CMS89F52x user manual

Enhanced 8-bit CMOS microcontroller with flash memory

Rev. 1.7.0

Index

1. Product Description

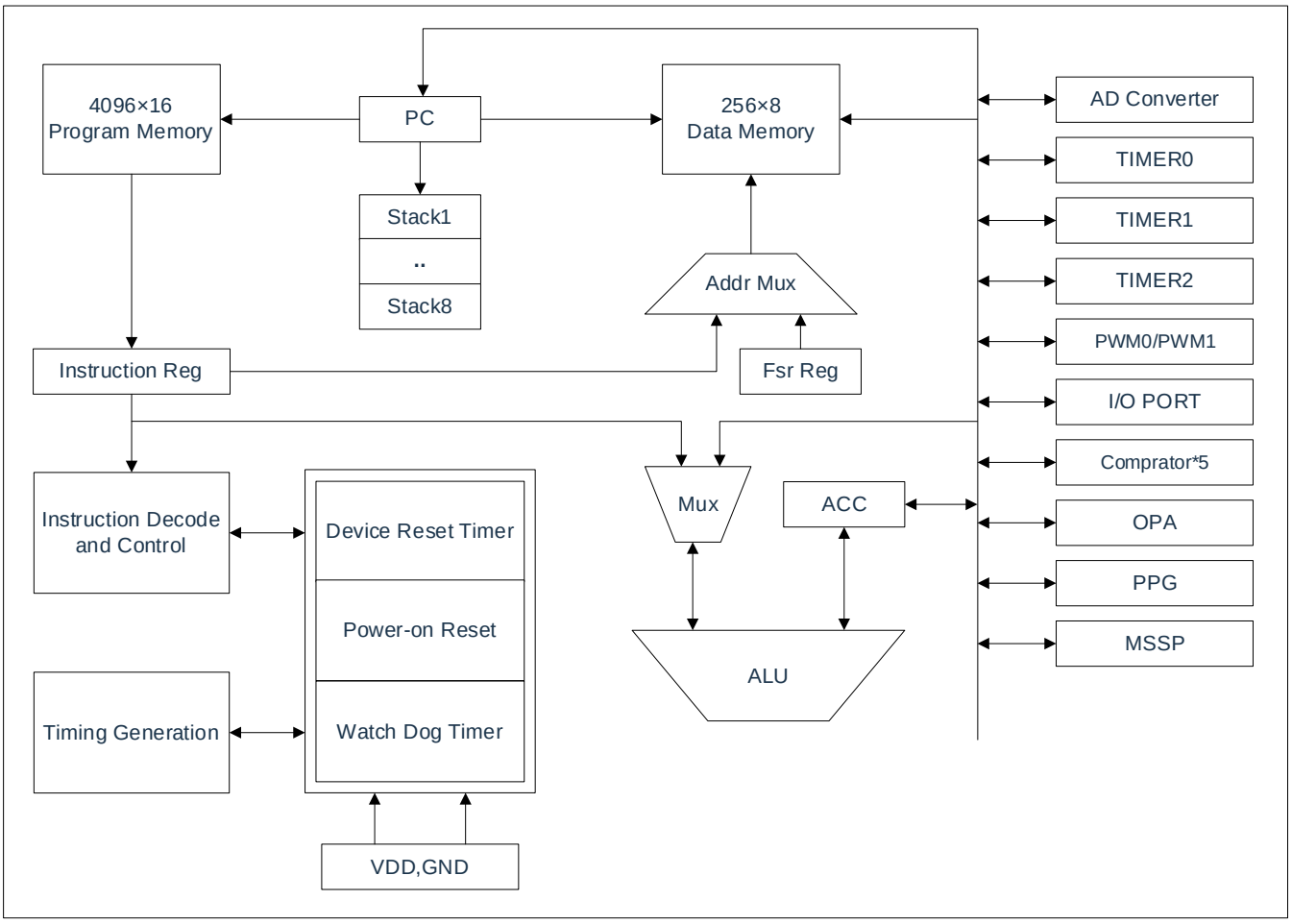
1.1 Features

- memory
  - Flash: 4Kx16Bit
  - Universal RAM: 256x8Bit
  - EEPROM: 32x16Bit
- 8 level stack buffers
- Built-in Low voltage detection circuit
- Interrupt sources:
  - 3 timer interrupts
  - External Interrupt
  - Other peripherals interrupt
- Timers:
  - 8-bit timer: TIMER0, TIMER2
  - 16-bit timer: TIMER1
- 2x 8bits PWM circuit with configurable and adjustable period and duty cycle
- Look up table function
- Working voltage: 3.5V~5.5V@8MHz
  3.5V~5.5V@4MHz
- Working temperature: -40℃~85℃
- Oscillation modes:
  - Internal RC: design frequency of 8MHz/16MHz
- Instruction's period (single instruction or double instructions)
- built-in WDT Timer
- High precision 10-bit ADC
- MSSP communication module (SPI/ I^2C)
- PPG Control module
  - 1 three-port (accessible) op amp, plus/minus ports can select ground internally.
  - 5 comparators in CMP, synchronize/over voltage/ voltage surge/ current surge comparators
  - 10-bits PPG Timer
  - PPG WDT Timer
  - Over voltage self-decremental PPG_Timer

Product specification

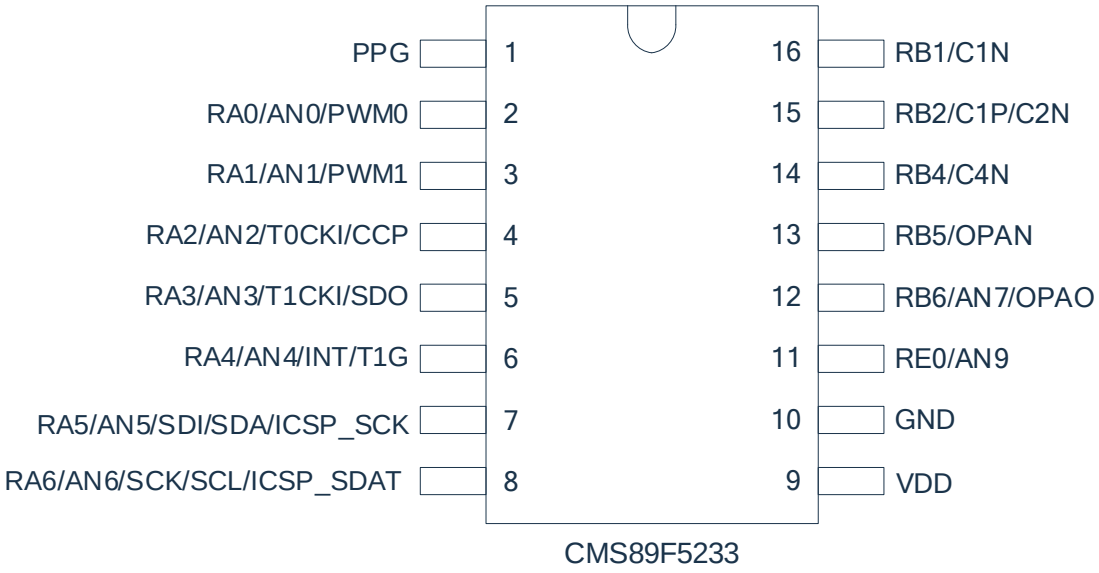
PRODUCT	FLASH	RAM	EEPROM	I/O	ADC	COMP	OPA	PACKAGE	PRODUCT	FLASH
CMS89F5233	4K×16Bit	256	32×16Bit	14	10Bit×9	3	1	DIP16/SOP16	CMS89F5233	4K×16Bit
CMS89F526	4K×16Bit	256	32×16Bit	18	10Bit×11	5	1	DIP20/SOP20	CMS89F526	4K×16Bit

1.2 System Structure Diagram



1.3 Pin Layout

1.3.1 CMS89F5233 Pin Map Diagram

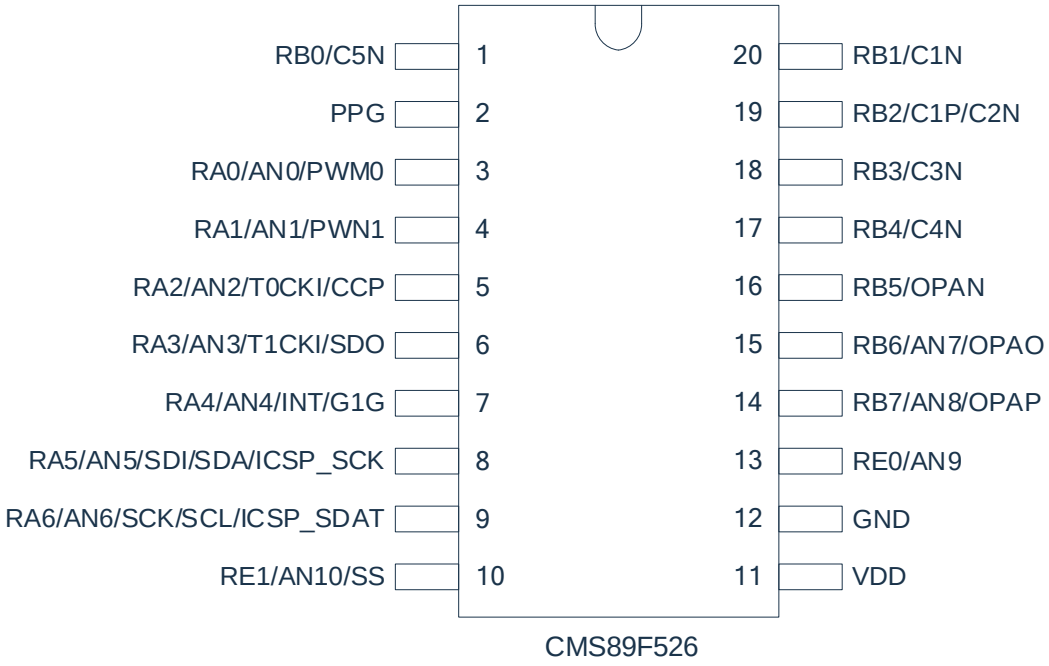


CMS89F5233

CMS89F5233 Pin Description:

Pin Name	IO Type	Pin Description
VDD, GND	P	Voltage input pin and ground
RA0-RA6	I/O	Programmable in/ push-pull out pin, with pull-up resistor function
RB1-RB2/ RB4-RB6	I/O	Programmable in/ push-pull out pin, with pull-up resistor function
RE0	I/O	Programmable in/ push-pull out pin, with pull-up resistor function
ICSP_SCK	I	Program clock input
ICSP_SDAT	I/O	Program data input/output
AN0-AN3/AN5-AN7/ AN9-AN10	I	AD Channel input pin
INT	I	External Interrupt input pin
T1G	I	TIMER1 gate control input pin
T0CKI/T1CKI	I	TIMER0/1 external clock input pin
CCP	I	Capture Mode input pin
SDI/SDA/SCK/SCL/SDO/SS	I/O	I^2C/SPI data/clock/control Pin
C1P	I	Synchronize comparator plus port input
C1N	I	Synchronize comparator minus port input
C2N	I	Over voltage comparator minus input
C4N	I	Current surge function comparator minus input pin
OPAN/OPAO	I/O	Op amp minus input/output pin
PPG	O	IGBT output control pin (Open Drain output)

1.3.2 CMS89F526 Pin Map Diagram



CMS89F526

CMS89F526 Pin description:

Pin Name	IO Type	Pin Description
VDD, GND	P	Voltage input pin and ground
RA0-RA6	I/O	Programmable in/ push-pull out pin, with pull-up resistor function
RB0-RB6	I/O	Programmable in/ push-pull out pin, with pull-up resistor function
RE0/RE1	I/O	Programmable in/ push-pull out pin, with pull-up resistor function
ICSP_SCK	I	Program clock input
ICSP_SDAT	I/O	Program data input/output
INT	I	External interrupt input
AN0-AN10	I	AD Channel input
T0CKI/T1CKI	I	TIMER0/1 External clock input
T1G	I	TIMER1 gate control input
CCP	I	Capture mode input
SDI/SDA/SCK/SCL/SDO/SS	I/O	I^2C/SPI data/clock/control pin
C1P	I	Synchronize comparator plus port input
C1N	I	Synchronize comparator minus port input
C2N	I	Over voltage comparator minus input
C3N	I	Selectable voltage surge function or over voltage function
C4N	I	Current surge function comparator minus input pin
C5N	I	Voltage surge function comparator minus input pin
OPAN/OPAO	I/O	Op amp minus input/output pin
PPG	O	IGBT output control pin (Open Drain output)

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 access or operate. It includes the following:

1. OSC (choice of oscillation)
   - INTRC Internal RC oscillation
2. WDT (watchdog selection)
   - 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 f is random.

1.5 Online Serial Programming

Can perform serial programming on MCU t 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.

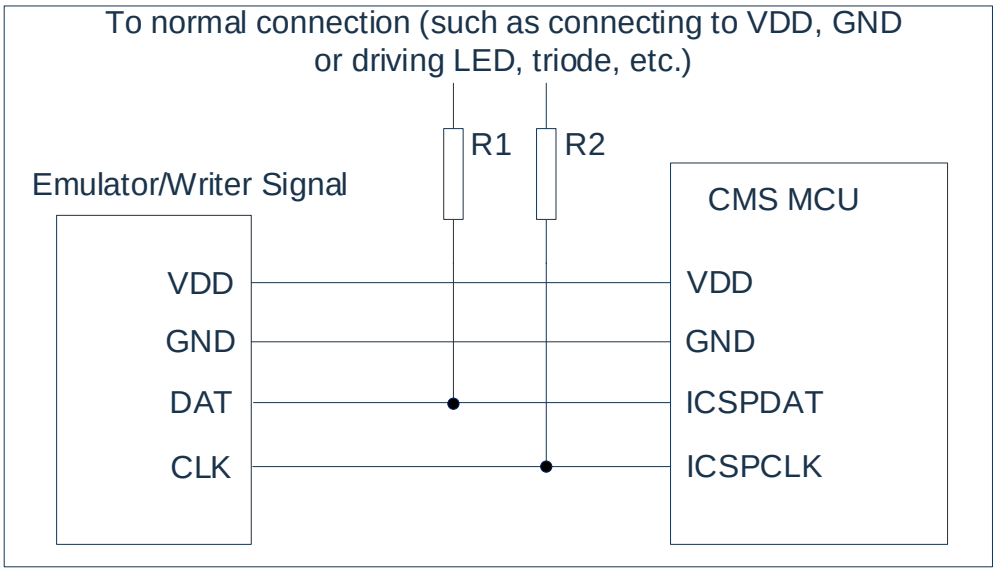


Fig 1-1: Typical connection for online serial programming

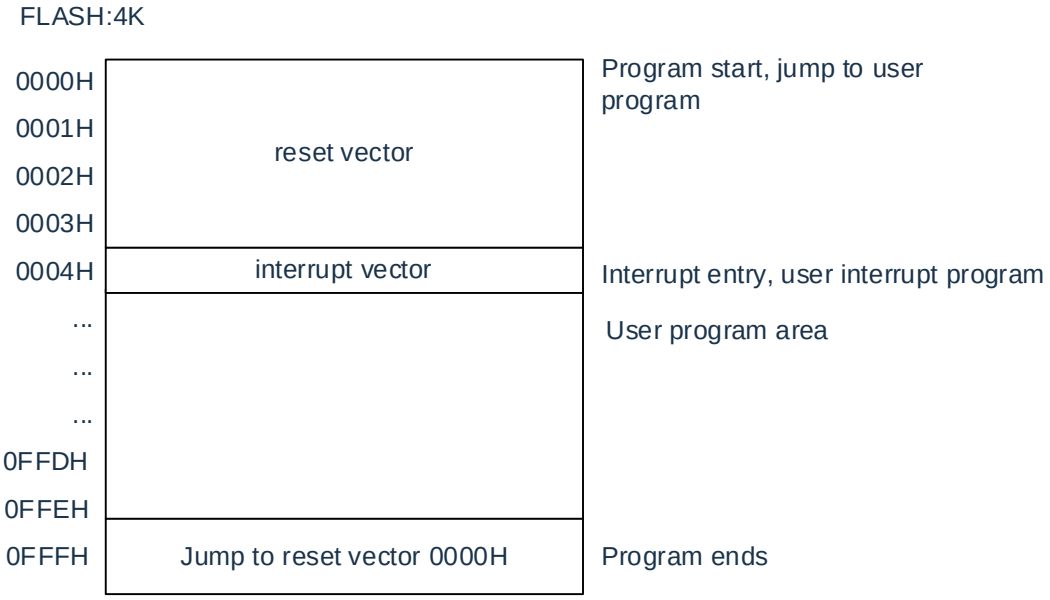
In the above figure, R1 and R2 are the electrical isolation devices, normally represented by resistor with the following resistance: R1≥4.7K、R2≥4.7K.

2. Central Processing Unit (CPU)

2.1 Memory

2.1.1 Program Memory

CMS89F5231/5232/5233/526 program memory space



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 interrupts 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
```

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
```

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 (110H) Read/write register to indicate higher 5 bits in the table.
- TABLE_SPL (111H) Read/write register to indicate lower 8 bits in the table.
- TABLE_DATAH (112H) 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.

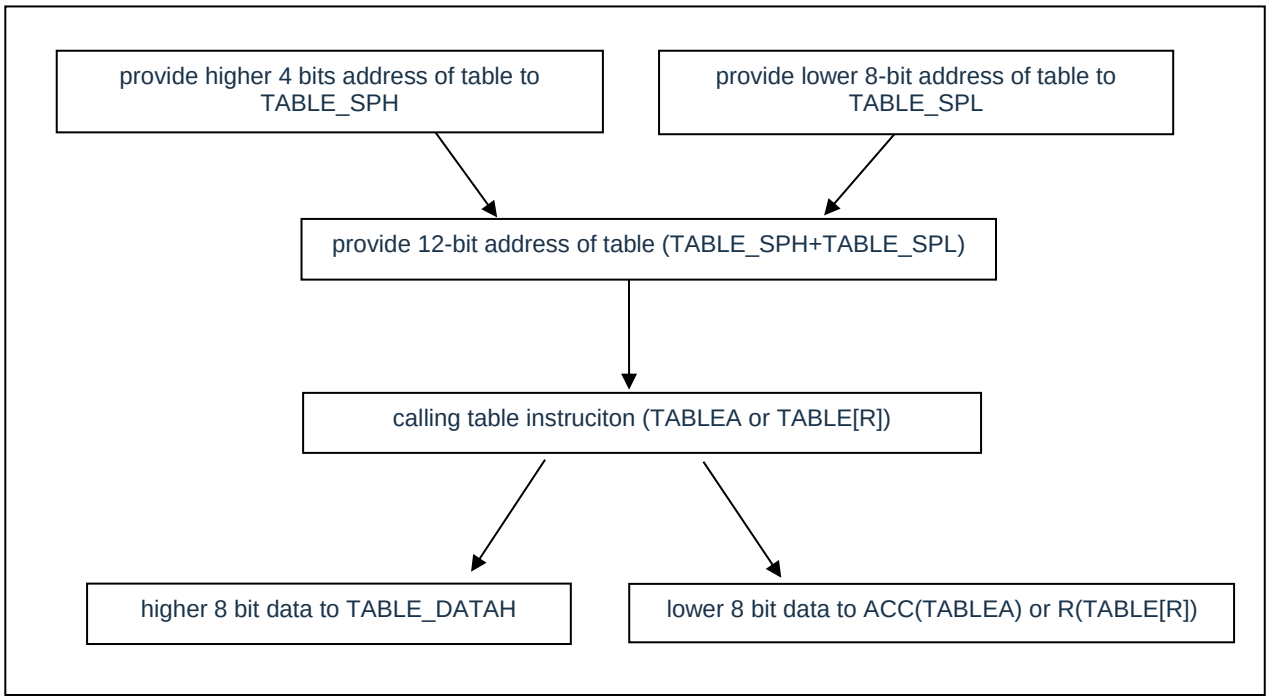


Fig 2-1: Flow chart for table usage

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 toLOOP1
00FEH:          JP          LOOP2           ; ACC=1, jump toLOOP2
00FFH:          JP          LOOP3           ; ACC=2, jump toLOOP3
0100H:          JP          LOOP4           ; ACC=3, jump to0000H address
0101H:          JP          LOOP5           ; ACC=4, jump to0001H address
0102H:          JP          LOOP6           ; ACC=5, jump to0002H address
```

Note: Since PCl 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 Register

2.1.2.1 CMS89F5231/5232/5233/526 Data Register List

	address		address		address		address
INDF	00h	INDF	80h	INDF	100h	INDF	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	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	WDTCON	105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
----	07h	----	87h	OPAADJ	107h	PAANSEL	187h
----	08h	----	88h	OPACON	108h	PBANSEL	188h
PORTE	09h	TRISE	89h	OPACON1	109h	PEANSEL	189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	----	8Eh	EEDATH	10Eh	CCPRL	18Eh
TMR1H	0Fh	OSCCON	8Fh	----	10Fh	CCPRH	18Fh
T1CON	10h	OSCTUNE	90h	TABLE_SPH	110h	CCPCON	190h
TMR2	11h	----	91h	TABLE_SPL	111h	SSPADD/SSPMSK	191h
T2CON	12h	PR2	92h	TABLE_DATAH	112h	SSPBUF	192h
----	13h	CM1CNT	93h	CM1ADJ	113h	SSPSTAT	193h
PPGTMRL	14h	WPUA	94h	CM2ADJ	114h	SSPCON	194h
PPGTMRH	15h	WPUB	95h	CM3ADJ	115h	SSPCON2	195h
PPGDLY	16h	WPUE	96h	CM4ADJ	116h	----	196h
PPGCON	17h	CM1CON	97h	CM5ADJ	117h	----	197h
PWM0DR	18h	CM2CON	98h	----	118h		198h
PWM0PR	19h	CM2CON1	99h	----	119h		199h
PWM0CR	1Ah	CM3CON	9Ah	----	11Ah		19Ah
PWM1DR	1Bh	CM3CON1	9Bh	----	11Bh		19Bh
PWM1PR	1Ch	CM4CON	9Ch	----	11Ch		19Ch
PWM1CR	1Dh	CM5CON	9Dh	----	11Dh		19Dh
----	1Eh	ADRESL	9Eh	----	11Eh		19Eh
ADCON0	1Fh	ADRESH	9Fh	----	11Fh	----	19Fh
Universal register 96 bytes	20h 6Fh	Universal register 80byte	A0h EFh	Universal register 80byte	120h 16Fh		1A0h 1EFh
	70h -- 7Fh	Fast memory space 70H-7FH	F0h -- FFh	Fast memory space 70H-7FH	170h -- 17Fh	Fast memory space 70H-7FH	1F0h -- 1FFh
BANK0		BANK1		BANK2		BANK3	

Data memory consists of 512×8 bits. It can be divided into to 2 function areas: special function register and general data registers. Most of data registers are able to write/read data, only some data memory are read-only. Special register address is from 00H-1FH, 80-9FH, 100-11FH, 180-19FH.

Summary of special registers in CMS89F52x Bank0

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
00h	INDF	Addressing this unit will use FSR content to (rather than physical register)								xxxx xxxx
01h	TMR0	TIMER0 data register								xxxx xxxx
02h	PCL	Lower bit of program counter								0000 0000
03h	STATUS	IRP	----	----	TO	PD	Z	DC	C	0--1 1xxx
04h	FSR	memory pointers for indirect addressing of data registers								xxxx xxxx
05h	PORTA	----	RA6	RA5	RA4	RA3	RA2	RA1	RA0	-xxx xxxx
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx
09h	PORTE	----	----	----	----	----	----	RE1	RE0	---- --xx
0Ah	PCLATH	----	---	----	Write buffer of higher 5 bits of program counter					---0 0000
0Bh	INTCON	GIE	PEIE	T0IE	INTE	----	T0IF	INTF	----	0000 -00-
0Ch	PIR1	EEIF	ADIF	SSPIF	BCLIF	CCPIF	----	TMR2IF	TMR1IF	0000 0-00
0Dh	PIR2	----	----	C5IF	C4IF	C3IF	C2IF	C1IF	PPGWDTIF	--00 0000
0Eh	TMR1L	Data register of 16-bits TIMER1 register lower bit								xxxx xxxx
0Fh	TMR1H	Data register of 16-bits TIMER1 register higher bit								xxxx xxxx
10h	T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	----	----	TMR1CS	TMR1ON	0000 --00
11h	TMR2	TIMER2 module register								0000 0000
12h	T2CON	----	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000
14h	PPGTMRL	PPGTMR low 8 bits								0000 0000
15h	PPGTMRH	----	----	----	----	----	----	PPGTMR high 2 bits		---- --00
16h	PPGDLY	----	----	----	----	PPGDLY				---- 0000
17h	PPGCON	DETC5F	DETC4F	----	RELOAD_EN	----	----	PPGMD	PPG_ON	11-0 --00
18h	PWM0DR	PWM0 duty cycle data register								xxxx xxxx
19h	PWM0PR	PWM0 period data register								xxxx xxxx
1Ah	PWM0CR	PWM0EN	PWM0MOD	----	PWM0POL	PWM0CKS [3: 0]				00-0 0000
1Bh	PWM1DR	PWM1 duty cycle data register								xxxx xxxx
1Ch	PWM1PR	PWM1 period data register								xxxx xxxx
1Dh	PWM1CR	PWM1EN	PWM1MOD	----	PWM1POL	PWM1CKS [3: 0]				00-0 0000
1Fh	ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON	0000 0000

Summary of special registers in CMS89F52x Bank1

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
80h	INDF	Addressing this unit will use FSR content to (rather than physical register)								xxxx xxxx
81h	OPTION_REG	----	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	-111 1111
82h	PCL	Lower bits of program counter								0000 0000
83h	STATUS	IRP	----	----	TO	PD	Z	DC	C	0—1 1xxx
84h	FSR	memory pointers for indirect addressing of data memory								xxxx xxxx
85h	TRISA	----	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	-111 1111
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111
89h	TRISE	----	----	----	----	----	----	TRISE1	TRISE0	---- --11
8Ah	PCLATH	----	----	----	Write buffer of higher 5 bits of program counter					---0 0000
8Bh	INTCON	GIE	PEIE	T01E	INTE	----	T0IF	INTF	----	0000 -00-
8Ch	PIE1	EEIE	ADIE	SSPIE	BCLIE	CCPIE	----	TMR2IE	TMR1IE	0000 0-00
8Dh	PIE2	----	----	C5IE	C4IE	C3IE	C2IE	C1IE	PPGWDTIE	--00 0000
8Fh	OSCCON	----	IRCF2	IRCF1	IRCF0	----	----	----	----	-110 ----
90h	OSCTUNE	----	----	----	TUN4	TUN3	TUN2	TUN1	TUN0	---0 0000
92h	PR2	TIMER2 period register								1111 1111
93h	CM1CNT	CM1OF	CM1CNT [6: 0]							0000 0000
94h	WPUA	----	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0	-000 0000
95h	WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0	0000 0000
96h	WPUE	----	----	----	----	----	----	WPUE1	WPUE0	---- --00
97h	CM1CON	CM1EN	CM1COFM	CM1CEN	CM1CLR	CM1NSL	----	----	----	0000 0---
98h	CM2CON	CM2EN	CM2COFM	CM2DBSEL [1: 0]		CM2PVSL [3: 0]				0000 0000
99h	CM2CON1	ATPEN	----	----	----	CM2COF	CM2COS [2: 0]			0--- 0000
9Ah	CM3CON	CM3EN	CM3COFM	CM3DBSEL [1: 0]		CM3PVSL [3: 0]				0000 0000
9Bh	CM3CON1	CM3M1	CM3M0	----	CM3CIS	CM3COF	CM3COS [2: 0]			00-0 0000
9Ch	CM4CON	CM4EN	CM4COFM	CM4DBSEL [1: 0]		CM4PVSL [3: 0]				0000 0000
9Dh	CM5CON	CM5EN	CM5COFM	CM5DBSEL [1: 0]		CM5PVSL [3: 0]				0000 0000
9Eh	ADRESL	A/D result register lower 8 bits								xxxx xxxx
9Fh	ADRESH	A/D result register higher 2 bits								---- --xx

Summary of special registers in CMS89F52x Bank2

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
100h	INDF	Addressing this unit will use FSR content to (rather than physical register)								xxxx xxxx
101h	TMR0	TIMER0 mode register								xxxx xxxx
102h	PCL	Lower bit of program counter (PC)								0000 0000
103h	STATUS	IRP	----	----	TO	PD	Z	DC	C	0--1 1xxx
104h	FSR	memory pointers for indirect addressing of data memory								xxxx xxxx
105h	WDTCON	----	----	----	----	----	----	----	SWDTEN	---- ---0
106h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx
107h	OPAADJ	OPAOUT	OPARS	--	OPAADJS [4: 0]					00-1 0000
108h	OPACON	OPAEN	OPAFM	OPAFE	----	OPAPS1	OPAPS0	OPANS1	OPANS0	001- 0000
109h	OPACON1	----	----	----	----	OPO2ADE	-----	ANRS1	ANRS0	---- 0-00
10Ah	PCLATH	----	----	----	Write buffer of higher 5 bits of program counter					---0 0000
10Bh	INTCON	GIE	PEIE	T01E	INTE	----	T0IF	INTF	----	0000 -00-
10Ch	EEDATA	EEDAT7	EEDAT6	EEDAT5	EEDAT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0	0000 0000
10Dh	EEADR	----	----	----	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0	---0 0000
10Eh	EEDATH	EEDATH7	EEDATH6	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0	--00 0000
110h	TABLE_SPH	Table high bits pointer								---x xxxx
111h	TABLE_SPL	Table lower bits pointer								xxxx xxxx
112h	TABLE_DATAH	Table higher bits data								xxxx xxxx
113h	CM1ADJ	CM1OUT	CM1CRS	CM1ADJ [5: 0]						0010 0000
114h	CM2ADJ	CM2OUT	CM2CRS	CM2ADJ [5: 0]						0010 0000
115h	CM3ADJ	CM3OUT	CM3CRS	CM3ADJ [5: 0]						0010 0000
116h	CM4ADJ	CM4OUT	CM4CRS	CM4ADJ [5: 0]						0010 0000
117h	CM5ADJ	CM5OUT	CM5CRS	CM5ADJ [5: 0]						0010 0000

Summary of special registers in CMS89F52x Bank3

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Reset value
180h	INDF	Addressing this unit will use FSR content to (rather than physical register)								xxxx xxxx
181h	OPTION_REG	----	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111
182h	PCL	Lower bits of program counter (PC)								0000 0000
183h	STATUS	IRP	----	----	TO	PD	Z	DC	C	0001 1xxx
184h	FSR	memory pointers for indirect addressing of data memory								xxxx xxxx
186h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB73	TRISB2	TRISB1	TRISB0	1111 1111
187h	PAANSEL	----	PAANS6	PAANS5	PAANS4	PAANS3	PAANS2	PAANS1	PAANS0	-000 0000
188h	PBANSEL	PBANS7	PBANS6	PBANS5	PBANS4	PBANS3	PBANS2	PBANS1	PBANS0	0000 0000
189h	PEANSEL	----	----	----	----	----	----	PEANS1	PEANS0	---- --00
18Ah	PCLATH	----	----	----	Write buffer of higher 5 bits of program counter					---0 0000
18Bh	INTCON	GIE	PEIE	T01E	INTE	----	T01F	INTF	----	0000 0000
18Ch	EECON1	EEPGD	----	----	----	WRERR	WREN	WR	RD	0--- x000
18Dh	EECON2	EEPROM control register 2 (not physical register)								---- ----
18Eh	CCPRL	Capture register lower bits								xxxx xxxx
18Fh	CCPRH	Capture register higher bits								xxxx xxxx
190h	CCPCON	CCPEN	----	----	CCPIS	CCPES	CPTM2	CPTM1	CPTM0	0--00000
191H	SSPMSK	MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0	11111111
191h	SSPADD	Synchronize Serial interface (I^2C mode) address register								0000 0000
192h	SSPBUF	Synchronize Serial interface receiver buffer / transmit buffer								xxxx xxxx
193h	SSPSTAT	SMP	CKE	D/A	P	S	R/W	UA	BF	0000 0000
194h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000
195h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000

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 (9th 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 9th 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.

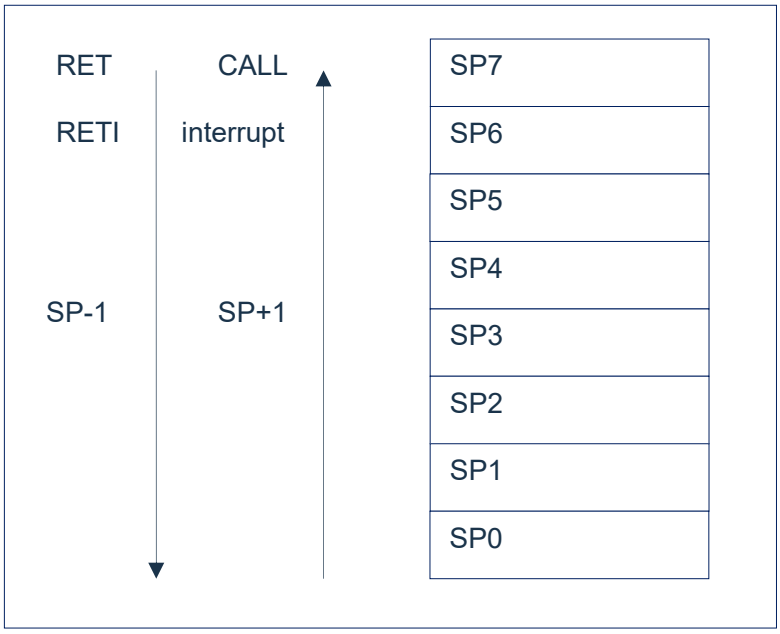


Fig 2-2: stack buffer working principle

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 cannot 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 an 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	Reserved	
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 carry happens to higher bits in Lower 4 bits of the result.
	0=	When no carry happens to higher bits happens in Lower 4 bits of the result.
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 Wake up MCU
0	1	WDT overflow non-sleep status
1	0	Key Press to Wake up MCU in sleep mode
1	1	Power On

TO/PD status after reset

2.6 Pre-scaler (OPTION_REG)

OPTION_REG register can be read or written. Including all types of control bit for configuration:

- TIMER0/WDT pre-scaler
- TIMER0

pre-scaler OPTION_REG (181H)

181H	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	1	1	1

Bit7 Reserved:

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 (Fosc/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 module
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 an 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.

```
CLR     TMR0            ; clear TMR0
CLRWDT                  ; clear WDT
LDIA    B'00xx1111'     ; essential step, must be included
LD      OPTION_REG, A   ; essential step, must be included
LDIA    B'00xx1xxx'     ; config new pre-scaler
LD      OPTION_REG, A
```

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

```
CLRWDT                  ; clear WDT
LDIA    B'00xx0xxx'     ; 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 address 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 returns 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 12 Bit, user can access lower 8 bits through PCL (02H). The higher 4 bits cannot be accessed. It can hold address for 4K×16Bit 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+1will be added to stack automatically);
CALL	PC=program defined address (original PC+1will be added to stack automatically);
RET、RETI、RET i	PC=value coming out from stack;
Operating on PCL	PC [11: 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, the overflow period of WDT is 18ms. The way to calculate WDT overflow is 18ms*pre-scaling parameter. If WDT 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 is 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 have 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 (105H)

105H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WDTCON	---	---	---	---	---	---	---	SWDTEN
R/W	---	---	---	---	---	---	---	R/W
Reset value	---	---	---	---	---	---	---	0

Bit7~Bit1 Not used, read as 0
Bit5~Bit5 Reserved (do not use)
Bit0 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 Overview

Clock signal is generated by internal oscillation, system clock (Fsys) is generated via CONFIG pre-scaling and register pre-scaling, 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.

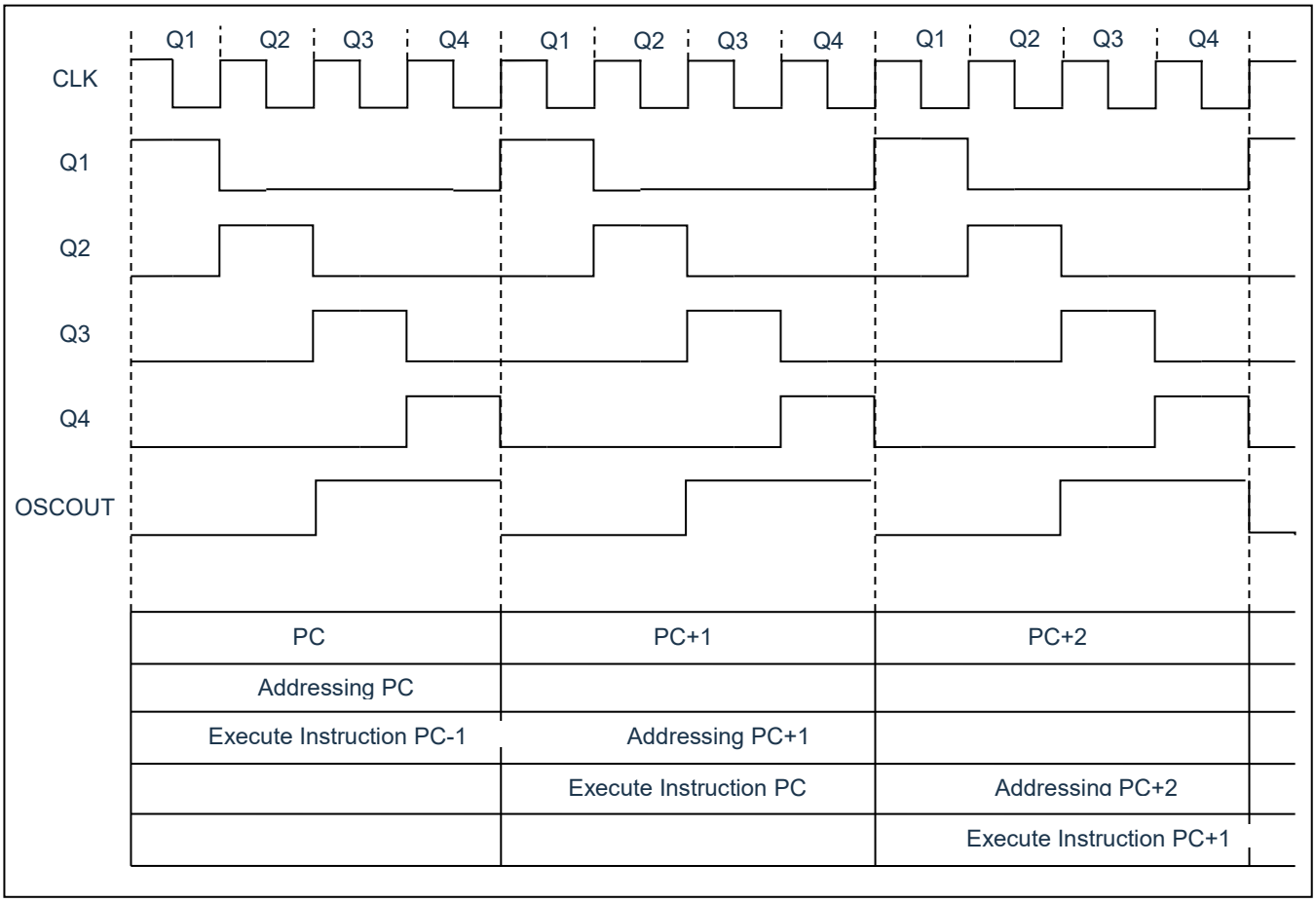


Fig 3-1: time series for clock and instruction period

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
16Mhz	500ns	250ns

3.2 System Oscillator

Chip has only 1 way of oscillation, internal RC oscillation.

3.2.1 Internal RC Oscillation

Default oscillation is internal RC oscillation. Its frequency Fosc is 8M/16M, which is set by OSCCON register. The oscillation frequency is calibrated while shipping, the deviation controlled within±3%.

3.3 Reset Time

Reset Time is the time for chip to change from reset to stable oscillation. The value is about 18ms@5V.

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. Oscillator tune register OSCTUNE can tune the frequency of internal oscillation in the software.

OSCCON (8FH)

8FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OSCCON	---	IRCF2	IRCF1	IRCF0	---	---	---	SCS
R/W	---	R/W	R/W	R/W	---	---	---	R/W
Reset value	---	1	1	0	---	---	---	0

Bit7 Not used, read 0

Bit6~Bit4 IRCF<2: 0>: Selection bit for frequency division of Internal oscillator

111= 8MHz
110= 4MHz (default)
101= 2MHz
100= 1MHz
011= 500KHz
010= 250KHz
001= 125KHz
000= 31KHz (LFINTOSC)

Bit3~Bit0 Reserved

Oscillator tuning register OSCTUNE (90H)

90H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OSCTUNE	---	---	---	TUN4	TUN3	TUN2	TUN1	TUN0
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 TUN<4: 0>: Frequency tuning bit

01111= Highest frequency
01110=
.
.
.
00001=
00000= Oscillators operate at the factory calibrated frequency
11111=
.
.
.
10000= Lowest frequency

Note: F_{OSC} as internal oscillator has frequency of 8MHz/16MHz; F_{SYS} is the working frequency of the system.

4. Reset

Chip has 4 ways of reset:

- power on reset.
- low voltage reset.
- watchdog overflow reset under normal working condition.
- Watchdog overflow reset under sleep mode.

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 Overview

Power off reset is used for voltage drop caused by external factors (such as interference or change in external load)

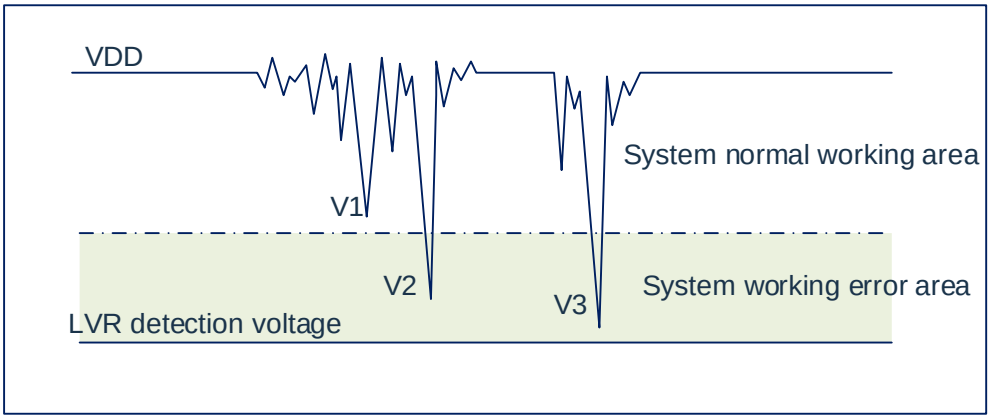


Fig 4-1: power off reset

The above is a typical power off reset case. VDD 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 VDD drops to V1, system still works normally. When VDD 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. VDD 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, VDD 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:

- Turn on low voltage detection function of MCU.
- Turn on watchdog timer.
- Lower working frequency of the system.
- Increase the gradient of the voltage drop.

Turn on Low voltage detection function of MCU

Chip has built-in low voltage detection (LVR) function, it can be controlled by programmed CONFIG, refer to Chapter 1.5 regarding programming CONFIG selection description. When LVR function is turned on, when system voltage drops lower than LVR voltage, LVR will be triggered, system will be reset.

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 Power off 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.
- Turn off oscillator driver device.
- 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 VDD 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 Wake Up from Sleep Mode

Wake up through any of the following events:

1. Watchdog timer awake (WDT force enable)
2. Peripheral's interrupt

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 Wake up happens.

When executes STOP instructions, next instruction (PC+1)is withdrawed first. If it is intended to Wake up the system using interrupt, the corresponding enable bit should be set to 1 for the interrupt. Wake up 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 waked up 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 waked up from sleep mode. STOP will be executed before system being fully awaken. 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 input port has a fixed status and avoid increasing sleep current when I/O is input; turn off AD and other peripherals module; 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
        LD      TRISE, 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 Wake Up time

When MCU is Wake up from sleep mode, oscillation stabilization requires certain wait time (reset time). The reference level of this time is 18ms.

6. I/O Port

Chip has 3 I/O port: PORTA、PORTB、PORTE (max. of 17 I/O). read/write port data register can directly read/write these ports.

port	bit	Pin description	I/O
PROTA	0	Schmitt trigger input, push-pull output, internal weak pull-up, AN0, PWM0	I/O
	1	Schmitt trigger input, push-pull output, internal weak pull-up, AN1, PWM1	I/O
	2	Schmitt trigger input, push-pull output, internal weak pull-up, AN2, TMR0 clock input, CCP input	I/O
	3	Schmitt trigger input, push-pull output, internal weak pull-up, AN3, TMR1 clock input, SPI Data Output port	I/O
	4	Schmitt trigger input, push-pull output, internal weak pull-up, AN4, external interrupt input, TMR1 gate control input	I/O
	5	Schmitt trigger input, push-pull output, internal weak pull-up, AN5, on-line programming and emulation data port, I^2C communication data port	I/O
	6	Schmitt trigger input, push-pull output, internal weak pull-up, AN6, on-line programming and emulation clock port, I^2C communication clock port	I/O
PORTB	0	Schmitt trigger input, push-pull output, internal weak pull-up, Voltage surge comparison negative input	I/O
	1	Schmitt trigger input, push-pull output, internal weak pull-up, synchronize comparator negative input	I/O
	2	Schmitt trigger input, push-pull output, internal weak pull-up, synchronize comparator positive input, over voltage comparator negative input	I/O
	3	Schmitt trigger input, push-pull output, internal weak pull-up, Selectable voltage surge or over voltage comparator negative input	I/O
	4	Schmitt trigger input, push-pull output, internal weak pull-up, Current surge comparison negative input	I/O
	5	Schmitt trigger input, push-pull output, internal weak pull-up, Op Amp negative input	I/O
	6	Schmitt trigger input, push-pull output, internal weak pull-up, AN7, Op Amp Output	I/O
	7	Schmitt trigger input, push-pull output, internal weak pull-up, AN8, Op Amp positive input	I/O
PORTE	0	Schmitt trigger input, push-pull output, internal weak pull-up, AN9	I/O
	1	Schmitt trigger input, push-pull output, internal weak pull-up, AN10	I/O

< Table 6-1: port configuration summary >

6.1 I/O Port Structure

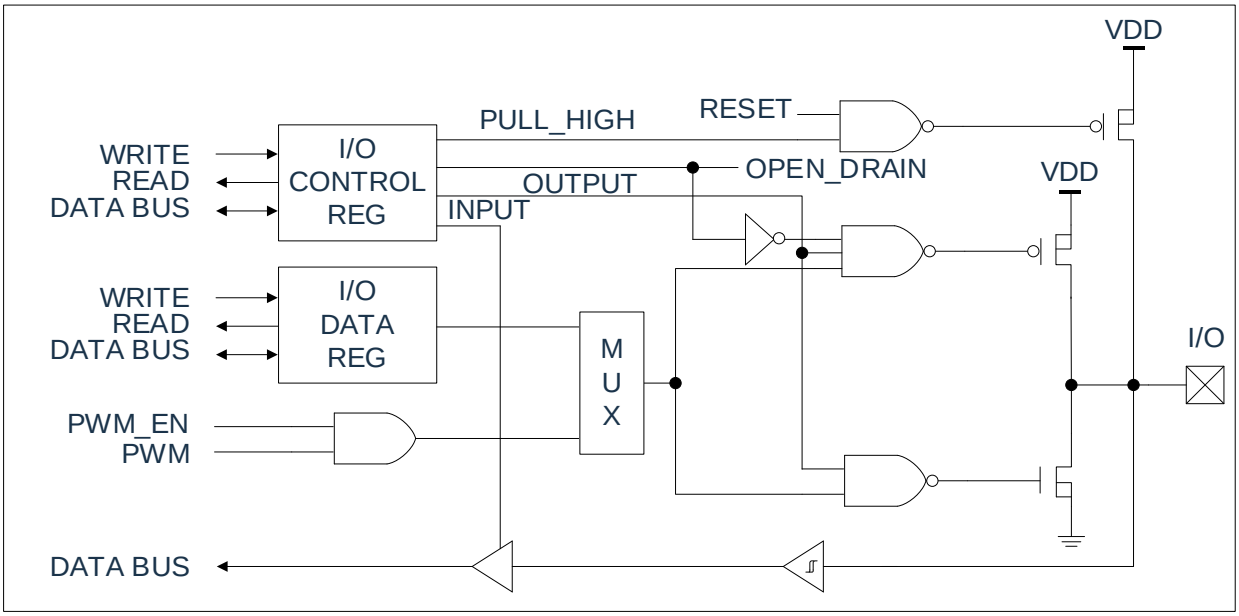


Fig 6-1: I/O port structure (1)

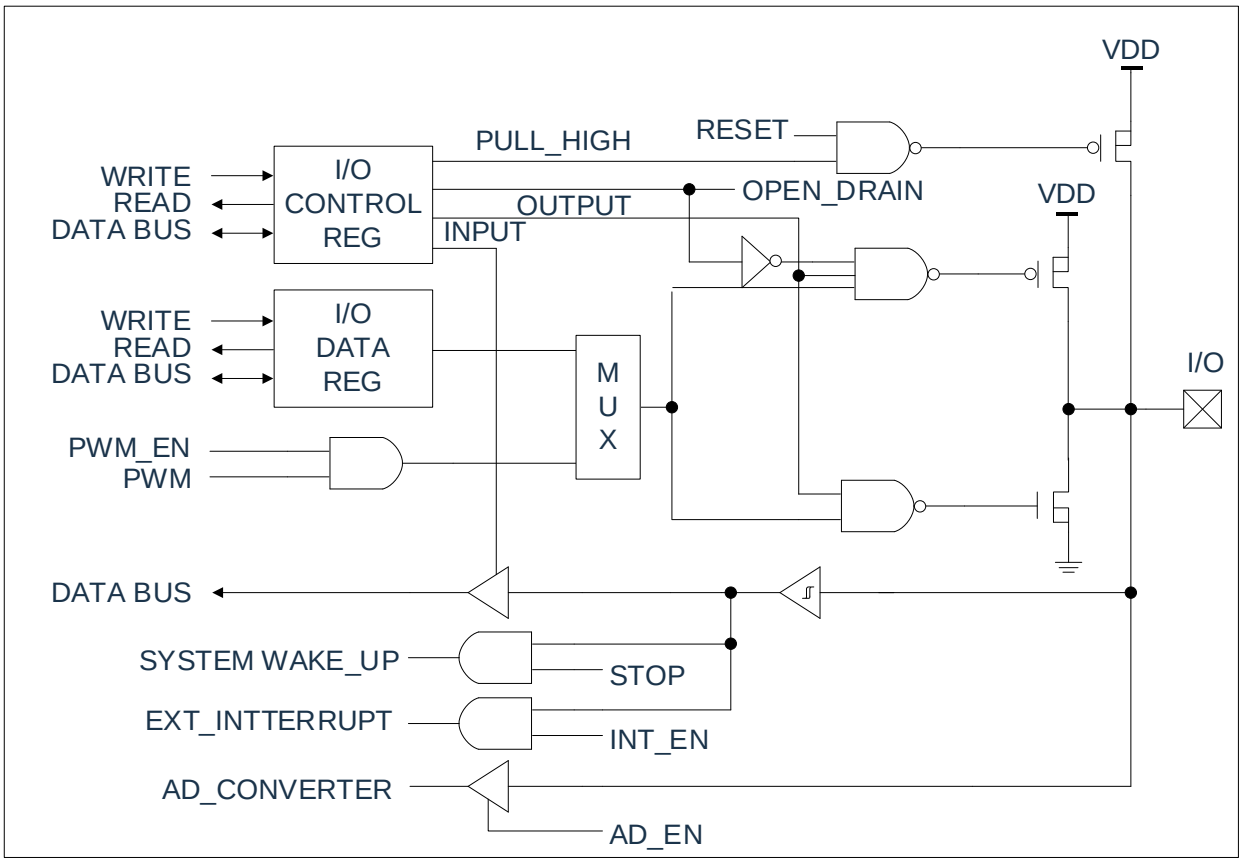


Fig 6-2: I/O port structure (2)

6.2 PORTA

6.2.1 PORTA Data and Direction Control

PORTA is 7 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 register 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. The IO pins which configured as analog input are always read as 0.

Note: Must initialize ANSEL register to configure analog channel to digital input. The IO pins which configured as analog input are always read as 0.

Registers related to PORTA ports are PORTA、TRISA、WPUA、PAANSEL … etc.

PORTA data register PORTA (05H)

05H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTA	---	RA6	RA5	RA4	RA3	RA2	RA1	RA0
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

Bit6~Bit0 PORTA<6: 0>: PORTAI/O pin bit;
1= Port pin voltage level>V_{IH};
0= Port pin voltage level<V_{IL}.

PORTA direction register TRISA (85H)

85H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISA	---	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
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

Bit6~Bit0 TRISA<6: 0>: PORTA Tri-state control bits;
1= PORTA pin set to be input (tri-state);
0= PORTA pin set to be output.

example: procedure for PORTA

```
CLR     PAANSEL     ; Configure all PORTA port to be digital IO ports
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 Analog Control Selection

The PAANSEL register is used to configure the input mode of I/O pin to analog mode. Setting the appropriate bit in PAANSEL 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 PAANSEL bit has no effect on the digital output function. The pin with TRIS cleared and PAANSEL 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 PAANSEL (187H)

187H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PAANSEL	---	PAANS6	PAANS5	PAANS4	PAANS3	PAANS2	PAANS1	PAANS0
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

Bit6~Bit0 PAANS <6: 0>: Analog selection bit, select the digital or analog function of pin PORTA<6: 0>

1= Analog input, pin is allocated to analog input.

0= Digital I/O, pin is allocated to port or special function.

6.2.3 PORTA Pull up Resistor

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 (94H)

107H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUA	---	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
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

Bit6~Bit0 WPUA<6: 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

6.3 PORTB

6.3.1 PORTB Data and Direction

PORTB is an 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. The IO pins which configured as analog input are always read as 0.

Related registers with PORTB port include PORTB, TRISB, PBANSEL, WPUB … 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
1= Port pin level >V_{IH};
0= Port pin level<V_{IL}

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 (tri-state)
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 Analog Selection Control

The PBANSEL register is used to configure the input mode of I/O pin to analog mode. Setting the appropriate bit in PBANSEL 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 PBANSEL bit has no effect on the digital output function. The pin whose TRIS is cleared and PBANSEL 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 PBANSEL (188H)

188H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PBANSEL	PBANS7	PBANS6	PBANS5	PBANS4	PBANS3	PBANS2	PBANS1	PBANS0
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 PBANS<15: 8>: Analog selection bits, select the analog or digital functions of pin PORTB<7: 0>.
1= analog input, pin is allocated as analog input.
0= Digital I/O, pin is allocated to port or special function.

6.3.3 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 disabled.

PORTB pull up resistance register WPUB (95H)

95H	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<6: 0>: Weak pull up register bit
1= Enable pull up
0= Disable pull up

Note: if Pin is configured as output, the weak pull up will be automatically disabled.

6.4 PORTE

6.4.1 PORTE Data and Direction

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

Reading the PORTE 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.

PORTE data register PORTE (09H)

09H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTE	---	---	---	---	---	---	RE1	RE0
R/W	---	---	---	---	---	---	R/W	R/W
Reset value	---	---	---	---	---	---	X	X

Bit1~Bit0 PORTD<2: 0>: PORTD I/O pin bit
1= Port output high level;
0= Port output low level

PORTE direction register TRISE (89H)

89H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISE	---	---	---	---	---	---	TRISE1	TRISE0
R/W	---	---	---	---	---	---	R/W	R/W
Reset value	---	---	---	---	---	---	1	1

Bit1~Bit0 TRISD<1: 0>: Control bit of PORTE tri-state
1= PORTE pin configured as input (tri-state)
0= PORTE pin configured as output

6.4.2 PORTE Pull up Resistance

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

PORTE pull up resistance register WPUE (96H)

96H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUD	---	---	---	---	---	---	WPUE1	WPUE0
R/W	---	---	---	---	---	---	R/W	R/W
Reset value	---	---	---	---	---	---	0	0

Bit1~Bit0 WPUE<1: 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 PORTE Analog selection

The PEANSEL register is used to configure the input mode of I/O pin to analog mode. Setting the appropriate bit in PEANSEL 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 PEANSEL bit has no effect on the digital output function. The pin whose TRIS is cleared and PEANSEL 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.

PORTE analog selection register PEANSEL (189H)

189H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PEANSEL	---	---	---	---	---	---	PEANS1	PEANS0
R/W	---	---	---	---	---	---	R/W	R/W
Reset value	---	---	---	---	---	---	0	0

Bit1~Bit0 PEANS<1: 0> Analog selection bits, select the analog or digital functions of pin PORTE<1: 0>.
1= analog input, pin is allocated as analog input.
0= Digital I/O, pin is allocated to port or special function.

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     PORTA       ; clear PORTA
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.

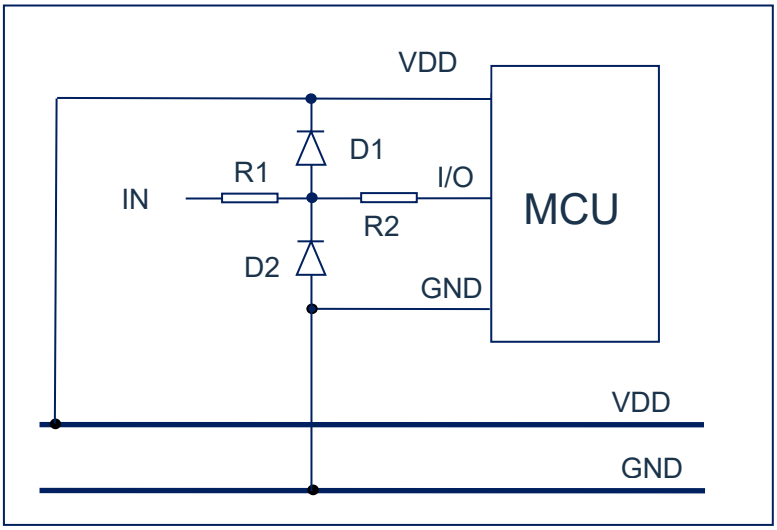


Fig 6-3: The input voltage is not within the specified range

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
- TIMER1 overflow interrupt
- TIMER2 match interrupt
- INT interrupt
- AD interrupt
- CCP interrupt
- Comparator interrupt
- PPGWDT overflow interrupt
- MSSP interrupt
- EEPROM write 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 enables 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.

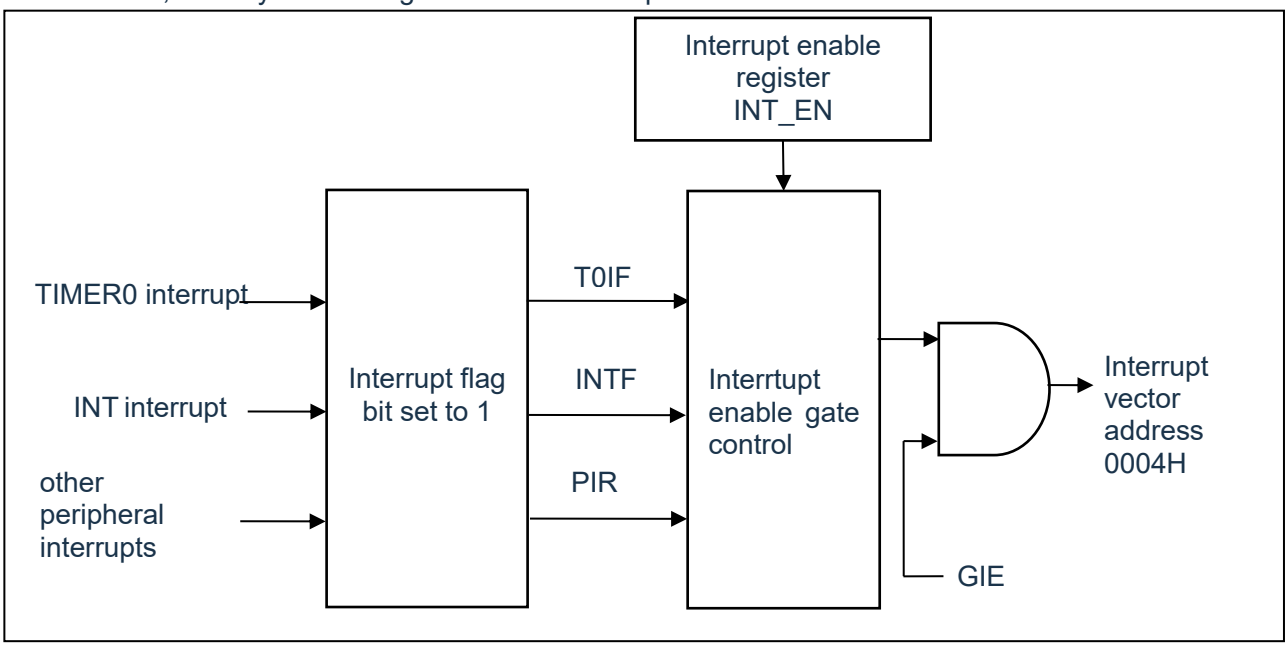


Fig 7-1: interrupt concept diagram

7.2 Interrupt Control Register

7.2.1 Interrupt Control Register

The interrupt control register INTCON is a readable and writable register, including the allowable and flag bits for TMR0 register overflow and PORTB port level change 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	---	T0IF	INTF	---
R/W	R/W	R/W	R/W	R/W	---	R/W	R/W	---
reset 值	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 Not used:
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 does not happen
Bit0 Not used:

Note: The T0IF bit is set as 1when 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 (8CH)

8CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE1	EEIE	ADIE	SSPIE	BCLIE	CCPIE	---	TMR2IE	TMR1IE
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 EEIE: EEPROM write operation interrupt enable bit
1= Enable EEPROM write operation interrupt
0= Disable EEPROM write operation interrupt

Bit6 ADIE: A/D converter (ADC)interrupt enable bit;
1= enable ADC interrupt;
0= disable ADC interrupt

Bit5 SSPIE: Primary synchronize serial port (MSSP) interrupt enable bit;
1= enable MSSP interrupt;
0= disable MSSP interrupt.

Bit4 BCLIE: Bus collision interrupt enable bit;
1= enable Bus collision interrupt;
0= disable Bus collision interrupt.

Bit3 CCPIE: CCP interrupt enable bit;
1= enable CCP interrupt;
0= disable CCP interrupt.

Bit2 Not used.

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 registerPIE2 (8DH)

8DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE2	---	---	C5IE	C4IE	C3IE	C2IE	C1IE	PPGWDTIE
R/W	---	---	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 C5IE: Comparator C5 interrupt enable bit;
1= enable Comparator C5 interrupt;
0= disable Comparator C5 interrupt.

Bit4 C4IE: Comparator C4 interrupt enable bit;
1= enable Comparator C4 interrupt;
0= disable Comparator C4 interrupt.

Bit3 C3IE: Comparator C3 interrupt enable bit;
1= enable Comparator C3 interrupt;
0= disable Comparator C3 interrupt.

Bit2 C2IE: Comparator C2 interrupt enable bit;
1= enable Comparator C2 interrupt;
0= disable Comparator C2 interrupt.

Bit1 C1IE: Comparator C1 interrupt enable bit;
1= enable Comparator C1 interrupt;
0= disable Comparator C1 interrupt.

Bit0 PPGWDTIE: PPGWDT overflow interrupt enable bit;
1= enable PPGWDT overflow interrupt;
0= disable PPGWDT overflow interrupt.

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	EEIF	ADIF	SSPIF	BCLIF	CCPIF	---	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 EEIF: EE Write operation interrupt flag bit;
1= Write operation completed (must clean to zero by software);
0= Write operation not completed or not yet started.

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 SSPIF: Main synchronous serial port (MSSP)interrupt flag bit.
1= The MSSP interrupt condition is met. Before returning from the interrupt service program, it must clear through software. The conditions for making this bit 1 are:
- SPI.
- transmit/receive happens.
- I^2C slave/master control.
- transmit/ receive happens.
- I^2C master control.
- The start condition that occurs is done by MSSP module.
- The stop condition that occurs is completed by MSSP module.
- The restart condition that occurs is done by MSSP module.
- The ack condition that occurs is done by MSSP module.
- The start condition occurs when the MSSP module is idle (multi-host system).
- The stop condition occurs when the MSSP module is idle (multi-host system);).

0= No MSSP interrupt condition is met.

Bit4 BCLIF: Bus collision interrupt flag bit;
1= When as I^2C master mode, bus collision occurs in MSSP;
0= bus collision not occurred.

Bit3 CCP1IF: CCP1 interrupt flag bit.
Capture mode:
1= Capture for TMR1 register happens (must clear through software);
0= Capture for TMR1 register not happen
Compare mode:
1= Compare match for TMR1 register happens (must clear through software);
0= Compare match for TMR1 register not happen.
PWM mode: Not used under this mode.

Bit2 reserved

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 registerPIR2 (0DH)

0DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIR2	---	---	C5IF	C4IF	C3IF	C2IF	C1IF	PPGWDTIF
R/W	---	---	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 C5IE: Comparator C5 interrupt flag bit;
1= Comparator output (C5OUT bit) changed (must be cleaned by SW)
0= Comparator output (C5OUT bit) not changed

Bit4 C4IE: Comparator C4 interrupt flag bit;
1= Comparator output (C4OUT bit) changed (must be cleaned by SW)
0= Comparator output (C4OUT bit) not changed

Bit3 C3IE: Comparator C3 interrupt flag bit;
1= Comparator output (C3OUT bit) changed (must be cleaned by SW)
0= Comparator output (C3OUT bit) not changed

Bit2 C2IE: Comparator C2 interrupt flag bit;
1= Comparator output (C2OUT bit) changed (must be cleaned by SW)
0= Comparator output (C2OUT bit) not changed

Bit1 C1IE: Comparator C1 interrupt flag bit;
1= Comparator output (C1OUT bit) changed (must be cleaned by SW)
0= Comparator output (C1OUT bit) not changed

Bit0 PPGWDTIE: PPGWDT overflow interrupt flag bit;
1= PPGWDT overflows (must be cleaned by SW)
0= PPGWDT not overflow

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 enables 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.

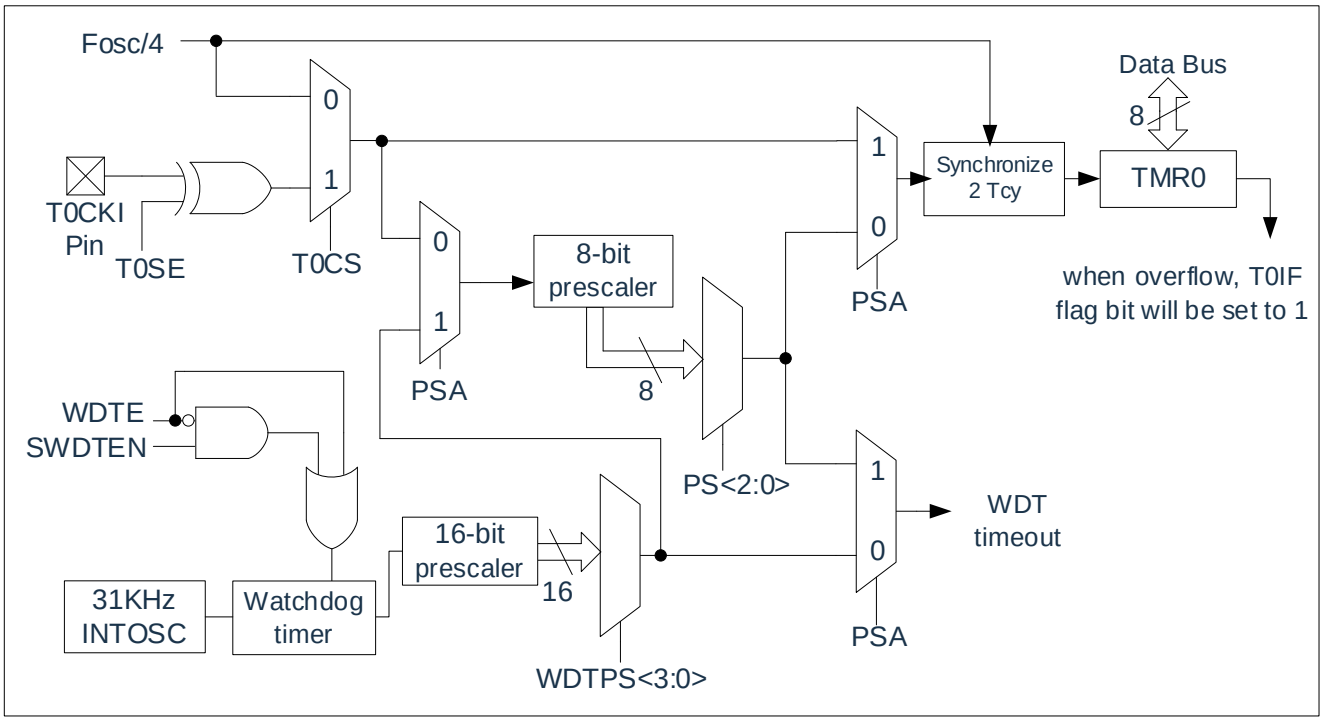


Fig 8-1: TIMER0/WDT module structure

Note:

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

8.2 Working Principle for TIMER0

The TIMER0 module 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 module 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 Each instruction period 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 module 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 module have a 1: 1 prescaler, the pre-scaler must be assigned to the WDT module.

The pre-scaler is not readable and writable. When the pre-scaler is assigned to the TIMER0 module, 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 Pre-scaler Switching Between TIMER0 and WDT Mode

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 module, the following instructions must be executed sequence.

Modify pre-scaler (TMR0-WDT)

```
        CLRWDT
        CLR         TMR0

        SETB        OPTION_REG, PSA     ; Select WDT
        CLRWDT

        LDIA        B'11111000'
        ANDA        OPTION_REG          ; Lower 3 bits set to 0
        ORIA        B'00000101'         ; Lower 3 bits set to 101, other bits remain
        LD          OPTION_REG, A
```

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

Modify pre-scaler (WDT-TMR0)

```
        CLRWDT
        
        LDIA    B'11110000'
        ANDA    OPTION_REG          ; lower 4 bits set to 0
        ORIA    B'00000101'         ; lower 4 bits set to 0101, other bits remain
        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 TMR0, 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 TMR0, etc. Please refer to 2.6 about the application of 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 (181H)

181H	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	1	1	1

Bit7 Reserved:

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= Interna instruction period clock (Fosc/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 module.

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 general

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

- 16-bit timer/counter register (TMR1H: TMR1L)
- 3-bit pre-scaler
- Wake up when overflow (external clock asynchronous mode only)
- Special event trigger function (with ECCP)
- via T1G pin gate control TIMER1 (enable counting)
- overflow interrupt
- Time base with capture/compare function

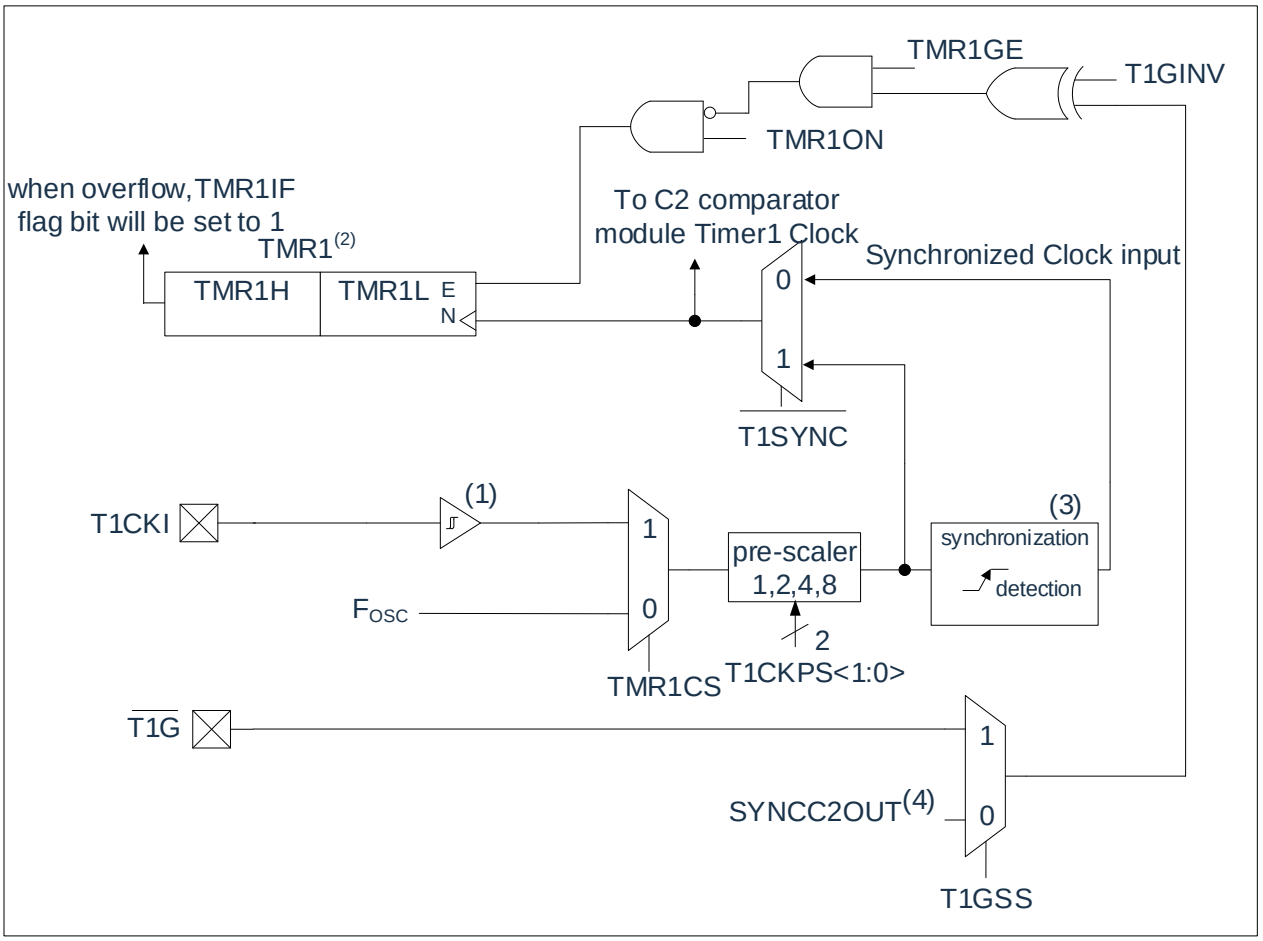


Fig 9-1: TIMER1 structure

Note:

1) The ST buffer is in low power mode when using the LP oscillator, but in high-speed mode when using T1CKI.
2) The Timer1 register increments on the rising edge.
3) Do not perform synchronous during sleep.
4) When CM2CON1 register C2SYNC bit set to 1, SYNCC2OUT synchronized.

9.2 Operation principle for TIMER1

TIMER1 module is a 16-bit incremental counter accessed through a pair of register TMR1H: TMR1L. TMR1L will only be written into internal buffer register, writing into TMR1H will load the internal buffer register into TIMER1 counter, therefore while performing write operation to TMR1L and TMR1H, must write TMR1L first, then write TMR1H register.

While TIMER1 is operating, the TMR1H: TMR1L register will increase in frequency with a multiple of Fosc. The specific multiple is determined by the TIMER1 pre-scaler.

9.3 TIMER1 pre-scaler

TIMER1 has four selections of prescaler ratios, allowing the clock input 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.

9.4 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. Since the timer is off in sleep mode, thus the TIMER1 interrupt cannot Wake up MCU.

9.5 TIMER1 relevant register

TIMER1 is mainly controlled by 3 RAM, TMR1 control register T1CON, data register TMR1L, TMR1H. Data register while assigning value must first assigns the lower bits TMR1L, followed by TMR1H.

TIMER1 Data register lower bits TMR1L (0EH)

0EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR1L								
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

TIMER1 Data register higher bits TMR1H (0FH)

0FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR1H								
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

TIMER1control register T1CON (10H)

10H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	---	---	TMR1CS	TMR1ON
R/W	R/W	R/W	R/W	R/W	---	---	R/W	R/W
Reset value	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~Bit2 Reserved:

Bit1 TMR1CS: TIMER1 clock source selection bit;
1= External clock source from T1CKI pin (rising edge trigger);
0= Internal clock source ($F_{osc/4}$).

Bit0 TMR1ON: TIMER1enable bit;
1= Enable TIMER1;
0= Disable TIMER1.

Note:
1) T1GINV bit can let TIMER1 gate control signal logical voltage level reversed, regardless gate control signal source.
2) TMR1GE bit must be set to 1, so to use T1G pin as gate control signal source of TIMER1.

10. TIMER2

10.1 TIMER2 general

TIMER2 module 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 postscaler ratio (1: 1 to 1: 16).

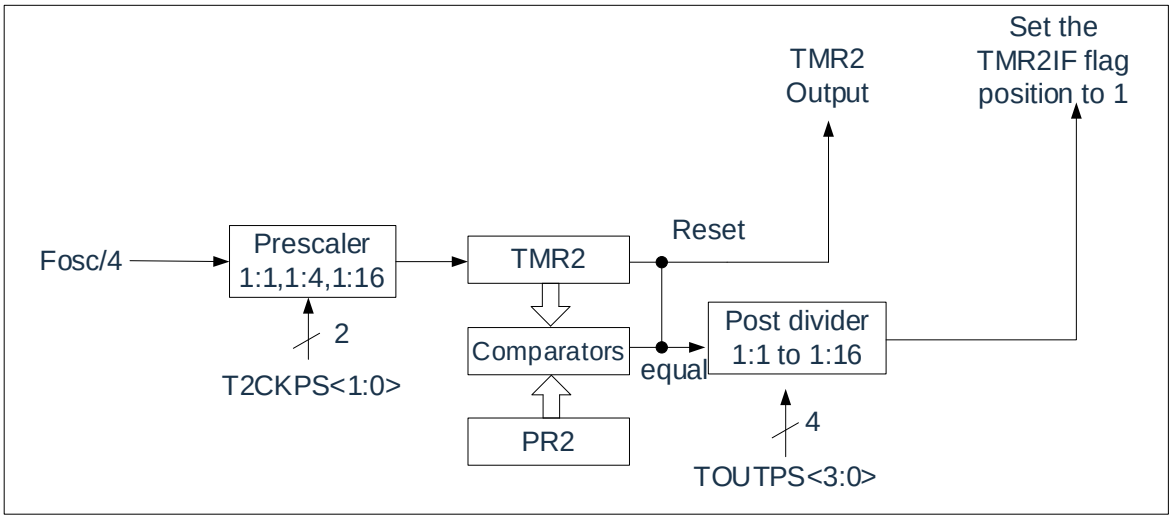


Fig 10-1: TIMER2 structure

10.2 Working principle of TIMER2

The input clock of the TIMER2 module is the system instruction clock (Fosc/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:

- Perform write operation to TMR2 Register
- Perform write operation to T2CON Register
- Any device reset occurs (power-on reset, watchdog timer reset, or undervoltage reset).

Note: Writing T2CON will not clear 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, read 0.

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 Overview

The analog-to-digital converter (ADC) can convert the analog input signal into a 10-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 10-bit binary result and save the result in the ADC result register (ADRESL and ADRESH).

ADC reference voltage is always generated internally. ADC can generate an interrupt after conversion is completed.

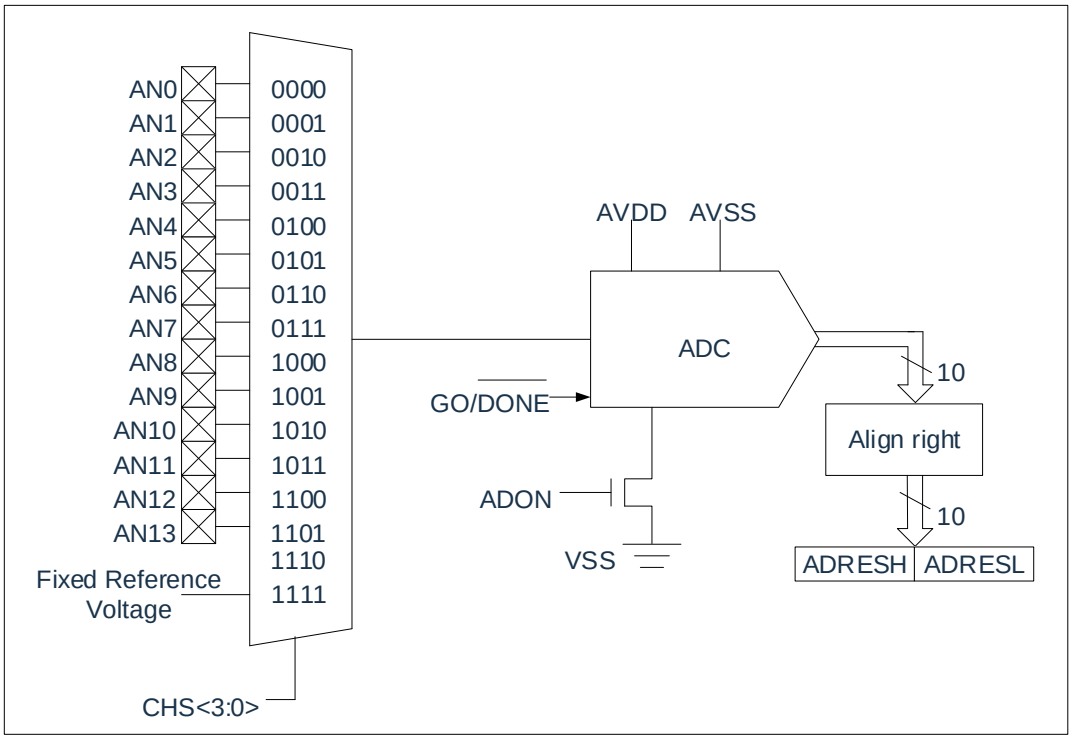


Fig 11-1: ADC structure

11.2 ADC Configuration

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

- Port configuration.
- Channel selection.
- ADC conversion clock source.
- Interrupt control.

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 and ANSEL 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 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 chapter 11.3 “ADC working principal”.

11.2.3 ADC Reference Voltage

The ADC reference voltage is always provided by the chip's VDD and GND.

11.2.4 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:

- Fosc/8
- Fosc/32
- Fosc/16
- F_{RC} (special internal oscillator)

The time to complete one-bit conversion is defined as TAD. A complete 10-bit conversion requires 41 TAD periods.

Must comply with the corresponding TAD 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.

Relationship of ADC clock period (TAD)and working frequency of chip (VDD=5.0V)

ADC clock period		Chip frequency		
ADC clock source	ADCS<1: 0>	8MHz	4MHz	1MHz
Fosc/8	00	49.0μs	98.0μs	392.0μs
Fosc/16	01	98.0μs	196.0μs	784.0μs
Fosc/32	10	196.0μs	392.0μs	1.5ms
FRC	11	1-3ms	1-3ms	1-3ms

Note: Suggest not to use the value in shadowed field.

11.2.5 ADC Interrupt

ADC module 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.

No matter the device is in working mode or sleep mode, interrupt can be generated. If the device is in sleep mode, the interrupt can Wake up the device. When the device is woken up from sleep state, always execute the next instruction after STOP instructions. If the user tries to use When the device wakes up from sleep mode and resumes code execution in order, global interrupt must be disabled. If global interrupt is allowed, the program will jump to the interrupt service routine for execution.

11.3 ADC Working Principle

11.3.1 Start Conversion

To enable ADC module, 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 module 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/DONEbit. 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 TAD 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 module and terminate any pending conversions.

11.3.4 Working Principle of ADC in Sleep Mode

ADC module can work in sleep mode. This operation requires ADC clock source to be set to FRC option. If FRC clock source is selected, ADC must wait for one more instruction period before starting conversion. This allows the execution of STOP instructions to reduce conversion If the ADC interrupt is allowed, the device will Wake up from sleep mode when the conversion ends. If the ADC interrupt is disabled, even if the ADON bit remains set, the ADC module will be closed after the conversion is complete. If the ADC clock source is not FRC, even if the ADON bit remains set, executing the STOP instructions will abort the current conversion and close the A/D module.

11.3.5 A/D Conversion Procedure

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

1. port configuration:
   - Forbidden pin to be configured to output driver (Refer TRIS register)
   - Configure pin as input pin (see TRIS register).
2. configuration ADC module:
   - Select ADC conversion clock.
   - Select ADC input channel.
   - Choose the format of the result.
   - Start the ADC module.
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 port
        SETB    ANSEL, 0        ; set PORTA.0 as analog port
        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 RAM

There are mainly 3 RAMs related to AD conversion, namely control register ADCON, data register ADRESH and ADRESL.

AD control register ADCON (1FH)

1FH	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= $F_{osc}/8$
01= $F_{osc}/16$
10= $F_{osc}/32$
11= F_{RC} (A dedicated internal oscillator generates a clock with a frequency of up to 500KHz)

Bit5~Bit2 CHS<3: 0>: The analog channel selection bits.
CHS<4: 0>:
0000= AN0
0001= AN1
0010= AN2
0011= AN3
0100= AN4
0101= AN5
0110= AN6
0111= AN7
1000= AN8
1001= AN9
1010= AN10
1011= -----
1100= -----
1101= -----
1110= CVREF
1111= Fixed Reference voltage (0.6V fixed reference voltage)

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, not consuming current.

AD data register high bit ADRESH (9FH)

9DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESH	----	----	----	----	----	----	ADRES9	ADRES8
read/write	----	----	----	----	----	----	R	R
Reset value	----	----	----	----	----	----	X	X

Bit1~Bit0 ADRES<9: 8>: ADC result register bit.

The higher 2 bits of the 10-bit conversion result.

AD data register lower bit ADRESL (9EH)

9CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESL	ADRES7	ADRES6	ADRES5	ADRES4	ADRES3	ADRES2	ADRES1	ADRES0
read/write	R	R	R	R	R	R	R	R
Reset value	X	X	X	X	X	X	X	X

Bit7~Bit0 ADRES<7: 0>: ADC result register bit.

The lower 8 bits of the 10-bit conversion result.

12. PWM Module

12.1 PWM Feature

- 8Bit precision.
- Output polarity configurable.
- Counter frequency configurable.
- Selectable center aligned or edge aligned Output mode

CMS89F52x has built-in 2 channel 8-bit PWM module. The PWM module can generate pulse modulated waveform which the period and duty cycle both are adjustable. The 2 channel PWM generated waveform will be output via RA0 and RA1.

12.2 PWM Relevant Registers

PWM functional related registers are control register PWM0CR, PWM1CR; period configure register PWM0PR, PWM1PR; duty cycle configure register PWM0DR, PWM1DR.

PWM0 duty cycle configure register PWM0DR (18H)

18H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM0DR	PWM0DR [7: 0]							
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

PWM0 period configure register PWM0PR (19H)

19H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM0PR	PWM0PR [7: 0]							
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

PWM0 control register PWM0CR (1AH)

1AH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM0CR	PWM0EN	PWM0MOD	---	PWM0POL	PWM0CKS [3: 0]			
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 PWM0EN: PWM0 enable bit
0: Disable
1: enable (PWM0 port configure to Output, and OutputPWM0 波形)
Bit6 PWM0MOD: PWM0 mode selection bit
0: Normal mode
1: back-to-back mode
Bit5 reserved
Bit4 PWM0POL: PWM0Output polarity selection bit
0: Normal Output
1: Reverse phased Output
Bit3~Bit0 PWM0CKS [3: 0]: PWM0 clock frequency selection bit

0000:	Fosc	1000:	Fosc/256
0001:	Fosc/2	1001:	Fosc/512
0010:	Fosc/4	1010:	Fosc/1024
0011:	Fosc/8	1011:	Fosc/2048
0100:	Fosc/16	1100:	Fosc/4096
0101:	Fosc/32	1101:	Fosc/8192
0110:	Fosc/64	111x:	Fosc/8192
0111:	Fosc/128		

PWM1 duty cycle configuration register PWM1DR (1BH)

1BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM1DR	PWM1DR [7: 0]							
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

PWM1 period configuration register PWM1PR (1CH)

1CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM1PR	PWM1PR [7: 0]							
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

PWM1 control register PWM1CR (1DH)

1DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM1CR	PWM1EN	PWM1MOD	---	PWM1POL	PWM1CKS [3: 0]			
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 PWM1EN: PWM1 enable bit
0: Disable
1: enable (PWM1 port configure to Output, and OutputPWM1 波形)
Bit6 PWM1MOD: PWM1 mode selection bit
0: Normal mode
1: back-to-back mode
Bit5 reserved
Bit4 PWM1POL: PWM1Output polarity selection bit
0: Normal Output
1: Reverse phased Output
Bit3~Bit0 PWM1CKS [3: 0]: PWM1 clock frequency selection bit

0000:	Fosc	1000:	Fosc/256
0001:	Fosc/2	1001:	Fosc/512
0010:	Fosc/4	1010:	Fosc/1024
0011:	Fosc/8	1011:	Fosc/2048
0100:	Fosc/16	1100:	Fosc/4096
0101:	Fosc/32	1101:	Fosc/8192
0110:	Fosc/64	111x:	Fosc/8192
0111:	Fosc/128		

- PWM0/PWM1 relevant parameters: n=0, 1
- PWMn period: Tpwmp= (PWMnPR [7: 0] +1) × 2PWMnCKS [3: 0] × Tsys (longest period: 262Ms@Fosc= 8MHz)
- PWMn high voltage pulse time: Tpwmh=PWMnDR [7: 0] ×2PWMnCKS [3: 0] ×Tsys
- PWMn duty cycle: Tpwmh / Tpwmp = PWMnDR [7: 0] / (PWMnPR [7: 0] +1) (configure range: 0%-100%, max precision 1/256)
- PWMn back-to-back mode duty cycle: (2*PWMnDR [7: 0] +1) / 2* (PWMnPR [7: 0] +1)
- If PWMnDR or PWMnPR is 0, then duty cycle is: 0%;If PWMnDR [7: 0] ≥PWMnPR [7: 0] +1, then duty cycle is: 100%.

13. Capture Module CCP

13.1 Capture CCP Register

The capture module is the peripheral that allow users to time and control different events. In capture mode, the peripheral can time the duration of the event. When CCP is used in capture mode, timer TIMER1 is required.

CCP control register CCPCON (190H)

190H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CCPCON	CCPEN	----	----	CCPIS	CCPES	CCPM2	CCPM1	CCPM0
read/write	R/W	----	----		R/W	R/W	R/W	R/W
Reset value	0	----	----	0	0	0	0	0

Bit7	CCPEN:	Capture function enable bit;
	0=	Disable capture function;
Bit6~Bit5	1=	Enable capture function.
Bit4	Reserved	
	CCPIS:	Capture clock source selection bit;
	0=	Clock source from RA2 port input;
Bit3	1=	Clock source from synchronized comparator Output.
Bit2~Bit0	CCPES:	Capture clock edge selection bit;
	0=	Capture at clock falling edge;
	1=	Capture at clock rising edge.
	CCPM<2: 0>:	Capture mode selection bit;
	000=	Capture every 1 clock:
	001=	Capture every 2 clock:
	010=	Capture every 4 clock:
	011=	Capture every 8 clock:
	100=	Capture every 16 clock:
	101=	Capture every 32 clock:
	110=	Capture every 64 clock:
	111=	Capture every 128 clock:

13.2 Capture Mode

The event type is selected by the mode selection bit CCPM2: CCPM0 (CCPCON<2: 0>). When a capture occurs, the interrupt request flag bit CCPIF in the PIR1 register is set to 1; it must be cleared by software. If another capture occurs before the value in the register CCPRH and CCPRL is read, then the previous capture value will be overwritten by the new capture value (see Figure 13-1).

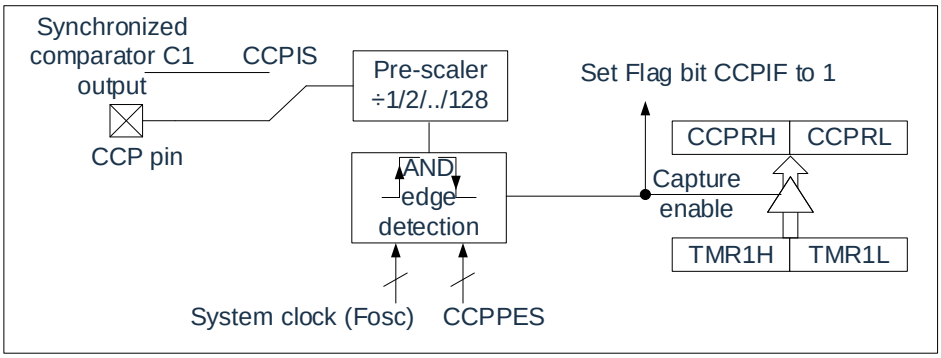


Fig 13-1: capture mode working structure

13.2.1 CCP Pin Configuration

In capture mode, the corresponding CCP pin should be configured as input by setting the corresponding TRIS control bit to 1.

> Note: If the CCPx pin is configured as output, a write operation to the port may trigger a capture event.

13.2.2 TIMER1 Mode Selection

TIMER1 must run in timer mode or synchronous counter mode CCP module to use the capture function. Capture operation cannot be performed in asynchronous counter mode.

13.2.3 Software Interrupt

When the capture mode is changed, a false capture interrupt may occur. The user should keep the CCPIE interrupt enable bit in the PIE1 register cleared to avoid false interrupts. The interrupt flag bit CCPIF in the PIR1 register should also be cleared after any change in the operation mode.

14. Master Control Synchronous Serial Port (MSSP)module

14.1 Master Control SSP (MSSP) Module Overview

master control synchronous serial port (Master Synchronous Serial Port, MSSP) module is a serial interface for communicating with other peripherals or microcontrollers. These peripherals devices can be serial EEPROM, shift register, display driver or A/D converter, etc.

MSSP module has the following two working modes:

- serial peripherals ports (SPI).
- I^2C.
  - Full master control mode.
  - Slave mode (Support broadcast address call).

I^2C interface supports the following modes at hardware level:

- master control mode.
- Multi master mode.
- Slave mode.

14.2 SPI Mode

SPI mode allows simultaneous transmit and receive 8-bit data at the same time. SPI supports all 4 modes of 3-wire communication.

The following three pins are used:

- serial data output (SDO)——RA3/SDO
- serial data input (SDI)——RA5/SDI/SDA
- serial clock (SCK)——RA6/SCK/SCL

  Also, in any slave mode, the 4^{th} pin can be used:
- slave selection (SS)——RE1/AN10/SS

14.2.1 SPI Related Register

SSPSTAT: SSP status register (193H)

193H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SSPSTAT	SMP	CKE	---	---	---	---	---	---
read/write	R/W	R/W	---	---	---	---	---	---
Reset value	0	0	---	---	---	---	---	---

Bit7 SMP: Sample Bit

SPI master mode:

1 = Sample input data at the end of data output time.

0 = Sample input data at the middle of data output time.

SPI Slave mode: When use SPI slave mode, must clean SMP.

Bit 6 CKE: SPI clock edge selection bit

CKP= 0

1= Transmit data at rising edge of SCK pin;

0= Transmit data at falling edge of SCK pin;

CKP = 1

1 = Transmit data at falling edge of SCK pin;

0 = Transmit data at rising edge of SCK pin;

Bit5~Bit0 Not used in SPI mode

SSPCON: SSP control register (194H)

194H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SSPCON	---	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
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 SSPOV: Receive overflow flag bit.

1= When SSPBUF still keeps the previous data, a new byte is received. When overflow occurs, the data in SSPSR will be lost. Overflow will only occur in slave mode. In slave mode, even if transmit data only, user must read SSPBUF to avoid overflow. In master control mode, the overflow bit is not set to 1, because every time you receive or transmit new data, it must be started by writing to the SSPBUF register (this bit must be clear through software).

0= No overflow.

Bit5 SSPEN: Synchronous serial port enable bit.

1= Enable serial port and configure SCK, SDO, SDI and SS as serial port pin.

0= disable serial port and configure these pins as I/O port pins.

Bit4 CKP: Clock polarity selection bit.

1= Clock is high when idle.

0= Clock is low when idle.

Bit3~Bit0 SSPM<3: 0>: Synchronous serial port mode selection bit;

0000= SPI master control mode, clock= $F_{osc}/4$;

0001= SPI master control mode, clock= $F_{osc}/16$;

0010= SPI master control mode, clock= $F_{osc}/64$;

0011= SPI master control mode, clock= TMR2 output/2;

0100= SPI slave mode, clock= SCK pin, enable SS pin control;

0101= SPI slave mode, clock= SCK pin, disable SS pin control, SS can be used as I/O pin;

0110= I2C slave mode, 7-bit address;

0111= I2C slave mode, 10-bit address;;

1000= I^2C master control mode, clock= F_{osc}/ (4 * (SSPADD+1)) ;

1001= Disable load function;

1010= reserved;

1011= reserved;

1100= reserved;

1101= reserved;

1110= I^2C slave mode, 7-bit address, and allow start bit and stop bit interrupt;

1111= I^2C slave mode, 10-bit address, and allow start bit and stop bit interrupt;

14.2.2 SPI Working Principle

When initializing the SPI, several options need to be specified. They can be specified by programming the corresponding control bits (SSPCON<5: 0> and SSPSTAT<7: 6>). These control bits are used to specify the following options:

- master control mode (SCK as clock output)
- Slave mode (SCK as clock input)
- clock polarity (SCK idle state)
- Sampling phase of input data (the middle or end of data output time)
- clock rate (only in master control mode)
- clock edge (output data on the rising/falling edge of SCK)
- slave selection mode (only in slave mode)

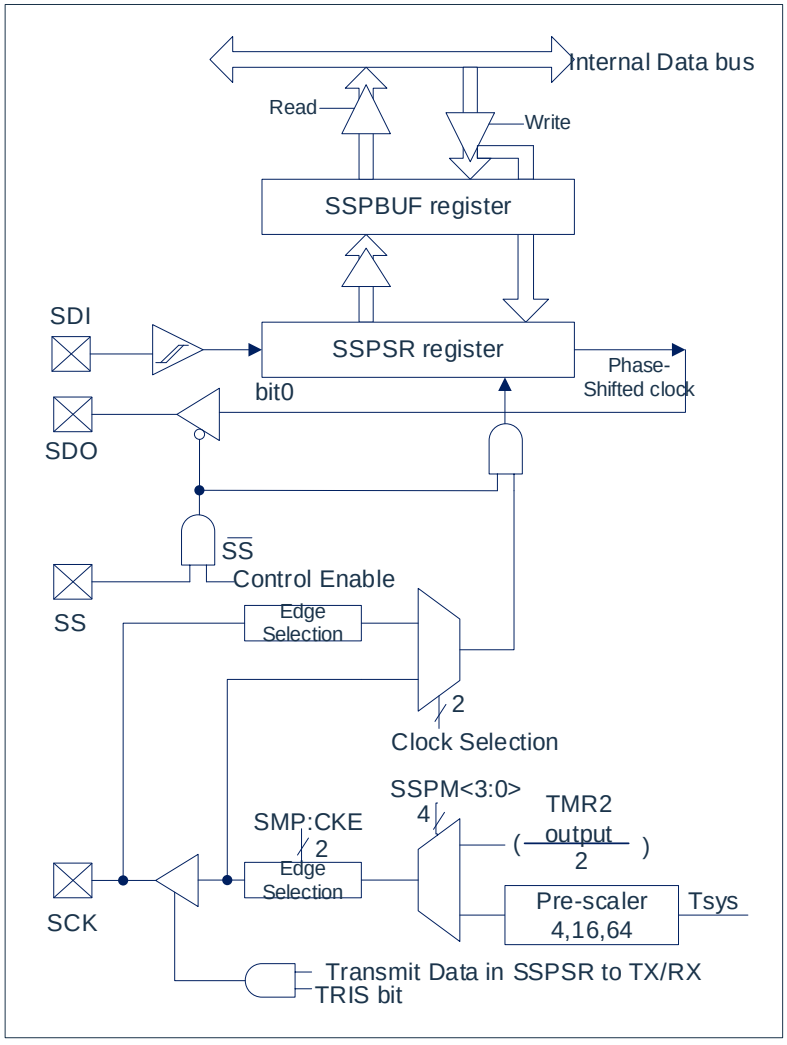


Fig 14-1 MSSP module block diagram in SPI mode

Note: I/O pin has diode protection to VDD and VSS.

MSSP module consists of a transmit/receive shift register (SSPSR) and a buffer register (SSPBUF). SSPSR moves data in and out of the device, with the most significant bit first. SSPBUF saves the data written to the SSPSR last time until the new receive the data is ready. Once the 8-bit data receive is completed, the byte is moved into the SSPBUF register. Then, the interrupt flag bit SSPIF of the PIR1 register is set to 1. This double-buffered data receive method (SSPBUF) allows reading the newly received data before starting to receive the next byte. During the data transmit/receive period, any attempt to write to the SSPBUF register will be ignored, and the write conflict detection bit WCOL of the SPCON register will be set to 1. At this time, the user must clear the WCOL bit by software., Otherwise it cannot be judged whether the next write operation to SSPBUF is successfully completed.

When the application software is waiting for the receive valid data, it should read the previous data in the SSPBUF before the next data byte to be transmitted is written into the SSPBUF. The buffer full flag bit BF (SSPSTAT register) is used to indicate when the SSPBUF has been loaded the received data (transmit is completed). If the SPI is only used as a transmitter, you don’t need to pay attention to the received data. MSSP interrupt can usually be used to determine when the transmit or receive is completed. If you do not use interrupt to handle the data transmission and reception and use software to query, this method also ensures that no write conflicts occur.

14.2.3 Enable SPI I/O

To enable the serial port, the MSSP enable bit SSPEN of the SSPCON register must be set to 1. To reset or reconfigure the SPI mode, first clear the SSPEN bit, reinitialize the SSPCON register, and then set the SSPEN bit to 1. This will set SDI, SDO, The SCK and SS pins are configured as serial port pins. To use these pins as serial ports, the data direction bits (in the TRIS register) must be programmed correctly, as follows:

- SDI controlled by SPI module.
- TRISA<3> of SDO must be cleared.
- The TRISA<6> bit of SCK (master control mode) must be cleared.
- The TRISA<6> bit of SCK (slave mode) must be set to 1.
- The TRISE<1> of SS must be set to 1.

For any unwanted serial port function, you can skip it by setting the corresponding data direction (TRIS) register to the opposite value.

14.2.4 Master Control Mode

The master device controls SCK, so it can start data transmission at any time. The master device determines when the slave device should broadcast data according to the software protocol.

In master control mode, once data is written into the SSPBUF register, it will start to transmit or receive. If SPI is only used as a receiver, you can disable SDO output (program it to input). SSPSR register is connected to the SDI pin at the set clock rate the signal performs continuous shift input. After each byte receive is completed, it will be treated as a normal receive byte and loaded into the SSPBUF register (corresponding to interrupt and status position 1). This can be used as a “line activity monitor” mode, which is very useful.

The clock polarity can be selected by programming the CKP bit of the SSPCON register accordingly. Figure 14-2, Figure 14-3, Figure 14-4, and Figure 14-5 show the SPI communication waveforms, where MSb is first transmitted. In master control mode, the SPI clock rate (bit rate) can be programmed by the user to one of the following rates:

- F_{osc}/4 (or TCY)
- F_{osc}/16 (or 4.TCY)
- F_{osc}/64 (or 16.TCY)
- TIMER2 output/2

Figure 14-2 shows the waveform of the master control mode. When the CKE bit of the SSPSTAT register is 1, the SDO data is valid before the clock edge appears on the SCK. The figure shows input sample change is determined by SMP bit of SSPSTAT register. The figure indicates the time to load the received data into the SSPBUF.

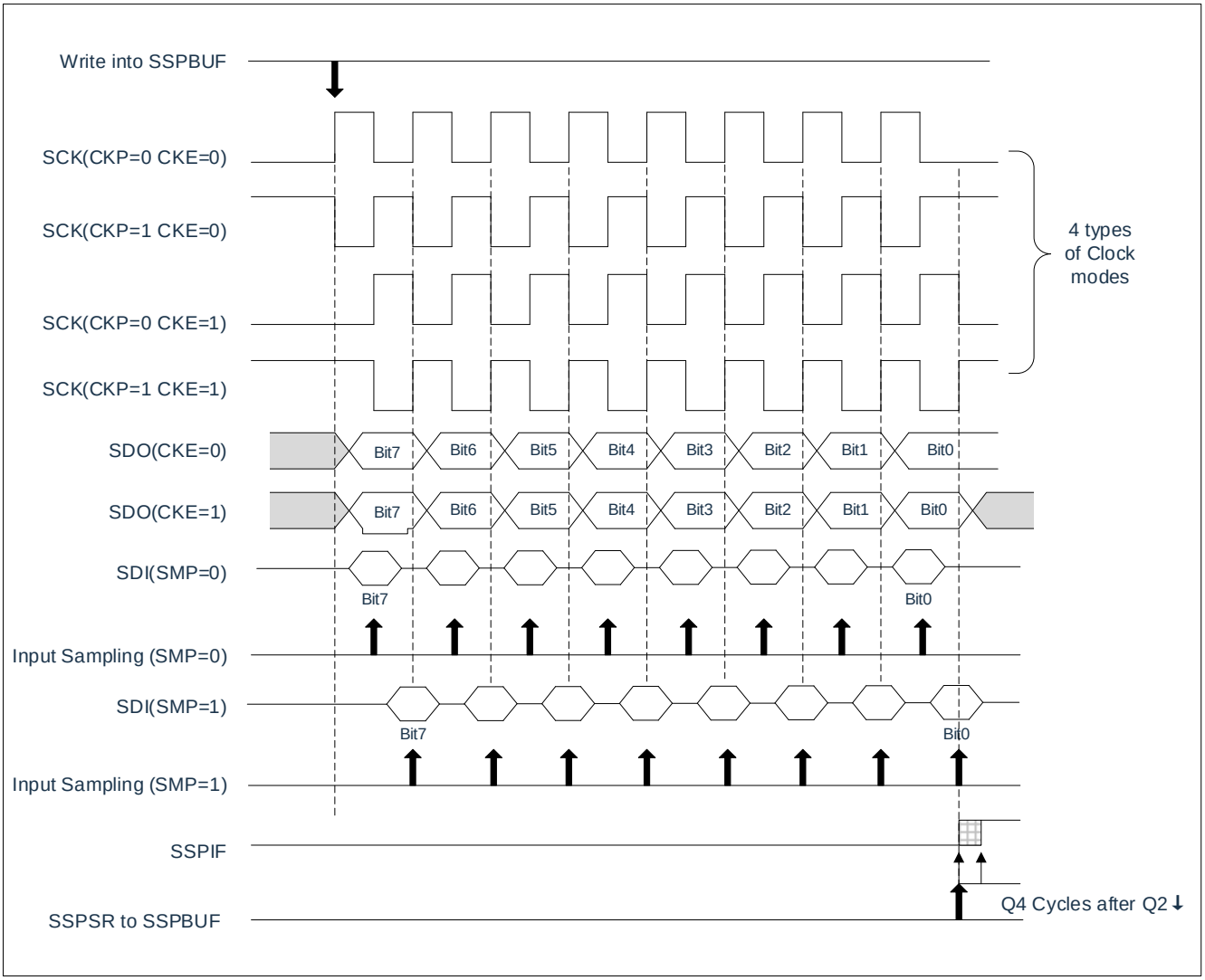


Fig14-2: SPI mode waveform (master control mode)

14.2.5 Slave Mode

In slave mode, when an external clock pulse appears on the SCK pin, transmit and receive data. When the last bit of data is latched, the SSPIF interrupt flag bit of PIR1register is set to 1.

In slave mode, the clock is provided by the external clock source on the SCK pin. The external clock must meet the minimum time requirements for high and low levels specified in the electrical specifications.

In the sleep state, the slave device can still transmit/receive data. When a byte is received, the device is awakened from the sleep state.

14.2.6 Slave Synchronous Selection

SS pin allows the device to work in synchronous slave mode. SPI must work in slave mode and enable SS pin control. To use SS pin as input in, the pin driver cannot be set to low level. When the SS pin is low, the transmit and receive of the data are enabled, and the SDO pin is used by the driver. When the SS pin is high, the SDO pin is no longer driven even during the data transmit process. It becomes a floating output. According to the needs of the application, an external pull up/ pull down resistor can be connected.

After SPI module reset, the bit counter is forced to 0. This can be achieved by forcing the SS pin to be pulled high or clearing the SSPEN bit. Connecting the SDO pin and the SDI pin can simulate a two-wire communication. When SPI When it needs to work as a receiver, SDO pin can be configured as input. This will disable the transmit data from SDO. Because SDI will not cause a bus conflict, it can always be reserved as input (SDI function).

Note

1) When SPI works in slave mode and SSpin control is enabled, if SSpin is set to VDD level, SPI module will be reset.
2) If CKE set to 1 (SSPSTAT register) and use SPI slave mode, then must enable SS pin control.

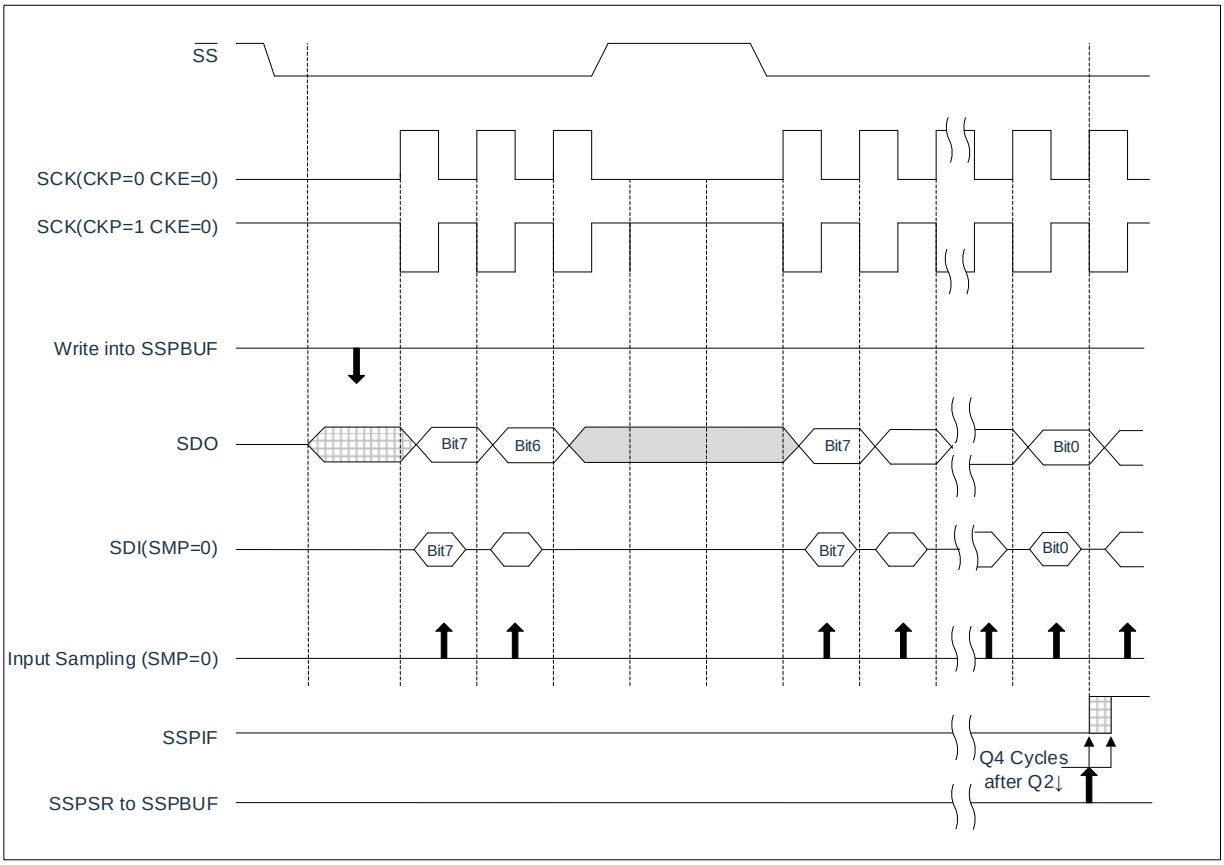


Fig 14-3: Slave synchronous waveform

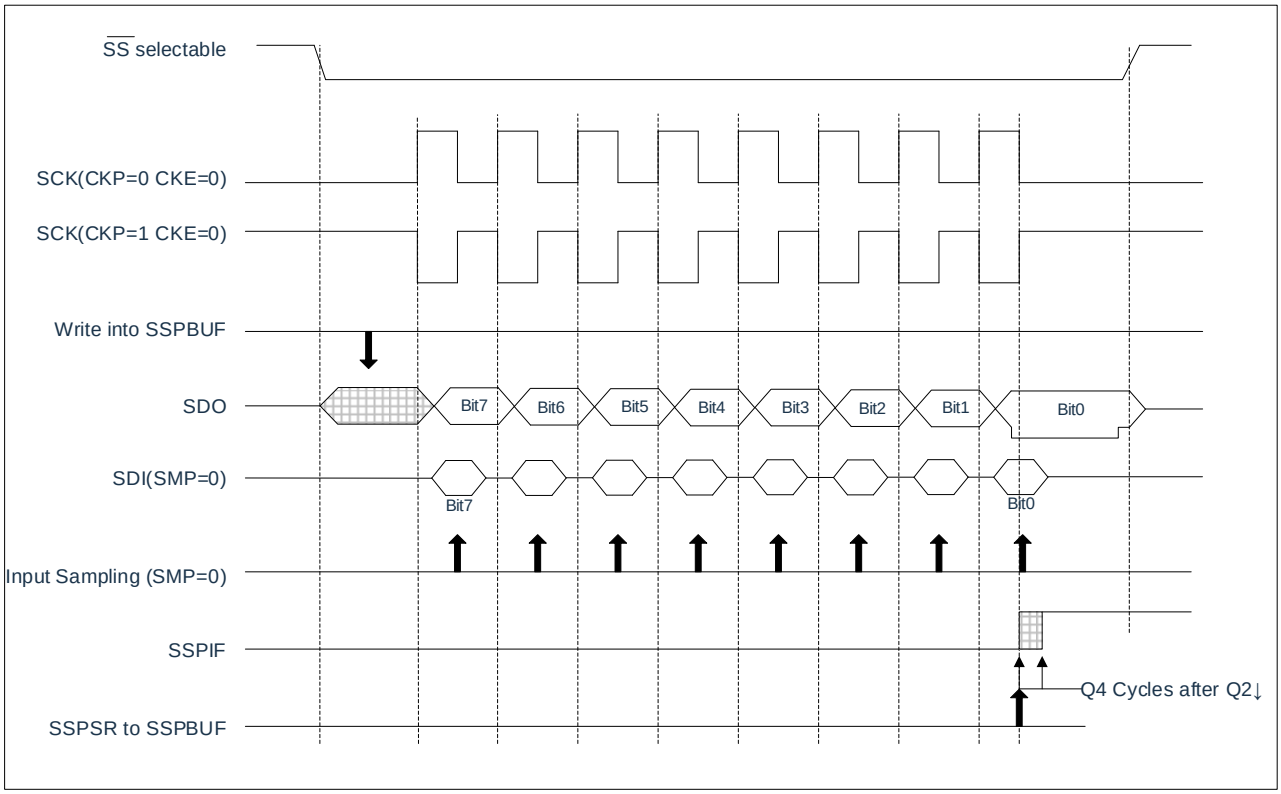


Fig 14-4: SPI mode waveform (slave mode, CKE=0)

14.2.7 Sleep Operation

Under master mode, if selected sleep mode, all module clocks will stop, and before the device is awakened, transmit/receive will remain in this stagnant state. When the device returns to running mode, the module will resume to transmit and receive data.

Under slave mode, SPI transmit/receive shift register is working asynchronized with MCU. Thus, while MCU is under sleep mode, the data can still be moved into SPI transmit/receiver shift register. When 8-bit data are all received, MSSP interrupt flag bit will be set to 1, and if interrupt is allowed, the MCU will be waked up.

14.2.8 Effect of Reset

reset will disable MSSP module and terminate the current transmission.

14.3 I²C Module

When MSSP module works in I²C mode, it can realize all master control and slave functions (including broadcast call support) and use hardware to provide interrupts of the start and stop bits to determine when the bus is idle (multi-master function). MSSP module implementation follows standardization, support 7-bit and 10-bit addressing.

There are two pins for data transmission. They are clock pin (SCL) ---RA6/SCK/SCL pin and data pin (SDA) --- RA5/SDI/SDA pin. The user must use TRISA<6: 5> bits to configure as input or output pins. By setting the MSSP enable bit, SSPEN, of the SSPCON register to 1, MSSP module function is enabled.

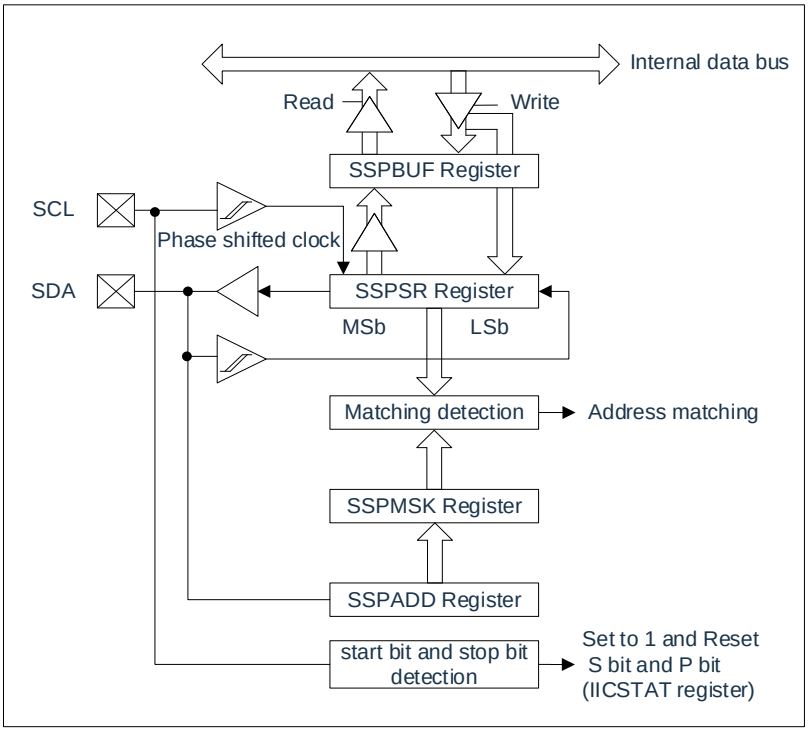


Fig14-6: MSSP block diagram (I²C mode)

Note: The I/O pin has protection diodes connected to VDD and VSS.

MSSP module has 7 registers for I²C operation. They are:

- MSSP control register1 (SSPCON)
- MSSP control register2 (SSPCON2)
- MSSP status register (SSPSTAT)
- serial receive/transmit buffer register (SSPBUF)
- MSSP shift register (SSPSR): not directly accessible
- MSSP address register (SSPADD)
- MSSP masking register (SSPMSK)

You can use SSPCON register to control the operation of I²C. You can use the SSPM<3: 0> mode selection bit (SSPCON register) to select one of the following I²C modes:

- I2C slave mode (7-bit address)
- I2C slave mode (10-bit address)
- I²C firmware control master operation, slave device idling
- I2C master control mode, clock=FSYS/ (4* (SSPADD+1))
- I2C slave mode, 7-bit address, allow start bit and stop bit interrupt
- I2C slave mode, 10-bit address, allow start bit and stop bit interrupt

If the SCL and SDA pins have been programmed as input pins (set the corresponding TRISA bit to 1), selecting any I²C mode and SSPEN bit as 1 will force the SCL and SDA pins to be open drain.

14.3.1 related register illustration

SSPSTAT: SSP status register (193H)

193H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SSPSTAT	SMP	CKE	D/A	P	S	R/W	UA	BF
read/write	R/W	R/W	R	R	R	R	R	R
Reset value	0	0	0	0	0	0	0	0

Bit7 SMP:

Bit6 1= Disable variable rate control, standard speed modes (100KHz和1MHz).

0= Enable variable rate control, high speed mode (400KHz).

Bit6 CKE: SPI clock edge selection bit.

Bit5 CKP= 0

1= Transmit data at rising edge of SCK pin;

0= Transmit data at falling edge of SCK pin;

CKP = 1

1 = Transmit data at falling edge of SCK pin;

0 = Transmit data at rising edge of SCK pin;

Bit4 P: Stop bit (this bit is cleared when MSSP module is disabled (SSPEN is cleared)).

1= Indicates that the stop bit was finally detected (the bit is 0 when reset).

0= Indicates that the stop bit was not detected at the end.

Bit3 S: Start bit (this bit is cleared when disable MSSP module (SSPEN is cleared)).

1= Indicates that the start bit was finally detected (the bit is 0 when reset).

0= The start bit was not detected at the end.

Bit2 R/W: Read/write bit.

This bit is used to save the R/W bit information after the last address match. This bit is only valid from the address match to the next start bit, stop bit or non-ACK bit.

In I^2C slave mode:

1= read.

0= write.

在I^2C master control mode:

1= transmitting.

0= not transmitting.

The result of logic OR operation between this bit and SEN, RSEN, PEN, RCEN or ACKEN will indicate whether MSSP is in idle mode.

Bit1 UA: Update address bit (only used in 10-bit I^2C mode).

1= Means user need to update the address of SSPADD register.

0= No need to update address.

Bit0 BF buffer full status bit.

receive:

1= receive complete, SSPBUF full.

0= receive not complete, SSPBUF empty.

transmit:

1 = data transmitting (not including ACK and stop bit), SSPBUF full.

0 = data transmit complete (not including ACK and stop bit), SSPBUF empty.

SSPCON: SSP control register (194H)

194H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
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 WCOL: Write conflict detection bit.

master control mode: 1= Trying to write to the SSPBUF register when I^2C does not meet the condition of starting to transmit data.

0= no conflict.

Slave mode:

1= While transmitting the previous work, write the SSPBUF register again (must clear through software).

0= no conflict.

Bit6 SSPOV: Receive overflow flag bit.

1= When the SSPBUF register still maintains the previous data, it receives a new byte. In the transmit mode, the SSPOV bit can be any value (this bit must be clear through software).

0= No overflow.

Bit5 SSPEN: Synchronous serial port enable bit (These pins must be correctly configured as input or output pins).

1= Enable serial port and configure SDA and SCL pin as serial port pin.

0= Disable serial port and configure these pins as I/O port pins.

Bit4 CKP: Clock polarity selection bit.

In I^2C slave mode: SCK release control.

1 = enable clock.

0 = Keep clock line is low (clock extension) (used to ensure data establishment time).

In I^2C master control mode: Not used.

Bit3~Bit0 SSPM<3: 0>: Synchronous serial port mode selection bit.

0000= SPI master control mode, clock= $F_{osc}/4$.

0001= SPI master control mode, clock= $F_{osc}/16$.

0010= SPI master control mode, clock= $F_{osc}/64$.

0011= SPI master control mode, clock= TMR2 output/2.

0100= SPI slave mode, clock= SCKpin, enable SS pin control.

0101= SPI slave mode, clock= SCKpin, disable SS pin control, SS can be used as I/O pin.

0110= I2C slave mode, 7-bit address;

0111= I2C slave mode, 10-bit address;;

1000= I^2C master control mode, clock= $F_{osc}/ (4 * (SSPADD+1))$;

1001= Disable load function;

1010= reserved;

1011= reserved;

1100= reserved;

1101= reserved;

1110= I^2C slave mode, 7-bit address, and allow start bit and stop bit interrupt;

1111= I^2C slave mode, 10-bit address, and allow start bit and stop bit interrupt;

SSPCON2: SSP control register2 (195H)

195H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SSPCON2	GCEM	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
read/write	R/W	R/W	R	R	R	R	R	R
Reset value	0	0	0	0	0	0	0	0

Bit7 GCEN: Broadcast call enable bit (only in I^2C slave mode).

1= It is allowed to generate interrupt when receiving to the general call address (0000h) in SSPSR.

0= Disable broadcast call address.

Bit6 ACKSTAT: ACK status bit (only in I^2C master control mode).

In master control transmit mode:

1 = Did not receive a response from the slave device.

0 = A response from the slave device has been received.

Bit5 ACKDT: ACK data bit (only in I^2C master control mode).

In master control receive mode: The value of the user's response sequence after the receive is completed.

1 = not respond.

0 = respond.

Bit4 ACKEN: ACK enable bit (only in I^2C master control mode).

In master control receive mode:

1 = Start the response sequence on the SDA and SCL pin, transmit ACKDT data bit. Automatically cleared by hardware.

0 = Response sequence idle.

Bit3 RCEN: Receive enable bit (only in I^2C master control mode).

1= Enable I^2C receive mode.

0= Receive idle.

Bit2 PEN: stop enable bit (only in I^2Cmaster control mode).

1 = Start stop condition on SDA and SCL pin. Automatically cleared by hardware.

0 =idle.

Bit1 RSEN: Repeat enable bit (only in I^2Cmaster control mode).

SCK release control

1= Initiate repeated start conditions on the SDA and SCL pins. Automatically cleared by hardware.

0= idle.

Bit0 SEN: Start enable bit (only used in I2C master mode)

In master control mode:

1 = Start the start conditions on the SDA and SCL pins. Automatically cleared by hardware.

0 = idle.

In slave mode:

1 = Both transmit and receive will enable clock extension (enable clock extension).

0 = disable clock extension.

14.3.2 master control mode

The master control mode works by generating interrupt when the start and stop conditions are detected. The stop (P) bit and the start (S) bit are cleared when reset or disable MSSP module. When the P bit is set to 1, the control of I2C bus can be obtained; otherwise, the bus is idle, and both the P and S bits are zero.

In master control mode, the SCL and SDA line is manipulated by the MSSP hardware. The following events will set the MSSP interrupt flag bit SSPIF to 1 (if MSSP interrupt is allowed, interrupt will be generated):

- Start condition
- Stop condition
- Data transmission byte has been transmitted/received
- Reply to transmit
- Repeated start conditions

14.3.3 I²C master control mode support

The master control mode can be enabled by setting the corresponding SSPM bit in SSPCON to 1 or clearing it and setting the SSPEN bit to 1. Once the master control mode is enabled, the user can select the following 6 operations:

1. Issue a start condition on SDA and SCL.
2. Issue a repeated start condition on SDA and SCL.
3. Write the SSPBUF register to start data/address transmit.
4. Generate a stop condition on SDA and SCL.
5. Configure the I2C port to receive data.
6. The response condition is generated after the data byte is received.

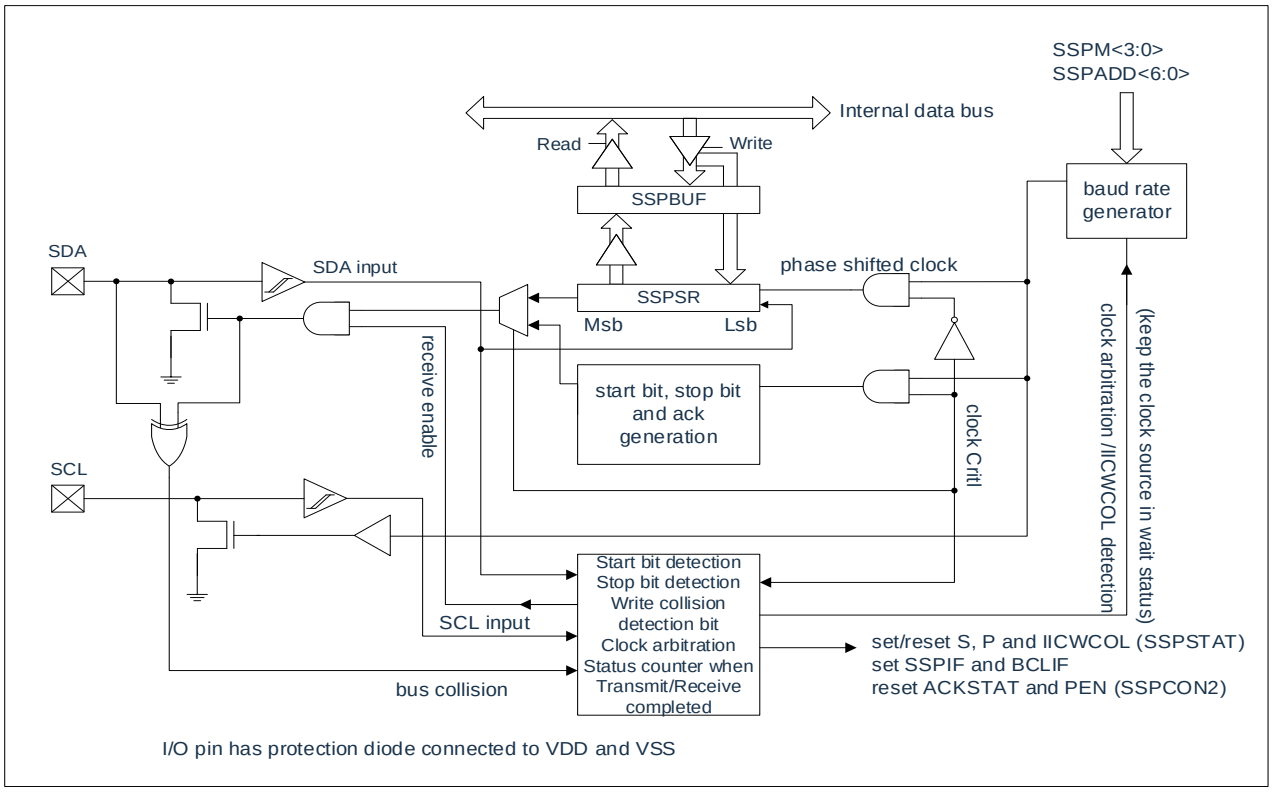


Fig 14-7: MSSP block diagram (I²C™ master control mode)

Note: When configured as I^2C master mode, MSSP module does not allow event queuing. For example, before the end of the start condition, the user is not allowed to issue another start condition and write to the SSPBUF register immediately to initiate the transfer. In this case, SSPBUF will not be written and the WCOL bit will be set to 1, which indicates that no write operation to SSPBUF has occurred.

14.3.3.1 I2C master control mode operation

All serial clock pulses and start/stop conditions are generated by the master device. The stop condition or the repeated start condition can end the transmission. Because the repeated start condition is also the beginning of the next serial transmission, the I^2C bus will not be released. In the master control transmit mode, the serial data is output through SDA, and the serial clock is output by SCL. The first byte of the transmit includes the address (7 bits) and read/write (R/W) bits of the receiver. In this case, R/W bit will be logic 0. Serial data transmits 8 bits each time. Every time a byte is transmitted, an acknowledge bit will be received. The output of the start and stop conditions indicates the start and end of the serial transmission.

In master control receive mode, the first byte of transmit includes the address (7 bits) of the transmit device and the R/W bit. In this case, the R/W bit will be logic 1. Therefore, the first byte of transmit byte is a 7-bit slave device address, followed by 1 to indicate receive. The serial data is received through SDA, while the serial clock is output by SCL. Every time 8 bits of serial data are received. Every time a byte is received, an answer bit will be transmitted. Start and stop conditions indicate the start and end of transmit, respectively.

In I^2C mode, the baud rate generator used in SPI mode is used to set the SCL clock frequency to 100KHz, 400KHz or 1MHz. The reload value of the baud rate generator is located in the lower 7 bits of the SSPADD register. When a write to SSPBUF occurs during operation, the baud rate generator will automatically start counting. If the specified operation is completed (i.e., the last data bit of transmit is followed by ACK), the internal clock will automatically stop counting, and the SCL pin will remain in its last state.

The following is a typical transmit event sequence:

- The user generates a start condition by setting the start enable bit SEN (SSPCON2 register) to 1.
- SSPIF set to 1. Before performing any other operations, MSSP module will wait for the required startup time.
- The user will load the SSPBUF from the device address to transmit.
- The address is moved out of the SDA pin until all 8 bits are transmitted.
- MSSP module shifts in the ACK bit from the slave device and writes its value into the ACKSTAT bit of the SSPCON2 register.
- MSSP module sets the SSPIF bit to 1 at the end of the 9th clock period, generating an interrupt.
- The user loads 8-bit data into SSPBUF.
- Data is moved out from the SDA pin until all 8 bits are transmitted.
- MSSP module shifts in the ACK bit from the slave device and writes its value into the ACKSTAT bit of the SSPCON2 register.
- MSSP module sets the SSPIF bit to 1 at the end of the 9th clock, generating an interrupt.
- The user generates a stop condition by setting the stop enable bit (PEN) bit (SSPCON2 register) to 1.
- Once the stop condition is completed, an interrupt will be generated.

14.3.4 Baud Rate Generator

In I²C master control mode, the baud rate generator reloaded value is located in the lower 7 bits of the SSPADD register (Figure 14-8). When the value is loaded, the baud rate generator will automatically start counting and decrement to 0, and then stop until the next reload. BRG will count down twice on the Q2 and Q4 clock periods in each instructions period (TCY). In I²C master control mode, BRG will be automatically reloaded. For example, when clock arbitration occurs, BRG will be reloaded when SCL pin is sampled to high level (Figure 14-9).

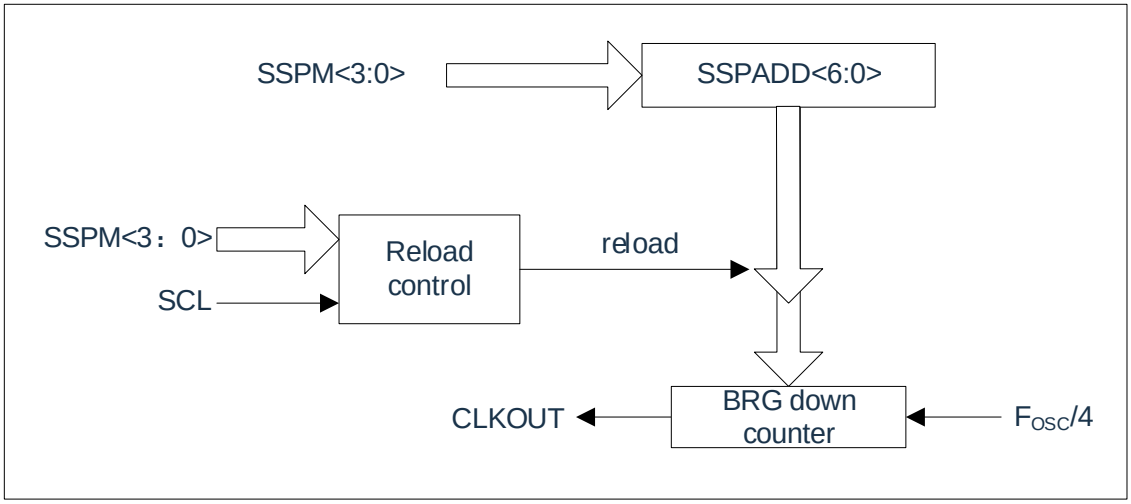


Fig 14-8: baud rate generator block diagram

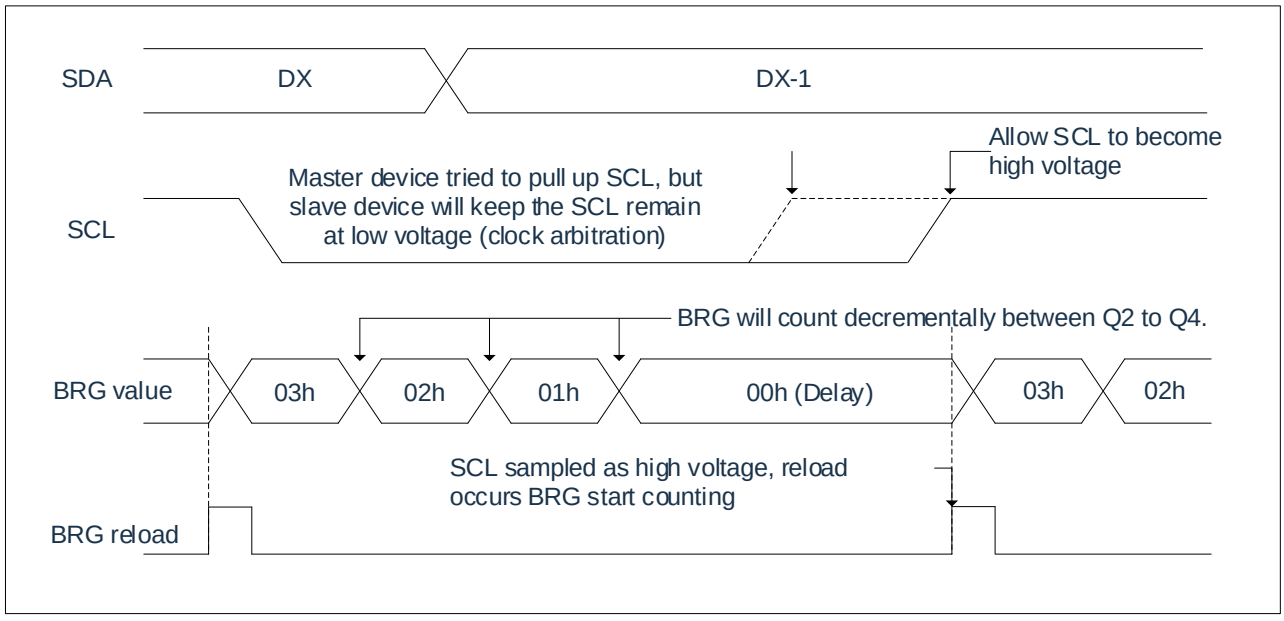


Fig 14-9: Time series of baud rate generator with clock arbitration

14.3.5 I²C Master Control Mode Transmit

Transmit a data byte and a 7-bit address or a 10 bit address the other half, can be achieved directly by writing a value to the SSPBUF register. This operation will set the buffer full flag bit BF to 1, and the baud rate generator will start counting, and at the same time start the next transmit. After the falling edge of SCL is valid, each bit of address/data will be shifted out to the SDA pin. In a baud rate generator full return count period (TBRG), SCL remains low. Data should be released to high level in SCL when SCL pin is released to high level, it will remain high for the entire TBRG. During this period and a period of time after the next SCL falling edge, the data on the SDA pin must remain stable. After the 8th bit is shifted out (the falling edge of the 8th clock period), the BF flag bit is cleared, and the master device releases SDA.

At this time, if an address match occurs or data is correctly received, the addressed slave device will respond with an ACK bit at the 9th bit time. The ACK status is written to the ACKDT bit at the falling edge of the 9th clock period. After master device receiving the response, the response status bit ACKSTAT will be cleared; if the response is not received, the bit will be set to 1. After the 9th clock, the SSPIF bit will be set to 1, and the master control clock (baud rate generator) will be suspended until the next data byte is loaded into SSPBUF, SCL pin remains low, and SDA remains unchanged.

After writing SSPBUF, each bit of address is shifted out on the falling edge of SCL until all 7 bits of address and R/W bit are shifted out. At the falling edge of the eighth clock, the master device pulls the SDA pin to high level to allow the slave device to send an acknowledgment response. At the falling edge of the 9th clock, the master device determines whether the address is recognized by the slave device by sampling the SDA pin. The status of the ACK bit is loaded into the ACKSTAT status bit (SSPCON2 register) .After the 9th clock falling edge of the transmit address, SSPIF is set to 1, the BF flag bit is cleared, the baud rate generator is turned off until the next write operation to SSPBUF, and the SCL pin remains low, allowing the SDA pin to suspend.

14.3.5.1 BF Status Indication

In transmit mode, the BF bit (SSPSTAT register) is set to 1 when the CPU writes SSPBUF and is cleared after all 8 bits of data are shifted out.

14.3.5.2 WCOL Status Indication bit

If the user writes SSPBUF during the transmit process (that is, when the SSPSR is still moving out of the data byte), WCOL is set to 1 and the contents of the buffer remain unchanged (no write operation has occurred). WCOL must clear through software.

14.3.5.3 ACKSTAT Status Indication

In transmit mode, when the slave device transmits a response (ACK=0), the ACKSTAT bit (SSPCON2 register) is cleared; when the slave device does not respond (ACK=1), the bit is 1. The slave device recognizes its address (Including the broadcast call address) or after receiving the data correctly, a response will be transmitted.

14.3.6 I²C Master Control Mode Receive

By programming receive enable bit RCEN (SSPCON2 register) to enable master control mode to receive. The baud rate generator starts counting, and each time the count returns, the state of the SCL pin changes (from high to low or from low to high), and data is shifted into SSPSR. After the falling edge of the eighth clock, the receive enable flag bit is automatically cleared, the content of SSPSR is loaded into SSPBUF, the BF flag bit is set to 1, the SSPIF flag bit is set to 1, the baud rate generator pauses counting, and the SCL remains at low level. At this time, MSSP is in idle state, waiting for the next command. When the CPU reads the buffer, the BF flag bit will be automatically cleared. By setting the response sequence enable bit ACKEN (SSPCON2 register) to 1, the user can end the receive transmit response bit.

14.3.6.1 BF Status Indication

When receiving, when the address or data byte is loaded from SSPSR into SSPBUF, the BF bit is set to 1, and the BF bit is cleared when reading the SSPBUF register.

14.3.6.2 SSPOV Status Flag

When receiving, when SSPSR receives only 8-bit data, SSPOV set to 1, BF flag has been set to 1 at last receive operation.

14.3.6.3 WCOL Status Indication

If the user writes SSPBUF during the receive process (that is, when the SSPSR is still moving into the data byte), the WCOL bit is set to 1, and the buffer content remains unchanged (no write operation has occurred).

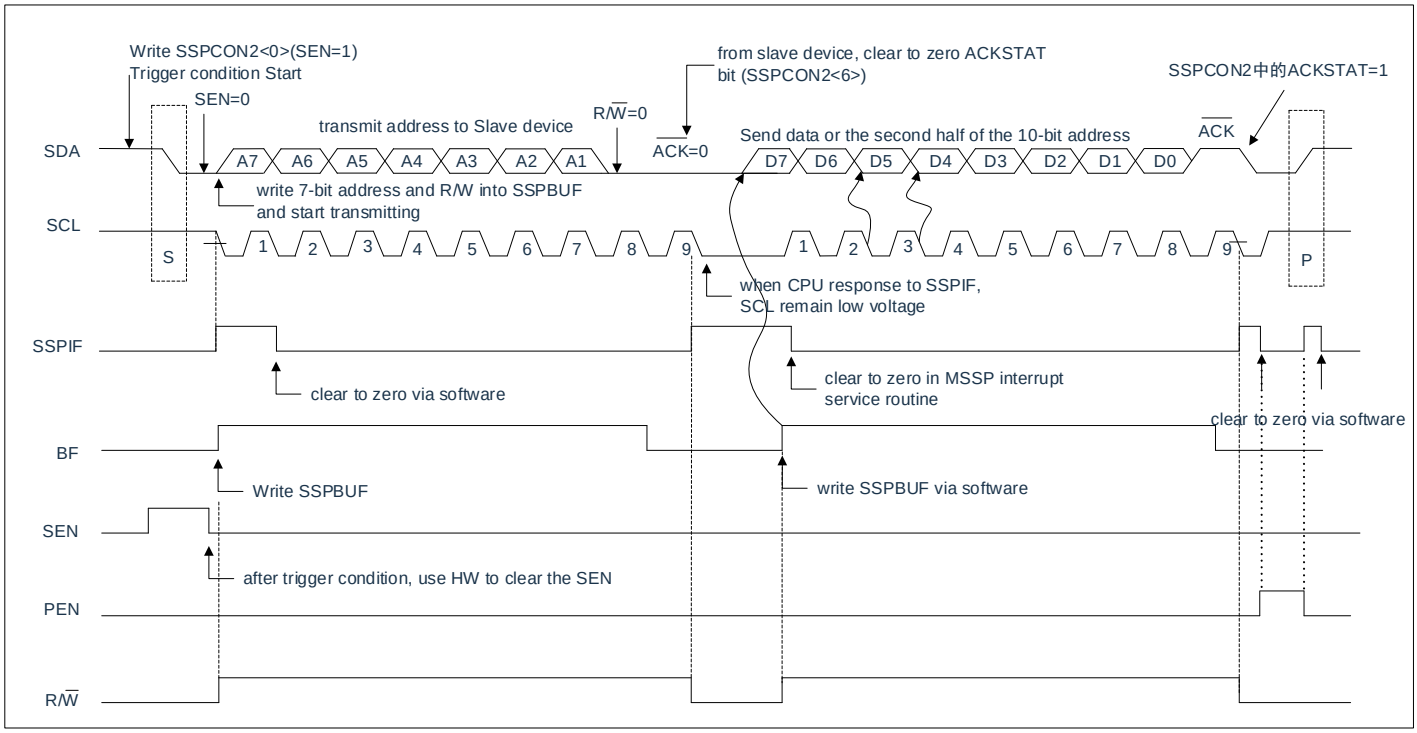


Fig14-10: time series of I²C™ master control mode transmit

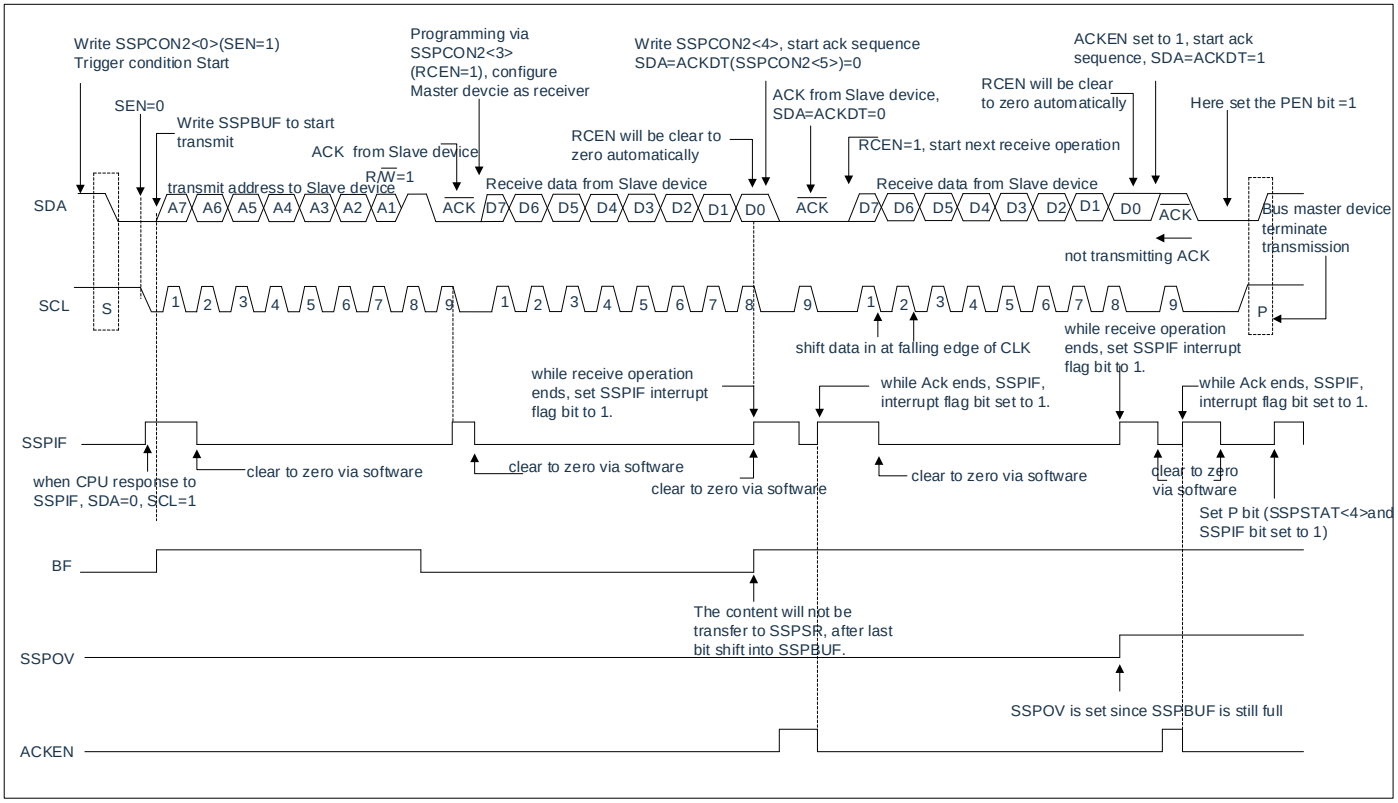


Fig 14-11: time series of I²C™ master control mode receive (7-bit address)

14.3.7 I²C Master Control Mode Start Condition Time Series

To initiate a start condition, the user should set the start condition enable bit SEN of the SSPCON2 register to 1. When both SDA and SCL pins are sampled as high, the baud rate generator reloads the contents of SSPADD<6: 0> and starts counting. When the baud rate generator timeout (TBRG) occurs, if both SCL and SDA are sampled as high level, the SDA pin is low level by the driver. When SCL is high level, setting the SDA driver to low level is the startup condition. Set the S bit (SSPSTAT register) to 1. Then the baud rate generator reloads the contents of SSPADD<6: 0> and resumes counting. When the baud rate generator times out (TBRG), the SEN bit of the SSPCON2 register will be automatically cleared by hardware. The baud rate generator is paused, the SDA line remains low, and the start condition ends.

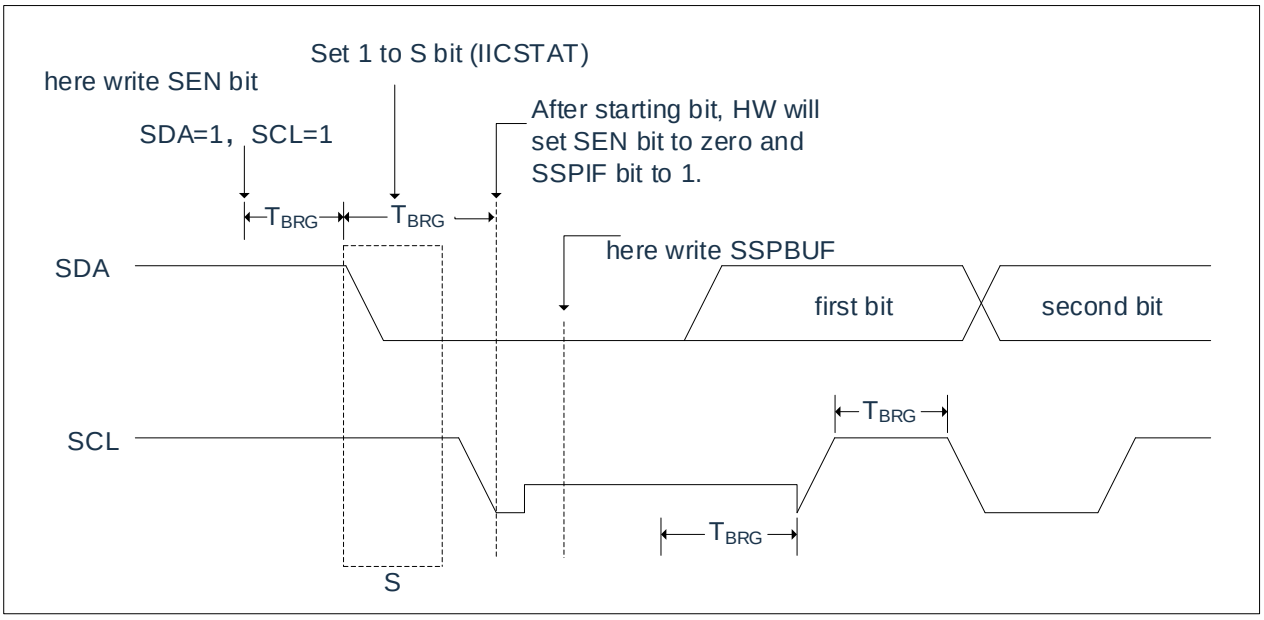


Fig 14-12: time series for the first starting bit

14.3.7.1 WCOL Status Indication

When the startup sequence is in progress, if the user writes SSPBUF, WCOL is set to 1, and the buffer content remains unchanged (no write operation has occurred).

Note: Since event queues are not allowed, the lower 5 bits of SSPCON2 cannot be written before the start condition ends.

14.3.8 I²C Master Control Mode Repeat Condition Time Series

When the RSEN bit (SSPCON2 register) is programmed to be high and the I2C logic module is in an idle state, a repeated start condition will occur. When the RSEN bit is 1, the SCL pin is pulled low. When the SCL pin is sampled low, baud rate generator loads the contents of SSPADD<6: 0> and starts counting. In a baud rate generator counting period (TBRG), the SDA pin is released (its pin level is pulled high). When baud rate generator timeout, if SDA is sampled as high, SCL pin will be pulled high. When SCL pin is sampled as high, the baud rate generator will be reloaded into the contents of SSPADD<6: 0> and start counting. SDA and SCL must be in one count period TBRG and sampled as high level. Then the SDA pin is pulled low (SDA = 0) and keeps a count period TBRG while SCL is high level. Then the RSEN bit (SSPCON2 register) will be automatically cleared, the baud rate generator will not be reloaded, and the SDA pin remains low. Once the start condition is detected on the SDA and SCL pins, the S bit (SSPSTAT register) will be set to 1. The SSPIF bit will not be set to 1 until the baud rate generator times out.

Once the SSPIF bit is set to 1, the user can write the 7-bit address into SSPBUF under 7-bit address mode or write default first address byte under 10-bit address mode. When the first 8 bits are transmitted and an ACK is received, the user can transmit another 8-bit address (in 10-bit address mode) or 8-bit data (in 7-bit address mode).

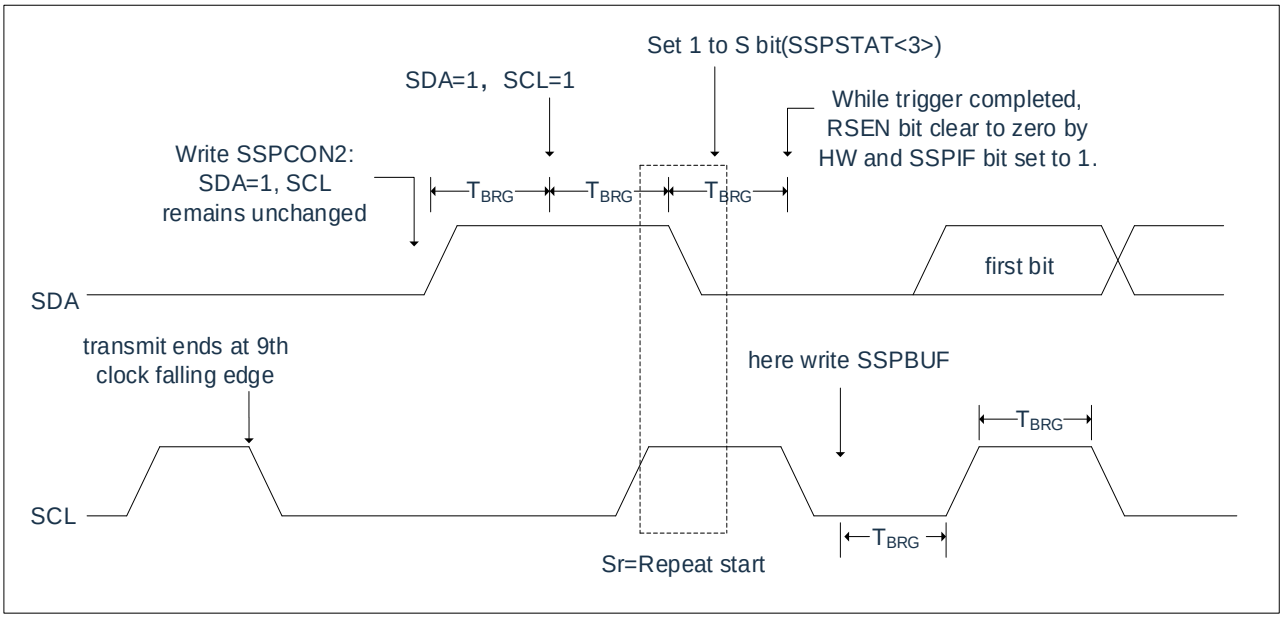


Fig 14-13: time series of repeat condition

14.3.8.1 WCOL Status Indication

When the repeated start sequence is in progress, if the user writes SSPBUF, WCOL is set to 1, and the buffer content remains unchanged (no write operation has occurred).

Note: As events are not allowed to be queued, the lower 5 bits of SSPCON2 cannot be written until the repeated start condition ends.

14.3.9 ACK Time Series

Set the ACK enable bit ACKEN (SSPCON2register) to 1 to enable the acknowledgement. When this bit is set to 1, the SCL pin is pulled low, and the content of the ACK data bit appears on the SDA pin. If the user wants to generate a response, it should clear the ACKDT bit to zero; otherwise, the user should set the ACKDT bit to 1 before the start of the ACK. Then the baud rate generator counts the full return period (TBRG), and then the SCL pin level is pulled high. When the SCL pin is sampled as at high level (clock arbitration), baud rate generator counts for another TBRG period. Then SCL pin is pulled low. After that, the ACKEN bit is automatically cleared, baud rate generator is turned off, and MSSP module enters idle mode.

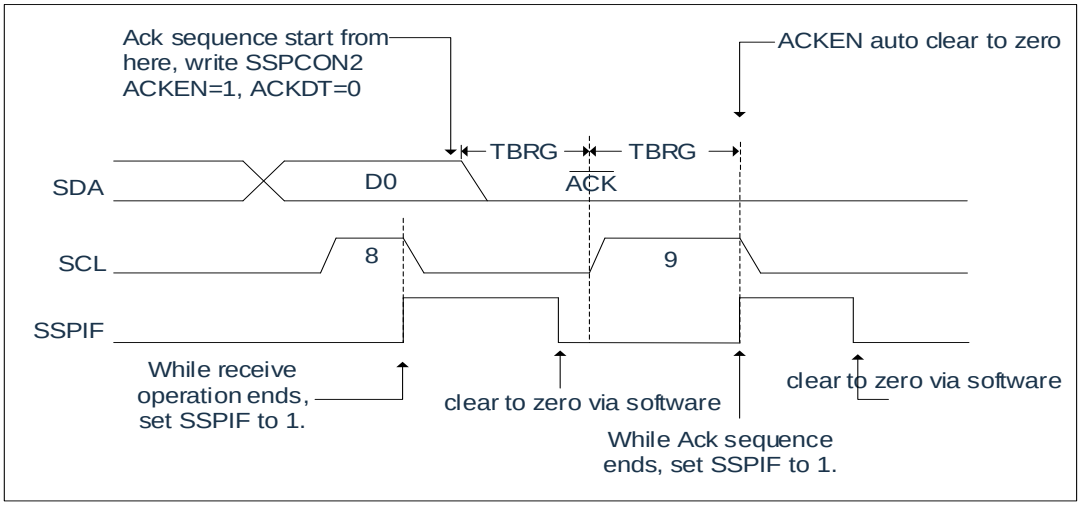


Fig 14-14: times series for ACK

Note: T_{BRG}= 1 baud rate generator period.

14.3.9.1 WCOL Status Indication bit

If the user writes SSPBUF while the ACK sequence is in progress, WCOL will be set to 1 and the contents of the buffer will remain unchanged (no write operation has occurred).

14.3.10 Stop Condition

At the end of receive/transmit, by setting the enable bit of the stop sequence, PEN (SSPCON2 register), the SDA pin will generate a stop bit. At the end of receive/transmit, the SCL pin will remain low after the falling edge of the 9th clock Level. When the PEN bit is 1, the master control device sets SDA low. When the SDA line is sampled low, the baud rate generator is reloaded with the value and counts down to 0. When the baud rate generator times out, The SCL pin is pulled to a high level, and after a TBRG (baud rate generator counts back to zero), SDA pin is pulled to a high level again. When SDA pin is sampled as high and SCL is also high, the P bit (SSPSTAT register) set to 1. After a TBRG period, the PEN bit is cleared and the SSPIF bit is set to 1.

14.3.10.1 WCOL Status Indication

If the user attempts to write SSPBUF during the stop sequence, the WCOL bit will be set to 1, and the contents of the buffer will not change (no write operation has occurred).

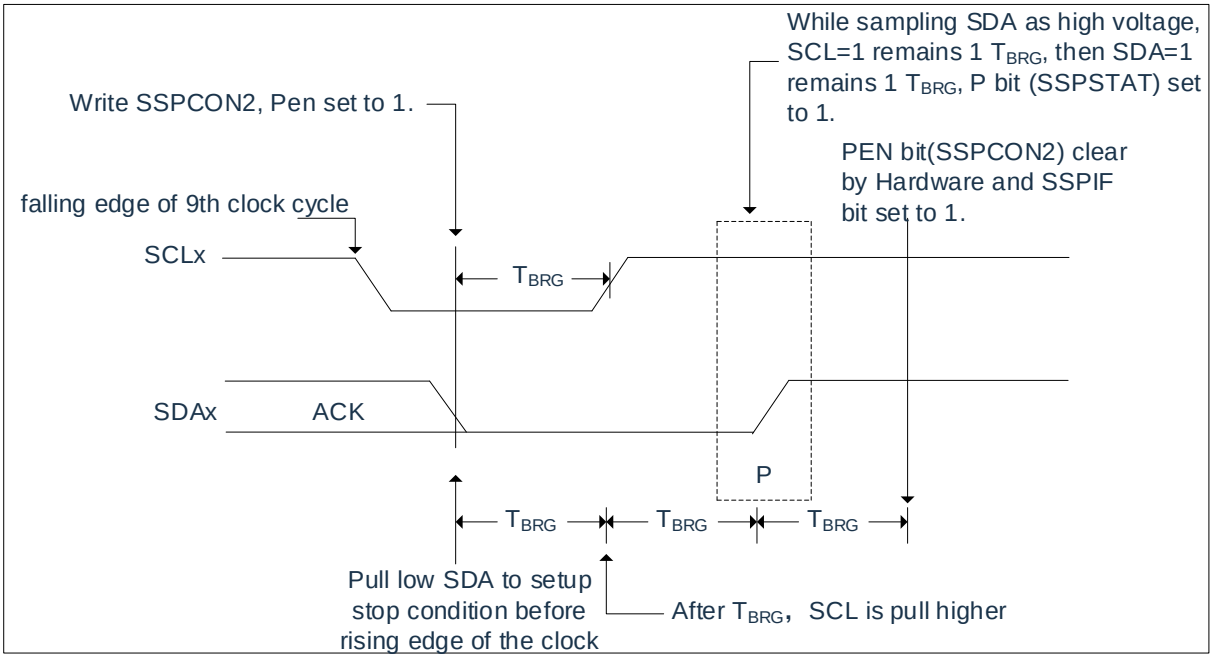


Fig14-15: stop condition receive or transmit mode

Note: T_{BRG}=1 baud rate generator period.

14.3.11 Clock Arbitration

If during any receive, transmit, or repeated start/stop conditions, the master device pulls up the SCL pin (allowing the SCL pin to float high), clock arbitration will occur. If the SCL pin is allowed to float high, the baud rate generator (BRG) will pause counting until the SC L pin is actually sampled high. When the SCL pin is sampled high, the baud rate generator will be reloaded with the contents of SSPADD<6: 0> and start counting. This can ensure that when the external device pulls the clock low, the SCL always maintains high for at least one BRG full return period.

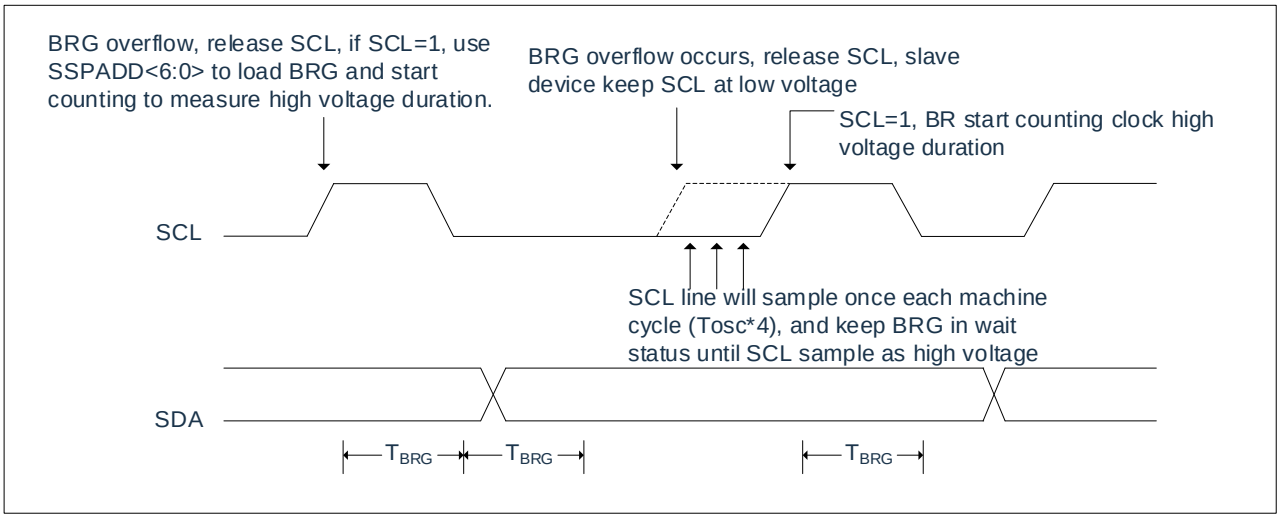


Fig 14-16: clock arbitration in master control transmit mode

14.3.12 Multi Master Mode

In multi-master mode, it can be determined when the bus is free by generating interrupt when the start and stop conditions are detected. The stop (P) bit and the start (S) bit are cleared when reset or disable MSSP module. When the P bit is set to 1, you can get control of the I^2C bus; otherwise, the bus is in an idle state, and the P and S bits are cleared. When the bus is busy, if a stop condition occurs, an interrupt will be generated (if MSSP interrupt is allowed).

When working in multi-master mode, you must monitor the SDA line for arbitration to see if the signal level is the expected output level. This check is done by hardware, and the result is placed in the BCLF bit.

Arbitration may fail under the following conditions:

- address transmission
- Start condition
- ACK conditions
- data transmission
- Repeated start condition

14.3.13 Multi Master Communication, Bus Conflict and Bus Arbitration

Multi-master mode is supported by bus arbitration. When the master device outputs the address/data bit to the SDA pin, if one master device outputs 1 on SDA by floating the SDA pin to high level, and the other master device outputs 0, bus arbitration will occur. If the expected data on the SDA pin is 1, and the data actually sampled on the SDA pin is 0, a bus conflict has occurred. The master device will set the bus conflict interrupt flag bit BCL1F to 1 and reset the I^2C port to idle state.

If a bus conflict occurs during the transmit process, the transmit stops, the BF flag bit is cleared, the SDA and SCL lines are pulled high, and SSPBUF is allowed to be written. After the bus conflict interrupt service program is executed, if the I^2C bus is free, user can resume communication by issuing a start condition. If a bus conflict occurs during the start, repeated start, stop, or response condition, the condition is aborted, the SDA and SCL lines are pulled high, and the corresponding control bit in the SSPCON2 register is cleared. After executing the bus conflict interrupt service program, if the I^2C bus is free, the user can resume communication by issuing a start condition. The master device will continue to monitor SDA and SCL pin. If a stop condition occurs, the SSPIF bit will be set to 1. No matter what bus occurs What is the progress of the transmit during conflict, writing SSPBUF will start transmitting data from the first data bit.

In multi-master mode, the interrupt can be generated when the start and stop conditions are detected to determine when the bus is free. When the P bit is set to 1, you can obtain control of the I2C bus, otherwise the bus is free, and the S and P bits are cleared.

14.3.14 Slave Mode

In slave mode, SCL pin and SDA pin must be configured as input (TRISA<6: 5> is set to 1). When needed (such as from the transmitter), the MSSP module will use output data to rewrite the input state.

When the address matches or the data transmitted after the address matches is received, the hardware will automatically generate an acknowledge (ACK) pulse, and load the data received in the SSPSR register at the time into the SSPBUF register.

As long as one of the following conditions is met, MSSP module will not generate this ACK pulse:

-The buffer full flag bit BF (SSPCON register) is 1 before the received data to be transmitted.

-Before receiving the transmitted data, the overflow flag bit SSPOV (SSPCON register) has been set 1.

In this case, the value of SSPSR register will not be loaded into SSPBUF, but the SSPIF bit of PIR1 register will be set to 1. The BF bit is cleared by reading the SSPBUF register, and the SSPOV bit is cleared by software.

To ensure normal operation, SCL clock input must meet the minimum high-level time and minimum low level time requirements.

14.3.14.1 Addressing

Once MSSP module is enabled, it will wait for the start condition to be generated. After the start condition occurs, 8 bits of data are shifted into the SSPSR register. All input bits are sampled on the rising edge of the clock (SCL) line. Register SSPSR<7: 1> The value will be compared with the value of the SSPADD register. The comparison is performed on the falling edge of the 8th clock pulse (SCL). If the address matches and the BF bit and SSPOV bit are zero, the following events will occur:

- The value of SSPSR register is loaded into SSPBUF register.
- The buffer full flag bit BF is set to 1.
- Generate ACK pulse.
- On the falling edge of the 9th SCL pulse, the MSSP interrupt flag bit SSPIF of the PIR1 register is set to 1 (interrupt is generated if interrupt is allowed). In 10-bit address mode, from MCU, 2 bytes of address need to be received, higher 5 bits of the 1st address byte will determine whether it is for 10-bit address. R/W (SSPSTAT register) must determine write operation, thus the MCU can receive the 2nd address byte. For 10-bit address, the first byte shall be 11110A9A80, while A9 and A8 are the highest 2 valid bits of the address.

10-bit address working sequence as following, while step 7-9 are specifically for slave transmitter:

1. Receive address 1st (high) byte (SSPIF bit of PIR1 register and BF and UA field of SSPSTAT register set to 1).
2. Use address 2nd (low) byte to update SSPADD register (UA bit clear and release SCL line).
3. Read SSPBUF Register (BF bit clear) and clear flag bit SSPIF to zero.
4. Receive address 2nd (low)byte (SSPIF bit、BF bit and UA bit set to 1).
5. Use address 1st (high) byte to update SSPADD register. If matching, release SCL line, and clear UA bit to zero.
6. Read SSPBUF Register (BF bit clear to zero) and set flag bit SSPIF to zero.
7. Receive repeat start condition
8. Receive address 1st (high) byte (SSPIF bit and BF bit set to 1).
9. Read SSPBUF Register (BF bit set to zero) and flag bit SSPIF to zero.

14.3.14.2 Receive

When the R/W bit of the address byte is cleared and an address match occurs, the R/W bit of the SSPSTAT register is cleared. The received address is loaded into the SSPBUF register.

When there is an address byte overflow condition, an acknowledge pulse (ACK) will not be generated. The overflow condition means that the BF bit (SSPSTAT register) is set to 1, or the SSPOV bit (SSPCON register) is set to 1. Each data transmission byte will generate an MSSP interrupt. The interrupt flag bit SSPIF of the PIR1 register must be cleared by software. The SSPSTAT register is used to determine the status of the byte.

14.3.14.3 Transmit

When the R/W bit of the received address byte is 1 and an address match occurs, the R/W bit of the SSPSTAT register is 1. The received address is loaded into the SSPBUF register. The ACK pulse is transmitted on the 9th bit while the SDA pin remains low. The transmitted data must be loaded into the SSPBUF register and also into the SSPSR register. Then the CKP bit (SSPCON register) should be set to 1 to enable the SCL pin. Before transmitting another clock pulse, the master control device must monitor the SCL pin. The slave device can suspend the data transmission with the master control device by extending the clock. 8 data bits are shifted out on the falling edge of the SCL input. This ensures that the SDA signal is valid during the SCL high level.

Each byte of data transmission will generate an MSSP interrupt. The SSPIF flag bit must be clear through software, and the SSPSTAT register is used to determine the status of the byte. The SSPIF bit is set at the falling edge of the 9th clock pulse. The ACK pulse from the main receiver will be latch on the rising edge of the 9th pulse of SCL input. If the SDA line is high (no ACK), then the data transfer has been completed. In this case, if the slave device latches the ACK, reset the slave logic (Reset SSPSTAT register), while the slave device monitors the appearance of the next start bit. If the SDA line is low (ACK), then the data to be transmitted must be loaded into the SSPBUF register, which will also load the SSPSR register. CKP should be set 1 to enable RB1/SCK/SCL.

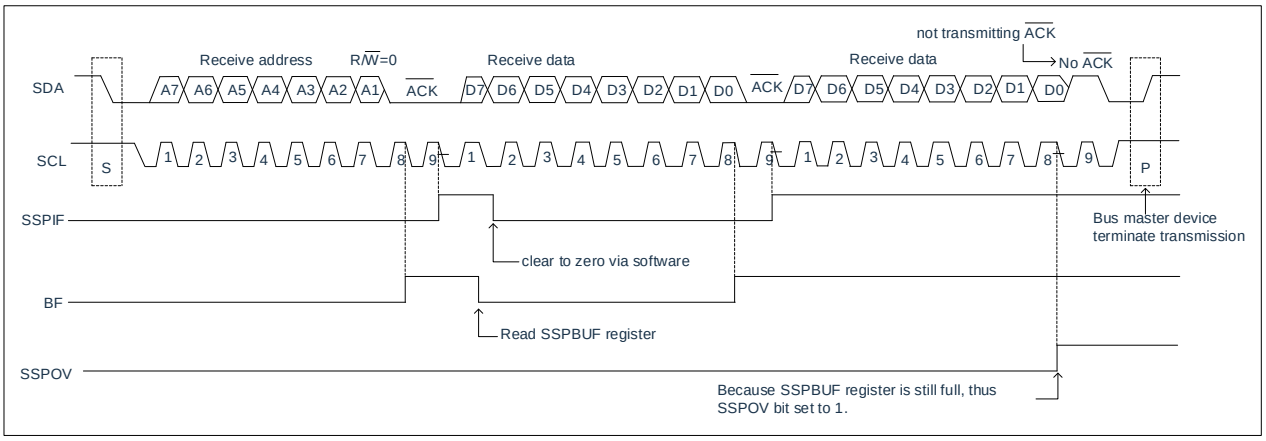


Fig 14-17: Time series for I²C™ slave mode receive (7-bit address)

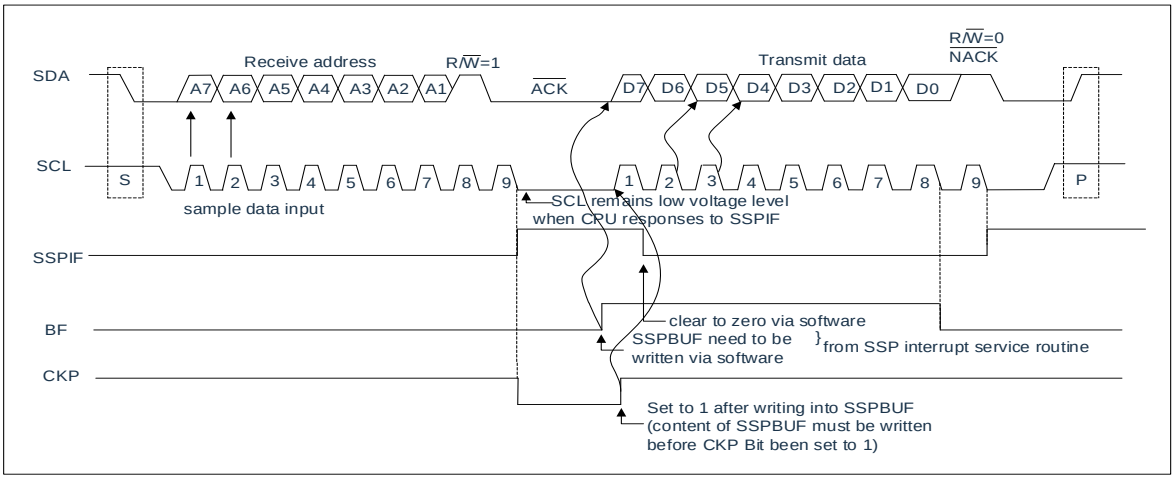


Fig 14-18: I²C™ slave mode transmit (7-bit address)

14.3.15 SSP Masking Register

In I²C slave mode, the SSP mask (SSPMSK) register is used to mask the value in the SSPSR register under the address compare operation. A bit of 0 in the SSPMSK register will make the corresponding bit in the SSPSR register a "don't care".

This register is reset to all 1s when any reset condition occurs. Therefore, it has no effect on the standard SSP operation before writing the mask value. The register must be initialized before selecting the I²C slave mode (7 bit or 10-bit address) by setting the SSPM<3: 0> bits. This register can only be accessed after the appropriate mode is selected through the SSPM<3: 0> bits of SSPCON.

The SSP masking register is valid in the following situations:

- 7-bit address mode: perform address compare with A<7: 1>.
- 10-bit address mode: only perform address compare with A<7: 0>

SSP masking is invalid during the period from receive to the first (high) byte of address.

SSPMSK: SSP masking register (191H) (1)

191H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SSPMSK	MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0 (2)
read/write	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~Bit1	MSK<7: 1>:	Mask bit.
	1=	Bit n of the received address is compared with SSPADD<n> to detect the match of the I²C address.
	0=	Bit n of the received address is not used to detect I²C address matching.
Bit 0	MSK<0>:	I²C slave mode 10-bit address mask bit (2).
	I²C slave mode 10-bit address (SSPM<3: 0> = 0111):	
		1 = Comparing Bit0 of the received address and SSPADD<0> to verify I2C address matching condition. 0 = Bit0 of the received address is not used to verify I2C address matching condition.

Note:
1) When the SSPCON bit SSPM<3: 0> = 1001, any read or write operation to the SSPADDSFR address is performed through the SSPMSK register.
2) In all other SSP modes, this bit is invalid

14.3.16 Operation under Sleep Mode

In sleep mode, I²C module can receive address or data. After address matching or byte transmission completed, it will Wake up MCU (if MSSP interrupt is enabled).

14.3.17 Effect of Reset

reset will disable MSSP module and terminate the current transmission.

15. Programmable Pulse Generator PPG

15.1 PPG Operation Principal

Specifically designed for induction cooker solution application, CMS89F52x internally integrates a programmable pulse Generator (refer as PPG in below paragraphs), this module is composed of 1 10bit Timer PPGTMR, 5 high precision comparators: synchronized comparator (COMP1), over voltage comparator (COMP2), Over voltage comparator 1 (COMP3), current surge comparator (COMP4), current voltage surge comparator (COMP5), 1 individual Watchdog Timer PPGWDT.

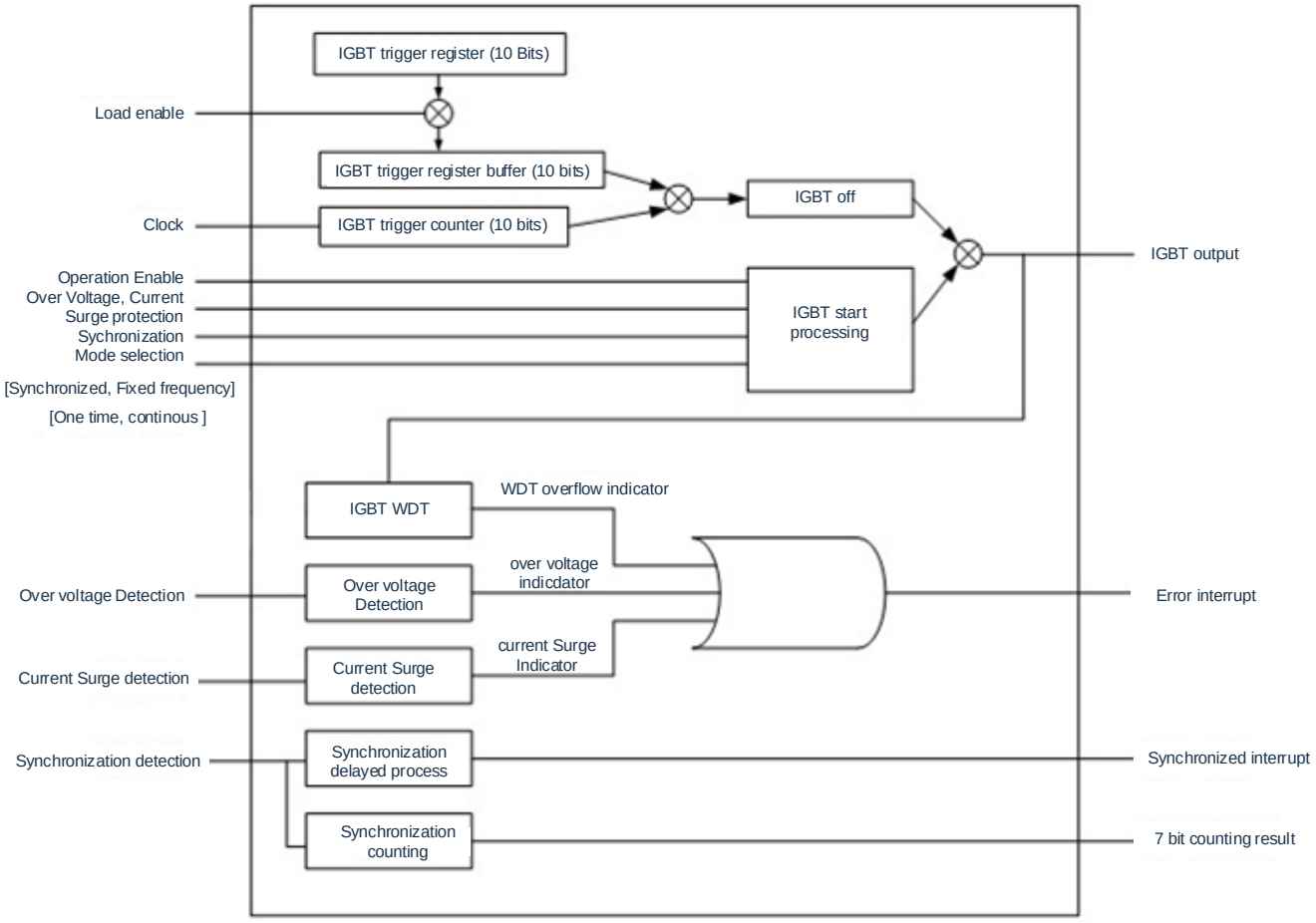


Figure 15-1: PPG Operation Principal

PPG Output Signal is One which only Output Low Voltage Level or high impedance, when PPG function turns off, it is set to high impedance.

PPGTMR is a 10-bit Timer, the lower 8 bits are stored in register PPGTMRL (14H) , while higher 2 bits are stored in register PPGTMRH (15H). When PPG Output is turned off, internally 10 bits counter is clear to zero. When PPG Output is turned on, counter starts counting, each oscillation cycle automatically increment by 1, when counter added to equal to PPGTMR value, PPG Output automatically turns off, counter will be clear to zero.

PPGTMR lower 8 bits register PPGTMRL (14H)

14H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PPGTMRL	PPGTMR lower 8 bits							
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

PPGTMR higher 2 bits PPGTMRH (15H)

15H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PPGTMRH	----	----	----	----	----	----	PPGTMR higher 2 bits	
R/W	----	----	----	----	----	----	R/W	R/W
Reset value	----	----	----	----	----	----	0	0

15.2 Related Pins of PPG

There are 6 pins related to PPG, as shown in below table:

Pin name	IO Type	Pin Description
C1N	I	Comparator1 Negative Input
C1P C2N	I	Comparator1 Positive Input Comparator2 Negative Input
C3N	I	Comparator3 Negative Input
C4N	I	Comparator4 Negative Input
C5N	I	Comparator5 Negative Input
PPG_OUT	O	PPG Output

15.3 PPG Operation Mode

PPG Module of CMS89F52x has 2 operational modes, they are:

- Single Output mode.
- Synchronized Output mode.

Control register related to PPG status is as following:

PPG Control Register PPGCON (17H)

17H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PPGCON	DETC5F	DETC4F	----	RELOAD_EN	DETC5EN	DETC4EN	PPGMD	PPG_ON
R/W	R/W	R/W	----	R/W	R/W	R/W	R/W	R/W
Reset Value	1	1	----	0	0	0	0	0

Bit7 DETC5F: Comparator5 Status Bit (PPG Status Bit);
0: Clear Comparator5 的 0->1 Flip flag, (if DETC5EN=1, then PPG re-open);
1: Has Comparator5 的 0->1 Flip, invalid while writing 1 (if DETC5EN=1, then PPG turn off).

Bit6 DETC4F: Comparator4 Status Bit (PPG Status Bit);
0: Clear Comparator4 的 0->1 Flip flag, (if DETC4EN=1, then PPG re-open);
1: Has Comparator4 的 0->1 Flip, invalid while writing 1 (if DETC4EN=1, then PPG turn off).

Bit5 Reserved

Bit4 RELOAD_EN: PPG TMR upload Enable;
0: Enable upload (in auto power saving mode, and allows PPG TMR auto -1);
1: Disable upload (in auto power saving mode, and dis-allows PPG TMR auto -1);

Bit3 DETC5EN: Comparator5 turn off PPG Enable bit;
0: Disable;
1: Enable.

Bit2 DETC4EN: Comparator4 turn off PPG Enable bit;
0: Disable;
1: Enable.

Bit1 PPGMD: PPG Output mode;
0: According to Comparator1 Synchronized Output;
1: Single Output.

Bit0 PPG_ON: PPG Output Enable Bit;
0: Disable;
1: Enable.

15.3.1 Single Output Mode

When system register PPGCON 1st bit is set to “1”, PPG will be in Single output operational mode. When set PPG Enable bit (PPGCON.0) to 1, then after PPG outputs 1 PPGTMR period low voltage level, it will be reconfigured as high impedance, and PPG Enable bit will be automatically clear to zero. PPG will stop operation. Single output Mode in normal situation will be used to detect whether inductor cooker has pan on it.

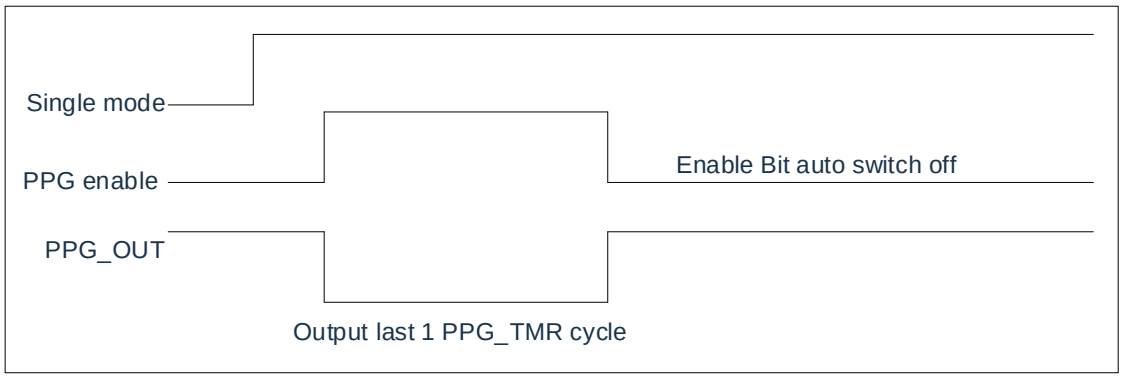


Figure 15-2: PPG Single Output mode timing sequence

15.3.2 Synchronized Output Mode

Synchronized Output refers to PPG Output is synchronized with Comparator1 flips. When System register 1st bit of PPGCON is set to “0”, PPG module will be in synchronized Output mode. In this mode, when Enable signal changed from “0” to “1”, after PPG Outputs 1 PPGTMR period of low voltage level it will turn off Output, then it will automatically continue outputting based on flips of Comparator1 output from “1” to “0”.

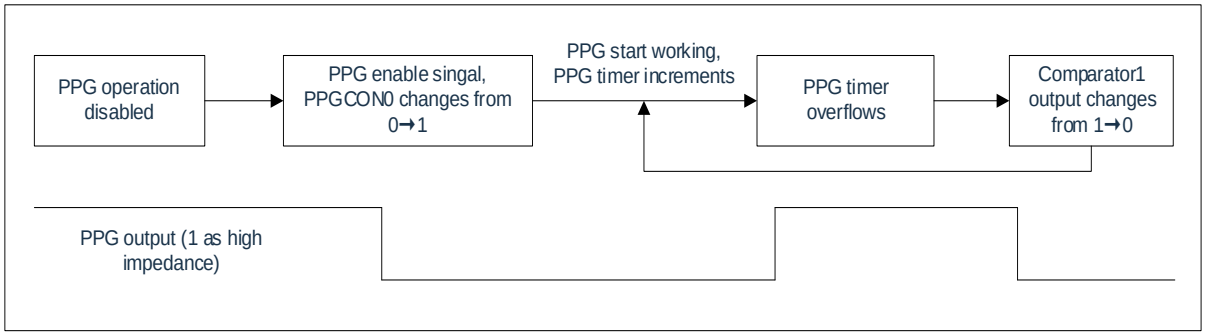


Figure 15-3: PPG Synchronized Output mode

15.4 Comparator

PPG Module has in total 5 Comparators: synchronized Comparator (COMP1), over voltage Comparator (COMP2), over voltage Comparator 1 (COMP3), Current surge Comparator (COMP4), Voltage Surge comparator (COMP5).

15.4.1 Synchronized Comparator COMP1

Synchronized Comparator is used to provide Synchronization Signal to PPG. When PPG operation in Synchronized mode, PPG_OUT will output Low voltage level only when the Comparator output changes from “1” to “0”, after last 1 PPGTMR period then turns off, wait for next flip of the Comparator, and again output the low voltage level.

It can be configured that the PPG is enabled only after a period of delay time after the synchronized Comparator flips, the minimal time is 0*Tsys, the longest is 64*Tsys, this time is determined by lower 4 bits of PPGDLY.

Comparator1 Control register CM1CON (97H)

97H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM1CON	CM1EN	CM1COFM	CM1CEN	CM1CLR	CM1NSL	----	----	----
R/W	R/W	R/W	R/W	R/W	R/W	----	----	----
Reset Value	0	0	0	0	0	----	----	----

Bit7 CM1EN: Comparator1Enable 位;
0: Disable, Comparator 不 Operation , Output0;
1: Enable, COMP1+、COMP1-as Comparator Input port

Bit6 CM1COFM: Comparator1 Adjust mode selection;
0: Normal Operation mode;
1: Adjusted mode.

Bit5 CM1CEN: Comparator1 Flip count Enable;
0: Disable counting.
1: Enable counting.

Bit4 CM1CLR: Comparator1 counting clear to zero;
0: Clear to zero;
1: Normal counting

Bit3 CM1NSL: Comparator1Negative internal grounding selection, only valid in adjust mode;
0: COMP1-connect I/O port (if CM1EN=1, EnableC1N channel port as Comparator1Negative Input);
1: COMP1-connect GND (C1N channel port as normal I/O port)

Bit2~Bit0 Reserved

PPG Delay timing control PPGDLY (16H)

16H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PPGDLY	----	----	----	----	PPGDLY			
R/W	----	----	----	----	R/W	R/W	R/W	R/W
Reset Value	----	----	----	----	0	0	0	0

Bit7~Bit4 Reserved

Bit3~Bit0 PPGDLY: PPG Delay Output;

0000: No delay;

0001: 4-5*Tsys;

0010: 8-9*Tsys;

….

1111: 64-65*Tsys.

Synchronized Comparator has flip counting function, it can be used to record the number of flips of its output. In order to make it effective, the function must set 5th bit of CM1CON to 1. While PPG counting function starts, when synchronized comparator output change from “1” to “0”, the counter will automatically increase by 1, max till 128. The counting result will be stored to CM1CNT register, the lower 7 bits represents counting number, the max bit represents whether the counting overflow occurs. When counting exceeds 128, the max bit will be 1, counting stops. The setting ‘0’ of 4th bit of CM1CON can be used to clear the counter to zero.

Comparator1 Flip counting register CM1CNT (93H)

93H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM1CNT	CM1OF	CM1COUNT [6: 0]						
R/W	R	R	R	R	R	R	R	R
Reset value	0	0	0	0	0	0	0	0

Bit7 CM1OF: Comparator1 Flip counting overflow flag;

0: No overflow;

1: Overflow.

Bit6~Bit0 CM1COUNT [6: 0]: Comparator1’s 1->0 flip counter, read only.

15.4.2 Over Voltage Comparator COMP2 and Surge Comparator COMP4/COMP5

Over voltage Comparator and Surge Comparator can both be used to constraint the PPG output, so to protect IGBT. The negative signal of Over voltage Comparator comes from RB2 Port input, The negative signal of voltage surge Comparator comes from RB0 port input, The negative signal of current surge Comparator comes from RB4 port input. Their positive signals are all from Chip internal, and software can be used to adjust resistor divided voltage.

When Negative voltage of the Over voltage Comparator is changed from the level lower than Positive Voltage to higher than Positive voltage, we call it IGBT over voltage. When Over voltage count meets the numbers configured by software, PPG will disable Output or reduce Output Time (via Software configuration), and generate interrupt flag, the interrupt flag bit must be clear to zero, before PPG can resume to work normally.

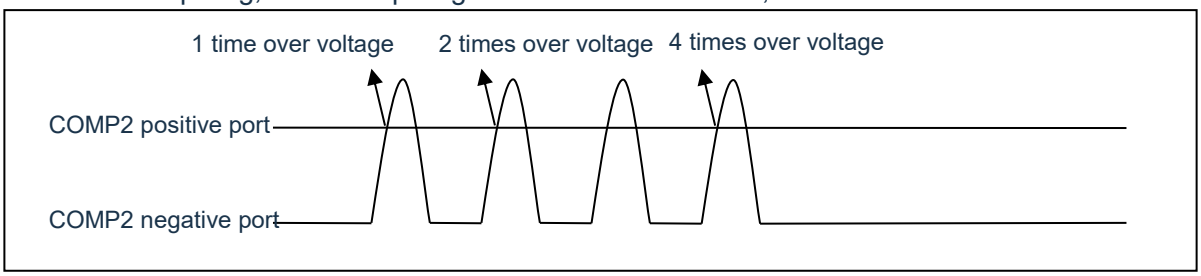


Figure 15-4: Over voltage Comparator COMP2

When Surge Comparator meets the valid voltage level as configured, and timing wise satisfied software configured time, PPG will disable output, and generate interrupt flag, the interrupt flag must be clear to zero, PPG can be in normal operation. The positive voltage of Surge Comparator can be selected to connect to Ground.

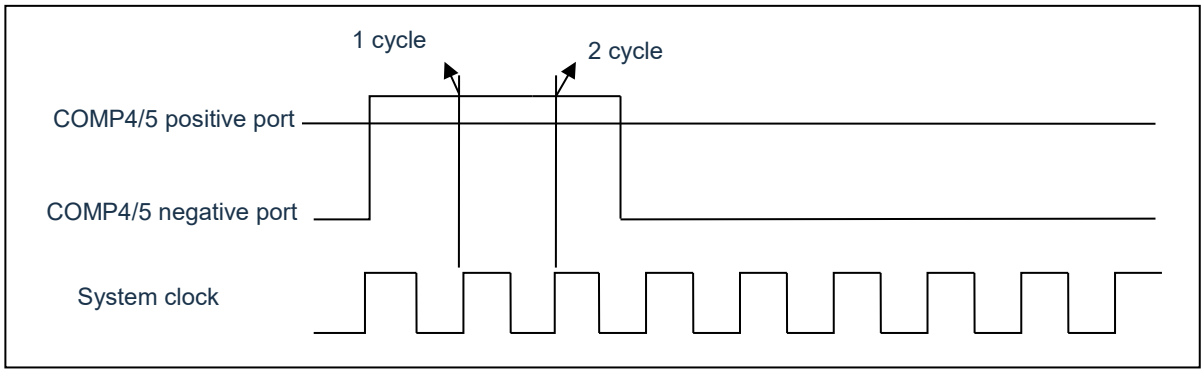


Figure15-5: OutputLow Voltage Level Validity Diagram

Related Register of COMP2, COMP4, COMP5:

Comparator2 Control Register CM2CON (98H)

98H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM2CON	CM2EN	CM2COFM	CM2DBSEL		CM2PVSL			
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 CM2EN: Comparator2 Enable bit;
0: Disable, Comparator2 not enabled , Output0;
1: Enable, Comparator2 counter overflow will affect PPG or entering into interrupt
Bit6 CM2COFM: Comparator2 Adjustment Mode selection;
0: Normal Operation mode;
1: Adjustment mode
Bit5~Bit4 CM2DBSEL: Comparator2 filter timing selection;
00: <=1Tsys (pulse width which can be filtered);
01: <=4Tsys;
10: <=8Tsys;
11: <=16Tsys.
Bit3~Bit0 CM2PVSL: Comparator2 Internal Positive Voltage Selection.
0000-1111: 0.4VDD-0.775VDD (Total 16 stages, each stage is 0.025VDD).

Comparator2 Control register 1CM2CON1 (99H)

99H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM2CON1	ATPEN	----	----	----	CM2COF	CM2CNT		
R/W	R/W	----	----	----	R/W	R/W	R/W	R/W
Reset value	0	----	----	----	1	0	0	0

Bit7 ATPEN: Comparator2 Count overflow auto reduce PPG_TMR Enable bit;
0: Disable, (after Comparator2 Counting clear to zero, CM2COF must be clear before start counting);
1: Enable, every time when overflow is detected, value of PPT_TMR automatically reduce by 1
(Comparator2 After Counter clear to zero automatically restart counting).
Bit6~Bit4 Reserved
Bit3 CM2COF: Comparator2 counter overflow flag, can clear to zero via software.
0: Comparator2 counter no overflow, write 0 to clear, (if ATPEN=0, Comparator2 Counter Enable Count);
1: Comparator2 Counter overflows, write 1 invalid, (if ATPEN=0, Then Comparator2 Count remains zero state).
Bit2~Bit0 CM2COS [2: 0]: Comparator2 Counter overflow requires number of pulse selection (Comparator2Output1->0 change will trigger counting Input);
000: Overflow after 1 time, or generate interrupt, and counter clear to zero:
001: 2 times;
010: 4 times;
...
111: 128 times.

Comparator4 Control register CM4CON (Current Surge) (9CH)

9CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM4CON	CM4EN	CM4COFM	CM4DBSEL		CM4PVSL			
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 CM4EN: Comparator4Enable 位 (Output high valid);
0: Disable, Comparator4 not in Operation , Output0;
1: Enable, Comparator4's 0->1 flip will affect PPG or enter into interrupt.
Bit6 CM4COFM: Comparator4 Adjust mode selection.
0: Normal Operation mode;
1: Adjustment mode.
Bit5~Bit4 CM4DBSEL: Comparator4 Filter timing selection;
00: <=1Tsys (pulse width which can be filtered);
01: <=4Tsys;
10: <=8Tsys;
11: <=16Tsys.
Bit3~Bit0 CM4PVSL: Comparator4 internal Positive Voltage Selection;
0000: Select Internal connection to GND;
0001-1111: 0.050VDD-0.400VDD (Total 16 stages, each stage is 0.025VDD)

Comparator5 Control register CM5CON (Voltage Surge) (9DH)

9DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM5CON	CM5EN	CM5COFM	CM5DBSEL		CM5PVSL			
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 CM5EN: Comparator5Enable bit (Output high valid);
0: Disable, Comparator5 not in Operation , Output0;
1: Enable, Comparator5's 0->1 flip will affect PPG or enter into interrupt.
Bit6 CM5COFM: Comparator5 Adjust mode selection.
0: Normal Operation mode;
1: Adjustment mode.
Bit5~Bit4 CM5DBSEL: Comparator5 Filter timing selection;
00: <=1Tsys (pulse width which can be filtered);
01: <=4Tsys;
10: <=8Tsys;
11: <=16Tsys.
Bit3~Bit0 CM5PVSL: Comparator5 internal Positive Voltage Selection;
0000-1111: Select Internal connection to GND;
0.050VDD-0.400VDD (Total 16 stages, each stage is 0.025VDD)

15.4.3 Over Voltage Comparator1- COMP3

Over voltage Comparator1 Negative signal from RB3 Port Input, Positive is from Chip Internal, the resistor divided voltage can be adjusted via software.

Related Register of COMP3:

Comparator3 Control Register CM3CON (9AH)

9AH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM3CON	CM3EN	CM3COFM	CM3DBSEL		CM3PVSL			
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 CM3EN: Comparator5Enable 位 (Output high valid);
0: Disable, Comparator5 not in Operation, Output0;
1: Enable, Comparator5's 0->1 flip will affect PPG or enter into interrupt.
Bit6 CM3COFM: Comparator5 Adjust mode selection.
0: Normal Operation mode;
1: Adjustment mode.
Bit5~Bit4 CM3DBSEL Comparator5 Filter timing selection;
00: <=1Tsys (pulse width which can be filtered);
01: <=4Tsys;
10: <=8Tsys;
11: <=16Tsys.
Bit3~Bit0 CM5PVSL: Comparator5 internal Positive Voltage Selection;
00001-1111: 0.4VDD-0.775VDD (Total 16 stages, each stage is 0.025VDD).

Comparator3 Control Register CM3CON1 (9BH)

9BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM3CON1	CM3M1	CM3M0	----	CM3CIS	CM3COF	CM3COS [2: 0]		
R/W	R/W	R/W	----	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	----	0	1	0	0	0

Bit7~Bit6	CM3M1-CM3M0:	Comparator3 Function selection;
	00:	Comparator3 does not affect PPG;
	01:	Comparator3 turn off PPG (Output low valid);
	10:	Comparator3 Reduce PPG_TMR (Sharing with Comparator2 to reduce PPG_TMR Module);
	11:	Comparator3 does not affect PPG.
Bit5	Reserved	
Bit4	CM3CIS:	Comparator3 Counter Trigger edge selection bit;
	0:	ComparatorOutput1->0 switching as counter trigger Input;
	1:	ComparatorOutput0->1 switching as counter trigger Input.
Bit3	CM3COF:	Comparator3 Counter overflow flag bit, can be clear to zero via software.
	0:	Comparator3 Counter no overflow, write 0 to clear, (if ATPEN=0, Comparator3 counter start to count);
	1:	Comparator3 counter overflows, write 1 invalid, (if ATPEN=0, then Comparator3 counter remains zero state)
Bit2~Bit0	CM3COS [2: 0]:	Comparator3 Counter overflow requires number of pulse selection (Counter trigger Input edge determined by CM3CIS);
	000:	Overflow after 1 time, or generate interrupt, and counter clear to zero:
	001:	2 times;
	010:	4 times;
	...	
	111:	128 times.

15.4.4 Comparator Calibration

Due to manufacture process deviation, in actual usage, the chip might have non-negligible Comparator out-of-balance voltage. In order to address this issue, CMS69F52x has integrated Comparator calibration function. The relevant RAM is as following:

Comparator1 Calibration Register CM1ADJ (113H)

113H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM1ADJ	CM1OUT	CM1CRS	CM1ADJ [5: 0]					
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	1	0	0	0	0	0

Bit7 CM1OUT: Comparator1Output, read only, write operation will not have impact;
0: ComparatorOutput0;
1: ComparatorOutput1.
Bit6 CM1CRS: Adjust mode Input port selection;
0: Negative Input;
1: Positive Input.
Bit5~Bit0 CM1ADJ [5: 0]: Comparator1 out-of-balance Voltage adjust;
000000: Adjust Negative to MIN (in normal circumsumce out of balance Voltage is positive);
111111: Adjust Positive to MIN (in normal circumsumce out of balance Voltage is negative).

Comparator2 Calibration Register CM2ADJ (114H)

114H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM2ADJ	CM2OUT	CM2CRS	CM2ADJ [5: 0]					
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset Value	0	0	1	0	0	0	0	0

Bit7 CM2OUT: Comparator2Output, read only, write operation will not have impact;
0: ComparatorOutput0;
1: ComparatorOutput1.
Bit6 CM2CRS: Adjust mode Input port selection;
0: Negative Input;
1: Positive Input.
Bit5~Bit0 CM2ADJ [5: 0]: Comparator2 out-of-balance Voltage adjust;
000000: Adjust Negative to MIN (in normal circumsumce out of balance Voltage is positive);
111111: Adjust Positive to MIN (in normal circumsumce out of balance Voltage is negative).

Comparator3 Calibration Register CM3ADJ (115H)

115H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM3ADJ	CM3OUT	CM3CRS	CM3ADJ [5: 0]					
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	1	0	0	0	0	0

Bit7 CM3OUT: Comparator3Output, read only, write operation will not have impact;
0: ComparatorOutput0;
1: ComparatorOutput1.
Bit6 CM3CRS: Adjust mode Input port selection;
0: Negative Input;
1: Positive Input.
Bit5~Bit0 CM3ADJ [5: 0]: Comparator3 out-of-balance Voltage adjust;
000000: Adjust Negative to MIN (in normal circumsumce out of balance Voltage is positive);
111111: Adjust Positive to MIN (in normal circumsumce out of balance Voltage is negative).

Comparator4 Calibration Register CM4ADJ (116H)

116H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM4ADJ	CM4OUT	CM4CRS	CM4ADJ [5: 0]					
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	1	0	0	0	0	0

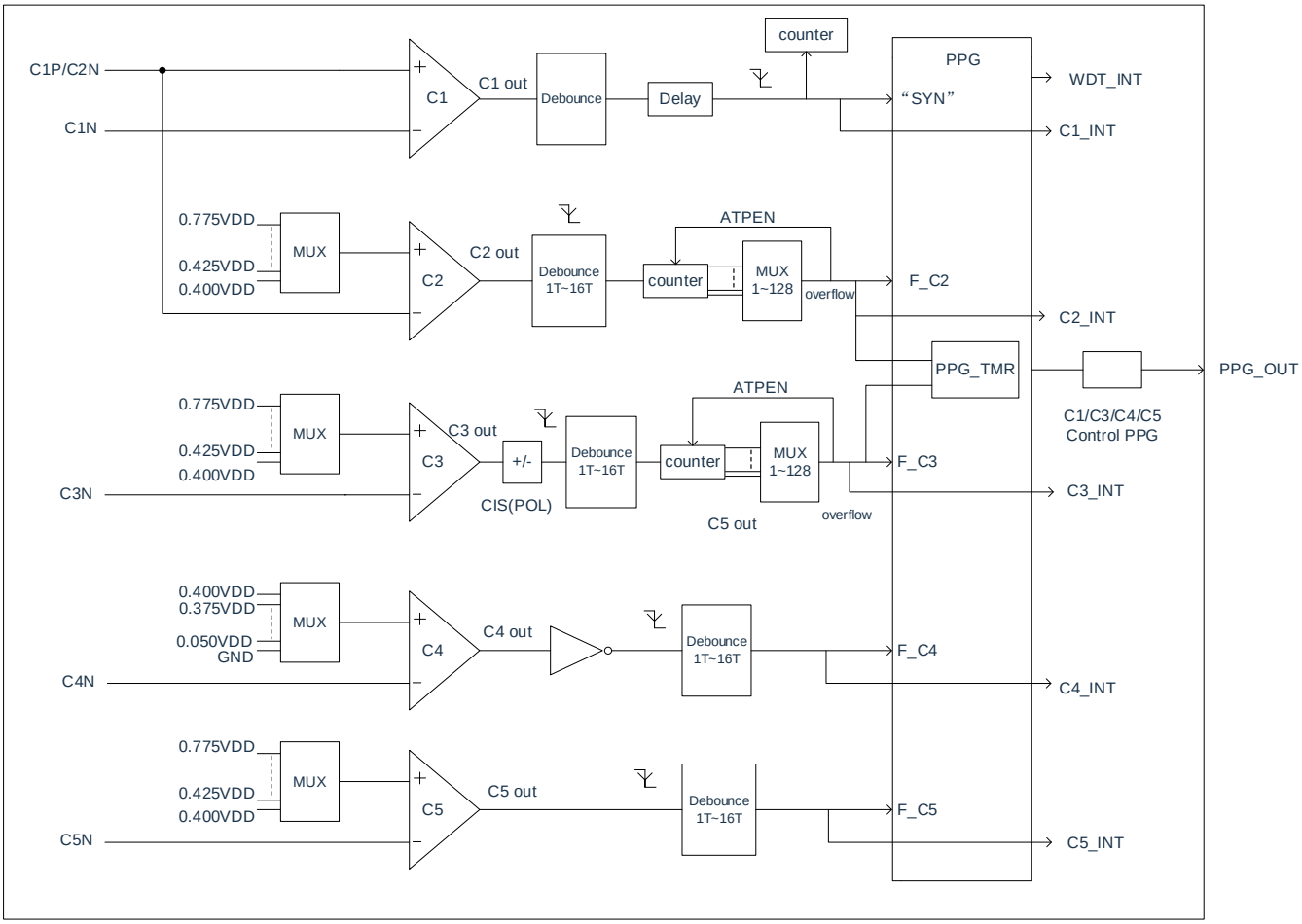
Bit7 CM4OUT: Comparator4Output, read only, write operation will not have impact;
0: ComparatorOutput0;
1: ComparatorOutput1.
Bit6 CM4CRS: Adjust mode Input port selection;
0: Negative Input;
1: Positive Input.
Bit5~Bit0 CM4ADJ [5: 0]: Comparator4 out-of-balance Voltage adjust;
000000: Adjust Negative to MIN (in normal circumsumce out of balance Voltage is positive);
111111: Adjust Positive to MIN (in normal circumsumce out of balance Voltage is negative).

Comparator5 Calibration Register CM5ADJ (117H)

117H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CM5ADJ	CM5OUT	CM5CRS	CM5ADJ [5: 0]					
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	1	0	0	0	0	0

Bit7 CM5OUT: Comparator5Output, read only, write operation will not have impact;
0: ComparatorOutput0;
1: ComparatorOutput1.
Bit6 CM5CRS: Adjust mode Input port selection;
0: Negative Input;
1: Positive Input.
Bit5~Bit0 CM5ADJ [5: 0]: Comparator5 out-of-balance Voltage adjust;
000000: Adjust Negative to MIN (in normal circumsumce out of balance Voltage is positive);
111111: Adjust Positive to MIN (in normal circumsumce out of balance Voltage is negative).

15.4.5 Comparator and PPG internal structure diagram



16. Data EEPROM Control

16.1 Data EEPROM Overview

Data EEPROM under normal working status is readable/writable. These memories are not directly mapped to the register file space, but indirectly addressed through the special function register (SFR). A total of 5 SFR registers are used to access these memories:

- EECON1
- EECON2
- EEDAT
- EEDATH
- EEADR

When accessing the data memory module of the device interface, the EEDAT and EEDATH register form a double byte word to save the 16-bit data to be read/write, and the EEADR register is used to store the address of EEDAT unit under access. The device of this series has 32-byte data EEPROM, address range is from 0H to 01FH.

EEPROM data memory allows byte to read/write. Byte write operation can automatically erase the target unit and write the new data into it (Erase before Write in).

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 data EEPROM. When the code is protected, the device programmer will no longer be able to access the data.

16.2 Related Register

16.2.1 EEADR Register

The EEADR register can address up to 32 bytes of data EEPROM.

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.

16.2.2 EECON1 and EECON2 Register

EECON1 is the control register to access the EEPROM.

The control bit EEPGD must be set to 1 in order to operate data EEPROM. When this bit is cleared, as with reset, any subsequent operations are invalid.

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 data 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 PIR2 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 data EEPROM write sequence.

EEPROM data register EEDAT (10CH)

10CH	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	0	0	0	0	0	0	0	0

Bit7~Bit0 EEDAT<7: 0>: The lower 8 bits of data to read or write from the data EEPROM.

EEPROM address register EEADR (10DH)

10DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEADR	----	----	----	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0
read/write	----	----	----	R/W	R/W	R/W	R/W	R/W
Reset value	----	----	----	0	0	0	0	0

Bit7~Bit5 ---- (not related)

Bit4~Bit0 EEADR<4: 0>: Specify the lower 8 bits of address for EEPROM read/write operations.

EEPROM data register EEDATH (10EH)

10EH	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	0	0	0	0	0	0	0	0

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

EEPROM control register EECON1 (18CH)

11BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EECON1	EEPGD	----	----	----	WRERR	WREN	WR	RD
read/write	R/W	----	----	----	R/W	R/W	R/W	R/W
Reset value	0	----	----	----	x	0	0	0

Bit7 EEPGD: Data EEPROM enable bit;
1= Enable operation of data EEPROM;
0= Disable operation of data EEPROM;
Bit6~Bit4 not used Read as 0.
Bit3 WRERR: EEPROM error flag bit;
1= Write early abortion (any WDT reset or undervoltage reset during normal operation)
0= Write complete.
Bit2 WREN: EEPROM write enable bit;
1= Enable write period;
0= Disable write data EEPROM.
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= Data EEPROM 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.

EEPROM Control register EECON1 (18DH)

18DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EECON2	---							
Read/write	W							

EECON2 is not physical register. Reading EECON2 will return all 0.

EECON2 register is only used while executing data EEPROM write sequence.

16.3 Read Data EEPROM Storage

To read the data EEPROM units, the user must write the address to the EEADR register, set the EEPGD control bit of the EECON1 register to 1, and then set the control bit RD to 1. Once the read control bit is set, the data storage 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 data will appear in EEDAT register. 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 data EEPROM

```
        LD      A, RADDR        ; Put the address to be read into the EEADR register
        LD      EEADR, A
        CLRB    EECON1, EEPGD   ; enable data EEPROM
        SETB    EECON1, RD      ; enable read signal
        NOP                     ; here read data, must add NOP instruction
        NOP
        LD      A, EEDAT        ; read and load data to ACC
```

16.4 Write Data EEPROM Storage

To write a data EEPROM storage unit, the user should first write the unit's address to the EEADR register and write data to the EEDATA 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.

After Executed SETB EECON1, WR instructions, the processor needs 2 instruction cycles to configure Erase/Write operation. The user must insert 2 NOP instructions after the instruction which set WR bit to 1. After executing write instructions, the processor will pause internal operation for 4ms (selectable). Because the clock and peripherals are still operating, this is not considered sleep mode. After write period completed, processor will resume it working status after 3rd instructions after the write instruction.

write data EEPROM storage

```
LD      A, ADDR          ; write address
LD      EEADR, A
LD      A, DATAL         ; write data
LD      EEDAT, A
LD      A, DATAH
LD      EEDATH, A
SETB    EECON1, EEPGD    ; enable operation EEPROM
SETB    EECON1, WREN     ; enable write signal
CLRB    INTCON, GIE      ; turn off interrupt
SZB     INTCON, GIE      ; confirm interrupt turned off
JP      $-2

LDIA    055H             ; write 55H and 0AAH to EECON2 register
LD      EECON2, A
LDIA    0AAH
LD      EECON2, A
SETB    EECON1, WR       ; start to write program memory
NOP                      ; write buffer needs delay
NOP

CLRB    EECON1, WREN
SETB    INTCON, GIE      ; enable interrupt
SZB     EECON1, WR       ; Judge whether write operation is completed,
                         during write operation, WREN bit must remain as
                         1
JP      $-1
CLRB    EECON1, WREN     ; write completed, turn off write enable bit
```

16.5 Precautions on EEPROM Operation

16.5.1 Write Verification

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

16.5.2 Protection to Avoid Writing Wrongly

In some cases, the user may not want to write data to the data EEPROM storage. 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

17. Operational Amplifier (OPA)

Chip has built-in one channel Operational Amplifier.

17.1 Operational Amplifier Introduction

CMS89F52x has built-in an Operational Amplifier, when its positive or negative can be configured by software to connect to ground directly or connect to ground via a resistor, Output can be configured using program via Internal RC filter to connect to AN9 or directly connect to AN7 corresponding AD conversion channel for detection. The principal is as following diagram:

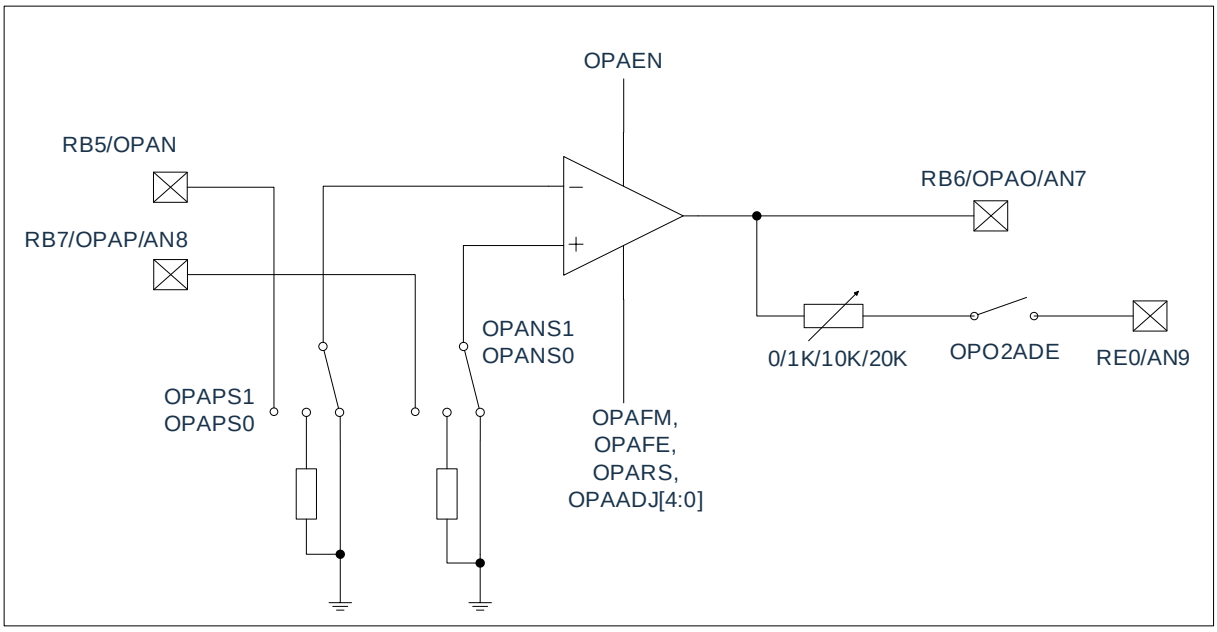


Figure 17-1: Operational Amplifier Operation Principal

Related Pin Description

Pin Name	IO Type	Pin Description
OPAN	I	Op-Amp Negative Input
OPAP	I	Op-Amp Positive Input
OPAO	O	Op-Amp Output
AN7	I	Op-Amp can connect to the AD channel port internally
AN9	I	Op-Amp can connect to the AD channel port internally, can use external capacitor for filtering connected to this port

17.2 Related Register of Operational Amplifier

There are 3 registers which is related to Op-Amp, they are OPACON, OPACON1 and OPAADJ.

Op-AMP control register OPACON (108H)

108H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPACON	OPAEN	OPAFM	OPAFE	----	OPAPS1	OPAPS0	OPANS1	OPANS0
R/W	R/W	R/W	R/W	----	R/W	R/W	R/W	R/W
Reset value	0	0	1	----	0	0	0	0

Bit7 OPAEN: Op-AMP Enable bit;
0: Op-AMP disable;
1: Op-AMP Enable.

Bit6 OPAFM: Op-AMP adjustment mode Enable;
0: Normal mode;
1: Adjustment mode.

Bit5 OPAFE: Op-AMP Output filter Enable;
0: Disable;
1: Enable.

Bit4 Reserved

Bit3~Bit2 OPAPS1-OPAPS0: Op-AMP Positive Input Selection bit;
00: Connect to GND;
01: Connect to 1K pull-down resistor;
1x: Connect to OPP Port (if OPAEN=1, Enable OPP Port as Op-AMP Positive Input).

Bit1~Bit0 OPANS1-OPANS0: Op-AMP Negative Input Selection Bit;
00: Connect to GND;
01: Connect to 1K pull-down resistor;
1x: Connect to OPP Port (if OPAEN=1, Enable OPP Port as Op-AMP Negative Input)

Op-AMP control register OPACON1 (109H)

109H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPACON1	----	----	----	----	OPO2ADE	----	ANRS1	ANRS0
R/W	----	----	----	----	R/W	----	R/W	R/W
Reset value	----	----	----	----	0	----	0	0

Bit7~Bit4 Reserved

Bit3 OPO2ADE: Op-AMP Output port OPAO connected to CAP*port Enable Bit;
0: Disable;
1: Enable.

Bit2 Reserved

Bit1~Bit0 ANRS1-ANRS0: Op-AMP Output port OPAO connected to CAP* port resistor selection bit (OPO2ADE=1 it is valid);
00: Op-AMP Output direct connect to CAP Port;
01: Op-AMP Output connect to CAP Port via 1K resistor;
10: Op-AMP Output connect to CAP Port via 10K resistor;
11: Op-AMP connect to CAP Port via 20K resistor;

Op-AMP Adjust register OPAADJ (107H)

107H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPAADJ	OPADOUT	OPARS	----	OPAADJS [4: 0]				
R/W	R	R/W	----	R/W	R/W	R/W	R/W	R/W
Reset value	0	0	----	1	0	0	0	0

Bit7 OPADOUT: Op-AMP Output in adjustment mode, read only.

Bit6 OPARS: Op-AMP adjustment mode Input port selection;

0: Negative Input;

1: Positive Input.

Bit5 Reserved

Bit4~Bit0 OPAADJ [4: 0]: Op-AMP adjustment out-of-balance Voltage adjustment bit.

18. Electrical Parameter

18.1 DC Electrical Parameter

Symbol	Parameter	Test Condition		MIN	TYP	MAX	Unit
		VDD	Condition				
VDD	Operation Voltage	-	8M	3.5	-	5.5	V
		-	4M	3.5	-	5.5	V
I_{DD}	Operation Current	5V	ADC Enable	-	3	-	mA
		3V	ADC Enable	-	2	-	mA
I_{STB}	Standby Current	5V	----		0.1		mA
		3V	----		0.1		mA
V_{IL}	Low Voltage Level Input Voltage	-	----	-	-	0.3VDD	V
V_{IH}	High Voltage Level Input Voltage	-	----	0.7VDD	-	-	V
V_{OH}	High Voltage Level Output Voltage	-	Without Loading	0.9VDD	-	-	V
V_{OL}	Low Voltage Level Output Voltage	-	Without Loading	-	-	0.1VDD	V
V_{ADI}	AD Port Input Voltage	-	----	0	-	VDD	V
V_{AD}	AD Module Operation Voltage	-	----	2.7	-	5.5	V
V_{EEPROM}	EEPROM module Operation Voltage	-	-----	3.0	-	5.5	V
E_{AD}	AD Conversion error	-	----	-	±2	-	-
R_{PH}	Pull up resistor value	5V	----	-	35	-	K
		3V	----	-	65	-	K
I_{OL}	Output sink Current	5V	V_{OL}=0.3VDD	-	60	-	mA
		3V	V_{OL}=0.3VDD	-	25	-	mA
I_{OH}	Output source Current	5V	V_{OH}=0.7VDD	-	15	-	mA
		3V	V_{OH}=0.7VDD	-	10	-	mA

18.2 OPA Electrical Characteristics

(VDD=5.0V, TA= 25°C, unless otherwise illustrated)

Symbol	Parameter	Test Condition	MIN value	TYP value	MAX value	Unit
DC electrical characteristics						
VDD	Operation Voltage	VDD=3.5~5.5V	3.5		5.5	V
I_{DD}	Operation Current	VDD=5.0V, V_{CM}=0V		0.85	1.5	mA
I_{OFF}	Cut Off Current	VDD=5.0V, V_{CM}=0V		3	100	nA
V_{OPOS}	Input out of balance Voltage	After calibration, VDD=5V V_{CM}=0V	-4		4	mV
V_{CM}	Common mode Voltage range		0		VDD-1.5V	V
V_{OH}	MAX Output Voltage	ILOAD=1mA			VDD=0.1	V
V_{OL}	MIN Output Voltage	ILOAD=1mA	0.1			V
PSRR	Power source Voltage rejection ratio*	V_{CM}=0V		50		dB
CMRR	Common mode rejection ratio*	VDD=5V V_{CM}=0~VDD-1.5V		85		dB
AC electrical characteristics						
A_{OL}	Open loop Gain*			70		dB
GBW	Gain Bandwidth*	R_L=1MΩ, C_L=100pF		0.4		MHz

* Means guaranteed by design, not mass production tested.

18.3 COMP Electrical Characteristics

(VDD=5.0V, TA= 25°C, unless otherwise illustrated)

Symbol	Parameter	Test Condition	MIN value	TYP value	MAX value	Unit
DC electrical characteristics						
VDD	Operation Voltage	VDD=3.5~5.5V	3.5		5.5	V
I_{DD}	Operation Current	VDD=5.0V, V_{CM}=0.1V		0.25	0.35	mA
I_{OFF}	Cut off Current	VDD=5.0V, V_{CM}=0.1V		3	100	nA
V_{OPOS}	Input out of balance Voltage	After calibration, VDD=5V V_{CM}=0.1V	-4		4	mV
V_{CM}	Common mode Voltage range		0		VDD-1.5V	V
PSRR	Power source Voltage rejection ratio*	V_{CM}=0.1V		100		dB
CMRR	Common mode rejection ratio*	VDD=5V V_{CM}=0~VDD-1.5V		90		dB
AC electrical characteristics						
A_{OL}	Open loop Gain*			85		dB
BW	Bandwidth*			120		MHz
Transient characteristics						
T_{STB}	Stabilization Time*	VDD=5.0V, V_{CM}=0.1V			1	us
T_{PGD}	Response Time*	VCOMP- = 1V, VCOMP+ = VCOMP- ±0.1V		40	100	ns

* Means guaranteed by design, not mass production tested.

18.4 AC Electrical Characteristics

Symbol	Parameter	Test Condition		MIN	TYP	MAX	Unit
		VDD	Condition				
T_{WDT}	WDT reset time	5V	---	-	18	-	ms
		3V	---	-	36	-	ms
T_{AD}	AD Conversion time	5V	---	-	41	-	CLK
		3V	---	-	41	-	CLK
T_{EEPROM}	EEPROM Write time	5V	---	-	2.5	-	ms
		3V	---	-	2.5	-	ms

18.5 Internal RC Oscillation Characteristics

18.5.1 Internal RC Oscillation Voltage Profile

Test Condition	Oscillation Frequency (TYP value) (Hz)
2.5V	8.0M
2.6V	8.1M
2.8V	8.2M
3.0V	8.3M
3.2V	8.3M
3.4V	8.2M
3.6V	8.2M
3.8V	8.2M
4.0V	8.1M
4.2V	8.1M
4.4V	8.1M
4.6V	8.0M
4.8V	8.0M
5.0V	8.0M
5.2V	8.0M
5.4V	7.9M
5.5V	7.8M

18.5.2 Internal RC Oscillation Temperature Profile

Test Condition	-20℃	25℃	40℃	60℃	85℃
Oscillation Frequency (TYP value) (Hz)	7.9M	8.0M	8.0M	8.1M	8.1M

19. Instructions

19.1 Instructions Table

mnemonic		operation	Instruction cycle	symbol
control				
NOP		Empty operation	1	None
STOP		Enter sleep mode	1	TO, PD
CLRWDT		Clear watchdog timer	1	TO, PD
Data transfer				
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
logic operation				
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
Shift operation				
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
Increase/decrease				
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	Instruction cycle	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
Bit operation				
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
look-up table				
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
Math operation				
ADDA	[R]	ACC+[R]→ACC	1	C, DC, Z, OV
ADDR	[R]	ACC+[R]→R	1	C, DC, Z, OV
ADDCA	[R]	ACC+[R]+C→ACC	1	Z, C, DC, OV
ADDCR	[R]	ACC+[R]+C→R	1	Z, C, DC, OV
ADDIA	i	ACC+i→ACC	1	Z, C, DC, OV
SUBA	[R]	[R]-ACC→ACC	1	C, DC, Z, OV
SUBR	[R]	[R]-ACC→R	1	C, DC, Z, OV
SUBCA	[R]	[R]-ACC-C→ACC	1	Z, C, DC, OV
SUBCR	[R]	[R]-ACC-C→R	1	Z, C, DC, OV
SUBIA	i	i-ACC→ACC	1	Z, C, DC, OV
HSUBA	[R]	ACC-[R]→ACC	1	Z, C, DC, OV
HSUBR	[R]	ACC-[R]→R	1	Z, C, DC, OV
HSUBCA	[R]	ACC-[R]- $\overline{C}$ →ACC	1	Z, C, DC, OV
HSUBCR	[R]	ACC-[R]- $\overline{C}$ →R	1	Z, C, DC, OV
HSUBIA	i	ACC-i→ACC	1	Z, C, DC, OV
Unconditional transfer				
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
Conditional transfer				
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

19.2 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 bit: 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 bit: 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
HSUBCR  R01     ; execute: R01= (80H-77H-C) =09H (C=0)
                           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 into 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 (oAH) 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 (oAH) 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 bit: 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 bit: 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 bit: C, DC, Z, OV

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 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
SUBA    R01     ; operation result: ACC=80H-77H=09H
```

SUBR [R]

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

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
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_SPH

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_SPH

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

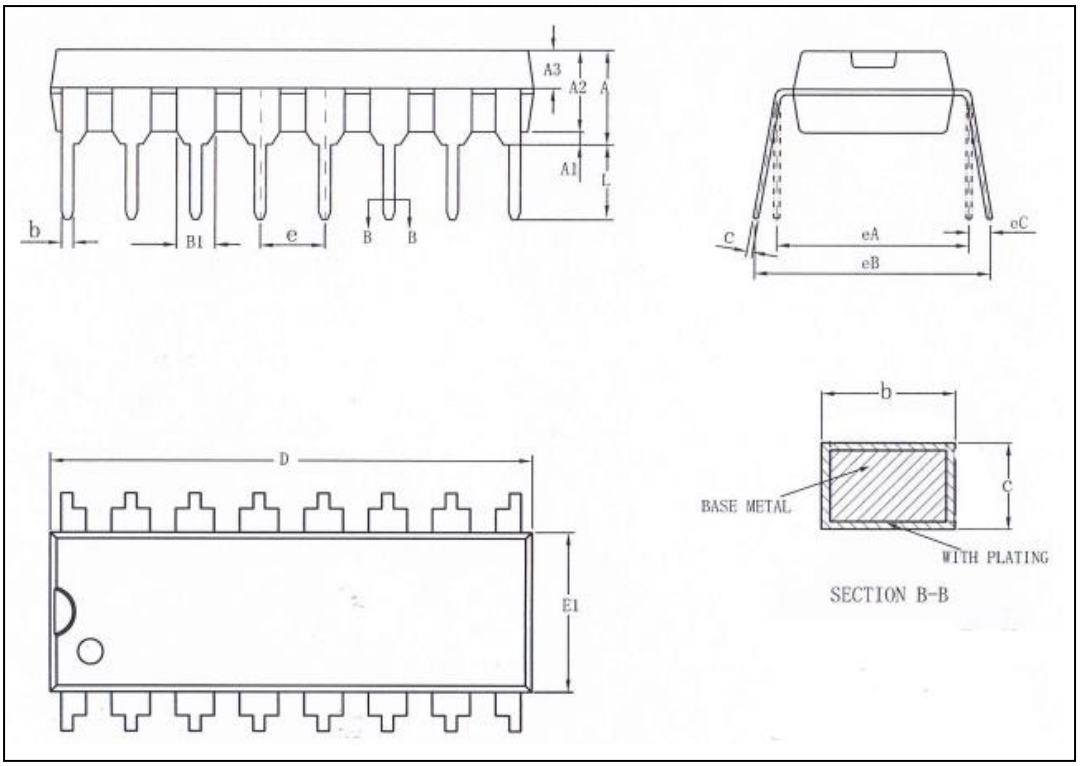
affected flag bit: Z

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
```

20. packaging

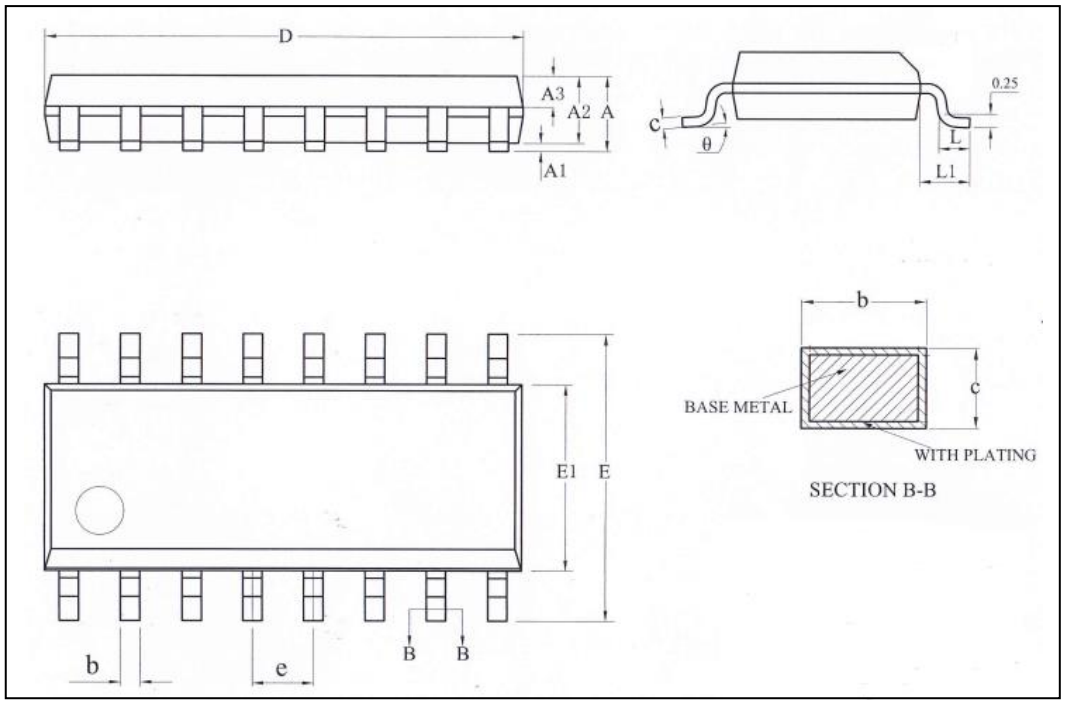
20.1 DIP16



Symbol	Millimeter		
	Min	Nom	Max
A	3.60	-	4.80
A1	0.50	-	-
A2	3.05	-	3.45
A3	1.40	-	1.60
b	0.38	-	0.55
B1	1.52REF		
c	0.21	-	0.35
D	19.00	-	19.40
E1	6.25	6.35	6.45
e	2.54BSC		
eA	7.62REF		
eB	7.62	-	10.90
eC	0	-	1.52
L	2.92	-	-

Caution: Package dimensions do not include mold flash or gate burrs.

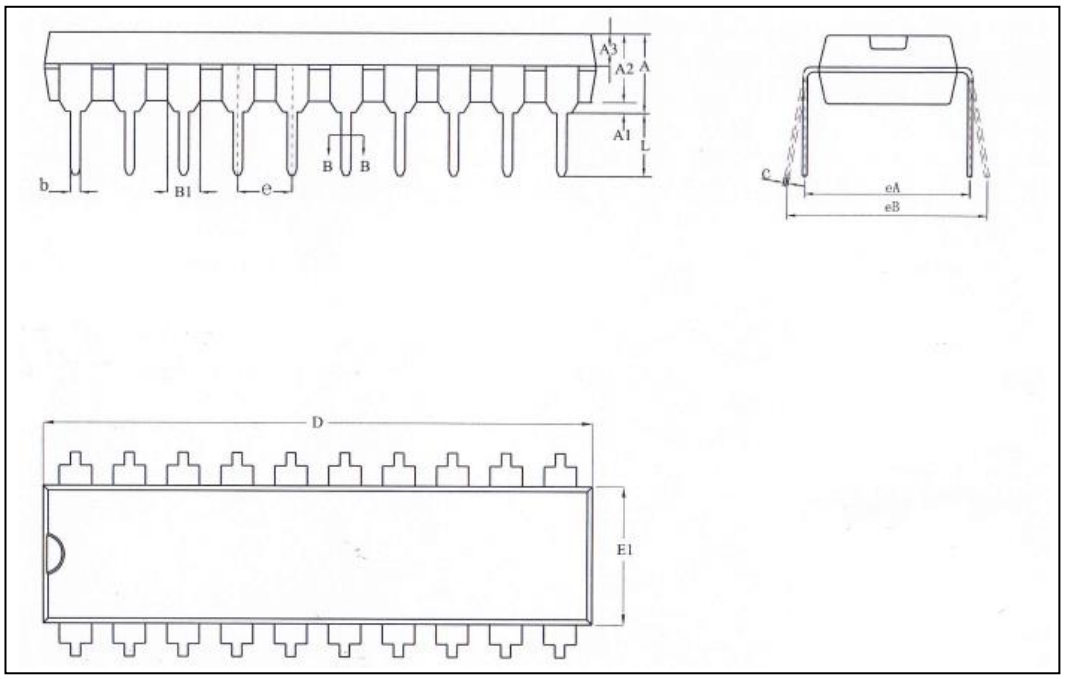
20.2 SOP16



Symbol	Millimeter		
	Min	Nom	Max
A	-	-	1.85
A1	0.05	-	0.25
A2	1.30	1.40	1.60
A3	0.60	0.65	0.71
b	0.35	-	0.51
c	0.19	-	0.26
D	9.70	9.90	10.10
E	5.80	6.00	6.20
E1	3.70	3.90	4.10
e	1.27BSC		
L	0.40	-	0.81
L1	1.05REF		
θ	0	-	8°

Caution: Package dimensions do not include mold flash or gate burrs.

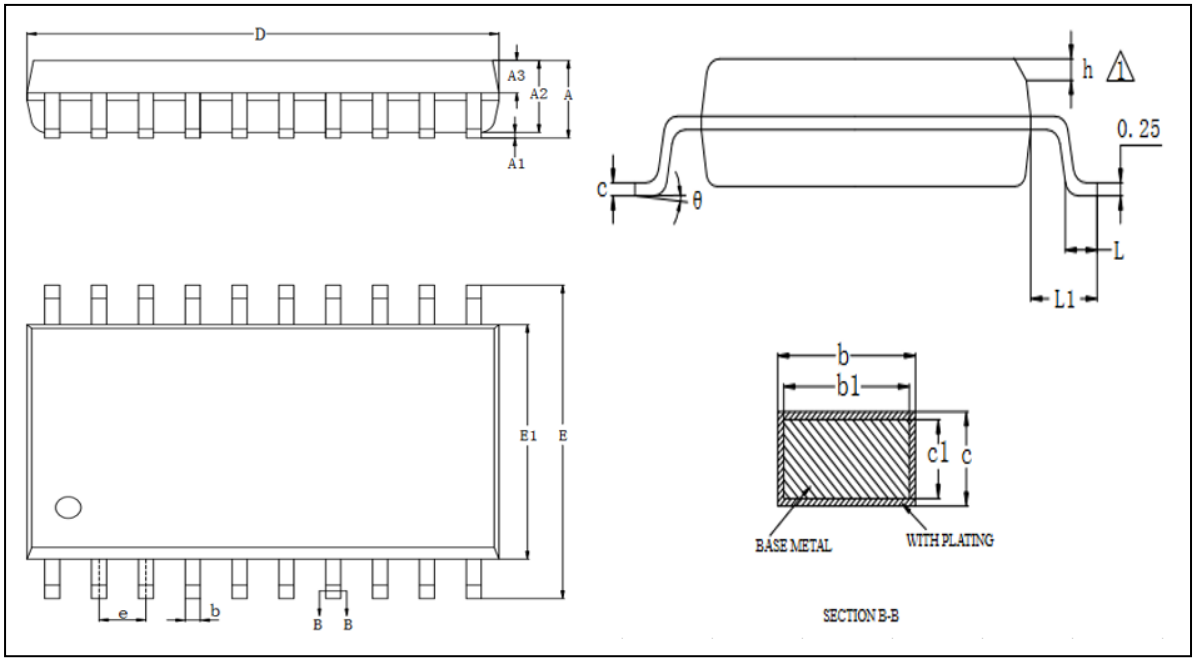
20.3 DIP20



Symbol	Millimeter		
	Min	Nom	Max
A	3.60	3.80	4.00
A1	0.51	-	-
A2	3.20	3.30	3.40
A3	1.47	1.52	1.58
b	0.44	-	0.52
B1	1.52REF		
D	25.80	-	26.33
E1	6.35	-	6.65
e	2.54BSC		
eA	7.62REF		
eB	7.62	-	9.30
L	3.00	-	-
θ	0	-	8°

Caution: Package dimensions do not include mold flash or gate burrs.

20.4 SOP20



Symbol	Millimeter		
	Min	Nom	Max
A	-	-	2.65
A1	0.10	-	0.30
A2	2.24	-	2.44
A3	0.97	1.02	1.07
b	0.39	-	0.47
b1	0.38	0.41	0.44
c	0.25	-	0.30
c1	0.24	0.25	0.26
D	12.65	-	12.90
E	10.10	10.30	10.50
E1	7.40	7.50	7.60
e	1.27BSC		
h	0.50REF		
L	0.70	-	1.01
L1	1.40REF		
θ	0	-	8°

Caution: Package dimensions do not include mold flash or gate burrs.

21. Version revision

<table>
<tr><th>Version number</th><th>time</th><th>Revised content</th></tr>
<tr><td>V1.0</td><td>July, 2016</td><td>CCP2IN2 change to CCP2IN0 (P26、P95)in whole document</td></tr>
<tr><td rowspan="2">V1.1</td><td>July, 2016</td><td>Pre-scaler application added 2 lines of instructions, to ensure it won’t cause reset.
<table>
<tr><td>CLR</td><td>TMR0</td><td>; TMR0 clear to zero</td></tr>
<tr><td>CLRWDT</td><td></td><td>; WDT clear to zero</td></tr>
<tr><td>LDIA</td><td>B’00xx1111’</td><td>;</td></tr>
<tr><td>LD</td><td>OPTION_REG, A</td><td></td></tr>
<tr><td>LDIA</td><td>B’00xx1xxx’</td><td>; Configure new pre-scaler</td></tr>
<tr><td>LD</td><td>OPTION_REG, A</td><td></td></tr>
</table></td></tr>
<tr><td>July, 2016</td><td>Added SSPMSK address</td></tr>
<tr><td>V1.2</td><td>Aug, 2016</td><td>1、 Chapter16.4
Original:
write data EEPROM storage
<table>
<tr><td></td><td>LD</td><td>A, ADDRL</td><td>; write address</td></tr>
<tr><td></td><td>LD</td><td>EEADR, A</td><td></td></tr>
<tr><td></td><td>LD</td><td>A, ADDRH</td><td></td></tr>
<tr><td></td><td>LD</td><td>EEADRH, A</td><td></td></tr>
<tr><td></td><td>LD</td><td>A, DATAL</td><td>; write data</td></tr>
</table>
Revised:
Selected part of the 4 lines, first 2 lines change to:
LD A, ADDR ; write address
LD EEADR, A
Delete the next 2 lines.

2、 Chapter 16.2.2
EEPROM address register EEADR (10DH)
<table>
<tr><td>10DH</td><td>Bit7</td><td>Bit6</td><td>Bit5</td><td>Bit4</td><td>Bit3</td><td>Bit2</td><td>Bit1</td><td>Bit0</td></tr>
<tr><td>EEADR</td><td>----</td><td>----</td><td>----</td><td>EEADR4</td><td>EEADR3</td><td>EEADR2</td><td>EEDAT1</td><td>EEDAT0</td></tr>
<tr><td>read/write</td><td>----</td><td>----</td><td>----</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
<tr><td>Reset value</td><td>----</td><td>----</td><td>----</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>
EEPROM control register EECON1 (18CH)
<table>
<tr><td>11BH</td><td>Bit7</td><td>Bit6</td><td>Bit5</td><td>Bit4</td><td>Bit3</td><td>Bit2</td><td>Bit1</td><td>Bit0</td></tr>
<tr><td>EECON1</td><td>EEPGD</td><td>----</td><td>----</td><td>----</td><td>WRERR</td><td>WREN</td><td>WR</td><td>RD</td></tr>
<tr><td>read/write</td><td>R/W</td><td>----</td><td>----</td><td>----</td><td>R/W</td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
<tr><td>Reset value</td><td>0</td><td>----</td><td>----</td><td>----</td><td>x</td><td>0</td><td>0</td><td>0</td></tr>
</table>
<table>
<tr><td>Bit7~Bit6</td><td>EEPGD:</td><td>Data EEPROM enable bit;</td></tr>
<tr><td></td><td>1=</td><td>Enable operation of data EEPROM;</td></tr>
<tr><td></td><td>0=</td><td>Disable operation of data EEPROM;</td></tr>
<tr><td>Bit6~Bit4</td><td>not used</td><td>Read as 0.</td></tr>
<tr><td>Bit3</td><td>WRERR:</td><td>EEPROM error flag bit;</td></tr>
<tr><td></td><td>1=</td><td>Write early abortion (any WDT reset or undervoltage reset during normal operation)</td></tr>
</table>
1 place, EEDATA1 change to EEADR1, EEDATA0 change to EEADR0,
2 place: Bit7-6 change to Bit7</td></tr>
</table>

3、Chapter 2.1.2

Summary of special registers in CMS89F52x Bank0

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
00h	INDF	Addressing this unit will use FSR content to (rather than physical register)							
01h	TMR0	TIMER0 data register							
02h	PCL	Lower bit of program counter							
03h	STATUS	IRP	----	----	TO	PD	Z	DC	C
04h	FSR	memory pointers for indirect addressing of data registers							
05h	PORTA	----	RA6	RA5	RA4	RA3	RA2	RA1	RA0
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
09h	PORTE	----	----	----	----	----	----	RE1	RE0
0Ah	PCLATH	----	----	----	Write buffer of higher 5 bits of program counter				
0Bh	INTCON	GIE	PEIE	T0IE	INTE	----	T0IF	INTF	----
0Ch	PIR1	EEIF	ADIF	SSPIF	BCLIF	CCPIF	----	TMR2IF	TMR1IF
0Dh	PIR2	----	----	C5IF	C4IF	C3IF	C2IF	C1IF	PPGIF
0Eh	TMR1L	Data register of 16-bits TIMER1 register lower bit							
0Fh	TMR1H	Data register of 16-bits TIMER1 register higher bit							
10h	T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	T0OSCEN	T1SYNC	TMR1CS	TMR1ON
11h	TMR2	TIMER2 module register							
12h	T2CON	----	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0

1 place: change to “PPGWDTIF”
2 places: change to “----” and “----”

4、Chapter 2.1.2

Summary of special registers in CMS89F52x Bank1

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
80h	INDF	Addressing this unit will use FSR content to (rather than physical register)							
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
82h	PCL	Lower bits of program counter							
83h	STATUS	IRP	----	----	TO	PD	Z	DC	C
84h	FSR	memory pointers for indirect addressing of data memory							
85h	TRISA	----	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	----
89h	TRISE	----	----	----	----	TRISE3	TRISE2	TRISE1	TRISE0
8Ah	PCLATH	----	----	----	Write buffer of higher 5 bits of program counter				
8Bh	INTCON	GIE	PEIE	T01E	INTE	----	T0IF	INTF	----
8Ch	PIE1	EEIE	ADIEF	SSPIE	BCLIE	CCPIE	----	TMR2IE	TMR1IE
8Dh	PIE2	----	----	C5IE	C4IE	C3IE	C2IE	C1IE	PPGIE
8Fh	OSCCON	----	IRCF2	IRCF1	IRCF0	----	----	----	----
90h	OSCTUNE	----	----	----	TUN4	TUN3	TUN2	TUN1	TUN0

1 place: change to “----”
2 place: change to” TRISB0”
3 place: change to “----”and “----‘
4 place: change to “ADIE”
5 place: change to “PPGWDTIE “

5、chapter 2.1.2

Summary of special registers in CMS89F52x Bank3

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
180h	INDF	Addressing this unit will use FSR content to (rather than physical register)							
181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
182h	PCL	Lower bits of program counter (PC)							
183h	STATUS	IRP	----	----	TO	PD	Z	DC	C
184h	FSR	memory pointers for indirect addressing of data memory							
186h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB73	TRISB2	TRISB1	TRISB0
187h	PAANSEL	----	PAANS6	PAANS5	PAANS4	PAANS3	PAANS2	PAANS1	PAANS0
188h	PBANSEL	PBANS7	PBANS6	PBANS5	PBANS4	PBANS3	PBANS2	PBANS1	PBANS0
189h	PEANSEL	----	----	----	----	----	----	PEANS1	PEANS0
18Ah	PCLATH	----	----	----	Write buffer of higher 5 bits of program counter				
18Bh	INTCON	GIE	PEIE	T01E	INTE	----	T01F	INTF	----
18Ch	EECON1	EEPGD	----	----	----	WRERR	WREN	WR	RD
18Dh	EECON2	EEPROM control register 2 (not physical register)							
18Eh	CCPRL	Capture register lower bits							
18Fh	CCPRH	Capture register higher bits							
190h	CCPCON	CCPEN	----	----	CCPIS	CCPES	CPTCS2	CPTCS1	CPTCS0
191H	SSPMSK	MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0

1 place: change to “----”
2 places: ” CPTCS2” change to” CCPM2”, “CPTCS1” change to” CCPM1”, “CPTCS0” change to” CCPM0”

		6、chapter 15.3 PPG Control Register PPGCON (17H) 1 17H | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 PPGCON | DETC5F | DETC4F | ---- | RELOAD_EN | DETC4EN | DETC3EN | PPGMD | PPG_ON R/W | R/W | R/W | ---- | R/W | R/W | R/W | R/W | R/W Reset Value | 1 | 1 | ---- | 0 | 0 | 0 | 0 | 0 Bit7 | DETC5F: | Comparator5 Status Bit (PPG Status Bit); | 0: | Clear Comparator5 的 0->1 Flip flag, (if DETC5EN=1, then PPG re-open); | 1: | Has Comparator5 的 0->1 Flip, invalid while writing 1 (if DETC5EN=1, then PPG turn off). 1place: “DETC4EN” change to “DETC5EN”, “DETC3EN” change to “DETC4EN”
V1.3	Nov, 2017	Modify multiple places in document for better elaboration.
V1.4	Apr, 2020	1、 Added OPA, COMP electrical parameters 2、 Corrected some mistakes in packaging info.
V1.5	Aug, 2021	Updated pin information
V1.6	Feb, 2022	Updated to new format
V1.7.0	Sep, 2024	Modified DIP16/SOP16/DIP20/SOP20 package dimensions