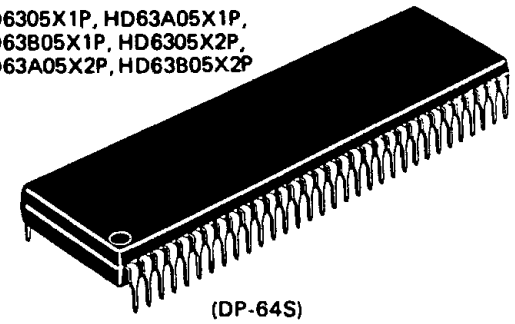
HD6305X1, HD63A05X1, HD63B05X1
HD6305X2, HD63A05X2, HD63B05X2
CMOS MCU (Microcomputer Unit)

The HD6305X1 and the HD6305X2 are memory expandable versions of the HD6305X0, which is CMOS 8-bit single chip microcomputer. A CPU, a clock generator, a 128-byte RAM, I/O terminals, two timers and a serial communication interface (SCI) are built in both chip of the HD6305X1 and the HD 6305X2. Their memory spaces are expandable to 16k bytes externally.

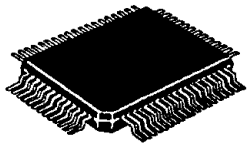
The HD6305X1 and the HD6305X2 have the same functions as the HD6305X0's except for the number of I/O terminals. The HD6305X1 has a 4k byte ROM and its memory space is expandable to 12k bytes externally. The HD6305X2 is a microcomputer unit which includes no ROM and its memory space is expandable to 16k bytes externally.

HD6305X1P, HD63A05X1P, HD63B05X1P, HD6305X2P, HD63A05X2P, HD63B05X2P



(DP-64S)

HD6305X1F, HD63A05X1F, HD63B05X1F, HD6305X2F, HD63A05X2F, HD63B05X2F



(FP-64)

■ HARDWARE FEATURES

- 8-bit based MCU
- 4k-bytes of internal ROM (HD6305X1)
  No internal ROM (HD6305X2)
- 128-bytes of RAM
- A total of 31 terminals, including 24 I/O's, 7 inputs
- Two timers
  - 8-bit timer with a 7-bit prescaler (programmable prescaler; event counter)
  - 15-bit timer (commonly used with the SCI clock divider)
- On-chip serial interface circuit (synchronized with clock)
- Six interrupts (two external, two timer, one serial and one software)
- Low power dissipation modes
  - Wait . . . . In this mode, the clock oscillator is on and the CPU halts but the timer/serial/interrupt function is operatable.
  - Stop . . . . In this mode, the clock stops but the RAM data, I/O status and registers are held.
  - Standby . . In this mode, the clock stops, the RAM data is held, and the other internal condition is reset.
- Minimum instruction cycle time
  - HD6305X1/X2 . . 1 μs (f = 1 MHz)
  - HD63A05X1/X2 . . 0.67 μs (f = 1.5 MHz)
  - HD63B05X1/X2 . . 0.5 μs (f = 2 MHz)
- Wide operating range
  V_{CC} = 3 to 6V (f = 0.1 to 0.5 MHz)
  - HD6305X1/X2 . . f = 0.1 to 1 MHz (V_{CC} = 5V ± 10%)
  - HD63A05X1/X2 . . f = 0.1 to 1.5 MHz (V_{CC} = 5V ± 10%)
  - HD63B05X1/X2 . . f = 0.1 to 2 MHz (V_{CC} = 5V ± 10%)

■ SOFTWARE FEATURES

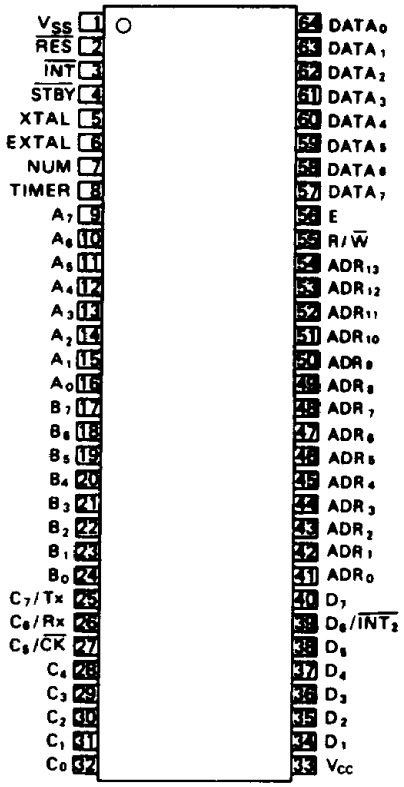
- Similar to HD6800
- Byte efficient instruction set
- Powerful bit manipulation instructions (Bit Set, Bit Clear, and Bit Test and Branch usable for all RAM bits and all I/O terminals)
- A variety of interrupt operations
- Index addressing mode useful for table processing
- A variety of conditional branch instructions
- Ten powerful addressing modes
- All addressing modes adaptable to RAM, and I/O instructions
- Three new instructions, STOP, WAIT and DAA, added to the HD6805 family instruction set

■ PROGRAM DEVELOPMENT SUPPORT TOOLS

- Cross assembler software for use with IBM PCs and compatibles
- In circuit emulator for use with IBM PCs and compatibles

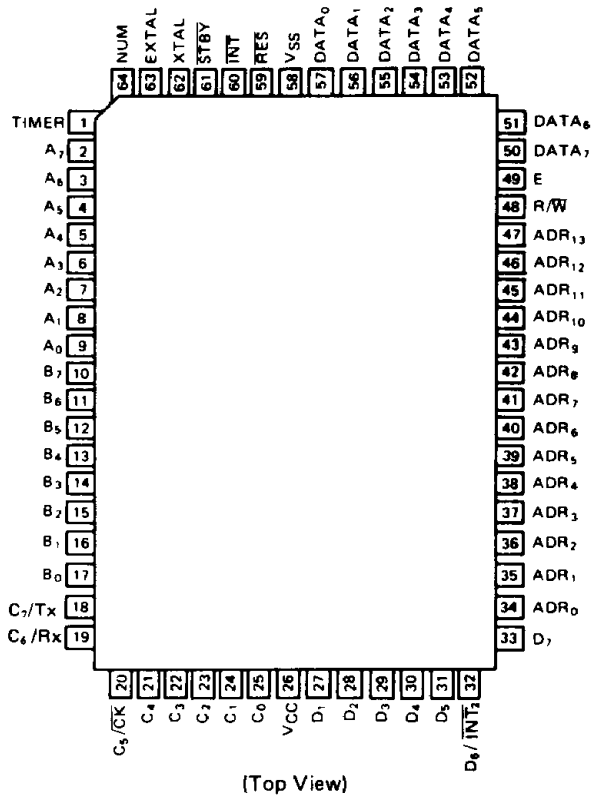
■ PIN ARRANGEMENT

- HD6305X1P, HD63A05X1P, HD63B05X1P, HD6305X2P, HD63A05X2P, HD63B05X2P



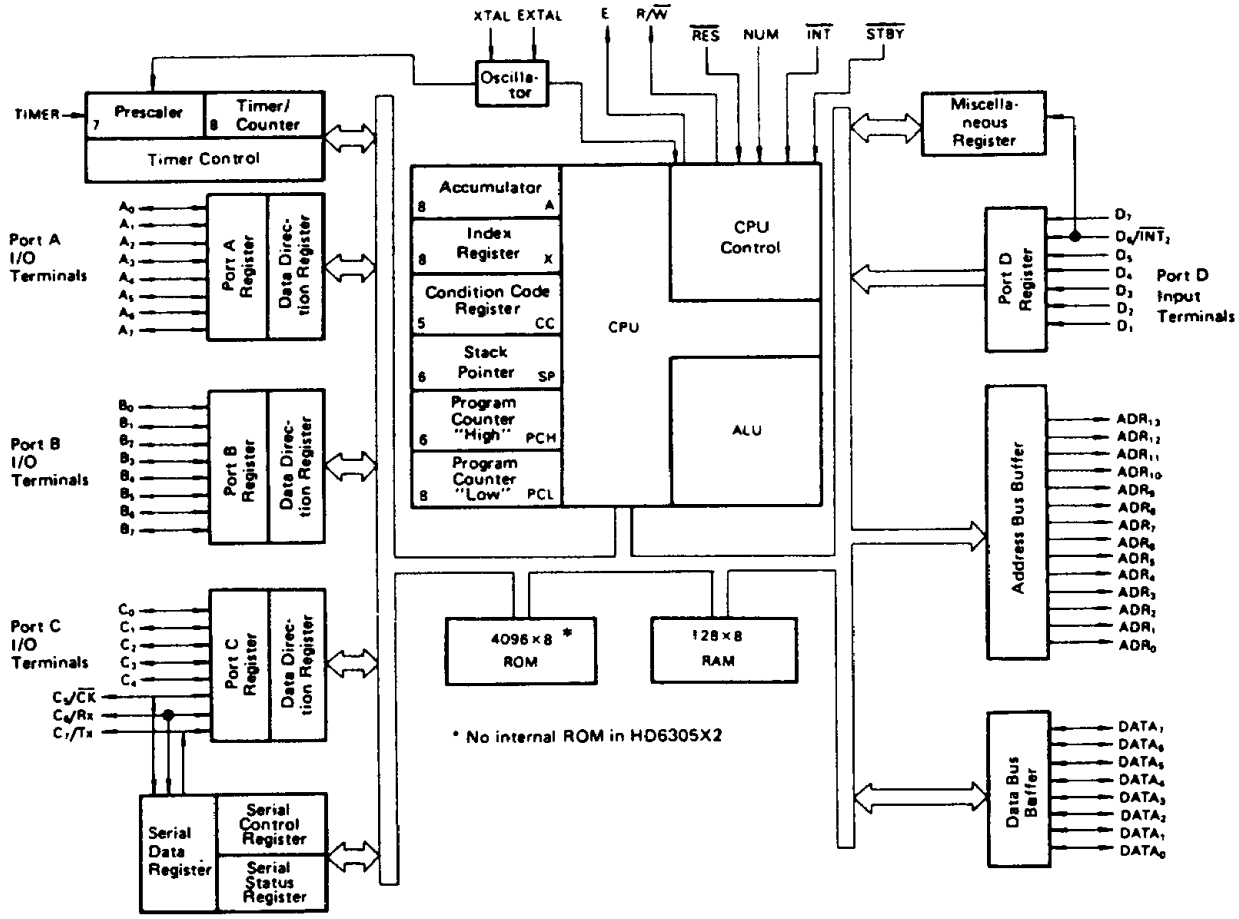
(Top View)

- **HD6305X1F, HD63A05X1F, HD63B05X1F, HD6305X2F, HD63A05X2F, HD63B05X2F**



(Top View)

■ BLOCK DIAGRAM



■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3 ~ +7.0	V
Input Voltage	V_{in}	−0.3 ~ V_{CC} + 0.3	V
Operating Temperature	T_{opr}	0 ~ +70	°C
Storage Temperature	T_{stg}	−55 ~ +150	°C

[NOTE] These products have a protection circuit in their input terminals against high electrostatic voltage or high electric fields. Notwithstanding, be careful not to apply any voltage higher than the absolute maximum rating to these high input impedance circuits. To assure normal operation, we recommended V_{in}, V_{out}; $V_{SS} \leqq (V_{in}$ or $V_{out}) \leqq V_{CC}$.

■ ELECTRICAL CHARACTERISTICS

• DC CHARACTERISTICS (V_{CC} = 5.0V±10%, V_{SS} = GND, Ta = 0 ~ +70°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage	$\overline{RES}$, $\overline{STBY}$	V_{IH}		V_{CC}-0.5	–	V_{CC}+0.3	V
	EXTAL			V_{CC}x0.7	–	V_{CC}+0.3	
	Other Inputs			2.0	–	V_{CC}+0.3	
Input "Low" Voltage	All Inputs	V_{IL}		-0.3	–	0.8	V
Output "High" Voltage	All Outputs	V_{OH}	I_{OH} = -200μA	2.4	–	–	V
			I_{OH} = -10μA	V_{CC}-0.7	–	–	
Output "Low" Voltage	All Outputs	V_{OL}	I_{OL} = 1.6mA	–	–	0.55	V
Input Leakage Current	TIMER, $\overline{INT}$, $D_1 \sim D_7$, $\overline{STBY}$	$\|I_{IL}\|$	Vin = 0.5 ~ V_{CC}-0.5	–	–	1.0	μA
Three-state Current	$A_0 \sim A_7$, $B_0 \sim B_7$, $C_0 \sim C_7$, $ADR_0 \sim ADR_{13}$*, $DATA_0 \sim DATA_7$, E*, R/$\overline{W}$*	$\|I_{TSI}\|$		–	–	1.0	μA
Current Dissipation**	Operating	I_{CC}	f = 1MHz***	–	5	10	mA
	Wait			–	2	5	mA
	Stop			–	2	10	μA
	Standby			–	2	10	μA
Input Capacitance	All Terminals	Cin	f = 1MHz, Vin = 0V	–	–	12	pF

* Only at standby
** V_{IH} min = V_{CC}-1.0V, V_{IL} max = 0.8V
*** The value at f = x MHz is given by using
I_{CC} (f = x MHz) = I_{CC} (f = 1MHz) x x

• AC CHARACTERISTICS (V_{CC} = 5.0V±10%, V_{SS} = GND, Ta = 0 ~ +70°C, unless otherwise noted.)

Item	Symbol	Test Condition	HD6305X1/X2			HD63A05X1/X2			HD63B05X1/X2			Unit
			min	typ	max	min	typ	max	min	typ	max	
Cycle Time	t_{cyc}	Fig. 1	1	–	10	0.666	–	10	0.5	–	10	μs
Enable Rise Time	t_{Er}		–	–	20	–	–	20	–	–	20	ns
Enable Fall Time	t_{Ef}		–	–	20	–	–	20	–	–	20	ns
Enable Pulse Width("High" Level)	PW_{EH}		450	–	–	300	–	–	220	–	–	ns
Enable Pulse Width("Low" Level)	PW_{EL}		450	–	–	300	–	–	220	–	–	ns
Address Delay Time	t_{AD}		–	–	250	–	–	190	–	–	180	ns
Address Hold Time	t_{AH}		40	–	–	30	–	–	20	–	–	ns
Data Delay Time	t_{DW}		–	–	200	–	–	160	–	–	120	ns
Data Hold Time (Write)	t_{HW}		40	–	–	30	–	–	20	–	–	ns
Data Set-up Time (Read)	t_{DSR}		80	–	–	60	–	–	50	–	–	ns
Data Hold Time (Read)	t_{HR}		0	–	–	0	–	–	0	–	–	ns

- **PORT TIMING (V_{CC} = 5.0V±10%, V_{SS} = GND, Ta = 0 ~ +70°C, unless otherwise noted.)**

Item	Symbol	Test Condition	HD6305X1/X2			HD63A05X1/X2			HD63B05X1/X2			Unit
			min	typ	max	min	typ	max	min	typ	max	
Port Data Set-up Time (Port A, B, C, D)	t_{PDS}	Fig. 2	200	–	–	200	–	–	200	–	–	ns
Port Data Hold Time (Port A, B, C, D)	t_{PDH}		200	–	–	200	–	–	200	–	–	ns
Port Data Delay Time (Port A, B, C)	t_{PDW}	Fig. 3	–	–	300	–	–	300	–	–	300	ns

- **CONTROL SIGNAL TIMING (V_{CC} = 5.0V±10%, V_{SS} = GND, Ta = 0 ~ +70°C, unless otherwise noted.)**

Item	Symbol	Test Condition	HD6305X1/X2			HD63A05X1/X2			HD63B05X1/X2			Unit
			min	typ	max	min	typ	max	min	typ	max	
$\overline{INT}$ Pulse Width	t_{IWL}		t_{cyc} +250	–	–	t_{cyc} +200	–	–	t_{cyc} +200	–	–	ns
$\overline{INT_2}$ Pulse Width	t_{IWL2}		t_{cyc} +250	–	–	t_{cyc} +200	–	–	t_{cyc} +200	–	–	ns
$\overline{RES}$ Pulse Width	t_{RWL}		5	–	–	5	–	–	5	–	–	t_{cyc}
Control Set-up Time	t_{CS}	Fig. 5	250	–	–	250	–	–	250	–	–	ns
Timer Pulse Width	t_{TWL}		t_{cyc} +250	–	–	t_{cyc} +200	–	–	t_{cyc} +200	–	–	ns
Oscillation Start Time (Crystal)	t_{OSC}	Fig.5, Fig.20*	–	–	20	–	–	20	–	–	20	ms
Reset Delay Time	t_{RHL}	Fig. 19	80	–	–	80	–	–	80	–	–	ms

* C_L = 22pF ±20%, R_s = 60Ω max.

- **SCI TIMING (V_{CC} = 5.0V±10%, V_{SS} = GND, Ta = 0 ~ +70°C, unless otherwise noted.)**

Item	Symbol	Test Condition	HD6305X1/X2			HD63A05X1/X2			HD63B05X1/X2			Unit
			min	typ	max	min	typ	max	min	typ	max	
Clock Cycle	t_{Scyc}	Fig. 6, Fig. 7	1	–	32768	0.67	–	21845	0.5	–	16384	μs
Data Output Delay Time	t_{TXD}		–	–	250	–	–	250	–	–	250	ns
Data Set-up Time	t_{SRX}		200	–	–	200	–	–	200	–	–	ns
Data Hold Time	t_{HRX}		100	–	–	100	–	–	100	–	–	ns

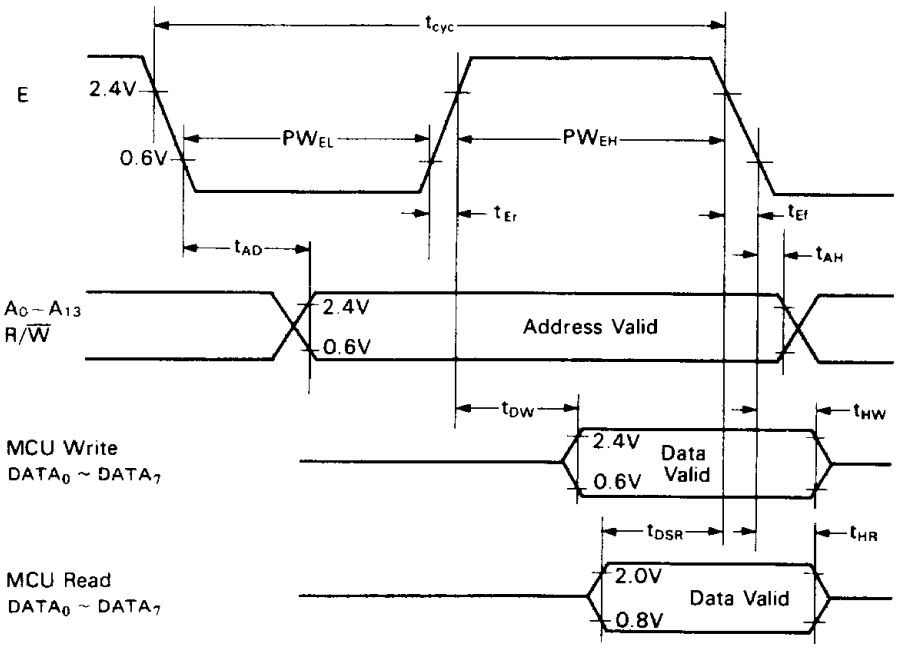


Figure 1 Bus Timing

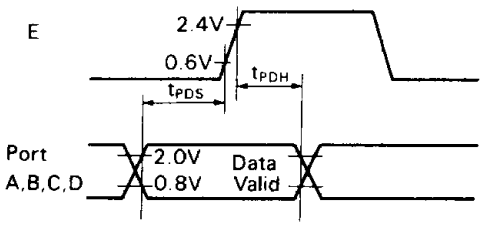


Figure 2 Port Data Set-up and Hold Times (MCU Read)

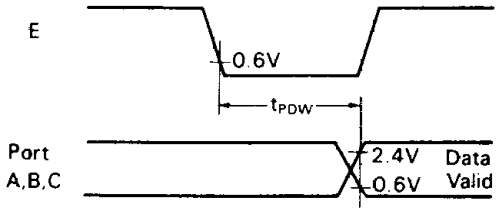


Figure 3 Port Data Delay Time (MCU Write)

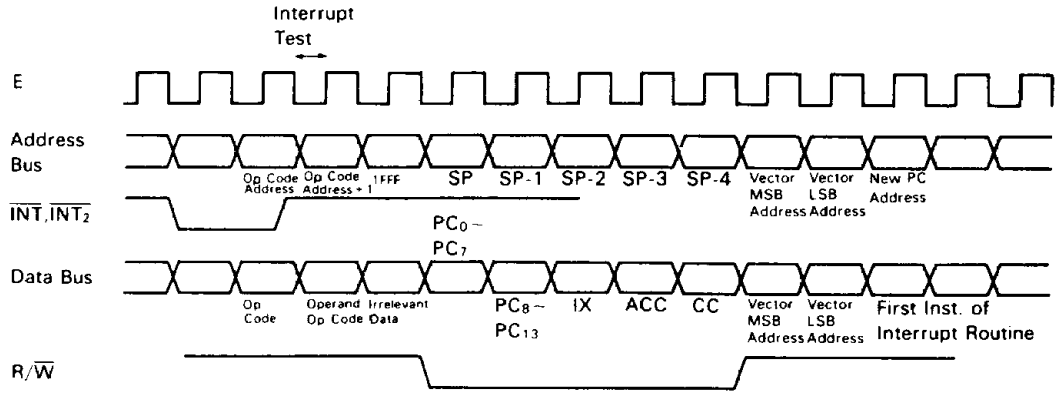


Figure 4 Interrupt Sequence

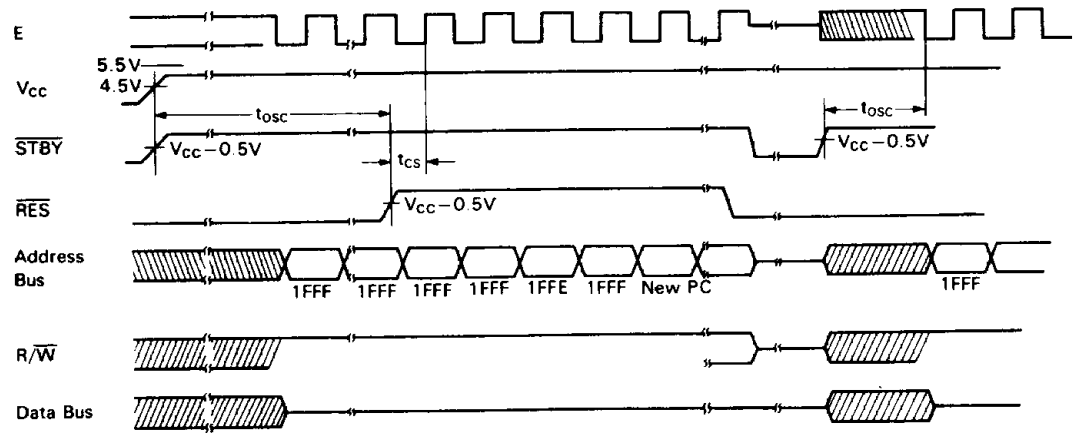


Figure5 Reset Timing

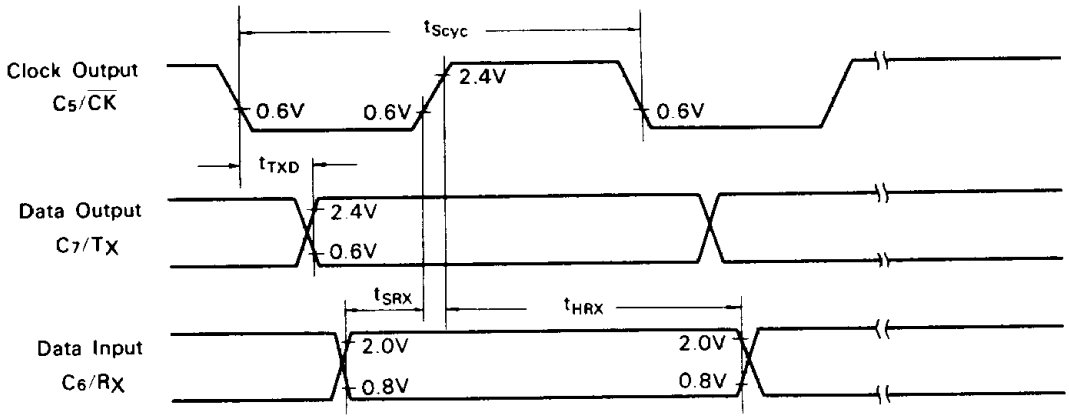


Figure6 SCI Timing (Internal Clock)

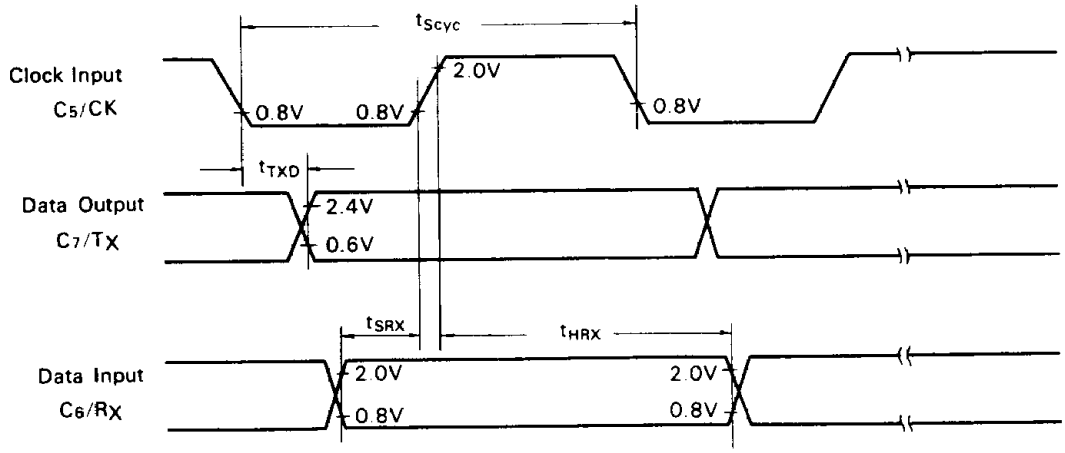
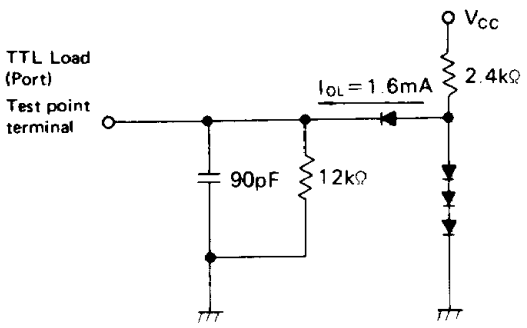


Figure7 SCI Timing(External Clock)



[NOTES] 1. The load capacitance includes stray capacitance caused by the probe, etc.
2. All diodes are 1S2074 Ⓗ.

Figure 8 Test Load

■ DESCRIPTION OF TERMINAL FUNCTIONS

The input and output signals of the MCU are described here.

• V_{CC}, V_{SS}

Voltage is applied to the MCU through these two terminals. V_{CC} is 5.0V ± 10%, while V_{SS} is grounded.

• $\overline{INT}$, $\overline{INT_2}$

External interrupt request inputs to the MCU. For details, refer to "INTERRUPT". The $\overline{INT_2}$ terminal is also used as the port D_6 terminal.

• XTAL, EXTAL

These terminals provide input to the on-chip clock circuit. A crystal oscillator (AT cut, 2.0 to 8.0 MHz) or ceramic filter is connected to the terminal. Refer to "INTERNAL OSCILLATOR" for using these input terminals.

• TIMER

This is an input terminal for event counter. Refer to "TIMER" for details.

• $\overline{RES}$

Used to reset the MCU. Refer to "RESET" for details.

• NUM

This terminal is not for user application. In case of the HD6305X1, this terminal should be connected to V_{CC} through 10kΩ resistance. In case of the HD6305X2, this terminal should be connected to V_{SS}.

• Enable (E)

This output terminal supplies E clock. Output is a single-phase, TTL compatible and 1/4 crystal oscillation frequency or 1/4 external clock frequency. It can drive one TTL load and a 90pF condenser.

• Read/Write (R/$\overline{W}$)

This TTL compatible output signal indicates to peripheral and memory devices whether MCU is in Read ("High"), or in Write ("Low"). The normal standby state is Read ("High"). Its output can drive one TTL load and a 90pF condenser.

• Data Bus ($DATA_0$ ~ $DATA_7$)

This TTL compatible three-state buffer can drive one TTL load and 90pF.

• Address Bus (ADR_0 ~ ADR_{13})

Each terminal is TTL compatible and can drive one TTL load and 90pF.

• Input/Output Terminals (A_0 ~ A_7, B_0 ~ B_7, C_0 ~ C_7)

These 24 terminals consist of three 8-bit I/O ports (A, B, C). Each of them can be used as an input or output terminal on a bit through program control of the data direction register. For details, refer to "I/O PORTS."

• Input Terminals (D_1 ~ D_7)

These seven input-only terminals are TTL or CMOS compatible. Of the port D's, D_6 is also used as $\overline{INT_2}$. If D_6 is used as a port, the $\overline{INT_2}$ interrupt mask bit of the miscellaneous register must be set to "1" to prevent an $\overline{INT_2}$ interrupt from being accidentally accepted.

• $\overline{STBY}$

This terminal is used to place the MCU into the standby mode. With $\overline{STBY}$ at "Low" level, the oscillation stops and the internal condition is reset. For details, refer to "Stand-by Mode."

The terminals described in the following are I/O pins for serial communication interface (SCI). They are also used as ports C_5, C_6 and C_7. For details, refer to "SERIAL COMMUNICATION INTERFACE."

• $\overline{CK}$ (C_5)

Used to input or output clocks for serial operation.

• Rx (C_6)

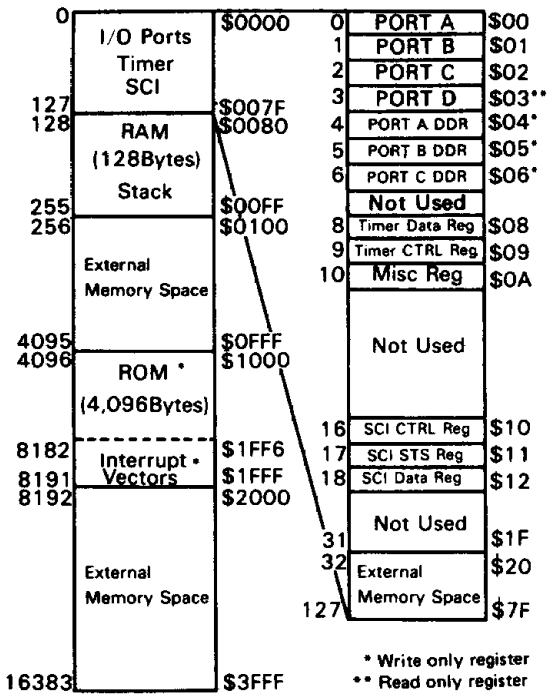
Used to receive serial data.

• Tx (C_7)

Used to transmit serial data.

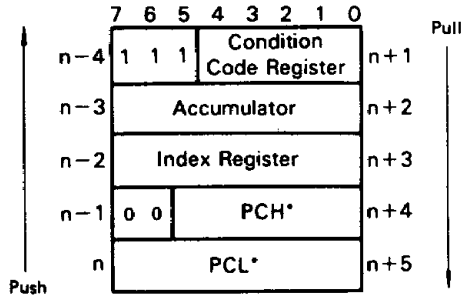
■ MEMORY MAP

The memory map of the MCU is shown in Fig. 9. $1000 ~ $1FFF of the HD6305X2 are external addresses. However, care should be taken to assign vector addresses to $1FF6 ~ $1FFF. During interrupt processing, the contents of the CPU registers are saved into the stack in the sequence shown in Fig. 10. This saving begins with the lower byte (PCL) of the program counter. Then the value of the stack pointer is decremented and the higher byte (PCH) of the program counter, index register (X), accumulator (A) and condition code register (CC) are stacked in that order. In a subroutine call, only the contents of the program counter (PCH and PCL) are stacked.



* ROM area ($1000 ~ $1FFF) in the HD6305X2 is changed into External Memory Space.

Figure 9 Memory Map of MCU



* In a subroutine call, only PCL and PCH are stacked.

Figure 10 Sequence of Interrupt Stacking

■REGISTERS

There are five registers which the programmer can operate.

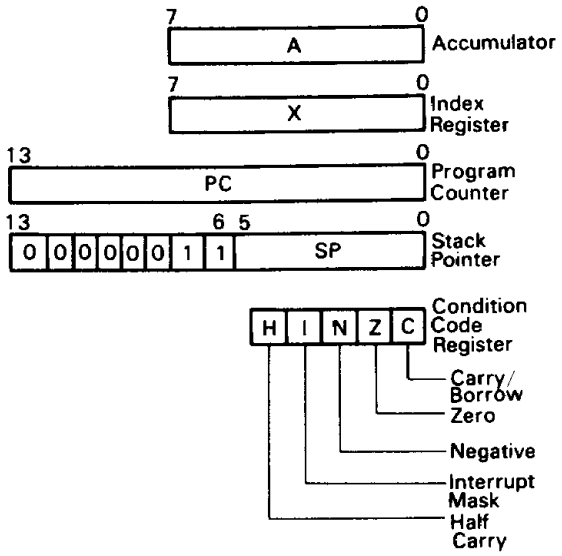


Figure 11 Programming Model

• Accumulator (A)

This accumulator is an ordinary 8-bit register which holds operands or the result of arithmetic operation or data processing.

• Index Register (X)

The index register is an 8-bit register, and is used for index addressing mode. Each of the addresses contained in the register consists of 8 bits which, combined with an offset value, provides an effective address.

In the case of a read/modify/write instruction, the index register can be used like an accumulator to hold operation data or the result of operation.

If not used in the index addressing mode, the register can be used to store data temporarily.

• Program Counter (PC)

The program counter is a 14-bit register that contains the address of the next instruction to be executed.

• Stack Pointer (SP)

The stack pointer is a 14-bit register that indicates the address of the next stacking space. Just after reset, the stack pointer is set at address $00FF. It is decremented when data is pushed, and incremented when pulled. The upper 8 bits of the stack pointer are fixed to 00000011. During the MCU being reset or during a reset stack pointer (RSP) instruction, the pointer is set to address $00FF. Since a subroutine or interrupt can use space up to address $00C1 for stacking, the subroutine can be used up to 31 levels and the interrupt up to 12 levels.

• Condition Code Register (CC)

The condition code register is a 5-bit register, each bit indicating the result of the instruction just executed. The bits can be individually tested by conditional branch instruc-

tions. The CC bits are as follows:

Half Carry (H): Used to indicate that a carry occurred between bits 3 and 4 during an arithmetic operation (ADD, ADC).

Interrupt (I): Setting this bit causes all interrupts, except a software interrupt, to be masked. If an interrupt occurs with the bit I set, it is latched. It will be processed the instant the interrupt mask bit is reset. (More specifically, it will enter the interrupt processing routine after the instruction following the CLI has been executed.)

Negative (N): Used to indicate that the result of the most recent arithmetic operation, logical operation or data processing is negative (bit 7 is logic "1").

Zero (Z): Used to indicate that the result of the most recent arithmetic operation, logical operation or data processing is zero.

Carry/ Borrow (C): Represents a carry or borrow that occurred in the most recent arithmetic operation. This bit is also affected by the Bit Test and Branch instruction and a Rotate instruction.

■ INTERRUPT

There are six different types of interrupt: external interrupts ($\overline{INT}$, $\overline{INT_2}$), internal timer interrupts (TIMER, $TIMER_2$), serial interrupt (SCI) and interrupt by an instruction (SWI).

Of these six interrupts, the $\overline{INT_2}$ and TIMER or the SCI and $TIMER_2$ generate the same vector address, respectively.

When an interrupt occurs, the program in progress stops and the then CPU status is saved onto the stack. And then, the interrupt mask bit (I) of the condition code register is set and the start address of the interrupt processing routine is obtained from a particular interrupt vector address. Then the interrupt routine starts from the start address. System can exit from the interrupt routine by an RTI instruction. When this instruction is executed, the CPU status before the interrupt (saved onto the stack) is pulled and the CPU restarts the sequence with the instruction next to the one at which the interrupt occurred. Table 1 lists the priority of interrupts and their vector addresses.

Table 1 Priority of Interrupts

Interrupt	Priority	Vector Address
$\overline{RES}$	1	$1FFE, $1FFF
SWI	2	$1FFC, $1FFD
$\overline{INT}$	3	$1FFA, $1FFB
TIMER/$\overline{INT_2}$	4	$1FF8, $1FF9
SCI/$TIMER_2$	5	$1FF6, $1FF7

A flowchart of the interrupt sequence is shown in Fig. 12. A block diagram of the interrupt request source is shown in Fig. 13.

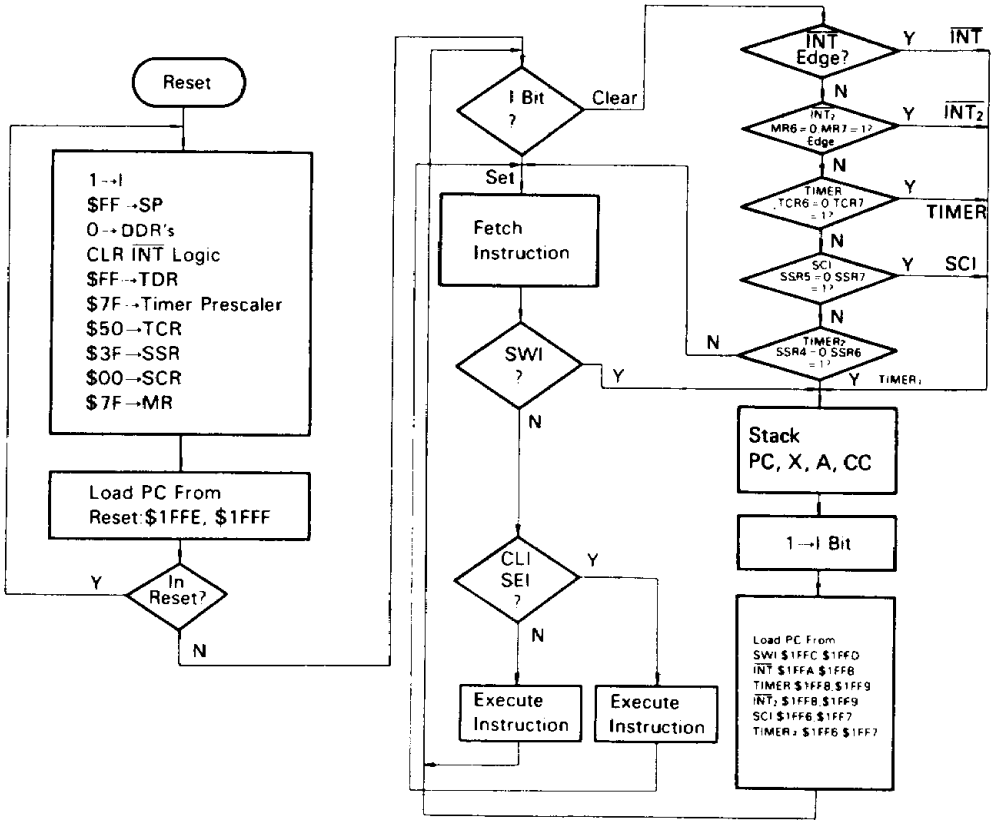


Figure 12 Interrupt Flow Chart

In the block diagram, both the external interrupts $\overline{INT}$ and $\overline{INT_2}$ are edge trigger inputs. At the falling edge of each input, an interrupt request is generated and latched. The $\overline{INT}$ interrupt request is automatically cleared if jumping is made to the $\overline{INT}$ processing routine. Meanwhile, the $\overline{INT_2}$ request is cleared if "0" is written in bit 7 of the miscellaneous register.

For the external interrupts ($\overline{INT}$, $\overline{INT_2}$), internal timer interrupts (TIMER, $TIMER_2$) and serial interrupt (SCI), each interrupt request is held, but not processed, if the I bit of the condition code register is set. Immediately after the I bit is cleared, the corresponding interrupt processing starts according to the priority.

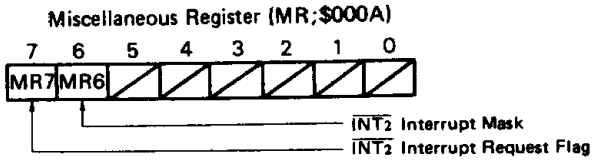
The $\overline{INT_2}$ interrupt can be masked by setting bit 6 of the miscellaneous register; the TIMER interrupt by setting bit 6 of the timer control register; the SCI interrupt by setting bit 5 of the serial status register; and the $TIMER_2$ interrupt by setting bit 4 of the serial status register.

The status of the $\overline{INT}$ terminal can be tested by a BIL or BIH instruction. The $\overline{INT}$ falling edge detector circuit and its latching circuit are independent of testing by these instructions. This is also true with the status of the $\overline{INT_2}$ terminal.

• Miscellaneous Register (MR; $000A)

The interrupt vector address for the external interrupt $\overline{INT_2}$ is the same as that for the TIMER interrupt, as shown in Table 1. For this reason, a special register called the miscellaneous register (MR; $000A) is available to control the $\overline{INT_2}$ interrupts.

Bit 7 of this register is the $\overline{INT_2}$ interrupt request flag. When the falling edge is detected at the $\overline{INT_2}$ terminal, "1" is set in bit 7. Then the software in the interrupt routine (vector addresses: $1FF8, $1FF9) checks bit 7 to see if it is $\overline{INT_2}$ interrupt. Bit 7 can be reset by software.



Miscellaneous Register (MR; $000A)

Bit 6 is the $\overline{INT_2}$ interrupt mask bit. If this bit is set to "1", then the $\overline{INT_2}$ interrupt is disabled. Both read and write are possible with bit 7 but "1" cannot be written in this bit by software. This means that an interrupt request by software is impossible.

When reset, bit 7 is cleared to "0" and bit 6 is set to "1"

■ TIMER

Figure 14 shows a MCU timer block diagram. The timer data register is loaded by software and, upon receipt of a clock input, begins to count down. When the timer data

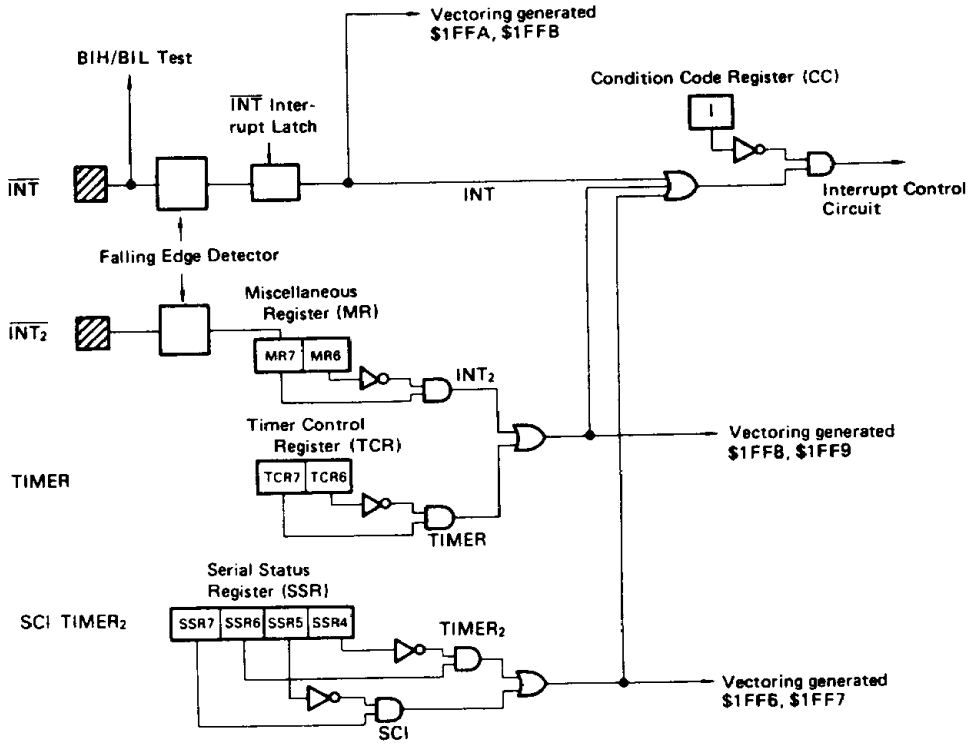


Figure 13 Interrupt Request Generation Circuitry

register (TDR) becomes "0", the timer interrupt request bit (bit 7) in the timer control register is set. In response to the interrupt request, the CPU saves its status into the stack and fetches timer interrupt routine address from addresses $1FF8 and $1FF9 and execute the interrupt routine. The timer interrupt can be masked by setting the timer interrupt mask bit (bit 6) in the timer control register. The mask bit (I) in the condition code register can also mask the timer interrupt.

The source clock to the timer can be either an external signal from the timer input terminal or the internal E signal (the oscillator clock divided by 4). If the E signal is used as the source, the clock input can be gated by the input to the timer input terminal.

Once the timer count has reached "0", it starts counting down with "$FF". The count can be monitored whenever desired by reading the timer data register. This permits the program to know the length of time having passed after the occurrence of a timer interrupt, without disturbing the contents of the counter.

When the MCU is reset, both the prescaler and counter are initialized to logic "1". The timer interrupt request bit (bit 7) then is cleared and the timer interrupt mask bit (bit 6) is set.

To clear the timer interrupt request bit (bit 7), it is necessary to write "0" in that bit.

TCR7	Timer interrupt request
0	Absent
1	Present

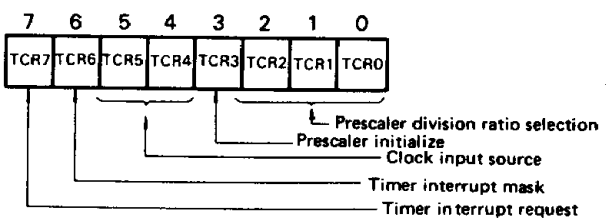
TCR6	Timer interrupt mask
0	Enabled
1	Disabled

• Timer Control Register (TCR; $0009)

Selection of a clock source, selection of a prescaler frequency division ratio, and a timer interrupt can be controlled by the timer control register (TCR; $0009).

For the selection of a clock source, any one of the four modes (see Table 2) can be selected by bits 5 and 4 of the timer control register (TCR).

Timer Control Register (TCR; $0009)



After reset, the TCR is initialized to "E under timer terminal control" (bit 5 = 0, bit 4 = 1). If the timer terminal is "1", the counter starts counting down with "$FF" immediately after reset.

When "1" is written in bit 3, the prescaler is initialized. This bit always shows "0" when read.

Table 2 Clock Source Selection

TCR		Clock input source
Bit 5	Bit 4	
0	0	Internal clock E
0	1	E under timer terminal control
1	0	No clock input (counting stopped)
1	1	Event input from timer terminal

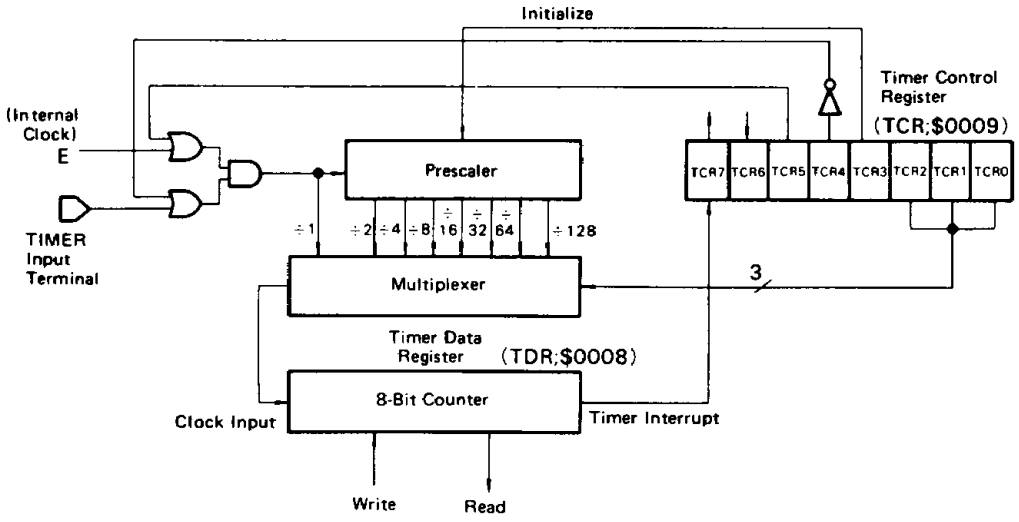


Figure 14 Timer Block Diagram

A prescaler division ratio is selected by the combination of three bits (bits 0, 1 and 2) of the timer control register (see Table 3). There are eight different division ratios: ÷1, ÷2, ÷4, ÷8, ÷16, ÷32, ÷64 and ÷128. After reset, the TCR is set to the ÷1 mode.

Table 3 Prescaler Division Ratio Selection

TCR			Prescaler division ratio
Bit 2	Bit 1	Bit 0	
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

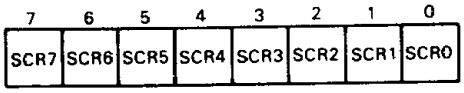
A timer interrupt is enabled when the timer interrupt mask bit is "0", and disabled when the bit is "1". When a timer interrupt occurs, "1" is set in the timer interrupt request bit. This bit can be cleared by writing "0" in that bit.

■SERIAL COMMUNICATION INTERFACE (SCI)

This interface is used for serial transmission or reception of 8-bit data. Sixteen transfer rates are available in the range from 1 μs to approx. 32 ms (for oscillation at 4 MHz).

The SCI consists of three registers, one octal counter and one prescaler. (See Fig. 15.) SCI communicates with the CPU via the data bus, and with the outside world through bits 5, 6 and 7 of port C. Described below are the operations of each register and data transfer.

●SCI Control Register (SCR; $0010)



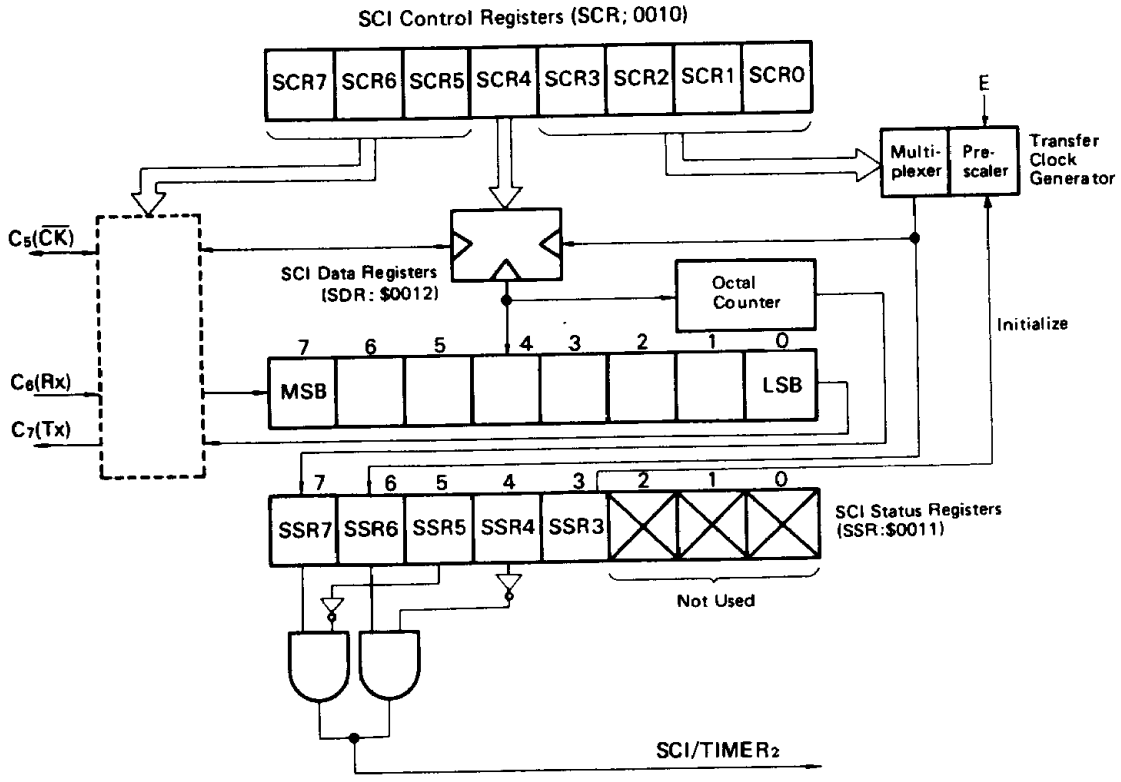


Figure 15 SCI Block Diagram

SCR7	C_7 terminal
0	Used as I/O terminal (by DDR).
1	Serial data output (DDR output)

SCR6	C_6 terminal
0	Used as I/O terminal (by DDR).
1	Serial data input (DDR input)

SCR5	SCR4	Clock source	C_5 terminal
0	0	–	Used as I/O terminal (by DDR).
0	1	–	
1	0	Internal	Clock output (DDR output)
1	1	External	Clock input (DDR input)

Bit 7 (SCR7)

When this bit is set, the DDR corresponding to the C_7 becomes "1" and this terminal serves for output of SCI data. After reset, the bit is cleared to "0".

Bit 6 (SCR6)

When this bit is set, the DDR corresponding to the C_6 becomes "0" and this terminal serves for input of SCI data. After reset, the bit is cleared to "0".

Bits 5 and 4 (SCR5, SCR4)

These bits are used to select a clock source. After reset, the bits are cleared to "0".

Bits 3 ~ 0 (SCR3 ~ SCR0)

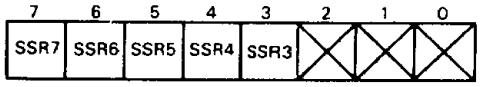
These bits are used to select a transfer clock rate. After reset, the bits are cleared to "0".

SCR3	SCR2	SCR1	SCR0	Transfer clock rate	
				4.00 MHz	4.194 MHz
0	0	0	0	1 μs	0.95 μs
0	0	0	1	2 μs	1.91 μs
0	0	1	0	4 μs	3.82 μs
0	0	1	1	8 μs	7.64 μs
≀	≀	≀	≀	≀	≀
1	1	1	1	32768 μs	1/32 s

•SCI Data Register (SDR; $0012)

A serial-parallel conversion register that is used for transfer of data.

•SCI Status Register (SSR; $0011)



Bit 7 (SSR7)

Bit 7 is the SCI interrupt request bit which is set upon completion of transmitting or receiving 8-bit data. It is cleared when reset or data is written to or read from the SCI data register with the SCR5="1". The bit can also be cleared by writing "0" in it.

Bit 6 (SSR6)

Bit 6 is the $TIMER_2$ interrupt request bit. $TIMER_2$ is used commonly with the serial clock generator, and SSR6 is set each time the internal transfer clock falls. When reset, the bit is cleared. It also be cleared by writing "0" in it. (For details, see $TIMER_2$.)

Bit 5 (SSR5)

Bit 5 is the SCI interrupt mask bit which can be set or cleared by software. When it is "1", the SCI interrupt (SSR7) is masked. When reset, it is set to "1".

Bit 4 (SSR4)

Bit 4 is the $TIMER_2$ interrupt mask bit which can be set or cleared by software. When the bit is "1", the $TIMER_2$ interrupt (SSR6) is masked. When reset, it is set to "1".

Bit 3 (SSR3)

When "1" is written in this bit, the prescaler of the transfer clock generator is initialized. When read, the bit always is "0".

Bits 2 ~ 0

Not used.

SSR7	SCI interrupt request
0	Absent
1	Present

SSR6	$TIMER_2$ interrupt request
0	Absent
1	Present

SSR5	SCI interrupt mask
0	Enabled
1	Disabled

SSR4	$TIMER_2$ interrupt mask
0	Enabled
1	Disabled

• Data Transmission

By writing the desired control bits into the SCI control registers, a transfer rate and a source of transfer clock are determined and bits 7 and 5 of port C are set at the serial data output terminal and the serial clock terminal, respectively. The transmit data should be stored from the accumulator or index register into the SCI data register. The data written in the SCI data register is output from the C_7/Tx terminal, starting with the LSB, synchronously with the falling edge of the serial clock. (See Fig. 16.) When 8 bit of

data have been transmitted, the interrupt request bit is set in bit 7 of the SCI status register with the rising edge of the last serial clock. This request can be masked by setting bit 5 of the SCI status register. Once the data has been sent, the 8th bit data (MSB) stays at the C_7/Tx terminal. If an external clock source has been selected, the transfer rate determined by bits 0 ~ 3 of the SCI control register is ignored, and the C_5/$\overline{CK}$ terminal is set as input. If the internal clock has been selected, the C_5/$\overline{CK}$ terminal is set as output and clocks are output at the transfer rate selected by bits 0 ~ 3 of the SCI control register.

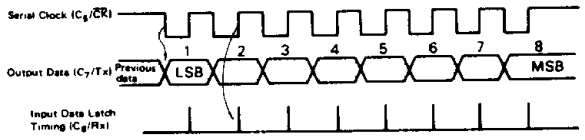


Figure 16 SCI Timing Chart

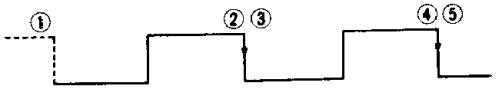
• Data Reception

By writing the desired control bits into the SCI control register, a transfer rate and a source of transfer clock are determined and bit 6 and 5 of port C are set at the serial data input terminal and the serial clock terminal, respectively. Then dummy-writing or -reading the SCI data register, the system is ready for receiving data. (This procedure is not needed after reading subsequent received data. It must be taken after reset and after not reading subsequent received data.

The data from the C_6/Rx terminal is input to the SCI data register synchronously with the rising edge of the serial clock (see Fig. 16). When 8 bits of data have been received, the interrupt request bit is set in bit 7 of the SCI status register. This request can be masked by setting bit 5 of the SCI status register. If an external clock source have been selected, the transfer rate determined by bits 0 ~ 3 of the SCI control register is ignored and the data is received synchronously with the clock from the C_5/$\overline{CK}$ terminal. If the internal clock has been selected, the C_5/$\overline{CK}$ terminal is set as output and clocks are output at the transfer rate selected by bits 0 ~ 3 of the SCI control register.

• TIMER$_2$

The SCI transfer clock generator can be used as a timer. The clock selected by bits 3 ~ 0 of the SCI control register (4 μs ~ approx. 32 ms (for oscillation at 4 MHz)) is input to bit 6 of the SCI status register and the TIMER$_2$ interrupt request bit is set at each falling edge of the clock. Since interrupt requests occur periodically, TIMER$_2$ can be used as a reload counter or clock.



① : Transfer clock generator is reset and mask bit (bit 4 of SCI status register) is cleared.
②, ④ : TIMER$_2$ interrupt request
③, ⑤ : TIMER$_2$ interrupt request bit cleared

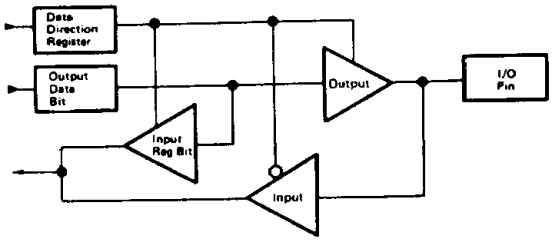
TIMER$_2$ is commonly used with the SCI transfer clock generator. If wanting to use TIMER$_2$ independently of the SCI, specify "External" (SCR5 = 1, SCR4 = 1) as the SCI clock source.

If "Internal" is selected as the clock source, reading or writing the SDR causes the prescaler of the transfer clock generator to be initialized.

■ I/O PORTS

There are 24 input/output terminals (ports A, B, C). Each I/O terminal can be selected for either input or output by the data direction register. More specifically, an I/O port will be input if "0" is written in the data direction register, and output if "1" is written in the data direction register. Port A, B or C reads latched data if it has been programmed as output, even with the output level being fluctuated by the output load. (See Fig. 17.)

When reset, the data direction register and data register go to "0" and all the input/output terminals are used as input.



Bit of data direction register	Bit of output data	Status of output	Input to CPU
1	0	0	0
1	1	1	1
0	X	3-state	Pin

Figure 17 Input/Output Port Diagram

Seven input-only terminals are available (port D). Writing to an input terminal is invalid.

All input/output terminals and input terminals are TTL compatible and CMOS compatible in respect of both input and output.

If I/O ports or input ports are not used, they should be connected to V_{SS} via resistors. With none connected to these terminals, there is the possibility of power being consumed despite that they are not used.

■ RESET

The MCU can be reset either by external reset input ($\overline{RES}$) or power-on reset. (See Fig. 18.) On power up, the reset input must be held "Low" for at least t_{OSC} to assure that the internal oscillator is stabilized. A sufficient time of delay can be obtained by connecting a capacitance to the $\overline{RES}$ input as shown in Fig. 19.

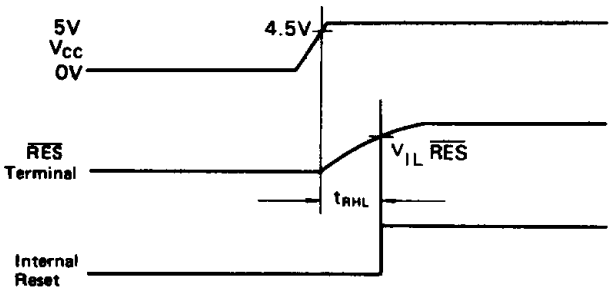


Figure 18 Power On and Reset Timing

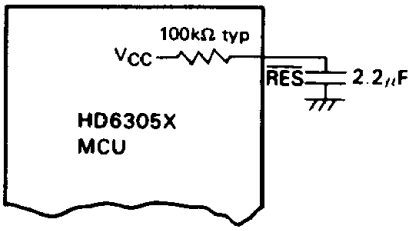


Figure 19 Input Reset Delay Circuit

■INTERNAL OSCILLATOR

The internal oscillator circuit is designed to meet the requirement for minimum external configurations. It can be driven by connecting a crystal (AT cut 2.0 ~ 8.0MHz) or ceramic oscillator between pins 5 and 6 depending on the required oscillation frequency stability.

Three different terminal connections are shown in Fig. 20. Figs. 21 and 22 illustrate the specifications and typical arrangement of the crystal, respectively.

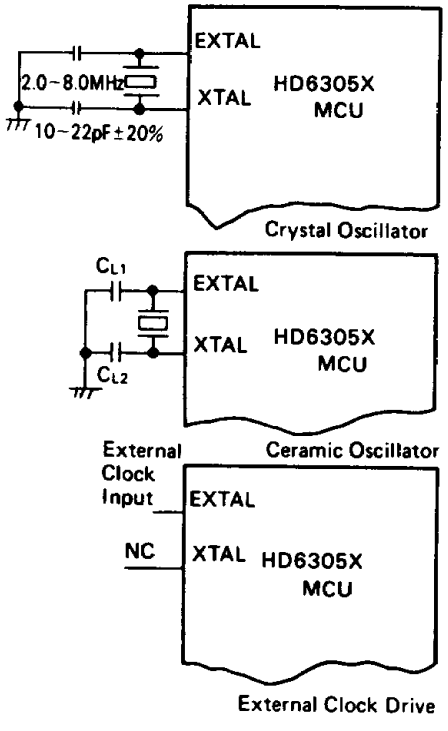


Figure 20 Internal Oscillator Circuit

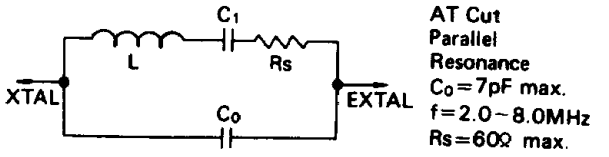
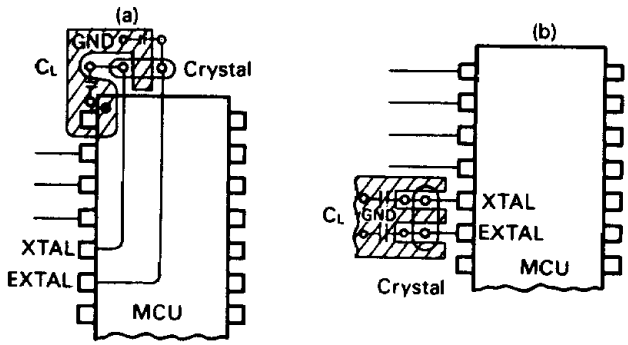


Figure 21 Parameters of Crystal



[NOTE] Use as short wirings as possible for connection of the crystal with the EXTAL and XTAL terminals. Do not allow these wirings to cross others.

Figure 22 Typical Crystal Arrangement

■LOW POWER DISSIPATION MODE

The HD6305X has three low power dissipation modes: wait, stop and standby.

• Wait Mode

When WAIT instruction being executed, the MCU enters into the wait mode. In this mode, the oscillator stays active but the internal clock stops. The CPU stops but the peripheral functions – the timer and the serial communication interface – stay active. (NOTE: Once the system has entered the wait mode, the serial communication interface can no longer be retriggered.) In the wait mode, the registers, RAM and I/O terminals hold their condition just before entering into the wait mode.

The escape from this mode can be done by interrupt ($\overline{INT}$, TIMER/$\overline{INT_2}$ or SCI/TIMER$_2$), $\overline{RES}$ or $\overline{STBY}$. The $\overline{RES}$ resets the MCU and the $\overline{STBY}$ brings it into the standby mode. (This will be mentioned later.)

When interrupt is requested to the CPU and accepted, the wait mode escapes, then the CPU is brought to the operation mode and vectors to the interrupt routine. If the interrupt is masked by the I bit of the condition code register, after releasing from the wait mode the MCU executes the instruction next to the WAIT. If an interrupt other than the $\overline{INT}$ (i.e., TIMER/$\overline{INT_2}$ or SCI/TIMER$_2$) is masked by the timer control

register, miscellaneous register or serial status register, there is no interrupt request to the CPU, so the wait mode cannot be released.

Fig. 23 shows a flowchart for the wait function.

• Stop Mode

When STOP instruction being executed, MCU enters into the stop mode. In this mode, the oscillator stops and the CPU and peripheral functions become inactive but the RAM, registers and I/O terminals hold their condition just before entering into the stop mode.

The escape from this mode can be done by an external interrupt ($\overline{INT}$ or $\overline{INT_2}$), $\overline{RES}$ or $\overline{STBY}$. The $\overline{RES}$ resets the MCU and the $\overline{STBY}$ brings into the standby mode.

When interrupt is requested to the CPU and accepted, the stop mode escapes, then the CPU is brought to the operation mode and vectors to the interrupt routine. If the interrupt is masked by the I bit of the condition code register, after releasing from the stop mode, the MCU executes the instruction next to the STOP. If the $\overline{INT_2}$ interrupt is masked by the miscellaneous register, there is no interrupt request to the MCU, so the stop mode cannot be released.

Fig. 24 shows a flowchart for the stop function. Fig. 25 shows a timing chart of return to the operation mode from the stop mode.

For releasing from the stop mode by an interrupt, oscillation starts upon input of the interrupt and, after the internal delay time for stabilized oscillation, the CPU becomes active. For restarting by $\overline{RES}$, oscillation starts when the $\overline{RES}$ goes "0" and the CPU restarts when the $\overline{RES}$ goes "1". The duration of $\overline{RES}$="0" must exceed t_{osc} to assure stabilized oscillation.

• Standby Mode

The MCU enters into the standby mode when the $\overline{STBY}$ terminal goes "Low". In this mode, all operations stop and the internal condition is reset but the contents of the RAM are hold. The I/O terminals turn to high-impedance state. The standby mode should escape by bringing $\overline{STBY}$ "High". The CPU must be restarted by reset. The timing of input signals at the $\overline{RES}$ and $\overline{STBY}$ terminals is shown in Fig. 26.

Table 4 lists the status of each parts of the MCU in each low power dissipation modes. Transitions between each mode are shown in Fig. 27.

(Note)

When I bit of condition code register is "1" and interrupt ($\overline{INT}$, TIMER/$\overline{INT_2}$, SCI/TIMER$_2$) is held, MCU does not enter WAIT mode by the execution of WAIT instruction.

In that case, after the 4 dummy cycles MCU executes the next instruction.

In the same way, when external interrupts ($\overline{INT}$, $\overline{INT_2}$) are held at the bit I set, MCU does not enter STOP mode by the execution of STOP instruction. In that case, also, MCU executes the next instruction after the 4 dummy cycles.

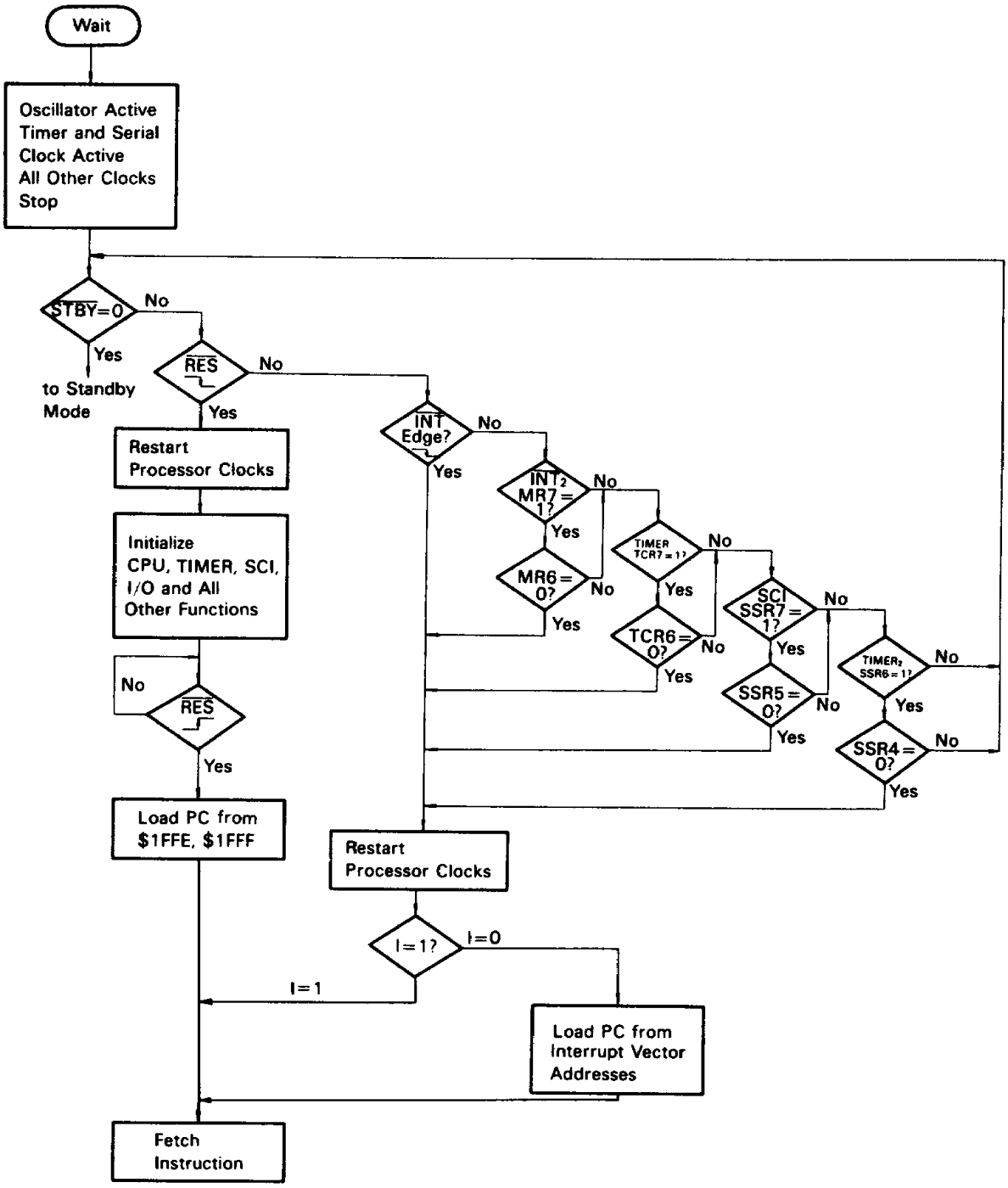


Figure 23 Wait Mode Flow Chart

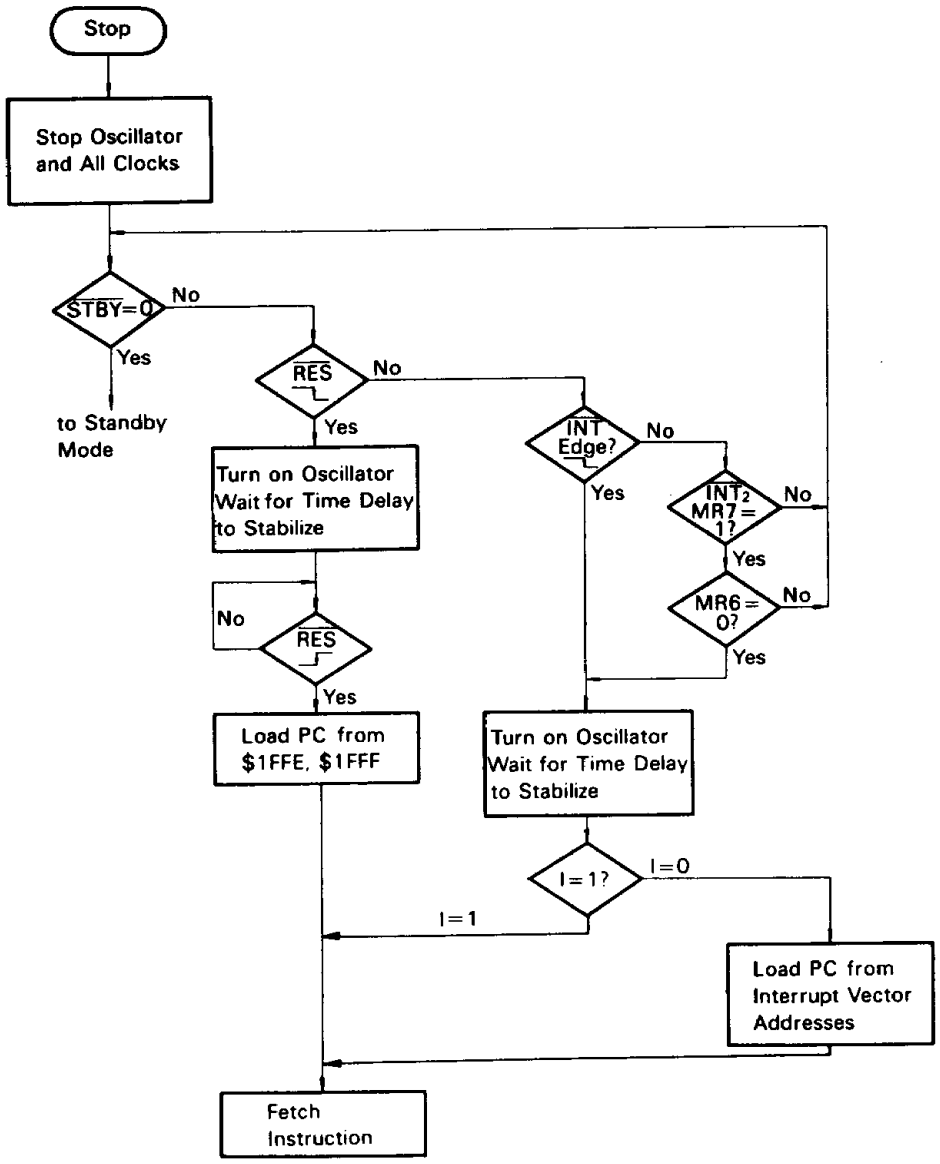
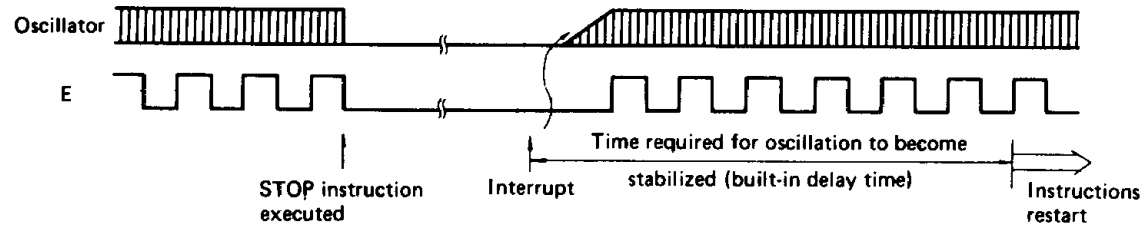
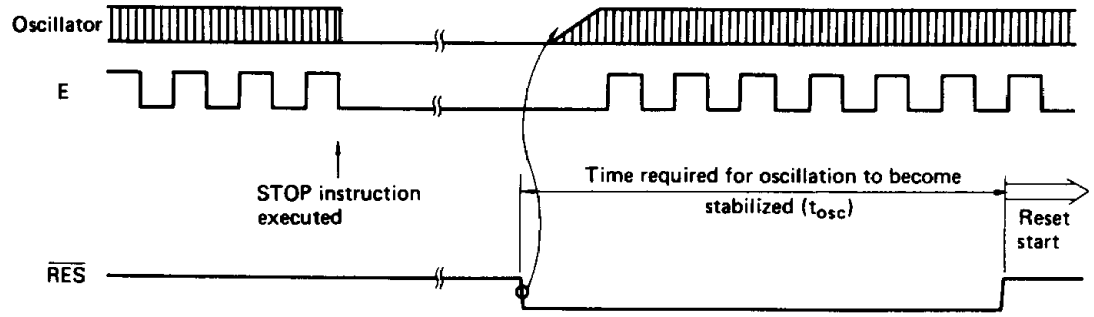


Figure 24 Stop Mode Flow Chart



(a) Restart by Interrupt



(b) Restart by Reset

Figure 25 Timing Chart of Releasing from Stop Mode

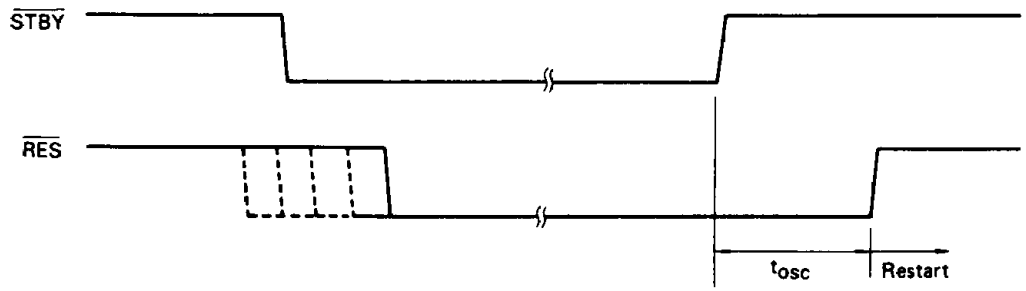


Figure 26 Timing Chart of Releasing from Standby Mode

Table 4 Status of Each Part of MCU in Low Power Dissipation Modes

Mode	Start		Condition						Escape
			Oscillator	CPU	Timer, Serial	Register	RAM	I/O terminal	
WAIT	Software	WAIT instruction	Active	Stop	Active	Keep	Keep	Keep	$\overline{STBY}$, $\overline{RES}$, $\overline{INT}$, $\overline{INT_2}$, each interrupt request of TIMER, $TIMER_2$, SCI
STOP		STOP instruction	Stop	Stop	Stop	Keep	Keep	Keep	$\overline{STBY}$, $\overline{RES}$, $\overline{INT}$, $\overline{INT_2}$
Standby	Hardware	$\overline{STBY}$="Low"	Stop	Stop	Stop	Reset	Keep	High impedance	$\overline{STBY}$="High"

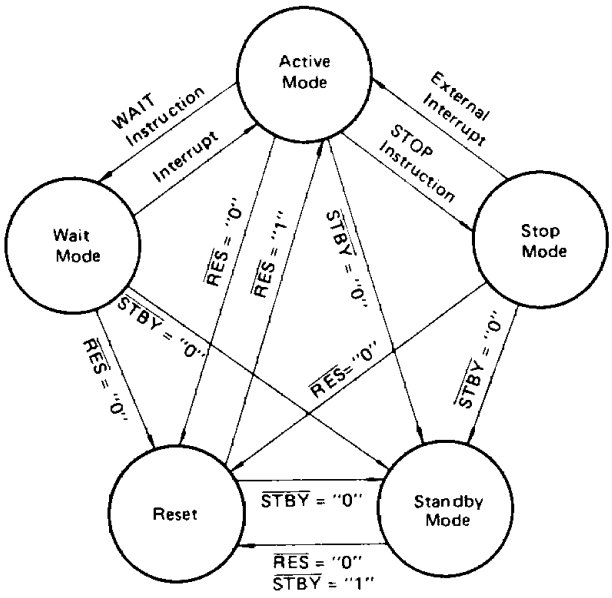


Figure 27 Transitions among Active Mode, Wait Mode, Stop Mode, Standby Mode and Reset

■BIT MANIPULATION

The MCU can use a single instruction (BSET or BCLR) to set or clear one bit of the RAM or an I/O port (except the write-only registers such as the data direction register). Every bit of memory or I/O within page 0 ($00 ~ $FF) can be tested by the BRSET or BRCLR instruction; depending on the result of the test, the program can branch to required destinations. Since bits in the RAM, or I/O can be manipulated, the user may use a bit within the RAM as a flag or handle a single I/O bit as an independent I/O terminal. Fig. 28 shows an example of bit manipulation and the validity of test instructions. In the example, the program is configured assuming that bit 0 of port A is connected to a zero cross detector circuit and bit 1 of the same port to the trigger of a triac.

The program shown can activate the triac within a time of 10μs from zero-crossing through the use of only 7 bytes on the ROM. The on-chip timer provides a required time of delay and pulse width modulation of power is also possible.

```
        :
SELF 1. BRCLR 0, PORT A, SELF 1
        BSET 1, PORT A
        BCLR 1, PORT A
        :
```

Figure 28 Example of Bit Manipulation

■ADDRESSING MODES

Ten different addressing modes are available to the MCU.

• Immediate

See Fig. 29. The immediate addressing mode provides access to a constant which does not vary during execution of the program.

This access requires an instruction length of 2 bytes. The effective address (EA) is PC and the operand is fetched from the byte that follows the operation code.

• Direct

See Fig. 30. In the direct addressing mode, the address of the operand is contained in the 2nd byte of the instruction. The user can gain direct access to memory up to the lower 255th address. All RAM and I/O registers are on page 0 of address space so that the direct addressing mode may be utilized.

• Extended

See Fig. 31. The extended addressing is used for referencing to all addresses of memory. The EA is the contents of the 2 bytes that follow the operation code. An extended addressing instruction requires a length of 3 bytes.

• Relative

See Fig. 32. The relative addressing mode is used with branch instructions only. When a branch occurs, the program counter is loaded with the contents of the byte following the operation code. EA = (PC) + 2 + Rel., where Rel. indicates a signed 8-bit data following the operation code. If no branch occurs, Rel. = 0. When a branch occurs, the program jumps to any byte in the range +129 to -127. A branch instruction requires a length of 2 bytes.

• Indexed (No Offset)

See Fig. 33. The indexed addressing mode allows access up to the lower 255th address of memory. In this mode, an instruction requires a length of one byte. The EA is the contents of the index register.

• Indexed (8-bit Offset)

See Fig. 34. The EA is the contents of the byte following the operation code, plus the contents of the index register. This mode allows access up to the lower 511th address of memory. Each instruction when used in the index addressing mode (8-bit offset) requires a length of 2 bytes.

• Indexed (16-bit Offset)

See Fig. 35. The contents of the 2 bytes following the operation code are added to content of the index register to compute the value of EA. In this mode, the complete memory can be accessed. When used in the indexed addressing mode (16-bit offset), an instruction must be 3 bytes long.

• Bit Set/Clear

See Fig. 36. This addressing mode is applied to the BSET and BCLR instructions that can set or clear any bit on page 0. The lower 3 bits of the operation code specify the bit to be set or cleared. The byte that follows the operation code indicates an address within page 0.

• Bit Test and Branch

See Fig. 37. This addressing mode is applied to the BRSET and BRCLR instructions that can test any bit within page 0 and can be branched in the relative addressing mode. The byte to be tested is addressed depending on the contents of the byte following the operation code. Individual bits within the byte to be tested are specified by the lower 3 bits of the operation code. The 3rd byte represents a relative value which will be added to the program counter when a branch condition is established. Each of these instructions should be 3 bytes long. The value of the test bit is written in the carry bit of the condition code register.

• Implied

See Fig. 38. This mode involves no EA. All information needed for execution of an instruction is contained in the operation code. Direct manipulation on the accumulator and index register is included in the implied addressing mode. Other instructions such as SWI and RTI are also used in this mode. All instructions used in the implied addressing mode should have a length of one byte.

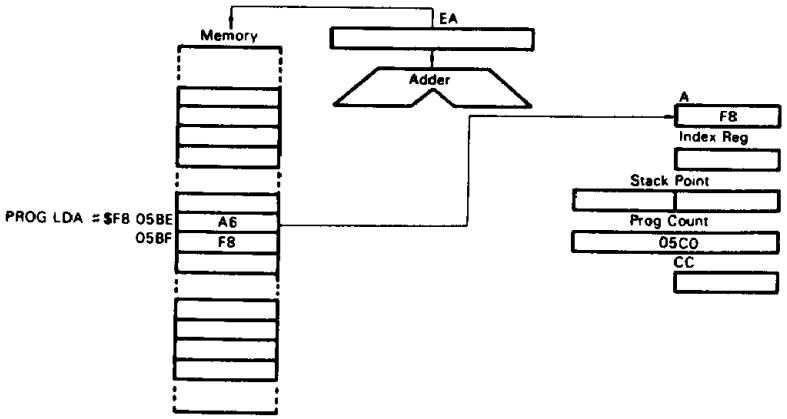


Figure 29 Example of Immediate Addressing

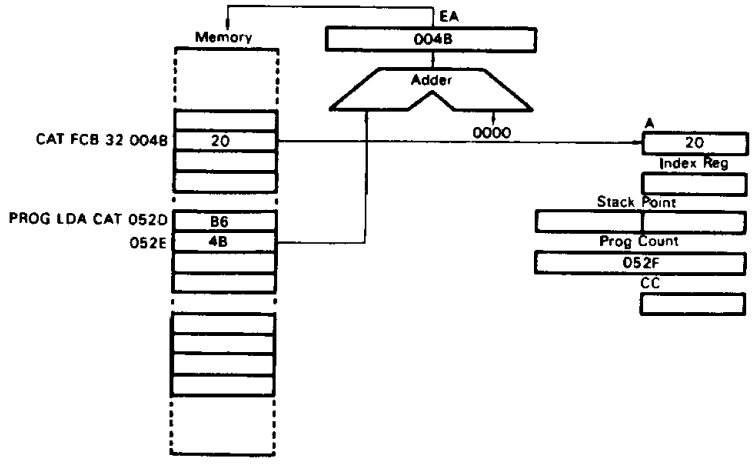


Figure 30 Example of Direct Addressing

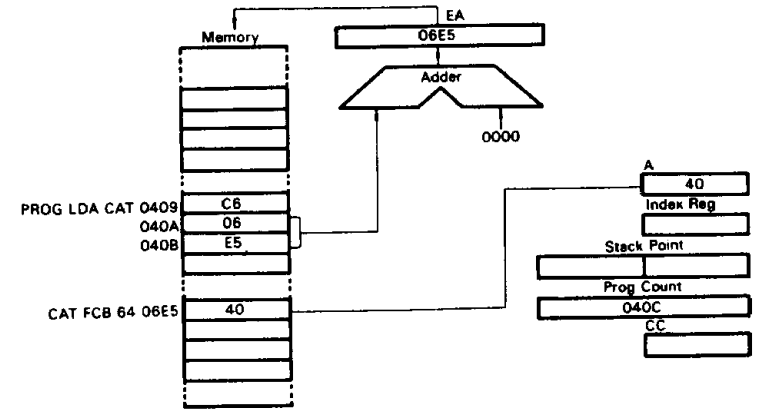


Figure 31 Example of Extended Addressing

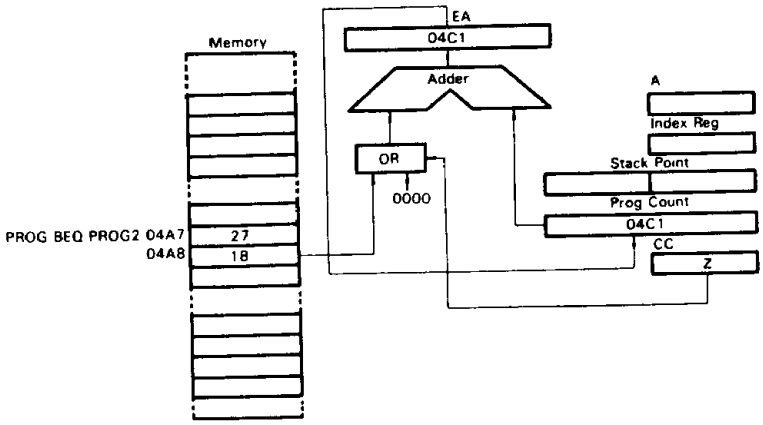


Figure 32 Example of Relative Addressing

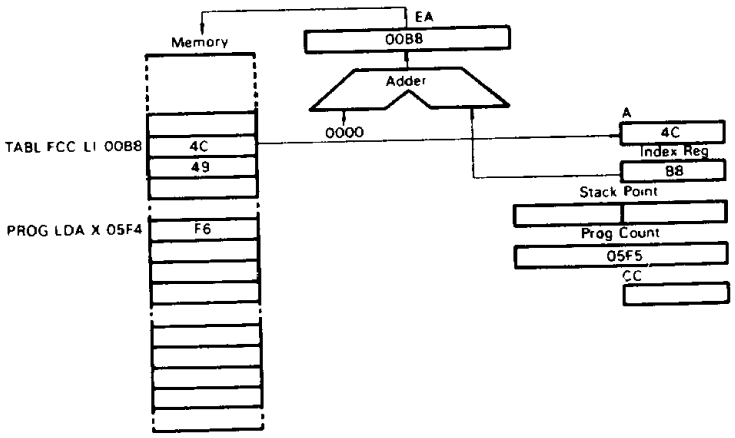


Figure 33 Example of Indexed (No Offset) Addressing

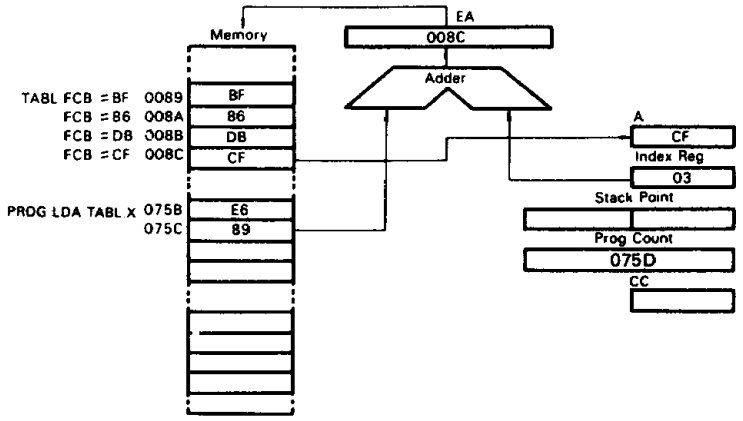


Figure 34 Example of Index (8-bit Offset) Addressing

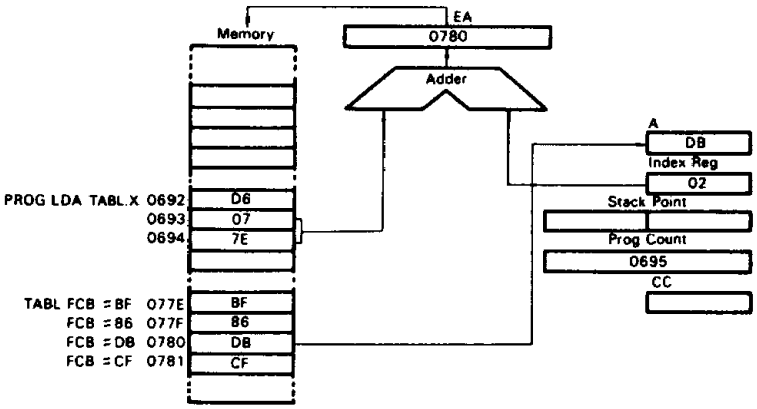


Figure 35 Example of Index (16-bit Offset) Addressing

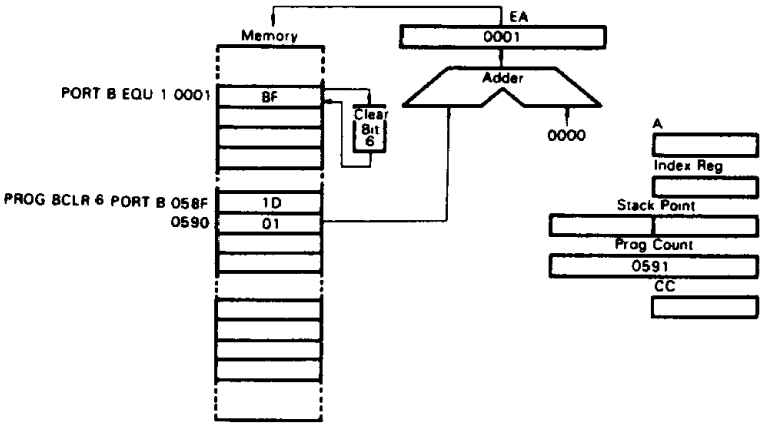


Figure 36 Example of Bit Set/Clear Addressing

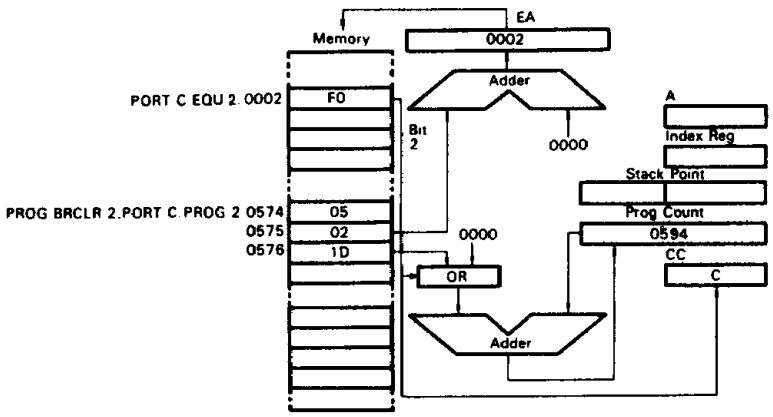


Figure 37 Example of Bit Test and Branch Addressing

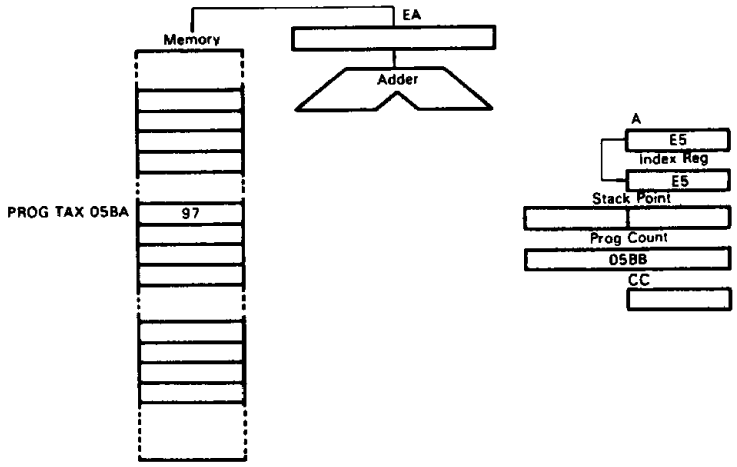


Figure 38 Example of Implied Addressing

■ INSTRUCTION SET

There are 62 basic instructions available to the HD6305X MCU. They can be classified into five categories: register/memory, read/modify/write, branch, bit manipulation, and control. The details of each instruction are described in Tables 5 through 11.

• Register/Memory Instructions

Most of these instructions use two operands. One operand is either an accumulator or index register. The other is derived from memory using one of the addressing modes used on the HD6305X MCU. There is no register operand in the unconditional jump instruction (JMP) and the subroutine jump instruction (JSR). See Table 5.

• Read/Modify/Write Instructions

These instructions read a memory or register, then modify or test its contents, and write the modified value into the memory or register. Zero test instruction (TST) does not write data, and is handled as an exception in the read/modify/write group. See Table 6.

• Branch Instructions

A branch instruction branches from the program sequence in progress if a particular condition is established. See Table 7.

• Bit Manipulation Instructions

These instructions can be used with any bit located up to the lower 255th address of memory. Two groups are available; one for setting or clearing and the other for bit testing and branching. See Table 8.

• Control Instructions

The control instructions control the operation of the MCU which is executing a program. See Table 9.

• List of Instructions in Alphabetical Order

Table 10 lists all the instructions used on the HD6305X MCU in the alphabetical order.

• Operation Code Map

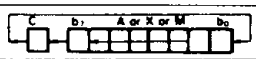
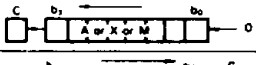
Table 11 shows the operation code map for the instructions used on the MCU.

Table 5 Register/Memory Instructions

Operations	Mnemonic	Addressing Modes: Immediate OP	#	~	Direct OP	#	~	Extended OP	#	~	Indexed (No Offset) OP	#	~	Indexed (8-Bit Offset) OP	#	~	Indexed (16-Bit Offset) OP	#	~	Boolean/ Arithmetic Operation	Condition Code H	I	N	Z	C
Load A from Memory	LDA	A6	2	2	B6	2	3	C6	3	4	F6	1	3	E6	2	4	D6	3	5	M→A	•	•	^	^	•
Load X from Memory	LDX	AE	2	2	BE	2	3	CE	3	4	FE	1	3	EE	2	4	DE	3	5	M→X	•	•	^	^	•
Store A in Memory	STA	–	–	–	B7	2	3	C7	3	4	F7	1	4	E7	2	4	D7	3	5	A→M	•	•	^	^	•
Store X in Memory	STX	–	–	–	BF	2	3	CF	3	4	FF	1	4	EF	2	4	DF	3	5	X→M	•	•	^	^	•
Add Memory to A	ADD	AB	2	2	BB	2	3	CB	3	4	FB	1	3	EB	2	4	DB	3	5	A+M→A	^	•	^	^	^
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	A+M+C→A	^	•	^	^	^
Subtract Memory	SUB	A0	2	2	B0	2	3	C0	3	4	F0	1	3	E0	2	4	D0	3	5	A−M→A	•	•	^	^	^
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	A−M−C→A	•	•	^	^	^
AND Memory to A	AND	A4	2	2	B4	2	3	C4	3	4	F4	1	3	E4	2	4	D4	3	5	A · M→A	•	•	^	^	•
OR Memory with A	ORA	AA	2	2	BA	2	3	CA	3	4	FA	1	3	EA	2	4	DA	3	5	A+M→A	•	•	^	^	•
Exclusive OR Memory with A	EOR	A8	2	2	B8	2	3	C8	3	4	F8	1	3	E8	2	4	D8	3	5	A⊕M→A	•	•	^	^	•
Arithmetic Compare A with Memory	CMP	A1	2	2	B1	2	3	C1	3	4	F1	1	3	E1	2	4	D1	3	5	A−M	•	•	^	^	^
Arithmetic Compare X with Memory	CPX	A3	2	2	B3	2	3	C3	3	4	F3	1	3	E3	2	4	D3	3	5	X−M	•	•	^	^	^
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	A · M	•	•	^	^	•
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	5	DD	3	6		•	•	•	•	•

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

Table 6 Read/Modify/Write Instructions

Operations	Mnemonic	Addressing Modes: Implied(A) OP	#	~	Implied(X) OP	#	~	Direct OP	#	~	Indexed (No Offset) OP	#	~	Indexed (8-Bit Offset) OP	#	~	Boolean/Arithmetic Operation	Condition Code H	I	N	Z	C
Increment	INC	4C	1	2	5C	1	2	3C	2	5	7C	1	5	6C	2	6	A+1→A or X+1→X or M+1→M	•	•	^	^	•
Decrement	DEC	4A	1	2	5A	1	2	3A	2	5	7A	1	5	6A	2	6	A−1→A or X−1→X or M−1→M	•	•	^	^	•
Clear	CLR	4F	1	2	5F	1	2	3F	2	5	7F	1	5	6F	2	6	00→A or 00→X or 00→M	•	•	0	1	•
Complement	COM	43	1	2	53	1	2	33	2	5	73	1	5	63	2	6	$\overline{A}$→A or $\overline{X}$→X or $\overline{M}$→M	•	•	^	^	1
Negate (2's Complement)	NEG	40	1	2	50	1	2	30	2	5	70	1	5	60	2	6	00−A→A or 00−X→X or 00−M→M	•	•	^	^	^
Rotate Left Thru Carry	ROL	49	1	2	59	1	2	39	2	5	79	1	5	69	2	6		•	•	^	^	^
Rotate Right Thru Carry	ROR	46	1	2	56	1	2	36	2	5	76	1	5	66	2	6		•	•	^	^	^
Logical Shift Left	LSL	48	1	2	58	1	2	38	2	5	78	1	5	68	2	6		•	•	^	^	^
Logical Shift Right	LSR	44	1	2	54	1	2	34	2	5	74	1	5	64	2	6		•	•	0	^	^
Arithmetic Shift Right	ASR	47	1	2	57	1	2	37	2	5	77	1	5	67	2	6		•	•	^	^	^
Arithmetic Shift Left	ASL	48	1	2	58	1	2	38	2	5	78	1	5	68	2	6	Equal to LSL	•	•	^	^	^
Test for Negative or Zero	TST	4D	1	2	5D	1	2	3D	2	4	7D	1	4	6D	2	5	A−00 or X−00 or M−00	•	•	^	^	•

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

Table 7 Branch Instructions

Operations	Mnemonic	Addressing Modes			Branch Test	Condition Code				
		Relative								
		OP	#	~		H	I	N	Z	C
Branch Always	BRA	20	2	3	None	•	•	•	•	•
Branch Never	BRN	21	2	3	None	•	•	•	•	•
Branch IF Higher	BHI	22	2	3	C+Z=0	•	•	•	•	•
Branch IF Lower or Same	BLS	23	2	3	C+Z=1	•	•	•	•	•
Branch IF Carry Clear	BCC	24	2	3	C=0	•	•	•	•	•
(Branch IF Higher or Same)	(BHS)	24	2	3	C=0	•	•	•	•	•
Branch IF Carry Set	BCS	25	2	3	C=1	•	•	•	•	•
(Branch IF Lower)	(BLO)	25	2	3	C=1	•	•	•	•	•
Branch IF Not Equal	BNE	26	2	3	Z=0	•	•	•	•	•
Branch IF Equal	BEQ	27	2	3	Z=1	•	•	•	•	•
Branch IF Half Carry Clear	BHCC	28	2	3	H=0	•	•	•	•	•
Branch IF Half Carry Set	BHCS	29	2	3	H=1	•	•	•	•	•
Branch IF Plus	BPL	2A	2	3	N=0	•	•	•	•	•
Branch IF Minus	BMI	2B	2	3	N=1	•	•	•	•	•
Branch IF Interrupt Mask Bit is Clear	BMC	2C	2	3	I=0	•	•	•	•	•
Branch IF Interrupt Mask Bit is Set	BMS	2D	2	3	I=1	•	•	•	•	•
Branch IF Interrupt Line is Low	BIL	2E	2	3	INT=0	•	•	•	•	•
Branch IF Interrupt Line is High	BIH	2F	2	3	INT=1	•	•	•	•	•
Branch to Subroutine	BSR	AD	2	5	——	•	•	•	•	•

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

Table 8 Bit Manipulation Instructions

Operations	Mnemonic	Addressing Modes						Boolean/ Arithmetic Operation	Branch Test	Condition Code				
		Bit Set/Clear			Bit Test and Branch									
		OP	#	~	OP	#	~			H	I	N	Z	C
Branch IF Bit n is set	BRSET n(n=0···7)	—	—	—	2·n	3	5	——	Mn=1	•	•	•	•	^
Branch IF Bit n is clear	BRCLR n(n=0···7)	—	—	—	01+2·n	3	5	——	Mn=0	•	•	•	•	^
Set Bit n	BSET n(n=0···7)	10+2·n	2	5	—	—	—	1→Mn	——	•	•	•	•	•
Clear Bit n	BCLR n(n=0···7)	11+2·n	2	5	—	—	—	0→Mn	——	•	•	•	•	•

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

Table 9 Control Instructions

Operations	Mnemonic	Addressing Modes			Boolean Operation	Condition Code				
		Implied								
		OP	#	~		H	I	N	Z	C
Transfer A to X	TAX	97	1	2	A→X	●	●	●	●	●
Transfer X to A	TXA	9F	1	2	X→A	●	●	●	●	●
Set Carry Bit	SEC	99	1	1	1→C	●	●	●	●	1
Clear Carry Bit	CLC	98	1	1	0→C	●	●	●	●	0
Set Interrupt Mask Bit	SEI	9B	1	2	1→I	●	1	●	●	●
Clear Interrupt Mask Bit	CLI	9A	1	2	0→I	●	0	●	●	●
Software Interrupt	SWI	83	1	10		●	1	●	●	●
Return from Subroutine	RTS	81	1	5		●	●	●	●	●
Return from Interrupt	RTI	80	1	8		?	?	?	?	?
Reset Stack Pointer	RSP	9C	1	2	$FF→SP	●	●	●	●	●
No-Operation	NOP	9D	1	1	Advance Prog. Cntr. Only	●	●	●	●	●
Decimal Adjust A	DAA	8D	1	2	Converts binary add of BCD charcters into BCD format	●	●	^	^	^*
Stop	STOP	8E	1	4		●	●	●	●	●
Wait	WAIT	8F	1	4		●	●	●	●	●

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

* Are BCD characters of upper byte 10 or more? (They are not cleared if set in advance.)

Table 10 Instruction Set (in Alphabetical Order)

Mnemonic	Addressing Modes										Condition Code				
	Implied	Immediate	Direct	Extended	Relative	Indexed (No Offset)	Indexed (8-Bit)	Indexed (16-Bit)	Bit Set/ Clear	Bit Test & Branch	H	I	N	Z	C
ADC		×	×	×		×	×	×			^	●	^	^	^
ADD		×	×	×		×	×	×			^	●	^	^	^
AND		×	×	×		×	×	×			●	●	^	^	●
ASL	×		×			×	×				●	●	^	^	^
ASR	×		×			×	×				●	●	^	^	^
BCC					×						●	●	●	●	●
BCLR									×		●	●	●	●	●
BCS					×						●	●	●	●	●
BEQ					×						●	●	●	●	●
BHCC					×						●	●	●	●	●
BHCS					×						●	●	●	●	●
BHI					×						●	●	●	●	●
(BHS)					×						●	●	●	●	●
BIH					×						●	●	●	●	●
BIL					×						●	●	●	●	●
BIT		×	×	×		×	×	×			●	●	^	^	●
(BLO)					×						●	●	●	●	●
BLS					×						●	●	●	●	●
BMC					×						●	●	●	●	●
BMI					×						●	●	●	●	●
BMS					×						●	●	●	●	●
BNE					×						●	●	●	●	●
BPL					×						●	●	●	●	●
BRA					×						●	●	●	●	●

(to be continued)

Condition Code Symbols:

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

Table 10 Instruction Set (in Alphabetical Order)

Mnemonic	Addressing Modes										Condition Code				
	Implied	Immediate	Direct	Extended	Relative	Indexed (No Offset)	Indexed (8-Bit)	Indexed (16-Bit)	Bit Set/ Clear	Bit Test & Branch	H	I	N	Z	C
BRN					×						•	•	•	•	•
BRCLR										×	•	•	•	•	^
BRSET										×	•	•	•	•	^
BSET									×		•	•	•	•	•
BSR					×						•	•	•	•	•
CLC	×										•	•	•	•	0
CLI	×										•	0	•	•	•
CLR	×		×			×	×				•	•	0	1	•
CMP		×	×	×		×	×	×			•	•	^	^	^
COM	×		×			×	×				•	•	^	^	1
CPX		×	×	×		×	×	×			•	•	^	^	^
DAA	×										•	•	^	^	^
DEC	×		×			×	×				•	•	^	^	•
EOR		×	×	×		×	×	×			•	•	^	^	•
INC	×		×			×	×				•	•	^	^	•
JMP			×	×		×	×	×			•	•	•	•	•
JSR			×	×		×	×	×			•	•	•	•	•
LDA		×	×	×		×	×	×			•	•	^	^	•
LDX		×	×	×		×	×	×			•	•	^	^	•
LSL	×		×			×	×				•	•	^	^	^
LSR	×		×			×	×				•	•	0	^	^
NEG	×		×			×	×				•	•	^	^	^
NOP	×										•	•	•	•	•
ORA		×	×	×		×	×	×			•	•	^	^	•
ROL	×		×			×	×				•	•	^	^	^
ROR	×		×			×	×				•	•	^	^	^
RSP	×										•	•	•	•	•
RTI	×										?	?	?	?	?
RTS	×										•	•	•	•	•
SBC		×	×	×		×	×	×			•	•	^	^	^
SEC	×										•	•	•	•	1
SEI	×										•	1	•	•	•
STA			×	×		×	×	×			•	•	^	^	•
STOP	×										•	•	•	•	•
STX			×	×		×	×	×			•	•	^	^	•
SUB		×	×	×		×	×	×			•	•	^	^	^
SWI	×										•	1	•	•	•
TAX	×										•	•	•	•	•
TST	×		×			×	×				•	•	^	^	•
TXA	×										•	•	•	•	•
WAIT	×										•	•	•	•	•

Condition Code Symbols:

H	Half Carry (From Bit 3)	C	Carry Borrow
I	Interrupt Mask	^	Test and Set if True, Cleared Otherwise
N	Negative (Sign Bit)	•	Not Affected
Z	Zero	?	Load CC Register From Stack

Table 11 Operation Code Map

	Bit Manipulation		Branch	Read/Modify/Write					Control		Register/Memory						
	Test & Branch	Set/ Clear	Rel	DIR	A	X	,X1	,X0	IMP	IMP	IMM	DIR	EXT	,X2	,X1	,X0	← HIGH
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	BRSET0	BSET0	BRA	NEG					RTI*	—	SUB						0
1	BRCLR0	BCLR0	BRN	—					RTS*	—	CMP						1
2	BRSET1	BSET1	BHI	—					—	—	SBC						2
3	BRCLR1	BCLR1	BLS	COM					SWI*	—	CPX						3
4	BRSET2	BSET2	BCC	LSR					—	—	AND						4
5	BRCLR2	BCLR2	BCS	—					—	—	BIT						5
6	BRSET3	BSET3	BNE	ROR					—	—	LDA						6
7	BRCLR3	BCLR3	BEQ	ASR					—	TAX*	—	STA				STA(+1)	7
8	BRSET4	BSET4	BHCC	LSL/ASL					—	CLC	EOR						8
9	BRCLR4	BCLR4	BHCS	ROL					—	SEC	ADC						9
A	BRSET5	BSET5	BPL	DEC					—	CLI*	ORA						A
B	BRCLR5	BCLR5	BMI	—					—	SEI*	ADD						B
C	BRSET6	BSET6	BMC	INC					—	RSP*	—	JMP(−1)					C
D	BRCLR6	BCLR6	BMS	TST(−1)	TST			TST(−1)	DAA*	NOP	BSR*	JSR(+2)		JSR(+1)		JSR(+2)	D
E	BRSET7	BSET7	BIL	—					STOP*	—	LDX						E
F	BRCLR7	BCLR7	BIH	CLR					WAIT*	TXA*	—	STX				STX(+1)	F
	3/5	2/5	2/3	2/5	1/2	1/2	2/6	1/5	1/*	1/1	2/2	2/3	3/4	3/5	2/4	1/3	

LOW

(NOTES) 1. "—" is an undefined operation code.
2. The lowermost numbers in each column represent a byte count and the number of cycles required (byte count/number of cycles).
The number of cycles for the mnemonics asterisked (*) is a follows:

RTI = 8	DAA = 2	TAX = 2	BSR = 5
RTS = 5	STOP = 4	RSP = 2	CLI = 2
SWI = 10	WAIT = 4	TXA = 2	SEI = 2

3. The parenthesized numbers must be added to the cycle count of the particular instruction.

- **Additional Instructions**

The following new instructions are used on the HD6305X:

DAA Converts the contents of the accumulator into BCD code.

WAIT Causes the MCU to enter the wait mode. For this mode, see the topic, Wait Mode.

STOP Causes the MCU to enter the stop mode. For this mode, see the topic, Stop Mode.

■ PRECAUTION 1—BOARD DESIGN OF OSCILLATION CIRCUIT

When connecting crystal and ceramic resonator with the XTAL and EXTAL pins to oscillate, observe the following in designing the board.

(1) Locate crystal, ceramic resonator, and load capacity C_1 and C_2 as near the LSI as possible. (Induction of noise from outside to the XTAL and EXTAL pins may cause trouble in oscillation.)

(2) Wire the signal lines to the neighbouring XTAL and EXTAL pins as far apart as possible.

(3) Board design of situating signal lines or power supply lines near the oscillator circuit as shown in Fig. 40, should not be used because of trouble in oscillation in induction. The resistor between the XTAL and EXTAL, and pins close to them should be 10M Ω or more.

■ PRECAUTION 2—PROGRAM OF WRITE ONLY REGISTER

Read/Modify/Write instructions are unavailable for changing the contents of Write Only Register (e.g. DDR; Data Direction Register of I/O port) of HD6305X, HD6305Y and HD63P05Y.

(1) Data cannot be read from write only register. (e.g. DDR of I/O port)

While read/modify/write instructions are executed in the following sequence.

(i) Reads the contents from appointed address.

(ii) Changes the data which has been read.

(iii) Turn the data back to the original address.

Thus, read/modify/write instructions cannot be applied to write only register such as DDR.

(2) For the same reason, do not set DDR of I/O port using BSET and BCLR instructions.

(3) Stored instructions (e.g. STA and STX, etc.) are available for writing into the write only register.

■ PRECAUTION 3—SENDING/RECEIVING PROGRAM OF SERIAL DATA

Be careful that malfunction may occur if SDR (SERIAL DATA REGISTER: $0012) is read or written during transmitting or receiving serial data.

■ PRECAUTION 4—WAIT/STOP INSTRUCTIONS PROGRAM

When I bit of condition code register is "1" and an interrupt ($\overline{INT_2}$, TIMER/$\overline{INT_2}$) is held, the MCU does not enter into WAID mode by executing the WAIT instruction.

In that case, after the 4 dummy cycles, the MCU executes the next instruction.

In the same way, when external interrupts ($\overline{INT}$, $\overline{INT_2}$) are held at the bit I set, the MCU does not enter into the STOP mode by executing STOP instruction. In that case the MCU executes the next instruction after the 4 dummy cycles.

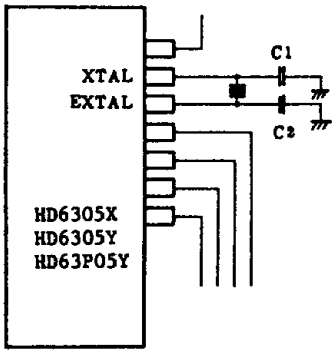


Figure 39 Design of Oscillation Circuit Board

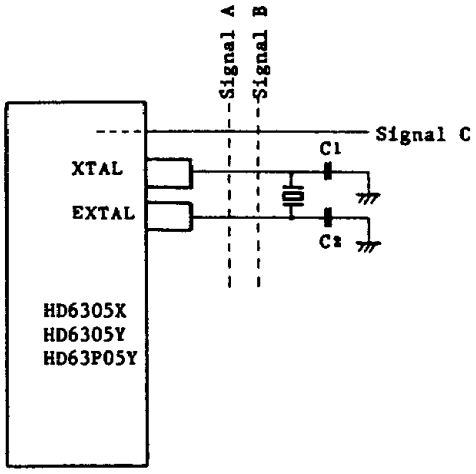


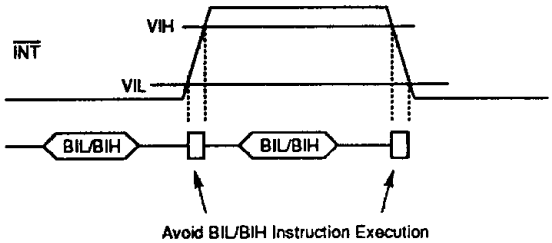
Figure 40 Example of Circuit Causing Trouble in Oscillation

■ PRECAUTION WHEN USING BIL/BIH INSTRUCTION

(1) Execute Instruction after the $\overline{INT}$ Voltage level has stabilized above V_{IH} or below V_{IL}.

(2) $\overline{INT}$ voltage level needs to be stabilized while BIL/BIH Instruction Execution.

There may be a malfunction by glitch on control signal if BIL/BIH Instruction Execution has exercized in unstablized $\overline{INT}$ signal level.



Avoid BIL/BIH Instruction Execution

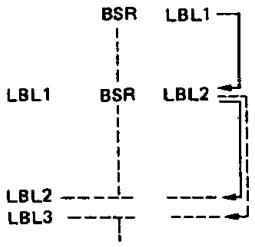
■ PRECAUTION TO USE BSR

If there is 2nd BSR programmed on the address which is directed by first BSR, 2nd BSR may not be executed correctly. For this reason, BSR should not be programmed on the address which is directed by first BSR.

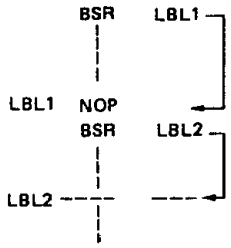
If necessary, please program as following.

(1) On the address which first BSR directed, NOP instruction should be inserted before second BSR.

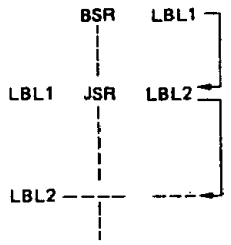
(2) On the address which first BSR directed, JSR instruction should be programmed instead of 2nd BSR.



example of malfunction of 2nd BSR execution



example of counter measure (NOP is inserted)



example of counter measure (JSR is used instead of BSR)