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

CA51F2 Series MCU User Guide

REV 2.2

Table of Contents

1 Introduction

CA51F2 series is 8-bit MCU based on 1T 8051 core and operates 10 times faster than the traditional 8051 chips with definitely better performance. The Flash program memory embedded can be programmed for times and offers users with three storage choices (8/16/32K) which brings great convenience to software development. Not only traditional 8051 chip features, CA51F2 also includes 12bit ADC, LCD/LED driver, Touch key, 16bit PMW, UART, RTC, brushless DC motor driver, multiplier/divider, LVD, and other function modules. It can operate in three Power Saving Modes (IDLE/STOP/LOW SPEED) in order to meet different power consumption needs. With great functions and anti-jamming feature, CA51F2 can be used in various fields such as car's or family audio system, small household appliance, bluetooth stereo, car's electronics, digital motors, sports equipment, motor control, health care, instrument and meters, security guard, power control, factory control and doorbell products.

2 Basic Features

- Core
  - CPU： 1T 8051, with highest speed 10 times faster than traditional 8051
  - Compatible with 8051 instruction set, with double DPTR mode
  - CPU frequency： Highest at 27MHz
- Memory
  - Flash : 8 / 16 / 32K byte , can be erased and overwritten for times
  - Flash could be divided into program storage and data storage. Data storage could be used to store data which need to be protected during power off and EEPROM can be omitted
  - RAM:256 bytes internal RAM, 2K bytes external RAM
- Operating Voltage
  - Operating Voltage： 1.8V - 5.5V
- Clock System
  - External High Speed Oscillator： 1 - 27MHz
  - External RTC Oscillator： 32.768KHz
  - Internal Low Speed RC Oscillator： 131KHz
  - Internal PLL： The ratio for frequency multiplication ranges from 2 to 10. Reference clock ranges from 2 to 4 MHz in Internal RC Oscillator
  - Internal High Speed RC Oscillator： 2 - 4MHz, with 1% precision (The factory original frequency is 3.6864MHz＠3.3V/25℃)
  - With External Clock Monitor Module embedded, the Clock System is able to monitor external clocks' status so that the external clocks will not crash due to oscillation stop

- RTC
  - The internal RTC module can count hours, minutes, days and weeks. It may also be used as alarm clock
  - Supports microsecond/half second interrupt function
- Interrupt System
  - 15 effective interrupt sources
  - Two levels for interrupt priority which also supports interrupt nesting
  - 10 external interrupt sources. For each external interrupt, any of the signal pin could be configured as interrupt input pin
- Timer
  - Three 16-bit general Timers: Timer 0, Timer 1, Timer 2
- General Purpose IO (GPIO)
  - Supports 62 GPIO at most（different models will be different）
  - Support push-pull, open-drain, strong pull-up, weak pull-up, strong pull-down, weak pull-down, high resistance mode
  - Different drive strength and flip speed can be set in push-pull mode
- Touch Key
  - Internal Touch Sensor Controller
  - Supports 24 touch channel at most
  - Touch can set internal charging and internal reference, can effectively suppress power supply low frequency interference
  - Supports multiplexing of touch pins and LED driver pins
  - Internal waterproof compensation mechanism
  - Excellent anti-jamming performance which conforms to EMC(CS) Standard
  - Support touch power saving mode, the lowest power consumption is less than 10uA
- Analog/Digital Converter(ADC)
  - Support 8-channel 12-bit SAR ADC with built-in op-amp and comparison function
  - Supports 3 Reference Voltage： VDD、Internal Reference Voltage, External Reference Voltage
  - When Internal Reference Voltage is selected, VDD could be measured as well
  - Supports signal amplification and narrowing with gain configurable
- PWM
  - Supports 8 channel PWM, any periods or duty cycles are configurable in 16 bits
  - Supports Complementary Mode and Deadtime Control which could be used to drive brushless DC motor
  - Supports center fixed mode or edge fixed mode
  - Supports to output internal clock directly
  - Supports PWM Interrupt
- LCD Driver
  - Supports 8com x 32seg、7com x 33seg、6com x 34seg、5com x 35seg、4com x 36seg at most
  - Configurable Duty Cycle：1/2、1/3、1/4、1/5、1/6、1/7、1/8 Duty
  - Bias voltage configurable：1/2、1/3、1/4 Bias
  - Supports 8 levels contrast adjustment
  - Supports 3 levels drive current which enables the user to modify according to different LCD screen

- LED Driver
  - Supports 8com x 32segm. at most
  - Supports 8 levels brightness adjustment
- Low Voltage Detector(LVD)
  - Voltage detectable ranges from 1.8 V to 4.8V which is also configurable
  - Low voltage reset/interrupt configurable
- Reset Mode
  - Supports variable reset sources： Hard Reset, Soft Reset, Watch Dog Reset, LVD Reset, Power On/Down Reset
- Watch Dog
  - 27bit Watch Dog Timer, 16 bits precision configurable, with Watch Dog Reset and Interrupt configurable as well
- Remote Receiver
  - Sampling counter module embedded(SAMPLE), with pulse width set by hardware module, which reduces the software codes needed
- UART
  - Supports 3 UART ports at most
  - Supports 1 byte receive buffer
- SPI
  - One 4-wire SPI port which supports Master-Slave mode
- I^2C
  - One I2C port embedded which supports Master-Slave mode and Standard/Fast/High Speed mode as well
  - I2C can set digital filtering to enhance I2C anti-interference performance
- Operational Amplifier and Analog Comparator
  - 4 analog comparators, 2 OPAMP and 1 capture counter
  - Internal or external input voltage can be selected as the reference voltage for analog comparator
  - 15-bit digital filter for analog comparator, supports the comparator interrupt
  - OPAMP, ADC and analog comparator used to expand detectable voltage range
  - Capture counter and analog comparator can be used together to measure the motor’s speed and detect locked rotor
- Brushless DC Motor Driver
  - 60° Hall and 120° Hall decoding module embedded
  - Manual control and automatic mode supported, with brake function as well
  - Multiple abnormalities detection
  - Supports DC motor driver without Hall when using analog comparator
- Multiplying/Dividing Unit(MDU)
  - Supports 16bit × 16bit multiplication in one clock cycle
  - Supports 32bit ÷ 32bit division in eight clock cycles
  - Supports 32 bits data left/right shifting in one clock cycle

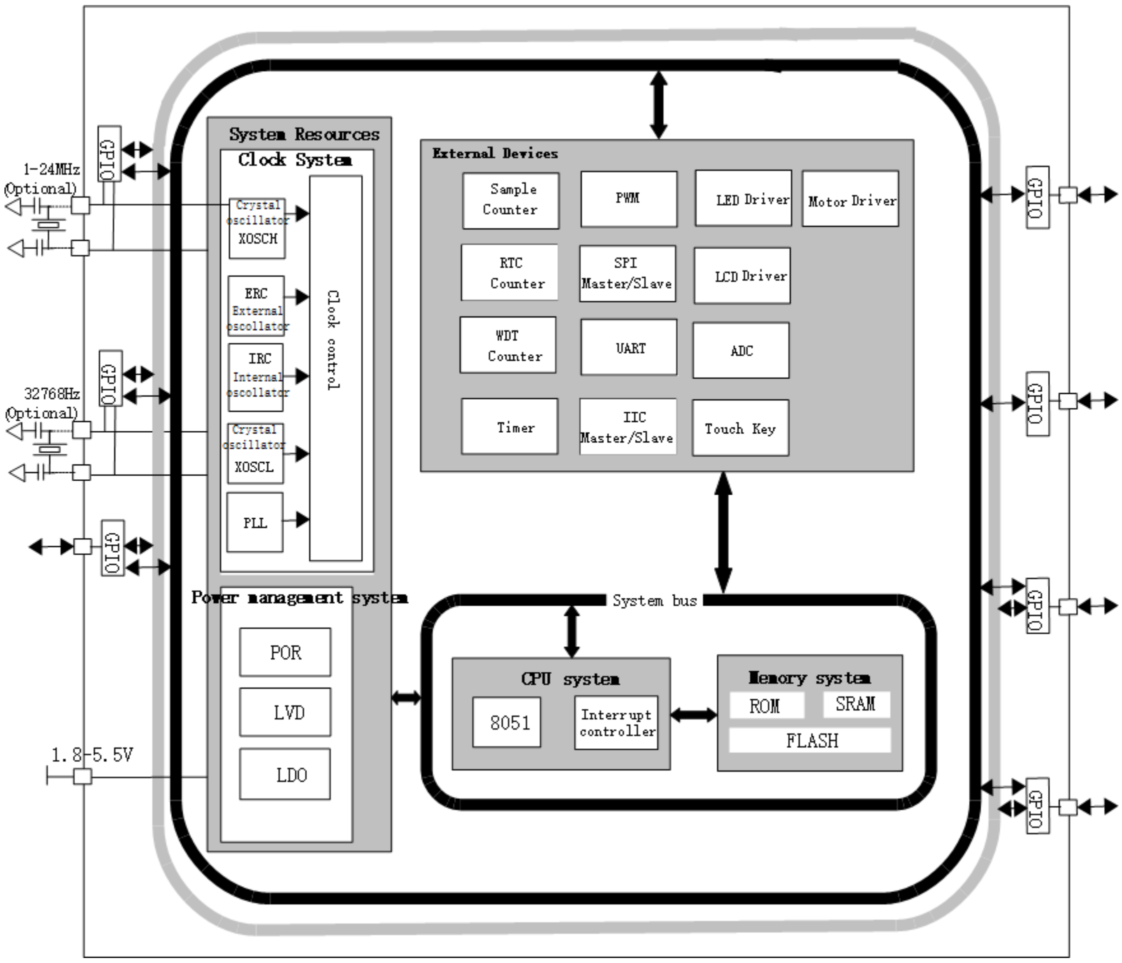
- Program Download and Simulation
    - Supports ISP and IAP
    - Supports simulation online
- Low power consumption
    - For STOP Mode, current<7uA
    - For IDLE Mode, current<12uA
    - For Low Speed Mode, current<20uA
- Package Type：LQFP64 (7 x7 mm) / LQFP48 (7 x7 mm)

3 Chip Model and Function Description

Table 3-1 CA51F2 Specific Models and Their Features

Models	Flash Storage [BYTE]	External Ram[BYTE]	External High Speed Oscillator	External Low Speed Crystal Oscillator[32.768KHz]	GPIO	UART	I^2C	SPI	16 bit PWM Channels	12 bit ADC Channels	SAMPLE Function	General operational amplifiers	Touch Key	LCD Drive [com x seg]	LED Drive[com x seg]	Brushless DC Motor Drive	Simulation On Chip	Package Type
CA51F251L2	8K	2K	--	√	46	3	√	√	5	6	——	√	15	4X25 5X24	5X24	--	√	LQFP48
CA51F252L2	16K	2K	--	√	46	3	√	√	5	6	——	√	15	4X25 5X24	5X24	--	√	LQFP48
CA51F253L2	32K	2K	--	√	46	3	√	√	5	6	——	√	15	4X25 5X24	5X24	--	√	LQFP48
CA51F251L3	8K	2K	√	√	62	3	√	√	8	8	√	√	24	8X32 7X33 6X34 5X35 4X36	8X32	√	√	LQFP64
CA51F252L3	16K	2K	√	√	62	3	√	√	8	8	√	√	24	8X32 7X33 6X34 5X35 4X36	8X32	√	√	LQFP64
CA51F253L3	32K	2K	√	√	62	3	√	√	8	8	√	√	24	8X32 7X33 6X34 5X35 4X36	8X32	√	√	LQFP64

4 Block Diagram



5 Pin Package and Description

5.1 Package Definition

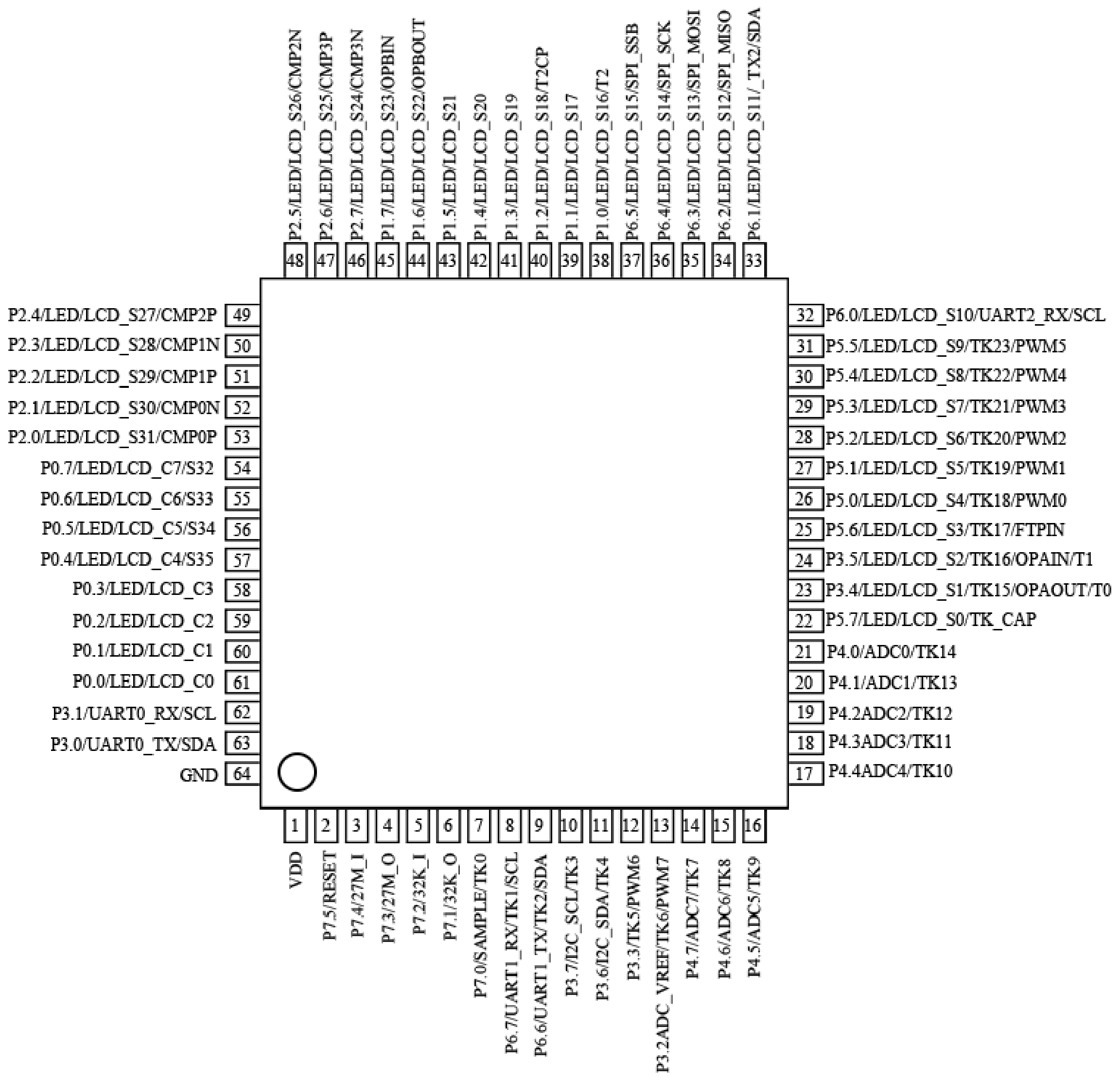

48 P2.5/LED/LCD_S26/CMP2N
47 P2.6/LED/LCD_S25/CMP3P
46 P2.7/LED/LCD_S24/CMP3N
45 P1.7/LED/LCD_S23/OPBIN
44 P1.6/LED/LCD_S22/OPBOUT
43 P1.5/LED/LCD_S21
42 P1.4/LED/LCD_S20
41 P1.3/LED/LCD_S19
40 P1.2/LED/LCD_S18/T2CP
39 P1.1/LED/LCD_S17
38 P1.0/LED/LCD_S16/T2
37 P6.5/LED/LCD_S15/SPI_SSB
36 P6.4/LED/LCD_S14/SPI_SCK
35 P6.3/LED/LCD_S13/SPI_MOSI
34 P6.2/LED/LCD_S12/SPI_MISO
33 P6.1/LED/LCD_S11/_TX2/SDA
P2.4/LED/LCD_S27/CMP2P 49
P2.3/LED/LCD_S28/CMP1N 50
P2.2/LED/LCD_S29/CMP1P 51
P2.1/LED/LCD_S30/CMP0N 52
P2.0/LED/LCD_S31/CMP0P 53
P0.7/LED/LCD_C7/S32 54
P0.6/LED/LCD_C6/S33 55
P0.5/LED/LCD_C5/S34 56
P0.4/LED/LCD_C4/S35 57
P0.3/LED/LCD_C3 58
P0.2/LED/LCD_C2 59
P0.1/LED/LCD_C1 60
P0.0/LED/LCD_C0 61
P3.1/UART0_RX/SCL 62
P3.0/UART0_TX/SDA 63
GND 64
32 P6.0/LED/LCD_S10/UART2_RX/SCL
31 P5.5/LED/LCD_S9/TK23/PWM5
30 P5.4/LED/LCD_S8/TK22/PWM4
29 P5.3/LED/LCD_S7/TK21/PWM3
28 P5.2/LED/LCD_S6/TK20/PWM2
27 P5.1/LED/LCD_S5/TK19/PWM1
26 P5.0/LED/LCD_S4/TK18/PWM0
25 P5.6/LED/LCD_S3/TK17/FTPIN
24 P3.5/LED/LCD_S2/TK16/OPAIN/T1
23 P3.4/LED/LCD_S1/TK15/OPAOUT/T0
22 P5.7/LED/LCD_S0/TK_CAP
21 P4.0/ADC0/TK14
20 P4.1/ADC1/TK13
19 P4.2ADC2/TK12
18 P4.3ADC3/TK11
17 P4.4ADC4/TK10
1 VDD
2 P7.5/RESET
3 P7.4/27M_I
4 P7.3/27M_O
5 P7.2/32K_I
6 P7.1/32K_O
7 P7.0/SAMPLE/TK0
8 P6.7/UART1_RX/TK1/SCL
9 P6.6/UART1_TX/TK2/SDA
10 P3.7/I2C_SCL/TK3
11 P3.6/I2C_SDA/TK4
12 P3.3/TK5/PWM6
13 P3.2ADC_VREF/TK6/PWM7
14 P4.7/ADC7/TK7
15 P4.6/ADC6/TK8
16 P4.5/ADC5/TK9

Figure 5-1-1 LQFP64 Package

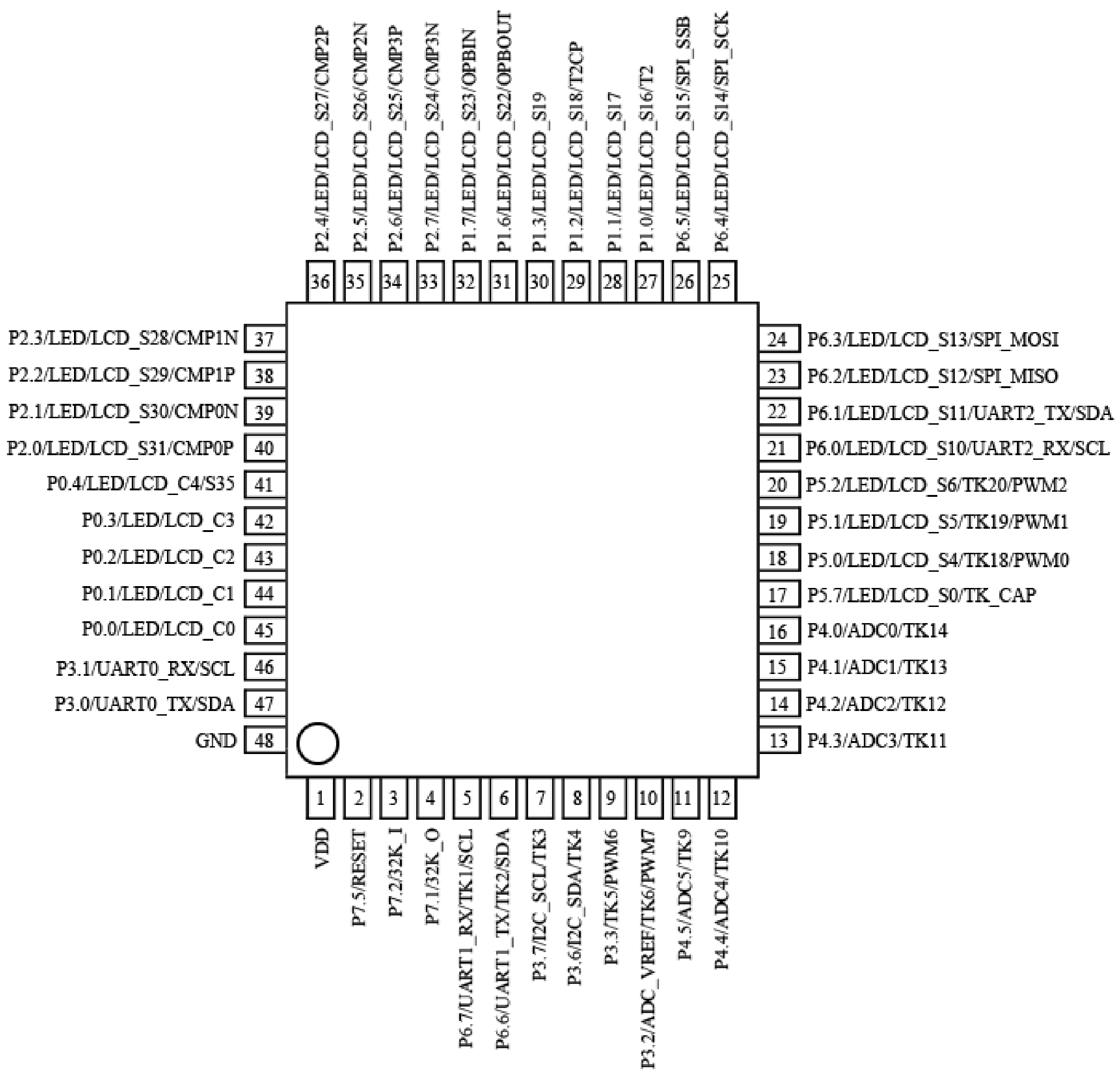

36 P2.4/LED/LCD_S27/CMP2P
35 P2.5/LED/LCD_S26/CMP2N
34 P2.6/LED/LCD_S25/CMP3P
33 P2.7/LED/LCD_S24/CMP3N
32 P1.7/LED/LCD_S23/OPBIN
31 P1.6/LED/LCD_S22/OPBOUT
30 P1.3/LED/LCD_S19
29 P1.2/LED/LCD_S18/T2CP
28 P1.1/LED/LCD_S17
27 P1.0/LED/LCD_S16/T2
26 P6.5/LED/LCD_S15/SPI_SSB
25 P6.4/LED/LCD_S14/SPI_SCK
P2.3/LED/LCD_S28/CMP1N 37
P2.2/LED/LCD_S29/CMP1P 38
P2.1/LED/LCD_S30/CMP0N 39
P2.0/LED/LCD_S31/CMP0P 40
P0.4/LED/LCD_C4/S35 41
P0.3/LED/LCD_C3 42
P0.2/LED/LCD_C2 43
P0.1/LED/LCD_C1 44
P0.0/LED/LCD_C0 45
P3.1/UART0_RX/SCL 46
P3.0/UART0_TX/SDA 47
GND 48
24 P6.3/LED/LCD_S13/SPI_MOSI
23 P6.2/LED/LCD_S12/SPI_MISO
22 P6.1/LED/LCD_S11/UART2_TX/SDA
21 P6.0/LED/LCD_S10/UART2_RX/SCL
20 P5.2/LED/LCD_S6/TK20/PWM2
19 P5.1/LED/LCD_S5/TK19/PWM1
18 P5.0/LED/LCD_S4/TK18/PWM0
17 P5.7/LED/LCD_S0/TK_CAP
16 P4.0/ADC0/TK14
15 P4.1/ADC1/TK13
14 P4.2/ADC2/TK12
13 P4.3/ADC3/TK11
1 VDD
2 P7.5/RESET
3 P7.2/32K_I
4 P7.1/32K_O
5 P6.7/UART1_RX/TK1/SCL
6 P6.6/UART1_TX/TK2/SDA
7 P3.7/I2C_SCL/TK3
8 P3.6/I2C_SDA/TK4
9 P3.3/TK5/PWM6
10 P3.2/ADC_VREF/TK6/PWM7
11 P4.5/ADC5/TK9
12 P4.4/ADC4/TK10

Figure 5-1-2 LQFP48 Package

5.2 Pin Description

Table 5-2-1 Pin Description

Pin Sequence Number		Pin Name	Pin Function	Default Function
LQFP64	LQFP48			
1	1	VDD	Power supply for the chip	Power supply for the chip
2	2	P7.5/RESET	General bi-directional I/O port Hard reset	Hard reset
3	-	P7.4/XTAL_IN_24M	General bi-directional I/O port External high speed crystal oscillator input	General bi-directional I/O port
4	-	P7.3/XTAL_OUT_24M	General bi-directional I/O port External high speed crystal oscillator output	General bi-directional I/O port
5	3	P7.2/XTAL_IN_32K	General bi-directional I/O port 32K external crystal oscillator input	32K external crystal oscillator input
6	4	P7.1/XTAL_OUT_32K	General bi-directional I/O port 32K external crystal oscillator output	32K external crystal oscillator output
7	-	P7.0/SAMPLE/TK[0]	General bi-directional I/O port Sample signal digital input Touch key analog channel input	General bi-directional I/O port
8	5	P6.7/UART[1]_RX/TK[1]/SCL	General bi-directional I/O port UART1 RX port I2C clock transmission port Touch key analog channel input	I2C clock transmission port
9	6	P6.6/UART[1]_TX/TK[2]/SDA	General bi-directional I/O port UART1 TX port I2C data transmission port Touch key analog channel input	I2C data transmission port

10	7	P3.7/I2C_SCL/TK[3]	General bi-directional I/O port I2C clock transmission port Touch key analog channel input	I2C clock transmission port
11	8	P3.6/I2C_SDA/TK[4]	General bi-directional I/O port I2C data transmission port Touch key analog channel input	I2C data transmission port
12	9	P3.3/TK[5]/PWM[6]	General bi-directional I/O port Touch key analog channel input PWM digital output	General bi-directional I/O port
13	10	P3.2/TK[6]/PWM[7]/ADC_REF	General bi-directional I/O port Touch key analog channel input PWM digital output ADC reference voltage input	General bi-directional I/O port
14	-	P4.7/ADC_CH[7]/TK[7]	General bi-directional I/O port ADC analog channel input Touch key analog channel input	General bi-directional I/O port
15	-	P4.6/ADC_CH[6]/TK[8]	General bi-directional I/O port ADC analog channel input Touch key analog channel input	General bi-directional I/O port
16	11	P4.5/ADC_CH[5]/TK[9]	General bi-directional I/O port ADC analog channel input Touch key analog channel input	General bi-directional I/O port
17	12	P4.4/ADC_CH[4]/TK[10]	General bi-directional I/O port ADC analog channel input Touch key analog channel input	General bi-directional I/O port
18	13	P4.3/ADC_CH[3]/TK[11]	General bi-directional I/O port ADC analog channel input Touch key analog channel input	General bi-directional I/O port
19	14	P4.2/ADC_CH[2]/TK[12]	General bi-directional I/O port ADC analog channel input Touch key analog channel input	General bi-directional I/O port
20	15	P4.1/ADC_CH[1]/TK[13]	General bi-directional I/O port ADC analog channel input Touch key analog channel input	General bi-directional I/O port

21	16	P4.0/ADC_CH[0]/TK[14]	General bi-directional I/O port ADC analog channel input Touch key analog channel input	General bi-directional I/O port
22	17	P5.7/LED_SEG[0]/LCD_SEG[0]/TK_CAP	General bi-directional I/O port LED SEG output LCD SEG output Touch key analog channel input Touch key external capacitor	General bi-directional I/O port
23	-	P3.4/T0/LED_SEG[1]/SEG[1]/TK[15]/OPAOUT	General bi-directional I/O port Timer T0 input LED SEG output LCD SEG output Touch key analog channel input Amplifier A output	General bi-directional I/O port
24	-	P3.5/T1/LED_SEG[2]/SEG[2]/TK[16]/OPAIN	General bi-directional I/O port Timer T1 input LED SEG l output LCD SEG output Touch key analog channel input Amplifier A input	General bi-directional I/O port
25	-	P5.6/LED_SEG[3]/SEG[3]/TK[17]/FTPIN	General bi-directional I/O port LED SEG output LCD SEG output Touch key analog channel input Motor error detection digital input	General bi-directional I/O port
26	18	P5.0/LED_SEG[4]/SEG[4]/TK[18]/PWM[0]	General bi-directional I/O port LED SEG output LCD SEG output Touch key analog channel input PWM digital output	General bi-directional I/O port
27	19	P5.1/LED_SEG[5]/SEG[5]/TK[19]/PWM[1]	General bi-directional I/O port LED SEG output LCD SEG output Touch key analog channel input PWM digital output	General bi-directional I/O port

28	20	P5.2/LED_SEG[6]/SEG[6]/TK[20]/PWM[2]	General bi-directional I/O port LED SEG output LCD SEG output Touch key analog channel input PWM digital output	General bi-directional I/O port
29	-	P5.3/LED_SEG[7]/SEG[7]/TK[21]/PWM[3]	General bi-directional I/O port LED SEG output LCD SEG output Touch key analog channel input PWM digital output	General bi-directional I/O port
30	-	P5.4/LED_SEG[8]/SEG[8]/TK[22]/PWM[4]	General bi-directional I/O port LED SEG output LCD SEG output Touch key analog channel input PWM digital output	General bi-directional I/O port
31	-	P5.5/LED_SEG[9]/SEG[9]/TK[23]/PWM[5]	General bi-directional I/O port LED SEG output LCD SEG output Touch key analog channel input PWM digital output	General bi-directional I/O port
32	21	P6.0/LED_SEG[10]/SEG[10]/UART[2]_RX/SCL	General bi-directional I/O port LED SEG output LCD SEG output UART [2]RX port IIC_SCL port	I2C_SCL port
33	22	P6.1/LED_SEG[11]/SEG[11]/UART[2]_TX/SDA	General bi-directional I/O port LED SEG output LCD SEG output UART [2]TX port IIC_SDA port	I2C_SDA port
34	23	P6.2/LED_SEG[12]/SEG[12]/SPI_MISO	General bi-directional I/O port LED SEG output LCD SEG output SPI_MISO port	General bi-directional I/O port
35	24	P6.3/LED_SEG[13]/SEG[13]/SPI_MOSI	General bi-directional I/O port LED SEG output LCD SEG output SPI _MOSI port	General bi-directional I/O port

36	25	P6.4/LED_SEG[14]/SEG[14]/SPI_SCK	General bi-directional I/O port LED output LCD SEG output SPI _SCK port	General bi-directional I/O port
37	26	P6.5/LED_SEG[15]/SEG[15]/SPI_SSB	General bi-directional I/O port LED SEG output LCD SEG output SPI _SSB port	General bi-directional I/O port
38	27	P1.0/T2/LED_SEG[16]/SEG[16]	General bi-directional I/O port TimerT2 input LED SEG output LCD SEG output	General bi-directional I/O port
39	28	P1.1/T2EX/LED_SEG[17]/SEG[17]	General bi-directional I/O port Timer T2EX input LED SEG output LCD SEG output	General bi-directional I/O port
40	29	P1.2/LED_SEG[18]/SEG[18]/T2CP	General bi-directional I/O port LED SEG output LCD SEG output Timer T2CP input	General bi-directional I/O port
41	30	P1.3/LED_SEG[19]/SEG[19]	General bi-directional I/O port LED SEG output LCD SEG output	General bi-directional I/O port
42	-	P1.4/LED_SEG[20]/SEG[20]	General bi-directional I/O port LED SEG output LCD SEG output	General bi-directional I/O port
43	-	P1.5/LED_SEG[21]/SEG[21]	General bi-directional I/O port LED SEG output LCD SEG output	General bi-directional I/O port
44	31	P1.6/LED_SEG[22]/SEG[22]/OPBOUT	General bi-directional I/O port LED SEG output LCD SEG output Amplifier B output	General bi-directional I/O port
45	32	P1.7/LED_SEG[23]/SEG[23]/OPBIN	General bi-directional I/O port LED SEG output LCD SEG Output Amplifier B input	General bi-directional I/O port

46	33	P2.7/LED_SEG[24]/SEG[24]/CMP3N	General bi-directional I/O port LED SEG output LCD SEG output Comparator Amplifier-3N input	General bi-directional I/O port
47	34	P2.6/LED_SEG[25]/SEG[25]/CMP3P	General bi-directional I/O port LED SEG output LCD SEG output Comparator Amplifier-3P input	General bi-directional I/O port
48	35	P2.5/LED_SEG[26]/SEG[26]/CMP2N	General bi-directional I/O port LED SEG output LCD SEG output Comparator Amplifier-2N input	General bi-directional I/O port
49	36	P2.4/LED_SEG[27]/SEG[27]/CMP2P	General bi-directional I/O port LED SEG output LCD SEG output Comparator Amplifier-2P input	General bi-directional I/O port
50	37	P2.3/LED_SEG[28]/SEG[28]/CMP1N	General bi-directional I/O port LED SEG output LCD SEG output Comparator Amplifier-1N input	General bi-directional I/O port
51	38	P2.2/LED_SEG[29]/SEG[29]/CMP1P	General bi-directional I/O port LED SEG output LCD SEG output ComparatorAmplifier-1P input	General bi-directional I/O port
52	39	P2.1/LED_SEG[30]/SEG[30]/CMP0N	General bi-directional I/O port LED SEG output LCD SEG output Comparator Amplifier-0N input	General bi-directional I/O port
53	40	P2.0/LED_SEG[31]/SEG[31]/CMP0P	General bi-directional I/O port LED SEG output LCD SEG output Comparator Amplifier-0P input	General bi-directional I/O port
54	-	P0.7/LED_COM[7]/COM[7]/SEG[32]	General bi-directional I/O port LED COM output LCD COM output LCD SEG output	General bi-directional I/O port
55	-	P0.6/LED_COM[6]/COM[6]/SEG[33]	General bi-directional I/O port LED COM output LCD COM output LCD SEG output	General bi-directional I/O port

56	-	P0.5/LED_COM[5]/COM[5]/SEG[34]	General bi-directional I/O port LED COM output LCD COM output LCD SEG output	General bi-directional I/O port
57	41	P0.4/LED_COM[4]/COM[4]/SEG[35]	General bi-directional I/O port LED COM output LCD COM output LCD SEG output	General bi-directional I/O port
58	42	P0.3/LED_COM[3]/COM[3]	General bi-directional I/O port LED COM output LCD COM output	General bi-directional I/O port
59	43	P0.2/LED_COM[2]/COM[2]	General bi-directional I/O port LED COM output LCD COM output	General bi-directional I/O port
60	44	P0.1/LED_COM[1]/COM[1]	General bi-directional I/O port LED COM output LCD COM output	General bi-directional I/O port
61	45	P0.0/LED_COM[0]/COM[0]	General bi-directional I/O port LED COM output LCD COM output	General bi-directional I/O port
62	46	P3.1/UART[0]_RX/SCL	General bi-directional I/O port UART 0 RX port I2C clock transmission port	I2C clock transmission port
63	47	P3.0/UART[0]_TX/SDA	General bi-directional I/O port UART 0 TX port I2C data transmission port	I2C data transmission port
64	48	GND	Ground	Ground

Note : For signal pin’s alternate function settings, please refer to Table 15-2-9 and Table 15-2-10

6 Central Processing Unit (CPU)

6.1 CPU Introduction

The core of CA51F2 Series is monocyclic 8051 CPU and make it fully compatible with original MCS-51 instruction set. A monocyclic 8051 CPU usually operates 10 times faster than standard 8051 one due to its pipeline structure.

The features of this CPU are：

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

6.2 Register Description

Program Counter (PC)

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

Accumulator(ACC)

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

General Register B

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

Stack Pointer (SP)

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

Data Pointer (DPTR)

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

Program Status Word (PSW)

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

Table 6-2-1 Accumulator ACC

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

Table 6-2-2 General Register B

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

Table 6-2-3 Stack Pointer SP

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

Table 6-2-4 Data Pointer DP0L

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

Table 6-2-5 Data Pointer DP0H

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

Table 6-2-6 Data Pointer DP1L

84H	7	6	5	4	3	2	1	0

DP1L	DP1L[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Table 6-2-7 Data Pointer DP1H

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

Table 6-2-8 Program Status Word PSW

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

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

Table 6-2-9 Register SPMAX

8100H	7	6	5	4	3	2	1	0
SPMAX	SPMAX[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit symbol	Description
7~0	SPMAX	SPMAX is used to record the maximum value of SP. Users can check this register using software to decide whether there is a risk that the stack may overflow

7 Memory Architecture

7.1 Random Access Memory(RAM)

CA51F2 series offers both internal RAM(256 bytes) and external RAM(2K bytes) for the users and the corresponding address are shown as follows:

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

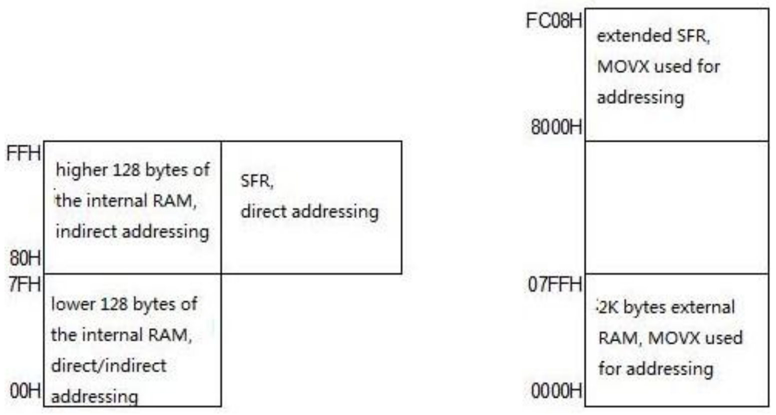

FFH
higher 128 bytes of the internal RAM, indirect addressing
SFR, direct addressing
80H
7FH
lower 128 bytes of the internal RAM, direct/indirect addressing
00H
FC08H
extended SFR, MOVX used for addressing
8000H
07FFH
2K bytes external RAM, MOVX used for addressing
0000H

Figure 7-1-1 RAM Architecture

7.2 Special Function Register(SFR)

The SFR architecture of CA51F2 series is compatible with traditional 8051 chip. SFR and the higher 128 bytes of the internal RAM both use the address 80H ~ FFH that only supports direct addressing, SFR mapping is shown in Table 7-2-1.

Table 7-2-1 Special Function Register Mapping Table

	Bit addressable	Not bit addressable						
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8H	EXIP	EPIE	EPIF	EPCON	IDLSTL	IDLSTH	STPSTL	STPSTH
F0H	B	RTCON	RTCS	RTCM	RTCH	RTCDL	RTCDH	INDEX
E8H	EXIE	RTCSS	RTAS	RTAM	RTAH	RTMSS	RTCIF	LVDCON
E0H	ACC	LXCON	LXCFG	LXDAT	LXDIVL	LXDIVH	MDUCON	MDUDAT
D8H	P5	P7	PWMEN	PWMUPD	PWMCMX	PWMCON	PWMCFG	PWMDIVL

D0H	PSW	PWMDIVH	PWMDUTL	PWMDUTH	PWMAIF	PWMBIF	PWMCIF	PWMDIF
C8H	T2CON	T2MOD	T2CL	T2CH	TL2	TH2	TKMSL	TKMSH
C0H	P4	TKCON	TKCFG	TKMTS	TKCHS	ATKCL	ATKCH	TKIF
B8H	IP	ADCON	ADCFGL	ADCFGH	ADCDL	ADCDH	CKMON	CKMIF
B0H	P3	I2CCON	I2CADR	I2CADM	I2CCCR	I2CDAT	I2CSTA	I2CFLG
A8H	IE	P6	WDCON	WDFLG	WDVTHL	WDVTHH	PLLCON	HVTH
A0H	P2	S2CON	S2BUF	S2RELL	S2RELH	SPCON	SPDAT	SPSTA
98H	S0CON	S0BUF	S1CON	S1BUF	S1RELL	S1RELH	RCMSHL	RCMSHH
90H	P1	RCCON	VCKDL	VCKDH	RCTAGL	RCTAGH	RCMSLL	RCMSLH
88H	TCON	TMOD	TL0	TL1	TH0	TH1	IT1CON	IT0CON
80H	P0	SP	DP0L	DP0H	DP1L	DP1H	PWCON	PCON

Due to limited SFR address space, CA51F2 series also added extended special function register in external RAM address space. The mapping is shown as follows。

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

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
8000H	P00F	P01F	P02F	P03F	P04F	P05F	P06F	P07F
8008H	P10F	P11F	P12F	P13F	P14F	P15F	P16F	P17F
8010H	P20F	P21F	P22F	P23F	P24F	P25F	P26F	P27F
8018H	P30F	P31F	P32F	P33F	P34F	P35F	P36F	P37F
8020H	P40F	P41F	P42F	P43F	P44F	P45F	P46F	P47F
8028H	P50F	P51F	P52F	P53F	P54F	P55F	P56F	P57F
8030H	P60F	P61F	P62F	P63F	P64F	P65F	P66F	P67F
8038H	P70F	P71F	P72F	P73F	P74F	P75F		
8040H	OPACON	OPBCON	-	-	-	-	-	-
8048H	CP0CON	CP1CON	CP2CON	CP3CON	CPCKS	CPSTA	CPVTC	
8050H	FT0SL	FT0SH	FT1SL	FT1SH	FT2SL	FT2SH	FT3SL	FT3SH
8058H	CTMCON	CTMVTHL	CTMVTHH	CTMCNTL	CTMCNTH	-	-	-
8060H	MOTCON	MOTCFG	MTGCON	MHLCON	MFPCON	MOTCMD	MTGDL	MOTIF
8068H	HDCT0	HDCT1	HDCT2	HDCT3	HDCT4	HDCT5	HDCT6	HDCT7
8070H	HDCT8	HDCT9	HDCT10	HDCT11	MOTPLC	-	-	-
8078H	SMCON	SMSTA	SMDIV	SMDATL	SMDATH	SMVTHL	SMVTHH	-
8080H	CKCON	CKSEL	CKDIV	IHCFGL	IHCFGH	ILCFGL	ILCFGH	TFCFG
8088H	ADCALL	ADCALH	ACPDLL	ACPDLH	ACPDHL	ACPDHH	-	-
8090H	TKMAXF	TLMINF	ATKNL	ATKNH	-	-	-	-
8100H	SPMAX	I2CIOS	-	-	-	-	-	-
8108H	TLCON	TLFLG	TLCKS	TLCNTKL	TLCNTKH	TLCNTLL	TLCNTLH	TLDIV
8110H	TLCOMS							LXCAD
8118H							FTCTL	TPCTL
8120H	P00C	P01C	P02C	P03C	P04C	P05C	P06C	P07C
8128H	P10C	P11C	P12C	P13C	P14C	P15C	P16C	P17C

8130H	P20C	P21C	P22C	P23C	P24C	P25C	P26C	P27C
8138H	P30C	P31C	P32C	P33C	P34C	P35C	P36C	P37C
8140H	P40C	P41C	P42C	P43C	P44C	P45C	P46C	P47C
8148H	P50C	P51C	P52C	P53C	P54C	P55C	P56C	P57C
8150H	P60C	P61C	P62C	P63C	P64C	P65C	P66C	P67C
8158H	P70C	P71C	P72C	P73C	P74C	P75C		
FC00H	MECON	FSCMD	FSDAT	LOCK	PADRD	PTSL	PTSH	REPSET

7.3 Flash

7.3.1 Function Introduction

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

7.3.2 Flash Architecture

- Flash consists of several sectors which are the smallest units for erasure. Each sector is 128 bytes.
- Flash can be divided into DATA area and PROGRAM area and the division unit is 256 bytes. PROGRAM area is used to store use’s program and DATA area is used to store data that needs to be protected during power off period.

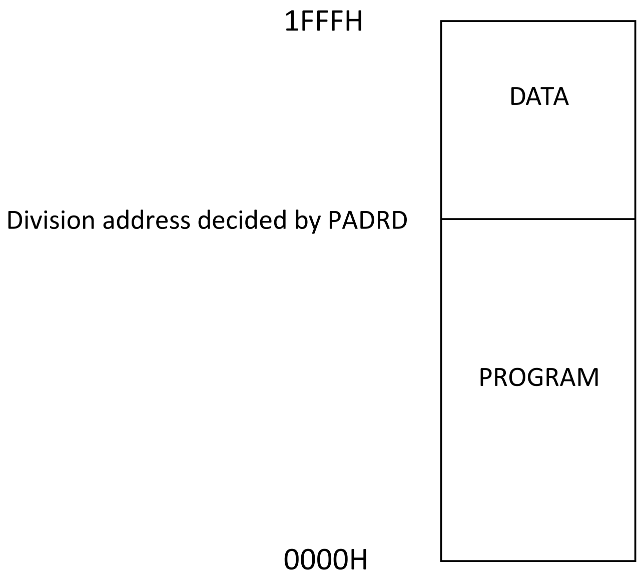


Figure 7-3-2 8K Flash Memory Structure

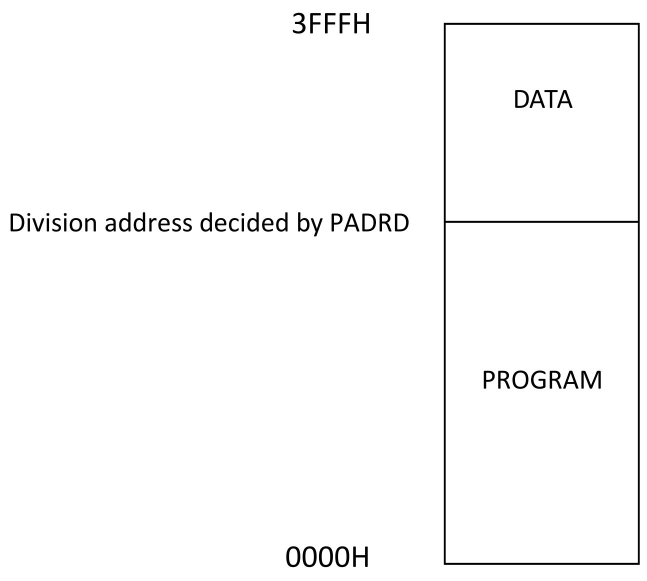


Figure 7-3-3 16K Flash Memory Structure

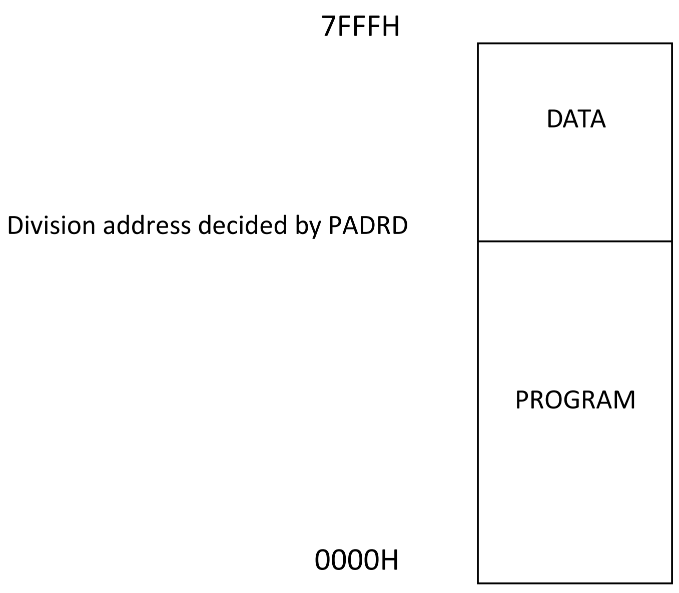


Figure 7-3-4 32K Flash Memory Structure

7.3.3 Flash Description

Table 7-3-3-1 Register MECON

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

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

Table 7-3-3-2 Register FSCMD

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

Bit number	Bit symbol	Description
7~4	-	-
3~0	CMD	Command register 0000：No operations 0100：Erase the whole Flash 0001：Read Flash DATA area 0010：Write Flash DATAarea 0011：Erase sectors of the Flash DATA area 1011：Erase blocks of the Flash DATA area (512 bytes for each block) 0101：Read Flash PROGRAM area

		0110： Write Flash PROGRAM area 0111： Erase sectors of the Flash PROGRAM area 1111： Erase blocks of the Flash PROGRAM area (512 bytes for each block) 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					FLKF	PLKF	DLKF	ILKF
W	LOCK[7:0]							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit symbol	Description
Write		
7~0	LOCK	28H： Unlock Flash programmable area 29H： Unlock Flash PROGRAM area 2AH： Unlock Flash DATA area AAH： Lock Flash, R/W forbidden
Read		
7	-	
6	-	-
5	-	
4	-	
3	FLKF	Programmable area unlocked flag, 1 indicates unlocked
2	PLKF	PROGRAM area unlocked flag, 1 indicates unlocked
1	DLKF	DATA area unlocked flag, 1 indicates unlocked
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						
7~0	PADRD	PROGRAM and DATA area division configuration register the unit for division is 256 bytes and when PADRD>0: The address space for PROGRAM area : 0 ~ (PADRD×256 - 1), The address space for DATA area : (PADRD×256) ~ 1FFFH/3FFFH/7FFFH. Note： 1. PADRD=0 indicates the whole Flash is DATA area 2. 1FFFH/3FFFH/7FFFH is the maximum address for 8K/16K/32K Flash 3. The maximum value for PADRD is 20H/40H/80H corresponding to 8K/16K/32K Flash respectively. PADRD can not be set to any values greater than the maximum.						

Table 7-3-3-6 Register PTS

FC05H	7	6	5	4	3	2	1	0
PTSL	PTS[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
FC06H	7	6	5	4	3	2	1	0
PTSH		PTS[14:8]						
R/W	-	R/W						
Initial Value		0	0	0	0	0	0	0
Bit number	Bit symbol	Description						
15	-							
14~0	PTS	target address pointer register						

7.3.4 Flash Control Example

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

For instance, if the user wants to divide a 32K Flash (1204 bytes DATA area and the remains for PROGRAM area), the program may like this:

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

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

◆ **Sector erasure of DATA area**

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

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

Note： sector number n=0、1、2……。

◆ **Write data into DATA area**

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

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

Note：

1. *When data is written continuously, only original address has be set. PTS will increase automatically after writing FSDAT each time.*
2. *For DATA area R/W, only the logical address of the DATA area which starts from 0 needs to be set, instead of*

the physical address.

◆ **Read data from DATA area**

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

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

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

◆ **Sector erasure of PROGRAM area**

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

```
------------------------------------------------------------------------------------------
FSCMD = 0;          //set CMD = 0
LOCK = 0x29;        //unlock PROGRAM area
PTSH = (unsigned char)((n*0x80)>>8); //set the higher bytes of the sector’s address
PTSL = (unsigned char)(n*0x80);          //set the lower bytes of the sector’s address
FSCMD = 7; //set CLEAR command
LOCK = 0xAA; //lock FLASH
------------------------------------------------------------------------------------------
```

Note： Sectors number n=0、1、2……。

◆ **Write data into PROGRAM area**

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

```
------------------------------------------------------------------------------------------
unsigned char i；
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 = 6; //set WRITE command
for(i=0;i<100;i++)
{
        FSDAT = 0xAA;      //write data continuously
}
```

```
LOCK = 0xAA;        //lock FLASH
-----------------------------------------------------------------------------------------
```

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

- **Read data from PROGRAM area**

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

```
-----------------------------------------------------------------------------------------
unsigned char i, pBuf ;
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<100;i++)
{
        *pBuf++ = FSDAT ;//read data continuously
}
LOCK = 0xAA;        //lock FLASH
-----------------------------------------------------------------------------------------
```

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

7.4 External RAM Mapped to Program Area

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

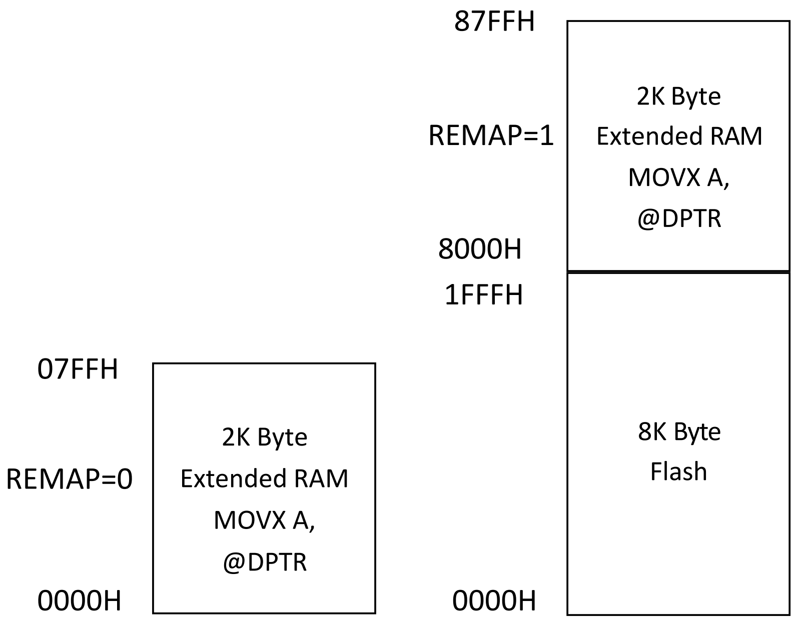

87FFH
REMAP=1
2K Byte Extended RAM MOVX A, @DPTR
8000H
1FFFH
8K Byte Flash
0000H
07FFH
REMAP=0
2K Byte Extended RAM MOVX A, @DPTR
0000H

Figure 7-4-1 XRAM Address mapping

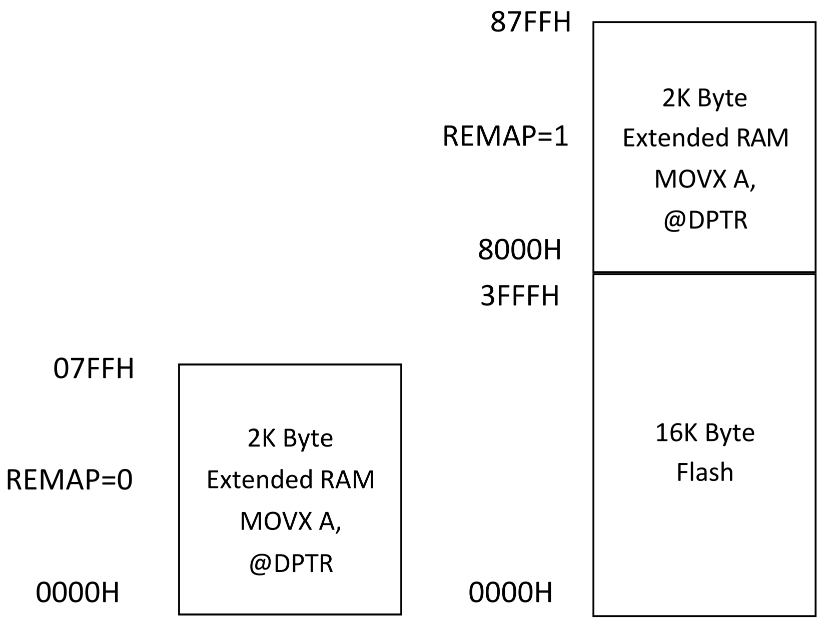

87FFH
REMAP=1
8000H
3FFFH
0000H
2K Byte Extended RAM MOVX A, @DPTR
16K Byte Flash
07FFH
REMAP=0
0000H
2K Byte Extended RAM MOVX A, @DPTR

Figure 7-4-2 XRAM Address mapping

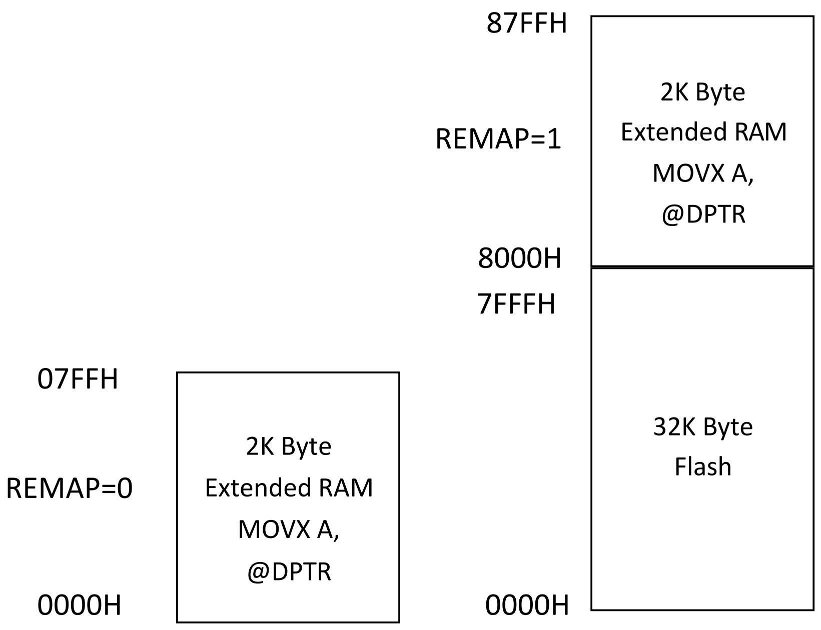

87FFH
REMAP=1
8000H
7FFFH
0000H
2K Byte Extended RAM MOVX A, @DPTR
32K Byte Flash
07FFH
REMAP=0
0000H
2K Byte Extended RAM MOVX A, @DPTR

Figure 7-4-3 XRAM Address mapping

8 Interrupt System

8.1 Function Introduction

CA51F2 Series include an enhanced interrupt control system with 15 interrupt entries. For each interrupt entry, there are several interrupt sources with 2 level interrupt priorities for each source. Each interrupt source has its independent interrupt vector, priority setting, interrupt enable control and interrupt flag. CPU enters corresponding Interrupt Service Routine after responding to the interrupt. It will then returns to the former status after receiving RETI. If there are multiple valid sources requesting interrupts, CPU will respond sequentially according to the interrupt priority set before. If the sources share the same priority, CPU will respond according to their natural priority (from the smallest address to largest address of the interrupt entries).

8.2 Interrupt Logic

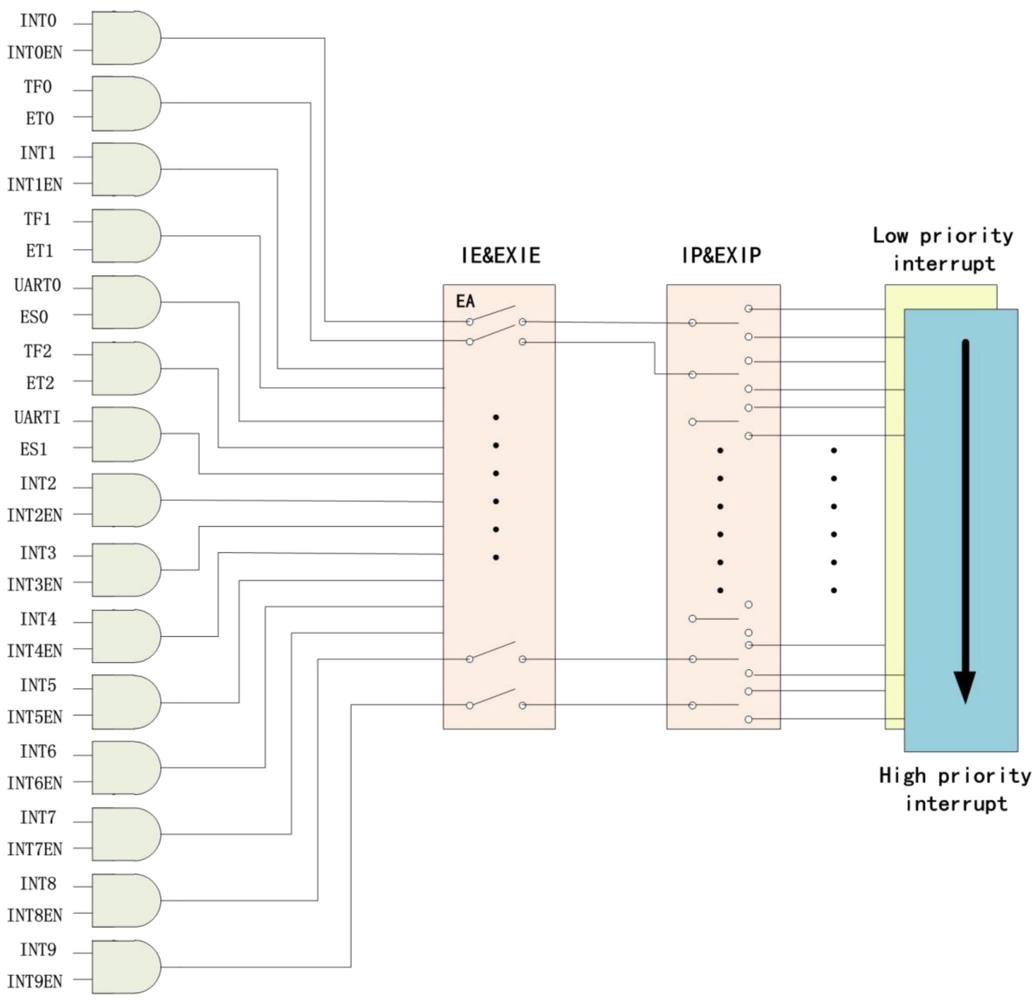


Figure 8-2-1 Interrupt Logic

8.3 Interrupt Vector Table

Table 8-3-1 Interrupt Vector Table

Interrupt	Interrupt source	Vector	Default Priority
INT0	INT0	03H	0
TF0	Timer 0	0BH	1
INT1	INT1	13H	2
TF1	Timer 1	1BH	3
UART0	UART0	23H	4
TF2	Timer 2	2BH	5
UART1	UART1	33H	6
INT2	ADC/External Interrupt 2	3BH	7
INT3	UART2/TK/External Interrupt 3	43H	8
INT4	LVD/External Interrupt 4	4BH	9
INT5	SPI/Clock Monitor/External Interrupt 5	53H	10
INT6	I2C/Analog Comparator/External Interrupt 6	5BH	11
INT7	WDT/MOTOR/External Interrupt 7	63H	12
INT8	RTC/Capture Counter/External Interrupt 8	6BH	13
INT9	SAMPLE/PWM/External Interrupt 9	73H	14

8.4 Interrupt Control Register

Table 8-4-1 Register IE

A8H	7	6	5	4	3	2	1	0
IE	EA	ES1	ET2	ES0	ET1	INT1EN	ET0	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 Global Interrupt 1：enable Global Interrupt
6	ES1	UART1 Interrupt enable control 0：disable UART1 Interrupt 1：enable UART1 Interrupt
5	ET2	Timer 2 Interrupt enable control

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

Table 8-4-2 Register EXIE

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

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

1	INT3EN	Interrupt 3 enable control(Interrupt 3 is used for UART2/TK/External Interrupt 3) 0：Disable 1：Enable
0	INT2EN	Interrupt 2 enable control(Interrupt 2 is used for ADC/External Interrupt 2) 0：Disable 1：Enable

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

Table 8-4-3 Register IP

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

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

Table 8-4-4 Register EXIP

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

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

8.5 External Interrupt

8.5.1 External Interrupt Introduction

INT0 and INT1 add the function of selecting any input port as the interrupt trigger source on the basis of the standard 8051. The system also extends 8 interrupt entries INT2~INT9 as external interrupts, each interrupt entry can also select any input port as the interrupt trigger source, and the extended external interrupts can be individually set to trigger interrupts on rising or falling edges. Each external interrupt can be used to wake up in STOP mode. EPIF is the external interrupt status register of INT2~INT9. Each configuration register EPCON0~EPCON7 corresponding to INT2~INT9 can be accessed by configuring index register INDEX as 0~7.

Note: INT0 and INT1 can be selected to be triggered by rising or falling edge, the selection bits are IT0 and IT1 respectively, see the description of register TCON for details.

8.5.2 External Interrupt Register

Table 8-5-2-1 Register IT0CON

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

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

Table 8-5-2-2 Register IT1CON

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

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

Table 8-5-2-3 Register EPIE

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

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

Table 8-5-2-4 Register EPIF

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

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

Table 8-5-2-5 Register EPCON

FBH	7	6	5	4	3	2	1	0
EPCON	EPPL	-	EPPS[5:0]					
R/W	R/W	-	R/W					
Initial Value	0	-	0	0	0	0	0	0
Note：EPCON is a register with index, INDEX=0~7 indicates EPCON0~EPCON7 respectively								

Bit number	Bit symbol	Description
7	EPPL	External Interrupt Trigger Edge Selection

		0：Rising edge 1：Falling edge
6	-	-
5~0	EPPS[5:0]	Interrupt Pin selection The table for pin numbers and corresponding pins please refer to Table 8-4-7

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

Pin name	Number	Pin name	Number
P00	0	P40	32
P01	1	P41	33
P02	2	P42	34
P03	3	P43	35
P04	4	P44	36
P05	5	P45	37
P06	6	P46	38
P07	7	P47	39
P10	8	P50	40
P11	9	P51	41
P12	10	P52	42
P13	11	P53	43
P14	12	P54	44
P15	13	P55	45
P16	14	P56	46
P17	15	P57	47
P20	16	P60	48
P21	17	P61	49
P22	18	P62	50
P23	19	P63	51
P24	20	P64	52
P25	21	P65	53
P26	22	P66	54
P27	23	P67	55
P30	24	P70	56
P31	25	P71	57
P32	26	P72	58
P33	27	P73	59
P34	28	P74	60
P35	29	P75	61
P36	30		
P37	31		

8.5.3 External Interrupt Control Method and Examples

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

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

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

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

```
--------------------------------------------------------------------------------
void INT1_init(void)
{
        P20F = 1;         ///IT1CON = 16;      //set P20 as the pin for Interrupt 0 (16 is the corresponding index number  for
P20)
        EX1 = 1;          //enable INT1 interrupt
        IE1 = 1;          //enable External Interrupt 1
        IT1 = 1;          //set falling edge trigger
        PX1 = 1;               //set INT0 with high priority
        EA = 1;                  //enable Global Interrupt
}
void INT1_ISR (void) interrupt 2
{
        //External Interrupt 1Interrupt Service Routine
}

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

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

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

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

9 Clock System

9.1 Clock System Introduction

The clock system includes the system clock generation, frequency division and assignment. CA51F2 Series has several clock sources as follows：

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

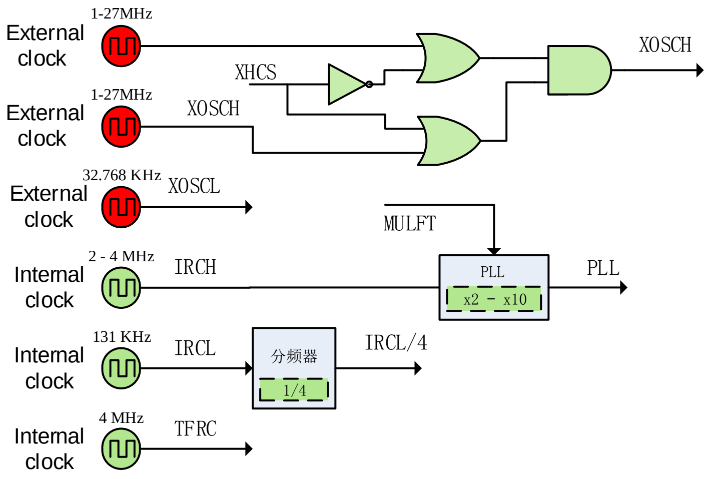


Figure 9-1-1 Clock Sources

Users can control the clock sources independently. They can disable or enable any of the clock sources in order to manage the power consumption flexibly.
All the clock sources can be set as system alarm clock and assigned to various peripherals as their clock sources. For more information you may refer to the Peripherals part.

9.1.1 Clock Special Name Definition

Symbol	Description	Symbol	Description
IRCH	2 - 4 MHz Internal RC Oscillator	ERC	External RC Oscillator
IRCL	131 KHz Internal RC Oscillator	TFRC	4MHz Internal RC Oscillator
XOSCL	32.768 KHz External Crystal Resonator	PLL	2~10 times PLL
XOSCH	1 - 27 MHz External RC Crystal Oscillator		

9.1.2 2 - 4 MHz Internal RC Oscillator(IRCH)

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

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

9.1.3 32.768 KHz External Crystal Resonator(XOSCL)

XOSCL is mainly used as the clock source of RTC for real-time timing to realize the clock function of the product. XOSCL is mainly used in low-power applications when set as the system clock, the lowest power consumption can be less than 13uA.XOSCL is turned on or off by the XLCKE bit of register CKCON. It should be noted that the start-up time of XOSCL is long, it takes about 1 second to reach It is necessary to wait for the XOSCL clock to stabilize before it can be used in applications.

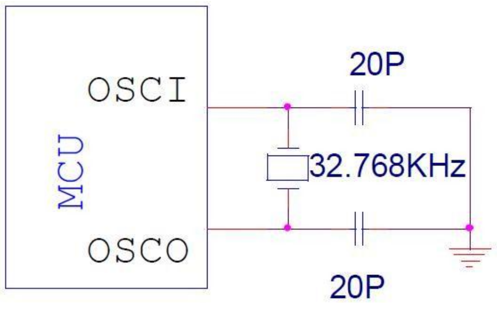


Figure 9-1-3-1 XOSCL Typical Circuit

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

9.1.4 131 KHz Internal RC Oscillator(IRCL)

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

9.1.5 4 MHz RC Internal Oscillator(TFRC)

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

9.1.6 PLL

2~10 times internal PLL takes IRCH as its reference clock and the maximum frequency of PLL is up to 40MHz. The chip can operate with high speed even without external high speed crystal oscillator. The register PLLCON can be used to enable/disable PLL and set its ratio. When PLL is enabled, users still have to wait until it is stable and then the PLL can be set as system clock or clock for other peripherals. The bit PLSTA of register PLLCON indicates whether PLL clock is stable or not. It usually takes about 50us for PLL to become stable.

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

9.1.7 External High Speed Crystal Resonator(XOSCH)and External RC Oscillator(ERC)

XOSCH and ERC share the same connecting pin, so only one of them can be used at one time and the selection is controlled by the XHCS of register CKCON. XOSCH can be enabled/disabled by setting the XHCKE of register CKCON. In addition, the flag XHSTA shows whether the XOSCH is stable now.

***Note**: For hardware design the crystal oscillator 's load capacitor should be as close as possible to the GND pin.*

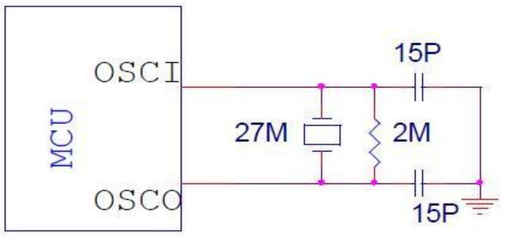


Figure 9-1-7-1 External High Speed Crystal Oscillator Typical Circuit

9.2 Clock Control Register Description

Table 9-2-1 Register CKCON

8080H	7	6	5	4	3	2	1	0
CKCON	ILCKE	IHCKE	TFCKE	XHCS	XLCKE	XLSTA	XHCKE	XHSTA
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	ILCKE	IRCL enable control 1：enable 0：disable *Note：* *When it is 1, the clock is enabled; when it is 0, if the system or other modules selected this clock source, the clock is still enabled.*
6	IHCKE	IRCH enable control 1：enable 0：disable *Note：* *When it is 1, the clock is enabled; when it is 0, if the system or other modules selected this clock source, the clock is still enabled.*
5	TFCKE	TFRC enable control 1：enable 0：disable *Note：* *When it is 1, the clock is enabled; when it is 0, if the system or other modules selected this clock source, the clock is still enabled.*
4	XHCS	XOSCH clock selection 0：the clock source for XOSCH is external high speed crystal oscillator 1：the clock source for XOSCH is external RC oscillator
3	XLCKE	XOSCL enable control 1：enable 0：disable *Note：*

		1. When it is 1, the clock is enabled; when it is 0, if the system or other modules selected this clock source, the clock is still enabled. 2 Since XOSCL is external clock, the corresponding pin function must be set as XOSCL function to use it
2	XLSTA	XOSCL clock stabilization flag (1 indicates it is stabilized)
1	XHCKE	XOSCH enable control 1：enable 0：disable *Note：* *1. When it is 1, the clock is enabled; when it is 0, if the system or other modules selected this clock source, the clock is still enabled.* *2. Since XOSCH is external clock, the corresponding pin function must be set as XOSCH function to use it*
0	XHSTA	XOSCH clock stabilization flag (1 indicates it is stabilized)

Table 9-2-2 Register PLLCON

AEH	7	6	5	4	3	2	1	0
PLLCON	PLLON	MULFT[3:0]				-	-	PLSTA
R/W	R/W	R/W	R/W	R/W	R/W	-	-	R/W
Initial Value	0	0	0	0	0	-	-	0

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

Table 9-2-3 Register IHCFGL、IHCFGH

8083H	7	6	5	4	3	2	1	0

IHCFGL	IHCFG[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0
8084H	7	6	5	4	3	2	1	0
IHCFGH	IHCFG[15:8]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
15~0	IHCFG	IRCH frequency correction register Note: This register is automatically loaded after power on value corresponds to frequency 3.6864MHz. It is not recommended to modify this value except for special applications.

Table 9-2-4 Register ILCFGL、 ILCFGH

8085H	7	6	5	4	3	2	1	0
ILCFGL	ILCFG[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0
8086H	7	6	5	4	3	2	1	0
ILCFGH	-	-	-	-	-	-	-	ILCFG[8]
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
8~0	ILCFG	IRCL frequency correction register Note: This register is automatically loaded after power on value corresponds to frequency 131KHz. It is not recommended to modify this value except for special applications.

Table 9-2-5 Register TFCFG

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

Bit number	Bit Symbol	Description
7~0	TFCFG	TFRC frequency correction register Note: This register is automatically loaded after power on value corresponds to frequency 4MHz. It is not recommended to modify this value except for special applications.

9.3 System Clock

All of the clock sources in CA51F2 Series can be set as system clock. The system clock is controlled by register CKCON, CKSEL and CKDIV. Users can disable/enable any of these clock sources, divide the frequency, change the system clock and so on by using these registers.

9.3.1 System Clock Architecture

Please refer to figure 9-3-1.

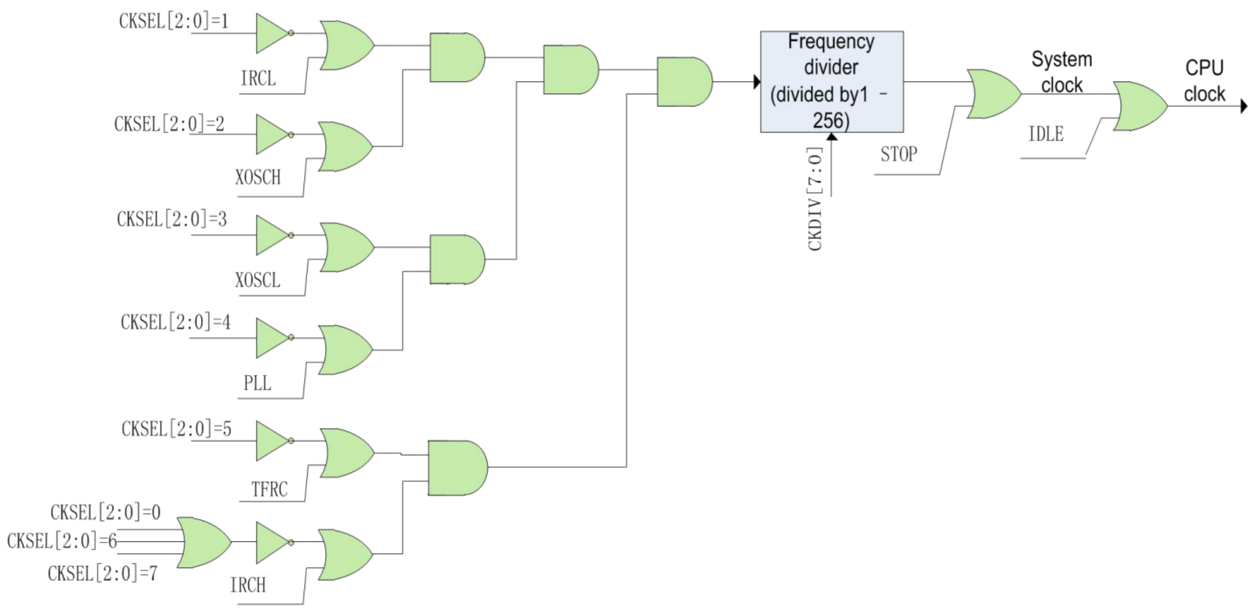


Figure 9-3-1 System Clock Architecture

9.3.2 System Clock Control Register Description

Table 9-3-2-1 Register CKSEL

8081H	7	6	5	4	3	2	1	0
CKSEL	RTCKS	-	-	-	-	CKSEL[2:0]		
R/W	R/W	-	-	-	-	R/W		
Initial Value	0	-	-	-	-	0	0	0

Bit number	Bit Symbol	Description
7	RTCKS	RTC clock selection

		0：XOSCL 1：IRCL Note： when IRCL is selected, the clock's frequency will be divided by 4 first and then used for RTC
6~3	-	-
2~0	CKSEL	System clock selection: 000：IRCH 001：IRCL 010：XOSCH/ERC 011：XOSCL 100：PLL 101：TFRC Others： IRCH Note: If you set IRCL as system clock, you must wait for about 1ms after enabling IRCL clock and then switch to system clock, otherwise an exception may occur.

Table 9-3-2-2 Register CKDIV

8082H	7	6	5	4	3	2	1	0
CKDIV	CKDIV[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
7~0	CKDIV	System clock frequency division： 00H：No division 01H：frequency divided by 2 02H：frequency divided by 3 03H：frequency divided by 4 …… FFH： frequency divided by 256						

9.3.3 System Clock Control Method and Example

- **Set IRCH as the system clock**

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

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

- **Set XOSCL as the system clock**

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

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

- **Set IRCL as the system clock**

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

```
------------------------------------------------------------------------------------------
#define ILCKE        (1<<7)
#define CKSEL_IRCL      1
void Sys_Clk_Set_IRCL(void)
{
        CKCON |= ILCKE;          //enable IRCL
        CKSEL = (CKSEL&0xF8) | CKSEL_IRCL;     //set IRCL as system clock
}
------------------------------------------------------------------------------------------
```

- **Set PLL as system clock**

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

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

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

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


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

- **Set XOSCH as the system clock**

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

```
------------------------------------------------------------------------------------
#define XHCKE               (1<<1)
#define XLSTA               (1<<2)
#define CKSEL_XOSCH          2
void Sys_Clk_Set_XOSCH(void)
{
        P74F = 3; //set P74 as XOSCH pin
        P73F = 3; //set P73 as XOSCH pin
        CKCON |= XHCKE;       //enable XOSCH clock
        while(!(CKCON & XHSTA));/wait for the stable state of XOSCH
        CKSEL = (CKSEL&0xF8) | CKSEL_XOSCH;        //set XOSCH as the system clock
}
------------------------------------------------------------------------------------
```

- **Set TFRC as the system clock**

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

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

9.4 Internal RC Oscillator Correction

9.4.1 Correction Module Introduction

Due to difference between manufacturing process, the factory frequency for RC oscillator, so correction is a must for RC oscillators. There is correction module embedded in CA51F2 Series chip which can corrects internal RC oscillators. There are 5 reference clock sources (IRCH、IRCL、XOSCL、XOSCH、TFRC) and 3 target clock sources (IRCH、IRCL、TFRC) for correction module. The reference clock source must be correct otherwise the target clock(the clock to be corrected) will be influenced. Figure 9-4-1 shows the circuit for correction module (VCLK is the reference clock and TCLK is the target clock).

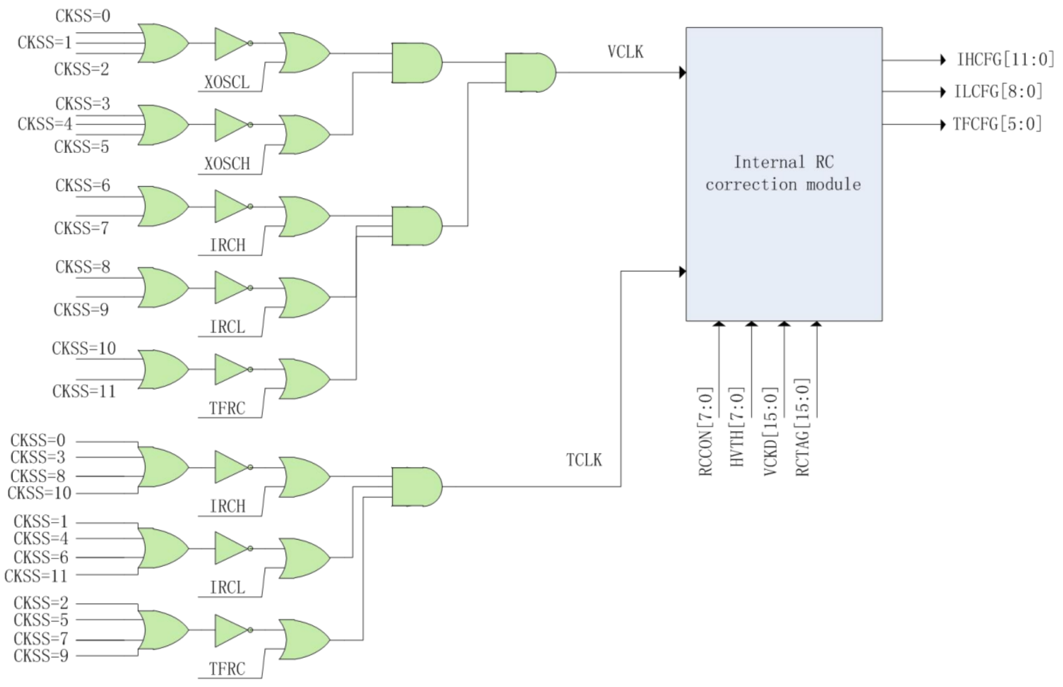


Figure 9-4-1 Correction Circuit Schematic

There are three working modes for RC correction module:

- **Count mode**

Counting mode is used for manual measurement of TCLK. TCLK counting is started after setting MODE (RCCON[7:6]) to 1, and counting is stopped after MODE is set to 0. The count value is stored in register RCMS (RCMSHH/RCMSHL/RCMSLH/RCMSLL) after stopping counting. In the application, the user can start TCLK counting within a determined time period, and the frequency of TCLK can be obtained by a simple calculation of the count value RCMS.

- **Measure mode**

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

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

- **Correction Mode**

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

VCLK cycle × VCKD ＝ TCLK target cycle × RCTAG

9.4.2 Correction Module Control Register

Table 9-4-2-1 Register RCCON

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

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

		0010：target clock TFRC, reference clock XOSCL 0011：target clock IRCH, reference clock XOSCH 0100：target clock IRCL, reference clock XOSCH 0101：target clock TFRC, reference clock XOSCH 0110：target clock IRCL, reference clock IRCH 0111：target clock TFRC, reference clock IRCH 1000：target clock IRCH, reference clock IRCL 1001：target clock TFRC, reference clock IRCL 1010：target clock IRCH, reference clock TFRC 1011：target clock IRCL, reference clock TFRC Others：invalid

Table 9-4-2-2 Register HVTH

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

Bit number	Bit Symbol	Description
7~0	HVTH	Threshold register for the difference between corrected value and target value in correction mode

Table 9-4-2-3 Register VCKDL、VCKDH

92H	7	6	5	4	3	2	1	0
VCKDL	VCKD[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
93H	7	6	5	4	3	2	1	0
VCKDH	VCKD[15:8]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
15~0	VCKD	The times reference clock will be multiplied in measure and correction mode (VCKD>1)

Table 9-4-2-4 Register RCTAGL、RCTAGH

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

95H	7	6	5	4	3	2	1	0
RCTAGH	RCTAGH[15:8]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
15~0	RCTAG	In correction mode, the target clock frequency division multiple is RCTAG frequency division multiple (RCTAG>=1).						

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

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

9.4.3 Correction Module Control Example

◆ **IRCH Correction**

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

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

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

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

◆ **IRCL Correction**

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

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

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

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

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

◆ **TFRC Correction**

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

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

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

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

9.5 External Clock Monitor

9.5.1 Function Introduction

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

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

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

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

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

9.5.2 External Clock Monitor Control Register

Table 9-5-2-1 Register CKMON

BEH	7	6	5	4	3	2	1	0
CKMON	MHE	IHE	ATH	HSW	MLE	ILE	ATL	LSW
R/W	R/W	R/W	R/W	W	R/W	R/W	R/W	W
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7	MHE	XOSCH clock monitor control, 1 enables it
6	IHE	XOSCH clock monitor interrupt enable control, 1 enables it
5	ATH	XOSCH automatic recovery enable control, 1 enables it Note: When XOSCH abnormality is detected, the corresponding circuit’s clock will change to IRCL. If

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

Table 9-5-2-2 Register CKMIF

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

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

10 Power Supply and Reset System

10.1 Power Supply

There is 1.8V - 5.5V source between VDD pin and VSS pin for CA51F2 series which supplies the power for the chip. VDD and LDO supply power for the analog system and LDO supplies power for the digital system.

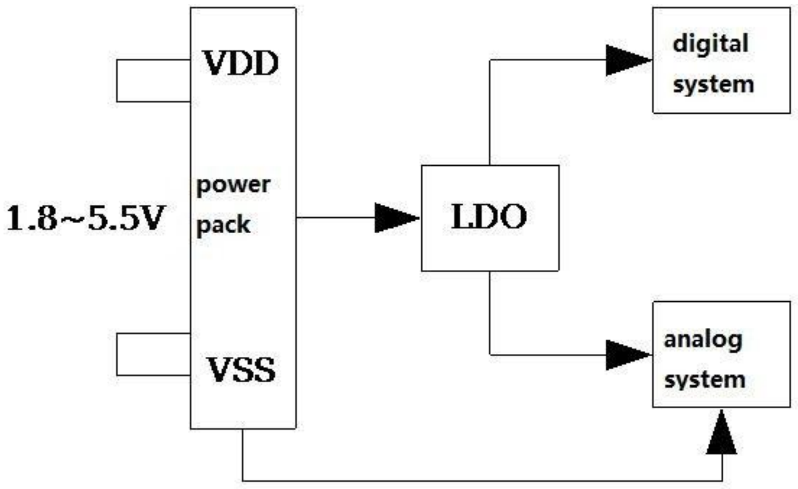


Figure 10-1-1 Power Supply Architecture

The Figure 10-1-2 is the typical circuit for power supply

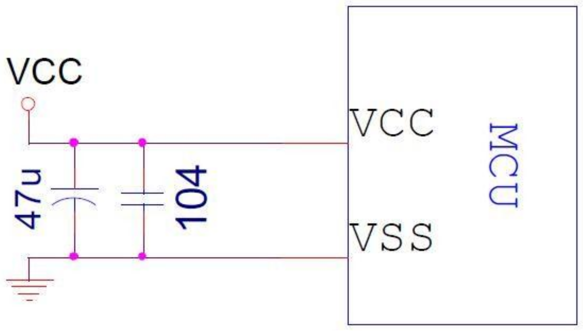


Figure 10-1-2 Typical Circuit for Power Supply

Note：1. The filter capacitors 10uF and 104 in the circuit below are the standard devices for the chip which can not be omitted, otherwise the chip may operate abnormally.
2.The above circuit and component parameters are for reference only, according to the
3.peripheral operating environment and different voltage supply parameters may need to be modified

10.1.1 LDO Function Introduction

There is an internal low dropout regulator (LDO) for CA51F2 Series chip. LDO module offers supply voltage for the chip. The output voltage of LDO is set by VLEVEL (PWCON[2:0]) and the default value for VLEVEL is 3, which implies the default voltage is 1.58V. When VDD/VSS is less than the output voltage set by VLEVEL, the output voltage will be VDD directly; when VDD/VSS is greater than the output voltage set by VLEVEL, LDO output the voltage set by VLEVE. High LDO voltage is benefits to clock module’s rapid start while low LDO voltage will lower the chip’s power consumption. There are two working modes for LDO: High Power mode and Low Power mode, which is selected by VHL (PWCON[3]). The load capacity is also different in different modes. The current is higher in High Power mode but with higher power consumption, while in Low Power mode it is vice versa. For the most time when the system is operating normally, LDO is usually High Power mode. The Low Power mode is usually used for Power Save Modes such as STOP, IDLE, Low Speed Mode etc.

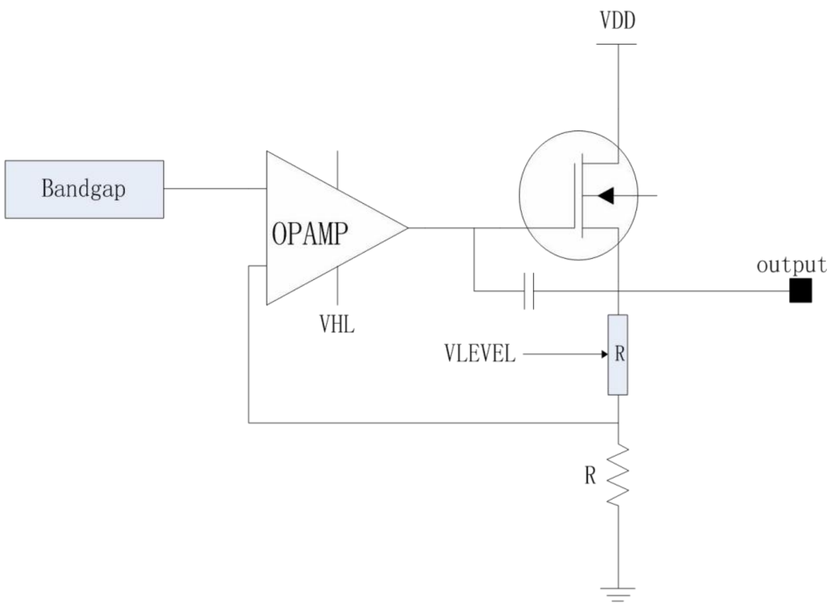


Figure 10-1-3 LDO Module Schematic

10.1.2 LDO Control Register

Table 10-1-2-1 Register PWCON

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

Bit Number	Bit symbol	discription
7~4	FLEVEL	Internal reference voltage (Bandgap) output adjustment bit 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 system automatically loads the internal reference voltage when it is powered on. Users are not allowed to modify the internal reference voltage.*
3	VHL	LDO Working mode control bit 1：high power mode 0：low power mode
2~0	VLEVEL	LDO Output voltage setting bit 000：1.31V 001：1.37V 010：1.43V 011：1.49V 100：1.55V 101：1.61V 110：1.67V 111：1.73V *Note:* *1. The internal clock circuit by the power supply, "change" the output voltage can cause the change of the internal clock frequency, in general, they keep the default value, voltage change is not recommended.* *2. Not allowed to set "output voltage is less than 1.5 V, otherwise may cause abnormal.*

10.2 Reset System

There are multiple internal and external reset sources for CA51F2 Series chip as figure 10-2-1 shows.

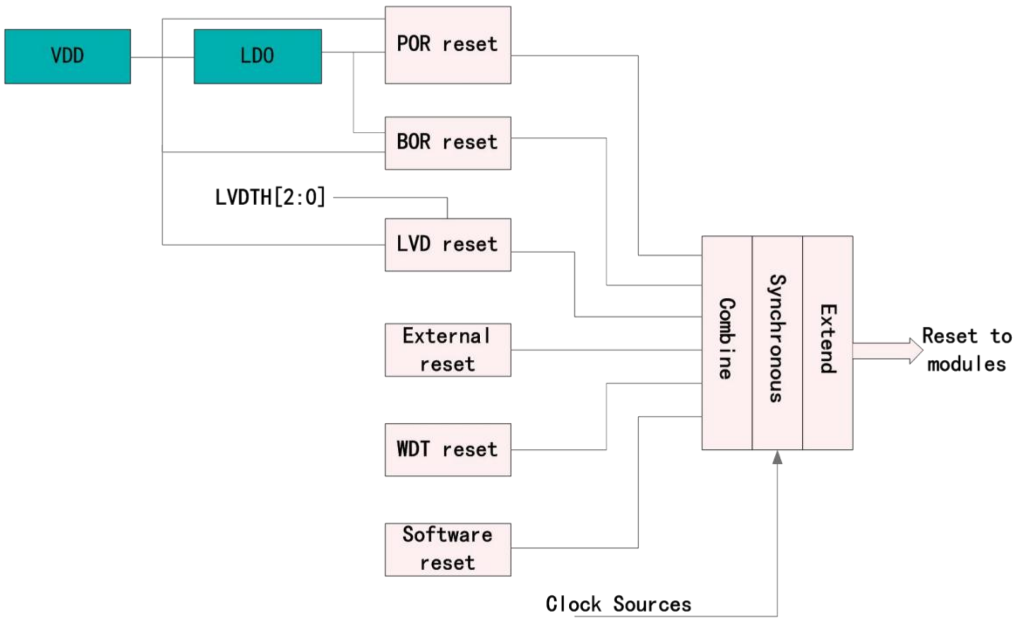


Figure 10-2-1 Reset System Architecture

- **Power On Reset(POR)**

System power on shows a gradually increasing curve form. 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 start from certain stable status. The POR signal will also be expanded by the internal counter to makes sure that all the analog modules can enter stable working status after Power On stage.

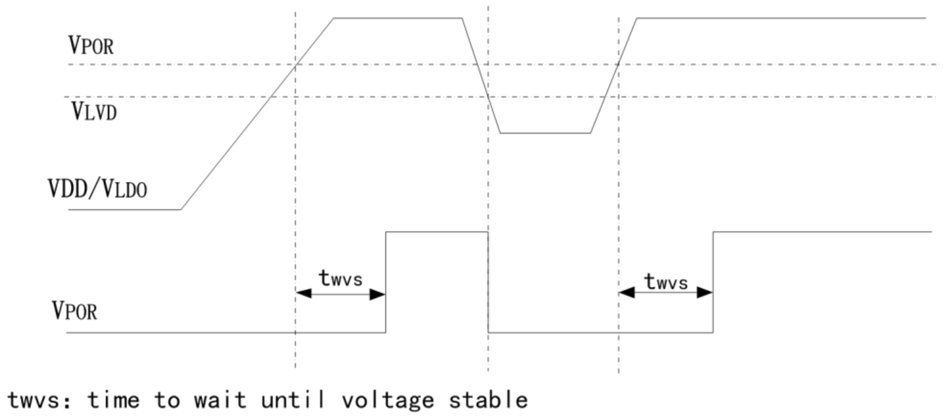


Figure 10-2-2 POR Circuit Example and Power On Stage

- **Brown Out Reset(BOR)**

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

- **Low Voltage Reset**

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

- **External Reset**

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

- **Watchdog Reset**

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

- **Soft Reset**

The program can soft reset the chip. When 1 is written to SWRST of register PCON, CPU sends out reset signal.

POR and external hard reset will reset all the circuits while LVD and WDT can reset other circuits but not reset themselves. (eg：After WDT reset, WDT registers remains former status while others are all reset) LVD/WDT and soft reset can not reset storage control circuit. Program starts from Mask ROM after POR and external hard reset. Program starts from where BOOT configuration points to after soft reset. PC will point to address 0 after any reset.

11 Power Consumption Management

There are 3 low power consumption modes for CA51F2 Series : IDLE, STOP and Low Speed mode. The system power consumption for IDLE, STOP and Low Speed mode is less than 12uA, 7uA and 20uA respectively.

11.1 IDLE mode

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

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

Any reset or interrupt will awake the chip. The clock will resume first and then the chip responds to the interrupt and enters the interrupt service routine after the CPU awakening. After the chip exits interrupt service routine , it will execute 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 work, hence users may disable them to save power.

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

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

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

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

***Note*:**
1.When it enters STOP/IDLE mode, setting LDO to low power consumption mode will reduce the power consumption effectively. However, it is a must to set LDO back to high power consumption mode when the chip exits STOP/IDLE mode, otherwise the chip will operate abnormally.
2. If the system clock is selected as IRCL or TFRC, IRCL or TFRC must not be turned off when entering STOP, otherwise an exception may occur when waking up from STOP.

11.3 Low Speed Mode

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

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

11.4 Related Register Description

Table 11-4-1 Register PCON

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

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

3	TSME	Test mode control 0：disable test mode 1：test mode enable Note: Test mode can be used for the chip’s online simulation
2	TSMODE	Test mode flag, 1 indicates that the chip is in test mode
1	STOP	STOP mode control, 1 enables STOP mode When STOP=1 and STPST=0, the chip will enter STOP mode. it will be cleared to 0 automatically after the chip exits STOP mode
0	IDLE	IDLE mode control, 1 enables IDLE mode When IDLE=1and IDLST=0, the chip will enter IDLE mode. it will be cleared to 0 automatically after the chip exits IDLE mode

Table 11-4-2 Register IDLSTL、IDLSTH

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

Bit number	Bit Symbol	Description
15	-	-
14	SMINT/PWMINT/EPIF[7]	Interrupt status of SAMPLE/PWM/External Interrupt 9 in IDLE mode
13	RTCINT/CTMINT/EPIF[6]	Interrupt status of RTC/Comparator Counter/External Interrupt 8 in IDLE mode
12	MOTINT/WDFLG[1]/EPIF[5]	Interrupt status of MOTOR/WDT/External Interrupt 7 in IDLE mode
11	I2CINT/CPINT/EPIF[4]	Interrupt status of I2C/Analog Comparator/External Interrupt 6 in IDLE mode
10	SPINT/CKMINT/EPIF[3]	Interrupt status of SPI/Clock Monitor/External Interrupt 5 in IDLE mode
9	LVDINT/EPIF[2]	Interrupt status of LVD/External Interrupt 4 in IDLE mode
8	TKINT/U2INT/EPIF[1]	Interrupt status of TK/UART2/External Interrupt 3 in IDLE mode
7	ADCINT/EPIF[0]	Interrupt status of ADC/External Interrupt 2 in IDLE mode
6	U1INT	Interrupt status of UART1 Interrupt in IDLE mode
5	T2INT	Interrupt status ofTimer2 in IDLE mode
4	U0INT	Interrupt status of UART0 in IDLE mode
3	TCON[7]	Interrupt status ofTimer1 in IDLE mode
2	PIF[1]	Interrupt status of External Interrupt 1 in IDLE mode
1	TCON[5]	Interrupt status of Timer0 in IDLE mode

0	PIF[0]	Interrupt status of External Interrupt0 in IDLE mode

Table 11-4-3 Register STPSTL、STPSTH

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

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

11.5 Low Power Consumption Control Example

◆ **STOP Mode Example**

The program is like：

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

```
        PWCON &=0xF7; //set LDO in low power consumption mode
        MECON |= (1<<6); //set FLASH in deep sleep mode
        while(STPSTH|STPSTL);// wait until all the interrupts are done
        PCON |= 0x02;        // enters STOP mode
        _nop_();
        _nop_();
        PWCON ｜=0x08; // The LDO must return to high power consumption mode after the chip exits STOP mode

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

- **IDLE Mode Example**

The program is like：

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

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

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

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

- **Low Speed Mode Example**

The program is like：

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

}

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

----

12 Timer(Timer0,Timer1,Timer2)

12.1 Timer0

12.1.1 Timer0 Introduction

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

- **Mode0**

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

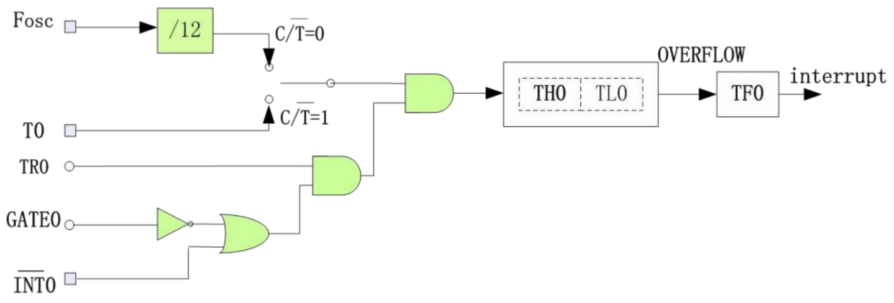


Figure 12-1-1-1 Timer0 Mode0/1

- **Mode2**

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

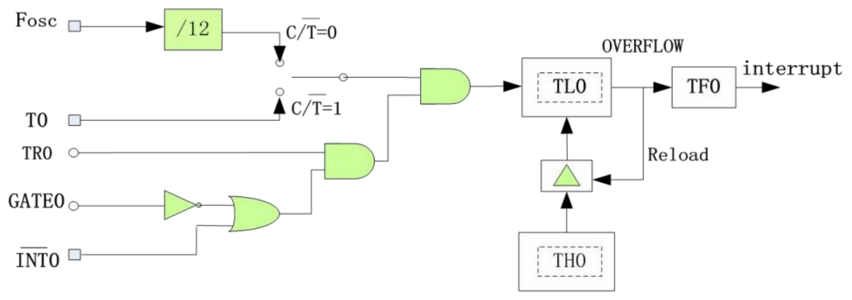


Figure 12-1-1-2 Timer0 Mode2

- **Mode3**

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

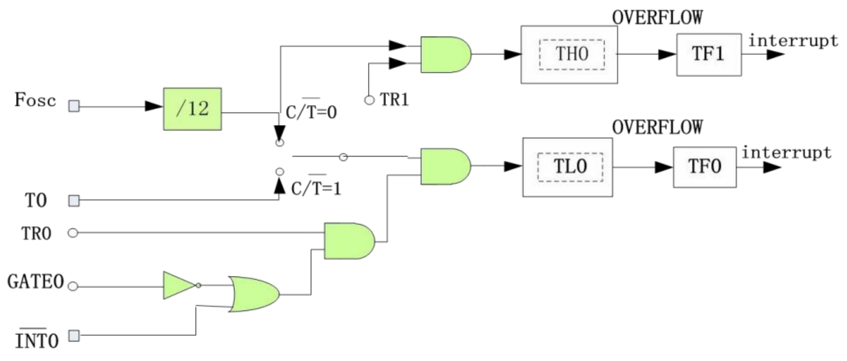


Figure 12-1-1-3 Timer0 Mode3

12.1.2 Timer0 Register Description

Table 12-1-2-1 Register TCON

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

Bit number	Bit Symbol	Description
7	TF1	Timer0 TH0 overflow flag in mode3 /Timer1 overflow flag, it is cleared automatically after the interrupt response
6	TR1	Timer1 enable control, 1 enables it
5	TF0	Timer0 overflow flag, it is cleared automatically after the interrupt response
4	TR0	Timer0 enable control, 1 enables it
3	IE1	External Interrupt1 enable control, 1 enables it
2	IT1	External Interrupt1 trigger type control

		0：External Interrupt1 is triggered when input pin signal is low 1：External Interrupt1 is triggered when input pin signal comes to falling edge
1	IE0	External Interrupt0 enable control, 1 enables it
0	IT0	External Interrupt0 trigger type control 0：External Interrupt0 is triggered when input pin signal is low 1：External Interrupt0is triggered when input pin signal comes to falling edge

Table 12-1-2-2 Register TMOD

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

Bit number	Bit Symbol	Description
7	GATE1	Timer1 gating control. When it equals 1, Timer1 is enabled/disabled by INT1
6	CT1	Timer1 Counter/Timer selection 0：Timer, the clock for it is the system clock with its frequency divided by 12 1：Counter, the clock for it is T1 input clock
5	T1M1	[T1M1,T1M0] for Timer1 mode selection 00：mode0, TL1 and TH1 make up a 13-bit Timer/Counter 01：mode1, TL1 and TH1 make up a 16-bit Timer/Counter 10：mode2, TL1 is a 8 bit Timer/Counter, TH1 is the automatic reload register 11：mode3, TH1/TL1 locked in this mode, it is the same as TR1=0
4	T1M0	
3	GATE0	Timer0 gating control. When it equals 1, Timer0 is enabled/disabled by INT0
2	CT0	Timer0Counter/Timer selection 0：Timer, the clock for it is the system clock with its frequency divided by 12 1：Counter, the clock for it is T0 input clock
1	T0M1	[T0M1,T0M0] Timer0 mode selection 00：mode0, TL0 and TH0 make up a 13-bit Timer/Counter 01：mode1, TL0 and TH0 make up a 16 bit Timer/Counter 10：mode2, TL0 is a 8 bit Timer/Counter, TH0 is the automatic reload register 11：mode3, TL0 and TH0 are two independent 8 bit Timer/Counter
0	T0M0	

Table 12-1-2-3 Register TL0

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

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

Table 12-1-2-4 Register TH0

8CH	7	6	5	4	3	2	1	0
TH0	TH0							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
7~0	TH0	Higher byte of Timer0's count value in mode0/1, reload value in mode2, count value in mode3						

12.2 Timer1

12.2.1 Timer1 Introduction

The timer/counter function can be selected by CT1 (TMOD[6]). When CT1=0 it operates as a timer; when CT1=1, it functions as a counter. As a timer, its clock is the system clock with frequency divided by 12. 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, timer 1 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 and should be ignored when read. When timer 1 overflows, the interrupt flag bit TF1 (TCON[7]) is set to 1. When the interrupt is responded to, the TF1 bit is automatically cleared to 0. When GATE1 (TCON[7]) =0, timer/counter is enabled and counted by TR1 (TCON[6]) bit. When GATE1=1, timer/counter is enabled and controlled by pin INT1. INT1 counts at high power level, and INT1 stops counting at low level.

- **Mode1**

Timer1 operates as a 16-bit timer/counter in this mode. TH1 stores the higher 8bits of the 16-bit timer/counter and TL1 stores the lower 8 bits. When Timer1 overflows, the interrupt flag TF1 (TCON[7]) will be set to 1. TF1 will be cleared automatically after the interrupt response. When GATE1 (TCON[7])=0, the Timer/Counter's is enabled/disabled by TR1 (TCON[6]). When GATE1=1, the timer/counter's is enabled/disabled by INT1. INT0 signal with high level with enable the counting and vice verse,

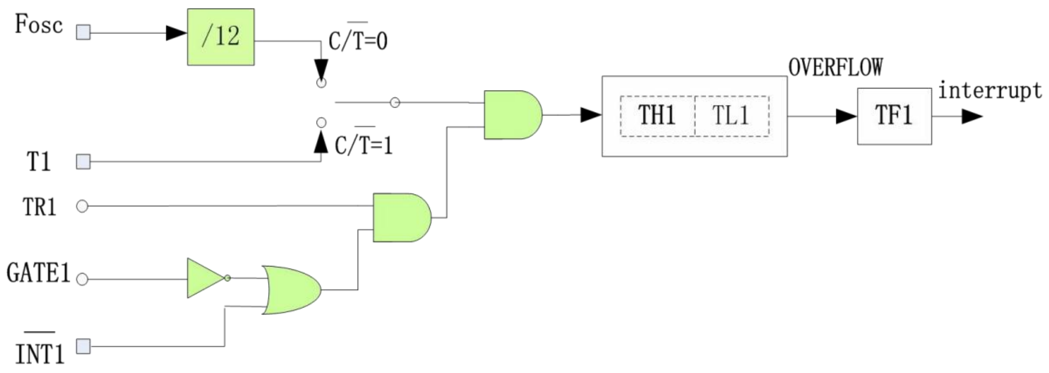


Figure 12-2-1 Timer1 Mode1

- **Mode2**

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

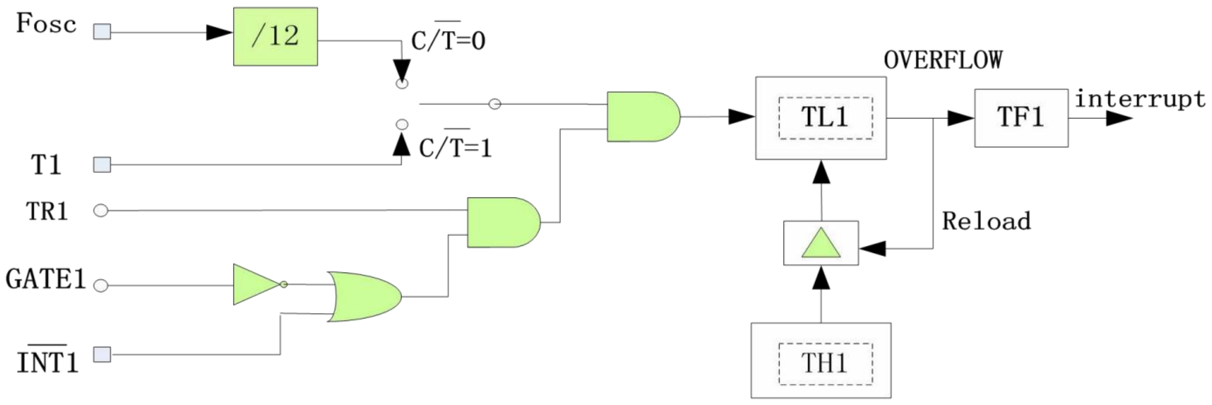


Figure 12-2-2 Timer1 Mode2

- **Mode3**

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

12.2.2 Timer1 Register Description

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

Table 12-2-2-1 Register TL1

8BH	7	6	5	4	3	2	1	0
TL1	TL1							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~0	TL1	Lower byte of Timer1 count value in mode0/1, count value in mode2/3

Table 12-2-2-2 Register TH1

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

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

12.3 Timer2

12.3.1 Timer2 Introduction

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

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

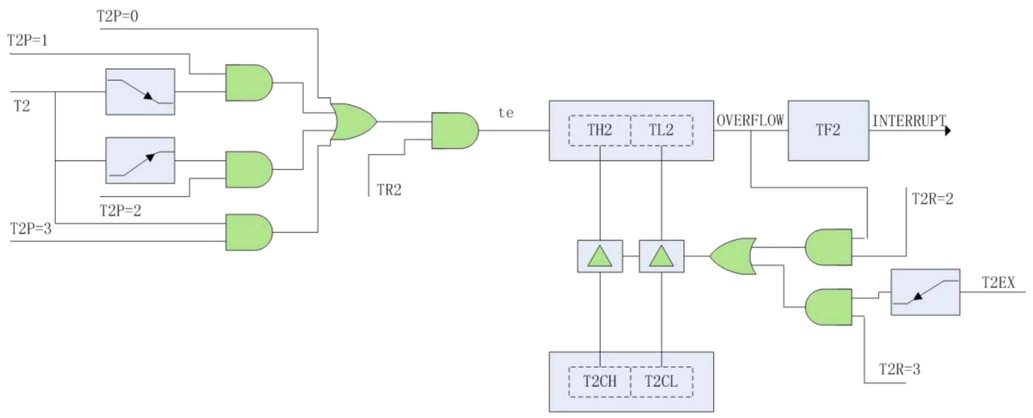


Figure 12-3-1-1 Timer2 Reload Mode

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

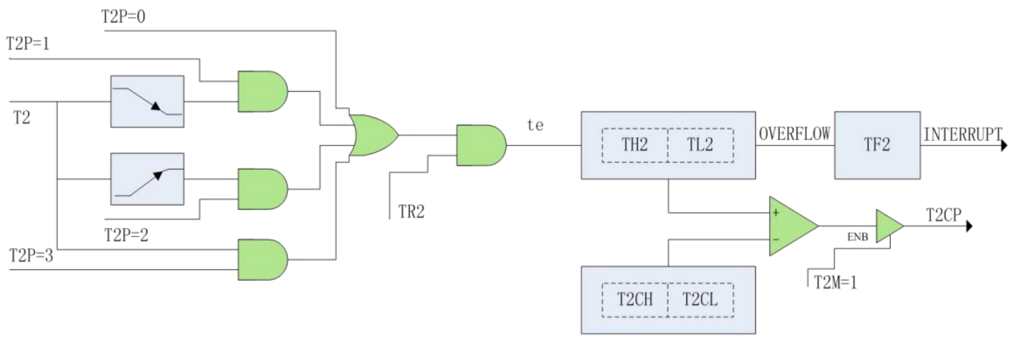


Figure 12-3-1-2 Timer2 Compare Mode Compare Mode

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

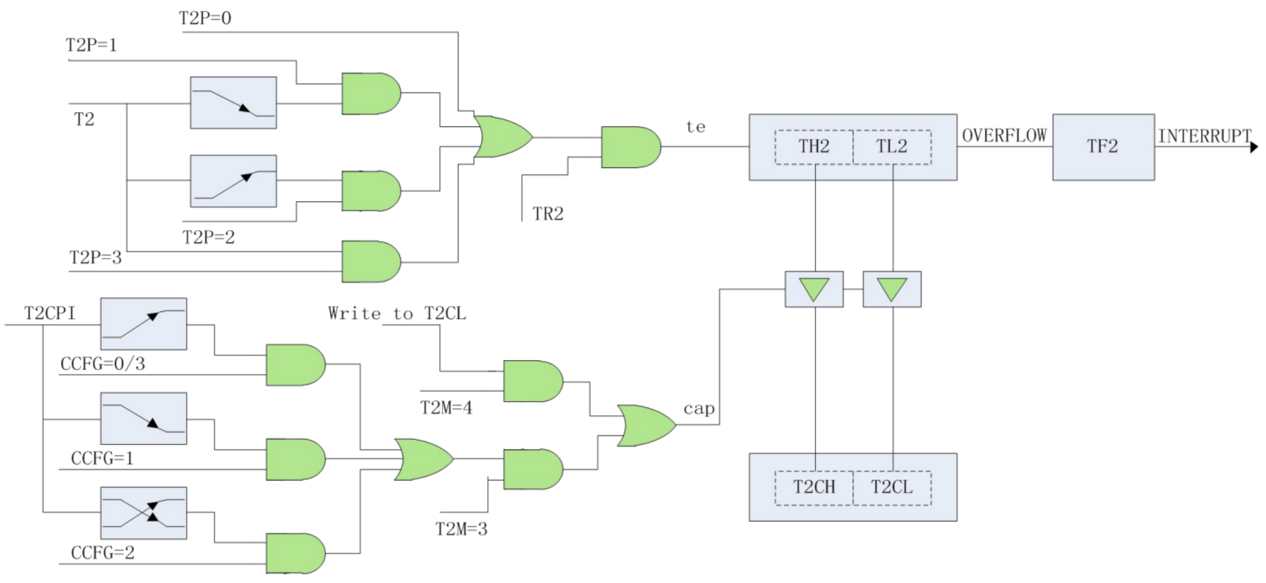


Figure 12-3-1-3 Timer2 Capture Mode

12.3.2 Timer2 Register Description

Table 12-3-2-1 Register T2CON

C8H	7	6	5	4	3	2	1	0
T2CON	-	TR2	T2R1	T2R0	T2IE	UCKS	T2P1	T2P0
R/W	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	-	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7	-	-
6	TR2	Timer2 enable control, 1 enables it
5	T2R1	[T2R1,T2R0]Timer2reload mode selection 10： mode0 11： mode1 Others： disable reload mode
4	T2R0	
3	T2IE	Timer2 interrupt enable control,1 enables it
2	UCKS	UART0 clock selection 0：Timer1 overflow impulse used for UART0 clock 1：Timer2 overflow impulse used for UART0 clock
1	T2P1	[T2P1,T2P0]Timer2 pin T2 function selection 00：Timer2 uses internal system clock for count instead of T2 01：Timer2 counts T2 falling edges 10：Timer2 counts T2 rising edges 11：Timer2 uses internal system clock for count which is gated by T2
0	T2P0	

Table 12-3-2-2 Register T2MOD

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

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

Table 12-3-2-3 Register T2CL

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

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

Table 12-3-2-4 Register T2CH

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

Bit number	Bit Symbol	Description

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

Table 12-3-2-5 Register TL2

CCH	7	6	5	4	3	2	1	0
TL2	TL2							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
7~0	TL2	Lower byte of the count value in Timer2						

Table 12-3-2-6 Register TH2

CDH	7	6	5	4	3	2	1	0
TH2	TH2							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
7~0	TH2	Higher byte of the count value in Timer2						

13 Watchdog Timer(WDT)

13.1 Watchdog Timer(WDT) Function Introduction

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

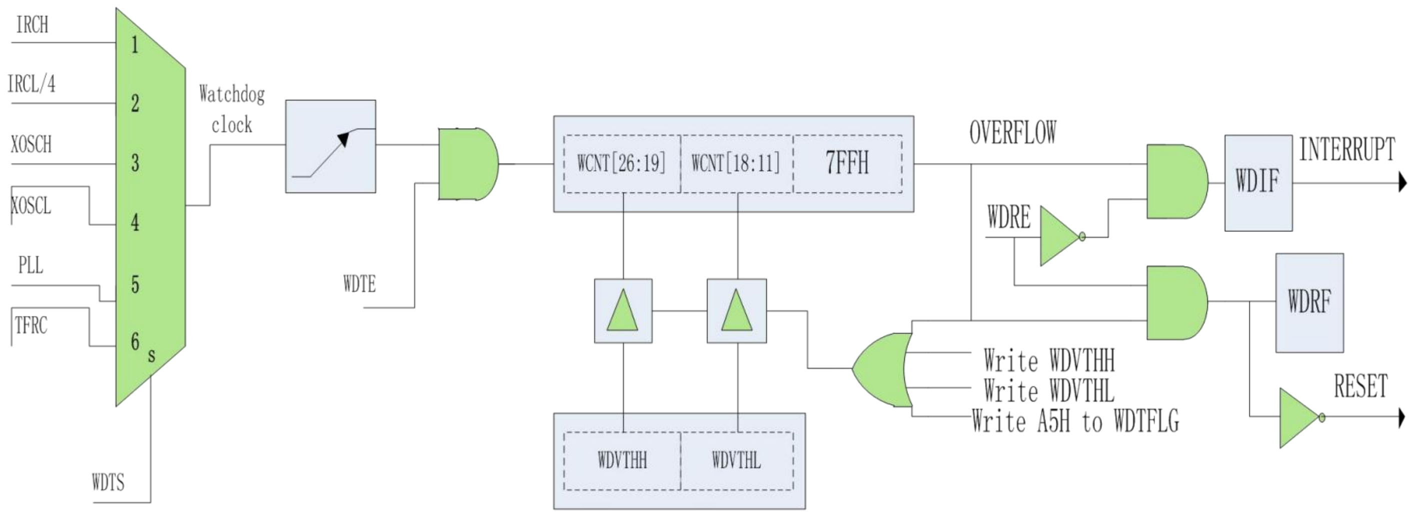


Figure 13-1-1 Watchdog Module Architecture

13.2 Watchdog Timer(WDT) Register Description

Table 13-2-1 Register WDCON

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

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

Table 13-2-2 Register WDFLG

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

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

Table 13-2-3 Register WDVTHL、WDVTHH

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

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

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 3.6864MHz. The watchdog works in interrupt mode and the overflow time is one second, the program is like:

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

◆ **Example for watchdog reset mode**

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

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

14 Real Time Clock(RTC)

14.1 RTC Function Introduction

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

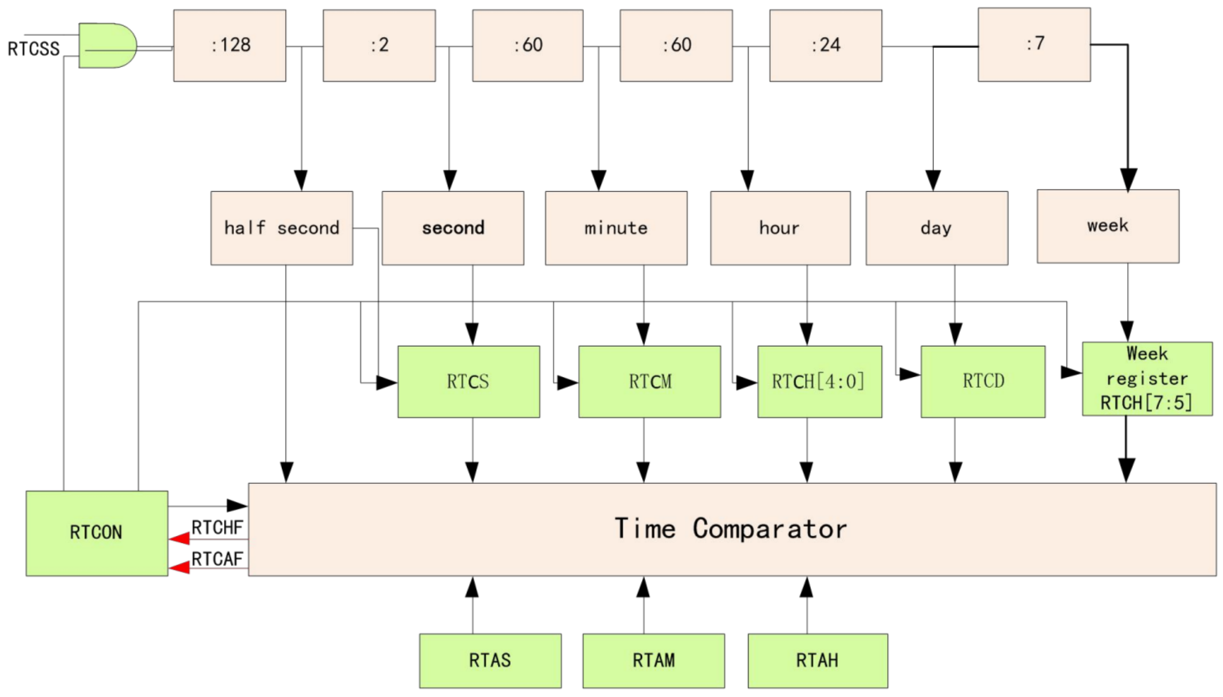


Figure 14-1-1 RTC Architecture

- **Enable/disable RTC**

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

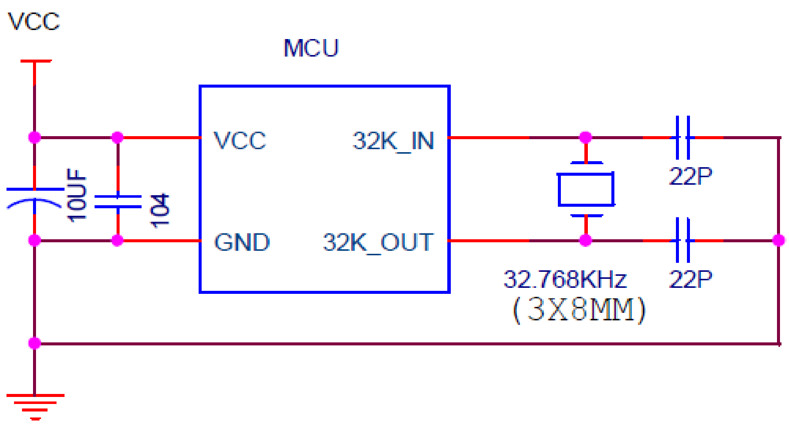
- **RTC Register R/W**

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

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

- **RTC Alarm Clock**

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



Important note: 1. During hardware design, the crystal oscillator load capacitor must be connected to the chip ground, and the crystal oscillator compensation capacitor should be as close as possible to the chip GND pin. 32.768 KHz semiconductor vibration requires the use of 3 mmx8mm crystals diameter specifications. 2. The above circuits and components parameters are for reference only, use different manufacturer crystals in circuit using parameters may need to change.

14.2 RTC Register Description

Table 14-2-1 Register RTCON

F1H	7	6	5	4	3	2	1	0
RTCON	RTCE	MSE	HSE	SCE	MCE	HCE	RTCWE	-
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	-
Initial Value	0	0	0	0	0	0	0	-
Bit number	Bit Symbol	Description						
7	RTCE	RTC clock enable control, 1 enables it						

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

Table 14-2-2 Register RTCSS

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

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

Table 14-2-3 Register RTCS

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

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

Table 14-2-4 Register RTCM

F3H	7	6	5	4	3	2	1	0
RTCM	-	-	RTCM[5:0]					
R/W	-	-	R/W					
Initial Value	-	-	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~6	-	-
5~0	RTCM	Minute counter ranges from 0 to 59, 1 is added to it every 1 minute

Table 14-2-5 Register RTCH

F4H	7	6	5	4	3	2	1	0
RTCH	RTCW[2:0]			RTCH[4:0]				
R/W	R/W			R/W				
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~5	RTCW	Week counter ranges from 1 to 7 which stands for Monday to Sunday, when it is set to 0, the week count function is disabled
4~0	RTCH	Hour counter ranges from 0 to 23, 1 is added to it every 1 hour

Table 14-2-6 Register RTCDL、RTCDH

F5H	7	6	5	4	3	2	1	0
RTCDL	RTCD[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
F6H	7	6	5	4	3	2	1	0
RTCDH	RTCD[15:8]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
15~0	RTCD	Day counter, 1 is added to it every day

Table 14-2-7 Register RTAS

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

Bit number	Bit Symbol	Description
7~6	-	-
5~0	RTAS	The alarm clock second setting (ranges from 0 to 59)

Table 14-2-8 Register RTAM

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

Bit number	Bit Symbol	Description
7~6	-	-
5~0	RTAM	The alarm clock minute setting (ranges from 0 to 59)

Table 14-2-9 Register RTAH

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

Bit number	Bit Symbol	Description
7~5	-	-
4~0	RTAH	The alarm clock hour setting (ranges from 0 to 23)

Table 14-2-10 Register RTMSS

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

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

Table 14-2-11 Register RTCIF

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

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

14.3 RTC Control Example

- **Write time to RTC**

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

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

- **Set the alarm clock time**

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

```
---------------------------------------------------------------------------------------
#define SCE(N)            (N<<4)    //N=0~1
#define MCE(N)            (N<<3)    //N=0~1
#define HCE(N)            (N<<2)    //N=0~1
#define RTC_AF            (1<<0)
Void RTM_init(void)
{
       RTAH = 11;    //set the hour for the alarm clock
       RTAM = 30;    //set the minute for the alarm clock
```

```
        RTAS = 0;       //set the second for the alarm clock
        RTCON |= SCE(1)|MCE(1)|HCE(1); //enables hour, minute and second compare
}
void RTC_ISR (void) interrupt 13
{
        if(RTCIF & RTC_AF)                      //alarm clock interrupt
        {
                RTCIF = RTC_AF;
//alarm clock interrupt service routine

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

◆ **RTC initialization**

RTC 初始化 the program is like：

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

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

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

		}
}
```
----

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

15.1 Function Introduction

The CA51F2 series chips have a maximum package of 62 I/O pins. Each I/O pin is a reusable function pin, which can not only be independently programmed as an input/output port, but also can be set as other function pins Function setting registers PnxF and PnxC are assigned to each pin (corresponding to pin Pnx respectively, where n=0~7 represents P0~P7,x=0~7, 0 to Pn.7), the user can configure the main function and other options of pins through registers PnxF and PnxC Pull-down resistors can be enabled for each I/O through PnxPUP/PnxPDP(PnxF[7]/PnxF[6]). The strong/weak pull-up resistors are selected by PU_SEL/PD_SEL(PnxC[5]/PnxC[4]). When PU_SEL/PD_SEL is 1, the strong pull-up resistors are selected, otherwise, the pull-down resistors are selected Weak up/Down, strong up/Down by default When I/O is set to output mode, when PnxOPR(PnxF[5]) is set to 1,I/O is open/miss output mode When I/O is push-pull output, DRV(PnxC[3:2]) can set the drive strength of IO push-pull output, and SR(PnxC[1:0]) can set the flip slope of IO output. When I/O output level is flipped, overshoot signal will be generated in THE I/O port due to inductance effect, which may have certain influence on the chip system. Reducing I/O output intensity and I/O speed can effectively reduce overshoot signal amplitude. In application, these two parameters can be flexibly configured when I/O is input mode, SMIT_EN(PnxC[6]) bit can be used to select smIT mode or inverter mode. When it is inverter mode, the triggering threshold of high and low levels is 1/2 In addition,P00~P07 can be reused as LED COM pins, which can be set to high irrigation level mode by SINK_EN(P0xC[7]). In high irrigation current mode, the irrigation current is greater than 60mA(refer to the electrical characteristics section for test conditions).

Main features of GPIO：

- High impedance mode configurable
- I/O structure can be independently set strong pull up, weak pull up, strong pull down, weak pull down resistance
- Data output latches support read - modify - write
- Supports a wide voltage range of 1.8 to 5.5V
- When it is set to push-pull output, the IO drive strength can be set independently
- When it is set to push-pull output, the I/O output speed can be set independently
- LED COM can set high filling current up to 60mA (test conditions refer to the description of electrical characteristics section)

Important note: All GPIO pin input voltages should not be higher than VDD pin voltages, otherwise the chip may work abnormally.

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

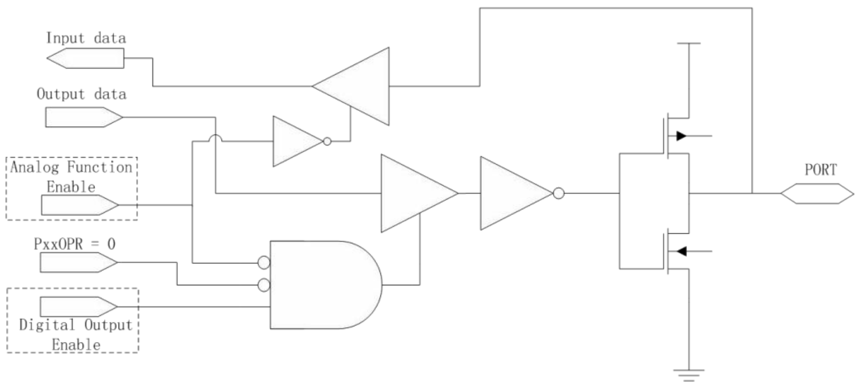


Figure 15-1-1 I/O Push-pull Mode Structure

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

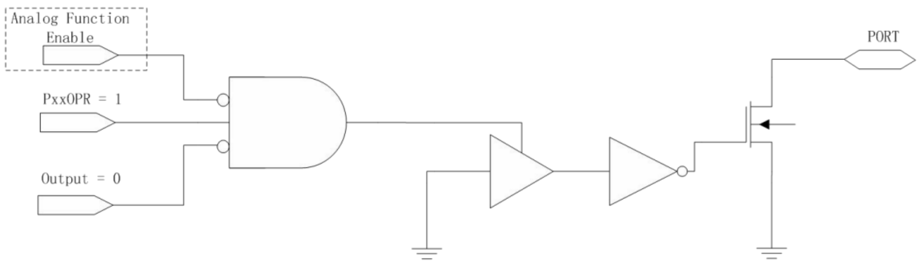


Figure 15-1-2 I/O Open-drain Mode Structure

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

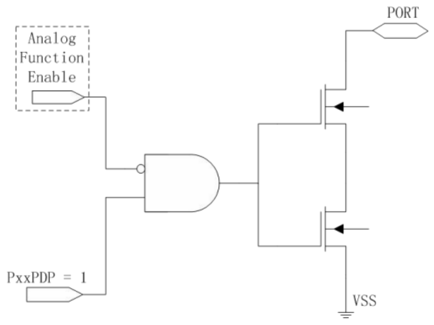


Figure 15-1-3 I/O Pull-down Mode Structure

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

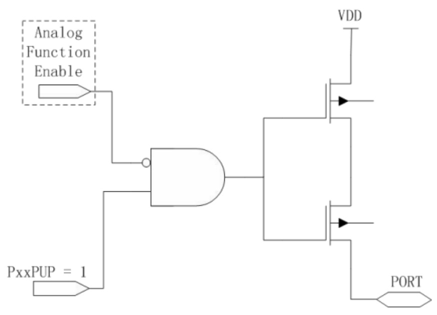


Figure 15-1-4 I/O Pull-up Mode Structure

15.2 Pin Register Description

Table 15-2-1 Register P0

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

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

Table 15-2-2 Register P1

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

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

Table 15-2-3 Register P2

A0H	7	6	5	4	3	2	1	0
P2	P27	P26	P25	P24	P23	P22	P21	P20
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

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

Table 15-2-4 Register P3

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

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

Table 15-2-5 Register P4

C0H	7	6	5	4	3	2	1	0
P4	P47	P46	P45	P44	P43	P42	P41	P40
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~0	P4x	Data register for pin P4x， valid when the pin function is set to GPIO 0：P4x is low level when the pin is set to input; when the pin set to output,P4x outputs low level signal 1：P4x is high level when the pin is set to input; when the pin set to output,P4x outputs high level signal

Table 15-2-6 Register P5

D8H	7	6	5	4	3	2	1	0
P5	P57	P56	P55	P54	P53	P52	P51	P50
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~0	P5x	Data register for pin P5x， valid when the pin function is set to GPIO 0： P5x is low level when the pin is set to input; when the pin set to output, P5x outputs low level signal 1： P5x is high level when the pin is set to input; when the pin set to output, P5x outputs high level signal

Table 15-2-7 Register P6

A9H	7	6	5	4	3	2	1	0
P6	P67	P66	P65	P64	P63	P62	P61	P60
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~0	P6x	Data register for pin P6x, valid when the pin function is set to GPIO 0： P6x is low level when the pin is set to input; when the pin set to output,P6x outputs low level signal 1： P6x is high level when the pin is set to input; when the pin set to output,P6x outputs high level signal

Table 15-2-8 Register P7

D9H	7	6	5	4	3	2	1	0
P7	-	-	P75	P74	P73	P72	P71	P70
R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	-	-	0	0	0	0	0	0

Bit number	Bit Symbol	Description
5~0	P7x	Data register for pin P7x， valid when the pin function is set to GPIO 0： P7x is low level when the pin is set to input; when the pin set to output,P7x outputs low level signal 1： P7x is high level when the pin is set to input; when the pin set to output,P7x outputs high level signal

Table 15-2-9 Pin Function Control Register

8000H	7	6	5	4	3	2	1	0
P00F	P00PUP	P00PDP	P00OPR	-	-	-	P00S	
R/W	R/W	R/W	R/W	-	-	-	R/W	
Initial Value	0	0	0	-	-	-	0	0

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

8002H	7	6	5	4	3	2	1	0
P02F	P02PUP	P02PDP	P02OPR	-	-	-	P02S	
R/W	R/W	R/W	R/W	-	-	-	R/W	
Initial Value	0	0	0	-	-	-	0	0

8003H	7	6	5	4	3	2	1	0
P03F	P03PUP	P03PDP	P03OPR	-	-	-	P03S	
R/W	R/W	R/W	R/W	-	-	-	R/W	
Initial Value	0	0	0	-	-	-	0	0

8004H	7	6	5	4	3	2	1	0
P04F	P04PUP	P04PDP	P04OPR	-	-	P04S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0

8005H	7	6	5	4	3	2	1	0
P05F	P05PUP	P05PDP	P05OPR	-	-	P05S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0

8006H	7	6	5	4	3	2	1	0
P06F	P06PUP	P06PDP	P06OPR	-	-	P06S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0

8007H	7	6	5	4	3	2	1	0
P07F	P07PUP	P07PDP	P07OPR	-	-	P07S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0

8008H	7	6	5	4	3	2	1	0

P10F	P10PUP	P10PDP	P10OPR	-	-	-	P10S	
R/W	R/W	R/W	R/W	-	-	-	R/W	
Initial Value	0	0	0	-	-	-	0	0
8009H	7	6	5	4	3	2	1	0
P11F	P11PUP	P11PDP	P11OPR	-	-	-	P11S	
R/W	R/W	R/W	R/W	-	-	-	R/W	
Initial Value	0	0	0	-	-	-	0	0
800AH	7	6	5	4	3	2	1	0
P12F	P12PUP	P12PDP	P12OPR	-	-	P12S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
800BH	7	6	5	4	3	2	1	0
P13F	P13PUP	P13PDP	P13OPR	-	-	-	P13S	
R/W	R/W	R/W	R/W	-	-	-	R/W	
Initial Value	0	0	0	-	-	-	0	0
800CH	7	6	5	4	3	2	1	0
P14F	P14PUP	P14PDP	P14OPR	-	-	-	P14S	
R/W	R/W	R/W	R/W	-	-	-	R/W	
Initial Value	0	0	0	-	-	-	0	0
800DH	7	6	5	4	3	2	1	0
P15F	P15PUP	P15PDP	P15OPR	-	-	-	P15S	
R/W	R/W	R/W	R/W	-	-	-	R/W	
Initial Value	0	0	0	-	-	-	0	0
800EH	7	6	5	4	3	2	1	0
P16F	P16PUP	P16PDP	P16OPR	-	-	P16S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
800FH	7	6	5	4	3	2	1	0
P17F	P17PUP	P17PDP	P17OPR	-	-	P17S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8010H	7	6	5	4	3	2	1	0
P20F	P20PUP	P20PDP	P20OPR	-	-	P20S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0

8011H	7	6	5	4	3	2	1	0
P21F	P21PUP	P21PDP	P21OPR	-	-	P21S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0

8012H	7	6	5	4	3	2	1	0
P22F	P22PUP	P22PDP	P22OPR	-	-	P22S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0

8013H	7	6	5	4	3	2	1	0
P23F	P23PUP	P23PDP	P23OPR	-	-	P23S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0

8014H	7	6	5	4	3	2	1	0
P24F	P24PUP	P24PDP	P24OPR	-	-	P24S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0

8015H	7	6	5	4	3	2	1	0
P25F	P25PUP	P25PDP	P25OPR	-	-	P25S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0

8016H	7	6	5	4	3	2	1	0
P26F	P26PUP	P26PDP	P26OPR	-	-	P26S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0

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

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

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

801AH	7	6	5	4	3	2	1	0
P32F	P32PUP	P32PDP	P32OPR	-	-	P32S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
801BH	7	6	5	4	3	2	1	0
P33F	P33PUP	P33PDP	P33OPR	-	-	P33S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
801CH	7	6	5	4	3	2	1	0
P34F	P34PUP	P34PDP	P34OPR	-	-	P34S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
801DH	7	6	5	4	3	2	1	0
P35F	P35PUP	P35PDP	P35OPR	-	-	P35S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
801EH	7	6	5	4	3	2	1	0
P36F	P36PUP	P36PDP	P36OPR	-	-	P36S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
801FH	7	6	5	4	3	2	1	0
P37F	P37PUP	P37PDP	P37OPR	-	-	P37S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8020H	7	6	5	4	3	2	1	0
P40F	P40PUP	P40PDP	P40OPR	-	-	P40S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8021H	7	6	5	4	3	2	1	0
P41F	P41PUP	P41PDP	P41OPR	-	-	P41S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8022H	7	6	5	4	3	2	1	0
P42F	P42PUP	P42PDP	P42OPR	-	-	P42S		
R/W	R/W	R/W	R/W	-	-	R/W		

Initial Value	0	0	0	-	-	0	0	0
8023H	7	6	5	4	3	2	1	0
P43F	P43PUP	P43PDP	P43OPR	-	-	P43S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8024H	7	6	5	4	3	2	1	0
P44F	P44PUP	P44PDP	P44OPR	-	-	P44S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8025H	7	6	5	4	3	2	1	0
P45F	P45PUP	P45PDP	P45OPR	-	-	P45S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8026H	7	6	5	4	3	2	1	0
P46F	P46PUP	P46PDP	P46OPR	-	-	P46S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8027H	7	6	5	4	3	2	1	0
P47F	P47PUP	P47PDP	P47OPR	-	-	P47S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8028H	7	6	5	4	3	2	1	0
P50F	P50PUP	P50PDP	P50OPR	-	-	P50S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8029H	7	6	5	4	3	2	1	0
P51F	P51PUP	P51PDP	P51OPR	-	-	P51S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
802AH	7	6	5	4	3	2	1	0
P52F	P52PUP	P52PDP	P52OPR	-	-	P52S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
802BH	7	6	5	4	3	2	1	0
P53F	P53PUP	P53PDP	P53OPR	-	-	P53S		

R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
802CH	7	6	5	4	3	2	1	0
P54F	P54PUP	P54PDP	P54OPR	-	-	P54S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
802DH	7	6	5	4	3	2	1	0
P55F	P55PUP	P55PDP	P55OPR	-	-	P55S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
802EH	7	6	5	4	3	2	1	0
P56F	P56PUP	P56PDP	P56OPR	-	-	P56S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
802FH	7	6	5	4	3	2	1	0
P57F	P57PUP	P57PDP	P57OPR	-	-	P57S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8030H	7	6	5	4	3	2	1	0
P60F	P60PUP	P60PDP	P60OPR	-	-	P60S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8031H	7	6	5	4	3	2	1	0
P61F	P61PUP	P61PDP	P61OPR	-	-	P61S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8032H	7	6	5	4	3	2	1	0
P62F	P62PUP	P62PDP	P62OPR	-	-	P62S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8033H	7	6	5	4	3	2	1	0
P63F	P63PUP	P63PDP	P63OPR	-	-	P63S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8034H	7	6	5	4	3	2	1	0

P64F	P64PUP	P64PDP	P64OPR	-	-	P64S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8035H	7	6	5	4	3	2	1	0
P65F	P65PUP	P65PDP	P65OPR	-	-	P65S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8036H	7	6	5	4	3	2	1	0
P66F	P66PUP	P66PDP	P66OPR	-	-	P66S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8037H	7	6	5	4	3	2	1	0
P67F	P67PUP	P67PDP	P67OPR	-	-	P67S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8038H	7	6	5	4	3	2	1	0
P70F	P70PUP	P70PDP	P70OPR	-	-	P70S		
R/W	R/W	R/W	R/W	-	-	R/W		
Initial Value	0	0	0	-	-	0	0	0
8039H	7	6	5	4	3	2	1	0
P71F	P71PUP	P71PDP	P71OPR	-	-	-	P71S	
R/W	R/W	R/W	R/W	-	-	-	R/W	
Initial Value	0	0	0	-	-	-	0	0
803AH	7	6	5	4	3	2	1	0
P72F	P72PUP	P72PDP	P72OPR	-	-	-	P72S	
R/W	R/W	R/W	R/W	-	-	-	R/W	
Initial Value	0	0	0	-	-	-	0	0
803BH	7	6	5	4	3	2	1	0
P73F	P73PUP	P73PDP	P73OPR	-	-	-	P73S	
R/W	R/W	R/W	R/W	-	-	-	R/W	
Initial Value	0	0	0	-	-	-	0	0
803CH	7	6	5	4	3	2	1	0
P74F	P74PUP	P74PDP	P74OPR	-	-	-	P74S	
R/W	R/W	R/W	R/W	-	-	-	R/W	
Initial Value	0	0	0	-	-	-	0	0

803DH	7	6	5	4	3	2	1	0
P75F	P75PUP	P75PDP	P75OPR	-	-	-	P75S	
R/W	R/W	R/W	R/W	-	-	-	R/W	
Initial Value	0	0	0	-	-	-	0	0

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

Note：Pnx → n=0~7, stands for P0~P7
x=0~7, stands for Pn.0~Pn.7

Table 15-2-10 Pin Control Register PxnC

8120H	7	6	5	4	3	2	1	0
P00C	SINK_EN	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	R/W	R/W	R/W	R/W	R/W		R/W	
Initial Value	0	1	1	1	1	1	1	1
8121H	7	6	5	4	3	2	1	0
P01C	SINK_EN	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	R/W	R/W	R/W	R/W	R/W		R/W	
Initial Value	0	1	1	1	1	1	1	1
8122H	7	6	5	4	3	2	1	0
P02C	SINK_EN	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	R/W	R/W	R/W	R/W	R/W		R/W	
Initial Value	0	1	1	1	1	1	1	1
8123H	7	6	5	4	3	2	1	0
P03C	SINK_EN	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	R/W	R/W	R/W	R/W	R/W		R/W	
Initial Value	0	1	1	1	1	1	1	1
8124H	7	6	5	4	3	2	1	0
P04C	SINK_EN	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	R/W	R/W	R/W	R/W	R/W		R/W	
Initial Value	0	1	1	1	1	1	1	1
8125H	7	6	5	4	3	2	1	0
P05C	SINK_EN	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	R/W	R/W	R/W	R/W	R/W		R/W	
Initial Value	0	1	1	1	1	1	1	1
8126H	7	6	5	4	3	2	1	0
P06C	SINK_EN	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	R/W	R/W	R/W	R/W	R/W		R/W	
Initial Value	0	1	1	1	1	1	1	1

8127H	7	6	5	4	3	2	1	0
P07C	SINK_EN	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	R/W	R/W	R/W	R/W	R/W		R/W	
Initial Value	0	1	1	1	1	1	1	1

8128H	7	6	5	4	3	2	1	0
P10C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8129H	7	6	5	4	3	2	1	0
P11C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
812AH	7	6	5	4	3	2	1	0
P12C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
812BH	7	6	5	4	3	2	1	0
P13C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
812CH	7	6	5	4	3	2	1	0
P14C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
812DH	7	6	5	4	3	2	1	0
P15C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
812EH	7	6	5	4	3	2	1	0
P16C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
812FH	7	6	5	4	3	2	1	0
P17C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1

8130H	7	6	5	4	3	2	1	0
P20C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8131H	7	6	5	4	3	2	1	0
P21C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8132H	7	6	5	4	3	2	1	0
P22C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8133H	7	6	5	4	3	2	1	0
P23C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8134H	7	6	5	4	3	2	1	0
P24C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8135H	7	6	5	4	3	2	1	0
P25C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	

Initial Value	-	1	1	1	1	1	1	1
8136H	7	6	5	4	3	2	1	0
P26C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8137H	7	6	5	4	3	2	1	0
P27C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8138H	7	6	5	4	3	2	1	0
P30C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8139H	7	6	5	4	3	2	1	0
P31C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
813AH	7	6	5	4	3	2	1	0
P32C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
813BH	7	6	5	4	3	2	1	0
P33C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
813CH	7	6	5	4	3	2	1	0
P34C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
813DH	7	6	5	4	3	2	1	0
P35C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
813EH	7	6	5	4	3	2	1	0
P36C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
813FH	7	6	5	4	3	2	1	0
P37C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8140H	7	6	5	4	3	2	1	0
P40C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8141H	7	6	5	4	3	2	1	0
P41C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8142H	7	6	5	4	3	2	1	0
P42C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8143H	7	6	5	4	3	2	1	0
P43C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8144H	7	6	5	4	3	2	1	0
P44C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1

8145H	7	6	5	4	3	2	1	0
P45C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8146H	7	6	5	4	3	2	1	0
P46C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8147H	7	6	5	4	3	2	1	0
P47C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8148H	7	6	5	4	3	2	1	0
P50C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8149H	7	6	5	4	3	2	1	0
P51C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
814AH	7	6	5	4	3	2	1	0
P52C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
814BH	7	6	5	4	3	2	1	0
P53C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
814CH	7	6	5	4	3	2	1	0
P54C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
814DH	7	6	5	4	3	2	1	0
P55C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
814EH	7	6	5	4	3	2	1	0
P56C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
814FH	7	6	5	4	3	2	1	0
P57C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8150H	7	6	5	4	3	2	1	0
P60C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8151H	7	6	5	4	3	2	1	0
P61C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8152H	7	6	5	4	3	2	1	0
P62C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8153H	7	6	5	4	3	2	1	0
P63C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8154H	7	6	5	4	3	2	1	0

P64C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8155H	7	6	5	4	3	2	1	0
P65C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8156H	7	6	5	4	3	2	1	0
P66C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8157H	7	6	5	4	3	2	1	0
P67C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8158H	7	6	5	4	3	2	1	0
P70C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
8159H	7	6	5	4	3	2	1	0
P71C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
815AH	7	6	5	4	3	2	1	0
P72C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
815BH	7	6	5	4	3	2	1	0
P73C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
815CH	7	6	5	4	3	2	1	0
P74C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1
815DH	7	6	5	4	3	2	1	0
P75C	-	SMIT_EN	PU_SEL	PD_SEL	DRV[1:0]		SR[1:0]	
R/W	-	R/W	R/W	R/W	R/W		R/W	
Initial Value	-	1	1	1	1	1	1	1

Note：

1. *PxnC stands for P00C~P75C，x=0~7 stands for P0~P7，n=0,1,2,...,7(when X=7 时 n=0~5)。*
2. *The SINK_EN bit exists only in the control register of IO with LED COM capability.*

Bit number	Bit symbol	description
7	SINK_EN	High sink current enable, 1 enabled *Note: IO must be set to push-pull output mode*
6	SMIT_EN	IO indicates the mode selection bit for input functions 0：inverter mode 1：SMIT mode
5	PU_SEL	Pull-up resistor selection bit 0:weak pull-up(pull-up resistance is 45K) 1:strong pull-up(pull-up resistance is 10K)
4	PD_SEL	Pull-down resistor selection bit 0:weak pull-down(pull-down resistance is 45K) 1:strong pull-down (pull-down resistance is 10K)
3~2	DRV	Output strength selection bit, the range: 0 to 3, the greater the value, the stronger the driving ability
1~0	SR	Output slope control bit, range: 0~3, the larger the value, the higher the IO flip slope (faster)

Table 15-2-11 Pin Alternate Function Mapping

Value / Name	0	1	2	3	4	5	6	7
P00S	High impedance	Digital input	Digital output	High impedance	High impedance	High impedance	VP3	High impedance
P01S	High impedance	digital input	digital output	High impedance	High impedance	High impedance	VP2	High impedance
P02S	High impedance	digital input	digital output	High impedance	High impedance	High impedance	VP1	High impedance
P03S	High impedance	digital input	digital output	High impedance	High impedance	High impedance	CUP1	High impedance
P04S	High impedance	digital input	digital output	High impedance	High impedance	High impedance	CUP2	High impedance
P05S	High impedance	digital input	digital output	I2C_SCL	High impedance	TK15	LCD_S29	High impedance
P06S	High impedance	digital input	digital output	I2C_SDA	PWM1	TK0	TLCOM	High impedance
P07S	High impedance	digital input	digital output	SWIM	High impedance	High impedance	LCD_S30	High impedance
P10S	High impedance	digital input	digital output	High impedance	High impedance	TK_CAP	LCD_S0	High impedance
P11S	High impedance	digital input	digital output	High impedance	High impedance	TK1	LCD_S1	High impedance
P12S	High impedance	digital input	digital output	High impedance	High impedance	TK2	LCD_S2	High impedance
P13S	High impedance	digital input	digital output	High impedance	High impedance	TK3	LCD_S3	High impedance
P14S	High impedance	digital input	digital output	High impedance	High impedance	TK4	LCD_S4	High impedance
P15S	High impedance	digital input	digital output	High impedance	High impedance	TK5	LCD_S5	High impedance
P16S	High impedance	digital input	digital output	High impedance	High impedance	TK6	LCD_S6	High impedance
P17S	High impedance	digital input	digital output	High impedance	High impedance	TK7	LCD_S7	High impedance
P20S	High impedance	digital input	digital output	High impedance	High impedance	TK8	LCD_S8	High impedance
P21S	High impedance	digital input	digital output	UART1_RX	High impedance	TK9	LCD_S9	High impedance
P22S	High impedance	digital input	digital output	UART1_TX	High impedance	TK10	LCD_S10	High impedance
P23S	High impedance	digital input/T1	digital output	High impedance	High impedance	TK11	LCD_S11	High impedance
P24S	High impedance	digital input/T2	digital output	High impedance	High impedance	TK12	LCD_S12	High impedance
P25S	High impedance	digital input/T2EX	digital output	T2CP	High impedance	TK13	LCD_S13	High impedance
P26S	High impedance	digital input/T0	digital output	High impedance	High impedance	TK14	LCD_S14	High impedance
P27S	High impedance	digital input	digital output	ADC0	High impedance	High impedance	LCD_S15	High impedance
P30S	High impedance	digital input	digital output	High impedance	PWM2	CLK_IN	LCD_S16	High impedance
P31S	High impedance	digital input	digital output	ADC1	High impedance	High impedance	LCD_S17	High impedance
P32S	High impedance	digital input	digital output	ADC2	High impedance	High impedance	LCD_S18	High impedance
P33S	High impedance	digital input	digital output	ADC3	High impedance	High impedance	LCD_S19	High impedance
P34S	High	digital input	digital output	ADC4	High	High	LCD_S20	High

	impedance				impedance	impedance		impedance
P35S	High impedance	digital input	digital output	ADC5	High impedance	High impedance	LCD_S21	High impedance
P36S	High impedance	digital input	digital output	ADC6	High impedance	High impedance	LCD_S22	High impedance
P37S	High impedance	digital input	digital output	ADC7	ADC_VREF	High impedance	LCD_S23	High impedance
P40S	High impedance	digital input	digital output	High impedance	High impedance	High impedance	LCD_S24	High impedance
P41S	High impedance	digital input	digital output	High impedance	High impedance	High impedance	LCD_S25	High impedance
P42S	High impedance	digital input	digital output	High impedance	High impedance	High impedance	LCD_S26	High impedance
P43S	High impedance	digital input	digital output	High impedance	High impedance	High impedance	LCD_S27	High impedance
P44S	High impedance	digital input	digital output	High impedance	High impedance	High impedance	LCD_S28	High impedance
P45S	High impedance	digital input	digital output	High impedance	High impedance	LCD_S31	LCD_C4	High impedance
P46S	High impedance	digital input	digital output	High impedance	High impedance	High impedance	LCD_C3	High impedance
P47S	High impedance	digital input	digital output	High impedance	High impedance	High impedance	LCD_C2	High impedance
P50S	High impedance	digital input	digital output	High impedance	High impedance	High impedance	LCD_C1	High impedance
P51S	High impedance	digital input	digital output	High impedance	High impedance	High impedance	LCD_C0	High impedance
P52S	High impedance	digital input	digital output	32K_O	High impedance	High impedance	High impedance	High impedance
P53S	High impedance	digital input	digital output	32K_I	High impedance	High impedance	High impedance	High impedance
P54S	High impedance	digital input	digital output	High impedance	PWM0/REM	High impedance	High impedance	High impedance
P55S	High impedance	digital input	digital output	RESET	High impedance	High impedance	High impedance	High impedance

15.3 Pin control Example

◆ **Set the Pin function**

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

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

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

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

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

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

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

```
-----------------------------------------------------------------------------------------
P20F = (1<<7) | 1;
-----------------------------------------------------------------------------------------
```

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

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

16 Sampling Counter(SAMPLE)

16.1 Function Introduction

The sampling counter uses pin SAMPLE to take samples. The input pulse must continue for at least 5 sampling clock cycle time other wise it can not be detected. The counter can be triggered by either rising or falling edge or both edges. There multiple clock sources for the sampling counter. The Figure 16-1-1 shows the architecture.

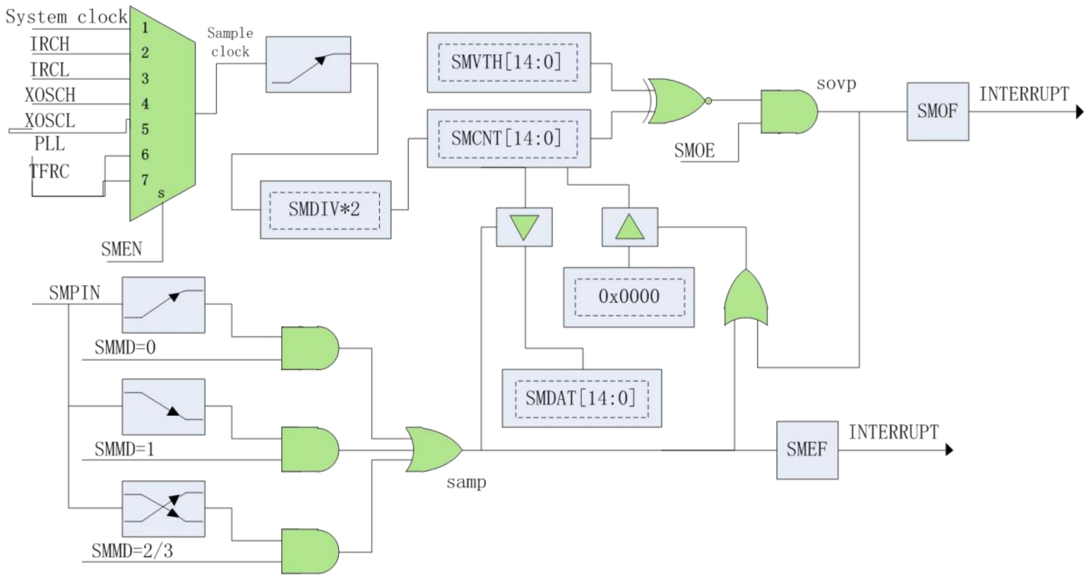


Figure 16-1-1 Sampling Counter Architecture

Uses can set the frequency division of the sampling clock by register SMDIV. The more the frequency is divided by, the larger pulse width can be sampled at one time. However, it will also worsen the accuracy. The sample pulse width threshold can be set by register SMVTHL and SMVTHH. When the count reach the threshold, it overflows and the counter will return to 0 and starts the count again. The interrupt flag SMOF will be generated at the same time. There are two reasons for the design: first, when the sample pulse width is too large and exceeds the counter's count circumscription, the software can still calculate the pulse width by using the number of the times it overflows; secondly, the threshold can be used to identify efficient pulse. Users may set maximum of the efficient pulse width as the threshold and once the there occurs overflow, the software should reset the program so that the next efficient pulse can be detected successfully.

After the counter is enabled, every time it detects the efficient edges, it will reset to 0 and starts counting again.The first efficient edge detected by the counter will not generate any interrupts and there will be no overflow interrupt before it either. Only second or other following efficient edges detected will generate the interrupt with interrupt flag SMEF and the count will be stored into register SMDATL and SMDATH. Software can compute the pulse width using the count.

The sampling counter can be used to realize infrared remote receiving and etc, which saves software codes and makes it convenient for software development.

16.2 SAMPLE Function Register Description

Table 16-2-1 Register SMCON

8078H	7	6	5	4	3	2	1	0
SMCON	SMEN[2:0]			SMIE	SMOE	-	SMMD[1:0]	
R/W	R/W			R/W	R/W	-	R/W	
Initial Value	0	0	0	0	0	-	0	0

Bit number	Bit Symbol	Description
7~5	SMEN	SAMPLE clock selection 000：disable 001： system clock 010： IRCH 011：IRCL 100：XOSCH 101：XOSCL 110：PLL 111：TFRC
4	SMIE	SAMPLE interrupt enable control, 1 enables it
3	SMOE	SAMPLE overflow threshold enable control 0：the threshold invalid, counter overflows when reaches 7FFFH, no interrupt for when it overflows 1：the threshold valid, there will an interrupt generated when it overflows
2	-	-
1~0	SMMD	Sampling edge selection 00：sample rising edges 01：sample falling edges Others：sample both rising and falling edges

Table 16-2-2 Register SMSTA

8079H	7	6	5	4	3	2	1	0
SMSTA	-	-	-	-	-	-	SMEF	SMOF
R/W	-	-	-	-	-	-	R	R
Initial Value	-	-	-	-	-	-	0	0

Bit number	Bit Symbol	Description
7~2	-	-
1	SMEF	SAMPLE edge interrupt flag, 1 indicates the interrupt, cleared by writing 1 to it
0	SMOF	SAMPLE overflow interrupt flag, 1 indicates the interrupt, cleared by writing 1 to it

Table 16-2-3 Register SMDIV

807AH	7	6	5	4	3	2	1	0
SMDIV	SMDIV[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~0	SMIDV	Sampling clock frequency divider, the coefficient for it is SMDIV * 2

Table 16-2-4 Register SMDAT

807BH	7	6	5	4	3	2	1	0
SMDATL	SMDAT[7:0]							
R/W	R							
Initial Value	0	0	0	0	0	0	0	0
807CH	7	6	5	4	3	2	1	0
SMDATH	LVBIT	SMDAT[14:8]						
R/W	R	R						
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
15(SMDATH.7)	LVBIT	Current trigger edge flag 0：falling edge 1：rising edge
14~0	SMDAT	SAMPLE counter register which stores the count value

Table 16-2-5 Register SMVTHL, SMVTHH

807DH	7	6	5	4	3	2	1	0
SMVTHL	SMVTH[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
807EH	7	6	5	4	3	2	1	0
SMVTHH	-	SMVTH[14:8]						
R/W	-	R/W						

Initial Value	-	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
15	-	-
14~0	SMVTH	The overflow threshold setting register for the counter

16.3 SAMPLE Control Example

Infrared remote realization by SAMPLE function, the program is like：

```
-------------------------------------------------------------------------------------------------------------------------------------------------
#define     SMEN_SYS_CLK     (1<<5)
#define     SMEF     (1<<1)
#define     SMOF     (1<<0)

#define SYN_WIDTH1          0x103b              //pulse width synchronization
#define DAT_1_WIDTH         0x02b1              //pulse width of bit 1
#define DAT_0_WIDTH         0x015c              //pulse width of bit 0
#define WIDTH               (DAT_0_WIDTH/6)

unsigned char OverFlowCount;                    //overflow counter
unsigned char IR_BitCount;                      //IR codes bit indicator
unsigned char IR_Code[4];                       //IR codes
bit IR_SyncFlag;                               //synchronization signal flag, synchronization signal received when it is 1
bit IR_RxEndFlag;

void Sample_init(void)
{
P70F = 3;                                                    //set P70 as SAMPLE function pin
     SMCON = SMEN_SYS_CLK | SMOE(1) | SMMD(1);               //initialize the SAMPLE function
     SMDIV = 6;                                              //set the frequency division for SAMPLE clock
     SMVTHL = (SYN_WIDTH1*2)%256;                            //set the overflow clock width
     SMVTHH = (SYN_WIDTH1*2)/256;
     SMCON |= SMIE(1);                                       //enables SAMPLE interrupt
     INT9EN = 1;                                             //enables INT9
}
void INT9_ISR (void) interrupt 14
{
     unsigned int PulseWidth;
     if(SMSTA & SMEF)
     {
          SMSTA |= SMEF;
          if(OverFlowCount == 0)
          {
               PulseWidth = (SMDATH&0x7F)*256 + SMDATL;      //get pulse width
```

```
                    if(!IR_SyncFlag)
                    {
                        if((PulseWidth > (SYN_WIDTH1-WIDTH*6)) && (PulseWidth < (SYN_WIDTH1+WIDTH*6)))
                                                        //if the it is not synchronized, judge whether the
current pulse is synchronization signal
                        {
                            IR_SyncFlag = 1;
                            IR_BitCount = 0;
                        }
                    }
                    else
                    {
                        if((PulseWidth > (DAT_1_WIDTH-WIDTH*2)) && (PulseWidth < (DAT_1_WIDTH+WIDTH*2)))
//judge whether current pulse is bit 1
                        {
                            IR_Code[IR_BitCount/8] |= (1<<(7-(IR_BitCount%8)));
                            IR_BitCount++;
                        }
                        else if((PulseWidth > (DAT_0_WIDTH-WIDTH)) && (PulseWidth < (DAT_0_WIDTH+WIDTH)))
//judge whether current pulse is bit 0
                        {
                            IR_Code[IR_BitCount/8] &= ~(1<<(7-(IR_BitCount%8)));
                            IR_BitCount++;
                        }
                        else
                        {
                            IR_SyncFlag = 0;          //if it is neither bit 1 nor bit 0, then wait for the next signal
                }
                        if((IR_BitCount == 32) && IR_SyncFlag)     //32 bit IR code received
                        {
    //read the IR code here
                            IR_SyncFlag = 0;
                            IR_RxEndFlag = 1;
                        }
                    }
                }
                OverFlowCount=0;                  //reset the overflow count
            }
            if(SMSTA & SMOF)
            {
                SMSTA |= SMOF;
                if(OverFlowCount < 0xFF)
                {
                    OverFlowCount++;          //overflow counter adds up
                }
```

```
}
        }
        void main(void)
        {
                Sample_init();
        IR_SyncFlag = 0;
                IR_RxEndFlag = 0;
                        OverFlowCount=0;
                        EA = 1;
                        while(1)
                        {
                                if(IR_RxEndFlag)
                                {
                                        IR_RxEndFlag = 0;
                                }
                        }
                }
```

----

17 Universal Asynchronous Receiver/Transmitter(UART)

17.1 UART0

17.1.1 Function Introduction

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

There are four working modes for UART0 which is shown as the table 17-1-1-1.

Table 17-1-1-1 UART0 Communication Mode

SM00	SM10	Mode	Description	Baud rate
0	0	0	Synchronous shift mode	Fclk/12
0	1	1	8 bit asynchronous mode	The baud rate is (2^ SMOD)*CPUCLK*(overflow rate of Timer1/2)/32, please refer to UCKS of T2CON
1	0	2	9 bit asynchronous mode	When SMOD＝0, the baud rate is Fclk/64 When SMOD＝1, the baud rate is Fclk/32
1	1	3	9 bit asynchronous mode	The baud rate is (2^ SMOD)*CPUCLK*(overflow rate of Timer1/2)/32, please refer to UCKS of T2CON

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

- **Mode0**

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

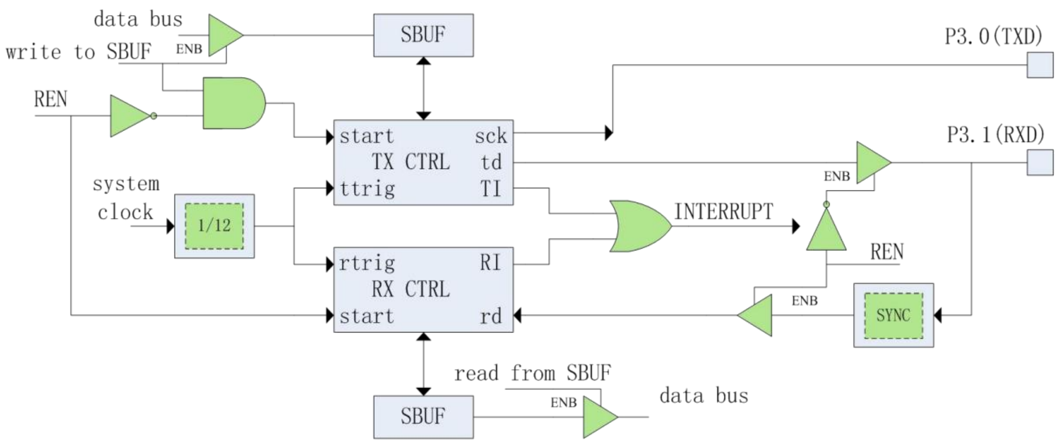


Figure 17-1-1-1 UART0 Mode0 Schematic

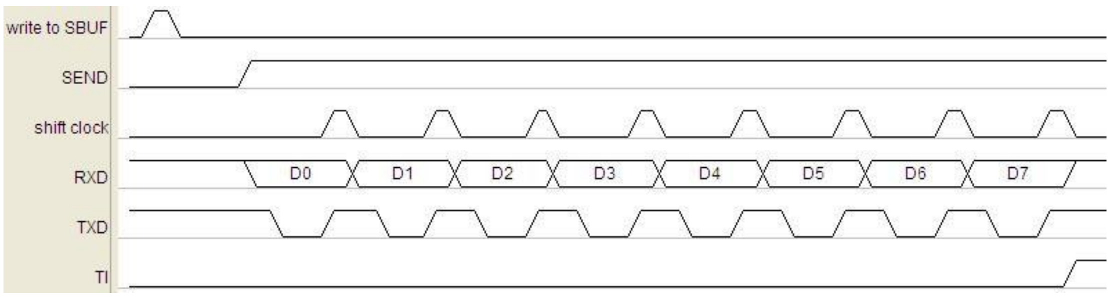


Figure 17-1-1-2 UART0 Data Transmission Waveform in Mode0

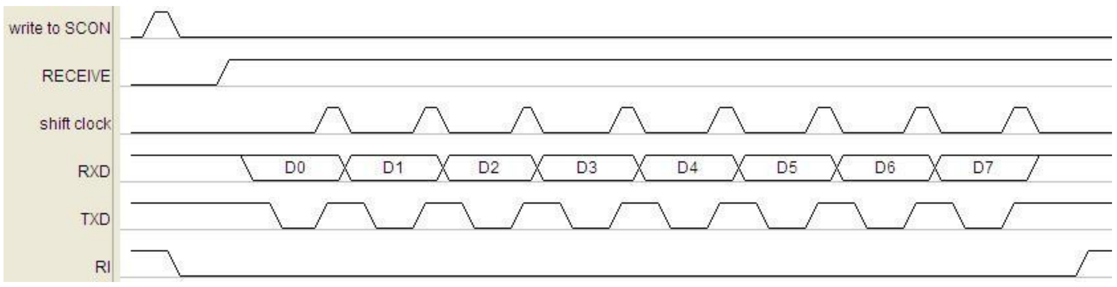


Figure 17-1-1-3 UART0 Data Receiving Waveform in Mode0

- **Mode1**

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

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

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

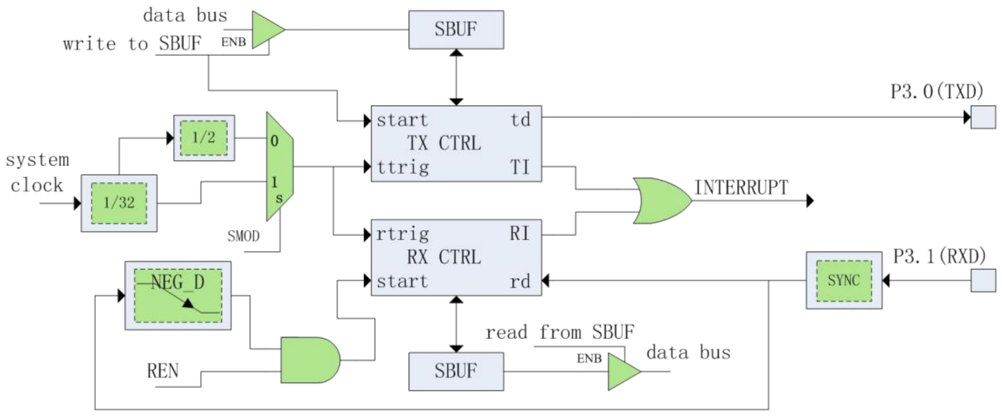


Figure 17-1-1-4 UART0 Mode1 Schematic

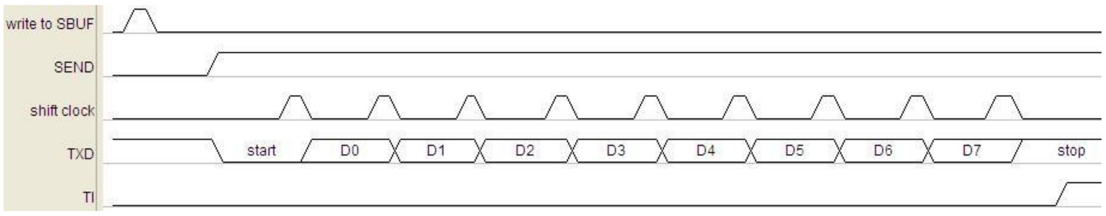


Figure 17-1-1-5 UART0 Data Transmission Waveform in Mode1

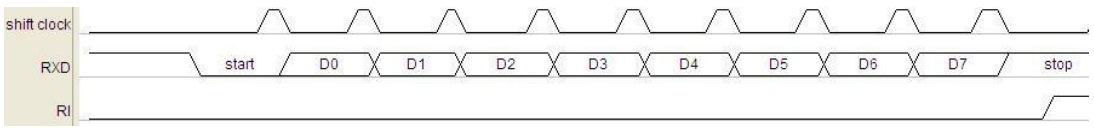


Figure 17-1-1-6 UART0 Data Receiving Waveform in Mode1

- **Mode2**

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

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

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

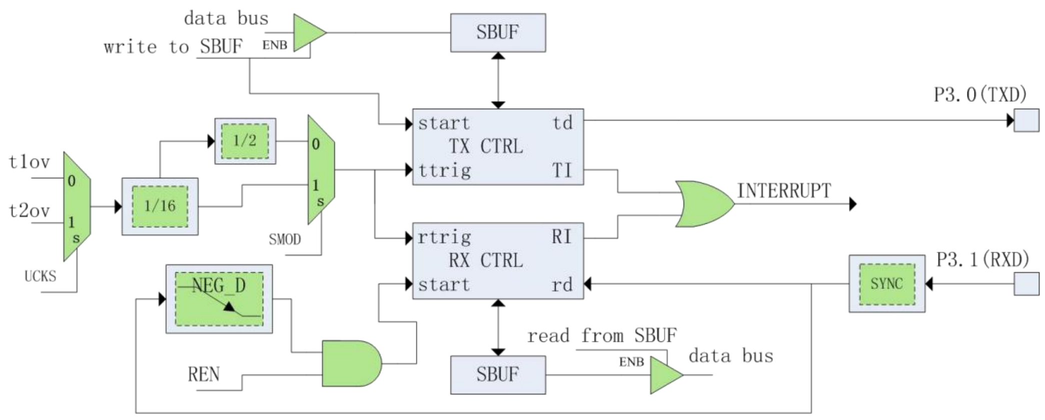


Figure 17-1-1-7 UART0 Mode2 Schematic

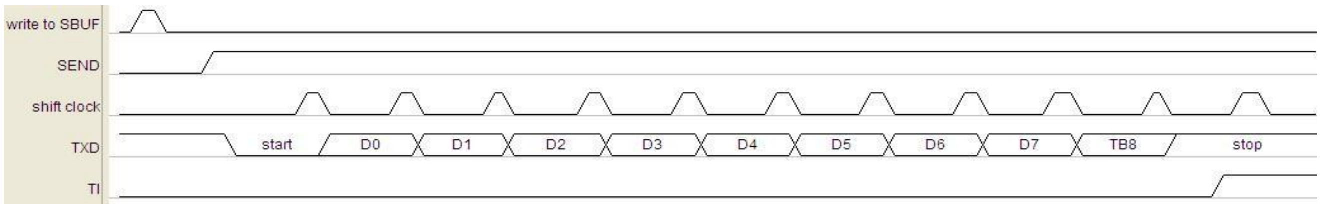


Figure 17-1-1-8 UART0 Data Transmission Waveform in Mode2

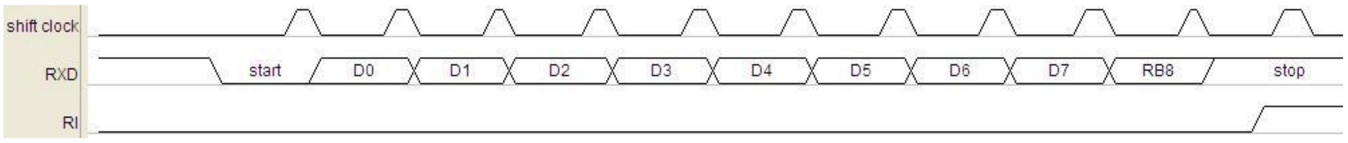


Figure 17-1-1-9 UART0 Data Receiving Waveform in Mode2

- **Mode3**

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

- **UART0 Multi-computer Communication**

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

17.1.2 Register Description

Table 17-1-2-1 Register S0CON

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

Bit number	Bit Symbol	Description
7	SM00	UART0 mode selection, for more information please refer to Table 17-1-1-1
6	SM10	
5	SM20	Multi-computer communication enable control, 1 enables
4	REN0	Serial receive enable control, 1 enables
3	TB80	The 9th data bit to transmit It will be transmitted as the 9th bit of the data in mode 2 and mode 3 and it is controlled by the software. This bit is used for UART0 to send data. (For instance, parity check or multi-computer communication)
2	RB80	The 9th bit of the data received It will be received as the 9th bit of the data in mode 2 and mode 3. This bit is used for UART0 to receive data. It is the stop bit in mode1; if SM2=1, it is the token bit for multi-host； it is not used in mode0.
1	TI0	Transmit interrupt flag, 1 indicates the interrupt, cleared by writing 0 to it
0	RI0	Receive interrupt flag, 1 indicates the interrupt, cleared by writing 0 to it

Table 17-1-2-2 Register S0BUF

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

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

17.2 UART1 and UART2

17.2.1 Introduction

UART1 and UART2 are two full duplex asynchronous serial data transceivers with the same design. UARTx(x=1 or 2, indicating UART1 or UART2) includes one byte buffer as well. There are two operating modes for UARTx as Table 17-2-1-1 shows.

Table 17-2-1-1 UARTx Operating Modes

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

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

Note：Whether UARTx baud rate will be doubled cannot be set by SMOD in register PCON.

Figure 17-2-1-1 shows the UARTx principal schematic.

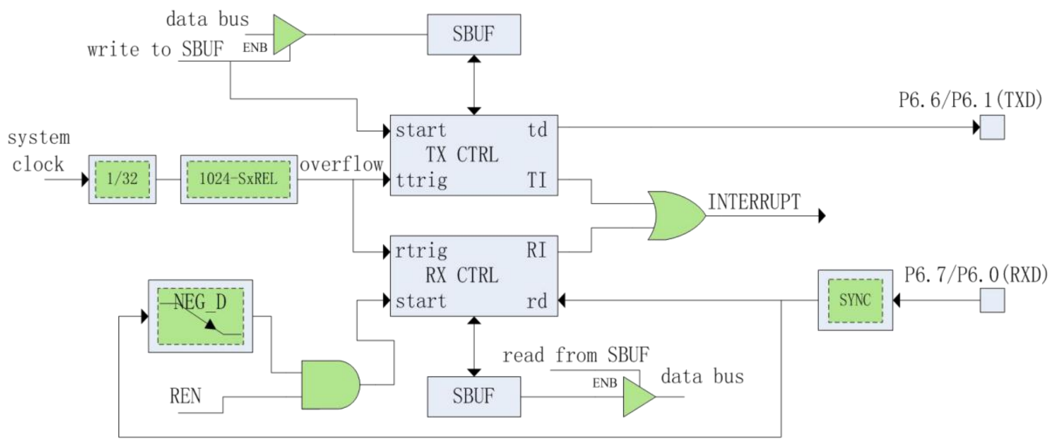


Figure 17-2-1-1 UARTx Principal Schematic

- **ModeA**

UARTx can transmit/receive 9-bit data asynchronously in ModeA. Writing data to register SxBUF will starts UARTx transmission. The first bit transmitted is the start bit (which is 0), and then the 9-bit data follows (with the least significant bit transmitted first). The ninth data bit is the TB8x of register SxCON. The last bit transmitted is the stop bit (which is 1). When UART0 is used as receiver, it is synchronized by detecting the falling edges of Pin RX. The lower 8-bit data will be stored in register SxBUF after the transmission is completed with the 9th data bit stored in RB8x.

- **ModeB**

Mode B differs from Mode A in that Mode B is an 8-bit data transfer, and the stop bit holds an efficient stop bit. Other functions are the same as Mode A.

- **UARTx Multi-computer Communication**

Multi-computer Communication can also be realized by UARTx in ModeA. If SM2x of register SxCON is set to 1, only when the 9th data is 1 (RB8x=1) the slave will generate receive interrupt, which makes multi-computer communication possible. The slaves can set their SM2x to 1 and host set the 9th data bit to 1 when it transfers address to the slaves. All the slaves will generate receive interrupt and the slaves' software then compare the address received to their own addresses. It the address matches, the matched slave will set SM2x=0. The host then set the 9th data bit to 0 for the following data transmission. Due to the other slaves remain SM2x = 1, thus only the address matched slave will generate receive interrupt.

17.2.2 UARTx Register Description

Table 17-2-2-1 Register S1CON

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

Bit number	Bit Symbol	Description
7	SM1	UART1 mode selection, for more information please refer to Table 17-2-1-1
6	U1IE	UART1 interrupt enable control, 1 enables it
5	SM21	Multi-computer communication enable control, 1 enables it
4	REN1	Serial receive enable control, 1 enables it
3	TB81	The 9th data bit to transmit It will be used for UART1 to receive the 9th bit of the data in modeA. Which is controlled by software. (For instance, parity check or multi-computer communication)
2	RB81	The 9th bit of the data received It will be used for UART1 to receive the 9th bit of the data in modeA. It is the stop bit received in modeB.
1	TI1	Transmit interrupt flag, 1 indicates the interrupt, cleared by writing 1 to it
0	RI1	Receive interrupt flag, 1 indicates the interrupt, cleared by writing 1 to it

Table 17-2-2-2 Register S1BUF

9BH	7	6	5	4	3	2	1	0
S1BUF	S1BUF[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~0	S1BUF	Receiver/Transmitter buffer for UART1 Writing data to S1BUF will starts the written data transmission Reading S1BUF will reads the data received

Table 17-2-2-3 Register S1RELL, S1RELH

9CH	7	6	5	4	3	2	1	0
S1RELL	S1RELL[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
9DH	7	6	5	4	3	2	1	0
S1RELH	-	-	-	-	-	-	S1REL[9:8]	
R/W	-	-	-	-	-	-	R/W	
Initial Value	-	-	-	-	-	-	0	0

Bit number	Bit Symbol	Description
9~0	S1REL	Baud rate configuration register The baud rate is CPUCLK/ (32 * (1024 - S1REL))

Table 17-2-2-4 Register S2CON

A1H	7	6	5	4	3	2	1	0
S2CON	SM2	U2IE	SM22	REN2	TB82	RB82	TI2	RI2
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	SM2	UART2 mode selection, for more information please refer to Table 17-2-1-1
6	U2IE	UART2 interrupt enable control, 1 enables it
5	SM22	Multi-computer communication enable control, 1 enables it
4	REN2	Serial receive enable control, 1 enables it
3	TB82	The 9th data bit to transmit It will be used for UART2 to receive the 9th bit of the data in modeA. Which is controlled by software.

		(For instance, parity check or multi-computer communication)
2	RB82	The 9th bit of the data received It will be used for UART2 to receive the 9th bit of the data in modeA. It is the stop bit received in modeB.
1	TI2	Transmit interrupt flag, 1 indicates the interrupt, cleared by writing 1 to it
0	RI2	Receive interrupt flag, 1 indicates the interrupt, cleared by writing 1 to it

Table 17-2-2-5 Register S2BUF

A2H	7	6	5	4	3	2	1	0
S2BUF	S2BUF[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
7~0	S2BUF	Receiver/Transmitter buffer Writing data to S2BUF will starts the written data transmission Reading S2BUF will reads the data received						

Table 17-2-2-6 Register S2REL

A3H	7	6	5	4	3	2	1	0
S2RELL	S2RELL[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
A4H	7	6	5	4	3	2	1	0
S2RELH	-	-	-	-	-	-	S2REL[9:8]	
R/W	-	-	-	-	-	-	R/W	
Initial Value	-	-	-	-	-	-	0	0
Bit number	Bit Symbol	Description						
9~0	S2REL	Baud rate configuration register The baud rate is CPUCLK/(32 * (1024 - S2REL))						

18 SPI

18.1 Function Introduction

SPI enables the chip to support half/full duplex synchronous data transmission with other devices. The peripheral devices could be other MCU, ADC, sensor or FLASH and etc. SPI nus can be 3 or 4 wires and the features are as follows.

- Both Master and Slave mode supported
- The transmission can either starts from the least or most significant bit
- 4 programmable bit rates
- Programmable polarity and phase
- Sends end interrupt flag
- Write conflict flag for protection
- Support error interrupt in master mode

Figure 18-1-1 and Figure 18-1-2 are the principle schematics for SPI Master Mode and Slave Mode respectively.

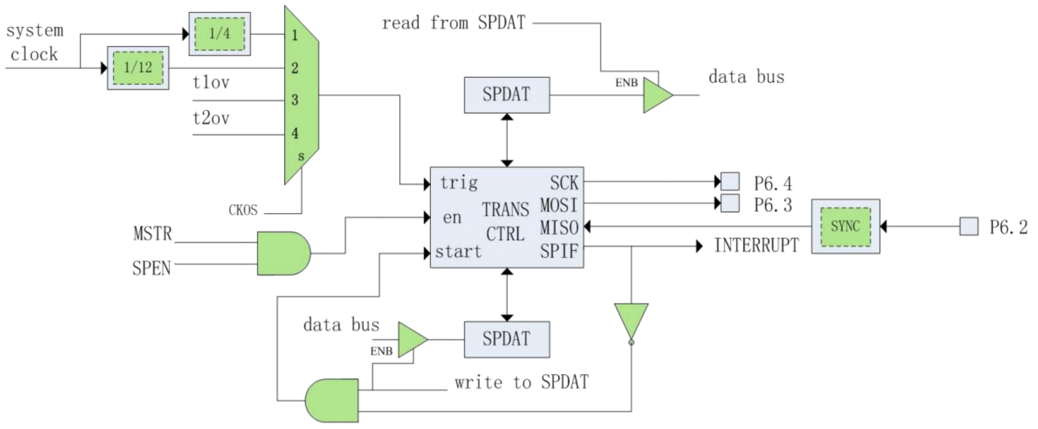


Figure 18-1-1 SPI Master Mode Schematic

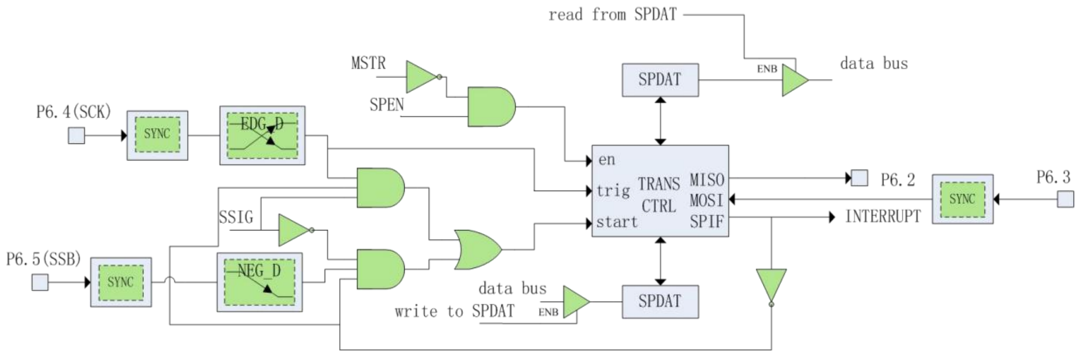


Figure 18-1-2 SPI Slave Mode Schematic

The operating modes for SPI is shown by the following table.

Table 18-1-1 SPI Operation Mode

Mode	Description
Master mode	All the transmission behavior can will be originated by the master, including SCK and SSB signal generation. When MSTR(SPCON[4]) is 1, SPI operates in Master mode. Users must select another pin as the chip select pin and connects slave's. SSB level will be pulled down before the data transmission and pulled up after the transmission. Writing register SPDAT will starts the data transmission in Master mode. The data will be shifted and output when clock's trigger edge comes.
Slave mode	When MSTR is set to 0, SPI operates in Slave mode When SSIG(SPCON[5])=1, SSB is invalid and there are only 3 wires for SPI communication and the default chip select is valid； when SSIG=0, SSB is valid and low level SSB implies that the slave is selected.

Table 18-1-2 SPI Port/Pin Description

Name	Description
MOSI	Master output, slave input This pin is the master's data output port when SPI operates as master; it is the slave's data input port when SPI operates as slave
MISO	Master input, slave output This pin is the master's data input port when SPI operates as master; it is the slave's data output port when SPI operates as slave
SCK	Serial clock This pin is the serial clock output port when SPI operates as master; it is the serial clock input port when SPI operates as slave
SSB	Slave selection This pin is the master's slave select input port when SPI operates as master; it is the slave select input port when SPI operates as slave

Table 18-1-3 SPI Phase and Polarity

Name	Description
CPHA	Phase control 0：SCK take data samples when odd edges(1,3,5,...,15)come 1：SCK take data samples when even edges(2,4,6,...,16)come
CPOL	Polarity control 0：SCK level is low when it is idle 1：SCK level is high when it is idle

Referring to Table18-1-3, the real waveform during the transmission is as Figure18-1-3 and 18-1-4 shows.

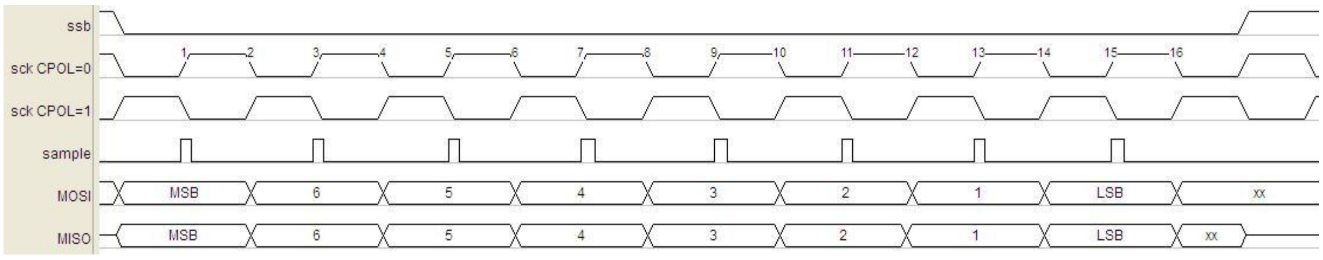


Figure 18-1-3 Sequence Diagram when CPHA=0

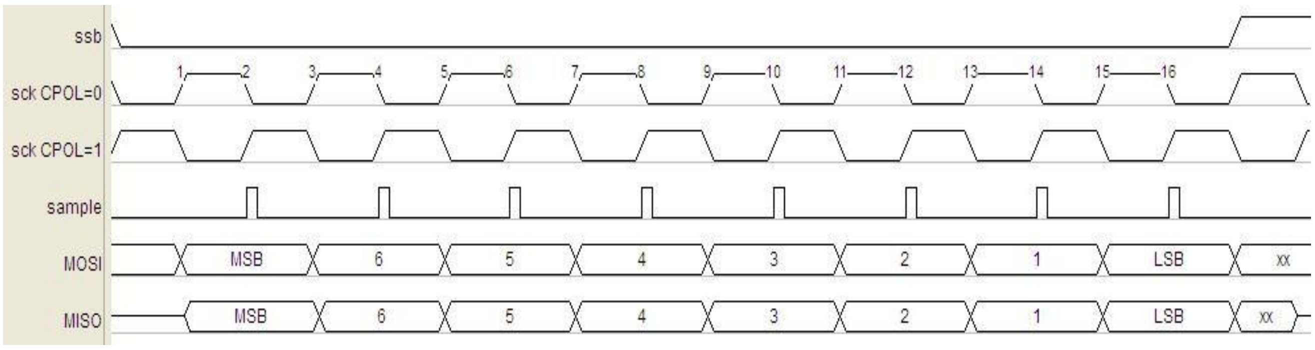


Figure 18-1-4 Sequence Diagram when CPHA=1

18.2 Register Description

Table 18-2-1 Register SPCON

A5H	7	6	5	4	3	2	1	0
SPCON	SPEN	LSBF	SSIG	MSTR	CPOL	CPHA	CKOS[1:0]	
R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7	SPEN	SPI module enable control, 1 enables it
6	LSBF	Whether receive/transmit the most/least significant bit first selection 0： receive/transmit the most significant bit first 1： receive/transmit the least significant bit first
5	SSIG	SSB pin disable control, the default value is 0 which indicates SSB signal is valid
4	MSTR	Master/Slave selection 0：Slave 1：Master
3	CPOL	Clock polarity selection 0：the clock signal is low by default 1：the clock signal is high by default

2	CPHA	Clock phase selection 0：take samples when the clock leaves idle state 1：take samples when the clock returns to idle state
1~0	CKOS	SPI output clock selection 00：1/8 system clock 01：1/24 system clock 10：Timer1 overflow flag used, transfer data every 2 overflows 11：Timer2 overflow flag used, transfer data every 2 overflows

Table 18-2-2 Register SPDAT

A6H	7	6	5	4	3	2	1	0
SPDAT	RBUF[7:0]							
R/W	R							
Initial Value	0	0	0	0	0	0	0	0
SPDAT	TBUF[7:0]							
R/W	W							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~0	SPDAT	Writing SPDAT will write to TBUF while reading SPDAT will read from RBUF

Table 18-2-3 Register SPSTA

A7H	7	6	5	4	3	2	1	0
SPSTA	SPIE	-	-	-	-	WCOL	MODF	SPIF
R/W	R/W	-	-	-	-	R/W	R/W	R/W
Initial Value	0	-	-	-	-	0	0	0

Bit number	Bit Symbol	Description
7	SPIE	SPI interrupt enable control, 1 enables it
6~3	-	-
2	WCOL	Write conflict flag. During the data transmission, if software writes SPDAT while the data cannot be written in, then there will be a write conflict flag. 1 indicates the conflict and it will be cleared by writing 1 to it. No interrupts when it is valid.
1	MODF	Fail mode flag with 1valid, it indicates that SSB logical level is not correct. It will be cleared by writing 1 to it. There will be interrupt when it is 1.
0	SPIF	Data transmission complete flag, 1 indicates the data transmission is over and it will be cleared by writing 1 to it. There will be interrupt when it is 1.

18.3 SPI Control Example

◆ **SPI master example**

As the Master, SPI sends 10-byte data to the Slave, the program is like：

```
-----------------------------------------------------------------------------------------------------------------------------------------------------------
#define SPEN(N)          (N<<7)
#define LSBF(N)          (N<<6)
#define SSIG(N)          (N<<5)
#define MSTR(N)          (N<<4)
#define CPOL(N)          (N<<3)
#define CPHA(N)          (N<<2)
#define CPOS(N)          (N)       //3:set system clock as the overflow of Timer2
                                   //2:set system clock as the overflow of Timer1
                                   //1:set system clock as its 1/8
                                   //0:set system clock as its 1/4

//define SPSTA
#define SPIE          (1<<7)    //SPI interrupt enable
#define WCOL          (1<<2)    //write conflict flag
#define MODF          (1<<1)    //fail mode
#define SPIF          (1<<0)    //transmission complete
unsigned char txIndex;
unsigned char txLength;
unsigned char txBuf[10]={0,1,2,3,4,5,6,7,8,9};
void SPI_init(void)
{
      P64F = 4;//set P64 as SPI SCK
      P62F = 4;//set P62 as SPI MISO
      P63F = 4;//set P63 as SPI MOSI
      P65F = 2;//set P65 as output, which is the chip select pin for SPI

      P6 |= 1<<5;      //set the chip select pin to high
      SPCON = SPEN(1) | LSBF(0) | SSIG(1) | MSTR(1) | CPOL(0) | CPHA(0) |CPOS(1);            //send  most
significant bit first and SSB valid, master mode with 1/4 system clock
      SPSTA |= SPIE;           //enables SPI interrupt
      INT5EN = 1;          //enables INT5 interrupt
}
void INT5_ISR (void) interrupt 10
{
        if(SPSTA & SPIF)                                //SPI interrupt
        {
                SPSTA |= SPIF;          //clear the SPI interrupt flag
                txIndex++ ;
                If(txIndex >= txLength)
                {
                        P6 |= 1<<5;     //data transmission completes, chip select pin set to high
                }
                else
                {
                        SPDAT = txBuf[txIndex];         //send the next one-byte data
                }
        }
        if(SPSTA&MODF)
        {
          SPSTA|=MODF;
        }
}
void main(void)
{
        SPI_init();
        EA = 1;
        txIndex = 0;
        txLength = 0;
        SPDAT = txBuf[txIndex];         //write SPDAT to start data transmission
        while(1)
        {
        }
}
```

- **SPI slave example**

As the slave, SPI receives data from the master, the program is like：

```
unsigned char rxIndex;
unsigned char txLength;
unsigned char rxBuf[10];
void SPI_init(void)
{
      P64F = 4;//set P64 as SPI SCK
      P62F = 4;//set P62 as SPI MISO
      P63F = 4;//set P63 as SPI MOSI
      P65F = 4;//set P65 as SPI SSB

        P6 |= 1<<5;      //set the chip select pin to high
        SPCON = SPEN(1) | LSBF(0) | SSIG(1) | MSTR(0) | CPOL(0) | CPHA(0) |CPOS(1);              //send most
significant bit first and SSB valid, slave mode with 1/4 system clock
        SPSTA |= SPIE;          //enables SPI interrupt
        INT5EN = 1;         //enables INT5 interrupt
    }
    void INT5_ISR (void) interrupt 10
    {
        if(SPSTA & SPIF)                        //SPI interrupt
        {
            SPSTA |= SPIF;      //clear SPI interrupt flag
            rxBuf[rxIndex++] = SPDAT；//receive one byte data
        }
        if(SPSTA&MODF)
        {
         SPSTA|=MODF;
        }
    }
    void main(void)
    {
        SPI_init();
        EA = 1;
        rxIndex= 0;
        while(1)
        {
        }
```

19 I²C Interface

19.1 Function Introduction

I^2C modules enables the chip to communicate with peripheral I^2C devices by serial transmission standard which complies with standard I^2C specification. It can be set to either slave or master and configured to standard/fast/high speed mode.

19.2 I^2C 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

19.3 I^2C Function Description

I^2C modules supports I^2C standard bus specification. I^2C bus includes 2 wires to transfer data among devices, one is SCL(Serial Clock) and the other is SDA(Serial Data), as Figure 19-3-1 shows. Since the it is open-drain port for I^2C, there must be pull-up resistor on I^2C 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.

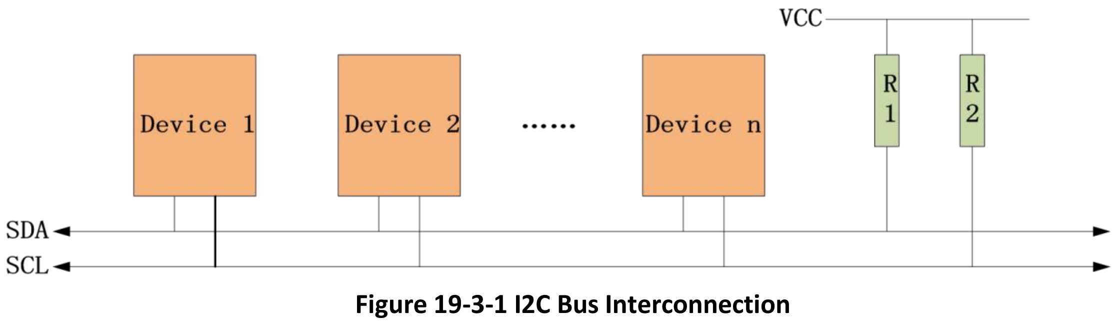


Figure 19-3-1 I2C Bus Interconnection

I^2C module principle is as Figure 19-3-2 shows.

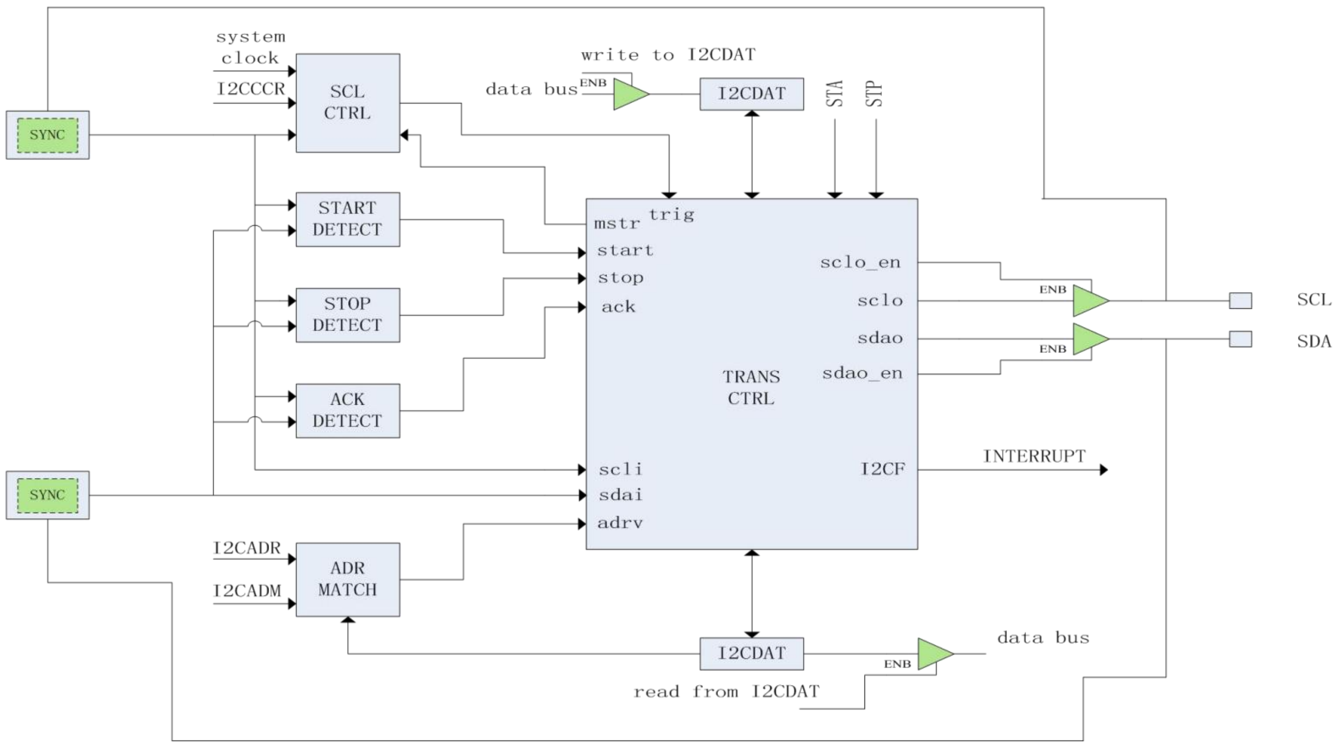


Figure 19-3-2 I²C Module Schematic

- **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.

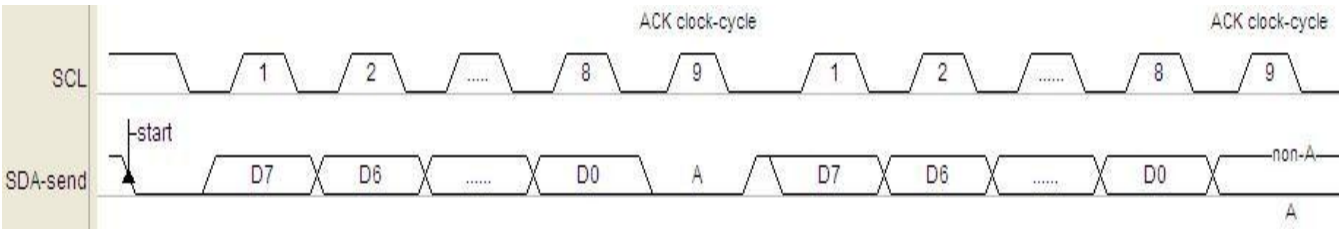


Figure 19-3-3 I2C Bus Data Transmission Format

- **Communication Process**

I²C enables the data transmission and generates the clock signal in Master mode. The serial data transmission always begins with START signal and ends with STOP signal. Both START and STOP signals are generated by the software in master mode. Setting STA=1 generates START signal and setting STP=1 generates STOP signal,

I²C can distinguish its address (7 bits) and the broadcast address in slave mode. Software can enable/disable its ability to recognize broadcast address by setting GCE.

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

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

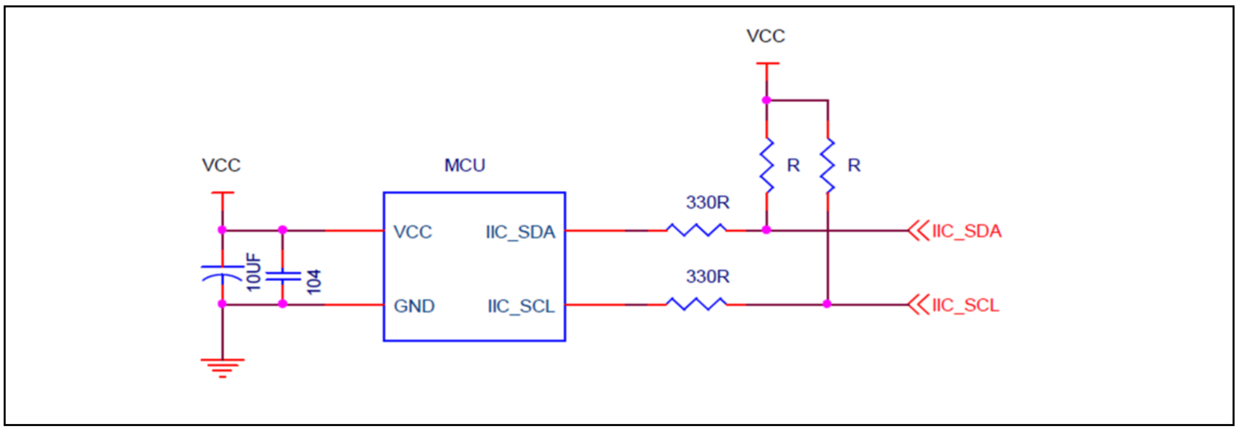
When there occurs interrupt I2CF, if SHD=1, SCL will be set to low by slave before I2CF is cleared. After the master detects that SCL is released, it master will then continue the next operation. On the other hand, if SHD=0, SCL will not be set to low by the slave, which makes it compatible with applications when the master I²C is simulated by software. Thus, the master's software must wait long enough so that the slave can deal with the response to every one byte data.

- **Clock Settings**

When I2C is set as slave, the master outputs SCL clock, and it has nothing to do with the slave's clock configuration. As a slave, the sampling clock of I2C is set by SMPDIV(I2CCCR[7:5]), and the filtering function is automatically activated when SMPDIV is not 0. As a master, the output clock frequency of SCL is determined by SMPDIV and I2CCKD(I2CCCR[4:0]) (please check the introduction of register section for details)..

19.4 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.



Note: The above component parameters are for reference only.

Figure 19-4-1 IIC Reference Circuit Diagrams

19.5 Register Description

Table 19-5-1 Register I2CCON

B1H	7	6	5	4	3	2	1	0
I2CCON	I2CE	I2CIE	STA	STP	SHD	AAK	CBSE	STFE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0
Bit number	Bit Symbol	Description						
7	I2CE	I^2C module enable control, 1 enables it						

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

Table 19-5-2 Register I2CADR

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

Bit number	Bit Symbol	Description
7	GCE	Broadcast address recognition(00H)enable control, 1 enables it
6~0	I2CADR	I^{2}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 19-5-3 Register I2CADM

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

Bit number	Bit Symbol	Description
7	SPFE	When SPFE=1, I2CF will be set to 1 if I^{2}C module detects the STOP signal
6~0	I2CADM	I^{2}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 19-5-4 Register I2CCCR

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

Bit number	Bit Symbol	Description
7~5	SMPDIV	I2C sample clock setting, I2C sample clock is the (2^smpdiv) division of the I2C clock, that is: 000：Fsample=Fi2cclk 001：Fsample=Fi2cclk/2 010：Fsample=Fi2cclk/4 ... 111：Fsample=Fi2cclk/128
4~0	I2CCKD	I2C SCL output clock frequency setting, SCL output clock frequency is the sampling frequency of (I2CCKD + 1) division, that is : Fscl= Fsample /(I2CCKD +1)) ***Note:*** *1. When SMPDIV= 0, if I2CCKD <9, it will be calculated by 9 automatically.* *2. When SMPDIV> 0, if I2CCKD < 7, it will be calculated by 7 automatically.*

Table 19-5-5 Register I2CDAT

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

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

Table 19-5-6 Register I2CSTA

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

Bit number	Bit Symbol	Description
7~0	I2CSTA	I^{2}C status register 00H: (master/slave) bus error 08H: (master/slave)START signal detected (valid only when STFE=1)

		18H: (master)address and write bit sent, ACK signal received 20H: (master)address and write bit sent, no ACK signal received 28H: (master)one byte data received/transmitted, ACK signal detected 30H: (master)one byte data received/transmitted, no ACK signal detected 38H: (master)arbitration lost (master will change to slave after arbitration lost) 40H: (master)address and read bit transmitted, ACK signal received 48H: (master)address and read bit transmitted, no ACK signal received 60H: (slave)address and write bit received, with ACK signal is sent 70H: (master/slave) broadcast address received with ACK signal is sent (master/slave will become slave) 80H: (slave)one byte data received/transmitted, ACK signal detected 88H: (slave)one byte data received/transmitted, no ACK signal detected A0H: (master/slave) STOP signal detected (valid only when SPFE=1) A8H: (slave)address and read bit received, with ACK signal is sent F8H: (master/slave) bus is idle

Table 19-5-7 Register I2CFLG

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

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

Table 19-5-8 Register I2CIOS

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

Bit number	Bit Symbol	Description
7~2	-	-
1~0	I2CS	I^2C pin selection

		00：pin P37 for SCL, pin P36 for SDA 01：pin P31 for SCL, pin P30 for SDA 10：pin P67 for SCL, pin P66 for SDA 11：pin P60 for SCL, pin P61 for SDA

19.6 I^{2}C Control Example

◆ **I^{2}C as master**

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

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

#define I2C_ADDR          0xCA                  // I2C slave address definition
unsigned char xdata WriteBuffer[20]={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19};
void main(void)
{
     unsigned char i;
   EA = 1;                                      //global interrupt enabled
/********** Select I2C port ***********************************************************/
   I2CIOS = 0;                    // Select P36, P37 as I2C communication pins
     P36F = 3| (1<<7);            // set P36 as I2C SDA, and enable pull-up function
   P37F = 3| (1<<7);      // set P37 as I2C SCL, and enable pull-up function

//   I2CIOS = 1;                  // Select P30, P31 as I2C communication pins
//   P30F =4 | (1<<7);            // set P30 as I2C SDA, and enable pull-up function
//   P31F =4 | (1<<7);            // set P31 as I2C SCL, and enable pull-up function

//   I2CIOS = 2;                  // Select P66, P67 as I2C communication pins
//   P66F = 5 | (1<<7);           // set P66 as I2C SDA, and enable pull-up function
//   P67F = 5 | (1<<7);           // set P67 as I2C SCL, and enable pull-up function

//   I2CIOS = 3;                  // Select P61, P60 as I2C communication pins
//   P61F = 5 | (1<<7);           // set P61 as I2C SDA, and enable pull-up function
//   P60F = 5 | (1<<7);           // set P60 as I2C SCL, and enable pull-up function
/**************************************************************************************/

   I2CCON = I2CE(1) | I2CIE(0) | STA(0) | STP(0)| CKHD(1) | AAK(1)| CBSE(0) | STFE(1);
   I2CADR = GCE(0);
   I2CCCR = 0x29;                               // set the I2C clock frequency division
   while(1)
   {
        I2CCON |= STA(1);                       //I2C master sends START signal
        while(!(I2CFLG & I2CF));                // waiting for the interrupt
```

```
                if(I2CSTA != 0x08)
                {
                   I2CFLG  |= I2CF;
                   goto SEND_STOP;
                }
                I2CDAT = I2C_ADDR;                  // send I²C slave address+write bit
                I2CFLG  |= I2CF;                    // clear the interrupt flag
                while(!(I2CFLG & I2CF));            // waiting for the interrupt flag
                if(I2CSTA != 0x18)
                {
                   I2CFLG  |= I2CF;
                   goto SEND_STOP;
                }

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

For instance, the master reads 20-byte data from the slave cyclically, the program is like:

```
      #define I2C_ADDR              0xCA            / I2C slave address definition
      unsigned char xdata ReadBuffer[20];
      void main(void)
      {
          unsigned char i;
          EA = 1;                                                          //          global
interrupt enabled
      /********** Select I2C port ****************************************************************/
          I2CIOS = 0;                       // set P36, P37 as I²C communication pin
            P36F = 3| (1<<7);               // set P36 as I²C SDA, and enable pull-up function
          P37F = 3| (1<<7);                 // set P37 as I²C SCL, and enable pull-up function

      //  I2CIOS = 1;                       // set P30, P31 as I²C communication pin
      //  P30F =4 | (1<<7);                 // set P30 as I²C SDA, and enable pull-up function
      //  P31F =4 | (1<<7);                 // set P31 as I²C SCL, and enable pull-up function

      //  I2CIOS = 2;                       // set P66, P67 as I²C communication pin
      //  P66F = 5 | (1<<7);                // set P66 as I²C SDA, and enable pull-up function
//   P67F = 5 | (1<<7);              // set P67 as I2C SCL, and enable pull-up function

//   I2CIOS = 3;                     // set P61, P60 as I2C communication pin
//   P61F = 5 | (1<<7);              // set P61 as I2C SDA, and enable pull-up function
//   P60F = 5 | (1<<7);              // set P60 as I2C SCL, and enable pull-up function
/*********************************************************************************/

     I2CCON = I2CE(1) | I2CIE(0) | STA(0) | STP(0)| CKHD(1) | AAK(1)| CBSE(0) | STFE(1);
     I2CADR = GCE(0);
     I2CCCR = 0x29;                                    // set the I2C clock frequency division
     while(1)
     {
             I2CCON |= STA(1);                         // I2C master sends START signal
             while(!(I2CFLG & I2CF));                  // waiting for the interrupt
             if(I2CSTA != 0x08)
             {
                     I2CFLG  |= I2CF;
                     goto SEND_STOP;
             }
             I2CDAT = I2C_ADDR;                        // send I2C slave address+write bit
             I2CFLG  |= I2CF;                          // clear the interrupt flag

             while(!(I2CFLG & I2CF));                  // waiting for the interrupt flag
             if(I2CSTA != 0x18)
             {
                     I2CFLG  |= I2CF;
                     goto SEND_STOP;
             }

             I2CDAT = 0;                               // send data register address
             I2CFLG  |= I2CF;                          // clear the interrupt flag
             while(!(I2CFLG & I2CF));                  // waiting for the interrupt flag
             if(I2CSTA != 0x28)
             {
                     I2CFLG  |= I2CF;
                     goto SEND_STOP;
             }

             I2CCON |= STA(1);                         // I2C master sends START signal
             I2CFLG  |= I2CF;                          // clear the interrupt flag
             while(!(I2CFLG & I2CF));                  // waiting for the interrupt flag
             if(I2CSTA != 0x08)
             {
                     I2CFLG  |= I2CF;
                     goto SEND_STOP;
             }

             I2CDAT = I2C_ADDR+1;                      // send I2C slave address+write bit
             I2CFLG  |= I2CF;                          // clear the interrupt flag
             while(!(I2CFLG & I2CF));                  // waiting for the interrupt flag
             if(I2CSTA != 0x40)
             {
                     I2CFLG  |= I2CF;
                     goto SEND_STOP;
             }
             I2CCON |= AAK(1);                         //set ACK

             for(i = 0; i < 20; i++)
             {
                     I2CFLG  |= I2CF;                  // clear the interrupt flag
```

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

◆ **I2C as slave**

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

```
------------------------------------------------------------------------------------------------------------------------------------------
#define I2C_ADDR            0xCA            // I2C slave address definition
unsigned char I2CDataIndex;
unsigned char regAddr;
bit iicReadMode;
unsigned char xdata Buffer[20]={0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19};//设置数据寄存器初值为 0~19
void INT6_ISR(void) interrupt 11
{
    unsigned char Sta_Temp;

    if(I2CFLG & I2CF)                                           //IIC  interrupt
    {
            Sta_Temp = I2CSTA;
            if(Sta_Temp == 0x60)                    //receive I²C slave address+write bit
            {
                    I2CDataIndex = 0xFF;            // Set 0xFF means that the first byte received later is the address
                    iicReadMode = 0;                        // Set slave receive status
                    I2CCON |= AAK(1);
            }
            else if(Sta_Temp == 0x80)               // Send or receive one byte of data, ACK signal is detected
            {
                    if(iicReadMode)                 // Send one byte of data
                    {
                            I2CDataIndex++;
                            I2CDAT = Buffer[I2CDataIndex + regAddr];   // Load the data into the transmit register and wait
                                                                          for the master to read it
                    }
                    else                                            // One byte of data received
                    {
                            if(I2CDataIndex == 0xFF)        /地址
                            {
                                    regAddr = I2CDAT;       // The first byte received is considered to be the address
                                    I2CDataIndex = 0;                       // Set the index to 0
```

```
                                        I2CCON |= AAK(1);
                                }
                                else                                                              //data
                                {
                                  Buffer[I2CDataIndex + regAddr] = I2CDAT;  // Received data is loaded into the data register
                                        I2CDataIndex++;                                           // Index accumulation
                                        I2CCON |= AAK(1);
                                }
                        }
                }
                else if(Sta_Temp==0xA8)                        // receive I²C slave address+write bit, send ACK signal
                {
                        I2CDAT = Buffer[I2CDataIndex + regAddr];          // Load the data into the transmit register and
                                                                            wait for the host to read it
                        iicReadMode = 1;                                  // Set to slave transmit state
                }
                else if(Sta_Temp == 0x88)              // Send or receive one byte of data, an ACK signal is detected
                {
                }
                I2CFLG  |= I2CF;                                          // clear the interrupt flag
        }
}

void main(void)
{
    EA = 1;                                                               // global interrupt enabled
/********** Select I2C port ***************************************************************/
    I2CIOS = 0;                     // Select P36, P37 as I2C communication pins
    P36F = 3| (1<<7);               // set P36 as I2C SDA, and enable pull-up function
    P37F = 3| (1<<7);               // set P37 as I2C SCL, and enable pull-up function

//  I2CIOS = 1;                     // Select P30, P31 as I2C communication pins
//  P30F =4 | (1<<7);               // set P30 as I2C SDA, and enable pull-up function
//  P31F =4 | (1<<7);               // set P31 as I2C SCL, and enable pull-up function

//  I2CIOS = 2;                     // Select P66, P67 as I2C communication pins
//  P66F = 5 | (1<<7);              // set P66 as I2C SDA, and enable pull-up function
//  P67F = 5 | (1<<7);              // set P67 as I2C SCL, and enable pull-up function

//  I2CIOS = 3;                     // Select P61, P60 as I2C communication pins
//  P61F = 5 | (1<<7);              // set P61 as I2C SDA, and enable pull-up function
//  P60F = 5 | (1<<7);              // set P60 as I2C SCL, and enable pull-up function
/***************************************************************************************/
    I2CCON = I2CE(1) | I2CIE(1) | STA(0) | STP(0)| CKHD(1) | AAK(1)| CBSE(0) | STFE(0);
    I2CADR = GCE(0)|(I2C_ADDR>>1);                        // Set I2C slave address
    I2CCCR = 0x20;                                        // set the I²C clock frequency division
      INT6EN = 1;                                         // I2C interrupt enable
    while(1)
    {
    }
}
---------------------------------------------------------------------------------------------------------------------------------------------
```

20 LCD/LED Driver

20.1 LCD Driver

20.1.1 Function Introduction

The internal LCD driver supports at most 8com x 32seg、7com x 33seg、6com x 34seg、5com x 35seg、4com x 36seg.(40 output pins in all). The programmable duty cycle can be：1/2、1/3、1/4、1/5、1/6、1/7、1/8. Programmable bias voltage could be：1/2、1/3、1/4. There are 8 levels for the driving ability which is also programmable. Figure 20-1-1-1 shows the principle of LCD.

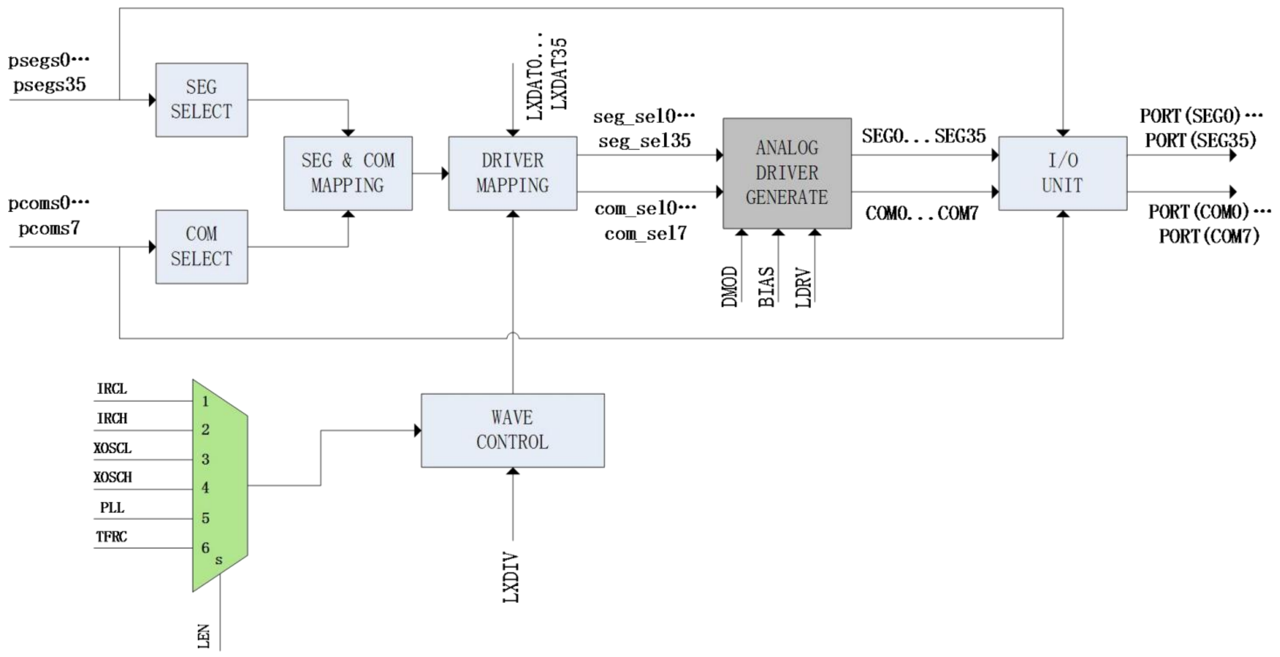


Figure 20-1-1-1 LCD Schematic

20.1.2 LCD Bias Voltage

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

- **LCD bias voltage 1/2**

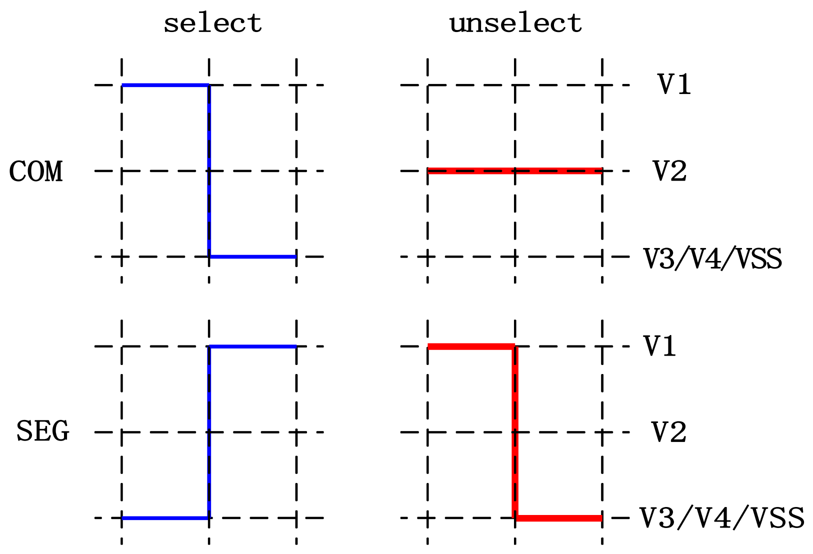


Figure 20-1-2-1 LCD bias voltage 1/2

- **LCD bias 1/3**

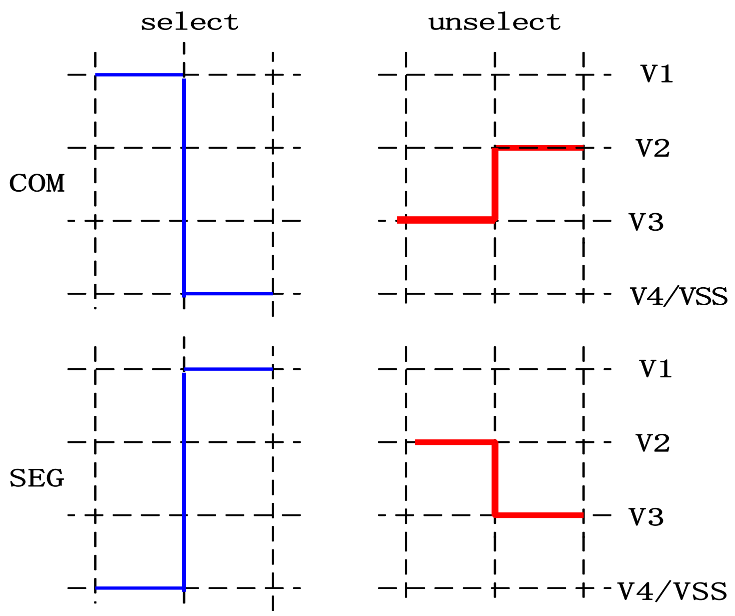


Figure 20-1-2-2 LCD bias voltage 1/3

- **LCD bias voltage 1/4**

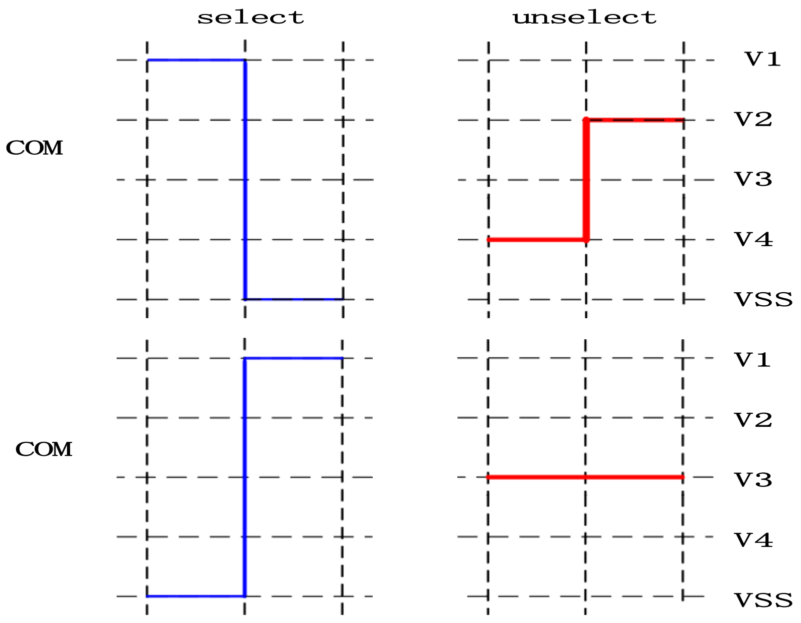


Figure 20-1-2-3 LCD bias voltage 1/4

20.1.3 LCD Function Description

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

The programmable duty cycle for LCD is：1/2、1/3、1/4、1/5、1/6、1/7、1/8, which is decided by the number of COM enabled. For instance, if 3 COM are enabled, then the duty cycle will be 1/3; if 8 COM are enabled, the duty cycle will be 1/8. Any of the COM can be enabled to and forms different combinations. However, the enabled COM pin number will correspond to real COM0, COM1, COM2... in order. For instance, pin COM3,COM5 and COM7 are the COM ports, then COM3 corresponds to real COM0, COM5 corresponds to real COM1 and COM7 corresponds to real COM2. Additionally, the duty cycle is 1/3. It is the same for SEG pins. For instance, pin SEG3, SEG5 and SEG7 are enabled, then SEG3 corresponds to real SEG0, SEG5 corresponds to real SEG1 and SEG7 corresponds to real SEG2. Those COM and SEG pins that are not enabled can still be used for other functions. It will not conflict with the LCD driver.

There is a 36 byte display buffer for LCD driver. LCD display buffer corresponds to the real COM and SEG. The 36 bytes corresponds to SEG0~SEG35 in order, with COM0~COM7 corresponding to 0~7 bit in each byte. The display can be visited by index register INDEX and data register LXDAT. INDEX=0~35 correspond SEG0~SEG35 display buffer respectively.

20.2 LED Driver

20.2.1 Function Introduction

Internal LED driver can support at most 8com x 32seg and shares the common display buffer and driving pins. There are 8 brightness levels for LED driver with global blink function within it. The blink frequency is 1Hz when the scanning frequency is 256Hz. Figure 20-2-1-1 shows the LED principle.

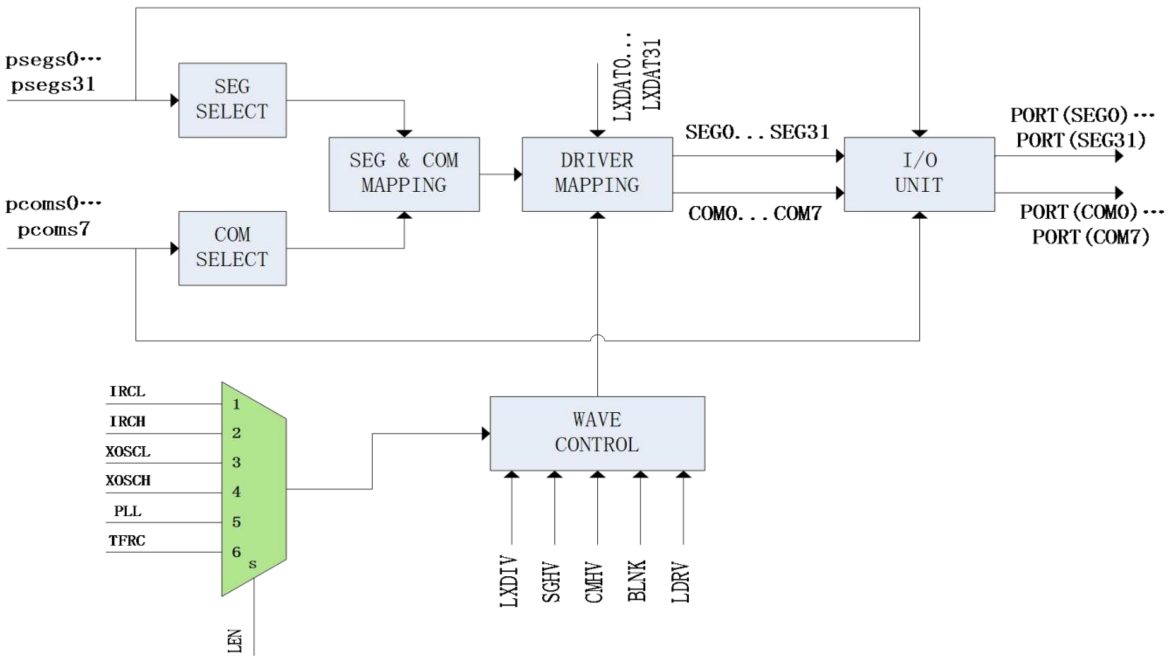


Figure 20-2-1-1 LED Schematic

Figure 20-2-1-2 shows the driving waveform for LED.

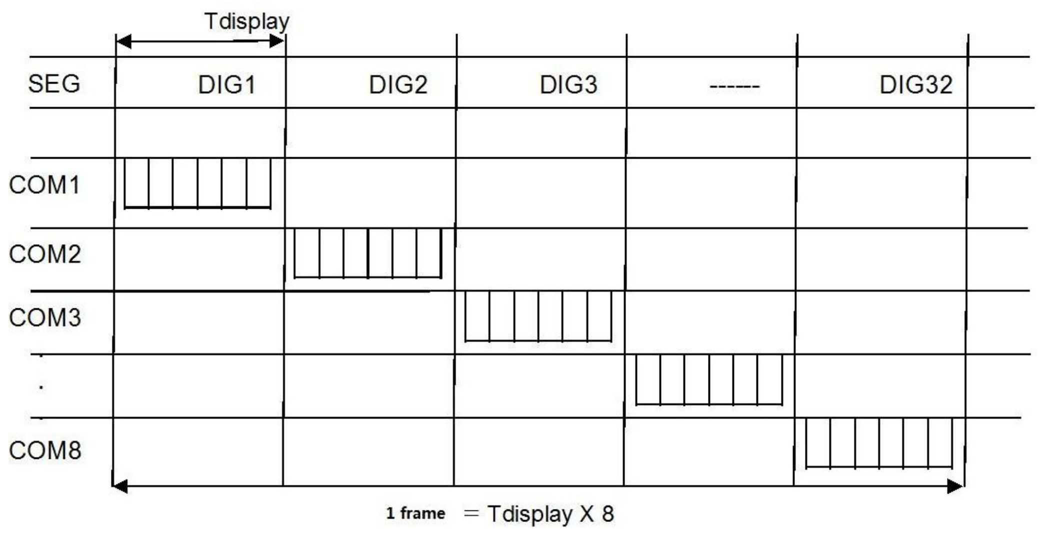


Figure 20-2-1-2 LED Driving Waveform

20.2.2 LED Function Description

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

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

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

20.3 LCD/LED Register Description

Table 20-3-1 Register LXCON

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

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

Table 20-3-2 Register LXCFG

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

Bit number	Bit Symbol	Description
LCD mode		
7~6	DMOD	LCD driver current selection 00：5uA 01：40uA 10：80uA 11：130uA
5~4	BIAS	LCD bias voltage selection 01：1/2 Bias 10：1/3 Bias Others：1/4 Bias
3	-	-
2~0	LDRV	LCD driver magnitude selection 000：Level 1(minimum) 001：Level 2 ... 111：Level 8(maximum)
LED mode		
7~6	-	-
5	COMHV	When it is 0/1, indicates COM valid output is 0/1
4	SEGHV	When it is 0/1, indicates SEG valid output is 0/1
3	BLNK	LED global blink control with 1 is valid Note： The blink time is the 128 frame LED’s output. For instance, if the clock for LED operation is 32.768KHz with LXDIV=0, the LED output frequency is 256Hz, then the blink frequency is1Hz.

2~0	LDRV	LED brightness selection 000：Level 1(darkest) 001：Level 2 010：Level 3 011：Level 4 100：Level 5 101：Level 6 110：Level 7 111：Level 8(brightest)

Table 20-3-3 Register LXDIV

E4H	7	6	5	4	3	2	1	0
LXDIVL	LXDIV[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
E5H	7	6	5	4	3	2	1	0
LXDIVH	-	-	-	-	LXDIV[11:8]			
R/W	-	-	-	-	R/W			
Initial Value	-	-	-	-	0	0	0	0

Bit number	Bit Symbol	Description
15~12	-	-
11~0	LXDIV	LXD clock frequency divider LCD frame scanning frequency=LXD clock frequency ÷((LXDIV+1) x 512) LED frame scanning frequency when high speed clock is selected=LXD clock frequency ÷((LXDIV+1) x 1024) LED frame scanning frequency when low speed clock is selected=LXD clock frequency ÷((LXDIV+1) x 128) ***Note：*** *1. The high-speed clocks for LED are IRCH/XOSCH/PLL/TFRC；* *The low-speed clock for LED is IRCL/XOSCL* *2. When IRCL is selected for LCD/LED, the frequency will be 1/4 of IRCL.* 3.*The typical frame scanning frequency for LCD is 64Hz, it is 256Hz for LED.*

Table 18-3-5 Register LXCAD

8117H	7	6	5	4	3	2	1	0
LXCAD	-	-	-	-	CAD_MOD[1:0]		CAD_LTH[1:0]	
R/W	-	-	-	-	R/W		R/W	
Initial Value	-	-	-	-	0	1	0	1

Bit number	Bit Symbol	Description
7~4	-	-

3~2	CAD_MOD	LCD_CAD Mode Selection 00： LCD_CAD off 01： LCD_CAD length controlled by digital signal Others： LCD_CAD length controlled by analog signal ***Note:** LCD_CAD can be optionally turned off in low-power mode, and the current can be reduced by about 5~10uA.*
1~0	CAD_LTH	Analog signal to control LCD_CAD length selection, valid only when CAD_MOD=2/3 00：4us 01：8us 10：12us 11：12us

Table 20-3-4 Register LXDAT

E3H	7	6	5	4	3	2	1	0
LXDAT	LXDAT[7:0]							
R/W	R/W							

Initial Value	0	0	0	0	0	0	0	0
Note：LXDAT is register with index, when INDEX=0~35, it indicates LXDAT0~LXDAT35 respectively								

Bit number	Bit Symbol	Description
7~0	LXDAT	Display buffer R/W register *Note: LXDAT32~LXDAT35 invalid for LED driver.*

Table 20-3-5 LCD/LED Display Buffer

INDEX	SEG	COM0	COM1	COM2	COM3	COM4	COM5	COM6	COM7
0	0	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
1	1	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
2	2	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
3	3	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
4	4	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
5	5	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
6	6	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
7	7	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
8	8	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
9	9	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
10	10	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
11	11	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
12	12	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
13	13	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
14	14	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
15	15	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
16	16	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
17	17	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
18	18	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
19	19	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
20	20	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
21	21	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
22	22	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
23	23	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
24	24	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
25	25	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
26	26	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
27	27	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
28	28	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
29	29	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
30	30	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
31	31	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	BIT7
32	32	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	BIT6	-
33	33	BIT0	BIT1	BIT2	BIT3	BIT4	BIT5	-	-

34	34	BIT0	BIT1	BIT2	BIT3	BIT4	-	-	-
35	35	BIT0	BIT1	BIT2	BIT3	-	-	-	-

20.4 LCD Driver Control Example

For instance, to drive LCD with 8com×20seg, 1/4bias, the program is like：

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

#define LEN_XOSCL           (3<<5)
#define LMOD_lcd        (0<<4)
#define DMOD_5ua        (0<<6)
#define BIAS_1_4        (0<<4)
#define LCDRV_LEV(N)        (N) //N=0~7
void LCD_init(void)
{
        unsigned char i;

        P00F = 3; //set P00 as COM0
        P01F = 3; //set P01 as COM1
        P02F = 3; //set P02 as COM2
        P03F = 3; //set P03 as COM3
        P04F = 3; //set P04 as COM4
        P05F = 3; //set P05 as COM5
        P06F = 3; //set P06 as COM6
        P07F = 3; //set P07 as COM7

        P57F = 3; //set P57 as SEG0
        P34F = 3; //set P34 as SEG1
        P35F = 3; //set P35 as SEG2
        P56F = 3; //set P56 as SEG3
        P50F = 3; //set P50 as SEG4
        P51F = 3; //set P51 as SEG5
        P52F = 3; //set P52 as SEG6
        P53F = 3; //set P53 as SEG7
        P54F = 3; //set P54 as SEG8
        P55F = 3; //set P55 as SEG9
        P60F = 3; //set P60 as SEG10
        P61F = 3; //set P61 as SEG11
        P62F = 3; //set P62 as SEG12
        P63F = 3; //set P63 as SEG13
        P64F = 3; //set P64 as SEG14
        P65F = 3; //set P65 as SEG15
        P10F = 3; //set P10 as SEG16
        P11F = 3; //set P11 as SEG17
        P12F = 3; //set P12 as SEG18
        P13F = 3; //set P13 as SEG19

        CKCON |= XLCKE;//enable XOSCL clock
        while(!(CKCON & XLSTA)); //wait until XOSCL clock becomes stable

        LXDIVH = 0;          //set the clock frequency division, the frame frequency of the LCD after division is 64Hz
        LXDIVL = 0;
        LXCFG = DMOD_5ua | BIAS_1_4 | LCDRV_LEV(7); //set the LCD driving current and magnitude, bias
        LXCON =   LEN_XOSCL | LMOD_lcd;      //set the clock source for LCD as XOSCL、set is as LCD mode

        for(i=0;i<20;i++) //write LCD display buffer
        {
                INDEX = i;          //set the index for display buffer
                LXDAT = 0;          //write buffer, clear the screen when 0 is written
        }
}
---------------------------------------------------------------------------------------------
```

20.5 LED Driver Control Example

For instance, to drive LED with 8com×20seg, with common cathode, the program is like：

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

#define LEN_XOSCL               (3<<5)
#define LMOD_led        (1<<4)
#define CMHV(N)                 (N<<5)    //N=0~1
#define SGHV(N)                 (N<<4)    //N=0~1
#define BLNK(N)                 (N<<3)    //N=0~1
void LED_init(void)
{
        unsigned char i;

        P00F = 3; //set P00 as COM0
        P01F = 3; //set P01 as COM1
        P02F = 3; //set P02 as COM2
        P03F = 3; //set P03 as COM3
        P04F = 3; //set P04 as COM4
        P05F = 3; //set P05 as COM5
        P06F = 3; //set P06 as COM6
        P07F = 3; //set P07 as COM7
        P57F = 3; //set P57 as SEG0
        P34F = 3; //set P34 as SEG1
        P35F = 3; //set P35 as SEG2
        P56F = 3; //set P56 as SEG3
        P50F = 3; //set P50 as SEG4
        P51F = 3; //set P51 as SEG5
        P52F = 3; //set P52 as SEG6
        P53F = 3; //set P53 as SEG7
        P54F = 3; //set P54 as SEG8
        P55F = 3; //set P55 as SEG9
        P60F = 3; //set P60 as SEG10
        P61F = 3; //set P61 as SEG11
        P62F = 3; //set P62 as SEG12
        P63F = 3; //set P63 as SEG13
        P64F = 3; //set P64 as SEG14
        P65F = 3; //set P65 as SEG15
        P10F = 3; //set P10 as SEG16
        P11F = 3; //set P11 as SEG17
        P12F = 3; //set P12 as SEG18
        P13F = 3; //set P13 as SEG19

        CKCON |= XLCKE;//enable XOSCL clock
        while(!(CKCON & XLSTA)); //wait until XOSCL clock is stable

        LXDIVH = 0;            //set clock frequency division, LED frame frequency is 256Hz after the division
        LXDIVL = 0;
        LXCFG = CMHV(0) | SGHV(1) | BLNK(0) | LEDRV_LEV(7);     //set low level valid for COM, high level valid
for SEG, disable the global blink, set the brightness to maximum
        LXCON =   LEN_XOSCL | LMOD_led;     //set the clock source for LED as XOSCL、set as LED mode

        for(i=0;i<20;i++)    //write the LED display buffer
        {
            INDEX = i;           //set the index for display buffer
            LXDAT = 0;           //write buffer, clear the screen when 0 is written
        }
}
--------------------------------------------------------------------------------------------
```

21 PWM

21.1 PWM Function Introduction

CA51F2 series chip can include at most 8 channels PWM outputs. PWM period and duty cycle can be configured with 16-bit range. There is center fixed and edge fixed mode for each PWM. In addition, PWM also supports the deadtime control and complementary output. There are 4 pairs of complementary channels consisting of 8 PWM in complementary mode. This is designed for brushless DC motor driver.

21.2 PWM Function Description

There is a 16-bit counter for each PWM channel and the cycle is set by register PWMDIV. Register PWMDUT sets the corresponding PWM's duty cycle. PWM is enabled by register PWMEN with each bit of it corresponds to one channel in PWM. There is a data refresh register PWMUPD for PWM module. When register PWMDIV, PWMDUT and PWMCKD is overwritten, corresponding bit of the register PWMUPD must be set to 1 and then the data can be refreshed. The corresponding bit of PWMUPD will be cleared after the data refresh. PWM pin can also output reversed phase by setting PWMTOG. There are multiple clock sources for PWM which is set for PWM pairs(PWM0 and PWM1, PWM2 and PWM3, PWM4 and PWM5, PWM6 and PWM7). In other words, the clock source is the same when the PWMs are in the same pair. The clock source can be selected by corresponding PWMCKS of register PWMCON, with the frequency division set by PWMCKD independently.

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

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

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

The single PWM output waveforms in edge/center fixed mode are shown as Figure 21-2-1 and Figure 21-2-2(note：for all the waveform blow,PWMDIV>PWMDUT>0). As the figure shows, with the same PWMDIV and PWMDUT, the PWM cycle for center fixed mode is 2 times longer than it for edge fixed mode.

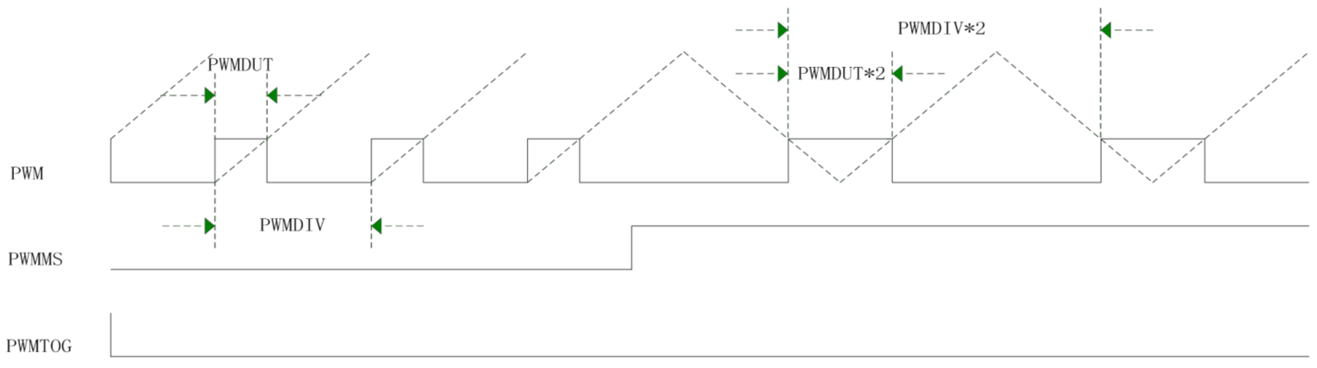


Figure 21-2-1 PWM Output Waveform when PWMTOG＝0

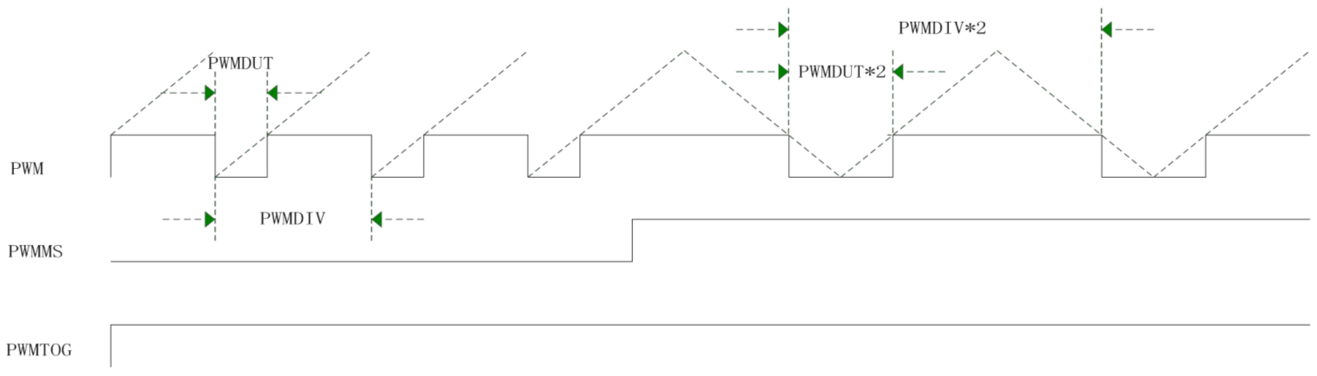


Figure 21-2-2 PWM Output Waveform when PWMTOG＝1

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

- **Complementary mode**

8 PWM can forms 4 pair complementary channels in this mode：PWM0 and PWM1, PWM2 and PWM3, PWM4 and PWM5, PWM6 and PWM7. The complementary mode for PWM is set by PWMMOD of control register PWMCON. In this complementary mode, PWM fixed mode,its cycle and duty cycle, the clock frequency division for it, are all set by the corresponding register for PWM0, PWM2, PWM4, PWM6. Only PWMTOG is still controlled by the channel's corresponding register independently. The Figure 21-1-3 shows the principle of this mode.

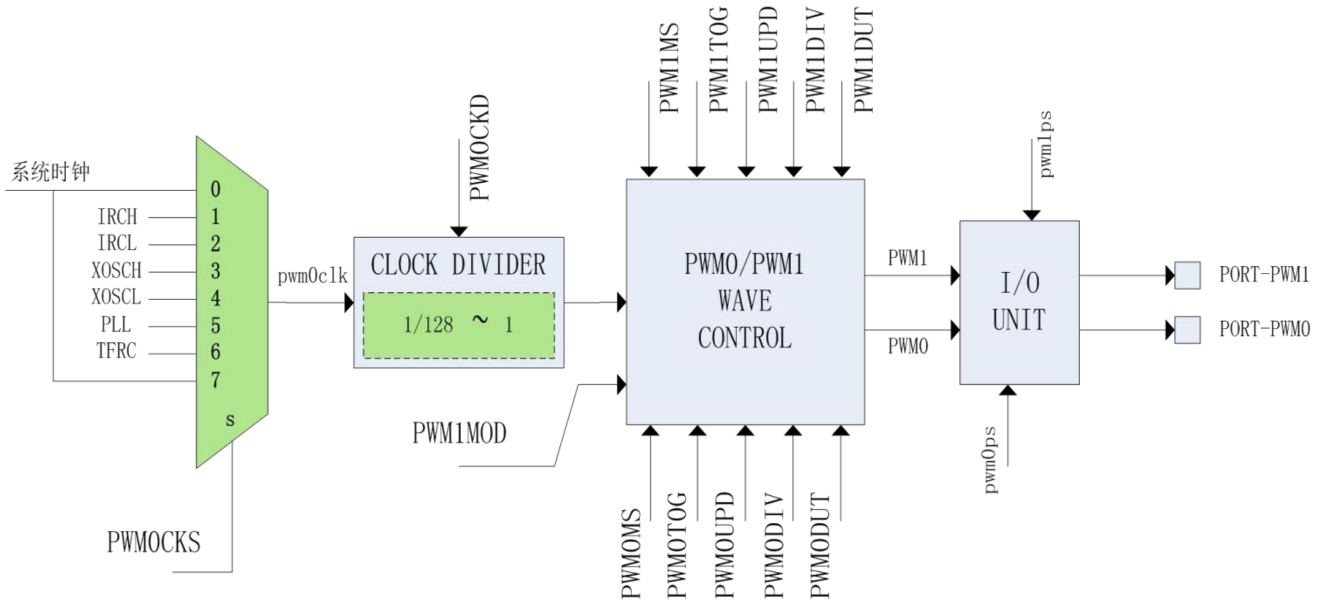


Figure 21-1-3 Schematic for PWM0 and PWM1

The phases are complementary for each pair as Figure 21-1-4 and Figure 21-1-5 shows(PWM0 and PWM1 for example)。

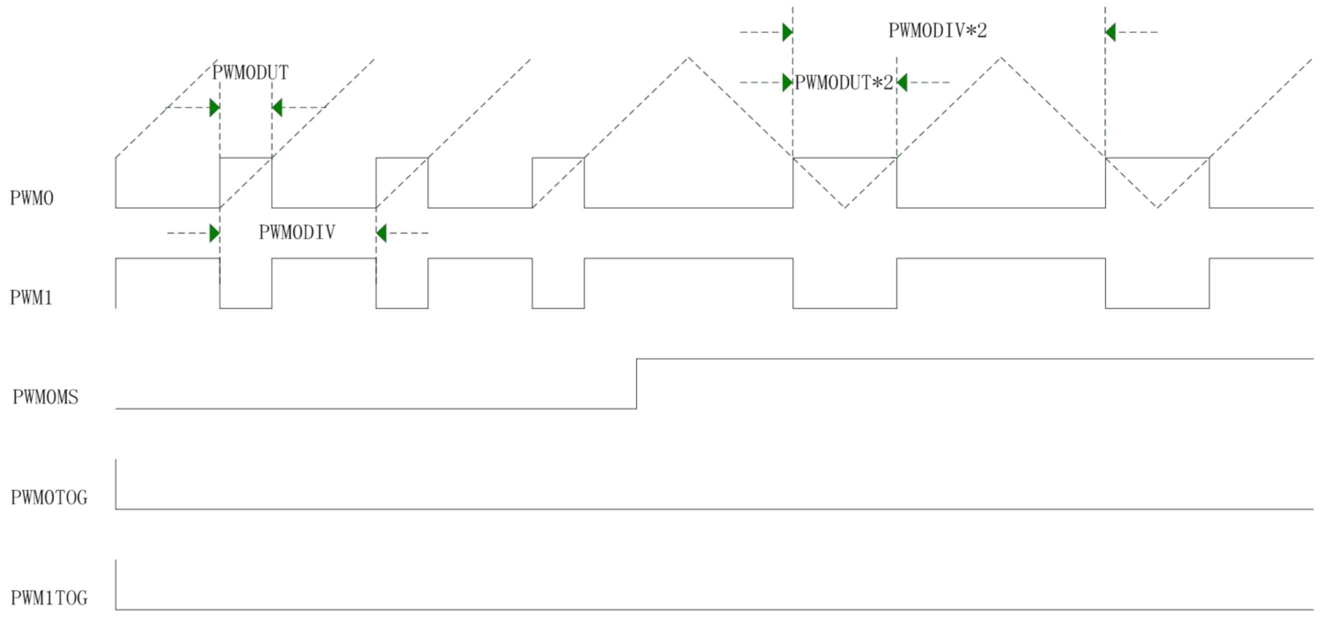


Figure 21-1-4 PWM0 and PWM1 Complementary Output Waveform when PWMTOG＝0

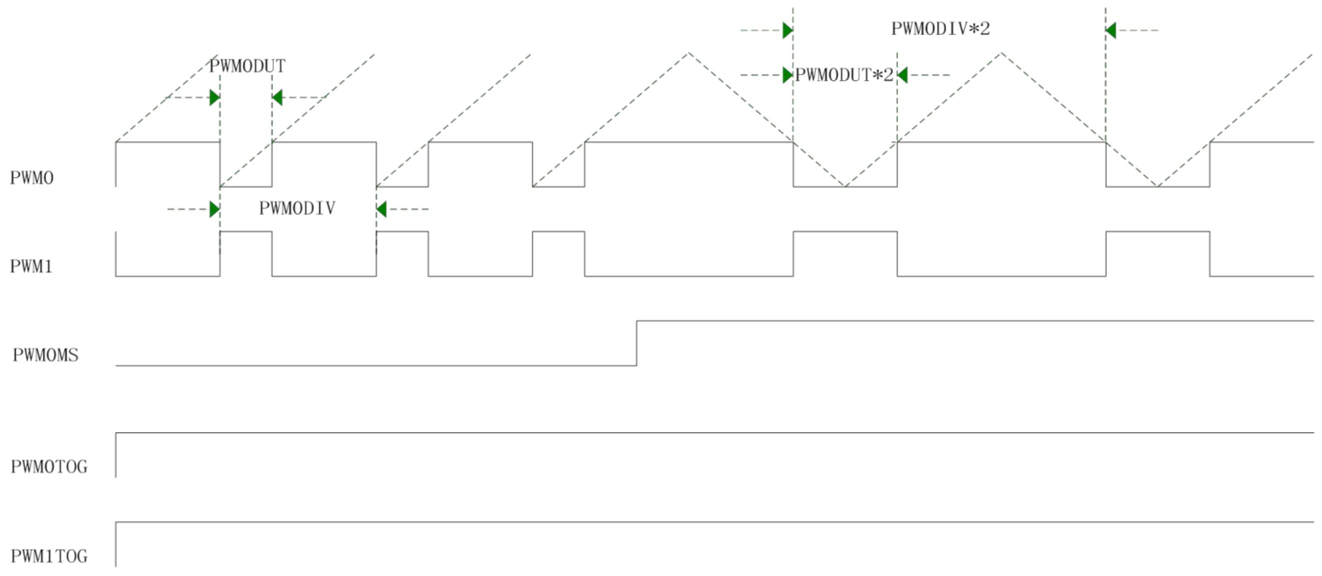


Figure 21-1-5 PWM0 and PWM1 Complementary Output Waveform when PWMTOG＝1

- **Deadtime control**

In bridge driver circuit, deadtime control is a must to avoid the two half bridges are conducted at the same time. The deadtime can be controlled by PWM1, PWM3,PWM5 and PWM7's corresponding register PWMDIV and PWMDUT. PWMDIV sets the deadtime on the left and PWMDUT sets the deadtime on the right. The deadtime must meet the requirements below (take PWM0 and PWM1, for example)：

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

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

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

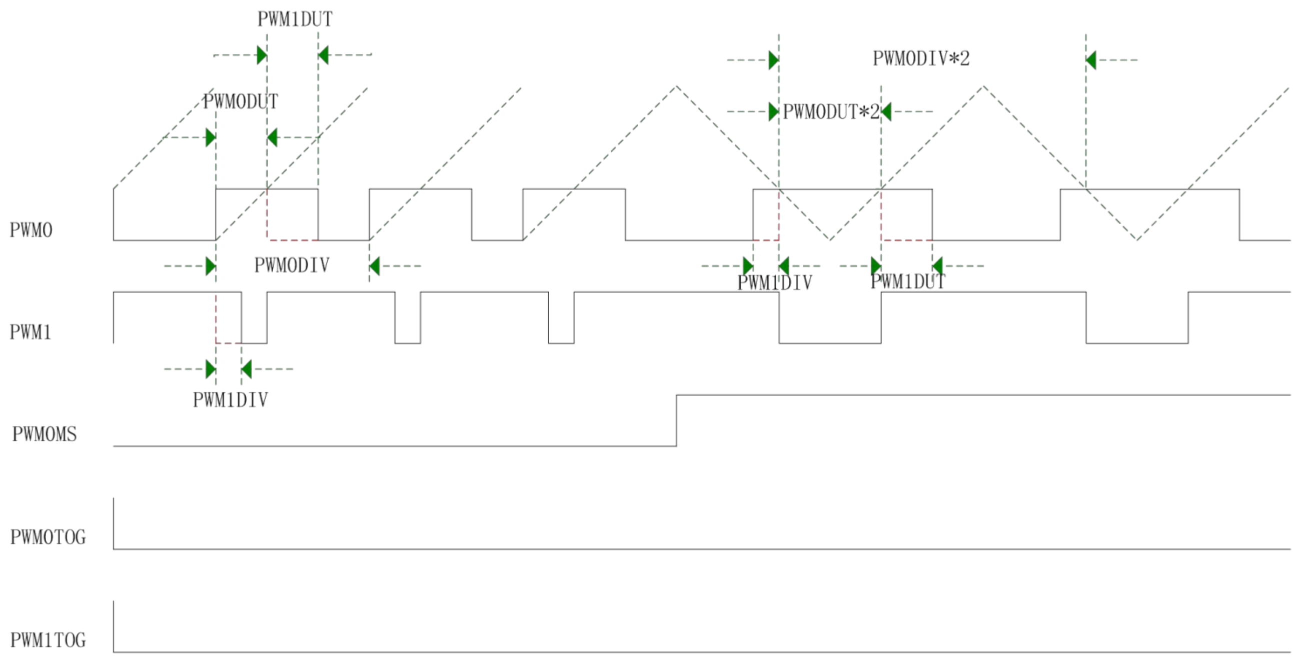


Figure 21-1-6 PWM0 and PWM1 Deadtime Control Waveform when PWMTOG＝0

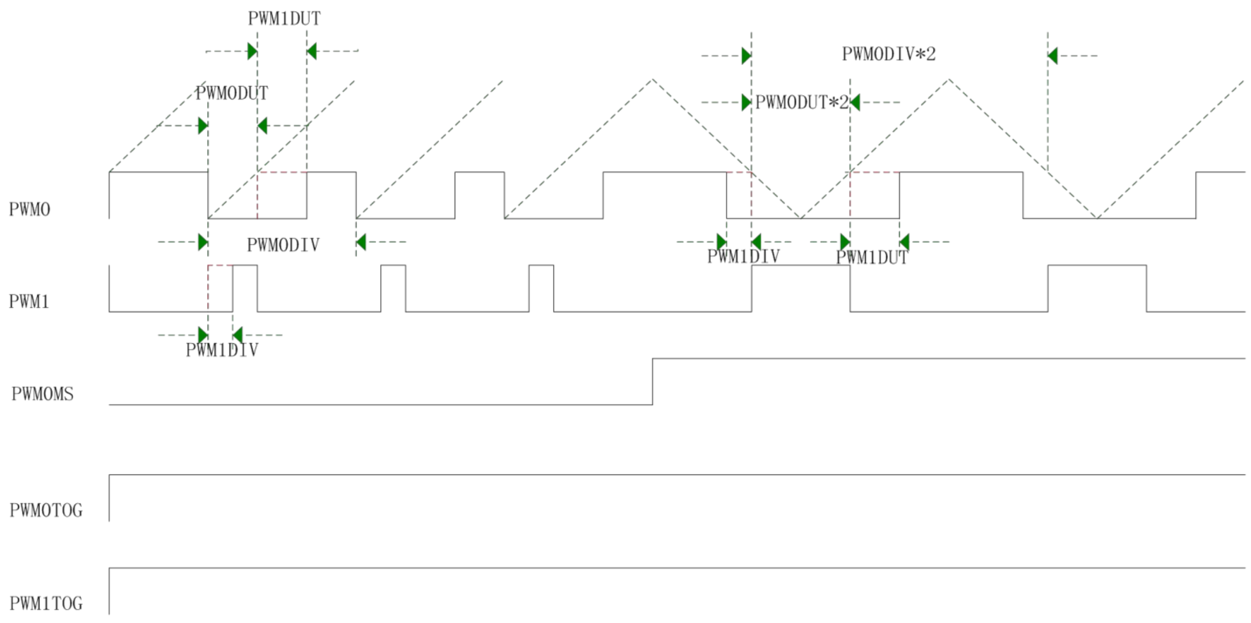


Figure 21-1-7 PWM0 and PWM1 Deadtime Control Waveform when PWMTOG＝1

- **PWM Interrupt**

PWM interrupt in enabled by PWMTIE, PWMZIE, PWMPIE and PWMNIE in register PWMCON. PWMTIE enables the interrupt when PWM counter's count reaches the peak (which equals PWMDIV). PWMZIE enables the interrupt when PWM counter's count reaches the bottom(which equals 0). PWMNIE enables the interrupt when the output pin falling edge comes. PWMPIE enables the interrupt when the output pin rising edge comes. PWMTIE and PWMZIE's corresponding interrupts do not exist in edge fixed mode. Register PWMAIF, PWMBIF, PWMCIF and PWMDIF are the interrupt status registers for the 8 channel's interrupts. PWMxTIF, PWMxZIF, PWMxNIF, PWMxPIF correspond to PWMTIE, PWMZIE, PWMNIE, PWMPIE respectively.

In addition, an interrupt may be triggered only when the interrupt event has occurred for several times. The number of times can be set by register PWMCMX. For instance, if PWMCMX=3 and PWMPIE=1, then there will be a rising edge interrupt when PWM pin rising edge has come 4 times.

21.3 PWM Register Description

Table 21-3-1 Register PWMEN

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

Bit number	Bit Symbol	Description
7~0	PWMEN	7~0 bit correspond to PWM channel 7~0's enable control, 1 enables it

Table 21-3-2 Register PWMUPD

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

Bit number	Bit Symbol	Description
7~0	PWMUPD	7~0 bit correspond to PWM channel 7~0's data refresh enable control, 1 enables it ***Note:*** *To refresh the channel's data(PWMDIV/PWMDUT/PWMCKD), set the PWMUPD corresponding position to 1 and the data will be refreshed when the PWM counter overflows. The corresponding bit will be cleared after the data refresh.*

Table 21-3-3 Register PWMCMX

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

Note：PWMCMX is register with index, INDEX=0~7 corresponds to PWMCMX0~PWMCMX7

Bit number	Bit Symbol	Description
7~0	PWMCMX	PWM interrupt events number for each channel Interrupt event number=PWMCMX+1, For instance, when INDEX=0 and PWMCMX=7, then only when the interrupt trigger events happens 8 times the interrupt flag will be set to 1.

Table 21-3-4 Register PWMCON

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

Bit number	Bit Symbol	Description
Note: *PWMCON①is the channel control for PWM0/PWM2/PWM4/PWM6* *PWMCON②is the channel control for PWM1/PWM3/PWM5/PWM7* *PWMCON is also register with index, INDEX=0~7 correspond to PWMCON0~PWMCON7 respectively*		
7	PWMTIE	PWM counter peak interrupt enable control, 1 enables it
6	PWMZIE	PWM counter bottom interrupt enable control, 1 enables it
5	PWMPIE	PWM rising edge interrupt enable control, 1 enables it
4	PWMNIE	PWM falling edge interrupt enable control, 1 enables it
3	PWMMS	PWM mode selection 0：edge fixed 1：center fixed
2~0	PWMCKS	PWM working clock selection 001：IRCH 010：IRCL 011：XOSCH 100：XOSCL 101：PLL 110：TFRC Others：system clock ***Note:*** *PWM0/PWM1 is set by PWMCKS0;* *PWM2/PWM3 is set by PWMCKS2;* *PWM4/PWM5 is set by PWMCKS4;* *PWM6/PWM7 is set by PWMCKS6.*
0	PWMMOD	Complementary mode enable control, 1 enables it ***Note:*** *When PWMMOD1=1, PWM0 and PWM1 enter the* complementary mode *When PWMMOD3=1, PWM2 and PWM3 enter the* complementary mode *When PWMMOD5=1, PWM4 and PWM5 enter the* complementary mode *When PWMMOD7=1, PWM6 and PWM7 enter the* complementary mode

Table 21-3-5 Register PWMCFG

DEH	7	6	5	4	3	2	1	0
PWMCFG	PWMTOG	PWMCKD[6:0]						
R/W	R/W	R/W						
Initial Value	0	0	0	0	0	0	0	0
Note：PWMCFG is register with index, INDEX=0~7 corresponds to PWMCFG0~PWMCFG7								
Bit number	Bit Symbol	Description						
7	PWMTOG	PWM phase reversion enable, 1 inverts the phase						
6~0	PWMCKD	PWM working clock frequency division setting						

		0000000：no division 0000001：frequency divided by 2 0000010：frequency divided by 3 ...... 1111110：frequency divided by 127 1111111：frequency divided by 128

Table 21-3-6 Register PWMDIVL、PWMDIVH

DFH	7	6	5	4	3	2	1	0
PWMDIVL	PWMDIV[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
D1H	7	6	5	4	3	2	1	0
PWMDIVH	PWMDIV[15:8]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Note：PWMDIV is register with index, INDEX=0~7 corresponds to PWMDIV0~PWMDIV7								

Bit number	Bit Symbol	Description
15~0	PWMDIV	PWM cycle configuration PWMDIV1/PWMDIV3/PWMDIV5/PWMDIV7 are for different use in complementary mode, please refer to register PWMDUT description

Table 21-3-7 Register PWMDUTL, PWMDUTH

D2H	7	6	5	4	3	2	1	0
PWMDUTL	PWMDUT[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
D3H	7	6	5	4	3	2	1	0
PWMDUTH	PWMDUT[15:8]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Note：PWMDUT is register with index, INDEX=0~7 corresponds to PWMDUT0~PWMDUT7								

Bit number	Bit Symbol	Description

15~0	PWMDUT	PWM duty cycle setting In complementary mode, PWMDUT1/PWMDUT3/PWMDUT5/PWMDUT7 are for different use as the table below： PWMDIV1: Controls width of the dead time on the left for PWM0/PWM1 PWMDUT1: controls width of the dead time on the right for PWM0/PWM1 PWMDIV3: Controls width of the dead time on the left for PWM2/PWM3 PWMDUT3: Controls width of the dead time on the right for PWM2/PWM3 PWMDIV5: controls width of the dead time on the left forPWM4/PWM5 PWMDUT5: Controls width of the dead time on the right for PWM4/PWM5 PWMDIV7: Controls width of the dead time on the left for PWM6/PWM7 PWMDUT7: Controls width of the dead time on the right for PWM6/PWM7

Table 21-3-8 Register PWMAIF

D4H	7	6	5	4	3	2	1	0
PWMAIF	PWM1TIF	PWM1ZIF	PWM1PIF	PWM1NIF	PWM0TIF	PWM0ZIF	PWM0PIF	PWM0NIF
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	PWM1TIF	PWM1 counter peak interrupt flag, cleared when 1 is written to it
6	PWM1ZIF	PWM1 counter bottom interrupt flag, cleared when 1 is written to it
5	PWM1PIF	PWM1 rising edge interrupt flag, cleared when 1 is written to it
4	PWM1NIF	PWM1 falling edge interrupt flag, cleared when 1 is written to it
3	PWM0TIF	PWM0 counter peak interrupt flag, cleared when 1 is written to it
2	PWM0ZIF	PWM0 counter bottom interrupt flag, cleared when 1 is written to it
1	PWM0PIF	PWM0 rising edge interrupt flag, cleared when 1 is written to it
0	PWM0NIF	PWM0 falling edge interrupt flag, cleared when 1 is written to it

Table 21-3-9 Register PWMBIF

D5H	7	6	5	4	3	2	1	0
PWMBIF	PWM3TIF	PWM3ZIF	PWM3PIF	PWM3NIF	PWM2TIF	PWM2ZIF	PWM2PIF	PWM2NIF
R/W	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7	PWM3TIF	PWM3 counter peak interrupt flag, cleared when 1 is written to it
6	PWM3ZIF	PWM3 counter bottom interrupt flag, cleared when 1 is written to it
5	PWM3PIF	PWM3 rising edge interrupt flag, cleared when 1 is written to it

4	PWM3NIF	PWM3 falling edge interrupt flag, cleared when 1 is written to it
3	PWM2TIF	PWM2 counter peak interrupt flag, cleared when 1 is written to it
2	PWM2ZIF	PWM2 counter bottom interrupt flag, cleared when 1 is written to it
1	PWM2PIF	PWM2 rising edge interrupt flag, cleared when 1 is written to it
0	PWM2NIF	PWM2 falling edge interrupt flag, cleared when 1 is written to it

Table 21-3-10 Register PWMCIF

D6H	7	6	5	4	3	2	1	0
PWMCIF	PWM5TIF	PWM5ZIF	PWM5PIF	PWM5NIF	PWM4TIF	PWM4ZIF	PWM4PIF	PWM4NIF
R/W	R	R	R	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7	PWM5TIF	PWM5 counter peak interrupt flag, cleared when 1 is written to it
6	PWM5ZIF	PWM5 counter bottom interrupt flag, cleared when 1 is written to it
5	PWM5PIF	PWM5 rising edge interrupt flag, cleared when 1 is written to it
4	PWM5NIF	PWM5 falling edge interrupt flag, cleared when 1 is written to it
3	PWM4TIF	PWM4 counter peak interrupt flag, cleared when 1 is written to it
2	PWM4ZIF	PWM4 counter bottom interrupt flag, cleared when 1 is written to it
1	PWM4PIF	PWM4 rising edge interrupt flag, cleared when 1 is written to it
0	PWM4NIF	PWM4 rising edge interrupt flag, cleared when 1 is written to it

Table 21-3-11 Register PWMDIF

D7H	7	6	5	4	3	2	1	0
PWMDIF	PWM7TIF	PWM7ZIF	PWM7PIF	PWM7NIF	PWM6TIF	PWM6ZIF	PWM6PIF	PWM6NIF
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	PWM7TIF	PWM7 counter peak interrupt flag, cleared when 1 is written to it
6	PWM7ZIF	PWM7 counter bottom interrupt flag, cleared when 1 is written to it
5	PWM7PIF	PWM7 rising edge interrupt flag, cleared when 1 is written to it
4	PWM7NIF	PWM7 falling edge interrupt flag, cleared when 1 is written to it
3	PWM6TIF	PWM6 counter peak interrupt flag, cleared when 1 is written to it
2	PWM6ZIF	PWM6 counter bottom interrupt flag, cleared when 1 is written to it
1	PWM6PIF	PWM6 rising edge interrupt flag, cleared when 1 is written to it
0	PWM6NIF	PWM6 falling edge interrupt flag, cleared when 1 is written to it

21.4 PWM Control Example

◆ **Single channel PWM output**

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

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

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

#define CKS_IH                      (1<<0)

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

void PWM_init(void)
{
        P50F = 5;                 //set P50 as PWM pin
        INDEX = PWM_CH0;     //set INDEX to PWM0
        PWMCON = TIE(0) | ZIE(0) | PIE(0) | NIE(0) | MS(0) | CKS_IH ;//disable PWM0 interrupt, set PWM0  as edge
fixed mode, set PWM0 clock source as IRCH
        PWMCFG = TOG(0) | 0; //disable the phase reversion, no frequency division for the clock

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

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

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

```
------------------------------------------------------------------------------------------
void PWM_init(void)
{
        P50F = 5;                 //set P50 as PWM pin
        INDEX = PWM_CH0;     //set INDEX to PWM0
        PWMCON = TIE(0) | ZIE(0) | PIE(0) | NIE(0) | MS(0) | CKS_IH ;//disable PWM0 interrupt, set PWM0  as edge
fixed mode, set PWM0 clock source as IRCH
```

```
        PWMCFG = TOG(0) | 0; //disable the phase reversion, no frequency division for the clock

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

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

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

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

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

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

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

#define CKS_IH              (1<<0)

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

void PWM_init(void)
{
        P50F = 5;           //set P50 as PWM0 pin
        P51F = 5;           //set P51as PWM1 pin

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

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

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

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

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

- **PWM interrupt example**

For instance, PWM0 is set to center fixed mode and turn on peak, bottom, rising edge, falling edge interrupts. The program is like：

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

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

#define CKS_IH                      (1<<0)

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

void PWM_init(void)
{
        P50F = 5;                 //set P50 as PWM pin
        INDEX = PWM_CH0;      //set INDEX to PWM0
        PWMCON = TIE(1) | ZIE(1) | PIE(1) | NIE(1) | MS(1) | CKS_IH ;//enables PWM0 interrupt, set PWM0 as center
fixed mode, set PWM0 clock source as IRCH
        PWMCFG = TOG(0) | 0; //disable the phase reversion, no frequency division for the clock

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

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

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

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

22 Analog/Digital Converter (ADC)

22.1 Function Introduction

Analog/digital converter is a 12-bit successive approximation(SAR) ADC, with at most 8 input channels. The clock source for ADC is the system clock with frequency division configurable. There are ADC multiple reference voltages for ADC. When internal voltage is selected as the reference voltage, it can be used to test the power supply voltage for the chip and there will be correction to ensure the chip’s consistency as well. There is also a compare mode for it with threshold configurable. Once it goes beyond the threshold, a corresponding interrupt occurs. The signal can be amplified/narrowed before the conversion when using ADC and OPAMP together.

22.2 Main Features

- 12 bit resolution
- 8 input channels at most
- Supports ADC interrupt
- ADC clock frequency division configurable
- Alternate reference voltage：internal reference voltage, VDD, external reference voltage
- Automatic data correction supported when internal reference voltage is selected
- Support configurable comparator mode
- Detected signal can converted after amplification/narrowing
- Input voltage range：VSS<=VIN<=VDD。

22.3 Block Diagram

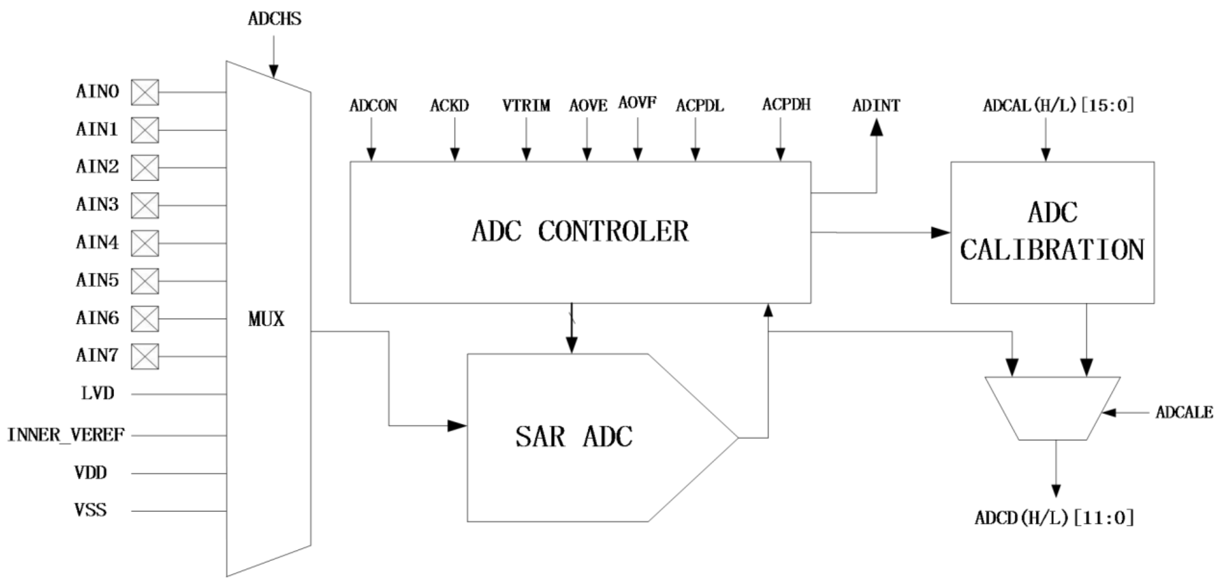


Figure 22-3-1 ADC Architecture

22.4 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. The shortest ADC conversion time is 0.5us. Figure 22-4-1 is the sequence diagram for ADC conversion.

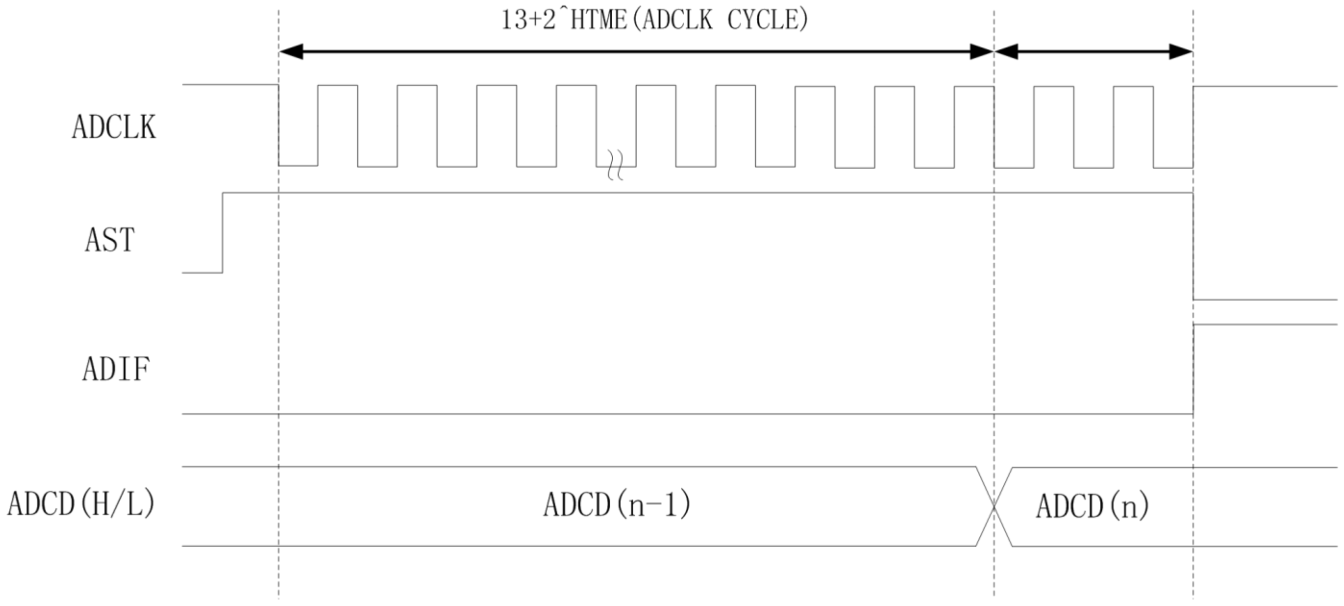


Figure 22-4-1 ADC Sequence Diagram

- **Compare Mode**

This mode is enabled by AOVE. When AOVE=1, ADC conversion result ADCD will be compared to threshold ADCPDL and ADCPDH after the conversion. When ADCD exceeds the threshold range, the compare interrupt flag will be set to 1. If the ADC interrupt is enabled then, the interrupt occurs.

- **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. The accurate AD value will be obtained by calculation according to the correction 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.

- **ADC and OPAMP Combination**

ADC detection signal can be amplified or attenuated by OPAMP A. For more information please refer to the OPAMP description.

22.5 Register Description

Table 22-5-1 Register ADCON

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

Bit number	Bit Symbol	Description
7	AST	ADC conversion 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 flag, cleared when 1 is written to it
4~2	HTME	The number of sampling periods is 2 power HTME
1~0	VSEL	ADC reference voltage selection 00: internal 1.5V(INNER_VREF)as reference voltage 01: external VDD 10: external VREF 11: internal 1.5V(INNER_VREF) as reference voltage

Table 22-5-2 Register ADCFGL

BAH	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](P40) 0010： enable channel AD_CH[1](P41) 0011： enable channel AD_CH[2](P42) 0100： enable channel AD_CH[3](P43) 0101： enable channel AD_CH[4](P44) 0110： enable channel AD_CH[5](P45) 0111： enable channel AD_CH[6](P46) 1000： enable channel AD_CH[7](P47) 1001： enable 1/4 VDD detection 1011： enable INNER_VREF detection 1100： enable LDO voltage detection 1101： enable VSS detection Others： disable the channels

Table 22-5-3 Register ADCFGH

BBH	7	6	5	4	3	2	1	0
ADCFGH	AOVF	AOVE	VTRIM					
R/W	R/W	R/W	R/W					
Initial Value	0	0	1	0	0	0	1	1

Bit number	Bit Symbol	Description
7	AOVF	overflow flag in compare mode
6	AOVE	Compare mode enable control, 1 enables it
5~0	VTRIM	Internal 1.5V reference voltage correction register, with accuracy ±1mV

Table 22-5-4 Register ADCAL

8088H	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
8089H	7	6	5	4	3	2	1	0
ADCALH	ADCAL[15:8]							
R/W	R/W							
Initial Value	0	0	0	0	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 22-5-5 Register ADCPDL

808AH	7	6	5	4	3	2	1	0
ADCPDLL	ADCPDL[3:0]				-	-	-	-
R/W	R/W				-	-	-	-
Initial Value	0	0	0	0	0	0	0	0
808BH	7	6	5	4	3	2	1	0
ADCPDLH	ADCPDL[11:4]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
15~0	ADCPDL	Threshold lower limit setting register in compare mode

Table 22-5-6 Register ADCPDH

808CH	7	6	5	4	3	2	1	0
ADCPDHL	ADCPDH[3:0]				-	-	-	-
R/W	R/W				-	-	-	-
Initial Value	0	0	0	0	0	0	0	0
808DH	7	6	5	4	3	2	1	0
ADCPDHH	ADCPDH[11:4]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
15~0	ADCPDH	Threshold upper limit setting register in compare mode

Table 22-5-7 Register ADCD

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

Initial Value	0	0	0	0	0	0	0	0

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

22.6 ADC Control Example

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

```
----------------------------------------------------------------------------------------------------
#define AST(N)          (N<<7)    //N=0~1
#define ADIE(N)         (N<<6)    //N=0~1
#define ADIF        (1<<5)    //interrupt flag
#define HTME(N)   (N<<2)    //N=0~7              //set the sampling time, it is the 2 power HTME clock cycle periods
#define VSEL(N)   (N)       //N=0~3              //reference voltage selection: 0-internal 1-VDD 2-external

#define ACKD(N)              (N<<5) //N=0~7
#define ADCALE(N)            (N<<4)    //N=0~1
#define ADCHS(N)        (N)  //N=0~15          //ADC channel selection, 1~13 corresponds to 0~12
void ADC_init(void)
{
        P40F = 3; //set P40 as ADC pin
        ADCON = AST(0) | ADIE(1) | HTME(7) | VSEL(1);//enable ADC interrupt, set the sampling cycle time, VDD
                                                                      selected as the reference voltage
        ADCFGL = ACKD(1) | ADCALE(0) | ADCHS(1);//set the ADC clock frequency division, set the ADC channel to
                                                                   ADC0
        ADCON |= AST(1); //enable AD conversion
        INT2EN = 1;          //enable INT2 interrupt
}
void ADC_ISR (void) interrupt 7
{
        unsigned int AD_Value;
        if(ADCON & ADIF)
        {
                ADCON |= ADIF; //clear ADC interrupt
                AD_Value = ADCDH*256 + ADCDL;     //read ADC value
                AD_Value >>= 4;
                ADCON |= AST(1); //enable next AD conversion
        }
        ......
}
----------------------------------------------------------------------------------------------------
```

23 Analog Comparator and Operational Amplifier(OPCMP)

23.1 Function Introduction

OPCMP module includes 4 comparators, 2 operational amplifiers and 1 capture counter. Either external input voltage or internal DA can be selected as the reference voltage for the comparator.

The comparator’s input could be set to IO input or analog input (for comparator 3, only analog input). When it is IO input, the logical level of the IO port is detected. There is a digital filter for each comparator with 15-bit filtering time configurable. The filtered signal reversion will the generate an interrupt signal.

The module includes two OPAMPs：OPAMP A and OPAMP B. OPAMP A and B can amplify/narrow the input signal of pin OPIN and output the processed signal with pin OPOUT. OPAMP A can amplify/narrow the signal detected by ADC and be the input for ADC. It expands the detectable voltage range for ADC. On the other hand, OPAMP B can be set to the input for comparator to detect small signal.

The capture counter can be used to capture the time interval between the comparator’s reversions and calculate the speed of the motor.

In the CA51F2 series of chips, the comparator is mainly designed for brushless DC motor drive. The above characteristics of the comparator can realize the Hall detection of the Hall motor and the detection of the zero-crossing point of the Hallless motor.

23.2 CMP Architecture

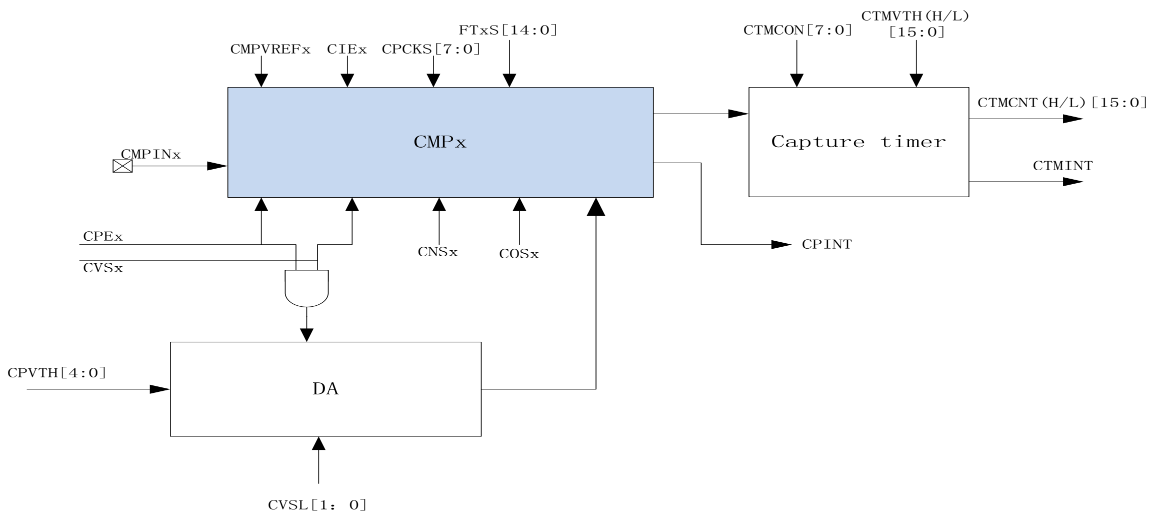


Figure 23-2-1 CMP Schematic

23.3 Function Description

23.3.1 OPAMP

OPAMP A is controlled by register OPACON. When OPAM=1, pin OPAIN(P35) is selected as the input and with pin OPAOUT(P34) selected as the output for the OPAMP. The amplification can be set by OPAS. When OPAM=2, the detection channel selected by ADC will be the input for OPAMP A. The output of OPAMP A will be the ADC's input. The signal can be amplified/narrowed according to the ratio set by OPAS which expands the detectable voltage range greatly.

OPAMP B is controlled by register OPBCON. When OPBE=1, OPAMP B is enabled, OPBPIN (P17) is the positive input of OPAMP B, OPBNIN (P13) is the negative input of OPAMP B, OPBOUT (P16) It is the output port of OPAMP B.

23.3.2 Comparator

The design for comparator 0,1 and 2 is identical. The design for comparator 3 is also the same as comparator 0,1 and 2 except the IO input can not be the trigger source for Comparator 3. There are 2 input pins for each comparator, CMPxN and CMPxP(x=0,1,2,3). CMPxN is the input reference voltage for comparator x and CMPxP is the positive input for it. The reference voltage can be either CMPxN input voltage or DA, which is selected by CVSx. When the CMPxP voltage is greater than the reference voltage, the comparator will output logical 1, and vice versa. When DA is selected as the reference voltage, the voltage source for DA can be BANDGAP,LDO output or VDD, which can be selected by VSEL. CPVTH sets the voltage division of the selected voltage source.

There is a 15-bit digital filter for each comparator, of which the threshold is set by the register FTxS. When the comparator’s output reverses, the digital filter starts counting. The clock for counting and its frequency division can be set by register CPCKS. When the count reaches the threshold of the digital filter, the comparator’s output logic will be refreshed at CPxD, which is in the comparator’s status register CPSTA. The interrupt flag CPxIF will be set to 1 as well. If the comparator’s output reverses before the count reaches threshold, then the counter starts counting again, which means only when the comparator’s output keeps for certain time set by the threshold, it is seen as valid output. If the filtered output reverses, the comparator interrupt occurs. There several selective trigger edges: rising edge, falling edge or both. The trigger edge can be selected by COSx and CP0IF is the interrupt flag for the comparator.

23.3.3 Capture Counter

The capture counter is a 16-bit counter with the same clock source as the comparator’s digital filter. The trigger event is the valid output of comparator. The trigger source is selected by CTMS and can be the rising edge of any of the 3 comparators or all together. The capture counter is enabled by CTME. When the count value reashes the threshold the corresponding interrupt occurs and the threshold interrupt flag CTMIVF will be set to 1. The counter continues counting. When the count reaches the maximum(FFFFH) the overflow interrupt occurs and the overflow interrupt flag CTMOVF is set to 1. In addition, the counter will be reset to 0. When the comparator’s trigger edge comes, the current count will be loaded to register CTMCNT and the counter will be reset to 0.

The capture counter is designed for brushless motor driver. The capture count corresponds to the interval time of the Hall status, and will be used to calculate the speed of the motor. The threshold can be used to detect the locked rotor with appropriate value. The threshold interrupt indicates the locked rotation.

23.4 Register Description

Table 23-4-1 Register OPACON

8040H	7	6	5	4	3	2	1	0

OPACON	OPAM		-			OPAS		
R/W	R/W		-			R/W		
Initial Value	0	0	-	-	-	0	0	0

Bit number	Bit Symbol	Description
7~6	OPAM	OPAMP A enable control 00： disable OPAMP A 01： OPAMP A enabled, pin OPAIN(P35) for OPAMP A's input, OPAOUT(P34) for OPAMP A's output 10： OPAMP A enabled, ADC detection channel for OPAMP A's input, OPAMP A's output for ADC's input 11： disable OPAMP A
5~3	-	-
2~0	OPAS	ADC amplification setting 000： 1/4 time 001： 1/3 time 010： 1/2 time 011： 5 times 100： 10 times 101： 15 times 110： 20 times 111： 30 times

Table 23-4-2 Register OPBCON

8041H	7	6	5	4	3	2	1	0
OPBCON	-	-	OPBE	-	-	-	-	-
R/W	-	-	R/W	-	-	-	-	-
Initial Value	-	-	0	-	-	-	-	-

Bit number	Bit Symbol	Description
7~6	-	-
5	OPBE	OPAMP B enable control , 1 enable it
4~0	-	-

Table 23-4-3 Register CP0CON

8048H	7	6	5	4	3	2	1	0
CP0CON	CPE0	CIE0	CVS0	CZS0	-	-	COS0[1:0]	
R/W	R/W	R/W	R/W	R/W	-	-	R/W	
Initial Value	0	0	0	0	-	-	0	0

Bit number	Bit Symbol	Description
7	CPE0	Comparator 0 enable control, 1 enables
6	CIE0	Comparator 0 interrupt enable control, 1 enables
5	CVS0	Reference voltage selection for Comparator 00：External 1：DA
4	CZS0	Comparator 0 delaying voltage selection 0: disable delaying 1: enable delaying
3~2	-	-
1~0	COS0	comparator interrupt trigger edge selection 00：rising edge 01：falling edge Others：both

Table 23-4-4 Register CP1CON

8049H	7	6	5	4	3	2	1	0
CP1CON	CPE1	CIE1	CVS1	CZS1	-	-	COS1[1:0]	
R/W	R/W	R/W	R/W	R/W	-	-	R/W	
Initial Value	0	0	0	0	-	-	0	0

Bit number	Bit Symbol	Description
7	CPE1	comparator1 enable control, 1 enables it
6	CIE1	comparator1 interrupt enable control, 1 enables it

5	CVS1	comparator1 reference voltage selection 0：external 1：DA
4	CZS1	comparator1delaying voltage selection 0: disable delaying 1: enable delaying
3~2	-	-
1~0	COS1	comparator1enable control, 1 enables it

Table 23-4-5 Register CP2CON

804AH	7	6	5	4	3	2	1	0
CP2CON	CPE2	CIE2	CVS2	CZS2	-	-	COS2[1:0]	
R/W	R/W	R/W	R/W	R/W	-	-	R/W	
Initial Value	0	0	0	0	-	-	0	0

Bit number	Bit Symbol	Description
7	CPE2	comparator2 enable control, 1 enables it
6	CIE2	comparator2 interrupt enable control, 1 enables it
5	CVS2	comparator2 reference voltage selection 0：external 1：DA
4	CZS2	comparator2delaying voltage selection 0: disable delaying 1: enable delaying
3~2	-	-
1~0	COS2	comparator2enable control, 1 enables it

Table 23-4-6 Register CP3CON

804BH	7	6	5	4	3	2	1	0
CP3CON	CPE3	CIE3	CVS3	CZS3	-	-	COS3[1:0]	
R/W	R/W	R/W	R/W	R/W	-	-	R/W	
Initial Value	0	0	0	0	-	-	0	0

Bit number	Bit Symbol	Description
7	CPE3	comparator3 enable control, 1 enables it
6	CIE3	comparator3 interrupt enable control, 1 enables it
5	CVS3	comparator3 reference voltage selection 0：external

		1：DA
4	CZS3	comparator3delaying voltage selection 0: disable delaying 1: enable delaying
3~2	-	-
1~0	COS3	comparator3enable control, 1 enables it

Table 23-4-7 Register CPCKS

804CH	7	6	5	4	3	2	1	0
CPCKS	CPDIV					CPCKSEL		
R/W	R/W					R/W		
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~3	CPDIV	Comparator clock frequency division selection 00000：no division 00001：frequency divided by 2 00010：frequency divided by 4 00011：frequency divided by 6 … 11111：frequency divided by 62
2~0	CPCKSEL	Comparator clock selection 000：system clock selected 001：IRCH 010：IRCL 011：XOSCH/ERC 100：XOSCL 101：TFRC 110：PLL 111：system clock selected

Table 23-4-8 Register CPSTA

804DH	7	6	5	4	3	2	1	0
CPSTA	CP3D	CP2D	CP1D	CP0D	CP3IF	CP2IF	CP1IF	CP0IF
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	CP3D	Analog comparator 3 output
6	CP2D	comparator 2 output or external pin P2.4 input

5	CP1D	comparator 1 output or external pin P2.2 input
4	CP0D	comparator 0 output or external pin P2.0 input
3	CP3IF	comparator 3 interrupt flag
2	CP2IF	comparator 2 interrupt flag
1	CP1IF	comparator 1 interrupt flag
0	CP0IF	comparator 0 interrupt flag

Table 23-4-9 Register CPVTC

804EH	7	6	5	4	3	2	1	0
CPVTC	VSEL[1:0]		-	CPVTH[4:0]				
R/W	R/W		-	R/W				
Initial Value	0	0	-	0	0	0	0	0

Bit number	Bit Symbol	Description
7~6	VSEL	DA module reference voltage selection 00: internal reference voltage 01：LDO output 10：VDD 11：reserved
5	-	-
4~0	CPVTH	DA module output voltage selection Output voltage = reference voltage÷(2^5)×(CPVTH+1)

Table 23-4-10 Register FT0S

8050H	7	6	5	4	3	2	1	0
FT0SL	FT0S[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
8051H	7	6	5	4	3	2	1	0
FT0SH	-	FT0S[14:8]						
R/W	-	R/W						
Initial Value	-	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
14~0	FT0S	The threshold for digital filter of comparator 0

Table 23-4-11 Register FT1S

8052H	7	6	5	4	3	2	1	0
FT1SL	FT1S[7:0]							

R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
8053H	7	6	5	4	3	2	1	0
FT1SH	-	FT1S[14:8]						
R/W	-	R/W						
Initial Value	-	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
14~0	FT1S	The threshold for digital filter of comparator 1

Table 23-4-12 Register FT2S

8054H	7	6	5	4	3	2	1	0
FT2SL	FT2S[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
8055H	7	6	5	4	3	2	1	0
FT2SH	-	FT2S[14:8]						
R/W	-	R/W						
Initial Value	-	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
14~0	FT2S	The threshold for digital filter of comparator 2

Table 23-4-13 Register FT3S

8056H	7	6	5	4	3	2	1	0
FT3SL	FT3S[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
8057H	7	6	5	4	3	2	1	0
FT3SH	-	FT3S[14:8]						
R/W	-	R/W						
Initial Value	-	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
14~0	FT3S	The threshold for digital filter of comparator 3

Table 23-4-14 Register CTMCON

8058H	7	6	5	4	3	2	1	0

CTMCON	CTME	CTMIE	CTMOE	CTMVE	CTMS		CTMIVF	CTMOVF
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	CTME	capture counter enable control，1 enables it
6	CTMIE	capture counter interrupt enable control，1 enables it
5	CTMOE	capture counter overflow enable control，1 enables it *Note: If CTMOE=1 and CTMIE=1，the counter's overflow will induce overflow interrupt. the corresponding interrupt flag is CTMOVF.*
4	CTMVE	capture counter threshold trigger enable control，1 enables it *Note: If CTMVE=1 and CTMIE=1，when the counter reaches the threshold there will be threshold trigger interrupt, the corresponding interrupt flag is CTMIVF.*
3~2	CTMS	capture counter trigger sourceselection 00：CP0D rising edge 01：CP1D rising edge 10：CP2D rising edge 11：rising edge of CP0D or CP1D or CP2D
1	CTMIVF	capture counter threshold trigger interrupt flag
0	CTMOVF	capture counter overflow interrupt flag

Table 23-4-15 Register CTMVTH

8059H	7	6	5	4	3	2	1	0
CTMVTHL	CTMVTH[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
805AH	7	6	5	4	3	2	1	0
CTMVTHH	CTMVTH[15:8]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
15~0	CTMVTH	The threshold for capture counter

Table 23-4-16 Register CTMCNT

805BH	7	6	5	4	3	2	1	0
CTMCNTL	CTMCNT[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

805CH	7	6	5	4	3	2	1	0
CTMCNTH	CTMCNT[15:8]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
15~0	CTMCNT	Capture counter’s count

24 Brushless DC Motor Driver(MOTOR)

24.1 Function Introduction

There is a brushless DC motor control module with Hall state decoding function embedded in CA51F2 Series chip. It supports 60° and 120° Hall. There is automatic, brake and manual control modes for it as well. The Hall state corresponding driving status can be set in automatic mode and implements 6-step phase changing for both positive and negative rotation. For the brake mode, there are also 2 modes: brake with driver and automatic stop. The brake can be done with brake instruction sent. Writing the register HDCT by software can set the motor's driving state in manual control mode. It reduces the coding time and improves the response speed for motor(without Hall device) driver and sinusoidal driver. There is multiple detection for the motor and the motor may pause or stop when the abnormality happens. The module's external PWM, analog comparator , ADC and etc can be used for the motor driver as well, which makes the driver usage more flexible.

24.2 Block Diagram

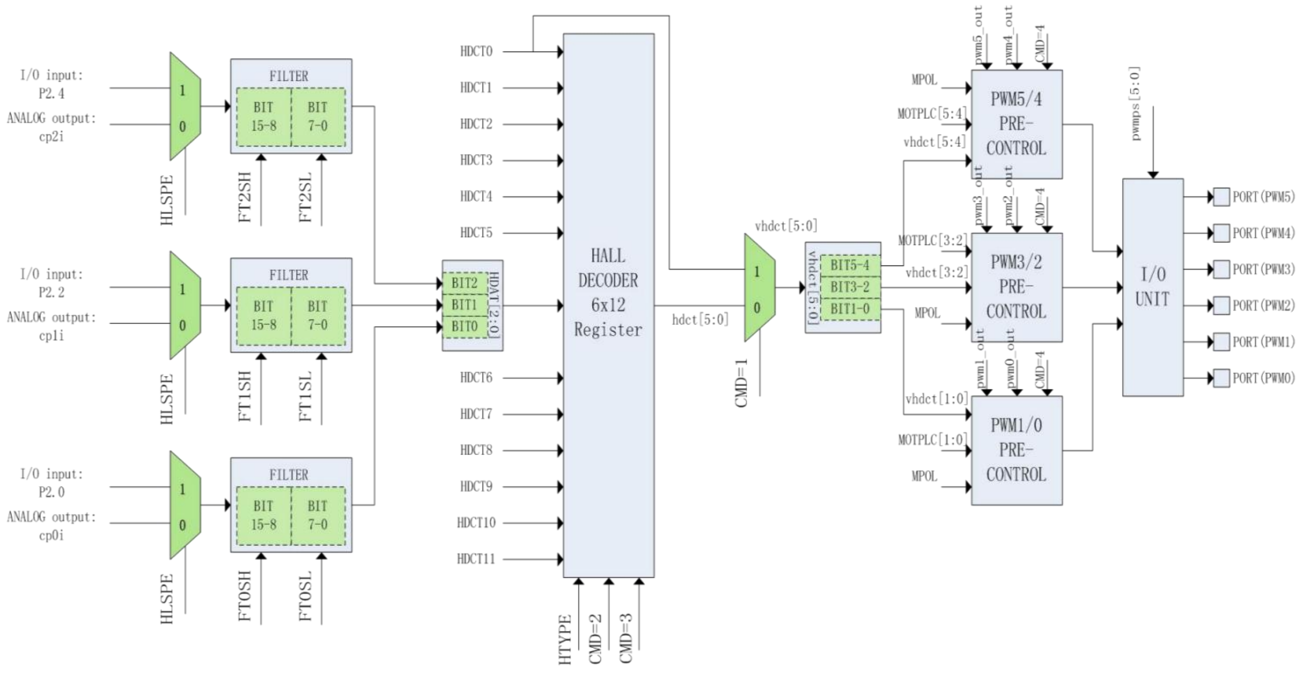


Figure 24-2-1 Block Diagram for DC Motor Driver

24.3 Function Description

24.3.1 Hall State Decoding Function

When the motor control module is enabled and set to automatic mode (MOTCMD=1 or MOTCMD=2), the decoding is enabled automatically. The Hall output of the motor connects to the input of analog comparator 0,1 and 2. The output of the 3 comparators will be the input HA, HB, HC for the decoder. The Hall state [HC:HB:HA] controls the motor driving by different registers according to different status. HDCT0~HDCT5 correspond to 6 phase control registers which makes the motor rotates forward, while HDCT6~HDCT11 correspond to 6 phase control registers which makes the motor rotates backward. Table 24-3-1-1 shows the relationship between the Hall state and HDCT.

Table 24-3-1-1 Hall state and the Corresponding HDCT

	120° Hall						60° Hall			
Rotates forward (MOTCMD=2)	HC	HB	HA	HDCT		Rotates forward (MOTCMD=2)	HC	HB	HA	HDCT
	0	0	1	HDCT0			0	0	1	HDCT0
	0	1	1	HDCT1			0	1	1	HDCT1
	0	1	0	HDCT2			1	1	1	HDCT2
	1	1	0	HDCT3			1	1	0	HDCT3
	1	0	0	HDCT4			1	0	0	HDCT4
	1	0	1	HDCT5			0	0	0	HDCT5
Rotates backward (MOTCMD=3)	0	0	1	HDCT6		Rotates backward (MOTCMD=3)	0	0	1	HDCT6
	0	1	1	HDCT7			0	1	1	HDCT7
	0	1	0	HDCT8			1	1	1	HDCT8
	1	1	0	HDCT9			1	1	0	HDCT9
	1	0	0	HDCT10			1	0	0	HDCT10
	1	0	1	HDCT11			0	0	0	HDCT11

24.3.2 Manual Control Mode

When MOTCMD=1, the motor works in manual control mode. The Hall decoding is invalid in this mode and the motor's state is controlled by HDCT0. This mode can be used when to drive the motor without Hall device or by sinusoidal waves. It will reduce the software codes and improve the response speed.

24.3.3 MASK Function

The block diagram for MASK function is shown as Figure 24-3-3-1.

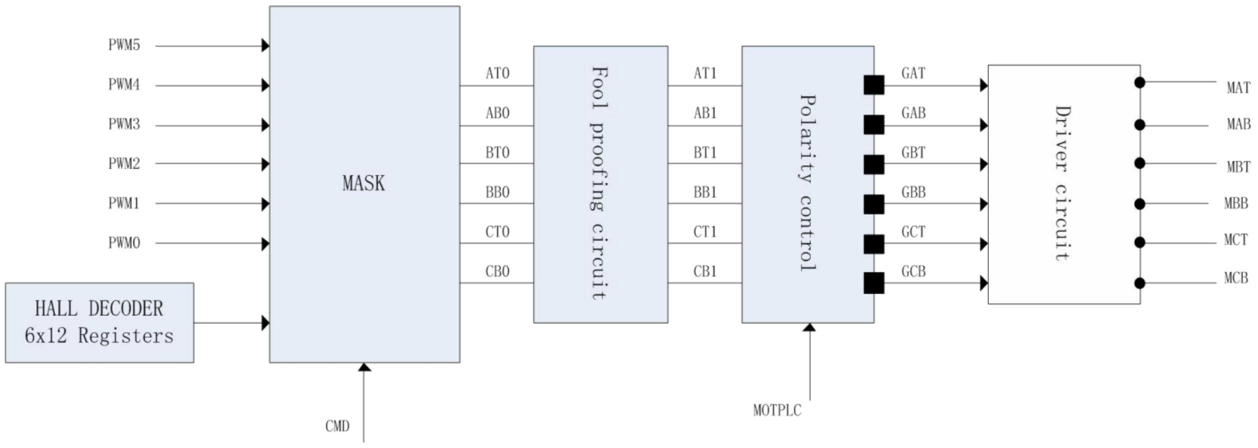


Figure 24-3-3-1 Block Diagram MASK Function

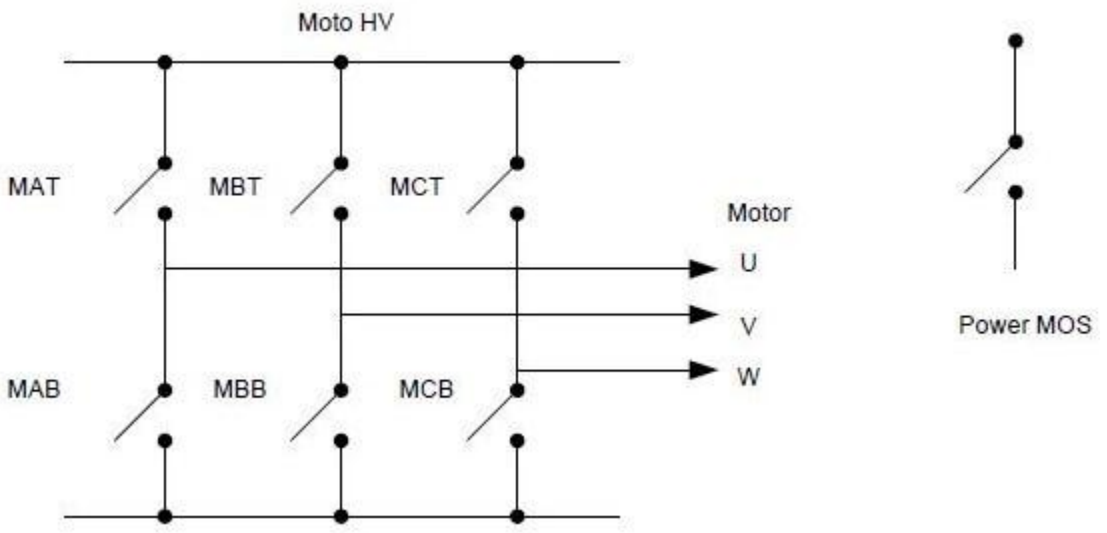


Figure 24-3-3-2 MASK Transformation

MASK function includes 3 modes: normal, brake and manual control.

- **Normal mode**

Either top driving or bottom driving can be selected by MPOL in normal mode. When MPOL=1, it is set to top driving and vice versa. (PWM output modulated in upper half bridge in top driving mode and in lower half bridge in bottom driving mode)

In MASK module, PWM0~PWM5 is used as the source driving signal and AT0(corresponds to PWM0)、AB0(corresponds to PWM1), BT0(corresponds to PWM2), BB0(corresponds to PWM3), CT0(corresponds to

PWM4), CB0(corresponds to PWM5) are defined as the post output of the MASK circuit. Table 22-2-2-1 shows the relationship between HDCT and AT0/AB0/BT0/BB0/CT0/CB0.

Table 24-3-3-1 the Relationship between HDCT and AT0/AB0/BT0/BB0/CT0/CB0

	HAT	HAB	AT0	AB0		HAT	HAB	AT0	AB0
Bottom driving (MPOL=0)	0	0	0	0	Top driving (MPOL=1)	0	0	0	0
	0	1	PWM0	PWM1		0	1	1	0
	1	0	0	1		1	0	PWM1	PWM0
	1	1	0	0		1	1	0	0
	HBT	HBB	BT0	BB0		HBT	HBB	BT0	BB0
Bottom driving (MPOL=0)	0	0	0	0	Top driving (MPOL=1)	0	0	0	0
	0	1	PWM2	PWM3		0	1	1	0
	1	0	0	1		1	0	PWM3	PWM2
	1	1	0	0		1	1	0	0
	HCT	HCB	CT0	CB0		HCT	HCB	CT0	CB0
Bottom driving (MPOL=0)	0	0	0	0	Top driving (MPOL=1)	0	0	0	0
	0	1	PWM4	PWM5		0	1	1	0
	1	0	0	1		1	0	PWM5	PWM4
	1	1	0	0		1	1	0	0

- **Brake mode**

When MOTCMD=4, MASK operates in brake mode. The driver circuit's lower half bridge is conductive while the upper half bridge is closed in brake mode. Table 24-3-3-2 shows the result.

Table 24-3-3-2 AT0/AB0/BT0/BB0/CT0/CB0 Output in Brake Mode

Brake mode(MOTCMD=4)	AT0	AB0	BT0	BB0	CT0	CB0
MPOL=0	0	1	0	1	0	1
MPOL=1	1	0	1	0	1	0

- **No driver mode**

When MOTCMD=0, MASK circuit operates in no driver mode. The driver circuit's lower and upper half bridge are all closed in no driver mode. Table 24-3-3-3 shows the result.

Table 24-3-3-3 AT0/AB0/BT0/BB0/CT0/CB0 Output in No Driver Mode

No driver mode (MOTCMD=0)	AT0	AB0	BT0	BB0	CT0	CB0
	0	0	0	0	0	0

- **Fool Proofing**

If there are software errors or external inference such like ESD, which makes the output for AT0/AB0, BT0/BB0 and CT0/CB0 is high at the same time, the fool proofing circuit will force AT1/AB1, BT1/BB1,CT1/CB1 output to be low so that the circuit will not be short.

Table 24-3-3-4 Fool Proofing Circuit Truth Table

AT0	AB0	AT1	AB1
1	1	0	0
BT0	BB0	BT1	BB1
1	1	0	0
CT0	CB0	CT1	CB1
1	1	0	0

- **Polarity control**

Register MOTPLC controls whether AT1/AB1/BT1/BB1/CT1/CB1 outputs inverted phase. Each bit in register MOTPLC corresponds to one channel and makes the motor driver configuration more flexible .

24.3.4 Motor Abnormality Detection and Protection

There are many detection methods for different motor abnormalities. Either pause or fault mode can be selected by ZSE for protection when there are any abnormalities. In pause mode, when the motor is abnormal, no driver mode or brake mode can be selected. The motor can work normally again when the abnormality is cleared up. In fault mode, when the abnormality is cleared up, motor can not work normally unless the software disables and then enables the motor module again. The Pause and fault bit is the indicators for pause and fault status.

There are several abnormality source detections for the motor: fault pin detection 、 ADC exceeding threshold detection, comparator3 interrupt detection, Hall state error detection, capture counter exceeding threshold detection.

◆ **Fault pin detection**

Fault pin connects to external detection circuit and the efficient level is set by FTPOL. The Fault pin's filtering time is set by FTPFTS. When FTPME=1, Fault pin detection is enabled, and the default mode is pause mode. However, if FTPSE=1, fault pin detection is in fault mode (fault mode is enabled only when FTPME=1 and FTPSE=1).When fault pin efficient level comes, fault pin detection interrupt flag is set to 1 and the motor stops rotation.

◆ **ADC exceeding threshold detection**

ADC is used for over-current or over-voltage detection. The ADC threshold interrupt is introduced in ADC chapter. When ADC and motor are used together, setting PWMTM=0 will enable the function and setting PWMTS selects the triggering PWM channel. The ADC delaying time can be set by register MTGDL. When ADCME=1, pause mode is enabled. When ADCME=1 and ADCSE=1, fault mode is enabled. When ADC exceeds to threshold set by users, ADC exceeding threshold flag will force the motor enter pause or fault mode.

◆ **Comparator3 interrupt detection**

Comparator3 can be used for over-current or over-voltage detection for the motor similar to ADC. CPME=1 enables the pause mode, CPME=1 and CPSE=1enables fault mode.

◆ **Hall state error detection**

HLME=1 enables pause mode, HLME=1 and HLSE=1 enables fault mode. When the Hall device is abnormal, the motor stops and the Hall state error interrupt flag HLIF is set to 1.

◆ **Capture counter exceeding threshold detection**

Capture counter is used to detect the speed of the motor. The threshold for the capture counter can be set by users which also corresponds to the speed when motor rotation is locked. CTME=1 enables the pause mode, while CTME=1 and CTSE=1 enables fault mode. The capture threshold interrupt occurs and the motor stops when the rotation is locked.

24.4 Motor Control Register Description

Table 24-4-1 Register MOTCON

8060H	7	6	5	4	3	2	1	0
MOTCON	MOTEN	ZSE	MPOL	FTPME	ADCME	CPME	HLME	CTME
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	MOTEN	MOTOR module enable control, 1 enables it
6	ZSE	No driver mode enable control, when it is 1, if error occurs, the motor will not be driving
5	MPOL	top/bottom PWM output selection, 0 for the bottom, 1 for top
4	FTPME	Fault Pin detection enable control, when it is 1, the module enters no driver mode or brake mode when Fault Pin efficient signal detected
3	ADCME	ADC detection enable control, when it is 1, the module enters no driver mode or brake mode when ADC conversion result exceeds the threshold set by users
2	CPME	Comparator 3 interrupt flag detection enable control, when it is 1, the module enters no driver mode or brake mode when comparator 3 interrupt flag detected
1	HLME	Hall decoding result detection enable control, when it is 1, the module enters no driver mode or brake mode when there is Hall decoding error
0	CTME	Capture counter interrupt flag detection enable control, when it is 1, the module enters no driver mode or brake mode when capture counter interrupt flag detected

Table 24-4-2 Register MOTCFG

8061H	7	6	5	4	3	2	1	0

MOTCFG	FTIE	HLIE	-	FTPSE	ADCSE	CPSE	HLSE	CTSE
R/W	R/W	R/W	-	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	-	0	0	0	0	0

Bit number	Bit Symbol	Description
7	FTIE	Fault Pin interrupt enable control, if it is 1, interrupt occurs when efficient Fault Pin signal detected
6	HLIE	Hall decoding error interrupt enable control, if it is 1, interrupt occurs when there is Hall decoding error
5	-	-
4	FTPSE	Similar to FTPME function, if it is 1, the module will be locked in no driver mode or brake mode when Fault Pin efficient signal detected, resumes only after MOTEN is cleared and set to 1 again
3	ADCSE	Similar to ADCME function, if it is 1, the module will be locked in no driver mode or brake mode when ADC conversion result exceeds the threshold set by users, resumes only after MOTEN is cleared and set to 1 again
2	CPSE	Similar to CPME function, if it is 1, the module will be locked in no driver mode or brake mode when comparator 3 interrupt flag detected, resumes only after MOTEN is cleared and set to 1 again
1	HLSE	Similar to HLME function, if it is 1, the module will be locked in no driver mode or brake mode when there is Hall decoding error, resumes only after MOTEN is cleared and set to 1 again
0	CTSE	Similar to CTPME function, if it is 1, the module will be locked in no driver mode or brake mode when capture counter interrupt flag detected, resumes only after MOTEN is cleared and set to 1 again

Table 24-4-3 Register MTGCON

8062H	7	6	5	4	3	2	1	0
MTGCON	PWMTS[2:0]			PWMTM	-	-	-	-
R/W	R/W			R/W	-	-	-	-
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	说明
7~5	PWMTS	PWM Selection when PWM interrupt used as trigger source： 0：disables the function 1：PWM0 interrupt as the trigger edge 2：PWM1 interrupt as the trigger edge 3：PWM2 interrupt as the trigger edge 4：PWM3 interrupt as the trigger edge 5：PWM4 interrupt as the trigger edge 6：PWM5 interrupt as the trigger edge
4	PWMTM	PWM interrupt target selection：

		0：triggers the ADC conversion 1：triggers TIMER2 counting
3~0	-	-

Table 24-4-4 Register MHLCON

8063H	7	6	5	4	3	2	1	0
MHLCON	HTYPE	HLSPE	-	-	-	HDAT[2:0]		
R/W	R/W	R/W	-	-	-	R		
Initial Value	1	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7	HTYPE	Hall type selection： 0：60 degree 1：120 degree
6	HLSPE	Input source selection for comparator's digital filter： 0： comparator0's digital filter input, which comes from comparator0's output comparator1's digital filter input, which comes from comparator1's output comparator2's digital filter input, which comes from comparator2's output 1：comparator0's digital filter input, which comes from external I/O, P2.0 comparator1's digital filter input, which comes from external I/O, P2.2 comparator2's digital filter input, which comes from external I/O, P2.4
5~3	-	-

2~0	HDAT	Hall sensor decoding data：
		HDAT[2] <-> HCD
		HDAT[1] <-> HBD
		HDAT[0] <-> HAD

Table 24-4-5 Register MFPCON

8064H	7	6	5	4	3	2	1	0
MFPCON	FTPOL	FTPFTS[6:0]						
R/W	R/W	R/W						
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7	FTPOL	Fault Pin efficient level selection, 0 indicates high level efficient, 1 indicates low level efficient
6~0	FTPFTS	Fault Pin filter configuration, 128 system clocks at most

Table 24-4-6 Register MOTCMD

8065H	7	6	5	4	3	2	1	0
MOTCMD	-	-	-	-	-	CMD[1:0]		

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

Bit number	Bit Symbol	Description
7~3	-	-
2~0	CMD	Motor control driver： 001： manual control 010： revolve forward 011： revolve backward 100： brake Others： idle

Table 24-4-7 Register MTGDL

8066H	7	6	5	4	3	2	1	0
MTGDL	MTGDL[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~0	MTGDL	ADC conversion start delaying register. Once valid trigger signal generated by PWM comes, the delay counter will be set to 0 and starts to count until it reaches MTGDL value. The maximum for it is 256 system clocks.

Table 24-4-8 Register MOTIF

8067H	7	6	5	4	3	2	1	0
MOTIF	FAULT	PAUSE	-	-	-	-	FTIF	HLIF
R/W	R	R	-	-	-	-	R	R
Initial Value	0	0	-	-	-	-	0	0

Bit number	Bit Symbol	Description
7	FAULT	Fault flag, 1 indicates the motor is in brake or no driver mode, resumes only when error source cleared and MOTEN cleared
6	PAUSE	Pause flag, 1 indicates the motor is in brake or no driver mode, resumes when error source cleared
5~2	-	-
1	FTIF	Fault Pin interrupt flag, 1 indicates the fault pin, cleared when 1 is written to it

0	HLIF	Hall decoding error interrupt flag, 1 indicates the error, cleared when 1 is written to it

Table 24-4-9 Register HDCT0

8068H	7	6	5	4	3	2	1	0
HDCT0	-	-	HDCT0[5:0]					
R/W	-	-	R/W					
Initial Value	-	-	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~6	-	-
5~0	HDCT0	BIT5-BIT4：HCT-HCB BIT3-BIT2：HBT-HBB BIT1-BIT0：HAT-HAB

Table 24-4-10 Register HDCT1

8069H	7	6	5	4	3	2	1	0
HDCT1	-	-	HDCT1[5:0]					
R/W	-	-	R/W					
Initial Value	-	-	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~6	-	-
5~0	HDCT1	BIT5-BIT4：HCT-HCB BIT3-BIT2：HBT-HBB BIT1-BIT0：HAT-HAB

Table 24-4-11 Register HDCT2

806AH	7	6	5	4	3	2	1	0
HDCT2	-	-	HDCT2[5:0]					
R/W	-	-	R/W					
Initial Value	-	-	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~6	-	-
5~0	HDCT2	BIT5-BIT4：HCT-HCB BIT3-BIT2：HBT-HBB BIT1-BIT0：HAT-HAB

Table 24-4-12 Register HDCT3

806BH	7	6	5	4	3	2	1	0
HDCT3	-	-	HDCT3[5:0]					
R/W	-	-	R/W					
Initial Value	-	-	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~6	-	-
5~0	HDCT3	BIT5-BIT4：HCT-HCB BIT3-BIT2：HBT-HBB BIT1-BIT0：HAT-HAB

Table 24-4-13 Register HDCT4

806CH	7	6	5	4	3	2	1	0
HDCT4	-	-	HDCT4[5:0]					
R/W	-	-	R/W					
Initial Value	-	-	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~6	-	-
5~0	HDCT4	BIT5-BIT4：HCT-HCB BIT3-BIT2：HBT-HBB BIT1-BIT0：HAT-HAB

Table 24-4-14 Register HDCT5

806DH	7	6	5	4	3	2	1	0
HDCT5	-	-	HDCT5[5:0]					
R/W	-	-	R/W					
Initial Value	-	-	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~6	-	-
5~0	HDCT5	BIT5-BIT4：HCT-HCB BIT3-BIT2：HBT-HBB BIT1-BIT0：HAT-HAB

Table 24-4-15 Register HDCT6

806EH	7	6	5	4	3	2	1	0
HDCT6	-	-	HDCT6[5:0]					
R/W	-	-	R/W					
Initial Value	-	-	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~6	-	-
5~0	HDCT6	BIT5-BIT4：HCT-HCB BIT3-BIT2：HBT-HBB BIT1-BIT0：HAT-HAB

Table 24-4-16 Register HDCT7

806FH	7	6	5	4	3	2	1	0
HDCT7	-	-	HDCT7[5:0]					
R/W	-	-	R/W					
Initial Value	-	-	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~6	-	-
5~0	HDCT7	BIT5-BIT4：HCT-HCB BIT3-BIT2：HBT-HBB BIT1-BIT0：HAT-HAB

Table 24-4-17 Register HDCT8

8070H	7	6	5	4	3	2	1	0
HDCT8	-	-	HDCT8[5:0]					
R/W	-	-	R/W					
Initial Value	-	-	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~6	-	-
5~0	HDCT8	BIT5-BIT4：HCT-HCB BIT3-BIT2：HBT-HBB BIT1-BIT0：HAT-HAB

Table 24-4-18 Register HDCT9

8071H	7	6	5	4	3	2	1	0
HDCT9	-	-	HDCT9[5:0]					
R/W	-	-	R/W					
Initial Value	-	-	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~6	-	-
5~0	HDCT9	BIT5-BIT4：HCT-HCB BIT3-BIT2：HBT-HBB BIT1-BIT0：HAT-HAB

Table 24-4-19 Register HDCT10

8072H	7	6	5	4	3	2	1	0
HDCT10	-	-	HDCT10[5:0]					
R/W	-	-	R/W					
Initial Value	-	-	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~6	-	-
5~0	HDCT10	BIT5-BIT4：HCT-HCB BIT3-BIT2：HBT-HBB BIT1-BIT0：HAT-HAB

Table 24-4-20 Register HDCT11

8073H	7	6	5	4	3	2	1	0
HDCT11	-	-	HDCT11[5:0]					
R/W	-	-	R/W					
Initial Value	-	-	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~6	-	-
5~0	HDCT11	BIT5-BIT4：HCT-HCB BIT3-BIT2：HBT-HBB BIT1-BIT0：HAT-HAB

Table 24-4-21 Register MOTPLC

8074H	7	6	5	4	3	2	1	0
MOTPLC	-	-	MOTPLC[5:0]					
R/W	-	-	R/W					
Initial Value	-	-	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~6	-	-
5~0	MOTPLC	When the motor circuit is enabled： BIT5=0/1： PWM5 output positive/negative BIT4=0/1： PWM4 output positive/ negative BIT3=0/1： PWM3 output positive/ negative BIT2=0/1： PWM2 output positive/ negative BIT1=0/1： PWM1 output positive/ negative BIT0=0/1： PWM0 output positive/ negative

25 Touch Key

25.1 Function Introduction

The touch function module of CA51F2 series chip has superior anti-interference performance, and can pass EFT, CS and other tests. The touch module can support up to 24 channels. When in use, the TK_CAP pin needs to be connected to a Cx capacitor. The capacitance range is 10nF~47nF, and the capacitance accuracy is within 10%. It is recommended to use polyester capacitors, X7R capacitors or NPO chip capacitors . Cx can directly affect the touch sensitivity. The smaller the Cx capacitance value, the lower the sensitivity, and the larger the capacitance value, the higher the sensitivity.

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

25.2 Main Features

- Great anti-jamming performance which meets the EMC(CS) Standard
- Supports 24 channels at most
- Supports low power consumption mode
- Touch interrupt supported
- Clock division supported for charging/discharging
- Supports manual control and automatic mode
- Selective levels for comparator’s threshold
- Touch can set internal charging and internal reference, which can effectively suppress low frequency interference of power supply
- Support touch pin and LED drive pin multiplexing
- Built-in waterproof compensation mechanism
- Waking up threshold configurable in STOP mode

25.3 Architecture

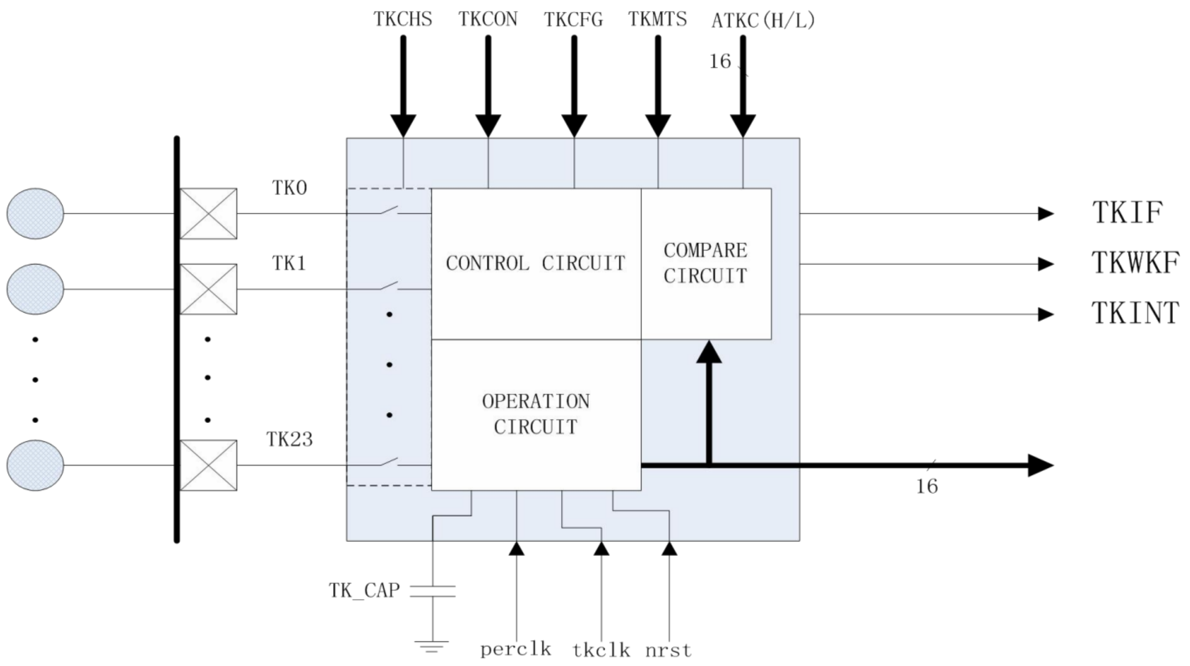


Figure 25-3-1 Touch Key Module Architecture

25.4 Function Description

25.4.1 Manual Control Mode and Automatic Mode

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

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

25.4.2 Touch Key Clock Frequency Division

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

25.4.3 Low Power consumption Mode

In order to realize the low power consumption application of the touch function, the touch module has designed a corresponding power saving mechanism. In the STOP mode, as long as the touch charging and discharging clock source TFRC and the low-speed clock (IRCL or XOSCL) are turned on, the touch module can maintain normal charging and discharging and counting. When the touch acquisition is completed, if TWKE=0, the touch acquisition completion interrupt will wake up the CPU, and the software can read the touch data after the CPU wakes up, and then enter the STOP mode again. In addition, the touch module is also designed with a touch threshold automatic comparison function. The user can set the trigger threshold of a group of channels through the threshold setting register. In the STOP mode, the touch controller can still compare the collected touch data with the threshold. When the touch data exceeds at the threshold, if TWKE=1, a threshold trigger interrupt will be generated and the CPU will be awakened. After the CPU is awakened, normal touch collection and judgment can be performed.

25.4.4 Touch Button Shared LED Driver

The shared LED driver of the touch buttons can realize the control of N touch buttons and N touch indicators, and only (N+1) pins are needed. Among them, the touch button and the LED drive positive terminal control share a pin, the LED negative terminal is connected to COM, and the touch and LED control are implemented in a time-sharing manner.

Each touch has a separate control bit TLENx (x=0~19, corresponding to TK0~TK19) to enable the shared LED drive function. It should be noted that the corresponding touch pin function must be turned on. After the shared LED is enabled, TLDATx (x=0~19, corresponding to LED0~LED19) can independently control the on and off of each LED. The COM pin can be selected by TLCOS.

Touch data collection and LED control are implemented in a time-sharing manner, where the time of touch data collection is defined by TLCNTK, and the time of LED scanning is defined by TLCNTL. Note that the time defined by TLCNTK is the total time collected by each group of touches. The number of touch channels in each group is 1~6. When the actual touch time is greater than the defined time, a TLERR interrupt will be generated. When the counter counts to the time defined by TLCNTK, a TNKOV interrupt is generated. After the touch acquisition phase is completed, it enters the LED scanning phase. TLCNTL defines the time of the LED scanning phase, this time will affect the duty cycle of the LED scanning, that is, it will affect the brightness of the LED, which can be adjusted as needed during the application. In the LED scanning phase, when the counter counts to the time defined by TLCNTL, a TLLOV interrupt will be generated. At this point, a complete touch shared LED cycle is completed. The following is a schematic diagram of the working stage.

Important reminder: In the application of the multiplexing mode of the touch pin and the LED drive pin, due to the diode junction capacitance of the LED lamp itself, the junction capacitance of different types of LED lamps is quite different, and this junction capacitance is when the LED lamp is on or off. The performance may be inconsistent (especially the white LED lamp is more obvious), this junction capacitance and its inconsistency will have an adverse effect on the touch, so when applying this mode, the LED lamp used should be carefully selected and cannot be replaced randomly after mass production. Varieties of LED lights.

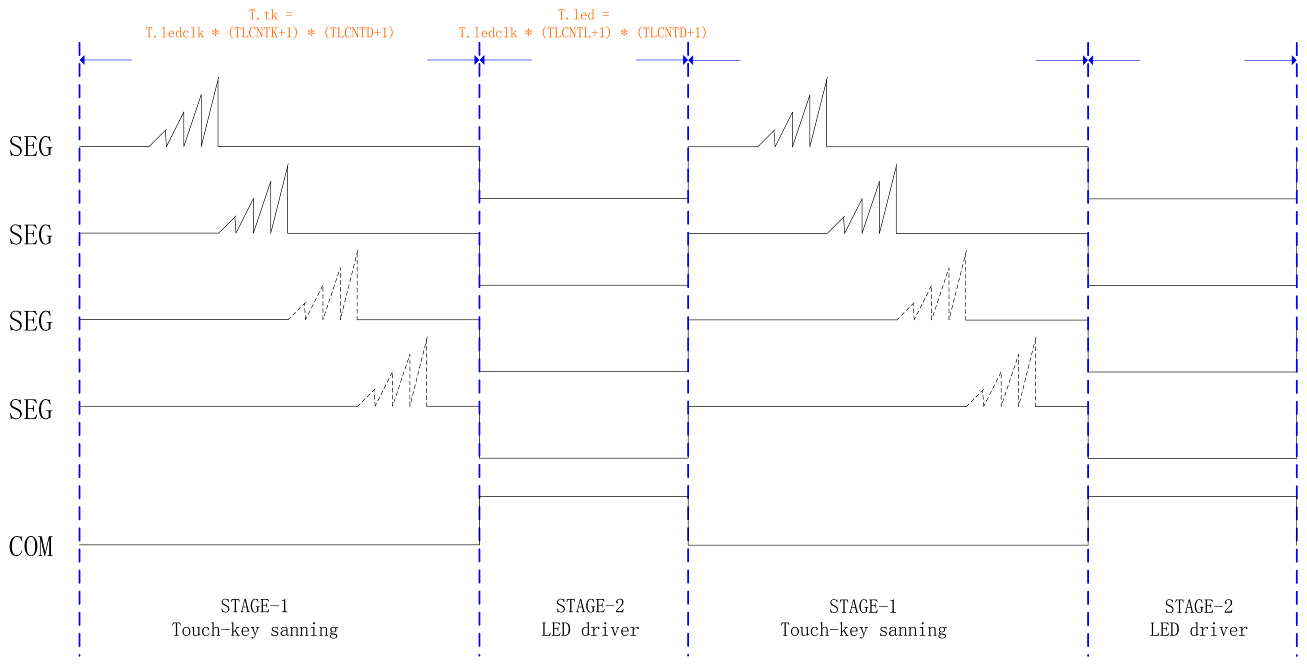


Figure 25-4-1 Schematic Diagram of Touch Shared LED Drive

25.4.5 Touch Internal Reference and Internal Op Amp

The touch module integrates an operational amplifier inside. The internal operational amplifier can be selected as the charging power supply for the touch keys through TKPWS (TKPWC[1]), and the charging voltage is selected through VDS (TKPWC[5:4]). In addition, the touch can also select the internal reference as the threshold voltage of the touch internal comparator through TKCVS (TKPWC[0]), and the internal reference voltage can be selected through VIRS (TKPWC[3:2]).

25.4.6 Touch Waterproof Compensation Mechanism

Touch is designed with a waterproof compensation mechanism, which can be turned on by TKPC (TKPWC[7:6]) as 2. After this function is turned on, the unselected touch pins will synchronously output the compensation waveform with the same charging frequency, which can effectively reduce the influence of parasitic capacitance between the touch buttons and realize the waterproof effect.
Note: When the waterproof compensation is turned on, it is best to choose an external power supply for the touch charging power supply.

25.5 Register Description

Table 25-5-1 Register TKCON

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

Bit number	Bit Symbol	Description
7	TKST	Data collection start enable control, 1 enables it, cleared automatically after the data collection
6	TKIE	TK interrupt enable control, 1 enables it
5	TMEN	Start mode selection 0: enabled by TKST 1: enabled by Timer
4	TWKE	Interrupt trigger selection 0: interrupt triggered when sampling is done 1: interrupt triggered when data collected exceeds the threshold
3	-	-

2~0	VRS	Reference voltage selection for comparator's threshold voltage (the threshold voltage is directly proportional with VDD) 0: maximum threshold voltage … 7: minimum threshold voltage

Table 25-5-2 Register TKCFG

C2H	7	6	5	4	3	2	1	0
TKCFG	TKDIV			TKTMS				
R/W	R/W			R/W				
Initial Value	1	1	1	1	1	1	1	1

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

Table 25-5-3 Register TKPWC

8103H	7	6	5	4	3	2	1	0
TKPWC	TKPC		VDS		VIRS		TKPWS	TKCVS
R/W	R/W		R/W		R/W		R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~6	TKPC	Touch button unsampled channel output control 00: Suspended 01: Low output 10: Output compensation Note: 1. This function is only valid for pins configured as touch key functions. 2. If the shared LED drive function is enabled, then the pin selected as the LED drive will disable this function.
5~4	VDS	Internal op amp output voltage selection 00：2V 01：2.5V 10：3V 11：4V
3~2	VIRS	Internal voltage reference selection 00：1.0V 01：1.5V 10：2.0V 11：2.5V
1	TKPWS	Charging power selection 0: select external power supply 1: Select internal op amp output
0	TKCVS	Charging reference voltage selection 0: select external voltage reference 1: Select internal voltage reference

Table 25-5-4 Register TKMTS

C4H	7	6	5	4	3	2	1	0
TKMTS	TKMTS[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~0	TKMTS	The start time setting register in timing mode the start time=(TKMTS+1) × 32 × low speed clock cycle period If the low speed clock’s frequency is 32.768K, the start time ranges from 0.977ms to 250ms.

Table 25-5-5 Register TKCHS

C4H	7	6	5	4	3	2	1	0
TKCHS	POL	NPOL	TKPS					

R/W	R/W	R/W	R/W					
Initial Value	0	0	0	0	0	0	0	0
Note：TKCHS is register with index, INDEX=0~5 indicates TKCHS0~TKCHS5 respectively								

Bit number	Bit Symbol	Description
7	POL	ATKnC direction setting for threshold comparison 0：interrupt when the data collected is less than the threshold 1：interrupt when the data collected is greater than the threshold
6	NPOL	ATKnN direction setting for threshold comparison 0：interrupt when the data collected is less than the threshold 1：interrupt when the data collected is greater than the threshold
5~0	TKPS	Channel selection 000000：disable TK0~TK23 000001：TK0 selected 000010：TK1 selected 000011：TK2 selected ...... 011000：TK23 selected 011001：Internal reference capacitor selected

Table 25-5-6 Register ATKC

C5H	7	6	5	4	3	2	1	0
ATKCL	ATKC[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
C6H	7	6	5	4	3	2	1	0
ATKCH	ATKC[15:8]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Note：ATKC is register with index, INDEX=0~5 indicates ATKC0~ATKC5 respectively								

Bit number	Bit Symbol	Description
15~0	ATKC	Compare threshold setting register, when TWKE=1, ATKC0~ATKC5 will be compared with TKMS0~TKMS5 automatically

Table 25-5-7 Register ATKN

8092H	7	6	5	4	3	2	1	0
ATKNL	ATKN[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
8093H	7	6	5	4	3	2	1	0
ATKNH	ATKN[15:8]							

R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Note： ATKC is register with index, INDEX=0~5 indicates ATKC0~ATKC5 respectively								

Bit number	Bit Symbol	Description
15~0	ATKN	Compare threshold setting register, when TWKE=1, ATK0N~ATK5N will be compared with TK0MS~TK5MS automatically

Table 25-5-8 Register TKMS

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

Bit number	Bit Symbol	Description
15~0	TKMS	Touch key sampling data register

Table 25-5-9 Register TKIF

C7H	7	6	5	4	3	2	1	0
TKIF	-		TKIF5	TKIF4	TKIF3	TKIF2	TKIF1	TKIF0
R/W	-		R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	-	-	0	0	0	0	0	0

Bit number	Bit Symbol	Description
7~6	-	-
5~0	TKIFx(x=5~0)	TK data collection interrupt flag, the bits correspond to 6 channels in order. When TWKE=1, TKIFx implies the data exceeds the ATKC or ATKN threshold

Table 25-5-10 Register TKMAXF

8090H	7	6	5	4	3	2	1	0
TKMAXF	-	-	TKMXF5	TKMXF4	TKMXF3	TKMXF2	TKMXF1	TKMXF0
R/W	-	-	R	R	R	R	R	R
Initial Value	-	-	0	0	0	0	0	0

Bit number	Bit Symbol	Description

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

Table 25-5-11 Register TKMINF

8091H	7	6	5	4	3	2	1	0
TKMINF	-	-	TKMNF5	TKMNF4	TKMNF3	TKMNF2	TKMNF1	TKMNF0
R/W	-	-	R	R	R	R	R	R
Initial Value	-	-	0	0	0	0	0	0

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

Table 25-5-12 Register TLEN

8106H	7	6	5	4	3	2	1	0
TLEN (INDEX=0)	TLEN7	TLEN6	TLEN5	TLEN4	TLEN3	TLEN2	TLEN1	TLEN0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0
TLEN (INDEX=1)	TLEN15	TLEN14	TLEN13	TLEN12	TLEN11	TLEN10	TLEN9	TLEN8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0
TLEN (INDEX=2)	TLEN23	TLEN22	TLEN21	TLEN20	TLEN19	TLEN18	TLEN17	TLEN16
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

Note：
TLENx=1 (x=0,1,2,...,23)， To enable LEDx, the touch button function must be selected corresponding to the TKx pin.

Bit number	Bit Symbol	Description
3~0 (INDEX=2)	TLEN23~TLEN16	LED23~LED16 enable ， 1 enable it
7~0 (INDEX=1)	TLEN15~TLEN8	LED15~LED8 enable， 1 enable it
7~0 (INDEX=0)	TLEN7~TLEN0	LED7~LED0 enable， 1 enable it

25.6 Touch Key Control Example

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

26 Low Voltage Detection(LVD)

26.1 Function Introduction

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

Figure 26-1-1 shows the architecture of LVD.

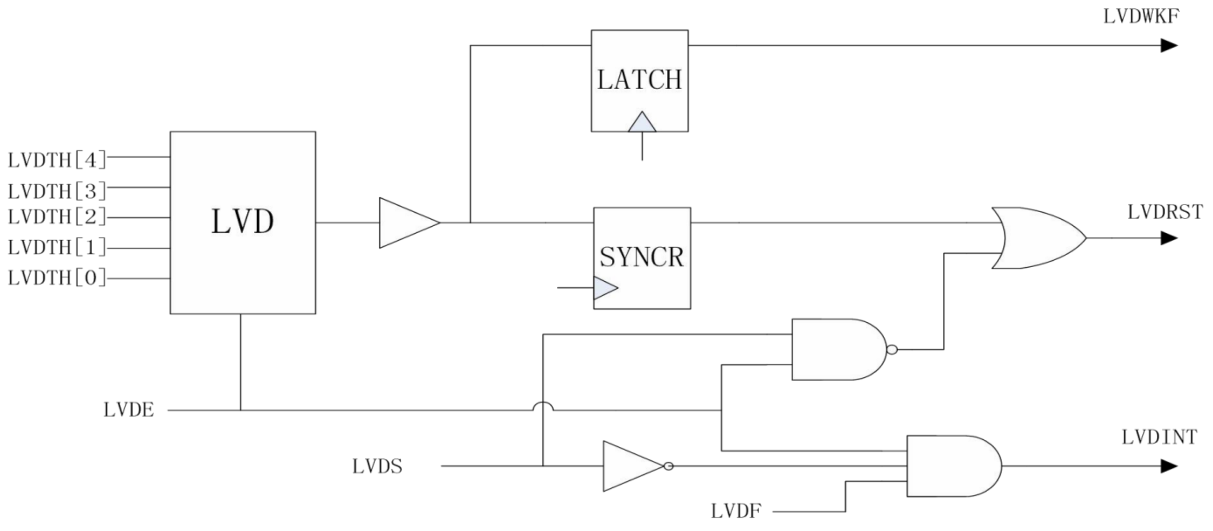


Figure 26-1-1 LVD Schematic

26.2 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.

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

26.3 Register Description

Table 26-3-1 Register LVDCON

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

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

26.4 LVD Control Example

LVD interrupt example

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

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

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

LVD reset example

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

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

27 Multiplier/Divider Unit(MDU)

27.1 Function Introduction

There are four types operation for MDU: 32 bit dividing 32 bit, 16 bit multiplying 16 bit, left shift, right shift. It takes one clock cycle periods to complete multiplication and shift operation and 8 clock cycle periods for division.

27.2 Architecture

Figure 27-2-1 shows the principle of multiplication circuit.

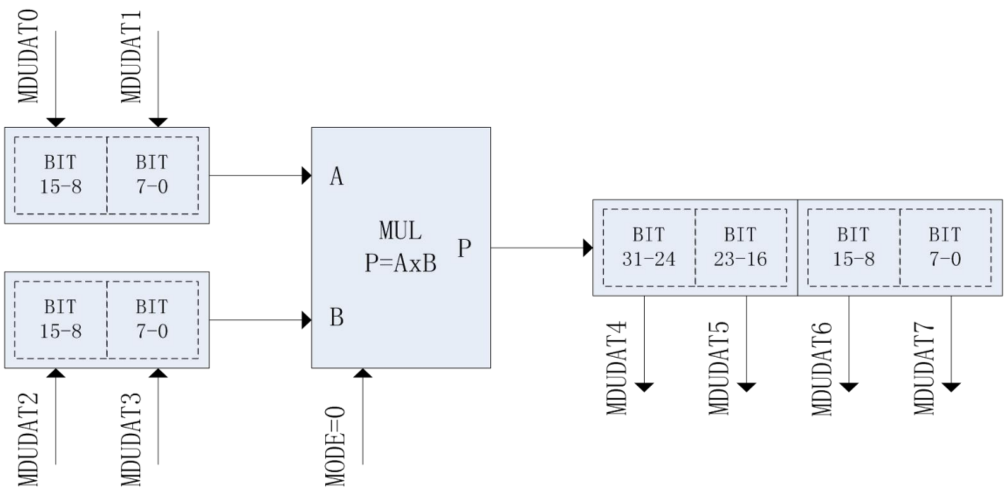


Figure 27-2-1 Multiplication Circuit Schematic

Figure 27-2-2 shows the principle of division circuit.

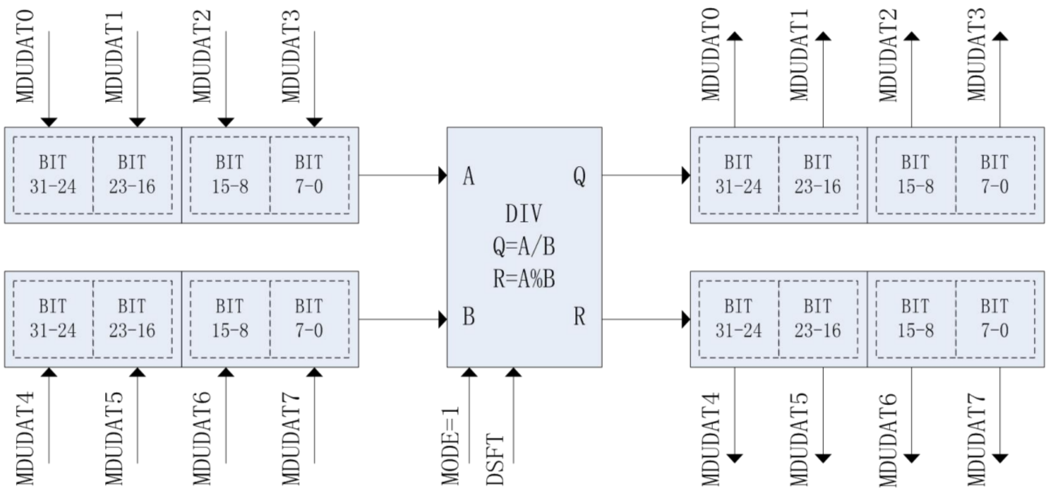


Figure 27-2-2 Division Circuit Schematic

Figure 27-2-3 shows the principle of shift circuit.

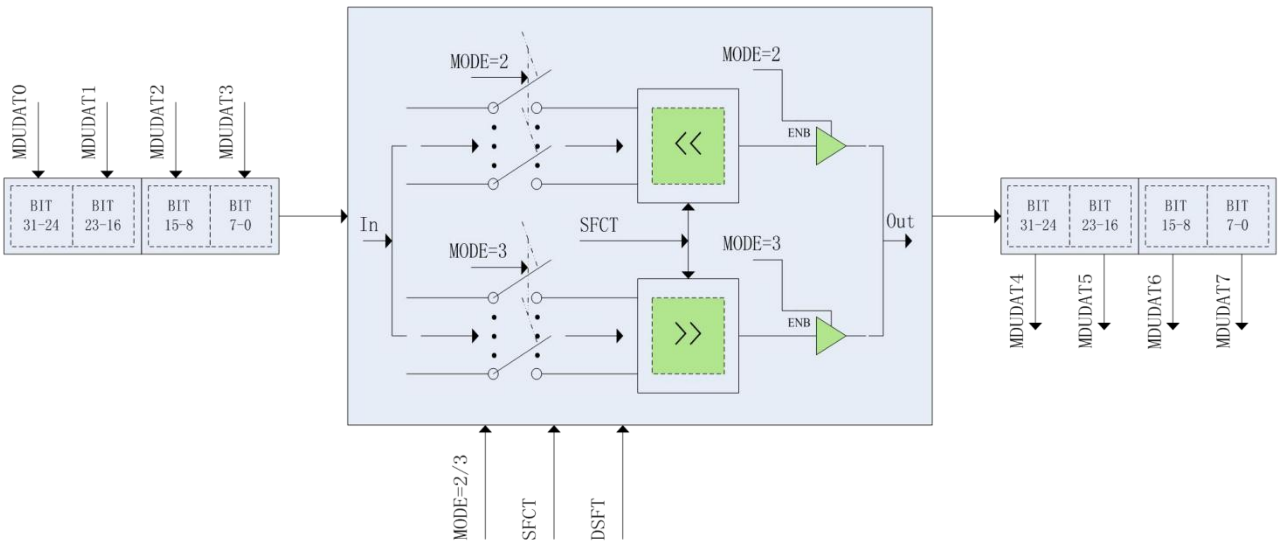


Figure 27-2-3 Shift Circuit Schematic

27.3 Function Description

27.3.1 Multiplier

When MODE=0, MDU is set as a 16bit×16bit multiplier. The multiplicand will be written to register MDUDAT0 and MDUDAT1, with multiplier written to register MDUDAT2 and MDUDAT3. It takes one clock cycle to complete the operation. The product will be obtained instantly after the multiplicand and multiplier are written to the registers. The product will be stored in register MDUDAT4, MDUDAT5, MDUDAT6 and MDUDAT7.

27.3.2 Divider

When MODE=1, MDU is set as a 32 bit ÷32 bit divider. The numerator will be written to register MDUDAT0, MDUDAT1, MDUDAT2 and MDUDAT3, with denominator written to register MDUDAT4, MDUDAT5, MDUDAT6 and MDUDAT7. DSFT must be set to 1 start the operation after the numerator and denominator are written to the registers. DSFT will be cleared automatically after the operation. The quotient will be stored in register MDUDAT0, MDUDAT1, MDUDAT2 and MDUDAT3, with remainder stored in register MDUDAT4, MDUDAT5, MDUDAT6 and MDUDAT7. Since it takes 8 clock cycles to do the division, hence users have to wait for 8 clock cycles or until DSFT is cleared and then read the result.

27.3.3 Shifter

When MODE=2 or MODE=3, MDU is set as a shifter(left shift when MODE=2, right shift when MODE=3). The number of bits to shift is set by SFCT, with the 32-bit data to be shifted written to register MDUDAT0, MDUDAT1, MDUDAT2 and MDUDAT3. Setting DSFT=1 will start the operation. Due to it takes one clock cycle only, the result can be read immediately after DSFT=1. The result will be stored in register MDUDAT4, MDUDAT5, MDUDAT6 and MDUDAT7.

27.4 Register Description

Table 27-4-1 Register MDUCON

E6H	7	6	5	4	3	2	1	0
MDUCON	MODE[1:0]		DSFT	SFCT[4:0]				
R/W	R/W		R/W	R/W				
Initial Value	0	0	0	-	0	0	0	0

Bit number	Bit Symbol	Description
7~6	MODE	Operation mode selection 00：multiplication 01：division 10：left shift 11：right shift
5	DSFT	Enable control for division and shift operation. 1 starts the operation. Invalid for multiplication.
4~0	SFCT	The number of times to shift = (SFCT + 1). Invalid for multiplication and division.

Table 27-4-2 Register MDUDAT

E7H	7	6	5	4	3	2	1	0
MDUDAT	MDUDAT[7:0]							
R/W	R/W							
Initial Value	0	0	0	0	0	0	0	0
Note：MDUDAT is register with index, INDEX=0~7 correspond to MDUDAT0～MDUDAT7 respectively								

Bit number	Bit Symbol	Description
7~0	MDUDAT	MDU data storage. No order requirement for reading/writing MDUDAT0th～ MDUDAT7th For multiplication, MDUDAT0：15th～8th bit of the multiplicand MDUDAT1： 7th～0th bit of the multiplicand MDUDAT2： 15th～8th bit of the multiplier MDUDAT3： 7th～0th bit of the multiplier For product, MDUDAT4： 31th～24th bit of the product MDUDAT5： 23th～16th bit of the product MDUDAT6： 15th～ 8th bit of the product MDUDAT7th： 7th～ 0th bit of the product

		For division, MDUDAT0th： 31th～24th bit of the numerator； MDUDAT1： 23th～16th bit of the numerator； MDUDAT2： 15th～8th bit of the numerator； MDUDAT3： 7th～0th bit of the numerator. MDUDAT4： 31th～24th bit of the denominator； MDUDAT5： 23th～16th bit of the denominator； MDUDAT6： 15th～ 8th bit of the denominator； MDUDAT7th： 7th～ 0th bit of the denominator Division result： MDUDAT0th： 31th～24th bit of the quotient； MDUDAT1： 23th～16th bit of the quotient； MDUDAT2： 15th～8th bit of the quotient； MDUDAT3： 7th～0th bit of the quotient. MDUDAT4： 31th～24th bit of the remainder； MDUDAT5： 23th～16th bit of the remainder； MDUDAT6： 15th～ 8th bit of the remainder； MDUDAT7th： 7th～0th bit of the remainder. For shift , MDUDAT0th：15th～8th bit of the source data； MDUDAT1： 7th～0th bit of the source data； MDUDAT2： 15th～8th bit of the source data； MDUDAT3： 7th～0th bit of the source data Shift result ： MDUDAT4：31th～24th bit of the destination data； MDUDAT5：23th～16th bit of the destination data； MDUDAT6：15th～ 8th bit of the destination data； MDUDAT7th：7th～ 0th bit of the destination data Note： For division operation, if the denominator is 0th, then the module will not do the operation. When DSFT is 1, overwriting the numerator or denominator will not influence the result for division.

27.5 MDU Control Example

Define the union below first：

```
------------------------------------------------------------------------------------------
typedef union
{
        unsigned long int     dwVal;
        unsigned int          wVal[2];
        unsigned char         bVal[4];
}
  DWORD_UNION;

typedef union
{
        unsigned int     wVal;
        unsigned char  bVal[2];
}
  WORD_UNION;
------------------------------------------------------------------------------------------
```

◆ **Example for Multiplication**

For instance , 65535 is multiplied by 1000, the program is like：

```
------------------------------------------------------------------------------------------
#define MOD_MULT       (0<<6)
void Mult(void)
{
            WORD_UNION Faciend;              //the multiplicand
            WORD_UNION Multiplier;           //the multiplier
            DWORD_UNION Product;             //product

            Faciend.wVal = 65535;
            Multiplier.wVal = 1000;

            MDUCON          = MOD_MULT;  //set MDU to multiplication mode

            INDEX = 0;
            MDUDAT = Faciend.bVal[0]; //write the higher 8 bits of the multiplicand
            INDEX = 1;
            MDUDAT = Faciend.bVal[1]; //write the lower of the multiplicand
             INDEX = 2;
            MDUDAT = Multiplier.bVal[0];//write the higher 8 bits of the multiplier
            INDEX = 3;
            MDUDAT = Multiplier.bVal[1];//write the lower 8 bits of the multiplier
            INDEX = 4;
            Product.bVal[0] = MDUDAT; //read the 24th~31th bit of the product
            INDEX = 5;
            Product.bVal[1] = MDUDAT; //read the 16th~23th bit of the product
            INDEX = 6;
            Product.bVal[2] = MDUDAT; //read the 8th~15th bit of the product
            INDEX = 7;
            Product.bVal[3] = MDUDAT; //read the 0th~7th bit of the product
}
------------------------------------------------------------------------------------------
```

◆ **Example for Division**

For instance , 0xFFFFFFFF is divided by 0x10000000, the program is like：

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

#define DSFT     (1<<5)
void Divid(void)
{
            DWORD_UNION Dividend;         //numerator
            DWORD_UNION Divisor;                      //denominator
            DWORD_UNION Quotient;         //quotient
            DWORD_UNION Remainder;        //remainder

            Dividend.dwVal = 0xffffffff;
            Divisor.dwVal =  0x10000000;

            MDUCON          = MOD_DIV; //set MDU as the division mode

            INDEX = 0;
            MDUDAT = Dividend.bVal[0];      //write 24th~31th bit of the numerator
            INDEX = 1;
            MDUDAT = Dividend.bVal[1];      //write 16th~23th bit of the numerator
            INDEX = 2;
            MDUDAT = Dividend.bVal[2];      //write 8th~15th bit of the numerator
             INDEX = 3;
            MDUDAT = Dividend.bVal[3];      //write 0th~7th bit of the numerator

            INDEX = 4;
            MDUDAT = Divisor.bVal[0]; //write 24th~31th bit of the denominator
            INDEX = 5;
            MDUDAT = Divisor.bVal[1]; //write 16th~23th bit of the denominator
            INDEX = 6;
            MDUDAT = Divisor.bVal[2]; //write 8th~15th bit of the denominator
            INDEX = 7;
            MDUDAT = Divisor.bVal[3]; //write 0th~7th bit of the denominator

            MDUCON          |=   DSFT;          //enables the division operation
            while(MDUCON & DSFT);           //wait for the result

            INDEX = 0;
            Quotient.bVal[0] = MDUDAT;//read the 24th~31th bit of quotient
             INDEX = 1;
            Quotient.bVal[1] = MDUDAT;//read the 16th~23th bit of quotient
            INDEX = 2;
            Quotient.bVal[2] = MDUDAT;//read the 8th~15th bit of quotient
            INDEX = 3;
            Quotient.bVal[3] = MDUDAT;//read the 0th~7th bit of quotient

            INDEX = 4;
            Remainder.bVal[0]= MDUDAT;     //read the 24th~31th bit of remainder
            INDEX = 5;
            Remainder.bVal[1]= MDUDAT;     //read the 16th~23th bit of remainder
            INDEX = 6;
            Remainder.bVal[2]= MDUDAT;     //read the 8th~15th bit of remainder
            INDEX = 7;
            Remainder.bVal[3]= MDUDAT;     //read the 0th~7th bit of remainder
}
      ------------------------------------------------------------------------------------------
```

◆ **Example for Shift Operation**

For instance , 0x88880001 left(or right) shift 8 bits,   the program is like：

```
------------------------------------------------------------------------------------------
#define MOD_SHIFT_LEFT          (2<<6)
#define MOD_SHIFT_RIGHT         (3<<6)
#define DSFT      (1<<5)
void Shift(void)
{
            DWORD_UNION SourceData;         //source data
            DWORD_UNION DestinationData; //destatioinn data

            MDUCON = MOD_SHIFT_LEFT|8;              //set the left shift and number of times
            //MDUCON = MOD_SHIFT_RIGHT|8; //set the right shift and number of times
            SourceData.dwVal = 0x88880001;

            INDEX = 0;
            MDUDAT = SourceData.bVal[0]; //write 24th~31th bit of the source data
            INDEX = 1;
```

```
        MDUDAT = SourceData.bVal[1]; //write 16th~23th bit of the source data
        INDEX = 2;
        MDUDAT = SourceData.bVal[2]; //write 8th~15th bit of the source data
        INDEX = 3;
        MDUDAT = SourceData.bVal[3]; //write 0th~7th bit of the source data
        MDUCON |= DSFT;         //enables shift operation

        INDEX = 4;
        DestinationData.bVal[0] = MDUDAT;       //read 24th~31th bit of the destination data
        INDEX = 5;
        DestinationData.bVal[1] = MDUDAT;       //read 16th~23th bit of the destination data
        INDEX = 6;
        DestinationData.bVal[2] = MDUDAT;       //read 8th~15th bit of the destination data
        INDEX = 7;
        DestinationData.bVal[3] = MDUDAT;       //read 0th~7th bit of the destination data
}
--------------------------------------------------------------------------------------------
```

28 Program Download and Simulation

28.1 Program Download

CA51F2 Series chip download programs using ISP method. The chip can connect to the download tool with UART port. Any of the UART ports can be used for ISP.

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

28.2 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.

29 Electrical Specification

29.1 Limit Parameter

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

Note：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.

29.2 DC Electrical Specification

Parameter	symbol	Typical			Unit	Condition
VDD		1.8V	3.3	5.5	V	
Working current	Iop1	2.92	3.46	3.49	mA	The system clock is XOSCH(24MHz), with other clocks disabled, LDO is set to the default value (high power mode, output voltage is 1.61V), No load for all the output pins. No floating for digital input pins. All the peripherals are disabled, with CPU executing instruction NOP
	Iop2	0.627	0.713	0.719	mA	The system clock is IRCH(3.6864MHz), with other clocks disabled, LDO is set to the default value (high power mode, output voltage is 1.61V), No load for all the output pins. No floating for digital input pins. All the peripherals are disabled, with CPU executing instruction NOP
	Iop3	2.78	3.29	3.31	mA	The system clock is PLL output with frequency multiplied by 6, and the reference clock is IRCH(3.6864MHz). Other clocks are disabled. LDO is set to the default value (high power mode, output voltage is 1.61V), No load for all the output pins. No floating for digital input pins. All the peripherals are disabled, with CPU executing instruction NOP

	Iop4	36.2	38.2	38.9	uA	The system clock is IRCL(131KHZ), with other clocks disabled. LDO is set to the default value (high power mode, output voltage is 1.61V), No load for all the output pins. No floating for digital input pins. All the peripherals are disabled, with CPU executing instruction NOP
	Iop5	23.6	24.6	25.2	uA	The system clock is XOSCL (32.768kHZ), other clocks are off, the LDO is set to low power mode, the output voltage is 1.61V, all output pins are unloaded, all digital input pins are not floating, all peripherals are off, and the CPU executes NOP instructions
	Iop6	26.1	29.0	31.8	uA	The system clock is XOSCL (32.768kHZ), other clocks are off, the LDO is set to low power mode, the output voltage is 1.61V, the LCD driver is turned on (without external LCD panel), and the LCD is set to minimum current drive, 1/3bias, 1/4duty , LCD clock is XOSCL, LCD_CAD is off (CAD_MOD=0), all LCD pins are on, all other output pins are unloaded, all digital input pins are not floating, and other peripherals are off
Current for STOP mode	I_{stp}	5.7	6.0	6.3	uA	All the clocks are disabled. No load for all the output pins. No floating for digital input pins. All the peripherals are disabled, with LDO set to low power consumption mode. Flash enters sleep mode and CPU enters STOP mode
Current for IDLE mode	I_{idl1}	1.81	2.08	2.10	mA	The system clock is XOSCH(24MHz), with all other clocks disabled. No load for all the output pins. No floating for digital input pins. All the peripherals are disabled, with LDO set to low power consumption mode. Flash enters sleep mode and CPU enters IDLE mode.
	I_{idl2}	0.396	0.444	0.448	mA	The system clock is IRCH(3.6864MHz), with all other clocks disabled. No load for all the output pins. No floating for digital input pins. All the peripherals are disabled, with LDO set to low power consumption mode. flash enters sleep mode and CPU enters IDLE mode
	I_{idl3}	1.73	1.97	1.98	mA	The system clock is PLL output with frequency multiplied by 6, and the reference clock is IRCH(3.6864MHz). All other clocks are disabled. No load for all the output pins. No floating for digital inputpins. All the peripherals are disabled. CPU enters IDLE mode

	Iidl4	17.6	18.4	18.9	uA	The system clock is IRCL(131KHZ), with other clocks disabled. No load for all the output pins. No floating for digital input pins. All the peripherals are disabled, with LDO set to low power consumption mode. CPU enters IDLE mode
	Iidl5	11.4	6.6	6.9	uA	The system clock is XOSCL(32.768KHz), with other clocks disabled. No load for all the output pins. No floating for digital input pins. All the peripherals are disabled, with LDO set to low power consumption mode. CPU enters IDLE mode
	Iidl6	13.8	16.3	18.9	uA	The system clock is XOSCL (32.768kHZ), other clocks are off, the LDO is set to low power mode, the output voltage is 1.61V, the LCD driver is turned on (without external LCD panel), and the LCD is set to minimum current drive, 1/3bias, 1/4duty , LCD clock is XOSCL, LCD_CAD is off (CAD_MOD=0), all LCD pins are on, all other output pins are unloaded, all digital input pins are not floating, CPU enters IDLE mode

High voltage for IO port input (Enable Schmitt mode)	Vhi1	VDD=1.8V	0.53	-	1.8	V	-
		VDD=3.3V	0.96		3.3		
		VDD=5V	1.42		5		
High voltage for IO port input (Disable Schmitt mode)	Vhi2	VDD=1.8V		0.5*VDD	VDD	V	-
		VDD=3.3V					
		VDD=5V					
Low voltage for IO port input (Enable Schmitt mode)	Vlo1	VDD=1.8V	0	-	0.49	V	-
		VDD=3.3V	0	-	0.87		
		VDD=5V	0	-	1.34		
High voltage for IO port input (Disable Schmitt mode)	Vlo2	VDD=1.8V	0	0.5*VDD		V	-
		VDD=3.3V					
		VDD=5V					
Push current for IO port	Ipu	VDD=3.3V	-	5.86	-	mA	IO is set to push-pull output mode, drive capacity is set to maximum, Vol=VDD-0.3V
		VDD=5V	-	8.45	-		
Sink current for IO port	Iol	VDD=3.3V	-	11.76	-	mA	IO is set to push-pull output mode, drive capacity is set to maximum, Vol=GND+0.3V
		VDD=5V	-	17.53	-		
Sink current for COM port	Isi	VDD=3.3V		65		mA	IO is set to push-pull output or LED COM pin function, drive capacity is set to maximum, Sink function is turned on, Vol=GND+0.3V
		VDD=5V		92			
Strong pull-down resistor for IO port	Rd1	VDD=1.8~5.5V		15		KΩ	-
Weak pull-down resistor for IO port	Rd2	VDD=1.8~5.5V	-	45	-	KΩ	-
Strong pull-up resistor for IO port	Ru1	VDD=1.8~5.5V	-	10	-	KΩ	-
Weak pull-up resistor for IO port	Ru2	VDD=1.8~5.5V		45		KΩ	

Note: The above parameters are the test results of typical chips selected at random and are for reference only.

29.3 AC Electrical Specification

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

Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
Time to start oscillation for IRCL	Trc1	-	50	-	us	IRCL frequency 131KHz
Time to start oscillation for IRCH	Trc2	-	10	-	us	IRCH frequency 3.6864MHz
Time to start oscillation for XOSCL	Tosc1	-	1	-	s	XOSCL frequency 32.768KHz
Time to start oscillation for XOSCH	Tosc2	-	2	-	ms	XOSCH frequency 24MHz
Time for PLL to be stable	Tpll	-	50	-	us	Reference clock is IRCH with frequency 3.6864MHz, PLL frequency multiplied by 6
Time of the reset pulse	Trst	-	0.5	-	us	

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

29.4 ADC Electrical Specification

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

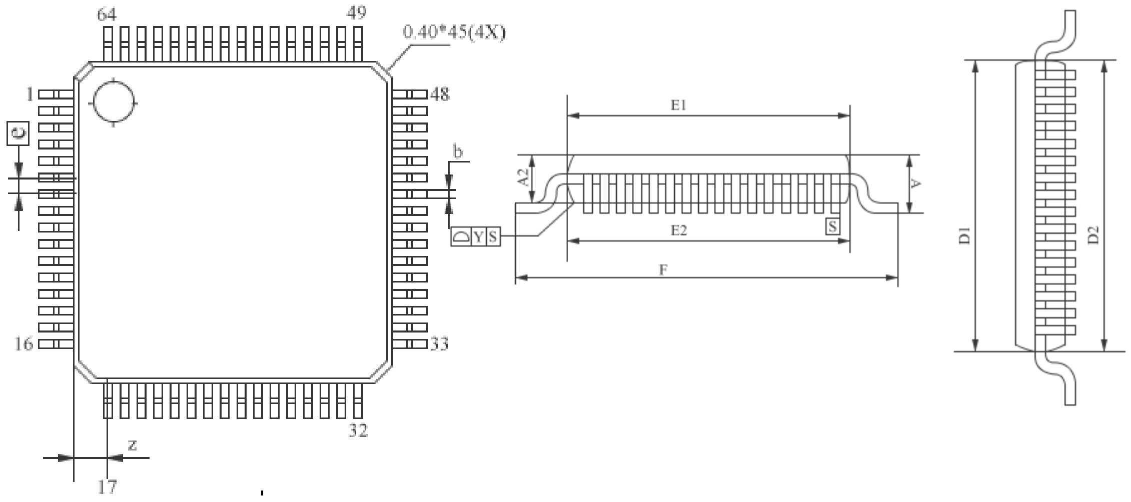
Parameter	Symbol	Minimum	Typical	Maximum	Unit	Condition
Working voltage	V_{AD}	1.8		5.5	V	
ADC precision	NR		12		Bit	GND<=Vin<=Vref
ADC input voltage	Vin	0	-	VDD	V	
ADC input resistance	Rin	2	-	-	MΩ	VDD=5V
ADC conversion current	IADC	-	180	-	uA	VDD=5V
Non-linear differential error	DNL	-	-	±3	LSB	VDD=5V
Non-linear integral error	INL	-	-	±3	LSB	VDD=5V
Full scale error	EF	-	±3	±4	LSB	VDD=5V
Offset error	Ez	-	±0.5	±1	LSB	VDD=5V
Conversion time	T_{CON}	-	16	-	Clock cycle	

Note：(1)ADC input resistance the its input resistance for DC mode.

(2)The internal resistance of the source must be less than 0KΩ for ADC test

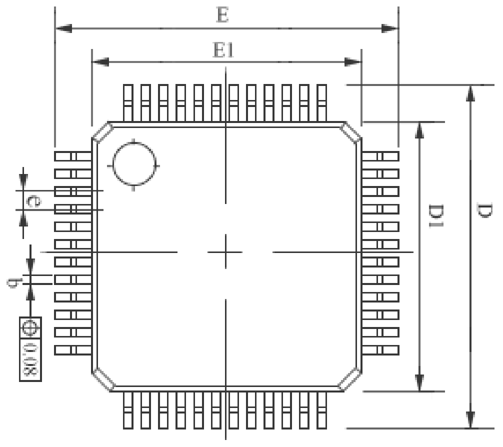
30 Package Type

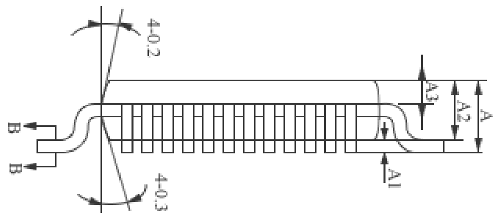
Package(1)(LQFP 64)



Sequence number	Minimum	Standard	Maximum
A	————	————	1.63
A2	1.30	1.40	1.50
D1	6.85	6.95	7.05
D2	6.90	7.00	7.10
E1	6.85	6.95	7.05
E2	6.90	7.00	7.10
e	————	0.40	————
F	8.80	9.0	9.20
Z	————	0.5	————

Package(2)(LQFP 48)

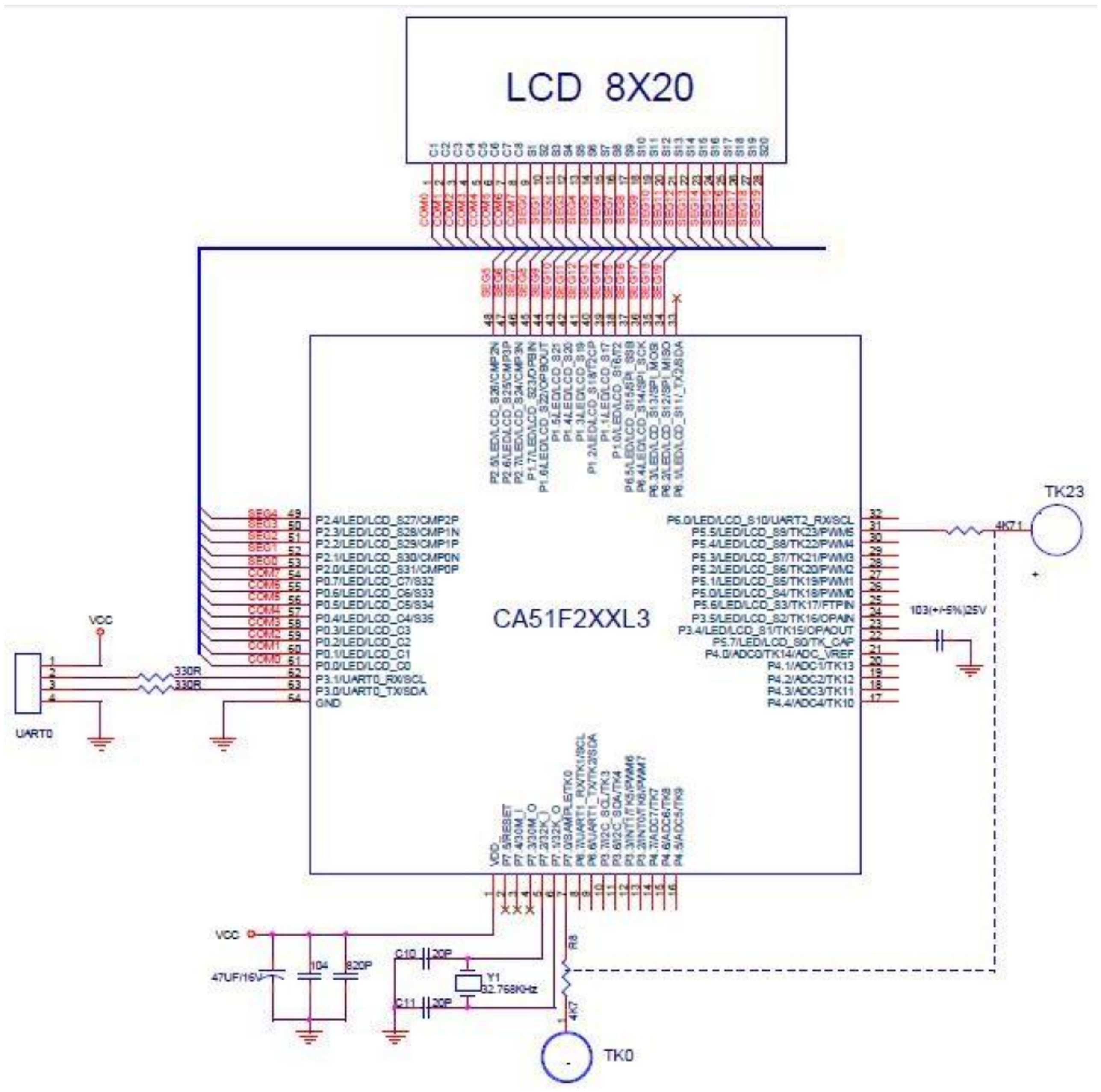




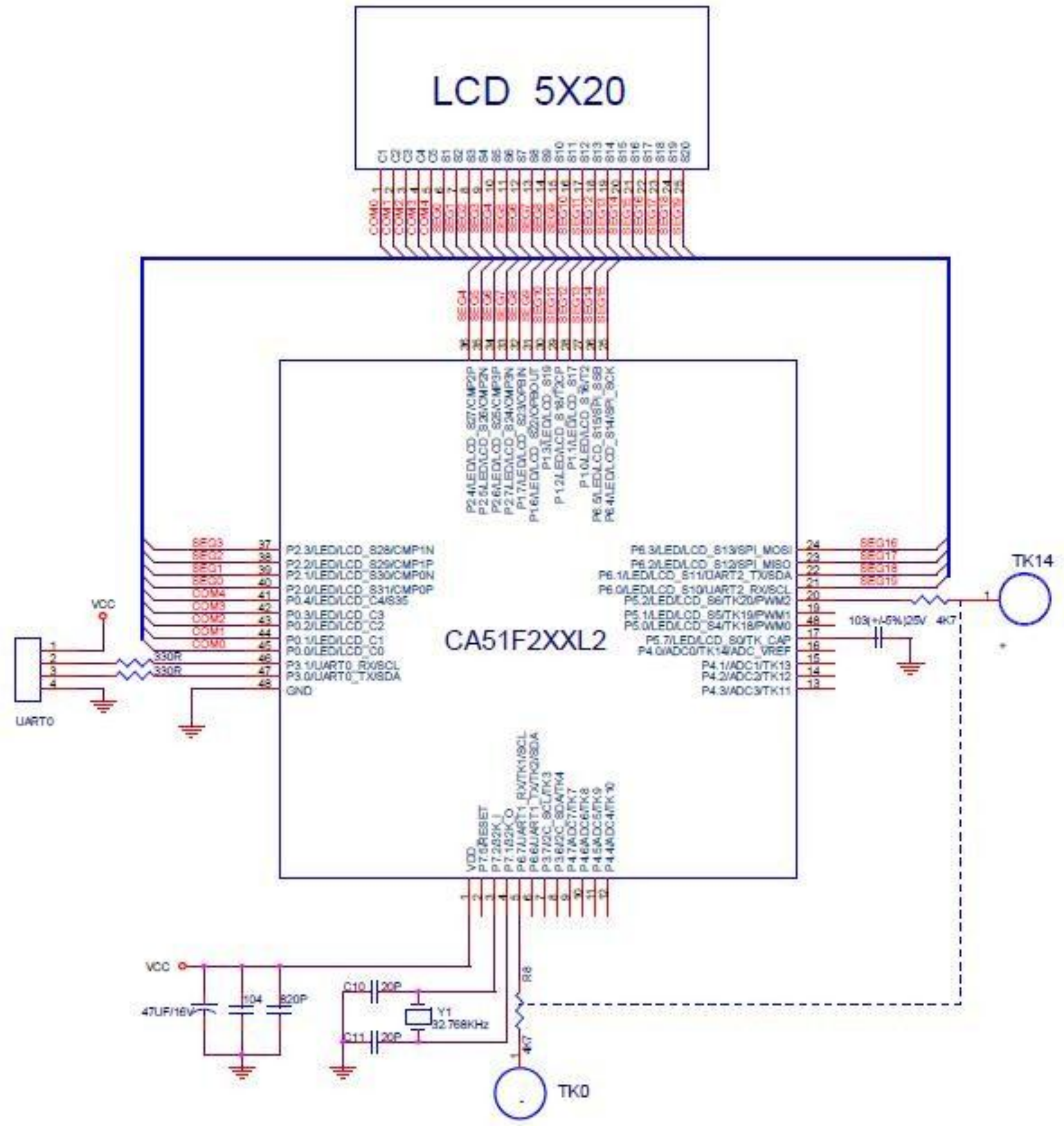
Sequence number	Minimum	Standard	Maximum
A	————	————	1.60
A1	0.05	————	0.15
A2	1.35	1.40	1.45
A3	0.59	0.54	0.69
b	0.18	————	0.27
D	8.80	9.00	9.20
D1	6.90	7.00	7.10
E	8.80	9.00	9.20
E1	6.90	7.00	7.10
e	0.50		

31 Typical Application Reference Circuit

Reference Circuit(1)



Reference Circuit(2)



32 Appendix

Appendix 1 Quick Reference List of Instruction Set

Mnemonic	Description	Description	Cycles
DATA TRANSFER			
MOV A,Rn	Move register to A	(A) ← (Rn)	1
MOV A,direct	Move direct to A	(A) ← (direct)	1
MOV A,@Ri	Move indirect RAM to A	(A) ← ((Ri))	1
MOV A,#data8	Move 8bit 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 8bit 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 indirect 8-byte 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-byte 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 PC 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	Accumulator half-byte swap	(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-byte immediate data to A	(A) ← (A) + #data	1
ADDC A, Rn	Add register to A with Carry	(A) ← (A) + (C) +	1

		(Rn)	
ADDC A, direct	Add direct byte to A with Carry	(A) ← (A) + (C) + (direct)	1
ADDC A, @Ri	Add indirect RAM to A with Carry	(A) ← (A) + (C) + ((Ri))	1
ADDC A, #data8	Add 8-byte 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(AxB=> BA)	temp16 ← (A) X (B) (A)←(temp.7,temp.6,...,temp.0) (B)←(temp.15,temp.14,...,temp.8)	4
DIV AB	Divide A by B (A/B=>A+B)	QUO ← (A) / (B) ......REM (A) ← QUO (B) ← REM	4

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

		(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-byte 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-byte 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-byte immediate data to A	(A) ← (A) | #data	1
ORL direct, A	OR A to direct byte	(direct) ← (direct) | (A)	1
ORL direct, #data8	OR 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-byte 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 immediate data to direct byte Clear A	(direct) ← (direct) ^ #data	2
CLR A	Clear A	(A) ← 0	1
CPL A	Complement A	(A) ← /(A)	1
RL A	Rotate A Left	(A) ← (A.6,A.5,...,A.0,A.7)	1
RLC A	Rotate A Left through Carry	C ← A.7 (A) ← (A.6,A.5,...,A.0,C)	1
RR A	Rotate A Right	(A) ← (A.0,A.7,...,A.2,A.1)	1
RRC A	Rotate A Right through Carry	C ← A.0 (A) ← (C,A.7,...,A.2,A.1)	1
PROGRAM AND MACHINE CONTROL			
ACALL addr11	Absolute subroutine call	(PC) ← (PC) + 2 (SP) ← (SP) + 1 ((SP)) ← (PC7-0) (SP) ← (SP) + 1 ((SP)) ← (PC15-8) (PC10-0) ← page	2

		address	
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-byte immediate to A & Jump if Not Equal	(PC) ← (PC) + 3
		IF (A) <> data
		THEN
		(PC) ← (PC) + relative offset
		IF (A) < data
		THEN
		(C) ← 1
			2

		ELSE (C) ← 0	

CJNE Rn, #data8, rel	Compare 8-byte 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
		THEN
		(PC) ← (PC) +
		relative offset
		IF ((Ri)) < data
		THEN
		(C) ← 1
		ELSE
		(C) ← 0
			2

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

		IF (bit) = 0 THEN (PC) ← (PC) + rel	
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 Mnemonic			
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		