Signetics

8X310
Interrupt Control Coprocessor

Product Specification

Military
Customer Specific Products

FEATURES

- **Three prioritized interrupts**
- **Subroutine handling capabilities**
- **4-level LIFO stack for return address storage**
- **Interrupt masking by software and hardware**
- **Stack full flag**
- **Directly compatible with 8X305 Microcontroller**
- **Bipolar ISL (Integrated Schottky Logic) and low-power Schottky technology**
- **Single +5V power supply**
- **0.6", 40-pin DIP**

PRODUCT DESCRIPTION

The Signetics 8X310 Interrupt Control Coprocessor (ICC) supports the 8X305 Microcontroller in systems that are interrupt driven and those that require subroutine handling capabilities.

As shown in Figure 1, the ICC provides three prioritized interrupt request lines, INT 0 (highest priority), INT 1 and INT 2. A Low-to-High transition applied to any of these input lines latches in an interrupt request which may be serviced when sampled by the ICC once each instruction cycle of the Microcontroller. When an interrupt request is serviced, the ICC forces the Microcontroller to jump to one of three fixed locations in program memory; instruction addresses 4, 5, and 6 correspond to INT 0, INT 1, and INT 2. At each of these addresses, the user programs a JMP instruction to another address where the user's interrupt service routine begins.

During interrupt servicing, the ICC also stores the proper return address into a four deep, Last-In-First-Out (LIFO) stack. At the conclusion of the interrupt service routine, the user program instructs the ICC to return to the main program at the location previously stored in the stack. The return operation is implemented by coding a special RETURN instruction which is decoded directly off the instruction bus by the ICC. There are five such special instructions relating to interrupt and subroutine handling functions performed by the ICC. These instruction codes are all treated as non-operational instructions (NOPs) by the Microcontroller.

An internal one-bit mask is used to inhibit interrupt servicing. Whenever the mask is set, the ICC does not respond to any pending interrupt requests; however, any requests remain latched for future servicing. The mask can be set and cleared either by the user program or automatically during certain ICC functions. The special instructions SET MASK and CLEAR MASK are provided for user control. The Interrupt Disable input also inhibits interrupt request servicing.

ORDERING INFORMATION

DESCRIPTION	ORDER CODE
40–Pin DIP	8X310/BQA

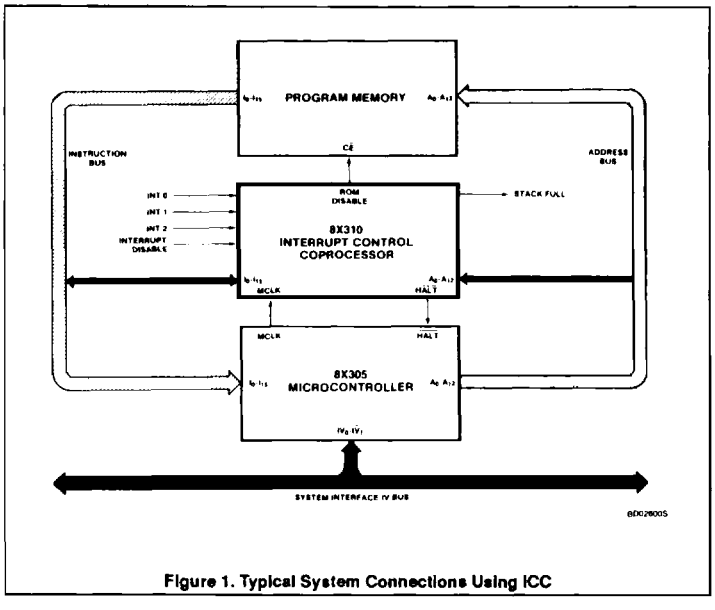
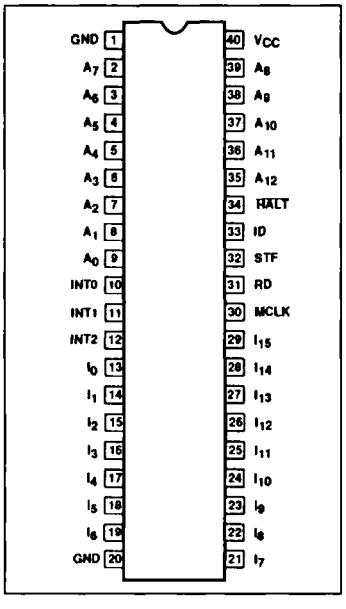


Figure 1. Typical System Connections Using ICC

PIN CONFIGURATION



PIN DESCRIPTION

PIN NO.	IDENTIFIER	FUNCTION
1, 20	GND	Ground. (Note: The printed circuit board should not use the ICC as a bridge for external ground.)
2 – 9, 35 – 39	A_7, A_0, A_{12} – A_8	Program address input lines from Microcontroller. Active High. A_0 is MSB.
10 – 12	INT 0 – INT 2	Interrupt request input pins. INT 0 has the highest priority and INT 2 the lowest — Edge-triggered on a Low-to-High transition.
13 – 19	I_0 – I_6,	Bidirectional instruction bus; I_0 is MSB. When acting as an input, the ICC decodes the instruction flow (binary pattern on I_0 – I_{15}) between program storage and the Microcontroller. During an interrupt or return cycle, the ICC outputs a JMP instruction to the Microcontroller via these lines — refer to FUNCTIONAL OPERATION of ICC.
30	MCLK	MasterCLocK — Active-High input from 8X305 Microcontroller used for a timing reference and system synchronization.
31	RD	ROM (or PROM) Disable — Active-High output used to disable normal program storage so that the ICC can force an instruction to the Microcontroller.
32	STF	STack Full — Active-High output. When the LIFO stack if full, STF goes High and remains High until at least one register in the 4-level stack is empty.
33	ID	Interrupt Disable — Active-High. When this input pin is driven High, servicing of all interrupt requests is suspended.
34	$\overline{HALT}$	Active-Low output. Suspends all processing operations of the Microcontroller during period when the source of instruction data is changing between the ICC and program storage.
40	V_{CC}	+5V power supply.

The ICC provides a facility for implementing subroutines in the user program. A special PUSH instruction directs the ICC to store the return address into the stack in a manner similar to interrupt servicing. The jump to the subroutine, however, is performed by the user program. Subroutines may be nested (called from within other subroutines) depending on remaining vacancies in the four deep stack.

In general, the ICC adds some useful and very flexible facilities to the 8X305-based system. It offers both hardware and software capabilities that can improve efficiency and decrease program size. These features, from both a chip and system aspect, are described in subsequent paragraphs.

FUNCTIONAL OPERATION

Basic Functions

The ICC performs the three general functions indicated below.

Function 1: Provides a means for the 8X305 Microcontroller to respond to interrupt requests by diverting the program flow of the 8X305 Microcontroller to the proper interrupt service routine or, in the case of a subroutine, the IC stores the return address in the 4-level LIFO stack (Figure 2.)

Function 2: Returns the user to the proper point in the man program for both interrupt and subroutine activities.

Function 3: Provides both automatic and programmed masking capabilities.

Interrupt Requests and Priority Considerations

An interrupt is requested when any one of the ICC input pins INT 0, INT 1, or INT 2 undergoes a Low-to-High transition; this request is temporarily stored in an internal edge-triggered latch that corresponds to the affected interrupt input. The interrupt request latches are part of the Priority and Mask Logic shown in Figure 2. Unless masked or otherwise disabled, the ICC samples these latches once each instruction cycle. Any or all of the latches may be set when sampled by the ICC; however, only the interrupt of highest priority will be serviced — the remaining interrupts will be held in queue. Thus, if INT 0, INT 1 and INT 2 simultaneously compete for service, INT 0 is the first to be serviced followed, in order, by INT 1 and INT 2; likewise, if INT 1 and INT 2 compete for service, INT 1, being of higher priority will be serviced first. The CLEAR INTERRUPT instruction resets all interrupt request latches without affecting an interrupt service routine that is already in progress.

The highest priority interrupt request will be serviced when sampled by the ICC provided interrupts in general are not inhibited and a previous interrupt of equal or higher priority is not currently being serviced. The general masking of interrupts is discussed later. To determine priorities, the ICC keeps track of any interrupt that is serviced until the corresponding service routine returns. A subsequent interrupt request may interrupt a service routine inrprogress only if it is of a higher priority than that of the current interrupt being serviced. If, for example, INT 1 is requested and serviced, then before its service routine finishes, a request on INT 0 can be serviced as a second level interruption. However, a request on INT 2 or a second request on INT 1 must wait until the original INT 1 service routine returns. The interrupt service routine that was interrupted will resume execution at the point of interruption when the higher priority service routine returns (i.e., in the same manner as when returning to the main program).

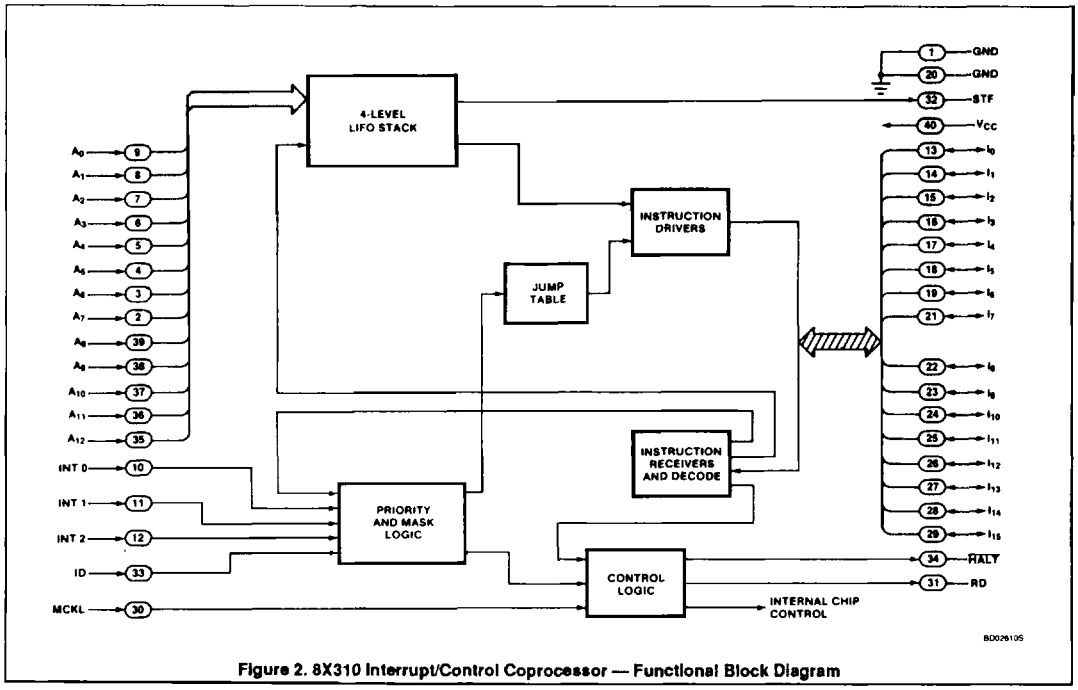


Figure 2. 8X310 Interrupt/Control Coprocessor — Functional Block Diagram

Interrupt Servicing

Interrupts are sampled only at the conclusion of an instruction cycle while the next instruction is being fetched from Program Memory.

When an interrupt request is serviced, the following general steps are performed:

- Address of the instruction that would normally be executed next is pushed into the 4-level LIFO Stack (Figure 2) for subsequent return to the main program.
- The ICC disables program storage and forces a JMP instruction onto the Instruction bus of the 8X305 Microcontroller. (Note: because of timing considerations, the HALT signal is driven low to suspend operation of the Microcontroller for one instruction cycle; this permits the source of instruction data to change from program storage to the ICC without conflict.) The JMP instruction from the ICC transfers the Microcontroller to one of the three fixed program locations shown below. In each of these addresses, the user will normally store a JMP instruction to the interrupt service routine for that particular interrupt. Details of these operations are described later.

INT 0 ..... Address 4

INT 1 ..... Address 5

INT 2 ..... Address 6

Return from Interrupt Service Routine

Upon completion of the interrupt service routine, the user codes the special RETURN instruction. When executed, the ICC performs the following steps:

- The return address is popped from the LIFO stack
- The ICC disables program storage and forces a JMP instruction onto the Microcontroller instruction bus with the return address from the stack. (The HALT signal is driven low for one instruction cycle.)
- The JMP instruction from the ICC transfers the Microcontroller to the instruction that was about to execute at the time the interrupt was taken.

A typical structure for a user program which handles interrupts is shown in the following example:

ADDRESS	INSTRUCTION		COMMENT
0		(any)	First instruction executed after system reset
•		•	
•		•	
3		JMP MAIN	Jump around interrupt vector locations.
4		JMP SERV0	Service INT 0 interrupt.
5		JMP SERV1	Service INT 1 interrupt.
6		JMP SERV2	Service INT 2 interrupt.
7	MAIN	(any)	Continue main program.
•		•	
•		•	
	SERV0	(any)	Begin INT 0 service routine.
•		•	
•		•	
		MAIN R6,R6	ICC RETURN instruction. End INT 0 service routine (resume main program execution).
	SERV1	(any)	Begin INT 1 service routine.
•		•	
•		•	
		MOVE R6,R6	RETURN from INT 1 service routine.
	SERV2	(any)	Begin INT 2 service routine.
•		•	
•		•	
		MOVE R6,R6	RETURN from INT 2 service routine.

Table 1. Instruction Set For ICC

INSTRUCTION	INSTRUCTION CODES		8X305 OPERATION	DESCRIPTION OF ICC OPERATION
	Assembler	Binary		
		I_0 = MSB ↓ I_{15} = LSB ↓		
SET MASK	MOVE R5,R5	000 00101 000 00101	NOP	When executed, sets interrupt mask, thus inhibiting all interrupt servicing.
CLEAR MASK	MOVE R4,R4	000 00100 000 00100	NOP	When executed, clears interrupt mask for all interrupts.
RETURN	MOVE R6,R6	000 00110 000 00110	NOP	When executed returns program to address at top of LIFO stack.
PUSH	MOVE R3,R3	000 00011 000 00011	NOP	Pushes "address + 2" onto stack if PUSH is programmed on odd address in program memory and "address + 1" if PUSH is programmed on even address.
CLEAR INTERRUPT	MOVE R2,R2	000 00010 000 00010	NOP	Clears all interrupt requests; an interrupt service routine that is in progress is unaffected.

Instruction Set

The five instructions shown in Table 1 allow the user to efficiently manage both the interrupt and subroutine capabilities of the ICC in an 8X310/8X305-based system. When an ICC instruction appears in the program, it is interpreted as a NOP by the 8X305 but is captured and decoded by the ICC to perform the desired function. The capture-and-decode functions of the chip are automatic. Assembly and object codes for each ICC instruction are shown in Table 1.

Interrupt Masking Operations

Certain operations performed by the ICC and also some system considerations require that program execution not be interrupted for a specified interval of time. The servicing of interrupts by the ICC can be inhibited in a number of ways. Any time interrupts are inhibited, the ICC ignores any latched interrupt requests. However, the interrupt request latches are not cleared so that any previously pending requests remain latched. Also, during an interval when interrupt servicing is inhibited, any new interrupt signals received will get latched. As soon as interrupt servicing is enabled, any latched requests can be serviced on a priority basis.

The primary means of inhibiting interrupt servicing is the internal one-bit mask (latch). This mask can be set (to inhibit interrupts) or cleared under control of the user program using the special ICC instructions SET MASK and CLEAR MASK — See Table 1. With these instructions, segments of the user program can be isolated so as to proceed without interruptions. Frequently, uninterruptable segments are needed at the very beginning of the user program (initialization routine) and at the beginning of, or throughout an interrupt service routine. To facilitate this, the ICC automatically sets the mask whenever the Microcontroller executes address zero (typically resulting from a system reset) and whenever the ICC services an interrupt. The ICC also automatically clears the mask after performing a RETURN operation from an interrupt service routine; a RETURN from a subroutine does not affect the status of the interrupt mask.

The Interrupt Disable (ID) input pin may also be used to inhibit interrupt servicing. Interrupt servicing remains disabled as long as a high level is applied to the input. The ID input has no effect, however, on the status of the internal interrupt mask.

To ensure proper program flow, the ICC suspends interrupt servicing momentarily during certain situations. During the cycle in which the Microcontroller encounters and XEC (Execute) instruction an interrupt will not be serviced. This is because the XEC causes the Microcontroller to issue an address of an instruction to be executed out of the sequence of normal program flow. This would not be a valid address.

Interrupts are also suspended during execution of a PUSH or RETURN instruction and the instruction immediately following. This ensures proper operation of the LIFO stack. In addition, no interrupts are latched or serviced and no special ICC instructions are decoded at address zero which resets the ICC.

Subroutine Calling

The ICC provides for subroutine calling by storing the proper return address into the LIFO stack under control of the user program. Two instructions are required to implement a subroutine call — a PUSH instruction executed by the ICC and a JMP to the subroutine executed by the Microcontroller. The PUSH instruction is normally programmed at an odd-numbered address in program memory immediately followed by the JMP. When the PUSH instruction is executed, the ICC finds the address of the next instruction (JMP to subroutine) on the Microcontroller's address bus, internally changes the least-significant bit to one (effectively adds one to the address) and stores this into the stack. Program execution proceeds normally and the Microcontroller makes the jump to the beginning of the subroutine. The subroutine may be located at any convenient place in program memory.

Upon completion of the subroutine, the user codes the RETURN instruction in the same manner as for an interrupt service routine. At that point, the ICC forces the Microcontroller to resume execution of the main program at the instruction immediately following the JMP-to-subroutine instruction.

The code for a typically subroutine call-and-return is shown in the following example.

ADDRESS	INSTRUCTION	COMMENT
X (any odd-numbered address)	MOVE R3,R3	PUSH instruction initiates subroutine call by causing ICC to push the address X + 2 onto the stack. (The PUSH instruction is interpreted as a NOP by the Microcontroller.)
X + 1 (even)	JMP SUBR	The Microcontroller JMPs to the beginning of the subroutine.
X + 2 (odd)	(any instruction)	Main program execution resumes here after RETURN from subroutine.
•	•	
•	•	
•	•	
SUBR (any address)	(any instruction)	Execution of subroutine starts here.
•	•	
•	•	
•	•	
(any address)	MOVE R6,R6	RETURN instruction causes ICC to transfer program back to X + 2.

Stack Operation

The LIFO stack holds up to four 13-bit program addresses which allows the ICC to return from a subroutine or interrupt service routine. When all four stack locations are filled, the STack Full (STF) output pin is driven high and remains high until a RETURN (or reset) operation occurs. If an additional interrupt is serviced or subroutine called while the stack is full, the stack will overflow and the oldest return address will be overwritten and lost. That is, the stack retains the four most recent entries. After an overflow, the status of the STF output is not valid (until a reset operation occurs).

To prevent an interrupt from overflowing the stack, the user can connect the STF output directly to the Interrupt Disable (ID) input of the ICC. Then, even if the internal mask and priorities permit interrupt servicing, the interrupt request must still wait for the most recent service routine or subroutine to return.

Because subroutine calling is controlled explicitly by the user software, the user can always ensure that subroutine nesting alone could not overflow the stack. However, care must be taken whenever calling a subroutine from within an interrupt service routine since the number of remaining stack locations may vary at the time the interrupt is taken. If, for example, three stack locations are already filled (STF is low) at the time an interrupt is serviced, then a subroutine call executed within the interrupt service routine would cause the stack to overflow and the earliest return address to be lost.

As mentioned earlier, whenever a RETURN operation is performed from an interrupt service routine, the internal interrupt mask is automatically cleared. A RETURN from a subroutine, however, does not affect the status of the mask. To accomplish this, a flag bit is added to each of the four stack locations which records whether each address pushed into the stack is caused by an interrupt or a subroutine call. This flag is read during a RETURN operation to determine whether or not the interrupt mask is cleared. This allows interrupt servicing and subroutine calls to be intermixed in any order.

Initialization

The ICC decodes address zero as a reset command to perform certain initialization functions. (Zero is the first address generated after the Microcontroller is reset.) Specifically, the instruction-bus drivers are placed in a high-impedance state, HALT output is set high, RD output is set low, and all interrupt request latches are cleared. The interrupt mask is set so that any initialization routine by the user will not be interrupted until a CLEAR MASK instruction (MOVE R4,R4) is executed. The LIFO stack is reset to an empty state and the STF output is set Low.

SYSTEM TIMING RELATIONSHIPS

Interrupt Servicing

Interrupt servicing begins at the end of the Microcontroller instruction cycle when the 8X305's MCLK signal goes from Low-to-High. Starting from this point, processing of the interrupt proceeds as follows:

From the rising edge of MCLK 1:

- Interrupt mask is set to inhibit other interrupts.
- HALT output is driven low to stop internal operation of the 8X305 Microcontroller for one instruction cycle; MCLK is unaffected. ROM Disable (RD) output is driven high to disable program memory.

From the falling edge of MCLK 1:

- Takes address of next instruction from address bus and pushes it onto the top of stack to be used as the return address to the main program.

From the rising edge of MCLK 2:

- The ICC forces a JMP onto the instruction bus to one of three fixed vector addresses:

INT 0	Address 4
INT 1	Address 5
INT 2	Address 6

- Releases HALT (High) which allows the Microcontroller to complete the JMP to the above specified vector location in program memory.

From the rising edge of MCLK 3:

- Instruction-bus drivers of the ICC are disabled.
- ROM Disable (RD) pin is cleared (Low) enabling the program memory which resumes control of the instruction bus.

Return Operation

When the interrupt service routine or subroutine is completed, the RETURN instruction initiates the following sequence of events:

From the rising edge of MCLK 1:

- Interrupts are temporarily inhibited through third MCLK cycle.
- HALT output is driven low to stop Microcontroller for one instruction cycle.
- RD output is set high to disable program memory.

From the rising edge of MCLK 2:

- HALT output is driven high (cleared).
- A JMP instruction to address stored at top of LIFO stack is forced onto the instruction bus by the ICC. (The stack is popped.)

From the rising edge of MCLK 3:

- Instruction-bus drivers of the ICC are disabled.
- RD is cleared enabling the program memory.
- If returning from an interrupt service routine (condition recorded in extra stack bit) the interrupt mask is cleared; otherwise the mask remains unaffected.

Once the preceding return actions are completed, the 8X305 Microcontroller will resume execution of the instruction at the return address.

APPLICATION HINTS

- When programming an interrupt service routine or subroutine, certain system operations typically need to be considered. In many interrupt-driven systems, a handshake signal is required to acknowledge the servicing of an interrupt request. The acknowledge signal may be transmitted by the interrupt service routine using a standard I/O port from the 8X300 Family.
- If the user wants to allow a higher priority interrupt request to interrupt a service routine, then the CLEAR MASK instruction should be programmed (perhaps after completing any critical operations).
- For both service routines and subroutines, the user may need to save the contents of some or all of the working registers of the Microcontroller so that operation of the main program is not upset. Registers may be written out to a working storage RAM such as 8X350 near the beginning of the routine, and restored from RAM just before returning to the main program.
- Certain subroutine calling techniques may be used to increase the efficiency of the user program. As shown in the following examples, a subroutine can automatically be repeated two, three or four times, if desired, without programming a loop.

SUBROUTINE AUTOMATICALLY EXECUTES TWICE			
Address			**Instruction**
X(even)	SUBR2	MOVE R3,R3	Push X + 1 onto stack.
X + 1(odd)		(start of subroutine)	
•		•	
•		•	
		MOVE R6,R6	RETURN—First time jumps to X+1; second time jumps back to main program.
SUBROUTINE AUTOMATICALLY EXECUTES THREE TIMES			
X(odd)	SUBR3	MOVE R3,R3	Push X + 2 onto stack.
X + 1(even)		MOVE R3,R3	Push X + 2 onto stack.
X + 2(odd)		(start of subroutine)	
•		•	
•		•	
SUBROUTINE AUTOMATICALLY EXECUTES FOUR TIMES			
X(even)	SUBR4	MOVE R3,R3	Push X + 1 onto stack.
X + 1(odd)		MOVE R3,R3	Push X + 3 onto stack.
X + 2(even)		NOP	
X + 3(odd)		(start of subroutine)	
•		•	
•		•	

- In a manner similar to the Microcontroller multi-way branch technique, one of several subroutines can be selected according to an index value.

SUBROUTINE CALL SELECTED BY VALUE IN R1			
Address			**Instruction**
X(ODD)		MOVE R3,R3	Push X + 2 onto stack.
X + 1(even)		XEC TABLE(R1)	Execute JMP at TABLE + (R1)
X + 2(odd)		(any)	Subroutine returns here.
•		•	
•		•	
(any)	TABLE	JMP SUB0	Call SUB0 if R1 = 0.
		JMP SUB1	Call SUB1 if R1 = 1.
		•	
		•	

ABSOLUTE MAXIMUM RATINGS

SYMBOL	PARAMETER	RATING	UNIT
V_{CC}	Supply voltage	+7	V_{DC}
V_I	Input voltage	+5.5	V_{DC}
V_O	Off-state output voltage	+5.5	V_{DC}
T_{STG}	Storage temperature range	–65 to +150	°C

DC ELECTRICAL CHARACTERISTICS $4.5V \leq V_{CC} \leq 5.5V$, $-55°C \leq T_C \leq +125°C$

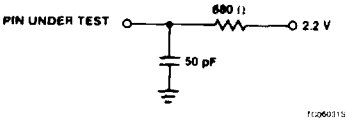
SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
V_{IH}	High level input voltage		2.0			V
V_{IL}	Low level input voltage				0.8	V
V_{OH}	High level output voltage	V_{CC} = MIN.; I_{OH} = –1.0mA	2.5			V
V_{OL}	Low level output voltage	V_{CC} = MIN.; I_{OL} = 2.5mA			0.4	V
V_{IK}	Input clamp-diode voltage	V_{CC} = MIN.; I_{IL} = –18mA			–1.5	V
I_{IH}	High level input current	V_{CC} = MAX; V_{IH} = 2.7V			100	µA
I_{IL}	Low level input current	V_{CC} = MAX; V_{IL} = 0.4V			–700	µA
I_{OS}	Short circuit output current	V_{CC} = MAX.; V_O = 0V	–15		–100	mA
I_{CC}	Supply current	V_{CC} = MAX.; $I_0 - I_{15}$ = Hi-Z				mA
		T_A = 25°C			205	
		T_A = –55°C[2]			230	
		T_C = 125°C			170	

AC ELECTRICAL CHARACTERISTICS $4.5V \leq V_{CC} \leq 5.5V, -55°C \leq T_C \leq +125°C$

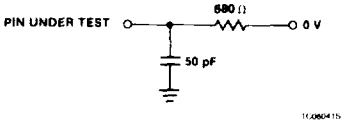
SYMBOL	PARAMETER	REFERENCES		TEST CONDITIONS	LIMITS			UNIT
		From	To		Min	Typ	Max	
Pulse Widths:								
t_{WIH}	Interrupt High	$\uparrow INT_i$	$\downarrow INT_i$		30			ns
t_{WIL}	Interrupt Low	$\downarrow INT_i$	$\uparrow INT_i$		35			ns
t_{WMH}	MCLK High	↑MCLK	↓MCLK	For all Functions	47			ns
Propagation Delays:								
t_{PRH}	RD High	↑MCLK	↑RD	Interrupt or Return			75	ns
t_{PRL}	RD Low	↑MCLK	↓RD	Interrupt or Return			17	ns
t_{PHL}	HALT Low	↑MCLK	↓HALT	Interrupt or Return			87	ns
t_{PHH}	HALT High	↑MCLK	↑HALT	Interrupt or Return			75	ns
t_{PSH}	Stack full High	↓MCLK	↑STF	Interrupt or Subroutine Call			105	ns
t_{PSL}	Stack full Low	↓MCLK	↓STF	Return or Reset			115	ns
Setup Times:								
t_{SIH}	Interrupt input setup[3]	$\uparrow INT_i$	↑MCLK		35			ns
t_{SA}	Address setup	$A_0 - A_{13}$	↑MCLK	Interrupt, Subroutine, Call, or Reset	0			ns
t_{SC}	Instruction setup[5]	I_{0-15}	↓MCLK	All Commands	Note 5			ns
t_{SD}	Interrupt disable setup[3]	ID	↑MCLK		30			ns
Hold and Reset Recovery Times:								
t_{HIL}	Interrupt low input hold[3]	↑MCLK	$\uparrow INT_i$				15	ns
t_{HA}	Address hold	↓MCLK	A_{0-13}	Subroutine Call or Reset			90	ns
t_{HC}	Instruction hold	↓MCLK	I_{0-15}	All Commands			55	ns
t_{HD}	Interrupt disable hold[3]	↑MCLK	ID				25	ns
t_{RI}	Interrupt reset recovery[4]	↓MCLK	$\uparrow INT_i$	Reset or Cancel Command	70			ns
Output Enable/Disable Delays:								
t_{OEC}	Instruction output enable	↑MCLK	I_{0-15}	Interrupt or Return			87	ns
t_{ODC}	Instruction output disable	↑MCLK	I_{0-15}	Interrupt or Return			47	ns

NOTES:
1. All electrical characteristics re guaranteed after power is applied and thermal equilibrium has been reached.
2. The 230mA value is worst case over the entire temperature range.
3. Parameters t_{SIH}, t_{HIL}, t_{SD}, and t_{HD} are used only to determine whether an interrupt request will be serviced during the current or a subsequent instruction cycle. the INT_i and ID inputs are asynchronous and transitions on either input may safely occur at any time with respect to MCLK. A low–to–high transition on INT_i occurring after t_{SIH} and before t_{HIL} means only that it cannot be determined for sure whether or not the interrupt request will be honored during the current instruction cycle. Similarly, transitions on ID between t_{SD} and t_{HD} make it uncertain as to whether or not masking applies during the current instruction cycle.
4. When clearing interrupt requests (including a reset operation), any new low–to–high transitions appearing at the INT_i inputs that occur before t_{RI} risk being cleared and therefore ignored; however, any transition after t_{RI} is certain to be latched.
5. t_{SC} (minimum) = 17ns — t_{PRL} (actual).
(The required instruction enable time for the program memory depends on the sum of the t_{PRL} and t_{SC}.)

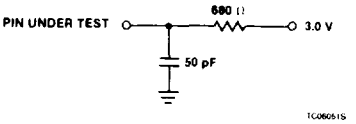
AC TEST CIRCUITS



RD, STF, and $\overline{\text{HALT}}$ Outputs



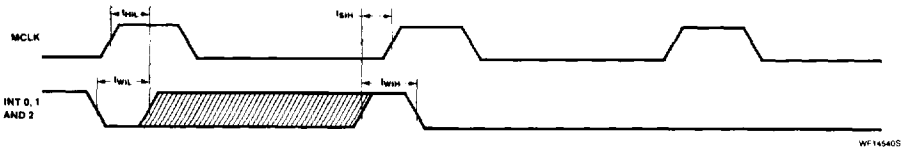
$I_0 - I_{15}$ Outputs (t_{OEC} and t_{ODC}) t_{PZH} and t_{PHZ}



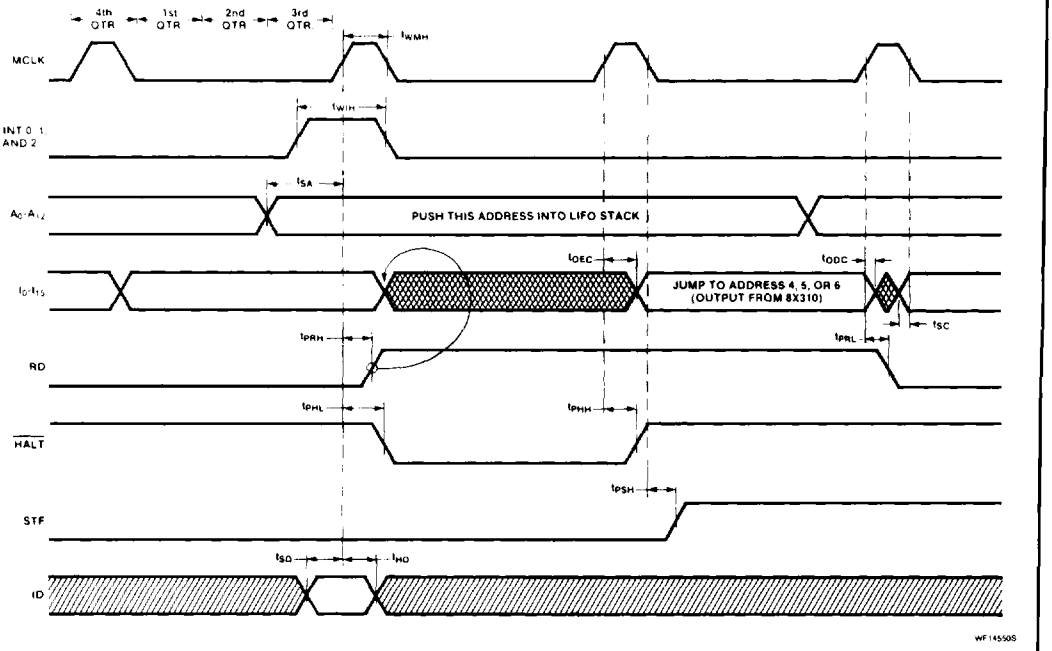
t_{PZL} and t_{PLZ}

NOTES:
1. V_M = 1.5V for all input signals and RD, STF, and HALT outputs.
2. For $I_0 - I_{15}$ (t_{OEC}): V_M = 1.5V
   For $I_0 - I_{15}$ (t_{ODC}): V_M (t_{PLZ}) = V_{OL} + 0.5V
   V_M (t_{PHZ}) = V_{OH} − 0.5V

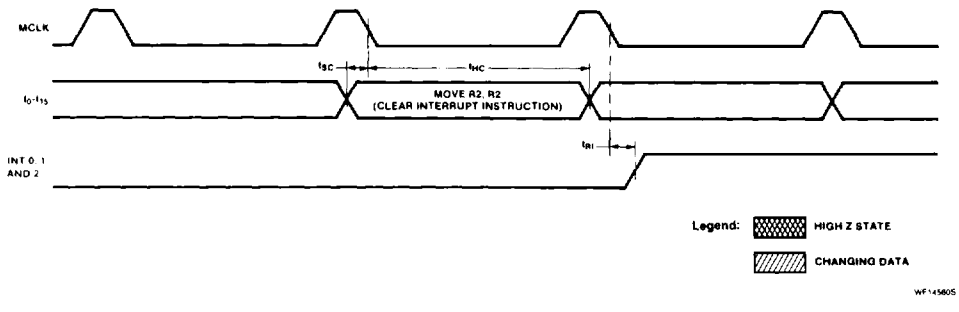
TIMING DIAGRAMS



Interrupt Request Timing

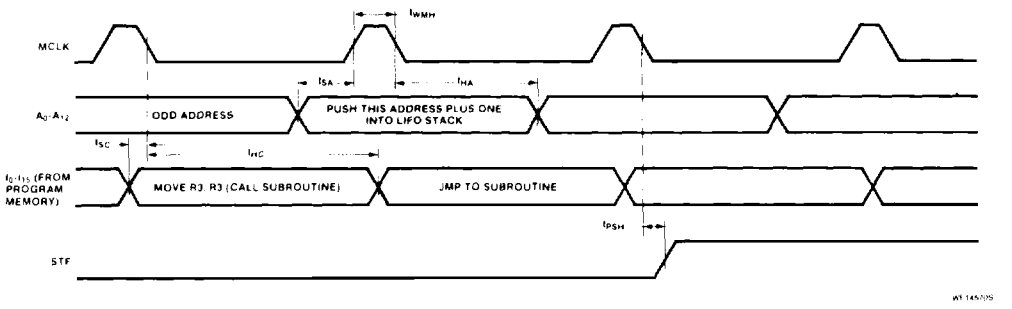


Interrupt Service Timing

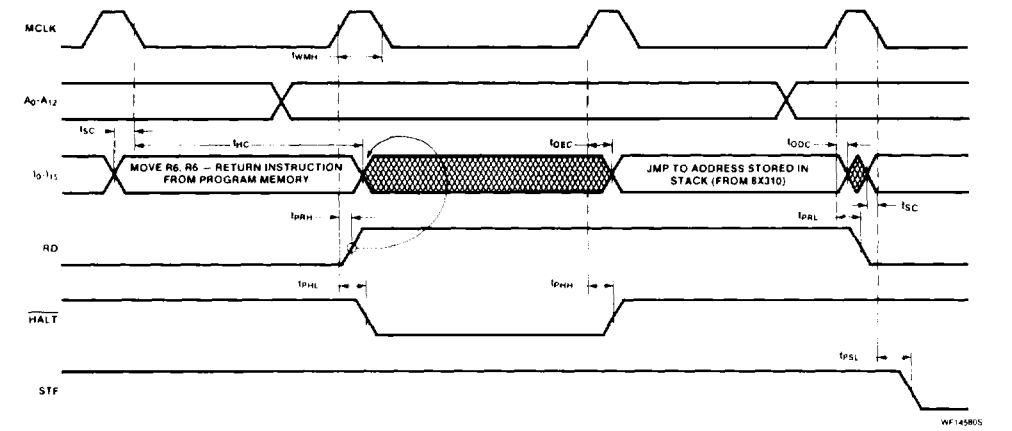


Clear Interrupt Instruction Timing

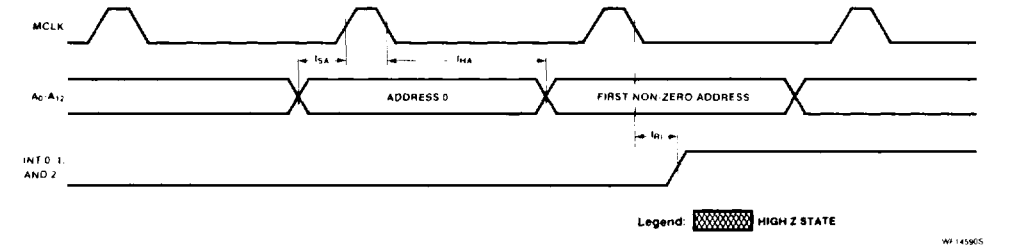
TIMING DIAGRAMS (Continued)



Call Subroutine Timing



Return Timing (From Interrupt or Subroutine)



Reset Timing