

**CMOS 8–Bit Microcontrollers****TMP90C400N/TMP90C401N****TMP90C400F/TMP90C401F****1. Outline and Characteristics**

The TMP90C400 is a high-speed advanced 8-bit microcontroller applicable to a variety of equipment.

With its 8-bit CPU, ROM, RAM, timer/event counter and general-purpose serial interface integrated into a single CMOS chip, the TMP90C400 allows the expansion of external memories for programs and data (up to 60K bytes). The TMP90C401 is the same as the TMP90C400 but without ROM.

The TMP90C400N/401N is in a shrink Dual Inline Package (SDIP64-P-750).

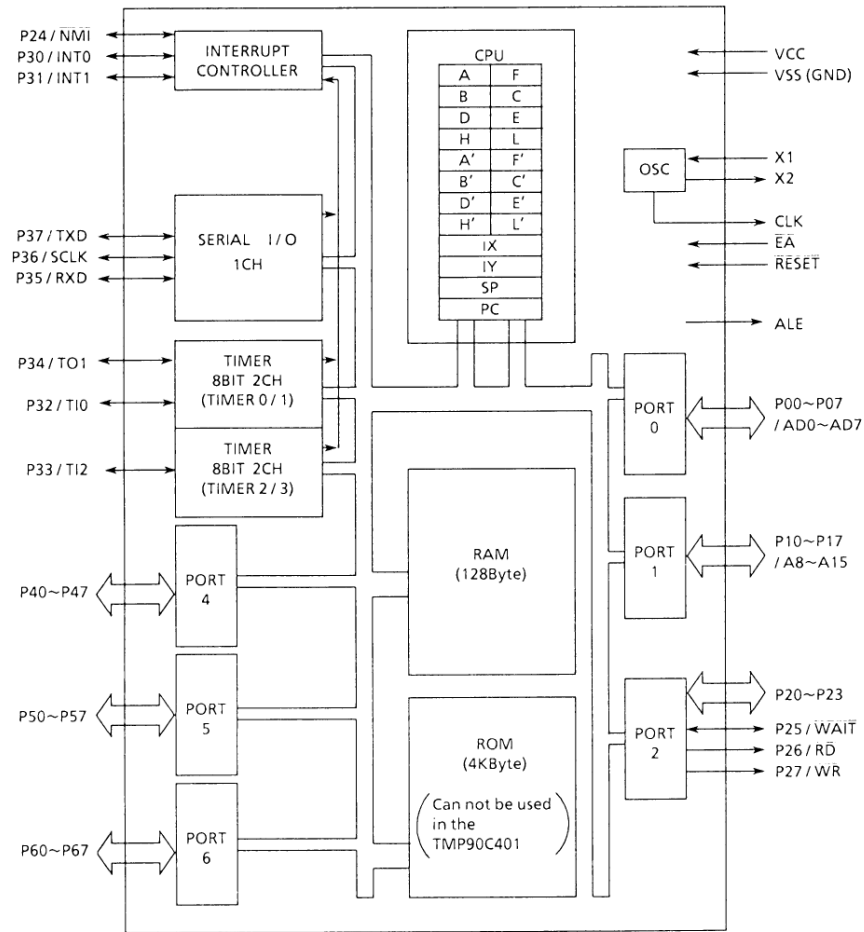
The TMP90C400F/401F is in a Quad Flat package (QFP64-P-1420A)

The characteristics of the TMP90C400 include:

- |   |  |
|---|--|
| <ul style="list-style-type: none"> <li>(1) Powerful instructions: 163 basic instructions, including Multiplication, division, 16-bit arithmetic operations, bit manipulation instructions</li> <li>(2) Minimum instruction executing time: 320ns (at 12.5MHz oscillation frequency)</li> <li>(3) Internal ROM: 4K bytes (The TMP90C401 does not have a built-in ROM)</li> </ul> | <ul style="list-style-type: none"> <li>(4) Internal RAM: 128 bytes</li> <li>(5) Memory expansion</li> <li style="padding-left: 20px;">External memory: 60K bytes</li> <li>(6) General-purpose serial interface (1 channel)<br/>Asynchronous mode, I/O interface mode</li> <li>(7) 8-bit timers (4 channel): (2 external clock inputs)</li> <li>(8) Port with zero-cross detection circuit (4 inputs)</li> <li>(9) Input/Output ports (56 pins) <ul style="list-style-type: none"> <li>- Ports with programmable pull-up resistor (22 pins)</li> <li>- Allows I/O selection on bit basis</li> <li>- Multiplexer ports of address data bus</li> </ul> </li> <li>(10) Interrupt function: 7 internal interrupts and 3 external interrupts</li> <li>(11) Micro Direct Memory Access (DMA) function (8 channels)</li> <li>(12) Standby function (4 HALT modes)</li> </ul> |
|---|--|

The information contained here is subject to change without notice.

The information contained herein is presented only as guide for the applications of our products. No responsibility is assumed by TOSHIBA for any infringements of patents or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of TOSHIBA or others. These TOSHIBA products are intended for usage in general electronic equipments (office equipment, communication equipment, measuring equipment, domestic electrification, etc.) Please make sure that you consult with us before you use these TOSHIBA products in equipments which require high quality and/or reliability, and in equipments which could have major impact to the welfare of human life (atomic energy control, spaceship, traffic signal, combustion control, all types of safety devices, etc.). TOSHIBA cannot accept liability to any damage which may occur in case these TOSHIBA products were used in the mentioned equipments without prior consultation with TOSHIBA.



**Figure 1. TMP90C400 Block Diagram**

## 2. Pin Assignment and Functions

This section describes the assignment of input/output pins, their names and functions.

### 2.1 Pin Assignment

Figure 2.1 (1) shows pin assignment of the TMP90C400N/401N.

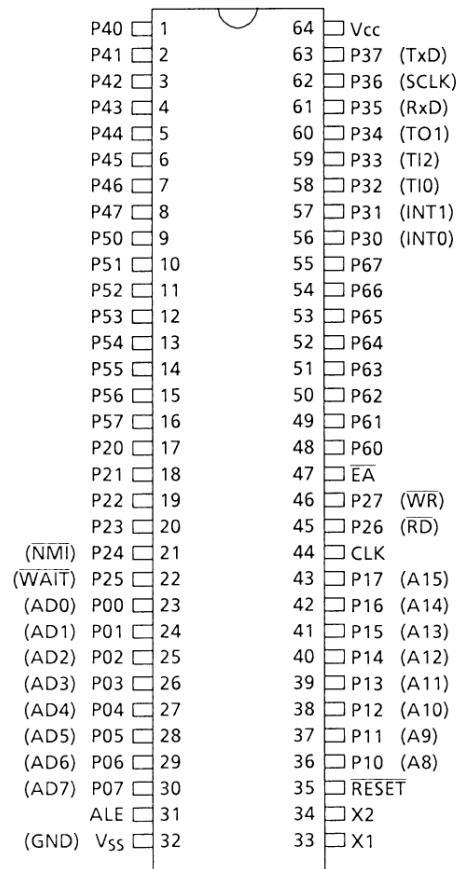
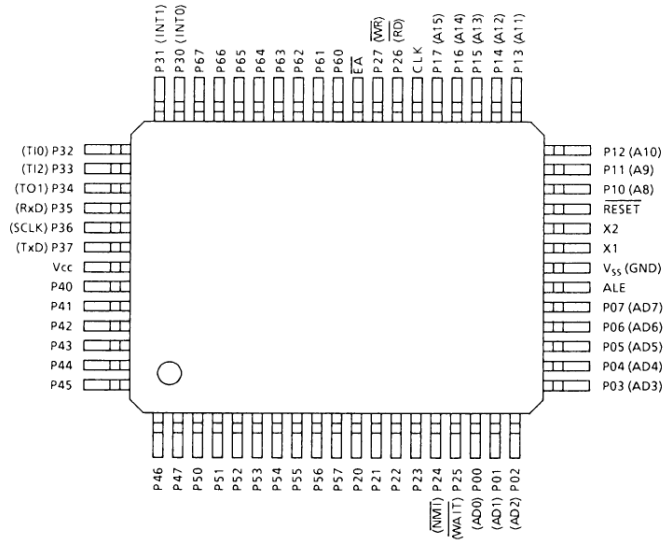


Figure 2.1 (1). Pin Assignment (Shrink Dual Inline Package)

**TMP90C400/401**

Figure 2.1 (2) shows Pin Assignment of the TMP90C400F/401F.


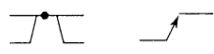



**Figure 2.1 (2). Pin Assignment (Flat Package)**

## 2.2 Pin Names and Functions

The names of input/output pins and their functions are summarized in Table 2.2.

**Table 2.2 Pin Names and Functions (1/2)**

Pin Name	No. of pins	I/O 3 states	Function
P00 ~ P07 /AD0 ~ AD7	8	I/O	Port 0: 8-bit I/O port that allows selection of input/output on byte basis
		3 states	Address/Data bus: Functions as 8-bit bidirectional address/data bus for external memory (For 401, fixed to address/data bus)
P10 ~ P17 /A8 ~ A15	8	I/O	Port 1: 8-bit I/O port that allows selection on byte basis
		Output	Address bus: Functions as address bus (upper 8 bits) by EXT1 set for external memory (For 401, fixed to address bus)
P20 ~ P23	4	I/O	Port 20 ~ 23: 4-bit I/O port with a pull-up resistor that can be programmed, and allows selection of input/output on bit basis
P24 /NMI	1	I/O	Port 24: 1-bit I/O port with a pull-up resistor that can be programmed, and allows selection of input/output on bit basis
		Input	Non-maskable interrupt request pin: Falling edge interrupt register pin 
P25 /WAIT	1	I/O	Port 25: 1-bit I/O port with a pull-up resistor that can be programmed, and allows selection of input/output on bit basis
		Input	Wait: Input pin for connecting slow speed memory of peripheral LSI
P26 /RD	1	Output	Port 26: 1-bit output port
		Output	Read: Generates strobe signal for reading external memory (For 401, fixed to $\overline{RD}$ )
P27 /WR	1	Output	Port 27: 1-bit output port
		Output	Write: Generates strobe signal for writing into external memory (for 401, fixed to $\overline{WR}$ )
P30 /INT0	1	I/O	Port 30: 1-bit I/O port with a pull-up resistor that can be programmed, and allows selection of input/output on bit basis
		Input	Interrupt request pin 0: Interrupt request pin (Level/rising edge is programmable) 
P31 /INT1	1	Input	Port 31: 1-bit I/O port with a pull-up resistor that can be programmed, and allows selection of input/output on bit basis
			Interrupt request pin 1: Rising edge interrupt request pin 
P32 /TIO	1	I/O	Port 32: 1-bit I/O port with a pull-up resistor that can be programmed, and allows selection of input/output on bit basis
		Input	Timer input 0: Counter input pin for Timer 0

**Table 2.2 Pin Names and Functions (2/2)**

Pin Name	No. of pins	I/O 3 states	Function
P33 /TI2	1	I/O	Port 33: 1-bit I/O port with a pull-up resistor that can be programmed, and allows selection of input/output on bit basis
		Input	Timer input 2: Counter input pin for Timer 2
P34 /TO1	1	I/O	Port 34: 1-bit I/O port with a pull-up resistor that can be programmed, and allows selection of input/output on bit basis
		Input	Timer input: Output of Timer 0 or 1
P35 /RxD	1	I/O	Port 35: 1-bit I/O port with a pull-up resistor that can be programmed, and allows selection of input/output on bit basis
P36 /SCLK	1	I/O	Port 36: 1-bit I/O port with a pull-up resistor that can be programmed, and allows selection of input/output on bit basis
		Output	Serial clock output
P37 TxD	1	I/O	Port 37: 1-bit I/O port with a pull-up resistor that can be programmed, and allows selection of input/output on bit basis
		Output	Transmitter serial data
P40 ~ P47	8	I/O	Port 4: 8-bit I/O port that allows I/O selection on bit basis
P50 ~ P57	8	I/O	Port 5: 1-bit I/O port with a pull-up resistor that can be programmed, and allows selection of input/output on bit basis
P60 ~ P67	8	I/O	Port 6: 8-bit I/O port that allows I/O selection on bit basis
ALE	1	Output	Address latch enable signal: The negative edge ALE supplies an address latch timing on AD0 ~ A07 for external memory
$\overline{EA}$	1	Input	External access: Connects with $V_{CC}$ pin in the TMP90C400 using internal ROM, and with GND pin in the TMP90C401 with no internal ROM
CLK	1	Output	Clock output: Generates clock pulse at 1/4 frequency of clock oscillation. It is pulled up internally during resetting.
$\overline{RESET}$	1	Input	Reset: Initializes the TMP90C400/401 (Built-in pull-up resistor)
X1/X2	2	Input/Output	Pin for quartz crystal or ceramic resonator (1 ~ 12.5MHz)
$V_{CC}$	1	–	Power supply (+5V)
$V_{SS}$	1	–	Ground (0V)

### 3. Operation

This chapter describes the functions and the basic operations of the TMP90C400/401 in every block.

The following is a description of TMP90C400 which can also be applied to TMP90C401, if not specifically defined otherwise.

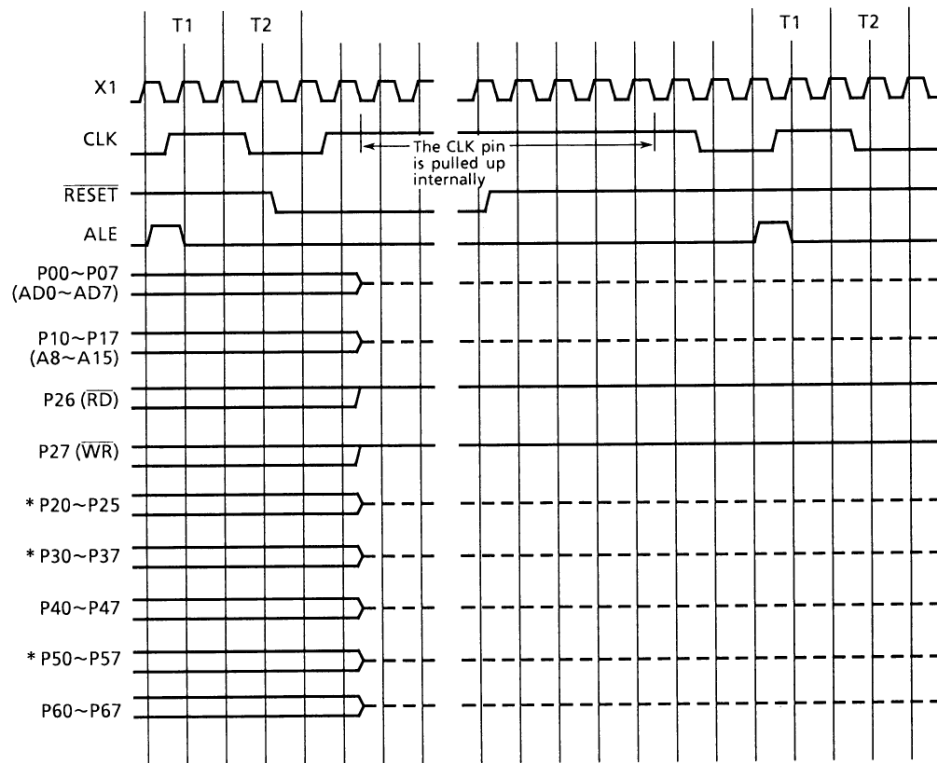
#### 3.1 CPU

The TMP90C400 includes a high performance 8-bit CPU. For the function of the CPU, see the book TLCS Series CPU Core Architecture concerning CPU operation. This chapter explains exclusively the functions of the CPU of TMP90C400 which are not described in that book.

#### 3.1.1 Reset

The basic timing of the reset operation is indicated in Figure 3.1 (1). In order to reset the TMP90C400, the  $\overline{\text{RESET}}$  input must be maintained at the "0" level for at least ten system clock cycles (10 stated: 2 $\mu$ sec at 10MHz) within an operating voltage band and with a stable oscillation. When a reset request is accepted, all I/O ports function as input ports (high impedance state). The P26 ( $\overline{\text{RD}}$ ), P27 ( $\overline{\text{WR}}$ ) and CLK pins that always function as output pins turn to the "1" level. The dedicated input ports remain unchanged. The registers of the CPU also remain unchanged. Note, however, that the program counter PC, the interrupt enable flag IFF are cleared to "0". Register A shows an undefined status.

When the reset is cleared, the CPU starts executing instructions from the address 0000H.

**Figure 3.1 (1a). TMP90C400 Reset Timing**

\* P20 ~ P25, P30 ~ P37 and P50 ~ P57, which have programmable pull-up resistors, remain "High" while resetting, unless input "Low".



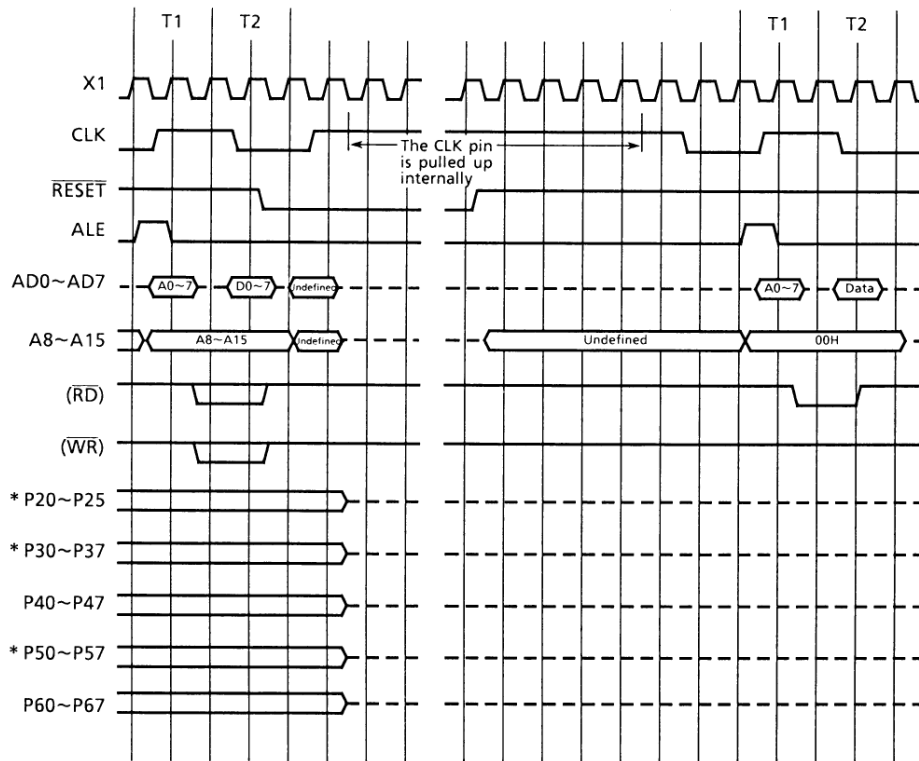


Figure 3.1 (1b). TMP90C401 Reset Timing

**3.1.2 EXF (Exchange Flag)**

For TMP90C400, “EXF”, which is inverted when the command “EXX” is executed to transfer data between the main register

and the auxiliary register, is allocated to the first bit of memory address FF8FH.

	7	6	5	4	3	2	1	0
STBMOD (FF8FH)								
bit Symbol	-	-	-	-	HALTM1	HALTM0	EXF	DRIVE
Read/Write	-	-	-	-	R/W		R	R/W
Resetting Value	-	-	-	-	0	0	Undefined	0
Function	-	-			Standby mode 00: RUN mode 01: STOP mode 10: IDLE1 mode 11: IDLE2 mode		Invert each time EXX instruction is executed	1: to drive pins in STOP mode

**TMP90C400/401**

**3.1.3 Wait Control**

to the 5th and 6th bits of memory address FF86H.

For TMP90C400, a wait control register (WAITC) is allocated

		7	6	5	4	3	2	1	0	
P2FR (FF86H)	bit Symbol	-	WAITC1	WAITC0	NMIC	-	-	-	EXT	
	Read/Write	-	R / W		R / W	-	-	-	W	
	Resetting Value	-	0	0	0	-	-	-	0	
Prohibit Read Modify Write	Function	-	Wait control 00: 2state wait 01: normal wait 10: non wait 11: reserved		NMI control 0: general-purpose port 1: input NMI		-	-	-	A8-15 control 0: general-purpose port 1: Address Bus

### 3.2 Memory Map

The TMP90C400 supports a program memory and a data memory of up to 60K bytes.

The program and data memory may be assigned to the address space from 0000H to FFFFH.

#### (1) Internal ROM

The TMP90C400 internally contains an 4K-byte ROM. The address space from 0000H ~ 0FFFH is provided to the ROM. The CPU starts executing a program from 0000H by resetting.

The addresses 0010H ~ 005FH in this internal ROM area are used for the entry area for the interrupt processing.

The TMP90C401 does not have a built-in ROM; therefore, the address space 0000H ~ 0FFFH is used as external memory space.

#### (2) Internal RAM

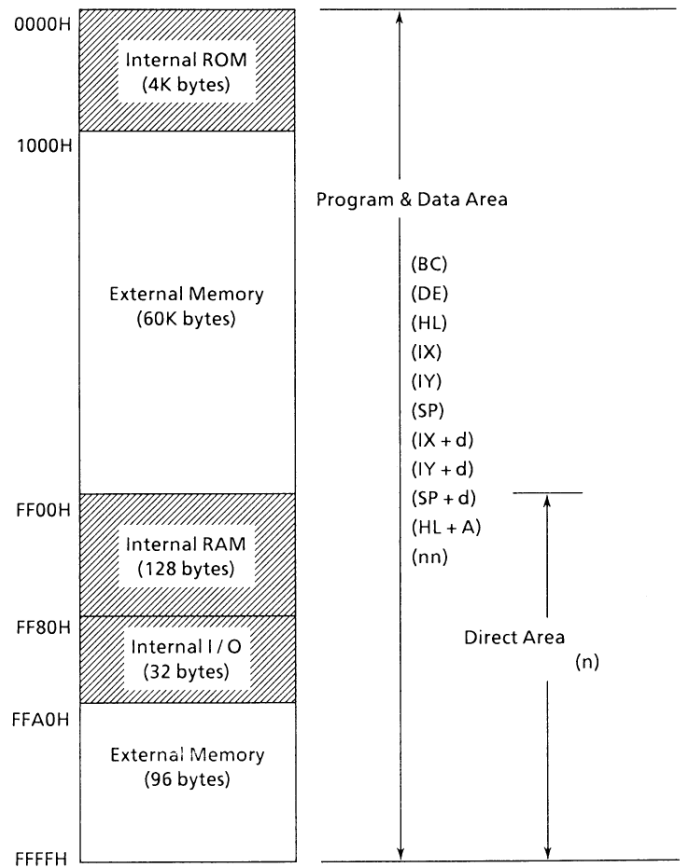
The TMP90C400 also contains a 128-byte RAM, which is allocated to the address space from FF00H ~ FF7FH. The CPU allows the access to a certain RAM area (FF00H ~ FF7FH, 128 bytes) by a short operation code (opcode) in a "direct addressing mode".

The addresses from FF20H to FF5FH in this RAM area can be used as parameter area for micro DMA processing (and for any other purposes when the micro DMA function is not used).

#### (3) Internal I/O

The TMP90C400 provides a 32-byte address space as an internal I/O area, whose addresses range from FF80H to FF9FH. This I/O area can be accessed by the CPU using a short opcode in the "direct addressing mode".

Figure 3.2 is a memory map indicating the areas accessible by the CPU in the respective addressing mode.



**Figure 3.2 (a). Memory Map of TMP90C400**

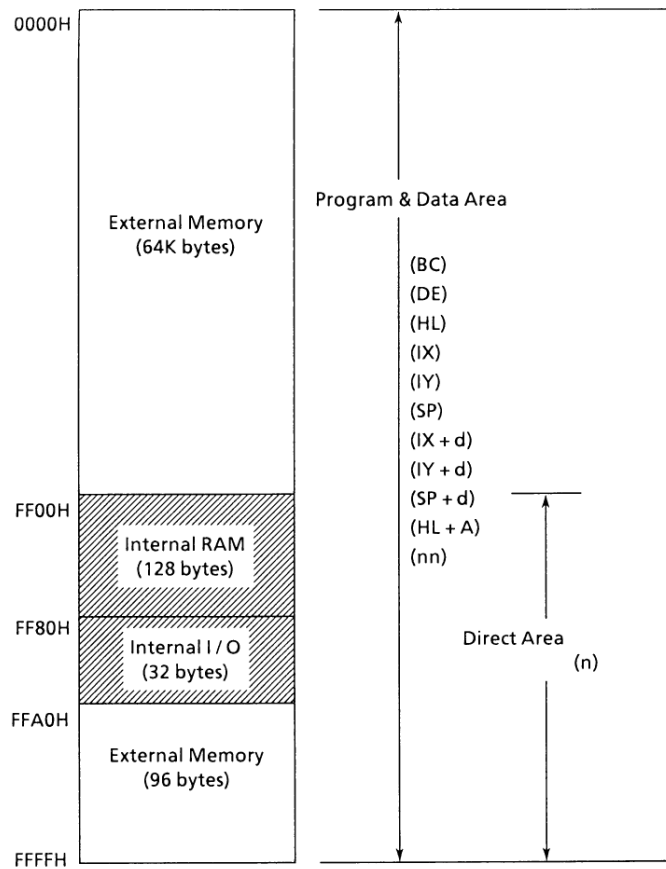


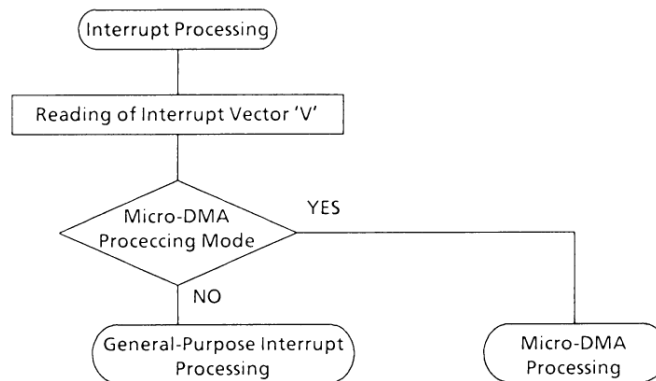
Figure 3.2 (b). Memory Map of TMP90C401

### 3.3 Interrupt Functions

The TMP90C400 supports a general purpose interrupt processing mode for internal and external interrupt requests and a micro DMA processing mode that enables automatic data transfer by the CPU. After the reset state is released, all interrupt requests are processed in the general purpose interrupt

processing mode. However, they can be processed in the micro DMA processing mode by using a DMA enable register to be described later.

Figure 3.3 (1) is a flowchart of the interrupt response sequence.



**Figure 3.3 (1). Interrupt Response Flowchart**

When an interrupt is requested, the request is transmitted to the CPU via an internal interrupt controller. The CPU starts the interrupt processing if it is a non-maskable or maskable interrupt requested in the EI state (interrupt enable flag (IFF = "1"). However, a maskable interrupt requested in the DI state (IFF = "0") is ignored. An interrupt request is sampled by the CPU at the falling edge of the CLK signal in the last bus cycle of each instruction.

By receiving an interrupt, the CPU reads out the interrupt vector from the internal interrupt controller to find out the interrupt source.

Then, the CPU checks if the interrupt requests the general purpose interrupt processing or the micro DMA processing, and proceeds to each processing.

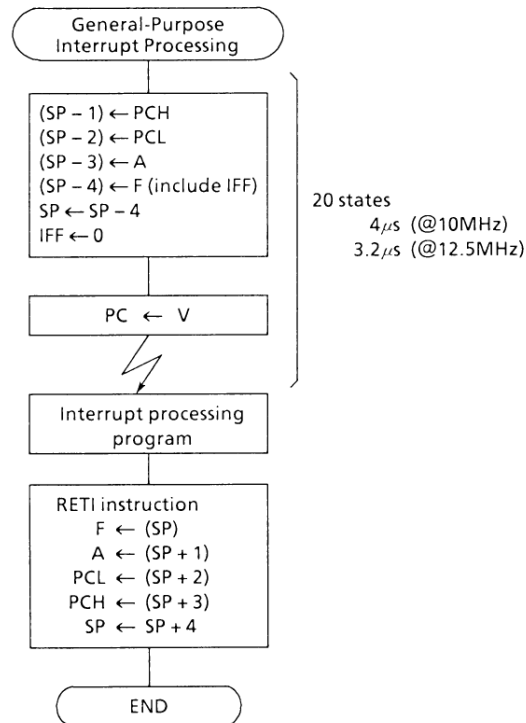
As the reading of an interrupt vectors is performed in the internal operating cycles, the bus cycle results in dummy cycles.

#### 3.3.1 General-purpose Interrupt Processing

A general-purpose interrupt is processed as shown in Figure 3.3 (2).

The CPU stores the contents of the program counter PC and the register pair AF (including the interrupt enable flag (IFF) before an interrupt) into the stack, and resets the interrupt enable flag IFF to "0" (disable interrupts). It then transfers the value of the interrupt vector "V" to the program counter, and the processing jumps to an interrupt processing program.

The overhead for the entire process from accepting an interrupt to jumping to an interrupt processing program is 20 states.



**Figure 3.3 (2). General-Purpose Interrupt Processing Flowchart**

An interrupt (maskable and non-maskable) processing program ends with a RETI instruction.

When this instruction is executed, the data previously stacked from the program counter PC and the register pair AF are restored. (Returns to the interrupt enable flag (IFF) before the interrupt.)

After the CPU reads out the interrupt vector, the interrupt source acknowledges that the CPU accepts the request, and clears the request.

A non-maskable interrupt cannot be disabled by program. A maskable interrupt, on the other hand, can be enabled or disabled by programming. An interrupt enable flip-flop (IFF) is provided on the bit 5 of Register F in the CPU. The interrupt is enabled or disabled by setting IFF to "1" by the EI instruction or to "0" by the DI instruction, respectively. If is reset to "0" by the reset operation or the acceptance of any interrupt (including non-maskable interrupt). The interrupt can be enabled after the subsequent instruction of EI instruction is executed.

Table 3.3 (1) lists the possible interrupt sources.

Table 3.3 (1) Interrupt Sources

Priority order	Type	Interrupt source	Vector Value ÷ 8	Vector Value	Start address of general-purpose interrupt processing	Start address of Micro DMA processing parameter
1	Non-maskable	SWI instruction	02H	10H	0010H	—
2		NMI (Input from NMI pin)	03H	18H	0018H	—
3	Maskable	INT0 (External input 0)	04H	20H	0020H	FF20H
4		INT0 (Timer 0)	05H	28H	0028H	FF28H
5		INT1 (Timer 1)	06H	30H	0030H	FF30H
6		INT2 (Timer 2)	07H	38H	0038H	FF38H
7		INT3 (Timer 3)	08H	40H	0040H	FF40H
8		INT1 (External input 1)	09H	48H	0048H	FF48H
9		INTRX (End of serial receiving)	0AH	50H	0050H	FF50H
10		INTTX (End of serial transmission)	0BH	58H	0058H	FF58H

The “priority order” of Table 3.3 (1) shows the order of the interrupt source to be acknowledged by the CPU when more than one interrupt are requested simultaneously.

In interrupt of 4 and 5 orders are requested simultaneously, for example, an interrupt of the “5th” priority is acknowledged after a “4th” priority interrupt processing has been completed by a RETI instruction. However, a lower priority interrupt can be acknowledged immediately by executing an EI instruction in a program that processes a higher priority interrupt.

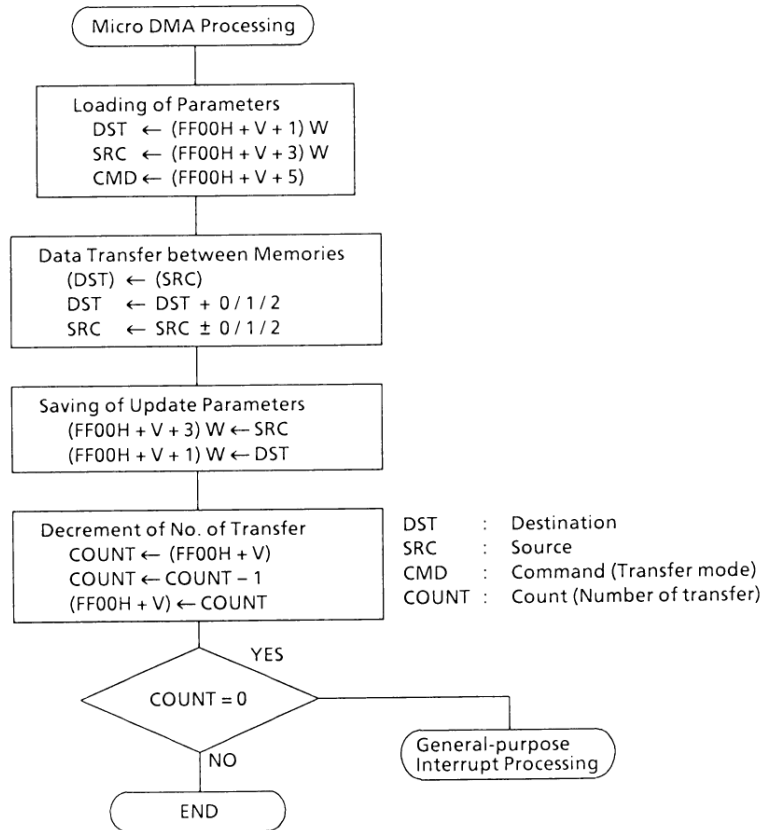
The internal interrupt controller merely determines the priority of the sources of interrupts to be acknowledged by the CPU when more than one interrupt are requested at a time. It is, therefore, unable to compare the priority of interrupt being executed with the one being requested. To permit another interrupt during a certain interrupt operation, set the interrupt enabling flag for the source of the interrupt to be allowed, and execute the EI command.

### 3.3.2 Micro DMA Processing

Figure 3.3 (3) is a flow chart of the micro DMA processing. Parameters (addresses of source and destination, and transfer mode) for the data transfer between memories are loaded by the CPU from an address modified by an interrupt vector value. After the data transfer between memories according to these parameter, these parameters are updated and saved into the original locations. The CPU then decrements the number of transfers, and completes the micro DMA processing unless the result is “0”.

If the number of transfer becomes “0”, the CPU proceeds to the general-purpose interrupt processing described in the previous item.



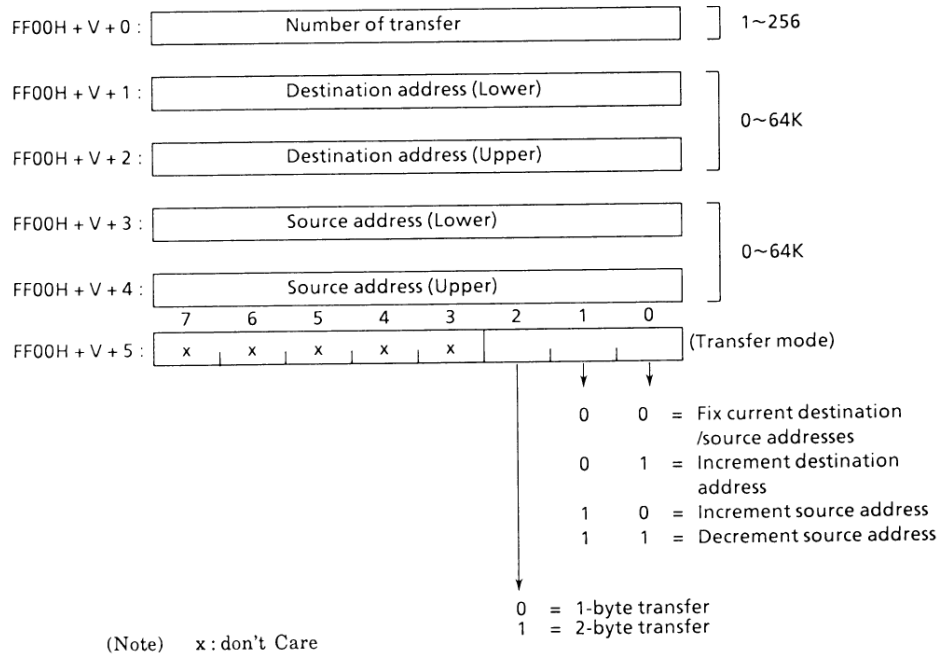


**Figure 3.3 (3). Micro DMA Processing Flowchart**

The micro DMA processing is performed by using only hardware to process interrupts mostly completed by simple data transfer. The use of hardware allows the micro DMA processing to handle the interrupt in a higher speed than the conventional

methods using software. The CPU registers are not affected by the micro DMA processing.

Figure 3.3 (4) shows the functions of parameters used in the micro DMA processing.



**Figure 3.3 (4). Parameters for Micro DMA Processing**

The parameters for the micro DMA processing are located in the internal RAM area (See Table 3.3 (1) Interrupt Sources). The start address of each parameter is “FF00H + interrupt vector value”, from which a six bytes’ space is used for the parameter. This space can be used for any other memory purposes if the micro DMA processing is not used.

The parameters normally consist of the number of transfer, addresses of destination and source, and transfer mode. The number of transfer indicates the number of data transfer accepted in the micro DMA processing.

The amount of data transferred by a single micro DMA processing is 1 or 2 bytes. The number of transfers is 256 when the number of transfers value is “00H”. Both the destination and source addresses are specified by 2 byte data. The address space available for the micro DMA processing ranges from 0000H to FFFFH.

Bits 0 and 1 of the transfer mode indicates the mode updating the source and/or destination, and the bit 2 indicates the data length (1 byte or 2 bytes).

Table 3.3 (2) shows the relation between the transfer mode and the result of updating the destination/source addresses.

Table 3.3 (2) Addresses Updated by Micro DMA Processing

Transfer Mode	Function	Destination address	Source address
000	1-byte transfer: Fix the current source/destination addresses	0	0
001	1-byte transfer: Increment the destination address	+1	0
010	1-byte transfer: Increment the source address	0	+1
011	1-byte transfer: Decrement the source address	0	-1
100	2-byte transfer: Fix the current source/destination addresses	0	0
101	2-byte transfer: Increment the destination address	+2	0
110	2-byte transfer: Increment the source address	0	+2
111	2-byte transfer: Decrement the source address	0	-2

In the 2 byte transfer mode, data are transferred as follows:

(Destination address) ←(Source address)

(Destination address + 1) ←(Source address + 1)

Similar data transfers are made in the modes that “decrement the source address”, but the updated address are different as shown in the Table 3.3 (2).

Address increment/decrement modes are applied to memory address space and fixed addressing modes are applied to the I/O address space. Because of that, the micro DMA was designed for both I/O to memory transfers and memory to I/O transfers.

Figure 3.3 (5) shows an example of the micro DMA processing that handles data receiving of internal serial I/O.

This is an example of executing “an interrupt processing program after serial data receiving” after receiving 7-frame data (assuming 1 frame = 1 byte for this example) and saving them into the memory addresses from FF00H to FF06H.

```

CALL SI0INIT ; Initial setting for serial addressing.
SET 1, (OFFE9H) ; Enable an interrupt for serial data
receiving.
SET 1, (OFFE9H) ; Set the micro DMA processing
mode for the interrupt.
LD (OFF50H),7 ; Set the number of transfer = 7
LDW (OFF51H),
OFF00H ; Set FF00H for the destination
address.
LDW (OFF53H),
OFFEBH ; Set FFEBH for the source (serial
receiving buffer) address.
LD (OFF55H),1 ; Set the transfer mode (1 byte
transfer: Increment destination
address.)

EI
:
:
ORG 0050H

```

Interrupt processing program  
after serial data receiving

```

RETI

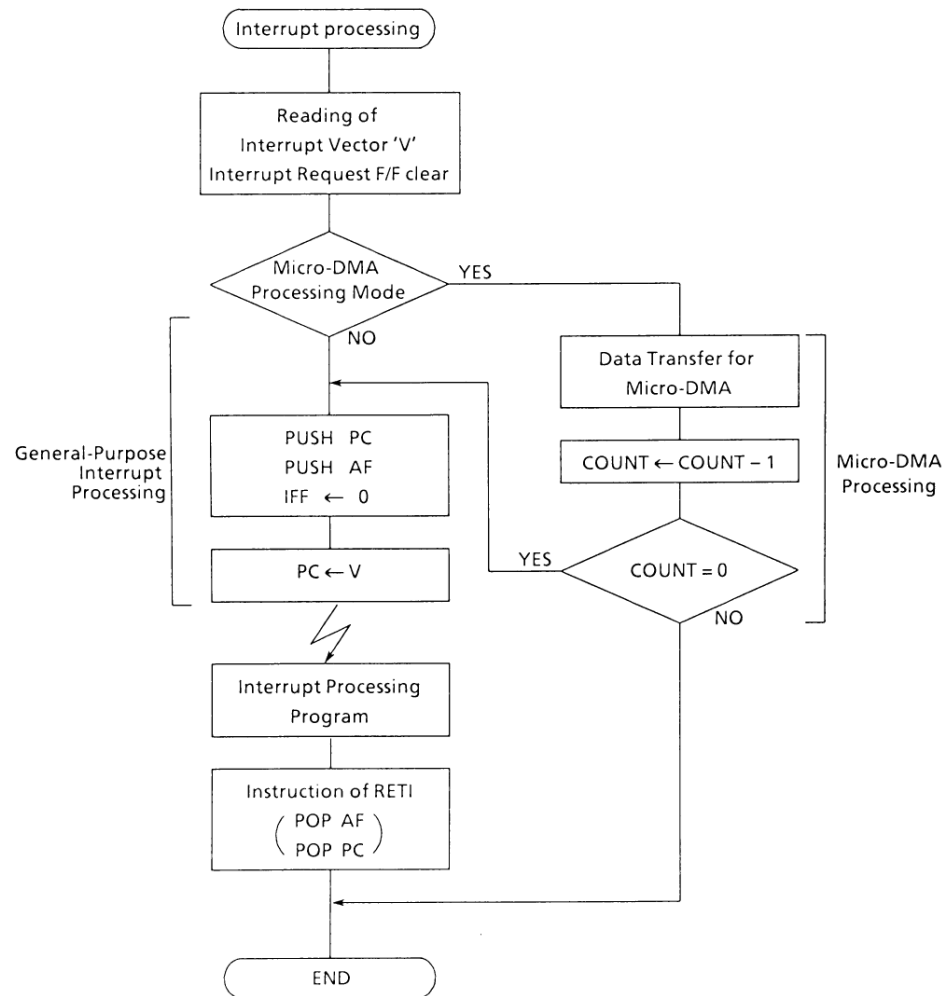
```

Figure 3.3 (5). Example of Micro DMA Processing

The bus operation in the general-purpose interrupt processing and the micro DMA processing is shown in "Table 1.4 (2) Bus Operation for Executing Instructions" in the previous section.

The micro DMA processing time (when the number of transfer is not decremented to 0) is 46 states without regard to the 1-byte/2-byte transfer mode.

Figure 3.3 (6) shows the interrupt processing flowchart.



**Figure 3.3 (6). Interrupt Processing Flowchart**

### 3.3.3 Interrupt Controller

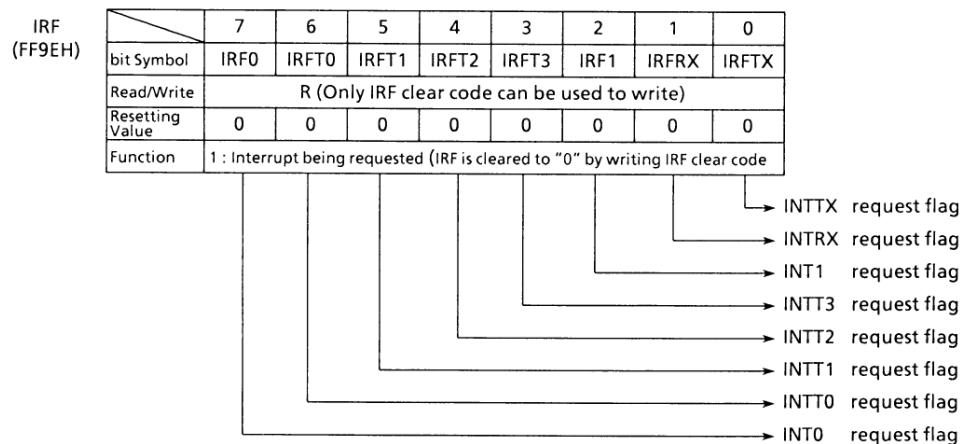
Figure 3.3 (8) outlines the interrupt circuit. The left half of this figure represents an interrupt controller, and the right side comprises the CPU interrupt request signal circuit and HALT release signal circuit.

The interrupt controller consists of Interrupt Request Flip-flops, Interrupt Enable flags, and micro DMA enable flags allocated to each of 14 channels. The Interrupt Request Flip-flops serve to latch interrupt requests from peripherals. Each flip-flop is reset to "0" when a reset or interrupt is acknowledged by the CPU and the vector of the interrupt channel is read into the CPU, or when the CPU executes an instruction that clears

an Interrupt Request Flip-flop for the specified channel (write "vector divided by 8" in the memory address FFC3H). For example, by executing.

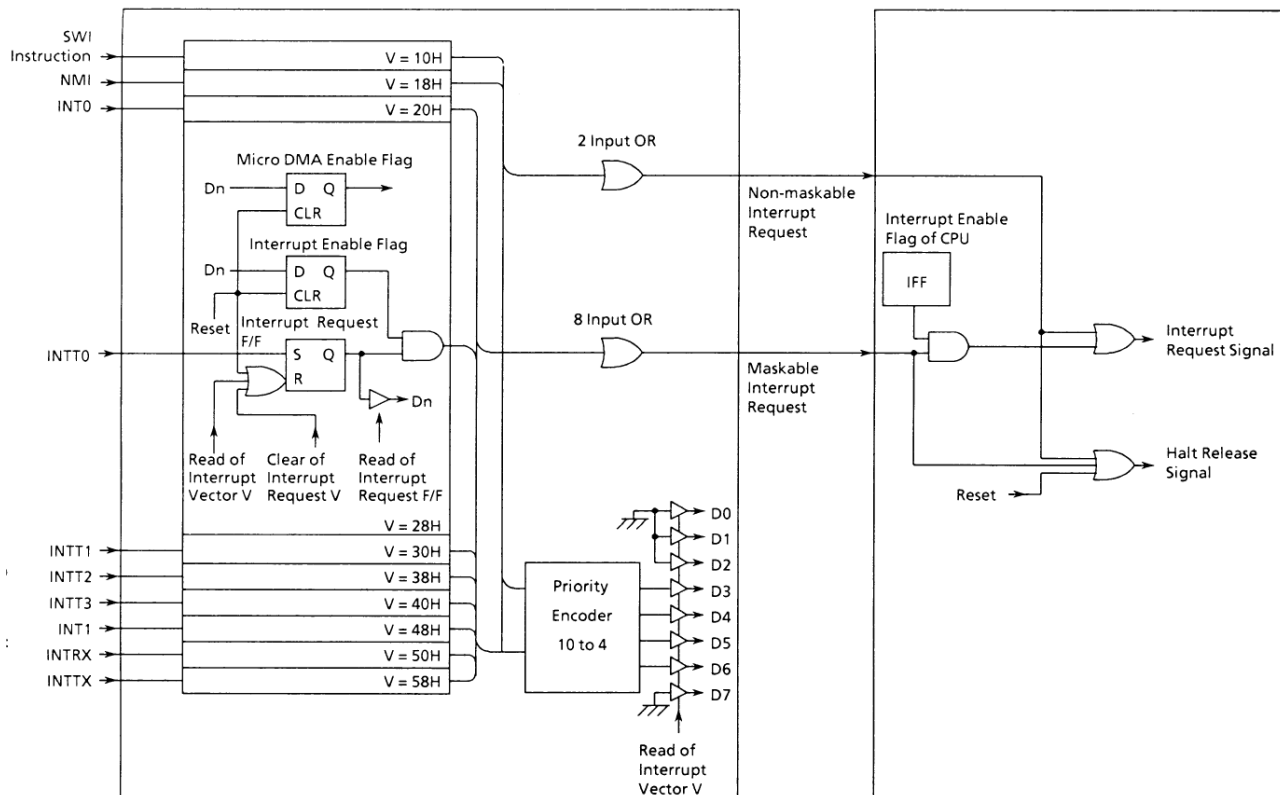
LD (FF9EH), 58H/8,

The Interrupt Request Flip-flops for the interrupt channel "INTT1" whose vector is 30H is reset to "0". The status of an Interrupt Request Flip-flops is found out by reading the memory address FFC2H or FFC3H. "0" denotes there is not interrupt request, and "1" denotes that an interrupt is request. Figure 3.3 (7) illustrates the bit configuration indicating the status of Interrupt Request Flip-flops.



(Caution) Writing "vector divided by 8" into the memory address FF9EH clears the Flip-Flop for the specified interrupt request.

Figure 3.3 (7). Configuration of Interrupt Request Flip-Flops



**Figure 3.3 (8). Block Diagram of Interrupt Controller**





The interrupt enable flags provided for all interrupt request channels are assigned to the memory address FF9CH. Setting the flags to "1" enables an interrupt of the respective channel. These flags are initialized to "0" by resetting.

Clear the interrupt enable flag in the DI status.

The micro DMA enable flag also provided for each interrupt request channel is assigned to the memory address FF9DH.

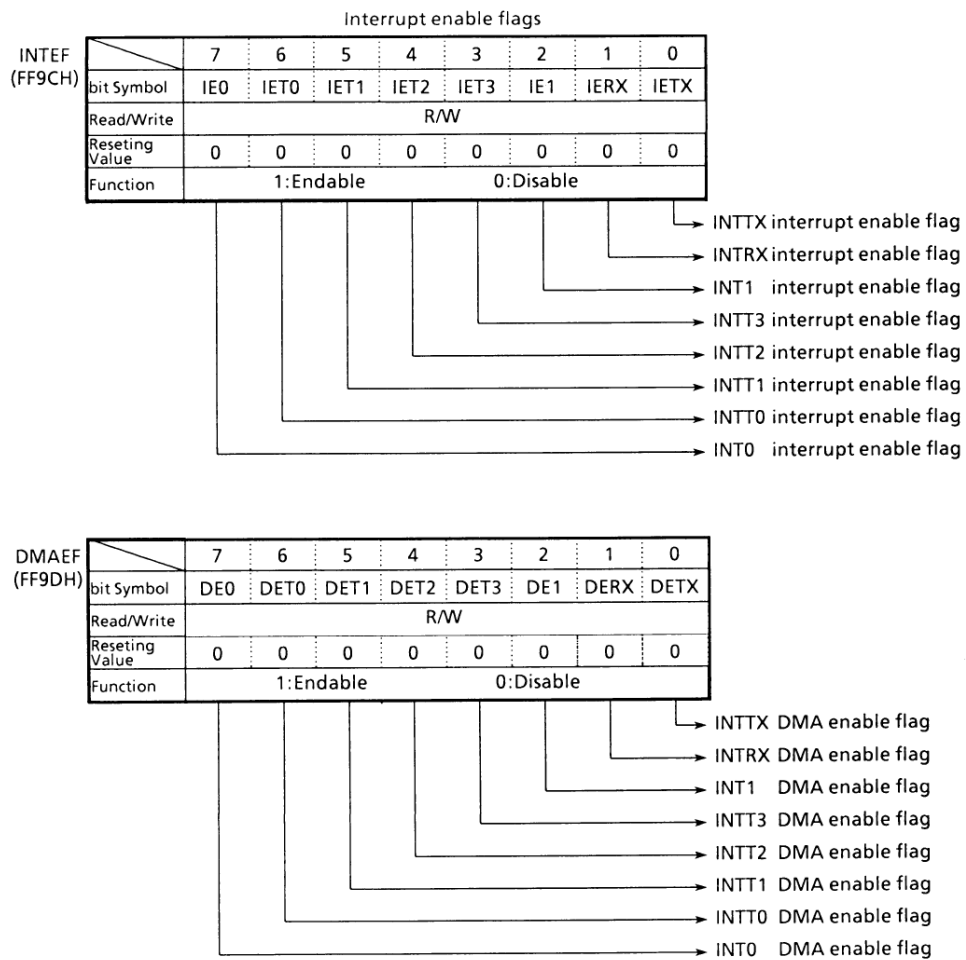
The interrupt processing for each channel is placed in the micro DMA processing mode by setting this flag to "1". This flag is initialized to "0" (general-purpose interrupt processing mode) by resetting.

Figure 3.3 (9) shows the bit configuration of the interrupt enable flags and micro DMA enable flags.

Interrupt	Common Terminal	Mode	How to set
$\overline{\text{NMI}}$	P24	 Falling edge	P2FR<NMIC> = 1
INT0	P30	 Level	INTMR<EDGE> = 0
		 Rising edge	INTMR<EDGE> = 1
INT1	P31	 Rising edge	-

Attention should be paid to the following three modes having special circuits:

INT0 Level mode	<p>IF INT0 is not an edge-based interrupt, the function of Interrupt Request Flip-flop is cancelled. Therefore, the interrupt request signal must be held until the interrupt request is acknowledged by the CPU. A change in the mode (edge to level) automatically clears the interrupt request flag.</p> <p>When the CPU has been put in the interrupt response sequence with INT0 level mode, it is necessary to leave INT0 at "1" until the second bus cycle of the interrupt response sequence is completed. Also, "1" must always be held until HALT is cleared when using the INT0 level mode to clear HALT. (Use care to prevent noise changing "1" back to "0".)</p> <p>When switching from the level mode to the edge mode, the interrupt request flag set in the level mode is not cleared; therefore, use the following sequence to clear the interrupt request flag.</p> <pre>DI LD (0FF9FH), 01H: switch from level to edge LD (0FF9EH), 04H: clear interrupt request flag EI</pre>
INTRX level mode	The Interrupt Request Flip-flop is cleared only by resetting or reading the serial channel receiving buffer, and not by an instruction.



**Figure 3.3 (9). Interrupt/Micro DMA Enable Flags**



### 3.4 Standby Function

When a HALT instruction is executed, the TMP90C400 selects one of the following modes as determined by the halt mode setting register:

- (1) RUN: Suspends only the CPU operation. The power consumption remains unchanged.
- (2) IDLE1: Suspends all internal circuits except the internal oscillator. In this mode, the power consumption is less than 1/10 of that in the normal operation.
- (3) IDLE2: Operate only the internal oscillator and specific internal I/O devices. The power consumption is about 1/3 of that in the normal operation.
- (4) STOP: Suspends all internal circuits including the internal oscillator. In this mode, the power consumption is considerably reduced.

The HALT mode set register (STBMOD <HALTM 1, 0> is assigned to the bits 2 and 3 of the memory address FFD8H in the internal I/O register area (other bits are used to control other functions). The register is reset to "00" (RUN mode) by resetting.

These HALT state can be released by an interrupt request or reset. The methods for releasing the HALT status are shown in Table 3.4 (2). Either a non-maskable or maskable interrupt with EI (enable interrupt) condition is acknowledged and interrupt processing is processed. A maskable interrupt with DI instruction that follows the HALT instruction, but the interrupt request flag is held at "1".

When the halt status is released by reset, however, note that it is not possible to hold the status (including built-in RAM) in effect immediately before entering the STOP status. In this case, it is recommended that an interrupt request be used for releasing.

STBMOD (FF8FH)	7	6	5	4	3	2	1	0
bit Symbol	-	-	-	-	HALTM1	HALTM1	EXF	DRIVE
Read/Write	-	-	-	-	R/W		R	R/W
Resetting Value	-	-	-	-	0	0	Undefined	0
Function	-	-	-	-	Stand-by mode 00:RUN mode 01:STOP mode 10:IDLE1 mode 11:IDLE2 mode		Invert each time EXX instruction is executed	1:to drive pin in STOP mode

See "3.4.4 STOP mode"

Exchange flag  
See "3.1.2 Registers"

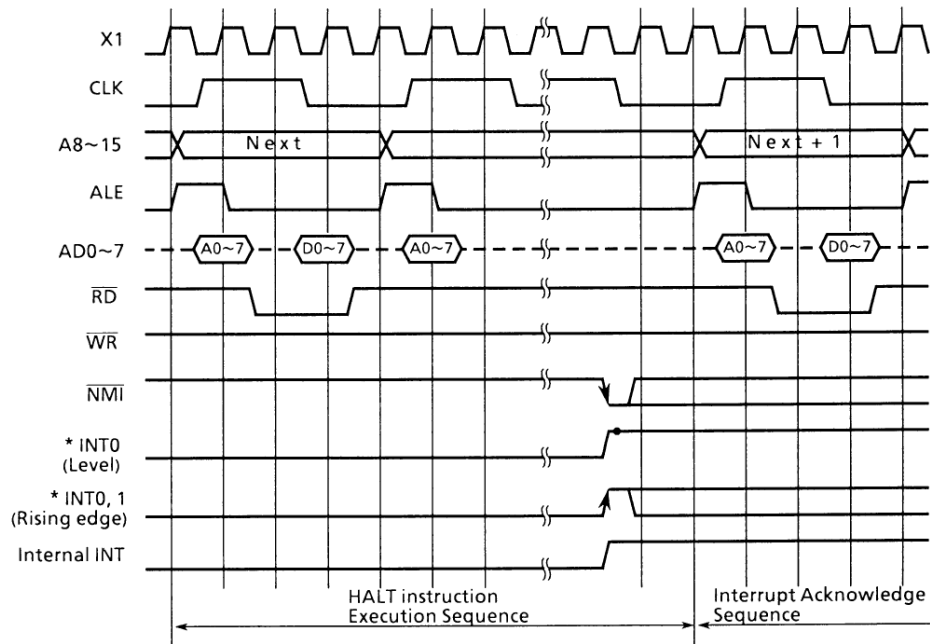
Figure 3.4 (1). HALT Mode Set Register

### 3.4.1 RUN Mode

Figure 3.4 (2) shows the timing for releasing the HALT state by interrupts in the RUN/IDLE 2 mode.

In the RUN mode, the system clock in the MCU continues to operate even after a HALT instruction is executed. Only the

CPU stops executing the instruction. Until the halt state is released, the CPU repeats dummy cycles. In the halt state, an interrupt request is sampled with the rising edge of the CLK signal



\*: The halt state can be released by external interrupt request (INT0, INT1) only in the RUN mode.

Figure 3.4 (2). Timing Chart for Releasing the Halt State by Interrupts in RUN/IDLE 2 Modes

### 3.4.2 IDLE 1 Mode

Figure 3.4 (3) illustrates the timing for releasing the HALT state by interrupts in the IDLE 1 mode.

In the IDLE 1 mode, only the internal oscillator and the watchdog timer operate. The system clock in the MCU stops, and the CLK signal is fixed at the "1".

In the halt state, an interrupt request is sampled asynchronously with the system clock, however the HALT release (restart of operating) synchronously with the system clock.

Note: Interrupt requests except external non-maskable interrupt (NMI) are prohibited through the HALT period in this mode.

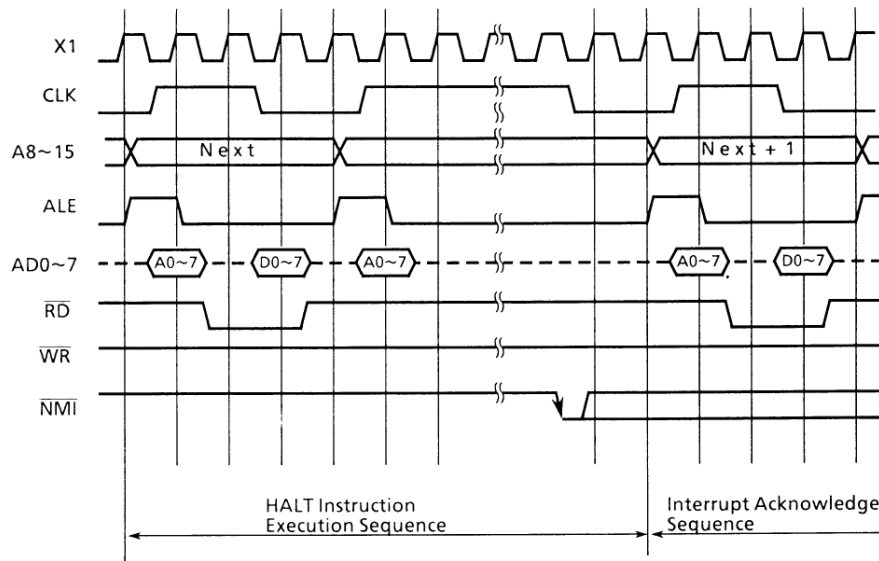


Figure 3.4 (3). Timing Chart of HALT Released by Interrupts in IDLE1 Mode

### 3.4.3 IDLE 2 Mode

Figure 3.4 (2) shows the timing of HALT release caused by interrupts in the RUN/IDLE 2 mode.

In the IDLE 2 mode, the halt state is released by an interrupt with the same timing as in the RUN mode, except the internal operation of the MCU. In the RUN mode, only the CPU stops executing the current instruction, and the system clock is supplied to all internal devices. In the IDLE 2 mode, however, the system clock is supplied to only specific internal I/O devices. As a result, the halt state in the IDLE 2 mode requires only a 1/3 of the power consumed in the RUN mode. In the IDLE 2 mode, the system clock is supplied to the following I/O devices:

- 8-bit timer
- Serial interface

Note: Interrupt requests by external interrupts (INT0, INT1) are prohibited through the HALT period in this mode.

### 3.4.4 STOP Mode

The STOP mode is selected to stop all internal circuits including the internal oscillator. In this mode, all pins except special pins are put in the high-impedance state, independent of the internal operation of the MCU.

All interrupt request are prohibited through the HALT period in this mode. (Note: that the external non-maskable interrupt pin ( $\overline{\text{NMI}}$ ) should be kept as "1".) By resetting, this mode can be cleared. However programmable pull-up resistor port remains as pulled up.

Table 3.4 (1) summarizes the state of these pins in the STOP mode. Note, however, that the pre-halt state (The status prior to execution of HALT instruction) of all output pins can be retained by setting the internal I/O register STBMOD <DRIVE (drive enable: bit 0 of memory address FF8FH) to "1". The content of this register is initialized to "0" by resetting.

The internal oscillator can also be restarted by the input of the  $\overline{\text{RESET}}$  signal at "0" to the CPU. In the Reset restart mode, however, the normal operation may not be performed in order to get the quick response of MCU when the power is turned on (Power on Reset). It is due to the unstable clock supplied immediately after restarting the internal oscillator. To avoid this, it is necessary to keep the  $\overline{\text{RESET}}$  signal at "0" long enough to release the HALT state in the STOP mode.

Table 3.4 (1) State of Pins in STOP Mode

	In/Out	DRVE = 0	DRVE = 1
P0	Input mode Output mode	– –	IN OUT
P1	Input mode Output mode	– –	IN OUT
P20 ~ P25	Input mode Output mode	– –	IN OUT
P26 P27	Output pin Output pin	– –	IN OUT
P30 ~ P33	Input mode Output mode	– –	IN* OUT
P34 ~ P37	Input mode Output mode	– –	IN OUT
P4	Input mode Output mode	– –	IN OUT
P5	Input mode Output mode	– –	IN OUT
P6	Input mode Output mode	– –	IN IN*
ALE	Input pin	"0"	"0"
CLK	Output pin	–	"1"
$\overline{\text{RESET}}$	Input pin	IN	IN
X1	Input pin	–	–
X2	Output pin	"1"	"1"

\*: Intermediate bias is still applied to this pin in the zero cross detect mode.

–: Indicates that input mode/input pin cannot be used for input and that the output mode/output pin have been set to high impedance.

**IN**: The input enable status.

IN: The input gate is operating. Fix the input voltage at either "0" or "1" to prevent the pin floating.

OUT: The output status.

Note: It is necessary to leave INTO at "1" until the second bus cycle of the interrupt response sequence is completed, when the STOP mode is released by the level mode of INTO.

**Table 3.4 (2) I/O Operation During Halt and How to Release the Halt Command**

Halt mode		Run	Idle2	Idle1	Stop	
STBMOD <HALTM1, 0>		00	11	10	01	
Operation Block	CPU	Halt				
	I/O port	Keeps the state when the halt command was executed.			See Table 3.4 (1)	
	8-bit timer	Operation		Halt		
	Serial interface					
	Interrupt controller					
	NMI	○	○	○	-	
Halt Releasing Source	Interrupt	INT0	○	-	-	-
		INTT0	○	○	-	-
		INTT1	○	○	-	-
		INTT2	○	○	-	-
		INTT3	○	○	-	-
		INT1	○	-	-	-
		INTRX	○	○	-	-
		INTTX	○	○	-	-
Reset	○	○	○	○		

○: Can be used to release the halt command.  
 -: Cannot be used to release the halt command.

### 3.5 Function of Ports

The TMP90C400 contains a total of 56 pins (TMP90C401: 38-pins) input/output ports. These ports function not only for

the general-purpose I/O but also for the input/output of the internal CPU and I/O. Table 3.5 describes the functions of these ports.

**Table 3.5 Functions of Ports**

Port name	Pin name	No. of pins	Direction	Direction set unit	Resetting Value	Pin name for internal function
Port 0	P00 ~ P07	8	I/O	Byte	Input	AD0 ~ AD7
Port 1	P10 ~ P17	8	I/O	Byte	Input	A8 ~ A15
Port 2	P20 ~ P23	4	I/O	Bit	Input	$\overline{\text{NMI}}$ }With programmable $\overline{\text{WAIT}}$ }pull-up resistor $\overline{\text{RD}}$ $\overline{\text{WR}}$
	P24	1	I/O	Bit	Input	
	P25	1	I/O	Bit	Input	
	P26	1	Output	–	Output	
Port 3	P27	1	Output	–	Output	
	P30	1	I/O	Bit	Input	INT0
	P31	1	I/O	Bit	Input	INT1
	P32	1	I/O	Bit	Input	T10
	P33	1	I/O	Bit	Input	T12 }With programmable
	P34	1	I/O	Bit	Input	T01 }pull-up resistor
	P35	1	I/O	Bit	Input	RxD
P36	1	I/O	Bit	Input	SCLK	
P37	1	I/O	Bit	Input	TxD	
Port 4	P40 ~ P47	8	I/O	Bit	Input	–
Port 5	P50 ~ P57	8	I/O	Bit	Input	With programmable pull-up resistor
Port 6	P60 ~ P67	8	I/O	Bit	Input	–

These port pins function as the general-purpose input/output ports by resetting. The port pins, for which input or output is programmably selectable, function as input ports by resetting. A separate program is required to use them for an internal function.

The TMP90C401 functions in the same way as the TMP90C400 except:

- Port 0 always functions as Address/data bus (AD0 to AD7).
- Port 1 always functions as Address bus (A8 to A15).
- P26 and P27 of port 2 always function as RD and WR pins, respectively.

### 3.5.1 Port 0 (P00 ~ P07)

Port 0 is an 8-bit general-purpose I/O port P0 whose I/O function is specified by the control register P0CR in bit. All bits of the control register are initialized to "0" by resetting, whereby Port 0 turns to the input mode, and the contents of the output latch register are undefined.

In addition to the general-purpose I/O port function, it functions as an address/data bus (AD0 ~ AD7). When accessing an external memory, it automatically functions as an address/data bus and clears P0CR to "0".

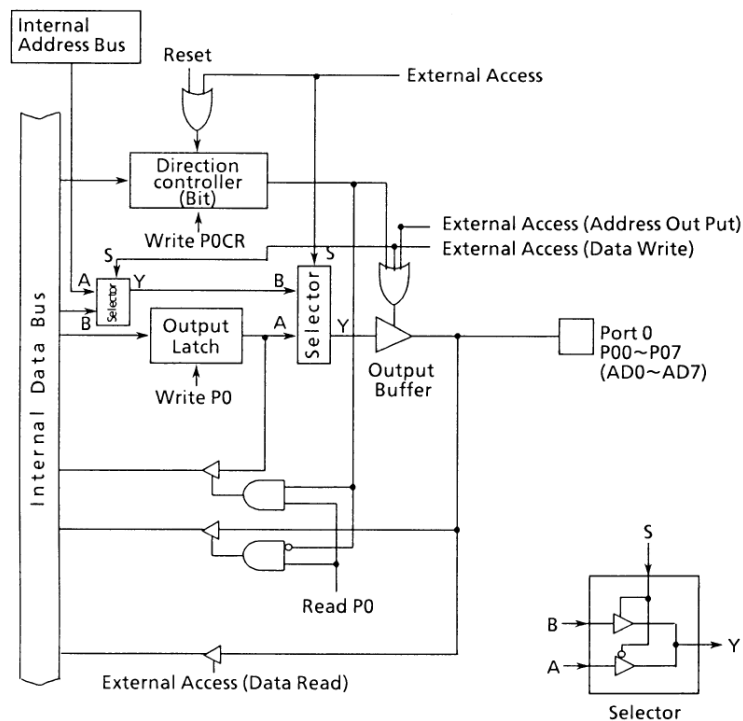


Figure 3.5 (1). Port 0



### 3.5.2 Port 1 (P10 ~ P17)

Port 1 is an 8-bit general-purpose I/O port P1 whose I/O function is specified by the control register P1CR in bit. All bits of the output latch and the control register are initialized to "0" by resetting, whereby Port 1 is put in the input mode.

In addition to the general-purpose I/O port function, it functions as a data bus (A8 ~ A15). The address bus function

can be specified by setting the external extension control register P2FR <EXT> to "1", and also setting the Port 1 control register P1CR to the output mode. When the value of the Port 1 control is set to "0", Port 1 turns to the input mode regardless of the value of the external extension control register. The <EXT> register is reset to "0" whereby Port 1 turns to the general-purpose I/O mode.

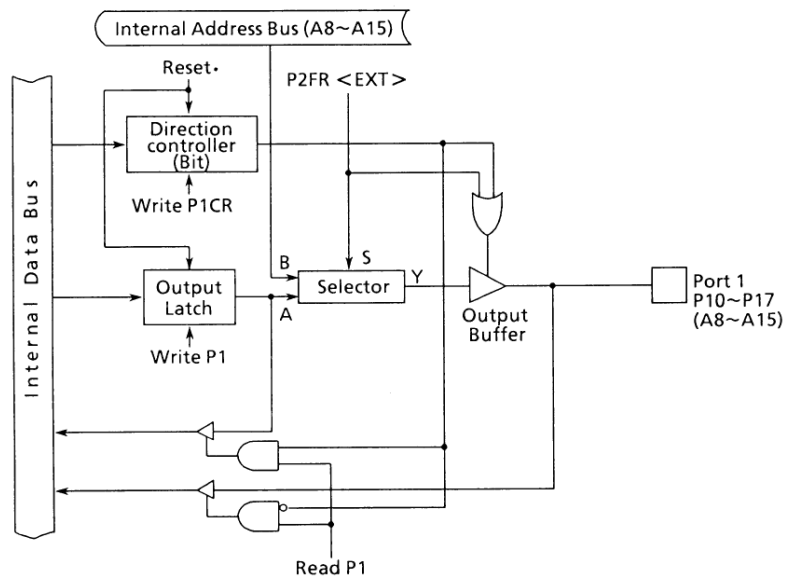
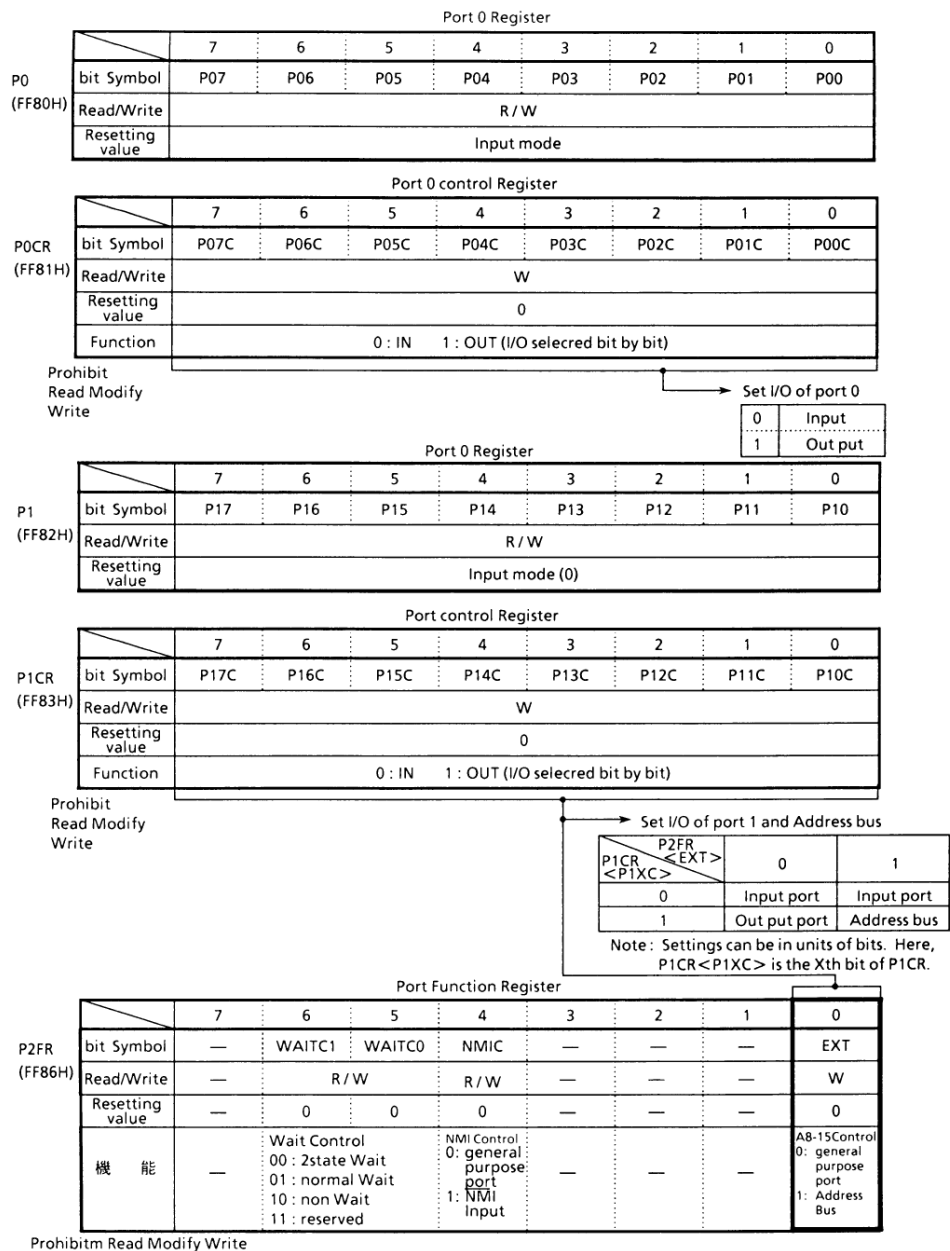


Figure 3.5 (2). Port 1



### 3.5.3 Port 2 (P20 ~ P27)

Port 2 includes a 6-bit (P20 ~ P25) general-purpose I/O port and a 2-bit (P26, P27) output port (P2: memory address FF84H). I/O functions are specified by the control register (P2CR: memory address FF85H) in bit basis.

A 6-bit I/O port has a programmable pull-up register which functions when the output latch register is set to "1". In addition to I/O function P24 controls non-maskable interrupt input pin ( $\overline{\text{NMI}}$ ), and P25 to P27 control external memory control. These additional functions are specified by the function register

(P2FR: memory address FF86H). By resetting, all bits are initialized to "1", and the control/function register to "0". As a result, I/O port turns to the pull-up input port and the output port outputs "1".

P26 and P27 automatically function as memory control pins ( $\overline{\text{RD}}$ ,  $\overline{\text{WR}}$ ) when accessing an external memory, and as general-purpose ports when accessing an internal memory. For accessing an external memory, the output latch registers P26 ( $\overline{\text{RD}}$ ) and P27 ( $\overline{\text{WR}}$ ) should be kept at "1" which is the initial value after resetting.

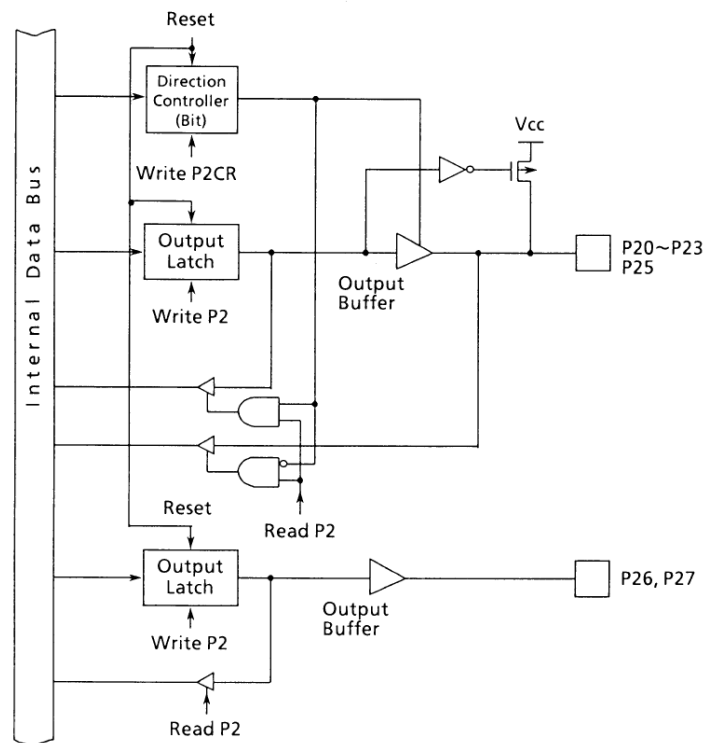
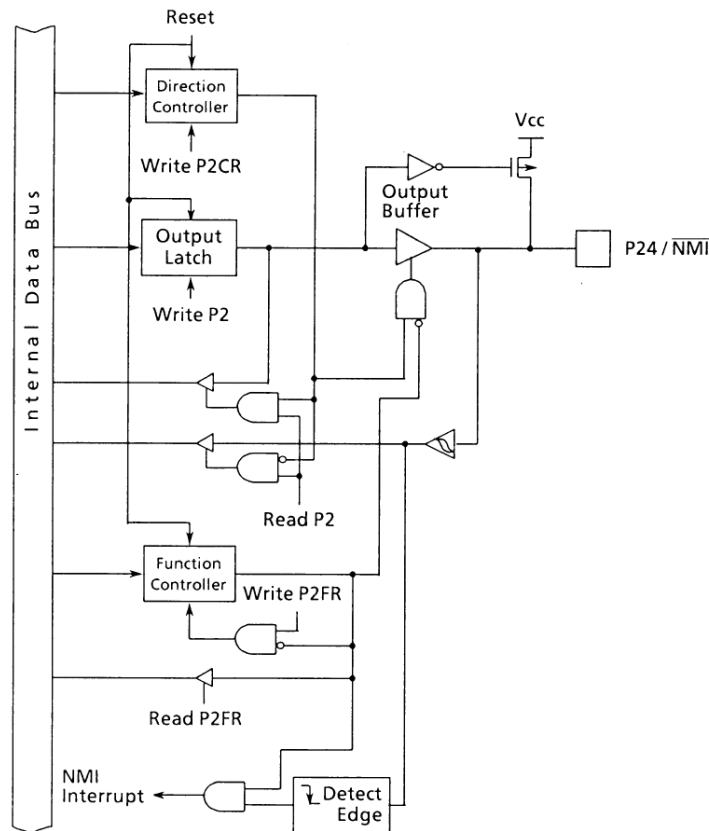


Figure 3.5 (5). Port 2 (except P24)

(1) P24/ $\overline{\text{NMI}}$ 

P24 is a general-purpose I/O port, shared with a non-maskable interrupt input pin ( $\overline{\text{NMI}}$ ). The  $\overline{\text{NMI}}$  pin is selected by the function register P2FR <NMIC>. By setting NMIC = 1, it turns to the  $\overline{\text{NMI}}$  input pin. This bit

gives the priority over the control register (P2CR). Since the  $\overline{\text{NMI}}$  pin is specified only one, the  $\overline{\text{NMI}}$  pin cannot be switched to the general-purpose port. The NMIC should be initialized to "0" by resetting in order to switch to the general-purpose I/O port mode.



**Figure 3.5 (6). Port 24**

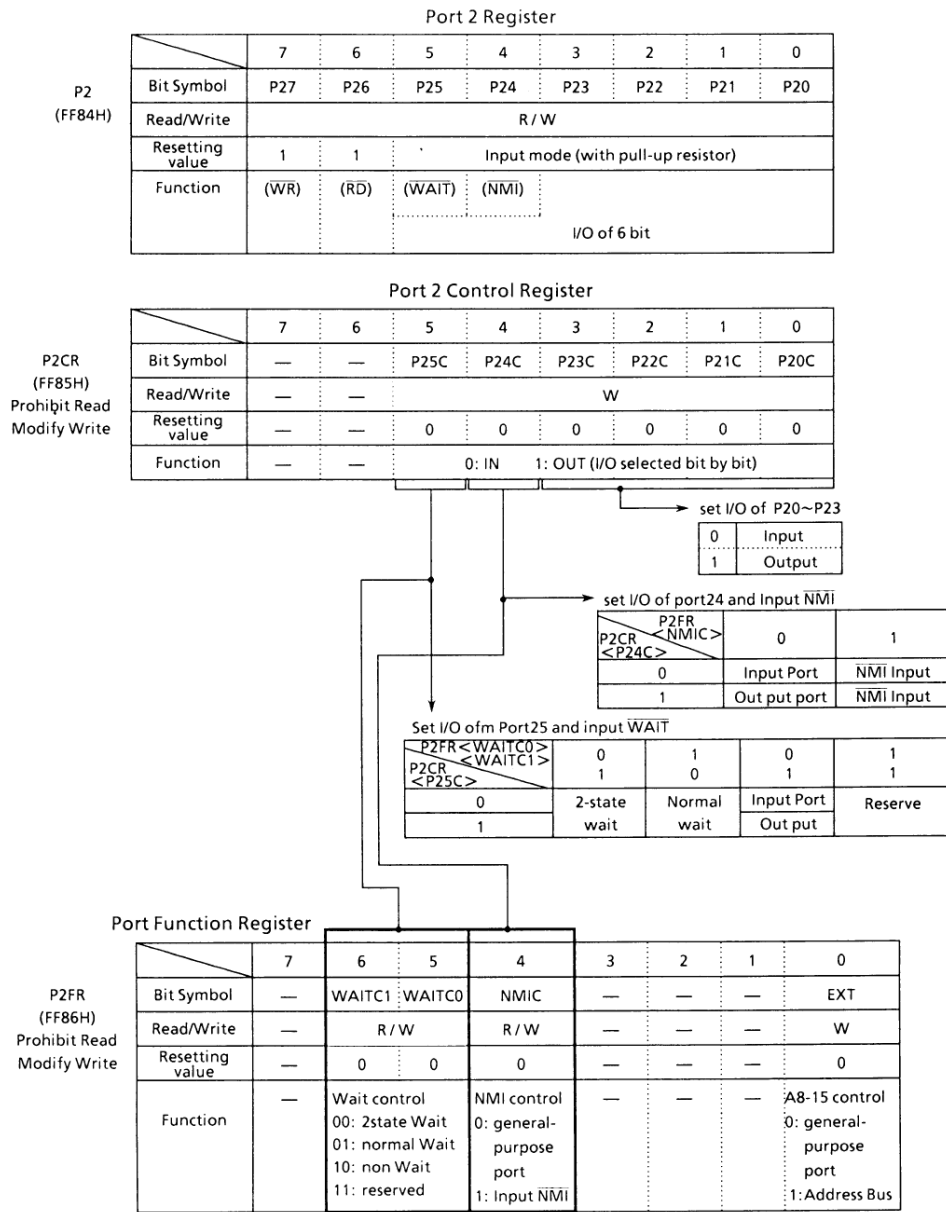


Figure 3.5 (7). Register for Port 2

**3.5.4 Port 3 (P30 ~ P37)**

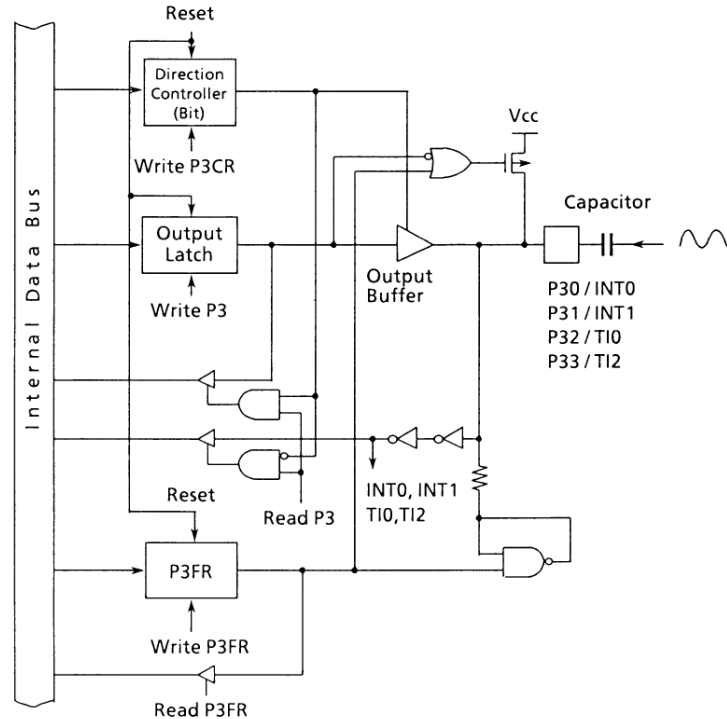
Port 3 is an 8-bit general-purpose I/O port (P3: memory address FF87H). The I/O is selected by the control register (P3CR: memory address FF88H) in bit basis. Port 3 also has a programmable pull-up resistor which functions when the output latch register is "1". By resetting, all bits are initialized to "1", and the control register to "0". As a result, Port 3 turns to the pull-up input port.

In addition to the I/O port function, Port 3 has the external interrupt request input, timer/event counter clock input, and timer output internal serial interface functions.

(1) P30 ~ P33

P30 ~ P33 are general-purpose I/O ports, shared between external interrupt request input pins (INT0, INT1) and timer/event counter clock input pins (T10, T12).

These ports have zero-cross detection circuits, which are connected to an external capacitor. This zero-cross detection disable/enable is specified by the function register (P3FR: memory address FF98H). When this register is initialized to "0", by resetting, the zero-cross detection becomes disabled.

**Figure 3.5 (8). Ports 30 ~ P33**

## (2) P34 ~ P37

P34 ~ P37 are general-purpose I/O ports, shared between the timer output pin (TO1) and internal serial interface I/O pins (RxD, SCLK, TxD). These ports are specified by the control register (P3CR: memory address FF88H) and the function register (P3FR: memory address FF98H). For example, the following is the procedures to assign P34 as TO1 pin:

- 1) Set the control register (P3CR: bit 4 of memory address FF88H) to "1" to make the output mode, and

- 2) Set the function register (TO1: bit 4 memory address FF98H) to "1".

The control and function registers are initialized to "0" by resetting, and the ports turn to the general-purpose I/O port mode.

Note: When assigning P34, P36, and P37 as TO1, SCLK, and TxD respectively, the pull-up register is executed not only by the value of the output latch register, but by that of TO1, SCLK, and TxD.

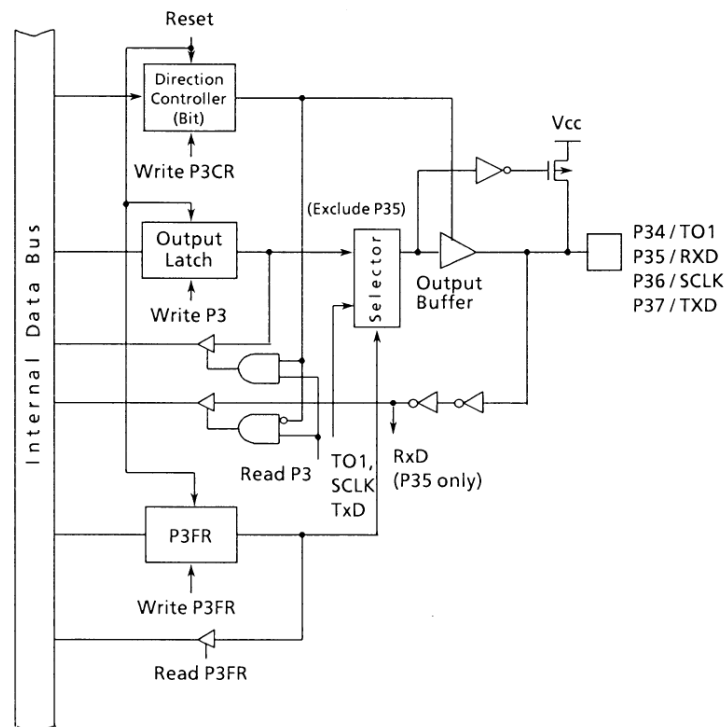
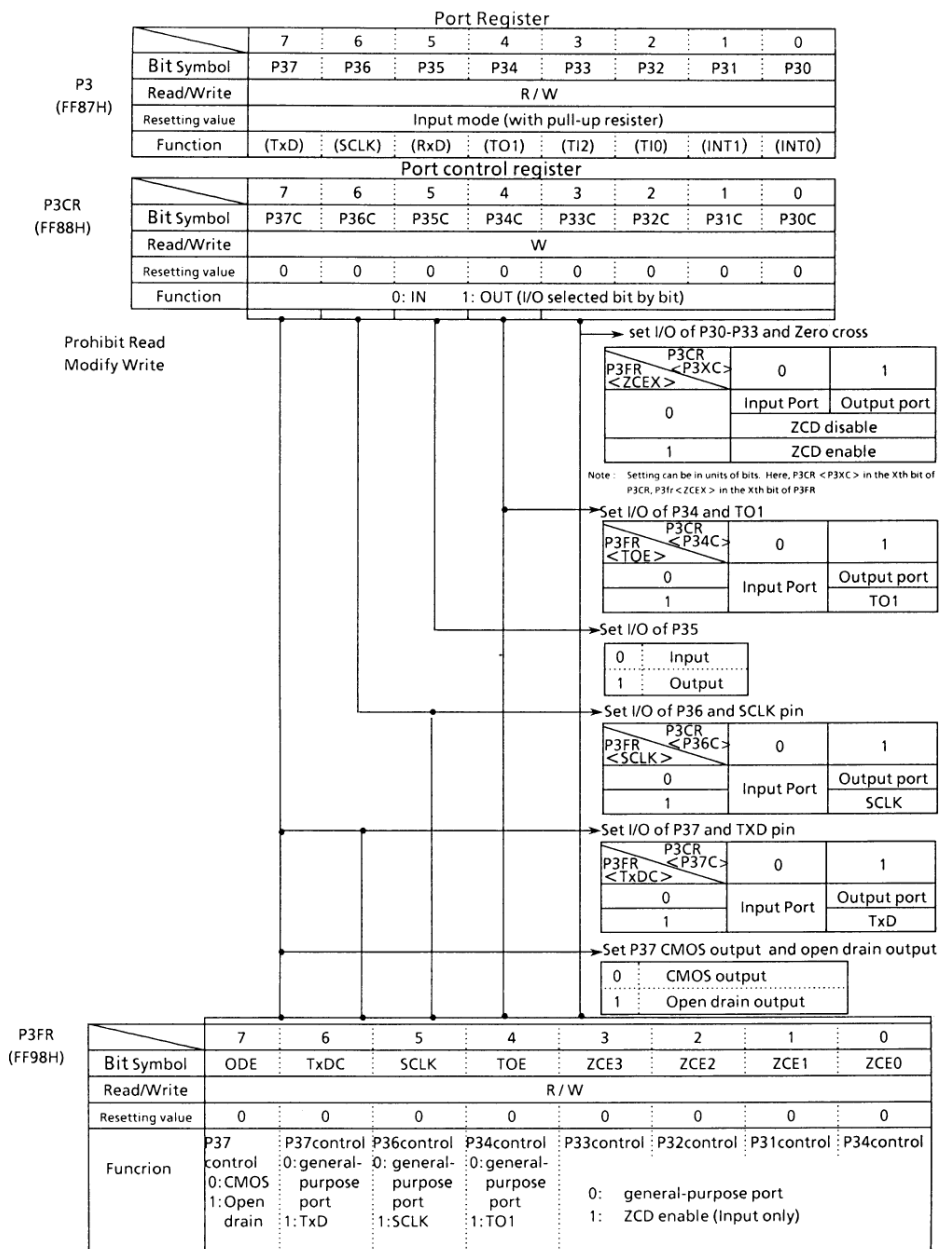


Figure 3.5 (9). Ports P34 ~ P37



**Figure 3.5 (10) Register for Port 3**



### 3.5.5 Port 4 (40 ~ P47)

Port 4 is an 8-bit general-purpose I/O port (P4: memory address FF89H). It is specified by the control register (P4CR: memory address FF8AH) in bit basis.

All bits of the output latch are initialized to "0" by resetting, and Port 4 turns to the input mode.

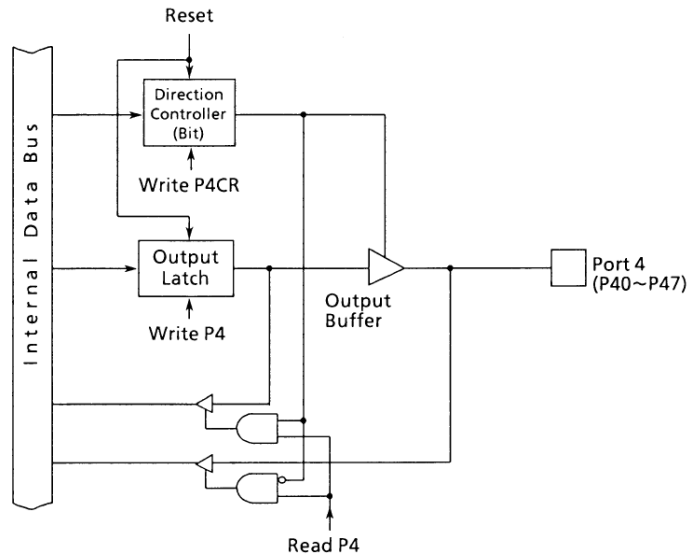
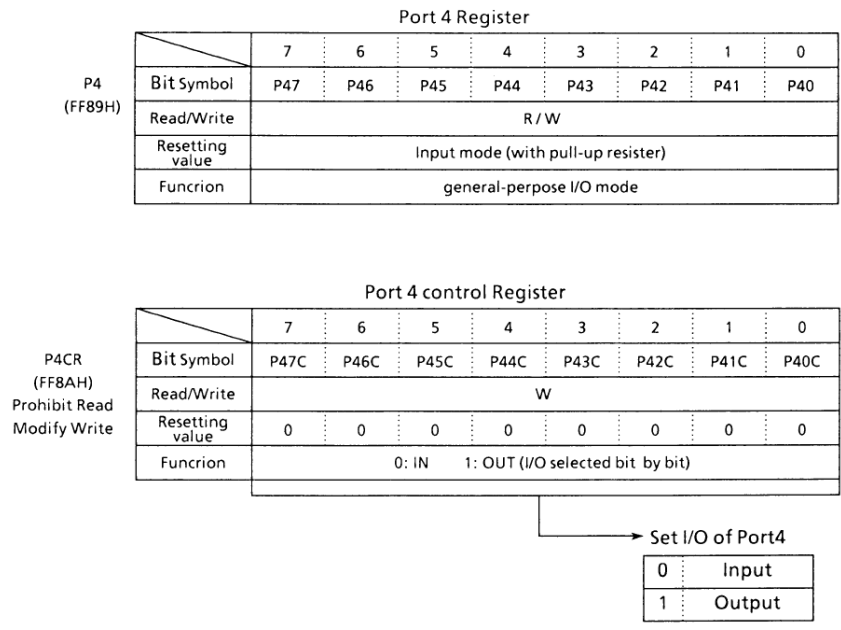


Figure 3.5 (11). Port 4



**Figure 3.5 (12). Register for Port 4**

### 3.5.6 Port 5 (P50 ~ P57)

Port 5 is an 8-bit general-purpose I/O port (P5: memory address FF8BH). It is specified by the control register (P5CR: memory address FF8CH) in bit basis.

All bits of Port 5 have the programmable pull-up register which functions when the output latch register is set to "1". All bits of the output latch are initialized to "1" and the control register to "0" by resetting, and Port 5 turns to the pull-up input port.

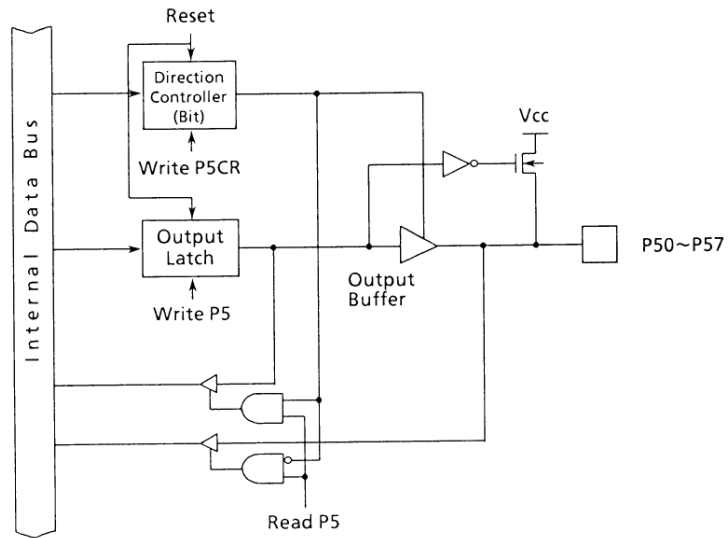


Figure 3.5 (13) Port 5

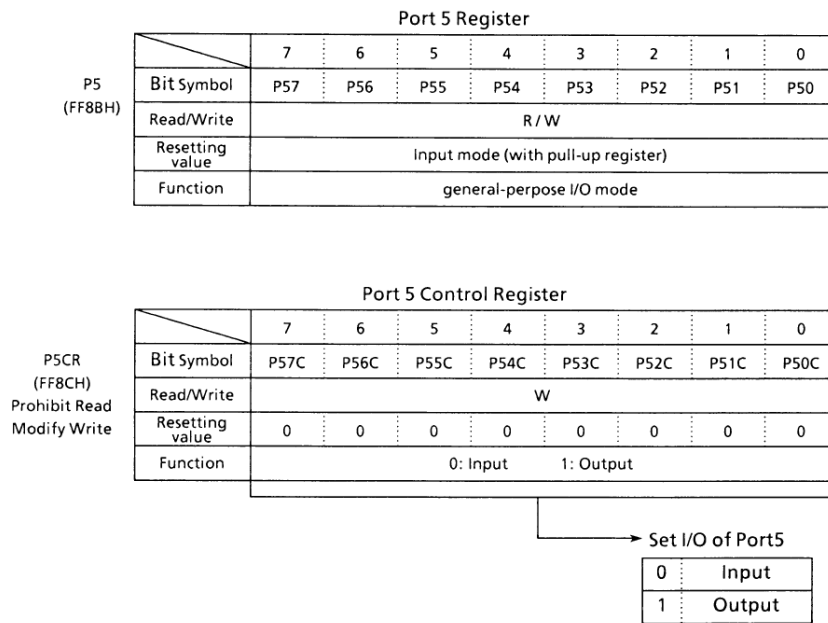


Figure 3.5 (14). Register for Port 5

### 3.5.7 Port 6 (P60 ~ P67)

Port 6 is an 8-bit general-purpose I/O port (P6: memory address FF8DH) whose function is specified by the control

register (P6CR: memory address FF8EH) for each bit. All bits of the output latch and the control register are initialized to "0" by resetting, and all bits of Port 6 enter in the input mode.

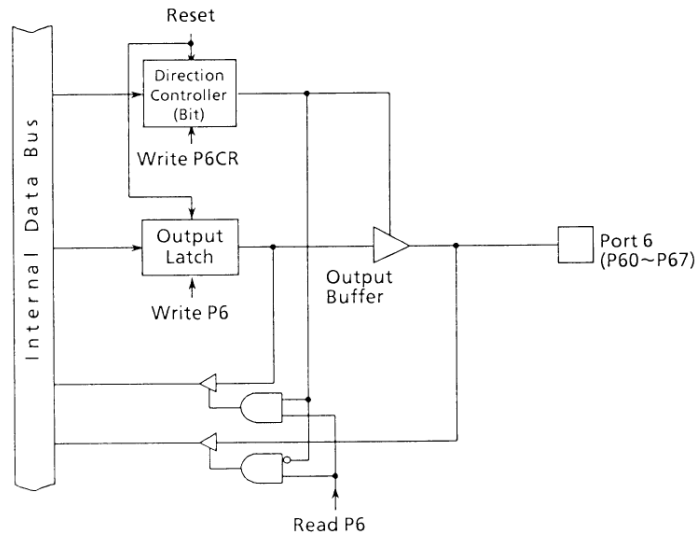


Figure 3.5 (15). Port P6

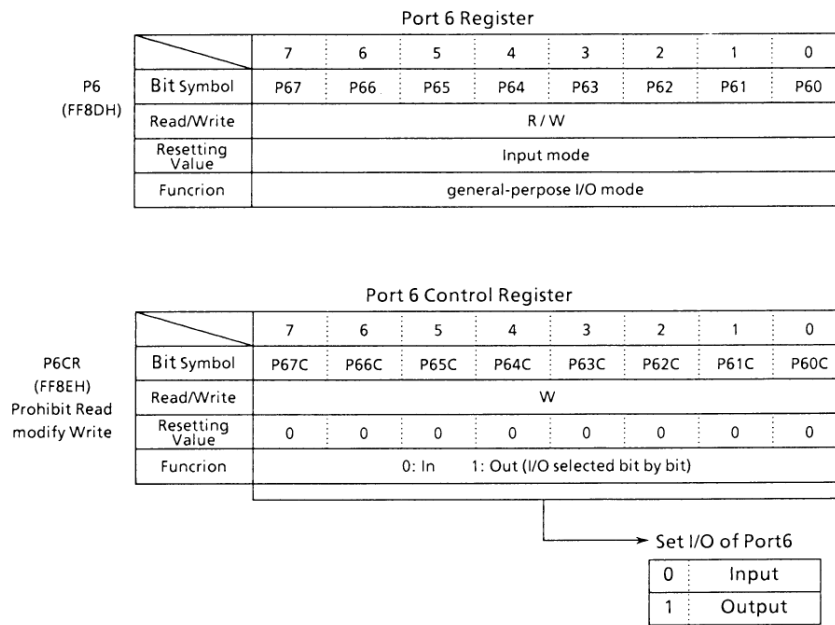


Figure 3.5 (16). Register for Port 6

### 3.6 Timers

The TMP90C400 incorporates four 8-bit timers.

The four 8-bit timers can operate independently, and also function as two 16-bit timers through cascade connection.

Timer 2 is provided with 8-bit timer/event counter and can be used as 16-bit counter by connecting with 8-bit counter or timer 3.

- 8-bit interval timer mode (4 timers)
- 16-bit interval timer mode (2 timers)
  - Possible arrangements: 8-bit x 2 and 16-bit x 1
- 8-bit programmable square wave (pulse) generation (PPG: variable duty with variable cycle) mode (Timers 1, 0)
- 8-bit pulse width modulation (PWM: variable duty with constant cycle) mode (Timer 1)
- 8-bit event counter mode (Timer 2)
- 16-bit event counter mode (Timer 2, 3)
- Software counter latch function (Timer 2, 3)

#### 3.6.1 8-bit Timers

The TMP90C400 incorporates four 8-bit interval timers (Timers 0, 1, 2 and 3), each of which can be operated independently. The cascade connection of Timer 0 and 1, or Timer 2 and 3 allows these timers used as 16-bit internal timers.

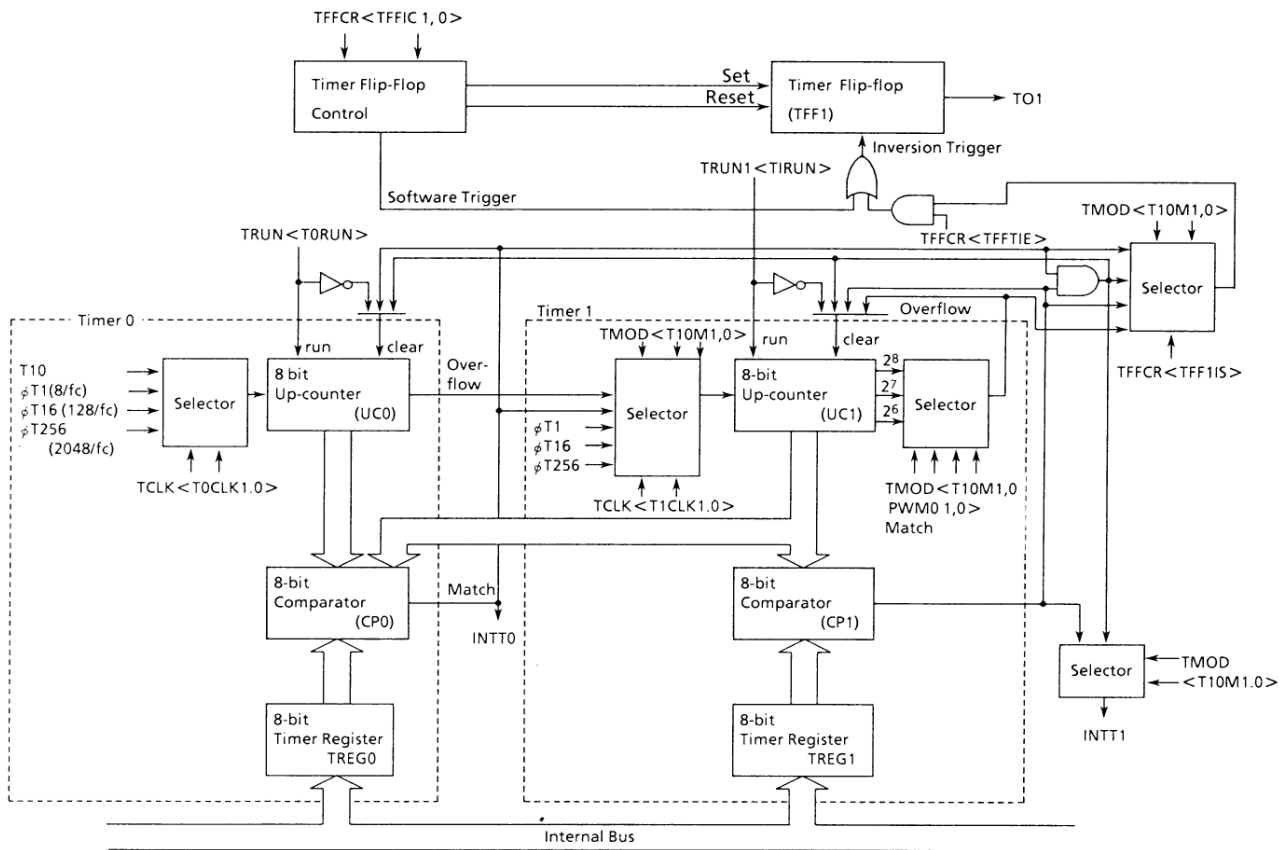
Figure 3.6 (1) shows a block diagram of the 8-bit timers (Timer 0 and Timer 1).

Figure 3.6 (2) shows a block diagram of the 8-bit timer/event counters (Timer 2 and Timer 3).

Each interval timer is composed of an 8-bit up-counter, an 8-bit timer register, with a timer flip-flop (TFF1) provided to each pair of Timer 0/1.

Internal clocks ( $\phi T1$ ,  $\phi T16$  and  $\phi T256$ ) for the input clock sources to the interval timers are generated by the 9-bit prescaler shown in Figure 3.6 (3).

Their operating modes of the 8-bit timers and flip-flops are controlled by 4 control registers (TCLK, TFFCR, TMOD and TRUN).



**Figure 3.6 (1). Block Diagram of 8-bit Timers (Timers 0 and 1)**



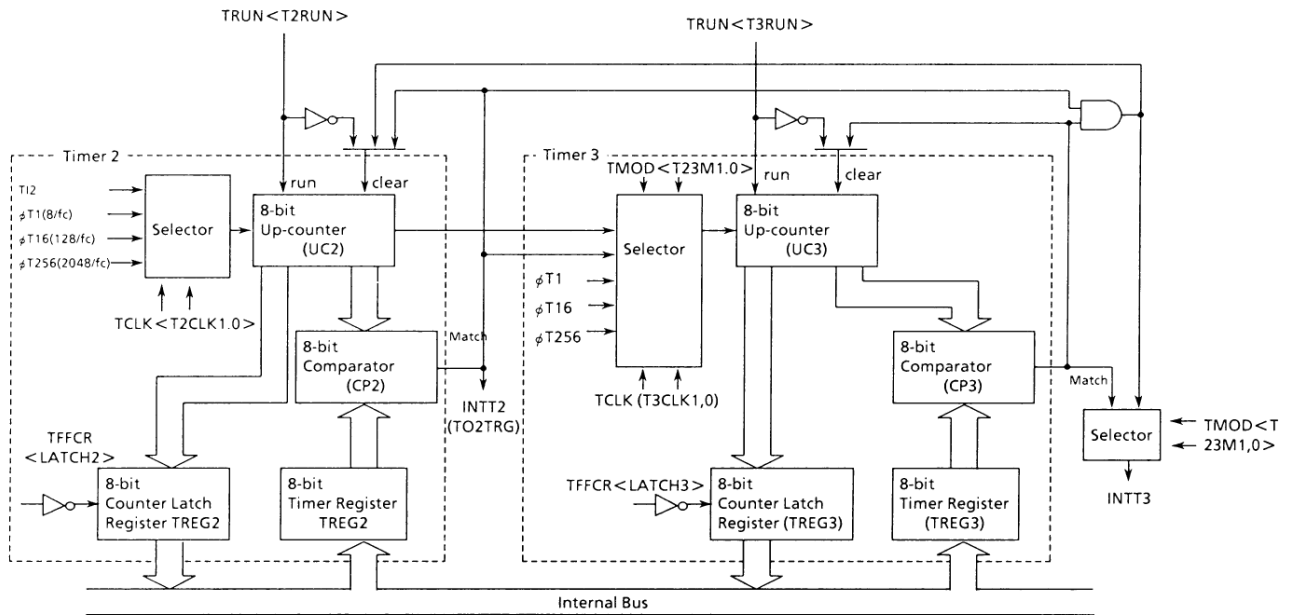


Figure 3.6 (2). Block Diagram of 8-bit Timer/Counter (Timers 2 and Timer 3)

**TMP90C400/401**

① Prescaler

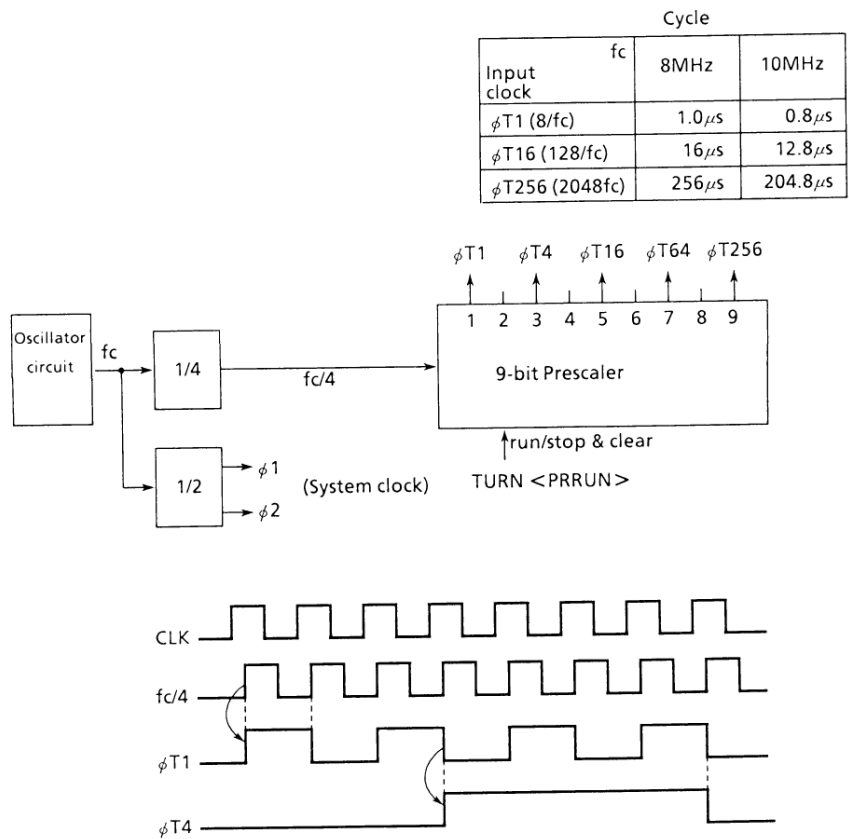
A 9-bit prescaler is provided to further divide the clock frequency already divided to a 1/4 of the frequency of the source clock ( $f_c$ ).

It generates a input clock pulse for the 8-bit timers, 16-bit timer/event counter, the baud-rate generator, etc.

For the 8-bit timers, three types of clock are generated ( $\phi T1$ ,  $\phi T16$  and  $\phi T256$ ) are used.

The prescaler can be run or stopped by using the 5th bit TRUN <PRRUN> of the timer control register TRUN. Setting <PRRUN> to "1" makes the prescaler count, and setting it to "0" clears the prescaler to stop.

<PRRUN> is initialized to "0" by resetting, and clears and stop the prescaler .



**Figure 3.6 (3). Prescaler**

## ② Up-counter

This is an 8-bit binary counter that counts up by an input clock pulse specified by an 8-bit timer clock control register (TCLK) and an 8-bit timer mode register.

The input clock pulse for Timer 0 and 2 is selected from  $\phi T1$  (8/fc),  $\phi T16$  (128/fc) and  $\phi T256$  (2048/fc) according to the setting of the TCLK register.

Example: When setting  $TCLK \langle T0CLK1, 0 \rangle = 0, 1$   $\phi T1$  is selected as the input clock pulse for Timer 0.

The input clock pulse to Timers 1 and 3 is selected according to the operating mode. In the 16-bit timer mode, the overflow output of Timers 0 and 2 is automatically selected as the input clock pulse, regardless of the setting of the TCLK register.

In the other operating modes, the clock pulse is selected among the internal clocks  $\phi T1$ ,  $\phi T16$  and  $\phi T256$ , and the output of the Timers 0 and 2 comparator (match signal) by the TCLK register setting.

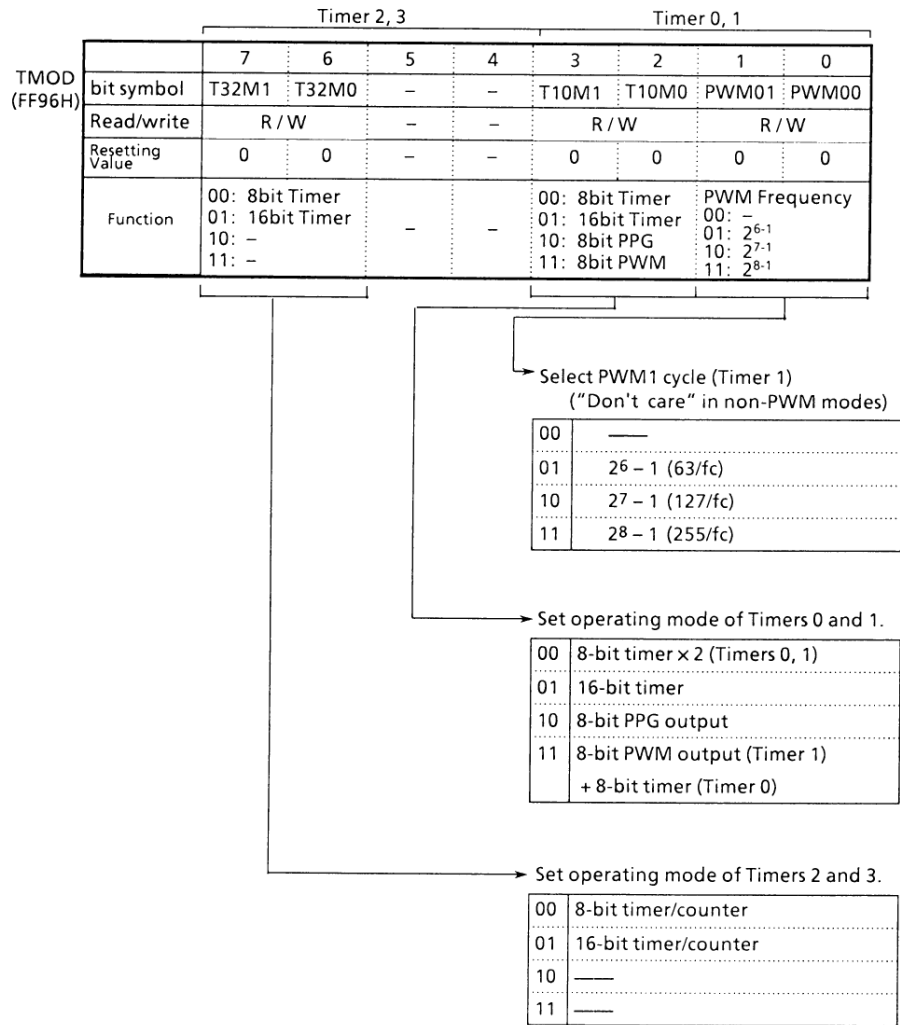
Example: If  $TMOD \langle T10M1, 0 \rangle = 0, 1$ , the overflow output of Timer 0 is selected as the input clock to Timer 1. (16-bit timer mode)

If  $TMOD \langle T10M1, 0 \rangle = 0, 0$  and  $TCLK \langle T1CLK1, 0 \rangle = 0, 1$ ,  $\phi T1$  is selected as the input clock to Timer 1. (8-bit timer mode)

The operating mode is selected by the TMOD register. This register is initialized to  $TMOD \langle T10M1, 0 \rangle = 0, 0$  /  $TMOD \langle T32M1, 0 \rangle = 0, 0$  by resetting, whereby the up-counter is placed in the 8-bit timer mode.

Functions, count, stop or clear of the up-counter can be controlled for each interval timer by the timer control register TRUN.

By resetting, all up-counters are cleared to stop the timers.



**Figure 3.5 (4). 8-bit Timer Mode Register TMOD**

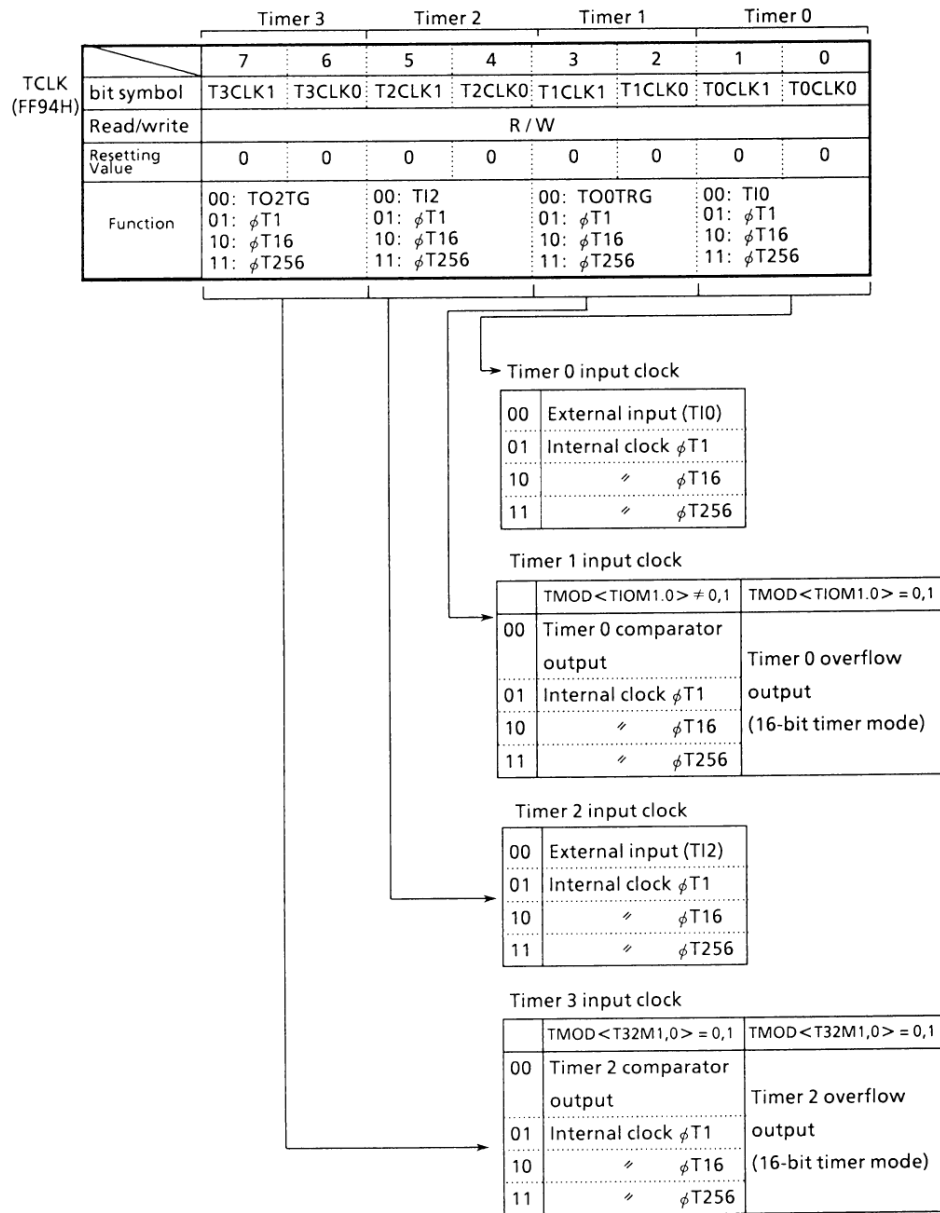
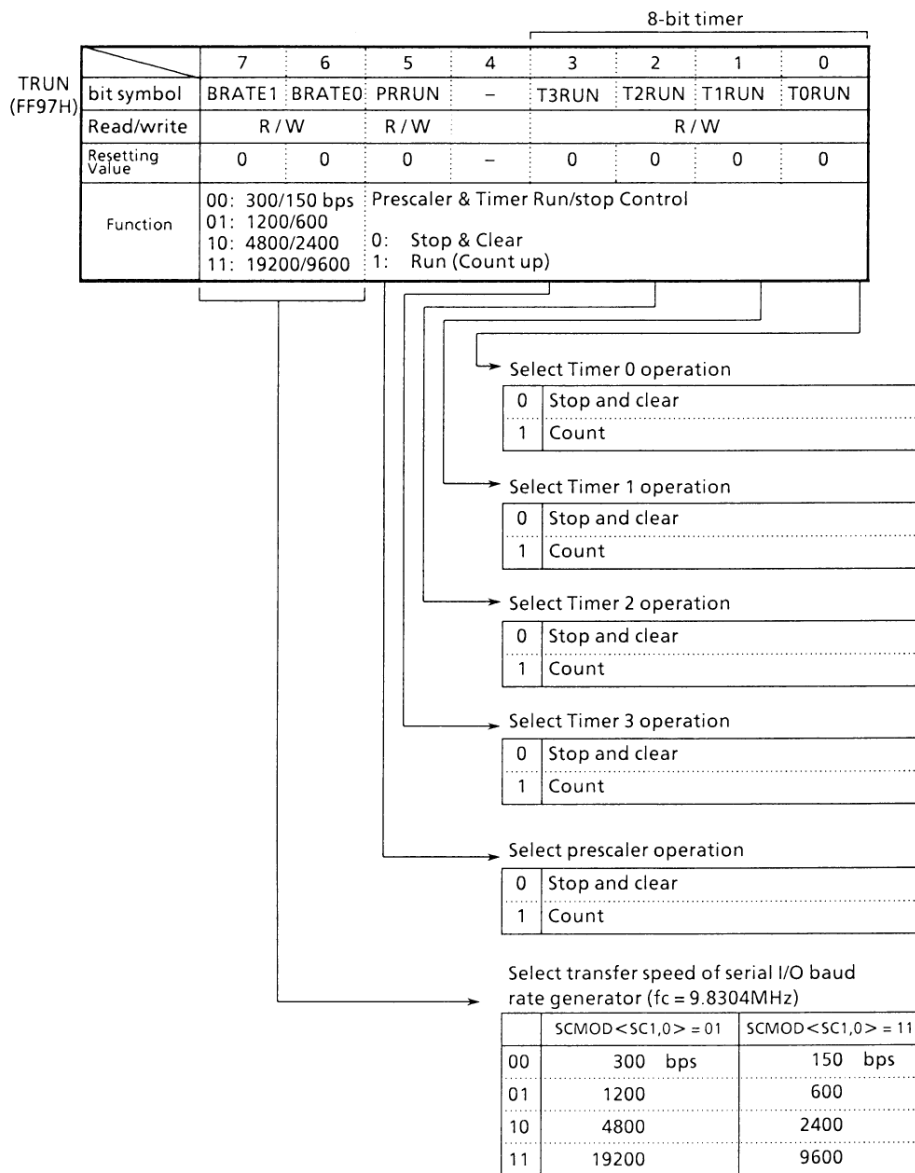


Figure 3.6 (5). 8-bit Timer Clock Control Register (TCLK)



**Figure 3.6 (6). Timer/Serial Channel Control Registers (TRUN)**

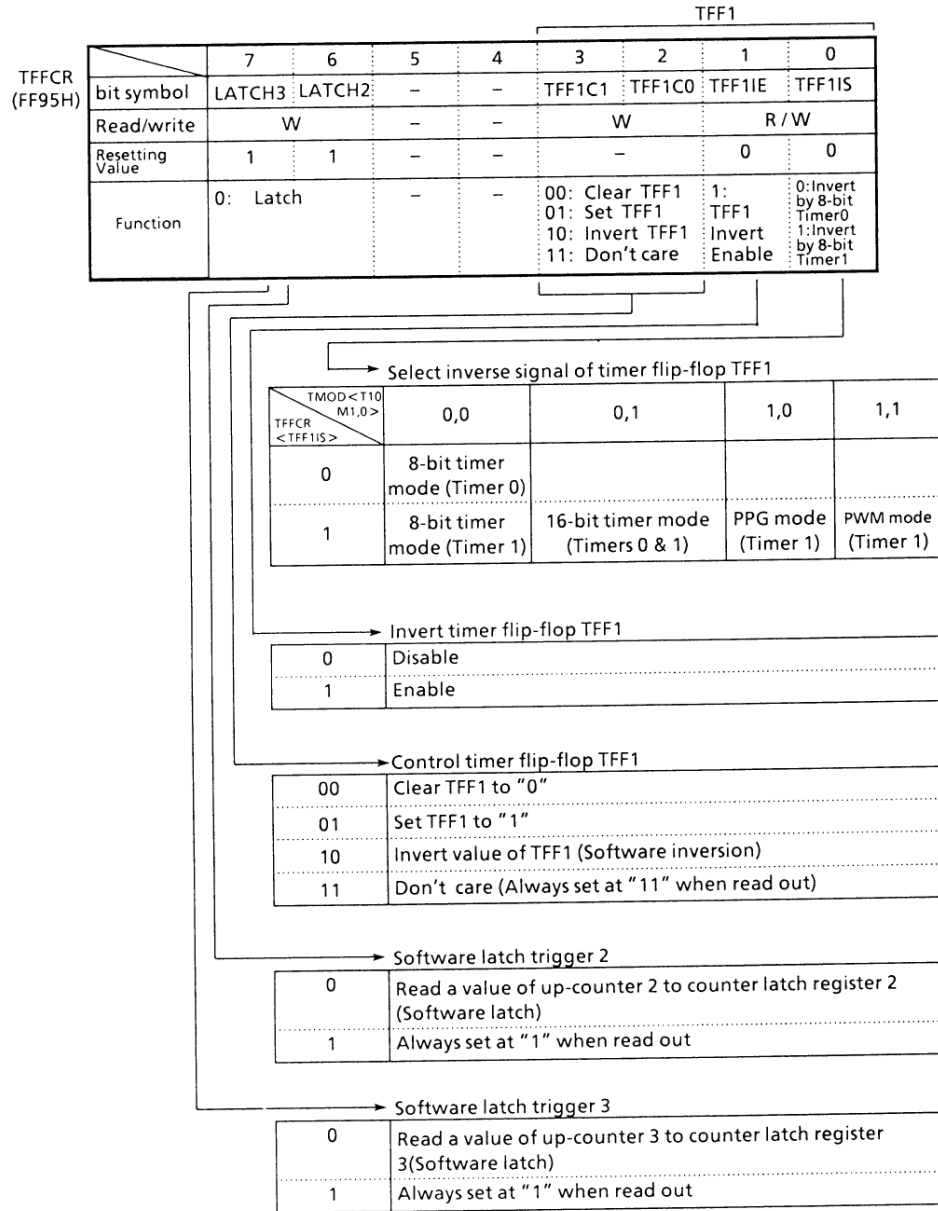
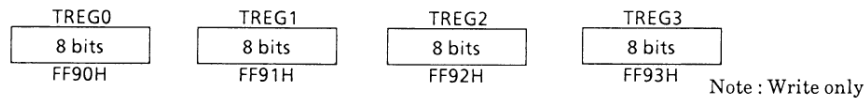


Figure 3.6 (7). 8-bit Timer Flip-Flop Control Register (TFFCR)

## ③ Timer registers



8-bit registers are provided to set the interval time. When the set value of a timer register matches that of an up-counter, the match signal of their comparators turn to the active mode. If “00H” is set, this signal becomes active when the up-counter overflows. When a new value is written to this register, it is then immediately input to the comparator.

The value of the timer register 0 and timer register 1 cannot be read out. Timer registers 2 and 3 are allocated to the same address with counter latch registers. The readout value becomes the counter latch register value. The written value becomes the timer register value.

## ④ Comparators

A comparator compares the values in an up-counter and a timer register. When they matches, the up-counter is cleared to “0”, and an interrupt signal (INTT0 ~ 3) is generated. If the timer flip-flop inversion is enabled by the timer flip-flop control register, the timer flip-flop is inverted.

## ⑤ Timer Flip-flop (Timer F/F)

The timer flip-flop is inverted by the match signal (output by comparator). Its status can be output to the timer output pin TO1 (also used as P34).

This timer F/F is controlled by a timer flip-flop control register (TFFCR).

In the case of TFF1 (Timer F/F for Timer 0 and Timer 1), the flip-flop operation is described as follows (refer to Figure 3.6 (7)):

- TFFCR <FF1IS> is a timer selection bit for inversion of TFF1. In the 8-bit timer mode, inversion is enabled by the match signal from Timer 0 if this bit is set to “0”, or by the signal from Timer 1 is set to “1”. In any other mode, <FF1IS> must be always set to “1”. It is initialized to “0” by resetting.

- TFFCR <FF1IE> controls the inversion of TFF1. Setting this bit to “1” enables the inversion and setting it to “0” disables.

FF1IE is initialized to “0” by resetting.

- The bits TFFCR <TFF1C1, 0> are used to set/reset TFF1 or enable its inversion by software. TFF1 is reset by writing “0, 0”, set by “0, 1” and inverted by “1, 0”. The 8-bit timers operate as follows:

## (1) 8-bit Timer Mode

The four interval timers 0, 1, 2 and 3 can operate independently as an 8-bit interval timer. Timer 2 and 3 do not provide timer output operation, but timer operations are the same as for Timer 0 and 1. The operation of Timer 1 is described in the following.

## ① Generating interrupts at specified intervals

Periodic interrupts can be generated by using Timer 1 (INTT1) in the following procedure:

- 1) stop Timer 1,
- 2) set the desired operating mode, input clock and cycle time in, the registers TMOD, TCLK and TREG1,
- 3) set INTT1 to “enable”, and
- 4) start the counting of Timer 1.

Example: To generate Timer 1 interrupt every 4.0μs at  $f_c = 10\text{MHz}$ , the registers should be set as follows:



	MSB	LSB								
	7	6	5	4	3	2	1	0		
TRUN	←	-	-	-	-	-	-	0	-	Stop Timer 1, and clear it to "0".
TMOD	←	-	-	-	-	0	0	X	X	Set the 8-bit timer mode.
TCLK	←	-	-	-	-	0	1	-	-	Select $\phi$ T1 (0.8 $\mu$ s @fc = 10 MHz) as the input clock.
TREG1	←	0	0	1	1	0	0	1	0	Set the timer register at 40 $\mu$ s/ $\phi$ T1 = 32H.
INTEF	←	-	-	1	-	-	-	-	-	Enable INTT1.
TRUN	←	-	-	1	-	-	-	1	-	Start Timer 1.

(Note) X: Don't care      -: No change

Refer to Table 3.6 (1) for selecting the input clock:

**Table 3.6 (1) 8-bit Timer Interrupt Cycle and Input Clock**

Interrupt cycle @fc = 10MHz	Resolution	Input clock
0.8 $\mu$ s ~ 204.8 $\mu$ s	0.8 $\mu$ s	$\phi$ T1 (8/fc)
12.8 $\mu$ s ~ 3.2768ms	12.8 $\mu$ s	$\phi$ T16 (128/fc)
204.8 $\mu$ s ~ 52.4288ms	204.8 $\mu$ s	$\phi$ T256 (2048/fc)

### ② Generating pulse at 50% duty

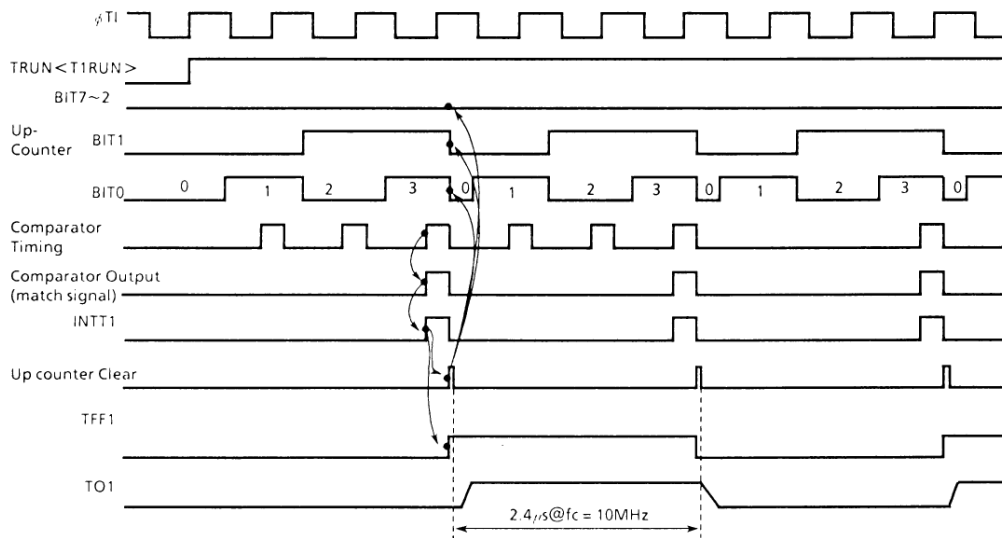
The timer flip-flop is inverted at specified intervals, and its status is output to a timer output pin TO1 (only Timers 0 and 1).

Example: To output pulse from TO1 at fc = 10MHz every 4.8 $\mu$ s, the registers should be set as follows:

This example uses Timer 1, but the same operation can be effected by using Timer 0.

	7	6	5	4	3	2	1	0		
TRUN	←	-	-	-	-	-	-	0	-	Stop Timer 1, and clear it to "0".
TMOD	←	-	-	-	-	0	0	X	X	Set the 8-bit timer mode.
TCLK	←	-	-	-	-	0	1	-	-	Select $\phi$ T1 as the input clock.
TREG1	←	0	0	0	0	0	0	1	1	Set the timer register at 4.8 $\mu$ s/ $\phi$ T1/2 = 3.
TFFCR	←	-	-	-	-	0	0	1	1	Clear TFF1 to "0", and set to invert by the match signal from Timer 1.
P3CR	←	-	-	-	1	-	-	-	-	} Select P34 as T01 pin.
P3FR	←	-	-	-	1	-	-	-	-	
TRUN	←	-	-	1	-	-	-	1	-	Start Timer 1.

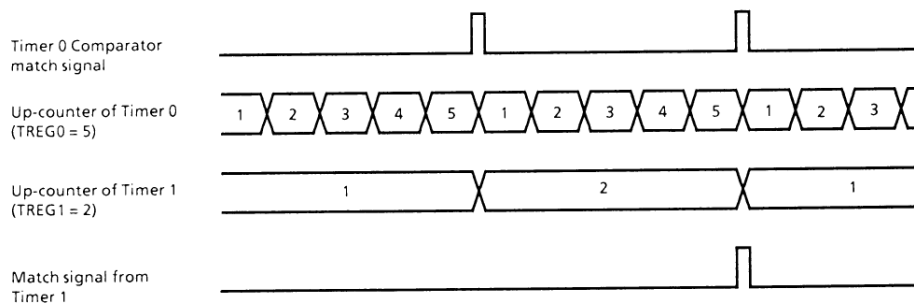
(Note) X: Don't care      -: No change



**Figure 3.6 (8). Pulse Output (50% duty) Timing Chart**

③ Making Timer 1 counting up Timer 0 match signal.

Select the 8-bit timer mode, and set the comparator output of Timer 0 as the input clock to Timer 1.



**Figure 3.6 (9)**

## ④ Output inversion by software

The timer flip-flops can be inverted by software regardless of the timer operation. Writing "10" into TFFCR <TFF1C1, 0> inverts TFF1.

## ⑤ Initial setting of Timer Flip-flops

The timer flip-flops can be initialized to either "0" or "1" without regard to the timer operation.

TFF1 is initialized to "0" by writing "00", and to "1" by writing "01" into TFFCR <TFF1C1,0>.

Note: Data from the timer flip-flops and timer registers cannot be read.

## (2) 16-bit Timer Mode

A pair of Timer 0 and Timer 1 or Timer 2 and Timer 3 can be used as one 16-bit interval timer.

Timers 2 and 3 do not provide output function, but timer operations are the same with Timer 0 and 1. The operation of a timer pair Timer 0 and Timer 1 is discussed. Cascade connection of Timer 0 and Timer 1 to use them as a 16-bit interval timer requires to set the <T10M1,0> of the mode register TMOD to "0,1".

By selecting the 16-bit timer mode, the overflow output of Timer 0 is automatically selected as the input clock to Timer 1, regardless of the set value of the clock control register TCLK. The input clock to Timer 0 is selected by TCLK. Table 3.6 (2) shows the relationship between timer (interrupt) cycle and input clock selection.

**Table 3.6 (2) 16-bit Timer (Interrupt) Cycle and Input Clock**

Timer (interrupt) cycle @fc = 10MHz	Resolution	Input clock to Timer 0
0.8 $\mu$ s ~ 52.43ms	0.8 $\mu$ s	$\phi$ T1 (8/fc)
12.8 $\mu$ s ~ 838.86ms	12.8 $\mu$ s	$\phi$ T16 (128/fc)
204.8 $\mu$ s ~ 13.425s	204.8 $\mu$ s	$\phi$ T256 (256/fc)

The lower 8 bits of the timer (interrupt) cycle is set by TREG0 and the upper eight bits of that is set by TREG1. Note that TREG0 must be always set first (Writing data into TREG0 disables the comparator temporarily, which is restarted by writing data into TREG1).

Example: To generate interrupts INTT1 at fc = 8MHz every 1 second, the timer registers TREG0 and TREG1 should be set as follows:

As  $\phi$ T16 (= 16 $\mu$ s @ 8MHz) is selected as the input clock, 1 sec/16 $\mu$ s = 62500 = F424H

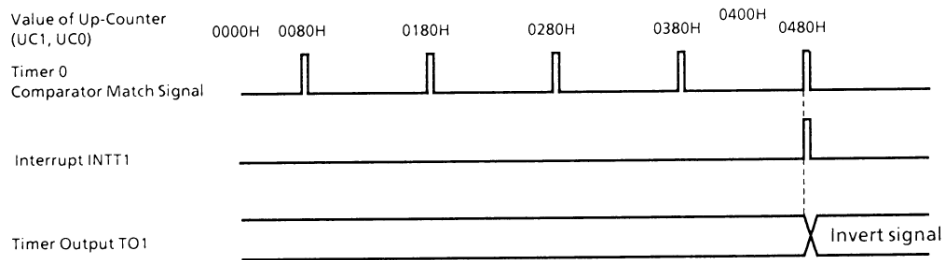
Therefore, TREG1 = F4H  
TREG0 = 24H

The match signal is generated by Timer 0 comparator each time the up-counter UC0 matches TREG0. In this case, the up-counter UC0 is not cleared, but the interrupt INTT0 is generated.

Timer 1 comparator also generates the match signal each time the up-counter UC1 match TREG1. When the match signal is generated simultaneously from comparators of Timer 0 and Timer 1, the up-counters UC0 and UC1 are cleared to "0", and the interrupt INTT1 is generated. If the timer flip-flop inversion is enabled by the Timer Flip-flop control register, the timer flip-flop TFF1 is inverted at the same time.

	Timer 0			Timer 1		
	INTT0	TO1	match	INTT1	TO1	match
16-bit Timer Mode (Count-up Timer 1 by overflow of Timer 0)	Interrupt is generated	Can't output (Can't output the matching with TREG0)	TREG0 (Continue counting when match)	Interrupt is generated	Can output *Can output the matching with both TREG0 and TREG1)	$TREG1 * 2^8 + TREG0$ (16-bit) (Cleared by matching with both registers.)
8-bit Timer Mode (Count-up Timer 1 by matching of Timer 0)	Interrupt is generated	Can output (Timer 0 or Timer 1)	TREG0 (Clear when Match)	Interrupt is generated	Can output (Timer 0 or Timer 1)	$TREG1 * TREG0$ (Multiplied Value) (Cleared by matching)

Example: Given TREG1 = 04H and TREG0 = 80H,



**Figure 3.6 (10)**

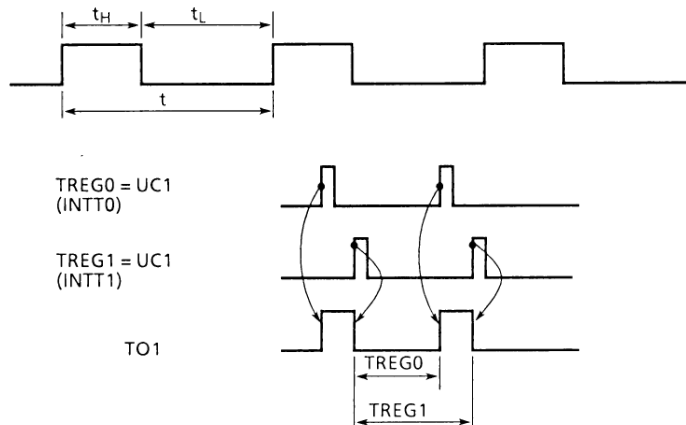
(3) 8-bit PPG (programmable pulse generation) output mode

Pulse can be generated at any frequency and duty

rate by Timer 1 or Timer 3. The output pulse may be either low-or high-active.

In this mode, Timers 0 cannot be used.

Pulse is output to TO1 (shared with P37).



In this mode, programmable pulse is generated by the inversion of the timer output put each time the 8-bit up-counter 1 (UC1) matches the timer register TREG0 or TREG1.

Note that the set value of TREG0 must be smaller than that of TREG1.

In this mode, the up-counter UC0 of Timer 0 cannot be used (Set TRUN <T0RUN> = 1, and count the Timer 0). The PPG mode is shown in Figure 3.6 (11).

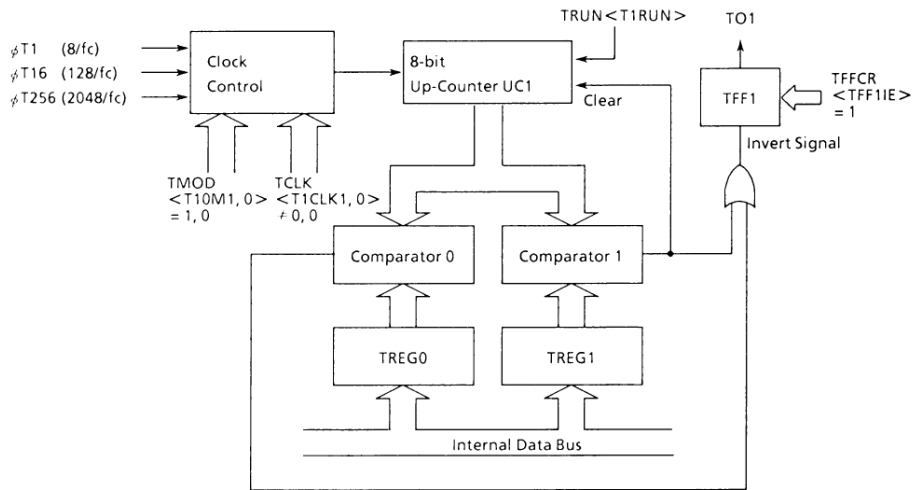
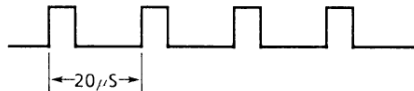


Figure 3.6 (11). Block Diagram of 8-bit PPG Mode

Example: Generate pulse at 50kHz and 1/4 duty rate

(@fc = 8MHz)



# TMP90C400/401

- Determine the set value of the timer registers.  
 To obtain the frequency of 50kHz,  
 Pulse cycle  $t = 1/50\text{kHz} = 20\mu\text{s}$   
 When  $\phi T1 = 1\mu\text{s}$  (@ 8MHz),  
 $20\mu\text{s} \div 1\mu\text{s} = 20$

Consequently, the timer register 1 (TREG1) should be set to  $20 = 14\text{H}$ .  
 Given a 1/4 duty,  $t \times 1/4 = 20 \times 1/4 = 5\mu\text{s}$   
 $5\mu\text{s} \div 1\mu\text{s} = 5$   
 As a result, the timer register 0 (TREG0) should be set to  $5 = 05\text{H}$ .

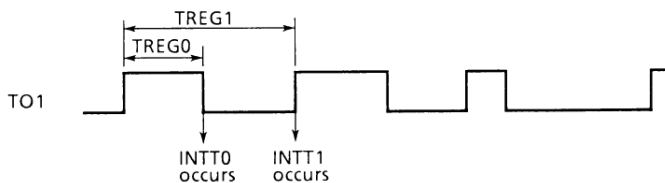
TRUN	←	- - - - - 0 0	Stop Timer 0 and Timer 1, and clear them to "0".
TCLK	←	- - - - 0 1 x x	Select $\phi T1$ as the input clock.
TMOD	←	- - - - 1 0 x x	Set 8-bit PPG mode.
TFFCR	←	- - - - 0 1 1 1	Set the output "H", and enable the inversion by Timer 1.
<div style="display: flex; align-items: center; margin-left: 100px;"> <div style="border-left: 1px solid black; border-bottom: 1px solid black; width: 100px; height: 100px; margin-right: 10px;"></div> <div style="margin-left: 10px;"> <p>Writing "00" provides negative logic pulse</p> </div> </div>			
TREG0	←	0 0 0 0 0 1 0 1	Write "5H".
TREG1	←	0 0 0 1 0 1 0 0	Write "14H".
SMMOD	←	- - - - x x 0 1	} Select P60 as the T01 pin.
P67CR	←	- - - - - - 1	
TRUN	←	- - 1 - - - 1 1	Start Timer 1.

(Note) X: Don't care      -: No change

## Precautions for PPG Output

By rewriting the content of the TREG (timer register), it is possible to make TMP90C400 output PPG. However, be careful, since the timing to rewrite TREG differs depending on the pulse width of PPG to be set. This problem is explained below by an example.

Example: To output PPG through 8 bit timers 0 and 1  
 TREG0: Pulse width  
 TREG1: Cycle



The pulse width is normally changed by the interrupt (INTT1) process routine in each cycle. However, when the pulse width to be set (the value to be written in TREG0) is small, trouble may occur, in that the timer counter exceeds the value of TREG0 before the interrupt process routine is set. Therefore, it is recommended to make the following decisions in INTT0 and INTT1 interrupt processes.

INTT0 process routine: The value of TREG0 is rewritten only when the value to be written in TREG0 is smaller than the current value of TREG0.

INTT1 process routine: On the contrary to INTT0, TREG0 is written only when the value to be written in TREG0 is larger than the current value of TREG0.

TMP90C400 cannot read the content of TREG, so it is necessary to buffer the content of TREG in a RAM (or the like) for making the above judgement.

(4) 8-bit PWM (pulse width modulation) mode

This mode is only available for Timer1, and generates 8-bit resolution PWM.

PWM is output to TO1 pin (shared with P34).

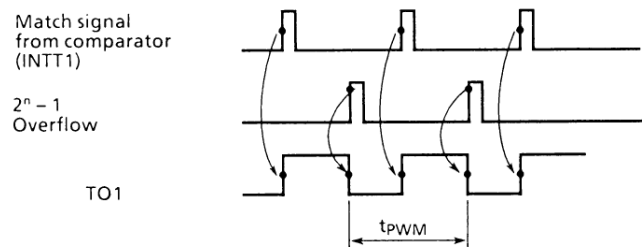
Timer 0 can be used as 8-bit timers.

The inversion of the timer output occurs when the up-counter (UC1) matches the set value of the timer register TREG1, as well as when an overflow of  $2^n - 1$  ( $n = 6, 7$  or  $8$  selected by TMOD <PWM01, 00>) occurred at the counter. The up-counter UC1 is cleared by the occurrence of an overflow of  $2^n - 1$ .

The following condition must be obtained when this PWM mode is used:

(Set value of timer register) < (set overflow value of  $2^n - 1$  counter)

(Set value of timer register)  $\neq 0$



The PWM mode is shown in Figure 3.6 (12).

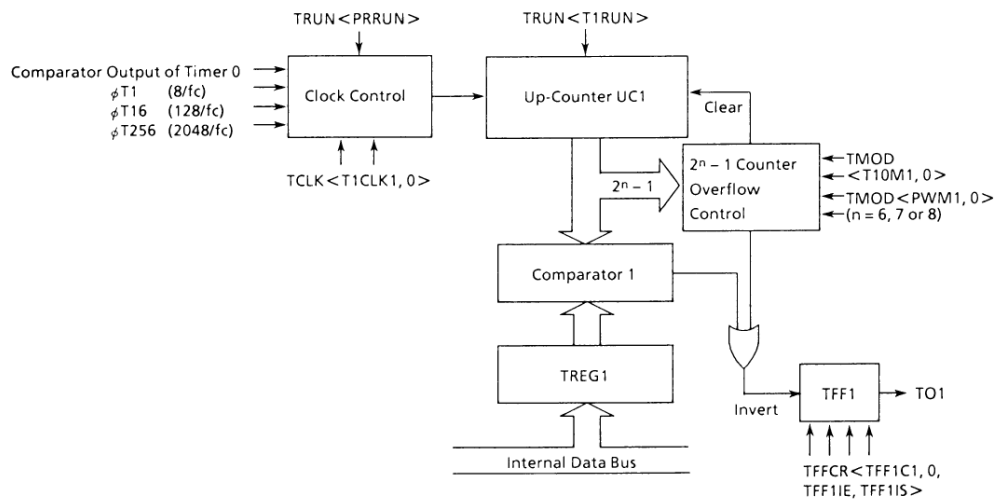
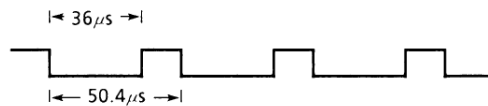


Figure 3.6 (12). Block Diagram of 8-bit PWM Mode

Example: Generate the following PWM to the TO1 pin (P34) at  $f_c = 10\text{MHz}$ .



Assuming the PWM cycle is  $50.4\ \mu\text{s}$  when  $\phi T1 = 0.8\ \mu\text{s}$  and  $@f_c = 10\text{MHz}$ ,  $50.4\ \mu\text{s}/0.8\ \mu\text{s} = 63 = 2^6 - 1$   
Consequently,  $n$  should be set at 6 (TMOD1, 0 = 01).

Given the "L" level period of  $36\ \mu\text{s}$ , setting  $\phi T1 = 0.8\ \mu\text{s}$  results:  $36\ \mu\text{s}/0.8\ \mu\text{s} = 45 = 2\text{DH}$ .  
As a result, TREG1 should be set at 2DH.



TRUN ← - - - - - 0 -	Stop Timer 1.
TCLK ← - - - - 0 1 - -	Select $\phi$ T1 as the input clock.
TMOD ← - - - - 1 1 0 1	Set the $2^6 - 1$ cycle in the PWM mode.
TFFCR ← - - - - 0 0 1 1	Set the initial output to 0 ("L" level).
TREG1 ← 0 0 1 0 1 1 0 1	Write "2DH".
P3CR ← - - - 1 - - - -	} Select P34 as T01 pin.
P3FR ← - - - 1 - - - -	
TRUN ← - - 1 - - - 1 -	Start Timer 1.

(Note) ×: Don't care      -: No change

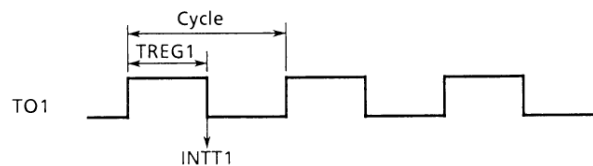
**Table 3.6 (3) PWM Cycle and Selection of  $2^n - 1$  counter**

	Expression	PWM cycle (@fc = 10MHz)		
		$\sigma$ T1 (8/fc)	$\sigma$ T16 (128/fc)	$\sigma$ T256 (2048/fc)
$2^6 - 1$	$(2^6 - 1) \times \sigma Tn$	50.4 $\mu$ s	806.4 $\mu$ s	12.9ms
$2^7 - 1$	$(2^7 - 1) \times \sigma Tn$	101.6 $\mu$ s	1625.6 $\mu$ s	26.0ms
$2^8 - 1$	$(2^8 - 1) \times \sigma Tn$	204.0 $\mu$ s	3264.0 $\mu$ s	52.2ms

### Precautions for PWM output

TMP90C400 can output PWM by the 8-bit timer. However, changing the pulse width of PWM requires special care. This problem is explained by the following example.

Example: To output PWM by 8-bit timer  
TREG1: Pulse width  
Cycle: Fixed ( $2^6 - 1$ ,  $2^7 - 1$ ,  $2^8 - 1$ )



In the PWM mode, INTT1 occurs at the coincidence with TREG1. However, the pulse width cannot be changed directly using the interrupt. (Depending on the value of TREG1 to be set, coincidence with TREG1 may be detected again in a single cycle, inverting the timer output.)

To eliminate this problem in changing the pulse width, it is effective to halt the timer with the INTT1 process, modify the value of TREG1, set the timer output to "1", and restart the timer. In the mean time, the output waveform loses shape when the pulse width is changed. This method is valid for a system that allows a deformed output waveform.

**3.6.2 8-bit Timer/Event Counter**

## (1) Event counter mode

Timer 2 is an 8-bit timer/event counter. It functions as not only as the 8-bit timer which is explained previously, but also as the counter.

Timer 2 turns to the event counter mode by setting the input clock of Timer 2 as the external counter (T12). Therefore, Timer 2 can be used as an 8-bit counter, and Timer 3 and an 8-bit timer. Timers 2 and 3 turn to a 16-bit counter by connecting to a cascade. The counter counts up at the rising edge of the T12 counter input. The T12 pin is also used for P33 and has the zero-cross detection function. To T12 pin is specified by setting  $TCLK<T2CLK>$  to "0, 0".

	MSB		LSB						
	7	6	5	4	3	2	1	0	
TRUN	←	-	-	-	-	0	-	-	Stop Timer 2.
TMOD	←	0	0	X	X	-	-	-	Set a timer to an 8-bit timer/counter mode
P3CR	←	-	-	-	-	0	-	-	Set P33 to an input mode * = 0: T12 is square wave * = 1: T12 is sigh wave (zero-cross)
P3FR	←	-	-	-	-	*	-	-	
INTEF	←	-	-	-	1	-	-	-	Enable INTT2
TCLK	←	-	-	0	0	-	-	-	Select the input clock of Timer 2 as the counter T12
TREG2	←	*	*	*	*	*	*	*	Set the number of counts
TRUN	←	-	-	1	-	-	1	-	Start Timer 2

(Note): Set a prescaler to "RUN" in an event counter mode.  
 X : Don't care      - : No change

## (2) Software counter latch

In an event counter mode, the up-counter value can be written to a counter latch register by the current running software. The present up-counter value is written to counter register latch TREG2 or TREG3 for

every setting of register TFFCR <LATCH2> or <LATCH3> containing "0". A prescaler should be set in "RUN" mode by setting register + <PRRUN> to "1".

Example: To latch the counter value every 40μs at  $f_c = 10\text{MHz}$ , the registers should be set as follows:

	MSB	LSB								
	7	6	5	4	3	2	1	0		
TRUN	←	-	-	-	-	0	0	-	Stop Timers 1 and 2, and clear to "0"	
TMOD	←	0	0	X	X	0	0	-	Set timer 1 to an 8-bit timer mode, and Timer 2 to an 8-bit timer/counter mode	
P3CR	←	-	-	-	0	-	-	-	Set P33 to an input mode	
TCLK	←	-	-	0	0	0	1	-	Select $\Phi T1$ ( $0.8\mu\text{s}$ at $f_c = 10\text{MHz}$ ) as the input clock of Timer 1, and select counter input T12 as input clock of Timer 2	
TREG1	←	0	0	1	1	0	0	1	0	Set the timer register 1 at $40\mu\text{s}/T1 = 50$
TREG2	←	*	*	*	*	*	*	*	*	Set the timer register 2
INTEF	←	-	-	1	-	-	-	-	Enable INTT1	
TRUN	←	-	-	1	-	-	1	1	-	Count Timers 1 and 2
Setting of INTT1 :										
TFFCR	←	-	0	-	-	-	-	-	-	Latch the counter value.
(Note): X : Don't care      - : No change										

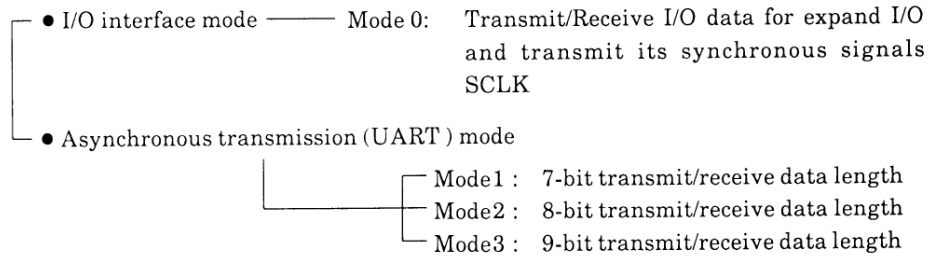
The latched counter value can be read by reading Timer register 2.

**3.7 Serial Channel**

The TMP90C400 incorporates a serial I/O channel for full

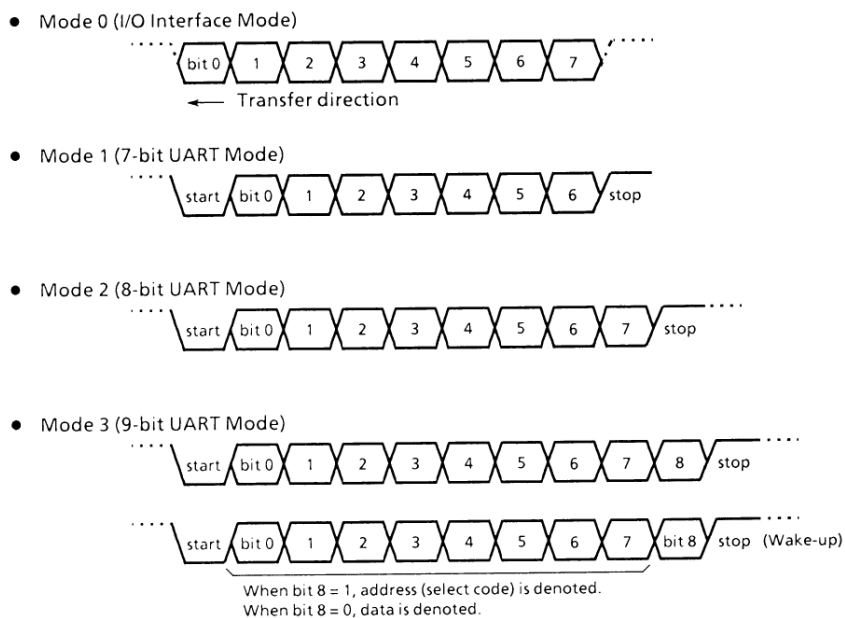
duplex asynchronous transmission (UART) and I/O expansion.

The serial channel has the following operating modes:



The mode 3 accommodates a wake-up function to start the slave controllers in a controller serial link (multi-controller system).

Figure 3.7 (1) shows the data format (1-frame data) in each mode.



**Figure 3.7 (1). Data Formats**

Data received and transmitted are stored temporarily into separate buffer registers to allow independent transmission and receiving (full-duplex).

In the I/O interface mode, however, the data transfer is half-duplex due to the single SCLK (serial clock) pin is used for transmission and receiving.

Serial channel pins (RxD, SCL, TxD pins) are shared among P35, P36 and P37, respectively. The pin function is selected by P3CR and P3FR registers. P35 can be used as RxD pin by setting P3CR <P35> to "0". Also P36 and P37 can be used as SCLK and TxD pins by setting P3CR6 <P37C, P36C> to 11 and P3FR <SCLK, TXDC> to 11, respectively.

The receiving buffer register has a double-buffer structure

to prevent overruns. The one buffer receives the next frame data while the other buffer stores the received data.

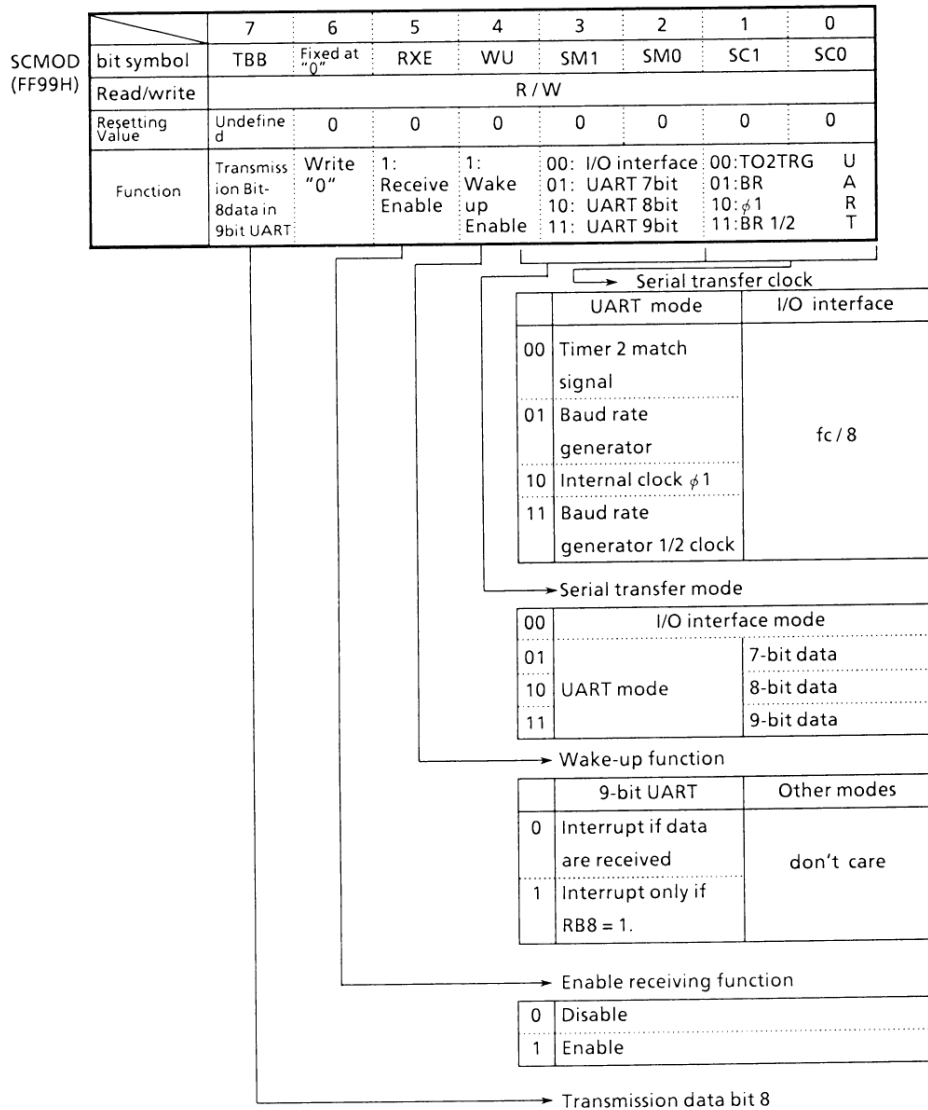
In the UART mode, a check function is added not to start the receiving operation by error start bits due to noise. The channel starts receiving data only when the start bit is detected to be normal at least twice in three samplings.

When an request is issued to the CPU to transmit data after the transmitting buffer becomes empty, or to read data after the receiving buffer stores data, the interrupt INTTX or INTRX occurs respectively, In receiving data, the flag SCCR <OERR, FERR> is set when an overrun error, or framing error occurs accordingly.

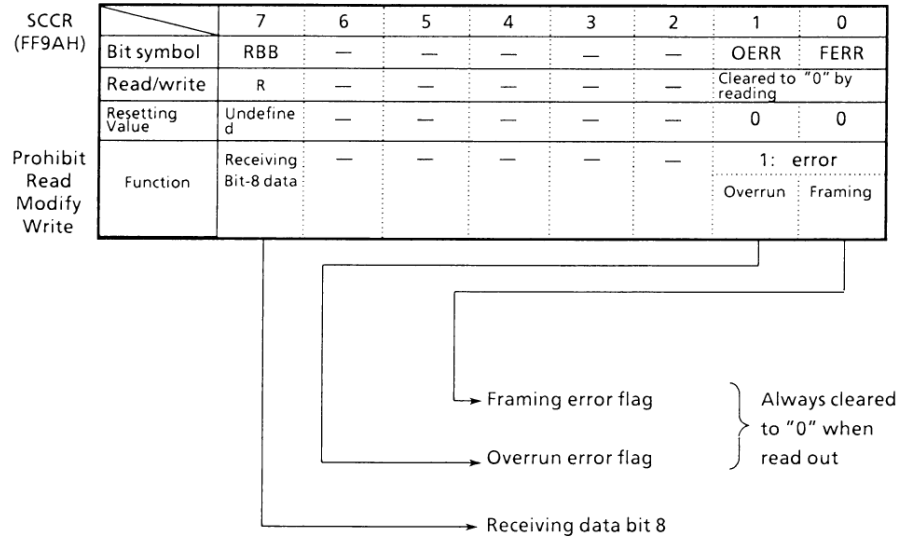
**3.7.1 Control Registers**

The serial channel is controlled by five control registers

(SCHMOD, SCCR, TRUN, P3CR, and P3FR). The received/transmitted data are stored into SCBUF.



**Figure 3.7 (2). Serial Channel Mode Register**



(Note) Since all error flags are cleared after readout, avoid testing for only one bit using a bit-testing instruction.

Figure 3.7 (3). Serial Channel Control Register

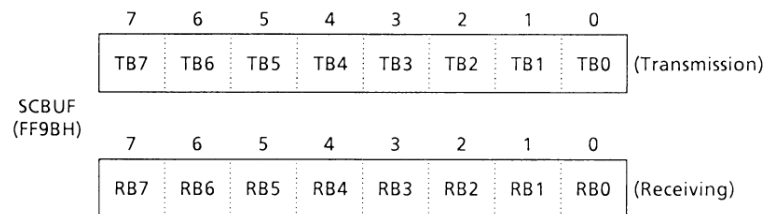
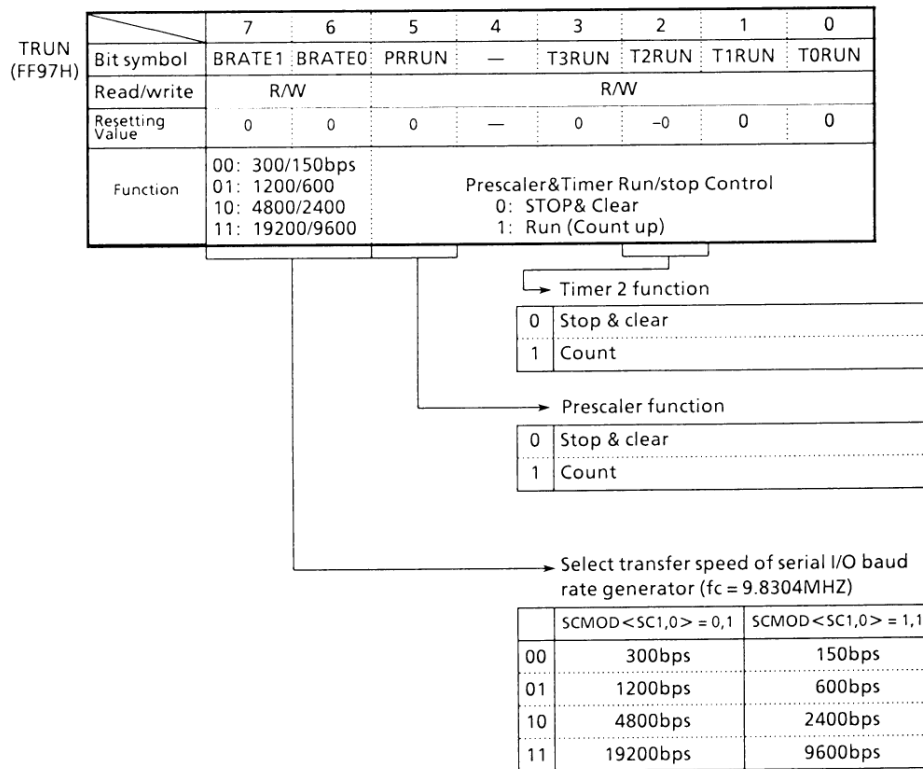


Figure 3.7 (4). Serial Transmission/Receiving Buffer Register



**Figure 3.7 (5). Timer/Serial Channel Operation Control Register**



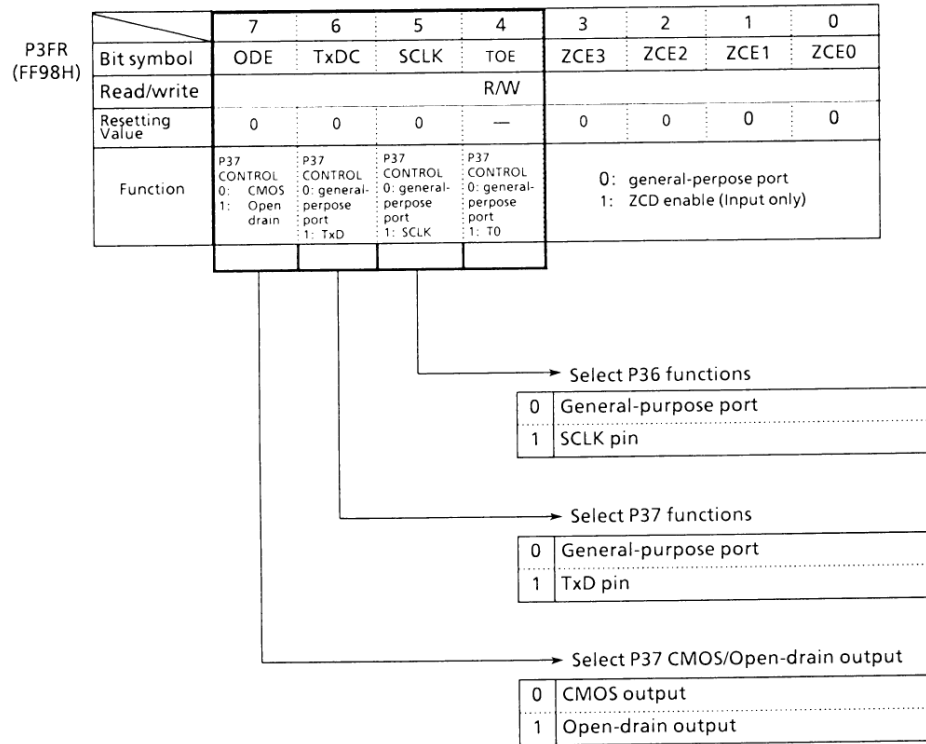
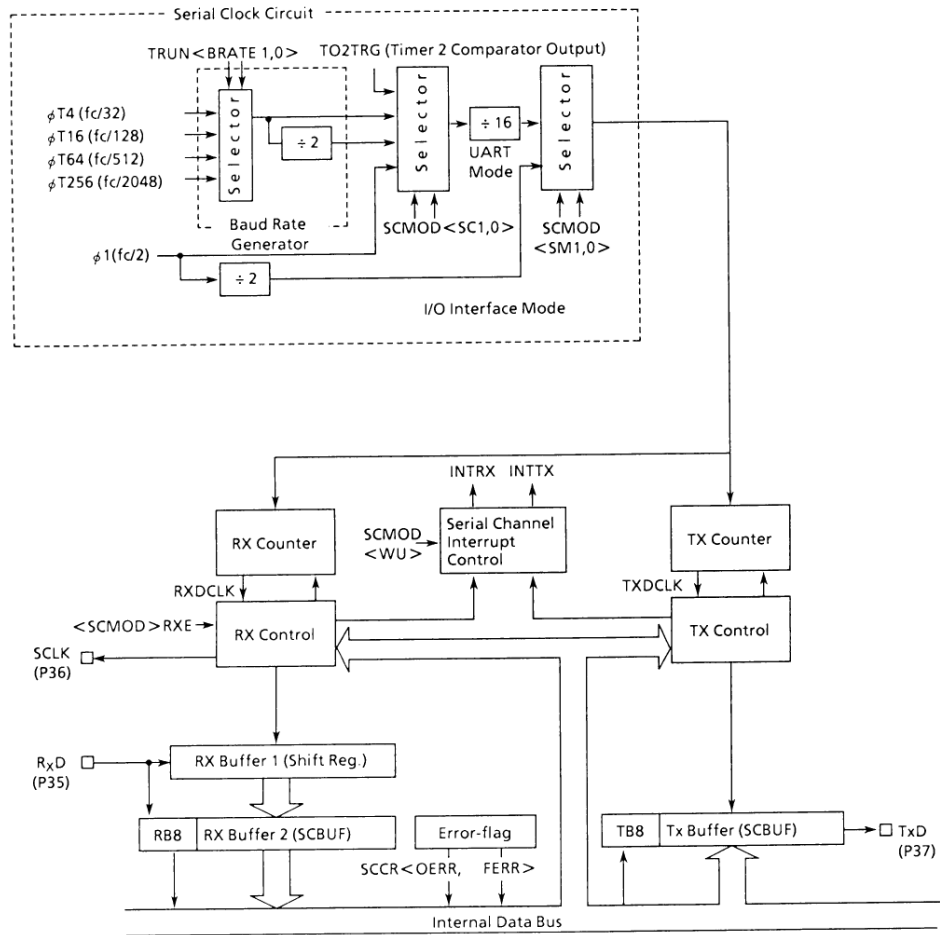


Figure 3.7 (6). Port 3 Function Register

### 3.7.2 Architecture

Figure 3.7 (8) is a block diagram of the serial channel.



**Figure 3.7 (8). Block Diagram of Serial Channel**

## ① Baud-rate generator

The baud-rate generator comprises a circuit that generates a clock pulse to determine the transfer speed for transmission/receiving in the asynchronous communication (UART) mode.

The input clock to the baud-rate generator  $\phi T4$  ( $fc/32$ ),  $\phi T16$  ( $fc/128$ ),  $\phi T64$  ( $fc/512$ ) or  $\phi T256$  ( $fc/2048$ ) is

generated by the 9-bit prescaler. One of these input clocks are selected by the timer/serial channel control register TRUN <BRATE1, 0>.

Also, either no frequency division or 1/2 division can be selected by the serial channel mode register SCMOD <SC1, 0>.

Table 3.7 (1) shows the baud-rate when  $fc = 9.8304\text{MHz}$ .

**Table 3.7 (1) Baud Rate Selection (1) Unit [bps]**

<BRATE 1, 0>	Input Clock	No Division (SC1, 0 = 01)	1/2 Division (SC1, 0 = 11)
00	$\phi T256$ ( $fc/2048$ )	300	150
01	$\phi T64$ ( $fc/512$ )	1200	600
10	$\phi T16$ ( $fc/128$ )	4800	2400
11	$\phi T4$ ( $fc/32$ )	19200	9600

@ $fc = 9.8304\text{MHz}$

**Table 3.7 (2) Baud Rate Selection (2)  
(When use Timer 2 with  $\phi T1$ ) Unit [Kbps]**

TREG2/ $fc$	12.288 MHz	12 MHz	9.8304 MHz	8 MHz	6.144 MHz
01H	96	–	76.8	62.5	48
02H	48	–	38.4	31.25	24
03H	32	31.25	–	–	16
04H	24	–	19.2	–	12
05H	19.2	–	–	–	9.6
08H	12	–	9.6	–	6
0AH	9.6	–	–	–	4.8
10H	6	–	4.8	–	3
14H	4.8	–	–	–	2.4

$$\text{Baud Rate} = \frac{1}{\text{TREG2}} \times \frac{1}{16} \times \text{Input clock of Timer2}$$

Input clock of Timer 2

$$\phi T1 = fc/8$$

$$\phi T16 = fc/128$$

$$\phi T256 = fc/2048$$

## ② Serial clock generating circuit

This circuit generates the basic transmit/ receive clock.

## 1) I/O interface mode

It generates a clock at a 1/8 frequency (1.25Mbit/s at 10MHz) of the system clock (fc). This clock is output from the SCLK pin (also used as P36).

## 2) Asynchronous communication (UART) mode

A basic clock (SIOCLK) is generated based on the above baud rate generator clock, the internal clock  $\sigma 1$  ( $f_c/2$ ) (SIOCLK = 5MHz, Transfer speed = 312.5Kb.p.s at 10MHz). or the match signal from Timer 2, as selected by SCMOD <SC1, 0> register.

## ③ Receiving counter

The receiving counter is a 4-bit binary counter used in the asynchronous communication (UART) mode and is counted by using SIOCLK. 16 pulse of SIOCLK is used for receiving 1-bit data. The data are sampled three times a 7th, 8th and 9th pulses and evaluated by the rule of majority. for example, if data sampled at the 7th, 8th and 9th clock are "1", "0" and "1", the received data is evaluated as "1". The sampled data "0", "0" and "1" is evaluated that the received data is "0".

## ④ Receiving control

## 1) I/O interface mode

The RxD signal is sampled on the rising edge of the shift clock which is output to the SCLK pin.

## 2) Asynchronous communication (UART) mode

The receiving control features a circuit for detecting the start bit by the rule of majority. When two or more "0" are detected during 3 samples, it is recognized as normal start bit and the receiving operation starts. Receiving data being received are also evaluated by the majority logic while receiving data.

## ⑤ Receive buffer

The receive buffer has a double-buffer structure to prevent overruns. Receive data are stored into the receive buffer 1 (shift register type) for each 1 bit. When 7 or 8 bits data are stored in the receive buffer 1, the stored data is transferred to the receive buffer 2 (SCBUF), and the interrupt INTRX occurs at the same time. The CPU reads out the receive buffer 2 (SCBUF). Data can be stored into the receive buffer 1 before the CPU reads out the receive buffer 2 (SCBUF).

Note, however, that an overrun occurs unless the CPU reads out the receive buffer 2 (SCBUF) before the receive buffer 1 receiving all bits of the next data.

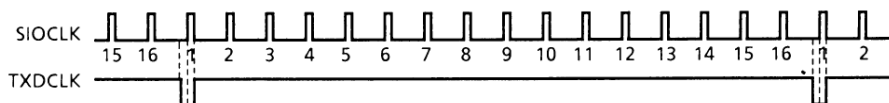
When an overrun occurs, the data in the receive buffer 2 and SCCR <RB8> are not lost, however, that in the receive buffer 1 are lost.

SCCR <RB8> stores the MSB in the 9-bit UART mode.

In the 9-bit UART mode, setting SCMOD <WU> to "1" enables the wake-up function of the slave controllers, and the interrupt INTRX occurs only if SCCR <RB8> = "1".

## ⑥ Transmission counter

This is a 4-bit binary counter used in the asynchronous communication (UART) mode. Like the receiving counter, it counts based on SIOCLK to generate a transmission clock TXDCLK for every 16 counts.



## ⑦ Transmission control

## 1) I/O interface mode

Data in the transmission buffer are output to the TxD pin bit by bit at the rising edge of the shift clock output from the SCLK pin.

## 2) Asynchronous communication (UART) mode

When the CPU have written data into the transmission buffer, transmission is started with the next rising edge of TxDCLK, and a transmission shift clock TxDSFT is generated.

## ⑧ Transmission buffer

The transmission buffer SCBUF shifts out the data written by the CPU from the LSB as based on the shift clock TXDSFT (Same period as TCDCLK) generated by the transmission control unit. When all bits are shifted out, the transmission buffer becomes empty, generating the interrupt INTTX.

## ⑨ Signal Generation Timing

## 1) UART mode

**Receiving**


mode	9 bit	8 bit, 7 bit
Interrupt timing	Center of last bit (Bit 8)	Center of stop bit
Framing error timing	Center of stop bit	↑
Over-run error timing	Center of last bit (Bit 8)	↑

Note: The occurrence of a framing error is delayed until after interruption. Therefore, to check for framing error during interrupt operation, an addition operation, such as waiting for 1-bit time, becomes necessary.

**Transmitting**

mode	9 bit	8 bit, 7 bit
Interrupt timing	Just before the stop bit	←

## 2) I/O expansion mode

Interrupt timing of receiving	Just after the last SCLK rising 
Interrupt timing of transmitting	↑

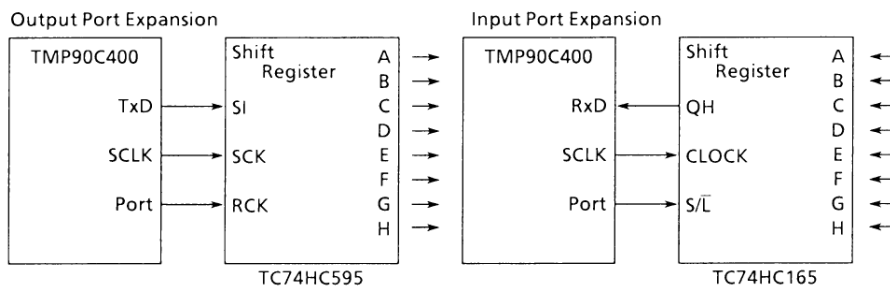
### 3.7.3 Operation

(1) Mode 0 (I/O Interface Mode)

This mode is used to increase the number of I/O pins

of the TMP90C400.

The TMP90C400 supplies the transmit/receive data and a synchronous clock (SCLK) to an external shift register.

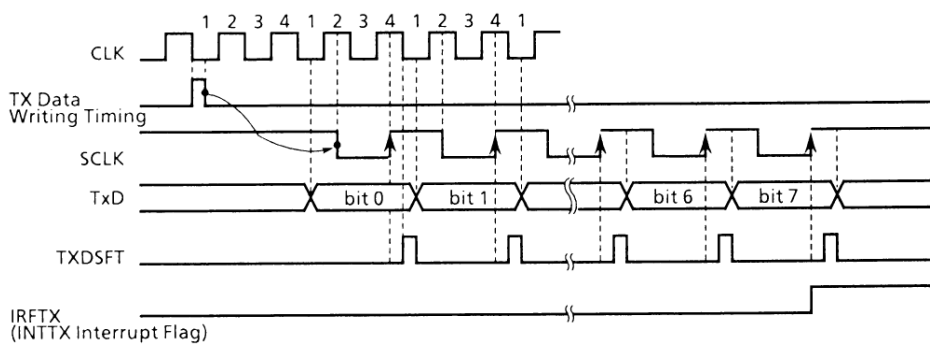


**Figure 3.7 (9). I/O Interface Mode**

① Transmission

Each timer the CPU writes data into the transmission

buffer, 8-bit data are output from TxD pin. When all data are output, IRFH<IRFTX> is set, and the interrupt INTTX occurs.



**Figure 3.7 (10). Transmitting Operation (I/O Interface Mode)**

Example: When transmitting data from P33 pin, the

control registers should be set as described below.

P3CR ← 1 1 - - - - -	} Select P36 as the SCLK pin, and P37 as the TXD pin.	
P3FR ← - 1 1 - - - - -		
SCMOD ← X X X X 0 0 X X		Set I/O interface Mode.
INTEF ← - - - - - 1		Enable INTTX interrupt.
SCBUF ← * * * * * * *		Set data for transmission

(Note) X ; don't care    - ; No change

② Receiving

Each time the CPU reads the receive data and clears the receive interrupt flag IRFH <IRFRX>, the next data are shifted into the receive buffer 1. When 8-bit data

are received, the data are transferred to the receive buffer 2 (SCBUF), which sets <IRFRX> and generates interrupt INTRX.

For receiving data, the receiving enable state is previously set SCMOD <RXE> = 1.

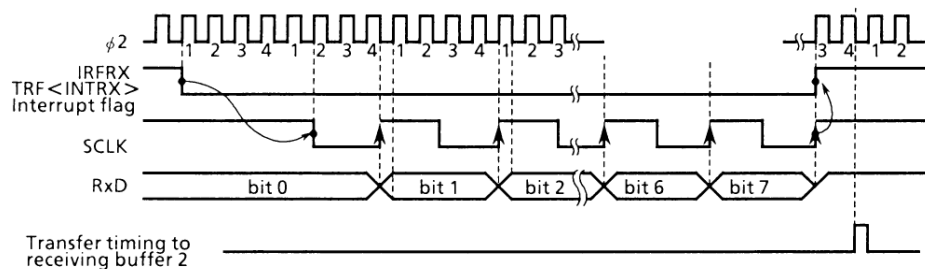


Figure 3.7 (11). Receiving Operation (I/O Interface Mode)

Example: When receiving from P35 pin, the control

registers should be set as described below.

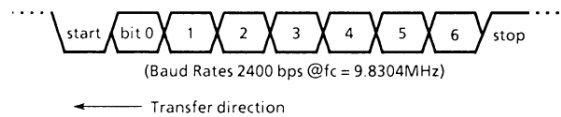
P3CR	←	-	-	1	0	-	-	-	-	-	-		} Select P36 as the SCLK pin, and P35 as the RXD pin.	
P3FR	←	-	-	1	-	-	-	-	-	-	-			
SCMOD	←	X	X	X	X	0	0	X	X					Set I/O interface Mode.
INTE	←	-	-	-	-	-	-	1	-					Enable INTRX interrupt.
SCMOD	←	-	-	1	-	-	-	-	-					Set RXE to "1".

(Note) X ; don't care      - ; No change

(2) Mode 1 (7-bit UART Mode)

Example: When transmitting data with the following format, the control registers should be set as described below.

The 7-bit UART mode is selected by setting the serial channel mode register SCMOD <SM1, 0> to "01".



P3CR	←	1	-	-	-	-	-	-	-		} Select P37 as the TxD pin.	
P3FR	←	-	1	-	-	-	-	-	-			
SCMOD	←	X	0	-	X	0	1	1	1			Set the transfer speed at 2,400 bps in the 7-bit UART mode.
TRUN	←	1	0	1	-	-	-	-	-			Enable INTTX interrupt.
SCBUF	←	*	*	*	*	*	*	*	*			Set data for transmission

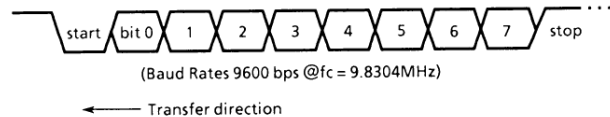
(Note) X ; Don't care      - ; No change



## (3) Mode 2 (8-bit UART mode)

The 8-bit UART mode is selected by setting SCMOD <SM1, 0> to "1, 0".

Example: When receiving data with the following format, the control registers should be set as described below.



## Main setting:

P3CR	← - - 0 - - - -	Select P35 as the RxD pin.
TRUN	← 1 1 1 - - - -	Set the transfer speed at 9,600 bps in the 8-bit UART mode.
SCMOD	← - 0 1 X 1 0 1 1	Enable INTTX interrupt.
INTEF	← - - - - - 1 -	

## INTRX processing:

Acc	← SCCR Δ 00000011	Check errors.
	if Acc ≠ 0 then error	
Acc	← SCBUF	Read out the received data.

(Note) X: Don't care    -: No change

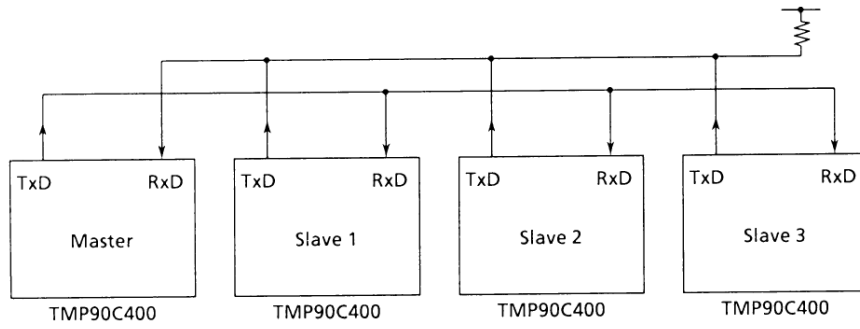
## (4) Mode 3 (9-bit UART Mode)

The 9-bit UART mode is selected by setting SCMOD <SM1, 0> = "11".

The MSB (9th bit) is written into SCMOD <TB8> for transmission, and into SCCR <RB8> for receiving. Writing into or reading from the buffer must begin with the MSB (9th bit) followed by SCBUF.

Wake-up function

In the 9-bit UART mode, setting SCMOD <WU> to "1" allows the wake-up operation as the slave controllers. The interrupt INTRX occurs only when SCCR <RB8> = 1.

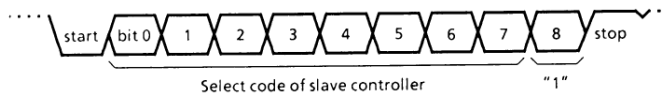


Note: for the wake-up operation, P33 should be always selected as the TxD pin of the slave controllers, and put in the open drain output mode.

**Figure 3.7 (12). Serial Link Using Wake-up Function**

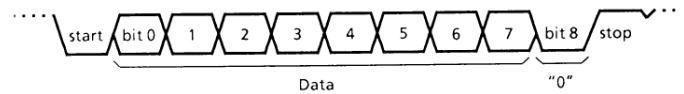
## Protocol

- ① Select the 9-bit UART mode for the master and slave controllers.
- ② Set the SCMOD <WU> bit of each slave controller to "1" to enable data receiving.
- ③ The master controller transmits 1-frame data including the 8-bit select code for the slave controllers. The MSB (8-bit) SCMOD <TB8> is set to "1".



- ④ Each slave controller receives the above frame, and clears the <WU> bit to "0" if the above select code matches its own select code.

- ⑤ The master controller transmits data to the specified slave controller (whose <WU> bit is cleared to "0") while setting the MSB (bit 8) <TB8> to "0".

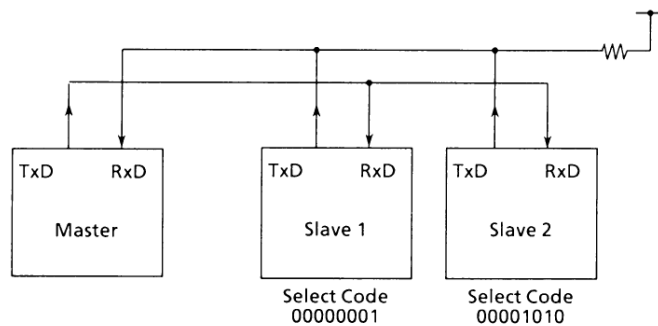


- ⑥ The slave controllers (with the SCMOD <WU> bit remaining at "1") ignore the receive data since the MSB (SCCR <RB8>) are set to "0" to disable the interrupt INTRX.

When the <WU> bit is cleared to "0", the interrupt INTRX is generated and receive data are read.

The slave controllers (WU = 0) transmits data to the master controller. It is possible that the the master controller to be indicated the end of data received by this transmit data.

Example: Link two slave controllers serially with the master controller, and use the internal clock  $\phi/2$  as the transfer clock.



● Set the master control

Main

P3CR ← 1 - 0 - - - - -	} Select P37 as TxD pin and P35 as RxD pin.	
P3FR ← 0 1 - - - - -		
INTEF ← - - - - - 1 1		Enable INTRX and INTTX.
SCMOD ← 1 0 1 0 1 1 1 0		Select $\phi 1(fc/2)$ as the transfer clock in the 9-bit UART mode.
SCBUF ← 0 0 0 0 0 0 0 1		Set the select code for the slave controller 1.

INTTX interrupt

SCMOD ← 0 - - - - -	Set SCMOD<TB8> to "0".
SCBUF ← * * * * *	Set data for transmission.

● Set the slave controller 2

Main

P3CR ← 1 - 0 - - - - -	} Select P37 as TxD pin and P35 as RxD pin.	
P3FR ← 1 1 - - - - -		
INTEF ← - - - - - 1 1		Enable INTRX and INTTX.
SCMOD ← 0 0 1 1 1 1 1 0		Set <WU> to 1 in the 9-bit UART mode (transfer clock : $\phi 1(fc/2)$ ).

(Note) X: Don't care-: No change

## 4. Electrical Characteristics

TMP90C400N/TMP90C400F/  
TMP90C401N/TMP90C401F

### 4.1 Absolute Maximum Ratings

Symbol	Parameter	Rating	Unit
$V_{CC}$	Supply voltage	-0.5 ~ +7	V
$V_{IN}$	Input voltage	-0.5 ~ $V_{CC} + 0.5$	V
$P_D$	Power dissipation ( $T_a = 85^\circ\text{C}$ )	F 500	mW
		N 600	
$T_{SOLDER}$	Soldering temperature (10s)	260	$^\circ\text{C}$
$T_{STG}$	Storage temperature	-65 ~ 150	$^\circ\text{C}$
$T_{OPR}$	Operating temperature	-40 ~ 85	$^\circ\text{C}$

### 4.2 DC Characteristics

$V_{CC} = 5V \pm 10\%$   $T_A = -40 \sim 85^\circ\text{C}$  (1 ~ 10MHz)  
 $T_A = -20 \sim 70^\circ\text{C}$  (1 ~ 12.5MHz)

Symbol	Parameter	Min	Max	Unit	Test Conditions
$V_{IL}$	Input Low Voltage (P0)	-0.3	0.8	V	—
$V_{IL1}$	P1, P2, P3,P4, P5, P6	-0.3	$0.3V_{CC}$	V	—
$V_{IL2}$	$\overline{\text{RESET}}$ , $\overline{\text{NMI}}$	-0.3	$0.25V_{CC}$	V	—
$V_{IL3}$	$\overline{\text{EA}}$	-0.3	0.3	V	—
$V_{IL4}$	X1	-0.3	$0.2V_{CC}$	V	—
$V_{IH}$	Input High Voltage (P0)	2.2	$V_{CC} + 0.3$	V	—
$V_{IH1}$	P1, P2, P3,P4, P5, P6	$0.7V_{CC}$	$V_{CC} + 0.3$	V	—
$V_{IH2}$	$\overline{\text{RESET}}$ , $\overline{\text{NMI}}$	$0.75V_{CC}$	$V_{CC} + 0.3$	V	—
$V_{IH3}$	$\overline{\text{EA}}$	$V_{CC} - 0.3$	$V_{CC} + 0.3$	V	—
$V_{IH4}$	X1	$0.8V_{CC}$	$V_{CC} + 0.3$	V	—
$V_{OL}$	Output Low Voltage	—	0.45	V	$I_{OL} = 1.6\text{mA}$
$V_{OH}$ $V_{OH1}$ $V_{OH2}$	Output High Voltage	2.4 $0.75V_{CC}$ $0.9V_{CC}$	—	V V V	$I_{OH} = -400\mu\text{A}$ $I_{OH} = -100\mu\text{A}$ $I_{OH} = -20\mu\text{A}$
$I_{DAR}$	Darlington Drive Current (8 I/O pins) (Note)	-0.1	-3.5	mA	$V_{EXT} = 1.5V$ $R_{EXT} = 1.1k\Omega$
$I_{LI}$	Input Leakage Current	0.02 (Typ)	$\pm 5$	$\mu\text{A}$	$0.0 \leq V_{in} \leq V_{CC}$
$I_{LO}$	Output Leakage Current	0.05 (Typ)	$\pm 10$	$\mu\text{A}$	$0.2 \leq V_{in} \leq V_{CC} - 0.2$
$I_{CC}$	Operating Current (RUN)	20 (Typ)	40	mA	$t_{osc} = 10\text{MHz}$ (25%Up @12.5MHz)
	Idle 1	1.5 (Typ)	5	mA	
	Idle 2	6 (Typ)	15	mA	
$I_{CC}$	STOP ( $T_A = -40 \sim 85^\circ\text{C}$ )	0.05 (Typ)	50	$\mu\text{A}$	$0.2 \leq V_{in} \leq V_{CC} - 0.2$
	STOP ( $T_A = 0 \sim 50^\circ\text{C}$ )		10	$\mu\text{A}$	
$V_{STOP}$	Power Down Voltage (@STOP)	2 RAM BACK UP	6	V	$V_{IL2} = 0.2V_{CC}$ , $V_{IH2} = 0.8V_{CC}$
$R_{RST}$	$\overline{\text{RESET}}$ Pull Up Register	50	150	$k\Omega$	—
CIO	Pin Capacitance	—	10	pF	testfreq = 1MHz
$V_{TH}$	Schmitt width $\overline{\text{RESET}}$ , $\overline{\text{NMI}}$	0.4	1.0 (Typ)	V	—

Note:  $I_{DAR}$  is guaranteed for a total of up to 8 optional ports.

## 4.3 AC Characteristics

$V_{CC} = 5V \pm 10\%$  TA = -40 ~ 85°C (1 ~ 10MHz)  
 CL = 50pF TA = -20 ~ 70°C (1 ~ 12.5MHz)

Symbol	Parameter	Variable		10MHz Clock		12.5MHz Clock		Unit
		Min	Max	Min	Max	Min	Max	
t <sub>OSC</sub>	Oscillation cycle (= x)	80	1000	100	–	80	–	ns
t <sub>CYC</sub>	CLK Period	4x	4x	400	–	320	–	ns
t <sub>WH</sub>	CLK High width	2x - 40	–	160	–	120	–	ns
t <sub>WL</sub>	CLK Low width	2x - 40	–	160	–	120	–	ns
t <sub>AL</sub>	A0 ~ 7 effective address → ALE fall	0.5x - 15	–	35	–	25	–	ns
t <sub>LA</sub>	ALE fall → A0 ~ 7 hold	0.5x - 15	–	35	–	25	–	ns
t <sub>LL</sub>	ALE Pulse width	x - 40	–	60	–	40	–	ns
t <sub>LC</sub>	ALE fall $\overline{RD}/\overline{WR}$ fall	0.5x - 30	–	20	–	10	–	ns
t <sub>CL</sub>	$\overline{RD}/\overline{WR}$ → ALE rise	0.5x - 20	–	30	–	20	–	ns
t <sub>ACL</sub>	A0 ~ 7 effective address → $\overline{RD}/\overline{WR}$ fall	x - 25	–	75	–	55	–	ns
t <sub>ACH</sub>	Upper effective address → $\overline{RD}/\overline{WR}$ fall	1.5x - 50	–	100	–	70	–	ns
t <sub>CA</sub>	$\overline{RD}/\overline{WR}$ fall → Upper address hold	0.5x - 20	–	30	–	20	–	ns
t <sub>ADL</sub>	A0 ~ 7 effective address → Effective data input	–	3.0x - 35	–	265	–	205	ns
t <sub>ADH</sub>	Upper effective address → Effective data input	–	3.5x - 55	–	295	–	225	ns
t <sub>RD</sub>	$\overline{RD}$ fall → Effective data input	–	2.0x - 50	–	150	–	110	ns
t <sub>RR</sub>	$\overline{RD}$ Pulse width	2.0x - 40	–	160	–	120	–	ns
t <sub>HR</sub>	$\overline{RD}$ rise → Data hold	0	–	0	–	0	–	ns
t <sub>RAE</sub>	$\overline{RD}$ rise → Address enable	x - 15	–	85	–	65	–	ns
t <sub>WW</sub>	WR pulse width	2.0x - 40	–	160	–	120	–	ns
t <sub>DW</sub>	Effective data → $\overline{WR}$ rise	2.0x - 50	–	150	–	110	–	ns
t <sub>WD</sub>	WR rise → Effective data hold	0.5x - 10	–	40	–	30	–	ns
t <sub>ACKH</sub>	Upper address → CLK fall	2.5x - 50	–	200	–	150	–	ns
t <sub>ACKL</sub>	Lower address → CLK fall	2.0x - 50	–	150	–	110	–	ns
t <sub>CKHA</sub>	CLK fall → Upper address hold	1.5x - 80	–	70	–	40	–	ns
t <sub>CCK</sub>	$\overline{RD}/\overline{WR}$ → CLK fall	x - 25	–	75	–	55	–	ns
t <sub>CKHC</sub>	CLK fall → $\overline{RD}/\overline{WR}$ rise	x - 60	–	40	–	20	–	ns
t <sub>DCK</sub>	Valid data CLK fall	x - 50	–	50	–	30	–	ns
t <sub>CWA</sub>	$\overline{RD}/\overline{WR}$ fall → Valid $\overline{WAIT}$	–	x - 40	–	60	–	40	ns
t <sub>AWAL</sub>	Lower address → Valid $\overline{WAIT}$	–	2.0x - 70	–	130	–	90	ns
t <sub>WAH</sub>	CLK fall → Valid $\overline{WAIT}$ hold	0	–	0	–	0	–	ns
t <sub>AWAH</sub>	Upper address → Valid $\overline{WAIT}$	–	2.5x - 70	–	180	–	130	ns
t <sub>CPW</sub>	CLK fall → Port Data Output	–	X + 200	–	300	–	280	ns
t <sub>PRC</sub>	Port Data Input → CLK fall	200	–	200	–	200	–	ns
t <sub>CPR</sub>	CLK fall → Port Data hold	100	–	100	–	100	–	ns

## AC Measuring Conditions

- Output level: High 2.2V/Low 0.8V, C<sub>L</sub> = 50pF  
 (However, CL = 100pF for AD0 ~ 7, A8 ~ 15, ALE,  $\overline{RD}$ ,  $\overline{WR}$ )
- Input level: High 2.4V/Low 0.45V (AD0 ~ AD7)  
 High 0.8V<sub>CC</sub>/Low 0.2V<sub>CC</sub> (excluding AD0 ~ AD7)

#### 4.4 Zero-Cross Characteristics

$V_{CC} = 5V \pm 10\%$   $TA = -40 \sim 85^{\circ}C$  (1 ~ 10MHz)  
 $TA = -20 \sim 70^{\circ}C$  (1 ~ 12.5MHz)

Symbol	Parameter	Condition	Min	Max	Unit
$V_{ZX}$	Zero-cross detection input	AC coupling $C = 0.1\mu F$	1	1.8	VAC p - p
$A_{ZX}$	Zero-cross accuracy	50/60Hz sine wave	–	135	mV
$F_{ZX}$	Zero-cross detection input frequency	–	0.04	1	kHz

#### 4.5 Serial Channel Timing-I/O Interface Mode

$V_{CC} = 5V \pm 10\%$   $TA = -40 \sim 85^{\circ}C$  (1 ~ 10MHz)  
 $CL = 50pF$   $TA = -20 \sim 70^{\circ}C$  (1 ~ 12.5MHz)

Symbol	Parameter	Variable		10MHz Clock		12.5MHz Clock		Unit
		Min	Max	Min	Max	Min	Max	
$t_{SCY}$	Serial Port Clock Cycle Time	8x	–	800	–	640	–	ns
$t_{OSS}$	Output Data Setup SCLK Rising Edge	6x - 150	–	450	–	330	–	ns
$t_{OHS}$	Output Data Hold After SCLK Rising Edge	2x - 120	–	80	–	40	–	ns
$t_{HSR}$	Input Data Hold After SCLK Rising Edge	0	–	0	–	0	–	ns
$t_{SRD}$	SCLK Rising Edge to Input DATA Valid	–	6x-150	–	450	–	330	ns





#### 4.6 8-bit Event Counter

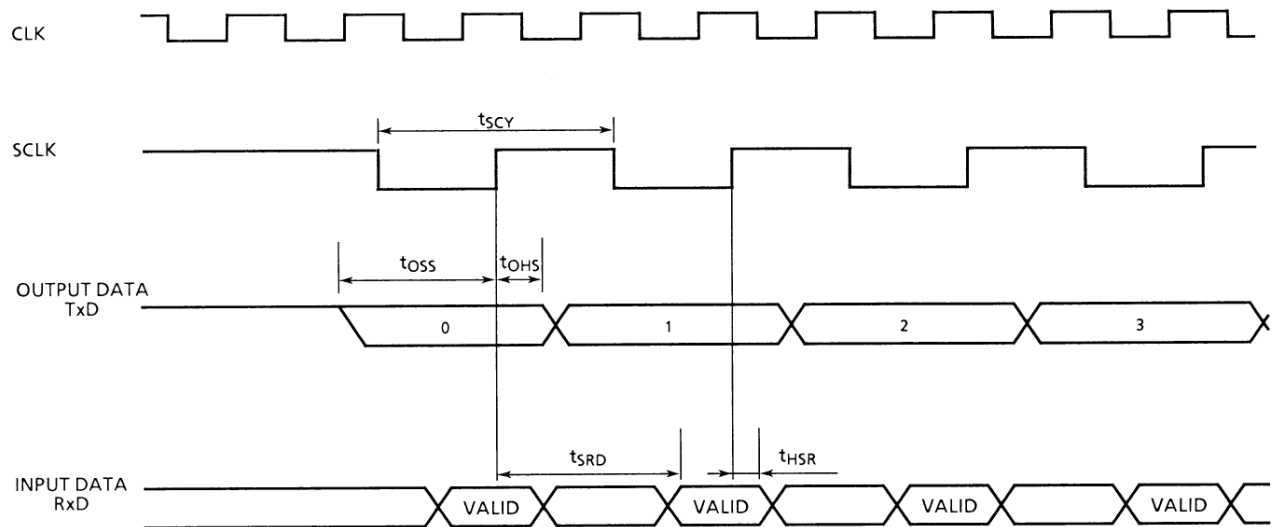
$V_{CC} = 5V \pm 10\%$   $TA = -40 \sim 85^{\circ}C$  (1 ~ 10MHz)  
 $TA = -20 \sim 70^{\circ}C$  (1 ~ 12.5MHz)

Symbol	Parameter	Variable		10MHz Clock		12.5MHz Clock		Unit
		Min	Max	Min	Max	Min	Max	
$t_{VCK}$	T12 clock cycle	8x + 100	–	900	–	740	–	ns
$t_{VCKL}$	T12 Low clock pulse width	4x + 40	–	440	–	360	–	ns
$t_{VCKH}$	T12 High clock pulse width	4x + 40	–	440	–	360	–	ns

#### 4.7 Interrupt Operation

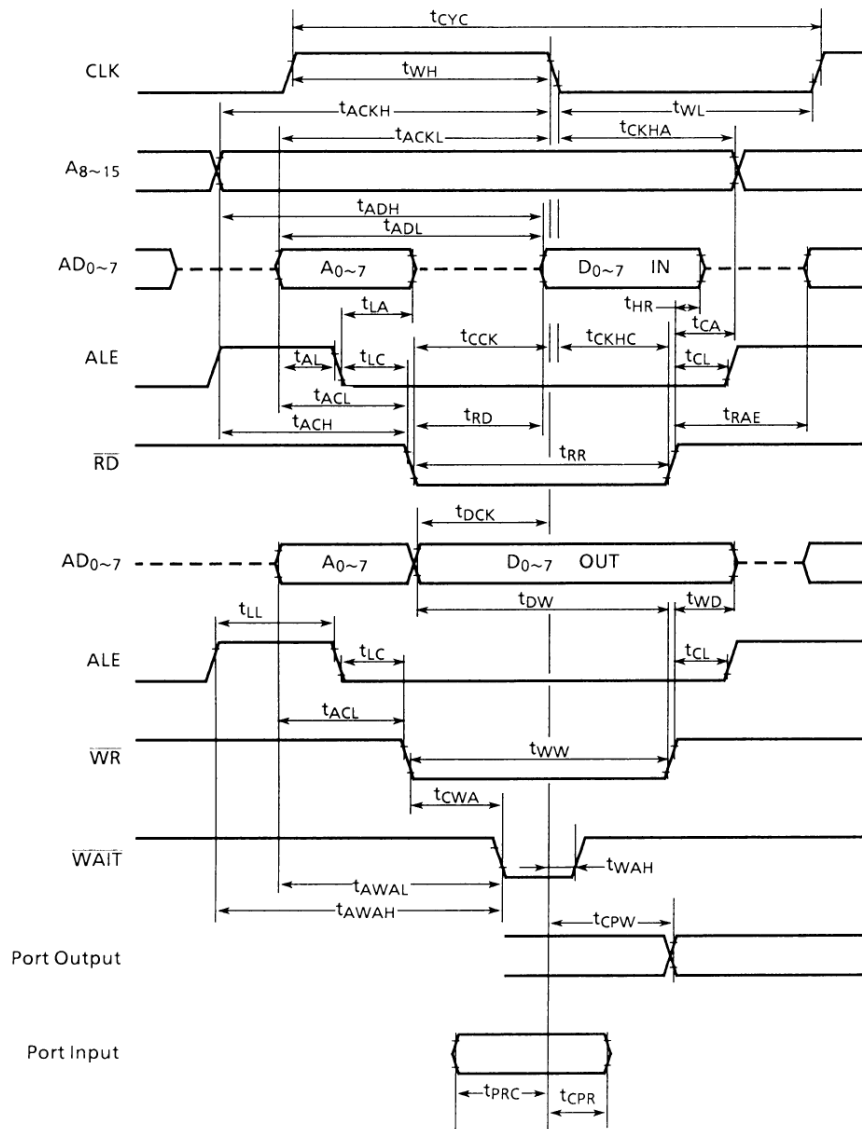
$V_{CC} = 5V \pm 10\%$   $TA = -40 \sim 85^{\circ}C$  (1 ~ 10MHz)  
 $TA = -20 \sim 70^{\circ}C$  (1 ~ 12.5MHz)

Symbol	Parameter	Variable		10MHz Clock		12.5MHz Clock		Unit
		Min	Max	Min	Max	Min	Max	
$t_{INTAL}$	$\overline{NMI}$ , INTO Low level pulse width 	4x	–	400	–	320	–	ns
$t_{INTAH}$	$\overline{NMI}$ , INTO High level pulse width 	4x	–	400	–	320	–	ns
$t_{INTBL}$	INT1 Low level pulse width 	8x + 100	–	900	–	740	–	ns
$t_{INTBH}$	INT1 High level pulse width 	8x + 100	–	900	–	740	–	ns

**4.8 I/O Interface Mode Timing**



### 4.9 Timing Chart



## 5. Table of Special Function Registers

The special function registers include the I/O ports and peripheral control registers allocated to the 32-byte addresses from FF80H to FF9FH.

- (1) I/O port
- (2) I/O port control
- (3) Timer/event counter control
- (4) Serial channel control
- (5) Interrupt control
- (6) Standby mode control

Format of table

Symbol	Name	Address	7	6	5	4	3	2	1	0

→ bit Symbol  
→ Read/Write  
→ Resetting Value  
→ Function

### TMP90C400 Special Function Register Address List

Address	Symbol	Address	Symbol
FF80H	P0	FF90H	TREG0
FF81H	P0CR	FF91H	TREG1
FF82H	P1	FF92H	TREG2
FF83H	P1CR	FF93H	TREG3
FF84H	P2	FF94H	TCLK
FF85H	P2CR	FF95H	TFFCR
FF86H	P2FR	FF96H	TMOD
FF87H	P3	FF97H	TRUN
FF88H	P3CR	FF98H	P3FR
FF89H	P4	FF99H	SCMOD
FF8AH	P4CR	FF9AH	SCCR
FF8BH	P5	FF9BH	SCBUF
FF8CH	P5CR	FF9CH	INTEF
FF8DH	P6	FF9DH	DMAEF
FF8EH	P6CR	FF9EH	IRFR
FF8FH	STBMOD	FF9FH	INTMR

## (1) I/O Port

MSB

LSB

Symbol	Name	Address	7	6	5	4	3	2	1	0
P0	Port 0	FF80H	P07	P06	P05	P04	P03	P02	P01	P00
			R/W							
			Input mode							
P1	Port 1	FF82H	P17	P16	P15	P14	P13	P12	P11	P10
			R/W							
			Input mode							
P2	Port 2	FF84H	P27	P26	P25	P24	P23	P22	P21	P20
			R/W							
			Input mode (with pull-up register )							
P3	Port 3	FF87H	P37	P36	P35	P34	P33	P32	P31	P30
			R/W							
			Input mode (with pull-up register)							
P4	Port 4	FF89H	P47	P46	P45	P44	P43	P42	P41	P40
			R/W							
			Input mode (with pull-up register)							
P5	Port 5	FF88H	P57	P56	P55	P54	P53	P52	P51	P50
			R/W							
			Input mode (with pull-up register)							
P6	Port 6	FF8DH	P67	P66	P65	P64	P63	P62	P61	P60
			R/W							
			Input mode							

Note: Read/Write

R/W: Either read or write is possible

R: Only read is possible.

W: Only write is possible.

## (2) I/O Port Control

MSB

LSB

Symbol	Name	Address	7	6	5	4	3	2	1	0
P0CR	Port 0 Control Reg.	FF81H  (prohibit RMW)	P07C	P06C	P05C	P04C	P03C	P02C	P01C	P00C
			W							
			0	0	0	0	0	0	0	0
			0: In 1: Out (I/O selected bit by bit)							
P1CR	Port 1 Control Reg.	FF83H  (prohibit RMW)	P17C	P16C	P15C	P14C	P13C	P12C	P11C	P10C
			W							
			0	0	0	0	0	0	0	0
			0: In 1: Out (I/O selected bit by bit)							
P2CR	Port 2 Control Reg.	FF85H  (prohibit RMW)	–	–	P25C	P24C	P23C	P22C	P21C	P20C
			W							
			0	0	0	0	0	0	0	0
			0: In 1: Out (I/O selected bit by bit)							

**TMP90C400/401**

## (2) I/O Port Control

MSB

LSB

Symbol	Name	Address	7	6	5	4	3	2	1	0	
P2CR	Port 2 Function Reg.	FF86H  (prohibit RMW)	–	WAITC1	WAITC0	NMIC	–	–	–	EXT	
			–	R/W		R/W	R/W	–	–	W	
			0	0	0	0	0	0	0	–	
			–	Wait control 00 : 2state Wait 01 : normal Wait 10 : non Wait 11 : Timer 0/1 Output		NMI control 0 : general- purpose port 1: input NMI		–	–	–	–
P3CR	Port 3 Control Reg.	FF88H  (prohibit RMW)	P37C	P36C	P35C	P34C	P33C	P32C	P31C	P30C	
			W								
			0	0	0	0	0	0	0	0	0
			0 : In 1 : Out (I/O selected bit by bit)								
P3CR	Port 3 Function Reg.	FF98H	ODE	TxDC	SCLK	TOE	ZCE3	ZCE2	ZCE1	ZCE0	
			W								
			0	0	0	0	0	0	0	0	0
			P37control 0 : CMOS 1 : Open drain		P37 Control 0 : general- purpose port 1 : TxD	P36 control 0 : general- purpose port 1 : SCLK	P34 control 0 : general- purpose port 1 : T01	P33 control :	P32 control 0 : general-purpose port 1 : ZCD enable (input only)	P31 control	P30 control
P4CR	Port 4 Control Reg.	FF8AH  (prohibit RMW)	P47C	P46C	P45C	P44C	P43C	P42C	P41C	P40C	
			W								
			0	0	0	0	0	0	0	0	0
			0 : In 1 : Out (I/O selected bit by bit)								
P5CR	Port 5 Control Reg.	FF8CH  (prohibit RMW)	P57C	P56C	P55C	P54C	P53C	P52C	P51C	P50C	
			W								
			0	0	0	0	0	0	0	0	0
			0 : In 1 : Out (I/O selected bit by bit)								
P6CR	Port 6 Control Reg.	FF8EH  (prohibit RMW)	P67C	P66C	P65C	P64C	P63C	P62C	P61C	P60C	
			W								
			0	0	0	0	0	0	0	0	0
			0 : In 1 : Out (I/O selected bit by bit)								

Note: Prohibit RMW: Prohibit Read Modify Write (Prohibit BIT/RES/SET Instructions)

## (3) Timer/event Counter Control

MSB

LSB

Symbol	Name	Address	7	6	5	4	3	2	1	0
TREG0	8bit Timer Register 0	FF90H prohibit RMW	–							
			W							
			Undefined							
TREG1	8bit Timer Register 1	FF91H prohibit RMW	–							
			W							
			Undefined							
TREG2	8bit Timer Counter Latch Register 2	0FF92H	–							
			R/W R : Counter Latch Register 2, W : 8bit Timer Register 2							
			Undefined							
TREG3	8bit Timer Latch Register 3	FF93H	–							
			W							
			R/W R : Counter Latch Register 3, W : 8bit Timer Register 3							

## (3) Timer/event Counter Control MSB

LSB

Symbol	Name	Address	7	6	5	4	3	2	1	0	
TCLK	8bit Timer Source Clock Control Reg.	0FF94H	T3CLK1	T3CLK0	T2CLK1	T2CLK0	T1CLK1	T1CLK0	T0CLK1	T0CLK0	
			R/W		R/W		R/W		R/W		
			0	0	0	0	0	0	0	0	
			00 : T02TRG 01 : $\emptyset$ T1 10 : $\emptyset$ T16 11 : $\emptyset$ T256		00 : T12 01 : $\emptyset$ T1 10 : $\emptyset$ T16 11 : $\emptyset$ T256 (8bit mode only)		00 : T00TRG 01 : $\emptyset$ T1 10 : $\emptyset$ T16 11 : $\emptyset$ T256		00 : T10 01 : $\emptyset$ T1 10 : $\emptyset$ T16 11 : $\emptyset$ T256 (8bit mode only)		
TFFCR	8bit Timer Flip-Flop Control Reg.	FF95H	LATCH3	LATCH3	–	–	TFF1C1	TFF1C0	TFF1IE	TFF1IS	
			W	W	–	–	W		R/W		
			1	1	–	–	–	–	0	0	
			0 : LATCH (one shot)		–	–	00 : Clear TFF1 01 : Set TFF1 10 : Invert TFF1 11 : Don't care		0 : Invert Disable 1 : Invert Enable		0 : Invert by 8bit timer 0 1 : Invert by 8bit Timer 1
TMOD	8bit Timer Mode Reg.	FF96H	T32M1	T32M0	–	–	T10M1	T10M0	PWM01	PWM00	
			R/W		–	–	R/W		R/W		
			0	0	–	–	0	0	0	0	
			00 : 8bit Timer/Counter 01 : 16bit Timer/Counter 10 : Don't care 11 : Don't care		–	–	00 : 8bit Timer 01 : 16bit Timer 10 : 8bit PPG 11 : 8bit PWM		00 : - 01 : $2^6 - 1$ PWM Period 10 : $2^7 - 1$ 11 : $2^8 - 1$		
TRUN	8bit Timer/Serial Channel Baud Rate Control Reg.	FF97H	BRATE1	BRATE0	PRRUN	–	T3RUN	T2RUN	T1RUN	T0RUN	
			R/W		R/W						
			0	0	0	0	0	0	0	0	
			00 : 300/150 bps 01 : 1200/600 10 : 4800/2400 11 : 19200/9600		Prescaler & Timer Run/Stop Control 0 : Stop & Clear 1 : Run (Count up)						

## (4) Serial Channel Control MSB

LSB

Symbol	Name	Address	7	6	5	4	3	2	1	0		
SCMOD	Serial Channel Mode Reg.	FF99H	TB8	Fixed at "0"	RXE	WU	SM1	SM0	SC1	SC0		
			R/W									
			Undefined	0	0	0	0	0	0	0		
			Transmission Bit-8 data in 9bit UART	Write "0"	1 : Receive Enable	1 : Wake up Enable	00 : I/O interface 01 : UART 7bit 10 : UART 8bit 11 : UART 9bit		00 : T02TRG 01 : BR 10 : $\emptyset$ 1 11 : BR 1/2		U A R T	
SCCR	Serial Channel Control Register	FF9AH	RB8	–	–	–	–	–	OERR	FERR		
			R	–	–	–	–	–	R (Cleared to "0" by reading)			
			Undefined	0	0	0	0	0	–	–		
			Receiving Bit 8 data	–	–	–	–	–	1 : error Overrun	1 : error Flaming		

**TMP90C400/401**

## (4) Serial Channel Control

MSB

LSB

Symbol	Name	Address	7	6	5	4	3	2	1	0
SCBUF	Serial Channel Buffer Register	FF9BH	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
			TB7	TB6	TB5	TB4	TB3	TB2	TB1	TB0
		prohibit RMW	R (Receiving)/W (Transmission)							
			Undefined							

Also refer to P3FR, TRUN register.

Note: BR: Baud Rate Generator

## (5) Interrupt Control

MSB

LSB

Symbol	Name	Address	7	6	5	4	3	2	1	0	
INTEF	Interrupt Enable Mask Reg.	FF9CH	IE0	IET0	IET1	IET2	IET3	IE1	IERX	IETX	
			R/W								
			0	0	0	0	0	0	0	0	
			1 : Enable 0 : Disable								
DMAEF	Micro DMA Enable Register	FF9DH	DE0	DET0	DET1	DET2	DET3	DE1	DERX	DETX	
										R/W	
			-	-	-	-	-	-	0	0	
			1 : Enable 0 : Disable								
IRFR	Interrupt Request Flag & IRF Clear	FF9EH	IRF0	IRFT0	IRFT1	IRFT2	IRFT3	IRF1	IRFRX	IRFTX	
			R (Only IRF clear code can be used to write)								
			0	0	0	0	0	0	0	0	
			1 : Interrupt being requested (IRF is cleared to "0" by writing IRF clear Code)								
INTMR	INT0 Mode Control Reg.	FF9FH	-	-	-	-	-	-	-	EDGE	
			-	-	-	-	-	-	-	R/W	
			-	-	-	-	-	-	-	0	
			-	-	-	-	-	-	-	0 : level 1 : ↑edge	

(6) Standby mode control

MSB

LSB

Symbol	Name	Address	7	6	5	4	3	2	1	0
STBMOD	Standby Mode Reg.	FF8FH	-	-	-	-	HALTM1	HALTM0	EXF	DRIVE
			-	-	-	-	R/W		R	R/W
			-	-	-	-	0	0	Undefined	0
			-	-	-	-	Standby mode 00 : RUN mode 01 : STOP mode 10 : IDLE1 mode 11 : IDLE2 mode		Invert each time EXX instruction is executed	1 : to drive pin in STOP mode

## 6. Port Section Equivalent Circuit Diagram

- Reading The Circuit Diagram

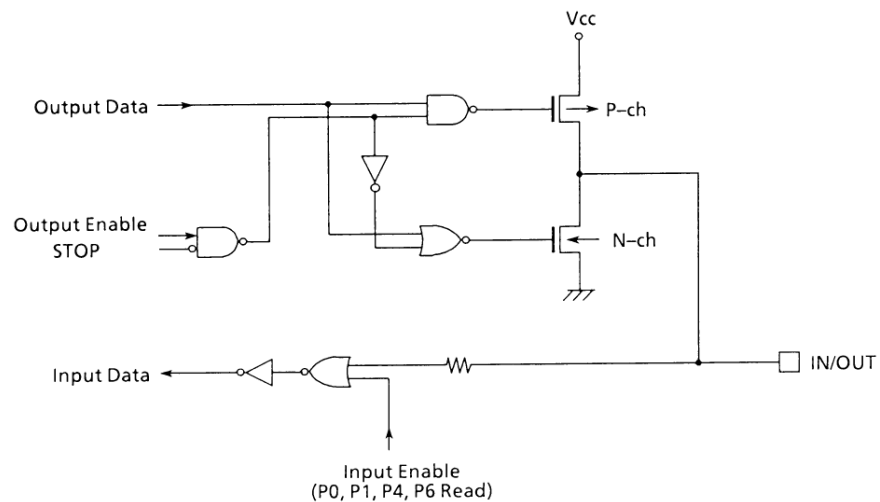
Basically, the gate singles written are the same as those used for the standard CMOS logic IC [74HCXX] series.

The dedicated signal is described below.

STOP: This signal becomes active "1" when the hold mode setting register is set to the STOP mode and the CPU executes the HALT instruction. When the drive enable bit [DRIVE] is set to "1", however, STP remains at "0".

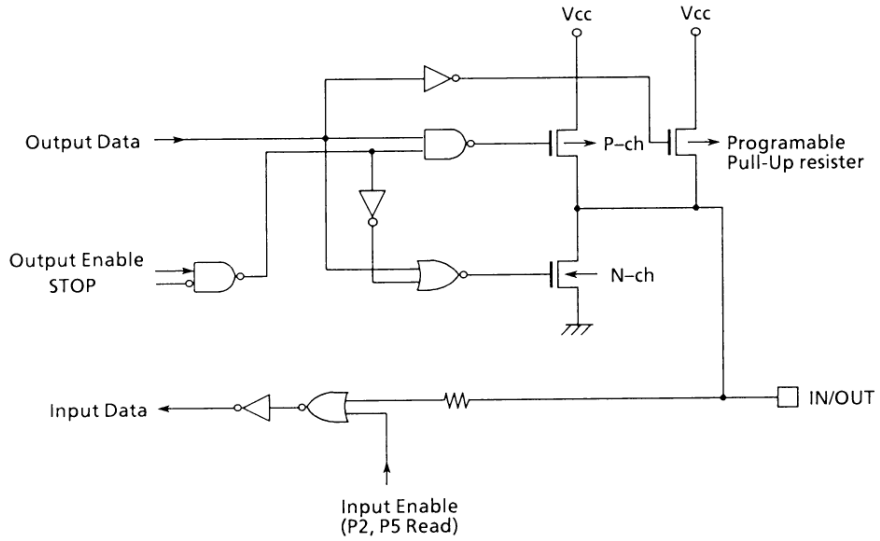
- The input protection resistans ranges from several tens of ohms to several hundreds of ohms.

- PO (AD0 ~ AD7), P1 (A8 ~ A15), P4, P6

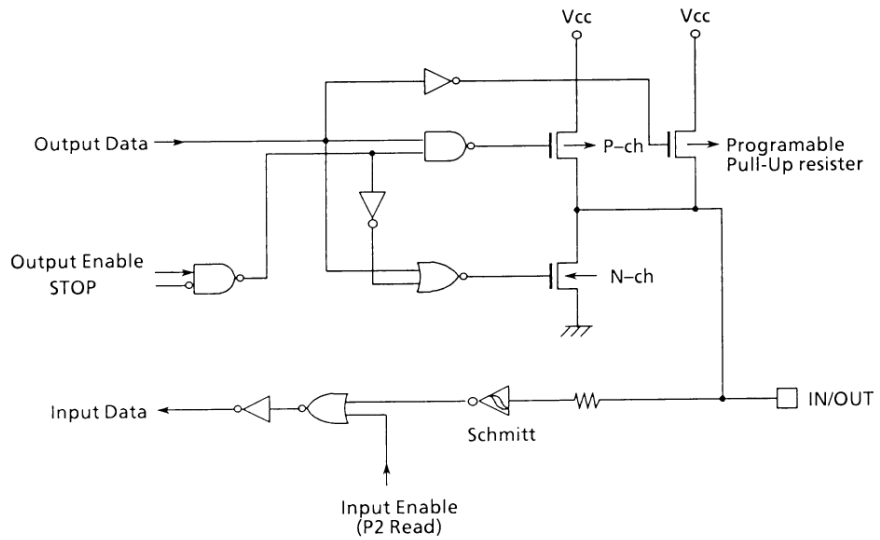




- P20 ~ P23, P25, P5

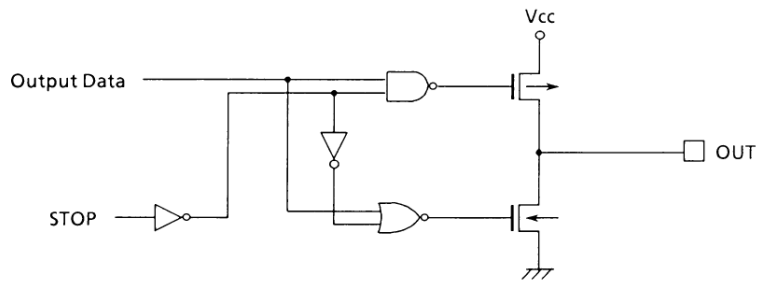


- P24 ( $\overline{\text{NMI}}$ )

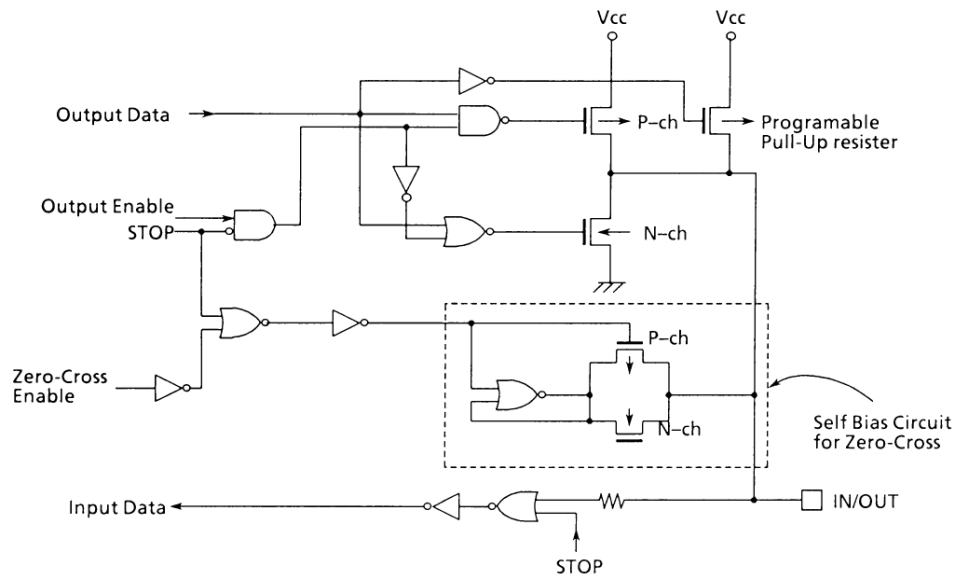


**TMP90C400/401**

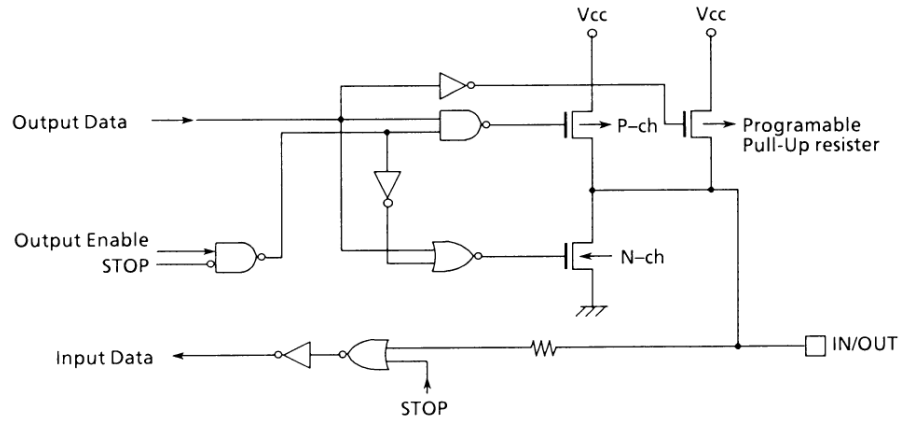
- P26 ( $\overline{RD}$ ), P27 ( $\overline{WR}$ )



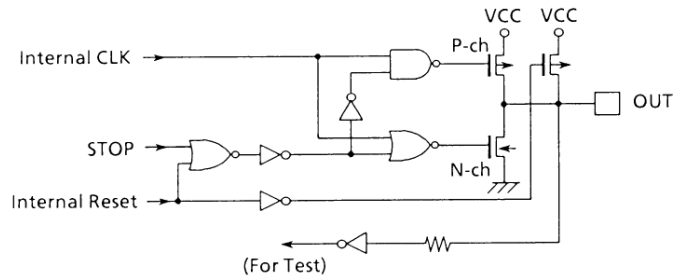
- P30 ~ P33



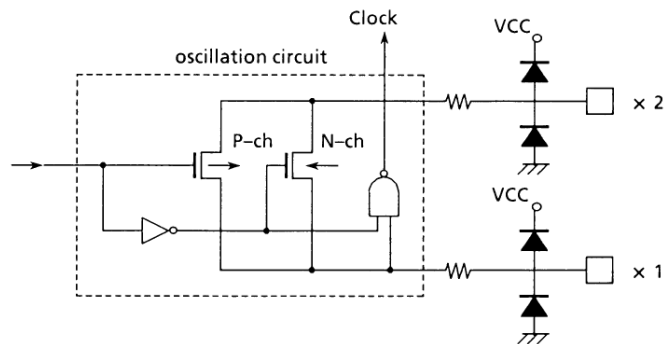
- P34 ~ P37



- CLK

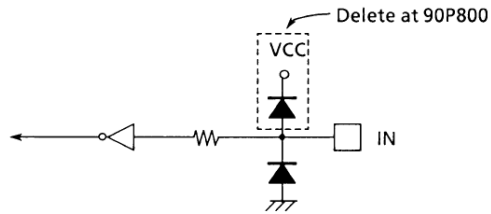


- X1, X2

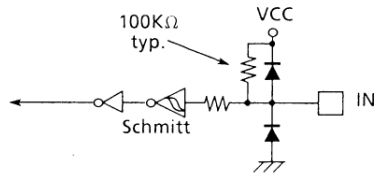


**TMP90C400/401**

- $\overline{EA}$



- $\overline{RESET}$



- ALE

