CACHIP ***Built - in 12 Bit ADC / Touch Key / LCD Driver / 1T 8051 Flash MCU***

CA51F3 Series MCU

User Guide

REV 2.2

IMPORTANT STATEMET: We reserve the right to make further clarifications regarding the reliability, functionality and design of all products listed below. We also reserve the right to make changes to all documentation for this product without notice. Customers should ask our sales staff for the latest documentation when using this product. Hereby declare!

Table of Contents

1 Introduction

CA51F3 series chip is 8-bit MCU based on 1T 8051 core and operates 10 times faster than the traditional 8051 chips with definitely better performance. The Flash program memory embedded can be programmed for times and offers users with three storage choices(8/16/32K) which brings great convenience to software development. Not only traditional 8051 chip features, CA51F3 also includes 12bit ADC, LCD/LED driver, Touch Key, 16bit PWM, UART, RTC, low voltage detection (LVD) and other function modules. It can operates in three Power Save Modes (IDLE/STOP/LOW SPEED) in order to meet different power consumption needs. With great functions and anti-jamming performance, CA51F3 could be used in various fields such as car's or family audio system, small household appliance, Bluetooth stereo, car's electronics, digital motor, sports equipment, motor control, health care, instrument and meters, security guard, power control, factory control and doorbell products.

2 Basic Features

- Core
  - CPU： 1T 8051, with highest speed 10 times faster than traditional 8051
  - Compatible with 8051 instruction set, with double DPTR mode
  - CPU frequency： Highest at 27MHz
- Memory
  - Flash : 8 / 16 / 32K byte , can be erased and overwritten for times
  - Flash could be divided into program storage and data storage. Data storage could be used to store data which need to be protected during power off and save EEPROM in theend of the day
  - RAM:256 bytes internal RAM, 2K bytes external RAM
- Operating Voltage
  - Operating Voltage： 1.8V - 5.5V
- Interrupt System
  - 15 effective interrupt source
  - Two levels for interrupt priority which also supports interrupt nesting
  - 10 external interrupt source. For each external interrupt, any of the signal pin could be configured as interrupt pin
- Clock System
  - External RTC Oscillator： 32.768KHz (supported by only some type of the chip)
  - Internal Low Speed RC Oscillator： 131KHz
  - Internal PLL： The ratio for frequency multiplication ranges from 2 to 10. Reference clock ranges from 2 to 4 MHz in Internal RC Oscillator
  - Internal High Speed RC Oscillator： 2 - 4MHz, with 1% possible error (The factory original frequency is 3.6864MHz@3.3V/25℃)
  - With External Clock Monitor Module embedded, the Clock System is able to monitor external clocks' status so that the external clocks will not crash due to oscillation stop

- RTC Function
  - The internal RTC module can count hours, minutes, seconds, days and weeks. It can also be used as alarm clock
  - Support millisecond/half-second interrupt function
- General Purpose IO (GPIO)
  - Supports 26 GPIO ports at most(may be different for different types)
  - Support push-pull, open-drain, strong pull-up, weak pull-up, strong pull-down, weak pull-down and high resistance mode
  - Different drive strength and flip speed can be set in push-pull mode
- Timer.
  - Three 16-bit general Timers: Timer 0, Timer 1, Timer 2
- Watch Dog
  - 27 bit Watch Dog Timer, 16 bit precision configurable, with Watch Dog Reset and Interrupt configurable as well
- UART
  - Supports up to 2 UART interfaces at most(may be different for different types)
  - Support 1 byte receive cache
- I^{2}C
  - One I^{2}C port embedded which supports Master-Slave mode and Standard/Fast/High Speed mode as well
  - I^{2}C can set digital filtering to enhance I^{2}C anti-interference performance.
- LCD Driver
  - Supports 5com x 8seg、4com x 9seg、3com x 10seg (may be different for different types)
  - Configurable Duty Cycle：1/2、1/3、1/4、1/5 Duty
  - Bias voltage configurable：1/2、1/3、1/4 Bias
  - Supports 8 levels contrast adjustment
  - Supports 3 levels drive current which enables the user to modify according to different LCD screen
- LED Driver
  - Supports 5com x 8seg、4com x 9seg、3com x 10seg (may be different for different types)
  - Supports 8 levels brightness adjustment
  - Led COM pin can set strong sink current mode to realize LED high brightness display effect.
- Analog/Digital Converter(ADC)
  - Supports 8 channel 12-bit SAR ADC (may be different for different types)
  - Supports 3 Reference Voltages： VDD, Internal Reference Voltage, External Reference Voltage
  - When Internal Reference Voltage is selected, VDD could be measured as well
- PWM
  - Supports 6 channel PWM, any periods or duty cycles are configurable in range of 16 bits (may be different for different types)
  - Supports Complementary Mode and Dead time Control which could be used to drive Brushless DC motor
  - Supports center fixed mode or edge fixed mode

- Supports to output internal clock directly
- Supports PWM Interrupt

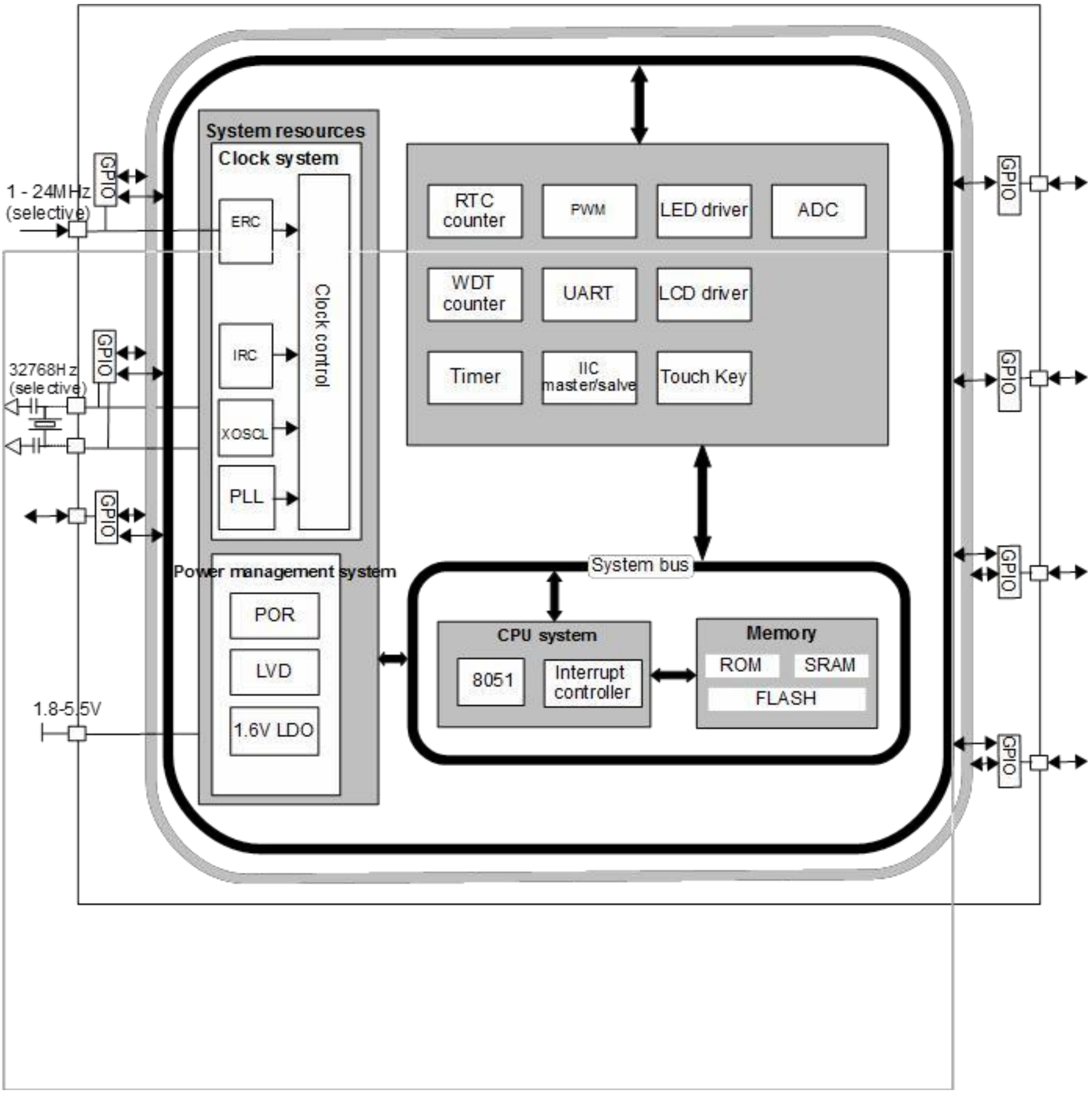
- Touch Key
  - Internal Touch Sensor Controller
  - Supports 20 touch channel at most(may be different for different types)
  - Touch can set internal charging and internal reference, can effectively suppress power supply low frequency interference
  - Support touch pins and LED driver pins multiplexing
  - Built-in waterproof compensation mechanism
  - High anti-interference, in line with EMC (CS) standards
  - Support touch power saving mode, the lowest power consumption is less than 10uA
- Low Voltage Detector(LVD)
  - Voltage detectable ranges from 1.8 V to 4.8V which is also configurable
  - Low voltage reset/interrupt configurable
- Reset Mode
  - Supports variable reset sources： Hard Reset, Soft Reset, Watch Dog Reset, LVD Reset, Power On/Down Reset
- Low power consumption
  - For STOP Mode, current<7uA
  - For IDLE Mode, current<12uA
  - For Low Speed Mode, current<20uA
- Program Download and Simulation
  - Supports ISP and IAP
  - Supports simulation online
- Package Type: SOP16/SOP20/TSSOP20/QFN20/SOP28/SSOP28

3 Chip Model and Function Description

Table 3-1 CA51F3 Specific Models and Their Features

Models	Flash Storage [BYTE]	External Ram[BYTE]	External Low Speed Crystal Oscillator[32.768KHz]	GPIO	UART	I^2C	16 bit PWM Channels	12 bit ADC Channels	Touch Key	LCD Drive [com x seg]	LED Drive[com x seg]	Simulation On Chip	Package Type
CA51F351S3	8K	2K	---	14	2	√	6	5	11	---	---	√	SOP16
CA51F351S4	8K	2K	---	18	2	√	6	7	15	3X5 4X4	3X5 4X4	√	SOP20
CA51F351P4	8K	2K	---	18	2	√	6	7	15	3X5 4X4	3X5 4X4	√	TSSOP20
CA51F351S6	8K	2K	√	26	2	√	6	8	20	3X10 4X9 5X8	3X10 4X9 5X8	√	SOP28
CA51F351P6	8K	2K	√	26	2	√	6	8	20	3X10 4X9 5X8	3X10 4X9 5X8	√	SSOP28
CA51F3N2	16K	2K	---	18	2	√	5	7	15	--	--	√	QFN20
CA51F352S4	16K	2K	---	18	2	√	6	7	15	3X5 4X4	3X5 4X4	√	SOP20
CA51F352P4	16K	2K	---	18	2	√	6	7	15	3X5 4X4	3X5 4X4	√	TSSOP20
CA51F353P4	32K	2K	---	18	2	√	6	7	15	3X5 4X4	3X5 4X4	√	TSSOP20
CA51F353S6	32K	2K	√	26	2	√	6	8	20	3X10 4X9 5X8	3X10 4X9 5X8	√	SOP28
CA51F353P6	32K	2K	√	26	2	√	6	8	20	3X10 4X9 5X8	3X10 4X9 5X8	√	SSOP28

4 Block Diagram



5 Pin Package and Description

5.1 Package Definition

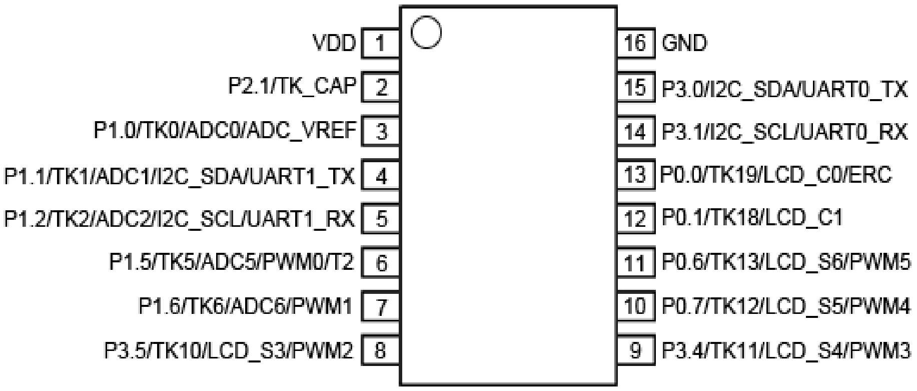


Figure 5-1-1 SOP16 Package

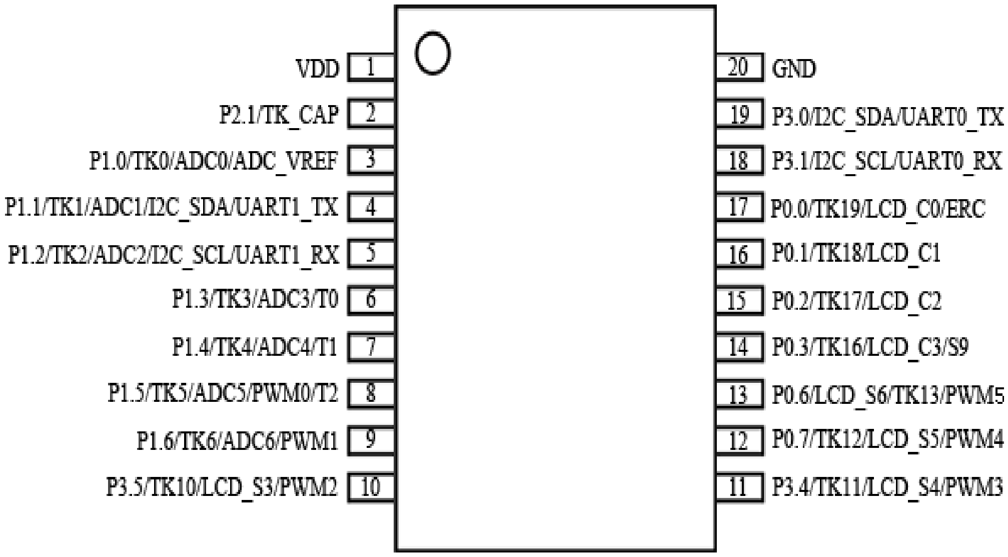


Figure 5-1-2 SOP20 / TSSOP20 Package

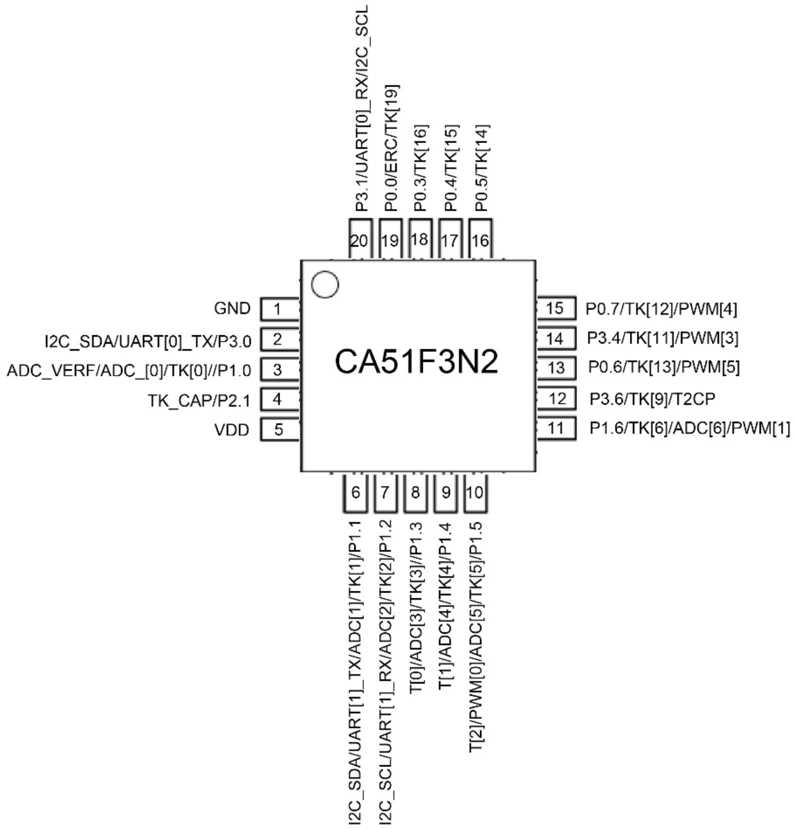


Figure 5-1-3 QFN20 Package

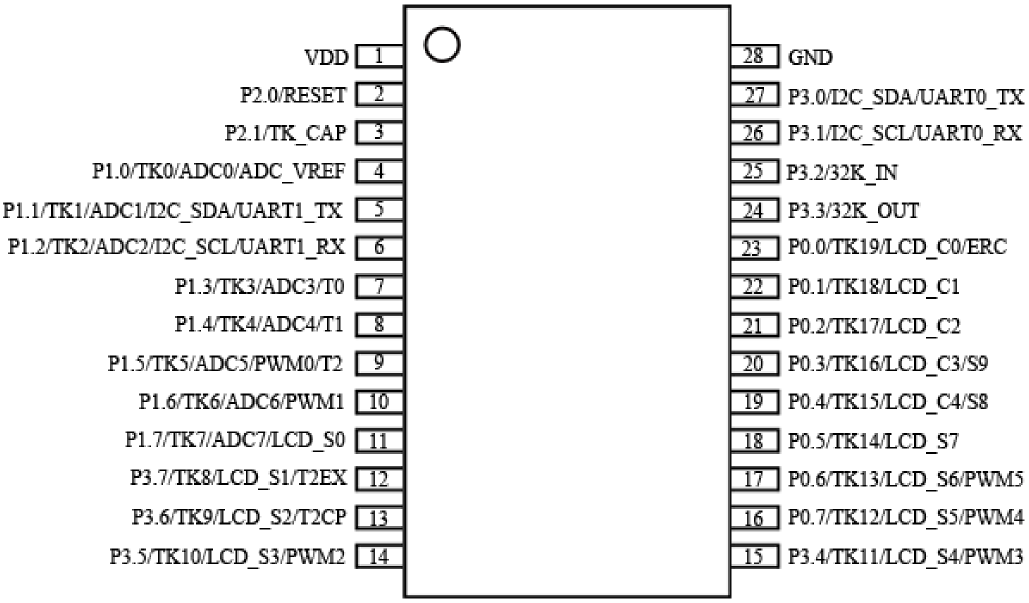


Figure 5-1-3 SOP28/SSOP28 Package

5.2 Pin Description

Table 5-2-1 Pin Description

Pin sequence number				Pin Name	Pin Function	Default Function
SOP28 SSOP28	SOP20 TSSOP2	SOP16	QFN20			
1	1	1	5	VDD	Power supply for the chip	Power supply for the chip
2			-	P2.0/RESET	General bi-directional I/O port Hard reset pin	Hard reset pin
3	2	2	4	P2.1/TK_CAP	General bi-directional I/O port Touch key input reference capacitance	General bi-directional I/O port
4	3	3	3	P1.0/TK[0]/ADC_CH0/ADC_VREF	General bi-directional I/O port Touch channel input ADC analog channel ADC external reference - voltage's input	General bi-directional I/O port
5	4	4	6	P1.1/UART1_TX/TK[1]/ADC_CH1/ I2C_SDA	General bi-directional I/O port Touch channel input ADC channel port UART1 TX port I2C data transfer port	I2C data transfer port
6	5	5	7	P1.2/UART1_RX/TK[2]/ADC_CH2/ I2C_SCL	General bi-directional I/O port Touch channel input ADC channel port UART1 RX port I2C clock transfer port	I2C clock transfer port
7	6	-	8	P1.3/TK[3]/ADC_CH3/T0	General bi-directional I/O port Touch channel input ADC channel port Input port for T0	General bi-directional I/O port

8	7		9	P1.4/TK[4]/ADC_CH4/T1	General bi-directional I/O port Touch channel input ADC channel port Input port for T1	General bi-directional I/O port
9	8	6	10	P1.5/TK[5]/ADC_CH5/PWM0/T2	General bi-directional I/O port Touch channel input ADC channel port PWM port Input port for T2	General bi-directional I/O port
10	9	7	11	P1.6/TK[6]/ADC_CH6/PWM1	General bi-directional I/O port Touch channel input ADC channel port PWM port	General bi-directional I/O port
11				P1.7/TK[7]/ADC_CH7/LED/LCD_S0	General bi-directional I/O port Touch channel input ADC channel port LCD driver SEGM. output LED SEGM. output	General bi-directional I/O port
12				P3.7/TK[8]/LED/LCD_S1/T2EX	General bi-directional I/O port Touch channel input LCD driver SEGM. output LED SEGM. output Input port for T2EX output	General bi-directional I/O port
13			12	P3.6/TK[9]/LED/LCD_S2/T2CP	General bi-directional I/O port Touch channel input LCD driver SEGM. output LED SEGM. output Input port for T2CP output	General bi-directional I/O port
14	10	8		P3.5/TK[10]/LED/LCD_S3/PWM2	General bi-directional I/O port Touch channel input LCD driver SEGM. output LED SEGM. output PWM port	General bi-directional I/O port

15	11	9	14	P3.4//TK[11]/LED/LCD_S4/PWM3	General bi-directional I/O port Touch channel input LCD driver SEGM. output LED SEGM. output PWM port	General bi-directional I/O port
16	12	10	15	P0.7/TK[12]/LED/LCD_S5/PWM4	General bi-directional I/O port Touch channel input LCD driver SEGM. output LED SEGM. output PWM port	General bi-directional I/O port
17	13	11	13	P0.6/TK[13]/LED/LCD_S6/PWM5	General bi-directional I/O port Touch channel input LCD driver SEGM. output LED SEGM. output PWM port	General bi-directional I/O port
18			16	P0.5/TK[14]/LED/LCD_S7	General bi-directional I/O port Touch channel input LCD driver SEGM. output LED SEGM. output	General bi-directional I/O port
19			17	P0.4/TK[15]/LCD_C4/LED/LCD_S8	General bi-directional I/O port Touch channel input LED/ LCD driver COM output LED/LCD driver SEGM. output	General bi-directional I/O port

20	14		18	P0.3/TK[16]/LCD_C3/LCD_S9	General bi-directional I/O port Touch channel input LED/LCD driver COM output LED/LCD driver SEGM. output	General bi-directional I/O port
21	15			P0.2/TK[17]/LCD_C2	General bi-directional I/O port Touch channel input LED/ LCD driver COM output	General bi-directional I/O port
22	16	12		P0.1/TK[18]/LCD_C1	General bi-directional I/O port Touch channel input LED/ LCD driver COM output	General bi-directional I/O port
23	17	13	19	P0.0/TK[19]/LCD_C0/ERC	General bi-directional I/O port Touch channel input LED/ LCD driver COM output ERC analog port	General bi-directional I/O port
24				P3.3/ XTAL_OUT_32K	General bi-directional I/O port Output for 32KHz external crystal oscillator	output for 32KHz external crystal oscillator
25				P3.2/ XTAL_IN_32K	General bi-directional I/O port Input for external 32K crystal oscillator	input for 32KHz external crystal oscillator
26	18	14	20	P3.1/UART0_RX/I2C_SCL	General bi-directional I/O port UART0 RX port I2C clock transfer port	I2C clock transfer port
27	19	15	2	P3.0/UART0_TX/I2C_SDA	General bi-directional I/O port UART0 TX port I2C data transfer port	I2C data transfer port
28	20	16	1	GND	Chip ground	GND

Note : For signal pin's alternate function settings, please refer to Table 15-2-5 and Table 15-2-6

6 Central Processing Unit(CPU)

6.1 CPU Introduction

The core of CA51F3 Series is monocyclic 8051 CPU and make it fully compatible with original MCS-51 instruction set. A monocyclic 8051 CPU usually operates 10 times faster than standard 8051 one due to its pipeline structure.

The features of this CPU are：

- 1T 8051 CPU
- Compatible with 8051instruction set, for more you may refer to instruction set in Appendix
- Double DPTR, so that the data could be moved quickly

6.2 Register Description

Program Counter(PC)

Program Counter (PC) is a 16-bit register without register address which is used to control the sequence of instructions. It is set to 0 after reset/power on and the machine will execute the program from zero address.

Accumulator(ACC)

Accumulator (ACC) is a special register and ‘A’ is used as its instruction mnemonic. It is often used to store the operand and result of logical/arithmetic computing.

General Register B

Register B cannot to be used without ACC in multiplying/dividing computing. Instruction MUL AB multiplies 8-bit unsigned number in ACC and B. The lower bytes (16 bit) and higher bytes(16 bit) of the computing result will be stored in A and B respectively. Furthermore, instruction DIV AB divides B by A, and the integer quotient will be stored in A with remainder stored in B. In addition, register B can also be used as general temporary storage register.

Stack Pointer (SP)

Stack Pointer(SP) is a 8 bit special register and indicates where the top of stack is in the internal RAM. It is initialized to 07H after a reset which makes stack actually starts from 08H. Since 08H~1FH belongs to working register group 1~3 , if they are used in program development, SP is recommended to be set to 80H or even higher.

Data Pointer (DPTR)

Data pointer DPTR0/DPTR1 are two 16-bit special register with their higher stored in register DP0H/DP1H respectively and lower bytes stored in register DP0H/DP1H respectively. By setting DPS(PSW.1) either of them can be used. For each DPTR, it can be seen as one 16-bit register or two independent 8-bit registers DP0H/DP1H and DP0L/DP1L.

Program Status Word (PSW)

Program Status Word(PSW) is a register indicates the statues of the CPU. The status bit of it will change correspondingly when the CPU is doing arithmetic or logical operations.

Table 6-2-1 Accumulator ACC

E0H	7	6	5	4	3	2	1	0
ACC	ACC[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Table 6-2-2 General Register B

F0H	7	6	5	4	3	2	1	0
B	B[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Table 6-2-3 Stack Pointer SP

81H	7	6	5	4	3	2	1	0
SP	SP[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	1	1	1

Table 6-2-4 Data Pointer DP0L

82H	7	6	5	4	3	2	1	0
DP0L	DP0L[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Table 6-2-5 Data Pointer DP0H

83H	7	6	5	4	3	2	1	0
DP0H	DP0H[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Table 6-2-6 Data Pointer DP1L

84H	7	6	5	4	3	2	1	0
DP1L	DP1L[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Table 6-2-7 Data Pointer DP1H

85H	7	6	5	4	3	2	1	0
DP1H	DP1H[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Table 6-2-8 Program Status Word PSW

D0H	7	6	5	4	3	2	1	0
PSW	CY	AC	F0	RS[1:0]		OV	DPS	P
R/W	R/W	R/W	R/W	R/W		R/W	R	R
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit symbol	Description
7	CY	Carry flag 0：There is no carry or borrow happened in arithmetic/logical operation 1：There is carry or borrow happened in arithmetic/logical operation
6	AC	Auxiliary Carry Flag 0：There is no auxiliary carry or borrow happened in arithmetic/logical operation 1：There is auxiliary carry or borrow happened in arithmetic/logical operation
5	F0	F0 flag It is defined by the user
4~3	RS	R0~R7 registers’ page selection 00：page 0(mapping to 00H-07H) 01：page 1(mapping to 08H-0FH) 10：page 2(mapping to 10H-17H) 11：page 3(mapping to 18H-1FH)
2	OV	Overflow flag 0：no overflow 1：overflow happened
1	DPS	DPTR selection register, 0 for DPTR0, 1 for DPTR1
0	P	Parity flag 0：the number of 1 in ACC A is even 1：the number of 1 in ACC A is odd

Table 6-2-9 Register SPMAX

8100H	7	6	5	4	3	2	1	0
SPMAX	SPMAX[7:0]							
R/W	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	1	1	1
Bit number	Bit symbol	Description						
7~0	SPMAX	SPMAX is used to record the maximum value of SP. Users can check this register using software to decide whether there is a risk that the stack may overflow						

7 Memory Architecture

7.1 Random Access Memory(RAM)

CA51F3 series offers both internal RAM(256 bytes) and external RAM(2K bytes) for the users and the corresponding address are shown as follows:

- Lower 128 bytes of the internal RAM(address：00H ~ 7FH)supports both direct addressing and indirect addressing.
- Higher 128 bytes of the internal RAM(address：80H ~ FFH)only supports indirect addressing.
- 2K bytes external RAM(address：0000H ~ 07FFH) supports indirect addressing by using MOVX.

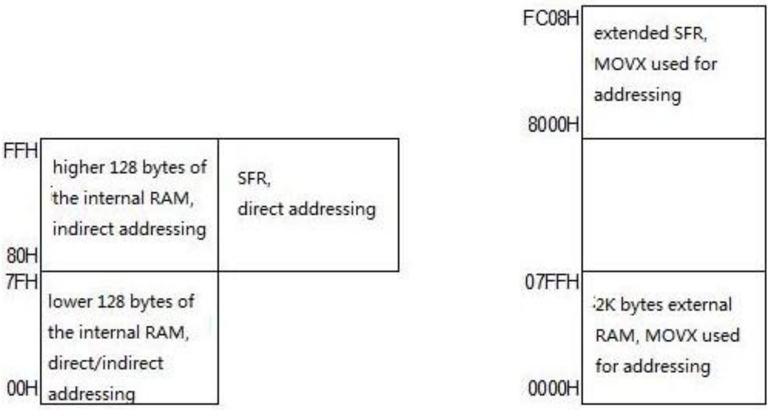

FFH
higher 128 bytes of the internal RAM, indirect addressing
SFR, direct addressing
80H
7FH
lower 128 bytes of the internal RAM, direct/indirect addressing
00H
FC08H
extended SFR, MOVX used for addressing
8000H
07FFH
2K bytes external RAM, MOVX used for addressing
0000H

Figure 7-1-1 RAM Architecture

7.2 Special Function Register(SFR)

The SFR architecture of CA51F3 series is compatible with traditional 8051 chip. SFR and the higher 128 bytes of the internal RAM both use the address 80H ~ FFH that only supports direct addressing, SFR mapping is shown in Table 7-2-1.

Table 7-2-1 Special Function Register Mapping Table

	Bit addressable	Not bit addressable						
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8H	EXIP	EPIE	EPIF	EPCON	IDLSTL	IDLSTH	STPSTL	STPSTH
F0H	B	RTCON	RTCS	RTCM	RTCH	RTCDL	RTCDH	-
E8H	EXIE	RTCSS	RTAS	RTAM	RTAH	RTMSS	RTCIF	LVDCON
E0H	ACC	LXCON	LXCFG	LXDAT	LXDIVL	LXDIVH	ATKNL	ATKNH
D8H	PWMEN	PWMUPD	PWMCMX	PWMCON	PWMCFG	PWMDIVL	PWMDIVH	PWMDUTL

D0H	PSW	PWMDUTH	PWMAIF	PWMBIF	PWMCIF	TKMAXF	TKMINF	HVTH
C8H	T2CON	T2MOD	T2CL	T2CH	TL2	TH2	TKMSL	TKMSH
C0H	INDEX	TKCON	TKCFG	TKMTS	TKCHS	ATKCL	ATKCH	TKIF
B8H	IP	ADCON	ADCFGL	ADCFGH	ADCDL	ADCDH	CKMON	CKMIF
B0H	P3	I2CCON	I2CADR	I2CADM	I2CCCR	I2CDAT	I2CSTA	I2CFLG
A8H	IE	WDCON	WDFLG	WDVTHL	WDVTHH	PLLCON	-	TFCFG
A0H	P2	CKCON	CKSEL	CKDIV	IHCFGL	IHCFGH	ILCFGL	ILCFGH
98H	S0CON	S0BUF	S1CON	S1BUF	S1RELL	S1RELH	RCMSHL	RCMSHH
90H	P1	RCCON	VCKDL	VCKDH	RCTAGL	RCTAGH	RCMSLL	RCMSLH
88H	TCON	TMOD	TL0	TL1	TH0	TH1	IT1CON	IT0CON
80H	P0	SP	DP0L	DP0H	DP1L	DP1H	PWCON	PCON

Due to limited SFR address space, CA51F3 series also added extended special function register in external RAM address space. The mapping is shown as follows.

Table 7-2-2 Extended Special Function Register Mapping Table

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
8000H	P00F	P01F	P02F	P03F	P04F	P05F	P06F	P07F
8008H	P10F	P11F	P12F	P13F	P14F	P15F	P16F	P17F
8010H	P20F	P21F	-	-	-	-	-	-
8018H	P30F	P31F	P32F	P33F	P34F	P35F	P36F	P37F
8020H	ADCALL	ADCALH	-	-	-	-	-	-
8100H	SPMAX	I2CIOS	-	TKPWC	-	-	TLEN	TLDAT
8108H	TLCON	TLFLG	TLCKS	TLCNTKL	TLCNTKH	TLCNTLL	TLCNTLH	TLDIV
8110H	TLCOMS	-	-	-	-	-	-	LXCAD
8118H	-	-	-	-	-	-	-	-
8120H	P00C	P01C	P02C	P03C	P04C	P05C	P06C	P07C
8128H	P10C	P11C	P12C	P13C	P14C	P15C	P16C	P17C
8130H	P20C	P21C	-	-	-	-	-	-
8138H	P30C	P31C	P32C	P33C	P34C	P35C	P36C	P37C
FC00H	MECON	FSCMD	FSDAT	LOCK	PADRD	PTSL	PTSH	REPSET

7.3 Flash

7.3.1 Function Introduction

Flash memory can be 8/16/ 32K byte according to different model and it can be erased and overwritten repeatedly. Flash is also controlled by a group of special registers, therefore users may use these registers to erase/overwrite/set write protect to the Flash and so on.

7.3.2 Flash Architecture

- Flash consists of several sectors which are the smallest units for erasure. Each sector is 128 bytes.

- Flash can be divided into DATA area and PROGRAM area and the division unit is 256 byte. PROGRAM area is used to store use's program and DATA area is used to store data that needs to be protected during power off period.

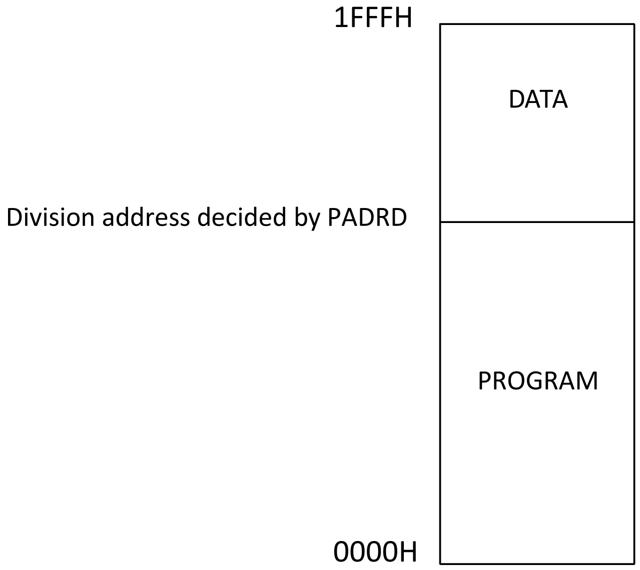


Figure 7-3-1 8K Flash Memory Structure

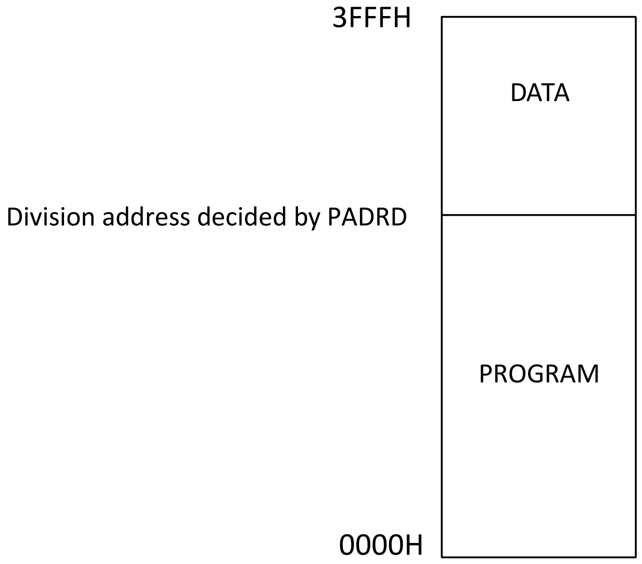


Figure 7-3-2 16K Flash Memory Structure

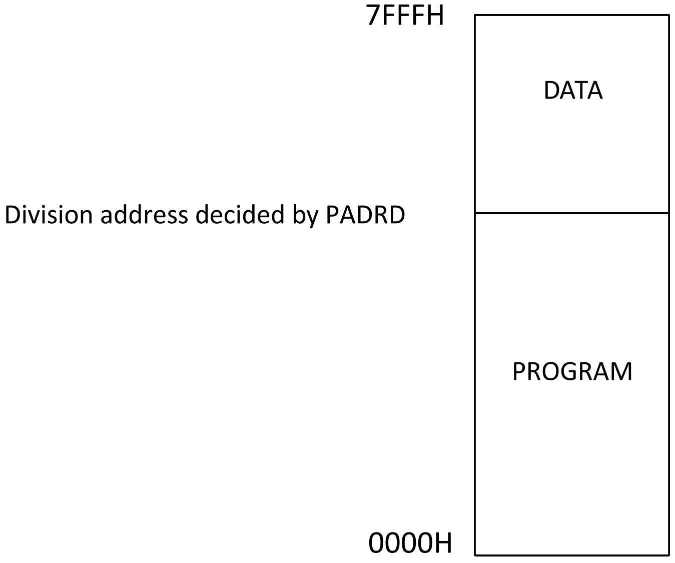


Figure 7-3-3 32K Flash Memory Structure

7.3.3 Flash Description

Table 7-3-3-1 Register MECON

FC00H	7	6	5	4	3	2	1	0
MECON	REMAP	DPSTB	-	-		-	BOOT[1:0]	
R/W	R/W	R/W					R/W	
Initial Value	0	0	-	-		-	0	0
Bit number	Bit symbol	Description						
7	REMAP	XRAM address mapping control 0：disable address mapping 1：enable address mapping, 2K XRAM mapping to program address apace (address:8000H~87FFH)						
6	DPSTB	Flash SLEEP mode control in IDLE/STOP mode 0：Flash in NORMAL mode while IDLE/STOP 1：Flash in SLEEP mode while IDLE/STOP *Note：If DPSTB=1, when the chip enters IDLE/STOP mode, the Flash will enter SLEEP mode simultaneously and the power consumption of the Flash in SLEEP mode is 50nA. When the chip exits IDLE/STOP mode, Flash exits SLEEP mode as well.*						
5~2	-	-						
1~0	BOOT	Programs start area control after soft reset 01：Program starts from XRAM after soft reset 10：Program starts from FLASH after soft reset						

Table 7-3-3-2 Register FSCMD

FC01H	7	6	5	4	3	2	1	0
FSCMD	IFEN	-	-	-	CMD[3:0]			
R/W	R/W	-	-	-	R/W			
Initial Value	0	-	-	-	0	0	0	0
Bit number	Bit symbol	Description						
7	IFEN	Information area access enable bit, this bit is required for access						
6~4	-							
3~0	CMD	Command Register 0000: No operation 0100: Flash whole erase 0001: Read Flash data area 0010: Write Flash data area 0011: Sector erase Flash data area 1011: Block erase Flash data area (each block is 512 bytes) 0101: Read Flash program area 0110: Write Flash program area 0111: Sector erase Flash program area 1111: Block erase Flash program area (512 bytes per block) *Note：* *1. The CMD is automatically cleared to 0 after the erase command is executed.* *2. CMD remains unchanged after the read and write commands are written and then completed by reading and writing FSDAT.*						

Table 7-3-3-3 Register FSDAT

FC02H	7	6	5	4	3	2	1	0
FSDAT	FSDAT[7:0]							
R/W	R/W							
Initial Value	-	-	-	-	-	-	-	-
Bit number	Bit symbol	Description						
7~0	FSDAT	Flash data register						

Table 7-3-3-4 Register LOCK

FC03H	7	6	5	4	3	2	1	0
LOCK								
R					FLKF	PLKF	DLKF	ILKF
W	LOCK[7:0]							
Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit symbol	Description						

Write		
7~0	LOCK	28H： Unlock Flash programmable area 29H： Unlock Flash PROGRAM area ： 2AH： Unlock Flash DATA area AAH： Lock Flash, R/W forbidden
Read		
7	-	
6	-	-
5	-	
4	-	
3	FLKF	Programmable area unlocked flag, 1 indicates unlocked
2	PLKF	PROGRAM area unlocked flag, 1 indicates unlocked
1	DLKF	DATA area unlocked flag, 1 indicates unlocked
-	-	-

Table 7-3-3-5 Register PADRD

FC04H	7	6	5	4	3	2	1	0
PADRD	PADRD[7:0]							
R/W	R/W							
Initial Value	1	0	0	0	0	0	0	0
Bit number	Bit symbol	Description						
7~0	PADRD	PROGRAM and DATA area division configuration register The unit for division is 256 bytes and when PADRD>0: The address space for PROGRAM area : 0 ~ (PADRD×256 - 1), The address space for DATA area : (PADRD×256) ~ 1FFFH/3FFFH/7FFFH. Note： 1.PADRD=0 indicates the whole Flash is DATA area 2.1FFFH/3FFFH/7FFFH is the maximum address for 8K/16K/32K Flash 3.The maximum value for PADRD is 20H/40H/80H corresponding to 8K/16K/32K Flash respectively. PADRD can not be set to any values greater than the maximum.						

Table 7-3-3-6 Register PTS

FC05H	7	6	5	4	3	2	1	0
PTSL	PTS[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
FC06H	7	6	5	4	3	2	1	0
PTSH		PTS[14:8]						
R/W	-	R/W						
Initial Value	-	0	0	0	0	0	0	0

Bit number	Bit symbol	Description
15	-	
14~0	PTS	target address pointer register

7.3.4 Flash Control Example

- **Divide Flash into DATA area and PROGRAM area**

For instance, if the user wants to divide a 32K Flash (1024 byte DATA area and the remains for PROGRAM area), the program may like this:

```
----------------------------------------------------------------------------------------
PADRD = 124; //The address for PROGRAM area will be 0~0x7BFF while the address for DATA area will be
0x7C00~0x7FFF
```

***Note**: This makes the physical address of the DATA area in FLASH 0x7C00~0x7FFF while the logical address is 0x0000~0x03FF. The logical address is used for DATA area’s R/W*

```
----------------------------------------------------------------------------------------
```

- **Sector erasure of DATA area**

Sector n of DATA area needs to be erased, for example, the program may as follows:

```
---------------------------------------------------------------------------------------
FSCMD = 0;          //set CMD=0
LOCK = 0x2A;        //unlock DATA area
PTSH = (unsigned char)((n*0x80)>>8); //set the higher bytes of the sector’s address
PTSL = (unsigned char)(n*0x80);          //set the lower bytes of the sector’s address
FSCMD = 3; //set clear
LOCK = 0xAA; //lock FLASH
----------------------------------------------------------------------------------------
```

- **Write data into DATA area**

For instance, write data 0xAA to DATA area of which address is n~(n+100), the program will be：

```
----------------------------------------------------------------------------------------
unsigned char i ；
FSCMD = 0;          //set CMD = 0
LOCK = 0x2A; //unlock DATA area
PTSH = (unsigned char)(n>>8); //set the higher 8 bits of data’s original address
PTSL = (unsigned char)n;          //set the lower 8 bits of data’s original address
FSCMD = 2; //set WRITE command
for(i=0;i<100;i++)
{
        FSDAT = 0xAA;      //write data continuously
}
FSCMO = 0;
LOCK = 0xAA;        //lock FLASH
----------------------------------------------------------------------------------------
```

Note：

1. When data is written continuously, only original address has be set. PTS will increase automatically after writing FSDAT each time.

2. For DATA area R/W, only the logical address of the DATA area which starts from 0 needs to be set, instead of *the physical address.*

- **Read data from DATA area**

For instance, the pointer pBuf reads data from DATA area of which address is n~(n+100), the program will be：

```
-----------------------------------------------------------------------------------------
unsigned char i,* pBuf ；
FSCMD = 0;           // set CMD = 0
LOCK = 0x2A; //unlock DATA area
PTSH = (unsigned char)(n>>8); //set the higher 8 bits of data’s original address
PTSL = (unsigned char)n;           //set the lower 8 bits of data’s original address
FSCMD = 1; //set READ command
for(i=0;i<100;i++)
{
        *pBuf++ = FSDAT ;//read data continuously
}
FSCMD = 0;
LOCK = 0xAA;       //lock FLASH
-----------------------------------------------------------------------------------------
```

Note：When data is written continuously, only original address has be set. PTS will increase automatically after writing FSDAT each time.

7.4 External RAM Mapped to Program Area

The 2K external RAM can be mapped as PROGRAM area as well. When REMAP=0 (for more you may refer to register MECON), the mapping address is 0000H~07FFH(soft reset is effective only when XRAM is running); when REMAP=1, the mapping address is 8000H~87FFH with the figure 7-5-1 below shows the mapping. Users may download the program to external RAM. When program is running, set REMAP = 1 and jump to mapping program area to execute. Similarly, users can set BOOT[1:0](please refer to register MECON) to 01, and then soft reset. The program starts from external RAM (the mapping address is 0000H~07FFH). Mapping program area offers convenience for IAP and so on.

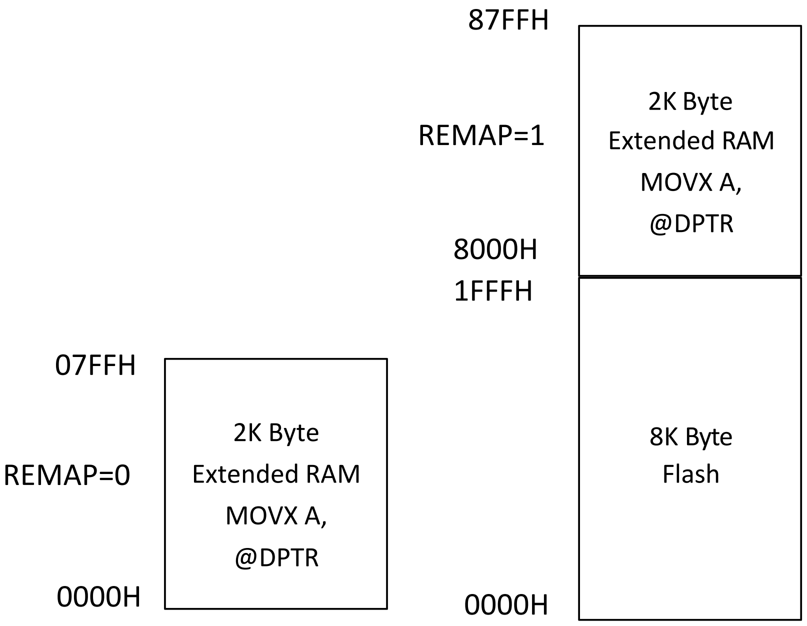


Figure 7-4-1 XRAM address mapping diagram (Flash capacity is 8K)

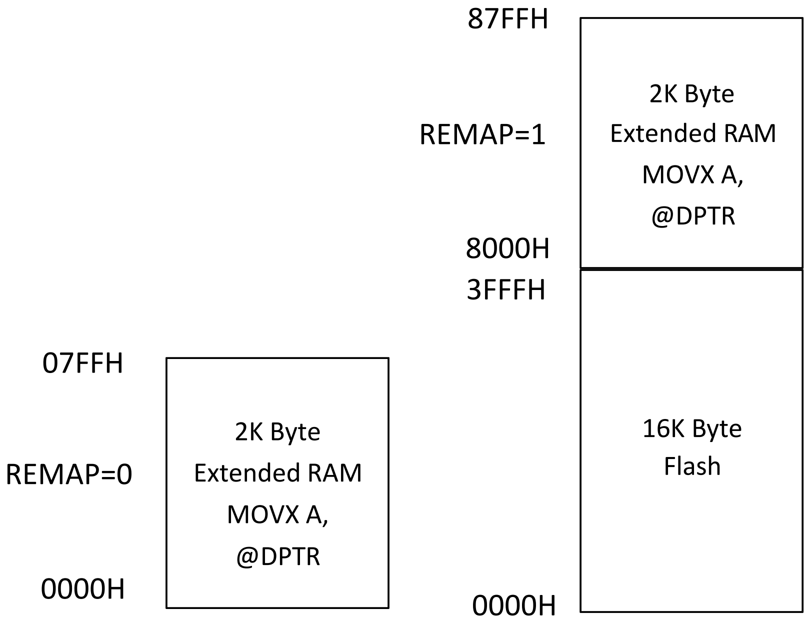


Figure 7-4-2 XRAM address mapping diagram (Flash capacity is 16K)

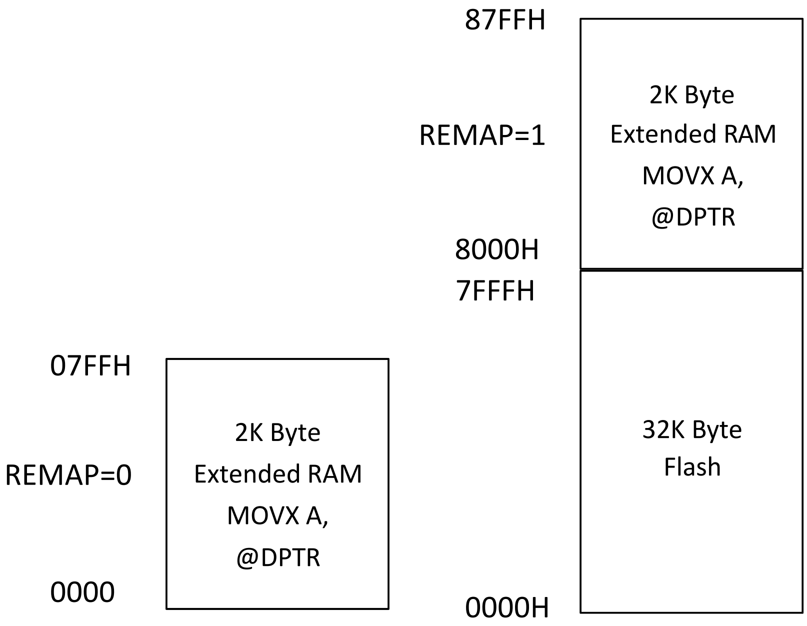


Figure 7-4-2 XRAM address mapping diagram (Flash capacity is 32K)

8 Interrupt System

8.1 Function Introduction

CA51F3 Series include a enhanced interrupt control system with 15 interrupt entries. For each interrupt entry, there are several interrupt sources with 2 level interrupt priorities for each source. Each interrupt source has its independent interrupt vector, priority setting, interrupt enable control and interrupt flag. CPU enters corresponding Interrupt Service Routine after responding to the interrupt. It will then returns to the former status after receiving RETI. If there are multiple valid sources requesting interrupts, CPU will respond sequentially according to the interrupt priority set before. If the sources share the same priority, CPU will respond according to their natural priority (from the smallest address to largest address of the interrupt entries).

8.2 Interrupt Logic

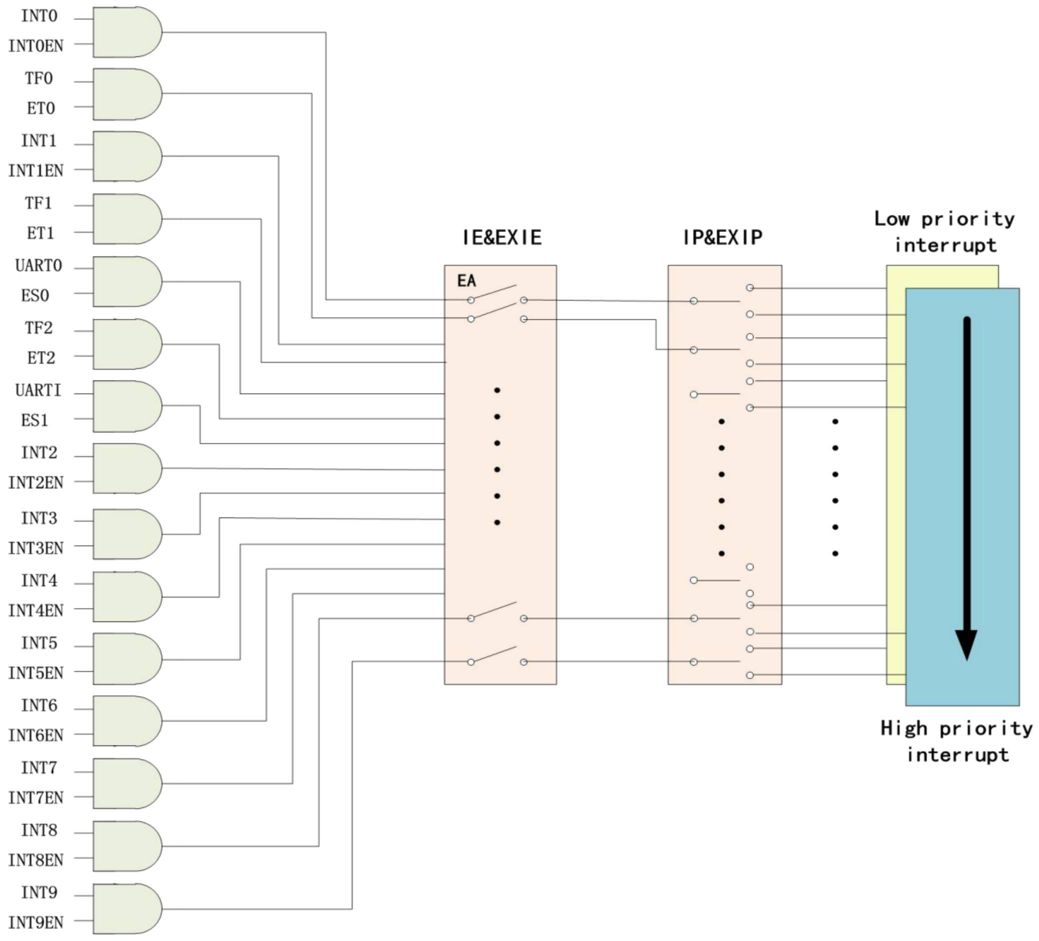


Figure 8-2-1 Interrupt Logic

8.3 Interrupt Vector Table

Table 8-3-1 Interrupt Vector Table

Interrupt	Interrupt source	Vector	Default Priority
INT0	INT0	03H	0
TF0	Timer 0	0BH	1
INT1	INT1	13H	2
TF1	Timer 1	1BH	3
UART0	UART0	23H	4
TF2	Timer 2	2BH	5
UART1	UART1	33H	6
INT2	ADC/External Interrupt 2	3BH	7
INT3	TK/External Interrupt 3	43H	8
INT4	LVD/External Interrupt 4	4BH	9
INT5	Clock Monitor/External Interrupt 5	53H	10
INT6	I2C/External Interrupt 6	5BH	11
INT7	WDT/External Interrupt 7	63H	12
INT8	RTC/External Interrupt 8	6BH	13
INT9	PWM/External Interrupt 9	73H	14

8.4 Interrupt Control Register

Table 8-4-1 Register IE

A8H	7	6	5	4	3	2	1	0
IE	EA	ES1	ET2	ES0	ET1	INT1EN	ET0	EX0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit symbol	Description
7	EA	Global Interrupt enable control 0：disable Global Interrupt 1：enable Global Interrupt
6	ES1	UART1 Interrupt enable control 0：disable UART1 Interrupt 1：enable UART1 Interrupt
5	ET2	Timer 2 Interrupt enable control

		0：disable Timer 2 Interrupt 1：enable Timer 2 Interrupt
4	ES0	UART0 Interrupt enable control 0：disable UART0 Interrupt 1：enable UART0 Interrupt
3	ET1	Timer 1 Interrupt enable control 0：disable Timer 1 Interrupt 1：enable Timer 1 Interrupt
2	EX1	External Interrupt 1 enable control 0：disable External Interrupt 1 1：enable External Interrupt 1
1	ET0	Timer 0 Interrupt enable control
0	EX0	External Interrupt 0 enable control 0：disable External Interrupt 0 1：enable External Interrupt 0

Table 8-4-2 Register EXIE

E8H	7	6	5	4	3	2	1	0
EXIE	INT9EN	INT8EN	INT7EN	INT6EN	INT5EN	INT4EN	INT3EN	INT2EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit symbol	Description
7	INT9EN	Interrupt 9 enable control(Interrupt 9 is used for PWM/External Interrupt 9) 0：Disable 1：Enable
6	INT8EN	Interrupt 8 enable control(Interrupt 8 is used for RTC/External Interrupt 8) 0：Disable 1：Enable
5	INT7EN	Interrupt 7 enable control(Interrupt 7 is used for WDT/External Interrupt 7) 0：Disable 1：Enable
4	INT6EN	Interrupt 6 enable control(Interrupt 6 is used for I2C/External Interrupt 6) 0：Disable 1：Enable
3	INT5EN	Interrupt 5 enable control(Interrupt 5 is used for Clock Monitor/External Interrupt 5) 0：Disable 1：Enable
2	INT4EN	Interrupt 4 enable control(Interrupt 4 is used for LVD/External Interrupt 4) 0：Disable 1：Enable

1	INT3EN	Interrupt 3 enable control(Interrupt 3 is used for TK/External Interrupt 3) 0：Disable 1：Enable
0	INT2EN	Interrupt 2 enable control(Interrupt 2 is used for ADC/External Interrupt 2) 0：Disable 1：Enable

Note: The enable controls of EXIE corresponds to Interrupt Vector which means the enable control for each interrupt source has to be set as well. For example, if External Interrupt 2 needs to be enabled, both INT2EN and EPIE2(External Interrupt 2 enable control) need to be set to 1.

Table 8-4-3 Register IP

B8H	7	6	5	4	3	2	1	0
IP	-	PS1	PT2	PS0	PT1	PX1	PT0	PX0
R/W	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit symbol	Description
7	-	-
6	PS1	UART1 priority control 0：low priority 1：high priority
5	PT2	Timer 2 priority control 0：low priority 1：high priority
4	PS0	UART0 priority control 0：low priority 1：high priority
3	PT1	Timer 1 priority control 0：low priority 1：high priority
2	PX1	External Interrupt 1 priority control 0：low priority 1：high priority
1	PT0	Timer 0 priority control 0：low priority 1：high priority
0	PX0	External Interrupt 0 priority control 0：low priority 1：high priority

Table 8-4-4 Register EXIP

F8H	7	6	5	4	3	2	1	0
EXIP	PX9	PX8	PX7	PX6	PX5	PX4	PX3	PX2
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit symbol	Description
7	PX9	Interrupt INT9 priority control 0：low priority 1：high priority
6	PX8	Interrupt INT8 priority control 0：low priority 1：high priority
5	PX7	Interrupt INT7 priority control 0：low priority 1：high priority
4	PX6	Interrupt INT6 priority control 0：low priority 1：high priority
3	PX5	Interrupt INT5 priority control 0：low priority 1：high priority
2	PX4	Interrupt INT4 priority control 0：low priority 1：high priority
1	PX3	Interrupt INT3 priority control 0：low priority 1：high priority
0	PX2	Interrupt INT2 priority control 0：low priority 1：high priority

8.5 External Interrupt

8.5.1 External Interrupt Introduction

For INT0 and INT1, arbitrary input ports can be selected as interrupt sources. The system also includes 8 extended Interrupt Entries INT2~INT9 as External Interrupt. For each interrupt entry, any input ports can be selected as interrupt source as well. Either rising or falling edge trigger can be selected independently for each Extended External Interrupt. The External Interrupt can also be waken up in STOP mode. The status register for INT2~INT9 External Interrupt is EPIF and the corresponding configuration register for INT2~INT9 which are EPCON0~EPCON7 also can be visited by setting the index of register (INDEX=0~7) .

Note: INT0 and INT1 can be triggered by rising edge or falling edge. The selection bits are it0 and it1 respectively. See the relevant description of register TCON for details.

8.5.2 External Interrupt Register

Table 8-5-2-1 Register IT0CON

8FH	7	6	5	4	3	2	1	0
IT0CON	-	-	IT0PS[5:0]					
R/W	-	-	R/W					
Initial Value	-	-	0	1	1	0	1	0

Bit number	Bit symbol	Description
7~6	-	-
5~0	IT0PS[5:0]	INT0 Interrupt pin selection The table for pin numbers and corresponding pins please refer to Table 8-5-2-6

Table 8-5-2-2 Register IT1CON

8EH	7	6	5	4	3	2	1	0
IT1CON	--	-	IT1PS[5:0]					
R/W	-	-	R/W					
Initial Value	-	-	1	1	0	1	1	1

Bit number	Bit symbol	Description
7~5	-	-
4~0	IT1PS[5:0]	INT1 Interrupt pin selection The table for pin numbers and corresponding pins please refer to Table 8-5-2-6

Table 8-5-2-3 Register EPIE

F9H	7	6	5	4	3	2	1	0
EPIE	EPIE9	EPIE8	EPIE7	EPIE6	EPIE5	EPIE4	EPIE3	EPIE2
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit symbol	Description
7	EPIE9	External Interrupt 9 enable control
6	EPIE8	External Interrupt 8 enable control
5	EPIE7	External Interrupt 7 enable control
4	EPIE6	External Interrupt 6 enable control
3	EPIE5	External Interrupt 5 enable control
2	EPIE4	External Interrupt 4 enable control
1	EPIE3	External Interrupt 3 enable control
0	EPIE2	External Interrupt 2 enable control

Table 8-5-2-4 Register EPIF

FAH	7	6	5	4	3	2	1	0
EPIF	EPIF9	EPIF8	EPIF7	EPIF6	EPIF5	EPIF4	EPIF3	EPIF2
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit symbol	Description
7	EPIF9	External Interrupt 9 Interrupt Flag, cleared to 0 when 1 is written to it
6	EPIF8	External Interrupt 8 Interrupt Flag, cleared to 0 when 1 is written to it
5	EPIF7	External Interrupt 7 Interrupt Flag, cleared to 0 when 1 is written to it
4	EPIF6	External Interrupt 6 Interrupt Flag, cleared to 0 when 1 is written to it
3	EPIF5	External Interrupt 5 Interrupt Flag, cleared to 0 when 1 is written to it
2	EPIF4	External Interrupt 4 Interrupt Flag, cleared to 0 when 1 is written to it
1	EPIF3	External Interrupt 3 Interrupt Flag, cleared to 0 when 1 is written to it
0	EPIF2	External Interrupt 2 Interrupt Flag, cleared to 0 when 1 is written to it

Table 8-5-2-5 Register EPCON

FBH	7	6	5	4	3	2	1	0
EPCON	EPPL	-	-	EPPS[5:0]				
R/W	R/W	-	-	R/W				
Initial Value	0	-	-	0	0	0	0	0
Note： EPCON is a register with index, INDEX=0~7 indicates EPCON0~EPCON7 respectively								
Bit number	Bit symbol	Description						
7	EPPL	External Interrupt Trigger Edge Selection 0： Rising edge 1： Falling edge						
6~5	-	-						
4~0	EPPS[4:0]	Interrupt Pin selection The table for pin numbers and corresponding pins please refer to Table 8-5-2-6						

Table 8-5-2-6 Index for Interrupt Pin

Pin name	Number
P00	0
P01	1
P02	2
P03	3
P04	4
P05	5
P06	6
P07	7
P10	8
P11	9
P12	10
P13	11
P14	12
P15	13
P16	14
P17	15
P20	16
P21	17
P30	24
P31	25
P32	26
P33	27
P34	28

P35	29
P36	30
P37	31

8.5.3 External Interrupt Control Method and Examples

- **External Interrupt 0/1 control example**

For instance, set P20 as the input pin for External Interrupt 0 and enable External Interrupt0, the program will be:

```
---------------------------------------------------------------------------------------
void INT0_init(void)
{
        P20F = 1;         //set P20 as input pin
        IT0CON = 16; //set P20 as the pin for Interrupt 0 ( 16 is the corresponding index number for
        P20) EX0 = 1;   //enable INT0 interrupt
        IE0 = 1;          //enable External
        Interrupt 0 IT0 = 1;    //set falling edge
        trigger
        PX0 = 1;              //set INT0 with high priority
        EA = 1;                  //enable Global Interrupt
}
void INT0_ISR (void) interrupt 0
{
        //External Interrupt0 Interrupt Service Routine
}
---------------------------------------------------------------------------------------
```

For instance, set P20 as the input pin for External Interrupt 1 and enable External Interrupt 1, the program will be:

```
---------------------------------------------------------------------------------------
void INT1_init(void)
{
        P20F = 1;         // set P20 as input pin
        IT1CON = 16;   //set P20 as the pin for Interrupt 1(16 is the corresponding index number for P20)
        EX1 = 1;          //enable INT1 interrupt
        IE1 = 1;          //enable External
        Interrupt 1 IT1 = 1;          //set falling
        edge trigger
        PX1 = 1;              //set INT0 with high priority
        EA = 1;                  //enable Global Interrupt

        }
void INT1_ISR (void) interrupt 2
{
        //External Interrupt 1Interrupt Service Routine
}

---------------------------------------------------------------------------------------
```

- **External Interrupt 2~9 control example**

Taking External Interrupt 2 for example, if P20 is set as the input pin for External Interrupt 2 and External Interrupt 2 is enable, the program may like this:

```
-----------------------------------------------------------------------------------------
void INT2_init(void)
{
        P20F = 1;         //set P20 as input pin
        INDEX = 0;              //set the index number for EPCON, 0~7 indicates External Interrupt 2~9
        EPCON = (0<<7) | 16;      //set rising edge trigger and set the index number for the interrupt pin (16 indicates P20)
                                  //To set falling edge trigger the codes may be EPCON = (1<<7) |
        16; INT2EN = 1; //enable INT2 interrupt
        EPIE |= 0x01; //enable External Interrupt 2
        EA = 1;                 //enable Global
        Interrupt
}
void INT2_ISR (void) interrupt 7
{
        if(EPIF & 0x01)               //judge the Interrupt Flag for External Interrupt 2
        {
               EPIF = 0x01; //write 1 to the Interrupt Flag to clear 0
                    //External Interrupt 2 Interrupt Service Routine
                ......
        }
}
-----------------------------------------------------------------------------------------
```

9 Clock System

9.1 Clock System Introduction

The clock system includes the system clock generation, frequency division and assignment. CA51F3 Series has several clock sources as follows：

- 2 - 4 MHz Internal RC Oscillator
- 131 KHz Internal RC Oscillator
- 4MHz Internal RC Oscillator
- Supports 1 - 27 MHz external RC resonator
- Supports 32.768 KHz external Crystal Resonator
- 2 - 10 times PLL

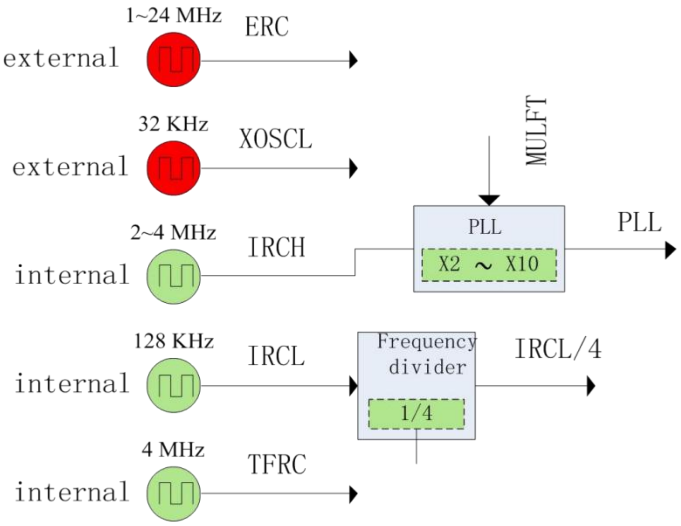


Figure 9-1-1 Clock Sources

Users can control the clock sources independently. They can disable or enable any of the clock sources in order to manage the power consumption flexibly.

All the clock sources can be set as system alarm clock and assigned to various peripherals as their clock sources. For more information you may refer to the Peripherals part.

9.1.1 Clock Special Name Definition

Symbol	Description	Symbol	Description
IRCH	2 - 4 MHz Internal RC Oscillator	ERC	External RC Oscillator
IRCL	131 KHz Internal RC Oscillator	TFRC	4MHz Internal RC Oscillator
XOSCL	32.768 KHz External Crystal Resonator	PLL	2~10 times PLL

9.1.2 2 - 4 MHz Internal RC Oscillator(IRCH)

IRCH is the default the system clock after Power On and can be enabled or disabled by setting the bit IHCKE of the register CKCON. The frequency of IRCH is 2~4MHz which can be set by using the register IHCFGH and IHCFGL. It can also be modified by Internal RC Correction Module by taking other clock sources as reference clock. The precision can be 1% and factory frequency is 3.6864MHz@3.3V/25℃.

Note：*Due to the difference between the manufacturing process of the chips, the IRCH frequency may vary even the value set for IHCFGH, IHCFGL is the same. It is the same for IRCL and TFRC.*

9.1.3 32.768 KHz External Crystal Resonator(XOSCL)

XOSCL is mainly used as the clock source for RTC for real time timing. With the lowest power consumption even below 13uA, XOSCL is usually set as system clock when low power consumption is needed. It is enabled/disabled by setting the bit XLCKE of register CKCON. The time to start oscillation of XOSCL is not short which means it usually takes about 1 second, hence users have to wait until it is completely stabilized. The bit XLSTA of register CKCON is the flag for XOSCL clock stabilization.

9.1.4 131 KHz Internal RC Oscillator(IRCL)

IRCL can be enabled/disabled by setting the ILCKE of register CKCON. Similar to XOSCL, when IRCL is set as system clock the power consumption decrease as well. If without 32.768 KHz crystal oscillator, IRCL can also be set as the clock source for RTC module when there is no need for high accuracy. The frequency of IRCL can be set by using the register ILCFGH and ILCFGL. It can also be modified by Internal RC Correction Module by taking other clock sources as reference clock with factory frequency at 131 KHz@3.3V/25℃

9.1.5 4 MHz RC Internal Oscillator(TFRC)

TFRC can be enabled/disabled by setting the TFCKE of register CKCON. It is mainly used as working frequency for Flash and clock for Touch Module when charging/discharging. The frequency of TFRC can be set by using the register TFCFG. It can also be modified by Internal RC Correction Module by taking other clock sources as reference clock with the factory frequency at 4 MHz@3.3V/25℃.

9.1.6 PLL

2~10 times internal PLL takes IRCH as its reference clock so the chip can operates with high speed even without external high speed crystal oscillator. The register PLLCON can be used to enable/disable PLL and set its ratio. When PLL is enabled, users still have to wait until it is stable and then the PLL can be set as system clock or clock for other peripherals. The bit PLSTA of register PLLCON indicates whether PLL clock is stable or not. It usually takes about 50us for PLL to become stable.

Note：Due to the maximum frequency for CPU is 27 MHz, PLL can not be set as CPU's clock when its frequency is greater than 27MHz, but it still can be set as the clock for other peripherals.

9.1.7 External RC Oscillator(ERC)

ERC can be enabled/disabled by XHCKE of register CKCON.

9.2 Clock Control Register Description

Table 9-2-1 Register CKCON

A1H	7	6	5	4	3	2	1	0
CKCON	ILCKE	IHCKE	TFCKE	-	XLCKE	XLSTA	XHCKE	-
R/W	R/W	R/W	R/W	-	R/W	R	R/W	-
Initial Value	0	0	0	-	1	0	0	-

Bit number	Bit Symbol	Description
7	ILCKE	IRCL enable control 1：enable 0：disable *Note：* *When it is 1, the clock is enabled;* *when it is 0, if the system or other modules selected this clock source, the clock is still enabled.*
6	IHCKE	IRCH enable control 1：enable 0：disable *Note：* *When it is 1, the clock is enabled;* *when it is 0, if the system or other modules selected this clock source, the clock is still enabled.*
5	TFCKE	TFRC enable control 1：enable 0：disable *Note：* *When it is 1, the clock is enabled;* *when it is 0, if the system or other modules selected this clock source, the clock is still enabled.*
-	-	-
3	XLCKE	XOSCL enable control 1：enable 0：disable *Note：* *1. When it is 1, the clock is enabled;* *when it is 0, if the system or other modules selected this clock source, the clock is still enabled.* *2 Since XOSCL is external clock, the corresponding pin function must be set as XOSCL function to use it*
2	XLSTA	XOSCL clock stabilization flag (1 indicates it is stabilized)

1	XHCKE	ERC enable control 1： enable 0： disable *Note：* *1. When it is 1, the clock is enabled;* *when it is 0, if the system or other modules selected this clock source, the clock is still enabled.* *2 Since ERC is external clock, the corresponding pin function must be set as ERC function to use it*
0	-	-

Table 9-2-2 Register PLLCON

AEH	7	6	5	4	3	2	1	0
PLLCON	PLLON	MULFT[3:0]				-	-	PLSTA
R/W	R/W	R/W	R/W	R/W	R/W	-	-	R
Initial Value	0	0	0	0	0	-	-	0
Bit number	Bit Symbol	Description						
7	PLLON	PLL enable control 0： disable 1： enable						
6~3	MULFT	PLL ratio set 0000: 2 times 0001: 3 times 0010: 4 times 0011: 5 times 0100: 6 times 0101: 7 times 0110 :8 times 0111: 9 times 1000: 10 times Others： Invalid						
2~1	-	-						
0	PLSTA	PLL clock stabilization flag, 1 indicated it is stabilized						

Table 9-2-3 Register IHCFGL, IHCFGH

A4H	7	6	5	4	3	2	1	0
IHCFGL	IHCFG[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	1	1	1	1	1	1	1
A5H	7	6	5	4	3	2	1	0
IHCFGH	IHCFG[15:8]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	-	-	-	-	1	0	0	0

Bit number	Bit Symbol	Description
11~0	IHCFG	IRCH frequency correction register

Table 9-2-4 Register ILCFGL, ILCFGH

8085H	7	6	5	4	3	2	1	0
ILCFGL	ILCFG[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0
A7H	7	6	5	4	3	2	1	0
ILCFGH	-	-	-	-	-	-	-	ILCFG[8]
R/W	-	-	-	-	-	-	-	R/W
Initial Value	-	-	-	-	-	-	-	1

Bit number	Bit Symbol	Description
8~0	ILCFG	IRCL frequency correction register

Table 9-2-5 Register TFCFG

8087H	7	6	5	4	3	2	1	0
TFCFG	TFCFG[7:0]							
R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	-	-	1	0	0	0	0	0

Bit number	Bit Symbol	Description
7~0	TFCFG	TFRC frequency correction register *Note: the value automatically loaded after power on of this register corresponds to the frequency of 3.6864MHz. It is not recommended to modify this value except for special applications.*

9.3 System Clock

All of the clock sources in CA51F3 Series can be set as system clock. The system clock is controlled by register CKCON, CKSEL and CKDIV. Users can disable/enable any of these clock sources, divide the frequency, change the system clock and so on by using these registers.

9.3.1 System Clock Architecture

Please refer to figure 9-3-1.

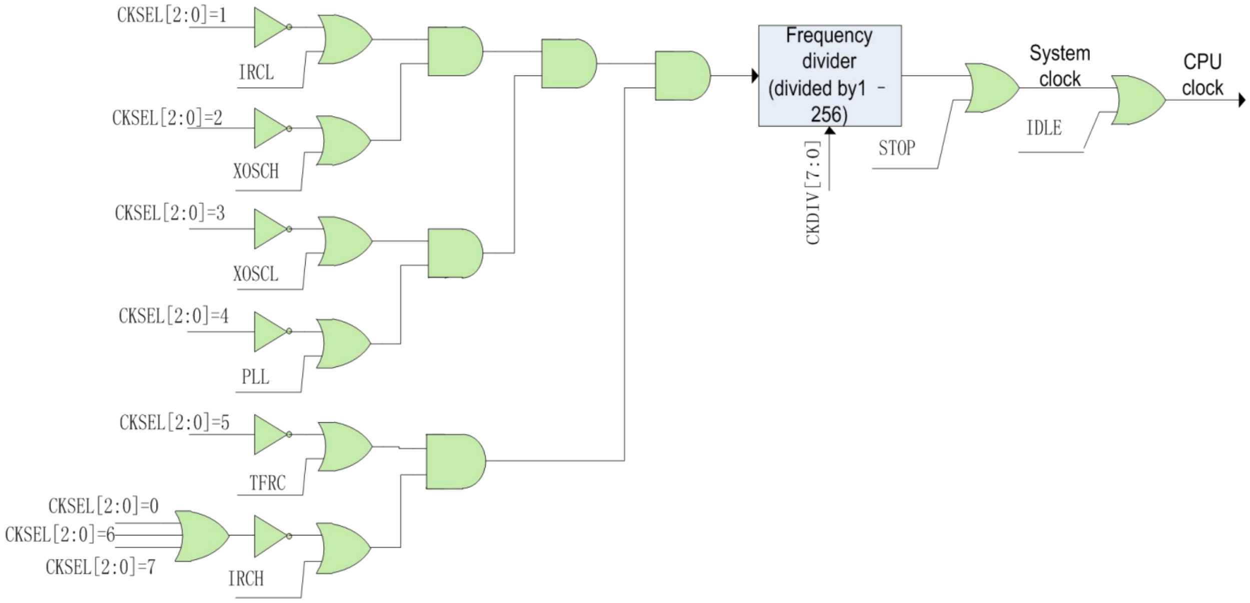


Figure 9-3-1 System Clock Architecture

9.3.2 System Clock Control Register Description

Table 9-3-2-1 Register CKSEL

A2H	7	6	5	4	3	2	1	0
CKSEL	RTCKS	-	-	-	-	CKSEL[2:0]		
R/W	R/W	-	-	-	-	R/W		
Initial Value	0	-	-	-	-	0	0	0
Bit number	Bit Symbol	Description						
7	RTCKS	RTC clock selection 0：XOSCL 1：IRCL *Note： when IRCL is selected, the clock's frequency will be divided by 4 first and then used for RTC*						
6~3	-	-						

2~0	CKSEL	System clock selection: 000：IRCH 001：IRCL 010：ERC 011：XOSCL 100：PLL 101：TFRC Others： IRCH *Note: if the IRCL is set as the system clock, you must wait about 1ms after enabling the IRCL clock before switching to the system clock, otherwise exceptions may occur.*

Table 9-3-2-2 Register CKDIV

A3H	7	6	5	4	3	2	1	0
CKDIV	CKDIV[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~0	CKDIV	System clock frequency division： 00H：No division 01H：frequency divided by 2 02H：frequency divided by 3 03H：frequency divided by 4 …… FFH：frequency divided by 256

9.3.3 System Clock Control Method and Example

- **Set IRCH as the system clock**

  To set IRCH as the system clock. The program is as follows:

```
-------------------------------------------------------------------------------------
#define IHCKE      (1<<6)
#define CKSEL_IRCH        0 void Sys_Clk_Set_IRCH(void)
{
    CKCON |= IHCKE; //enable IRCH
    CKSEL = (CKSEL&0xF8) | CKSEL_IRCH;      //set IRCH as system clock
}
-------------------------------------------------------------------------------------
```

- **Set XOSCL as the system clock**

  To set XOSCL as the system clock. The program is as follows:

```
----------------------------------------------------------------------------------------
#define XLCKE      (1<<3)
#define XLSTA      (1<<2)
#define CKSEL_XOSCL 3
void Sys_Clk_Set_XOSCL(void)
{
    P32F = 3;//set P32 and P33 as crystal oscillator pin
    P33F = 3;
    CKCON |= XLCKE;          //enable XOSCL
    while(!(CKCON & XLSTA)); //wait for XOSCL stabilization
    CKSEL = (CKSEL&0xF8) | CKSEL_XOSCL;//set XOSCL as system clock
}
----------------------------------------------------------------------------------------
```

- **Set IRCL as the system clock**

  To set IRCL as the system clock. The program is as follows:

```
----------------------------------------------------------------------------------------
#define ILCKE      (1<<7)
#define CKSEL_IRCL        1 void Sys_Clk_Set_IRCL(void)
{
    CKCON |= ILCKE;          //enable IRCL
    Delay_ms(1);// Delay 1ms after enabling IRCL and wait for IRCL clock to stabilize
    CKSEL = (CKSEL&0xF8) | CKSEL_IRCL;      //set IRCL as system clock
}
----------------------------------------------------------------------------------------
```

- **Set PLL as the system clock**

To set PLL as the system clock. The program is as follows:

```
-----------------------------------------------------------------------------------
#define IHCKE      (1<<6)

// Register RCCON definition
#define PLLON(N)            (N<<7)             //N=0~1
#define MULFT(N)            (N<<3)             //N=0~8
#define PLSTA               (1<<0)
#define CKSEL_PLL           4

void Sys_Clk_Set_PLL(unsigned char Multiple)         //Multiple times
{
            if(Multiple < 2 || Multiple > 8) return;
            CKCON |= IHCKE;
            PLLCON = PLLON(1) | MULFT(Multiple-2);
            while(!(PLLCON & PLSTA));
            CKSEL = (CKSEL&0xF8) | CKSEL_PLL;
}
-------------------------------------------------------------------------------------
```

- **Set TFRC as the system clock**

To set TFRC as the system clock. The program is as follows:

```
-------------------------------------------------------------------------------------
#define TFCKE      (1<<5)
#define CKSEL_TFRC          5 void Sys_Clk_Set_TFRC(void)
{
   CKCON |= TFCKE;
   CKSEL = (CKSEL&0xF8) | CKSEL_TFRC;
}
-------------------------------------------------------------------------------------
```

9.4 Internal RC Oscillator Correction

9.4.1 Correction Module Introduction

Due to difference between manufacturing process, the factory frequency for RC oscillator, so correction is a must for RC oscillators. There is correction module embedded in CA51F3 Series chip which can corrects internal RC oscillators. There are 5 reference clock sources (IRCH、IRCL、XOSCL、ERC、TFRC) and 3 target clock sources (IRCH 、 IRCL 、 TFRC) for correction module. The reference clock source must be correct otherwise the target clock(the clock to be corrected) will be influenced. Figure 9-4-1 shows the circuit for correction module (VCLK is the reference clock and TCLK is the target clock).

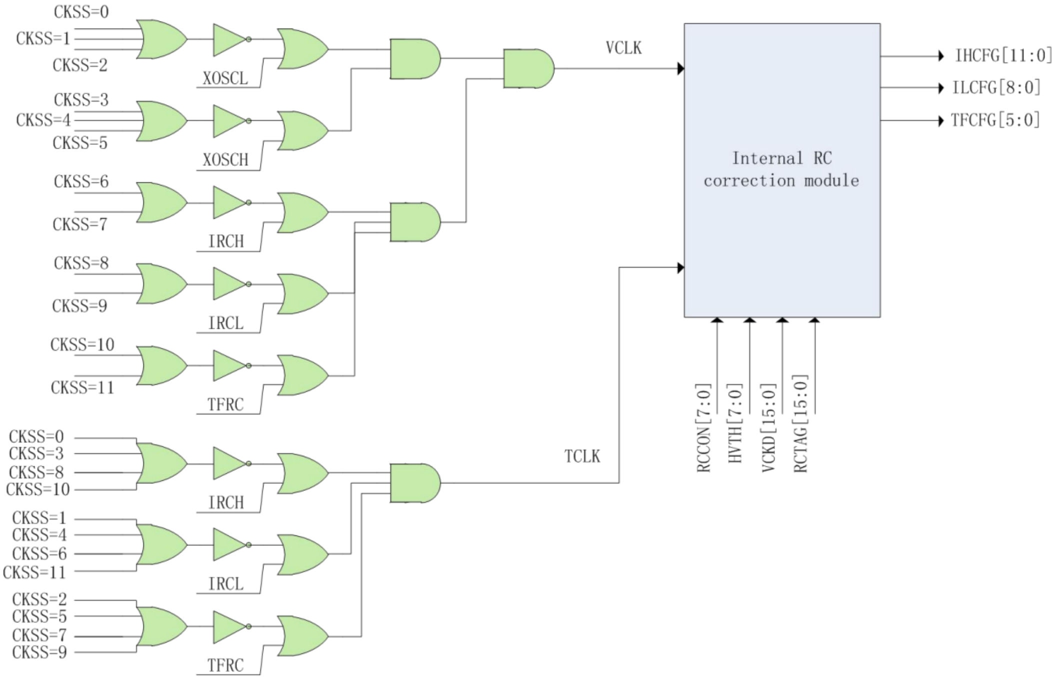


Figure 9-4-1 Correction Circuit Schematic

There are three working modes for RC correction module:

- **Count mode**

This mode is mainly used for measure TCLK manually. Setting MODE(RCCON[7:6]) to 1 enables TCLK count and to 0 disables the count function. The count will be stored in register RCMS(RCMSHH/RCMSHL/RCMSLH/RCMSLL). Users may enable TCLK count in certain time and the get its frequency by calculating the count value and time.

- **Measure mode**

In Measure mode, TCLK is counted in several VCLK clock cycles and the frequency is deduced using the count get. MODE is set to 2 to start the measurement and the count will be stored in register RCMS after measurement with MODE cleared to 0 automatically. To get better precision, the time cycle for the measure should be as long as possible. It can be set by using the register VCKD(VCKDH/VCKDL)which means the measurement cycle will be VCKD times longer than VCLK's cycle. Hence the frequency of TCLK can be calculated as follows：

TCLK's cycle ＝ (VCLK's cycle × VCKD) ÷ RCMS

- **Correction Mode**

The correction mode uses dichotomy and compares the TCLK count with certain value (corresponding to the target frequency) constantly. When the difference reaches the threshold HVTH which is set by users, there will be a HMSK flag. By setting the MSE, users can decide whether the correction stops or the dichotomy continues till end when HMSK=1. Lower HVTH usually means better precision but longer time taken. Thus, users are recommended to set register HVTH according to their acceptance to TCLK clock's frequency error. Setting MODE to 3 enables the correction and MODE is cleared after the correction. The single cycle time should be as long as possible to to get better the precision while it also makes the time for correction increase. The relationship between VCLK and TCLK is：

VCLK cycle × VCKD ＝ TCLK target cycle × RCTAG

9.4.2 Correction Module Control Register

Table 9-4-2-1 Register RCCON

91H	7	6	5	4	3	2	1	0
RCCON	MODE[1:0]		MSE	HMSK	CKSS[3:0]			
R/W	R/W		R	R	R/W			
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~6	MODE	Working mode selection 01：count mode, setting MODE to 0 will exits count mode 10：measure mode, MODE cleared after completion 11：correction mode, MODE cleared after completion
5	MSE	Exit selection in correction mode 0：wait for the correction ends even HMSK=1 detected 1：Exit when HMSK=1
4	HMSK	The difference between the corrected value and target value is less than HVTH in correction mode, the HMSK will be set to 1, otherwise it is 0
3~0	CKSS	Clock matching selection 0000：target clock IRCH, reference clock XOSCL 0001：target clock IRCL, reference clock XOSCL 0010：target clock TFRC, reference clock XOSCL

		0011：target clock IRCH, reference clock ERC 0100：target clock IRCL, reference clock ERC 0101：target clock TFRC, reference clock ERC 0110：target clock IRCL, reference clock IRCH 0111：target clock TFRC, reference clock IRCH 1000：target clock IRCH, reference clock IRCL 1001：target clock TFRC, reference clock IRCL 1010：target clock IRCH, reference clock TFRC 1011：target clock IRCL, reference clock TFRC Others：invalid

Table 9-4-2-2 Register HVTH

AFH	7	6	5	4	3	2	1	0
HVTH	HVTH[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
7~0	HVTH	Threshold register for the difference between corrected value and target value in correction mode						

Table 9-4-2-3 Register VCKDL, VCKDH

92H	7	6	5	4	3	2	1	0
VCKDL	VCKD[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
93H	7	6	5	4	3	2	1	0
VCKDH	VCKD[15:8]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
15~0	VCKD	The times reference clock will be multiplied in measure and correction mode (VCKD>1)						

Table 9-4-2-4 Register RCTAGL, RCTAGH

94H	7	6	5	4	3	2	1	0
RCTAGL	RCTAGL[7:0]							
R/W	R/W							

Initial Value	0	0	0	0	0	0	0	0
95H	7	6	5	4	3	2	1	0
RCTAGH	RCTAGH[15:8]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
15~0	RCTAG	The times target clock will be multiplied in correction mode (RCTAG>=1)						

Table 9-4-2-5 Register RCMSLL,RCMSLH, RCMSHL, RCMSHH

96H	7	6	5	4	3	2	1	0
RCMSLL	RCMS[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
97H	7	6	5	4	3	2	1	0
RCMSLH	RCMS[15:8]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
9EH	7	6	5	4	3	2	1	0
RCMSHL	RCMS[23:16]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
9FH	7	6	5	4	3	2	1	0
RCMSHH	RCMS[31:24]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
31~0	RCMS	To store the count when count mode ends To store the result when measure mode ends Meaningless in correction mode						

9.4.3 Correction Module Control Example

- **IRCH Correction**

    To set XOSCL as the reference clock and IRCH as target clock with correction target frequency 3.6864MHz, the program is：

```
    ------------------------------------------------------------------------------------------
    #define IHCKE           (1<<6)
    #define XLCKE           (1<<3)
    #define XLSTA           (1<<2)

    #define MODE(N)         (N<<6)      //N=0~3
    #define MSEX(N)         (N<<5)      //N=0~1
    #define CKSS(N)         N           //N=0~11

    CKCON |= IHCKE;         //enable IRCH clock
    CKCON |= XLCKE;         //enable XOSCL clock
    while(!(CKCON & XLSTA)); //wait until XOSCL clock becomes stable
    RCTAGH =     ((3686400*400)/32768)/256; //set target frequency
    RCTAGL =     ((3686400*400)/32768)%256;
    VCKDH = 400/256;                 //set the reference clock frequency, the frequency after division is
    81.92HZ VCKDL = 400%256;
    RCCON = MODE(3) | MSEX(0) | CKSS(0); //set IRCH as target clock and XOSCL as reference clock , and set the
details for correction mode
                                                       //enable correction
    mode while(RCCON&0xC0); //wait for the correction ends
    ------------------------------------------------------------------------------------------
```

- **IRCL Correction**

    To set XOSCL as the reference clock and IRCL as target clock with correction target frequency 131KHz, the program is：

```
   ------------------------------------------------------------------------------------------
  #define ILCKE  (1<<7)
  #define XLCKE           (1<<3)
  #define XLSTA           (1<<2)

  #define MODE(N)         (N<<6)      //N=0~3
  #define MSEX(N)         (N<<5)      //N=0~1
  #define CKSS(N)         N           //N=0~11

  CKCON |= ILCKE;                 //enable IRCL clock
  CKCON |= XLCKE;                 //enable XOSCL clock
 while(!(CKCON & XLSTA)); //wait until XOSCL clock becomes stable
 RCTAGH = ((131000*400)/32768)/256;  //set target frequency
```

```
RCTAGL =        ((131000*400)/32768)%256;
VCKDH = 400/256;                //set the reference clock frequency, the frequency after division is
81.92HZ VCKDL = 400%256;
     RCCON = MODE(3) | MSEX(0) | CKSS(1); //set IRCL as target clock and XOSCL as reference clock ,  and set
the details for correction mode
                                              //enable correction
mode while(RCCON&0xC0); //wait for the correction ends
 ------------------------------------------------------------------------------------
```

- **TFRC Correction**

To set XOSCL as the reference clock and TFRC as target clock with correction target frequency 4MHz, the program is：

```
------------------------------------------------------------------------------------
#define TFCKE              (1<<5)
#define XLCKE              (1<<3)
#define XLSTA              (1<<2)

#define MODE(N)            (N<<6)        //N=0~3
#define MSEX(N)            (N<<5)        //N=0~1
#define CKSS(N)            N
                           //N=0~11

CKCON |= TFCKE;                      //enable TFRC clock
CKCON |= XLCKE;                      //enable XOSCL clock
while(!(CKCON & XLSTA)); //wait until XOSCL clock becomes stable
RCTAGH =       ((4000000*400)/32768)/256; //set target frequency
RCTAGL =       ((4000000*400)/32768)%256;
VCKDH = 400/256;                     //set the reference clock frequency, the frequency after division is
81.92HZ VCKDL = 400%256;
RCCON = MODE(3) | MSEX(0) | CKSS(2); //set IRCH as target clock and XOSCL as reference clock , and set the
details for correction mode
                                              //enable correction
mode while(RCCON&0xC0); //wait for the correction ends
------------------------------------------------------------------------------------
```

9.5 External Clock Monitor

9.5.1 Function Introduction

The Clock Monitor module is used to monitor anomalies and handle them to increase the reliability of the system. If an external clock is set as the system clock but it stops, the system clock will change to IRCL when the Clock Monitor module enabled. Similarly, when the Clock Monitor module enabled, if an external clock is set as the RTC or WDT clock but it stops, the RTC or WDT clock will change to IRCL with frequency divided by 4.

The external high speed clock (ERC) monitor can be enabled by setting MHE and the interrupt enable is controlled by IHE. When ERC is abnormal, the clock abnormality flag XHFD=1. If ATH = 1, the clock source return to ERC as soon as ERC is normal again； when ATH=0, as long as the interrupt flag XHFD is cleared, the clock source will return back to ERC.

External low speed clock (XOSCL) monitor is enabled by MLE and the interrupt enable is controlled by ILE. When XOSCL is abnormal, the clock abnormality flag XLFD=1. If ATL = 1, the clock source return to XOSCL as soon as XOSCL is normal again；when ATL=0, as long as the interrupt flag XLFD is cleared, the clock source will return back to XOSCL.

Flag HSP and LSP indicates the current status of ERC and XOSCL respectively. HSP=1 or LSP=1 shows ERC or XOSCL is abnormal respectively and the clock switches to internal RC clock.

The system can also be waken up in STOP or IDLE mode.

9.5.2 External Clock Monitor Control Register

Table 9-5-2-1 Register CKMON

BEH	7	6	5	4	3	2	1	0
CKMON	MHE	IHE	ATH	HSW	MLE	ILE	ATL	LSW
R/W	R/W	R/W	R/W	W	R/W	R/W	R/W	W
Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
7	MHE	ERC clock monitor control, 1 enables it						
6	IHE	ERC clock monitor interrupt enable control, 1 enables it						
5	ATH	ERC automatic recovery enable control, 1 enables it Note： When ERC abnormality is detected, the corresponding circuit’s clock will change to IRCL. If						

		ATH = 1, the corresponding circuit's clock will change back to ERC as long as ERC operates normally again; if ATH=0, users must write 1 to HSW to make ERC operates as the corresponding circuit's clock again;
4	HSW	Write only, writing 1 to in will clear HSP(CKMIF[7])
3	MLE	XOSCL clock monitor control, 1 enables it
2	ILE	XOSCL clock monitor interrupt enable control, 1 enables it
1	ATL	XOSCL automatic recovery enable control, 1 enables it Note： When XOSCL abnormality is detected, the corresponding circuit's clock will change to IRCL. If ATL = 1, the corresponding circuit's clock will change back to XOSCL as long as XOSCL operates normally again; if ATL=0, users must write 1 to LSW to make XOSCL operates as the corresponding circuit's clock again;
0	LSW	Write only, writing 1 to in will clear LSP(CKMIF[6])

Table 9-5-2-2 Register CKMIF

BFH	7	6	5	4	3	2	1	0
CKMIF	HSP	LSP	-	-	-	-	XHFD	XLFD
R/W	R	R	-	-	-	-	R	R
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7	HSP	ERC abnormal and clocks switched to internal clock flag, 1indicate it is using internal clock
6	LSP	XOSCL abnormal and clocks switched to internal clock flag, 1indicate it is using internal clock
5~2	-	-
1	XHFD	ERC abnormality interrupt flag, 1 indicates the abnormality. When 1 is written to it, it will be cleared to 0
0	XLFD	XOSCL abnormality interrupt flag, 1 indicates the abnormality. When 1 is written to it, it will be cleared to 0

10 Power Supply and Reset System

10.1 Power Supply

There is 1.8V - 5.5V source between VDD pin and VSS pin for CA51F3 series which supplies the power for the chip. VDD and LDO supply power for the analog system and LDO supplies power for the digital system.

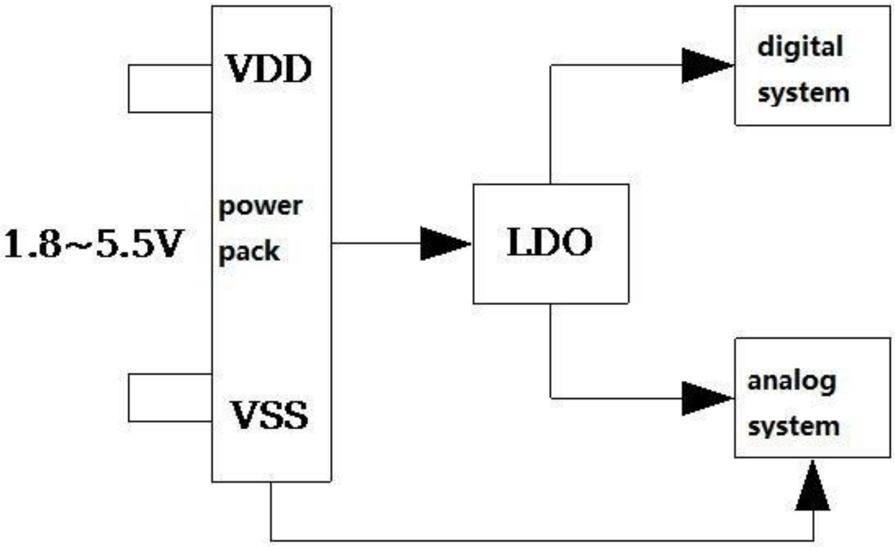


Figure 10-1-1 Power Supply Architecture

The Fig10-1-2 is the typical circuit for power supply

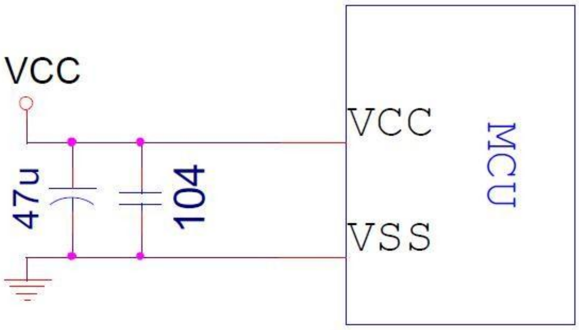


Figure 10-1-2 Typical Circuit for Power Supply

Note： 1.The filter capacitors 47uF and 104 in the circuit below are the standard devices for the chip which can not be omitted, otherwise the chip may operates abnormally.

2. The above circuit and component parameters are for reference only, and may need to be modified according to the peripheral working environment and different voltage power supply parameters.

10.1.1 LDO Function Introduction

There is an internal low dropout regulator (LDO) for CA51F3 Series chip. LDO module offers supply voltage for the chip. The output voltage of LDO is set by VLEVEL (PWCON[2:0]) and the default value for VLEVEL is 5, which implies the default voltage is 1.61V. When VDD/VSS is less than the output voltage set by VLEVEL, the output voltage will be VDD directly; when VDD/VSS is greater than the output voltage set by VLEVEL, LDO output the voltage set by VLEVE. High LDO voltage is benefits to clock module's rapid start while low LDO voltage will lower the chip's power consumption. There are two working modes for LDO: High Power mode and Low Power mode, which is selected by VHL (PWCON[3]). The load capacity is also different in different modes. The current is higher in High Power mode but with higher power consumption, while in Low Power mode it is vice versa. For the most time when the system is operating normally, LDO is usually High Power mode. The Low Power mode is usually used for Power Save Modes such as STOP, IDLE, Low Speed Mode etc.

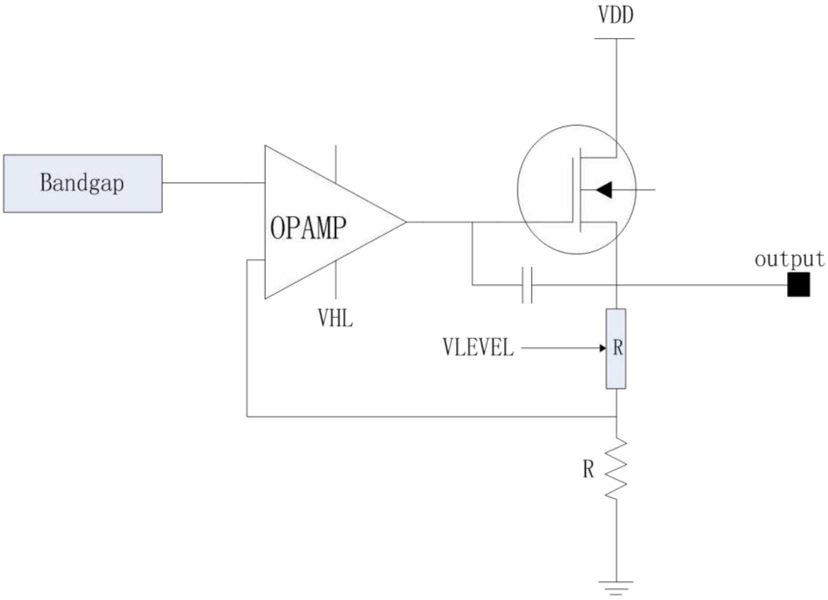


Figure 10-1-3 LDO Module Schematic

10.1.2 LDO Control Register

Table 10-1-2-1 Register PWCON

86H	7	6	5	4	3	2	1	0
PWCON	FLEVEL[3:0]				VHL	VLEVEL[2:0]		
R/W	R/W				R/W	R/W		
Initial Value	0	1	1	1	1	1	0	1
Bit number	Bit Symbol	Description						
7~4	FLEVEL	Internal reference voltage(Bandgap)output adjustment 0000：0.825V 0001：0.850V 0010：0.875V 0011：0.900V 0100：0.925V						

		0101：0.950V 0110：0.975V 0111：1.000V 1000：1.025V 1001：1.050V 1010：1.075V 1011：1.100V 1100：1.125V 1101：1.150V 1110：1.175V 1111：1.200V *Note : The internal reference voltage is automatically loaded by the system at power on, and the user is not allowed to modify it.*
3	VHL	LDO working mode selection 1：High Power mode 0：Low Power mode
2~0	VLEVEL	LDO Output voltage setting bit field 000：1.31V 001：1.37V 010：1.43V 011：1.49V 100：1.55V 101：1.61V 110：1.67V 111：1.73V *Note:* *1. The internal clock circuit is powered by the LDO, changing the LDO output voltage will cause a change in the internal clock frequency, in general, the LDO voltage to maintain the default value, it is not recommended to modify.* *2. It is not allowed to set the LDO output voltage less than 1.5V, otherwise it may cause abnormalities.*

10.2 Reset System

There are multiple internal and external reset sources for CA51F3 Series chip as figure 10-2-1 shows.

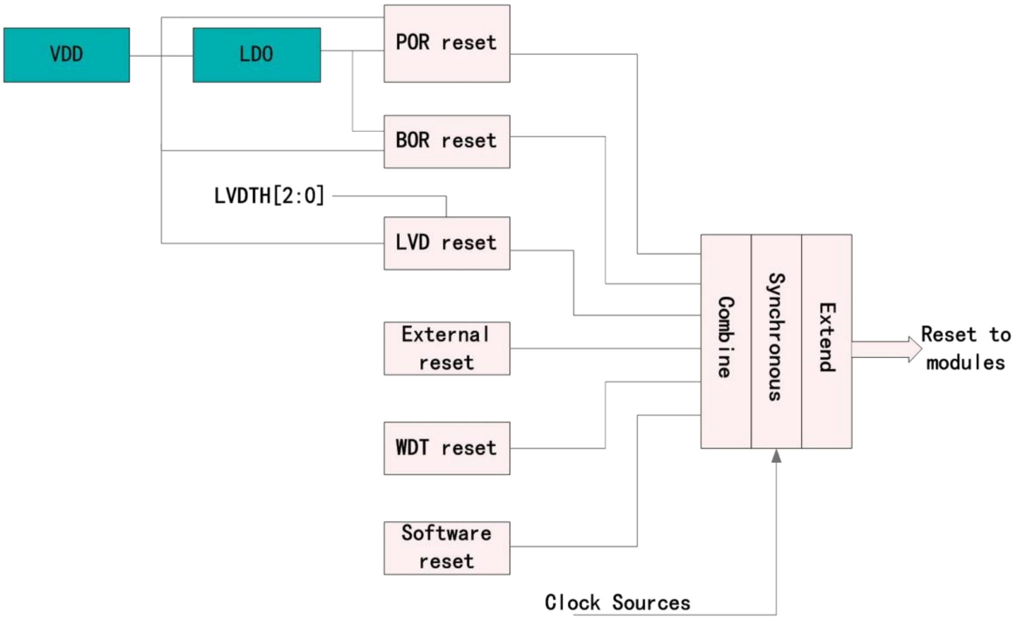


Figure 10-2-1 Reset System Architecture

- **Power On Reset(POR)**

It usually takes some time for the system to reach normal working voltage. The POR is mainly based on VDD and LDO. The POR signal is valid when the voltage is below the detection threshold.

The POR circuit ensures that the chip remains reset during the Power On period hence the chip can starts from certain stable status. The POR signal will also be expanded by the internal counter to makes sure that all the analog modules can enter stable working status after Power On stage.

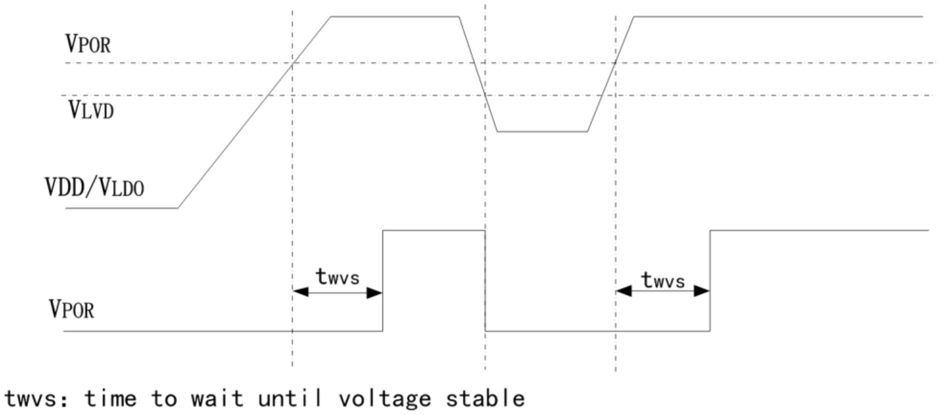


Figure 10-2-2 POR Circuit Example and Power On Stage

- **Brown Out Reset(BOR)**

BOR offers alarm signal for the chip when the voltage drops (eg. Inference or load changes). Once the VDD or internal LDO output voltage is below a certain threshold, it will reset the chip to avoid program error or system abnormality.

Special reminder: the reset pin (P20) is a multi-function pin (P20 is the reset function by default). P20 can only be switched from the reset function to other functions (such as input, output and high resistance). After P20 is switched to other function pins, it can no longer be switched to the reset function, otherwise abnormal reset will be caused.

- **Low Voltage Reset**

The Low Voltage Detection (LVD) can detect VDD in multiple working modes. When VDD voltage is below the threshold set by LVD for 20us it will generates reset signal (on the premise of that LVD is Reset mode).

- **External Reset**

By pulling down the reset pin(RESET), external device can reset the chip as well. RESET can reset the whole in normal working modes, while in STOP mode, the hard reset will awaken the chip first and then reset it. Usually, RESET is pulled up internally and will not influence the internal reset circuit.

- **Watchdog Reset**

The WDT (watchdog timer) is responsible for monitor the how processor do with instructions. With proper configuration, if the WDT is not refreshed in certain time, a reset signal will be generated. WDT is disabled after POR, but users can enable and configure it if necessary.

- **Soft Reset**

The program can soft reset the chip. When 1 is written to SWRST of register PCON, CPU sends out reset signal. Program starts from where BOOT configuration points to after soft reset. PC will point to address 0 after any reset.

11 Power Consumption Management

There are 3 low power consumption modes for CA51F3 Series: IDLE, STOP and Low Speed mode. The system power consumption for IDLE, STOP and Low Speed mode is less than 7uA, 12uA and 20uA respectively.

11.1 IDLE mode

CPU stops working in this mode. All the clocks can be disabled to save power before entering IDLE mode except the main clock. Peripherals can also be enabled/disabled before entering IDLE mode according to user's needs. Those enabled peripherals will operates normally in IDLE mode.

Register IDLST(IDLSTH and IDLSTL) needs to be checked before entering IDLE mode. If all the bits are 0, CPU will enter IDLE mode normally when the mode is set as IDLE. However, if NOT all the bits are 0, CPU will not enter IDLE mode and remains in normal working mode although the mode is set as IDLE. To deal with this situation, users must complete the IDLST corresponding interrupt processing first and then set the mode as IDLE again.

Any reset or interrupt will awake the chip. The clock will resume first and then the chip responds to the interrupt and enters the interrupt service routine after the CPU awakening. After the chip exits interrupt service routine , it will executes the instructions after the instruction which set IDLE to 1. When it exits IDLE mode, IDLE will be cleared automatically.

What must be mentioned is that there should be two “nop” instructions after setting IDLE to 1 to avoid program error.

11.2 STOP mode

The STOP mode is deeper low power consumption mode than IDLE. STOP mode is able to stop all the clocks (include the main clock) and clock generation circuits. If WDT and RTC are enabled, their clock module will still works, hence users may disable them to save power.

Similar to IDLE mode, before entering STOP mode, register STPST(STPSTH and STPSTL) has to check if all the bits are 0. If there are any 1, then they should be processed first to ensure the chip will enter STOP mode successfully.

The STOP mode can be awoken by external interrupt, LVD reset or interrupt, hard reset, RTC interrupt, WDT interrupt or reset, clock monitor interrupt and touch key interrupt. If it is awoken by an interrupt, the chip will resume clock first and respond to the interrupt, and then enters corresponding the interrupt service routine. After the chip exits the interrupt service routine, it will executes the instructions after the setting STOP to 1 instruction. The STOP will be cleared automatically when the chip exits STOP mode.

To arouse the chip better, it is recommended to set the internal clock as system clock before entering STOP mode because it will take longer time waiting for stable status when using external clock.

When the chip enters STOP mode, the last clock edge will disable system clock and then the chip enters STOP mode entirely. What must be mentioned is that there should be three “nop” instructions after setting STOP to 1 avoid program error.

***Important Note*：**

1.When it enters STOP/IDLE mode, setting LDO to low power consumption mode will reduce the power consumption effectively. However, it is a must to set LDO back to high power consumption mode when the chip exits STOP/IDLE mode, otherwise the chip will operates abnormally.

2．If the system clock is selected as IRCL or TFRC, IRCL or TFRC must not be turned off when entering STOP, otherwise an exception may occur when waking up from STOP.

11.3 Low Speed Mode

Since the power consumption is influenced by the its speed, so it will reduce the power consumption effectively if the main clock runs with low speed. The system supports two low speed clock sources： IRCL and XOSCL. The current will be less than 15uA if XOSCL is set as the system clock and will be less than 25uA if IRCL is set as the system clock.

For more information about IRCL and XOSCL you may refer to the Chapter 9 Clock System.

11.4 Related Register Description

Table 11-4-1 Register PCON

87H	7	6	5	4	3	2	1	0
PCON	SMOD	-	SWRST	-	-	TSMODE	STOP	IDLE
R/W	R/W	-	W	-	-	R	W	W
Initial Value	0	-	0	-	-	0	0	0

Bit number	Bit Symbol	Description
7	SMOD	UART0 baud rate ratio control When UART0 is working in mode1,2 and 3, setting SMOD=1 will double baud rate just like standard 8051
6	-	-
5	SWRST	Soft reset control, 1 enables Soft reset mode Setting SWRST=1 will generate soft reset signal, it will be cleared to 0 automatically after the reset
4~3	-	-
2	TSMODE	Test mode flag, 1 indicates that the chip is in test mode

1	STOP	STOP mode control, 1 enables STOP mode When STOP=1 and STPST=0, the chip will enter STOP mode. it will be cleared to 0 automatically after the chip exits STOP mode
0	IDLE	IDLE mode control, 1 enables IDLE mode When IDLE=1and IDLST=0, the chip will enter IDLE mode. it will be cleared to 0 automatically after the chip exits IDLE mode

Table 11-4-2 Register IDLSTL, IDLSTH

FCH	7	6	5	4	3	2	1	0
IDLSTL	IDLST[7:0]							
R/W	R							
Initial Value	0	0	0	0	0	0	0	0
FDH	7	6	5	4	3	2	1	0
IDLSTH		IDLST[14:8]						
R/W		R						
Initial Value	-	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
15	-	-
14	PWMINT/EPIF[7]	Interrupt status of PWM/External Interrupt 9 in IDLE mode
13	RTCINT/EPIF[6]	Interrupt status of RTC/External Interrupt 8 in IDLE mode
12	WDFLG[1]/EPIF[5]	Interrupt status of WDT/External Interrupt 7 in IDLE mode
11	I2CINT/EPIF[4]	Interrupt status of I2C/External Interrupt 6 in IDLE mode
10	CKMINT/EPIF[3]	Interrupt status of Clock Monitor/External Interrupt 5 in IDLE mode
9	LVDINT/EPIF[2]	Interrupt status of LVD/External Interrupt 4 in IDLE mode
8	TKINT/EPIF[1]	Interrupt status of TK/External Interrupt 3 in IDLE mode
7	ADCINT/EPIF[0]	Interrupt status of ADC/External Interrupt 2 in IDLE mode
6	U1INT	Interrupt status of UART1 Interrupt in IDLE mode
5	T2INT	Interrupt status of Timer2 in IDLE mode
4	U0INT	Interrupt status of UART0 in IDLE mode
3	TCON[7]	Interrupt status of Timer1 in IDLE mode
2	PIF[1]	Interrupt status of External Interrupt1 in IDLE mode
1	TCON[5]	Interrupt status of Timer0 in IDLE mode
0	PIF[0]	Interrupt status of External Interrupt0 in IDLE mode

Table 11-4-3 Register STPSTL、STPSTH

FEH	7	6	5	4	3	2	1	0
STPSTL	STPST[7:0]							
R/W	R							
Initial Value	0	0	0	0	0	0	0	0
FFH	7	6	5	4	3	2	1	0
STPSTH	STPST[15:8]							
R/W	R							

Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
15	RTCWKF	Interrupt status of RTC in STOP mode						
14	WDTWKF	Interrupt status of WDT in STOP mode						
13	I2CWKF	Interrupt status of I2C in STOP mode						
12	CKMWKF	Interrupt status of Clock Monitor in STOP mode						
11	LVDWKF	Interrupt status of LVD in STOP mode						
10	TKWKF	Interrupt status of Touch Key in STOP mode						
9	EPWKF[7]	Interrupt status of External Interrupt9 in STOP mode						
8	EPWKF[6]	Interrupt status of External Interrupt8 in STOP mode						
7	EPWKF[5]	Interrupt status of External Interrupt7 in STOP mode						
6	EPWKF[4]	Interrupt status of External Interrupt6 in STOP mode						
5	EPWKF[3]	Interrupt status of External Interrupt5 in STOP mode						
4	EPWKF[2]	Interrupt status of External Interrupt4 in STOP mode						
3	EPWKF[1]	Interrupt status of External Interrupt3 in STOP mode						
2	EPWKF[0]	Interrupt status of External Interrupt2 in STOP mode						
1	PWKF[1]	Interrupt status of External Interrupt1 in STOP mode						
0	PWKF[0]	Interrupt status of External Interrupt0 in STOP mode						

11.5 Low Power Consumption Control Example

- **STOP Mode Example**

The program is like：

```
-----------------------------------------------------------------------------------------------
void Stop(void)
{
        bit IE_EA;
        I2CCON = 0; //disable I2C for it is the default enabled, otherwise the IRCH cannot be disabled
        CKCON = 0; //disable all the clocks
        PWCON &=0xF7; //set LDO in low power consumption mode
        MECON |= (1<<6); //set FLASH in deep sleep mode
        while(STPSTH|STPSTL);// wait until all the interrupts are done
        IE_EA = EA;//Save global interrupt enable bit status
        EA = 0;
        PCON |= 0x02;        // enters STOP mode
        _nop_();
        _nop_();
        _nop_();
        EA = IE_EA;
        PWCON ｜=0x08; // The LDO must return to high power consumption mode after the chip exits STOP mode
    }
    ------------------------------------------------------------------------------------------
```

IDLE Mode Example

The program is like：

```
    ------------------------------------------------------------------------------------------
    void idle(void)
    {
        I2CCON = 0; //disable I2C for it is the default enabled, otherwise the IRCH cannot be disabled
        CKCON = 0; //disable all the clocks except main clock

        Sys_Clk_Set_XOSCL();    //the main clock switches to XOSCL, please refer to the note below
        //Sys_Clk_Set_IRCL();     //the main clock switches to IRCL, please refer to the note below

        PWCON &=0xF7; //set LDO in low power consumption mode
        MECON |= (1<<6); //set FLASH in deep sleep mode
        while(IDLSTH|IDLSTL); //wait until all the interrupts are done
        PCON |= 0x01;        //enters IDLE mode
        _nop_();
        _nop_();
        PWCON ｜=0x08; //The LDO must return to high power consumption mode after the chip exits IDLE mode
    }
```

Note: Since the main clock is still enabled in IDLE mode, if it is high speed clock then the power consumption remains high. Thus, it is very necessary to switch the main clock to low speed clock before entering Low Speed mode.

```
    ------------------------------------------------------------------------------------------
```

Low Speed Mode Example

The program is like：

```
    ------------------------------------------------------------------------------------------
    void LowSpeedMode(void)
    {
        I2CCON = 0; //disables I2C for it is the default enabled, otherwise the IRCH cannot be disabled
        Sys_Clk_Set_XOSCL(); //the main clock switches to XOSCL, please refer to the note below
        //Sys_Clk_Set_IRCL(); //the main clock switches to IRCL, please refer to the note below
        CKCON = 0;  //disable all the clocks except main clock
        PWCON &=0xF7; //set LDO in low power consumption mode
    }
```

Note：Since the LDO must return to high power consumption mode after the chip exits Low Speed mode,similar to STOP/IDLE example

```
    ------------------------------------------------------------------------------------------
```

12 Timer(Timer0,Timer1,Timer2)

12.1 Timer0

12.1.1 Timer0 Introduction

The timer/counter function can be selected by CT0 (TMOD[2]). When CT0=0, it operates as a timer; when CT0=1, it functions as a counter. As a timer, its clock is the system clock with frequency divided by 12. As a counter, its clock is the input clock for T0. Because it takes 2 clock cycles to detect the T0 input signal edge change, so when it operates as a counter, the maximum input baud rate is 1/2 of the internal system clock frequency. There is no limit for T0 input signal's duty cycle. However, in order to identify the 0 and 1 clearly, the signal has to keep for at least one internal system clock cycle. There are for modes for Timer0 which are selected by T0M0 andT0M1 (TMOD[1:0]).

- **Mode0**

Timer0 is a 13 bit timer/counter in this mode. The higher 8 bits are stored in TH0 and the lower 5 bits are stored in TL0[4:0] with TL0[7:5] invalid. When Timer0 overflows, the interrupt flag TF0(TCON[5]) will be set to 1. TF0 will be cleared to 0 automatically after the interrupt response. When GATE0(TCON[3])=0, the timer/counter's is enabled/disabled by TR0 (TCON[4]). When GATE0=1, the timer/counter's is enabled/disabled by INT0. INT0 signal with high level with enable the counting and vice verse.

- **Mode1**

Timer0 is a 16 bit timer/counter in this mode. The function is the same as Mode0.

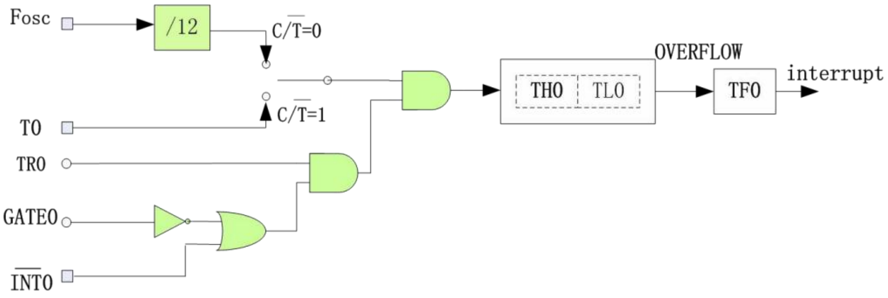


Figure 12-1-1-1 Timer0 Mode0/1

- **Mode2**

Timer0 is a 8 bit automatic reload counter/timer in this mode and only TL0 counts up automatically. When TL0 count overflows, there will be an interrupt flag TF0. The initial value for the count will be reloaded to TL0 from TH0 as well. The other settings are the same as mode0/1.

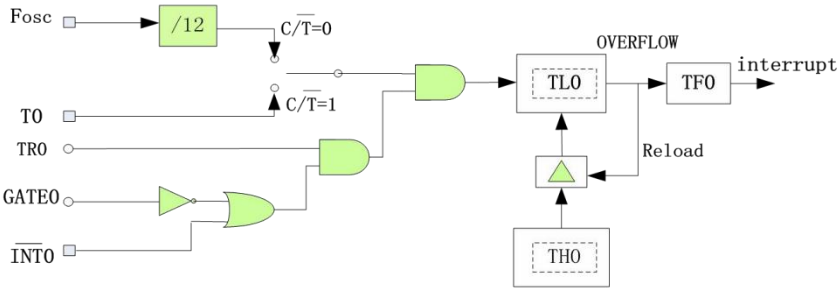


Figure 12-1-1-2 Timer0 Mode2

- **Mode3**

TL0 and TH0 are two independent 8 bit counter/timer in this mode. TL0 can be used as timer or counter while TH0 can only be used as timer. TL0 will be controlled by CT0, GATE0, TR0, TF0 and INT0, and TH0 will only be controlled by TR1 and TF1. The control method is the same as mode0/1. When Timer0 is working in mode3, Timer1 and TH0 both are controlled by TR1. Due to TF1 is used for TH0 already, at the same time, Timer1 can only be used when there is no need for interrupt.(eg, UART baud rate generation)

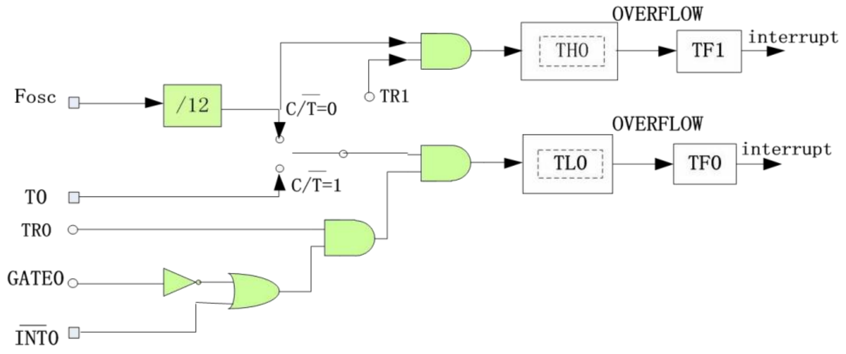


Figure 12-1-1-3 Timer0 Mode3

12.1.2 Timer0 Register Description

Table 12-1-2-1 Register TCON

88H	7	6	5	4	3	2	1	0
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7	TF1	Timer0 TH0 overflow flag in mode3 /Timer1 overflow flag, it is cleared to 0 automatically after the interrupt response
6	TR1	Timer1 enable control, 1 enables it
5	TF0	Timer0 overflow flag, it is cleared to 0 automatically after the interrupt response
4	TR0	Timer0 enable control, 1 enables it
3	IE1	External Interrupt1 enable control, 1 enables it
2	IT1	External Interrupt1 trigger type control 0：External Interrupt1 is triggered when input pin signal is rising edge 1：External Interrupt1 is triggered when input pin signal comes to falling edge
1	IE0	External Interrupt0 enable control, 1 enables it
0	IT0	External Interrupt0 trigger type control 0：External Interrupt0 is triggered when input pin signal is rising edge 1：External Interrupt0 is triggered when input pin signal comes to falling edge

Table 12-1-2-2 Register TMOD

89H	7	6	5	4	3	2	1	0
TMOD	GATE1	CT1	T1M1	T1M0	GATE0	CT0	T0M1	T0M0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7	GATE1	Timer1 gating control. When it equals 1, Timer1 is enabled/disabled by INT1
6	CT1	Timer1 Counter/Timer selection 0：Timer, the clock for it is the system clock with its frequency divided by 12 1：Counter, the clock for it is T1 input clock
5	T1M1	[T1M1,T1M0] for Timer1 mode selection
4	T1M0	00：mode0, TL1 and TH1 make up a 13 bit Timer/Counter 01：mode1, TL1 and TH1 make up a 16 bit Timer/Counter 10：mode2, TL1 is a 8 bit Timer/Counter, TH1 is the automatic reload register 11：mode3, TH1/TL1 locked in this mode, it is the same as TR1=0
3	GATE0	Timer0 gating control. When it equals 1, Timer0 is enabled/disabled by INT0
2	CT0	Timer0 Counter/Timer selection 0：Timer, the clock for it is the system clock with its frequency divided by 12 1：Counter, the clock for it is T0 input clock
1	T0M1	[T0M1,T0M0] Timer0 mode selection
0	T0M0	00：mode0, TL0 and TH0 make up a 13 bit Timer/Counter 01：mode1, TL0 and TH0 make up a 16 bit Timer/Counter 10：mode2, TL0 is a 8 bit Timer/Counter, TH0 is the automatic reload register 11：mode3, TL0 and TH0 are two independent 8 bit Timer/Counter

Table 12-1-2-3 Register TL0

8AH	7	6	5	4	3	2	1	0
TL0	TL0							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~0	TL0	Lower byte of Timer0 count value in mode0/1, count value in mode2/3

Table 12-1-2-4 Register TH0

8CH	7	6	5	4	3	2	1	0
TH0	TH0							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~0	TH0	Higher byte of Timer0's count value in mode0/1, reload value in mode2, count value in mode3

12.2 Timer1

12.2.1 Timer1 Introduction

The timer/counter function can be selected by CT1 (TMOD[6]). When CT1=0 it operates as a timer; when CT1=1, it functions as a counter. As a counter, its clock is the input clock for T1. Because it takes 2 clock cycles to detect the T1 input signal edge change, so when it operates as a counter, the maximum input baud rate is 1/2 of the internal system clock frequency. There is no limit for T1 input signal's duty cycle. However, in order to identify the 0 and 1 clearly, the signal has to keep for at least one internal system clock cycle time. There are for modes for Timer1 which are selected by T1M0 andT1M1 (TMOD[5:4]).

- **Mode0**

Timer1 operates as a 13 bit timer/counter in this mode. TH1 stores the higher 8bits of the 16 bit timer/counter and TL1 stores the lower 5 bits, TL1[7:5] is invalid and should be ignored when read. When timer1 overflows, interrupt flag bit TF1(TCON[7]) will be set to 1. When interrupt is responded, TF1 bit will be automatically cleared to 0. When GATE1(TCON[7]) = 0, the timer/counter is enabled to count by TR1 (TCON[6]) bit, when GATE1 = 1, the timer/counter is enabled by INT1. INT1 signal with high level with enable the counting and vice verse,

- **Mode1**

Timer1 operates as a 16 bit timer/counter in this mode. TH1 stores the higher 8bits of the 16 bit timer/counter and TL1 stores the lower 8 bits. When Timer1 overflows, the interrupt flag TF1(TCON[7]) will be set to 1. TF1 will be cleared automatically after the interrupt response. When GATE1(TCON[7])=0, the Timer/Counter's is enabled/disabled by TR1(TCON[6]). When GATE1=1, the timer/counter's is enabled/disabled by INT1. INT1 signal with high level with enable the counting and vice verse,

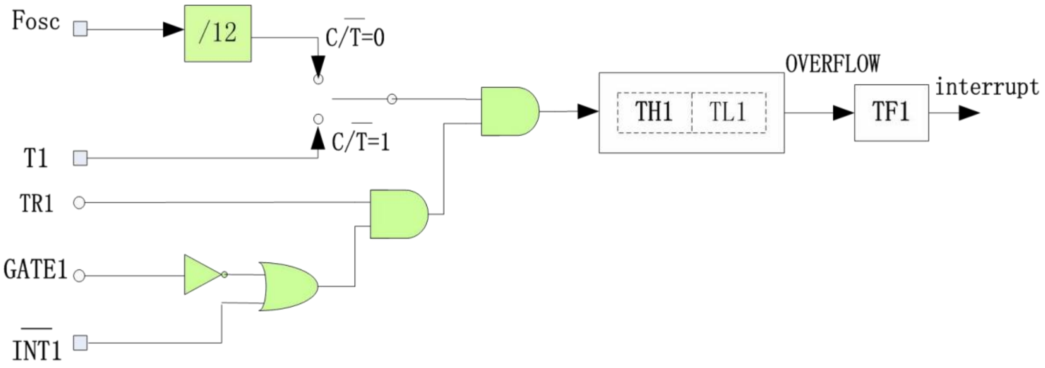


Figure 12-2-1-1 Timer1 Mode1

- **Mode2**

Timer1 is a 8 bit automatic reload counter/timer in this mode and only TL1 counts up automatically. When TL1 count overflows, there will be an interrupt flag TF1. The initial value for the count will be reloaded to TL1 from TH1 as well. The other settings are the same as mode0/1.

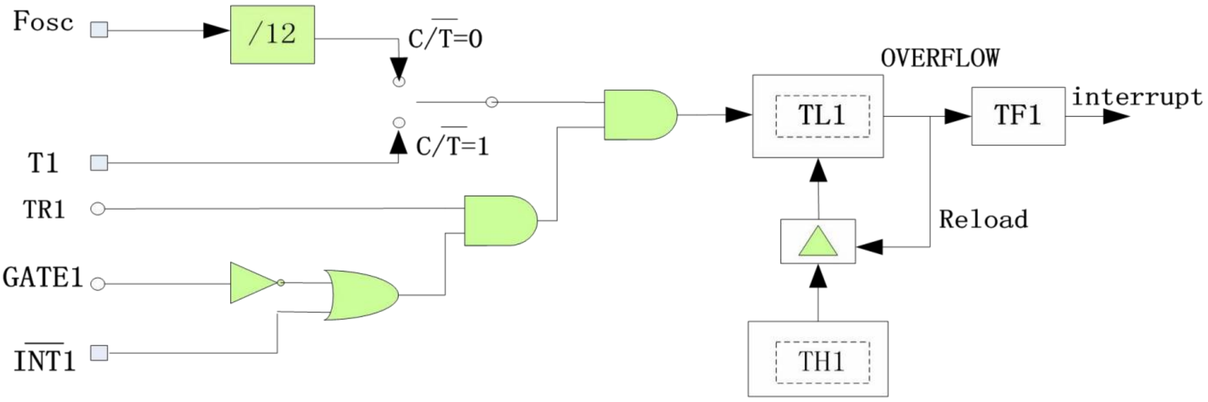


Figure 12-2-1-2 Timer1 Mode2

- **Mode3**

TH1 and TL1 are locked in this mode, which makes it the same as TR1=0.

12.2.2 Timer1 Register Description

For the register TCON and TMOD please refer to Table12-1-2-1 and Table 12-1-2-2.

Table 12-2-2-1 Register TL1

8BH	7	6	5	4	3	2	1	0
TL1	TL1							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
7~0	TL1	Lower byte of Timer1 count value in mode0/1, count value in mode2/3						

Table 12-2-2-2 Register TH1

8DH	7	6	5	4	3	2	1	0
TH1	TH1							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						

7~0	TH1	Higher byte of Timer1's count value in mode0/1, reload value in mode2, count value in mode3

12.3 Timer2

12.3.1 Timer2 Introduction

Timer2 is a 16 bit (TH2 and TL2) timer/counter. T2P0 and T2P1 can be used to select different control modes or clock sources. When T2P=0 or 3, the system clock is directly selected as the clock for Timer2 (Unlike Timer0/1, the frequency of the system clock is not divided by 12).When T2P=0, Timer2 is enabled/disabled by TR2; when T2P=2, it is electrical level gated by T2. When the level of T2 is high, the count is enabled, and when it is low, the count stops. When T2P=1 or 2, the input signal of T2 is selected as the count clock. It counts the falling edges when T2P=1 and rising edges when T2P=2.

The working modes of Timer2 can be selected by setting T2M0 and T2M1. When T2M=0, Timer2 operates as a counter/timer. TH2 and TL2 counts up as a 16 bit counter. Two reload modes can be selected or disabled by setting T2R0 and T2R1 in this mode. T2CH and T2CL stores the reload value in reload mode. If T2R=2, Timer2 will reload the initial count value from T2CH and T2CL to TH2 and TL2 when it overflows. If T2R=3, it reloads when pin T2EX comes to falling edge. The reload flag is set to 1 after the reload. If Timer2 interrupt enables reload interrupt, RF2 can be cleared to 0 by writing 1 to it.

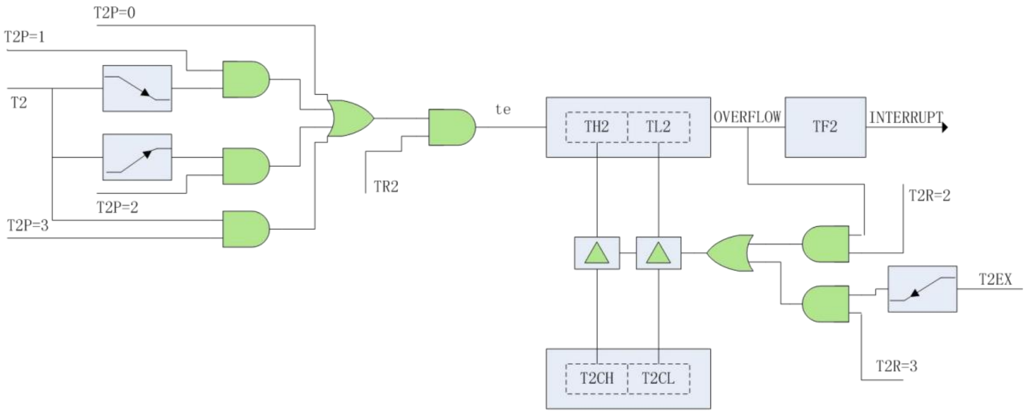


Figure 12-3-1-1 Timer2 Reload Mode

When T2M=1, Timer2 operates in compare mode. When TH2 and TL2 are greater than T2CH and T2CL, the pin T2CP output is high level, otherwise T2CP outputs low level signal.

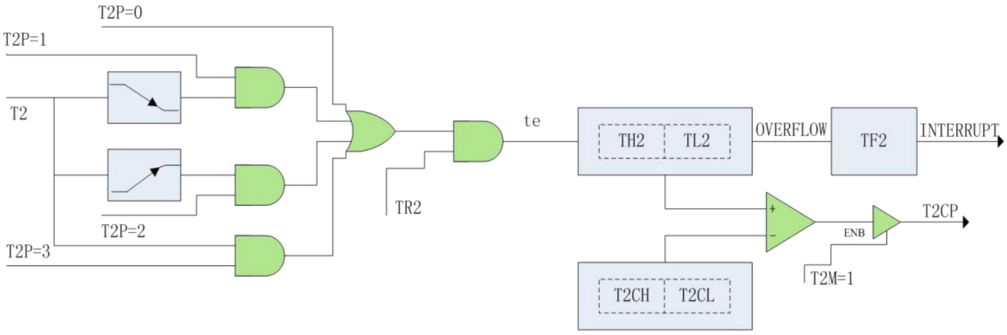


Figure 12-3-1-2 Timer2 Compare Mode Compare Mode

When T2M=2 or 3, Timer2 operates in capture mode. If T2M=2, when T2CP trigger edge comes, Timer2's count TH2 and TL2 will be latched to T2CH and T2CL. The trigger edge can be set by CCFG. The capture flag CF2 will be set to 1 after the capture happened. If Timer2 enables capture interrupt, CF2 can be cleared to 0 by writing 1 to it. When T2M=3, writing register T2CL will trigger the latch, and the value written will not be stored. Capture will not set CF2 to 1 in this mode。

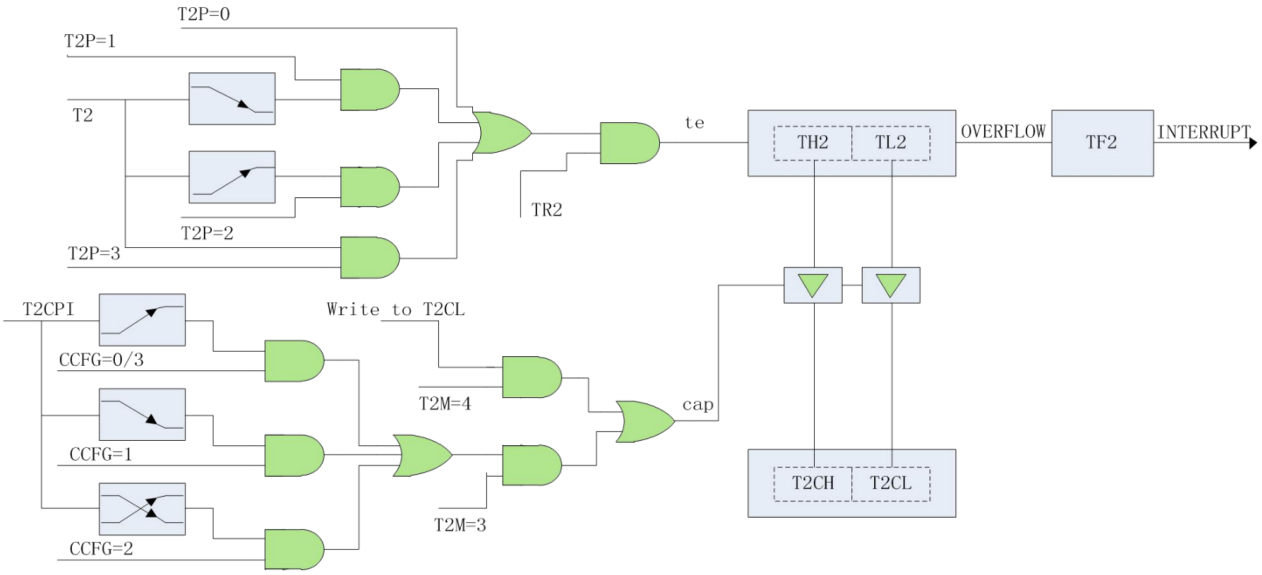


Figure 12-3-1-3 Timer2 Capture Mode

12.3.2 Timer2 Register Description

Table 12-3-2-1 Register T2CON

C8H	7	6	5	4	3	2	1	0
T2CON	-	TR2	T2R1	T2R0	T2IE	UCKS	T2P1	T2P0
R/W	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	-	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
7	-	-						
6	TR2	Timer2 enable control, 1 enables it						
5	T2R1	[T2R1,T2R0]Timer2 reload mode selection 10：mode0 11：mode1 Others：disable reload mode						
4	T2R0							
3	T2IE	Timer2 interrupt enable control,1 enables it						

2	UCKS	UART0 clock selection 0：Timer1 overflow impulse used for UART0 clock 1：Timer2 overflow impulse used for UART0 clock
1	T2P1	[T2P1,T2P0]Timer2 pin T2 function selection 00：Timer2 uses internal system clock for count instead of T2 01：Timer2 counts T2 falling edges 10：Timer2 counts T2 rising edges 11：Timer2 uses internal system clock for count which is gated by T2
0	T2P0	

Table 12-3-2-2 Register T2MOD

C9H	7	6	5	4	3	2	1	0
T2MOD	TF2	CF2	RF2	CCFG1	CCFG0	-	T2M1	T2M0
R/W	R/W	R/W	R/W	R/W	R/W	-	R/W	R/W
Initial Value	0	0	0	0	0	-	0	0

Bit number	Bit Symbol	Description
7	TF2	Timer2 counter overflow interrupt flag, cleared to 0 by writing 1 to it
6	CF2	Capture interrupt flag, cleared to 0 by writing 1 to it
5	RF2	Automatic reload interrupt flag, cleared to 0 by writing 1 to it
4	CCFG1	[CCFG1,CCFG0] capture mode trigger selection, valid when T2M＝3 or T2M＝4 01：falling edge 10：rising or falling edge Others：rising edge
3	CCFG0	
2	-	-
1	T2M1	working mode selection 00：Timer/ counter mode 01：compare mode 10：capture mode 0 11：capture mode 1
0	T2M0	

Table 12-3-2-3 Register T2CL

CAH	7	6	5	4	3	2	1	0
T2CL	T2CL							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~0	T2CL	T2CL is the lower byte of reload value in reload mode T2CL is the lower byte of compare value in compare mode T2CL is the lower byte of capture value in capture mode

Table 12-3-2-4 Register T2CH

CBH	7	6	5	4	3	2	1	0
T2CH	T2CH							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~0	T2CH	T2CL is the higher byte of reload value in reload mode T2CL is the higher byte of compare value in compare mode T2CL is the higher byte of capture value in capture mode

Table 12-3-2-5 Register TL2

CCH	7	6	5	4	3	2	1	0
TL2	TL2							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~0	TL2	Lower byte of the count value in Timer2

Table 12-3-2-6 Register TH2

CDH	7	6	5	4	3	2	1	0
TH2	TH2							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~0	TH2	Higher byte of the count value in Timer2

13 Watchdog Timer(WDT)

13.1 Watchdog Timer(WDT) Function Introduction

The watchdog timer is a 27 bit backward counter with alternate clock sources. When the clock frequency is 3.6864MHz, the count time can be 0.56ms - 36.4s with 16 bit adjustment precision. The watchdog is mainly used for monitoring the system so that CPU will not break down due to external interference. If the software can not refresh WDT before it overflows, the watchdog will generate internal reset or interrupt. Writing A5H to register WDFLG will refresh the watchdog and reading WDFLG will get the status of the watchdog. If the watchdog is enabled in STOP mode, then the clock selected by the watchdog will works normally. In addition, if the interrupt function is also enabled for watchdog, it will awaken CPU in STOP mode.

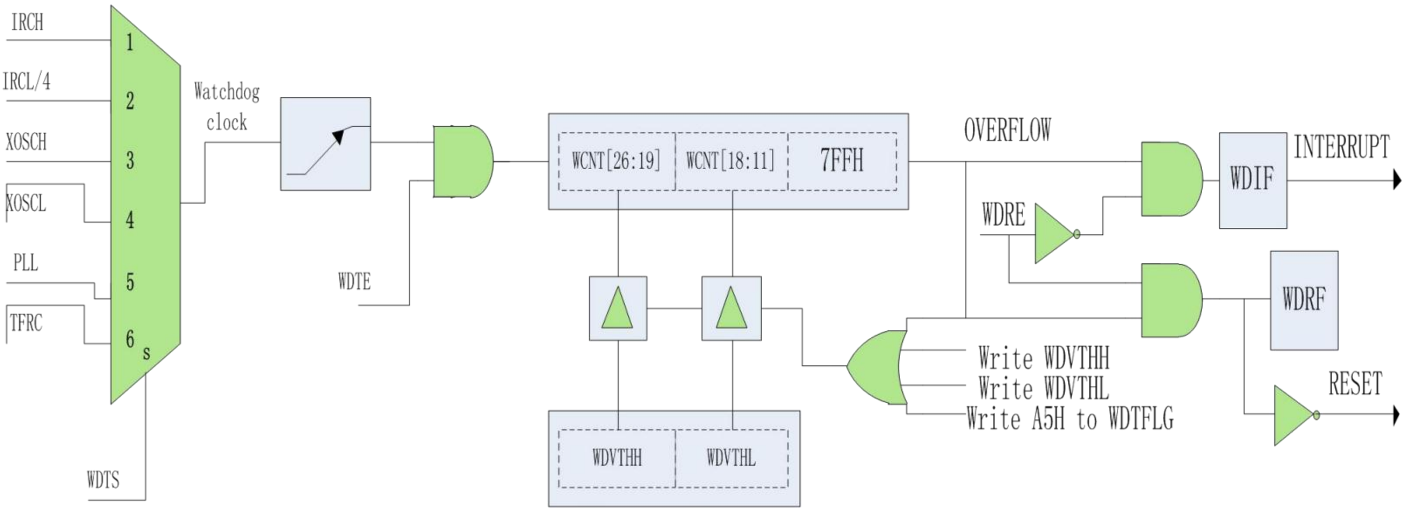


Figure 13-1-1 Watchdog Module Architecture

13.2 Watchdog Timer(WDT) Register Description

Table 13-2-1 Register WDCON

AAH	7	6	5	4	3	2	1	0
WDCON	WDTS[2:0]			-	-	-	-	WDRE
R/W	R/W	R/W	R/W	-	-	-	-	R/W
Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
7~5	WDTS	WDT clock selection 001： IRCH						

		010：IRCL with frequency divided by 4 011：ERC 100：XOSCL 101：PLL 110：TFRC Others：WDT disabled
4~1	-	
0	WDRE	WDT function selection 0：interrupt happens when WDT overflows 1：reset happens when WDT overflows

Table 13-2-2 Register WDFLG

ABH	7	6	5	4	3	2	1	0
WDFLG							WDIF	WDRF
R/W	-						R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~2	-	-
1	WDIF	WDT interrupt flag, writing A5H to the register will clear it
0	WDRF	WDT reset flag, writing A5H to the register will clear it

Table 13-2-3 Register WDVTHL、WDVTHH

ACH	7	6	5	4	3	2	1	0
WDVTHL	WDVTH[7:0]							
R/W	R/W							
Initial Value	1	1	1	1	1	1	1	1
ADH	7	6	5	4	3	2	1	0
WDVTHH	WDVTH[15:8]							
R/W	R/W							
Initial Value	1	1	1	1	1	1	1	1

Bit number	Bit Symbol	Description
15~0	WDVTH	WDT threshold setting, the equation is as follows: WDT trigger time ＝ (WDVTH * 800H+7FFH) * clock cycle If WDT clock is 3.6864M, it covers 0.56ms ~ 36s

13.3 Watchdog Timer Control Example

- **Watchdog interrupt mode example**

For instance, IRCH is set for the watchdog clock and the frequency for it is 3.6864MHz. The watchdog works in interrupt mode and the overflow time is one second, the program is like:

```
-----------------------------------------------------------------------------------------
#define WDTS_IRCH            (1<<5)
#define WDRE_reset           (1<<0)
#define WDRE_int        (0<<0)
void WDT_init(void)
{
        WDCON = WDTS_IRCH | WDRE_int;        //set the clock as IRCH and watchdog in interrupt
        mode WDVTHH = 0x07;                  //set one second as the time for watchdog
        WDVTHL = 0x08;
        INT7EN = 1;                         //enable WDT interrupt
        WDFLG = 0xA5;                        //refresh the watchdog
        EA = 1;                              //enable total interrupt
}
void WDT_ISR (void) interrupt 12
{
        if(WDFLG & 0x02)
        {
                // watch dog interrupt service routine
                WDFLG = 0xA5;//refresh the watchdog
        }
}
--------------------------------------------------------------------------------------
```

- **Example for watchdog reset mode**

For instance, IRCH is set for the watchdog clock and the frequency for it is 3.6864MHz. The watchdog works in reset mode and the overflow time is one second, the program is like：

```
-----------------------------------------------------------------------------------------
#define WDTS_IRCH            (1<<5)
#define WDRE_reset           (1<<0)
#define WDRE_int        (0<<0)
void WDT_init(void)
{
        WDCON = WDTS_IRCH | WDRE_reset;          //set the clock as IRCH and watchdog in reset
        mode WDVTHH = 0x07;                  //set one second as the time for watchdog
        WDVTHL = 0x08;
        WDFLG = 0xA5;                        //refresh the watchdog
}
-----------------------------------------------------------------------------------------
```

14 Real Time Clock(RTC)

14.1 RTC Function Introduction

The internal RTC is a real time clock module including millisecond, second, minute, hour, day and week registers and with alarm clock function embedded. The main clock source for it is the 32.768KHz external crystal oscillator. If the RTC time matches the time set by users, there will be an interrupt which makes it very convenient for product with (alarm) clock. In addition, RTC can set millisecond/half second interrupt with the interrupt time configurable for millisecond interrupt. Without 32.768KHz external crystal oscillator, the IRCL with frequency divided by 4 can also be the clock source for RTC when there is no need for high accuracy. In STOP/IDLE mode, RTC can be enabled and operates as the trigger source to waken the chip. Figure 14-1-1 shows the RTC architecture.

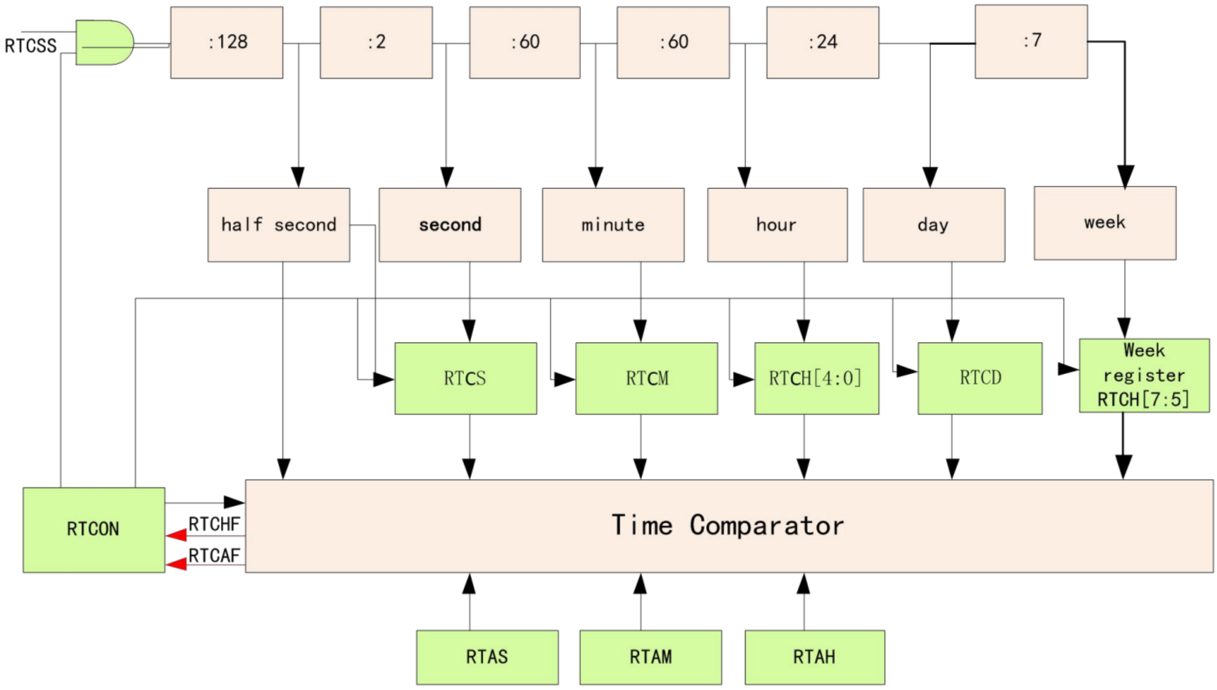


Figure 14-1-1 RTC Architecture

- **Enable/disable RTC**

RTC can be enabled/disabled by RTCE (RTCON[7]). RTC starts counting after RTCE=1 and if RTCE=0, all the registers of RTC module will be latched. It is a must to wait 300us after the RTC is enabled and then write the time register, otherwise it is invalid. Since RTC clock source is the 32.768KHz external crystal oscillator, it must wait until the oscillator works normally and then the RTC can be enabled.

- **RTC Register R/W**

RTCWE(RTCON[1]) enables/disables RTC registers (RTCSS, RTCS, RTCM, RTCH, RTCDL, RTCDH) writing.

When RTCWE = 1, users have to 50us to overwrite RTC registers. RTCWE waits 50us after the writing and then changes to 0. Any illegal time which is beyond the second/minute/hour range will be seen as the maximum value of the register. The microsecond register RTCSS will be cleared when second/minute/hour/week is written. RTC registers can be read directly.

- **RTC Alarm Clock**

When RTC time matches alarm clock time, there will be an interrupt generated and the flag is RTCAF。Users can set the alarm clock time using register RTAS, RTAM and RTAH instead of setting RTCWE. Any illegal time which is beyond the second/minute/hour range will be seen as the maximum value of the register. Users may set corresponding compare enable control(HCE、MCE、SCE)to compare the values in register RTAS, RTAM and RTAH. If the enable control is set to 0, the corresponding time compare will be ignored (For instance, HCE=1, MCE=0, SCE=1, then only the hour and second will be compared, with minute default matched). In the end of the day, the alarm clock can occurs only once with all the compare enabled or several times periodically(every minute/hour/day) by select specific compares enabled.

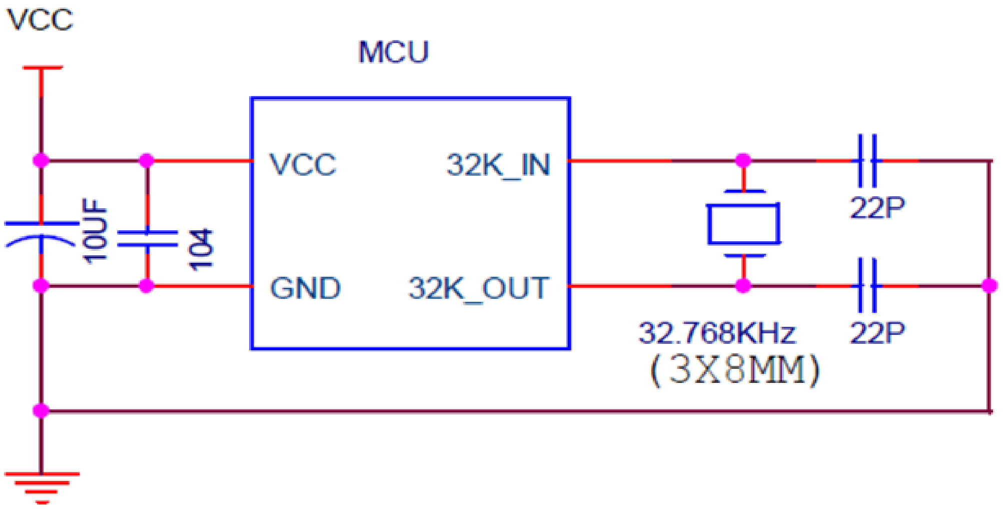


图 14-1-2 XOSCL 典型电路图

Important reminder:

1. Hardware design of the crystal load capacitor ground must be connected to the chip ground, crystal compensation capacitor as close as possible to the chip GND pins. 32.768KHz quartz crystal requires the use of 3mmx8mm diameter crystal specifications.
2. The above circuit and component parameters are for reference only, the use of different manufacturers of crystal oscillators in the use of the circuit parameters may need to be modified.

14.2 RTC Register Description

Table 14-2-1 Register RTCON

F1H	7	6	5	4	3	2	1	0
RTCON	RTCE	MSE	HSE	SCE	MCE	HCE	RTCWE	-
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	-
Initial Value	0	0	0	0	0	0	0	-

Bit number	Bit Symbol	Description
7	RTCE	RTC clock enable control, 1 enables it
6	MSE	The millisecond interrupt enable control, 1 enables it
5	HSE	The half second interrupt enable control, 1 enables it
4	SCE	The alarm clock second compare enable control, 1 enables it
3	MCE	The alarm clock minute compare enable control, 1 enables it
2	HCE	The alarm clock hour compare enable control, 1 enables it
1	RTCWE	Clock write enable control, 1 enables it
0	-	-

Table 14-2-2 Register RTCSS

E9H	7	6	5	4	3	2	1	0
RTCSS	RTCSS[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	-

Bit number	Bit Symbol	Description
7~0	RTCE	RTC microsecond counter, 1 is added to it every 1/256 second

Table 14-2-3 Register RTCS

F2H	7	6	5	4	3	2	1	0
RTCS	-	-	RTCS[5:0]					
R/W	-	-	R/W					
Initial Value	-	-	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~6	-	-
5~0	RTCS	Second counter ranges from 0 to 59, 1 is added to it every 1 second

Table 14-2-4 Register RTCM

F3H	7	6	5	4	3	2	1	0
RTCM	-	-	RTCM[5:0]					
R/W	-	-	R/W					
Initial Value	-	-	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
7~6	-	-						
5~0	RTCM	Minute counter ranges from 0 to 59, 1 is added to it every 1 minute						

Table 14-2-5 Register RTCH

F4H	7	6	5	4	3	2	1	0
RTCH	RTCW[2:0]			RTCH[4:0]				
R/W	R/W			R/W				
Initial Value	x	x	x	0	0	0	0	0
Bit number	Bit Symbol	Description						
7~5	RTCW	Week counter ranges from 1 to 7 which stands for Monday to Sunday, when it is set to 0, the week count function is disabled *Note: RTCW is an indeterminate value after power-on reset, and the initial value must be written before it can be applied.*						
4~0	RTCH	Hour counter ranges from 0 to 23, 1 is added to it every 1 hour						

Table 14-2-6 Register RTCDL、RTCDH

F5H	7	6	5	4	3	2	1	0
RTCDL	RTCD[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
F6H	7	6	5	4	3	2	1	0
RTCDH	RTCD[15:8]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
15~0	RTCD	Day counter, 1 is added to it every day						

Table 14-2-7 Register RTAS

EAH	7	6	5	4	3	2	1	0
RTAS	-	-	RTAS[5:0]					
R/W	-	-	R/W					
Initial Value	-	-	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
7~6	-	-						
5~0	RTAS	The alarm clock second setting (ranges from 0 to 59)						

Table 14-2-8 Register RTAM

EBH	7	6	5	4	3	2	1	0
RTAM	-	-	RTAM[5:0]					
R/W	-	-	R/W					
Initial Value	-	-	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
7~6	-	-						
5~0	RTAM	The alarm clock minute setting (ranges from 0 to 59)						

Table 14-2-9 Register RTAH

ECH	7	6	5	4	3	2	1	0
RTAH	-	-	-	RTAH[4:0]				
R/W	-	-	-	R/W				
Initial Value	-	-	-	0	0	0	0	0
Bit number	Bit Symbol	Description						
7~5	-	-						
4~0	RTAH	The alarm clock hour setting (ranges from 0 to 23)						

Table 14-2-10 Register RTMSS

EDH	7	6	5	4	3	2	1	0
RTMSS	RTMSS[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						

7~0	RTMSS	RTC millisecond interrupt threshold register, millisecond interrupt time =(RTMSS+1)× 128 × RTC clock cycle. If the RTC clock frequency is 32.768KHz, then the time unit is 128×(1/32.768)=3.90625ms

Table 14-2-11 Register RTCIF

EEH	7	6	5	4	3	2	1	0
RTCIF	-	-	-	-	-	RTCMF	RTCHF	RTCAF
R/W	-	-	-	-	-	R	R	R
Initial Value	-	-	-	-	-	0	0	0

Bit number	Bit Symbol	Description
7~3	-	-
2	RTCMF	RTC millisecond interrupt flag, cleared to 0 when 1 is written to it
1	RTCHF	RTC half second interrupt flag, cleared to 0 when 1 is written to it
0	RTCAF	RTC alarm clock interrupt flag, cleared to 0 when 1 is written to it

14.3 RTC Control Example

- **Write time to RTC**

To write hour, minute and second, the program is like：

```
-----------------------------------------------------------------------------------
#define RTCE        (1<<7)
void RTC_WriteHour(unsigned char hour)    //hour=0~23
{
    RTCON |= RTCWE; // enable time writing
    RTCH = hour;    //write the hour
    Delay_50us(1);  //must delay 50 microseconds
    RTCON &= ~RTCWE;//disables time writing
}
void RTC_WriteMinute(unsigned char minute)//minute=0~59
{
        RTCON |= RTCWE;//enable time writing
        RTCM = minute;//write the minute
        Delay_50us(1);//must delay 50 microseconds
        RTCON &= ~RTCWE;//disables time writing
}
void RTC_WriteSecond(unsigned char second)//second=0~59
{
        RTCON |= RTCWE;//enable time writing
        RTCS = second;//write the second
        Delay_50us(1);//must delay 50 microseconds
        RTCON &= ~RTCWE;//disables time writing
}
--------------------------------------------------------------------------------------
```

- **Set the alarm clock time**

For instance, set the alarm clock time 11:30:0 with hour, minute and second compare enabled, the program is like：

```
--------------------------------------------------------------------------------------
#define SCE(N)          (N<<4)    //N=0~1
#define MCE(N)          (N<<3)    //N=0~1
#define HCE(N)          (N<<2)    //N=0~1
#define RTC_AF          (1<<0)
Void RTM_init(void)
{
        RTAH = 11;    //set the hour for the alarm clock
        RTAM = 30;    //set the minute for the alarm clock
        RTAS = 0;     //set the second for the alarm clock
        RTCON |= SCE(1)|MCE(1)|HCE(1); //enables hour, minute and second compare
}
void RTC_ISR (void) interrupt 13
{
        if(RTCIF & RTC_AF)                    //alarm clock interrupt
        {
                RTCIF = RTC_AF;
               //alarm clock interrupt service routine

        }
......
}
----------------------------------------------------------------------------------------
```

- **RTC initialization**

RTC initial the program is like：

```
---------------------------------------------------------------------------------------
#define XLCKE              (1<<3)
#define XLSTA              (1<<2)
void RTC_init(void)
{
        CKCON |= XLCKE;                 //enable XOSCL clock
        while(!(CKCON & XLSTA));//wait until XOSCL clock stable
        RTCON = RTCE(1) | MSE(1) | HSE(1); //enable RTC, millisecond interrupt and half second
        interrupt RTC_WriteHour(10);            //write the hour
        RTC_WriteMinute(30);//write         the
        minute   RTC_WriteSecond(0);   //write
        the second RTM_init(); //set the alarm
        clock
        RTMSS = 0;      //set the time for millisecond
        interrupt INT8EN = 1; //enable RTC interrupt
}
void RTC_ISR (void) interrupt 13
{
        if(RTCIF & RTC_MF)                    //millisecond interrupt
        {
                RTCIF = RTC_MF;
                //millisecond interrupt service routine

        }
         if(RTCIF & RTC_HF) //half second interrupt
        {
              RTCIF = RTC_HF;
```

```
            //half second interrupt service routine
        }
        if(RTCIF & RTC_AF)                        //the alarm clock interrupt
        {
            RTCIF = RTC_AF;
            //alarm clock interrupt service routine

        }
}
```

----

15 General Purpose Input/Output(GPIO) and Alternate Functions

15.1 Function Introduction

General Purpose Input/Output is used for data transmission between the chip and external environment, There are at most 26 I/O pins for CA51F3 Series chip package and all of the pins are alternate function pins. They can not only be independently programmed as input/output port, be also be configured as pins for other functions. For each pin, there are function register PnxF and PnxC (corresponding to pin Pnx n=0~7, stands for P0~P7, x=0~7, stands for Pn.0~Pn.7). Users can configure the main function and other options by setting register PnxF and PnxC.

Each I/O can be enabled by PnxPUP/PnxPDP(PnxF[7]/PnxF[6]) to enable pull-up/down resistors, strong/weak pull-up/pull down resistors are selected by PU_SEL/PD_SEL(PnxC[5]/PnxC[4]), when PU_SEL/PD_SEL is 1, it is selected as strong pull-up/pull down, otherwise it is weak pull-up/down, the default is strong pull-up/pull down.

When I/O is set to output mode, I/O is open-drain output mode when PnxOPR(PnxF[5]) is 1.
When I/O is push-pull output, DRV(PnxC[3:2]) can set IO push-pull output drive strength, SR(PnxC[1:0]) can set IO output flip slope, when I/O output level flip, due to inductance effect, it will generate overshoot signal in I/O port, this overshoot signal may cause some impact on chip system, reduce I/O output strength and Reduce the I/O output strength and I/O speed can effectively reduce the amplitude of the overshoot signal, in the application, these two parameters can be flexibly configured.

When I/O is input mode, SMIT mode or Inverter mode can be selected by SMIT_EN (PnxC[6]) bit. When Inverter mode is selected, the trigger threshold value of high and low level is 1/2 VDD, and the default is Smit mode.

In addition, P00~P04 can be multiplexed as LED COM pins, which can be set to high potting level mode by SINK_EN(P0xC[7]). In high potting current mode, the potting current is greater than 60mA (refer to the electrical characteristics chapter for test conditions).

Main features of GPIO:

- High impedance mode configurable
- The strong pull-up, weak pull-up strong pull-down or weak pull-down resistors can be set independently for the I/O structure
- Open-drain or push-pull output can be selected for the output mode
- The data output latch can be read/written/revised
- Supports 1.8~5.5V voltage
- When set to push-pull output, the IO drive intensity can be set independently
- When set to push-pull output, the IO output speed can be set independently
- LED COM can be set to a high sink current of up to 60mA (test conditions can be found in the electrical characteristics section)

Important reminder:

All GPIO pin input voltages must not be higher than the VDD pin voltage, otherwise it may cause the chip to work abnormally.

The Figure 15-1-1 shows GPIO Push-pull Mode Structure.

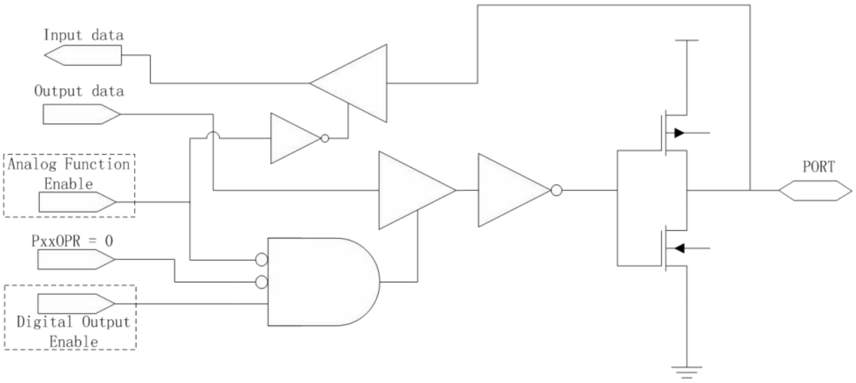


Figure 15-1-1 I/O Push-pull Mode Structure

The Figure 15-1-2 shows GPIO Open-drain Mode Structure.

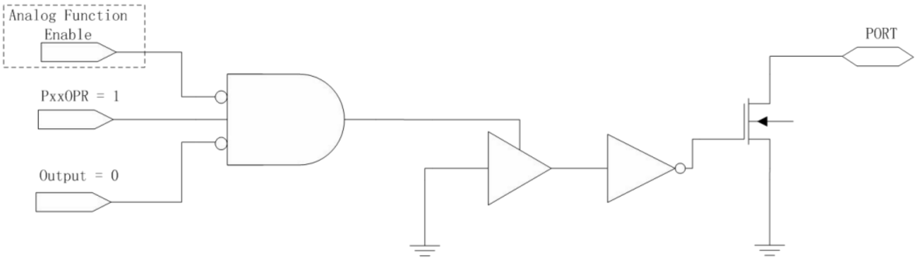


Figure 15-1-2 I/O Open-drain Mode Structure

The Figure 15-1-3 shows GPIO Pull-down Mode Structure.

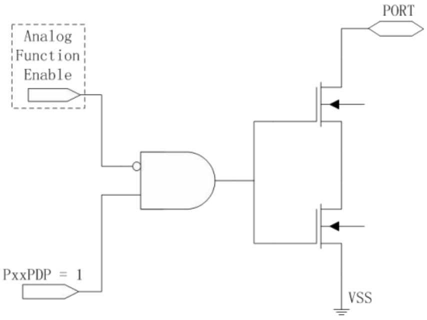


Figure 15-1-3 I/O Pull-down Mode Structure

The Figure 15-1-4 shows GPIO Pull-up Mode Structure

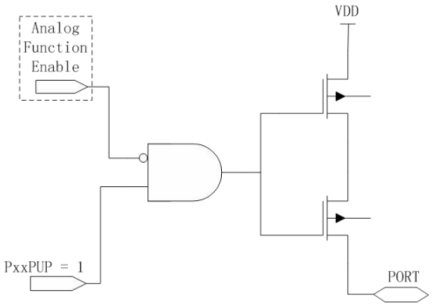


Figure 15-1-4 I/O Pull-up Mode Structure

15.2 Pin Register Description

Table 15-2-1 Register P0

80H	7	6	5	4	3	2	1	0
P0	P07	P06	P05	P04	P03	P02	P01	P00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~0	P0x	Data register for pin P0x，valid when the pin function is set to GPIO 0：P0x is low level when the pin is set to input; when the pin set to output,P0x outputs low level signal 1：P0x is high level when the pin is set to input; when the pin set to output,P0x outputs high level signal

Table 15-2-2 Register P1

90H	7	6	5	4	3	2	1	0
P1	P17	P16	P15	P14	P13	P12	P11	P10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~0	P1x	Data register for pin P1x, valid when the pin function is set to GPIO 0:P1x is low level when the pin is set to input; when the pin set to output,P1x outputs low level signal 1:P1x is high level when the pin is set to input; when the pin set to output,P1x outputs high level signal

Table 15-2-3 Register P2

A0H	7	6	5	4	3	2	1	0
P2	-	-	-	-	-	-	P21	P20
R/W	-	-	-	-	-	-	R/W	R/W
Initial Value	-	-	-	-	-	-	0	0

Bit number	Bit Symbol	Description
7~0	P2x	Data register for pin P2x， valid when the pin function is set to GPIO 0：P2x is low level when the pin is set to input; when the pin set to output,P2x outputs low level signal 1：P2x is high level when the pin is set to input; when the pin set to output,P2x outputs high level signal

Table 15-2-4 Register P3

B0H	7	6	5	4	3	2	1	0
P3	P37	P36	P35	P34	P33	P32	P31	P30
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~0	P3x	Data register for pin P3x, valid when the pin function is set to GPIO 0：P3x is low level when the pin is set to input; when the pin set to output,P3x outputs low level signal 1：P3x is high level when the pin is set to input; when the pin set to output,P3x outputs high level signal

Table 15-2-5 Pin Function Control Register

8000H	7	6	5	4	3	2	1	0
P00F	P00PUP	P00PDP	P00OPR	-	-	P00S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8001H	7	6	5	4	3	2	1	0
P01F	P01PUP	P01PDP	P01OPR	-	-	P01S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8002H	7	6	5	4	3	2	1	0
P02F	P02PUP	P02PDP	P02OPR	-	-	P02S		
R/W	R/W	R/W	R/W	-	-	R/W		

Initial Value	0	0	0	-	-	0	0	0
8003H	7	6	5	4	3	2	1	0
P03F	P03PUP	P03PDP	P03OPR	-	-	P03S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8004H	7	6	5	4	3	2	1	0
P04F	P04PUP	P04PDP	P04OPR	-	-	P04S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8005H	7	6	5	4	3	2	1	0
P05F	P05PUP	P05PDP	P05OPR	-	-	P05S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8006H	7	6	5	4	3	2	1	0
P06F	P06PUP	P06PDP	P06OPR	-	-	P06S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8007H	7	6	5	4	3	2	1	0
P07F	P07PUP	P07PDP	P07OPR	-	-	P07S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8008H	7	6	5	4	3	2	1	0
P10F	P10PUP	P10PDP	P10OPR	-	-	P10S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8009H	7	6	5	4	3	2	1	0
P11F	P11PUP	P11PDP	P11OPR	-	-	P11S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
800AH	7	6	5	4	3	2	1	0
P12F	P12PUP	P12PDP	P12OPR	-	-	P12S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
800BH	7	6	5	4	3	2	1	0
P13F	P13PUP	P13PDP	P13OPR	-	-	P13S		

R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
800CH	7	6	5	4	3	2	1	0
P14F	P14PUP	P14PDP	P14OPR	-	-	P14S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
800DH	7	6	5	4	3	2	1	0
P15F	P15PUP	P15PDP	P15OPR	-	-	P15S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
800EH	7	6	5	4	3	2	1	0
P16F	P16PUP	P16PDP	P16OPR	-	-	P16S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
800FH	7	6	5	4	3	2	1	0
P17F	P17PUP	P17PDP	P17OPR	-	-	P17S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8010H	7	6	5	4	3	2	1	0
P20F	P20PUP	P20PDP	P20OPR	-	-	P20S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8011H	7	6	5	4	3	2	1	0
P21F	P21PUP	P21PDP	P21OPR	-	-	P21S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8018H	7	6	5	4	3	2	1	0
P30F	P30PUP	P30PDP	P30OPR	-	-	P30S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8019H	7	6	5	4	3	2	1	0
P31F	P31PUP	P31PDP	P31OPR	-	-	P31S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
801AH	7	6	5	4	3	2	1	0

P32F	P32PUP	P32PDP	P32OPR	-	-	P32S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0

801BH	7	6	5	4	3	2	1	0
P33F	P33PUP	P33PDP	P33OPR	-	-	P33S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0

801CH	7	6	5	4	3	2	1	0
P34F	P34PUP	P34PDP	P34OPR	-	-	P34S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0

8017H	7	6	5	4	3	2	1	0
P27F	P27PUP	P27PDP	P27OPR	-	-	P27S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0

8018H	7	6	5	4	3	2	1	0
P30F	P30PUP	P30PDP	P30OPR	-	-	P30S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0

8019H	7	6	5	4	3	2	1	0
P31F	P31PUP	P31PDP	P31OPR	-	-	P31S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0

801AH	7	6	5	4	3	2	1	0
P32F	P32PUP	P32PDP	P32OPR	-	-	P32S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0

801BH	7	6	5	4	3	2	1	0
P33F	P33PUP	P33PDP	P33OPR	-	-	P33S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0

801CH	7	6	5	4	3	2	1	0
P34F	P34PUP	P34PDP	P34OPR	-	-	P34S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0

801DH	7	6	5	4	3	2	1	0
P35F	P35PUP	P35PDP	P35OPR	-	-	P35S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
801EH	7	6	5	4	3	2	1	0
P36F	P36PUP	P36PDP	P36OPR	-	-	P36S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
801FH	7	6	5	4	3	2	1	0
P37F	P37PUP	P37PDP	P37OPR	-	-	P37S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0

Bit number	Bit Symbol	Description
7	PnxPUP	Pull-up resistor enable control 0: disable pull-up resistor 1: enable pull-up resistor
6	PnxPDP	Pull-down resistor enable control 0: disable pull-down resistor 1: enable pull-down resistor
5	PnxOPR	Open-drain enable control, only valid when the pin is set to be digital output 0: disable open-drain 1: enable open-drain

Note: Pnx → n=0~7, stands for P0~P3

x=0~7, stands for Pn.0~Pn.7

Table 15-2-6 Pin Control Register PxnC

8120H	7	6	5	4	3	2	1	0
P00C	SINK_EN	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	R/W	R/W	R/W	R/W	R/W		R/W	
Initial Value	0	1	1	1	1	1	1	1
8121H	7	6	5	4	3	2	1	0
P01C	SINK_EN	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	R/W	R/W	R/W	R/W	R/W		R/W	
Initial Value	0	1	1	1	1	1	1	1
8122H	7	6	5	4	3	2	1	0
P02C	SINK_EN	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	R/W	R/W	R/W	R/W	R/W		R/W	
Initial Value	0	1	1	1	1	1	1	1
8123H	7	6	5	4	3	2	1	0
P03C	SINK_EN	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	R/W	R/W	R/W	R/W	R/W		R/W	
Initial Value	0	1	1	1	1	1	1	1
8124H	7	6	5	4	3	2	1	0
P04C	SINK_EN	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	

R/W	R/W	R/W	R/W	R/W	R/W		R/W	
Initial Value	0	1	1	1	1	1	1	1
8125H	7	6	5	4	3	2	1	0
P05C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8126H	7	6	5	4	3	2	1	0
P06C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8127H	7	6	5	4	3	2	1	0
P07C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8128H	7	6	5	4	3	2	1	0
P10C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8129H	7	6	5	4	3	2	1	0
P11C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
812AH	7	6	5	4	3	2	1	0
P12C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
812BH	7	6	5	4	3	2	1	0
P13C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
812CH	7	6	5	4	3	2	1	0
P14C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
812DH	7	6	5	4	3	2	1	0
P15C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
812EH	7	6	5	4	3	2	1	0
P16C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
812FH	7	6	5	4	3	2	1	0
P17C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8130H	7	6	5	4	3	2	1	0
P20C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8131H	7	6	5	4	3	2	1	0
P21C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8138H	7	6	5	4	3	2	1	0
P30C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1

8139H	7	6	5	4	3	2	1	0
P31C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
813AH	7	6	5	4	3	2	1	0
P32C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
813BH	7	6	5	4	3	2	1	0
P33C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
813CH	7	6	5	4	3	2	1	0
P34C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
813DH	7	6	5	4	3	2	1	0
P35C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
813EH	7	6	5	4	3	2	1	0
P36C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
813FH	7	6	5	4	3	2	1	0
P37C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1

Note：

1. *PxnC stands for P00C~P37C, where x=0,1,2,3 stands for P0~P3, n=0,1,2,... ,7 (n=0,1 when x=2).*
2. *The SINK_EN bit is present only on the control register of the IO with LED COM function.*

位编号	位符号	说明
7	SINK_EN	High sink current enable bit, 1 enables it *Note: IO must be set to push-pull output mode*
6	SMIT_EN	Mode selection bit for IO input function 0：Inverter mode 1：SMIT mode
5	PU_SEL	Pull-up resistor selection bit 0 Weak pull-up (pull-up resistor of 45K) 1: Strong pull-up (pull-up resistor of 10K)
4	PD_SEL	Pull-down resistor selection bit 0 Weak pull-down (pull-down resistor of 45K) 1：Strong pull-down (pull-down resistor of 15K)
3~2	DRV	Output strength selection bit, range: 0~3, the larger the value, the stronger the drive capability
1~0	SR	Output slope control bit, range: 0~3, the larger the value, the higher the IO flip slope (the faster the speed)

Table 15-2-7 Pin Alternate Function Mapping

value name	0	1	2	3	4	5	6	7
P00S	high impedance	digital input	digital output	COM0	TK19/LED[19]	ERC	high impedance	high impedance
P01S	high impedance	digital input	digital output	COM1	TK18/LED[18]	high impedance	high impedance	high impedance
P02S	high impedance	digital input	digital output	COM2	TK17/LED[17]	high impedance	high impedance	high impedance
P03S	high impedance	digital input	digital output	COM3	TK16/LED[16]	SEG9	high impedance	high impedance
P04S	high impedance	digital input	digital output	COM4	TK15/LED[15]	SEG8	high impedance	high impedance
P05S	high impedance	digital input	digital output	SEG7	TK14/LED[14]	high impedance	high impedance	high impedance
P06S	high impedance	digital input	digital output	SEG6	TK13/[13]	high impedance	PWM5	high impedance
P07S	high impedance	digital input	digital output	SEG5	TK12/LED[12]	high impedance	PWM4	high impedance
P10S	high impedance	digital input	digital output	high impedance	TK0/LED[0]	ADC_CH0/ ADC_VREF	high impedance	high impedance
P11S	high impedance	digital input	digital output	UART1_TX	TK1/LED[1]	ADC_CH1	I2C_SDA	
P12S	high impedance	digital input	digital output	UART1_RX	TK2/LED[2]	ADC_CH2	I2C_SCL	high impedance
P13S	high impedance	digital input	digital output	high impedance	TK3/LED[3]	ADC_CH3	high impedance	high impedance
P14S	high impedance	digital input	digital output	high impedance	TK4/LED[4]	ADC_CH4	high impedance	high impedance
P15S	high impedance	digital input	digital output	high impedance	TK5/LED[5]	ADC_CH5	PWM0	high impedance

P16S	high impedance	digital input	digital output	high impedance	TK6/LED[6]	ADC_CH6	PWM1	high impedance
P17S	high impedance	digital input	digital output	SEG0	TK7/LED[7]	ADC_CH7	high impedance	high impedance
P20S	high impedance	digital input	digital output	RESET	-	-	-	-
P21S	high impedance	digital input	digital output	TK_CAP	-	-	-	-
P30S	high impedance	digital input	digital output	UART0_TX	IIC_SDA	high impedance	high impedance	high impedance
P31S	high impedance	digital input	digital output	UART0_RX	IIC_SCL	high impedance	high impedance	high impedance
P32S	high impedance	digital input	digital output	32K_XIN	-	-	-	-
P33S	high impedance	digital input	digital output	32K_XOUT	-	-	-	-
P34S	high impedance	digital input	digital output	SEG4	TK11/LED[11]	high impedance	PWM3	high impedance
P35S	high impedance	digital input	digital output	SEG3	TK10/LED[10]	high impedance	PWM2	high impedance
P36S	high impedance	digital input	digital output	SEG2	TK9/LED[9]	T2CP	high impedance	high impedance
P37S	high impedance	digital input/T2EX	digital output	SEG1	TK8/LED[8]	high impedance	high impedance	high impedance

15.3 Pin control Example

- **Set the Pin function**

For instance, P20 is set to be push-pull output, the program is like：

```
------------------------------------------------------------------------------------------
P20F = 2;
------------------------------------------------------------------------------------------
```

P20 is set to be open-drain output, the program is like：

```
------------------------------------------------------------------------------------------
P20F = (1<<5)|2;
------------------------------------------------------------------------------------------
```

P20 is set to be open-drain output with pull-up enabled, the program is like：

```
------------------------------------------------------------------------------------------
P20C |= (1<<5);
 P20F = (1<<7)｜(1<<5)｜2;
------------------------------------------------------------------------------------------
```

P20 is set to be input with weak pull-up enabled, the program is like：

```
------------------------------------------------------------------------------------------
P20C &= ~(1<<5);
P20F = (1<<7)｜1;
------------------------------------------------------------------------------------------
```

P20 is set to be LCD/LED SEG31, the program is like：

```
------------------------------------------------------------------------------------------
P20F = 3;
------------------------------------------------------------------------------------------
```

P00 is set to LED COM and enables high sink current mode with the following program.

```
  ------------------------------------------------------------------------------------------
P00C |= (1<<7);
P00F = 3;
------------------------------------------------------------------------------------------
```

16 Universal Asynchronous Receiver/Transmitter(UART)

16.1 UART0

16.1.1 Function Introduction

UART0 is a full duplex synchronous/asynchronous serial data transceiver compatible with standard 8051. UART0 receiver includes a one byte buffer which means the received one byte data will be sent to the buffer and the receiver can receive new data at the same time. The previous data must be read before the current data is received completely, otherwise it will be covered by the new data. Register S0BUF is the data transmit/receive register for UART0. In fact, S0BUF includes two registers physically: one is the data transmit register and the other is data receive register. Writing S0BUF will write data into the transmit register and start the data transmission, while read S0BUF will read one byte data from the receiver register.

There are four working modes for UART0 which is shown as the table 16-1-1-1.

Table 16-1-1-1 UART0 Communication Mode

SM00	SM10	Mode	Description	Baud rate
0	0	0	Synchronous shift mode	Fclk/12
0	1	1	8 bit asynchronous mode	The baud rate is (2*SMOD)*CPUCLK*(overflow rate of Timer 1/2)/32, please refer to UCKS of T2CON
1	0	2	9 bit asynchronous mode	When SMOD＝0, the baud rate is Fclk/64 When SMOD＝1, the baud rate is Fclk/32
1	1	3	8 bit asynchronous mode	The baud rate is (2*SMOD)*CPUCLK*(overflow rate of Timer 1/2)/32, please refer to UCKS of T2CON

Note： Since the clock of Timer2 is directly from system clock without frequency division, hence when Timer2 is set to the clock for UART0, the baud rate will be higher. When the system clock frequency is 3.6864MHz, the baud rate can be at most 115200.

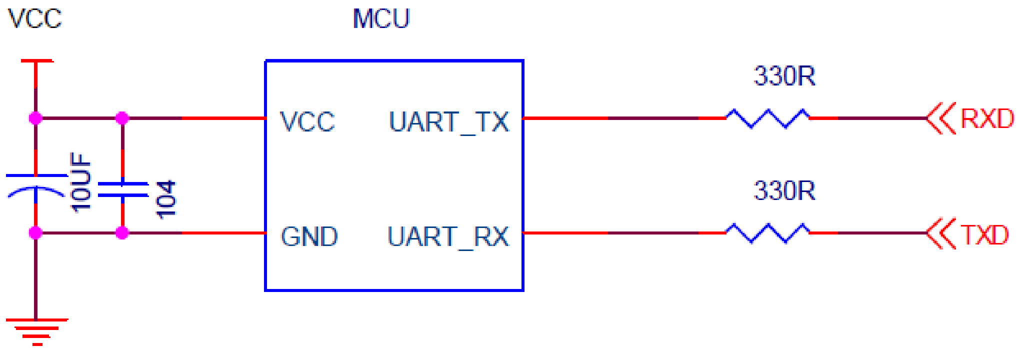


Figure 16-1-1-2 UART0/1 Reference Circuit Diagram

- **Mode0**

UART0 transmits/receives data synchronously in Mode0. Pin TX outputs the shift clock. Pin RX is used to transmit/receive data. The transmission data is 8 bits and the transfer starts from the least significant bit. The baud rate is 1/12 of the main clock frequency. Writing data to register S0BUF will starts the UART0 transmission. On the other hand, REN of register S0CON must be 1 and RI0 flag must be cleared when it is used as a receiver. Once a one-byte data is received, the RI0 will be set to 1.

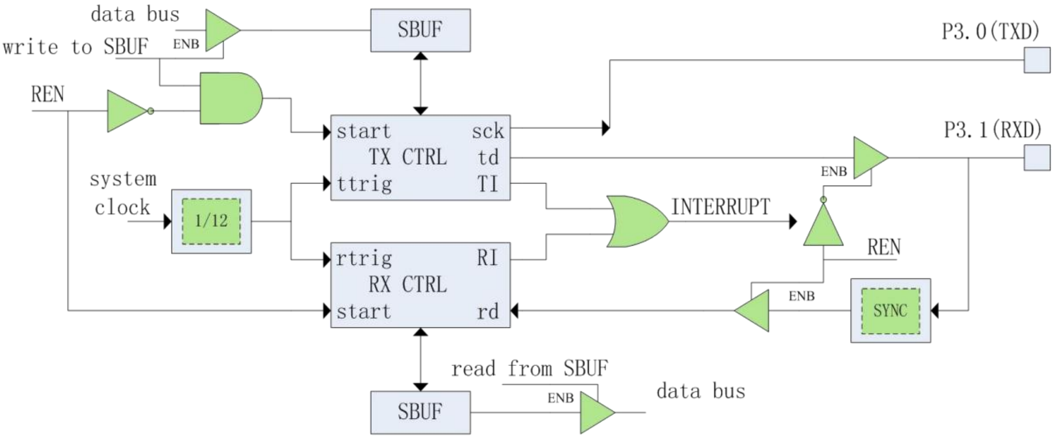


Figure 16-1-1-3 UART0 Mode0 Schematic

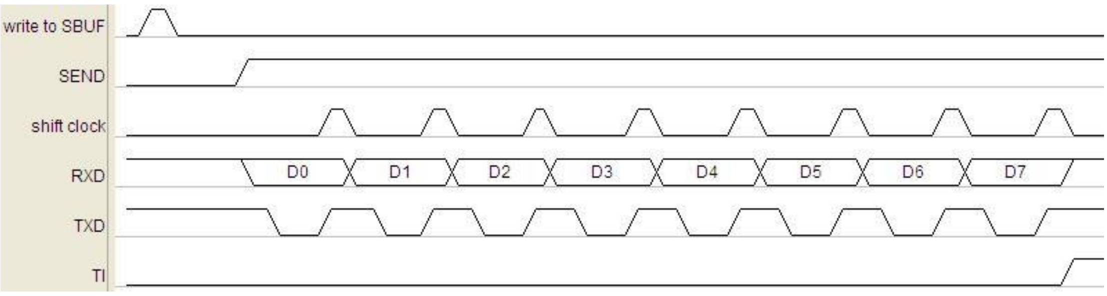


Figure 16-1-1-2 UART0 Data Transmission Waveform in Mode0

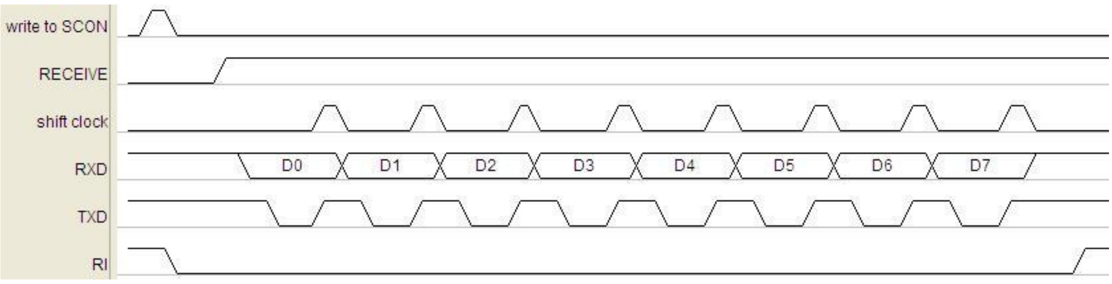


Figure 16-1-1-3 UART0 Data Receiving Waveform in Mode0

- **Mode1**

UART0 can transmit and receive 8 bit data asynchronously at the same time in Mode1. Either the overflow signal of Timer1 or Timer2 can be selected as the UART0 clock by setting UCKS (please refer to register T2CON). Thus, setting overflow rate will modify the UART0 baud rate as well. The SMOD(please refer to register PCON) can be used to select the whether the baud rate will be doubled.

Writing data register S0BUF will starts UART0 transmission. The first bit transmitted is the start bit (which is 0), and then the 8 bit data follows (with the least significant bit transmitted first). The last bit transmitted is the stop bit (which is 1).

When UART0 is used as receiver, it is synchronized by detecting the falling edges of Pin RX. The 8 bit data will be stored in register S0BUF after the transmission is completed with efficient stop bit’s value stored in RB80(S0CON[2]).

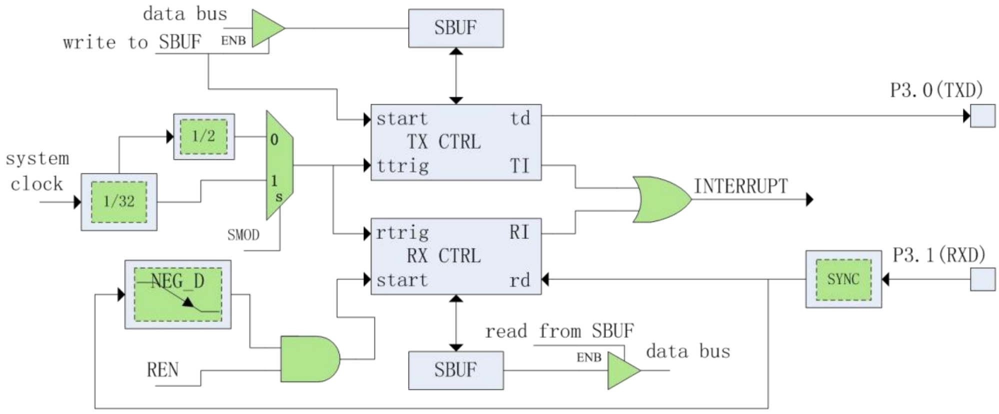


Figure 16-1-1-4 UART0 Mode1 Schematic

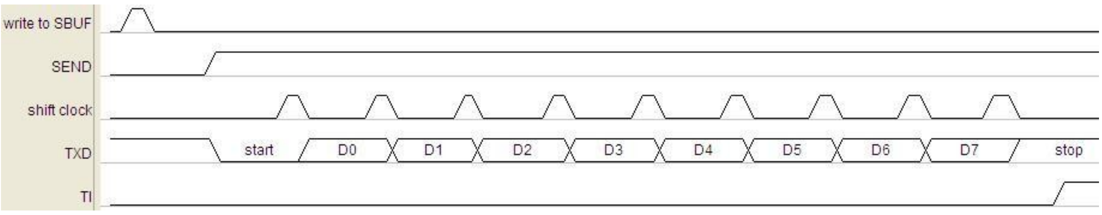


Figure 16-1-1-5 UART0 Data Transmission Waveform in Mode1

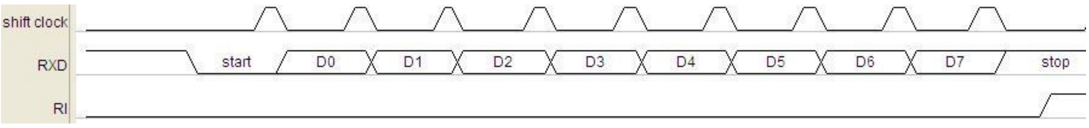


Figure 16-1-1-6 UART0 Data Receiving Waveform in Mode1

- **Mode2**

UART0 sends and receives 9 bit data asynchronously and simultaneously in mode2. The baud rate can be Fsys/32 or Fsys/64 selected by SMOD in register PCON.

Writing data to register S0BUF will starts UART0 transmission. The first bit transmitted is the start bit (which is 0), and then the 9 bit data follows (with the least significant bit transmitted first). The ninth data bit is the TB80 of register S0CON. The last bit transmitted is the stop bit (which is 1).

The transfer process is turned on by writing to the S0BUF register. The first bit transmitted is the start bit (which is 0), and then the 9 bit data follows with the least significant bit transmitted first. The ninth data bit is the TB80 of register S0CON. The last bit transmitted is the stop bit (which is 1).

When UART0 is used as receiver, it is synchronized by detecting the falling edges of Pin RX. The lower 8 bit data will be stored in register S0BUF after the transmission is completed with the 9th data bit stored in RB80(S0CON[2]).

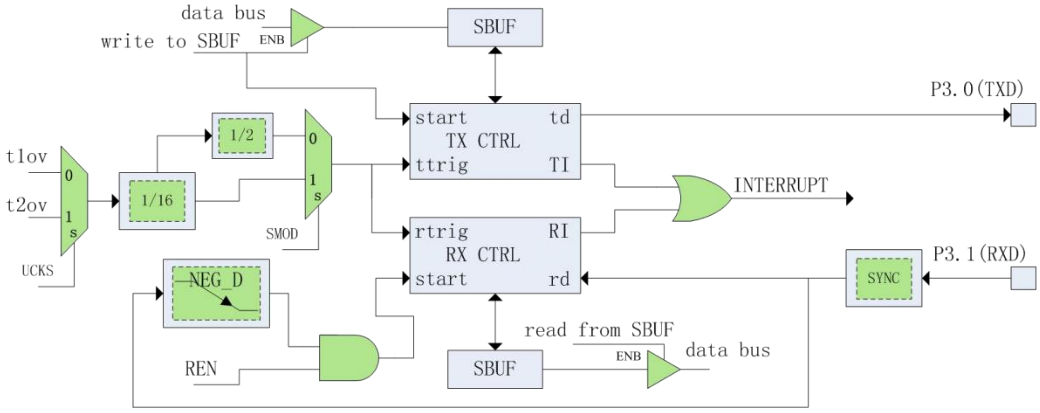


Figure 16-1-1-7 UART0 Mode2 Schematic

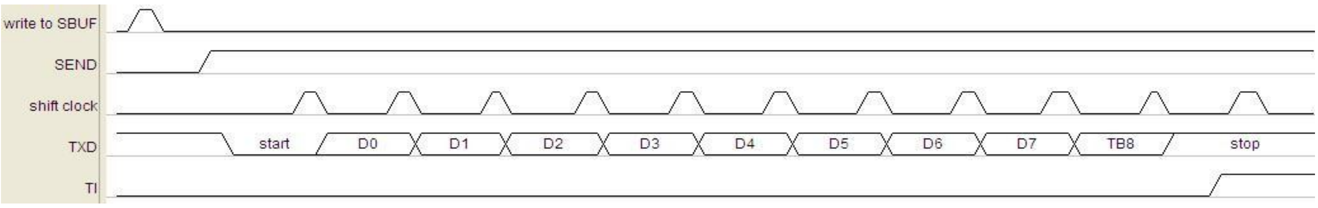


Figure 16-1-1-8 UART0 Data Transmission Waveform in Mode2

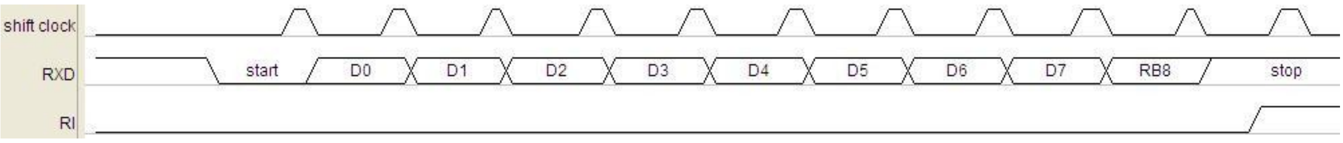


Figure 16-1-1-9 UART0 Data Receiving Waveform in Mode2

- **Mode3**

The only difference between mode2 and mode3 is that the baud rate in mode3 can be generated by Timer1 or Timer2, referring to the mode1 schematic. For the baud rate setting please refer to mode1 and for other functions please refer to mode2.

- **UART0 Multi-computer Communication**

Multi-computer Communication can also be realized by UART0 in mode2 and mode3. If SM20 of register S0CON is set to 1, only when the 9th data is 1 (RB80=1) ,the slave will generate receive interrupt, which makes multi-computer communication possible. The slaves can set their SM20 to 1 and host set the 9th data bit to 1 when it transfers address to the slaves. All the slaves will generate receive interrupt and the slaves' software then compare the address received to their own addresses. It the address matches, the matched slave will set SM20=0. The host then set the 9th data bit to 0 for the following data transmission. Due to the other slaves remain SM20 = 1, thus only the address matched slave will generate receive interrupt.

16.1.2 Register Description

Table 16-1-2-1 Register S0CON

98H	7	6	5	4	3	2	1	0
S0CON	SM00	SM10	SM20	REN0	TB80	RB80	TI0	RI0
R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7	SM00	UART0 mode selection, for more information please refer to Table 16-1-1-1
6	SM10	
5	SM20	Multi-computer communication enable control, 1 enables
4	REN0	Serial receive enable control, 1 enables
3	TB80	The 9th data bit to transmit It will be transmitted as the 9th bit of the data in mode 2 and mode 3 and it is controlled by the software (For instance, parity check or multi-computer communication)
2	RB80	The 9th bit of the data received It will be used for UART0 to receive the 9th bit of the data in mode2 and mode3. It is the stop bit in mode1; if SM2=1, it is the token bit for multi-host； it is not used in mode0
1	TI0	Transmit interrupt flag, 1 indicates the interrupt, cleared by writing 0 to it
0	RI0	Receive interrupt flag, 1 indicates the interrupt, cleared by writing 0 to it

Table 16-1-2-2 Register S0BUF

99H	7	6	5	4	3	2	1	0
S0BUF	S0BUF[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~0	S0BUF	Receiver/Transmitter buffer Writing data to S0BUF will starts the data transmission Reading S0BUF will reads the data received

16.2 UART1

16.2.1 Introduction

UART1 is a full duplex asynchronous serial data transceivers and includes one byte buffer as well. There are two operating modes for UART1 as Table 16-2-1-1 shows.

Table 16-2-1-1 UART1 Operating Modes

SM1	Mode	Description	Baud rate
0	A	9 bit asynchronous mode, the same as UART0 mode2 and mode3	CPUCLK/(32*(1024-S1REL))
1	B	8 bit asynchronous mode, the same as UART0 mode1	CPUCLK/(32*(1024-S1REL))

The principle of UART1 is the same as UART0 asynchronous mode(mode 1/2/3) with different baud rate configuration. The waveform of UART0 can be the reference waveform for both modeA and modeB. Unlike UART0, UART1 include a special baud rate generator hence the baud rate will be configured by register SxRELL and SxRELH.

Note： Whether UART1 baud rate will be doubled can not be set by SMOD in register PCON.

Figure 16-2-1-1 shows the UART1 principle schematic.

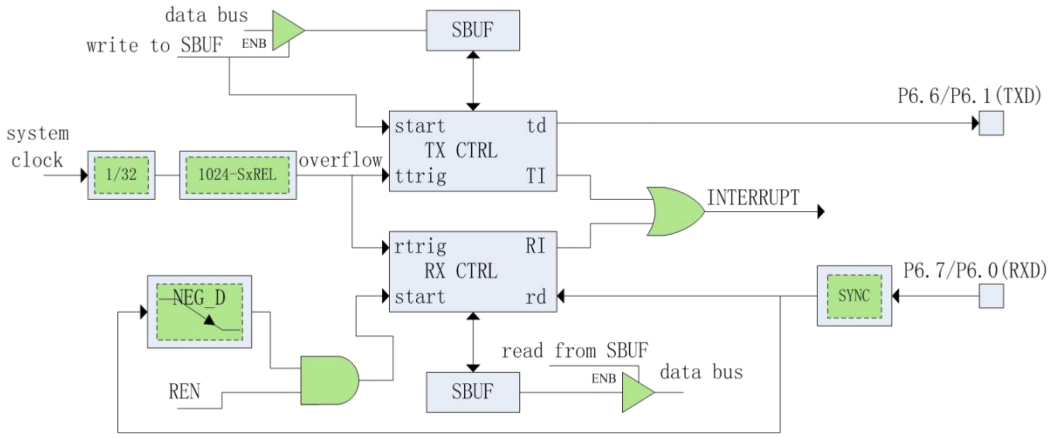


Figure 16-2-1-1 UART1 Principle Schematic

- **ModeA**

UART1 can transmit/receive 9 bit data asynchronously in ModeA. Writing data to register SxBUF will starts UART1 transmission. The first bit transmitted is the start bit (which is 0), and then the 9 bit data follows (with the least significant bit transmitted first).The ninth data bit is the TB81 of register S1CON. The last bit transmitted is the stop bit (which is 1).When UART0 is used as receiver, it is synchronized by detecting the falling edges of Pin RX. The lower 8 bit data will be stored in register S1BUF after the transmission is completed with the 9th data bit stored in RB81.

- **ModeB**

The difference between ModeA and ModeB is that only the transmission data is only 8 bits with the efficient stop bit stored. ModeA and ModeB are the same for other functions.

- **UART1 Multi-computer Communication**

Multi-computer Communication can also be realized by UART1 in ModeA. If SM21 of register S1CON is set to 1, only when the 9th data is 1 (RB81=1) the slave will generate receive interrupt, which makes multi-computer communication possible. The slaves can set their SM21 to 1 and host set the 9th data bit to 1 when it transfers address to the slaves. All the slaves will generate receive interrupt and the slaves' software then compare the address received to their own addresses. It the address matches, the matched slave will set SM21=0. The host then set the 9th data bit to 0 for the following data transmission. Due to the other slaves remain SM21 = 1, thus only the address matched slave will generate receive interrupt.

16.2.2 UART1 Register Description

Table 16-2-2-1 Register S1CON

9AH	7	6	5	4	3	2	1	0
S1CON	SM1	U1IE	SM21	REN1	TB81	RB81	TI1	RI1
R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7	SM1	UART1 mode selection, for more information please refer to Table 16-2-1-1
6	U1IE	UART1 interrupt enable control, 1 enables it
5	SM21	Multi-computer communication enable control, 1 enables it
4	REN1	Serial receive enable control, 1 enables it
3	TB81	The 9th data bit to transmit It will be transmitted as the 9th bit of the data in modeA and it is controlled by the software (For instance, parity check or multi-computer communication)
2	RB81	The 9th bit of the data received It will be used for UART0 to receive the 9th bit of the data in modeA. It is the stop bit received in modeB
1	TI1	Transmit interrupt flag, 1 indicates the interrupt, cleared to 0 by writing 1 to it
0	RI1	Receive interrupt flag, 1 indicates the interrupt, cleared to 0 by writing 1 to it

Table 16-2-2-2 Register S1BUF

9BH	7	6	5	4	3	2	1	0
S1BUF	S1BUF[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
7~0	S1BUF	Receiver/Transmitter buffer for UART1 Writing data to S1BUF will starts the written data transmission Reading S1BUF will reads the data received						

Table 16-2-2-3 Register S1RELL、S1RELH

9CH	7	6	5	4	3	2	1	0
S1RELL	S1RELL[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
9DH	7	6	5	4	3	2	1	0
S1RELH	-	-	-	-	-	-	S1REL[9:8]	
R/W	-	-	-	-	-	-	R/W	
Initial Value	-	-	-	-	-	-	0	0
Bit number	Bit Symbol	Description						
9~0	S1REL	Baud rate configuration register The baud rate is CPUCLK/(32 * (1024 - S1REL))						

17 I²C Interface

17.1 Function Introduction

I²C modules enables the chip to communicate with peripheral I²C devices by serial transmission standard which complies with standard I²C specification. It can be set to either slave or master and configured to standard/fast/high speed mode.

17.2 I²C Main Features

- Simple but strong communication port, bi-directional bus with 2 wires
- Slave/Master mode configurable
- Able to operate in receiver/transmitter mode
- 7 bit slave address
- Supports multimaster's arbitration
- Broadcast function supported

17.3 I²C Function Description

I²C modules supports I²C standard bus specification. I²C bus includes 2 wires to transfer data among devices, one is SCL(Serial Clock) and the other is SDA(Serial Data), as Figure 19-3-1 shows. Since the it is open-drain port for I²C, there must be pull-up resistor on I²C bus. The pull-up resistor can be connected externally or enabled internally. Each device that connects to the bus has its own 7-bit address.

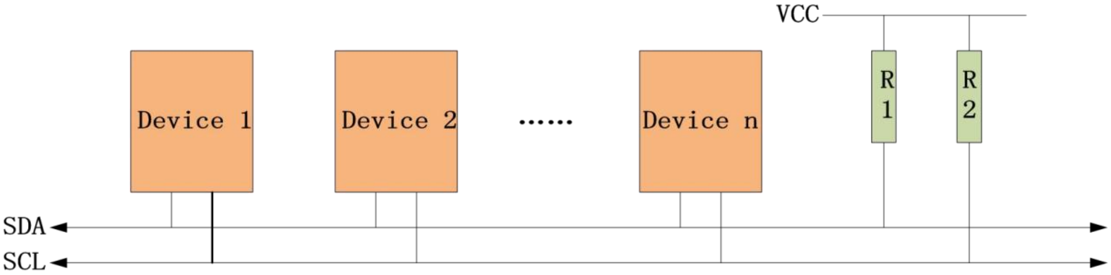


Figure 17-3-1 I2C Bus Interconnection

I²C module principle is as Figure 17-3-2 shows.

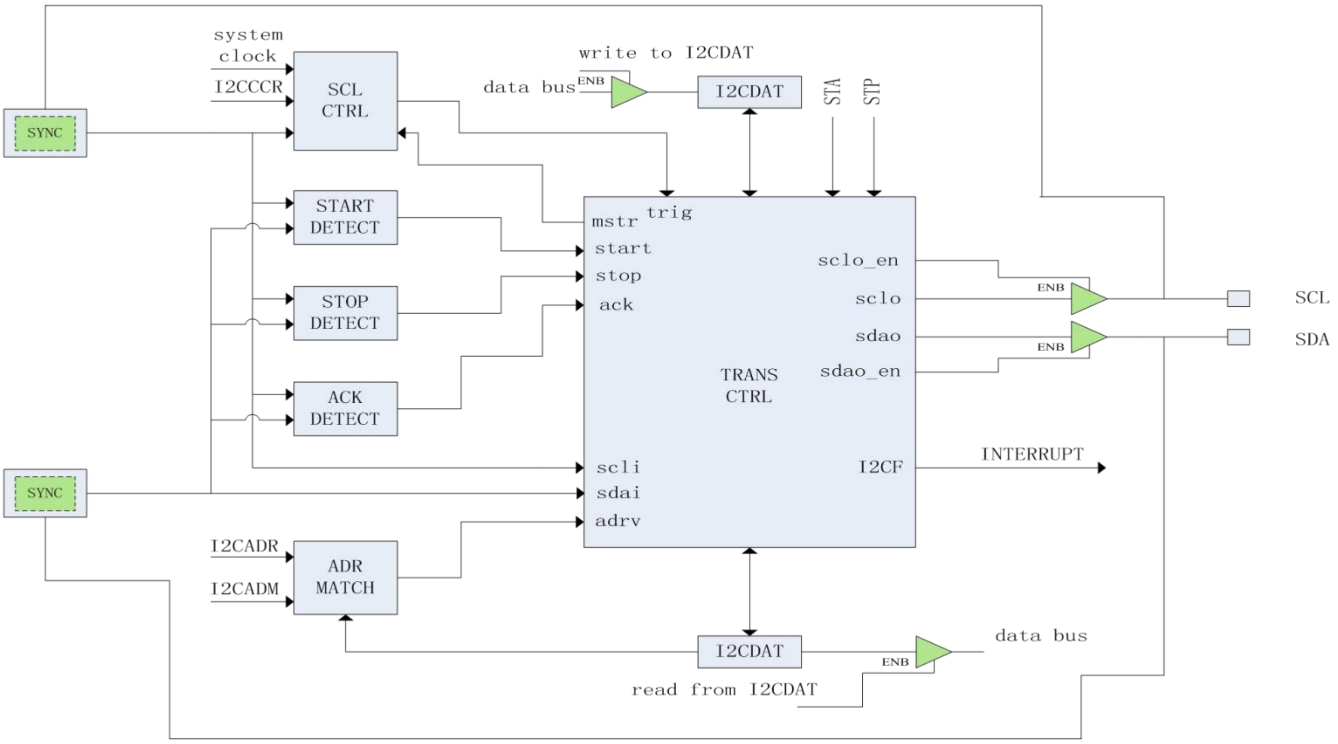


Figure 17-3-2 I2C Module Schematic

- **I²C Mode Selection**

I²C can operates in the following 4 modes： slave transmit mode, slave receive mode, master transmit mode, master receive mode. I²C operates in slave mode by default. I²C changes to master mode after the START signal generated and returns slave mode when the arbitration fails or STOP signal is generated.

- **I2C Bus Data Transmission Pattern**

There are usually 4 stages for the standard I²C communication: START signal, slave address transfer, data transmission and STOP signal. The data transmitted on I²C bus is always 8 bits with the most significant bit sent first. There must be a ACK following every one byte data. However, there is no byte limits for the data transmission. The master sends STOP signal after the transmission is over and terminates the communication.

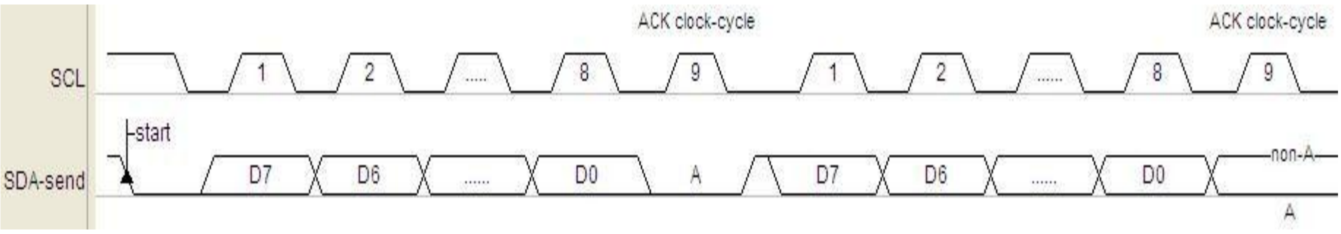


Figure 17-3-3 I2C Bus Data Transmission Format

- **Communication Process**

I²C enables the data transmission and generates the clock signal in Master mode. The serial data transmission always begins with START signal and ends with STOP signal. Both START and STOP signals are generated by the software in master mode. Setting STA=1 generates START signal and setting STP=1 generates STOP signal, I²C can distinguish its address (7 bits) and the broadcast address in slave mode. Software can enable/disable its ability to recognize broadcast address by setting GCE.

Both address and data are transmitted in bytes. The address will be sent by the master after the START signal. The receiver must reply with a ACK signal in the 9th clock cycle after one byte information is transferred. The ACK can be set by AAK while it must be set before the one byte information transfer completes. When the one byte information is received, the ACK signal will be generated automatically.

Every time when one byte data is received/transmitted or arbitration fails (and etc.) there will be an interrupt flag I2CF. The status of the event will be indicated by register I2CSTA (for more information please refer to register I2CSTA). The software decides the next operation according to the status of the event when interrupt occurs. Clearing the interrupt flag I2CF will start the next operation.

When there occurs interrupt I2CF, if SHD=1, SCL will be set to low by slave before I2CF is cleared. After the master detects that SCL is released, it master will then continue the next operation. On the other hand, if SHD=0, SCL will not be set to low by the slave, which makes it compatible with applications when the master I^2C is simulated by software. Thus, the master's software must wait long enough so that the slave can deal with the response to every one byte data.

- **I2C Clock Settings**

When the I2C interface is used as a slave, the clock of SCL is input by the host and is independent of the clock configuration of the slave. When acting as a slave, the sampling clock of I2C is set by SMPDIV (I2CCCR[7:5]), and the filtering function is automatically activated when SMPDIV is not 0. When acting as a master, the output clock frequency of SCL is determined by SMPDIV and I2CCKD(I2CCCR[4:0]) (please check the introduction of register section for details).

17.4 I^2C Communication Pin Mapping

There are different mappings for I^2C communication pins which could be selected by register I2CIOS. For more information please refer to register I2CIOS description.

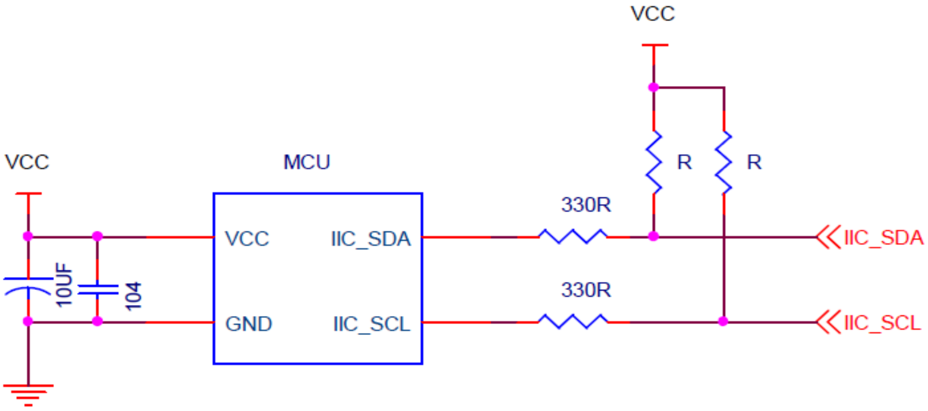


Figure 17-4-1 IIC Reference Circuit Diagram

17.5 Register Description

Table 17-5-1 Register I2CCON

B1H	7	6	5	4	3	2	1	0
I2CCON	I2CE	I2CIE	STA	STP	SHD	AAK	CBSE	STFE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	1	0	0	0	0	1	0	0

Bit number	Bit Symbol	Description
7	I2CE	I^{2}C module enable control, 1 enables it
6	I2CIE	I^{2}C interrupt enable control, 1 enables it1
5	STA	I^{2}C START signal transfer control, valid when it is 1, it will be cleared to 0 automatically when START signal detected
4	STP	I^{2}C STOP signal transfer control, valid when it is 1, it will be cleared to 0 automatically when STOP signal detected
3	SHD	When it is 1, if I2CF=1, I2CF will make SCL remain low after SCL becomes low
2	AAK	I^{2}C ACK signal transfer control, 1 enables it Note： When I^{2}C is configured as slave, this bit must be set to 1 beforehand, otherwise even the address matches it will not reply ACK
1	CBSE	CBUS compatible enable control When it is set to 1, the ACK will be ignored during the transmission to be compatible with CBUS bus. Since the address for CBUS bus is 7 bits, thus HCE must be set to 0.
0	STFE	When STFE=1, I2CF will be set to 1 if I^{2}C module detects the START signal

Table 17-5-2 Register I2CADR

B2H	7	6	5	4	3	2	1	0
I2CADR	GCE	I2CADR[6:0]						
R/W	R/W	R/W						
Initial Value	1	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7	GCE	Broadcast address recognition(00H)enable control, 1 enables it
6~0	I2CADR	I2C slave address, only valid when it operates as slave Note： (when AAK=1) when the address is 7 bits and the higher 7 bits of first received address matches I2CADR, reply with ACK and enters slave mode

Table 17-5-3 Register I2CADM

B3H	7	6	5	4	3	2	1	0
I2CADM	SPFE	I2CADM[6:0]						
R/W	R/W	R/W						
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7	SPFE	When SPFE=1, I2CF will be set to 1 if I2C module detects the STOP signal
6~0	I2CADM	I2C address mask by bit control, valid only when it operates as slave When I2CADM[n](n=0~6)=1, the corresponding address bit I2CADR[n] will not be compared (which means no matter what is received, it is seen as matched)

Table 17-5-4 Register I2CCCR

B4H	7	6	5	4	3	2	1	0
I2CCCR	SMPDIV[2:0]			I2CCKD[4:0]				
R/W	R/W							
Initial Value	0	0	1	0	0	0	0	0

Bit Number	Bit Symbol	Description
7~5	SMPDIV	I2C sample clock setting, the I2C sample clock is the smpdiv power division of 2 of the I2C operating clock, i.e. 000：F_{sample}=F_{i2cclk} 001：F_{sample}=F_{i2cclk}/2 010：F_{sample}=F_{i2cclk}/4 ... 111：F_{sample}=F_{i2cclk}/128
4~0	I2CCKD	I2C SCL output clock frequency setting, SCL output clock frequency is the (I2CCKD + 1) division of the sampling frequency, i.e. F_{scl}= F_{sample} /(I2CCKD +1)) *Note:* *1. When SMPDIV= 0, if the setting I2CCKD is less than 9, it will be automatically calculated by 9.* *2. When SMPDIV>0, if the setting I2CCKD is less than 7, it will be calculated by 7 automatically.* ***Note:*** *1. When I2CCCR[7:5] = 0, if a value less than 9 is written to I2CCCR[4:0], the value of 9 will be calculated automatically* *2. When I2CCCR[7:5] > 0, if a value less than 7 is written to I2CCCR[4:0], the value of 7 will be calculated automatically.*

Table 17-5-5 Register I2CDAT

B5H	7	6	5	4	3	2	1	0
I2CDAT	I2CDAT[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~0	I2CDAT	Data buffer for receiving/transmission *Note：* *When I2CF is 1, it is recommended to make I2CF remain 1 when users overwrite/read I2CDAT. I2CF should be cleared after the process is over, and then the transmission continues so that there will be no transmission errors.*

Table 17-5-6 Register I2CSTA

B6H	7	6	5	4	3	2	1	0
I2CSTA	I2CSTA[7:0]							
R/W	R							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~0	I2CSTA	I^{2}C status register 00H: (master/slave) bus error 08H: (master/slave)START signal detected (valid only when STFE=1) 18H: (master)address and write bit sent, ACK signal received 20H: (master)address and write bit sent, no ACK signal received 28H: (master)one byte data received/transmitted, ACK signal detected 30H: (master)one byte data received/transmitted, no ACK signal detected 38H: (master)arbitration lost(master will change to slave after arbitration lost) 40H: (master)address and read bit transmitted, ACK signal received 48H: (master)address and read bit transmitted, no ACK signal received 60H: (slave)address and write bit received, with ACK signal is sent 70H: (master/slave)broadcast address received with ACK signal is sent(master/slave will become slave) 80H: (slave)one byte data received/transmitted, ACK signal detected 88H: (slave)one byte data received/transmitted, no ACK signal detected A0H: (master/slave)STOP signal detected(valid only when SPFE=1) A8H: (slave)address and read bit received, with ACK signal is sent F8H: (master/slave) bus is idle

Table 17-5-7 Register I2CFLG

B7H	7	6	5	4	3	2	1	0
I2CFLG	-	-	-	-	-	-	-	I2CF
R/W	-	-	-	-	-	-	-	R
Initial Value	-	-	-	-	-	-	-	0

Bit number	Bit Symbol	Description
7~1	-	-
0	I2CF	I^{2}C interrupt flag, 1 indicates the interrupt, cleared to 0 by writing 1 to it Note： *1. I2CF will be set to 1 every time after a one-byte data or the address transmission completes (with ACK/NAK received/sent).* *2. I2CF will be set to 1 when there is bus error.* *3. If STFE=0, I2CF will not be set to 1 when START signal detected.* *4. If SPFE=0, I2CF will not be set to 1 when STOP signal detected.*

Table 17-5-8 Register I2CIOS

8101H	7	6	5	4	3	2	1	0
I2CIOS	I2CKS	-	-	-	-	-	-	I2CS
R/W	R/W	-	-	-	-	-	-	R/W
Initial Value	0	-	-	-	-	-	-	0

Bit Number	Bit Symbol	Description
7	I2CKS	I2C operating clock select bit 0: System Clock 1: IRCH
6~1	-	-
0	I2CS	I2C Pin Select Bit 0: SCL on pin P31, SDA on pin P30 1: SCL on pin P12, SDA on pin P11

17.6 I^{2}C Control Example

- **I^{2}C as master**

For instance , the master sends 20 byte data to the slave cyclically, the program is like：

```
//I2CCON 定义
#define I2CE(N)                (N<<7)
#define I2CIE(N)               (N<<6)
#define STA(N)                         (N<<5)
#define STP(N)                 (N<<4)
#define CKHD(N)                (N<<3)
#define AAK(N)                 (N<<2)
#define CBSE(N)                (N<<1)
#define STFE(N)                (N<<0)
//I2CADR 定义
#define   GCE(N)               (N<<7) //N = 0~1
//I2CFLG 定义
#define I2CF                           (1<<0)

#define I2C_ADDR               0xCA              // Define the I2C slave address
unsigned char xdata WriteBuffer[20]={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19};
void main(void)
{
    unsigned char i;
   EA = 1;                                       // Enable global interrupt
/**********选择 I2C 端口*************************************************************/
//   I2CIOS = 0;                         // Select P30,P31 as I2C communication pins
//   P30F = 4 | (1<<7);                  // Set P30 as I2C SDA and turn on the upper
//   P31F = 4 | (1<<7);                  // Set P31 as I2C SCL and turn on the upper

   I2CIOS = 1;                           // Select P11,P12 as I2C communication pins
     P11F = 6|(1<<7);                    // Set P11 as I2C SDA and turn on the upper
     P12F = 6|(1<<7);                    // Set P12 as I2C SCL and turn on the upper
// Choose one of the above two groups of ports
/*************************************************************************************/
   I2CCON = I2CE(1) | I2CIE(0) | STA(0) | STP(0)| CKHD(1) | AAK(1)| CBSE(0) | STFE(1);
   I2CADR = GCE(0);
   I2CCCR = 0x29;                                          // Setting the I2C clock
   while(1)
   {
         I2CCON |= STA(1);               // I2C host sends START signal
         while(!(I2CFLG & I2CF));                  // Waiting for the interrupt flag to be generated
         if(I2CSTA != 0x08)
         {
            I2CFLG  |= I2CF;
            goto SEND_STOP;
         }
         I2CDAT = I2C_ADDR;                                // Host sends slave address + write bit
         I2CFLG  |= I2CF;                                  // Clear the interrupt flag
         while(!(I2CFLG & I2CF));                  // Waiting for the interrupt flag to be generated
         if(I2CSTA != 0x18)
         {
            I2CFLG  |= I2CF;
            goto SEND_STOP;
         }
```

```
            I2CDAT = 0;                        // Host send data register address
            I2CFLG  |= I2CF;                                  // Clear the interrupt flag
            while(!(I2CFLG & I2CF));                // Waiting for the interrupt flag to be generated
            if(I2CSTA != 0x28)
            {
                I2CFLG  |= I2CF;
              goto SEND_STOP;
            }
            for(i = 0; i < 20; i++)                               //Host sends 20 data
            {
              I2CDAT =WriteBuffer[i];
              I2CFLG  |= I2CF;                                // Clear the interrupt flag
              while(!(I2CFLG & I2CF));              // Waiting for the interrupt flag to be generated
              if(I2CSTA != 0x28)
              {
                      I2CFLG  |= I2CF;
                      goto SEND_STOP;
              }
            }
SEND_STOP:
            I2CCON |= STP(1);                                 // Send STOP signal
            I2CFLG  |= I2CF;
            Delay_ms(100);
        }
}
```

For example, if the host reads 20 bytes of data cyclically from the slave, the program is as follows.

```
#define I2C_ADDR              0xCA               // Define the I2C slave address
unsigned char xdata ReadBuffer[20];
void main(void)
{
    unsigned char i;
    EA = 1;                             // Open global interrupt
/**********选择 I2C 端口*************************************************************/
//   I2CIOS = 0;                        // Select P30,P31 as I2C communication pins
//   P30F = 4 | (1<<7);                 // Set P30 as I2C SDA and turn on the upper
//   P31F = 4 | (1<<7);                 // Set P31 as I2C SCL and turn on the upper

    I2CIOS = 1;                         // Select P11,P12 as I2C communication pins
      P11F = 6|(1<<7);                  // Set P11 as I2C SDA and turn on the upper
      P12F = 6|(1<<7);                  // Set P12 as I2C SCL and turn on the upper
// Choose one of the above two groups of ports
/**********************************************************************************/
    I2CCON = I2CE(1) | I2CIE(0) | STA(0) | STP(0)| CKHD(1) | AAK(1)| CBSE(0) | STFE(1);
    I2CADR = GCE(0);
    I2CCCR = 0x29;                                // Setting the I2C clock
    while(1)
    {
              I2CCON |= STA(1);                               // I2C host sends START signal
              while(!(I2CFLG & I2CF));              // Waiting for the interrupt flag to be generated
              if(I2CSTA != 0x08)
              {
                      I2CFLG  |= I2CF;
                      goto SEND_STOP;
              }
              I2CDAT = I2C_ADDR;                // Host sends slave address + write bit
              I2CFLG  |= I2CF;                                // Clear the interrupt flag
```

```
        while(!(I2CFLG & I2CF));                // Waiting for the interrupt flag to be generated
        if(I2CSTA != 0x18)
        {
                I2CFLG  |= I2CF;
                goto SEND_STOP;
        }

        I2CDAT = 0;                                     // Host send data register address
        I2CFLG  |= I2CF;                                // Clear the interrupt flag
        while(!(I2CFLG & I2CF));                // Waiting for the interrupt flag to be generated
        if(I2CSTA != 0x28)
        {
                I2CFLG  |= I2CF;
                goto SEND_STOP;
        }

        I2CCON |= STA(1);                               // I2C host sends START signal
        I2CFLG  |= I2CF;                                // Clear the interrupt flag
        while(!(I2CFLG & I2CF));                // Waiting for the interrupt flag to be generated
        if(I2CSTA != 0x08)
        {
                I2CFLG  |= I2CF;
                goto SEND_STOP;
        }

        I2CDAT = I2C_ADDR+1;            // Host sends slave address + read bit
        I2CFLG  |= I2CF;                                // Clear the interrupt flag
        while(!(I2CFLG & I2CF));                // Waiting for the interrupt flag to be generated
        if(I2CSTA != 0x40)
        {
                I2CFLG  |= I2CF;
                goto SEND_STOP;
        }
        I2CCON |= AAK(1);                       // Set answer bit

        for(i = 0; i < 20; i++)
        {
                I2CFLG  |= I2CF;                        // Clear the interrupt flag
                while(!(I2CFLG & I2CF));        // Waiting for the interrupt flag to be generated
                if(I2CSTA != 0x28 && I2CSTA != 0x30)
                {
                        I2CFLG  |= I2CF;
                        goto SEND_STOP;
                }
                ReadBuffer[i] = I2CDAT;         // Read data to data register
                if(i < 19)
                {
                        I2CCON |= AAK(1);       // If it is not the last byte, preset ACK status
                }
                else
                {
                        I2CCON &= ~AAK(1);      // If it is the last byte, no ACK is sent
                }
        }
SEND_STOP:
        I2CCON |= STP(1);                               // Send STOP signal
        I2CFLG  |= I2CF;
        Delay_ms(100);
    }
}
```

- **I2C as slave**

As the slave, it supports the master to read or write data to it, the program is like：

```
-----------------------------------------------------------------------------------------------------------------------------------------
#define I2C_ADDR               0xCA                // Define the I2C slave address
unsigned char I2CDataIndex;
unsigned char regAddr;
bit iicReadMode;
unsigned char xdata Buffer[20]={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19};// Set the initial value of data
register to 0~19
void INT6_ISR(void) interrupt 11
{
     unsigned char Sta_Temp;

     if(I2CFLG & I2CF)                             //IIC  interrupt
     {
               Sta_Temp = I2CSTA;
               if(Sta_Temp == 0x60)              // Receive slave address + write bit
               {
                         I2CDataIndex = 0xFF;      // Set to 0xFF means that the first byte received later is the address
                         iicReadMode = 0;          // Set to slave receive status
                         I2CCON |= AAK(1);
               }
               else if(Sta_Temp == 0x80)           // Send or receive one byte of data, an answer signal is detected
               {
                         if(iicReadMode)                    // Send one byte of data
                         {
                         I2CDataIndex++;
                         I2CDAT = Buffer[I2CDataIndex + regAddr]; // Load the data into the transmit register
and wait for the host to read it
                         }
                         else                                              // One byte of data received
                         {
                                   if(I2CDataIndex == 0xFF)           // Address
                                   {
                                             regAddr = I2CDAT;         // The first byte received is considered to be the address
                                             I2CDataIndex = 0;                  // Set the index value to 0
                                             I2CCON |= AAK(1);
                                   }
                                   else                                              // Data
                                   {
                                   Buffer[I2CDataIndex + regAddr] = I2CDAT;// Received data is loaded into the data register
                                             I2CDataIndex++;                     // Index value accumulation
                                             I2CCON |= AAK(1);
                                   }
                         }
               }
               else if(Sta_Temp==0xA8)                          // Receive slave address + read bit, send ACK signal
               {
               I2CDAT = Buffer[I2CDataIndex + regAddr];  // Load the data into the transmit register and wait for
 the host to read it
                         iicReadMode = 1;                          // Set to slave transmit state
               }
               else if(Sta_Temp == 0x88)           // Send or receive one byte of data, an answer signal is detected
               {
               }
               I2CFLG  |= I2CF;                    // Clear the interrupt flag
     }
}

void main(void)
{
    EA = 1;                                         // Open global interrupt
/**********选择 I2C 端口*************************************************************/
//    I2CIOS = 0;                             // Select P30,P31 as I2C communication pins
//    P30F = 4 | (1<<7);                      // Set P30 as I2C SDA and turn on the upper
//    P31F = 4 | (1<<7);                      // Set P31 as I2C SCL and turn on the upper

    I2CIOS = 1;                               // Select P11,P12 as I2C communication pins
      P11F = 6|(1<<7);                        // Set P11 as I2C SDA and turn on the upper
      P12F = 6|(1<<7);                        // Set P12 as I2C SCL and turn on the upper
// Choose one of the above two groups of ports
/******************************************************************************/
    I2CCON = I2CE(1) | I2CIE(1) | STA(0) | STP(0)| CKHD(1) | AAK(1)| CBSE(0) | STFE(0);
    I2CADR = GCE(0)|(I2C_ADDR>>1);                    // Set I2C slave address
    I2CCCR = 0x20;                                            // Setting the I2C sample clock
      INT6EN = 1;                                                     // I2C interrupt on
    while(1)
    {
    }
}
```

----

18 LCD/LED Driver

18.1 LCD Driver

18.1.1 Function Introduction

The internal LCD driver supports at most 5com x 8seg、4comx 9seg、3comx10seg.(13 output pins in all). The programmable duty cycle can be：1/2、1/3、1/4、1/5. Programmable bias voltage could be：1/2、1/3、1/4. There are 8 levels for the driving ability which is also programmable.

Figure 18-1-1-1 shows the principle of LCD.

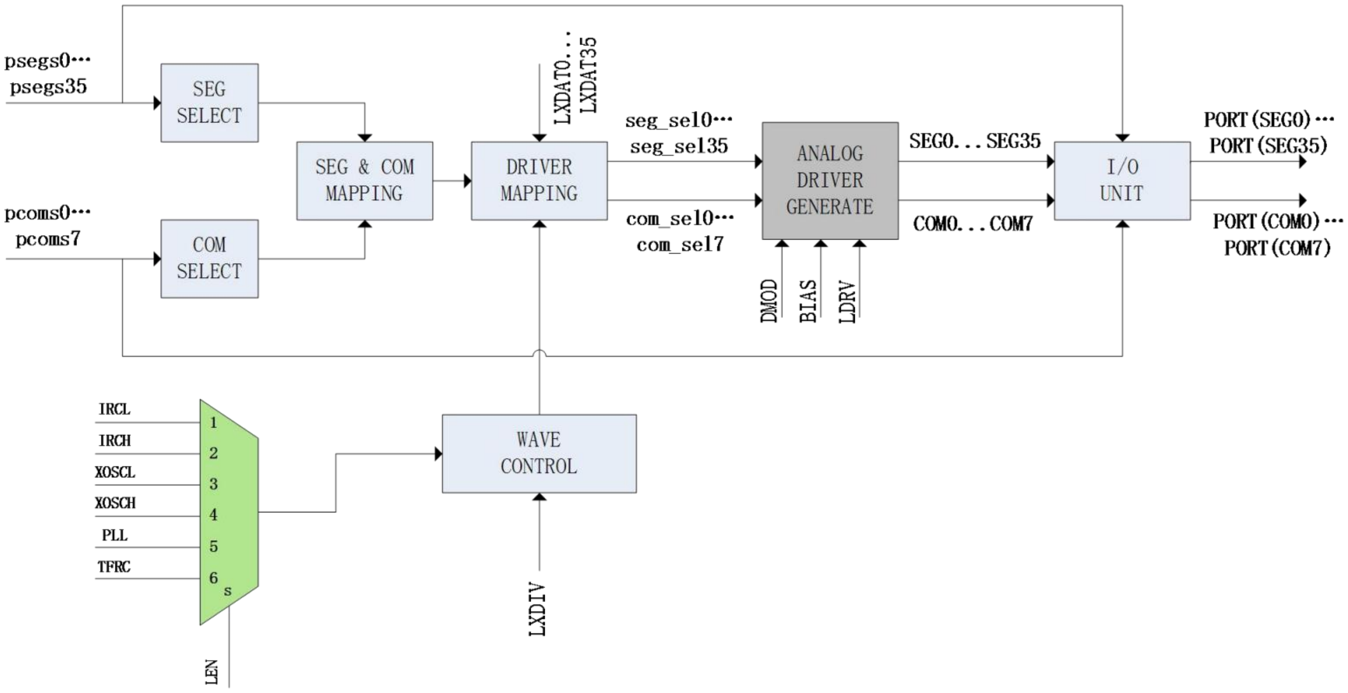


Figure 18-1-1-1 LCD Schematic

18.1.2 LCD Bias Voltage

LCD programmable bias voltage can be: 1/2, 1/3, 1/4. The corresponding signals are as follows.

- **LCD bias voltage 1/2**

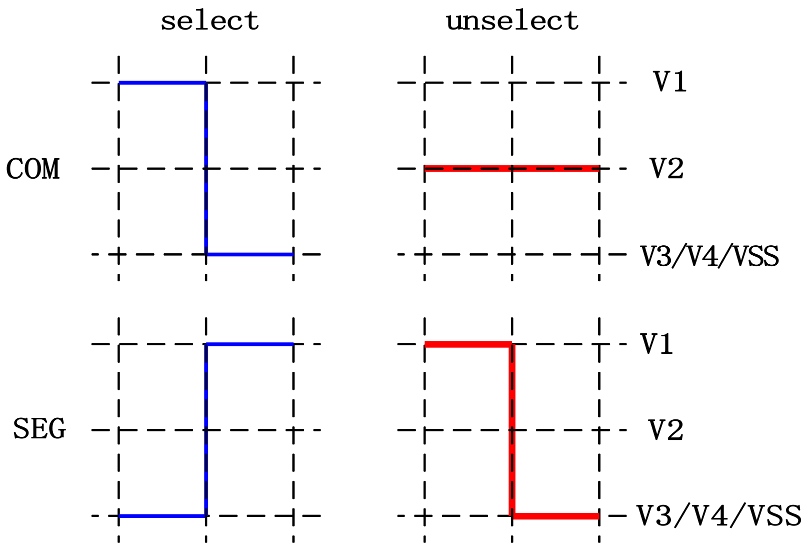


Figure 18-1-2-1 LCD bias voltage 1/2

- **LCD bias 1/3**

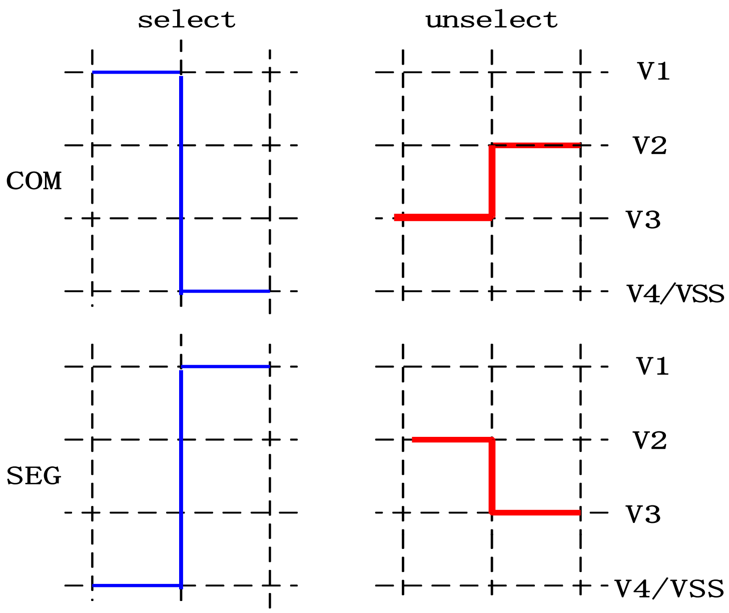


Figure 18-1-2-2 LCD bias voltage 1/3

- **LCD bias voltage 1/4**

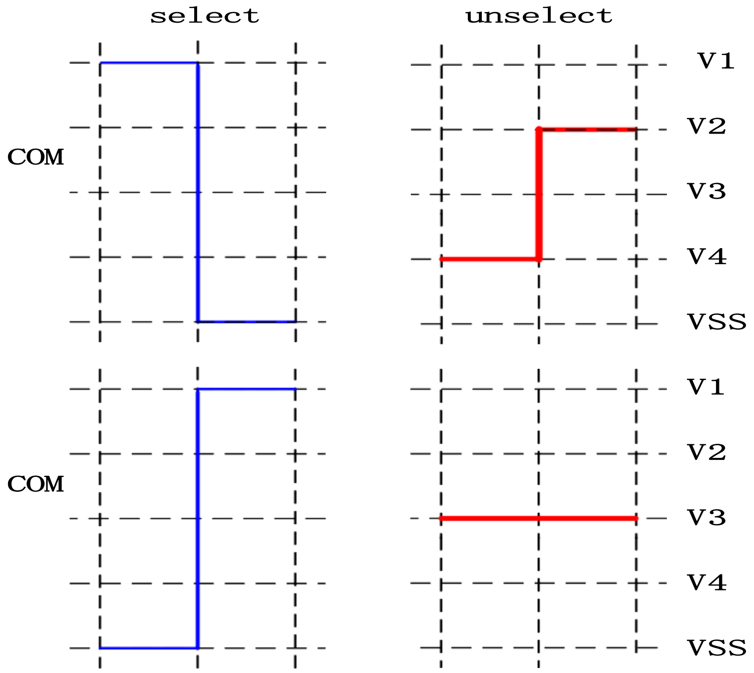


Figure 18-1-2-3 LCD bias voltage 1/4

18.1.3 LCD Function Description

LCD mode can be selected by setting LMOD=1. The clock source for LCD driver can be selected by LEN. When the clock source is selected, LCD driver will be enabled simultaneously. The selected clock must be enabled and operates normally before the selection. Register LXDIV is the frequency divider for LCD clock and can be used to set different ratio for different clock sources. The typical frequency for LCD frame scanning is 64Hz. In addition, there are 8 levels for LCD drive strength which are set by LDRV. The corresponding output voltages for different levels also vary greatly which can be modified according to the LCD monitor. Similarly, there are also 4 levels for drive current which can be set by DMOD to meet different power consumption requirements. The power consumption will decrease with smaller drive current while the noise in the pin will be greater at the same time as well.

The duty cycle of LCD programmable is 1/2, 1/3, 1/4, 1/5. The duty cycle is determined by the number of enabled COMs, for example, if 3 COMs are enabled, the duty cycle is 1/3, if 5 COMs are enabled, the duty cycle is 1/5. The enabling of COMs does not need to be done in the order of COM pin numbers, it can be done in any combination, but the enabled COM pins correspond to the actual COM0, COM1, COM2, etc. by serial number. But the enabled COM pins correspond to the actual COM0, COM1, COM2... from the smallest to the largest. The SEG pins can also be enabled in any combination, but the enabled SEG pins correspond to the actual SEG0, SEG1, SEG2... from smallest to largest by serial number. For example, enable pins SEG3, SEG5, SEG7... SEG3 corresponds to the actual SEG0, SEG5 corresponds to the actual SEG1, SEG7 corresponds to the actual SEG2.... COM and SEG pins that are not enabled can be set for other functions, and there is no conflict with the LCD driver.

The LCD driver has 10 bytes of LCD display cache, which corresponds to the actual COM and SEG. 10

bytes of display cache correspond to SEG0~SEG9 in order, while COM0~COM4 correspond to bits 0~4 of each byte. The display cache is accessed through the index register INDEX and the data register LXDAT, and setting INDEX to 0~9 corresponds to the display cache of SEG0~SEG9 in order. display can be visited by index register INDEX and data register LXDAT. INDEX=0~9 correspond to SEG0~SEG9 display buffer respectively.

18.2 LED Driver

18.2.1 Function Introduction

The built-in LED driver supports up to 5com x 8seg, 4comx 9seg, 3comx10seg, sharing display cache and driver pins with LCD driver. 8 levels of brightness adjustment for LED driver. Built-in global blinking function, blinking at a frequency of 1Hz when the scan frequency is 256Hz, LED COM can be set to high perfusion current mode (see IO chapter for details), LED for common cathode connection can achieve high brightness display effect.

Figure 18-2-1-1 LED Schematic

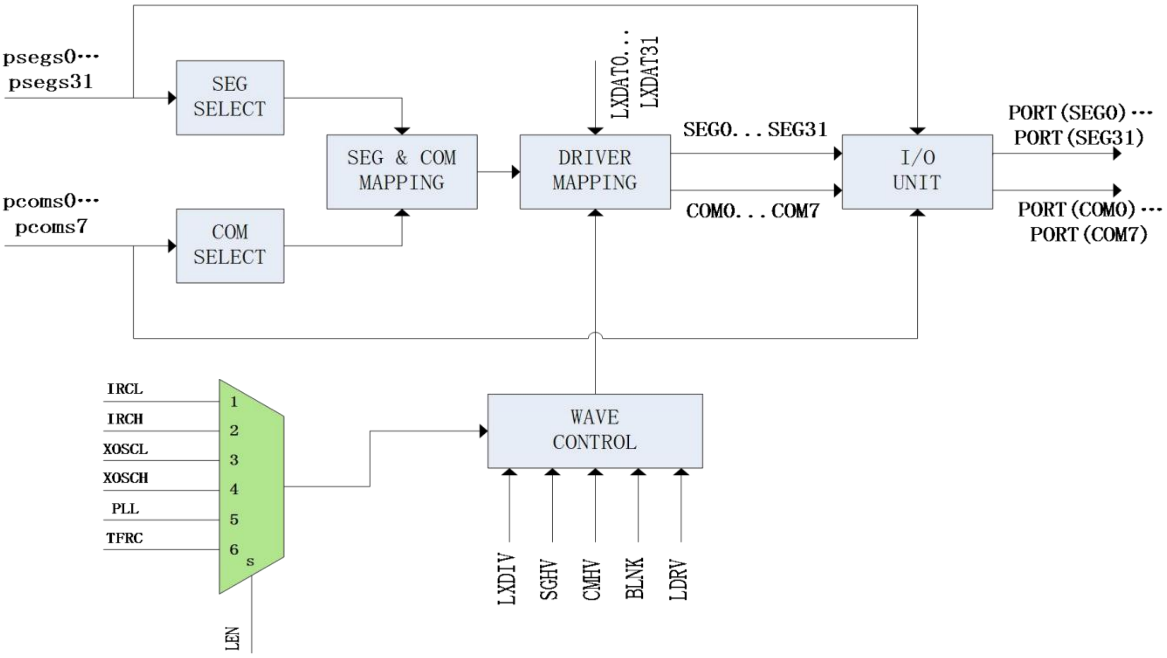


Figure 18-2-1-2 shows the driving waveform for LED.

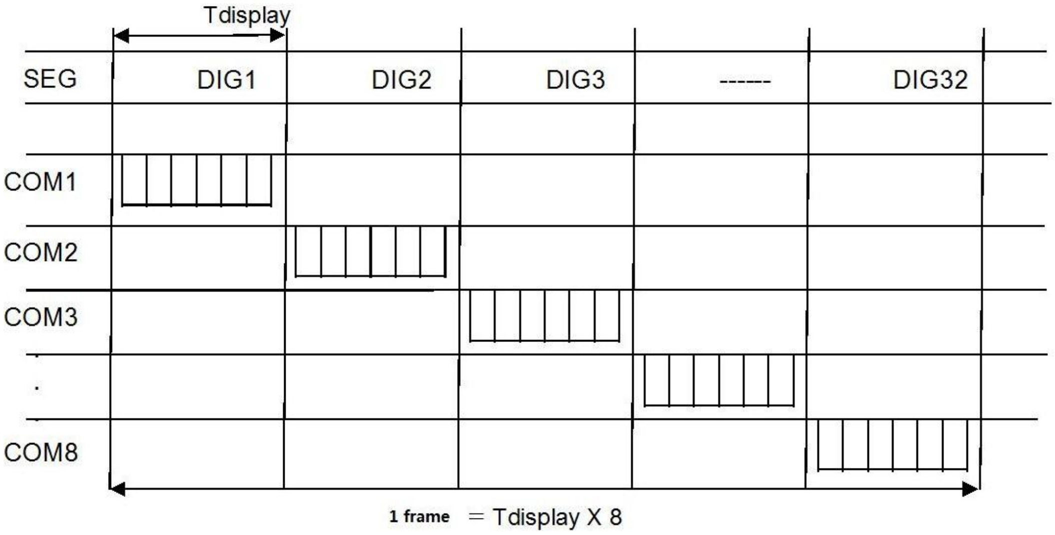


Figure 18-2-1-2 LED Driving Waveform

18.2.2 LED Function Description

LED mode can be selected by setting LMOD=0. LED driver selects the clock source by LEN and will be enabled after the selection. The selected clock must be enabled and operates normally before the selection. Register LXDIV is the frequency divider for LED clock and can be used to set different ratio for different clock sources. The typical frequency for LED frame scanning is 256Hz. In addition, there are 8 brightness levels for LED driver which are set by LDRV. The duty cycle varies according to the level as well.

The basic duty cycle for LED driver is also decided by the number of COM enabled. Any of the COM pins and SEG pins can be enabled, similar to LCD. For more information you may also refer to the related description for LCD.

LED driver and LCD driver share the common display buffer. For buffer visiting and other pins/ports relationship you may also refer to the description for LCD driver.

18.3 LCD/LED Register Description

Table 18-3-1 Register LXCON

E1H	7	6	5	4	3	2	1	0
LXCON	LEN[2:0]			LMOD	-	-	-	-
R/W	R/W			R/W	-	-	-	-
Initial Value	0	0	0	0	-	-	-	-

Bit number	Bit Symbol	Description
7~5	LEN	LCD/LED clock selection
		001：IRCL 010：IRCH 011：XOSCL 100：ERC 101：PLL 110：TFRC Others: module disabled
4	LMOD	Mode selection 0：LCD mode 1： LED mode
3~0	-	-

Table 18-3-2 Register LXCFG

E2H	7	6	5	4	3	2	1	0
LCD mode								
LXCFG	DMOD[1:0]		BIAS[1:0]		-	LDRV[2:0]		
R/W	R/W		R/W		-	R/W		
Initial Value	0	0	0	0	-	0	0	0
LED mode								
LXCFG	-	-	COMHV	SEGHV	BLNK	LDRV[2:0]		
R/W	-	-	R/W	R/W	R/W	R/W		
Initial Value	-	-	0	0	0	0	0	0

Bit number	Bit Symbol	Description
LCD mode		
7~6	DMOD	LCD driver current selection 00：5uA 01：40uA 10：80uA 11：130uA
5~4	BIAS	LCD bias voltage selection 01：1/2 Bias 10：1/3 Bias Others：1/4 Bias
3	-	-
2~0	LDRV	LCD driver magnitude selection 000：Level 1(minimum) 001：Level 2 ... 111：Level 8(maximum)
LED mode		
7~6	-	-
5	COMHV	When it is 0/1, indicates COM valid output is 0/1
4	SEGHV	When it is 0/1, indicates SEG valid output is 0/1

3	BLNK	LED global blink control with 1 is valid Note： The blink time is the 128 frame LED’s output. For instance, if the clock for LED operation is 32.768KHz with LXDIV=0, the LED output frequency is 256Hz, then the blink frequency is1Hz.
2~0	LDRV	LED brightness selection 000： Level 1(darkest) 001： Level 2 010： Level 3 011： Level 4 100： Level 5 101： Level 6 110： Level 7 111： Level 8(brightest)

Table 18-3-3 Register LXDIV

E4H	7	6	5	4	3	2	1	0
LXDIVL	LXDIV[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
E5H	7	6	5	4	3	2	1	0
LXDIVH	-	-	-	-	LXDIV[11:8]			
R/W	-	-	-	-	R/W			
Initial Value	-	-	-	-	0	0	0	0
Bit number	Bit Symbol		Description					
15~12	-		-					
11~0	LXDIV		LXD clock frequency divider LCD frame scanning frequency=LXD clock frequency ÷((LXDIV+1) x 512) LED frame scanning frequency when high speed clock is selected=LXD clock frequency ÷((LXDIV+1) x 1024) LED frame scanning frequency when low speed clock is selected=LXD clock frequency ÷((LXDIV+1) x 128) Note： 1. The high speed clocks for LED are IRCH/ERC/PLL/TFRC； The low speed clock for LED are IRCL/XOSCL					
			2. When IRCL is selected for LCD/LED, the frequency will be 1/4 of IRCL . 3.The typical frame scanning frequency for LCD 64Hz, it is 256Hz for LED.					

Table 18-3-4 Register LXDAT

E3H	7	6	5	4	3	2	1	0
LXDAT	LXDAT[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Note：LXDAT is register with index, when INDEX=0~9, it indicates LXDAT0~LXDAT9 respectively								
Bit number	Bit Symbol	Description						
7~0	LXDAT	Display buffer R/W register						

Table 18-3-5 Register LXCAD

8117H	7	6	5	4	3	2	1	0
LXCAD	-	-	-	-	CAD_MOD[1:0]		CAD_LTH[1:0]	
R/W	-	-	-	-	R/W		R/W	
Initial Value	-	-	-	-	0	1	0	1
Bit numbe	Bit Symbo	Description						
7~4	-	-						
3~2	CAD_MOD	LCD_CAD mode selection 00： Turn off LCD_CAD 01： LCD_CAD length is determined by the digital signal length Others: LCD_CAD length controlled by analog signal Note: In low-power mode, you can choose to turn off LCD_CAD, the current can be reduced by about 5~10uA .						
1~0	CAD_LTH	Analog signal to control LCD_CAD length selection, valid only when CAD_MOD=2/3 00：4us 01：8us 10：12us 11：12us						

Table 18-3-6 LCD/LED Display Buffer

INDEX	SEG	COM0	COM1	COM2	COM3	COM4
0	0	BIT 0	BIT 1	BIT 2	BIT 3	BIT 4
1	1	BIT 0	BIT 1	BIT 2	BIT 3	BIT 4
2	2	BIT 0	BIT 1	BIT 2	BIT 3	BIT 4
3	3	BIT 0	BIT 1	BIT 2	BIT 3	BIT 4
4	4	BIT 0	BIT 1	BIT 2	BIT 3	BIT 4
5	5	BIT 0	BIT 1	BIT 2	BIT 3	BIT 4
6	6	BIT 0	BIT 1	BIT 2	BIT 3	BIT 4
7	7	BIT 0	BIT 1	BIT 2	BIT 3	BIT 4
8	8	BIT 0	BIT 1	BIT 2	BIT 3	-
9	9	BIT 0	BIT 1	BIT 2	-	-

18.4 LCD Driver Control Example

For instance, to drive LCD with 5com×8seg, 1/4bias, the program is like：

```
------------------------------------------------------------------------------------------
#define XLCKE               (1<<3)
#define XLSTA               (1<<2)

#define LEN_XOSCL                (3<<5)
#define LMOD_lcd            (0<<4)
#define DMOD_5ua            (0<<6)
#define BIAS_1_4            (0<<4)
#define LCDRV_LEV(N)             (N) //N=0~7
void LCD_init(void)
{
    unsigned char i;
    P00F = 3; //set P00 as COM0
    P01F = 3; //set P01 as COM1
    P02F = 3; //set P02 as COM2
    P03F = 3; //set P03 as COM3
    P04F = 3; //set P04 as COM4
    P57F = 3; //set P57 as SEG0
    P34F = 3; //set P34 as SEG1
    P35F = 3; //set P35 as SEG2
    P56F = 3; //set P56 as SEG3
    P50F = 3; //set P50 as SEG4
    P51F = 3; //set P51 as SEG5
    P52F = 3; //set P52 as SEG6
    P53F = 3; //set P53 as SEG7

    CKCON |= XLCKE;//enable XOSCL clock
    while(!(CKCON & XLSTA)); //wait until XOSCL clock becomes stable
    LXDIVH = 0;              //set the clock frequency division, the frame frequency of the LCD after division is 64Hz
    LXDIVL = 0;
    LXCFG = DMOD_5ua | BIAS_1_4 | LCDRV_LEV(7); //set the LCD driving current and magnitude, bias
    LXCON =  LEN_XOSCL | LMOD_lcd;         //set the clock source for LCD as XOSCL、set is as LCD mode

    for(i=0;i<8;i++) //write LCD display buffer
    {
        INDEX = i;          //set the index for display buffer
        LXDAT = 0;          //write buffer, clear the screen when 0 is written
    }
}
------------------------------------------------------------------------------------------
```

18.5 LED Driver Control Example

For instance, to drive LED with 5com×8seg, with common cathode, the program is like：

```
-----------------------------------------------------------------------------------------
#define XLCKE           (1<<3)
#define XLSTA           (1<<2)

#define LEN_XOSCL           (3<<5)
#define LMOD_led        (1<<4)
#define CMHV(N)             (N<<     //N=0~
                            5)             1
#define SGHV(N)           (N<<4)     //N=0~
                                           1
#define BLNK(N)           (N<<3)     //N=0~
                                           1
void LED_init(void)
{
    unsigned char i;
    P00F = 3; //set P00 as COM0
    P01F = 3; //set P01 as COM1
    P02F = 3; //set P02 as COM2
    P03F = 3; //set P03 as COM3
    P04F = 3; //set P04 as COM4
    P57F = 3; //set P57 as SEG0
    P34F = 3; //set P34 as SEG1
    P35F = 3; //set P35 as SEG2
    P56F = 3; //set P56 as SEG3
    P50F = 3; //set P50 as SEG4
    P51F = 3; //set P51 as SEG5
    P52F = 3; //set P52 as SEG6
    P53F = 3; //set P53 as SEG7
    CKCON |= XLCKE;//enable XOSCL clock
    while(!(CKCON & XLSTA)); //wait until XOSCL clock is stable
    LXDIVH = 0;             //set clock frequency division, LED frame frequency is 256Hz after the division
    LXDIVL = 0;
    LXCFG = CMHV(0) | SGHV(1) | BLNK(0) | LEDRV_LEV(7);      //set low level valid for COM, high level valid
for SEG, disable the global blink, set the brightness to maximum
    LXCON = LEN_XOSCL | LMOD_led; //set the clock source for LED as XOSCL 、 set as LED mode
    for(i=0;i<8;i++) //write the LED display buffer
    {
        INDEX = i;          //set the index for display buffer
        LXDAT = 0;          //write buffer, clear the screen when 0 is written
    }
}
-----------------------------------------------------------------------------------------
```

19 PWM

19.1 PWM Function Introduction

CA51F3 series chip can include at most 6 channels PWM outputs. PWM period and duty cycle can be configured with 16 bit range. There is center fixed and edge fixed mode for each PWM. In addition, PWM also supports the dead time control and complementary output. There are 3 pairs of complementary channels consisting of 6 PWM in complementary mode. This is designed for Brushless DC motor driver.

19.2 PWM Function Description

There is a 16-bit counter for each PWM channel and the cycle is set by register PWMDIV. Register PWMDUT sets the corresponding PWM's duty cycle. PWM is enabled by register PWMEN with each bit of it corresponds to one channel in PWM. There is a data refresh register PWMUPD for PWM module. When register PWMDIV, PWMDUT and PWMCKD is overwritten, corresponding bit of the register PWMUPD must be set to 0 and then the data can be refreshed. PWM pin can also output reversed phase by setting PWMTOG. There are multiple clock sources for PWM which is set for PWM pairs (PWM0 and PWM1, PWM2 and PWM3, PWM4，PWM5). In other words, the clock source is the same when the PWM"s are in the same pair. The clock source can be selected by corresponding PWMCKS of register PWMCON for PWM0、PWM2、PWM4. with the frequency division set by PWMCKD independently.

- **Edge fixed mode and center fixed mode**

The edge/center fixed mode for PWM is selected by PWMMS. PWM counter starts to count from 0 after PWM is enabled. When the count valuer is less than PWMDUT, PWM pin outputs high level signal (PWMTOG=0). When the count equals or is greater than PWMDUT, PWM pin outputs low level signal (PWMTOG=0).

In edge fixed mode, when the count equals PWMDIV, a PWM cycle completes and the PWM counter is cleared to start counting again. While in center fixed mode, when the count equals PWMDIV, the direction of counting reverses and starts down counting. When it is down counting and the count is less than PWMDUT, PWM pin outputs high level signal(PWMTOG=0). When the count is greater than or equals PWMDUT, PWM pin outputs low level signal(PWMTOG=0). When the count goes down to 0, a PWM cycle completes and the counter starts up counting again for another cycle.

The single PWM output waveforms in edge/center fixed mode are shown as Figure 19-2-1 and Figure 19-2-2(note： for all the waveform blow, PWMDIV>PWMDUT>0). As the figure shows, with the same PWMDIV and PWMDUT, the PWM cycle for center fixed mode is 2 times longer than it for edge fixed mode.

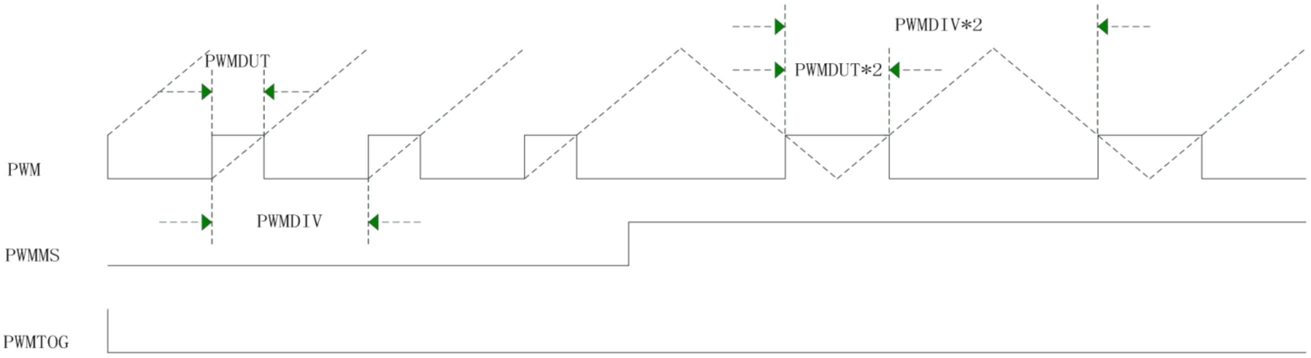


Figure 19-2-1 PWM Output Waveform when PWMTOG = 0

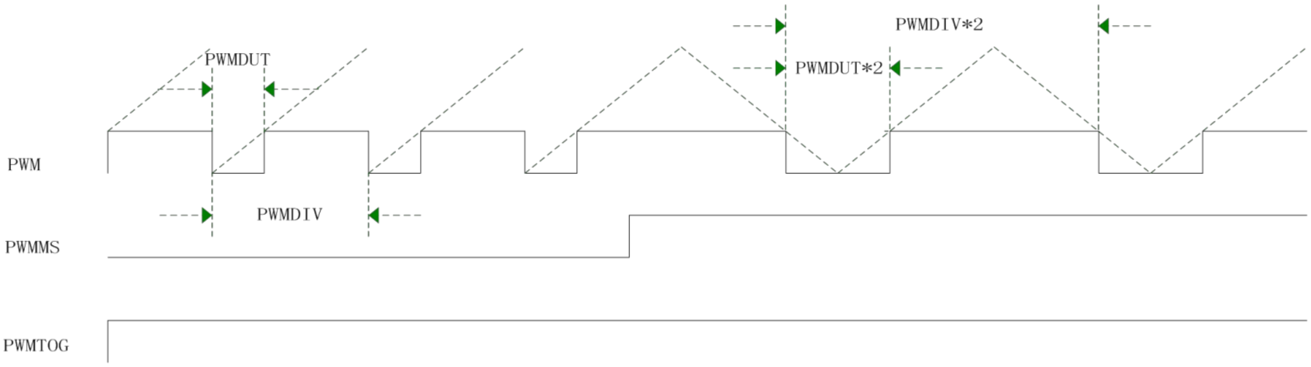


Figure 19-2-2 PWM Output Waveform when PWMTOG = 1

It is worth noting that When PWMDIV=0, PWM pin output the PWM clock directly. If PWMCKD=0, PWM pin's output is the selected clock source. When PWMDIV is not 0 but PWMDUT=0, PWM pin outputs low level signal (PWMTOG＝0)；while PWMDUT>=PWMDIV>0, PWM pin outputs high level signal(PWMTOG＝0)。

- **Complementary mode**

6 PWM can forms 3 pair complementary channels in this complementary mode：PWM0 and PWM1, PWM2 and PWM3, PWM4 and PWM5. The complementary mode of PWM is set by the PWMMOD bit of the control register PWMCON of PWM1, PWM3, PWM5. In this complementary mode, PWM fixed mode, its cycle and duty cycle, the clock frequency division for it, are all set by the corresponding register for PWM0, PWM2, PWM4. Only PWMTOG is still controlled by the channel's corresponding register independently. The Figure 19-2-3 shows the principle of this mode.

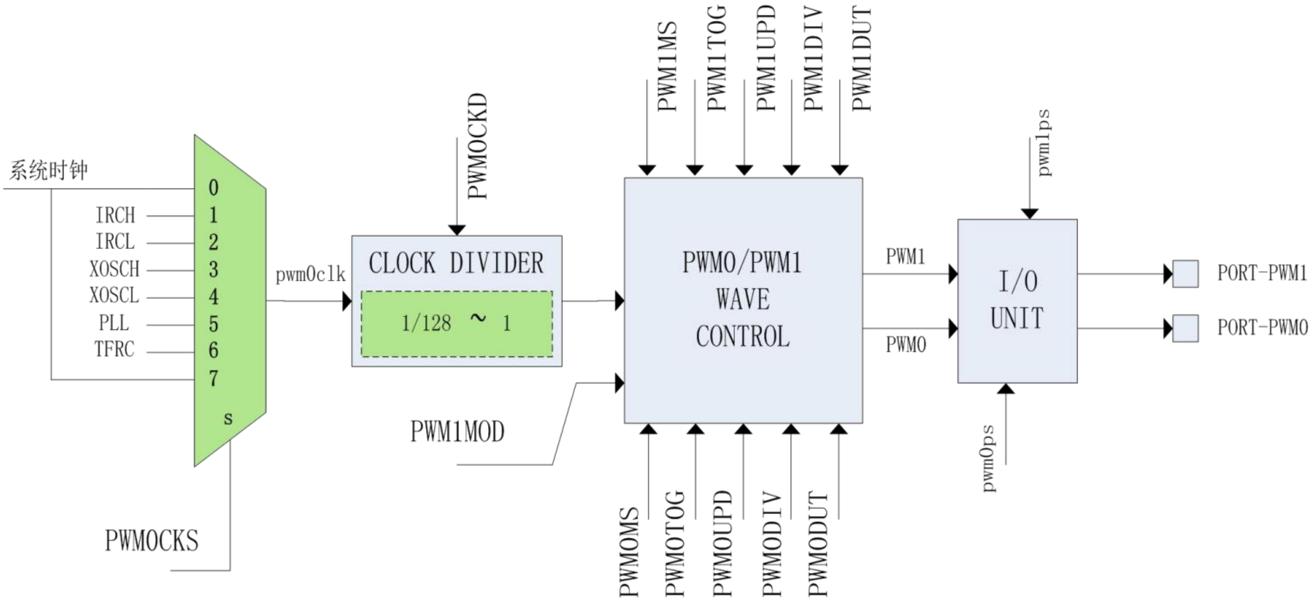


Figure 19-2-3 Schematic for PWM0 and PWM1

The phases are complementary for each pair as Figure 19-2-4 and Figure 19-2-5 shows (PWM0 and PWM1 for example).

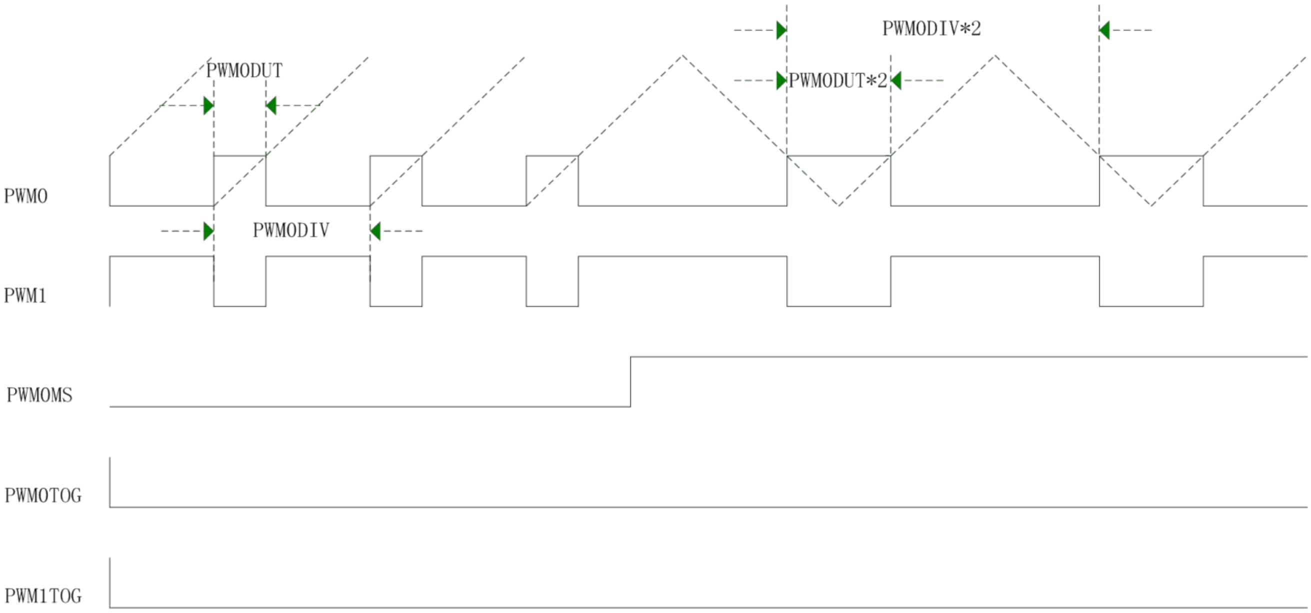


Figure 19-2-4 PWM0 and PWM1 Complementary Output Waveform when PWMTOG = 0

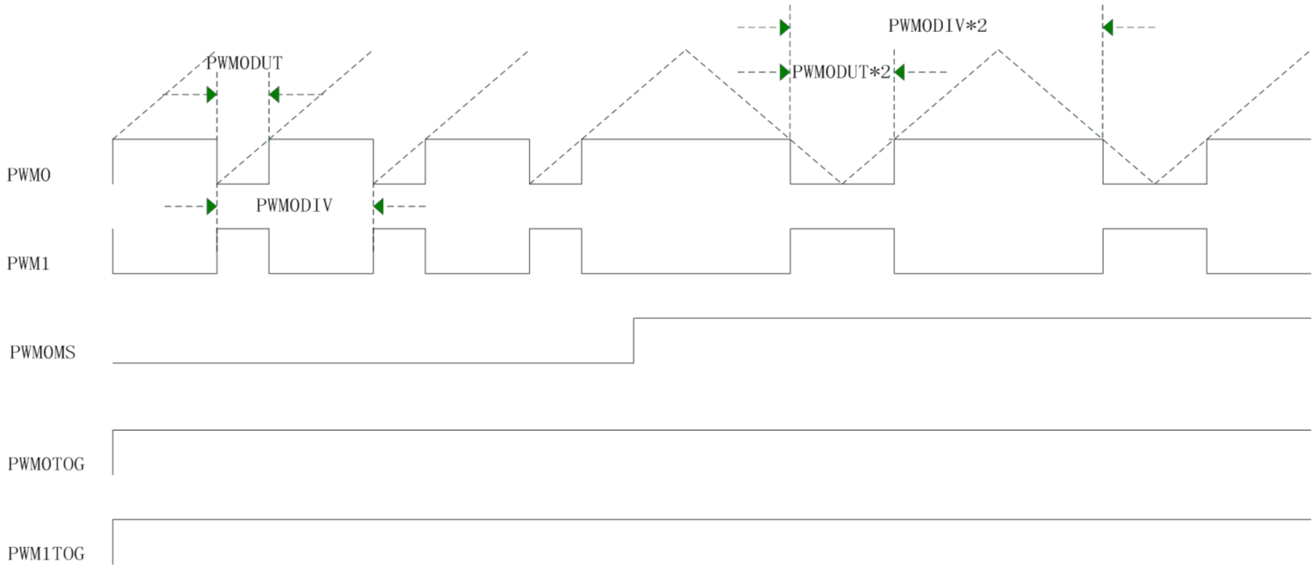


Figure 19-2-5 PWM0 and PWM1 Complementary Output Waveform when PWMTOG＝1

- **Deadtime control**

In the bridge drive circuit, in order to prevent the upper and lower half bridges from being turned on at the same time, it is necessary to insert a dead zone control in the PWM complementary signal. The deadtime can be controlled by PWM1, PWM3 and PWM5's corresponding register PWMDIV and PWMDUT. PWMDIV sets the deadtime on the left and PWMDUT sets the deadtime on the right. The deadtime must meet the requirements below (take PWM0 and PWM1, for example)：

In edge fixed mode, PWMDIV1<PWMDUT0 and PWMDUT1<(PWMDIV0 – PWMDUT0);

In center fixed mode, PWMDIV1< (PWMDIV0 – PWMDUT0) * 2 or PWMDUT1<(PWMDIV0 – PWMDUT0) * 2.

Figure 21-1-6 shows the output waveform for deadtime control (taking PWM0, PWM1 for example).

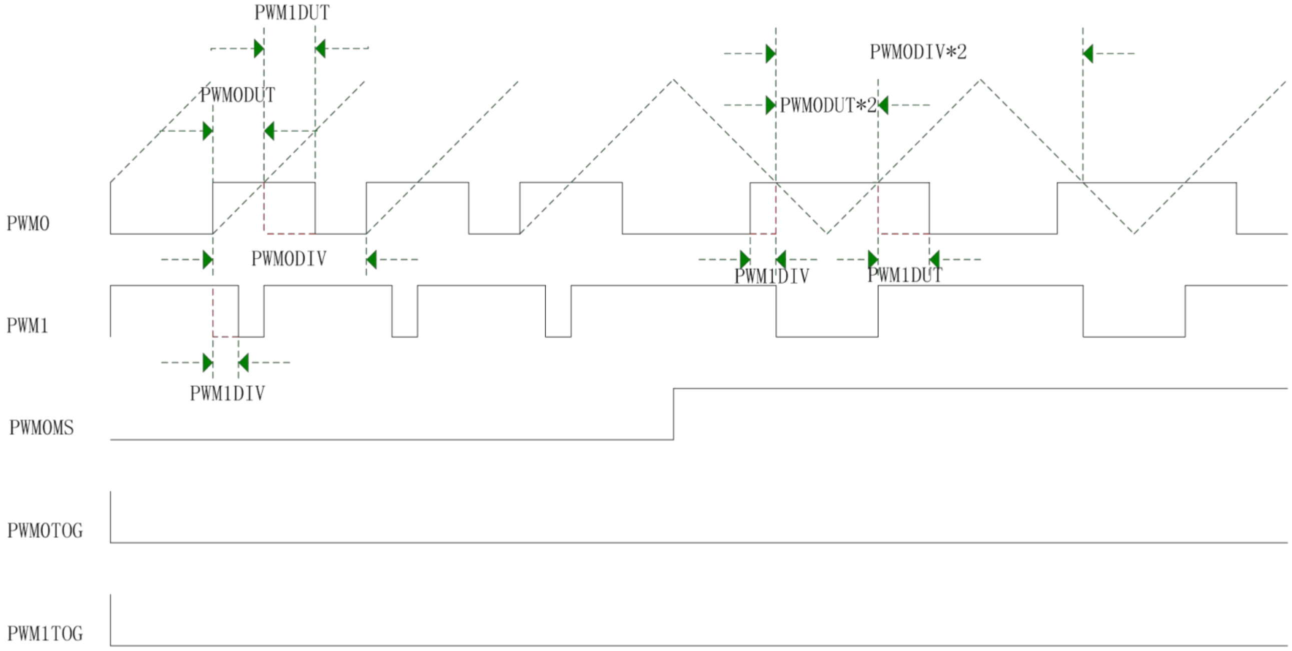


Figure 19-2-6 PWM0 and PWM1 Deadtime Control Waveform when PWMTOG＝0

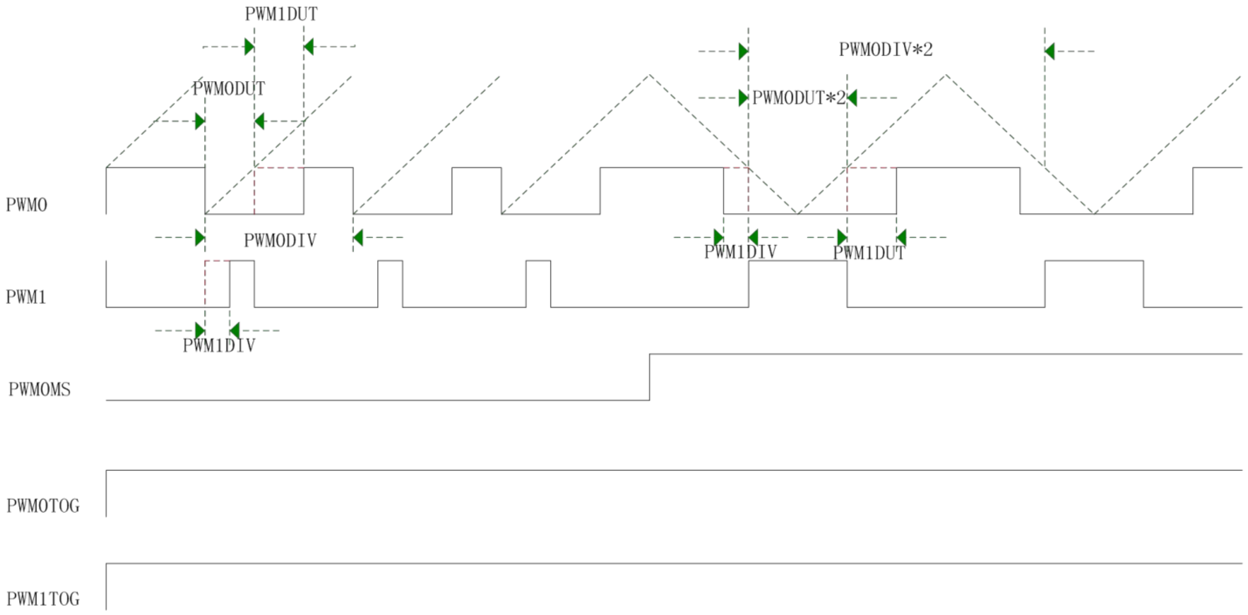


Figure 19-2-7 PWM0 and PWM1 Deadtime Control Waveform when PWMTOG＝1

- **PWM Interrupt**

PWM interrupt in enabled by PWMTIE, PWMZIE, PWMPIE and PWMNIE in register PWMCON. PWMTIE enables the interrupt when PWM counter's count reaches the peak (which equals PWMDIV). PWMZIE enables the interrupt when PWM counter's count reaches the bottom(which equals 0). PWMNIE enables the interrupt when the output pin falling edge comes. PWMPIE enables the interrupt when the output pin rising edge comes. PWMTIE and PWMZIE's corresponding interrupts do not exist in edge fixed mode. Register PWMAIF、 PWMBIF、 PWMCIF are the interrupt status registers for the 6 channel's interrupts. PWMxTIF、PWMxZIF、 PWMxNIF、 PWMxPIF correspond to PWMTIE, PWMZIE, PWMNIE, PWMPIE respectively.

In addition, PWM can set how many interrupt events occur to generate an interrupt through the register PWMCMX For instance, if PWMCMX=3 and PWMPIE=1, then there will be a rising edge interrupt when PWM pin rising edge has come 4 times.

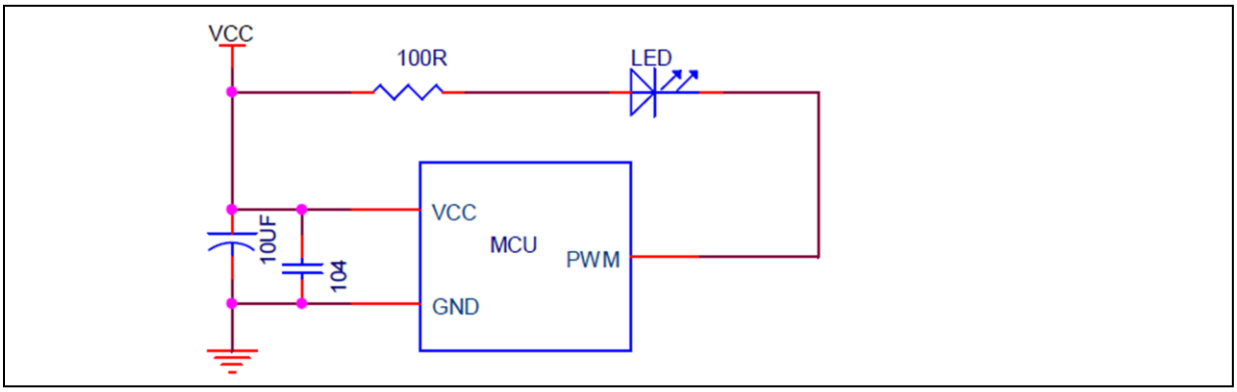


Figure 19-2-8 PWM drive LED lamp reference circuit
(the above design circuit and component parameters are for reference only)

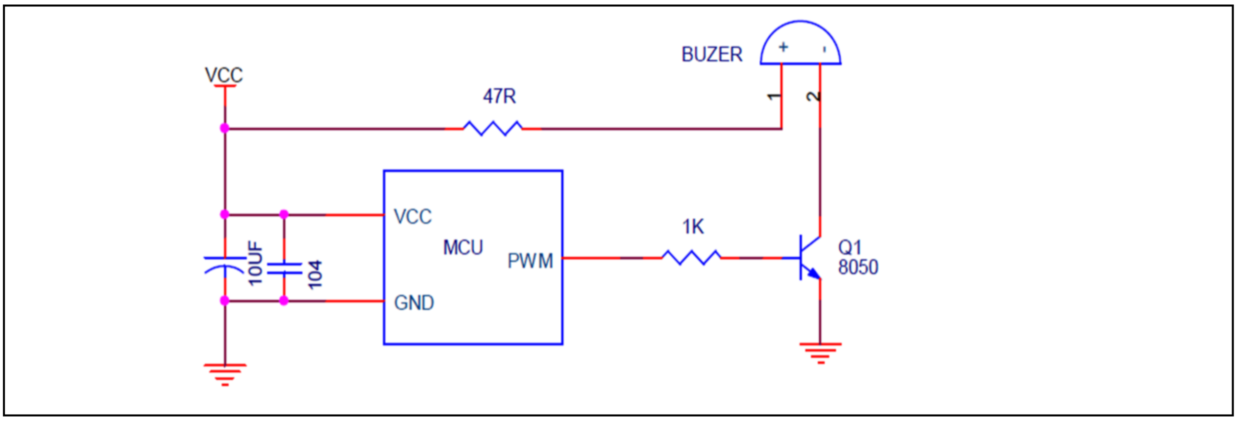


Figure 19-2-9 PWM driven DC buzzer reference circuit
(the above design circuit and component parameters are for reference only)

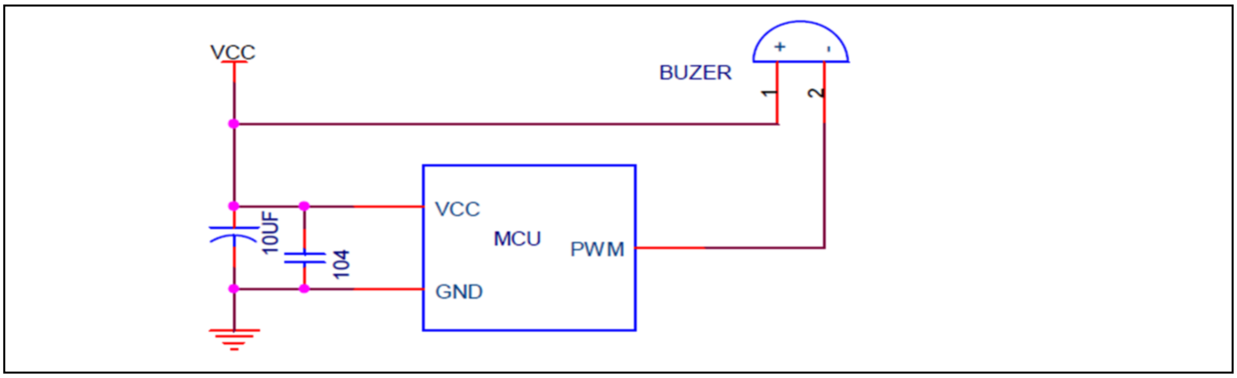
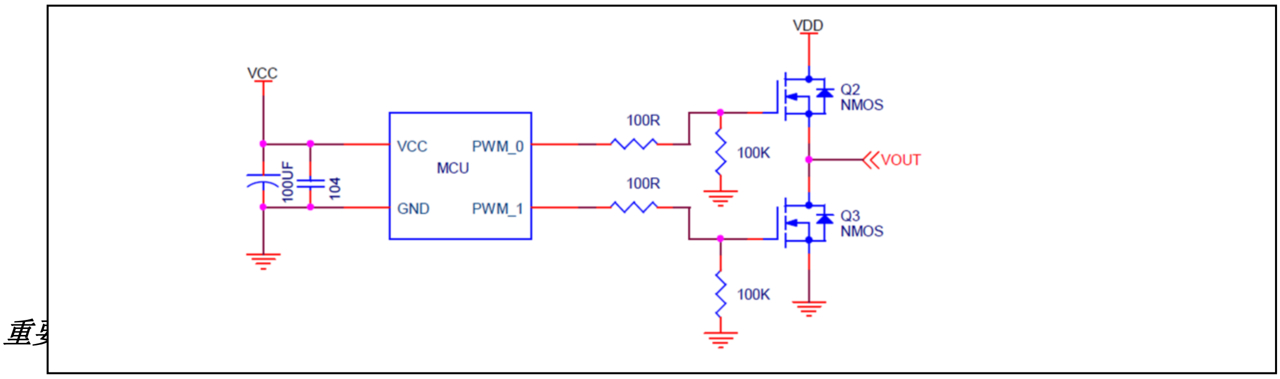


Figure 19-2-9 PWM driven AC buzzer reference circuit
(the above design circuit and component parameters are for reference only)



重要

Figure 19-2-9 PWM complementary reference circuit
(the above design circuit and component parameters are for reference only)

19.3 PWM Register Description

Table 19-3-1 Register PWMEN

D8H	7	6	5	4	3	2	1	0
PWMEN	-	-	PWMEN[5:0]					
R/W	-	-	R/W					
Initial Value	0	0	0	0	0	0	0	0
Bit number		Bit Symbol	Description					
5~0		PWMEN	5~0 bit correspond to PWM channel 5~0's enable control, 1 enables it					

Table 19-3-2 Register PWMUPD

D9H	7	6	5	4	3	2	1	0
PWMUPD	-	-	PWMUPD[5:0]					
R/W	-	-	R/W					
Initial Value	-	-	0	0	0	0	0	0
Bit number		Bit Symbol	Description					
5~0		PWMUPD	5~0 bit correspond to PWM channel 5~0's data refresh enable control, 1 enables it *Note：* *After configuring the data of a certain channel (PWMDIV/PWMDUT/PWMCKD), set the bit of the corresponding channel of PWMUPD to 1, so that the data will be updated after the PWM counter overflows, and the corresponding bit will be automatically cleared to 0 after the data update is completed.*					

Table 19-3-3-3 Register PWMCMX

DAH	7	6	5	4	3	2	1	0
PWMCMX	PWMCMX[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Note：PWMCMX is register with index, INDEX=0~5 corresponds to PWMCMX0~PWMCMX5								
Bit number	Bit Symbol		Description					
7~0	PWMCMX		PWM interrupt events number for each channel Interrupt event number=PWMCMX+1, For instance, when INDEX=0and, PWMCMX=7, then only when the interrupt trigger events happens 8 times the interrupt flag will be set to 1.					

Table 19-3-4 Register PWMCON

DBH	7	6	5	4	3	2	1	0
PWMCON①	PWMTIE	PWMZIE	PWMPIE	PWMNIE	PWMMS	PWMCKS[2:0]		
R/W	R/W	R/W	R/W	R/W	R/W	R/W		
Initial Value	0	0	0	0	0	0	0	0
PWMCON②	PWMTIE	PWMZIE	PWMPIE	PWMNIE	PWMMS	-	-	PWMMOD

R/W	R/W	R/W	R/W	R/W	R/W	-	-	R/W
Initial Value	0	0	0	0	0	-	-	0
Bit number	Bit Symbol	Description						
Note： *PWMCON①is the channel control for PWM0/PWM2/PWM4* *PWMCON②is the channel control for PWM1/PWM3/PWM5* *PWMCON is also register with index, INDEX=0~5 correspond to PWMCON0~PWMCON5 respectively*								
7	PWMTIE	PWM counter peak interrupt enable control, 1 enables it						
6	PWMZIE	PWM counter bottom interrupt enable control, 1 enables it						
5	PWMPIE	PWM rising edge interrupt enable control, 1 enables it						
4	PWMNIE	PWM falling edge interrupt enable control, 1 enables it						
3	PWMMS	PWM mode selection 0：edge fixed mode 1:center fixed mode						
2~0	PWMCKS	PWM working clock selection bit 001：IRCH 010：IRCL 011：ERC 100：XOSCL 101：PLL 110：TFRC Others：system clock *Note：* *PWM0/PWM1 is set by PWMCKS0；* *PWM2/PWM3 is set by PWMCKS2；* *PWM4/PWM5 is set by PWMCKS4；*						
0	PWMMOD	Complementary mode enable control, 1 enables it *Note：* *When PWMMOD1=1, PWM0 and PWM1 enter the* complementary mode *When PWMMOD3=1, PWM2 and PWM3 enter the* complementary mode *When PWMMOD5=1, PWM4 and PWM5 enter the* complementary mode						

Table 19-3-5 Register PWMCFG

DCH	7	6	5	4	3	2	1	0
PWMCFG	PWMTOG	PWMCKD[6:0]						
R/W	R/W	R/W						
Initial Value	0	0	0	0	0	0	0	0
Note：PWMCFG is register with index, INDEX=0~5 corresponds to PWMCFG0~PWMCFG5								
Bit number	Bit Symbol	Description						
7	PWMTOG	PWM phase reversion enable, 1 inverts the phase						
6~0	PWMCKD	PWM working clock frequency division setting 0000000：no division 0000001：frequency divided by 2 0000010：frequency divided by 3 ...... 1111110：frequency divided by 127 1111111：frequency divided by 128						

Table 19-3-6 Register PWMDIVL、PWMDIVH

DDH	7	6	5	4	3	2	1	0
PWMDIVL	PWMDIV[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
DEH	7	6	5	4	3	2	1	0
PWMDIVH	PWMDIV[15:8]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Note：PWMDIV is register with index, INDEX=0~5 corresponds to PWMDIV0~PWMDIV5								
Bit number	Bit Symbol	Description						
15~0	PWMDIV	PWM cycle configuration PWMDIV1/PWMDIV3/PWMDIV5 are for different use in complementary mode, please refer to register PWMDUT description						

Table 19-3-7 Register PWMDUTL, PWMDUTH

<table>
<tr><td>DFH</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>PWMDUTL</td><td colspan="8">PWMDUT[7:0]</td></tr>
<tr><td>R/W</td><td colspan="8">R/W</td></tr>
<tr><td>Initial Value</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>D1H</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>PWMDUTH</td><td colspan="8">PWMDUT[15:8]</td></tr>
<tr><td>R/W</td><td colspan="8">R/W</td></tr>
<tr><td>Initial Value</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td colspan="9"><i>Note：PWMDUT is register with index, INDEX=0~5 corresponds to PWMDUT0~PWMDUT5</i></td></tr>
<tr><td>Bit number</td><td colspan="2">Bit Symbol</td><td colspan="6">Description</td></tr>
<tr><td>15~0</td><td colspan="2">PWMDUT</td><td colspan="6">PWM Duty Cycle Configuration Register
In complementary mode, PWMDUT1/PWMDUT3/PWMDUT5 have different meanings, as shown in the following table:
<table>
<tr><td>PWMDIV1</td><td>Control the width of the dead zone on the left of PWM0/PWM1</td></tr>
<tr><td>PWMDUT1</td><td>Control the width of the dead zone on the right side of PWM0/PWM1</td></tr>
<tr><td>PWMDIV3</td><td>Control the width of the dead zone on the left side of PWM2/PWM3</td></tr>
<tr><td>PWMDUT3</td><td>Control the width of the dead zone on the right side of PWM2/PWM3</td></tr>
<tr><td>PWMDIV5</td><td>Control the width of the dead zone on the left of PWM4/PWM5</td></tr>
<tr><td>PWMDUT5</td><td>Control the width of the dead zone on the right side of PWM4/PWM5</td></tr>
</table></td></tr>
</table>

Table 19-3-8 Register PWMAIF

<table>
<tr><td>D2H</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>PWMAIF</td><td>PWM1TIF</td><td>PWM1ZIF</td><td>PWM1PIF</td><td>PWM1NIF</td><td>PWM0TIF</td><td>PWM0ZIF</td><td>PWM0PIF</td><td>PWM0NIF</td></tr>
<tr><td>R/W</td><td>R</td><td>R</td><td>R</td><td>R</td><td>R</td><td>R</td><td>R</td><td>R</td></tr>
<tr><td>Initial Value</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td colspan="9"></td></tr>
<tr><td>Bit number</td><td>Bit Symbol</td><td colspan="7">Description</td></tr>
<tr><td>7</td><td>PWM1TIF</td><td colspan="7">PWM1 counter vertex interrupt flag bit, write 1 to clear 0</td></tr>
<tr><td>6</td><td>PWM1ZIF</td><td colspan="7">PWM1 counter lowest point interrupt flag bit, write 1 to clear 0</td></tr>
<tr><td>5</td><td>PWM1PIF</td><td colspan="7">PWM1 rising edge interrupt flag bit, write 1 to clear 0</td></tr>
<tr><td>4</td><td>PWM1NIF</td><td colspan="7">PWM1 falling edge interrupt flag bit, write 1 to clear 0</td></tr>
<tr><td>3</td><td>PWM0TIF</td><td colspan="7">PWM0 counter vertex interrupt flag bit, write 1 to clear 0</td></tr>
<tr><td>2</td><td>PWM0ZIF</td><td colspan="7">PWM0 counter lowest point interrupt flag bit, write 1 to clear 0</td></tr>
<tr><td>1</td><td>PWM0PIF</td><td colspan="7">PWM0 rising edge interrupt flag bit, write 1 to clear 0</td></tr>
<tr><td>0</td><td>PWM0NIF</td><td colspan="7">PWM0 falling edge interrupt flag bit, write 1 to clear 0</td></tr>
</table>

Table 19-3-9 Register PWMBIF

D3H	7	6	5	4	3	2	1	0
PWMBIF	PWM3TIF	PWM3ZIF	PWM3PIF	PWM3NIF	PWM2TIF	PWM2ZIF	PWM2PIF	PWM2NIF
R/W	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
7	PWM3TIF	PWM3 counter vertex interrupt flag bit, write 1 to clear 0						
6	PWM3ZIF	PWM3 counter lowest point interrupt flag bit, write 1 to clear 0						
5	PWM3PIF	PWM3 rising edge interrupt flag bit, write 1 to clear 0						
4	PWM3NIF	PWM3 falling edge interrupt flag bit, write 1 to clear 0						
3	PWM2TIF	PWM2 counter vertex interrupt flag bit, write 1 to clear 0						
2	PWM2ZIF	PWM2 counter lowest point interrupt flag bit, write 1 to clear 0						
1	PWM2PIF	PWM2 rising edge interrupt flag bit, write 1 to clear 0						
0	PWM2NIF	PWM2 falling edge interrupt flag bit, write 1 to clear 0						

Table 19-3-10 Register PWMCIF

D4H	7	6	5	4	3	2	1	0
PWMCIF	PWM5TIF	PWM5ZIF	PWM5PIF	PWM5NIF	PWM4TIF	PWM4ZIF	PWM4PIF	PWM4NIF
R/W	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
7	PWM5TIF	PWM5 counter vertex interrupt flag bit, write 1 to clear 0						
6	PWM5ZIF	PWM5 counter lowest point interrupt flag bit, write 1 to clear 0						
5	PWM5PIF	PWM5 rising edge interrupt flag bit, write 1 to clear 0						
4	PWM5NIF	PWM5 falling edge interrupt flag bit, write 1 to clear 0						
3	PWM4TIF	PWM4 counter vertex interrupt flag bit, write 1 to clear 0						
2	PWM4ZIF	PWM4 counter lowest point interrupt flag bit, write 1 to clear 0						
1	PWM4PIF	PWM4 rising edge interrupt flag bit, write 1 to clear 0						
0	PWM4NIF	PWM4 falling edge interrupt flag bit, write 1 to clear 0						

19.4 PWM Control Example

- **Single channel PWM output**

For instance , PWM0 outputs 30KHz clock with duty cycle 30%, the program is like：

```
------------------------------------------------------------------------------------------
#define PWM_CH0         0

#define TIE(N)              (N<<7)     //N=0~1
#define ZIE(N)              (N<<6)     //N=0~1
#define PIE(N)              (N<<5)     //N=0~1
#define NIE(N)                    (N<<4)     //N=0~1
#define MS(N)               (N<<3)     //N=0~1

#define CKS_IH                    (1<<0)

#define TOG(N)                    (N<<7)     //N=0~1
void PWM_init(void)
{
      P15F = 6;                //set P15 as PWM pin
      INDEX = PWM_CH0;      //set INDEX to PWM0
      PWMCON = TIE(0) | ZIE(0) | PIE(0) | NIE(0) | MS(0) | CKS_IH ;//disable PWM0 interrupt, set PWM0  as edge
fixed mode, set PWM0 clock source as IRCH
      PWMCFG = TOG(0) | 0; //disable the phase reversion, no frequency division for the clock

      PWMDIVH     = 0;
      PWMDIVL     = 123;      //3686400/30000=123
      PWMDUTH     = 0;
      PWMDUTL     = 37;//123*0.3=37

      PWMUPD  ｜= (1<<PWM_CH0); //set PWM refresh
      while(PWMUPD); //wait until PWM setting refresh completes, necessary before enabling PWM
      PWMEN   ｜= (1<<PWM_CH0);     //enable PWM0
}
------------------------------------------------------------------------------------------
```

For instance , PWM0 outputs the IRCH clock directly, the program is like：

```
------------------------------------------------------------------------------------------
void PWM_init(void)
{
      P15F = 6;                //set P15 as PWM pin
      INDEX = PWM_CH0;      //set INDEX to PWM0
      PWMCON = TIE(0) | ZIE(0) | PIE(0) | NIE(0) | MS(0) | CKS_IH ;//disable PWM0 interrupt, set PWM0  as edge
fixed mode, set PWM0 clock source as IRCH
      PWMCFG = TOG(0) | 0; //disable the phase reversion, no frequency division for the clock
```

```
    PWMDIVH     = 0; //set PWMDIV and PWMDUT to 0 to output the clock source
    PWMDIVL     = 0;
    PWMDUTH     = 0;
    PWMDUTL     = 0;

    PWMUPD ｜= (1<<PWM_CH0); //set PWM refresh
    while(PWMUPD); //wait until PWM setting refresh completes, necessary before enabling PWM

    PWMEN  ｜= (1<<PWM_CH0);     /enable /PWM0
}
------------------------------------------------------------------------------------------------
```

◆ **PWM complementary output and deadtime control example**

Taking PWM0,PWM1 for instance, the 2 PWM output 30KHz complementary clock with 50% for duty cycle, 2 cycles deadtime is inserted at the same time, the program is like：

```
------------------------------------------------------------------------------------------------
#define PWM_CH0         0
#define PWM_CH1         1

#define TIE(N)          (N<<7)    //N=0~1
#define ZIE(N)          (N<<6)    //N=0~1
#define PIE(N)          (N<<5)    //N=0~1
#define NIE(N)          (N<<4)    //N=0~1
#define MS(N)           (N<<3)    //N=0~1

#define CKS_IH                (1<<0)

#define TOG(N)                (N<<7)    //N=0~1
#define MOD(N)                (N<<0)    //N=0~1

void PWM_init(void)
{
    P15F = 6;           //set P15 as PWM0 pin
    P16F = 6;           //set P15 as PWM1 pin

    INDEX = PWM_CH0;    //set INDEX to PWM0
    PWMCON = TIE(0) | ZIE(0) | PIE(0) | NIE(0) | MS(0) | CKS_IH ;//disable PWM0 interrupt, set PWM0  as edge
fixed mode, set PWM0 clock source as IRCH
    PWMCFG = TOG(0) | 0; //disable the phase reversion, no frequency division for the clock

    PWMDIVH     = 0;
    PWMDIVL     = 123;    //3686400/30000=123
    PWMDUTH     = 0;
    PWMDUTL     = 61;//123*0.5=61
```

```
        INDEX = PWM_CH1;     //set INDEX to PWM1
        PWMCON = TIE(0) | ZIE(0) | PIE(0) | NIE(0) | MS(0) | MOD(1) ;//set PWM0, PWM1 as complementary mode
        PWMCFG = TOG(0) | 0; //disable the phase reversion, no frequency division for the clock

        PWMDIVH     = 0;
        PWMDIVL     = 2; //insert 2 cycle period deadtime on the left, set it to 0 if no deadtime needed
        PWMDUTH     = 0;
        PWMDUTL     = 2; //insert 2 cycle period deadtime on the right, set it to 0 if no deadtime needed
        PWMUPD ｜= (1<<PWM_CH0)｜(1<<PWM_CH1); //set PWM refresh
        while(PWMUPD); //wait until PWM setting refresh completes, necessary before enabling PWM

        PWMEN  ｜= (1<<PWM_CH0)｜(1<<PWM_CH1);      //enables PWM0 and PWM1
}
-----------------------------------------------------------------------------------------
```

- **PWM interrupt example**

For instance , PWM0 is set to center-aligned mode with vertex, bottom, rising edge and falling edge interrupts turned on,the program is like：

```
-----------------------------------------------------------------------------------------

#define PWM_CH0                    0

#define TIE(N)                     (N<<7)  //N=0~1
#define ZIE(N)                     (N<<6)  //N=0~1
#define PIE(N)                     (N<<5)  //N=0~1
#define NIE(N)                     (N<<4)  //N=0~1
#define MS(N)                      (N<<3)  //N=0~1


#define CKS_IH                  (1<<0)

#define TOG(N)                  (N<<7)    //N=0~1

void PWM_init(void)
{
        P15F = 6;                //set P15 as PWM pin
        INDEX = PWM_CH0;     //set INDEX to PWM0
        PWMCON = TIE(1) | ZIE(1) | PIE(1) | NIE(1) | MS(1) | CKS_IH ;//enables PWM0 interrupt, set PWM0 as center
fixed mode, set PWM0 clock source as IRCH
        PWMCFG = TOG(0) | 0; //disable the phase reversion, no frequency division for the clock
        PWMDIVH  = 0;
        PWMDIVL  = 123;   //3686400/30000=123
        PWMDUTH = 0;
        PWMDUTL  = 37;     //123*0.3=37

        PWMUPD ｜= (1<<PWM_CH0); //set PWM refresh
        while(PWMUPD); //wait until PWM setting refresh completes, necessary before enabling PWM
```

```
          PWMEN  ｜= (1<<PWM_CH0);    //enable PWM0

          PWMCMAX = 0;   //there will be interrupt for every PWM cycle
          INT9EN = 1;        //enable INT9 interrupt
      }
      void INT9_ISR(void) interrupt 14
      {
          if(PWMAIF & TIF0)              // invalid for edge fixed mode
          {
            PWMAIF = TIF0;
              //peak interrupt service routine
          }
if(PWMAIF & ZIF0)                        //invalid for edge fixed mode
          {
              PWMAIF = ZIF0;
              //bottom interrupt service routine
              ...
          }
          if(PWMAIF & PIF0)
          {
              PWMAIF = PIF0;
              //rising edge interrupt service routine
              ...
          }
          if(PWMAIF & NIF0)
          {
             PWMAIF = NIF0;
              //falling edge interrupt service routine..
          }
              ......
      }
```

20 Analog/Digital Converter (ADC)

20.1 Function Introduction

Analog/digital converter is a 12-bit successive approximation (SAR) ADC, with at most 8 input channels. The clock source for ADC is the system clock with frequency division configurable. There are ADC multiple reference voltages for ADC. When internal voltage is selected as the reference voltage, it can be used to test the power supply voltage for the chip and there will be correction to ensure the chip's consistency as well. There is also a compare mode for it with threshold configurable.ADC and operational amplifier are used together, and the detection signal can be reduced before conversion.

20.2 Main Features

- 12 bit resolution
- 8 input channels at most
- Supports ADC interrupt
- ADC clock frequency division configurable
- Alternate reference voltage：internal reference voltage, VDD, external reference voltage
- Support the measurement of VDD and reference ground voltage
- Built-in operational amplifier, supports detection signal reduction, and the reduction multiple is optional
- Input voltage range：VSS<=VIN<=VDD。

20.3 Block Diagram

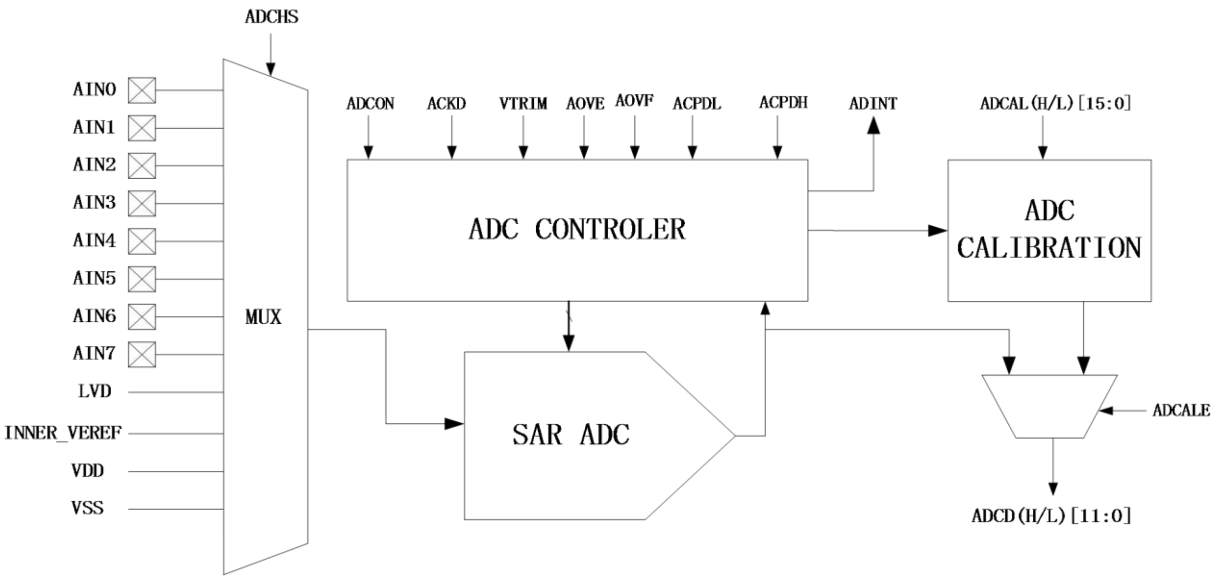


Figure 20-3-1 ADC Architecture

20.4 Function Introduction

ADC can be enabled by AST. When AST=1, the input voltage selected by ADCHS will be analog/digital converted. The clock for ADC is the system clock with frequency division set by ACKD beforehand. When ADC clock is constant, the time for single conversion is set by HTME. The conversion time is (13+2^HTME) ADC clock cycle periods. 12-bit A/D will be stored in register ADCDH and ADCDL after the conversion. AST will be cleared automatically 2.5 clock cycles later. The interrupt flag ADCIF will be set to 1 at the same time. If ADC interrupt is enabled then, ADC interrupt occurs.Figure 22-4-1 shows the conversion timing diagram of ADC.

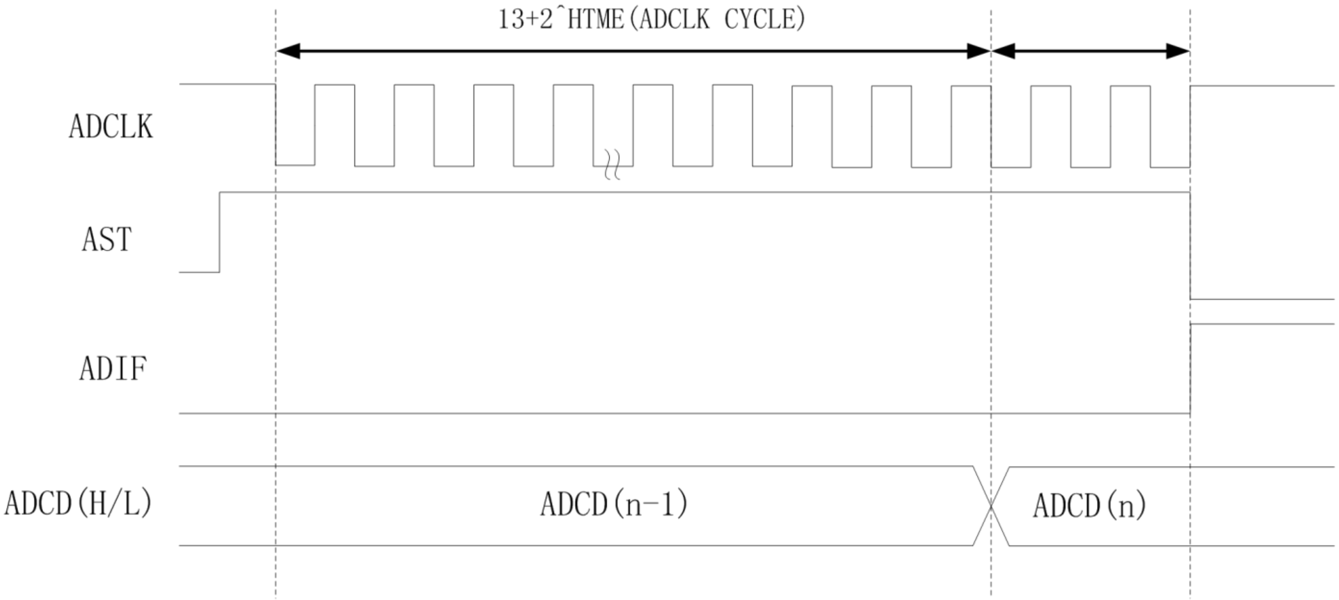


Figure 20-4-1 ADC Sequence Diagram

- **ADC Data Calibration**

The ADC internal reference voltage standard is 1.5V, but due to the influence of the manufacturing process, this voltage has a certain deviation from the theoretical value and needs to be corrected during application. The chip has a built-in voltage correction mechanism, which can be fine-tuned by setting VTRIM (ADCFGH [5:0]). The chip has already corrected this voltage during the CP test phase. The correction parameters are stored in the internal information area of the chip. The user only needs to set the This parameter can be loaded into VTRIM. The calibration parameter loading procedure is as follows:

```
unsigned char read_inner_trim(void)
{
        unsigned char value;
        FSCMD = 0x80;
        PTSH = 0x00;
        PTSL = 0x24;
        FSCMD = 0x81;
        value = FSDAT;
        FSCMD = 0;
        return (value&0x3f);   //Returns the ADC internal voltage correction value
}
void adc_init(void)
{
```

```
………
        ADCFGH = (ADCFGH&0xc0)| read_inner_trim();          //Load the ADC internal voltage correction
                                                              value into the register
        ………
  }
        ----------------------------------------------------------------------------------------------------------------------------------
```

- **OPAMP function**

ADC detection signal can be amplified or attenuated with ratio set by AOPS, which can effectively expand the detection signal range. The reduction factor is set by AOPS, please refer to the description of the register section for details.

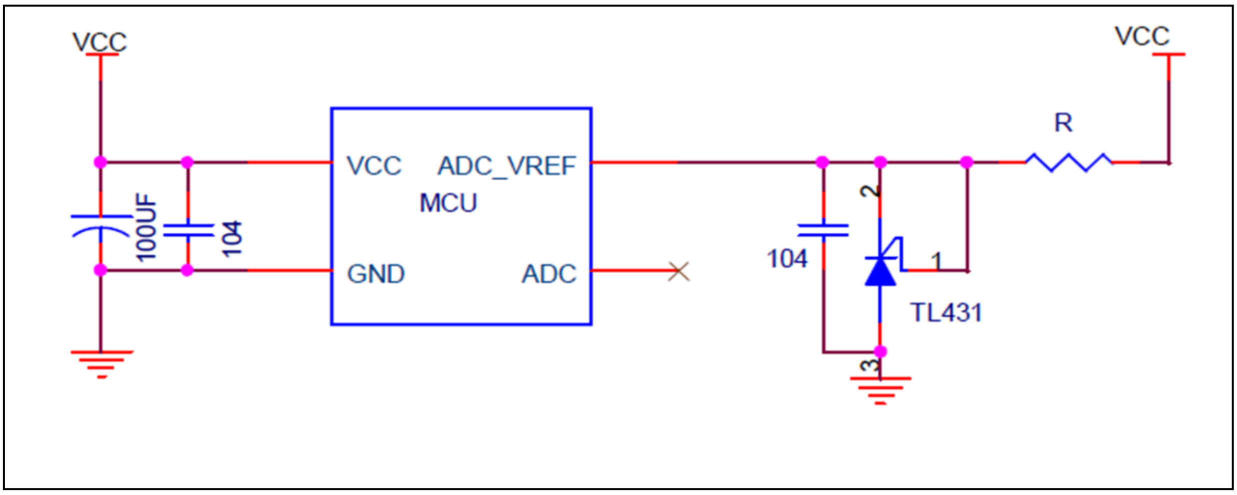
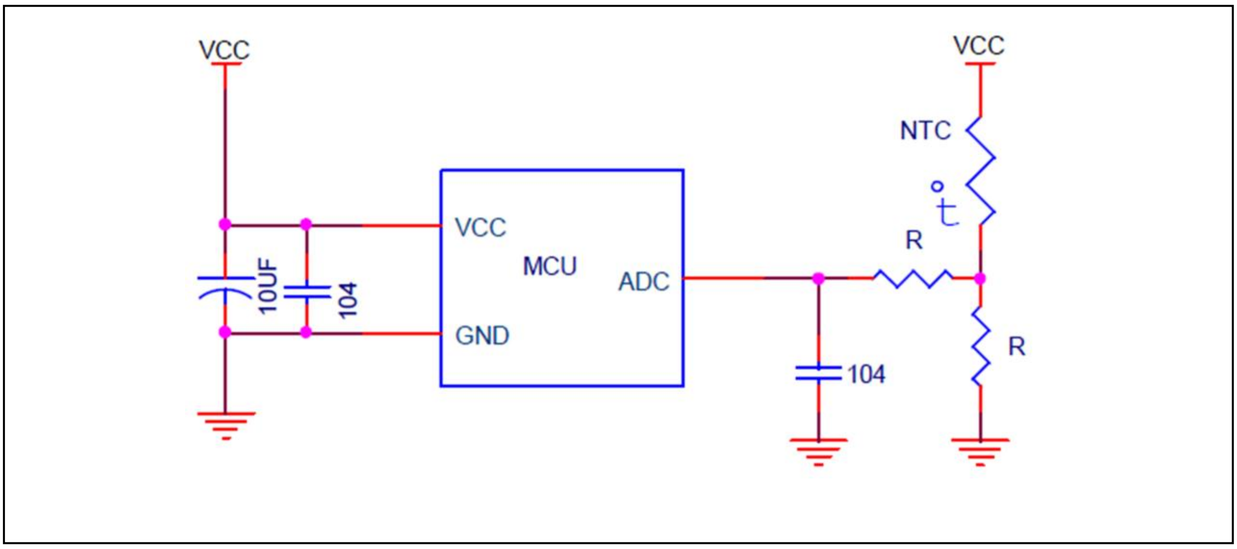


Figure 20-4-2 ADC external reference application reference circuit (the above design circuit and component parameters are for reference only)



Important reminder:
the external voltage to be detected by the ADC cannot be higher than the supply voltage of the chip VCC

Figure 20-4-3 ADC external reference application reference circuit (the above design circuit and component parameters are for reference only)

20.5 Register Description

Table 20-5-1 Register ADCON

B9H	7	6	5	4	3	2	1	0
ADCON	AST	ADIE	ADCIF	HTME			VSEL[1:0]	
R/W	R/W	R/W	R/W	R/W			R/W	
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7	AST	ADC conversion start control bit, write 1 to start the conversion, and the hardware will automatically clear it to 0 after the conversion
6	ADIE	ADC interrupt enable bit, 1 is valid
5	ADCIF	ADC interrupt flag bit, write 1 to clear 0
4~2	HTME	The number of sample-and-hold periods is 2 to the power of HTME
1~0	VSEL	ADC reference voltage selection bit 00: Internal 1.5V (INNER_VREF) as the reference voltage 01: External VDD 10: External VREF 11: Internal 1.5V (INNER_VREF) as the reference voltage Note: When the reference voltage is external VREF, the voltage of VREF must be greater than 1.1V.

Table 20-5-2 Register ADCFGL

BAH	7	6	5	4	3	2	1	0
ADCFGL	ACKD				ADCHS			
R/W	R/W				R/W			
Initial Value	0	0	0		0	0	0	0

Bit number	Bit Symbol	Description
7~5	ACKD	ADC clock division setting 000：no division 001：frequency divided by 2 010：frequency divided by 4 … 111：frequency divided by 14
4	–	–

3~0	ADCHS	ADC channel enable selection 0000：disable the channels 0001：enable channel AD_CH[0](P10) 0010：enable channel AD_CH[1](P11) 0011：enable channel AD_CH[2](P12) 0100：enable channel AD_CH[3](P13) 0101：enable channel AD_CH[4](P14) 0110：enable channel AD_CH[5](P15) 0111：enable channel AD_CH[6](P16) 1000：enable channel AD_CH[7](P17) 1001：enable 1/4 VDD detection 1011：enable INNER_VREF detection 1100：enable LDO voltage detection 1101：enable VSS detection Others：disable the channels

Table 20-5-3 Register ADCFGH

BBH	7	6	5	4	3	2	1	0
ADCFGH	AOPS[1:0]		VTRIM					
R/W	R/W		R/W					
Initial Value	0	0	1	0	0	0	1	1
Bit number	Bit Symbol		Description					
7~6	AOPS		Ratio for OPAMP 00： no narrowing 01：1/4 10：1/3 11：1/2					
5~0	VTRIM		Internal 1.5V reference voltage correction register, with accuracy ±1mV					

Table 20-5-4 Register ADCD

BCH	7	6	5	4	3	2	1	0
ADCDL	ADCDL[3:0]				-			
R/W	R/W				-			
Initial Value	0	0	0	0	-	-	-	-
BDH	7	6	5	4	3	2	1	0
ADCDH	ADCDH[11:4]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol		Description					
11~0	ADCD		ADC conversion result					

20.6 ADC Control Example

For instance, external VDD is selected as the ADC reference voltage, channel 0 selected, ADC interrupt enabled, the program is like：

```
    ------------------------------------------------------------------------------------------------
 #define AST(N)                  (N<<7)  //N=0~1
 #define ADIE(N)                 (N<<6)  //N=0~1
 #define ADIF                            (1<<5)  // Interrupt flag
      #define HTME(N) (N<<2)//N=0~7// Sampling time setting, the time is the clock cycle of the HTME power of 2
      #define VSEL(N) (N)     //N=0~3          // Select reference voltage 0-internal 1-VDD 2-external


    #define ACKD(N)                (N<<5) //N=0~7
    #define ADCALE(N)              (N<<4)    //N=0~1
    #define ADCHS(N)         (N)   //N=0~15          //ADC channel selection, 1~13 corresponds to 0~12
    void ADC_init(void)
    {
         P10F = 5; //set P40 as ADC pin
         ADCON = AST(0) | ADIE(1) | HTME(7) | VSEL(1);//enable ADC interrupt, set the sampling cycle time, VDD
    selected as the reference voltage
         ADCFGL = ACKD(1) | ADCALE(0) | ADCHS(1);//set the ADC clock frequency division, set the ADC channel to
    ADC0
         ADCON |= AST(1); //enable AD conversion
         INT2EN = 1;          //enable INT2 interrupt
    }
    void ADC_ISR (void) interrupt 7
    {
         unsigned int AD_Value;
         if(ADCON & ADIF)
         {
              ADCON |= ADIF; //clear ADC interrupt
              AD_Value = ADCDH*256 + ADCDL;        //read ADC value
              AD_Value >>= 4;
              ADCON |= AST(1); //enable next AD conversion
              }
              .....
              }
```

21 Touch Key

21.1 Function Introduction

With great anti-jamming performance, CA51F3 series chip can pass EFT,CS test and etc. The Touch Key module supports 20 channels at most,and the TK_CAP pin needs to be connected to a Cx capacitor with a capacitance range of 10nF~47nF and a capacitance accuracy of 10% or less, and it is recommended to use polyester capacitors, X7R capacitors or NPO chip capacitors. Cx can directly affect the touch sensitivity. The smaller the Cx capacitance value, the lower the sensitivity, and the larger the capacitance value, the higher the sensitivity.

For applications with low-power requirements, a mechanism is also designed to allow the chip to work normally even when in STOP mode.

21.2 Main Features

- Great anti-jamming performance which meets the EMC(CS) Standard
- Supports 20 channels at most
- Supports low power consumption mode
- Touch interrupt supported
- Clock division supported for charging/discharging
- Supports manual control and automatic mode
- Selective levels for comparator’s threshold
- Touch can set internal charging and internal reference, can effectively suppress power supply low frequency interference
- Supports multiplexing of touch pins and LED driver pins
- Built-in waterproof compensation mechanism
- Waking up threshold configurable in STOP mode

21.3 Architecture

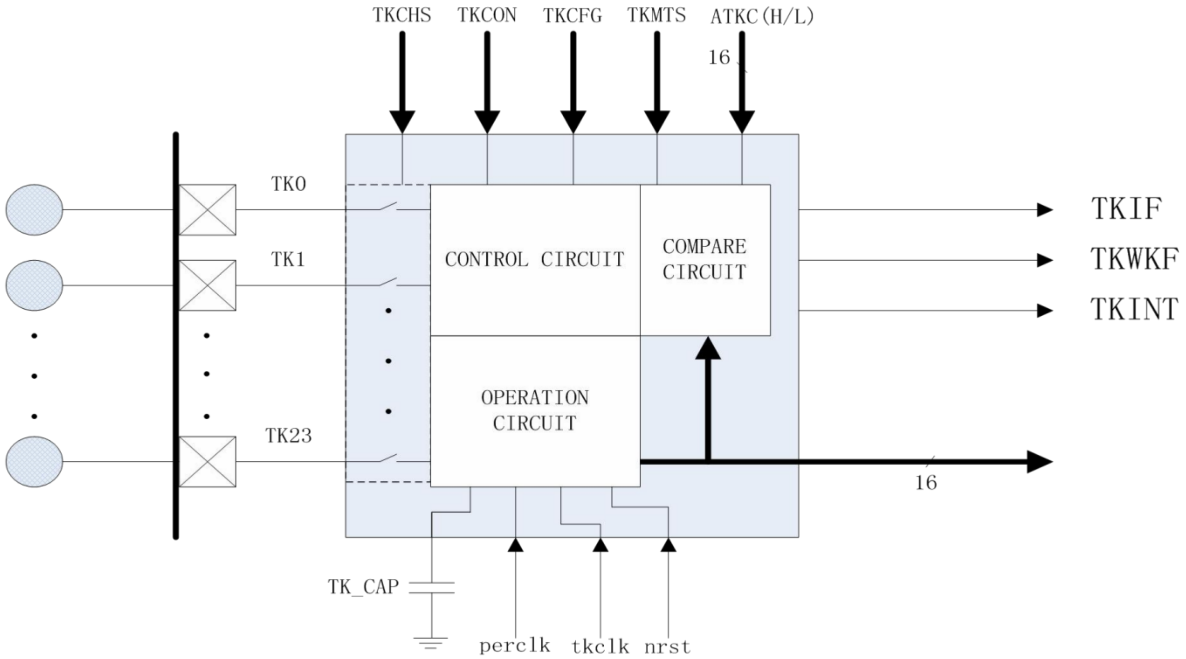


Figure 21-3-1 Touch Key Module Architecture

21.4 Function Description

21.4.1 Manual Control Mode and Automatic Mode

The touch data collection can be enabled by TKST in manual control mode. When TKST=1, the module starts to collect the data through channel selected. There are at most 6 channels for one group for the channels selection, which is set by the register TKCHS with index. Every time the collection is enabled, one group of channels' data will be collected. TKST will be cleared automatically when the collection is over. The corresponding channel's interrupt flag will be set to 1. The touch data can be read from register TKMS by setting register INDEX then.

Manual control mode and automatic mode can be selected by TMEN. In automatic mode, unlike in manual control mode, the touch data collection is enabled by timer's timing. The clock source for the timer can be IRCL or XOSCL, which is selected by RTCKS in register CKSEL；the timing can be set by register TKMTS.

21.4.2 Touch Key Clock Frequency Division

The clock source for electrode charging/discharging is TFRC, which is extremely important for the touch module's performance. When the clock frequency for charging/discharging is too high, the touch electrode may not be charged properly, which makes the data change too small when fingers touch the key. The frequency prescale can be set by TKDIV. With proper frequency, touch module will perform even better.

21.4.3 Low Power consumption Mode

As long as the clock source TFRC and low speed clock(IRCL or XOSCL) are enabled in STOP mode, the touch Key module is able to charge/discharge normally. If TWKE=0, the data collection interrupt will awaken the CPU and then software will read data collected. The chip enters STOP mode again when data reading us over. the module also includes threshold compare function. Users can set trigger threshold and the collected data will be compared with the threshold set by users in STOP mode. When the data exceeds the threshold, if TWKE=1 then, the interrupt will awaken CPU. CPU will collect the data and do judgement after being waken up.

21.4.4 Touch key common LED driver

Touch button common LED driver can be achieved by N touch button and N touch indicator control only need (N + 1) pins. Among them, the touch keys and LED driver positive control shared pins, LED negative terminal connected to COM, touch and LED control is achieved in a time-sharing manner.

Each touch has a separate control bit TLENx (x=0~19, corresponding to TK0~TK19) to enable the common LED drive function, it should be noted that the corresponding touch pin function must be turned on. After common LED enable, TLDATx(x=0~19, corresponding to LED0~LED19) can control each LED independently. COM pin can be selected by TLCOMS.

Touch data acquisition and LED control are implemented in a time-sharing manner, where the time for touch data acquisition is defined by TLCNTK, and the time for LED scanning is defined by TLCNTL. Note that the time defined by TLCNTK is the total time for each group of touch acquisition, and the number of touch channels in each group is 1~6. When the actual touch time is greater than the defined time, a TLERR interrupt will be generated. When the counter counts to the time defined by TLCNTK, the TLKOV interrupt is generated. Once the touch acquisition phase is completed, it enters the LED scan phase. TLCNTL defines the time of the LED scan phase, which affects the duty cycle of the LED scan, that is, it affects the brightness of the LED, and can be adjusted as needed during the application. In the LED scan phase, when the counter counts to the time defined by TLCNTL, TLLOV interrupt will be generated, so that a complete touch common LED cycle is completed. The following is a schematic of the operating phases.

Important reminder: in the application of touch pins and LED driver pin multiplexing mode, due to the existence of the diode junction capacitance of the LED itself, the junction capacitance of different types of LED lights have a large difference, and this junction capacitance in the LED lights on and off when the performance may be inconsistent (especially white LED lights are more obvious), this junction capacitance and its inconsistency will cause adverse effects on the touch, so in the application of this This junction capacitance and its inconsistency will have an adverse effect on touch, so when applying this mode, the LEDs used should be strictly selected, and the LEDs should not be changed casually after mass production.

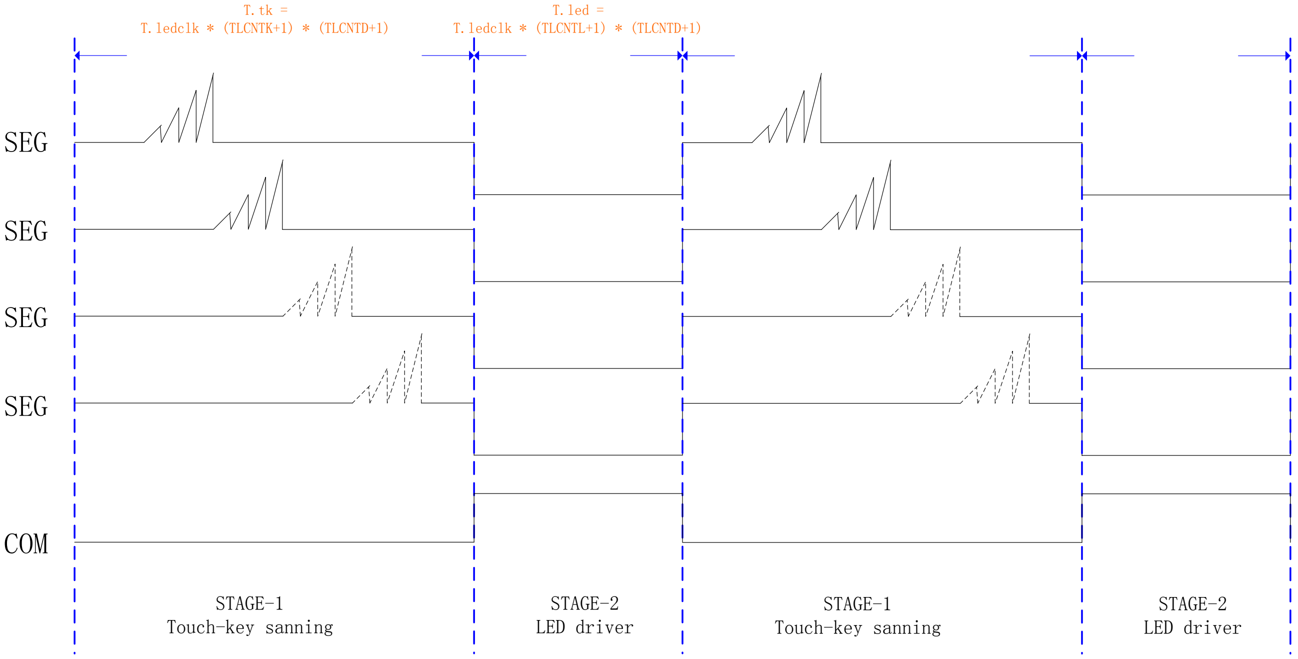


Figure 21-5-1 Touch common LED driver schematic

21.4.5 Touch internal reference and internal op-amp

The touch module integrates an internal op-amp, which can be selected as the charging power supply for the touch keys via TKPWS (TKPWC[1]), and the charging voltage via VDS (TKPWC[5:4]). Alternatively, touch can also select the internal reference as the threshold voltage for the touch internal comparator via TKCVS(TKPWC[0]), and the internal reference voltage is selected via VIRS(TKPWC[3:2]).

21.4.6 Touch waterproof compensation mechanism

Touch is designed with a waterproof compensation mechanism, which can be turned on by TKPC (TKPWC[7:6]) for 2. After turning on this function, the non-selected touch pins will be synchronized to output the same compensation waveform as the charging frequency, which can effectively reduce the impact of parasitic capacitance between touch keys and achieve the effect of waterproof. Note: When turning on the waterproof compensation, the touch charging power supply should preferably be selected as external power supply.

21.5 Register Description

Table 21-5-1 Register TKCON

C1H	7	6	5	4	3	2	1	0
TKCON	TKST	TKIE	TMEN	TWKE	-	VRS[2:0]		
R/W	R/W	R/W	R/W	R/W	-	R/W		
Initial Value	0	0	0	0	-	0	0	0
Bit number	Bit Symbol		Description					

7	TKST	Data collection start enable control, 1 enables it, cleared automatically after the data collection
6	TKIE	TK interrupt enable control, 1 enables it
5	TMEN	Start mode selection 0: enabled by TKST 1: enabled by Timer
4	TWKE	Interrupt trigger selection 0: interrupt triggered when sampling is done 1: interrupt triggered when data collected exceeds the threshold
3	-	-
2~0	VRS	Reference voltage selection for comparator’s threshold voltage (the threshold voltage is directly proportional with VDD) 0： maximum thresholdvoltage … 7： minimum thresholdvoltage

Table 21-5-2 Register TKCFG

C2H	7	6	5	4	3	2	1	0
TKCFG	TKDIV			TKTMS				
R/W	R/W			R/W				
Initial Value	1	1	1	1	1	1	1	1
Bit number	Bit Symbol	Description						
7~5	TKDIV	Frequency division selection for touch key clock 000： no division 001： frequency divided by 2 010： frequency divided by 3 … 111： frequency divided by 8						
4~0	TKTMS	The discharging time setting for external modulation capacitor Discharging time = TKTMS x 128 x clock cycle period When TKDIV=0, the discharging time ranges from 32us to 992us *Note： TKTMS cannot be set to 0*						

Table 21-5-3 Register TKPWC

8103H	7	6	5	4	3	2	1	0
TKPWC	TKPC		VDS		VIRS		TKPWS	TKCVS
R/W	R/W		R/W		R/W		R/W	R/W
Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
7~6	TKPC	Touch button undersampled channel output control 00： hover 01： Low output						

		10： Output compensation *Note：* *1. This function is only available for pins that are configured for the touch button function* *2. If the shared LED drive function is enabled, then the pin selected as the LED drive, this function will be disabled.*
5~4	VDS	Internal op-amp output voltage selection 00：2V 01：2.5V 10：3V 11：4V
3~2	VIRS	Internal voltage reference selection 00：1.0V 01：1.5V 10：2.0V 11：2.5V
1	TKPWS	Charging power selection 0： Select external power supply 1： Select internal op-amp output
0	TKCVS	Charging reference voltage selection 0： Selecting an external voltage reference 1： Selecting the internal voltage reference

Table 25-5-4 Register TKMTS

C3H	7	6	5	4	3	2	1	0
TKMTS	TKMTS[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
7~0	TKMTS	The start time setting register in timing mode the start time=(TKMTS+1) × 32 × low speed clock cycle period If the low speed clock’s frequency is 32.768K, the start time ranges from 0.977ms to 250ms.						

Table 25-5-5 Register TKCHS

C4H	7	6	5	4	3	2	1	0
TKCHS	POL	NPOL	TKPS					
R/W	R/W	R/W	R/W					
Initial Value	0	0	0	0	0	0	0	0
Note：TKCHS is register with index, INDEX=0~5 indicates TKCHS0~TKCHS5 respectively								
Bit number	Bit Symbol	Description						
7	POL	ATKnC direction setting for threshold comparison 0： interrupt when the data collected is less than the threshold 1： interrupt when the data collected is greater than the threshold						

6	NPOL	ATKnN direction setting for threshold comparison 0：interrupt when the data collected is less than the threshold 1：interrupt when the data collected is greater than the threshold
5~0	TKPS	Channel selection 000000：disable TK0~TK19 000001：TK0 selected 000010：TK1 selected 000011：TK2 selected ...... 010100：TK19 selected 011001：Internal reference capacitor selected

Table 25-5-6 Register ATKC

C5H	7	6	5	4	3	2	1	0
ATKCL	ATKC[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
C6H	7	6	5	4	3	2	1	0
ATKCH	ATKC[15:8]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Note：ATKC is register with index, INDEX=0~5 indicates ATKC0~ATKC5 respectively								
Bit number	Bit Symbol		Description					
15~0	ATKC		Compare threshold setting register, when TWKE=1, ATKC0~ATKC5 will be compared with TKMS0~TKMS5 automatically					

Table 25-5-7 Register ATKN

E6H	7	6	5	4	3	2	1	0
ATKNL	ATKN[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
E7H	7	6	5	4	3	2	1	0
ATKNH	ATKN[15:8]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Note：ATKC is register with index, INDEX=0~5 indicates ATKC0~ATKC5 respectively								
Bit number	Bit Symbol		Description					
15~0	ATKN		Compare threshold setting register, when TWKE=1, ATK0N~ATK5N will be compared with TK0MS~TK5MS automatically					

Table 25-5-8 Register TKMS

CEH	7	6	5	4	3	2	1	0
TKMSL	TKMS[7:0]							
R/W	R							
Initial Value	0	0	0	0	0	0	0	0
CFH	7	6	5	4	3	2	1	0
TKMSH	TKMS[15:8]							
R/W	R							
Initial Value	0	0	0	0	0	0	0	0
Note：TKMS is register with index, INDEX =0~5 indicates TKMS0~TKMS5 respectively								
Bit number	Bit Symbol		Description					
15~0	TKMS		Touch key sampling data register					

Table 25-5-9 Register TKIF

C7H	7	6	5	4	3	2	1	0
TKIF	-		TKIF5	TKIF4	TKIF3	TKIF2	TKIF1	TKIF0
R/W	-		R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	-	-	0	0	0	0	0	0
Bit number	Bit Symbol		Description					
7~6	-		-					
5~0	TKIFx(x=5~0)		TK data collection interrupt flag, the bits correspond to 6 channels in order. When TWKE=1, TKIFx implies the data exceeds the ATKC or ATKN threshold					

Table 25-5-10 Register TKMAXF

D5H	7	6	5	4	3	2	1	0
TKMAXF	-	-	TKMXF5	TKMXF4	TKMXF3	TKMXF2	TKMXF1	TKMXF0
R/W	-	-	R	R	R	R	R	R
Initial Value	-	-	0	0	0	0	0	0
Bit number	Bit Symbol		Description					
7~6	-		-					
5~0	TKMXFx(x=5~0)		1 indicates that TKxMS exceeds ATKxC threshold, while 0 indicates TKxMS does not exceed ATKxC threshold. The polarity is set by POLx ； if TWKE=1, setting TKMXFx to 1 will set TKIFx as well； the software can not do anything to it					

Table 25-5-11 Register TKMINF

D6H	7	6	5	4	3	2	1	0
TKMINF	-	-	TKMNF5	TKMNF4	TKMNF3	TKMNF2	TKMNF1	TKMNF0
R/W	-	-	R	R	R	R	R	R
Initial Value	-	-	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~6	-	-
5~0	TKMNFx(x=5~0)	1 indicates that TKxMS exceeds ATKxN threshold, while 0 indicates TKxMS does not exceed ATKxN threshold. The polarity is set by NPOLx； if TWKE=1, setting TKMXFx to 1 will set TKIFx as well； the software can not do anything to it

Table 25-5-12 Register TLEN

8106H	7	6	5	4	3	2	1	0
TLEN (INDEX=0)	TLEN7	TLEN6	TLEN5	TLEN4	TLEN3	TLEN2	TLEN1	TLEN0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0
TLEN (INDEX=1)	TLEN15	TLEN14	TLEN13	TLEN12	TLEN11	TLEN10	TLEN9	TLEN8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0
TLEN (INDEX=2)	-	-	-	-	TLEN19	TLEN18	TLEN17	TLEN16
R/W	-	-	-	-	R/W	R/W	R/W	R/W
Initial Value	-	-	-	-	0	0	0	0

Note：

TLENx=1 (x=0,1,2,... ,19), LEDx to enable, the corresponding TKx pin must select the touch button function

Bit number	Bit Symbol	Description
3~0 (INDEX=2)	TLEN19~TLEN16	LED19~LED16 enable, 1 valid
7~0 (INDEX=1)	TLEN15~TLEN8	LED15~LED8 enable, 1 valid
7~0 (INDEX=0)	TLEN7~TLEN0	LED7~LED0 enable, 1 valid

Table 25-5-13 Register TLDAT

8107H	7	6	5	4	3	2	1	0
TLDAT (INDEX=0)	TLDAT7	TLDAT6	TLDAT5	TLDAT4	TLDAT3	TLDAT2	TLDAT1	TLDAT0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0
TLDAT (INDEX=1)	TLDAT15	TLDAT14	TLDAT13	TLDAT12	TLDAT11	TLDAT10	TLDAT9	TLDAT8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0
TLDAT (INDEX=2)	-	-	-	-	TLDAT19	TLDAT18	TLDAT17	TLDAT16
R/W	-	-	-	-	R/W	R/W	R/W	R/W
Initial Value	-	-	-	-	0	0	0	0

Bit number	Bit Symbol	Description
3~0 (INDEX=2)	TLDAT19~TLDAT16	LED19~LED16 data, 1 means drive effective level
7~0 (INDEX=1)	TLDAT15~TLDAT8	LED15~LED8 data, 1 means drive effective level
7~0 (INDEX=0)	TLDAT7~TLDAT0	LED7~LED0 data, 1 means drive effective level

Table 25-5-14 Register TLCON

8108H	7	6	5	4	3	2	1	0
TLCON	TLEIE	TLKIE	TLLIE	-	-	TLLVS		TLPOL
R/W	R	R/W	R/W	-	-	R/W		R/W
Initial Value	0	0	0	-	-	0	0	0

Bit number	Bit Symbol	Description
7	TLEIE	TLERR interrupt enable, 1 valid
6	TLKIE	TLKOV interrupt enable, 1 valid
5	TLLIE	TLLOV interrupt enable, 1 valid
4~3	-	-
2~1	TLLVS	Touch key scan phase, COM pin level selection 00: Output high 01: Output low 10: Pull-up high Others: Reserved
0	TLPOL	LED drive phase, effective drive level selection 0: drive level high valid 1: Drive level low valid

Table 25-5-15 Register TLFLG

8109H	7	6	5	4	3	2	1	0
TLFLG	TLERR	TLKOV	TLLOV	-	-	-	-	-
R/W	R/W	R/W	R/W	-	-	-	-	-
Initial Value	0	0	0	-	-	-	-	-

Bit number	Bit Symbol	Description
7	TLERR	Touch key scan phase time setting too short flag 0: indicates that the time set by the user is enough for the touch button scanning 1: indicates that the time set by the user is not enough, the timer has finished counting and the channel has not been scanned *Note.* *This bit is the hardware auto-setting and auto-zeroing bit. Considering the convenience of user use, this bit can also be written 1 to clear 0 by software.*
6	TLKOV	Touch key scan stage timer count full flag, 1 means count full, software write 1 to clear 0
5	TLLOV	LED drive stage timer count full flag, 1 means count full, software write 1 clear 0
4~0	-	-

Table 25-5-16 Register TLCKS

810AH	7	6	5	4	3	2	1	0
TLCKS	-	-	-	-	-	TLCKS[2:0]		
R/W	-	-	-	-	-	R/W		
Initial Value	-	-	-	-	-	0	0	0

Bit number	Bit Symbol	Description
7~3	-	-
2~0	TLCKS	LED driver operating clock selection 001: IRCL 010：IRCH 011：XOSCL

		Others: Off

Table 25-5-17 Register TLCNTK

810BH	7	6	5	4	3	2	1	0
TLCNTKL	TLCNTK[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
810CH	7	6	5	4	3	2	1	0
TLCNTKH	TLCNTK[15:8]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
15~0	TLCNTK	Touch key scan phase time configuration register $T_{.tk}$ = $T_{.ledclk}$ x (TLCNTK+1) x (TLDIV+1) *Note:* *T.tk, indicates the time of the touch button scan phase; T.ledclk, indicates the period of the operating clock of the LED driver circuit.*

Table 25-5-18 Register TLCNTL

810DH	7	6	5	4	3	2	1	0
TLCNTLL	TLCNTL[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
810EH	7	6	5	4	3	2	1	0
TLCNTLH	TLCNTL[15:8]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
15~0	TLCNTL	LED drive phase time configuration register $T_{.led}$= $T_{.ledclk}$x (TLCNTL+1) x (TLDIV+1) *Note:* *T.led, indicates the time of the LED drive phase*

Table 25-5-19 Register TLDIV

810FH	7	6	5	4	3	2	1	0
TLDIV	-	-	-	-	TLDIV[3:0]			
R/W	-	-	-	-	R/W			
Initial Value	-	-	-	-	0	0	0	0

Bit number	Bit Symbol	Description
7~4	-	-
3~0	TLDIV	$T_{.ledclk}$ clock dividing register The dividing frequency multiplier is (TLDIV+1)

Table 25-5-20 Register TLCOMS

8110H	7	6	5	4	3	2	1	0
TLCOMS	-	-	-	-	TLCMS[3:0]			
R/W	-	-	-	-	R/W			
Initial Value	-	-	-	-	0	0	0	0

Bit number	Bit Symbol	Description
7~4	-	-
3~0	TLCMS	COM pin selection control 0001: Select P0.0 0010: Select P0.1 0011: Select P0.2 0100: Select P0.3 0101: Select P0.4 Others: Close

21.6 Touch Key Control Example

Note：For this part please refer to our company' s standard touch key library software and related documents.

22 Low Voltage Detection(LVD)

22.1 Function Introduction

Low voltage Detection (LVD) is used to monitor the chip's power supply VDD, with detectable range 1.8V~4.8V. When VDD is lower than the voltage set, either interrupt or reset occurs.

Figure 22-1-1 shows the architecture of LVD.

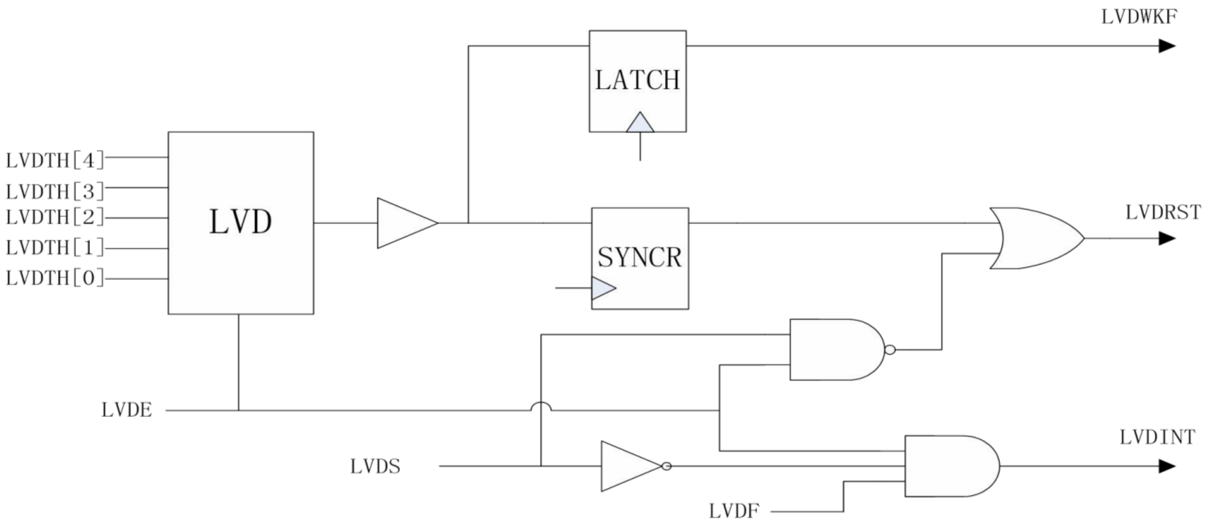


Figure 22-1-1 LVD Schematic

22.2 Function Description

LVD function is enabled by LVDE and the trigger voltage is set by LVDTH. When VDD is lower than the trigger voltage, the LVDF will be set to1. If LVDS=0 then, there will be an interrupt; if LVDS=1, it will generate a reset signal. However, LVD reset signal will not reset itself, which means register LVDCON remains its status.If it is set to interrupt mode, when VDD is continuously lower than the set voltage, only one interrupt will be generated, and the interrupt will only be generated again when VDD is higher than the set voltage and then falls below the threshold voltage again.

Note: Affected by the process, the set trigger voltage has some deviation from the actual value, the deviation value is less than ±50mv.

22.3 Register Description

Table 22-3-1 Register LVDCON

EFH	7	6	5	4	3	2	1	0
LVDCON	LVDE	LVDS	LVDF	-	LVDTH[3:0]			
R/W	R/W	R/W	R/W	-	R/W			
Initial Value	0	0	0	-	0	0	0	0

Bit number	Bit Symbol	Description
7	LVDE	LVD enable control, 1 enables it
6	LVDS	LVD function selection 0：interrupt 1：reset
5	LVDF	LVD flag, cleared when 1 is written to it
4	-	-
3~0	LVDTH	LVD trigger level selection 0000：1.8V 0001：2.0V 0010：2.2V 0011：2.4V 0100：2.6V 0101：2.8V 0110：3.0V 0111：3.2V 1000：3.4V 1001：3.6V 1010：3.8V 1011：4.0V 1100：4.2V 1101：4.4V 1110：4.6V 1111：4.8V

22.4 LVD Control Example

LVD interrupt example

For instance , set LVD to interrupt mode with trigger voltage 3V, the program is like：

```
------------------------------------------------------------------------------------------
#define LVDE(N)          (N<<7)    //N=0~1
#define LVDS_reset (1<<6)
#define LVDS_int    (0<<6)
#define LVDF         (1<<5)
#define LVDTH_3V        6
void LVD_init(void)
{
        LVDCON = LVDE(1) | LVDS_int | LVDTH_3V; //enables LVD and set it to interrupt mode, set the trigger voltage to 3V
        INT4EN = 1; //enables INT4 interrupt
}
void INT4_ISR (void) interrupt 9
{
        if(LVDCON & LVDF)
        {
                LVDCON |= LVDF;         //clear LVD interrupt flag
//LVD interrupt service routine
                ...

        }
        ...
}
------------------------------------------------------------------------------------------
```

LVD reset example

For instance , set LVD to reset mode with trigger voltage 3V, the program is like：

```
------------------------------------------------------------------------------------------
#define LVDE(N)          (N<<7)    //N=0~1
#define LVDS_reset (1<<6)
#define LVDS_int    (0<<6)
#define LVDF         (1<<5)
#define LVDTH_3V        6
void LVD_init(void)
{
        LVDCON = LVDE(1) | LVDS_reset | LVDTH_3V;//enables LVD and set it to reset mode, set the trigger voltage to 3V
}
```

23 Program Download and Simulation

23.1 Program Download

CA51F3 Series chip download programs using ISP method. The chip can connect to the download tool with UART port. Any of the UART ports can be used for ISP.

For more download steps please refer to *CACHIP development tools manual*.

23.2 Online Simulation

CA51F3 Series chip supports online simulation. Chip can communicate with the emulator with IIC interface. The default port for IIC is P11(IIC SDA) and P12(IIC SCL). Since the IIC is used for communication between the chip and emulator, the IIC port can not be set as other functions and IIC function can not be used in software either, otherwise the simulation will not be enabled. The speed of IIC is decided by the main clock, so the main clock can not be set as low speed clock by the software. In addition, it can not enter power save mode either, otherwise the communication between the chip and emulator will be influenced.

When TSME=0(PCON[3]), the chip is forbidden to enter simulation mode. TSMODE(PCON[2]) will be set to 1 in simulation mode. The software can decide whether to enter power save mode or switch to low speed clock according to the status of TSMODE.

For more details about the simulation function please refer to the documents related to emulator.

24 Electrical Specification

24.1 Limit Parameter

Parameter	Minimum	Maximum	Unit
DC voltage for power supply	-0.3	6	V
Input voltage for I/O pin	-0.3	VDD+0.3	V
Working temperature	-40	85	℃
Storage temperature	-55	125	℃
CPU working frequency	-	27	MHz

Note：Exceeding the "limit parameter" range may cause damage to the chip. It is not possible to predict the working state of the chip outside the above range. If the chip is operated outside the marked range for a long time, the reliability of the chip may be affected

24.2 DC Electrical Specification

Chip Parameters	Sym bols	Operating Voltage	Minimum	Typical	Maximum	Unit	Test conditions
Operating current	Iop1	VDD=1.8V		0.496		mA	System clock is IRCH (3.6864MHz), other clocks are off, LDO is set to default value (high power mode, output voltage is 1.61V), no load on all output pins, all digital input pins are not floating, all peripherals are off, CPU executes NOP instruction
		VDD=3.3V		0.546			
		VDD=5V		0.550			
	Iop2	VDD=1.8V		2.18		mA	System clock is PLL output, PLL is set to 6x, reference clock IRCH frequency is 3.6864MHz, other clocks are off, LDO is set to default value (high power mode, output voltage is 1.61V), no load on all output pins, all digital input pins are not floating, all peripherals are off, CPU executes NOP instruction
		VDD=3.3V		2.47			
		VDD=5V		2.49			
	Iop3	VDD=1.8V		22.6		uA	System clock is IRCL (131kHZ), other clocks are off, LDO is set to low power mode, output voltage is 1.61V, no load on all output pins, all digital input pins are not floating, all peripherals are off, CPU executes NOP instruction
		VDD=3.3V		23.6			
		VDD=5V		24			
	Iop4	VDD=1.8V		12.8		uA	System clock is XOSCL (32.768kHZ), other clocks are off, LDO is set to low power mode, output voltage is 1.61V, no load on all output pins, all digital input pins are not floating, all peripherals are off, CPU executes NOP instruction
		VDD=3.3V		13.3			
		VDD=5V		13.6			
	Iop5	VDD=1.8V		15.1		uA	System clock is XOSCL (32.768kHZ), other clocks off, LDO set to low power mode, output voltage is 1.61V, LCD driver on (no
		VDD=3.3V		17.6			
		VDD=5V		20.3			

							external LCD panel), LCD set to minimum current drive, 1/3bias, 1/4duty, LCD clock is XOSCL, LCD_CAD off (CAD_MOD=0), all LCD pins on, all other output pins no load, all digital input pins not floating, other peripherals off
STOP mode current	I_{stp}	VDD=1.8V		5.3		uA	All clocks are off, all output pins are unloaded, all digital input pins are not floating, all peripherals are off, LDO is set to low power mode, Flash goes to sleep mode, and CPU goes to STOP mode.
		VDD=3.3V		5.5			
		VDD=5V		5.7			
IDLE mode current	I_{idl1}	VDD=1.8V		0.269		mA	The system clock is set to IRCH (3.6864MHz), other clocks are off, all output pins have no load, all digital input pins do not float, all peripherals are off, LDO is set to low power mode, Flash enters sleep mode, and CPU enters IDLE mode.
		VDD=3.3V		0.284			
		VDD=5V		0.286			
	I_{idl2}	VDD=1.8V		0.469		mA	The system clock is PLL output, PLL is set to 6x, the reference clock IRCH frequency is 3.6864MHz, other clocks are off, all output pins have no load, all digital input pins do not float, all peripherals are off, and the CPU enters IDLE mode.
		VDD=3.3V		0.500			
		VDD=5V		0.504			
	I_{idl3}	VDD=1.8V		13.6		uA	The system clock is set to IRCL (131KHz), other clocks are off, all output pins are unloaded, all digital input pins are not floating, all peripherals are off, the LDO is set to low power mode, and the CPU enters IDLE mode.
		VDD=3.3V		13.9			
		VDD=5V		14.2			
	I_{idl4}	VDD=1.8V		10.3		uA	The system clock is set to XOSCL (32.768KHz), other clocks are off, all output pins have no load, all digital input pins do not float, all peripherals are off, LDO is set to low power mode, Flash enters sleep mode, and CPU enters IDLE mode.
		VDD=3.3V		10.5			
		VDD=5V		10.8			
	I_{idl5}	VDD=1.8V		11.6		uA	The system clock is XOSCL (32.768kHZ), other clocks are off, LCD driver is turned on, LCD is set to minimum current drive, 1/3bias, 1/4duty, LCD_CAD is off (CAD_MOD=0), LCD clock is XOSCL, all LCD pins are turned on, all other output pins have no load, all digital input pins are not float, CPU enters IDLE mode.
		VDD=3.3V		13.7			
		VDD=5V		16.3			
IO port input high voltage (Smit mode on)	V_{hi1}	VDD=1.8V	0.75	-	1.8	V	-
		VDD=3.3V	1.20		3.3		
		VDD=5V	1.50		5		
IO port input high voltage (Smit mode off)	V_{hi2}	VDD=1.8V		0.5*VDD	VDD	V	-
		VDD=3.3V					
		VDD=5V					
IO port input low voltage (Smit mode on)	V_{lo1}	VDD=1.8V	0	-	0.62	V	-
		VDD=3.3V	0	-	0.85		
		VDD=5V	0	-	1.20		
IO port input low voltage (smit mode off)	V_{lo2}	VDD=1.8V	0	0.5*VDD		V	-
		VDD=3.3V					
		VDD=5V					
IO port push current	I_{pu}	VDD=3.3V	-	6.05	-	mA	IO set to push-pull output mode, drive capability set to maximum, Vol=VDD-0.3V
		VDD=5V	-	8.46	-		
IO port current filling	I_{ol}	VDD=3.3V	-	13.34	-	mA	IO set to push-pull output mode, drive capability set to maximum, Vol = GND + 0.3V
		VDD=5V	-	19.05	-		
COM port	Isi	VDD=3.3V		70		mA	IO set to push-pull output or LED COM

current filling		VDD=5V		96			pin function, drive capability set to maximum, Sink function on, Vol=GND+0.3V
IO port strong pull-down resistor	Rd1	VDD=1.8~5.5 V		15		KΩ	-
IO port weak pull-down resistor	Rd2	VDD=1.8~5.5 V	-	45	-	KΩ	-
IO port strong pull-up resistor	Ru1	VDD=1.8~5.5 V	-	10	-	KΩ	-
IO port weak pull-up resistor	Ru2	VDD=1.8~5.5 V		45		KΩ	

Note: The above parameters are typical chip test results randomly selected for reference only.

24.3 AC Electrical Specification

AC Electrical Specification(VDD=1.8-5.5V, TA=25℃, unless there are other explanations)

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
Internal low-speed clock (IRCL) start-up time	Trc1	-	50	-	us	IRCL frequency 131KHz
Internal high-speed clock (IRCH) start-up time	Trc2	-	10	-	us	IRCH frequency 3.6864MHz
External low-speed clock (XOSCL) start-up time	Tosc1	-	1	-	s	XOSCL frequency 32.768KHz
Time for PLL to be stable	Tpll	-	50	-	us	Reference cloclk is IRCH with frequency 3.6864MHz, PLL frequency multiplied by 6
Time of the reset pulse	Trst	-	0.5	-	us	

Note：

VDD=3.3V,TA=25 ℃, the factory frequency for internal high speed clock is 3.6864MHz, with deviation less than 1%..

24.4 ADC Electrical Specification

ADC Electrical Specification (Ta=25℃,VDD for reference voltage)

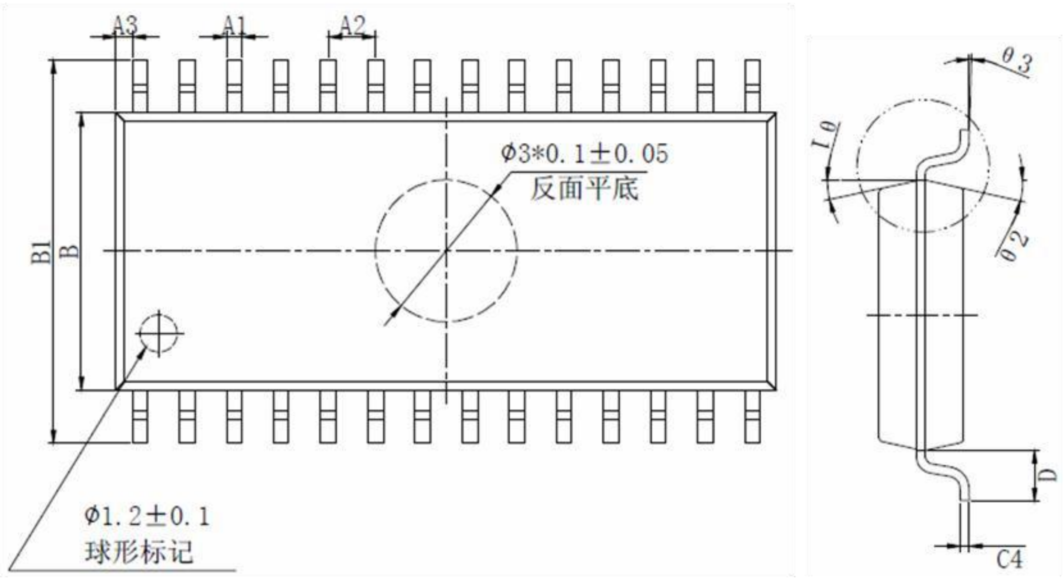
Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
Working voltage	V_{AD}	1.8		5.5	V	
ADC precision	NR		11		Bit	GND<=Vin<=Vref,ADC reference voltage is VDD or external reference
			10		Bit	GND<=Vin<=Vref,ADC reference voltage is internal reference
ADC external reference voltage	Vex	1.1	-	VDD	V	
ADC input voltage	Vin	0	-	VDD	V	
ADC input resistance	Rin	2	-	-	MΩ	VDD=5V
ADC conversion current	IADC	-	180	-	uA	VDD=5V
Non-linear differential error	DNL	-	-	±3	LSB	VDD=5V
Non-linear integral error	INL	-	-	±3	LSB	VDD=5V
Full scale error	EF	-	±3	±4	LSB	VDD=5V
Offset error	Ez	-	±0.5	±1	LSB	VDD=5V
Conversion time	T_{CON}	-	16	-	Clock cycle	

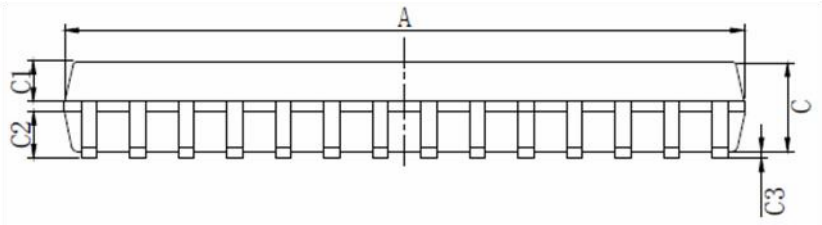
Note: (1) ADC input resistance is the input resistance of ADC itself under DC conditions;

(2) When testing ADC, the internal resistance of the signal source of the connection path needs to be less than 10KΩ

25 Package Type

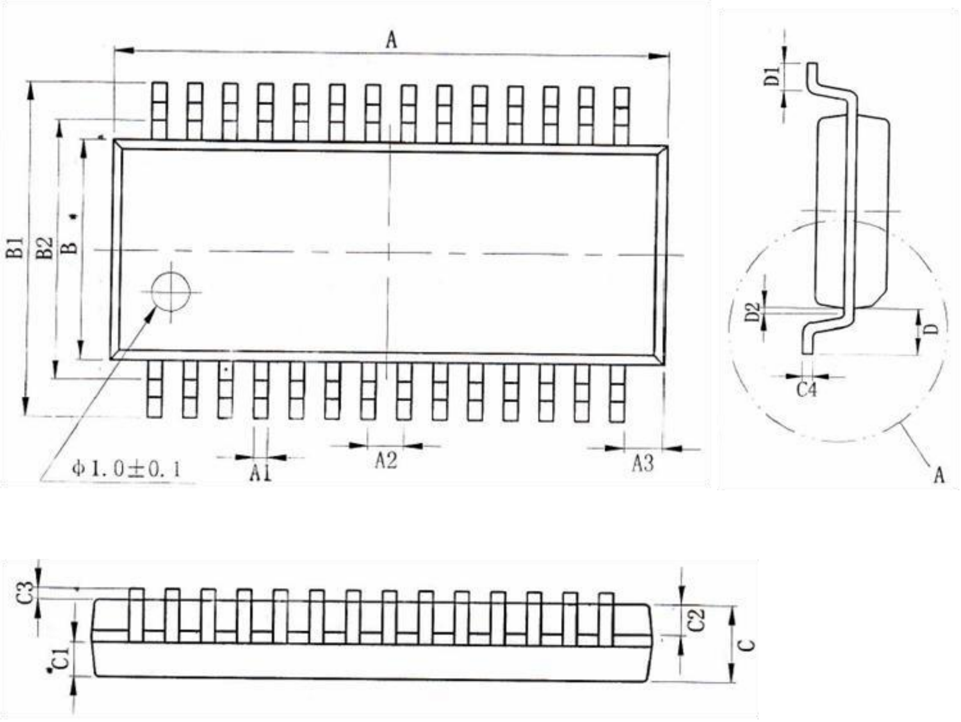
Package(1) SOP28





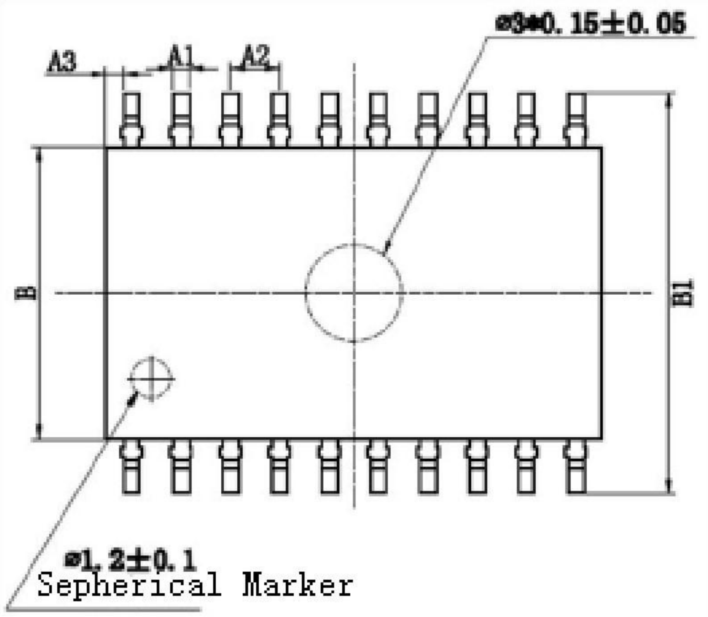
Sequence number	Minimum (mm)	Standard (mm)	Maximum (mm)
A	17.90	18.00	18.10
A1	0.356	0.40	0.456
A2	1.24	1.27	1.30
A3	---	0.542 TYP	---
B	7.40	7.50	7.60
B1	10.206	10.30	10.406
C	2.18	2.23	2.28
C1	0.938	1.0	1.038
C2	0.938	1.0	1.038
C3	0.03	0.09	0.17
D	1.353	1.40	1.453
C4	0.244	0.25	0.264

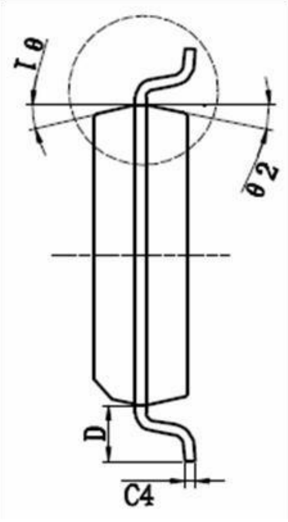
Package(2) SSOP28

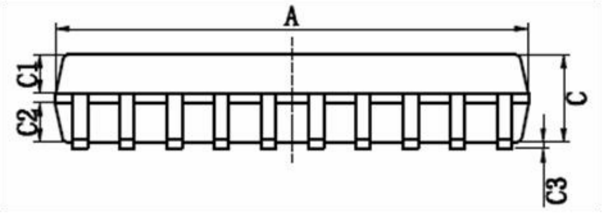


Sequence number	Minimum(mm)	Standard(mm)	Maximum(mm)
A	9.80	9.90	10.00
A1	---	0.254TYP	---
A2	---	0.635TYP	---
A3	---	0.695TYP	---
B	3.85	3.90	3.95
B1	5.85	6.00	6.24
B2	---	5.00TYP	---
C	1.40	1.50	1.60
C1	0.61	0.66	0.71
C2	0.54	0.59	0.64
C3	0.05	0.15	0.25
C4	0.203	0.215	0.233
D	---	1.05TYP	---
D1	0.40	0.55	0.70
D2	0.15	0.20	0.25

Package(3) SOP20

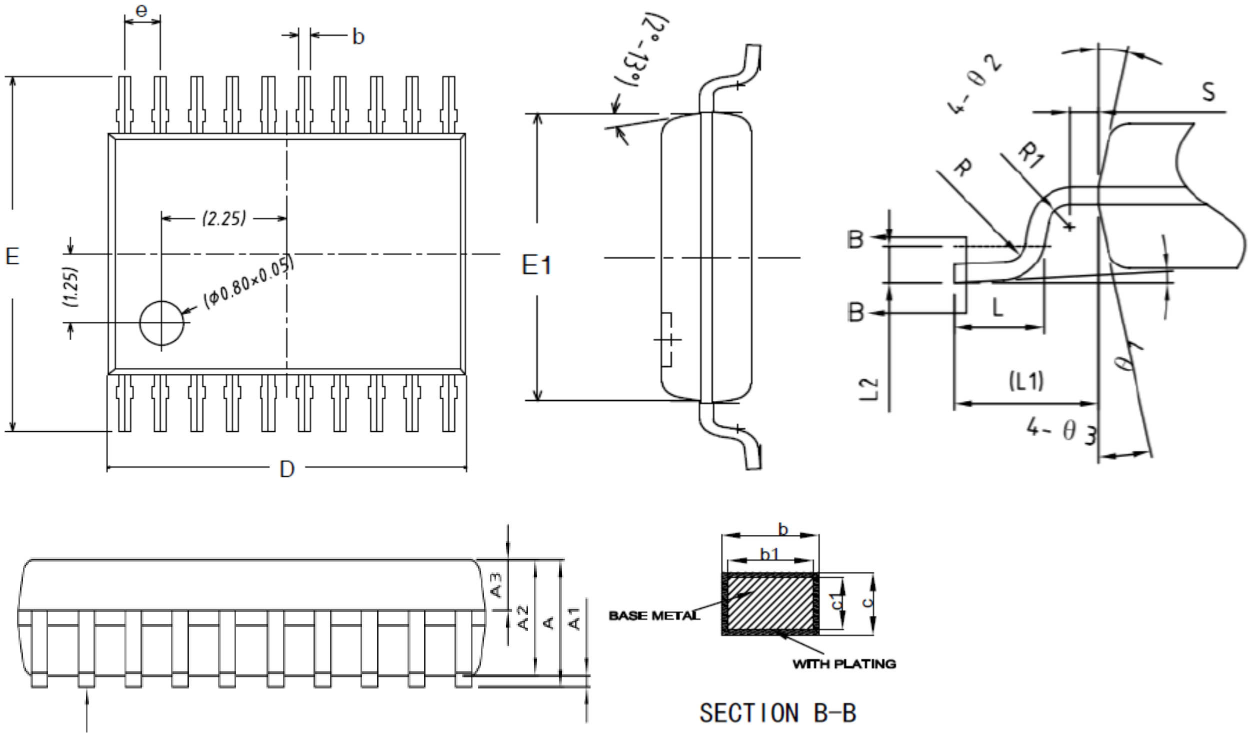






Sequence number	Minimum (mm)	Standard (mm)	Maximum (mm)
A	12.65	12.70	12.80
A1	0.381	0.40	0.431
A2	1.24	1.27	1.30
A3	0.45	0.455	0.46
B	7.40	7.50	7.60
B1	10.206	10.30	10.406
C	2.18	2.23	2.28
C1	0.938	1.0	1.038
C2	0.938	1.0	1.038
C3	0.145	0.175	0.205
D	1.353	1.40	1.453
C4	0.246	0.25	0.262

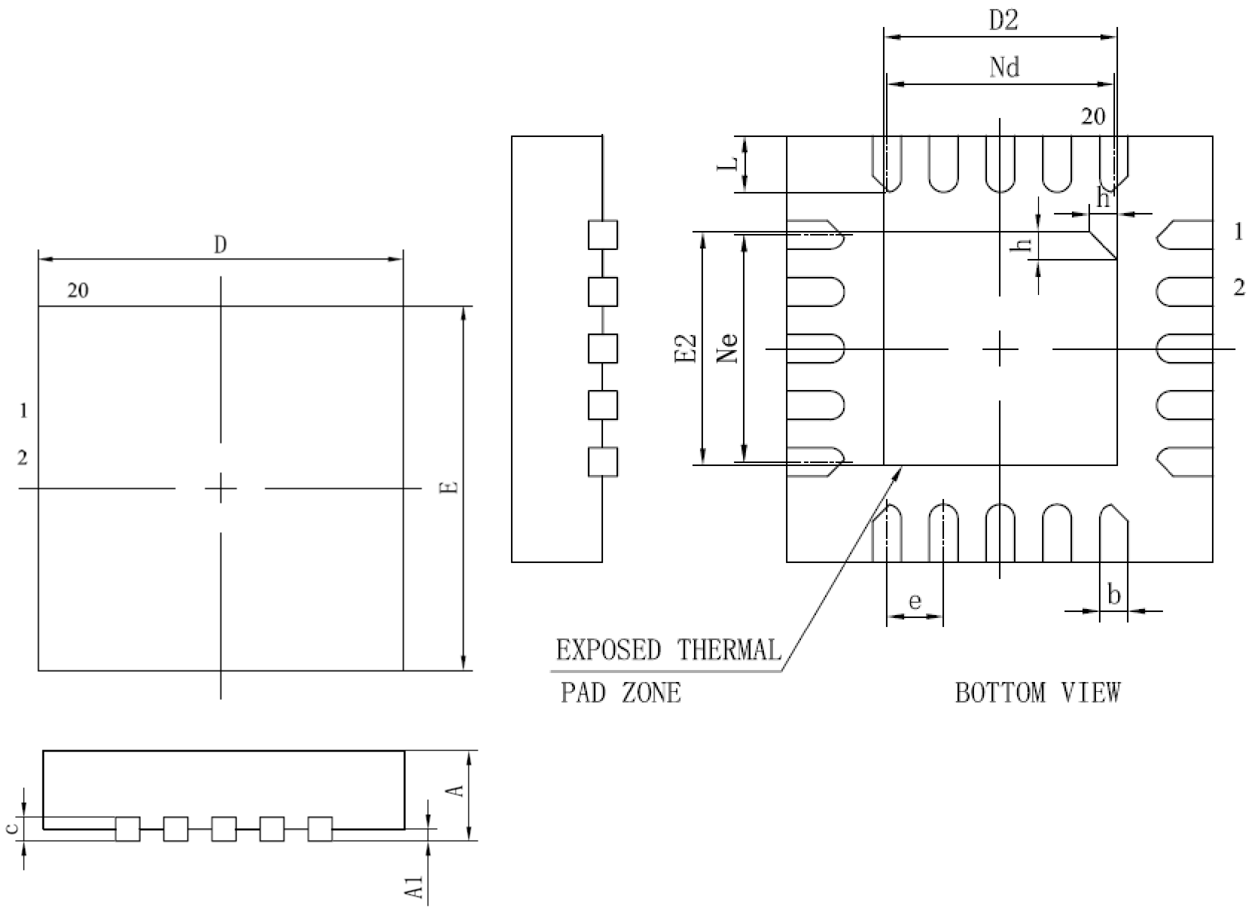
Package(4) TSSOP20



SECTION B-B

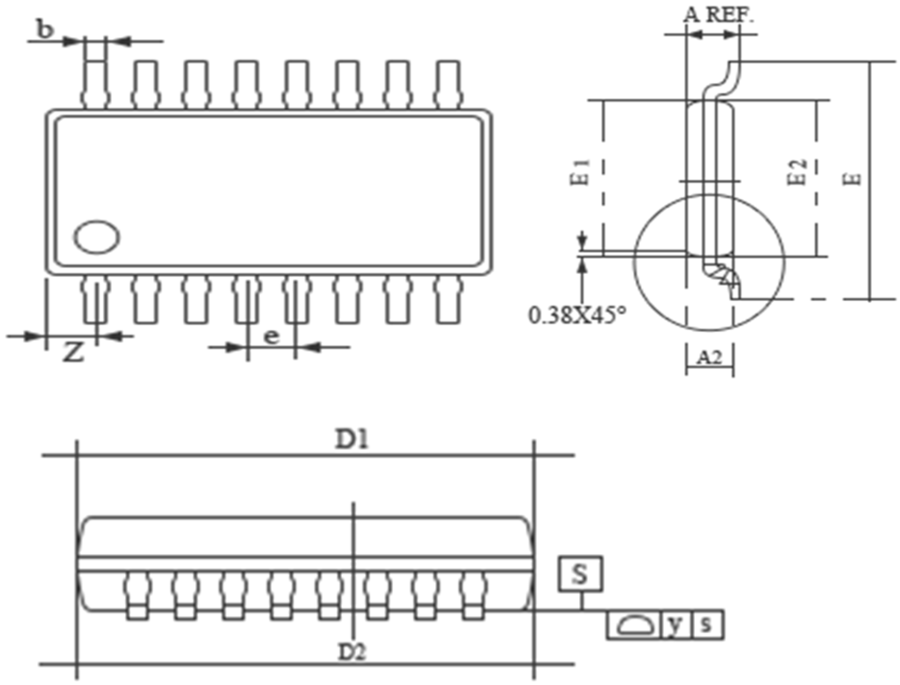
Sequence number	Minimum (mm)	Standard (mm)	Maximum (mm)
A	1.0	---	1.1
A1	0.05	---	0.15
A2	---	---	0.95
A3	0.39	---	0.40
b	0.20	0.22	0.24
c	0.10	---	0.19
c1	0.10		0.15
D	6.40	6.45	6.50
E	6.25	6.40	6.55
E1		4.35	4.40
L	0.50	0.60	0.70
e	0.55	0.65	0.75
L2	0.25BSC		
R	0.09		
L1	1.0REF		

Package(5) QFN20(3X3MM)



Sequence number	Minimum (mm)	Standard (mm)	Maximum (mm)
A	0.70	0.75	0.80
A1	---	0.02	0.05
b	0.15	0.20	0.25
c	0.18	0.20	0.25
D	2.90	3.00	3.10
D2	1.55	1.65	1.75
e	0.40BSC		
Ne	1.60BSC		
Nd	1.60BSC		
E	2.90	3.00	3.10
E2	1.55	1.65	1.75
L	0.35	0.40	0.45
h	0.20	0.25	0.30

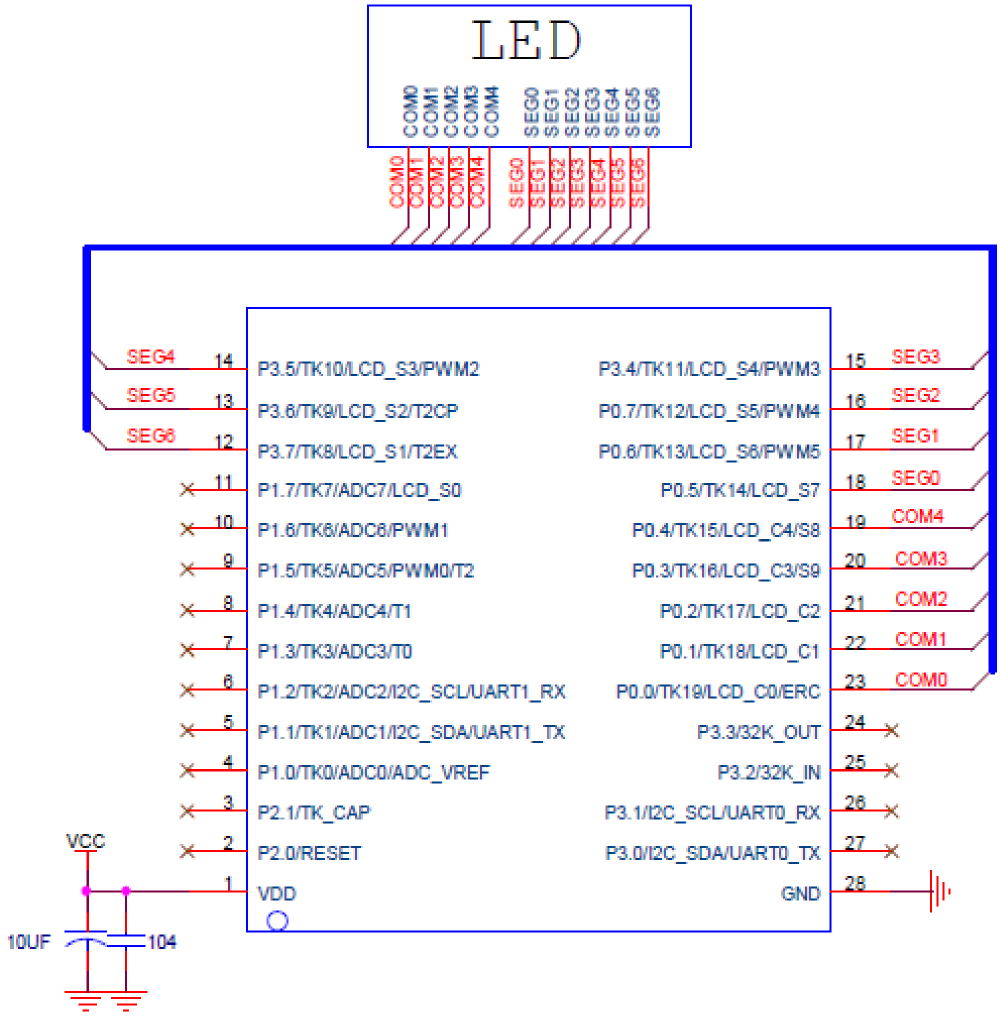
Package(6) SOP16



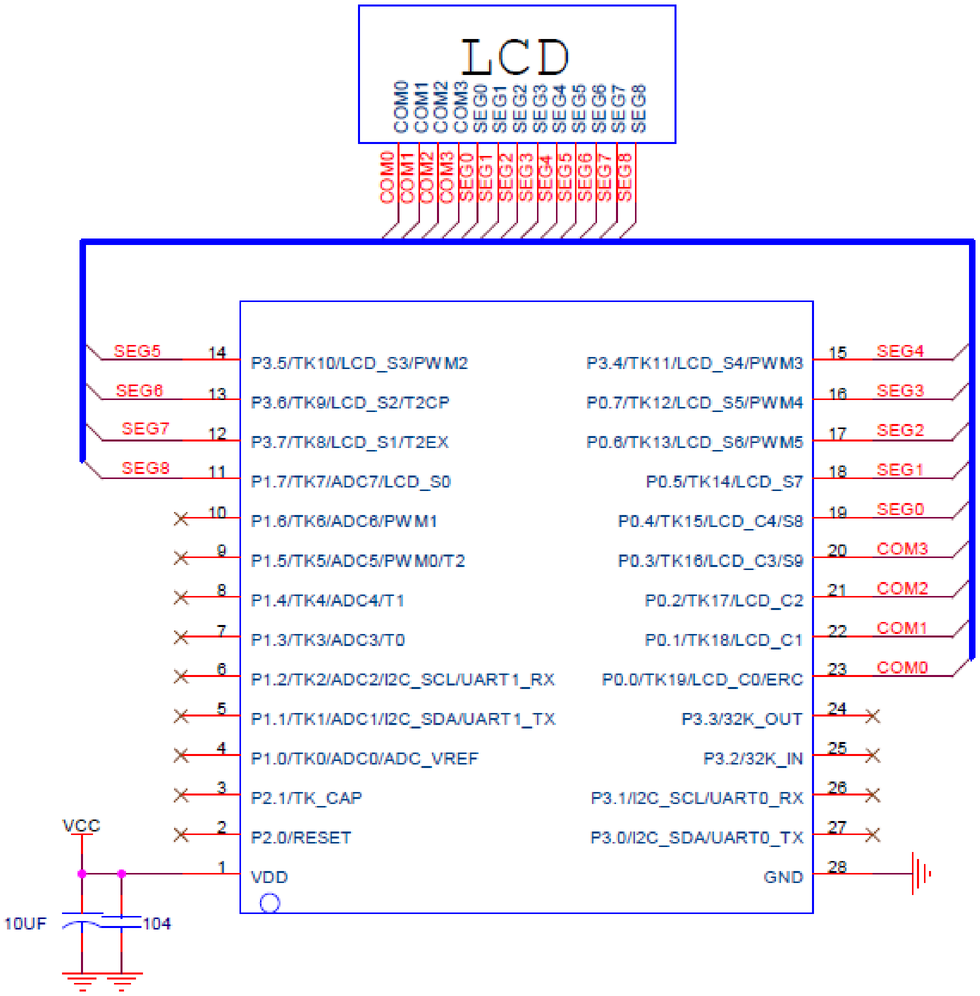
Sequence number	Minimum (mm)	Standard (mm)	Maximum (mm)
A	1.500	1.600	1.700
A2	1.400	1.450	1.500
b	0.356	0.406	0.456
D1	9.70	9.90	10.10
D2	9.75	9.95	10.15
E	5.90	6.000	6.100
E1	3.800	3.900	4.000
E2	3.850	3.950	4.050
e	————	1.27	————
Z	————	0.505	————

26 Typical Application Reference Circuit

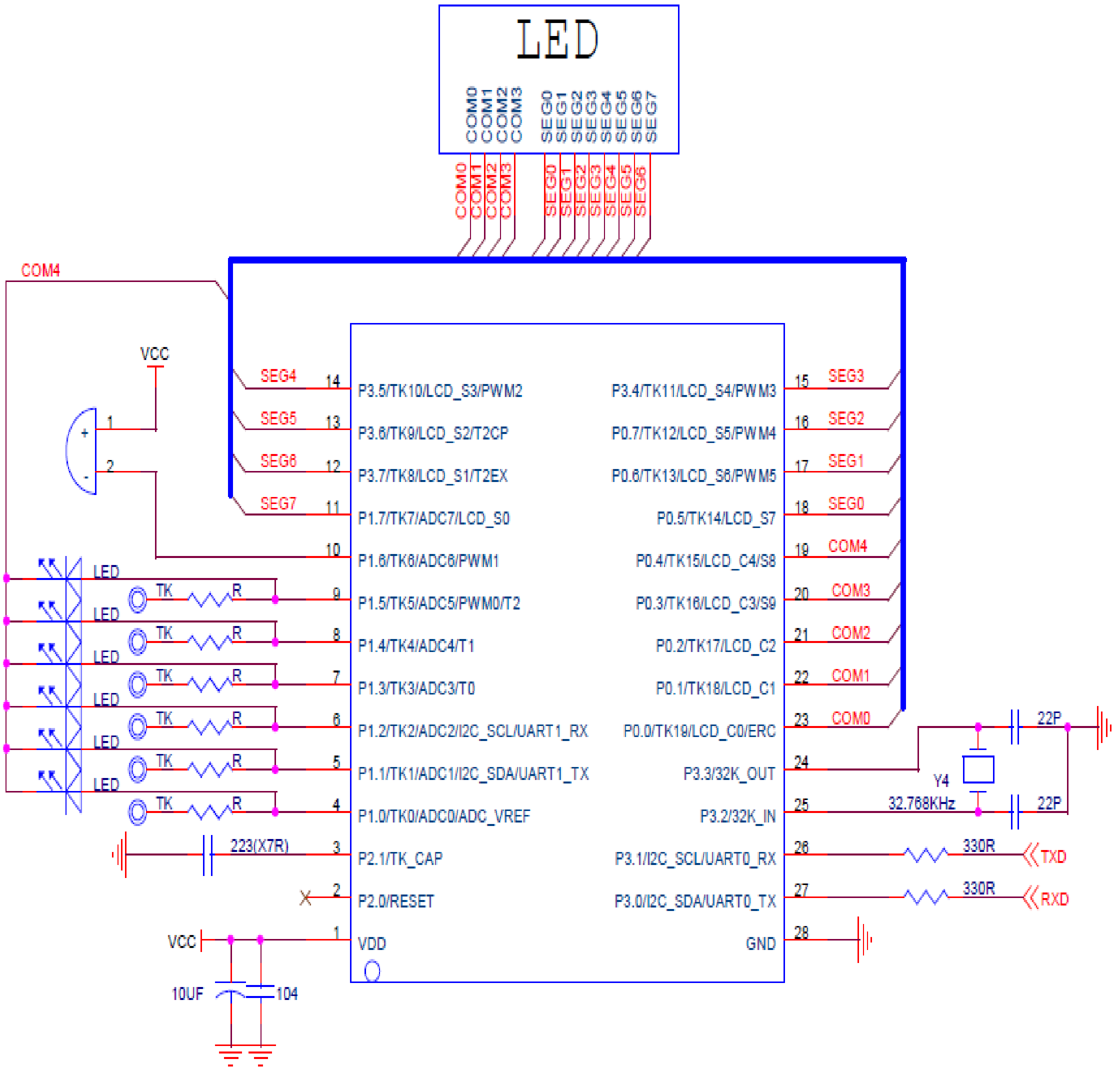
Reference Circuit(1)



Reference Circuit(2)



Reference Circuit(3)


LED
COM0 COM1 COM2 COM3 SEG0 SEG1 SEG2 SEG3 SEG4 SEG5 SEG6 SEG7
COM4
VCC
+
-
1
2
SEG4 14 P3.5/TK10/LCD_S3/PWM2
SEG5 13 P3.6/TK9/LCD_S2/T2CP
SEG6 12 P3.7/TK8/LCD_S1/T2EX
SEG7 11 P1.7/TK7/ADC7/LCD_S0
10 P1.6/TK6/ADC6/PWM1
9 P1.5/TK5/ADC5/PWM0/T2
8 P1.4/TK4/ADC4/T1
7 P1.3/TK3/ADC3/T0
6 P1.2/TK2/ADC2/I2C_SCL/UART1_RX
5 P1.1/TK1/ADC1/I2C_SDA/UART1_TX
4 P1.0/TK0/ADC0/ADC_VREF
3 P2.1/TK_CAP
2 P2.0/RESET
1 VDD
P3.4/TK11/LCD_S4/PWM3 15 SEG3
P0.7/TK12/LCD_S5/PWM4 16 SEG2
P0.6/TK13/LCD_S6/PWM5 17 SEG1
P0.5/TK14/LCD_S7 18 SEG0
P0.4/TK15/LCD_C4/S8 19 COM4
P0.3/TK16/LCD_C3/S9 20 COM3
P0.2/TK17/LCD_C2 21 COM2
P0.1/TK18/LCD_C1 22 COM1
P0.0/TK19/LCD_C0/ERC 23 COM0
P3.3/32K_OUT 24
P3.2/32K_IN 25
P3.1/I2C_SCL/UART0_RX 26
P3.0/I2C_SDA/UART0_TX 27
GND 28
LED
TK
R
223(X7R)
VCC
10UF
104
22P
22P
Y4
32.768KHz
330R
TXD
330R
RXD

27 Appendix

Appendix 1 Instruction Set Quick Reference Table

Mnemonic	Description	Description	Cycles
	DATA TRANSFER		
MOV A,Rn	Move register to A	(A) ← (Rn)	1
MOV A,direct	Move direct byte to A	(A) ← (direct)	1
MOV A,@Ri	Move indirect RAM to A	(A) ← ((Ri))	1
MOV A,#data8	Move 8-bit immediate data to A	(A) ← #data	1
MOV Rn,A	Move A to register	(Rn) ← (A)	1
MOV Rn,direct	Move direct byte to register	(Rn) ← (direct)	2
MOV Rn,#data8	Move 8-bit immediate data to register	(Rn) ← #data	1
MOV direct,A	Move A to direct byte	(direct) ← (A)	1
MOV direct,Rn	Move register to direct byte	(direct) ← (Rn)	2
MOV direct,direct	Move direct byte to direct byte	(direct) ← (direct)	2
MOV direct,@Ri	Move indirect RAM to direct byte	(direct) ← ((Ri))	2
MOV direct,#data8	Move 8-bit immediate data to direct byte	(direct) ← #data	2
MOV @Ri,A	Move A to indirect RAM	((Ri)) ← (A)	1
MOV @Ri,direct	Move direct byte to indirect RAM	((Ri)) ← (direct)	2
MOV @Ri,#data8	Move 8-bit immediate data to indirect RAM	((Ri)) ← #data	1
MOV DPTR,#data16	Load Data Pointer with 16-bit constant	(DPTR) ← #data116	2
MOV A,@A+DPTR	Move Code byte relative to DPTR to A	(A) ← ((A)) + (DPTR)	2
MOV A,@A+PC	Move Code byte relative to FFe to A	(PC) ← (PC) + 1 (A) ← ((A) + (PC))	2
MOVX A,@Ri	Move External RAM (8-bit addr) to A	(A) ← ((Ri))	2
MOVX A,@DPTR	Move External RAM (16-bit addr) to A	(A) ← ((DPTR))	2
MOVX @Ri,A	Move A to External RAM (8-bit addr)	((Ri)) ← (A)	2
MOVX @DPTR,A	Move A to External RAM (16-bit addr)	(DPTR) ← (A)	2
PUSH direct	Push direct byte onto stack	(SP) ← (SP) + 1 ((SP)) ← (direct)	2
POP DIRECT	Pop direct byte from stack	(direct) ← ((SP)) (SP) ← (SP) - 1	2
XCH A,Rn	Exchange register with A	(A) ↔ (Rn)	1
XCH A,direct	Exchange direct byte with A	(A) ↔ (direct)	1
XCH A,@Ri	Exchange indirect RAM with A	(A) ↔ ((Ri))	1
XCHD A,@Ri	Exchange low-order Digit indirect RAM with A	(A.3,...,A.0) ↔ ((Ri).3,...,(Ri).0)	1
SWAP A	Swap nibbles within A	(A.3,...,A.0) ↔ (A.7,...,A.4)	1
	ARITHMETIC OPERATIONS		
ADD A, Rn	Add register to A	(A) ← (A) + (Rn)	1
ADD A, direct	Add direct byte to A	(A) ← (A) + (direct)	1
ADD A, @Ri	Add indirect RAM to A	(A) ← (A) + ((Ri))	1
ADD A, #data8	Add 8-bit immediate data to A	(A) ← (A) + #data	1
ADDC A, Rn	Add register to A with Carry	(A) ← (A) + (C) + (Rn)	1
ADDC A, direct	Add direct byte to A with Carry	(A) ← (A) + (C) +	1

		(direct)	
ADDC A, @Ri	Add indirect RAM to A with Carry	(A) ← (A) + (C) + ((Ri))	1
ADDC A, #data8	Add 8-bit immediate data to A with Carry	(A) ← (A) + (C) + #data	1
SUBB A, Rn	Subtract register from A with Borrow	(A) ← (A) - (C) - (Rn)	1
SUBB A, direct	Subtract direct byte from A with Borrow	(A) ← (A) - (C) - (direct)	1
SUBB A, @Ri	Subtract indirect RAM from A with Borrow	(A) ← (A) - (C) - ((Ri))	1
SUBB A, #data8	Subtract immediate data from A with Borrow	(A) ← (A) - (C) - #data	1
INC A	Increment A	(A) ← (A) + 1	1
INC Rn	Increment register	(Rn) ← (Rn) + 1	1
INC direct	Increment direct byte	(direct) ← (direct) + 1	1
INC @Ri	Increment indirect RAM	((Ri)) ← ((Ri)) + 1	1
INC DPTR	Increment Data Pointer	(DPTR) ← (DPTR) + 1	2
DEC A	Decrement A	(A) ← (A) – 1	1
DEC Rn	Decrement register	(Rn) ← (Rn) - 1	1
DEC direct	Decrement direct byte	(direct) ← (direct) - 1	1
DEC @Ri	Decrement indirect RAM	((Ri)) ← ((Ri)) - 1	1
MUL AB	Multiply A & B (A × B => BA)	temp16 ← (A) X (B) (A)←(temp.7,temp.6,...,temp.0) (B)←(temp.15,temp.14,...,temp.8)	4
DIV AB	Divide A by B(A/B => A +B)	QUO ← (A) / (B) ......REM (A) ← QUO (B) ← REM	4

DA A	Decimal Adjust A	IF (A.3,...,A.0) > 9 || AC = 1
		THEN
		temp16 ← (A) + 0x06
		(A) ← (temp.7,...,temp.0)
		
		IF (temp16) > 0xFF
		THEN
		CY ← 1
		
		IF (A.7,...,A.4) > 9 || CY = 1
		THEN
		temp16 ← (A) + 0x60
		(A) ← (temp.7,...,temp.0)
			1

		IF (temp16) > 0xFF
		THEN
		CY ← 1	

LOGICAL OPERATIONS			
ANL A, Rn	AND register to A	(A) ← (A) & (Rn)	1
ANL A, direct	AND direct byte to A	(A) ← (A) & (direct)	1
ANL A, @Ri	AND indirect RAM to A	(A) ← (A) & ((Ri))	1
ANL A, #data8	AND 8-bit immediate data to A	(A) ← (A) & #data	1
ANL direct, A	AND A to direct byte	(direct) ← (direct) & (A)	1
ANL direct, #data8	AND 8-bit immediate data to direct byte	(direct) ← (direct) & #data	2
ORL A, Rn	OR register to A	(A) ← (A) | (Rn)	1
ORL A, direct	OR direct byte to A	(A) ← (A) | (direct)	1
ORL A, @Ri	OR indirect RAM to A	(A) ← (A) | ((Ri))	1
ORL A, #data8	OR 8-bit immediate data to A	(A) ← (A) | #data	1
ORL direct, A	OR A to direct byte	(direct) ← (direct) | (A)	1
ORL direct, #data8	OR 8-bit immediate data to direct byte	(direct) ← (direct) | #data	2
XRL A, Rn	Exclusive-OR register to A	(A) ← (A) ^ (Rn)	1
XRL A, direct	Exclusive-OR direct byte to A	(A) ← (A) ^ (direct)	1
XRL A, @Ri	Exclusive-OR indirect RAM to A	(A) ← (A) ^ ((Ri))	1
XRL A, #data8	Exclusive-OR 8-bit immediate data to A	(A) ← (A) ^ #data	1
XRL direct, A	Exclusive-OR A to direct byte	(direct) ← (direct) ^ (A)	1
XRL direct, #data8	Exclusive-OR 8-bit immediate data to direct byte	(direct) ← (direct) ^ #data	2
CLR A	Clear A	(A) ← 0	1
CPL A	Complement A	(A) ← /(A)	1
RL A	Rotate A Left	(A) ← (A.6,A.5,...,A.0,A.7)	1
RLC A	Rotate A Left through Carry	C ← A.7 (A) ← (A.6,A.5,...,A.0,C)	1
RR A	Rotate A Right	(A) ← (A.0,A.7,...,A.2,A.1)	1
RRC A	Rotate A Right through Carry	C ← A.0 (A) ← (C,A.7,...,A.2,A.1)	1
PROGRAM AND MACHINE CONTROL			
ACALL addr11	Absolute subroutine call	(PC) ← (PC) + 2 (SP) ← (SP) + 1 ((SP)) ← (PC7-0) (SP) ← (SP) + 1 ((SP)) ← (PC15-8) (PC10-0) ← page address	2
LACLL addr16	Long subroutine call	(PC) ← (PC) + 3 (SP) ← (SP) + 1	2

		((SP)) ← (PC7-0) ((SP)) ← (PC15-8) (PC) ←addr15-0	
RET	Return from subroutine	(PC15-8) ← ((SP)) (SP) ← (SP) - 1 (PC7-0) ← ((SP)) (SP) ← (SP) - 1	2
RETI	Return from interrupt	(PC15-8) ← ((SP)) (SP) ← (SP) - 1 (PC7-0) ← ((SP)) (SP) ← (SP) - 1	2
AJMP addr11	Absolute Jump	(PC) ← (PC) + 2 (PC10-0) ← page address	2
LJMP addr16	Long Jump	(PC) ← (PC) + 3 (SP) ← (SP) + 1 ((SP)) ← (PC7-0) (SP) ← (SP) + 1 ((SP)) ← (PC15-8) (PC10-0) ←addr15-0	2
SJMP rel	Short Jump (relative addr)	(PC) ← (PC) + 2 (PC) ← (PC) + rel	2
JMP @A+DPTR	Jump indirect relative to DPTR	(PC) ← (A) + (DPTR)	2

JZ rel	Jump if A is Zero	(PC) ← (PC) + 2
		IF (A) = 0
		THEN
		(PC) ← (PC) + rel
			2

JNZ rel	Jump if A is Not Zero	(PC) ← (PC) + 2
		IF (A) <> 0
		THEN
		(PC) ← (PC) + rel
			2

CJNE A, direct, rel	Compare direct to A & Jump if Not Equal	(PC) ← (PC) + 3
		IF (A) <> (direct)
		THEN
		(PC) ← (PC) + relative offset
		IF (A) < (direct)
		THEN
		(C) ← 1
		ELSE
		(C) ← 0
			2

CJNE A, #data8, rel	Compare 8-bit immediate to A & Jump if Not Equal	(PC) ← (PC) + 3
		IF (A) <> data
		THEN
		(PC) ← (PC) + relative offset
		IF (A) < data
		THEN
		(C) ← 1
		ELSE
		(C) ← 0
			2

CJNE Rn, #data8, rel	Compare 8-bit immediate to reg. & Jump if	(PC) ← (PC) + 3	2

	Not Equal	IF (Rn) <> data
		THEN
		(PC) ← (PC) + relative offset
		IF (Rn) < data
		THEN
		(C) ← 1
		ELSE
		(C) ← 0	

CJNE @Ri, #data8, rel	Compare 8-bit immediate to ind. & Jump if Not Equal	(PC) ← (PC) + 3
		IF ((Ri)) <> data
		THEN
		(PC) ← (PC) + relative offset
		IF ((Ri)) < data
		THEN
		(C) ← 1
		ELSE
		(C) ← 0
			2

DJNZ Rn, rel	Decrement register & Jump if Not Zero	(PC) ← (PC) + 2 (Rn) ← (Rn) - 1 IF (Rn) <> 0 THEN (PC) ← (PC) + rel	2
DJNZ direct, rel	Decrement direct byte & Jump if Not Zero	(PC) ← (PC) + 2 (direct) ← (direct) - 1 IF (direct) <> 0 THEN (PC) ← (PC) + rel	2
NOP	No operation	(PC) ← (PC) + 1	1
BOOLEAN VARIABLE MANIPULATION			
CLR C	Clear Carry flag	(C) ← 0	1
CLR bit	Clear direct bit	(bit) ← 0	1
SETB C	Set Carry flag	(C) ← 1	1
SETB bit	Set direct bit	(bit) ← 1	1
CPL C	Complement Carry flag	(C) ← /(C)	1
CPL bit	Complement direct bit	(bit) ← /(bit)	1
ANL C, bit	AND direct bit to Carry flag	(C) ← (C) & (bit)	2
ANL C, /bit	AND complement of direct bit to Carry flag	(C) ← (C) & /(bit)	2
ORL C, bit	OR direct bit to Carry flag	(C) ← (C) | (bit)	2
ORL C, /bit	OR complement of direct bit to Carry flag	(C) ← (C) | /(bit)	2
MOV C, bit	Move direct bit to Carry flag	(C) ← (bit)	1
MOV bit, C	Move Carry flag to direct bit	(bit) ← (C)	2
JC rel	Jump if Carry flag is set	(PC) ← (PC) + 2 IF (C) = 1 THEN (PC) ← (PC) + rel	2
JNC rel	Jump if No Carry flag	(PC) ← (PC) + 2 IF (C) = 0 THEN (PC) ← (PC) + rel	2
JB bit, rel	Jump if direct Bit is set	(PC) ← (PC) + 3 IF (bit) = 1 THEN (PC) ← (PC) + rel	2
JNB bit, rel	Jump if direct Bit is Not set	(PC) ← (PC) + 3 IF (bit) = 0 THEN (PC) ← (PC) + rel	2
JBC bit, rel	Jump if direct Bit is set & Clear bit	(PC) ← (PC) + 3	2

<table>
<tr><td></td><td></td><td>IF (bit) = 1 THEN
(bit) ← 0
(PC) ← (PC) + rel</td><td></td></tr>
<tr><td colspan="4">Pseudo-instruction</td></tr>
<tr><td>ORG</td><td colspan="3">Set program start address</td></tr>
<tr><td>END</td><td colspan="3">Mark the end of source code</td></tr>
<tr><td>EQU</td><td colspan="3">Define constants</td></tr>
<tr><td>SET</td><td colspan="3">Define integer numbers</td></tr>
<tr><td>DATA</td><td colspan="3">Assign a value to the data address</td></tr>
<tr><td>BYTE</td><td colspan="3">Assigning values to byte type symbols</td></tr>
<tr><td>WROD</td><td colspan="3">Assigning values to word type symbols</td></tr>
<tr><td>BIT</td><td colspan="3">Name the address of the bit</td></tr>
<tr><td>ALTNAME</td><td colspan="3">Replace reserved words with custom names</td></tr>
<tr><td>DB</td><td colspan="3">Load a contiguous block of memory with byte-type data</td></tr>
<tr><td>DW</td><td colspan="3">Load a contiguous block of memory with word data</td></tr>
<tr><td>DS</td><td colspan="3">Set aside a contiguous storage area or load specified bytes</td></tr>
<tr><td>INCLUDE</td><td colspan="3">Insert a source file into the program</td></tr>
<tr><td>TITLE</td><td colspan="3">Add a header row to the list file</td></tr>
<tr><td>NOLIST</td><td colspan="3">No list file is generated during assembly</td></tr>
<tr><td>NOCODE</td><td colspan="3">When the condition is compiled, the list is not generated if the condition is false</td></tr>
</table>