Built - in 12 Bit ADC / PWM / Touch Key / 1T 8051 16K Flash MCU

CA51F1 Series MCU
User Guide

REV 1.0

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

Table of Contents

1 Introduction

CA51F1 series chip is an 8-bit MCU based on the 1T 8051 core. Normally, it runs 10 times faster than the traditional 8051 chip with superior performance. The 16K Flash program memory embedded can be programmed for times and brings great convenience to software development. Not only retains the basic characteristics of the traditional 8051 chip, but also integrates functional modules such as 12Bit ADC, Touch Key, 16 Bit PWM, UART, I^2C, LED cascade control and low voltage detection (LVD). Supports three power-saving modes: IDLE, STOP and low-speed operation to adapt to applications with different power consumption requirements. Powerful functions and superior anti-interference performance make it widely used in various home appliance control, household lighting, Bluetooth speakers, toy control and various consumer electronic 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: 16K byte , can be erased and overwritten for times
  - Flash could be divided into program storage and data storage. Data space function is similar to EEPROM, which is used to store the data that needs to be saved when the power is off
  - RAM:256 bytes internal RAM
- **Operating Voltage**
  - Operating Voltage: 2.2 - 5.5V
- **Clock System**
  - Internal Low Speed RC Oscillator: 100KHz
  - Internal High Speed RC Oscillator: 16MHz，the accuracy is±2%(3.3V@25℃)
  - 2.2V-5.5V highest frequency 8MHz，2.7-5.5V highest frequency 16MHz
- **TMC**
  - Clock source is the built-in low-speed RC oscillator, and the minimum unit of interruption time is 512 low-speed RC oscillator clock cycles
  - Configurable interrupt time is 1-256 minimum unit time

- **Interrupt System**
  - 7 effective interrupt source
  - Two levels for interrupt priority which also supports interrupt nesting
  - 2 external interrupt source. For each external interrupt

- **Timer**
  - Two 16-bit general Timers: Timer 0, Timer 1

- **General Purpose IO(GPIO）**
  - Supports 6 GPIO at most and also supports push-pull,open-drain, pull-up, pull-down and high-impedance modes

- **Analog/Digital Converter(ADC)**
  - Supports 6 channel 12-bit SAR ADC
  - Supports 2 Reference Voltage: VDD and Internal Reference Voltage(1.5V)
  - When Internal Reference Voltage is selected, VDD could be measured as well

- **Touch Key**
  - Supports 5 touch channel at most
  - Internal charging and internal reference can be set
  - Built-in touch frequency hopping function, which can significantly improve the anti-voltage pulse injection (CS) performance
  - Excellent anti-jamming performance which conforms to EMC(CS) Standard
  - Support touch power saving mode

- **PWM**
  - Supports 3 channel PWM, any periods or duty cycles are configurable in 16 bits
  - Supports to output internal clock directly

- **LED cascade control**
  - Supports 1 channel LED cascade control, scanning frequency greater than 400Hz/S, data transmission speed 800Kbps
  - It supports direct control of WS2812 or similar driver chips, meeting the requirements of monochrome or colorful LED light strip products.

- **Reset Mode**
  - Supports variable reset sources： Hard , Soft Reset, Watch Dog Reset, LVD Reset, Power On/Down Reset

- **Low Voltage Detector（LVD）**
  - The detection voltage can be set to 2.7V, 3.3V, 3.7V and 4.2V
  - Low voltage reset/interrupt configurable

- **Watch Dog**
  - 27bit Watch Dog Timer, 16 bit precision configurable, with Watch Dog Reset and Interrupt configurable as well

- **UART**
  - Support 1 UART interface
  - Support 1 byte receive buffer

- **I^2C**
  - One I^2C port embedded which supports Master-Slave mode and Standard/Fast/High Speed mode as well

- **Program Download and Simulation**
  - Supports ISP and IAP
  - Supports simulation online

- **Low power consumption**
  - For STOP Mode, current<5uA
  - For IDLE Mode, current<20uA
  - For Low Speed Mode, current<40uA

- **Package Type：** SOP8/DFN8(2X3MM)

3 Model Selection Table

Table 3-1 CA51F1 Specific Models and Their Features

Models	Flash Storage [BYTE]	Internal Ram[BYTE]	Internal High Speed RC Oscillator	Internal Low Speed RC Oscillator	GPIO	UART	I²C	16 bit PWM Channels	Touch Key	12 bit ADC	LED cascade control	General 16 bit Timer	ISP	Simulation On Chip	Operating Voltage	Package Type
CA51F152S1	16K	256	√	√	6	√	√	3	5	6	1	2	√	√	2.2-5.5	SOP8
CA51F152N1	16K	256	√	√	6	√	√	3	5	6	1	2	√	√	2.2-5.5	DFN8 (2X3mm)

4 Block Diagram

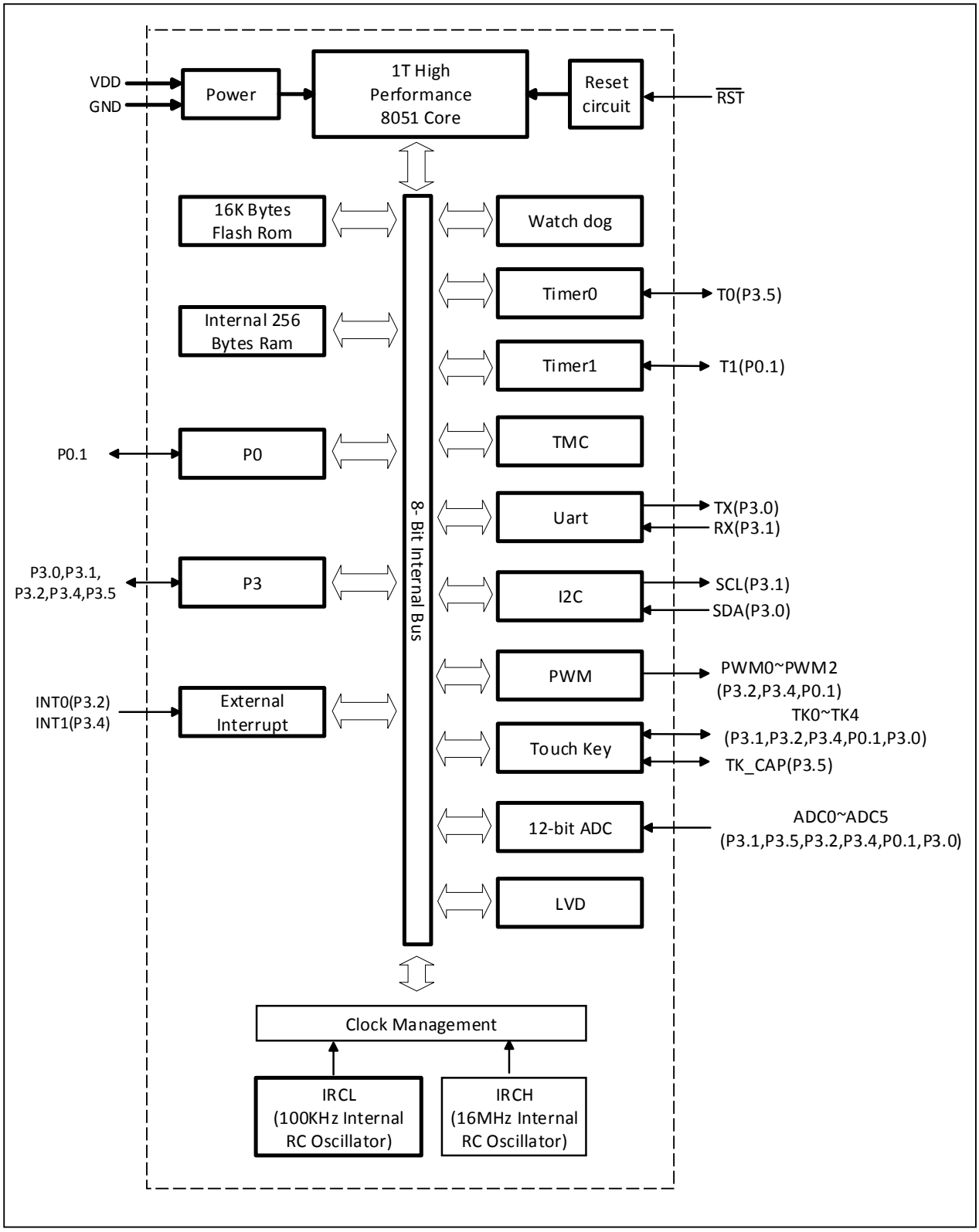


Figure 4-1-1 Chip Block Diagram

5 Pin Package and Description

5.1 Package Definition

Model：CA51F152S1/N1

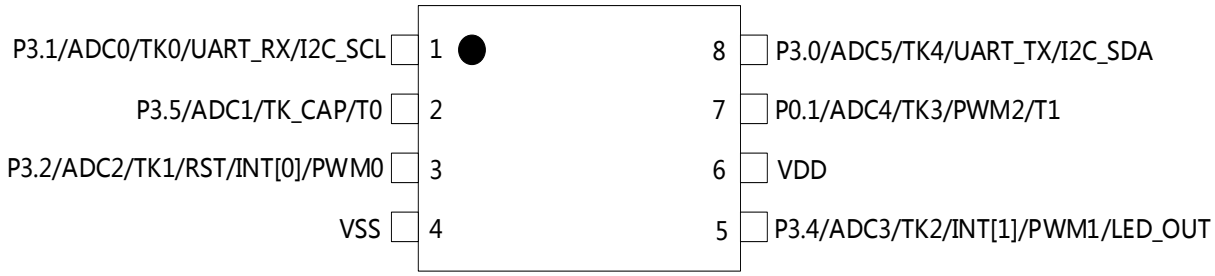


Figure 5-1-1 SOP8/DFN8 Package

5.2 Pin Description

Table 5-2-1 Pin Description

Pin Sequence Number SOP8/DFN8	Pin Name	Pin Function	Default Function
1	P3.1/ADC0/TK0/UART_RX/I2C_SCL	General bidirectional I/O port UART0_RX transmission port ADC analog channel input Touch button analog channel input I^2C clock transmission port	I^2C clock transmission port
2	P3.5/ADC1/TK_CAP/T0	General bidirectional I/O port ADC analog channel input Touch the external capacitance input port	General bidirectional I/O port
3	P3.2/ADC2/TK1/RST/INT[0]/PWM0	General bidirectional I/O port ADC analog channel input Touch button analog channel input Hardware reset pin PWM signal output	Hardware reset pin
4	VSS	Power ground pin	Power ground pin
5	P3.4/ADC3/TK2/INT[1]/PWM1/LED_OUT	General bidirectional I/O port External reset pin Touch button analog channel input ADC analog channel input PWM signal output LED cascade control	General bidirectional I/O port
6	VDD	Chip power supply pin	Power supply for the chip
7	P0.1/ADC4/TK3/PWM2/T1	General bidirectional I/O port ADC analog channel input Touch button analog channel input PWM signal output	General bidirectional IO port
8	P3.0/ADC5/TK4/UART_TX/I2C_SDA	General bidirectional I/O port UART0_TX transmission port ADC analog channel input I^2C data transmission port	I^2C data transmission port

6 Central Processing Unit(CPU)

6.1 CPU Introduction

Core of CA51F1 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 A is even 1：the number of 1 in ACC A is odd

Table 6-2-9 Register SPMAX

F3H	7	6	5	4	3	2	1	0
SPMAX	SPMAX[7:0]							
R/W	R	R	R	R	R	R	R	R
Initial value	0	0	0	0	0	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)

CA51F1 series offers both internal RAM(256 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.

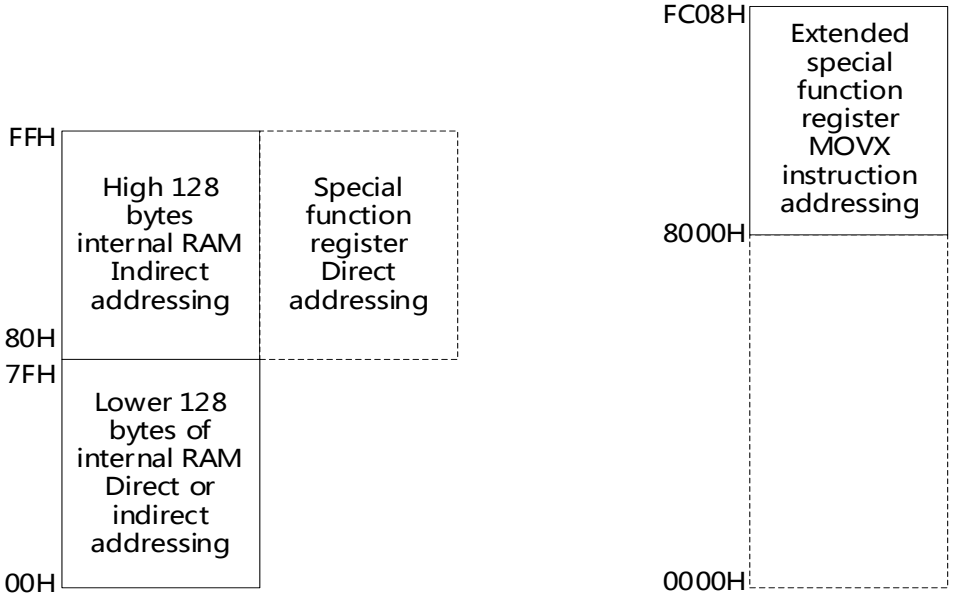


Figure 7-1-1 RAM Architecture

7.2 Special Function Register (SFR)

The SFR architecture of CA51F1 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 (SFR) 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	-	-	-	-
F0H	B	-	-	SPMAX	-	-	-	-
E8H	LVDCON	-	-	-	-	-	-	IDCODE
E0H	ACC	TK0MSL	TK0MSH	TK1MSL	TK1MSH	TK2MSL	TK2MSH	TK3MSL
D8H	UDCKS	TK3MSH	TK4MSL	TK4MSH	TK5MSL	TK5MSH	TKCKS	TKPWC
D0H	PSW	-	-	-	-	TMCON	TMSNU	I2CIOS
C8H	CKCON	CKDIV	IHCFG	-	-	-	-	TPCTL
C0H	I2CCON	I2CADR	I2CADM	I2CCR	I2CDAT	I2CSTA	I2CFLG	PWMEN
B8H	IP	PWM0CON	PWM1CON	PWM2CON	-	-	-	-
B0H	P3	PWM0CKD	PWM1CKD	PWM2CKD	-	-	-	-
A8H	IE	PWM0DIVL	PWM0DIVH	PWM1DIVL	PWM1DIVH	PWM2DIVL	PWM2DIVH	-
A0H	WDCON	WDFLG	WDVTHL	WDVTHH	-	-	-	-
98H	SCON	SBUF	-	PWM0DUTL	PWM0DUTH	PWM1DUTL	PWM1DUTH	PWM2DUTL
90H	-	PWM2DUTH	LEDUTL	LEDUTH	LEDWTML	LEDWTMH	LEDAT	LEFLG
88H	TCON	TMOD	TL0	TL1	TH0	TH1	IDLST	STPST
80H	P0	SP	DP0L	DP0H	DP1L	DP1H	PWCON	PCON

Due to limited SFR address space, CA51F1 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	-	P01F	-	-	-	-	-	-
8008H	-	-	-	-	-	-	-	-
8018H	P30F	P31F	P32F	-	P34F	P35F	-	-
8060H	ADCON	ADCFGL	ADCDL	ADCDH	ADCALL	ADCALH	-	-
8068H	SRELL	SRELH						
8120H	-	P01C	-	-	-	-	-	-
8128H	-	-	-	-	-	-	-	-
8138H	P30C	P31C	P32C	-	P34C	P35C	-	-
FC00H	MECON	FSCMD	FSDAT	LOCK	PADRD	PTSL	PTSH	-
FC08H	-	-	-	-	-	-	-	-

7.3 Flash

7.3.1 Function Introduction

The Flash memory contains a 16K byte Flash main data area, and the Flash memory can be repeatedly erased and written. Flash memory is controlled by a set of specific registers. Users can use these registers to perform operations such as reading, writing, erasing, setting write protection, and so on.

7.3.2 Flash Memory Organization

- Flash is composed of several sectors which are the smallest units for erasure. Each sector is 64 bytes.
- Flash write operations are performed in units of pages, and 64 bytes must be written at one time. Single-byte writes are not supported.
- 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. The function is similar to EEPROM.

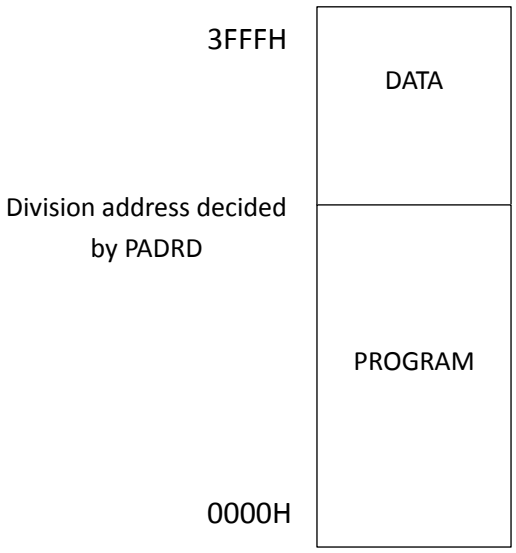


Figure 7-3-2-1 16K Flash Memory Structure

7.3.3 Flash Register Description

Table 7-3-3-1 Register MECON

FC00H	7	6	5	4	3	2	1	0
MECON	-	DPSTB	-	-	-	-	-	-
R/W	-	R/W	-	-	-	-	-	-
Initial Value	-	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. When the chip exits IDLE/STOP mode, Flash exits SLEEP mode as well.*
5~0	-	-

Table 7-3-3-2 Register FSCMD

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

Bit number	Bit symbol	Description
7	IFEN	Information area access enable bit, this bit needs to be set when accessing
6~4	-	-
3	CLRPL	Clear the data in the Flash latch
2~0	CMD	Command register 000：No operations 100：Flash sector erase 001：Read Flash DATA area 010：Write Flash DATA area 011：Erase one sector of Flash data area 101：Read Flash PROGRAM area 110：Write Flash PROGRAM area 111：Erase one sector of 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
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	1	0	0	0	0	0	0	0
Bit number	Bit symbol	Description						
7	-	-						

6~0	PADRD	PROGRAM and DATA area division configuration register The unit for division is 128 bytes and when PADRD>0: The address space for PROGRAM area : 0 ~ (PADRD×128 - 1), The address space for DATA area: (PADRD×128) ~ 3FFFH. *Note：* *1.PADRD=0 indicates the whole Flash is DATA area* *2.The maximum value of PADRD is 80H, 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[13:8]					
R/W	-	-	R/W					
Initial Value	-	-	0	0	0	0	0	0

Bit number	Bit symbol	Description
15~14	-	-
13~0	PTS	Target address pointer register, when writing FSDAT operation, data will be written into PTS[5:0] related flash latch for temporary storage, PTS[5:0] corresponds to the lower 6 bits of the actual write operation; when sending a write command, Need to set the relevant page address PTS[13:6]. It is best to reconfigure the PTS address for each read, write, and erase operation. For continuous read operations, only the first address operation of the continuous read operation can be set.

7.3.4 Flash Control Example

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

For instance, if the user wants to divide a 16K Flash (128 byte DATA area and the remains for PROGRAM area), the program may like this:

```
-------------------------------------------------------------------------------
PADRD = 127; //The address of the program area is: 0~0x3F7F, and the address of the data area is: 0x3F80~0x3FFF
-------------------------------------------------------------------------------
```

Note: This makes the physical address of the DATA area in FLASH 0x3F80~0x3FFF while the logical address is 0x0000~0x007F. The logical address is used for DATA area's R/W

◆ **Sector erasure of DATA area**

For instance, to erase page n of the data space, the procedure is as follows:

```
-------------------------------------------------------------------------------
FSCMD = 0; //Set CMD = 0
LOCK = 0x2A; //Unlock DATA area
FSCMD = 8;    //Set erase latch
PTSH = (unsigned char)((n*0x40)>>8); //Set sector high address
PTSL = (unsigned char)(n*0x40);       //Set sector low address
FSCMD = 3;    //Set data area erase command
LOCK = 0xAA; //Lock FLASH
-------------------------------------------------------------------------------
```

Note: Page number n=0、1、2……。

◆ **Write data into DATA area**

For instance, write data 0xAA to DATA area of which address is n~(n+63), the program will be:

```
-------------------------------------------------------------------------------
unsigned char i;
FSCMD = 0; //Set CMD = 0
LOCK = 0x2A; //Unlock DATA area
PTSH = 0;      //Set the start address of the page latch
PTSL = 0;      //Set the start address of the page latch
FSCMD = 8;    //Set erase latch
for(i=0;i<64;i++)
{
    FSDAT = 0xAA; //Write 1 page of data continuously
}
```

```
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
LOCK = 0xAA; //Lock FLASH
```

Note:

1. *When writing data continuously, you only need to set the first address. After each FSDAT is written, the data pointer register PTS will automatically accumulate.*
2. *When writing the data area, the address set is the logical address of the data area, not the physical address of FLASH. The logical address starts from 0.*
3. *Data can only be written in page units, and 64 bytes must be written each time.*

◆ **Read data from DATA area**

For instance, the pointer dataBuf reads data from DATA area of which address is n~(n + 63), the program will be:

```
unsigned int i,dataBuf[64];
FSCMD =       0;
LOCK = CMD_DATA_AREA_UNLOCK;          //Unlock DATA area
PTSH = (unsigned char)(Address>>8);   //Fill in the high address
PTSL = (unsigned char)Address;        //Fill in the low address
FSCMD = CMD_DATA_AREA_READ;           //Perform a read operation
for(i = 0; i < Length; i++)
{
    dataBuf[i]= FSDAT;
}
FSCMD = 0;
LOCK = CMD_FLASH_LOCK;                //Lock FLASH
```

Note:

1. *When data is read continuously, only original address has be set. PTS will increase automatically after writing FSDAT each time.*
2. *Data reading does not need to be in units of sector , and any number of bytes can be read continuously.*

◆ **Sector erasure of PROGRAM area**

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

```
FSCMD = 0; //Set CMD = 0
```

```
LOCK = 0x29; //Unlock the PROGRAM area
FSCMD = 8;    //Set erase latch
PTSH = (unsigned char)((n*0x40)>>8); //Set sector high address
PTSL = (unsigned char)(n*0x40);       //Set sector low address
FSCMD = 7;    //Set erase command
LOCK = 0xAA; //Lock FLASH
```

Note: Sector number n=0、1、2.....

◆ **Write data into PROGRAM area**

For instance, Write data 0xAA to the program space address n~(n+63), the program is as follows:

```
unsigned char i;
FSCMD = 0; //Set CMD = 0
LOCK = 0x29; //Unlock the PROGRAM area
FSCMD = 8;    //Set erase latch
PTSH = 0;      //Set the start address of the page latch
PTSL = 0;      //Set the start address of the page latch
for(i=0;i<64;i++)
{
    FSDAT = 0xAA; //Write 1page data continuously
}
PTSH = (unsigned char)(n>>8);     //Set the higher 8 bits of the first address of the data
PTSL = (unsigned char)n;          //Set the lower 8 bits of the first address of the data
FSCMD = 6;    //Set write command
LOCK = 0xAA; //Lock FLASH
```

Note:

1. *When data is read continuously, only original address has be set. PTS will increase automatically after writing FSDAT each time*
2. *Data can only be written in sector units, and 64 bytes must be written each time.*

◆ **Read data from PROGRAM area**

For instance, the pointer dataBuf reads data from PROGRAM area of which address is n~(n+63), the program will be:

```
unsigned char i, dataBuf[64];
FSCMD = 0; //Set CMD = 0
```

```
LOCK = 0x29; //Unlock the PROGRAM 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 = 5;    //Set READ command
for(i=0;i<64;i++)
{
      dataBuf[i] = FSDAT ; //Write data continuously
}
FSCMD   =     0;
LOCK = 0xAA; //Lock FLASH
```

Note:

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

2. Data reading does not need to be in units of sector , and any number of bytes can be read continuously.

8 Interrupt System

8.1 Function Introduction

CA51F1 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

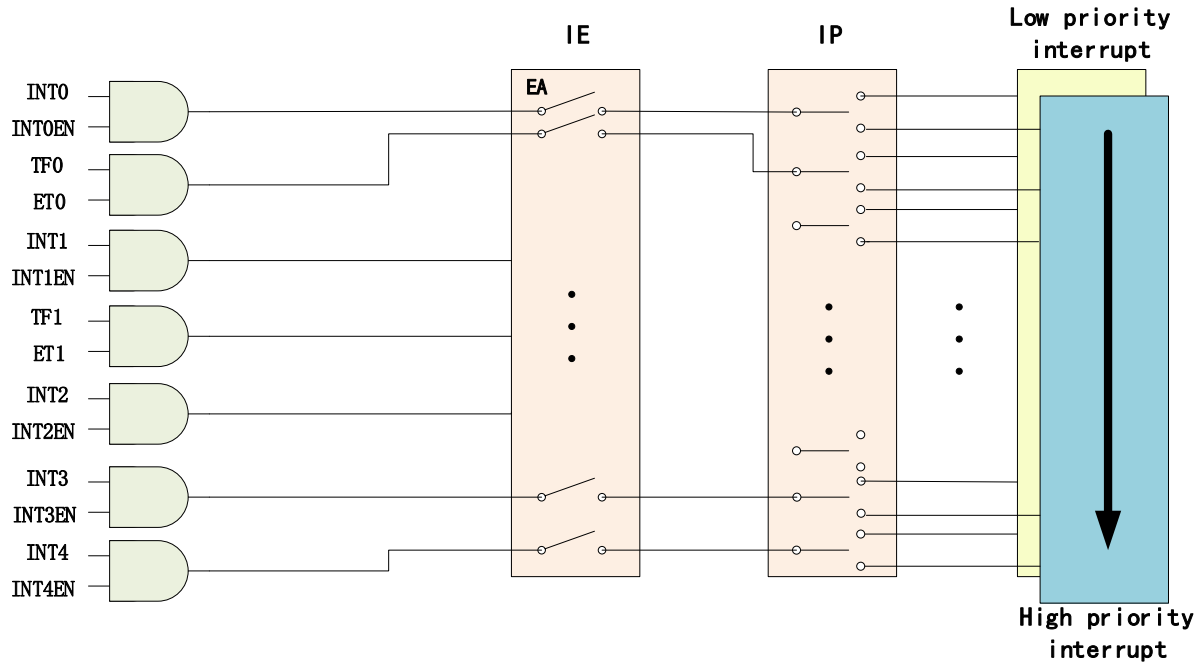


Figure 8-2-1 Interrupt Logic

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	UART	23H	4
INT3	Touch button interrupt/TMC Interrupt	2BH	5
INT4	WDT Interrupt /I^{2}C 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 WDT/I^{2}C/LVD/external interrupt 4) 0：disable 1：enable
5	INT3EN	Interrupt 3 enable control (Interrupt 3 is used for TMC/TK) 0：disable 1：enable
4	INT2EN	Interrupt 2 enable control (Interrupt 2 is used for UART) 0：disable 1：enable
3	ET1	Timer 1 Interrupt enable control 0：disable 1：enable
2	EX1	Interrupt 1 enable control (interrupt 1 is used for external interrupt 1)

		0：disable 1：enable
1	ET0	Timer 0 interrupt enable control 0：disable 1：enable
0	EX0	Interrupt 0 enable control (interrupt 0 is used for external interrupt 0) 0：disable 1：enable

Note：The enable control of IE corresponds to interrupt vector which means the enable control for each interrupt source has to be set as well.

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

The interrupt pins of INT0/INT1 are P3.2/P3.4 respectively, and their functions are basically compatible with the standard 8051. INT0 and INT1 can be triggered by rising or falling edge. The selection bits are IT0 and IT1 respectively. For details, please refer to the related description of register TCON. INT0 and INT1 can be used to wake up from STOP mode.

8.5.2 External Interrupt Control Example

◆ **External Interrupt 0/1 Control Example**

For instance, enable external interrupt 0, the program will be:

```
---------------------------------------------------------------------------------------
void INT0_init(void)
{
    P32F = 1;       //The interrupt pin of external interrupt 0 is P32, set P32 as input function
    EX0 = 1;        //INT0 interrupt enable
    IE0 = 1;        //External interrupt 0 enable
    IT0 = 1;        //Set as falling edge interrupt
    PX0 = 1;        //Set INT0 as high priority
    EA = 1;         //Enable Global Interrupt
    }
void INT0_ISR (void) interrupt 0
{
    // External Interrupt0 interrupt service routine
}
---------------------------------------------------------------------------------------
```

For instance, enable external interrupt 1, the program will be:

```
---------------------------------------------------------------------------------------
void INT1_init(void)
{
    P34F = 1;       //The interrupt pin of external interrupt 1 is P34, set P34 as input function
    EX1 = 1;        //INT1 interrupt enable
    IE1 = 1;        //External interrupt 1 enable
    IT1 = 1;        //Set as falling edge interrupt
    PX1 = 1;        //Set INT1 as high priority
    EA = 1;         //Enable Global interrupt
}
void INT1_ISR (void) interrupt 2
{
    //External Interrupt1 interrupt service routine
}

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

9 Clock System

9.1 Clock System Introduction

CA51F1 Series has several clock sources as follows：

- 16 MHz Internal RC Oscillator
- 100 KHz Internal RC Oscillator

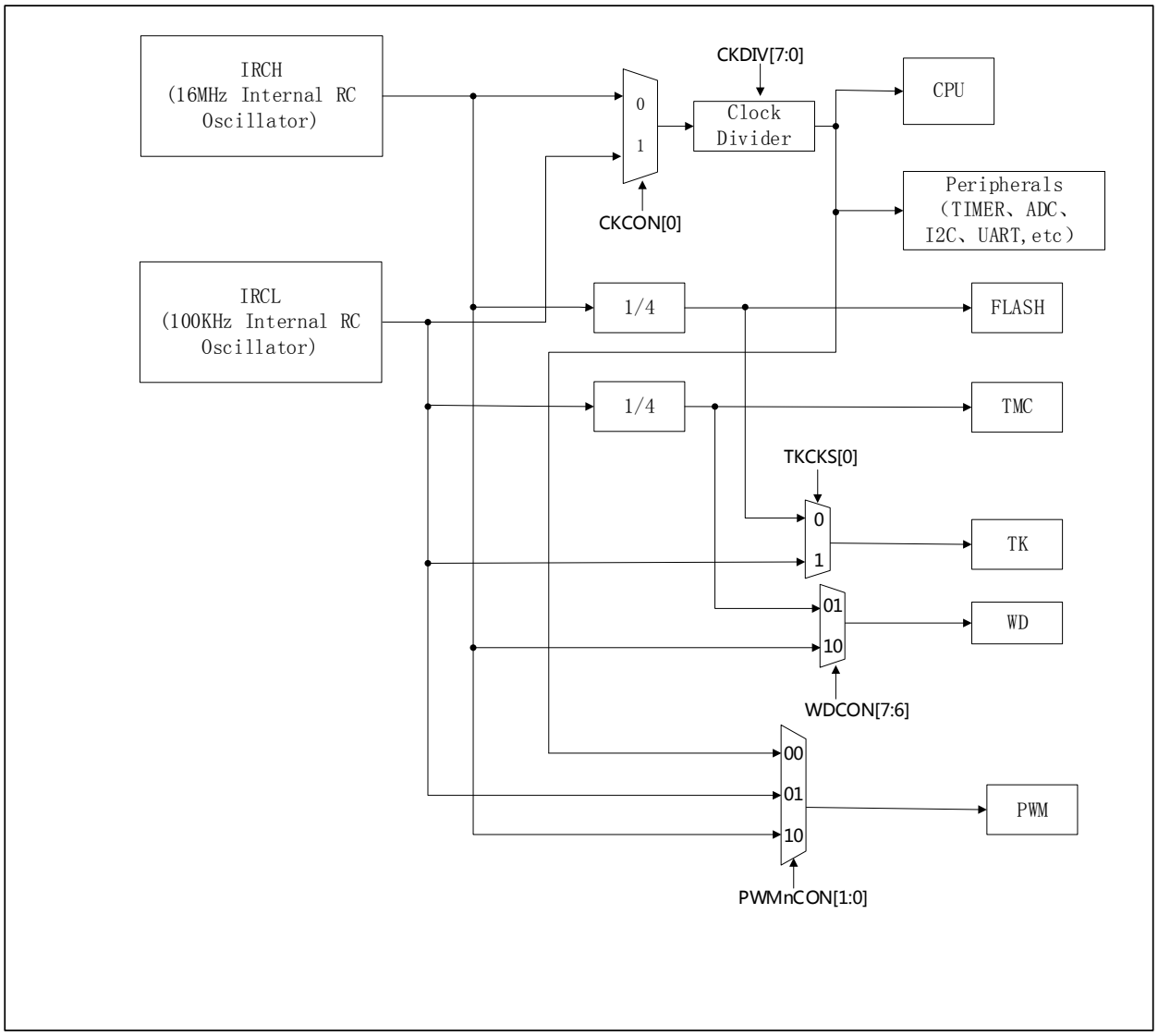


Figure 9-1-1 Clock Structure Diagram

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.

9.1.1 Clock Special Name Definition

Name abbreviation	Description
IRCH	16 MHz Internal RC Oscillator
IRCL	100 kHz Internal RC Oscillator

9.1.2 16 MHz Internal RC Oscillator（IRCH）

IRCH is the default system clock after the chip is powered on. It can be turned on or off by the IHCKE bit in the CKCON register. After the chip leaves the factory, the frequency of IRCH is corrected to 16MHz@3.3V/25℃, and the clock accuracy is ±1%.

9.1.3 100KHz Internal RC Oscillator（IRCL）

IRCL can be turned on or off by the ILCKE bit in the CKCON register. IRCL is set as the system clock to achieve low power consumption of the system. The clock accuracy of IRCL is ±25%@3.3V/25℃.

9.1.4 Clock Control Register Description

Table 9-2-2-6 Register IHCFG

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

Bit number	Bit Symbol	Description
7~0	IHCFG	IRCH frequency adjustment register

9.2 System Clock

System clock is controlled by register CKCON, CKSEL and CKDIV. Users can disable/enable any of these clock sources, divide the frequency, change the system clock and so on by using these registers.

System clock has two clock options: IRCH and IRCL. After power-on, the default system clock is IRCH, and the CKDIV value is 1, that is, the system clock is power-on by default to IRCH's two-divided frequency. If the CPU needs to run at a higher frequency Frequency, software can set CKDIV to 0.

9.2.1 System Clock Architecture

System clock structure diagram is shown in Figure 9-2-1.

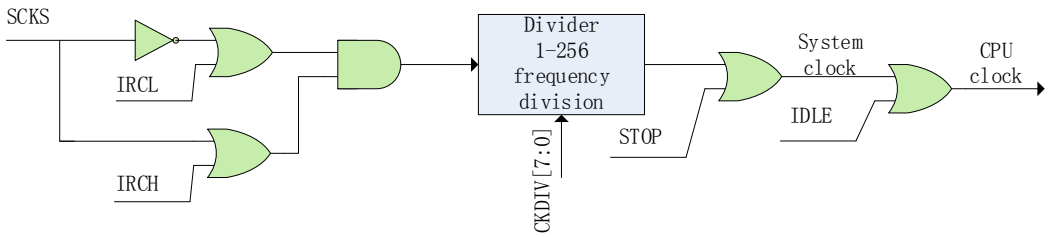


Figure 9-2-1 System Clock Architecture

9.2.2 System Clock Control Register Description

Table 9-3-1 Register CKCON

C8H	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

Bit number	Bit Symbol	Description
7	IHCKE	IRCH enable control 1：Open 0：Close *Note：* *When this bit is 1, the clock module is turned on, but when this bit is 0, if the system or other modules select the clock source, the clock will still be turned on.*
6	ILCKE	IRCH enable control 1：Open 0：Close *Note：* *When this bit is 1, the clock module is turned on, but when this bit is 0, if the system or other modules select the clock source, the clock will still be turned on.*
-	-	-
1-0	SCKS	System clock selection 0：IRCH 1：IRCL

Table 9-3-2 Register CKDIV

C9H	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	1
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
    CKCON = (CKCON&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, wait for IRCL stabilization
    CKCON = (CKCON&0xFE) | CKSEL_IRCL;       //Set IRCL as system clock}
--------------------------------------------------------------------------------
```

10 Power Supply and Reset System

10.1 Power Supply

Connect a 2.2V-5.5V power supply between the VDD and VSS pins of the CA51F1 series chip. This power supply can directly supply power to the internal digital and analog systems of the chip. It should be noted that under different power supply voltage conditions, the maximum operating frequency and power consumption of the chip are not the same. For details, please refer to the electrical characteristics chapter.

The BANDGAP reference voltage is also designed inside the chip as the reference voltage source for ADC internal reference voltage, LVD voltage, touch internal op amp, etc. After the reference voltage source is calibrated at the factory, the accuracy is ±30mV.

10.1.1 Internal reference voltage control register

Table 10-1-2-1 Register PWCON

86H	7	6	5	4	3	2	1	0
PWCON	FLEVEL[3:0]				**VREFS**	-	-	-
R/W	R/W				**R/W**	-	-	-
Initial Value	0	1	1	1	**0**	-	-	-

Bit number	Bit Symbol	Description
7~4	FLEVEL	Internal reference voltage(Bandgap) output adjustment 0000：0.825V 0001：0.850V 0010：0.875V 0011：0.900V 0100：0.925V 0101：0.950V 0110：0.975V 0111：1.000V 1000：1.025V 1001：1.050V 1010：1.075V 1011：1.100V 1100：1.125V

		1101：1.150V 1110：1.175V 1111：1.200V *Note: The accuracy of this reference voltage after factory calibration is ±30mV. The calibration value is automatically loaded after power-on, and the user is not allowed to change it.*
3	VREFS	Reference voltage drive selection： 0：0.8uA drive capacity, default 1：Op-amp driver
2-0	-	-

10.2 Reset System

There are multiple internal and external reset sources for CA51F1 Series chip as figure 10-2-1 shows.

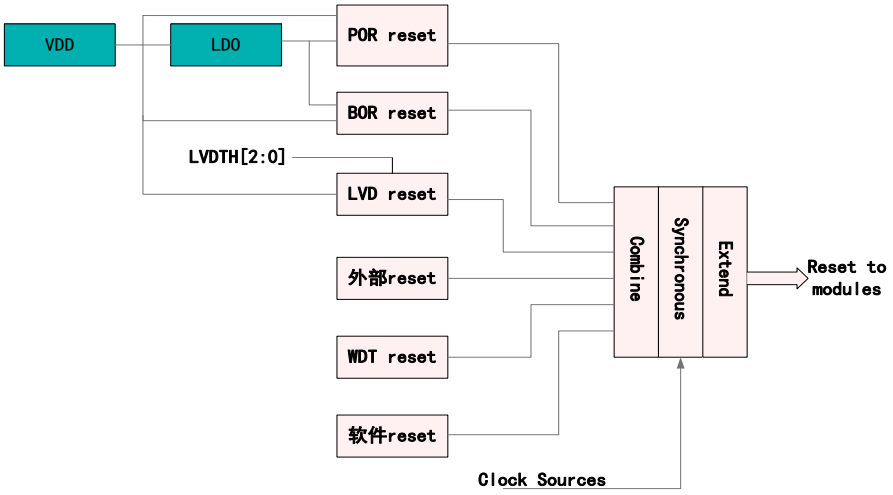
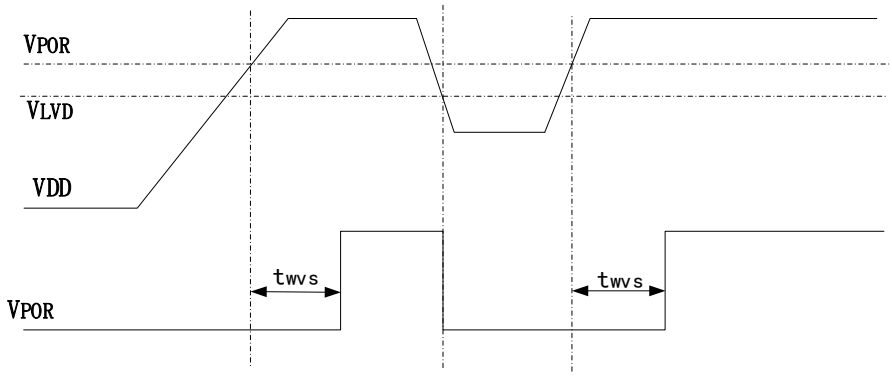


Figure 10-2-1 Reset System Architecture

- **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 untial 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 power-on reset, 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) PC will point to address 0 after any reset.

11 Power Consumption Management

There are 3 low power consumption modes for CA51F2 Series : IDLE, STOP and Low Speed mode. The system power consumption for IDLE, STOP and Low Speed mode is less than 20uA, 5uA and 40uA respectively.

11.1 IDLE Mode

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

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

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

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

11.2 STOP Mode

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

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

The STOP mode can be awoken by external interrupt, LVD reset or interrupt, hard reset, RTC interrupt, WDT interrupt or reset, clock monitor interrupt and touch key interrupt. If it is awoken by an interrupt, the chip will

resume clock first and respond to the interrupt, and then enters corresponding the interrupt service routine. After the chip exits the interrupt service routine, it will executes the instructions after the setting STOP to 1 instruction.The STOP will be cleared automatically when the chip exits STOP mode.

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

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

11.3 Low Speed Mode

Since the power consumption of the chip is directly related to the operating speed, switching the main clock to low-speed clock operation can also significantly reduce power consumption. When the system is set to IRCL (the frequency is 100KHz), the current is less than 40uA.

11.4 Related Register Description

Table 11-4-1 Register PCON

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

Bit number	Bit Symbol	Description
7~6	-	-
5	SWRST	Soft reset control, 1 enables Set SWRST=1 to generate a soft reset, and automatically clear to 0 after reset is generated
4~3	-	-
2	TSMODE	Online emulation mode flag, 1 means that the chip is working in online emulation mode
1	STOP	STOP mode control bit,1 enables 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 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 I^2C/WDT/LVD/External Interrupt 4 in IDLE mode
5	TKINT/TMINT/EPIF[1]	Interrupt status of Touch Key/TMC/External Interrupt 3 in IDLE mode
4	UART/EPIF[0]/ADC	Interrupt status of UART2/External Interrupt 2 in IDLE mode/ADC
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 11-4-2 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 WDT/LVD/I^2C in STOP mode
5	TKWKF/TMWKF	Interrupt status of Touch Key/TMC in STOP mode
4	EPWKF[2]	Interrupt status of interrupt 4 in STOP mode
3	EPWKF[1]	Interrupt status of interrupt 3 in STOP mode
2	EPWKF[0]	Interrupt status of interrupt 2 in STOP mode
1	PWKF[1]	Interrupt status of interrupt 1 in STOP mode
0	PWKF[0]	Interrupt status of interrupt 0 in STOP mode

11.5 Low Power Consumption Control Example

◆ **STOP Mode Example**

STOP mode program is as follows:

```
---------------------------------------------------------------------------------------
void Stop(void)
{
    I2CCON = 0;   //Disable I2C for it is the default enabled, otherwise the IRCH cannot be disabled
    CKCON = 0;    //Disable all the clocks
    MECON |= (1<<6);  //Set FLASH in deep sleep mode
    PCON |= 0x02;     //Enters STOP mode
    _nop_();
    _nop_();
    _nop_();
}
---------------------------------------------------------------------------------------
```

◆ **IDLE Mode Example**

IDLE mode program is as follows:

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

#define CKSEL_IRCH    0
#define CKSEL_IRCL    1

void Idle(void)
{
    CKCON |= ILCKE;                         //Enable IRCL clock
    Delay_ms(1);                            //Delay 1ms, wait for IRCL clock to stabilize
    CKCON = (CKCON&0xFE) | CKSEL_IRCL; //System clock switch to IRCL
    I2CCON = 0;                   //Disable I2C for it is the default enabled, otherwise the IRCH cannot be disabled
    CKCON&=~IHCKE;                          //Disable IRCH clock
    MECON |= (1<<6);                        //Set FLASH in deep sleep mode
    while(IDLST&0x7F);                  //If an interrupt is not responded, wait for the interrupt to be responded
    PCON |= 0x01;                           // Enters IDLE mode
    _nop_();
    _nop_();
}
```

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**

Low Speed mode procedure is as follows:

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

#define CKSEL_IRCH   0
#define CKSEL_IRCL   1

void LowSpeedMode(void)
{
    I2CCON = 0;             //Disable I2C for it is the default enabled, otherwise the IRCH cannot be disabled
    CKCON |= ILCKE;                                     //Enable IRCL clock
    Delay_ms(1);                                            //Delay 1ms, wait for IRCL clock to stabilize
    CKCON = (CKCON&0xFE) | CKSEL_IRCL;                  //System clock switch to IRCL
    CKCON &= ~IHCKE;                                    //Disable IRCH clock
}
---------------------------------------------------------------------------------------------
```

12 General 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 by TR0 (TCON[4]). When GATE0=1, the timer/counter's is enabled/disabled by INT0. INT0 signal with high level with enable the counting and vice verse.

- **Mode1**

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

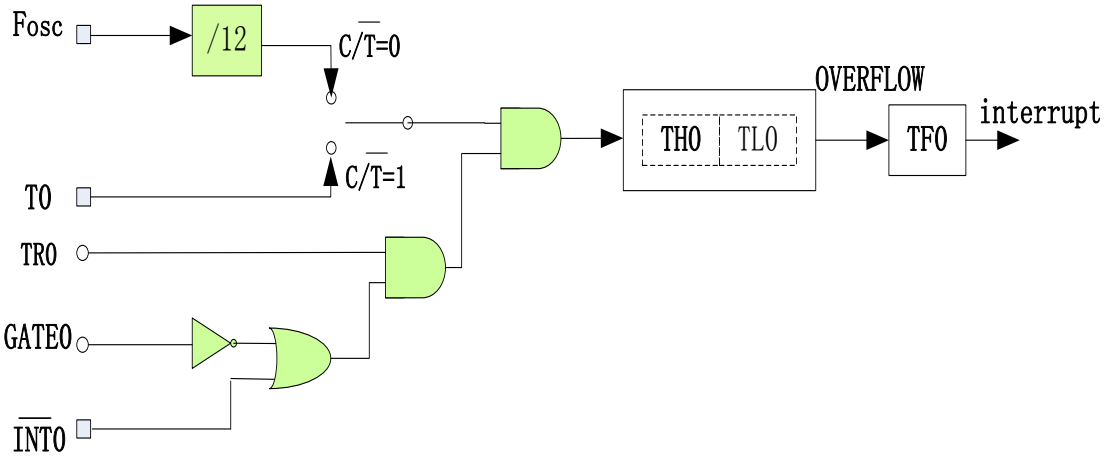


Figure 12-1-1-1 Timer0 Mode0/1

- **Mode2**

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

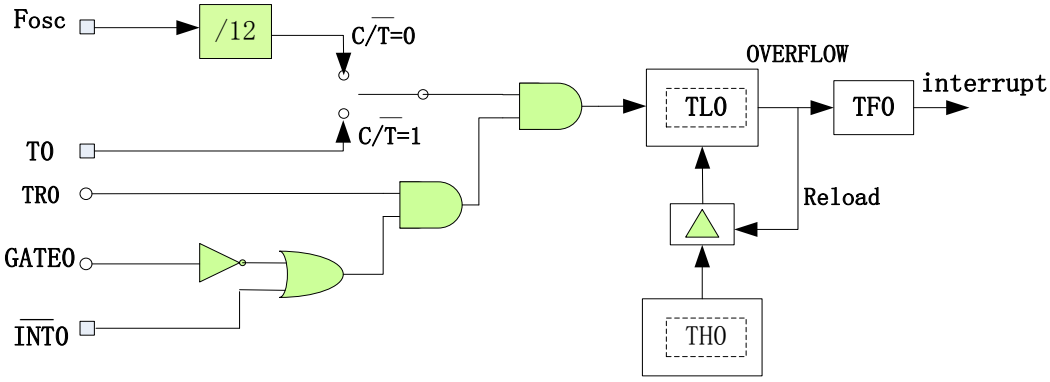


Figure 12-1-1-2 Timer0 Mode2

- **Mode3**

TL0 and TH0 are two independent 8 bit counter/timer in this mode. TL0 can be used as timer or counter while TH0 can only be used as 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).

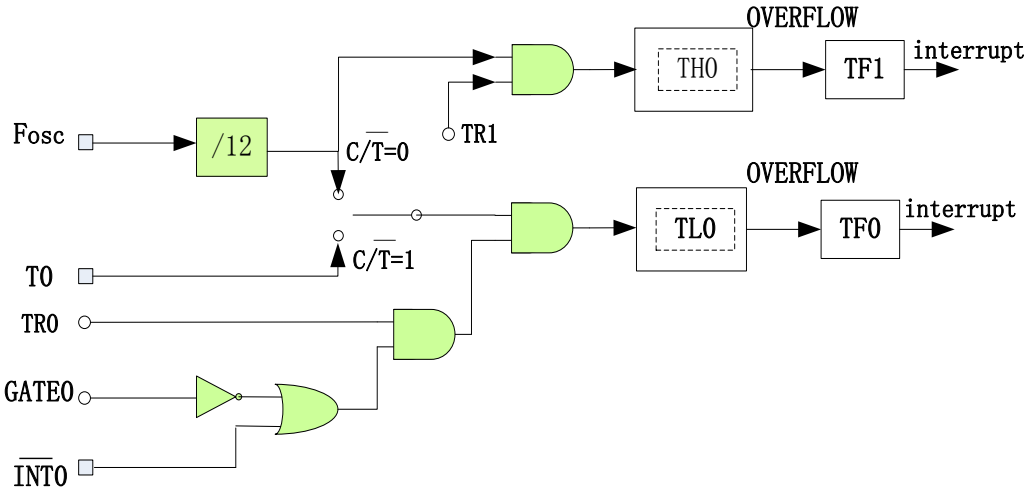


Figure 12-1-1-3 Timer0 Mode3

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 comes to 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 comes to 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	Timer0 Counter/Timer selection 0：Timer, the clock for it is the system clock with its frequency divided by 12 1：Counter, the clock for it is T0 input clock
1	T0M1	[T0M1,T0M0] Timer0 mode selection 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**

In this mode, Timer1 acts as a 13-bit timer/counter, TH1 stores the upper 8 bits of the 13-bit timer/counter, TL1[4:0] stores the lower 5 bits, and TL1[7:5] is invalid , Should be ignored when reading. When Timer1 overflows, the interrupt flag bit TF1 (TCON[7]) will be set. After the interrupt is serviced, the TF1 bit will be automatically cleared to 0. When GATE1 (TCON[7])=0, the timer/counter is enabled to count by the TR1 (TCON[6]) bit. When GATE1=1, the timer/counter is enabled by pin INT1, and INT1 is high Count when the level is high, and stop counting when INT1 is low.

- **Mode1**

Timer1 acts as a 16-bit timer/counter, TH1 stores the upper 8 bits of the 16-bit timer/counter, and TL1 stores the lower 8 bits. When Timer1 overflows, the interrupt flag bit TF1 (TCON[7]) will be set. After the interrupt is serviced, the TF1 bit will be automatically cleared to 0. When GATE1 (TCON[7])=0, the timer/counter is enabled to count by the TR1 (TCON[6]) bit. When GATE1=1, the timer/counter is enabled by pin INT1, and INT1 is high Count when the level is high, and stop counting when INT1 is low.

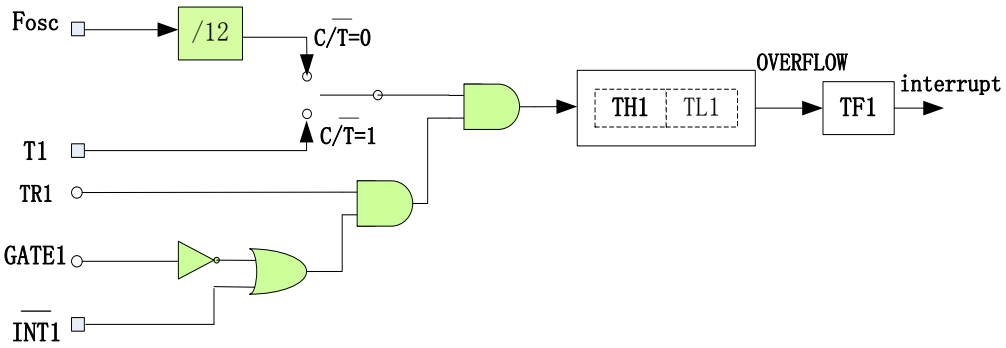


Figure 12-2-1 Timer1 Modes0/1

- **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.

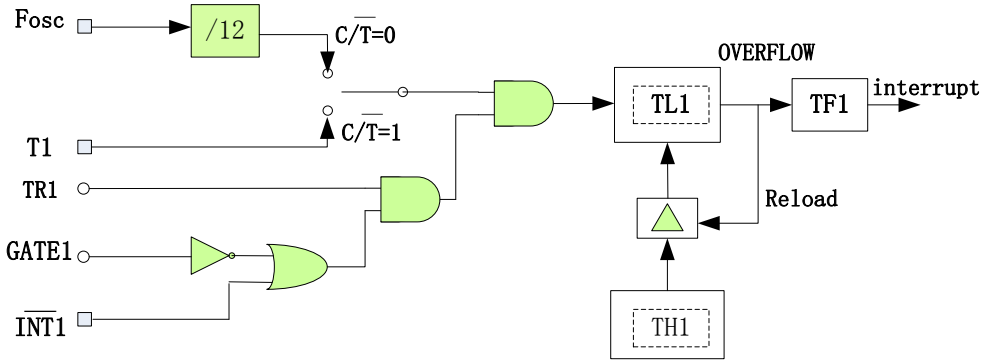


Figure 12-2-2 Timer1 Modes2

- **Mode3**

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

12.2.2 Timer1 Register Description

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

Table 12-2-2-1 Register TL1

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

Table 12-2-2-1 Register TH1

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

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

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.

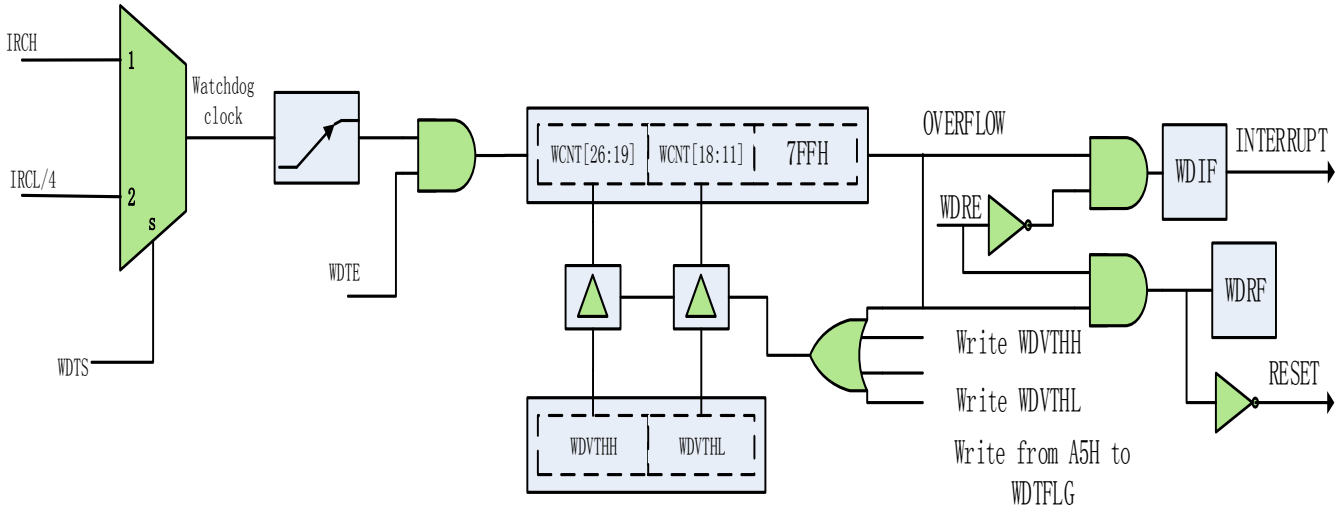


Figure 13-1-1 Watchdog Module Architecture

13.2 Watchdog Timer(WDT) Register Description

Table 13-2-1 Register WDCON

A0H	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	1	1	1	1	1	1	1	1
A3H	7	6	5	4	3	2	1	0
WDVTHH	WDVTH[15:8]							
R/W	R/W							
Initial Value	1	1	1	1	1	1	1	1

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

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 globe interrupt
}
void WDT_ISR (void) interrupt 6
{
    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 of the TMC timer is IRCL, the minimum unit of interrupt is 512 IRCL clock cycles, and the configurable interrupt time is 1~256 minimum unit time. In STOP/IDLE mode, the TMC interrupt can wake up the CPU.

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	Enabled TME, 1 enable
6~1	-	-
0	TMF	TMC interrupt flag, 1 enable, write 1 to clear 0

Table 14-2-2 Register TMSNU

D6H	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 interrupt time of TMC is (TMSNU+1) ×512× Tircl *Note: Tircl is one cycle time of the IRCL clock.*

14.3 TMC Control Example

Set TMC as the minimum unit time interrupt (that is, 512 IRCL clock cycles), the procedure is as follows:

```
---------------------------------------------------------------------------------------
#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)             //Determine the TMC interrupt flag
    {
        TMCON |= TMF;           //Clear the TMC interrupt flag

    }
}

void TMC_init(void)
{
    CKCON |= ILCKE;    //Enable the IRCL clock
    TMCON = TME(1);    //Enable TMC
    TMSNU = 0;         //Set 1 minimum unit time (that is, 512 IRCL clock cycles) to generate an interrupt
    INT3EN =1;         //Enable TMC interrupt
    EA = 1;            //Enable globe interrupt
}
---------------------------------------------------------------------------------------
```

15 General Purpose Input/Output(GPIO)

15.1 Function Introduction

CA51F1 series chip has 6 I/O pins in the largest package, and each pin is a multiplex function pin, which can not only be independently programmed as an input/output port, but also can be set as other function pins. Each pin is assigned a function setting register PnxF (corresponding to pin Pnx respectively, where n=0, 3, representing P0, P3, x=0~7, representing Pn.0~Pn.7), the user can pass Register PnxF configures the main function of the pin and other options. For details, see the introduction to the register section.

Main features of GPIO：

- High impedance mode configurable
- The pull-up or pull-down can be set independently for the I/O structure
- The output mode can be open-drain output or push-pull output
- Data output latching supports read-modify-write
- Supports 2.2~5.5V wide voltage

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

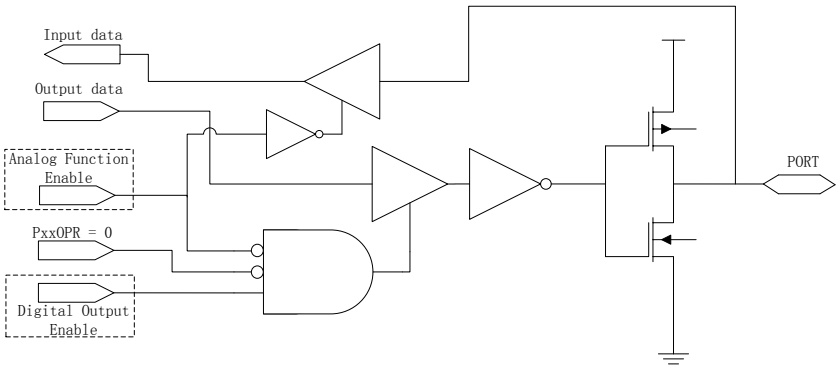


Figure 15-1-1 I/O Push-pull Mode Structure

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

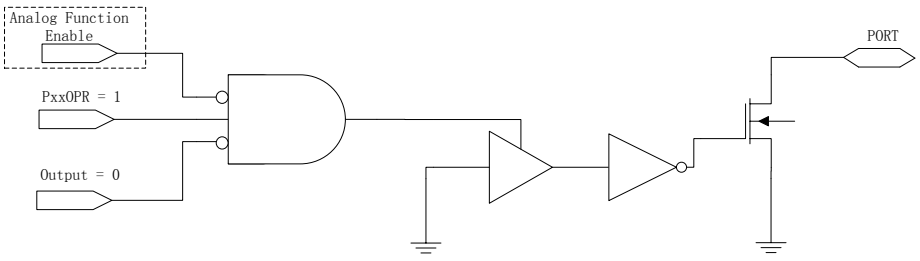


Figure 15-1-2 I/O Open-drain Mode Structure

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

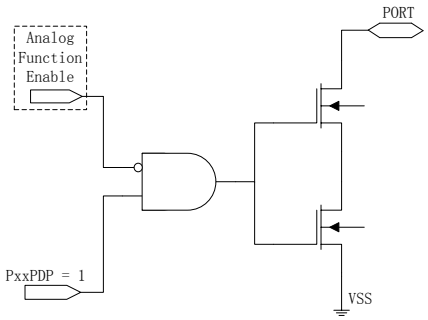


Figure 15-1-3 I/O Pull-down Mode Structure

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

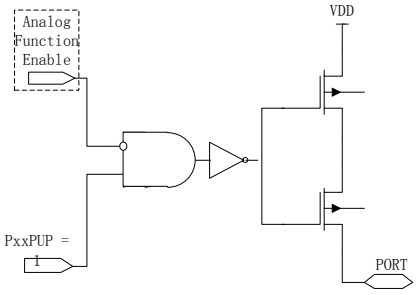


Figure 15-1-4 I/O Pull-up Mode Structure

15.2 Pin Register Description

Table 15-2-1 Register P0

80H	7	6	5	4	3	2	1	0
P0	-	-	-	-	-	-	P01	-
R/W	-	-	-	-	-	-	R/W	-
Initial value	-	-	-	-	-	-	0	-

Bit number	Bit Symbol	Description
7~2	-	-
1	P01	Data register for pin P01，valid when the pin function is set to GPIO 0：P01 is low level when the pin is set to input; when the pin set to output,P01 outputs low level signal 1：P01 is high level when the pin is set to input; when the pin set to output,P01 outputs high level signal

0	-	-

Table 15-2-2 Register P3

B0H	7	6	5	4	3	2	1	0
P3	-	-	P35	P34	-	P32	P31	P30
R/W	-	-	R/W	R/W	-	R/W	R/W	R/W
Initial value	-	-	0	0	-	0	0	0

Bit number	Bit Symbol	Description
7~6	-	-
5~0	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

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
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	1	1
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	1	1
800DH	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
800EH	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
800FH	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	1	1

Bit number	Bit Symbol	Description
7	PnxPUP	Pull-up resistor enable control 0: disable pull-up resistor 1: enable pull-up resistor *Note: Pull-up resistance is 30K*
6	PnxPDP	Pull-down resistor enable control 0: disable pull-down resistor 1: enable pull-down resistor *Note: Pull-down resistance is 30K*
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

Table 15-2-5 Register PnxC

8121H	7	6	5	4	3	2	1	0
P01C	-	SMIT_EN	-	-	-	-	DRV	SR
R/W	-	R/W	-	-	-	-	R/W	R/W
Initial value	-	1	-	-	-	-	1	1
8138H	7	6	5	4	3	2	1	0
P30C	-	SMIT_EN	-	-	-	-	DRV	SR
R/W	-	R/W	-	-	-	-	R/W	R/W
Initial value	-	1	-	-	-	-	0	0
8139H	7	6	5	4	3	2	1	0
P31C	-	SMIT_EN	-	-	-	-	DRV	SR
R/W	-	R/W	-	-	-	-	R/W	R/W
Initial value	-	1	-	-	-	-	0	0
8126H	7	6	5	4	3	2	1	0
P32C	-	SMIT_EN	-	-	-	-	DRV	SR
R/W	-	R/W	-	-	-	-	R/W	R/W
Initial value	-	1	-	-	-	-	0	0
8127H	7	6	5	4	3	2	1	0
P34C	-	SMIT_EN	-	-	-	-	DRV	SR
R/W	-	R/W	-	-	-	-	R/W	R/W
Initial value	-	1	-	-	-	-	0	0
8138H	7	6	5	4	3	2	1	0
P35C	-	SMIT_EN	-	-	-	-	DRV	SR
R/W	-	R/W	-	-	-	-	R/W	R/W
Initial value	-	1	-	-	-	-	0	0

Bit number	Bit Symbol	Description
7	-	-
6	SMIT_EN	SMIT enable for 1 input, inverter enable for 0 input
5~2	-	-
1	DRV	Output intensity selection 0: Weak drive 1: Strong drive
0	SR	Output slope control 0: The slowest slope control 1: The fastest slope control

Note：Pnx → n=0、3, stands for P0、P3
x=0~7, stands for Pn.0~Pn.7

Table 15-2-6 Pin Alternate Function Mapping

Take value / Name	0	1	2	3	4	5	6	7
P01S	high impedance	digital intput	Digital output	ADC[4]	TK[3]	T1	PWM2	high impedance
P30S	high impedance	digital intput	digital output	ADC[5]	TK[4]	UART_TX	I2C_SDA	high impedance
P31S	high impedance	digital intput	digital output	ADC[0]	TK[0]	UART_RX	I2C_SCL	high impedance
P32S	high impedance	digital intput	Digital output	ADC[2]	TK[1]	high impedance	PWM0	RESET
P34S	high impedance	digital intput	Digital output	ADC[3]	TK[2]	high impedance	PWM1	high impedance
P35S	high impedance	digital intput	digital output	ADC[1]	TK_CAP	T0	high impedance	high impedance

15.3 Pin control Example

- **Set the Pin function**

  For instance, P01 is set to be push-pull output, the program is like：

```
P01F = 2;
```

  P01 is set to be open-drain output, the program is like：

```
P01F = (1<<5)|2;
```

  P01 is set to be open-drain output with pull-up enabled, the program is like：

```
P01F = (1<<7) | (1<<5) | 2;
```

  P01 is set to be input with pull-up enabled, the program is like

```
P01F = (1<<7) | 1;
```

16 Universal Serial Interface（UART）

16.1 Function Introduction

UART is a full-duplex asynchronous serial data transceiver, UART has a byte receive buffer. UART has two different working modes, as shown in Table 16-1-1-1.

SM	Mode	Description	Baud rate
0	A	9 bit asynchronous mode	CPUCLK/(32*(1024-SREL))
1	B	8 bit asynchronous mode	CPUCLK/(32*(1024-SREL))

Table 16-1-1-1 UART Communication Mode

UART has designed a special baud rate generator, and the baud rate is configured through registers SRELL and SRELH.

Figure 16-1-1-1 is a schematic diagram of the UART principle.

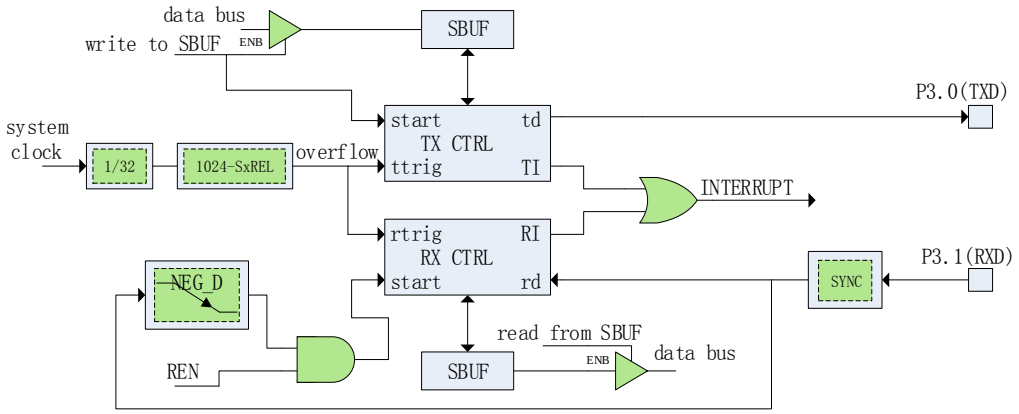


Figure 16-1-1-1 Schematic Diagram Of UART Working Principle

- **Mode A**

In mode A, the UART can simultaneously send and receive 9-bit data asynchronously. Writing data to the register SBUF will start the UART data transmission. The first bit transmitted is the start bit (0), then 9 bits of data (low bit is sent first), the 9th bit is the TB81 bit of the register SCON, and the last bit transmitted is the stop bit (1). In the receiving state, the UART synchronizes by detecting the falling edge of the pin RX. After the transfer process is completed, the lower 8 bits of data are stored in the register SBUF, and the 9th bit of data is stored in the RB8 bit.

- **Mode B**

The difference between mode B and mode A is that mode B is 8-bit data transmission, and the stop bit stores the effective stop bit. Other functions are the same as Mode A.

- **UART Multi-computer Communication**

In UART mode A, there is a mechanism specially suitable for multi-computer communication. When the SM21 bit of the register SCON is 1, only the slave that receives the 9th bit data as 1 (RB8=1) will generate a receive interrupt. This function can be used for multi-computer communication. The slave sets all their SM21 bits to 1. When the master transmits the address of the slave, the 9th bit is set to 1, so that all slaves will generate a receiving interrupt; the software of the slave compares their own address with the received address, and if they are consistent, they are addressed Set SM21=0 for the slave, and then set the 9th bit to 0 when the master continues to transmit the following data, because other slaves SM21 are still set to 1, so that only the addressed slave will generate a receive interrupt.

16.2Register Description

Table 16-2-1 Register SCON

98H	7	6	5	4	3	2	1	0
SCON	SM	UIE	SM2	REN	TB8	RB8	TI	RI
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	SM	UART mode selection, for more information please refer to Table 17-2-1-1
6	UIE	UART interrupt enable, 1 enables
5	SM2	Multi-computer communication enable control, 1 enables
4	REN	Serial receive enable control, 1 enables
3	TB8	The 9th data bit to transmit It will be transmitted as the 9th bit of the data in mode A and it is controlled by the software (For instance, parity check or multi-computer communication)
2	RB8	The 9th bit of the data received It will be used for UART to receive the 9th bit of the data in mode A It is the stop bit in mode B
1	TI	Transmit interrupt flag, 1 enables , cleared by writing 0 to it
0	RI	Receive interrupt flag, 1 enables , cleared by writing 0 to it

Table 16-2-2 Register SBUF

99H	7	6	5	4	3	2	1	0
SBUF	SBUF[7:0]							

R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

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

Table 16-2-3 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	Fast baud rate configuration enable control , 1 enable ***Note：*** *When UDE=0, the UART baud rate is in accordance with the original configuration, UDE=1, and the UART baud rate is configured by DNUM.* Fast baud rate configuration enable control, 1 enable
6~5	-	-
4~0	DNUM	Fast baud rate configuration register, only enable when UDE=1 When sending, it must meet DNUM>=0; when receiving, DNUM>=6 BR = F_{sys}/((DNUM+1)*(1024-SREL))

Table 16-2-4 Register SRELL、SRELH

8068H	7	6	5	4	3	2	1	0
SRELL	SREL[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
8069H	7	6	5	4	3	2	1	0
SRELH	-	-	-	-	-	-	SREL[9:8]	
R/W	-	-	-	-	-	-	R/W	
Initial Value	-	-	-	-	-	-	0	0

Bit number	Bit Symbol	Description
9~0	SREL	Baud rate configuration register UDE=0 时， BR = F_{sys}/(32 * (1024 － SREL)) UDE=1 时， BR = F_{sys}/((DNUM+1)*(1024-SREL))

17 I²C Interface

17.1 Function Introduction

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

17.2 I²C Main Features

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

17.3 I²C Function Description

I²C modules supports I²C standard bus specification. I²C bus includes 2 wires to transfer data among devices, one is SCL(Serial Clock) and the other is SDA(Serial Data), as Figure 17-3-1 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.

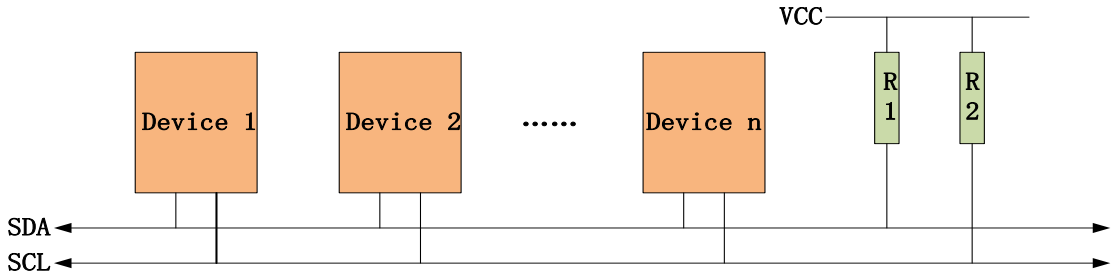


Figure 17-3-1 I²C Bus Interconnection

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

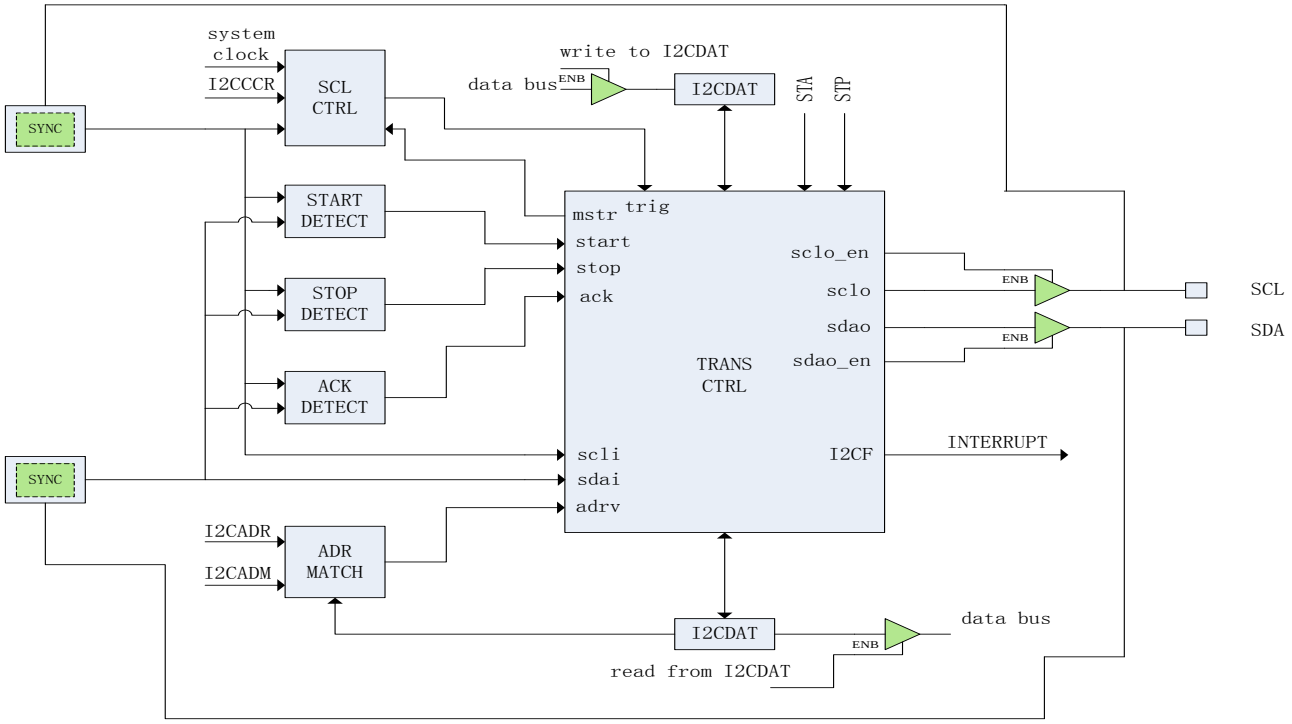


Figure 17-3-2 I²C Module Schematic

- **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.

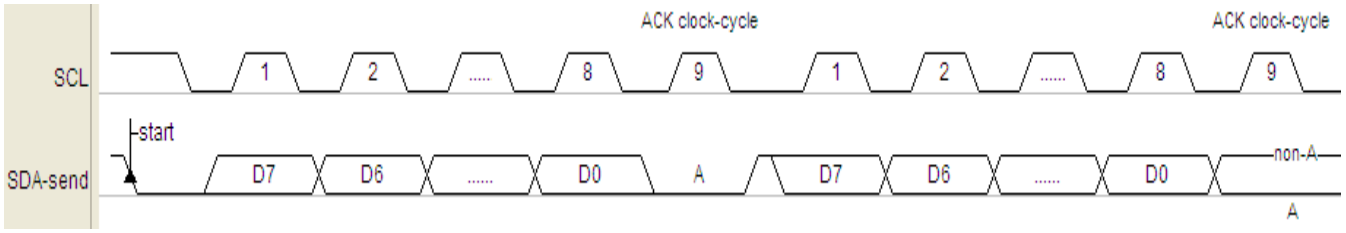


Figure 17-3-3 I²C Bus Data Transmission Format

- **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. Within the 9th clock cycle after 8 clocks of a byte transmission, the receiver must send back an answer bit to the transmitter. The answer bit is set via the AAK bit and the set answer bit must be set before a byte has been transmitted ,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.

The STOP signal generated by the master at the end of communication also generates the interrupt flag I2CSTP on the slave, indicating the completion of the communication process. When there occurs interrupt I2CF, if SHD=1, SCL will be set to low by slave. 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 I²C is simulated by software. Thus, the master's software must wait long enough so that the slave can deal with the response.

When I²C 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.

17.4 I²C Communication Pin Mapping

There are different mappings for I2C 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	1	0	0	0	0	1	0	0

Bit number	Bit Symbol	Description
7	I2CE	I² C module enable control, 1 enables it
6	I2CIE	I² C module enable control, 1 enables it

5	STA	I² C START signal transfer control,valid when it is 1, it will be cleared automatically when START signal
4	STP	I2C STOP signal transfer control, valid when it is 1, it will be cleared automatically when STOP signal
3	SHD	When it is 1, if I2CF=1, I2CF will make SCL remain low after SCL becomes low
2	AAK	I² C ACK signal transfer control, 1 enables it *Note：* *When I²C is configured as slave, this bit must be set to 1 beforehand, otherwise even the address matches it will not reply ACK, and thus cannot be addressed*
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
0	STFE	When STFE=1, I2CF will be set to 1 if I² C module detects the START signal

Table 17-5-2 Register I2CADR

C1H	7	6	5	4	3	2	1	0
I2CADR	GCE	I2CADRL[6:0]						
R/W	R/W	R/W						
Initial Value	1	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7	GCE	Broadcast address recognition(00H)enable control, 1 enables it
6~0	I2CADRL	I² C 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	I2CADML[6:0]						
R/W	R/W	R/W						
Initial Value	0	0	0	0	0	0	0	0

Bit numbe	Bit Symbol	Description
7	SPFE	When SPFE=1, I2CF will be set to 1 if I² C module detects the STOP signal
6~0	I2CADML	I² C 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 I2CCR

C3H	7	6	5	4	3	2	1	0
I2CCR	I2CCR[7:0]							
R/W	R/W							
Initial Value	0	0	1	0	0	0	0	0

Bit number	Bit Symbo	Bit Symbo
7~0	I2CCR	I^2 C clock setting register The sampling frequency is $2^{I2CCR[7:5]}$ divisions of the I^2C operating clock, when I2CCR[7:5] is equal to 000：$F_{sample}=F_{i2cclk}$ 001：$F_{sample}=F_{i2cclk}/2$ 010：$F_{sample}=F_{i2cclk}/4$ … 111：$F_{sample}=F_{i2cclk}/128$ The output frequency is the (I2CCCR[4:0]+1) division of the sampling frequency, $F_{scl}=F_{i2cclk}/(2^{I2CCCR[7:5]}*(I2CCCR[4:0]+1))$ For example I2CCCR[4:0]=9, when I2CCR[7:5] is equal to 000：$F_{scl}=F_{i2cclk}/(1*10)$ 001：$F_{scl}=F_{i2cclk}/(2*10)$ 010：$F_{scl}=F_{i2cclk}/(4*10)$ … 111：$F_{scl}=F_{i2cclk}/(128*10)$ *Note：* *1. When I2CCCR[7:5] = 0, if a value less than 9 is written to I2CCR[4:0], I2CCR[4:0] is automatically counted as a value of 9.* *2. When I2CCCR[7:5] > 0, if a value less than 7 is written to I2CCR[4:0], I2CCR[4:0] is automatically counted as a value of 7.*

Table 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	1	1	1	1	1	0	0	0

Bit number	Bit Symbol	Description
7~0	I2CSTA	I^2C 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 Symbo	Description
7~1	-	-
0	I2CF	I^2Cinterrupt 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.*

18 PWM

18.1 PWM Function Introduction

CA51F1 series chip can include at most 3 channels PWM outputs. PWM period and duty cycle can be configured with 16-bit range.

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.

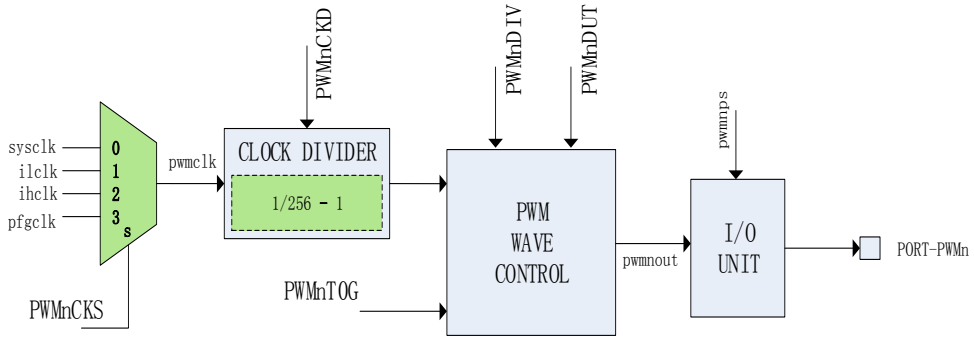


Figure 18-1-1 PWM Schematic

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

- **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. A PWM interrupt will be generated at this time.

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

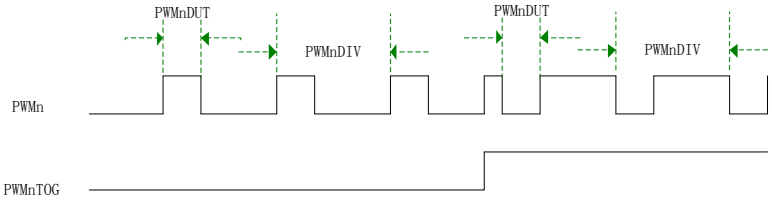


Figure 18-1-2 PWM Output Waveform

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.

- **cascade LED drive with single wire**

PWM1 channel supports cascade LED drive with single wire and drive sequence diagram for cascade LED is shown as follows.

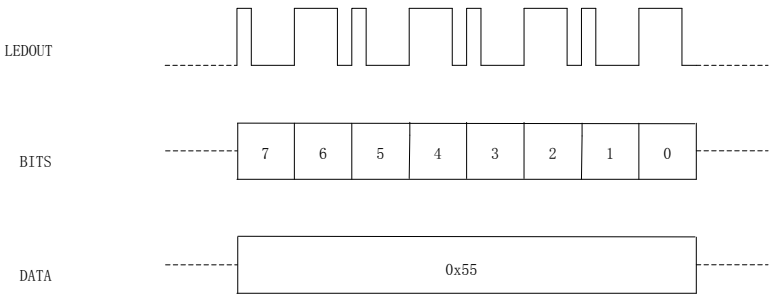


Figure 18-2-2 Sequence Diagram for Cascade LED

Figure 18-2-3 shows the bit coding.

Figure 18-2-3 Bit Coding

For the cascade LED sequence diagram, the high level time for bit 0 code is set byPWM1DUT while the high level time for bit 1 code is set by LEDUTH. The cycle time is set by PWM1DIV. When PWM1MOD≠0, the cascade LED drive is enabled and LEDAT is the data register for LED. When LEF=0, users can write LED data to LEDAT. Writing LEDAT will starts the data transfer for LED drive. When LED transmitter is transferring data, LEBSY is set to 1; while when the transmitter is idle, LEBSY becomes 0. There is one-byte buffer for LED transmitter. When there is data in both data register and buffer register, LEF is set to 1. When all the data in buffer register is has been sent, When all the data in buffer register is has been sent, at the same time, LEF is set to 0. LEF=0 shows LEDAT is ready for new data. When PWM1MOD≠0, it implies a waiting time will be inserted after PWM1MOD byte data is sent. The waiting time will be set by LEDWTM.

When PWM1POL=1, the data of LEDAT will be reversed. For example, if 01010101B is written, the data sent is actually 10101010B.

18.2 PWM Register Description

Table 18-2-1 Register PWMEN

C7H	7	6	5	4	3	2	1	0
PWMEN	-	-	-	-	-	PWM2EN	PWM1EN	PWM0EN
R/W	-	-	-	-	-	R/W	R/W	R/W
Initial Value	-	-	-	-	-	0	0	0

Bit number	Bit Symbol	Description
7~3	-	-
2	PWM2EN	PWM2 enable control，1 enables it
1	PWM1EN	PWM1 enable control，1 enables it
0	PWM0EN	PWM0 enable control，1 enables it

Table 18-2-2 Register PWMCON

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

Note: The following n=0, 1, 2.

Bit number	Bit Symbol	Description
7	-	-
6	PWMnTOG	PWMn output reverse control, 1 reverses it
5~3	PWM1MOD	When PWM1 is used for LED drive, it is the number of byte for one transfer. 0 indicated that PWM1 is not used for LED drive, while 1~7 indicates every time PWM1 sends 1~7 byte data it will pause once *Note：* *1. This configuration item is only available for PWM1CON.* *2. For more information please refer to LEDWTM.*
2	PWM1POL	When PWM1 is used as the LED driver, the transmit data take reverse enable control bit, 1 is valid *Note：* *1. This configuration item is only available for PWM1CON.* *2.* When PWM1MOD！=0，the corresponding PWM1POL is meaningful *3.* When PWM1POL=1, if the corresponding LEDAT=01010101B, then the data actually sent is 10101010B.
1~0	PWMnCKS	PWM working clock source selection 00：System clock 01：IRCL 10：IRCH

Table 18-2-3 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

Note: The following n=0, 1, 2.

Bit number	Bit Symbol	Description
7~0	PWMnCKD	PWM Operating Clock Prescaler Configuration Register 00H：No division 01H：frequency divided by 2 02H：frequency divided by 3 …... FEH：frequency divided by 255 FFH：frequency divided by 256

Table 18-2-4 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
Note: The following n=0, 1, 2.								
Bit number	Bit Symbol	Description						
15~0	PWMnDIV	PWMn Cycle Configuration Register						

Table 18-2-5 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
Bit number	Bit Symbol	Description						
15~0	PWMnDUT	PWMn Duty Cycle Configuration Register						

Table 18-2-6 Register LEDAT

96H	7	6	5	4	3	2	1	0
LEDAT	LEDAT[7:0]							
R/W	R/W							
Initial value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
7~0	LEDAT	LED driver data *Note： The LEDAT data is sent in order from MSB to LSB.*						

Table 18-2-7 Register LEDUTL、LEDUTH

92H	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
93H	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 sending data "1" duty ratio configuration register *Note:*						

		1. In the driving waveform of the cascaded LED, the period of each 1-bit data is determined by the corresponding PWM1DIV, and the duty cycle of data "1" is determined by LEDUT, and the duty cycle of data "0" is determined by PWM1DUT; *2. If LEDAT=01010101B and corresponding PWMPOL=1, then the actual data sent is 1-0-1-0-1-0-1-0 in the order of BIT7-BIT6-BIT5-BIT4-BIT3-BIT2-BIT1-BIT0 -0, and 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, that is, LEDUT takes effect after PWMPOL.*

Table 18-2-8 Register LEDWTML、LEDWTMH

94H	7	6	5	4	3	2	1	0
LEDWTML	LEDWTM[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
LEDWTMH	LEDWTM[15:8]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit Number	Bit Symbol	Description						
15~0	LEDWTM	LED pause time configuration register, used in combination with PWM1MOD configuration register *Note ：* *After each PWM1MOD byte data transmission, pause (LEDWTM+1) PWM operating clock and go to the next transmission*						

Table 18-2-9 Register LEFLG

97H	7	6	5	4	3	2	1	0
LEFLG	-	-	-	-	-	LEBSY	-	-
R/W	-	-	-	-	-	R	-	-
Initial Value	-	-	-	-	-	0	-	-
Bit Number	Bit Symbol	Description						
7~3	-							
2	LEBSY	LEDAT data sending busy flag, 1 indicates that the data in the LEDAT data cache has not been sent, 0 indicates that all the data has been sent						
1~0	-	-						

18.3 PWM Function Control Example

◆ **Single channel PWM output**

Take PWM0 as an example, the PWM clock source is IRCH (IRCH frequency is 16MHz), the output frequency is 30K clock, the duty cycle is 30%, the program is like:

```
---------------------------------------------------------------------------------------
//PWMxCON
#define TOG(n)              (n<<6)
#define PWM_CKS_IH          (2<<0)
void PWM_init(void)
{
    PWM0CON = TOG(1)|PWM_CKS_IH;    // Set PWM clock, whether PWM is inverted output
    PWM0CKD = 0;                    // Set the frequency division factor, set to 0 means no division
    PWM0DIVH = 0x02;                // Set DIV value, 16000000/30000=0x215
    PWM0DIVL = 0x15;
    PWM0DUTH = 0x00;                // Set the DUT value with a duty cycle of 30%
    PWM0DUTL = 0xA0;
    P32F = 6;                       // Set P32 to PWM pin function
    PWMEN |= (1<<0);                // PWM0 enable
}
---------------------------------------------------------------------------------------
```

◆ **Single-wire cascade LED driver strip control routines**

Take PWM1 as an example, to drive 8 levels of RGB LEDs, so that the RGB LEDs can change color cyclically, the program is like:

```
---------------------------------------------------------------------------------------
//PWMxCON
#define TOG(n)              (n<<6)
#define PWM_CKS_SYS         (0<<0)
#define PWM_CKS_IL          (1<<0)
#define PWM_CKS_IH          (2<<0)

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

void PWM_init(void)
{
    PWM1CON = TOG(0)| PWMMOD(3) | PWMPOL(0)   | PWM_CKS_IH;       // Set IRCH as PWM clock source and insert pause time after sending 3 bytes
    PWM1CKD = 0;                    // Set the frequency division factor, set to 0 means no division
    PWM1DIVH  = 0;                  // Set bit cycle time
    PWM1DIVL  = 20;
```

```
    PWM1DUTH =     0;            // Set Bit Code 0 Time
    PWM1DUTL  = 6;
    LEDUTH= 0;                   // Set Bit Code 1 Time
    LEDUTL = 13;
    LEDWTMH   = 0;        // Set pause time
    LEDWTML   = 50;
    P34F = 6;                    // Set P34 to PWM pin function
    PWMEN |= (1<<1);             //enable PWM1
}
void main(void)
{
    PWM_init(void);
    while(1)
    {
        unsigned char i;
        static unsigned char color_index = 0;
        code unsigned char LED_DAT[][3] =
        {
            {0xff,0x00,0x00},
            {0xff,0xff,0x00},
            {0x00,0xff,0x00},
            {0x00,0xff,0xff},
            {0x00,0x00,0xff},
            {0xff,0x00,0xff},
        };
        for(i=0;i<24;i++)
        {
            while(LEFLG & LEF0);
            LEDAT = LED_DAT[color_index][i%3];
        }
        color_index++;
        if(color_index>=6)
        color_index=0;
        Delay_ms(500);
    }
}
```

---

19 Analog/Digital Converter（ADC）

19.1 Function Introduction

Analog/digital converter is a 12-bit successive approximation(SAR) ADC, with at most 6 input channels. The clock source for ADC is the system clock with frequency division configurable. There are ADC multiple reference voltages for ADC. When internal voltage is selected as the reference voltage, it can be used to test the power supply voltage for the chip and there will be correction to ensure the chip's consistency as well.

19.2 Main Features

- 12-bit resolution
- 6 input channels at most
- ADC clock frequency division configurable
- Alternate reference voltage： internal reference voltage, VDD
- Support VDD and reference ground voltage measurements
- Support automatic data correction when internal reference voltage is selected
- Input voltage range：VSS<=VIN<=VDD。

19.3 Structure Block Diagram

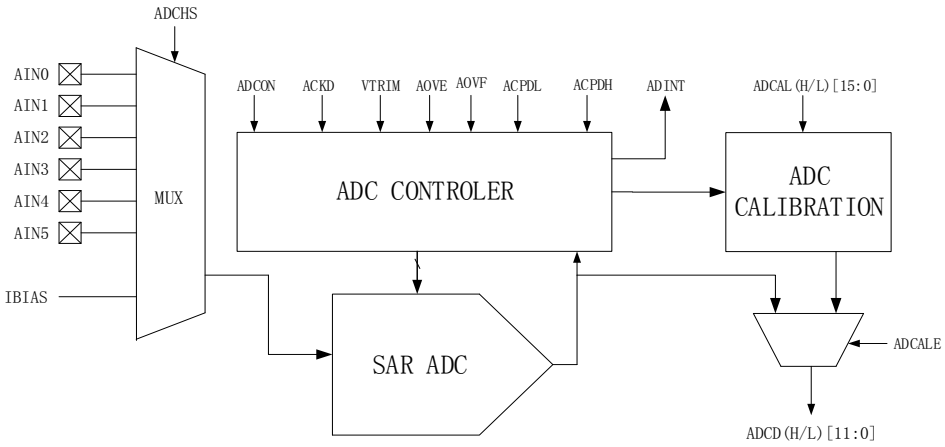


Figure 19-3-1 ADC Architecture

19.4 Function Introduction

ADC can be enabled by AST. When AST=1, the input voltage selected by ADCHS will be analog/digital

converted. The clock for ADC is the system clock with frequency division set by ACKD beforehand. When ADC clock is constant, the time for single conversion is set by HTME. The conversion time is (13+2^HTME) ADC clock cycle periods. 12-bit A/D will be stored in register ADCDH and ADCDL after the conversion. AST will be cleared automatically 2.5 clock cycles later. The interrupt flag ADCIF will be set to 1 at the same time. If ADC interrupt is enabled then, ADC interrupt occurs. Figure 19-4-1 is the sequence diagram for ADC conversion.

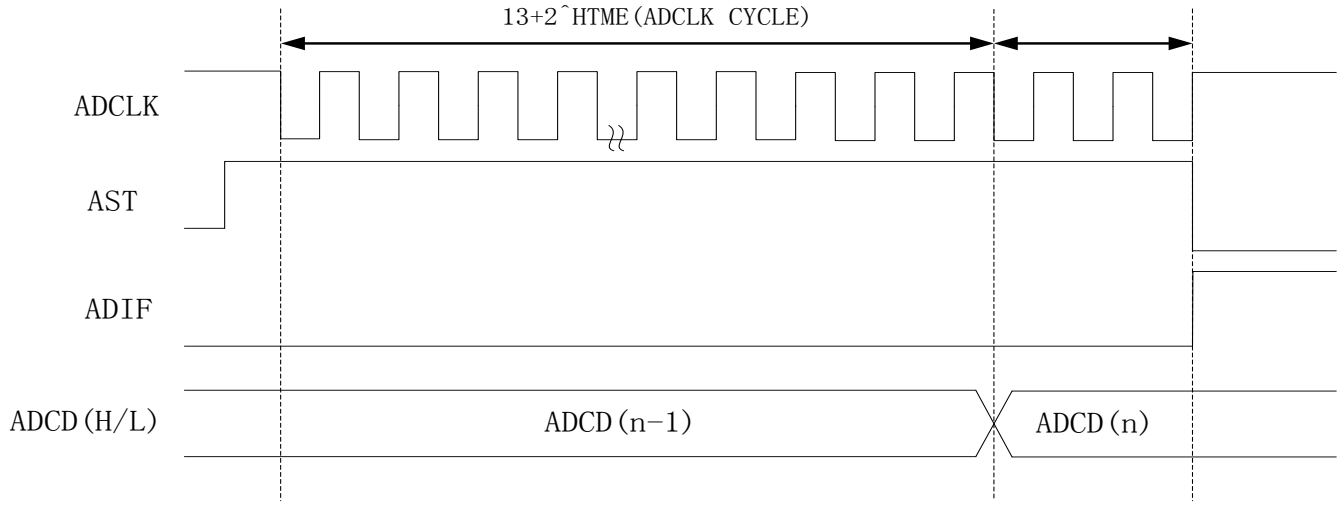


Figure 19-4-1 ADC Sequence Diagram

- **ADC Data Calibration**

When internal voltage(1.5V) is selected as the reference voltage, due to the discreteness of the chips, the internal voltage in each chip can not be exactly the same which induces different ADC conversion results consequentially. Thus, it is necessary to correct the AD value after the conversion. The internal voltage will be tested and a correction value will be obtained when chips leave factory. When the chip's powered on, the correction value will be loaded into register ADCALL and ADCALH. when the ADC conversion is completed, the AD value is automatically converted in equal proportion according to the value of correction registers ADCALL and ADCALH, resulting in an accurate AD value. The final accurate result for AD will be stored in register ADCD. The function can be enabled by ADCALE. Users only need to set ADCALE=1 and the correction will be done automatically.

19.5 Register Description

Table 19-5-1 Register ADCON

8060H	7	6	5	4	3	2	1	0
ADCON	AST	ADIE	ADCIF	HTME			-	VSEL
R/W	R/W	R/W	R/W	R/W			-	R/W
Initial Value	0	0	0	0	1	0	0	0

Bit Number	Bit Symbol	Description
7	AST	ADC conversion enable control, the conversion starts when 1 is written to it, the hardware will clear it automatically after the conversion
6	ADIE	ADC interrupt enable control, 1 enables it
5	ADCIF	ADC interrupt enable control, 1 enables it
4~2	HTME	The number of sampling periods is 2 power HTME
1	-	-
0	VSEL	ADC reference voltage selection 0: Internal 1.5V (INNER_VREF) as the reference voltage 1: Power supply as reference voltage

Table 19-5-2 Register ADCFGL

8061H	7	6	5	4	3	2	1	0
ADCFGL	ACKD			ADCALE	ADCHS			
R/W	R/W			R/W	R/W			
Initial Value	0	0	0	1	0	0	0	0

Bit Number	Bit Symbol	Description
7~5	ACKD	ADC clock frequency division setting 000： no division 001： frequency divided by 2 010： frequency divided by 4 … 111： frequency divided by 14
4	ADCALE	ADC calibration enable control, 1 enables it Valid only when the internal 1.5V is selected as the reference voltage. When ADCALE=1, ADC conversion result will be calibrated according to register ADCAL. For more information please refer to register ADCAL description
3~0	ADCHS	ADC channel enable selection 0000： disable the channels 0001： enable channel AD_CH[0](P31) 0010： enable channel AD_CH[1](P35) 0011： enable channel AD_CH[2](P32) 0100： enable channel AD_CH[3](P34)

		0101： enable channel AD_CH[4](P01) 0110： enable channel AD_CH[5](P30) 0111： enable 1/4 VDD detection Others： disable the channel

Table 19-5-4 Register ADCAL

8065H	7	6	5	4	3	2	1	0
ADCALL	ADCAL[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
8065H	7	6	5	4	3	2	1	0
ADCALH	ADCAL[15:8]							
R/W	R/W							
Initial Value	0	0	0	1	0	0	0	0

Bit Number	Bit Symbol	Description
15~0	ADCAL	ADC calibration register, valid only when ADCALE=1 and the internal 1.5V is selected as reference voltage. When it is valid, the ADC output is ： ADCDL=(ADC conversion result*ADCAL)/32768

Table 19-5-5 Register ADCD

8062H	7	6	5	4	3	2	1	0
ADCDL	ADCDL[3:0]				-			
R/W	R				-			
Initial Value	0	0	0	0	-	-	-	-
8063H	7	6	5	4	3	2	1	0
ADCDH	ADCDH[11:4]							
R/W	R							
Initial Value	0	0	0	0	0	0	0	0

Bit Number	Bit Symbol	Description
11~0	ADCD	ADC conversion result

19.6 ADC Control Example

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

program is like：

```
--------------------------------------------------------------------------------
void main(void)
{
   P31F = P31_ADC0_SETTING;
   ADCON = AST(0) | ADIE(0) | HTME(7) | VSEL(ADC_REF_VDD);        // Set ADC reference voltage to VDD
         ADCFGL = ACKD(7) | ADCALE(1) | ADCHS(ADC_CH0);      // Select ADC0 channel
   while(1)
   {
         unsigned int AD_Value;
         ADCON |= AST(1);                                    // Enable ADC conversion
         while(!(ADCON & ADIF));                             // Wait for ADC conversion to complete
         ADCON |= ADIF;                                      // Clear ADC interrupt flag
         AD_Value = ADCDH*256 + ADCDL;                       // Read AD value
         AD_Value >>= 4;
   }
}
   --------------------------------------------------------------------------------
```

20 Capacitive touch keys（Touch Key）

20.1 Function Introduction

The touch function module of CA51F1 series chip has superior anti-interference performance and can pass EFT, CS test and etc. The touch module can support 5 channels at most. For applications with low power consumption requirements, the chip is also designed to work and wake up normally when in STOP mode to achieve the product power saving function.

20.2 Main Features

- High anti-interference performance in accordance with EMC(CS) standards
- Supports 5 channels at most
- Supports low power consumption mode
- Touch interrupt support
- 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

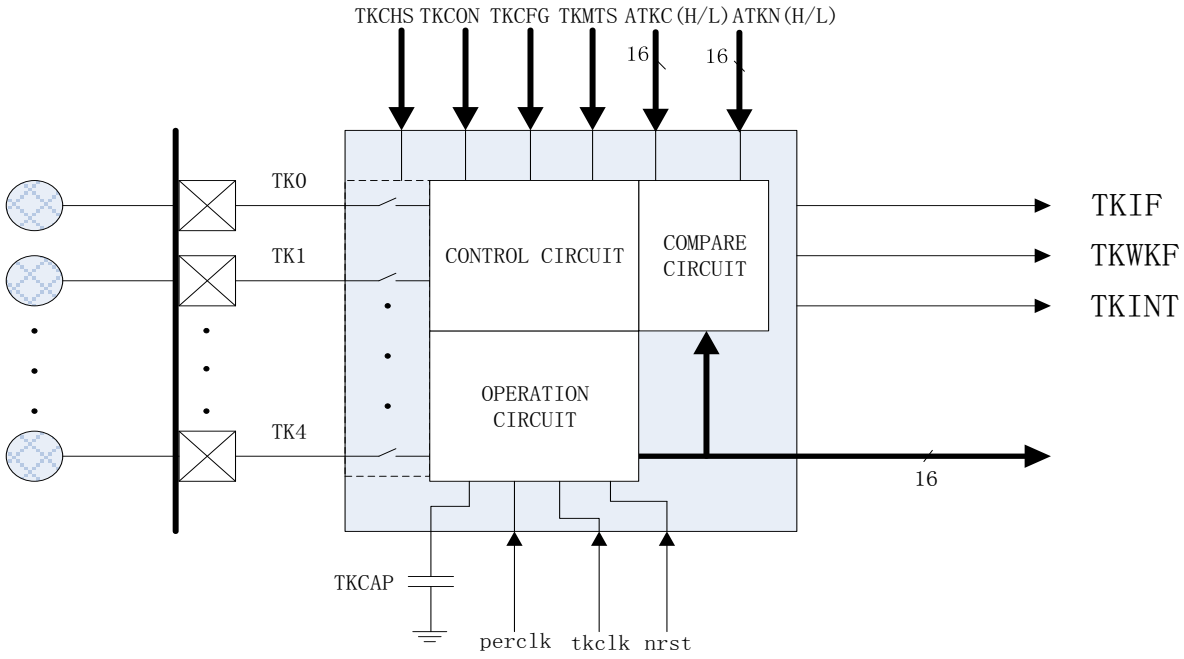


Figure 20-3-1 Touch Key Module Architecture

20.4 Function Description

20.4.1 Enable Touch Channel

Touch module supports a total of five external channels and one internal channel, when the touch external channel corresponding to the pin function register is set to 4, touch channel enable, for example: channel 0 corresponding to the pin P31, when P31F is set to 4, touch channel 0 enable. For the internal reference channel, set TCAPSEL to 1 to enable.

20.4.2 Manual Control Mode and Automatic Mode

Set TMEN=0 to select for manual mode. In manual mode, touch data acquisition is initiated via the TKST bit. When TKST=1 is set, the touch module starts data acquisition for the corresponding touch channel. When data acquisition is complete, the TKST bit is automatically cleared to 0, the interrupt flag bit TKIF of the corresponding channel is set to 1, and the acquired touch data is stored in register TKnMS (TK0~TK4 corresponds to TK0MS~TK4MS, and the internal reference channel corresponds to TK5MS).

Set TMEN=1 to select the automatic mode. Unlike the manual mode, the touch data acquisition in auto mode is started by the timer timer, the clock source of the timer is IRCL, and the timing time is set by register TKMTS.

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

20.4.3 Touch Key Clock Frequency Division

The clock source of the touch controller for charging and discharging the touch electrodes is the 4-division frequency of IRCH or IRCL. The clock frequency of charging and discharging is critical to the performance of touch, and when the charging and discharging frequency is too high, it may cause insufficient charging of the touch electrodes and thus lead to a smaller amount of touch data change when the finger touches. The frequency division can be set by TKDIV. and by setting a reasonable value, the touch performance can be better. Note that the TKDIV setting is not valid after the touch frequency hopping function is enabled.

20.4.4 Low Power Consumption Mode

In order to achieve the touch function of low-power applications, the touch module is designed with the corresponding power saving mechanism. In STOP mode, as long as the touch charge and discharge clock source (IRCH or IRCL, due to the high static power consumption of IRCH, touch power saving mode generally choose IRCL as the touch charge and discharge clock) and low speed clock (IRCL) is on, the touch module can maintain normal charge and discharge and counting. When the touch acquisition is complete, touch acquisition completion interrupt will wake up the CPU, the software can read the touch data after the CPU wakes up, and then enter the STOP mode again.

20.4.5 Touch Frequency Hopping Function

To enhance the touch anti-voltage pulse injection performance, the chip is designed with a touch frequency hopping function.

Touch frequency hopping function is enabled by the FAEN bit, after enabling, touch the charge and discharge clock in the touch charge and discharge source clock (i.e. IRCH quadrature or IRCL) on the basis of each charge and discharge in order to 2, 3, 4, 5 divisions of the order of change, can effectively reduce the voltage pulse injection when a frequency band injection interference signal on the touch interference amplitude.

20.6 Register Description

Table 20-6-1 Register TKCON

F8H	7	6	5	4	3	2	1	0
TKCON	TKST	TKIE	TMEN	FAEN	TCAPSEL	VRS[2:0]		
R/W	R/W	R/W	R/W	R/W	R/W	R/W		
Initial Value	0	0	0	0	0	1	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	FAEN	0: not enable frequency adjustment 1: frequency regulation is enabled, the touch clock is switched once per cycle, and the frequency has the 1-2-3-4 divisions of the reference frequency switched in turn, once per cycle.
3	TCAPSEL	Built-in reference channel selection bit, 1 valid
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 20-6-2 Register TKPWC

DFH	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

Bit Number	Bit Symbol	Description
7~6	TKPC	Touch button undersampled channel output control 00: Suspend 01: Output low 10: Output compensation (i.e. waterproof compensation mode)
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: 1.5V 01: 2.0V 10: 2.5V 11: 3.0V
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 20-6-3 Register TKCKS

DEH	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 (100K) clock

Table 20-6-4 Register TKCFG

F9H	7	6	5	4	3	2	1	0
TKCFG	TKDIV			TKTMS				
R/W	R/W			R/W				
Initial Value	0	0	0	1	1	1	1	1

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

Table 20-6-5 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 Start-up time = (TKMTS+1) x 128 x IRCL clock period. Since the IRCL clock frequency is 100KHz, the time range is 1.28ms - 328ms.

Table 20-6-9 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 20-6-10 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	Internal reference channel touch acquisition interrupt flag bit, write 1 to clear 0
4	TKIF4	External touch channel 4 (TK4) touch acquisition interrupt flag bit, write 1 to clear 0
3	TKIF3	External touch channel 3 (TK3) touch acquisition interrupt flag bit, write 1 to clear 0
2	TKIF2	External touch channel 2 (TK2) touch acquisition interrupt flag bit, write 1 to clear 0
1	TKIF1	External touch channel 1 (TK1) touch acquisition interrupt flag bit, write 1 to clear 0
0	TKIF0	External touch channel 0 (TK0) touch acquisition interrupt flag bit, write 1 to clear 0

21 Low Voltage Detection（LVD）

21.1 Function Introduction

Low voltage detection (LVD) is used to monitor the chip's own power supply VDD, with detectable range 2.2V/2.7V/3.7V/4.2V (four levels are selectable). When VDD is lower than the voltage set, either interrupt or reset occurs

Note: Due the manufacturing process, the LVD trigger voltage may be slightly different.

Figure 21-1-1 shows the architecture of LVD

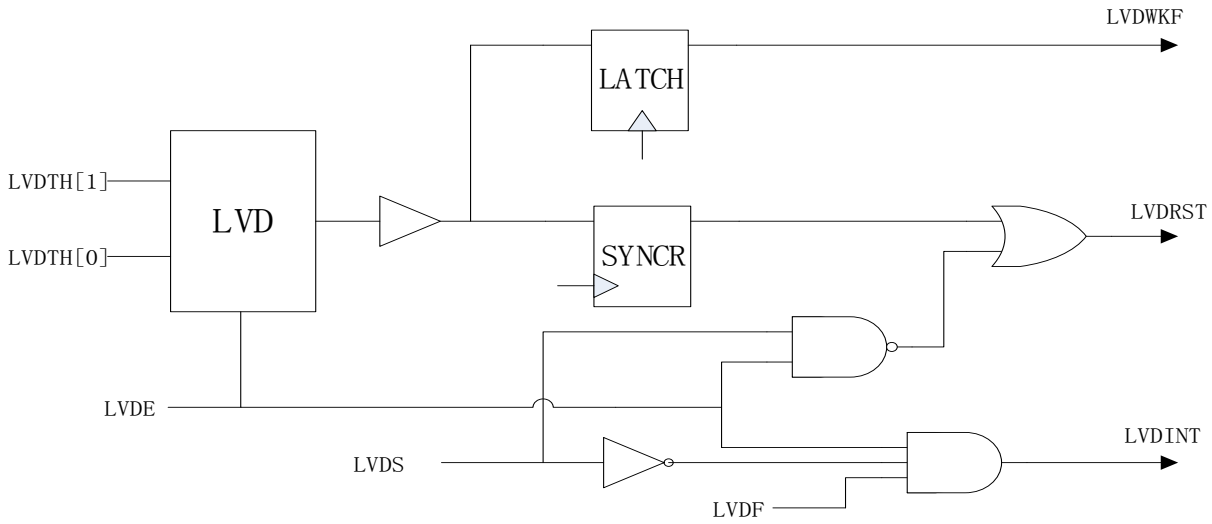


图 21-1-1 LVD Schematic

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

EFH	7	6	5	4	3	2	1	0
LVDCON	LVDE	LVDS	LVDF				LVDTH[1:0]	
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	LVDE	LVD enable bit, 1 valid
6	LVDS	LVD function selection bit 0：interrupt 1：reset
5	LVDF	LVD generate flag bit, write 1 to clear 0
1~0	LVDTH	LVD trigger level selection bit field 00：2.2V 01：2.7V 10：3.7V 11：4.2V

21.4 LVD control example

LVD interrupt examples

For instance，set LVD interrupt mode，detecting voltage 2.2V，the procedures are as follows：

```
-------------------------------------------------------------------------------------------
#define LVDE(N)                 (N<<7)          //N=0~1
#define LVDS_reset      (1<<6)
#define LVDS_int        (0<<6)
#define LVDF                    (1<<5)

#define LVDTH_2p2V          0
void LVD_init(void)
{
    LVDCON = LVDE(1) | LVDF | LVDS_int | LVDTH_2p2V;  //set LVD enable, set LVD interrupt mode, detect voltage 2.2V
    INT4EN = 1;   //INT4 interrupt enable
    EA = 1;           //turn on total interrupt
}

void LVD_ISR(void) interrupt 6              //LVD interrupt procedure
{
    if(LVDCON & LVDF)
    {
        LVDCON |= LVDF;                     //clear LVD interrupt flag
        /LVD interrupt procedure

    }
}
-------------------------------------------------------------------------------------------
```

LVD Reset example

For instance，set LVD reset mode，detecting voltage 2.2V，the procedures are as follows:：

```
-------------------------------------------------------------------------------------------
#define LVDE(N)         (N<<7)          //N=0~1
#define LVDS_reset      (1<<6)
#define LVDS_int        (0<<6)
#define LVDF                    (1<<5)

#define LVDTH_2p2V          0
void LVD_init(void)
{
    LVDCON = LVDE(1) | LVDF | LVDS_reset | LVDTH_2p2V;// set LVD enable, set LVD reset mode, detecting voltage 2.2V
}
-------------------------------------------------------------------------------------------
```

22 Program Download & Simulation

22.1 Program Download

CA51F152 Series Chips Mainly uses ISP method to download programs，which connects to the download tool via I2C interface，The upgrade interface is P30(I2C SDA),P31(I2C SCL)

For more details about the program download procedure, please refer to the "CACHIP Development Download Tool Instructions".

22.2 Online Simulation

CA51F2 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 Specification

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	℃

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 Specification

Parameter	Symbol	Operating Voltage	Minimum	Typical	Maximum	Unit	Test conditions
Working curren	Iop1	VDD=2.5V		2.26		mA	The system clock is IRCH (16MHz), with other clocks disabled. No load for all the output pins. No floating for digital input pins. All the peripherals are disabled, with CPU executing instruction NOP
		VDD=3.3V		2.90			
		VDD=5V		4.43			
	Iop3	VDD=2.5V		24.7		uA	The system clock is IRCL (100kHz), with other clocks disabled. No load for all the output pins. No floating for digital input pins. All the peripherals are disabled, with CPU executing instruction NOP
		VDD=3.3V		28.9			
		VDD=5V		38.6			
Current for STOP mode	I_{stp}	VDD=2.5V		2.0		uA	All the clocks are disabled. No load for all the output pins. No floating for digital input pins. All the peripherals are disabled. flash enters sleep mode and CPU enters STOP mode.
		VDD=3.3V		2.1			
		VDD=5V		2.3			
Current for IDLE mode	I_{idl1}	VDD=2.5V		0.96		mA	The system clock is IRCH (16MHz), with other clocks disabled. No load for all the output pins. No floating for digital input pins. All the peripherals are disabled, flash enters sleep mode and CPU enters IDLE mode.
		VDD=3.3V		1.26			
		VDD=5V		2.05			

	I_{idl2}	VDD=2.5V		9.0		uA	The system clock is IRCL (100Khz), with other clocks disabled. No load for all the output pins. No floating for digital input pins. All the peripherals are disabled, CPU enters IDLE mode.
		VDD=3.3V		11.2			
		VDD=5V		16.5			
High voltage for IO port input (Smit mode on)	V_{hi1}	VDD=2.5V	1.20	-	2.5	V	-
		VDD=3.3V	1.60		3.3		
		VDD=5V	2.20		5		
High voltage for IO port input (Smit mode off)	V_{hi2}	VDD=2.5V		0.5*VDD	VDD	V	-
		VDD=3.3V					
		VDD=5V					
Low voltage for IO port input (Smit mode on)	V_{lo1}	VDD=2.5V	0	-	1.00	V	-
		VDD=3.3V	0	-	1.30		
		VDD=5V	0	-	1.90		
Low voltage for IO port input (smit mode off)	V_{lo2}	VDD=2.5V	0	0.5*VDD		V	-
		VDD=3.3V					
		VDD=5V					
General GPIO push current	I_{pu}	VDD=3.3V	-	3.27	-	mA	IO set to push-pull output mode, drive capability set to maximum，Vol=VDD-0.3V
		VDD=5V	-	4.12	-		
IO port Irrigation current	I_{ol}	VDD=3.3V	-	5.63	-	mA	IO set to push-pull output mode, drive capability set to maximum，Vol=GND+0.3V
		VDD=5V	-	7.35	-		
IO port pull-down resistor	R_{d1}	VDD=2.5~5.5V		30		KΩ	-
IO port pull-up resistor	R_{u1}	VDD=2.5~5.5V	-	30	-	KΩ	-

23.3 AC Electrical Specification

AC Electrical Specification(VDD=2.2-5.5V, TA=25℃, unless there are other explanations)

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Conditions
Time to start oscillation for IRCL	Trc1	-	50	-	us	IRCL frequency is 100K
Time to start oscillation for IRCH	Trc2	-	10	-	us	IRCH frequency is 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%..

23.4 Minimum operating voltage

CPU Frequency (Unit：MHz)	**Minimum operating voltage （Unit:V）**
8	2.2-5.5V
16	2.7-5.5V

23.5 Temperature Characteristic for Internal High Speed RC

◆ IRCH Temperature Characteristics

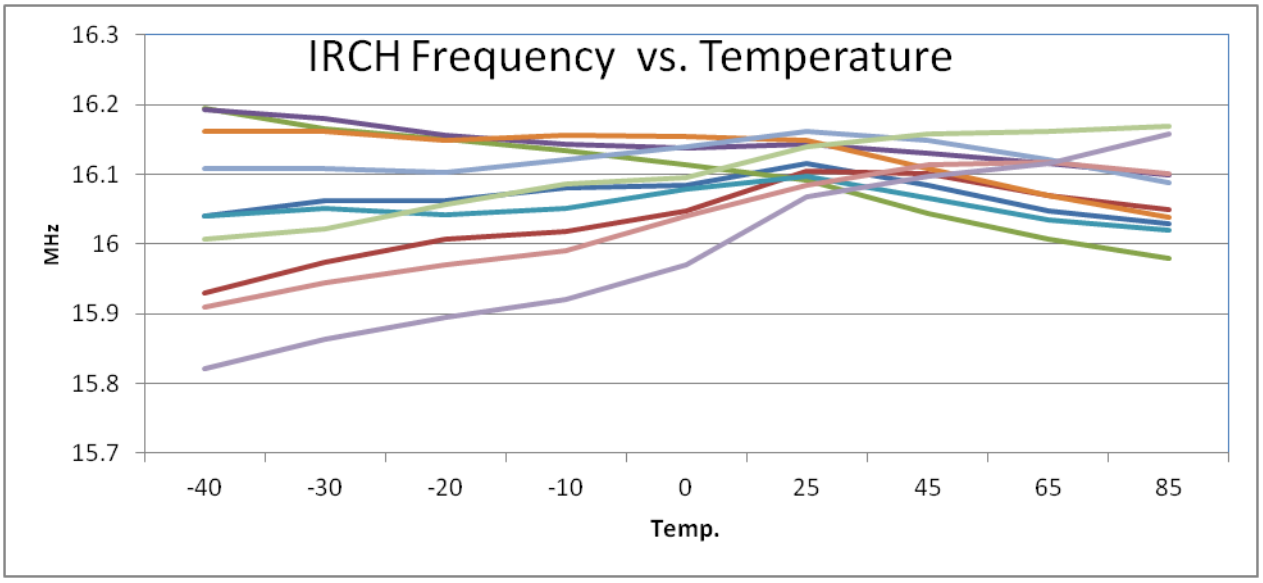


Figure 23-5-1 IRCH Temperature characteristics graph

Note: The above graphical data is a random sample of 10 chips measured, the data is for reference only.

◆ IRCL Temperature Characteristics

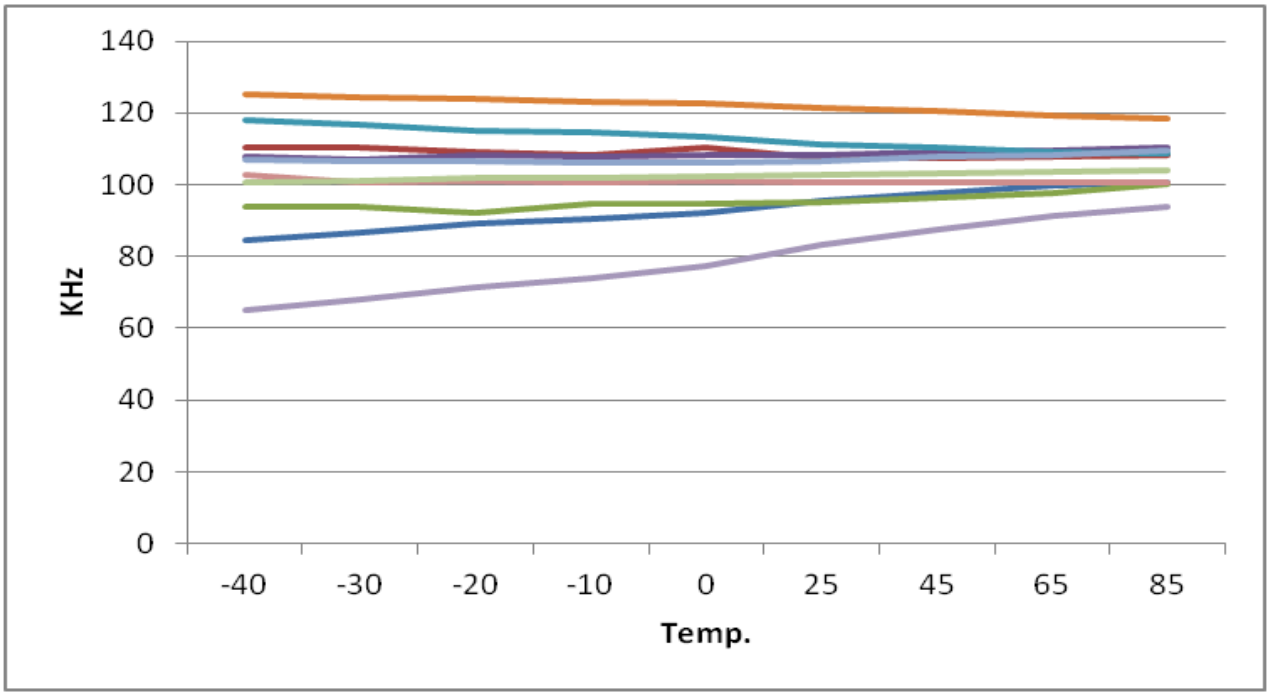
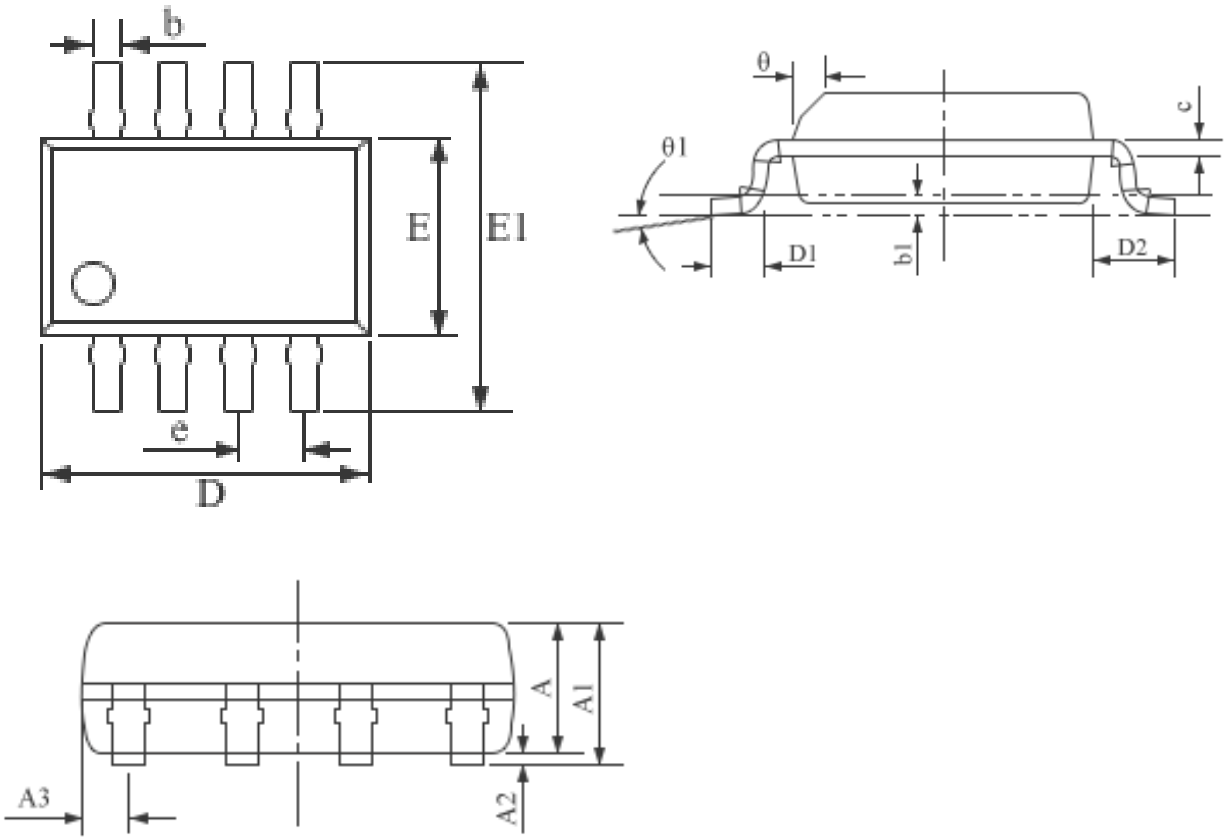


Figure 23-5-2 IRCL Temperature characteristics graph

Note: The above graphical data is a random sample of 10 chips measured, the data is for reference only.

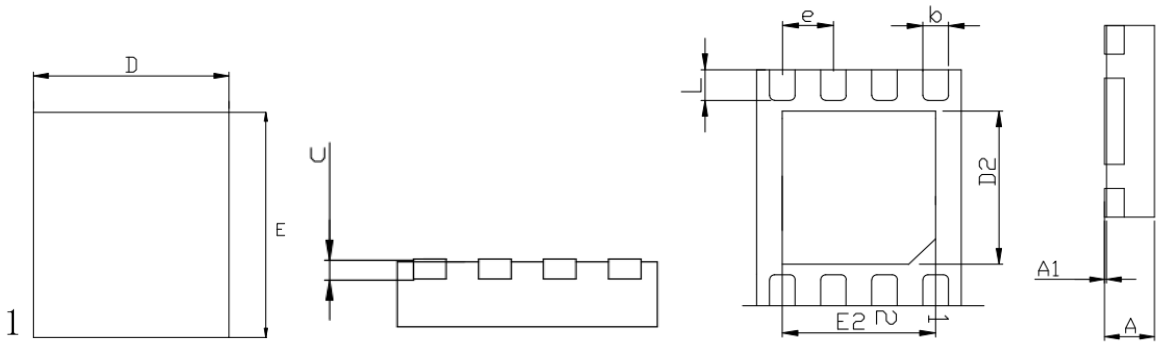
24 Package Type

Package (I) (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 (II) (DFN8 2X3mm)



序号	最小值(mm)	标准值(mm)	最大值(mm)
A	0.70	0.75	0.80
A1	-----	0.02	0.05
b	0.23	0.28	0.33
c		0.203BSC	
D	1.90	2.00	2.10
D2	1.40	1.50	1.60
e		0.50BSC	
E	2.90	3.00	3.10
E2	1.40	1.50	1.60
L	0.28	0.33	0.38

25 Appendix

Appendix 1 Instruction Set Quick Reference Table

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

ARITHMETIC OPERATIONS			
ADD A, Rn	Add register to A	(A) ← (A) + (Rn)	1
ADD A, direct	Add direct byte to A	(A) ← (A) + (direct)	1
ADD A, @Ri	Add indirect RAM to A	(A) ← (A) + ((Ri))	1
ADD A, #data8	Add 8-bit immediate data to A	(A) ← (A) + #data	1
ADDC A, Rn	Add register to A with Carry	(A) ← (A) + (C) + (Rn)	1
ADDC A, direct	Add direct byte to A with Carry	(A) ← (A) + (C) + (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	1

		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)
		
		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	(direct) ← (direct) ^	2

	byte	#data	
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
		ELSE
		(C) ← 0
			2

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

		THEN
		(PC) ← (PC) +
		relative offset
		IF ((Ri)) < data
		THEN
		(C) ← 1
		ELSE
		(C) ← 0	

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		