CMOS 8-BIT MICROCONTROLLER

TMP87C405AM

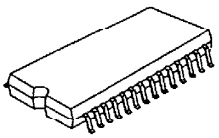
87C405A is high speed and high performance 8-bit single chip microcomputers to operate on low voltage and low power consumption. This MCU contains ROM, RAM, input/output ports and multi-function timer/counter.

Part No.	ROM	RAM	PACKAGE	OTP MCU	OPERATION VOLTAGE RANGE
TMP87C405AM	4K byte	256 byte	SOP28-P-450-1.27	TMP87P808M	2.7 V to 5.5 V at 4.2 MHz

Features

- ◆ 8-bit single chip microcomputer TLCS-870 series
- ◆ Minimum instruction execution time : 0.5 μs (at 8 MHz, gear ratio 1 / 1)
- ◆ 129 types & 412 basic instructions
  - Multiplication (8 bits × 8 bits, 16 bits ÷ 8 bits) : Execution time 3.5 μs (at 8 MHz, gear ratio 1 / 1))
  - Bit manipulations (Set / Clear / Complement / Load / Store / Test / Exclusive or)
  - 16-bit data operations
  - 1-byte jump / call (Short relative jump / Vector call)
- ◆ 9 interrupt sources (External : 4, Internal : 5)
  - All sources have independent latches each, and nested interrupt control is available.
  - Edge-selectable external interrupts with noise reject
  - High-speed task switching by register bank changeover
- ◆ Input / Output ports (22 pins)
  - High current output : 6 pins (Typ.7 mA)
- ◆ Two 16-bit Timer/Counters
  - Timer, Eventcounter, Programmable pulse generator output, Pulse width measurement, External trigger timer, Window modes
- ◆ Time Base Timer
  - Interrupt frequency types : 8 types (1 to 16384 Hz)
- ◆ Divider output function (frequency : 4 types)
- ◆ Watchdog Timer
- ◆ Two Power saving operating modes
  - STOP mode : Oscillation stops. Battery/Capacitor back-up Port output hold/high-impedance
  - IDLE1 mode : CPU stops, and Peripherals operate using high-frequency clock. Release by interrupts.
- ◆ Operating voltage : 2.7 to 5.5 V at 4.2 MHz/4.5 to 5.5 V at 8 MHz
- ◆ Emulation pod : BM87C408M0A

SOP28-P-450-1.27

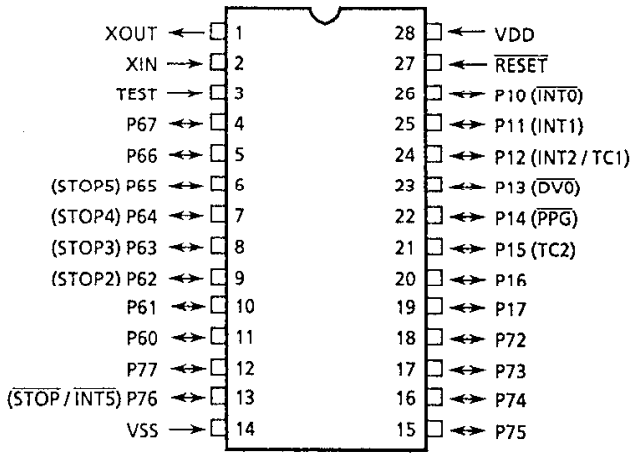


TMP87C405AM

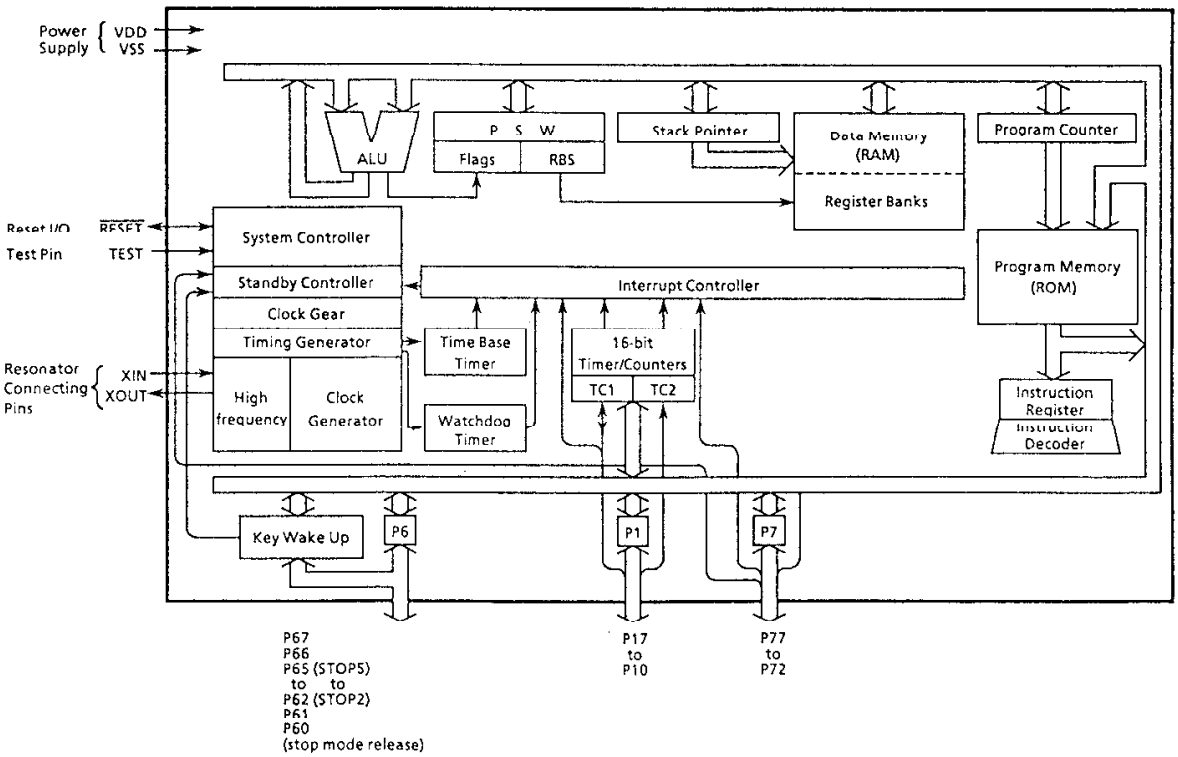
980901EBP1

Pin Assignments (TOP VIEW)

SOP28-P-450-1.27



Block Diagram



Pin Function

Pin Name	Input / Output	Function	
P17, P16	I/O	8-bit programmable input/output port (tri-states) Each bit of the port can be individually configured as an input or an output under software control. When used as an external interrupt input or a timer counter input, the input mode is configured. When used as a divider output or a PPG output, the latch must be set to "1" and the output mode is configured.	
P15 (TC2)	I/O (Input)		Timer/Counter 2 input
P14 ($\overline{PPG}$)	I/O (Output)		Programmable pulse generator output
P13 ($\overline{DVO}$)			Divider output
P12 (INT2 / TC1)	I/O (Input)		External interrupt input 2 or Timer/Counter 1 input
P11 (INT1)			External interrupt input 1
P10 ($\overline{INT0}$)			External interrupt input 0
P67, P66	I/O	8-bit programmable input/output port (tri-states). Each bit of the port can be individually configured as an input or an output under software control. When used as stop mode release input , the input mode is configured.	
P65 (STOP5)	I/O (Input)		
P64 (STOP4)			
P63 (STOP3)			
P62 (STOP2)			
P61, P60			
P77	I/O	6-bit programmable input/output port (tri-states). Each bit of the port can be individually configured as an input or an output under software control. When used as an external interrupt input, the input mode is configured.	
P76 ($\overline{STOP}$/$\overline{INT5}$)	I/O (Input)		STOP mode release input/External interrupt 5 input
P75 to P72	I/O		
XIN, XOUT	Input, Output	Resonator connecting pins for high-frequency clock. For inputting external clock, XIN is used and XOUT is opened.	
$\overline{RESET}$	I/O	Reset signal input or watchdog timer output/address-trap-reset output.	
TEST	Input	Test pin for outgoing test. Be tied to low.	
VDD	Power Supply	2.7 to 5.5 V	
VSS		0 V (GND)	

Operational Description

1. CPU Core Functions

The CPU core consists of a CPU, a system clock controller, an interrupt controller, and a watchdog timer. This section provides a description of the CPU core, the program memory, the data memory, and the reset circuit.

1.1 Memory Address Map

The TLCS-870 Series is capable of addressing 64K bytes of memory. Figure 1-1 shows the memory address maps of the 87C405A. In the TLCS-870 Series, the memory is organized 4 address spaces (ROM, RAM, SFR, and DBR). It uses a memory mapped I/O system, and all I/O registers are mapped in the SFR/DBR address spaces. There are 16 banks of general-purpose registers. The register banks are also assigned to the RAM address space.

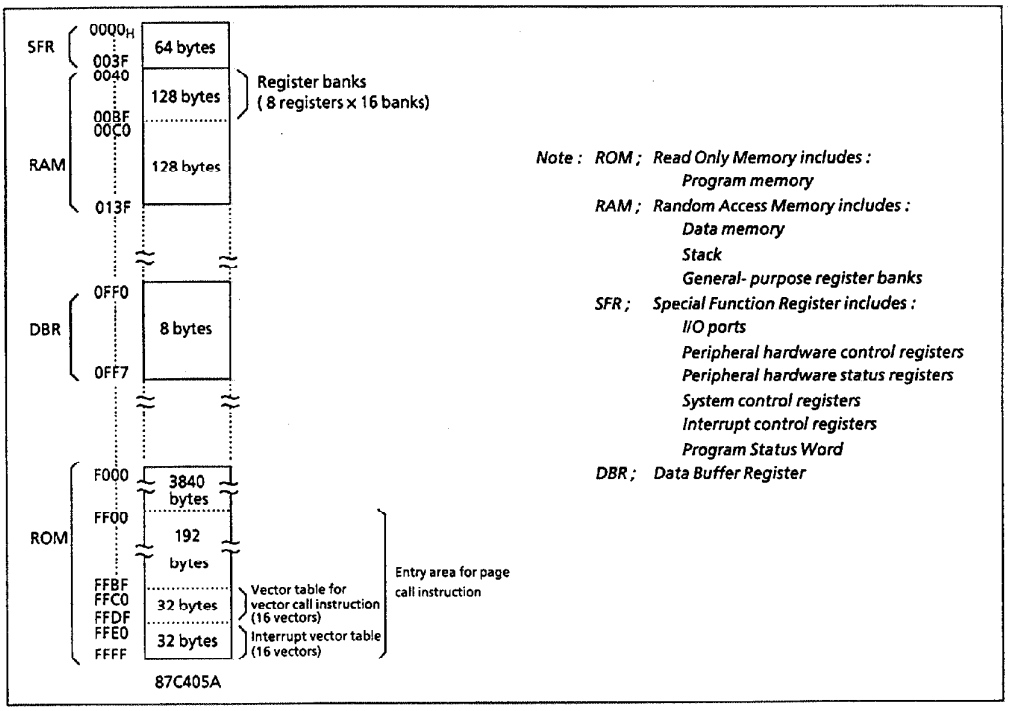


Figure 1-1. Memory address map

1.2 Program Memory (ROM)

The 87C405A have a 4K bytes (addresses F000 to FFFF$_H$) of program memory (mask programmed ROM). Addresses FF00 to FFFF$_H$ of program memory is also used for a special purpose.

(1) Interrupt vector table (addresses FFE0 to FFFF$_H$)
This table consists of a reset vector and 16 interrupt vectors (2 bytes/vector). These vectors store a reset start address and interrupt service routine entry addresses.

(2) Vector table for vector call instructions (addresses FFC0 to FFDF$_H$)
This table stores call vectors (subroutine entry address, 2 bytes/vector) for the vector call instructions [CALLV a]. There are 16 vectors. The CALLV instruction increases memory efficiency when utilized for frequently used subroutine calls (called from 3 or more locations).

(3) Entry area(addresses FF00 to FFFF$_H$) for page call instructions
This is the subroutine entry address area for the page call instructions [CALLP a]. Addresses FF00 to FFBF$_H$ are normally used because addresses FFC0 to FFFF$_H$ are used for the vector tables.

Programs and fixed data are stored in the program memory. The instruction to be executed next is read from the address indicated by the current contents of the program counter (PC). There are relative jump and absolute jump instructions. The concepts of page or bank boundaries are not used in the program memory concerning any jump instruction.

Example : The relationship between the jump instructions and the PC.

① 5-bit PC-relative jump [JRS cc, $+2+d]
F8C4H: JRS T, $+2+08H
When JF = 1, the jump is made to F8CE$_H$, which is 08$_H$ added to the contents of the PC. (The PC contains the address of the instruction being executed +2; therefore, in this case, the PC contents are F8C4$_H$ + 2 = F8C6$_H$.)

② 8-bit PC-relative jump [JR cc, $+2+d]
F8C4H: JR Z, $+2+80H
When ZF = 1, the jump is made to F846$_H$, which is FF80$_H$ (-128) added to the current contents of the PC.

③ 16-bit absolute jump [JP a]
F8C4H: JP 0F235H
An unconditional jump is made to address F235$_H$. The absolute jump instruction can jump anywhere within the entire 64K-bytes space.

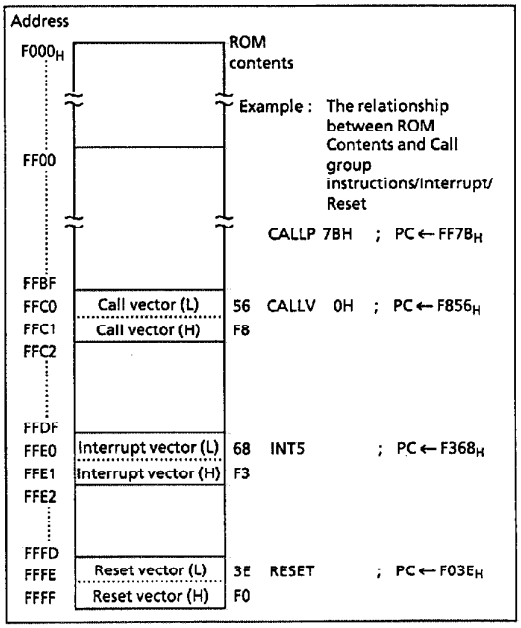


Figure 1-2. Program memory map

In the TLCS-870 Series, the same instruction used to access the data memory is also used to read out fixed data stored in the program memory. The register offset PC-relative addressing (PC + A) instructions can also be used, and the code conversion, table look-up and n-way multiple jump processing can easily be programmed.

Example 1 : Loads the ROM contents at the address specified by the HL register pair contents into the accumulator (87C405 : HL ≥ $F000_H$)

```
LD      A, (HL)          ; A←ROM (HL)
```

Example 2 : Converts BCD to 7-segment code (common anode LED). When A = 05_H, 92_H is output to port P1 after executing the following program.


```
        ADD     A, TABLE – $ – 4     ; P1 ←ROM (TABLE + A)
        LD      (P1), (PC + A)
        JRS     T, SNEXT             ; Jump to SNEXT
TABLE : DB      0C0H, 0F9H, 0A4H, 0B0H, 99H, 92H, 82H, 0D8H, 80H, 98H
SNEXT :
```

Notes : "$" is a header address of ADD instruction. DB is a byte data definition instruction.

Example 3 : N-way multiple jump in accordance with the contents of accumulator (0 ≤ A ≤ 3).

```
SHLC    A            ; if A = 00H then PC←F234H
JP      (PC + A)       if A = 01H then PC←F378H
                       if A = 02H then PC←FA37H
                       if A = 03H then PC←F1B0H
DW      0F234H, 0F378H, 0FA37H, 0F1B0H
```

Notes : DW is a word data definition instruction. Word = 2 bytes.

```
SHLC A
JP (PC + A)
34
F2
78
F3
37
FA
B0
F1
```

1.3 Program Counter (PC)

The program counter (PC) is a 16-bit register which indicates the program memory address where the instruction to be executed next is stored. After reset, the reset vector stored in the vector table (addresses $FFFF_H$ and $FFFE_H$) is loaded into the PC; therefore, program execution is possible from any desired address. For example, when $F0_H$ and $3E_H$ are stored at addresses $FFFF_H$ and $FFFE_H$, respectively, the execution starts from address $F03E_H$ after reset.

The TLCS-870 Series utilizes pipelined processing (instruction pre-fetch); therefore, the PC always indicates 2 addresses in advance. For example, while a 1-byte instruction stored at address $F123_H$ is being executed, the PC contains $F125_H$.

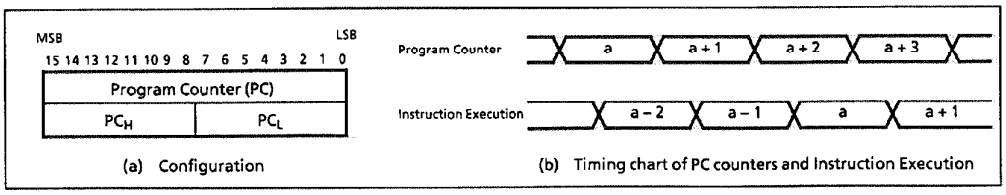


Figure 1-3. Program counter

1.4 Data Memory (RAM)

The 87C405A has a 256 bytes (addresses 0040 to $013F_H$) of data memory (static RAM). Figure 1.4 shows the data memory map.

Addresses 0000 to $00FF_H$ are used as a direct addressing area to enhance instructions which utilize this addressing mode; therefore, addresses 0040 to $00FF_H$ in the data memory can also be used for user flags or user counters.

Example 1 : If bit 2 at data memory address $00C0_H$ is "1", 00_H is written to data memory at address $00E3_H$; otherwise, FF_H is written to the data memory at address $00E3_H$.

```
        TEST    (00C0H).2       ; if (00C0H)2 = 0 then jump
        JRS     T,SZERO
        CLR     (00E3H)         ; (00E3H)←00H
        JRS     T,SNEXT
SZERO : LD      (00E3H), 0FFH   ; (00E3H)←FFH
SNEXT :
```

Example 2 : Increments the contents of data memory at address $00F5_H$, and clears to 00_H when 10_H is exceeded.

```
        INC     (00F5H)
        AND     (00F5H), 0FH
```

General-purpose register banks (8 registers × 16 banks) are also assigned to the 128 bytes of addresses 0040_H to $00BF_H$. Access as data memory is still possible even when being used for registers. For example, when the contents of the data memory at address 0040_H is read out, the contents of the accumulator in the bank 0 are also read out.

The stack can be located anywhere within the data memory except the register bank area. For more details on the stack, see section "1.7 Stack and Stack Pointer".

With the TLCS-870 Series, programs in data memory cannot be executed. If the program counter indicates a specific data memory address (addresses 0040 to $013F_H$), an address-trap-reset is generated due to bus error. (Output from the $\overline{RESET}$ pin goes low.)

The data memory contents become unstable when the power supply is turned on; therefore, the data memory should be initialized by an initialization routine.

Example 1 : Clears RAM to 0 except the bank 0

```
        LD      HL, 0048H       ; Sets start address to HL register pair
        LD      A, H            ; Sets initial data (A)
        LD      BC, 00F7H       ; Sets number of byte to BC register pair
SRAMCLR : LD    (HL+), A
        DEC     BC
        JRS     F, SRAMCLR
```

Note : "$" is a header address of ADD instruction. The general-purpose registers are mapped in the RAM; therefore, do not clear RAM at the current bank addresses. Clears RAM to 0 except the bank 0.

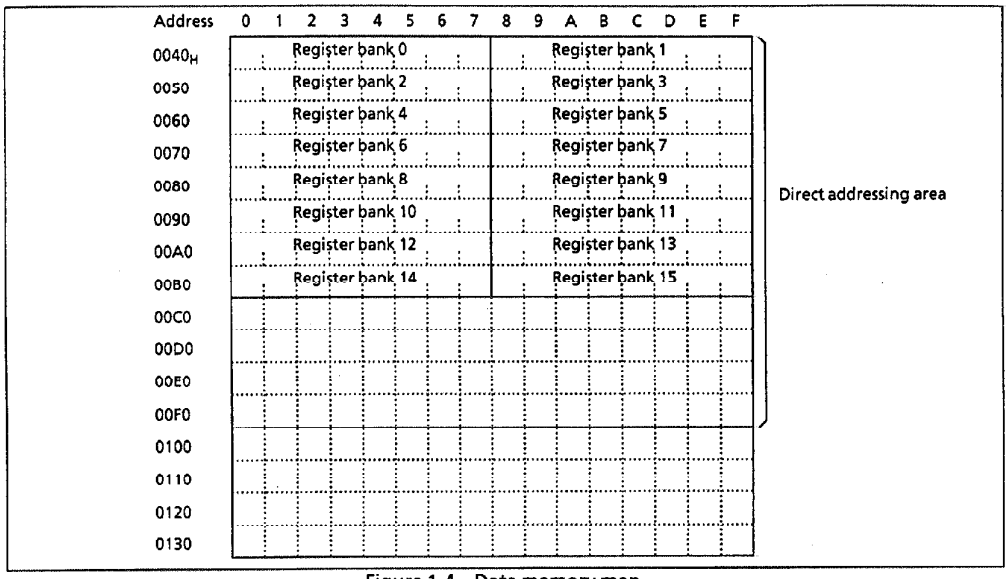


Figure 1-4. Data memory map

1.5 General-purpose Register Banks

General-purpose registers are mapped into addresses 0040 to 00BF$_H$ in the data memory. There are 16 register banks, and each bank contains eight 8-bit registers W, A, B, C, D, E, H, and L. Figure 1-5 shows the general-purpose register bank configuration. The unused register banks can be used as a data memory.

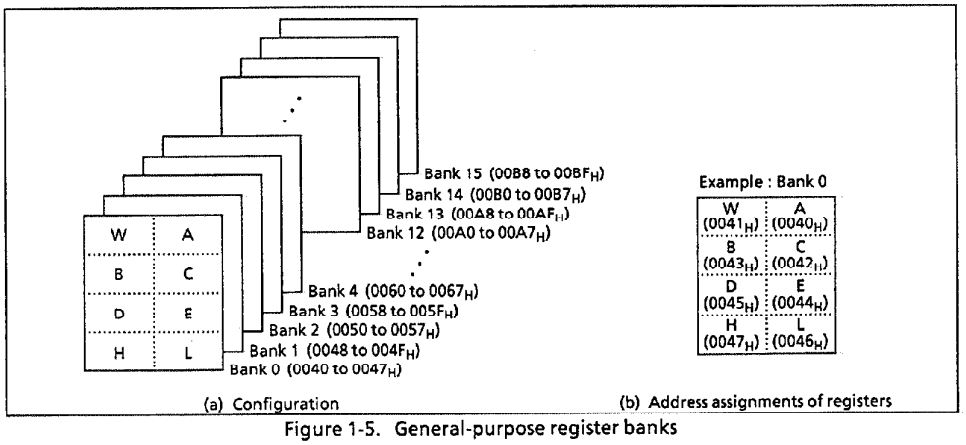


Figure 1-5. General-purpose register banks

In addition to access in 8-bit units, the registers can also be accessed in 16-bit units as the register pairs WA, BC, DE, and HL. Besides its function as a general-purpose register, the register also has the following functions.

(1) A, WA

The A register functions as an 8-bit accumulator and WA the register pair functions as a 16-bit accumulator (W is high byte and A is low byte). Registers other than A can also be used as accumulators for 8-bit operations.

```
Examples : ①   ADD  A, B          ; Adds B contents to A contents and stores the result into A.
           ②   SUB  WA, 1234H     ; Subtracts 1234H from WA contents and stores the result into WA.
           ③   SUB  E, A          ; Subtracts A contents from B contents, and stores the result into E.
```

(2) HL, DE

The HL register functions as a data pointer/index register/base register, and the DE register pair function as a data pointer to specify the memory address.

HL also has an auto-post-increment and auto-pre-decrement functions. This function simplifies multiple digit data processing, software LIFO (last-in first-out) processing, etc.

```
Example 1 : ①   LD   A, (HL)       ; Loads the memory contents at the address specified by HL into A.
            ②   LD   A, (HL+52H)   ; Loads the memory contents at the address specified by the value
                                     obtained by adding 52H to HL contents into A.
            ③   LD   A, (HL+C)     ; Loads the memory contents at the address specified by the value
                                     obtained by adding the register C contents to HL contents into A.
            ④   LD   A, (HL+)      ; Loads the memory contents at the address specified by HL into A.
                                     Then increments HL.
            ⑤   LD   A, (-HL)      ; Decrement HL. Then loads the memory contents at the address
                                     specified by new HL into A.
```

TLCS-870 Series can transfer data directly memory to memory, and operate directly between memory data and memory data. This facilitates the programming of block processing.

Example 2 : Block transfer

```
            LD      B, m        ; m = n - 1 (n : Number of bytes to transfer)
            LD      HL, DSTA    ; Sets destination address
            LD      DE, SRCA    ; Sets source address
SLOOP :     LD      (HL), (DE)  ; (HL) ← (DE)
            INC     HL          ; HL ← HL + 1
            INC     DE          ; DE ← DE + 1
            DEC     B           ; B ← B - 1
            JRS     F, SLOOP    ; if B ≧ 0 then loop
```

(3) B, C, BC

Registers B and C can be used as 8-bit buffers or counter, and the BC register pair can be used as a 16-bit buffer or counter. The C register functions as an offset register for register offset index addressing (refer to example 1 ③ above) and as a divisor register for the division instruction.

Example 1 : Repeat processing

```
            LD      B, n        ; Sets n as the number of repetitions
SREPEAT :   processing
            DEC     B
            JRS     F, SREPEAT
```

Example 2 : Division (16-bit ÷ 8-bit)

```
            DIV     WA, C       ; Divides the WA contents by the C contents, places the
                                  quotient in A and the remainder in W.
```

The general-purpose banks are selected by the 4-bit register bank selector (RBS). During reset, the RBS is initialized to "0". The bank selected by the RBS is called the current bank.
Together with the flag, the RBS is assigned to address $003F_H$ in the SFR as the program status word (PSW). There are 3 instructions [LD RBS, n], [PUSH PSW], [POP PSW] to access the PSW. The PSW can be also operated by the memory access instruction.

```
Example 1 :  Incrementing the RBS
             INC   (003FH)     ; RBS ← RBS + 1
Example 2 :  Reading the RBS
             LD    A, (003FH)  ; A ← RBS (The flags are simultaneously read in this instruction.)
```

Highly efficient programming and high speed task switching are possible by using bank changeover to save registers during interrupt and to transfer parameters during subroutine processing. During interrupt, the RBS is automatically saved onto the stack. The bank used before the interrupt is automatically restored by executing an interrupt return instruction [RETI]/[RETN] ; therefore, there is no need for the RBS save/restore software processing.
The TLCS-870 Series supports a maximum of 15 interrupt sources. One bank is assigned to the main program, and one bank can be assigned to each source. Also, to increase the efficiency of data memory usage, assign the same bank to interrupt sources which are not nested.

```
Example : Saving/restoring registers during interrupt task using bank changeover.
PINT1 :   LD          RBS, n          ; RBS ← n (Bank changeover)
          Interrupt processing
          RETI                        : Maskable interrupt return (Bank automatic restoring)
```

1.6 Program Status Word (PSW)

The program status word (PSW) consists of a register bank selector (RBS) and flags, and the PSW is assigned to address $003F_H$ in the SFR.
The RBS can be read and written using the memory access instruction, however the flags can only be read. When writing to the PSW, the change specified by the instruction is made without writing data to the flags. For example, when the instruction [LD (003FH), 05H] is executed, "5" is written to the RBS and the JF is set to "1", but the other flags are not affected.
During interrupt, PSW is saved to the stack with the program counter. The PSW is restored from the stack by executing return instructions [RETI]/[RETN].
[PUSH PSW] and [POP PSW] are the PSW access instructions.

1.6.1 Register bank selector (RBS)

The register bank selector (RBS) is a 4-bit register used to select general-purpose register banks. For example, when RBS = 2, bank 2 is currently selected. During reset, the RBS is initialized to "0".

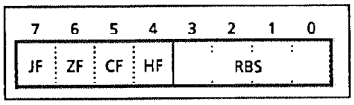


Figure 1-6. PSW (Flags, RBS) configuration

1.6.2 Flags (FLAG)

The flags are configured with the upper 4 bits : a zero flag, a carry flag, a half carry flag and a jump status flag. The flags are set or cleared under conditions specified by the instruction. These flags except the half carry flag are used as jump condition "cc" for conditional jump instructions [JR cc, $ + 2 + d], [JRS cc, $ + 2 + d]. After reset, the jump status flag is initialized to "1", other flags are not affected.

(1) **Zero flag (ZF)**

The ZF is set to "1" if the operation result or the transfer data is 00_H (for 8-bit operations and data transfers)/0000_H (for 16-bit operations); otherwise the ZF is cleared to "0".
During the bit manipulation instructions, the ZF is cleared to "0" if the contents of the specified bit is "1". This flag is set to "1" when the upper 8 bits of the product are 00_H during the multiplication instruction, and when 00_H for the remainder during the division instruction; otherwise it is cleared to "0".

(2) **Carry flag (CF)**

The CF is set to "1" when a carry occurred during addition or a borrow occurred during subtraction; otherwise the CF is cleared to "0". During division, this flag is set to "1" when the divisor is 00_H (divided by zero error), or when the quotient is 100_H or higher (quotiont-overflow error). The CF is also affected during the shift/rotate instructions. The data shifted out from a register is set to the CF. This flag is also a 1-bit register (a boolean accumulator) for the bit manipulation instructions. Set/clear/invert are possible with the CF manipulation instructions.

Example : Bit manipulation (The result of exclusive-OR between bit 5 content of address 07_H and bit 0 content of address $9A_H$ is written to bit 2 of address 01_H.)

```
LD      CF, (0007H) . 5     ; (0001H)2 ← (0007H)5 ∀ (009AH)0
XOR     CF, (009AH) . 0
LD      (0001H) . 2, CF
```

(3) **Half carry flag (HF)**

The HF is set to "1" when a carry occurred to bit 4 of the operation result during an 8-bit addition, or when a borrow occurred from bit 4 of the result during an 8-bit subtraction. This flag is useful in the decimal adjustment for BCD operations (adjustments using the [DAA r], or [DAS r] instructions).

Example : BCD operation
(The A becomes 47_H after executing the following program when $A = 19_H$, $B = 28_H$.)

```
ADD     A, B        ; A ← 41H, HF ← 1, CF = 0
DAA     A           ; A ← 41H + 06H = 47H (decimal-adjust)
```

(4) **Jump status flag (JF)**

The JF is usually set to "1". Zero or carry information is set to the JF after operation. The JF provides the jump condition for conditional jump instructions [JR T/F, $ + 2 = d], [JRS T/F, $ + 2 + d] (T or F is a condition code).

Example : Jump status flag and conditional jump instruction

```
INC     A
JRS     T, SLABLE1      ; Jump when a carry is caused by the immediately
 ⋮                        preceding operation instruction.
LD      A, (HL)
JRS     T, SLABLE2      ; JF is set to "1" by the immediately preceding
 ⋮                        instruction, making it an unconditional jump
                          instruction.
```

Example : The accumulator and flags becomes as shown below after executing the following instructions when the WA register pair, the HL register pair, the data memory at address $00C5_H$, the carry flag and the half carry flag contents being "$219A_H$", "$00C5_H$", "$D7_H$", "1", and "0", respectively.

Instruction	Accumulator after execution	Flag after execution			
		JF	ZF	CF	HF
ADDC A, (HL)	72	1	0	1	1
SUBB A, (HL)	C2	1	0	1	0
CMP A, (HL)	9A	0	0	1	0
AND A, (HL)	92	0	0	1	0
LD A, (HL)	D7	1	0	1	0
ADD A, 66H	00	1	1	1	1

Instruction	Accumulator after execution	Flag after execution			
		JF	ZF	CF	HF
INC A	9B	0	0	1	0
ROLC A	35	1	0	1	0
RORC A	CD	0	0	0	0
ADD WA, 0F508H	16A2	1	0	1	0
MUL W, A	13DA	0	0	1	0
SET A.5	BA	1	1	1	0

1.7 Stack, Stack Pointer

1.7.1 Stack

The stack provides the area in which the return address or status, etc. are saved before a jump is performed to the processing routine during the execution of a subroutine call instruction or the acceptance of an interrupt.

On a subroutine call instruction [CALL a] / [CALLP a] / [CALLV n], the return address is saved (the upper byte is pushed first, followed by the lower byte). During software interrupt instruction [SWI] execution or interrupt, the program status word is saved, then the return address is saved.

When returning from the processing routine, executing a subroutine return instruction [RET] restores the contents to the PC from the stack; executing an interrupt return instruction [RETI] / [RETN] restores the contents to the PC and the PSW.

The stack can be located anywhere within the data memory.

1.7.2 Stack pointer (SP)

The stack pointer (SP) is a 16-bit register to point out the first start address on the stack. The SP is post-decrement when a subroutine call or a push instruction is executed, or when an interrupt is accepted; and the SPC is pre-incremented when a return or a pop instruction is executed. The stack deepens to the direction of the lower address. Figure 1-8 shows the change of the stack access and the SP.

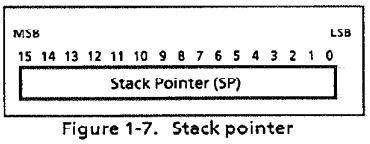


Figure 1-7. Stack pointer

The SP is not initialized hardware-wise but requires initialization by an initialize routine (sets the highest stack address). [LD SP, mn], [LD SP, gg] and [LD gg, SP] are the SP access instructions (mn; 16-bit immediate data, gg; register pair).

```
Example 1 : To initialize the SP
            LD      SP, 013FH       ; SP←013F_H
Example 2 : TO read the SP
            LD      HL, SP          ; HL←SP
```

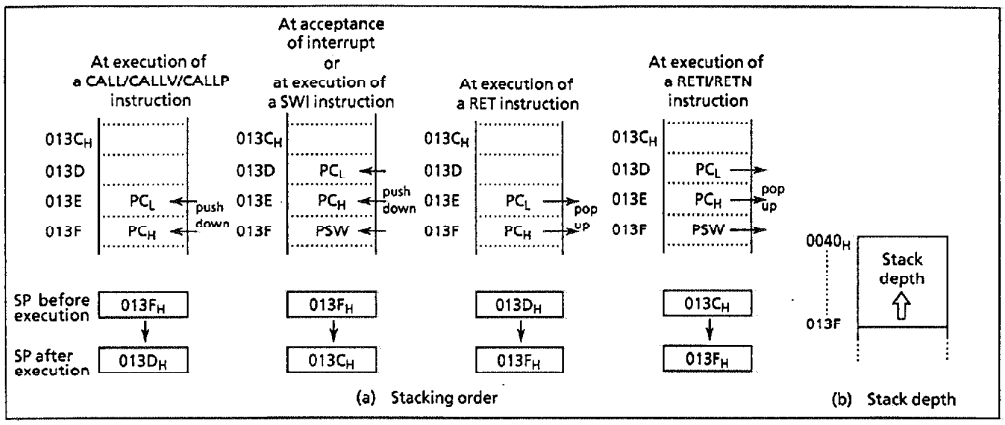


Figure 1-8. Stack

1.8 System Clock Controller

The system clock controller consists of a clock generator, a timing generator, a clock gear, and a stand-by controller.

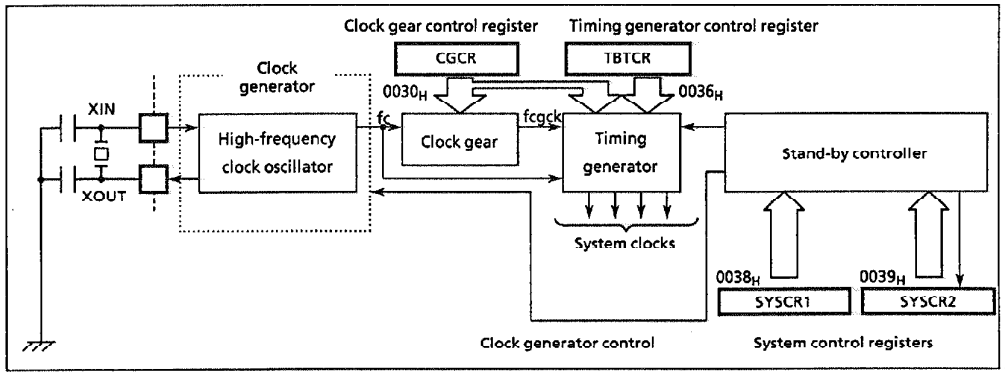


Figure 1-9. System clock controller

1.8.1 Clock generator

The clock generator generates the basic clock which provides the system clocks supplied to the CPU core and peripheral hardware.

The high-frequency (fc) clocks can be easily obtained by connecting a resonator between the XIN/XOUT pins. Clock input from an external oscillator is also possible.

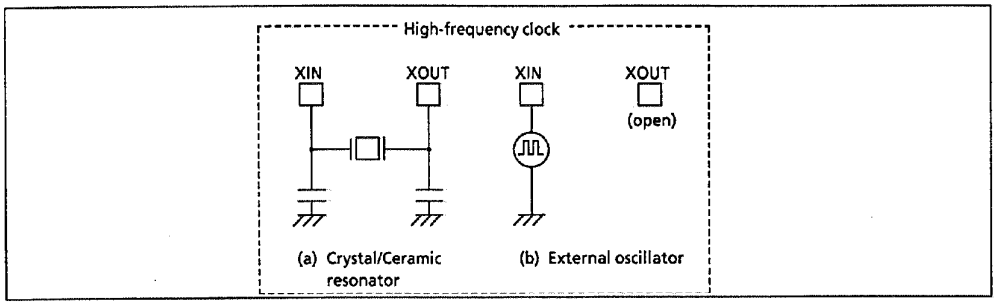


Figure 1-10. Example of resonator connection

Note : Accurate Adjustment of the Oscillation Frequency :
Although no hardware to externally and directly monitor the basic clock pulse is provided, the oscillation frequency can be adjusted by making the program to output fixed frequency pulses to the port while disabling all interrupts and monitoring this pulse. With a system requiring adjustment of the oscillation frequency, the adjusting program must be created beforehand.

1.8.2 Clock gear

A clock generates the basic high-frequency clock which provides the system clocks supplied to the CPU core. The clock gear selects the high-frquency clock from fc, fc/2, fc/4 and fc/8. Power consumption can be redued by switching of the high-frequency from fc to fc/2, fc/4 and fc/8. The clock gear consists of a divided-by-8 prescaler with a mutiplexer.

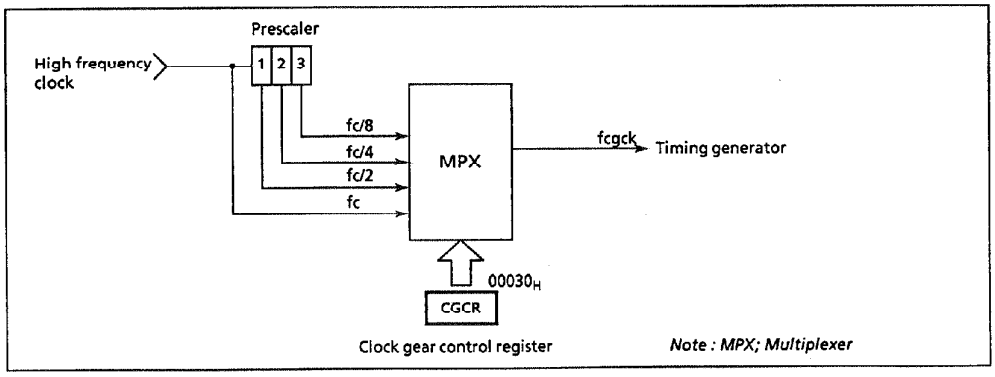


Figure 1-11. Configuration of clock gear

CGCR (00030_H)

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

(Initial value : 000* 1000)

FCGCK	Gear clock selection (write) / gear clock monitor (read)	0*** : reserved 1000 : fc 1001 : fc/2 1010 : fc/4 1011 : fc/8 1100 : reserved 1101 : reserved 111* : reserved	R/W

*Note1: fc ; High-frequency clock * ; don't care*
Note2: Bit 4 in CGCR is always read in as "1" when a read instruction is executed.
Note3: Always set bit 7 to 5 to "0"

Figure 1-12. Clock gear control register

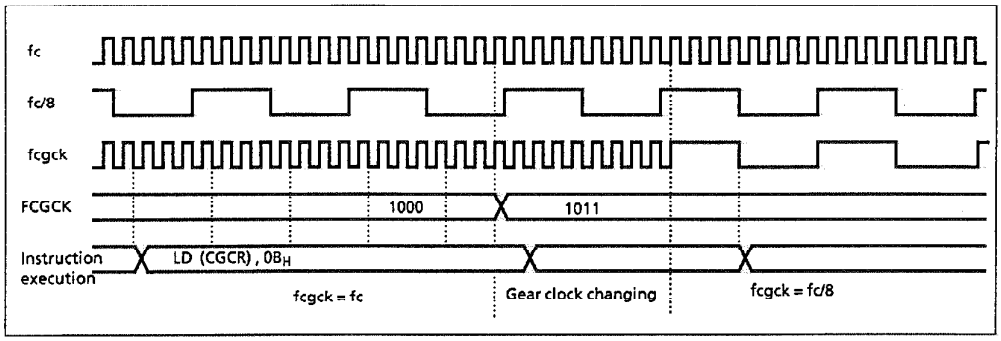


Figure 1-13. Example of clock exchangeable timing by clock gear

1.8.3 Timing generator

The timing generator generates from the basic clock the various system clocks supplied to the CPU core and peripheral hardware. The timing generator provides the following functions:

① Generation of main system clock
② Generation of divider output ($\overline{DVO}$) pulses
③ Generation of source clocks for time base timer
④ Generation of source clocks for watchdog timer
⑤ Generation of internal source clocks for timer/counters
⑥ Generation of warm-up clocks for releasing STOP mode

(1) Configuration of timing generator

The timing generator consists of a 21-stage divider with a divided-by-2 prescaler. During reset and at releasing STOP mode, the divider is cleared to "0", however; the prescaler is not cleared.

Note : Even if the main system clock is changed by the clock gear, the output from the divider is not changed. The peripheral circuit using high-speed divider output (1st output) can not be used when the main system clock slows down.

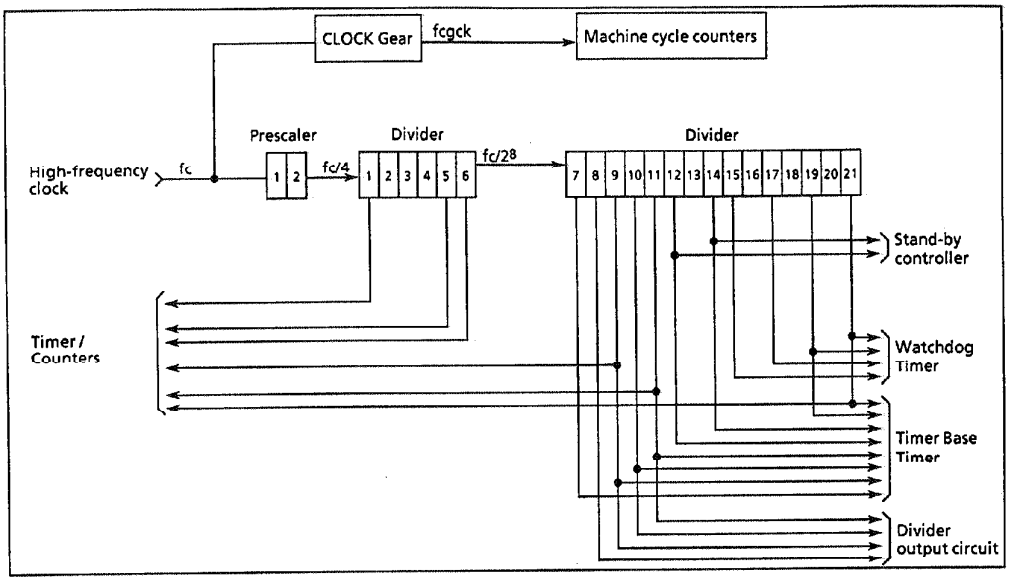


Figure 1-14. Configuration of timing generator

(2) Machine cycle

Instruction execution and built-in hardware operation are synchronized with the system clock.

The minimum instruction execution unit is called an "machine cycle". There are a total of 10 different types of instructions for the TLCS-870 Series: ranging from 1-cycle instructions which require one machine cycle for execution to 10-cycle instructions which require 10 machine cycles for execution.

A machine cycle consists of 4 states (S0 to S3), and each state consists of one main system clock.

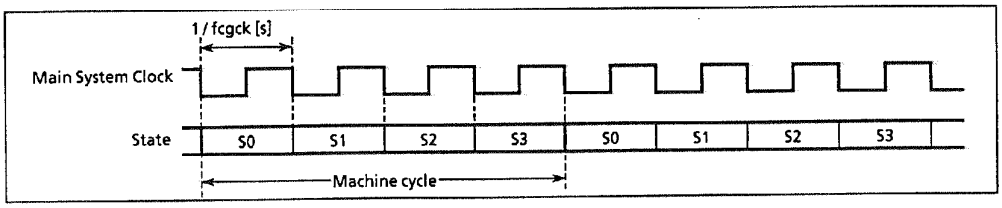


Figure 1-15. Machine cycle

Table 1-1. Machine cycle example

Frequency \ Machine cycle	fcgck = fc	fcgck = fc/2	fcgck = fc/4	fcgck = fc/8
fc = 8 MHz	0.5 μs	1 μs	2 μs	4 μs
fc = 4 MHz	1 μs	2 μs	4 μs	8 μs

1.8.4 Stand-by controller

The stand-by controller starts and stops the oscillation circuits. These modes are controlled by the system control registers (SYSCR1, SYSCR2). Figure 1.16 shows the operating mode transition diagram and Figure 1.17 shows the system control registers.

(1) **Operation mode**

The machine cycle time is 4/fcgck [s]

① NORMAL mode

In this mode, both the CPU core and on-chip peripherals operate. The TMP87C408 is placed in this mode after reset.

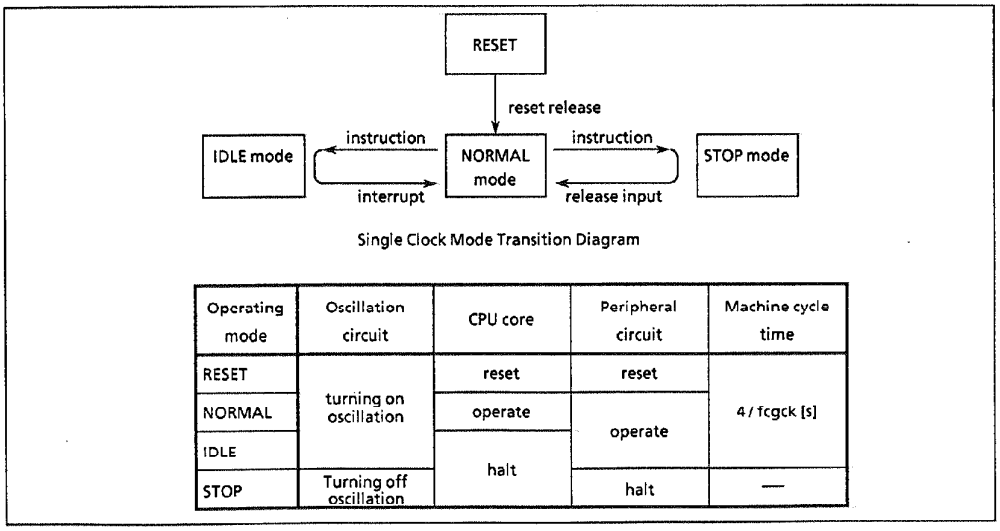
② IDLE mode

In this mode, the CPU and the watchdog timer are halted; however, on-chip peripherals remain active. IDLE mode is started by the system control register 2, and IDLE mode is released to NORMAL mode by an interrupt request from on-chip peripherals or external interrupt inputs. When IMF (interrupt master enable flag) is "1" (interrupt enable), the execution will resume upon acceptance of the interrupt, and the operation will return to normal after the interrupt service is completed. When IMF is "0" (interrupt disable), the execution will resume with the instruction which follows IDLE mode start instruction.

③ STOP mode

In this mode, the internal oscillation circuit is turned off, causing all system operations to be halted. The internal status immediately prior to the halt is held with the lowest power consumption during this mode. The output status of input output ports can be set to either output hold or high-impedance under software control.

STOP mode is started by the system control register 1, and STOP mode is released by STOP input pin (either level sensitive or edge sensitive can be selected). After the warming-up period is completed, the execution resumes with the next instruction which follows the STOP mode start instruction.



Single Clock Mode Transition Diagram

Operating mode	Oscillation circuit	CPU core	Peripheral circuit	Machine cycle time
RESET	turning on oscillation	reset	reset	4 / fcgck [s]
NORMAL		operate	operate	
IDLE		halt		
STOP	Turning off oscillation		halt	—

Figure 1-16. Operating mode transition diagram

System Control Register 1

SYSCR1 (0038_H)

7	6	5	4	3	2	1	0
STOP	RELM	0	OUTEN	WUT			

(Initial value : 0000 00**)

STOP	STOP mode start	0 : CPU core and peripherals remain active 1 : CPU core and peripherals are halted	R/W
RELM	Release method for STOP mode	0 : $\overline{\text{STOP}}$ pin input rising edge release 1 : $\overline{\text{STOP}}$ pin input "H" level release	
OUTEN	Port output control during STOP mode	0 : High-impedance 1 : Remain unchanged	
WUT	Warming-up time at releasing STOP mode	00 : 3×2^{16} / fc 01 : 2^{16} / fc 10 : 3×2^{14} / fc 11 : 2^{14} / fc	

Note 1 : Always set bit 5 to "0".

Note 2 : Bits 1, 0 in SYSCR1 is read in as undefined value when a read instruction is executed.

Note 3 : fc ; clock [Hz]
** ; don't care*

Note 4 : When the STOP mode is started by specifying OUTEN = "0", the internal input of port is fixed to "0" and the interrupt of the falling edge may be set.

System Control Register 2

SYSCR2 (0039_H)

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

(Initial value : 1000 ****)

XEN	High-frequency oscillator control	0 : Turn off oscillation 1 : Turn on oscillation	R/W
IDLE	IDLE mode start	0 : CPU, WDT operate 1 : CPU, WDT halt (IDLE mode start)	

Note 1 : A reset is applied if XEN is cleared to "0".

Note 2 : Always set bit 6, 5 to "0".

*Note 3 : WDT ; Watchdog timer, * ; don't care*

Note 4 : Bits 3 to 0 in SYSCR2 are read in as "1" when a read instruction is executed.

Figure 1-17. System control registers 1, 2

1.8.5 Operating mode control

(1) STOP mode (STOP)

STOP mode is controlled by the system control register 1 and the $\overline{STOP}$ pin input. The $\overline{STOP}$ pin is also used both as a port P76 and an $\overline{INT5}$ (external interrupt input 5) pin. The STOP mode is started by setting STOP (bit 7 in SYSCR1) to "1". During STOP mode, the following status is maintained.

① High-frequency oscillations are turned off, and all internal operations are halted.
② The data memory, registers (except for DBR), PSW, and port output latches are all held in the status in effect before STOP mode was entered. The port output can select either output hold or high-impedance by setting OUTEN (bit 4 in SYSCR1).
③ The divider of the timing generator is cleared to "0".
④ The program counter holds the address of the instruction after the following instruction which started the STOP mode. [for example, SET (SYSCR1)]

STOP mode includes a level sensitive release mode and an edge-sensitive release mode, either of which can be selected with RELM (bit 6 in SYSCR1).

a. Level-sensitive release mode (RELM = 1)

In this mode, $\overline{STOP}$ mode is released by setting the STOP pin high. This mode is used for capacitor back-up when the main power supply is cut off and long term battery back-up.
When the $\overline{STOP}$ pin input is high, executing an instruction which starts the STOP mode will not place in STOP mode but instead will immediately start the release sequence (WARM-up). Thus, to confirm that the $\overline{STOP}$ pin input is low. The following method can be used for confirmation: Using an external interrupt input INT5 ($\overline{INT5}$ is a falling edge-sensitive input).

Example : Starting STOP mode with an INT5 interrupt.

```
PINT5 :   TEST   (P7) . 6              ; To reject noise, the STOP mode does not start if
          JRS    F, SINT5                port P76 is at high.
          LD     (SYSCR1), 01000000B   ; Sets up the level-sensitive release mode.
          SET    (SYSCR1) . 7          ; Starts STOP mode
          LDW    (IL), 1110011101010111B ; IL7, 5, 3 ← 0 (clears interrupt latches)
SINT5 :   RETI
```

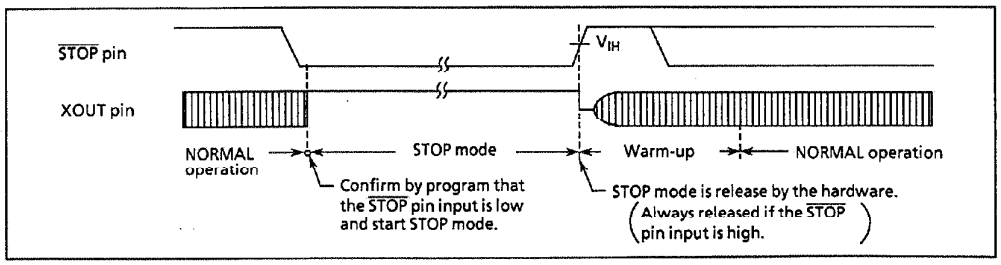


Figure 1-18. Level-sensitive release mode

Note 1 : After warm-up start, even if $\overline{STOP}$ pin input is low again, STOP mode does not restart.
Note 2 : When changing to the level-sensitive release mode from the edge-sensitive release mode, the release mode is not switched until a rising edge of the $\overline{STOP}$ pin input is detected.

b. Edge-sensitive release mode (RELM = "0")

In this mode, STOP mode is released by a rising edge of the $\overline{STOP}$ pin input. This is used in applications where a relatively short program is repeatedly executed at periodic intervals. This periodic signal (for example, a clock from a low-power consumption oscillator) is input to the $\overline{STOP}$ pin.

In the edge-sensitive release mode, STOP mode is started even when the $\overline{STOP}$ pin input is high.

Example : Starting STOP mode operation in the edge-sensitive release mode

```
LD      (SYSCR1), 00000000B     ; OUTEN ← 0 (specifies high-impedance)
DI                              ; IMF ← 0
SET     (SYSCR1). STOP          ; STOP ← 1 (activates STOP mode)
LDW     (IL), 1110011101010111B ; IL7, 5, 3 ← 0 (clears interrupt latches)
EI                              ; IMF ← 1
```

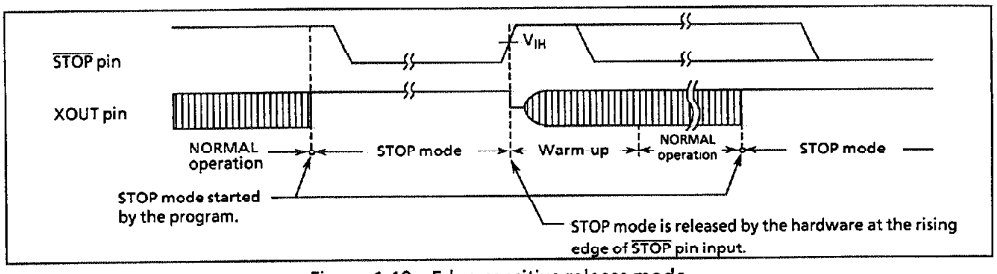


Figure 1-19. Edge-sensitive release mode

STOP mode is released by the following sequence:

① The oscillator is turned on.

② A warming-up period is inserted to allow oscillation time to stabilize. During warm-up, all internal operations remain halted. Four different warming-up times can be selected with WUT (bits 3 and 2 in SYSCR1) as determined by the resonator characteristics.

③ When the warming-up time has elapsed, normal operation resumes with the instruction following the STOP mode start instruction. The start is made after the divider of the timing generator is cleared to "0".

Table 1-2. Warming-up time example

Warming-up Time [ms]		
WUT	at fc = 4.194304 MHz	at fc = 8 MHz
00	46.87	24.57
01	15.62	8.19
10	11.73	6.15
11	3.91	2.05

Note : The warming-up time is obtained by dividing the basic clock by the divider: therefore, the warming-up time may include a certain amount of error if there is any fluctuation of the oscillation frequency when STOP mode is released. Thus, the warming-up time must be considered an approximate value.

STOP mode can also be released by setting the $\overline{RESET}$ pin low, which immediately performs the normal reset operation.

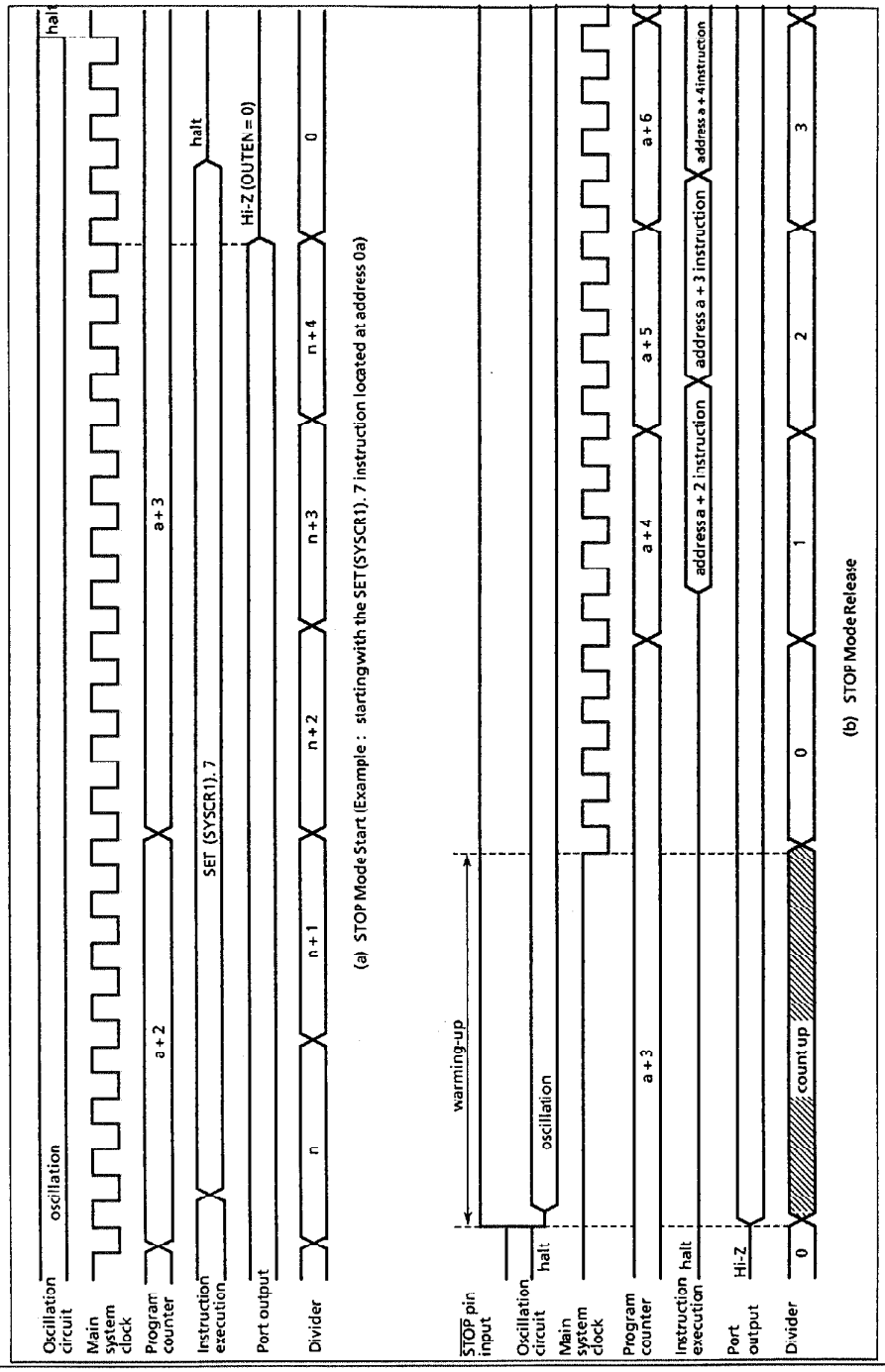


Figure 1-20. STOP mode start/release

Note : When STOP mode is released with a low hold voltage, the following cautions must be observed.
The power supply voltage must be at the operating voltage level before releasing STOP mode. The $\overline{\text{RESET}}$ pin input must also be high, rising together with the power supply voltage. In this case, if an external time constant circuit has been connected, the $\overline{\text{RESET}}$ pin input voltage will increase at a slower rate than the power supply voltage. At this time, there is a danger that a reset may occur if input voltage level of the $\overline{\text{RESET}}$ pin drops below the non-inverting high-level input voltage (hysteresis input).

(2) IDLE mode

IDLE mode is controlled by the system control register 2 (SYSCR2) and maskable interrupts. The following status is maintained during IDLE mode.

① Operation of the CPU and watchdog timer is halted. On-chip peripherals continue to operate.
② The data memory, CPU registers, PSW, and port output latches are all held in the status in effect before IDLE mode was entered.
③ The program counter holds the address of the instruction after the following instruction which started IDLE mode.

Example : Starting IDLE mode.

```
SET     (SYSCR2).4
```

IDLE mode includes a normal release mode and an interrupt release mode. Selection is made with the interrupt master enable flag (IMF). Releasing the IDLE mode returns to NORMAL.

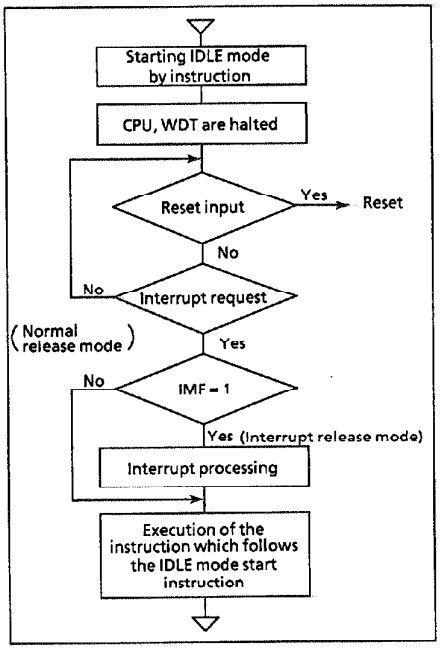


Figure 1-21. IDLE mode

a. Normal release mode (IMF = "0")

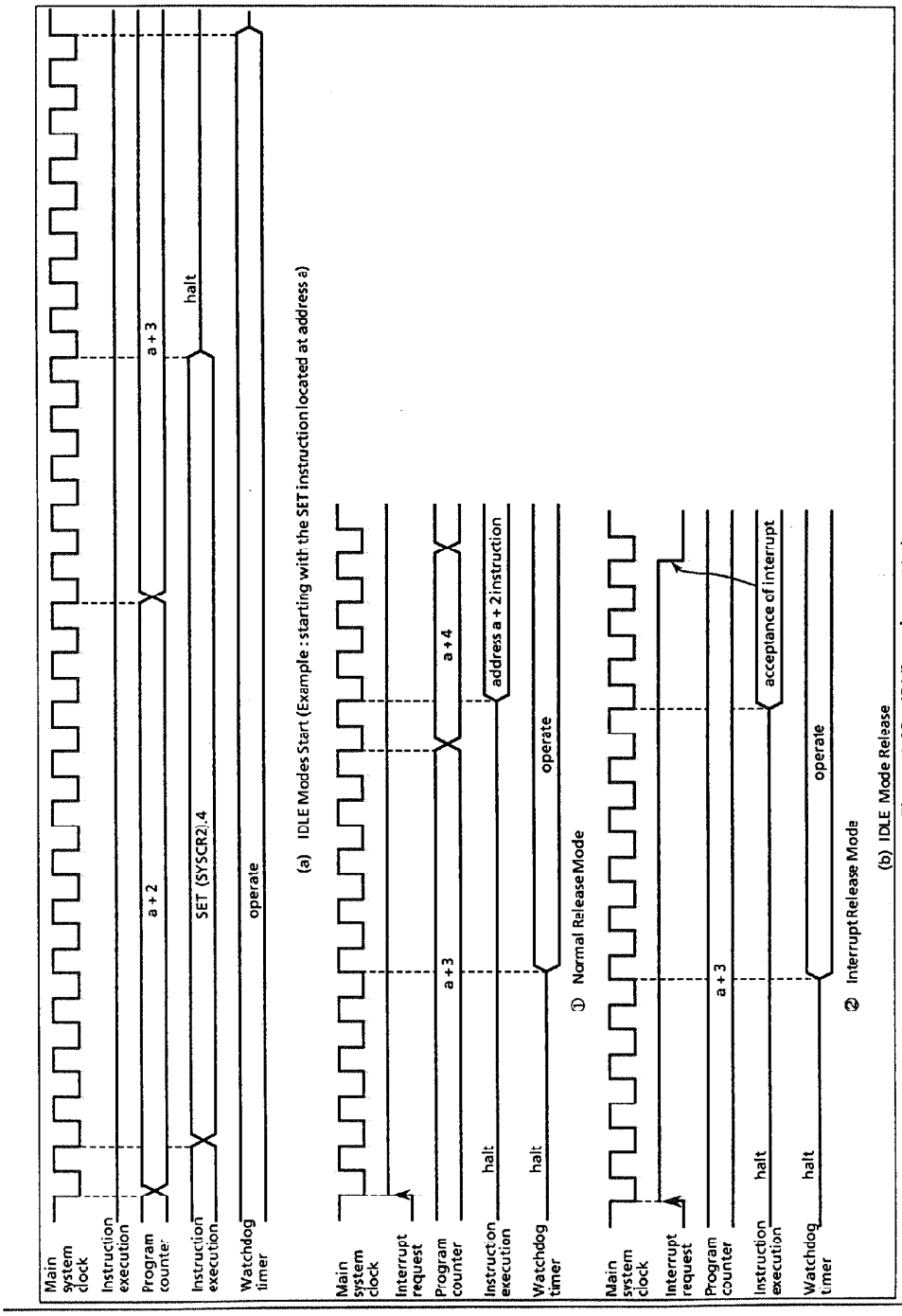
IDLE mode is released by any interrupt source enabled by the individual interrupt enable flag (EF) or an external interrupt 0 ($\overline{\text{INT0}}$) request. Execution resumes with the instruction following the IDLE mode start instruction. The interrupt latches (IL) of the interrupt source used for release is required to be cleared to "0" by load instruction.

b. Interrupt release mode (IMF = "1")

IDLE mode is released and interrupt processing is started by any interrupt source enabled with the individual interrupt enable flag (EF) or an external interrupt 0 ($\overline{\text{INT0}}$) request. After the interrupt is processed, the execution resumes from the instruction following the instruction which started IDLE mode.

IDLE mode can also be released by setting the $\overline{\text{RESET}}$ pin low, which immediately performs the reset operation.

Note : When a watchdog timer interrupt is generated immediately before the IDLE mode is started, the watchdog timer interrupt will be processed by IDLE mode will not be started.



87C405A-23

Figure 1-22. IDLE mode start/release

1.9 Interrupt Controller

The 87C405A has a total of 9 interrupt sources: 4 externals and 5 internals. Nested interrupt control with priorities is also possible. Two of the internal sources are pseudo non-maskable interrupts; the remainder are all maskable interrupts.

Interrupt latches (IL) that hold the interrupt requests are provided for interrupt sources. Each interrupt vector is independent. The interrupt latch is set to "1" when an interrupt request is generated and requests the CPU to accept the interrupt. The acceptance of maskable interrupts can be selectively enabled and disabled by the program using the interrupt master enable flag (IMF) and the individual interrupt enable flags (EF). When two or more interrupts are generated simultaneously, the interrupt is accepted in the highest priority order as determined by the hardware. Figure 1-23 shows the interrupt controller.

Table 1-3. Interrupt sources

Interrupt Source			Enable Condition	Interrupt Latch	Vector Table Address	Priority
Internal / External	(Reset)		Non-Maskable	—	$FFFE_H$	High 0
Internal	INTSW	(Software interrupt)	Pseudo non-maskable	—	$FFFC_H$	1
Internal	INTWDT	(Watchdog timer interrupt)		IL_2	$FFFA_H$	2
External	INT0	(External interrupt 0)	IMF = 1, INT0EN = 1	IL_3	$FFF8_H$	3
Internal	INTTC1	(16-bit timer / counter 1 interrupt)	IMF · EF_4 = 1	IL_4	$FFF6_H$	4
External	INT1	(External interrupt 1)	IMF · EF_5 = 1	IL_5	$FFF4_H$	5
Internal	INTTBT	(Time base timer interrupt)	IMF · EF_6 = 1	IL_6	$FFF2_H$	6
External	INT2	(External interrupt 2)	IMF · EF_7 = 1	IL_7	$FFF0_H$	7
RESERVED						
RESERVED						
RESERVED						
RESERVED						
RESERVED						
RESERVED						
Internal	INTTC2	(16-bit timer / counter 2 interrupt)	IMF · EF_{14} = 1	IL_{14}	$FFE2_H$	8
External	INT5	(External interrupt 5)	IMF · EF_{15} = 1	IL_{15}	$FFE0_H$	Low 9

(1) Interrupt latches (IL_{15} to IL_2)

Interrupt latches are provided for each source, except for a software interrupt. The latch is set to "1" when an interrupt request is generated, and requests the CPU to accept the interrupt. The latch is cleared to "0" just after the interrupt is accepted. All interrupt latches are initialized to "0" during reset.

The interrupt latches are assigned to addresses 003C and 003DH in the SFR. Each latch can be cleared to "0" individually by an instruction; however, the read-modify-write instruction such as bit manipulation or operation instructions cannot be used (Do not clear IL2 for a watchdog timer interrupt to "0"). Thus, interrupt requests can be canceled and initialized by the program. Note that interrupt latches cannot be directly set to "1" by any instruction. The contents of interrupt latches can be read out by an instruction. Therefore, testing interrupt requests by software is possible.

```
Example 1 : Clears interrupt latches
            LDW     (IL), 1111110100111111B ; IL9 to IL6←0
Example 2 : Reads interrupt latches
            LD      WA, (IL)                ; W←ILH, A←ILL
Example 3 : Tests an interrupt latch
            TEST    (IL).7                  ; IL7 = 1 then jump
            JR      F, SSET
                .
                .
                .
                .
                .
            SSET:
```

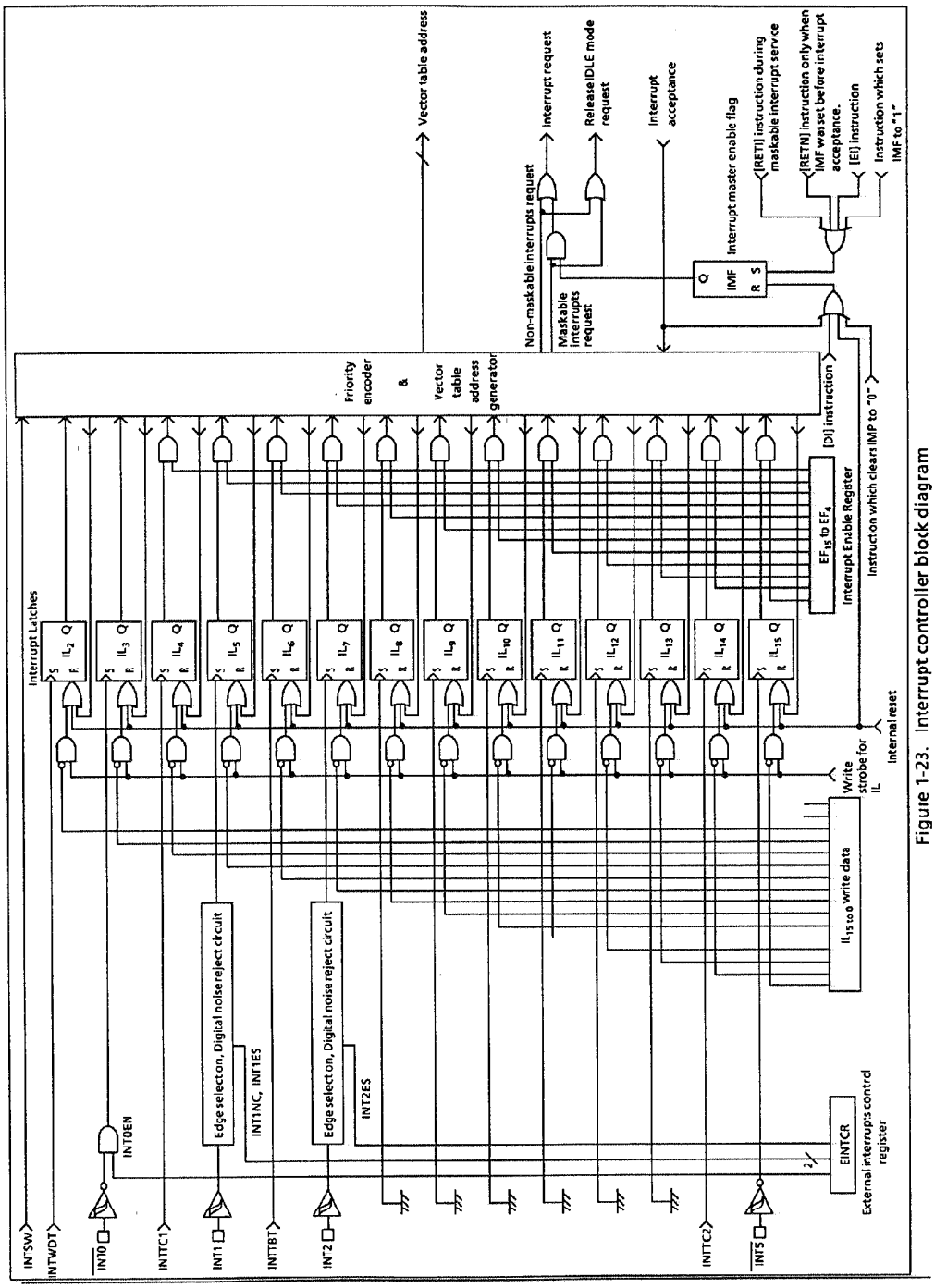


Figure 1-23. Interrupt controller block diagram

87C405A-26

(2) Interrupt enable register (EIR)

The interrupt registers (EIR) enable and disable the acceptance of interrupts except for the pseudo non-maskable interrupts (software and watchdog timer interrupts). Pseudo non-maskable interrupts are accepted regardless of the contents of the EIR; however, the pseudo non-maskable interrupts cannot be nested more than once at the same time.

The EIR consists of an interrupt master enable flag (IMF) and individual interrupt enable flags (EF). These registers are assigned to addresses $003A_H$ and $003B_H$ in the SFR, and can be read and written by an instruction (including read-modify-write instructions such as bit manipulation instructions).

① Interrupt master enable flag (IMF)

The interrupt master enable flag (IMF) enables and disables the acceptance of all interrupts. Clearing this flag to "0" disables the acceptance of all maskable interrupts. Setting to "1" enables the acceptance of interrupts.

When an interrupt is accepted, this flag is cleared to "0" to temporarily disable the acceptance of maskable interrupts. After execution of the interrupt service program, this flag is set to "1" by the maskable interrupt return instruction [RETI] to again enable the acceptance of interrupts. If an interrupt request has already occurred, interrupt service starts immediately after execution of the [reti] instruction.

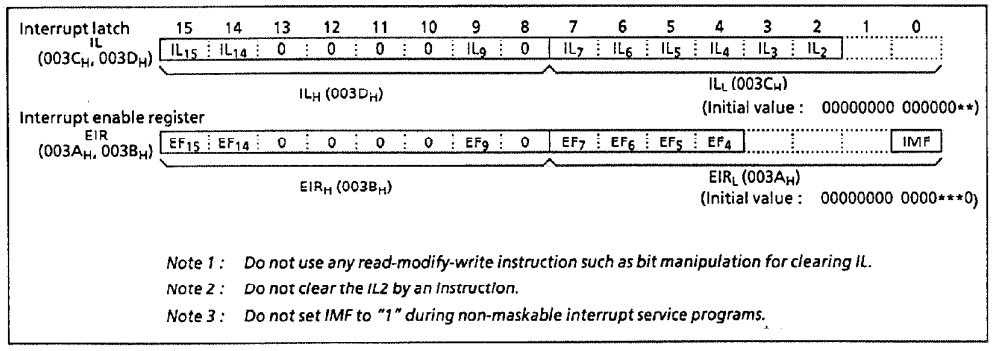
Pseudo non-maskable interrupts are returned by the [RETN] instruction. In this case, the IMF is set to "1" only when pseudo non-maskable interrupt service is started with interrupt acceptance enabled (IMF = 1). Note that IMF remains "0" when cleared in the interrupt service program.

The IMF is assigned to bit 0 at address $003A_H$ in the SFR, and can be read and written by an instruction. IMF is normally set and cleared by the [EI] and [DI] instructions, and the IMF is initialized to "0" during reset.

② Individual interrupt enable flags (EF_{15} to EF_4)

These flags enable and disable the acceptance of individual maskable interrupts, except for an external interrupt 0. Setting the corresponding bit of an individual interrupt enable flag to "1" enables acceptance of an interrupt, setting the bit to "0" disables acceptance.

```
Example 1 :  Sets EF for individual interrupt enable, and sets IMF to "1"
             LDW     (EIR), 1100000010100001B ;  EF15, EF14, EF7, EF5, IMF←1
Example 2 :  Sets an individual interrupt enable flag to "1"
             SET     (EIRH).1                 ;  EF9←1
```



Note 1 : Do not use any read-modify-write instruction such as bit manipulation for clearing IL.

Note 2 : Do not clear the IL2 by an instruction.

Note 3 : Do not set IMF to "1" during non-maskable interrupt service programs.

Figure 1-24. Interrupt latch (IL) and interrupt enable register (EIR)

1.9.1 Interrupt sequence

An interrupt request is held until the interrupt is accepted or the interrupt latch is cleared to "0" by a reset or an instruction. Interrupt acceptance sequence requires 8 machine cycles after the completion of the current instruction execution. The interrupt service task terminates upon execution of an interrupt return instruction [RETI] (for maskable interrupts or [RETN](pseudo non-maskable interrupts). Figure 1-25 shows the timing chart of interrupt acceptance and interrupt return instruction.

(1) Interrupt acceptance processing

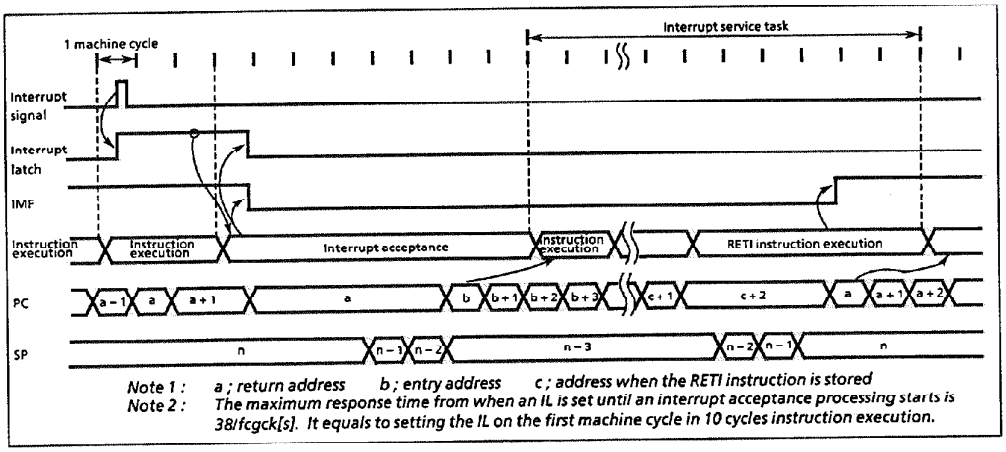
① The Interrupt master enable flag (IMF) is cleared to "0" to temporarily disable the acceptance of any following maskable interrupts. When a non-maskable interrupt is accepted, the acceptance of any following interrupts is temporarily disabled.

② The interrupt latch (IL) for the interrupt source accepted is cleared o "0".

③ The contents of the program counter and the program status word are saved (pushed) onto the stack. (pushed down in order of PSW, PC_H, PC_L). The contents of Stack Pointer (SP) is decreased by 3.

④ The entry address of the interrupt service program is read from the vector table address corresponding to the interrupt source, and the entry address is loaded to the program counter.

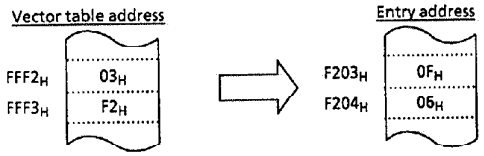
⑤ The instruction stored at the entry address of the interrupt service program is executed.



Note 1 : a ; return address b ; entry address c ; address when the RETI instruction is stored
Note 2 : The maximum response time from when an IL is set until an interrupt acceptance processing starts is 38/fcgck[s]. It equals to setting the IL on the first machine cycle in 10 cycles instruction execution.

Figure 1-25. Timing chart of interrupt acceptance and interrupt return instruction

Example : Correspondence between vector tale address for INTTBT and the entry address of the interrupt service program.



A maskable interrupt is not accepted until the IMF is set to "1" even if a maskable interrupt of higher priority than that of the current interrupt interrupt being serviced. When nested interrupt service is necessary, the IMF is set to "1" in the interrupt service program. In this case, acceptable interrupt sources are selectively enabled by the individual interrupt enable flags. However, an acceptance of external interrupt 0 cannot be disabled by the EF, therefore, if disablement is necessary, either the external interrupt function must be disabled with the external interrupt control register (INT0EN) or interrupt processing must be avoided by the program. (When INT0EN = 0, the interrupt latch IL3 is not set, therefore, the falling edge of the INT0 pin input cannot be detected.)

Example 1 : Disables an external interrupt 0 using INT0EN:

```
CLR     (EINTCR). INT0EN    ; INT0EN←0
```

Example 2 : Disables the processing of external interrupt 0 under the software control (using bit 0 at address 00F0H as the interrupt processing disable switch):

```
PINT0 :  TEST  (00F0H) . 0  ; Returns without interrupt processing if (00F0H)0 = 1.
         JRS   T, SINT0
         RETI
SINT0 :  interrupt processing
         RETI
           :
VINT0 :  DW    PINT0
```

(2) General-purpose register save / restore processing

During interrupt acceptance processing, the program counter and the program status word are automatically saved on the stack, but not the accumulator and other registers. These registers are saved by the program if necessary. Also, when nesting multiple interrupt services, it is necessary to avoid using the same data memory area for saving registers.

The following method is used to save / restore the general-purpose registers:

① General-purpose register save / restore by register bank changeover:

General-purpose registers can be saved at high-speed by switching to a register bank that is not in use. Normally, bank 0 is used for the main task and banks 1 to 15 are assigned to interrupt service tasks. To increase the efficiency of data memory utilization, the same bank is assigned for interrupt sources which are not nested.

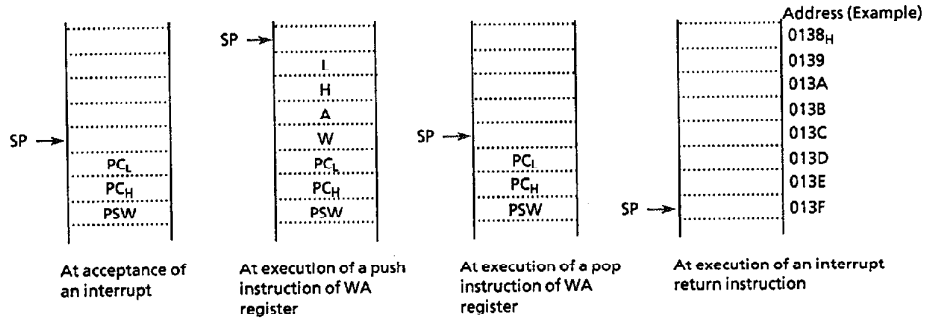
The switched bank is automatically restored by executing an interrupt return instruction [RETI] or [RETN]. Therefore, it is not necessary for a program to save the RBS.

Example : Register Bank Changeover

```
PINTxx :  LD     RBS, n      ; Switches to bank n (1 μs at 8 MHz)
          Interrupt processing
          RETI               ; Restores bank and Returns
```

② General-purpose register save / restore using push and pop instructions:
To save only a specific register, and when the same interrupt source occurs more than once, the general-purpose registers can be saved/restored using push/pop instructions.

```
Example :    Register save using push and pop instructions
PINTxx :  PUSH    WA          ; Save WA register pair
          PUSH    HL          ; Save HL register pair
          Interrupt processing
          POP     HL          ; Restore HL register pair
          POP     WA          ; Restore WA register pair
          RETI                ; Return
```



③ General-purpose registers save/restore using data transfer instruction:
Data transfer instructions can be used to save only a specific general-purpose register during processing of a single interrupt.

```
Example :    Saving / Restoring registers by data memory transfer instructions
PINTxx :  LD      (GSAVA), A    ; Save A register
          Interrupt processing
          LD      A, (GSAVA)    ; Restore A register
          RETI                  ; Return
```

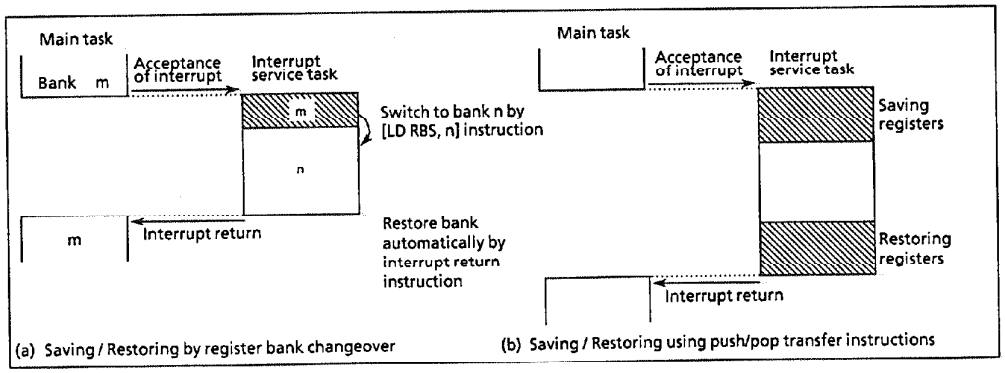


Figure 1-26. Saving / restoring general-purpose registers

(3) Interrupt return

The interrupt return instructions perform the following operations.

[RETI] Maskable interrupt return	[RETN] Non-maskable interrupt return
① The contents of the program counter and the program status word are restored from the stack.	① The contents of the program counter and program status word are restored from the stack.
② The stack pointer is incremented 3 times.	② The stack pointer is incremented 3 times.
③ The interrupt master enable flag is set to "1".	③ The interrupt master enable flag is set to "1" only when a non-maskable interrupt is accepted in interrupt enable status. However, the interrupt master enable flag remains at "0" when so clear by an interrupt service program.

Interrupt requests are sampled during the final cycle of the instruction being executed. Thus, the next interrupt can be accepted immediately after the interrupt return instruction is executed.

Note : When the interrupt processing time is longer than the interrupt request generation time, the interrupt service is performed but not the main task.

1.9.2 Software interrupt (INTSW)

Executing the [SWI] instruction generates a software interrupt and immediately starts interrupt processing (INTSW is highest prioritized interrupt). However, if processing of a non-maskable interrupt is already underway, executing the SWI instruction will not generate a software interrupt but will result in the same operation as the [NOP] instruction.

Note : Software interrupt generates during non-maskable interrupt processing to use SWI instruction for software break in a development tool.

Use the [SWI] instruction only for detection of the address error of for debugging.

① Address error detection

FF_H is read if for some cause such as noise the CPU attempts to fetch an instruction from a non-existent memory address. Code FF_H is the SWI instruction, so a software interrupt is generated and an address error is detected. The address error detection range can be further expanded by writing FF_H to unused areas of the program memory. Address-trap-reset is generated for instruction fetch from a part of RAM area (addresses 0040 to $013F_H$) or SFR area (0000 to $003F_H$).

Note : The fetch data from addresses, BF80 to $BFFF_H$ for 87C405A and 87P808 is not "FF_H", because the outgoing test ROM is contained.

② Debugging

Debugging efficiency can be increased by placing the SWI instruction at the software break point setting address.

1.9.3 External interrupts

The 87C405A has four external interrupt inputs. Two of these are equipped with digital noise rejection circuits(pulse inputs of less than a certain time are eliminated as noise).

Edge selection is also possible with INT1, INT2 pin. The $\overline{INT0}$ / P10 pin can be configured as either an external interrupt input pin or an input/output port, and is configured as an input port during reset.

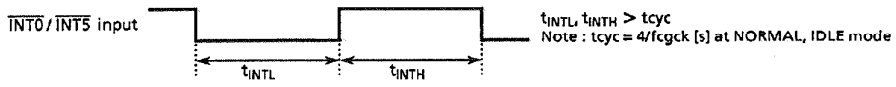
Edge selection, noise rejection control and $\overline{INT0}$ / P10 pin function selection are performed by the external interrupt control register.

Table 1-4. External interrupts

Source	Pin	Secondary function pin	Enable conditions	Edge	Digital noise rejection circuit
INT0	$\overline{INT0}$	P10	IMF = 1, INT0EN = 1	Falling edge	– (Hysteresis input)
INT1	INT1	P11	IMF · EF_5 = 1	Falling edge or Rising edge	Pulses of less than 15/fc or 63/fc[s] are eliminated as noise. Pulses equal to or more than 48/fc [s] or 192/fc [s] are regarded as signals.
INT2	INT2	P12 / TC1	IMF · EF_7 = 1		Pulses of less than 7/fc[s] are eliminated as noise. Pulses equal to or more than 24/fc [s] are regarded as signals.
INT5	$\overline{INT5}$	P76 / $\overline{STOP}$	IMF · EF_{15} = 1	Falling edge	– (Hysteresis input)

Note 1 : The noise rejection function is also affected to detect the edge of Timer / Counter input (TC1 pin).

Note 2 : The pulse width (both "H" and "L" level) for input to the $\overline{INT0}$ and $\overline{INT5}$ pins must be over 1 machine cycle.



Note 3 : If a noiseless signal is input to the external interrupt pin in the NORMAL or IDLE mode, the maximum time from the edge of input signal until the IL is set is as follows:

① INT1 pin 49/fc [s] (at INT1NC = 1), 193/fc [s] (at INT1NC = 0)

② INT2 pin 25/fc [s]

Note 4 : When INT0EN = 0, the interrupt latch IL3 is not set even if the falling edge of INT0 pin input is detected.

```
Example :  Activating stop mode
           LD    (SYSCR1), 01000000B        ;  OUTEN←0 (Specifies High-impedance)
           DI                               ;  IMF←0
           SET   (SYSCR1), STOP             ;  STOP←1 (Activates STOP mode)
           LDW   (IL), 1111111101010111B    ;  IL7, 5, 3←0 (Clears interrupt latches)
           EI                               ;  IMF←1
```

EINTCR (0037_H)

7	6	5	4	3	2	1	0
INT1NC	INT0EN				INT2ES	INT1ES	

(Initial value : 00*0 000*)

INT1NC	INT1 noise reject time select	0 : Pulses of less than 63/fc [s] are eliminated as noise 1 : Pulses of less than 15/fc [s] are eliminated as noise	R/W
INT0EN	P10 / $\overline{INT0}$ pin configuration	0 : P10 input / output port 1 : $\overline{INT0}$ pin (Port P10 should be set to an input mode.)	
INT2 ES INT1 ES	INT2, INT1 edge select	0 : Rising edge 1 : Falling edge	

*Note : fc ; High-frequency clock [Hz] * ; don't care*

Figure 1-27. External interrupt control register

1.10 Watchdog Timer (WDT)

The Watchdog Timer rapidly detects the CPU malfunction such as endless looping caused by noise or the like, and resumes the CPU to the normal state.

The Watchdog Timer signal for detecting malfunction can be selected either a reset output or a non-maskable interrupt request. However, selection is possible only once after reset.

After reset, the signal is initialised to the reset output.

When the Watchdog Timer is not being used for malfunction detection, it can be used as a timer to generate an interrupt at fixed intervals.

1.10.1 Watchdog timer configuration

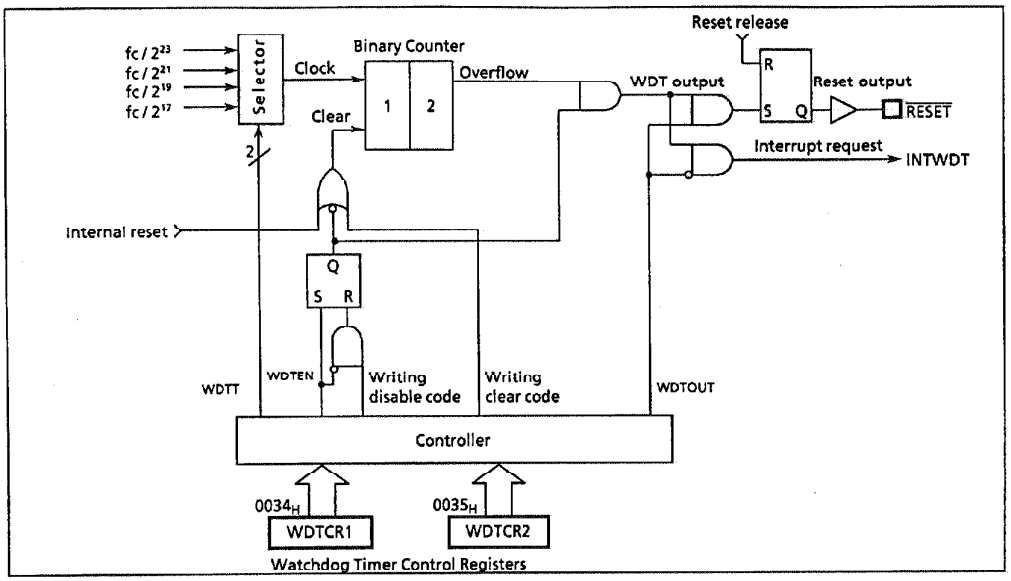


Figure 1-28. Watchdog timer configuration

1.10.2 Watchdog timer control

Figure 1-29 shows the Watchdog Timer control registers. The Watchdog Timer is automatically enabled after reset.

(1) Malfunction detection methods using the watchdog timer

The CPU malfunction is detected as follows.

① Setting the detection time, selecting output, and clearing the binary counter.

② Repeatedly clearing the binary counter within the setting detection time.

If the CPU malfunction occurs for any cause, the Watchdog Timer output will become active at the rising of an overflow from the binary counters unless the binary counters are cleared. At this time, when WDTOUT = 1 a reset is generated, which drives the $\overline{\text{RESET}}$ pin to reset the internal hardware. When WDTOUT = 0, a Watchdog Timer interrupt (INTWDT) is generated.

The Watchdog Timer temporarily stops counting in the STOP mode including warm-up or IDLE mode, and automatically restarts (continues counting) when the STOP / IDLE mode is released.

Example : Sets the Watchdog Timer detection time to 2^{21}/fc[s] and resets the CPU malfunction.

```
        LD      (WDTCR2), 4EH       ; Clears the binary counters
        LD      (WDTCR1), 00001101B ; WDTT←10, WDTOUT←1
Within 3/4
of WDT  LD      (WDTCR2), 4EH       ; Clears the binary counters
detection                             (Always clear immediately after changing WDTT)
time
        LD      (WDTCR2), 4EH       ; Clears the binary counters
Within 3/4
of WDT
detection
time    LD      (WDTCR2), 4EH       ; Clears the binary counters
```

Watchdog Timer Control Register 1

WDTCR1 (0034_H)

7	6	5	4	3	2	1	0
				WDTEN	WDTT		WDTOUT

(Initial value **** 1001)

WDTEN	Watchdog timer enable / disable	0 : Disable (It is necessary to write the disable code to WDTCR2.) 1 : Enable	write only
WDTT	Watchdog timer detection time	00 : 2^{25}/fc [s] 01 : 2^{23}/fc 10 : 2^{21}/fc 11 : 2^{19}/fc	
WDTOUT	Watchdog timer output select	0 : Interrupt request 1 : Reset output	

Note 1 : WDTOUT cannot be set to "1" by program after clearing WDTOUT to "0".
*Note 2 : fc ; High-frequency clock[Hz] * ; don't care*
Note 3 : WDTCR1 is a write-only register and must not be used with any of read-modify-write instructions.
Note 4 : Disable the Watchdog Timer or clear the counter just before switching to STOP mode. When the counter is cleared just before switching to STOP mode, clear the counter again subsequently to releasing STOP mode.

Watchdog Timer Control Register 2

WDTCR2 (0035_H)

7	6	5	4	3	2	1	0

(Initial value **** ****)

WDTCR2	Watchdog Timer control code write register	$4E_H$: Watchdog Timer binary counter clear (clear code) $B1_H$: Watchdog Timer disable (disable code) Others : invalid	write only

Note 1 : The disable code is invalid unless written when WDTEN = 0.
*Note 2 : * ; don't care*
Note 3 : Since WDTCR2 is a write-only register, read-modify-write instructions (e.g., bit manipulating instructions such as SET or CLR and arithmetic instructions such as AND or OR) cannot be used for read / write to this register.
Note 4 : To clear binary counter doesn't initialize the source clock, therefore, it is recommended to clear binary counter within 3/4 of the detection period.

Figure 1-29. Watchdog timer control registers

(2) Watchdog timer enable

The Watchdog Timer is enabled by setting WDTEN (bit 3 in WDTCR1). WDTEN is initialized to "1" during reset, so the Watchdog Timer operates immediately after reset is released.

(3) Watchdog timer disable

The Watchdog Timer is disabled by writing the disable code ($B1_H$) to WDTCR2 after clearing WDTEN (bit 3 in WDTCR1) to "0". The Watchdog Timer is not disabled if this procedure is reversed and the disable code is written to WDTCR2 before WDTEN is cleared to "0". During disabling the Watchdog Timer , the binary counters are cleared to "0".

Example: Disables Watchdog Timer

```
LDW  (WDTCR1), 0B101H    ; WDTEN←0, WDTCR2←disable code
```

Table 1-5. Watchdog timer detection time

Detection time	
WDTT	fc = 8 MHz
00	4.194 s
01	1.048 s
10	262.1 ms
11	65.5 ms

1.10.3 Watchdog timer interrupt (INTWDT)

This is a pseudo non-maskable interrupt which can be accepted regardless of the contents of the EIR. If a Watchdog Timer interrupt or a software interrupt is already accepted, however, the new Watchdog Timer interrupt waits until the previous interrupt processing is completed (the end of the [RETN] instruction execution).

The stack pointer (SP) should be initialized before using the Watchdog Timer output as an interrupt source with WDTOUT.

Example: Watchdog Timer interrupt setting up.

```
LD  SP, 013FH              ;  Sets the stack pointer
LD  (WDTCR1), 00001000B    ;  WDTOUT←0
```

1.10.4 Watchdog timer reset

If the Watchdog Timer output becomes active, a reset is generated, which drives the $\overline{\text{RESET}}$ pin low to reset the internal hardware. The reset output time is 12/fcgck to 16/fcgck [s] (1.5 to 2.0 μs at 8 MHz, 3.0 to 4.0 μs at 4 MHz, gear ratio 1/1). The $\overline{\text{RESET}}$ pin is sink open drain input / output with pull up resistor.

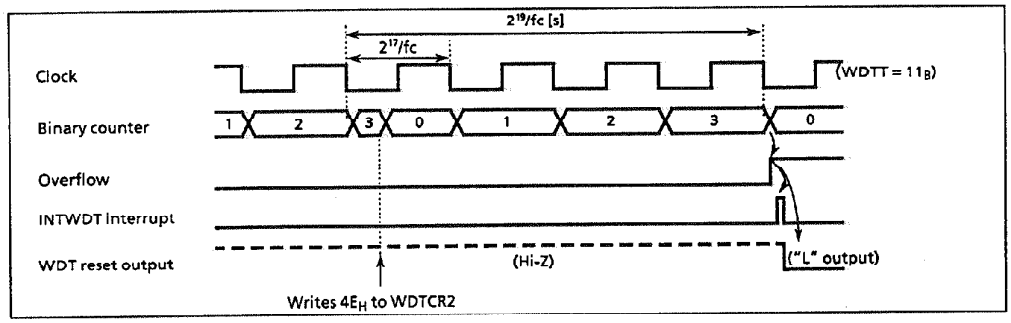


Figure 1-30. Watchdog timer interrupt / reset

1.11 Reset Circuit

87C405A has four types of reset generation procedures: an External reset input, an Address trap reset, a Watchdog Timer reset and a System clock reset.

Table 1-6 shows on-chip hardware initialization by reset action.

The internal source reset circuit (Watchdog Timer reset, Address trap reset, and System clock reset) is not initialized when power is turned on. Thus, output from the $\overline{\text{RESET}}$ pin may go low (maximum 16/fc[s] (2 μs at 8 MHz, 4 μs at 4 MHz).

Table 1-6. On-chip hardware initialization by reset action

On-chip Hardware		Initial Value	On-chip Hardware	Initial Value
Program counter	(PC)	$(FFFF_H)$ - $(FFFE_H)$	Prescaler and Divider of Timing generator	0
Register bank selector	(RBS)	0	Watchdog Timer	Enable
Jump status flag	(JF)	1		
Interrupt master enable flag	(IMF)	0	Output latches of input/output port	Refer to I/O port circuitry
Interrupt individual enable flags	(EF)	0		
Interrupt latches	(IL)	0	Control register	Refer to control registers

1.11.1 External reset input

The $\overline{\text{RESET}}$ pin contains a hysteresis input with an internal pull-up resistor. When the $\overline{\text{RESET}}$ pin is held at low for at least 3 machine cycles (12/fcgck [s]) with the power supply voltage within the operating voltage range and oscillation stable, a reset is applied and the internal state is initialized.

When the $\overline{\text{RESET}}$ pin input goes high, the reset operation is released and the program execution starts at the vector address stored at addresses FFFE to $FFFF_H$.

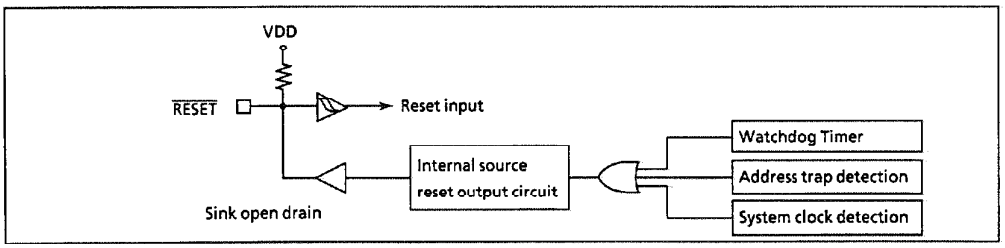
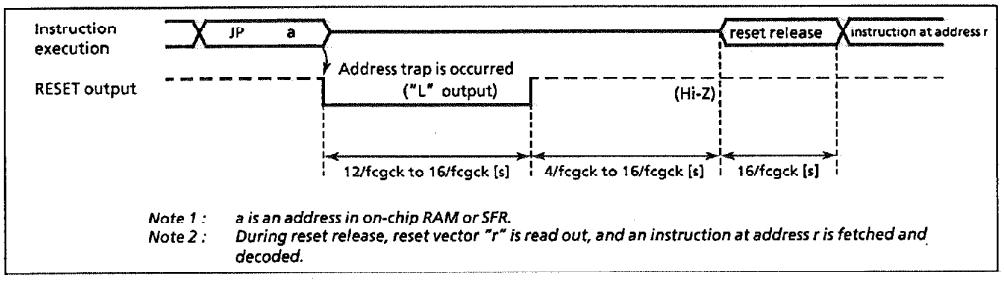


Figure 1-31. Reset circuit

1.11.2 Address trap reset

An Address trap reset is one of fail-safe function that detects CPU malfunction such as endless looping caused by noise or the like. If the CPU attempts to fetch an instruction from a part of RAM or SFR, an internal reset will be generated. Then, the $\overline{\text{RESET}}$ pin output will go low. The reset time is 12/fcgck to 16/fcgck [s] (1.5 to 2.0 μs at 8MHz, 3.0 to 4.0 μs at 4MHz).



Note 1 : a is an address in on-chip RAM or SFR.
Note 2 : During reset release, reset vector "r" is read out, and an instruction at address r is fetched and decoded.

Figure 1-32. Address trap reset

1.11.3 Watchdog timer reset

Refer to Section "1.6 Watchdog Timer".

1.11.4 System clock reset

Clearing XEN to "0" stops a system clock, and causes CPU to deadlock. This can be prevented by automatically generating a reset signal whenever XEN = 0 is detected to continue the oscillation. Then the $\overline{\text{RESET}}$ pin output goes low . The reset time is 12/fcgck to 16/fcgck [s] (1.5 μs ~ 2.0 μs at 8MHz, 3.0 to 4.0 μs at 4MHz).

2. On-chip Peripherals Functions

2.1 Special function register (SFR)

The TLCS-870 Series uses the memory mapped I/O system, and all peripheral control and data transfers are performed through the special function registers (SFR) .

The SFR are mapped to addresses 0000 to $003F_H$. Figure 2-1 shows the 87C405 SFR.

Address	Read	Write
0000_H	reserved	
01	P1 port	
02	reserved	
03	"	
04	"	
05	"	
06	P6 port	
07	P7 port	
08	reserved	
09	"	
0A	–	–
0B	–	P1CR (P1 port I/O control)
0C	–	P6CR (P6 port I/O control)
0D	–	P7CR1 (P7 port I/O control 1)
0E	reserved	
0F	reserved	
10	–	$TREG1A_L$ (Timer register 1A)
11	–	$TREG1A_H$
12	$TREG1B_L$ (Timer register 1B)	
13	$TREG1B_H$	
14	–	TC1CR (TC1 control)
15	–	TC2CR (TC2 control)
16	–	$TREG2_L$ (Timer register 2)
17	–	$TREG2_H$
18	reserved	
19	"	
1A	"	
1B	"	
1C	"	
1D	"	
1E	"	
1F	"	

Address	Read	Write
0020_H	reserved	
21	reserved	
22	reserved	
23	"	
24	"	
25	"	
26	"	
27	P7CR2 (P7 port I/O control 2)	
28	reserved	
29	"	
2A	"	
2B	"	
2C	"	
2D	"	
2E	"	
2F	STOPCR (Key Wake-up control)	
30	CGCR (Clock Gear control)	
31	reserved	
32	"	
33	"	
34	–	WDTCR1 (Watchdog Timer control)
35	–	WDTCR2
36	TBTCR (TBT / TG / DVO control)	
37	EINTCR (External interrupt control)	
38	SYSCR1 (System control)	
39	SYSCR2	
3A	EIR_L (Interrupt enable register)	
3B	EIR_H	
3C	IL_L (Interrupt latch)	
3D	IL_H	
3E	reserved	
3F	PSW	RBS (Register bank selector)

(a) Special Function Register

Note 1 : Do not access reserved areas by the program.

Note 2 : – ; cannot be accessed.

Note 3 : When defining address $003F_H$ with assembler symbols, use GPSW and GRBS.

Note 4 : Write-only registers and interrupt latches cannot use the read-modify-write instructions (bit manipulation instructions such as SET, CLR, etc. and logical operation instructions such as AND, OR, etc.)

Note 5 : PSW ; Program Status Word

Figure 2-1. SFR

2.2 I/O Ports

The 87C405A has 3 ports, 22 pin input / output ports.

① P1 port ; 8-bit I/O port (External interrupt input, Timer/Counter input/output, and Divider output)
② P6 port ; 8-bit I/O port (STOP mode release input)
③ P7 port ; 6-bit I/O port (External interrupt and Timer/Counter input / output)

Each output port contains a latch, which holds the output data. All input ports do not have latches, so the external input data should either be held externally until read or reading should be performed several times before processing. Figure 2-2 shows input / output timing examples.

External data is read from an I/O port in the S1 state of the read cycle during execution of the read instruction. This timing can not be recognized from outside, so that transient input such as chattering must be processed by the program. Output data output changes in the S2 state of the write cycle during execution of the instruction which writes to an I/O port.

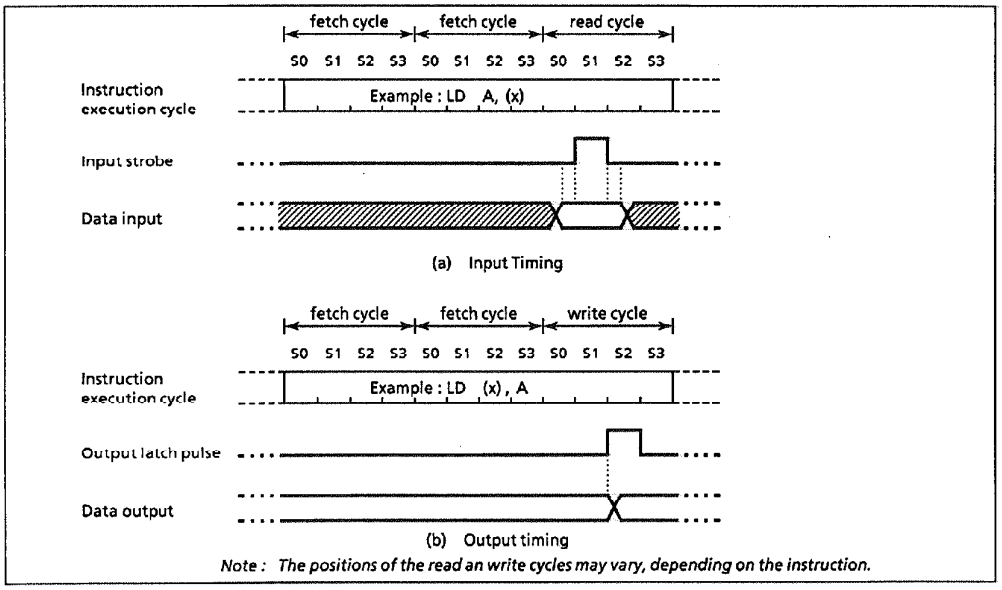


Figure 2-2. Input / output timing (Example)

When reading an I/O port except programmable I/O ports, whether the pin input data or the output latch contents are read depends on the instructions, as shown below :

(1) Instructions that read the output latch contents
① XCH r, (src)
② SET / CLR / CPL (src).b
③ SET / CLR / CPL (pp).g
④ LD (src).b, CF
⑤ LD (pp).b,CF
⑥ ADD / ADDC / SUB / SUBB / AND / OR / XOR (src), n
⑦ (src) side of ADD/ADDC/SUB/SUBB/AND/OR/XOR (src), (HL)

(2) Instructions that read the pin input data
Instructions other than the above (1) and (HL) side of ADD/ADDC/SUB/SUBB/AND/OR/XOR (scr), (HL)

2.2.1 Port P1(P17 to P10)

Port P1 is an 8-bit input / output port which can be configured as an input or an output in on-bit unit. Input / output mode is specified by the port P1 input/output control register (P1CR). During reset, the P1CR is initialized to "0", which configures port P1 as an input. The P1 output latches are also initialized to "0"Port P1 is also used as an External interrupt input, a Timer / Counter input, and a Divider output. When used as secondary function pin, the input pins should be set to the input mode, and the output pins should be set to the output mode and beforehand the output latch should be set to "1". It is recommended that pins P11 and P12 should be used as External interrupt inputs, Timer / Counter Input, or Input ports. The interrupt latch is set at the rising or falling edge of the output when used as output ports. Pin 10 can be configured as either an input / output ports with INT0EN or an External interrupt input. During reset, pin P10 is configured as an input port.

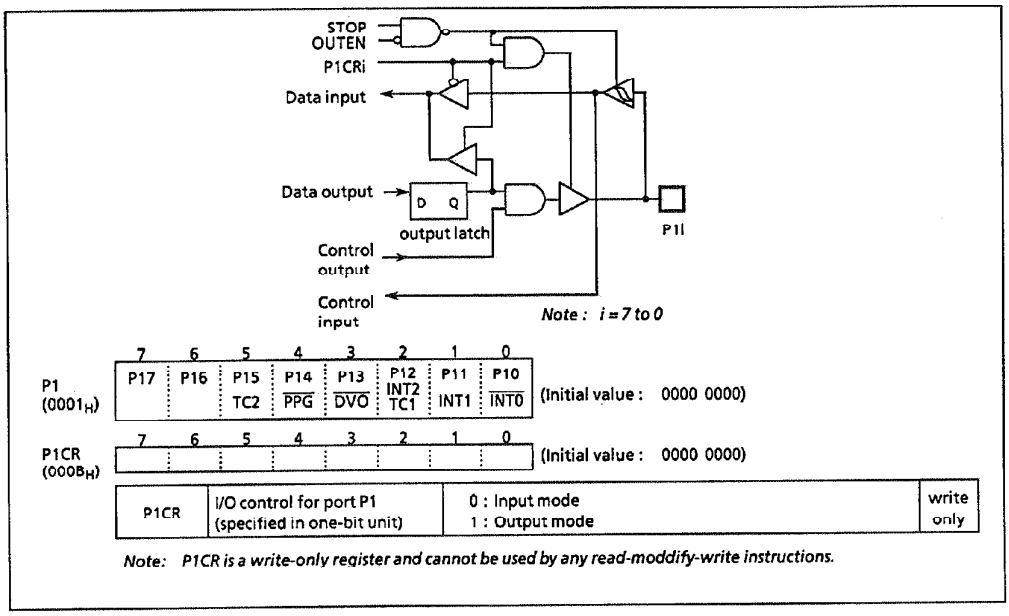


Figure 2-3. Port 1 and P1CR

Example : Sets P17, P16 and P14 as output ports, P13 and P11 as input ports, and the others as function pins. Internal output data is "1" for the P17 and P14 pins, and "0" for the P16 pin.

```
LD    (EINTCR), 01000000B      ; INT0EN←1
LD    (P1), 10111111B          ; P17←1, P14←1, P16←0
LD    (P1CR), 11010000B
```

Note : Ports set to the input mode read the pin states. When input pin and output in exist in port P1 together, the contents of the output latch of ports set to the input mode may be rewritten by executing the bit manipulation instructions. Pins set to output mode read a value of the output latch.

2.2.2 Port P6 (P67 to P60)

Port P6 is an 8-bit general-purpose input/output port which can be configured as an input or an output in one-bit unit. P62 to P65 are used as key wake-up inputs. Input / output mode is specified by port P6 input/output control register (P6CR) . During reset, P6CR is set to "0", port P6 is in input mode. During reset, the output latches of port P6 is initialized to "0". P6CR is write-only register. In the case of using port P6 as key wake-up inputs, please refer to Section "2.11 Key wake up".

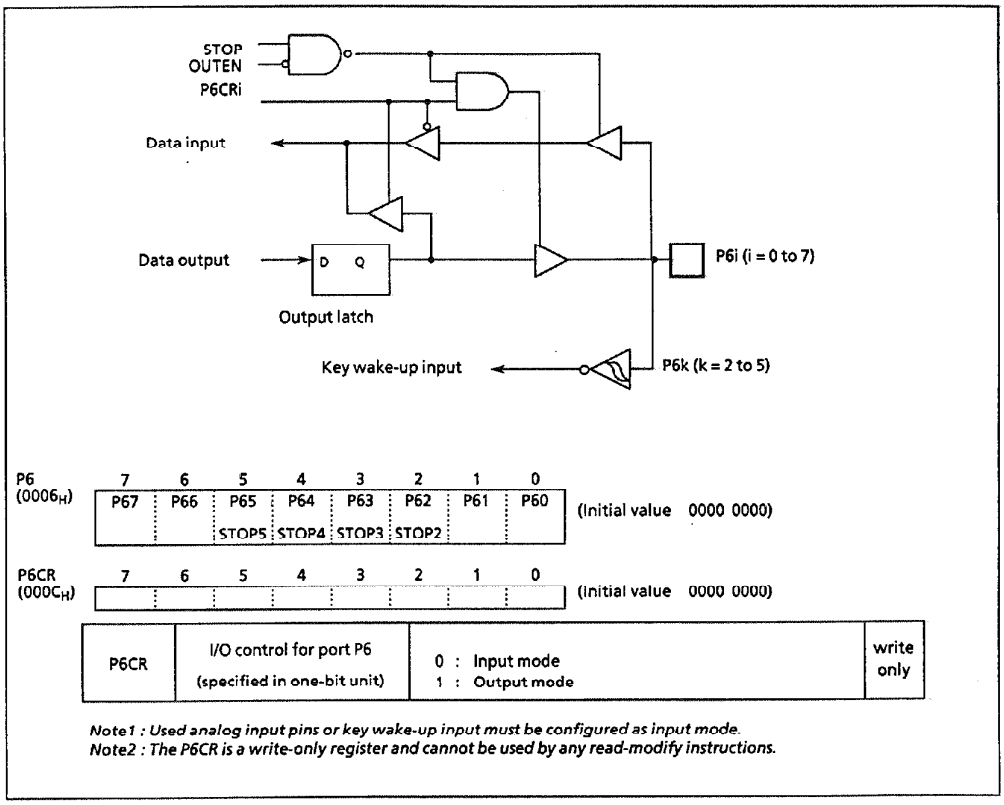
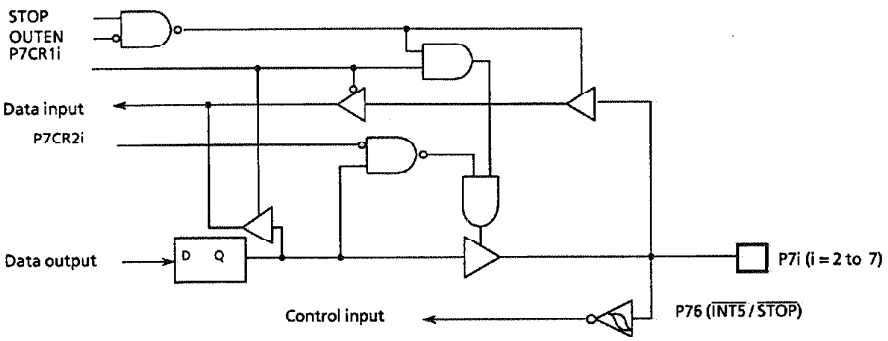


Figure 2-4. Port P6 and P6CR

Note : Ports set to the input mode read the pin states. When input pin and output in exist in port P1 together, the contents of the output latch of ports set to the input mode may be rewritten by executing the bit manipulation instructions. Pins set to output mode read a value of the output latch.

2.2.3 Port P7 (P77 to P72)

Port P7 is a 6-bit general-purpose input / output port which can be configured as either input or output in one-bit unit. Input / output mode is specified by port 7 input / output control register 1 (P7CR1). Input/output circuit is specified by port 7 input / output control register 2 (P7CR2). During reset, P7CR1 is cleared to "0", and port P7 is configured as an input mode. The output latches are initialized to "0". P7CR1 is write-only register. P76 is also used as an External interrupt input or as a STOP mode release input.



P7 (0007_H)

7	6	5	4	3	2	1	0
P77	P76 STOP / INT5	P75	P74	P73	P72	–	–

(Initial value 0000 00**)

P7CR1 ($000D_H$)

7	6	5	4	3	2	1	0
						–	–

(Initial value 0000 00**)

P7CR1	I/O control for P7 (specified in one-bit unit)	0 : Input mode 1 : Output mode	write only

P7CR2 (0027_H)

7	6	5	4	3	2	1	0
						–	–

(Initial value 1111 11**)

P7CR2	I/O control for P7 (specified in one-bit unit)	0 : Open drain input / output 1 : Tri-state input / output	R/W

Note : The P7CR is a write-only register and cannot be used by any read-modify instructions.

Figure 2-5. Port P7 and P7CR

Note 1: Ports set to the input mode read the pin states. When input pin and output in exist in port P7 together, the contents of the output latch of ports set to the input mode may be rewritten by executing the bit manipulation instructions. Pins set to output mode read a value of the output latch.

Note 2: In case of using P62 to P65 as key wake-up inputs, use P76 (including INT5/STOP) only as an input, do not set it as output.

*Note 3: * ; Don't care*

Example : The lower 2-bit of Port P7 is set to an output port and the others are set to an input port.

```
LD    (P7CR1), 0FH      ; P7CR1←00001111
```

2.3 Time Base Timer (TBT)

The Time-base timer is used to generate the base time for key scan and dynamic display processing. For this purpose, it generates a time-base timer interrupt (INTTBT) at predetermined intervals.

This interrupt is generated beginning with the first rising edge of the source clock (the timing generator's divider output selected by TBTCK) after the time-base timer is enabled. Note that since the divider cannot be cleared by a program, the first interrupt only may occur earlier than the set interrupt period. (See Figure 2-6, (b).)

When selecting the interrupt frequency, make sure the time-base timer is disabled. (Do not change the selected interrupt frequency when disabling the active timer either.) However, you can select the interrupt frequency simultaneously when enabling the timer.

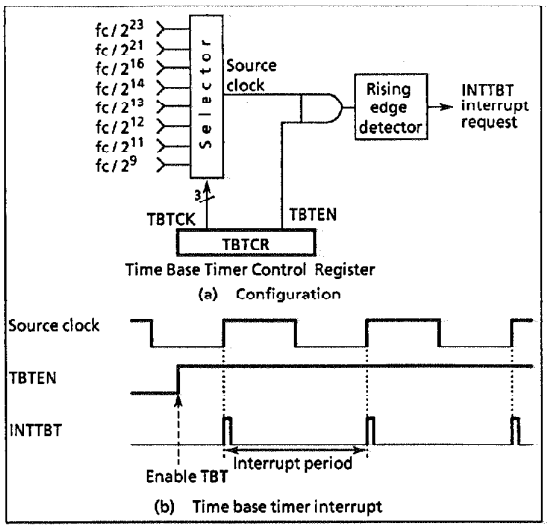


Figure 2-6. Time base timer

TBTCR (0036_H) (Initial value : 0**0 0***)

7	6	5	4	3	2	1	0
(DVOEN)	(DVCK)		0	TBTEN		TBTCK	

TBTEN	Time Base Timer enable/disable	0 : Disable 1 : Enable	R/W
TBTCK	Time base timer interrupt frequency select	000 : $fc/2^{23}$ [Hz] 001 : $fc/2^{21}$ 010 : $fc/2^{16}$ 011 : $fc/2^{14}$ 100 : $fc/2^{13}$ 101 : $fc/2^{12}$ 110 : $fc/2^{11}$ 111 : $fc/2^{9}$	

*Note 1 : fc ; clock [Hz], * ; don't care*
Note 2 : The fourth bit in TBTCR must be to "0".

Figure 2-7. Time base timer control register

Table 2-1. Time base timer interrupt frequency [Hz]

TBTCK	NORMAL, IDLE mode (at fc = 8 MHz)
000	0.95
001	3.81
010	122.07
011	488.28
100	976.56
101	1953.12
110	3906.25
111	15625

2.4 Divider Output ($\overline{DVO}$)

A 50% duty pulse can be output using the Divider output circuit, which is useful for piezo-electric buzzer drive. Divider output is from pin P13 ($\overline{DVO}$). The P13 output latch should be set to "1" and then the P13 should be configured as an output mode.

TBTCR (0036_H)

7	6	5	4	3	2	1	0
DVOEN	DVOCK		0	(TBTEN)		(TBTCK)	

(Initial value : 0**0 0***)

DVOEN	Divider output enable/disable	0 : Disable 1 : Enable	R/W
DVOCK	Divider output ($\overline{DVO}$ pin) frequency selection	00 : $fc/2^{13}$ [Hz] 01 : $fc/2^{12}$ 10 : $fc/2^{11}$ 11 : $fc/2^{10}$	

*Note : fc ; High-frequency clock [Hz], * ; don't care*

Figure 2-8. Divider output control register

Example : 1 kHz pulse output (at fc = 8 MHz)

```
SET     (P1).3                  ; P13 output latch ←1
LD      (P1CR), 00001000B       ; Configures P13 as an output mode
LD      (TBTCR), 10000000B      ; DVOEN←1, DVOCK←00
```

Table 2-2. Frequency of divider output [kHz]

DVOCK	At fc = 4 MHz	At fc = 8 MHz
00	0.512	0.976
01	1.024	1.953
10	2.048	3.906
11	4.096	7.812

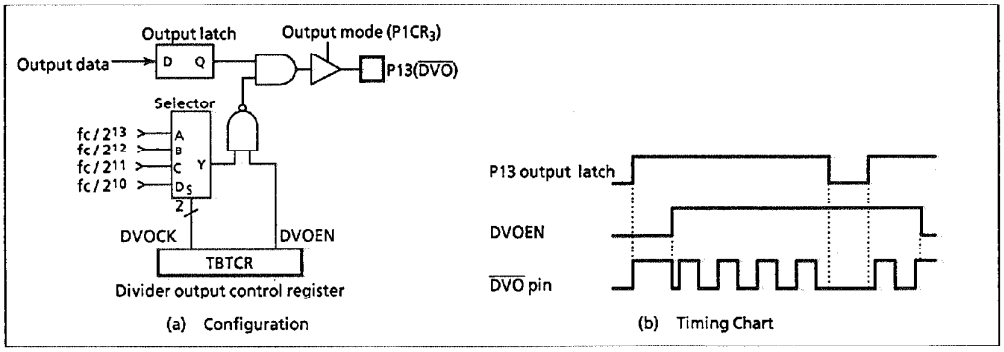


Figure 2-9. Divider output

2.5 16-bit Timer/Counter 1 (TC1)

2.5.1 Configuration

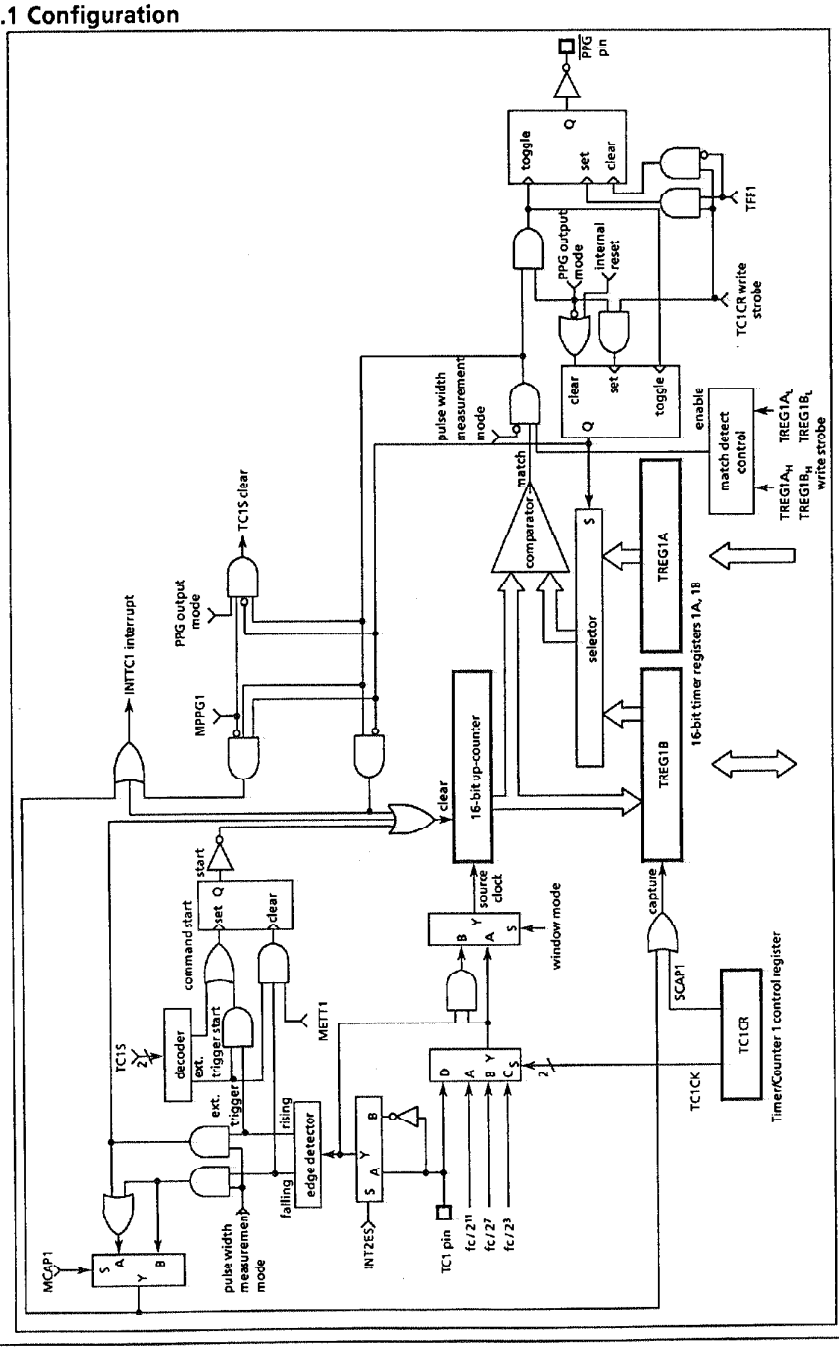


Figure 2-10. Timer/counter 1

2.5.2 Control

The Timer/Counter 1 is controlled by a Timer/Counter 1 control register (TC1CR) and two 16-bit timer registers (TREG1A and TREG1B).

Timer register

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TREG1A (0010_H, 0011_H)	$TREG1A_H$ (0011_H)								$TREG1A_L$ (0010_H)							
TREG1B (0012_H, 0013_H)	$TREG1B_H$ (0013_H)								$TREG1B_L$ (0012_H)							

Write only

Timer/Counter — Read / Write (Write available in only PPG output mode)

	7	6	5	4	3	2	1	0
TC1CR (0014_H)	TFF1	SCAP1 MCAP1 METT1 MPPG1	TC1S		TC1CK		TC1M	

(Initial value : 0000 0000)

TC1M	TC1 mode select	00 : Timer / External trigger timer / Event counter mode 01 : Window mode 10 : Pulse width measurement mode 11 : PPG (Programmable pulse generate) output mode	write only
TC1CK	TC1 source clock select	00 : Internal clock fc / 2^{11} [Hz] 01 : Internal clock fc / 2^7 10 : Internal clock fc / 2^3 11 : External clock (TC1 pin input)	
TC1S	TC1 start control	00 : Stop & counter clear 01 : Command start 10 : reserved 11 : External trigger start	
SCAP1	Software capture control	0 : – 1 : Software capture trigger	
MCAP1	Pulse width measurement control	0 : Double edge capture 1 : Single edge capture	
METT1	External trigger timer control	0 : Trigger start 1 : Trigger start & stop	
MPPG1	PPG output control	0 : Continuous pulse 1 : Single pulse	
TFF1	Timer F/F1 control	0 : Clear 1 : Set	

Note 1 : fc ; High-frequency clock [Hz]

Note 2 : Writing to the low-byte of the timer registers ($TREG1A_L$, $TREG1B_L$), the comparison is inhibited until the high-byte ($TREG1A_H$, $TREG1B_H$) is written. (Only the low-byte of the timer registers cannot be changed.) After writing to the high-byte, the comparison within 1 cycle(during instruction execution) is ignored.

Note 3 : Set the mode, source clock, edge (INT2ES), PPG output control and timer F/F1 control when TC1 stops (TC1S = 00).

Note 4 : Software capture can be used in only timer and event counter modes. SCAP1 is automatically cleared to "0" after software capture.

Note 5 : Values to be loaded to timer registers must satisfy the following condition.
TREG1A > TREG1B > 0 (PPG output mode) ; TREG1A > 0 (except for PPG output mode)

Note 6 : Always write "0" to TFF1 except the PPG output mode.

Note 7 : TREG1B can be written only in PPG output mode.

Note 8 : TC1CR is a write-only register, which cannot be accessed by any read-modify-write instructions such as bit operate, etc.

Note 9 : When fc/23 is selected as a source clock in the pulse width measurement mode, the lowest bit (bit0) of the counter value (TREGIB) is always read in as "0".
When others are selected as a source clock, the counter value is read according to the count.

Figure 2-11. Timer registers and control register TC1

2.5.3 Function

Timer/Counter 1 has six operating modes: Timer, External trigger timer, Event counter, Window, Pulse width measurement, and Programmable pulse generator output mode.

(1) **Timer** mode

In this mode, counting up is performed using the internal clock. The contents of the Timer register 1A (TREG1A) are compared with the contents of up-counter. If a match is fund, an INTTC1 interrupt is generated, and the counter is cleared to "0". Counting up resumes after the counter is cleared. The current contents of up-counter can be transferred to the Timer register 1B (TREG1B) by setting SCAP1(bit 6 in TC1CR) to "1"(software capture function). SCAP1 is automatically cleared to "0" after capturing.

Table 2-3. Timer/counter 1 source clock (internal clock) (at fc = 8MHz)

TC1CK	Resolution [μs]	Maximum time setting [s]
00	256	16.8
01	16	1.0
10	1	65.5 m

Example 1 : Sets the Timer mode with source clock fc/2^{11}[Hz] and generates an interrupt 1 s. later (at fc = 8 MHz).

```
LDW     (TREG1A), 1000H          ; Sets the timer register (1s ÷ 2^11 / fc = 1000H)
SET     (EIRL). EF4              ; Enables INTTC1 interrupt
EI
LD      (TC1CR), 00010000B       ; Starts TC1
```

Note : TC1CR is a write-only register, which cannot start by [SET(TC1CR).4] instruction.

Example 2 : Software capture

```
LD      (TC1CR) , 010100008      ; SCAP1←1
LD      WA, (TREG1B)             ; Reads captured value
```

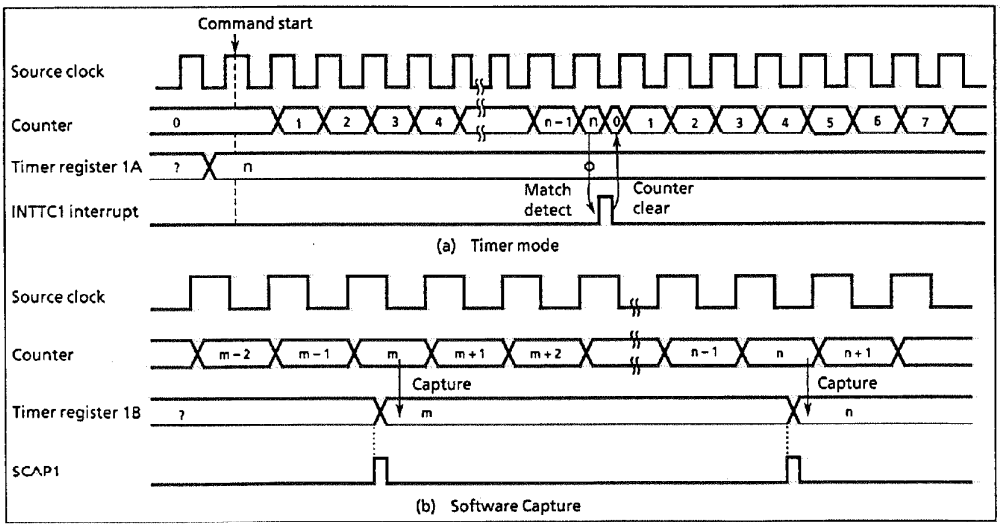


Figure 2-12. Timer mode timing chart

(2) **External trigger timer** mode

In this mode, counting up is started by an external trigger. This trigger is the edge of the TC1 pin input. Either the riging or the falling edge can be selected. Edge selection is the same as for INT2 pin. Source clock is used an internal clock selected. The contents of TREG1A is compared with the contents of up-counter. If a match is found, an INTTC1 interrupt is generated, and the counter is cleared to "0" and halted. The counter is restarted by the selected edge of the TC1 pin input.

When the edge input is opposite to the edge input way of the count start trigger at METTI (bit 6 in TC1CR) = 1, the counter is cleared, and count stops. In this mode, pulse input with a constant pulse width generates interrupt. When METT1 is "0", the opposite edge input is ignored. The edge of TC1 pin input before match detection is also ignored.

The TC1 pin input has the same noise rejection as the INT2 pin, therefore, pulses of 7/fc [s] or less are rejected as noise in NORMAL or IDLE mode. A pulse width of 24/fc [s] or more is required for edge detection.

Example 1 :Generates interrupt after 100 μs from TC1 pin input rising edge (at fc = 8 MHz).

```
LD      (EINTCR), 00000000B     ; INT2ES←0 (rising edge)
LDW     (TREG1A), 0064H         ; 100 μs ÷ 2^3 / fc = 64H
SET     (EIRL).EF4              ; Enables INTTC1 interrupt
EI
LD      (TC1CR), 00111000B      ; Starts TC1 external trigger, METT = 0
```

Example 2 : When "L" level pulses of 4ms or more is input to TC1 pin, generates interrupt. (at fc = 8 MHz)

```
LD      (EINTCR), 00000100B     ; INT2ES←1 (falling edge)
LDW     (TREG1A), 00FAH         ; 4 ms ÷ 2^7 / fc = FAH
SET     (EIRL).EF4              ; Enables INTTC1 interrupt
EI
LD      (TC1CR), 01110100B      ; Starts TC1 external trigger , METT = 1
```

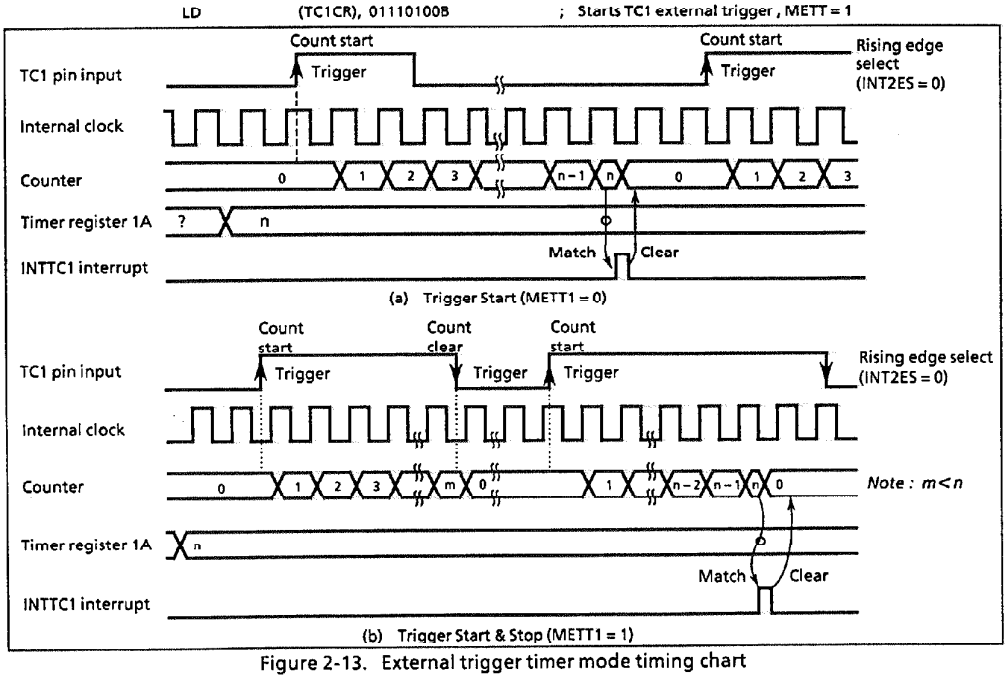


Figure 2-13. External trigger timer mode timing chart

(3) Event counter mode

In this mode, events are counted on the edge of the TC1 pin input. Either the rising and the falling edge can be selected with INT2 pin. The contents of TREG1A are compared with the contents of up-counter. If a match is found, an INTTC1 interrupt is generated, and the counter is cleared. The maximum applied frequency is fc/24 [Hz] in NORMAL or IDLE mode.

Setting SCAP1 to "1" transfers the current contents of up-counter to TREG1B (software capture).

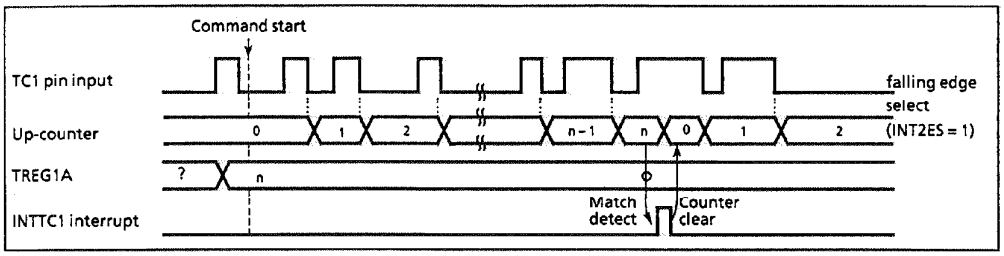


Figure 2-14. Event counter mode timing chart

(4) Window mode

Counting up is performed on the rising edge of the pulse that is the logical AND-ed product of the TC1 pin input (Window pulse) and an internal clock. The contents of TREG1A are compared with the contents of up-counter. If a match is found, an INTTC1 interrupt is generated, and the counter is cleared. Positive or negative logic for the TC1 pin input can be selected with INT2 pin.

It is necessary that the maximum applied frequency be such that the counter value can be analyzed by the program. That is, the frequency must be considerably slower than the selected internal clock.

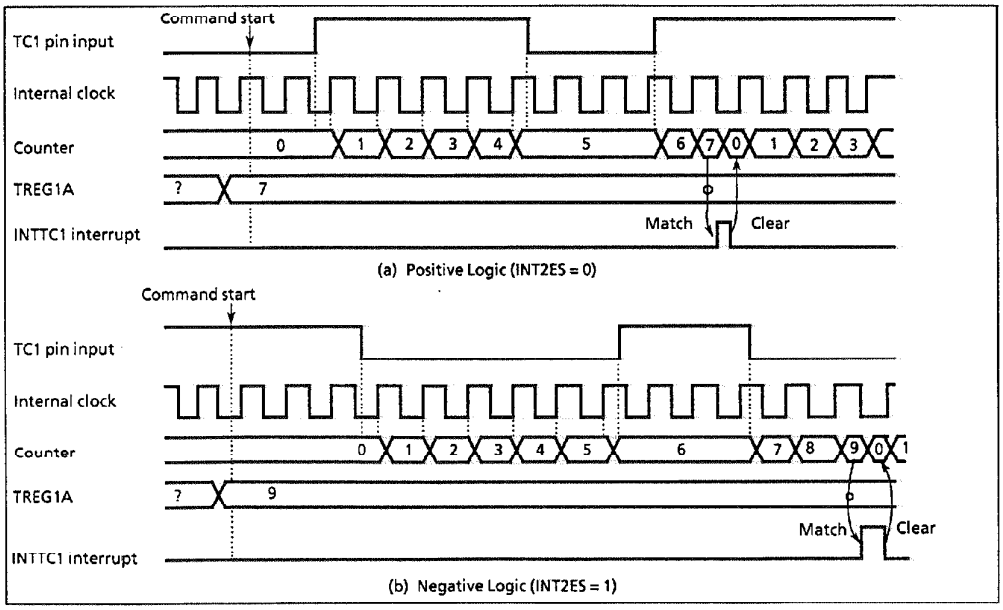


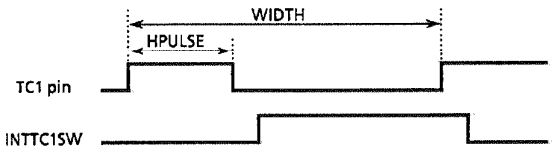
Figure 2-15. Window mode timing chart

(5) **Pulse width measurement** mode

Counting is started by a trigger of the rising (falling) edge of the TC1 pin input (set to external trigger start by TC1CR). The source clock is used an internal clock. On the next falling (rising) edge, the counter contents are transferred to TREG1B and an interrupt is generated. The counter is cleared when the single edge capture mode is set. When double edge capture is set, the counter continues and, at the next rising (falling) edge, the counter contents are again transferred to TREG1B. If a falling (rising) edge capture value is required, it is necessary to read out TREG1B contents until a rising (falling) edge is detected. Falling or rising edge is selected with INT2ES, and single edge or double edge is selected with MCAP1 (bit 6 in TC1CR).

Example : Duty measurement (Resolution fc/27[Hz])

```
                CLR     (INTTC1SW). 0          ;  INTTC1 service switch initial setting
                LD      (EINTCR), 00000000B    ;  Sets the rising edge at the INT2ES edge
                LD      (TC1CR), 00000110B     ;  Sets the TC1 mode and source clock
                SET     (EIRL). EF4            ;  Enables INTTC1
                EI
                LD      (TC1CR), 00110110B     ;  Starts TC1 with an external trigger in MCAP1 = 0
                :
PINTTC1 :       CPL     (INTTC1SW). 0          ;  Complements INTTC1 service switch
                JRS     F, SINTTC1
                LD      (HPULSE), (TREG1BL)    ;  Reads TREG1B
                LD      (HPULSE + 1), (TREG1BH)
                RETI
SINTTC1 :       LD      (WIDTH), (TREG1BL)     ;  Reads TREG1B (Period)
                LD      (WIDTH + 1), (TREG1BH)
                :                              ;  Duty calculation
                RETI
                :
VINTTC1 :       DW      PINTTC1
```



Note: When fc/23 is selected as a source clock in the pulse width measurement mode, the lowest bit (bit0) of the counter value (TREGIB) is always read in as "0".
When others are selected as a source clock, the counter value is read according to the count.

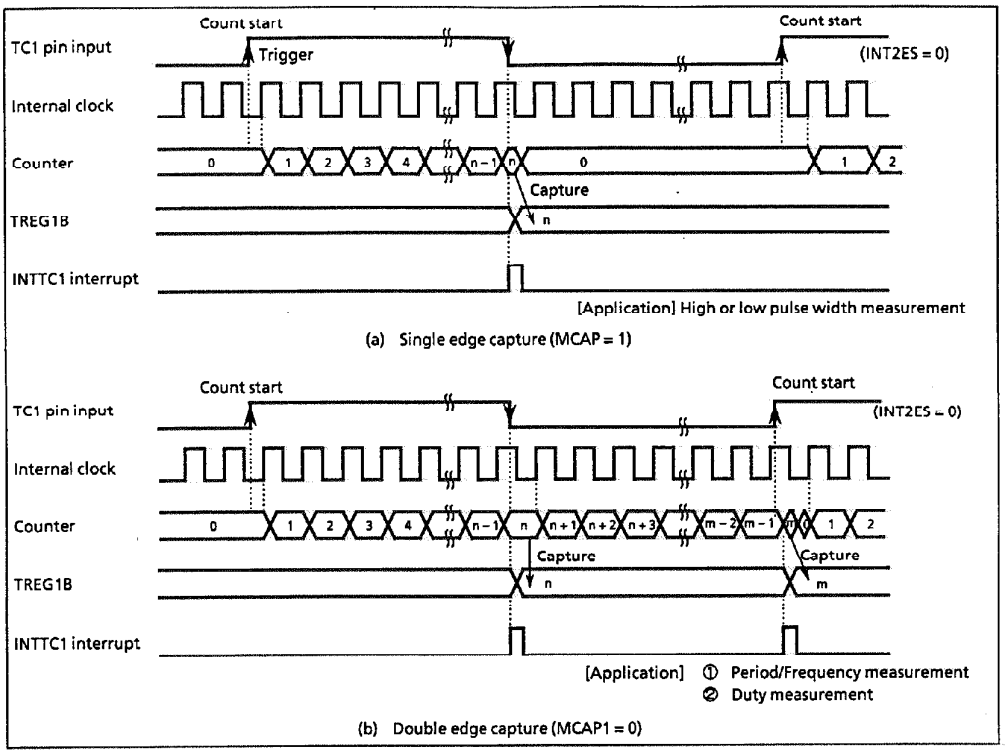


Figure 2-16. Pulse width measurement mode

(6) **Programmable pulse generate (PPG) output mode**

Counting is started by an edge of the TC1 pin input (either the rising or falling edge can be selected) or by a command. Edge select is the same as for INT2 pin. The source clock is used an internal clock. First, the contents of TREG1B are compared with the contents of the up-counter. If a match is found, Timer F/F1 output is toggled. INTTC1 interrupt is generated at continuous output (MPPG1 = 0) Next, Timer F/F1 is again toggled and the counter is cleared by matching with TREG1A. An INTTC1 interrupt is generated at this time. Timer F/F1 output is connected to the P14 ($\overline{PPG}$) pin. In the case of PPG output, set the P14 output latch to "1" and configure as an output mode. Timer F/F1 is cleared to "0" during reset. The Timer F/F1 value can also be set by TFF1 (bit 7 in TC1CR) and either a positive or negative logic pulse output is available. Also, writing to the TREG1B is not possible unless the Timer/Counter 1 is set to the PPG output mode.

Example : "H" level 800 μs, "L"" level 200 μs pulse output at fc = 8 MHz

```
SET   (P1).4              ;  P14 output latch←1
LD    (P1CR), 00010000B   ;  Sets P14 to an output mode
LD    (TC1CR), 10001011B  ;  Sets PPG output mode
LDW   (TREG1A), 03E8H     ;  Sets a period (1 ms ÷ 1 μs = 03E8H)
LDW   (TREG1B), 00C8H     ;  Sets "L" level pulse width (200 μs ÷ 1 μs = 00C8H)
LD    (TC1CR), 10010011B  ;  Start
```

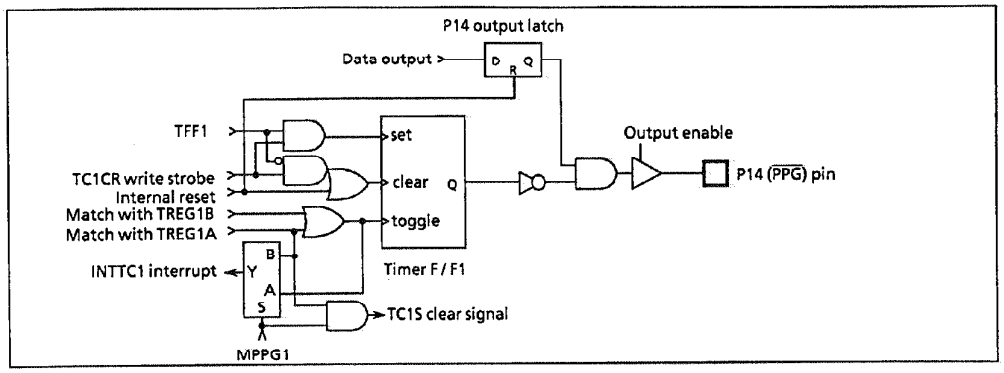


Figure 2-17. $\overline{\text{PPG}}$ output

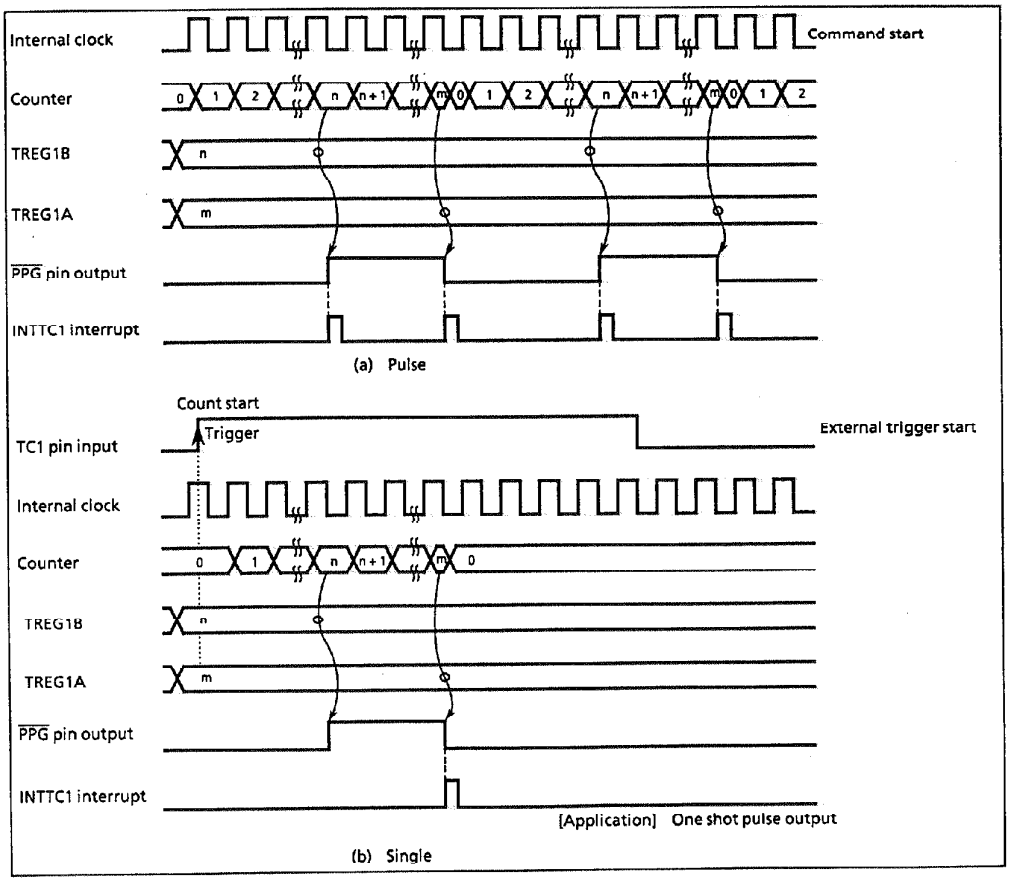


Figure 2-18. PPG output mode timing chart

2.6 16-bit Timer/Counter 2 (TC2)

2.6.1 Configuration

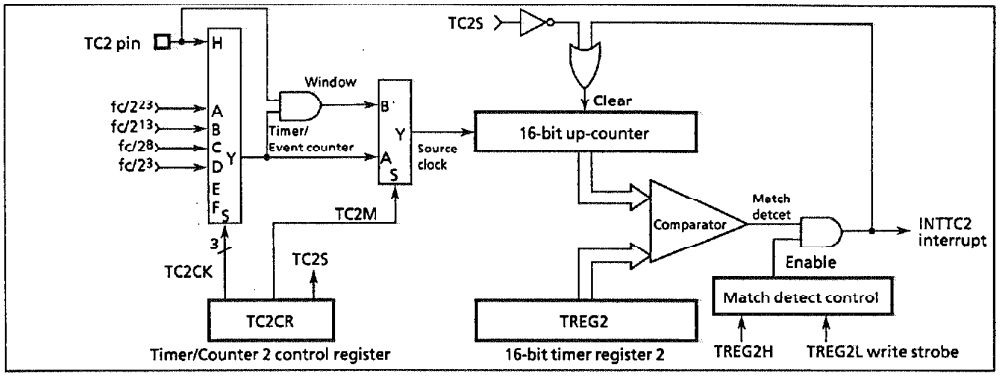
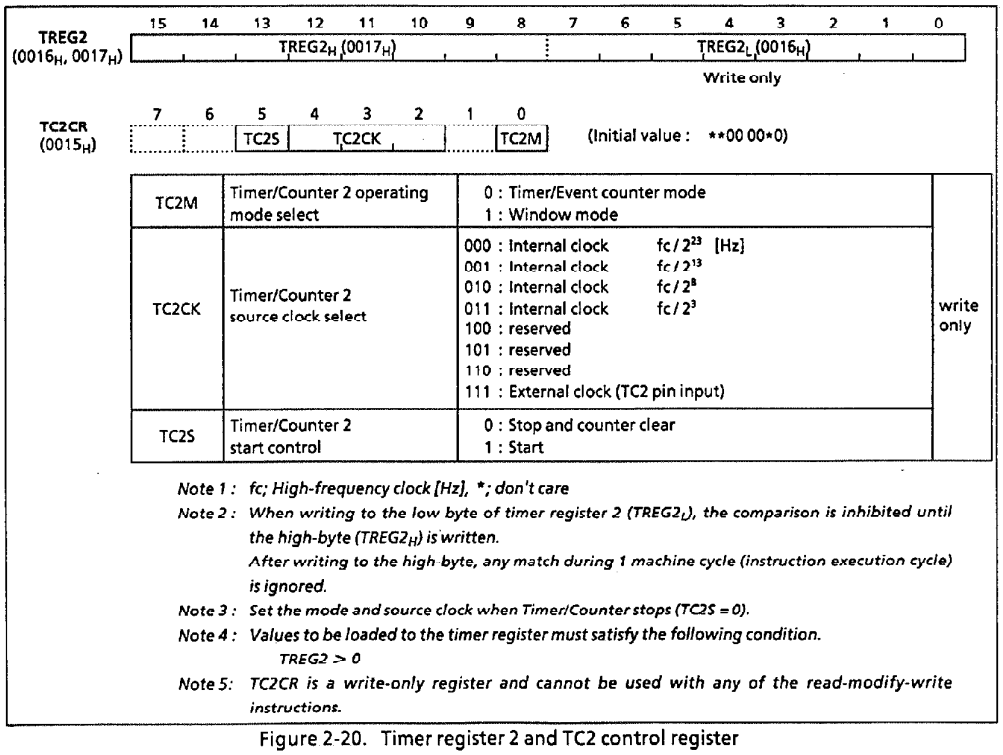


Figure 2-19. Timer/counter 2 (TC2)

2.6.2 Control

The Timer/Counter 2 is controlled by a Timer/Counter 2 control register (TC2CR) and a 16-bit timer register 2 (TREG2).



TC2M	Timer/Counter 2 operating mode select	0 : Timer/Event counter mode 1 : Window mode	write only
TC2CK	Timer/Counter 2 source clock select	000 : Internal clock fc / 2^{23} [Hz] 001 : Internal clock fc / 2^{13} 010 : Internal clock fc / 2^{8} 011 : Internal clock fc / 2^{3} 100 : reserved 101 : reserved 110 : reserved 111 : External clock (TC2 pin input)	
TC2S	Timer/Counter 2 start control	0 : Stop and counter clear 1 : Start	

*Note 1 : fc; High-frequency clock [Hz], *; don't care*

Note 2 : When writing to the low byte of timer register 2 ($TREG2_L$), the comparison is inhibited until the high-byte ($TREG2_H$) is written.
After writing to the high byte, any match during 1 machine cycle (instruction execution cycle) is ignored.

Note 3 : Set the mode and source clock when Timer/Counter stops (TC2S = 0).

Note 4 : Values to be loaded to the timer register must satisfy the following condition.
$TREG2 > 0$

Note 5: TC2CR is a write-only register and cannot be used with any of the read-modify-write instructions.

Figure 2-20. Timer register 2 and TC2 control register

2.6.3 Function

The Timer/Counter 2 has three operating modes: Timer, Event counter and Window modes.

(1) **Timer** mode

In this mode, the internal clock is used for counting up. The contents of TREG2 are compared with the contents of up-counter. If a match is found, an INTTC2 interrupt is generated, and the counter is cleared. Counting up is resumed after the counter is cleared.

Table 2-4. Source clock (Internal Clock) for timer/counter 2 (at 8MHz)

TC2CK	Resolution	Maximum time setting
000	1.05 s	19.1 h
001	1.02 ms	1.1 min
010	32 μs	2.1 s
011	1 μs	65.5 ms

Example : Sets the Timer mode with source clock $fc/2^3$ [Hz] and generates an interrupt every 25 ms (at fc = 8 MHz).

```
LDW     (TREG2), 61A8H          ; Sets the TREG2 (25 ms ÷ 2^3 / fc = 61A8H)
SET     (EIRH).EF14             ; Enables INTTC2 interrupt
EI
LD      (TC2CR), 00101100B      ; Starts TC2
```

(2) **Event counter** mode

In this mode, events are counted on the rising edge of the TC2 pin input. The contents of TREG2 are compared with the contents of the up-counter. If a match is found, an INTTC2 interrupt is generated, and the counter is cleared. The maximum frequency applied to the TC2 pin is $fc/2^4$ [Hz] in NORMAL or IDLE mode. But, a pulse width of 2 machine cycles or more is required for both "H" and "L" level.

Example : Sets the Event counter mode and generates an INTTC2 interrupt 640 counts later

```
LDW     (TREG2), 640            ; Sets the TREG2
SET     (EIRH).EF14             ; Enables INTTC2 interrupt
EI
LD      (TC2CR), 00111100B      ; Starts TC2
```

(3) **Window** mode

In this mode, counting up is performed by an internal clock during "H" level of TC2 external pin input (Window pulse). The contents of TREG2 are compared with the contents of up-counter. If a match is found, an INTTC2 interrupt is generated, and the up-counter is cleared to "0".

It is necessary that the maximum applied frequency must be considerably slower than the selected internal clock.

Example : Inputs "H" level pulse of 120 ms or more and generates interrupt. (at fc = 8 MHz).

```
LDW     (TREG2), 0078H          ; Sets TREG2 (120 ms ÷ 2^13/fc = 0078H)
SET     (EIRH).EF14             ; Enables INTTC2 interrupt
EI
LD      (TC2CR), 00100101B      ; Starts TC2
```

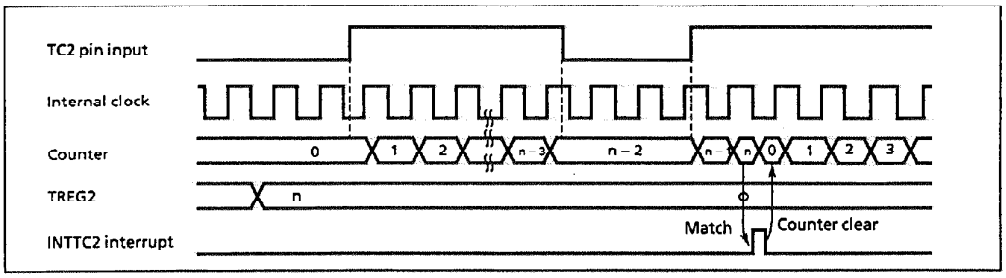


Figure 2-21. Window mode timing chart

2.9 Key Wake Up

The 87C405A can control STOP mode with four pins such as P62, P63, P64 and P65 other than P76 (INT5/STOP).

When the STOP mode is controlled in port inputs of P62, P63, P64 and P65, system register 1 (SYSCR1) must start the STOP mode (Level release mode).

STOP mode control register

STOPCR ($002F_H$)

7	6	5	4	3	2	1	0
–	–	STOP5	STOP4	STOP3	STOP2	–	–

(Initial value **0000**)

STOP2	P62 port releases from STOP mode	0 : Disable 1 : Enable	Read / Write
STOP3	P63 port releases from STOP mode	0 : Disable 1 : Enable	
STOP4	P64 port releases from STOP mode	0 : Disable 1 : Enable	
STOP5	P65 port releases from STOP mode	0 : Disable 1 : Enable	

Figure 2-22. Stop mode control register

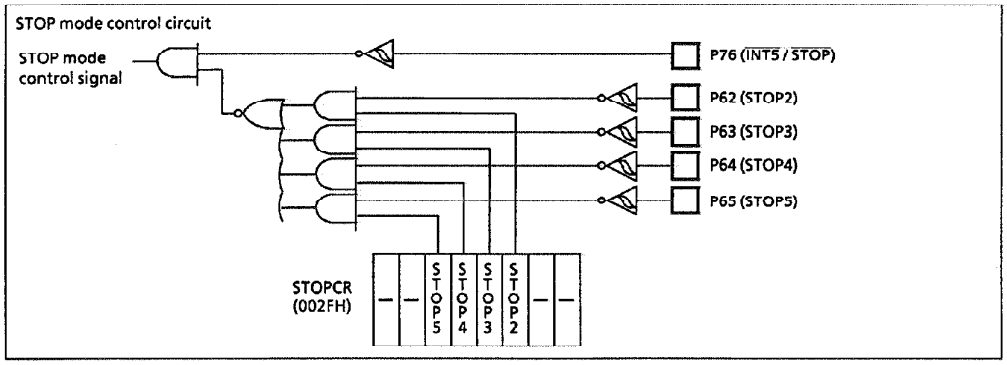


Figure 2-23. Stop mode control circuit

Each bit of P62 to P65 can be individually connected with internal pull-up resistor under the STOP mode control register (STOPCR).

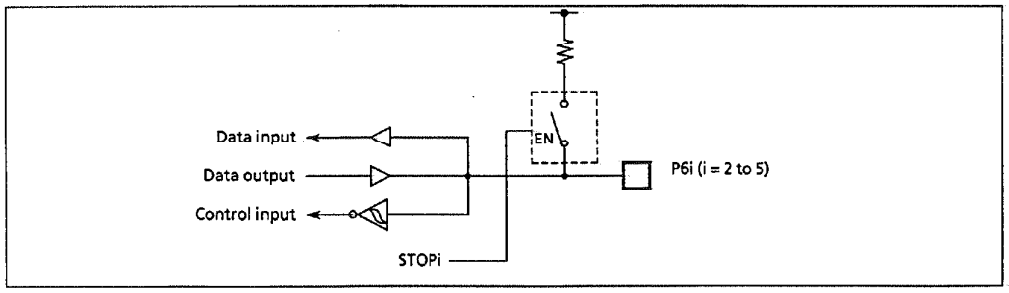


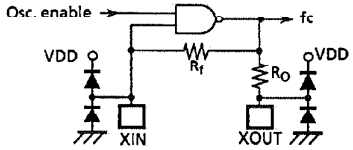
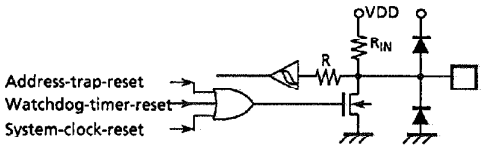
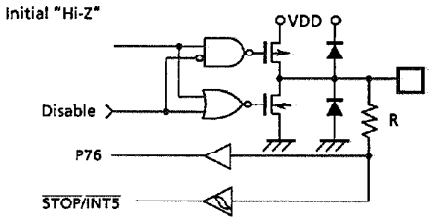
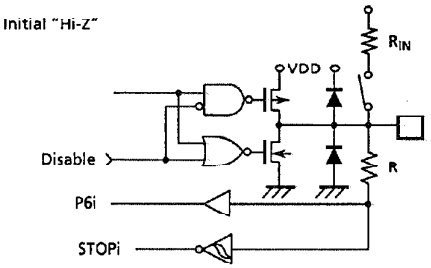
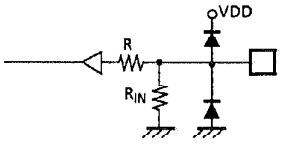
Figure 2-24. Port P6i (i = 2 to 5)

Note: In case of using P62 to P65 as key wake-up input, use P76 (including INT5/STOP) only as an input, do not set it as output.

Input / output Circuit

(1) Control pins

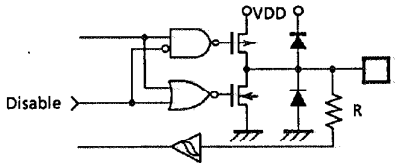
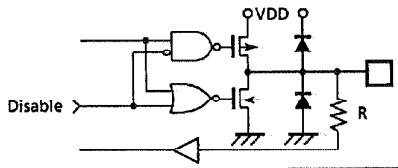
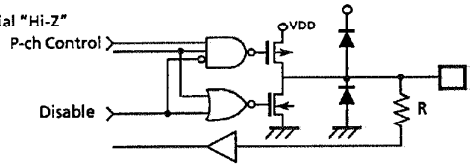
The input/output circuits of the 87C405A control pins are shown below.

Control Pin	I/O	Input / Output Circuitry and Code	Remarks
XIN XOUT	Input Output		High-frequency resonator connecting pin R_f = 1.2 MΩ (typ.) R_O = 1.5 kΩ (typ.)
RESET	I/O		Sink open drain output Hysteresis input Pull-up resistor R_{IN} = 220 kΩ (typ.) R = 1 kΩ (typ.)
$\overline{\text{STOP}}$/$\overline{\text{INT5}}$ (P76)	Input		Hysteresis input R = 1 kΩ (typ.)
STOPi (P6i)	Input		Hysteresis input i = 2~5 Pull-up resistor R_{IN} = 70 kΩ (typ.) R = 1 kΩ (typ.)
TEST	Input		Pull-down resister R_{IN} = 70 kΩ (typ.) R = 1 kΩ (typ.)

Note: The 87P808 does not have a pull-down resistor for TEST pin. Always fix to low level.

(2) Input / output ports

The input / output circuits of the 87C405A input / output ports are shown below.

Port	I/O	Input / Output Circuitry and Code	Remarks
P1	I/O	Initial "Hi-Z" 	Tri-state I/O Hysteresis input R = 1 kΩ (typ.)
P6	I/O	Initial "Hi-Z" 	Tri-state I/O R = 1 kΩ (typ.)
P7	I/O	Initial "Hi-Z" 	Tri-state I/O R = 1 kΩ (typ.)

Electrical Characteristics

87C405A

Absolute maximum ratings (V_{SS} = 0 V)

PARAMETER		SYMBOL	CONDITIONS	RATINGS	UNIT
Supply Voltage		V_{DD}		− 0.3 to 6.5	V
Input Voltage		V_{IN}		− 0.3 to V_{DD} + 0.3	V
Output Voltage		V_{OUT}		− 0.3 to V_{DD} + 0.3	V
Output Current (Per 1 pin)	IOL	I_{OUT1}	P1, P6	3.2	mA
		I_{OUT2}	P7 (Middle current port)	15	mA
	IOH	I_{OUT3}	P1, P6, P7	− 1.8	mA
Output Current (Total)	IOL	Σ I_{OUT1}	P1, P6	50	mA
		Σ I_{OUT2}	P7 (Middle current port)	60	mA
	IOH	Σ I_{OUT3}	P1, P6, P7	30	mA
Power Dissipation [Topr = 70 °C]		PD		180	mW
Soldering Temperature (time)		Tsld		260 (10 s)	°C
Storage Temperature		Tstg		− 55 to 125	°C
Operating Temperature		Topr		− 30 to 70	°C

Note: The absolute maximum ratings are rated values which must not be exceeded during operation, even for an instant. Any one of the ratings must not be exceeded. If any absolute maximum rating is exceeded, a device may break down or its performance may be degraded, causing it to catch fire or explode resulting in injury to the user. Thus, when designing products which include this device, ensure that no absolute maximum rating value will ever be exceeded.

Recommended operating conditions (V_{SS} = 0 V, Topr = − 30 to 70 °C)

PARAMETER	SYMBOL	PINS	CONDITIONS	Min.	Max.	UNIT
Supply Voltage	V_{DD}		fc = 8 MHz: NORMAL mode	4.5	5.5	V
			fc = 8 MHz: IDLE mode	4.5		
			fc = 4.2 MHz: NORMAL mode	2.7		
			fc = 4.2 MHz: IDLE mode	2.7		
			STOP mode	2.0		
Input High Voltage	V_{IH1}	Except hysteresis input	$V_{DD} \geqq 4.5$ V	$V_{DD} \times 0.70$	V_{DD}	V
	V_{IH2}	Hysteresis input	$V_{DD} \geqq 4.5$ V	$V_{DD} \times 0.75$		
	V_{IH3}		2.7 V ≦ V_D < 4.5 V	$V_{DD} \times 0.90$		
Input Low Voltage	V_{IL1}	Except hysteresis input	$V_{DD} \geqq 4.5$ V	0	$V_{DD} \times 0.30$	V
	V_{IL2}	Hysteresis input	$V_{DD} \geqq 4.5$ V		$V_{DD} \times 0.25$	
	V_{IL3}		2.7 V ≦ V_{DD} < 4.5 V		$V_{DD} \times 0.10$	
Clock Frequency	fc	XIN, XOUT	V_{DD} = 4.5 to 5.5 V	1.0	8.0	MHz
			V_{DD} = 2.7 to 5.5 V		4.2	

Note1: The recommended operating conditions for a device are operating conditions under which it can be guaranteed that the device will operate as specified. If the device is used under operating conditions other than the recommended operating conditions (supply voltage, operating temperature range, specified AC/DC values etc.), malfunction may occur. Thus, when designing products which include this device, ensure that the recommended operating conditions for the device are always adhered to.

Note2: Clock frequency fc : Supply voltage range is specified in NORMAL mode and IDLE mode.

Note3: Minimum of clock frequency : 400 kHz ≦ fcgck

D.C. characteristics

(V_{SS} = 0 V, Topr = – 30 to 70 °C)

PARAMETER	SYMBOL	PINS	CONDITIONS			Min.	Typ.	Max.	UNIT
Hysteresis Voltage	V_{HS}	Hysteresis inputs				–	0.9	–	V
Input Current	I_{IN1}	TEST	V_{DD} = 5.5 V V_{IN} = 5.5 V / 0 V			–2	–	2	μA
	I_{IN2}	Tri-state ports							
	I_{IN3}	RESET, STOPi							
Input Resistance	R_{IN1}	TEST				30	70	150	kΩ
	R_{IN2}	RESET				100	220	450	
	R_{IN3}	STOPi	i = 2 to 5			30	70	150	
Output Leakl Current	I_{LO}	Tri-state ports	V_{DD} = 5.5 V, V_{OUT} = 5.5 V / 0V			–2	–	2	μA
Output High Voltage	V_{OH2}	Tri-state ports	V_{DD} = 4.5 V, I_{OH} = – 0.7 mA			4.1	–	–	V
Output Low Voltage	V_{OL}	Except XOUT and P7	V_{DD} = 4.5V, I_{OL} = 1.6 mA			–	–	0.4	V
Output Low Current	I_{OL3}	P7	V_{DD} = 4.5 V, V_{OL} = 1.0 V			–	7	–	mA
Supply Current in NORMAL mode	I_{DD}		V_{DD} = 5.5 V fc = 8 MHz V_{IN} = 5.3 V / 0.2V	fcgck	fc	–	6.5	10	mA
					fc/2	–	4.0	6.4	
					fc/4	–	2.6	4.7	
					fc/8	–	1.9	3.9	
Supply Current in IDLE mode				fcgck	fc	–	3.3	5.0	
					fc/2	–	2.4	3.9	
					fc/4	–	1.9	3.5	
					fc/8	–	1.6	3.3	
Supply Current in NORMAL mode			V_{DD} = 3.0 V fc = 4.19 MHz V_{IN} = 2.8 V / 0.2V	fcgck	fc	–	1.5	2.5	
					fc/2	–	0.85	1.6	
					fc/4	–	0.6	1.2	
					fc/8	–	0.4	1.0	
Supply Current in IDLE mode				fcgck	fc	–	0.8	1.4	
					fc/2	–	0.55	1.1	
					fc/4	–	0.45	0.9	
					fc/8	–	0.35	0.85	
Supply Current in STOP mode			V_{DD} = 5.5 V V_{IN} = 5.3 V / 0.2 V			–	0.5	10	μA

Note 1: Typical values show those at Topr = 25 °C, VDD = 5 V.

Note 2: Input Current IIN1, IIN3 : The current through resistor is not included, when the input resistor (pull-up or pull-down) is contained.

A.C. characteristics (I) (V_{SS} = 0 V, V_{DD} = 4.5 to 5.5 V, Topr = – 30 to 70 °C)

PARAMETER	SYMBOL	CONDITIONS	Min.	Typ.	Max.	UNIT
Machine Cycle Time	tcy	In NORMAL mode	0.5	–	4	μs
		In IDLE mode				
High Level Clock Pulse Width	t_{WCH}	For external clock operation	50	–	–	ns
Low Level Clock Pulse Width	t_{WCL}	fc = 8 MHz				

A.C. characteristics (II) (V_{SS} = 0 V, V_{DD} = 2.7 to 5.5 V, Topr = – 30 to 70 °C)

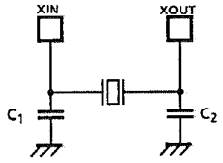
PARAMETER	SYMBOL	CONDITIONS	Min.	Typ.	Max.	UNIT
Machine Cycle Time	tcy	In NORMAL mode	0.95	–	4	μs
		In IDLE mode				
High Level Clock Pulse Width	t_{WCH}	For external clock operation	110	–	–	ns
Low Level Clock Pulse Width	t_{WCL}	fc = 4.2 MHz				

Recommended oscillating conditions (I) (V_{SS} = 0 V, V_{DD} = 4.5 to 5.5 V, Topr = – 30 to 70 °C)

PARAMETER	Oscillator	Oscillation Frequency	Recommended Oscillator		Recommended Conditions C_1	Recommended Conditions C_2
High-frequency Oscillation	Ceramic Resonator	8 MHz (VDD = 4.5 to 5.5 V)	KYOCERA	KBR8.0M	30 pF	30 pF
			MURATA	CSAC8.00MT	30 pF	30 pF
			MURATA	CSA8.00MTZ CST8.00MTW CSTS8.00MT	–	–
		4.19 MHz (VDD = 2.7 to 5.5 V)	MURATA	CSA4.19MG	30 pF	30 pF
			MURATA	CST4.19MGW	–	–
		4 MHz (VDD = 2.7 to 5.5 V)	KYOCERA	KBR4.0MS	30 pF	30 pF
	Crystal Oscillator	8 MHz (VDD = 4.5 to 5.5 V)	TOYOCOM	210B 8.0000	20 pF	20 pF
		4 MHz (VDD = 2.7 to 5.5 V)	TOYOCOM	204B 4.000		

Recommended oscillating conditions (II) (V_{SS} = 0 V, V_{DD} = 2.7 to 5.5 V, Topr = – 30 to 70 °C)

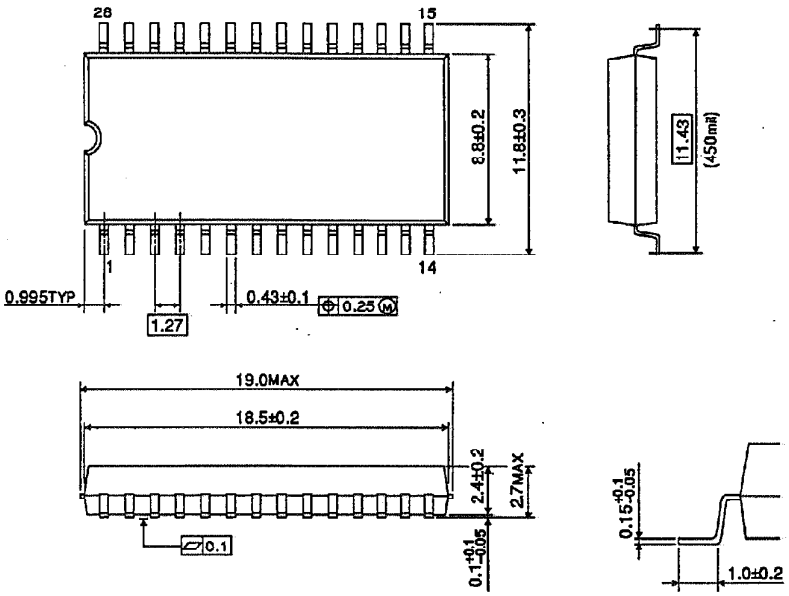
PARAMETER	Oscillator	Oscillation Frequency	Recommended Oscillator		Recommended Conditions C_1	Recommended Conditions C_2
High-frequency Oscillation	Ceramic Resonator	4.19 MHz (VDD = 2.7 to 5.5 V)	MURATA	CSA4.19MG	30 pF	30 pF
			MURATA	CST4.19MGW	–	–
		4 MHz (VDD = 2.7 to 5.5 V)	MURATA	CSA4.00MG	30 pF	30 pF
				CSA4.00MGC	–	–
			MURATA	CST4.00MGW CSTC4.00MG	–	–
			MURATA	CSTCS4.00MG	–	–



(1) High-frequency Oscillation

Note: When used in high electric field such as a picture tube, the package is recommended to be electrically shielded to maintain a regular operation.

SOP28-P-450-1.27 Unit : mm



SDIP28-P-400-1.78

Unit : mm