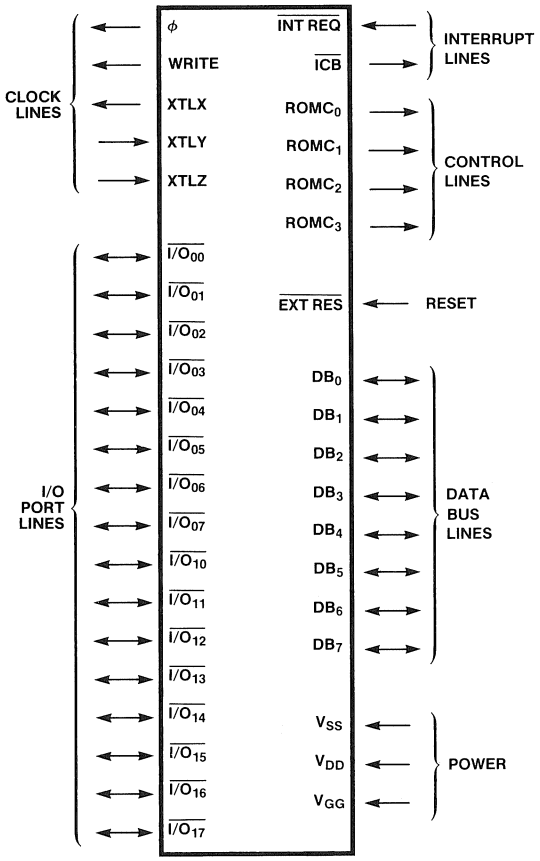
Microprocessor Product

Description

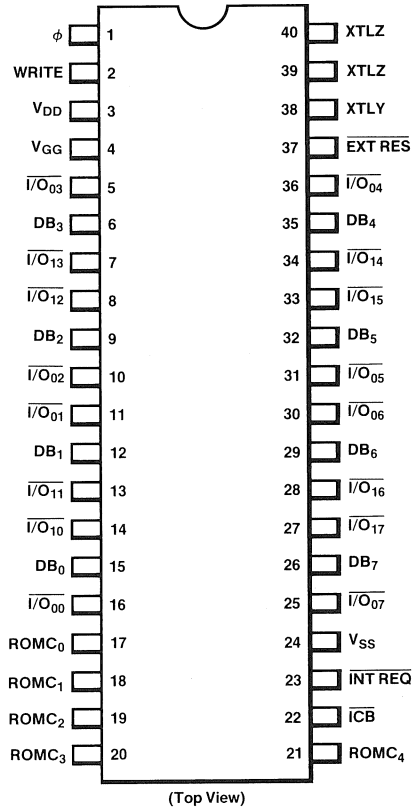
The Fairchild F3850 is the Central Processing Unit (CPU) for the F8 8-Bit Microprocessor family. The F3850 contains more than 70 instructions in its instruction set and operates on 8-bit units of information.

- **N-channel Isoplanar MOS Technology**
- **2 μs Cycle Time**
- **64-Byte Scratchpad on the CPU Chip**
- **Two Bidirectional, 8-Bit I/O Ports, with Output Latches**
- **8-Bit Arithmetic and Logic Unit, Supporting Both Binary and Decimal Arithmetic**
- **Interrupt Control Logic**
- **Power-on Reset Logic**
- **Clock Generation Logic Within the CPU Chip, With Crystal and External Clock Generation**
- **More Than 70 Instructions**
- **+5 V and +12 V Power Supplies**
- **Low Power Dissipation (Typically Less Than 330 mW)**

Signal Functions



Connection Diagram



(Top View)

Fairchild Camera and Instrument, Microprocessor Division
3420 Central Expressway, Santa Clara, CA 95051
(408) 773-1000 TWX MICPSNTA

Device Organization

The logical organization and pins for the F3850 CPU are illustrated in *Figure 1.*

Arithmetic and Logic Unit

The arithmetic and logic unit (ALU) provides all data manipulating logic for the F3850. It contains logic that operates on a single 8-bit source data word or combines two 8-bit words of source data to generate a single 8-bit result. Additional information is reported in status flags, where appropriate.

Operations performed on two units of source data include addition, compare, and the Boolean operations (AND, OR, Exclusive-OR). The two sources are input to the ALU through the left and right multiplexer buses; the result is placed on the result bus.

Operations performed on a single 8-bit unit of source data include complement, increment, decrement, shift right, shift left, and clear. The source is input to the ALU through either the left or right multiplexer bus; the result is placed on the result bus.

Instruction Register

The CPU contains registers for storing various types of data. The instruction register holds an 8-bit code, which defines the operations to be performed by the CPU.

The contents of the instruction register are decoded by control unit logic, which generates signals to enable specific sequences of logic operations within the CPU chip. In response to the contents of the instruction register, the control unit also generates five signals, $ROMC_0$ through $ROMC_4$, that control operations throughout the microprocessor system.

Accumulator

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

Scratchpad and ISAR

The scratchpad provides 64 8-bit registers that may be used as general-purpose RAM memory (see *Figure 2*).

Figure 2 F8 Programming Model

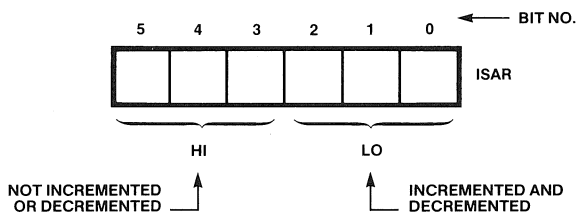
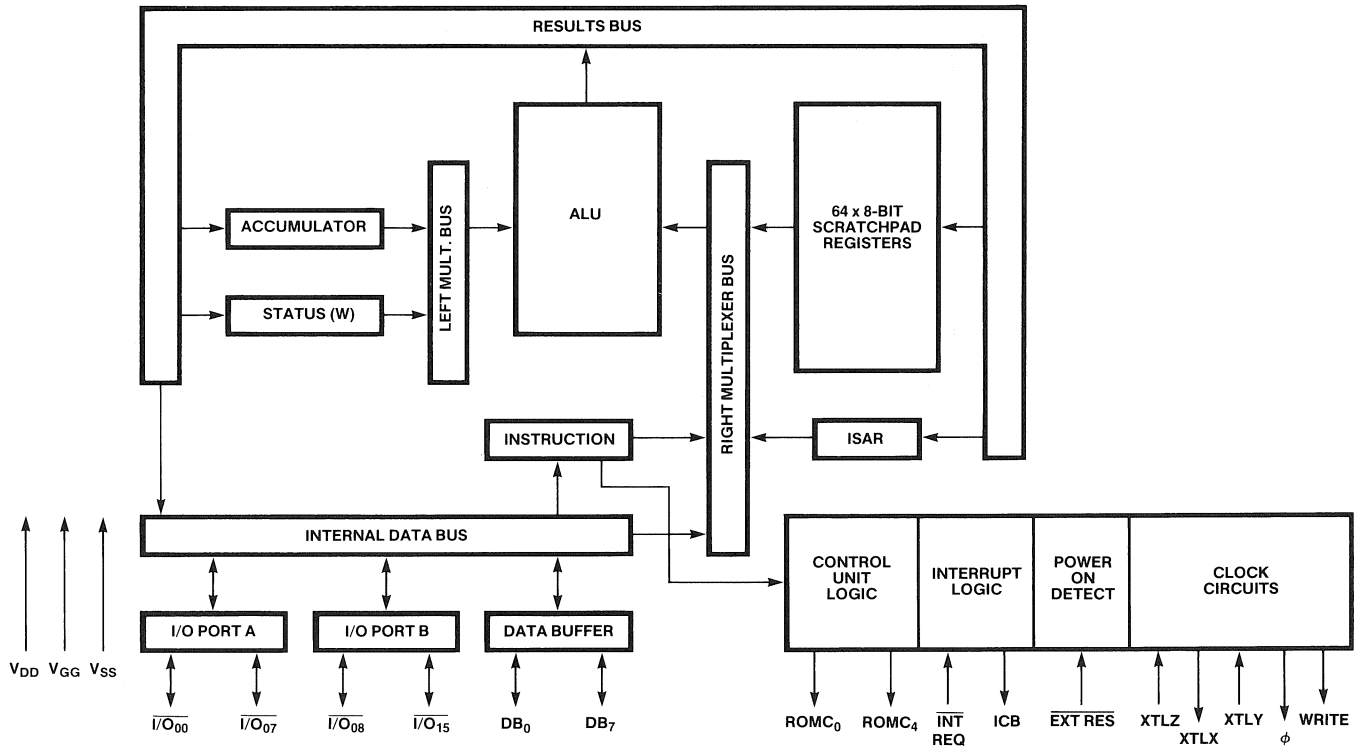


Figure 1 F3850 CPU Logical Organization

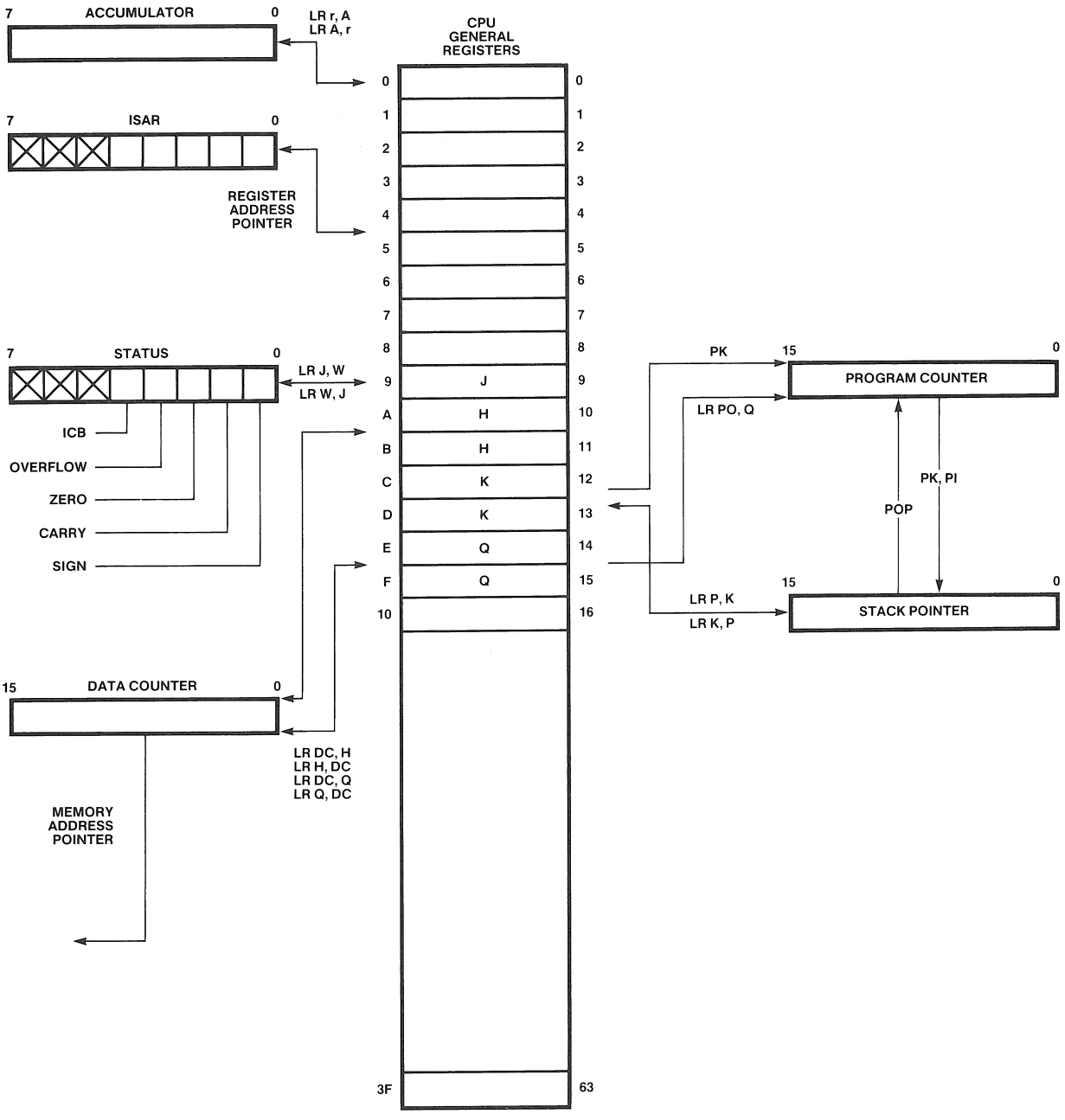


The indirect scratchpad address register (ISAR) is a 6-bit register used to address the 64 scratchpad registers.

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

The ISAR may be visualized as holding two octal digits, HI and LO, as illustrated in *Figure 3.* This division of the ISAR is important, since a number of instructions increment or decrement the con-

Figure 3 ISAR Register



tents of the ISAR, when referencing scratchpad bytes through the ISAR. This makes it easy to reference a buffer consisting of contiguous scratchpad bytes. However, only the low-order octal digit (LO) is incremented or decremented; thus ISAR is incremented from O'27'* to O'20', not to O'30'. Similarly, ISAR is decremented from O'20' to O'27', not to O'17'. This feature of the ISAR is very useful in that it greatly simplifies many program sequences.

Selected scratchpad registers are reserved for direct communication with other registers within the F8 system, as illustrated in *Figure 4.*

Scratchpad register 9 (O'11') is used as temporary storage for the CPU status register (W register). Scratchpad registers 10 through 15 (O'12' through O'17') communicate directly with data and program memory address registers that are maintained on the F3851, F3852, and F3853 chips. *Figure 4* identifies the data transfers that can be implemented by executing a single F8 instruction. For example, the illustration:

W register of F3850 CPU ↔ J

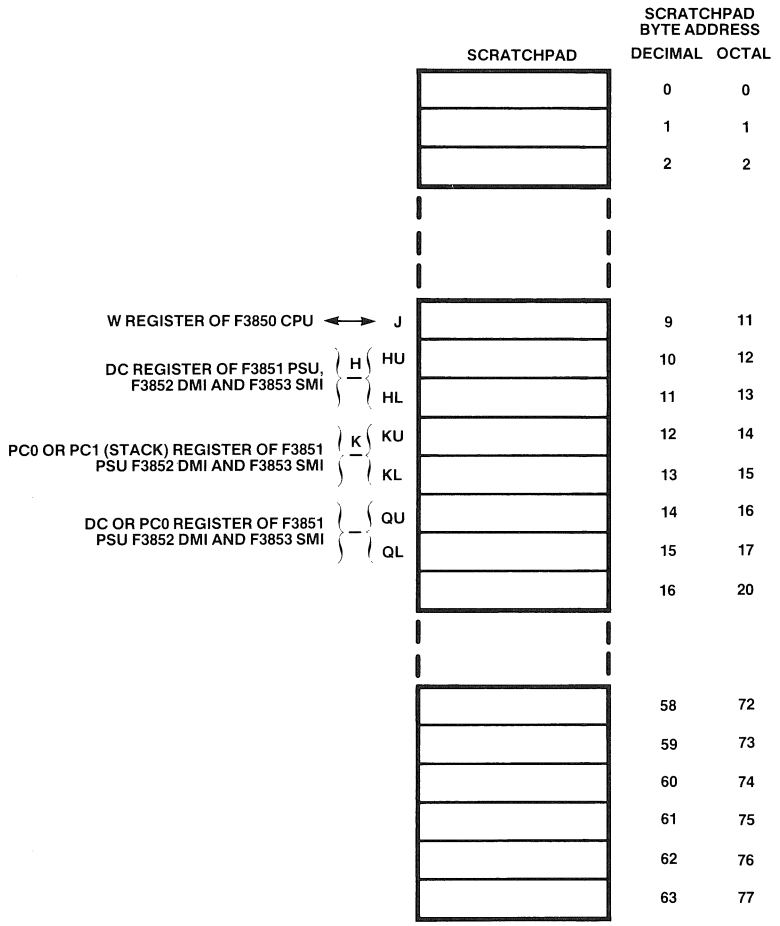
means that a single instruction can move the contents of the W (or status) register to scratchpad register 9 (J register). Another single instruction can move data in the opposite direction.

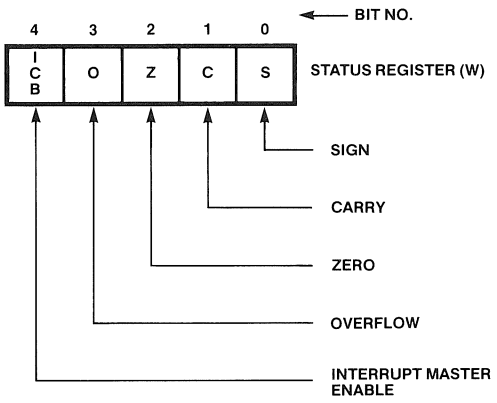
*The notation O'nn' represents an octal number.

Status Registers

The status (W) register holds five status flags. *Table 1* summarizes the way each flag is used. Note that status flags are selectively modified following execution of different instructions. See the "Instruction Execution" section for a discussion of the way individual F8 instructions modify status flags.

Figure 4 F3850 CPU Scratchpad Registers





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.

Table 1 Summary of Status Bits

$$\text{OVERFLOW} = \text{CARRY}_7 + \text{CARRY}_6$$
$$\text{ZERO} = \overline{\text{ALU}_7}\ \overline{\text{ALU}_6}\ \overline{\text{ALU}_5}\ \overline{\text{ALU}_4}\ \overline{\text{ALU}_3}\ \overline{\text{ALU}_2}\ \overline{\text{ALU}_1}\ \overline{\text{ALU}_0}$$
$$\text{CARRY} = \text{CARRY}_7$$
$$\text{SIGN} = \overline{\text{ALU}_7}$$

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; e.g.:

	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 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 a carry 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 a carry out of bits 6 and 7. The simplification of signed binary arithmetic is described in the *F8 and F3870 Guide to Programming*; examples are presented below:

	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 1 1 0 0 1 0 0

There is a carry out of bit 6 and a carry 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 is set to 1; interrupts are disallowed when the ICB is reset to 0.

Control Unit

The control unit decodes the contents of the instruction register and generates two sets of control signals. These signals are transparent to the user.

Five control signals (ROMC_0 through ROMC_4) are output by the control unit to identify operations that other chips of the F8 family must perform. These signals are described in the "ROMC Signals" section.

Interrupt Logic

This logic handles the interrupt requests. For a complete description refer to the "Interrupt" discussion within the "Instruction Execution" section.

Power on Detect

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

Signal Descriptions

The F3850 input and output signals are described in *Table 2.*

Table 2 F3850 Signal Descriptions

Mnemonic	Pin No.	Name	Description
Clock			
ϕ	1	Clock	These output signals drive all other devices in the F8 family.
WRITE	2	Write	
XTLX	39	Crystal Clock	The XTLX output signal is used when generating the system clock in the crystal mode (with the XTLY and XTLZ signals).
XTLY	38	External Clock	The XTLY input signal is used with the XTLX signal when generating the system clock in the crystal mode, and is also used for operating in the external clock mode.
XTLZ	40	Crystal Clock	This input signal must be grounded for crystal clock or external clock.
I/O Port			
$\overline{I/O}_{00}$–$\overline{I/O}_{07}$	16, 11, 10, 5, 36, 31, 30, 25	I/O Port Zero	These bidirectional signals are ports through which the CPU communicates with logic external to the microprocessor system.
$\overline{I/O}_{10}$–$\overline{I/O}_{17}$	14, 13, 8, 7, 34, 33, 28, 27	I/O Port One	
Interrupt			
$\overline{ICB}$	22	Interrupt Control Bit	The $\overline{ICB}$ output signal indicates whether or not the CPU is currently ignoring the $\overline{INT\ REQ}$ line. If the $\overline{ICB}$ signal is low, the CPU responds to interrupt requests; if the $\overline{ICB}$ signal is high, the CPU ignores interrupt requests.
$\overline{INT\ REQ}$	23	Interrupt Request	This input line is used to signal the CPU that an interrupt is being requested.The F3851 PSU, F3861 and F3871 PIOs, and F3853 SMI devices contain logic to initiate interrupt requests by pulling the $\overline{INT\ REQ}$ signal low. The CPU acknowledges interrupt requests by outputting the appropriate ROMC signals.
Control			
$ROMC_0$–$ROMC_4$	17—21	Control	The ROMC output signals control logic operations for other devices in the F8 family. These signals assume a state early in each machine cycle and hold that state for the duration of the cycle. Refer to the "Instruction Execution" section for further discussion and a summary table of the ROMC interpretation by CPU logic.

Table 2 F3850 Signal Descriptions (Continued)

Mnemonic	Pin No.	Name	Description
Reset			
$\overline{EXT}$ $\overline{RES}$	37	External Reset	This input signal can be used to externally reset the system. When the line is pulled low, a program originating at memory address 0 is executed.
Data Bus			
DB_0–DB_7	15, 12, 9, 6, 35, 32, 29, 26	Data Bus	These eight bidirectional signals are data bus lines that link the F3850 CPU with all other F8 devices in the system. They are multiplexed lines used to transfer data and addresses.
Power			
V_{DD}	3	Power Supply	Nominal +5 Vdc
V_{GG}	4	Power Supply	Nominal +12 Vdc
V_{SS}	24	Ground	Common power and signal return

Clock Circuits

A unique feature of the F8 microprocessor is that clock logic forms an integral part of the F3850 CPU chip. The F3850 CPU offers two methods of generating a system clock: 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. Capacitors C_1 and C_2 should be approximately 15 pF. The characteristics of the crystal used in this mode of clock generation are summarized as:

Frequency: 1 to 2 MHz, typical AT cut
Mode of Oscillation: Fundamental
Operating Temp. Range: 0°C to +70°C
Drive Level: 10 mW
Frequency Tolerance: $f_0 = 1$ or 2 MHz ±1000 ppm @ C_L = 20 pF

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 F3850 CPU may receive its timing from a master F3850 CPU by having the master ϕ output drive the slave XTLY input.

Figure 5 Crystal Mode Clock Generation

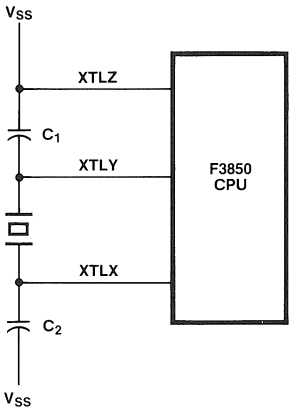


Figure 6 External Mode Clock Generation

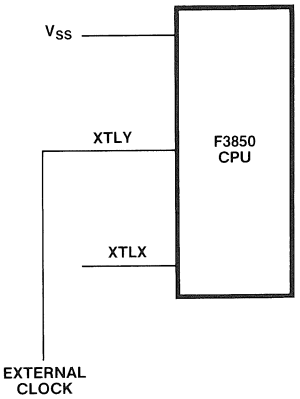


Figure 7 illustrates the timing characteristics of the clock signal needed for external mode clock generation and the timing characteristics of the ϕ and WRITE signals generated by the CPU.

Timing Signal Outputs
In response to the three clock mode inputs, the F3850 CPU outputs two timing signals: clock signal ϕ and instruction cycle control signal WRITE. As shown in *Figure 7*, ϕ is the signal used to synchronize the entire microprocessor system. The WRITE signal defines the duration of each machine cycle. Refer to the "Instruction Execution" section. Parameters and specifications for the timing signals are detailed in the "Timing Characteristics" section.

Instruction Execution

The F3850 CPU logic controls instruction execution through the ϕ and WRITE timing signals, plus the five ROMC control lines. Devices external to the F3850 CPU must respond directly to these signals.

Instruction Cycle
All instructions are executed in cycles that are timed by the trailing edge of WRITE.

There are two types of instruction cycle: the short cycle, which is four ϕ periods long, and the long cycle, which is six ϕ periods long. The long cycle is sometimes referred to as 1.5 cycles. *Figure 7* illustrates the short cycle (PW_S) and the long cycle (PW_L). Note that WRITE high appears only at the end of an instruction cycle.

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

ROMC Signals
The CPU logic uses the five ROMC signals to identify operations that devices must perform during any instruction cycle. The 32 possible ROMC states are described in the "ROMC Signal Functions" section. The state of the ROMC signals and the operation they identify last through one instruction cycle.

The general distribution of logic among devices of the F8 family and general data movements associated with instruction execution are given in the *F8 and F3870 Guide to Programming.*

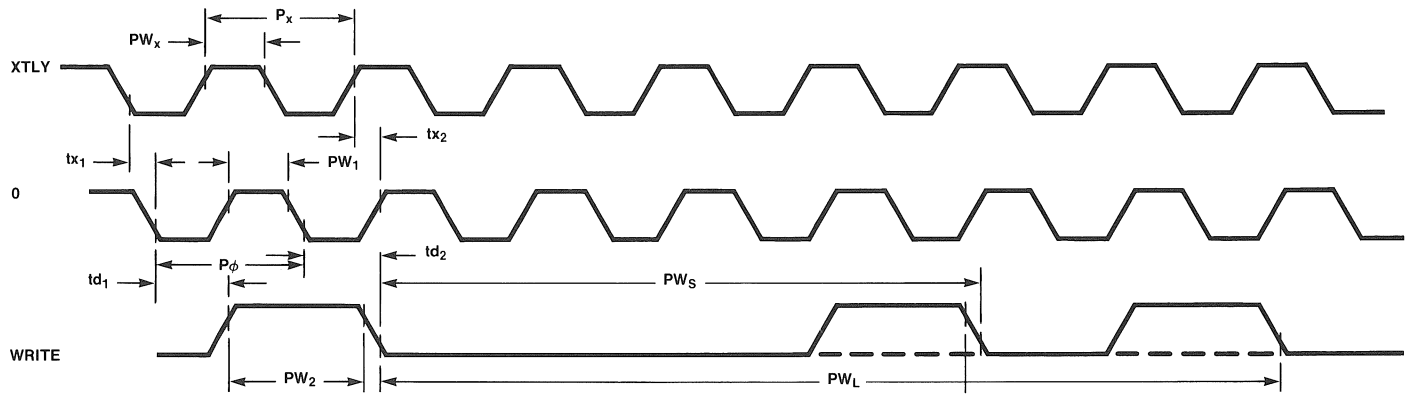
Memory addressing logic is located on the F3851 Program Storage Unit (PSU), the F3852 Dynamic Memory Interface (DMI), and the F3853 Static Memory Interface (SMI) devices. Each of these devices contains registers to address programs (PC0 and PC1) or data (DC0 or DC1). The F3851 PSU does not have a DC1 register.

Unlike other microprocessors, the F3850 CPU does not output addresses at the start of memory access sequences; a simple command to access the memory location addressed by PC0 or DC0 is sufficient, since the device receiving the memory access command contains PC0 and DC0 registers. (The PC1 and DC1 are buffer registers for PC0 and DC0.)

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

There will be as many PC0 and DC0 registers in a microcomputer system as there are PSU, DMI, and SMI devices; the ambiguity of which unit will respond to a memory read or write command is resolved by ensuring that all PC0 and DC0 registers contain the same information at all times. Every PSU, DMI, and SMI device

Figure 7 Clock Generation Timing Signals



has a unique address space, i.e., a unique block of memory addresses within which it responds to memory access commands.

For example, an F3851 PSU may have an address space of H'0000' through H'03FF'; an F3852 DMI may have an address space of H'0400' through H'07FF'. If a microcomputer system has these two memory devices and no others, then the F3851 PSU will respond to memory access commands when the PC0 or DC0 registers (whichever are identified as the address source) contain a value between H'0000' and H'03FF'; the F3852 DMI will respond to addresses in the range H'0400' through H'07FF'. No device will respond to addresses beyond H'07FF', even though such addresses may exist in PC0 and/or DC0.

Each device compares its address space with the contents of PC0 and DC0, whichever is identified as the address source, and only responds to a memory access command if the contents of PC0 or DC0 is within the device's address space.

If all memory address registers (PC0, PC1, DC0, and DC1) are to contain the same information, then ROMC states that require any of these registers' contents to be modified must be acted upon by all devices containing any of these four registers. If devices are not to compete when an ROMC state specifies that a memory access must be performed, then only a device whose address space includes the identified memory address must respond to the ROMC state.

As illustrated in *Figure 8,* the five ROMC signals that define the ROMC state are output early in the instruction cycle and are maintained stable for the duration of the instruction cycle; i.e., only one ROMC state can be specified per instruction cycle. Therefore, devices can only be called upon to perform one instruction execution related operation per one instruction cycle.

As referenced in the "ROMC Signal Functions" section, each ROMC state is identified by individual signal line states (1 for high, 0 for low), and by a two-digit hexadecimal code. The hexadecimal code is used to identify ROMC states throughout this data sheet. Also given in the "ROMC Signal Functions" section is the instruction cycle length (short or long) implied by each code, plus the way in which codes must be interpreted by the other F8 devices.

Instruction Execution Sequence

Every instruction execution sequence ends with an instruction code being fetched from memory to identify the next instruction cycle. The instruction code is loaded into the CPU instruction register, out of which it is decoded by the CPU control unit logic. An instruction fetch is executed during the last instruction cycle of the previous instruction, as illustrated in *Figure 9.*

There is a group of F8 instructions that cause operations to occur entirely within the F3850 CPU. These instructions do not use the data bus, therefore can execute in one cycle. Since one-cycle instructions do not use the data bus, no ROMC state needs to be generated for the one-cycle instruction being executed; therefore, as illustrated in *Figure 9,* ROMC state 0 is specified, causing the instruction fetch of the next instruction.

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

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

1. Operations may all be internal to the CPU. This will be the last or the only cycle for an instruction, and will specify ROMC state 0, as illustrated in *Figure 9.*

Figure 8 ROMC Timing Signals Output by F3850 CPU

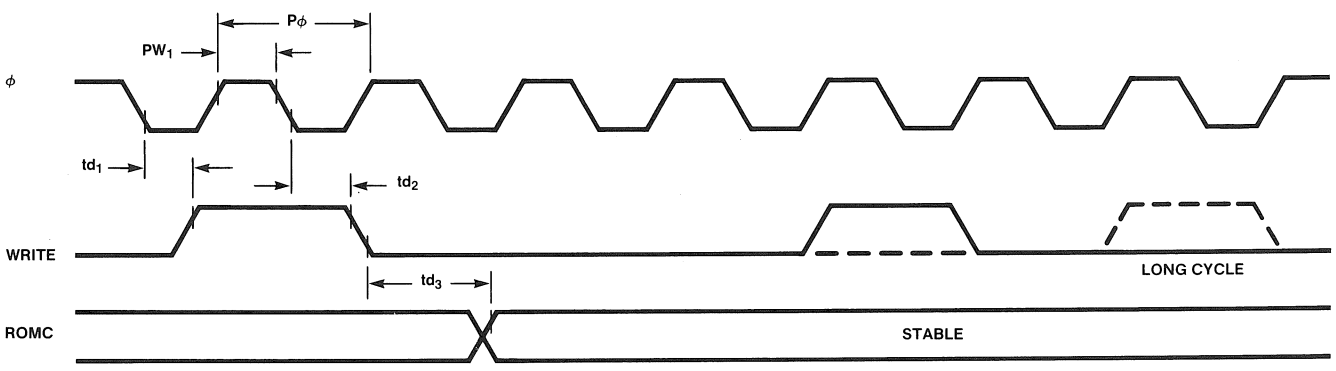


Figure 9A Short Cycle Instruction Fetch

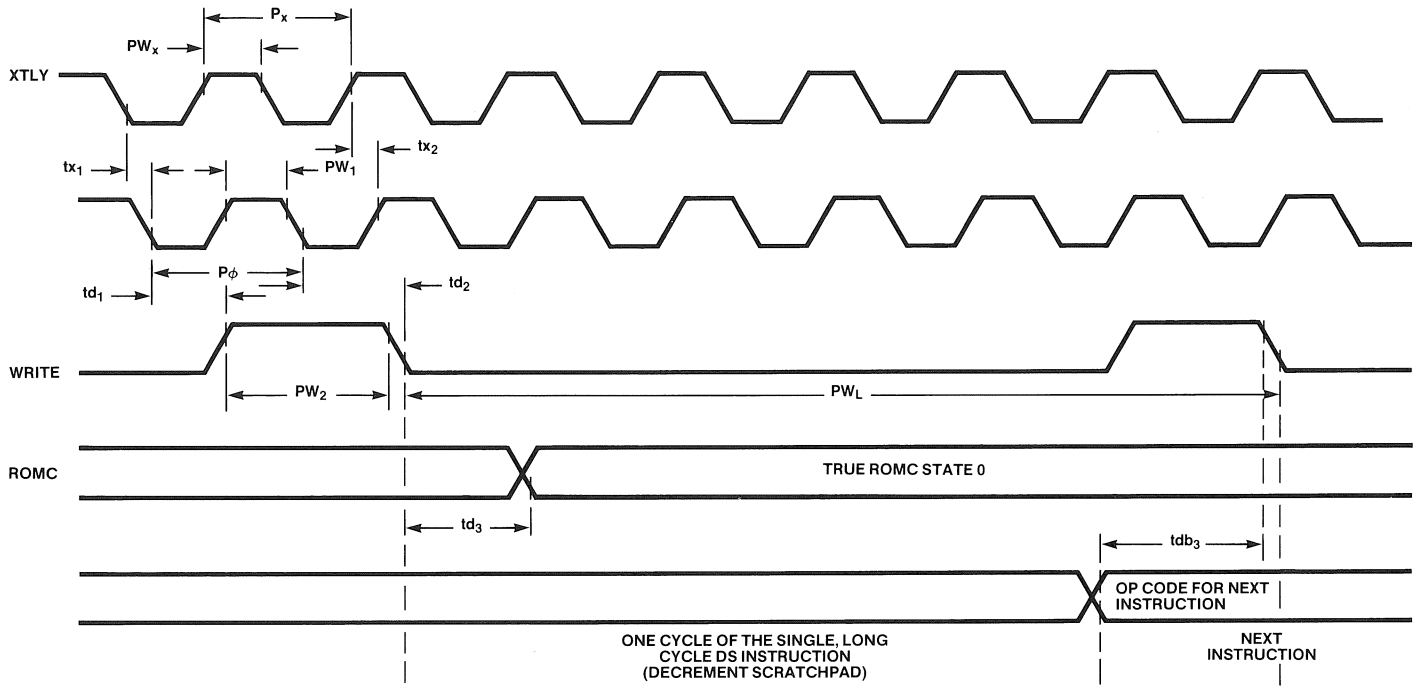
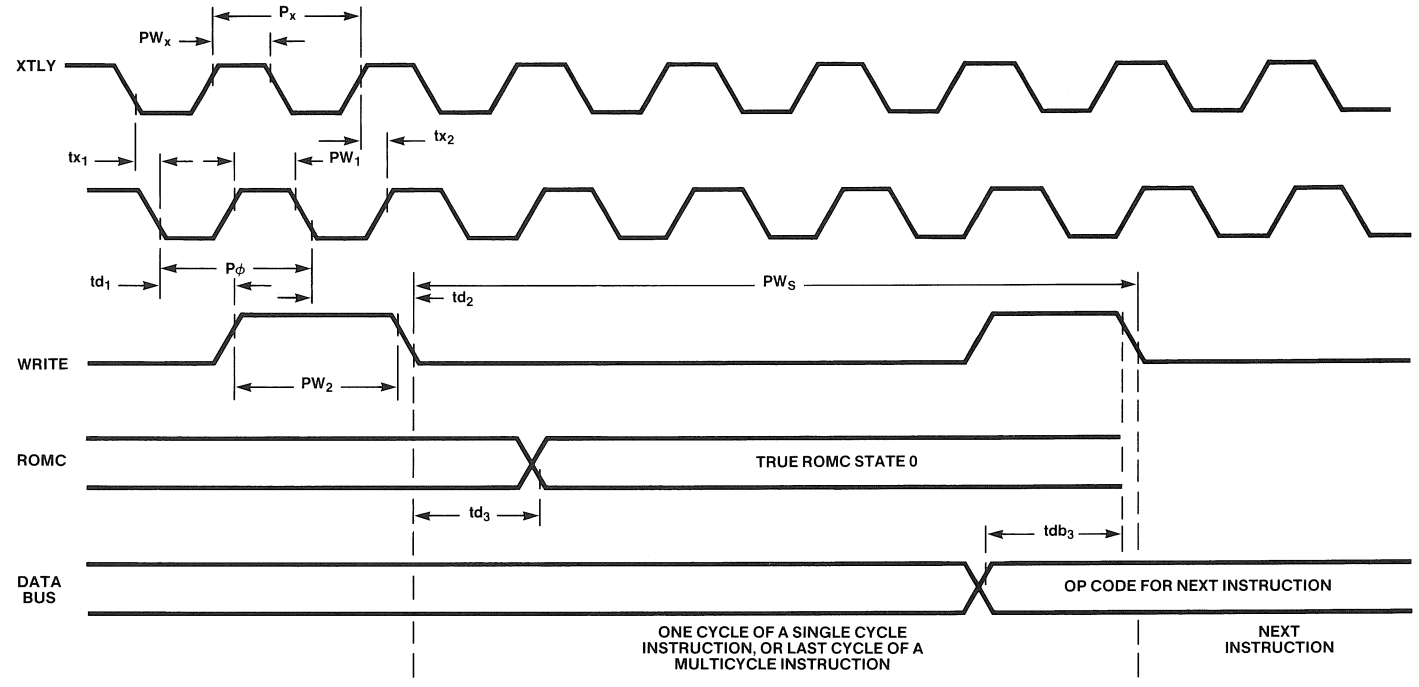


Figure 9B Long Cycle Instruction Fetch (During DS Only)



2. Data may be transferred between the F3850 CPU and memory devices. See the "Referencing Memory" section.
3. Data may be transferred from one memory device to all memory devices. The CPU is not the transmitter or the receiver of data in this transfer. See the "Memory-to-Memory Data Transfers" section.
4. Data may be transferred to or from an I/O port, as described in the "Input/Output Interfacing" section.
5. An interrupt may be acknowledged, as described in the "Interrupts" section.

Every F8 instruction is executed as one, or a sequence of, standard instruction cycles. Timing for the standard instruction cycles is illustrated in *Figures 9, 10, 11* and *12.*

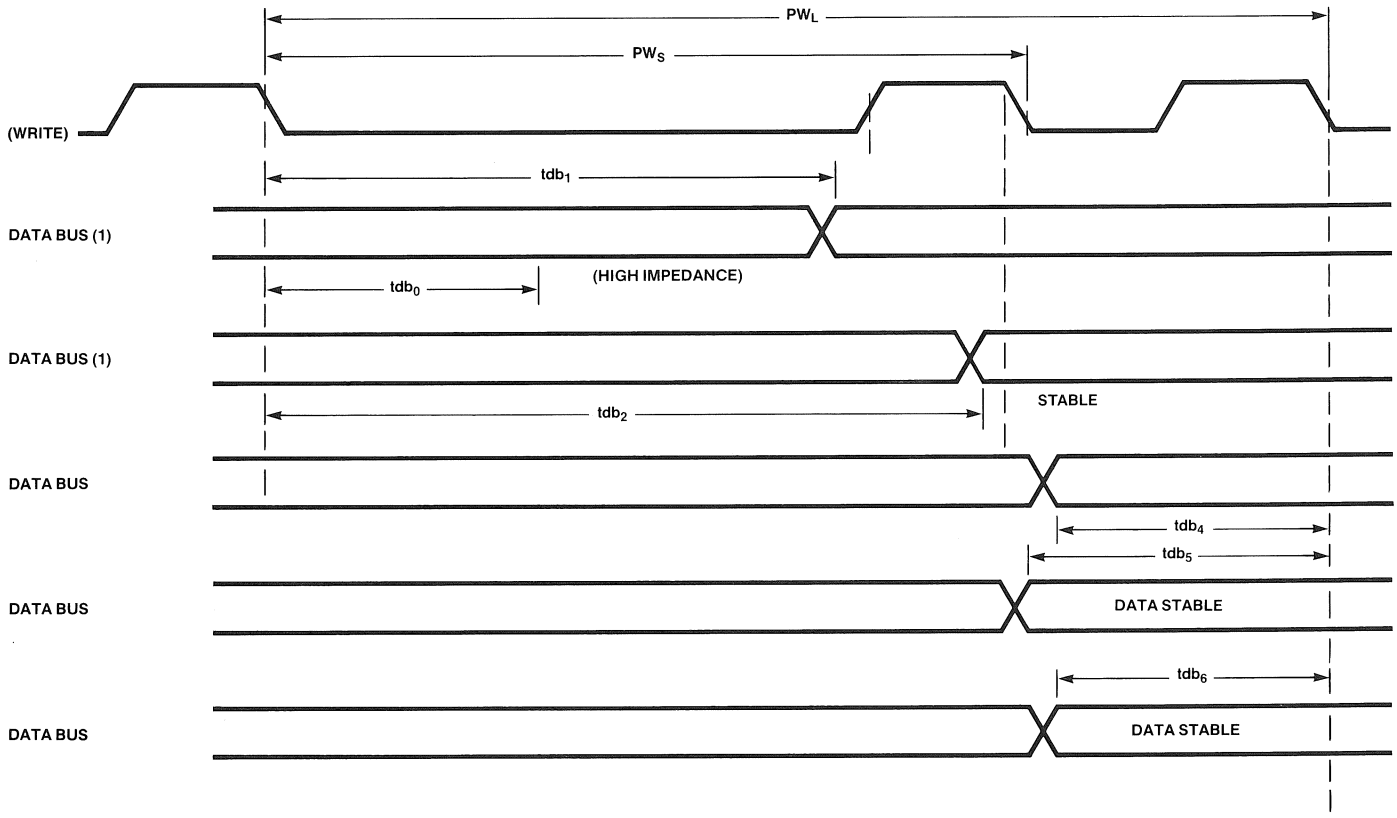
Refer to the "Instruction Cycle Execution and Timing" section for a list of the instruction cycles and their associated ROMC state.

Referencing Memory

Memory may be referenced during an instruction cycle either to transfer the data from the CPU to a memory word or to transfer data from a memory word to the CPU. A memory reference occurs as shown in *Figure 10.*

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

Figure 10 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).

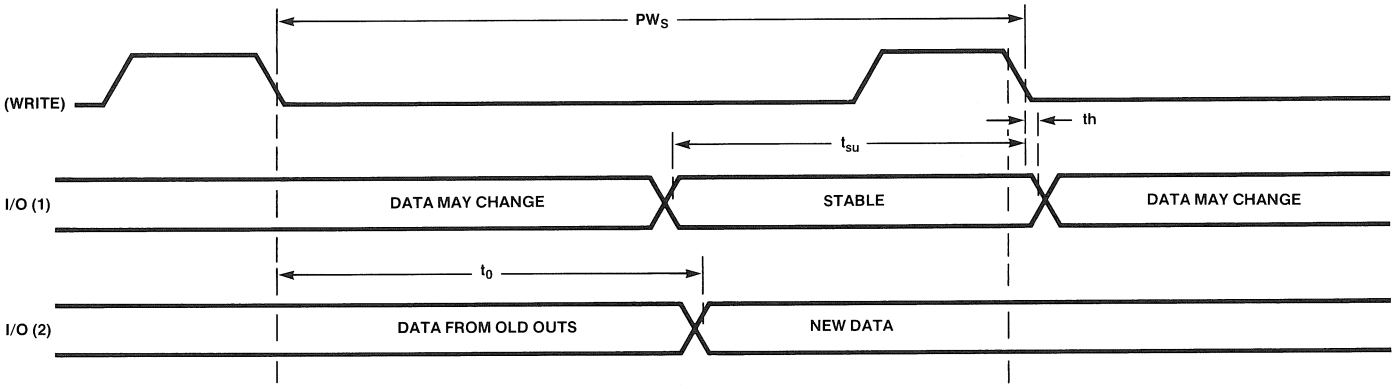
Delay tdb_0 is the delay for the CPU to stop driving the data bus.

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

tdb_3: Destination — IR (instruction Fetch)
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.

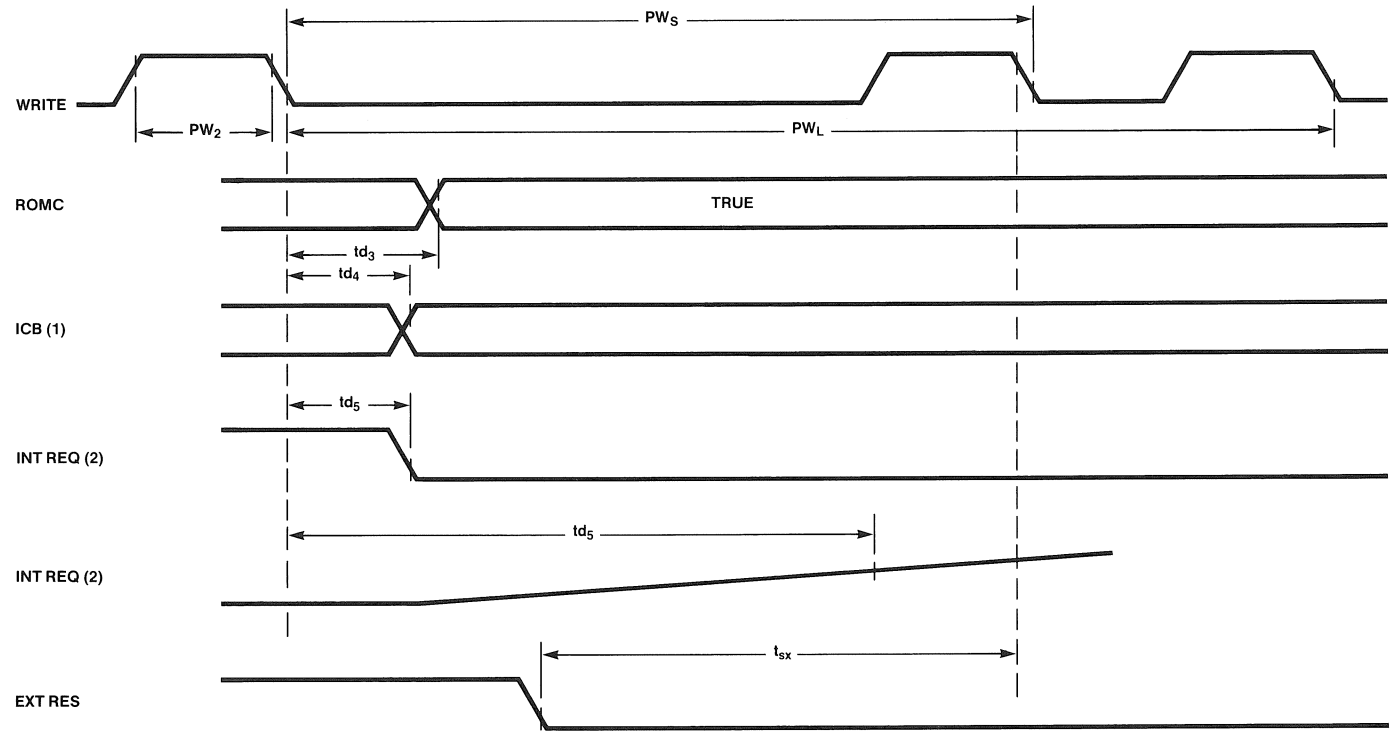
Figure 11 Timing for Data Input or Output at I/O Port Pins



(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.

Figure 12 Interrupt Signals Timing



(1) The $\overline{ICB}$ signal will go from a 1 to a 0 following the execution of the E1 instruction and will go from a 0 to a 1 following either the execution of the D1 instruction or the CPU's acknowledgement of an interrupt.

(2) This is an input to the CPU chip and is generated by a PSU or F3853 M1 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 2.0 V.

If data is being input to the CPU, then the delay before incoming data must be stable depends on the destination of the data, as illustrated in *Figure 10.*

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

The instruction fetch may also be viewed as a memory reference operation where the destination is the instruction register. Timing for this case is illustrated in *Figure 9.*

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

Three ROMC states (C, E, and 11) specify operations of this type, and *Figure 10* illustrates timing for the data transfer.
In *Figure 10,* tdb_2 is the delay until data from memory or a memory address register is stable on the data bus.

Input/Output Interfacing
Programmed I/O in the F8 microcomputer system is influenced by the design of the I/O port pins. As illustrated in *Figure 13*, each I/O port pin is a "wire-AND" structure between an internal latch and an external signal, if any. The latch is always loaded directly from the accumulator.

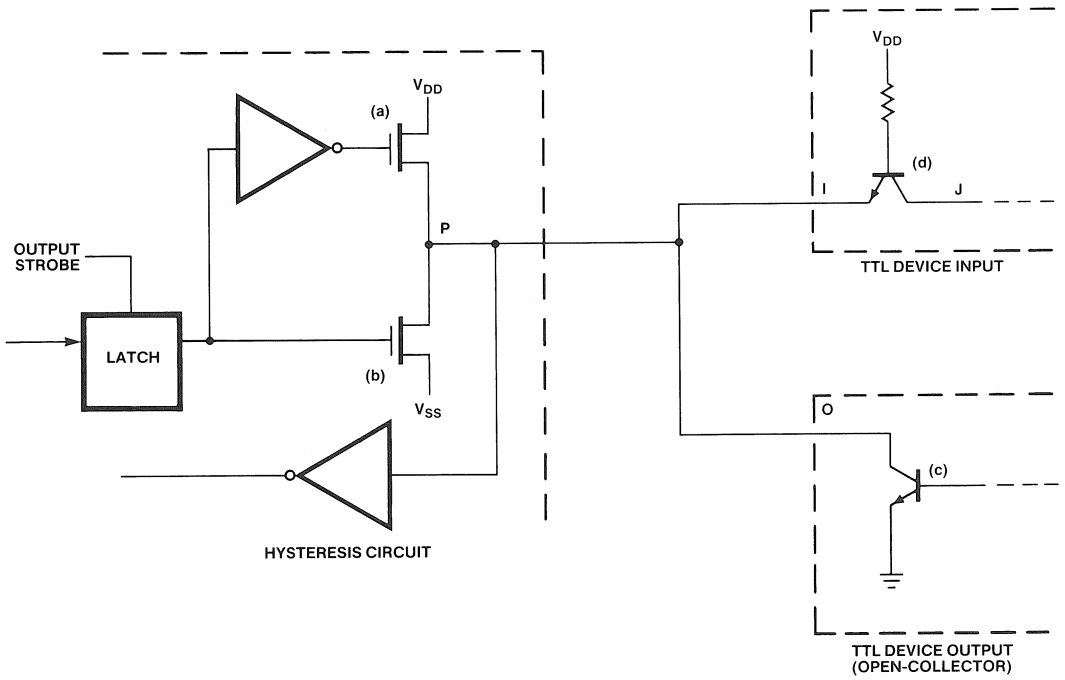
Each F8 I/O pin can be set high or low under program control. If a 1 (high) is presented at the latch, then gate (b) turns on and gate (a) turns off, so that P is at V_{SS} (low). If a 0 (low) is presented at the latch, then gate (a) turns on and gate (b) turns off, so that P is at V_{DD} (high).

When outputting data through an I/O port, the pin can be connected directly to a TTL gate input ("TTL Device Input" in *Figure 13*). Data is input to the pin from a "TTL Device Output" in *Figure 13.*

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, 1, P, and FET(b). This is transferred as a low level to the rest of the circuits in the TTL device and results in a high or low level at the output of the device, depending on its characteristics. If the level at P is set high, transistor (d) does not conduct current, and a high level is transferred by (d).

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

Figure 13 F8 I/O Port Bit



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

All I/O port bits should be set for a high level, before data input, to prevent incoming logic zeros from being "masked" by logic ones present at the port from previous outputs. In some instances, the ability to mask bits of a port to logic 1 is useful. (Note that logic 1 becomes a 0 V electrical level at the I/O pin; logic 0 corresponds to a high electrical level.)

The F8 CPU can execute two types of programmed I/O operation:

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

Input/Output operations that use the two CPU I/O ports execute in two instruction cycles. During the first cycle, the fetched instruction is decoded; the data bus is unused. In this cycle data is either sent from the accumulator to the I/O latch or enabled from the I/O pin to the accumulator, depending on whether the instruction is an output or an input. At the falling edge of the WRITE signal (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 11* illustrates I/O timing.

Note that for the data input (INS) the setup 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 the WRITE signal marking the beginning of the second cycle in the two-cycle instruction.

Input/Output instructions that address I/O ports with an I/O port address greater than 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 data sheet that describes the device containing the addressed I/O port.

Interrupts

There are three CPU signals with interrupt processing; timing for all signals is illustrated in *Figure 12*.

An interrupt sequence is initiated by pulling either the $\overline{\text{INT REQ}}$ signal or the $\overline{\text{EXT RES}}$ signal low. In the case of the $\overline{\text{INT REQ}}$ signal nothing happens unless the $\overline{\text{ICB}}$ signal is low. Also, nothing happens until the next interruptable instruction comes to the end of execution. In the case of the $\overline{\text{EXT RES}}$ signal, execution of the interrupt routine begins in the machine cycle immediately following that in which the signal goes low, provided that the setup time specified in *Figure 12* has been met. The $\overline{\text{EXT RES}}$ signal response logic ignores the $\overline{\text{ICB}}$ signal.

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

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

The $\overline{\text{ICB}}$ signal is pulled low by the E1 instruction and is returned high by the D1 instruction.

Instruction Set Summary

The F3850 CPU instruction set is summarized in *Table 3*. This section does not attempt to give complete directions for programming the F8 microcomputer system; it explains signals and timing associated with the execution of every instruction. Refer to *F8 and F3870 Guide to Programming* for programming details. The columns used in *Table 3* are described below.

Op Code—The Op Code is the instruction mnemonic that 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 that represents the operand field. Any portion of object code that does not represent the operand field must represent the mnemonic field. *Table 4* 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. Refer to *Table 4* for 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 (four clock period) cycle or

the letter L for a long (six clock period) cycle. Thus, the entry

S

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

S
L
S

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

ROMC State—This is the state, as identified in the "ROMC Signal Functions" section, that is output by the F3850 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 9* and *10.* For the exceptions, *Figure 11* is required.

The ROMC lines are always set after a delay of td_3, as shown in *Figure 9.* 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 the trailing edge of the WRITE signal high, for a long or short instruction cycle, as illustrated in *Figure 10.* There are two cases for tdb_3, however, as illustrated in *Figure 9.* These are identified in *Table 4* as 3S for *Figure 9A* and 3L for *Figure 9B;* tdb_1 through tdb_6 are otherwise identified by the numbers 1 through 6.

Cycles that do not use the data bus are identified by 0 in the timing column; *Figure 8* illustrates timing in this case.

Cycle	Represents
0	*Figure 8*
1	tdb_1 in *Figure 10*
2	tdb_2 in *Figure 10*
3S	tdb_3 in *Figure 9A*
3L	tdb_3 in *Figure 9B*
4	tdb_4 in *Figure 10*
5	tdb_5 in *Figure 10*
6	tdb_6 in *Figure 10*

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 affected
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 is reset to 0 (cleared).

Function—The effect of each instruction cycle is described in this column using symbology given in *Table 4.*

Instruction Cycle Execution and Timing

Table 3 lists the instruction cycles, plus the ROMC state associated with each cycle, for every F8 instruction. Note that instructions are described in the table by order of ascending instruction (first byte) object code. *Table 4* lists the symbology used in *Table 3.*

Table 3 Instruction Cycle 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)

Table 3 Instruction Cycle Execution and Timing (Continued)

Op Code	Operand(s)	Object Code	Cycle	ROMC State	Timing	Status Flags O	Z	C	S	Interrupt	Function
LR	K, P	08	L	7	5	—	—	—	—		r12 ← (PC1U)
			L	B	5	—	—	—	—		r13 ← (PC1L)
			S	0	3S	—	—	—	—		
LR	P, K	09	L	15	2	—	—	—	—		PC1U ← (r12)
			L	18	2	—	—	—	—		PC1L ← (r13)
			S	0	3S	—	—	—	—		
LR	A, IS	0A	S	0	3S	—	—	—	—		A ← (ISAR)
LR	IS, A	0B	S	0	3S	—	—	—	—		ISAR ← (A)
PK		0C	L	12	2	—	—	—	—		PC1 ← (PC0);
			L	14	2	—	—	—	—		PC0L ← (r13)
			S	0	3S	—	—	—	—		PC0U ← (r12)
LR	P0, Q	0D	L	17	2	—	—	—	—	x	
			L	14	2	—	—	—	—		PC0L ← (r15)
			S	0	3S	—	—	—	—		PC0U ← (r14)
LR	Q, DC	0E	L	6	3	—	—	—	—		r14 ← (DC0U)
			L	9	5	—	—	—	—		r15 ← (DC0L)
			S	0	3S	—	—	—	—		
LR	DC,Q	0F	L	16	2	—	—	—	—		DC0U ← (r14)
			L	19	2	—	—	—	—		DC0L ← (r15)
			S	0	3S	—	—	—	—		
LR	DC, H	10	L	16	2	—	—	—	—		DC0U ← (r10)
			L	19	2	—	—	—	—		DC0L ← (r11)
			S	0	3S	—	—	—	—		
LR	H, DC	11	L	6	5	—	—	—	—		r10 ← (DC0U)
			L	9	5	—	—	—	—		r11 ← (DC0L)
			S	0	3S	—	—	—	—		
SR	1	12	S	0	3S	0	1/0	0	1		Shift (A) right one bit position (zero fill)
SL	1	13	S	0	3S	0	1/0	0	1/0		Shift (A) left one bit position (zero fill)
SR	4	14	S	0	3S	0	1/0	0	1		Shift (A) right four bit positions (zero fill)
SL	4	15	S	0	3S	0	1/0	0	1/0		Shift (A) left four bit positions (zero fill)
LM		16	L	2	6	—	—	—	—		A ← ((DC0))
			S	0	3S	—	—	—	—		
ST		17	L	5	1	—	—	—	—		(DC) ← (A)
			S	0	3S	—	—	—	—		
COM		18	S	0	3S	0	1/0	0	1/0		A ← (A) ⊕ H'FF' Complement accumulator
LNK		19	S	0	3S	1/0	1/0	1/0	1/0		A ← (A) + (C)
DI		1A	S	1C	0	—	—	—	—	y	Clear ICB
			S	0	3S	—	—	—	—		
EI		1B	S	1C	0	—	—	—	—		Set ICB
			S	0	3S	—	—	—	—	x	
POP		1C	S	4	0	—	—	—	—		PC0 ← (PC1)
			S	0	3S	—	—	—	—	x	

Table 3 Instruction Cycle Execution and Timing (Continued)

Op Code	Operand(s)	Object Code	Cycle	ROMC State	Timing	Status Flags O	Z	C	S	Interrupt	Function
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) v 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	—	—	—	—		
CI	aa	25	L	3	4	—	—	—	—		Perform H'aa' + $\overline{(A)}$
	→	aa	S	0	3S	1/0	1/0	1/0	1/0		+ 1. Do not save result, but modify status flags to reflect result.
IN	PP	26	L	3	2	—	—	—	—		DB ← PP
		PP	L	1B	6	0	1/0	0	1/0		A ← (I/O Port PP)
			S	0	3S	—	—	—	—		
OUT	PP	27	L	3	2	—	—	—	—		DB ← PP
		PP	L	1A	1	—	—	—	—		I/O Port PP ← (A)
			S	0	3S	—	—	—	—	x	
PI	iijj	28	L	3	6	—	—	—	—		A ← H'ii'
	→	ii	S	D	0	—	—	—	—		PC1 ← (PC0) + 1
	→	jj	L	C	2	—	—	—	—		PC0L ← H'jj'
			L	14	1	—	—	—	—		PC0U ← (A)
			S	0	3S	—	—	—	—	x	
JMP	iijj	29	L	3	6	—	—	—	—		A ← H'ii'
	→	ii	L	C	2	—	—	—	—		PC0L ← H'jj'
	→	jj	L	14	1	—	—	—	—		PC0U ← (A)
			S	0	3S	—	—	—	—	x	
DCI	iijj	2A	L	11	2	—	—	—	—		DC0U ← ii
	→	ii	S	3	0						(increment PC0)
	→	jj	L	E	2	—	—	—	—		DC0L ← jj
			S	3	0						(increment PC0)
			S	0	3S	—	—	—	—		
NOP		2B	S	0	0	—	—	—	—		
XDC		2C	S	1D	0	—	—	—	—		DC0 ⇄ DC1
			S	0	0	—	—	—	—		
DS	r	3r	L	0	3L	1/0	1/0	1/0	1/0		r ← (r) + H'FF' Decrement scratchpad byte
LR	A, r	4r	S	0	3S	—	—	—	—		A ← (r)
LR	r, A	5r	S	0	3S	—	—	—	—		r ← (A)
LISU	e	6e	S	0	3S	—	—	—	—		ISARU ← 0'e'

Table 3 Instruction Cycle Execution and Timing (Continued)

Op Code	Operand(s)	Object Code	Cycle	ROMC State	Timing	Status Flags O	Z	C	S	Interrupt	Function
LISL	e	68 + e	S	0	3S	—	—	—	—		ISARL ← 0'e'
LIS	a	7a	S	0	3S	—	—	—	—		A ← H'0a'
BT	e, ii	8e	S	1C	0	—	—	—	—		Test e ∧ W register
		ii	S	3	0	—	—	—	—		Res = 0 so PC0 = (PC0) + 2
			S	0	3S						
			S	1C	0	—	—	—	—		Test e ∧ W register
			L	1	2	—	—	—	—		Res ≠ 0 so PC0 = (PC0)
						—	—	—	—		+ H'ii' + 1
			S	0	3S						
AM		88	L	2	4	—	—	—	—		
						1/0	1/0	1/0	1/0		A ← (A) + ((DC0)) Binary,
			S	0	3S						DC0 ← (DC) + 1
AMD		89	L	2	4	—	—	—	—		
						1/0	1/0	1/0	1/0		A ← (A) + ((DC0)) Decimal,
											DC0 ← (DC0) + 1
			S	0	3S	—	—	—	—		
NM		8A	L	2	4	0	1/0	0	1/0		A ← (A) ∧ ((DC0));
			S	0	3S	—	—	—	—		DC0 ← (DC0) + 1
OM		8B	L	2	4	0	1/0	0	1/0		A ← (A) ∧ ((DC0));
			S	0	3S	—	—	—	—		DC0 ← (DC0) + 1
XM		8C	L	2	4	0	1/0	0	1/0		A ← (A) ⊕ ((DC0));
			S	0	3S	—	—	—	—		DC0 ← (DC0) + 1
CM		8D	L	2	4	1/0	1/0	1/0	1/0		Set status flags on basis
			S	0	3S	—	—	—	—		of ((DC)) + (A) + 1;
						—	—	—	—		DC0 ← (DC0) + 1
ADC		8E	L	A	1	—	—	—	—		DC ← (DC) + (A)
			S	0	3S	—	—	—	—		
BR7	ii	8F	S	3	0	—	—	—	—		PC0 ← (PC0) + 2
		ii	S	0	3S	—	—	—	—		because (ISARL) = 7
			L	1	2	—	—	—	—		PC0 ← (PC0) + H'ii' + 1
			S	0	3S	—	—	—	—		because (ISARL) ≠ 7
BF	t, ii	9t	S	1C	0	—	—	—	—		Test t ∧ W. register
		ii	L	1	2	—	—	—	—		Res = 0 so PC0 = (PC0)
			S	0	3S	—	—	—	—		+ H'ii' + 1
			S	1C	0	—	—	—	—		Test t ∧ W. register
			S	3	0	—	—	—	—		Res ≠ 0 so PC0 = (PC0) + 2
			S	0	3S	—	—	—	—		
INS	0 or 1	A0, A1	S	1C	0	0	1/0	0	1/0		A ← (I/O Port 0 or 1)
			S	0	3S	—	—	—	—		
INS	4	A4	L	1C	0	0	1/0	0	1/0		DB ← Port address (4 thru 15)
	thru	thru	L	1B	6	—	—	—	—		
	15	AF	S	0	3S	—	—	—	—		A ← (Port 4 thru 15)
OUTS	0 or 1	B0, B1	S	1C	0	—	—	—	—		I/O Port 0 or 1 ← (A)
			S	0	3S	—	—	—	—		
OUTS	4	B4	L	1C	0	—	—	—	—		DB ← Port address (4 thru 15)
	thru	thru	L	1A	1	—	—	—	—		
	15	BF	S	0	3S	—	—	—	—	x	Port (4 thru 15) (A)

Table 3 Instruction Cycle Execution and Timing (Continued)

Op Code	Operand(s)	Object Code	Cycle	ROMC State	Timing	Status Flags O	Z	C	S	Interrupt	Function
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) v (r)
INTRPT		xx	L	1C	0	—	—	—	—		IDLE
			L	0F	2	—	—	—	—		PC0L ← Int. address (lower byte); PC1 ← PC0
			L	13	2	—	—	—	—	y	PC0U ← Int. address (upper byte)
			S	0	3S	—	—	—	—	x	
RESET		xx	S	1C	0	—	—	—	—		IDLE
			L	8	1	—	—	—	—	y	PC0 ← 0, PC1 ← PC0
			S	0	3S	—	—	—	—	x	

Table 4 Instruction Execution and Timing Symbology

Symbol	Interpretation
A	The accumulator
$\overline{(A)}$	The complement of accumulator contents
a	A single hexadecimal digit being interpreted as data
aa	Two hexadecimal digits being interpreted as a single byte of data or as the high order byte of 16 bits of data
bb	Two hexadecimal digits being interpreted as the low order byte of 16 bits of data
Binary	Binary arithmetic specified
C	The carry status flag
DB	F8 system data bus
DC0	The primary data counter register
DC0L	The low order byte of the primary data counter register
DC0U	The high order byte of the primary data counter register
DC1	The secondary data counter register
Decimal	Decimal arithmetic specified
e	A single octal digit being interpreted as data
H	Scratchpad bytes 10 and 11
ii	Two hexadecimal digits being interpreted as the high order byte of a 16-bit address or as a simple byte address displacement
ISAR	The 6-bit scratchpad address register
ISARL	The low order three bits of ISAR
ISARU	The high order three bits of ISAR
J	Scratchpad byte 9
jj	Two hexadecimal digits being interpreted as the low order byte of a 16-bit address
K	Scratchpad bytes 12 and 13
KL	Scratchpad byte 13
KU	Scratchpad byte 12
O	The overflow status flag
P	A single hexadecimal digit being interpreted as an I/O port address (0–15)
PP	Two hexadecimal digits being interpreted as an I/O port address (0–255)
PC0	The program counter register
PC0L	The low order byte of the program counter register
PC0U	The high order byte of the program counter register
PC1	The stack register
PC1L	The low order byte of the stack register
PC1U	The high order byte of the stack register
Q	Scratchpad bytes 14 and 15
QL	Scratchpad byte 15
QU	Scratchpad byte 14

Table 4 Instruction Execution and Timing Symbology (Continued)

Symbol	Interpretation
r	Single hexadecimal digit interpreted as scratchpad address: 4 = 0 through B for locations 0 through B in scratchpad r = C for ISAR as address source with no change after access r = D for ISAR as address source with ISARL = ISARL + 1 after access r = E for ISAR as address source with ISARL = ISARL−1 after access r = F is not allowed
S	The sign status flag
t	A single hexadecimal digit identifying a status condition that is tested by a Branch on Condition instruction
W	The status register

Symbol	Interpretation
Z	The zero status flag
∧	The logical OR of 8-bit quantities on each side of this symbol is specified
v	The logical AND 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

ROMC Signal Functions

Table 5 describes the ROMC signals and their functions.

Table 5 ROMC Signal Functions

ROMC 4 3 2 1 0	HEX	Cycle Length	Function
0 0 0 0 0	00	S, L	Instruction Fetch. The device whose address space includes the contents of the PC0 register must place on the data bus the op code addressed by PC0; then all devices increment the contents of PC0.
0 0 0 0 1	01	L	The device whose address space includes the contents of the PC0 register must place on the data bus the contents of the memory location addressed by PC0; then all devices add the 8-bit value on the data bus, as a signed binary number, to PC0.
0 0 0 1 0	02	L	The device whose DC0 addresses a memory word within the address space of that device must place on the data bus the contents of the memory location addressed by DC0; then all devices increment DC0.
0 0 0 1 1	03	L, S	Similar to 00, except that it is used for Immediate Operand fetches (using PC0) instead of instruction fetches.
0 0 1 0 0	04	S	Copy the contents of PC1 into PC0.
0 0 1 0 1	05	L	Store the data bus contents into the memory location pointed to by DC0; increment DC0.
0 0 1 1 0	06	L	Place the high order byte of DC0 on the data bus.
0 0 1 1 1	07	L	Place the high order byte of PC1 on the data bus.
0 1 0 0 0	08	L	All devices copy the contents of PC0 into PC1. The CPU outputs zero on the data bus in this ROMC state. Load the data bus into both halves of PC0, thus clearing the register.
0 1 0 0 1	09	L	The device whose address space includes the contents of the DC0 register must place the low order byte of DC0 onto the data bus.
0 1 0 1 0	0A	L	All devices add the 8-bit value on the data bus, treated as a signed binary number, to the data counter.
0 1 0 1 1	0B	L	The device whose address space includes the value in PC1 must place the low order byte of PC1 on the data bus.

Table 5 ROMC Signal Functions (Continued)

ROMC 4 3 2 1 0	HEX	Cycle Length	Function
0 1 1 0 0	0C	L	The device whose address space includes the contents of the PC0 register must place the contents of the memory word addressed by PC0 onto the data bus; then all devices move the value that has just been placed on the data bus into the low order byte of PC0.
0 1 1 0 1	0D	S	All devices store in PC1 the current contents of PC0, incremented by 1; PC0 is unaltered.
0 1 1 1 0	0E	L	The device whose address space includes the contents of PC0 must place the contents of the word addressed by PC0 onto the data bus. The value on the data bus is then moved to the low order byte of DC0 by all devices.
0 1 1 1 1	0F	L	The interrupting device with highest priority must place the low order byte of the interrupt vector on the data bus. All devices must copy the contents of PC0 into PC1. All devices must move the contents of the data bus into the low order byte of PC0.
1 0 0 0 0	10	L	Inhibit any modification to the interrupt priority logic.
1 0 0 0 1	11	L	The device whose memory space includes the contents of PC0 must place the contents of the addressed memory word on the data bus. All devices must then move the contents of the data bus to the upper byte of DC0.
1 0 0 1 0	12	L	All devices copy the contents of PC0 into PC1. All devices then move the contents of the data bus into the low order byte of PC0.
1 0 0 1 1	13	L	The interrupting device with highest priority must move the high order half of the interrupt vector onto the data bus. All devices must move the contents of the data bus into the high order byte of PC0. The interrupting device resets its interrupt circuitry (so that it is no longer requesting CPU servicing and can respond to another interrupt).
1 0 1 0 0	14	L	All devices move the contents of the data bus into the high order byte of PC0.
1 0 1 0 1	15	L	All devices move the contents of the data bus into the high order byte of PC1.
1 0 1 1 0	16	L	All devices move the contents of the data bus into the high order byte of DC0.
1 0 1 1 1	17	L	All devices move the contents of the data bus into the low order byte of PC0.
1 1 0 0 0	18	L	All devices move the contents of the data bus into the low order byte of PC1.
1 1 0 0 1	19	L	All devices move the contents of the data bus into the low order byte of DC0.
1 1 0 1 0	1A	L	During the prior cycle, an I/O port timer or interrupt control register was addressed; the device containing the addressed port must move the current contents of the data bus into the addressed port.
1 1 0 1 1	1B	L	During the prior cycle, the data bus specified the address of an I/O port. The device containing the addressed I/O port must place the contents of the I/O port on the data bus. (Note that the contents of timer and interrupt control registers cannot be read back onto the data bus.)
1 1 1 0 0	1C	L or S	None.
1 1 1 0 1	1D	S	Devices with DC0 and DC1 registers must switch registers. Devices without a DC1 register perform no operation.
1 1 1 1 0	1E	L	The device whose address space includes the contents of PC0 must place the low order byte of PC0 onto the data bus.
1 1 1 1 1	1F	L	The device whose address space includes the contents of PC0 must place the high order byte of PC0 onto the data bus.

Timing Characteristics

The timing characteristics of the F3850 are described in *Table 6.* $V_{DD} = +5\ V \pm 5\%$, $V_{GG} = +12\ V \pm 5\%$, $V_{SS} = 0\ V$, $T_A = O°C$ to $+70°C$

Table 6 F3850 CPU Signal Timing Characteristics

Symbol	Characteristic	Min	Typ	Max	Units	Test Conditions
P_x*	External Input Period	0.5		1.0	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 Extended Temp. Range			250 500	ns ns	$C_L = 100$ pF
tx_2	Ext. to ϕ + to + Delay Extended Temp. Range			250 500	ns ns	$C_L = 100$ pF
$P\phi$	ϕ Period	0.5		1.0	μs	
PW_1	ϕ Pulse Width	180		$P\phi$ - 180	ns	t_r, $t_f = 50$ ns; $C_L = 100$ pF
td_1	ϕ to WRITE + Delay Extended Temp. Range		150	250 400	ns ns	$C_L = 100$ pF
td_2	ϕ to WRITE - Delay Extended Temp. Range		150	250 400	ns ns	$C_L = 100$ pF
PW_2	WRITE Pulse Width	$P\phi$ - 100		$P\phi$	ns	t_r, t_f 50 ns typ; $C_L = 100$ pF
PW_S	WRITE Period; Short		$4P\phi$			
PW_L	WRITE Period; Long		$6P\phi$			
td_3	WRITE to ROMC Delay	80	300	550	ns	$C_L = 100$ pF
td_4*	WRITE to $\overline{ICB}$ Delay			350	ns	$C_L = 50$ pF
td_5	WRITE to $\overline{INT\ \ REQ}$ Delay			430	ns	$C_L = 100$ pF
t_{sx}*	$\overline{EXT\ \ RES}$ Setup Time	1.0			μs	$C_L = 20$ pF
t_{su}*	I/O Setup Time	300			ns	
t_h*	I/O Hold Time	50			ns	
t_o*	I/O Output Delay			2.5	μs	$C_L = 50$ pF
tdb_1*	WRITE to Data Bus Stable		0.6	1.3	μs	$C_L = 100$ pF
tdb_2	WRITE to Data Bus Stable	$2P\phi$		$2P\phi + 1.0$	μs	$C_L = 100$ pF
tdb_3*	Data Bus Setup	200			ns	
tdb_4*	Data Bus Setup	500			ns	
tdb_5	Data Bus Setup	500			ns	
tdb_6*	Data Bus Setup	500			ns	

1. Symbols marked with an asterisk (*) refer to parameters that are most frequently of importance when interfacing to an F8 system. They encompass I/O timing, external timing generation, and possible external RAM timing. The remaining parameters are typically those that are only relevant between F8 devices, and not normally of concern to the user.
2. Input and output capacitance is 3 to 5 pF typical on all pins except V_{DD}, V_{GG}, and V_{SS}.
3. If $\overline{INT\ \ REQ}$ is being supplied asynchronously, it can be pulled down at any time except during a fetch cycle that has been preceded by a non-privileged instruction. In that case $\overline{INT\ \ REQ}$ must go down according to the requirements of td_5.

DC Characteristics

The DC characteristics of the F3850 are provided in *Table 7*. $V_{DD} = +5\ V \pm 5\%$, $V_{GG} = +12\ V \pm 5\%$, $V_{SS} = 0V$, $T_A = 0°C$ to $+70°C$

Table 7 F3850 CPU Signal DC Characteristics

Signal	Symbol	Characteristic	Min	Max	Unit	Test Conditions
ϕ, WRITE	V_{OH}	Output High Voltage	4.4	V_{DD}	V	$I_{OH} = -50\ \mu A$
	V_{OL}	Output Low Voltage	V_{SS}	0.4	V	$I_{OL} = 1.6$ mA
	V_{OH}	Output High Voltage	2.9		V	$I_{OH} = -100\ \mu A$
XTLY	V_{IH}	Input High Voltage	4.5	V_{GG}	V	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	V	
	I_{IH}	Input High Current	5	50	μA	$V_{IN} = V_{DD}$
	I_{IL}	Input Low Current	−10	−120	μA	$V_{IN} = V_{SS}$
$ROMC_{0-4}$	V_{OH}	Output High Voltage	3.9	V_{DD}	V	$I_{OH} = -100\ \mu A$
	V_{OL}	Output Low Voltage	V_{SS}	0.4	V	$I_{OL} = 1.6$ mA
DB_{0-7}	V_{IH}	Input High Voltage	2.9	V_{DD}	V	
	V_{IL}	Input Low Voltage	V_{SS}	0.8	V	
	V_{OH}	Output High Voltage	3.9	V_{DD}	V	$I_{OH} = -100\ \mu A$
	V_{OL}	Output Low Voltage	V_{SS}	0.4	V	$I_{OL} = 1.6$ mA
	I_{IH}	Input High Current		3	μA	$V_{IN} = 7$ V 3-State mode
	I_{IL}	Input Low Current		−3	μA	$V_{IN} = V_{SS}$ 3-State mode
I/O_{0-17}	V_{OH}	Output High Voltage	3.9	V_{DD}	V	$I_{OH} = -30\ \mu A$
	V_{OH}	Output High Voltage	2.9	V_{DD}	V	$I_{OH} = -150\ \mu A$
	V_{OL}	Output Low Voltage	V_{SS}	0.4	V	$I_{OL} = 1.6$ mA
	V_{IH}	Input High Voltage(1)	2.9	V_{DD}	V	Internal pull-up to V_{DD}
	V_{IL}	Input Low Voltage	V_{SS}	0.8	V	
	I_{IL}	Input Low Current		−1.6(4)	mA	$V_{IN} = 0.4$ V(2)
$\overline{EXT\ RES}$	V_{IH}	Input High Voltage	3.5	V_{DD}	V	Internal pull-up to V_{DD}
	V_{IL}	Input Low Voltage	V_{SS}	0.8	V	
	I_{IL}	Input Low Current	−0.1	−1.0	mA	$V_{IN} = V_{SS}$
$\overline{INT\ REQ}$	V_{IH}	Input High Voltage	3.5	V_{DD}	V	Internal pull-up to V_{DD}
	V_{IL}	Input Low Voltage	V_{SS}	0.8	V	
	I_{IL}	Input Low Current	−0.1	−1.0	mA	$V_{IN} = V_{SS}$
$\overline{ICB}$	V_{OH}	Output High Voltage	3.9	V_{DD}	V	$I_{OH} = -10\ \mu A$
	V_{OH}	Output High Voltage	2.9	V_{DD}	V	$I_{OH} = -100\ \mu A$
	V_{OL}	Output Low Voltage	V_{SS}	0.4	V	$I_{OL} = 100\ \mu A$

1. Hysteresis input circuit provides additional 0.3 V noise immunity while internal pull-up provides TTL compatibility.
2. Measured while F8 port is outputting a high level.
3. Guaranteed but not tested.
4. −1.8 V max. for extended temperature range.
5. Positive current is defined as conventional current flowing into the pin referenced.

Supply Currents

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
I_{DD}	V_{DD} Current		45	75	mA	f = 2 MHz, Outputs Unloaded
I_{GG}	V_{GG} Current		12	30	mA	f = 2 MHz, Outputs Unloaded

Recommended Operating Ranges

The recommended operating ranges of the F3850 are shown below.

Absolute Maximum Ratings

V_{GG}	−0.3 V, +15 V
V_{DD}	−0.3 V, +7 V
XTLX, XTLY, and XTLZ	−0.3 V, +15 V
All other inputs	−0.3 V, +7 V
Storage temperature	−55° C, +150° C
Operating temperature	0° C, +70° C

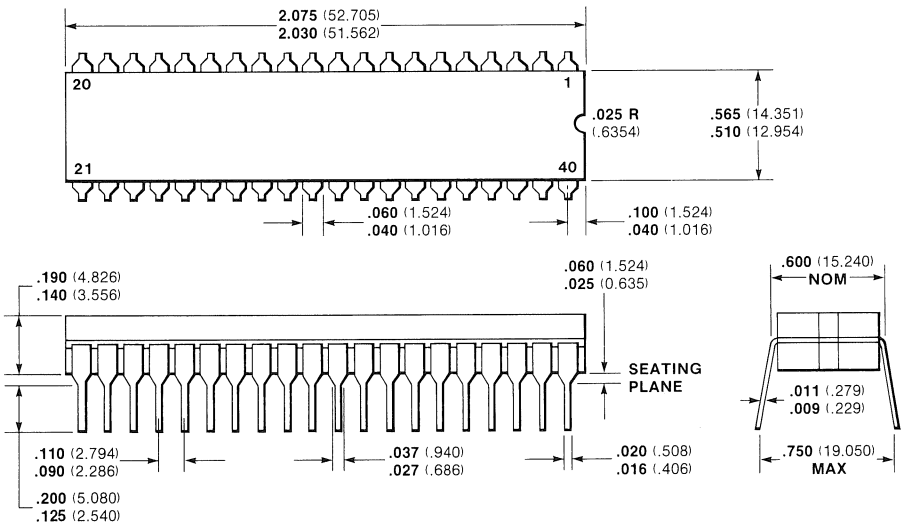
These are stress ratings only, and functional operation at these ratings, or under any conditions above those indicated in this data sheet, is not implied. Exposure to the absolute maximum rating conditions for extended periods of time may affect device reliability, and exposure to stresses greater than those listed may cause permanent damage to the device.

	Supply Voltage (V_{DD})			Supply Voltage (V_{GG})			
Part Number	Min	Typ	Max	Min	Typ	Max	V_{SS}
F3850	+4.75 V	+5 V	+5.25 V	+11.4 V	+12 V	+12.6 V	0 V

Ordering Information

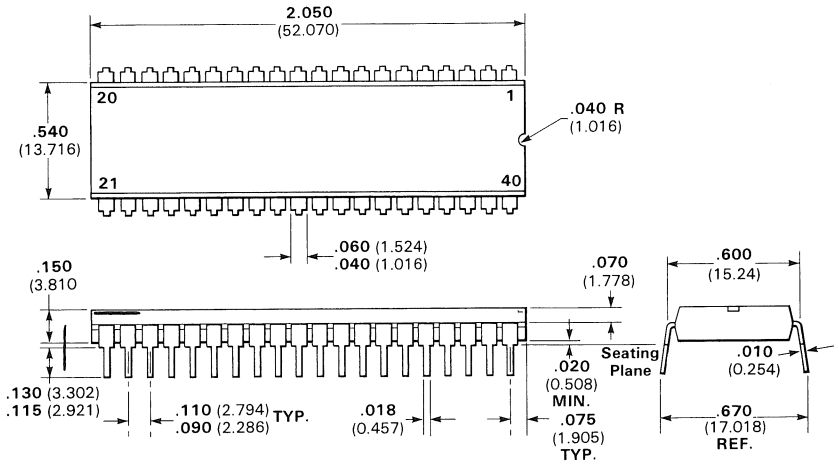
Order Code	Package	Temperature Range
F3850DC	Ceramic	0° C to +70° C
F3850DL	Ceramic	−40° C to +85° C
F3850DM	Ceramic	−55° C to +125° C
F3850PC	Plastic	0° C to +70° C
F3850PL	Plastic	−40° C to +85° C
F3850PM	Plastic	−55° C to +125° C

Package Information
40-Pin Ceramic Dual-In-Line



Notes
All dimensions are in inches **bold** and millimeters (parentheses).
Pin material is nickel gold-plated kovar.
Cap is kovar.
Base is ceramic.
Package weight is 6.5 grams.

40-Pin Plastic Dual-In-Line



Notes
All dimensions are in inches **bold** and millimeters (parentheses).
Pins are tin-plated kovar.
Package material is plastic.