Technical Summary
8-Bit Microcomputer Unit

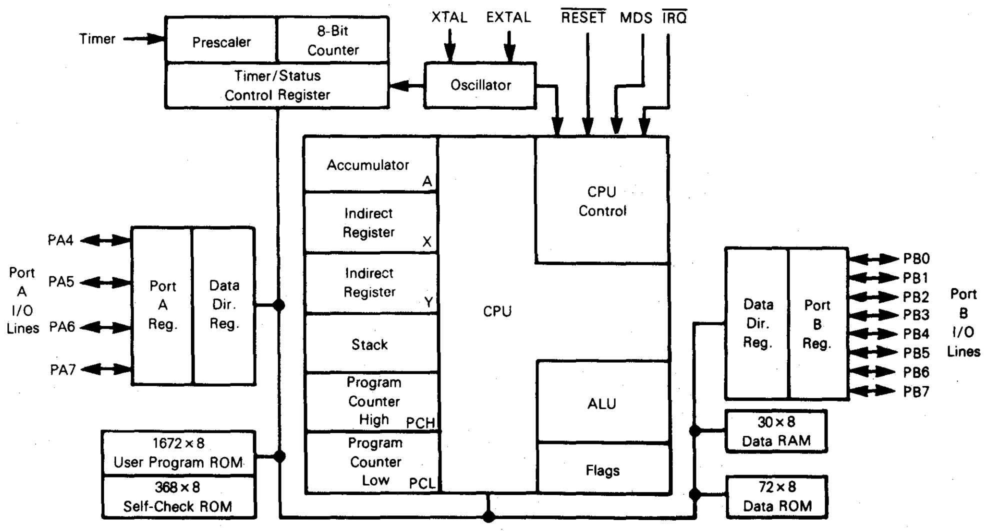
MC68HC04J3 HCMOS microcomputer unit (MCU) device is a member of the M6804 Family of single-chip microcomputers. This device is tremendously versatile and cost effective. These qualities are based on the MCU's simple design and ability to process 8-bit variables, one bit at a time.

This technical summary contains limited information on the MC68HC04J3. For detailed information, refer to the advanced information data sheet for the MC68HC04J2, MC68HC04J3, and MC68HC04P3 8-bit microcomputers (MC68HC04J2/D) or to the *M6804 MCU Manual* (DLE404/D).

Major hardware and software features of the MC68HC04J3 MCU are:

- On-Chip Clock Generator
- Memory Mapped I/O
- Software Programmable 8-Bit Timer with 7-Bit Prescaler
- Single Instruction Memory Examine/Change
- 72 Bytes of Data ROM
- 30 Bytes of User RAM
- User Selectable Input Drive Options
- Optional Pull Down Devices on I/O Ports
- Mask Selectable Edge- or Level-Sensitive Interrupt Pin
- True Bit Manipulation
- Bit Test and Branch Instruction
- 368 Bytes Self-Check ROM
- Conditional Branches
- Timer Pin is Software Programmable as Clock Input or Timer Output
- 1672 Bytes of User Program ROM

BLOCK DIAGRAM



This document contains information on a new product. Specifications and information herein are subject to change without notice.

SIGNAL DESCRIPTION

V_{DD} AND V_{SS}

Power is supplied to the microcomputer using these two pins. V_{DD} is power, and V_{SS} is ground.

$\overline{IRQ}$

This pin provides the capability for asynchronously applying an external interrupt to the microcomputer. A pullup resistor on this pin is a manufacturing mask option.

EXTAL AND XTAL

These pins provide control input for the on-chip clock oscillator circuit. A crystal, a resistor/capacitor combination, or an external signal is connected to these pins to provide a system clock. Selection is made by a manufacturing mask option. The different clock generator options are shown in Figure 1, along with crystal specifications.

Internal Clock Options

The crystal oscillator start-up time is a function of many variables. To ensure rapid oscillator start-up, neither the crystal characteristics nor load capacitances should exceed recommendations. When using the on-board oscillator, the MCU should remain in a reset condition, with the $\overline{RESET}$ pin voltage below V_{IRES+}, until the oscillator has stabilized at its operating frequency.

TIMER

Two TIMER input modes as well as an output mode are available. In the input modes, the TIMER pin is configured as either a TIMER enable, or as the TIMER clock. In the output mode, the TIMER pin may generate transitions upon each occurrence of timer underflow.

$\overline{RESET}$

The $\overline{RESET}$ pin is used to restart the processor to the beginning of a program. The program counter is loaded with the address of the restart vector. This should be a jump instruction to the first instruction of the main program. Together with the MDS pin, the $\overline{RESET}$ pin selects the operating mode of the MCU. A pullup resistor on this pin is a manufacturing mask option.

MDS

The mode select (MDS) pin places the MCU into special operating modes. When this pin is logic high at the exit of the reset state, the decoded state of PA6 and PA7 is latched to determine the operating mode. This choice can be either the single-chip, self-check, or ROM verify mode. However, if MDS is logic low at the end of the reset state, the single-chip operating mode is automatically selected. No external diodes, switches, transistors, etc. are required for single-chip mode selection.

INPUT/OUTPUT LINES (PA4-PA7, PB0-PB7)

These 12 lines are arranged into one 4-bit port (A) and one 8-bit port (B). All lines are programmable as either inputs or outputs under software control of the data direction registers.

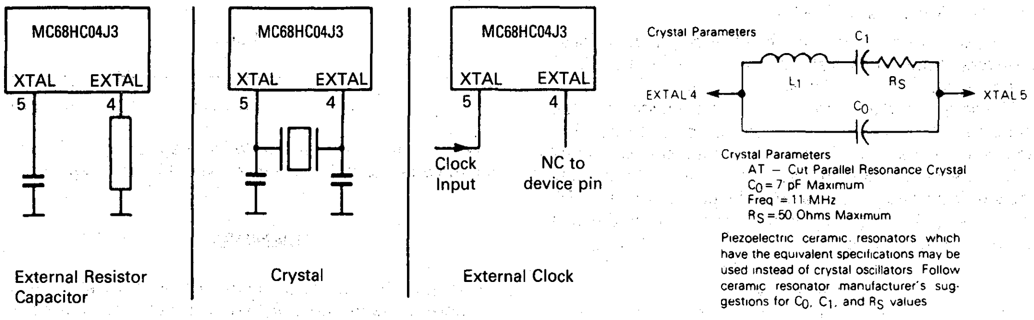
PROGRAMMING

INPUT/OUTPUT PROGRAMMING

There are 12 input/output pins. The 12 bidirectional lines can be selected to have internal pulldowns at the time of manufacture. All pins of each port are programmable as inputs or outputs under the control of the data direction registers (DDR).

The port I/O programming is accomplished by writing the corresponding bit in the port DDR to a logic one for output or a logic zero for input, as shown in Figure 2. When the registers are programmed as outputs, the latched data is readable regardless of the logic levels at the output pin due to output loading.

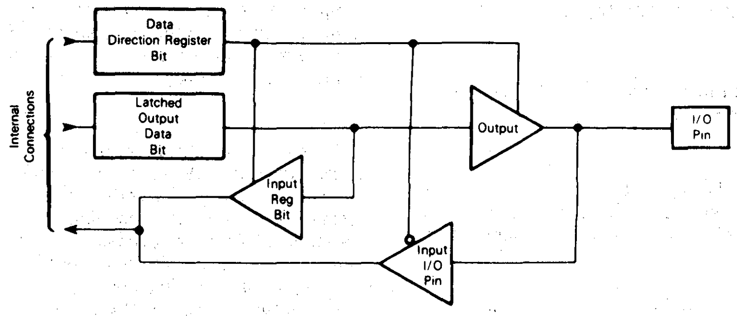
All the I/O pins are CMOS compatible as both inputs and outputs. Their standard configuration as outputs is three-state drive. Port B outputs are LED compatible. In addition, certain pins of both ports may be ordered equipped with pull down resistors.



Crystal Parameters
AT – Cut Parallel Resonance Crystal
C_0 = 7 pF Maximum
Freq = 11 MHz
R_S = 50 Ohms Maximum

Piezoelectric ceramic resonators which have the equivalent specifications may be used instead of crystal oscillators Follow ceramic resonator manufacturer's suggestions for C_0, C_1, and R_S values

Figure 1. Clock Generator Options and Crystal Parameters



Data Direction Register Bit	Output Data Bit	Output State	Input To MCU
1	0	0	0
1	1	1	1
0	X	Hi-Z	Pin

Figure 2. Typical I/O Port Circuitry

Any write to a port writes to all of its data bits even though the port DDR may be set to input. This can be used as a tool to initialize the data registers and avoid undefined outputs. However, care must be exercised when using read-modify-write instructions. The data read corresponds to the pin level if the DDR is an input or to the latched output data when the DDR is an output.

Pull Down Device Option

The use of pull down devices on particular groupings of I/O ports is a manufacturing mask option available to the user. It is of use in applications where keyboards are interfaced directly to the MCU and similar situations. This option is available in the following configurations:

I/O Port	Resistor-Option Pin Groupings
Port A	PA4-PA7
Port B	PB3-PB7, PB4-PB7, PB1-PB2, PB0

Port Data Registers ($00, $01)

The port data registers are not initialized on reset. These registers should be initialized before changing the DDR bits to avoid undefined levels.

The source of data read from the port register is either the port I/O pin or previously latched output data. The source depends upon the contents of the corresponding DDR. The destination of data written to the port data register is an output data latch. If the corresponding DDR for the port I/O pin is programmed as an output, the data appears on the port pin.

Port A ($00)

7	6	5	4	3	2	1	0
				X	X	X	X

Port B ($01)

7	6	5	4	3	2	1	0

With regard to Port A only, the four MSB bits are unused. These are "don't care" (X) bits when written to but are always logic high when read.

Port Data Direction Registers ($04, $05)

Port DDRs configure the port pins as either outputs or inputs. Each port pin can be programmed individually to function as input or output. A zero in the pin's corresponding DDR bit programs it as an input; a logic one programs it as an output. On reset, all the DDRs are initialized to a logic zero state to put the ports in the input mode.

Port A ($04)

7	6	5	4	3	2	1	0
				0	0	0	0

Port B ($05)

7	6	5	4	3	2	1	0

With regard to Port A DDR only, the four MSB bits are cleared after reset. These bits must not be set (logic one).

MEMORY

The MCU memory map (Figure 3) consists of 4352 bytes of addressable memory, I/O register locations, and stack space. This MCU has three separate memory spaces: program space, data space, and stack space.

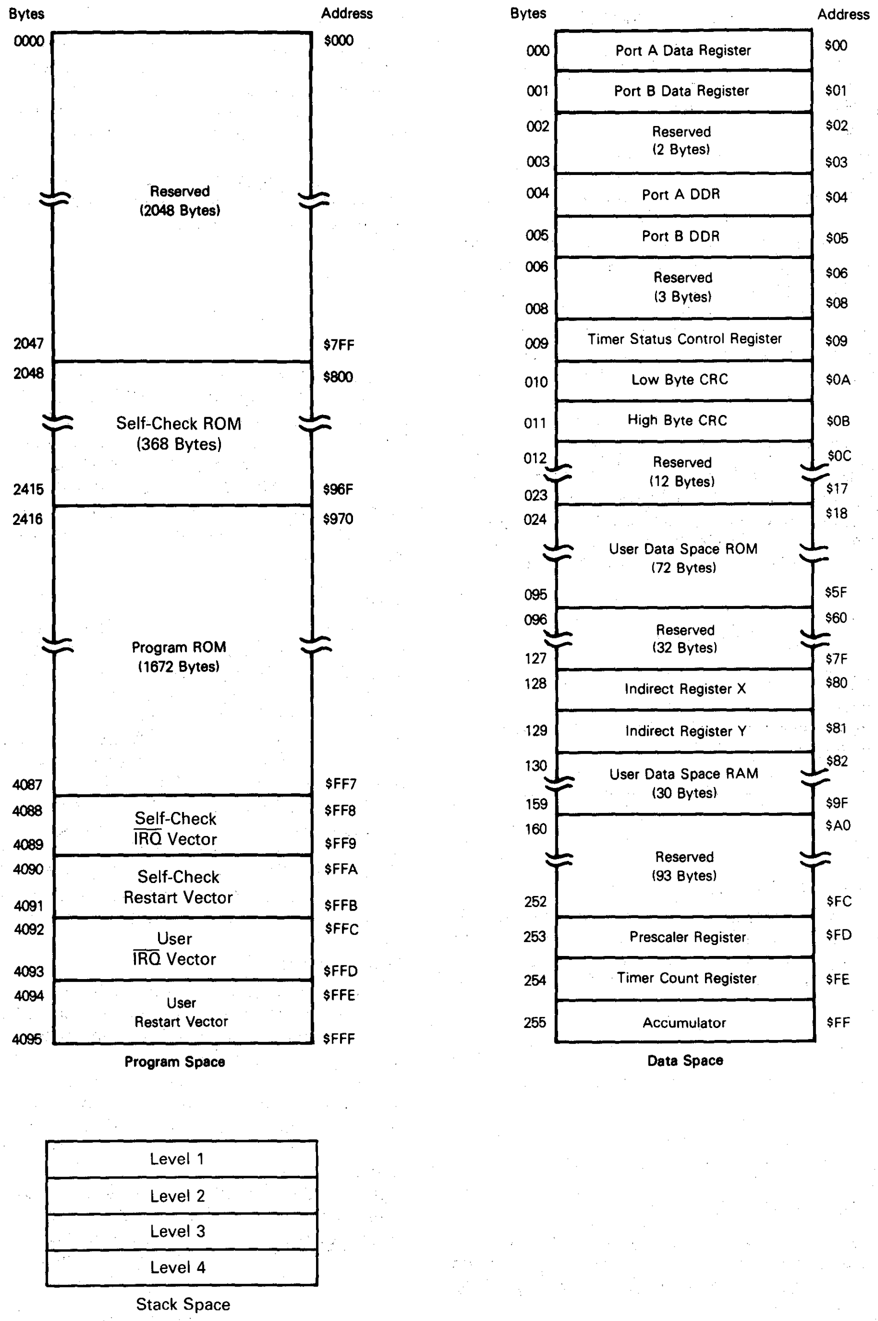


Figure 3. Memory Map

The MCU is capable of addressing 4096 bytes of program space memory with its program counter and 256 bytes of data space memory with its instructions. Program space memory includes self-check ROM, program ROM, self-check and user program vectors, and reserved memory locations.

A non-accessible subroutine stack space RAM is provided. This stack space consists of a last-in-first-out (LIFO) register. This register is used with inherent addressing to stack the return address for subroutines.

Indirect X and Y register locations $80 and $81 are generally used as pointers for such tasks as indirect addressing to data space locations. Short direct addressing allows access to the four data space addresses $80-$83 with single byte opcodes. The operations allowed are increment, decrement, load, and store. Data space locations $82 and $83 can be used for 8-bit counter locations.

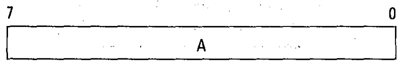
Program ROM Protect

A manufacturing mask option available to the user enables program ROM protection. Enabled, this option prevents the ROM contents from being output during self-check/ROM verify. This option does not prevent a go, no-go test of the ROM contents using the ROM verify mode.

REGISTERS

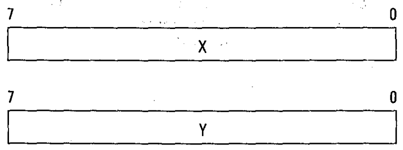
ACCUMULATOR (A)

The accumulator is a general purpose 8-bit register used to hold operands and results of arithmetic calculations or data manipulations.



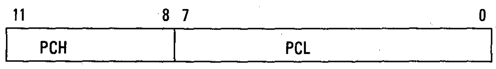
INDIRECT REGISTERS (X,Y)

These two registers are used to maintain pointers to other memory locations in data space. They are used in the register-indirect addressing mode and can be accessed with the direct, indirect, short direct, or bit set/clear modes.



PROGRAM COUNTER (PC)

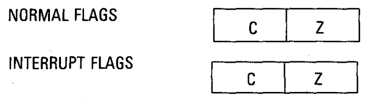
The program counter is a 12-bit register that contains the address of the next byte to be fetched from program space. The program counter is contained in low byte (PCL) and high nibble (PCH).



FLAGS (C,Z)

The first flag, the carry (C) bit, is set on a carry or borrow out of the arithmetic logic unit (ALU). It is cleared if the arithmetic operation does not result in a carry or borrow. The C bit is also set to the value of the bit tested in a bit test instruction. It participates in the rotate left (ROLA) instruction, as well.

The second flag, the zero (Z) bit, is set if the result of the last arithmetic or logic operation was equal to zero. Otherwise, it is cleared. Bit test instructions do not affect the Z bit.



There are two sets of these flags. One set is for interrupt processing (the interrupt mode flags). The other set is for normal operations (the program mode flags). When an interrupt occurs, a context switch is made from the program flags to the interrupt flags. An RTI forces the context switch back. While in either mode, only the flags for that mode are available. A context switch does not affect the value of the C or Z bits. Both sets of flags are cleared by $\overline{\text{RESET}}$.

STACK

A last-in-first-out (LIFO) stack is incorporated in the MCU that eliminates the need for a stack pointer. This non-accessible subroutine stack space is implemented in separate RAM, 12-bits wide. Whenever a subroutine call or interrupt occurs, the contents of the PC are shifted into the top register of the stack. At the same time, the top register is shifted one level deeper. This happens to all registers, with the bottom register falling out of the stack.

Whenever a return from subroutine or interrupt occurs, the top register is shifted into the PC and all lower registers are shifted one level higher. The stack RAM is four levels deep. If the stack is pulled more than four times with no pushes, then the address that was stored in the bottom level of the stack is shifted into the PC.

CRC Registers

Two eight bit registers are implemented in RAM primarily as self-check and ROM verify modes. The two registers are memory mapped in data space at addresses $0A (CRC low), and $0B (CRC high).

Provided no write or read/modify/write operation is used to change the contents of these two locations, the registers are configured to perform CRC calculations. By simply reading a register, a pseudo-random number can be generated.

If a write or a read/modify/write is performed on addresses $0A or $0B, then the CRC circuitry is disabled. Both registers can be used as RAM locations until the next $\overline{\text{RESET}}$. $\overline{\text{RESET}}$ enables the CRC circuitry again.

SELF CHECK

The MCU implements two forms of internal check, self check and ROM verify. Self check performs an extensive functional check of the MCU using a signature analysis technique. ROM verify uses a similar method to check the contents of program ROM.

Self-check mode is selected by holding the MDS and PA7 pins logic high, and PA6 logic low as $\overline{\text{RESET}}$ goes low to high. ROM verify mode is entered by holding MDS, PA7, and PA6 logic high as $\overline{\text{RESET}}$ goes low to high. Unimplemented program space ROM locations are also tested. Monitoring the self-check mode's stages for successful completion requires external circuitry.

RESET

The MCU can be reset by initial power up or by external reset input ($\overline{\text{RESET}}$).

POWER-ON-RESET (POR)

During a power-on-reset, the timer is used to count 1920 external clock cycles. This allows the oscillator to stabilize before releasing the internal reset, irrespective of the state of the $\overline{\text{RESET}}$ pin. If the $\overline{\text{RESET}}$ pin is low at the end of the delay, the processor remains in the reset condition.

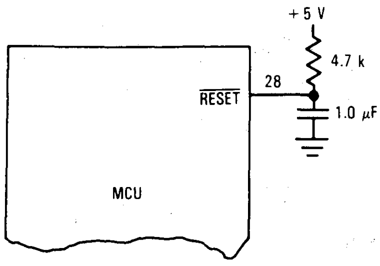


Figure 4. Power Up $\overline{\text{RESET}}$ Delay Circuit

$\overline{\text{RESET}}$

A reset can also be achieved by pulling the $\overline{\text{RESET}}$ pin to logic low for a minimum of two clock cycles. The delay is not implemented in this case.

INTERRUPT

There are two ways this MCU can be interrupted: by applying a logic low signal to the $\overline{\text{IRQ}}$ pin, or by a positive transition of the TMZ bit of TSCR with the ETI bit set. However, a manufacturing mask option determines whether the falling edge or the actual low level of the $\overline{\text{IRQ}}$ pin is sensed to indicate an interrupt.

EXTERNAL INTERRUPT EDGE-SENSITIVE OPTION

When the $\overline{\text{IRQ}}$ pin is pulled low, the internal interrupt request latch is set. Prior to each instruction fetch, this interrupt request latch is tested. If its output is low, an interrupt sequence is initated at the end of the current instruction, provided the interrupt mask is cleared. Figure contains a flowchart that illustrates the interrupt and instruction processing sequences.

The interrupt sequence consists of one cycle during which:

The interrupt request latch is cleared;

The interrupt mode flags are selected;

The program counter (PC) is saved on the stack;

The interrupt mask is set; and

The $\overline{\text{IRQ}}$ vector jump address is loaded into the PC.

The $\overline{\text{IRQ}}$ vector jump address is \$FFC-\$FFD in the single-chip mode and \$FF8-\$FF9 in the self-check mode. The contents of these locations are not decoded as an address to which the PC should jump. Instead, they are decoded like any other ROM word. So, it is essential that the vector contents specify a JMP instruction in addition to the starting address of the interrupt service routine. If required, this routine should save the values of the accumulator and the X and Y registers, since these values are not stored on the stack.

Internal processing of the interrupt continues until a return from interrupt (RTI) instruction is processed. During RTI, the interrupt mask is cleared and the program mode flags are selected. The next instruction of the program is then fetched and executed.

When STOP is processed, the interrupt mask is cleared and the oscillator stopped. Checks are made for either $\overline{\text{RESET}}$ or $\overline{\text{IRQ}}$. If $\overline{\text{RESET}}$ is detected, the $\overline{\text{RESET}}$ sequence is initiated. If $\overline{\text{IRQ}}$ is detected, the system oscillator is enabled along with the clock. In both cases, a delay is executed by the timer to allow oscillator stabilization before the CPU is enabled and the interrupt serviced.

When WAIT is processed, the interrupt mask is cleared and the CPU clock disabled. The interrupt latch is tested. Detection of $\overline{\text{RESET}}$ initiates the $\overline{\text{RESET}}$ sequence. Detection of $\overline{\text{IRQ}}$ or timer interrupt enables the CPU clock and initiates servicing of the interrupts.

When RTI is processed, the program counter is pulled from the stack. The program flags are selected and the interrupt mask cleared. The interrupt latch is then tested before the next instruction.

When the interrupt was initially detected and the interrupt sequence started, the interrupt request latch was cleared so that the next interrupt could be detected. This was done even as the first interrupt was being serviced. However, even though the second interrupt set the interrupt request latch during the first interrupt's processing, the second interrupt's sequence cannot begin until completion of the interrupt service routine for the first interrupt. Completion of an interrupt service routine is always accomplished using an RTI instruction to return to the main program. The interrupt mask, which is not directly available to the programmer, is cleared during the last cycle of the RTI instruction.

EXTERNAL INTERRUPT EDGE/LEVEL-SENSITIVE OPTION

The edge/level-sensitive option performs as described in the preceding section but adds the potential for level-sensitive operation. Level-sensitive operation tests the state of the $\overline{\text{IRQ}}$ and initiates an interrupt service routine if the $\overline{\text{IRQ}}$ pin is found to be logic low.

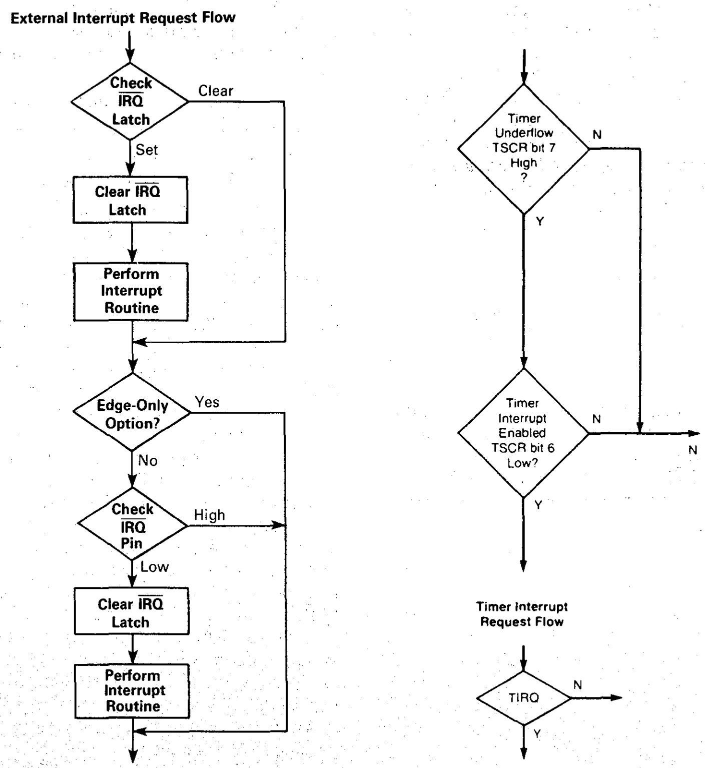

External Interrupt Request Flow
Check IRQ Latch
Clear
Set
Clear IRQ Latch
Perform Interrupt Routine
Edge-Only Option?
Yes
No
Check IRQ Pin
High
Low
Clear IRQ Latch
Perform Interrupt Routine
Timer Underflow TSCR bit 7 High ?
N
Y
Timer Interrupt Enabled TSCR bit 6 Low?
N
Y
N
Timer Interrupt Request Flow
TIRQ
N
Y

Figure 5. Interrupt Sequences

POWER UP AND TIMING

During the power up sequence, the interrupt mask is closed. This precludes any false interrupts. The PC is also loaded with the appropriate restart vector (jump instruction).

To open the interrupt mask, the user should do a JSR to an initialization subroutine that ends with an RTI instead of an RTS. The RTI opens the interrupt mask. Typical $\overline{\text{RESET}}$ and $\overline{\text{IRQ}}$ processes and their relationship to the interrupt mask are shown in Figure 7.

Maximum interrupt response time is eight machine cycles. This includes five cycles for the longest instruction plus one for stacking the PC and switching flags. Two additional cycles are used to synchronize $\overline{\text{IRQ}}$ input with the internal machine cycle frequency.

TIMER INTERRUPT

A timer interrupt is requested by a transition of the TMZ bit of the timer status/control register (TSCR) from logic low to high. Such a positive transition is caused either by the timer count register reaching the all zero state, or by any program instruction that writes a one to the TMZ bit.

The timer interrupt request is maskable by clearing bit 6 of the TSCR (ETI bit). ETI is cleared by $\overline{\text{RESET}}$.

During the interrupt routine, to determine whether an interrupt was caused externally or by the timer, it is necessary to test the state of the TMZ bit in the TSCR.

It is important to service a timer interrupt and clear the TMZ bit before the timer counter underflows again. Otherwise, because only a single interrupt can be latched, there is no way of telling how many timer interrupts occur while the original interrupt is being serviced.

LOW-POWER MODES

STOP

The STOP instruction places the MCU in its lowest power consumption mode. In the STOP mode, the internal oscillator is turned off, causing all internal processing and the timer to be halted. Current consumption is thus dropped to leakage levels.

Providing the supply voltage remains within data sheet limits, the contents of the TSCR, accumulator, and all data space RAM remain unchanged in STOP mode.

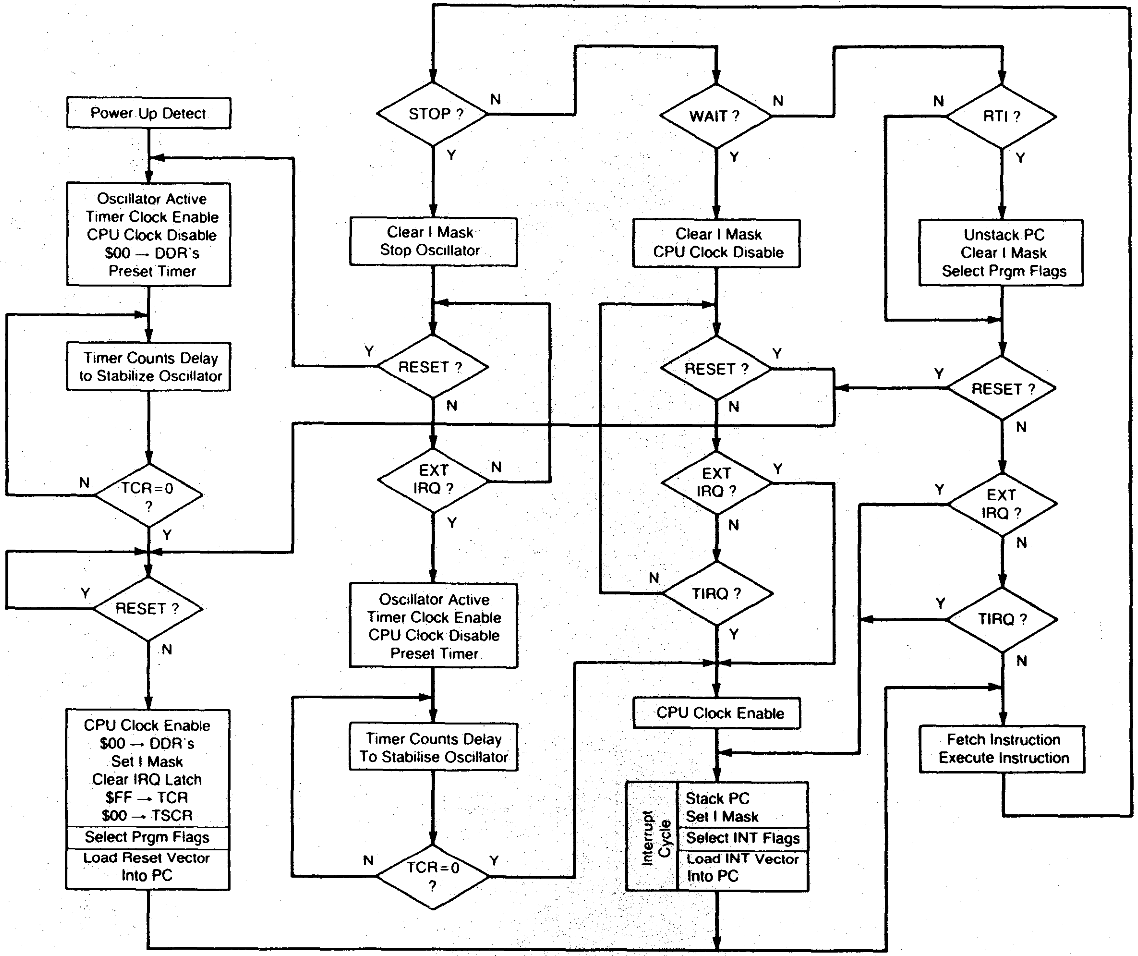


Figure 6. Instruction Processing Sequence

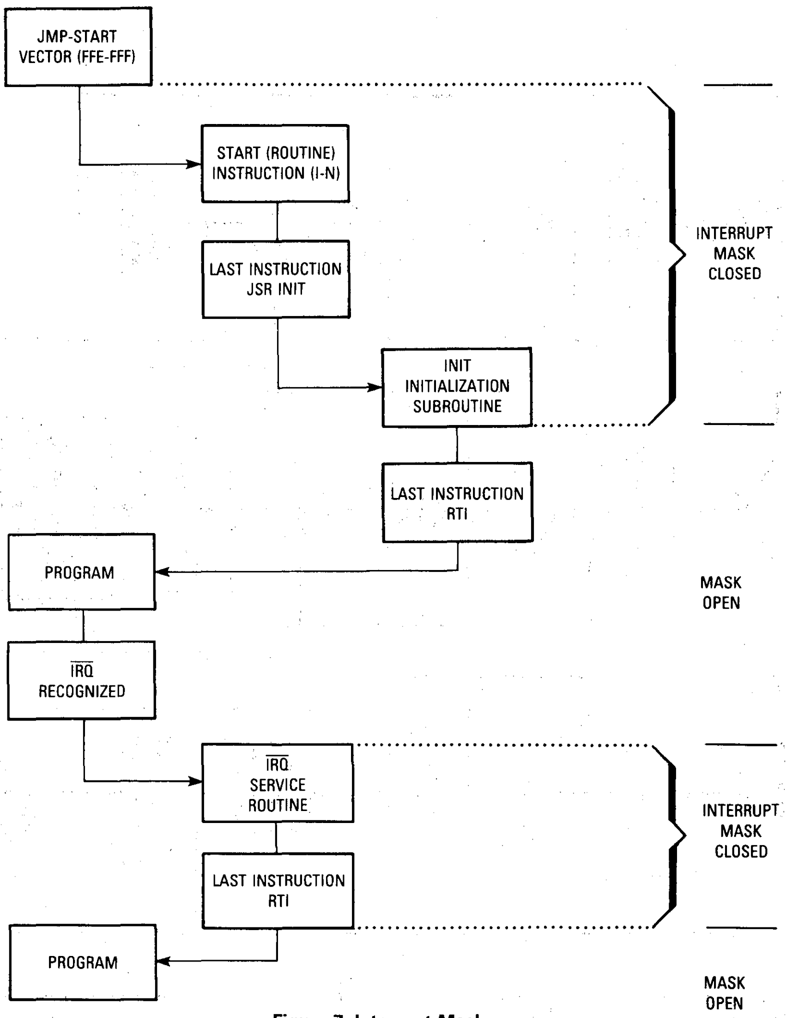

JMP-START VECTOR (FFE-FFF)
START (ROUTINE) INSTRUCTION (I-N)
LAST INSTRUCTION JSR INIT
INIT INITIALIZATION SUBROUTINE
LAST INSTRUCTION RTI
PROGRAM
$\overline{IRQ}$ RECOGNIZED
$\overline{IRQ}$ SERVICE ROUTINE
LAST INSTRUCTION RTI
PROGRAM
INTERRUPT MASK CLOSED
MASK OPEN
INTERRUPT MASK CLOSED
MASK OPEN

Figure 7. Interrupt Mask

Causing an interrupt or reset by pulling the $\overline{RESET}$ or $\overline{IRQ}$ pins low is the only way to bring the processor out of STOP mode. During this exit from STOP, the timer is used to provide the delay time necessary for the oscillator to stabilize. So, the prescaler and timer count register contents must be considered corrupted.

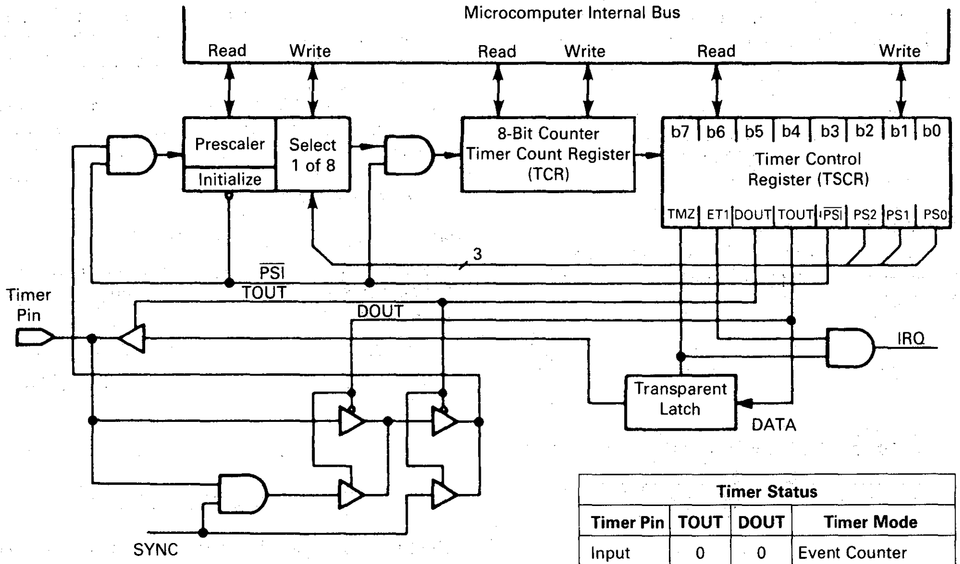
WAIT

The WAIT instruction places the MCU in a low power consumption mode, but the WAIT mode consumes somewhat more power than the STOP mode. In the WAIT mode, the internal clock is disabled from all internal circuitry except for the timer. So, all internal processing is halted. However, the timer continues to decrement normally if the PSI bit of TSCR is set.

During the WAIT mode, external interrupts are enabled. All other registers, memory, and I/O lines remain in their last state. Pulling the $\overline{IRQ}$ or $\overline{RESET}$ pin to logic low causes an exit from the WAIT mode. In addition, ETI bit of TSCR can be enabled by software prior to entering the WAIT state. This allows an exit from WAIT via a timer interrupt as well as via external interrupts.

TIMER

A block diagram of the MC68HC04J3 timer circuitry is shown in Figure 8. The timer logic in the MCU is comprised of a simple 8-bit counter called the timer counter. This counter is decremented by a 7-bit prescaler at a rate determined by the timer status/control register (TSCR).



Timer Status			
Timer Pin	**TOUT**	**DOUT**	**Timer Mode**
Input	0	0	Event Counter
	0	1	Input Gated Mode
Output	1	0	Output
	1	1	

Figure 8. Timer Block Diagram

PRESCALER

The prescaler is a 7-bit counter used to extend the maximum interval of the overall timer. This counter is clocked by a signal from the TIMER pin or by the internal sync pulse. It divides the frequency received by some factor to create the prescaler output. The factor by which the TIMER pin signal is divided is called the prescaler tap. The value of this tap is selected by three bits of the TSCR (PS0-PS2). These bits control the division of the prescaler input within the range of divide-by-2^0, to divide-by-2^7.

TIMER COUNTER

The timer counter, which may be read or loaded under program control, is decremented from a maximum value of 256 toward zero by the prescaler output. Both are decremented on rising clock edges.

The prescaler register and timer count register are readable and writeable. A write to either one will take precedence over the normal counter function. For example, if a value is written to the timer count register and this write and a decrement-to-zero occur at the same time, the write takes precedence and TSCR bit one (TMZ) is not set until the next timer time out.

TIMER PIN

The TIMER pin may be programmed as either an input or an output. Its status depends on the value of TSCR bits 4 (DOUT) and 5 (TOUT). Two distinct input modes exist; input gated mode and input event counter mode. This relationship is shown in the TIMER pin status section of Figure 8. The frequency of the internal clock applied to the TIMER pin must be less than t_{byte}, which is the frequency of the oscillator divided by either 12, 24, or 48, then multiplied by the clock divide ratio. Whether f_{osc} is divided by 12, 24, or 48 is a manufacturing mask option.

TIMER INPUT EVENT COUNTER MODE

In the timer input event counter mode, both TOUT and DOUT are logic zero. The TIMER pin is effectively connected directly to prescaler input. So, the timer/prescaler is clocked by the signal applied from the TIMER pin.

TIMER INPUT GATED MODE

In the input gated mode, TOUT is logic zero and DOUT is logic one. The timer pin is an input which decrements the prescaler each machine cycle as long as the timer pin is logic high. When the pin is logic low, counting is inhibited. This mode permits the counting of the period of time during which the timer pin is logic high, based on the system clock and prescaler values. Gate times are $f_{osc}/12$, $f_{osc}/24$, and $f_{ocs}/48$.

TIMER OUTPUT MODE

In the output mode, TOUT is logic one and the TIMER pin is connected to the DOUT latch. So, the timer prescaler is clocked by the internal sync pulse. This pulse is a divide-by-12, 24 or 48 of the internal oscillator depending on the mask option. However, in the output mode, once the prescaler decrements the timer count register to zero, the low TSCR bit 1 (TMZ) bit state is used to drive the data latched at TSCR bit 4 (DOUT) onto the TIMER pin.

NOTE

TMZ is normally set to logic one when TCR decrements to zero and the timer times out. However, it may be set by a write of $00 to TCR or by a write to bit 7 of TSCR.

TIMER COUNT REGISTER ($FE)

The timer count register reflects the current count in the internal 8-bit counter. The register is the counter and can be written.

7							0
MSB							LSB
RESET:							
1	1	1	1	1	1	1	1

TIMER STATUS/CONTROL REGISTER (TSCR) ($09)

7	6	5	4	3	2	1	0
TMZ	ETI	TOUT	DOUT	$\overline{\text{PSI}}$	PS2	PS1	PS0
RESET:							
0	0	0	0	0	0	0	0

TMZ — Timer Zero
1 = Timer count register has reached the all-zero state since the last time the TMZ bit was read.
0 = This bit is cleared by a read of the TSCR if TMZ is read as logic one.

ETI — Enable Timer Interrupt
1 = Timer interrupt enabled.
0 = Timer interrupt disabled.

TOUT — Timer Output
1 = Output mode is selected for the timer.
0 = Input modes are selected for the timer.

DOUT — Data Output
In the input mode, latched data at this bit is sent to the TIMER pin when both the TMZ and TOUT bits are logic high.
In the input mode:
1 = Timer input gated mode is selected
0 = Timer input event counter mode is selected

$\overline{\text{PSI}}$ — Prescaler Initialization
1 = Prescaler begins to decrement.
0 = Prescaler is initialized and counting is inhibited.

PS0-PS2
These bits are used to select the prescaler tap. The coding of the bits is shown below:

PS2	PS1	PS0	Divide By
0	0	0	1
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

It is recommended that MVI or loading and storing instructions be used when changing bit values in the TSCR. Read-modify-write instructions can cause the TMZ to assume an unexpected state.

During reset, the TSCR is set to all zeroes. The TIMER pin is in the high impedance input mode; and DOUT LATCH is forced to a logic high. At the same time, PS0-PS2 coding sets the prescaler tap at divide-by-one, and bit 3 initializes the prescaler.

TIMER PRESCALER REGISTER ($FD)

The timer prescaler register reflects the current count of the 7-bit prescaler. This register is the prescaler counter and can be written.

6						0
MSB						LSB
RESET:						
1	1	1	1	1	1	1

INSTRUCTION SET

The MCU has a set of 42 basic instructions. They can be divided into five different types: register/memory, read/modify/write, branch, bit manipulation, and control. The following paragraphs briefly explain each type.

REGISTER/MEMORY INSTRUCTIONS

Most of these instructions use two operands. One operand is the accumulator; the other is obtained from memory using one of the addressing modes. Refer to the following list of instructions.

Function	Mnemonic
Load A from Memory	LDA
Load XP from Memory	LDX
Load YP from Memory	LDY
Store A in Memory	STA
Add to A	ADD
Subtract from A	SUB
AND Memory to A	AND
Transfer A to XP	TAX
Transfer A to YP	TAY
Transfer YP to A	TYA
Transfer XP to A	TPA
Clear A	CLRA
Clear XP	CLRX
Clear YP	CLRY
Complement A	COMA
Rotate A Left and Carry	ROLA
Arithmetic Compare with Memory	CMP
Move Immediate Value to Memory	MVI
Arithmetic Left Shift of A	ASLA

READ-MODIFY-WRITE INSTRUCTIONS

These instructions read a memory location or a register, modify or test its contents, and write the modified value back to memory or to the register. All INC and DEC forms along with all bit manipulation instructions use this method. Refer to the following list of instructions.

Function	Mnemonic
Increment Memory Location	INC
Increment A	INCA
Increment XP	INCX
Increment YP	INCY
Decrement Memory Location	DEC
Decrement A	DECA
Decrement XP	DECX
Decrement YP	DECY

BRANCH INSTRUCTIONS

This set of instructions branches if a particular condition is met; otherwise, no operation is performed. Branch instructions are two byte instructions. Refer to the following list of instructions.

Function	Mnemonic
Branch if Carry Clear	BCC
Branch if Higher or Same	(BHS)
Branch if Carry Set	BCS
Branch if Lower	(BLO)
Branch if Not Equal	BNE
Branch if Equal	BEQ

BIT MANIPULATION INSTRUCTIONS

The MCU is capable of setting or clearing any bit which resides in the 256 bytes of the memory space, where all port registers, port DDRs, timer, timer control, and on-chip RAM reside. An additional feature allows the software to test and branch on the state of any bit within these 256 locations. The bit set, bit clear, and bit test and branch functions are all implemented with a single instruction. For the test and branch instructions, the value of the bit tested is also placed in the carry bit of the condition code register. Refer to the following list of instructions.

Function	Mnemonic
Branch If Bit n is Set	BRSET n(n=0 . . . 7)
Branch If Bit n is Clear	BRCLR n(n=0 . . . 7)
Set Bit n	BSET n(n=0 . . . 7)
Clear Bit n	BCLR n(n=0 . . . 7)

CONTROL INSTRUCTIONS

These instructions are used to control processor operation during program execution. The jump conditional (JMP) and jump to subroutine (JSR) instructions have no register operand. Refer to the following list of instructions.

Function	Mnemonic
Return from Subroutine	RTS
Return from Interrupt	RTI
No Operation	NOP
Jump to Subroutine	JSR
Jump Unconditional	JMP
Stop	STOP
Wait	WAIT

IMPLIED INSTRUCTIONS

Since the accumulator and all other registers are located in RAM, many implied instructions exist. Some of the instructions recognized and translated by the assembler are shown below:

Mnemonic	Becomes	Mnemonic	Becomes
ASLA	ADD $FF	INCX	INC $80
BHS	BCC	INCY	INC $81
BLO	BCS	LDXI	MVI $80 DATA
CLRA	SUB $FF	LDYI	MVI $81 DATA
CLRX	MVI $80 #0	NOP	BEQ (PC) +1
CLRY	MVI $81 #0	TAX	STA $80
DECA	DEC $FF	TAY	STA $81
DECX	DEC $80	TXA	LDA $80
DECY	DEC $81	TYA	LDA $81
INCA	INC $FF		

Some examples of valuable instructions not specifically recognized by the assembler are shown below:

Mnemonic	Meaning
BCLR 7,$FF	Ensures A is plus
BSET 7, $FF	Ensures A is minus
BRCLR 7, $FF	Branch if A is plus
BRSET 7, $FF	Branch if A is minus
BRCLR 7, $80	Branch if X is plus (BXPL)
BRSET 7, $80	Branch if X is minus (BXMI)
BRCLR 7, $81	Branch if Y is plus (BYPL)
BRSET 7, $81	Branch if Y is minus (BYMI)

OPCODE MAP

Table 1 is a listing of all the instruction set opcodes applicable to the MC68HC04J3 MCU.

Table 1. Opcode Map

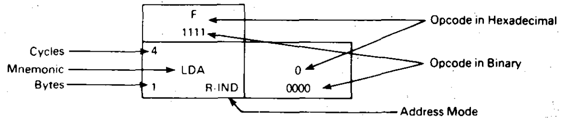
	Branch Instructions								Register/Memory, Control, and Read/Modify/Write Instructions				Bit Manipulation Instructions		Register/Memory and Read/Modify/Write		
Hi / Low	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111	Hi / Low
0 0000	2 BNE 1 REL	2 BNE 1 REL	2 BEQ 1 REL	2 BEQ 1 REL	2 BCC 1 REL	2 BCC 1 REL	2 BCS 1 REL	2 BCS 1 REL	4 JSRn 2 EXT	4 JMPn 2 EXT	•	4 MVI 3 IMM	5 BRCLR0 3 BTB	4 BCLR0 2 BSC	4 LDA 1 R IND	4 LDA 1 R IND	0 0000
1 0001	2 BNE 1 REL	2 BNE 1 REL	2 BEQ 1 REL	2 BEQ 1 REL	2 BCC 1 REL	2 BCC 1 REL	2 BCS 1 REL	2 BCS 1 REL	4 JSRn 2 EXT	4 JMPn 2 EXT	•	•	5 BRCLR1 3 BTB	4 BCLR1 2 BSC	4 STA 1 R IND	4 STA 1 R IND	1 0001
2 0010	2 BNE 1 REL	2 BNE 1 REL	2 BEQ 1 REL	2 BEQ 1 REL	2 BCC 1 REL	2 BCC 1 REL	2 BCS 1 REL	2 BCS 1 REL	4 JSRn 2 EXT	4 JMPn 2 EXT	•	2 RTI 1 INH	5 BRCLR2 3 BTB	4 BCLR2 2 BSC	4 ADD 1 R IND	4 ADD 1 R IND	2 0010
3 0011	2 BNE 1 REL	2 BNE 1 REL	2 BEQ 1 REL	2 BEQ 1 REL	2 BCC 1 REL	2 BCC 1 REL	2 BCS 1 REL	2 BCS 1 REL	4 JSRn 2 EXT	4 JMPn 2 EXT	•	2 RTS 1 INH	5 BRCLR3 3 BTB	4 BCLR3 2 BSC	4 SUB 1 R IND	4 SUB 1 R IND	3 0011
4 0100	2 BNE 1 REL	2 BNE 1 REL	2 BEQ 1 REL	2 BEQ 1 REL	2 BCC 1 REL	2 BCC 1 REL	2 BCS 1 REL	2 BCS 1 REL	4 JSRn 2 EXT	4 JMPn 2 EXT	•	4 COMA 1 INH	5 BRCLR4 3 BTB	4 BCLR4 2 BSC	4 CMP 1 R IND	4 CMP 1 R IND	4 0100
5 0101	2 BNE 1 REL	2 BNE 1 REL	2 BEQ 1 REL	2 BEQ 1 REL	2 BCC 1 REL	2 BCC 1 REL	2 BCS 1 REL	2 BCS 1 REL	4 JSRn 2 EXT	4 JMPn 2 EXT	•	4 ROLA 1 INH	5 BRCLR5 3 BTB	4 BCLR5 2 BSC	4 AND 1 R IND	4 AND 1 R IND	5 0101
6 0110	2 BNE 1 REL	2 BNE 1 REL	2 BEQ 1 REL	2 BEQ 1 REL	2 BCC 1 REL	2 BCC 1 REL	2 BCS 1 REL	2 BCS 1 REL	4 JSRn 2 EXT	4 JMPn 2 EXT	•	2 **STOP** 1 **INH**	5 BRCLR6 3 BTB	4 BCLR6 2 BSC	4 INC 1 R IND	4 INC 1 R IND	6 0110
7 0111	2 BNE 1 REL	2 BNE 1 REL	2 BEQ 1 REL	2 BEQ 1 REL	2 BCC 1 REL	2 BCC 1 REL	2 BCS 1 REL	2 BCS 1 REL	4 JSRn 2 EXT	4 JMPn 2 EXT	•	2 **WAIT** 1 **INH**	5 BRCLR7 3 BTB	4 BCLR7 2 BSC	4 DEC 1 R IND	4 DEC 1 R IND	7 0111
8 1000	2 BNE 1 REL	2 BNE 1 REL	2 BEQ 1 REL	2 BEQ 1 REL	2 BCC 1 REL	2 BCC 1 REL	2 BCS 1 REL	2 BCS 1 REL	4 JSRn 2 EXT	4 JMPn 2 EXT	4 INC 1 SD	4 DEC 1 SD	5 BRSET0 3 BTB	4 BSET0 2 BSC	4 LDA 2 IMM	4 LDA 2 DIR	8 1000
9 1001	2 BNE 1 REL	2 BNE 1 REL	2 BEQ 1 REL	2 BEQ 1 REL	2 BCC 1 REL	2 BCC 1 REL	2 BCS 1 REL	2 BCS 1 REL	4 JSRn 2 EXT	4 JMPn 2 EXT	4 INC 1 SD	4 DEC 1 SD	5 BRSET1 3 BTB	4 BSET1 2 BSC	#	4 STA 2 DIR	9 1001
A 1010	2 BNE 1 REL	2 BNE 2 REL	2 BEQ 1 REL	2 BEQ 1 REL	2 BCC 1 REL	2 BCC 1 REL	2 BCS 1 REL	2 BCS 1 REL	4 JSRn 2 EXT	4 JMPn 2 EXT	4 INC 1 SD	4 DEC 1 SD	5 BRSET2 3 BTB	4 BSET2 2 BSC	4 ADD 2 IMM	4 ADD 2 DIR	A 1010
B 1011	2 BNE 1 REL	2 BNE 1 REL	2 BEQ 1 REL	2 BEQ 1 REL	2 BCC 1 REL	2 BCC 1 REL	2 BCS 1 REL	2 BCS 1 REL	4 JSRn 2 EXT	4 JMPn 2 EXT	4 INC 1 SD	4 DEC 1 SD	5 BRSET3 3 BTB	4 BSET3 2 BSC	SUB IMM	4 SUB 2 DIR	B 1011
C 1100	2 BNE 1 REL	2 BNE 1 REL	2 BEQ 1 REL	2 BEQ 1 REL	2 BCC 1 REL	2 BCC 1 REL	2 BCS 1 REL	2 BCS 1 REL	4 JSRn 2 EXT	4 JMPn 2 EXT	4 LDA 1 SD	4 STA 1 SD	5 BRSET4 3 BTB	4 BSET4 2 BSC	4 CMP 2 IMM	CMP DIR	C 1100
D 1101	2 BNE 1 REL	2 BNE 1 REL	2 BEQ 1 REL	2 BEQ 1 REL	2 BCC 1 REL	2 BCC 1 REL	2 BCS 1 REL	2 BCS 1 REL	4 JSRn 2 EXT	4 JMPn 2 EXT	4 LDA 1 SD	4 STA 1 SD	5 BRSET5 3 BTB	4 BSET5 2 BSC	4 AND 2 IMM	4 AND 2 DIR	D 1101
E 1110	2 BNE 1 REL	2 BNE 1 REL	2 BEQ 1 REL	2 BEQ 1 REL	2 BCC 1 REL	2 BCC 1 REL	2 BCS 1 REL	2 BCS 1 REL	4 JRSn 2 EXT	4 JMPn 2 EXT	4 LDA 1 SD	4 STA 1 SD	5 BRSET6 3 BTB	4 BSET6 2 BSC	#	4 INC 2 DIR	E 1110
F 1111	2 BNE 1 REL	2 BNE 1 REL	2 BEQ 1 REL	2 BEQ 1 REL	2 BCC 1 REL	2 BCC 1 REL	2 BCS 1 REL	2 BCS 1 REL	4 JSRn 2 EXT	4 JMPn 2 EXT	4 LDA 1 SD	4 STA 1 SD	5 BRSET7 3 BTB	4 BSET7 2 BSC	#	4 DEC 2 DIR	F 1111

Abbreviations for Address Modes

INH	Inherent
S-D	Short Direct
B-T-B	Bit Test and Branch
IMM	Immediate
DIR	Direct
EXT	Extended
REL	Relative
BSC	Bit Set/Clear
R-IND	Register Indirect

•	Indicates Instruction Reserved for Future Use
#	Indicates Illegal Instruction

LEGEND



ADDRESSING MODES

The MCU has nine different addressing modes to provide the programmer with an opportunity to optimize the code for all situations. It deals with objects in three different address spaces: program space, data space, and stack space. The term "effective address" (EA) is used in describing the various addressing modes. Effective address is defined as the address from which the argument for an instruction is fetched or stored.

IMMEDIATE

In the immediate addressing mode, the operand is located in program ROM. It is contained in the byte immediately following the opcode. The immediate addressing mode is used to access constants that do not change during program execution, such as a constant used to initialize a loop counter.

DIRECT

In the direct addressing mode, the effective address of the argument is contained in a single byte following the opcode byte. Direct addressing allows the user to directly address the 256 bytes in memory with a single two-byte instruction.

SHORT DIRECT

In the short direct addressing mode, the MCU has four locations in data space RAM which it can use, ($80, $81, $82, and $83). The opcode determines the data space RAM location, and the instruction is only one byte. Short direct addressing is a subset of the direct addressing mode. The X and Y registers are at locations $80 and $81, respectively.

EXTENDED

In the extended addressing mode, the effective address of the argument is obtained by concatenating the four least-significant bits of the opcode with the byte following the opcode to form a 12-bit address. Instructions using the extended addressing mode, such as JMP or JSR, are capable of branching anywhere in program space. An extended addressing mode instruction is two bytes long.

RELATIVE

The relative addressing mode is only used in conditional branch instructions. In relative addressing, the contents of the 8-bit signed byte (the offset) following the opcode is added to the PC if, and only if, the branch conditions are true. Otherwise, control proceeds to the next instruction. The span of relative addressing is from −15 to +16 from the opcode address. The programmer need not calculate the offset when using the Motorola assembler, since it calculates the proper offset and checks to see that it is within the span of the branch.

BIT SET/CLEAR

In the bit set/clear addressing mode, the bit to be set or cleared is part of the opcode. The byte following the opcode specifies the direct addressing of the byte in which the specified bit is to be set or cleared. Thus, any bit in the 256 locations of data space memory that can be written to can be set or cleared with a single two-byte instruction.

CAUTION

The corresponding DDRs for ports A and B are write only registers (registers at $04, $05). A read operation on these registers is undefined. Since BSET and BCLR are read-modify-write functions, they cannot be used to set or clear a DDR bit; all "unaffected" bits would be set. Write all DDR bits in a port using a single-store instruction.

BIT TEST AND BRANCH

The bit test and branch addressing mode is a combination of direct addressing and relative addressing. The bit that is to be tested and its condition (set or clear) is included in the opcode. The data space address of the byte to be tested is in the single byte immediately following the opcode byte. The third byte is sign extended to twelve bits and becomes the offset added to the PC if the condition is true. This single three-byte instruction allows the program to branch based on the condition of any readable bit in the 256 locations of memory. The span of branching is from −125 to +130 from the opcode address. The state of the tested bit is also transferred to the carry flag.

REGISTER-INDIRECT

In the register-indirect addressing mode, the operand is at the address in data space pointed to by the contents of one of the indirect registers, X or Y. The particular indirect register is selected by bit 4 of the opcode. Bit 4 decodes into an address that represents the register, $80 or $81. A register-indirect instruction is one byte long.

INHERENT

In the inherent addressing mode, all the information necessary to execute the instruction is contained in the opcode. These instructions are one byte long.

ELECTRICAL SPECIFICATIONS

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{DD}	−0.3 to +7.0	V
Input Voltage	V_{in}	$V_{SS}-0.3$ to $V_{DD}+0.3$	V
Current Drain per Pin Excluding V_{DD} and V_{SS}	I	10	mA
Total Current for Ports A, B, C EXTAL, TIM — Sink	I	30	mA
Total Current for Ports A, B, C EXTAL, TIM — Source	I	15	
Operating Temperature Range (Comm.)	T_A	0 to 70	°C
Operating Temperature Range (Ind.)	T_A	−40 to +85	°C
Storage Temperature Range	T_{stg}	−55 to +150	°C
Junction Temperature Plastic	T_J	150	°C

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields. However, it is advised that normal precautions be taken to avoid applications of any voltage higher than maximum-rated voltages to this high-impedance circuit. For proper operation it is recommended that V_{in} and V_{out} be constrained to the range $V_{SS} \leq (V_{in}) \leq V_{DD}$. Reliability of operation is enhanced if unused inputs except EXTAL are connected to an appropriate logic voltage level (e.g., either V_{SS} or V_{CC}).

THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Unit
Thermal Resistance Plastic	θ_{JA}	70	°C/W

POWER CONSIDERATIONS

The average chip-junction temperature, T_J, in °C can be obtained from:

$$T_J = T_A + (P_D \cdot \theta_{JA}) \quad (1)$$

where:

T_A	= Ambient Temperature, °C
θ_{JA}	= Package Thermal Resistance, Junction-to-Ambient, °C/W
P_D	= $P_{INT} + P_{PORT}$
P_{INT}	= $I_{CC} \times V_{CC}$, Watts — Chip Internal Power
P_{PORT}	= Port Power Dissipation, Watts — User Determined

For most applications $P_{PORT} < P_{INT}$ and can be neglected. P_{PORT} may become significant if the device is configured to drive Darlington bases or sink LED loads.

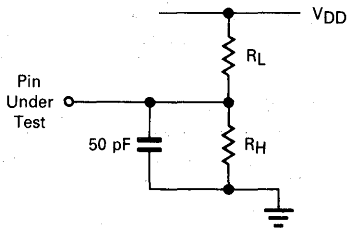
An approximate relationship between P_D and T_J (if P_{PORT} is neglected) is:

$$P_D = K \div (T_J + 273°C) \quad (2)$$

Solving equations (1) and (2) for K gives:

$$K = P_D \cdot (T_A + 273°C) + \theta_{JA} \cdot P_D^2 \quad (3)$$

where K is a constant pertaining to the particular part. K can be determined from equation (3) by measuring P_D (at equilibrium) for a known T_A. Using this value of K, the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A.



V_{DD} = +4.5V
I_{OL}/I_{OH} = 800 µA
$R_L = R_H$ = 4.6 kΩ
V_{DD} = +2.7V
I_{OL}/I_{OH} = 200 µA
$R_L = R_H$ = 10.5 kΩ
V_{DD} = +2.0V
I_{OL}/I_{OH} = 100 µA
$R_L = R_H$ = 16 kΩ

Figure 9. Equivalent Test Load

CONTROL TIMING CHARACTERISTICS

Characteristic	Symbol	Min	Typ	Max	Unit
(V_{DD} = +5 Vdc ±10%, V_{SS} = 0 Vdc; T_A = 0°C to 70°C					
Oscillator Frequency	f_{osc}	0	—	11.0	MHz
PHI1 Clock Frequency	f_{CL}	0	—	5.5	MHz
Cycle Time (Min)	t_{cyc}	2.2	—	—	µs
$\overline{IRQ}$ Pulse Width	t_{IWL}	$2 \times t_{cyc}$	—	—	µs
$\overline{RESET}$ Pulse Width	RWL	$2 \times t_{cyc}$	—	—	µs
Oscillator Clock Pulse Width	t_{OL}, t_{OH}	45	—	—	ns
V_{DD} = +3 Vdc ±10%, V_{SS} = 0 Vdc, T_A = 0°C to 70°C					
Oscillator Frequency	f_{osc}	—	—	11	MHz
PHI1 Clock Frequency	f_{CL}	—	—	4.2	MHz
Cycle Time (Min)	t_{cyc}	2.9	—	—	µs
$\overline{IRQ}$ Pulse Width	t_{IWL}	$2 \times t_{cyc}$	—	—	µs
$\overline{RESET}$ Pulse Width	t_{RWL}	$2 \times t_{cyc}$	—	—	µs
Oscillator Clock Pulse Width	t_{OL}, t_{OH}	45	—	—	ns
V_{DD} = +2.2 Vdc ±10%, V_{SS} = 0 Vdc, T_A = 0°C to 70°C					
Oscillator Frequency	f_{osc}	0	—	8.4	MHz
PHI1 Clock Frequency	f_{CL}	0	—	2.1	MHz
Cycle Time (Min)	t_{cyc}	5.7	—	—	µs
$\overline{IRQ}$ Pulse Width	t_{IWL}	$2 \times t_{cyc}$	—	—	µs
$\overline{RESET}$ Pulse Width	t_{RWL}	$2 \times t_{cyc}$	—	—	µs
Oscillator Clock Pulse Width	t_{OL}, t_{OH}	45	—	—	ns

NOTE: 2 V operation is a user-selectable option only. Prior consultation with the factory is required.

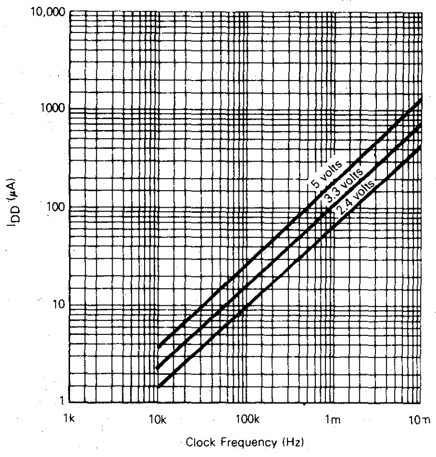


Figure 10. Typical RUN Current vs Clock Frequency

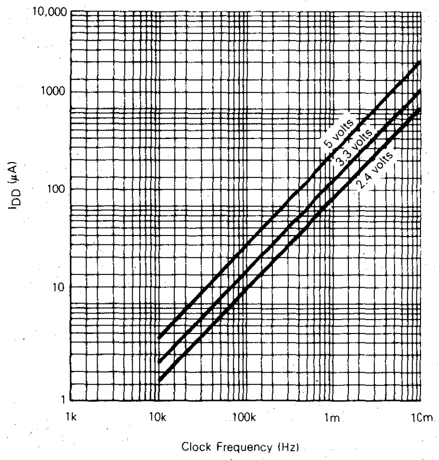


Figure 11. Typical WAIT Current vs Clock Frequency

DC ELECTRICAL CHARACTERISTICS (Typical pull-down sink current for $V_{out}=V_{DD}$ is 50 µA.)

Characteristic	Symbol	Min	Typ	Max	Unit
$V_{DD}=+5$ Vdc ±10%, $V_{SS}=0$ Vdc, $T_A=0°C$ to 70° C					
Output Voltage, $I_{Load}<10.0$ µA	V_{OL} V_{OH}	— $V_{DD}-0.1$	— —	0.1 —	V
Output High Voltage, $I_{Load}=+800$ µA) Ports, TIM Output Low Voltage, $I_{Load}=+800$ µA) Ports, TIM	V_{OH} V_{OL}	$V_{DD}-0.4$ —	— —	— 0.4	V
Input High Voltage Ports, TIM, XTAL, MDS $\overline{\text{IRQ}}$, $\overline{\text{RESET}}$	V_{IH} V_{IH}	$0.7\times V_{DD}$ $0.8\times V_{DD}$	— —	V_{DD} V_{DD}	V
Input Low Voltage Ports, TIM, XTAL, MDS $\overline{\text{IRQ}}$, $\overline{\text{RESET}}$	V_{IL} V_{IL}	V_{SS} V_{SS}	— —	$0.3\times V_{DD}$ $0.2\times V_{DD}$	V
Total Supply Current $C_L=50$ pF, Ports, TIM, RUN No dc load, $t_{cyc}=1/f_{CL}$ (max), WAIT* $V_{IL}=0.2$ V, $V_{IH}=V_{DD}-0.2$ V STOP*	 I_{DD} I_{DD} I_{DD}	 — — —	 2 0.5 3	 3 1 5	 mA mA µA
I/O Ports Input Leakage $V_{SS}<V_I<V_{DD}$	I_{IL}	—	—	±1	µA
Input Current $\overline{\text{RESET}}$, $\overline{\text{IRQ}}$, TIM	I_{in}	—	—	±1	µA
Capacitance per Pin PORTS (as Input or Output) $\overline{\text{RESET}}$, $\overline{\text{IRQ}}$, TIM, XTAL, MDS	C_{out} C_{in}	— —	— —	12 8	pF
$V_{DD}=+3$ Vdc ±10%, $V_{SS}=0$ Vdc, $T_A=0°C$ to 70°C					
Output Voltage, $I_{Load}<10.0$ µA	V_{OL} V_{OH}	— $V_{DD}-0.1$	— —	0.1 —	V
Output High Voltage, $I_{Load}=-200$ µA) Ports, TIM Output Low Voltage, $I_{Load}=+200$ µA) Ports, TIM	V_{OH} V_{OL}	$V_{DD}-0.3$ —	— —	— 0.3	V
Input High Voltage Ports, TIM, XTAL, MDS $\overline{\text{IRQ}}$, $\overline{\text{RESET}}$	V_{IH} V_{IH}	$0.7\times V_{DD}$ $0.8\times V_{DD}$	— —	V_{DD} V_{DD}	V
Input Low Voltage Ports, TIM, MDS, XTAL $\overline{\text{IRQ}}$, $\overline{\text{RESET}}$	V_{IL} V_{IL}	V_{SS} V_{SS}	— —	$0.3\times V_{DD}$ $0.2\times V_{DD}$	V
Total Supply Current $C_L=50$ pF, Ports, TIM, RUN No dc load, $t_{cyc}=1/f_{CL}$(Max), WAIT* $V_{IL}=0.2$ V, $V_{IH}=V_{DD}-0.2$ V STOP*	 I_{DD} I_{DD} I_{DD}	 — — —	 0.8 0.3 1.5	 1.5 0.5 4	 mA mA µA
I/O Ports Input Leakage $V_{SS}<V_I<V_{DD}$	I_{IL}	—	—	±1	µA
Input Current $\overline{\text{RESET}}$, $\overline{\text{IRQ}}$, TIM	I_{in}	—	—	±1	µA
Capacitance per Pin PORTS (as Input or Output) RESET, IRQ, TIM, XTAL, MDS	C_{out} C_{in}	— —	— —	12 8	pF
$V_{DD}=+2.2$ Vdc ±10%, $V_{SS}=0$ Vdc, $T_A=0°C$ to 70°C					
Output Voltage, $I_{Load}<10.0$ µA	V_{OL} V_{OH}	— $V_{DD}-0.1$	— —	0.1 —	V
Output High Voltage, $I_{Load}=-100$µA) Ports, TIM Output Low Voltage, $I_{Load}=+100$ µA) Ports, TIM	V_{OH} V_{OL}	$V_{DD}-0.3$ —	— —	— 0.3	V
Input High Voltage Ports, TIM, XTAL, MDS IRQ, RESET	V_{IH} V_{IH}	$0.7\times V_{DD}$ $0.8\times V_{DD}$	— —	V_{DD} V_{DD}	V
Input Low Voltage Ports, TIM, MDS, XTAL IRQ, RESET	V_{IL} V_{IL}	V_{SS} V_{SS}	— —	$0.3\times V_{DD}$ $0.2\times V_{DD}$	V
Total Supply Current $C_L=50$ pF, Ports, TIM, RUN No dc load, $t_{cyc}=1/f_{CL}$(Max), WAIT* $V_{IL}=0.2$v, $V_{IH}=V_{DD}-0.2$ V, STOP*	 I_{DD} I_{DD} I_{DD}	 — — —	 0.6 0.2 1	 1 0.3 3	 mA mA µA
I/O Ports Input Leakage $V_{SS}<V_I<V_{DD}$	I_{IL}	—	—	±1	µA
Input Current $\overline{\text{REST}}$, $\overline{\text{IRQ}}$, TIM	I_{in}	—	—	±1	µA
Capacitance per Pin PORTS (as Input or Output) RESET, IRQ, TIM, XTAL, MDS	C_{out} C_{in}	— —	— —	12 8	pF

*Measured under the following conditions:
- All ports and timer pin are configured as input
- XTAL is driven by a square wave input
- EXTAL is open circuit
- port pull downs not enabled

NOTE: Typical pull-down sink current for $V_{out}=V_{DD}$ is 50 µA.

ORDERING INFORMATION

The following information is required when ordering a custom MCU. The information may be transmitted to Motorola using the following media:

MDOS, disk file
MS-DOS disk file (360K)
EPROM(s) 2516, 2716, 2532, 2732

To initiate a ROM pattern for the MCU, it is necessary to first contact the local field service office, a sales person, or a Motorola representative.

FLEXIBLE DISKS

Several types of flexible disks (MDOS™ or MS™-DOS disk file), programmed with the customer's program (positive logic sense for address and data), may be submitted for pattern generation. The diskette should be clearly labeled with the customer's name, date, project or product name, and the filename containing the pattern.

In addition to the program pattern, a file containing the program source code listing can be included. This data will be kept confidential and used to expedite the process in case of any difficulty with the pattern file.

MDOS Disk File

MDOS is Motorola's Disk Operating System available on the EXORciser® development system. The disk media submitted must be a single-sided, single-density, 8-inch MDOS compatible floppy diskette. The diskette must contain the minimum set of MDOS system files in addition to the pattern file.

The .LO output of the M6804 cross assembler should be furnished. In addition, the file must be produced (using the ROLLOUT command) containing the absolute image of the M6804 memory. It is necessary to include the entire memory image of both program and data space. All unused bytes, including those in the user space, must be set to logic zero.

MS-DOS/PC-DOS Disk File

MS-DOS is Microsoft's Disk Operating System. Disk media submitted must be standard density (360K), double-sided 5 1/4 inch compatible floppy diskette. The diskette must contain the object file code in Motorola's S–record format. The S–record format is a character-based object file format generated by M6804 cross assemblers and linkers on IBM® PC style machines.

EPROMS

Four K of EPROM are necessary to contain the entire MC68HC04J3 program. Two 2516 or 2716 type EPROMs or a single 2532 or 2732 type EPROM can be submitted for pattern generation. The EPROM is programmed with the customer program using positive logic sense for address and data. Submissions on two EPROMs must be clearly marked. All unused bytes, including the user's space, must be set to zero.

If the MC68HC04J3 MCU ROM pattern is submitted on one 2532 or 2732 EPROM, or on two 2516 or 2716 type EPROMs, memory map addressing is one-for-one. The data space ROM runs from EPROM address $018 to $05F and program space ROM runs from EPROM address $970 to $FF7, with vectors from $FFC to $FFF.

For shipment to Motorola, EPROMs should be placed in a conductive IC carrier and packed securely. Styrofoam is not acceptable for shipment.

Verification Media

All original pattern media, EPROMs or floppy disks, are filed for contractual purposes and are not returned. A computer listing of the ROM code will be generated and returned along with a listing verification form. The listing should be thoroughly checked, and the verification form should be completed, signed, and returned to Motorola. The signed verification form constitutes the contractual agreement for the creation of the customer mask. To aid in the verification process, Motorola will program *customer supplied* blank EPROM(s) or DOS disks from the data file used to create the custom mask.

ROM Verification Units (RVUs)

Ten MCUs containing the customer's ROM pattern will be sent for program verification. These units will have been made using the custom mask but are for the purpose of ROM verification only. For expediency, the MCUs are unmarked, packaged in ceramic, and tested with five volts at room temperature. These RVUs are free with the minimum order quantity but are not production parts. These RVUs are not guaranteed by Motorola Quality Assurance.

Ordering Information

The following table provides generic information pertaining to the package type, temperature, and order numbers for the MC68HC04J3.

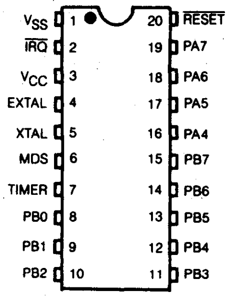
Ordering Information

Package Type	Temperature	Order Number
Plastic (P Suffix)	0°C to 70°C −40°C to +85°C	MC68HC04J3P MC68HC04J3CP

MDOS is a trademark of Motorola Inc.
MS-DOS is a trademark of Microsoft, Inc.
EXORciser is a registered trademark of Motorola Inc.
IBM is a registered trademark of Internal Business Machines Corporation.

MECHANICAL DATA

PIN ASSIGNMENTS



3