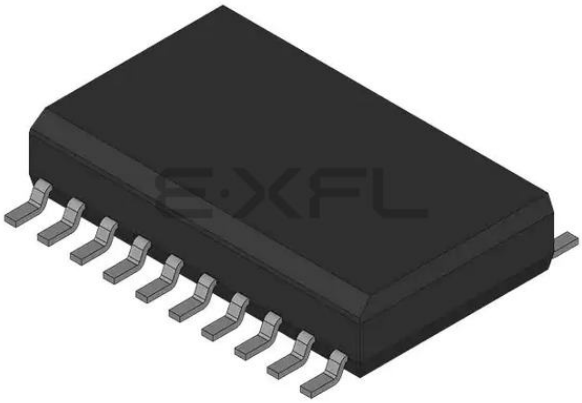


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	HC08
Core Size	8-Bit
Speed	8MHz
Connectivity	-
Peripherals	LED, LVD, POR, PWM
Number of I/O	15
Program Memory Size	4KB (4K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	128 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 3.3V
Data Converters	A/D 12x8b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	20-SOIC (0.295", 7.50mm Width)
Supplier Device Package	20-SOIC
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc68hrc908jk3cdw

6.3 Features

- Object code fully upward-compatible with M68HC05 Family
- 16-bit stack pointer with stack manipulation instructions
- 16-bit index register with x-register manipulation instructions
- 8-MHz CPU internal bus frequency
- 64-Kbyte program/data memory space
- 16 addressing modes
- Memory-to-memory data moves without using accumulator
- Fast 8-bit by 8-bit multiply and 16-bit by 8-bit divide instructions
- Enhanced binary-coded decimal (BCD) data handling
- Modular architecture with expandable internal bus definition for extension of addressing range beyond 64 Kbytes
- Low-power stop and wait modes

6.4 CPU Registers

Figure 6-1 shows the five CPU registers. CPU registers are not part of the memory map.

6.4.2 Index Register

The 16-bit index register allows indexed addressing of a 64-Kbyte memory space. H is the upper byte of the index register, and X is the lower byte. H:X is the concatenated 16-bit index register.

In the indexed addressing modes, the CPU uses the contents of the index register to determine the conditional address of the operand.

The index register can serve also as a temporary data storage location.

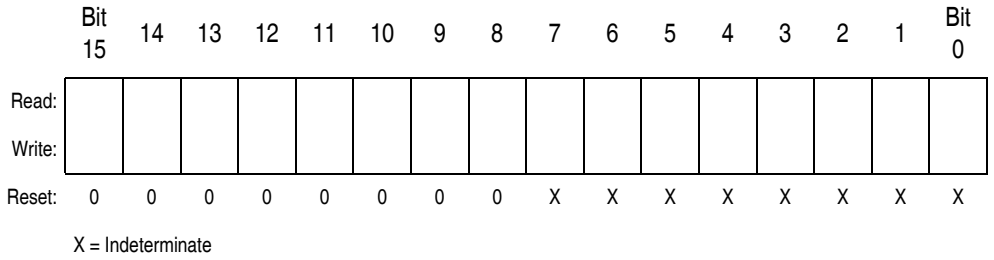


Figure 6-3. Index Register (H:X)

6.4.3 Stack Pointer

The stack pointer is a 16-bit register that contains the address of the next location on the stack. During a reset, the stack pointer is preset to $00FF. The reset stack pointer (RSP) instruction sets the least significant byte to $FF and does not affect the most significant byte. The stack pointer decrements as data is pushed onto the stack and increments as data is pulled from the stack.

In the stack pointer 8-bit offset and 16-bit offset addressing modes, the stack pointer can function as an index register to access data on the stack. The CPU uses the contents of the stack pointer to determine the conditional address of the operand.

Table 6-1. Instruction Set Summary

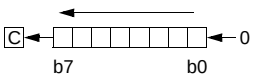
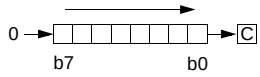
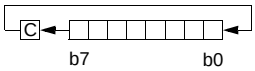
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
EOR *#opr* EOR *opr* EOR *opr* EOR *opr*,X EOR *opr*,X EOR ,X EOR *opr*,SP EOR *opr*,SP	Exclusive OR M with A	A ← (A ⊕ M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A8 B8 C8 D8 E8 F8 9EE8 9ED8	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
INC *opr* INCA INCX INC *opr*,X INC ,X INC *opr*,SP	Increment	M ← (M) + 1 A ← (A) + 1 X ← (X) + 1 M ← (M) + 1 M ← (M) + 1 M ← (M) + 1	↕	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3C 4C 5C 6C 7C 9E6C	dd ff ff	4 1 1 4 3 5
JMP *opr* JMP *opr* JMP *opr*,X JMP *opr*,X JMP ,X	Jump	PC ← Jump Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BC CC DC EC FC	dd hh ll ee ff ff 	2 3 4 3 2
JSR *opr* JSR *opr* JSR *opr*,X JSR *opr*,X JSR ,X	Jump to Subroutine	PC ← (PC) + *n* (*n* = 1, 2, or 3) Push (PCL); SP ← (SP) – 1 Push (PCH); SP ← (SP) – 1 PC ← Unconditional Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BD CD DD ED FD	dd hh ll ee ff ff 	4 5 6 5 4
LDA *#opr* LDA *opr* LDA *opr* LDA *opr*,X LDA *opr*,X LDA ,X LDA *opr*,SP LDA *opr*,SP	Load A from M	A ← (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A6 B6 C6 D6 E6 F6 9EE6 9ED6	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
LDHX *#opr* LDHX *opr*	Load H:X from M	H:X ← (M:M + 1)	0	–	–	↕	↕	–	IMM DIR	45 55	ii jj dd	3 4
LDX *#opr* LDX *opr* LDX *opr* LDX *opr*,X LDX *opr*,X LDX ,X LDX *opr*,SP LDX *opr*,SP	Load X from M	X ← (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	AE BE CE DE EE FE 9EEE 9EDE	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
LSL *opr* LSLA LSLX LSL *opr*,X (Same as ASL) LSL ,X LSL *opr*,SP	Logical Shift Left		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	4 1 1 4 3 5

Table 6-1. Instruction Set Summary

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
LSR *opr* LSRA LSR*X* LSR *opr*,X LSR ,X LSR *opr*,SP	Logical Shift Right		↕	–	–	0	↕	↕	DIR INH INH IX1 IX SP1	34 44 54 64 74 9E64	dd ff ff	4 1 1 4 3 5
MOV *opr,opr* MOV *opr,*X+ MOV #*opr,opr* MOV X+,*opr*	Move	$(M)_{Destination} \leftarrow (M)_{Source}$ H:X ← (H:X) + 1 (IX+D, DIX+)	0	–	–	↕	↕	–	DD DIX+ IMD IX+D	4E 5E 6E 7E	dd dd dd ii dd dd	5 4 4 4
MUL	Unsigned multiply	X:A ← (X) × (A)	–	0	–	–	–	0	INH	42		5
NEG *opr* NEGA NEGX NEG *opr*,X NEG ,X NEG *opr*,SP	Negate (Two's Complement)	M ← –(M) = $00 – (M) A ← –(A) = $00 – (A) X ← –(X) = $00 – (X) M ← –(M) = $00 – (M) M ← –(M) = $00 – (M)	↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	30 40 50 60 70 9E60	dd ff ff	4 1 1 4 3 5
NOP	No Operation	None	–	–	–	–	–	–	INH	9D		1
NSA	Nibble Swap A	A ← (A[3:0]:A[7:4])	–	–	–	–	–	–	INH	62		3
ORA #*opr* ORA *opr* ORA *opr* ORA *opr*,X ORA *opr*,X ORA ,X ORA *opr*,SP ORA *opr*,SP	Inclusive OR A and M	A ← (A) | (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	AA BA CA DA EA FA 9EEA 9EDA	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
PSHA	Push A onto Stack	Push (A); SP ← (SP) – 1	–	–	–	–	–	–	INH	87		2
PSHH	Push H onto Stack	Push (H); SP ← (SP) – 1	–	–	–	–	–	–	INH	8B		2
PSHX	Push X onto Stack	Push (X); SP ← (SP) – 1	–	–	–	–	–	–	INH	89		2
PULA	Pull A from Stack	SP ← (SP + 1); Pull (A)	–	–	–	–	–	–	INH	86		2
PULH	Pull H from Stack	SP ← (SP + 1); Pull (H)	–	–	–	–	–	–	INH	8A		2
PULX	Pull X from Stack	SP ← (SP + 1); Pull (X)	–	–	–	–	–	–	INH	88		2
ROL *opr* ROLA ROLX ROL *opr*,X ROL ,X ROL *opr*,SP	Rotate Left through Carry		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	39 49 59 69 79 9E69	dd ff ff	4 1 1 4 3 5

At the beginning of an interrupt, the CPU saves the CPU register contents on the stack and sets the interrupt mask (I bit) to prevent additional interrupts. At the end of an interrupt, the RTI instruction recovers the CPU register contents from the stack so that normal processing can resume. **Figure 7-9** shows interrupt entry timing. **Figure 7-10** shows interrupt recovery timing.

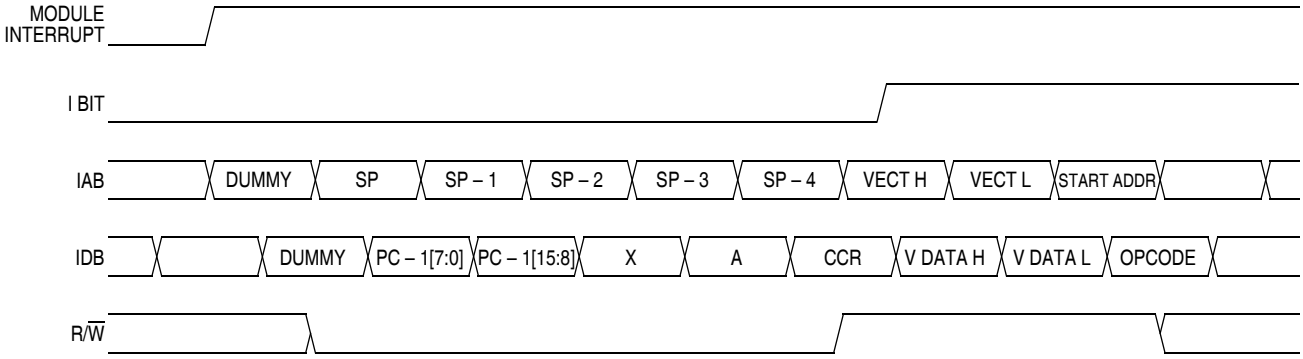


Figure 7-9. Interrupt Entry

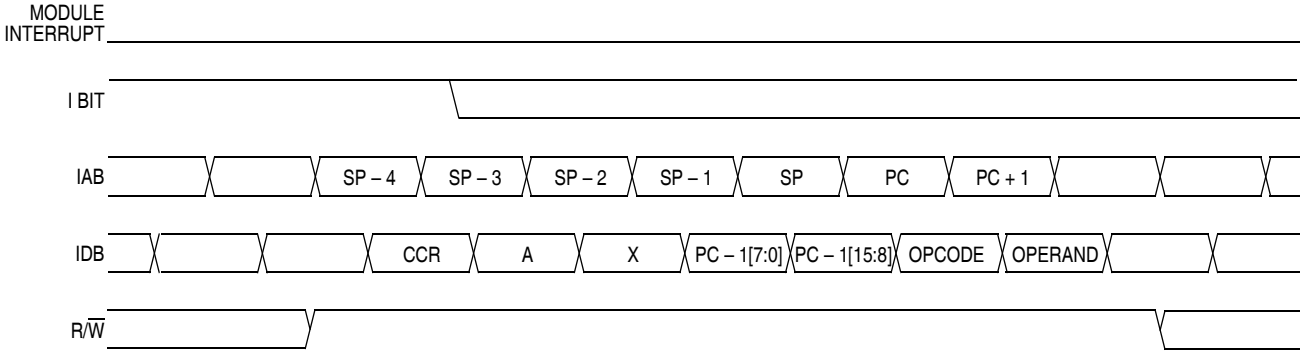


Figure 7-10. Interrupt Recovery

7.6.1.1 Hardware Interrupts

A hardware interrupt does not stop the current instruction. Processing of a hardware interrupt begins after completion of the current instruction. When the current instruction is complete, the SIM checks all pending hardware interrupts. If interrupts are not masked (I bit clear in the condition code register), and if the corresponding interrupt enable bit is

7.6.2.3 Interrupt Status Register 3

Address: $FE06

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	0	0	IF15
Write:	R	R	R	R	R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 7-14. Interrupt Status Register 3 (INT3)

IF15 — Interrupt Flags

These flags indicate the presence of interrupt requests from the sources shown in **Table 7-3**.

1 = Interrupt request present
0 = No interrupt request present

Bit 1 to 7 — Always read 0

7.6.3 Reset

All reset sources always have equal and highest priority and cannot be arbitrated.

7.6.4 Break Interrupts

The break module can stop normal program flow at a software-programmable break point by asserting its break interrupt output. (See **Section 17. Break Module (BREAK)**.) The SIM puts the CPU into the break state by forcing it to the SWI vector location. Refer to the break interrupt subsection of each module to see how each module is affected by the break state.

7.6.5 Status Flag Protection in Break Mode

The SIM controls whether status flags contained in other modules can be cleared during break mode. The user can select whether flags are

protected from being cleared by properly initializing the break clear flag enable bit (BCFE) in the break flag control register (BFCR).

Protecting flags in break mode ensures that set flags will not be cleared while in break mode. This protection allows registers to be freely read and written during break mode without losing status flag information.

Setting the BCFE bit enables the clearing mechanisms. Once cleared in break mode, a flag remains cleared even when break mode is exited. Status flags with a two-step clearing mechanism — for example, a read of one register followed by the read or write of another — are protected, even when the first step is accomplished prior to entering break mode. Upon leaving break mode, execution of the second step will clear the flag as normal.

7.7 Low-Power Modes

Executing the WAIT or STOP instruction puts the MCU in a low-power-consumption mode for standby situations. The SIM holds the CPU in a non-clocked state. The operation of each of these modes is described below. Both STOP and WAIT clear the interrupt mask (I) in the condition code register, allowing interrupts to occur.

7.7.1 Wait Mode

In wait mode, the CPU clocks are inactive while the peripheral clocks continue to run. **Figure 7-15** shows the timing for wait mode entry.

A module that is active during wait mode can wake up the CPU with an interrupt if the interrupt is enabled. Stacking for the interrupt begins one cycle after the WAIT instruction during which the interrupt occurred. In wait mode, the CPU clocks are inactive. Refer to the wait mode subsection of each module to see if the module is active or inactive in wait mode. Some modules can be programmed to be active in wait mode.

Wait mode can also be exited by a reset or break. A break interrupt during wait mode sets the SIM break stop/wait bit, SBSW, in the break

Table 9-2 is a summary of the vector differences between user mode and monitor mode.

Table 9-2. Monitor Mode Vector Differences

Modes	Functions						
	COP	Reset Vector High	Reset Vector Low	Break Vector High	Break Vector Low	SWI Vector High	SWI Vector Low
User	Enabled	$FFFE	$FFFF	$FFFC	$FFFD	$FFFC	$FFFD
Monitor	Disabled(1)	$FEFE	$FEFF	$FEFC	$FEFD	$FEFC	$FEFD

Notes:
1. If the high voltage (V_{DD} + V_{HI}) is removed from the $\overline{IRQ1}$ pin or the $\overline{RST}$ pin, the SIM asserts its COP enable output. The COP is a mask option enabled or disabled by the COPD bit in the configuration register.

When the host computer has completed downloading code into the MCU RAM, the host then sends a RUN command, which executes an RTI, which sends control to the address on the stack pointer.

9.4.2 Baud Rate

The communication baud rate is dependant on oscillator frequency. The state of PTB3 also affects baud rate if entry to monitor mode is by $\overline{IRQ1}$ = V_{DD} + V_{HI}. When PTB3 is high, the divide by ratio is 1024. If the PTB3 pin is at logic zero upon entry into monitor mode, the divide by ratio is 512.

Table 9-3. Monitor Baud Rate Selection

Monitor Mode Entry By:	Input Clock Frequency	PTB3	Baud Rate
$\overline{IRQ1}$ = V_{DD} + V_{HI}	4.9152 MHz	0	9600 bps
	9.8304 MHz	1	9600 bps
	4.9152 MHz	1	4800 bps
Blank reset vector, $\overline{IRQ1}$ = V_{DD}	9.8304 MHz	X	9600 bps
	4.9152 MHz	X	4800 bps

Address: $0021 TCNTH

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
Write:								
Reset:	0	0	0	0	0	0	0	0

Address: $0022 TCNTL

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 10-5. TIM Counter Registers (TCNTH:TCNTL)

10.10.3 TIM Counter Modulo Registers (TMODH:TMODL)

The read/write TIM modulo registers contain the modulo value for the TIM counter. When the TIM counter reaches the modulo value, the overflow flag (TOF) becomes set, and the TIM counter resumes counting from $0000 at the next clock. Writing to the high byte (TMODH) inhibits the TOF bit and overflow interrupts until the low byte (TMODL) is written. Reset sets the TIM counter modulo registers.

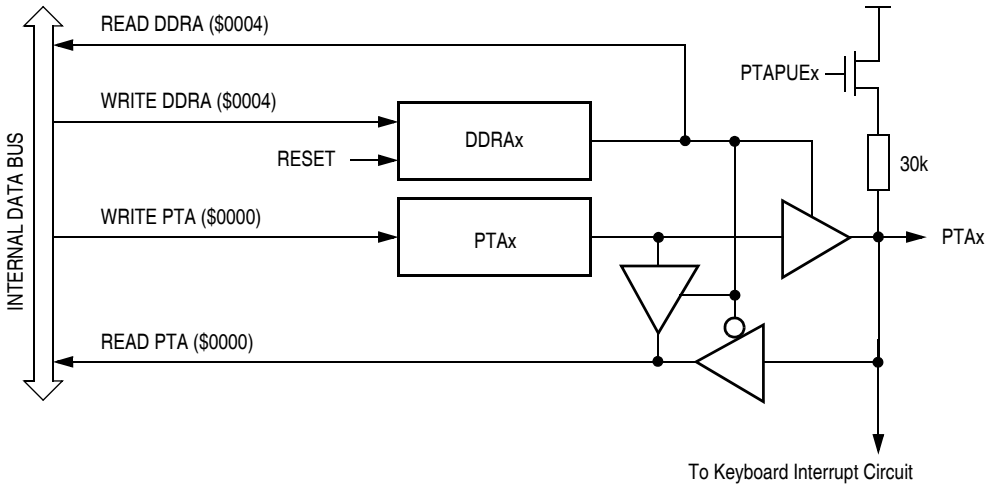


Figure 12-4. Port A I/O Circuit

When DDRAx is a logic 1, reading address $0000 reads the PTAx data latch. When DDRAx is a logic 0, reading address $0000 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit.

12.4.2 Port A Input Pull-up Enable Register (PTAPUE)

The Port A Input Pull-up Enable Register (PTAPUE) contains a software configurable pull-up device for each if the seven port A pins. Each bit is individually configurable and requires the corresponding data direction register, DDRAx be configured as input. Each pull-up device is automatically and dynamically disabled when its corresponding DDRAx bit is configured as output.

DDRB[7:0] — Data Direction Register B Bits

These read/write bits control port B data direction. Reset clears DDRB[7:0], configuring all port B pins as inputs.

1 = Corresponding port B pin configured as output

0 = Corresponding port B pin configured as input

NOTE: *Avoid glitches on port B pins by writing to the port B data register before changing data direction register B bits from 0 to 1.* ***Figure 12-8*** *shows the port B I/O logic.*

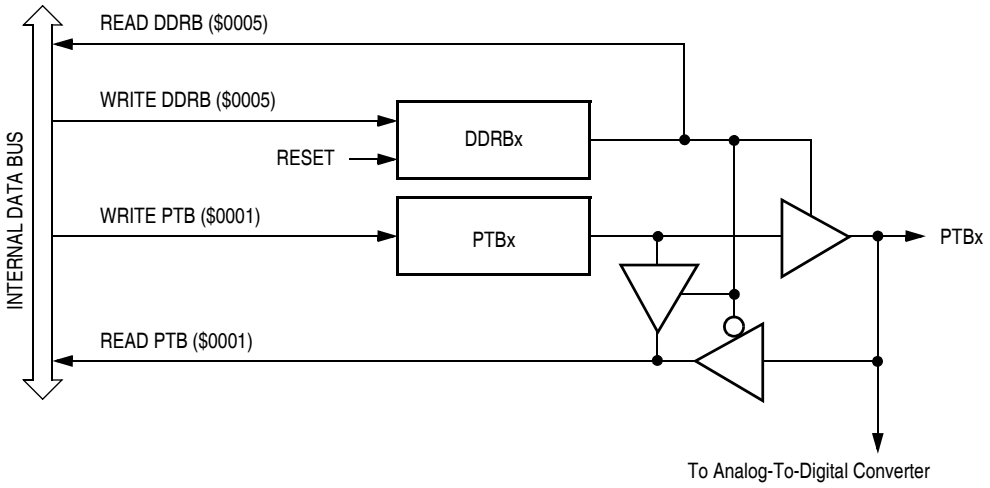


Figure 12-8. Port B I/O Circuit

When DDRBx is a logic 1, reading address $0001 reads the Hotbox data latch. When DDRBx is a logic 0, reading address $0001 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. **Table 12-2**summarizes the operation of the port B pins.

Table 12-2. Port B Pin Functions

DDRB Bit	PTB Bit	I/O Pin Mode	Accesses to DDRB	Accesses to PTB	
			Read/Write	Read	Write
0	X(1)	Input, Hi-Z(2)	DDRB7-DDRB0	Pin	PTB[7:0](3)
1	X	Output	DDRB7-DDRB0	Pin	PTB[7:0]

1. X = don't care
2. Hi-Z = high impedance
3. Writing affects data register, but does not affect the input.

IRQF1 — IRQ1 Flag

This read-only status bit is high when the IRQ1 interrupt is pending.

1 = $\overline{\text{IRQ1}}$ interrupt pending

0 = $\overline{\text{IRQ1}}$ interrupt not pending

ACK1 — IRQ1 Interrupt Request Acknowledge Bit

Writing a logic one to this write-only bit clears the IRQ1 latch. ACK1 always reads as logic zero. Reset clears ACK1.

IMASK1 — IRQ1 Interrupt Mask Bit

Writing a logic one to this read/write bit disables IRQ1 interrupt requests. Reset clears IMASK1.

1 = IRQ1 interrupt requests disabled

0 = IRQ1 interrupt requests enabled

MODE1 — IRQ1 Edge/Level Select Bit

This read/write bit controls the triggering sensitivity of the $\overline{\text{IRQ1}}$ pin. Reset clears MODE1.

1 = $\overline{\text{IRQ1}}$ interrupt requests on falling edges and low levels

0 = $\overline{\text{IRQ1}}$ interrupt requests on falling edges only

Address: $001E

	Bit 7	6	5	4	3	2	1	Bit 0
Read: / Write:	IRQPUD	R	R	LVIT1	LVIT0	R	R	R
Reset:	0	0	0	Not affected	Not affected	0	0	0
POR:	0	0	0	0	0	0	0	0

R = Reserved

Figure 13-4. Configuration Register 2 (CONFIG2)

IRQPUD — $\overline{\text{IRQ1}}$ Pin Pull-up control bit

1 = Internal pull-up is disconnected

0 = Internal pull-up is connected between $\overline{\text{IRQ1}}$ pin and V_{DD}

Section 15. Computer Operating Properly (COP)

15.1 Contents

15.2 Introduction

The computer operating properly (COP) module contains a free-running counter that generates a reset if allowed to overflow. The COP module helps software recover from runaway code. Prevent a COP reset by clearing the COP counter periodically. The COP module can be disabled through the COPD bit in the CONFIG1 register.

15.5 COP Control Register

The COP control register is located at address $FFFF and overlaps the reset vector. Writing any value to $FFFF clears the COP counter and starts a new timeout period. Reading location $FFFF returns the low byte of the reset vector.

Address: $FFFF

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Low byte of reset vector							
Write:	Clear COP counter							
Reset:	Unaffected by reset							

Figure 15-3. COP Control Register (COPCTL)

15.6 Interrupts

The COP does not generate CPU interrupt requests.

15.7 Monitor Mode

The COP is disabled in monitor mode when V_{DD} + V_{HI} is present on the $\overline{IRQ1}$ pin or on the $\overline{RST}$ pin.

15.8 Low-Power Modes

The WAIT and STOP instructions put the MCU in low-power consumption standby modes.

15.8.1 Wait Mode

The COP continues to operate during wait mode. To prevent a COP reset during wait mode, periodically clear the COP counter in a CPU interrupt routine.

18.8 5V Oscillator Characteristics

Table 18-6. Oscillator Component Specifications (5V)

Characteristic	Symbol	Min	Typ	Max	Unit
Crystal frequency, XTALCLK	$f_{OSCXCLK}$	—	10	32	MHz
RC oscillator frequency, RCCLK	f_{RCCLK}	2	10	12	MHz
External clock reference frequency(1)	$f_{OSCXCLK}$	dc	—	32	MHz
Crystal load capacitance(2)	C_L	—	—	—	
Crystal fixed capacitance(2)	C_1	—	$2 \times C_L$	—	
Crystal tuning capacitance(2)	C_2	—	$2 \times C_L$	—	
Feedback bias resistor	R_B	—	10 MΩ	—	
Series resistor(2), (3)	R_S	—	—	—	
RC oscillator external R	R_{EXT}	See **Figure 18-1**			
RC oscillator external C	C_{EXT}	—	10	—	pF

NOTES:
1. No more than 10% duty cycle deviation from 50%
2. Consult crystal vendor data sheet
3. Not Required for high frequency crystals

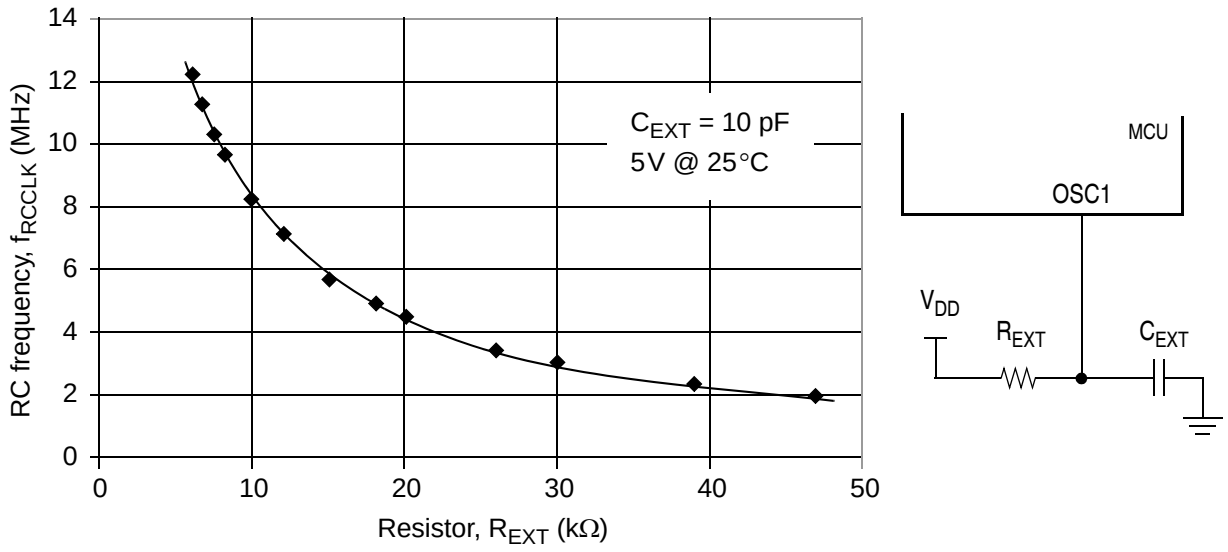
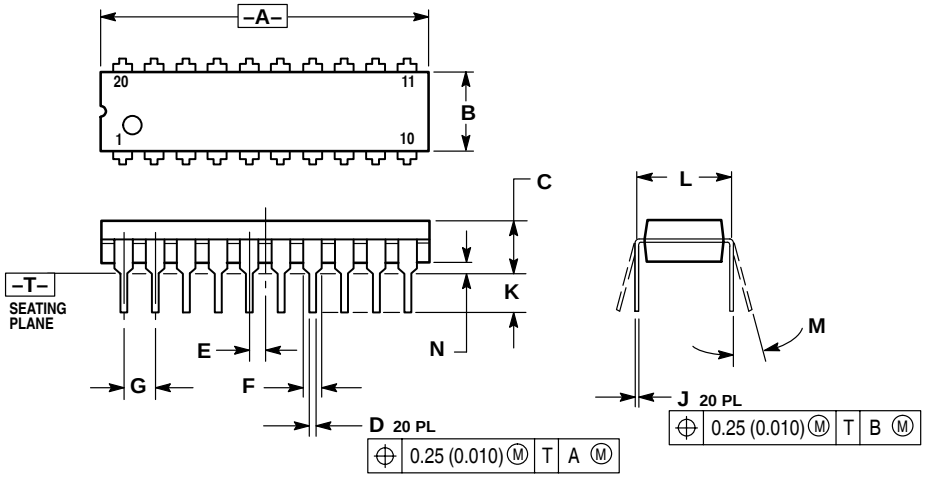


Figure 18-1. RC vs. Frequency (5V @25°C)

19.3 20-Pin PDIP

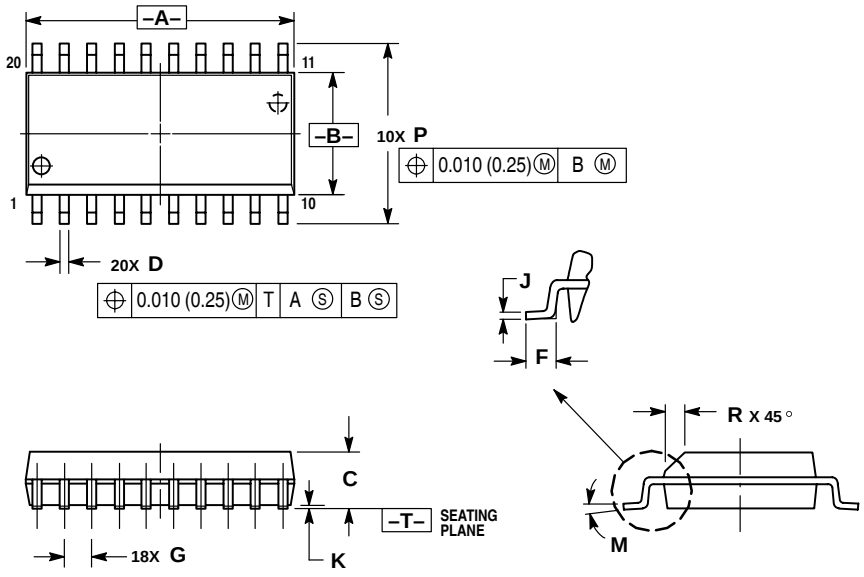


NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSION L TO CENTER OF LEAD WHEN FORMED PARALLEL.
4. DIMENSION B DOES NOT INCLUDE MOLD FLASH.

	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	1.010	1.070	25.66	27.17
B	0.240	0.260	6.10	6.60
C	0.150	0.180	3.81	4.57
D	0.015	0.022	0.39	0.55
E	0.050 BSC		1.27 BSC	
F	0.050	0.070	1.27	1.77
G	0.100 BSC		2.54 BSC	
J	0.008	0.015	0.21	0.38
K	0.110	0.140	2.80	3.55
L	0.300 BSC		7.62 BSC	
M	0 °	15 °	0 °	15 °
N	0.020	0.040	0.51	1.01

Figure 19-1. 20-Pin PDIP (Case #738)

19.4 20-Pin SOIC



NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. DIMENSIONS A AND B DO NOT INCLUDE MOLD PROTRUSION.
4. MAXIMUM MOLD PROTRUSION 0.150 (0.006) PER SIDE.
5. DIMENSION D DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.13 (0.005) TOTAL IN EXCESS OF D DIMENSION AT MAXIMUM MATERIAL CONDITION.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	12.65	12.95	0.499	0.510
B	7.40	7.60	0.292	0.299
C	2.35	2.65	0.093	0.104
D	0.35	0.49	0.014	0.019
F	0.50	0.90	0.020	0.035
G	1.27 BSC		0.050 BSC	
J	0.25	0.32	0.010	0.012
K	0.10	0.25	0.004	0.009
M	0 °	7 °	0 °	7 °
P	10.05	10.55	0.395	0.415
R	0.25	0.75	0.010	0.029

Figure 19-2. 20-Pin SOIC (Case #751D)

Section 20. Ordering Information

20.1 Contents

20.2 Introduction

This section contains ordering numbers for the MC68H(R)C908JL3, MC68H(R)C908JK3, and MC68H(R)C908JK1.