CMOS 16-Bit Microcontroller
TMP93CW76F

1. Outline and Feature

TMP93CW76F is a high-speed advanced 16-bit microcontroller developed for application with VCR system control, software servo motor control, VFT driver and timer control.

In addition to basics such as I/O ports, the TMP93CW76F has high-speed/high-precision signal measuring circuit, PWM (Pulse-Width-Modulator) and high-precision real timing pulse generator.

The device characteristics are as follows:

(1) Original 16-bit CPU (900L_CPU)
- TLCS-90 instruction mnemonic upward compatible
- 16 Mbyte linear address space
- General-purpose registers and register bank system
- 16-bit multiplication/division and bit transfer/arithmetic instructions
- High-speed micro DMA: 4 channels (2 μs / 2 byte at 16 MHz)

(2) Minimum instruction execution time: 250 ns at 16 MHz

(3) Internal ROM: 128 Kbyte

(4) Internal RAM: 2.5 Kbyte

(5) 20-bit time-base-counter (TBC)
- free running counter
- accuracy: 125 ns (at 16 MHz)
- overflow: 131 ms (at 16 MHz)

(6) 8-bit timer (TC0): 1 channel
- for CTL linear time counter

(7) 16-bit timer (TC1-5): 5 channels
- C-sync count, capstan FG count, general : (3 channels)

(8) Timing pulse generator (TPG): 2 channels
- (16-bit-timing data + 6-bit-output data) with 8-stages FIFO : 1 channel
- (16-bit-timing data + 4-bit-output data) : 1 channel
- accuracy: 500 ns (at 16 MHz)

980910EBP1

- For a discussion of how the reliability of microcontrollers can be predicted, please refer to Section 1.3 of the chapter entitled Quality and Reliability Assurance / Handling Precautions.
- TOSHIBA is continually working to improve the quality and the reliability of its products. Nevertheless, semiconductor devices in general can malfunction or fail due to their inherent electrical sensitivity and vulnerability to physical stress. It is the responsibility of the buyer, when utilizing TOSHIBA products, to observe standards of safety, and to avoid situations in which a malfunction or failure of a TOSHIBA product could cause loss of human life, bodily injury or damage to property. In developing your designs, please ensure that TOSHIBA products are used within specified operating ranges as set forth in the most recent products specifications. Also, please keep in mind the precautions and conditions set forth in the TOSHIBA Semiconductor Reliability Handbook.
- The products described in this document are subject to the foreign exchange and foreign trade laws.
- The information contained herein is presented only as a guide for the applications of our products. No responsibility is assumed by TOSHIBA CORPORATION for any infringements of intellectual property or other rights of the third parties which may result from its use. No license is granted by implication or otherwise under any intellectual property or other rights of TOSHIBA CORPORATION or others.
- The information contained herein is subject to change without notice.

Purchase of TOSHIBA I^{2}C components conveys a license under the Philips I^{2}C Patent Rights to use these components in an I^{2}C system, provided that the system conforms to the I^{2}C Standard Specification as defined by Philips.

This datasheet has been downloaded from http://www.digchip.com at this page

(9) Pulse width modulation outputs (PWM)
- 14-bit PWM: 3 channels (for controlling capstan,drum and tuner)
- 8-bit PWM: 1 channel (for controlling volume)
- carrier frequency: 48.8 kHz (at 16 MHz)

(10) 24-bit time base counter capture circuit (Capture 0)
- (18-bit timing data + 6 bit trigger data) with 8-stages FIFO: 1 channel
- capture input sources: Remote-control-input (RMTIN), V-sync, CTL, Drum-PG, general (1 channel)
- accuracy: 500 ns (at 16 MHz)

(11) 17-bit time base counter capture circuit (Capture 1/2)
- (16-bit timing data + 1-bit trigger data): 2 channels
- capture input sources: Drum-FG, Capstan-FG
- accuracy: 125 ns (at 16 MHz)

(12) VISS/VASS detection circuit (VISS/VASS)
- CTL duty detection
- VASS data 16-bit latch

(13) Composite-sync-signal (C-sync) input (C-sync In)
- Vertical-sync-signal (V-sync) separation (V-sepa)

(14) Head Amp switch/Color Rotary control (HA/CR)

(15) Pseudo-V/H generator (PV/PH)

(16) 8-bit A/D converter (ADC): 10 channels
- Conversion speed: 95 states (11.8 μs at 16 MHz)

(17) Serial Channel (SIO): 1channel

(18) Serial bus I/F
- 8-bit sync: 1 channel
- I^2CBUS with 8-stages FIFO: 1 channel/2 ports

(19) Watch dog timer (WDT)

(20) Interrupt controller (INTC)
- CPU: 8 sources $\rightarrow$ SWI instruction and Illegal instruction
- Internal: 17 sources ┐ 7-level priority can be set.
- External: 5 sources ┘

(21) I/O ports
- 67-I/O ports (multiplexed functional pins.)
  - • • • High Break Down Voltage PortE, F are Included: 14-I/O ports
- 8-Input ports (P40/AIN2-P47/AIN9)
- 10 Output ports (PortD, C: High Break Down Voltage)

(22) Standby function: 2-halt modes (IDLE, STOP)

(23) System clock function
- Dual clock operation16 MHz (High-speed: normal)/32 kHz (Low-speed:slow)
  - • • • 17-bit Real Time Counter built in

(24) Operating Voltage
- Vcc = 2.7 to 5.5 V (at 32 kHz)
- Vcc = 4.5 to 5.5 V (at 16 MHz)

(25) Package
- 100 pin QFP 14 mm $\times$ 20 mm (Pin pitch: 0.65 mm)
- Type nameQFP100-P-1420-0.65A

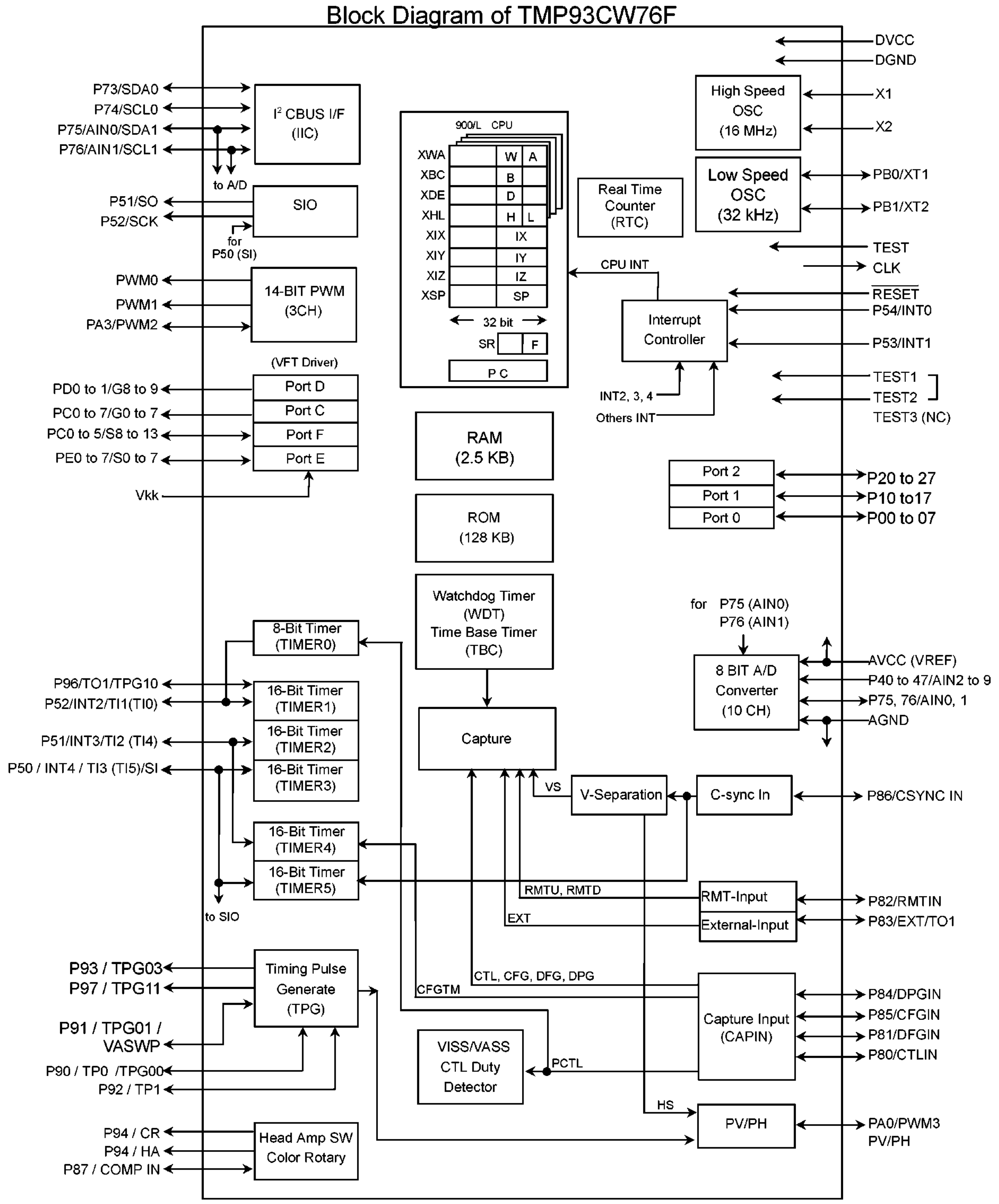


Figure 1.1 TMP93CW76F Block Diagram

2. Pin Assignment and Functions

The assignment of input and output pins for the TMP93CW76F, their names and functions are described below.

2.1 Pin Assignment

Figure 2.1.1 shows pin assignment of the TMP93CW76F.

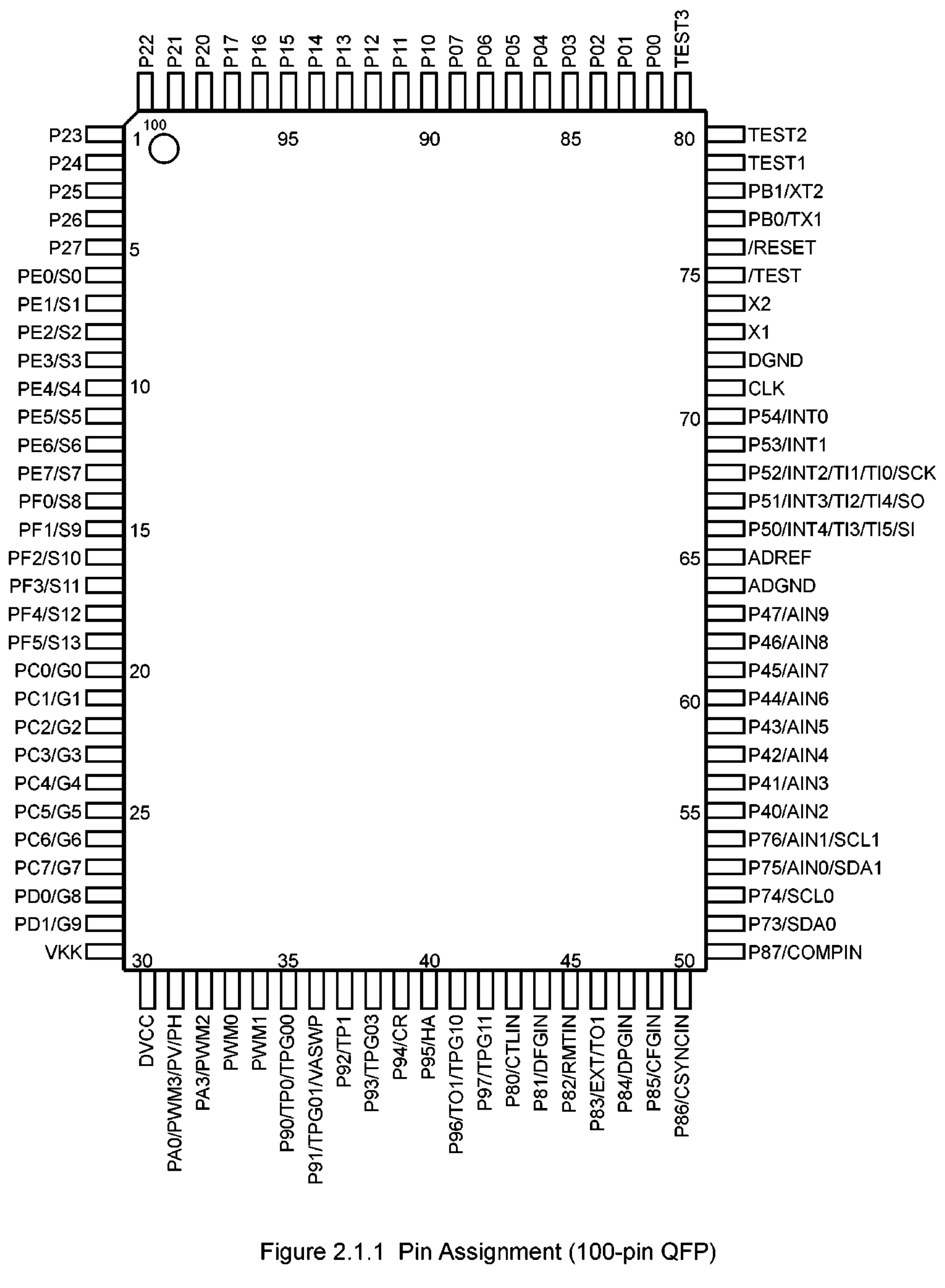


Figure 2.1.1 Pin Assignment (100-pin QFP)

2.2 Pin Names and Functions

The names of input/output pins and their functions are described below.

Table 2.2.1 Pin Names and Function (1/3)

Pin name	Number of pins	I/O	Functions
P00 to P07	8	I/O	port0: I/O ports
P10 to P17	8	I/O	port1: I/O ports
P20 to P27	8	I/O	port2: I/O ports
P40 to P47	8	Input	port4: Input ports
/AIN2 to AIN9		Input	Analog input: Input to A/D converter
P50	1	I/O	Port50: I/O port(schmitt input)
/INT4		Input	External Interrupt request input 4: Rising edge/Falling edge programable
/TI3		Input	16bit timer3(TC3): Input 3
/TI5		Input	16bit timer5(TC5): input 5
/SI		Input	SIO received channel
P51	1	I/O	Port51: I/O port(schmitt input)
/INT3		Input	External Interrupt request input 3: Rising edge/Falling edge programable
/TI2		Input	16 bit timer2(TC2): Input 2
/TI4		Input	16 bit timer4(TC4): input 4
/SO		Input	SIO sending channel
P52	1	I/O	Port52: I/O port(schmitt input)
/INT2		Input	External Interrupt request input 2: Rising edge/Falling edge programable
/TI1		Input	16 bit timer1(TC1): Input 1
/TI0		Input	8 bit Timer(TC0): Input 0
/SCK		Input	SIO clock line
P53	1	I/O	Port53: I/O port(schmitt input)
/INT1		Input	External Interrupt request pin1: Rising edge/Level(Low-->High) programable SU/TU-FG(Reel Tacho)
P54	1	I/O	Port54: I/O port(schmitt input)
/INT0		I/O	External Interrupt request pin0: Rising edge/Falling edge programable
P73	1	I/O	Port73: I/O port(schmitt input)
/SDA0		I/O	I^{2}CBUS SDA0 line
P74	1	I/O	Port74: I/O port(schmitt input)
/SCL0		I/O	I^{2}CBUS SCL0 line
P75	1	I/O	Port75: I/O port(schmitt input)
/SDA1		I/O	I^{2}CBUS SDA1 line
/AIN0		Input	Analog input 0: Analog input signal for A/D converter
P76	1	I/O	Port4: Input ports(schmitt input)
/SCL1		I/O	I^{2}CBUS SCL1 line
AIN1		Input	Analog input 1: Analog input signal for A/D converter
P80	1	I/O	Port80: I/O port(schmitt input)
/CTLIN		Input	PBCTL Capture input(Capture 0)
P81	1	I/O	Port81: I/O port(schmitt input)
/DFGIN		Input	DFG Capture input(Capture 1)

Table 2.2.1 Pin Names and Function (2/3)

Pin name	Number of pins	I/O	Functions
P82 /RMTIN	1	I/O Input	Port82: I/O port(schmitt input) Remote Control Signal Capture input
P83 /EXT /TO1	1	I/O Input Output	Port83: I/O port(schmitt input) External Capture input(Capture 0) Timer Out 1
P84 /DPGIN	1	I/O Input	Port84: I/O port(schmitt input) DPG Capture input(Capture 0)
P85 /CFGIN	1	I/O Input	Port85: I/O port(schmitt input) DFG Capture input(Capture 2)
P86 /CSYNCIN	1	I/O I/O	Port86: I/O port(schmitt input) C-sync Capture input
P87 /COMPIN	1	I/O I/O	Port87: I/O port(schmitt input) Envelope Comparate Input(to HA/CR)
P90 /TP0 /TPG00	1	I/O Output Output	Port90: I/O port Timing Pulse output 0 TPG00: TPG output 00
P91 /VASWP / TPG01	1	I/O Output Output	Port91: I/O port Video/Audio head switching control signal output TPG01: TPG output 01
P92 /TP1	1	I/O Output	Port92: I/O port Timing Pulse output 1
P93 /TPG03	1	I/O Output	Port93: I/O port TPG03: TPG output 03
P94 /CR	1	I/O Output	Port94: I/O port Color Rotary Output
P95 /HA	1	I/O Output	Port95: I/O port Head Amp Switching Control Output
P96 /TO1 /TPG10	1	I/O Output Output	Port96: I/O port Timer Out 1 TPG10: TPG output 10
P97 /TPG11	1	I/O Output	Port97: I/O port TPG11: TPG output 11
PA0 /PV-PH /PWM3	1	I/O Output Output	PortA0: I/O port PV/PH 3-state Output PWM(8bit)output 3: Volume control
PA3 /PWM2	1	I/O Output	PortA3: I/O port PWM(14bit)output 2: Vol.-syn tuner PWM
PWM0	1	Output	PWM(14bit) output0: Capstan PWM
PWM1	1	Output	PWM(14bit) output1: Drum PWM
PB0 /XT1	1	I/O Input	PortB0: I/O port(Open Drain Output) Low Frequency Oscillator connecting pin
PB1 /XT2	1	I/O Output	PortB1: I/O port(Open Drain Output) Low Frequency Oscillator connecting pin
PC0 to PC7 /G0 to G7	8	Output Output	PortC: High break down voltage Outputs Grid Outputs

Table 2.2.1 Pin Names and Function (3/3)

Pin name	Number of pins	I/O	Functions
PD0,1 /G8, 9	2	Output Output	PortD: High break down voltage Outputs Grid Outputs
PE0 to PE7 /S0 to S7	8	I/O	PortE: I/O ports (with pull-down R) Segment Outputs
PF0 to PF5 /S8 to S13	6	I/O	PortF: I/O ports (with pull-down R) Segment Outputs
TEST1	1	Output	TEST1 should be connected with TEST2 pin.
TEST2	1	Input	
TEST3(NC)	1	Output	TEST3(NC) should be open connection.
CLK	1	Output	Clock output: Output (System Clock ÷ 2) clock. pull-up during reset. can be set to Output disable for reducing noise.(Initial Disable)
$\overline{\text{TEST}}$	1	Input	Test pin: Always set to “Vcc”level
$\overline{\text{RESET}}$	1	Input	Reset: Initializes LSI. (with pull-up R)
X1	1	Input	High Frequency Oscillator connecting pins (16 MHz)
/X2	1	Output	High Frequency Oscillator connecting pins (16 MHz)
VKK	1		VFT Driver power supply pin
DVCC	1		Power supply pin
DGND	1		GND pin (0 V)
ADREF	1		A/D reference Voltage input
ADGND	1		A/D ground input

3. Operation

This section describes the functions and basic operational blocks of TMP93CW76F devices.
See the “7. Points of Concern and Restrictions” for the using notice and restrictions for each block.

3.1 CPU

TMP93CW76F devices have a built-in high-performance 16-bit CPU (900 / L CPU). (For CPU operation, see TLCS-900 / L CPU in the previous section).

This section describes CPU functions unique to the TMP93CW76F that are not described in the previous section.

3.1.1 Reset

To reset the TMP93CW76F, the RESET input must be kept at 0 for at least 10 system clocks. (1.25 μs at 16 MHz) within the operating voltage range and with a stable oscillation.

When reset is accepted, the CPU sets as follows:

- Program Counter (PC) according to Reset Vector that is stored FFFF00H to FFFF02H.

PC (7 to 0)	← stored data in location FFFF00H
PC (15 to 8)	← stored data in location FFFF01H
PC (23 to 16)	← stored data in location FFFF02H

- Stack pointer (XSP) for system mode to 100H.
- IFF2 to 0 bits of status register to 111. (Sets mask register to interrupt level 7.)
- MAX bit of status register to 1. (Sets to maximum mode)
- Bits RFP2 to 0 of status register to 000. (Sets register banks to 0.)

When reset is released, instruction execution starts from PC (reset vector). CPU internal registers other than the above are not changed.

When reset is accepted, processing for built-in I/Os, ports, and other pins is as follows

- Initializes built-in I/O registers as per specifications.
- Sets port pins (including pins also used as built-in I/Os) to general-purpose input / output port mode.

Note: By resetting, register in the CPU except program counter (PC),status register (SR) and stack pointer (XSP) and the data in internal RAM are not changed.

3.2 Memory Map

Figure 3.2.1 is a memory map of the TMP93CW76F.

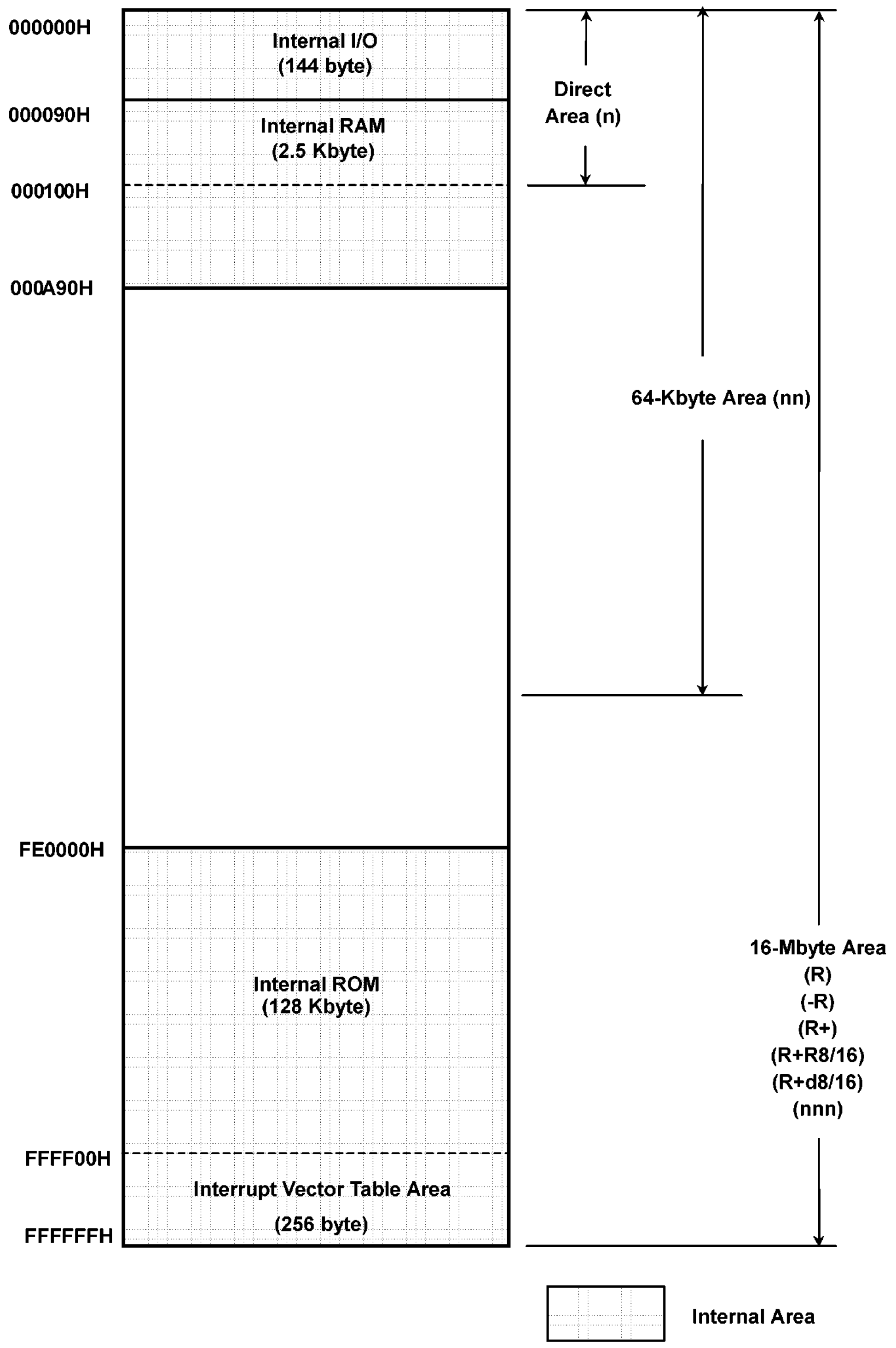


Figure 3.2.1 Memory map

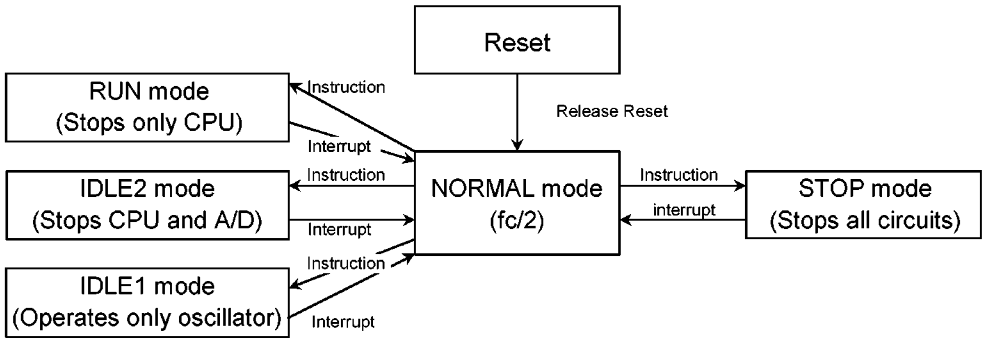
3.3 Dual Clock, Standby Function

Dual Clock, Stand by Control Circuits consist of System clock Controller, Timing clock Generator for I/O Block, Real Time Clock Generator, and Standby Controller.

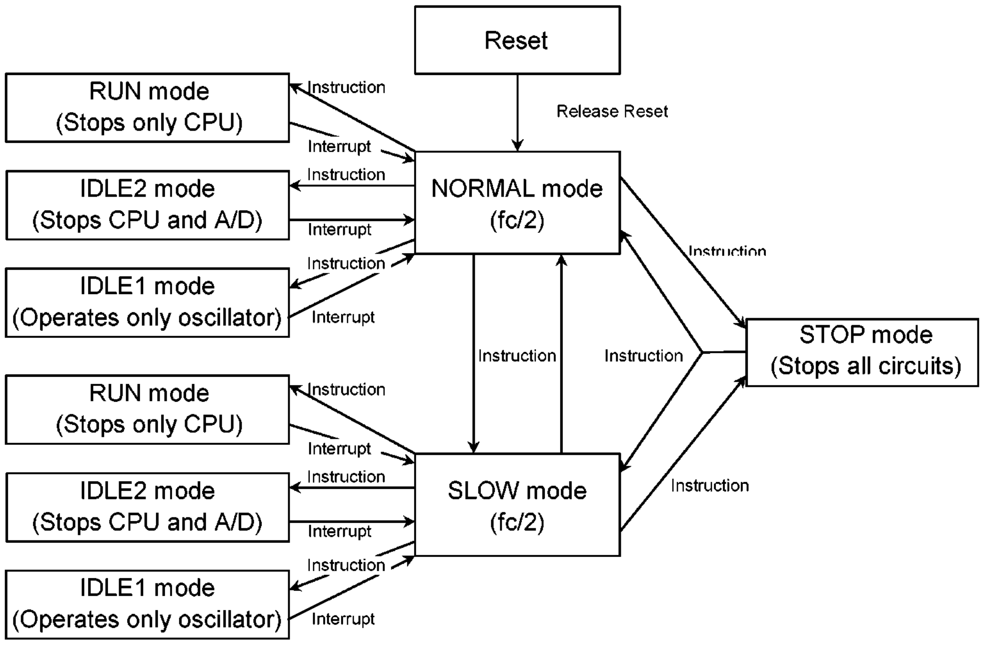
The Oscillator operating mode is classified to Single Clock mode (only X1, X2 pin) and Dual Clock mode (X1, X2, XT1, XT2 pin).

Figure 3.3.1 shows a transition figure. Figure 3.3.2 shows the block diagram.

Figure 3.3.3 shows I/O registers. Table 3.3.1 shows the internal operation and system clock.


Reset
Release Reset
RUN mode (Stops only CPU)
IDLE2 mode (Stops CPU and A/D)
IDLE1 mode (Operates only oscillator)
NORMAL mode (fc/2)
STOP mode (Stops all circuits)
Instruction
Interrupt
interrupt

(a) Single Clock mode transition figure


Reset
Release Reset
RUN mode (Stops only CPU)
IDLE2 mode (Stops CPU and A/D)
IDLE1 mode (Operates only oscillator)
NORMAL mode (fc/2)
SLOW mode (fc/2)
STOP mode (Stops all circuits)
Instruction
Interrupt

(b) Dual Clock mode transition figure

Figure 3.3.1 Transition Figure

The Clock Frequency input from X1, X2 pin is called fc, and the Clock Frequency input from XT1, XT2 pin is called fs. The clock frequency selected by SYSCR1<SYSCK> is called system clock f_{FPH}. The devided clock of f_{FPH} is called system clock f_{SYS}, and the 1 cycle of f_{SYS} is called 1 state.

Table 3.3.1 Internal operation and system clock

Operating Mode		Oscillator		CPU	internal I/O	System clock f_{SYS}
		High Frequency (fc)	Low Frequency (fs)			
Single Clock	RESET	oscillation	stop	reset	reset	fc/2
	NORMAL			operate	operate	
	RUN			stop		
	IDLE2				stop only A/D	
	IDLE1				stop	
	STOP	stop				stop
Dual Clock	RESET	oscillation	stop	reset	reset	fc/2
	NORMAL		progamable	operate	operate	
	SLOW	programable	oscillation			fs/2
	RUN	Oscillator being used as system clock: oscillation Other oscillator: programmable		stop		programable (fc/2, fs/2)
	IDLE2				stop only A/D	
	IDLE1				stop	
	STOP	stop				stop

The TMP93CW76F does not have a clock gear circuit.

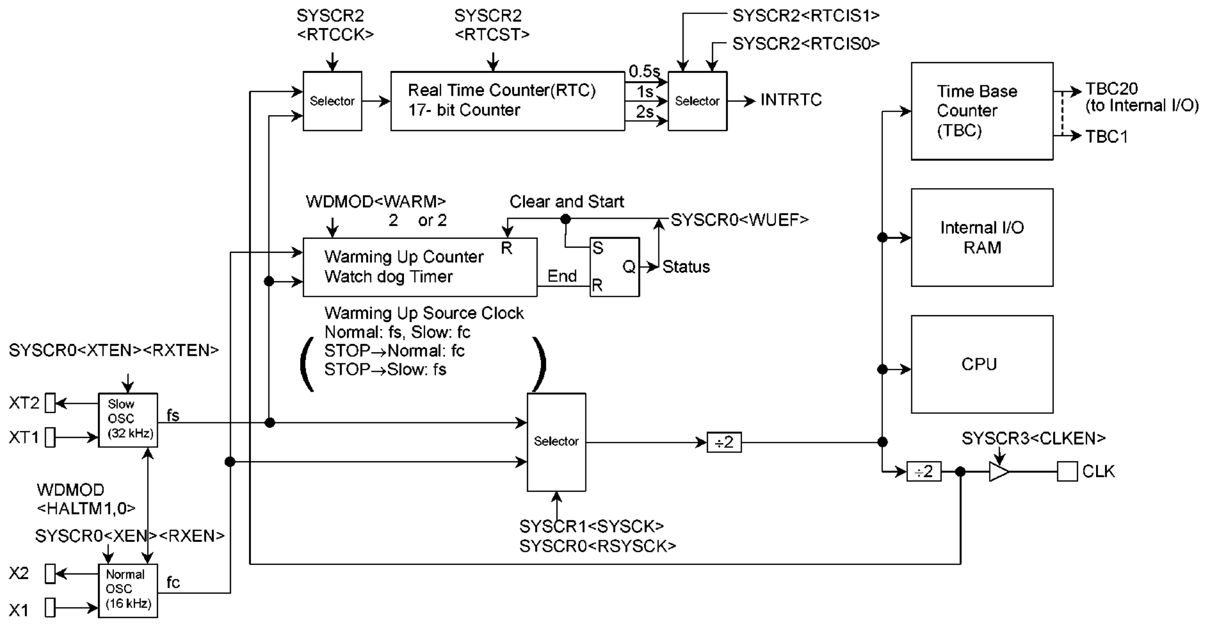


Figure 3.3.2 Block Diagram of Dual Clock, Standby circuits

SYSCR0 (006EH)

	7	6	5	4	3	2	1	0
bit Symbol	XEN	XTEN	RXEN	RXTEN	RSYSCK	WUEF		
Read/Write	R/W							
After reset	1	0	1	0	0	0		
Function	High Frequency oscillator (fc) 0: stop 1: oscillation	Low Frequency oscillator (fs) 0: stop 1: oscillation	High Frequency oscillator (fc) after released STOP mode 0: stop 1: oscillation	Low Frequency oscillator (fs) after released STOP mode 0: stop 1: oscilation	slect clock after released STOP mode 0: fc 1: fs	Warm-Up Timer 0 write: don't care 1 write: start timer 0 read:end warm-up 1 read: not end warm-up		

SYSCR1 (006FH)

	7	6	5	4	3	2	1	0
bit Symbol					SYSCK			
Read/Write	(R/W)				R/W	(R/W)		
After reset	(0)				0	(0)	(0)	(0)
Function	Always set to"0"				select system clock 0: fc 1: fs	Always set to "0"		

SYSCR2 (006CH)

	7	6	5	4	3	2	1	0
bit Symbol				RTCCK	RTCST	RTCIS1	RTCIS0	
Read/Write	(R/W)	(R/W)		R/W	R/W	R/W		
After reset	(0)	(0)		0	0	0		
Function	Always set to "0"			RTC clock source select 0: fs (32 kHz) 1: fc/4 or fs/4	RTC count 0: Stop & Counter Clear 1: Start	Interval time cotrol of RTC interrupt 00: fc/2^{15} or fs/2^{15} 01: fc/2^{16} or fs/2^{16} 10: fc/2^{14} or fs/2^{14} 11: Reserved [Hz]		

WDMOD (005CH)

	7	6	5	4	3	2	1	0
bit Symbol	WDTE	WDTP1	WDTP0	WARM	HALTM1	HALTM0	RESCR	DRVE
Read/Write	R/W							
After reset	1	0	0	0	0	0	0	0
Function	WDT control 0: disable 1: enable	WDT Detection Time 00: $2^{15}/f_{SYS}$ 01: $2^{17}/f_{SYS}$ 10: $2^{19}/f_{SYS}$ 11: $2^{21}/f_{SYS}$		Warm-Up Timer 0: 2^{14}/ frequency inputted 1: 2^{16}/ frequency inputted	Stand by mode 00: RUN mode 01: STOP mode 10: IDLE1 mode 11: IDLE2 mode		0: Don't care 1: Connects WDT output to RESET pin Internsly	0: I/O off 1: Drive the pin in STOP mode

Note1: SYSCR1<bit 6-4>and SYSCR0<bit 1-0>area read as "1".

Note2: Writing "0" to SYSCR1<SYSCK>enables the high-frequency oscillator regardless of the value of SYSCR0<XEN>. Additionally, writing "1" to <SYSCK>ragister enables the low-frequency oscillator regardless of the value of SYSCR0<XTEN>.

Note3: When SYSCR2<RTCCK> is "1", RTC input clock is fc/4 or fs/4 depending on operation mode.

Figure 3.3.3 I/O registers about Dual Clock, Standby

3.3.1 System Clock Controller

The system clock controller generates system clock (f_{SYS}) for CPU core and internal I/O. It contains two oscillation circuits. The register SYSCR1<SYSCK> changes system clock to either fc or fs, SYSCR0<XEN>, <XTEN> controls enable / disable each oscillator.

The system clock (f_{SYS}) is set to fc/2 because of <XEN>="1", <XTEN>="0", <SYSCK>="0" by resetting.

For example, f_{SYS} is set to 8 MHz by resetting the case of 16-MHz oscillator is connected to X1, X2 pins.

The high frequency (fc) and low frequency (fs) clocks can be easily obtained by connecting a resonator to the X1 / X2, XT1 / XT2 pins, respectively. Clock input from an external oscillator is also possible.

The XT1, XT2 pins have also Port PB0, PB1 function. Therefore the case of single clock mode, the XT1, XT2 pins can be used as I/O port pins.

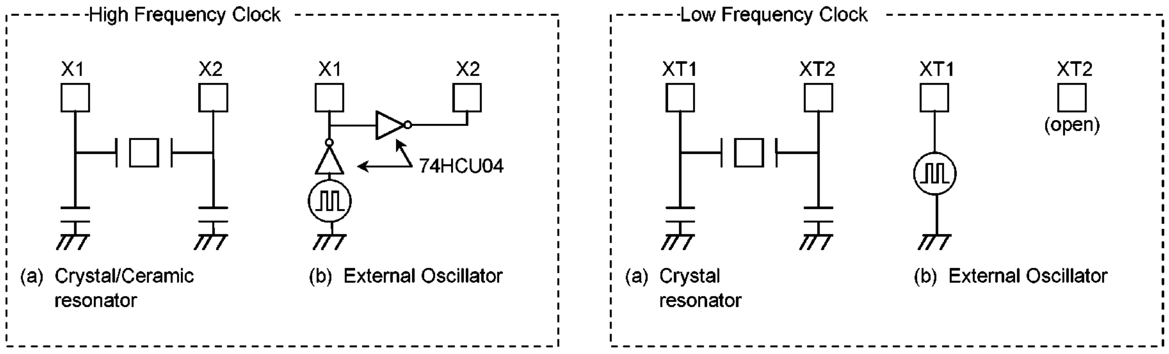


Figure 3.3.4 Examples of Resonator Connection

Note: Note on using the low frequency oscillation circuit.
In connecting the low frequency resonator to ports PB0 and PB1, it is necessary to make the following settings to reduce the power consumption.
(connecting with resonators) PBCR<PB0C, PB1C>="11", PB<PB0, PB1>="00"
(connecting with oscillators) PBCR<PB0C, PB1C>="11", PB<PB0, PB1>="10"

(1) Switching from NORMAL to SLOW mode

When the resonator is connected to X1, X2, or XT1, XT2 pin, the warm-up timer is used to change the operation frequency after getting stabilized oscillation.

The warm-up time can be selected by WDMOD<WARM>.

This starting and ending of warm-up timer are performed like the following example 1, 2 by program.

Note 1: The warm-up timer is also used as a watchdog timer. So, when it is used as a warm-up timer, the watchdog timer must be disabled.

Note 2: The case of using oscillator (not resonator) with stabilized oscillation, a warm-up timer is not need.

Note 3: The warm-up timer is operated by a oscillation clock. Therefore, warm-up time has an error.

Table 3.3.2 Warm-up Time

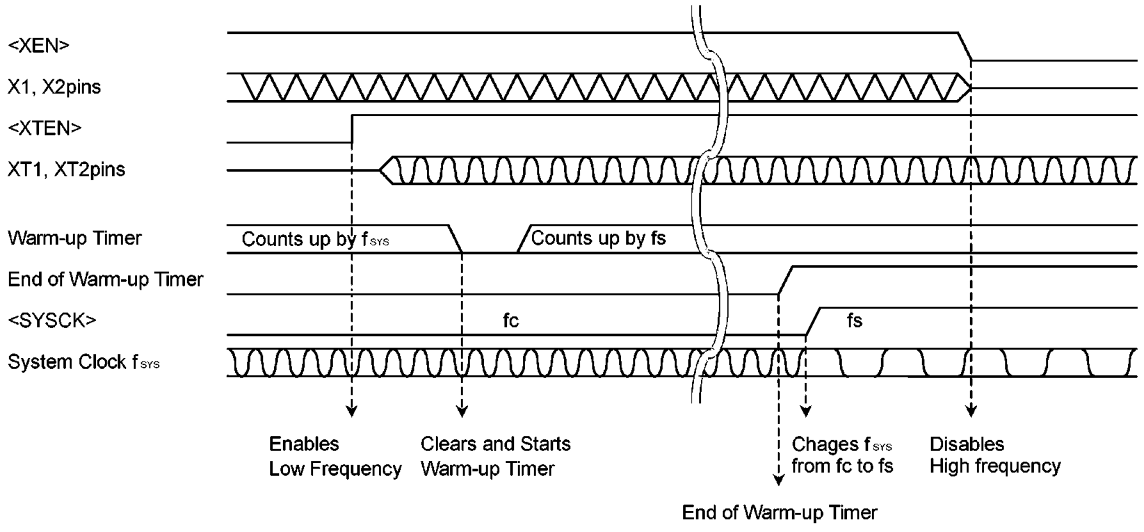
Warm-up Time WDMOD<WARM>	Change to NORMAL	Change to SLOW
0 (2^{14}/frequency)	1.024 (ms)	500 (ms)
1 (2^{16}/frequency)	4.096 (ms)	2000 (ms)

at fc = 16 MHz,
fs = 32.768 kHz

Clock Setting Example 1:

Changing from the high frequency (fc) to the low frequency (fs).

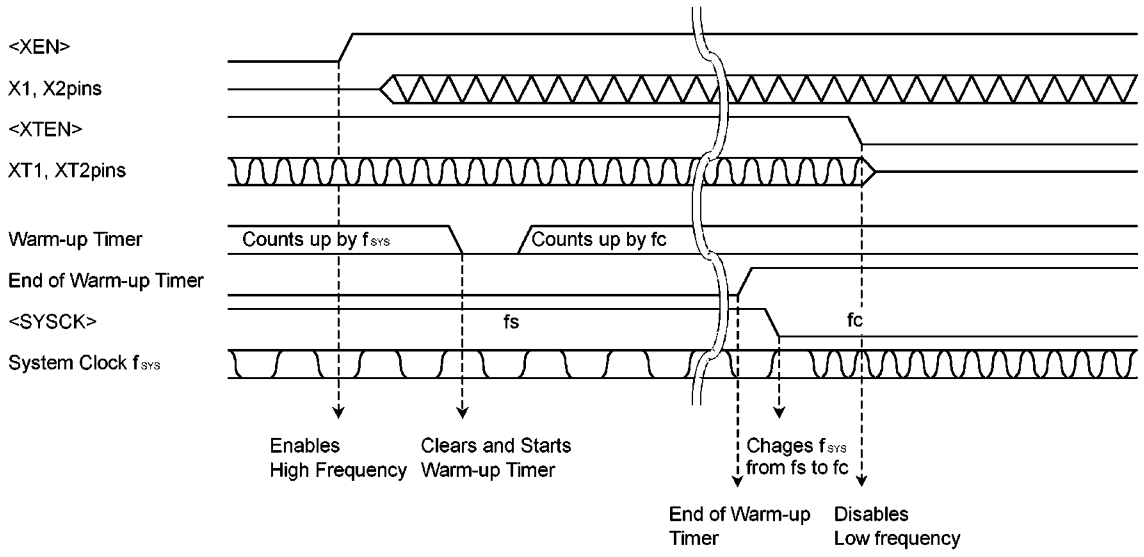
```
SYSCR0   EQU   006EH
SYSCR1   EQU   006FH
WDCR     EQU   005DH
WDMOD    EQU   005CH
         RES   7,(WDMOD)     ; }Disables Watchdog Timer.
         LD    (WDCR),B1H    ; }
         SET   4,(WDMOD)     ; Sets Warm-up Time to 2^16/fs.
         SET   6,(SYSCR0)    ; Enables Low Frequency (fs).
         SET   2,(SYSCR0)    ; Clears and stars Warm-up Timer.
WUP:     BIT   2,(SYSCR0)    ; }Detects End of Warm-up Timer.
         JR    NZ,WUP        ; }
         SET   3,(SYSCR1)    ; Changes fSYS from fc to fs.
         RES   7,(SYSCR0)    ; Disables High Frequency Oscillation.
         SET   7,(WDMOD)     ; Enables Watchdog Timer.
```



Clock Setting Example 2:

Changing from the low frequency (fs) to the high frequency (fc).

```
SYSCR0    EQU    006EH
SYSCR1    EQU    006FH
WDCR      EQU    005DH
WDMOD     EQU    005CH
          RES    7,(WDMOD)      ; } Disables Watchdog Timer.
          LD     (WDCR),B1H     ; }
          RES    4,(WDMOD)      ; Sets Warm-up Time to 2^14/fc.
          SET    7,(SYSCR0)     ; Enables High Frequency (fc).
          SET    2,(SYSCR0)     ; Clears and stars Warm-up Timer.
WUP:      BIT    2,(SYSCR0)     ; } Detects End of Warm-up Timer.
          JR     NZ,WUP         ; }
          RES    3,(SYSCR1)     ; Changes fSYS from fs to fc.
          RES    6,(SYSCR0)     ; Disables Low Frequency Oscillation.
          SET    7,(WDMOD)      ; Enables Watchdog Timer.
```



3.3.2 Timing Clock Generator

The timing clock generator generates sorts of system clock from the basic clock (fc or fs), providing for CPU core and peripheral hardwares.

(1) Architecture

The timing clock generator consists of the system clock generator and the time base counter (TBC) which generates system clock for peripheral hardwares. After resetting, the system clock is generated from high frequency clock (fc) (NORMAL mode). Both executing the instruction and operating the internal hardwares are synchronized by this system clock.

(2) Time Base Counter

The time base counter consists of a 20-bit up-counter counted by a basic clock divided-by 2 (fc/2 or fs/2), 16-bit data register and control register.

Figure 3.3.5 Shows the structure of the time-base counter (TBC).

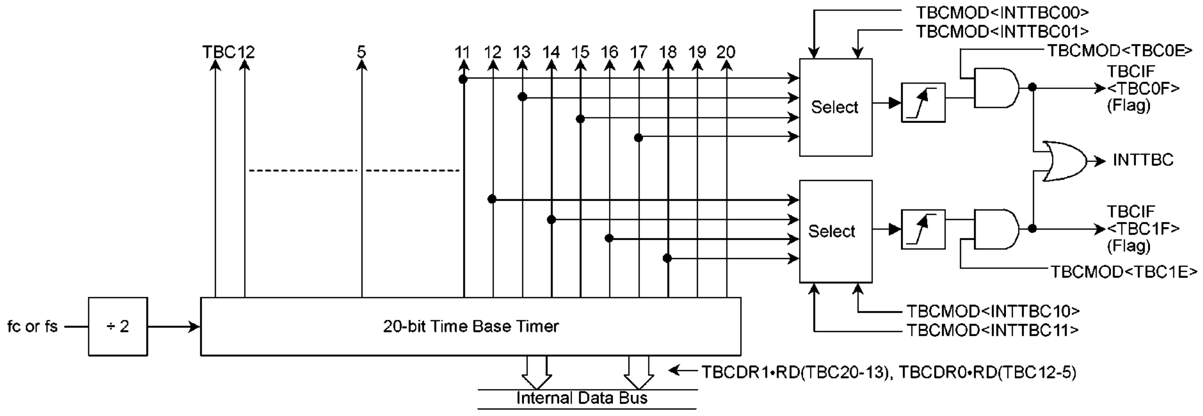


Figure 3.3.5 Shows the structure of the time-base counter (TBC).

Time Base Counter.Control Register

TBCMOD	7	6	5	4	3	2	1	0	
(0023H)			TBC1E	TBC0E	INT TBC11	INT TBC10	INT TBC01	INT TBC00	(Reset Value **00 0000)

TBC1E	INTTBC Interrupt Enable / Disable	00: INTTBC Interrupt Disable 01: INTTBC0 Interrupt Enable 10: INTTBC1 Interrupt Enable 11: INTTBC0 / INTTBC1 Interrupt Enable	R/W
TBC0E			
INTTBC11	INTTBC1 Interrupt Source Clock Selection	00: TBC12 01: TBC14 10: TBC16 11: TBC18	
INTTBC10			
INTTBC01	INTTBC0 Interrupt Source Clock Selection	00: TBC11 01: TBC13 10: TBC15 11: TBC17	
INTTBC00			

Time Base Counter Data register 0

TBCDR0	7	6	5	4	3	2	1	0	
(0024H)	TBC12	TBC11	TBC10	TBC9	TBC8	TBC7	TBC6	TBC5	(Reset Value 0000 0000) Read only

Time Base Counter Data Register 1

TBCDR1	7	6	5	4	3	2	1	0	
(0025H)	TBC20	TBC19	TBC18	TBC17	TBC16	TBC15	TBC14	TBC13	(Reset Value 0000 0000) Read only

Time Base Counter Interrupt Request Flag Register

TBCIF	7	6	5	4	3	2	1	0	
(0026H)			TBC1F	TBC0F					(Reset Value **00 ****)

TBC1E	INTTBC1 Interrupt Request Flag	0 (W): Clear 1 (W): (Inhibit) 0 (R): No interrupt request 1 (R): Interrupt request	R/W
TBC0F	INTTBC0 Interrupt Request Flag	0 (W): Clear 1 (W): (Inhibit) 0 (R): No interrupt request 1 (R): Interrupt request	

Note: The data at which TBCDRO is read is latched in TBCDR1.
When TBCDR1 is used, the datas must be read TBCDR0 and TBCDR1 in this order.

① Operation

The time-base counter outputs (TBC1 to TBC20) are used as clock source or timing data for Timer/Counter, Capture (CAP0/CAP1/CAP2), timing pulse generator (TPG) and other peripheral I/O blocks. The contents of time-base counter outputs TBC5 to TBC20 can be read by reading the time-base counter data registers, TBCDR0 and TBCDR1. Note that the data registers must be read in order of TBCDR0 and then TBCDR1.

Time-base counter interrupt requests (INTTBC) can be generated on the rising edges of counter outputs TBC11 to TBC18. The interrupt source is selected by the time-base counter control register TBCMOD <INTTBC11, INTTBC10, INTTBC01 and INTTBC00>. The INTTBC interrupt requests are comprised of two interrupt request signals INTTBC0 and INTTBC1 that are logical OR' ed to generate an interrupt request. Which interrupt is requested can be identified by reading the time base counter interrupt request flag register TBCIF <TBC0F> and <TBC1F>.

The INTTBC0 flag <TBC0F> and INTTBC1 flag <TBC1F> can be cleared by writing "0" in the register.

Table 3.3.3 lists the interval time of time-base counter outputs.

Table 3.3.3 interval time of Time-base Counter (f = fc)

	TBC1	TBC2	TBC3	TBC4	TBC5	TBC6	TBC7	TBC8	TBC9	TBC10
Interval Time [s]	$2^2/f$	$2^3/f$	$2^4/f$	$2^5/f$	$2^6/f$	$2^7/f$	$2^8/f$	$2^9/f$	$2^{10}/f$	$2^{11}/f$
at 16 MHz [μs]	0.25	0.5	1.0	2.0	4.0	8.0	16.0	32.0	64.0	128.0

TBC11	TBC12	TBC13	TBC14	TBC15	TBC16	TBC7	TBC18	TBC19	TBC20
$2^{12}/f$	$2^{13}/f$	$2^{14}/f$	$2^{15}/f$	$2^{16}/f$	$2^{17}/f$	$2^{18}/f$	$2^{19}/f$	$2^{20}/f$	$2^{21}/f$
256	512	1024	2048	4096	8192	16384	32768	65536	131072

3.3.3 Real Time Counter (RTC)

The TMP93CW76F has the real time counter (RTC) which generates a periodic interrupt request. The RTC is controlled by System Control Register2 (SYSCR2).

The RTC is a 17bit binary counter and its clock source is selected either low frequency clock (fs) or system clock (f_{SYS}/2) by <RTCCK>. To start/stop the counter is controlled by <RTCST>.

The period of interrupt request INTRTC is selected from 3 types by setting <RTCIS1, RTCIS0>.

Table 3.3.4 shows the period of interrupt request INTRTC.

SYSCR2 (006CH)

	7	6	5	4	3	2	1	0
bit Symbol				RTCCK	RTCST	RTCIS1	RTCIS0	
Read/Write	(R/W)	(R/W)		R/W	R/W	R/W		
After reset	(0)	(0)		0	0	0		
Function	Always set to "0"			RTC clock source select 0: fs (32 kHz) 1: fc/4 or fs/4	RTC count 0: Stop & Counter Clear 1: Start	Interval time cotrol of RTC interrupt 00: fc/2^{17} or fs/2^{15} fs/2^{17} 01: fc/2^{18} or fs/2^{18} fs/2^{18} 10: fc/2^{16} or fs/2^{14} fs/2^{16} 11: Inhibit [Hz]		

Note: When SYSCR2<RTCCR> is set to "1", RTC input clock is assigned to fc/4 or fd/4 depending on the peration mode.

Table 3.3.4 INTRTC Interrupt Interval

<RTCCK>	<RTCIS>	INTRTC interrupt interval
0 (fs = 32.768 kHz)	00	1 s
	01	2 s
	10	0.5 s
1 (f_{SYS} = fc/2, fc = 16 MHz)	00	16.384 ms
	01	32.768 ms
	10	8.192 ms

3.3.4 Standby Controller

(1) Halt mode

When the HALT instruction is executed, the operating mode changes RUN, IDLE2, IDLE1 or STOP mode depending on the contents of watchdog timer mode register WDMOD<HALTM1,0>. Figure 3.3.6 shows the alternative states of watchdog timer mode registers.

WDMOD (005CH)

	7	6	5	4	3	2	1	0
bit Symbol	WDTE	WDTP1	WDTP0	WARM	HALTM1	HALTM0	RESCR	DRVE
Read/Write	R/W							
After reset	1	0	0	0	0	0	0	0
Function	Watchdog timer 0: disable 1: enable	Watchdog timer detect time selection 00: $2^{15}/f_{SYS}$ 01: $2^{17}/f_{SYS}$ 10: $2^{19}/f_{SYS}$ 11: $2^{21}/f_{SYS}$		Warm-Up Timer 0: 2^{14}/ frequency inputted 1: 2^{16}/ frequency selection	Standby mode 00: RUN mode 01: STOP mode 10: IDLE1 mode 11: IDLE2 mode		0: — 1: Connect internally WDT out to Reset pin	STOP mode pin control 1: drive pins in STOP mode

Pin state control in STOP mode

0	I/O off
1	Retains the state befor halt

Halt mode setting

00	RUN mode (only CPU stop)
01	STOP mode (all circuits stop)
10	IDLE1 mode (only oscillator operating)
11	IDLE2 mode (partial I/O operating)

Warm-up time selection at returning from the stop mode (seeTable3-3-5)

0	2^{14}/ select clock frequency
1	2^{16}/ select clock frequency

Figure 3.3.6 Watchdog timer mode register

The features of RUN, IDLE2, IDLE1 and STOP modes are as follows.

① RUN: Only the CPU halts; power consumption remains unchanged.

② IDLE2: The built-in oscillator and the specified I/O operates.
The Power Consumption is redced to 1/2 than that during NORMAL operation.

③ IDLE1: Only the built-in oscillator operates, while all other built-in circuits stop. The Power Consumption is reduced to 1/5 or less than that during NORMAL operation.

④ STOP: All internal circuits including the built-in oscillator stop. This greatly reduces power consumption.

The operations in the halt state is described in Table 3.3.5.

Table 3.3.5 I/O Operation During Halt mode

Halt mode	RUN	IDLE2	IDLE1	STOP
WDMOD<HALTM1,0>	00	11	10	01
CPU	Stop	Stop	Stop	Stop
I/O port	Keep the state when the "HALT" instruction was executed.	Keep the state when the "HALT" instruction was executed.	Keep the state when the "HALT" instruction was executed.	See Table 3.3.8
8-bit Timer(TC0)	Operate	Operate	Stop	Stop
16-bit Timer (TC 1,2,3,4,5)	Operate	Operate	Stop	Stop
Dual Clock	Operate	Operate	Stop	Stop
Watchdog Timer	Operate	Operate	Stop	Stop
Interrupt controller	Operate	Operate	Stop	Stop
SIO	Operate	Operate	Stop	Stop
I^{2}CBUS	Operate	Operate	Stop	Stop
8-bit PWM	Operate	Operate	Stop	Stop
14-bit PWM	Operate	Operate	Stop	Stop
Timing Pulse Generator (TPG 0,1)	Operate	Operate	Stop	Stop
Color Rotary	Operate	Operate	Stop	Stop
VISS/VASS	Operate	Operate	Stop	Stop
CSYNC	Operate	Operate	Stop	Stop
PV/PH	Operate	Operate	Stop	Stop
Capture input (Capture1/2)	Operate	Operate	Stop	Stop
Capture 0	Operate	Operate	Stop	Stop
Remote Control Input (RMTIN)	Operate	Operate	Stop	Stop
Time Base Counter (TBC)	Operate	Operate	Stop	Stop
A/D Converter	Operate	Stop	Stop	Stop
Real Time Counter (RTC)	Operate	Operate	Stop	Stop

(2) How to Release the Halt mode

These HALT states can be released by resetting or requesting an interrupt. The HALT release sources are determined by the combinations between the states of interrupt mask register <IFF2 to 0> and the halt modes. The details for releasing the HALT status are shown in Table 3.3.6.

• Released by requesting an interrupt

The operating released from the HALT mode depends on the interrupt enabled status. When the interrupt request level set before executing the HALT instruction exceeds the value of the interrupt mask register, the interrupt due to the source is processed after releasing the halt mode, and CPU starts executing an instruction that follows the HALT instruction. When the interrupt request level set before executing the HALT instruction is less than the value of the interrupt mask register, releasing the halt mode is not executed. (in non-maskable interrupts, interrupt processing is processed after releasing the halt mode regardless of the value of the mask register.)

However only for INT0 and INT1 interrupts, even if the interrupt request level set before executing the HALT instruction is less than the value of the interrupt mask register, releasing the halt mode is executed. In this case, interrupt processing is not processed, and CPU starts executing the instruction next to the HALT instruction, but the interrupt request flag is held at "1".

- Release by resetting

  Releasing all halt status is executed by resetting.

  When the STOP mode is released by RESET, it is necessary enough resetting time (3 ms or more) to set the operation of the oscillator to be stable.

  When releasing the halt mode by resetting, the internal RAM data keeps the state before the "HALT" instruction is executed. However the other setting contents are initialized. (Releasing due to interrupts keep the state before the "HALT" instruction is executed.)

Table 3.3.6 Halt releasing source and Halt releasing operation

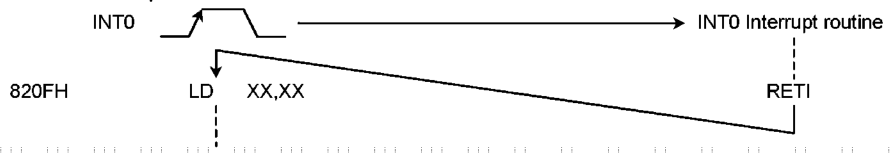
Interrupt receiving status			Interrupt enable (Interrupt level)≥(Interrupt mask)		Interrupt disable (Interrupt level)<(Interrupt mask)	
Halt mode			IDEL	STOP	IDEL	STOP
Halt releasing source	Interrupt	INTWD	×	×	—	—
		INT0,1	⊙	⊙*1	O	O*1
		INTCAP0,1	⊙	×	×	×
		INTTTG0,1	⊙	×	×	×
		INTI2CB	⊙	×	×	×
		INTTBC	⊙	×	×	×
		INTSIO	⊙	×	×	×
		INTVA	⊙	×	×	×
		INT2,3,4	⊙	×	×	×
		INTT0-5	⊙	×	×	×
		INTAD	×	×	×	×
		INTRTC	⊙	×	×	×
	Reset Input		⊙	⊙	⊙	⊙

⊙: After releasing the halt mode, CPU starts interrupt processing (RESET initalizes LSI)
O: After releasing the halt mode, CPU starts executing an instruction that follows the HALT instruction.
×: It can not be used to release the halt mode.
—: This combination type does not exist because the priority level (interrupt request level) of non-maskable interrupts is fixed to highest priority level "7".
*1: Releasing the halt mode is executed after passing the warming-up time.
Note: When releasing the halt mode is executed by INT0 or INT1 interrupt of the level mode in the interrupt enabled status, hold level "H" until starting interrupt processing . If level "L" is set before starting interrupt processing, interrupt processing is incorrently started.

(Example releasing "RUN" mode)

INT0 interrupt releases HALT state when the IDLE mode is on.

```
Address       :
8200H       LD      (IIMC0),01H      ;  INT0 interrupt input enable
8203H       LD      (IIMC1),00H      ;  Selects interrupt rising edge for INT0
8206H       LD      (INT0CP1),06H    ;  Sets interrupt level to '6' for INT0
8209H       EI      5                ;  Sets interrupt level to '5' for CPU
820BH       LD      (WDMOD),08H      ;  Sets HALT mode to 'IDLE'
820EH       HALT                     ;  halts CPU
              :
```



When halt is released by reset, the states (including those of the internal RAM) before halt state was entered can be maintained. However, if the HALT instruction is executed within the internal RAM, the contents of the RAM way not be maintained. In this case, we recommend releasing the halt state using INT0.

(3) Operation

① RUN mode

In the RUN mode, the system clock continues to operate even after a HALT instruction is executed. Only the CPU stops executing the instruction. In the HALT state, an interrupt request is sampled with the falling edge of the internal "CLK" signal.

Releasing the RUN mode is executed by the external/internal interrupts. (See Table 3.3.6 Halt releasing source and Halt releasing operation.)

Figure 3.3.7 shows the interrupt timing for releasing the HALT state by interrupts in the RUN/IDLE2 mode.

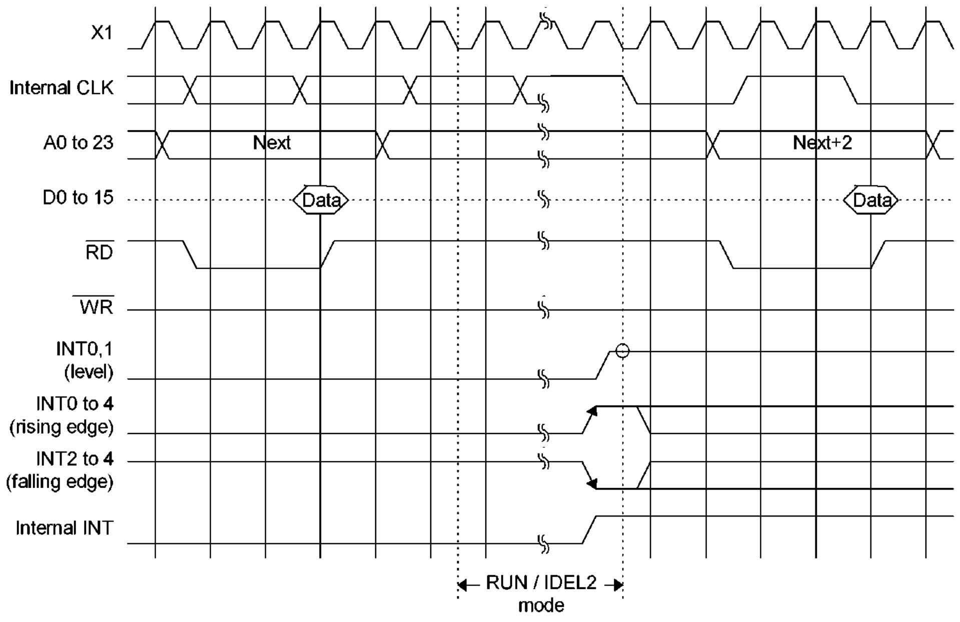


Figure 3.3.7 Timing Chart for Releasing the HALT State by Interrupt in RUN/IDLE2 modes

② IDLE2 mode

In the IDLE2 mode, the system clock is supplied to only specific internal I/O devices, and the CPU stops executing the current instruction.

In the IDLE2 mode, the HALT state is released by an interrupt with the same timing as in the RUN mode. The IDLE2 mode is released by external/internal interrupt, except INTWD/INTAD interrupts. (See table 3.3.6 Halt releasing source and Halt releasing operation.)

In the IDLE2 mode, the watchdog timer should be disabled before entering the halt status to prevent the watchdog timer interrupt occurring just after releasing the halt mode.

③ IDLE1 mode

In the IDLE1 mode, only the internal oscillator operates. The system clock in the MCU stops.

In the HALT state, and interrupt request is sampled aynchronunsly with the system clock, however the HALT release (restart of operation) is performed synchronously with it.

IDLE1 mode is released by external interrupts (INT0, INT1). (See table 3.3.6 Halt releasing source and Halt releasing operation.)

Figure 3.3.8 illustrates the timing for releasing the HALT state by interrupts in the IDLE1 mode.

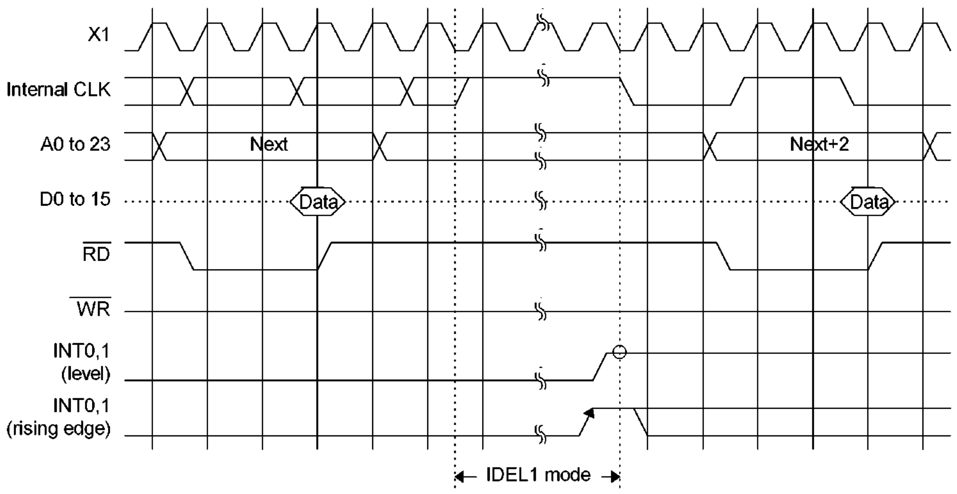


Figure 3.3.8 Timing Chart of HALT Released by Interrupts in IDLE1 Mode

④ STOP mode

The STOP mode is selected to stop all internal circuits including the internal oscillator. The pin status in the STOP mode depends on setting of a bit in the watchdog timer mode register WDMOD<DRVE>. (See Figure 3.3.6 for setting of WDMOD <DRVE>.) Table 3.3.8 summarizes the state of these pins in the STOP mode.

The STOP mode is released by external interrupts (INT0, INT1). When the STOP mode is released, the system clock output starts after warm-up time required to attain stable oscillation. The warm-up time can be set using WDMOD<WARM>. See the example of warm-up time (Table 3.3.7).

Figure 3.3.9 illustrates the timing for releasing the HALT state by interrupts during the STOP mode.

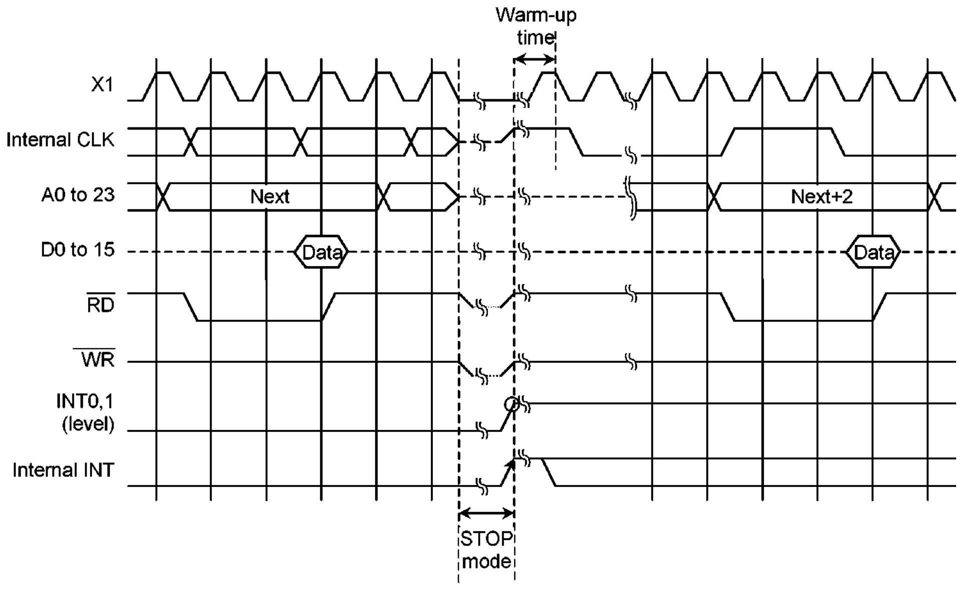


Figure 3.3.9 Timing Chart of HALT State Release by Interrupts in STOP Mode

Table 3.3.7 The example of Warm-up time after releasing the stop mode

Clock operation frequency after the stop mode	Warm-up time [ms]		Clock frequency
	WDMOD<WARM> = 0	WDMOD<WARM> = 1	
fc	1.024	4.096	fc = 16 MHz
fs	500	2000	fs = 32.768 kHz

How to calculate the warm-up time

WDMOD<WARM> = "0": Clock operation frequency after the 2^{14}/STOP mode

WDMOD<WARM> = "1": Clock operation frequency after the 2^{16}/STOP mode

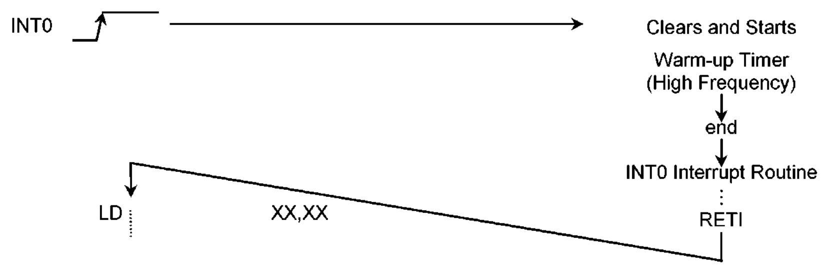
The NORMAL / SLOW mode selection is possible after released STOP mode.

This is selected by SYSCR0 <RSYSCK>. Therefore, Setting to <RSYSCK>, <RXEN>, <RXTEN> is necessary before "HALT" instruction is executed.

(Setting Example)

The STOP mode is entered when the low frequency (fs) operates, and after that high frequency operates after releasing by INT0.

```
Address
            SYSCR0 EQU          006EH
            SYSCR1 EQU          006FH
            WDMOD  EQU          005CH
8FFDH              LD           (SYSCR1),08H       ; fsys = fs/2
9000H              RES          4,(WDMOD)          ; Sets Warm-up Time to 2^14/fc
9002H              LD           (SYSCR0),-11000- -B ; Operates High Frequency after relesed.
                                  (Note)           -: no change
9005H              HALT
```



```
9006H              LD           XX,XX
```

Note: When different modes are used before and after STOP mode as the above mentioned, there is possible to release the HALT mode without changing the operation mode by acceptance of the halt release interrupt request during execution of "HALT" instruction (during 8 states). In the system which accepts the interrupts during execution "HALT" instruction, set the same operation mode before and after the STOP mode.

Table 3.3.8 Pin states in STOP mode

Pin Name	I/O	TMP93CW76F	
		WDMOD <DRVE> = "0"	WDMOD <DRVE> = "1"
CLK	Output	Hz	"H" Level Output
X1	Input	—	—
X2	Output	"H" Level Output	"H" Level Output
PB0/PB1 XT1 XT2	Input/Output Input Output	— /Hz — "H" Level Output	- /Output — "H" Level Output
P40 to P47/AIN2 to AIN9	Input	—	—
P00 to P07, P10 to P17, P20 to P27, P50 to P54, P73 to P76, P80 to P87, P90 to P97, PA0, PA3	Input Output	— Hz	— Output
PWM0, 1	Output	Hz	Output
PE0 to PE7, PF0 to PF5	Input Output	— Hz	— Output
PD0, PD1,PC0 to PC7	Output	Hz	Output
P53 / INT1, P54 / INT0	Input Output	Input Hz	Input Output
$\overline{\text{TEST}}$	Input	"L" Level fixed	"L" Level fixed
$\overline{\text{RESET}}$	Input	Input	Input
ADREF	Input	Open status is available	Open status is available

—	: Inputs are not accepted.
Input	: Input gate in operation. Fix input voltage level to 0 or 1 so that the input pin stays constant.
Output	: Output state
Hz	: High Impedance

3.4 Interrupts

TLCS-900 interrupts are controlled by the CPU interrupt mask flip-flop <IFF2 to 0> and the built-in interrupt controller.

Altogether the TMP93CW76F has the following 30 interrupt sources:

Internal interrupts 25
- Software interrupts: 8
  (Illegal instruction execution is included: 1)
- Interrupts from built-in I/Os: 17

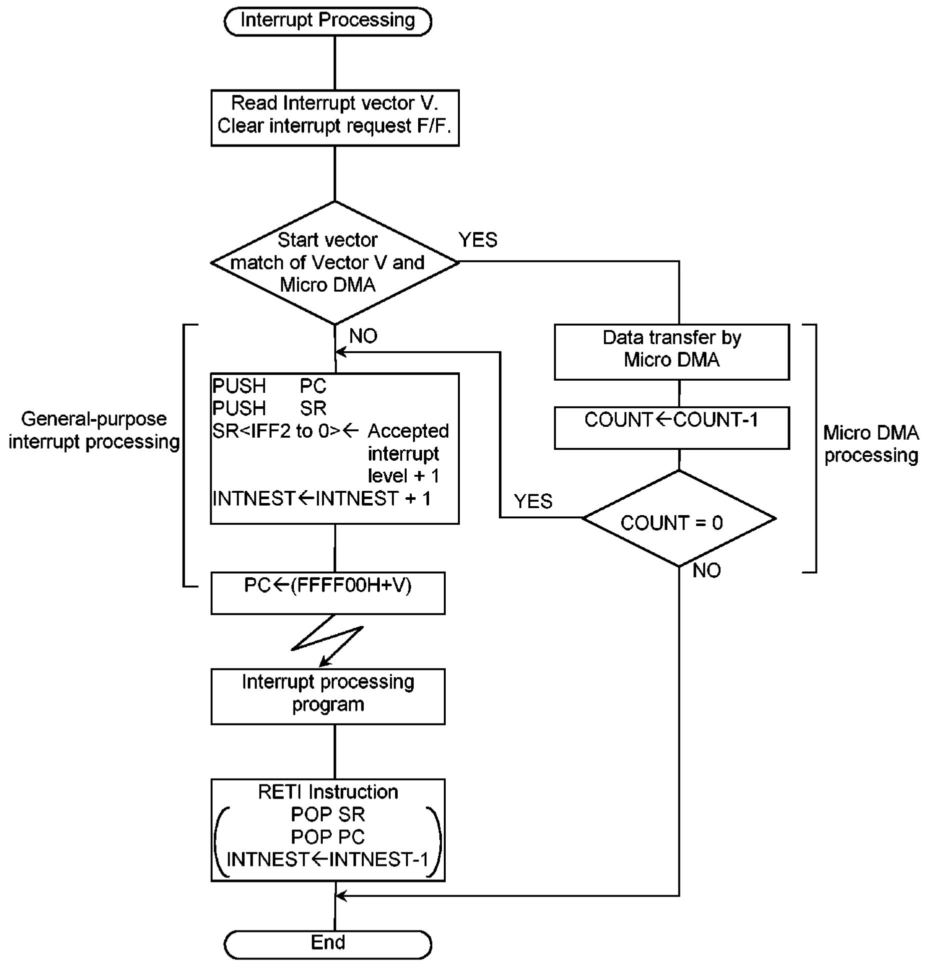
External interrupt............. 5
- External pins (INT0, INT1, INT2 to INT4)

A fixed individual interrupt vector number is assigned to each interrupt source; six levels of priority can also be assigned to each maskable interrupt. Non-maskable interrupts have a fixed priority of 7.

When an interrupt is generated, the interrupt controller sends the value of the priority of the interupt source to the CPU. When more than one interrupt is generated simultaneously, the interrupt controller sends the value of the highest priority (7 for non-maskable interrupts is the highest) to the CPU.

The CPU compares the value of the priority sent with the value in the CPU interrupt mask register (IFF2 to 0). If the value is greater than that the CPU interrupt mask register, the interrupt is accepted. The value in the CPU interrupt mask register (IFF2 to 0) can be changed using the EI instruction (Executing EI n changes the contents of <IFF2 to 0> to n). For example, programming EI 3 enables acceptance of maskable interrupts with a priority of 3 or greater, and non-maskable interrupts which are set in the interrupt controller. The DI instruction (<IFF2 to 0> = 7) operates in the same way as the EI 7 instruction. Since the priority values for maskable interrupts are 0 to 6, the DI instruction is used to disable acceptance of maskable interrupts. The EI instruction becomes effective immediately after execution (With the TLCS-90, the EI instruction becomes effective after execution of the subsequent instruction).

In addition to the general-purpose interrupt processing mode described above, there is also a Micro DMA processing mode . Micro DMA is a mode used by the CPU to automatically transfer byte or word data. It enables the CPU to process interrupts such as data saves to built-in I/Os at high speed.



Interrupt Processing
Read Interrupt vector V.
Clear interrupt request F/F.
Start vector match of Vector V and Micro DMA
YES
NO
Data transfer by Micro DMA
COUNT←COUNT-1
COUNT = 0
YES
NO
Micro DMA processing
General-purpose interrupt processing
PUSH PC
PUSH SR
SR<IFF2 to 0>← Accepted interrupt level + 1
INTNEST←INTNEST + 1
PC←(FFFF00H+V)
Interrupt processing program
RETI Instruction
POP SR
POP PC
INTNEST←INTNEST-1
End

Figure 3.4.1 Interrupt Processing Flowchart

3.4.1 General-Purpose Interrupt Processing

When accepting an interrupt, the CPU operates as follows. Software interrupt and illegal instruction execution interrupt execute (2), (4) and (5) except (1) and (3).

(1) The CPU reads the interrupt vector from the interrupt controller. When more than one interrupt with the same level is generated simultaneously, the interrupt controller generates interrupt vectors in accordance with the default priority (which is fixed as follows: the smaller the vector value, the higher the priority), then clears the interrupt request.

(2) The CPU pushes the program counter and the status register to the system stack area (area indicated by the system mode stack pointer (XSP)).

(3) The CPU sets a value in the CPU interrupt mask register <IFF2 to 0> that is higher by 1 than the value of the accepted interrupt level. However, if the value is 7, 7 is set without an increment.

(4) The CPU increments the INTNEST (Interrupt Nesting Counter).

(5) The CPU jumps to address stored at FFFF00H + interrupt vector, then starts the interrupt processing routine.

The following diagram shows all the above processing state number.

Bus Width of Stack Area	Bus Width of Interrupt Vector Area	Interrupt processing state number
16 bit	16 bit	25

To return to the main routine after completion of the interrupt processing, the RETI instruction is usually used. Executing this instruction restores the contents of the program counter and the status registers and decrements INTNEST (Interrupt Nesting Counter).

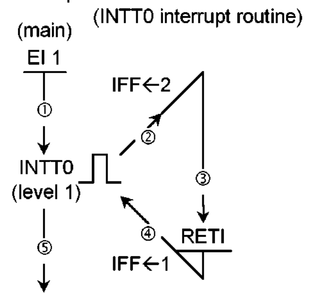
Though acceptance of non-maskable interrupts cannot be disabled by program, acceptance of maskable interrupts can. A priority can be set for each source of maskable interrupts. The CPU accepts an interrupt request with a priority higher than the value in the CPU mask register <IFF2 to 0>. The CPU mask register <IFF2 to 0> is set to a value higher by 1 than the priority of the accepted interrupt. Thus, if an interrupt with a level higher than the interrupt being processed is generated, the CPU accepts the interrupt with the higher level, causing interrupt processing to nest.

The interrupt request with a priority higher than the accepted now interrupt during the CPU is processing above (1) to (5) is accepted before the 1'st instruction in the interrupt processing routine, causing interrupt processing to nest. (This is the same case of over lapped each Non-Maskable interrupt (level “7").) The CPU does not accept an interrupt request of the same level as that of the interrupt being processed.

The CPU mask register <IFF2 to 0> is initialized "7" after reset. It is disable maskable interrupt.

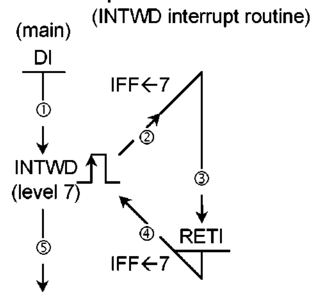
The following (1) to (5) show a flowchart of interrupt processing.

(1) Maskable interrupt



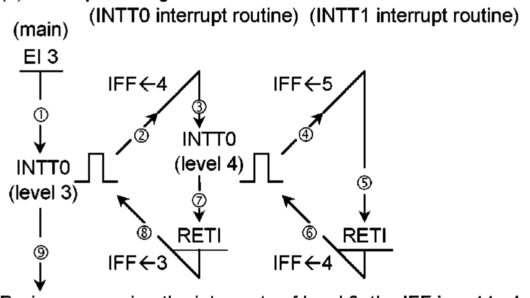
During execution of the main program, the CPU accepts an interrupt request. The CPU increments the IFF so that the interrupts of level 1 are not accepted during processing the interrupt routine.

(2) Non-maskable interrupt



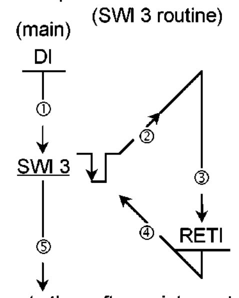
DI instruction is executed in the main program, so that the interrupts of only level 7 are accepted. The CPU does not increment the IFF even if the CPU accepts an interrupt request of level 7.

(3) Interrupt nesting



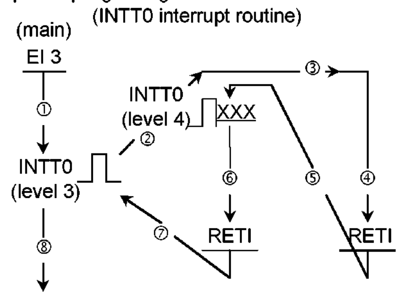
During processing the interrupts of level 3, the IFF is set to 4. When an interrupt with a level higher than level 4 is generated, the CPU accepts the interrupt processing to nest.

(4) Software interrupt



The CPU accepts the software interrupt request during DI status(IFF = 7) because of the level 7. The IFF is not changed by the software interrupts.

(5) Interrupt sampling timing



If an interrupt with a level higher than the interrupt being processed is generated, the CPU accepts the interrupt with the higher level. The program counter which returns at ⑤ is the start address of INTT0 interrupt routine.

(example)___(underline)	: Instruction
①,②,···	: Execution flow

The address FFFF00H to FFFFFFH (256 byte) of the TMP93CW76F are assigned for interrupt vector area.

Table 3.4.1 TMP93CW76F Interrupt Table

Default priority	Type	Interrupt source	Vector value "V"	Address refer to vector	High-speed micro DMA start vector	Input Level/ Edge
1	non-	Reset,or SWI0 instruction	0000H	FFFF00H	—	
2	maskable	SWI1 instruction	0004H	FFFF04H	—	
3		INTUNDEF:illegal	0008H	FFFF08H	—	
4		SWI3 instruction	000CH	FFFF0CH	—	
5		SWI4 instruction	0010H	FFFF10H	—	
6		SWI5 instruction	0014H	FFFF14H	—	
7		SWI6 instruction	0018H	FFFF18H	—	
8		SWI7 instruction	001CH	FFFF1CH	—	
9		(Reserved: NMI)	0020H	FFFF20H	—	
10		INTWD: watchdog timer	0024H	FFFF24H	09H	Edge
11	maskable	INT0: External interrupt input 0	0028H	FFFF28H	0AH	Edge
12		INTCAP1: Capture 1 interrupt	002CH	FFFF2CH	0BH	Level
13		INTCAP0: Capture 0 interrupt	0030H	FFFF30H	0CH	Level
14		INTTPG0: Timing Pulse Generator 0 interrupt	0034H	FFFF34H	0DH	Edge (Note1)
15		INTTPG1: Timing Pulse Generator1 interrupt	0038H	FFFF38H	0EH	Edge
16		INTI2CB: I^{2}CBUS interrupt	003CH	FFFF3CH	0FH	Edge
17		INTTBC: Time Base Counter interrupt	0040H	FFFF40H	10H	Edge
18		INTVA: VISS/VASS detection	0044H	FFFF44H	11H	Edge
19		INT1: External interrupt input 1	0048H	FFFF48H	12H	Edge/Level
20		INT2/INTSIO: External input 2/SIO interrupt	004CH	FFFF4CH	13H	Edge/Level (Note2)
21		INT3: External interrupt input 3	0050H	FFFF50H	14H	Edge
22		INT4: External interrupt input 4	0054H	FFFF54H	15H	Edge
23		INTT0: 8-bit Timer 0 (TC0)	0058H	FFFF58H	16H	Edge
24		INTT1: 16-bit Timer 1 (TC1)	005CH	FFFF5CH	17H	Edge
25		INTT2: 16-bit Timer 2 (TC2)	0060H	FFFF60H	18H	Edge
26		INTT3: 16-bit Timer 3 (TC3)	0064H	FFFF64H	19H	Edge
27		INTT4: 16-bit Timer 4 (TC4)	0068H	FFFF68H	1AH	Edge
28		INTT5: 16-bit Timer 5 (TC5)	006CH	FFFF6CH	1BH	Edge
29		INTAD: A/D conversion completion	0070H	FFFF70H	1CH	Level
30		INTRTC: Real Time Counter (RTC)	0074H	FFFF74H	1DH	Edge
— to —		(Reserved) to (Reserved)	0078H to 00FCH	FFFF78H to FFFFFCH	1EH to 3FH	

Note1 : The INTTPG0 signal is generated as a level signal (FIFO empty) or as an edge signal (TPG03 output), but the interrupt controller receives the INTTPG0 only as an edge signal. When the INTTPG0 is used for a FIFO empty interrupt (a level signal), the interrupt controller also leaves a request flag (Flip/Flop) after clearing FIFO empty by setting next TPG0 data in an interrupt routine. Therefore, in this case, the INTTPG0 request flag has to be cleared before executing RETI instruction.

Note2 : Either INT2 or INTSIO is selected. INT2 is generated as an edge signal and INTSIO is generated as a level signal.

Setting to Reset / Interrupt Vector

① Reset Vector

FFFF00H	PC(7 to 0)
FFFF01H	PC(15 to 8)
FFFF02H	PC(23 to 16)
FFFF03H	XX

The vector base addresses are depended on the products.

Type No.	Vector base address	PC setting sequence after reset
TMP93CW76F	FFFF00H	PC(7 to 0) ← address FFFF00H PC(15 to 8) ← address FFFF01H PC(23 to 16) ← address FFFF02H

② Interrupt Vector (except Reset Vector)

Address refer to vector	+0	PC(7 to 0)
	+1	PC(15 to 8)
	+2	PC(23 to 16)
	+3	XX

XX: don’t care

(Setting Example)

Sets the Reset Vector: FF0000H, INTWD Vector: FF9ABCH, INTAD Vector: FE3456H.

```
ORG     FFFF00H
DL      FF0000H     ; Reset = FF0000H

ORG     FFFF24H
DL      FF9ABCH     ; INTWD = FF9ABCH

ORG     FFFF84H
DL      FE3456H     ; INTAD = FE3456H

ORG     FF0000H
LD      A,B                     Note:
        ⋮                          ORG,DL are Assembler Directives.
ORG     FF9ABCH                  ( ORG: control location counter
LD      B,C                      ( DL: defines long word (32-bits) data

ORG     FE3456H
LD      C,A
        ⋮
```

3.4.2 Micro DMA

In addition to the conventional interrupt processing, the TMP93PW76 also has a Micro DMA function. When an interrupt is accepted, in addition to an interrupt vector, the CPU receives data indicating whether processing is Micro DMA mode or general-purpose interrupt. If Micro DMA mode is requested, the CPU performs Micro DMA processing.

The TMP93PW76 can process at very high speed because it has transfer parameters in dedicated registers in the CPU. Since those dedicated registers are assigned as CPU control registers, they can only be accessed by the LDC instruction.

(1) Micro DMA operation

Micro DMA operation starts when the accepted interrupt vector value matches the Micro DMA start vector value. The Micro DMA has four channels so that it can be set for up to four types of interrupt source.

When a Micro DMA interrupt is accepted, data is automatically transferred from the transfer source address to the transfer destination address set in the control register, and the transfer counter is decremented. If the value in the counter after decrementing is other than 0, Micro DMA processing is completed; if the value in the counter after decrementing is 0, general-purpose interrupt processing is performed.

There are two data transfer modes: one-byte mode and one-word mode. Incrementing, decrementing, and fixing the transfer source/destination address after transfer can be done in both modes. Therefore data can easily be transferred between I/O and memory and between I/Os. For details of transfer modes, see the description of transfer mode registers.

The transfer counter has 16 bits, so up to 65536 transfers (the maximum when the initial value of the transfer counter is 0000H) can be performed for one interrupt source by Micro DMA processing.

When the transfer counter is decremented to "0" after data is transferred with micro DMA, general-purpose interrupt processing is performed. After processing the general-purpose interrupt, starting the interrupts of the same channel restarts the transfer counter from 65536. If necessary, reset the transfer counter.

Interrupt sources processed by Micro DMA processing are those with the Micro DMA start vectors listed in Table 3.4.1.

The following timing chart is a Micro DMA cycle of the Transfer Address INC rement mode.

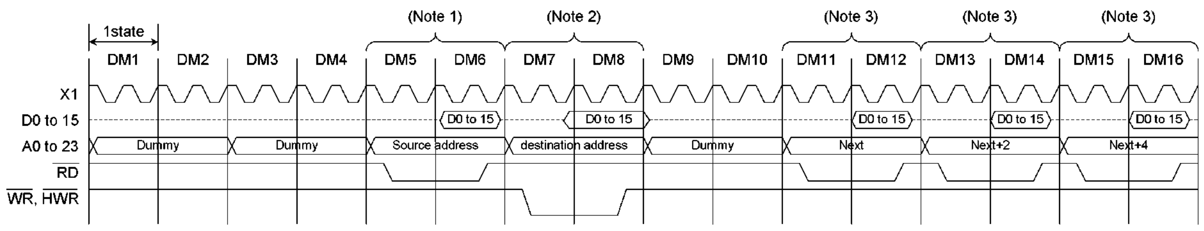


Figure 3.4.2 Micro DMA cycle (COUNT ≠ 0)

Note 1: These 2 states are added in the case that the bus width of the source address area is 8 bits.

Note 2: These 2 states are added in the case that the bus width of the destination address area is 8 bits.

Note 3: This may be a dummy cycle with an instruction queue buffer.

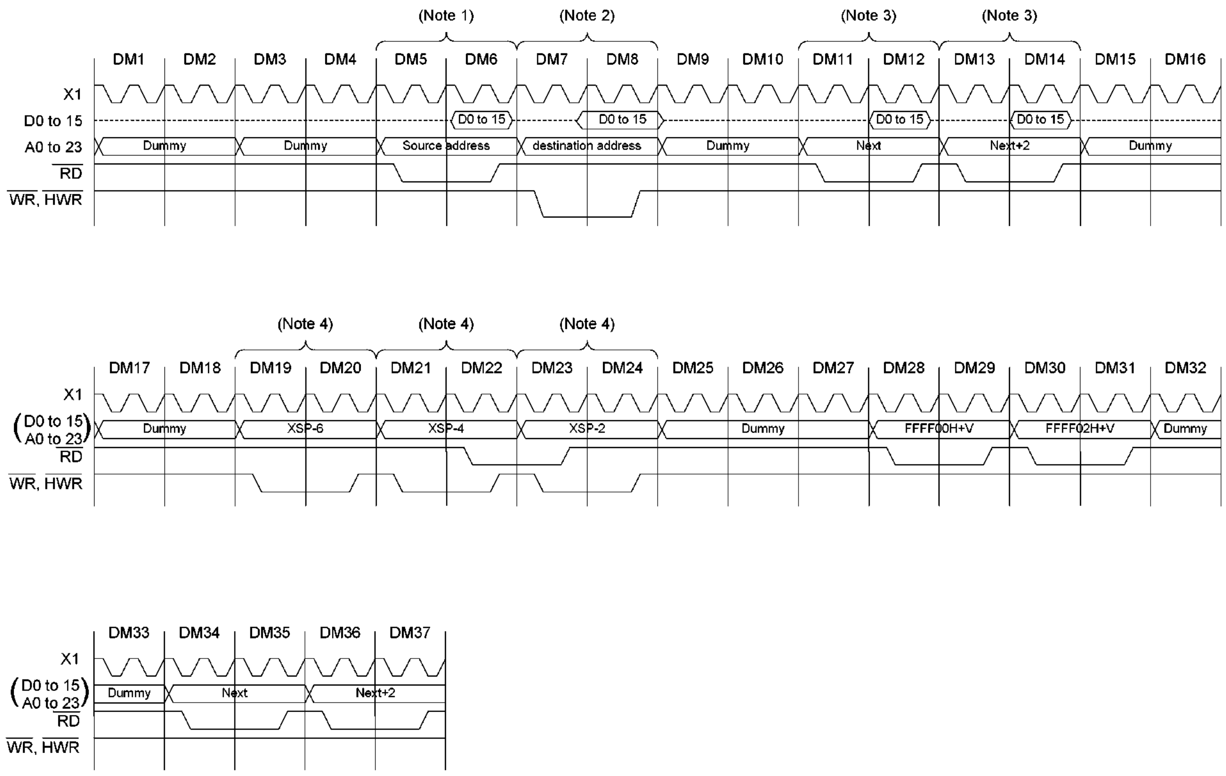


Figure 3.4.3 Micro DMA cycle (COUNT = 0)

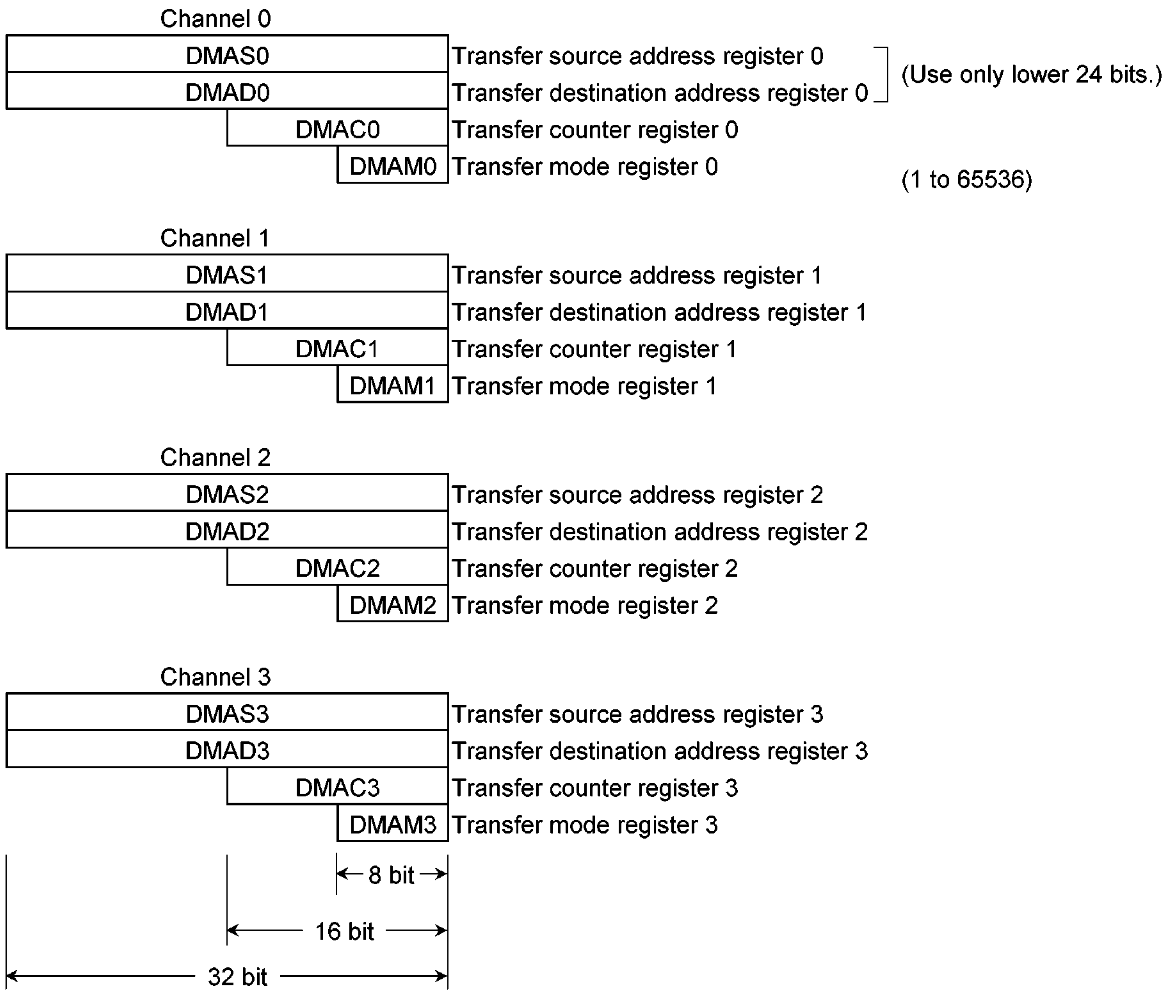
Note 1: These 2 states are added in the case that the bus width of the source address area is 8 bits.

Note 2: These 2 states are added in the case that the bus width of the destination address area is 8 bits.

Note 3: This may be a dummy cycle with an instruction queue buffer.

Note 4: These 2 states are added in the case of the bus width of stack address area is 8 bits.

(2) Register configuration (CPU control register)

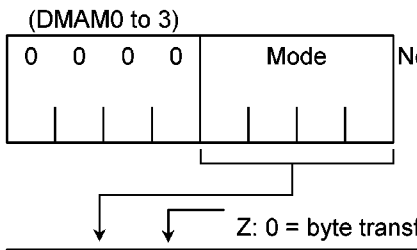

Channel 0
DMAS0
DMAD0
DMAC0
DMAM0
Transfer source address register 0
Transfer destination address register 0
Transfer counter register 0
Transfer mode register 0
(Use only lower 24 bits.)
(1 to 65536)
Channel 1
DMAS1
DMAD1
DMAC1
DMAM1
Transfer source address register 1
Transfer destination address register 1
Transfer counter register 1
Transfer mode register 1
Channel 2
DMAS2
DMAD2
DMAC2
DMAM2
Transfer source address register 2
Transfer destination address register 2
Transfer counter register 2
Transfer mode register 2
Channel 3
DMAS3
DMAD3
DMAC3
DMAM3
Transfer source address register 3
Transfer destination address register 3
Transfer counter register 3
Transfer mode register 3
8 bit
16 bit
32 bit

These Control Registers can not be set only “LDC cr, r" instruction.

Example:

```
LD          XWA, 100H
LDC         DMAS0, XWA
LD          XWA, 50H
LDC         DMAD0, XWA
LD          WA, 40H
LDC         DMAC0, WA
LD          A, 05H
LDC         DMAM0, A
```

(3) Transfer mode register details



Note: When setting values for this register, set the upper 4 bits to 0

Z: 0 = byte transfer, 1 = word transfer

execution time

0	0	0	Z	Transfer destination address INC mode (DMADn +) ← (DMASn) DMACn←DMACn -1 if DMACn = 0 then INT.	for I/O to memory	16 states (2 μs)
0	0	1	Z	Transfer destination address DEC mode (DMADn -) ← (DMASn) DMACn←DMACn -1 if DMACn = 0 then INT.	for I/O to memory	16 states (2 μs)
0	1	0	Z	Transfer source address INC mode (DMADn) ← (DMASn +) DMACn←DMACn -1 if DMACn = 0 then INT.	for memory to I/O	16 states (2 μs)
0	1	1	Z	Transfer source address DEC mode (DMADn) ← (DMASn -) DMACn←DMACn -1 if DMACn = 0 then INT.	for memory to I/O	16 states (2 μs)
1	0	0	Z	Fixed address mode (DMADn) ← (DMASn) DMACn←DMACn -1 if DMACn = 0 then INT.	I/O to I/O	16 states (2 μs)
1	0	1	1	Counter mode (DMASn) ← (DMASn + 1) DMACn←DMACn -1 if DMACn = 0 then INT.	for interrupt counter	11 states (1.4 μs)

(1 states = 125 ns at 16 MHz, High frequency mode)

Note1: n: corresponds to micro DMA channels 0 to 3.
DMADn + / DMASn +: Post-increment(Increments register value after transfer.)
DMADn - / DMASn -: Post-decrement(Decrement register value after transfer.)
Note2: Execution time: When setting source address/destination address area to 16-bit bus, 0WAIT.
Note3: Do not use the codes other than the above mantioned codes for transfer mode register.

3.4.3 Interrupt Controller

Figure 3.4.4 is a block diagram of the interrupt circuits. The left half of the diagram shows the interrupt controller; the right half includes the CPU interrupt request signal circuit and the HALT release signal circuit.

Each interrupt channel (total of 28 channels) in the interrupt controller has an interrupt request flip-flop, and interrupt priority setting register. The interrupt controller also has registers for storing the Micro DMA start vector. The interrupt request fip-flop is used to latch interrupt requests from peripheral devices. The flip-flop is cleared to 0 at reset, when the CPU reads the interrupt channel vector after the acceptance of interrupt, or when the CPU executes an instruction that clears the interrupt of that channel (writes 0 in the clear bit of the interrupt priority setting register).

For example, to clear the INT0 interrupt request, set the register after the DI instruction as follows.

```
LD   (INT0CP1) ← ---- 0 --- B
```

The status of the interrupt request flip-flop is detected by reading the clear bit. Detects whether there is an interrupt request for an interrupt channel.

The interrupt priority can be set by writing the priority in the interrupt priority setting register (eg, INT0CP1, INTCP0TG0, etc.) provided for each interrupt source. Interrupt levels to be set are from 1 to 6. Writing 0 or 7 as the interrupt priority disables the corresponding interrupt request. The priority of the non-maskable interrupt (watchdog timer, etc.) is fixed to 7. If interrupt requests with the same interrupt level are generated simultaneously, interrupts are accepted in accordance with the default priority (the smaller the vector value, the higher the priority).

The interrupt controller sends the interrupt request with the highest priority among the simultaneous interrupts and its vector address to the CPU. The CPU compares the priority value <IFF2 to 0> set in the Status Register by the interrupt request signal with the priority value sent; if the latter is higher, the interrupt is accepted. Then the CPU sets a value higher than the priority value by 1 in the CPU SR<IFF2 to 0>. Interrupt requests where the priority value equals or is higher than the set value are accepted simultaneously during the previous interrupt routine. When interrupt processing is completed (after execution of the RETI instruction) , the CPU restores the priority value saved in the stack before the interrupt was generated to the CPU SR<IFF2 to 0>.

The interrupt controller also has four registers used to store the Micro DMA start vector. These are I/O registers; unlike other Micro DMA registers (DMAS, DMAD, DMAM, and DMAC). Writing the start vector of the interrupt source for the Micro DMA processing (see Table 3.4.1), enables the corresponding interrupt to be processed by Micro DMA processing. The values must be set in the Micro DMA parameter registers (eg, DMAS and DMAD) prior to the Micro DMA processing.

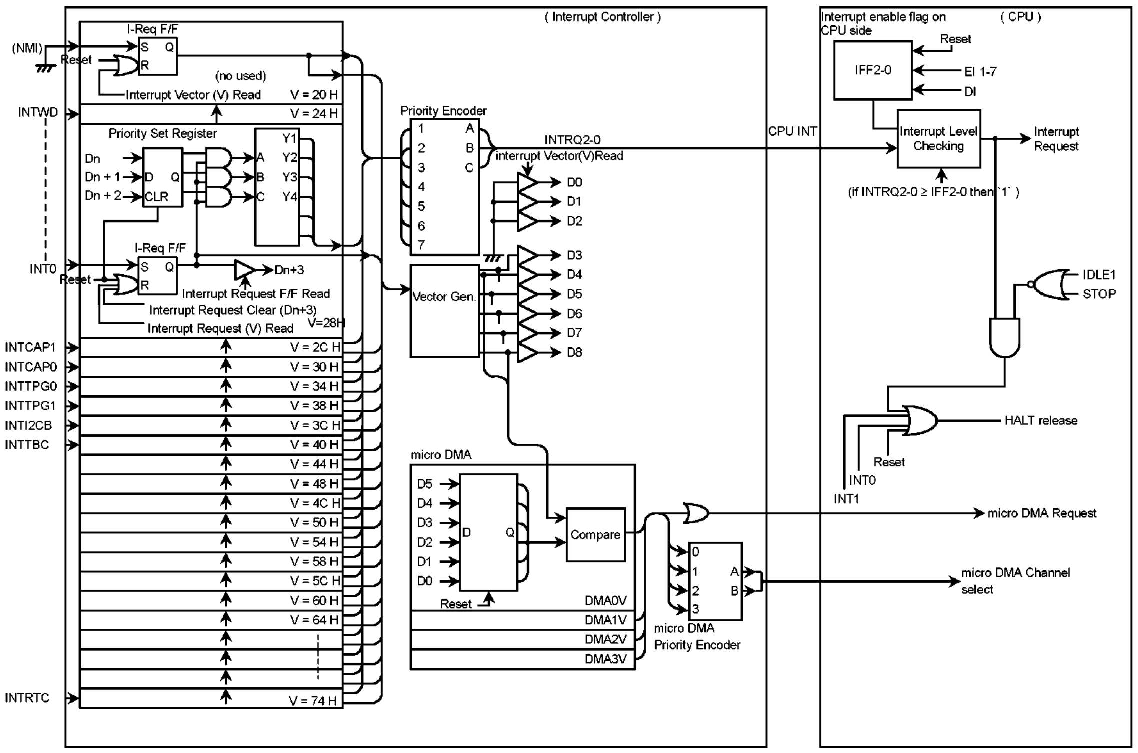


Figure 3.4.4 Block Diagram of Interrupt Controller

(1) Interrupt priority setting register

(Prohibit read-modify-write)

Symbol	Name	Address	7	6	5	4	3	2	1	0
INT0CP1	INT0 / CAP1 Interrupt Setting	0070H	INTCAP1				INT0			
			ICAP1C	ICAP1M2	ICAP1M1	ICAP1M0	I0C	I0M2	I0M1	I0M0
			R/W	W	W	W	R/W	W	W	W
			0	0	0	0	0	0	0	0
INTCP0TG0	CAP0 / TPG0 Interrupt Setting	0071H	INTTPG0				INTCAP0			
			ITPG0C	ITPG0M2	ITPG0M1	ITPG0M0	ICAP0C	ICAP0M2	ICAP0M1	ICAP0M0
			R/W	W	W	W	R/W	W	W	W
			0	0	0	0	0	0	0	0
INTTP1I2C	TPG1 / I²CBUS Interrupt Setting	0072H	INTI2CB				INTTPG1			
			II2CC	II2CM2	II2CM1	II2CM0	ITPG1C	ITPG1M2	ITPG1M1	ITPG1M0
			R/W	W	W	W	R/W	W	W	W
			0	0	0	0	0	0	0	0
INTTBCVA	TBC / VA Interrupt Setting	0073H	INTVA				INTTBC			
			IVAC	IVAM2	IVAM1	IVAM0	ITBCC	ITBCM2	ITBCM1	ITBCM0
			R/W	W	W	W	R/W	W	W	W
			0	0	0	0	0	0	0	0
INT12SIO	INT1 / INT2 / SIO Interrupt Setting	0074H	INT2 / SIO				INT1			
			I2C/ SIOC	I2M2/ SIOM2	I2M1/ SIOM1	I2M0/ SIOM0	I1C	I1M2	I1M1	I1M0
			R/W	W	W	W	R/W	W	W	W
			0	0	0	0	0	0	0	0
INT43	INT3 / INT4 Interrupt Setting	0075H	INT4				INT3			
			I4C	I4M2	I4M1	I4M0	I3C	I3M2	I3M1	I3M0
			R/W	W	W	W	R/W	W	W	W
			0	0	0	0	0	0	0	0
INTT1T0	Timer 0 / Timer1 Interrupt Setting	0076H	INTT1				INTT0			
			IT1C	IT1M2	IT1M1	IT1M0	IT0C	IT0M2	IT0M1	IT0M0
			R/W	W	W	W	R/W	W	W	W
			0	0	0	0	0	0	0	0
INTT3T2	Timer2 / Timer3 Interrupt Setting	0077H	INTT3				INTT2			
			IT3C	IT3M2	IT3M1	IT3M0	IT2C	IT2M2	IT2M1	IT2M0
			R/W	W	W	W	R/W	W	W	W
			0	0	0	0	0	0	0	0
INTT5T4	Timer4 / Timer5 Interrupt Setting	0078H	INTT5				INTT4			
			IT5C	IT5M2	IT5M1	IT5M0	IT4C	IT4M2	IT4M1	IT4M0
			R/W	W	W	W	R/W	W	W	W
			0	0	0	0	0	0	0	0
INTADRTC	AD / RTC Interrupt Setting	0079H	INTRTC				INTAD			
			IRTCC	IRTM2	IRTM1	IRTM0	IADC	IADM2	IADCM1	IADCM0
			R/W	W	W	W	R/W	W	W	W
			0	0	0	0	0	0	0	0

← Interrupt source
← bit Symbol
← Read/Write
← After reset

IxxM2	IxxM1	IxxM0	Function (Write)
0	0	0	Prohibit interrupt request.
0	0	1	Set interrupt request level to "1".
0	1	0	Set interrupt request level to "2".
0	1	1	Set interrupt request level to "3".
1	0	0	Set interrupt request level to "4".
1	0	1	Set interrupt request level to "5".
1	1	0	Set interrupt request level to "6".
1	1	1	Prohibit interrupt request.

IxxC	Function (Read)	Function (Write)
0	Indicates no interrupt request.	Clears interrupt request flag.
1	Indicates interrupt request.	don't care

Note 1: Read-modify-write is prohibited.

Note 2: Note about clearing interrupt request flag (1)
The interrupt request flag of INTCAP1,INTCAP0,INTSIO and INTAD are not cleared by writing "0" to IxxC because of they are level interrupts. They can be cleared only by resetting , reading captured data / ADREG /SCBUF or writing SCBUF.

Note 3: Note about clearing interrupt request flag (2)
When the INTTPG0 is used for a FIFO empty interrupt (a level signal), the interrupt controller also leaves a request flag (Flip/Flop) after clearing FIFO empty by setting next TPG0 data in an interrupt routin. Therefore, in this case, the INTTPG0 request flag has to be cleared before executing RETI instruction.

Figure 3.4.5 Interrupt priority setting register

(2) External interrupt control

Interrupt Input Mode Control Register

		7	6	5	4	3	2	1	0
IIMC0	bit Symbol				I4IE (INT4)	I3IE (INT3)	I2IE (INT2)	I1IE (INT1)	I0IE (INT0)
(005EH)	Read/Write				W				
	After reset				0	0	0	0	0
	Function				External interrupt input enable/disable 0: Disable 1: Enable				

Prohibit read-modify-write

Interrupt Input Mode Control Register 1

		7	6	5	4	3	2	1	0
IIMC1	bit Symbol	I4EG	I3EG	I2EG	I1EG	I0EG		INTTPG0E	INTTPG0S
(005FH)	Read/Write	W						R/W	R/W
	After reset	0	0	0	0	0		0	0
	Function	INT4 edge selection 0: Rising 1: Falling	INT3 edge selection 0: Rising 1: Falling	INT2 edge selection 0: Rising 1: Falling	INT1 edge/level selection 0: Rising edge 1: Level	INT0 edge selection 0: Rising edge 1: Falling edge		INTTPG0 interrupt TPG03 edge selection 0: Rising 1: Falling	INTTPG0 interrupt selection 0: FIFO empty interrupt 1: FIFO empty/ TPG03 interrupt

Prohibit read-modify-write

Note1: The INT0 and INT1 pins can also be used for standby release as described later. When these pins are not used for standby release, setting IIMC0<I1IE, I0IE> to "00" maintain the port function during standby mode.

Note2: When the active edge is changed by the IIMC1<I4EG, I3EG, I2EG, I1EG,I0EG>, they must be changed after disabling interrupt.

Execution example:

```
LD    (INT0CP1)    , XXXX0000B  ; Disable the INT0 interrupt, Clear the INT0 interrupt request flag.
LD    (IIMC0)      , XXXXXXX0B  ; Disable the INT0 input
LD    (IIMC1)      , XXXX1XXXB  ; Change the active edge to Level from Rising edge.
LD    (INT0CP1)    , XXXX0nnnB  ; Set interrupt level "nnn" for INT0, Clear the interrupt request flag.
```

Note3: The minimum pulse width for the active edge is 2/fc [s] (250 ns at fc = 16 MHz).

Figure 3.4.6 Interrupt Input Mode Control Register

Table 3.4.2 Setting of External Interrupt Pin Functions

Interrupt	Shared pin	Mode	Setting method
INT0	P54	Rising edge	IIMC1<I0EG> = 0,IIMC0<I0IE> = 1
		Falling edge	IIMC1<I0EG> = 1,IIMC0<I0IE> = 1
INT1	P53	Rising edge	IIMC1<I1EG> = 0,IIMC0<I1IE> = 1
		Level	IIMC1<I1EG> = 1,IIMC0<I1IE> = 1
INT2	P52	Rising edge	IIMC1<I2EG> = 0,IIMC0<I2IE> = 1
		Falling edge	IIMC1<I2EG> = 1,IIMC0<I2IE> = 1
INT3	P51	Rising edge	IIMC1<I3EG> = 0,IIMC0<I3IE> = 1
		Falling edge	IIMC1<I3EG> = 1,IIMC0<I3IE> = 1
INT4	P50	Rising edge	IIMC1<I4EG> = 0,IIMC0<I4IE> = 1
		Falling edge	IIMC1<I4EG> = 1,IIMC0<I4IE> = 1

(3) Micro DMA start vector

When the CPU reads the interrupt vector after accepting an interrupt, it simultaneously compares the interrupt vector with each channel's Micro DMA start vector (bits 3 to 8 of the interrupt vector). When the two match, the interrupt from the channel whose value matched is processed in Micro DMA mode.

If the interrupt vector matches more than two channels, the channel with the lower channel number has a higher priority.

Micro DMA0 Start Vector

		7	6	5	4	3	2	1	0
DMA0V	bit Symbol			DMA0V5	DMA0V4	DMA0V3	DMA0V2	DMA0V1	DMA0V0
(007CH)	Read/Write			W					
Prohibit read-modify-	After reset			0	0	0	0	0	0
write	Function	Micro DMA channel 0 processed by matching bits 3 to 8 of the interrupt vector.							

Micro DMA1 Start Vector

		7	6	5	4	3	2	1	0
DMA1V	bit Symbol			DMA1V5	DMA1V4	DMA1V3	DMA1V2	DMA1V1	DMA1V0
(007DH)	Read/Write			W					
Prohibit read-modify-	After reset			0	0	0	0	0	0
write	Function	Micro DMA channel 1 processed by matching bits 3 to 8 of the interrupt vector.							

Micro DMA2 Start Vector

		7	6	5	4	3	2	1	0
DMA2V	bit Symbol			DMA2V5	DMA2V4	DMA2V3	DMA2V2	DMA2V1	DMA2V0
(007EH)	Read/Write			W					
Prohibit read-modify-	After reset			0	0	0	0	0	0
write	Function	Micro DMA channel 2 processed by matching bits 3 to 8 of the interrupt vector.							

Micro DMA3 Start Vector

		7	6	5	4	3	2	1	0
DMA3V	bit Symbol			DMA3V5	DMA3V4	DMA3V3	DMA3V2	DMA3V1	DMA3V0
(007FH)	Read/Write			W					
Prohibit read-modify-	After reset			0	0	0	0	0	0
write	Function	Micro DMA channel 3 processed by matching bits 3 to 8 of the interrupt vector.							

Figure 3.4.7 Micro DMA Start Vector

(4) Notes

The instruction execution unit and the bus interface unit of this CPU operate independently of each other. Therefore, if the instruction used to clear an interrupt request flag of an interrupt is fetched before the interrupt is generated, it is possible that the CPU might execute the fetched instruction to clear the interrupt request flag while reading the interrupt vector after accepting the interrupt. If so, the CPU would read the default vector “FFFF28H" and start the interrupt processing from the address “FFFF28H".

To avoid this, make sure that the instruction used to clear the interrupt request flag comes after the DI instruction.