Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	8051
Core Size	8-Bit
Speed	12MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	30
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.5V
Data Converters	A/D 15x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	32-VFQFN Exposed Pad
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z51f0811qux

Table of Contents

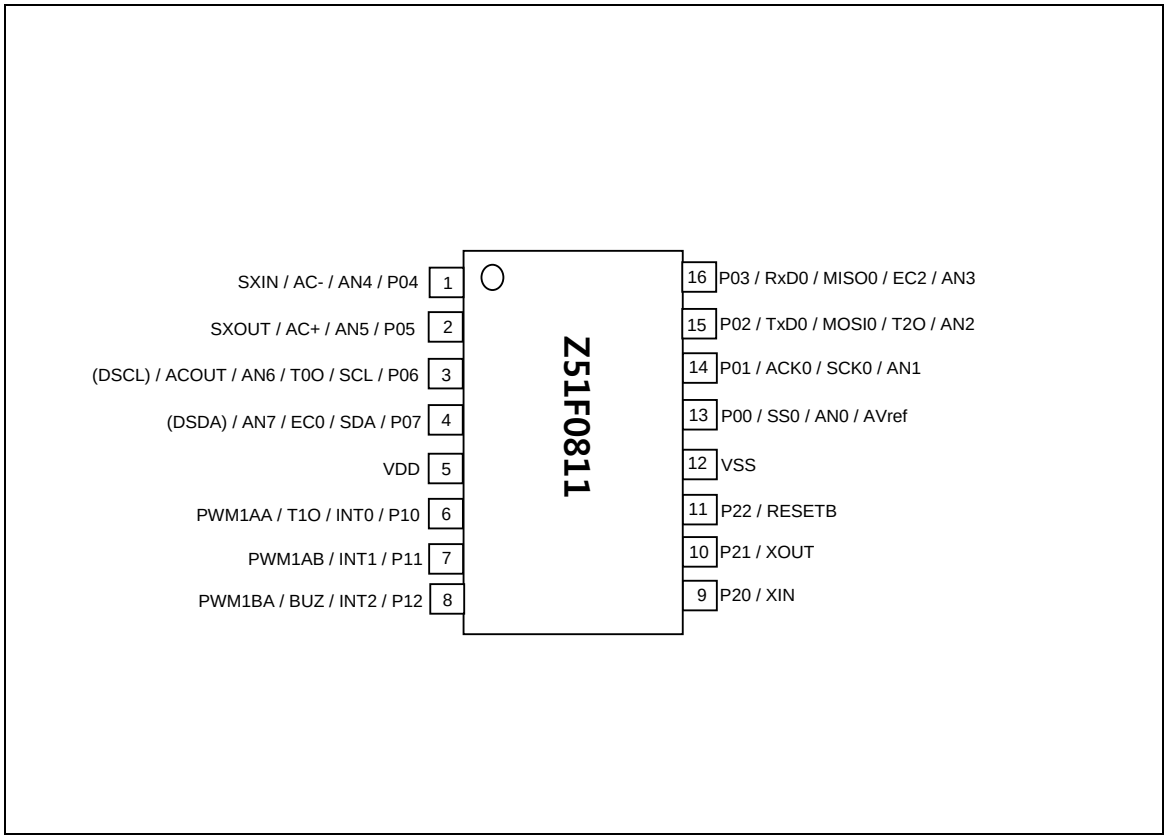

Figure 3-4 Z51F0811 16 TSSOP Pin assignment

6.2 External Interrupt I/O Port

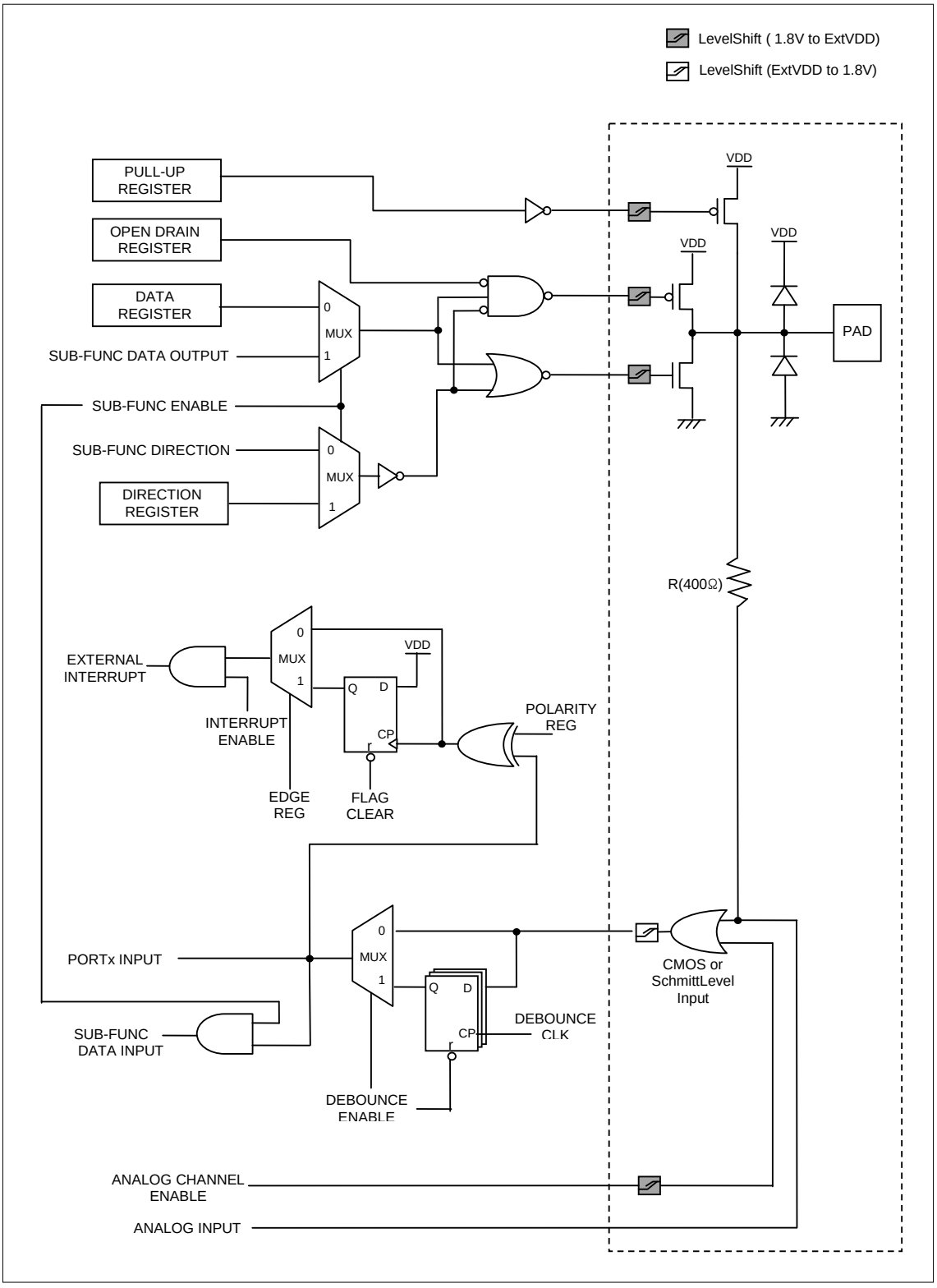

Figure 6-2 External Interrupt I/O Port

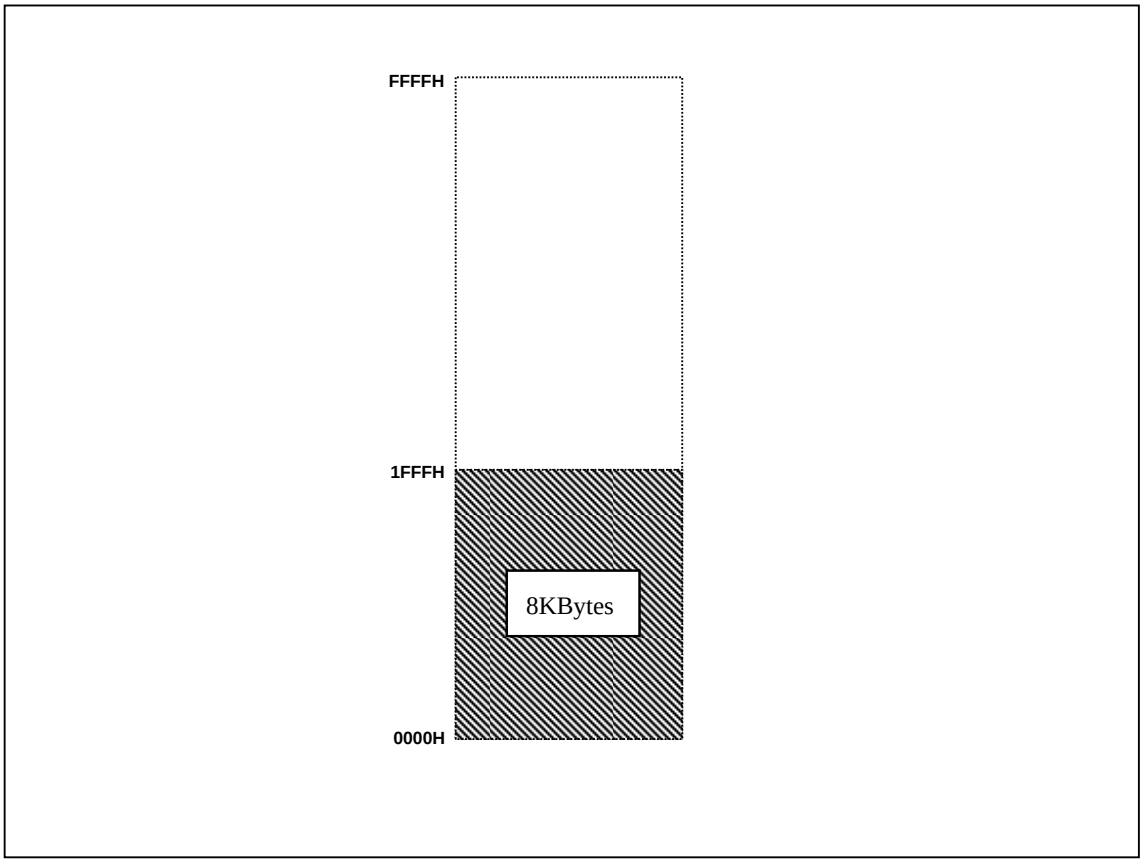

Figure 8-1 Program memory

- User Function Mode: 8KBytes Included Interrupt Vector Region
- Non-volatile and reprogramming memory: Flash memory based on EEPROM cell

1 Enable

INT2E Enable or disable External Interrupt 2

0 Disable

1 Enable

INT1E Enable or disable External Interrupt 1

0 Disable

1 Enable

INT0E Enable or disable External Interrupt 0

0 Disable

1 Enable

IE1 (Interrupt Enable Register 1) : A9H

7	6	5	4	3	2	1	0
-	-	INT11E	INT10E	INT9E	INT8E	INT7E	INT6E
-	-	R/W	R/W	R/W	R/W	R/W	R/W

Initial value : 00H

INT11E Enable or disable USART1 Tx Interrupt

0 Disable

1 Enable

INT10E Enable or disable USART1 Rx Interrupt

0 Disable

1 Enable

INT9E Enable or disable I^2C Interrupt

0 Disable

1 Enable

INT8E Enable or disable SPI0 Interrupt

0 Disable

1 Enable

INT7E Enable or disable USART0 Tx Interrupt

0 Disable

1 Enable

INT6E Enable or disable USART0 Rx Interrupt

0 Disable

1 Enable

IE2 (Interrupt Enable Register 2) : AAH

7	6	5	4	3	2	1	0
-	-	INT17E	INT16E	INT15E	INT14E	INT13E	INT12E
-	-	R/W	R/W	R/W	R/W	R/W	R/W

Initial value : 00H

INT17E Enable or disable EEPROM Interrupt

0 Disable

1 Enable

INT16E Enable or disable Timer 4 Interrupt

0 Disable

1 Enable

INT15E Enable or disable Timer 3 Interrupt

11.5.1.2 8 Bit Timer/Counter Mode

The 8-bit Timer/Counter Mode is selected by control registers as shown in Figure 11-6.

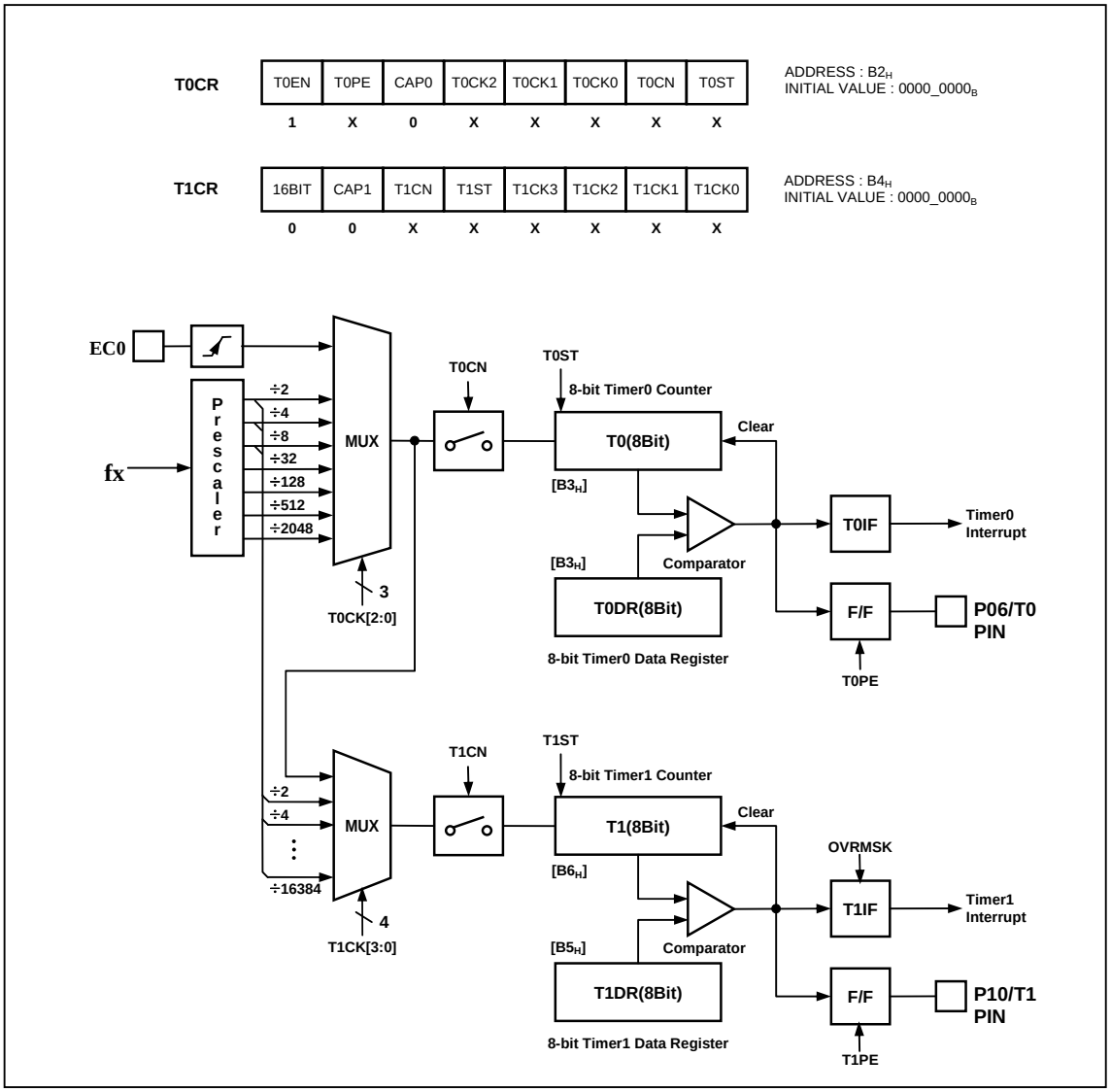

Figure 11-6 8 Bit Timer/Event Counter 0, 1 Block Diagram

The two 8-bit timers have each counter and data register. The counter register is increased by internal or external clock input. The timer 0 can use the input clock with one of 2, 4, 8, 32, 128, 512, 2048 prescaler division rates (T0CK[2:0]). The timer 1 can use the input clock with one of 1, 2, 8 ~ 16384 and timer 0 overflow clock (T1CK[3:0]). When the value of T0,1 value and the value of T0DR, T1DR are respectively identical in Timer 0, 1, the interrupt of TimerP 0, 1 occurs. The external clock (EC0) counts up the timer at the rising edge. If EC0 is selected from T0CK[2:0], EC0 port becomes input port. The timer 1 can't use the external EC0 clock.

11.5.1.5 16 Bit Capture Mode

The 16-bit capture mode is the same operation as 8-bit capture mode, except that the timer register uses 16 bits.

The clock source is selected from T0CK[2:0] and T1CK[3:0] must set 1111b and 16BIT bit must set to '1'. The 16-bit mode setting is shown as Figure 11-13.

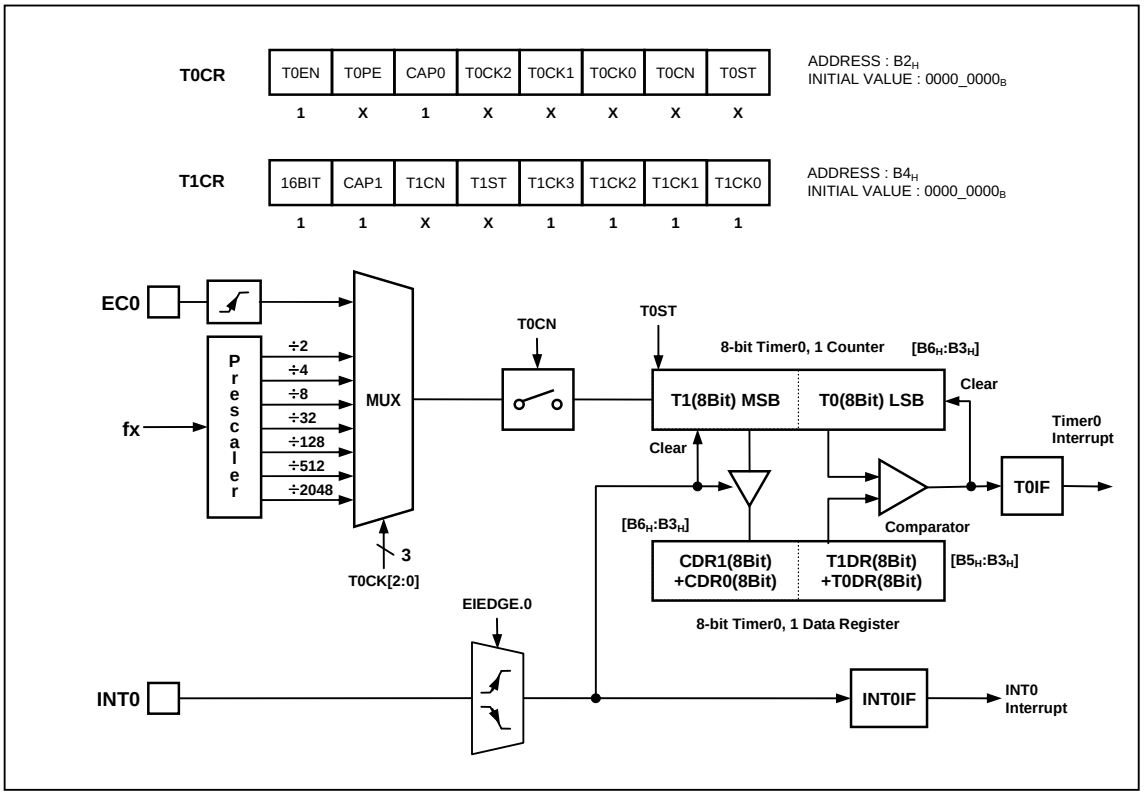

Figure 11-13 16-bit Capture Mode of Timer 0, 1

On operating PWM, it is possible that it is changed the phase and the frequency by using BMOD bit (back-to-back mode) in T1PCR register. (Figure 11-18, Figure 11-19, Figure 11-20 referred)

In the back-to-back mode, the counter of PWM repeats up/down count. In fact, the effective duty and period becomes twofold of the register set values. (Figure 11-18, Figure 11-19 referred)

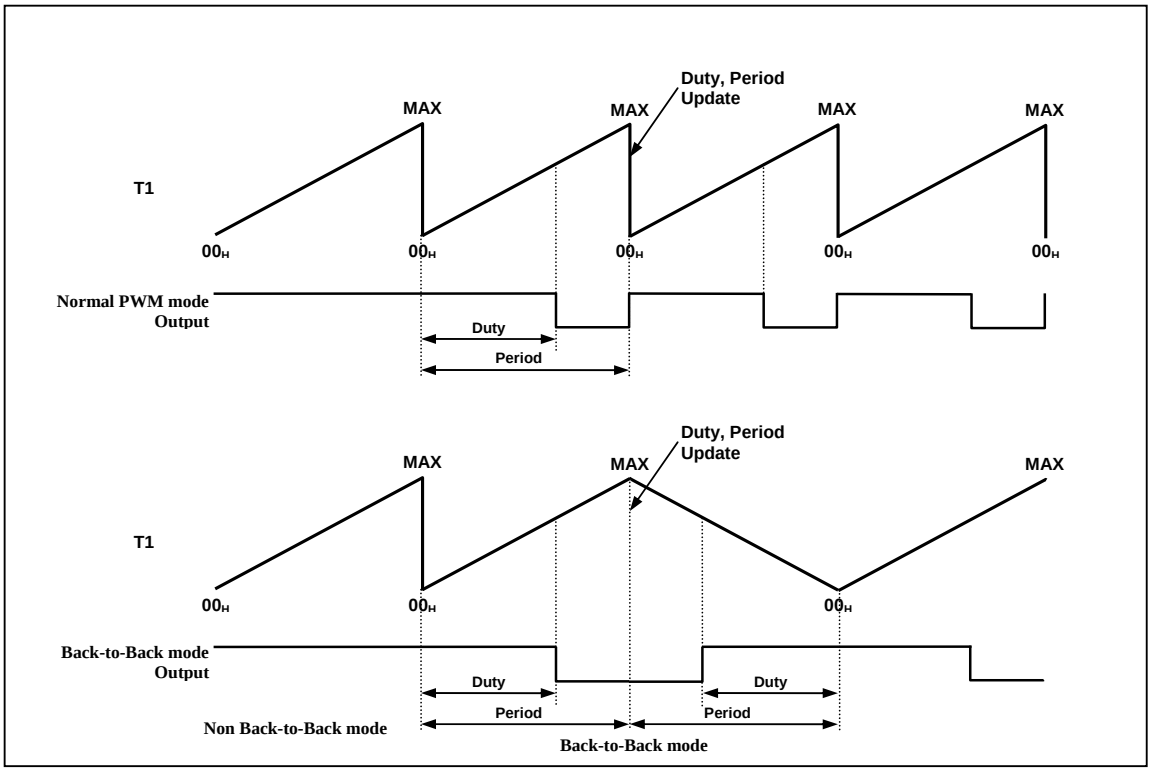

Figure 11-18 Example of PWM Output Waveform

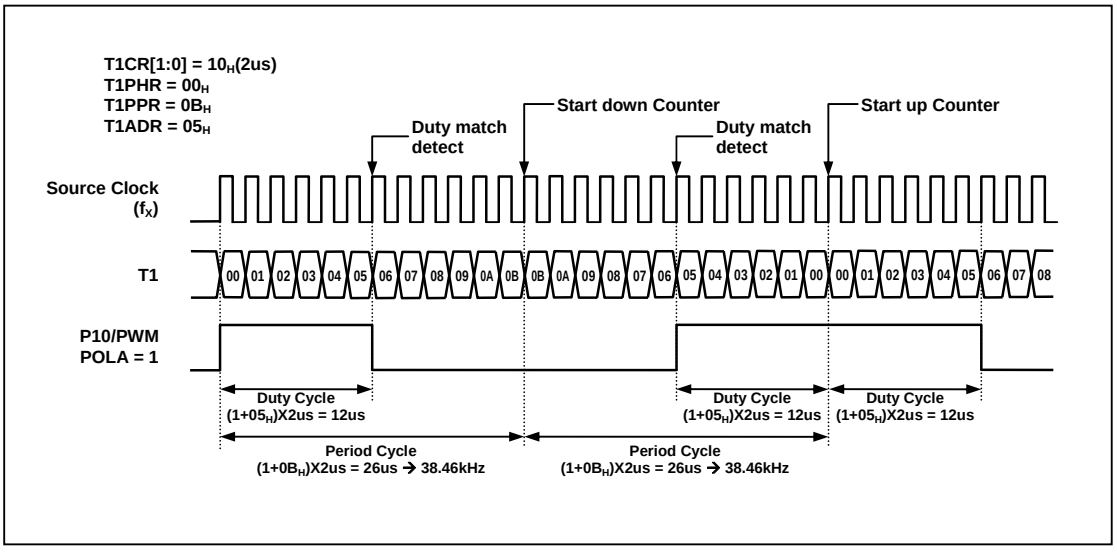

Figure 11-19 Example of PWM waveform in Back-to-Back mode at 4Mhz

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Initial value : 00H

PWM1E Control PWM
- 0 PWM disable
- 1 PWM enable

ESYNC Select the operation of External Sync Mode
- 0 External Sync Mode disable
- 1 External Sync Mode enable (using with BLNKB(P16))

BMOD Control Back-To-Back Mode operation
- 0 BtB mode disable (only up count)
- 1 BtB mode enable (Up/Down count)

PHLT Control PWM
- 0 PWM running
- 1 PWM stop

UPDT Determine the update time of PPR, PDR
- 0 Update at period match
- 1 Update at any time (after 3 timer clock, update)

UALL Control update all duty register
- 0 Write duty register separately
- 1 Write all duty registers (via A duty)

NOPS1[1:0] Select on-Overlap prescaler
Note) fpwm: PWM operation clock frequency

NOPS1	NOPS0	description
0	0	fpwm
0	1	fpwm/2
1	0	fpwm/4
1	1	fpwm/8

PCOE/ PCBOE	Select Channel C/CB operation	
	0	P1C (or P1CB) output disable
	1	P1C (or P1CB) output enable

T1PCR3 (Timer 1 PWM Control Register 3) : BEH

7	6	5	4	3	2	1	0
T1_PE	POLA	POLB	POLC	POCON	HCKE	-	PLLPDB
R/W	R/W	R/W	R/W	R/W	R/W	-	R/W

Initial value : 00H

T1_PE	Control Timer1/PWM1 Output port	
	0	T1, PWM1 Output operation disable
	1	T1, PWM1 Output operation enable
POLA	Configure PWM A-ch polarity	
	0	Negative (Duty Match time, Clear)
	1	Positive (Duty Match time, Set)
POLB	Configure PWM B-ch polarity	
	0	Negative (Duty Match time, Clear)
	1	Positive (Duty Match time, Set)
POLC	Configure PWM C-ch polarity	
	0	Negative (Duty Match time, Clear)
	1	Positive (Duty Match time, Set)
POCON	Control PWM output operation	
	0	PWM output control disable
	1	PWM output control enable
HCKE	Select High frequency Note) fCK is system frequency, Fout is PLL output frequency	
	0	High frequency disable
	1	High frequency enable (Fout > 3* fCK)
PLLPDB	Control PLL power Down mode	
	0	PLL disable (power down mode)
	1	PLL enable (for stable, needs 1ms wait)

T1DLYA (PWM1 Non-Overlap Delay Register for channel A/AB) : BFH

7	6	5	4	3	2	1	0
DLYA3	DLYA2	DLYA1	DLYA0	DLYAB3	DLYAB2	DLYAB1	DLYAB0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Initial value : 00H

DLYA[3:0]	PWM A channel Output Delay (Rising edge only)
DLYAB[3:0]	PWM AB channel Output Delay (Rising edge only)

T1DLYB (PWM1 Non-Overlap Delay Register for channel B/BB) : C2H

7	6	5	4	3	2	1	0
DLYB3	DLYB2	DLYB1	DLYB0	DLYBB3	DLYBB2	DLYBB1	DLYBB0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Initial value : 00H

DLYB[3:0] PWM B channel Output Delay (Rising edge only)

DLYBB[3:0] PWM BB channel Output Delay (Rising edge only)

T1DLYC (PWM1 Non-Overlap Delay Register for channel C/CB) : C3H

7	6	5	4	3	2	1	0
DLYC3	DLYC2	DLYC1	DLYC0	DLYCB3	DLYCB2	DLYCB1	DLYCB0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Initial value : 00H

DLYC[3:0] PWM C channel Output Delay (Rising edge only)

DLYCB[3:0] PWM CB channel Output Delay (Rising edge only)

T1ISR (Timer 1 Interrupt Status Register) : C4H

7	6	5	4	3	2	1	0
IOVR	IBTM	ICMA	ICMB	ICMC	ICAP	-	-
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Initial value : 00H

IOVR Overflow (match with T1DR in Timer mode or T1PPR in PWM mode) interrupt status
Note) for clear, write '1' to this bit
0 Overflow no occurrence
1 Overflow occurrence

IBTM Timer Bottom (goto zero) interrupt status in PWM Back-to-Back mode
Note) for clear, write '1' to this bit
0 Timer Bottom no occurrence
1 Timer Bottom occurrence

ICMA PWM A-ch Duty Match interrupt status
Note) for clear, write '1' to this bit
0 PWM A-ch Duty Match no occurrence
1 PWM A-ch Duty Match occurrence

ICMB PWM B-ch Duty Match interrupt status
Note) for clear, write '1' to this bit
0 PWM B-ch Duty Match no occurrence
1 PWM B-ch Duty Match occurrence

ICMC PWM C-ch Duty Match interrupt status
Note) for clear, write '1' to this bit
0 PWM C-ch Duty Match no occurrence
1 PWM C-ch Duty Match occurrence

ICAP Timer Capture event interrupt status
Note) for clear, write '1' to this bit
0 Timer Capture event no occurrence
1 Timer Capture event occurrence

T1MSK (Timer 1 Interrupt Mask Register) : C5H

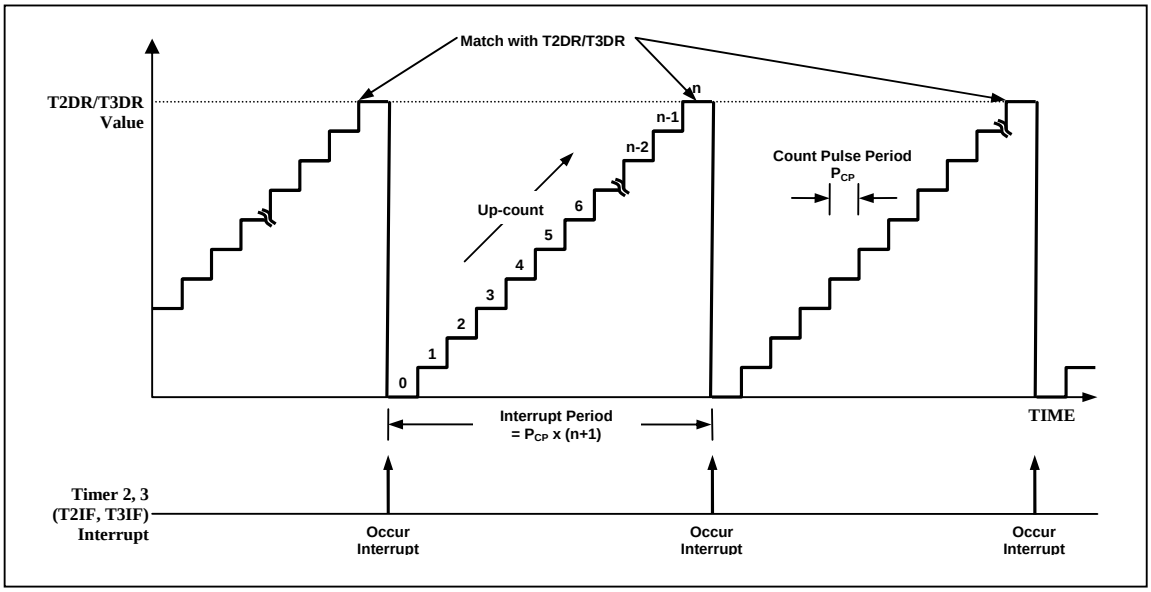

Figure 11-27 Timer/Event Counter2, 3 Example

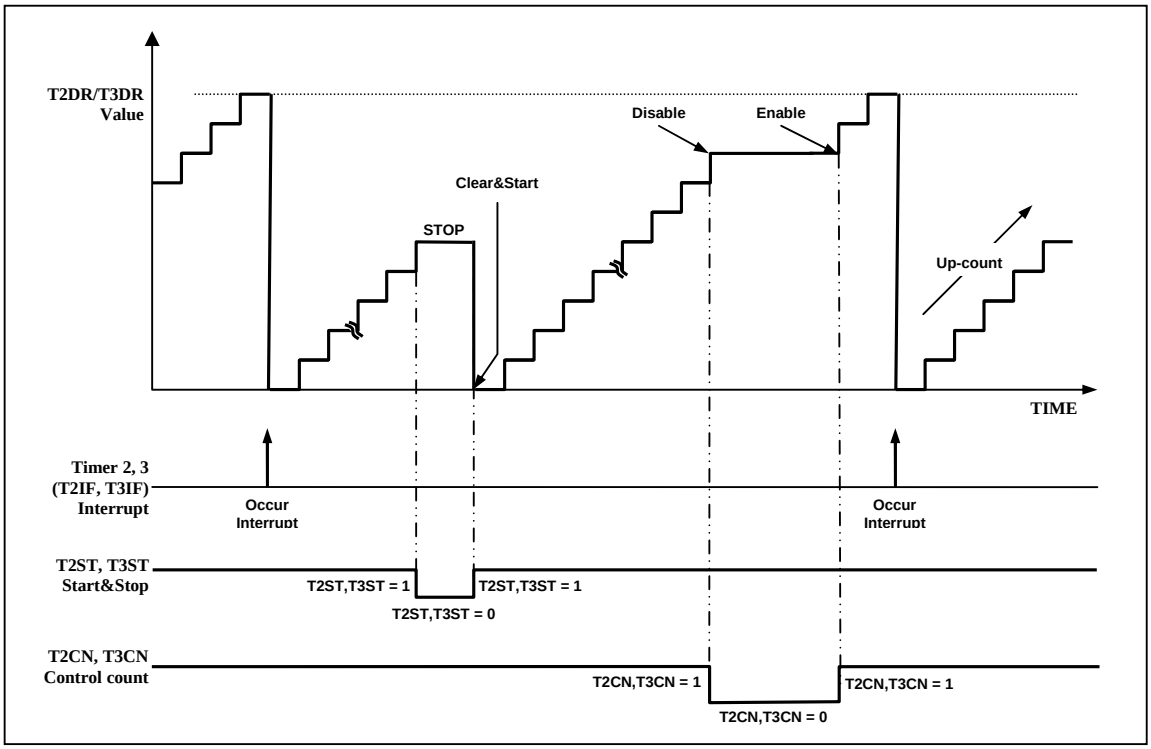

Figure 11-28 Operation Example of Timer/Event Counter2, 3

11.5.2.3 16-Bit Timer/Counter Mode

The timer register is being run with all 16bits. A 16-bit timer/counter register T2, T3 are incremented from 0003H to FFFFH until it matches T2DR, T3DR and then resets to 0000H. the match output

7 Bit	62.4KHz	31.2KHz	3.91KHz

The POL bit of T3CR register decides the polarity of duty cycle. If the duty value is set same to the period value, the PWM output is determined by the bit POL (1: High, 0: Low). And if the duty value is set to "00H", the PWM output is determined by the bit POL (1: Low, 0: High). If duty value and period value are equal, PWM output is not retain high or low but toggle.

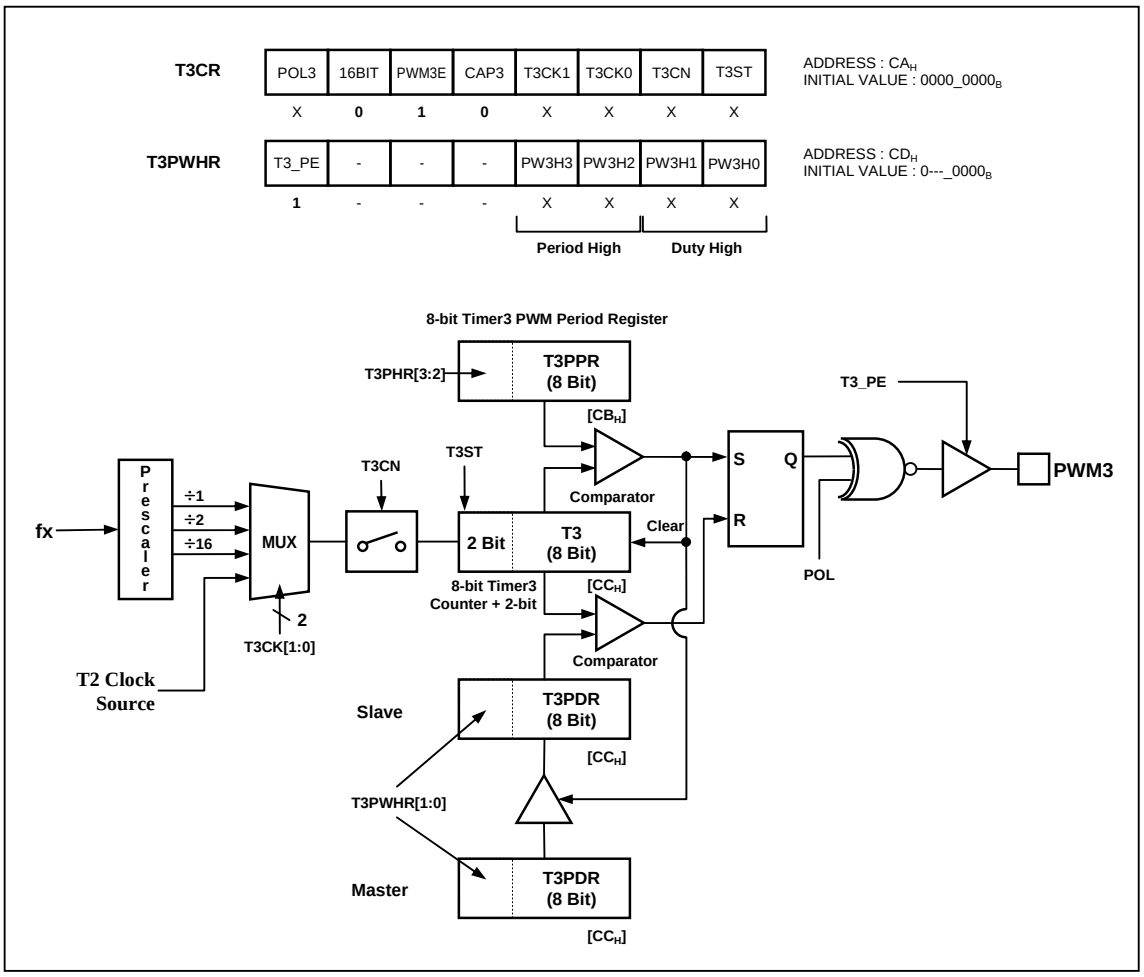

Figure 11-34 PWM Mode

11.7 USART

11.7.1 Overview

The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) is a highly flexible serial communication device. The main features are listed below.

- Full Duplex Operation (Independent Serial Receive and Transmit Registers)
- Asynchronous or Synchronous Operation
- Master or Slave Clocked Synchronous and SPI Operation
- Supports all four SPI Modes of Operation (Mode 0, 1, 2, 3)
- LSB First or MSB First Data Transfer @SPI mode
- High Resolution Baud Rate Generator
- Supports Serial Frames with 5,6,7,8, or 9 Data Bits and 1 or 2 Stop Bits
- Odd or Even Parity Generation and Parity Check Supported by Hardware
- Data OverRun Detection
- Framing Error Detection
- Digital Low Pass Filter
- Three Separate Interrupts on TX Complete, TX Data Register Empty and RX Complete
- Double Speed Asynchronous Communication Mode

USART has three main parts of Clock Generator, Transmitter and Receiver. The Clock Generation logic consists of synchronization logic for external clock input used by synchronous or SPI slave operation, and the baud rate generator for asynchronous or master (synchronous or SPI) operation. The Transmitter consists of a single write buffer, a serial shift register, parity generator and control logic for handling different serial frame formats. The write buffer allows a continuous transfer of data without any delay between frames. The receiver is the most complex part of the USART module due to its clock and data recovery units. The recovery unit is used for asynchronous data reception. In addition to the recovery unit, the Receiver includes a parity checker, a shift register, a two level receive FIFO (UDATAx) and control logic. The Receiver supports the same frame formats as the Transmitter and can detect Frame Error, Data OverRun and Parity Errors.

11.7.10 SPI Mode

The USART can be set to operate in industrial standard SPI compliant mode. The SPI mode has the following features.

- Full duplex, three-wire synchronous data transfer
- Master or Slave operation
- Supports all four SPI modes of operation (mode0, 1, 2, and 3)
- Selectable LSB first or MSB first data transfer
- Double buffered transmit and receive
- Programmable transmit bit rate

When SPI mode is enabled (UMSEL[1:0]=3), the Slave Select (SS) pin becomes active low input in slave mode operation, or can be output in master mode operation if SPISS bit is set.

Note that during SPI mode of operation, the pin RXD is renamed as MISO and TXD is renamed as MOSI for compatibility to other SPI devices.

11.7.10.1 SPI Clock formats and timing

To accommodate a wide variety of synchronous serial peripherals from different manufacturers, the USART has a clock polarity bit (UCPOL) and a clock phase control bit (UCPHA) to select one of four clock formats for data transfers. UCPOL selectively insert an inverter in series with the clock. UCPHA chooses between two different clock phase relationships between the clock and data. Note that UCPHA and UCPOL bits in UCTRLx1 register have different meanings according to the UMSEL[1:0] bits which decides the operating mode of USART.

Table below shows four combinations of UCPOL and UCPHA for SPI mode 0, 1, 2, and 3.

11.9.8.3 Slave Transmitter

To operate I^2C in slave transmitter, follow the recommended steps below.

1. If the main operating clock (SCLK) of the system is slower than that of SCL, load value 0x00 into I2CSDAHR to make SDA change within one system clock period from the falling edge of SCL. Note that the hold time of SDA is calculated by SDAH x period of SCLK where SDAH is multiple of number of SCLK coming from I2CSDAHR. When the hold time of SDA is longer than the period of SCLK, I^2C (slave) cannot transmit serial data properly.
2. Enable I^2C by setting IICEN bit and INTEN bit in I2CMR. This provides main clock to the peripheral.
3. When a START condition is detected, I^2C receives one byte of data and compares it with SLA bits in I2CSAR. If the GCALLEN bit in I2CSAR is enabled, I^2C compares the received data with value 0x00, the general call address.
4. If the received address does not equal to SLA bits in I2CSAR, I^2C enters idle state ie, waits for another START condition. Else if the address equals to SLA bits and the ACKEN bit is enabled, I^2C generates SSEL interrupt and the SCL line is held LOW. Note that even if the address equals to SLA bits, when the ACKEN bit is disabled, I^2C enters idle state. When SSEL interrupt occurs, load transmit data to I2CDR and write arbitrary value to I2CSR to release SCL line.
5. 1-Byte of data is being transmitted.
6. In this step, I^2C generates TEND interrupt and holds the SCL line LOW regardless of the reception of ACK signal from master. Slave can select one of the following cases.

   1) No ACK signal is detected and I^2C waits STOP or repeated START condition.
   2) ACK signal from master is detected. Load data to transmit into I2CDR.

   After doing one of the actions above, write arbitrary value to I2CSR to release SCL line. In case of 1) move to step 7 to terminate communication. In case of 2) move to step 5. In either case, a repeated START condition can be detected. For that case, move step 4.
7. This is the final step for slave transmitter function of I^2C, handling STOP interrupt. The STOP bit indicates that data transfer between master and slave is over. To clear I2CSR, write arbitrary value to I2CSR. After this, I^2C enters idle state.

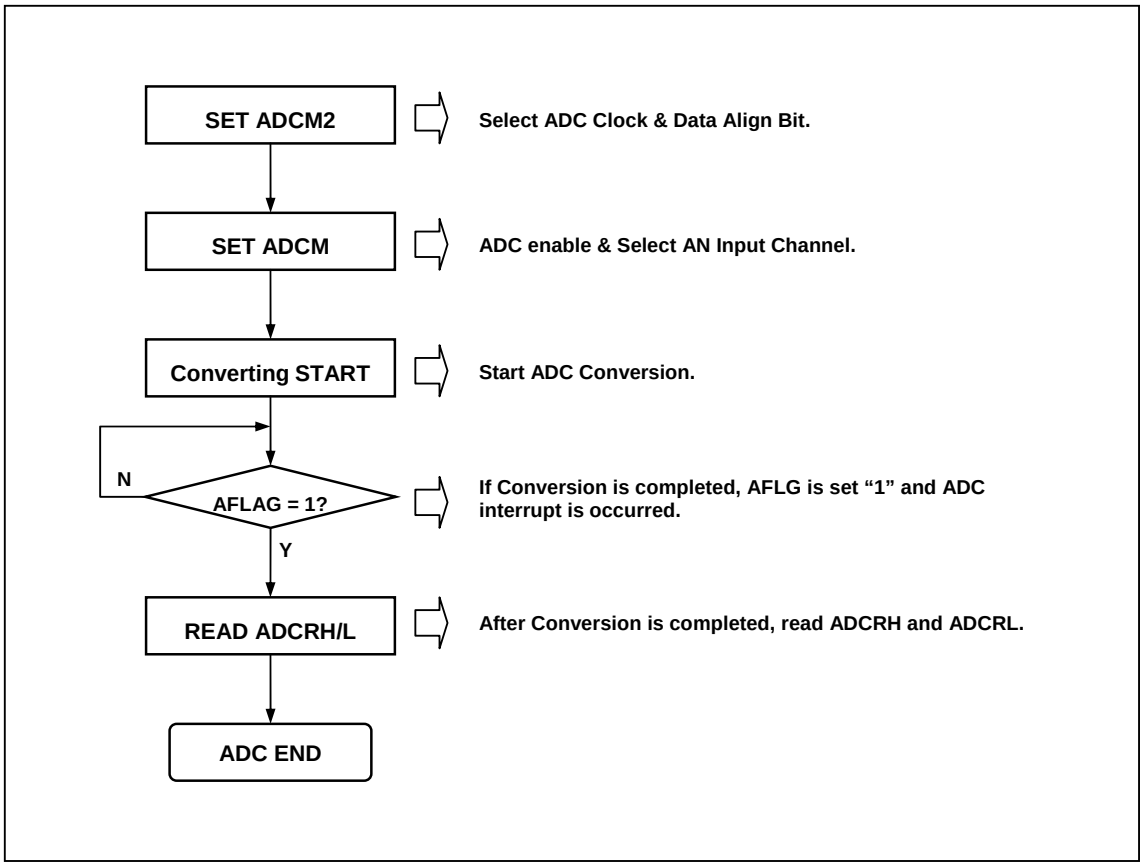

Figure 11-66 Converter Operation Flow

11.10.4 Register Map

Name	Address	Dir	Default	Description
ADCM	9AH	R/W	8FH	A/D Converter Mode Register
ADCRH	9BH	R	-	A/D Converter Result High Register
ADCRL	9CH	R	-	A/D Converter Result Low Register
ADCM2	9BH	R/W	8FH	A/D Converter Mode 2 Register

11.10.5 ADC Register description

The ADC Register consists of A/D Converter Mode Register (ADCM), A/D Converter Result High Register (ADCRH), A/D Converter Result Low Register (ADCRL), A/D Converter Mode 2 Register (ADCM2).

Note) when STBY bit is set to '1', ADCM2 can be read. If ADC enables, it is possible only to write ADCM2.When reading, ADCRH is read.

11.10.6 Register description for ADC

ADCM (A/D Converter Mode Register) : 9AH

7	6	5	4	3	2	1	0
STBY	ADST	REFSEL	AFLAG	ADSEL3	ADSEL2	ADSEL1	ADSEL0
R/W	R/W	R/W	R	R/W	R/W	R/W	R/W

Initial value : 8FH

STBY Control operation of A/D standby (power down)

0 ADC module enable

1 ADC module disable (power down)

ADST Control A/D Conversion stop/start.

0 ADC Conversion Stop

1 ADC Conversion Start

REFSEL A/D Converter reference selection

0 Internal Reference (VDD)

1 External Reference(AVREF, AN0 disable)

AFLAG A/D Converter operation state

0 During A/D Conversion

1 A/D Conversion finished

ADSEL[3:0] A/D Converter input selection

ADSEL3	ADSEL2	ADSEL1	ADSEL0	Description
0	0	0	0	Channel0(AN0)
0	0	0	1	Channel1(AN1)
0	0	1	0	Channel2(AN2)
0	0	1	1	Channel3(AN3)
0	1	0	0	Channel4(AN4)
0	1	0	1	Channel5(AN5)
0	1	1	0	Channel6(AN6)
0	1	1	1	Channel7(AN7)
1	0	0	0	Channel8(AN8)
1	0	0	1	Channel9(AN9)
1	0	1	0	Channel10(AN10)
1	0	1	1	Channel11(AN11)
1	1	0	0	Channel12(AN12)
1	1	0	1	Channel13(AN13)
1	1	1	0	Channel14(AN14)
1	1	1	1	Channel15(VDD18)

15.5 Parallel Mode

15.5.1 Overview

Parallel program mode transfers address and data by byte. 3-byte address can be entered by one from the lease significant byte of address. If only LSB is changed, only one byte can be transferred. And if the second byte is changed, the first and second byte can be transferred. Upper 4-bit of the most significant byte selects memory to be accessed. Table 15-4 shows memory type to be accessible by parallel mode. Address auto-increment is supported when read or write data without address.

The erase and program sequence of Flash and data EEPROM is identical to that of ISP mode except the entrance of parallel mode . Refer to Table 15-5 for the entrance method for parallel mode.

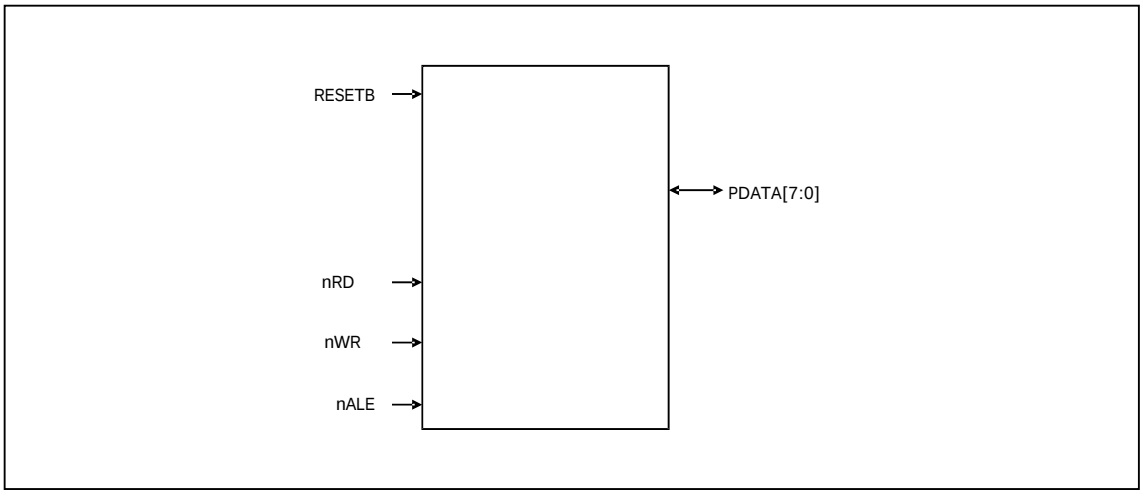

Figure 15-7 Pin diagram for parallel programming

Table 15-4 The selection of memory type by ADDRH[7:4]

ADDRH[7:4]				Memory Type
0	0	0	0	Program Memory
0	0	0	1	External Memory
0	0	1	0	SFR

- Using compare jump instructions with input port, it could cause error due to the timing conflict inside the MCU.
- Compare jump Instructions which cause potential error used with input port condition:

```
JB      bit, rel   ; jump on direct bit=1
JNB     bit, rel   ; jump on direct bit=0
JBC     bit, rel   ; jump on direct bit=1 and clear
CJNE    A, dir, rel        ; compare A, direct jne relative
DJNZ    dir, rel   ; decrement direct byte, jnz relative
```

- It is only related with Input port. Internal parameters, SFRs and output bit ports don't cause an y error by using compare jump instructions.
- If input signal is fixed, there is no error in using compare jump instructions.

- Error status example

```
while(1){
  if (P00==1){ P10=1; }
  else { P10=0; }
  P11^=1;
}
```

```
zzz:   JNB     080.0, xxx  ; it possible to be error
       SETB    088.0
       SJMP    yyy
xxx:   CLR     088.0
yyy:   MOV     C,088.1
       CPL     C
       MOV     088.1,C
       SJMP    zzz
```

```
unsigned char ret_bit_err(void)
{
  return !P00;
}
```

```
       MOV     R7, #000
       JB      080.0, xxx  ; it possible to be error
       MOV     R7, #001
xxx:   RET
```

- Preventative measures (2 cases)
  - Do not use input bit port for bit operation but for byte operation. Using byte operation instead of bit operation will not cause any error in using compare jump instructions for input port.

```
while(1){
 if ((P0&0x01)==0x01){ P10=1; }
  else { P10=0; }
   P11^=1;
}
```

```
zzz:   MOV     A, 080      ; read as byte
       JNB     0E0.0, xxx  ; compare
       SETB    088.0
       SJMP    yyy
xxx:   CLR     088.0
yyy:   MOV     C,088.1
       CPL     C
       MOV     088.1,C
       SJMP    zzz
```