CACHIP *Built - in 16-bit PMW / Touch Key / 1T 8051 8K Flash MCU*

CA51F5 Series MCU
User Guide

REV 2.2

Table of Contents

1 Introduction

CA51F5 series is 8-bit MCU based on 1T 8051 core and operates 10 times faster than traditional 8051 chips with definitely better performance. The Flash program memory embedded can be programmed for times and offers users with 8K FLASH which brings great convenience to software development. Not only traditional 8051 chip features, CA51F5 also includes Touch Key, 16-bit PMW, 5-bit DAC, UART, I^{2}C, RGB_LED controller, LVD and other functional modules. It can operate in three Power Save Modes (IDLE/STOP/LOW SPEED) in order to meet different power consumption needs. With great functions and anti-jamming feature, CA51F5 could be used in various fields such as all kinds of household lighting, family audio touch control system, household appliance touch control system, Bluetooth stereo, table lamp and bathroom mirror lamp, landscape and atmosphere lamp belt 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
- Memory
  - Flash : 8K 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, 512 bytes external RAM
- Operating Voltage
  - Operating Voltage： 1.8V - 5.5V
- Clock System
  - Internal Low Speed RC Oscillator： 131KHz
  - Internal High Speed RC Oscillator： 16MHz, with 2% precision (The factory original frequency is 3.6864MHz@3.3V/25℃)
- TMC Function

  - The clock source is the low speed internal RC oscillator and the minimum time unit for interrupt is 512 clock cycle periods of it.
  - Configurable interrupt time ranges from 1 to 256 minimum time units.

- Interrupt System
  - 7 effective interrupt sources
  - Two levels for interrupt priority which also supports interrupt nesting
  - 5 external interrupt source. For 3 of them, any of the signal pin could be configured as interrupt pin
- Timer
  - Three 16-bit general Timers: Timer 0, Timer 1, Timer 2
- General Purpose IO (GPIO)
  - Supports 14 GPIO at most
  - Supports push-pull, open-drain, strong pull-up, weak pull-up, strong pull-down, weak pull-down and high-impedance modes
  - Different driving strength and turnover speed can be set in push-pull mode
- Touch Key
  - Internal Touch Sensor Controller
  - Supports 13 touch channels at most
  - Touch to set the internal charging and internal reference which can effectively suppress the low-frequency interference of the power source
  - Internal waterproof compensation mechanism
  - Excellent anti-jamming performance which conforms to EMC(CS) Standard
  - Support touch power saving mode
- PWM
  - Supports 6 channel PWM, any periods or duty cycles are configurable in 16 bits
  - Supports to output internal clock directly
  - Supports PWM Interrupt
  - Supports 2 cascade LED driver, with scanning frequency more than 400Hz and data transmission speed 800Kbps
  - Support to control WS2812 or similar driver chips, which meet the requirement for monochromatic or colorful LED strip

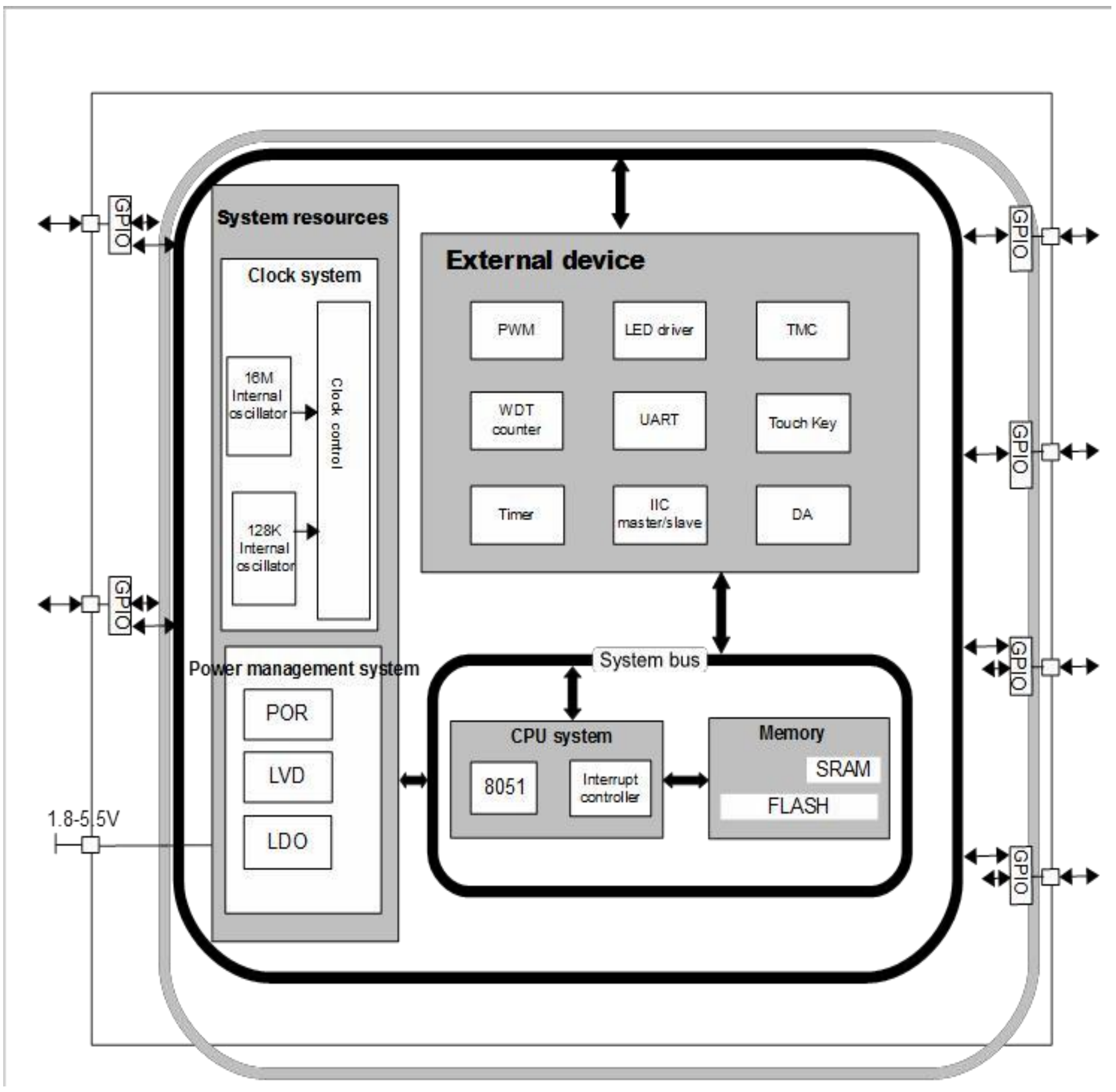
- Low Voltage Detector(LVD)
  - Configurable voltage detection range 1.7 - 4.8V
  - Low voltage reset/interrupt configurable
  - Choose to detect VDD voltage or pin input voltage
- DAC function
  - Support two 5-bit DAC output channels, 32 levels for voltage configurable for each channel
- Reset Mode
  - Supports variable reset sources： Hard Reset, Soft Reset, Watch Dog Reset, LVD Reset, Power On/Down Reset
- Watch Dog
  - 27bit Watch Dog Timer, 16 bit precision configurable, with Watch Dog Reset and Interrupt configurable as well
- UART
  - Supports 1 UART port at most
  - Supports 1 byte receive buffer
- **I^2C**
  - One I^2C port embedded which supports Master-Slave mode and Standard/Fast/High Speed mode as well
  - I^2C can set digital filtering to enhance I^2C anti-interference performance
- Program Download and Simulation
  - Supports ISP and IAP
  - Supports simulation online
- Low power consumption
  - For STOP Mode, current<6uA
  - For IDLE Mode, current<13uA
  - For Low Speed Mode, current<20uA
- Package Type：SOP16/MSOP10/DFN8/SOP8

3 Chip Model and Function Description

Table 3-1 CA51F5 Specific Models and Their Features

Models	Flash Storage [BYTE]	External Ram[BYTE]	Internal High Speed Crystal Oscillator	External Low Speed Crystal Oscillator	GPIO	UART	I^2C	16 bit PWM Channels	Touch Key	5-bit D/A	Cascade LED Driver	ISP	Simulation On Chip	Working Voltage[V]	Package Type
CA51F551S1	8K	512	√	√	6	1	√	3	5	1	1	√	√	1.8-5.5	SOP8
CA51F551M2	8K	512	√	√	8	1	√	5	7	1	2	√	√	1.8-5.5	MSOP10
CA51F551N1	8K	512	√	√	6	1	√	3	5	1	1	√	√	1.8-5.5	DFN8L 2X2MM
CA51F551S3	8K	512	√	√	14	1	√	6	13	2	2	√	√	1.8-5.5	SOP16

4 Block Diagram



5 Pin Package and Description

5.1 Package Definition

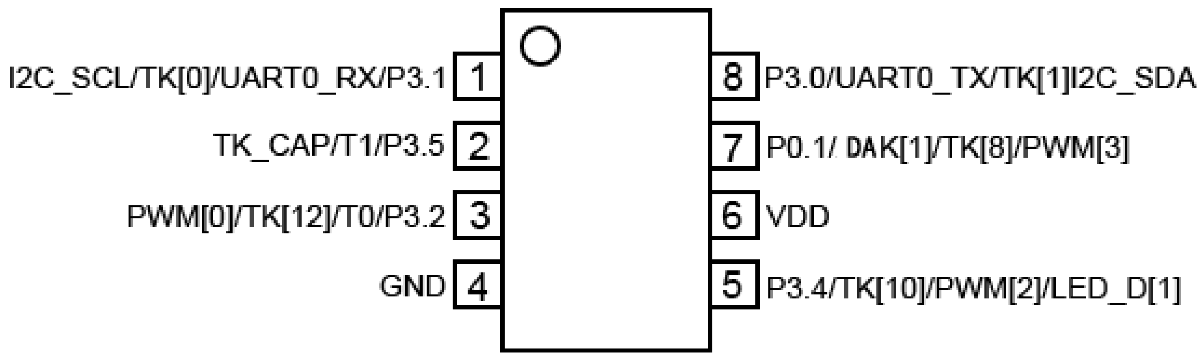

I2C_SCL/TK[0]/UART0_RX/P3.1
1
TK_CAP/T1/P3.5
2
PWM[0]/TK[12]/T0/P3.2
3
GND
4
8
P3.0/UART0_TX/TK[1]I2C_SDA
7
P0.1/ DAK[1]/TK[8]/PWM[3]
6
VDD
5
P3.4/TK[10]/PWM[2]/LED_D[1]

Figure 5-1-1 SOP8 Package

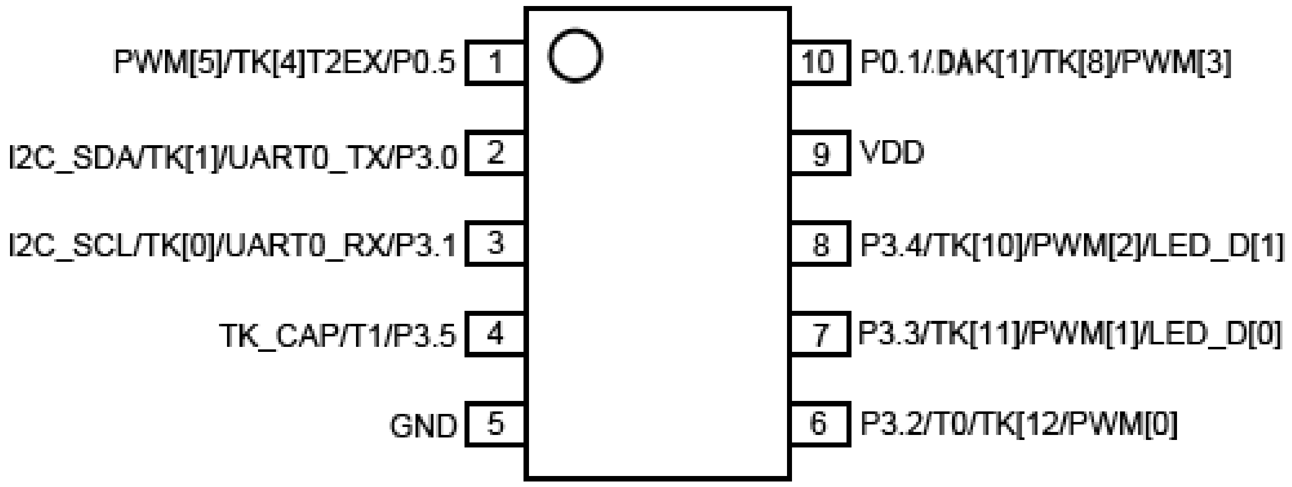

PWM[5]/TK[4]T2EX/P0.5
1
I2C_SDA/TK[1]/UART0_TX/P3.0
2
I2C_SCL/TK[0]/UART0_RX/P3.1
3
TK_CAP/T1/P3.5
4
GND
5
10
P0.1/DAK[1]/TK[8]/PWM[3]
9
VDD
8
P3.4/TK[10]/PWM[2]/LED_D[1]
7
P3.3/TK[11]/PWM[1]/LED_D[0]
6
P3.2/T0/TK[12/PWM[0]

Figure 5-1-2 MSOP10 Package

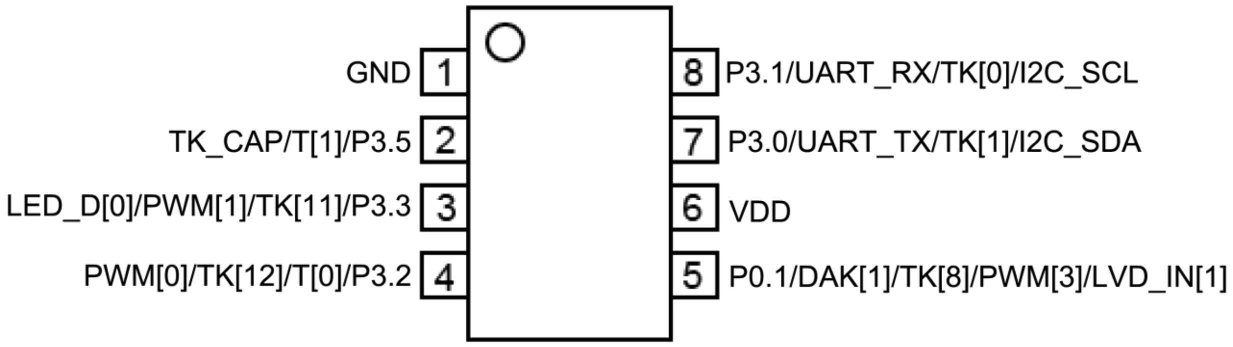

GND
1
TK_CAP/T[1]/P3.5
2
LED_D[0]/PWM[1]/TK[11]/P3.3
3
PWM[0]/TK[12]/T[0]/P3.2
4
8
P3.1/UART_RX/TK[0]/I2C_SCL
7
P3.0/UART_TX/TK[1]/I2C_SDA
6
VDD
5
P0.1/DAK[1]/TK[8]/PWM[3]/LVD_IN[1]

Figure 5-1-3 DFN8L(2x2mm) Package

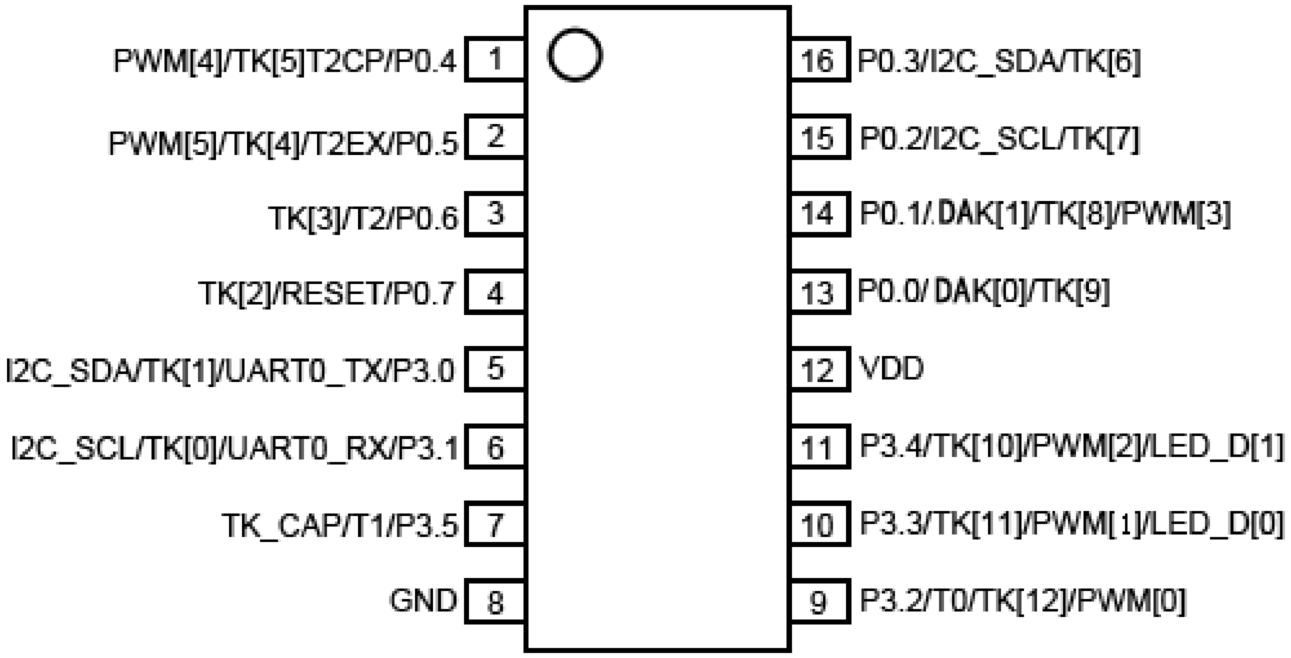


Figure 5-1-4 SOP16 Package

5.2 Pin Description

Table 5-2-1 Pin Description

Pin Sequence Number				Pin Name	Pin Function	Default Function
SOP16	MSOP10	DFN8	SOP8			
1	-	-	-	P0.4/T2CP/TK[5]/PWM[4]	General bi-directional I/O port Timer T2CP input Touch key channel input PWM output	General bi-directional I/O port
2	1	--	-	P0.5/T2EX/TK[4]/PWM[5]	General bi-directional I/O port Timer T2EX input Touch key channel input PWM output	General bi-directional I/O port
3	-	--	-	P0.6/T2/TK[3]	General bi-directional I/O port Timer T2 input Touch key channel input	General bi-directional I/O port
4	-	--	-	P0.7/RESET/TK[2]	General bi-directional I/O port Hard reset pin Touch key channel input	hard reset pin
5	2	7	8	P3.0/I2C_SDA/TK[1]/UART0_TX	General bi-directional I/O port I2C data transmit port Touch key channel input UART0_TX port	I2C data transmit port
6	3	8	1	P3.1/I2C_SCL/TK[0]/UART0_RX	General bi-directional I/O port I2C clock transmission port Touch key channel input UART0_RX port	I2C clock transmission port
7	4	2	2	P3.5/T1/TK_CAP	General bi-directional I/O Timer T1 input Input for external capacitor of TK	General bi-directional
8	5	1	4	GND	Chip Ground	Ground

9	6	4	3	P3.2/T0/TK[12]/PWM[0]	General bi-directional I/O port Timer T0 input Touch key channel input PWM output	General bi-directional I/O port
10	7	3	-	P3.3/TK[11]/PWM[1]/LED_D0	General bi-directional I/O port Touch key channel input PWM output LED Scan control port	General bi-directional I/O port
11	8	--	5	P3.4/TK[10]/PWM[2]/LED_D1	General bi-directional I/O port Touch key channel input PWM output LED Scan control port	General bi-directional I/O port
12	9	6	6	VDD	Chip power	VDD
13	-	--	-	P0.0/DAK[0]/TK[9]	General bi-directional I/O port DAC KEY port Touch key channel input	General bi-directional I/O port
14	10	5	7	P0.1/DAK[1]/TK[8]/PWM[3]	General bi-directional I/O port DAC KEY port Touch key channel input PWM output	General bi-directional I/O port
15	-	-	-	P0.2/I2C_SCL/TK[7]	General bi-directional I/O port I2C clock transmission port Touch key channel input	General bi-directional I/O port
16	-	-	-	P0.3/I2C_SDA/TK[6]	General bi-directional I/O port I2C data transmit port Touch key channel input	General bi-directional I/O port

Note：For signal pin’s alternate function settings, please refer to Table 15-2-3 and Table 15-2-5

6 Central Processing Unit(CPU)

6.1 CPU Introduction

The core of CA51F5 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 selector, 0 for DPTR0, 1 for DPTR1
0	P	Parity flag 0: the number of 1 in ACC is even 1: the number of 1 in ACC in odd

Table 6-2-9 Register SPMAX

8100H	7	6	5	4	3	2	1	0
SPMAX	SPMAX[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

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)

CA51F5 series offers both internal RAM(256 bytes) and external RAM(512 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.
- 512 bytes external RAM(address：0000H ~ 01FFH) supports indirect addressing by using MOVX.

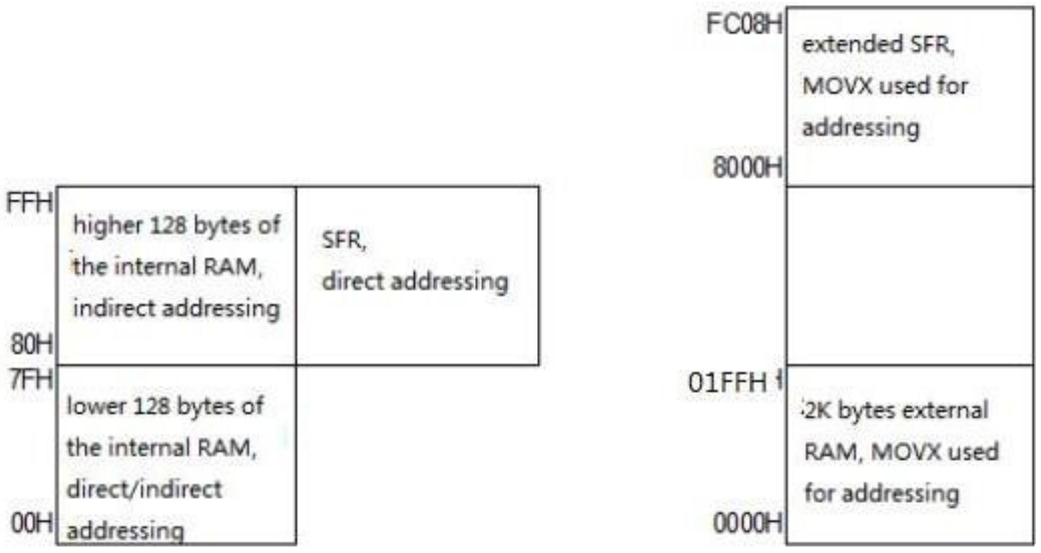

FFH
higher 128 bytes of the internal RAM, indirect addressing
SFR, direct addressing
80H
7FH
lower 128 bytes of the internal RAM, direct/indirect addressing
00H
FC08H
extended SFR, MOVX used for addressing
8000H
01FFH
2K bytes external RAM, MOVX used for addressing
0000H

Figure 7-1-1 RAM Architecture

7.2 Special Function Register(SFR)

The SFR architecture of CA51F5 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	TKCON	TKCFG	TKMTS	TKIF	TK0CHS	TK1CHS	TK2CHS	TK3CHS
F0H	B	TK4CHS	TK5CHS	ATK0CL	ATK0CH	ATK1CL	ATK1CH	ATK2CL
E8H	LVDCON	ATK2CH	ATK3CL	ATK3CH	ATK4CL	ATK4CH	ATK5CL	ATK5CH
E0H	ACC	TK0MSL	TK0MSH	TK1MSL	TK1MSH	TK2MSL	TK2MSH	TK3MSL
D8H	UDCKS	TK3MSH	TK4MSL	TK4MSH	TK5MSL	TK5MSH	AK0CON	AK1CON
D0H	PSW	EP0CON	EP1CON	EP2CON	EPIF	TMCON	TMSNU	LEDAT0
C8H	T2CON	T2MOD	T2CL	T2CH	TL2	TH2	LEWTML	LEWTMH
C0H	I2CCON	I2CADR	I2CADM	I2CCCR	I2CDAT	I2CSTA	I2CFLG	LEDAT1
B8H	IP	PWM0CON	PWM1CON	PWM2CON	PWM3CON	PWM4CON	PWM5CON	LEFLG
B0H	P3	PWM0CKD	PWM1CKD	PWM2CKD	PWM3CKD	PWM4CKD	PWM5CKD	PWMIF
A8H	IE	PWM0DIVL	PWM0DIVH	PWM1DIVL	PWM1DIVH	PWM2DIVL	PWM2DIVH	PWM3DIVL
A0H	WDCON	WDFLG	WDVTHL	WDVTHH	PWM3DIVH	PWM4DIVL	PWM4DIVH	PWM5DIVL
98H	S0CON	S0BUF	PWM5DIVH	PWM0DUTL	PWM0DUTH	PWM1DUTL	PWM1DUTH	PWM2DUTL
90H	PWMEN	PWM2DUTH	PWM3DUTL	PWM3DUTH	PWM4DUTL	PWM4DUTH	PWM5DUTL	PWM5DUTH
88H	TCON	TMOD	TL0	TL1	TH0	TH1	IDLST	STPST
80H	P0	SP	DP0L	DP0H	DP1L	DP1H	PWCON	PCON

Due to limited SFR address space, CA51F5 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
8018H	P30F	P31F	P32F	P33F	P34F	P35F	-	-
8030H	CKCON	CKDIV	IHCFG	ILCFGL	ILCFGH	-	-	-
8050H	ATK0NL	ATK0NH	ATK1NL	ATK1NH	ATK2NL	ATK2NH	ATK3NL	ATK3NH
8058H	ATK4NL	ATK4NH	ATK5NL	ATK5NH	TKMAXF	TKMINF	-	-
8060H	LEDUTL	LEDUTH	-	-	-	-	-	-
8100H	SPMAX	I2CIOS	TKCKS	TKPWC	-	LVDCFG	TLEN0	TLDAT0
8108H	TLCON	TLFLG	TLCKS	TLCNTKL	TLCNTKH	TLCNTLL	TLCNTLH	TLDIV
8110H	TLCOMS	TLEN1	TLDAT1	-	-	-	-	-
8118H	-	-	-	-	-	-	-	-
8120H	P00C	P01C	P02C	P03C	P04C	P05C	P06C	P07C
8138H	P30C	P31C	P32C	P33C	P34C	P35C	-	-
FC00H	MECON	FSCMD	FSDAT	LOCK	PADRD	PTSL	PTSH	-

7.3 Flash Memory

7.3.1 Function Introduction

Flash memory is 8K byte 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 128 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.

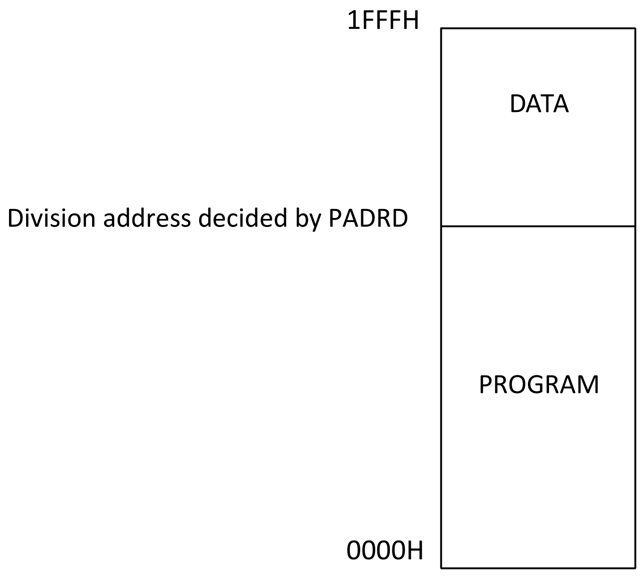


Figure 7-3-1 8K Flash Memory Structure

7.3.3 Flash Description

Table 7-3-3-1 Register MECON

FC00H	7	6	5	4	3	2	1	0
MECON	-	DPSTB	-	-		-	BOOT	
R/W	-	R/W					R/W	
Initial Value	-	0	-	-		-	0	0

Bit number	Bit symbol	Description
7		
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.*
5~1	-	-
0	BOOT	Programs start area control after soft reset 0: Program starts from FLASH after soft reset 1: Program starts from XRAM after soft reset

Table 7-3-3-2 Register FSCMD

FC01H	7	6	5	4	3	2	1	0
FSCMD	-	-	-	-	-	CMD[2:0]		
R/W	-	-	-	-	-	R/W		
Initial Value	-	-	-	-	-	0	0	0

Bit number	Bit symbol	Description
7~3	-	-
2~0	CMD	Command register 000: No operations 100: Erase the whole Flash 001: Read Flash DATA area 010: Write Flash DATAarea 011: Erase one sector of the Flash DATA area 101: Read Flash PROGRAM area 110: Write Flash PROGRAM area 111: Erase one sector of the Flash PROGRAM area *Note*: *1. CMD will be cleared automatically after erasure command executed*

		2. CMD remains unchanged after R/W commands and the R/W operations will be done by reading/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	0	0	0	0	0	0	0	0
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		REPE			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								
7~4	-							
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						
0	-	-						

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	0	1	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 *Note :* *1.PADRD=0 indicates the whole Flash is DATA area* *2.The maximum value for PADRD is 40H. 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[12:8]				
R/W	-	-	-	R/W				
Initial Value	-	-	-	0	0	0	0	0

Bit number	Bit symbol	Description
15~13	-	
12~0	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 8K Flash 128 byte DATA area and the remains for PROGRAM area, the program may like this:

```
----------------------------------------------------------------------------------------
PADRD = 63; //The address for PROGRAM area will be 0~0x7BFF while the address for DATA area will be
0x1F80~0x1FFF
----------------------------------------------------------------------------------------
```

__Note:__ This makes the physical address of the DATA area in FLASH 0x1F80~0x1FFF while the logical address is 0x0000~0x007F. 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
---------------------------------------------------------------------------------------
```

Note： sector number n=0、1、2……。

- **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
}
FSCMD = 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 read 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 512 byte external RAM can be mapped as PROGRAM area as well and the mapping address is 2000H~21FFH. Users may download the program to external RAM. When program is running,It jump to mapping program area to execute. Similarly, users can set BOOT(please refer to register MECON) to 1, and then soft reset. The program starts from external RAM (the mapping address is 0000H~01FFH). Mapping program area offers convenience for IAP and so on.

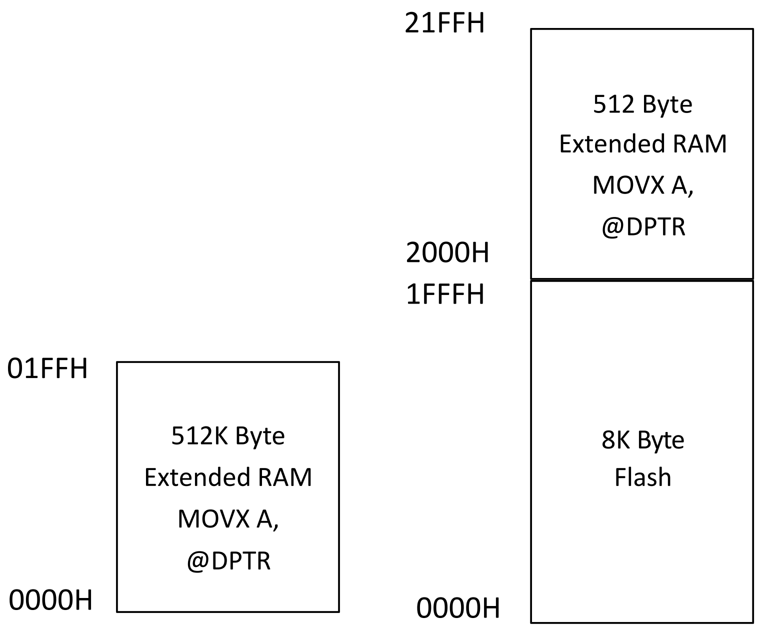

21FFH
512 Byte Extended RAM MOVX A, @DPTR
2000H
1FFFH
01FFH
512K Byte Extended RAM MOVX A, @DPTR
0000H
8K Byte Flash
0000H

Figure 7-4-1 XRAM Address mapping

8 Interrupt System

8.1 Function Introduction

CA51F5 Series include a enhanced interrupt control system with 7 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

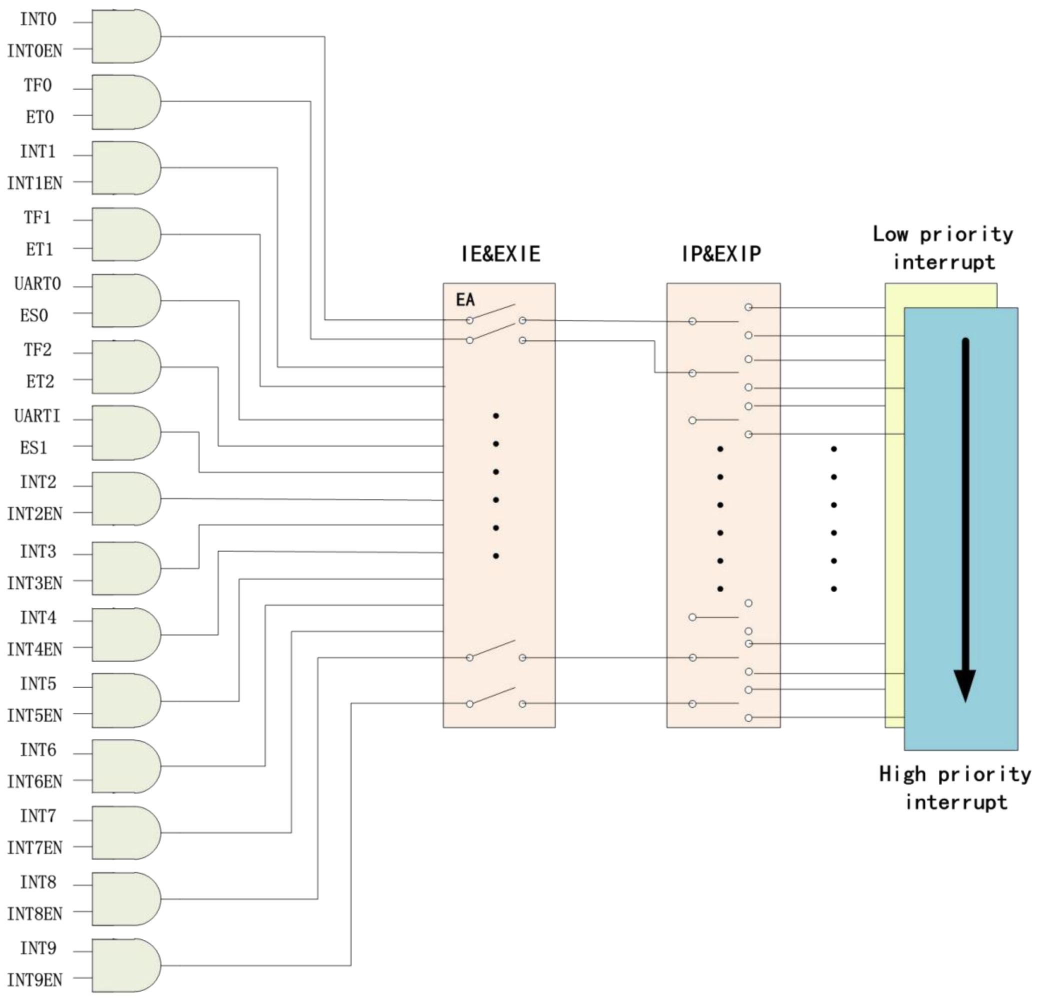


Figure 8-2-1 Interrupt Logic

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
INT2	UART0/External Interrupt 2/PWM Interrupt	23H	4
INT3	Timer2/External Interrupt 3/TK Interrupt/TMC Interrupt	2BH	5
INT4	External Interrupt 4/WDT Interrupt/I2C Interrupt/ LVD Interrupt	33H	6

8.4 Interrupt Control Register

Table 8-4-1 Register IE

A8H	7	6	5	4	3	2	1	0
IE	EA	INT4EN	INT3EN	INT2EN	ET1	INT1EN	ET0	INT0EN
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 1: enable
6	INT4EN	Interrupt 4 enable control(Interrupt 4 is used for LVD/External Interrupt 4) 0: Disable 1: Enable
5	INT3EN	Interrupt 3 enable control(Interrupt 3 is used for UART2/TK/External Interrupt 3) 0: Disable 1: Enable
4	INT2EN	Interrupt 2 enable control(Interrupt 2 is used for ADC/External Interrupt 2) 0: Disable 1: Enable
3	ET1	Timer 1 Interrupt enable control 0: disable Timer 1 Interrupt 1: enable Timer 1 Interrupt

2	INT1EN	External Interrupt 1 enable control 0: disable 1: enable
1	ET0	Timer 0 Interrupt enable control
0	INT0EN	Interrupt 0 enable control bit (interrupt 0 is used for external interrupt 0) 0: disable 1: enable

Table 8-4-2 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	Interrupt INT4 priority control 0: low priority 1: high priority
5	PT2	Interrupt INT3 priority control 0: low priority 1: high priority
4	PS0	Interrupt INT2 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

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 3 extended Interrupt Entries INT2~INT4 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~INT4 External Interrupt is EPIF and the corresponding configuration register for INT2~INT4 are EPCON0~EPCON2.

Note: INT0 and INT1 can be triggered by rising edge or falling edge and the selection bits are IT0 and IT1 respectively. Refer to the Register TCON for details.

8.5.2 External Interrupt Register

Table 8-5-1 Register EPIF

D4H	7	6	5	4	3	2	1	0
EPIF	-	-	-	-	-	EPIF2	EPIF1	EPIF0
R/W	-	-	-	-	-	R/W	R/W	R/W
Initial Value	-	-	-	-	-	0	0	0

Bit number	Bit symbol	Description
7~3		
2	EPIF2	External Interrupt 4 Interrupt Flag, cleared when 1 is written to it
1	EPIF1	External Interrupt 3 Interrupt Flag, cleared when 1 is written to it
0	EPIF0	External Interrupt 2 Interrupt Flag, cleared when 1 is written to it

Table 8-5-2 Register EPCON

D1H	7	6	5	4	3	2	1	0
EP2CON	EPIE0	EPPL0	-	-	EPPS0[3:0]			
R/W	R/W	R/W	-	-	R/W			
Initial value	0	0	-	-	0	0	0	0
D2H	7	6	5	4	3	2	1	0
EP1CON	EPIE1	EPPL1	-	-	EPPS1[3:0]			
R/W	R/W	R/W	-	-	R/W			
Initial value	0	0	-	-	0	0	0	0

D3H	EPIE2	EPPL2	-	-	EPPS2[3:0]			
R/W	R/W	R/W	-	-	R/W			
Initial value	0	0	-	-	0	0	0	0
Note: n stands for 0/1/2								
Bit number	Bit symbol	Description						
7	EPIEn	External interrupt enable control 0: disable 1:enable *Note: n=0/1/2 corresponds to external interrupt 2/3/4 respectively*						
6	EPPLn	External Interrupt Trigger Edge Selection 0： Rising edge 1： Falling edge *Note: n=0/1/2 corresponds to external interrupt 2/3/4 respectively*						
5~4	-							
3~0	EPPSn[3:0]	Interrupt Pin selection The table for pin numbers and corresponding pins please refer to Table 8-5-3 *Note: n=0/1/2 corresponds to external interrupt 2/3/4 respectively*						

Table 8-5-3 Index for Interrupt Pin

Pin name	**Number**	**Pin name**	**Number**
P00	0	P30	8
P01	1	P31	9
P02	2	P32	10
P03	3	P33	11
P04	4	P34	12
P05	5	P35	13
P06	6		
P07	7		

8.5.3 External Interrupt Control Method and Examples

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

  For instance, to enable External Interrupt0, the program will be:

```
-------------------------------------------------------------------------------------------
void INT0_init(void)
{
        P32F = 1;//The interrupt pin    external
                        interrupt 0 is P32, set P32 as
                        input function
        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 example, to enable external interrupt 1, the program is as follows:

```
-------------------------------------------------------------------------------------------
void INT1_init(void)
{
        P33F = 1;         //set The interrupt pin of external
                                interrupt 1 is P33, set P33 as input
                                function
        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 1 Interrupt Service Routine
}

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

◆ **External Interrupt 2~4 control example**

Taking External Interrupt 2 for example, if P00 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)
{
    P00F = 1;       //set P00 as input pin
    EP0CON = (1<<7) | (0<<6) | 0;//set rising edge trigger and set the index number for the interrupt pin
                                    //To set falling edge trigger the codes may be EPCON = (1<<7) |(1<<6) | 0;;
    INT2EN = 1; //enable INT2 interrupt
    EA = 1;                 //enable Global Interrupt
}
void INT2_ISR (void) interrupt 4
{
    if(EPIF & 0x01)             //judge the Interrupt Flag for External Interrupt 2
    {
         EPIF = 0x01; //write 1 to the Interrupt Flag to clear it
             //External Interrupt 2Interrupt Service Routine
        ......
    }
}
------------------------------------------------------------------------------------------
```

9 Clock System

9.1 Clock System Introduction

CA51F5 Series has several clock sources as follows：

- 16 MHz Internal RC Oscillator
- 131 KHz Internal RC Oscillator

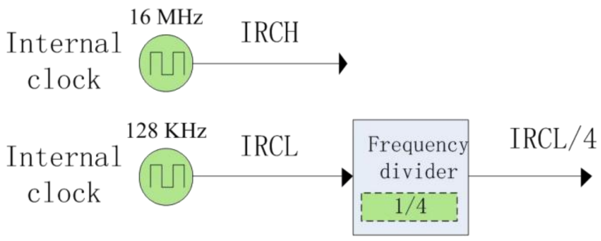


Figure 9-1-1 Clock Sources

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
IRCH	16 MHz Internal RC Oscillator
IRCL	131 z Internal RC Oscillator

9.1.2 16MHz 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 precision error is 2% at most and factory frequency is 16MHz@3.3V/25℃.

9.1.3 131 KHz Internal RC Oscillator(IRCL)

IRCL can be enabled/disabled by setting the ILCKE of register CKCON. When IRCL is set as system clock the power consumption decrease as well. The factory frequency for it is 131 KHz

9.2 Clock Control Register Description

Table 9-2-1 Register CKCON

8030H	7	6	5	4	3	2	1	0
CKCON	IHCKE	ILCKE	-	-	-	-	-	SCKS
R/W	R/W	R/W	-	-	-	-	-	R/W
Initial Value	0	0	0	0	0	0	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.*
5~1	-	
0	SCK	System clock selection 0: IRCH selected 1: IRCL selected

Table 9-2-2 Register IHCFG

8082H	7	6	5	4	3	2	1	0
IHCFG	IHCFG[1:0]		IHCFG[7:2]					
R/W	R/W		R/W					
Initial Value	-	-	-	-	-	-	-	-

Bit number	Bit Symbol	Description
7~6	IHCFGL[1:0]	Low 2-bit IRCH frequency adjustment register
5~0	IHCFGL[7:2]	High 6-bit IRCH frequency adjustment register

Table 9-2-3 Register ILCFG、ILCFGH

8033H	7	6	5	4	3	2	1	0
ILCFGL	ILCFG[7:0]							
R/W	R/W							
Initial Value	-	-	-	-	-	-	-	-
8034H	7	6	5	4	3	2	1	0
ILCFGH	-	-	-	-	-	-	-	ILCFG[8]
R/W	-	-	-	-	-	-	-	R/W
Initial Value	-	-	-	-	-	-	-	-
Bit number	Bit Symbol	Description						
15~9	-	-						
8~0	ILCFG	IRCL frequency adjustment						

9.3 System Clock

The system clock is controlled by register CKCON 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

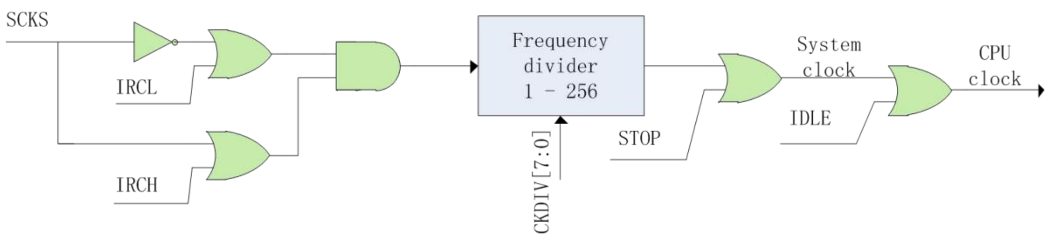


Figure 9-3-1 System Clock Architecture

9.3.2 System Clock Control Register Description

Table 9-3-2-1 Register CKDIV

8031H	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<<7)
#define CKSEL_IRCH     0
void Sys_Clk_Set_IRCH(void)
{
      CKCON |= IHCKE; //enable IRCH
      CKSEL = (CKSEL&0xFE) | CKSEL_IRCH;          //set IRCH as system clock
}
------------------------------------------------------------------------------------------
```

◆ **Set IRCL as the system clock**

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

```
------------------------------------------------------------------------------------------
#define ILCKE        (1<<6)
#define CKSEL_IRCL     1
void Sys_Clk_Set_IRCL(void)
{
      CKCON |= ILCKE;          //enable IRCL
      Delay_ms(1);                   //delay 1ms and wait for IRCL stabilization
      CKSEL = (CKSEL&0xFE) | CKSEL_IRCL; //set IRCL as system clock
}
```

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.

----

10 Power Supply and Reset System

10.1 Power Supply

There is 1.8V - 5.5V source between VDD pin and VSS pin for CA51F5 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.

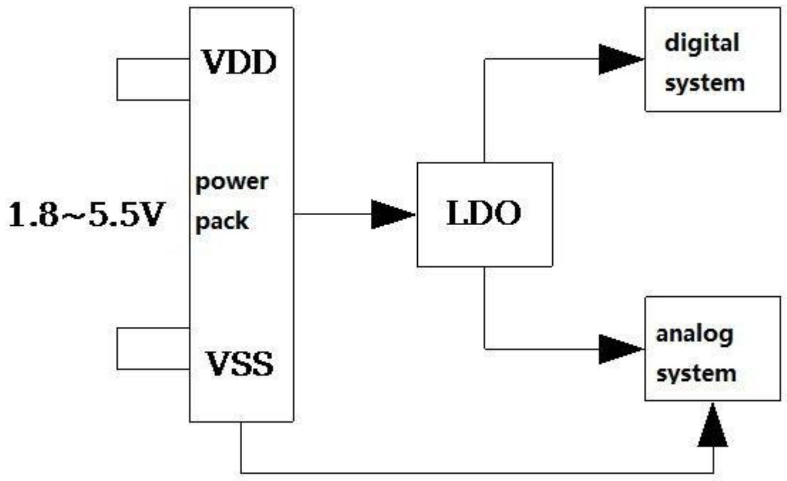


Figure 10-1-1 Power Supply Architecture

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

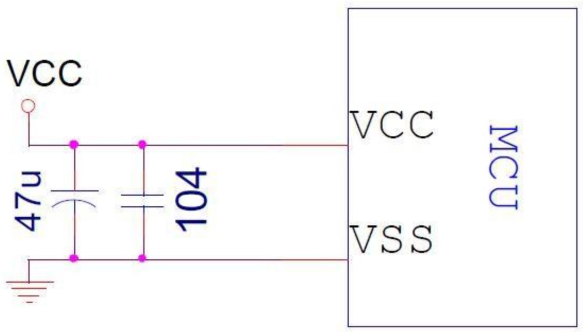


Figure 10-1-2 Typical Circuit for Power Supply

Important reminders:

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 CA51F5 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.

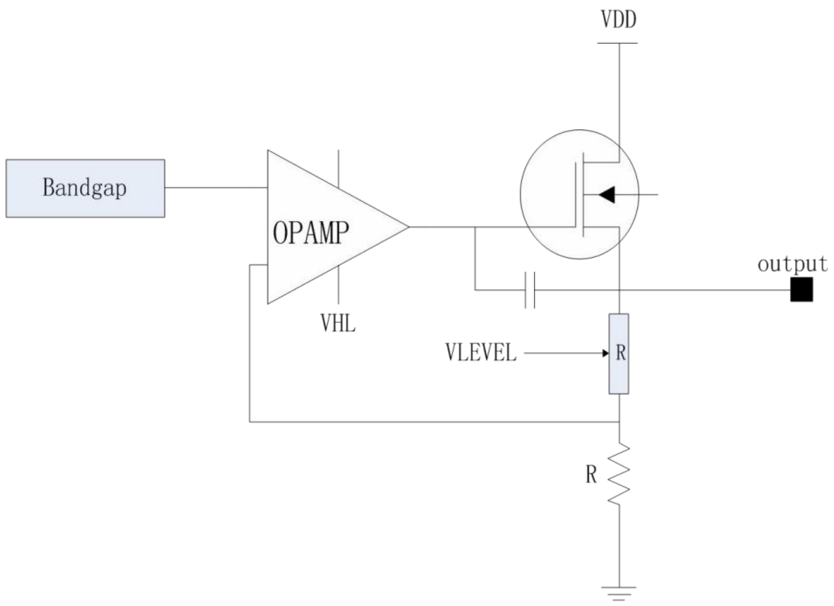


Figure 10-1-3 LDO Module Schematic

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						
		Internal reference voltage(Bandgap)output adjustment 0000: 0.825V 0001: 0.850V 0010: 0.875V						

7~4	FLEVEL	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: when the internal reference voltage is powered on, it is automatically loaded by the system and cannot be modified by the user.*
3	VHL	LDO working mode selection 1: High Power mode 0: Low Power mode
2~0	VLEVEL	LDO output voltage selection 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 LDO. Changing the output voltage of LDO will cause the change of internal clock frequency. Generally, the LDO voltage can maintain the default value and is not recommended to be modified.* *2.It is not recommended to set LDO output voltage below 1.5V otherwise the chip may operates abnormally*

10.2 Reset System

There are multiple internal and external reset sources for CA51F5 Series chip as figure 10-2-1 shows.

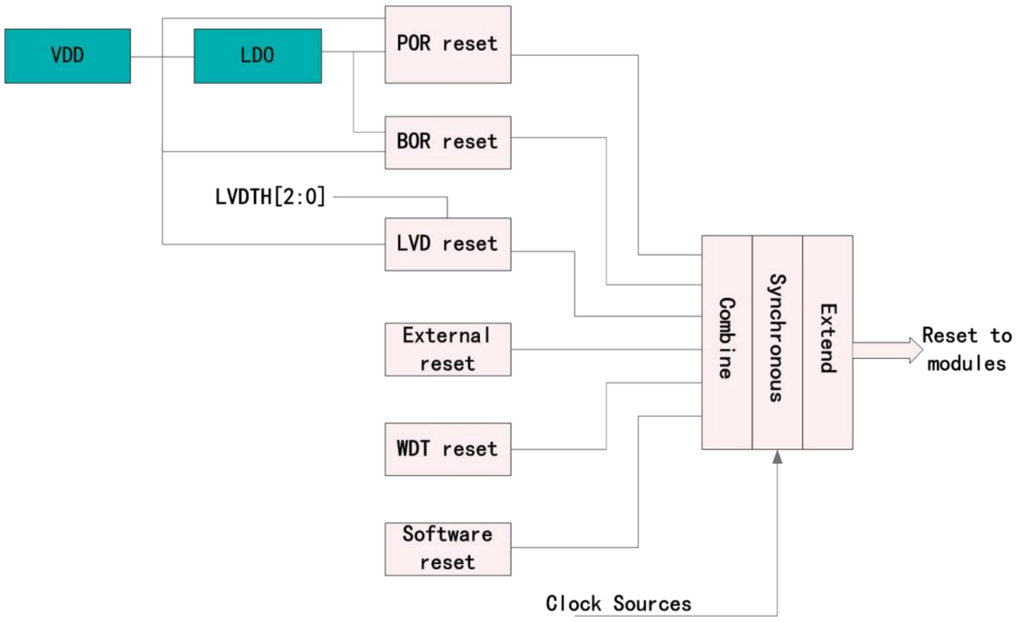
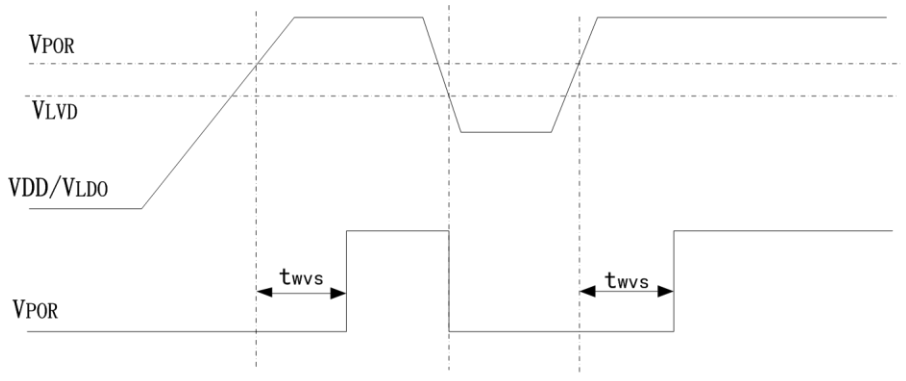


Figure 10-2-1 Reset System Architecture

- **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.



twvs: time to wait until voltage stable

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.

- **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.

POR and external hard reset will reset all the circuits while LVD and WDT can reset other circuits but not reset themselves. (eg：After WDT reset, WDT registers remains former status while others are all reset) LVD/WDT and soft reset can not reset storage control circuit. Program starts from Mask ROM after POR and external hard reset. 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 CA51F5 Series : IDLE, STOP and Low Speed mode. The system power consumption for IDLE, STOP and Low Speed mode is less than 12uA, 6uA 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.

Note that the instruction to set IDLE needs to be immediately followed by two nop instructions to prevent program errors.

11.2 STOP mode

The STOP mode can seen as a 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.

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, IRCL cannot be closed 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(131kHz IRCL). The current will be less than 20uA if IRCL is set as the system clock.

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 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 IDLST

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

Bit number	Bit Symbol	Description
7	-	-
6	I2CINT/WDIF/LVDINT/EPIF[2]	Interrupt status of LVD/External Interrupt 4/WDT/I^{2}C in IDLE mode
5	T2INT/TKINT/TMINT/EPIF[1]	Interrupt status of TK/Timer2/External Interrupt 3/TMC in IDLE mode
4	TI/RI/PWMINT/EPIF[0]	Interrupt status of UART0/PWM/External Interrupt 2 in IDLE mode
3	TF1	Interrupt status of Timer1 in IDLE mode
2	PIF[1]	Interrupt status of External Interrupt1 in IDLE mode
1	TF0	Interrupt status of Timer0 in IDLE mode
0	PIF[0]	Interrupt status of External Interrupt0 in IDLE mode

Table 11-4-3 Register STPST

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

Bit number	Bit Symbol	Description
7	-	-
6	WDTWKF/LVDWKF/I2CWKF	Interrupt status of LVD/External Interrupt 4/WDT/I^{2}C in STOP mode
5	TKWKF/TMWKF	Interrupt status of TK/Timer2/External Interrupt 3/TMC in STOP mode
4	EPWKF[2]	Interrupt status of UART0/PWM/External Interrupt 2 in STOP mode
3	EPWKF[1]	Interrupt status of Timer1 in STOP mode
2	EPWKF[0]	Interrupt status of External Interrupt1 in STOP mode
1	PWKF[1]	Interrupt status of Timer0 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：

```
-------------------------------------------------------------------------------------------
#define IHCKE           (1<<7)
#define ILCKE           (1<<6)

#define CKSEL_IRCH     0
#define CKSEL_IRCL     1

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
        IE_EA = EA; //Save global interrupt enable bit status
        EA = 0;              //In order to ensure that three NOP commands are executed after waking up from STOP
                             //mode, STOP is entered after closing the interrupt, and interrupt is opened after
                             executing //three NOP commands. Note: turning off the global interrupt does not affect
                             the interrupt //wake-up STOP
        PCON |= 0x02;        // enters STOP mode
        _nop_();
        _nop_();
        _nop_();
        EA = IE_EA;            //Restore the original global interrupt switch state
        PWCON ｜=0x08; //The LDO must return to high power consumption mode after the chip exits STOP mode
}
----------------------------------------------------------------------------------------------
```

◆ **IDLE Mode Example**

IDLE Mode program is like：

```
#define IHCKE           (1<<7)
#define ILCKE           (1<<6)

#define CKSEL_IRCH     0
#define CKSEL_IRCL     1

void Idle(void)
{
        bit IE_EA;
        I2CCON = 0; //disable I2C for it is the default enabled, otherwise the IRCH cannot be disabled
        CKCON |= ILCKE; //enable IRCL clock
        CKCON |= CKSEL_IRCL;      //switch system clock to IRCL
        CKCON&=~ IHCKE;      //disable IRCH

        PWCON &=0xf7; //set LDO in low power consumption mode
        MECON |= (1<<6); //set FLASH in deep sleep mode
        IE_EA = EA;            // Save global interrupt enable bit status
        EA = 0;                //In order to ensure that three NOP commands are executed after waking up from IDLE
                               //mode, IDLE is entered after closing the interrupt, and interrupt is opened after
                               //executing three NOP commands. Note: turning off the global interrupt does not affect
                               //the interrupt wake-up IDLE

        PCON |= 0x01;          //enters IDLE mode
        _nop_();
        _nop_();
        _nop_();
        EA = IE_EA;            //Restore the original global interrupt switch state
        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：

```
    #define IHCKE              (1<<7)
    #define ILCKE              (1<<6)

    #define CKSEL_IRCH     0
    #define CKSEL_IRCL     1

    void LowSpeedMode(void)
    {
        I2CCON = 0; //disables I2C for it is the default enabled, otherwise the IRCH cannot be disabled
          CKCON |= ILCKE;       //enables IRCL clock
          CKCON |= CKSEL_IRCL;    //system clock switches to IRCL
          CKCON&=~ IHCKE;    //disables IRCH
          PWCON &=0xF7; //set LDO in low power consumption mode
    }
```

Note：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**

Timer 0 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 automatically after the interrupt response.When GATE0 (TCON[3])=0, the timer/counter's is enabled/disabled byTR0 (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.

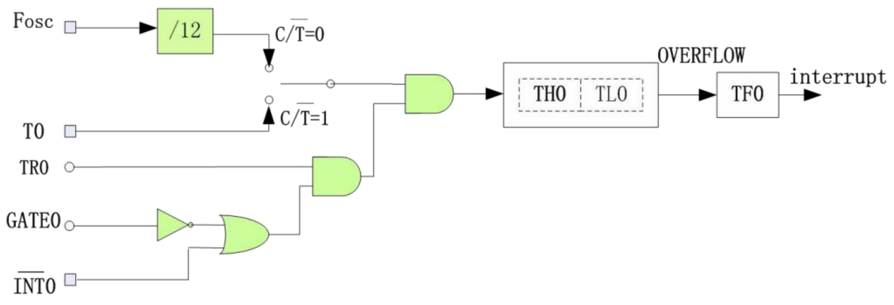


Figure 12-1-1-1 Timer0 Mode0/1

- **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.

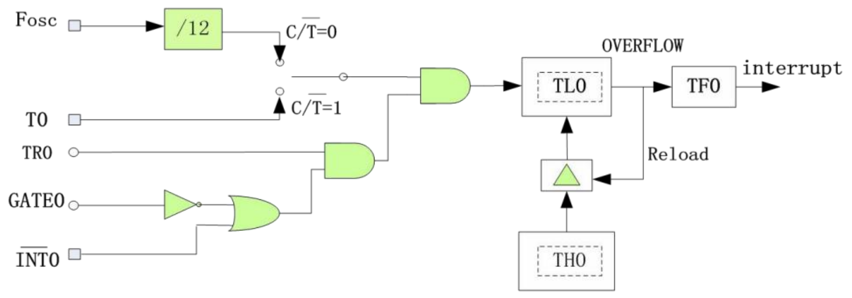


Figure 12-1-1-2 Timer0 Mode2

- **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 counter. 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

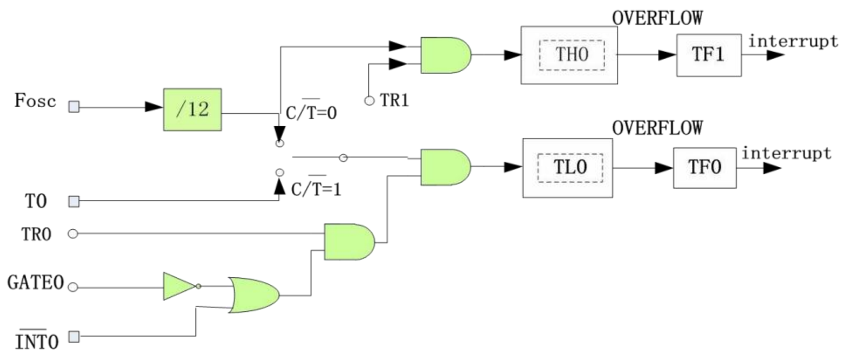


Figure 12-1-1-3 Timer0 Mode3

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 automatically after the interrupt response
6	TR1	Timer1 enable control, 1 enables it
5	TF0	Timer0 overflow flag, it is cleared 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 Interrupt0is 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 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
4	T1M0	
3	GATE0	Timer0 gating control. When it equals 1, Timer0 is enabled/disabled by INT0
2	CT0	Timer0Counter/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 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
0	T0M0	

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 is a 13 bit timer/counter in this mode. The higher 8 bits are stored in TH1 and the lower 5 bits are stored in TL1[4:0] with TL1[7:5] invalid. When Timer1 overflows, the interrupt flag TF1(TCON[7]) will be set to 1. TF1 will be cleared to 0 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.

- **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. INT0 signal with high level with enable the counting and vice verse.

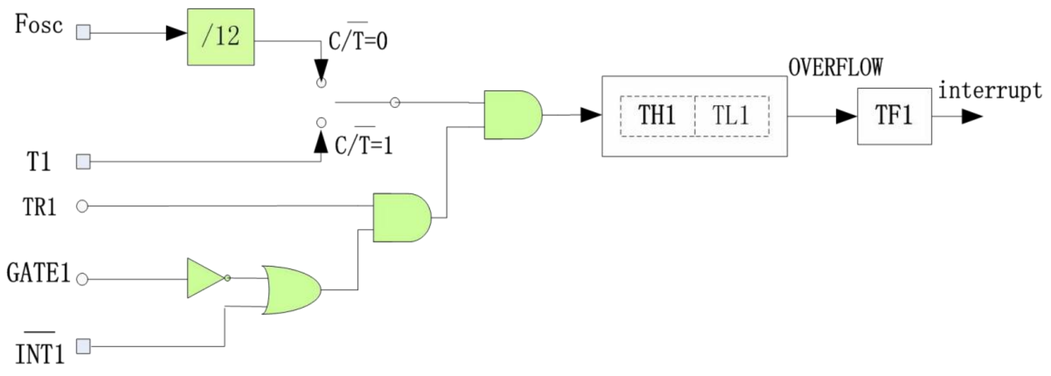


Figure 12-2-1 Timer1 Mode 0/1

- **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.

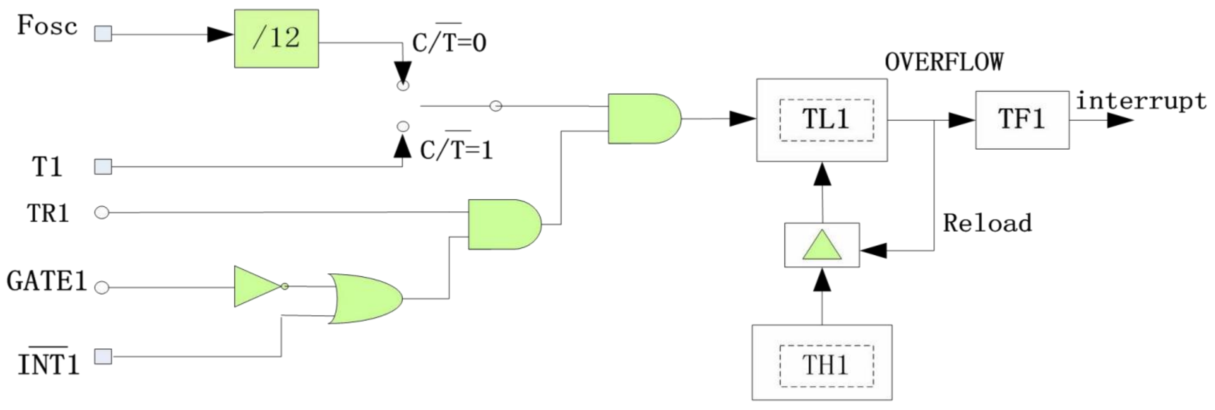


Figure 12-2-2 Timer1 Mode2

- **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 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	Low 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	High 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

Timer 2 is a 16-bit (TH2, TL2) timer/counter. T2P0 and T2P1 bits can select different control methods or clock sources. When T2P=0, 3, the system clock is selected as timer 2 clock (note: unlike timer 0, 1, the clock does not go through 12 divisions); when T2P=0, timer 2 is enabled by TR2 bit; when T2P=3, it is gated by T2 level, when T2 is high, the count is enabled, when T2 is low, the count is stopped. When T2P=1, 2, the input signal of T2 is selected as the counting clock, when T2P=1, the falling edge of T2 is detected to count, when T2P=2, the rising edge of T2 is detected.

Timer 2 can set different working modes by T2M0 and T2M1 bits. When T2M=0, Timer2 works in timer/counter mode and TH2, TL2 is automatically accumulated as 16-bit counter; in this mode, two different reload modes can be selected by setting T2R0, T2R1 bits or turn off the reload function, in reload mode, T2CH, T2CL stores the reload value, when T2R=2, Timer2 overflow will load the count initial value from T2CH, T2CL to load the count initial value to TH2, TL2, and when T2R=3, the reload is performed on the falling edge of pin T2EX. When the reload event occurs, the reload interrupt flag RF2 is set to 1. If timer 2 interrupt enable will trigger the reload interrupt, RF2 clears 0 by writing 1.

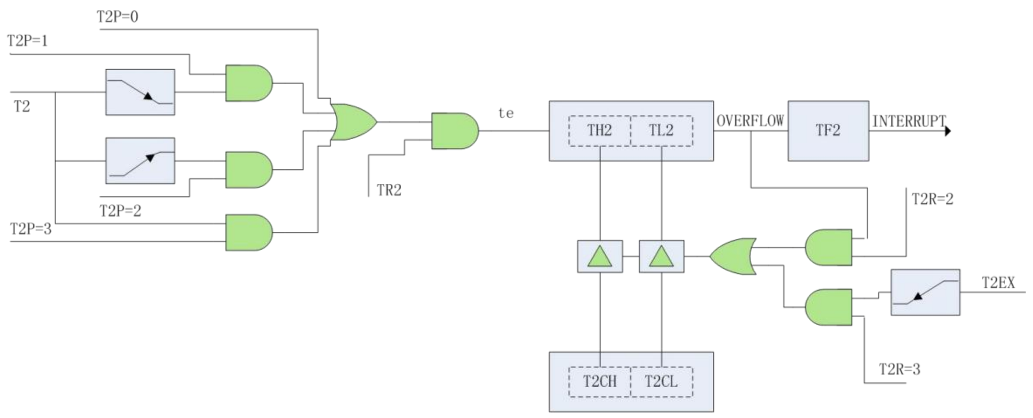


Figure 12-3-1-1 Timer2 Reload Mode

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.

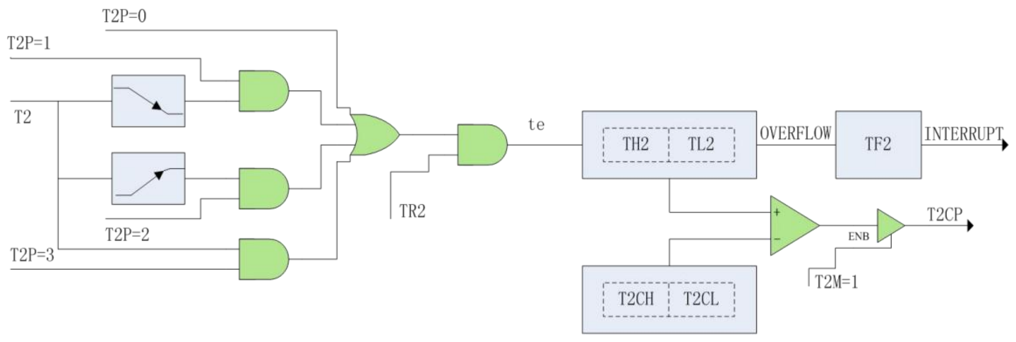


Figure 12-3-1-2 Timer2 Compare Mode Compare Mode

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 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。

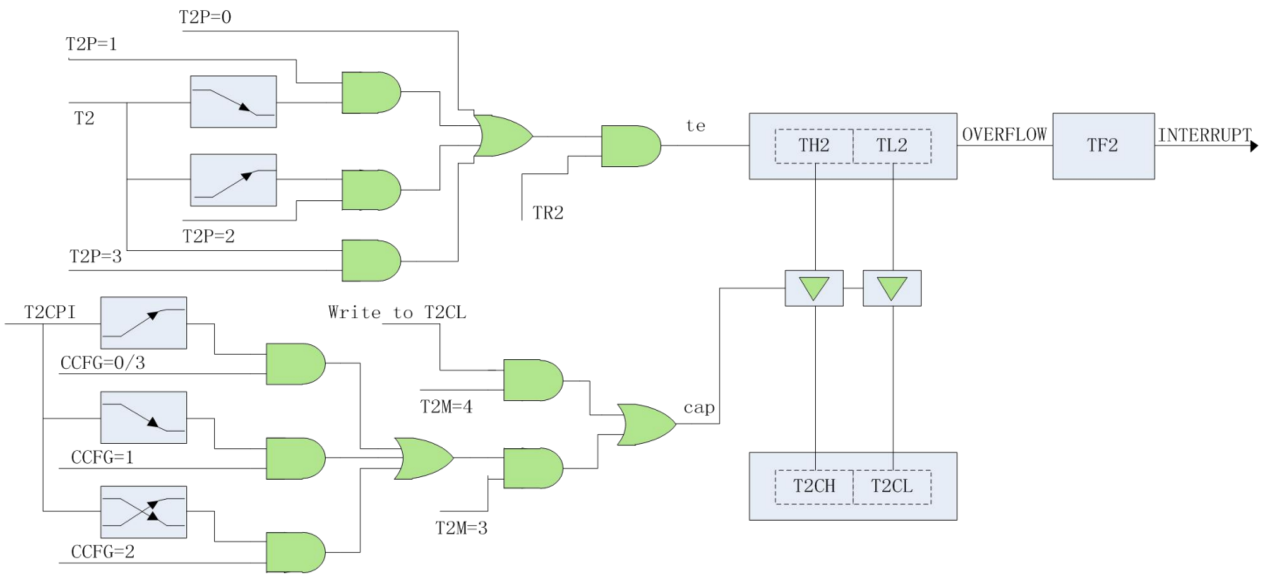


Figure 12-3-1-3 Timer2 Capture Mode

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
Initial Value	-	-	-	0	0	-	0	0

Bit number	Bit Symbol	Description
7	TF2	Timer2 counter overflow interrupt flag, cleared by writing 1 to it
6	CF2	Capture interrupt flag, cleared by writing 1 to it
5	RF2	Automatic reload interrupt flag, cleared 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 16MHz, the count time can be 0.128ms - 4.096s 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.

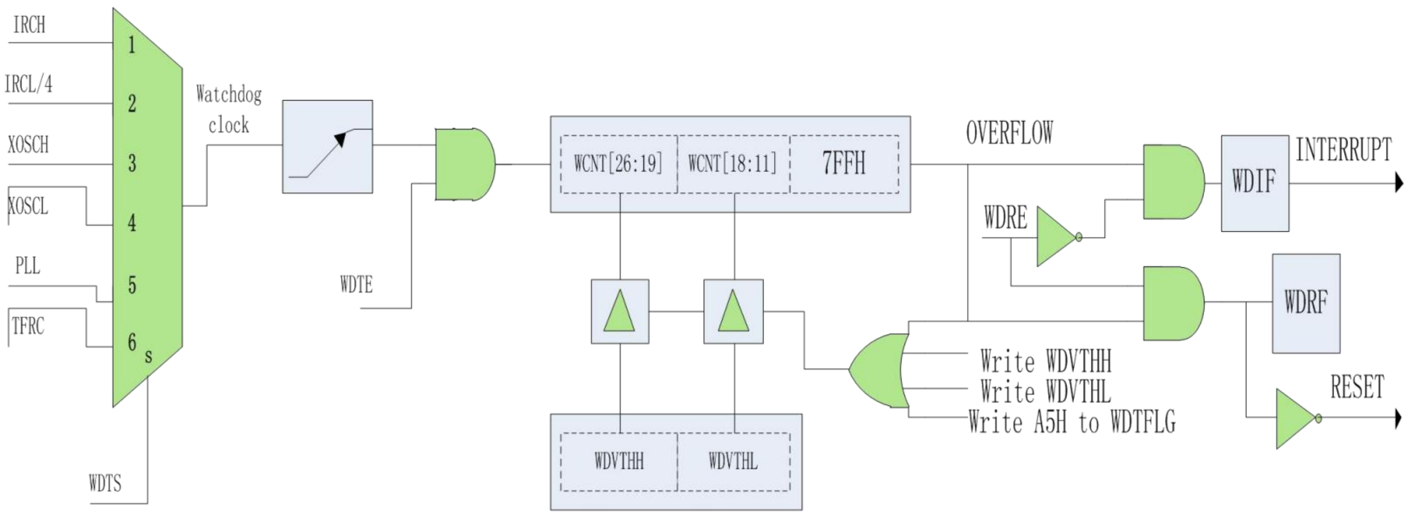


Figure 13-1-1 Watchdog Module Architecture

13.2 Watchdog Timer(WDT) Register Description

Table 13-2-1 Register WDCON

AAH	7	6	5	4	3	2	1	0
WDCON	WDTS[1:0]			-	-	-	-	WDRE
R/W	R/W			-	-	-	-	R/W
Initial Value	0	0		-	-	-	-	0
Bit number	Bit Symbol	Description						
7~6	WDTS	WDT clock selection 01: IRCH						

		10: IRCL with frequency divided by 4 Others: WDT disabled
5~1	-	
0	WDRE	WDT function selection 0: interrupt happens when WDT overflows 1: reset happens when WDT overflows

Table 13-2-2 Register WDFLG

A1H	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

A2H	7	6	5	4	3	2	1	0
WDVTHL	WDVTH[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
ADH	7	6	5	4	3	2	1	0
WDVTHH	WDVTH[15:8]							
R/W	R/W							
Initial Value	-	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
15~0	WDVTH	WDT threshold setting, the equation is as follows: WDT trigger time ＝ (WDVTH * 800H+7FFH) * clock cycle

13.3 Watchdog Timer Control Example

◆ **Example for Watchdog interrupt mode**

For instance, IRCH is set for the watchdog clock and the frequency for it is 16MHz. The watchdog works in interrupt mode and the overflow time is one second, the program is like:

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

#define WDTS_IRCH           (1<<6)
#define WDTS_IRCL           (2<<6)

#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 = 0x1E;                    //set one second as the time for watchdog
    WDVTHL = 0x83;
    WDFLG = 0xA5;                     //refresh the watchdog
    INT4EN = 1;                       //enable watchdog interrupt
    EA = 1;                         //enable global interrupt

}
void WDT_ISR (void) interrupt 8
{
    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 16MHz. The watchdog works in reset mode and the overflow time is one second, the program is like：

```
----------------------------------------------------------------------------------------
#define WDTS_IRCH           (1<<6)
#define WDTS_IRCL           (2<<6)

#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 = 0x1e;                          //set one second as the time for watchdog
        WDVTHL = 0x83;
        WDFLG = 0xA5;                           //refresh the watchdog
}
```

----

14 TMC Timer

14.1 TMC Function Introduction

The clock source for TMC Timer is IRCL. The minimum time unit for its interrupt is 512 IRCL clock cycles, with1~256 minimum time units configurable for the interrupt.TMC interrupt is able to awaken CPU in STOP/IDLE mode.

14.2 TMC Register Description

Table 14-2-1 Register TMCON

D5H	7	6	5	4	3	2	1	0
TMCON	TME	-	-	-	-	-	-	TMF
R/W	R/W	-	-	-	-	-	-	R
Initial value	0	-	-	-	-	-	-	0
Bit number	Bit Symbol	Description						
7	RTCE	TME module enable control, 1 enables it						
6~1	-	-						
0	TMF	TMC interrupt flag, 1 enables it, cleared when 1 is written to it						

Table 14-2-2 Register TMSNU

E6H	7	6	5	4	3	2	1	0
TMSNU	TMSNU[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	-
Bit number	Bit Symbol	Description						
7~0	TMSNU	TMC interrupt time configuration register, the time for TMC interrupt is （TMSNU+1）×512×Tircl *Note: Tircl is one clock cycle period of IRCL*						

14.3 TMC Control Example

Set the minimum time for TMC interrupt which is 512 IRCL clock cycles， the program is like：

```
--------------------------------------------------------------------------------------
#define TME(N)          (N<<7)    //N=0-1
#define TMF             (1<<0)

#define IHCKE           (1<<7)
#define ILCKE           (1<<6)

void INT3_ISR (void) interrupt 5
{
        if(TMCON & TMF)             //judge the TMC interrupt flag
        {
                TMCON |= TMF;       //clear TMC interrupt flag

        }
}


void TMC_init(void)
{
        CKCON |= ILCKE;     //enables IRCL clock
        TMCON = TME(1); //enables TMC
        TMSNU = 0;      //set one time unit (512IRCL clock cycles) for the interrupt
        INT3EN =1;      //enables TMC interrupt
        EA = 1;         //enables global interrupt
}
--------------------------------------------------------------------------------------
```

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 14 I/O pins for CA51F5 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 is a function register PnxF corresponding to pin Pnx (n=0 or 3, stands for P0 or P3, x=0~7, stands for Pn.0~Pn.7). Users can configure the main function and other options by setting register PnxF. For more information please refer to the register description.

Each I/O can enable the up/down resistance through PnxPUP/PnxPDP(PnxF[7]/PnxF[6]), and the strong/weak up/down resistance is controlled by PU_SEL/PD_SEL(PnxC[5]/PnxC[4]) selection, when PU_SEL/PD_SEL = 1, select strong up/down, otherwise it is weak up/down and the default is strong up/down.

When I/O is set to output mode, when PnxOPR(PnxF[5]) = 1, I/O is open drain output mode.

When I/O is push-pull output, the IO push-pull output driving strength can be set through DRV(PnxC[3:2]) and the IO output turnover slope can be set through SR(PnxC[1:0]). When the I/O output level is reversed, an overshoot signal will be generated at the I/O port due to inductance effect. This overshoot signal may have a certain impact on the chip system. Reducing I/O output strength and I/O speed can effectively reduce the overshoot signal amplitude. These two parameters can be configured flexibly during application.

When I/O is input mode, you can use SMIT_EN(PnxC[6]) bit selects Schmidt mode or inverter mode. When inverter mode is selected, the trigger threshold value of high and low levels is 1/2 VDD, and Schmidt mode is selected by default.

Main features of GPIO：

- High impedance mode configurable
- The strong pull-up, weak pull-up, strong pull-down and 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 it is push-pull output, the IO driving strength can be set independently
- When it is push-pull output, the IO output speed can be set independently

Important: the input voltage of all GPIO pins shall not be higher than the VDD pin voltage, otherwise the chip may work abnormally.

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

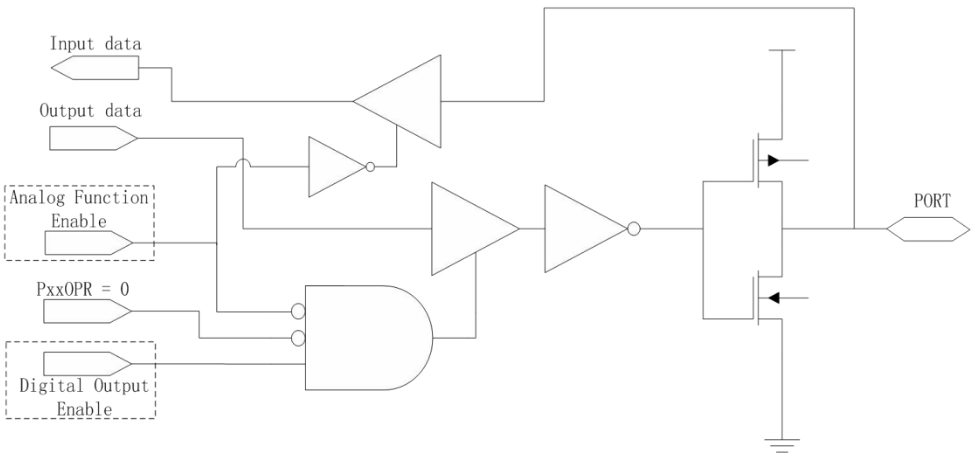


Figure 15-1-1 I/O Push-pull Mode Structure

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

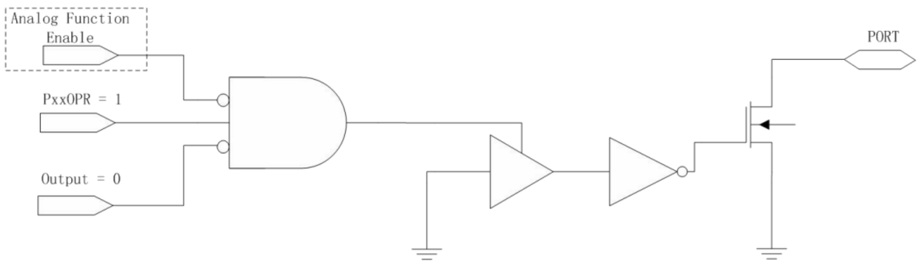


Figure 15-1-2 I/O Open-drain Mode Structure

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

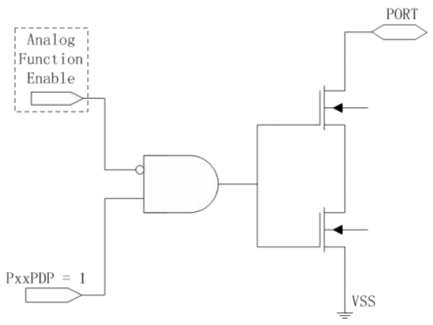


Figure 15-1-3 I/O Pull-down Mode Structure

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

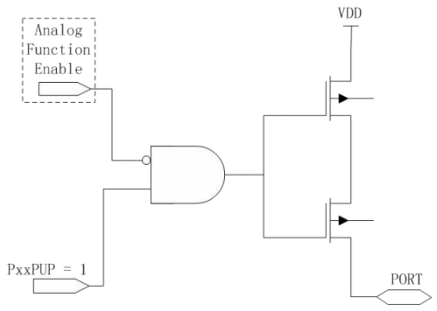


Figure 15-1-4 I/O Pull-up Mode Structure

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 P3

B0H	7	6	5	4	3	2	1	0
P3	-	-	P35	P34	P33	P32	P31	P30
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
7~0	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 15-2-3 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

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

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, 3, stands for P0, P3
x=0~7, stands for Pn.0~Pn.7

Table 15-2-4 Register PnxC

8120H	7	6	5	4	3	2	1	0
P00C	-	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
8121H	7	6	5	4	3	2	1	0
P01C	-	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
8122H	7	6	5	4	3	2	1	0
P02C	-	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
8123H	7	6	5	4	3	2	1	0
P03C	-	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
8124H	7	6	5	4	3	2	1	0
P04C	-	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
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
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

Bit number	Bit symbol	Description
7	-	-
6	SMIT_EN	Mode selection bit when IO is an input 0: Inverter mode 1: SMIT mode
5	PU_SEL	Pull up resistance selection bit 0: weak pull-up (pull-up resistance is 45k) 1: Strong pull-up (pull-up resistance is 10K)
4	PD_SEL	Pull down resistance selection bit 0: weak pull-down (pull-up resistance is 45k) 1: Strong pull-down (pull-up resistance is 15K)
3~2	DRV	Output strength selection bit, range: 0 ~ 3. The larger the value, the stronger the driving ability
1~0	SR	Output slope control bit, range: 0 ~ 3. The larger the value, the higher the IO turnover slope (the faster the speed)

Table 15-2-5 Pin Alternate Function Mapping

Value Name	0	1	2	3	4	5	6	7
P00S	high impedance	digital input	digital output	DAK[0]	TK[9]	high impedance	high impedance	LVD_IN0
P01S	high impedance	digital input	digital output	DAK[1]	TK[8]	PWM[3]	TLED_COM	LVD_IN1
P02S	high impedance	digital input	digital output	I2C_SCL	TK[7]	high impedance	high impedance	high impedance
P03S	high impedance	digital input	digital output	I2C_SDA	TK[6]	high impedance	high impedance	high impedance
P04S	high impedance	digital input	digital output	T2CP	TK[5]	PWM[4]	high impedance	high impedance
P05S	high impedance	digital input	digital output	T2EX	TK[4]	PWM[5]	high impedance	high impedance
P06S	high impedance	digital input	digital output	T2	TK[3]	high impedance	high impedance	high impedance
P07S	high impedance	digital input	digital output	RESET	TK[2]	high impedance	high impedance	high impedance
P30S	high impedance	digital input	digital output	I2C_SDA	TK[1]	UART0_RX	high impedance	high impedance
P31S	high impedance	digital input	digital output	I2C_SCL	TK[0]	UART0_TX	high impedance	high impedance
P32S	high impedance	digital input/INT0	digital output	T0	TK[12]	PWM[0]	high impedance	high impedance

P33S	high impedance	digital input/INT1	digital output	high impedance	TK[11]	PWM[1]	high impedance	high impedance
P34S	high impedance	digital input	digital output	high impedance	TK[10]	PWM[2]	high impedance	high impedance
P35S	high impedance	digital input	digital output	T1	TKCAP	high impedance	high impedance	high impedance

15.3 Pin control Example

◆ **Set the Pin function**

For instance, P00 is set to be push-pull output, the program is like：

```
-----------------------------------------------------------------------------------------
------- P00F = 2;
-----------------------------------------------------------------------------------------
```

P00 is set to be open-drain output, the program is like：

```
-----------------------------------------------------------------------------------------
------- P00F = (1<<5)|2;
-----------------------------------------------------------------------------------------
```

P00 is set to be open-drain output with pull-up enabled, the program is like：

```
-----------------------------------------------------------------------------------------
-------
P00C |= (1<<5);
P00F = (1<<7) | (1<<5) | 2;
-----------------------------------------------------------------------------------------
```

P00 is set to be input with pull-up enabled, the program is like：

```
-----------------------------------------------------------------------------------------
-------
P00C |= (1<<5);
P00F = (1<<7) | 1;
-----------------------------------------------------------------------------------------
```

16 Universal Serial Interface (UART0)

16.1 Function Introduction

UART0 is a full-duplex synchronous/asynchronous serial data transceiver (full-duplex means that data can be sent and received at the same time), which is basically compatible with the standard 8051. UART0 receiver has a one-byte cache, which means that one byte of the received data will be sent to the cache register, while the receiver can receive new data, of course, before the new one-byte data is received, the previously received Of course, before the new one byte data is received, the previous received one byte data must be read, otherwise it will be overwritten by the new data. Register S0BUF is the send/receive data register of UART0. Physically, S0BUF is actually two registers, one is the data send register and the other is the data receive register. Writing S0BUF will write data to the send register and start data sending, while reading S0BUF will read the received one byte data in the receive register.

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

Table 16-1-1 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 Timer1/2)/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	9 bit asynchronous mode	The baud rate is 2*SMOD*CPUCLK*(overflow rate of Timer1/2)/32, please refer to UCKS of T2CON

Note：
1.The prerequisite for the baud rate above is that UDE=0. When UDE=1, the baud rate is decided by DNUM. 2.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.

- **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.

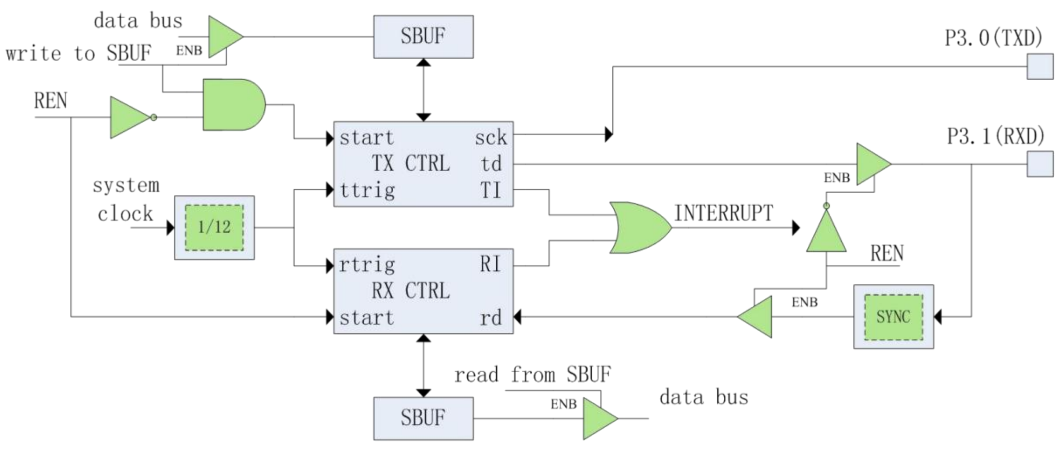


Figure 16-1-1 UART0 Mode0 Schematic

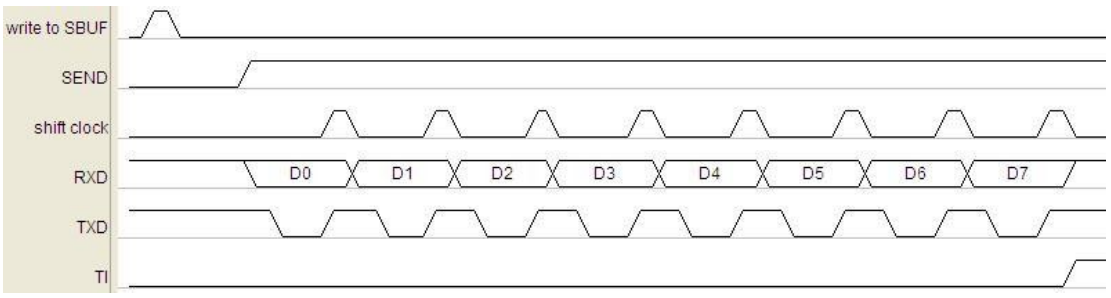


Figure 16-1-2 UART0 Data Transmission Waveform in Mode0

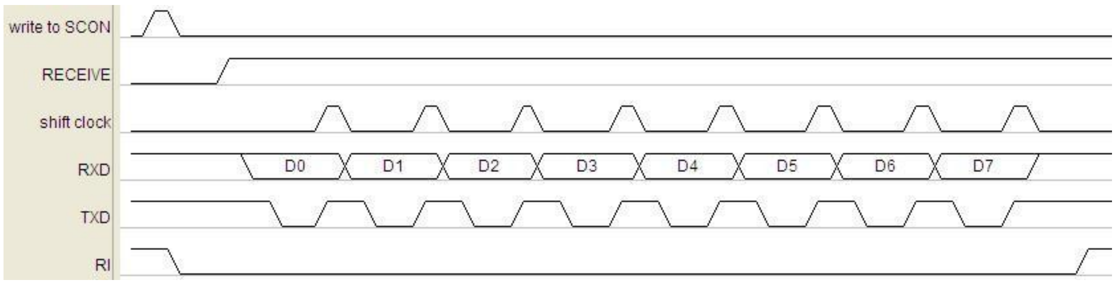


Figure 16-1-3 UART0 Data Receiving Waveform in Mode0

- **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]).

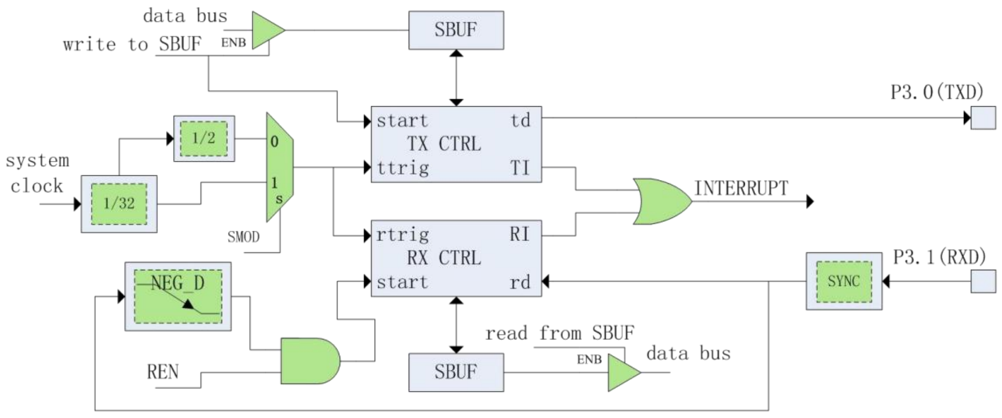


Figure 16-1-4 UART0 Mode1 Schematic

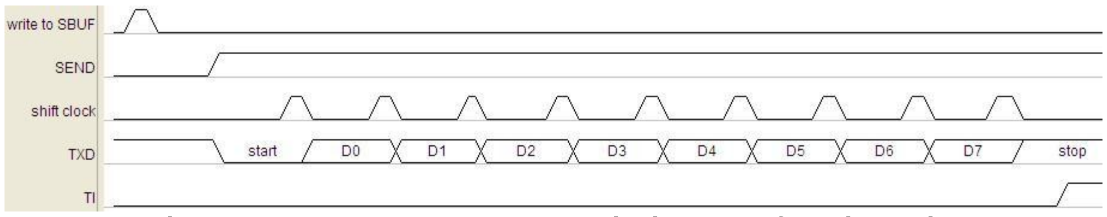


Figure 16-1-5 UART0 Data Transmission Waveform in Mode1

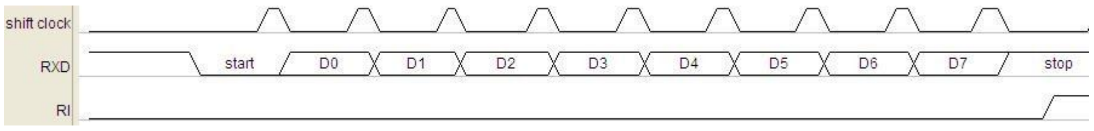


Figure 16-1-6 UART0 Data Receiving Waveform in Mode1

- **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).

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]).

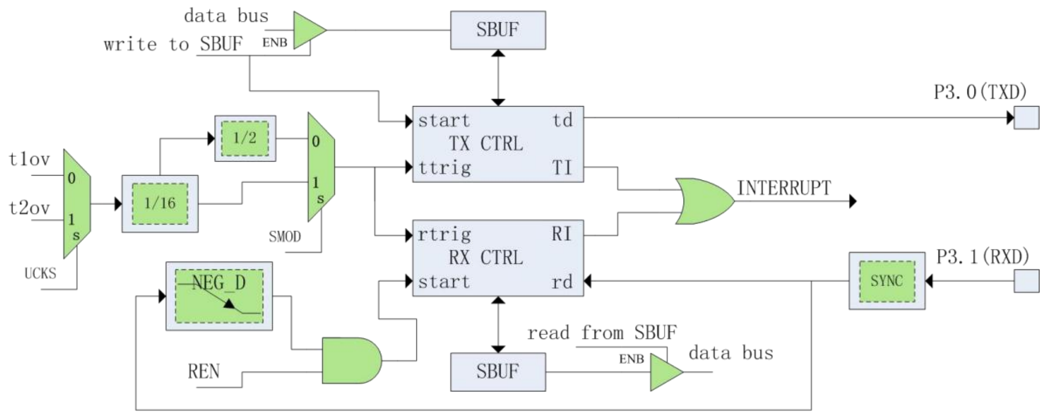


Figure 16-1-7 UART0 Mode2 Schematic

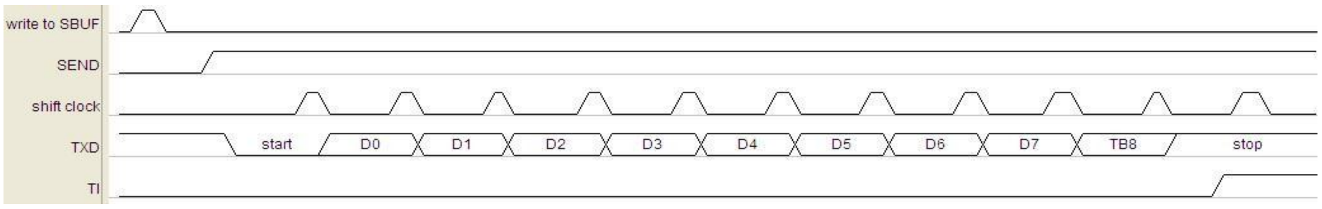


Figure 16-1-8 UART0 Data Transmission Waveform in Mode2

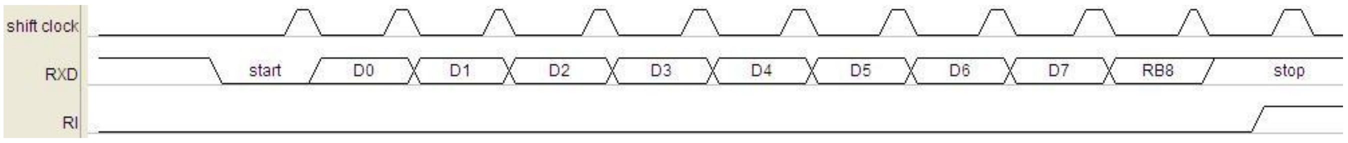


Figure 16-1-9 UART0 Data Receiving Waveform in Mode2

- **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.

- **Rapid baud rate setting**

For UART0 in standard 51 MCU, the baud rate of it is constantly 1/32 of the overflow rate of the timer. However, due to the main clock' frequency for CA51F5 series is 16MHz (or 16MHz with frequency division), there will be deviation between the frequency configured and the standard frequency. Therefore, correction process for baud rate is necessary and is designed for CA51F5 series MCU. The baud rate of UART is not constantly 1/32 of the overflow rate of the timer any more. It is decided by the register UDCKS instead. For example, when the UART baud rate is constantly 1/32 of the overflow rate of the timer2, to set the baud rate to115200, the corresponding equation will be: 16000000÷32÷115200=4.34. Due to only integer can be set for the timer, then the result will be 4 (which means for every 4 system clock cycles, the timer overflows once), and the error will be about 8.5%. Such significant error will induce abnormal communication. Since the system clock is fixed, frequency division is a must to get correct baud rate. If it is assumed the timer overflows every 5 clock cycles, then we have： 16000000 ÷115200 ÷5=27.78. With frequency divided by 28, the baud rate will be114285 then. The error is about 0.8% compared to 115200 which will not influence the UART communication usually. In the end, for higher baud rate, the frequency can be divided by other number less than 28

The frequency is divided by 32 by default for the chip which is the same as the standard 51 chip. To change the default setting, users can set UDE=1 to enable the function. DNUM indicated the number the frequency will be divided by. For more information you may refer to the description for register UDCKS.

Caution: when UDE=1, SMOD(PCON[7]) will be invalid.

16.2 Register Description

Table 16-2-1 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/W	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 17-1-1-1
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 16-2-2 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

Table16-2-3 Register UDCKS

D8H	7	6	5	4	3	2	1	0
UDCKS	UDE	-	-	DNUM[4：0]				
R/W	R/W	-	-	R/W				
Initial Value	0	-	-	0	0	0	0	0

Bit number	Bit Symbol	Description
7	UDE	Rapid baud rate setting enable control, 1 enables it ***Note：*** *When UDE=0, the configuration of the baud rate for UART0 remains the same; while UDE=1，the baud rate for UART0 is configured by DNUM.*
6~5	-	-
4~0	DNUM	Rapid baud rate setting register, valid only when UDE=1 mode0：for transmission, it is a must that DNUM>=2； for receiving , it is a must that DNUM>=3 mode1：for transmission, it is a must that DNUM>=0； for receiving , it is a must that DNUM>=6 mode2：for transmission, it is a must that DNUM>=2； for receiving , it is a must that DNUM>=7 mode3：for transmission, it is a must that DNUM>=0； for receiving , it is a must that DNUM>=0

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 Characteristics. 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
- Supports standard/fast/high speed mode, with the speed 100K/400K/3MKBit/s respectively

17.3 I²C Function Description

I²C modules supports I²C standard bus Characteristics. 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.

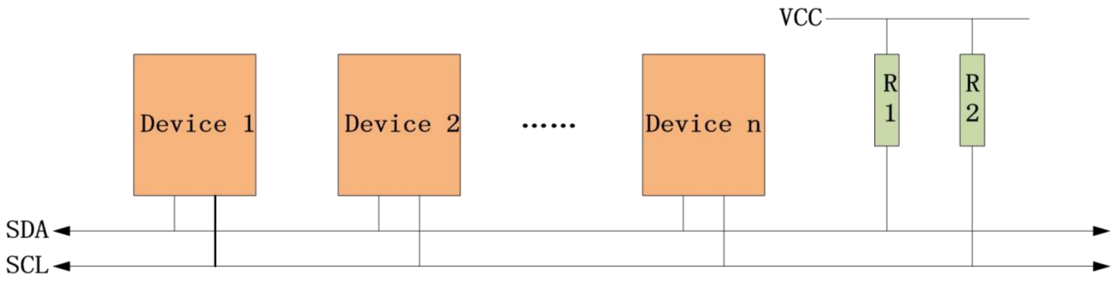


Figure 17-3-1 I2C Bus Interconnection

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

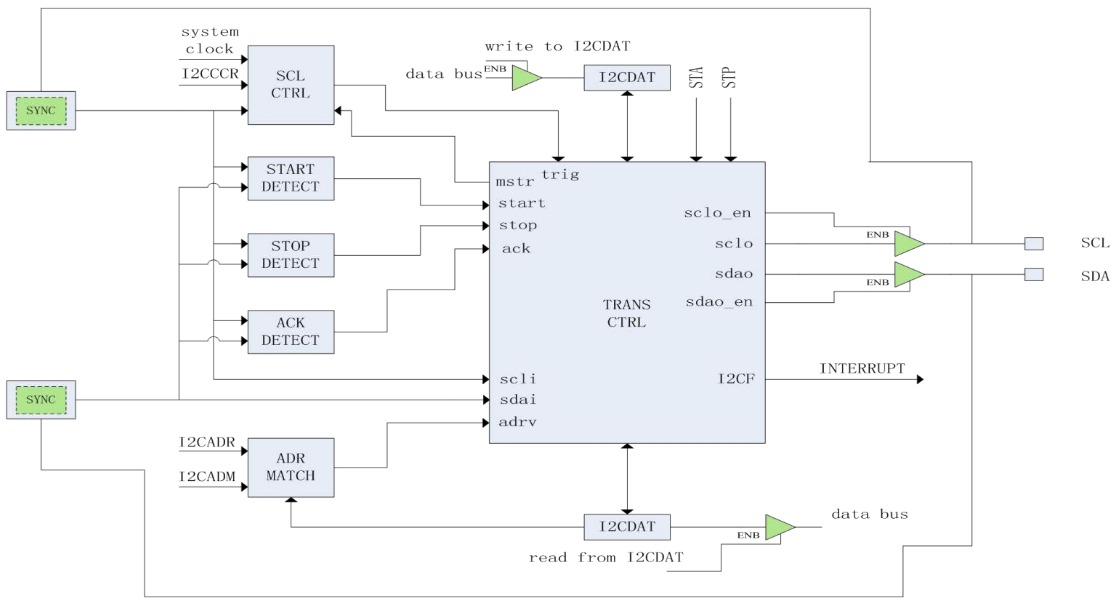


Figure 17-3-2 I2C Module Schematic

- **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.

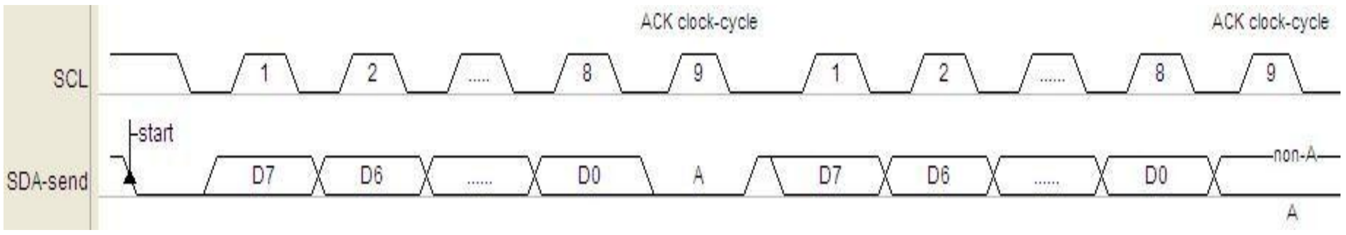


Figure 17-3-3 I2C Bus Data Transmission Format

- **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^{2}C 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.

- **I^{2}C Clock setting**

When I^{2}C is set as slave, the master outputs SCL clock, and it has nothing to do with the slave’s clock configuration. As for the slave, SCL must remain low for at least 6.5 system clock cycles periods and high for at least 2.5 system clock cycles periods. In the end, the frequency of SCL sent by external master can be at most 1/9 of the system clock.

17.4 I^{2}C Communication Pin Mapping

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

17.5 Register Description

Table 17-5-1 Register I2CCON

C0H	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	0	0	0	0	0	0	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^2C START signal transfer control,valid when it is 1, it will be cleared automatically when START signal detected
4	STP	I^2C STOP signal transfer control, valid when it is 1, it will be cleared 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^2C ACK signal transfer control, 1 enables it Note： When I^2C 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 GCE must be set to 0.
0	STFE	When STFE=1, I2CF will be set to 1 if I^2C module detects the START signal

Table 17-5-2 Register I2CADR

C1H	7	6	5	4	3	2	1	0
I2CADR	GCE	I2CADR[6:0]						
R/W	R/W	R/W						
Initial Value	0	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	I^2C 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

C2H	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 I^2C module detects the STOP signal
6~0	I2CADM	I^2C 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 of 2 division of the I2C operating clock, i.e. 000: Fsample=Fi2cclk 001: Fsample=Fi2cclk/2 010: Fsample=Fi2cclk/4 ... 111: Fsample=Fi2cclk/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. Fscl= Fsample /(I2CCKD +1)) Remarks. 1. When SMPDIV= 0, if the setting I2CCKD is less than 9, it will be calculated by 9 automatically. 2. When SMPDIV>0, if I2CCKD is set less than 7, it will be calculated by 7 automatically. Remark. 1. When I2CCCR[7:5] = 0, if the value less than 9 is written to I2CCCR[4:0], it will be automatically calculated by the value of 9. 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 I2CTable 17-5-5 Register I2CDAT

C4H	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

C5H	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

C6H	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 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	-	-	-	-	-	-	I2CS	
R/W	-	-	-	-	-	-	R/W	
Initial Value	-	-	-	-	-	-	0	

Bit number	Bit Symbol	Description
7~1	-	-

0	I2CS	I^{2}C pin selection
		0：pin P31 for SCL, pin P30 for SDA 1：pin P02 for SCL, pin P03 for SDA

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 define
#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 define
#define   GCE(N)                (N<<7) //N = 0~1
//I2CFLG define
#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;                                                                   // Open global interrupt
/********** Select I2C port ***********************************************************/
//   I2CIOS = 0;                        // Select P30,P31 as I2C communication pins
//   P30F = 3 | (1<<7);                 // Set P30 as I2C SDA and turn on the upper
//   P31F = 3 | (1<<7);                 // Set P31 as I2C SCL and turn on the upper

     I2CIOS = 1;                        // Select P03,P02 as I2C communication pins
      P03F = 3|(1<<7);                  // Set P03 as I2C SDA and turn on the upper
      P02F = 3|(1<<7);                  // Set P02 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 = 0x4c;                                         // 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
/********** Select I2C port ****************************************************************/
//   I2CIOS = 0;                           // Select P30,P31 as I2C communication pins
//   P30F = 3 | (1<<7);                    // Set P30 as I2C SDA and turn on the upper
//   P31F = 3 | (1<<7);                    // Set P31 as I2C SCL and turn on the upper

     I2CIOS = 1;                           // Select P03,P02 as I2C communication pins
       P03F = 3|(1<<7);                    // Set P03 as I2C SDA and turn on the upper

       P02F = 3|(1<<7);                    // Set P02 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 = 0x4c;                                     // 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 Routines

As a slave, it supports the host to write or read data with the following procedure.

```
------------------------------------------------------------------------------------------------------------------------------------------------
#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
/**********Select I2C port*****************************************************************/
//  I2CIOS = 0;                     // Select P30,P31 as I2C communication pins
//  P30F = 3 | (1<<7);              // Set P30 as I2C SDA and turn on the upper
//  P31F = 3 | (1<<7);              // Set P31 as I2C SCL and turn on the upper

    I2CIOS = 1;                     // Select P03,P02 as I2C communication pins
      P03F = 3|(1<<7);              // Set P03 as I2C SDA and turn on the upper
      P02F = 3|(1<<7);              // Set P02 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
      INT4EN = 1;                              // I2C interrupt on
      while(1)
      {
      }
}
```
----

18 PWM

18.1 PWM Function Introduction

CA51F5 series chip can include at most 6 channels PWM outputs. PWM period and duty cycle can be configured with 16 bit range. PWM1(LED_D0)/PWM2(LED_D1) also support cascade LED drive. The scanning frequency is greater than 400Hz/S with data transmission speed at 800Kbps. It can be used to control WS2812 or similar chips, which meet the requirements for monochromatic or colorful LED strip.

18.2 PWM Function Description

There is a 16-bit counter for each PWM channel and the cycle is set by register PWMnDIV. Register PWMnDUT sets the corresponding PWM's duty cycle. PWM is enabled by register PWMEN with each bit of it corresponds to one channel in PWM. Whether the PWM pin outputs reversed signal is set by PWMnTOG. The clock source for each PWM channel can be selected by corresponding PWMnCKS of register PWMnCON, with the frequency division set by PWMnCKD independently.

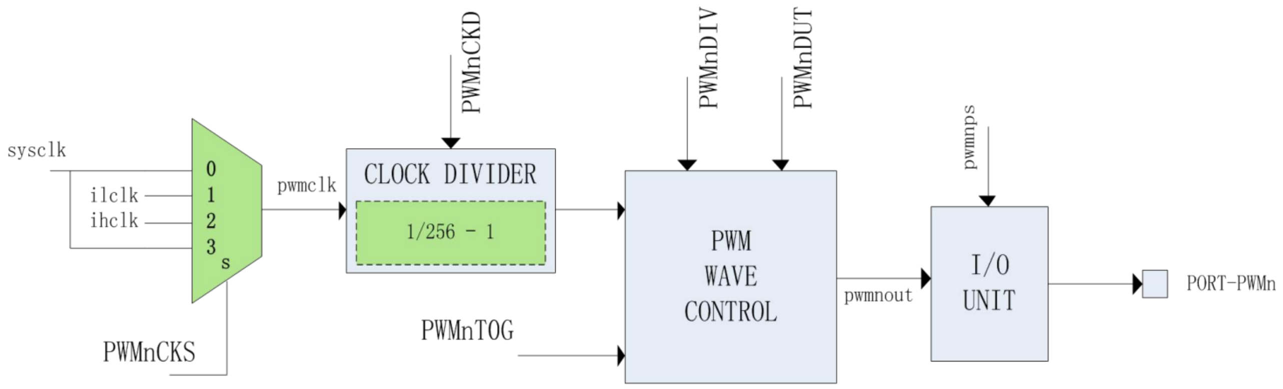


Figure 18-2-1 PWM Schematic

Note:

The 'n' in PWMnDIV, PWMnDUT and etc indicates 0/1/2/3/4/5, which stands for the control/configuration register for PWM channel 0/1/2/3/4/5.

- **PWM output waveform**

When PWM is enabled, PWM starts counting. When the count is less than or equal to PWMnDUT, PWM pin outputs high level signal (PWMnTOG=0); when the count value is greater than PWMnDUT, PWM pin outputs low level signal (PWMnTOG=0). When the count equals to PWMnDIV, a PWM cycle completes and the counter will starts counting again. PWM interrupt occurs at the same time.

When PWMnDIV>PWMnDUT>0, the PWM waveform is as follows.

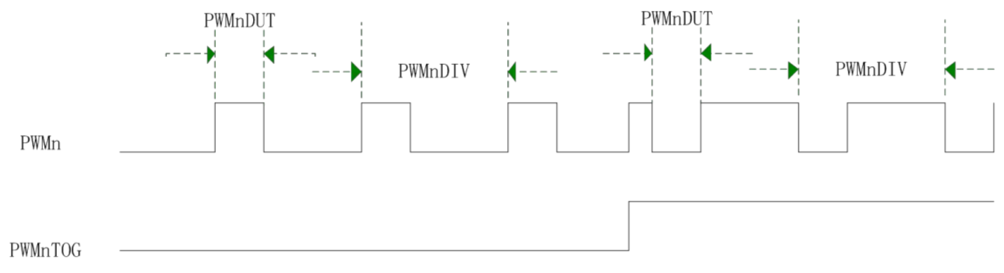


Figure 18-2-2 PWM Output Waveform

When PWMnDIV=0, PWM pin will output the PWM clock directly. If PWMnCKD=0 then, PWM pin outputs the clock source selected. On the other hand, if PWMnCKD ！ =0, PWM pin will output the clock source with frequency divided by (PWMnCKD+1)When PWMnDIV ≠ 0 and PWMnDUT=0, PWM pin outputs low/high level (PWMnTOG ＝ 0/1); when PWMnDUT>=PWMnDIV>0, PWM pin outputs high/low level signal(PWMnTOG＝0/1).

- **PWM interrupt**

PWM interrupt can be enabled by PWMnIE of register PWMnCON. When the PWM counter's count reaches the peak (=PWMnDIV), the interrupt occurs. There are 6 interrupt flags for 6 channels in register PWMIF.

- **Cascade LED drive with single wire**

PWM1/PWM2 channel supports cascade LED drive with single wire and drive sequence diagram for cascade LED is shown Figure 18-2-3 as follows. **Figure 18-2-3 Sequence Diagram for Cascade LED**

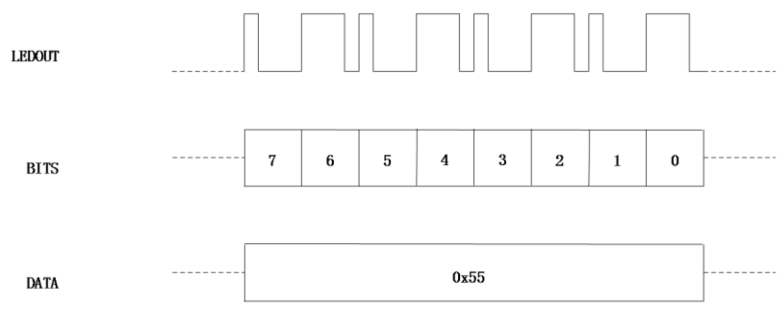


Figure 18-2-3 shows the bit coding.

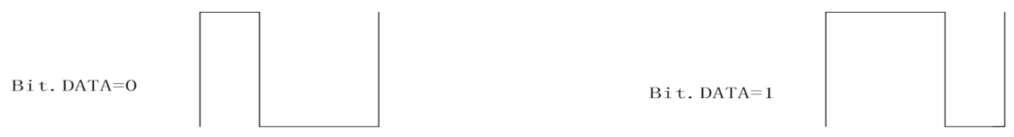


Figure 18-2-4 Bit Coding

For the cascade LED sequence diagram, the high level time for bit 0 code is set by PWMnDUT(n=1/2) while the high level time for bit 1 code is set by LEDUTH. The cycle time for one bit is set by PWMnDIV. When PWMnMOD≠0, the cascade LED drive is enabled and LEDAT0/LEDAT1 is the data register for LEDn(n=0/1) respectively. When LEFn(n=0/1), users can write LED data to LEDAT0/LEDAT1. Writing LEDAT0/LEDAT1 will starts the data transfer for LED drive. When LEDn(n=0/1) transmitter is transferring data, LEBSYn(n=0/1) is set to1; while when the transmitter is idle, LEBSYn becomes 0. There is one-byte buffer for LED transmitter. When there is data in both data register and buffer register, LEFn(n=0/1) is set to 1. When all the data in buffer register is has been sent, the data from data register will be loaded to it automatically. At the same time, LEFn(n=0/1) is set to 0. LEFn(n=0/1)=0 shows LEDAT0/LEDAT1 is ready for new data. When PWMnMOD≠0, it implies a waiting time will be inserted after PWMnMOD byte data is sent. The waiting time will be set by LEWTM.

When PWMnPOL=1(n=1/2), the data of LEDAT0/LEDAT1 will be reversed. For example, if 01010101B is written, the data sent is actually 10101010B.

18.3 PWM Register Description

Table 18-3-1 Register PWMEN

90H	7	6	5	4	3	2	1	0
PWMEN	-	-	PWM5EN	PWM4EN	PWM3EN	PWM2EN	PWM1EN	PWM0EN
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	PWM5EN	PWM5 enable control bit, 1 active
4	PWM4EN	PWM4 enable control bit, 1 active
3	PWM3EN	PWM3 enable control bit, 1 active
2	PWM2EN	PWM2 enable control bit, 1 active
1	PWM1EN	PWM1 enable control bit, 1 active
0	PWM0EN	PWM0 enable control bit, 1 active

Table 18-3-2 Register PWMCON

B9H	7	6	5	4	3	2	1	0
PWM0CON	PWM0IE	PWM0TOG	-	-	-	-	PWM0CKS[1:0]	
R/W	R/W	R/W	-	-	-	-	R/W	
Initial value	0	0	-	-	-	-	0	0
BAH	7	6	5	4	3	2	1	0
PWM1CON	PWM1IE	PWM1TOG	PWM1MOD[2:0]			PWM1POL	PWM1CKS[1:0]	
R/W	R/W	R/W	R/W			R/W	R/W	
Initial value	0	0	0	0	0	0	0	0
BBH	7	6	5	4	3	2	1	0
PWM2CON	PWM2IE	PWM2TOG	PWM2MOD[2:0]			PWM2POL	PWM2CKS[1:0]	

R/W	R/W	R/W	R/W			R/W	R/W	
Initial value	0	0	0	0	0	0	0	0
BCH	7	6	5	4	3	2	1	0
PWM3CON	PWM3IE	PWM3TOG	-	-	-	-	PWM3CKS[1:0]	
R/W	R/W	R/W	-	-	-	-	R/W	
Initial value	0	0	-	-	-	-	0	0
BDH	7	6	5	4	3	2	1	0
PWM4CON	PWM4IE	PWM4TOG	-	-	-	-	PWM4CKS[1:0]	
R/W	R/W	R/W	-	-	-	-	R/W	
Initial value	0	0	-	-	-	-	0	0
BEH	7	6	5	4	3	2	1	0
PWM5CON	PWM5IE	PWM5TOG	-	-	-	-	PWM5CKS[1:0]	
R/W	R/W	R/W	-	-	-	-	R/W	
Initial value	0	0	-	-	-	-	0	0

Bit number	Bit symbol	description
Note: PWM1/PWM2 can drive cascaded LEDs, so add control bits related to LED driving.		
7	PWMnIE	PWMn counter overflow interrupt enable control bit, 1 valid
6	PWMnTOG	PWMn output inverted enable control bit, 1 valid
5~3	PWMmMOD	When PWMm is used as LED driver, the number of bytes continuously sent configuration register, 0 means PWMm is not used as LED driver, 1~7 means PWMm pauses 1 time for every 1~7 bytes of data sent Remark: 1. m means 1/2 . 2. Refer to LEWTM for detailed use.
2	PWMmPOL	When PWMm is used as LED driver, send data reversal enable control bit, 1 is valid Remarks: 1. the value of the corresponding PWMmPOL is meaningful when PWMmMOD!=0. 2. When PWMmPOL=1, if the corresponding LEDAT=01010101B, then what will actually be sent will be 10101010B.
1~0	PWMnCKS	PWM operating clock selection bit 01: IRCL 10: IRCH Others: System Clock

Table 18-3-3 Register PWMCKD

B1H	7	6	5	4	3	2	1	0
PWM0CKD	PWM0CKD[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
B2H	7	6	5	4	3	2	1	0
PWM1CKD	PWM1CKD[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
B3H	7	6	5	4	3	2	1	0
PWM2CKD	PWM2CKD[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
B4H	7	6	5	4	3	2	1	0
PWM3CKD	PWM3CKD[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
B5H	7	6	5	4	3	2	1	0
PWM4CKD	PWM4CKD[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
B6H	7	6	5	4	3	2	1	0
PWM5CKD	PWM5CKD[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0

Bit number	Bit symbol	description
7~0	PWMnCKD	PWM operating clock pre-division configuration register 00H: No frequency division 01H： 2 divisions 02H： 3 divisions ...... FEH： 255 divisions FFH： 256 divisions

Table 18-3-4 Register PWMDIVL、PWMDIVH

A9H	7	6	5	4	3	2	1	0
PWM0DIVL	PWM0DIV[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
AAH	7	6	5	4	3	2	1	0
PWM0DIVH	PWM0DIV[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
ABH	7	6	5	4	3	2	1	0
PWM1DIVL	PWM1DIV[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
ACH	7	6	5	4	3	2	1	0
PWM1DIVH	PWM1DIV[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
ADH	7	6	5	4	3	2	1	0
PWM2DIVL	PWM2DIV[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
AEH	7	6	5	4	3	2	1	0
PWM2DIVH	PWM2DIV[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
AFH	7	6	5	4	3	2	1	0
PWM3DIVL	PWM3DIV[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
A4H	7	6	5	4	3	2	1	0
PWM3DIVH	PWM3DIV[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
A5H	7	6	5	4	3	2	1	0
PWM4DIVL	PWM4DIV[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
A6H	7	6	5	4	3	2	1	0
PWM4DIVH	PWM4DIV[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
A7H	7	6	5	4	3	2	1	0
PWM5DIVL	PWM5DIV[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
9AH	7	6	5	4	3	2	1	0
PWM5DIVH	PWM5DIV[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0

Bit number	Bit symbol	description
15~0	PWMnDIV	PWMn Cycle Configuration Register

Table 18-3-5 Register PWMDUTL、PWMDUTH

9BH	7	6	5	4	3	2	1	0
PWM0DUTL	PWM0DUT[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
9CH	7	6	5	4	3	2	1	0
PWM0DUTH	PWM0DUT[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
9DH	7	6	5	4	3	2	1	0
PWM1DUTL	PWM1DUT[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
9EH	7	6	5	4	3	2	1	0
PWM1DUTH	PWM1DUT[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
9FH	7	6	5	4	3	2	1	0
PWM2DUTL	PWM2DUT[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
91H	7	6	5	4	3	2	1	0
PWM2DUTH	PWM2DUT[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
92H	7	6	5	4	3	2	1	0
PWM3DUTL	PWM3DUT[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
PWM3DUTH	PWM3DUT[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
94H	7	6	5	4	3	2	1	0
PWM4DUTL	PWM4DUT[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
PWM4DUTH	PWM4DUT[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
96H	7	6	5	4	3	2	1	0
PWM5DUTL	PWM5DUT[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
PWM5DUTH	PWM5DUT[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
Bit number	Bit symbol		description					
15~0	PWMnDUT		PWMn Duty Cycle Configuration Register					

Table 18-3-6 Register PWMIF

B7H	7	6	5	4	3	2	1	0
PWMIF	-	-	PWMIF5	PWMIF4	PWMIF3	PWMIF2	PWMIF1	PWMIF0
R/W	-	-	R	R	R	R	R	R
Initial value	-	-	0	0	0	0	0	0
Bit number	Bit symbol	description						
7~6	-	-						
5	PWMIF5	PWM5 interrupt flag bit, 1 valid, write 1 to clear 0						
4	PWMIF4	PWM4 interrupt flag bit, 1 valid, write 1 to clear 0						
3	PWMIF3	PWM3 interrupt flag bit, 1 valid, write 1 to clear 0						
2	PWMIF2	PWM2 interrupt flag bit, 1 valid, write 1 to clear 0						
1	PWMIF1	PWM1 interrupt flag bit, 1 valid, write 1 to clear 0						
0	PWMIF0	PWM0 interrupt flag bit, 1 valid, write 1 to clear 0						

Table 18-3-7 Register LEDAT

D7H	7	6	5	4	3	2	1	0
LEDAT0	LEDAT0[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
C7H	7	6	5	4	3	2	1	0
LEDAT1	LEDAT1[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
Bit number	Bit symbol	description						
7~0	LEDATx	LED driver data Remark. 1. x means 0/1, LED0/LED1 corresponds to PWM1/PWM2. 2. The data of LEDAT is sent in the order from MSB to LSB.						

Table 18-3-8 Register LEDUTL、LEDUTH

8060H	7	6	5	4	3	2	1	0
LEDUTL	LEDUT[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
8061H	7	6	5	4	3	2	1	0
LEDUTH	LEDUT[15:8]							

R/W	R/W							
Initial value	0	0	0	0	0	0	0	0

Bit number	Bit symbol	description
15~0	LEDUT	*LED transmit data "1" duty cycle configuration register* *Remarks:* *1. in the cascaded LED drive waveform, the period of each bit of data is determined by the corresponding PWMmDIV, while the duty cycle of data "1" is determined by LEDUT and the duty cycle of data "0" is determined by PWMmDUT.* *2. If LEDAT=01010101B, and the corresponding PWMPOL=1, then the actual data sent in the order of BIT7-BIT6-BIT5-BIT4-BIT3-BIT2-BIT1-BIT0 is 1-0-1-0-1-0-1-1-0, and the duty cycle of BIT7/BIT5/BIT3/BIT1 is determined by the duty cycle of BIT7/BIT5/BIT3/BIT1 is determined by LEDUT and the duty cycle of BIT6/BIT4/BIT2/BIT0 is determined by the corresponding PWMDUT, i.e., the LEDUT starts after PWMPOL.* *3. the duty cycle of the data "1" of LED1/LED2 is determined by the same LEDUT.*

Table 18-3-9 Register LEWTML、LEWTMH

CEH	7	6	5	4	3	2	1	0
LEWTML	LEWTM[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
CFH	7	6	5	4	3	2	1	0
LEWTMH	LEWTM[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0

Bit number	Bit symbol	description
15~0	LEWTM	LED pause time configuration register, used in combination with PWMmMOD configuration register Remarks: 1. After each PWMmMOD byte data is sent, pause (LEWTM+1) PWM operating clock and then go to the next transmission. 2. The pause time of LED1/LED2 are both determined by the same LEWTM.

Table 18-3-10 Register LEFLG

BFH	7	6	5	4	3	2	1	0
LEFLG	-	LEBSY1	-	-	-	LEBSY0	-	-
R/W	-	R	-	-	-	R	-	-
Initial value	-	0	-	-	-	0	-	-

Bit number	Bit symbol	Description
7	-	
6	LEBSY1	LEDAT1 data sending busy flag, 1 means the data in LEDAT1's data cache has not been sent at this time, 0 means all the data has been sent
5~4	-	-
3	-	
2	LEBSY0	LEDAT1 data sending busy flag, 1 means the data in LEDAT0's data cache has not been sent at this time, 0 means all the data has been sent
1~0	-	-

18.4 PWM Control Example

◆ **Single channel PWM output**

Taking PWM0 for example, the clock source for PWM is IRCH （ frequency of IRCH is 16MHz ） , the output clock's frequency is 30K with duty cycle 30%, the program is like:

```
    -------------------------------------------------------------------------------------------
    //PWMxCON
  #define PWMIE(N)           (N<<7)     //N=0-1
  #define PWMTOG(N)          (N<<6)     //N=0-1
  #define PWMMOD(N)          (N<<3)     //N=0-7
  #define PWMPOL(N)          (N<<2)     //N=0-1
  #define PWMCKS_SYS         0
  #define PWMCKS_IL          1
#define PWMCKS_IH            2

  #define IHCKE        (1<<7)
  void PWM_init(void)
  {
    P32F = 5;       //set P32 as PWM
    pin CKCON |= IHCKE;
                    //enables IRCH
    clock

    PWM0DIVH = 0x02;      //set the DIV,
    16000000/30000=0x215 PWM0DIVL = 0x15;

    PWM0DUTH = 0x00;      //set DUT, the duty cycle
    is 30% PWM0DUTL = 0xA0;

    PWM0CON = PWMIE(0) | PWMTOG(0) | PWMCKS_IH;      //set IRCH as the clock source for PWM , with
  interrupt disabled
    PWM0CKD = 0;               //set the frequency division, 0 indicates that there is no frequency division

    PWMEN |= (1<<0);           //enables PWM0
  }
 -------------------------------------------------------------------------------------------
```

◆ **Sing wire cascade LED strip drive example**

Taking PWM1 for example, to drive 8 level RGB LED, and make RGB LED change its color cyclically, the program is like:

```
 -------------------------------------------------------------------------------------------
    //PWMxCON
  #define PWMIE(N)           (N<<7)     //N=0-1
  #define PWMTOG(N)          (N<<6)     //N=0-1
```

```
#define PWMMOD(N)            (N<<3)      //N=0-7
#define PWMPOL(N)            (N<<2)      //N=0-1
#define PWMCKS_SYS       0
#define PWMCKS_IL            1
#define PWMCKS_IH            2


//LEFLG
#define LEF0       (1<<3)
#define LEBSY0   (1<<2)

void PWM_init(void)
{
    P33F = 5;                //set P33 as the
    PWM pin CKCON |= IHCKE;
                             //enables IRCH
    clock

    PWM1DIVH = 0;       //set the cycle period for
    one bit PWM1DIVL = 20;

    PWM1DUTH = 0;      //set the time for bit 0
    code PWM1DUTL = 6;

    LEDUTH = 0;           //set the time for bit 1
    code LEDUTL = 13;

    PWM1CON = PWMIE(0) | PWMTOG(0) | PWMMOD(3) | PWMPOL(0) | PWMCKS_IH;       //set IRCH as the  clock
                                                                   source for PWM, insert pause
                                                                   time after 3-byte data is sent
    PWM1CKD = 0;       //set the frequency division, when it is 0, it means there is no frequency division

    LEDWTMH = 0;        //set the
    pause time LEDWTML = 50;

    PWMEN |= (1<<1);         //enables PWM1
}
code unsigned char LED_DAT[][3] = //LED data table
{
    {0xff,0x00,0x00},
    {0xff,0xff,0x00},
    {0x00,0xff,0x00},
    {0x00,0xff,0xff},
    {0x00,0x00,0xff},
    {0xff,0x00,0xff},
  };
void main(void)
  {
      unsigned char i;
      unsigned char
      color_index=0;
      PWM_init();
      while(1)
      {
            for(i=0;i<24;i++)      //3 byte data for RGB LED's each level, 24 bytes for 8 levels totally
            {
                  while(LEFLG & LEF0);      //wait to write LEDDAT

                  LEDAT0 = LED_DAT[color_index][i%3]; //write LED data to LEDAT register

            }
            color_index ++;               //change color
            cyclically if(color_index> 6) color_index =0;
            Delay1S();                          //wait for one second
            .......
      }
  }
```

19 Digital-to-analog converters (DAK)

19.1 Function Introduction

There are two DA output channels in this module. For each DA channel, there are 32 voltage levels configurable. This module is designed to output different key information for the main chip to detect by ADC, when CA51F5 chip is used as a touch chip.

DAK can be enabled by AKnE in AKnCON. When AKnE=1 and the corresponding pin is selected as DAK function, the corresponding pin will output the voltage set by users according to AKnS.

Note：n=0/1 corresponds to DAK0/DAK1 respectively.

19.2 Register Description

Table 19-2-1 Register AKCON

DEH	7	6	5	4	3	2	1	0
AK0CON	AK0E	-	-	AK0S[4:0]				
R/W	R/W	-	-	R/W				
Initial value	0	-	-	0	0	0	0	0
DFH	7	6	5	4	3	2	1	0
AK1CON	AK1E	-	-	AK1S[4:0]				
R/W	R/W	-	-	R/W				
Initial value	0	-	-	0	0	0	0	0

Bit number	Bit symbol	Description
7	AKnE	DAKn channel enable control register, 1 enables the channel
6~5	-	-
4~0	AKnS	DAKn output voltage configuration DAKn output voltage: (AKnS+1)÷ 32)×Vdd, with error ±1%

19.3 DAK Control Example

Taking DAK0 for example, if users set DAK0 pin to output 1/2 Vdd, the program is like:

```
------------------------------------------------------------------------------------------
//AKxCON definition
#define AKE(N)            (N<<7)
#define AKS(N)            N                  //N=0-31

void DAK0_init(void)
{
        P00F = 3;
        AK0CON = AKE(1) | AKS(15);
}
------------------------------------------------------------------------------------------
```

20 Touch Key

20.1 Function Introduction

The touch function module of CA51F5 series chip has superior anti-interference performance and can pass EFT, CS and other tests. Cx can directly affect the touch sensitivity, the smaller the Cx capacitance, the lower the sensitivity, the larger the capacitance, the higher the sensitivity.
For applications with low-power requirements, a mechanism is also designed to allow the chip to work normally even in STOP mode.

20.2 Main Features

- Great anti-jamming performance which meets the EMC(CS) Standard
- Supports 13 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
- Waking up threshold configurable in STOP mode

20.3 Architecture

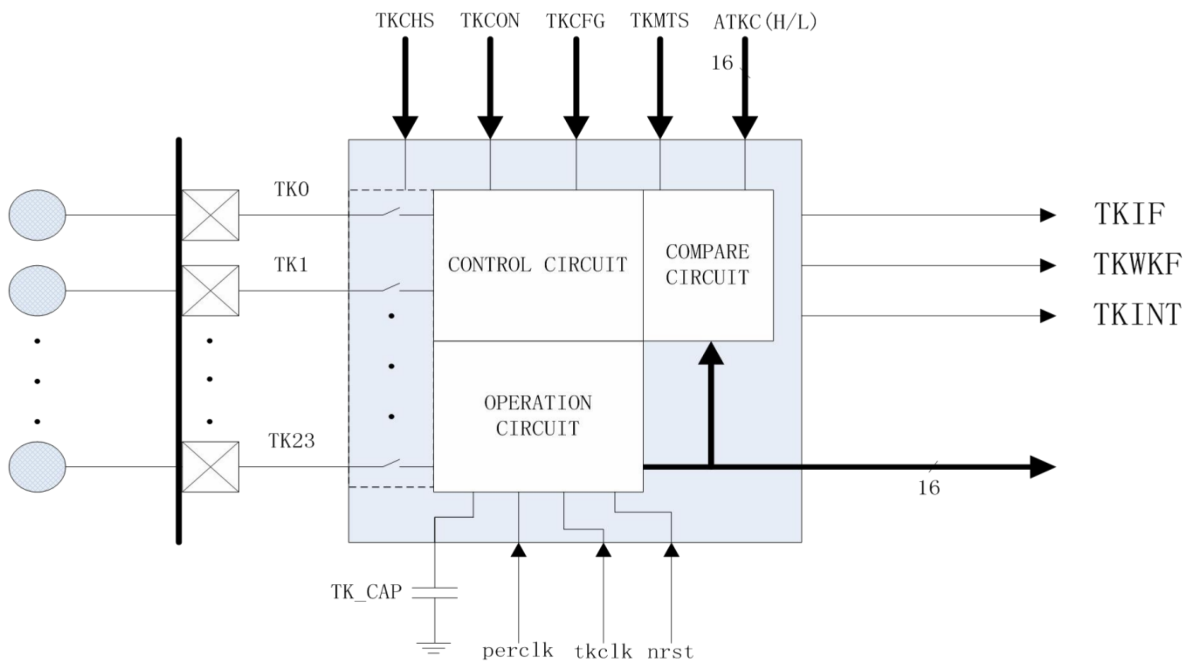


Figure 20-3-1 Touch Key Module Architecture

20.4 Function Description

20.4.1 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；the timing can be set by register TKMTS.

Remark:

The "n" in the TKnCHS etc. register means 0/1/2/3/4/5.

20.4.2 Touch Key Clock Frequency Division

The clock source for electrode charging/discharging is IRCH with frequency divided by 4 (which means the its frequency is 4 MHz). The clock's frequency 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.

20.4.3 Low Power consumption Mode

As long as the clock source IRCH and low speed clock (IRCL) 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.

20.4.4 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=0~12, corresponding to TK0~TK12) to enable the common LED drive function, it should be noted that the corresponding touch pin function must be turned on. After the common LED enable, TLDATx (x=0~12, corresponding to LED0~LED12) can control each LED independently. com pin is P01.

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. A diagram of the operating phases is shown below.

Important reminder: In the touch pin and LED driver pin multiplexing mode applications, due to the existence of the diode junction capacitance of the LED itself, the junction capacitance of different types of LEDs vary greatly, and this junction capacitance in the LED light and off the performance may be inconsistent (especially white LEDs 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.

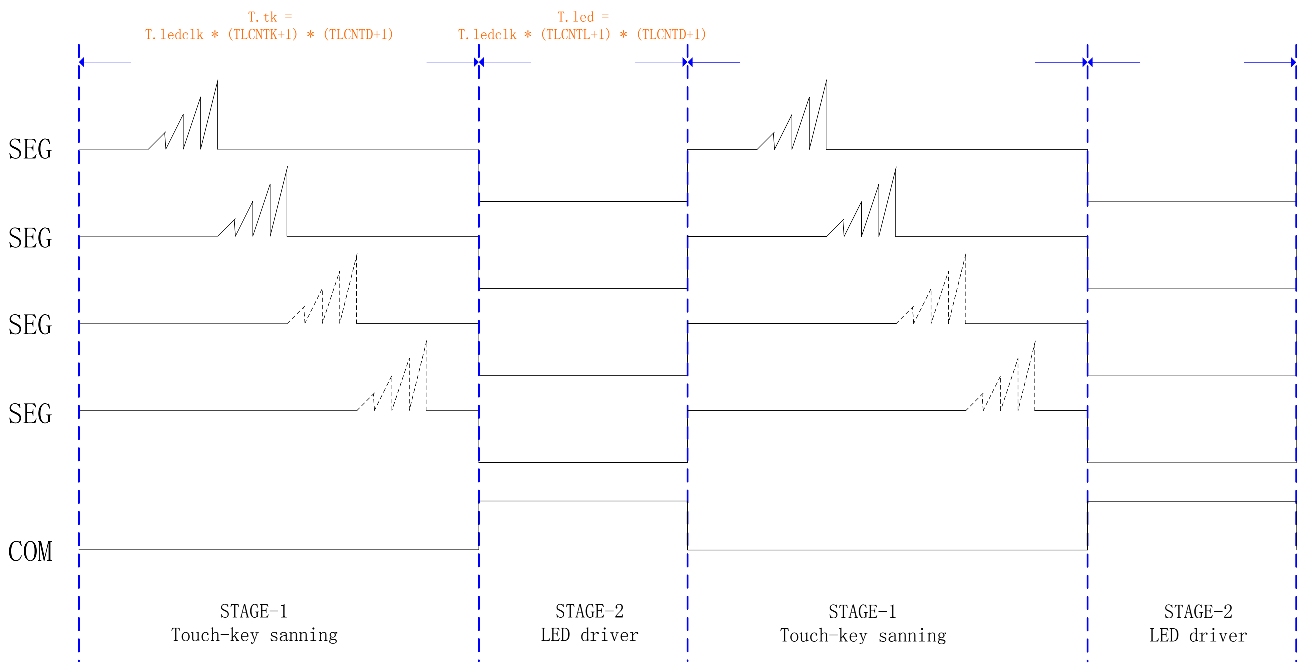


Figure 20-4-5 Touch common LED driver schematic

20.5 Register Description

Table 20-5-1 Register TKCON

F8H	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 threshold voltage … 7： minimum threshold voltage

Table 20-5-2 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: Suspend* *01: Output low* *10: Output compensation* *Notes.* *1. This function is only available for the pin that is configured for the touch button function.* *2. If the shared LED drive function is enabled, this function will be disabled for the pin selected as LED drive.*
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 supply selection 0: Select external power supply 1: Select internal op-amp output
0	TKCVS	Charge reference voltage selection 0： Select external voltage reference 1： Select internal voltage reference

Table 20-5-3 Register TKCKS

8102H	7	6	5	4	3	2	1	0
TKCKS	-	-	-	-	-	-	-	TKSIL
R/W	-	-	-	-	-	-	-	R/W
Initial value	-	-	-	-	-	-	-	0
Bit number	Bit symbol	description						
7~1	-	-						
0	TKSIL	Touch key sampling clock selection 0: Select the 16M quadrature (4M) clock 1: Select slow (131K) clock						

Table 20-5-4 Register TKCFG

F9H	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		Touch clock crossover selection 000: No frequency division 001: 2-division 010: 3-division ... 111: 8-division					
4~0	TKTMS		*External modulation capacitor discharge time setting* *Discharge time = TKTMS x 128 x charge/discharge clock cycles* *With TKDIV=0, the discharge time range is: 32us - 992us* *Note: TKTMS cannot be set to 0.*					

Table 20-5-5 Register TKMTS

FAH	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) × 128 ×IRCL cycle period, Since the frequency of IRCL is 131kHz,, so the start time ranges from 1ms to 256ms.					

Table 20-5-6 Register TKCHS

FCH	7	6	5	4	3	2	1	0
TK0CHS	POL0	NPOL0	-	-	TKPS0[3:0]			
R/W	R/W	R/W	-	-	R/W			
Initial value	0	0	-	-	0	0	0	0
FDH	7	6	5	4	3	2	1	0
TK1CHS	POL1	NPOL1	-	-	TKPS1[3:0]			
R/W	R/W	R/W	-	-	R/W			
Initial value	0	0	-	-	0	0	0	0
FEH	7	6	5	4	3	2	1	0
TK2CHS	POL2	NPOL2	-	-	TKPS2[3:0]			
R/W	R/W	R/W	-	-	R/W			

Initial value	0	0	-	-	0	0	0	0
FFH	7	6	5	4	3	2	1	0
TK3CHS	POL3	NPOL3	-	-	TKPS3[3:0]			
R/W	R/W	R/W	-	-	R/W			
Initial value	0	0	-	-	0	0	0	0
F1H	7	6	5	4	3	2	1	0
TK4CHS	POL4	NPOL4	-	-	TKPS4[3:0]			
R/W	R/W	R/W	-	-	R/W			
Initial value	0	0	-	-	0	0	0	0
F2H	7	6	5	4	3	2	1	0
TK5CHS	POL5	NPOL5	-	-	TKPS5[3:0]			
R/W	R/W	R/W	-	-	R/W			
Initial value	0	0	-	-	0	0	0	0

Bit number	Bit symbol	description
7	POLn	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	NPOLn	ATKnN direction setting for threshold comparison CA51F5 105 Table 20-5-5 Register ATKC 0：interrupt when the data collected is less than the threshold 1：interrupt when the data collected is greater than the threshold
5~4	-	-
3~0	TKPSn	Channel n selection 000000：disable TK0~TK23 000001：TK0 selected 000010：TK1 selected 000011：TK2 selected ...... 011000：TK23 selected 011001：Internal reference capacitor selected

Table 20-5-7 Register ATKC

F3H	7	6	5	4	3	2	1	0
ATK0CL	ATK0C[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
F4H	7	6	5	4	3	2	1	0
ATK0CH	ATK0C[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
F5H	7	6	5	4	3	2	1	0
ATK1CL	ATK1C[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
ATK1CH	ATK1C[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
F7H	7	6	5	4	3	2	1	0
ATK2CL	ATK2C[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
E9H	7	6	5	4	3	2	1	0
ATK2CH	ATK2C[15:8]							

R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
EAH	7	6	5	4	3	2	1	0
ATK3CL	ATK3C[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
EBH	7	6	5	4	3	2	1	0
ATK3CH	ATK3C[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
ECH	7	6	5	4	3	2	1	0
ATK4CL	ATK4C[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
EDH	7	6	5	4	3	2	1	0
ATK4CH	ATK4C[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
EEH	7	6	5	4	3	2	1	0
ATK5CL	ATK5C[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
EFH	7	6	5	4	3	2	1	0
ATK5CH	ATK5C[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0

Bit number	Bit symbol	description
15~0	ATKnC	Compare threshold setting register, when TWKE=1, ATKC0~ATKC5 will be compared with TK0MS~TK5MS automatically

Table 20-5-8 Register AKTN

8050H	7	6	5	4	3	2	1	0
ATK0NL	ATK0N[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
8051H	7	6	5	4	3	2	1	0
ATK0NH	ATK0N[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
8052H	7	6	5	4	3	2	1	0
ATK1NL	ATK1N[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
8053H	7	6	5	4	3	2	1	0
ATK1NH	ATK1N[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
8054H	7	6	5	4	3	2	1	0
ATK2NL	ATK2N[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
8055H	7	6	5	4	3	2	1	0
ATK2NH	ATK2N[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
8056H	7	6	5	4	3	2	1	0

ATK3NL	ATK3N[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
8057H	7	6	5	4	3	2	1	0
ATK3NH	ATK3N[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
8058H	7	6	5	4	3	2	1	0
ATK4NL	ATK4N[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
8059H	7	6	5	4	3	2	1	0
ATK4NH	ATK4N[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
805AH	7	6	5	4	3	2	1	0
ATK5NL	ATK5N[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
805BH	7	6	5	4	3	2	1	0
ATK5NH	ATK5N[15:8]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0

Bit number	Bit symbol	description
15~0	ATKnN	Compare threshold setting register, when TWKE=1, ATK0N~ATK5N will be compared with TK0MS~TK5MS automatically

Table 20-5-9 Register TKMS

E1H	7	6	5	4	3	2	1	0
TK0MSL	TK0MS[7:0]							
R/W	R							
Initial value	0	0	0	0	0	0	0	0
E2H	7	6	5	4	3	2	1	0
TK0MSH	TK0MS[15:8]							
R/W	R							
Initial value	0	0	0	0	0	0	0	0
E3H	7	6	5	4	3	2	1	0
TK1MSL	TK1MS[7:0]							
R/W	R							
Initial value	0	0	0	0	0	0	0	0
E4H	7	6	5	4	3	2	1	0
TK1MSH	TK1MS[15:8]							
R/W	R							
Initial value	0	0	0	0	0	0	0	0
E5H	7	6	5	4	3	2	1	0
TK2MSL	TK2MS[7:0]							
R/W	R							
Initial value	0	0	0	0	0	0	0	0
E6H	7	6	5	4	3	2	1	0
TK2MSH	TK2MS[15:8]							
R/W	R							
Initial value	0	0	0	0	0	0	0	0
E7H	7	6	5	4	3	2	1	0
TK3MSL	TK3MS[7:0]							
R/W	R							
Initial value	0	0	0	0	0	0	0	0
D9H	7	6	5	4	3	2	1	0

TK3MSH	TK3MS[15:8]							
R/W	R							
Initial value	0	0	0	0	0	0	0	0
DAH	7	6	5	4	3	2	1	0
TK4MSL	TK4MS[7:0]							
R/W	R							
Initial value	0	0	0	0	0	0	0	0
DBH	7	6	5	4	3	2	1	0
TK4MSH	TK4MS[15:8]							
R/W	R							
Initial value	0	0	0	0	0	0	0	0
DCH	7	6	5	4	3	2	1	0
TK5MSL	TK5MS[7:0]							
R/W	R							
Initial value	0	0	0	0	0	0	0	0
DDH	7	6	5	4	3	2	1	0
TK5MSH	TK5MS[15:8]							
R/W	R							
Initial value	0	0	0	0	0	0	0	0

Bit number	Bit symbol	description
15~0	TKnMS	Touch key sampling data register

Table 25-5-10 Register TKIF

FBH	7	6	5	4	3	2	1	0
TKIF	-	-	TKIF5	TKIF4	TKIF3	TKIF2	TKIF1	TKIF0
R/W	-	-	R	R	R	R	R	R
Initial value	-	-	0	0	0	0	0	0

Bit number	Bit symbol	description
7~6	-	-
5	TKIF5	TK data collection interrupt flag for the 6th channel When TWKE=1, TKIF5 implies the TK5MS exceeds the ATK5C or ATK5N threshold , cleared when 1 is written to it
4	TKIF4	TK data collection interrupt flag for the 5th channel When TWKE=1, TKIF5 implies the TK5MS exceeds the ATK5C or ATK5N threshold , cleared when 1 is written to it
3	TKIF3	TK data collection interrupt flag for the 4th channel When TWKE=1, TKIF5 implies the TK5MS exceeds the ATK5C or ATK5N threshold , cleared when 1 is written to it
2	TKIF2	TK data collection interrupt flag for the 3th channel When TWKE=1, TKIF5 implies the TK5MS exceeds the ATK5C or ATK5N threshold , cleared when 1 is written to it
1	TKIF1	TK data collection interrupt flag for the 2th channel When TWKE=1, TKIF5 implies the TK5MS exceeds the ATK5C or ATK5N threshold , cleared when 1 is written to it
0	TKIF0	TK data collection interrupt flag for the 1th channel When TWKE=1, TKIF5 implies the TK5MS exceeds the ATK5C or ATK5N threshold , cleared when 1 is written to it

Table 20-5-11RegisterTKMAXF

805CH	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	TKMXF5	1 indicates that TKxMS exceeds ATKxC threshold, while 0 indicates TKxMS does not exceed ATKxC threshold. The polarity is set by POL5； if TWKE=1, setting TKMXFx to 1 will set TKIFx as well; the software can not do anything to it
4	TKMXF4	1 indicates that TKxMS exceeds ATKxC threshold, while 0 indicates TKxMS does not exceed ATKxC threshold. The polarity is set by POL4； if TWKE=1, setting TKMXFx to 1 will set TKIFx as well; the software can not do anything to it
3	TKMXF3	1 indicates that TKxMS exceeds ATKxC threshold, while 0 indicates TKxMS does not exceed ATKxC threshold. The polarity is set by POL3； if TWKE=1, setting TKMXFx to 1 will set TKIFx as well; the software can not do anything to it
2	TKMXF2	1 indicates that TKxMS exceeds ATKxC threshold, while 0 indicates TKxMS does not exceed ATKxC threshold. The polarity is set by POL2； if TWKE=1, setting TKMXFx to 1 will set TKIFx as well; the software can not do anything to it
1	TKMXF1	1 indicates that TKxMS exceeds ATKxC threshold, while 0 indicates TKxMS does not exceed ATKxC threshold. The polarity is set by POL1； if TWKE=1, setting TKMXFx to 1 will set TKIFx as well; the software can not do anything to it
0	TKMXF0	1 indicates that TKxMS exceeds ATKxC threshold, while 0 indicates TKxMS does not exceed ATKxC threshold. The polarity is set by POL0； if TWKE=1, setting TKMXFx to 1 will set TKIFx as well; the software can not do anything to it

Table20-5-12 Register TKMINF

805DH	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	TKMNF5	1 indicates that TKxMS exceeds ATKxN threshold, while 0 indicates TKxMS does not exceed ATKxN threshold. The polarity is set by NPOL5； if TWKE=1, setting TKMXFx to 1 will set TKIFx as well; the software can not do anything to it
4	TKMNF4	1 indicates that TKxMS exceeds ATKxN threshold, while 0 indicates TKxMS does not exceed ATKxN threshold. The polarity is set by NPOL4； if TWKE=1, setting TKMXFx to 1 will set TKIFx as well; the software can not do anything to it
3	TKMNF3	1 indicates that TKxMS exceeds ATKxN threshold, while 0 indicates TKxMS does not exceed ATKxN threshold. The polarity is set by NPOL3； if TWKE=1, setting TKMXFx to 1 will set TKIFx as well; the software can not do anything to it
2	TKMNF2	1 indicates that TKxMS exceeds ATKxN threshold, while 0 indicates TKxMS does not exceed ATKxN threshold. The polarity is set by NPOL2； if TWKE=1,

		setting TKMXFx to 1 will set TKIFx as well； the software can not do anything to it
1	TKMNF1	1 indicates that TKxMS exceeds ATKxN threshold, while 0 indicates TKxMS does not exceed ATKxN threshold. The polarity is set by NPOL1 ； if TWKE=1, setting TKMXFx to 1 will set TKIFx as well； the software can not do anything to it
0	TKMNF0	1 indicates that TKxMS exceeds ATKxN threshold, while 0 indicates TKxMS does not exceed ATKxN threshold. The polarity is set by NPOL0 ； if TWKE=1, setting TKMXFx to 1 will set TKIFx as well； the software can not do anything to it

Table20-5-13 Register TLEN

8106H	7	6	5	4	3	2	1	0
TLEN0	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
8111H	7	6	5	4	3	2	1	0
TLEN1	-	-	-	TLEN12	TLEN11	TLEN10	TLEN9	TLEN8
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W
Initial value	-	-	-	0	0	0	0	0

Remark:
1. TLENx=1 (x=0,1,2,... ,12), LEDx to be enabled, the corresponding TKx pin must be selected for the touch button function. 2.
2. The user can choose any or all of the pins within the touch button pin range as the common LED driver pins.

Bit number	Bit symbol	description
7~5 (TLEN1)	-	-
4~0 (TLEN1)	TLEN12~TLEN8	LED12~LED8 Enabled, 1 active
7~0 (TLEN0)	TLEN7~TLEN0	LED7~LED0 Enabled, 1 active

Table20-5-14Register TLDAT

8107H	7	6	5	4	3	2	1	0
TLDAT0	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
8112H	7	6	5	4	3	2	1	0
TLDAT1	-	-	-	TLDAT12	TLDAT11	TLDAT10	TLDAT9	TLDAT8
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W
Initial value	-	-	-	0	0	0	0	0

Bit number	Bit symbol	description
7~0 (TLDAT1)	-	-
7~0 (TLDAT1)	TLDAT12~TLDAT8	LED12~LED8 data, 1Table shows the effective level of drive
7~0 (TLDAT0)	TLDAT7~TLDAT0	LED7~LED0 data, 1Table shows the effective level of drive

Table20-5-15Register 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 20-5-16 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: Table indicates that the time set by the user is enough for the touch button scanning 1: Table indicates that the time set by the user is not enough, the timer has finished counting and the channel has not been scanned Remark. This bit is the hardware auto-setting and auto-zeroing bit. Considering the convenience of users, this bit can also be written 1 to clear 0 by software.
6	TLKOV	Touch key scanning stage timer count full flag, 1Table shows count full, software write 1 to clear 0
5	TLLOV	LED driver stage timer count full flag, 1Table shows count full, software write 1 to clear 0
4~0	-	-

Table20-5-17Register 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/4 Others: Off

Table 20-5-18 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 configurationRegister* *T.tk = T.ledclk x (TLCNTK+1) x (TLDIV+1)* *Notes.* *1. T.tk, Table shows the time of the touch key scan phase; T.ledclk, Table shows the period of the operating clock of the LED driver circuit.* *2. the time of touch button scanning phase, the minimum must meet the time of 1 way touch button scanning, the maximum only needs to meet the time of 6 way touch button scanning, even if the user selects all 20 way channels, but the circuit only scans 6 way at most each time, and then scans the other way in time.* *3. it takes about 1ms to scan each channel of touch buttons.*

Table 20-5-19 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	$T_{.led}$= $T_{.ledclk}$x (TLCNTL+1) x (TLDIV+1) x N *Notes.* *1. T.led, Table shows the time of LED drive phase; N, Table shows the number of all enabled LED drives. the total number of valid COMs.* *In addition to ensuring sufficient time for each stage, the user must also consider the touch key scanning sensitivity and the frequency of the LED driving waveform, which must generally be above 64Hz.* *3. to connect the second point, the user generally do not set too many LED drive, but theoretically you can choose all 20 roads, as long as the control time.* *Translated with www.DeepL.com/Translator (free version)*

Table 20-5-20 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 divider register Frequency division multiplier is (TLDIV+1)

Table 20-5-21 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 P0.1 Other: off

20.6 Touch Key Control Example

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

21 Low Voltage Detection(LVD)

21.1 Function Introduction

Low voltage Detection(LVD) is used to monitor the chip's power supply VDD, with detectable range 1.7V~4.8V. When VDD is lower than the voltage set, either interrupt or reset occurs.

Figure 21-1-1 shows the architecture of LVD.

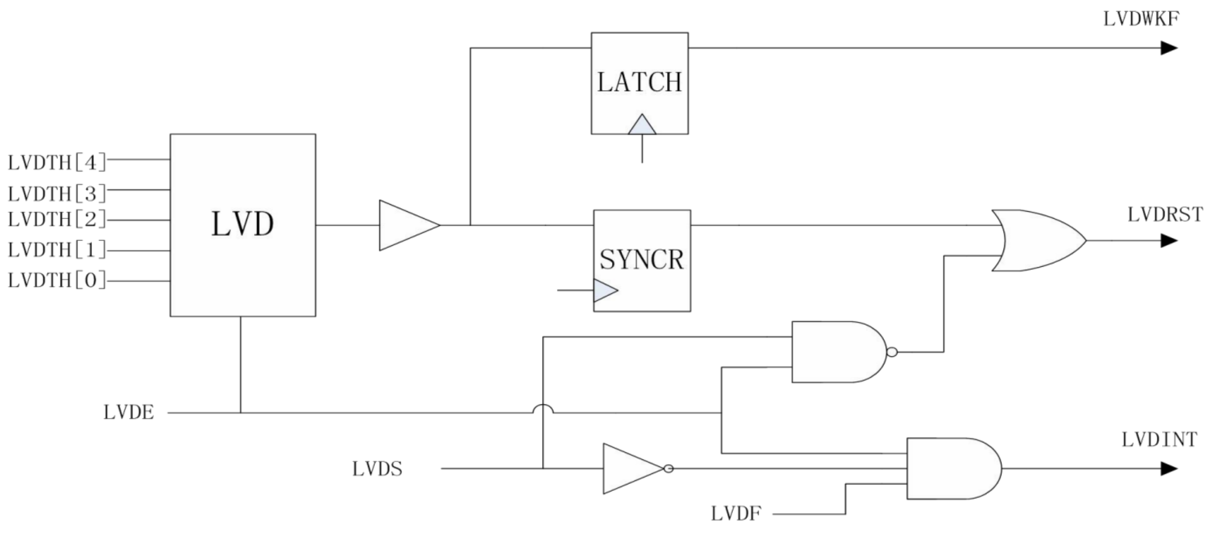


Figure 21-1-1 LVD Schematic

21.2 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. As a result, if VDD is still lower than the trigger voltage set by users after the reset, it will be reset forever. Similarly, the interrupt will occur repeatedly if VDD is still lower than the trigger voltage set by users after the interrupt.

21.3 Register Description

Table 21-3-1 Register LVDCON

E8H	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 00000： 1.7V 00001： 1.8V 00010： 1.9V …… 11101： 4.6V 11110： 4.7V 11111： 4.8V

Table 21-3-2 Register LVDCFG

8105H	7	6	5	4	3	2	1	0
LVDCFG	LVSEL	XLVSEL	-	-	-	-	-	-
R/W	R/W	R/W	-	-	-	-	-	-
Initial value	0	0	-	-	-	-	-	-

Bit number	Bit symbol	description
7	LVSEL	LVD detection voltage selection 0: Detect VDD voltage 1： Detect external voltage
6	XLVSEL	LVD detects external voltage pin selection 0： Detect P0.0 voltage 1： Detect P0.1 voltage
5~0	-	-

21.4 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        13
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 6
{
        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        13
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
}
```

22 Program Download and Simulation

22.1 Program Download

CA51F5 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*.

22.2 Online Simulation

CA51F5 Series chip supports online simulation. Chip can communicate with the emulator with IIC interface. The default port for IIC is P30(IIC SDA) and P31(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.

23 Electrical Characteristics

23.1 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	-45	125	℃
CPU working frequency	-	16	MHz

Note: When the parameters exceed the limits above, the working status of the chip is unpredictable which may lead to severe damage to the chip. Working in such environment for a long time will influence the reliability of the chip.

23.2 DC Electrical Characteristics

Chip parameters	symbol	Operating Voltage	Minimum	Typical	Maximum	unit	condition
Operating current	Iop1	VDD=1.8V		1.28		mA	System clock is IRCH (16MHz), 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		1.50			
		VDD=5V		1.55			
	Iop3	VDD=1.8V		19.2		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		20.2			
		VDD=5V		20.9			
STOP Mode Current	Istp	VDD=1.8V		5.1		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		5.3			
		VDD=5V		5.7			
IDLE Mode Current	Iidl1	VDD=1.8V		0.537		mA	The system clock is set to IRCH (16MHz), other 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 enters sleep mode, and CPU enters IDLE mode.
		VDD=3.3V		0.629			
		VDD=5V		0.641			
	Iidl3	VDD=1.8V		11		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
		VDD=3.3V		11.6			
		VDD=5V		12.1			

							enters IDLE mode.
IO port input high voltage (Smit mode on)	Vhi1	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)	Vhi2	VDD=1.8V		0.5*VDD	VDD	V	-
		VDD=3.3V					
		VDD=5V					
IO port input low voltage (Smit mode on)	Vlo1	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)）	Vlo2	VDD=1.8V	0	0.5*VDD		V	-
		VDD=3.3V					
		VDD=5V					
IO port push current	Ipu	VDD=3.3V	-	4.27	-	mA	IO set to push-pull output mode, drive capability set to maximum, Vol=VDD-0.3V
		VDD=5V	-	6.07	-		
IO port	Iol	VDD=3.3V	-	11.33	-	mA	IO set to push-pull output mode, drive capability set to maximum, Vol = GND + 0.3V
		VDD=5V	-	16.05	-		
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

22.3 AC Electrical Characteristics

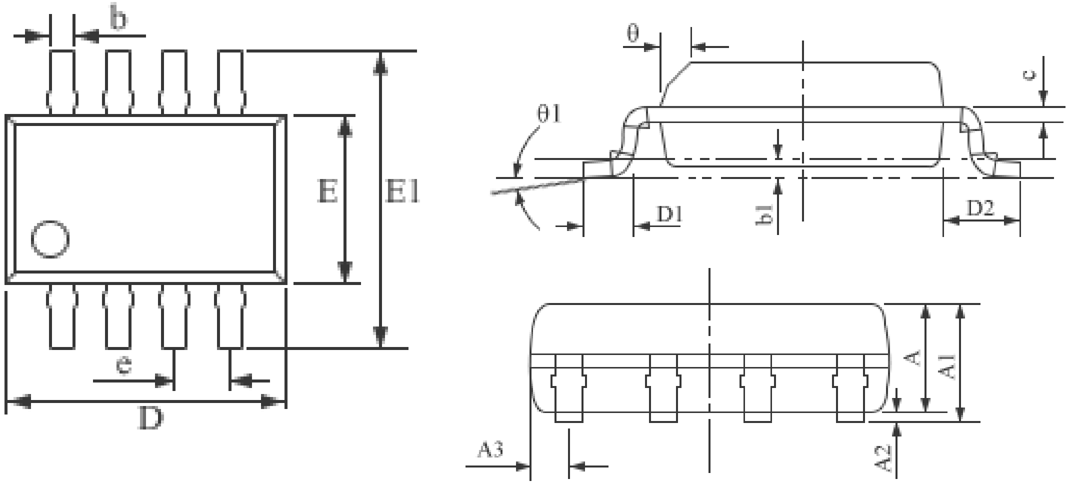
AC Electrical Characteristics (VDD=1.8-5.5V, TA=25℃, unless there are other explanations)

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
Time to start oscillation for IRCL	Trc1	-	50	-	us	IRCL frequency 131KHz
Time to start oscillation for IRCH	Trc2	-	10	-	us	IRCH frequency 16MHz
Time of the reset pulse	Trst	-	0.5	-	us	

Note：VDD=3.3V,TA=25 ℃, the factory frequency for internal high speed clock is 16MHz, with deviation less than 1%..

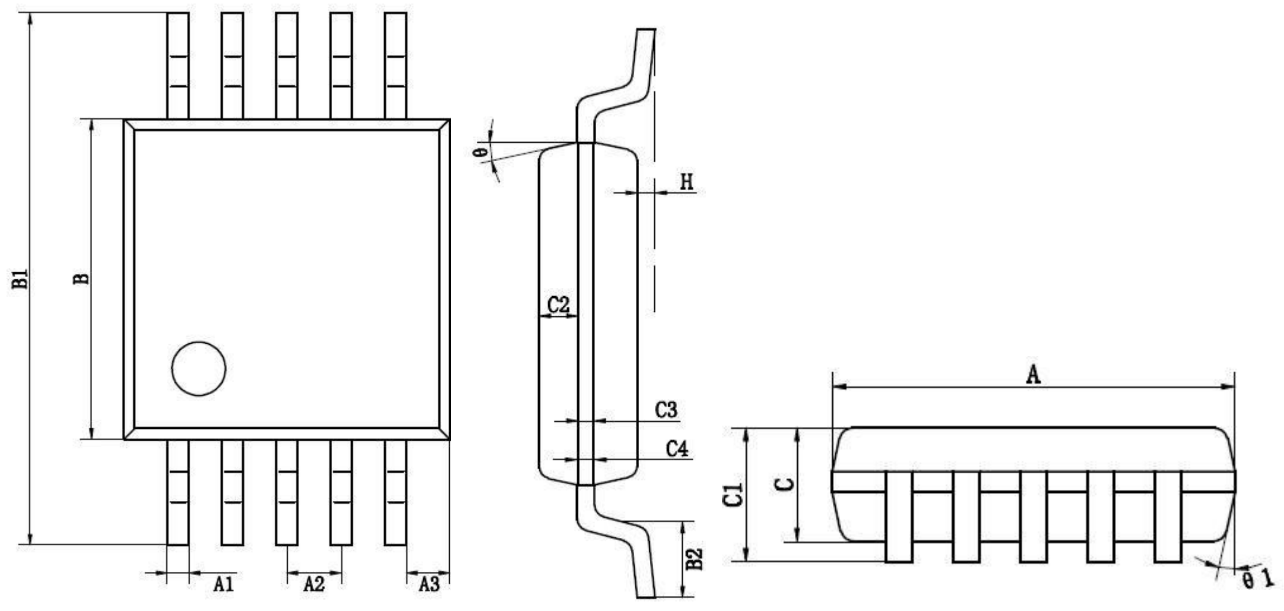
24 Package Type

Package(1)(SOP8)



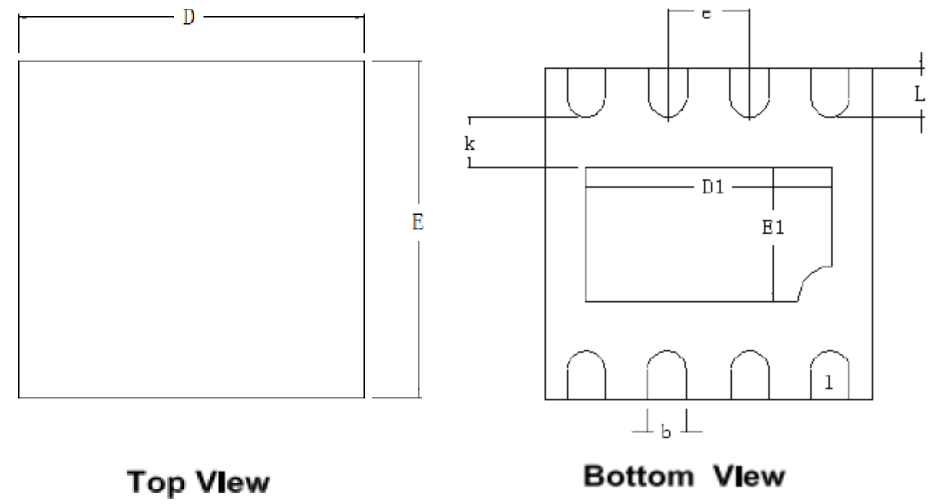
Sequence number	Minimum(mm)	Standard(mm)	Maximum(mm)
A	1.40	1.45	1.50
A1	1.55	1.60	1.65
A2	0.10	0.15	0.20
A3	0.50	0.535	0.540
b	0.354	0.406	0.504
b1	0.155	0.150	0.175
c	0.20	0.203	0.210
D	4.830	4.880	4.910
D1	0.610	0.660	0.710
D2	1.045	1.050	1.0505
e	———	1.270	———
E	3.810	3.910	3.96
E1	5.900	6.000	6.10

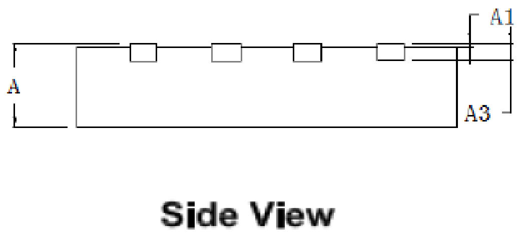
Package(2)(MSOP10)



Sequence number	Minimum(mm)	Standard(mm)	Maximum(mm)
A	2.90	3.00	3.10
A1	0.18	0.20	0.25
A2	0.50TYP		
A3	0.40TYP		
B	2.90	3.00	3.10
B1	4.70	4.90	5.10
B2	0.45	0.60	0.75
C	0.75	0.85	0.95
C1	---	---	1.10
C2	0.328TYP		
C3	0.152		
C4	0.15	0.19	0.23
H	0.02	---	0.15

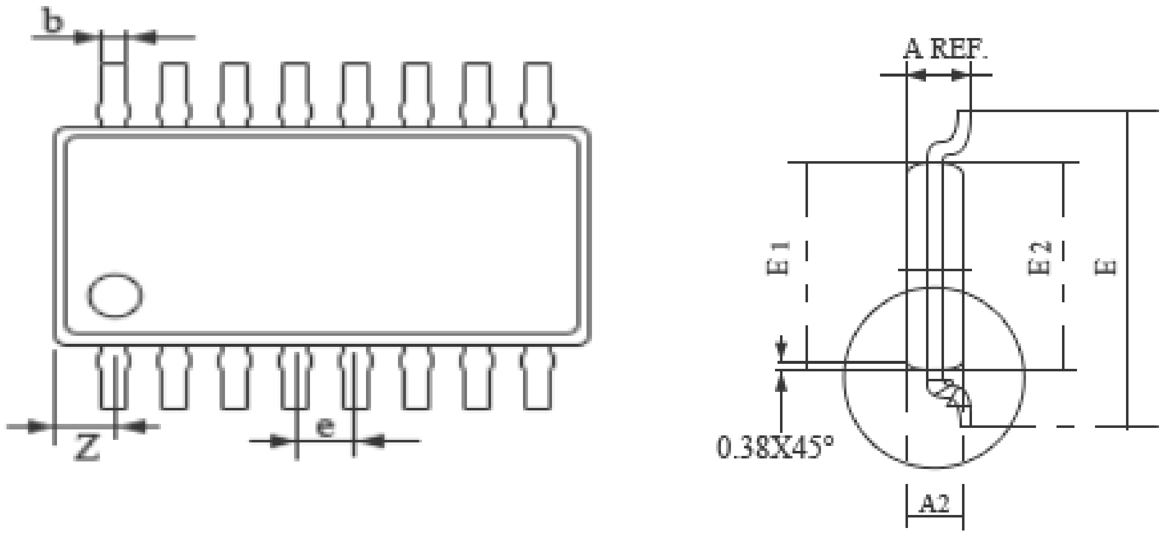
Package(3)(DFN8L 2X2MM)

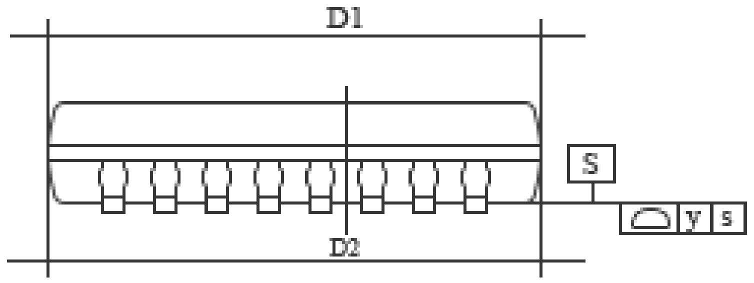




序号	最小值(mm)	标准值(mm)	最大值(mm)
A	0.70	0.75	0.80
A1	0	0.02	0.05
A3	0.203 REF		
D	1.85	2.00	2.15
E	1.85	2.00	2.15
D1	1.45	1.50	1.55
E1	0.75	0.80	0.85
K	0.30 BSC		
b	0.20	0.23	0.26
e	0.50		
L	0.30	0.35	0.40

Package(4)(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	————

25 Appendix

Appendix 1 Instruction Set Quick Reference Table

Instruction	Description	Description	Periodicity
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) +	1

		(Rn)	
ADDC A, direct	Add direct byte to A with Carry	(A) ← (A) + (C) + (direct)	1
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 ((SP)) ← (PC7-0) ((SP)) ← (PC15-8) (PC) ←addr15-0	2

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 Not Equal	(PC) ← (PC) + 3
		IF (Rn) <> data
		THEN
		(PC) ← (PC) + relative offset
		IF (Rn) < data
		THEN
		(C) ← 1
			2

		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 IF (bit) = 1 THEN (bit) ← 0 (PC) ← (PC) + rel	2
Pseudo-command			
ORG	Set program start address		
END	Mark the end of source code		
EQU	Define constants		

SET	Define integer numbers
DATA	Assign a value to the data address
BYTE	Assigning values to byte type symbols
WROD	Assigning values to word type symbols
BIT	Name the address of the bit
ALTNAME	Replace reserved words with custom names
DB	Load a contiguous block of memory with byte-type data
DW	Load a contiguous block of memory with word data
DS	Set aside a contiguous storage area or load specified bytes
INCLUDE	Insert a source file into the program
TITLE	Add a header row to the list file
NOLIST	No list file is generated during assembly
NOCODE	When the condition is compiled, the list is not generated if the condition is false