1. This technical data may be controlled under U.S. Export Administration Regulations and may be subject to the approval of the U.S. Department of Commerce prior to export. Any export or re-export directly or indirectly, in contravention of the U.S. Export Administration Regulations is strictly prohibited.

2. LIFE SUPPORT POLICY

   Toshiba products described in this databook are not authorized for use as critical components in life support systems without the written consent of the appropriate officer of Toshiba America, Inc. Life support systems are either systems intended for surgical implant in the body or systems which sustain life.

   A critical component is any component of a life support system whose failure to perform may cause a malfunction or failure of the life support system, or may affect its safety or effectiveness.

3. The information in this databook has been carefully checked and is believed to be reliable, however, no responsibility can be assumed for inaccuracies that may not have been caught. All information in this databook is subject to change without prior notice. Furthermore, Toshiba cannot assume responsibility for the use of any license under the patent rights of Toshiba or any third parties.

4. No part of this manual may be transferred or reproduced without prior permission of Toshiba Corporation.

Preface

Thank you very much for making use of TOSHIBA microcomputer LSIs and development systems.

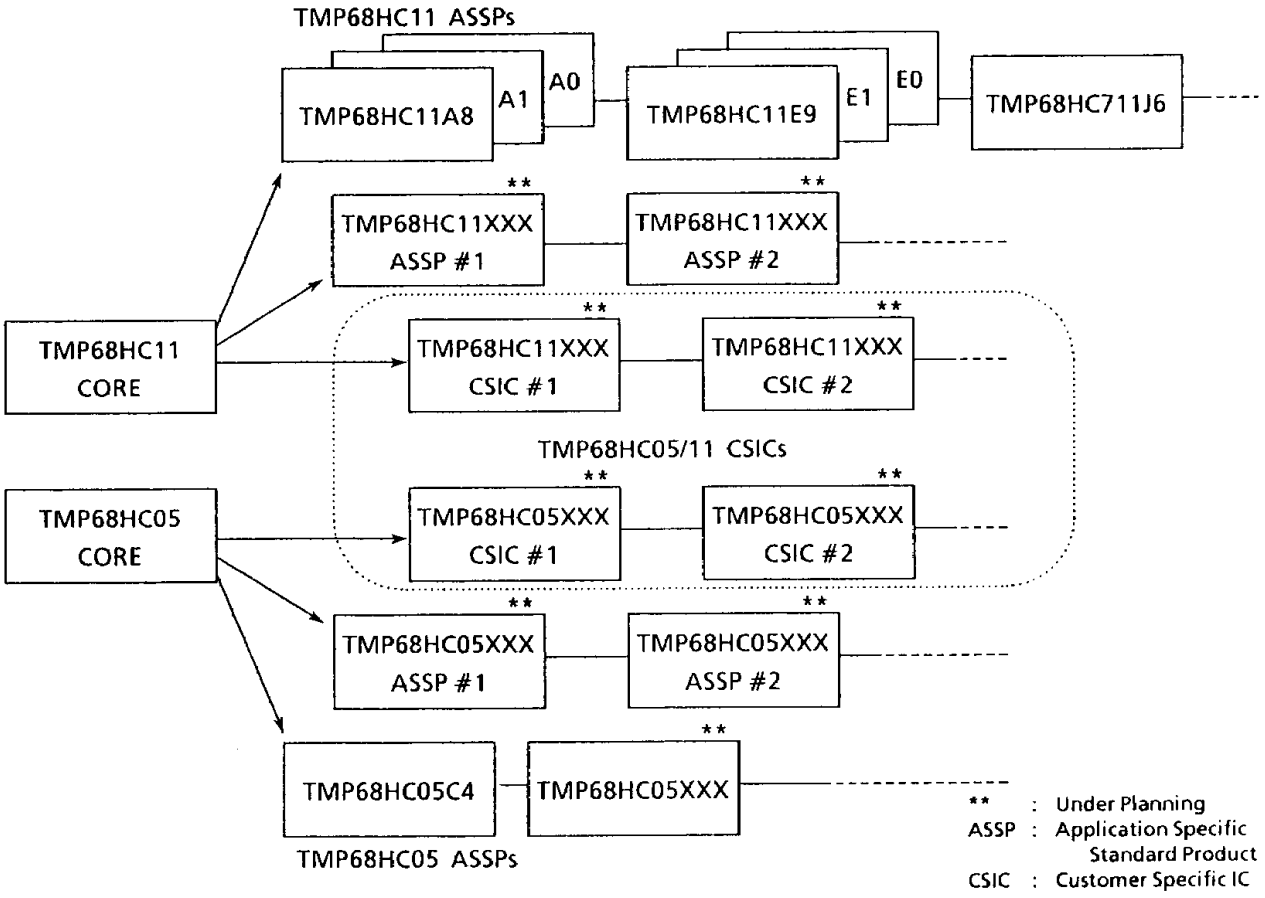
Toshiba offers various microcomputers for wide application in consumer appliances and industrial equipment. In this manual, we introduce you the 8-bit microcontroller family TMP68HC05/11 with its functions and characteristics. The TMP68HC05/11 family is the Toshiba 8-bit single chip microcontroller family which is developed by technical cooperation with Motorola Inc., and is compatible with the Motorola M68HC05/11 family.

The TMP68HC05 is a low-cost single chip microcontroller with sophisticated on-chip two way serial interface functions for a slave controller unit in various applications.

The TMP68HC11 is an advanced single chip microcontroller contains 512 bytes EEPROM (except TMP68HC11A0/E0) , 8-channel 8-bit A/D Converter, enhanced two way serial interface and enhanced timer system.

Toshiba plans to expand the TMP68HC05/11 family line-up and to develop the CSIC (Customer Specific IC) products based on the 68HC05/11 Core.

TMP68HC05/11 Family Line-up

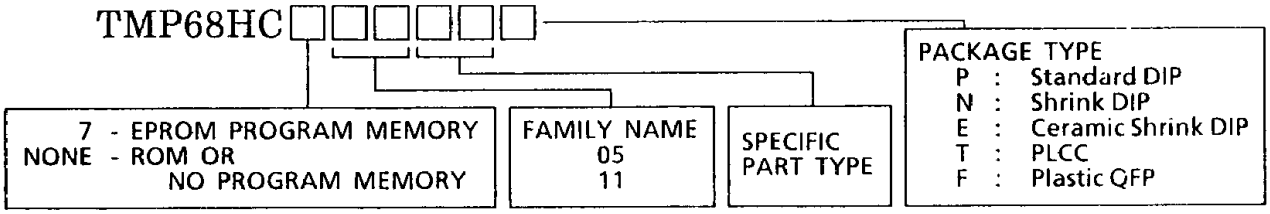


Memory Type and Package Type of the TMP68HC05/11 Family

Device NO.	ROM (Byte)	EEPROM (Byte)	RAM (Byte)	DIP		PLCC	Prastic QFP
				Plastic Standard	Plastic Shrink		
TMP68HC05C4	4160	—	176	40	—	44	44
TMP68HC11A8	8K	512	256	48	64	52	—
TMP68HC11A1	—	512	256	48	64	52	—
TMP68HC11A0	—	—	256	48	64	52	—
TMP68HC11E9	12K	512	512	—	64	52	—
TMP68HC11E1	—	512	512	—	64	52	—
TMP68HC11E0	—	—	512	—	64	52	—
TMP68HC711J6	16K (EPROM)	—	512	—	64, *64 with window	68	—

* : Ceramic DIP

Order Number



CONTENTS

1. INTRODUCTION

1.1 GENERAL

The TMP68HC05C4 CMOS Microcomputer is a member of the 68HC05 Family of low-cost single-chip microcomputers. This 8-bit microcomputer unit (MCU) contains an on-chip oscillator, CPU, RAM, ROM, I/O, two serial interface systems, and timer. The fully static design allows operation at frequencies down to dc, further reducing its already low-power consumption.

1.2 FEATURES

The following are some of the hardware and software highlights of the TMP68HC05C4.

HARDWARE FEATURES

- CMOS Technology
- 8-Bit Architecture
- Power Saving Stop, Wait, and Data Retention Modes
- Fully Static Operation
- 176 Bytes of On-Chip RAM
- 4160 Bytes of On-Chip ROM
- 24 Bidirectional I/O Lines
- 2.1 MHz Internal Operating Frequency at 5 Volts;1.0 MHz at 3 Volts
- Internal 16-Bit Timer Similar to MC6801 Timer
- Serial Communications Interface System
- Serial Peripheral Interface System
- Self-Check Mode
- External, Timer, Serial Communications Interface, and Serial Peripheral Interface Interrupts
- Master Reset and Power-On Reset
- Single 3- to 6-Volt Supply (2 Volt Data Retention Mode)
- On-Chip Oscillator with RC or Crystal/Ceramic Resonator Mask Options
- 40-Pin Dual-In-Line Package
- 44-Lead PLCC (Plastic Leaded Chip Carrier) Chip Carrier and 44-Pin QFP (Quad Flat Package) Also Available

SOFTWARE FEATURES

- Similar to MC6800
- 8×8 Unsigned Multiply Instruction
- Efficient Use of Program Space
- Versatile Interrupt Handling
- True Bit Manipulation
- Addressing Modes with Indexed Addressing for Tables
- Efficient Instruction Set
- Memory Mapped I/O
- Two Power-Saving Standby Modes
- Upward Software Compatible with the M146805 CMOS Family
- Complete Development System Support on EXOR ciser and HDS-200

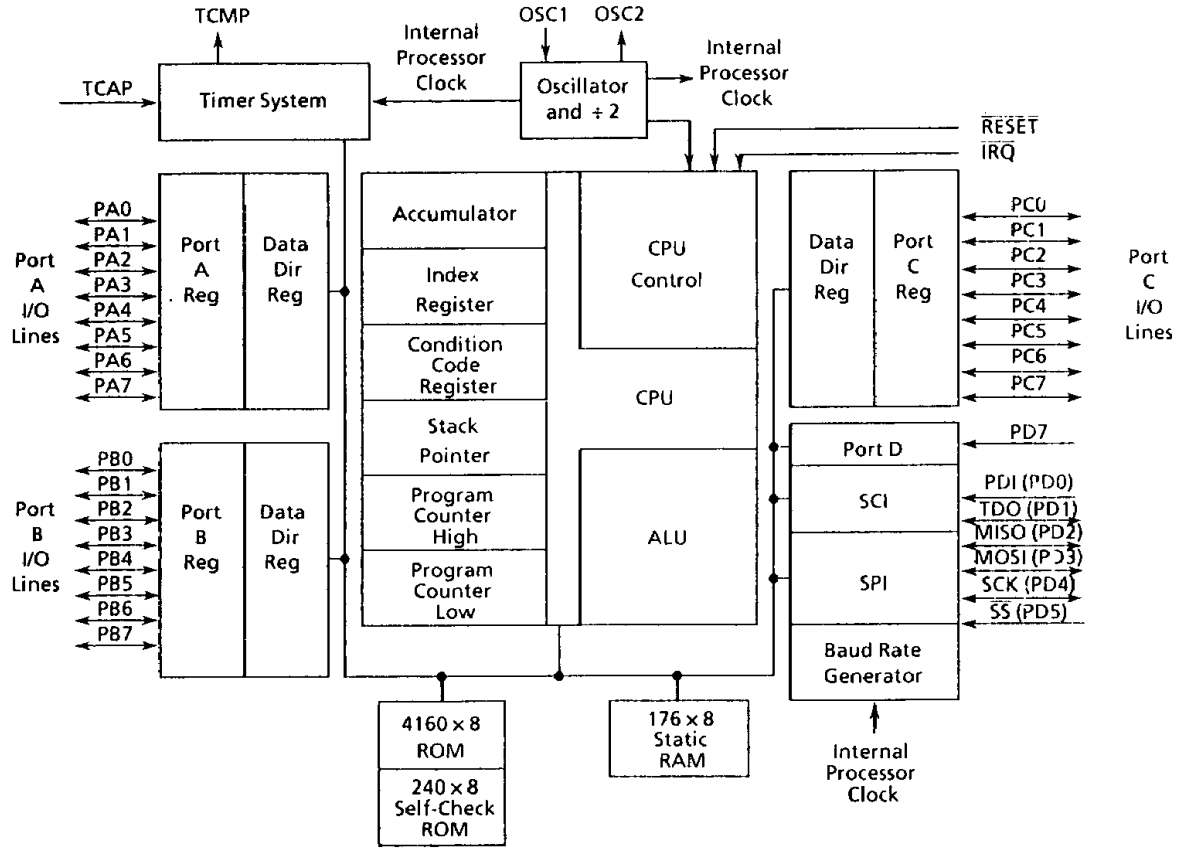


Figure 1.1 TMP68HC05C4 Microcomputer Block Diagram

2. FUNCTIONAL PIN DESCRIPTION, INPUT/OUTPUT PROGRAMMING, MEMORY, CPU REGISTERS, AND SELF-CHECK

This section provides a description of the functional pins, input/output programming, memory, CPU registers, and self-check.

2.1 FUNCTIONAL PIN DESCRIPTION

2.1.1 V_{DD} and V_{SS}

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

2.1.2 $\overline{IRQ}$ (Maskable Interrupt Request)

$\overline{IRQ}$ is a mask programmable option which provides two different choices of interrupt triggering sensitivity. These options are: 1) negative edge-sensitive triggering only, or 2) both negative edge-sensitive and level-sensitive triggering. In the latter case, either type of input to the $\overline{IRQ}$ pin will produce the interrupt. The MCU completes the current instruction before it responds to the interrupt request. When the $\overline{IRQ}$ pin goes low for at least one t_{ILIH}, a logic one is latched internally to signify an interrupt has been requested. When the MCU completes its current instruction, the interrupt latch is tested. If the interrupt latch contains a logic one, and the interrupt mask bit (1 bit) in the condition code register is clear, the MCU then begins the interrupt sequence.

If the option is selected to include level-sensitive triggering, then the $\overline{IRQ}$ input requires an external resistor to V_{DD} for "wire-OR" operation. See INTERRUPTS in Section 3 for more detail concerning interrupts.

2.1.3 $\overline{RESET}$

The $\overline{RESET}$ input is not required for startup but can be used to reset the MCU internal state and provide an orderly software startup procedure. Refer to RESETS in Section 3 for a detailed description.

2.1.4 TCAP

The TCAP input controls the input capture feature for the on-chip programmable timer system. Refer to INPUT CAPTURE REGISTER in Section 4 for additional information.

2.1.5 TCMP

The TCMP pin provides an output for the output compare feature of the on-chip timer system. Refer to OUTPUT COMPARE REGISTER in Section 4 for additional information.

2.1.6 OSC1, OSC2

The TMP68HC05C4 can be configured by mask option to accept either a crystal/ceramic resonator input or an RC network to control the internal oscillator. The internal clocks are derived by a divide-by-two of the internal oscillator frequency (f_{osc}).

2.1.6.1 CRYSTAL.

The circuit shown in Figure 2.1(b) is recommended when using a crystal. The internal oscillator is designed to interface with an AT-cut parallel resonant quartz crystal resonator in the frequency range specified for f_{osc} (refer to paragraph 9.7 or 9.8 Control Timing). Use of an external CMOS oscillator is recommended when crystals outside the specified ranges are to be used. The crystal and components should be mounted as close as possible to the input pins to minimize output distortion and startup stabilization time. Refer to paragraph 9.5 or 9.6 for V_{DD} specifications.

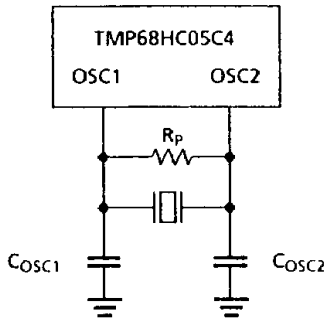
Crystal

	2 MHz	4 MHz	Units
R_{SMAX}	400	75	Ω
C_0	5	7	pF
C_1	0.008	0.012	μF
C_{OSC1}	15-40	15-30	pF
C_{OSC2}	15-30	15-25	pF
R_P	10	10	MΩ
Q	30	40	K

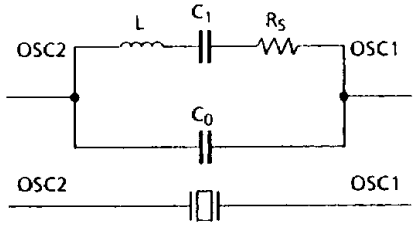
Ceramic Resonator

	2-4 MHz	Units
R_S (typical)	10	Ω
C_0	40	pF
C_1	4.3	pF
C_{OSC1}	30	pF
C_{OSC2}	30	pF
R_P	1-10	MΩ
Q	1250	–

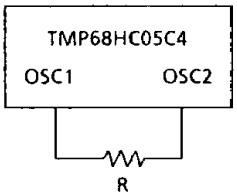
(a) Crystal/Ceramic Resonator Parameters



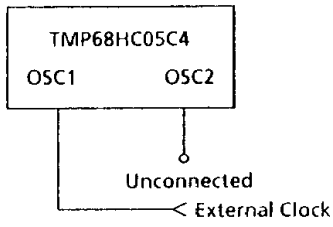
(b) Crystal/Ceramic Resonator Oscillator Connections



(c) Equivalent Crystal Circuit



(d) RC Oscillator Connections



(e) External Clock Source Connections
(Either Crystal or RC Mask Options)

Figuer 2.1 Oscillator Connections

2.1.6.2 CERAMIC RESONATOR.

A ceramic resonator may be used in place of the crystal in cost-sensitive applications. The circuit in Figure 2.1(b) is recommended when using a ceramic resonator. Figure 2.1(a) lists the recommended capacitance and feedback resistance values. The manufacturer of the particular ceramic resonator being considered should be consulted for specific information.

2.1.6.3 RC.

If the RC oscillator option is selected, then a resistor is connected to the oscillator pins as shown in Figure 2.1(d). The relationship between R and f_{osc} is shown in Figure 2.2.

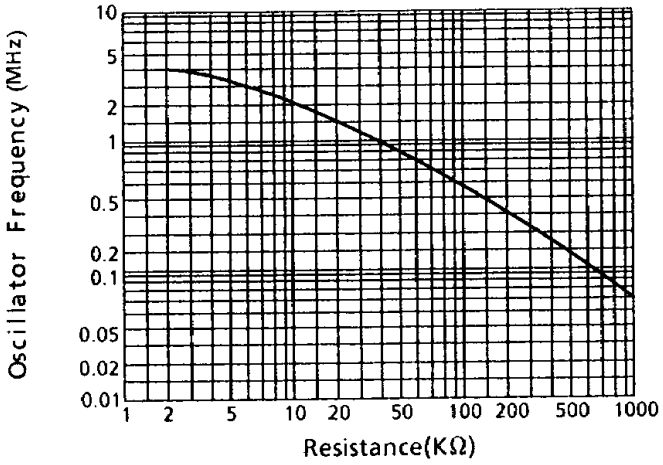


Figure 2.2 Typical Frequency vs Resistance for RC Oscillator Option Only(V_{DD}=5.0V)

2.1.6.4 EXTERNAL CLOCK.

An external clock should be applied to the OSC1 input with the OSC2 pin not connected, as shown in Figure 2.1(e). An external clock may be used with either the RC or crystal oscillator option. The t_{OXOV} or t_{ILCH} specifications do not apply when using an external clock input. The equivalent specification of the external clock source should be used in lieu of t_{OXOV} or t_{ILCH}.

2.1.7 PA0-PA7

These eight I/O lines comprise port A. The state of any pin is software programmable and all port A lines are configured as input during power-on or reset. Refer to INPUT/OUTPUT PROGRAMMING paragraph for a detailed description of I/O programming.

2.1.8 PB0-PB7

These eight lines comprise port B. The state of any pin is software programmable and all port B lines are configured as input during power-on or reset. Refer to INPUT/OUTPUT PROGRAMMING paragraph for a detailed description of I/O programming.

2.1.9 PC0-PC7

These eight lines comprise port C. The state of any pin is software programmable and all port C lines are configured as input during power-on or reset. Refer to INPUT/OUTPUT PROGRAMMING paragraph for a detailed description of I/O programming.

2.1.10 PD0-PD5, PD7

These seven lines comprise port D, a fixed input port that is enabled during power-on. All enabled special functions (SPI and SCI) affect the pins on this port. Four of these lines, PD2/MISO, PD3/MOSI, PD4/SCK, and PD5/$\overline{SS}$, are used in the serial peripheral interface (SPI) discussed in Section 6. Two of these lines, PD0/PDI and PD1/TDO, are used in the serial communications interface (SCI) discussed in Section 5. Refer to 2.2 INPUT/OUTPUT PROGRAMMING for a detailed description of I/O programming.

2.2 INPUT/OUTPUT PROGRAMMING

2.2.1 Parallel Ports

Ports A, B, and C may be programmed as an input or an output under software control. The direction of the pins is determined by the state of the corresponding bit in the port data direction register (DDR). Each 8-bit port has an associated 8-bit data direction register. Any port A, port B, or port C pin is configured as an output if its corresponding DDR bit is set to a logic one. A pin is configured as an input if its corresponding DDR bit is cleared to a logic zero. At power-on or reset, all DDRs are cleared, which configure all port A, B, and C pins as inputs. The data direction registers are capable of being written to or read by the processor. Refer to Figure 2.3 and Table 2.1. During the programmed output state, a read of the data register actually reads the value of the output data latch and not the I/O pin.

Table 2.1 I/O Pin Functions

R/$\overline{W}$*	DDR	I/O Pin Function
0	0	The I/O pin is in input mode. Data is written into the output data latch.
0	1	Data is written into the output data latch and output to the I/O pin.
1	0	The state of the I/O pin is read.
1	1	The I/O pin is in an output mode. The output data latch is read.

*R/$\overline{W}$ is an internal signal.

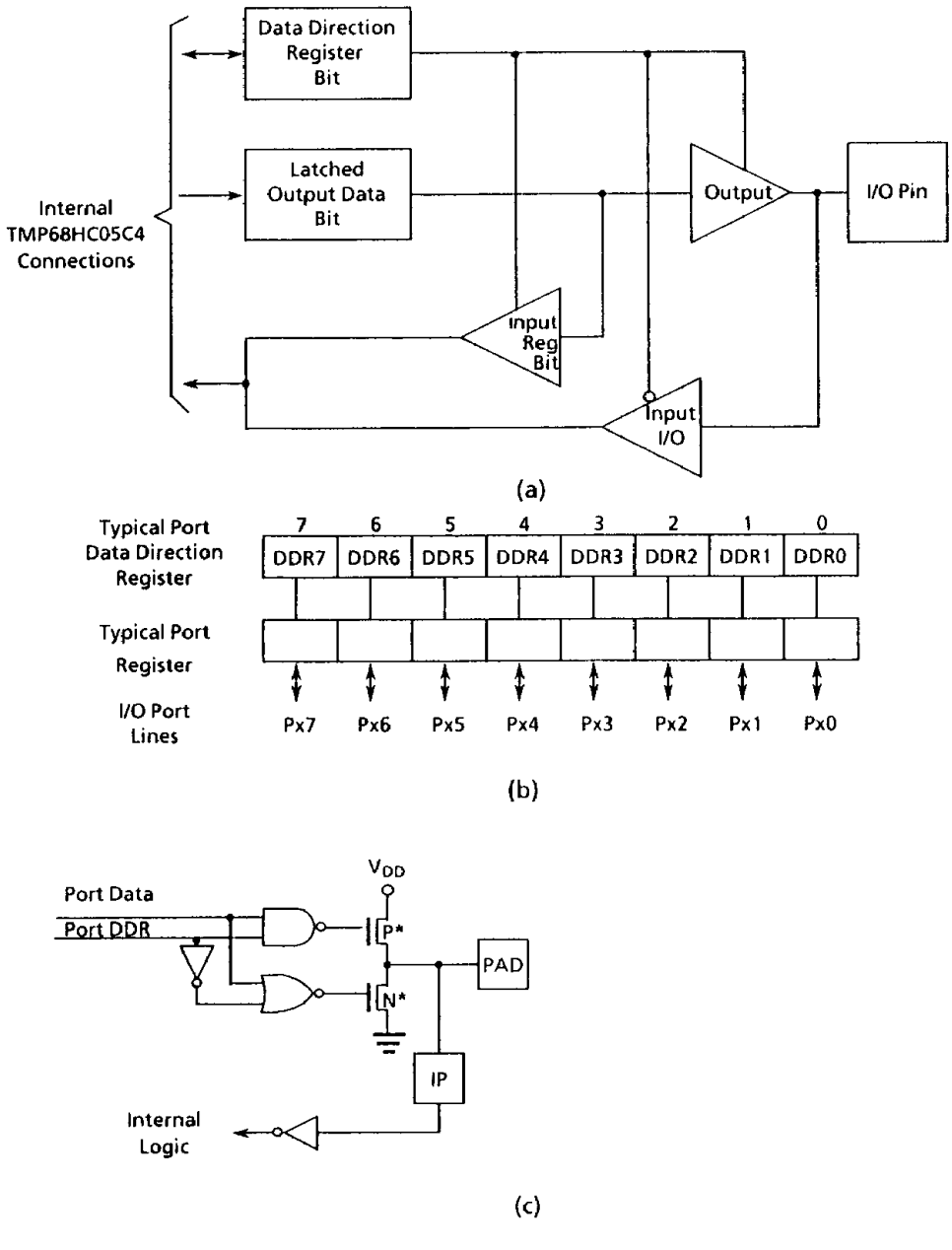
2.2.2 Fixed Port

Port D is a 7-bit fixed input port (PD0-PD5, PD7) that continually monitors the external pins whenever the SPI or SCI systems are disabled. During power-on reset or external reset all seven bits become valid input ports because all special function output drivers are disabled. For example, with the serial communications interface (SCI) system enabled, (RE=TE=1) PD0 and PD1 inputs will read zero. With the serial peripheral interface (SPI) system disabled (SPE=0) PD2 through PD5 will read the state of the pin at the time of the read operation. No data register is associated with the port when it is used as an input.

Note: It is recommended that all unused inputs and I/O ports be tied to an appropriate logic level (e.g., either V_{DD} or V_{SS}).

2.2.3 Serial Port (SCI and SPI)

The serial communications interface (SCI) and serial peripheral interface (SPI) use the port D pins for their functions. The SCI function requires two of the pins (PD0-PD1) for its receive data input (RDI) and transmit data output (TDO) respectively, whereas the SPI function requires four of the pins (PD2-PD5) for its serial data input/output (MISO), serial data output/input (MOSI), system clock (SCK), and slave select ($\overline{SS}$) respectively. Refer to SECTION 5 SERIAL COMMUNICATIONS INTERFACE and SECTION 6 SERIAL PERIPHERAL INTERFACE for a more detailed discussion.



Notes :

1. *Denotes devices have same physical size, and are enhancement type.
2. IP = Input protection.
3. Latch-up protection not shown.

Figure 2.3 Parallel Port I/O Circuitry

2.3 MEMORY

As shown in Figure 2.4, the MCU is capable of addressing 8192 bytes of memory and I/O registers with its program counter. The TMP68HC05C4 MCU has implemented 4601 bytes of these locations. The first 256 bytes of memory (page zero) include: 25 bytes of I/O features such as data ports, the port DDRs, timer, serial peripheral interface (SPI), and serial communication interface (SCI);48 bytes of user ROM, and 176 bytes of RAM. The next 4096 bytes complete the user ROM. The self-check ROM (224 bytes) and self-check vectors (16 bytes) are contained in memory locations $1F00 through $1FEF. The 16 highest address bytes contain the user defined reset and the interrupt vectors. Seven bytes of the lowest 32 memory locations are unused and the 176 bytes of user RAM include up to 64 bytes for the stack. Since most programs use only a small part of the allocated stack locations for interrupts and/or subroutine stacking purposes, the unused bytes are usable for program data storage.

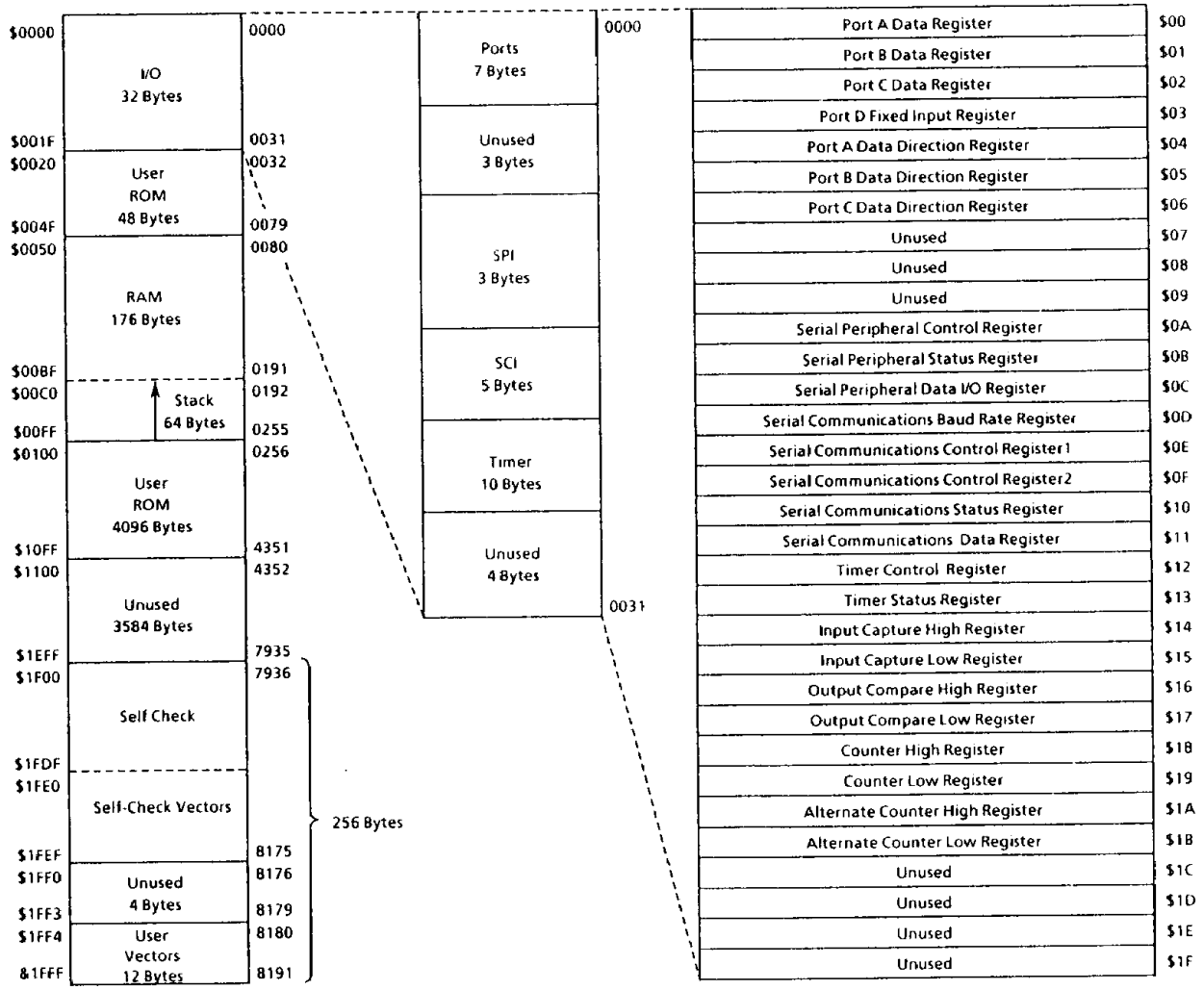


Figure 2.4 TMP68HC05C4 Memory Map

2.4 CPU REGISTERS

The TMP68HC05C4 CPU contains five registers, as shown in the programming model of Figure 2.5. The interrupt stacking order is shown in Figure 2.6.

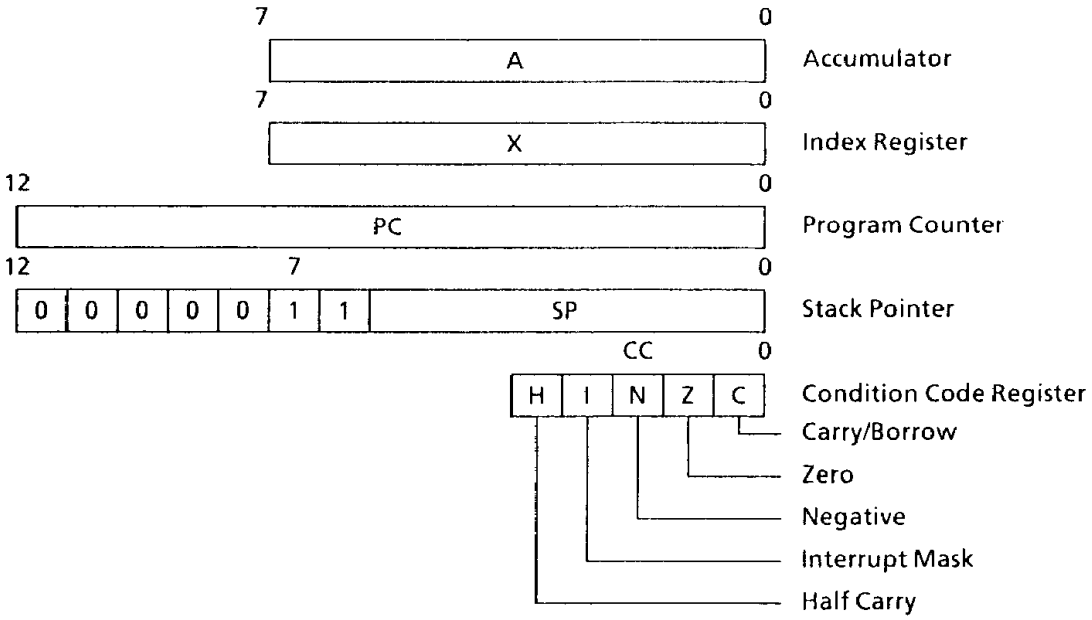
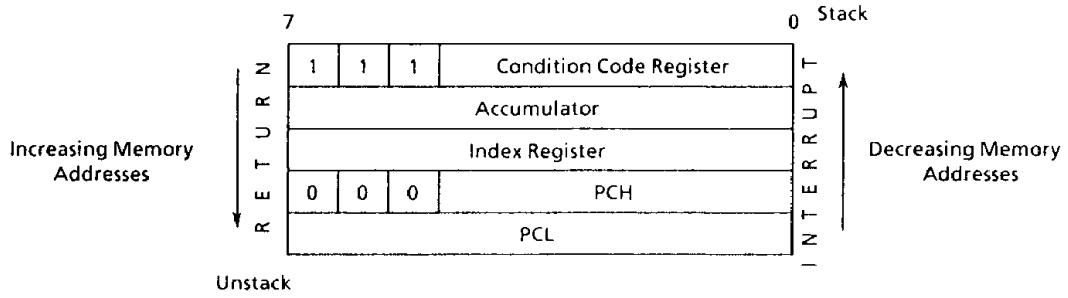


Figure 2.5 Programming Model



Note : Since the Stack Pointer decrements during pushes, the PCL is stacked first, followed by PCH, etc. Pulling from the stack is in the reverse order.

Figure 2.6 Stacking Order

2.4.1 Accumulator (A)

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

2.4.2 Index Register (X)

The X register is an 8-bit register which is used during the indexed modes of addressing. It provides an 8-bit value which is used to create an effective address. The

index register is also used for data manipulations with the read-modify-write type of instructions and as a temporary storage register when not performing addressing operations.

2.4.3 Program Counter (PC)

The program counter is a 13-bit register that contains the address of the next instruction to be executed by the processor.

2.4.4 Stack Pointer (SP)

The stack pointer is a 13-bit register containing the address of the next free locations on the push-down/pop-up stack. When accessing memory, the seven most significant bits are permanently configured to 0000011. These seven bits are appended to the six least significant register bits to produce an address within the range of $00FF to $00C0. The stack area of RAM is used to store the return address on subroutine calls and the machine state during interrupts. During external or power-on reset, and during a reset stack pointer (RSP) instruction, the stack pointer is set to its upper limit ($00FF). Nested interrupt and/or subroutines may use up to 64 (decimal) locations. When the 64 locations are exceeded, the stack pointer wraps around and points to its upper limit ($00FF), thus, losing the previously stored information. A subroutine call occupies two RAM bytes on the stack, while an interrupt uses five RAM bytes.

2.4.5 Condition Code Register (CC)

The condition code register is a 5-bit register which indicates the results of the instruction just executed as well as the state of the processor. These bits can be individually tested by a program and specified action taken as a result of their state. Each bit is explained in the following paragraphs.

2.4.5.1 HALF CARRY BIT (H).

The H bit is set to a one when a carry occurs between bits 3 and 4 of the ALU during an ADD or ADC instruction. The H bit is useful in binary coded decimal subroutines.

2.4.5.2 INTERRUPT MASK BIT (I).

When the I bit is set, all interrupts are disabled. Clearing this bit enables the interrupts. If an external interrupt occurs while the I bits is set, the interrupt is latched and is processed after the I bit is next cleared;therefore, no interrupts are lost because of the I bit being set. An internal interrupt can be lost if it is cleared while the I bit is set (refer to SECTION 4 PROGRAMMABLE TIMER, SECTION 5 SERIAL COMMUNICATIONS INTERFACE, and SECTION 6 SERIAL PERIPHERAL INTERFACE for more information).

2.4.5.3 NEGATIVE (N).

When set, this bit indicates that the result of the last arithmetic, logical, or data manipulation is negative (bit 7 in the result is a logic one).

2.4.5.4 ZERO (Z).

When set, this bit indicates that the result of the last arithmetic, logical, or data manipulation is zero.

2.4.5.5 CARRY/BORROW (C).

Indicates that a carry or borrow out of the arithmetic logic unit (ALU) occurred during the last arithmetic operation. This bit is also affected during bit test and branch instructions, shifts, and rotates.

2.5 SELF-CHECK

The self-check capability of the TMP68HC05C4 MCU provides an internal check to determine if the device is functional. Self-check is performed using the circuit shown in the schematic diagram of Figure 2.7. As shown in the diagram, port C pins PC0-PC3 are monitored (light emitting diodes are shown but other devices could be used) for the self-check results. The self-check mode is entered by applying a 9 V dc input (through a 4.7 KΩ resistor) to the $\overline{IRQ}$ pin and 5 V dc input (through a 4.7 KΩ resistor) to the TCAP pin and then depressing the reset switch to execute a reset. After reset, the following seven tests are performed automatically:

I/O – Functionally exercises ports A, B, and C

RAM – Counter test for each RAM byte

Timer – Tracks counter register and checks OCF flag

SCI – Transmission Test;checks for RDRF, TDRE, TC, and FE flags

ROM – Exclusive OR with odd ones parity result

SPI – Transmission test with check for SPIF, WCOL, and MODF flags

INTERRUPTS – Tests external, timer, SCI, and SPI interrupts.

Self-check results (using the LEDs as monitors) are shown in Table 2.2. The following subroutines are available to user programs and do not require any external hardware.

2.5.1 Timer Test Subroutine

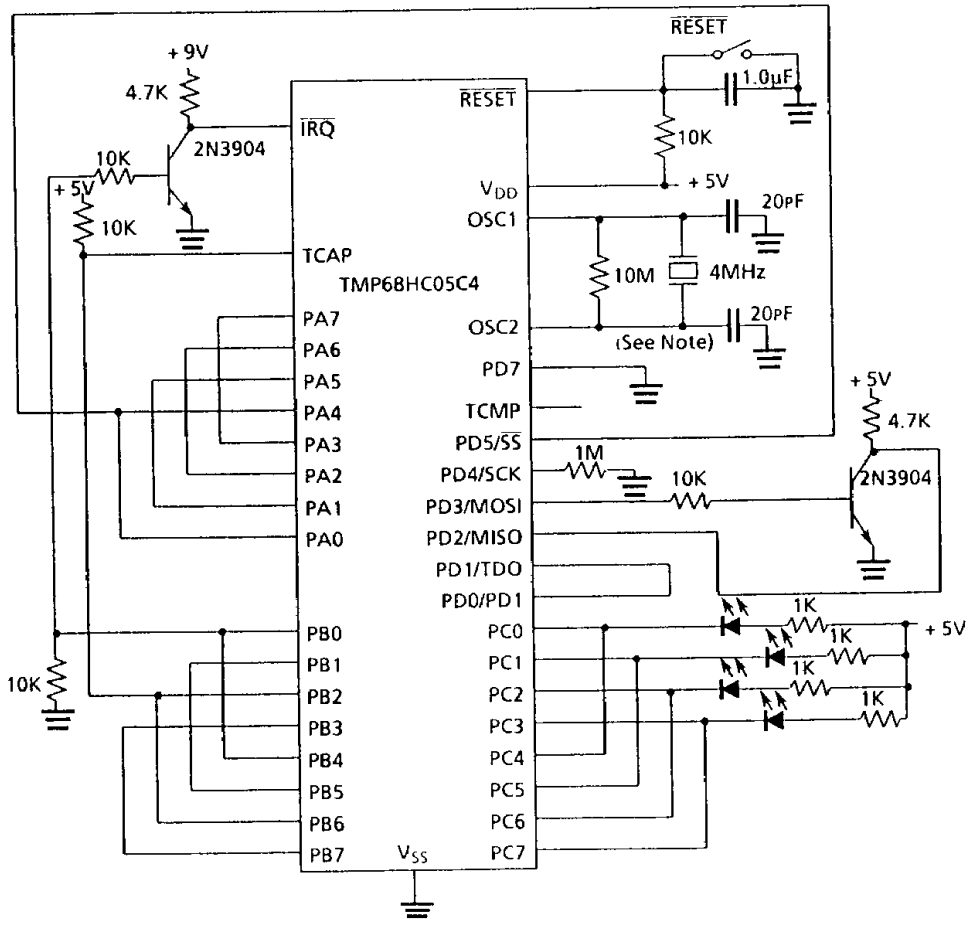
This subroutine returns with the Z bit cleared if any error is detected;otherwise, the Z bit is set.

This subroutine is called at location $1F0E. The output compare register is first set to the current timer state. Because the timer is free running and has only a divide-by-four prescaler, each timer count cannot be tested. The test reads the timer once every 10 counts (40 cycles) and checks for correct counting. The test tracks the counter until the timer wraps around, triggering the output compare flag in the timer status register. RAM locations $0050 and $0051 are overwritten. Upon return to the user's program, X=40. If the test passed, A=0.

2.5.2 ROM Checksum Subroutine

This subroutine returns with the Z bit cleared if any error is detected;otherwise, the Z bit is set.

This subroutine is called at location $1F93 with RAM location $0053 equal to $01 and A=0. A short routine is set up and executed in RAM to compute a checksum of the entire ROM pattern. Upon return to the user's program, X=0. If the test passed, A=0. RAM locations $0050 through $0053 are overwritten.



Note : The RC Oscillator Option may also be used in this circuit.

Figuer 2.7 Self-Check Circuit Schematic Diagram

Table 2.2 Self-Check Results

PC3	PC2	PC1	PC0	Remarks
1	0	0	1	Bad I/O
1	0	1	0	Bad RAM
1	0	1	1	Bad Timer
1	1	0	0	Bad SCI
1	1	0	1	Bad ROM
1	1	1	0	Bad SPI
1	1	1	1	Bad Interrupts or $\overline{IRQ}$ Request
Flashing				Good Device
Ail Others				Bad Device, Bad Port C, etc.

0 Indicates LED on;1 Indicates LED is off.

3. RESETS, INTERRUPTS, LOW POWER, AND DATA RETENTION MODES

3.1 RESETS

The TMP68HC05C4 has two reset modes:an active low external reset pin ($\overline{RESET}$) and a power-on reset function;refer to Figure 3.1.

3.1.1 $\overline{RESET}$ Pin

The $\overline{RESET}$ input pin is used to reset the MCU to provide an orderly software startup procedure. When using the external reset mode, the $\overline{RESET}$ pin must stay low for a minimum of one and one half t_{cyc}. The $\overline{RESET}$ pin contains an internal Schmitt Trigger as part of its input to improve noise immunity.

3.1.2 Power-On Reset

The power-on reset occurs when a positive transition is detected on V_{DD}. The power-on reset is used strictly for power turn-on conditions and should not be used to detect any drops in the power supply voltage. There is no provision for a power-down reset. The power-on circuity provides for a 4064 t_{cyc} delay from the time that the oscillator becomes active. If the external $\overline{RESET}$ pin is low at the end of the 4064 t_{cyc} time out, the processor remains in the reset condition unit $\overline{RESET}$ goes high. The user must ensure that V_{DD} has risen to a point where the MCU can operate properly prior to the time the 4064 POR reset cycles have elapsed. If there is doubt, the external $\overline{RESET}$ pin should remain low until such time that V_{DD} has risen to the minimum operating voltage specified.

Table 3.1 shows the actions of the two resets on internal circuits, but not necessarily in order of occurrence (X indicates that the condition occurs for the particular reset).

3.2 INTERRUPTS

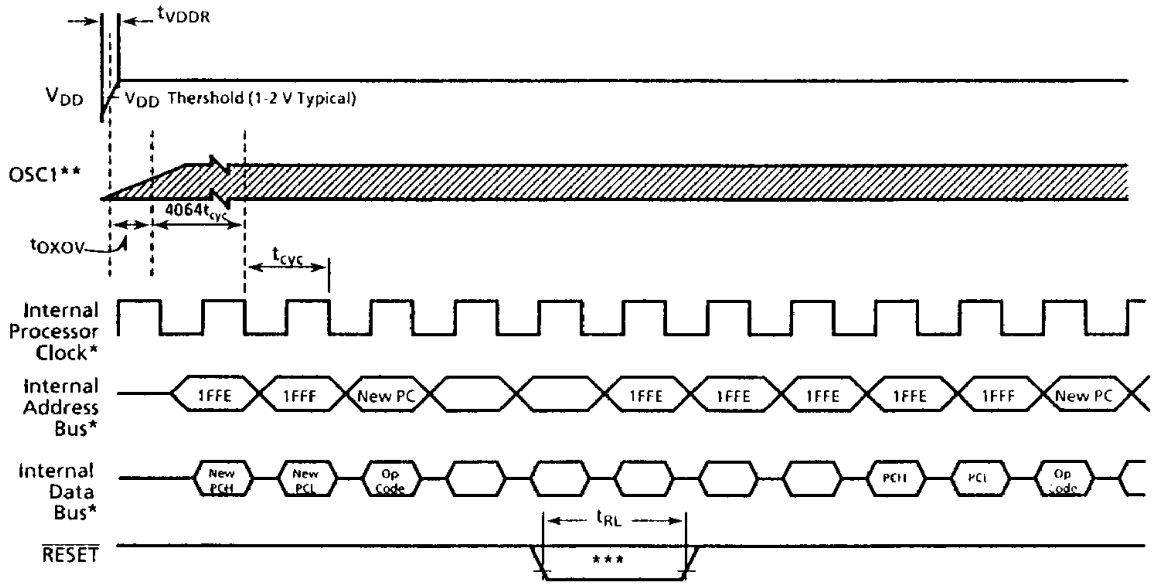
Systems often require that normal processing be interrupted so that some external event may be serviced. The TMP68HC05C4 may be interrupted by one of five different methods:either one of four maskable hardware interrupts I($\overline{IRQ}$, SPI, SCI, or Timer) and one non-maskable software interrupt (SWI). Interrupts such as Timer, SPI, and SCI have several flags which will cause the interrupt. Generally, interrupt flags are located in read-only status registers, whereas their equivalent enable bits are located in associated control registers. The interrupt flags and enable bits are never contained in the same register. If the enable bit is a logic zero it blocks the interrupt from occurring but does not inhibit the flag from being set. Reset clears all enable bits to preclude interrupts during the reset procedure.

The general sequence for clearing an interrupt is a software sequence of first accessing the status register while the interrupt flag is set, followed by a read or write of an associated register. When any of these interrupts occur, and if the enable bit is a

logic one, normal processing is suspended at the end of the current instruction execution. Interrupts cause the processor registers to be saved on the stack (see Figure 2.6) and the interrupt mask (I bit) set to prevent additional interrupts. The appropriate interrupt vector then points to the starting address of the interrupt service routine (refer to Figure 2.4 for vector location). Upon completion of the interrupt service routine, the RTI instruction (which is normally a part of the service routine) causes the register contents to be recovered from the stack followed by a return to normal processing. The stack order is shown in Figure 2.6.

Note : The interrupt mask bit (I bit) will be cleared if and only if the corresponding bit stored in the stack is zero.

A discussion of interrupts, plus a table listing vector addresses for all interrupts including reset, in the TMP68HC05C4 is provided in Table 3.2.



* Internal timing signal and bus information not available externally.
** OSC1 line is not meant to represent frequency. It is only used to represent time.
*** The next rising edge of the internal processor clock following the rising edge of $\overline{\text{RESET}}$ initiates the reset sequence.

Figure 3.1 Power-On Reset and $\overline{\text{RESET}}$

Table 3.1 Reset Action on Internal Circuit

Condition	$\overline{\text{RESET}}$ Pin	Power-On Reset
Timer Prescaler reset to zero state	×	×
Timer counter configured to $FFFC	×	×
Timer output compare (TCMP) bit reset to zero	×	×
All timer interrupt enable bits cleared (ICIE, OCIE, and TOIE) to disable timer interrupts. The OLVL timer bit is also cleared by reset.	×	×
All data direction registers cleared to zero (input)	×	×
Configure stack pointer to $00FF	×	×
Force internal address bus to restart vector ($1FFE-$1FFF)	×	×
Set bit in condition code register to a logic one	×	×
Clear STOP latch	×*	×
Clear external interrupt latch	×	×
Clear WAIT latch	×	×
Disable SCI (serial control bits TE = 0 and RE = 0). Other SCI bits cleared by reset include: TIE, TCIE, RIE, ILIE, RWU, SBK, RDRF, IDLE, OR, NF, and FE.	×	×
Disable SPI (serial output enable condtrol bit SPE = 0). Other SPI bits cleared by reset include: SPIE, MSTR, SPIF, WCOL, and MODF.	×	×
Set serial status bits TDRE and TC	×	×
Clear all serial interrupt enable bits (SPIE, TIE, and TCIE)	×	×
Place SPI system in slave mode (MSTR = 0)	×	×
Clear SCI prescaler rate control bits SCP0-SCP1	×	×

*Indicates that timeout still occurs.

Table 3.2 Vector Address for Interrupts and Reset

Register	Flag Name	Interrupts	CPU Interrupt	Vector Address
N/A	N/A	Reset	$\overline{\text{RESET}}$	$1FFE-$1FFF
N/A	N/A	Software	SWI	$1FFC-$1FFD
N/A	N/A	Exdternal Interrupt	$\overline{\text{IRQ}}$	$1FFA-$1FFB
Timer Status	ICF	Input Capture	TIMER	$1FF8-$1FF9
	OCF	Output Compare		
	TOF	Timer Overflow		
SCI Status	TDRE	Transmit Buffer Empty	SCI	$1FF6-$1FF7
	TC	Transmit Complete		
	RDRF	Receiver Buffer Full		
	IDLE	Idle Line Detect		
	OR	Overrun		
SPI Status	SPIF	Transfer Complete	SPI	$1FF4-$1FF5
	MODF	Mode Fault		

3.2.1 Hardware Controlled Interrupt Sequence

The following three functions ($\overline{\text{RESET}}$, STOP, and WAIT) are not in the strictest sense an interrupt;however, they are acted upon in a similar manner. Flowcharts for hardware interrupts are shown in Figure 3.2, and for STOP and WAIT are provided in Figure 3.3. A discussion is provided below.

(a) —A low input on the $\overline{\text{RESET}}$ input pin causes the program to vector to its starting address which is specified by the contents of memory locations $1FFE and $1FFF. The I bit in the condition code register is also set. Much of the MCU is configured to a known state during this type of reset as previously described in RESETS paragraph 3.1.

(b) STOP—The STOP instruction causes the oscillator to be turned off and the processor to "sleep" until an external interrupt ($\overline{\text{IRQ}}$) or reset occurs.

(c) WAIT—The WAIT instruction causes all processor clocks to stop, but leaves the Timer, SCI, and SPI clocks running. This "rest" state of the processor can be cleared by reset, an external interrupt ($\overline{\text{IRQ}}$), Timer interrupt, SPI interrupt, or SCI interrupt. There are no special wait vectors for these individual interrupts.

3.2.2 Software Interrupt (SWI)

The software interrupt is an executable instruction. The action of the SWI instruction is similar to the hardware interrupts. The SWI is executed regardless of the state of the interrupt mask (I bit) in the condition code register. The interrupt service routine address is specified by the contents of memory location $1FFC and $1FFD.

3.2.3 External Interrupt

If the interrupt mask (I bit) of the condition code register has been cleared and the external interrupt pin ($\overline{\text{IRQ}}$) has gone low, then the external interrupt is recognized. When the interrupt is recognized, the current state of the CPU is pushed onto the stack and the I bit is set. This masks further interrupts until the present one is serviced. The interrupt service routine address is specified by the contents of memory location $1FFA and $1FFB. Either a level-sensitive and negative edge-sensitive trigger, or a negative edge-sensitive only trigger are available as a mask option. Figure 3.4 shows both a functional and mode timing diagram for the interrupt line. The timing diagram shows two different treatments of the interrupt line ($\overline{\text{IRQ}}$) to the processor. The first method shows single pulses on the interrupt line spaced far enough apart to be serviced. The minimum time between pulses is a function of the number of cycles required to execute the interrupt service routine plus 21 cycles. Once a pulse occurs, the next pulse should not occur until the MCU software has exited the routine (an RTI occurs). The second configuration shows several interrupt lines "wire-ORed" to form the interrupts at the

processor. Thus, if after servicing one interrupt the interrupt line remains low, then the next interrupt is recognized.

Note : The internal interrupt latch is cleared in the first part of the service routine;therefore, one (and only one) external interrupt pulse could be latched during t_{ILIL} and serviced as soon as the I bit is cleared.

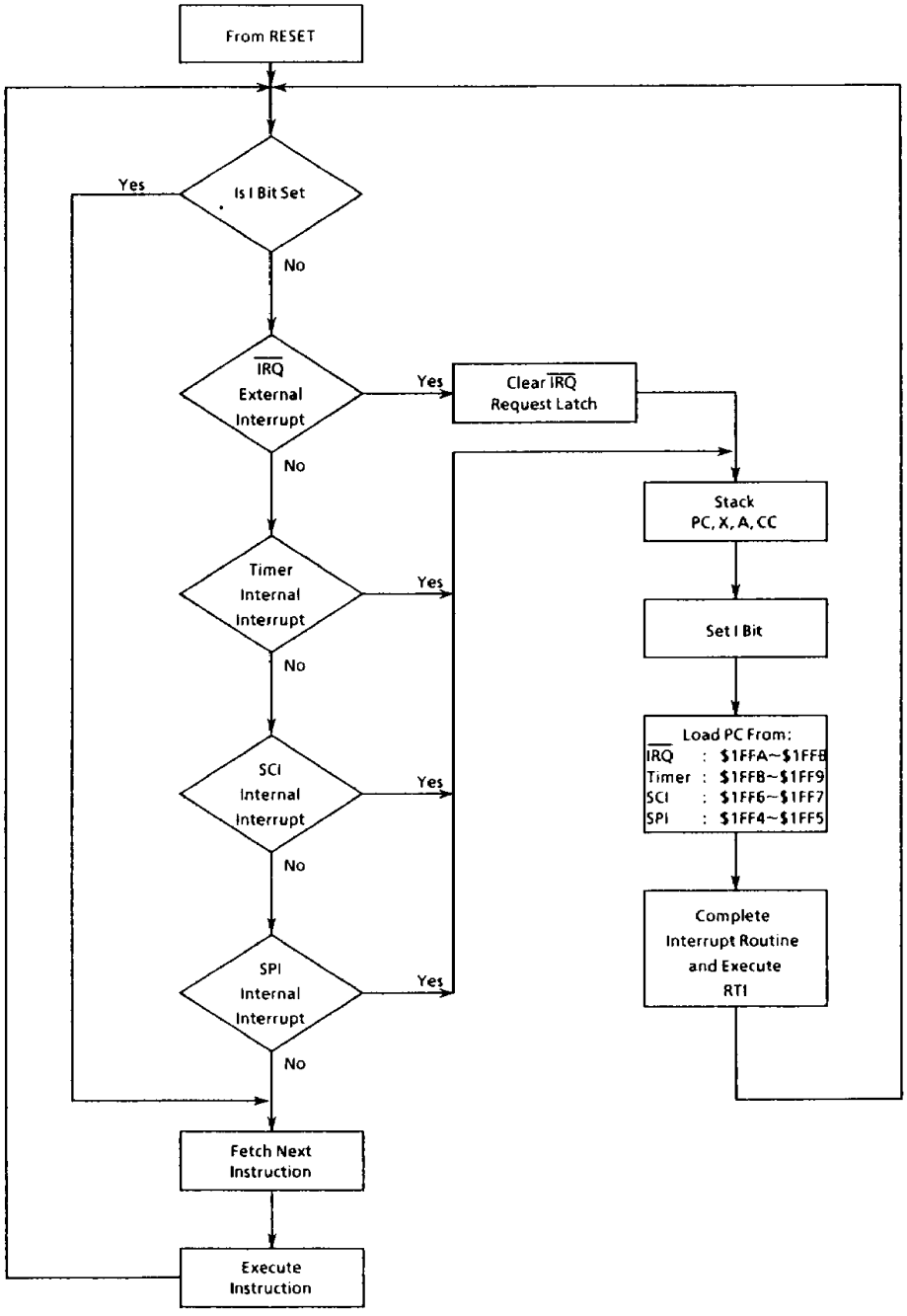


Figure 3.2 Hardware Interrupt Flowchart

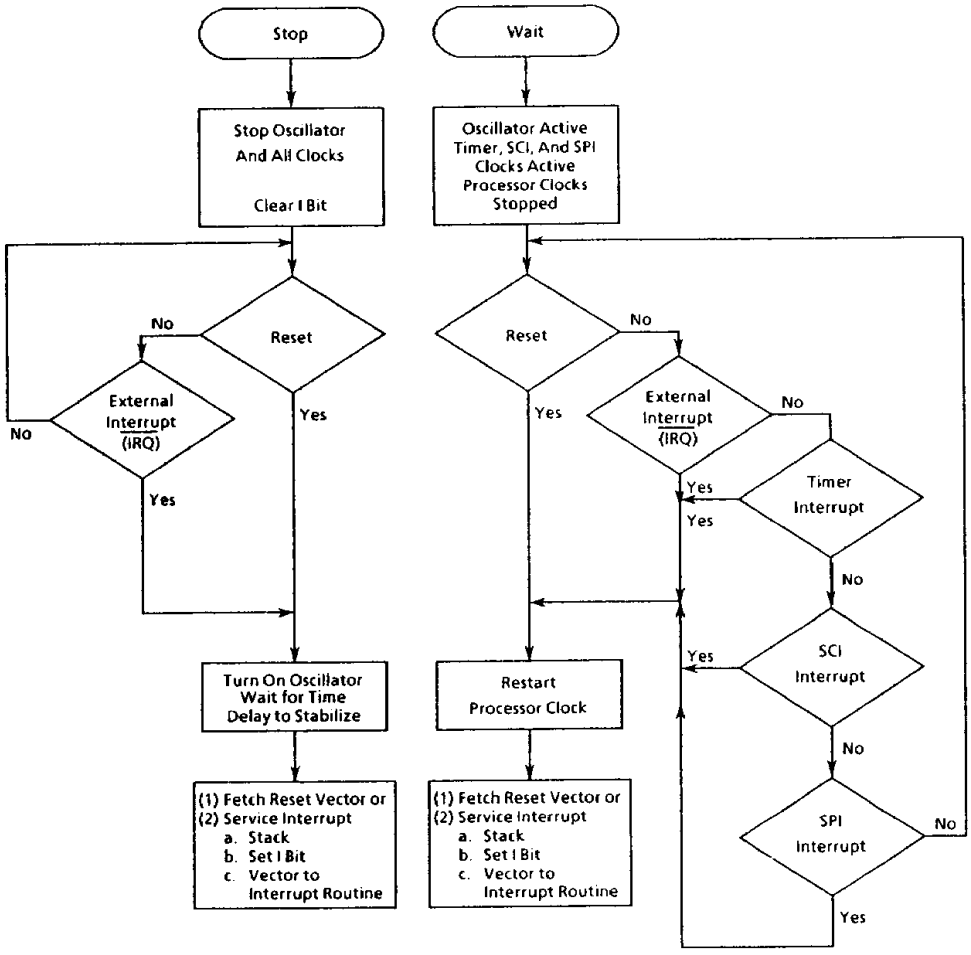
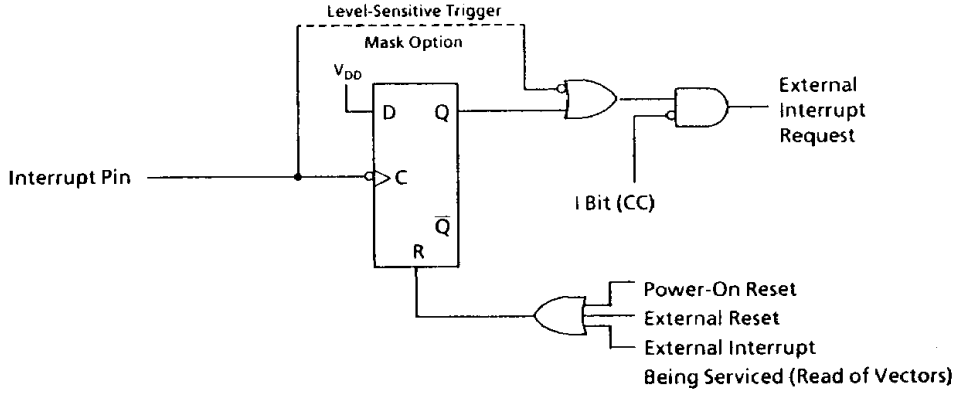
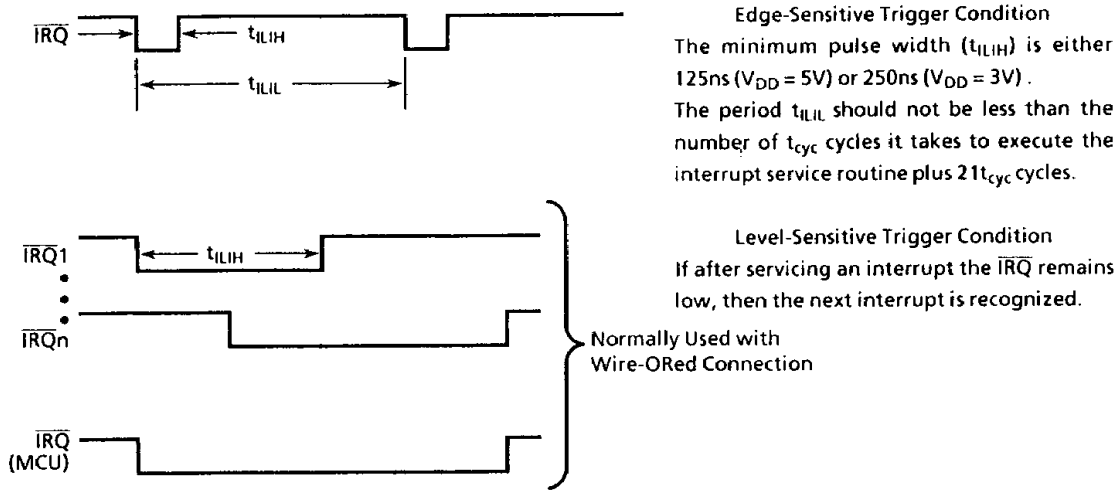


Figure 3.3 STOP/WAIT Flowcharts



(a)Interrupt Function Diagram



(b)Interrupt Mode Diagram

Figure 3.4 External Interrupt

3.2.4 Timer Interrupt

There are three different timer interrupt flags that will cause a timer interrupt whenever they are set and enabled. These three interrupt flags are found in the three most significant bits of the timer status register (TSR, location $13) and all three will vector to the same interrupt service routine ($1FF8-$1FF9).

All interrupt flags have corresponding enable bits (ICIE, OCIE, and TOIE)in the timer control register (TCR, location $12). Reset clears all enable bits, thus preventing an interrupt from occurring during the reset time period. The actual processor interrupt is generated only if the I bit in the condition code register is also cleared. When the interrupt is recognized, the current machine state is pushed onto the stack and I bit is set. This masks further interrupts until the present one is serviced. The interrupt service routine address is specified by the contens of memory location $1FF8 and $1FF9.

The general sequence for clearing an interrupt is a software sequence of accessing the status register while the flag is set, followed by a read or write of an associated register. Refer to SECTION 4 PROGRAMMABLE TIMER for additional information about the timer circuitry.

3.2.5 Serial Communications Interface (SCI) Interrupts

An interrupt in the serial communications interface (SCI) occurs when one of the interrupt flag bits in the serial communications status register is set, provided the I bit in the condition code register is clear and the enable bit in the serial communications control register 2 (location $0F) is enabled. When the interrupt is recognized, the current state of the machine is pushed onto the stack and the I bit in the condition code register is set. This masks further interrupts until the present one is serviced. The SCI interrupt causes the program counter to vector to memory location $1FF6 and $1FF7 which contains the starting address of the interrupt service routine. Software in the serial interrupt service routine must determine the priority and cause of the SCI interrupt by examining the interrupt flags and the status bits located in the serial communications status register (location $10). The general sequence for clearing an interrupt is a software sequence of accessing the serial communications status register while the flag is set followed by a read or write of an associated register. Refer to SECTION 5 SERIAL COMMUNICATIONS INTERFACE for a description of the SCI system and its interrupts.

3.2.6 Serial Peripheral Interface (SPI) Interrupts

An interrupt in the serial peripheral interface (SPI) occurs when one of the interrupt flag bits in the serial peripheral status register (location $0B) is set, provided the I bit in the condition code register is clear and the enable bit in the serial peripheral control register (location $0A) is enabled. When the interrupt is recognized, the current state of the machine is pushed onto the stack and the I bit in the condition code register is set. This masks further interrupts until the present one is serviced. The SPI interrupt causes the program counter to vector to memory location $1FF4 and $1FF5 which contains the starting address of the interrupt service routine. Software in the serial peripheral interrupt service routine must determine the priority and cause of the SPI interrupt by examining the interrupt flag bits located in the SPI status register. The general sequence for clearing an interrupt is a software sequence of accessing the status register while the flag is set, followed by a read or write of an associated register. Refer to SECTION 6 SERIAL PERIPHERAL INTERFACE for a description of the SPI system and its interrupts.

3.3 LOW POWER MODES

3.3.1 STOP Instruction

The STOP instruction places the TMP68HC05C4 in its lowest power consumption mode. In the STOP mode the internal oscillator is turned off, causing all internal processing to be halted;refer to Figure 3.3. During the STOP mode, the I bit in the condition code register is cleared to enable external interrupts. All other registers and memory remain unaltered and all input/output lines remain unchanged. This continues until an external interrupt ($\overline{IRQ}$) or reset is sensed at which time the internal oscillator is turned on. The external interrupt or reset causes the program counter to vector to memory location $1FFA and $1FFB or $1FFE and $1FFF which contains the starting address of the interrupt or reset service routine respectively.

3.3.2 WAIT Instruction

The WAIT instruction places the TMP68HC05C4 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 remains active, and all CPU processing is stopped;however, the programmable timer, serial peripheral interface, and serial communications interface systems remain active. Refer to Figure 3.3. During the WAIT mode, the I bit in the condition code register is cleared to enable all interrupts. All other registers and memory remain unaltered and all parallel input/output lines remain unchanged. This continues until any interrupt or reset is sensed. At this time the program counter vectors to the memory location ($1FF4 through $1FFF) which contains the starting address of the interrupt or reset service routine.

3.4 DATA RETENTION MODE

The contents of RAM and CPU registers are retained at supply voltages as low as 2.0 Vdc. This is referred to as the data retention mode, where the data is held, but the device is not guaranteed to operate.

4. PROGRAMMABLE TIMER

4.1 INTRODUCTION

The programmable timer, which is preceded by a fixed divide-by-four prescaler, can be used for many purposes, including input waveform measurements while simultaneously generating an output waveform. Pulse widths can vary from several microseconds to many seconds. A block diagram of the timer is shown in Figure 4.1 and timing diagrams are shown in Figures 4.2 through 4.5.

Because the timer has a 16-bit architecture, each specific functional segment (capability) is represented by two registers. These registers contain the high and low byte of that functional segment. Generally, accessing the low byte of a specific timer function allows full control of that function;however, an access of the high byte inhibits that specific timer function until the low byte is also accessed.

Note : The I bit in the condition code register should be set while manipulating both the high and low byte register of a specific timer function to ensure that an interrupt does not occur. This prevents interrupts from occurring between the time that the high and low bytes are accessed.

The programmable timer capabilities are provided by using the following ten addressable 8-bit registers (note the high and low represent the significance of the byte). A description of each register is provided below.

Timer Control Register (TCR) location $12,

Timer Status Register (TSR) location $13,

Input Capture High Register location $14,

Input Capture Low Register location $15,

Output Compare High Register location $16,

Output Compare Low Register location $17,

Counter High Register location $18,

Counter Low Register location $19,

Alternate Counter High Register location $1A, and

Alternate Counter Low Register location $1B.

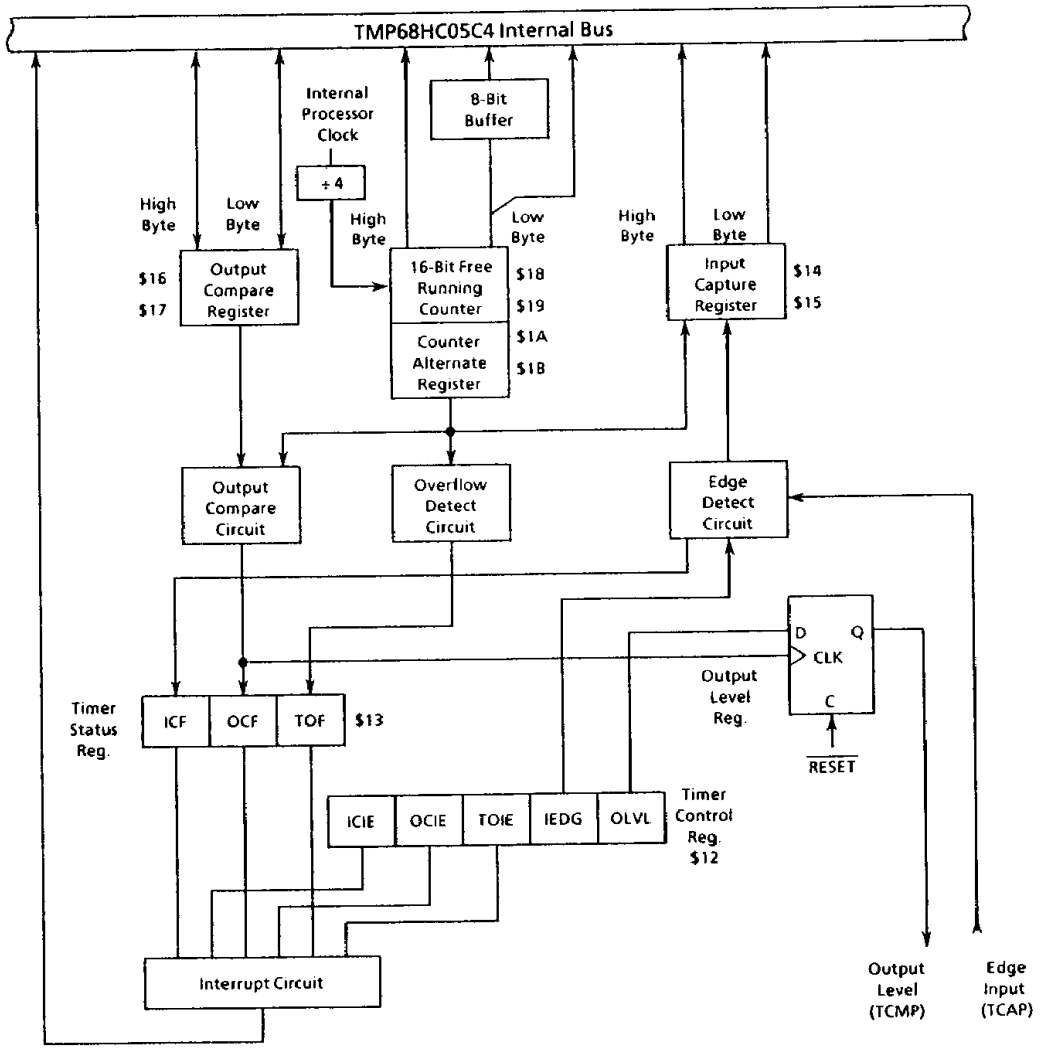
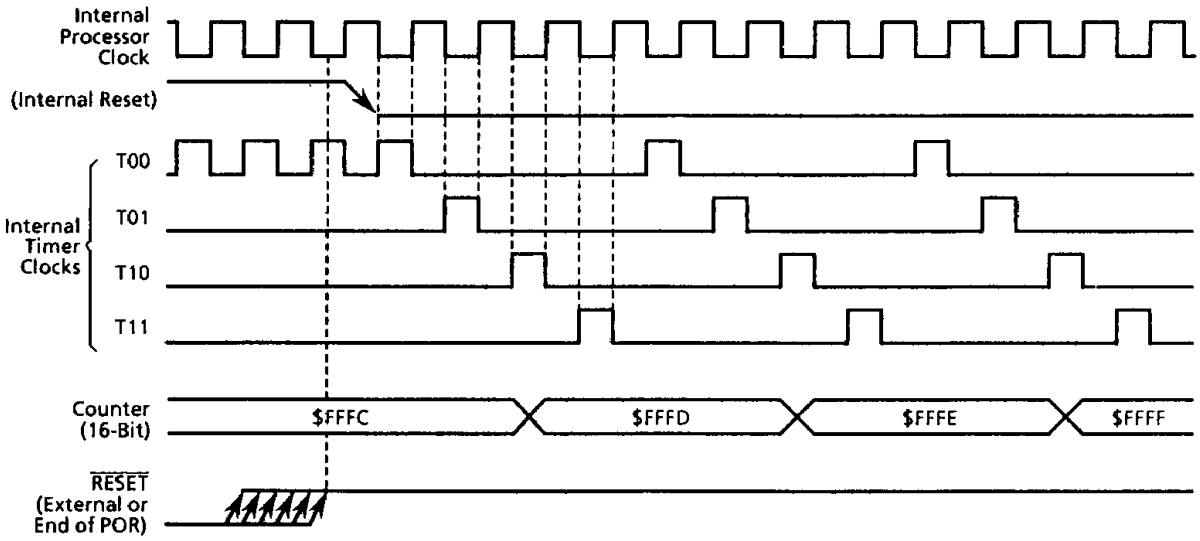
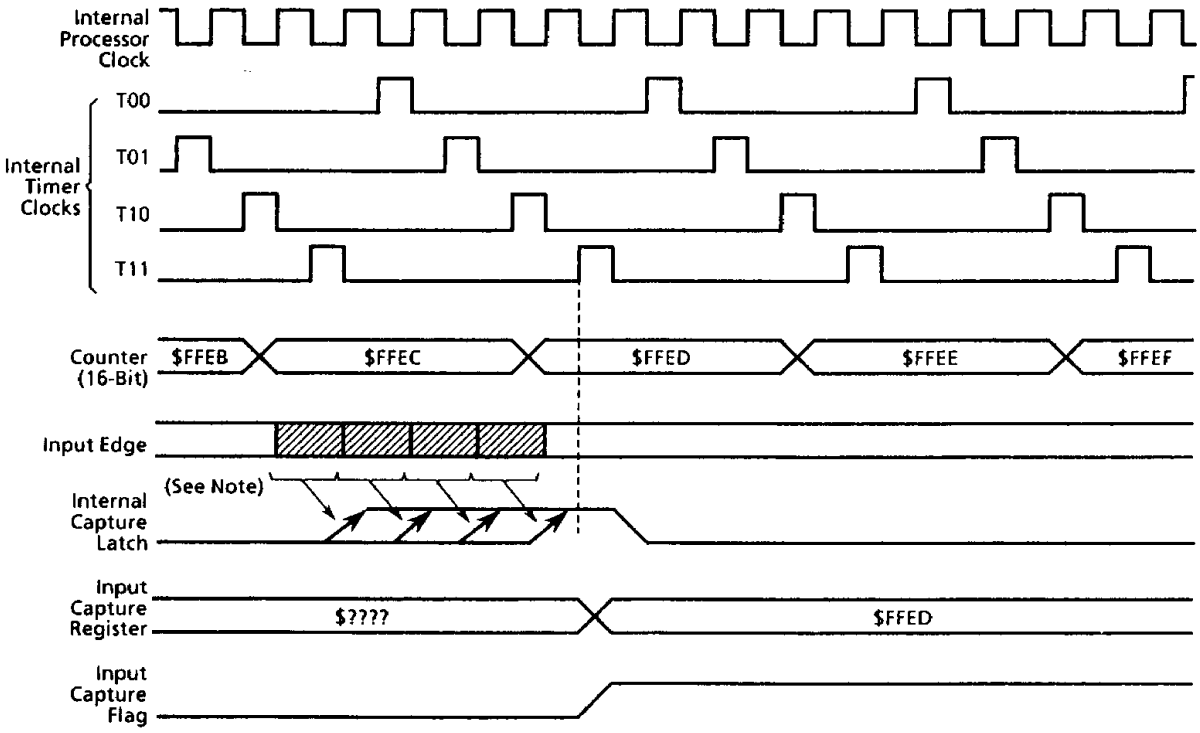


Figure 4.1 Programmable Timer Block Diagram



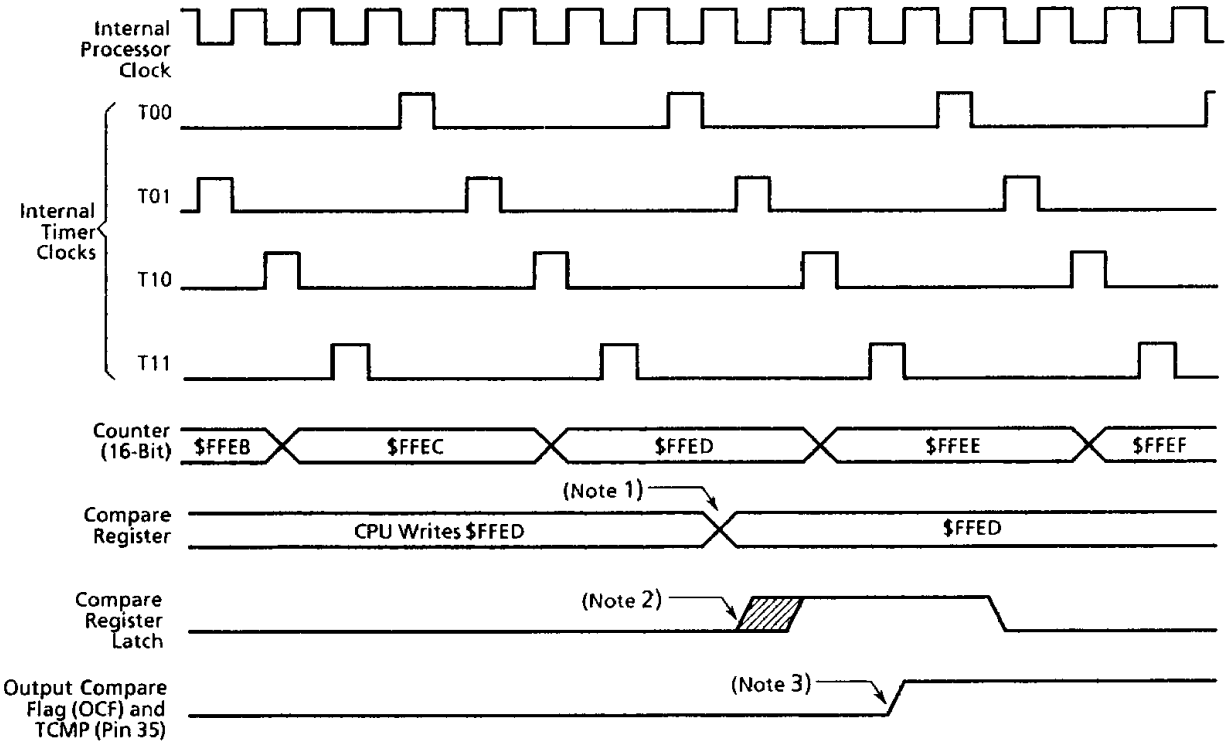
Note : The Counter Register and Timer Control Register are the only ones affected by $\overline{\text{RESET}}$.

Figure 4.2 Timer State Timing Diagram for Reset



Note : If the input edge occurs in the shaded area from one timer state T10 to the other timer state T10 the input capture flag is set during the next state T11.

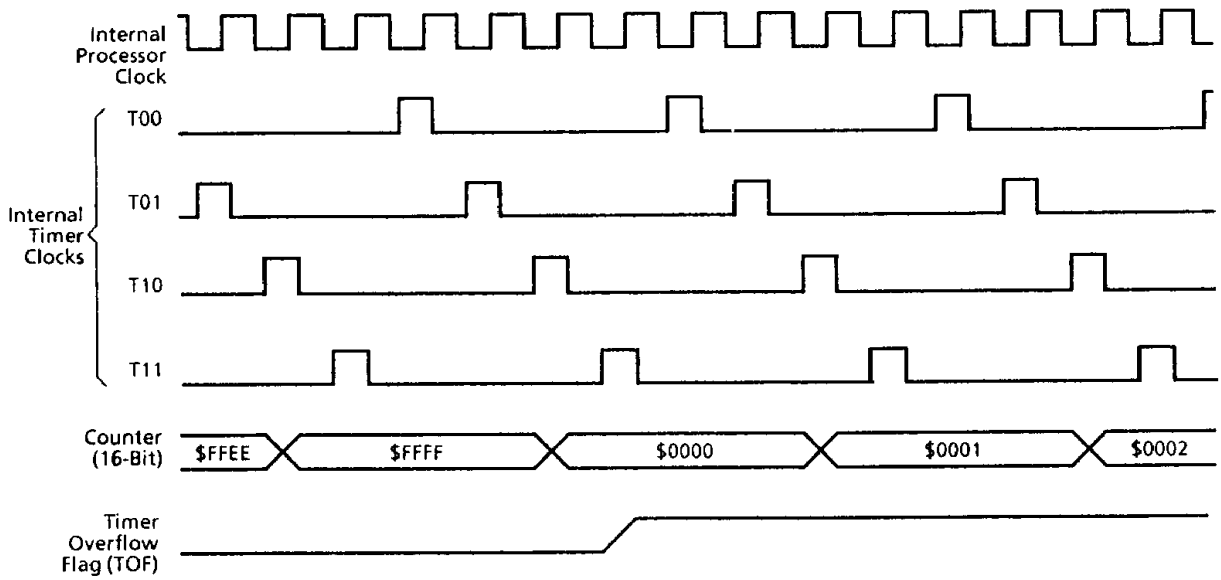
Figure 4.3 Timer State Timing Diagram for Input Capture



Notes :

1. The CPU write to the compare register may take place at any time, but a compare only occurs at timer state T01. Thus, a 4-cycle difference may exist between the write to the compare register and the actual compare.
2. Internal compare takes place during timer state T01.
3. OCF is set at the timer state T11 which follows the comparison match ($FFED in this example).

Figure 4.4 Timer State Timing Diagram for Output Compare



Note : The TOF bit is set at timer state T11 (transition of counter from $FFFF to $0000). It is cleared by a read of the timer status register during the internal processor clock high time followed by a read of the counter low register.

Figure 4.5 Timer State Diagram for Timer Overflow

4.2 COUNTER

The key element in the programmable timer is a 16-bit free running counter, or counter register, preceded by a prescaler which divides the internal processor clock by four. The prescaler gives the timer a resolution of 2.0 microseconds if the internal processor clock is 2.0 MHz. The counter is clocked to increasing values during the low portion of the internal processor clock. Software can read the counter at any time without affecting its value.

The double byte free running counter can be read from either of two locations $18-$19 (called counter register at this location), or $1A-$1B (counter alternate register at this location). A read sequence containing only a read of the least significant byte of the free running counter ($19, $1B) will receive the count value at the time of the read. If a read of the free running counter or counter alternate register first addresses the most significant byte ($18, $1A) it causes the least significant byte ($19, $1B) to be transferred to a buffer. This buffer value remains fixed after the first most significant byte "read" even if the user reads the most significant byte several times. This buffer is accessed when reading the free running counter or counter alternate register least significant byte ($19 or $1B), and thus completes a read sequence of the total counter value. Note that in reading either the free running counter or counter alternate register, if the most significant byte is read, the least significant byte must also be read in order to complete the sequence.

The free running counter is configured to $FFFC during reset and is always a read-only register. During a power-on-reset (POR), the counter is also configured to $FFFC and begins running after the oscillator startup delay. Because the free running counter is 16 bits preceded by a fixed divide-by-four prescaler, the value in the free running counter repeats every 262, 144 MPU internal processor clock cycles. When the counter rolls over from $FFFF to $0000, the timer overflow flag (TOF) bit is set. An interrupt can also be enabled when counter rollover occurs by setting its interrupt enable bit (TOIE).

4.3 OUTPUT COMPARE REGISTER

The output compare register is a 16-bit register, which is made up of two 8-bit registers at locations $16 (most significant byte) and $17 (least significant byte). The output compare register can be used for several purposes such as, controlling an output waveform or indicating when a period of time has elapsed. The output compare register is unique in that all bits are readable and writable and are not altered by the timer hardware. Reset does not affect the contents of this register and if the compare function is not utilized, the two bytes of the output compare register can be used as storage locations.

The contents of the output compare register are compared with the contents of the free running counter once during every four internal processor clocks. If a match is found, the corresponding output compare flag (OCF) bit is set and the corresponding output level (OLVL) bit is clocked (by the output compare circuit pulse) to an output level register. The values in the output compare register and the output level bit should be changed after each successful comparison in order to control an output waveform or establish a new elapsed timeout. An interrupt can also accompany a successful output compare provided the corresponding interrupt enable bit, OCIE, is set.

After a processor write cycle to the output compare register containing the most significant byte ($16), the output compare function is inhibited until the least significant byte ($17) is also written. The user must write both bytes (locations) if the most significant byte is written first. A write made only to the lest significant byte ($17) will not inhibit the compare function. The free running counter is updated every four internal processor clock cycles due to the internal prescaler. The minimum time required to update the output compare register is a function of the software program rather than the internal hardware.

A processor write may be made to either byte of the output compare register without affecting the other byte. The output level (OLVL) bit is clocked to the input of the output level register regardless of whether the output compare flag (OCF) is set or clear. A valid output compare must occur before the OLVL bit becomes available at the output compare pin (TCMP).

Because neither the output compare flag (OCF bit) or output compare register is affected by reset, care must be exercised when initializing the output compare function with software. The following procedure is recommended:

(1) Write to the high byte of the output compare register to inhibit further compares until the low byte is written.

(2) Read the timer status register to arm the OCF if it is already set.

(3) Write to the low byte of the output compare register to enable the output compare function with the flag clear.

The advantage of this procedure is to prevent the OCF bit from being set between the time it is read and the write to the output compare register. A software example is shown below.

```
B7  16  STA  OCMPHI   INHIBIT OUTPUT COMPARE
B6  13  LDA  TSTAT    ARM OCF BIT IF SET
BF  17  STX  OCMPLD   READY FOR NEXT COMPARE
```

4.4 INPUT CAPTURE REGISTER

The two 8-bit registers which make up the 16-bit input capture register are read-only and are used to latch the value of the free running counter after a defined transition is sensed by the corresponding input capture edge detector. The level transition which triggers the counter transfer is defined by the corresponding input edge bit (IEDG). Reset does not affect the contents of the input capture register.

The result obtained by an input capture will be one more than the value of the free running counter on the rising edge of the internal processor clock preceding the external transition (refer to timing diagram shown in Figure 4.3). This delay is required for internal synchronization. Resolution is affected by the prescaler allowing the timer to only increment every four internal processor clock cycles.

The free running counter contents are transferred to the input capture register on each proper signal transition regardless of whether the input capture flag (ICF) is set or clear. The input capture register always contains the free running counter value which corresponds to the most recent input capture.

After a read of the most significant byte of the input capture register ($14), counter transfer is inhibited until the least significant byte ($15) of the input capture register is also read. This characteristic forces the minimum pulse period attainable to be determined by the time used in the capture software routine and its interaction with the main program. The free running counter increments every four internal processor clock cycles due to the prescaler.

A read of the least significant byte ($15) of the input capture register does not inhibit the free running counter transfer. Again, minimum pulse periods are ones which allow software to read the least significant byte ($15) and perform needed operations. There is no conflict between the read of the input capture register and the free running counter transfer since they occur on opposite edges of the internal processor clock.

4.5 TIMER CONTROL REGISTER (TCR)

The timer control register (TCR, location $12) is an 8-bit read/write register which contains five control bits. Three of these bits control interrupts associated with each of the three flag bits found in the timer status register (discussed below). The other two bits control: 1) which edge is significant to the input capture edge detector (i.e., negative or positive), and 2) the next value to be clocked to the output level register in response to a successful output compare. The timer control register and the free running counter are the only sections of the timer affected by reset. The TCMP pin is forced low during external reset and stays low until a valid compare changes it to a high. The timer control register is illustrated below followed by a definition of each bit.

7	6	5	4	3	2	1	0	
ICIE	OCIE	TOIE	0	0	0	IEDG	OLVL	$12

B7, ICIE — If the input capture interrupt enable (ICIE) bit is set, a timer interrupt is enabled when the ICF status flag (in the timer status register) is set. If the ICIE bit is clear, the interrupt is inhibited. The ICIE bit is cleared by reset.

B6, OCIE — If the output compare interrupt enable (OCIE) bit is set, a timer interrupt is enabled whenever the OCF status flag is set. If the OCIE bit is clear, the interrupt is inhibited. The OCIE bit is cleared by reset.

B5, TOIE — If the timer overflow interrupt enable (TOIE) bit is set, a timer interrupt is enabled whenever the TOF status flag (in the timer status register) is set. If the TOIE bit is clear, the interrupt is inhibited. The TOIE bit is cleared by reset.

B1, IEDG — The value of the input edge (IEDG) bit determines which level transition on TCMP pin will trigger a free running counter transfer to the input capture register. Reset does not affect the IEDG bit.

0 = negative edge

1 = positive edge

B0, OLVL The value of the output level (OLVL) bit is clocked into the output level register by the next successful output compare and will appear at TCMP pin. This bit and the output level register are cleared by reset.

0 = low output

1 = high output

4.6 TIMER STATUS REGISTER (TSR)

The timer status register (TSR) is an 8-bit register of which the three most significant bits contain read-only status information. These three bits indicate the following:

(1) A proper transition has taken place at TCAP pin with an accompanying transfer of the free running counter contents to the input capture register,

(2) A match has been found between the free running counter and the output compare register, and

(3) A free running counter transition from $FFFF to $0000 has been sensed (timer overflow).

The timer status register is illustrated below followed by a definition of each bit. Refer to timing diagrams shown in Figure 4.2, 4.3, and 4.4 for timing relationship to the timer status register bits.

7	6	5	4	3	2	1	0	
ICF	OCF	TOF	0	0	0	0	0	$13

B7, ICF The input capture flag (ICF) is set when a proper edge has been sensed by the input capture edge detector. It is cleared by a processor access of the timer status register (with ICF set) followed by accessing the low byte ($15) of the input capture register. Reset does not affect the input compare flag.

B6, OCF The output compare flag (OCF) is set when the output compare register contents matches the contents of the free running counter. The OCF is cleared by accessing the timer status register (with OCF set) and then accessing the low byte ($17) of the output compare register. Reset does not affect the output compare flag.

B5, TOF The timer overflow flag (TOF) bit is set by a transition of the free running counter from $FFFF to $0000. It is cleared by accessing the timer status register (with TOF set) followed by an access of the free running counter least significant byte ($19). Reset does not affect the TOF bit.

Accessing the timer status register satisfies the first condition required to clear any status bits which happen to be set during the access. The only remaining step is to provide an access of the register which is associated with the status bit. Typically, this presents no problem for the input capture and output compare functions.

A problem can occur when using the timer overflow function and reading the free running counter at random times to measure an elapsed time. Without incorporating the proper precautions into software, the timer overflow flag could unintentionally be cleared if: 1) the timer status register is read or written when TOF is set, and 2) the least significant byte of the free running counter is read but not for the purpose of servicing the flag. The counter alternate register at address $1A and $1B contains the same value as the free running counter (at address $18 and $19);therefore, this alternate register can be read at any time without affecting the timer overflow flag in the timer status register.

During STOP and WAIT instructions, the programmable timer functions as follows: during the wait mode, the timer continues to operate normally and may generate an interrupt to trigger the CPU out of the wait state;during the stop mode, the timer holds at its current state, retaining all data, and resumes operation from this point when an external interrupt is received.

5. SERIAL COMMUNICATIONS INTERFACE (SCI)

5.1 INTRODUCTION

A full-duplex asynchronous serial communications interface (SCI) is provided with a standard NRZ format and a variety of baud rates. The SCI transmitter and receiver are functionally independent, but use the same data format and bit rate. The serial data format is standard mark/space (NRZ) which provide one start bit, eight or nine data bits, and one stop bit. "Baud" and "bit rate" are used synonymously in the following description.

5.1.1 SCI Two Wire System Features

- Standard NRZ (mark/space) format.
- Advanced error detection method includes noise detection for noise duration of up to 1/16 bit time.
- Full-duplex operation (simultaneous transmit and receive).
- Software programmable for one of 32 different baud rates.
- Software selectable word length (eight or nine bit words).
- Separate transmitter and receiver enable bits.
- SCI may be interrupt driven.
- Four separate enable bits available for interrupt control.

5.1.2 SCI Receiver Features

- Receiver wake-up function (idle or address bit).
- Idle line detect.
- Framing error detect.
- Noise detect.
- Overrun detect.
- Receiver data register full flag.

5.1.3 SCI Transmitter Features

- Transmit data register empty flag.
- Transmit complete flag.
- Break send.

Any SCI two-wire system requires receive data in (RDI) and transmit data out (TDO).

5.2 DATA FORMAT

Receive data in (RDI) or transmit data out (TDO) is the serial data which is presented between the internal data bus and the output pin (TDO), and between the input pin (RDI) and the internal data bus. Data format is as shown for the NRZ in Figure 5.1 and must meet the following criteria:

(1) A high level indicates a logic one and a low level indicates a logic zero.

(2) The idle line is in a high (logic one) state prior to transmission/reception of a message.

(3) A start bit (logic zero) is transmitted/received indicating the start of a message.

(4) The data is transmitted and received least-significant-bit first.

(5) A stop bit (high in the tenth or eleventh bit position) indicates the byte is complete.

(6) A break is defined as the transmission or reception of a low (logic zero) for some multiple of the data format.

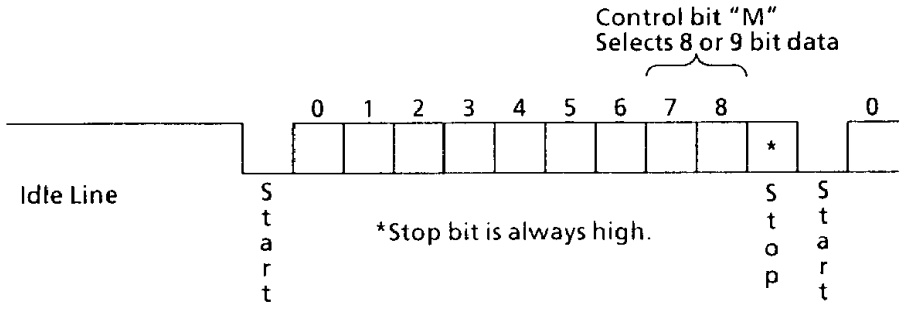


Figure 5.1 Data Format

5.3 WAKE-UP FEATURE

In a typical multiprocessor configuration, the software protocol will usually identify the addresse (s) at the beginning of the message. In order to permit uninterested MPUs to ignore the remainder of the message, a wake-up feature is included whereby all further SCI receiver flag (and interrupt) processing can be inhibited until its data line returns to the idle state. An SCI receiver is re-enabled by an idle string of at least ten (or eleven) consecutive ones. Software for the transmitter must provide for the required idle string between consecutive messages and prevent it from occurring within messages.

The user is allowed a second method of providing the wake-up feature in lieu of the idle string discussed above. This method allows the user to insert a logic one in the most significant bit of the transmit data word which needs to be received by all "sleeping" processors.

5.4 RECEIVE DATA IN

Receive data in is the serial data which is presented from the input pin via the SCI to the internal data bus. While waiting for a start bit, the receiver samples the input at a rate which is 16 times higher than the set baud rate. This 16 times higher-than-baud rate is referred to as the RT rate in Figures 5.2 and 5.3, and as the receiver clock in Figure 5.7. When the input (idle) line is detected low, it is tested for three more sample times (referred to as the start edge verification samples in Figure 5.2). If at least two of these three verification samples detect a logic low, a valid start bit is assumed to have been detected (by a logic low following the three start qualifiers) as shown in Figure 5.2; however, if in two or more of the verification samples a logic high is detected, the line is

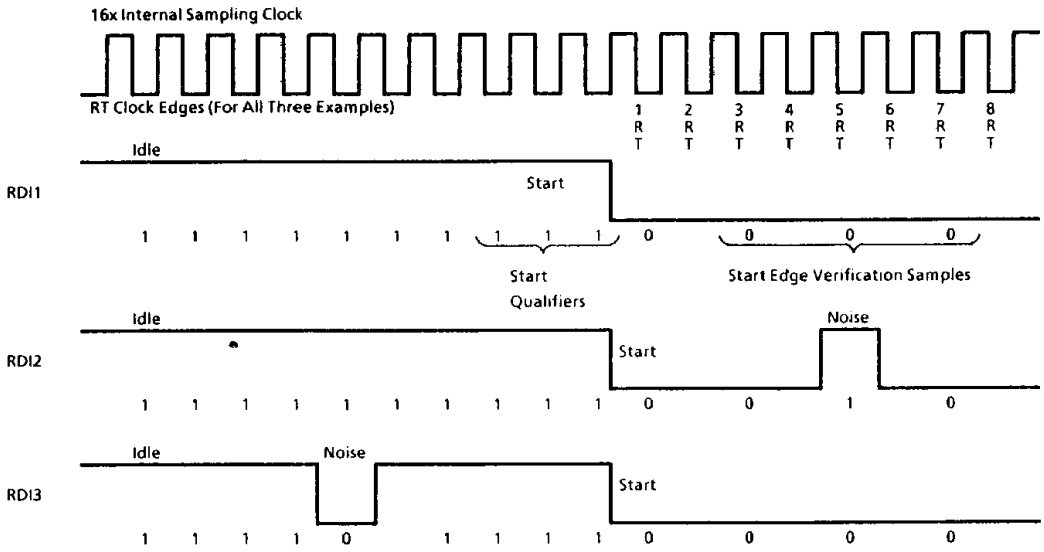


Figure 5.2 Examples of Start-Bit Sampling Technique

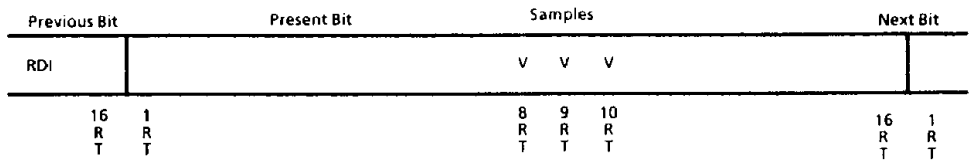


Figure 5.3 Sampling Technique Used on All Bits

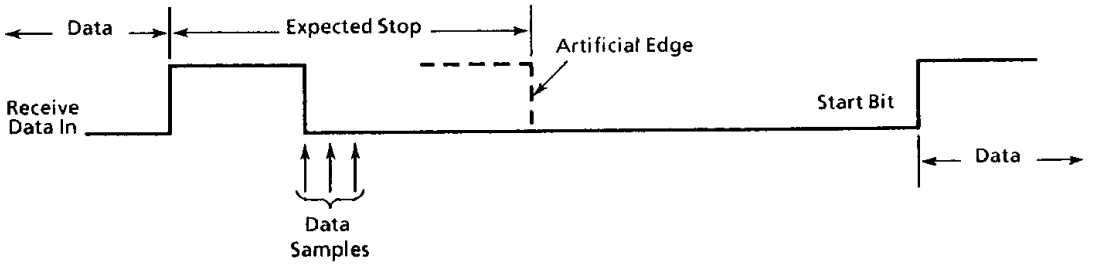
assumed to be idle. (A noise flag is set if one of the three verification samples detects a logic high, thus a valid start bit could be assumed and a noise flag still set.) The receiver clock generator is controlled by the baud rate register (see Figures 5.6 and 5.7);however, the serial communications interface is synchronized by the start bit (independent of the transmitter).

Once a valid start bit is detected, the start bit, each data bit, and the stop bit are sampled three times at RT intervals of 8RT, 9RT, and 10RT (1RT is the position where the bit is expected to start) as shown in Figure 5.3. The value of the bit is determined by voting logic which takes the value of the majority of samples (two or three out of three). A noise flag is set when all three samples on a valid start bit or a data bit or the stop bit do not agree. (As discussed above, a noise flag is also set when the start bit verification samples do not agree.)

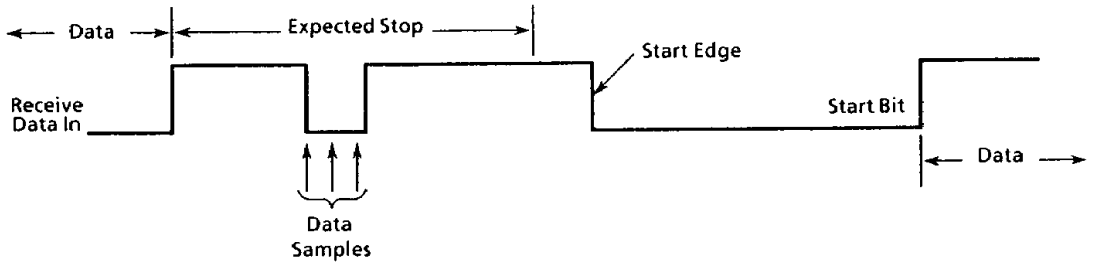
5.5 START BIT DETECTION FOLLOWING A FRAMING ERROR

If there has been a framing error without detection of a break (10 zeros for 8-bit format or 11 zeros for 9-bit format), the circuit continues to operate as if there actually were a stop bit and the start edge will be placed artificially. The last bit received in the data shift register is inverted to a logic one, and the three logic one start qualifiers (shown in Figure 5.2) are forced into the sample shift register during the interval when detection of a start bit is anticipated (see Figure 5.4);therefore the start bit will be accepted no sooner than it is anticipated.

If the receiver detects that a break (RDRF=1, FE=1, receiver data register=$00) produced the framing error, the start bit will not be artificially induced and the receiver must actually receive a logic one bit before start. See Figure 5.5.



(a) Case 1, Receive Line Low During Artificial Edge



(b) Case 2, Receive Line High During Expected Start Edge

Figure 5.4 SCI Artificial Start Following A Framing Error

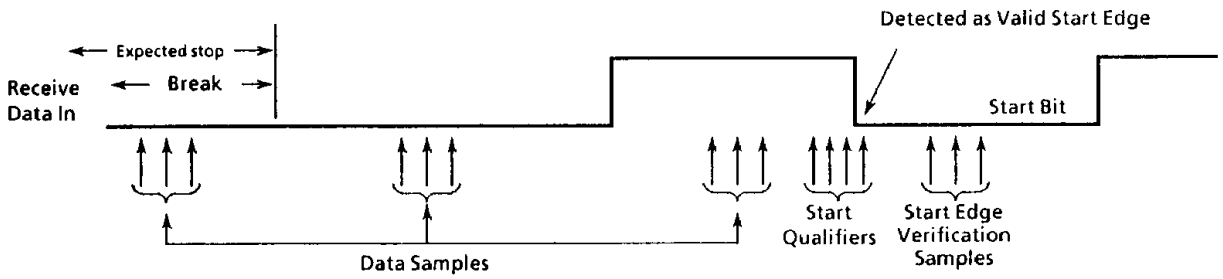


Figure 5.5 SCI Start Bit Following A Break

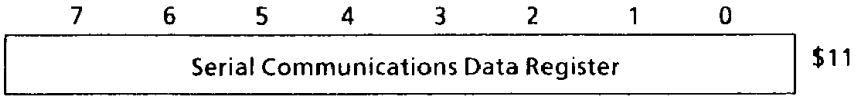
5.6 TRANSMIT DATA OUT (TDO)

Transmit data out is the serial data which is presented from the internal data bus via the SCI and then to the output pin. Data format is as discussed above and shown in Figure 5.1. The transmitter generates a bit time by using a derivative of the RT clock, thus producing a transmission rate equal to 1/16 that of the receiver sample clock.

5.7 REGISTERS

There are five different registers used in the serial communications interface (SCI) and the internal configuration of these registers is discussed in the following paragraphs. A block diagram of the SCI system is shown in Figure 5.6.

5.7.1 Serial Communications Data Register (SCDAT)



The serial communications data register (SCDAT) performs two functions in the serial communications interface;i.e. it acts as the receive data register when it is read and as the transmit data register when it is written. Figure 5.6 shows this register as two separate registers, namely:the receive data register (RDR) and the transmit data register (TDR). As shown in Figure 5.6, the TDR (transmit data register) provides the parallel interface from the internal data bus to the transmit shift register and the receive data register (RDR) provides the interface from the receive shift register to the internal data bus.

When SCDAT is read, it becomes the receive data register and contains the last byte of data received. The receive data register, represented above, is a read-only register containing the last byte of data received from the shift register for the internal data bus. The RDRF bit (receive data register full bit in the serial communications status register) is set to indicate that a byte has been transferred from the input serial shift register to the serial communications data register. The transfer is synchronized with the receiver

bit rate clock (from the receive control) as shown in Figure 5.6. All data is received least-significant-bit first.

When SCDAT is written, it becomes the transmit data register and contains the next byte of data to be transmitted. The transmit data register, also represented above, is a write-only register containing the next byte of data to be applied to the transmit shift register from the internal data bus. As long as the transmitter is enabled, data stored in the serial communications data register is transferred to the transmit shift register (after the current byte in the shift register has been transmitted). The transfer from the SCDAT to the transmit shift register is synchronized with the bit rate clock (from the transmit control) as shown in Figure 5.6. All data is transmitted least-significant-bit first.

5.7.2 Serial Communications Control Register 1 (SCCR1)

7	6	5	4	3	2	1	0	
R8	T8	-	M	WAKE	-	-	-	$0E

The serial communications control register 1 (SCCR1) provides the control bits which:1) determine the word length (either 8 or 9 bits), and 2) selects the method used for the wake-up feature. Bits 6 and 7 provide a location for storing the ninth bit for longer bytes.

B7, R8 — If the M bit is a one, then this bit provides a storage location for the ninth bit in the receive data byte. Reset does not affect this bit.

B6, T8 — If the M bit is a one, then this bit provides a storage location for the ninth bit in the transmit data byte. Reset does not affect this bit.

B4, M — The option of the word length is selected by the configuration of this bit and is shown below. Reset does not affect this bit.

0 = 1 start bit, 8 data bits, 1 stop bit

1 = 1 start bit, 9 data bits, 1 stop bit

B3, WAKE — This bit allows the user to select the method for receiver "wake up". If the WAKE bit is a logic zero, an idle line condition will "wake up" the receiver. If the WAKE bit is set to a logic one, the system acknowledges an address bit (most significant bit). The address bit is dependent on both the WAKE bit and the M bit level (table shown below). (Additionally, the receiver does not use the wake-up feature unless the RWU control bit in serial communications control register 2 is set as discussed below.) Reset does not affect this bit.

Wake	M	Method of Receiver "Wake-Up"
0	X	Detection of an idle line allows the next data byte received to cause the receive data register to fill and produce an RDRF flag.
1	0	Detection of a received one in the eighth data bit allows an RDRF flag and associated error flags.
1	1	Detection of a received one in the ninth data bit allows an RDRF flag and associated error flags.

5.7.3 Serial Communications Control Register 2 (SCCR2)

7	6	5	4	3	2	1	0	
TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK	$0F

The serial communications control register 2 (SCCR2) provides the control bits which:individually enable/disable the transmitter or receiver, enable the system interrupts, and provide the wake-up enable bit and a "send break code" bit. Each of these bits is described below. (The individual flags are discussed in the 5.7.4 Serial Communications Status Register.)

B7, TIE — When the transmit interrupt enable bit is set, the SCI interrupt occurs provided TDRE is set (see Figure 5.6). When TIE is clear, the TDRE interrupt is disabled. Reset clears the TIE bit.

B6, TCIE — When the transmission complete interrupt enable bit is set, the SCI interrupt occurs provided TC is set (see Figure 5.6). When TCIE is clear, the TC interrupt is disabled. Reset clears the TCIE bit.

B5, RIE — When the receive interrupt enable bit is set, the SCI interrupt occurs provided OR is set or RDRF is set (see Figure 5.6). When RIE is clear, the OR and RDRF interrupts are disabled. Reset clears the RIE bit.

B4,ILIE — When the idle interrupt enable bit is set, the SCI interrupt occurs provided IDLE is set (see Figure 5.6). When ILIE is clear, the IDLE interrupt is disabled. Reset clears the ILIE bit.

B3, TE — When the transmit enable bit is set, the transmit shift register output is applied to the TDO line. Depending on the state of control bit M in serial communications control register 1, a preamble of 10 (M=0) or 11 (M=1) consecutive ones is transmitted when software sets the TE bit from a cleared state. If a transmission is in progress, and TE is written to a zero, then the transmitter will wait until after the present byte has been transmitted before placing the TDO pin in the idle high-impedance state. If the TE bit has been written to a zero and then set to a one before the

current byte is transmitted, the transmitter will wait until that byte is transmitted and will then initiate transmission of a new preamble. After the preamble is transmitted, and provided the TDRE bit is set (no new data to transmit), the line remains idle (driven high while TE=1);otherwise, normal transmission occurs. This function allows the user to "neatly" terminate a transmission sequence. After loading the last byte in the serial communications data register and receiving the interrupt from TDRE, indicating the data has been transferred into the shift register, the user should clear TE. The last byte will then be transmitted and the line will go idle (high impedance). Reset clears the TE bit.

B2, RE — When the receive enable bit is set, the receiver is enabled. When RE is clear, the receiver is disabled and all of the status bits associated with the receiver (RDRF, IDLE, OR, NF, and FE) are inhibited. Reset clears the RE bit.

B1, RWU — When the receiver wake-up bit is set, it enables the "wake up" function. The type of "wake up" mode for the receiver is determined by the WAKE bit discussed above (in the SCCR1). When the RWU bit is set, no status flags will be set. Flags which were set previously will not be cleared when RWU is set. If the WAKE bit is cleared, RWU is cleared after receiving 10 (M=0) or 11 (M=1) consecutive ones. Under these conditions, RWU cannot be set if the line is idle. If the WAKE bit is set, RWU is cleared

after receiving an address bit. The RDRF flag will then be set and the address byte will be stored in the receiver data register. Reset clears the RWU bit.

B0, SBK — When the send break bit is set the transmitter sends zeros in some number equal to a multiple of the data format bits. If the SBK bit is toggled set and clear, the transmitter sends 10 (M=0) or 11 (M=1) zeros and then reverts to idle or sending data. The actual number of zeros sent when SBK is toggled depends on the data format set by the M bit in the serial communications control register 1;therefore, the break code will be synchronous with respect to the data stream. At the completion of the break code, the transmitter sends at least one high bit to guarantee recognition of a valid start bit. Reset clears the SBK bit.

5.7.4 Serial Communications Status Register (SCSR)

7	6	5	4	3	2	1	0	
TDRE	TC	RDRF	IDLE	OR	NF	FE	-	$10

The serial communications status register (SCSR) provides inputs to the interrupt logic circuits for generation of the SCI system interrupt. In addition, a noise flag bit and a framing error bit are also contained in the SCSR.

B7, TDRE The transmit data register empty bit is set to indicate that the contents of the serial communications data register have been transferred to the transmit serial shift register. If the TDRE bit is clear, it indicates that the transfer has not yet occurred and a write to the serial communications data register will overwrite the previous value. The TDRE bit is cleared by accessing the serial communications status register (with TDRE set), followed by writing to the serial communications data register. Data can not be transmitted unless the serial communications status register is accessed before writing to the serial communications data register to clear the TDRE flag bit. Reset sets the TDRE bit.

B6, TC The transmit complete bit is set at the end of a data frame, preamble, or break condition if:

(1) TE=1, TDRE=1, and no pending data, preamble, or break is to be transmitted;or

(2) TE=0, and the data, preamble, or break (in the transmit shift register) has been transmitted.

The TC bit is a status flag which indicates that one of the above conditions has occurred. The TC bit is cleared by accessing the serial communications status register (with TC set), followed by writing to the serial communications data register. It does not inhibit the transmitter function in any way. Reset sets the TC bit.

B5, RDRF When the receive data register full bit is set, it indicates that the receiver serial shift register is transferred to the serial communications data register. If multiple errors are detected in any one received word, the NF, FE, and RDRF bits will be affected as appropriate during the same clock cycle. The RDRF bit is cleared when the serial communications status register is accessed (with RDRF set) followed by a read of the serial communications data register. Reset clears the RDRF bit.

B4, IDLE When the idle line detect bit is set, it indicates that a receiver idle line is detected (receipt of a minimum number of ones to constitute the number of bits in the byte format). The minimum number of ones needed will be 10 (M=0) or 11 (M=1). This allows a receiver that is not in the wake-up mode to detect the end of a message, detect the preamble of a new message, or to resynchronize with the transmitter. The IDLE bit is cleared by accessing the serial communications status register (with IDLE set)

followed by a read of the serial communications data register. The IDLE bit will not be set again until after an RDRF has been set;i.e., a new idle line occurs. The IDLE bit is not set by an idle line when the receiver "wakes up" from the wake-up mode. Reset clears the IDLE bit.

B3, OR — When the overrun error bit is set, it indicates that the next byte is ready to be transferred from the receive shift register to the serial communications data register when it is already full (RDRF bit is set). Data transfer is then inhibited until the RDRF bit is cleared. Data in the serial communications data register is valid in this case, but additional data received during an overrun condition (including the byte causing the overrun) will be lost. The OR bit is cleared when the serial communications status register is accessed (with OR set), followed by a read of the serial communications data register. Reset clears the OR bit.

B2, NF — The noise flag bit is set if there is noise on a "valid" start bit or if there is noise on any of the data bits or if there is noise on the stop bit. It is not set by noise on the idle line nor by invalid (false) start bits. If there is noise, the NF bit is not set until the RDRF flag is set. Each data bit is sampled three times as described above in RECEIVE DATA IN and shown in Figure 5.3. The NF bit represents the status of the byte in the serial communications data register. For the byte being received (shifted in)

there will also be a "working" noise flag the value of which will be transferred to the NF bit when the serial data is loaded into the serial communications data register. The NF bit does not generate an interrupt because the RDRF bit gets set with NF and can be used to generate the interrupt. The NF bits is cleared when the serial communications status register is accessed (with NF set), followed by a read of the serial communications data register. Reset clears the NF bit.

B1, FE — The framing error bit is set when the byte boundaries in the bit stream are not synchronized with the receiver bit counter (generated by a "lost" stop bit). The byte is transferred to the serial communications data register and the RDRF bit is set. The FE bit does not generate an interrupt because the RDRF bit is set at the same time as FE and can be used to generate the interrupt. Note that if the byte received causes a framing error and it will also cause an overrun if transferred to the serial communications data register, then the overrun bit will be set, but not the framing error bit, and the byte will not be transferred to the serial communications data register. The FE bit is cleared when the serial communications status register is accessed (with FE set) followed by a read of the serial communications data register. Reset clears the FE bit.

5.7.5 Baud Rate Register

7	6	5	4	3	2	1	0	
–	–	SCP1	SCP0	–	SCR2	SCR1	SCR0	$0D

The baud rate register provides the means for selecting different baud rates which may be used as the rate control for the transmitter and receiver. The SCP0-SCP1 bits function as a prescaler for the SCR0-SCR2 bits. Together, these five bits provide multiple, baud rate combinations for a given crystal frequency.

B5, SCP1
B4, SCP0

These two bits in the baud rate register are used as a prescaler to increase the range of standard baud rates controlled by the SCR0-SCR2 bits. A table of the prescaler internal processor clock division versus bit levels is provided below. Reset clears SCP1-SCP0 bits (divide-by-one).

SCP1	SCP0	Internal Processor Clock Divide By
0	0	1
0	1	3
1	0	4
1	1	13

B2, SCR2
B1, SCR1
B0, SCR0

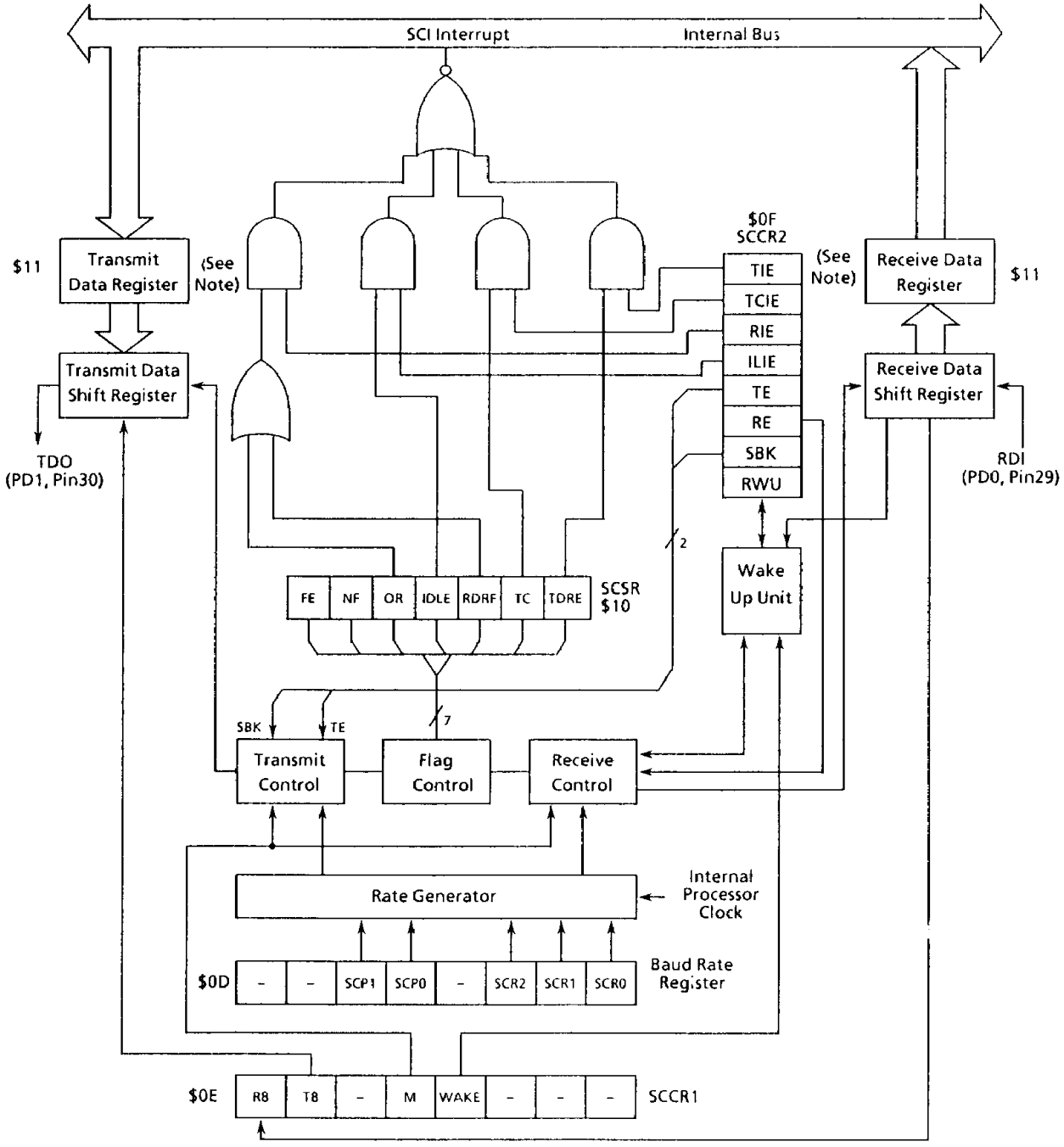
These three bits in the baud rate register are used to select the baud rates of both the transmitter and receiver. A table of baud rates versus bit levels is shown below. Reset does not affect the SCR2-SCR0 bits.

SCR2	SCR1	SCR0	Prescaler Output 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

The diagram of Figure 5.7 and Tables 5.1 and 5.2 illustrate the divided chain used to obtain the baud rate clock (transmit clock). Note that there is a fixed rate divide-by-16 between the receive clock (RT) and the transmit clock (Tx). The actual divider chain is controlled by the combined SCP0-SCP1 and SCR0-SCR2 bits in the baud rate register as

illustrated. All divided frequencies shown in the first table represent the final transmit clock (the actual baud rate) resulting from the internal processor clock division shown in the "divide-by" column only (prescaler division only). The second table illustrates how the prescaler output can be further divided by action of the SCI select bits (SCR0-SCR2). For example, assume that a 9600 Hz baud rate is required with a 2.4576 MHz external crystal. In this case the prescaler bits (SCP0-SCP1) could be configured as a divide-by-one or a divide-by-four. If a divide-by-four prescaler is used, then the SCR0-SCR2 bits must be configured as a divide-by-two. This results in a divide-by-128 of the internal processor clock to produce a 9600 Hz baud rate clock. Using the same crystal, the 9600 baud rate can be obtained with a prescaler divide-by-one and the SCR0-SCR2 bits configured for a divide-by-eight.

Note : The crystal frequency is internally divided-by-two to generate the internal processor clock.



Note: The Serial Communications Data Register (SCDAT) is controlled by the internal R/W signal. It is the transmit data register when written and receive data register when read.

Figure 5.6 Serial Communications Interface Block Diagram

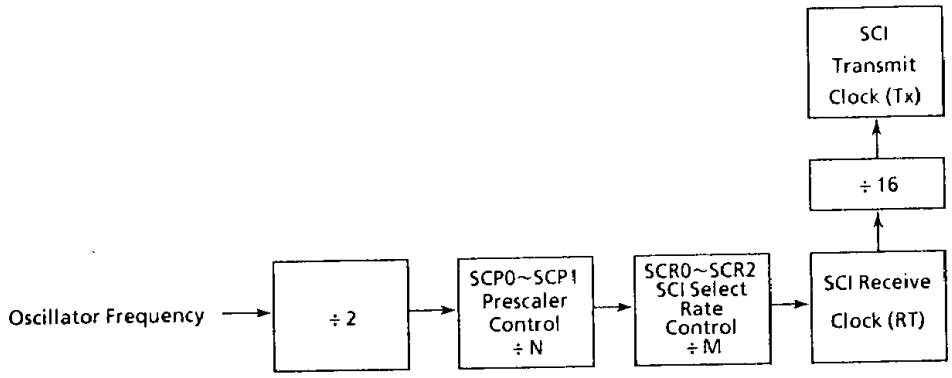


Figure 5.7 Rate Generator Division

Table 5.1 Prescaler Highest Baud Rate Frequency Output

SCP Bit		Clock* Divided By	Crystal Frequency MHz				
1	0		4.194304	4.0	2.4576	2.0	1.8432
0	0	1	131.072 KHz	125.000 KHz	76.80 KHz	62.50 KHz	57.60 KHz
0	1	3	43.691 KHz	41.666 KHz	25.60 KHz	20.833 KHz	19.20 KHz
1	0	4	32.768 KHz	31.250 KHz	19.20 KHz	15.625 KHz	14.40 KHz
1	1	13	10.082 KHz	9600 Hz	5.907 KHz	4800 Hz	4430 Hz

* The clock in the "Clock Divided By" column is the internal processor clock.

Note: The divided frequencies shown in Table 5.1 represent baud rates which are the highest transmit baud rate (Tx) that can be obtained by a specific crystal frequency and only using the prescaler division. Lower baud rates may be obtained by providing a further division using the SCI rate select bits as shown below for some representative prescaler outputs.

Table 5.2 Transmit Baud Rate Output for a Given Prescaler Output

SCR Bits			Divide By	Representative Highest Prescaler Baud Rate Output				
2	1	0		131.072 KHz	32.768 KHz	76.80 KHz	19.20 KHz	9600 KHz
0	0	0	1	131.072 KHz	32.768 KHz	76.80 KHz	19.20 KHz	9600 Hz
0	0	1	2	65.536 KHz	16.384 KHz	38.40 KHz	9600 Hz	4800 Hz
0	1	0	4	32.768 KHz	8.192 KHz	19.20 KHz	4800 Hz	2400 Hz
0	1	1	8	16.384 KHz	4.096 KHz	9600 Hz	2400 Hz	1200 Hz
1	0	0	16	8.192 KHz	2.048 KHz	4800 Hz	1200 Hz	600 Hz
1	0	1	32	4.096 KHz	1.024 KHz	2400 Hz	600 Hz	300 Hz
1	1	0	64	2.048 KHz	512 Hz	1200 Hz	300 Hz	150 Hz
1	1	1	128	1.024 KHz	256 Hz	600 Hz	150 Hz	75 Hz

Note: Table 5.2 illustrates how the SCI select bits can be used to provide lower transmitter baud rates by further dividing the prescaler output frequency. The five examples are only representative samples. In all cases, the baud rates shown are transmit baud rates (trasmit clock) and the receiver clock is 16 times higher in frequency than the actual baud rate.

6. SERIAL PERIPHERAL INTERFACE (SPI)

6.1 INTRODUCTION AND FEATURES

6.1.1 Introduction

The serial peripheral interface (SPI) is an interface built into the TMP68HC05C4 MCU which allows several TMP68HC05C4 MCUs, or TMP68HC05C4 plus peripheral devices, to be interconnected within a single "black box" or on the same printed circuit board. In a serial peripheral interface (SPI), separate wires (signals) are required for data and clock. In the SPI format, the clock is not included in the data stream and must be furnished as a separate signal. An SPI system may be configured in one containing one master MCU and several slave MCUs, or in a system in which an MCU is capable of being either a master or a slave.

Figure 6.1 illustrates two different system configurations. Figure 6.1a represents a system of five different MCUs in which there are one master and four slaves (0, 1, 2, 3). In this system four basic lines (signals) are required for the MOSI (master out slave in), MISO (master in slave out), SCK (serial clock), and $\overline{SS}$ (slave select) lines. Figure 6.1b represents a system of five MCUs in which three can be master or slave and two are slave only.

6.1.2 Features

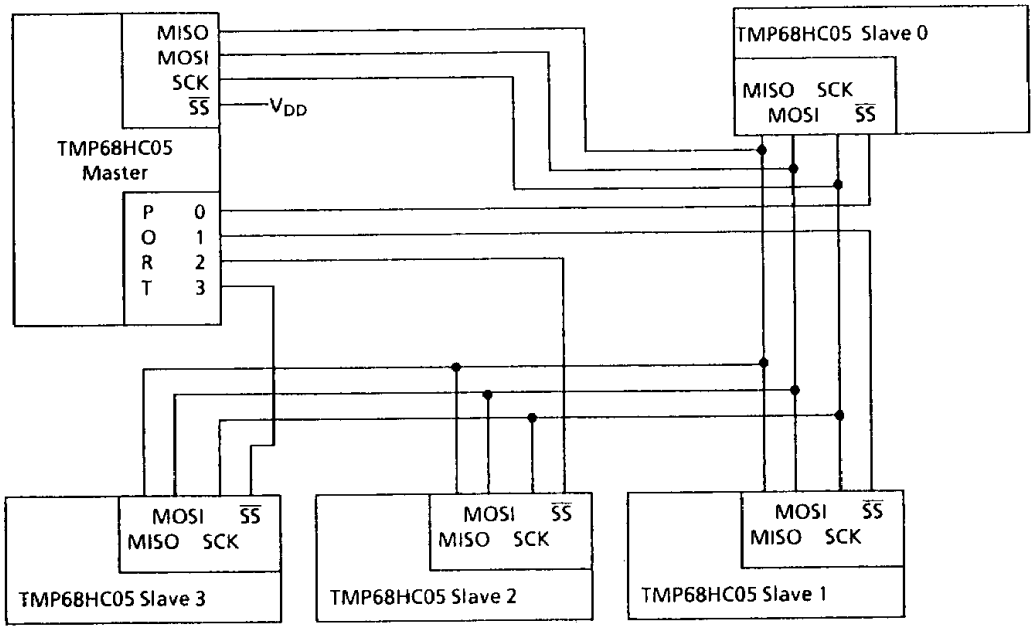
- Full duplex, three-wire synchronous transfers
- Master or slave operation
- 1.05 MHz (maximum) master bit frequency
- 2.1 MHz (maximum) slave bit frequency
- Four programmable master bit rates
- Programmable clock polarity and phase
- End of transmission interrupt flag
- Write collision flag protection
- Master-Master mode fault protection capability

6.2 SIGNAL DESCRIPTION

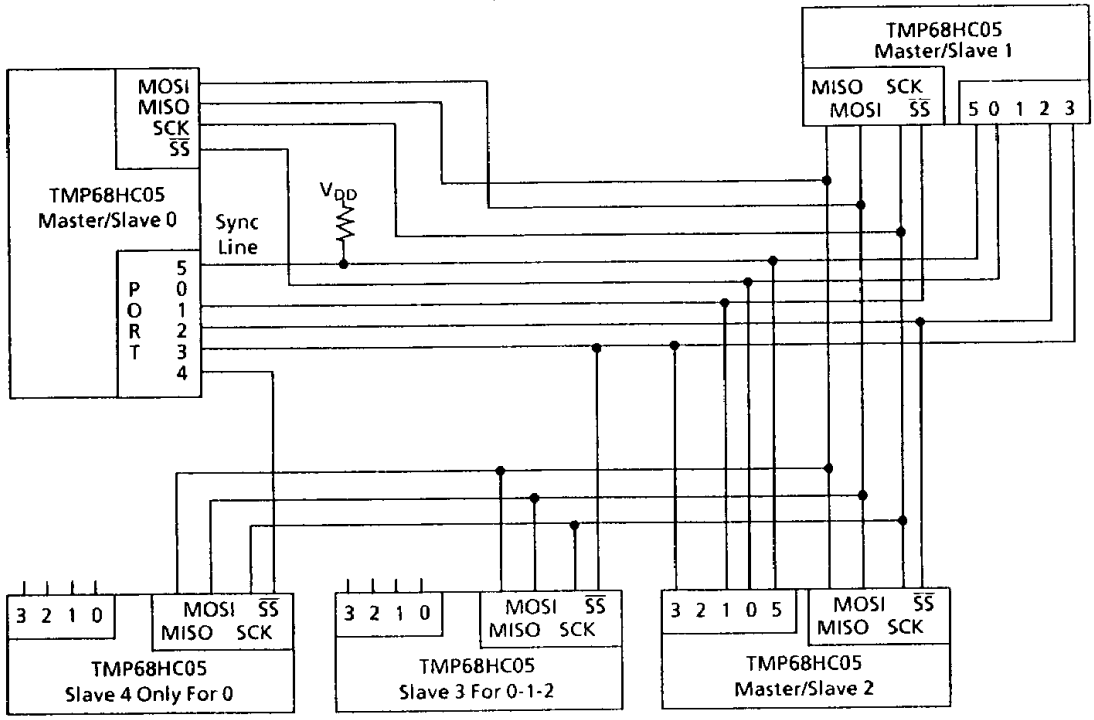
The four basic signals (MOSI, MISO, SCK, and $\overline{SS}$) discussed above are described in the following paragraphs. Each signal function is described for both the master and slave mode.

6.2.1 Master Out Slave In (MOSI)

The MOSI pin is configured as a data output in a master (mode) device and as a data input in a slave (mode) device. In this manner data is transferred serially from a master to a slave on this line;most



a. Single Master, Four Slaves



b. Three Master/Slave, Two Slaves

Figure 6.1 Master-Slave System Configuration

significant bit first, least significant bit last. The timing diagrams of Figure 6.2 summarize the SPI timing diagram shown in Section 9, and show the relationship between data and clock (SCK). As shown in Figure 6.2, four possible timing relationships may be chosen by using control bits CPOL and CPHA. The master device always allows data to be applied on the MOSI line a half-cycle before the clock edge (SCK) in order for the slave device to latch the data.

Note : Both the slave device(s) and a master device must be programmed to similar timing modes for proper data transfer.

When the master device transmits data to a second (slave) device via the MOSI line, the slave device responds by sending data to the master device via the MISO line. This implies full duplex transmission with both data out and data in synchronized with the same clock signal (one which is provided by the master device). Thus, the byte transmitted is replaced by the byte received and eliminates the need for separate transmit-empty and receiver-full status bits. A single status bit (SPIF) is used to signify that the I/O operation is complete.

Configuration of the MOSI pin is a function of the MSTR bit in the serial peripheral control register (SPCR, location $0A). When a device is operating as a master, the MOSI pin is an output because the program in firmware sets the MSTR bit to a logic one.

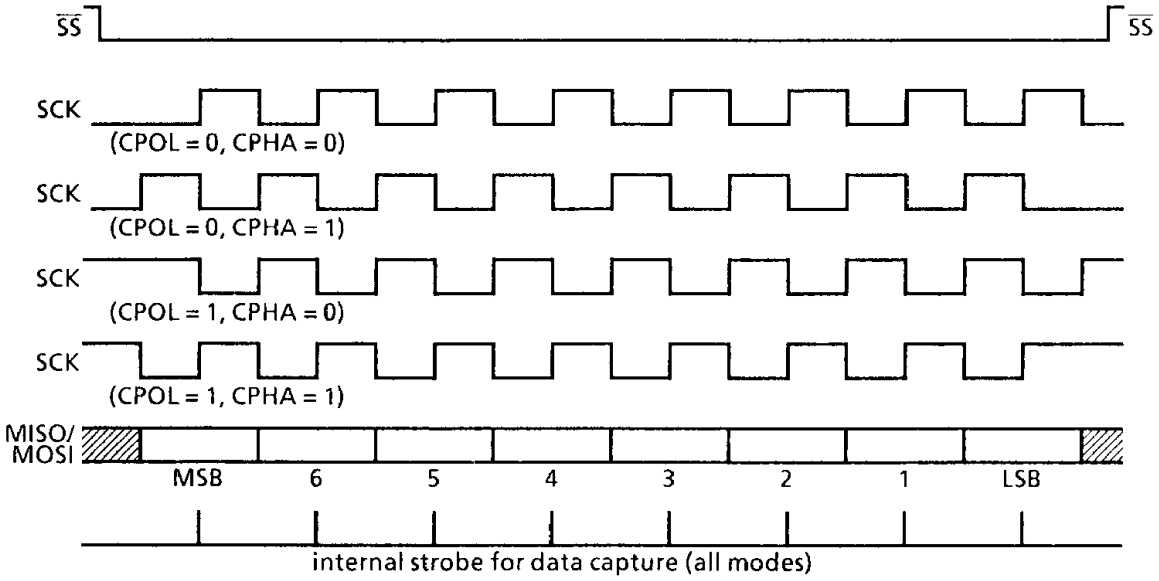


Figure 6.2 Data Clock Timing Diagram

6.2.2 Master In Slave Out (MISO)

The MISO pin is configured as an input in a master (mode) device and as an output in a slave (mode) device. In this manner data is transferred serially from a slave to a master on this line;most significant bit first, least significant bit last. The MISO pin of a slave device is placed in the high-impedance state if it is not selected by the master; i.e.,

its $\overline{SS}$ pin is a logic one. The timing diagram of Figure 6.2 shows the relationship between data and clock (SCK). As shown in Figure 6.2, four possible timing relationships may be chosen by using control bits CPOL and CPHA. The master device always allows data to be applied on the MOSI line a half-cycle before the clock edge (SCK) in order for the slave device to latch the data.

Note : The slave device(s) and a master device must be programmed to similar timing modes for proper data transfer.

When the master device transmits data to a slave device via the MOSI line, the slave device responds by sending data to the master device via the MISO line. This implies full duplex transmission with both data out and data in synchronized with the same clock signal (one which is provided by the master device). Thus, the byte transmitted is replaced by the byte received and eliminates the need for separate transmit-empty and receiver-full status bits. A single status bit (SPIF) in the serial peripheral status register (SPSR, location $0B) is used to signify that the I/O operation is complete.

In the master device, the MSTR control bit in the serial peripheral control register (SPCR, location $0A) is set to a logic one (by the program) to allow the master device to receive data on its MISO pin. In the slave device, its MISO pin is enabled by the logic level of the $\overline{SS}$ pin;i.e., if $\overline{SS}$ = 1 then the MISO pin is placed in the high-impedance state, whereas, if $\overline{SS}$ = 0 the MISO pin is an output for the slave device.

6.2.3 Slave Select ($\overline{SS}$)

The slave select ($\overline{SS}$) pin is a fixed input (PD5, pin 34), which receives an active low signal that is generated by the master device to enable slave device(s) to accept data. To ensure that data will be accepted by a slave device, the $\overline{SS}$ signal line must be a logic low prior to occurrence of SCK (system clock) and must remain low until after the last (eighth) SCK cycle. Figure 6.2 illustrates the relationship between SCK and the data for two different level combinations of CPHA, when $\overline{SS}$ is pulled low. These are: 1) with CPHA = 1 or 0, the first bit of data is applied to the MISO line for transfer, and 2) when CPHA = 0 the slave device is prevented from writing to its data register. Refer to the WCOL status flag in the serial peripheral status register (location $0B) description for further information on the effects that the $\overline{SS}$ input and CPHA control bit have on the I/O data register. A high level $\overline{SS}$ signal forces the MISO (master in slave out) line to the high-impedance state. Also, SCK and the MOSI (master out slave in) line are ignored by a slave device when its $\overline{SS}$ signal is high.

When a device is a master, it constantly monitors its $\overline{SS}$ signal input for a logic low. The master device will become a slave device any time its $\overline{SS}$ signal input is detected low. This ensures that there is only one master controlling the $\overline{SS}$ line for a particular system. When the $\overline{SS}$ line is detected low, it clears the MSTR control bit (serial peripheral control register, location $0A). Also, control bit SPE in the serial peripheral

control register is cleared which causes the serial peripheral interface (SPI) to be disabled (port D SPI pins become inputs). The MODF flag bit in the serial peripheral status register (location $0B) is also set to indicate to the master device that another device is attempting to become a master. Two devices attempting to be outputs are normally the result of a software error;however, a system could be configured which would contain a default master which would automatically "take-over" and restart the system.

6.2.4 Serial Clock (SCK)

The serial clock is used to synchronize the movement of data both in and out of the device through its MOSI and MISO pins. The master and slave devices are capable of exchanging a data byte of information during a sequence of eight clock pulses. Since the SCK is generated by the master device, the SCK line becomes an input on all slave devices and synchronizes slave data transfer. The type of clock and its relationship to data are controlled by the CPOL and CPHA bits in the serial peripheral control register (location $0A) discussed below. Refer to Figure 6.2 for timing.

The master device generates the SCK through a circuit driven by the internal processor clock. Two bits (SPR0 and SPR1) in the serial peripheral control register (location $0A) of the master device select the clock rate. The master device uses the SCK to latch incoming slave device data on the MISO line and shifts out data to the slave device on the MOSI line. Both master and slave devices must be operated in the same timing mode as controlled by the CPOL and CPHA bit in the serial peripheral control register. In the slave device, SPR0, SPR1 have no effect on the operation of the serial peripheral interface. Timing is shown in Figure 6.2.

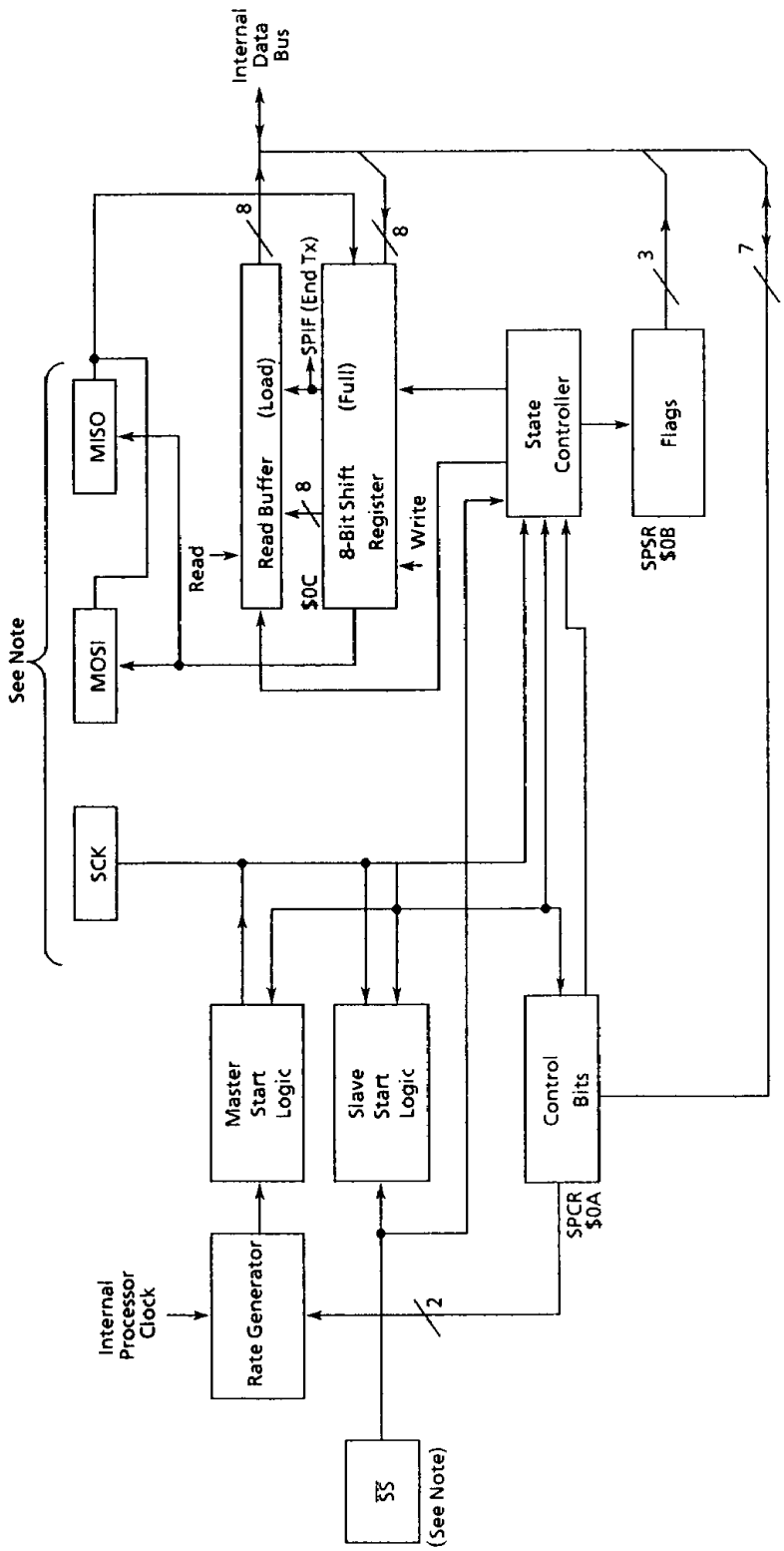
6.3 FUNCTIONAL DESCRIPTION

A block diagram of the serial peripheral interface (SPI) is shown in Figure 6.3. In a master configuration, the master start logic receives an input from the CPU (in the form of a write to the SPI rate generator) and originates the system clock (SCK) based on the internal processor clock. This clock is also used internally to control the state controller as well as the 8-bit shift register. As a master device, data is parallel loaded into the 8-bit shift register (from the internal bus) during a write cycle and then shifted out serially to the MOSI pin for application to the slave device(s). During a read cycle, data is applied serially from a slave device via the MISO pin to the 8-bit shift register. After the 8-bit shift register is loaded, its data is parallel transferred to the read buffer and then is made available to the internal data bus during a CPU read cycle.

In a slave configuration, the slave start logic receives a logic low (from a master device) at the $\overline{SS}$ pin and a system clock input (from the same master device) at the SCK pin. Thus, the slave is synchronized with the master. Data from the master is received serially at the slave MOSI pin and loads the 8-bit shift register. After the 8-bit shift

register is loaded, its data is parallel transferred to the read buffer and then is made available to the internal data bus during a CPU read cycle. During a write cycle, data is parallel loaded into the 8-bit register from the internal data bus and then shifted out serially to the MISO pin for application to the master device.

Figure 6.4 illustrates the MOSI, MISO, and SCK master-slave interconnections. Note that in Figure 6.4 the master $\overline{SS}$ pin is tied to a logic high and the slave $\overline{SS}$ pin is a logic low. Figure 6.1 provides a larger system connection for these same pins. Note that in Figure 6.1, all $\overline{SS}$ pins are connected to a port pin of a master/slave device. In this case any of the devices can be a slave.



Note : The $\overline{SS}$, SCK, MOSI, and MISO are external pins which provide the following functions:

a. MOSI – Provides serial output to slave unit(s) when device is configured as a master. Receives serial input from master unit when device is configured as a slave unit.

b. MISO – Receives serial input from slave unit(s) when device is configured as a master. Provides serial output to master when device is configured as a slave unit.

c. SCK – Provides system clock when device is configured as a master unit. Receives system clock when device is configured as a slave unit.

d. $\overline{SS}$ – Provides a logic low to select a slave device for a transfer with a master device.

Figure 6.3 Serial Peripheral Interface Block Diagram

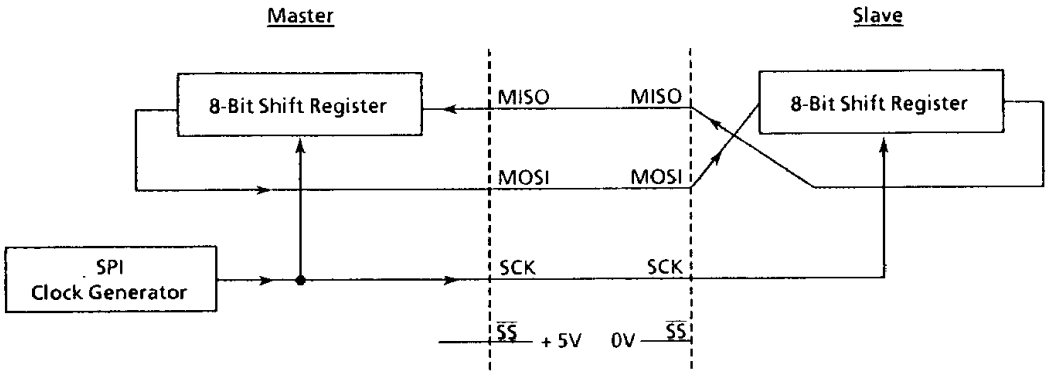


Figure 6.4 Serial Peripheral Interface Master-Slave Interconnection

6.4 REGISTERS

There are three registers in the serial parallel interface which provide control, status, and data storage functions. These registers which include the serial peripheral control register (SPCR, location $0A), serial peripheral status register (SPSR, location $0B), and serial peripheral data I/O register (SPDR, location $0C) are described below.

6.4.1 Serial Peripheral Control Register (SPCR)

7	6	5	4	3	2	1	0	
SPIE	SPE	-	MSTR	CPOL	CPHA	SPR1	SPR0	$0A

The serial peripheral control register bits are defined as follows:

B7, SPIE — When the serial peripheral interrupt enable bit is high, it allows the occurrence of a processor interrupt, and forces the proper vector to be loaded into the program counter if the serial peripheral status register flag bit (SPIF and/or MODF) is set to a logic one. It does not inhibit the setting of a status bit. The SPIE bit is cleared by reset.

B6, SPE — When the serial peripheral output enable control bit is set, all output drive is applied to the external pins and the system is enabled. When the SPE bit is set, it enables the SPI system by connecting it to the external pins thus allowing it to interface with the external SPI bus. The pins that are defined as output depend on which mode (master or slave) the device is in. Because the SPE bit is cleared by reset, the SPI system is not connected to the external pins upon reset.

B4, MSTR — The master bit determines whether the device is a master or a slave. If the MSTR bit is a logic zero it indicates a slave device and a logic one denotes a master device. If the master mode is selected, the function of the SCK pin changes from an input to an output and the function of the MISO

and MOSI pins are reversed. This allows the user to wire device pins MISO to MISO, and MOSI to MOSI, and SCK to SCK without incident. The MSTR bit is cleared by reset;therefore, the device is always placed in the slave mode during reset.

B3, CPOL The clock polarity bit controls the normal or steady state value of the clock when data is not being transferred. The CPOL bit affects both the master and slave modes. It must be used in conjunction with the clock phase control bit (CPHA) to produce the wanted clock-data relationship between a master and a slave device. When the CPOL bit is a logic zero, it produces a steady state low value at the SCK pin of the master device. If the CPOL bit is a logic one, a high value is produced at the SCK pin of the master device when data is not being transferred. The CPOL bit is not affected by reset. Refer to Figure 6.2.

B2, CPHA The clock phase bit controls the relationship between the data on the MISO and MOSI pins and the clock produced or received at the SCK pin. This control has effect in both the master and slave modes. It must be used in conjunction with the clock polarity control bit (CPOL) to produce the wanted clock-data relation. The CPHA bit in general selects the clock edge which captures data and allows it to change states. It has its greatest impact on the first bit transmitted (MSB) in that it does or does not allow a clock transition before the first data capture edge. The CPHA bit is not affected by reset. Refer to Figure 6.2.

B1, SPR1
B0, SPR0 These two serial peripheral rate bits select one of four baud rates to be used as SCK if the device is a master;however they have no effect in the slave mode. The slave device is capable of shifting data in and out at a maximum rate which is equal to the CPU clock. A rate table is given below for the generation of the SCK from the master. The SPR1 and SPR0 bits are not affected by reset.

SPR1	SPR0	Internal Processor Clock Divide By
0	0	2
0	1	4
1	0	16
1	1	32

6.4.2 Serial Peripheral Status Register (SPSR)

7	6	5	4	3	2	1	0	
SPIF	WCOL	–	MODF	–	–	–	–	$0B

The status flags which generate a serial peripheral interface (SPI) interrupt may be blocked by the SPIE control bit in the serial peripheral control register. The WCOL bit does not cause an interrupt. The serial peripheral status register bits are defined as follows:

B7, SPIF The serial peripheral data transfer flag bit notifies the user that a data transfer between the device and an external device has been completed. With the completion of the data transfer, SPIF is set, and if SPIE is set, a serial peripheral interrupt (SPI) is generated. During the clock cycle that SPIF is being set, a copy of the received data byte in the shift register is moved to a buffer. When the data register is read, it is the buffer that is read. During an overrun condition, when the master device has sent several bytes of data and the slave device has not responded to the first SPIF, only the first byte sent is contained in the receive buffer and other bytes are losat.

The transfer of data is initiated by the master device writing its serial peripheral data register.

Clearing the SPIF bit is accomplished by a software sequence of accessing the serial peripheral status register while SPIF is set and followed by a write to or a read of the serial peripheral data register. While SPIF is set, all writes to the serial peripheral data register are inhibited until the serial peripheral status register is read. This occurs in the master device. In the slave device, SPIF can be cleared (using a similar sequence) during a second transmission;however, it must be cleared before the second SPIF in order to prevent an overrun condition. The SPIF bit is cleared by reset.

B6, WCOL The function of the write collision status bit is to notify the user that an attempt was made to write the serial peripheral data register while a data transfer was taking place with an external device. The transfer continues uninterrupted;therefore, a write will be unsuccessful. A "read collision" will never occur since the received data byte is placed in a buffer in which access is always synchronous with the MCU operation. If a "write collision" occurs, WCOL is set but no SPI interrupt is generated. The WCOL bit is a status flag only.

Clearing the WCOL bit is accomplished by a software sequence of accessing the serial peripheral status register while WCOL is set, followed by 1) a read of the serial peripheral data register prior to the SPIF bit being set, or 2) a read or write of the serial peripheral data register after the SPIF bit is set. A write to the serial peripheral data register (SPDR) prior to the SPIF bit being set, will result in generation of another WCOL status flag. Both the SPIF and WCOL bits will be cleared in the same sequence. If a second transfer has started while trying to clear (the previously set) SPIF and WCOL bits with a clearing sequence containing a write to the serial peripheral data register, only the SPIF bit will be cleared.

A collision of a write to the serial peripheral data register while an external data transfer is taking place can occur in both the master mode and the slave mode, although with proper programming the master device should have sufficient information to preclude this collision.

Collision in the master device is defined as a write of the serial peripheral data register while the internal rate clock (SCK) is in the process of transfer. The signal on the $\overline{SS}$ pin is always high on the master device.

A collision in a slave device is defined in two separate modes. One problem arises in a slave device when the CPHA control bit is a logic zero. When CPHA is a logic zero, data is latched with the occurrence of the first clock transition. The slave device does not have any way of knowing when that transition will occur; therefore, the slave device collision occurs when it attempts to write the serial peripheral data register after its $\overline{SS}$ pin has been pulled low. The $\overline{SS}$ pin of the slave device freezes the data in its serial peripheral data register and does not allow it to be altered if the CPHA bit is a logic zero. The master device must raise the $\overline{SS}$ pin of the slave device high between each byte it transfers to the slave device.

The second collision mode is defined for the state of the CPHA control bit being a logic one. With the CPHA bit set, the slave device will be receiving a clock (SCK) edge prior to the latch of the first data transfer. This first clock edge will freeze the data in the slave device I/O register and allow the msb onto the external MISO pin of the slave device. The $\overline{SS}$ pin low state enables the slave device but the drive onto the MISO pin does not take place until the first data transfer clock edge. The WCOL bit will only be set if the I/O register is accessed while a transfer is taking place. By definition of the second collision mode, A master device might hold a

slave device $\overline{SS}$ pin low during a transfer of several bytes of data without a problem.

A special case of WCOL occurs in the slave device. This happens when the master device starts a transfer sequence (an edge or SCK for CPHA = 1;or an active $\overline{SS}$ transition for CPHA = 0) at the same time the slave device CPU is writing to its serial peripheral interface data register. In this case it is assumed that the data byte written (in the slave device serial peripheral interface) is lost and the contents of the slave device read buffer becomes the byte that is transferred. Because the master device receives back the last byte transmitted, the master device can detect that a fatal WCOL occurred.

Since the slave device is operating asynchronously with the master device, the WCOL bit may be used as an indicator of a collision occurrence. This helps alleviate the user from a strict real-time programming effort. The WCOL bit is cleaed by reset.

B4, MODF The function of the mode fault flag is defined for the master mode (device). If the device is a slave device the MODF bit will be prevented from toggling from a logic zero to a logic one;however, this does not prevent the device from being in the slave mode with the MODF bit set. The MODF bit is normally a logic zero and is set only when the master device has its $\overline{SS}$ pin pulled low. Toggling the MODF bit to a logic one affects the internal serial peripheral interface (SPI) system in the following ways:

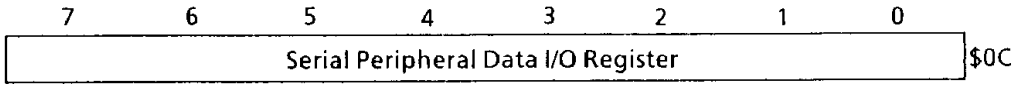
(1) MODF is set and SPI interrupt is generated if SPIE = 1.

(2) The SPE bit is forced to a logic zero. This blocks all output drive from the device, disables the SPI system.

(3) The MSTR bit is forced to a logic zero, thus forcing the device into the slave mode.

Clearing the MODF is accomplished by a software sequence of accessing the serial peripheral status register while MODF is set followed by a write to the serial peripheral control register. Control bits SPE and MSTR may be restored to their original set state during this clearing sequence or after the MODF bit has been cleared. Hardware does not allow the user to set the SPE and MSTR bit while MODF is a logic one unless it is during the proper clearing sequence. The MODF flag bit indicates that there might have been a multi-master conflict for system control and allows a proper exit from system operation to a reset or default system state. The MODF bit is cleated by reset.

6.4.3 Serial Peripheral Data I/O Register (SPDR)



The serial peripheral data I/O register (SPDR) is used to transmit and receive data on the serial bus. Only a write to this register will initiate transmission/reception of another byte and this will only occur in the master device. A slave device writing to its data I/O register will not initiate a transmission. At the completion of transmitting a byte of data, the SPIF status bit is set in both the master and slave devices. A write or read of the serial peripheral data I/O register, after accessing the serial peripheral status register with SPIF set, will clear SPIF.

During the clock cycle that the SPIF bit is being set, a copy of the received data byte in the shift register is being moved to a buffer. When the user reads the serial peripheral data I/O register, the buffer is actually being read. During an overrun condition, when the master device has sent several bytes of data and the slave device has not internally responded to clear the first SPIF, only the first byte is contained in the receive buffer of the slave device;all others are lost. The user may read the buffer at any time. The first SPIF must be cleared by the time a second transfer of data from the shift register to the read buffer is initiated or an overrun condition will exist.

A write to the serial peripheral data I/O register is not buffered and places data directly into the shift register for transmission.

The ability to access the serial peripheral data I/O register is limited when a transmission is taking place. It is important to read the discussion defining the WCOL and SPIF status bits to understand the limits on using the serial peripheral data I/O register.

6.5 SERIAL PERIPHERAL INTERFACE (SPI) SYSTEM CONSIDERATIONS

There are two types of SPI systems;single master system and multi-master systems. Figure 6.1 illustrates both of these systems and a discussion of each is provided below.

Figure 6.1 a illustrates how a typical single master system may be configured, using an M6805 HCMOS family device as the master and four M6805 HCMOS family devices as slaves. As shown, the MOSI, MISO, and SCK pins are all wired to equivalent pins on each of the five devices. The master device generates the SCK clock, the slave devices all receive it. Since the M6805 HCMOS master device is the bus master, it internally controls the function of its MOSI and MISO lines, thus writing data to the slave devices on the MOSI and reading data from the slave devices on the MISO lines. The master device selects the individual slave devices by using four pins of a parallel port to control the four $\overline{SS}$ pins of the slave devices. A slave device is selected when the master device

pulls its $\overline{SS}$ pin low. The $\overline{SS}$ pins are pulled high during reset since the master device ports will be forced to be inputs at that time, thus disabling the slave devices. Note that the slave devices do not have to be enabled in a mutually exclusive fashion except to prevent bus contention on the MISO line. For example, three slave devices, enabled for a transfer, are permissible if only one has the capability of being read by the master. An example of this is a write to several display drivers to clear a display with a single I/O operation. To ensure that proper data transmission is occurring between the master device and a slave device, the master device may have the slave device respond with a previously received data byte (this data byte could be inverted or at least be a byte that is different from the last one sent by the master device). The master device will always receive the previous byte back from the slave device if all MISO and MOSI lines are connected and the slave has not written its data I/O register. Other transmission security methods might be defined using ports for handshake lines or data bytes with command fields.

A multi-master system may also be configured by the user. A system of this type is shown in Figure 6.1b. An exchange of master control could be implemented using a handshake method through the I/O ports or by an exchange of code messages through the serial peripheral interface system. The major device control that plays a part in this system is the MSTR bit in the serial peripheral control register and the MODF bit in the serial peripheral status register.

7. EFFECTS OF STOP AND WAIT MODES ON THE TIMER AND SERIAL SYSTEMS

7.1 INTRODUCTION

The STOP and WAIT instructions have different effects on the programmable timer, serial communications interface (SCI), and serial peripheral interface (SPI) systems. These different effects are discussed separately below.

7.2 STOP MODE

When the processor executes the STOP instruction, the internal oscillator is turned off. This halts all internal CPU processing including the operation of the programmable timer, serial communications interface, and serial peripheral interface. The only way for the MCU to "wake up" from the stop mode is by receipt of an external interrupt (logic low on $\overline{IRQ}$ pin) or by the detection of a reset (logic low on $\overline{RESET}$ pin or a power-on reset). The effects of the stop mode on each of the MCU systems (Timer, SCI, and SPI) are described separately.

7.2.1 Timer During Stop Mode

When the MCU enters the stop mode, the timer counter stops counting (the internal processor clock is stopped) and remains at that particular count value until the stop mode is exited by an interrupt (if exited by reset the counter is forced to $FFFC). If the stop mode is exited by an external low on the $\overline{IRQ}$ pin, then the counter resumes from its stopped value as if nothing had happened. Another feature of the programmable timer, in the stop mode, is that if at least one valid input capture edge occurs at the TCAP pin, the input capture detect circuitry is armed. This action does not set any timer flags or "wake up" the MCU, but when the MCU does "wake up" there will be an active input capture flag (and data) from that first valid edge which occurred during the stop mode. If the stop mode is exited by an external reset (logic low on $\overline{RESET}$ pin), then no such input capture flag or data action takes place even if there was a valid input capture edge (at the TCAP pin) during the MCU stop mode.

7.2.2 SCI During Stop Mode

When the MCU enters the stop mode, the baud rate generator which derives the receiver and transmitter is shut down. This essentially stops all SCI activity. The receiver is unable to receive and transmitter is unable to transmit. If the STOP instruction is executed during a transmitter transfer, that transfer is halted. When the stop mode is exited, that particular transmission resumes (if the exit is the result of a low input to the $\overline{IRQ}$ pin). Since the previous transmission resumes after an $\overline{IRQ}$ interrupt stop mode exit, the user should ensure that the SCI transmitter is in the idle state when the STOP instruction is executed. If the receiver is receiving data when the STOP instruction is executed, received data sampling is stopped (baud rate generator

stops) and the rest of the data is lost. For the above reasons, all SCI transactions should be in the idle state when the STOP instruction is executed.

7.2.3 SPI During Stop Mode

When the MCU enters the stop mode, the baud rate generator which drives the SPI shuts down. This essentially stops all master mode SPI operation, thus the master SPI is unable to transmit or receive any data. If the STOP instruction is executed during an SPI transfer, that transfer is halted until the MCU exits the stop mode (provided it is an exit resulting from a logic low on the $\overline{IRQ}$ pin). If the stop mode is exited by a reset, then the appropriate control/status bits are cleared and the SPI is disabled. If the device is in the slave mode when the STOP instruction is executed, the slave SPI will still operate. It can still accept data and clock information in addition to transmitting its own data back to a master device.

At the end of a possible transmission with a slave SPI in the stop mode, no flags are set until a logic low $\overline{IRQ}$ input results in an MCU "wake up". Caution should be observed when operating the SPI (as a slave) during the stop mode because none of the protection circuity (write collision, mode fault, etc.) is active.

It should also be noted that when the MCU enters the stop mode all enabled output drivers (TDO, TCMP, MISO, MOSI, and SCK ports) remain active and any sourcing currents from these outputs will be part of the total supply current required by the device.

7.3 WAIT MODE

When the MCU enters the wait mode, the CPU clock is halted. All CPU action is suspended;however, the timer, SCI, and SPI systems remain active. In fact an interrupt from the timer, SCI, or SPI (in addition to a logic low on the $\overline{IRQ}$ or $\overline{RESET}$ pins) causes the processor to exit the wait mode. Since the three systems mentioned above operate as they do in the normal mode, only a general discussion of the wait mode is provided below.

The wait mode power consumption depends on how many systems are active. The power consumption will be highest when all the systems (timer, TCMP, SCI, and SPI) are active. The power consumption will be the least when the SCI and SPI systems are disabled (timer operation cannot be disabled in the wait mode). If a non-reset exit from the wait mode is performed (e.g., timer overflow interrupt exit), the state of the remaining systems will be unchanged. If a reset exit from the wait mode is performed all the systems revert to the disabled reset state.

8. INSTRUCTION SET AND ADDRESSING MODES

8.1 INSTRUCTION SET

The MCU has a set of 62 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. All the instructions within a given type are presented in individual tables.

All of the instructions used in the M146805 CMOS Family are used in the TMP 68HC05C4MCU, plus an additional one;the multiply (MUL) instruction. This instruction allows for unsigned multiplication of the contents of the accumulator (A) and the index register (X). The high order product is then stored in the index register and the low order product is stored in the accumulator. A detailed definition of the MUL instruction is shown below.

Operation: X:A←X*A

Description: Multiplies the eight bits in the index register by the eight bits in the accumulator to obtain a 16-bit unsigned number in the concatenated accumulator and index register.

Condition Codes:
H : Cleared
I : Not affected
N : Not affected
Z : Not affected
C : Cleared

Source Form(s): MUL

Addressing Mode	Cycles	Bytes	Opcode
Inherent	11	1	$42

8.1.1 Register/Memory Instructions

Most of these instructions use two operands. The first operand is either the accumulator or the index register. The second operand is obtained from memory using one of the addressing modes. The operand for the jump unconditional (JMP) and jump to subroutine (JSR) instructions is the program counter. Refer to Table 8.1.

8.1.2 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. The test for negative or zero (TST) instruction is an exception to the read-modify-write sequence since it does not modify the value. Refer to Table 8.2.

Table 8.1 Register/Memory Instructions

Function	Mnem.	Addressing Modes: Immediate Op Code	Immediate # Bytes	Immediate # Cycles	Direct Op Code	Direct # Bytes	Direct # Cycles	Extended Op Code	Extended # Bytes	Extended # Cycles	Indexed (No Offset) Op Code	Indexed (No Offset) # Bytes	Indexed (No Offset) # Cycles	Indexed (8-Bit Offset) Op Code	Indexed (8-Bit Offset) # Bytes	Indexed (8-Bit Offset) # Cycles	Indexed (16-Bit Offset) Op Code	Indexed (16-Bit Offset) # Bytes	Indexed (16-Bit Offset) # Cycles
Load A from Memory	LDA	A6	2	2	B6	2	3	C6	3	4	F6	1	3	E6	2	4	D6	3	5
Load X from Memory	LDX	AE	2	2	BE	2	3	CE	3	4	FE	1	3	EE	2	4	DE	3	5
Store A in Memory	STA	–	–	–	B7	2	4	C7	3	5	F7	1	4	E7	2	5	D7	3	6
Store X in Memory	STX	–	–	–	BF	2	4	CF	3	5	FF	1	4	EF	2	5	DF	3	6
Add Memory to A	ADD	AB	2	2	BB	2	3	CB	3	4	FB	1	3	EB	2	4	DB	3	5
Add Memory and Carry to A	ADC	A9	2	2	B9	2	3	C9	3	4	F9	1	3	E9	2	4	D9	3	5
Subtract Memory	SUB	A0	2	2	B0	2	3	C0	3	4	F0	1	3	E0	2	4	D0	3	5
Subtract Memory from A with Borrow	SBC	A2	2	2	B2	2	3	C2	3	4	F2	1	3	E2	2	4	D2	3	5
AND Memory to A	AND	A4	2	2	B4	2	3	C4	3	4	F4	1	3	E4	2	4	D4	3	5
OR Memory with A	ORA	AA	2	2	BA	2	3	CA	3	4	FA	1	3	EA	2	4	DA	3	5
Exclusive OR Memory with A	EOR	A8	2	2	B8	2	3	C8	3	4	F8	1	3	E8	2	4	D8	3	5
Arithmetic Compare A with Memory	CMP	A1	2	2	B1	2	3	C1	3	4	F1	1	3	E1	2	4	D1	3	5
Arithmetic Compare X with Memory	CPX	A3	2	2	B3	2	3	C3	3	4	F3	1	3	E3	2	4	D3	3	5
Bit Test Memory with A (Logical Compare)	BIT	A5	2	2	B5	2	3	C5	3	4	F5	1	3	E5	2	4	D5	3	5
Jump Unconditional	JMP	–	–	–	BC	2	2	CC	3	3	FC	1	2	EC	2	3	DC	3	4
Jump to Subroutine	JSR	–	–	–	BD	2	5	CD	3	6	FD	1	5	ED	2	6	DD	3	7

Table 8.2 Read-Modify-Write Instructions

Function	Mnemonic	Addressing Modes: Inherent (A) Op Code	# Bytes	# Cycles	Inherent (X) Op Code	# Bytes	# Cycles	Direct Op Code	# Bytes	# Cycles	Indexed (No Offset) Op Code	# Bytes	# Cycles	Indexed (8-Bit Offset) Op Code	# Bytes	# Cycles
Increment	INC	4C	1	3	5C	1	3	3C	2	5	7C	1	5	6C	2	6
Decrement	DEC	4A	1	3	5A	1	3	3A	2	5	7A	1	5	6A	2	6
Clear	CLR	4F	1	3	5F	1	3	3F	2	5	7F	1	5	6F	2	6
Complement	COM	43	1	3	53	1	3	33	2	5	73	1	5	63	2	6
Nagate (2's Complement)	NEG	40	1	3	50	1	3	30	2	5	70	1	5	60	2	6
Rotate Left Thru Carry	ROL	49	1	3	59	1	3	39	2	5	79	1	5	69	2	6
Rotate Right Thru Carry	ROR	46	1	3	56	1	3	36	2	5	76	1	5	66	2	6
Logical Shift Left	LSL	48	1	3	58	1	3	38	2	5	78	1	5	68	2	6
Logical Shift Right	LSR	44	1	3	54	1	3	34	2	5	74	1	5	64	2	6
Arithmetic Shift Right	ASR	47	1	3	57	1	3	37	2	5	77	1	5	67	2	6
Test for Negative or Zero	TST	4D	1	3	5D	1	3	3D	2	4	7D	1	4	6D	2	5
Multiply	MUL	42	1	11	–	–	–	–	–	–	–	–	–	–	–	–

8.1.3 Branch Instructions

Most branch instruction test the state of the condition code register and if certain criteria are met, a branch is executed. This adds an offset between −127 and +128 to the current program counter. Refer to Table 8.3.

Table 8.3 Branch Instructions

Function	Reiative Addressing Mode			
	Mnemonic	Op Code	# Bytes	# Cycles
Branch Always	BRA	20	2	3
Branch Never	BRN	21	2	3
Branch IFF Higher	BHI	22	2	3
Branch IFF Lower or Same	BLS	23	2	3
Branch IFF Carry Clear	BCC	24	2	3
(Branch IFF Higher or Same)	(BHS)	24	2	3
Branch IFF Carry Set	BCS	25	2	3
(Branch IFF Lower)	(BLO)	25	2	3
Branch IFF Not Equal	BNE	26	2	3
Branch IFF Equal	BEQ	27	2	3
Branch IFF Half Carry Clear	BHCC	28	2	3
Branch IFF Half Carry Set	BHCS	29	2	3
Branch IFF Plus	BPL	2A	2	3
Branch IFF Minus	BMI	2B	2	3
Branch IFF Interrupt Mask Bit is Clear	BMC	2C	2	3
Branch IFF Interrupt Mask Bit is Set	BMS	2D	2	3
Branch IFF Interrupt Line is Low	BIL	2E	2	3
Branch IFF Interrupt Line is High	BIH	2F	2	3
Branch to Subroutine	BSR	AD	2	6

8.1.4 Bit Manipulation Instructions

The MCU is capable of setting or clearing any bit which resides in the first 256 bytes of the memory space except for ROM, port D data location ($03), serial peripheral status register ($0B), serial communications status register ($10), timer status register ($13), and timer input capture register ($14-$15). All port register, port DDRs, timer, two serial systems, on-chip RAM, and 48 bytes of ROM reside in the first 256 bytes (page zero). An additional feature allows the software to test and branch on the state of any bit within the first 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 automatically placed in the carry bit of the condition code register, Refer to Table 8.4.

Table 8.4 Bit Manipulation Instructions

Function	Mnemonic	Addressing Modes					
		Bit Set/Clear			Bit Test and Branch		
		Op Code	# Bytes	# Cycles	Op Code	# Bytes	# Cycles
Branch IFF Bit n is Set	BRSET n (n = 0...7)	–	–	–	2•n	3	5
Branch IFF Bit n is Clear	BRCLR n (n = 0...7)	–	–	–	01 + 2•n	3	5
Set Bit n	BSET n (n = 0...7)	10 + 2•n	2	5	–	–	–
Clear Bit n	BCLR n (n = 0...7)	11 + 2•n	2	5	–	–	–

8.1.5 Control Instructions

These instructions are register reference instructions and are used to control processor operation during program execution. Refer to Table 8.5.

Table 8.5 Control Instructions

Function	Mnemonic	Inherent		
		Op Code	# Bytes	# Cycles
Transfer A to X	TAX	97	1	2
Transfer X to A	TXA	9F	1	2
Set Carry Bit	SEC	99	1	2
Clear Carry Bit	CLC	98	1	2
Set Interrupt Mask Bit	SEI	9B	1	2
Clear Interrupt Mask Bit	CLI	9A	1	2
Software Interrupt	SWI	83	1	10
Return from Subroutine	RTS	81	1	6
Return from Interrupt	RTI	80	1	9
Reset Stack Pointer	RSP	9C	1	2
No-Operation	NOP	9D	1	2
Stop	STOP	8E	1	2
Wait	WAIT	8F	1	2

8.1.6 Alphabetical Listing

The complete instruction set is given in alphabetical order in Table 8.6.

8.1.7 Opcode Map

Table 8.7 is an opcode map for the instructions used on the MCU.

8.2 ADDRESSING MODES

The MCU uses ten different addressing modes to provide the programmer with an opportunity to optimize the code to all situations. The various indexed addressing modes

make it possible to locate data tables, code conversion tables, and scaling tables anywhere in the memory space. Short indexed accesses are single byte instructions, while the longest instructions (three bytes) permit accessing tables throughout memory. Short absolute (direct) and long absolute (extended) addressing are also included. One and two byte direct addressing instructions access all data bytes in most applications. Extended addressing permits jump instructions to reach all memory. Table 8.7 shows the addressing modes for each instruction, with the effects each instruction has on the condition code register.

The term "effective address" (EA) is used in describing the various addressing modes, and is defined as the byte address to or from which the argument for an instruction is fetched or stored. The ten addressing modes of the processor are described below. Parentheses are used to indicate "contents of " the location or register referred to;e.g., (PC) indicates the contents of the location pointed to by the PC. An arrow indicates "is replaced by", and a colon indicates concatenation of two bytes. For additional details and graphical illustrations, refer to the M6805 HMOS Family Microcomputer/ Microprocessor User's Manual.

Table 8.6 Instruction Set (1/2)

Mnemonic	addressing Modes										Condition Codes				
	Inherent	Immediate	Direct	Extended	Relative	Indexed (No Offset)	Indexed (8 Bits)	Indexed (16 Bits)	Bit Set/Clear	Bit Test & Branch	H	I	N	Z	C
ADC		X	X	X		X	X	X			Λ	●	Λ	Λ	Λ
ADD		X	X	X		X	X	X			Λ	●	Λ	Λ	Λ
AND		X	X	X		X	X	X			●	●	Λ	Λ	●
ASL	X		X			X	X				●	●	Λ	Λ	Λ
ASR	X		X			X	X				●	●	Λ	Λ	Λ
BCC					X						●	●	●	●	●
BCLR									X		●	●	●	●	●
BCS					X						●	●	●	●	●
BEQ					X						●	●	●	●	●
BHCC					X						●	●	●	●	●
BHCS					X						●	●	●	●	●
BHI					X						●	●	●	●	●
BHS					X						●	●	●	●	●
BIH					X						●	●	●	●	●
BIL					X						●	●	●	●	●
BIT		X	X	X		X	X	X			●	●	Λ	Λ	●
BLO					X						●	●	●	●	●
BLS					X						●	●	●	●	●
BMC					X						●	●	●	●	●
BMI					X						●	●	●	●	●
BMS					X						●	●	●	●	●
BNE					X						●	●	●	●	●
BPL					X						●	●	●	●	●
BRA					X						●	●	●	●	●
BRN					X						●	●	●	●	●
BRCLR										X	●	●	●	●	Λ
BRSET										X	●	●	●	●	Λ
BSET									X		●	●	●	●	●
BSR					X						●	●	●	●	●
CLC	X										●	●	●	●	0
CLI	X										●	0	●	●	●
CLR	X		X			X	X				●	●	0	1	●
CMP		X	X	X		X	X	X			●	●	Λ	Λ	Λ
COM	X		X			X	X				●	●	Λ	Λ	1
CPX		X	X	X		X	X	X			●	●	Λ	Λ	Λ
DEC	X		X			X	X				●	●	Λ	Λ	●
EOR		X	X	X		X	X	X			●	●	Λ	Λ	●
INC	X		X			X	X				●	●	Λ	Λ	●
JMP			X	X		X	X	X			●	●	●	●	●
JSR			X	X		X	X	X			●	●	●	●	●

Condition Code Symbols:

H	Half Carry (From Bit 3)	Λ	Test and Set if True Cleared Otherwise
I	Interrupt Mask	●	Not Affected
N	Negate (Sign Bit)	?	Load CC Register From Stack
Z	Zero	0	Cleared
C	Carry/Borrow	1	Set

Table 8.6 Instruction Set (2/2)

Mnemonic	addressing Modes										Condition Codes				
	Inherent	Immediate	Direct	Extended	Relative	Indexed (No Offset)	Indexed (8 Bits)	Indexed (16 Bits)	Bit Set/Clear	Bit Test & Branch	H	I	N	Z	C
LDA		X	X	X		X	X	X			●	●	Λ	Λ	●
LDX		X	X	X		X	X	X			●	●	Λ	Λ	●
LSL	X		X			X	X				●	●	Λ	Λ	Λ
LSR	X		X			X	X				●	●	0	Λ	Λ
MUL	X										0	●	●	●	0
NEG	X		X			X	X				●	●	Λ	Λ	Λ
NOP	X										●	●	●	●	●
ORA		X	X	X		X	X	X			●	●	Λ	Λ	●
ROL	X		X			X	X				●	●	Λ	Λ	Λ
ROR	X		X			X	X				●	●	Λ	Λ	Λ
RSP	X										●	●	●	●	●
RTI	X										?	?	?	?	?
RTS	X										●	●	●	●	●
SBC		X	X	X		X	X	X			●	●	Λ	Λ	Λ
SEC	X										●	●	●	●	1
SEI	X										●	1	●	●	●
STA			X	X		X	X	X			●	●	Λ	Λ	●
STOP	X										●	0	●	●	●
STX			X	X		X	X	X			●	●	Λ	Λ	●
SUB		X	X	X		X	X	X			●	●	Λ	Λ	Λ
SWI	X										●	1	●	●	●
TAX	X										●	●	●	●	●
TST	X		X			X	X				●	●	Λ	Λ	●
TXA	X										●	●	●	●	●
WAIT	X										●	0	●	●	●

Condition Code Symbols:

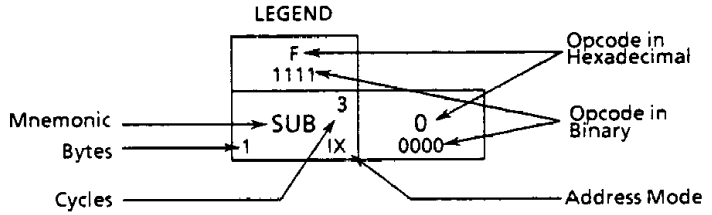
H	Half Carry (From Bit 3)	Λ	Test and Set if True Cleared Otherwise
I	Interrupt Mask	●	Not Affected
N	Negate (Sign Bit)	?	Load CC Register From Stack
Z	Zero	0	Cleared
C	Carry/Borrow	1	Set

Table 8.7 MC68HC05C4 HCMOS Instruction Set Opcode Map

	Bit Manipulation		Branch	Read/Modify/Write					Control		Register/Memory						
	BTB	BSC	REL	DIR	INH	INH	IX1	IX	INH	INH	IMM	DIR	EXT	IX2	IX1	IX	
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	BRSET0 5 3 BTB	BSET0 5 2 BSC	BRA 3 2 REL	NEG 5 2 DIR	NEGA 3 1 INH	NEGX 3 1 INH	NEG 6 2 IX1	NEG 5 1 IX	RTI 9 1 INH		SUB 2 2 IMM	SUB 3 2 DIR	SUB 4 3 EXT	SUB 5 3 IX2	SUB 4 2 IX1	SUB 3 1 IX	0 0000
1 0001	BRCLR0 5 3 BTB	BCLR0 5 2 BSC	BRN 3 2 REL						RTS 6 1 INH		CMP 2 2 IMM	CMP 3 2 DIR	CMP 4 3 EXT	CMP 5 3 IX2	CMP 4 2 IX1	CMP 3 1 IX	1 0001
2 0010	BRSET1 5 3 BTB	BSET1 5 2 BSC	BHI 3 2 REL		MUL 11 1 INH						SBC 2 2 IMM	SBC 3 2 DIR	SBC 4 3 EXT	SBC 5 3 IX2	SBC 4 2 IX1	SBC 3 1 IX	2 0010
3 0011	BRCLR1 5 3 BTB	BCLR1 5 2 BSC	BLS 3 2 REL	COM 5 2 DIR	COMA 3 1 INH	COMX 3 1 INH	COM 6 2 IX1	COM 5 1 IX	SWI 10 1 INH		CPX 2 2 IMM	CPX 3 2 DIR	CPX 4 3 EXT	CPX 5 3 IX2	CPX 4 2 IX1	CPX 3 1 IX	3 0011
4 0100	BRSET2 5 3 BTB	BSET2 5 2 BSC	BCC 3 2 REL	LSR 5 2 DIR	LSRA 3 1 INH	LSRX 3 1 INH	LSR 6 2 IX1	LSR 5 1 IX			AND 2 2 IMM	AND 3 2 DIR	AND 4 3 EXT	AND 5 3 IX2	AND 4 2 IX1	AND 3 1 IX	4 0100
5 0101	BRCLR2 5 3 BTB	BCLR2 5 2 BSC	BCS 3 2 REL								BIT 2 2 IMM	BIT 3 2 DIR	BIT 4 3 EXT	BIT 5 3 IX2	BIT 4 2 IX1	BIT 3 1 IX	5 0101
6 0110	BRSET3 5 3 BTB	BSET3 5 2 BSC	BNE 3 2 REL	ROR 5 2 DIR	RORA 3 1 INH	RORX 3 1 INH	ROR 6 2 IX1	ROR 5 1 IX			LDA 2 2 IMM	LDA 3 2 DIR	LDA 4 3 EXT	LDA 5 3 IX2	LDA 4 2 IX1	LDA 3 1 IX	6 0110
7 0111	BRCLR3 5 3 BTB	BCLR3 5 2 BSC	BEQ 3 2 REL	ASR 5 2 DIR	ASRA 3 1 INH	ASRX 3 1 INH	ASR 6 2 IX1	ASR 5 1 IX		TAX 2 1 INH		STA 4 2 DIR	STA 5 3 EXT	STA 6 3 IX2	STA 5 2 IX1	STA 4 1 IX	7 0111
8 1000	BRSET4 5 3 BTB	BSET4 5 2 BSC	BHCC 3 2 REL	LSL 5 2 DIR	LSLA 3 1 INH	LSLX 3 1 INH	LSL 6 2 IX1	LSL 5 1 IX		CLC 2 1 INH	EOR 2 2 IMM	EOR 3 2 DIR	EOR 4 3 EXT	EOR 5 3 IX2	EOR 4 2 IX1	EOR 3 1 IX	8 1000
9 1001	BRCLR4 5 3 BTB	BCLR4 5 2 BSC	BHCS 3 2 REL	ROL 5 2 DIR	ROLA 3 1 INH	ROLX 3 1 INH	ROL 6 2 IX1	ROL 5 1 IX		SEC 2 1 INH	ADC 2 2 IMM	ADC 3 2 DIR	ADC 4 3 EXT	ADC 5 3 IX2	ADC 4 2 IX1	ADC 3 1 IX	9 1001
A 1010	BRSET5 5 3 BTB	BSET5 5 2 BSC	BPL 3 2 REL	DEC 5 2 DIR	DECA 3 1 INH	DECX 3 1 INH	DEC 6 2 IX1	DEC 5 1 IX		CLI 2 1 INH	ORA 2 2 IMM	ORA 3 2 DIR	ORA 4 3 EXT	ORA 5 3 IX2	ORA 4 2 IX1	ORA 3 1 IX	A 1010
B 1011	BRCLR5 5 3 BTB	BCLR5 5 2 BSC	BMI 3 2 REL							SEI 2 1 INH	ADD 2 2 IMM	ADD 3 2 DIR	ADD 4 3 EXT	ADD 5 3 IX2	ADD 4 2 IX1	ADD 3 1 IX	B 1011
C 1100	BRSET6 5 3 BTB	BSET6 5 2 BSC	BMC 3 2 REL	INC 5 2 DIR	INCA 3 1 INH	INCX 3 1 INH	INC 6 2 IX1	INC 5 1 IX		RSP 2 1 INH		JMP 2 2 DIR	JMP 3 3 EXT	JMP 4 3 IX2	JMP 3 2 IX1	JMP 2 1 IX	C 1100
D 1101	BRCLR6 5 3 BTB	BCLR6 5 2 BSC	BMS 3 2 REL	TST 4 2 DIR	TSTA 3 1 INH	TSTX 3 1 INH	TST 5 2 IX1	TST 4 1 IX		NOP 2 1 INH	BSR 6 2 REL	JSR 5 2 DIR	JSR 6 3 EXT	JSR 7 3 IX2	JSR 6 2 IX1	JSR 5 1 IX	D 1101
E 1110	BRSET7 5 3 BTB	BSET7 5 2 BSC	BIL 3 2 REL						STOP 2 1 INH		LDX 2 2 IMM	LDX 3 2 DIR	LDX 4 3 EXT	LDX 5 3 IX2	LDX 4 2 IX1	LDX 3 1 IX	E 1110
F 1111	BRCLR7 5 3 BTB	BCLR7 5 2 BSC	BIH 3 2 REL	CLR 5 2 DIR	CLRA 3 1 INH	CLRX 3 1 INH	CLR 6 2 IX1	CLR 5 1 IX	WAIT 2 1 INH	TXA 2 1 INH		STX 4 2 DIR	STX 5 3 EXT	STX 6 3 IX2	STX 5 2 IX1	STX 4 1 IX	F 1111

Abbreviations for Address Modes

INH	Inherent
A	Accumulator
X	Index Register
IMM	Immediate
DIR	Direct
EXT	Extended
REL	Relative
BSC	Bit Set/Clear
BTB	Bit Test and Branch
IX	Indexed (No Offset)
IX1	Indexed. 1 Byte (8-Bit) Offset
IX2	Indexed. 2 Byte (16-Bit) Offset



8.2.1 Inherent

In inherent instructions, all the information necessary to execute the instruction is contained in the opcode. Operations specifying only the index register or accumulator, and no other arguments, are included in this mode.

8.2.2 Immediate

In immediate addressing, the operand is contained in the byte immediately following the opcode. Immediate addressing is used to access constants which do not change during program execution (e.g., a constant used to initialize a loop counter).

$$EA = PC + 1; PC \leftarrow PC + 2$$

8.2.3 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 lowest 256 bytes in memory with a single two byte instruction. This includes all on-chip RAM and I/O registers, and 128 bytes of on-chip ROM. Direct addressing is efficient in both memory and time.

$$EA = (PC + 1); PC \leftarrow PC + 2$$

$$\text{Address Bus High} \leftarrow 0; \text{Address Bus Low} \leftarrow (PC + 1)$$

8.2.4 Extended

In the extended addressing mode, the effective address of the argument is contained in the two bytes following the opcode. Instructions with extended addressing modes are capable of referencing arguments anywhere in memory with a single three-byte instruction. When using the Motorola assembler, the user need not specify whether an instruction uses direct or extended addressing. The assembler automatically selects the most efficient addressing mode.

$$EA = (PC + 1):(PC + 2); PC \leftarrow PC + 3$$

$$\text{Address Bus High} \leftarrow (PC + 1); \text{Address Bus Low} \leftarrow (PC + 2)$$

8.2.5 Indexed, No Offset

In the indexed, no offset addressing mode, the effective address of the argument is contained in the 8-bit index register. Thus, this addressing mode can access the first 256 memory locations. These instructions are only one byte long. This mode is used to move a pointer through a table or to address a frequently referenced RAM or I/O location.

$$EA = X; PC \leftarrow PC + 1$$

$$\text{Address Bus High} \leftarrow 0; \text{Address Bus Low} \leftarrow X$$

8.2.6 Indexed, 8-Bit Offset

Here the EA is obtained by adding the contents of the byte following the opcode to that of the index register;therefore, the operand is located anywhere within the lowest 511 memory locations. For example, this mode of addressing is useful for selecting the mth element in an element table. All instructions are two bytes. The content of the index register (X) is not changed. The content of (PC + 1) is an unsigned 8-bit integer. One byte offset indexing permits look-up tables to be easily accessed in either RAM or ROM.

$$EA = X + (PC + 1); PC \leftarrow PC + 2$$

$$\text{Address Bus High} \leftarrow K; \text{Address Bus Low} \leftarrow X + (PC + 1)$$

where:

K = The carry from the addition of X + (PC + 1)

8.2.7 Indexed, 16-Bit Offset

In the indexed, 16-bit offset addressing mode, the effective address is the sum of the contents of the unsigned 8-bit index register and the two unsigned bytes following the opcode. This addressing mode can be used in a manner similar to indexed 8-bit offset, except that this three byte instruction allows tables to be anywhere in memory (e.g., jump tables in ROM). As with direct and extended, the M6805 assembler determines the most efficient form of indexed offset;8-or 16-bit. The content of the index register is not changed.

$$EA = X + [(PC + 1):(PC + 2))]; PC \leftarrow PC + 3$$

$$\text{Address Bus High} \leftarrow (PC + 1) + K;$$

$$\text{Address Bus Low} \leftarrow X + (PC + 2)$$

where:

K = The carry from the addition of X + (PC + 2)

8.2.8 Relative

Relative addressing is only used in branch instructions. In relative addressing, the content of the 8-bit signed byte following the opcode (the offset) is added to the PC if and only if the branch condition is true. Otherwise, control proceeds to the next instruction. The span of relative addressing is limited to the range of −126 to +129 bytes from the branch instruction opcode location. The Motorola assembler calculates the proper offset and checks to see if it is within the span of the branch.

$$EA = PC + 2 + (PC + 1); PC \leftarrow EA \text{ if branch taken;}$$

$$\text{otherwise, } EA = PC \leftarrow PC + 2$$

8.2.9 Bit Set/Clear

Direct addressing and bit addressing are combined in instructions which set and clear individual memory and I/O bits. In the bit set and clear instructions, the byte is specified as a direct address in the location following the opcode. The first 256 addressable locations are thus accessed. The bit to be modified within that byte is specified in the first three bits of the opcode. The bit set and clear instructions occupy two bytes, one for the opcode (including the bit number) and the other to address the byte which contains the bit of interest.

$$EA = (PC + 1); PC \leftarrow PC + 2$$

$$\text{Address Bus High} \leftarrow 0; \text{Address Bus Low} \leftarrow (PC + 1)$$

8.2.10 Bit Test and Branch

Bit test and branch is a combination of direct addressing, bit set/clear addressing, and relative addressing. The actual bit to be tested, within the byte, is specified within the low order nibble of the opcode. The address of the data byte to be tested is located via a direct address in the location following the opcode byte (EA1). The signed relative 8-bit offset is in the third byte (EA2) and is added to the PC if the specified bit is set or cleared in the specified memory location. This single three byte instruction allows the program to branch based on the condition of any bit in the first 256 locations of memory.

$$EA1 = (PC + 1)$$

$$\text{Address Bus High} \leftarrow 0; \text{Address Bus Low} \leftarrow (PC + 1)$$

$$EA2 = PC + 3 + (PC + 2); PC \leftarrow EA2 \text{ if branch taken;}$$

$$\text{otherwise, } PC \leftarrow PC + 3$$

9. ELECTRICAL SPECIFICATIONS

9.1 INTRODUCTION

This section contains the electrical specifications and associated timing information for the TMP68HC05C4.

9.2 MAXIMUM RATINGS

(Voltages Referenced to V_{SS})

Ratings	Symbol	Value	Unit
Supply Voltage	V_{DD}	− 0.3 to + 7.0	V
Input Voltage	Vin	V_{SS} − 0.3 to V_{DD} + 0.3	V
Self-Check Mode ($\overline{IRQ}$ Pin Only)	Vin	V_{SS} − 0.3 to $2 \times V_{DD}$ + 0.3	V
Current Drain Per Pin Excluding V_{DD} and V_{SS}	I	25	mA
Operating Temperature Range TMP68HC05C4 (Standard)	T_A	T_L to T_H 0 to + 70	℃
Storage Temperature Range	T_{stg}	− 65 to + 150	℃

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 application 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 $V_{SS} \leq (V_{in} \text{ or } V_{out}) \leq V_{DD}$. Reliability of operation is enhanced if unused inputs are connected to an appropriate logic voltage level (e.g., either V_{SS} or V_{DD}).

9.3 THERMAL CHARACTERISTICS

Characteristics	Symbol	Value	Unit
Thermal Resistance	θ_{JA}		℃/W
Ceramic		50	
Plastic		60	
Plastic Leaded Chip Carrier (PLCC)		70	

V_{DD} = 4.5V

Pins	R1	R2	C
PA0-PA7, PB0-PB7, PC0-PC7, PD6	3.26kΩ	2.38kΩ	50pF
PD1-PD4	1.9kΩ	2.26kΩ	200pF

V_{DD} = 3.0V

Pins	R1	R2	C
PA0-PA7, PB0-PB7, PC0-PC7, PD6	10.91kΩ	6.32kΩ	50pF
PD1-PD4	6kΩ	6kΩ	200pF

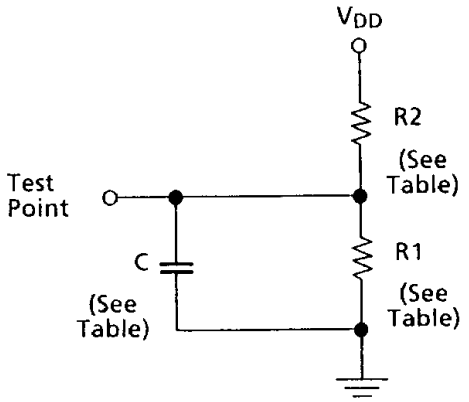


Figure 9.1 Equivalent Test Load

9.4 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

$P_D = P_{INT} + P_{I/O}$

$P_{INT} = I_{CC} \times V_{CC}$, Watts – Chip Internal Power

$P_{I/O}$ = Power Dissipation on Input and Output Pins – User Determined

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

An approximate relationship between P_D and T_J (if $P_{I/O}$ 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.

9.5 DC ELECTRICAL CHARACTERISTICS

(V_{DD} = 5.0V ± 10%, V_{SS} = 0V, T_A = T_L to T_H unless otherwise noted)

Characteristic	Symbol	Min	Typ	Max	Unit
Output Voltage, I_{Load} = ≤10.0μA	V_{OL} V_{OH}	– V_{DD} – 0.1	– –	0.1 –	V V
Output High Voltage (I_{Load} = 0.8mA) PA0-PA7, PB0-PB7, PC0-PC7, TCMP (See Figure 9.2) (I_{Load} = 1.6mA) PD1-PD4 (See Figure 9.3)	V_{OH} V_{OH}	V_{DD} – 0.8 V_{DD} – 0.8	– –	– –	V V
Output Low Voltage (See Figure 9.4) (I_{Load} = 1.6mA) PA0-PA7, PB0-PB7, PC0-PC7, PD1-PD4, TCMP	V_{OL}	–	–	0.4	V
Input High Voltage PA0-PA7, PB0-PB7, PC0-PC7, PD0-PD5, PD7, TCAP, $\overline{IRQ}$, $\overline{RESET}$, OSC1	V_{IH}	0.7 × V_{DD}	–	V_{DD}	V
Input Low Voltage PA0-PA7, PB0-PB7, PC0-PC7, PD0-PD5, PD7, TCAP, $\overline{IRQ}$, $\overline{RESET}$, OSC1	V_{IL}	V_{SS}	–	0.2 × V_{DD}	V
Data Retention Mode (0° to 70°C)	V_{RM}	2.0	–	–	V
Supply Current (See Notes) Run (See Figures 9.5 and 9.6) Wait (See Figures 9.5 and 9.6) Stop (See Figure 9.6) 25°C 0° to 70°C	 I_{DD} I_{DD} I_{DD} I_{DD}	 – – – –	 3.5 1.6 2.0 –	 7.0 4.0 50 140	 mA mA μA μA
I/O Ports Hi-Z Leakage Current PA0-PA7, PB0-PB7, PC0-PC7, PD1-PD4	I_{IL}	–	–	± 10	μA
Input Current $\overline{RESET}$, $\overline{IRQ}$, TCAP, OSC1, PD0, PD5, PD7	I_{in}	–	–	± 1	μA
Capacitance Ports (as Input or Output) $\overline{RESET}$, $\overline{IRQ}$, TCAP, OSC1, PD0-PD5, PD7	 C_{out} C_{in}	 – –	 – –	 12 8	 PF PF

Notes :

1. All values shown reflect average measurements.
2. Typical values at midpoint of voltage range, 25°C only.
3. Wait I_{DD}:Only timer system active (SPE = TE = RE = 0). If SPI, SCI active (SPE = TE = RE = 1) add 10% current draw.
4. Run (Operating) I_{DD}, Wait I_{DD}:Measured using external square wave clock source (fosc = 4.2 MHz), all inputs 0.2 V from rail, no DC loads, less than 50 pF on all outputs, C_L = 20 pF on OSC2.
5. Wait, Stop I_{DD}:All ports configured as inputs, V_{IL} = 0.2 V, V_{IH} = V_{DD} – 0.2 V.
6. Stop I_{DD} measured with OSC1 = V_{SS}.
7. Wait I_{DD} is affected linearly by the OSC2 capacitance.

9.6 DC ELECTRICAL CHARACTERISTICS

(V_{DD} = 3.3V ± 10%, V_{SS} = 0V, T_A = T_L to T_H unless otherwise noted)

Characteristic	Symbol	Min	Typ	Max	Unit
Output Voltage, $I_{Load} \leq \pm 10.0\mu A$	V_{OL} V_{OH}	– $V_{DD}-0.1$	– –	0.1 –	V V
Output High Voltage (I_{Load} = 0.2mA) PA0-PA7, PB-PB7, PC0-PC7, TCMP (See Figure 9.2) (I_{Load} = 0.4mA) PD1-PD4 (See Figure 9.3)	V_{OH} V_{OH}	$V_{DD}-0.3$ $V_{DD}-0.3$	– –	– –	V V
Output Low Voltage (See Figure 9.4) (I_{Load} = 0.4mA) PA0-PA7, PB0-PB7, PC0-PC7, PD1-PD4, TCMP	V_{OL}	–	–	0.3	V
Input High Voltage PA0-PA7, PB0-PB7, PC0-PC7, PD0-PD5, PD7, TCAP, $\overline{IRQ}$, $\overline{RESET}$, OSC1	V_{IH}	$0.7 \times V_{DD}$	–	V_{DD}	V
Input Low Voltage PA0-PA7, PB0-PB7, PC0-PC7, PD0-PD5, PD7, TCAP, $\overline{IRQ}$, $\overline{RESET}$, OSC1	V_{IL}	V_{SS}	–	$0.2 \times V_{DD}$	V
Data Retention Mode (0° to 70°C)	V_{RM}	2.0	–	–	V
Supply Current (See Notes) Run (See Figures 9.5 and 9.7) Wait (See Figures 9.5 snd 9.7) Stop (See Figure 9.7) 25°C 0° to 70°C	 I_{DD} I_{DD} I_{DD} I_{DD}	 – – – –	 1.0 0.5 1.0 –	 2.5 1.4 30 80	 mA mA μA μA
I/O Ports Hi-Z Leakage Current PA0-PA7, PB0-PB7, PC0-PC7, PD1-PD4	I_{IL}	–	–	± 10	μA
Input Current $\overline{RESET}$, $\overline{IRQ}$, TCAP, OSC1, PD0, PD5, PD7	I_{in}	–	–	± 1	μA
Capacitance Ports (as Input or Output) $\overline{RESET}$, $\overline{IRQ}$, TCAP, OSC1, PD0-PD5, PD7	 C_{out} C_{in}	 – –	 – –	 12 8	 PF PF

Notes :

1. All values shown reflect average measurements.
2. Typical values at midpoint of voltage range, 25°C only.
3. Wait I_{DD}:Only timer system active (SPE = TE = RE = 0). If SPI, SCI active (SPE = TE = RE = 1) add 10% current draw.
4. Run (Operating) I_{DD}, Wait I_{DD}: Measured using external square wave clock source (fosc = 2.0 MHz), all input 0.2 V from rail, no DC loads, less than 50 pF on all outputs, C_L = 20 pF on OSC2.
5. Wait, Stop I_{DD}:All ports configured as inputs, V_{IL} = 0.2 V, V_{IH} = V_{DD} – 0.2 V.
6. Stop I_{DD} measured with OSC1 = V_{SS}.
7. Wait I_{DD} is affected linearly by the OSC2 capacitance.

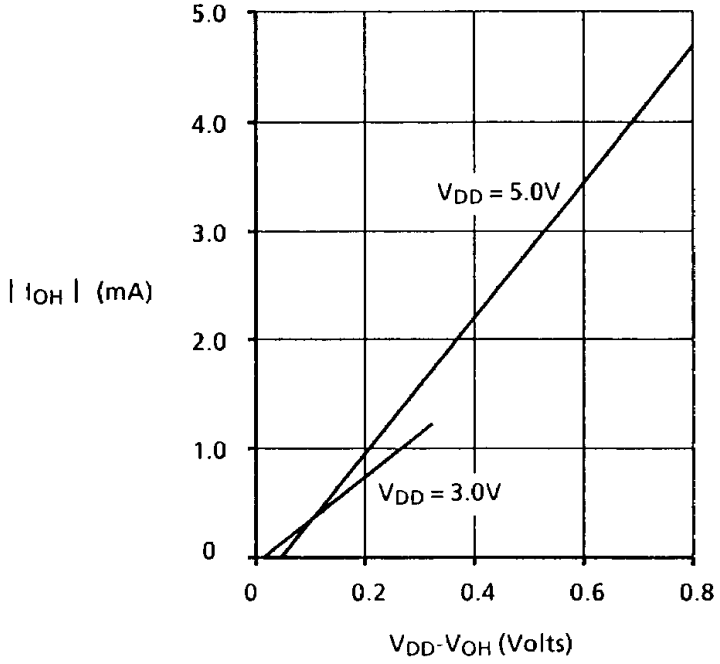


Figure 9.2 Typical V_{OH} vs I_{OH} for Ports A, B, C, and TCMP

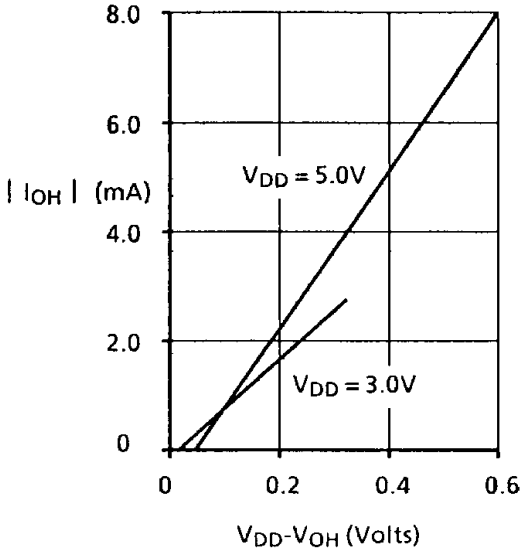


Figure 9.3 Typical V_{OH} vs I_{OH} for PD1-PD4

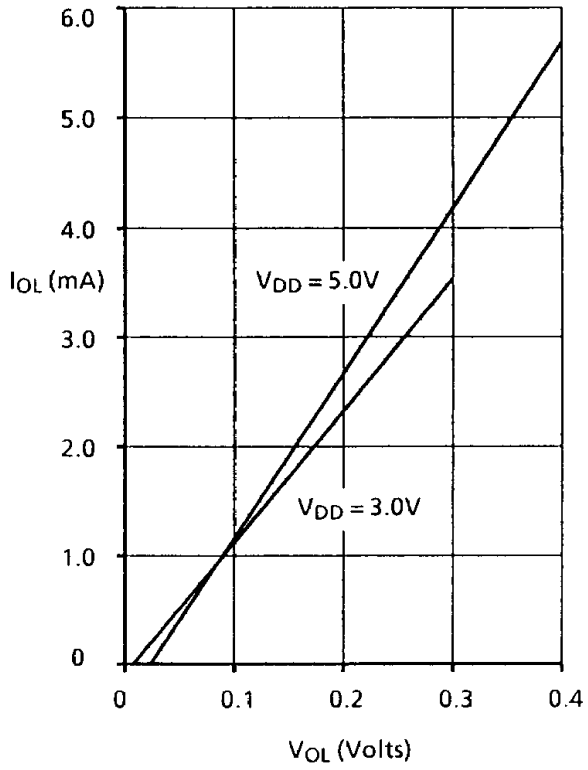


Figure 9.4 Typical V_{OL} vs I_{OL} for all Ports

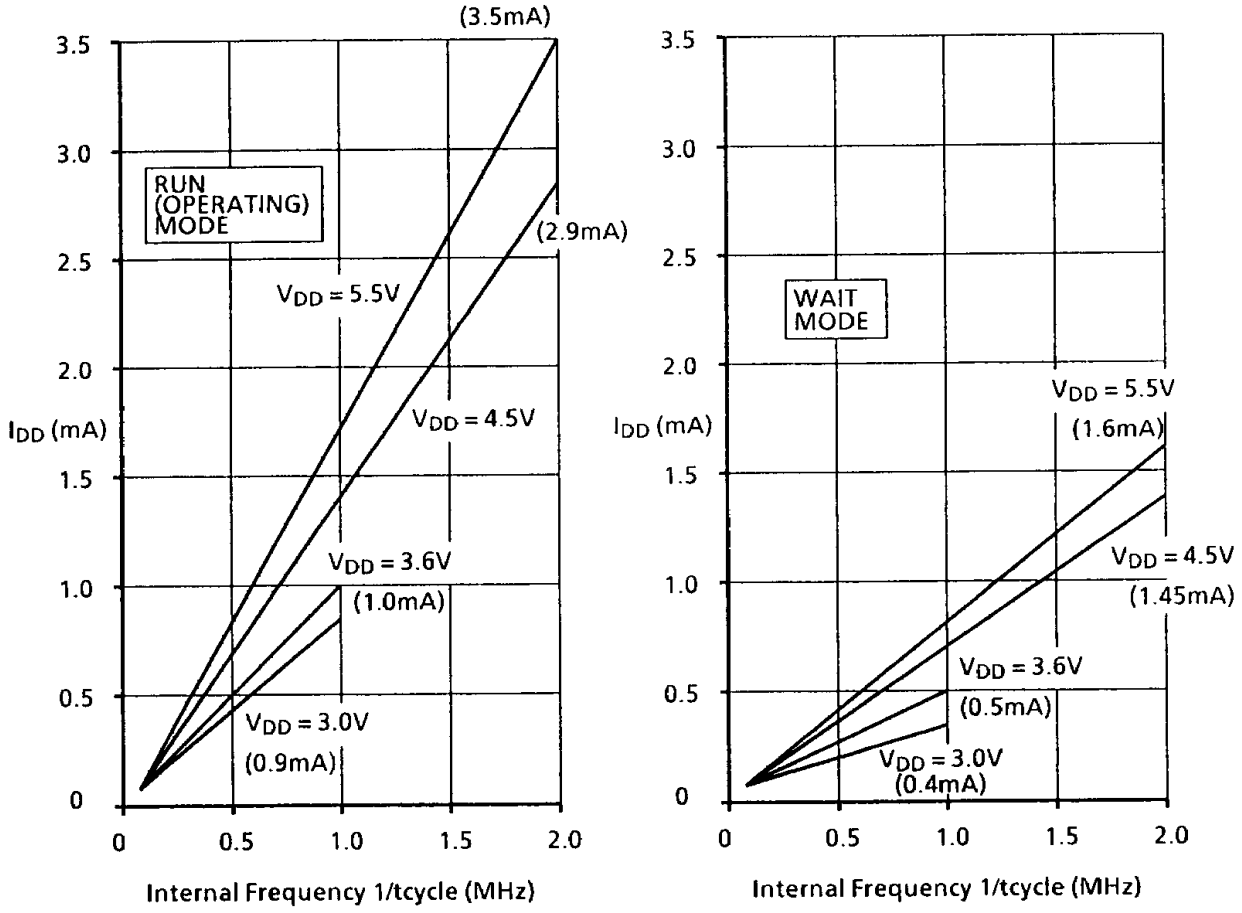


Figure 9.5 Typical Current vs Internal Frequency for Run and Wait Modes

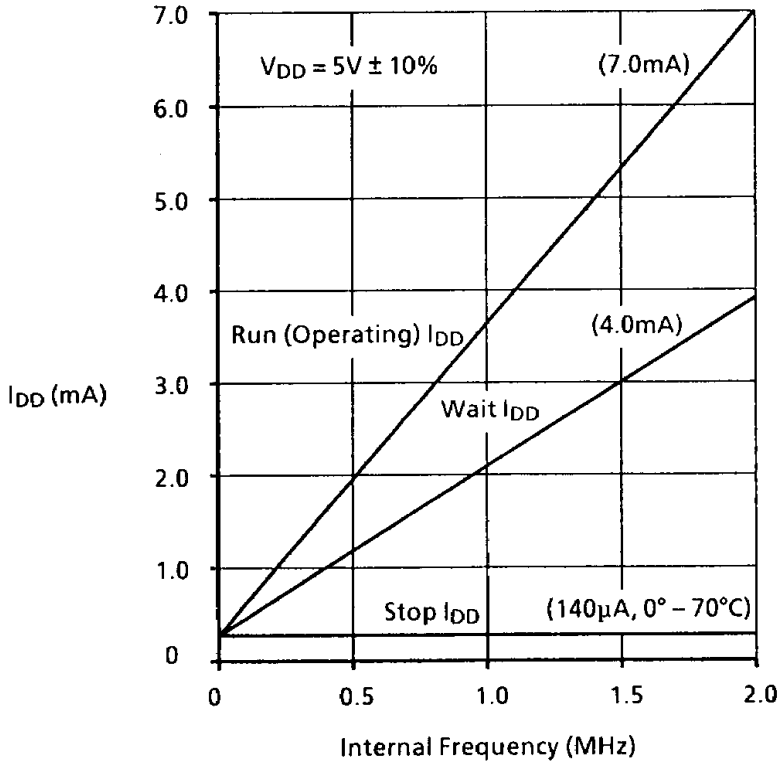


Figure 9.6 Maximum I_{DD} vs Frequency for V_{DD} = 5.0Vdc

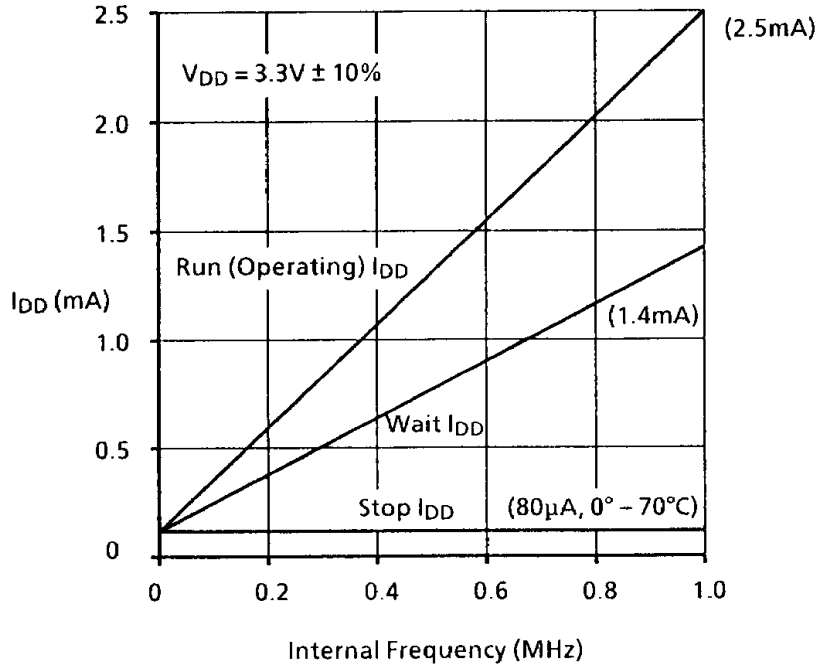


Figure 9.7 Maximum I_{DD} vs Frequency for V_{DD} = 3.3Vdc

9.7 CONTROL TIMING

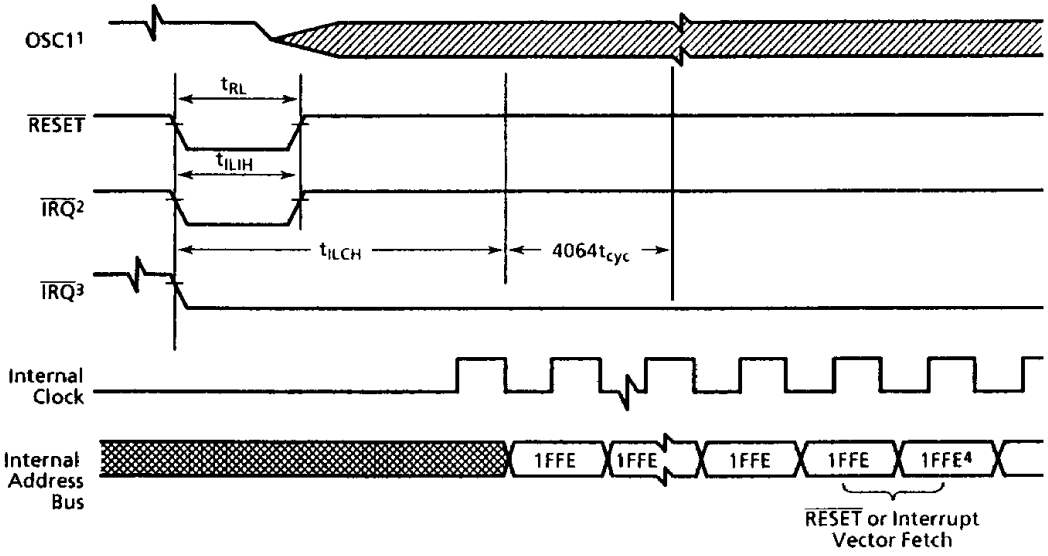
(V_{DD}=5.0V±10%, V_{SS}=0V, T_A=T_L to T_H)

Characteristic	Symbol	Min	Max	Unit
Frequency of Operation Crystal Option External Clock Option	 f_{osc} f_{osc}	 – dc	 4.2 4.2	 MHz MHz
Internal Operating Frequency Crystal (f_{osc} ÷ 2) External Clock (f_{osc} ÷ 2)	 f_{op} f_{op}	 – dc	 2.1 2.1	 MHz MHz
Cycle Time (See Figure 3.1)	t_{cyc}	480	–	ns
Crystal Oscillator Startup Time (See Figure 3.1)	t_{OXOV}	–	100	ms
Stop Recovery Startup Time (Crystal Oscillator) (See Figure 9.8)	t_{ILCH}	–	100	ms
$\overline{\text{RESET}}$ Pulse Width (See Figure 3.1)	t_{RL}	1.5	–	t_{cyc}
Timer Resolution** Input Capture Pulse Width (See Figure 9.9) Input Capture Pulse Period (See Figure 9.9)	 t_{RESL} t_{TH}, t_{TL} t_{TLTL}	 4.0 125 ***	 – – –	 t_{cyc} ns t_{cyc}
Interrupt Pulse Width Low (Edge-Triggered) (See Figure 3.4)	t_{ILIH}	125	–	ns
Interrupt Pulse Period (See Figure 3.4)	t_{ILIL}	*	–	t_{cyc}
OSC1 Pulse Width	t_{OH}, t_{OL}	90	–	ns

* The minimum period t_{ILIL} should not be less than the number of cycle times it takes to execute the interrupt service routine plus 21 t_{cyc}.

** Since a 2-bit prescaler in the timer must count four internal cycles (t_{cyc}), this is the limiting minimum factor in determining the timer resolution.

*** The minimum period t_{TLTL} should not be less than the number of cycle times it takes to execute the capture interrupt service routine plus 24 t_{cyc}.



Notes:
1. Represents the internal gating of the OSC1 pin.
2. $\overline{\text{IRQ}}$ pin edge-sensitive mask option.
3. $\overline{\text{IRQ}}$ pin level and edge-sensitive mask option.
4. $\overline{\text{RESET}}$ vector address shown for timing example.

Figure 9.8 Stop Recovery Timing Diagram

9.8 CONTROL TIMING

(V_{DD} = 3.3V ± 10%, V_{SS} = 0V, T_A = T_L to T_H)

Characteristic	Symbol	Min	Max	Unit
Frequency of Operation Crystal Option External Clock Option	 f_{osc} f_{osc}	 – dc	 2.0 2.0	 MHz MHz
Internal Operating Frequency Crystal (f_{osc} ÷ 2) External Clock (f_{osc} ÷ 2)	 f_{op} f_{op}	 – dc	 1.0 1.0	 MHz MHz
Cycle Time (See Figure 3.1)	t_{cyc}	1000	–	ns
Crystal Oscillator Startup Time (See Figure 3.1)	t_{OXOV}	–	100	ms
Stop Recovery Startup Time (Crystal Oscillator) (See Figure 9.8)	t_{ILCH}	–	100	ms
$\overline{RESET}$ Pulse Width-Excluding Power-Up (See Figure 3.1)	t_{RL}	1.5	–	t_{cyc}
Timer Resolution** Input Capture Pulse Width (See Figure 9.9) Input Capture Pulse Period (See Figure 9.9)	 t_{RESL} t_{TH}, t_{TL} t_{TLTL}	 4.0 250 ***	 – – –	 t_{cyc} ns t_{cyc}
Interrupt Pulse Width Low (Edge-Triggered) (See Figure 3.4)	t_{ILIH}	250	–	ns
Interrupt Pulse Period (See Figure 3.4)	t_{ILIL}	*	–	t_{cyc}
OSC1 Pulse Width	t_{OH}, t_{OL}	200	–	ns

* The minimum period t_{ILIL} should not be less than the number of cycle times it takes to execute the interrupt service routine plus 21 t_{cyc}.

** Since a 2-bit prescaler in the timer must count four internal cycles (t_{cyc}), this is the limiting minimum factor in determining the timer resolution.

*** The minimum period t_{TLTL} should not be less than the number of cycle times it takes to execute the capture interrupt service routine plus 24 t_{cyc}.

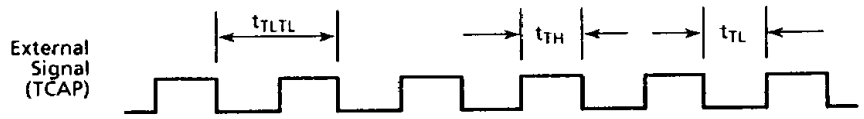


Figure 9.9 Timer Relationships

9.9 SERIAL PERIPHERAL INTERFACE (SPI) TIMING

($V_{DD}=5.0V\pm10\%$, $V_{SS}=0V$, $T_A=T_L$ to T_H)

Num.	Characteristic	Symbol	Min	Max	Unit
	Operating Frequency Master Slave	 $f_{op\,(m)}$ $f_{op\,(s)}$	 dc dc	 0.5 2.1	f_{op} MHz
1	Cycle Time Master Slave	 $t_{cyc\,(m)}$ $t_{cyc\,(s)}$	 2.0 480	 – –	t_{cyc} ns
2	Enable Lead Time Master Slave	 $t_{lead\,(m)}$ $t_{lead\,(s)}$	 * 240	 – –	ns
3	Enable Lag Time Master Slave	 $t_{lag\,(m)}$ $t_{lag\,(s)}$	 * 240	 – –	ns
4	Clock (SCK) High Time Master Slave	 $t_{w}\,(SCKH)_{m}$ $t_{w}\,(SCKH)_{s}$	 340 190	 – –	 ns ns
5	Clock (SCK) Low Time Master Slave	 $t_{w}\,(SCKL)_{m}$ $t_{w}\,(SCKL)_{s}$	 340 190	 – –	 ns ns
6	Data Setup Time (Inputs) Master Slave	 $t_{su\,(m)}$ $t_{su\,(s)}$	 100 100	 – –	 ns ns
7	Data Hold Time (Inputs) Master Slave	 $t_{h\,(m)}$ $t_{h\,(s)}$	 100 100	 – –	 ns ns
8	Access Time (Time to Data Active from High Impedance State) Slave	t_a	 0	 120	ns
9	Disable Time (Hold Time to High-Impedance State) Slave	t_{dis}	 –	 240	ns
10	Data Valid Master (Before Capture Edge) Slave (After Enable Edge) **	 $t_{v\,(m)}$ $t_{v\,(s)}$	 0.25 –	 – 240	 $t_{cyc\,(m)}$ ns
11	Data Hold Time (Outputs) Master (After Capture Edge) Slave (After Enable Edge)	 $t_{ho\,(m)}$ $t_{ho\,(s)}$	 0.25 0	 – –	 $t_{cyc\,(m)}$ ns
12	Rise Time (20% V_{DD} to 70% V_{DD}, C_L = 200pF) SPI Outputs (SCK, MOSI, MISO) SPI Inputs (SCK, MOSI, MISO, $\overline{SS}$)	 t_{rm} t_{rs}	 – –	 100 2.0	 ns μs
13	Fall Time (70% V_{DD} to 20% V_{DD}, C_L = 200pF) SPI Outputs (SCK, MOSI, MISO) SPI Inputs (SCK, MOSI, MISO, $\overline{SS}$)	 t_{fm} t_{fs}	 – –	 100 2.0	 ns μs

* Signal production depends on software.

** Assumes 200 pF load on all SPI pins.

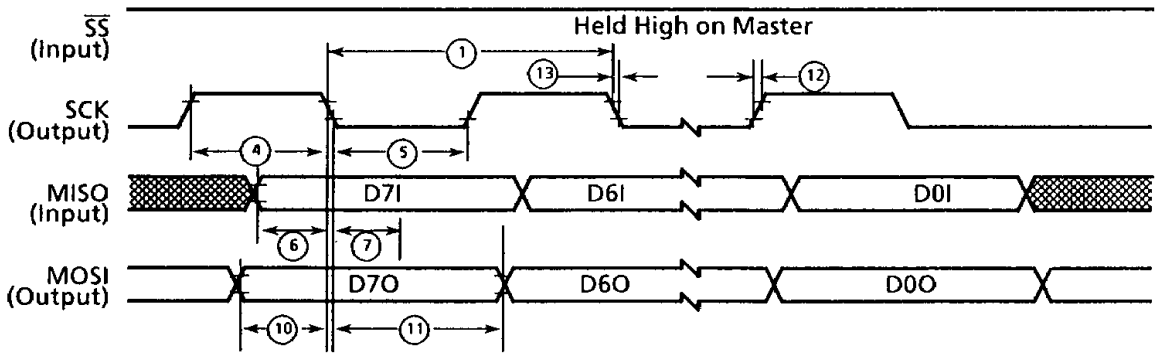
9.10 SERIAL PERIPHERAL INTERFACE (SPI) TIMING

(V_{DD} = 3.3V ± 10%, V_{SS} = 0V, T_A = T_L to T_H)

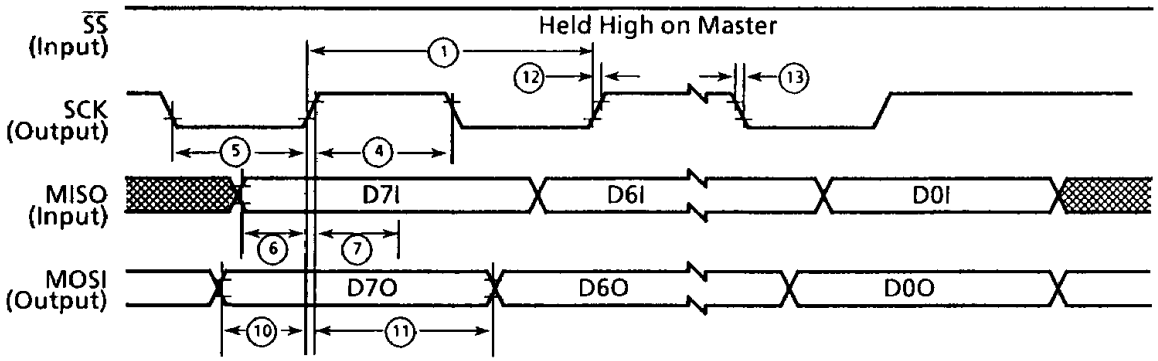
Num.	Characteristic	Symbol	Min	Max	Unit
	Operating Frequency Master Slave	 f_{op} (m) f_{op} (s)	 dc dc	 0.5 1.0	f_{op} MHz
1	Cycle Time Master Slave	 t_{cyc} (m) t_{cyc} (s)	 2.0 1.0	 – –	t_{cyc} μs
2	Enable Lead Time Master Slave	 t_{lead} (m) t_{lead} (s)	 * 500	 – –	ns
3	Enable Lag Time Master Slave	 t_{lag} (m) t_{lag} (s)	 * 500	 – –	ns
4	Clock (SCK) High Time Master Slave	 t_w (SCKH) m t_w (SCKH) s	 720 400	 – –	 μs ns
5	Clock (SCK) Low Time Master Slave	 t_w (SCKL) m t_w (SCKL) s	 720 400	 – –	 μs ns
6	Data Setup Time (Inputs) Master Slave	 t_{su} (m) t_{su} (s)	 200 200	 – –	 ns ns
7	Data Hold Time (Inputs) Master Slave	 t_h (m) t_h (s)	 200 200	 – –	 ns ns
8	Access Time (Time to Data Active from High-Impedance State) Slave	t_a	 0	 250	ns
9	Disable Time (Hold Time to High-Impedance State) Slave	t_{dis}	 –	 500	ns
10	Data Valid Master (Before Capture Edge) Slave (After Enable Edge) **	 t_v (m) t_v (s)	 0.25 –	 – 500	 t_{cyc} (m) ns
11	Data Hold Time (Outputs) Master (After Capture Edge) Slave (After Enable Edge)	 t_{ho} (m) t_{ho} (s)	 0.25 0	 – –	 t_{cyc} (m) ns
12	Rise Time (20% V_{DD} to 70% V_{DD}, C_L = 200pF) SPI Outputs (SCK, MOSI, MISO) SPI Inputs (SCK, MOSI, MISO, $\overline{SS}$)	 t_{rm} t_{rs}	 – –	 200 2.0	 ns μs
13	Fall Time (70% V_{DD} to 20% V_{DD}, C_L = 200pF) SPI Outputs (SCK, MOSI, MISO) SPI Inputs (SCK, MOSI, MISO, $\overline{SS}$)	 t_{fm} t_{fs}	 – –	 200 2.0	 ns μs

* Signal production depends on software.

** Assumes 200 pF load on all SPI pins.

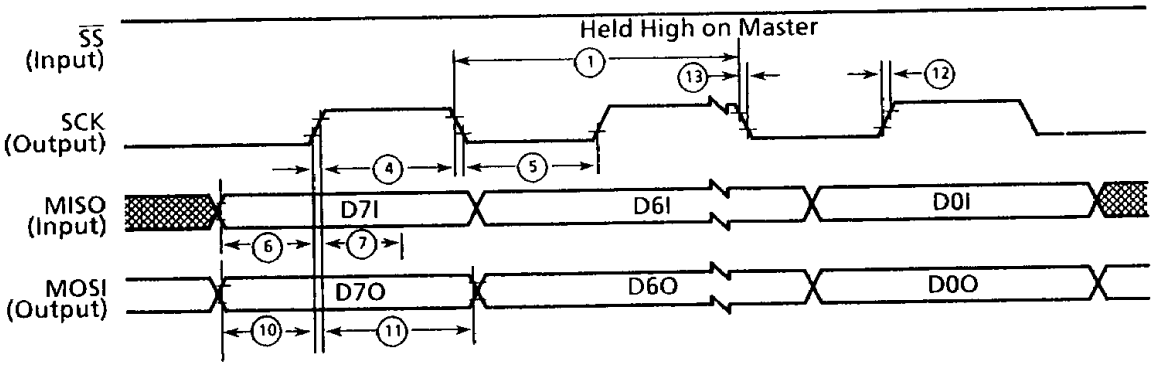


(a) SPI Master Timing CPOL = 0, CPHA = 1

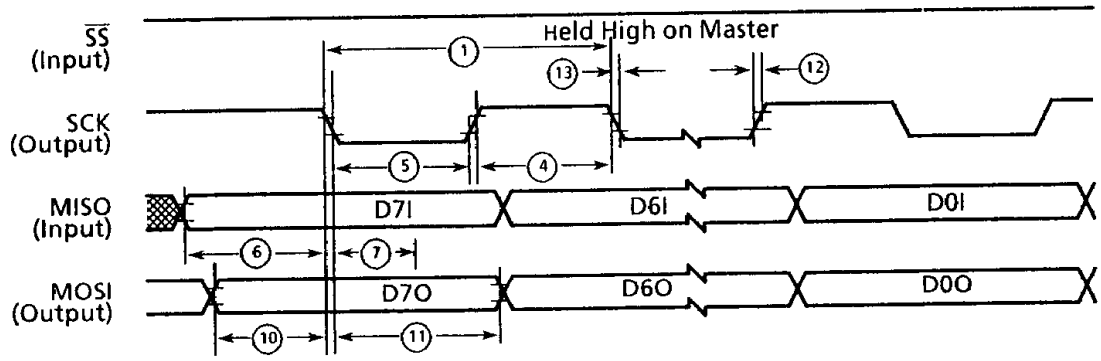


(b) SPI Master Timing CPOL = 1, CPHA = 1

Figure 9.10 SPI Timing Diagrams
(Sheet1 of 4)



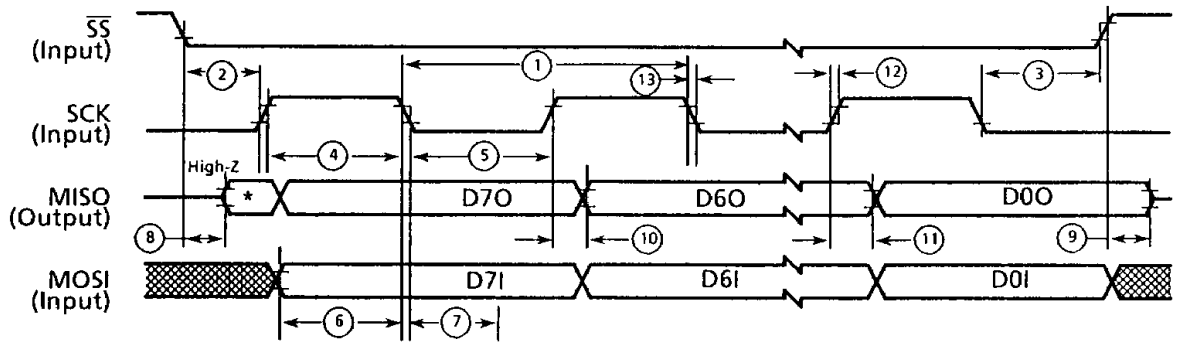
(C) SPI Master Timing CPOL = 0, CPHA = 0



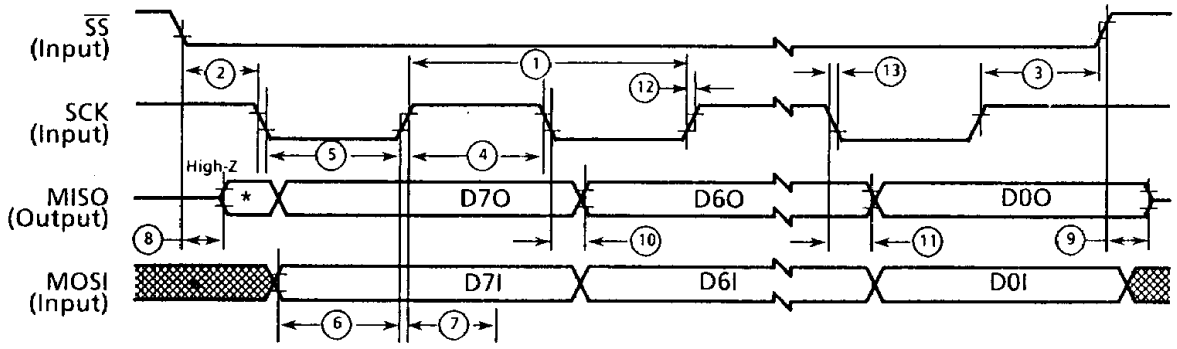
(d) SPI Master Timing CPOL = 1, CPHA = 0

Note: Measurement points are V_{OL}, V_{OH}, V_{IL}, and V_{IH}

Figure 9.10 SPI Timing Diagrams
(Sheet2 of 4)

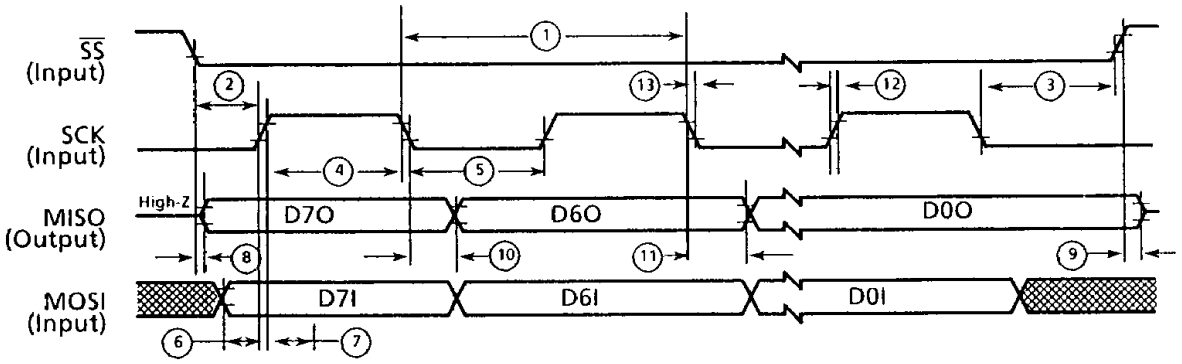


(e) SPI Slave Timing CPOL = 0, CPHA = 1

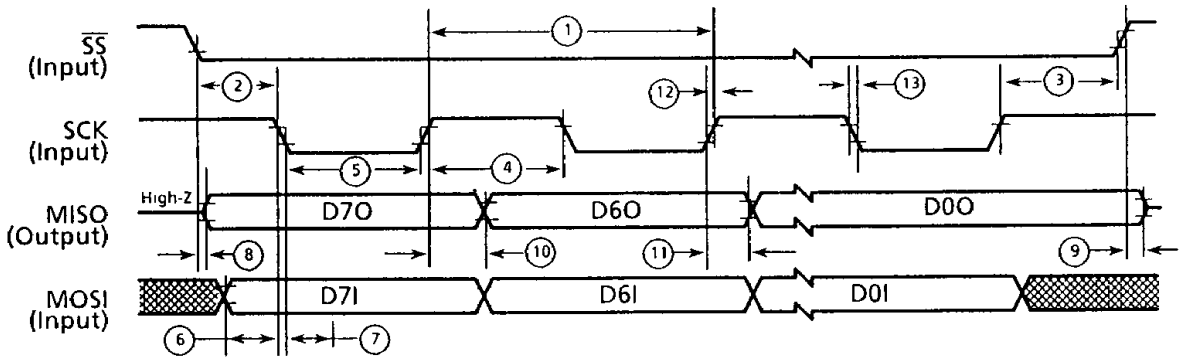


(f) SPI Slave Timing CPOL = 1, CPHA = 1

Figure 9.10 SPI Timing Diagrams
(Sheet3 of 4)



(g) SPI Slave Timing CPOL = 0, CPHA = 0



(h) SPI Slave Timing CPOL = 1, CPHA = 0

Notes :
(1) Measurement points are V_{OL}, V_{OH}, V_{IL}, and V_{IH}
(2) *Denotes undefined, either high or low.

Figure 9.10 SPI Timing Diagrams
(Sheet4 of 4)

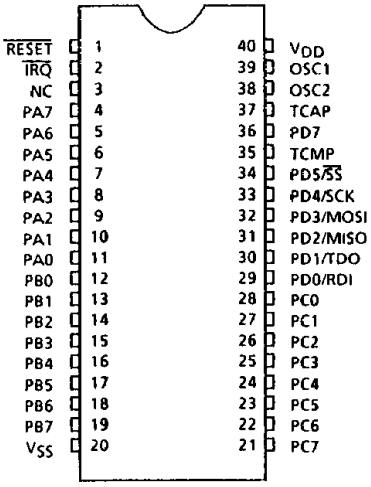
10. MECHANICAL DATA

This section contains the pin assignment and package dimension diagrams for the TMP68HC05C4 microcomputer.

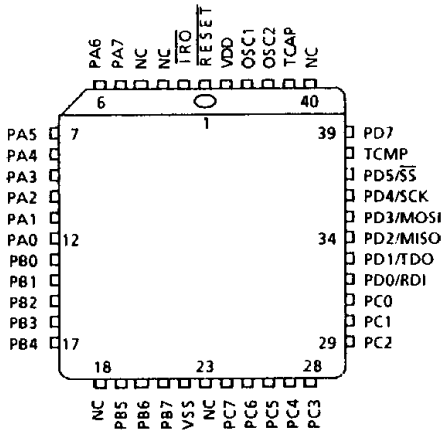
10.1 PIN ASSIGNMENT

UNIT : mm

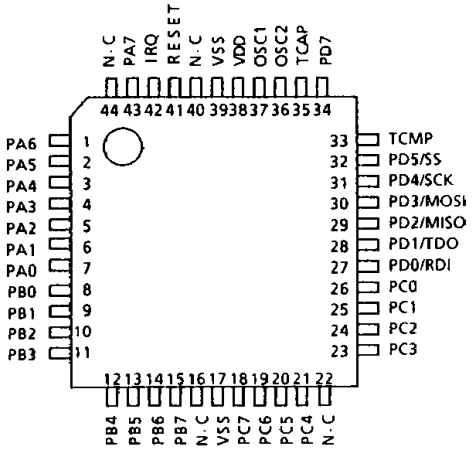
P SUFFIX
40-Pin Dual-in-Line Package



T SUFFIX
44-Lead PLCC Package

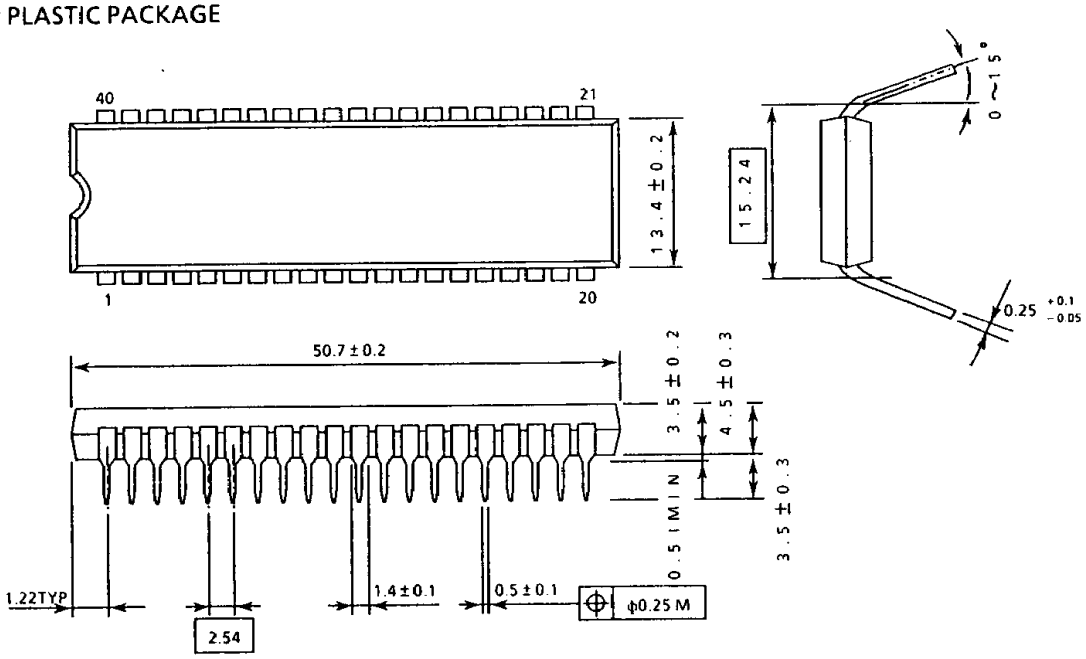


U SUFFIX
44-Pin Quad Flat Package

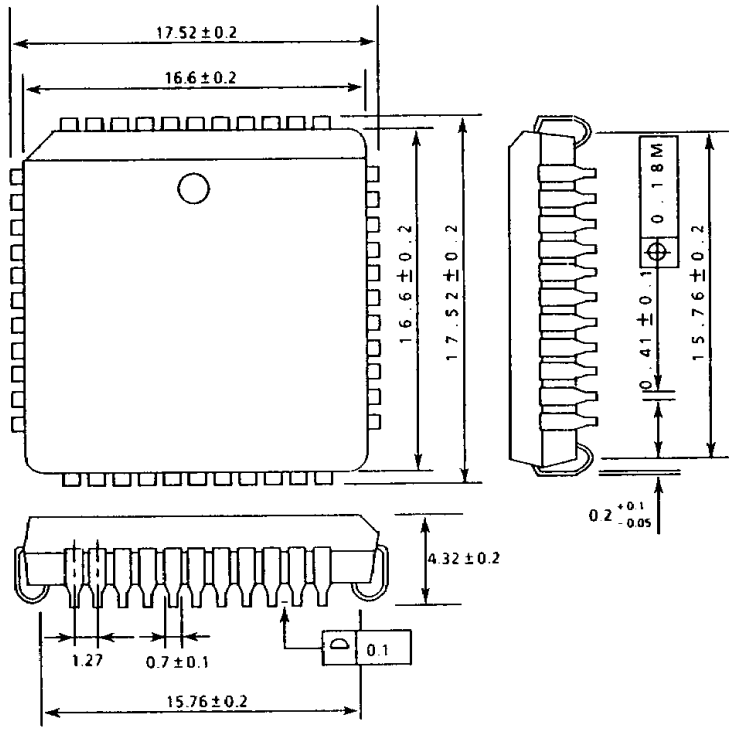


10.2 PACKAGE DIMENSIONS

P SUFFIX
DIP PLASTIC PACKAGE



T SUFFIX
PLCC PACKAGE



U SUFFIX
QFP PACKAGE

UNIT : mm