



# CMS79FT61x user manual

**Enhanced 8-bit CMOS microcontroller with flash memory**

**Rev. 1.2.0**

Please be reminded about following CMS's policies on intellectual property

\* Cmsemicon Limited (denoted as 'our company' for later use) has already applied for relative patents and entitled legal rights. Any patents related to CMS's MCU or other products is not authorized to use. Any individual, organization or company which infringes our company's intellectual property rights will be forbidden and stopped by our company through any legal actions, and our company will claim the lost and required for compensation of any damage to the company.

\* The name of Cmsemicon Limited and logo are both trademarks of our company.

\* Our company preserve the rights to further elaborate on the improvements about products' function, reliability and design in this manual. However, our company is not responsible for any usage about this manual. The applications and their purposes in this manual are just for clarification, our company does not guarantee that these applications are feasible without further improvements and changes, and our company does not recommend any usage of the products in areas where people's safety is endangered during accident. Our company's products are not authorized to be used for life-saving or life support devices and systems.

## Index

<b>1. Product Description .....</b>	<b>7</b>
1.1 Features.....	7
1.2 System Structure Diagram .....	9
1.3 Pin Allocation.....	10
1.3.1 CMS79FT611 Pin Out Diagram.....	10
1.3.2 CMS79FT613 Pin Out Diagram .....	10
1.3.3 CMS79FT616 Pin Out Diagram .....	10
1.4 System Configuration Register.....	12
1.5 Online Serial Programming .....	13
<b>2. Central Processing Unit (CPU) .....</b>	<b>14</b>
2.1 Memory .....	14
2.1.1 Program Memory .....	14
2.1.1.1 Reset Vector (0000H) .....	14
2.1.1.2 Interrupt Vector .....	15
2.1.1.3 Look-up Table .....	16
2.1.1.4 Jump Table .....	18
2.1.2 Data Memory.....	19
2.2 Addressing Mode .....	24
2.2.1 Direct Addressing.....	24
2.2.2 Immediate Addressing.....	24
2.2.3 Indirect Addressing.....	24
2.3 Stack .....	25
2.4 Accumulator (ACC) .....	26
2.4.1 General .....	26
2.4.2 ACC Applications .....	26
2.5 Program Status Register (STATUS) .....	27
2.6 Pre-scaler (OPTION_REG) .....	29
2.7 Program Counter (PC) .....	31
2.8 Watchdog Timer (WDT).....	32
2.8.1 WDT Period.....	32
2.8.2 Watchdog Timer Control Register WDTCON .....	32
<b>3. System Clock .....</b>	<b>33</b>
3.1 General .....	33
3.2 System Oscillator .....	34
3.3 Reset Time.....	34
3.4 Oscillator Control Register .....	34
3.5 Clock Block Diagram.....	35
<b>4. Reset.....</b>	<b>36</b>
4.1 Power on Reset.....	36
4.2 Power off Reset.....	37
4.2.1 Power off Reset General.....	37
4.2.2 Improvements for Power off Reset.....	38
4.3 Watchdog Reset.....	38
<b>5. Sleep Mode.....</b>	<b>39</b>
5.1 Enter Sleep Mode.....	39
5.2 Awaken from Sleep Mode .....	39

5.3	Interrupt Awakening.....	40
5.4	Sleep Mode Application.....	40
5.5	Sleep Mode Awaken Time.....	40
<b>6.</b>	<b>I/O port.....</b>	<b>41</b>
6.1	I/O Port Structure .....	42
6.2	PORTA .....	45
6.2.1	PORTA Data and Direction Control.....	45
6.2.2	PORTA Pull-up Resistance .....	46
6.2.3	PORTA Pull-down Resistor .....	46
6.2.4	PORTA Analog Control Selection.....	47
6.2.5	PortA Level Change Interrupt.....	48
6.2.6	PORTA Drive Current Control .....	49
6.3	PORTB.....	50
6.3.1	PORTB Data and Direction .....	50
6.3.2	PORTB Pull-up Resistance.....	51
6.3.3	PORTB Pull-down Resistor.....	51
6.3.4	PORTB Analog Selection Control.....	52
6.3.5	PORTB Level Change Interrupt .....	53
6.3.6	PORTB Drive Current Control.....	54
6.4	PORTC.....	55
6.4.1	PORTC Data and Direction.....	55
6.4.2	PORTC Pull-up Resistance.....	56
6.4.3	PORTC Pull-down Resistor.....	56
6.4.4	PORTC Analog Control Selection .....	57
6.5	I/O Usage.....	58
6.5.1	Write I/O Port .....	58
6.5.2	Read I/O Port.....	58
6.6	Precautions for I/O Port Usage.....	59
<b>7.</b>	<b>Interrupt.....</b>	<b>60</b>
7.1	Interrupt General .....	60
7.2	Interrupt Control Register.....	61
7.2.1	Interrupt Control Register.....	61
7.2.2	Peripherals Interrupt Enable Register .....	62
7.2.3	Peripherals Interrupt Request Register.....	64
7.3	Protection Methods for Interrupt.....	66
7.4	Interrupt Priority and Multi-interrupt Nesting.....	66
<b>8.</b>	<b>TIMER0.....</b>	<b>67</b>
8.1	TIMER0 General .....	67
8.2	Working Principle for TIMER0 .....	68
8.2.1	8-bit Timer Mode .....	68
8.2.2	8-bit Counter Mode .....	68
8.2.3	Software Programmable Pre-scaler.....	68
8.2.4	Switch Between TIMER0 and WDT Module Pre-scaler .....	69
8.2.5	TIMER0 Interrupt.....	69
8.3	TIMER0 Related Register .....	70
<b>9.</b>	<b>TIMER1.....</b>	<b>71</b>
9.1	TIMER1 Overview .....	71
9.2	Operating Principle for TIMER1.....	71

9.3	Clock Source Selection .....	72
9.3.1	Internal Clock Source.....	72
9.3.2	External Clock Source.....	72
9.4	TIMER1 Pre-scaler.....	73
9.5	TIMER1 Operating Principle Under Asynchronous Counter Mode.....	73
9.5.1	Read and Write Operations to TIMER1 in Asynchronous Counter Mode.....	73
9.6	TIMER1 Gate Control.....	74
9.7	TIMER1 Interrupt.....	74
9.8	TIMER1 Working Principle During Sleep.....	74
9.9	TIMER1 Control Register .....	75
<b>10.</b>	<b>TIMER2.....</b>	<b>76</b>
10.1	TIMER2 General .....	76
10.2	Operating Principle of TIMER2.....	77
10.3	TIMER2 Related Register .....	78
<b>11.</b>	<b>Analog to Digital Conversion (ADC) .....</b>	<b>79</b>
11.1	ADC General.....	79
11.2	ADC Configuration .....	80
11.2.1	Port Configuration .....	80
11.2.2	Channel Selection.....	80
11.2.3	ADC Internal Reference Voltage .....	80
11.2.4	ADC Reference Voltage.....	80
11.2.5	Converter Clock .....	81
11.2.6	ADC Interrupt.....	81
11.2.7	Output Formatting .....	81
11.3	ADC Working Principle.....	82
11.3.1	Start Conversion .....	82
11.3.2	Complete Conversion.....	82
11.3.3	Stop Conversion.....	82
11.3.4	Working Principle of ADC in Sleep Mode.....	82
11.3.5	A/D Conversion Procedure .....	83
11.4	ADC Related Register.....	84
<b>12.</b>	<b>Touch Button.....</b>	<b>87</b>
12.1	Touch Button Module Overview.....	87
12.2	Precautions of Using Touch Button Module.....	87
<b>13.</b>	<b>PWM Module .....</b>	<b>88</b>
13.1	Pin Configuration.....	88
13.2	Description of The Relevant Registers.....	88
13.3	PWM Register Write Sequence.....	93
13.4	PWM Period.....	93
13.5	PWM Duty Cycle .....	94
13.6	Change in System Clock Frequency .....	94
13.7	Programmable Dead-time Delay Mode .....	94
13.8	PWM Settings.....	95
13.9	PWM Cycle Interrupt .....	95
<b>14.</b>	<b>Universal Synchronous/asynchronous Transceiver (USART).....</b>	<b>96</b>
14.1	USART Asynchronous Mode.....	98
14.1.1	USART Asynchronous Generator .....	98
14.1.1.1	Enable Transmit .....	98

14.1.1.2	Transmit Data.....	99
14.1.1.3	Transmit Interrupt .....	99
14.1.1.4	TSR Status .....	99
14.1.1.5	Transmit 9-bit Character.....	99
14.1.1.6	Configure Asynchronous Transmit.....	100
14.1.2	USART Asynchronous Receiver .....	101
14.1.2.1	Enable Receiver .....	101
14.1.2.2	Receive Data.....	101
14.1.2.3	Receive Interrupt .....	102
14.1.2.4	Receive Frame Error .....	102
14.1.2.5	Receive Overflow Error.....	102
14.1.2.6	Receive 9-bit Character.....	102
14.1.2.7	Asynchronous Receive Configuration.....	103
14.2	Clock Precision for Asynchronous Operations .....	104
14.3	USART Related Register.....	104
14.4	USART Baud Rate Generator (BRG) .....	106
14.5	USART Synchronous Mode .....	107
14.5.1	Synchronous Master Control Mode.....	107
14.5.1.1	Master Control Clock .....	107
14.5.1.2	Clock Polarity.....	107
14.5.1.3	Synchronous Master Control Transmit .....	108
14.5.1.4	Synchronous Master Control Transmit Configuration .....	109
14.5.1.5	Synchronous Master Control Receive .....	110
14.5.1.6	Slave Clock.....	110
14.5.1.7	Receive Overflow Error.....	110
14.5.1.8	Receive 9-bit Character.....	110
14.5.1.9	Synchronous Master Control Receive Configuration .....	111
14.5.2	Synchronous Slave Mode .....	112
14.5.2.1	USART Synchronous Slave Transmit.....	112
14.5.2.2	Synchronous Slave Transmit Configuration.....	112
14.5.2.3	USART Synchronous Slave Receive.....	112
14.5.2.4	Synchronous Slave Receive Configuration.....	113
<b>15.</b>	<b>Program EEPROM and Program Memory Control .....</b>	<b>114</b>
15.1	General .....	114
15.2	Related Register.....	115
15.2.1	EEADR and EEADRH Register.....	115
15.2.2	EECON1 and EECON2 Register .....	115
15.3	Read Program EEPROM .....	118
15.4	Write Program EEPROM.....	119
15.5	Read Program Memory.....	121
15.6	Write Program Memory .....	121
15.7	Precautions on Program EEPROM .....	122
15.7.1	Programming Time for Program EEPROM .....	122
15.7.2	Number of Times for Programming EEPROM.....	122
15.7.3	Write Verification .....	122
15.7.4	Protection to Avoid Writing Wrongly .....	123
<b>16.</b>	<b>Electrical Parameter .....</b>	<b>124</b>
16.1	Limit Parameter.....	124
16.2	DC Feature.....	125
16.3	ADC Electrical Characteristics.....	126

---

16.4	ADC Internal LDO Reference Voltage Characteristics .....	126
16.5	Power-on Reset Characteristics.....	126
16.6	AC Electrical Characteristics .....	127
<b>17.</b>	<b>Instructions .....</b>	<b>128</b>
17.1	Instructions Table .....	128
17.2	Instructions Illustration.....	130
<b>18.</b>	<b>Packaging.....</b>	<b>146</b>
18.1	SOP8.....	146
18.2	SOP16.....	147
18.3	SOP20.....	148
<b>19.</b>	<b>Version Revision.....</b>	<b>149</b>

# 1. Product Description

## 1.1 Features

- ◆ Memory
  - Flash: 2Kx16
  - Universal RAM: 256x8
- ◆ 8 level stack buffer
- ◆ Clean instructions (68 instructions)
- ◆ Look-up table
- ◆ Built-in WDT timer
- ◆ Built-in low voltage detection circuit
- ◆ Interrupt source
  - 3 timer interrupt
  - Interrupt for change in electrical level PortA/ PortB port
  - Other peripherals interrupt
- ◆ Timer
  - 8-bit timer: TIMER0, TIMER2
  - 16-bit timer: TIMER1
- ◆ Built-in 128-byte EEPROM
  - Can re-write/erase up to 100000 times
- ◆ Built-in touch button detection mod
  - No need for external touch capacitor
  - Max support 8 touch channels
- ◆ Working voltage: 2.6V~5.5V@16MHz  
2.0V~5.5V@8MHz
- ◆ Working temperature: -40°C~85°C
- ◆ Internal high precision RC osc: design frequency of 8MHz/16MHz
- ◆ Instructions period (single instruction or double instructions)
- ◆ IO characteristics
  - PORTA/PORTB support various source current selections:  
4mA/8mA/12mA/16mA/20mA/24mA/28mA
  - PORTA/PORTB support various sink current selections:  
60mA/120mA
  - PORTA/PORTB support voltage change wakeup
  - All IO built-in pull-up/pull-down resistors
- ◆ 10-bit PWM module with Complementary output
  - Support 2 channel Complementary outputs + 1 channel independent period output or 4 channel common period output + 1 channel independent period output
- ◆ Built-in 1 USART communication module
  - Support Synchronized master/slave mode and asynchronized full duplex mode
  - Can configure RA3/RA4 or RC0/RC1
- ◆ High precision 12 bit ADC
  - Built-in high precision 1.2V reference voltage  
 $\pm 1.5\% @ V_{DD} = 2.5V \sim 5.5V \quad T_A = 25^\circ C$   
 $\pm 2\% @ V_{DD} = 2.5V \sim 5.5V \quad T_A = -40^\circ C \sim 85^\circ C$
  - Selectable internal reference source:  
2V/2.4V

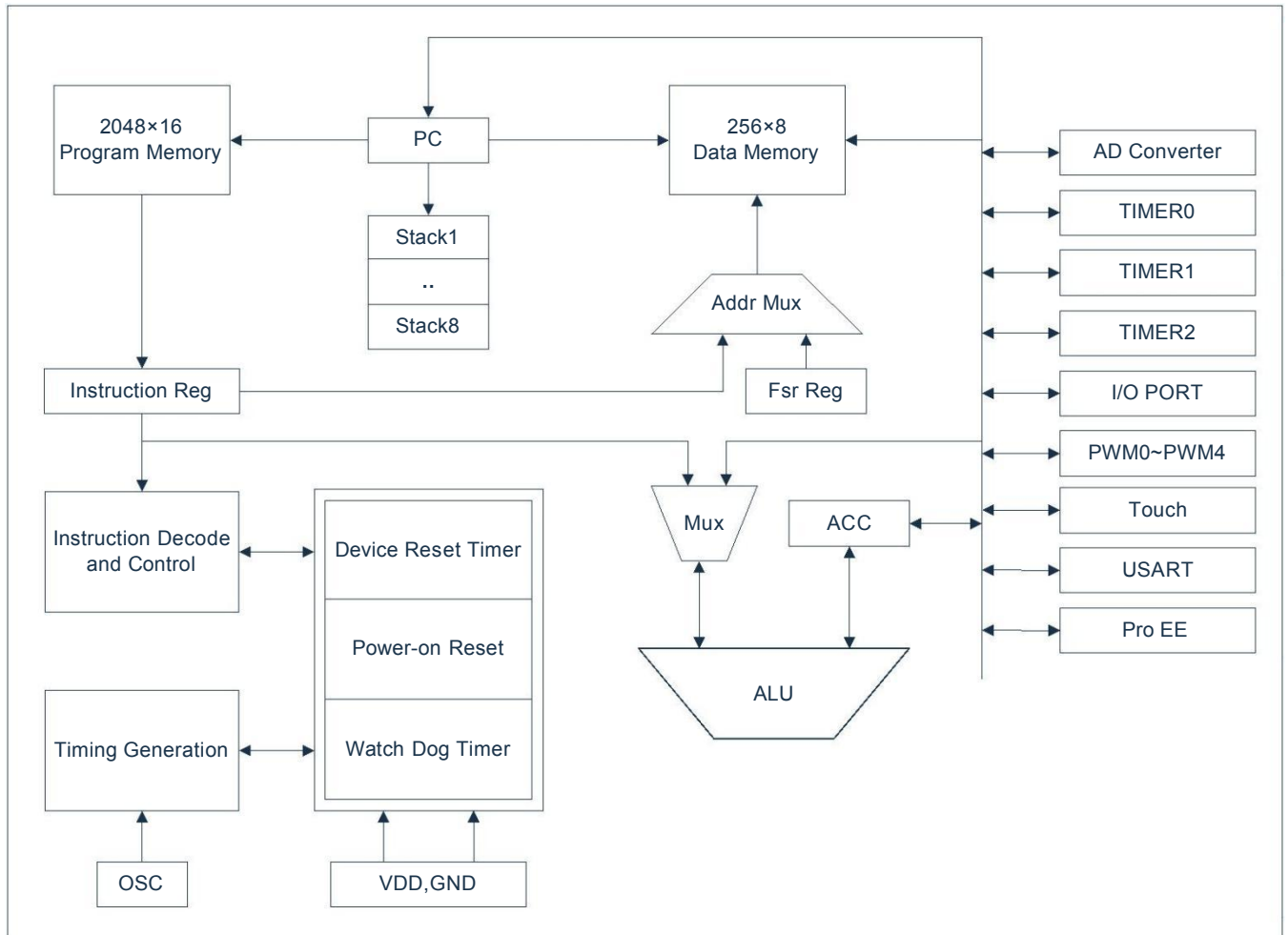
## Product specification

PRODUCT	ROM	RAM	Pro EE	I/O	Touch	ADC	USART	PACKAGE
CMS79FT611	2Kx16	256x8	128x8	6	4	12Bitx6	1	SOP8
CMS79FT613	2Kx16	256x8	128x8	14	7	12Bitx14	1	SOP16
CMS79FT616	2Kx16	256x8	128x8	18	8	12Bitx18	1	SOP20

Note: ROM---program memory    Pro EE---program EEPROM

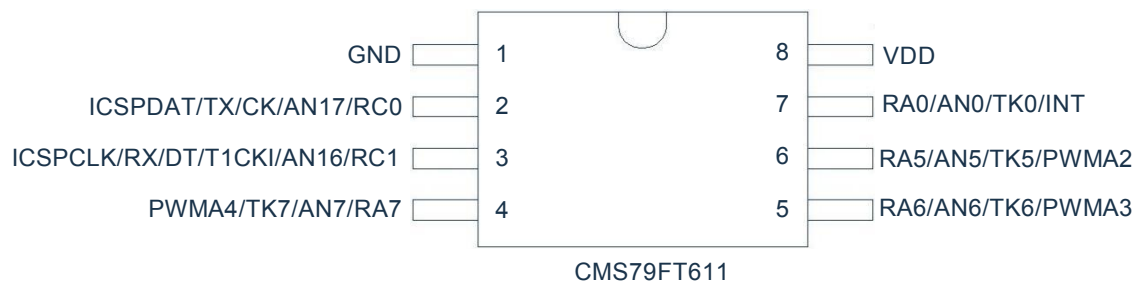


## 1.2 System Structure Diagram

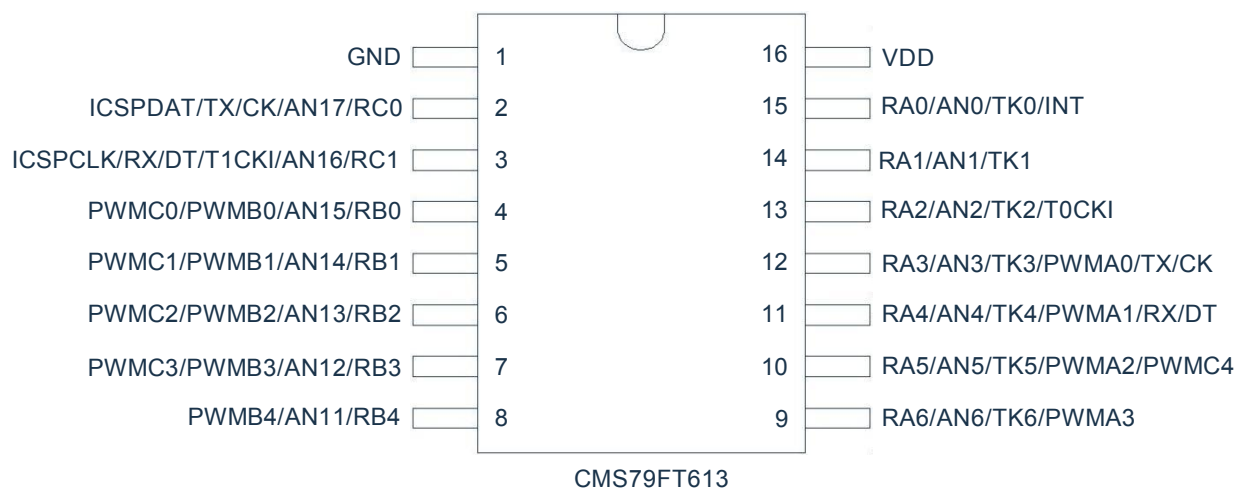


## 1.3 Pin Allocation

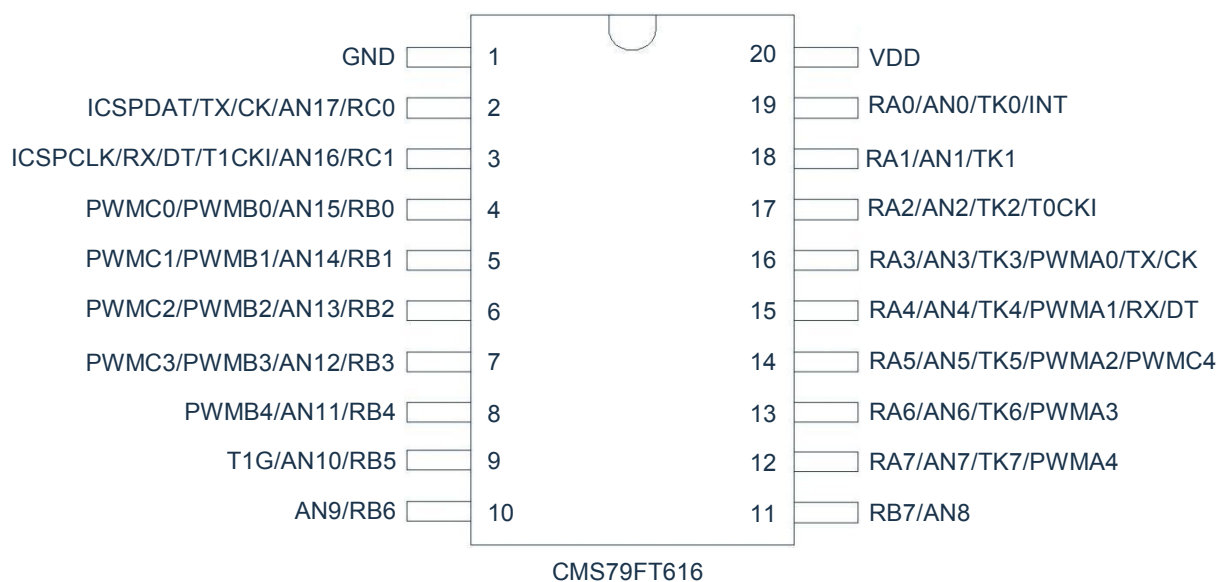
### 1.3.1 CMS79FT611 Pin Out Diagram



### 1.3.2 CMS79FT613 Pin Out Diagram



### 1.3.3 CMS79FT616 Pin Out Diagram



#### Note:

- 1) serial port interface function of RA3 and RA4 and RC0/RC1 serial port interface function is configured by CONFIG
- 2) PWMx function is configured by PWMCON1 register

## CMS79FT61x Pin description:

Pin name	IO type	description
V <sub>DD</sub> , GND	P	Voltage input pin and ground
RA0-RA7	I/O	Programmable in/ push-pull out pin, with pull-up/pull-down resistor function, electrical level interrupt function
RB0-RB7	I/O	Programmable in/ push-pull out pin, with pull-up/pull-down resistor function, electrical level interrupt function
RC0-RC1	I/O	Programmable in/ push-pull out pin, with pull-up/pull-down resistor function
ICSPCLK/ICSKCLK	I/O	Program clock/data pin
AN0-AN17	I	12 bits ADC input pin
T0CKI	I	TIMER0 external clock input pin
T1CKI	I	TIMER1 external clock input pin
T1G	I	TIMER1 gate control input pin
TX/CK	O	asynchronous transmit output/synchronous clock input/output pin (can configure at different IO port)
RX/DT	I	asynchronous receive input/synchronous data input/output pin (can configure at different IO port)
PWMx0-4	O	PWM0-4 output function (can configure at different IO port)
INT	I	External interrupt input pin
TK0-TK7	I	Touch key input pin

## 1.4 System Configuration Register

System configuration register (CONFIG) is the initial FLASH choice of the MCU. It can only be burned by CMS burner. User cannot visit. It includes the following:

1. INTRC\_SEL (internal oscillation frequency)
  - ◆ INTRC8M                      F<sub>HSI</sub> choose internal 8MHz RC oscillation
  - ◆ INTRC16M                   F<sub>HSI</sub> choose internal 16MHz RC oscillation
2. WDT (watchdog choice)
  - ◆ ENABLE                      Enable watchdog timer
  - ◆ DISABLE                    Disable watchdog timer
3. PROTECT (encryption)
  - ◆ DISABLE                    Disable FLASH code encryption
  - ◆ ENABLE                    Enable FLASH code encryption, after which the read value from burning the simulator is uncertain.
4. LVR\_SEL (low voltage detection selection)
  - ◆ 2.0V
  - ◆ 2.6V
5. SLEEP\_LVREN (sleep state LVR enable bit)
  - ◆ DISABLE                    In Sleep state LVR function closed
  - ◆ ENABLE                    In Sleep state LVR function turned on
6. USART\_SEL (TX/RX) (USART port selection)
  - ◆ RC0/RC1                    Select RC0 as TX port, RC1 as RX port
  - ◆ RA3/RA4                    Select RA3 as TX port, RA4 as RX port
7. ICSPPORT\_SEL (simulation port selection)
  - ◆ ICSP                        ICSPCLK、DAT port keep as simulation port, all functions disabled
  - ◆ NORMAL                    ICSPCLK、DAT port as normal port

## 1.5 Online Serial Programming

Can perform serial programming on MCU at the final application circuit. Programming is done through the following:

- Power wire
- Ground wire
- Data wire
- Clock wire

This ensures users to use un-programmed devices to make circuit and only program the MCU just before the product being delivered. Therefore, the latest version of firmware can be burned into the MCU.

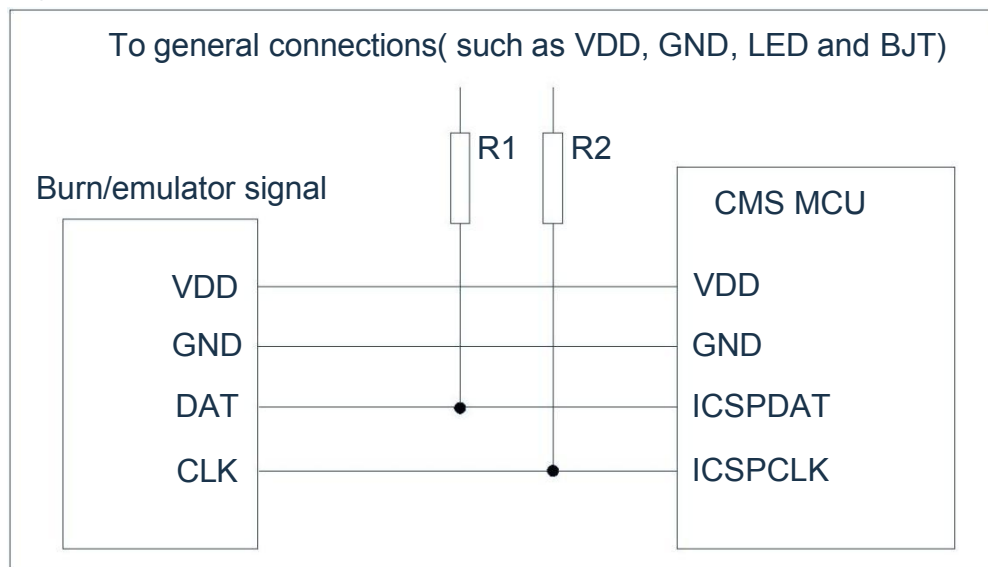


Fig 1-1: Typical connection for online serial programming

In the above figure, R1 and R2 are the electrical isolation devices, normally represented by resistor with the following resistance:  $R1 \geq 4.7K$ 、 $R2 \geq 4.7K$ .

## 2. Central Processing Unit (CPU)

### 2.1 Memory

#### 2.1.1 Program Memory

CMS79FT61x program memory space

FLASH:2K

0000H	reset vector	Program start, jump to user program
0001H		
0002H		
0003H		
0004H	interrupt vector	Interrupt entry, user interrupt program
...		User program area
...		
...		
7FDH		
7FEH		
7FFH	Jump to reset vector 0000H	Program ends

##### 2.1.1.1 Reset Vector (0000H)

MCU has 1-byte long system reset vector (0000H). It has 3 ways to reset:

- ◆ power-on reset
- ◆ watchdog reset
- ◆ low voltage reset (LVR)

When any above reset happens, program will start to execute from 0000H, system register will be recovered to default value. PD and TO from STATUS register can determine the which reset is performed from above. The following program illustrates how to define the reset vector from FLASH.

example: define reset vector

	ORG	0000H	;system reset vector
	JP	START	
	ORG	0010H	;start of user program
START:			
	...		;user program
	...		
	END		;program end

### 2.1.1.2 Interrupt Vector

The address for interrupt vector is 0004H. Once the interrupt responds, the current value for program counter PC will be saved to stack buffer and jump to 0004H to execute interrupt service program. All interrupt will enter 0004H. User will determine which interrupt to execute according to the bit of register of interrupt flag bit. The following program illustrate how to write interrupt service program.

example: define interrupt vector, interrupt program is placed after user program

	ORG	0000H	;system reset vector
	JP	START	
	ORG	0004H	;start of user program
INT_START:	CALL	PUSH	;save ACC and STATUS
	...		;user interrupt program
	...		
INT_BACK:	CALL	POP	;back to ACC and STATUS
	RETI		;interrupt back
START:			
	...		;user program
	...		
	END		;program end

Note: MCU does not provide specific unstack and push instructions, so user needs to protect interrupt scene.

example: interrupt-in protection

PUSH:			
	LD	ACC_BAK, A	;save ACC to ACC_BAK
	SWAPA	STATUS	;swap half-byte of STATUS
	LD	STATUS_BAK, A	;save to STATUS_BAK
	RET		;back

example: interrupt-out restore

POP:			
	SWAPA	STATUS_BAK	;swap the half-byte data from STATUS_BAK to ACC
	LD	STATUS, A	;pass the value in ACC to STATUS
	SWAPR	ACC_BAK	;swap the half-byte data in ACC_BAK
	SWAPA	ACC_BAK	; swap the half-byte data from ACC_BAK to ACC
	RET		;back

### 2.1.1.3 Look-up Table

Any address in FLASH can be use as look-up table.

Related instructions:

- TABLE [R] Pass the lower byte in table to register R, pass higher byte to TABLE\_DATAH.
- TABLEA Pass the lower byte in table to ACC, pass higher byte to TABLE\_DATAH.

related register:

- TABLE\_SPH (9AH) Read/write register to indicate higher 3 bits in the table.
- TABLE\_SPL (9BH) Read/write register to indicate lower 8 bits in the table.
- TABLE\_DATAH (94H) Read only register to save higher bit information in the table

Note: Write the table address into TABLE\_SPH and TABLE\_SP before using look-up. If main program and interrupt service program both use look-up table instructions, the value for TABLE\_SPH in the main program may change due to the look-up instructions from interrupt and hence cause error. Avoid using look-up table instruction in both main program and interrupt service. Disable the interrupt before using the look-up table instruction and enable interrupt after the look-up instructions are done.

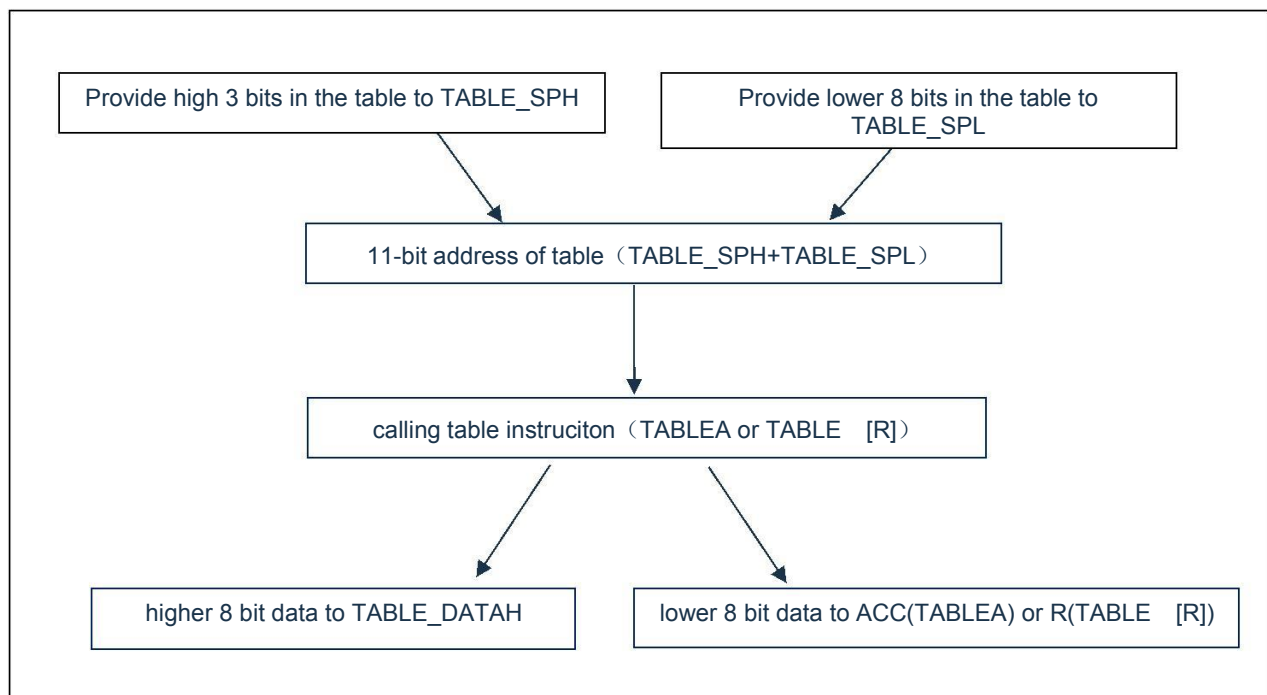


Fig 2-1: Flow chart for table usage



The following illustrates how to use the table in the program.

...		;continue from user program
LDIA	02H	;lower bits address in the table
LD	TABLE_SPL, A	
LDIA	06H	; higher bits address in the table
LD	TABLE_SPH, A	
TABLE	R01	;table instructions, pass the lower 8 bits (56H) to R01
LD	A, TABLE_DATAH	;pass the higher 8 bits from look-up table (34H) to ACC
LD	R02, A	;pass the value from ACC (34H)to R02
...		;user program
ORG	0600H	;start address of table
DW	1234H	;table content at 0600H
DW	2345H	;table content at 0601H
DW	3456H	;table content at 0602H
DW	0000H	;table content at 0603H

#### 2.1.1.4 Jump Table

Jump table can achieve multi-address jump feature. Since the addition of PCL and ACC is the new value of PCL, multi-address jump is then achieved through adding different value of ACC to PCL. If the value of ACC is n, then PCL+ACC represent the current address plus n. After the execution of the current instructions, the value of PCL will add 1 (refer to the following examples). If PCL+ACC overflows, then PC will not carry. As such, user can achieve multi-address jump through setting different values of ACC.

PCLATH is the PC high bit buffer register. Before operating on PCL, value must be given to PCLATH.

example: correct illustration of multi-address jump

FLASH address			
	LDIA	01H	
	LD	PCLATH, A	;must give value to PCLATH
	...		
0110H:	ADDR	PCL	;ACC+PCL
0111H:	JP	LOOP1	;ACC=0, jump to LOOP1
0112H:	JP	LOOP2	;ACC=1, jump to LOOP2
0113H:	JP	LOOP3	;ACC=2, jump to LOOP3
0114H:	JP	LOOP4	;ACC=3, jump to LOOP4
0115H:	JP	LOOP5	;ACC=4, jump to LOOP5
0116H:	JP	LOOP6	;ACC=5, jump to LOOP6

example: wrong illustration of multi-address jump

FLASH address			
	CLR	PCLATH	
	...		
00FCH:	ADDR	PCL	;ACC+PCL
00FDH:	JP	LOOP1	;ACC=0, jump to LOOP1
00FEH:	JP	LOOP2	;ACC=1, jump to LOOP2
00FFH:	JP	LOOP3	;ACC=2, jump to LOOP3
0100H:	JP	LOOP4	;ACC=3, jump to 0000H address
0101H:	JP	LOOP5	;ACC=4, jump to 0001H address
0102H:	JP	LOOP6	;ACC=5, jump to 0002H address

Note: Since PCI overflow will not carry to the higher bits, the program cannot be placed at the partition of the FLASH space when using PCL to achieve multi-address jump.

## 2.1.2 Data Memory

List of data memory of CMS79FT61x


Address		Address		Address		Address	
INDF	00H	INDF	80H	INDF	100H	INDF	180H
TMR0	01H	OPTION_REG	81H	----	101H	----	181H
PCL	02H	PCL	82H	PCL	102H	PCL	182H
STATUS	03H	STATUS	83H	STATUS	103H	STATUS	183H
FSR	04H	FSR	84H	FSR	104H	FSR	184H
PORTA	05H	TRISA	85H	PIR1	105H	----	185H
PORTB	06H	TRISB	86H	PIE1	106H	----	186H
WPUA	07H	WPDB	87H	PIR2	107H	----	187H
WPUB	08H	OSCCON	88H	PIE2	108H	----	188H
----	09H	WDTCON	89H	----	109H	----	189H
PCLATH	0AH	PCLATH	8AH	PCLATH	10AH	PCLATH	18AH
INTCON	0BH	INTCON	8BH	INTCON	10BH	INTCON	18BH
----	0CH	EECON1	8CH	TMR1L	10CH	----	18CH
IOCB	0DH	EECON2	8DH	TMR1H	10DH	----	18DH
PWMD23H	0EH	EEDAT	8EH	T1CON	10EH	----	18EH
PWM01DT	0FH	EEDATH	8FH	----	10FH	----	18FH
PWM23DT	10H	EEADR	90H	ANSEL0	110H	----	190H
TMR2	11H	PR2	91H	ANSEL1	111H	----	191H
T2CON	12H	PORTC	92H	ANSEL2	112H	----	192H
PWMCON0	13H	TRISC	93H	PAHEN	113H	----	193H
PWMCON1	14H	TABLE_DATAH	94H	PALEN	114H	----	194H
PWMTL	15H	IOCA	95H	PBHEN	115H	----	195H
PWMTH	16H	EEADRH	96H	PBLEN	116H	----	196H
PWMD0L	17H	WPDA	97H	TXSTA	117H	----	197H
PWMD1L	18H	WPDC	98H	RCSTA	118H	----	198H
PWMD2L	19H	WPUC	99H	SPBRG	119H	----	199H
PWMD3L	1AH	TABLE_SPH	9AH	TXREG	11AH	----	19AH
PWMD4L	1BH	TABLE_SPL	9BH	RCREG	11BH	----	19BH
PWMD01H	1CH	ADCON1	9CH	SEGCUR	11CH	----	19CH
PWMCON2	1DH	ADCON0	9DH	----	11DH	----	19DH
PWM4TL	1EH	ADRESH	9EH	----	11EH	----	19EH
----	1FH	ADRESL	9FH	----	11FH	----	19FH
	20H		A0H		120H		1A0H
General purpose registers 96 bytes		General purpose registers 80 bytes		General purpose registers 80 bytes		----	
		Fast memory region 70H-7FH		Fast memory region 70H-7FH		Fast memory region 70H-7FH	
BANK0		BANK1		BANK2		BANK3	

Data memory consists of  $512 \times 8$  bits. It can be divided into to space: special function register and universal data memory. Most of data memory are able to write/read data, only some data memory are read-only. Special register address is from 00H-1FH, 80-9FH, 100-11FH, 180-18BH.

**Summary of special registers in CMS79FT61x Bank0**

address	name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
00H	INDF	Addressing this unit will use FSR content for data register (not physical register)								xxxxxxx
01H	TMR0	TIMER0 data register								xxxxxxx
02H	PCL	Lower bit of program counter								00000000
03H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
04H	FSR	memory pointers for indirect addressing of data memory								xxxxxxx
05H	PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	xxxxxxx
06H	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxxxxx
07H	WPUA	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0	00000000
08H	WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0	00000000
0AH	PCLATH	----	----	----	----	----	Write buffer of higher 3 bits of program counter			-----000
0BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
0DH	IOCB	IOCB7	IOCB6	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	IOCB0	00000000
0EH	PWMD23H	----	----	PWMD3[9: 8]		----	----	PWMD2[9: 8]		--00--00
0FH	PWM01DT	----	----	PWM01 dead zone delay time						--000000
10H	PWM23DT	----	----	PWM23 dead zone delay time						--000000
11H	TMR2	TIMER2 mod register								00000000
12H	T2CON	----	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-0000000
13H	PWMCON0	CLKDIV[2: 0]			PWM4EN	PWM3EN	PWM2EN	PWM1EN	PWM0EN	00000000
14H	PWMCON1	PWMIO_SEL[1: 0]		PWM2DTEN	PWM0DTEN	----	----	DT_DIV[1: 0]		0000--00
15H	PWMTL	PWM period lower bit register								00000000
16H	PWMTH	----	----	PWMD4[9: 8]		PWM4T9	PWM4T8	PWMT9	PWMT8	--000000
17H	PWMD0L	PWM0 duty cycle lower bit register								00000000
18H	PWMD1L	PWM1 duty cycle lower bit register								00000000
19H	PWMD2L	PWM2 duty cycle lower bit register								00000000
1AH	PWMD3L	PWM3 duty cycle lower bit register								00000000
1BH	PWMD4L	PWM4 duty cycle lower bit register								00000000
1CH	PWMD01H	----	----	PWMD1[9: 8]		----	----	PWMD0[9: 8]		--00--00
1DH	PWMCON2	----	----	----	PWM4DIR	PWM3DIR	PWM2DIR	PWM1DIR	PWM0DIR	---00000
1EH	PWM4TL	PWM4 period lower bit register								00000000

**Summary of special registers in CMS79FT61x Bank1**

address	name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
80H	INDF	Addressing this unit will use FSR content for data register (not physical register)								xxxxxxx
81H	OPTION_REG	----	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	-1111011
82H	PCL	Lower bit of program counter								00000000
83H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
84H	FSR	memory pointers for indirect addressing of data memory								xxxxxxx
85H	TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	11111111
86H	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	11111111
87H	WPDB	WPDB7	WPDB6	WPDB5	WPDB4	WPDB3	WPDB2	WPDB1	WPDB0	00000000
88H	OSCCON	----	IRCF2	IRCF1	IRCF0	----	----	----	----	-110----
89H	WDTCON	----	----	----	----	----	----	----	SWDTEN	-----0
8AH	PCLATH	----	----	----	----	----	Write buffer of higher 3 bits of program counter			----000
8BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
8CH	EECON1	EEPGD	----	EETIME1	EETIME0	WRERR	WREN	WR	RD	0-00x000
8DH	EECON2	EEPROM control register2 (not physical register)								-----
8EH	EEDAT	EEDAT7	EEDAT6	EEDAT5	EEDAT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0	xxxxxxxx
8FH	EEDATH	EEDATH7	EEDATH6	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0	xxxxxxxx
90H	EEADR	EEADR7	EEADR6	EEADR5	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0	00000000
91H	PR2	TIMER2 period register								00000000
92H	PORTC	----	----	----	----	----	----	RC1	RC0	-----xx
93H	TRISC	----	----	----	----	----	----	TRISC1	TRISC0	-----11
94H	TABLE_DATAH	Data for high bits in table								xxxxxxx
95H	IOCA	IOCA7	IOCA6	IOCA5	IOCA4	IOCA3	IOCA2	IOCA1	IOCA0	00000000
96H	EEADRH	----	----	----	----	----	EEADRH2	EEADRH1	EEADRH0	----000
97H	WPDA	WPDA7	WPDA6	WPDA5	WPDA4	WPDA3	WPDA2	WPDA1	WPDA0	00000000
98H	WPDC	----	----	----	----	----	----	WPDC1	WPDC0	-----00
99H	WPUC	----	----	----	----	----	----	WPUC1	WPUC0	-----00
9AH	TABLE_SPH	----	----	----	----	----	Pointers for higher 3 bits of the table			----xxx
9BH	TABLE_SPL	Pointers for low bits in table								xxxxxxx
9CH	ADCON1	ADFM	CHS4	----	----	----	LDO_EN	LDO_SEL[1: 0]		00---000
9DH	ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/ 	ADON	00000000
9EH	ADRESH	ADC result register higher byte								xxxxxxx
9FH	ADRESL	ADC result register lower byte								xxxxxxx

**Summary of special registers in CMS79FT61x Bank2**

address	name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
100H	INDF	Addressing this unit will use FSR content for data register (not physical register)								xxxxxxx
102H	PCL	Lower bit of program counter								00000000
103H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
104H	FSR	memory pointers for indirect addressing of data memory								xxxxxxx
105H	PIR1	----	ADIF	RCIF	TXIF	----	PWMIF	TMR2IF	TMR1IF	-000-000
106H	PIE1	----	ADIE	RCIE	TXIE	----	PWMIE	TMR2IE	TMR1IE	-000-000
107H	PIR2	TKM1IF	TKM0IF	----	EEIF	----	----	RACIF	----	00-0—0-
108H	PIE2	TKM1IE	TKM0IE	----	EEIE	----	----	RACIE	----	00-0—0-
10AH	PCLATH	----	----	----	----	----	Write buffer of higher 3 bits of program counter			----000
10BH	INTCON	GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	00000000
10CH	TMR1L	16 bit TIMER1 register lower byte data register								xxxxxxx
10DH	TMR1H	16 bit TIMER1 register higher byte data register								xxxxxxx
10EH	T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	----	T1SYNC	TMR1CS	TMR1ON	000-0000
110H	ANSEL0	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0	00000000
111H	ANSEL1	ANS15	ANS14	ANS13	ANS12	ANS11	ANS10	ANS9	ANS8	00000000
112H	ANSEL2	----	----	----	----	----	----	ANS17	ANS16	-----00
113H	PAHEN	PAHEN7	PAHEN6	PAHEN5	PAHEN4	PAHEN3	PAHEN2	PAHEN1	PAHEN0	00000000
114H	PALEN	PALEN7	PALEN6	PALEN5	PALEN4	PALEN3	PALEN2	PALEN1	PALEN0	00000000
115H	PBHEN	PBHEN7	PBHEN6	PBHEN5	PBHEN4	PBHEN3	PBHEN2	PBHEN1	PBHEN0	00000000
116H	PBLEN	PBLEN7	PBLEN6	PBLEN5	PBLEN4	PBLEN3	PBLEN2	PBLEN1	PBLEN0	00000000
117H	TXSTA	CSRC	TX9EN	TXEN	SYNC	SCKP	STOPBIT	TRMT	TX9D	00000010
118H	RCSTA	SPEN	RX9EN	SREN	CREN	RCIDL	FERR	OERR	RX9D	00001000
119H	SPBRG	USART baud rate 8 bit register								00000000
11AH	TXREG	USART transmit data register								00000000
11BH	RCREG	USART receive data register								xxxxxxx
11CH	SEGCUR	----	----	----	----	----	SEG_ISEL[2: 0]			----000

### Summary of special registers in CMS79FT61x Bank3

address	name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
180H	INDF	Addressing this unit will use FSR content for data register (not physical register)								xxxxxxx
182H	PCL	Lower bit of program counter								00000000
183H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
184H	FSR	memory pointers for indirect addressing of data memory								xxxxxxx
18AH	PCLATH	---	---	---	---	---	Write buffer of higher 3 bits of program counter			----000
18BH	INTCON	GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF	00000000

## 2.2 Addressing Mode

### 2.2.1 Direct Addressing

Operate on RAM through accumulator (ACC)

example: pass the value in ACC to 30H register

LD	30H, A
----	--------

example: pass the value in 30H register to ACC

LD	A, 30H
----	--------

### 2.2.2 Immediate Addressing

Pass the immediate value to accumulator (ACC).

example: pass immediate value 12H to ACC

LDIA	12H
------	-----

### 2.2.3 Indirect Addressing

Data memory can be direct or indirect addressing. Direct addressing can be achieved through INDF register, INDF is not physical register. When load/save value in INDF, address is the value in FSR register (lower 8 bits) and IRP bit in STATUS register (9<sup>th</sup> bit) , and point to the register of this address. Therefore after setting the FSR register and the IRP bit of STATUS register, INDF register can be regarded as purpose register. Read INDF (FSR=0) indirectly will produce 00H. Write INDF register indirectly will cause an empty action. The following example shows how indirect addressing works.

example: application of FSR and INDF

LDIA	30H	
LD	FSR, A	;Points to 30H for indirect addressing
CLRB	STATUS, IRP	;clear the 9 <sup>th</sup> bit of pointer
CLR	INDF	;clear INDF, which mean clear the 30H address RAM that FSR points to

example: clear RAM (20H-7FH) for indirect addressing:

	LDIA	1FH	
	LD	FSR, A	;Points to 1FH for indirect addressing
	CLRB	STATUS, IRP	
LOOP:			
	INCR	FSR	;address add 1, initial address is 30H
	CLR	INDF	;clear the address where FSR points to
	LDIA	7FH	
	SUBA	FSR	
	SNZB	STATUS, C	;clear until the address of FSR is 7FH
	JP	LOOP	



## 2.3 Stack

Stack buffer of the chip has 8 levels. Stack buffer is not part of data memory nor program memory. It cannot be written nor read. Operation on stack buffer is through stack pointers, which also cannot be written nor read. After system resets, SP points to the top of the stack. When sub-program happens or interrupts happens, value in program counter (PC) will be transferred to stack buffer. When return from interrupt or return from sub-program, value is transferred back to PC. The following diagram illustrates its working principle.

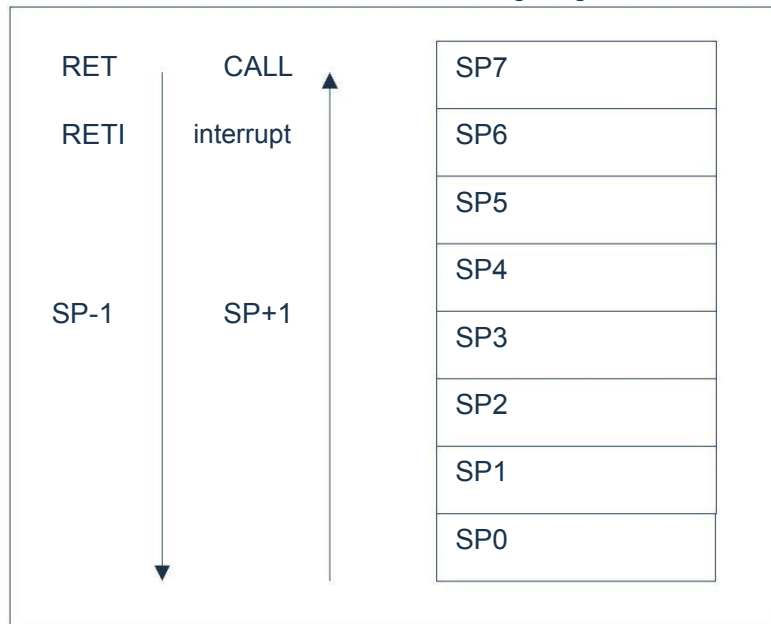


Fig 2-2: stack buffer working principle

Stack buffer will follow one principle: 'first in last out'

Note: stack buffer has only 8 levels, if the stack is full and interrupt happens which can not be screened out, then only the indication bit of the interrupt will be noted down. The response for the interrupt will be suppressed until the pointer of stack starts to decrease. This feature can prevent overflow of the stack caused by interrupt. Similarly, when stack is full and sub-program happens, then stack will overflow and the contents which enter the stack first will be lost, only the last 8 return address will be saved.

## 2.4 Accumulator (ACC)

### 2.4.1 General

ALU is the 8-bit arithmetic-logic unit. All math and logic related calculations in MCU are done by ALU. It can perform addition, subtraction, shift and logical calculation on data; ALU can also control STATUS to represent the status of the product of the calculation.

ACC register is a 8-bit register to store the product of calculation of ALU. It does not belong to data memory. It is in CPU and used by ALU during calculation. Hence it cannot be addressed. It can only be used through the instructions provided.

### 2.4.2 ACC Applications

example: use ACC for data transfer

LD	A, R01	;pass the value in register R01 to ACC
LD	R02, A	;pass the value in ACC to register R02

example: use ACC for immediate addressing

LDIA	30H	;load the ACC as 30H
ANDIA	30H	;run 'AND' between value in ACC and immediate number 30H, save the result in ACC
XORIA	30H	; run 'XOR' between value in ACC and immediate number 30H, save the result in ACC

example: use ACC as the first operand of the double operand instructions

HSUBA	R01	;ACC-R01, save the result in ACC
HSUBR	R01	;ACC-R01, save the result in R01

example: use ACC as the second operand of the double operand instructions

SUBA	R01	;R01-ACC, save the result in ACC
SUBR	R01	; R01-ACC, save the result in R01

## 2.5 Program Status Register (STATUS)

STATUS register includes:

- ◆ status of ALU.
- ◆ Reset status.
- ◆ Selection bit of Data memory (GPR and SFR)

Just like other registers, STATUS register can be the target register of any other instruction. If A instructions that affects Z, DC or C bit that use STATUS as target register, then it cannot write on these 3 status bits. These bits are cleared or set to 1 according to device logic. TO and PD bit also cannot be written. Hence the instructions which use STATUS as target instruction may not result in what is predicted.

For example, CLRSTATUS will clear higher 3 bits and set the Z bit to 1. Hence the value of STATUS will be 000u u1uu (u will not change.). Hence, it is recommended to only use CLRB, SETB, SWAPA and SWAPR instructions to change STATUS register because these will not affect any status bits.

program status register STATUS (03H)

03H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	1	1	X	X	X

Bit7	IRP: Selection bit of register memory (for indirect addressing) 1= Bank2 and Bank3 (100h-1FFh); 0= Bank0 and Bank1 (00h-FFh).
Bit6~Bit5	RP[1: 0]: Selection bit of memory; 00: Select Bank 0; 01: Select Bank 1; 10: Select Bank 2; 11: Select Bank 3.
Bit4	TO: Time out bit; 1= Power on or CLRWDT instructions or STOP instructions; 0= WDT time out.
Bit3	PD: Power down; 1= Power on or CLRWDT instructions; 0= STOP instructions.
Bit2	Z: Bit for result in zero; 1= Result is 0; 0= Result is not 0
Bit1	DC: Carry bit; 1= When 4 <sup>th</sup> bit of result carry to higher bit; 0= When 4 <sup>th</sup> bit of result does not carry to higher bit;
Bit0	C: Carry/borrow bit ; 1= When carry happens at the highest bit; 0= When no carry happens at the highest bit;

TO and PD bit can reflect the reason for reset of chip. The following is the events which affects the TO and PD and the status of TO and PD after these events.

events	TO	PD
Power on	1	1
WDT overflow	0	X
STOP instructions	1	0
CLRWDT instructions	1	1
sleep	1	0

Events which affect TO/PD

TO	PD	Reset reason
0	0	WDT overflow awaken MCU
0	1	WDT overflow non-sleep status
1	1	Power on

TO/PD status after reset

## 2.6 Pre-scaler (OPTION\_REG)

OPTION\_REG register can be read or written. Each control bit for configuration is as follow:

- ◆ TIMER0/WDT pre-scaler
- ◆ TIMER0

pre-scaler OPTION\_REG (81H)

81H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION_REG	-	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
Read/write	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	-	1	1	1	1	0	1	1

Bit7 Not used:

Bit6 INTEDG: Edge selection bit for triggering interrupt  
 1= INT pin rising edge triggered interrupt  
 0= INT pin falling edge triggered interrupt

Bit5 T0CS: Selection bit for TIMER0 clock source.  
 0= Internal instructions period clock ( $F_{SYS}/4$ ).  
 1= transition edge on T0CKI pin

Bit4 T0SE: Edge selection bit for TIMER0 clock source  
 0= Increase when T0CKI pin signal transit from low to high  
 1= Increase when T0CKI pin signal transit from high to low

Bit3 PSA: pre-scaler allocation  
 0= pre-scaler allocates to TIMER0 mod  
 1= pre-scaler allocates to WDT

Bit2~Bit0 PS2~PS0: configuration bit for pre-allocation parameters.

PS2	PS1	PS0	TMR0 frequency ratio	WDT frequency ratio
0	0	0	1: 2	1: 1
0	0	1	1: 4	1: 2
0	1	0	1: 8	1: 4
0	1	1	1: 16	1: 8
1	0	0	1: 32	1: 16
1	0	1	1: 64	1: 32
1	1	0	1: 128	1: 64
1	1	1	1: 256	1: 128

Pre-scaler register is a 8-bit counter. When surveil on register WDT, it is a post scaler; when it is used as timer or counter, it is called pre-scaler. There is only 1 physical scaler and can only be used for WDT or TIMER0, but not at the same time. This means that if it is used for TIMER0, the WDT cannot use pre-scaler and vice versa.

When used for WDT, CLRWDT instructions will clear pre-scaler and WDT timer

When used for TIMER0, all instruction related to writing TIMER0 (such as : CLR TMR0, SETB TMR0, 1 .etc.)will clear pre-scaler.

Whether TIMER0 or WDT uses pre-scaler is full controlled by software. This can be changed dynamically. To avoid unintended chip reset, when switch from TIMER0 to WDT, the following instructions should be executed.

CLRWDT		;clear WDT
LDIA	B'xxx1xxx'	;set new pre-scaler
LD	OPTION_REG, A	
CLRWDT		;clear WDT

When switch from WDT to TIMER0 mod, the following instructions should be executed.

CLRWDT		;clear WDT
LDIA	B'xxx0xxx'	;set new pre-scaler
LD	OPTION_REG, A	

Note: in order for TIMER0 to have 1: 1 pre-scaling, pre-scaler can be allocated to WDT through PSA position 1 of selection register.

## 2.7 Program Counter (PC)

program counter (PC) controls the instruction sequence in program memory FLASH, it can addressing the whole range of FLASH. After obtaining instruction code, PC will increase by 1 and point to the address of the next instruction code. When executing jump, passing value to PCL, sub-program, initializing reset, interrupt, interrupt return, sub-program return and other actions, PC will load the address which is related to the instruction, rather than the address of the next instruction.

When encountering condition jump instructions and the condition is met, the instruction read during the current instruction will be discarded and an empty instruction period will be inserted. After this, the correct instruction can be obtained. If not, the next instruction will follow the order.

Program counter (PC) is 11 Bit, user can access lower 8 bits through PCL (02H). The higher 3 bits cannot be accessed. It can hold address for  $2K \times 16$ Bit program. Passing a value to PCL will cause a short jump which range until the 256 address of the current page.

Note: When using PCL for short jump, it is needed to pass some value to PCLATH

The following are the value of PC under special conditions.

reset	PC=0000;
interrupt	PC=0004 (original PC+1 will be add to stack automatically);
CALL	PC=program defined address (original PC+1 will be add to stack automatically);
RET、RETI、RET i	PC=value coming out from stack;
Operating on PCL	PC[10: 8] unchanged, PC[7: 0]=user defined value;
JP	PC=program defined value;
Other instructions	PC=PC+1;

## 2.8 Watchdog Timer (WDT)

Watchdog timer is a self-oscillated RC oscillation timer. There is no need for any external devices. Even the main clock of the chip stops working, WDT can still function/ WDT overflow will cause reset.

### 2.8.1 WDT Period

WDT and TIMER0 share 8-bit pre-scaler. After all reset, default overflow period of WDT is 144ms. The WDT overflow period is 18ms\*pre-scaling parameter. If WDT overflow period needs to be changed, you can configure OPTION\_REG register. The overflow period is affected by environmental temperature, voltage of the power source and other parameter.

“CLRWDT” and “STOP” instructions will clear counting value inside the WDT timer and pre-scaler (when pre-scaler is allocated to WDT). WDT generally is used to prevent the system and MCU program from being out of control . Under normal condition, WDT should be cleared by “CLRWDT” instructions before overflow to prevent reset being generated. If program is out of control for some reason such that “CLRWDT” instructions is not able to execute before overflow, WDT overflow will then generate reset to make sure the system restarts. If reset is generated by WDT overflow, then ‘TO’ bit of STATUS will be cleared to 0. User can judge whether the reset is caused by WDT overflow according to this.

Note:

- 1) If WDT is used, ‘CLRWDT’ instructions must be placed somewhere in the program to make sure it is cleared before WDT overflow. If not, chip will keep resetting and the system cannot function normally.
- 2) It is not allowed to clear WDT during interrupt so that the main program ‘run away’ can be detected.
- 3) There should be 1 clear WDT in the main program. Try not to clear WDT inside the sub program, so that the protection feature of watchdog timer can be used largely.
- 4) Different chips has slightly different overflow time in watchdog timer. When setting clear time for WDT, try to leave extra time for WDT overflow time so that unnecessary WDT reset can be avoided.

### 2.8.2 Watchdog Timer Control Register WDTCON

WDTCON (89H)

89H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WDTCON	---	---	---	---	---	---	---	SWDTEN
R/W	---	---	---	---	---	---	---	R/W
Reset value	---	---	---	---	---	---	---	0

Bit7~Bit1  
Bit0

Not used,  
SWDTEN: Software enable or disable watchdog timer bit  
1= Enable WDT  
0= Disable WDT (reset value)

Note: if WDT configuration bit in CONFIG equals 1, then WDT is always enabled and is unrelated to the status of control bit of SWDTEN. if WDT configuration bit in CONFIG equals 0, then it is able to disable WDT using the control bit of SWDTEN.



## 3. System Clock

### 3.1 General

Clock signal is generated by internal oscillation, 4 non-overlapping orthogonal clock signals called Q1、Q2、Q3、Q4 are produced. Inside IC , each Q1 makes program counter (PC)increase 1, Q4 obtain this instruction from program memory unit and lock it inside instructions register. Compile and execute the instruction obtained between next Q1 and Q4, which means that 4 clock period for 1 executed instruction. The following diagram illustrate the time series of clock and execution of instruction period.

1 instruction period contains 4 Q period. The execution of instructions has pipeline structure. Obtaining instructions only require 1 instruction period, compiling and executing use another instruction period. Since pipeline structure is used, the effective executing time for every instruction is 1 instruction period. If 1 instruction cause PC address to change (such as JP), then the pre-loaded instruction code is useless and 2 instruction period is needed to complete this instruction. This is why every operation on PC consumes 2 clock period.

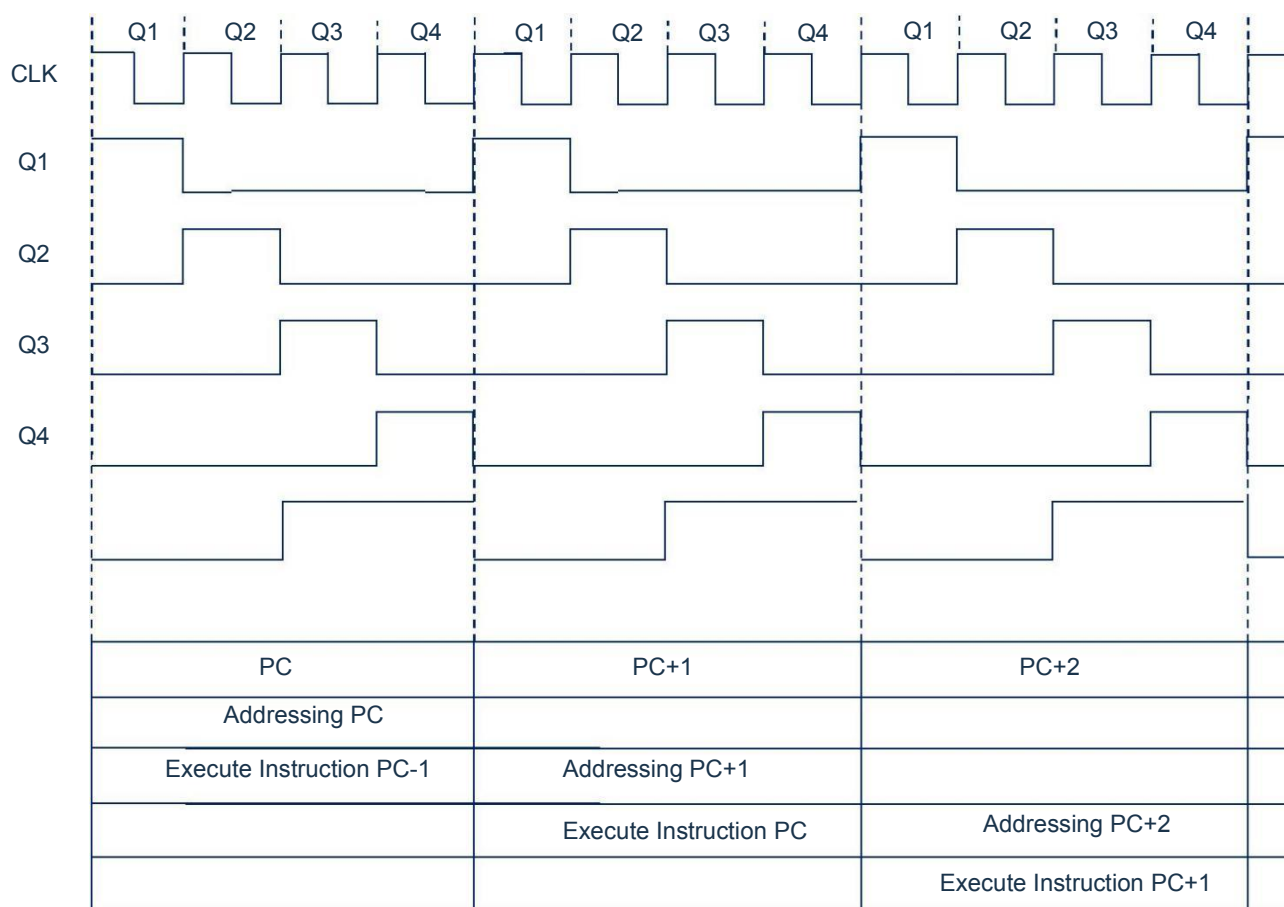


Fig 3-1: time series for clock and instruction period

Following is the relationship between working frequency of system and the speed of instructions:

System frequency ( $F_{sys}$ )	Double instruction period	Single instruction period
1MHz	8μs	4μs
2MHz	4μs	2μs
4MHz	2μs	1μs
8MHz	1μs	500ns

## 3.2 System Oscillator

Chip has built-in high precision internal RC oscillation.

Its frequency is 8MHz or 16MHz, which is set by OSCCON register.

## 3.3 Reset Time

Reset Time is the time for chip to change from reset to stable oscillation. The value is about 18ms.

Note: Reset time exists for both power on reset and other resets.

## 3.4 Oscillator Control Register

Oscillator control (OSCCON) register controls the system clock and frequency selection.

OSCCON (88H)

88H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OSCCON	---	IRCF2	IRCF1	IRCF0	---	---	---	-
R/W	---	R/W	R/W	R/W	---	---	---	-
reset value	---	1	1	0	---	---	---	-

Bit7	Not used, read 0
Bit6~Bit4	IRCF<2: 0>: Selection bit for frequency of Internal oscillator
	111= $F_{SYS} = F_{HSI} / 1$
	110= $F_{SYS} = F_{HSI} / 2$ (default)
	101= $F_{SYS} = F_{HSI} / 4$
	100= $F_{SYS} = F_{HSI} / 8$
	011= $F_{SYS} = F_{HSI} / 16$
	010= $F_{SYS} = F_{HSI} / 32$
	001= $F_{SYS} = F_{HSI} / 64$
	000= 32kHz (LFINTOSC).
Bit3~Bit0	Not used

Note:  $F_{HSI}$  as internal oscillator can select frequency of 8MHz or 16MHz;  $F_{SYS}$  is the working frequency of the system.

## 3.5 Clock Block Diagram

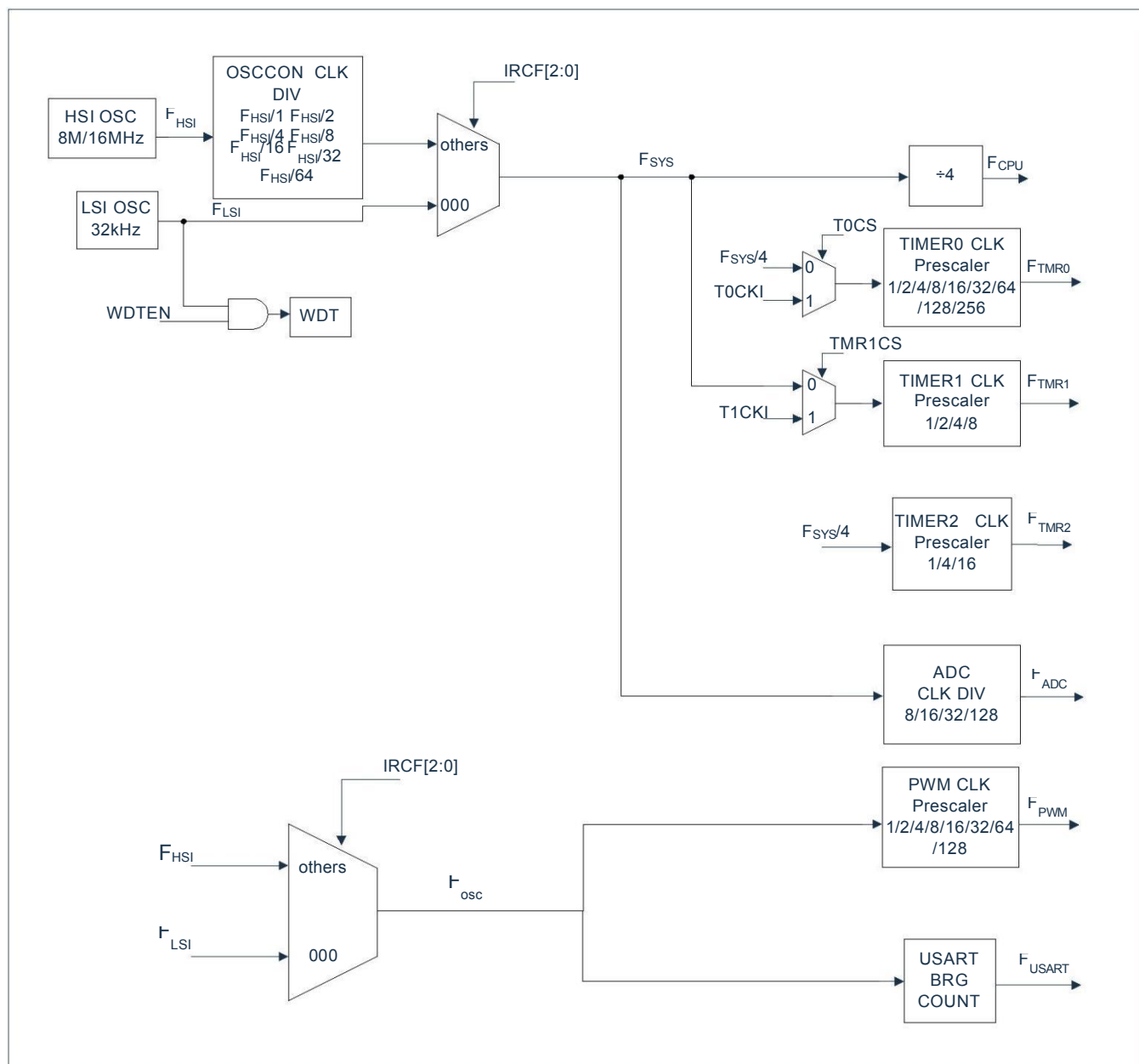


Fig 3-2: clock block diagram

## 4. Reset

Chip has 3 ways of reset:

- ◆ power on reset;
- ◆ low voltage reset;
- ◆ watchdog overflow reset under normal working condition;

When any reset happens, all system registers reset to default condition, program stops executing and PC is cleared. When finishing resetting, program executes from reset vector 0000H. TO and PD bit from STATUS can provide information for system reset (see STATUS). User can control the route of the program according to the status of PD and TO.

Any reset requires certain respond time. System provides completed reset procedures to make sure the reset is processed normally.

### 4.1 Power on Reset

Power on reset is highly related to LVR. Power on process of the systems should be increasing, after passing some time, the normal electrical level is then reached. The normal time series for power on is as follows:

- Power on: system detects the voltage of the source to increase and wait for it to stabilize;
- System initialization: all system register set to initial value;
- Oscillator starts working: oscillator starts to provide system clock;
- Executing program: power on process ends, program starts to be executed.

## 4.2 Power off Reset

### 4.2.1 Power off Reset General

Power off reset is used for voltage drop caused by external factors (such as interference or change in external load). Voltage drop may enter system dead zone. System dead zone means power source cannot satisfy the minimal working voltage of the system.

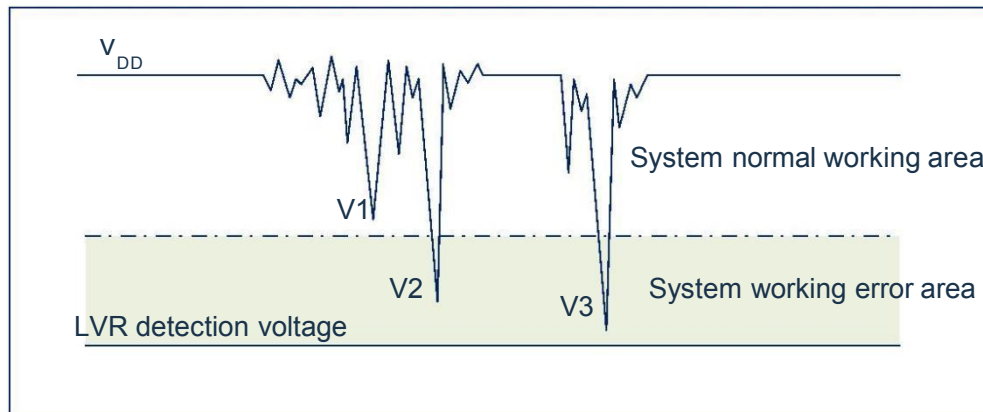


Fig 4-1: power off reset

The above is a typical power off reset case.  $V_{DD}$  is under serious interference and the voltage is dropped to a low value. The system works normally above the dotted line and the system enters an unknown situation below the dotted line. This zone is called dead zone. When  $V_{DD}$  drops to  $V1$ , system still works normally. When  $V_{DD}$  drops to  $V2$  and  $V3$ , system enters the dead zone and may cause error.

System will enter the dead zone under the following situation:

- DC:
  - Battery provides the power under DC. When the voltage of the battery is too low or the driver of MCU is over-loaded, system voltage may drop and enter the dead zone. Here, power source will not drop further to LVD detection voltage, hence system remains staying at the dead zone.
- AC:
  - When the system is powered by AC, voltage of DC is affected by the noise in AC source. When external over-loaded, such as driving motor, this action will also interfere the DC source.  $V_{DD}$  drops below the minimal working voltage due to interference, system may enter unstable working condition.
  - Under AC condition, system power on/off take long time. Power on protection can ensure the system to power on normally, but power off situation is similar to DC case, when AC source is off,  $V_{DD}$  drops and may enter dead zone easily.

As illustrated in the above diagram, the normal working voltage is higher than the system reset voltage, at the same time, reset voltage is decided by LVR. When the execution speed increases, the minimal working voltage should increase. However, the system reset voltage is fixed, hence there is a dead zone between the minimal working voltage and system reset voltage.

### 4.2.2 Improvements for Power off Reset

Suggestions to improve the power off reset:

- ◆ Choose higher LVR voltage;
- ◆ Turn on watchdog timer;
- ◆ Lower working frequency of the system;
- ◆ Increase the gradient of the voltage drop.

#### Watchdog timer

Watchdog timer is used to make sure the program is run normally. When system enter the dead zone or error happens, watchdog timer overflow and system reset.

#### Lower the working speed of the system

Higher the working frequency, higher the minimal working voltage system. Dead zone is increase when system works at higher frequency. Therefore lower the working speed can lower the minimal working voltage and then decrease the probability of entering the dead zone.

#### Increase the gradient of the voltage drop

This method is used under AC. Voltage drops slowly under AC and cause the system to stay longer at the dead zone. If the system is power on at this moment, error may happen. It is then suggested to insert a resistor between power source and ground to ensure the MCU pass the dead zone and enter the reset zone faster.

## 4.3 Watchdog Reset

Watchdog reset is a protection for the system. Under normal condition, program clear the watchdog timer. If error happens and system is under unknown status, watchdog timer overflow and then system reset. After watchdog reset, system restarts and enter normal working condition.

Time series for watchdog reset:

- Watchdog timer status : system detects watchdog timer. If overflow, then system reset;
- initialization: all system register set to default;
- oscillator starts working: oscillator starts to provide system clock;
- program: reset ends, program starts to be executed.

For applications of watchdog timer, see chapters at 2.8

## 5. Sleep Mode

### 5.1 Enter Sleep Mode

System can enter sleep mode when executing STOP instructions. If WDT enabled, then:

- ◆ WDT is cleared and continue to run.
- ◆ PD bit in STATUS register is cleared.
- ◆ TO bit set to 1.
- ◆ I/O port keep at the status before STOP (driver is high level, low lower, or high impedance).

Under sleep mode, to avoid current consumption, all I/O pin should keep at  $V_{DD}$  or GND to make sure no external circuit is consuming the current from I/O pin. To avoid input pin suspend and invoke current, high impedance I/O should be pulled to high or low level externally. Internal pull up resistance should also be considered.

### 5.2 Awaken from Sleep Mode

Awaken through any of the following events:

1. Watchdog timer awake (WDT force enable)
2. PORTA electrical level interrupt or PORTB electrical level interrupt
3. Other peripherals interrupt

The above 2 events are regards as the extension of the execution of the program. TO and PD bit in STATUS register are used to find the reason for reset. PD is set to 1 when power on and clear to 0 when STOP instruction is executing. TO is cleared when WDT awaken happens.

When executes STOP instructions, next instruction (PC+1) is withdrawn first. If it is intended to awaken the system using interrupt, the corresponding enable bit should be set to 1 for the interrupt. Awaken is not related to GIE bit. If GIE is cleared, system will continue to execute the instruction after STOP instruction, and then jump to interrupt address (0004h) to execute. To avoid instruction after STOP instruction being executed, user should put one NOP instruction after STOP instruction. When system is awakened from sleep mode, WDT will be cleared to 0 and has nothing to do with the reason for awakening.

## 5.3 Interrupt Awakening

When forbidden overall interrupt (GIE clear), and there exist 1 interrupt source with its interrupt enable bit and indication bit set to 1, one event from the following will happen:

- If interrupt happens before STOP instructions, then STOP instruction is executed as NOP instructions. Hence, WDT and its pre-scaler and post-scaler will not be cleared, and TO bit will not be set to 1, PD will not be cleared to 0.
- If interrupt happens during or after STOP instruction, then system is awoken from sleep mode. STOP will be executed before system being fully awoken. Hence, WDT and its pre-scaler, post-scaler will be cleared to, TO bit set to 1 and PD bit cleared to 0. Even if the indication bit is 0 before executing the STOP instruction, it can be set to 1 before STOP instruction is finished. To check whether STOP is executed, PD bit can be checked, if is 1, then STOP instruction is executed as NOP. Before executing STOP instruction, 1 CLRWDT instruction must be executed to make sure WDT is cleared.

## 5.4 Sleep Mode Application

Before system enters sleep mode, if user wants small sleep current, please check all I/O status. If suspended I/O port is required by user, set all suspended ports as output to make sure each I/O has a fixed status and avoid increasing sleep current when I/O is input; turn off AD and other peripherals mod; WDT functions can be turned off to decrease the sleep current.

example: procedures for entering sleep mode

SLEEP_MODE:			
CLR	INTCON		; disable interrupt
LDIA	B'00000000'		
LD	TRISA, A		
LD	TRISB, A		;all I/O set as output
LD	TRISC, A		
...			;turn off other functions
LDIA	0A5H		
LD	SP_FLAG, A		;set sleep status memory register
CLRWDT			;clear WDT
STOP			;execute STOP instruction

## 5.5 Sleep Mode Awaken Time

When MCU is awoken from sleep mode, oscillation reset time is needed. This time is  $136 * T_{\text{SYS}}$  clock period under internal high speed oscillation mode,  $12 * T_{\text{SYS}}$  clock period under low speed oscillation mode, detailed relationship as shown in table below.

System main clock source	System clock frequency (IRCF<2: 0>)	$T_{\text{WAIT}}$
Internal high speed RC oscillation ( $F_{\text{HSI}}$ )	$F_{\text{SYS}} = F_{\text{HSI}}$	$T_{\text{WAIT}} = 136 * 1 / F_{\text{HSI}}$
	$F_{\text{SYS}} = F_{\text{HSI}} / 2$	$T_{\text{WAIT}} = 136 * 2 / F_{\text{HSI}}$
	...	...
	$F_{\text{SYS}} = F_{\text{HSI}} / 64$	$T_{\text{WAIT}} = 136 * 64 / F_{\text{HSI}}$
Internal low speed RC oscillation ( $F_{\text{LFINTOSC}}$ )	----	$T_{\text{WAIT}} = 11 / F_{\text{LFINTOSC}}$



## 6. I/O port

Chip has 3 I/O port: PORTA、PORTB、PORTC (max. of 18 I/O).read/write port data register can directly read/write these ports.

port	bit	Pin description	I/O
PORTA	0	Schmitt trigger input, pull-up/pull-down input, push-pull output, AN0, touch input, external interrupt input	I/O
	1	Schmitt trigger input, pull-up/pull-down input, push-pull output, AN1, touch input	I/O
	2	Schmitt trigger input, pull-up/pull-down input, push-pull output, AN2, touch input, TMR0 clock input	I/O
	3	Schmitt trigger input, pull-up/pull-down input, push-pull output, AN3, touch input, PWM output, asynchronized serial port output, synchronize serial port clock	I/O
	4	Schmitt trigger input, pull-up/pull-down input, push-pull output, AN4, touch input, PWM output, asynchronized serial port input, synchronize serial port data	I/O
	5	Schmitt trigger input, pull-up/pull-down input, push-pull output, AN5, touch input, PWM output, TMR1 gate control input	I/O
	6	Schmitt trigger input, pull-up/pull-down input, push-pull output, AN6, touch input, PWM output	I/O
	7	Schmitt trigger input, pull-up/pull-down input, push-pull output, AN7, touch input, PWM output	I/O
PORTB	0	Schmitt trigger input, pull-up/pull-down input, push-pull output, AN15, touch input, PWM output	I/O
	1	Schmitt trigger input, pull-up/pull-down input, push-pull output, AN14, touch input, PWM output	I/O
	2	Schmitt trigger input, pull-up/pull-down input, push-pull output, AN13, touch input, PWM output	I/O
	3	Schmitt trigger input, pull-up/pull-down input, push-pull output, AN12, touch input, PWM output	I/O
	4	Schmitt trigger input, pull-up/pull-down input, push-pull output, AN11, touch input, PWM output	I/O
	5	Schmitt trigger input, pull-up/pull-down input, push-pull output, AN10	I/O
	6	Schmitt trigger input, pull-up/pull-down input, push-pull output, AN9	I/O
	7	Schmitt trigger input, pull-up/pull-down input, push-pull output, AN8	I/O
PORTC	0	Schmitt trigger input, pull-up/pull-down input, push-pull output, AN17, programming data input/output, asynchronized serial port output, synchronize serial port clock	I/O
	1	Schmitt trigger input, pull-up/pull-down input, push-pull output, AN18, programming clock input, TMR1 clock input, asynchronized serial port input, synchronize serial port data	I/O

< Table 6-1: port configuration summary >

## 6.1 I/O Port Structure

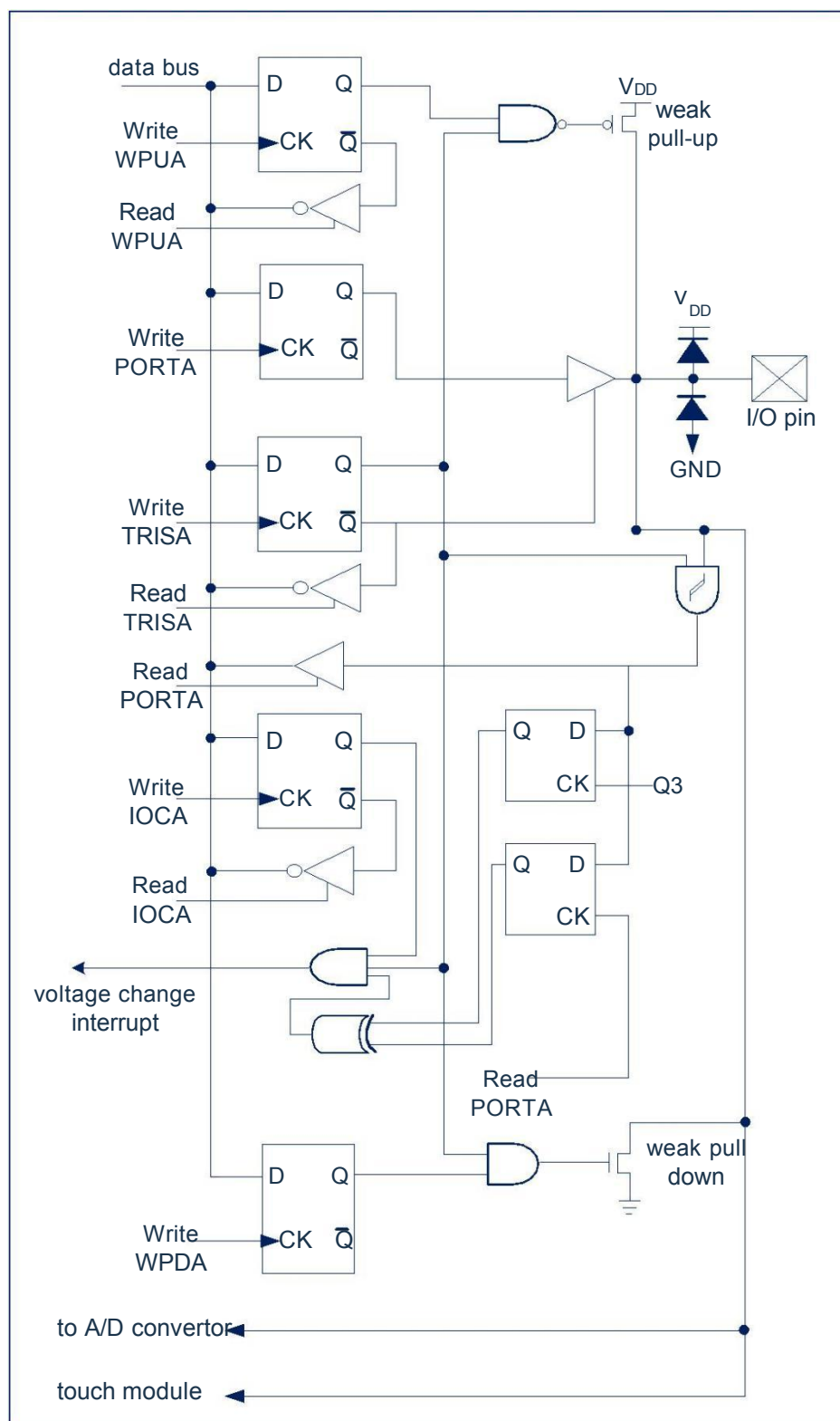
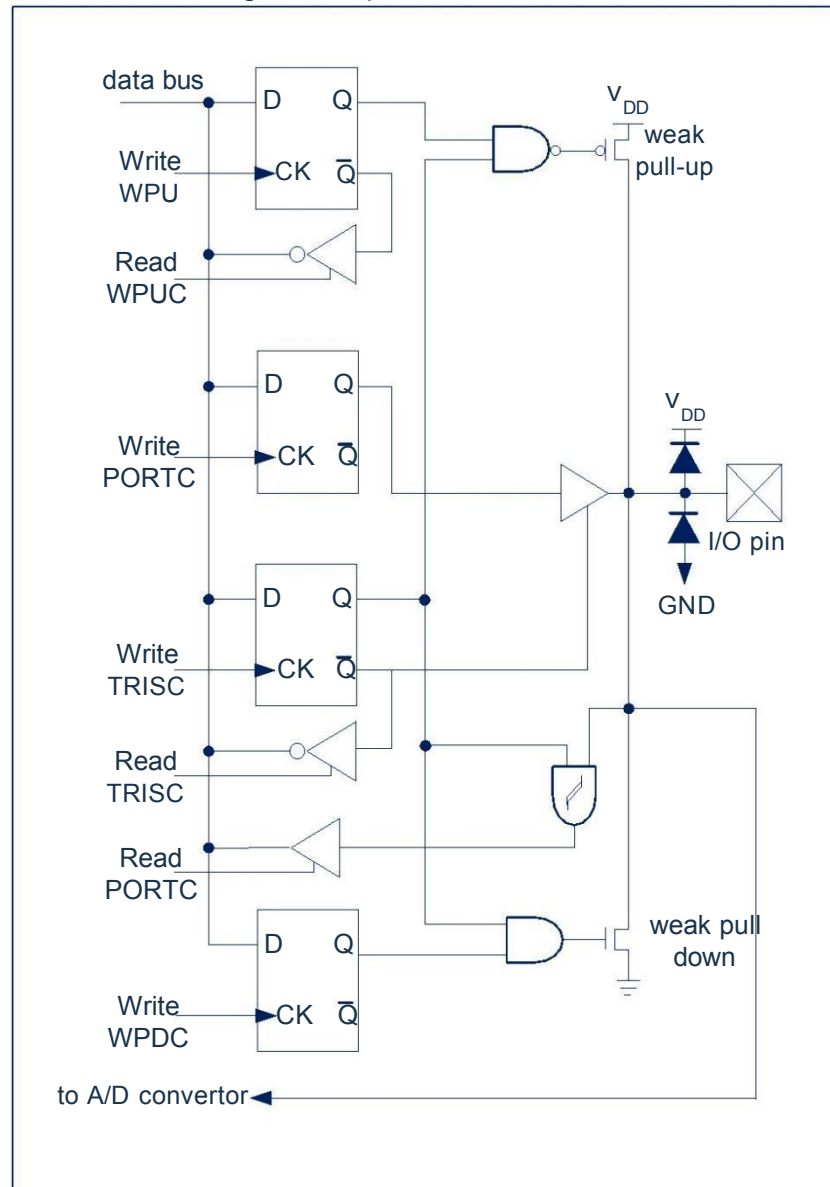


Fig 6-1: I/O port structure PORTA

Fig 6-2: I/O port structure PORTB



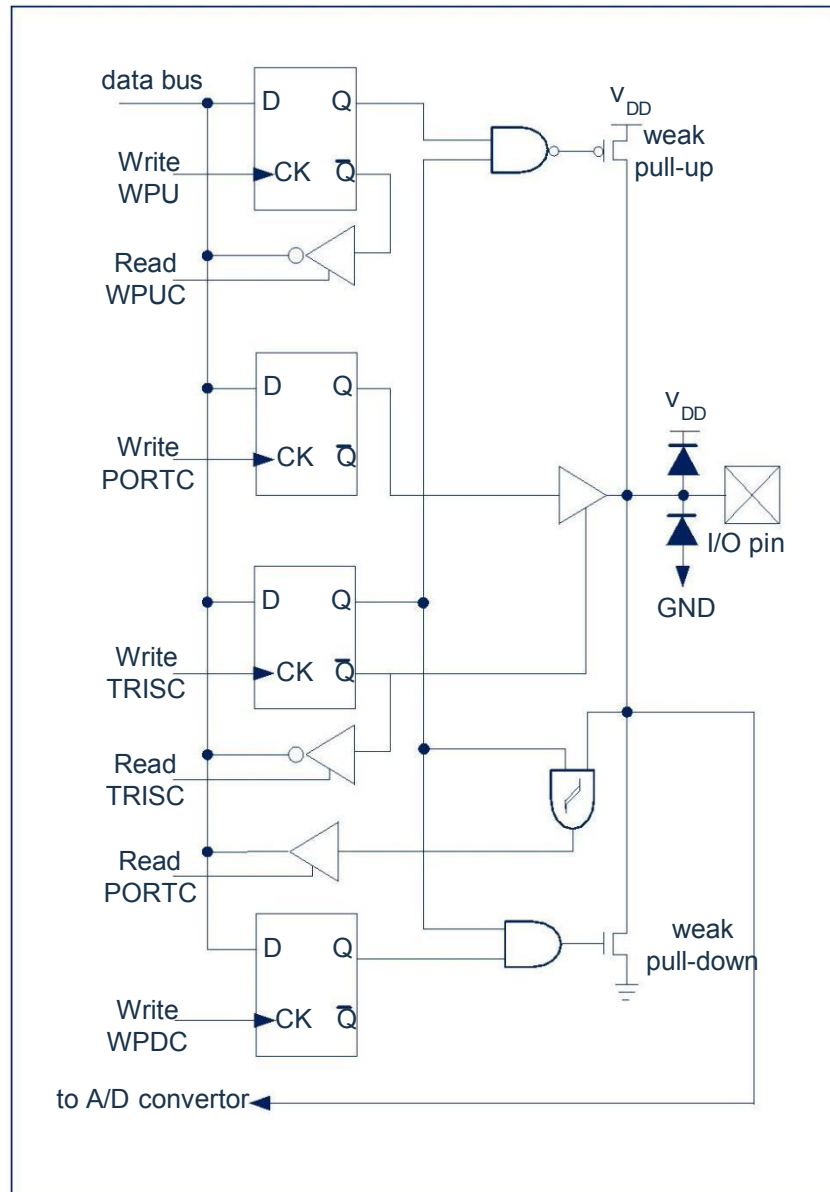


Fig 6-3: I/O port structure PORTC

## 6.2 PORTA

### 6.2.1 PORTA Data and Direction Control

PORTA is 8 Bit bi-directional port. Its corresponding data direction register is TRISA. Setting 1 bit of TRISA to be 1 can configure the corresponding pin to be input. Setting 1 bit of TRISA to be 0 can configure the corresponding pin to be output.

Reading PORTA register reads the pin status. Writing PORTA write to port latch. All write operation are read-change-write. Hence, write 1 port means read the pin electrical level of the port, change the value and write the value into port latch. Even when PORTA pin is used as analog input, TRISA registers still control the direction of PORTA pin. When use PORTA pin as analog input, user must make sure the bits in TRISA register are kept as 1.

Registers related to PORTA ports are PORTA, TRISA, WPUA, WPDA, IOCA, ANSEL0 and etc.

PORTA data register PORTA (05H)

05H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
reset value	X	X	X	X	X	X	X	X

Bit7~Bit0      PORTA<7: 0>: PORTA I/O pin bit;  
                   TRISAx=1  
                   1= Port pin level>V<sub>IH</sub>;  
                   0= Port pin level<V<sub>IL</sub>.  
                   TRISAx=0  
                   1= Port output high level;  
                   0= Port output low level.

PORTA direction register TRISA (85H)

85H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
reset value	1	1	1	1	1	1	1	1

Bit7~Bit0      TRISA<7: 0>: PORTA;  
                   1= PORTA pin set to be input;  
                   0= PORTA pin set to be output

Example: procedure for PORTA

LDIA	B'11110000'	;set PORTA<3: 0> as output port, PORTA<7: 4>as input port
LD	TRISA, A	
LDIA	03H	;PORTA<1: 0>output high level, PORTA<3: 2>output low level
LD	PORTA, A	;since PORTA<7: 4>are input ports, 0 or 1 does not matter

### 6.2.2 PORTA Pull-up Resistance

Each PORTA pin has an internal weak pull up that can be individually configured. The control bits WPUA<7: 0> enable or disable each weak pull up. When the port pin is configured as output, its weak pull up will be automatically cut off.

PORTA pull up resistance register WPUA (07H)

07H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUA	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 WPUA<7: 0>: Weak pull up register bit  
1= Enable pull up  
0= Disable pull up

Note: If pin is configured as output, weak pull up will be automatically disabled

### 6.2.3 PORTA Pull-down Resistor

Each PORTA pin has an individually configurable internal weak pull-down. The control bits WPDA < 7: 0 > enable or disable each weak pull-down. When the port pin is configured as an output, its weak pulldown is automatically cut off.

PORTA pull-up resistor register WPDA (97H).

97H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPDA	WPDA7	WPDA6	WPDA5	WPDA4	WPDA3	WPDA2	WPDA1	WPDA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 WPDA<7: 0>: Weak pull-down register bits.  
1= Enable pull-down.  
0= Pull-down is prohibited.

Note: Weak pulldowns are automatically disabled if the pins are configured as outputs.

## 6.2.4 PORTA Analog Control Selection

The ANSEL0 register is used to configure the input mode of I/O pin to analog mode. Setting the appropriate bit in ANSEL0 to 1 will cause all digital read operations of the corresponding pin to return to 0 and make the analog function of the pin work normally. The state of the ANSEL0 bit has no effect on the digital output function. The pin with TRIS cleared and ANSEL0 set to 1 will still be used as a digital output, but the input mode will become an analog mode. This can cause unpredictable results when performing read-modify-write operations on the affected port.

PORT A analog selection register ANSEL0 (110H)

110H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL0	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0      ANS<7: 0>: Analog selection bit, select the digital or analog function of pin AN<7: 0>  
                          1= Analog input  
                          0= Digital I/O

## 6.2.5 PortA Level Change Interrupt

All PORTA pins can be individually configured as level-change interrupt pins. The control bit IOCA < 7: 0> allow or disable this interrupt function on each pin. Disables the level change interrupt function of the pin during power-on reset.

For a pin that has allowed a level change interrupt, the value on that pin is compared to the old value latched at the last time the PORTA was read. The output of the "mismatch" of the last read operation will be logically or computationally performed to place the PORTA level change interrupt flag (RACIF) in the PIR2 register to 1.

The interrupt wakes the device from hibernation and the user can clear the interrupt in the interrupt service program in the following ways:

1. Read or write to PORTA. This ends the mismatched state of the pin levels.
2. Zero out the flag bit RACIF.

The mismatch state will continuously place the RACIF flag at position 1. Read or write PORTA will end the mismatch state and allow the RACIF flag bit to be zeroed. The latch will keep the last read value unaffected by undervoltage reset. After the reset, if no match persists, the RACIF flag bit will continue to be set at 1.

**Note:** If you perform a read operation (Q2Start of the cycle)/O the level of the pin changes RANDIF the interrupt flag bit is not set1. In addition, because reads or writes to a port affect all bits of that port, special care must be taken when using multiple pins in level-varying interrupt mode. You may not notice a level change on one pin when dealing with a change in level on the other pin.

PORTA level-varying interrupt register IOKA (95H).

95H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOCA	IOCA7	IOCA6	IOCA5	IOCA4	IOCA3	IOCA2	IOCA1	IOCA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0      IOCA<7: 0>      The level change of the PORTA interrupts the control bit.

1=      Allow level changes to be interrupted.

0=      Disables interrupts for level changes.



## 6.2.6 PORTA Drive Current Control

All PORTA pins can select different outputs of high-level drive current and low-level drive current, placing PAHEN registers. Somewhere in 1 (=1) can make the output of the corresponding PORTA pin drive current high. Selected, the current size selection is controlled by the SEGCUR register. Placing a position in the PALEN register at 1 (=1) can make the output of the corresponding PORTA pin Low drive current is optional.

PORTA high-level drive current control register PAHEN (113H).

113H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PAHEN	PAHEN7	PAHEN6	PAHEN5	PAHEN4	PAHEN3	PAHEN2	PAHEN1	PAHEN0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PAHEN<7: 0> The high-level output of the PORTA drives the current control bit.  
 1= Allows PORTA high-level drive current selectable (0mA to 28mA);  
 0= Disable high level drive current optional, default is (28mA).

PORTA drives the low-level current control register PALEN (114H).

114H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PALEN	PALEN7	PALEN6	PALEN5	PALEN4	PALEN3	PALEN2	PALEN1	PALEN0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PALEN<7: 0> The portA's low-level output drives the current control bit.  
 1= Large drive current (120mA);  
 0= Small drive current (60mA).

PORTA/PORTB high-level drive current control register SEGCUR (11CH).

11CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEGCUR	---	---	---	---	---	SEG_ISEL[2: 0]		
R/W	---	---	---	---	---	R/W	R/W	R/W
Reset value	---	---	---	---	---	0	0	0

Bit7~Bit3 Unused  
 Bit2~Bit0 SEG\_ISEL<2: 0> The high-level output of PORTA/PORTB drives the current select bit.  
 000= 0mA;  
 001= 4mA;  
 010= 8mA;  
 011= 12mA;  
 100= 16mA;  
 101= 20mA;  
 110= 24mA;  
 111= 28mA.

## 6.3 PORTB

### 6.3.1 PORTB Data and Direction

PORTB is a 8Bit wide bi-directional port. The corresponding data direction register is TRISB. Set a bit in TRISB to 1 (=1) to make the corresponding PORTB pin as the input pin. Clearing a bit in TRISB (=0) will make the corresponding PORTB pin as the output pin.

Reading the PORTB register reads the pin status and writing to the register will write the port latch. All write operations are read-modify-write operations. Therefore, writing a port means to read the pin level of the port first, modify the read value, and then write the modified value into the port data latch. Even when the PORTB pin is used as an analog input, the TRISB register still controls the direction of the PORTB pin. When using the PORTB pin as an analog input, the user must ensure that the bits in the TRISB register remain set as 1.

Related registers with PORTB port include PORTB, TRISB, WPUB, WPDB, ANSEL1, etc.

PORTB data register PORTB (06H)

06H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	X	X	X	X	X	X	X	X

Bit7~Bit0      PORTB<7: 0>:    PORTB I/O pin bit  
                          TRISBx=1  
                          1=    Port pin level >V<sub>IH</sub>;  
                          0=    Port pin level <V<sub>IL</sub>  
                          TRISBx=0  
                          1=    Port output high level;  
                          0=    Port output low level

PORTB direction register TRISB (86H)

86H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	1	1	1	1	1	1	1	1

Bit7~Bit0      TRISB<7: 0>:    PORTB tri-state control bit  
                          1=    PORTB pin configured as input  
                          0=    PORTB pin configured as output

Example: PORTB port procedure

CLR	PORTB	;clear data register
LDIA	B'00110000'	;set PORTB<5: 4> as input port, others as output port
LD	TRISB, A	

### 6.3.2 PORTB Pull-up Resistance

Each PORTB pin has an internal weak pull up that can be individually configured. The control bits WPUB<7: 0> enable or disable each weak pull up. When the port pin is configured as output, its weak pull up will be automatically cut off.

PORTB pull up resistance register WPUB (08H)

08H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0      WPUB<7: 0>: Weak pull up register bit  
                          1= Enable pull up  
                          0= Disable pull up

Note: If the pin is configured as output or analog input, weak pull up will be automatically prohibited.

### 6.3.3 PORTB Pull-down Resistor

Each PORTB pin has an internal weak pulldown that can be individually configured. The control bits WPDB < 7: 0> enable or disable each weak pull-down. When the port pin is configured as an output, its weak pulldown is automatically cut off.

PORTB pull-down resistor register WPDB (87H).

87H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPDB	WPDB7	WPDB6	WPDB5	WPDB4	WPDB3	WPDB2	WPDB1	WPDB0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0      WPDB<7: 0>: Weak pull-down register bits.  
                          1= Enable pull-down.  
                          0= Pull-down is prohibited.

Note: Weak pulldowns are automatically disabled if the pins are configured as outputs.

### 6.3.4 PORTB Analog Selection Control

The ANSEL1 register is used to configure the input mode of I/O pin to analog mode. Setting the appropriate bit in ANSEL1 to 1 will cause all digital read operations of the corresponding pin to return to 0 and make the analog function of the pin work normally. The state of the ANSEL1 bit has no effect on the digital output function. The pin whose TRIS is cleared and ANSEL1 is set to 1 is still used as a digital output, but the input mode will become an analog mode. This can cause unpredictable results when executing read-modify-write operations on the affected port.

PORTB analog selection register ANSEL1 (111H)

111H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL1	ANS15	ANS14	ANS13	ANS12	ANS11	ANS10	ANS9	ANS8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0      ANS<15: 8>:    Analog selection bits, select the analog or digital functions of pin AN<15: 8>.

1=    Pin set as analog input.

0=    Digital I/O

### 6.3.5 PORTB Level Change Interrupt

All PORTB pins can be individually configured as level change interrupt pins. The control bit IOCB<7: 0> allows or disables the interrupt function of each pin. Disable pin level change interrupt function when power on reset.

For the pin that has allowed level change interrupt, compare the value on the pin with the old value latched when PORTB was read last time. Perform a logical OR operation with the output "mismatch" of the last read operation to set the PORTB level change interrupt flag (RBIF) in the PIR2 register as 1.

This interrupt can wake up the device from sleep mode, and the user can clear the interrupt in the interrupt service program in the following ways:

- Read or write to PORTB. This will end the mismatch state of the pin level.
- Clear the flag bit RBIF.

The mismatch status will continuously set the RBIF flag bit as 1. Reading or writing PORTA will end the mismatch state and allow the RBIF flag to be cleared. The latch will keep the last read value from the undervoltage reset. After reset, if the mismatch still exists, the RBIF flag will continue to be set as 1.

**Note:** If the level of the I/O pin changes during the read operation (beginning of the Q2 cycle), the RBIF interrupt flag bit will not be set as 1. In addition, since reading or writing to a port affects all bits of the port, special care must be taken when using multiple pins in interrupt-on-change mode. When dealing with the level change of one pin, you may not notice the level change on the other pin.

PORTB level change interrupt register IOCB (0DH)

0DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOCB	IOCB7	IOCB6	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	IOCB0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0      IOCB<7: 0>      Control bit of level change interrupt of PORTB  
                          1=      enable level change interrupt  
                          0=      disable level change interrupt

### 6.3.6 PORTB Drive Current Control

All PORTB pins can be selected for different outputs of high-level drive current and low-level drive current, with P A position in the BHEN register 1 (=1) can make the corresponding PORTB pin The output high drive current is selectable, and the current size selection is controlled by the SEGCUR register. Placing a position in the PBLN register at 1 (=1) can make the corresponding PORTB The output low drive current of the pins is optional.

PORTB high-level drive current control register PBHEN (115H).

115H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PBHEN	PBHEN7	PBHEN6	PBHEN5	PBHEN4	PBHEN3	PBHEN2	PBHEN1	PBHEN0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PBHEN<7: 0> The high-level output of the PORTB drives the current control bit.  
1= Allows PORTB high drive current selectable (0mA to 28mA).  
0= Disable high level drive current optional, default is (28mA).

PORTB drives the current control register PBLN (116H) at a low level

116H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PBLN	PBLN7	PBLN6	PBLN5	PBLN4	PBLN3	PBLN2	PBLN1	PBLN0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0 PBLN<7: 0> The low-level output of the PORTB drives the current control bit.  
1= Large drive current (120mA);  
0= Small drive current (60mA).

PORTA/PORTB high-level drive current control register SEGCUR (11CH).

11CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEGCUR	---	---	---	---	---	SEG_ISEL[2: 0]		
R/W	---	---	---	---	---	R/W	R/W	R/W
Reset value	---	---	---	---	---	0	0	0

Bit7~Bit3 Unused  
Bit2~Bit0 SEG\_ISEL<2: 0> The high-level output of PORTA/PORTB drives the current select bit.  
0>  
000= 0mA;  
001= 4mA;  
010= 8mA;  
011= 12mA;  
100= 16mA;  
101= 20mA;  
110= 24mA;  
111= 28mA。

## 6.4 PORTC

### 6.4.1 PORTC Data and Direction

PORTC is an 2bit wide bidirectional port. The corresponding data direction register is TRISC. Set a certain position in TRISC to 1 (=1) to make the corresponding PORTC pin as the input pin. Clearing a bit in TRISC (=0) will make the corresponding PORTC pin as the output pin.

Reading the PORTC register reads the pin status and writing to the register will write the port latch. All write operations are read-modify-write operations. Therefore, writing a port means reading the pin level of the port first, modifying the read value, and then writing the modified value to the port data latch. Even when the PORTC pin is used as an analog input, the TRISC registers still control the orientation of the PORTC pin. When using the PORTC pin as an analog input, the user must ensure that the bits in the TRISC register remain in the Set 1 state. I/O pins configured as analog inputs are always read as 0.

The registers related to the PORTC port are PORTC, TRISC, WPUC, WPDC, ANSEL2, etc.

PORTC data register PORTC (92H)

92H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTC	---	---	---	---	---	---	RC1	RC0
R/W	---	---	---	---	---	---	R/W	R/W
Reset value	---	---	---	---	---	---	X	X

Bit7~Bit2                      Not used  
 Bit1~Bit0      PORTC<1: 0>    PORTC I/O pin bit  
                                  TRISCx=1  
                                  1=    Port pin level >V<sub>IH</sub>;  
                                  0=    Port pin level <V<sub>IL</sub>  
                                  TRISCx=0  
                                  1=    Port output high level;  
                                  0=    Port output low level

PORTC direction register TRISC (93H)

93H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISC	---	---	---	---	---	---	TRISC1	TRISC0
R/W	---	---	---	---	---	---	R/W	R/W
Reset value	---	---	---	---	---	---	1	1

Bit7~Bit2                      Not used  
 Bit1~Bit0      TRISC<1: 0>:    Control bit of PORTC tri-state  
                                  1=    PORTC pin configured as input  
                                  0=    PORTC pin configured as output

example: procedure for PORTC

CLR	PORTC	;clear data register
LDIA	B'00000001'	;set PORTC<0> as input, PORTC<1> as output
LD	TRISC, A	

### 6.4.2 PORTC Pull-up Resistance

Each PORTC pin has an internal weak pull up that can be individually configured. The control bits WPUC<1: 0> enable or disable each weak pull up. When the port pin is configured as output, its weak pull up will be automatically cut off.

PORTC pull up resistance register WPUC (99H)

99H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUC	---	---	---	---	---	---	WPUC1	WPUC0
R/W	---	---	---	---	---	---	R/W	R/W
Reset value	---	---	---	---	---	---	0	0

Bit7~Bit2                      Not used  
 Bit7~Bit0      WPUC<7: 0>: Weak pull up register bit  
                                  1= Enable pull up  
                                  0= Disable pull up

Note: If the pin is configured as output, weak pull up will be automatically disabled

### 6.4.3 PORTC Pull-down Resistor

Each PORTC pin has an internal weak pull-down that can be individually configured. The control bits WPDC < 1: 0> enable or disable each weak pull-down. When the port pin is configured as an output, its weak pulldown is automatically cut off.

PORTC pull-down resistor register WPDC (98H).

98H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPDC	---	---	---	---	---	---	WPDC1	WPDC0
R/W	---	---	---	---	---	---	R/W	R/W
Reset value	---	---	---	---	---	---	0	0

Bit7~Bit2                      Unused  
 Bit1~Bit0      WPDC<1: 0>: Weak pull-down register bits.  
                                  1= Enable pull-down.  
                                  0= Pull-down is prohibited.

Note: Weak pulldowns are automatically disabled if the pins are configured as outputs.



#### 6.4.4 PORTC Analog Control Selection

The ANSEL2 register is used to configure the input mode of I/O pin to analog mode. Setting the appropriate bit in ANSEL2 to 1 will cause all digital read operations of the corresponding pin to return to 0 and make the analog function of the pin work normally. The state of the ANSEL2 bit has no effect on the digital output function. The pin with TRIS cleared and ANSEL2 set to 1 is still used as a digital output, but the input mode will become an analog mode. This can cause unpredictable results when performing read-modify-write operations on the affected port.

PORTC analog selection register ANSEL2 (112H)

112H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL2	---	---	---	---	---	---	ANS17	ANS16
R/W	---	---	---	---	---	---	R/W	R/W
Reset value	---	---	---	---	---	---	0	0

Bit7~Bit2          Not used

Bit1~Bit0      ANS<17: 16>:    Analog selection bit, select the analog or digital function of pin AN<17: 16> respectively.

1=    Pin set as Analog input

0=    Set as Digital I/O

## 6.5 I/O Usage

### 6.5.1 Write I/O Port

The chip's I/O port register, like the general universal register, can be written through data transmission instructions, bit manipulation instructions, etc.

Example: write I/O port program

LD	PORTA, A	;pass value of ACC to PORTA
CLRB	PORTB, 1	;clear PORTB.1
CLR	PORTC	;clear PORTC
SET	PORTA	;set all output port of PORTA as 1
SETB	PORTB, 1	;set PORTB.1as 1

### 6.5.2 Read I/O Port

Example: write I/O port program

LD	A, PORTA	;pass value of PORTA to ACC
SNZB	PORTA, 1	; check whether PORTA, port 1 is 1, if it is 1, skip the next statement
SZB	PORTA, 1	; check if PORTA, 1 port is 0, if 0, skip the next statement

Note: When the user reads the status of an I/O port, if the I/O port is an input port, the data read back by the user will be the state of the external level of the port line. If the I/O port is an output port then The read value will be the data of the internal output register of this port.

## 6.6 Precautions for I/O Port Usage

When operating the I/O port, pay attention to the following aspects:

1. When I/O is converted from output to input, it is necessary to wait for several instruction periods for the I/O port to stabilize.
2. If the internal pull up resistor is used, when the I/O is converted from output to input, the stable time of the internal level is related to the capacitance connected to the I/O port. The user should set the waiting time according to the actual situation. Prevent the I/O port from scanning the level by mistake.
3. When the I/O port is an input port, its input level should be between "VDD+0.7V" and "GND-0.7V". If the input port voltage is not within this range, the method shown in the figure below can be used.

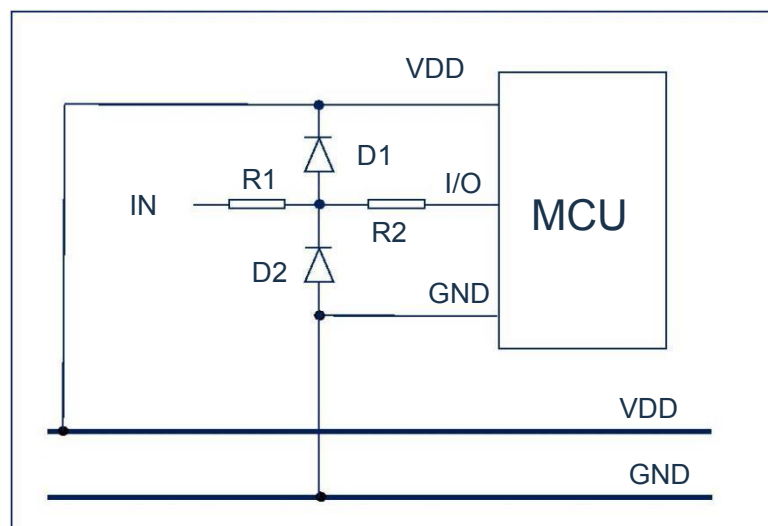


Fig 6-3: The input voltage is not within the specified range

4. If a longer cable is connected to the I/O port, please add a current limiting resistor near the chip I/O to enhance the MCU's anti-EMC capability.

## 7. Interrupt

### 7.1 Interrupt General

The chip has the following interrupt source:

- ◆ TIMER0 overflow interrupt
- ◆ TIMER2 match interrupt
- ◆ PORTA level change interrupt
- ◆ PWM period interrupt
- ◆ USART receive/transmit interrupt
- ◆ Touch key M1 detection completion interrupt
- ◆ TIMER1 overflow interrupt
- ◆ INT interrupt
- ◆ A/D interrupt
- ◆ PORTB level change interrupt
- ◆ Program EEPROM write interrupt
- ◆ Touch key M0 detection completion interrupt

The interrupt control register (INTCON) and the peripherals interrupt request register (PIR1, PIR2) record various interrupt requests in their respective flag bits. The INTCON register also includes various interrupt enable bits and global interrupt enable bits.

The global interrupt enable bit GIE (INTCON<7>) allows all unmasked interrupts when set to 1, and prohibits all interrupts when cleared. Each interrupt can be prohibited through the corresponding enable bits in the INTCON, PIE1, and PIE2 registers. GIE is cleared when reset.

Executing the "return from interrupt" instructions, RETI, will exit the interrupt service program and set the GIE bit to 1, thereby re-allowing unshielded interrupt.

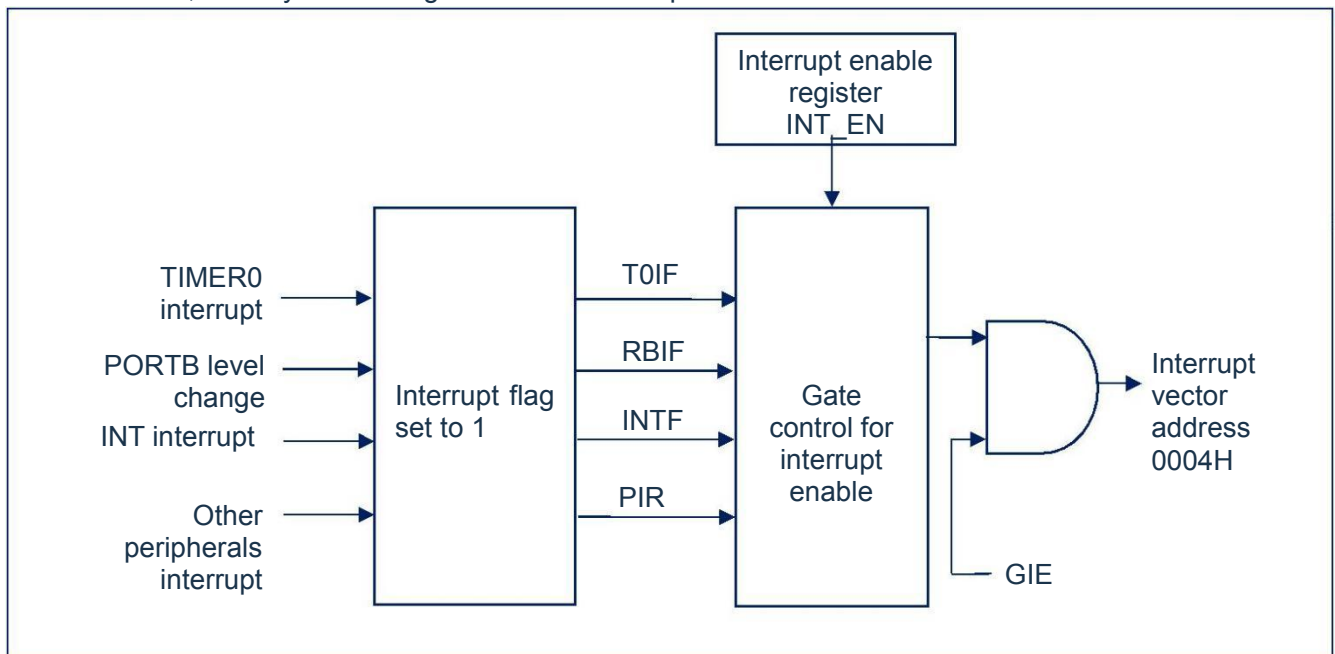


Fig 7-1: interrupt theory

## 7.2 Interrupt Control Register

### 7.2.1 Interrupt Control Register

The interrupt control register INTCON is a readable and writable register, including the enable and flag bits for TMR0 register overflow and external interrupt.

When an interrupt condition occurs, regardless of the state of the corresponding interrupt enable bit or the global enable bit GIE (in the INTCON register), the interrupt flag bit will be set to 1. The user software should ensure that the corresponding interrupt flag bit is cleared before allowing an interrupt.

Interrupt control register INTCON (0BH)

0BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

- Bit7      GIE: Global interrupt enable bit;  
             1= Enable all unshielded interrupt;  
             0= Disable all interrupt
- Bit6      PEIE: Peripherals interrupt enable bit;  
             1= Enable all unshielded peripherals interrupt;  
             0= Disable all peripherals interrupt.
- Bit5      T0IE: TIMER0 overflow interrupt enable bit;  
             1= Enable TIMER0 interrupt;  
             0= Disable TIMER0 interrupt
- Bit4      INTE: INT external interrupt enable bit;  
             1= Enable INT external interrupt;  
             0= Disable INT external interrupt
- Bit3      RBIE: PORTB level change interrupt enable bit (1);  
             1= Enable PORTB level change interrupt;  
             0= Disable PORTB level change interrupt
- Bit2      T0IF: TIMER0 overflow interrupt enable bit (2);  
             1= TMR0 register overflow already (must clear through software);  
             0= TMR0 register not overflow
- Bit1      INTF: INT external interrupt flag bit;  
             1= INT external interrupt happens (must clear through software);  
             0= INT external interrupt not happen
- Bit0      RBIF: PORTB level change interrupt flag bit;  
             1= The level of at least one pin in the PORTB port has changed (must clear through software);  
             0= None of the PORTB universal I/O pin status has changed.

**Note:**

- 1) The IOCB register must also be enabled, and the corresponding port must be set to input state.
- 2) The T0IF bit is set as 1 when TMR0 rolls over to 0. Reset will not change TMR0 and should be initialized before clearing the T0IF bit.

### 7.2.2 Peripherals Interrupt Enable Register

The peripherals interrupt enable register has PIE1 and PIE2. Before allowing any peripherals interrupt, the PEIE bit of the INTCON register must be set to 1.

Peripherals interrupt enable register PIE1 (106H)

106H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE1	---	ADIE	RCIE	TXIE	---	PWMIE	TMR2IE	TMR1IE
R/W	---	R/W	R/W	R/W	---	R/W	R/W	R/W
Reset value	---	0	0	0	---	0	0	0

Bit7	Not used
Bit6	ADIE: A/D converter (ADC) interrupt enable bit; 1= enable ADC interrupt; 0= disable ADC interrupt
Bit5	RCIE: USART receive interrupt enable bit; 1= enable USART receive interrupt; 0= disable USART receive interrupt.
Bit4	TXIE: USART transmit interrupt enable bit; 1= enable USART transmit interrupt; 0= disable USART transmit interrupt.
Bit3	Not used:
Bit2	PWMIE: PWM period interrupt enable bit; 1= enable PWM period interrupt; 0= disable PWM period interrupt.
Bit1	TMR2IE: TIMER2 and PR2 match interrupt enable bit; 1= enable TMR2 and PR2 match interrupt; 0= disable TMR2 and PR2 match interrupt.
Bit0	TMR1IE: TIMER1 overflow interrupt enable bit; 1= enable TIMER1 overflow interrupt; 0= disable TIMER1 overflow interrupt.

### Peripherals interrupt enable register PIE2 (108H)

108H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE2	TKM1IE	TKM0IE	---	EEIE	---	---	RACIE	---
R/W	R/W	R/W	---	R/W	---	---	R/W	---
Reset value	0	0	---	0	---	---	0	---

Bit7	TKM1IE	Touch key module M1 detection completion interrupt enable bit; 1= enable touch key module M1 detection completion interrupt ; 0= disable touch key module M1 detection completion interrupt .
Bit6	TKM0IE	Touch key module M0 detection completion interrupt enable bit; 1= enable touch key module M0 detection completion interrupt ; 0= disable touch key module M0 detection completion interrupt .
Bit5	Not used.	
Bit4	EEIE:	EEPROM write operation interrupt enable bit; 1= enable EEPROM write operation interrupt; 0= disable EEPROM write operation interrupt.
Bit3~Bit2	Not used	
Bit1	RACIE:	PORTA electrical level change interrupt enable bit; 1= enable PORTA electrical level change interrupt ; 0= disable PORTA electrical level change interrupt .
Bit0	Not used	

### 7.2.3 Peripherals Interrupt Request Register

The peripherals interrupt request register is PIR1 and PIR2. When an interrupt condition occurs, regardless of the state of the corresponding interrupt enable bit or the global enable bit GIE, the interrupt flag bit will be set to 1. The user software should ensure that the interrupt is set before allowing an interrupt. The corresponding interrupt flag bit is cleared.

Peripherals interrupt request register PIR1 (0CH)

0CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIR1	---	ADIF	RC0IF	TX0IF	SSPIF	CCP1IF	TMR2IF	TMR1IF
R/W	---	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	0	0	0	0	0	0	0

Bit7	Not used
Bit6	ADIF: A/D converter interrupt flag bit; 1= A/D conversion complete (must clear through software); 0= A/D conversion not complete or not start.
Bit5	RCIF: USART receive interrupt flag bit; 1= USART receive buffer full (clear through reading RCREG); 0= USART receive buffer empty.
Bit4	TXIF: USART transmit interrupt flag bit; 1= USART transmit buffer empty (clear through TXREG); 0= USART transmit buffer full.
Bit3	Not used
Bit2	PWMIF: PWM period interrupt flag bit 1=: PWM period interrupt occurs (must clear by software) 0=: PWM period interrupt not occurred PWM mode: Not used under this mode.
Bit1	TMR2IF: TIMER2 and PR2 match interrupt flag bit. 1= TIMER2 and PR2 match happens (must clear through software); 0= TIMER2 and PR2 not match.
Bit0	TMR1IF: TIMER1 overflow interrupt flag bit. 1= TMR1 register overflow (must clear through software); 0= TMR1 register not overflow.



### Peripherals interrupt request register PIR2 (0DH)

0DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIR2	LVDIF	---	---	EEIF	BCLIF	TX1IF	RX1IF	CCP2IF
R/W	R/W	---	---	R/W	R/W	R/W	R/W	R/W
Reset value	0	---	---	0	0	0	0	0

Bit7            LVDIF: LVD interrupt flag bit;  
                  1= LVD interrupt happens;  
                  0= LVD interrupt not happen.

Bit6~Bit5      Not used.

Bit4            EEIF: Program EEPROM write operation interrupt flag bit;  
                  1= write operation complete (must clear through software);  
                  0= write operation not complete or not start.

Bit3            BCLIF: bus conflict interrupt flag bit;  
                  1= When configured as I<sup>2</sup>C master control mode, bus conflict happens in MSSP;  
                  0= No bus conflict.

Bit2            TX1IF: USART1 transmit interrupt flag bit;  
                  1= USART1 transmit buffer full (clear through writing TXREG1);  
                  0= USART1 transmit buffer empty.

Bit1            RC1IF: USART1 receive interrupt flag bit;  
                  1= USART1 receive buffer full (clear through reading RCREG1);  
                  0= USART1 receive buffer empty.

Bit0            CCP2IF: CCP2 interrupt flag bit.  
                  Capture mode:  
                                  1= capture of TMR1 register happens (must clear through software);  
                                  0= capture of TMR1 register not happen.

                 Compare mode:  
                                  1= compare match of TMR1 register happens (must clear through software);  
                                  0= compare match of TMR1 register not happen.

                 PWM mode: Not used under this mode.

## 7.3 Protection Methods for Interrupt

After an interrupt request occurs and is responded, the program goes to 0004H to execute the interrupt sub-routine. Before responding to the interrupt, the contents of ACC and STATUS must be saved. The chip does not provide dedicated stack saving and unstack recovery instructions, and the user needs to protect ACC and STATUS by himself to avoid possible program operation errors after the interrupt ends.

Example: Stack protection for ACC and STATUS

	ORG	0000H	
	JP	START	;start of user program address
	ORG	0004H	
	JP	INT_SERVICE	;interrupt service program
	ORG	0008H	
START:			
	...		
	...		
INT_SERVICE:			
PUSH:			;entrance for interrupt service program, save ACC and STATUS
	LD	ACC_BAK, A	;save the value of ACC (ACC_BAK needs to be defined)
	SWAPA	STATUS	
	LD	STATUS_BAK, A	;save the value of STATUS (STATUS_BAK needs to be defined)
	...		
	...		
POP:			;exit for interrupt service program, restore ACC and STATUS
	SWAPA	STATUS_BAK	
	LD	STATUS, A	;restore STATUS
	SWAPR	ACC_BAK	;restore ACC
	SWAPA	ACC_BAK	
	RETI		

## 7.4 Interrupt Priority and Multi-interrupt Nesting

The priority of each interrupt of the chip is equal. When an interrupt is in progress, it will not respond to the other interrupt. Only after the "RETI" instructions are executed, the next interrupt can be responded to.

When multiple interrupts occur at the same time, the MCU does not have a preset interrupt priority. First, the priority of each interrupt must be set in advance; second, the interrupt enable bit and the interrupt control bit are used to control whether the system responds to the interrupt. In the program, the interrupt control bit and interrupt request flag must be checked.

## 8. TIMER0

### 8.1 TIMER0 General

TIMER0 is composed of the following functions:

- 8-bit timer/counter register (TMR0);
- 8-bit pre-scaler (shared with watchdog timer);
- Programmable internal or external clock source;
- Programmable external clock edge selection;
- overflow interrupt.

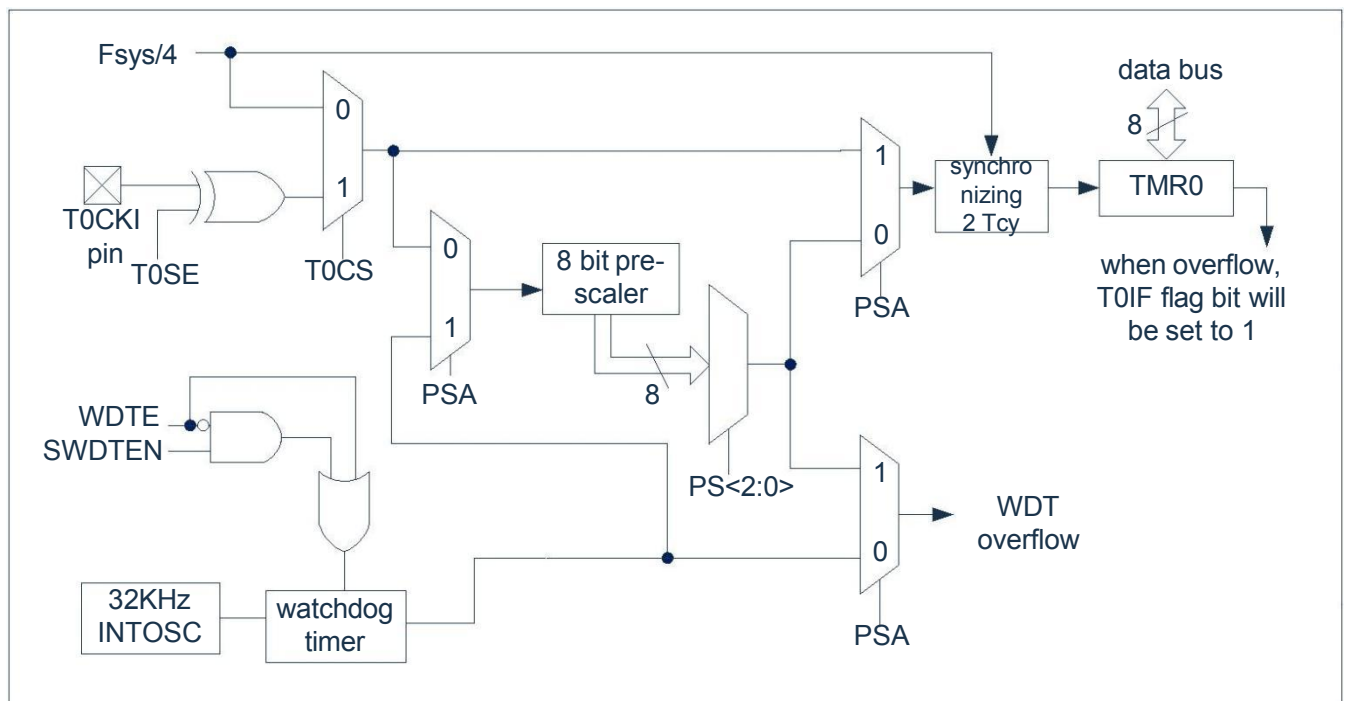


Fig 8-1: TIMER0/WDT mod structure

Note:

1. T0SE, T0CS, PSA, PS<2: 0> are the bits in OPTION\_REG register.
2. SWDTEN is a bit in the WDTCON register.
3. WDTE bit is in CONFIG.

## 8.2 Working Principle for TIMER0

The TIMER0 mod can be used as an 8-bit timer or an 8-bit counter.

### 8.2.1 8-bit Timer Mode

When used as a timer, the TIMER0 mod will be incremented every instruction period (without pre-scaler). The timer mode can be selected by clearing the T0CS bit of the OPTION\_REG register to 0. If a write operation is performed to the TMR0 register, the next two instruction periods will be prohibited from incrementing. The value written to the TMR0 register can be adjusted so that a delay of two instruction periods is included when writing TMR0.

### 8.2.2 8-bit Counter Mode

When used as a counter, the TIMER0 mod will increment on every rising or falling edge of the T0CKI pin. The incrementing edge depends on the T0SE bit of the OPTION\_REG register. The counter mode can be selected by setting the T0CS bit of the OPTION\_REG register to 1.

### 8.2.3 Software Programmable Pre-scaler

TIMER0 and watchdog timer (WDT) share a software programmable pre-scaler, but they cannot be used at the same time. The allocation of the pre-scaler is controlled by the PSA bit of the OPTION\_REG register. To allocate the pre-scaler to TIMER0, the PSA bit must be cleared to 0.

TIMER0mod has 8 selections of prescaler ratio, ranging from 1: 2 to 1: 256. The prescaler ratio can be selected through the PS<2: 0> bits of the OPTION\_REG register. To make TIMER0 mod have a 1: 1 prescaler, the pre-scaler must be assigned to the WDT mod.

The pre-scaler is not readable and writable. When the pre-scaler is assigned to the TIMER0 mod, all instructions written to the TMR0 register will clear the pre-scaler. When the pre-scaler is assigned to the WDT, the CLRWDT instructions will also clear the pre- scaler and WDT.

### 8.2.4 Switch Between TIMER0 and WDT Module Pre-scaler

After assigning the pre-scaler to TIMER0 or WDT, an unintentional device reset may occur when switching the prescaler. To change the pre-scaler from TIMER0 to WDT mod, the following instructions must be executed sequence.

Modify pre-scaler (TMR0-WDT)

CLRWDT		;clear WDT
LDIA	B'xxx1xxx'	;set new pre-scaler
LD	OPTION_REG, A	
CLRWDT		;clear WDT

To change the pre-scaler from WDT to TIMER0 mod, the following sequence of instructions must be executed.

Modify pre-scaler (WDT-TMR0)

CLRWDT		;clear WDT
LDIA	B'xxx0xxx'	;set new pre-scaler
LD	OPTION_REG, A	

### 8.2.5 TIMER0 Interrupt

When the TMR0 register overflows from FFh to 00H, a TIMER0 interrupt is generated. Every time the TMR0 register overflows, regardless of whether TIMER0 interrupt is allowed, the T0IF interrupt flag bit of the INTCON register will be set to 1. The T0IF bit must be cleared in software. TIMER0 interrupt enable bit is the T0IE bit of the INTCON register.

Note: Because the timer is turned off in sleep mode, the TIMER0 interrupt cannot wake up the processor.

## 8.3 TIMER0 Related Register

There are two registers related to TIMER0, 8-bit timer/counter (TMR0), and 8-bit programmable control register (OPTION\_REG).

TMR0 is an 8-bit readable and writable timer/counter, OPTION\_REG is an 8-bit write-only register, the user can change the value of OPTION\_REG to change the working mode of TIMER0, etc. Please refer to the application of 0 prescaler register (OPTION\_REG).

8-bit timer/counter TMR0 (01H)

01H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR0								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	X	X	X	X	X	X	X	X

OPTION\_REG register (81H)

81H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION_REG	---	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
Read/write	---	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	---	1	1	1	1	0	1	1

Bit7	Not used:							
Bit6	INTEDG: Interrupt edge selection bit.							
	1= The rising edge of the INT pin triggers interrupt.							
	0= The falling edge of the INT pin triggers interrupt.							
Bit5	T0CS: TMR0 clock source selection bit.							
	1= Transition edge of T0CKI pin.							
	0= Internal instruction period clock ( $F_{sys}/4$ ).							
Bit4	T0SE: TIMER0 clock source edge selection bit.							
	1= Increment when the T0CKI pin signal transitions from high to low.							
	0= Increment when the T0CKI pin signal transitions from low to high.							
Bit3	PSA: pre-scaler allocation bit.							
	1= pre-scaler allocated to WDT.							
	0= pre-scaler allocated toTIMER0 mod.							
Bit2~Bit0	PS2~PS0: Pre-allocated parameter configuration bits.							
	PS2	PS1	PS0	TMR0 Frequency division ratio		WDT Frequency division ratio		
	0	0	0	1: 2		1: 1		
	0	0	1	1: 4		1: 2		
	0	1	0	1: 8		1: 4		
	0	1	1	1: 16		1: 8		
	1	0	0	1: 32		1: 16		
	1	0	1	1: 64		1: 32		
	1	1	0	1: 128		1: 64		
	1	1	1	1: 256		1: 128		

## 9. TIMER1

### 9.1 TIMER1 Overview

TIMER1 mod is a 16-bit timer/counter with the following characteristics:

- ◆ 16-bit timer/counter register (TMR1H: TMR1L)
- ◆ 3-bit pre-scaler
- ◆ Synchronous or asynchronous operation
- ◆ Wake up when overflow (external clock asynchronous mode only)
- ◆ Programmable internal or external clock source
- ◆ Through Comparator or T1G pinmate control TIMER1 (enable counting)
- ◆ overflow interrupt

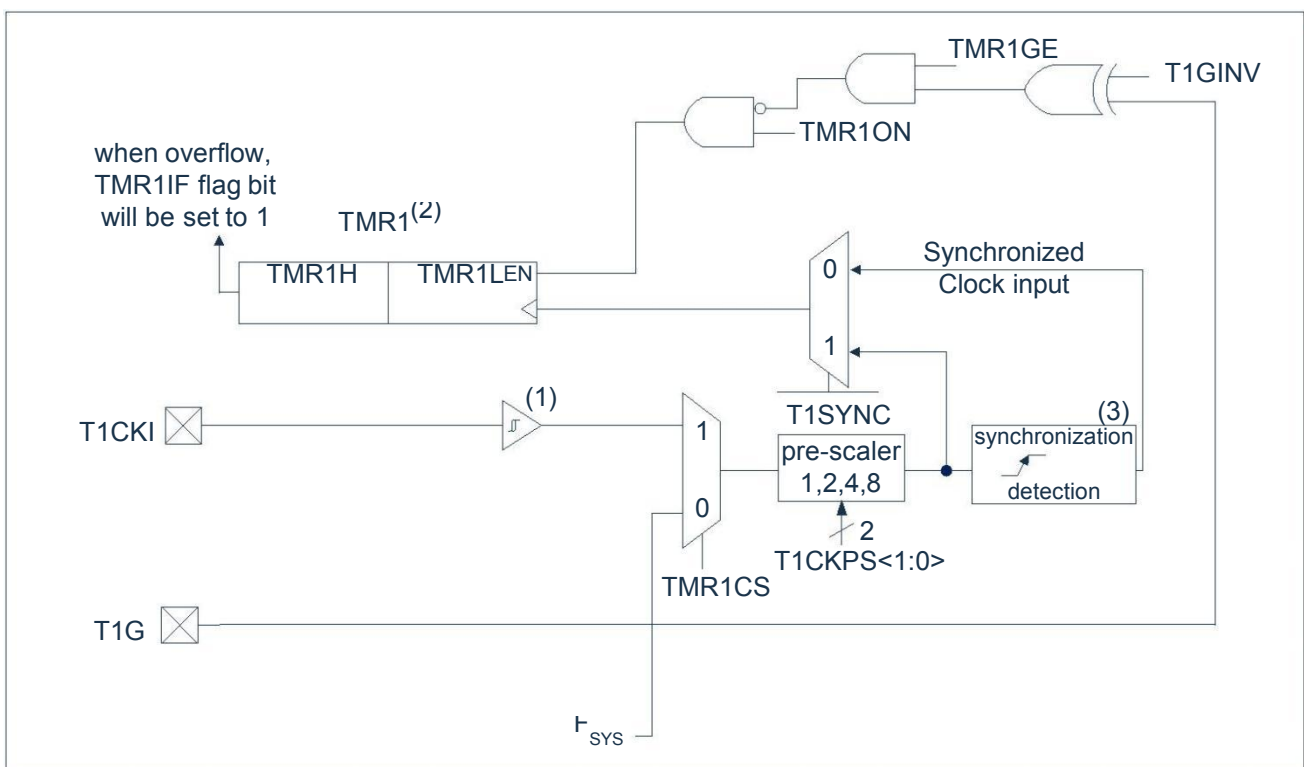


Fig 9-1: TIMER1 structure

Note:

- 1) The ST buffer is in high speed mode when using T1CKI.
- 2) The Timer1 register increments on the rising edge.
- 3) Do not perform synchronous during sleep.

### 9.2 Operating Principle for TIMER1

TIMER1 mod is a 16-bit incremental counter accessed through a pair of register TMR1H: TMR1L. Writing to TMR1H or TMR1L can directly update the counter.

When used with internal clock source, this mod can be used as a counter. When used with external clock source, this mod can be used as a timer or counter.

## 9.3 Clock Source Selection

The TMR1CS bit of the T1CON register is used to select the clock source. When TMR1CS=0, the frequency of the clock source is  $F_{SYS}$ . When TMR1CS=1, the clock source is provided by external.

clock source	TMR1CS
$F_{SYS}$	0
T1CKIpin	1

### 9.3.1 Internal Clock Source

After selecting the internal clock source, the TMR1H: TMR1L register will increase in frequency with a multiple of  $F_{SYS}$ . The specific multiple is determined by the TIMER1 pre-scaler.

### 9.3.2 External Clock Source

After selecting the external clock source, TIMER1mod can be used as a timer or counter.

When counting, TIMER1 is incremented on the rising edge of external clock input T1CKI. In addition, the clock in counter mode can be synchronous or asynchronous with the microcontroller system clock.

In counter mode, when one or more of the following conditions occur, a falling edge must be passed before the counter can count up for the first time on the subsequent rising edge (see Figure 9-2):

- Enable TIMER1.
- A write operation was performed on TMR1H or TMR1L.
- When TIMER1 is disabled, T1CKI is high; when TIMER1 is re-enabled, T1CKI is low.

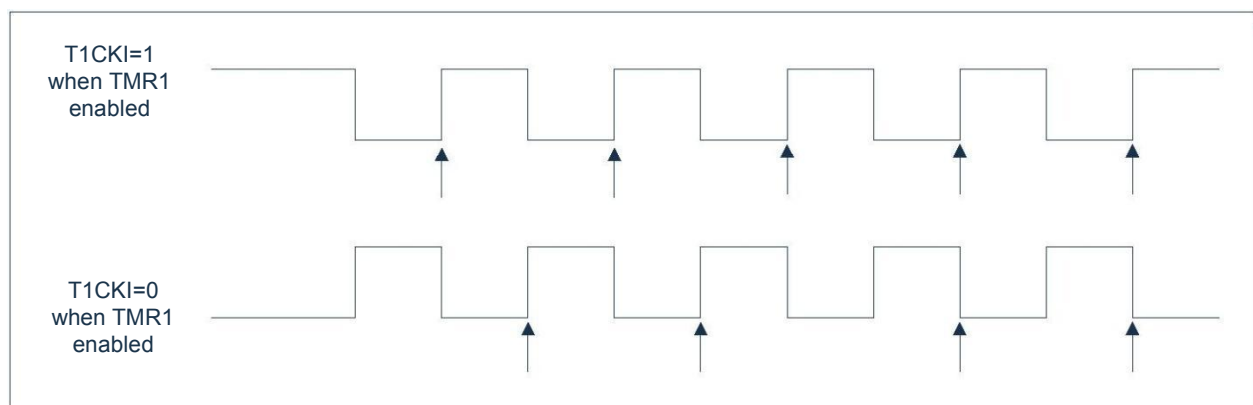


Fig 9-2: incremental edge of TIMER1

**Note:**

- 1) The arrow indicates that the counter is incrementing.
- 2) In the counter mode, a falling edge must be passed before the counter can perform the first increment technique on the subsequent rising edge.



## 9.4 TIMER1 Pre-scaler

TIMER1 has four selections of prescaler ratios, allowing the input clock to be divided by 1, 2, 4 or 8. The T1CKPS bit of the T1CON register controls the prescaler counter. The prescaler counter cannot be directly read or written; but, The prescaler counter can be cleared by writing to TMR1H or TMR1L.

## 9.5 TIMER1 Operating Principle Under Asynchronous Counter Mode

If the control bit T1SYNC in the T1CON register is set to 1, the external clock input will not be synchronous. The timer continues to count up asynchronously with the internal phase clock. The timer will continue to run in the sleep state, and will generate an interrupt during overflow, thereby waking up Processor. However, you should be especially careful when using software to read/write timers (see Section 9.5.1 "Read and Write to TIMER1 in Asynchronous Counter Mode").

**Note:**

- 1) When switching from synchronous operation to asynchronous operation, an increment may be missed.
- 2) When switching from asynchronous operation to synchronous operation, a false increment may occur.

### 9.5.1 Read and Write Operations to TIMER1 in Asynchronous Counter Mode

When the timer uses an external asynchronous clock to work, the read operation of TMR1H or TMR1L will ensure that it is valid (the hardware is responsible). But users should keep in mind that reading two 8-bit values to read a 16-bit timer has its own problems. This is because the timer may overflow between two read operations.

For write operations, it is recommended that the user stop the timer before writing the required value. When the register is counting up, writing data to the timer register may cause write contention. This will cause unpredictability in the register pair TMR1H: TMR1L Value.

## 9.6 TIMER1 Gate Control

Software can configure the TIMER1 gate control signal source as T1G pin, which allows the device to directly use T1G to time external events.

Note: The TMR1GE bit of the T1CON register must be set to 1 to use the gate control signal of TIMER1.

You can use the T1GINV bit of the T1CON register to set the polarity of the TIMER1 gate control signal. The gate control signal can come from T1Gpin. This bit can configure TIMER1 to time the high-level time or low-level time between events.

## 9.7 TIMER1 Interrupt

After a pair of TIMER1 registers (TMR1H: TMR1L) count up to FFFFH, the overflow returns to 0000H. When TIMER1 overflows, the TIMER1 interrupt flag bit of the PIR1 register is set to 1. To allow the overflow interrupt, the user should set the following bit to 1:

- ◆ TIMER1 interrupt enable bit in PIE1 register;
- ◆ PEIE bit in INTCON register;
- ◆ GIE bit in INTCON register.

Clear the TMR1IF bit in the interrupt service program to clear the interrupt.

Note: Before allowing the interrupt again, the register pair TMR1H: TMR1L and the TMR1IF bit should be cleared.

## 9.8 TIMER1 Working Principle During Sleep

TIMER1 can work in sleep mode only when it is set to asynchronous counter mode. In this mode, the external crystal or clock source can be used to make the counter count up. The timer can wake up the device through the following settings:

- ◆ The TMR1ON bit in the T1CON register must be set to 1;
- ◆ The TMR1IE bit in the PIE1 register must be set to 1;
- ◆ The PEIE bit in the INTCON register must be set to 1.

The device will be woken up at overflow and execute the next instruction. If the GIE bit in the INTCON register is 1, the device will call the interrupt service routine (0004h).

## 9.9 TIMER1 Control Register

TIMER1 control register T1CON (10EH)

10EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	---	T1SYNC	TMR1CS	TMR1ON
R/W	R/W	R/W	R/W	R/W	---	R/W	R/W	R/W
Reset value	0	0	0	0	---	0	0	0

Bit7	T1GINV:	TIMER1 gate control signal polarity bit; 1= TIMER1 gate control signal is active high (TIMER1 counts when the gate control signal is high level); 0= The TIMER1 gate control signal is active low (TIMER1 counts when the gate control signal is low).
Bit6	TMR1GE:	TIMER1 gate control enable bit. If TMR1ON=0, ignore this bit. If TMR1ON=1: 1= TIMER1 counting is controlled by TIMER1gate control function; 0=TIMER1always counts.
Bit5~Bit4	T1CKPS<1: 0>:	TIMER1 input clock frequency ratio selection bit; 11= 1: 8; 10= 1: 4; 01= 1: 2; 00= 1: 1.
Bit3	Not used:	
Bit2	T1SYNC:	TIMER1 external clock input synchronous control bit. TMR1CS=1: 1= not synchronous with external clock input; 0= synchronous with external clock input. TMR1CS=0: ignore this bit, TIMER1 uses internal clock.
Bit1	TMR1CS:	TIMER1 clock source selection bit; 1= external clock source T1CKI. 0= Internal clock source $F_{sys}$ .
Bit0	TMR1ON:	TIMER1enable bit; 1= Enable TIMER1; 0= Disable TIMER1.

## 10. TIMER2

### 10.1 TIMER2 General

TIMER2 mod is an 8-bit timer/counter with the following characteristics:

- ◆ 8-bit timer register (TMR2);
- ◆ 8-bit period register (PR2);
- ◆ Interrupt when TMR2 matches PR2;
- ◆ Software programmable prescaler ratio (1: 1, 1: 4 and 1: 16);
- ◆ Software programmable post-scaler ratio (1: 1 to 1: 16).

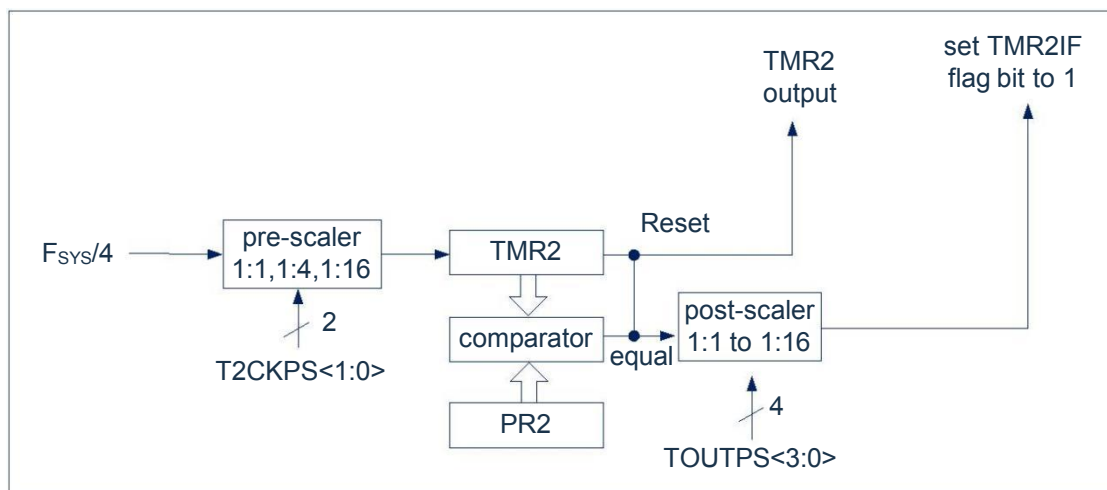


Fig 10-1: TIMER2 structure

## 10.2 Operating Principle of TIMER2

The input clock of the TIMER2 mod is the system instruction clock ( $F_{\text{SYS}}/4$ ). The clock is input to the TIMER2 pre-scaler. There are several division ratios to choose from: 1: 1, 1: 4 or 1: 16. pre-scaler The output is then used to increment TMR2register.

Continue to compare the values of TMR2 and PR2 to determine when they match. TMR2 will increase from 00H until it matches the value in PR2. When a match occurs, the following two events will occur:

- TMR2 is reset to 00h in the next increment period;
- TIMER2 post-scaler increments.

The matching output of the TIMER2 and PR2 comparator is then input to the post-scaler of TIMER2. The post-scaler has a prescaler ratio of 1: 1 to 1: 16 to choose from. The output of the TIMER2 post-scaler is used to make PIR1 The TMR2IF interrupt flag bit of the register is set to 1.

Both TMR2 and PR2 registers can be read and written. At any reset, TMR2 register is set to 00h and PR2 register is set to FFh.

Enable TIMER2 by setting the TMR2ON bit of the T2CON register; disable TIMER2 by clearing the TMR2ON bit.

The TIMER2 pre-scaler is controlled by the T2CKPS bit of the T2CON register; the TIMER2 postscaler is controlled by the TOUTPS bit of the T2CON register.

The pre-scaler and postscaler counters are cleared under the following conditions:

- When TMR2ON=0
- Any device reset occurs (power-on reset, watchdog timer reset, or undervoltage reset).

Note: Writing T2CON will not clear TMR2. When TMR2ON=0, the TMR2 register will be cleared; must set TMR2ON to 1 in order to be able to perform write operation to TMR2.

## 10.3 TIMER2 Related Register

There are two registers related to TIMER2, namely data memory TMR2 and control register T2CON.

TIMER2 data register TMR2 (11H)

11H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR2								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	X	X	X	X	X	X	X	X

TIMER2 control register T2CON (12H)

12H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T2CON	----	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
Read write	----	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	----	0	0	0	0	0	0	0

Bit7	Not used
Bit6~Bit3	TOUTPS<3: 0>: TIMER2 output frequency division ratio selection bit. 0000= 1: 1; 0001= 1: 2; 0010= 1: 3; 0011= 1: 4; 0100= 1: 5; 0101= 1: 6; 0110= 1: 7; 0111= 1: 8; 1000= 1: 9; 1001= 1: 10; 1010= 1: 11; 1011= 1: 12; 1100= 1: 13; 1101= 1: 14; 1110= 1: 15; 1111= 1: 16.
Bit2	TMR2ON: TIMER2 enable bit; 1= Enable TIMER2; 0= Disable TIMER2.
Bit1~Bit0	T2CKPS<1: 0>: TIMER2 clock frequency division ratio selection bit; 00= 1; 01= 4; 1x= 16.

## 11. Analog to Digital Conversion (ADC)

### 11.1 ADC General

The analog-to-digital converter (ADC) can convert the analog input signal into a 12-bit binary number that represents the signal. The analog input channels used by the device share a sample and hold circuit. The output of the sample and hold circuit is connected to the input of the analog to digital converter. The analog-to-digital converter uses the successive approximation method to generate a 12-bit binary result, and save the result in the ADC result register (ADRESL and ADRESH). ADC generates an interrupt after conversion completes.

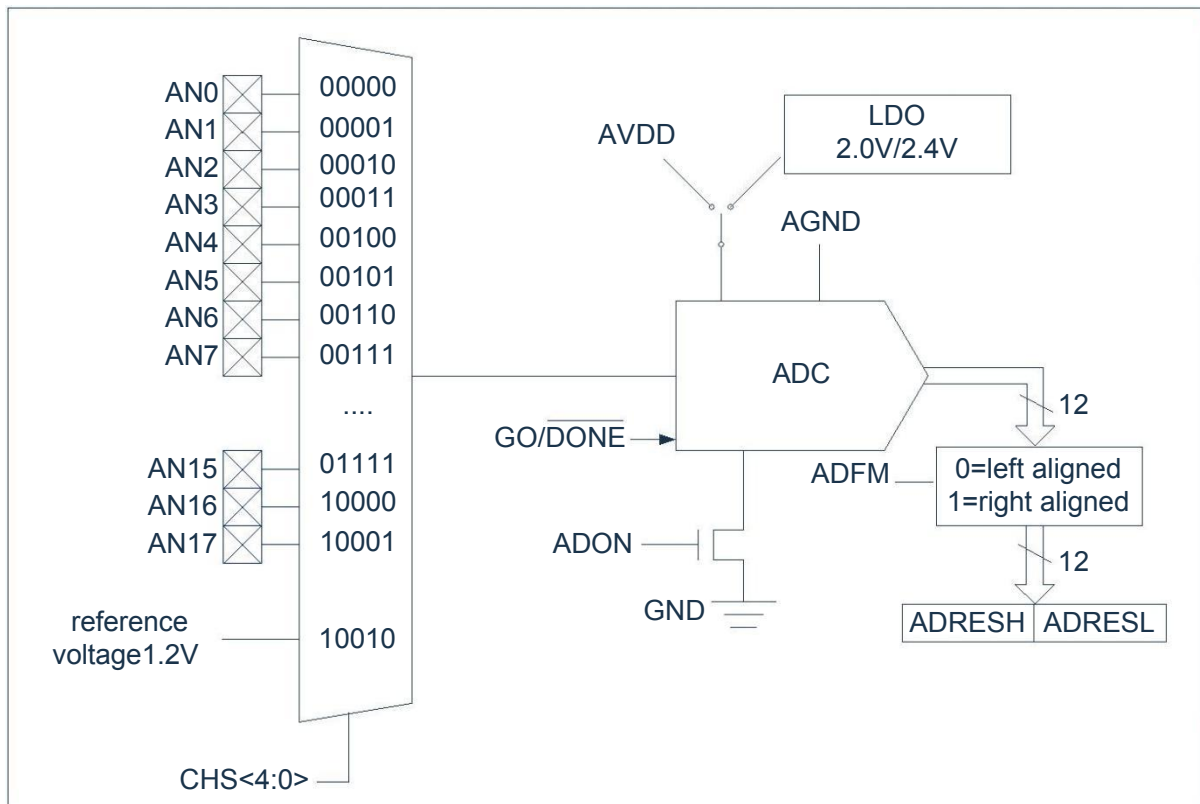


Fig 11-1: ADC structure

## 11.2 ADC Configuration

When configuring and using ADC, the following factors must be considered:

- ◆ Port configuration;
- ◆ Channel selection;
- ◆ ADC reference voltage;
- ◆ ADC conversion clock source;
- ◆ Interrupt control;
- ◆ The storage format of the result.

### 11.2.1 Port Configuration

ADC can convert both analog signal and digital signal. When converting analog signal, the I/O pin should be configured as analog input pin by setting the corresponding TRIS bit to 1. For more information, please refer to the corresponding port chapter.

Note: Applying analog voltage to pins defined as digital inputs may cause overcurrent in the input buffer.

### 11.2.2 Channel Selection

The CHS bit of the ADCON0/ADCON1 register determines which channel is connected to the sample and hold circuit.

If the channel is changed, a certain delay will be required before the next conversion starts. For more information, please refer to the "ADC working principle" chapter.

### 11.2.3 ADC Internal Reference Voltage

The chip has a built-in 1.2V reference voltage, and when this reference voltage needs to be detected, the CHS [4: 0] bit needs to be set to 10010.

### 11.2.4 ADC Reference Voltage

The ADC reference voltage is always provided by the chip's  $V_{DD}$  and GND.



## 11.2.5 Converter Clock

The ADCS bit of the ADCON0 register can be set by software to select the clock source for conversion. There are 4 possible clock frequencies to choose from:

- ◆  $F_{SYS}/8$                       ◆  $F_{SYS}/32$
- ◆  $F_{SYS}/16$                      ◆  $F_{SYS}/128$

The time to complete one-bit conversion is defined as  $T_{AD}$ . A complete 12-bit conversion requires 49  $T_{AD}$  periods.

Must comply with the corresponding  $T_{AD}$  specification to get the correct conversion result. The following table is an example of correct selection of ADC clock.

Note: Unless FRC is used, any change in the system clock frequency will change the ADC clock frequency, which will negatively affect the ADC conversion results.

For different reference voltages and different VDDs, you need to refer to the following table to set a reasonable frequency division.

Reference voltage	Working voltage (V)	Fastest division setting		Conversion time (us)
		$F_{SYS} = 16\text{MHz}$	$F_{SYS} = 8\text{MHz}$	
$V_{DD}$	3.0~5.5	$F_{SYS}/16$	$F_{SYS}/8$	16
$V_{DD}$	2.7~3.0	$F_{SYS}/32$	$F_{SYS}/16$	32
Internal reference 2.4V	4.0~5.5	$F_{SYS}/32$	$F_{SYS}/16$	32
Internal reference 2.4V	2.7~4.0	$F_{SYS}/128$	$F_{SYS}/128$	128/256
Internal reference 2.0V	2.7~5.5	$F_{SYS}/128$	$F_{SYS}/128$	128/256

## 11.2.6 ADC Interrupt

ADC mod allows an interrupt to be generated after the completion of the analog-to-digital conversion. The ADC interrupt flag bit is the ADIF bit in PIR1register. The ADC interrupt enable bit is the ADIE bit in PIE1register. The ADIF bit must be cleared by software. The ADIF bit after each conversion is completed Will be set to 1, regardless of whether ADC interrupt is allowed.

## 11.2.7 Output Formatting

The result of 12-bit A/D conversion can be in two formats: left-justified or right-justified. The output format is controlled by the ADFM bit in ADCON1register.

When ADFM=0, the AD conversion result is left aligned and the AD conversion result is 12Bit; when ADFM=1, the AD conversion result is right aligned, and the AD conversion result is 10 Bit.

## 11.3 ADC Working Principle

### 11.3.1 Start Conversion

To enable ADC mod, you must set the ADON bit of the ADCON0 register to 1, and set the GO/ ("DONE") bit of the ADCON0 register to 1 to start analog-to-digital conversion.

Note: It is not possible to set GO/DONE position to 1 with the same instructions that open A/D module.

### 11.3.2 Complete Conversion

When the conversion is complete, the ADC mod will:

- Clear the GO/DONE bit;
- Set ADIF flag bit to 1;
- Update the ADRESH: ADRESL register with the new conversion result.

### 11.3.3 Stop Conversion

If you must terminate the conversion before conversion is completed, you can use software to clear the GO/DONE bit. The ADRESH: ADRESL register will not be updated with the uncompleted analog-to-digital conversion result. Therefore, the ADRESH: ADRESL register will remain on The value obtained by the second conversion. In addition, after the A/D conversion is terminated, a delay of  $2 T_{AD}$  must be passed before the next acquisition can be started. After the delay, the input signal of the selected channel will automatically start to be collected.

Note: Device reset will force all registers to enter the reset state. Therefore, reset will close the ADC mod and terminate any pending conversions.

### 11.3.4 Working Principle of ADC in Sleep Mode

ADC module can not work in sleep mode.

### 11.3.5 A/D Conversion Procedure

The following steps give an example of using ADC for analog-to-digital conversion:

1. port configuration:
  - Configure pin as input pin (see TRIS register).
2. configuration ADC mod:
  - Select ADC reference voltage, if it is switched from  $V_{DD}$  to internal 2.0V/2.4V voltage, it will start detecting AD after waiting for at least 200us.
  - Select ADC conversion clock;
  - Select ADC input channel;
  - Choose the format of the result;
  - Start the ADC mod.
3. configuration ADC interrupt (optional):
  - Clear ADC interrupt flag bit;
  - Allow ADC interrupt;
  - Allow peripherals interrupt;
  - Allow global interrupt.
4. Wait for the required acquisition time.
5. Set GO/DONE to 1 to start conversion.
6. Wait for the ADC conversion to end by one of the following methods:
  - Query GO/ ("DONE") bit;
  - Wait for ADC interrupt (allow interrupt).
7. Read ADC results.
8. Clear the ADC interrupt flag bit (if interrupt is allowed, this operation is required).

**Note:** If the user tries to resume sequential code execution after waking the device from sleep mode, the global interrupt must be disabled.

example: AD conversion

LDIA	B'10000000'	
LD	ADCON1, A	
SETB	TRISA, 0	;set PORTA.0 as input
LDIA	B'11000001'	
LD	ADCON0, A	
CALL	DELAY	;delay
SETB	ADCON0, GO	
SZB	ADCON0, GO	;wait ADC to complete
JP	\$_1	
LD	A, ADRESH	;save the highest bit of ADC
LD	RESULTH, A	
LD	A, ADRESL	; save the lowest bit of ADC
LD	RESULTL, A	

## 11.4 ADC Related Register

There are mainly 4 RAMs related to AD conversion, namely control register ADCON0 and ADCON1, data register ADRESH and ADRESL.

AD control register ADCON0 (9DH)

9DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit6      ADCS<1: 0>: A/D conversion clock selection bit.

00=  $F_{SYS}/8$

01=  $F_{SYS}/16$

10=  $F_{SYS}/32$

11=  $F_{SYS}/128$

Bit5~Bit2      CHS<3: 0>: The analog channel selection bit.

CHS<4: 0>: CHS4 at ADCON1 register

00000= AN0

00001= AN1

00010= AN2

00011= AN3

... ..

10000= AN16

10001= AN17

10010= fixed reference voltage (1.2V fixed reference voltage)

Others Reserved

Bit1      GO/DONE: A/D conversion status bit.

1= A/D conversion is in progress. Set this bit to 1 to start A/D conversion. When A/D conversion is completed, this bit is automatically cleared by hardware.

0= A/D conversion complete or not in progress.

Bit0      ADON: ADC enable bit.

1= Enable ADC;

0= Disable ADC

## AD data register high bit ADCON1 (9CH)

9CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON1	ADFM	CHS4	----	----	----	LDO_EN	----	LDOSSEL
Read/write	R/W	R/W	----	----	----	R/W	----	R/W
Reset value	0	0	----	----	----	0	----	0

Bit7            ADFM:    A/D conversion result format selection bit

1=    Right alignment

0=    left alignment

Bit6            CHS4:    Channel selection bit

Bit5~Bit3       Not used

Bit2            LDO\_EN    ADC internal reference LDO enable bit

1=    enable, ADC的VREF输入为LDO

0=    disable, ADC的VREF输入为VDD

Bit1            Not used

Bit0            LDOSSEL    AD reference voltage selection bit

1=    2.0V

0=    2.4V

## AD data register high bit ADRESH (9EH), ADFM=0

9EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESH	ADRES11	ADRES10	ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4
read/write	R	R	R	R	R	R	R	R
Reset value	X	X	X	X	X	X	X	X

Bit7~Bit0        ADRES<11: 4>:    ADC result register bit.

The higher 8 bits of the 12-bit conversion result.

## AD data register lower bit ADRESL (9FH), ADFM=0

9FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESL	ADRES3	ADRES2	ADRES1	ADRES0	----	----	----	----
read/write	R	R	R	R	----	----	----	----
Reset value	X	X	X	X	----	----	----	----

Bit7~Bit4        ADRES<3: 0>:    ADC result register bit.

The lower 4 bits of the 12-bit conversion result.

Bit3~Bit0        Not used

[illegible]

The higher 2 bits of the 12-bit conversion result.

9FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESL	ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4	ADRES3	ADRES2
read/write	R	R	R	R	R	R	R	R
Reset value	X	X	X	X	X	X	X	X

The 2-9 bits of the 12-bit conversion result.

Note: In the case of ADFM=1, the AD conversion result only saves the upper 10 bits of the 12-bit result, where ADRESH saves the upper 2 bits, and ADRESL saves the 2nd to 9th digits.

## 12. Touch Button

### 12.1 Touch Button Module Overview

The touch detection mod is an integrated circuit designed to realize a human touch interface. It can replace mechanical touch buttons to achieve a waterproof and dustproof, sealed and isolated, sturdy and beautiful operation interface.

technical parameter:

- ◆ 1-8 buttons selectable.
- ◆ No need for external touch capacitance

### 12.2 Precautions of Using Touch Button Module

- ◆ The ground wire of the detection part of the touch button should be separately connected to an independent ground, and another point is connected to the common ground of the whole machine.
- ◆ Avoid high-voltage, high-current, high-frequency operation of the motherboard and the touch circuit board. If it is unavoidable, try to stay away from the area of the high-voltage current or add shielding on the motherboard.
- ◆ The connection between the sensor pad and the touch chip should be as short and thin as possible. If the PCB process allows it, try to use a line width of 0.1mm.
- ◆ The connection between the sensor panel and the touch chip should not cross the signal line with strong interference and high frequency.
- ◆ Do not use other signal lines around 0.5mm from the sensor panel to the touch chip.

This series of microcontrollers provides multiple touch button functions. The button function is built into the microcontroller, does not require external components, and is easily operated through internal registers.

## 13. PWM Module

The chip contains a 10-bit PWM module that can be configured as four common-period, independent duty cycle outputs + 1 independent output, or 2 complementary outputs + 1 independent output.

The PWM output port can be selected by PWMIO\_SEL [1: 0] bit, and there are 3 groups to choose from, namely PWMA0~A4, PWMB0~B4, P WMC0~C4. PWM0/PWM1, PWM2/PWM3 can be configured with complementary forward and reverse outputs.

### 13.1 Pin Configuration

The corresponding PWM pin should be configured as an output by corresponding TRIS control position 0.

### 13.2 Description of The Relevant Registers

PWM control register PWMCON0 (13H).

1PM	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMCON0	CLKDIV[2: 0]			PWM4EN	PWM3EN	PWM2EN	PWM1EN	PWM0EN
Read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit5 CLKDIV[2: 0]: PWM clock divider.

111=  $F_{osc}/128$

110=  $F_{osc}/64$

101=  $F_{osc}/32$

100=  $F_{osc}/16$

011=  $F_{osc}/8$

010=  $F_{osc}/4$

001=  $F_{osc}/2$

000=  $F_{osc}/1$

Bit4~Bit0 PWMxEN: PWMx enable bit.

1= Enable PWMx.

0= Disable PWMx.



PWM control register PWMCON1 (14H)

14H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMCON1	PWMIO_SEL[1: 0]		PWM2DTEN	PWM0DTEN	---	---	DT_DIV[1: 0]	
Read/Write	R/W	R/W	R/W	R/W	---	---	R/W	R/W
Reset Value	0	0	0	0	---	---	0	0

Bit7~6 PWMIO\_SEL[1: 0]: PWM IO selection.

1X= PWM is assigned in **Group C**, PWM0-RB0, PWM1-RB1, PWM2-RB2, PWM3-RB3, PWM4-RA5

01= PWM is assigned in **Group B**, PWM0-RB0, PWM1-RB1, PWM2-RB2, PWM3-RB3, PWM4-RB4

00= PWM is assigned in **Group A**, PWM0-RA3, PWM1-RA4, PWM2-RA5, PWM3-RA6, PWM4-RA7

Bit5 PWM2DTEN: PWM2 dead-zone enable bit.

1= Enabling the PWM2 dead-zone function, PWM2 and PWM3 form a pair of complementary outputs.

0= Disables the PWM2 dead-zone feature.

Bit4 PWM0DTEN: PWM0 dead-zone enable bit.

1= Enabling the PWM0 dead-zone function, PWM0 and PWM1 form a pair of complementary outputs.

0= Disables the PWM0 dead-zone feature.

Bit3~Bit2 Not used

Bit1~Bit0 DT\_DIV[1: 0] Dead-zone clock source divider.

11=  $F_{osc}/8$

10=  $F_{osc}/4$

01=  $F_{osc}/2$

00=  $F_{osc}/1$

PWM control register PWMCON2 (1DH)

1DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMCON2	---	---	---	PWM4DIR	PWM3DIR	PWM2DIR	PWM1DIR	PWM0DIR
R/W	---	---	---	R/W	R/W	R/W	R/W	R/W
Reset Value	---	---	---	0	0	0	0	0

Bit7~Bit5 Not used

Bit4~Bit0 The PWMxDIR PWM output inverse control bit.

1= PWMx takes the inverse output.

0= PWMx normal output.

PWM0~PWM3 cycle lower bits register PWMTL (15H)

15H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMTL	PWMT[7: 0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset Value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMT[7: 0]: The PWM0~PWM3 lower 8 bits of the cycle.

#### PWM4 period lower bits register PWM4TL(1EH)

1EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM4TL	PWM4T[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset Value	0	0	0	0	0	0	0	0

Bit7~Bit0      PWM4T[7:0]:    PWM4 lower 8 bits of the cycle

#### Higher bit of period register PWMTH (16H)

16H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMTH	---	---	PWMD4[9:8]		PWM4T[9:8]		PWMT[9:8]	
Read/Write	---	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset Value	---	---	0	0	0	0	0	0

Bit7~Bit6      Not used

Bit5~Bit4      PWMD4[9:8]:    PWM4 higher 2 bits of duty cycle.

Bit3~Bit2      PWM4T[9:8]:    PWM4 higher 2 bits of period.

Bit1~Bit0      PWMT[9:8]:    PWM0~PWM3 higher 2 bits of period.

**Note:** Writing to PWMD4 [9:8] does not take effect immediately, and a write to PWMD4L operation is required to take effect.

#### PWM0 duty cycle lower bit register PWMD0L (17H)

17H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD0L	PWMD0[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset Value	0	0	0	0	0	0	0	0

Bit7~Bit0      PWMD0[7:0]:    PWM0 lower 8 bits of duty cycle .

#### PWM1 duty cycle lower bit register PWMD1L (18H)

18H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD1L	PWMD1[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset Value	0	0	0	0	0	0	0	0

Bit7~Bit0      PWMD1[7:0]:    PWM1 lower 8 bits of duty cycle .

PWM2 duty cycle lower bit register PWMD2L (19H).

19H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD2L	PWMD2[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset Value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD2[7:0]: PWM2 lower 8 bits of duty cycle。

PWM3 duty cycle lower bit register PWMD3L (1AH)

1AH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD3L	PWMD3[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset Value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD3[7:0]: PWM3 lower 8 bits of duty cycle。

PWM4 duty cycle lower bit register PWMD4L (1BH)

1BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD4L	PWMD4[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset Value	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD4[7:0]: PWM4 lower 8 bits of duty cycle。

PWM0 and PWM1 duty cycle higher bits register PWMD01H (1CH)

1CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD01H	---	---	PWMD1[9:8]		---	---	PWMD0[9:8]	
Read/Write	---	---	R/W	R/W	---	---	R/W	R/W
Reset Value	---	---	0	0	---	---	0	0

Bit7~Bit6 Not used

Bit5~Bit4 PWMD1[9:8]: PWM1 higher 2 bits of duty cycle

Bit3~Bit2 Not used

Bit1~Bit0 PWMD0[9:8]: PWM0 higher 2 bits of duty cycle.

Note: Writing to PWMD1 [9:8] does not take effect immediately, and a write to PWMD1L operation is required to take effect.

Writing to PWMD0 [9:8] does not take effect immediately, and requires a write to PWMD0L operation to take effect.

PWM2 and PWM3 duty cycle higher bits register PWMD23H (0EH).

0EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD23H	---	---	PWMD3[9:8]		---	---	PWMD2[9:8]	
Read/Write	---	---	R/W	R/W	---	---	R/W	R/W
Reset Value	---	---	0	0	---	---	0	0

Bit7~Bit6            Not used  
 Bit5~Bit4        PWMD3[9:8]:   PWM3 higher 2 bits of duty cycle.  
 Bit3~Bit2            Not used  
 Bit1~Bit0        PWMD2[9:8]:   PWM2 higher 2 bits of duty cycle.

Note: Writing to PWMD3 [9:8] does not take effect immediately, and a write to PWMD3L operation is required to take effect. Writing to PWMD2 [9:8] does not take effect immediately, and a write to PWMD2L operation is required to take effect.

PWM0 and PWM1 dead zone time register PWM01DT (0FH)

0FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM01DT	---	---	PWM01DT[5: 0]					
Read/Write	---	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset Value	---	---	0	0	0	0	0	0

Bit7~Bit6            Not used  
 Bit5~Bit0        PWM01DT[5: 0]:   PWM0 and PWM1 dead zone time。

PWM2 and PWM3 dead zone time register PWM23DT (10H)

10H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM23DT	---	---	PWM23DT[5: 0]					
Read/Write	---	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset Value	---	---	0	0	0	0	0	0

Bit7~Bit6            Not used  
 Bit5~Bit0        PWM23DT[5: 0]:   PWM2 and PWM3 dead zone time。

### 13.3 PWM Register Write Sequence

Since the 10-bit PWM duty cycle value is allocated in two registers, when modifying the duty cycle, the program always modifies the two registers successively, in order to ensure the correctness of the duty cycle value, the chip internally designed a cache loading function. The operation of the 10-bit duty cycle value needs to be carried out strictly in the following order:

- 1) Write a high 2-digit value, at this time the high 2-digit value is only written to the internal cache;
- 2) Write the low 8-bit value, at which point the full 10-bit duty cycle value is latched.

### 13.4 PWM Period

The PWM period is specified by writing the PWMTH and PWMTL registers.

Equation 1: PWM Period Calculation Formula:

$$\text{PWM period} = [PWMT + 1] * T_{osc} * (CLKDIV \text{ frequency division value})$$

Note:  $T_{osc} = 1/F_{osc}$

When the PWM cycle counter is equal to PWMT, the following event occurs in the next increment count cycle:

- ◆ The PWM period counter is cleared to zero;
- ◆ The PWMx pin is set to 1;
- ◆ The PWM new period value is latched;
- ◆ The PWM new duty cycle value is latched;
- ◆ Generate PWM cycle interrupt flag bits; (PWM0~3 modules only).

## 13.5 PWM Duty Cycle

The PWM duty cycle can be specified by writing a 10-bit value to the following multiple registers: PWMDxL, PWMDxxH.

The PWMDxL and PWMDxxH registers can be written at any time, but the value of the duty cycle is not updated to the internal latch until the PWM cycle counter is equal to PWMT (i.e. end of the cycle).

Equation 2: Pulse width calculation formula:

$$\text{Pulse width} = (\text{PWMDx}[9:0] + 1) * T_{\text{osc}} * (\text{CLKDIV frequency division value})$$

Equation 3: PWM Duty Cycle Calculation Formula:

$$\text{duty cycle} = \frac{\text{PWMDx}[9:0] + 1}{\text{PWMT}[9:0] + 1}$$

Both the PWM cycle and the PWM duty cycle have double buffers inside the chip. This double buffering structure is extremely important to avoid glitches during PWM operation.

## 13.6 Change in System Clock Frequency

The PWM frequency is only related to the chip oscillation clock, and any change in the system clock frequency will not affect the PWM frequency.

## 13.7 Programmable Dead-time Delay Mode

The complementary output mode can be enabled by setting the PWMxDT\_EN, and the dead time delay function can be automatically enabled after the complementary output is enabled.

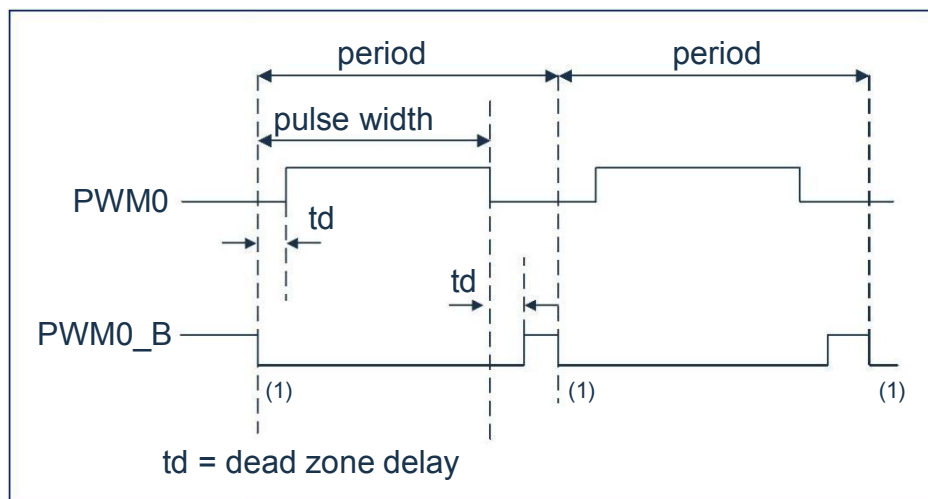


Figure 13-1: Example of PWM dead-time delay output

The dead zone time calculation common formula is:

$$td = (\text{PWMxDT}[5:0] + 1) * T_{\text{osc}} * (\text{DT\_DIV frequency division value})$$

## 13.8 PWM Settings

The following steps should be performed when using the PWM module:

1. Set the IO\_SEL control bit and select the PWM output IO port;
2. By placing the corresponding TRIS position 1, make it the input pin;
3. By loading PWMTH, the PWMTL register sets the PWM cycle;
4. By loading PWMDxxH, PWMDxL register sets the PWM duty cycle;
5. To use the complementary output mode, set the PWMCON1 [6: 5] bit and load the PWMxxDT register to set the dead zone time;
6. Clear the PWMIF flag bit;
7. Set the PWMCN0 [4: 0] bit to enable the corresponding PWM output;
8. After the start of a new PWM cycle, enable the PWM output:
  - Wait for PWMIF position 1;
  - Enable the PWM pin output driver by zeroing the corresponding TRIS bit.

## 13.9 PWM Cycle Interrupt

The PWM0~3 module will touch the PWM cycle interrupt flag at the end of the cycle, and PWM4 will not trigger an interrupt.

## 14. Universal Synchronous/asynchronous Transceiver (USART)

The universal synchronous/asynchronous transmitter (USART) mod is a serial I/O communication peripheral. This mod includes all the clock generators, shift registers and data buffers necessary to perform input or output serial data transmissions that are not related to device program execution. USART It can also be called a serial communication interface (Serial Communications Interface, SCI), it can be configured as a duplex asynchronous system that can communicate with peripherals such as CRT terminals and personal computers; it can also be configured as an integrated circuit with A/D or D/A , Serial EEPROM and other peripherals or half-duplex synchronous system of other microcontroller communication. The microcontroller with which it communicates usually does not have an internal clock that generates baud rate, it needs a master control synchronous device to provide an external clock signal.

The USART mod includes the following functions:

- ◆ Duplex asynchronous transmit and receive
  - ◆ Single character output buffer
  - ◆ Double character input buffer
  - ◆ Frame error detection from receive to character
  - ◆ Half-duplex synchronous slave mode
  - ◆ Character length can be programmed to 8 or 9 bits
  - ◆ Input buffer overflow error detection
  - ◆ Half-duplex synchronous master control mode
- In synchronous mode, programmable clock polarity

Figure 14-1 and Figure 14-2 below are the block diagrams of the USART transmitter.

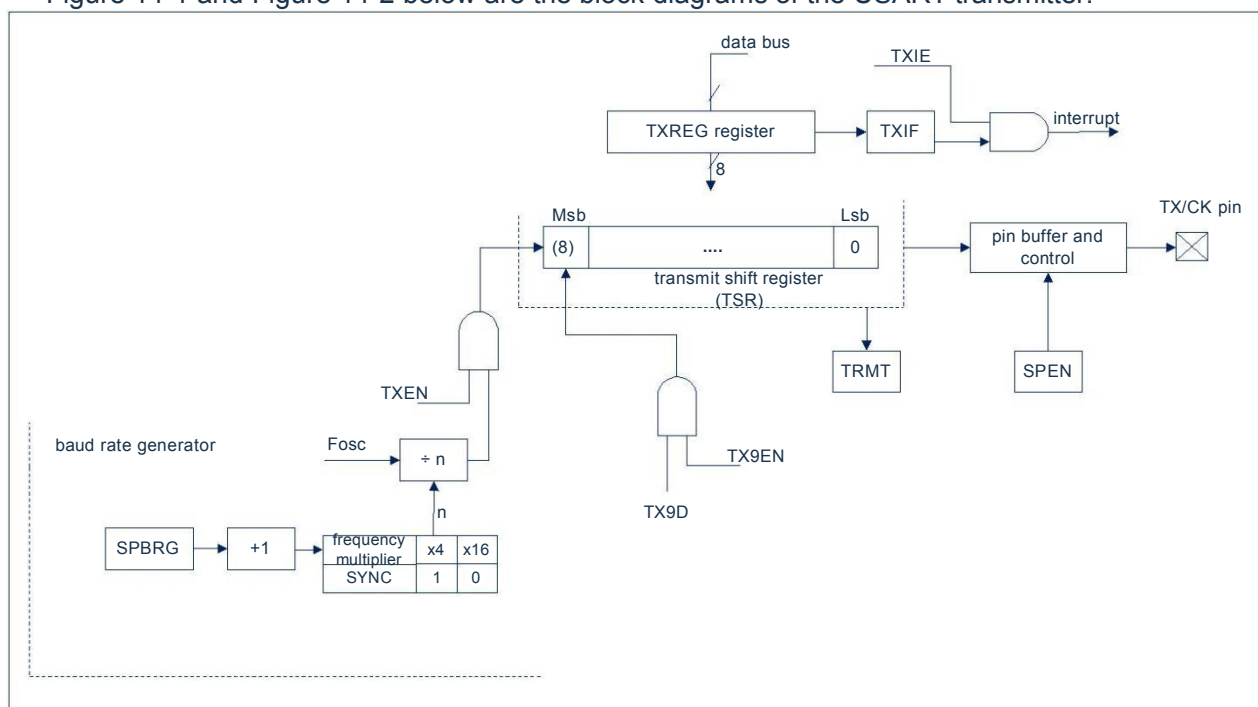


Fig 14-1: USART transmit block diagram



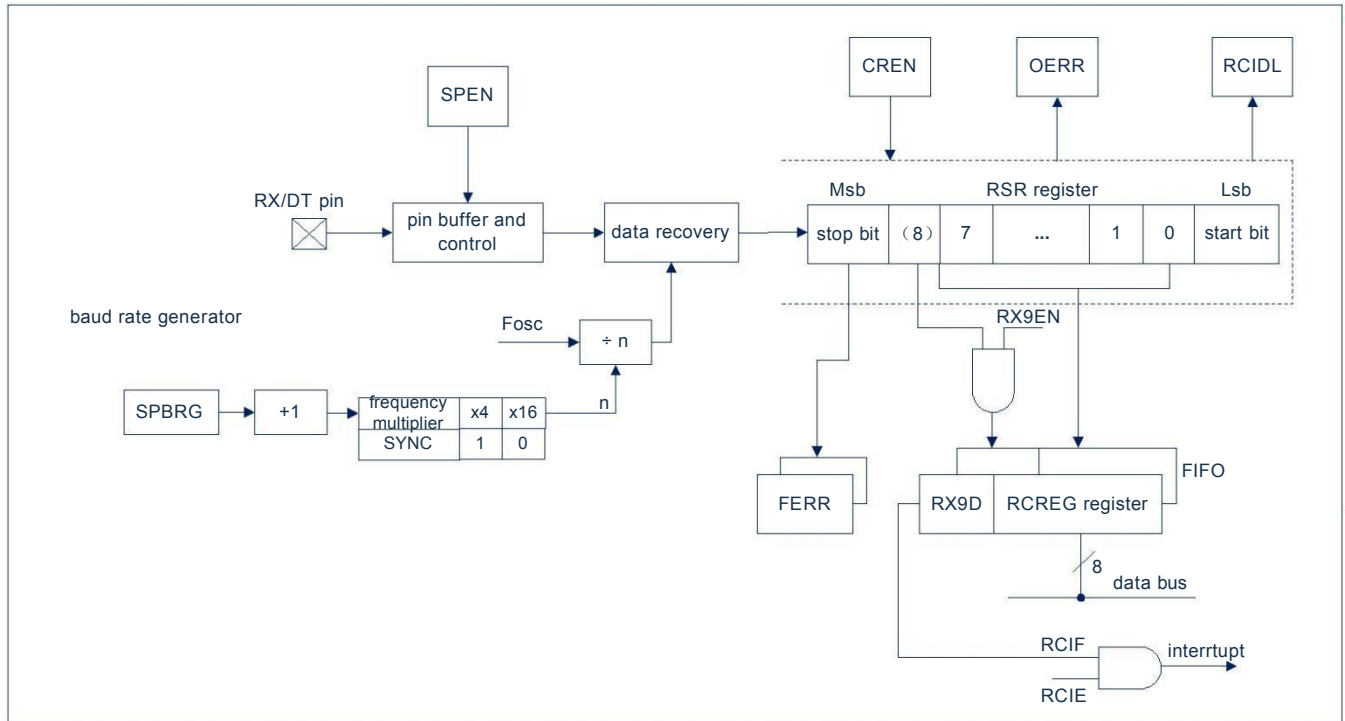


Fig 14-2: USART receive block diagram

The operation of the USART mod is controlled by 2 registers:

- transmit status and control register (TXSTA)
- Receive status and control register (RCSTA)

## 14.1 USART Asynchronous Mode

USART uses the standard non-return-to-zero (NRZ) format for transmit and receive data. Two levels are used to implement NRZ:

It represents the VOH mark state (mark state) of 1 data bit, and the VOL space state (space state) of 0 data bit. When using NRZ format to continuously transmit data bits of the same value, the output level will maintain the level of the bit, and It will return the mid-level value after each bit is transmitted. NRZ transmit port is idle in the mark state. The character of each transmit includes a start bit, followed by 8 or 9 data bits and one or more terminations The stop bit of character transmit. The start bit is always in the space state, and the stop bit is always in the mark state. The most commonly used data format is 8 bits. The duration of each transmit bit is  $1/\text{(baud rate)}$ . On-chip dedicated 8 Bit/16-bit baud rate generator can be used to generate standard baud rate frequency through system oscillator.

USART first transmit and receive LSB. USART's transmitter and receiver are functionally independent, but use the same data format and baud rate. Hardware does not support parity check, but it can be implemented by software (parity bit is the first 9 data bits).

### 14.1.1 USART Asynchronous Generator

Figure 14-1 shows the block diagram of the USART transmit device. The core of the transmit device is the serial transmit shift register (TSR), which cannot be directly accessed by software. TSR obtains data from the TXREG transmit buffer register.

#### 14.1.1.1 Enable Transmit

Enable USART transmit by configuring the following three control bits for asynchronous operation:

- TXEN=1
- SYNC=0
- SPEN=1

It is assumed that all other USART control bits are in their default state.

Set the TXEN bit of the TXSTA register to 1 to enable the USART transmitter circuit. Clear the SYNC bit of the TXSTA register to zero and use the USART configuration for asynchronous operation.

Note:

- 1) When the SPEN bit and TXEN bit are set to 1, the SYNC bit is cleared, TX/CKI/O pin is automatically configured as an output pin, regardless of the state of the corresponding TRIS bit.
- 2) When the SPEN bit and CREN bit are set to 1, the SYNC bit is cleared, and RX/DTI/O pin is automatically configured as an input pin, regardless of the state of the corresponding TRIS bit.

#### 14.1.1.2 Transmit Data

Write a character to the TXREG register to start transmit. If this is the first character, or the previous character has been completely removed from the TSR, the data in TXREG will be immediately transmitted to the TSR register. If all or part of the TSR is still stored The previous character, the new character data will be stored in TXREG until the stop bit of the previous character is transmitted. Then, after the stop bit is transmitted, after a TCY, the data to be processed in TXREG will be transmitted to TSR. When After data is transmitted from TXREG to TSR, the start bit, data bit, and stop bit sequence are transmitted immediately.

#### 14.1.1.3 Transmit Interrupt

As long as the USART transmitter is enabled and there is no data to be transmitted in TXREG, the TXIF interrupt flag bit of the PIR1 register is set to 1. In other words, only when the TSR is busy processing the character and there are new characters queued for transmit in the TXREG, the TXIF bit It is in the cleared state. When writing TXREG, the TXIF flag bit is not cleared immediately. TXIF is cleared at the second instructions period after writing the instructions. Querying TXIF immediately after writing TXREG will return an invalid result. TXIF is a read-only bit and cannot Set or cleared by software.

TXIF interrupt can be enabled by setting the TXIE interrupt enable bit of PIE1 register. However, as long as TXREG is empty, the TXIF flag bit will be set to 1 regardless of the status of the TXIE enable bit.

If you want to use interrupt when transmitting data, set the TXIE bit to 1 only when the data is to be transmitted. After writing the last character to be transmitted to TXREG, clear the TXIE interrupt enable bit.

#### 14.1.1.4 TSR Status

The TRMT bit of the TXSTA register indicates the status of the TSR register. The TRMT bit is a read-only bit. When the TSR register is empty, the TRMT bit is set to 1, and when a character is transferred from the TXREG to the TSR register, the TRMT is cleared. The TRMT bit remains Clear the state until all bits are removed from the TSR register. There is no interrupt logic related to this bit, so the user must query this bit to determine the state of the TSR bit.

Note: The TSR register is not mapped to the data memory, so the user cannot directly access it.

#### 14.1.1.5 Transmit 9-bit Character

The USART supports 9-bit character transmit. When the TX9EN bit of the TXSTA register is 1, the USART will shift out 9 bits of each character to be transmitted. The TX9D bit of the TXSTA register is the 9th bit, which is the highest data bit. When the 9-bit data is transmitted, it must Before writing the 8 least significant bits to TXREG, write the TX9D data bit. After writing the TXREG register, the 9 data bits will be transferred to the TSR shift register immediately.

#### 14.1.1.6 Configure Asynchronous Transmit

1. Initialize the SPBRG register to obtain the required baud rate (refer to "USART baud rate generator (BRG) section")
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit to 1.
3. If 9-bit transmit is required, set the TX9EN control bit to 1. When the receiver is set for address detection, set the 9th bit of the data bit to 1, indicating that the 8 lowest data bits are address.
4. Set the TXEN control bit to 1 to enable transmit; this will cause the TXIF interrupt flag bit to be set to 1.
5. If interrupt is required, set the TXIE interrupt enable bit in PIE1 register to 1; if the GIE and PEIE bits in the INTCON register are also set to 1, interrupt will occur immediately.
6. If you choose to transmit 9-bit data, the 9th bit should be loaded into the TX9Ddata bit.
7. Load 8-bit data into TXREG register to start transmitting data.

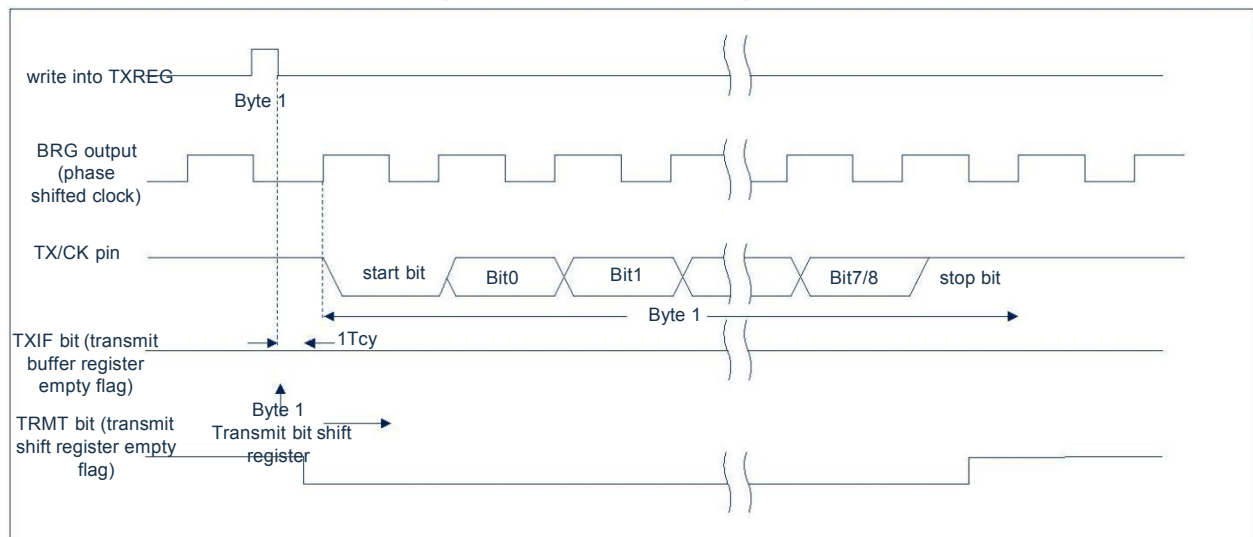


Fig 14-3: asynchronous transmit

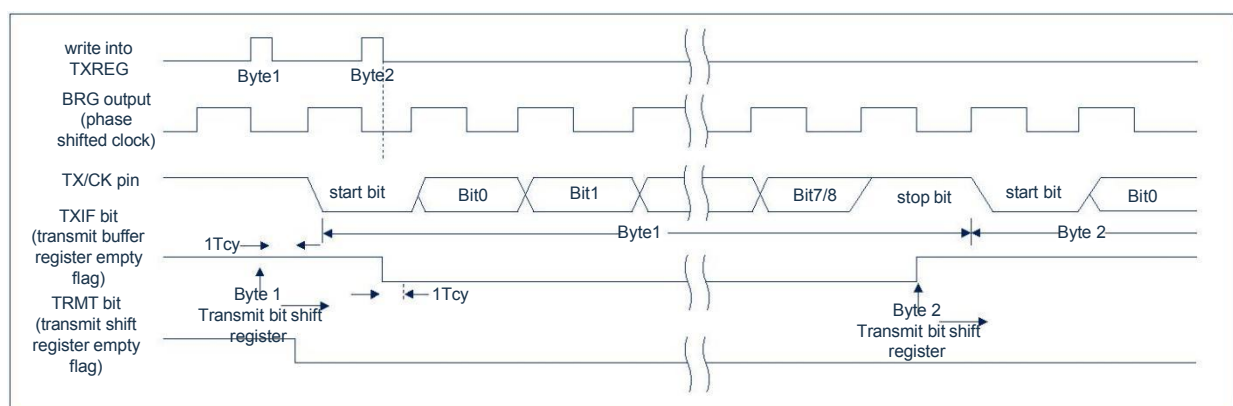


Fig14-4: asynchronous transmit (back to back)

**Note:** This time series diagram shows two consecutive transmit.

### 14.1.2 USART Asynchronous Receiver

Asynchronous mode is usually used in RS-232 system. Figure 14-2 shows the block diagram of the receiver. Receive data and driver data recovery circuit on RX/DT pin. The data recovery circuit is actually a 16 times baud rate as the operating frequency High-speed shifter, while the serial receive shift register (Receive Shift Register, RSR) works at the bit rate. When all the 8-bit or 9-bit data bits of the character are shifted in, they are immediately transferred to a 2-character FIFO (FIFO) buffer. FIFO buffer allows to receive 2 complete characters and the start bit of the third character, and then software must provide the received data to the USART receiver. FIFO and RSR register cannot be directly accessed by software. The RCREG register accesses the received data.

#### 14.1.2.1 Enable Receiver

Enable the USART receiver by configuring the following three control bits for asynchronous operation.

- CREN=1
- SYNC=0
- SPEN=1

Assuming that all other USART control bits are in the default state. Set the CREN bit of the RCSTA register to 1 to enable the USART receiver circuit. Clear the SYNC bit of the TXSTA register to zero and configure the USART for asynchronous operation.

Note:

- 1) When the SPEN bit and TXEN bit are set to 1, the SYNC bit is cleared, and the TX/CKI/O pin is automatically configured as an output pin, regardless of the state of the corresponding TRIS bit.
- 2) When the SPEN bit and CREN bit are set to 1, the SYNC bit is cleared, and the RX/DTI/O pin is automatically configured as an input pin, regardless of the state of the corresponding TRIS bit.

#### 14.1.2.2 Receive Data

Receiver data recovery circuit starts the receive character at the falling edge of the first bit. The first bit, usually called the start bit, is always 0. The data recovery circuit counts half a bit time to the center of the start bit. Check whether the bit is still zero. If the bit is not zero, the data recovery circuit will give up receiving the character without error, and continue to look for the falling edge of the start bit. If the zero check of the start bit passes, then the data recovery circuit counts a complete bit time and reaches the center position of the next bit. The majority detection circuit samples the bit and moves the corresponding sampling result 0 or 1 into the RSR. Repeat the process until all data bits are completed Sampling and moving it all into RSR register. Measure the time of the last bit and sample its level. This bit is the stop bit and is always 1. If the data recovery circuit samples 0 at the stop bit position, the character frame error The flag will be set to 1, otherwise, the frame error flag of the character will be cleared.

When all data bits and stop bits are received, the character in the RSR will be immediately transferred to the receive FIFO of the USART and the RCIF interrupt flag bit of PIR1 register is set to 1. The character at the top of the FIFO is moved out of the FIFO by reading the RCREG register.

Note: If you receive FIFO overflow, you cannot continue to receive other characters until the overflow condition is cleared.

#### 14.1.2.3 Receive Interrupt

As long as the USART receiver is enabled and there is unread data in the receive FIFO, the RCIF interrupt flag bit in the PIR1 register will be set to 1. The RCIF interrupt flag bit is read-only and cannot be set or cleared by software.

RCIF interrupt is enabled by setting all of the following bits:

- RCIE interrupt enable bit of PIE1 register;
- PEIE peripherals interrupt enable bit of INTCON register;
- GIE global interrupt enable bit of INTCON register.

If there is unread data in the FIFO, regardless of the state of the interrupt enable bit, the RCIF interrupt flag bit will be set to 1.

#### 14.1.2.4 Receive Frame Error

Each character in the Receive FIFO buffer has a corresponding frame error status bit. The frame error indicates that the stop bit was not received within the expected time.

The framing error status is obtained by the FERR bit of the RCSTA register. The FERR bit must be read after reading the RCREG register.

Framing error (FERR=1) will not prevent receiving more characters. There is no need to clear the FERR bit.

Clearing the SPEN bit of the RCSTA register will reset the USART and forcibly clear the FERR bit. Framing error itself will not cause interrupt.

Note: If all characters received in the receive FIFO buffer have framing errors, repeated reading of RCREG will not clear the FERR bit.

#### 14.1.2.5 Receive Overflow Error

The receive FIFO buffer can store 2 characters. However, if the third character is received before accessing the FIFO, an overflow error will occur. At this time, the OERR bit of the RCSTA register will be set to 1. The character inside FIFO buffer can be read, but before the error is cleared, no other characters can be received. The error can be cleared by clearing the CREN bit of the RCSTA register or by clearing the SPEN bit of the RCSTA register to make USART reset.

#### 14.1.2.6 Receive 9-bit Character

The USART supports 9-bit data receive. When the RX9EN bit of the RCSTA register is set to 1, the USART will shift the 9 bits of each character received into the RSR. You must read the RX9D data bit after reading the lower 8 bits in RCREG.

### 14.1.2.7 Asynchronous Receive Configuration

1. Initialize the SPBRG register to obtain the required baud rate.  
(Please refer to the "USART baud rate generator (BRG)" section.)
2. Set the SPEN bit to 1 to enable the serial port. The SYNC bit must be cleared to perform asynchronous operations.
3. If interrupt is required, set the RCIE bit in the PIE1 register and the GIE and PEIE bits in the INTCON register to 1.
4. If you need to receive 9 bits of data, set the RX9EN bit to 1.
5. Set the CREN bit to 1 to enable receive.
6. When a character is transferred from the RSR to the receive buffer, set the RCIF interrupt flag bit to 1. If the RCIE interrupt enable bit is also set to 1, an interrupt will also be generated.
7. Read the RCREG register and get the received 8 low data bits from the receive buffer.
8. Read the RCSTA register to get the error flag bit and the 9th data bit (if 9-bit data receive is enabled).
9. If overflow occurs, clear the OERR flag by clearing the CREN receiver enable bit.

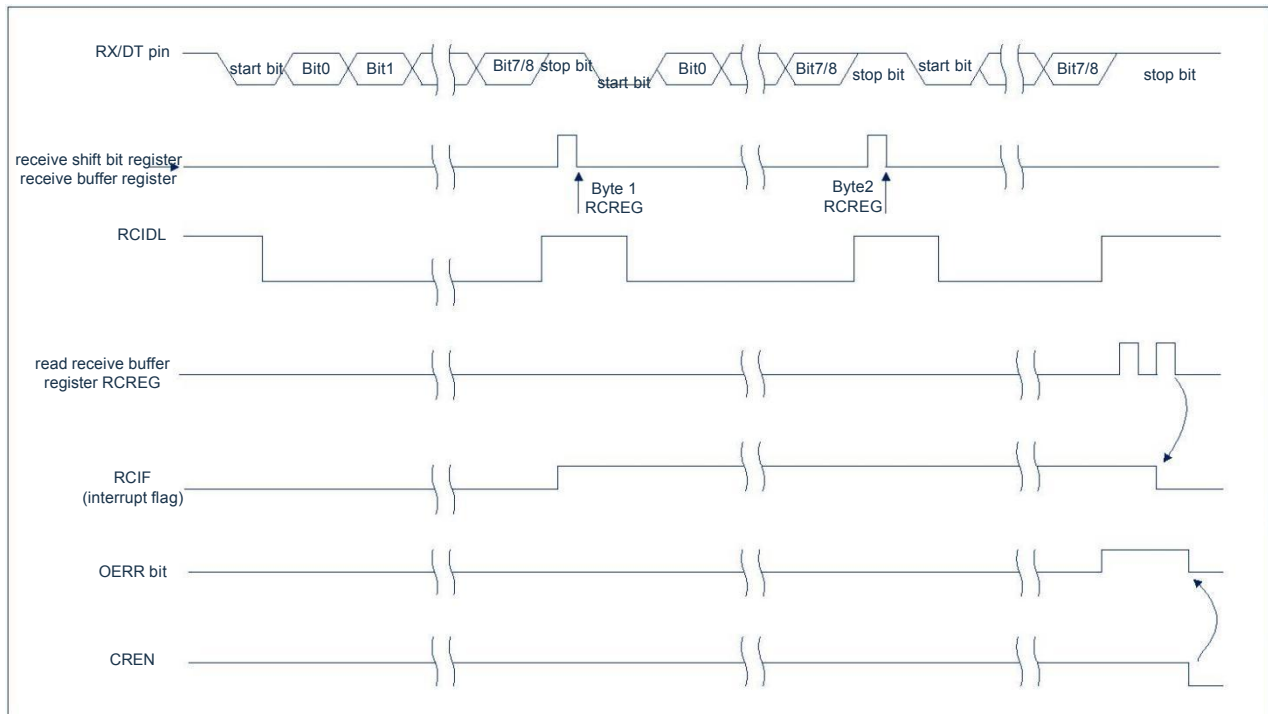


Fig 14-5: asynchronous receive

**Note:** This time series diagram shows the situation of three words received in RX input pin. Reading RCREG (receive buffer) after the third word results in OERR (overflow) bit 1.

## 14.2 Clock Precision for Asynchronous Operations

The output of the internal oscillation circuit (INTOSC) is calibrated by the manufacturer. However, when  $V_{DD}$  or temperature changes, INTOSC will undergo frequency drift, which directly affects the asynchronous baud rate. One way to adjust the value of the baud rate generator. When adjusting the baud rate generator, the adjusted resolution may not be sufficient to compensate for the gradual change in peripheral clock frequency.

## 14.3 USART Related Register

TXSTA: Transmit status and control registers (117H)

117H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TXSTA	CSRC	TX9EN	TXEN (1)	SYNC	SCKP	STOPBIT	TRMT	TX9D
Read and write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Reset value	0	0	0	0	0	0	1	0

Bit7	CSRC:	Clock source select bit;
	Asynchronous mode:	Any value;
	Sync Mode:	1 = Master mode (clock signal generated by the internal BRG); 0 = Driven mode (clock generated by an external clock source).
Bit6	TX9:	9 bits transmit enable bits;
	1=	Select 9 bits to send;
	0=	Select 8 bits to send.
Bit5	TXEN:	Send enable bit (1);
	1=	Enable sending;
	0=	Prohibit sending.
Bit4	SYNC:	USART mode select bit;
	1=	Synchronous mode;
	0=	Asynchronous mode.
Bit3	SCKP:	Synchronous clock polarity select bit.
	Asynchronous mode:	1= Reverse the level of the data character and send it to the TX/CK pin; 0= Send data characters directly to the TX/CK pin.
	Sync Mode:	0= Transmit data on the rising edge of the clock; 1= Transmits data on the clock descent edge.
Bit2	STOPBIT:	Stop bit selection (valid only for asynchronous sends).
	1=	1 stop bit;
	0=	2-bit stop bit. (When the program sends data by judging the TRMT bit, STOPBIT needs to select 2 bits.)
Bit1	TRMT:	Transmit shift register status bits;
	1=	TSR is empty;
	0=	The TSR is full.
Bit0	TX9D:	The 9th bit of the data sent. Can be address/data bits or parity bits.

Note: In synchronous mode, SREN/CREN will subvert the value of TXEN.



### RCSTA: Receive status and control registers (118H)

118H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RCSTA	SPEN	RX9EN	SREN	CREN	RCIDL	FERR	OERR	RX9D
Read and write	R/W	R/W	R/W	R/W	R	R	R	R
Reset value	0	0	0	0	1	0	0	0

Bit7	SPEN:	Serial port enable bit; 1= Enable the serial port (configure the RX/DT and TX/CK pins as serial port pins); 0= Disable the serial port (remain in the reset state).
Bit6	RX9:	9 bits receive enable bits; 1= Select 9-bit Receive; 0= Select 8-bit receive.
Bit5	SREN:	Single-byte receive enable bits. Asynchronous mode: Any value. Synchronous master mode: 1 = Enables single-byte receive; 0 = Single-byte reception is prohibited. Zeros the bit after the receive is complete.
Bit4	Synchronous driven mode:	Any value. CREN: Continuously receive enable bits. Asynchronous mode: 1 = Enable receive; 0 = No receive. Sync Mode: 1 = Enable continuous reception until zero CREN enable bit (CREN overrides SREN); 0 = Continuous reception is prohibited.
Bit3	RCIDL:	Receives the idle flag bit. Asynchronous mode: 1= The receiver is idle 0= The start bit has been received and the receiver is receiving data Sync Mode: Any value.
Bit2	FERR:	Frame error bit. 1= Frame error (can be updated by reading the RCREG register and receives the next valid byte); 0= There are no frame errors.
Bit1	OERR:	Overflow error bit. 1= Overflow error (can be cleared by clearing the CREN bit); 0= There are no overflow errors.
Bit0	RX9D:	Bit 9 of the data received. This bit can be an address/data bit or a parity bit and must be calculated by the user firmware.

## 14.4 USART Baud Rate Generator (BRG)

The baud rate generator (BRG) is an 8-bit, dedicated to supporting the asynchronous and synchronous working modes of USART.

The SPBRG register determines the period of the free-running baud rate timer.

Table 14-1 contains the formula for calculating baud rate. Formula 1 is an example of calculating baud rate and baud rate error.

Table 14-1 shows the typical baud rate and baud rate error values under various asynchronous modes that have been calculated, which is convenient for you to use.

Writing a new value to the SPBRG register pair will cause the BRG timer to reset (or clear). This can ensure that BRG can output a new baud rate without waiting for a timer overflow.

If the system clock changes during a valid receive process, a receive error may occur or data loss may occur. To avoid this problem, the state of the RCIDL bit should be checked to ensure that the receive operation is idle before changing the system clock.

formula1: calculate baud rate error

For device with  $F_{SYS}=8\text{MHz}$ , target baud rate=9600bps, asynchronous mode is 8-bit BRG:

$$\text{target baud rate} = \frac{F_{osc}}{16([SPBRG] + 1)}$$

solve SPBRG:

$$X = \frac{\frac{F_{osc}}{\text{target baud rate}}}{16} - 1 = \frac{\frac{8000000}{9600}}{16} - 1 = [51.08] = 51$$

$$\text{calculated baud rate} = \frac{8000000}{16(51+1)} = 9615$$

$$\text{error} = \frac{\text{calculated baud rate} - \text{target baud rate}}{\text{target baud rate}} = \frac{(9615 - 9600)}{9600} = 0.16\%$$

Table 14-1: baud rate formula

Configuration bit	BRG/USART mode	baud rate formula
SYNC		
0	8 位/asynchronous	$F_{osc}/[16 (n+1)]$
1	8 位/synchronous	$F_{osc}/[4 (n+1)]$

note: n= value of SPBRG register.

Table 14-2: baud rate in asynchronous mode

Target baud rate	SYNC=0					
	$F_{osc} = 8.00\text{MHz}$			$F_{osc} = 16.00\text{MHz}$		
	Real baud rate	error (%)	SPBRG value	Real baud rate	error (%)	SPBRG value
2400	2404	0.16	207	----	----	----
9600	9615	0.16	51	9615	0.16	103
10417	10417	0	47	10417	0	95
14400	14286	-0.8%	34	14286	-0.8%	68
19200	19230	0.16	25	19230	0.16	51

## 14.5 USART Synchronous Mode

Synchronous serial communication is usually used in a system with a master control device and one or more slave devices. The master control device contains the necessary circuits to generate the baud rate clock and provides clock for all devices in the system. The slave device can use master control clock, so no internal clock generation circuit is needed.

In synchronous mode, there are two signal lines: bi-directional data line and clock line. The slave device uses the external clock provided by the master control device to move the serial data in or out of the corresponding receive and transmit shift register. Because of the use of bi-directional data lines, synchronous operation can only use half-duplex mode. Half-duplex means: master control device and slave device can receive and transmit data, but can not receive or transmit at the same time. USART can be used as a master control device, or as a slave device.

### 14.5.1 Synchronous Master Control Mode

The following bits are used to configure the USART for synchronous master control operation:

- SYNC=1
- CSRC=1
- SREN=0 (to transmit); SREN=1 (to receive)
- CREN=0 (to transmit); CREN=1 (to receive)
- SPEN=1

Set the SYNC bit of the TXSTA register to 1 to use the USART configuration for synchronous operation. Set the CSRC bit of the TXSTA register to 1 to configure the device as a master control device. Clear the SREN and CREN bits of the RCSTA register to zero to ensure that the device is in transmit mode. Otherwise, the device is configured to receive mode. Set the SPEN bit of the RCSTA register to 1, enable USART.

#### 14.5.1.1 Master Control Clock

Synchronous data transmission uses an independent clock line to transmit data synchronously. The device configured as a master control device transmits clock signal on the TX/CK pin. When the USART is configured for synchronous transmit or receive operation, the TX/CK output driver automatically enables. Serial data bits are changed on the rising edge of each clock to ensure that they are valid on the falling edge. The time of each data bit is a clock period, and there can only be as many clock periods as there are data bits.

#### 14.5.1.2 Clock Polarity

The device provides clock polarity options to be compatible with Microwire. The clock polarity is selected by the SCKP bit of the TXSTA register. Set the SCKP bit to 1 to set the clock idle state to high. When the SCKP bit is 1, data on the falling edge of each clock changes. Clear the SCKP bit and set the clock idle state to low. When the SCKP bit is cleared, data changes on each rising edge of the clock.

#### 14.5.1.3 Synchronous Master Control Transmit

The RX/DT pin output data of the device. When the USART configuration is synchronous master control transmit operation, the RX/DT and TX/CK output pins of the device are automatically enabled.

Write a character to the TXREG register to start the transmit. If all or part of the previous character is still stored in the TSR, the new character data is stored in TXREG until the stop bit of the previous character is transmitted. If this is the first character, Or the previous character has been completely removed from the TSR, the data in TXREG will be immediately transferred to the TSR register. When the character is transferred from TXREG to TSR, it will immediately begin to transmit data. Each data bit changes on the rising edge of the master control clock and remain effective until the rising edge of the next clock.

Note: The TSR register is not mapped to the data memory, so the user cannot directly access it.

#### 14.5.1.4 Synchronous Master Control Transmit Configuration

1. Initialize the SPBRG register to obtain the required baud rate.  
(Please refer to the chapter "USART baud rate generator (BRG)" section.)
2. Set the SYNC, SPEN and CSRC bits to 1, enable synchronous master control serial port.
3. Clear the SREN and CREN bits to disable receive mode.
4. Set the TXEN bit to 1 to enable transmit mode.
5. If you need to transmit a 9-bit character, set TX9EN to 1.
6. If interrupt is required, set the TXIE bit in the PIE1 register and the GIE and PEIE bits in the INTCON register to 1.
7. If you choose to transmit 9-bit character, you should load the 9th bit of data into the TX9D bit.
8. Start transmit by loading data into TXREG register.

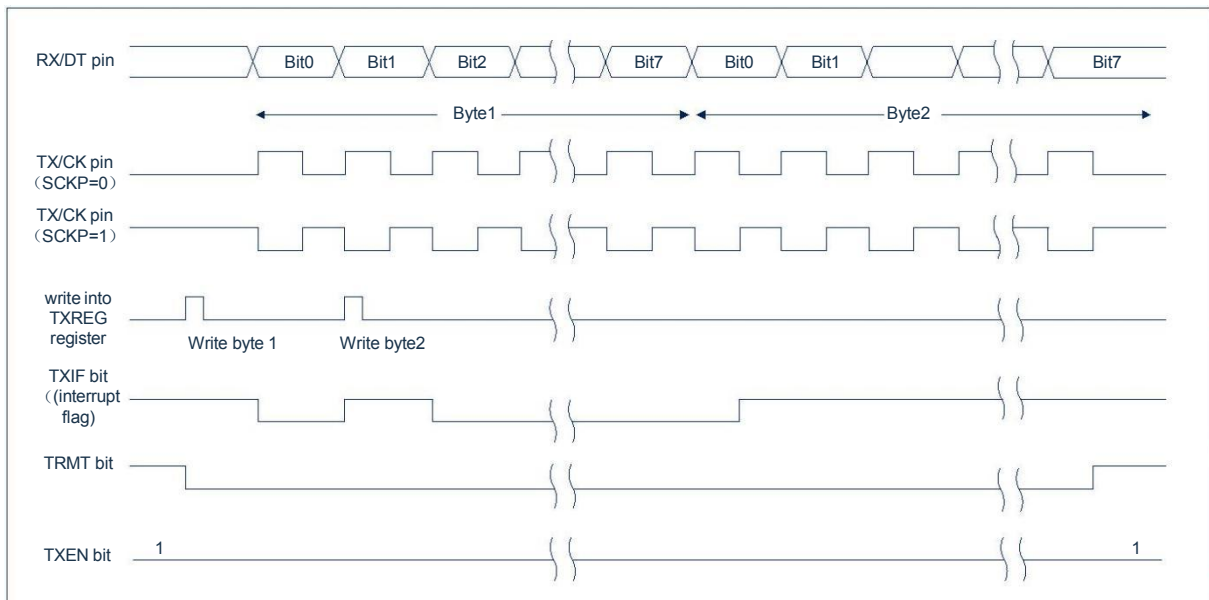


Fig 14-6: synchronous transmit

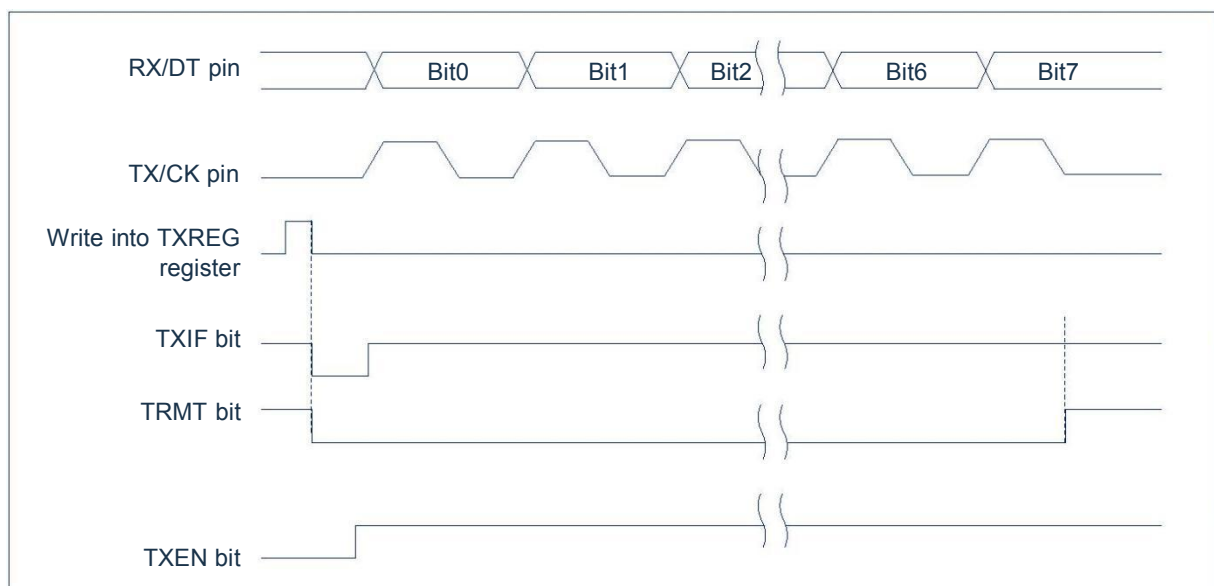


Fig 14-7: synchronous transmit (through TXEN)

#### **14.5.1.5 Synchronous Master Control Receive**

RX/DT pin receive data. When the USART configuration is synchronous master control receive, the output driver of the RX/DT pin of the device is automatically disabled.

In synchronous mode, set the single word receive enable bit (SREN bit of RCSTA register) or continuous receive enable bit (CREN bit of RCSTA register) to 1 enable receive. When SREN is set to 1, the CREN bit is cleared, the number of clock period generated is as much as the number of data bit in single character. After a character transmission is over, the SREN bit is automatically cleared. When CREN is set to 1, a continuous clock will be generated until CREN is cleared. If CREN is cleared during a character transmission, The CK clock stops immediately and discards the incomplete character. If both SREN and CREN are set to 1, when the first character transfer is completed, the SREN bit is cleared, and CREN takes precedence.

Set the SREN or CREN bit to 1, start receiving. Sample the data on RX/DT pin at the falling edge of the TX/CK clock pin signal, and shift the sampled data into the receive shift register (RSR). When the RSR receives a complete character, the RCIF bit is set to 1, the character is automatically moved into the 2 byte receive FIFO. The lower 8 bits of the top character in the receive FIFO can be read through RCREG. As long as there are unread characters in the receive FIFO, the RCIF bit remains as 1.

#### **14.5.1.6 Slave Clock**

Synchronous data transmission uses an independent clock line synchronous with the data line. Clock signal on the TX/CK line of the slave device is received. When the device is configured to operate synchronously from the transmit or receive, the output driver of the TX/CK pin automatically disable. The serial data bit is changed at the leading edge of the clock signal to ensure that it is valid on the back edge of each clock. Each clock period can only transmit one bit of data, so how many data bits must be received is determined by how many data bits transmitted.

#### **14.5.1.7 Receive Overflow Error**

The receive FIFO buffer can store 2 characters. Before reading the RCREG to access the FIFO, if the third character is received completely, an overflow error will occur. At this time, the OERR bit of the RCSTA register will be set to 1. The previous data in the FIFO is not Will be rewritten. Two characters in the FIFO buffer can be read, but before the error is cleared, no other characters can be received. The OERR bit can only be cleared by clearing the overflow condition. If an overflow occurs, the SREN bit is set to 1, the CREN bit is in the cleared state, and the error is cleared by reading the RCREG register. If CREN is set to 1 during overflow, you can clear the CREN bit of the RCSTA register or clear the SPEN bit to reset USART, to clear the error.

#### **14.5.1.8 Receive 9-bit Character**

The USART supports receive 9-bit characters. When the RX9EN bit of the RCSTA register is 1, the USART moves the 9-bit data of each character received into the RSR. When reading 9-bit data from the receive FIFO buffer, it must read 8 lower bit of RCREG first.

### 14.5.1.9 Synchronous Master Control Receive Configuration

1. Initialize the SPBRG register to obtain the required baud rate. (Note: SPBRG>05H must be met)
2. Set the SYNC, SPEN and CSRC bits to 1 to enable synchronous master control serial port.
3. Make sure to clear the CREN and SREN bits.
4. If interrupt is used, set the GIE and PEIE bits of the INTCON register to 1, and set the RCIE bit of the PIE1 register to 1.
5. If you need to receive a 9-bit character, set the RX9EN bit to 1.
6. Set the SREN bit to 1 to enable receive, or set the CREN bit to 1 to enable continuous receive.
7. When the character receive is completed, set the RCIF interrupt flag bit to 1. If the enable bit RCIE is set to 1, an interrupt will also be generated.
8. Read the RCREG register to get the received 8-bit data.
9. Read the RCSTA register to get the 9th data bit (when 9-bit receive is enabled), and judge whether an error occurs during the receive process.
10. If an overflow error occurs, clear the CREN bit of the RCSTA register or clear SPEN to reset USART to clear the error.

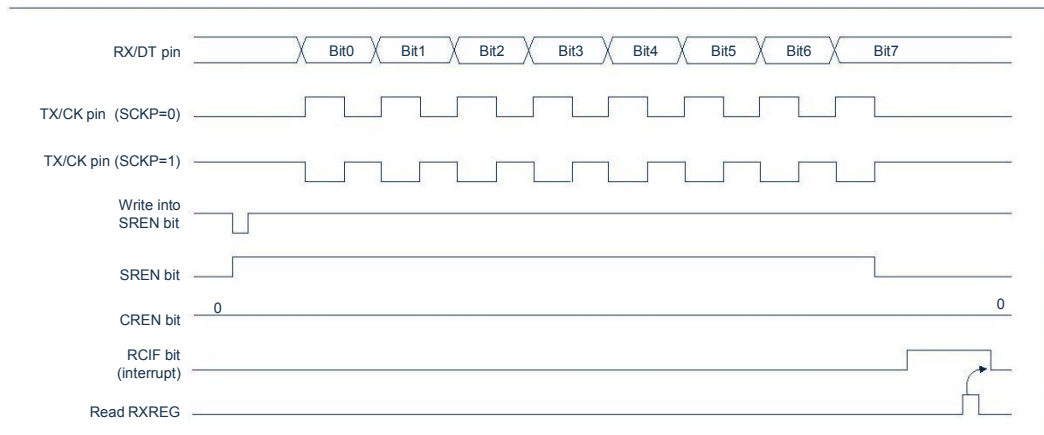


Fig 14-8: synchronous receive (master control mode, SREN)

Note: The time series diagram illustrates the synchronous master control mode when SREN=1.

## 14.5.2 Synchronous Slave Mode

The following bits are used to configure USART for synchronous slave operation:

- SYNC=1
- CSRC=0
- SREN=0 (to transmit); SREN=1 (to receive)
- CREN=0 (to transmit); CREN=1 (to receive)
- SPEN=1

Set the SYNC bit of the TXSTA register to 1 to configure the device for synchronous operation. Set the CSRC bit of the TXSTA register to 1 to configure the device as a slave device. Clear the SREN and CREN bits of the RCSTA register to zero to ensure that the device is in transmit mode. Otherwise, the device will be configured as receive mode. Set the SPEN bit of the RCSTA register to 1, enable USART.

### 14.5.2.1 USART Synchronous Slave Transmit

The working principle of synchronous master control and slave mode is the same (refer to section "synchronous master control transmission").

### 14.5.2.2 Synchronous Slave Transmit Configuration

1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. Clear the CREN and SREN bits.
3. If interrupt is used, set the GIE and PEIE bits of the INTCON register to 1, and also set the TXIE bit of the PIE1 register.
4. If you need to transmit 9-bit data, set the TX9EN bit to 1.
5. Set the TXEN bit to 1 to enable transmit.
6. If you choose to transmit 9-bit data, write the most significant bit to the TX9D bit.
7. Write the lower 8 bits of data to the TXREG register to start transmission.

### 14.5.2.3 USART Synchronous Slave Receive

Except for the following differences, the working principle of synchronous master control and slave mode is the same.

1. The CREN bit is always set to 1, so the receiver cannot enter the idle state.
2. SREN bit, can be "any value" in slave mode.



---

**14.5.2.4 Synchronous Slave Receive Configuration**

1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. If interrupt is used, set the GIE and PEIE bits of the INTCON register to 1, and also set the RCIE bit of the PIE1 register.
3. If you need to receive a 9-bit character, set the RX9EN bit to 1.
4. Set the CREN bit to 1, enable receive.
5. When the receive is completed, set the RCIF bit to 1. If RCIE is set to 1, an interrupt will also be generated.
6. Read the RCREG register and get the received 8 low data bits from the receive FIFO buffer.
7. If you enable 9-bit mode, get the most significant bit from the RX9D bit of the RCSTA register.

If an overflow error occurs, clear the CREN bit of the RCSTA register or clear the SPEN bit to reset USART to clear the error.

## 15. Program EEPROM and Program Memory Control

### 15.1 General

The devices in this series have 2K words of program memory, the address range is from 000H to 7FFFH, which is read-only in all address ranges; the device has a 128 byte program EEPROM, and the address range is 0H to 07FH, which is available in all address ranges. It can be read/write.

These memories are not directly mapped to the register file space, but indirectly addressed through the special function register (SFR). A total of 6 SFR registers are used to access these memories:

- EECON1
- EECON2
- EEDAT
- EEDATH
- EEADR
- EEADRH

When accessing the program EEPROM, the EEDAT register stores 8-bit read/write data, and the EEADR register stores the address of the program EEPROM unit being accessed.

When accessing the program memory of the device, the EEDAT and EEDATH register form a double byte word to save the 16-bit data to be read, and the EEADR and EEADRH register form a double byte word to save the 11-bit EEPROM cell address to be read.

Program memory allows reading in units of bytes. Program EEPROM allows byte read/write. A byte write operation can automatically erase the target cell and write new data (erase before writing).

The writing time is controlled by the on-chip timer. The writing and erasing voltages are generated by the on-chip charge pump, which is rated to work within the voltage range of the device for byte or word operations.

When the device is protected by code, the CPU can still continue to read/write the program EEPROM and program memory. When the code is protected, the device programmer will no longer be able to access the program EEPROM or program memory.

**Note:**

- 1) Program memory refers to ROM space, that is, the space where instructions code is stored, which can only be read; Program EEPROM is a space for storing user data, which can be read/write.
- 2) The normal writing voltage range of program EEPROM is 3.0V~5.5V, writing current is 20mA@V<sub>DD</sub>=5V.

## 15.2 Related Register

### 15.2.1 EEADR and EEADRH Register

The EEADR and EEADRH registers can address up to 128 bytes of program EEPROM or up to 2K bytes of program memory.

When the program memory address value is selected, the high byte of the address is written into the EEADRH register and the low byte is written into the EEADR register. When the program EEPROM address value is selected, only the low byte of the address is written into the EEADR register.

### 15.2.2 EECON1 and EECON2 Register

EECON1 is the control register to access the program EEPROM.

The control bit EEPGD determines whether to access program memory or program EEPROM. When this bit is cleared, as with reset, any subsequent operations will be performed on the program EEPROM. When this bit is set to 1, any subsequent operations will be performed on the program memory. Program memory is read-only.

The control bits RD and WR start reading and writing respectively. Software can only set these bits to 1 and cannot be cleared. After the read or write operation is completed, they are cleared by hardware. Since the WR bit cannot be cleared by software, it can be used to avoid accidentally terminating write operations prematurely.

-When WREN is set to 1, the program EEPROM is allowed to be written. When power is on, the WREN bit is cleared. When the normal write operation is LVR reset or WDT timeout reset interrupt, the WRERR bit will be set to 1. In these cases, after reset, the user can check the WRERR bit and rewrite the corresponding unit.

-When the write operation is completed, the interrupt flag bit EEIF in the PIR1 register is set to 1. This flag bit must be cleared by software.

EECON2 is not a physical register. Reading result of EECON2 is all 0s.

The EECON2 register is only used when executing the program EEPROM write sequence.

EEPROM data register EEDAT (8EH)

8EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEDAT	EEDAT7	EEDAT6	EEDAT5	EEDAT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0
read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	X	X	X	X	X	X	X	X

Bit7~Bit0      EEDAT<7: 0>: To read or write the lower 8 bits of data from the program EEPROM, or read the lower 8 bits of data from the program memory.

EEPROM address register EEADR (90H)

90H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEADR	EEADR7	EEADR6	EEADR5	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0
read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	0	0	0	0	0	0

Bit7~Bit0      EEADR<7: 0>: Specify the lower 8 bits of address for program EEPROM read/write operations, or the lower 8 bits of address for program memory read operations.

EEPROM data register EEDATH (8FH)

8FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEDATH	EEDATH7	EEDATH6	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0
read/write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	X	X	X	X	X	X	X	X

Bit7~Bit0      EEDATH<7: 0>: The upper 8 bits of data read from the program EEPROM/program memory.

EEPROM address register EEADRH (96H)

96H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEADRH	---	---	---	---	---	EEADRH2	EEADRH1	EEADRH0
read/write	---	---	---	---	---	R/W	R/W	R/W
Reset value	---	---	---	---	---	0	0	0

Bit7~Bit3      not used, read 0.

Bit2~Bit0      EEADRH<2: 0>: Specify the upper 3-bit address of the program memory read operation.

## EEPROM control register EECON1 (8CH)

8CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EECON1	EEPGD	---	EETIME1	EETIME0	WRERR	WREN	WR	RD
read/write	R/W	---	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	---	0	0	X	0	0	0

Bit7	EEPGD:	Program/program EEPROM selection bit; 1= Operate program memory; 0= Operate program EEPROM.
Bit6	not used	
Bit5~Bit4	EETIME[1: 0]	Maximum programming waiting time; (For more EETIME information, please refer to Figure 15-1) 00= 1.25ms 01= 2.5ms (VDD=4.0~5.5V, suggested TEMP=0~85°C) 10= 5ms 11= 10ms (suggested other than 2.5ms)
Bit3	WRERR:	EEPROM error flag bit; 1= Write error (any WDT reset or undervoltage reset during normal operation, or the time set by EETIME is up but the self-check has not been successful); 0= Write complete.
Bit2	WREN:	EEPROM write enable bit; 1= Enable write period; 0= Disable write memory.
Bit1	WR:	Write control bit; 1= Start write period (Once the write operation is completed, this bit is cleared by hardware, and the WR bit can only be set to 1, but not cleared by software); 0= Write period complete.
Bit0	RD:	Read control bit; 1= Start the memory read operation (the RD is cleared by hardware, and the RD bit can only be set to 1, but not cleared by software); 0= Not start memory read operation.

## 15.3 Read Program EEPROM

To read the program EEPROM cell, the user must write the address to the EEADR register, clear the EEPGD control bit of the EECON1 register, and then set the control bit RD to 1. Once the read control bit is set, the program EEPROM controller will use the second instruction period to read data. This will cause the second instruction following the “SETB EECON1, RD” instruction to be ignored (1). In the next clock period, the corresponding address value of the program EEPROM will be latched into the EEDAT register. In, the user can read these two registers in subsequent instructions. EEDAT will save this value until the next time the user reads or writes data to the unit.

**Note:** The two instructions after the program memory read operation must be NOP. This prevents the user from executing dual period instructions on the next instruction after the RD position is 1.

example: read program EEPROM

```
EEPDATA_READ:
    LD        A, RADDR        ; Put the address to be read into the EEADR register
    LD        EEADR, A
    CLRB      EECON1, EEPGD    ;access data memory
    SETB      EECON1, RD      ;start reading
    NOP
    NOP
    LD        A, EEDAT         ;read and load data to ACC
    LD        RDATA, A
EEPDATA_READ_BACK:
    RET
```

## 15.4 Write Program EEPROM

To write a program EEPROM storage unit, the user should first write the unit's address to the EEADR register and write data to the EEDAT register. Then the user must start writing each byte in a specific order.

If you do not follow the following instructions exactly (that is, first write 55h to EECON2, then write Aah to EECON2, and finally set the WR bit to 1) to write each byte, the write operation will not be started. Interrupt should be disabled in this code.

In addition, the WREN bit in EECON1 must be set to 1 to enable write operations. This mechanism can prevent EEPROM from being written by mistake due to code execution errors (abnormal) (i.e. program runaway). When not updating EEPROM, the user should always keep the WREN bit cleared. The WREN bit cannot be cleared by hardware.

After a write process is started, clearing the WREN bit will not affect the write period. Unless the WREN bit is set, the WR bit will not be set to 1. When the write period is completed, the WR bit is cleared by hardware and the EE write is completed interrupt flag bit (EEIF) is set to 1. user can allow this interrupt or query this bit. EEIF must be cleared by software.

Note: During the writing of the program EEPROM, the CPU will stop working, the CLRWDT command must be executed before the writing operation starts to avoid WDT overflow to reset the chip during this period.

example: write program EEPROM

```

EEPDATA_WRITE:
    LD        A, WADDR                ; Put the address to be written into the EEADR
                                        register
    LD        EEADR, A
    LD        A, WDATA                ; put the data to be written to the EEDAT register
    LD        EEDAT, A
    CLRWDTC
    CLR        EECON1
    SETB      EECON1, EETIME0
    SETB      EECON1, EETIME1        ;EE programming time 10ms, user-defined
    CLRB      EECON1, EEPGD          ;access data memory
    SETB      EECON1, WREN          ;enable write period
    CLRB      F_GIE_ON              ;save interrupt enabled status
    SZB       INTCON, GIE
    SETB      F_GIE_ON
    CLRB      INTCON, GIE            ;disable interrupt
    SZB       INTCON, GIE            ;ensure interrupt is disabled
    JP        $-2

    LDIA      055H
    LD        EECON2, A
    LDIA      0AAH
    LD        EECON2, A
    SETB      EECON1, WR            ;start writing
    NOP
    NOP
    CLRWDTC
    CLRB      EECON1, WREN          ;write complete, turn off write enable bit

    SZB       F_GIE_ON              ;restore interrupt enabled status
    SETB      INTCON, GIE

    SNZB      EECON1, WRERR          ;check EEPROM write
    JP        EEPDATA_WRITE_BACK
    SZDECR    WERR_C                ; Exit when the count expires, user-defined
    JP        EEPDATA_WRITE          ;rewrite when EEPROM write error

EEPDATA_WRITE_BACK:
    RET

```



## 15.5 Read Program Memory

To read the program memory unit, the user must write the high and low bits of the address to the EEADR and EEADRH registers respectively, set the EEPGD bit of EECON1 register to 1, and then set the control bit RD to 1. Once the read control bit is set, the program memory controller will use the second instructions period to read data. This will cause the second instructions following the "SETB EECON1, RD" instructions to be ignored. In the next clock period, the value of the corresponding address of the program memory will be latched to EEDAT. In the EEDATH register, the user can read these two registers in the subsequent instructions. The EEDAT and EEDATH register will save this value until the next time the user reads or writes data to the unit.

Note:

- 1) The two instructions after the program memory read operation must be NOP. This prevents the user from executing double period instructions in the next instruction after the RD position is 1.
- 2) If the WR bit is 1 when EEPGD=1, it will reset to 0 immediately without performing any operation.

example: read flash program memory

LD	A, RADDR	; Put the address to be read into the EEADR register
LD	EEADR, A	
LD	A, RADDRH	; Put the high bit of the address to be read into EEADRH register
LD	EEADRH, A	
SETB	EECON1, EEPGD	;select to operate on program memory
SETB	EECON1, RD	;enable read
NOP		
NOP		
LD	A, EEDAT	;save read data
LD	RDATL, A	
LD	A, EEDATH	
LD	RDATH, A	

## 15.6 Write Program Memory

program memory is read only, cannot be written.

## 15.7 Precautions on Program EEPROM

### 15.7.1 Programming Time for Program EEPROM

The program EEPROM programming time is not fixed. The time required to program different data varies from 100us to 10ms. The EETIME bit of the EECON1 register determines the maximum time for program EEPROM programming. Program EEPROM mod built-in self-calibration during the programming process, if the self-verification is successful or the time set by EETIME has expired, the write operation will be terminated when one of the conditions is met. During the programming, the CPU stops working, the peripherals mod works normally, and the program needs to be well dealt with accordingly.

### 15.7.2 Number of Times for Programming EEPROM

The number of times of the program EEPROM are related to the programming time set by EETIME, as well as voltage and temperature. For details, please refer to the following diagram.

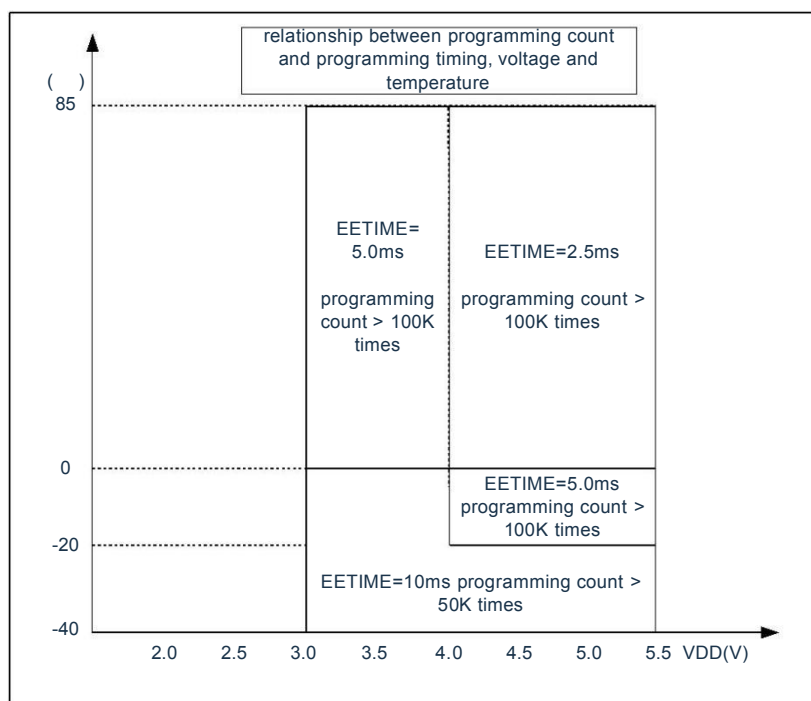


Fig 15-1 The relationship between program EEPROM programming times and programming time, voltage and temperature

### 15.7.3 Write Verification

According to specific applications, good programming habits generally require verification of the value written into the program EEPROM against the expected value.

---

#### **15.7.4 Protection to Avoid Writing Wrongly**

In some cases, the user may not want to write data to the program EEPROM. In order to prevent accidental writing of EEPROM, various protection mechanisms are embedded in the chip. The WREN bit is cleared when the power is turned on. Moreover, the power-on delay timer (the delay time is 18ms) ) Will prevent writing to the EEPROM.

The start sequence of the write operation and the WREN bit will work together to prevent false write operations in the following situations:

- Undervoltage
- Power glitch
- Software failure

## 16. Electrical Parameter

### 16.1 Limit Parameter

Supplying voltage.....	GND-0.3V~GND+6.0V
storage temperature .....	-50°C~125°C
working temperature.....	-40°C~85°C
port input voltage.....	GND-0.3V~V <sub>DD</sub> +0.3V
Maximum source current for all ports .....	200mA
Maximum sink current for all ports .....	-150mA

Note: If the device operating conditions exceed the above "limit parameters", it may cause permanent damage to the device. The above values are only the maximum value of the operating conditions. We do not recommend that the device operate outside the range specified in this specification. The device works for a long time. Under extreme conditions, its stability will be affected.

## 16.2 DC Feature

( $V_{DD}=5V$ ,  $T_A=25^{\circ}C$ , Unless otherwise indicated)

Symbol	Parameter	Test condition		Min	Typ	Max	Unit
		$V_{DD}$	condition				
$V_{DD}$	Working voltage	-	$F_{SYS}=16MHz$	2.6	-	5.5	V
		-	$F_{SYS}=8MHz$	2.0	-	5.5	V
$I_{DD}$	Working current	5V	$F_{SYS}=16MHz$	-	3.8	-	mA
		3V	$F_{SYS}=16MHz$	-	2.6	-	mA
		5V	$F_{SYS}=8MHz$	-	2.7	-	mA
		3V	$F_{SYS}=8MHz$	-	2	-	mA
		5V	Programming EEPROM	-	20	-	mA
		3V	Programming EEPROM	-	10	-	mA
$I_{STB}$	Static current	5V	----	-	0.1	2	$\mu A$
		3V	----	-	0.1	1	$\mu A$
$V_{IL}$	Low level input voltage	-	----	-	-	$0.3V_{DD}$	V
$V_{IH}$	High level input voltage	-	----	$0.7V_{DD}$	-	-	V
$V_{OH}$	High level output voltage	-	No load	$0.9V_{DD}$	-	-	V
$V_{OL}$	Low level output voltage	-	No load	-	-	$0.1V_{DD}$	V
$V_{EEPROM}$	EEPROM mod w/r voltage		----	3.0	-	5.5	V
$R_{PH}$	pull up resistor resistance	5V	$V_O=0.5V_{DD}$	-	38	-	K $\Omega$
		3V	$V_O=0.5V_{DD}$	-	70	-	K $\Omega$
$R_{PL}$	pull down resistor resistance	5V	$V_O=0.5V_{DD}$	-	34	-	K $\Omega$
		3V	$V_O=0.5V_{DD}$	-	60	-	K $\Omega$
$I_{OL1}$	Output port sink current (normal I/O port)	5V	$V_{OL}=0.3V_{DD}$	-	60	-	mA
		3V	$V_{OL}=0.3V_{DD}$	-	25	-	mA
$I_{OL2}$	Output port sink current (PALEN/PBLEN location 1)	5V	$V_{OL}=0.3V_{DD}$	-	120	-	mA
		3V	$V_{OL}=0.3V_{DD}$	-	50	-	mA
$I_{OH1}$	Output port source current (normal I/O port)	5V	$V_{OH}=0.7V_{DD}$	-	-20	-	mA
		3V	$V_{OH}=0.7V_{DD}$	-	-9	-	mA
$I_{OH2}$	Output port source current (PAHEN/PBHEN location 1)	5V	$V_{OH}=0.7V_{DD}$ SEG_ISEL=4mA	-	-3.7	-	mA
		5V	$V_{OH}=0.7V_{DD}$ SEG_ISEL=8mA	-	-7.4	-	mA
		5V	$V_{OH}=0.7V_{DD}$ SEG_ISEL=12mA	-	-11	-	mA
		5V	$V_{OH}=0.7V_{DD}$ SEG_ISEL=16mA	-	-15	-	mA
		5V	$V_{OH}=0.7V_{DD}$ SEG_ISEL=20mA	-	-19	-	mA
		5V	$V_{OH}=0.7V_{DD}$ SEG_ISEL=24mA	-	-23	-	mA
		5V	$V_{OH}=0.7V_{DD}$ SEG_ISEL=28mA	-	-28	-	mA
$V_{BG}$	Internal reference voltage 1.2V	$V_{DD}=2.5\sim 5.5V$ $T_A=25^{\circ}C$		-1.5%	1.2	1.5%	V
		$V_{DD}=2.5\sim 5.5V$ $T_A=-40\sim 85^{\circ}C$		-2.0%	1.2	2.0%	V

## 16.3 ADC Electrical Characteristics

(T<sub>A</sub> = 25°C, Unless otherwise indicated)

Symbol	Parameter	Test conditions	Min	Typ	Max	Unit
V <sub>ADC</sub>	ADC operating voltage	AD <sub>VREF</sub> =V <sub>DD</sub> , F <sub>ADC</sub> =1MHz	3.0	-	5.5	In
		AD <sub>VREF</sub> =V <sub>DD</sub> , F <sub>ADC</sub> =500kHz	2.7	-	5.5	In
		AD <sub>VREF</sub> =2.4V, F <sub>ADC</sub> =250kHz	2.7	-	5.5	In
		AD <sub>VREF</sub> =2.0V, F <sub>ADC</sub> =250kHz	2.7	-	5.5	In
I <sub>ADC</sub>	ADC conversion current	V <sub>ADC</sub> =5V, AD <sub>VREF</sub> =V <sub>DD</sub> , F <sub>ADC</sub> =500kHz	-	-	500	uA
		V <sub>ADC</sub> =3V, AD <sub>VREF</sub> =V <sub>DD</sub> , F <sub>ADC</sub> =500kHz	-	-	200	uA
REHEARSE	ADC input voltage	V <sub>ADC</sub> =5V, AD <sub>VREF</sub> =V <sub>DD</sub> , F <sub>ADC</sub> =250kHz	0	-	V <sub>ADC</sub>	In
DNL	Differential nonlinearity error	V <sub>ADC</sub> =5V, AD <sub>VREF</sub> =V <sub>DD</sub> , F <sub>ADC</sub> =250kHz	-	-	±2	LSB
INL	Integral nonlinearity error	V <sub>ADC</sub> =5V, AD <sub>VREF</sub> =V <sub>DD</sub> , F <sub>ADC</sub> =250kHz	-	-	±2	LSB
I <sub>ADC</sub>	ADC conversion time	-	-	16	-	T <sub>ADCCLK</sub>

## 16.4 ADC Internal LDO Reference Voltage Characteristics

Symbol	Parameter	Test conditions	Min	Typ	Max	Unit
AD <sub>VREF1</sub>	Voltage temperature characteristics of LDO=2.0V	V <sub>DD</sub> =5V T <sub>A</sub> =25°C	-0.6%	2.0	+0.6%	V
		V <sub>DD</sub> =2.7~5.5V T <sub>A</sub> =25°C	-1.0%	2.0	+1.0%	V
		V <sub>DD</sub> =2.7~5.5V T <sub>A</sub> = -40°C~85°C	-1.5%	2.0	+1.5%	V
AD <sub>VREF2</sub>	Voltage temperature characteristics of LDO=2.4V	V <sub>DD</sub> =5V T <sub>A</sub> =25°C	-0.6%	2.4	+0.6%	V
		V <sub>DD</sub> =2.7~5.5V T <sub>A</sub> =25°C	-1.0%	2.4	+1.0%	V
		V <sub>DD</sub> =2.7~5.5V T <sub>A</sub> = -40°C~85°C	-1.5%	2.4	+1.5%	V

## 16.5 Power-on Reset Characteristics

(T<sub>A</sub> = 25°C, Unless otherwise indicated)

Symbol	Parameter	Test conditions	Min	Typ	Max	Unit
T <sub>VDD</sub>	V <sub>DD</sub> rise rate	-	0.05			V/ms
V <sub>LVR2</sub>	LVR set voltage = 2.0V	V <sub>DD</sub> =1.8~5.5V	1.9	2.0	2.1	In
V <sub>LVR1</sub>	LVR set voltage = 2.6V	V <sub>DD</sub> =2.4~5.5V	2.5	2.6	2.7	In

## 16.6 AC Electrical Characteristics

(T<sub>A</sub>=25°C, Unless otherwise indicated)

Symbol	Parameter	Test the condition		Min	Typ	Max	Unit
		V <sub>DD</sub>	Conditions				
T <sub>WDT</sub>	WDT reset time	5V	-	-	16	-	ms
		3V	-	-	16	-	ms
I <sub>EEPROM</sub>	EEPROM programming time	5V	F <sub>HSI</sub> =8MHz/16MHz	-	-	10	ms
		3V	F <sub>HSI</sub> =8MHz/16MHz	-	-	10	ms
F <sub>RC</sub>	Internal vibration frequency stability	V <sub>DD</sub> =4.0~5.5V	T <sub>A</sub> =25°C	-1.5%	8	+1.5%	MHz
		V <sub>DD</sub> =2.5~5.5V	T <sub>A</sub> =25°C	-2.0%	8	+2.0%	MHz
		V <sub>DD</sub> =4.0~5.5V	T <sub>A</sub> = -40~85°C	-2.5%	8	+2.5%	MHz
		V <sub>DD</sub> =2.5~5.5V	T <sub>A</sub> = -40~85°C	-3.5%	8	+3.5%	MHz
		V <sub>DD</sub> =2.0~5.5V	T <sub>A</sub> = -40~85°C	-5.0%	8	+5.0%	MHz
		V <sub>DD</sub> =4.0~5.5V	T <sub>A</sub> =25°C	-1.5%	16	+1.5%	MHz
		V <sub>DD</sub> =2.6~5.5V	T <sub>A</sub> =25°C	-2.0%	16	+2.0%	MHz
		V <sub>DD</sub> =4.0~5.5V	T <sub>A</sub> = -40~85°C	-2.5%	16	+2.5%	MHz
		V <sub>DD</sub> =2.6~5.5V	T <sub>A</sub> = -40~85°C	-3.5%	16	+3.5%	MHz

## 17. Instructions

### 17.1 Instructions Table

Mnemonic	Operation	Instructions period	Symbol
<b>control-3</b>			
NOP	Empty operation	1	None
STOP	Enter sleep mode	1	TO, PD
CLRWDT	Clear watchdog timer	1	TO, PD
<b>Data transfer-4</b>			
LD [R], A	Transfer content to ACC to R	1	NONE
LD A, [R]	Transfer content to R to ACC	1	Z
TESTZ [R]	Transfer the content of data memory data memory	1	Z
LDIA i	Transfer I to ACC	1	NONE
<b>logic operation -16</b>			
CLRA	Clear ACC	1	Z
SET [R]	Set data memory R	1	NONE
CLR [R]	Clear data memory R	1	Z
ORA [R]	Perform 'OR' on R and ACC, save the result to ACC	1	Z
ORR [R]	Perform 'OR' on R and ACC, save the result to R	1	Z
ANDA [R]	Perform 'AND' on R and ACC, save the result to ACC	1	Z
ANDR [R]	Perform 'AND' on R and ACC, save the result to R	1	Z
XORA [R]	Perform 'XOR' on R and ACC, save the result to ACC	1	Z
XORR [R]	Perform 'XOR' on R and ACC, save the result to R	1	Z
SWAPA [R]	Swap R register high and low half byte, save the result to ACC	1	NONE
SWAPR [R]	Swap R register high and low half byte, save the result to R	1	NONE
COMA [R]	The content of R register is reversed, and the result is stored in ACC	1	Z
COMR [R]	The content of R register is reversed and the result is stored in R	1	Z
XORIA i	Perform 'XOR' on i and ACC, save the result to ACC	1	Z
ANDIA i	Perform 'AND' on i and ACC, save the result to ACC	1	Z
ORIA i	Perform 'OR' on i and ACC, save the result to ACC	1	Z
<b>Shift operation-8</b>			
RRCA [R]	Data memory rotates one bit to the right with carry, the result is stored in ACC	1	C
RRCR [R]	Data memory rotates one bit to the right with carry, the result is stored in R	1	C
RLCA [R]	Data memory rotates one bit to the left with carry, the result is stored in ACC	1	C
RLCR [R]	Data memory rotates one bit to the left with carry, the result is stored in R	1	C
RLA [R]	Data memory rotates one bit to the left without carry, and the result is stored in ACC	1	NONE
RLR [R]	Data memory rotates one bit to the left without carry, and the result is stored in R	1	NONE
RRA [R]	Data memory does not take carry and rotates to the right by one bit, and the result is stored in ACC	1	NONE
RRR [R]	Data memory does not take carry and rotates to the right by one bit, and the result is stored in R	1	NONE
<b>increase/decrease-4</b>			
INCA [R]	Increment data memory R, result stored in ACC	1	Z
INCR [R]	Increment data memory R, result stored in R	1	Z



Mnemonic	Operation	Instructions period	Symbol
DECA [R]	Decrement data memory R, result stored in ACC	1	Z
DECR [R]	Decrement data memory R, result stored in R	1	Z
<b>Bit operation-2</b>			
CLRB [R], b	Clear some bit in data memory R	1	NONE
SETB [R], b	Set some bit in data memory R 1	1	NONE
<b>look-up table-2</b>			
TABLE [R]	Read FLASH and save to TABLE_DATAH and R	2	NONE
TABLEA	Read FLASH and save to TABLE_DATAH and ACC	2	NONE
<b>Math operation-16</b>			
ADDA [R]	ACC+[R]→ACC	1	C, DC, Z,
ADDR [R]	ACC+[R]→R	1	C, DC, Z,
ADDCA [R]	ACC+[R]+C→ACC	1	Z, C, DC,
ADDCR [R]	ACC+[R]+C→R	1	Z, C, DC,
ADDIA i	ACC+i→ACC	1	Z, C, DC,
SUBA [R]	[R]-ACC→ACC	1	C, DC, Z,
SUBR [R]	[R]-ACC→R	1	C, DC, Z,
SUBCA [R]	[R]-ACC-C→ACC	1	Z, C, DC,
SUBCR [R]	[R]-ACC-C→R	1	Z, C, DC,
SUBIA i	i-ACC→ACC	1	Z, C, DC,
HSUBA [R]	ACC-[R]→ACC	1	Z, C, DC,
HSUBR [R]	ACC-[R]→R	1	Z, C, DC,
HSUBCA [R]	ACC-[R]- $\overline{C}$ →ACC	1	Z, C, DC,
HSUBCR [R]	ACC-[R]- $\overline{C}$ →R	1	Z, C, DC,
HSUBIA i	ACC-i→ACC	1	Z, C, DC,
<b>Unconditional transfer -5</b>			
RET	Return from subroutine	2	NONE
RET i	Return from subroutine, save I to ACC	2	NONE
RETI	Return from interrupt	2	NONE
CALL ADD	Subroutine call	2	NONE
JP ADD	Unconditional jump	2	NONE
<b>Conditional transfer-8</b>			
SZB [R], b	If the b bit of data memory R is "0", skip the next instruction	1 or 2	NONE
SNZB [R], b	If the b bit of data memory R is "1", skip the next instruction	1 or 2	NONE
SZA [R]	data memory R is sent to ACC, if the content is "0", skip the next instruction	1 or 2	NONE
SZR [R]	If the content of data memory R is "0", skip the next instruction	1 or 2	NONE
SZINCA [R]	Add "1" to data memory R and put the result into ACC, if the result is "0", skip the next one instruction	1 or 2	NONE
SZINCR [R]	Add "1" to data memory R, put the result into R, if the result is "0", skip the next instruction	1 or 2	NONE
SZDECA [R]	Data memory R minus "1", the result is put into ACC, if the result is "0", skip the next instruction	1 or 2	NONE
SZDECR [R]	Data memory R minus "1", put the result into R, if the result is "0", skip the next one instructions	1 or 2	NONE

## 17.2 Instructions Illustration

### **ADDA [R]**

operation: Add ACC to R, save the result to ACC

period: 1

Affected flag bit: C, DC, Z, OV

example:

```
LDIA    09H           ;load 09H to ACC
LD      R01, A        ;load ACC (09H) to R01
LDIA    077H          ;load 77H to ACC
ADDA    R01           ;execute: ACC=09H + 77H =80H
```

### **ADDR [R]**

operation: Add ACC to R , save the result to R

period: 1

Affected flag bit: C, DC, Z, OV

example:

```
LDIA    09H           ;load 09H to ACC
LD      R01, A        ; load ACC (09H) to R01
LDIA    077H          ; load 77H to ACC
ADDR    R01           ;execute: R01=09H + 77H =80H
```

### **ADDCA [R]**

operation: Add ACC to C, save the result to ACC

period: 1

affected flag bit: C, DC, Z, OV

example:

```
LDIA    09H           ; load 09H to ACC
LD      R01, A        ; load ACC (09H) to R01
LDIA    077H          ; load 77H to ACC
ADDCA   R01           ;execute: ACC= 09H + 77H + C=80H (C=0)
                        ACC= 09H + 77H + C=81H (C=1)
```

### **ADDCR [R]**

operation: Add ACC to C, save the result to R

period: 1

affected flag bit: C, DC, Z, OV

example:

```
LDIA    09H           ; load 09H to ACC
LD      R01, A        ; load ACC (09H) to R01
LDIA    077H          ; load 77H to ACC
ADDCR   R01           ;execute: R01 = 09H + 77H + C=80H (C=0)
                        R01 = 09H + 77H + C=81H (C=1)
```

**ADDIA**
**i**

operation: Add i to ACC, save the result to ACC

period: 1

affected flag bit: C, DC, Z, OV

example:

```
LDIA      09H          ; load 09H to ACC
ADDIA     077H         ; execute: ACC = ACC (09H) + i (77H)=80H
```

**ANDA**
**[R]**

operation: Perform 'AND' on register R and ACC, save the result to ACC

period: 1

affected flag Z

example:

```
LDIA      0FH          ; load 0FH to ACC
LD        R01, A        ; load ACC (0FH) to R01
LDIA      77H          ; load 77H to ACC
ANDA      R01          ; execute: ACC = (0FH and 77H)=07H
```

**ANDR**
**[R]**

operation: Perform 'AND' on register R and ACC, save the result to R

period: 1

affected flag bit: Z

example:

```
LDIA      0FH          ; load 0FH to ACC
LD        R01, A        ; load ACC (0FH) to R01
LDIA      77H          ; load 77H to ACC
ANDR      R01          ; execute: R01 = (0FH and 77H)=07H
```

**ANDIA**
**i**

operation: Perform 'AND' on i and ACC, save the result to ACC

period: 1

affected flag Z

example:

```
LDIA      0FH          ; load 0FH to ACC
ANDIA     77H          ; execute: ACC = (0FH and 77H)=07H
```

**CALL**
**add**

operation: Call subroutine

period: 2

affected flag bit: none

example:

```
CALL      LOOP          ; Call the subroutine address whose name is defined as "LOOP"
```

### CLRA

operation: ACC clear

period: 1

affected flag  
bit: Z

example:

```
CLRA                                ;execute: ACC=0
```

### CLR [R]

operation: Register R clear

period: 1

affected flag  
bit: Z

example:

```
CLR      R01                        ;execute: R01=0
```

### CLRB [R], b

operation: Clear b bit on register R

period: 1

affected flag  
bit: none

example:

```
CLRB      R01, 3                    ;execute: 3rd bit of R01 is 0
```

### CLRWDT

operation: Clear watchdog timer

period: 1

affected flag  
bit: TO, PD

example:

```
CLRWDT                                ;watchdog timer clear
```

### COMA [R]

operation: Reverse register R, save the result to ACC

period: 1

affected flag  
bit: Z

example:

```
LDIA      0AH                        ;load 0AH to ACC
LD         R01, A                    ;load ACC (0AH) to R01
COMA      R01                        ;execute: ACC=0F5H
```

**COMR [R]**

operation: Reverse register R, save the result to R

period: 1

affected flag  
bit: Z

example:

```
LDIA    0AH           ; load 0AH to ACC
LD      R01, A         ; load ACC (0AH) to R01
COMR    R01            ;execute: R01=0F5H
```

**DECA [R]**

operation: Decrement value in register , save the result to ACC

period: 1

affected flag  
bit: Z

example:

```
LDIA    0AH           ;load 0AH to ACC
LD      R01, A         ; load ACC (0AH) to R01
DECA    R01            ;execute: ACC= (0AH-1)=09H
```

**DECR [R]**

operation: Decrement value in register , save the result to R

period: 1

affected flag  
bit: Z

example:

```
LDIA    0AH           ; load 0AH to ACC
LD      R01, A         ; load ACC (0AH) to R01
DECR    R01            ;execute: R01= (0AH-1)=09H
```

**HSUBA [R]**

operation: ACC subtract R, save the result to ACC

period: 1

affected flag  
bit: C, DC, Z, OV

example:

```
LDIA    077H           ; load 077H to ACC
LD      R01, A         ; load ACC (077H) to R01
LDIA    080H           ; load 080H to ACC
HSUBA   R01            ;execute: ACC= (80H-77H)=09H
```

**HSUBR [R]**

operation: ACC subtract R, save the result to R

period: 1

affected flag bit: C, DC, Z, OV

example:

```
LDIA    077H           ; load 077H to ACC
LD      R01, A         ; load ACC (077H) to R01
LDIA    080H           ; load 080H to ACC
HSUBR   R01            ;execute: R01= (80H-77H)=09H
```

**HSUBCA [R]**

operation: ACC subtract C, save the result to ACC

period: 1

affected flag bit: C, DC, Z, OV

example:

```
LDIA    077H           ; load 077H to ACC
LD      R01, A         ; load ACC (077H) to R01
LDIA    080H           ; load 080H to ACC
HSUBCA  R01            ;execute: ACC= (80H-77H-C)=09H (C=0)
                        ACC= (80H-77H-C)=08H (C=1)
```

**HSUBCR [R]**

operation: ACC subtract C, save the result to R

period: 1

affected flag bit: C, DC, Z, OV

example:

```
LDIA    077H           ; load 077H to ACC
LD      R01, A         ; load ACC (077H) to R01
LDIA    080H           ; load 080H to ACC
HSUBC   R01            ;execute: R01= (80H-77H-C)=09H (C=0)
R                        R01= (80H-77H-C)=08H (C=1)
```

**INCA [R]**

operation: Register R increment 1, save the result to ACC

period: 1

affected flag bit: Z

example:

```
LDIA    0AH            ; load 0AH to ACC
LD      R01, A         ; load ACC (0AH) to R01
INCA    R01            ;execute: ACC= (0AH+1)=0BH
```

**INCR [R]**

operation: Register R increment 1, save the result to R

period: 1

affected flag bit: Z

example:

```
LDIA    0AH           ; load 0AH to ACC
LD      R01, A        ; load ACC (0AH) to R01
INCR    R01           ; execute: R01= (0AH+1)=0BH
```

**JP add**

operation: Jump to add address

period: 2

affected flag bit: none

example:

```
JP      LOOP          ; jump to the subroutine address whose name is defined as "LOOP"
```

**LD A, [R]**

operation: Load the value of R to ACC

period: 1

affected flag bit: Z

example:

```
LD      A, R01         ; load R01 to ACC
LD      R02, A         ; load ACC to R02, achieve data transfer from R01→R02
```

**LD [R], A**

operation: Load the value of ACC to R

period: 1

affected flag bit: none

example:

```
LDIA    09H           ; load 09H to ACC
LD      R01, A         ; execute: R01=09H
```

**LDIA i**

operation: Load in to ACC

period: 1

affected flag bit: none

example:

```
LDIA    0AH           ; load 0AH to ACC
```

## NOP

operation: Empty instructions

period: 1

affected flag  
bit: none

example:  
  
NOP  
NOP

## ORIA

i

operation: Perform 'OR' on I and ACC, save the result to ACC

period: 1

affected flag  
bit: Z

example:  
  
LDIA        0AH                    ; load 0AH to ACC  
ORIA        030H                ;execute: ACC = (0AH or 30H)=3AH

## ORA

[R]

operation: Perform 'OR' on R and ACC, save the result to ACC

period: 1

affected flag  
bit: Z

example:  
  
LDIA        0AH                    ; load 0AH to ACC  
LD           R01, A                ;load ACC (0AH) to R01  
LDIA        30H                    ;load 30H to ACC  
ORA         R01                    ;execute: ACC= (0AH or 30H)=3AH

## ORR

[R]

operation: Perform 'OR' on R and ACC, save the result to R

period: 1

affected flag  
bit: Z

example:  
  
LDIA        0AH                    ; load 0AH to ACC  
LD           R01, A                ; load ACC (0AH) to R01  
LDIA        30H                    ; load 30H to ACC  
ORR         R01                    ;execute: R01= (0AH or 30H)=3AH



## RET

operation: Return from subroutine

period: 2

affected flag bit: none

example:

```
CALL    LOOP           ; Call subroutine LOOP
NOP                                           ; This statement will be executed after RET instructions return
...                                           ; others
```

LOOP:

```
...                                           ;subroutine
RET                                           ;return
```

## RET

i

operation: Return with parameter from the subroutine, and put the parameter in ACC

period: 2

affected flag bit: none

example:

```
CALL    LOOP           ; Call subroutine LOOP
NOP                                           ; This statement will be executed after RET instructions return
...                                           ;others
```

LOOP:

```
...                                           ;subroutine
RET     35H           ;return, ACC=35H
```

## RETI

operation: Interrupt return

period: 2

affected flag bit: none

example:

```
INT_START                               ;interrupt entrance
...                                     ;interrupt procedure
RETI                                     ;interrupt return
```

## RLCA

[R]

operation: Register R rotates to the left with C and save the result into ACC

period: 1

affected flag bit: C

example:

```
LDIA    03H           ;load 03H to ACC
LD       R01, A        ;load ACC to R01, R01=03H
RLCA     R01           ;operation result: ACC=06H (C=0);
                        ACC=07H (C=1)
                        C=0
```

**RLCR** [R]

operation: Register R rotates one bit to the left with C, and save the result into R

period: 1

affected flag C

example:

```
LDIA    03H           ; load 03H to ACC
LD      R01, A        ; load ACC to R01, R01=03H
RLCR    R01           ;operation result: R01=06H (C=0);
                        R01=07H (C=1);
                        C=0
```

**RLA** [R]

operation: Register R without C rotates to the left, and save the result into ACC

period: 1

affected flag  
bit: none

example:

```
LDIA    03H           ; load 03H to ACC
LD      R01, A        ; load ACC to R01, R01=03H
RLA     R01           ;operation result: ACC=06H
```

**RLR** [R]

operation: Register R without C rotates to the left, and save the result to R

period: 1

affected flag  
bit: none

example:

```
LDIA    03H           ; load 03H to ACC
LD      R01, A        ; load ACC to R01, R01=03H
RLR     R01           ;operation result: R01=06H
```

**RRCA** [R]

operation: Register R rotates one bit to the right with C, and puts the result into ACC

period: 1

affected flag  
bit: C

example:

```
LDIA    03H           ; load 03H to ACC
LD      R01, A        ; load ACC to R01, R01=03H
RRCA    R01           ;operation result: ACC=01H (C=0);
                        ACC=081H (C=1);
                        C=1
```

**RRCR [R]**

operation: Register R rotates one bit to the right with C, and save the result into R

period: 1

affected flag bit: C

example:

```
LDIA    03H           ; load 03H to ACC
LD      R01, A        ; load ACC to R01, R01=03H
RRCR    R01           ;operation result: R01=01H (C=0);
                        R01=81H (C=1);
                        C=1
```

**RRA [R]**

operation: Register R without C rotates one bit to the right, and save the result into ACC

period: 1

affected flag bit: none

example:

```
LDIA    03H           ; load 03H to ACC
LD      R01, A        ; load ACC to R01, R01=03H
RRA     R01           ;operation result: ACC=81H
```

**RRR [R]**

operation: Register R without C rotates one bit to the right, and save the result into R

period: 1

affected flag bit: none

example:

```
LDIA    03H           ; load 03H to ACC
LD      R01, A        ; load ACC to R01, R01=03H
RRR     R01           ;operation result: R01=81H
```

**SET [R]**

operation: Set all bits in register R as 1

period: 1

affected flag bit: none

example:

```
SET     R01           ;operation result: R01=0FFH
```

**SETB [R], b**

operation: Set b bit in register R 1

period: 1

affected flag bit: none

example:

```
CLR     R01           ;R01=0
SETB    R01, 3        ;operation result: R01=08H
```

## STOP

operation: Enter sleep

period: 1

affected flag TO, PD

example:

```
STOP ; The chip enters the power saving mode, the CPU and oscillator
stop working, and the IO port keeps the original state
```

## SUBIA

i

operation: ACC minus I, save the result to ACC

period: 1

affected flag C, DC, Z, OV  
bit:

example:

```
LDIA    077H    ;load 77H to ACC
SUBIA    80H    ;operation result: ACC=80H-77H=09H
```

## SUBA

[R]

operation: Register R minus ACC, save the result to ACC

period: 1

affected flag C, DC, Z, OV  
bit:

example:

```
LDIA    080H    ;load 80H to ACC
LD      R01, A   ;load ACC to R01, R01=80H
LDIA    77H     ;load 77H to ACC
SUBA    R01     ;operation result: ACC=80H-77H=09H
```

## SUBR

[R]

operation: Register R minus ACC, save the result to R

period: 1

affected flag C, DC, Z, OV  
bit:

example:

```
LDIA    080H    ; load 80H to ACC
LD      R01, A   ; load ACC to R01, R01=80H
LDIA    77H     ; load 77H to ACC
SUBR    R01     ;operation result: R01=80H-77H=09H
```

**SUBCA [R]**

operation: Register R minus ACC minus C, save the result to ACC

period: 1

affected flag bit: C, DC, Z, OV

example:

```
LDIA      080H      ; load 80H to ACC
LD        R01, A    ; load ACC to R01, R01=80H
LDIA      77H       ; load 77H to ACC
SUBCA     R01        ; operation result: ACC=80H-77H-C=09H (C=0);
                      ACC=80H-77H-C=08H (C=1);
```

**SUBCR [R]**

operation: Register R minus ACC minus C, save the result to ACC

period: 1

affected flag bit: C, DC, Z, OV

example:

```
LDIA      080H      ; load 80H to ACC
LD        R01, A    ; load ACC to R01, R01=80H
LDIA      77H       ; load 77H to ACC
SUBCR     R01        ; operation result: R01=80H-77H-C=09H (C=0);
                      R01=80H-77H-C=08H (C=1)
```

**SWAPA [R]**

operation: Register R high and low half byte swap, the save result into ACC

period: 1

affected flag bit: none

example:

```
LDIA      035H      ;load 35H to ACC
LD        R01, A    ; load ACC to R01, R01=35H
SWAPA     R01        ;operation result: ACC=53H
```

**SWAPR [R]**

operation: Register R high and low half byte swap, the save result into R

period: 1

affected flag bit: none

example:

```
LDIA      035H      ; load 35H to ACC
LD        R01, A    ; load ACC to R01, R01=35H
SWAPR     R01        ;operation result: R01=53H
```

**SZB [R], b**

operation: Determine the bit b of register R, if it is 0 then jump, otherwise execute in sequence

period: 1 or 2

affected flag bit: none

example:

```
SZB      R01, 3      ;determine 3rd bit of R01
JP       LOOP        ;if is 1, execute, jump to LOOP
JP       LOOP1       ; if is 0, jump, execute, jump to LOOP1
```

**SNZB [R], b**

operation: Determine the bit b of register R, if it is 1 then jump, otherwise execute in sequence

period: 1 or 2

affected flag bit: none

example:

```
SNZB     R01, 3      ; determine 3rd bit of R01
JP       LOOP        ; if is 0, execute, jump to LOOP
JP       LOOP1       ; if is 1, jump, execute, jump to LOOP1
```

**SZA [R]**

operation: Load the value of R to ACC, if it is 0 then jump, otherwise execute in sequence

period: 1 or 2

affected flag bit: none

example:

```
SZA      R01          ;R01→ACC
JP       LOOP        ;if R01 is not 0, execute, jump to LOOP
JP       LOOP1       ;if R01 is 0, jump, execute, jump to LOOP1
```

**SZR [R]**

operation: Load the value of R to R, if it is 0 then jump, otherwise execute in sequence

period: 1 or 2

affected flag bit: None

example:

```
SZR      R01          ;R01→R01
JP       LOOP        ; if R01 is not 0, execute, jump to LOOP
JP       LOOP1       ; if R01 is 0, jump, execute, jump to LOOP1
```

**SZINCA [R]**

operation: Increment register by 1, save the result to ACC, if it is 0 then jump, otherwise execute in sequence

period: 1 or 2

affected flag bit: none

example:

```
SZINCA    R01            ;R01+1→ACC
JP        LOOP          ; if ACC is not 0, execute, jump to LOOP
JP        LOOP1         ; if ACC is 0, jump, execute, jump to LOOP1
```

**SZINCR [R]**

operation: Increment register by 1, save the result to R, if it is 0 then jump, otherwise execute in sequence

period: 1 or 2

affected flag bit: none

example:

```
SZINCR    R01            ;R01+1→R01
JP        LOOP          ; if R01 is not 0, execute, jump to LOOP
JP        LOOP1         ; if R01 is 0, jump, execute, jump to LOOP1
```

**SZDECA [R]**

operation: decrement register by 1, save the result to ACC, if it is 0 then jump, otherwise execute in sequence

period: 1 or 2

affected flag bit: none

example:

```
SZDECA    R01            ;R01-1→ACC
JP        LOOP          ; if ACC is not 0, execute, jump to LOOP
JP        LOOP1         ; if ACC is 0, jump, execute, jump to LOOP1
```

**SZDECR [R]**

operation: Decrement register by 1, save the result to R, if it is 0 then jump, otherwise execute in sequence

period: 1 or 2

affected flag bit: none

example:

```
SZDECR    R01            ;R01-1→R01
JP        LOOP          ; if R01 is not 0, execute, jump to LOOP
JP        LOOP1         ; if R01 is 0, jump, execute, jump to LOOP1
```

**TABLE [R]**

operation: Look-up table, the lower 8 bits of the look-up table result are placed in R, and the high bits are placed in the dedicated register TABLE\_DATAH

period: 2

affected flag bit: none

example:

```
LDIA    01H           ;load 01H to ACC
LD      TABLE_SPH, A ;load ACC to higher bits of table address, TABLE_SPH=1
LDIA    015H          ;load 15H to ACC
LD      TABLE_SPL, A ; load ACC to lower bits of table address, TABLE_SPL=15H

TABLE   R01           ;look-up table 0115H address, operation result:
                        TABLE_DATAH=12H, R01=34H
...
ORG     0115H
DW      1234H
```

**TABLEA**

operation: Look-up table, the lower 8 bits of the look-up table result are placed in ACC, and the high bits are placed in the dedicated register TABLE\_DATAH

period: 2

affected flag bit: none

example:

```
LDIA    01H           ; load 01H to ACC
LD      TABLE_SPH, A ; load ACC to higher bits of table address, TABLE_SPH=1
LDIA    015H          ; load 15H to ACC
LD      TABLE_SPL, A ; load ACC to lower bits of table address, TABLE_SPL=15H

TABLEA           ;look-up table 0115H address, operation result:
                  TABLE_DATAH=12H, ACC=34H
...
ORG     0115H
DW      1234H
```

**TESTZ [R]**

operation: Pass the R to R, as affected Z flag bit

period: 1

affected flag bit: Z

example:

```
TESTZ   R0           ;
SZB     STATUS, Z     ;check Z flag bit, if it is 0 then jump
JP      Add1          ;if R0 is 0, jump to address Add1
JP      Add2          ;if R0 is not 0, jump to address Add2
```



**XORIA****i**

operation: Perform 'XOR' on I and ACC, save the result to ACC

period: 1

affected flag  
bit: Z

example:

```
LDIA      0AH          ;load 0AH to ACC
XORIA     0FH          ;execute: ACC=05H
```

**XORA****[R]**

operation: Perform 'XOR' on I and ACC, save the result to ACC

period: 1

affected flag  
bit: Z

example:

```
LDIA      0AH          ; load 0AH to ACC
LD        R01, A       ;load ACC to R01, R01=0AH
LDIA      0FH          ;load 0FH to ACC
XORA      R01          ;execute: ACC=05H
```

**XORR****[R]**

operation: Perform 'XOR' on I and ACC, save the result to R

period: 1

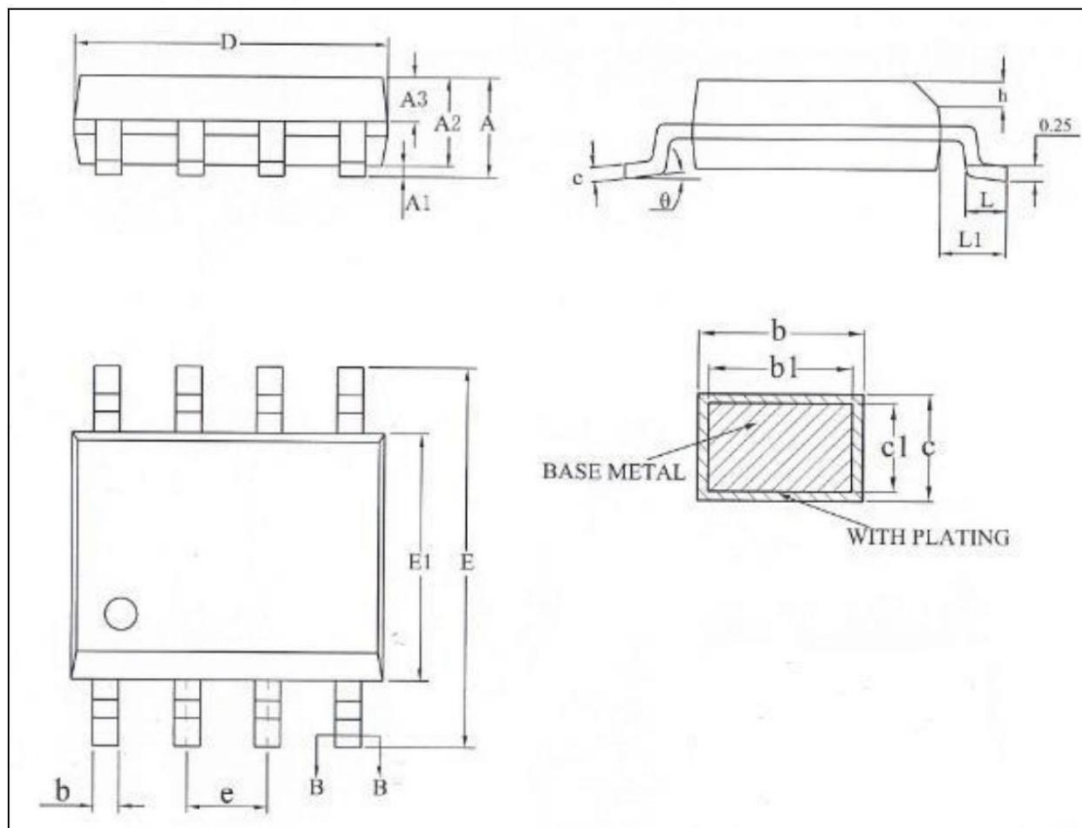
affected flag  
bit: Z

example:

```
LDIA      0AH          ; load 0AH to ACC
LD        R01, A       ; load ACC to R01, R01=0AH
LDIA      0FH          ; load 0FH to ACC
XORR      R01          ;execute: R01=05H
```

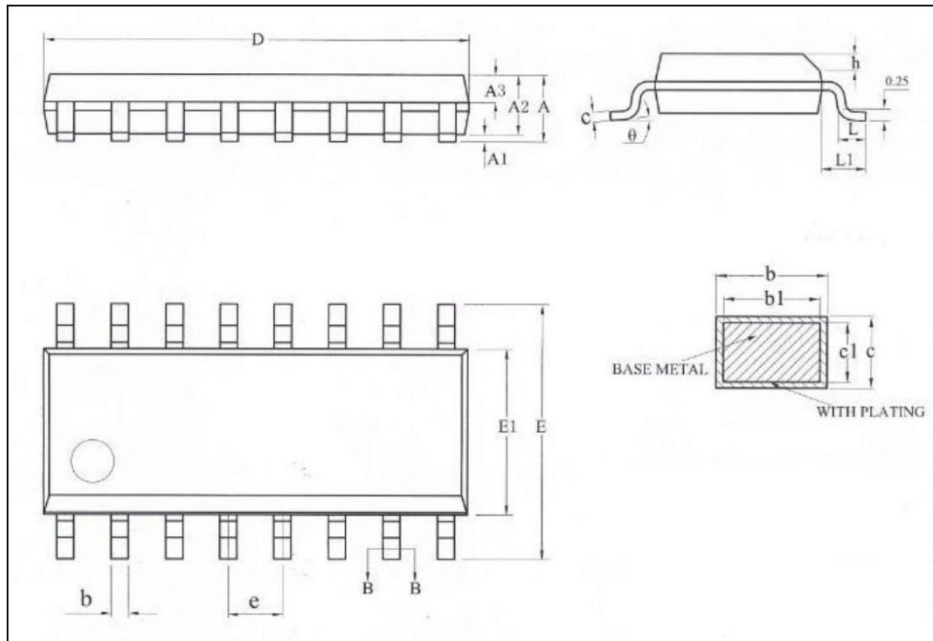
## 18. Packaging

### 18.1 SOP8



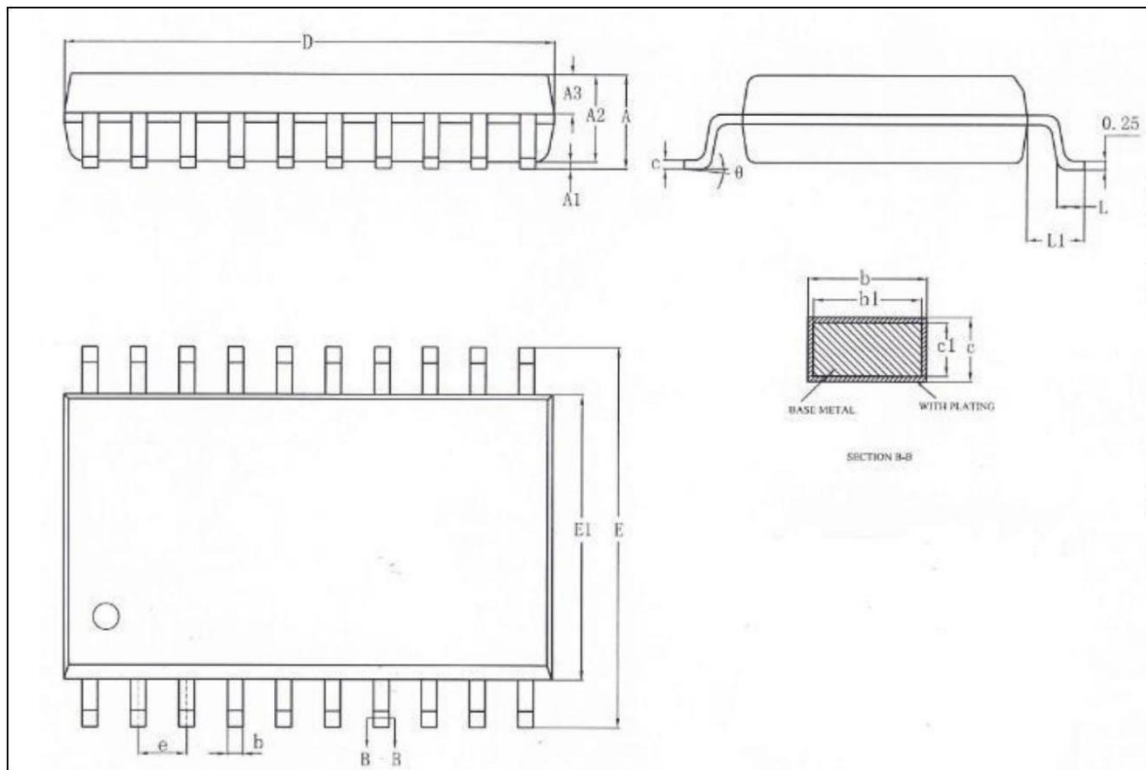
Symbol	Millimeter		
	Min	Nom	Max
A	-	-	1.75
A1	0.10	-	0.225
A2	1.30	1.40	1.50
A3	0.60	0.65	0.70
b	0.39	-	0.47
b1	0.38	0.41	0.44
c	0.20	-	0.24
c1	0.19	0.20	0.21
D	4.80	4.90	5.00
E	5.80	6.00	6.20
E1	3.80	3.90	4.00
e	1.27BSC		
h	0.25	-	0.50
L	0.5	-	0.80
L1	1.05REF		
θ	0	-	8°

## 18.2 SOP16



Symbol	Millimeter		
	Min	Nom	Max
A	-	-	1.75
A1	0.10	-	0.225
A2	1.30	1.40	1.50
A3	0.60	0.65	0.70
b	0.39	-	0.47
b1	0.38	0.41	0.44
c	0.20	-	0.24
c1	0.19	0.20	0.21
D	9.80	9.90	10.00
E	5.80	6.00	6.20
E1	3.80	3.90	4.00
e	1.27BSC		
h	0.25	-	0.50
L	0.50	-	0.80
L1	1.05REF		
θ	0	-	8°

## 18.3 SOP20



Symbol	Millimeter		
	Min	Nom	Max
A	-	-	2.65
A1	0.10	-	0.30
A2	2.25	2.30	2.35
A3	0.97	1.02	1.07
b	0.39	-	0.47
b1	0.38	0.41	0.44
c	0.25	-	0.29
c1	0.24	0.25	0.26
D	12.70	12.80	12.90
E	10.10	10.30	10.50
E1	7.40	7.50	7.60
e	1.27BSC		
L	0.70	-	1.00
L1	1.40REF		
θ	0	-	8°

## 19. Version Revision

Version number	Time	Revised content
V1.0	Jan, 2020	Initial version
V1.1	May, 2022	<ol style="list-style-type: none"> <li>1. correct RCIF and TXIF of PIR1 register to read-only</li> <li>2. correct description of TXIF of PIR1 register</li> <li>3. correct USART asynchronized transmission diagram14-3 and diagram14-4</li> </ol>
V1.2.0	Apr, 2023	<ol style="list-style-type: none"> <li>1. correct the description of receiving interrupts in USART 14.1.2.3</li> <li>2. correct the FERR frame error bit in the RCSTA register to read-only</li> <li>3. Delete description of sleep wake-up in ADC interrupt</li> <li>4. Correct 18.3 Packaging Dimensions</li> <li>5. Add clock block diagram</li> <li>6. Correct the internal high-speed oscillation frequency to <math>F_{HSI}</math> and correct the clock sources of other modules according to the clock block diagram</li> <li>7. Correct the Sleep wakeup wait time and ADC internal LDO reference voltage characteristics</li> </ol>