Shenzhen Guoxin AI Co., Ltd.

STC8H family of Microcontrollers Reference Manual

Technical support website: http://www.STCMCUDATA.com

Date of update: **2022/3/9**

Contents

1 Overview

STC8H series of microcontrollers do not require an external crystal oscillator and external reset circuit. They are 8051 microcontrollers with the properties of strong anti-interference/ultra low price/high speed/low power consumption. Under the same operating frequency, STC8H series of microcontrollers are about 12 times faster (11.2 ~ 13.2 times) than traditional 8051. To execute all 111 instructions in sequence, the STC8H series microcontroller only needs 147 clocks, while the traditional 8051 requires 1944 clocks. STC8H series of microcontrollers are single clock/machine cycle (1T) microcontrollers produced by STC. They are new generation 8051 microcontrollers with wide voltage/high speed / high reliability / low power consumption / strong antistatic / strong anti-interference, and is super encrypted. The instruction codes are fully compatible with traditional 8051.

High precision of ±0.3% @+25℃ RC clock is integrated in MCU with −1.38% to +1.42% temperature drift under the temperature range of -40℃ to +85℃, and 0.88% to +1.05% temperature drift under temperature range from -20℃ to +65℃. The frequency of RC clock can be set from 4MHz to 35MHz when programming a MCU using ISP. Note: The maximum frequency must be controlled below 35MHz when the temperature range is -40℃ to + 85℃. Moreover, high reliable reset circuit is integrated in MCU with 4 levels optional reset threshold voltages, which can be selected when user programming using ISP. So, external expensive crystal and the external reset circuit can be eliminated completely.

There are three optional clock sources inside the MCU, internal high precision IRC which can be adjusted appropriately, internal 32KHz low speed IRC, external 4MHz~33MHz oscillator or external clock signal. The clock source can be freely chosen in user codes. After the clock source is selected, it may be 8-bit divided and then be supplied to the CPU and the peripherals, such as timers, UARTs, SPI, and so on.

Two low power modes are provided in MCU, the IDLE mode and the STOP mode. In IDLE mode, MCU stops clocking CPU, CPU stops executing instructions without clock, while all peripherals are still working. At this moment, the power consumption is about 1.0mA at 6MHz working frequency. The STOP mode is the power off or power-down mode. At this momont, the main clock stops, CPU and all peripherals stop working, and the power consumption can be reduced to about 0.6uA when VCC is 5.0V, 0.4uA when VCC is 3.3V.

The Power-down mode can be woke-up by one of the following interrupts: INT0(P3.2), INT1(P3.3), INT2(P3.6), INT3(P3.7), INT4(P3.0), T0(P3.4), T1(P3.5), T2(P1.2), T3(P0.4), T4(P0.6), RXD(P3.0/P3.6/P1.6/P4.3), RXD2(P1.0/P4.6), RXD3(P0.0/P5.0), RXD4(P0.2/P5.2), I2C_SDA(P1.4/P2.4/P3.3), Comparator, LVD, Power-down wake-up timer.

Rich digital peripherals and analog peripherals are provided in MCU, including UARTs, timers, enhanced PWMs and I2C, SPI, USB, ultra-high speed ADC and comparator, which can meet the requirements of users when designing a product.

The enhanced dual data pointers are integrated in the STC8H series of microcontrollers. Using user codes, the function of automatic increasing or decreasing of data pointer and automatic switching of two sets of data pointers can be realized.

Products Line	I/O	UART	Timers	ADC	Enhanced PWM	CMP	SPI	I2C	USB	MDU16	LED DRV	Touch Key	RTC	I/O Int.	Color LCM	LCD DRV	DMA
STC8H1K08 family	**17**	**2**	**3**	**9_{CH}*10_{B}**	●	●	●	●									
STC8H1K28 family	**29**	**2**	**5**	**12_{CH}*10_{B}**	●	●	●	●									
STC8H3K64S4 family	**45**	**4**	**5**	**12_{CH}*12_{B}**	●	●	●	●		●				●			
STC8H3K64S2 family	**45**	**2**	**5**	**12_{CH}*12_{B}**	●	●	●	●		●				●			
STC8H8K64U family Version A	**60**	**4**	**5**	**15_{CH}*12_{B}**	●	●	●	●	●	●							
STC8H8K64U familyVersion B	**60**	**4**	**5**	**15_{CH}*12_{B}**	●	●	●	●	●	●			●	●	●		●
STC8H2K64T family	**44**	**4**	**5**	**15_{CH}*12_{B}**	●	●	●	●		●	●	●	●	●			
STC8H4K64TLR family	**44**	**4**	**5**	**15_{CH}*12_{B}**	●	●	●	●		●	●	●	●	●	●		●
STC8H4K64TLCD family	**60**	**4**	**5**	**15_{CH}*12_{B}**	●	●	●	●		●		●	●	●	●	●	●
STC8H4K64LCD family	**61**	**4**	**5**	**15_{CH}*12_{B}**	●	●	●	●		●			●	●	●	●	●
STC8H1K08TR family	**16**	**2**	**3**	**15_{CH}*12_{B}**	●	●	●	●		●		●	●	●	●		●

2 Features, Price and Pins

2.1 STC8H1K08-36I-TSSOP20/QFN20 family

2.1.1Features and Price

- **Selection and price (No external crystal and external reset required with 9 channels 10-bit ADC)**

MCU	Operating voltage（V）	Flash Code Memory (100 thousand times) (Byte)	Idata, Internal DATA RAM(Byte)	Xdata, Internal extended SRAM (Byte)	Enhanced Dual DPTR(increasing or decreasing)	EEPROM 100 thousand times) (Byte)	Maximum I/O Lines	Traditional I/O interrupt(INT0/INT1/INT2/INT3/INT4) (can wake-up CPU)	UARTs which can wake-up CPU	SPI	I^{2}C which can wake-up CPU	Timers/Counters (T0-T2 Pin can wake-up CPU)	16-bit advanced PWM timers with Complementary symmetrical dead-time	Power-down Wake-up timer	9-channels high speed ADC (8 PWMs can be used as 8 DACs)	Comparator (May be used as ADC to detect external power-down)	Internal LVD interrupt (can wake-up CPU)	Watch-dog Timer	Internal high reliable reset circuit with 4 level onitional reset threshold	Internal high presision Clock (adjustabal under 35MHz)	Clock output and Reset	Program encrypted transmission (Anti-blocking)	Password can be set for next update	Support RS485 download	Support software USB download directly	Online debugging	Price & Package TSSOP20 <6.5mm*6.5mm>	 QFN20 <3mm*3mm>	Main product supply information
STC8H1K08	1.9-5.5	8K	256B	1K	2	4K	17	Y	2	Y	Y	3	8	Y	10bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	¥1.9	√	Available
STC8H1K17	1.9-5.5	17K	256B	1K	2	IAP	17	Y	2	Y	Y	3	8	Y	10bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	-	-	√	√	
~~STC8H1K12~~	~~1.9-5.5~~	~~12K~~	~~256B~~	~~1K~~	~~2~~	~~IAP~~	~~17~~	~~Y~~	~~2~~	~~Y~~	~~Y~~	~~3~~	~~8~~	~~Y~~	~~10bit~~	~~Y~~	~~Y~~	~~Y~~	~~Y~~	~~Y~~	~~Y~~	~~Y~~	~~Y~~	~~Y~~	~~Y~~	~~Y~~	Not available		

- **Core**
  - ✓ Ultra-high speed 8051 Core with single clock per machine cycle, which is called 1T, and the speed is about 12 times faster than traditional 8051
  - ✓ Fully compatible instruction set with traditional 8051
  - ✓ 17interrupt sources and 4 interrupt priority levels
  - ✓ Online debugging is supported

- **Operating voltage**
  - ✓ 1.9V～5.5V

- **Operating temperature**
  - ✓ -40℃~85℃(The chip is produced in -40℃～125℃ process. Please refer to the description of the electrical characteristics chapter for applications beyond the temperature range)

- **Flash memory**
  - ✓ Up to 17Kbytes of Flash memory to be used for storing user code
  - ✓ Configurable size EEPROM, 512bytes single page for being erased, which can be repeatedly erased more than 100 thousand times.
  - ✓ In-System-Programming, ISP in short, can be used to update the application code. No dedicated programmer is needed.
  - ✓ Online debugging with single chip is supported, and no dedicated emulator is needed. The number of breakpoints is unlimited theoratically.

- **SRAM**
  - ✓ 128 bytes internal direct access RAM (DATA, use keyword *data* to declare in C language program)

- ✓ 128 bytes internal indirect access RAM (IDATA, use keyword *idata* to declare in C language program)
- ✓ 1024 bytes internal extended RAM (internal XDATA, use keyword *xdata* to declare in C language program)

➢ **Clock**

- ✓ Internal high precise RC clock(IRC for short, ranges from 4MHz to 36MHz), adjustable while ISP and can be divided to lower frequency by user software, 100KHz for instance.
  - ✓ Error:±0.3% (at the temperature 25℃)
  - ✓ -1.35%～+1.30% temperature drift (at the temperature range of -40℃ to +85℃)
  - ✓ -0.76%～+0.98% temperature drift (at the temperature range of -20℃ to 65℃)
- ✓ Internal 32KHz low speed IRC with large error
- ✓ External 4MHz~33MHz oscillator or external clock

The three clock sources above can be selected freely by user code.

➢ **Reset**

- ✓ Hardware reset
  - ✓ Power-on reset. (**Effective when the chip does not enable the low voltage reset function**)
  - ✓ Reset by reset pin. The default function of P5.4 is the I/O port. P5.4 pin can be set as the reset pin while ISP download. (**Note: When the P5.4 pin is set as the reset pin, the reset level is low**.)
  - ✓ Watch dog timer reset
  - ✓ Low voltage detection reset. 4 low voltage detection levels are provided, 2.0V, 2.4V, 2.7V, 3.0V.
- ✓ Software reset
  - ✓ Writing the reset trigger register using software

➢ **Interrupts**

- ✓ 17 interrupt sources: INT0(Supports rising edge and falling edge interrupt), INT1(Supports rising edge and falling edge interrupt), INT2(Supports falling edge interrupt only), INT3(Supports falling edge interrupt only), INT4(Supports falling edge interrupt only), timer 0, timer 1, timer 2, UART 1, UART 2, ADC, LVD, SPI, I^{2}C, comparator, PWMA, PWMB
- ✓ 4 interrupt priority levels
- ✓ Interrupts that can wake up the CPU in clock stop mode: INT0(P3.2), INT1(P3.3), INT2(P3.6), INT3(P3.7), INT4(P3.0), T0(P3.4), T1(P3.5), T2(P1.2), RXD(P3.0/P3.6/P1.6), RXD2(P1.0), I2C_SDA(P1.4/P3.3), Comparator interrupt, LVD interrupt, Power-down wake-up timer.

➢ **Digital peripherals**

- ✓ 3 16-bit timers: timer0, timer1, timer2, where the mode 3 of timer 0 has the Non-Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
- ✓ 2 high speed UARTs: UART1, UART2, whose maximum baudrate clock may be FOSC/4
- ✓ 8 channels/2 groups of enhanced PWMs, which can realize control signals with dead time, and support external fault detection function. In addition, it also supports 16-bit timers, 8 external interrupts, 8 channels of external capture and pulse width measurement functions.
- ✓ SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
- ✓ I^{2}C: Master mode or slave mode are supported.

➢ **Analog peripherals**

- ✓ 9 channels (channel 0 to channel 1, channel 8 to channel 14) ultra high speed ADC which supports 10-bit precision. The maximum speed can be 500K(Half a million ADC conversions per second)
- ✓ Channel 15 of ADC is used to test the internal reference voltage. (The default internal reference voltage is 1.19V when the chip is shipped)
- ✓ A set of comparator (the CMP+ port and all ADC input ports can be selected as the positive terminal of the comparator, so the comparator can be used as a multi-channel comparator for time division multiplexing)
- ✓ DAC: 8 channels advanced PWMs timers can be used as 8 channels DAC

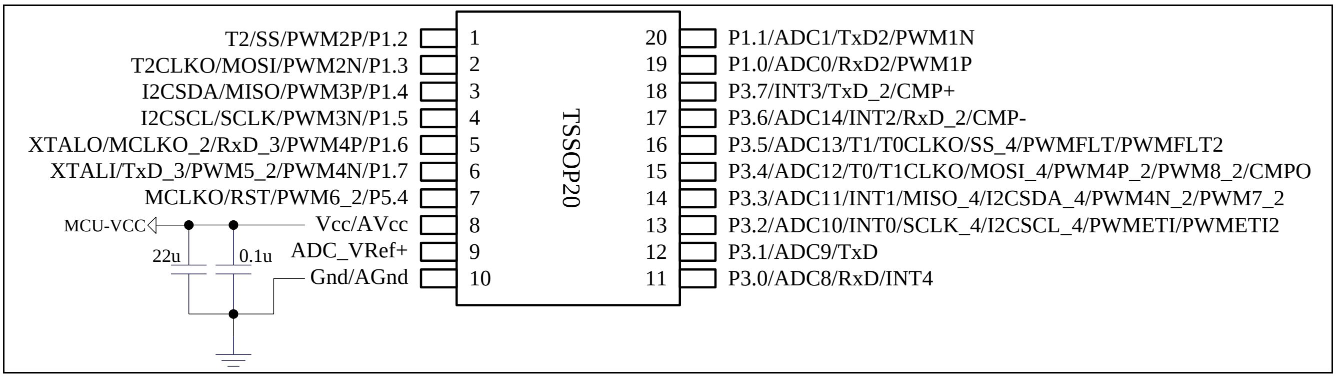
➢ **GPIO**

- ✓ Up to 17 GPIOs: P1.0~P1.7, P3.0~P3.7, P5.4
- ✓ 4 modes for all GPIOs: quasi_bidirectional mode, push-pull outputmode, open drain mode, high-impedance input mode
- ✓ Except for P3.0 and P3.1, all other I/O ports are in a high-impedance state after power-on. User must configure the I/O ports mode before using them. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.

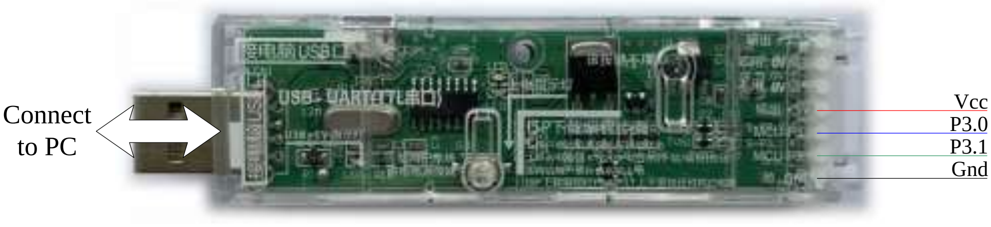
➢ **Package**

- ✓ TSSOP20 <6.5mm*6.5mm>, QFN20 <3mm*3mm>

2.1.2 Pinouts



Note:
1. ADC's external reference power supply pin ADC_VRef+ must not be floating. It must be connected to an external reference power supply or directly connected to Vcc.
2. If USB download is not required, P3.0/P3.1/P3.2 cannot be at low level at the same time when the chip is reset.



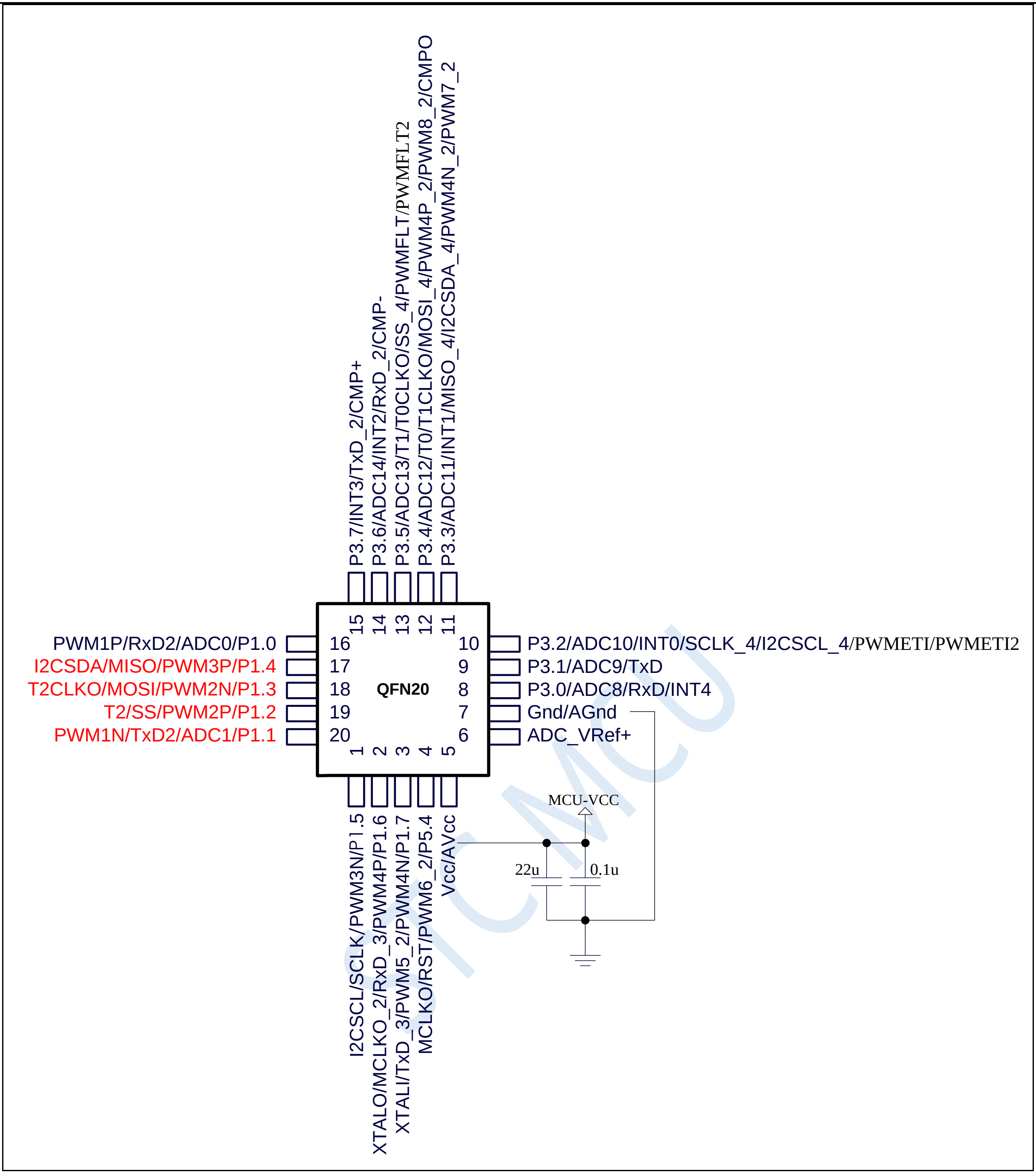
universal USB to UART tool

ISP download steps:

1. **Connect the universal USB to UART tool to the target chip according to the connection method shown in the figure above.**
2. **Press the power button to confirm that the target chip is in a power-off state (the power-on LED is off).** **Note: When the tool is powered on for the first time, there is no external power supply, so if it is the first time to use this tool, you can skip this step.**
3. **Click the "Download/Program" button in the STC-ISP download software.**
4. **Press the power button again to power on the target chip (the power-on LED is on).**
5. **Start ISP download.** **Note: It has been found that when using the USB cable for ISP download, if the USB cable is too thin and the voltage drop on the USB cable is too large, this will result in insufficient power supply during the ISP download. Therefore, please be sure to use the booster USB cable for ISP download.**

Note:

1. **Except for P3.0 and P3.1, all other I/O ports are in high-impedance input state after power-on. User must set the I/O port mode firstly when using I/O.**
2. **All I/O ports can be set to quasi-bidirectional port mode, push-pull output mode, open-drain output mode or high-impedance input mode. In addition, each I/O can enable the internal 4K pull-up resistor independently.**
3. **When P5.4 is enabled as the reset pin, the reset level is low.**


P3.7/INT3/TxD_2/CMP+
P3.6/ADC14/INT2/RxD_2/CMP-
P3.5/ADC13/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2
P3.4/ADC12/T0/T1CLKO/MOSI_4/PWM4P_2/PWM8_2/CMPO
P3.3/ADC11/INT1/MISO_4/I2CSDA_4/PWM4N_2/PWM7_2
15
14
13
12
11
PWM1P/RxD2/ADC0/P1.0
I2CSDA/MISO/PWM3P/P1.4
T2CLKO/MOSI/PWM2N/P1.3
T2/SS/PWM2P/P1.2
PWM1N/TxD2/ADC1/P1.1
16
17
18
19
20
QFN20
10
9
8
7
6
P3.2/ADC10/INT0/SCLK_4/I2CSCL_4/PWMETI/PWMETI2
P3.1/ADC9/TxD
P3.0/ADC8/RxD/INT4
Gnd/AGnd
ADC_VRef+
1
2
3
4
5
I2CSCL/SCLK/PWM3N/P1.5
XTALO/MCLKO_2/RxD_3/PWM4P/P1.6
XTALI/TxD_3/PWM5_2/PWM4N/P1.7
MCLKO/RST/PWM6_2/P5.4
Vcc/AVcc
MCU-VCC
22u
0.1u

2.1.3 Pin descriptions

Pin number		name	type	description
TSSOP20	QFN20			
1	19	P1.2	I/O	Standard IO port
		SS	I/O	Slave selection of SPI
		T2	I	Timer2 external input
		PWM2P	I/O	Capture of external signal/Positive of PWMB pulse output
2	18	P1.3	I/O	Standard IO port
		MOSI	I/O	Master Output/Slave Input of SPI
		T2CLKO	O	Clock out of timer 2
		PWM2N	I/O	Capture of external signal/Negative of PWMB pulse output
3	17	P1.4	I/O	Standard IO port
		MISO	I/O	Master Iutput/Slave Onput of SPI
		SDA	I/O	Serial data line of I2C
		PWM3P	I/O	Capture of external signal/ Positive of PWM3 pulse output
4	1	P1.5	I/O	Standard IO port
		SCLK	I/O	Serial Clock of SPI
		SCL	I/O	Serial Clock line of I2C
		PWM3N	I/O	Capture of external signal/ Negative of PWM3 pulse output
5	2	P1.6	I/O	Standard IO port
		RxD_3	I	Serial input of UART1
		PWM4P	I/O	Capture of external signal/ Positive of PWM4 pulse output
		MCLKO_2	O	Master clock output
		XTALO	O	Connect to external oscillator
6	3	P1.7	I/O	Standard IO port
		TxD_3	O	Serial Transmit pin of UART 1
		PWM4N	I/O	Capture of external signal/ Negative of PWM4 pulse output
		PWM5_2	I/O	Capture of external signal/ Positive of PWM5 pulse output
		XTALI	I	Connect to external oscillator
7	4	P5.4	I/O	Standard IO port
		NRST	I	Reset pin
		MCLKO	O	Main clock output
		PWM6_2	I/O	Capture of external signal/ Positive of PWM6 pulse output
8	5	Vcc	Vcc	Power Supply
		AVcc	Vcc	Power Supply for ADC
9	6	VREF+	I	Reference voltage pin of ADC
10	7	Gnd	Gnd	Ground
		AGnd	Gnd	ADC Ground
11	8	P3.0	I/O	Standard IO port
		ADC8	I	ADC analog input 8
		RxD	I	Serial input of UART1
		INT4	I	External interrupt 4
12	9	P3.1	I/O	Standard IO port
		ADC9	I	ADC analog input 9
		TxD	O	Serial Transmit pin of UART 1

Pin number		name	type	description
TSSOP20	QFN20			
13	10	P3.2	I/O	Standard IO port
		ADC10	I	ADC analog input 10
		INT0	I	External interrupt 0
		SCLK_4	I/O	Serial Clock of SPI
		SCL_4	I/O	Serial Clock line of I2C
		PWMETI	I	PWM External trigger input pin
		PWMETI2	I	PWM External trigger input pin 2
14	11	P3.3	I/O	Standard IO port
		ADC11	I	ADC analog input 11
		INT1	I	External interrupt 1
		MISO_4	I/O	Master Iutput/Slave Onput of SPI
		SDA_4	I/O	Serial data line of I2C
		PWM4N_4	I/O	Capture of external signal/ Negative of PWM4 pulse output
		PWM7_2	I/O	Capture of external signal/ Positive of PWM7 pulse output
15	12	P3.4	I/O	Standard IO port
		ADC12	I	ADC analog input 12
		T0	I	Timer0 external input
		T1CLKO	O	Clock out of timer 1
		MOSI_4	I/O	Master Output/Slave Input of SPI
		PWM4P_4	I/O	Capture of external signal/ Positive of PWM4 pulse output
		PWM8_2	I/O	Capture of external signal/ Positive of PWM8 pulse output
		CMPO	O	Comparator output
16	13	P3.5	I/O	Standard IO port
		ADC13	I	ADC analog input 13
		T1	I	Timer1 external input
		T0CLKO	O	Clock out of timer 0
		SS_4	I/O	Slave selection of SPI
		PWMFLT	I	PWMA external anomaly detection pin
		PWMFLT2	I	PWMB external anomaly detection pin
17	14	P3.6	I/O	Standard IO port
		ADC14	I	ADC analog input 14
		INT2	I	External interrupt 2
		RxD_2	I	Serial input of UART1
		CMP-	I	Comparator negative input
18	15	P3.7	I/O	Standard IO port
		INT3	I	External interrupt 3
		TxD_2	O	Serial Transmit pin of UART 1
		CMP+	I	Comparator positive input
19	16	P1.0	I/O	Standard IO port
		ADC0	I	ADC analog input 0
		RxD2	I	Serial input of UART2
		PWM1P	I/O	Capture of external signal/ Positive of PWMA pulse output
20	20	P1.1	I/O	Standard IO port
		ADC1	I	ADC analog input 1
		TxD2	O	Serial Transmit pin of UART 2
		PWM1N	I/O	Capture of external signal/ Negative of PWMA pulse output

2.2 STC8H1K28-36I-LQFP32/QFN32 family

2.2.1 Features and Price

- **Selection and price (No external crystal and external reset required)**

MCU	Operating voltage (V)	Flash Code Memory (100 thousand times) (Byte)	Idata, Internal DATA RAM(Byte)	Xdata, Internal extended SRAM (Byte)	Enhanced Dual DPTR increasing or decreasing	EEPROM 100 thousand times) (Byte)	Maximum I/O Lines	Traditional I/O interrupt(INT0/INT1/INT2/INT3/INT4) (can wake-up CPU)	UARTs which can wake-up CPU	SPI	I^{2}C which can wake-up CPU	Timers/Counters (T0-T4 Pin Can wake-up CPU)	16-bit advanced PWM timer with Complementary symmetrical dead-time	Power-down Wake-up timer	12-channels high speed ADC (8 PWM can be used as 8 DACs)	Comparator (May be used as ADC to detect external power-down)	Internal LVD interrupt (can wake-up CPU)	Watch-dog Timer	Internal high reliable reset circuit with 4 level optional reset threshold	Internal high presision Clock (adjustbal under 35MHz)	Clock output and Reset	Program encrypted transmission (Anti-blocking)	Password can be set for next update	Support RS485 download	Support software USB download directly	Online debugging	Price & Package		Main product supply information
																											LQFP32<9mm*9mm>	QFN32<4mm*4mm>	
STC8H1K16	1.9-5.5	16K	256	1K	2	12K	29	Y	2	Y	Y	5	8	Y	10bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	¥2.4	¥2.4	Available
STC8H1K24	1.9-5.5	24K	256	1K	2	4K	29	Y	2	Y	Y	5	8	Y	10bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	¥2.5	¥2.5	
STC8H1K28	1.9-5.5	28K	256	1K	2	IAP	29	Y	2	Y	Y	5	8	Y	10bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	¥2.6	¥2.6	
STC8H1K33	1.9-5.5	33K	256	1K	2	IAP	29	Y	2	Y	Y	5	8	Y	10bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	-	-	√	√	

- **Core**
  - ✓ Ultra-high speed 8051 Core with single clock per machine cycle, which is called 1T and the speed is about 12 times faster than traditional 8051
  - ✓ Fully compatible instruction set with traditional 8051
  - ✓ 19 interrupt sources and 4 interrupt priority levels
  - ✓ Online debugging is supported
- **Operating voltage**
  - ✓ 1.9V～5.5V
- **Operating temperature**
  - ✓ -40℃~85℃(The chip is -40℃～125℃ process. Please refer to the description of the electrical characteristics chapter for applications beyond the temperature range)
- **Flash memory**
  - ✓ Up to 33Kbytes of Flash memory to be used for storing user code
  - ✓ Configurable size EEPROM, 512bytes single page for being erased, which can be repeatedly erased more than 100 thousand times.
  - ✓ In-System-Programming, ISP in short, can be used to update the application code. No dedicated programmer is needed.
  - ✓ Online debugging with single chip is supported, and no dedicated emulator is needed. The number of breakpoints is unlimited theoratically.
- **SRAM**

  - ✓ 128 bytes internal direct access RAM (DATA, use keyword *data* to declare in C language program)
  - ✓ 128 bytes internal indirect access RAM(IDATA, use keyword *idata* to declare in C language program)
  - ✓ 1024 bytes internal extended RAM (internal XDATA, use keyword *xdata* to declare in C language program)

- ➢ **Clock**
  - ✓ Internal high precise RC clock (IRC for short, ranges from 4MHz to 38MHz), adjustable while ISP and can be divided to lower frequency by user software, 100KHz for instance.
    - ✓ Error:±0.3% (at the temperature 25℃)
    - ✓ -1.35%～+1.30% temperature drift (at the temperature range of -40 ℃ to +85 ℃)
    - ✓ -0.76%～+0.98% temperature drift (at the temperature range of -20℃ to 65℃)
  - ✓ Internal 32KHz low speed IRC with large error
  - ✓ External 4MHz~33MHz oscillator or external clock

    The three clock sources above can be selected freely by user code.

- ➢ **Reset**
  - ✓ Hardware reset
    - ✓ Power-on reset. (**Effective when the chip does not enable the low voltage reset function**)
    - ✓ Reset by reset pin. The default function of P5.4 is the I/O port. P5.4 pin can be set as the reset pin while ISP download. (**Note: When the P5.4 pin is set as the reset pin, the reset level is low**.)
    - ✓ Watch dog timer reset
    - ✓ Low voltage detection reset. 4 low voltage detection levels are provided, 2.0V, 2.4V, 2.7V, 3.0V.
  - ✓ Software reset
    - ✓ Writing the reset trigger register using software

- ➢ **Interrupts**
  - ✓ 19 interrupt sources: INT0(Supports rising edge and falling edge interrupt), INT1(Supports rising edge and falling edge interrupt), INT2(Supports falling edge interrupt only), INT3(Supports falling edge interrupt only), INT4(Supports falling edge interrupt only), timer0, timer1, timer2, timer3, timer4, UART1, UART2, ADC, LVD, SPI, I^{2}C, Comparator, PWMA,PWMB
  - ✓ 4 interrupt priority levels
  - ✓ Interrupts that can wake up the CPU in clock stop mode: INT0(P3.2), INT1(P3.3), INT2(P3.6), INT3(P3.7), INT4(P3.0), T0(P3.4), T1(P3.5), T2(P1.2), T3(P0.4), T4(P0.6), RXD(P3.0/P3.6/P1.6), RXD2(P1.0), I2C_SDA(P1.4/P2.4/P3.3), Comparator, LVD, Power-down wake-up timer.

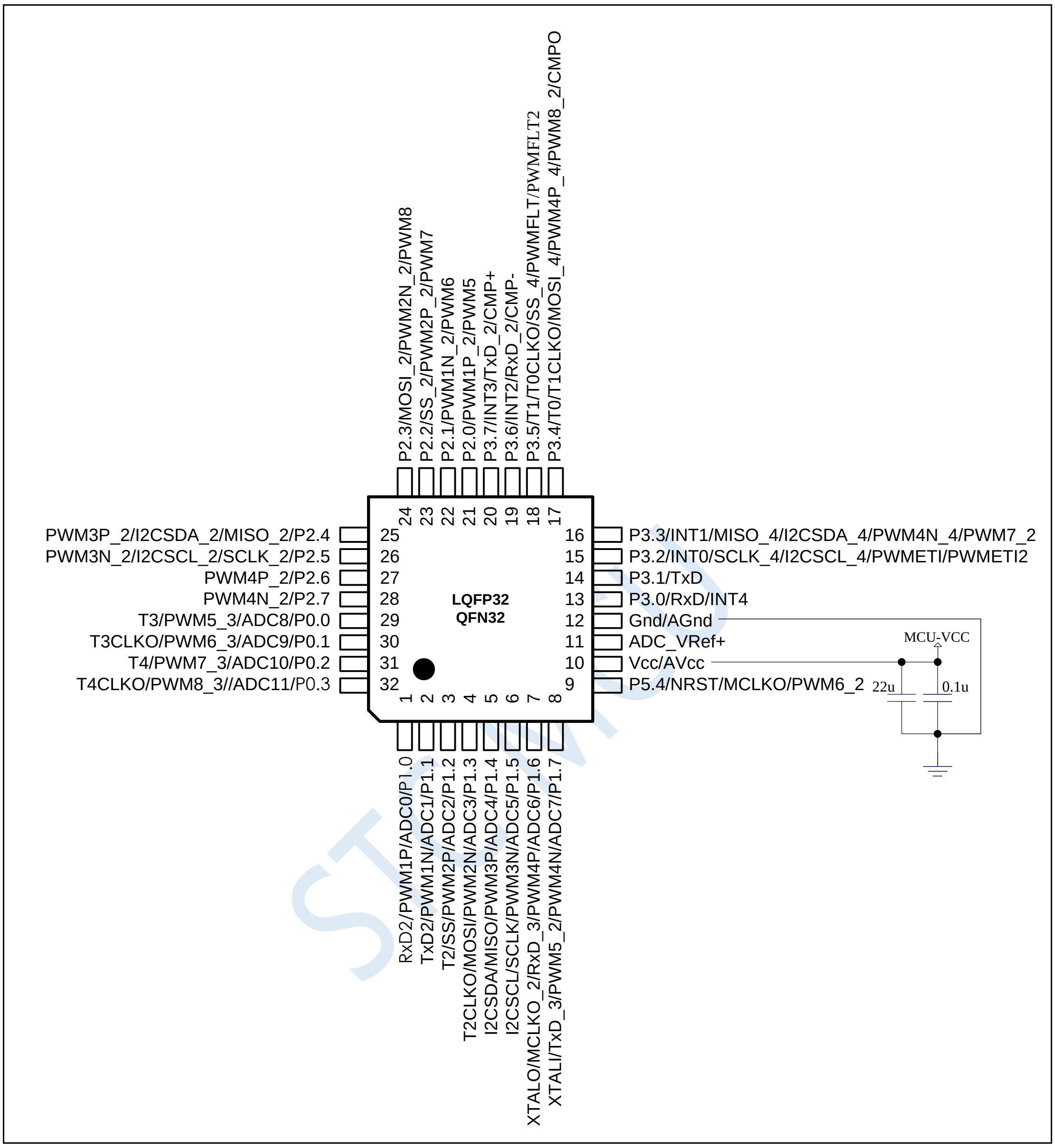
- ➢ **Digital peripherals**
  - ✓ 5 16-bit timers: timer0, timer1, timer2, timer3, timer4, where the mode 3 of timer 0 has the Non-Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
  - ✓ 2 high speed UARTs: UART1, UART2, whose maximum baudrate may be FOSC/4.
  - ✓ 8 channels/2 groups of enhanced PWMs, which can realize control signals with dead time, and support external fault detection function. In addition, it also supports 16-bit timers, 8 external interrupts, 8 external captures and pulse width measurement functions.
  - ✓ SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
  - ✓ I^{2}C: Master mode or slave mode are supported.

- ➢ **Analog peripherals**
  - ✓ 12 channels (channel 0 to channel 11) ultra high speed ADC which supports 10-bit precision. The maximum speed can be 500K(Half a million ADC conversions per second)
  - ✓ Channel 15 of ADC is used to test the internal reference voltage. (The default internal reference voltage is 1.19V when the chip is shipped)
  - ✓ A set of comparator (the CMP+ port and all ADC input ports can be selected as the positive terminal of the comparator, so the comparator can be used as a multi-channel comparator for time division multiplexing)
  - ✓ DAC: 8 channels advanced PWM timer can be used as 8 channels DAC

- ➢ **GPIO**
  - ✓ Up to 29 GPIOs: P0.0~P0.3, P1.0~P1.7, P2.0~P2.7, P3.0~P3.7, P5.4
  - ✓ 4 modes for all GPIOs: quasi_bidirectional mode, push-pull outputmode, open drain mode, high-impedance input mode
  - ✓ Except for P3.0 and P3.1, all other I/O ports are in a high-impedance state after power-on. User must configure the I/O ports mode before using them. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.

- ➢ **Package**
  - ✓ LQFP32 <9mm*9mm>, QFN32 <4mm*4mm>

2.2.2 Pinouts


P2.3/MOSI_2/PWM2N_2/PWM8
P2.2/SS_2/PWM2P_2/PWM7
P2.1/PWM1N_2/PWM6
P2.0/PWM1P_2/PWM5
P3.7/INT3/TxD_2/CMP+
P3.6/INT2/RxD_2/CMP-
P3.5/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2
P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO
24 23 22 21 20 19 18 17
PWM3P_2/I2CSDA_2/MISO_2/P2.4 25
PWM3N_2/I2CSCL_2/SCLK_2/P2.5 26
PWM4P_2/P2.6 27
PWM4N_2/P2.7 28
T3/PWM5_3/ADC8/P0.0 29
T3CLKO/PWM6_3/ADC9/P0.1 30
T4/PWM7_3/ADC10/P0.2 31
T4CLKO/PWM8_3//ADC11/P0.3 32
LQFP32
QFN32
16 P3.3/INT1/MISO_4/I2CSDA_4/PWM4N_4/PWM7_2
15 P3.2/INT0/SCLK_4/I2CSCL_4/PWMETI/PWMETI2
14 P3.1/TxD
13 P3.0/RxD/INT4
12 Gnd/AGnd
11 ADC_VRef+
10 Vcc/AVcc
9 P5.4/NRST/MCLKO/PWM6_2
MCU-VCC
22u
0.1u
1 2 3 4 5 6 7 8
RxD2/PWM1P/ADC0/P1.0
TxD2/PWM1N/ADC1/P1.1
T2/SS/PWM2P/ADC2/P1.2
T2CLKO/MOSI/PWM2N/ADC3/P1.3
I2CSDA/MISO/PWM3P/ADC4/P1.4
I2CSCL/SCLK/PWM3N/ADC5/P1.5
XTALO/MCLKO_2/RxD_3/PWM4P/ADC6/P1.6
XTALI/TxD_3/PWM5_2/PWM4N/ADC7/P1.7

Note:
1. ADC's external reference power supply pin ADC_VRef+ must not be floating. It must be connected to an external reference power supply or directly connected to Vcc.
2. If USB download is not required, P3.0/P3.1/P3.2 cannot be at low level at the same time when the chip is reset.

The download steps using ISP and notes are the same as the circumstances in 2.1.2.

Note:

1. **Except for P3.0 and P3.1, all other I/O ports are in high-impedance input state after power-on. User must set the I/O port mode firstly when using I/O.**
2. **All I/O ports can be set to quasi-bidirectional port mode, strong push-pull output mode, open-drain output mode or high-impedance input mode. In addition, each I/O can independently enable the internal 4K pull-up resistor.**
3. **When P5.4 is enabled as the reset pin, the reset level is low.**

2.2.3 Pin descriptions

Pin number		name	type	description
LQFP32/QFN32				
1		P1.0	I/O	Standard IO port
		RxD2	I	Serial input of UART2
		ADC0	I	ADC analog input 0
		PWM1P	I/O	Capture of external signal/Positive of PWMA pulse output
2		P1.1	I/O	Standard IO port
		TxD2	O	Serial Transmit pin of UART 2
		ADC1	I	ADC analog input 1
		PWM1N	I/O	Capture of external signal/Negative of PWMA pulse output
3		P1.2	I/O	Standard IO port
		ADC2	I	ADC analog input 2
		SS	I/O	Slave selection of SPI
		T2	I	Timer2 external input
		PWM2P	I/O	Capture of external signal/ Positive of PWMB pulse output
4		P1.3	I/O	Standard IO port
		ADC3	I	ADC analog input 3
		MOSI	I/O	Master Output/Slave Input of SPI
		T2CLKO	O	Clock out of timer 2
		PWM2N	I/O	Capture of external signal/ Negative of PWMB pulse output
5		P1.4	I/O	Standard IO port
		ADC4	I	ADC analog input 4
		MISO	I/O	Serial Clock line of I2C
		SDA	I/O	Serial data line of I2C
		PWM3P	I/O	Capture of external signal/ Positive of PWM3 pulse output
6		P1.5	I/O	Standard IO port
		ADC5	I	ADC analog input 5
		SCLK	I/O	Serial Clock of SPI
		SCL	I/O	Serial Clock line of I2C
		PWM3N	I/O	Capture of external signal/ Negative of PWM3 pulse output
7		P1.6	I/O	Standard IO port
		ADC6	I	ADC analog input 6
		RxD_3	I	Serial input of UART1
		PWM4P	I/O	Capture of external signal/ Positive of PWM4 pulse output
		MCLKO_2	O	Main clock output
		XTALO	O	Output pin of external crystal oscillator
8		P1.7	I/O	Standard IO port
		ADC7	I	ADC analog input 7
		TxD_3	O	Serial Transmit pin of UART 1
		PWM4N	I/O	Capture of external signal/Negative of PWM4 pulse output
		PWM5_2	I/O	Capture of external signal/ Positive of PWM5 pulse output
		XTALI	I	External crystal/external clock input pin

Pin number		name	type	description
LQFP32/QFN32				
9		P5.4	I/O	Standard IO port
		RST	I	Reset pin
		MCLKO	O	Main clock output
		PWM6_2	I/O	Capture of external signal/ Positive of PWM6 pulse output
10		Vcc	Vcc	Power Supply
		AVcc	Vcc	Power Supply for ADC
11		ADC_VREF+	I	Reference voltage pin of ADC.It can be directly connected to the VCC of the MCU if the requirement is not high
12		Gnd	Gnd	Ground
		AGnd	Gnd	ADC Ground
13		P3.0	I/O	Standard IO port
		RxD	I	Serial input of UART1
		INT4	I	External interrupt 4
14		P3.1	I/O	Standard IO port
		TxD	O	Serial Transmit pin of UART 1
15		P3.2	I/O	Standard IO port
		INT0	I	External interrupt 0
		SCLK_4	I/O	Serial Clock of SPI
		SCL_4	I/O	Serial Clock line of I2C
		PWMETI	I	PWM external trigger input pin
		PWMETI2	I	PWM external trigger input pin 2
16		P3.3	I/O	Standard IO port
		INT1	I	External interrupt 1
		MISO_4	I/O	Serial Clock line of I2C
		SDA_4	I/O	Serial data line of I2C
		PWM4N_4	I/O	Capture of external signal/ Negative of PWM4 pulse output
		PWM7_2	I/O	Capture of external signal/ Positive of PWM7 pulse output
17		P3.4	I/O	Standard IO port
		T0	I	Timer0 external input
		T1CLKO	O	Clock out of timer 1
		MOSI_4	I/O	Master Output/Slave Input of SPI
		PWM4P_4	I/O	Capture of external signal/ Positive of PWM4 pulse output
		PWM8_2	I/O	Capture of external signal/ Positive of PWM8 pulse output
		CMPO	O	Comparator output
18		P3.5	I/O	Standard IO port
		T1	I	Timer1 external input
		T0CLKO	O	Clock out of timer 0
		SS_4	I/O	Slave selection of SPI
		PWMFLT	I	PWMA external anomaly detection pin
		PWMFLT2	I	PWMB external anomaly detection pin
19		P3.6	I/O	Standard IO port
		INT2	I	External interrupt 2
		RxD_2	I	Serial input of UART1
		CMP-	I	Comparator negative input
20		P3.7	I/O	Standard IO port
		INT3	I	External interrupt 3
		TxD_2	O	Serial Transmit pin of UART 1
		CMP+	I	Comparator positive input

Pin number	name	type	description

LQFP32/QFN32				
21		P2.0	I/O	Standard IO port
		PWM1P_2	I/O	Capture of external signal/ Positive of PWMA pulse output
		PWM5	I/O	Capture of external signal/ Positive of PWM5 pulse output
22		P2.1	I/O	Standard IO port
		PWM1N_2	I/O	Capture of external signal/ Negative of PWMA pulse output
		PWM6	I/O	Capture of external signal/ Positive of PWM6 pulse output
23		P2.2	I/O	Standard IO port
		PWM2P_2	I/O	Capture of external signal/ Positive of PWMB pulse output
		PWM7	I/O	Capture of external signal/ Positive of PWM7 pulse output
		SS_2	I/O	Slave selection of SPI
24		P2.3	I/O	Standard IO port
		PWM2N_2	I/O	Capture of external signal/ Negative of PWMB pulse output
		PWM8	I/O	Capture of external signal/ Positive of PWM8 pulse output
		MOSI_2	I/O	Master Output/Slave Input of SPI
25		P2.4	I/O	Standard IO port
		PWM3P_2	I/O	Capture of external signal/ Positive of PWM3 pulse output
		MISO_2	I/O	Serial Clock line of I2C
		SDA_2	I/O	Serial data line of I2C
26		P2.5	I/O	Standard IO port
		PWM3N_2	I/O	Capture of external signal/ Negative of PWM3 pulse output
		SCLK_2	I/O	Serial Clock of SPI
		SCL_2	I/O	Serial Clock line of I2C
27		P2.6	I/O	Standard IO port
		PWM4P_2	I/O	Capture of external signal/ Positive of PWM4 pulse output
28		P2.7	I/O	Standard IO port
		PWM4N_2	I/O	Capture of external signal/ Negative of PWM4 pulse output
29		P0.0	I/O	Standard IO port
		ADC8	I	ADC analog input 8
		RxD3	I	Serial input of UART3
		T3	I	Timer3 external input
		PWM5_3	I/O	Capture of external signal/ Positive of PWM5 pulse output
30		P0.1	I/O	Standard IO port
		ADC9	I	ADC analog input 9
		TxD3	O	Serial Transmit pin of UART 3
		T3CLKO	O	Clock out of timer 3
		PWM6_3	I/O	Capture of external signal/ Positive of PWM6 pulse output

Pin number		name	type	description
LQFP32/QFN32				
31		P0.2	I/O	Standard IO port
		ADC10	I	ADC analog input 10
		RxD4	I	Serial input of UART4

		T4	I	Timer4 external input
		PWM7_3	I/O	Capture of external signal/ Positive of PWM7 pulse output
32		P0.3	I/O	Standard IO port
		ADC11	I	ADC analog input 11
		TxD4	O	Serial Transmit pin of UART 4
		T4CLKO	O	Clock out of timer 4
		PWM8_3	I/O	Capture of external signal/ Positive of PWM8 pulse output

2.3 STC8H3K64S2-45I-LQFP48/32,QFN48/32,TSSOP20 family

2.3.1 Features and Price(Quasi 16-bit MCU with 16-bit hardware multiplier and divider MDU16)

- **Selection and price (No external crystal and external reset required with 12 channels 12-bit ADC)**

MCU	Operating voltage（V）	Flash Code Memory (100 thousand times) (Byte)	idata Internal DATA RAM	xdata Internal extended SRAM (Byte)	Enhanced Dual DPTR increasing or decreasing	EEPROM 100 thousand times) (Byte)	Maximum I/O Lines	Traditional I/O interrupt(INT0/INT1/INT2/INT3/INT4) (can wake-up CPU)	All I/O ports support interrupts and can wake up MCU	UARTs which can wake-up CPU	SPI which can wake-up CPU	I^2C which can wake-up CPU	MDU16 (Hardware 16-bit Multiplier and Divider)	Timers/Counters (T0-T4 Pin Can wake-up CPU)	16-bit advanced PWM timer with Complementary symmetrical dead-time	Power-down Wake-up timer	12-bit high speed ADC (8 PWMs can be used as 8 DACs)	Comparator (May be used as ADC to detect external power-down)	Internal LVD interrupt (can wake-up CPU)	Watch-dog Timer	Internal high reliable reset circuit with 4 levels optional reset threshold voltage	Internal high precision Clock (adjustbal under 45MHz)	Clock output and Reset	Program encrypted transmission (Anti-blocking)	Password can be set for next update	Support RS485 download	Support software USB download directly	Online debug itself	Price & Package					products supply information
																													LQFP48 <9mm*9mm>	QFN48 <6mm*6mm>	LQFP32<9mm*9mm>	QFN32<4mm*4mm>	TSSOP20 <6.5mm*6.5mm>	
STC8H3K32S2	1.9-5.5	32K	256	3K	2	32K	43	Y	Y	2	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	¥3. 6		¥3. 5		¥2. 5	Available
STC8H3K48S2	1.9-5.5	48K	256	3K	2	16K	43	Y	Y	2	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	¥3. 7	√	¥3. 6	√	¥2. 6	
~~STC8H3K60S2~~	~~1.9-5.5~~	~~60K~~	~~256~~	~~3K~~	~~2~~	~~4K~~	~~43~~	~~Y~~	~~Y~~	~~2~~	~~Y~~	~~Y~~	~~Y~~	~~5~~	~~8~~	~~Y~~	~~12bit~~	~~Y~~	~~Y~~	~~Y~~	~~Y~~	~~Y~~	~~Y~~	~~Y~~	~~Y~~	~~Y~~	~~Y~~	~~Y~~	~~√~~	~~√~~	~~√~~	~~√~~	~~√~~	
STC8H3K64S2	1.9-5.5	64K	256	3K	2	IAP	43	Y	Y	2	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	¥3. 8	√	¥3. 7	√	¥2. 7	

- **Core**
  - ✓ Ultra-high speed 8051 Core with single clock per machine cycle, which is called 1T and the speed is about 12 times faster than traditional 8051
  - ✓ Fully compatible instruction set with traditional 8051
  - ✓ 19 interrupt sources and 4 interrupt priority levels
  - ✓ Online debugging is supported

- **Operating voltage**
  - ✓ 1.9V～5.5V

- **Operating temperature**
  - ✓ -40℃~85℃(The chip is produced in -40℃～125℃ process. Please refer to the description of the electrical characteristics chapter for applications beyond the temperature range)

- **Flash memory**
  - ✓ Up to 64Kbytes of Flash memory to be used for storing user code
  - ✓ Configurable EEPROM size, 512bytes single page for being erased, which can be repeatedly erased more than 100 thousand times.
  - ✓ In-System-Programming, ISP in short, can be used to update the application code. No special programmer is needed.
  - ✓ Online debugging with single chip is supported, and no special emulator is needed. The number of breakpoints is unlimited theoratically.

- **SRAM**
  - ✓ 128 bytes internal direct access RAM (DATA, use keyword *data* to declare in C language program)
  - ✓ 128 bytes internal indirect access RAM(IDATA, use keyword *idata* to declare in C language program)
  - ✓ 3072 bytes internal extended RAM (internal XDATA, use keyword *xdata* to declare in C language program)

- **Clock**
  - Internal high precise RC clock IRC(IRC for short, ranges from 4MHz to 45MHz), adjustable while ISP and can be divided to lower frequency by user software, 100KHz for instance.
    - Error:±0.3% (at the temperature 25℃)
    - -1.35%～+1.30% temperature drift (at the temperature range of -40℃ to +85℃)
    - -0.76%～+0.98% temperature drift (at the temperature range of -20℃ to 65℃)
  - Internal 32KHz low speed IRC with large error
  - External crystal (4MHz～45MHz) and external clock

  The three clock sources above can be selected freely by user code.
  - Important note about the internal high-speed IRC of STC8H3K64S2 series B version products
    - Due to manufacturing reasons, the internal high-speed IRC of some chips may have a blind area between 34MHz and 36MHz. It is recommended not to set the operating frequency in this area.
    - The temperature drift of the internal high-speed IRC at low temperature is larger than that at the higher temperature, and the temperature drift in the low frequency range is larger than that in the high frequency range. Generally, the operating frequency of 20MHz~40MHz, the temperature drift at 85℃ can be controlled within 0.8%.

- **Reset**
  - Hardware reset
    - Power-on reset. (Effective when the chip does not enable the low voltage reset function)
    - Reset by reset pin. The default function of P5.4 is the I/O port. The P5.4 pin can be set as the reset pin while ISP download. (Note: When the P5.4 pin is set as the reset pin, the reset level is low.)
    - Watch dog timer reset
    - Low voltage detection reset. 4 low voltage detection levels are provided, 2.0V, 2.4V, V2.7, V3.0.
  - Software reset
    - Writing the reset trigger register using software

- **Interrupts**
  - 19 interrupt sources: INT0(Supports rising edge and falling edge interrupt), INT1(Supports rising edge and falling edge interrupt), INT2(Supports falling edge interrupt only), INT3(Supports falling edge interrupt only), INT4(Supports falling edge interrupt only), timer 0, timer 1, timer 2, timer 3, timer 4, UART 1, UART 2, ADC, LVD, SPI, I^{2}C, comparator, PWMA, PWMB
  - 4 interrupt priority levels
  - Interrupts that can wake up the CPU in clock stop mode: INT0(P3.2), INT1(P3.3), INT2(P3.6), INT3(P3.7), INT4(P3.0), T0(P3.4), T1(P3.5), T2(P1.2), T3(P0.4), T4(P0.6), RXD(P3.0/P3.6/P1.6/P4.3), RXD2(P1.0/P4.6), I2C_SDA(P1.4/P2.4/P3.3), Comparator interrupt, LVD interrupt, Power-down wake-up timer.

- **Digital peripherals**
  - 5 16-bit timers: timer0, timer1, timer2, timer3, timer4, where the mode 3 of timer 0 has the Non-Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
  - 2 high speed UARTs: UART1, UART2, whose maximum baudrate clock may be FOSC/4
  - 8 channels/2 groups of enhanced PWMs, which can realize control signals with dead time, and support external fault detection function. In addition, it also supports 16-bit timers, 8 external interrupts, 8 external captures and pulse width measurement functions.
  - SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
  - I^{2}C: Master mode or slave mode are supported.
  - MDU16: Hardware 16-bit Multiplier and Divider which supports 32-bit divided by 16-bit, 16-bit divided by 16-bit, 16-bit multiplied by 16-bit, data shift, and data normalization operations.
  - I/O port interrupt: All I/Os support interrupts, each group of I/O interrupts has an independent interrupt entry address, all I/O interrupts can support 4 types interrupt mode: high level interrupt, low level interrupt, rising edge interrupt, falling edge interrupt. (Note: The I/O port interrupts of the STC8H3K64S2 series A version of the chip cannot wake up CPU from power-down. The I/O port interrupts of the B version chip can wake up CPU from power-down, but only have one level of interrupt priority)

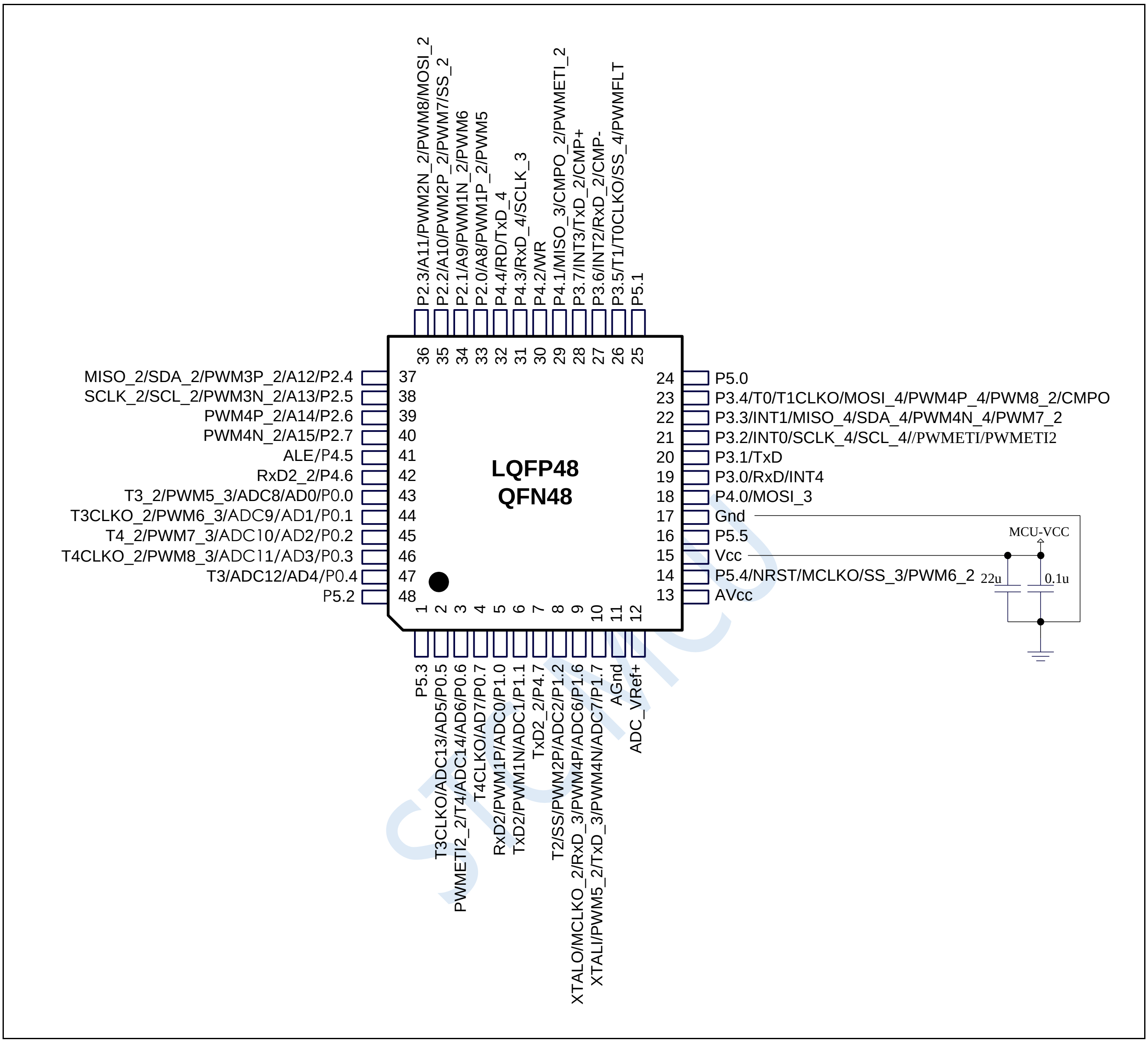
- **Analog peripherals**
  - Ultra high speed ADC which supports 12-bit precision 12 channels analog-to-digital convertor (channel 0 to channel 2, channel 6 to channel 14. No channel 3 and channel 5 because P1.3, P1.4 and P1.5 do not exist). The maximum speed can be 800K(800K ADC conversions per second)
  - Channel 15 of ADC is used to test the internal reference voltage. (The default internal reference voltage is 1.19V when the chip is shipped)
  - A set of comparator (the CMP+ port and all ADC input ports can be selected as the positive terminal of the comparator, so the comparator can be used as a multi-channel comparator for time division multiplexing)
  - DAC: 8 channels advanced PWMs timers can be used as 8 channels DAC

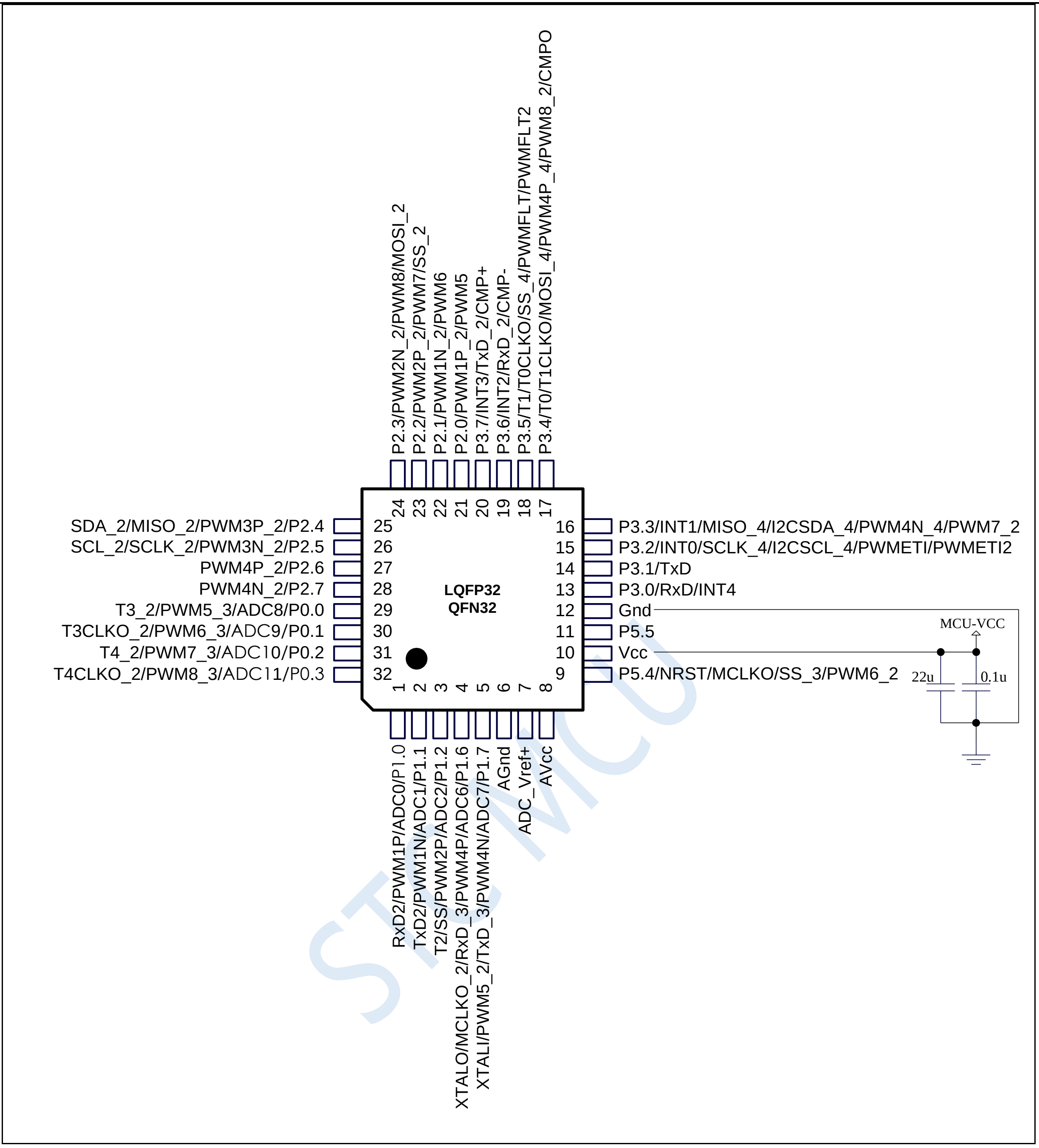
- **GPIO**
  - Up to 43 GPIOs: P0.0~P0.7, P1.0~P1.2, P1.6~P1.7, P2.0~P2.7, P3.0~P3.7, P4.0~P4.7, P5.0~P5.5
  - 4 modes for all GPIOs: quasi_bidirectional mode, push-pull output mode, open drain mode, high-impedance input mode
  - Except for P3.0 and P3.1, all other I/O ports are in a high-impedance state after power-on. User must set the I/O ports mode before using them. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.

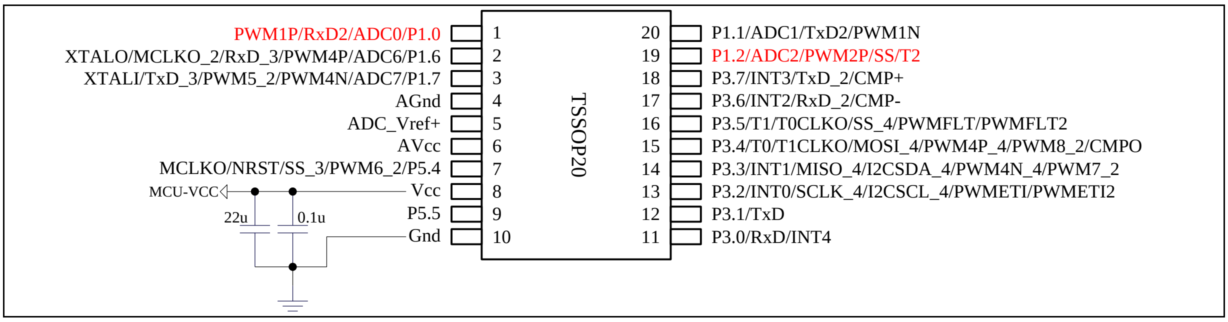
- **Package**
  - LQFP48 <9mm*9mm>, QFN48 <6mm*6mm>, LQFP32 <9mm*9mm>, QFN32 <4mm*4mm>, TSSOP20

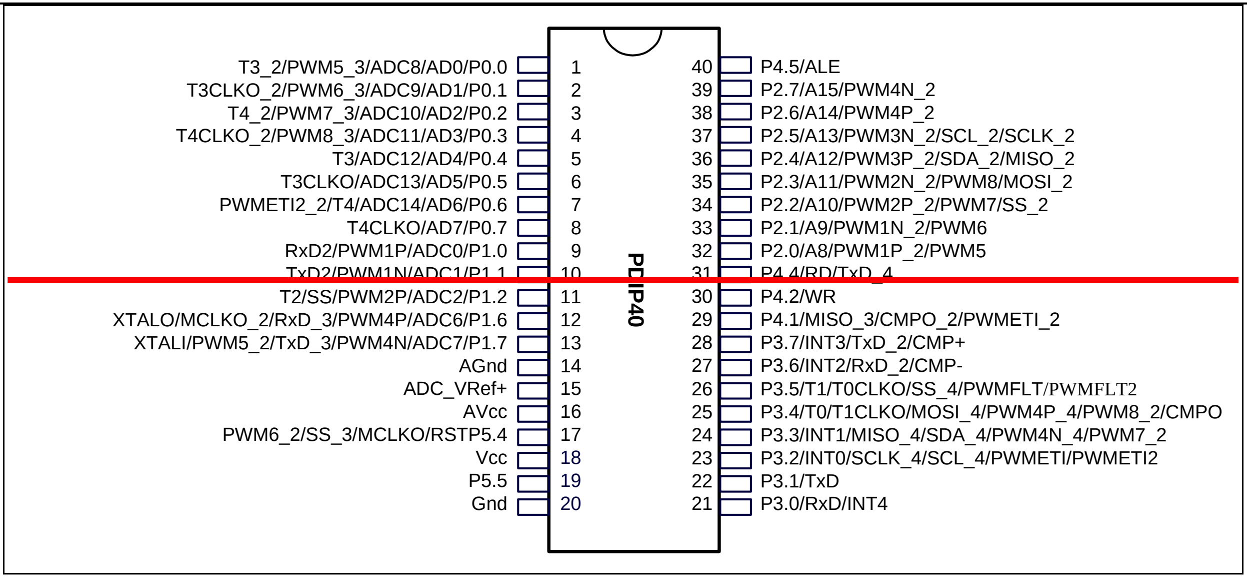
<6.5mm*6.5mm> (There are no samples for LQFP32, QFN32, TSSOP2 at the moment, there will be later, please order in advance if necessary.)

2.3.2 Pinouts


LQFP48
QFN48
P2.3/A11/PWM2N_2/PWM8/MOSI_2 36
P2.2/A10/PWM2P_2/PWM7/SS_2 35
P2.1/A9/PWM1N_2/PWM6 34
P2.0/A8/PWM1P_2/PWM5 33
P4.4/RD/TxD_4 32
P4.3/RxD_4/SCLK_3 31
P4.2/WR 30
P4.1/MISO_3/CMPO_2/PWMETI_2 29
P3.7/INT3/TxD_2/CMP+ 28
P3.6/INT2/RxD_2/CMP- 27
P3.5/T1/T0CLKO/SS_4/PWMFLT 26
P5.1 25
MISO_2/SDA_2/PWM3P_2/A12/P2.4 37
SCLK_2/SCL_2/PWM3N_2/A13/P2.5 38
PWM4P_2/A14/P2.6 39
PWM4N_2/A15/P2.7 40
ALE/P4.5 41
RxD2_2/P4.6 42
T3_2/PWM5_3/ADC8/AD0/P0.0 43
T3CLKO_2/PWM6_3/ADC9/AD1/P0.1 44
T4_2/PWM7_3/ADC10/AD2/P0.2 45
T4CLKO_2/PWM8_3/ADC11/AD3/P0.3 46
T3/ADC12/AD4/P0.4 47
P5.2 48
24 P5.0
23 P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO
22 P3.3/INT1/MISO_4/SDA_4/PWM4N_4/PWM7_2
21 P3.2/INT0/SCLK_4/SCL_4//PWMETI/PWMETI2
20 P3.1/TxD
19 P3.0/RxD/INT4
18 P4.0/MOSI_3
17 Gnd
16 P5.5
15 Vcc
14 P5.4/NRST/MCLKO/SS_3/PWM6_2
13 AVcc
MCU-VCC
22u
0.1u
1 P5.3
2 T3CLKO/ADC13/AD5/P0.5
3 PWMETI2_2/T4/ADC14/AD6/P0.6
4 T4CLKO/AD7/P0.7
5 RxD2/PWM1P/ADC0/P1.0
6 TxD2/PWM1N/ADC1/P1.1
7 TxD2_2/P4.7
8 T2/SS/PWM2P/ADC2/P1.2
9 XTALO/MCLKO_2/RxD_3/PWM4P/ADC6/P1.6
10 XTALI/PWM5_2/TxD_3/PWM4N/ADC7/P1.7
11 AGnd
12 ADC_VRef+


P2.3/PWM2N_2/PWM8/MOSI_2
P2.2/PWM2P_2/PWM7/SS_2
P2.1/PWM1N_2/PWM6
P2.0/PWM1P_2/PWM5
P3.7/INT3/TxD_2/CMP+
P3.6/INT2/RxD_2/CMP-
P3.5/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2
P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO
24
23
22
21
20
19
18
17
SDA_2/MISO_2/PWM3P_2/P2.4
SCL_2/SCLK_2/PWM3N_2/P2.5
PWM4P_2/P2.6
PWM4N_2/P2.7
T3_2/PWM5_3/ADC8/P0.0
T3CLKO_2/PWM6_3/ADC9/P0.1
T4_2/PWM7_3/ADC10/P0.2
T4CLKO_2/PWM8_3/ADC11/P0.3
25
26
27
28
29
30
31
32
LQFP32
QFN32
16
15
14
13
12
11
10
9
P3.3/INT1/MISO_4/I2CSDA_4/PWM4N_4/PWM7_2
P3.2/INT0/SCLK_4/I2CSCL_4/PWMETI/PWMETI2
P3.1/TxD
P3.0/RxD/INT4
Gnd
P5.5
Vcc
P5.4/NRST/MCLKO/SS_3/PWM6_2
MCU-VCC
22u
0.1u
1
2
3
4
5
6
7
8
RxD2/PWM1P/ADC0/P1.0
TxD2/PWM1N/ADC1/P1.1
T2/SS/PWM2P/ADC2/P1.2
XTALO/MCLKO_2/RxD_3/PWM4P/ADC6/P1.6
XTALI/PWM5_2/TxD_3/PWM4N/ADC7/P1.7
AGnd
ADC_Vref+
AVcc


PWM1P/RxD2/ADC0/P1.0
XTALO/MCLKO_2/RxD_3/PWM4P/ADC6/P1.6
XTALI/TxD_3/PWM5_2/PWM4N/ADC7/P1.7
AGnd
ADC_Vref+
AVcc
MCLKO/NRST/SS_3/PWM6_2/P5.4
MCU-VCC
Vcc
22u
0.1u
P5.5
Gnd
1
2
3
4
5
6
7
8
9
10
TSSOP20
20
19
18
17
16
15
14
13
12
11
P1.1/ADC1/TxD2/PWM1N
P1.2/ADC2/PWM2P/SS/T2
P3.7/INT3/TxD_2/CMP+
P3.6/INT2/RxD_2/CMP-
P3.5/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2
P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO
P3.3/INT1/MISO_4/I2CSDA_4/PWM4N_4/PWM7_2
P3.2/INT0/SCLK_4/I2CSCL_4/PWMETI/PWMETI2
P3.1/TxD
P3.0/RxD/INT4


PDIP40
T3_2/PWM5_3/ADC8/AD0/P0.0 1
T3CLKO_2/PWM6_3/ADC9/AD1/P0.1 2
T4_2/PWM7_3/ADC10/AD2/P0.2 3
T4CLKO_2/PWM8_3/ADC11/AD3/P0.3 4
T3/ADC12/AD4/P0.4 5
T3CLKO/ADC13/AD5/P0.5 6
PWMETI2_2/T4/ADC14/AD6/P0.6 7
T4CLKO/AD7/P0.7 8
RxD2/PWM1P/ADC0/P1.0 9
TxD2/PWM1N/ADC1/P1.1 10
T2/SS/PWM2P/ADC2/P1.2 11
XTALO/MCLKO_2/RxD_3/PWM4P/ADC6/P1.6 12
XTALI/PWM5_2/TxD_3/PWM4N/ADC7/P1.7 13
AGnd 14
ADC_VRef+ 15
AVcc 16
PWM6_2/SS_3/MCLKO/RSTP5.4 17
Vcc 18
P5.5 19
Gnd 20
40 P4.5/ALE
39 P2.7/A15/PWM4N_2
38 P2.6/A14/PWM4P_2
37 P2.5/A13/PWM3N_2/SCL_2/SCLK_2
36 P2.4/A12/PWM3P_2/SDA_2/MISO_2
35 P2.3/A11/PWM2N_2/PWM8/MOSI_2
34 P2.2/A10/PWM2P_2/PWM7/SS_2
33 P2.1/A9/PWM1N_2/PWM6
32 P2.0/A8/PWM1P_2/PWM5
31 P4.4/RD/TxD_4
30 P4.2/WR
29 P4.1/MISO_3/CMPO_2/PWMETI_2
28 P3.7/INT3/TxD_2/CMP+
27 P3.6/INT2/RxD_2/CMP-
26 P3.5/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2
25 P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO
24 P3.3/INT1/MISO_4/SDA_4/PWM4N_4/PWM7_2
23 P3.2/INT0/SCLK_4/SCL_4/PWMETI/PWMETI2
22 P3.1/TxD
21 P3.0/RxD/INT4

Note:

1. ADC's external reference power supply pin ADC_VRef+ must not be floating. It must be connected to an external reference power supply or directly connected to Vcc.
2. If USB download is not required, P3.0/P3.1/P3.2 cannot be at low level at the same time when the chip is reset.

The download steps using ISP and notes are the same as the circumstances in 2.1.2.

2.3.3 Pin descriptions

Pin number				name	type	description
LQFP48 QFN48	LQFP32 QFN32	TSSOP20				
1				P5.3	I/O	Standard IO port
2				P0.5	I/O	Standard IO port
				AD5	I	Address/data bus
				ADC13	I	ADC analog input 13
				T3CLKO	O	Clock out of timer 3
3				P0.6	I/O	Standard IO port
				AD6	I	Address/data bus
				ADC14	I	ADC analog input 14
				T4	I	Timer4 external input
				PWMETI2_2	I	Enhance PWM external anomaly detection pin2
4				P0.7	I/O	Standard IO port
				AD7	I	Address/data bus
				T4CLKO	O	Clock out of timer 4
5	1	1		P1.0	I/O	Standard IO port
				ADC0	I	ADC analog input 0
				PWM1P	I/O	Capture of external signal/Positive of PWMA pulse output
				RxD2	I	Serial input of UART2
6	2	20		P1.1	I/O	Standard IO port
				ADC1	I	ADC analog input 1
				PWM1N	I/O	Capture of external signal/Negative of PWMA pulse output
				TxD2	I	Serial Transmit pin of UART 2
7				P4.7	I/O	Standard IO port
				TxD2_2	I	Serial Transmit pin of UART 2
8	3	19		P1.2	I/O	Standard IO port
				ADC2	I	ADC analog input
				PWM2P	I/O	Capture of external signal/ Positive of PWM2 pulse output
				SS	I	Slave selection of SPI (it is output with regard to master)
				T2	I	Timer2 external input
9	4	2		P1.6	I/O	Standard IO port
				ADC6	I	ADC analog input 6
				RxD_3	I	Serial input of UART1
				PWM4P	I/O	Capture of external signal/Positive of PWM4 pulse output
				MCLKO_2	O	Master clock output
				XTALO	O	Connect to external oscillator
10	5	3		P1.7	I/O	Standard IO port
				ADC7	I	ADC analog input 7
				TxD_3	O	Serial Transmit pin of UART 1
				PWM4N	I/O	Capture of external signal/Negative of PWM4 pulse output
				PWM5_2	I/O	Capture of external signal/ Pulse output of PWM5
				XTALI	I	Input pin of external crystal oscillator/external clock

Pin number				name	type	description
LQFP48 QFN48	LQFP32 QFN32	TSSOP20				

Pin number				name	type	description
LQFP48 QFN48	LQFP32 QFN32	TSSOP20				
11	6	4		AGnd	Gnd	ADC Ground
12	7	5		ADC_VRef+	I	ADC external reference voltage source input pin, which can be directly connected to MCU VCC when the requirements are not high
13	8	6		AVcc	Vcc	ADC Power Supply
14	9	7		P5.4	I/O	Standard IO port
				NRST	I	Reset pin (MCU will reset when it is low level)
				MCLKO	O	Main clock output
				SS_3	I	Slave selection of SPI (it is output with regard to master)
				PWM6_2	I/O	Capture of external signal/ Pulse output of PWM6
15	10	8		Vcc	Vcc	Power Supply
16	11	9		P5.5	I/O	Standard IO port
17	12	10		Gnd	Gnd	Ground
18				P4.0	I/O	Standard IO port
				MOSI_3	I/O	Master Output/Slave Input of SPI
19	13	11		P3.0	I/O	Standard IO port
				RxD	I	Serial input of UART1
				INT4	I	External interrupt4
20	14	12		P3.1	I/O	Standard IO port
				TxD	O	Serial Transmit pin of UART 1
21	15	13		P3.2	I/O	Standard IO port
				INT0	I	External interrupt0
				SCLK_4	I/O	Clock of SPI
				SCL_4	I/O	Clock line of I2C
				PWMETI	I	External trigger input pin of PWM
				PWMETI2	I	External trigger input pin PWM2
22	16	14		P3.3	I/O	Standard IO port
				INT1	I	External interrupt1
				MISO_4	I/O	Master Iutput/Slave Onput of SPI
				SDA_4	I/O	Data line of I2C
				PWM4N_4	I/O	Capture input /pulse negative output of PWM4
				PWM7_2	I/O	Capture input /pulse output of PWM7
23	17	15		P3.4	I/O	Standard IO port
				T0	I	Timer0 external input
				T1CLKO	O	Clock out of timer 1
				MOSI_4	I/O	Master Output/Slave Input of SPI
				PWM4P_4	I/O	Capture input / pulse positive output of PWM4
				PWM8_2	I/O	Capture input / pulse output of PWM8
				CMPO	O	Output of comparator
24				P5.0	I/O	Standard IO port
25				P5.1	I/O	Standard IO port
26	18	16		P3.5	I/O	Standard IO port
				T1	I	Timer1 external input
				T0CLKO	O	Clock out of timer 0
				SS_4	I	Slave selection of SPI (it is output with regard to master)
				PWMFLT	I	External abnormal detection pin of Enhanced PWM
27	19	17		P3.6	I/O	Standard IO port
				INT2	I	External interrupt2
				RxD_2	I	Serial input of UART1
				CMP-	I	Negative input of comparator
28	20	18		P3.7	I/O	Standard IO port
				INT3	I	External interrupt3
				TxD_2	O	Serial Transmit pin of UART 1

				CMP+	I	Positive input of comparator
29				P4.1	I/O	Standard IO port
				MISO_3	I/O	Master Iutput/Slave Onput of SPI
				CMPO_2	O	Output of comparator
				PWMETI_2	I	External trigger input pin of PWM
30				P4.2	I/O	Standard IO port
				WR	O	Write signal of external bus
31				P4.3	I/O	Standard IO port
				RxD_4	I	Serial input of UART1
				SCLK_3	I/O	Clock of SPI
32				P4.4	I/O	Standard IO port
				RD	O	Read signal of external bus
				TxD_4	O	Serial Transmit pin of UART 1
33	21			P2.0	I/O	Standard IO port
				A8	I	Address bus
				PWM1P_2	I/O	Capture of external signal/Pulse positive output of PWMA
				PWM5	I/O	Capture of external signal/Pulse output of PWM5
34	22			P2.1	I/O	Standard IO port
				A9	I	Address bus
				PWM1N_2	I/O	Capture of external signal/Pulse negative output of PWMA
				PWM6	I/O	Capture of external signal/Pulse output of PWM6
35	23			P2.2	I/O	Standard IO port
				A10	I	Address bus
				SS_2	I	Slave selection of SPI (it is output with regard to master)
				PWM2P_2	I/O	Capture of external signal/Pulse positive output of PWMB
				PWM7	I/O	Capture of external signal/Pulse output of PWM7
36	24			P2.3	I/O	Standard IO port
				A11	I	Address bus
				MOSI_2	I/O	Master Output/Slave Input of SPI
				PWM2N_2	I/O	Capture of external signal/Pulse negative output of PWMB
				PWM8	I/O	Capture of external signal/Pulse output of PWM8
37	25			P2.4	I/O	Standard IO port
				A12	I	Address bus
				MISO_2	I/O	Master Iutput/Slave Onput of SPI
				SDA_2	I/O	Data line of I2C
				PWM3P_2	I/O	Capture of external signal/Pulse positive output of PWM3

Pin number				name	type	description
LQFP48 QFN48	LQFP32 QFN32	TSSOP20				
38	26			P2.5	I/O	Standard IO port
				A13	I	Address bus
				SCLK_2	I/O	Clock of SPI
				SCL_2	I/O	Clock line of I2C
				PWM3N_2	I/O	Capture of external signal/Pulse negative output of PWM3
39	27			P2.6	I/O	Standard IO port
				A14	I	Address bus
				PWM4P_2	I/O	Capture of external signal/Pulse positive output of PWM4

40	28			P2.7	I/O	Standard IO port
				A15	I	Address bus
				PWM4N_2	I/O	Capture of external signal/Pulse negative output of PWM4
41				P4.5	I/O	Standard IO port
				ALE	O	Address Latch Enable signal
42				P4.6	I/O	Standard IO port
				RxD2_2	I	Serial input of UART2
43	29			P0.0	I/O	Standard IO port
				AD0	I	Address/Data bus
				ADC8	I	ADC analog input channel 8
				PWM5_3	I/O	Capture of external signal/Pulse output of PWM5
				T3_2	I	Timer3 external input
44	30			P0.1	I/O	Standard IO port
				AD1	I	Address/Data bus
				ADC9	I	ADC analog input channel 9
				PWM6_3	I/O	Capture of external signal/Pulse output of PWM6
				T3CLKO_2	O	Clock out of timer 3
45	31			P0.2	I/O	Standard IO port
				AD2	I	Address/Data bus
				ADC10	I	ADC analog input channel 10
				PWM7_3	I/O	Capture of external signal/Pulse output of PWM7
				T4_2	I	Timer4 external input
46	32			P0.3	I/O	Standard IO port
				AD3	I	Address/Data bus
				ADC11	I	ADC analog input channel 11
				PWM8_3	I/O	Capture of external signal/Pulse output of PWM8
				T4CLKO_2	O	Clock out of timer 4
47				P0.4	I/O	Standard IO port
				AD4	I	Address/Data bus
				ADC12	I	ADC analog input channel 12
				T3	I	Timer3 external input
48				P5.2	I/O	Standard IO port

2.4 STC8H3K64S4-45I-LQFP48/32, QFN48/32, TSSOP20 family

2.4.1 Features and Price(Quasi 16-bit MCU with 16-bit hardware multiplier and divider MDU16)

- **Selection and price (No external crystal and external reset required with 12 channels 12-bit ADC)**

MCU	Operating voltage（V）	Flash Code Memory (100 thousand times) (Byte)	idata Internal DATA RAM (Byte)	xdata Internal extended SRAM (Byte)	Enhanced Dual DPTR increasing or decreasing	EEPROM 100 thousand times) (Byte)	Maximum I/O Lines	Traditional I/O interrupt(INT0/INT1/INT2/INT3/INT4) (can wake-up CPU)	All I/O ports support interrupts and can wake up MCU	UARTs which can wake-up CPU	SPI which can wake-up CPU	I²C which can wake-up CPU	MDU16 (Hardware 16-bit Multiplier and Divider)	Timers/Counters (T0-T4 Pin Can wake-up CPU)	16-bit advanced PWM timer with Complementary symmetrical dead-time	Power-down Wake-up timer	12-bit high speed ADC (8 PWMs can be used as 8 DACs)	Comparator (May be used as ADC to detect external power-down)	Internal LVD interrupt (can wake-up CPU)	Watch-dog Timer	Internal high reliable reset circuit with 4 levels optional reset threshold voltage	Internal high presision Clock (adjustbal under 45MHz)	Clock output and Reset	Program encrypted transmission (Anti-blocking)	Password can be set for next update	Support RS485 download	Support software USB download directly	Online debug itself	Price & Package				products supply information
																													LQFP48 <9mm*9mm>	QFN48 <6mm*6mm>	LQFP32<9mm*9mm>	QFN32<4mm*4mm>	
~~STC8H3K32S4~~	~~1.9-5.5~~	~~32K~~	~~256~~	~~3K~~	~~2~~	~~32K~~	~~43~~	~~Y~~	~~Y~~	~~4~~	~~Y~~	~~Y~~	~~Y~~	~~5~~	~~8~~	~~Y~~	~~12bit~~	~~Y~~	~~Y~~	~~Y~~	~~Y~~	~~Y~~	~~Y~~	~~Y~~	~~Y~~	~~Y~~	~~Y~~	~~Y~~	~~√~~	~~√~~	~~√~~	~~√~~	Available
STC8H3K48S4	1.9-5.5	48K	256	3K	2	16K	43	Y	Y	4	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	¥4	√	¥4	√	
STC8H3K60S4	1.9-5.5	60K	256	3K	2	4K	43	Y	Y	4	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	√	√	√	√	
STC8H3K64S4	1.9-5.5	64K	256	3K	2	IAP	43	Y	Y	4	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	¥4	¥4	¥4	¥4	

- **Core**
  - ✓ Ultra-high speed 8051 Core with single clock per machine cycle, which is called 1T and the speed is about 12 times faster than traditional 8051
  - ✓ Fully compatible instruction set with traditional 8051
  - ✓ 21 interrupt sources and 4 interrupt priority levels
  - ✓ Online debugging is supported

- **Operating voltage**
  - ✓ 1.9V～5.5V

- **Operating temperature**
  - ✓ -40℃~85℃ (In order to work in a wider temperature range, please use an external clock or use a lower operating frequency)

- **Flash memory**
  - ✓ Up to 64Kbytes of Flash memory to be used for storing user code
  - ✓ Configurable EEPROM size, 512bytes single page for being erased, which can be repeatedly erased more than 100 thousand times.
  - ✓ In-System-Programming, ISP in short, can be used to update the application code. No special programmer is needed.
  - ✓ Online debugging with single chip is supported, and no special emulator is needed. The number of breakpoints is unlimited theoratically.

- **SRAM**
  - ✓ 128 bytes internal direct access RAM (DATA, use keyword *data* to declare in C language program)
  - ✓ 128 bytes internal indirect access RAM (IDATA, use keyword *idata* to declare in C language program)
  - ✓ 3072 bytes internal extended RAM (internal XDATA, use keyword *xdata* to declare in C language program)

- ➢ **Clock**
  - ✓ Internal high precise RC clock IRC(IRC for short, ranges from 4MHz to 45MHz), adjustable while ISP and can be divided to lower frequency by user software, 100KHz for instance.
    - ✓ Error:±0.3% (at the temperature 25℃)
    - ✓ -1.35%～+1.30% temperature drift (at the temperature range of -40℃ to +85℃)
    - ✓ -0.76%～+0.98% temperature drift (at the temperature range of -20℃ to 65℃)
  - ✓ Internal 32KHz low speed IRC with large error
  - ✓ External crystal (4MHz～45MHz) and external clock

  The three clock sources above can be selected freely by user code.
  - ✓ Important note about the internal high-speed IRC of STC8H3K64S4 series B version products
    - ✓ Due to manufacturing reasons, the internal high-speed IRC of some chips may have a blind area between 34MHz and 36MHz. It is recommended not to set the operating frequency in this area.
    - ✓ The temperature drift of the internal high-speed IRC at low temperature is larger than that at the higher temperature, and the temperature drift in the low frequency range is larger than that in the high frequency range. Generally, the operating frequency of 20MHz~40MHz, the temperature drift at 85℃ can be controlled within 0.8%.

- ➢ **Reset**
  - ✓ Hardware reset
    - ✓ Power-on reset.(Effective when the chip does not enable the low voltage reset function)
    - ✓ Reset by reset pin. The default function of P5.4 is the I/O port. The P5.4 pin can be set as the reset pin while ISP download. (Note: When the P5.4 pin is set as the reset pin, the reset level is low.)
    - ✓ Watch dog timer reset
    - ✓ Low voltage detection reset. 4 low voltage detection levels are provided, 2.0V, 2.4V, V2.7, V3.0.
  - ✓ Software reset
    - ✓ Writing the reset trigger register using software

- ➢ **Interrupts**
  - ✓ 21 interrupt sources: INT0(Supports rising edge and falling edge interrupt), INT1(Supports rising edge and falling edge interrupt), INT2(Supports falling edge interrupt only), INT3(Supports falling edge interrupt only), INT4(Supports falling edge interrupt only), timer 0, timer 1, timer 2, timer 3, timer 4, UART 1, UART 2, UART 3, UART 4, ADC, LVD, SPI, I^2C, comparator, PWMA, PWMB
  - ✓ 4 interrupt priority levels
  - ✓ interruption that can wake up the CPU in clock stop mode：INT0(P3.2), INT1(P3.3), INT2(P3.6), INT3(P3.7), INT4(P3.0), T0(P3.4), T1(P3.5), T2(P1.2), T3(P0.4), T4(P0.6), RXD(P3.0/P3.6/P1.6/P4.3), RXD2(P1.0/P4.6), RXD3(P0.0/P5.0), RXD4(P0.2/P5.2), I2C_SDA(P1.4/P2.4/P3.3), Comparator interrupt, LVD interrupt, Power-down wake-up timer.

- ➢ **Digital peripherals**
  - ✓ 5 16-bit timers: timer0, timer1, timer2, timer3, timer4, where the mode 3 of timer 0 has the Non-Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
  - ✓ 4 high speed UARTs: UART1, UART 2, UART 3, UART 4, whose maximum baudrate may be FOSC/4.
  - ✓ 8 channels/2 groups of enhanced PWMs, which can realize control signals with dead time, and support external fault detection function. In addition, it also supports 16-bit timers, 8 external interrupts, 8 external captures and pulse width measurement functions.
  - ✓ SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
  - ✓ I^2C: Master mode or slave mode are supported.
  - ✓ MDU16: Hardware 16-bit Multiplier and Divider which supports 32-bit divided by 16-bit, 16-bit divided by 16-bit, 16-bit multiplied by 16-bit, data shift, and data normalization operations.
  - ✓ I/O port interrupt: All I/Os support interrupts, each group of I/O interrupts has an independent interrupt entry address, all I/O interrupts can support 4 types interrupt mode: high level interrupt, low level interrupt, rising edge interrupt, falling edge interrupt. (Note: The I/O port interrupts of the STC8H3K64S4 series A version of the chip cannot wake up CPU from power-down. The I/O port interrupts of the B version chip can wake up CPU from power-down, but only have one level of interrupt priority)

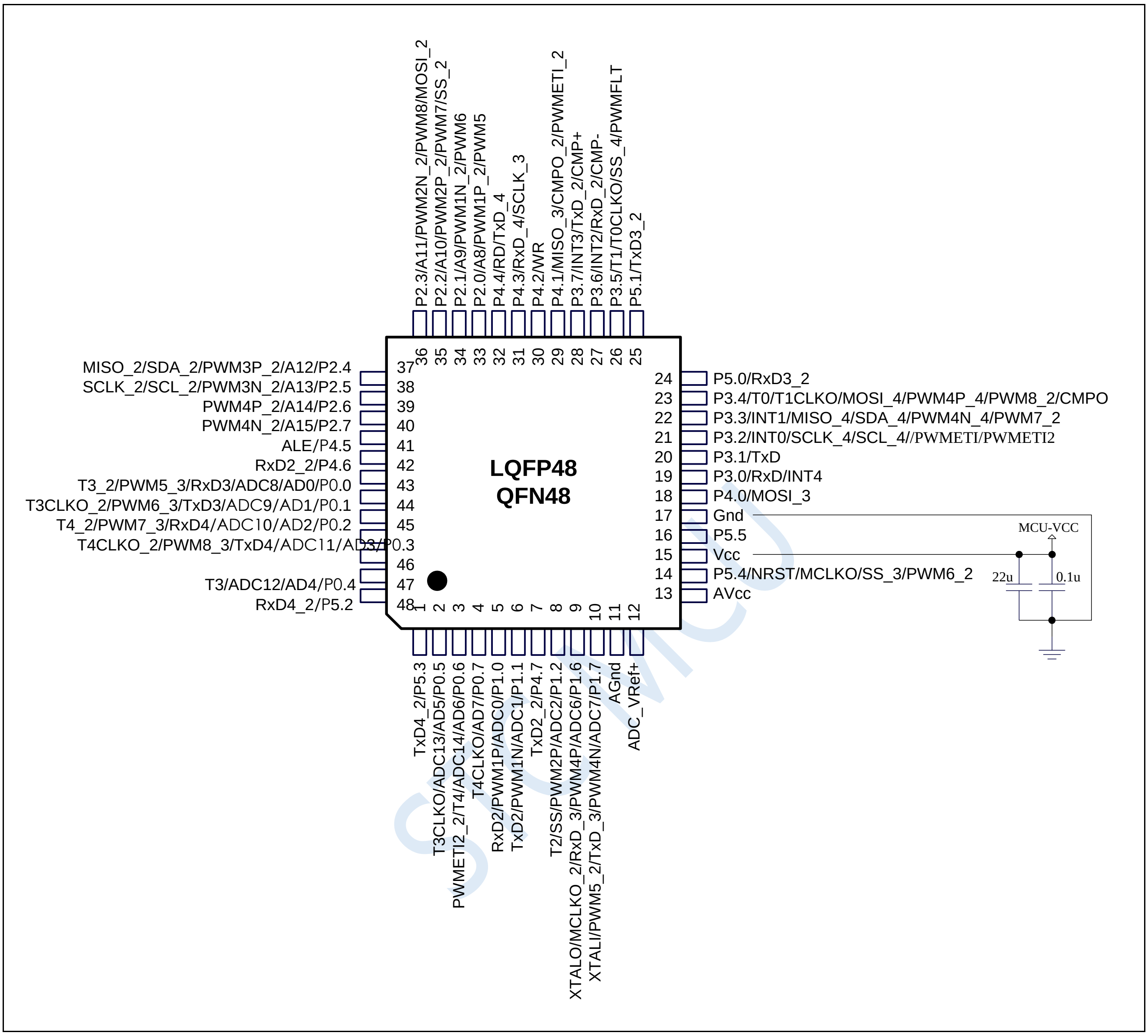
- ➢ **Analog peripherals**
  - ✓ Ultra high speed ADC which supports 12-bit precision 12 channels analog-to-digital convertors (channel 0 to channel 2, channel 6 to channel 14. No channel 3 and channel 5 because P1.3, P1.4 and P1.5 do not exist). The maximum speed can reach 800K(800K ADC conversions per second)
  - ✓ Channel 15 of ADC is used to test the internal reference voltage. (The default internal reference voltage is 1.19V when the chip is shipped)
  - ✓ A set of comparator (the CMP+ port and all ADC input ports can be selected as the positive terminal of the comparator, so the comparator can be used as a multi-channel comparator for time division multiplexing)
  - ✓ DAC: 8 channels advanced PWMs timers can be used as 8 channels DAC

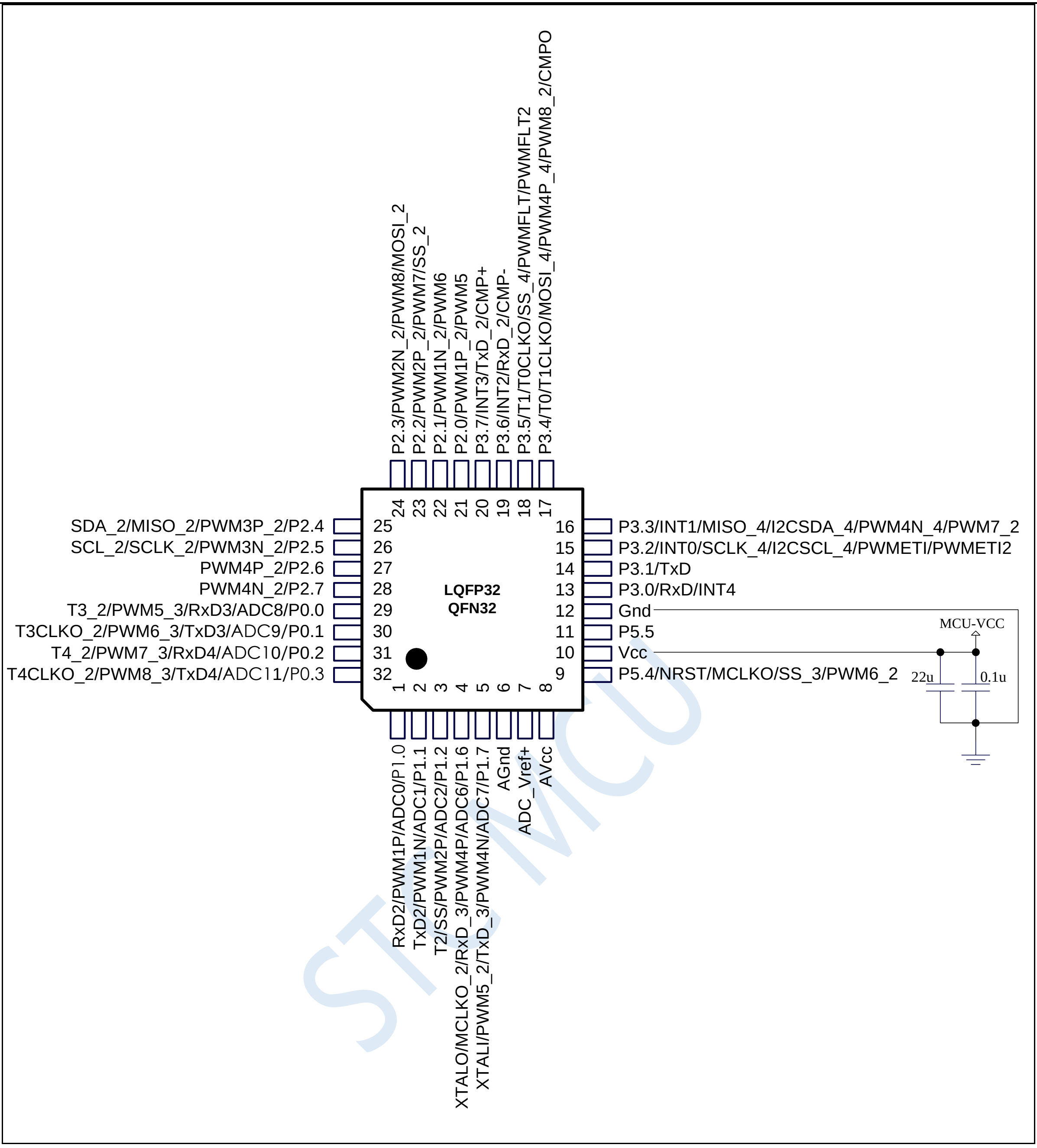
- ➢ **GPIO**
  - ✓ Up to 43 GPIOs: P0.0~P0.7, P1.0~ P1.2, P1.6~ P1.7, P2.0~P2.7, P3.0~P3.7, P4.0~P4.7, P5.0~P5.5
  - ✓ 4 modes for all GPIOs: quasi_bidirectional mode, push-pull outputmode, open drain mode, high-impedance input mode
  - ✓ Except for P3.0 and P3.1, all other I/O ports are in a high-impedance state after power-on. User must set the I/O ports mode before using them. In addition, , the internal 4K pull-up resistor of every I/O can be enabled independently.

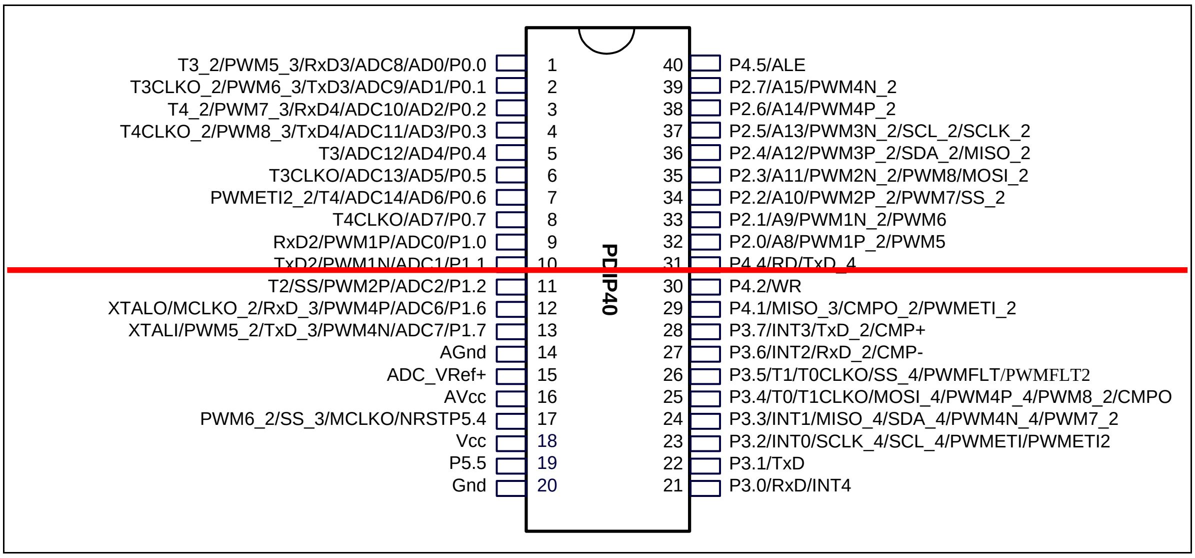
- ➢ **Package**
  - ✓ LQFP48 <9mm*9mm>, QFN48 <6mm*6mm>, LQFP32 <9mm*9mm>, QFN32 <4mm*4mm> (There are

no samples for LQFP32, QFN32 at the moment, there will be later, please order in advance if necessary.)

2.4.2 Pinouts


LQFP48
QFN48
P2.3/A11/PWM2N_2/PWM8/MOSI_2 36
P2.2/A10/PWM2P_2/PWM7/SS_2 35
P2.1/A9/PWM1N_2/PWM6 34
P2.0/A8/PWM1P_2/PWM5 33
P4.4/RD/TxD_4 32
P4.3/RxD_4/SCLK_3 31
P4.2/WR 30
P4.1/MISO_3/CMPO_2/PWMETI_2 29
P3.7/INT3/TxD_2/CMP+ 28
P3.6/INT2/RxD_2/CMP- 27
P3.5/T1/T0CLKO/SS_4/PWMFLT 26
P5.1/TxD3_2 25
MISO_2/SDA_2/PWM3P_2/A12/P2.4 37
SCLK_2/SCL_2/PWM3N_2/A13/P2.5 38
PWM4P_2/A14/P2.6 39
PWM4N_2/A15/P2.7 40
ALE/P4.5 41
RxD2_2/P4.6 42
T3_2/PWM5_3/RxD3/ADC8/AD0/P0.0 43
T3CLKO_2/PWM6_3/TxD3/ADC9/AD1/P0.1 44
T4_2/PWM7_3/RxD4/ADC10/AD2/P0.2 45
T4CLKO_2/PWM8_3/TxD4/ADC11/AD3/P0.3
46
T3/ADC12/AD4/P0.4 47
RxD4_2/P5.2 48
24 P5.0/RxD3_2
23 P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO
22 P3.3/INT1/MISO_4/SDA_4/PWM4N_4/PWM7_2
21 P3.2/INT0/SCLK_4/SCL_4//PWMETI/PWMETI2
20 P3.1/TxD
19 P3.0/RxD/INT4
18 P4.0/MOSI_3
17 Gnd
16 P5.5
15 Vcc
14 P5.4/NRST/MCLKO/SS_3/PWM6_2
13 AVcc
MCU-VCC
22u
0.1u
1 TxD4_2/P5.3
2 T3CLKO/ADC13/AD5/P0.5
3 PWMETI2_2/T4/ADC14/AD6/P0.6
4 T4CLKO/AD7/P0.7
5 RxD2/PWM1P/ADC0/P1.0
6 TxD2/PWM1N/ADC1/P1.1
7 TxD2_2/P4.7
8 T2/SS/PWM2P/ADC2/P1.2
9 XTALO/MCLKO_2/RxD_3/PWM4P/ADC6/P1.6
10 XTALI/PWM5_2/TxD_3/PWM4N/ADC7/P1.7
11 AGnd
12 ADC_VRef+


P2.3/PWM2N_2/PWM8/MOSI_2
P2.2/PWM2P_2/PWM7/SS_2
P2.1/PWM1N_2/PWM6
P2.0/PWM1P_2/PWM5
P3.7/INT3/TxD_2/CMP+
P3.6/INT2/RxD_2/CMP-
P3.5/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2
P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO
24
23
22
21
20
19
18
17
SDA_2/MISO_2/PWM3P_2/P2.4 25
SCL_2/SCLK_2/PWM3N_2/P2.5 26
PWM4P_2/P2.6 27
PWM4N_2/P2.7 28
T3_2/PWM5_3/RxD3/ADC8/P0.0 29
T3CLKO_2/PWM6_3/TxD3/ADC9/P0.1 30
T4_2/PWM7_3/RxD4/ADC10/P0.2 31
T4CLKO_2/PWM8_3/TxD4/ADC11/P0.3 32
LQFP32
QFN32
16 P3.3/INT1/MISO_4/I2CSDA_4/PWM4N_4/PWM7_2
15 P3.2/INT0/SCLK_4/I2CSCL_4/PWMETI/PWMETI2
14 P3.1/TxD
13 P3.0/RxD/INT4
12 Gnd
11 P5.5
10 Vcc
9 P5.4/NRST/MCLKO/SS_3/PWM6_2
MCU-VCC
22u
0.1u
1
2
3
4
5
6
7
8
RxD2/PWM1P/ADC0/P1.0
TxD2/PWM1N/ADC1/P1.1
T2/SS/PWM2P/ADC2/P1.2
XTALO/MCLKO_2/RxD_3/PWM4P/ADC6/P1.6
XTALI/PWM5_2/TxD_3/PWM4N/ADC7/P1.7
AGnd
ADC_Vref+
AVcc
STC MCU


T3_2/PWM5_3/RxD3/ADC8/AD0/P0.0 1
T3CLKO_2/PWM6_3/TxD3/ADC9/AD1/P0.1 2
T4_2/PWM7_3/RxD4/ADC10/AD2/P0.2 3
T4CLKO_2/PWM8_3/TxD4/ADC11/AD3/P0.3 4
T3/ADC12/AD4/P0.4 5
T3CLKO/ADC13/AD5/P0.5 6
PWMETI2_2/T4/ADC14/AD6/P0.6 7
T4CLKO/AD7/P0.7 8
RxD2/PWM1P/ADC0/P1.0 9
TxD2/PWM1N/ADC1/P1.1 10
T2/SS/PWM2P/ADC2/P1.2 11
XTALO/MCLKO_2/RxD_3/PWM4P/ADC6/P1.6 12
XTALI/PWM5_2/TxD_3/PWM4N/ADC7/P1.7 13
AGnd 14
ADC_VRef+ 15
AVcc 16
PWM6_2/SS_3/MCLKO/NRSTP5.4 17
Vcc 18
P5.5 19
Gnd 20
PDIP40
40 P4.5/ALE
39 P2.7/A15/PWM4N_2
38 P2.6/A14/PWM4P_2
37 P2.5/A13/PWM3N_2/SCL_2/SCLK_2
36 P2.4/A12/PWM3P_2/SDA_2/MISO_2
35 P2.3/A11/PWM2N_2/PWM8/MOSI_2
34 P2.2/A10/PWM2P_2/PWM7/SS_2
33 P2.1/A9/PWM1N_2/PWM6
32 P2.0/A8/PWM1P_2/PWM5
31 P4.4/RD/TxD_4
30 P4.2/WR
29 P4.1/MISO_3/CMPO_2/PWMETI_2
28 P3.7/INT3/TxD_2/CMP+
27 P3.6/INT2/RxD_2/CMP-
26 P3.5/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2
25 P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO
24 P3.3/INT1/MISO_4/SDA_4/PWM4N_4/PWM7_2
23 P3.2/INT0/SCLK_4/SCL_4/PWMETI/PWMETI2
22 P3.1/TxD
21 P3.0/RxD/INT4

Note:
1. ADC's external reference power supply pin ADC_VRef+ must not be floating. It must be connected to an external reference power supply or directly connected to Vcc.
2. If USB download is not required, P3.0/P3.1/P3.2 cannot be at low level at the same time when the chip is reset.

The download steps using ISP and notes are the same as the circumstances in 2.1.2.

2.4.3 Pin descriptions

Pin number		name	type	description
LQFP48 QFN48	LQFP32 QFN32			
1		P5.3	I/O	Standard IO port
		TxD4_2	O	Transmit pin of UART 4
2		P0.5	I/O	Standard IO port
		AD5	I	Address/data bus
		ADC13	I	ADC analog input 13
		T3CLKO	O	Clock out of timer 3
3		P0.6	I/O	Standard IO port
		AD6	I	Address/data bus
		ADC14	I	ADC analog input 14
		T4	I	Timer4 external input
		PWMFLT2_2	I	Enhance PWM external anomaly detection pin
4		P0.7	I/O	Standard IO port
		AD7	I	Address/data bus
		T4CLKO	O	Clock out of timer 4
5	1	P1.0	I/O	Standard IO port
		ADC0	I	ADC analog input 0
		PWM1P	I/O	Capture of external signal/ Pulse positive output of PWMA
		RxD2	I	Input of UART2
6	2	P1.1	I/O	Standard IO port
		ADC1	I	ADC analog input 1
		PWM1N	I/O	Capture of external signal/ Pulse negative output of PWMA
		TxD2	I	Transmit pin of UART 2
7		P4.7	I/O	Standard IO port
		TxD2_2	I	Transmit pin of UART 2
8	3	P1.2	I/O	Standard IO port
		ADC2	I	ADC analog input 2
		PWM2P	I/O	Capture of external signal/ Pulse positive output of PWM2
		SS	I	Slave selection of SPI (it is output with regard to master)
		T2	I	Timer2 external input
9	4	P1.6	I/O	Standard IO port
		ADC6	I	ADC analog input 6
		RxD_3	I	Input of UART1
		PWM4P	I/O	Capture of external signal/ Pulse positive output of PWM4
		MCLKO_2	O	Main clock output
		XTALO	O	Connect to external oscillator
10	5	P1.7	I/O	Standard IO port
		ADC7	I	ADC analog input 7
		TxD_3	O	Transmit pin of UART 1
		PWM4N	I/O	Capture of external signal/Pulse negative output of PWM4
		PWM5_2	I/O	Capture of external signal/Pulse output of PWM5
		XTALI	I	Connect to external oscillator
11	6	AGnd	Gnd	ADC Ground

Pin number		name	type	description
LQFP48 QFN48	LQFP32 QFN32			
12	7	ADC_VRef+	I	External reference input pin, which can be directly connected to MCU's VCC when the requirements are not high.
13	8	AVcc	Vcc	ADC Power Supply

14	9	P5.4	I/O	Standard IO port
		NRST	I	Reset pin(low level reset)
		MCLKO	O	Main clock output
		SS_3	I	Slave selection of SPI (it is output with regard to master)
		PWM6_2	I/O	Capture of external signal/Pulse output of PWM6
15	10	Vcc	Vcc	Power supply
16	11	P5.5	I/O	Standard IO port
17	12	Gnd	Gnd	Ground
18		P4.0	I/O	Standard IO port
		MOSI_3	I/O	Master Output/Slave Input of SPI
19	13	P3.0	I/O	Standard IO port
		RxD	I	Input of UART1
		INT4	I	External interrupt 4
20	14	P3.1	I/O	Standard IO port
		TxD	O	Transmit pin of UART 1
21	15	P3.2	I/O	Standard IO port
		INT0	I	External interrupt 0
		SCLK_4	I/O	Serial Clock of SPI
		SCL_4	I/O	Serial Clock line of I2C
		PWMETI	I	PWM external trigger input pin
		PWMET2	I	PWM external trigger input pin 2
22	16	P3.3	I/O	Standard IO port
		INT1	I	External interrupt 1
		MISO_4	I/O	Master Iutput/Slave Onput of SPI
		SDA_4	I/O	Serial data line of I2C
		PWM4N_4	I/O	Capture of external signal/Pulse negative output of PWM4
		PWM7_2	I/O	Capture of external signal/ Pulse output of PWM7
23	17	P3.4	I/O	Standard IO port
		T0	I	Timer0 external input
		T1CLKO	O	Clock out of timer 1
		MOSI_4	I/O	Master Output/Slave Input of SPI
		PWM4P_4	I/O	Capture of external signal/ Pulse positive output of PWM4
		PWM8_2	I/O	Capture of external signal/Pulse output of PWM8
		CMPO	O	Comparator output
24		P5.0	I/O	Standard IO port
		RxD3_2	I	Input of UART3
25		P5.1	I/O	Standard IO port
		TxD3_2	O	Transmit pin of UART 3
26	18	P3.5	I/O	Standard IO port
		T1	I	Timer1 external input
		T0CLKO	O	Clock out of timer 0
		SS_4	I	Slave selection of SPI (it is output with regard to master)
		PWMFLT	I	Enhance PWM external anomaly detection pin

Pin number		name	type	description
LQFP48/QFN48	LQFP32/QFN32			
27	19	P3.6	I/O	Standard IO port
		INT2	I	External interrupt 2
		RxD_2	I	Input of UART1
		CMP-	I	Comparator negative input
28	20	P3.7	I/O	Standard IO port
		INT3	I	External interrupt 3
		TxD_2	O	Transmit pin of UART 1
		CMP+	I	Comparator positive input
29		P4.1	I/O	Standard IO port

		MISO_3	I/O	Master Iutput/Slave Onput of SPI
		CMPO_2	O	Comparator output
		PWMETI_2	I	PWM external trigger input pin
30		P4.2	I/O	Standard IO port
		WR	O	Write signal of external bus
31		P4.3	I/O	Standard IO port
		RxD_4	I	Input of UART1
		SCLK_3	I/O	Serial Clock of SPI
32		P4.4	I/O	Standard IO port
		RD	O	Read signal of external bus
		TxD_4	O	Transmit pin of UART 1
33	21	P2.0	I/O	Standard IO port
		A8	I	Address bus
		PWM1P_2	I/O	Capture of external signal/ Pulse positive output of PWMA
		PWM5	I/O	Capture of external signal/Pulse output of PWM5
34	22	P2.1	I/O	Standard IO port
		A9	I	Address bus
		PWM1N_2	I/O	Capture of external signal/Pulse negative output of PWMA
		PWM6	I/O	Capture of external signal/ Pulse output of PWM6
35	23	P2.2	I/O	Standard IO port
		A10	I	Address bus
		SS_2	I	Slave selection of SPI (it is output with regard to master)
		PWM2P_2	I/O	Capture of external signal/Pulse positive output of PWMB
		PWM7	I/O	Capture of external signal/Pulse output of PWM7
36	24	P2.3	I/O	Standard IO port
		A11	I	Address bus
		MOSI_2	I/O	Master Output/Slave Input of SPI
		PWM2N_2	I/O	Capture of external signal/Pulse negative output of PWMB
		PWM8	I/O	Capture of external signal/Pulse output of PWM8

Pin number		name	type	description
LQFP48 QFN48	LQFP32/ QFN32			
37	25	P2.4	I/O	Standard IO port
		A12	I	Address bus
		MISO_2	I/O	Master Iutput/Slave Onput of SPI
		SDA_2	I/O	Serial data line of I2C
		PWM3P_2	I/O	Capture of external signal/ Pulse positive output of PWM3
38	26	P2.5	I/O	Standard IO port
		A13	I	Address bus
		SCLK_2	I/O	Serial Clock of SPI
		SCL_2	I/O	Serial Clock line of I2C
		PWM3N_2	I/O	Capture of external signal/Pulse negative output of PWM3
39	27	P2.6	I/O	Standard IO port
		A14	I	Address bus
		PWM4P_2	I/O	Capture of external signal/Pulse positive output of PWM4
40	28	P2.7	I/O	Standard IO port
		A15	I	Address bus
		PWM4N_2	I/O	Capture of external signal/Pulse negative output of PWM4
41		P4.5	I/O	Standard IO port
		ALE	O	Address Latch Enable signal
42		P4.6	I/O	Standard IO port
		RxD2_2	I	Input of UART2
43	29	P0.0	I/O	Standard IO port
		AD0	I	Address/data bus
		ADC8	I	ADC analog input 8
		RxD3	I	Input of UART3
		PWM5_3	I/O	Capture of external signal/Pulse output of PWM5
		T3_2	I	Timer3 external input
44	30	P0.1	I/O	Standard IO port
		AD1	I	Address/data bus
		ADC9	I	ADC analog input 9
		TxD3	O	Transmit pin of UART 3
		PWM6_3	I/O	Capture of external signal/ Pulse output of PWM6
		T3CLKO_2	O	Clock out of timer 3
45	31	P0.2	I/O	Standard IO port
		AD2	I	Address/data bus
		ADC10	I	ADC analog input 10
		RxD4	I	Input of UART4
		PWM7_3	I/O	Capture of external signal/ Pulse output of PWM7
		T4_2	I	Timer4 external input
46	32	P0.3	I/O	Standard IO port
		AD3	I	Address/data bus
		ADC11	I	ADC analog input 11
		TxD4	O	Transmit pin of UART 4
		PWM8_3	I/O	Capture of external signal/Pulse output of PWM8
		T4CLKO_2	O	Clock out of timer 4
47		P0.4	I/O	Standard IO port
		AD4	I	Address/data bus
		ADC12	I	ADC analog input 12
		T3	I	Timer3 external input
48		P5.2	I/O	Standard IO port
		RxD4_2	I	Input of UART4

2.5 STC8H8K64U-45I-LQFP64/48,QFN64/48 (USB family)

2.5.1 Features and Price(Quasi 16-bit MCU with 16-bit hardware multiplier and divider MDU16)

➢ **Selection and price (No external crystal and external reset required with 15 channels 12-bit ADC)**

MCU	Operating voltage（V）	Flash Code Memory (100 thousand times) (Byte)	idata Internal DATA RAM(Byte)	xdata Internal extended SRAM (Byte)	Enhanced Dual DPTR increasing or decreasing	EEPROM 100 thousand times) (Byte)	Maximum I/O Lines	Traditional I/O interrupt(INT0/INT1/INT2/INT3/INT4) (can wake-up CPU)	All I/O ports support interrupts and can wake up MCU	DMA 8080/6800 interface/ LCM driver(8-bit and 16-bit)	RTC	DMA UARTs which can wake-up CPU	Full speed USB	DMA SPI which can wake-up CPU	I^{2}C which can wake-up CPU	MDU16 (Hardware 16-bit Multiplier and Divider)	Timers/Counters (T0-T4 Pin Can wake-up CPU)	16-bit advanced PWM timer with Complementary symmetrical dead-time	Power-down Wake-up timer	DMA 15 channels high speed ADC (8 PWMs can be used as 8 DACs)	Comparator (May be used as ADC to detect external power-down)	Internal LVD interrupt (can wake-up CPU)	Watch-dog Timer	Internal high reliable reset circuit with 4 levels optional reset threshold voltage	Internal high precision Clock (adjustbal under 45MHz)	Clock output and Reset	Program encrypted transmission (Anti-blocking)	Password can be set for next update	Support RS485 download	Support hardware USB download directly and debugging	Online debug itself	Price & Package				products supply information
																																LQFP64 <12mm*12mm>	QFN64 <8mm*8mm>	LQFP48 <9mm*9mm>	QFN48 <6mm*6mm>	
STC8H8K32U	1.9-5.5	32K	256	8K	2	32K	60	Y	Y	Y	Y	4	Y	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	√	√	√		Available
STC8H8K48U	1.9-5.5	48K	256	8K	2	16K	60	Y	Y	Y	Y	4	Y	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	¥5	√	¥5		
STC8H8K60U	1.9-5.5	60K	256	8K	2	4K	60	Y	Y	Y	Y	4	Y	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	√	√	√		
STC8H8K64U	1.9-5.5	64K	256	8K	2	IAP	60	Y	Y	Y	Y	4	Y	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	¥5	√	¥5		

➢ **Core**

- ✓ Ultra-high speed 8051 Core with single clock per machine cycle, which is called 1T and the speed is about 12 times faster than traditional 8051
- ✓ Fully compatible instruction set with traditional 8051
- ✓ 22 interrupt sources and 4 interrupt priority levels
- ✓ Online debugging is supported

➢ **Operating voltage**

- ✓ 1.9V～5.5V

➢ **Operating temperature**

- ✓ -40℃~85℃(The chip is produced in -40℃～125℃ process. Please refer to the description of the electrical characteristics chapter for applications beyond the temperature range)

➢ **Flash memory**

- ✓ Up to 64Kbytes of Flash memory to be used to store user code
- ✓ Configurable EEPROM size, 512bytes single page for being erased, which can be repeatedly erased more than 100 thousand times.
- ✓ In-System-Programming, ISP in short, can be used to update the application code. No special programmer is needed.
- ✓ Online debugging with single chip is supported, and no special emulator is needed. The number of breakpoints is unlimited theoratically.

➢ **SRAM**

- ✓ 128 bytes internal direct access RAM (DATA, use keyword *data* to declare in C language program)

- ✓ 128 bytes internal indirect access RAM(IDATA, use keyword *idata* to declare in C language program)
- ✓ 8192 bytes internal extended RAM (internal XDATA, use keyword *xdata* to declare in C language program)
- ✓ 1280 bytes USB Data RAM

➢ **Clock**

- ✓ Internal high precise RC clock IRC(IRC for short, ranges from 4MHz to 45MHz), adjustable while ISP and can be divided to lower frequency by user software, 100KHz for instance.
  - ✓ Error:±0.3% (at the temperature 25℃)
  - ✓ -1.35%～+1.30% temperature drift (at the temperature range of -40℃ to +85℃)
  - ✓ -0.76%～+0.98% temperature drift (at the temperature range of -20℃ to 65℃)
- ✓ Internal 32KHz low speed IRC with large error
- ✓ External crystal (4MHz～33MHz) and external clock

  Users can freely choose the above 3 clock sources

➢ **Reset**

- ✓ Hardware reset
  - ✓ Power-on reset. Measured voltage is 1.69V~1.82V. (Effective when the chip does not enable the low voltage reset function)

    The power-on reset voltage is a voltage range consisting of an upper limit voltage and a lower limit voltage. When the operating voltage drops from 5V / 3.3V to the lower limit threshold voltage of the power-on reset, the chip is in a reset state; when the voltage rises from 0V to the upper threshold voltage of power-on reset, the chip is released from the reset state.
  - ✓ Reset by reset pin. The default function of P5.4 is the I/O port. The P5.4 pin can be set as the reset pin while ISP download. (Note: When the P5.4 pin is set as the reset pin, the reset level is low.)
  - ✓ Watch dog timer reset
  - ✓ Low voltage detection reset. 4 low voltage detection levels are provided, 1.9V, 2.3V, 2.8V, 3.0V. Each level of low-voltage detection voltage is a voltage range consisting of an upper limit voltage and a lower limit voltage. When the operating voltage drops from 5V / 3.3V to the lower limit threshold voltage of low-voltage detection, the low-voltage detection takes effect. When the voltage rises from 0V to the upper threshold voltage, the low voltage detection becomes effective.
- ✓ Software reset
  - ✓ Writing the reset trigger register using software

➢ **Interrupts**

- ✓ 22 interrupt sources: INT0(Supports rising edge and falling edge interrupt), INT1(Supports rising edge and falling edge interrupt), INT2(Supports falling edge interrupt only), INT3(Supports falling edge interrupt only), INT4(Supports falling edge interrupt only), timer 0, timer 1, timer 2, timer 3, timer 4, UART 1, UART 2, UART 3, UART 4, ADC, LVD, SPI, I^{2}C, comparator, PWMA, PWMB, USB
- ✓ 4 interrupt priority levels
- ✓ Interrupts that can wake up the CPU in clock stop mode: INT0(P3.2), INT1(P3.3), INT2(P3.6), INT3(P3.7), INT4(P3.0), T0(P3.4), T1(P3.5), T2(P1.2), T3(P0.4), T4(P0.6), RXD(P3.0/P3.6/P1.6/P4.3), RXD2(P1.0/P4.6), RXD3(P0.0/P5.0), RXD4(P0.2/P5.2), I2C_SDA(P1.4/P2.4/P3.3), Comparator interrupt, LVD interrupt, Power-down wake-up timer.

➢ **Digital peripherals**

- ✓ 5 16-bit timers: timer0, timer1, timer2, timer3, timer4, where the mode 3 of timer 0 has the Non Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
- ✓ 4 high speed UARTs: UART1, UART2, UART3, UART4, whose maximum baudrate clock may be FOSC/4
- ✓ 8 channels/2 groups of enhanced PWM, which can realize control signals with dead time, and support external fault detection function. In addition, supports 16-bit timers, 8 external interrupts, 8 external captures and pulse width measurement functions.
- ✓ SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
- ✓ I^{2}C: Master mode or slave mode are supported.
- ✓ MDU16: Hardware 16-bit Multiplier and Divider which supports 32-bit divided by 16-bit, 16-bit divided by 16-bit, 16-bit multiplied by 16-bit, data shift, and data normalization operations.
- ✓ USB: USB2.0/USB1.1 compatible with full speed USB, 6 two-way endpoints, support 4 endpoint transmission modes (control transmission, interrupt transmission, batch Quantum and synchronous transfers), each endpoint has a 64 byte buffer
- ✓ RTC: Support year, month, day, hour, minute, second, sub-second (1/128 second). And supports clock interrupt and a set of alarm clocks (Note: A version of the chip does not have this function)
- ✓ I/O port interrupt: All I/Os support interrupts, each group of I/O interrupts has an independent interrupt entry address, all I/O interrupts can support 4 types interrupt mode: high level interrupt, low level interrupt, rising edge interrupt, falling edge interrupt. Provides 4 levels of interrupt priority and supports power-down wake-up function. (Note: A version of the chip does not have this function)
- ✓ DMA: support Memory-To-Memory, SPI, UART1TX/UART1RX, UART2TX/UART2RX, UART3TX/UART3RX, UART4TX/UART4RX, ADC(Automatically calculates the average of multiple ADC results), LCM (Note: A version of the chip does not have this function)
- ✓ LCM (TFT color screen) dirver: support 8080 and 6800 interface, and support 8-bit and 16-bit data width (Note: A version of the chip does not have this function)
  - ✓ 8 bits 8080 data bus: 8 bits data lines (TD0~TD7), READ signael (TRD)c WRITE signal (TWR), RS line (TRS)
  - ✓ 16 bits 8080 bus: 16 bits data lines (TD0~TD15), READ signael (TRD)c WRITE signal (TWR), RS line (TRS)
  - ✓ 8 bits 6800 bus: 8 bits data lines (TD0~TD7), enable signal (TE) , READ and WRITE signal (TRW) , RS line (TRS)
  - ✓ 16 bits 6800 bus: 16 bits data lines (TD0~TD15), enable signal (TE) , READ and WRITE signal (TRW) , RS line (TRS)
  - ✓ Note: If you use 8-bit data lines to control the TFT screen, you generally need TD0~D7, TRD/TWR/TRS, 11 data and

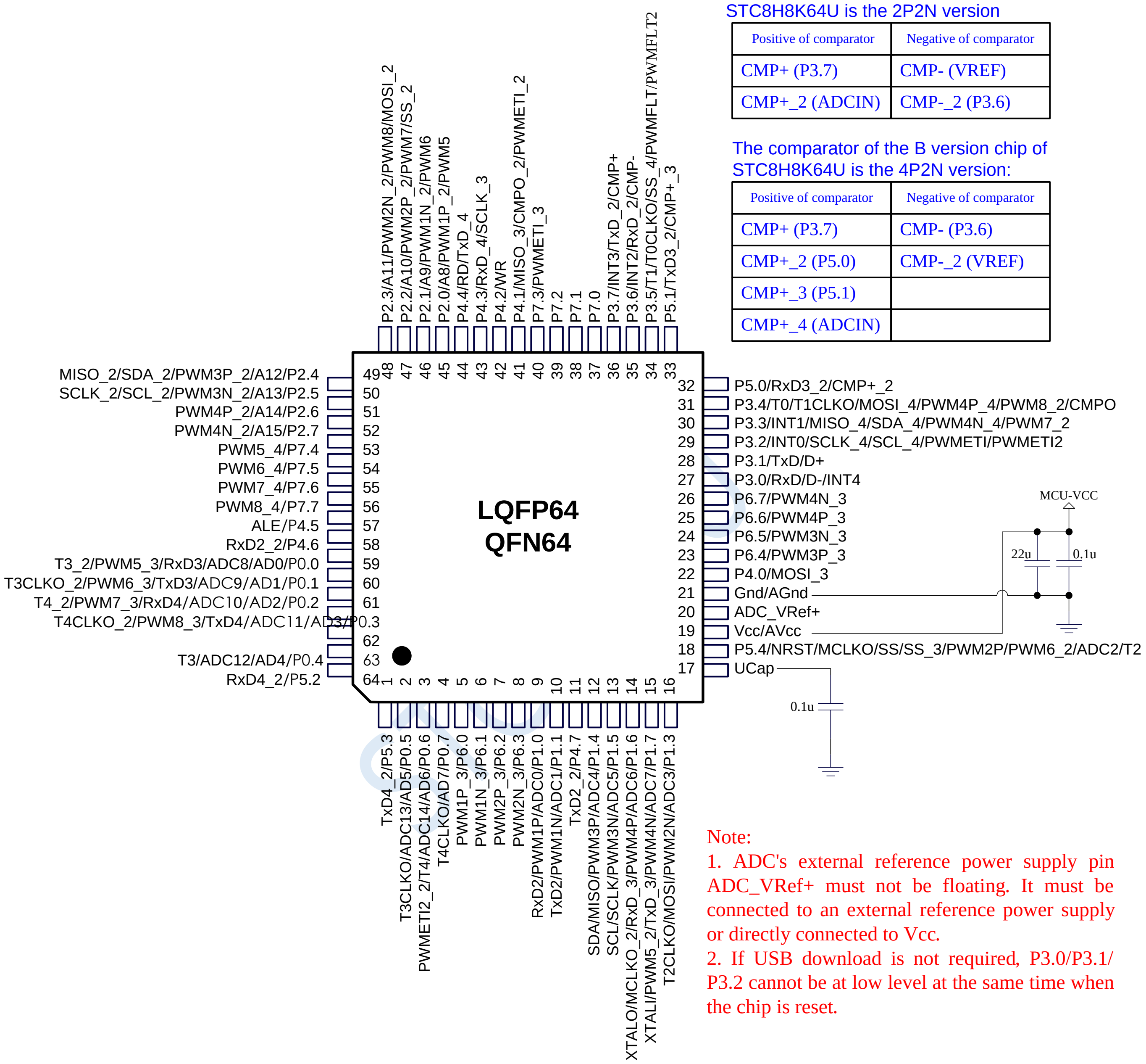
control lines, plus 2 common I/Os to control chip selection and reset (many TFT color screen chip selections and reset manufacturer has carried out automatic processing, does not need software control)

- **Analog peripherals**
  - ✓ Ultra high speed ADC which supports 12-bit precision 15 channels (channel 0 to channel 14) analog-to-digital conversion. The maximum speed can be 800K(800K ADC conversions per second)
  - ✓ ADC channel 15 is used to test the internal reference voltage. (The default internal reference voltage is 1.19V when the chip is shipped)
  - ✓ Comparator. A set of comparator (For A version of the chip, the CMP+ port and all ADC input ports can be selected as the positive terminal of the comparator, the CMP- port and the internal reference voltage 1.19V can be selected as the negative terminal of the comparator. For B version of the chip, the CMP+, CMP+_2, CMP+_3 port and all ADC input ports can be selected as the positive terminal of the comparator, the CMP- port and the internal reference voltage 1.19V can be selected as the negative terminal of the comparator. So the comparator can be used as a multi-channel comparator for time division multiplexing)
  - ✓ DAC: 8 channels advanced PWM timer can be used as 8 channels DAC

- **GPIO**
  - ✓ Up to 61 GPIOs: P0.0~P0.7, P1.0~ P1.7(no P1.2), P2.0~P2.7, P3.0~P3.7, P4.0~P4.7, P5.0~P5.4, P6.0~P6.7, P7.0~P7.7
  - ✓ 4 modes for all GPIOs: quasi_bidirectional mode, push-pull outputmode, open drain mode, high-impedance input mode
  - ✓ Except for P3.0 and P3.1, all other I/O ports are in a high-impedance state after power-on. User must set the I/O ports mode before using them. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.

- **Package**
  - ✓ LQFP64 <12mm*12mm>, QFN64 <8mm*8mm>, LQFP48 <9mm*9mm>, QFN48 <6mm*6mm>, LQFP32<9mm*9mm> (Not available not), TSSOP20 (Not available not)

2.5.2 Pinouts



The comparator of the A version chip of STC8H8K64U is the 2P2N version

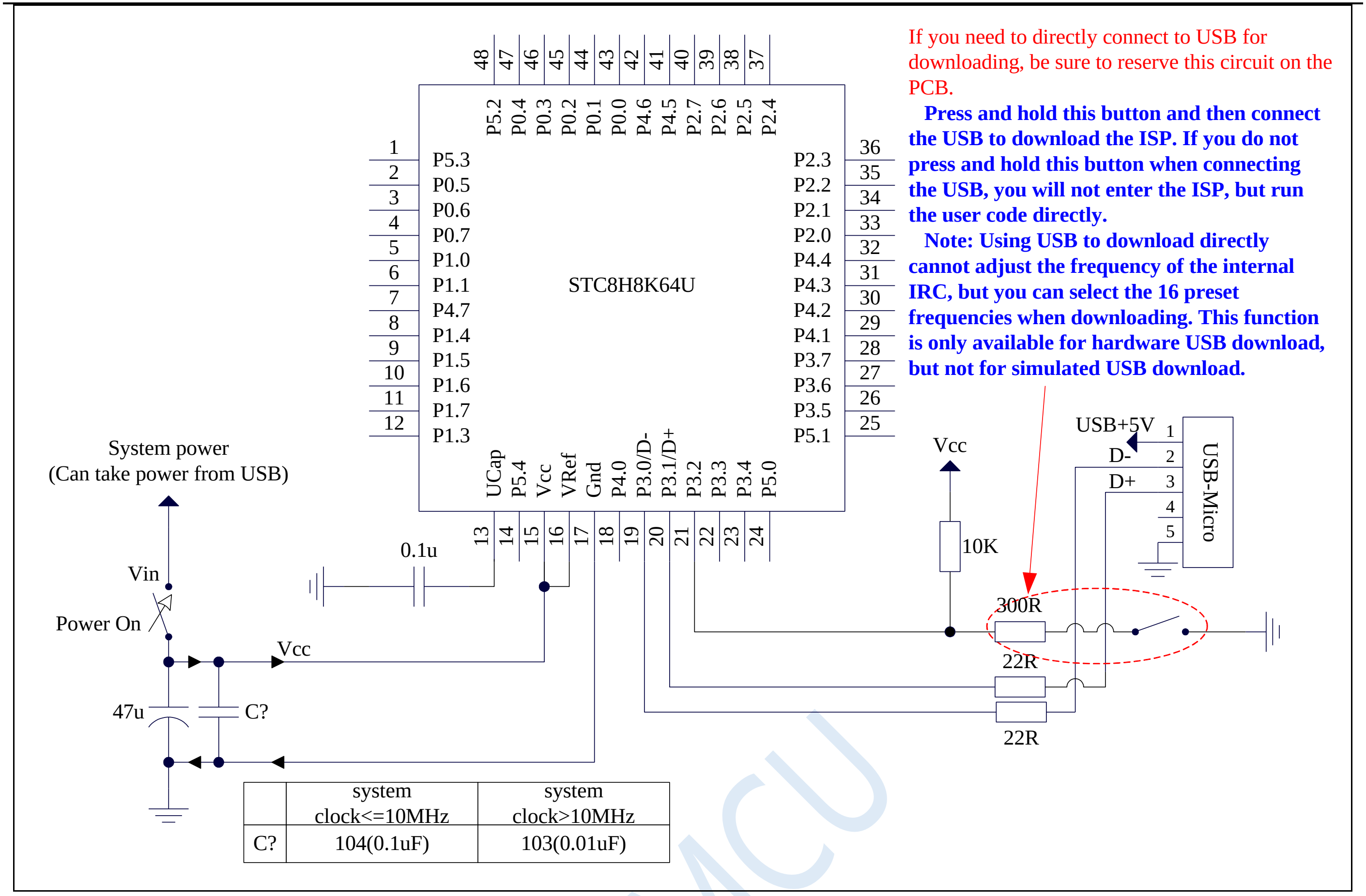
Positive of comparator	Negative of comparator
CMP+ (P3.7)	CMP- (VREF)
CMP+_2 (ADCIN)	CMP-_2 (P3.6)

The comparator of the B version chip of STC8H8K64U is the 4P2N version:

Positive of comparator	Negative of comparator
CMP+ (P3.7)	CMP- (P3.6)
CMP+_2 (P5.0)	CMP-_2 (VREF)
CMP+_3 (P5.1)	
CMP+_4 (ADCIN)	

Note:

1. ADC's external reference power supply pin ADC_VRef+ must not be floating. It must be connected to an external reference power supply or directly connected to Vcc.
2. If USB download is not required, P3.0/P3.1/ P3.2 cannot be at low level at the same time when the chip is reset.



If you need to directly connect to USB for downloading, be sure to reserve this circuit on the PCB.

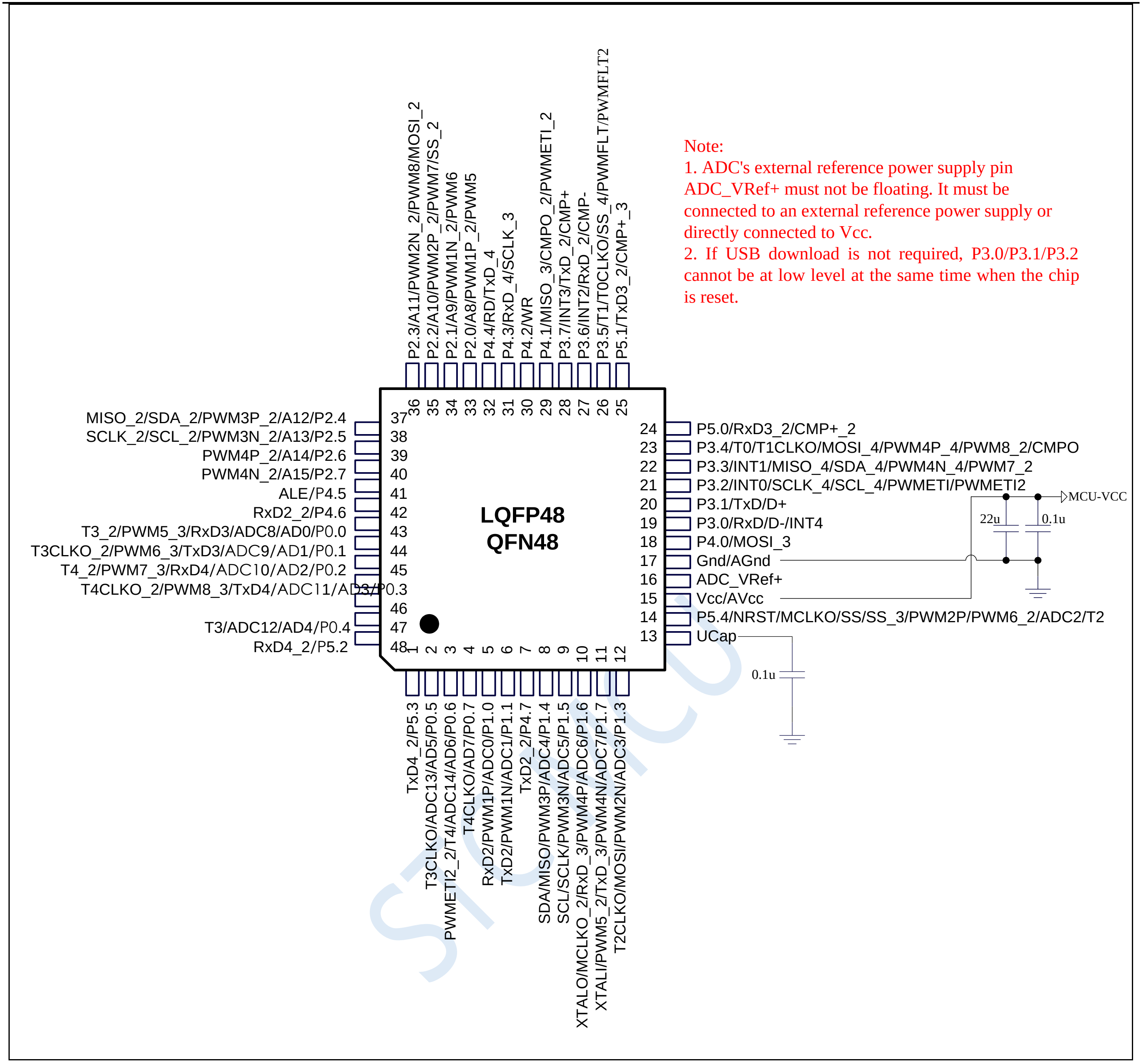
Press and hold this button and then connect the USB to download the ISP. If you do not press and hold this button when connecting the USB, you will not enter the ISP, but run the user code directly.

Note: Using USB to download directly cannot adjust the frequency of the internal IRC, but you can select the 16 preset frequencies when downloading. This function is only available for hardware USB download, but not for simulated USB download.

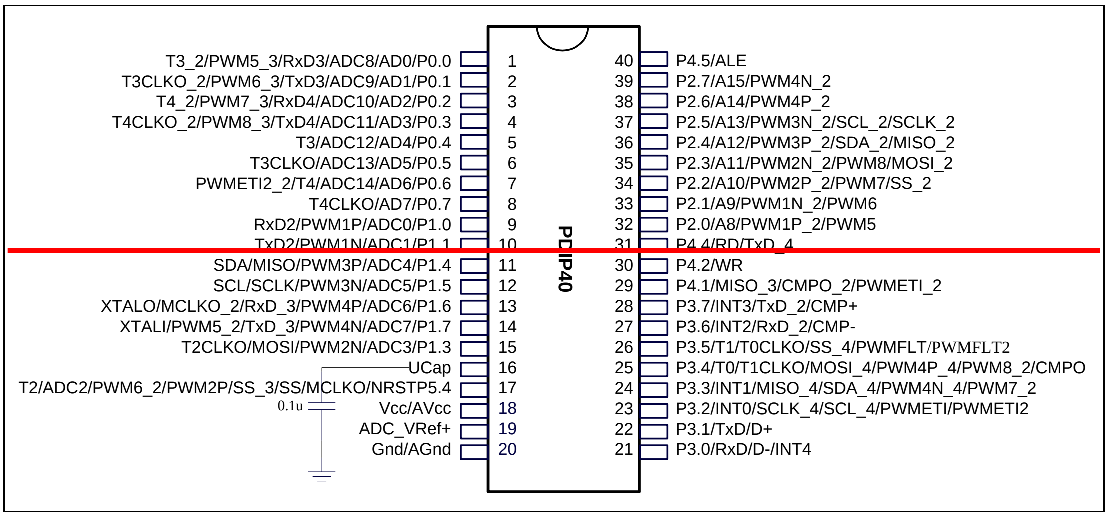
	system clock<=10MHz	system clock>10MHz
C?	104(0.1uF)	103(0.01uF)

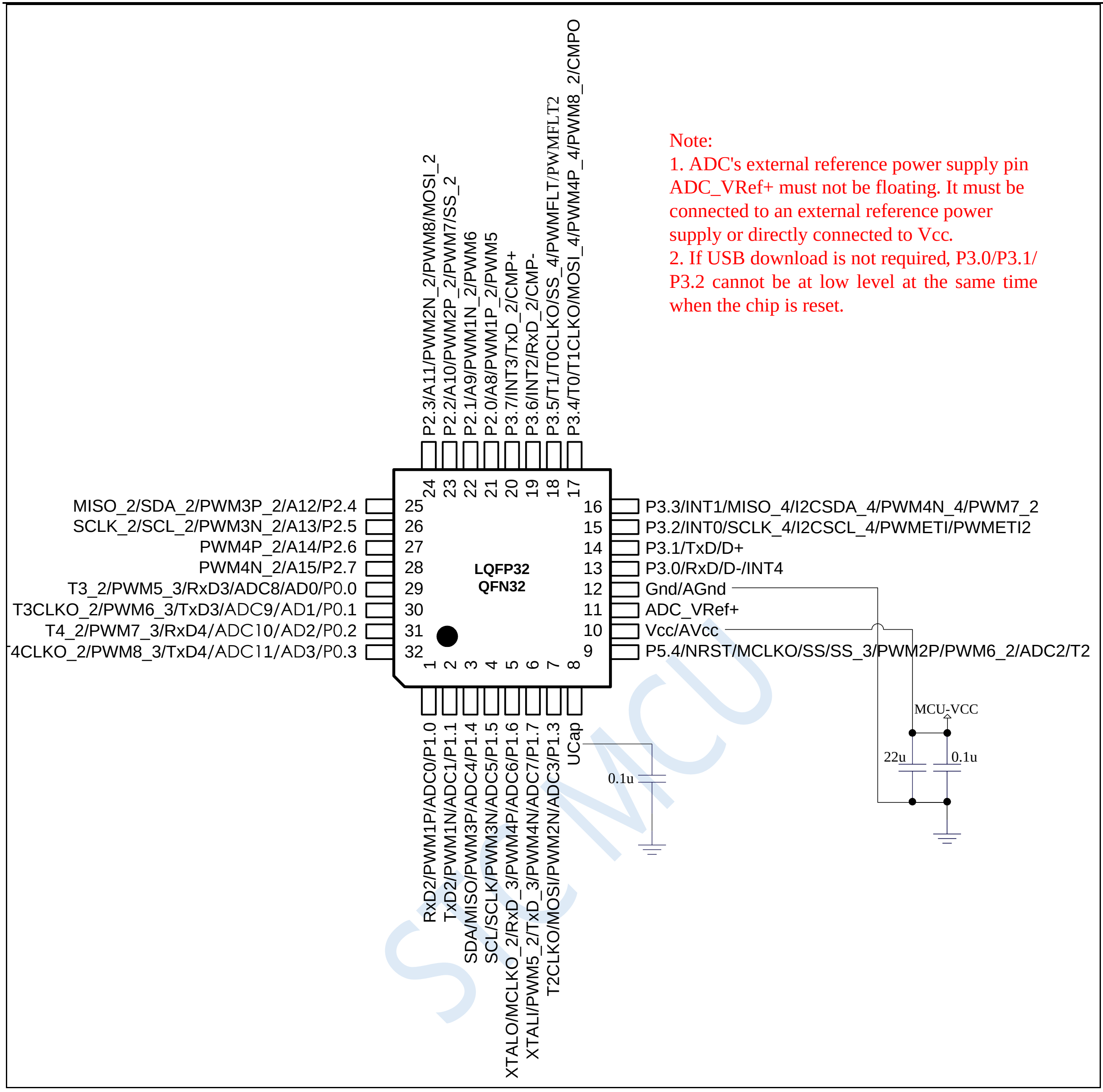
The download steps using ISP and notes are the same as the circumstances in 2.1.2.

When the user uses hardware USB to download code to STC8H8K64U series through ISP, the internal IRC frequency cannot be adjusted, but the user can choose 16 internal preset frequencies (respectively 5.5296M, 6M, 11.0592M, 12M, 18.432M, 20M, 22.1184m, 24M, 27M, 30M, 33.1776m, 35M, 36.864m, 40M, 44.2368m and 48M). The user can only select one of the frequencies from the drop-down list, and cannot manually enter other frequencies. (If you use serial port to download, you can input any frequency between 4M and 48M).

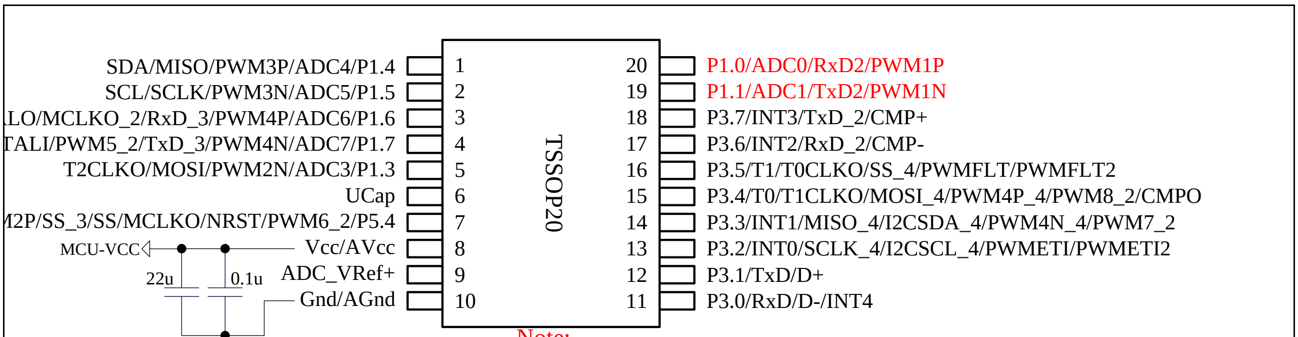

LQFP48
QFN48
P2.3/A11/PWM2N_2/PWM8/MOSI_2 36
P2.2/A10/PWM2P_2/PWM7/SS_2 35
P2.1/A9/PWM1N_2/PWM6 34
P2.0/A8/PWM1P_2/PWM5 33
P4.4/RD/TxD_4 32
P4.3/RxD_4/SCLK_3 31
P4.2/WR 30
P4.1/MISO_3/CMPO_2/PWMETI_2 29
P3.7/INT3/TxD_2/CMP+ 28
P3.6/INT2/RxD_2/CMP- 27
P3.5/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2 26
P5.1/TxD3_2/CMP+_3 25
MISO_2/SDA_2/PWM3P_2/A12/P2.4 37
SCLK_2/SCL_2/PWM3N_2/A13/P2.5 38
PWM4P_2/A14/P2.6 39
PWM4N_2/A15/P2.7 40
ALE/P4.5 41
RxD2_2/P4.6 42
T3_2/PWM5_3/RxD3/ADC8/AD0/P0.0 43
T3CLKO_2/PWM6_3/TxD3/ADC9/AD1/P0.1 44
T4_2/PWM7_3/RxD4/ADC10/AD2/P0.2 45
T4CLKO_2/PWM8_3/TxD4/ADC11/AD3/P0.3 46
T3/ADC12/AD4/P0.4 47
RxD4_2/P5.2 48
1 TxD4_2/P5.3
2 T3CLKO/ADC13/AD5/P0.5
3 PWMETI2_2/T4/ADC14/AD6/P0.6
4 T4CLKO/AD7/P0.7
5 RxD2/PWM1P/ADC0/P1.0
6 TxD2/PWM1N/ADC1/P1.1
7 TxD2_2/P4.7
8 SDA/MISO/PWM3P/ADC4/P1.4
9 SCL/SCLK/PWM3N/ADC5/P1.5
10 XTALO/MCLKO_2/RxD_3/PWM4P/ADC6/P1.6
11 XTALI/PWM5_2/TxD_3/PWM4N/ADC7/P1.7
12 T2CLKO/MOSI/PWM2N/ADC3/P1.3
24 P5.0/RxD3_2/CMP+_2
23 P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO
22 P3.3/INT1/MISO_4/SDA_4/PWM4N_4/PWM7_2
21 P3.2/INT0/SCLK_4/SCL_4/PWMETI/PWMETI2
20 P3.1/TxD/D+
19 P3.0/RxD/D-/INT4
18 P4.0/MOSI_3
17 Gnd/AGnd
16 ADC_VRef+
15 Vcc/AVcc
14 P5.4/NRST/MCLKO/SS/SS_3/PWM2P/PWM6_2/ADC2/T2
13 UCap
MCU-VCC
22u
0.1u
0.1u

Note:
1. ADC's external reference power supply pin ADC_VRef+ must not be floating. It must be connected to an external reference power supply or directly connected to Vcc.
2. If USB download is not required, P3.0/P3.1/P3.2 cannot be at low level at the same time when the chip is reset.


PDIP40
T3_2/PWM5_3/RxD3/ADC8/AD0/P0.0 1
T3CLKO_2/PWM6_3/TxD3/ADC9/AD1/P0.1 2
T4_2/PWM7_3/RxD4/ADC10/AD2/P0.2 3
T4CLKO_2/PWM8_3/TxD4/ADC11/AD3/P0.3 4
T3/ADC12/AD4/P0.4 5
T3CLKO/ADC13/AD5/P0.5 6
PWMETI2_2/T4/ADC14/AD6/P0.6 7
T4CLKO/AD7/P0.7 8
RxD2/PWM1P/ADC0/P1.0 9
TxD2/PWM1N/ADC1/P1.1 10
SDA/MISO/PWM3P/ADC4/P1.4 11
SCL/SCLK/PWM3N/ADC5/P1.5 12
XTALO/MCLKO_2/RxD_3/PWM4P/ADC6/P1.6 13
XTALI/PWM5_2/TxD_3/PWM4N/ADC7/P1.7 14
T2CLKO/MOSI/PWM2N/ADC3/P1.3 15
UCap 16
T2/ADC2/PWM6_2/PWM2P/SS_3/SS/MCLKO/NRSTP5.4 17
Vcc/AVcc 18
ADC_VRef+ 19
Gnd/AGnd 20
0.1u
40 P4.5/ALE
39 P2.7/A15/PWM4N_2
38 P2.6/A14/PWM4P_2
37 P2.5/A13/PWM3N_2/SCL_2/SCLK_2
36 P2.4/A12/PWM3P_2/SDA_2/MISO_2
35 P2.3/A11/PWM2N_2/PWM8/MOSI_2
34 P2.2/A10/PWM2P_2/PWM7/SS_2
33 P2.1/A9/PWM1N_2/PWM6
32 P2.0/A8/PWM1P_2/PWM5
31 P4.4/RD/TxD_4
30 P4.2/WR
29 P4.1/MISO_3/CMPO_2/PWMETI_2
28 P3.7/INT3/TxD_2/CMP+
27 P3.6/INT2/RxD_2/CMP-
26 P3.5/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2
25 P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO
24 P3.3/INT1/MISO_4/SDA_4/PWM4N_4/PWM7_2
23 P3.2/INT0/SCLK_4/SCL_4/PWMETI/PWMETI2
22 P3.1/TxD/D+
21 P3.0/RxD/D-/INT4


LQFP32
QFN32
P2.3/A11/PWM2N_2/PWM8/MOSI_2
P2.2/A10/PWM2P_2/PWM7/SS_2
P2.1/A9/PWM1N_2/PWM6
P2.0/A8/PWM1P_2/PWM5
P3.7/INT3/TxD_2/CMP+
P3.6/INT2/RxD_2/CMP-
P3.5/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2
P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO
24
23
22
21
20
19
18
17
MISO_2/SDA_2/PWM3P_2/A12/P2.4 25
SCLK_2/SCL_2/PWM3N_2/A13/P2.5 26
PWM4P_2/A14/P2.6 27
PWM4N_2/A15/P2.7 28
T3_2/PWM5_3/RxD3/ADC8/AD0/P0.0 29
T3CLKO_2/PWM6_3/TxD3/ADC9/AD1/P0.1 30
T4_2/PWM7_3/RxD4/ADC10/AD2/P0.2 31
T4CLKO_2/PWM8_3/TxD4/ADC11/AD3/P0.3 32
16 P3.3/INT1/MISO_4/I2CSDA_4/PWM4N_4/PWM7_2
15 P3.2/INT0/SCLK_4/I2CSCL_4/PWMETI/PWMETI2
14 P3.1/TxD/D+
13 P3.0/RxD/D-/INT4
12 Gnd/AGnd
11 ADC_VRef+
10 Vcc/AVcc
9 P5.4/NRST/MCLKO/SS/SS_3/PWM2P/PWM6_2/ADC2/T2
1
2
3
4
5
6
7
8
RxD2/PWM1P/ADC0/P1.0
TxD2/PWM1N/ADC1/P1.1
SDA/MISO/PWM3P/ADC4/P1.4
SCL/SCLK/PWM3N/ADC5/P1.5
XTALO/MCLKO_2/RxD_2/PWM4P/ADC6/P1.6
XTALI/PWM5_2/TxD_3/PWM4N/ADC7/P1.7
T2CLKO/MOSI/PWM2N/ADC3/P1.3
UCap
0.1u
MCU-VCC
22u
0.1u
STC MCU

Note:
1. ADC's external reference power supply pin ADC_VRef+ must not be floating. It must be connected to an external reference power supply or directly connected to Vcc.
2. If USB download is not required, P3.0/P3.1/P3.2 cannot be at low level at the same time when the chip is reset.


TSSOP20
SDA/MISO/PWM3P/ADC4/P1.4 1
SCL/SCLK/PWM3N/ADC5/P1.5 2
LO/MCLKO_2/RxD_3/PWM4P/ADC6/P1.6 3
TALI/PWM5_2/TxD_3/PWM4N/ADC7/P1.7 4
T2CLKO/MOSI/PWM2N/ADC3/P1.3 5
UCap 6
I2P/SS_3/SS/MCLKO/NRST/PWM6_2/P5.4 7
Vcc/AVcc 8
ADC_VRef+ 9
Gnd/AGnd 10
20 P1.0/ADC0/RxD2/PWM1P
19 P1.1/ADC1/TxD2/PWM1N
18 P3.7/INT3/TxD_2/CMP+
17 P3.6/INT2/RxD_2/CMP-
16 P3.5/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2
15 P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO
14 P3.3/INT1/MISO_4/I2CSDA_4/PWM4N_4/PWM7_2
13 P3.2/INT0/SCLK_4/I2CSCL_4/PWMETI/PWMETI2
12 P3.1/TxD/D+
11 P3.0/RxD/D-/INT4
MCU-VCC
22u
0.1u

Note:
1. ADC's external reference power supply pin ADC_VRef+ must not be floating. It must be connected to an external reference power supply or directly connected to Vcc.
2. If USB download is not required, P3.0/P3.1/P3.2 cannot be at low level at the same time when the chip is reset.

2.5.3 Pin descriptions

Pin number		name	type	description
LQFP64/QFN64	LQFP48/QFN48			
1	1	P5.3	I/O	Standard IO port
		TxD4_2	O	Transmit pin of UART 4
2	2	P0.5	I/O	Standard IO port
		AD5	I	Address/data bus
		ADC13	I	ADC analog input 13
		T3CLKO	O	Clock out of timer 3
3	3	P0.6	I/O	Standard IO port
		AD6	I	Address/data bus
		ADC14	I	ADC analog input 14
		T4	I	Timer4 external input
		PWMFLT2_2	I	Enhance PWM external anomaly detection pin 2
4	4	P0.7	I/O	Standard IO port
		AD7	I	Address/data bus
		T4CLKO	O	Clock out of timer 4
5		P6.0	I/O	Standard IO port
		PWM1P_3	I/O	Capture of external signal/Positive of PWMA pulse output
6		P6.1	I/O	Standard IO port
		PWM1N_3	I/O	Capture of external signal/Negative of PWMA pulse output
7		P6.2	I/O	Standard IO port
		PWM2P_3	I/O	Capture of external signal/Positive of PWMB pulse output
8		P6.3	I/O	Standard IO port
		PWM2N_3	I/O	Capture of external signal/Negative of PWMB pulse output
9	5	P1.0	I/O	Standard IO port
		ADC0	I	ADC analog input 0
		PWM1P	I/O	Capture of external signal/Positive of PWMA pulse output
		RxD2	I	Input of UART2
10	6	P1.1	I/O	Standard IO port
		ADC1	I	ADC analog input 1
		PWM1N	I/O	Capture of external signal/Negative of PWMA pulse output
		TxD2	I	Transmit pin of UART 2
11	7	P4.7	I/O	Standard IO port
		TxD2_2	I	Transmit pin of UART 2
12	8	P1.4	I/O	Standard IO port
		ADC4	I	ADC analog input 4
		PWM3P	I/O	Capture of external signal/Positive of PWM3 pulse output
		MISO	I/O	Master Iutput/Slave Onput of SPI
		SDA	I/O	Serial data line of I2C

Pin number		name	type	description
LQFP64/QFN64	LQFP48/QFN48			
13	9	P1.5	I/O	Standard IO port
		ADC5	I	ADC analog input 5
		PWM3N	I/O	Capture of external signal/Negative of PWM3 pulse output
		SCLK	I/O	Serial Clock of SPI
		SCL	I/O	Serial Clock line of I2C
14	10	P1.6	I/O	Standard IO port
		ADC6	I	ADC analog input 6
		RxD_3	I	Input of UART1
		PWM4P	I/O	Capture of external signal/Positive of PWM4 pulse output
		MCLKO_2	O	Main clock output
		XTALO	O	Connect to external oscillator
15	11	P1.7	I/O	Standard IO port
		ADC7	I	ADC analog input 7
		TxD_3	O	Transmit pin of UART 1
		PWM4N	I/O	Capture of external signal/Negative of PWM4 pulse output
		PWM5_2	I/O	Capture of external signal/Pulse output of PWM5
		XTALI	I	Connect to external oscillator
16	12	P1.3	I/O	Standard IO port
		ADC3	I	ADC analog input 3
		MOSI	I/O	Master Output/Slave Input of SPI
		PWM2N	I/O	Capture of external signal/Negative of PWM2 pulse output
		T2CLKO	O	Clock out of timer 2
17	13	UCAP	I	USB core power stabilizer
18	14	P5.4	I/O	Standard IO port
		NRST	I	Reset pin
		MCLKO	O	Main clock output
		SS_3	I	Slave selection of SPI (it is output with regard to master)
		SS	I	Slave selection of SPI (it is output with regard to master)
		PWM2P	I/O	Capture of external signal/Positive of PWM2 pulse output
		PWM6_2	I/O	Capture of external signal/Pulse output of PWM6
		T2	I	Timer2 external input
		ADC2	I	ADC analog input 2
19	15	Vcc	Vcc	Power Supply
		AVcc	Vcc	ADC Power Supply
20	16	Vref+	I	ADC external reference voltage input pin, which can be directly connected to MCU VCC when the requirements are not high.
21	17	Gnd	Gnd	Ground
		AGnd	Gnd	ADC Ground
22	18	P4.0	I/O	Standard IO port
		MOSI_3	I/O	Master Output/Slave Input of SPI
23		P6.4	I/O	Standard IO port
		PWM3P_3	I/O	Capture of external signal/Positive of PWM3 pulse output

Pin number	name	type	description

LQFP64/QFN64	LQFP48/QFN48			
24		P6.5	I/O	Standard IO port
		PWM3N_3	I/O	Capture of external signal/Negative of PWM3 pulse output
25		P6.6	I/O	Standard IO port
		PWM4P_3	I/O	Capture of external signal/Positive of PWM4 pulse output
26		P6.7	I/O	Standard IO port
		PWM4N_3	I/O	Capture of external signal/Negative of PWM4 pulse output
27	19	P3.0	I/O	Standard IO port
		D-	I/O	USB data port
		RxD	I	Input of UART1
		INT4	I	External interrupt 4
28	20	P3.1	I/O	Standard IO port
		D+	I/O	USB data port
		TxD	O	Transmit pin of UART 1
29	21	P3.2	I/O	Standard IO port
		INT0	I	External interrupt 0
		SCLK_4	I/O	Serial Clock of SPI
		SCL_4	I/O	Serial Clock line of I2C
		PWMETI	I	PWM external trigger input pin
		PWMETI2	I	PWM external trigger input pin 2
30	22	P3.3	I/O	Standard IO port
		INT1	I	External interrupt 1
		MISO_4	I/O	Master Iutput/Slave Onput of SPI
		SDA_4	I/O	Serial data line of I2C
		PWM4N_4	I/O	Capture of external signal/Negative of PWM4 pulse output
		PWM7_2	I/O	Capture of external signal/Pulse output of PWM7
31	23	P3.4	I/O	Standard IO port
		T0	I	Timer0 external input
		T1CLKO	O	Clock out of timer 1
		MOSI_4	I/O	Master Output/Slave Input of SPI
		PWM4P_4	I/O	Capture of external signal/Positive of PWM4 pulse output
		PWM8_2	I/O	Capture of external signal/Pulse output of PWM8
		CMPO	O	Output of comparator
32	24	P5.0	I/O	Standard IO port
		RxD3_2	I	Input of UART 3
		CMP+_2	I	Positive input of comparator
33	25	P5.1	I/O	Standard IO port
		TxD3_2	O	Transmit pin of UART 3
		CMP+_3	I	Positive input of comparator
34	26	P3.5	I/O	Standard IO port
		T1	I	Timer1 external input
		T0CLKO	O	Clock out of timer 0
		SS_4	I	Slave selection of SPI (it is output with regard to master)
		PWMFLT1	I	Enhance PWM external anomaly detection pin
		PWMFLT2	I	Enhance PWM external anomaly detection pin

Pin number		name	type	description
LQFP64/QFN64	LQFP48/QFN48			

35	27	P3.6	I/O	Standard IO port
		INT2	I	External interrupt 2
		RxD_2	I	Input of UART1
		CMP-	I	Negative input of comparator
36	28	P3.7	I/O	Standard IO port
		INT3	I	External interrupt 3
		TxD_2	O	Transmit pin of UART 1
		CMP+	I	Positive input of comparator
37		P7.0	I/O	Standard IO port
38		P7.1	I/O	Standard IO port
39		P7.2	I/O	Standard IO port
40		P7.3	I/O	Standard IO port
		PWMETI_3	I	PWM external trigger input pin
41	29	P4.1	I/O	Standard IO port
		MISO_3	I/O	Master Iutput/Slave Onput of SPI
		CMPO_2	O	Output of comparator
		PWMETI_3	I	PWM external trigger input pin
42	30	P4.2	I/O	Standard IO port
		WR	O	WRITE signal of external bus
43	31	P4.3	I/O	Standard IO port
		RxD_4	I	Input of UART1
		SCLK_3	I/O	Serial Clock of SPI
44	32	P4.4	I/O	Standard IO port
		RD	O	READ signal of external bus
		TxD_4	O	Transmit pin of UART 1
45	33	P2.0	I/O	Standard IO port
		A8	I	Address bus
		PWM1P_2	I/O	Capture of external signal/Pulse positive output of PWMA
		PWM5	I/O	Capture of external signal/Pulse output of PWM5
46	34	P2.1	I/O	Standard IO port
		A9	I	Address bus
		PWM1N_2	I/O	Capture of external signal/Pulse negative output of PWMA
		PWM6	I/O	Capture of external signal/Pulse output of PWM6
47	35	P2.2	I/O	Standard IO port
		A10	I	Address bus
		SS_2	I	Slave selection of SPI (it is output with regard to master)
		PWM2P_2	I/O	Capture of external signal/Pulse positive output of PWMB
		PWM7	I/O	Capture of external signal/Pulse output of PWM7
48	36	P2.3	I/O	Standard IO port
		A11	I	Address bus
		MOSI_2	I/O	Master Output/Slave Input of SPI
		PWM2N_2	I/O	Capture of external signal/Pulse negative output of PWMB
		PWM8	I/O	Capture of external signal/Pulse output of PWM8

Pin number		name	type	description
LQFP64/QFN64	LQFP48/QFN48			
49	37	P2.4	I/O	Standard IO port
		A12	I	Address bus
		MISO_2	I/O	Master Iutput/Slave Onput of SPI
		SDA_2	I/O	Serial data line of I2C
		PWM3P_2	I/O	Capture of external signal/Pulse positive output of PWM3
50	38	P2.5	I/O	Standard IO port

		A13	I	Address bus
		SCLK_2	I/O	Serial Clock of SPI
		SCL_2	I/O	Serial Clock line of I2C
		PWM3N_2	I/O	Capture of external signal/Pulse negative output of PWM3
51	39	P2.6	I/O	Standard IO port
		A14	I	Address bus
		PWM4P_2	I/O	Capture of external signal/Pulse positive output of PWM4
52	40	P2.7	I/O	Standard IO port
		A15	I	Address bus
		PWM4N_2	I/O	Capture of external signal/Pulse negative output of PWM4
53		P7.4	I/O	Standard IO port
		PWM5_4	I/O	Capture of external signal/Pulse output of PWM5
54		P7.5	I/O	Standard IO port
		PWM6_4	I/O	Capture of external signal/Pulse output of PWM6
55		P7.6	I/O	Standard IO port
		PWM7_4	I/O	Capture of external signal/Pulse output of PWM7
56		P7.7	I/O	Standard IO port
		PWM8_4	I/O	Capture of external signal/Pulse output of PWM8
57	41	P4.5	I/O	Standard IO port
		ALE	O	Address Latch Enable signal
58	42	P4.6	I/O	Standard IO port
		RxD2_2	I	Input of UART2
59	43	P0.0	I/O	Standard IO port
		AD0	I	Address/data bus
		ADC8	I	ADC analog input 8
		RxD3		Input of UART 3
		PWM5_3	I/O	Capture of external signal/Pulse output of PWM5
		T3_2	I/O	Timer3 external input
60	44	P0.1	I/O	Standard IO port
		AD1	I	Address/data bus
		ADC9	I	ADC analog input 9
		TxD3	O	Transmit pin of UART 3
		PWM6_3	I/O	Capture of external signal/Pulse output of PWM6
		T3CLKO_2	I/O	Clock out of timer 3

Pin number		name	type	description
LQFP64/QFN64	LQFP48/QFN48			
61	45	P0.2	I/O	Standard IO port
		AD2	I	Address/data bus
		ADC10	I	ADC analog input 10
		RxD4	I	Input of UART4
		PWM7_3	I/O	Capture of external signal/Pulse output of PWM7
		T4_2	I/O	Timer4 external input
62	46	P0.3	I/O	Standard IO port
		AD3	I	Address/data bus
		ADC11	I	ADC analog input 11
		TxD4	O	Transmit pin of UART 4
		PWM8_3	I/O	Capture of external signal/Pulse output of

				PWM8
		T4CLKO_2	I/O	Clock out of timer 4
63	47	P0.4	I/O	Standard IO port
		AD4	I	Address/data bus
		ADC12	I	ADC analog input 12
		T3	I	Timer3 external input
64	48	P5.2	I/O	Standard IO port
		RxD4_2	I	Input of UART4

2.6 STC8H4K64TLR-45I-LQFP48/QFN48/LQFP32/TSSOP20 (touch key/LED/RTC family)

2.6.1 Features and Price(Quasi 16-bit MCU with 16-bit hardware multiplier and divider MDU16)

- **Selection and price (No external crystal and external reset required with 15 channels 12-bit ADC)**

MCU	Operating voltage（V）	Flash Code Memory (100 thousand times) (Byte)	idata Internal DATA RAM(Byte)	xdata Internal extended SRAM (Byte)	Enhanced Dual DPTR increasing or decreasing	EEPROM 100 thousand times) (Byte)	Maximum I/O Lines	Traditional I/O interrupt(INT0/INT1/INT2/INT3/INT4) (can wake-up CPU)	All I/O ports support interrupts and can wake up MCU	DMA UARTs which can wake-up CPU	DMA 8080/6800 interface/ LCM driver(8-bit and 16-bit)	LED dirver	Touch key	RTC	DMA SPI which can wake-up CPU	I²C which can wake-up CPU	MDU16 (Hardware 16-bit Multiplier and Divider)	Timers/Counters (T0-T4 Pin Can wake-up CPU)	16-bit advanced PWM timer with Complementary symmetrical dead-time	Power-down Wake-up timer	DMA 15 channels high speed ADC (8 PWMs can be used as 8 DACs)	Comparator (May be used as ADC to detect external power-down)	Internal LVD interrupt (can wake-up CPU)	Watch-dog Timer	Internal high reliable reset circuit with 4 levels optional reset threshold voltage	Internal high presission Clock (adjustbal under 45MHz)	Clock output and Reset	Program encrypted transmission (Anti-blocking)	Password can be set for next update	Support RS485 download	Support software USB download directly	Online debug itself	Price & Package				products supply information
																																	LQFP48 <9mm*9mm>	QFN48 <6mm*6mm>	LQFP32	TSSOP20	
STC8H4K32TLR	1.9-5.5	32K	256	4K	2	32K	44	Y	Y	4	Y	Y	Y	Y	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	√		√	√	Some available
STC8H4K48TLR	1.9-5.5	48K	256	4K	2	16K	44	Y	Y	4	Y	Y	Y	Y	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	¥3.5		√	√	
STC8H4K64TLR	1.9-5.5	64K	256	4K	2	IAP	44	Y	Y	4	Y	Y	Y	Y	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	¥3.5		¥3.5	¥3	

- **Core**
  - ✓ Ultra-high speed 8051 Core with single clock per machine cycle, which is called 1T and the speed is about 12 times faster than traditional 8051
  - ✓ Fully compatible instruction set with traditional 8051
  - ✓ 41 interrupt sources and 4 interrupt priority levels
  - ✓ Online debugging is supported

- **Operating voltage**
  - ✓ 1.9V～5.5V

- **Operating temperature**
  - ✓ -40℃~85℃(The chip is produced in -40℃～125℃ process. Please refer to the description of the electrical characteristics chapter for applications beyond the temperature range)

- **Flash memory**
  - ✓ Up to 64Kbytes of Flash memory to be used to store user code
  - ✓ Configurable EEPROM size, 512bytes single page for being erased, which can be repeatedly erased more than 100 thousand times.
  - ✓ In-System-Programming, ISP in short, can be used to update the application code. No special programmer is needed.
  - ✓ Online debugging with single chip is supported, and no special emulator is needed. The number of breakpoints is unlimited theoratically.

- **SRAM**

- ✓ 128 bytes internal direct access RAM (DATA, use keyword *data* to declare in C language program)
- ✓ 128 bytes internal indirect access RAM (IDATA, use keyword *idata* to declare in C language program)
- ✓ 4096 bytes internal extended RAM (internal XDATA, use keyword *xdata* to declare in C language program)

➢ **Clock**

- ✓ Internal high precise RC clock IRC(IRC for short, ranges from 4MHz to 45MHz), adjustable while ISP and can be divided to lower frequency by user software, 100KHz for instance.
  - ✓ Error:±0.3% (at the temperature 25℃)
  - ✓ -1.35%～+1.30% temperature drift (at the temperature range of -40℃ to +85℃)
  - ✓ -0.76%～+0.98% temperature drift (at the temperature range of -20℃ to 65℃)
- ✓ Internal 32KHz low speed IRC with large error
- ✓ External crystal (4MHz～33MHz) and external clock

  Users can freely choose the above 3 clock sources

➢ **Reset**

- ✓ Hardware reset
  - ✓ Power-on reset. Measured voltage is 1.69V~1.82V. (Effective when the chip does not enable the low voltage reset function)

    The power-on reset voltage is a voltage range consisting of an upper limit voltage and a lower limit voltage. When the operating voltage drops from 5V / 3.3V to the lower limit threshold voltage of the power-on reset, the chip is in a reset state; when the voltage rises from 0V to the upper threshold voltage of power-on reset, the chip is released from the reset state.
  - ✓ Reset by reset pin. The default function of P5.4 is the I/O port. The P5.4 pin can be set as the reset pin while ISP download. (Note: When the P5.4 pin is set as the reset pin, the reset level is low.)
  - ✓ Watch dog timer reset
  - ✓ Low voltage detection reset. 4 low voltage detection levels are provided, 1.9V, 2.3V, 2.8V, 3.0V. Each level of low-voltage detection voltage is a voltage range consisting of an upper limit voltage and a lower limit voltage. When the operating voltage drops from 5V / 3.3V to the lower limit threshold voltage of low-voltage detection, the low-voltage detection takes effect. When the voltage rises from 0V to the upper threshold voltage, the low voltage detection becomes effective.
- ✓ Software reset
  - ✓ Writing the reset trigger register using software

➢ **Interrupts**

- ✓ 41 interrupt sources: INT0(Supports rising edge and falling edge interrupt), INT1(Supports rising edge and falling edge interrupt), INT2(Supports falling edge interrupt only), INT3(Supports falling edge interrupt only), INT4(Supports falling edge interrupt only), timer 0, timer 1, timer 2, timer 3, timer 4, UART 1, UART 2, UART 3, UART 4, ADC, LVD, SPI, I^{2}C, comparator, PWMA, PWMB, RTC, TKS, P1, P2, P3, P4, P5, LCM driver, DMA receive and transmit interrupts of UART 1, DMA receive and transmit interrupts of UART 2, DMA receive and transmit interrupts of UART 3, DMA receive and transmit interrupts of UART 4, DMA interrupt of SPI, DMA interrupt of ADC, DMA interrupt of LCM driver and DMA interrupt of memory-to-memory.
- ✓ 4 interrupt priority levels
- ✓ Interrupts that can wake up the CPU in clock stop mode: INT0(P3.2), INT1(P3.3), INT2(P3.6), INT3(P3.7), INT4(P3.0), T0(P3.4), T1(P3.5), T2(P1.2), T3(P0.4), T4(P0.6), RXD(P3.0/P3.6/P1.6/P4.3), RXD2(P1.0/P4.6), RXD3(P0.0/P5.0), RXD4(P0.2/P5.2), I2C_SDA(P1.4/P2.4/P3.3), SPI_SS(P5.4/P2.2/P3.5), Comparator interrupt, LVD interrupt, Power-down wake-up timer and interrupts of all I/O ports.

➢ **Digital peripherals**

- ✓ 5 16-bit timers: timer0, timer1, timer2, timer3, timer4, where the mode 3 of timer 0 has the Non-Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
- ✓ 4 high speed UARTs: UART1, UART2, UART3, UART4, whose maximum baudrate clock may be FOSC/4
- ✓ 8 channels/2 groups of enhanced PWM, which can realize control signals with dead time, and support external fault detection function. In addition, supports 16-bit timers, 8 external interrupts, 8 external captures and pulse width measurement functions.
- ✓ SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
- ✓ I^{2}C: Master mode or slave mode are supported.
- ✓ MDU16: Hardware 16-bit Multiplier and Divider which supports 32-bit divided by 16-bit, 16-bit divided by 16-bit, 16-bit multiplied by 16-bit, data shift, and data normalization operations.
- ✓ RTC: Support year, month, day, hour, minute, second, sub-second (1/128 second). And supports clock interrupt and a set of alarm clocks (Note: A version of the chip does not have this function)
- ✓ I/O port interrupt: All I/Os support interrupts, each group of I/O interrupts has an independent interrupt entry address, all I/O interrupts can support 4 types interrupt mode: high level interrupt, low level interrupt, rising edge interrupt, falling edge interrupt. Provides 4 levels of interrupt priority and supports power-down wake-up function.
- ✓ DMA: support Memory-To-Memory, SPI, UART1TX/UART1RX, UART2TX/UART2RX, UART3TX/UART3RX, UART4TX/UART4RX, ADC(Automatically calculates the average of multiple ADC results), LCM
- ✓ LCM (TFT color screen) dirver: support 8080 and 6800 interface, and support 8-bit and 16-bit data width (Note: A version of the chip does not have this function)
  - ✓ 8 bits 8080 data bus: 8 bits data lines (TD0~TD7), READ signael (TRD)c WRITE signal (TWR), RS line (TRS)
  - ✓ 16 bits 8080 bus: 16 bits data lines (TD0~TD15), READ signael (TRD)c WRITE signal (TWR), RS line (TRS)
  - ✓ 8 bits 6800 bus: 8 bits data lines (TD0~TD7), enable signal (TE) , READ and WRITE signal (TRW) , RS line (TRS)
  - ✓ 16 bits 6800 bus: 16 bits data lines (TD0~TD15), enable signal (TE) , READ and WRITE signal (TRW) , RS line (TRS)
  - ✓ Note: If you use 8-bit data lines to control the TFT screen, you generally need TD0~D7, TRD/TWR/TRS, 11 data and

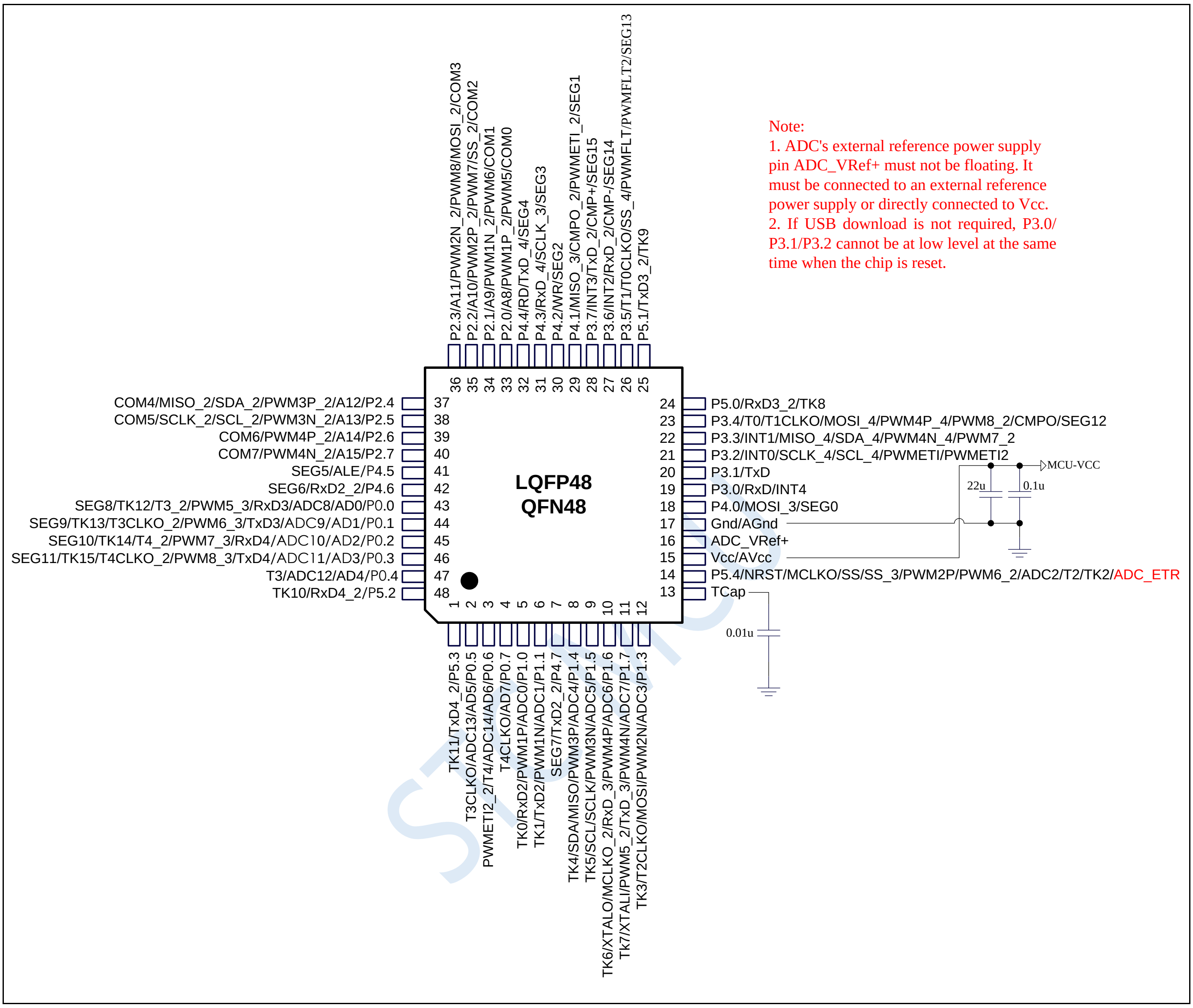
control lines, plus 2 common I/Os to control chip selection and reset (many TFT color screen chip selections and reset manufacturer has carried out automatic processing, does not need software control)

- **Analog peripherals**
  - ✓ Ultra high speed ADC which supports 12-bit precision 15 channels (channel 0 to channel 14) analog-to-digital conversion. The maximum speed can be 800K(800K ADC conversions per second)
  - ✓ ADC channel 15 is used to test the internal reference voltage. (The default internal reference voltage is 1.19V when the chip is shipped)
  - ✓ Comparator. A set of comparator (The CMP+ port and all ADC input ports can be selected as the positive terminal of the comparator, the CMP- port and the internal reference voltage 1.19V can be selected as the negative terminal of the comparator. So the comparator can be used as a multi-channel comparator for time division multiplexing)
  - ✓ Touch key: The microcontroller supports up to 16 touch keys. Every touch key can be enabled independently. The internal reference voltage is adjustable with 4 levels. Charge and discharge time settings and internal working frequency settings are flexible. The touch key supports wake-up CPU from low-power mode.
  - ✓ LED driver: The microcontroller can drive up to 128 (8 * 8 * 2) LEDs, support common negative mode, common positive mode and common negative/common positive mode, and support 8 levels of gray adjustment (brightness adjustment).
  - ✓ DAC: 8 channels advanced PWM timer can be used as 8 channels DAC

- **GPIO**
  - ✓ Up to 44 GPIOs: P0.0~P0.7, P1.0~P1.7(No P1.2), P2.0~P2.7, P3.0~P3.7, P4.0~P4.7, P5.0~P5.4
  - ✓ 4 modes for all GPIOs: quasi_bidirectional mode, push-pull outputmode, open drain mode, high-impedance input mode
  - ✓ Except for P3.0 and P3.1, all other I/O ports are in a high-impedance state after power-on. User must set the I/O ports mode before using them. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.

- **Package**
  - ✓ LQFP48 <9mm*9mm>, QFN48 <6mm*6mm>

2.6.2 Pinouts

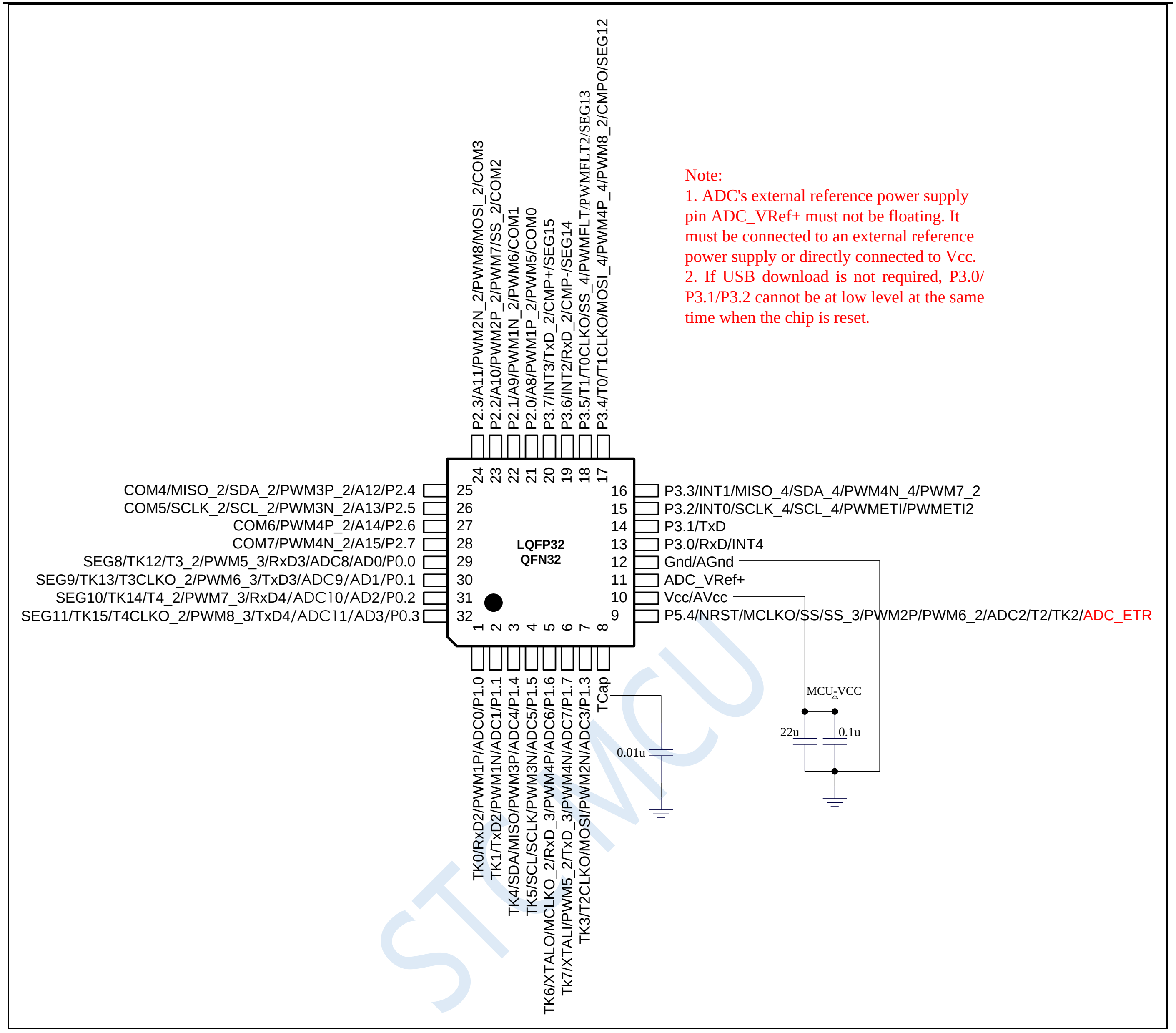

P2.3/A11/PWM2N_2/PWM8/MOSI_2/COM3
P2.2/A10/PWM2P_2/PWM7/SS_2/COM2
P2.1/A9/PWM1N_2/PWM6/COM1
P2.0/A8/PWM1P_2/PWM5/COM0
P4.4/RD/TxD_4/SEG4
P4.3/RxD_4/SCLK_3/SEG3
P4.2/WR/SEG2
P4.1/MISO_3/CMPO_2/PWMETI_2/SEG1
P3.7/INT3/TxD_2/CMP+/SEG15
P3.6/INT2/RxD_2/CMP-/SEG14
P3.5/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2/SEG13
P5.1/TxD3_2/TK9
36 35 34 33 32 31 30 29 28 27 26 25
COM4/MISO_2/SDA_2/PWM3P_2/A12/P2.4 37
COM5/SCLK_2/SCL_2/PWM3N_2/A13/P2.5 38
COM6/PWM4P_2/A14/P2.6 39
COM7/PWM4N_2/A15/P2.7 40
SEG5/ALE/P4.5 41
SEG6/RxD2_2/P4.6 42
SEG8/TK12/T3_2/PWM5_3/RxD3/ADC8/AD0/P0.0 43
SEG9/TK13/T3CLKO_2/PWM6_3/TxD3/ADC9/AD1/P0.1 44
SEG10/TK14/T4_2/PWM7_3/RxD4/ADC10/AD2/P0.2 45
SEG11/TK15/T4CLKO_2/PWM8_3/TxD4/ADC11/AD3/P0.3 46
T3/ADC12/AD4/P0.4 47
TK10/RxD4_2/P5.2 48
LQFP48
QFN48
24 P5.0/RxD3_2/TK8
23 P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO/SEG12
22 P3.3/INT1/MISO_4/SDA_4/PWM4N_4/PWM7_2
21 P3.2/INT0/SCLK_4/SCL_4/PWMETI/PWMETI2
20 P3.1/TxD
19 P3.0/RxD/INT4
18 P4.0/MOSI_3/SEG0
17 Gnd/AGnd
16 ADC_VRef+
15 Vcc/AVcc
14 P5.4/NRST/MCLKO/SS/SS_3/PWM2P/PWM6_2/ADC2/T2/TK2/ADC_ETR
13 TCap
MCU-VCC
22u
0.1u
0.01u
1 2 3 4 5 6 7 8 9 10 11 12
TK11/TxD4_2/P5.3
T3CLKO/ADC13/AD5/P0.5
PWMETI2_2/T4/ADC14/AD6/P0.6
T4CLKO/AD7/P0.7
TK0/RxD2/PWM1P/ADC0/P1.0
TK1/TxD2/PWM1N/ADC1/P1.1
SEG7/TxD2_2/P4.7
TK4/SDA/MISO/PWM3P/ADC4/P1.4
TK5/SCL/SCLK/PWM3N/ADC5/P1.5
TK6/XTALO/MCLKO_2/RxD_3/PWM4P/ADC6/P1.6
TK7/XTALI/PWM5_2/TxD_3/PWM4N/ADC7/P1.7
TK3/T2CLKO/MOSI/PWM2N/ADC3/P1.3

Note:
1. ADC's external reference power supply pin ADC_VRef+ must not be floating. It must be connected to an external reference power supply or directly connected to Vcc.
2. If USB download is not required, P3.0/P3.1/P3.2 cannot be at low level at the same time when the chip is reset.

The download steps using ISP and notes are the same as the circumstances in 2.1.2.

Note:

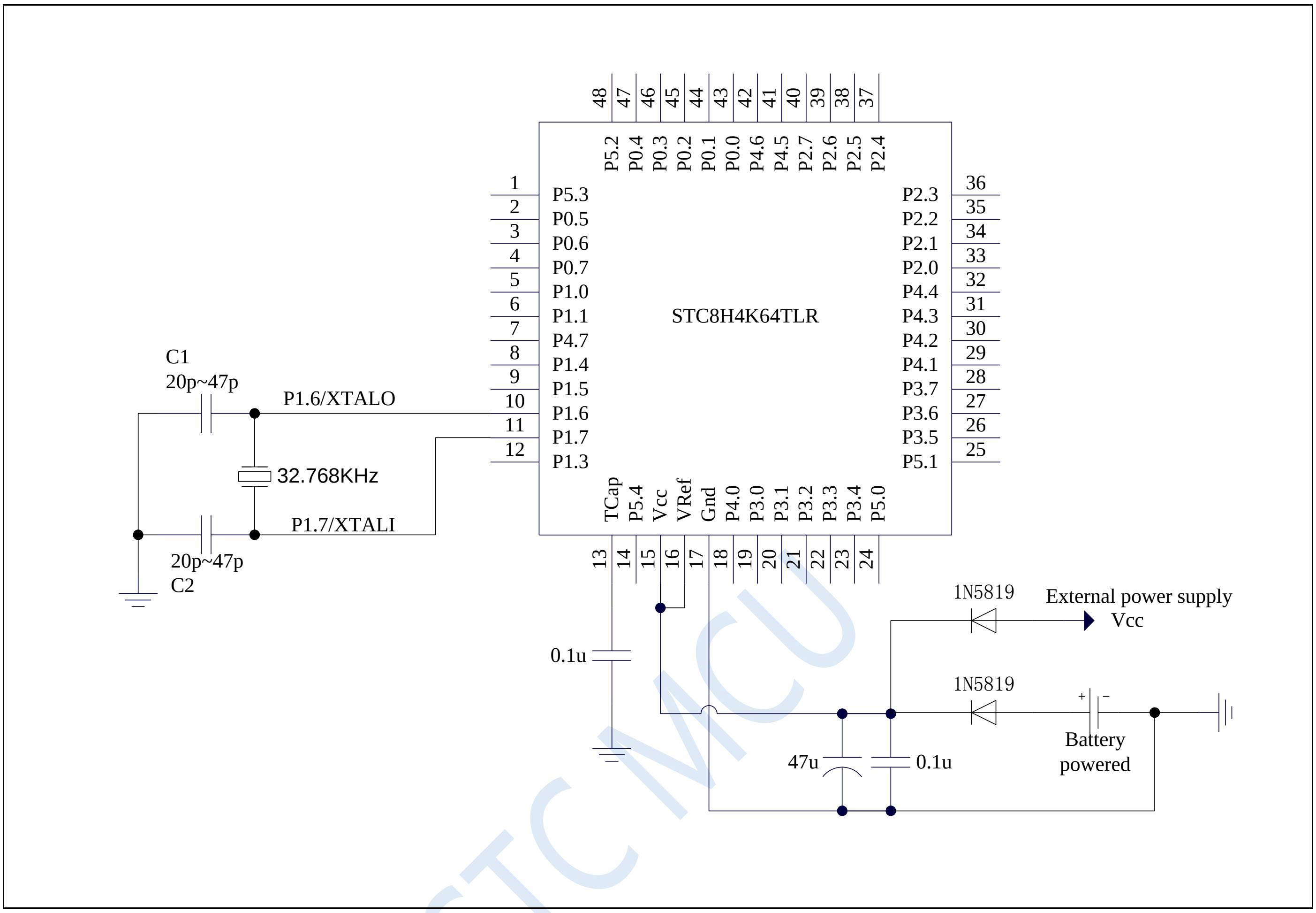
1. **Except for P3.0 and P3.1, all other I/O ports are in high-impedance input state after power-on. User must set the I/O port mode firstly when using I/O.**
2. **All I/O ports can be set to quasi-bidirectional port mode, strong push-pull output mode, open-drain output mode or high-impedance input mode. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.**
3. **When P5.4 is enabled as the reset pin, the reset level is low.**


P2.3/A11/PWM2N_2/PWM8/MOSI_2/COM3
P2.2/A10/PWM2P_2/PWM7/SS_2/COM2
P2.1/A9/PWM1N_2/PWM6/COM1
P2.0/A8/PWM1P_2/PWM5/COM0
P3.7/INT3/TxD_2/CMP+/SEG15
P3.6/INT2/RxD_2/CMP-/SEG14
P3.5/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2/SEG13
P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO/SEG12
24
23
22
21
20
19
18
17
COM4/MISO_2/SDA_2/PWM3P_2/A12/P2.4
COM5/SCLK_2/SCL_2/PWM3N_2/A13/P2.5
COM6/PWM4P_2/A14/P2.6
COM7/PWM4N_2/A15/P2.7
SEG8/TK12/T3_2/PWM5_3/RxD3/ADC8/AD0/P0.0
SEG9/TK13/T3CLKO_2/PWM6_3/TxD3/ADC9/AD1/P0.1
SEG10/TK14/T4_2/PWM7_3/RxD4/ADC10/AD2/P0.2
SEG11/TK15/T4CLKO_2/PWM8_3/TxD4/ADC11/AD3/P0.3
25
26
27
28
29
30
31
32
LQFP32
QFN32
16
15
14
13
12
11
10
9
P3.3/INT1/MISO_4/SDA_4/PWM4N_4/PWM7_2
P3.2/INT0/SCLK_4/SCL_4/PWMETI/PWMETI2
P3.1/TxD
P3.0/RxD/INT4
Gnd/AGnd
ADC_VRef+
Vcc/AVcc
P5.4/NRST/MCLKO/SS/SS_3/PWM2P/PWM6_2/ADC2/T2/TK2/ADC_ETR
1
2
3
4
5
6
7
8
TK0/RxD2/PWM1P/ADC0/P1.0
TK1/TxD2/PWM1N/ADC1/P1.1
TK4/SDA/MISO/PWM3P/ADC4/P1.4
TK5/SCL/SCLK/PWM3N/ADC5/P1.5
TK6/XTALO/MCLKO_2/RxD_3/PWM4P/ADC6/P1.6
TK7/XTALI/PWM5_2/TxD_3/PWM4N/ADC7/P1.7
TK3/T2CLKO/MOSI/PWM2N/ADC3/P1.3
TCap
0.01u
MCU-VCC
22u
0.1u
STCMCU

Note:
1. ADC's external reference power supply pin ADC_VRef+ must not be floating. It must be connected to an external reference power supply or directly connected to Vcc.
2. If USB download is not required, P3.0/P3.1/P3.2 cannot be at low level at the same time when the chip is reset.

The download steps using ISP and notes are the same as the circumstances in 2.1.2.

2.6.3 RTC reference circuit diagram (No VBAT pin)



2.6.4 Pin descriptions

Pin number		name	type	description
LQFP48/QFN48	LQFP32/QFN32			
1		P5.3	I/O	Standard IO port
		TxD4_2	O	Transmit pin of UART 4
		TK11	I	Touch key
2		P0.5	I/O	Standard IO port
		AD5	I	Address/data bus
		ADC13	I	ADC analog input 13
		T3CLKO	O	Clock out of timer 3
3		P0.6	I/O	Standard IO port
		AD6	I	Address/data bus
		ADC14	I	ADC analog input 14
		T4	I	Timer4 external input
		PWMFLT2_2	I	Enhance PWM external anomaly detection pin 2
4		P0.7	I/O	Standard IO port
		AD7	I	Address/data bus
		T4CLKO	O	Clock out of timer 4
5	1	P1.0	I/O	Standard IO port
		ADC0	I	ADC analog input 0
		PWM1P	I/O	Capture of external signal/ Positive of PWMA pulse output
		RxD2	I	Input of UART2
		TK0	I	Touch key
6	2	P1.1	I/O	Standard IO port
		ADC1	I	ADC analog input 1
		PWM1N	I/O	Capture of external signal/ Negative of PWMA pulse output
		TxD2	I	Transmit pin of UART 2
		TK1	I	Touch key
7		P4.7	I/O	Standard IO port
		TxD2_2	I	Transmit pin of UART 2
		SEG7	O	LED driver
8	3	P1.4	I/O	Standard IO port
		ADC4	I	ADC analog input 4
		PWM3P	I/O	Capture of external signal/Positive of PWM3 pulse output
		MISO	I/O	Master Iutput/Slave Onput of SPI
		SDA	I/O	Serial data line of I2C
		TK4	I	Touch key
9	4	P1.5	I/O	Standard IO port
		ADC5	I	ADC analog input 5
		PWM3N	I/O	Capture of external signal/Negative of PWM3 pulse output
		SCLK	I/O	Serial Clock of SPI
		SCL	I/O	Serial Clock line of I2C
		TK5	I	Touch key

Pin number		name	type	description
LQFP48/QFN48	LQFP32/QFN32			
10	5	P1.6	I/O	Standard IO port
		ADC6	I	ADC analog input 6
		RxD_3	I	Input of UART 1
		PWM4P	I/O	Capture of external signal/Positive of PWM4 pulse output
		MCLKO_2	O	Main clock output
		XTALO	O	Connect to external oscillator
		TK6	I	Touch key
11	6	P1.7	I/O	Standard IO port
		ADC7	I	ADC analog input 7
		TxD_3	O	Transmit pin of UART 1
		PWM4N	I/O	Capture of external signal/Negative of PWM4 pulse output
		PWM5_2	I/O	Capture of external signal/Pulse output of PWM5
		XTALI	I	Connect to external oscillator
		TK7	I	Touch key
12	7	P1.3	I/O	Standard IO port
		ADC3	I	ADC analog input 3
		MOSI	I/O	Master Output/Slave Input of SPI
		PWM2N	I/O	Capture of external signal/Negative of PWM2 pulse output
		T2CLKO	O	Clock out of timer 2
		TK3	I	Touch key
13	8	TCAP	I	Charge and discharge capacitance of Touch key
14	9	P5.4	I/O	Standard IO port
		NRST	I	Reset pin (low level reset)
		MCLKO	O	Main clock output
		SS_3	I	Slave selection of SPI (it is output with regard to master)
		SS	I	Slave selection of SPI (it is output with regard to master)
		PWM2P	I/O	Capture of external signal/Positive of PWM2 pulse output
		PWM6_2	I/O	Capture of external signal/Pulse output of PWM6
		T2	I	Timer2 external input
		ADC2	I	ADC analog input 2
		TK2	I	Touch key
		ADC_ETR	I	ADC external trigger pin
15	10	Vcc	Vcc	Power Supply
		AVcc	Vcc	ADC Power Supply
16	11	ADC_VRef+	I	Reference voltage pin of ADC, which can be directly connected to the VCC of the MCU when the requirements are not high
17	12	Gnd	Gnd	Ground
		AGnd	Gnd	ADC Ground
18		P4.0	I/O	Standard IO port
		MOSI_3	I/O	Master Output/Slave Input of SPI
		SEG0	O	LED dirver
19	13	P3.0	I/O	Standard IO port
		RxD	I	Input of UART1
		INT4	I	External interrupt4

Pin number		name	type	description
LQFP48/QFN48	LQFP32/QFN32			
20	14	P3.1	I/O	Standard IO port
		TxD	O	Transmit pin of UART 1

21	15	P3.2	I/O	Standard IO port
		INT0	I	External interrupt0
		SCLK_4	I/O	Serial Clock of SPI
		SCL_4	I/O	Serial Clock line of I2C
		PWMETI	I	PWM external trigger input pin
		PWMET2	I	PWM external trigger input pin 2
22	16	P3.3	I/O	Standard IO port
		INT1	I	External interrupt1
		MISO_4	I/O	Master Iutput/Slave Onput of SPI
		SDA_4	I/O	Serial data line of I2C
		PWM4N_4	I/O	Capture of external signal/Negative of PWM4 pulse output
		PWM7_2	I/O	Capture of external signal/Pulse output of PWM7
23	17	P3.4	I/O	Standard IO port
		T0	I	Timer0 external input
		T1CLKO	O	Clock out of timer 1
		MOSI_4	I/O	Master Output/Slave Input of SPI
		PWM4P_4	I/O	Capture of external signal/Positive of PWM4 pulse output
		PWM8_2	I/O	Capture of external signal/Pulse output of PWM8
		CMPO	O	Output of comparator
		SEG12	O	LED dirver
24		P5.0	I/O	Standard IO port
		RxD3_2	I	Input of UART3
		TK8	I	Touch key
25		P5.1	I/O	Standard IO port
		TxD3_2	O	Transmit pin of UART 3
		TK9	I	Touch key
26	18	P3.5	I/O	Standard IO port
		T1	I	Timer1 external input
		T0CLKO	O	Clock out of timer 0
		SS_4	I	Slave selection of SPI (it is output with regard to master)
		PWMFLT	I	Enhance PWMA external anomaly detection pin
		PWMFLT2	I	Enhance PWMB external anomaly detection pin
		SEG13	O	LED dirver
27	19	P3.6	I/O	Standard IO port
		INT2	I	External interrupt2
		RxD_2	I	Input of UART1
		CMP-	I	Negative input of comparator
		SEG14	O	LED dirver
28	20	P3.7	I/O	Standard IO port
		INT3	I	External interrupt3
		TxD_2	O	Transmit pin of UART 1
		CMP+	I	Positive input of comparator
		SEG15	O	LED dirver

Pin number		name	type	description
LQFP48/QFN48	LQFP32/QFN32			
29		P4.1	I/O	Standard IO port
		MISO_3	I/O	Master Iutput/Slave Onput of SPI
		CMPO_2	O	Output of comparator
		PWMETI_3	I	PWM external trigger input pin
		SEG1	O	LED dirver
30		P4.2	I/O	Standard IO port
		WR	O	WRITE signal of external bus

		SEG2	O	LED dirver
31		P4.3	I/O	Standard IO port
		RxD_4	I	Input of UART1
		SCLK_3	I/O	Serial Clock of SPI
		SEG3	O	LED dirver
32		P4.4	I/O	Standard IO port
		RD	O	READ signal of external bus
		TxD_4	O	Transmit pin of UART 1
		SEG4	O	LED dirver
33	21	P2.0	I/O	Standard IO port
		A8	I	Address bus
		PWM1P_2	I/O	Capture of external signal/Positive of PWMA pulse output
		PWM5	I/O	Capture of external signal/Pulse output of PWM5
		COM0	O	LED dirver
34	22	P2.1	I/O	Standard IO port
		A9	I	Address bus
		PWM1N_2	I/O	Capture of external signal/Negative of PWMA pulse output
		PWM6	I/O	Capture of external signal/Pulse output of PWM6
		COM1	O	LED dirver
35	23	P2.2	I/O	Standard IO port
		A10	I	Address bus
		SS_2	I	Slave selection of SPI (it is output with regard to master)
		PWM2P_2	I/O	Capture of external signal/Positive of PWMB pulse output
		PWM7	I/O	Capture of external signal/Pulse output of PWM7
		COM2	O	LED dirver
36	24	P2.3	I/O	Standard IO port
		A11	I	Address bus
		MOSI_2	I/O	Master Output/Slave Input of SPI
		PWM2N_2	I/O	Capture of external signal/Negative of PWMB pulse output
		PWM8	I/O	Capture of external signal/Pulse output of PWM8
		COM3	O	LED dirver
37	25	P2.4	I/O	Standard IO port
		A12	I	Address bus
		MISO_2	I/O	Master Iutput/Slave Onput of SPI
		SDA_2	I/O	Serial data line of I2C
		PWM3P_2	I/O	Capture of external signal/Positive of PWM3 pulse output
		COM4	O	LED dirver
38	26	P2.5	I/O	Standard IO port
		A13	I	Address bus
		SCLK_2	I/O	Serial Clock of SPI
		SCL_2	I/O	Serial Clock line of I2C
		PWM3N_2	I/O	Capture of external signal/Negative of PWM3 pulse output
		COM5	O	LED dirver

Pin number		**name**	**type**	**description**
LQFP48/QFN48	**LQFP32/QFN32**			
39	27	P2.6	I/O	Standard IO port
		A14	I	Address bus
		PWM4P_2	I/O	Capture of external signal/Positive of PWM4 pulse output
		COM6	O	LED dirver
40	28	P2.7	I/O	Standard IO port
		A15	I	Address bus
		PWM4N_2	I/O	Capture of external signal/Negative of PWM4 pulse output
		COM7	O	LED dirver
41		P4.5	I/O	Standard IO port

		ALE	O	Address Latch Enable signal
		SEG5	O	LED dirver
42		P4.6	I/O	Standard IO port
		RxD2_2	I	Input of UART2
		SEG6	O	LED dirver
43	29	P0.0	I/O	Standard IO port
		AD0	I	Address/data bus
		ADC8	I	ADC analog input 8
		RxD3	I	Input of UART3
		PWM5_3	I/O	Capture of external signal/Pulse output of PWM5
		T3_2	I	Timer3 external input
		TK12	I	Touch key
		SEG8	O	LED dirver
44	30	P0.1	I/O	Standard IO port
		AD1	I	Address/data bus
		ADC9	I	ADC analog input 9
		TXD3	O	Transmit pin of UART 3
		PWM6_3	I/O	Capture of external signal/Pulse output of PWM6
		T3CLKO_2	O	Clock out of timer 3
		TK13	I	Touch key
		SEG9	O	LED dirver
45	31	P0.2	I/O	Standard IO port
		AD2	I	Address/data bus
		ADC10	I	ADC analog input 10
		RxD4	I	Input of UART4
		PWM7_3	I/O	Capture of external signal/Pulse output of PWM7
		T4_2	I	Timer4 external input
		TK14	I	Touch key
		SEG10	O	LED dirver
46	32	P0.3	I/O	Standard IO port
		AD3	I	Address/data bus
		ADC11	I	ADC analog input 11
		TxD4	O	Transmit pin of UART 4
		PWM8_3	I/O	Capture o f external signal/Pulse output of PWM8
		T4CLKO_2	O	Clock out of timer 4
		TK15	I	Touch key
		SEG11	O	LED dirver
47		P0.4	I/O	Standard IO port
		AD4	I	Address/data bus
		ADC12	I	ADC analog input 12
		T3	I	Timer3 external input
48		P5.2	I/O	Standard IO port
		RxD4_2	I	Input of UART4

2.7 STC8H4K64TLCD-45I-LQFP64/QFN64/LQFP48/QFN48 (Touch key/LCD/RTC family)

2.7.1 Features and Price(Quasi 16-bit MCU with 16-bit hardware multiplier and divider MDU16)

- **Selection and price (No external crystal and external reset required with 15 channels 12-bit ADC)**

MCU	Operating voltage（V）	Flash Code Memory (100 thousand times) (Byte)	idata Internal DATA RAM(Byte)	xdata Internal extended SRAM (Byte)	Enhanced Dual DPTR increasing or decreasing	EEPROM 100 thousand times) (Byte)	Maximum I/O Lines	Traditional I/O interrupt(INT0/INT1/INT2/INT3/INT4) (can wake-up CPU)	All I/O ports support interrupts and can wake up MCU	DMA UARTs which can wake-up CPU	DMA 8080/6800 interface/ LCM driver(8-bit and 16-bit)	LCD driver (4COM*40SEG)	Touch key	RTC	DMA SPI which can wake-up CPU	I²C which can wake-up CPU (No DMA)	MDU16 (Hardware 16-bit Multiplier and Divider)	Timers/Counters (T0-T4 Pin can wake-up CPU)	16-bit advanced PWM timer with Complementary symmetrical dead-time	Power-down Wake-up timer	DMA 15 channels high speed ADC (8 PWMs can be used as 8 DACs)	Comparator (May be used as ADC to detect external power-down)	Internal LVD interrupt (can wake-up CPU)	Watch-dog Timer	Internal high reliable reset circuit with 4 levels optional reset threshold voltage	Internal high precision Clock (adjustbal under 45MHz)	Clock output and Reset	Program encrypted transmission (Anti-blocking)	Password can be set for next update	Support RS485 download	Support software USB download directly	Online debug itself	Price & Package				products supply information
																																	LQFP64 <12mm*12mm>	QFN64 <8mm*8mm>	LQFP48 <9mm*9mm>	QFN48 <6mm*6mm>	
STC8H4K32TLCD	1.9-5.5	32K	256	4K	2	32K	60	Y	Y	4	Y	Y	Y	Y	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	√				Samples
STC8H4K48TLCD	1.9-5.5	48K	256	4K	2	16K	60	Y	Y	4	Y	Y	Y	Y	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	√				
STC8H4K64TLCD	1.9-5.5	64K	256	4K	2	IAP	60	Y	Y	4	Y	Y	Y	Y	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	√				

- **Core**
  - ✓ Ultra-high speed 8051 Core with single clock per machine cycle, which is called 1T and the speed is about 12 times faster than traditional 8051
  - ✓ Fully compatible instruction set with traditional 8051
  - ✓ 43 interrupt sources and 4 interrupt priority levels
  - ✓ Online debugging is supported
- **Operating voltage**
  - ✓ 1.9V～5.5V
- **Operating temperature**
  - ✓ -40℃~85℃(The chip is produced in -40℃～125℃ process. Please refer to the description of the electrical characteristics chapter for applications beyond the temperature range)
- **Flash memory**
  - ✓ Up to 64Kbytes of Flash memory to be used to store user code
  - ✓ Configurable EEPROM size, 512bytes single page for being erased, which can be repeatedly erased more than 100 thousand times.
  - ✓ In-System-Programming, ISP in short, can be used to update the application code. No special programmer is needed.
  - ✓ Online debugging with single chip is supported, and no special emulator is needed. The number of breakpoints is unlimited theoratically.
- **SRAM**
  - ✓ 128 bytes internal direct access RAM (DATA, use keyword *data* to declare in C language program)
  - ✓ 128 bytes internal indirect access RAM (IDATA, use keyword *idata* to declare in C language program)
  - ✓ 4096 bytes internal extended RAM (internal XDATA, use keyword *xdata* to declare in C language program)
- **Clock**
  - ✓ Internal high precise RC clock IRC(IRC for short, ranges from 4MHz to 45MHz), adjustable while ISP and can be divided to lower frequency by user software, 100KHz for instance.
    - ✓ Error:±0.3% (at the temperature 25℃)
    - ✓ -1.35%～+1.30% temperature drift (at the temperature range of -40℃ to +85℃)
    - ✓ -0.76%～+0.98% temperature drift (at the temperature range of -20℃ to 65℃)
  - ✓ Internal 32KHz low speed IRC with large error
  - ✓ External crystal (4MHz～33MHz) and external clock

    Users can freely choose the above 3 clock sources
- **Reset**
  - ✓ Hardware reset
    - ✓ Power-on reset. Measured voltage is 1.69V~1.82V. (Effective when the chip does not enable the low voltage reset

    function)
    The power-on reset voltage is a voltage range consisting of an upper limit voltage and a lower limit voltage. When the operating voltage drops from 5V / 3.3V to the lower limit threshold voltage of the power-on reset, the chip is in a reset state; when the voltage rises from 0V to the upper threshold voltage of power-on reset, the chip is released from the reset state.
    - Reset by reset pin. The default function of P5.4 is the I/O port. The P5.4 pin can be set as the reset pin while ISP download. (Note: When the P5.4 pin is set as the reset pin, the reset level is low.)
    - Watch dog timer reset
    - Low voltage detection reset. 4 low voltage detection levels are provided, 1.9V, 2.3V, 2.8V, 3.0V. Each level of low-voltage detection voltage is a voltage range consisting of an upper limit voltage and a lower limit voltage. When the operating voltage drops from 5V / 3.3V to the lower limit threshold voltage of low-voltage detection, the low-voltage detection takes effect. When the voltage rises from 0V to the upper threshold voltage, the low voltage detection becomes effective.
- Software reset
    - Writing the reset trigger register using software

- **Interrupts**
    - 43 interrupt sources: INT0(Supports rising edge and falling edge interrupt), INT1(Supports rising edge and falling edge interrupt), INT2(Supports falling edge interrupt only), INT3(Supports falling edge interrupt only), INT4(Supports falling edge interrupt only), timer 0, timer 1, timer 2, timer 3, timer 4, UART 1, UART 2, UART 3, UART 4, ADC, LVD, SPI, I^{2}C, comparator, PWMA, PWMB, RTC, TKS, P1, P2, P3, P4, P5, P6, P7, LCM driver, DMA receive and transmit interrupts of UART 1, DMA receive and transmit interrupts of UART 2, DMA receive and transmit interrupts of UART 3, DMA receive and transmit interrupts of UART 4, DMA interrupt of SPI, DMA interrupt of ADC, DMA interrupt of LCM driver and DMA interrupt of memory-to-memory.
    - 4 interrupt priority levels
    - Interrupts that can wake up the CPU in clock stop mode: INT0(P3.2), INT1(P3.3), INT2(P3.6), INT3(P3.7), INT4(P3.0), T0(P3.4), T1(P3.5), T2(P1.2), T3(P0.4), T4(P0.6), RXD(P3.0/P3.6/P1.6/P4.3), RXD2(P1.0/P4.6), RXD3(P0.0/P5.0), RXD4(P0.2/P5.2), I2C_SDA(P1.4/P2.4/P3.3), SPI_SS(P5.4/P2.2/P3.5), Comparator interrupt, LVD interrupt, Power-down wake-up timer and interrupts of all I/O ports.

- **Digital peripherals**
    - 5 16-bit timers: timer0, timer1, timer2, timer3, timer4, where the mode 3 of timer 0 has the Non-Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
    - 4 high speed UARTs: UART1, UART2, UART3, UART4, whose maximum baudrate clock may be FOSC/4
    - 8 channels/2 groups of enhanced PWM, which can realize control signals with dead time, and support external fault detection function. In addition, supports 16-bit timers, 8 external interrupts, 8 external captures and pulse width measurement functions.
    - SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
    - I^{2}C: Master mode or slave mode are supported.
    - MDU16: Hardware 16-bit Multiplier and Divider which supports 32-bit divided by 16-bit, 16-bit divided by 16-bit, 16-bit multiplied by 16-bit, data shift, and data normalization operations.
    - RTC: Support year, month, day, hour, minute, second, sub-second (1/128 second). And supports clock interrupt and a set of alarm clocks (Note: A version of the chip does not have this function)
    - I/O port interrupt: All I/Os support interrupts, each group of I/O interrupts has an independent interrupt entry address, all I/O interrupts can support 4 types interrupt mode: high level interrupt, low level interrupt, rising edge interrupt, falling edge interrupt. Provides 4 levels of interrupt priority and supports power-down wake-up function.
    - DMA: support Memory-To-Memory, SPI, UART1TX/UART1RX, UART2TX/UART2RX, UART3TX/UART3RX, UART4TX/UART4RX, ADC(Automatically calculates the average of multiple ADC results), LCM
    - LCM (TFT color screen) dirver: support 8080 and 6800 interface, and support 8-bit and 16-bit data width (Note: A version of the chip does not have this function)
        - 8 bits 8080 data bus: 8 bits data lines (TD0~TD7), READ signael (TRD)c WRITE signal (TWR), RS line (TRS)
        - 16 bits 8080 bus: 16 bits data lines (TD0~TD15), READ signael (TRD)c WRITE signal (TWR), RS line (TRS)
        - 8 bits 6800 bus: 8 bits data lines (TD0~TD7), enable signal (TE) , READ and WRITE signal (TRW) , RS line (TRS)
        - 16 bits 6800 bus: 16 bits data lines (TD0~TD15), enable signal (TE) , READ and WRITE signal (TRW) , RS line (TRS)
        - Note: If you use 8-bit data lines to control the TFT screen, you generally need TD0~D7, TRD/TWR/TRS, 11 data and control lines, plus 2 common I/Os to control chip selection and reset (many TFT color screen chip selections and reset manufacturer has carried out automatic processing, does not need software control)
    - LCD dirver: support up to 4COM*40 SEGs and 8 levels grayscale adjustment

- **Analog peripherals**
    - Ultra high speed ADC which supports 12-bit precision 15 channels (channel 0 to channel 14) analog-to-digital conversion. The maximum speed can be 800K(800K ADC conversions per second)
    - ADC channel 15 is used to test the internal reference voltage. (The default internal reference voltage is 1.19V when the chip is shipped)
    - Comparator. A set of comparator (The CMP+ port and all ADC input ports can be selected as the positive terminal of the comparator. So the comparator can be used as a multi-channel comparator for time division multiplexing)
    - Touch key: The microcontroller supports up to 16 touch keys. Every touch key can be enabled independently. The internal reference voltage is adjustable with 4 levels. Charge and discharge time settings and internal working frequency settings are flexible. The touch key supports wake-up CPU from low-power mode.
    - DAC: 8 channels advanced PWM timer can be used as 8 channels DAC

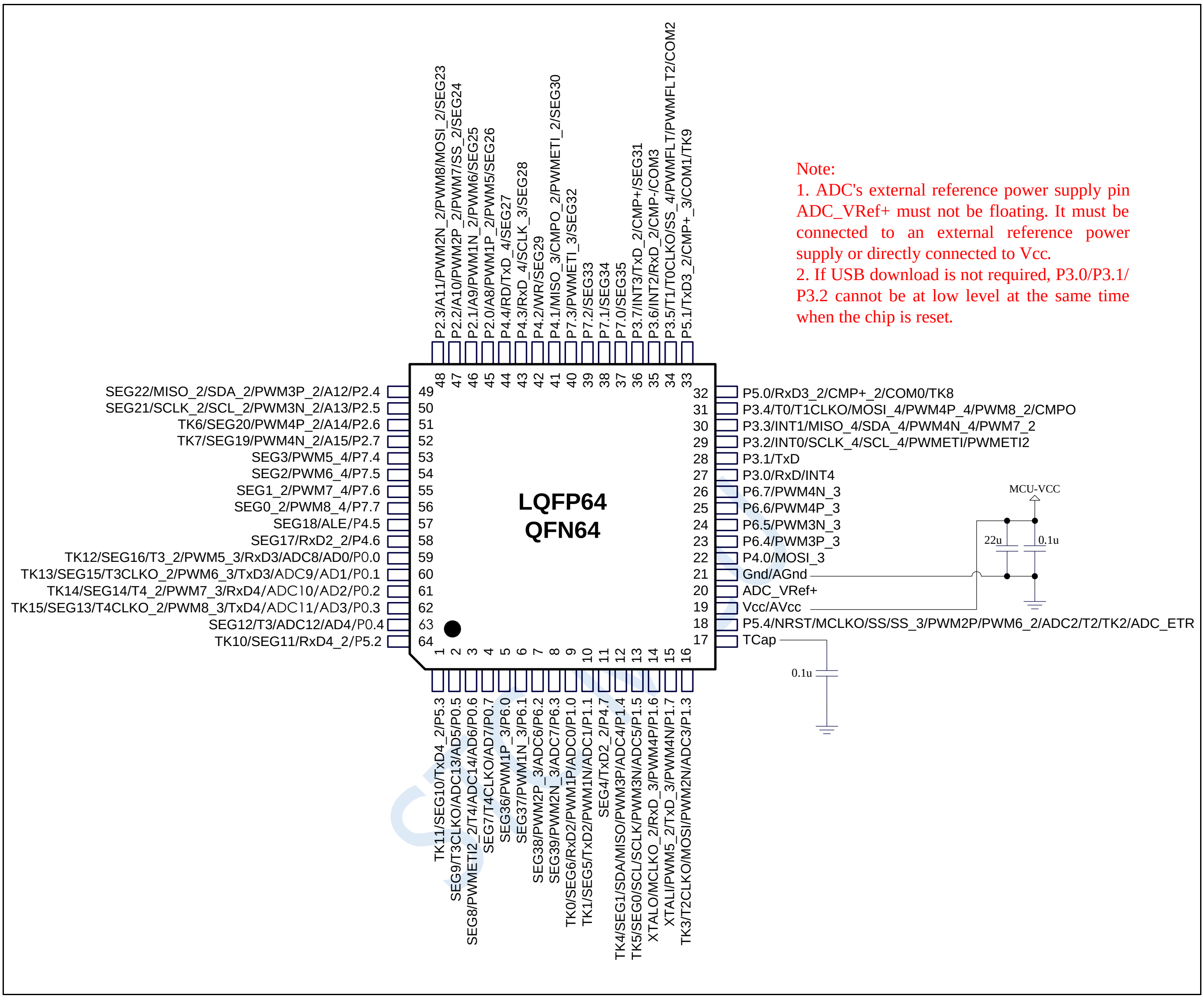
- **GPIO**

- ✓ Up to 60 GPIOs: P0.0~P0.7, P1.0~P1.7(No P1.2), P2.0~P2.7, P3.0~P3.7, P4.0~P4.7, P5.0~P5.4, P6.0~P6.7, P7.0~P7.7
- ✓ 4 modes for all GPIOs: quasi_bidirectional mode, push-pull outputmode, open drain mode, high-impedance input mode
- ✓ Except for P3.0 and P3.1, all other I/O ports are in a high-impedance state after power-on. User must set the I/O ports mode before using them. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.

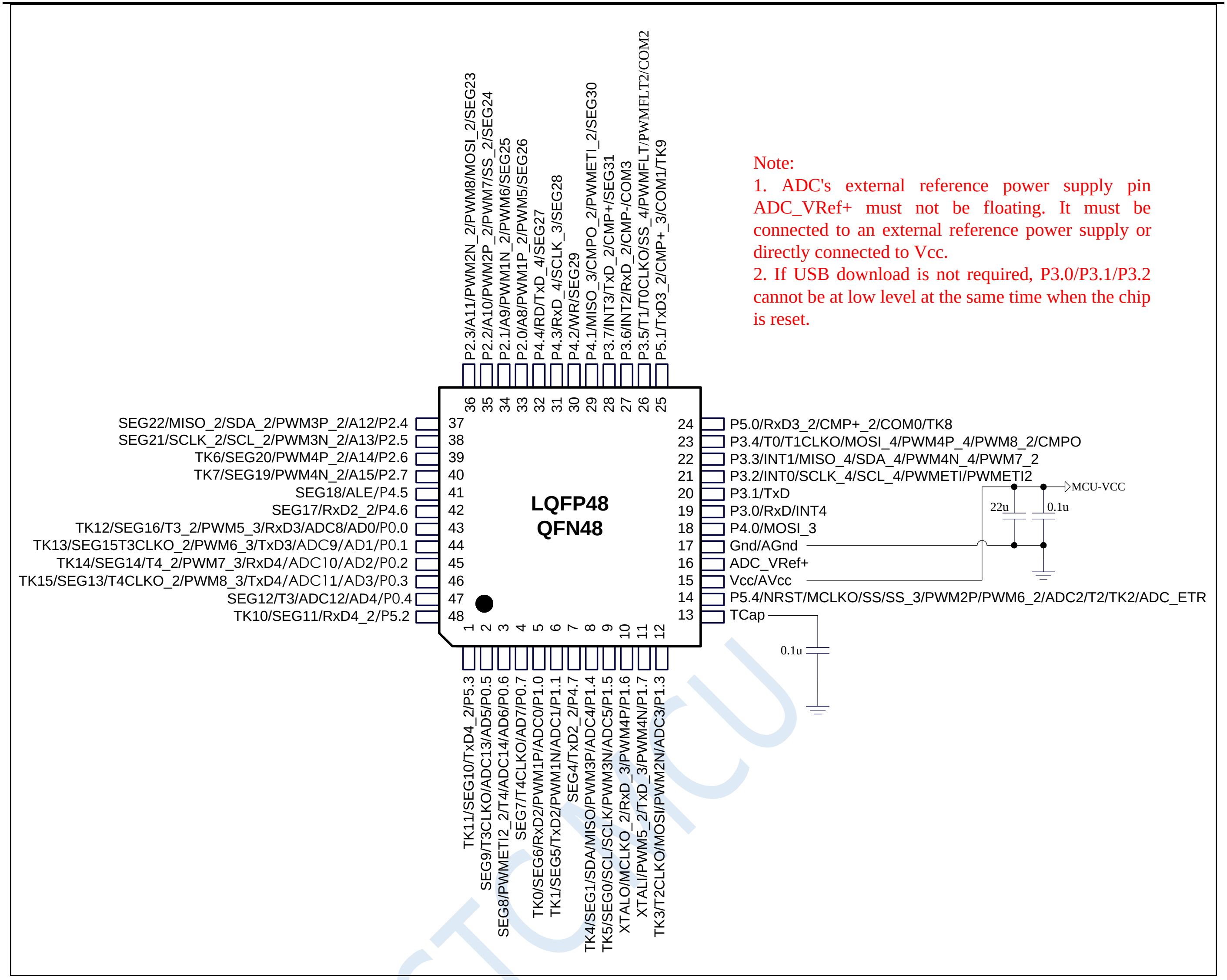
➢ **Package**

- ✓ LQFP64 <12mm*12mm>, QFN64 <8mm*8mm>, LQFP48 <9mm*9mm>, QFN48 <6mm*6mm>

2.7.2 Pinouts


LQFP64
QFN64
1 TK11/SEG10/TxD4_2/P5.3
2 SEG9/T3CLKO/ADC13/AD5/P0.5
3 SEG8/PWMETI2_2/T4/ADC14/AD6/P0.6
4 SEG7/T4CLKO/AD7/P0.7
5 SEG36/PWM1P_3/P6.0
6 SEG37/PWM1N_3/P6.1
7 SEG38/PWM2P_3/ADC6/P6.2
8 SEG39/PWM2N_3/ADC7/P6.3
9 TK0/SEG6/RxD2/PWM1P/ADC0/P1.0
10 TK1/SEG5/TxD2/PWM1N/ADC1/P1.1
11 SEG4/TxD2_2/P4.7
12 TK4/SEG1/SDA/MISO/PWM3P/ADC4/P1.4
13 TK5/SEG0/SCL/SCLK/PWM3N/ADC5/P1.5
14 XTALO/MCLKO_2/RxD_3/PWM4P/P1.6
15 XTALI/PWM5_2/TxD_3/PWM4N/P1.7
16 TK3/T2CLKO/MOSI/PWM2N/ADC3/P1.3
17 TCap
18 P5.4/NRST/MCLKO/SS/SS_3/PWM2P/PWM6_2/ADC2/T2/TK2/ADC_ETR
19 Vcc/AVcc
20 ADC_VRef+
21 Gnd/AGnd
22 P4.0/MOSI_3
23 P6.4/PWM3P_3
24 P6.5/PWM3N_3
25 P6.6/PWM4P_3
26 P6.7/PWM4N_3
27 P3.0/RxD/INT4
28 P3.1/TxD
29 P3.2/INT0/SCLK_4/SCL_4/PWMETI/PWMETI2
30 P3.3/INT1/MISO_4/SDA_4/PWM4N_4/PWM7_2
31 P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO
32 P5.0/RxD3_2/CMP+_2/COM0/TK8
33 P5.1/TxD3_2/CMP+_3/COM1/TK9
34 P3.5/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2/COM2
35 P3.6/INT2/RxD_2/CMP-/COM3
36 P3.7/INT3/TxD_2/CMP+/SEG31
37 P7.0/SEG35
38 P7.1/SEG34
39 P7.2/SEG33
40 P7.3/PWMETI_3/SEG32
41 P4.1/MISO_3/CMPO_2/PWMETI_2/SEG30
42 P4.2/WR/SEG29
43 P4.3/RxD_4/SCLK_3/SEG28
44 P4.4/RD/TxD_4/SEG27
45 P2.0/A8/PWM1P_2/PWM5/SEG26
46 P2.1/A9/PWM1N_2/PWM6/SEG25
47 P2.2/A10/PWM2P_2/PWM7/SS_2/SEG24
48 P2.3/A11/PWM2N_2/PWM8/MOSI_2/SEG23
SEG22/MISO_2/SDA_2/PWM3P_2/A12/P2.4 49
SEG21/SCLK_2/SCL_2/PWM3N_2/A13/P2.5 50
TK6/SEG20/PWM4P_2/A14/P2.6 51
TK7/SEG19/PWM4N_2/A15/P2.7 52
SEG3/PWM5_4/P7.4 53
SEG2/PWM6_4/P7.5 54
SEG1_2/PWM7_4/P7.6 55
SEG0_2/PWM8_4/P7.7 56
SEG18/ALE/P4.5 57
SEG17/RxD2_2/P4.6 58
TK12/SEG16/T3_2/PWM5_3/RxD3/ADC8/AD0/P0.0 59
TK13/SEG15/T3CLKO_2/PWM6_3/TxD3/ADC9/AD1/P0.1 60
TK14/SEG14/T4_2/PWM7_3/RxD4/ADC10/AD2/P0.2 61
TK15/SEG13/T4CLKO_2/PWM8_3/TxD4/ADC11/AD3/P0.3 62
SEG12/T3/ADC12/AD4/P0.4 63
TK10/SEG11/RxD4_2/P5.2 64
MCU-VCC
22u
0.1u
0.1u

Note:
1. ADC's external reference power supply pin ADC_VRef+ must not be floating. It must be connected to an external reference power supply or directly connected to Vcc.
2. If USB download is not required, P3.0/P3.1/ P3.2 cannot be at low level at the same time when the chip is reset.


LQFP48
QFN48
P2.3/A11/PWM2N_2/PWM8/MOSI_2/SEG23 36
P2.2/A10/PWM2P_2/PWM7/SS_2/SEG24 35
P2.1/A9/PWM1N_2/PWM6/SEG25 34
P2.0/A8/PWM1P_2/PWM5/SEG26 33
P4.4/RD/TxD_4/SEG27 32
P4.3/RxD_4/SCLK_3/SEG28 31
P4.2/WR/SEG29 30
P4.1/MISO_3/CMPO_2/PWMETI_2/SEG30 29
P3.7/INT3/TxD_2/CMP+/SEG31 28
P3.6/INT2/RxD_2/CMP-/COM3 27
P3.5/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2/COM2 26
P5.1/TxD3_2/CMP+_3/COM1/TK9 25
SEG22/MISO_2/SDA_2/PWM3P_2/A12/P2.4 37
SEG21/SCLK_2/SCL_2/PWM3N_2/A13/P2.5 38
TK6/SEG20/PWM4P_2/A14/P2.6 39
TK7/SEG19/PWM4N_2/A15/P2.7 40
SEG18/ALE/P4.5 41
SEG17/RxD2_2/P4.6 42
TK12/SEG16/T3_2/PWM5_3/RxD3/ADC8/AD0/P0.0 43
TK13/SEG15T3CLKO_2/PWM6_3/TxD3/ADC9/AD1/P0.1 44
TK14/SEG14/T4_2/PWM7_3/RxD4/ADC10/AD2/P0.2 45
TK15/SEG13/T4CLKO_2/PWM8_3/TxD4/ADC11/AD3/P0.3 46
SEG12/T3/ADC12/AD4/P0.4 47
TK10/SEG11/RxD4_2/P5.2 48
1 TK11/SEG10/TxD4_2/P5.3
2 SEG9/T3CLKO/ADC13/AD5/P0.5
3 SEG8/PWMETI2_2/T4/ADC14/AD6/P0.6
4 SEG7/T4CLKO/AD7/P0.7
5 TK0/SEG6/RxD2/PWM1P/ADC0/P1.0
6 TK1/SEG5/TxD2/PWM1N/ADC1/P1.1
7 SEG4/TxD2_2/P4.7
8 TK4/SEG1/SDA/MISO/PWM3P/ADC4/P1.4
9 TK5/SEG0/SCL/SCLK/PWM3N/ADC5/P1.5
10 XTALO/MCLKO_2/RxD_3/PWM4P/P1.6
11 XTALI/PWM5_2/TxD_3/PWM4N/P1.7
12 TK3/T2CLKO/MOSI/PWM2N/ADC3/P1.3
24 P5.0/RxD3_2/CMP+_2/COM0/TK8
23 P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO
22 P3.3/INT1/MISO_4/SDA_4/PWM4N_4/PWM7_2
21 P3.2/INT0/SCLK_4/SCL_4/PWMETI/PWMETI2
20 P3.1/TxD
19 P3.0/RxD/INT4
18 P4.0/MOSI_3
17 Gnd/AGnd
16 ADC_VRef+
15 Vcc/AVcc
14 P5.4/NRST/MCLKO/SS/SS_3/PWM2P/PWM6_2/ADC2/T2/TK2/ADC_ETR
13 TCap
MCU-VCC
22u
0.1u
0.1u

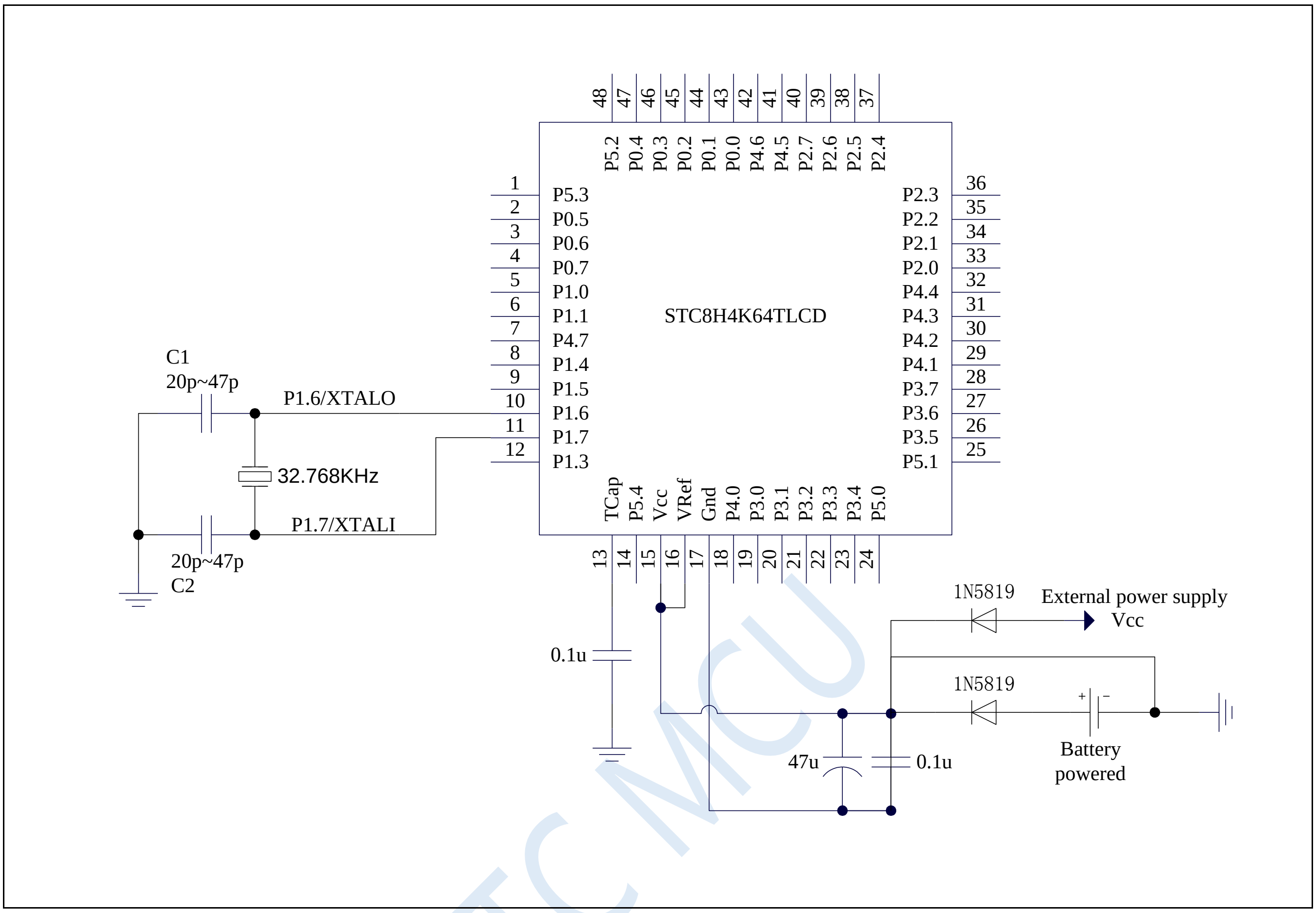
Note:
1. ADC's external reference power supply pin ADC_VRef+ must not be floating. It must be connected to an external reference power supply or directly connected to Vcc.
2. If USB download is not required, P3.0/P3.1/P3.2 cannot be at low level at the same time when the chip is reset.

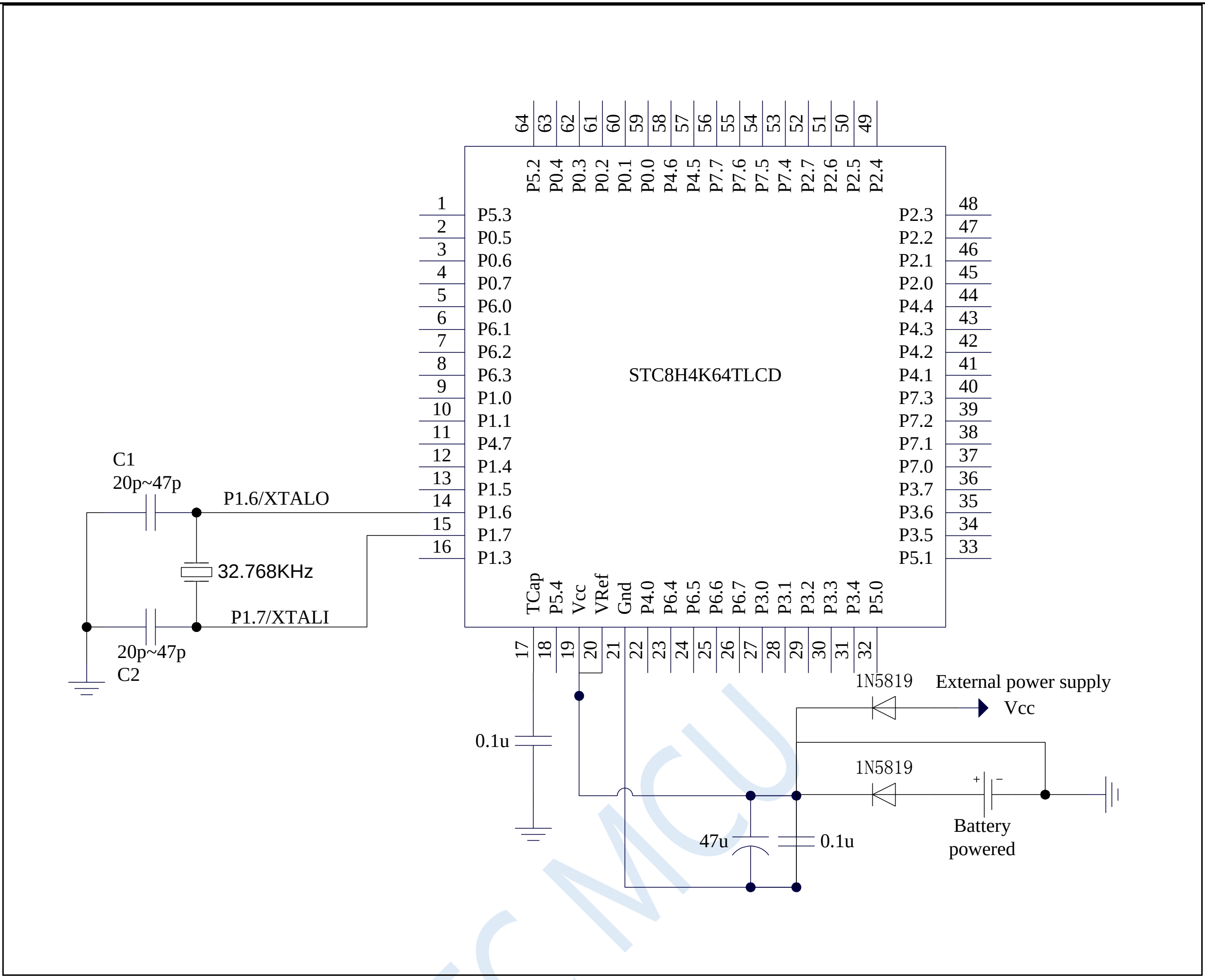
The download steps using ISP and notes are the same as the circumstances in 2.1.2.

Note:

1. **Except for P3.0 and P3.1, all other I/O ports are in high-impedance input state after power-on. User must set the I/O port mode firstly when using I/O.**
2. **All I/O ports can be set to quasi-bidirectional port mode, strong push-pull output mode, open-drain output mode or high-impedance input mode. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.**
3. **When P5.4 is enabled as the reset pin, the reset level is low.**

2.7.3 RTC reference circuit diagram (No VBAT pin)




STC8H4K64TLCD
64 P5.2
63 P0.4
62 P0.3
61 P0.2
60 P0.1
59 P0.0
58 P4.6
57 P4.5
56 P7.7
55 P7.6
54 P7.5
53 P7.4
52 P2.7
51 P2.6
50 P2.5
49 P2.4
1 P5.3
2 P0.5
3 P0.6
4 P0.7
5 P6.0
6 P6.1
7 P6.2
8 P6.3
9 P1.0
10 P1.1
11 P4.7
12 P1.4
13 P1.5
14 P1.6
15 P1.7
16 P1.3
48 P2.3
47 P2.2
46 P2.1
45 P2.0
44 P4.4
43 P4.3
42 P4.2
41 P4.1
40 P7.3
39 P7.2
38 P7.1
37 P7.0
36 P3.7
35 P3.6
34 P3.5
33 P5.1
17 TCap
18 P5.4
19 Vcc
20 VRef
21 Gnd
22 P4.0
23 P6.4
24 P6.5
25 P6.6
26 P6.7
27 P3.0
28 P3.1
29 P3.2
30 P3.3
31 P3.4
32 P5.0
C1
20p~47p
P1.6/XTALO
32.768KHz
P1.7/XTALI
20p~47p
C2
0.1u
1N5819
External power supply
Vcc
1N5819
+
−
Battery
powered
47u
0.1u

2.7.4 Pin descriptions

Pin number		name	type	description
LQFP64/QFN64	LQFP48/QFN48			
1	1	P5.3	I/O	Standard IO port
		TxD4_2	O	Transmit pin of UART 4
		SEG10	O	LCD driver SEG line
		TK11	I	Touch key
2	2	P0.5	I/O	Standard IO port
		AD5	I	Address/data bus
		ADC13	I	ADC analog input 13
		T3CLKO	O	Clock out of timer 3
		SEG9	O	LCD driver SEG line
3	3	P0.6	I/O	Standard IO port
		AD6	I	Address/data bus
		ADC14	I	ADC analog input 14
		T4	I	Timer4 external input
		PWMFLT2_2	I	Enhance PWM external anomaly detection pin 2
		SEG8	O	LCD driver SEG line
4	4	P0.7	I/O	Standard IO port
		AD7	I	Address/data bus
		T4CLKO	O	Clock out of timer 4
		SEG7	O	LCD driver SEG line
5		P6.0	I/O	Standard IO port
		PWM1P_3	I/O	Capture of external signal/Positive of PWMA pulse output
		SEG36	O	LCD driver SEG line
6		P1.1	I/O	Standard IO port
		PWM1N_3	I/O	Capture of external signal/Negative of PWMA pulse output
		SEG37	O	LCD driver SEG line
7		P6.2	I/O	Standard IO port
		ADC6	I	ADC analog input 6
		PWM2P_3	I/O	Capture of external signal/Positive of PWMB pulse output
		SEG38	O	LCD driver SEG line
8		P6.3	I/O	Standard IO port
		ADC7	I	ADC analog input 7
		PWM2N_3	I/O	Capture of external signal/Negative of PWMB pulse output
		SEG39	O	LCD driver SEG line
9	5	P1.0	I/O	Standard IO port
		ADC0	I	ADC analog input 0
		PWM1P	I/O	Capture of external signal/ Positive of PWMA pulse output
		RxD2	I	Input of UART2
		SEG6	O	LCD driver SEG line
		TK0	I	Touch key
10	6	P1.1	I/O	Standard IO port
		ADC1	I	ADC analog input 1
		PWM1N	I/O	Capture of external signal/ Negative of PWMB pulse output
		TxD2	I	Input of UART 2
		SEG5	O	LCD driver SEG line
		TK1	I	Touch key
11	7	P4.7	I/O	Standard IO port
		TxD2_2	I	Transmit pin of UART 2
		SEG4	O	LCD driver SEG line

Pin number		name	type	description
LQFP64/QFN64	LQFP48/QFN48			
12	8	P1.4	I/O	Standard IO port
		ADC4	I	ADC analog input 4
		PWM3P	I/O	Capture of external signal/Positive of PWM3 pulse output
		MISO	I/O	Master Input /Slave Output of SPI
		SDA	I/O	Serial data line of I2C
		SEG1	O	LCD driver SEG line
		TK4	I	Touch key
13	9	P1.5	I/O	Standard IO port
		ADC5	I	ADC analog input 5
		PWM3N	I/O	Capture of external signal/Negative of PWM3 pulse output
		SCLK	I/O	Serial Clock of SPI
		SCL	I/O	Serial Clock line of I2C
		SEG0	O	LCD driver SEG line
		TK5	I	Touch key
14	10	P1.6	I/O	Standard IO port
		RxD_3	I	Input of UART1
		PWM4P	I/O	Capture of external signal/Positive of PWM4 pulse output
		MCLKO_2	O	Main clock output
		XTALO	O	Connect to external oscillator
15	11	P1.7	I/O	Standard IO port
		TxD_3	O	Transmit pin of UART 1
		PWM4N	I/O	Capture of external signal/Negative of PWM4 pulse output
		PWM5_2	I/O	Capture of external signal/Pulse output of PWM5
		XTALI	I	Connect to external oscillator
16	12	P1.3	I/O	Standard IO port
		ADC3	I	ADC analog input 3
		MOSI	I/O	Master Output/Slave Input of SPI
		PWM2N	I/O	Capture of external signal/Negative of PWM2 pulse output
		T2CLKO	O	Clock out of timer 2
		TK3	I	Touch key
17	13	TCAP	I	Charge and discharge capacitance of Touch key
18	14	P5.4	I/O	Standard IO port
		NRST	I	Reset pin
		MCLKO	O	Main clock output
		SS_3	I	Slave selection of SPI (it is output with regard to master)
		SS	I	Slave selection of SPI (it is output with regard to master)
		PWM2P	I/O	Capture of external signal/Positive of PWM2 pulse output
		PWM6_2	I/O	Capture of external signal/Pulse output of PWM6
		T2	I	Timer2 external input
		ADC2	I	ADC analog input 2
		TK2	I	Touch key
		ADC_ETR	I	ADC external trigger pin
19	15	Vcc	Vcc	Power Supply
		AVcc	Vcc	ADC Power Supply
20	16	ADC_VRef+	I	Reference voltage pin of ADC, which can be directly connected to the VCC of the MCU when the requirements are not high
21	17	Gnd	Gnd	Ground
		AGnd	Gnd	ADC Ground

Pin number		name	type	description
LQFP64/QFN64	LQFP48/QFN48			
22	18	P4.0	I/O	Standard IO port
		MOSI_3	I/O	Master Output/Slave Input of SPI

23		P6.4	I/O	Standard IO port
		PWM3P_3	I/O	Capture of external signal/Positive of PWM3 pulse output
24		P6.5	I/O	Standard IO port
		PWM3N_3	I/O	Capture of external signal/Negative of PWM3 pulse output
25		P6.6	I/O	Standard IO port
		PWM4P_3	O	Capture of external signal/Positive of PWM4 pulse output
26		P6.7	I/O	Standard IO port
		PWM4N_3	I/O	Capture of external signal/Negative of PWM4 pulse output
27	19	P3.0	I/O	Standard IO port
		RxD	I	Input of UART1
		INT4	I	External interrupt 4
28	20	P3.1	I/O	Standard IO port
		TxD	O	Transmit pin of UART 1
29	21	P3.2	I/O	Standard IO port
		INT0	I	External interrupt0
		SCLK_4	I/O	Serial Clock of SPI
		SCL_4	I/O	Serial Clock line of I2C
		PWMETI	I	PWM external trigger input pin
		PWMET2	I	PWM external trigger input pin 2
30	22	P3.3	I/O	Standard IO port
		INT1	I	External interrupt1
		MISO_4	I/O	Master Iutput/Slave Onput of SPI
		SDA_4	I/O	Serial data line of I2C
		PWM4N_4	I/O	Capture of external signal/Negative of PWM4 pulse output
		PWM7_2	I/O	Capture of external signal/Pulse output of PWM7
31	23	P3.4	I/O	Standard IO port
		T0	I	Timer0 external input
		T1CLKO	O	Clock out of timer 1
		MOSI_4	I/O	Master Output/Slave Input of SPI
		PWM4P_4	I/O	Capture of external signal/Positive of PWM4 pulse output
		PWM8_2	I/O	Capture of external signal/Pulse output of PWM8
		CMPO	O	Output of comparator
32	24	P5.0	I/O	Standard IO port
		RxD3_2	I	Iutput of UART 3
		CMP+_2	I	Positive input of comparator
		COM0	O	LCD driver COM line
		TK8	I	Touch key
33	25	P5.1	I/O	Standard IO port
		TxD3_2	O	Transmit pin of UART 3
		CMP+_3	I	Positive input of comparator
		COM1	O	LCD driver COM line
		TK9	I	Touch key

Pin number		name	type	description
LQFP64/QFN64	LQFP48/QFN48			
34	26	P3.5	I/O	Standard IO port
		T1	I	Timer1 external input
		T0CLKO	O	Clock out of timer 0
		SS_4	I	Slave selection of SPI (it is output with regard to master)
		PWMFLT	I	Enhance PWMA external anomaly detection pin
		PWMFLT2	I	Enhance PWMB external anomaly detection pin
		COM2	O	LCD driver COM line
35	27	P3.6	I/O	Standard IO port
		INT2	I	External interrupt2
		RxD_2	I	Input of UART1

		CMP-	I	Negative input of comparator
		COM3	O	LCD driver COM line
36	28	P3.7	I/O	Standard IO port
		INT3	I	External interrupt3
		TxD_2	O	Transmit pin of UART 1
		CMP+	I	Positive input of comparator
		SEG31	O	LCD driver SEG line
37		P7.0	I/O	Standard IO port
		SEG35	O	LCD driver SEG line
38		P7.1	I/O	Standard IO port
		SEG34	O	LCD driver SEG line
39		P7.2	I/O	Standard IO port
		SEG33	O	LCD driver SEG line
40		P7.3	I/O	Standard IO port
		PWMETI_3	I	Enhance PWMA external trigger input pin
		SEG32	O	LCD driver SEG line
41	29	P4.1	I/O	Standard IO port
		MISO_3	I/O	Master Iutput/Slave Onput of SPI
		CMPO_2	O	Output of comparator
		PWMETI_2	I	PWM external trigger input pin
		SEG30	O	LCD driver SEG line
42	30	P4.2	I/O	Standard IO port
		WR	O	WRITE signal of external bus
		SEG29	O	LCD driver SEG line
43	31	P4.3	I/O	Standard IO port
		RxD_4	I	Input of UART1
		SCLK_3	I/O	Serial Clock of SPI
		SEG28	O	LCD driver SEG line
44	32	P4.4	I/O	Standard IO port
		RD	O	READ signal of external bus
		TxD_4	O	Transmit pin of UART 1
		SEG27	O	LCD driver SEG line
45	33	P2.0	I/O	Standard IO port
		A8	I	Address bus
		PWM1P_2	I/O	Capture of external signal/Positive of PWMA pulse output
		PWM5	I/O	Capture of external signal/Pulse output of PWM5
		SEG26	O	LCD driver SEG line
46	34	P2.1	I/O	Standard IO port
		A9	I	Address bus
		PWM1N_2	I/O	Capture of external signal/Negative of PWMA pulse output
		PWM6	I/O	Capture of external signal/Pulse output of PWM6
		SEG25	O	LCD driver SEG line

Pin number		name	type	description
LQFP64/QFN64	LQFP48/QFN48			
47	35	P2.2	I/O	Standard IO port
		A10	I	Address bus
		SS_2	I	Slave selection of SPI (it is output with regard to master)
		PWM2P_2	I/O	Capture of external signal/Positive of PWMB pulse output
		PWM7	I/O	Capture of external signal/Pulse output of PWM7
		SEG24	O	LCD driver SEG line
48	36	P2.3	I/O	Standard IO port
		A11	I	Address bus
		MOSI_2	I/O	Master Output/Slave Input of SPI
		PWM2N_2	I/O	Capture of external signal/Negative of PWMB pulse output
		PWM8	I/O	Capture of external signal/Pulse output of PWM8

		SEG23	O	LCD driver SEG line
49	37	P2.4	I/O	Standard IO port
		A12	I	Address bus
		MISO_2	I/O	Master Iutput/Slave Onput of SPI
		SDA_2	I/O	Serial data line of I2C
		PWM3P_2	I/O	Capture of external signal/Positive of PWM3 pulse output
		SEG22	O	LCD driver SEG line
50	38	P2.5	I/O	Standard IO port
		A13	I	Address bus
		SCLK_2	I/O	Serial Clock of SPI
		SCL_2	I/O	Serial Clock line of I2C
		PWM3N_2	I/O	Capture of external signal/Negative of PWM3 pulse output
		SEG21	O	LCD driver SEG line
51	39	P2.6	I/O	Standard IO port
		A14	I	Address bus
		PWM4P_2	I/O	Capture of external signal/Positive of PWM4 pulse output
		SEG20	O	LCD driver SEG line
		TK6	I	Touch key
52	40	P2.7	I/O	Standard IO port
		A15	I	Address bus
		PWM4N_2	I/O	Capture of external signal/Negative of PWM4 pulse output
		SEG19	O	LCD driver SEG line
		TK7	I	Touch key
53		P7.4	I/O	Standard IO port
		PWM5_4	I/O	Capture of external signal/Pulse output of PWM5
		SEG3	O	LCD driver SEG line
54		P7.5	I/O	Standard IO port
		PWM6_4	I/O	Capture of external signal/Pulse output of PWM6
		SEG2	O	LCD driver SEG line
55		P7.6	I/O	Standard IO port
		PWM7_4	I/O	Capture of external signal/Pulse output of PWM7
		SEG1_2	O	LCD driver SEG line
56		P7.7	I/O	Standard IO port
		PWM8_4	I/O	Capture of external signal/Pulse output of PWM8
		SEG0_2	O	LCD driver SEG line
57	41	P4.5	I/O	Standard IO port
		ALE	O	Address Latch Enable signal
		SEG18	O	LCD driver SEG line

Pin number		name	type	description
LQFP64/QFN64	LQFP48/QFN48			
58	42	P4.6	I/O	Standard IO port
		RxD2_2	I	Input of UART2
		SEG17	O	LCD driver SEG line
59	43	P0.0	I/O	Standard IO port
		AD0	I	Address/data bus
		ADC8	I	ADC analog input 8
		RxD3	I	Input of UART3
		PWM5_3	I/O	Capture of external signal/Pulse output of PWM5
		T3_2	I	Timer3 external input
		SEG16	O	LCD driver SEG line
		TK12	I	Touch key
60	44	P0.1	I/O	Standard IO port
		AD1	I	Address/data bus
		ADC9	I	ADC analog input 9
		TxD3	O	Transmit pin of UART 3

		PWM6_3	I/O	Capture of external signal/Pulse output of PWM6
		T3CLKO_2	O	Clock out of timer 3
		SEG15	O	LCD driver SEG line
		TK13	I	Touch key
61	45	P0.2	I/O	Standard IO port
		AD2	I	Address/data bus
		ADC10	I	ADC analog input 10
		RxD4	I	Input of UART4
		PWM7_3	I/O	Capture of external signal/Pulse output of PWM7
		T4_2	I	Timer4 external input
		SEG14	O	LCD driver SEG line
		TK14	I	Touch key
62	46	P0.3	I/O	Standard IO port
		AD3	I	Address/data bus
		ADC11	I	ADC analog input 11
		TxD4	O	Transmit pin of UART 4
		PWM8_3	I/O	Capture o f external signal/Pulse output of PWM8
		T4CLKO_2	O	Clock out of timer 4
		SEG13	O	LCD driver SEG line
		TK15	I	Touch key
63	47	P0.4	I/O	Standard IO port
		AD4	I	Address/data bus
		ADC12	I	ADC analog input 12
		T3	I	Timer3 external input
		SEG12	O	LCD driver SEG line
64	48	P5.2	I/O	Standard IO port
		RxD4_2	I	Input of UART4
		SEG11	O	LCD driver SEG line
		TK10	I	Touch key

2.8 STC8H1K08TR-36I-TSSOP20/QFN20(Touch key/RTC family)

2.6.1 Features and Price(Quasi 16-bit MCU with 16-bit hardware multiplier and divider MDU16)

- Selection and price (No external crystal and external reset required with 15 channels 12-bit ADC)

MCU	Operating voltage（V）	Flash Code Memory (100 thousand times) (Byte)	idata Internal DATA RAM(Byte)	xdata Internal extended SRAM (Byte)	Enhanced Dual DPTR increasing or decreasing	EEPROM 100 thousand times) (Byte)	Maximum I/O Lines	Traditional I/O interrupts (INT0/INT1/INT2/INT3/INT4)	All I/O ports support interrupts and can wake MCU	DMA UARTs which can wake-up CPU	Touch key	RTC	DMA SPI which can wake-up CPU	I^{2}C which can wake-up CPU	MDU16 (Hardware 16-bit Multiplier and Divider)	Timers/Counters (T0-T4 Pin can wake-up CPU)	16-bit advanced PWM timer with Complementary symmetrical dead-time	Power-down Wake-up timer	DMA 15 channels high speed ADC (8 PWMs can be used as 8 DACs)	Comparator (May be used as ADC to detect external power-down)	Internal LVD interrupt (can wake-up CPU)	Watch-dog Timer	Internal high reliable reset circuit with 4 levels optional reset threshold voltage	Internal high presision Clock (adjustbal under 36MHz)	Clock output and Reset	Program encrypted transmission (Anti-blocking)	Password can be set for next update	Support RS485 download	Support software USB download directly	Online debug itself	Price & Package	products supply information

																															TSSOP20	QFN20	
STC8H1K08TR	1.9-5.5	8K	256	1K	2	4K	16	Y	Y	2	Y	Y	Y	Y	Y	3	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	¥2	¥2	Some available
STC8H1K17TR	1.9-5.5	17K	256	1K	2	IAP	16	Y	Y	2	Y	Y	Y	Y	Y	3	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	√	√	

- **Core**
  - ✓ Ultra-high speed 8051 Core with single clock per machine cycle, which is called 1T and the speed is about 12 times faster than traditional 8051
  - ✓ Fully compatible instruction set with traditional 8051
  - ✓ 29 interrupt sources and 4 interrupt priority levels
  - ✓ Online debugging is supported

- **Operating voltage**
  - ✓ 1.9V～5.5V

- **Operating temperature**
  - ✓ -40℃~85℃(The chip is produced in -40℃～125℃ process. Please refer to the description of the electrical characteristics chapter for applications beyond the temperature range)

- **Flash memory**
  - ✓ Up to 17Kbytes of Flash memory to be used for storing user code
  - ✓ Configurable EEPROM size, 512bytes single page for being erased, which can be repeatedly erased more than 100 thousand times.
  - ✓ In-System-Programming, ISP in short, can be used to update the application code. No special programmer is needed.
  - ✓ Online debugging with single chip is supported, and no special emulator is needed. The number of breakpoints is unlimited theoratically.

- **SRAM**
  - ✓ 128 bytes internal direct access RAM (DATA, use keyword *data* to declare in C language program)
  - ✓ 128 bytes internal indirect access RAM (IDATA, use keyword *idata* to declare in C language program)
  - ✓ 1024 bytes internal extended RAM (internal XDATA, use keyword *xdata* to declare in C language program)

- **Clock**
  - ✓ Internal high precise RC clock IRC(IRC for short, ranges from 4MHz to 36MHz), adjustable while ISP and can be divided to lower frequency by user software, 100KHz for instance.
    - ✓ Error:±0.3% (at the temperature 25℃)
    - ✓ -1.35%～+1.30% temperature drift (at the temperature range of -40℃ to +85℃)
    - ✓ -0.76%～+0.98% temperature drift (at the temperature range of -20℃ to 65℃)
  - ✓ Internal 32KHz low speed IRC with large error
  - ✓ External crystal (4MHz～33MHz) and external clock

    Users can freely choose the above 3 clock sources

- **Reset**
  - ✓ Hardware reset
    - ✓ Power-on reset. Measured voltage is 1.69V~1.82V. (Effective when the chip does not enable the low voltage reset function)

      The power-on reset voltage is a voltage range consisting of an upper limit voltage and a lower limit voltage. When the operating voltage drops from 5V / 3.3V to the lower limit threshold voltage of the power-on reset, the chip is in a reset state; when the voltage rises from 0V to the upper threshold voltage of power-on reset, the chip is released from the reset state.
    - ✓ Reset by reset pin. The default function of P5.4 is the I/O port. The P5.4 pin can be set as the reset pin while ISP download. (Note: When the P5.4 pin is set as the reset pin, the reset level is low.)
    - ✓ Watch dog timer reset
    - ✓ Low voltage detection reset. 4 low voltage detection levels are provided, 1.9V, 2.3V, 2.8V, 3.0V. Each level of low-voltage detection voltage is a voltage range consisting of an upper limit voltage and a lower limit voltage. When the operating voltage drops from 5V / 3.3V to the lower limit threshold voltage of low-voltage detection, the low-voltage detection takes effect. When the voltage rises from 0V to the upper threshold voltage, the low voltage detection becomes effective.
  - ✓ Software reset
    - ✓ Writing the reset trigger register using software

- **Interrupts**
  - ✓ 29 interrupt sources: INT0(Supports rising edge and falling edge interrupt), INT1(Supports rising edge and falling edge

interrupt), INT2(Supports falling edge interrupt only), INT3(Supports falling edge interrupt only), INT4(Supports falling edge interrupt only), timer 0, timer 1, timer 2, UART 1, UART 2, ADC, LVD, SPI, I^{2}C, comparator, PWMA, PWMB, RTC, TKS, P1, P3, P5, DMA receive and transmit interrupts of UART 1, DMA receive and transmit interrupts of UART 2, DMA interrupt of SPI, DMA interrupt of ADC and DMA interrupt of memory-to-memory.

- ✓ 4 interrupt priority levels
- ✓ Interrupts that can wake up the CPU in clock stop mode: INT0(P3.2), INT1(P3.3), INT2(P3.6), INT3(P3.7), INT4(P3.0), T0(P3.4), T1(P3.5), T2(P1.2), RXD(P3.0/P3.6/P1.6/P4.3), RXD2(P1.0/P4.6), I2C_SDA(P1.4/P2.4/P3.3), SPI_SS(P5.4/P2.2/P3.5), Comparator interrupt, LVD interrupt, Power-down wake-up timer and interrupts of all I/O ports.

➢ **Digital peripherals**

- ✓ 3 16-bit timers: timer0, timer1, timer2, where the mode 3 of timer 0 has the Non-Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
- ✓ 2 high speed UARTs: UART1, UART2, whose maximum baudrate clock may be FOSC/4
- ✓ 8 channels/2 groups of enhanced PWM, which can realize control signals with dead time, and support external fault detection function. In addition, supports 16-bit timers, 8 external interrupts, 8 external captures and pulse width measurement functions.
- ✓ SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
- ✓ I^{2}C: Master mode or slave mode are supported.
- ✓ MDU16: Hardware 16-bit Multiplier and Divider which supports 32-bit divided by 16-bit, 16-bit divided by 16-bit, 16-bit multiplied by 16-bit, data shift, and data normalization operations.
- ✓ RTC: Support year, month, day, hour, minute, second, sub-second (1/128 second). And supports clock interrupt and a set of alarm clocks (Note: A version of the chip does not have this function)
- ✓ I/O port interrupt: All I/Os support interrupts, each group of I/O interrupts has an independent interrupt entry address, all I/O interrupts can support 4 types interrupt mode: high level interrupt, low level interrupt, rising edge interrupt, falling edge interrupt. Provides 4 levels of interrupt priority and supports power-down wake-up function.
- ✓ DMA: support Memory-To-Memory, SPI, UART1TX/UART1RX, UART2TX/UART2RX, ADC(Automatically calculates the average of multiple ADC results)

➢ **Analog peripherals**

- ✓ Ultra high speed ADC which supports 12-bit precision 15 channels (channel 0 to channel 14) analog-to-digital conversion. The maximum speed can be 800K(800K ADC conversions per second)
- ✓ ADC channel 15 is used to test the internal reference voltage. (The default internal reference voltage is 1.19V when the chip is shipped)
- ✓ Comparator. A set of comparator (The CMP+ port and all ADC input ports can be selected as the positive terminal of the comparator. So the comparator can be used as a multi-channel comparator for time division multiplexing)
- ✓ Touch key: The microcontroller supports up to 16 touch keys. Every touch key can be enabled independently. The internal reference voltage is adjustable with 4 levels. Charge and discharge time settings and internal working frequency settings are flexible. The touch key supports wake-up CPU from low-power mode.
- ✓ DAC: 8 channels advanced PWM timer can be used as 8 channels DAC

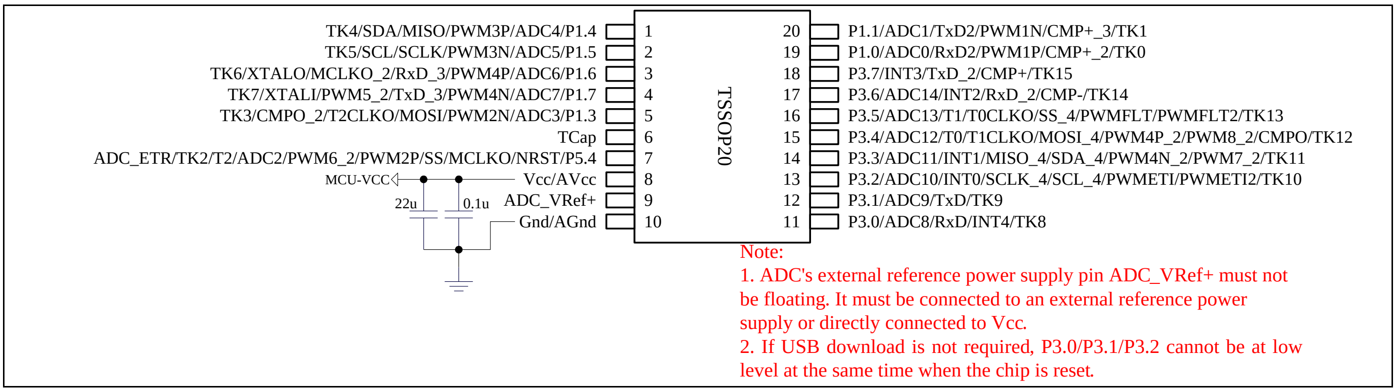
➢ **GPIO**

- ✓ Up to 16 GPIOs: P1.0~P1.7(No P1.2), P3.0~P3.7, P5.4
- ✓ 4 modes for all GPIOs: quasi_bidirectional mode, push-pull outputmode, open drain mode, high-impedance input mode
- ✓ Except for P3.0 and P3.1, all other I/O ports are in a high-impedance state after power-on. User must set the I/O ports mode before using them. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.

➢ **Package**

- ✓ TSSOP20, QFN20

2.8.2 Pinouts


TK4/SDA/MISO/PWM3P/ADC4/P1.4
TK5/SCL/SCLK/PWM3N/ADC5/P1.5
TK6/XTALO/MCLKO_2/RxD_3/PWM4P/ADC6/P1.6
TK7/XTALI/PWM5_2/TxD_3/PWM4N/ADC7/P1.7
TK3/CMPO_2/T2CLKO/MOSI/PWM2N/ADC3/P1.3
TCap
ADC_ETR/TK2/T2/ADC2/PWM6_2/PWM2P/SS/MCLKO/NRST/P5.4
MCU-VCC
Vcc/AVcc
22u
0.1u
ADC_VRef+
Gnd/AGnd
1
2
3
4
5
6
7
8
9
10
TSSOP20
20
19
18
17
16
15
14
13
12
11
P1.1/ADC1/TxD2/PWM1N/CMP+_3/TK1
P1.0/ADC0/RxD2/PWM1P/CMP+_2/TK0
P3.7/INT3/TxD_2/CMP+/TK15
P3.6/ADC14/INT2/RxD_2/CMP-/TK14
P3.5/ADC13/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2/TK13
P3.4/ADC12/T0/T1CLKO/MOSI_4/PWM4P_2/PWM8_2/CMPO/TK12
P3.3/ADC11/INT1/MISO_4/SDA_4/PWM4N_2/PWM7_2/TK11
P3.2/ADC10/INT0/SCLK_4/SCL_4/PWMETI/PWMETI2/TK10
P3.1/ADC9/TxD/TK9
P3.0/ADC8/RxD/INT4/TK8

Note:
1. ADC's external reference power supply pin ADC_VRef+ must not be floating. It must be connected to an external reference power supply or directly connected to Vcc.
2. If USB download is not required, P3.0/P3.1/P3.2 cannot be at low level at the same time when the chip is reset.

The download steps using ISP and notes are the same as the circumstances in 2.1.2.

2.8.3 Pin descriptions

Pin number		name	type	description
TSSOP20	QFN20			
1	18	P1.4	I/O	Standard IO port
		ADC4	I	ADC analog input 4
		PWM3P	I/O	Capture of external signal/Positive of PWM3 pulse output
		MISO	I/O	Master Iutput/Slave Onput of SPI
		SDA	I/O	Serial data line of I2C
		TK4	I	Touch key
2	19	P1.5	I/O	Standard IO port
		ADC5	I	ADC analog input 5
		PWM3N	I/O	Capture of external signal/Negative of PWM3 pulse output
		SCLK	I/O	Serial Clock of SPI
		SCL	I/O	Serial Clock line of I2C
		TK5	I	Touch key
3	20	P1.6	I/O	Standard IO port
		ADC6	I	ADC analog input 6
		RxD_3	I	Input of UART 1
		PWM4P	I/O	Capture of external signal/Positive of PWM4 pulse output
		MCLKO_2	O	Main clock output
		XTALO	O	Connect to external oscillator
		TK6	I	Touch key
4	1	P1.7	I/O	Standard IO port
		ADC7	I	ADC analog input 7
		TxD_3	O	Transmit pin of UART 1
		PWM4N	I/O	Capture of external signal/Negative of PWM4 pulse output
		PWM5_2	I/O	Capture of external signal/Pulse output of PWM5
		XTALI	I	Connect to external oscillator
		TK7	I	Touch key
5	2	P1.3	I/O	Standard IO port
		ADC3	I	ADC analog input 3
		MOSI	I/O	Master Output/Slave Input of SPI
		PWM2N	I/O	Capture of external signal/Negative of PWM2 pulse output
		T2CLKO	O	Clock out of timer 2
		CMPO_2	O	Output of comparator
		TK7	I	Touch key
6	3	TCAP	I	Charge and discharge capacitance of Touch key
7	4	P5.4	I/O	Standard IO port
		NRST	I	Reset pin (low level reset)
		MCLKO	O	Main clock output
		SS	I	Slave selection of SPI (it is output with regard to master)
		PWM2P	I/O	Capture of external signal/Positive of PWM2 pulse output
		PWM6_2	I/O	Capture of external signal/Pulse output of PWM6
		T2	I	Timer2 external input
		ADC2	I	ADC analog input 2
		TK2	I	Touch key
		ADC_ETR	I	ADC external trigger pin
8	5	Vcc	Vcc	Power Supply
		AVcc	Vcc	ADC Power Supply

Pin number		name	type	description
TSSOP20	QFN20			
9	6	ADC_VRef+	I	Reference voltage pin of ADC, which can be directly connected to the VCC of the MCU when the requirements are not high
10	7	Gnd	Gnd	Ground
		AGnd	Gnd	ADC Ground
11	8	P3.0	I/O	Standard IO port
		RxD	I	Input of UART1
		INT4	I	External interrupt4
		ADC8	I	ADC analog input 8
		TK8	I	Touch key
12	9	P3.1	I/O	Standard IO port
		TxD	O	Transmit pin of UART 1
		ADC9	I	ADC analog input 9
		TK9	I	Touch key
13	10	P3.2	I/O	Standard IO port
		INT0	I	External interrupt0
		SCLK_4	I/O	Serial Clock of SPI
		SCL_4	I/O	Serial Clock line of I2C
		PWMETI	I	PWM external trigger input pin
		PWMET2	I	PWM external trigger input pin 2
		ADC10	I	ADC analog input 10
		TK10	I	Touch key
14	11	P3.3	I/O	Standard IO port
		INT1	I	External interrupt1
		MISO_4	I/O	Master Iutput/Slave Onput of SPI
		SDA_4	I/O	Serial data line of I2C
		PWM4N_4	I/O	Capture of external signal/Negative of PWM4 pulse output
		PWM7_2	I/O	Capture of external signal/Pulse output of PWM7
		ADC11	I	ADC analog input 11
		TK11	I	Touch key
15	12	P3.4	I/O	Standard IO port
		T0	I	Timer0 external input
		T1CLKO	O	Clock out of timer 1
		MOSI_4	I/O	Master Output/Slave Input of SPI
		PWM4P_4	I/O	Capture of external signal/Positive of PWM4 pulse output
		PWM8_2	I/O	Capture of external signal/Pulse output of PWM8
		CMPO	O	Output of comparator
		ADC12	I	ADC analog input 12
		TK12	I	Touch key
16	13	P3.5	I/O	Standard IO port
		T1	I	Timer1 external input
		T0CLKO	O	Clock out of timer 0
		SS_4	I	Slave selection of SPI (it is output with regard to master)
		PWMFLT	I	Enhance PWMA external anomaly detection pin
		PWMFLT2	I	Enhance PWMB external anomaly detection pin
		ADC13	I	ADC analog input 13
		TK13	I	Touch key

Pin number		name	type	description
TSSOP20	QFN20			
17	14	P3.6	I/O	Standard IO port

		INT2	I	External interrupt2
		RxD_2	I	Input of UART1
		CMP-	I	Negative input of comparator
		ADC14	I	ADC analog input 14
		TK14	I	Touch key
18	15	P3.7	I/O	Standard IO port
		INT3	I	External interrupt3
		TxD_2	O	Transmit pin of UART 1
		CMP+	I	Positive input of comparator
		TK15	I	Touch key
19	16	P1.0	I/O	Standard IO port
		ADC0	I	ADC analog input 0
		PWM1P	I/O	Capture of external signal/ Positive of PWMA pulse output
		RxD2	I	Input of UART2
		TK0	I	Touch key
		CMP+_2	I	Positive input of comparator
20	17	P1.1	I/O	Standard IO port
		ADC1	I	ADC analog input 1
		PWM1N	I/O	Capture of external signal/ Negative of PWMA pulse output
		TxD2	I	Transmit pin of UART 2
		TK1	I	Touch key
		CMP+_3	I	Positive input of comparator

2.9 STC32G12K128-LQFP64/48/32,PDIP40 (New product notice)

2.9.1 Features and Price

- **Selection and price (No external crystal and external reset required with 15 channels 12-bit ADC)**

MCU	Operating voltage（V）	Flash Code Memory (100 thousand times) (Byte)	edata Internal extended DATA RAM which can be used as stack(Byte)	xdata Internal extended SRAM (Byte)	Enhanced Dual DPTR increasing or decreasing	EEPROM 100 thousand times) (Byte)	Maximum I/O Lines	Traditional I/O interrupt(INT0/INT1/INT2/INT3/INT4) (can wake-up CPU)	All I/O ports support interrupts and can wake up MCU	DMA 8080/6800 interface/ LCM driver(8-bit and 16-bit)	RTC	DMA UARTs which can wake-up CPU	DMA USARTs which can wake-up CPU	CAN	LIN	Full-spped USB	DMA SPI which can wake-up CPU	I^{2}C which can wake-up CPU	MDU32 (Hardware 32-bit Multiplier and Divider)	Timers/Counters (T0-T4 pin Ccn wake-up CPU)	16-bit advanced PWM timer with Complementary symmetrical dead-time	Power-down Wake-up timer	DMA 15 channels high speed ADC (8 PWMs can be used as 8 DACs)	Comparator (May be used as ADC to detect external power-down)	Internal LVD interrupt (can wake-up CPU)	Watch-dog Timer	Internal high reliable reset circuit with 4 levels optional reset threshold voltage	Internal high presision Clock (adjustbal under 33MHz)	Clock output and Reset	Program encrypted transmission (Anti-blocking)	Password can be set for next update	Support RS485 download	Support hardware USB download and debug directly	Online debug itself	Price & Package				products supply information
																																			LQFP64 <12mm*12mm>	LQFP48 <9mm*9mm>	LQFP32<9mm*9mm>	PDIP40	
STC32G12K64	1.9-5.5	64K	4K	8K	2	64K	60	Y	Y	Y	Y	2	2	2	Y	Y	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	¥5	¥5	√	√	Some availa
STC32G12K128	1.9-5.5	128K	4K	8K	2	IAP	60	Y	Y	Y	Y	2	2	2	Y	Y	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	¥5	¥5	√	√	

- **Core**
  - ✓ Ultra-high speed 32-bit 8051 Core with single clock per machine cycle, which is called 1T and the speed is about 70 times faster than traditional 8051
  - ✓ 49 interrupt sources and 4 interrupt priority levels
  - ✓ Online debugging is supported

- **Operating voltage**
  - ✓ 1.9V～5.5V
  - ✓ Built-in LDO

- **Operating temperature**
  - ✓ -40℃~85℃

- **Flash memory**
  - ✓ Up to 128Kbytes of Flash memory to be used for storing user code
  - ✓ Configurable EEPROM size, 512bytes single page for being erased, which can be repeatedly erased more than 100 thousand times.
  - ✓ In-System-Programming, ISP in short, can be used to update the application code. No special programmer is needed.
  - ✓ Online debugging with single chip is supported, and no special emulator is needed. The number of breakpoints is unlimited theoratically.
  - ✓ Support hardware emulation of SWD interface (requires STC-USB Link1 tool)

- **SRAM**
  - ✓ 4K bytes internal SRAM (EDATA)
  - ✓ 8K bytes internal extended RAM (internal XDATA

    Notes on using xdata:

    When defining variables, single-byte variables can be defined in xdata, and multi-byte (2-byte, 4-byte) variables need to be defined in edata.

- **Clock**

- Internal high precise RC clock IRC(IRC for short), adjustable while ISP and can be divided to lower frequency by user software.
  - Error:±0.3% (at the temperature 25℃)
  - -1.35%～+1.30% temperature drift (at the temperature range of -40℃ to +85℃)
  - -0.76%～+0.98% temperature drift (at the temperature range of -20℃ to 65℃)
- Internal 32KHz low speed IRC with large error
- External crystal (4MHz～33MHz) and external clock
- Internal PLL output clock

  Users can freely choose the above 4 clock sources

- **Reset**
  - Hardware reset
    - Power-on reset. Measured voltage is 1.7V~1.9V. (Effective when the chip does not enable the low voltage reset function)
    - Reset by reset pin. The default function of P5.4 is the I/O port. The P5.4 pin can be set as the reset pin while ISP download. (Note: When the P5.4 pin is set as the reset pin, the reset level is low.)
    - Watch dog timer reset
    - Low voltage detection reset. 4 low voltage detection levels are provided, 2.0V, 2.3V, 2.7V, 3.0V.
  - Software reset
    - Writing the reset trigger register using software

- **Interrupts**
  - 49 interrupt sources: INT0, INT1, INT2, INT3, INT4, timer 0, timer 1, timer 2, timer 3, timer 4, USART1, USART2, UART 3, UART 4, ADC, LVD, SPI, I^2C, comparator, PWMA, PWMB, USB, CAN, CAN2, LIN, LCMIF color screen interface, RTC, all I/O interrupts (8 groups), DMA receive and transmit interrupts of USART 1, DMA receive and transmit interrupts of USART 2, DMA receive and transmit interrupts of UART 3, DMA receive and transmit interrupts of UART 4, DMA interrupt of I2C, DMA interrupt of SPI, DMA interrupt of ADC, DMA interrupt of LCD driver and DMA interrupt of memory-to-memory.
  - 4 interrupt priority levels

- **Digital peripherals**
  - 5 16-bit timers: timer0, timer1, timer2, timer3, timer4, where the mode 3 of timer 0 has the Non-Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
  - 2 high speed USARTs: USART1, USART2, whose maximum baudrate clock may be FOSC/4. The following modes are supported, synchronous serial port mode, asynchronous serial port mode, SPI mode, LIN mode, infrared mode (IrDA), smart card mode (ISO7816)
  - 2 high speed UARTs: UART3, UART4, whose maximum baudrate clock may be FOSC/4
  - 2 groups of enhanced PWM, which can realize 8 channels(4 groups complementary symmetry) control signals with dead time, and support external fault detection function.
  - SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
  - I^2C: Master mode or slave mode are supported.
  - ICE: Hardware support emulation.
  - RTC: Support year, month, day, hour, minute, second, sub-second (1/128 second). And supports clock interrupt and a set of alarm clocks.
  - USB: USB2.0/USB1.1 compatible with full-speed USB, 6 bidirectional endpoints, support 4 endpoint transfer modes (control transfer, interrupt transfer, bulk transfer and isochronous transfer), each endpoint has a 64-byte buffer.
  - CAN: Two independent CAN 2.0 control units.
  - LIN: An independent LIN control unit (supports versions 1.3 and 2.1), USART1 and USART2 can support two sets of LIN.
  - MDU32: Hardware 32-bit Multiplier and Divider which supports 32-bit divided by 32-bit, 32-bit multiplied by 32-bit operations.
  - I/O port interrupt: All I/Os support interrupts, each group of I/O interrupts has an independent interrupt entry address, all I/O interrupts can support 4 types interrupt mode: high level interrupt, low level interrupt, rising edge interrupt, falling edge interrupt. Provides 4 levels of interrupt priority and supports power-down wake-up function.
  - LCD dirver: support 8080 and 6800 interface, and support 8-bit and 16-bit data width.
  - DMA: Support SPI shift to receive data to memory, SPI shift to send data from memory, I2C send data from memory, I2C receive data to memory, USART 1/2 and UART 3/4 receive data to memory, USART 1/2 and UART 3/4 send data from memory, ADC automatically sample data to memory (calculate average value at the same time), LCD driver send data from memory, and copy data from memory to memory
  - Hardware digital ID: support 32 bytes

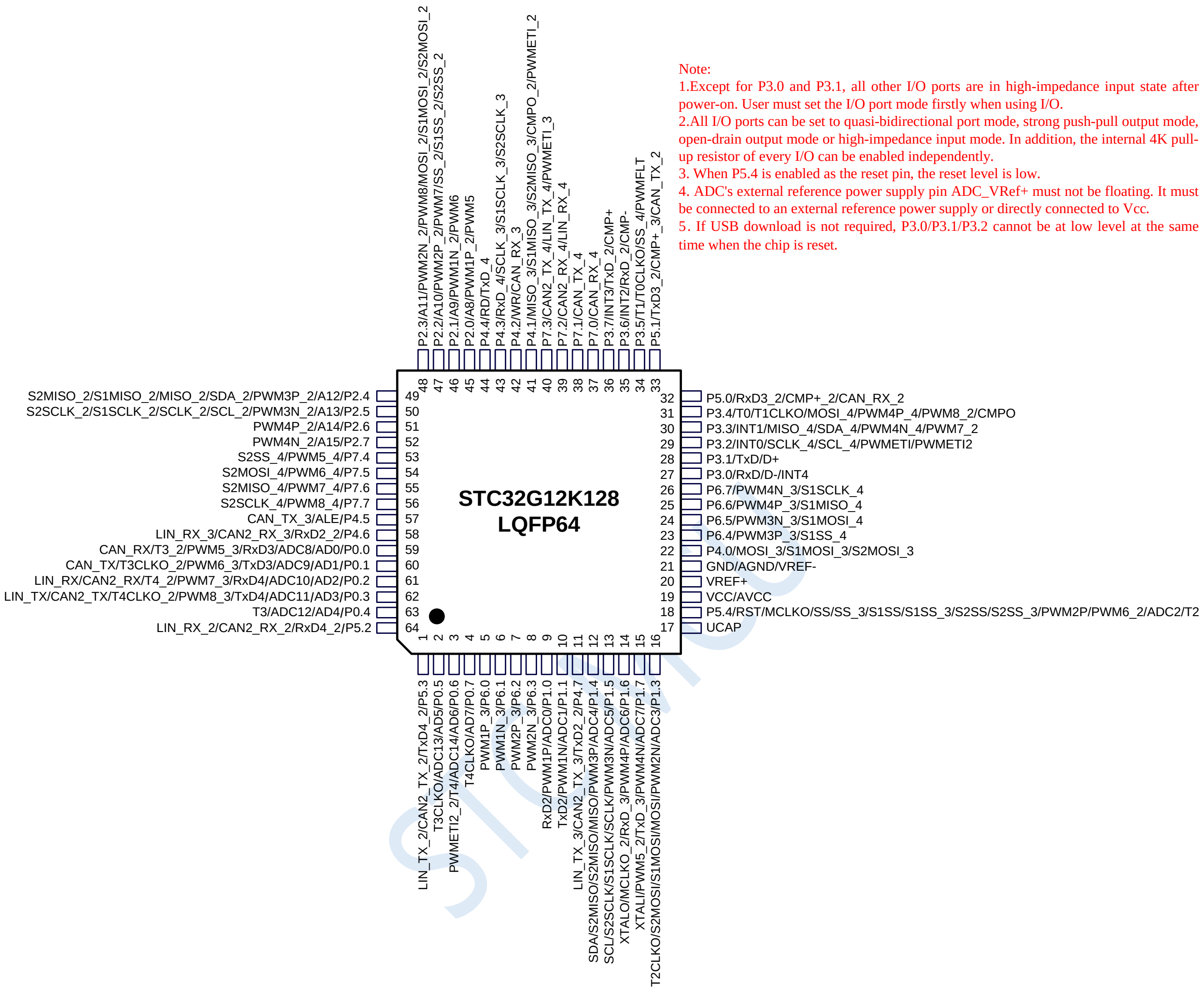
- **Analog peripherals**
  - Ultra high speed ADC which supports 12-bit precision 15 channels (channel 0 to channel 14) analog-to-digital conversion. ADC channel 15 is used to test the internal reference voltage. (The default internal reference voltage is 1.19V when the chip is shipped, the error is ±1%)
  - Comparator. A set of comparator

- **GPIO**
  - Up to 60 GPIOs: P0.0~P0.7, P1.0~P1.7(No P1.2), P2.0~P2.7, P3.0~P3.7, P4.0~P4.7, P5.0~P5.4, P6.0~P6.7, P7.0~P7.7
  - 4 modes for all GPIOs: quasi_bidirectional mode, push-pull outputmode, open drain mode, high-impedance input mode
  - Except for P3.0 and P3.1, all other I/O ports are in a high-impedance state after power-on. User must set the I/O ports mode before using them. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.

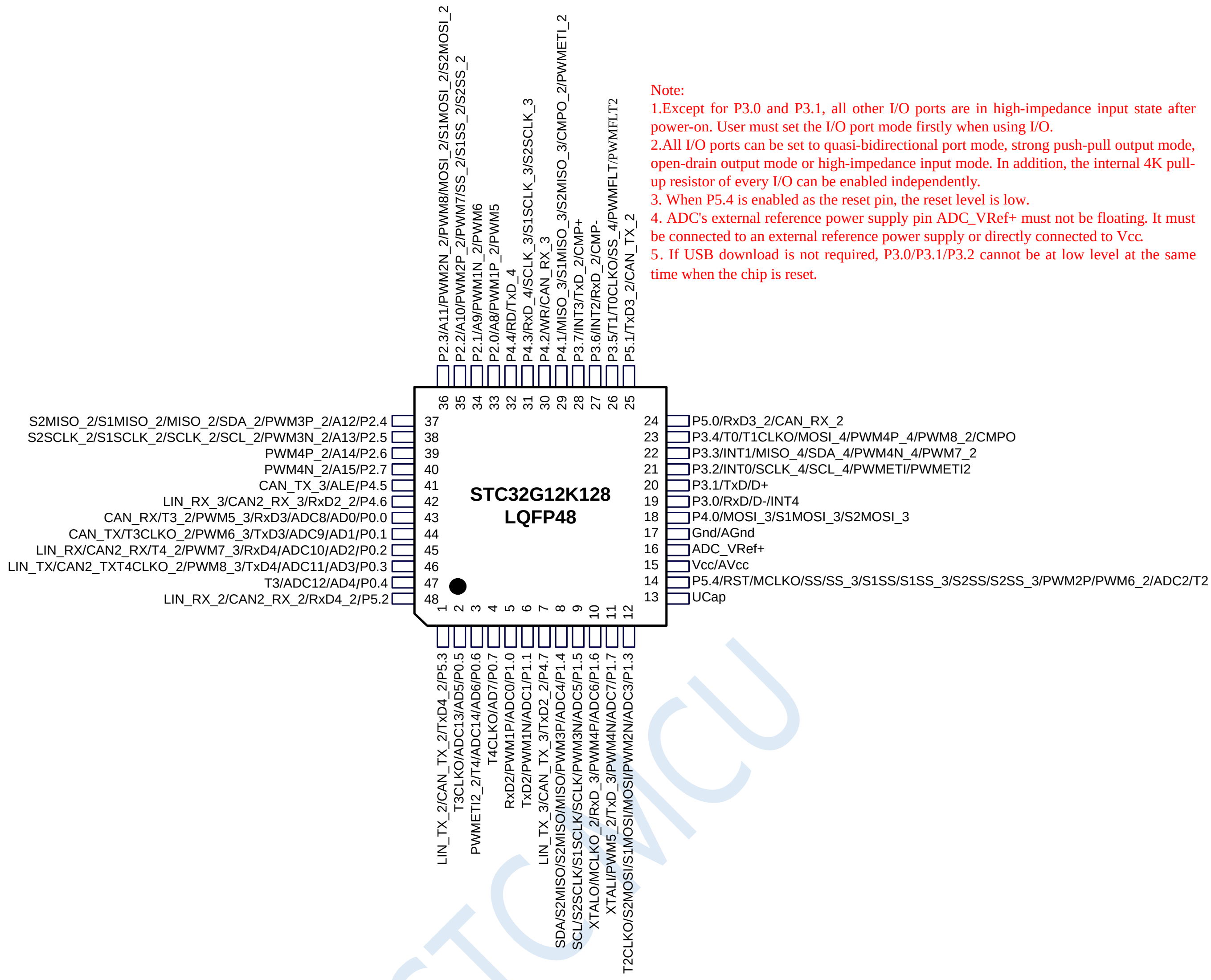
- **Package**

- ✓ LQFP64, LQFP48, LQFP32, PDIP40

2.9.2 Pinouts


STC32G12K128
LQFP64
1 LIN_TX_2/CAN2_TX_2/TxD4_2/P5.3
2 T3CLKO/ADC13/AD5/P0.5
3 PWMETI2_2/T4/ADC14/AD6/P0.6
4 T4CLKO/AD7/P0.7
5 PWM1P_3/P6.0
6 PWM1N_3/P6.1
7 PWM2P_3/P6.2
8 PWM2N_3/P6.3
9 RxD2/PWM1P/ADC0/P1.0
10 TxD2/PWM1N/ADC1/P1.1
11 LIN_TX_3/CAN2_TX_3/TxD2_2/P4.7
12 SDA/S2MISO/S2MISO/MISO/PWM3P/ADC4/P1.4
13 SCL/S2SCLK/S1SCLK/SCLK/PWM3N/ADC5/P1.5
14 XTALO/MCLKO_2/RxD_3/PWM4P/ADC6/P1.6
15 XTALI/PWM5_2/TxD_3/PWM4N/ADC7/P1.7
16 T2CLKO/S2MOSI/S1MOSI/MOSI/PWM2N/ADC3/P1.3
17 UCAP
18 P5.4/RST/MCLKO/SS/SS_3/S1SS/S1SS_3/S2SS/S2SS_3/PWM2P/PWM6_2/ADC2/T2
19 VCC/AVCC
20 VREF+
21 GND/AGND/VREF-
22 P4.0/MOSI_3/S1MOSI_3/S2MOSI_3
23 P6.4/PWM3P_3/S1SS_4
24 P6.5/PWM3N_3/S1MOSI_4
25 P6.6/PWM4P_3/S1MISO_4
26 P6.7/PWM4N_3/S1SCLK_4
27 P3.0/RxD/D-/INT4
28 P3.1/TxD/D+
29 P3.2/INT0/SCLK_4/SCL_4/PWMETI/PWMETI2
30 P3.3/INT1/MISO_4/SDA_4/PWM4N_4/PWM7_2
31 P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO
32 P5.0/RxD3_2/CMP+_2/CAN_RX_2
33 P5.1/TxD3_2/CMP+_3/CAN_TX_2
34 P3.5/T1/T0CLKO/SS_4/PWMFLT
35 P3.6/INT2/RxD_2/CMP-
36 P3.7/INT3/TxD_2/CMP+
37 P7.0/CAN_RX_4
38 P7.1/CAN_TX_4
39 P7.2/CAN2_RX_4/LIN_RX_4
40 P7.3/CAN2_TX_4/LIN_TX_4/PWMETI_3
41 P4.1/MISO_3/S1MISO_3/S2MISO_3/CMPO_2/PWMETI_2
42 P4.2/WR/CAN_RX_3
43 P4.3/RxD_4/SCLK_4/S1SCLK_3/S2SCLK_3
44 P4.4/RD/TxD_4
45 P2.0/A8/PWM1P_2/PWM5
46 P2.1/A9/PWM1N_2/PWM6
47 P2.2/A10/PWM2P_2/PWM7/SS_2/S1SS_2/S2SS_2
48 P2.3/A11/PWM2N_2/PWM8/MOSI_2/S1MOSI_2/S2MOSI_2
49 S2MISO_2/S1MISO_2/MISO_2/SDA_2/PWM3P_2/A12/P2.4
50 S2SCLK_2/S1SCLK_2/SCLK_2/SCL_2/PWM3N_2/A13/P2.5
51 PWM4P_2/A14/P2.6
52 PWM4N_2/A15/P2.7
53 S2SS_4/PWM5_4/P7.4
54 S2MOSI_4/PWM6_4/P7.5
55 S2MISO_4/PWM7_4/P7.6
56 S2SCLK_4/PWM8_4/P7.7
57 CAN_TX_3/ALE/P4.5
58 LIN_RX_3/CAN2_RX_3/RxD2_2/P4.6
59 CAN_RX/T3_2/PWM5_3/RxD3/ADC8/AD0/P0.0
60 CAN_TX/T3CLKO_2/PWM6_3/TxD3/ADC9/AD1/P0.1
61 LIN_RX/CAN2_RX/T4_2/PWM7_3/RxD4/ADC10/AD2/P0.2
62 LIN_TX/CAN2_TX/T4CLKO_2/PWM8_3/TxD4/ADC11/AD3/P0.3
63 T3/ADC12/AD4/P0.4
64 LIN_RX_2/CAN2_RX_2/RxD4_2/P5.2

Note:
1.Except for P3.0 and P3.1, all other I/O ports are in high-impedance input state after power-on. User must set the I/O port mode firstly when using I/O.
2.All I/O ports can be set to quasi-bidirectional port mode, strong push-pull output mode, open-drain output mode or high-impedance input mode. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.
3. When P5.4 is enabled as the reset pin, the reset level is low.
4. ADC's external reference power supply pin ADC_VRef+ must not be floating. It must be connected to an external reference power supply or directly connected to Vcc.
5. If USB download is not required, P3.0/P3.1/P3.2 cannot be at low level at the same time when the chip is reset.


STC32G12K128
LQFP48
1 LIN_TX_2/CAN_TX_2/TxD4_2/P5.3
2 T3CLKO/ADC13/AD5/P0.5
3 PWMETI2_2/T4/ADC14/AD6/P0.6
4 T4CLKO/AD7/P0.7
5 RxD2/PWM1P/ADC0/P1.0
6 TxD2/PWM1N/ADC1/P1.1
7 LIN_TX_3/CAN_TX_3/TxD2_2/P4.7
8 SDA/S2MISO/S2MISO/MISO/PWM3P/ADC4/P1.4
9 SCL/S2SCLK/S1SCLK/SCLK/PWM3N/ADC5/P1.5
10 XTALO/MCLKO_2/RxD_3/PWM4P/ADC6/P1.6
11 XTALI/PWM5_2/TxD_3/PWM4N/ADC7/P1.7
12 T2CLKO/S2MOSI/S1MOSI/MOSI/PWM2N/ADC3/P1.3
13 UCap
14 P5.4/RST/MCLKO/SS/SS_3/S1SS/S1SS_3/S2SS/S2SS_3/PWM2P/PWM6_2/ADC2/T2
15 Vcc/AVcc
16 ADC_VRef+
17 Gnd/AGnd
18 P4.0/MOSI_3/S1MOSI_3/S2MOSI_3
19 P3.0/RxD/D-/INT4
20 P3.1/TxD/D+
21 P3.2/INT0/SCLK_4/SCL_4/PWMETI/PWMETI2
22 P3.3/INT1/MISO_4/SDA_4/PWM4N_4/PWM7_2
23 P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO
24 P5.0/RxD3_2/CAN_RX_2
25 P5.1/TxD3_2/CAN_TX_2
26 P3.5/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2
27 P3.6/INT2/RxD_2/CMP-
28 P3.7/INT3/TxD_2/CMP+
29 P4.1/MISO_3/S1MISO_3/S2MISO_3/CMPO_2/PWMETI_2
30 P4.2/WR/CAN_RX_3
31 P4.3/RxD_4/SCLK_3/S1SCLK_3/S2SCLK_3
32 P4.4/RD/TxD_4
33 P2.0/A8/PWM1P_2/PWM5
34 P2.1/A9/PWM1N_2/PWM6
35 P2.2/A10/PWM2P_2/PWM7/SS_2/S1SS_2/S2SS_2
36 P2.3/A11/PWM2N_2/PWM8/MOSI_2/S1MOSI_2/S2MOSI_2
37 S2MISO_2/S1MISO_2/MISO_2/SDA_2/PWM3P_2/A12/P2.4
38 S2SCLK_2/S1SCLK_2/SCLK_2/SCL_2/PWM3N_2/A13/P2.5
39 PWM4P_2/A14/P2.6
40 PWM4N_2/A15/P2.7
41 CAN_TX_3/ALE/P4.5
42 LIN_RX_3/CAN2_RX_3/RxD2_2/P4.6
43 CAN_RX/T3_2/PWM5_3/RxD3/ADC8/AD0/P0.0
44 CAN_TX/T3CLKO_2/PWM6_3/TxD3/ADC9/AD1/P0.1
45 LIN_RX/CAN2_RX/T4_2/PWM7_3/RxD4/ADC10/AD2/P0.2
46 LIN_TX/CAN2_TXT4CLKO_2/PWM8_3/TxD4/ADC11/AD3/P0.3
47 T3/ADC12/AD4/P0.4
48 LIN_RX_2/CAN2_RX_2/RxD4_2/P5.2

Note:

1.Except for P3.0 and P3.1, all other I/O ports are in high-impedance input state after power-on. User must set the I/O port mode firstly when using I/O.

2.All I/O ports can be set to quasi-bidirectional port mode, strong push-pull output mode, open-drain output mode or high-impedance input mode. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.

3. When P5.4 is enabled as the reset pin, the reset level is low.

4. ADC's external reference power supply pin ADC_VRef+ must not be floating. It must be connected to an external reference power supply or directly connected to Vcc.

5. If USB download is not required, P3.0/P3.1/P3.2 cannot be at low level at the same time when the chip is reset.

2.9.3 Pin descriptions

Pin number		name	type	description
LQFP64	LQFP48			
1	2	P5.3	I/O	Standard IO port
		TxD4_2	O	Transmit pin of UART 4
		CAN2_TX_2	O	Transmit pin of CAN2
		LIN_TX_2	O	Transmit pin of LIN
2	2	P0.5	I/O	Standard IO port
		AD5	I	Address/data bus
		ADC13	I	ADC analog input 13
		T3CLKO	O	Clock out of timer 3
3	3	P0.6	I/O	Standard IO port
		AD6	I	Address/data bus
		ADC14	I	ADC analog input 14
		T4	I	Timer4 external input
		PWMFLT2_2	I	Enhance PWM external anomaly detection pin 2
4	4	P0.7	I/O	Standard IO port
		AD7	I	Address/data bus
		T4CLKO	O	Clock out of timer 4
5		P6.0	I/O	Standard IO port
		PWM1P_3	I/O	Capture of external signal/ Positive of PWM1 pulse output
6		P6.1	I/O	Standard IO port
		PWM1N_3	I/O	Capture of external signal/ Negative of PWM1 pulse output
7		P6.2	I/O	Standard IO port
		PWM2P_3	I/O	Capture of external signal/ Positive of PWM2 pulse output
8		P6.3	I/O	Standard IO port
		PWM2N_3	I/O	Capture of external signal/ Negative of PWM2 pulse output
9	5	P1.0	I/O	Standard IO port
		ADC0	I	ADC analog input 0
		PWM1P	I/O	Capture of external signal/ Positive of PWMA pulse output
		RxD2	I	Input of USART2
10	6	P1.1	I/O	Standard IO port
		ADC1	I	ADC analog input 1
		PWM1N	I/O	Capture of external signal/Negative of PWMA pulse output
		TxD2	I	Transmit pin of USART 2
11	7	P4.7	I/O	Standard IO port
		TxD2_2	I	Transmit pin of USART 2
		CAN2_TX_3	O	Transmit pin of CAN2
		LIN_TX_3	O	Transmit pin of LIN
12	8	P1.4	I/O	Standard IO port
		ADC4	I	ADC analog input 4
		PWM3P	I/O	Capture of external signal/Positive of PWM3 pulse output
		MISO	I/O	Master Iutput/Slave Onput of SPI
		S1MISO	I/O	Master Iutput/Slave Onput of USART1-SPI
		S2MISO	I/O	Master Iutput/Slave Onput of USART2-SPI
		SDA	I/O	Serial data line of I2C
13	9	P1.5	I/O	Standard IO port
		ADC5	I	ADC analog input 5
		PWM3N	I/O	Capture of external signal/Negative of PWM3 pulse output
		SCLK	I/O	Serial Clock of SPI
		S1SCLK	I/O	Serial Clock of USART1-SPI
		S2SCLK	I/O	Serial Clock of USART2-SPI
		SCL	I/O	Serial Clock line of I2C

Pin number		name	type	description
LQFP64	LQFP48			
14	10	P1.6	I/O	Standard IO port
		ADC6	I	ADC analog input 6
		RxD_3	I	Input of USART 1
		PWM4P	I/O	Capture of external signal/Positive of PWM4 pulse output
		MCLKO_2	O	Main clock output
		XTALO	O	Connect to external oscillator
15	11	P1.7	I/O	Standard IO port
		ADC7	I	ADC analog input 7
		TxD_3	O	Transmit pin of USART 1
		PWM4N	I/O	Capture of external signal/Negative of PWM4 pulse output
		PWM5_2	I/O	Capture of external signal/Pulse output of PWM5
		XTALI	I	Connect to external oscillator
16	12	P1.3	I/O	Standard IO port
		ADC3	I	ADC analog input 3
		MOSI	I/O	Master Output/Slave Input of SPI
		S1MOSI	I/O	Master Output/Slave Input of USART1-SPI
		S2MOSI	I/O	Master Output/Slave Input of USART2-SPI
		PWM2N	I/O	Capture of external signal/Negative of PWM2 pulse output
		T2CLKO	O	Clock out of timer 2
17	13	UCAP	I	USB core power regulator pin
18	14	P5.4	I/O	Standard IO port
		NRST	I	Reset pin (low level reset)
		MCLKO	O	Main clock output
		SS_3	I	Slave selection of SPI (it is output with regard to master)
		SS	I	Slave selection of SPI (it is output with regard to master)
		S1SS_3	I	Slave selection of USART1-SPI (it is output with regard to master)
		S1SS	I	Slave selection of USART1-SPI (it is output with regard to master)
		S2SS_3	I	Slave selection of USART2-SPI (it is output with regard to master)
		S2SS	I	Slave selection of USART2-SPI (it is output with regard to master)
		PWM2P	I/O	Capture of external signal/Positive of PWM2 pulse output
		PWM6_2	I/O	Capture of external signal/Pulse output of PWM6
		T2	I	Timer2 external input
		ADC2	I	ADC analog input 2
19	15	Vcc	Vcc	Power Supply
		AVcc	Vcc	ADC Power Supply
20	16	VRef+	I	Reference voltage pin of ADC
21	17	Gnd	Gnd	Ground
		AGnd	Gnd	ADC Ground
		VRef-	I	Reference voltage ground pin of ADC
22	18	P4.0	I/O	Standard IO port
		MOSI_3	I/O	Master Output/Slave Input of SPI
		S1MOSI_3	I/O	Master Output/Slave Input of USART1-SPI
		S2MOSI_3	I/O	Master Output/Slave Input of USART2-SPI
23		P6.4	I/O	Standard IO port
		PWM3P_3	I/O	Capture of external signal/Positive of PWM3 pulse output
		S1SS_4	I	Slave selection of USART1-SPI (it is output with regard to master)
24		P6.5	I/O	Standard IO port
		PWM3N_3	I/O	Capture of external signal/Negative of PWM3 pulse output
		S1MOSI_4	I/O	Master Output/Slave Input of USART1-SPI
25		P6.6	I/O	Standard IO port
		PWM4P_3	I/O	Capture of external signal/Positive of PWM4 pulse output
		S1MISO_4	I/O	Master Output/Slave Input of USART1-SPI

Pin number		name	type	description
LQFP64	LQFP48			

26		P6.7	I/O	Standard IO port
		PWM4N_3	I/O	Capture of external signal/Negative of PWM4 pulse output
		S1SCLK_4	I/O	Serial Clock of USART1-SPI
27	19	P3.0	I/O	Standard IO port
		D-	I/O	USB data line
		RxD	I	Input of USART1
		INT4	I	External interrupt4
28	20	P3.1	I/O	Standard IO port
		D+	I/O	USB data line
		TxD	O	Transmit pin of USART1
29	21	P3.2	I/O	Standard IO port
		INT0	I	External interrupt0
		SCLK_4	I/O	Serial Clock of SPI
		SCL_4	I/O	Serial Clock line of I2C
		PWMETI	I	PWM external trigger input pin
		PWMET2	I	PWM external trigger input pin 2
30	22	P3.3	I/O	Standard IO port
		INT1	I	External interrupt1
		MISO_4	I/O	Master Iutput/Slave Onput of SPI
		SDA_4	I/O	Serial data line of I2C
		PWM4N_4	I/O	Capture of external signal/Negative of PWM4 pulse output
		PWM7_2	I/O	Capture of external signal/Pulse output of PWM7
31	23	P3.4	I/O	Standard IO port
		T0	I	Timer0 external input
		T1CLKO	O	Clock out of timer 1
		MOSI_4	I/O	Master Output/Slave Input of SPI
		PWM4P_4	I/O	Capture of external signal/Positive of PWM4 pulse output
		PWM8_2	I/O	Capture of external signal/Pulse output of PWM8
		CMPO	O	Output of comparator
32	24	P5.0	I/O	Standard IO port
		RxD3_2	I	Input of UART3
		CMP+_2	I	Positive input of comparator
		CAN_RX_2	I	Receive pin of CAN
33	25	P5.1	I/O	Standard IO port
		TxD3_2	O	Transmit pin of UART 3
		CMP+_3	I	Positive input of comparator
		CAN_TX_2	I	Transmit pin of CAN
34	26	P3.5	I/O	Standard IO port
		T1	I	Timer1 external input
		T0CLKO	O	Clock out of timer 0
		SS_4	I	Slave selection of SPI (it is output with regard to master)
		PWMFLT	I	Enhance PWMA external anomaly detection pin
35	27	P3.6	I/O	Standard IO port
		INT2	I	External interrupt2
		RxD_2	I	Input of USART1
		CMP-	I	Negative input of comparator
36	28	P3.7	I/O	Standard IO port
		INT3	I	External interrupt3
		TxD_2	O	Transmit pin of USART 1
		CMP+	I	Positive input of comparator
37		P7.0	I/O	Standard IO port
		CAN_RX_4	I	Receive pin of CAN

Pin number		name	type	description
LQFP64	LQFP48			
38		P7.1	I/O	Standard IO port
		CAN_TX_4	O	Transmit pin of CAN
40		P7.3	I/O	Standard IO port

		CAN2_TX_4	O	Transmit pin of CAN2
		LIN_TX_4	O	Transmit pin of LIN
		PWMETI_3	I	PWM external trigger input pin
41	29	P4.1	I/O	Standard IO port
		MISO_3	I/O	Master Iutput/Slave Onput of SPI
		S1MISO_3	I/O	Master Iutput/Slave Onput of USART1-SPI
		S2MISO_3	I/O	Master Iutput/Slave Onput of USART2-SPI
		CMPO_2	O	Output of comparator
		PWMETI_3	I	PWM external trigger input pin
42	30	P4.2	I/O	Standard IO port
		WR	O	WRITE signal of external bus
		CAN_RX_3	I	Receive pin of CAN
43	31	P4.3	I/O	Standard IO port
		RxD_4	I	Input of USART1
		SCLK_3	I/O	Serial Clock of SPI
		S1SCLK_3	I/O	Serial Clock of USART1-SPI
		S2SCLK_3	I/O	Serial Clock of USART2-SPI
44	32	P4.4	I/O	Standard IO port
		RD	O	READ signal of external bus
		TxD_4	O	Transmit pin of USART 1
45	33	P2.0	I/O	Standard IO port
		A8	I	Address bus
		PWM1P_2	I/O	Capture of external signal/Positive of PWMA pulse output
		PWM5	I/O	Capture of external signal/Pulse output of PWM5
46	34	P2.1	I/O	Standard IO port
		A9	I	Address bus
		PWM1N_2	I/O	Capture of external signal/Negative of PWMA pulse output
		PWM6	I/O	Capture of external signal/Pulse output of PWM6
47	35	P2.2	I/O	Standard IO port
		A10	I	Address bus
		SS_2	I	Slave selection of SPI (it is output with regard to master)
		S1SS_2	I	Slave selection of USART1-SPI (it is output with regard to master)
		S2SS_2	I	Slave selection of USART2-SPI (it is output with regard to master)
		PWM2P_2	I/O	Capture of external signal/Positive of PWMB pulse output
		PWM7	I/O	Capture of external signal/Pulse output of PWM7
48	36	P2.3	I/O	Standard IO port
		A11	I	Address bus
		MOSI_2	I/O	Master Output/Slave Input of SPI
		S1MOSI_2	I/O	Master Output/Slave Input of USART1-SPI
		S2MOSI_2	I/O	Master Output/Slave Input of USART2-SPI
		PWM2N_2	I/O	Capture of external signal/Negative of PWMB pulse output
		PWM8	I/O	Capture of external signal/Pulse output of PWM8

Pin number		name	type	description
LQFP64	LQFP48			
49	37	P2.4	I/O	Standard IO port
		A12	I	Address bus
		MISO_2	I/O	Master Iutput/Slave Onput of SPI
		S1MISO_2	I/O	Master Iutput/Slave Onput of USART1-SPI
		S2MISO_2	I/O	Master Iutput/Slave Onput of USART2-SPI
		SDA_2	I/O	Serial data line of I2C
		PWM3P_2	I/O	Capture of external signal/Positive of PWM3 pulse output
50	38	P2.5	I/O	Standard IO port

		A13	I	Address bus
		SCLK_2	I/O	Serial Clock of SPI
		S1SCLK_2	I/O	Serial Clock of USART1-SPI
		S2SCLK_2	I/O	Serial Clock of USART2-SPI
		SCL_2	I/O	Serial Clock line of I2C
		PWM3N_2	I/O	Capture of external signal/Negative of PWM3 pulse output
51	39	P2.6	I/O	Standard IO port
		A14	I	Address bus
		PWM4P_2	I/O	Capture of external signal/Positive of PWM4 pulse output
52	40	P2.7	I/O	Standard IO port
		A15	I	Address bus
		PWM4N_2	I/O	Capture of external signal/Negative of PWM4 pulse output
53		P7.4	I/O	Standard IO port
		PWM5_4	I/O	Capture of external signal/Pulse output of PWM5
		S2SS_4	I	Slave selection of USART2-SPI (it is output with regard to master)
54		P7.5	I/O	Standard IO port
		PWM6_4	I/O	Capture of external signal/Pulse output of PWM6
		S2MOSI_4	I/O	Master Iutput/Slave Onput of USART2-SPI
55		P7.6	I/O	Standard IO port
		PWM7_4	I/O	Capture of external signal/Pulse output of PWM7
		S2MISO_4	I/O	Master Iutput/Slave Onput of USART2-SPI
56		P7.7	I/O	Standard IO port
		PWM8_4	I/O	Capture of external signal/Pulse output of PWM8
		S2SCLK_4	I/O	Serial Clock of USART2-SPI
57	41	P4.5	I/O	Standard IO port
		ALE	O	Address Latch Enable signal
		CAN_TX_3	O	Transmit pin of CAN
58	42	P4.6	I/O	Standard IO port
		RxD2_2	I	Input of USART2
		CAN2_RX_3	I	Receive pin of CAN2
		LIN_RX_3	I	Receive pin of LIN
59	43	P0.0	I/O	Standard IO port
		AD0	I	Address/data bus
		ADC8	I	ADC analog input 8
		RxD3	I	Input of UART3
		PWM5_3	I/O	Capture of external signal/Pulse output of PWM5
		CAN_RX	I	Receive pin of CAN

Pin number		name	type	description
LQFP64	LQFP48			
60	44	P0.1	I/O	Standard IO port
		AD1	I	Address/data bus
		ADC9	I	ADC analog input 9
		TxD3	O	Transmit pin of UART 3
		PWM6_3	I/O	Capture of external signal/Pulse output of PWM6
		CAN_TX	O	Transmit pin of CAN
61	45	P0.2	I/O	Standard IO port
		AD2	I	Address/data bus
		ADC10	I	ADC analog input 10
		RxD4	I	Input of UART4
		PWM7_3	I/O	Capture of external signal/Pulse output of PWM7

		CAN2_RX	I	Receive pin of CAN2
		LIN_RX	I	Receive pin of LIN
62	46	P0.3	I/O	Standard IO port
		AD3	I	Address/data bus
		ADC11	I	ADC analog input 11
		TxD4	O	Transmit pin of UART 4
		PWM8_3	I/O	Capture o f external signal/Pulse output of PWM8
		CAN2_TX	O	Transmit pin of CAN2
		LIN_TX	O	Transmit pin of LIN
63	47	P0.4	I/O	Standard IO port
		AD4	I	Address/data bus
		ADC12	I	ADC analog input 12
		T3	I	Timer3 external input
64	48	P5.2	I/O	Standard IO port
		RxD4_2	I	Input of UART4
		CAN2_RX_2	I	Receive pin of CAN2
		LIN_RX_2	I	Receive pin of LIN

2.10 STC32G6K64-LQFP48/LQFP32/PDIP40 (New product notice)

2.10.1 Features and Price

- **Selection and price (No external crystal and external reset required with 15 channels 12-bit ADC)**

MCU	Operating voltage（V）	Flash Code Memory (100 thousand times) (Byte)	edata Internal extended DATA RAM which can be used as stack(Byte)	xdata Internal extended SRAM (Byte)	Enhanced Dual DPTR increasing or decreasing	EEPROM 100 thousand times) (Byte)	Maximum I/O Lines	Traditional I/O interrupt(INT0/INT1/INT2/INT3/INT4) (can wake-up CPU)	All I/O ports support interrupts and can wake up MCU	DMA 8080/6800 interface/ LCM driver(8-bit and 16-bit)	RTC	DMA UARTs which can wake-up CPU	DMA USARTs which can wake-up CPU	CAN	LIN	DMA SPI which can wake-up CPU	I^{2}C which can wake-up CPU	MDU32 (Hardware 32-bit Multiplier and Divider)	Timers/Counters (T0-T4 pin Ccn wake-up CPU)	16-bit advanced PWM timer with Complementary symmetrical dead-time	Power-down Wake-up timer	DMA 15 channels high speed ADC (8 PWMs can be used as 8 DACs)	Comparator (May be used as ADC to detect external power-down)	Internal LVD interrupt (can wake-up CPU)	Watch-dog Timer	Internal high reliable reset circuit with 4 levels optional reset threshold voltage	Internal high presision Clock (adjustbal under 33MHz)	Clock output and Reset	Program encrypted transmission (Anti-blocking)	Password can be set for next update	Support RS485 download	Support software USB download directly	Online debug itself	Price & Package LQFP48 <9mm*9mm>	QFN48 <6mm*6mm>	LQFP32<9mm*9mm>	PDIP40	products supply information
STC32G6K48	1.9-5.5	48K	4K	2K	2	48K	45	Y	Y	Y	Y	2	2	2	Y	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	¥4	√	√	√	availa
STC32G6K64	1.9-5.5	64K	4K	2K	2	IAP	45	Y	Y	Y	Y	2	2	2	Y	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	¥4	√	√	√	

- **Core**
  - ✓ Ultra-high speed 32-bit 8051 Core with single clock per machine cycle, which is called 1T and the speed is about 70 times faster than traditional 8051
  - ✓ 48 interrupt sources and 4 interrupt priority levels
  - ✓ Online debugging is supported

- **Operating voltage**
  - ✓ 1.9V～5.5V
  - ✓ Built-in LDO

- **Operating temperature**
  - ✓ -40℃~85℃

- **Flash memory**
  - ✓ Up to 64Kbytes of Flash memory to be used for storing user code
  - ✓ Configurable EEPROM size, 512bytes single page for being erased, which can be repeatedly erased more than 100 thousand times.
  - ✓ In-System-Programming, ISP in short, can be used to update the application code. No special programmer is needed.
  - ✓ Online debugging with single chip is supported, and no special emulator is needed. The number of breakpoints is unlimited theoratically.
  - ✓ Support hardware emulation of SWD interface (requires STC-USB Link1 tool)

- **SRAM**
  - ✓ 4K bytes internal SRAM (EDATA)
  - ✓ 2K bytes internal extended RAM (internal XDATA

    Notes on using xdata:

    When defining variables, single-byte variables can be defined in xdata, and multi-byte (2-byte, 4-byte) variables need to be defined in edata.

- **Clock**

- ✓ Internal high precise RC clock IRC(IRC for short), adjustable while ISP and can be divided to lower frequency by user software.
  - ✓ Error:±0.3% (at the temperature 25℃)
  - ✓ -1.35%～+1.30% temperature drift (at the temperature range of -40℃ to +85℃)
  - ✓ -0.76%～+0.98% temperature drift (at the temperature range of -20℃ to 65℃)
- ✓ Internal 32KHz low speed IRC with large error
- ✓ External crystal (4MHz～33MHz) and external clock
- ✓ Internal PLL output clock

  Users can freely choose the above 4 clock sources

- ➢ **Reset**
  - ✓ Hardware reset
    - ✓ Power-on reset. Measured voltage is 1.7V~1.9V. (Effective when the chip does not enable the low voltage reset function)
    - ✓ Reset by reset pin. The default function of P5.4 is the I/O port. The P5.4 pin can be set as the reset pin while ISP download. (Note: When the P5.4 pin is set as the reset pin, the reset level is low.)
    - ✓ Watch dog timer reset
    - ✓ Low voltage detection reset. 4 low voltage detection levels are provided, 2.0V, 2.3V, 2.7V, 3.0V.
  - ✓ Software reset
    - ✓ Writing the reset trigger register using software

- ➢ **Interrupts**
  - ✓ 48 interrupt sources: INT0, INT1, INT2, INT3, INT4, timer 0, timer 1, timer 2, timer 3, timer 4, USART1, USART2, UART 3, UART 4, ADC, LVD, SPI, I^{2}C, comparator, PWMA, PWMB, CAN, CAN2, LIN, LCMIF color screen interface, RTC, all I/O interrupts (8 groups), DMA receive and transmit interrupts of USART 1, DMA receive and transmit interrupts of USART 2, DMA receive and transmit interrupts of UART 3, DMA receive and transmit interrupts of UART 4, DMA interrupt of I2C, DMA interrupt of SPI, DMA interrupt of ADC, DMA interrupt of LCD driver and DMA interrupt of memory-to-memory.
  - ✓ 4 interrupt priority levels

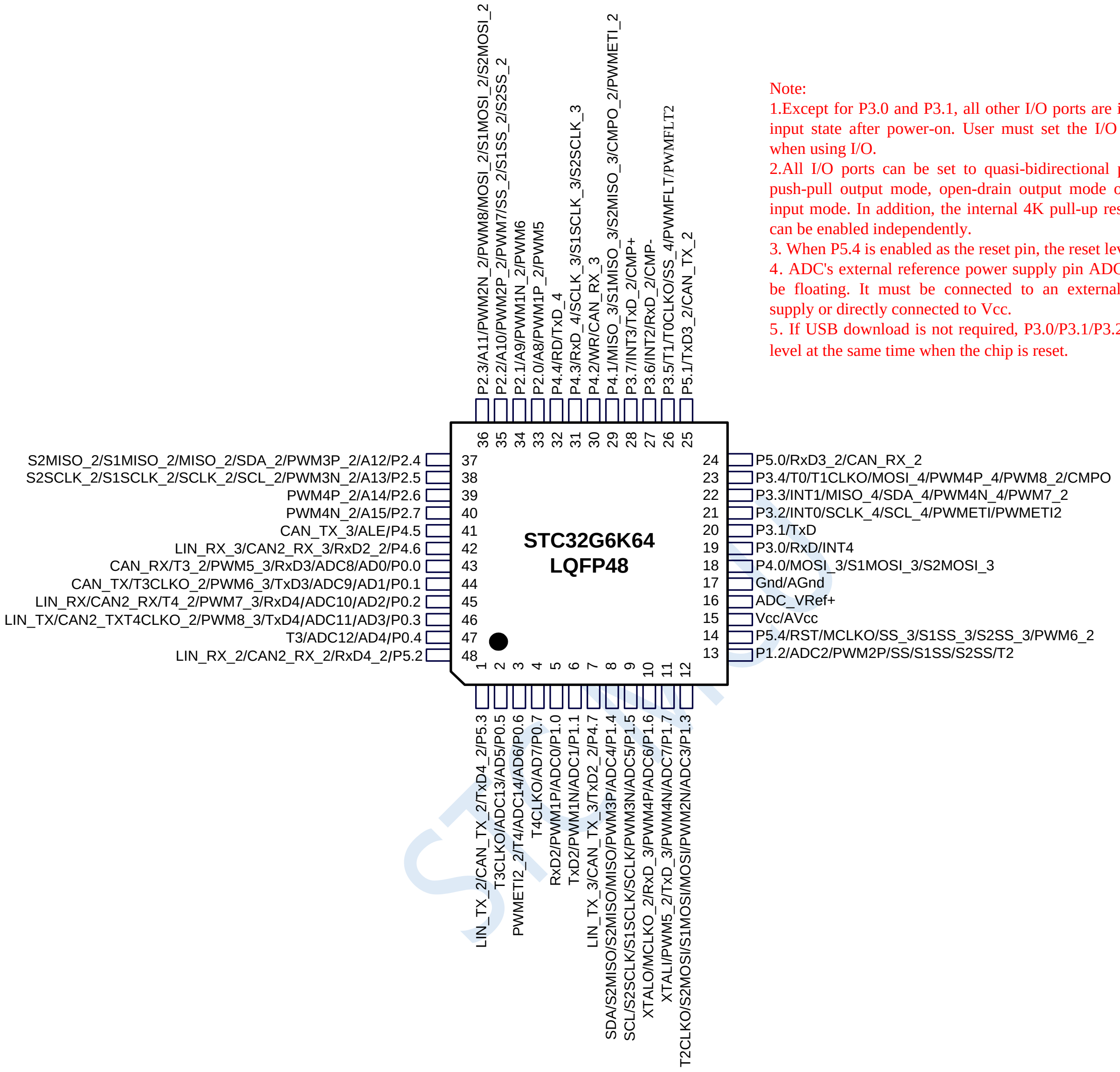
- ➢ **Digital peripherals**
  - ✓ 5 16-bit timers: timer0, timer1, timer2, timer3, timer4, where the mode 3 of timer 0 has the Non-Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
  - ✓ 2 high speed USARTs: USART1, USART2, whose maximum baudrate clock may be FOSC/4. The following modes are supported, synchronous serial port mode, asynchronous serial port mode, SPI mode, LIN mode, infrared mode (IrDA), smart card mode (ISO7816)
  - ✓ 2 high speed UARTs: UART3, UART4, whose maximum baudrate clock may be FOSC/4
  - ✓ 2 groups of enhanced PWM, which can realize 8 channels(4 groups complementary symmetry) control signals with dead time, and support external fault detection function.
  - ✓ SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
  - ✓ I^{2}C: Master mode or slave mode are supported.
  - ✓ ICE: Hardware support emulation.
  - ✓ RTC: Support year, month, day, hour, minute, second, sub-second (1/128 second). And supports clock interrupt and a set of alarm clocks.
  - ✓ CAN: Two independent CAN 2.0 control units.
  - ✓ LIN: An independent LIN control unit (supports versions 1.3 and 2.1), USART1 and USART2 can support two sets of LIN.
  - ✓ MDU32: Hardware 32-bit Multiplier and Divider which supports 32-bit divided by 32-bit, 32-bit multiplied by 32-bit operations.
  - ✓ I/O port interrupt: All I/Os support interrupts, each group of I/O interrupts has an independent interrupt entry address, all I/O interrupts can support 4 types interrupt mode: high level interrupt, low level interrupt, rising edge interrupt, falling edge interrupt. Provides 4 levels of interrupt priority and supports power-down wake-up function.
  - ✓ LCD dirver: support 8080 and 6800 interface, and support 8-bit and 16-bit data width.
  - ✓ DMA: Support SPI shift to receive data to memory, SPI shift to send data from memory, I2C send data from memory, I2C receive data to memory, USART 1/2 and UART 3/4 receive data to memory, USART 1/2 and UART 3/4 send data from memory, ADC automatically sample data to memory (calculate average value at the same time), LCD driver send data from memory, and copy data from memory to memory
  - ✓ Hardware digital ID: support 32 bytes

- ➢ **Analog peripherals**
  - ✓ Ultra high speed ADC which supports 12-bit precision 15 channels (channel 0 to channel 14) analog-to-digital conversion. ADC channel 15 is used to test the internal reference voltage. (The default internal reference voltage is 1.19V when the chip is shipped, the error is ±1%)
  - ✓ Comparator. A set of comparator

- ➢ **GPIO**
  - ✓ Up to 45 GPIOs: P0.0~P0.7, P1.0~P1.7, P2.0~P2.7, P3.0~P3.7, P4.0~P4.7, P5.0~P5.4
  - ✓ 4 modes for all GPIOs: quasi_bidirectional mode, push-pull outputmode, open drain mode, high-impedance input mode
  - ✓ Except for P3.0 and P3.1, all other I/O ports are in a high-impedance state after power-on. User must set the I/O ports mode before using them. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.

- ➢ **Package**
  - ✓ LQFP48, LQFP32, PDIP40

2.10.2 Pinouts


P2.3/A11/PWM2N_2/PWM8/MOSI_2/S1MOSI_2/S2MOSI_2
P2.2/A10/PWM2P_2/PWM7/SS_2/S1SS_2/S2SS_2
P2.1/A9/PWM1N_2/PWM6
P2.0/A8/PWM1P_2/PWM5
P4.4/RD/TxD_4
P4.3/RxD_4/SCLK_3/S1SCLK_3/S2SCLK_3
P4.2/WR/CAN_RX_3
P4.1/MISO_3/S1MISO_3/S2MISO_3/CMPO_2/PWMETI_2
P3.7/INT3/TxD_2/CMP+
P3.6/INT2/RxD_2/CMP-
P3.5/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2
P5.1/TxD3_2/CAN_TX_2
36
35
34
33
32
31
30
29
28
27
26
25
S2MISO_2/S1MISO_2/MISO_2/SDA_2/PWM3P_2/A12/P2.4 37
S2SCLK_2/S1SCLK_2/SCLK_2/SCL_2/PWM3N_2/A13/P2.5 38
PWM4P_2/A14/P2.6 39
PWM4N_2/A15/P2.7 40
CAN_TX_3/ALE/P4.5 41
LIN_RX_3/CAN2_RX_3/RxD2_2/P4.6 42
CAN_RX/T3_2/PWM5_3/RxD3/ADC8/AD0/P0.0 43
CAN_TX/T3CLKO_2/PWM6_3/TxD3/ADC9/AD1/P0.1 44
LIN_RX/CAN2_RX/T4_2/PWM7_3/RxD4/ADC10/AD2/P0.2 45
LIN_TX/CAN2_TXT4CLKO_2/PWM8_3/TxD4/ADC11/AD3/P0.3 46
T3/ADC12/AD4/P0.4 47
LIN_RX_2/CAN2_RX_2/RxD4_2/P5.2 48
STC32G6K64
LQFP48
24 P5.0/RxD3_2/CAN_RX_2
23 P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO
22 P3.3/INT1/MISO_4/SDA_4/PWM4N_4/PWM7_2
21 P3.2/INT0/SCLK_4/SCL_4/PWMETI/PWMETI2
20 P3.1/TxD
19 P3.0/RxD/INT4
18 P4.0/MOSI_3/S1MOSI_3/S2MOSI_3
17 Gnd/AGnd
16 ADC_VRef+
15 Vcc/AVcc
14 P5.4/RST/MCLKO/SS_3/S1SS_3/S2SS_3/PWM6_2
13 P1.2/ADC2/PWM2P/SS/S1SS/S2SS/T2
1
2
3
4
5
6
7
8
9
10
11
12
LIN_TX_2/CAN_TX_2/TxD4_2/P5.3
T3CLKO/ADC13/AD5/P0.5
PWMETI2_2/T4/ADC14/AD6/P0.6
T4CLKO/AD7/P0.7
RxD2/PWM1P/ADC0/P1.0
TxD2/PWM1N/ADC1/P1.1
LIN_TX_3/CAN_TX_3/TxD2_2/P4.7
SDA/S2MISO/S2MISO/MISO/PWM3P/ADC4/P1.4
SCL/S2SCLK/S1SCLK/SCLK/PWM3N/ADC5/P1.5
XTALO/MCLKO_2/RxD_3/PWM4P/ADC6/P1.6
XTALI/PWM5_2/TxD_3/PWM4N/ADC7/P1.7
T2CLKO/S2MOSI/S1MOSI/MOSI/PWM2N/ADC3/P1.3

Note:
1.Except for P3.0 and P3.1, all other I/O ports are in high-impedance input state after power-on. User must set the I/O port mode firstly when using I/O.
2.All I/O ports can be set to quasi-bidirectional port mode, strong push-pull output mode, open-drain output mode or high-impedance input mode. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.
3. When P5.4 is enabled as the reset pin, the reset level is low.
4. ADC's external reference power supply pin ADC_VRef+ must not be floating. It must be connected to an external reference power supply or directly connected to Vcc.
5. If USB download is not required, P3.0/P3.1/P3.2 cannot be at low level at the same time when the chip is reset.

2.10.3 Pin descriptions

Pin number		name	type	description
LQFP48				
1		P5.3	I/O	Standard IO port
		TxD4_2	O	Transmit pin of UART 4
		CAN2_TX_2	O	Transmit pin of CAN2
		LIN_TX_2	O	Transmit pin of LIN
2		P0.5	I/O	Standard IO port
		AD5	I	Address/data bus
		ADC13	I	ADC analog input 13
		T3CLKO	O	Clock out of timer 3
3		P0.6	I/O	Standard IO port
		AD6	I	Address/data bus
		ADC14	I	ADC analog input 14
		T4	I	Timer4 external input
		PWMFLT2_2	I	Enhance PWM external anomaly detection pin 2
4		P0.7	I/O	Standard IO port
		AD7	I	Address/data bus
		T4CLKO	O	Clock out of timer 4
5		P1.0	I/O	Standard IO port
		ADC0	I	ADC analog input 0
		PWM1P	I/O	Capture of external signal/ Positive of PWMA pulse output
		RxD2	I	Input of USART2
6		P1.1	I/O	Standard IO port
		ADC1	I	ADC analog input 1
		PWM1N	I/O	Capture of external signal/Negative of PWMA pulse output
		TxD2	I	Transmit pin of USART 2
7		P4.7	I/O	Standard IO port
		TxD2_2	I	Transmit pin of USART 2
		CAN2_TX_3	O	Transmit pin of CAN2
		LIN_TX_3	O	Transmit pin of LIN
8		P1.4	I/O	Standard IO port
		ADC4	I	ADC analog input 4
		PWM3P	I/O	Capture of external signal/Positive of PWM3 pulse output
		MISO	I/O	Master Iutput/Slave Onput of SPI
		S1MISO	I/O	Master Iutput/Slave Onput of USART1-SPI
		S2MISO	I/O	Master Iutput/Slave Onput of USART2-SPI
		SDA	I/O	Serial data line of I2C
9		P1.5	I/O	Standard IO port
		ADC5	I	ADC analog input 5
		PWM3N	I/O	Capture of external signal/Negative of PWM3 pulse output
		SCLK	I/O	Serial Clock of SPI
		S1SCLK	I/O	Serial Clock of USART1-SPI
		S2SCLK	I/O	Serial Clock of USART2-SPI
		SCL	I/O	Serial Clock line of I2C
10		P1.6	I/O	Standard IO port
		ADC6	I	ADC analog input 6
		RxD_3	I	Input of USART 1
		PWM4P	I/O	Capture of external signal/Positive of PWM4 pulse output
		MCLKO_2	O	Main clock output
		XTALO	O	Connect to external oscillator

Pin number		name	type	description
LQFP48				
11		P1.7	I/O	Standard IO port
		ADC7	I	ADC analog input 7
		TxD_3	O	Transmit pin of USART 1
		PWM4N	I/O	Capture of external signal/Negative of PWM4 pulse output
		PWM5_2	I/O	Capture of external signal/Pulse output of PWM5
		XTALI	I	Connect to external oscillator
12		P1.3	I/O	Standard IO port
		ADC3	I	ADC analog input 3
		MOSI	I/O	Master Output/Slave Input of SPI
		S1MOSI	I/O	Master Output/Slave Input of USART1-SPI
		S2MOSI	I/O	Master Output/Slave Input of USART2-SPI
		PWM2N	I/O	Capture of external signal/Negative of PWM2 pulse output
		T2CLKO	O	Clock out of timer 2
13		P1.2	I/O	Standard IO port
		ADC2	I	ADC analog input 2
		PWM2P	I/O	Capture of external signal/Positive of PWM2 pulse output
		SS	I	Slave selection of SPI (it is output with regard to master)
		S1SS	I	Slave selection of USART1-SPI (it is output with regard to master)
		S2SS	I	Slave selection of USART2-SPI (it is output with regard to master)
		T2	I	Timer2 external input
14		P5.4	I/O	Standard IO port
		RST	I	Reset pin (low level reset)
		MCLKO	O	Main clock output
		SS_3	I	Slave selection of SPI (it is output with regard to master)
		S1SS_3	I	Slave selection of USART1-SPI (it is output with regard to master)
		S2SS_3	I	Slave selection of USART2-SPI (it is output with regard to master)
		PWM6_2	I/O	Capture of external signal/Pulse output of PWM6
15		Vcc	Vcc	Power Supply
		AVcc	Vcc	ADC Power Supply
16		VRef+	I	Reference voltage pin of ADC
17		Gnd	Gnd	Ground
		AGnd	Gnd	ADC Ground
		VRef-	I	Reference voltage ground pin of ADC
18		P4.0	I/O	Standard IO port
		MOSI_3	I/O	Master Output/Slave Input of SPI
		S1MOSI_3	I/O	Master Output/Slave Input of USART1-SPI
		S2MOSI_3	I/O	Master Output/Slave Input of USART2-SPI
19		P3.0	I/O	Standard IO port
		RxD	I	Input of USART1
		INT4	I	External interrupt4
20		P3.1	I/O	Standard IO port
		TxD	O	Transmit pin of USART1
21		P3.2	I/O	Standard IO port
		INT0	I	External interrupt0
		SCLK_4	I/O	Serial Clock of SPI
		SCL_4	I/O	Serial Clock line of I2C
		PWMETI	I	PWM external trigger input pin
		PWMET2	I	PWM external trigger input pin 2

Pin number		name	type	description
LQFP48				

22		P3.3	I/O	Standard IO port
		INT1	I	External interrupt1
		MISO_4	I/O	Master Iutput/Slave Onput of SPI
		SDA_4	I/O	Serial data line of I2C
		PWM4N_4	I/O	Capture of external signal/Negative of PWM4 pulse output
		PWM7_2	I/O	Capture of external signal/Pulse output of PWM7
23		P3.4	I/O	Standard IO port
		T0	I	Timer0 external input
		T1CLKO	O	Clock out of timer 1
		MOSI_4	I/O	Master Output/Slave Input of SPI
		PWM4P_4	I/O	Capture of external signal/Positive of PWM4 pulse output
		PWM8_2	I/O	Capture of external signal/Pulse output of PWM8
		CMPO	O	Output of comparator
24		P5.0	I/O	Standard IO port
		RxD3_2	I	Input of UART3
		CMP+_2	I	Positive input of comparator
		CAN_RX_2	I	Receive pin of CAN
25		P5.1	I/O	Standard IO port
		TxD3_2	O	Transmit pin of UART 3
		CMP+_3	I	Positive input of comparator
		CAN_TX_2	I	Transmit pin of CAN
26		P3.5	I/O	Standard IO port
		T1	I	Timer1 external input
		T0CLKO	O	Clock out of timer 0
		SS_4	I	Slave selection of SPI (it is output with regard to master)
		PWMFLT	I	Enhance PWMA external anomaly detection pin
27		P3.6	I/O	Standard IO port
		INT2	I	External interrupt2
		RxD_2	I	Input of USART1
		CMP-	I	Negative input of comparator
28		P3.7	I/O	Standard IO port
		INT3	I	External interrupt3
		TxD_2	O	Transmit pin of USART 1
		CMP+	I	Positive input of comparator
29		P4.1	I/O	Standard IO port
		MISO_3	I/O	Master Iutput/Slave Onput of SPI
		S1MISO_3	I/O	Master Iutput/Slave Onput of USART1-SPI
		S2MISO_3	I/O	Master Iutput/Slave Onput of USART2-SPI
		CMPO_2	O	Output of comparator
		PWMETI_3	I	PWM external trigger input pin
30		P4.2	I/O	Standard IO port
		WR	O	WRITE signal of external bus
		CAN_RX_3	I	Receive pin of CAN
31		P4.3	I/O	Standard IO port
		RxD_4	I	Input of USART1
		SCLK_3	I/O	Serial Clock of SPI
		S1SCLK_3	I/O	Serial Clock of USART1-SPI
		S2SCLK_3	I/O	Serial Clock of USART2-SPI
32		P4.4	I/O	Standard IO port
		RD	O	READ signal of external bus
		TxD_4	O	Transmit pin of USART 1

Pin number		name	type	description
LQFP48				
33		P2.0	I/O	Standard IO port
		A8	I	Address bus

		PWM1P_2	I/O	Capture of external signal/Positive of PWMA pulse output
		PWM5	I/O	Capture of external signal/Pulse output of PWM5
34		P2.1	I/O	Standard IO port
		A9	I	Address bus
		PWM1N_2	I/O	Capture of external signal/Negative of PWMA pulse output
		PWM6	I/O	Capture of external signal/Pulse output of PWM6
35		P2.2	I/O	Standard IO port
		A10	I	Address bus
		SS_2	I	Slave selection of SPI (it is output with regard to master)
		S1SS_2	I	Slave selection of USART1-SPI (it is output with regard to master)
		S2SS_2	I	Slave selection of USART2-SPI (it is output with regard to master)
		PWM2P_2	I/O	Capture of external signal/Positive of PWMB pulse output
		PWM7	I/O	Capture of external signal/Pulse output of PWM7
36		P2.3	I/O	Standard IO port
		A11	I	Address bus
		MOSI_2	I/O	Master Output/Slave Input of SPI
		S1MOSI_2	I/O	Master Output/Slave Input of USART1-SPI
		S2MOSI_2	I/O	Master Output/Slave Input of USART2-SPI
		PWM2N_2	I/O	Capture of external signal/Negative of PWMB pulse output
		PWM8	I/O	Capture of external signal/Pulse output of PWM8
37		P2.4	I/O	Standard IO port
		A12	I	Address bus
		MISO_2	I/O	Master Iutput/Slave Onput of SPI
		S1MISO_2	I/O	Master Iutput/Slave Onput of USART1-SPI
		S2MISO_2	I/O	Master Iutput/Slave Onput of USART2-SPI
		SDA_2	I/O	Serial data line of I2C
		PWM3P_2	I/O	Capture of external signal/Positive of PWM3 pulse output
38		P2.5	I/O	Standard IO port
		A13	I	Address bus
		SCLK_2	I/O	Serial Clock of SPI
		S1SCLK_2	I/O	Serial Clock of USART1-SPI
		S2SCLK_2	I/O	Serial Clock of USART2-SPI
		SCL_2	I/O	Serial Clock line of I2C
		PWM3N_2	I/O	Capture of external signal/Negative of PWM3 pulse output
39		P2.6	I/O	Standard IO port
		A14	I	Address bus
		PWM4P_2	I/O	Capture of external signal/Positive of PWM4 pulse output
40		P2.7	I/O	Standard IO port
		A15	I	Address bus
		PWM4N_2	I/O	Capture of external signal/Negative of PWM4 pulse output
41		P4.5	I/O	Standard IO port
		ALE	O	Address Latch Enable signal
		CAN_TX_3	O	Transmit pin of CAN
42		P4.6	I/O	Standard IO port
		RxD2_2	I	Input of USART2
		CAN2_RX_3	I	Receive pin of CAN2
		LIN_RX_3	I	Receive pin of LIN

Pin number		**name**	**type**	**description**
LQFP48				
43		P0.0	I/O	Standard IO port
		AD0	I	Address/data bus
		ADC8	I	ADC analog input 8
		RxD3	I	Input of UART3
		PWM5_3	I/O	Capture of external signal/Pulse output of PWM5
		CAN_RX	I	Receive pin of CAN

44		P0.1	I/O	Standard IO port
		AD1	I	Address/data bus
		ADC9	I	ADC analog input 9
		TxD3	O	Transmit pin of UART 3
		PWM6_3	I/O	Capture of external signal/Pulse output of PWM6
		CAN_TX	O	Transmit pin of CAN
45		P0.2	I/O	Standard IO port
		AD2	I	Address/data bus
		ADC10	I	ADC analog input 10
		RxD4	I	Input of UART4
		PWM7_3	I/O	Capture of external signal/Pulse output of PWM7
		CAN2_RX	I	Receive pin of CAN2
		LIN_RX	I	Receive pin of LIN
46		P0.3	I/O	Standard IO port
		AD3	I	Address/data bus
		ADC11	I	ADC analog input 11
		TxD4	O	Transmit pin of UART 4
		PWM8_3	I/O	Capture o f external signal/Pulse output of PWM8
		CAN2_TX	O	Transmit pin of CAN2
		LIN_TX	O	Transmit pin of LIN
47		P0.4	I/O	Standard IO port
		AD4	I	Address/data bus
		ADC12	I	ADC analog input 12
		T3	I	Timer3 external input
48		P5.2	I/O	Standard IO port
		RxD4_2	I	Input of UART4
		CAN2_RX_2	I	Receive pin of CAN2
		LIN_RX_2	I	Receive pin of LIN

2.11 STC32F12K60-LQFP48/LQFP32/PDIP40 (New product notice)

2.11.1 Features and Price

- **Selection and price (No external crystal and external reset required with 15 channels 12-bit ADC)**

MCU	Operating voltage（V）	Flash Code Memory (100 thousand times) (Byte)	edata Internal extended DATA RAM which can be used as stack(Byte)	xdata Internal extended SRAM (Byte)	Enhanced Dual DPTR increasing or decreasing	EEPROM 100 thousand times) (Byte)	Maximum I/O Lines	Traditional I/O interrupt(INT0/INT1/INT2/INT3/INT4) (can wake-up CPU)	All I/O ports support interrupts and can wake up MCU	DMA 8080/6800 interface/ LCM driver(8-bit and 16-bit)	RTC	DMA UARTs which can wake-up CPU	DMA USARTs which can wake-up CPU	CAN	LIN	DMA SPI which can wake-up CPU	I^{2}C which can wake-up CPU	MDU32 (Hardware 32-bit Multiplier and Divider)	FPMU(single-precision floating-point arithmetic)	Timers/Counters (T0-T4 pin Ccn wake-up CPU)	16-bit advanced PWM timer with Complementary symmetrical dead-time	Power-down Wake-up timer	DMA 15 channels high speed ADC (8 PWMs can be used as 8 DACs)	Comparator (May be used as ADC to detect external power-down)	Internal LVD interrupt (can wake-up CPU)	Watch-dog Timer	Internal high reliable reset circuit with 4 levels optional reset threshold voltage	Internal high precision Clock (adjustbal under 33MHz)	Clock output and Reset	Program encrypted transmission (Anti-blocking)	Password can be set for next update	Support RS485 download	Full speed USB, support hardware USB download and debug directly	Online debug itself	Price & Package				products supply information
																																			LQFP48 <9mm*9mm>	QFN48 <6mm*6mm>	LQFP32<9mm*9mm>	PDIP40	
STC32F12K48	1.9-5.5	48K	8K	4K	2	48K	44	Y	Y	Y	Y	2	2	2	Y	Y	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	¥5	√	√	√	availa
STC32F12K60	1.9-5.5	60K	8K	4K	2	IAP	44	Y	Y	Y	Y	2	2	2	Y	Y	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	¥5	√	√	√	

- **Core**
  - ✓ Ultra-high speed 32-bit 8051 Core with single clock per machine cycle, which is called 1T and the speed is about 70 times faster than traditional 8051
  - ✓ 50 interrupt sources and 4 interrupt priority levels
  - ✓ Online debugging is supported

- **Operating voltage**
  - ✓ 1.9V～5.5V
  - ✓ Built-in LDO

- **Operating temperature**
  - ✓ -40℃~85℃

- **Flash memory**
  - ✓ Up to 60Kbytes of Flash memory to be used for storing user code
  - ✓ Configurable EEPROM size, 512bytes single page for being erased, which can be repeatedly erased more than 100 thousand times.
  - ✓ In-System-Programming, ISP in short, can be used to update the application code. No special programmer is needed.
  - ✓ Online debugging with single chip is supported, and no special emulator is needed. The number of breakpoints is unlimited theoratically.
  - ✓ Support hardware emulation of SWD interface (requires STC-USB Link1 tool)

- **SRAM**
  - ✓ 8K bytes internal SRAM (EDATA)
  - ✓ 4K bytes internal extended RAM (internal XDATA

    Notes on using xdata:

    When defining variables, single-byte variables can be defined in xdata, and multi-byte (2-byte, 4-byte) variables need to be defined in edata.

- **Clock**
  - ✓ Internal high precise RC clock IRC(IRC for short), adjustable while ISP and can be divided to lower frequency by user

software.

  - ✓ Error:±0.3% (at the temperature 25℃)
  - ✓ -1.35%～+1.30% temperature drift (at the temperature range of -40℃ to +85℃)
  - ✓ -0.76%～+0.98% temperature drift (at the temperature range of -20℃ to 65℃)
- ✓ Internal 32KHz low speed IRC with large error
- ✓ External crystal (4MHz～33MHz) and external clock
- ✓ Internal PLL output clock

  Users can freely choose the above 4 clock sources

➢ **Reset**

- ✓ Hardware reset
  - ✓ Power-on reset. Measured voltage is 1.7V~1.9V. (Effective when the chip does not enable the low voltage reset function)
  - ✓ Reset by reset pin. The default function of P5.4 is the I/O port. The P5.4 pin can be set as the reset pin while ISP download. (Note: When the P5.4 pin is set as the reset pin, the reset level is low.)
  - ✓ Watch dog timer reset
  - ✓ Low voltage detection reset. 4 low voltage detection levels are provided, 2.0V, 2.3V, 2.7V, 3.0V.
- ✓ Software reset
  - ✓ Writing the reset trigger register using software

➢ **Interrupts**

- ✓ 50 interrupt sources: INT0, INT1, INT2, INT3, INT4, timer 0, timer 1, timer 2, timer 3, timer 4, USART1, USART2, UART 3, UART 4, ADC, LVD, SPI, I^{2}C, comparator, PWMA, PWMB, USB, CAN, CAN2, LIN, LCMIF color screen interface, RTC, all I/O interrupts (6 groups), I2S audio interface, DMA receive and transmit interrupts of I2S audio interface, DMA receive and transmit interrupts of USART 1, DMA receive and transmit interrupts of USART 2, DMA receive and transmit interrupts of UART 3, DMA receive and transmit interrupts of UART 4, DMA interrupt of I2C, DMA interrupt of SPI, DMA interrupt of ADC, DMA interrupt of LCD driver and DMA interrupt of memory-to-memory.
- ✓ 4 interrupt priority levels

➢ **Digital peripherals**

- ✓ 5 16-bit timers: timer0, timer1, timer2, timer3, timer4, where the mode 3 of timer 0 has the Non-Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
- ✓ 2 high speed USARTs: USART1, USART2, whose maximum baudrate clock may be FOSC/4. The following modes are supported, synchronous serial port mode, asynchronous serial port mode, SPI mode, LIN mode, infrared mode (IrDA), smart card mode (ISO7816)
- ✓ 2 high speed UARTs: UART3, UART4, whose maximum baudrate clock may be FOSC/4
- ✓ 2 groups of enhanced PWM, which can realize 8 channels(4 groups complementary symmetry) control signals with dead time, and support external fault detection function.
- ✓ SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
- ✓ I^{2}C: Master mode or slave mode are supported.
- ✓ ICE: Hardware support emulation.
- ✓ RTC: Support year, month, day, hour, minute, second, sub-second (1/128 second). And supports clock interrupt and a set of alarm clocks.
- ✓ USB: USB2.0/USB1.1 compatible with full-speed USB, 6 bidirectional endpoints, support 4 endpoint transfer modes (control transfer, interrupt transfer, bulk transfer and isochronous transfer), each endpoint has a 64-byte buffer.
- ✓ I2S: Audio interface
- ✓ CAN: Two independent CAN 2.0 control units.
- ✓ LIN: An independent LIN control unit (supports versions 1.3 and 2.1), USART1 and USART2 can support two sets of LIN.
- ✓ MDU32: Hardware 32-bit Multiplier and Divider which supports 32-bit divided by 32-bit, 32-bit multiplied by 32-bit operations.
- ✓ I/O port interrupt: All I/Os support interrupts, each group of I/O interrupts has an independent interrupt entry address, all I/O interrupts can support 4 types interrupt mode: high level interrupt, low level interrupt, rising edge interrupt, falling edge interrupt. Provides 4 levels of interrupt priority and supports power-down wake-up function.
- ✓ LCD dirver: support 8080 and 6800 interface, and support 8-bit and 16-bit data width.
- ✓ DMA: Support SPI shift to receive data to memory, SPI shift to send data from memory, I2C send data from memory, I2C receive data to memory, USART 1/2 and UART 3/4 receive data to memory, USART 1/2 and UART 3/4 send data from memory, ADC automatically sample data to memory (calculate average value at the same time), LCD driver send data from memory, and copy data from memory to memory
- ✓ Hardware digital ID: support 32 bytes

➢ **Analog peripherals**

- ✓ Ultra high speed ADC which supports 12-bit precision 15 channels (channel 0 to channel 14) analog-to-digital conversion. ADC channel 15 is used to test the internal reference voltage. (The default internal reference voltage is 1.19V when the chip is shipped, the error is ±1%)
- ✓ Comparator. A set of comparator

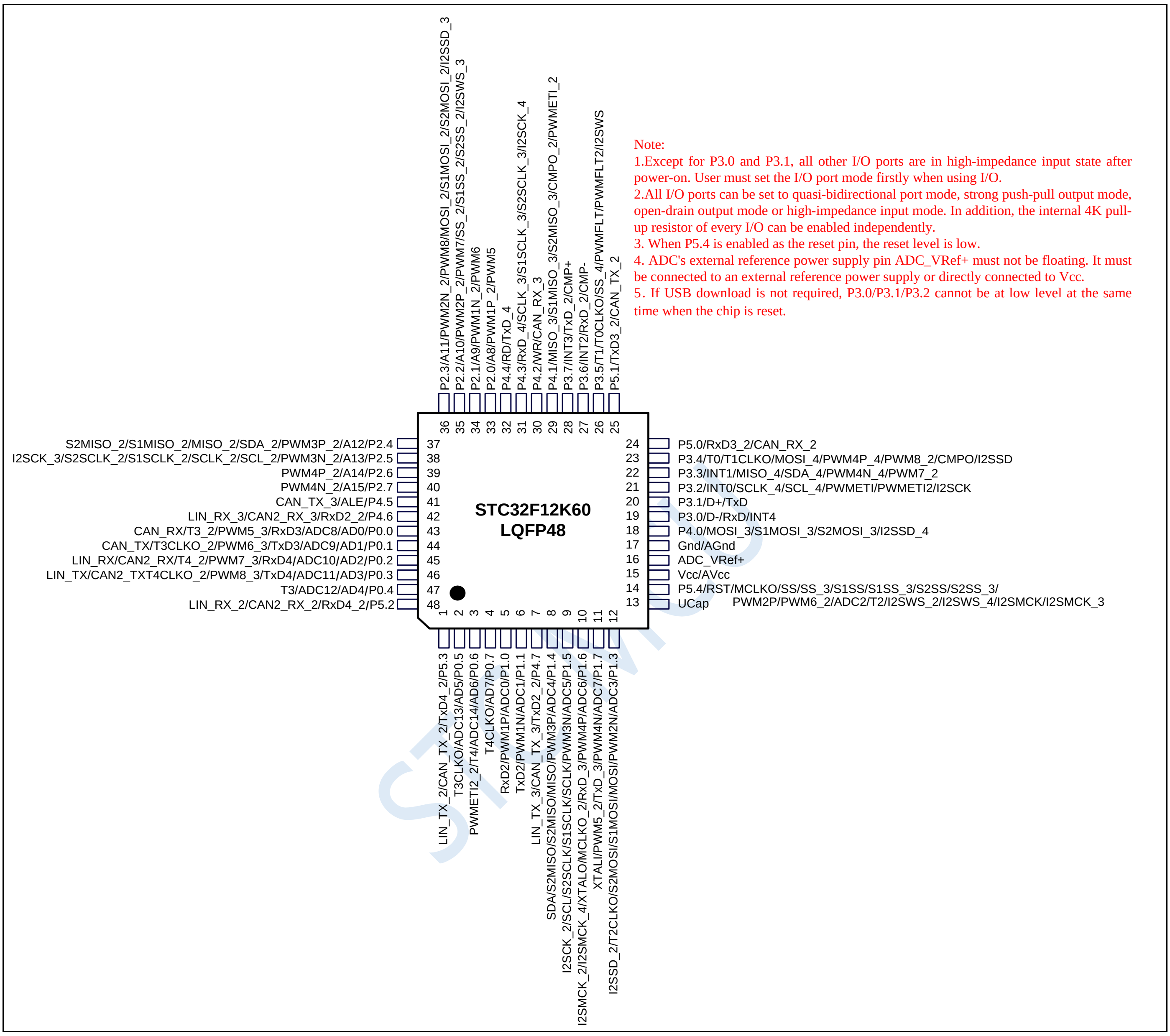
➢ **GPIO**

- ✓ Up to 44 GPIOs: P0.0~P0.7, P1.0~P1.7(No P1.2), P2.0~P2.7, P3.0~P3.7, P4.0~P4.7, P5.0~P5.4
- ✓ 4 modes for all GPIOs: quasi_bidirectional mode, push-pull outputmode, open drain mode, high-impedance input mode
- ✓ Except for P3.0 and P3.1, all other I/O ports are in a high-impedance state after power-on. User must set the I/O ports mode before using them. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.

➢ **Package**

✓ LQFP48, LQFP32, PDIP40

2.11.2 Pinouts


STC32F12K60
LQFP48
1 LIN_TX_2/CAN_TX_2/TxD4_2/P5.3
2 T3CLKO/ADC13/AD5/P0.5
3 PWMETI2_2/T4/ADC14/AD6/P0.6
4 T4CLKO/AD7/P0.7
5 RxD2/PWM1P/ADC0/P1.0
6 TxD2/PWM1N/ADC1/P1.1
7 LIN_TX_3/CAN_TX_3/TxD2_2/P4.7
8 SDA/S2MISO/S2MISO/MISO/PWM3P/ADC4/P1.4
9 I2SCK_2/SCL/S2SCLK/S1SCLK/SCLK/PWM3N/ADC5/P1.5
10 I2SMCK_2/I2SMCK_4/XTALO/MCLKO_2/RxD_3/PWM4P/ADC6/P1.6
11 XTALI/PWM5_2/TxD_3/PWM4N/ADC7/P1.7
12 I2SSD_2/T2CLKO/S2MOSI/S1MOSI/MOSI/PWM2N/ADC3/P1.3
13 UCap
14 P5.4/RST/MCLKO/SS/SS_3/S1SS/S1SS_3/S2SS/S2SS_3/PWM2P/PWM6_2/ADC2/T2/I2SWS_2/I2SWS_4/I2SMCK/I2SMCK_3
15 Vcc/AVcc
16 ADC_VRef+
17 Gnd/AGnd
18 P4.0/MOSI_3/S1MOSI_3/S2MOSI_3/I2SSD_4
19 P3.0/D-/RxD/INT4
20 P3.1/D+/TxD
21 P3.2/INT0/SCLK_4/SCL_4/PWMETI/PWMETI2/I2SCK
22 P3.3/INT1/MISO_4/SDA_4/PWM4N_4/PWM7_2
23 P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO/I2SSD
24 P5.0/RxD3_2/CAN_RX_2
25 P5.1/TxD3_2/CAN_TX_2
26 P3.5/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2/I2SWS
27 P3.6/INT2/RxD_2/CMP-
28 P3.7/INT3/TxD_2/CMP+
29 P4.1/MISO_3/S1MISO_3/S2MISO_3/CMPO_2/PWMETI_2
30 P4.2/WR/CAN_RX_3
31 P4.3/RxD_4/SCLK_3/S1SCLK_3/S2SCLK_3/I2SCK_4
32 P4.4/RD/TxD_4
33 P2.0/A8/PWM1P_2/PWM5
34 P2.1/A9/PWM1N_2/PWM6
35 P2.2/A10/PWM2P_2/PWM7/SS_2/S1SS_2/S2SS_2/I2SWS_3
36 P2.3/A11/PWM2N_2/PWM8/MOSI_2/S1MOSI_2/S2MOSI_2/I2SSD_3
37 S2MISO_2/S1MISO_2/MISO_2/SDA_2/PWM3P_2/A12/P2.4
38 I2SCK_3/S2SCLK_2/S1SCLK_2/SCLK_2/SCL_2/PWM3N_2/A13/P2.5
39 PWM4P_2/A14/P2.6
40 PWM4N_2/A15/P2.7
41 CAN_TX_3/ALE/P4.5
42 LIN_RX_3/CAN2_RX_3/RxD2_2/P4.6
43 CAN_RX/T3_2/PWM5_3/RxD3/ADC8/AD0/P0.0
44 CAN_TX/T3CLKO_2/PWM6_3/TxD3/ADC9/AD1/P0.1
45 LIN_RX/CAN2_RX/T4_2/PWM7_3/RxD4/ADC10/AD2/P0.2
46 LIN_TX/CAN2_TXT4CLKO_2/PWM8_3/TxD4/ADC11/AD3/P0.3
47 T3/ADC12/AD4/P0.4
48 LIN_RX_2/CAN2_RX_2/RxD4_2/P5.2

Note:

1.Except for P3.0 and P3.1, all other I/O ports are in high-impedance input state after power-on. User must set the I/O port mode firstly when using I/O.

2.All I/O ports can be set to quasi-bidirectional port mode, strong push-pull output mode, open-drain output mode or high-impedance input mode. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.

3. When P5.4 is enabled as the reset pin, the reset level is low.

4. ADC's external reference power supply pin ADC_VRef+ must not be floating. It must be connected to an external reference power supply or directly connected to Vcc.

5. If USB download is not required, P3.0/P3.1/P3.2 cannot be at low level at the same time when the chip is reset.

2.11.3 Pin descriptions

Pin number		name	type	description
LQFP48				
1		P5.3	I/O	Standard IO port
		TxD4_2	O	Transmit pin of UART 4
		CAN2_TX_2	O	Transmit pin of CAN2
		LIN_TX_2	O	Transmit pin of LIN
2		P0.5	I/O	Standard IO port
		AD5	I	Address/data bus
		ADC13	I	ADC analog input 13
		T3CLKO	O	Clock out of timer 3
3		P0.6	I/O	Standard IO port
		AD6	I	Address/data bus
		ADC14	I	ADC analog input 14
		T4	I	Timer4 external input
		PWMFLT2_2	I	Enhance PWM external anomaly detection pin 2
4		P0.7	I/O	Standard IO port
		AD7	I	Address/data bus
		T4CLKO	O	Clock out of timer 4
5		P1.0	I/O	Standard IO port
		ADC0	I	ADC analog input 0
		PWM1P	I/O	Capture of external signal/ Positive of PWMA pulse output
		RxD2	I	Input of USART2
6		P1.1	I/O	Standard IO port
		ADC1	I	ADC analog input 1
		PWM1N	I/O	Capture of external signal/Negative of PWMA pulse output
		TxD2	I	Transmit pin of USART 2
7		P4.7	I/O	Standard IO port
		TxD2_2	I	Transmit pin of USART 2
		CAN2_TX_3	O	Transmit pin of CAN2
		LIN_TX_3	O	Transmit pin of LIN
8		P1.4	I/O	Standard IO port
		ADC4	I	ADC analog input 4
		PWM3P	I/O	Capture of external signal/Positive of PWM3 pulse output
		MISO	I/O	Master Iutput/Slave Onput of SPI
		S1MISO	I/O	Master Iutput/Slave Onput of USART1-SPI
		S2MISO	I/O	Master Iutput/Slave Onput of USART2-SPI
		SDA	I/O	Serial data line of I2C
9		P1.5	I/O	Standard IO port
		ADC5	I	ADC analog input 5
		PWM3N	I/O	Capture of external signal/Negative of PWM3 pulse output
		SCLK	I/O	Serial Clock of SPI
		S1SCLK	I/O	Serial Clock of USART1-SPI
		S2SCLK	I/O	Serial Clock of USART2-SPI
		SCL	I/O	Serial Clock line of I2C
		I2SCK_2	I/O	Serial Clock line of I2S
10		P1.6	I/O	Standard IO port
		ADC6	I	ADC analog input 6
		RxD_3	I	Input of USART 1
		PWM4P	I/O	Capture of external signal/Positive of PWM4 pulse output
		MCLKO_2	O	Main clock output
		XTALO	O	Connect to external oscillator
		I2SMCK_2	O	I2S main clock line
		I2SMCK_4	O	I2S main clock line

Pin number		name	type	description
LQFP48				
11		P1.7	I/O	Standard IO port
		ADC7	I	ADC analog input 7
		TxD_3	O	Transmit pin of USART 1
		PWM4N	I/O	Capture of external signal/Negative of PWM4 pulse output
		PWM5_2	I/O	Capture of external signal/Pulse output of PWM5
		XTALI	I	Connect to external oscillator
12		P1.3	I/O	Standard IO port
		ADC3	I	ADC analog input 3
		MOSI	I/O	Master Output/Slave Input of SPI
		S1MOSI	I/O	Master Output/Slave Input of USART1-SPI
		S2MOSI	I/O	Master Output/Slave Input of USART2-SPI
		PWM2N	I/O	Capture of external signal/Negative of PWM2 pulse output
		T2CLKO	O	Clock out of timer 2
		I2SSD_2	I/O	Data line of I2S
13		UCAP	I	USB core power regulator pin
14		P5.4	I/O	Standard IO port
		RST	I	Reset pin (low level reset)
		MCLKO	O	Main clock output
		SS_3	I	Slave selection of SPI (it is output with regard to master)
		SS	I	Slave selection of SPI (it is output with regard to master)
		S1SS_3	I	Slave selection of USART1-SPI (it is output with regard to master)
		S1SS	I	Slave selection of USART1-SPI (it is output with regard to master)
		S2SS	I	Slave selection of USART2-SPI (it is output with regard to master)
		PWM2P	I/O	Capture of external signal/Positive of PWM2 pulse output
		PWM6_2	I/O	Capture of external signal/Pulse output of PWM6
		T2	I	T2
		ADC2	I	ADC2
		I2SWS_2	I/O	I2S channel selection line
		I2SWS_4	I/O	I2S channel selection line
		I2SMCK	O	I2S main clock line
		I2SMCK_3	O	I2S main clock line
15		Vcc	Vcc	Power Supply
		AVcc	Vcc	ADC Power Supply
16		VRef+	I	Reference voltage pin of ADC
17		Gnd	Gnd	Ground
		AGnd	Gnd	ADC Ground
		VRef-	I	Reference voltage ground pin of ADC
18		P4.0	I/O	Standard IO port
		MOSI_3	I/O	Master Output/Slave Input of SPI
		S1MOSI_3	I/O	Master Output/Slave Input of USART1-SPI
		S2MOSI_3	I/O	Master Output/Slave Input of USART2-SPI
		I2SSD_4	I/O	Data line of I2S
19		P3.0	I/O	Standard IO port
		D-	I/O	USB data line
		RxD	I	Input of USART1
		INT4	I	External interrupt4
20		P3.1	I/O	Standard IO port
		D+	I/O	USB data line
		TxD	O	Transmit pin of USART1

Pin number		name	type	description
LQFP48				

21		P3.2	I/O	Standard IO port
		INT0	I	External interrupt0
		SCLK_4	I/O	Serial Clock of SPI
		SCL_4	I/O	Serial Clock line of I2C
		PWMETI	I	PWM external trigger input pin
		PWMET2	I	PWM external trigger input pin 2
		I2SCK	I/O	Clock line of I2S
22		P3.3	I/O	Standard IO port
		INT1	I	External interrupt1
		MISO_4	I/O	Master Iutput/Slave Onput of SPI
		SDA_4	I/O	Serial data line of I2C
		PWM4N_4	I/O	Capture of external signal/Negative of PWM4 pulse output
		PWM7_2	I/O	Capture of external signal/Pulse output of PWM7
23		P3.4	I/O	Standard IO port
		T0	I	Timer0 external input
		T1CLKO	O	Clock out of timer 1
		MOSI_4	I/O	Master Output/Slave Input of SPI
		PWM4P_4	I/O	Capture of external signal/Positive of PWM4 pulse output
		PWM8_2	I/O	Capture of external signal/Pulse output of PWM8
		CMPO	O	Output of comparator
		I2SSD	I/O	Data line of I2S
24		P5.0	I/O	Standard IO port
		RxD3_2	I	Input of UART3
		CMP+_2	I	Positive input of comparator
		CAN_RX_2	I	Receive pin of CAN
25		P5.1	I/O	Standard IO port
		TxD3_2	O	Transmit pin of UART 3
		CMP+_3	I	Positive input of comparator
		CAN_TX_2	I	Transmit pin of CAN
26		P3.5	I/O	Standard IO port
		T1	I	Timer1 external input
		T0CLKO	O	Clock out of timer 0
		SS_4	I	Slave selection of SPI (it is output with regard to master)
		PWMFLT	I	Enhance PWMA external anomaly detection pin
		I2SWS	I/O	I2S channel selection line
27		P3.6	I/O	Standard IO port
		INT2	I	External interrupt2
		RxD_2	I	Input of USART1
		CMP-	I	Negative input of comparator
28		P3.7	I/O	Standard IO port
		INT3	I	External interrupt3
		TxD_2	O	Transmit pin of USART 1
		CMP+	I	Positive input of comparator
29		P4.1	I/O	Standard IO port
		MISO_3	I/O	Master Iutput/Slave Onput of SPI
		S1MISO_3	I/O	Master Iutput/Slave Onput of USART1-SPI
		S2MISO_3	I/O	Master Iutput/Slave Onput of USART2-SPI
		CMPO_2	O	Output of comparator
		PWMETI_3	I	PWM external trigger input pin
30		P4.2	I/O	Standard IO port
		WR	O	WRITE signal of external bus
		CAN_RX_3	I	Receive pin of CAN

Pin number		name	type	description
LQFP48				
31		P4.3	I/O	Standard IO port
		RxD_4	I	Input of USART1
		SCLK_3	I/O	Serial Clock of SPI
		S1SCLK_3	I/O	Serial Clock of USART1-SPI
		S2SCLK_3	I/O	Serial Clock of USART2-SPI
		I2SCK_4	I/O	Clock line of I2S
32		P4.4	I/O	Standard IO port
		RD	O	READ signal of external bus
		TxD_4	O	Transmit pin of USART 1
33		P2.0	I/O	Standard IO port
		A8	I	Address bus
		PWM1P_2	I/O	Capture of external signal/Positive of PWMA pulse output
		PWM5	I/O	Capture of external signal/Pulse output of PWM5
34		P2.1	I/O	Standard IO port
		A9	I	Address bus
		PWM1N_2	I/O	Capture of external signal/Negative of PWMA pulse output
		PWM6	I/O	Capture of external signal/Pulse output of PWM6
35		P2.2	I/O	Standard IO port
		A10	I	Address bus
		SS_2	I	Slave selection of SPI (it is output with regard to master)
		S1SS_2	I	Slave selection of USART1-SPI (it is output with regard to master)
		S2SS_2	I	Slave selection of USART2-SPI (it is output with regard to master)
		PWM2P_2	I/O	Capture of external signal/Positive of PWM2 pulse output
		PWM7	I/O	Capture of external signal/Pulse output of PWM7
		I2SWS_3	I/O	I2S channel selection line
36		P2.3	I/O	Standard IO port
		A11	I	Address bus
		MOSI_2	I/O	Master Output/Slave Input of SPI
		S1MOSI_2	I/O	Master Output/Slave Input of USART1-SPI
		S2MOSI_2	I/O	Master Output/Slave Input of USART2-SPI
		PWM2N_2	I/O	Capture of external signal/Negative of PWM2 pulse output
		PWM8	I/O	Capture of external signal/Pulse output of PWM8
		I2SSD_3	I/O	Data line of I2S
37		P2.4	I/O	Standard IO port
		A12	I	Address bus
		MISO_2	I/O	Master Iutput/Slave Onput of SPI
		S1MISO_2	I/O	Master Iutput/Slave Onput of USART1-SPI
		S2MISO_2	I/O	Master Iutput/Slave Onput of USART2-SPI
		SDA_2	I/O	Serial data line of I2C
		PWM3P_2	I/O	Capture of external signal/Positive of PWM3 pulse output
38		P2.5	I/O	Standard IO port
		A13	I	Address bus
		SCLK_2	I/O	Serial Clock of SPI
		S1SCLK_2	I/O	Serial Clock of USART1-SPI
		S2SCLK_2	I/O	Serial Clock of USART2-SPI
		SCL_2	I/O	Serial Clock line of I2C
		PWM3N_2	I/O	Capture of external signal/Negative of PWM3 pulse output
		I2SCK_3	I/O	Clock line of I2S
39		P2.6	I/O	Standard IO port
		A14	I	Address bus
		PWM4P_2	I/O	Capture of external signal/Positive of PWM4 pulse output

Pin number		name	type	description
LQFP48				
40		P2.7	I/O	Standard IO port
		A15	I	Address bus
		PWM4N_2	I/O	Capture of external signal/Negative of PWM4 pulse output
41		P4.5	I/O	Standard IO port
		ALE	O	Address Latch Enable signal
		CAN_TX_3	O	Transmit pin of CAN
42		P4.6	I/O	Standard IO port
		RxD2_2	I	Input of USART2
		CAN2_RX_3	I	Receive pin of CAN2
		LIN_RX_3	I	Receive pin of LIN
43		P0.0	I/O	Standard IO port
		AD0	I	Address/data bus
		ADC8	I	ADC analog input 8
		RxD3	I	Input of UART3
		PWM5_3	I/O	Capture of external signal/Pulse output of PWM5
		CAN_RX	I	Receive pin of CAN
44		P0.1	I/O	Standard IO port
		AD1	I	Address/data bus
		ADC9	I	ADC analog input 9
		TxD3	O	Transmit pin of UART 3
		PWM6_3	I/O	Capture of external signal/Pulse output of PWM6
		CAN_TX	O	Transmit pin of CAN
45		P0.2	I/O	Standard IO port
		AD2	I	Address/data bus
		ADC10	I	ADC analog input 10
		RxD4	I	Input of UART4
		PWM7_3	I/O	Capture of external signal/Pulse output of PWM7
		CAN2_RX	I	Receive pin of CAN2
		LIN_RX	I	Receive pin of LIN
46		P0.3	I/O	Standard IO port
		AD3	I	Address/data bus
		ADC11	I	ADC analog input 11
		TxD4	O	Transmit pin of UART 4
		PWM8_3	I/O	Capture o f external signal/Pulse output of PWM8
		CAN2_TX	O	Transmit pin of CAN2
		LIN_TX	O	Transmit pin of LIN
47		P0.4	I/O	Standard IO port
		AD4	I	Address/data bus
		ADC12	I	ADC analog input 12
		T3	I	Timer3 external input
48		P5.2	I/O	Standard IO port
		RxD4_2	I	Input of UART4
		CAN2_RX_2	I	Receive pin of CAN2
		LIN_RX_2	I	Receive pin of LIN

3 Function pins switch

Some special peripherals of STC8H series of microcontrollers can be switched among several I/O pins to realize one peripheral used as multiple device through time-sharing, such as UART, SPI, PWM, I2C and bus control pins.

3.1 Register related to function pin switch

Symbol	Description	Address	Bit Address and Symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
P_SW1	Peripheral port switch register 1	A2H	S1_S[1:0]		-	-	SPI_S[1:0]		0	-	nnxx,000x
P_SW2	Peripheral port switch register 2	BAH	EAXFR	–	I2C_S[1:0]		CMPO_S	S4_S	S3_S	S2_S	0x00,0000

Symbol	Description	Address	Bit Address and Symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
MCLKOCR	Main clock output control register	FE05H	MCLKO_S	MCLKODIV[6:0]							0000,0000
PWMA_PS	PWM1 switch register	FEB2H	C4PS[1:0]		C3PS[1:0]		C2PS[1:0]		C1PS[1:0]		0000,0000
PWMB_PS	PWM2 switch register	FEB6H	C8PS[1:0]		C7PS[1:0]		C6PS[1:0]		C5PS[1:0]		0000,0000
PWMA_ETRPS	PWMA ETR select register	FEB0H						BRKBPS	ETRAPS[1:0]		xxxx,xxx0
PWMB_ETRPS	PWMB ETR select register	FEB4H						BRKBPS	ETRAPS[1:0]		xxxx,xxx0
T3T4PIN	T3/T4 select register	FEACH	-	-	-	-	-	-	-	T3T4SEL	xxxx,xxx0

3.1.1Peripheral port switch register 1(P_SW1)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P_SW1	A2H	S1_S[1:0]				SPI_S[1:0]		0	-

S1_S[1:0]：USART1 pin selection bits

S1_S[1:0]	RxD	TxD
00	P3.0	P3.1
01	P3.6	P3.7
10	P1.6	P1.7
11	P4.3	P4.4

SPI_S[1:0]：SPI pin selection bits

SPI_S[1:0]	SS	MOSI	MISO	SCLK
00	P1.2/P5.4	P1.3	P1.4	P1.5
01	P2.2	P2.3	P2.4	P2.5
10	P5.4	P4.0	P4.1	P4.3
11	P3.5	P3.4	P3.3	P3.2

3.1.2Peripheral port switch register 2(P_SW2)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P_SW2	BAH	EAXFR	–	I2C_S[1:0]		CMPO_S	S4_S	S3_S	S2_S

EAXFR：Extended RAM area Special Function Register (XFR) access control register

0: Access to XFR is prohibited

1: Enable access to XFR.

When XFR needs to be accessed, EAXFR must be set to 1 before XFR can be read or written properly

I2C_S[1:0]：I^{2}C pin selection bits

I2C_S[1:0]	SCL	SDA
00	P1.5	P1.4
01	P2.5	P2.4
10	P7.7	P7.6
11	P3.2	P3.3

CMPO_S：Comparator output pin selection bit

CMPO_S	CMPO
0	P3.4
1	P4.1

S4_S：UART4 pin selection bit

S4_S	RxD4	TxD4
0	P0.2	P0.3
1	P5.2	P5.3

S3_S：UART3 pin selection bit

S3_S	RxD3	TxD3
0	P0.0	P0.1
1	P5.0	P5.1

S2_S：USART2 pin selection bit

S2_S	RxD2	TxD2
0	P1.0	P1.1
1	P4.6	P4.7

3.1.3 Clock selection register(MCLKOCR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
MCLKOCR	FE05H	MCLKO_S	MCLKODIV[6:0]						

MCLKO_S：Main clock out pin selection bit

MCLKO_S	MCLKO
0	P5.4
1	P1.6

3.1.4 T3/T4 selection register(T3T4PIN)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
T3T4PIN	FEACH	-	-	-	-	-	-	-	T3T4SEL

T3T4SEL：T3/T3CLKO/T4/T4CLKO pin selection bit

T3T4SEL	T3	T3CLKO	T4	T4CLKO
0	P0.4	P0.5	P0.6	P0.7
1	P0.0	P0.1	P0.2	P0.3

3.1.5 Advanced PWM selection register(PWMx_PS)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_PS	FEB2H	C4PS[1:0]		C3PS[1:0]		C2PS[1:0]		C1PS[1:0]	
PWMB_PS	FEB6H	C8PS[1:0]		C7PS[1:0]		C6PS[1:0]		C5PS[1:0]	

C1PS[1:0]：Advanced PWM channel 1 out pin selection bit

C1PS[1:0]	PWM1P	PWM1N
00	P1.0	P1.1
01	P2.0	P2.1
10	P6.0	P6.1
11	-	-

C2PS[1:0]：Advanced PWM channel 2 out pin selection bit

C2PS[1:0]	PWM2P	PWM2N
00	P1.2/P5.4	P1.3
01	P2.2	P2.3
10	P6.2	P6.3
11	-	-

C3PS[1:0]：Advanced PWM channel 3 out pin selection bit

C3PS[1:0]	PWM3P	PWM3N
00	P1.4	P1.5
01	P2.4	P2.5
10	P6.4	P6.5
11	-	-

C4PS[1:0]：Advanced PWM channel 4 out pin selection bit

C4PS[1:0]	PWM4P	PWM4N
00	P1.6	P1.7
01	P2.6	P2.7
10	P6.6	P6.7
11	P3.4	P3.3

C5PS[1:0]：Advanced PWM channel 5 out pin selection bit

C5PS[1:0]	PWM5
00	P2.0
01	P1.7
10	P0.0
11	P7.4

C6PS[1:0]：Advanced PWM channel 6 out pin selection bit

C6PS[1:0]	PWM6
00	P2.1
01	P5.4
10	P0.1
11	P7.5

C7PS[1:0]：Advanced PWM channel 7 out pin selection bit

C7PS[1:0]	PWM7
00	P2.2
01	P3.3
10	P0.2
11	P7.6

C8PS[1:0]：Advanced PWM channel 8 out pin selection bit

C8PS[1:0]	PWM8
00	P2.3
01	P3.4
10	P0.3
11	P7.7

3.1.6 Advanced PWM Pin selection register(PWMx_ETRPS)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_ETRPS	FEB0H						BRKAPS	ETRAPS [1:0]	
PWMA_ETRPS	FEB4H						BRKBPS	ETRBPS [1:0]	

ETRAPS [1:0]：Advanced PWMA External trigger leg ERI select bit

ETRAPS [1:0]	PWMETI
00	P3.2
01	P4.1
10	P7.3
11	-

ETRBPS [1:0]：Advanced PWMB External trigger leg ERI select bit

ETRBPS [1:0]	PWMET2
00	P3.2
01	P0.6
10	-
11	-

BRKAPS [1:0]：Advanced PWMA brake pin PWMFLT selector bit

BRKAPS [1:0]	PWMFLT

00	P3.5
01	Comparator output

BRKBPS [1:0]： Advanced PWMB brake pin PWMFLT selector bit

BRKBPS [1:0]	PWMFLT2
00	P3.5
01	Comparator output

3.2 Example Routines

3.2.1 USART1 switch

C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"

sfr     P_SW1     =    0xa2;

sfr     P0M1      =    0x93;
sfr     P0M0      =    0x94;
sfr     P1M1      =    0x91;
sfr     P1M0      =    0x92;
sfr     P2M1      =    0x95;
sfr     P2M0      =    0x96;
sfr     P3M1      =    0xb1;
sfr     P3M0      =    0xb2;
sfr     P4M1      =    0xb3;
sfr     P4M0      =    0xb4;
sfr     P5M1      =    0xc9;
sfr     P5M0      =    0xca;

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P_SW1 = 0x00;                             //RXD/P3.0, TXD/P3.1
//      P_SW1 = 0x40;                             //RXD_2/P3.6, TXD_2/P3.7
//      P_SW1 = 0x80;                             //RXD_3/P1.6, TXD_3/P1.7
//      P_SW1 = 0xc0;                             //RXD_4/P4.3, TXD_4/P4.4

        while (1);
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz

P_SW1       DATA      0A2H

P0M1        DATA      093H
P0M0        DATA      094H
P1M1        DATA      091H
P1M0        DATA      092H
P2M1        DATA      095H
```

```
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN

                ORG             0100H
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             P_SW1,#00H                      ;RXD/P3.0, TXD/P3.1
;               MOV             P_SW1,#40H                      ;RXD_2/P3.6, TXD_2/P3.7
;               MOV             P_SW1,#80H                      ;RXD_3/P1.6, TXD_3/P1.7
;               MOV             P_SW1,#0C0H                     ;RXD_4/P4.3, TXD_4/P4.4

                SJMP            $

                END
```

3.2.2 USART2 switch

C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"

sfr         P_SW2           =       0xba;

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;
```

```
void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P_SW2 = 0x00;                                   //RXD2/P1.0, TXD2/P1.1
//      P_SW2 = 0x01;                                   //RXD2_2/P4.6, TXD2_2/P4.7

        while (1);
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz

P_SW2           DATA            0BAH

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN

                ORG             0100H
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             P_SW2,#00H                      ;RXD2/P1.0, TXD2/P1.1
```

```
;               MOV         P_SW2,#01H                  ;RXD2_2/P4.0, TXD2_2/P4.2

                SJMP        $

                END
```

3.2.3 UART3 switch

C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"

sfr         P_SW2           =       0xba;

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P_SW2 = 0x00;                                   //RXD3/P0.0, TXD3/P0.1
//      P_SW2 = 0x02;                                   //RXD3_2/P5.0, TXD3_2/P5.1

        while (1);
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz

P_SW2           DATA        0BAH

P0M1            DATA        093H
P0M0            DATA        094H
P1M1            DATA        091H
```

```
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN

                ORG             0100H
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             P_SW2,#00H                      ;RXD3/P0.0, TXD3/P0.1
;               MOV             P_SW2,#02H                      ;RXD3_2/P5.0, TXD3_2/P5.1

                SJMP            $

                END
```

3.2.4 UART4 switch

C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"

sfr         P_SW2           =       0xba;

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;
```

```
void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P_SW2 = 0x00;                                   //RXD4/P0.2, TXD4/P0.3
//      P_SW2 = 0x04;                                   //RXD4_2/P5.2, TXD4_2/P5.3

        while (1);
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz

P_SW2           DATA            0BAH

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN

                ORG             0100H
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             P_SW2,#00H              ;RXD4/P0.2, TXD4/P0.3
```

```
;               MOV         P_SW2,#04H                  ;RXD4_2/P5.2, TXD4_2/P5.3

                SJMP        $

                END
```

3.2.5 SPI switch

C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"

sfr         P_SW1           =       0xa2;

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P_SW1 = 0x00;                           //SS/P1.2, MOSI/P1.3, MISO/P1.4, SCLK/P1.5
//      P_SW1 = 0x04;                           //SS_2/P2.2, MOSI_2/P2.3, MISO_2/P2.4, SCLK_2/P2.5
//      P_SW1 = 0x08;                           //SS_3/P5.4, MOSI_3/P4.0, MISO_3/P4.1, SCLK_3/P4.3
//      P_SW1 = 0x0c;                           //SS_4/P3.5, MOSI_4/P3.4, MISO_4/P3.3, SCLK_4/P3.2

        while (1);
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz

P_SW1           DATA        0A2H

P0M1            DATA        093H
```

```
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN

                ORG             0100H
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             P_SW1,#00H                      ;SS/P1.2, MOSI/P1.3, MISO/P1.4, SCLK/P1.5
;               MOV             P_SW1,#04H                      ;SS_2/P2.2, MOSI_2/P2.3, MISO_2/P2.4, SCLK_2/P2.5
;               MOV             P_SW1,#08H                      ;SS_3/P5.4, MOSI_3/P4.0, MISO_3/P4.1, SCLK_3/P4.3
;               MOV             P_SW1,#0CH                      ;SS_4/P3.5, MOSI_4/P3.4, MISO_4/P3.3, SCLK_4/P3.2

                SJMP            $

                END
```

3.2.6 I2C switch

C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"

sfr         P_SW2           =       0xba;

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0          =     0xb4;
sfr         P5M1          =     0xc9;
sfr         P5M0          =     0xca;

void main()
{
            P0M0 = 0x00;
            P0M1 = 0x00;
            P1M0 = 0x00;
            P1M1 = 0x00;
            P2M0 = 0x00;
            P2M1 = 0x00;
            P3M0 = 0x00;
            P3M1 = 0x00;
            P4M0 = 0x00;
            P4M1 = 0x00;
            P5M0 = 0x00;
            P5M1 = 0x00;

            P_SW2 = 0x00;                           //SCL/P1.5, SDA/P1.4
//          P_SW2 = 0x10;                           //SCL_2/P2.5, SDA_2/P2.4
//          P_SW2 = 0x20;                           //SCL_3/P7.7, SDA_3/P7.6
//          P_SW2 = 0x30;                           //SCL_4/P3.2, SDA_4/P3.3

            while (1);
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz

P_SW2       DATA        0BAH

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN

            ORG         0100H
MAIN:
            MOV         SP, #5FH
            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
            MOV         P3M1, #00H
```

```
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             P_SW2,#00H                ;SCL/P1.5, SDA/P1.4
;               MOV             P_SW2,#10H                ;SCL_2/P2.5, SDA_2/P2.4
;               MOV             P_SW2,#20H                ;SCL_3/P7.7, SDA_3/P7.6
;               MOV             P_SW2,#30H                ;SCL_4/P3.2, SDA_4/P3.3

                SJMP            $
                END
```

3.2.7 Comparator output switch

C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"

sfr         P_SW2       =       0xba;

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P_SW2 = 0x00;                                   //CMPO/P3.4
//      P_SW2 = 0x08;                                   //CMPO_2/P4.1

        while (1);
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz

P_SW2           DATA            0BAH

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN

                ORG             0100H
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             P_SW2,#00H                      ;CMPO/P3.4
;               MOV             P_SW2,#08H                      ;CMPO_2/P4.1

                SJMP            $

                END
```

3.2.8 Main clock output switch

C language code

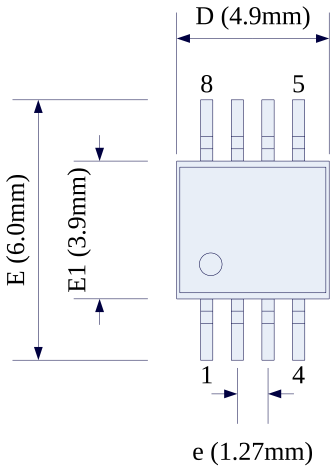
```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"

#define     CLKOCR          (*(unsigned char volatile xdata *)0xfe00)

sfr         P_SW2           =    0xba;

sfr         P0M1            =    0x93;
sfr         P0M0            =    0x94;
sfr         P1M1            =    0x91;
sfr         P1M0            =    0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P_SW2 = 0x80;
        CLKOCR = 0x04;                                  //IRC/4 output via MCLKO/P5.4
//      CLKOCR = 0x84;                                  //IRC/4 output via MCLKO_2/P1.6
        P_SW2 = 0x00;

        while (1);
}
```

Assembly code

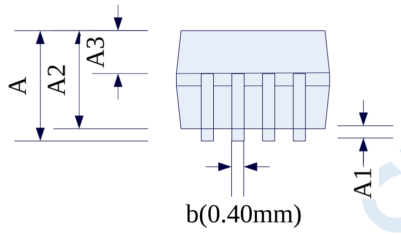
```
; Operating frequency for test is 11.0592MHz

P_SW2           DATA            0BAH

CLKOCR          EQU             0FE05H

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN

                ORG             0100H
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             P_SW2,#80H
                MOV             A,#04H                          ;IRC/4 output via MCLKO/P5.4
;               MOV             A,#84H                          ;IRC/4 output via MCLKO_2/P1.6
                MOV             DPTR,#CLKOCR
                MOVX            @DPTR,A
                MOV             P_SW2,#00H

                SJMP            $

                END
```

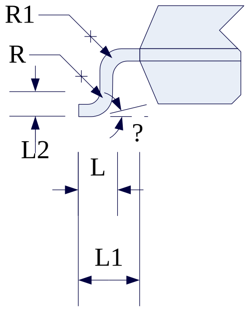
4 Package Dimensions

4.1 SOP8 Package mechanical data

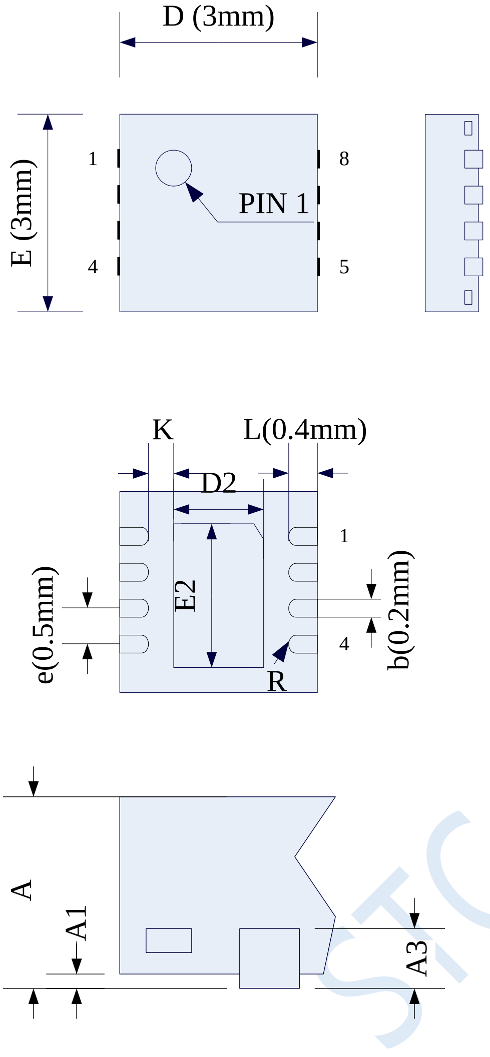


General size			
			mm
SYMBOL	MIN	TYP	MAX
A	1.35	1.60	1.75
A1	0.10	0.15	0.25
A2	1.25	1.45	1.65
A3	0.55	0.65	0.75
b	0.35	0.40	0.45
D	4.80	4.90	5.00
E	5.80	6.00	6.20
E1	3.80	3.90	4.00
e	1.27BSC		
L	0.45	0.60	0.80
L1	1.04REF		
L2	0.25BSC		
R1	0.07	-	-
R2	0.07	-	-





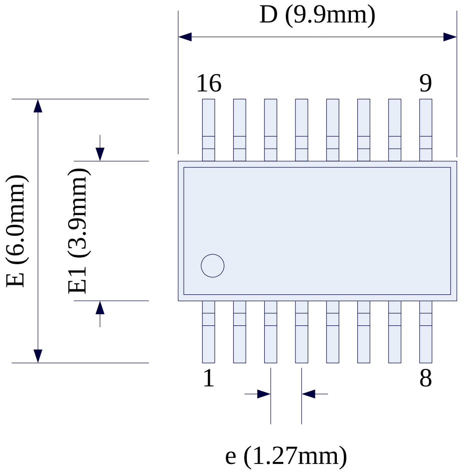
4.2 DFN8 Package mechanical data（3mm*3mm）



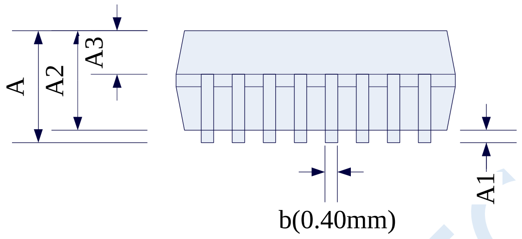
General size			
			mm
SYMBOL	MIN	TYP	MAX
A	0.70	0.75	0.80
A1	0	0.02	0.05
A2	0.50	0.55	0.60
A3	-	0.203REF	-
b	0.175	0.20	0.225
D	2.89	3.00	3.11
E	2.89	3.00	3.11
D2	1.65	1.70	1.75
E2	2.35	2.40	2.45
e	0.45	0.50	0.55
L	0.381	0.40	0.419
K	0.275REF		
R	0.125	-	-

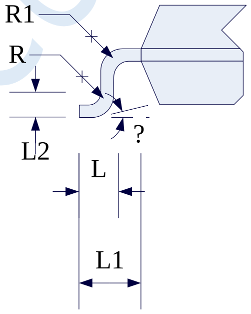
The back metal sheet (substrate) of STC's existing DFN8 packaged chip is not grounded inside the chip. It can be grounded or ungrounded on the user's PCB board, which will not affect the performance of the chip.

4.3 SOP16 Package mechanical data

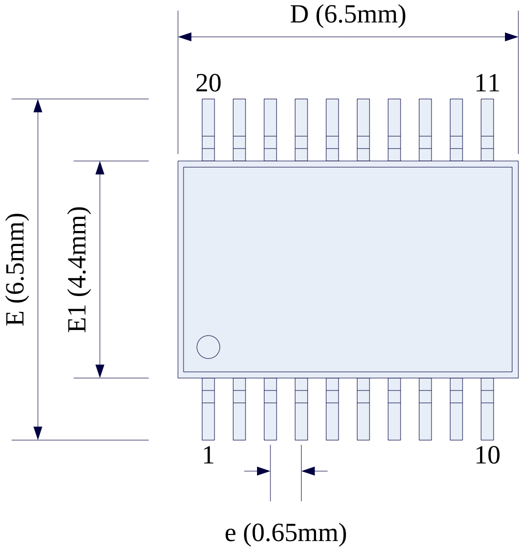


General size			
			mm
SYMBOL	MIN	TYP	MAX
A	1.35	1.60	1.75
A1	0.10	0.15	0.25
A2	1.25	1.45	1.65
A3	0.55	0.65	0.75
b	0.35	0.40	0.45
D	9.80	9.90	10.00
E	5.80	6.00	6.20
E1	3.80	3.90	4.00
e	1.27BSC		
L	0.45	0.60	0.80
L1	1.04REF		
L2	0.25BSC		
R1	0.07	-	-
R2	0.07	-	-

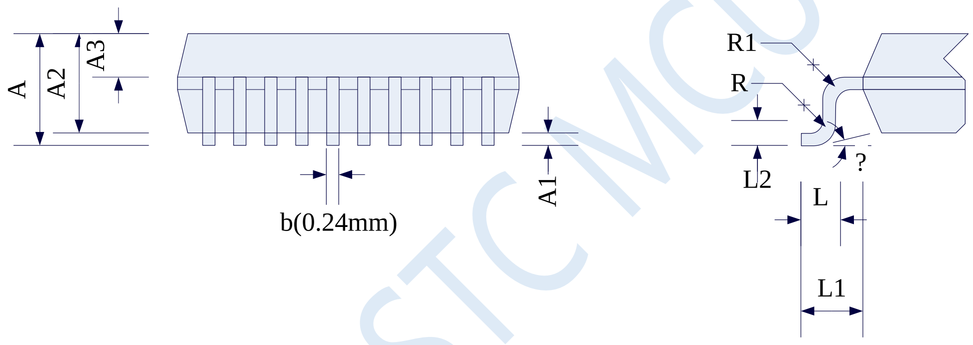




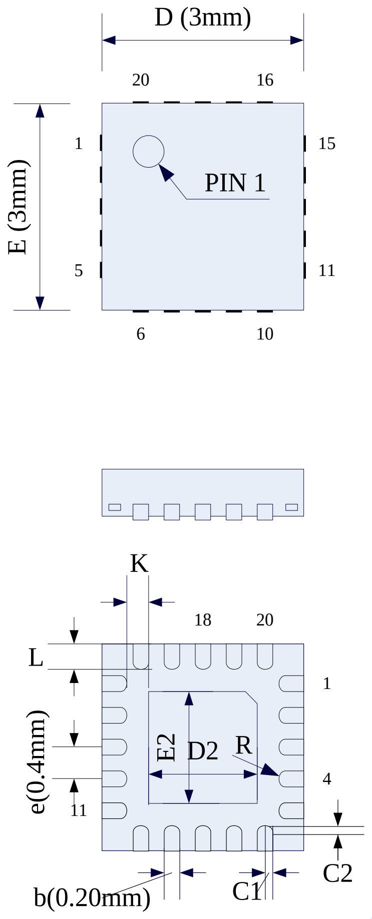
4.4 TSSOP20 Package mechanical data



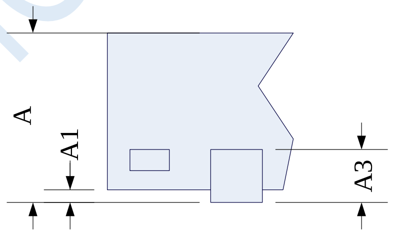
General size			
			mm
SYMBOL	MIN	TYP	MAX
A	-	-	1.20
A1	0.05	-	0.15
A2	0.90	1.00	1.05
A3	0.34	0.44	0.54
b	0.20	0.24	0.28
D	6.40	6.50	6.60
E	6.20	6.50	6.60
E1	4.30	4.40	4.50
e	0.65BSC		
L	0.45	0.60	0.75
L1	1.00REF		
L2	0.25BSC		
R1	0.09	-	-
R2	0.09	-	-



4.5 QFN20 Package mechanical data（3mm*3mm）

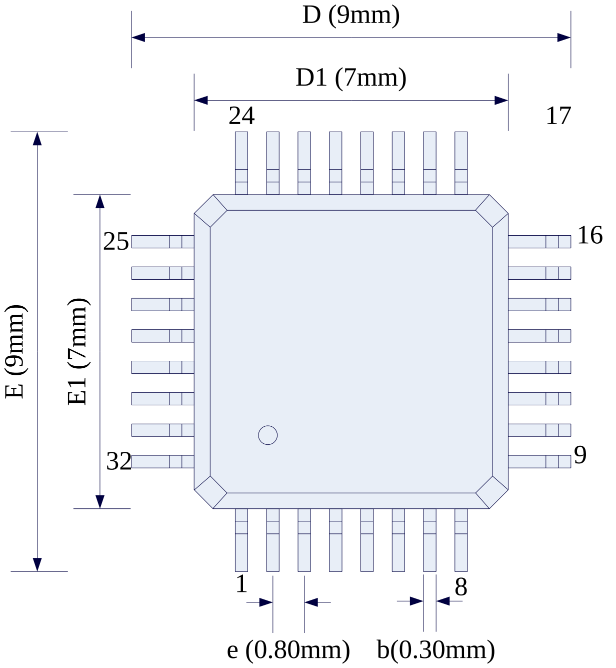


General size			
			mm
SYMBOL	MIN	TYP	MAX
A	0.70	0.75	0.80
A1	0	0.02	0.05
A2	0.50	0.55	0.60
A3	-	0.20REF	-
b	0.15	0.20	0.25
D	2.90	3.00	3.10
E	2.90	3.00	3.10
D2	1.40	1.50	1.60
E2	1.40	1.50	1.60
e	0.30	0.40	0.50
L	0.35	0.40	0.45
K	0.35REF		
R	0.085	-	-
C1	-	0.07	-
C2	-	0.07	-

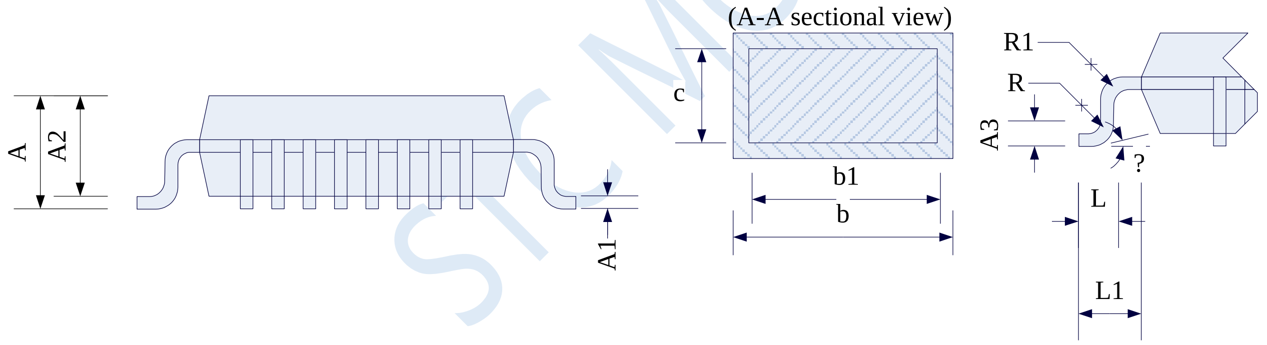


The back metal sheet (substrate) of STC's existing DFN8 packaged chip is not grounded inside the chip. It can be grounded or ungrounded on the user's PCB board, which will not affect the performance of the chip.

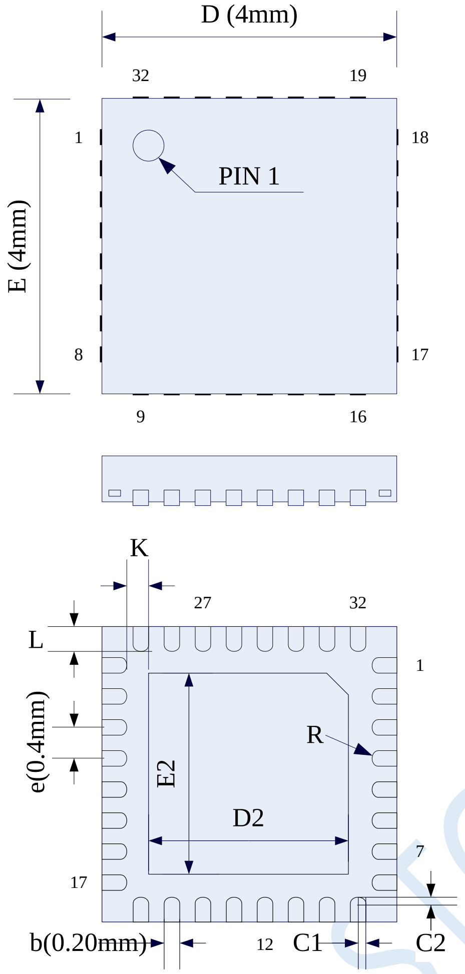
4.6 LQFP32 Package mechanical data（9mm*9mm）



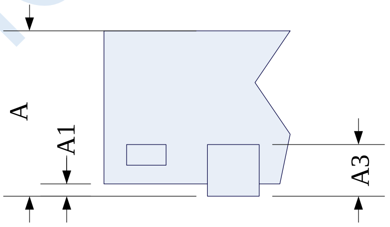
General size			
			mm
SYMBOL	MIN	TYP	MAX
A	1.45	1.55	1.65
A1	0.01	-	0.21
A2	1.35	1.40	1.45
A3	-	0.254	-
b1	0.30	0.35	0.40
b	0.31	0.37	0.43
c	-	0.127	-
D	8.80	9.00	9.20
D1	6.90	7.00	7.10
E	8.80	9.00	9.20
E1	6.90	7.00	7.10
e	0.70	0.80	0.90
L	0.43	-	0.71
L	1.00REF		
L1	0.25BSC		
R	0.1	-	0.25
R1	0.1	-	-
?	0 °	-	10 °



4.7 QFN32 Package mechanical data（4mm*4mm）

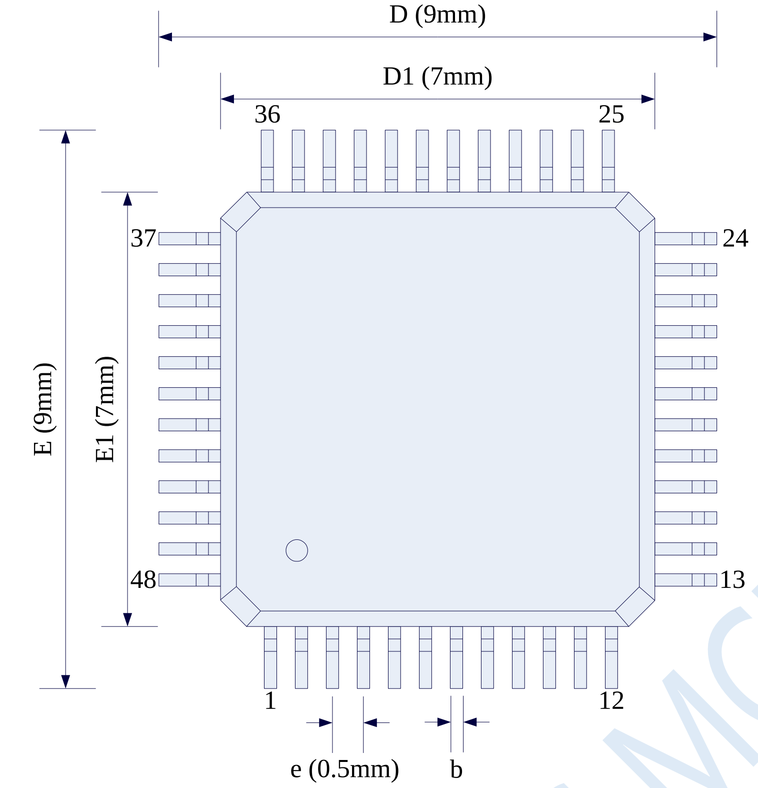


General size			
			mm
SYMBOL	MIN	TYP	MAX
A	0.70	0.75	0.80
A1	0	0.02	0.05
A2	0.50	0.55	0.60
A3	-	0.20REF	-
b	0.15	0.20	0.25
D	3.90	4.00	4.10
E	3.90	4.00	4.10
D2	2.60	2.70	2.80
E2	2.60	2.70	2.80
e	0.30	0.40	0.50
L	0.35	0.40	0.45
K	0.25REF		
R	0.09	-	-
C1	-	0.16	-
C2	-	0.16	-

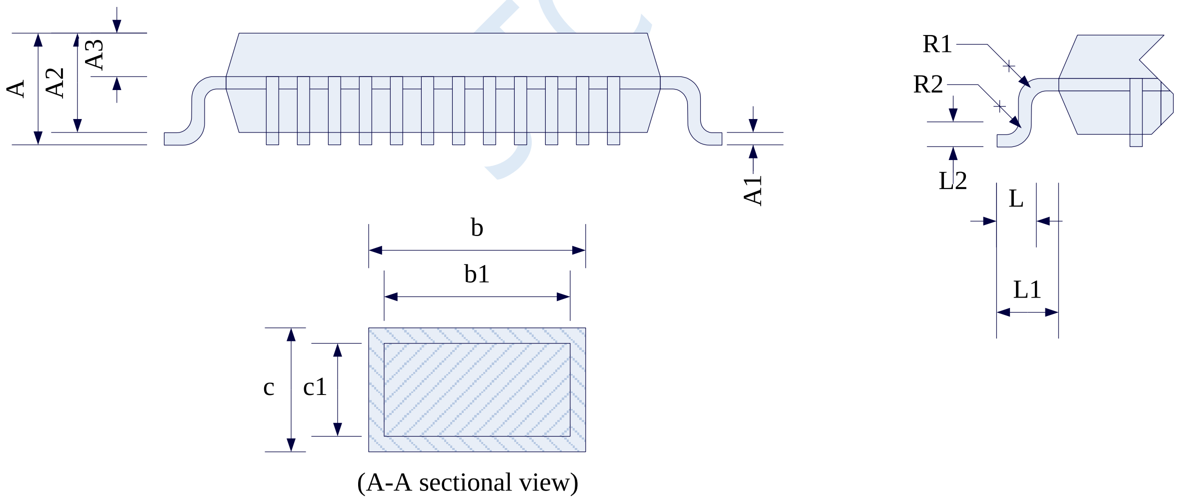


The back metal sheet (substrate) of STC's existing DFN8 packaged chip is not grounded inside the chip. It can be grounded or ungrounded on the user's PCB board, which will not affect the performance of the chip.

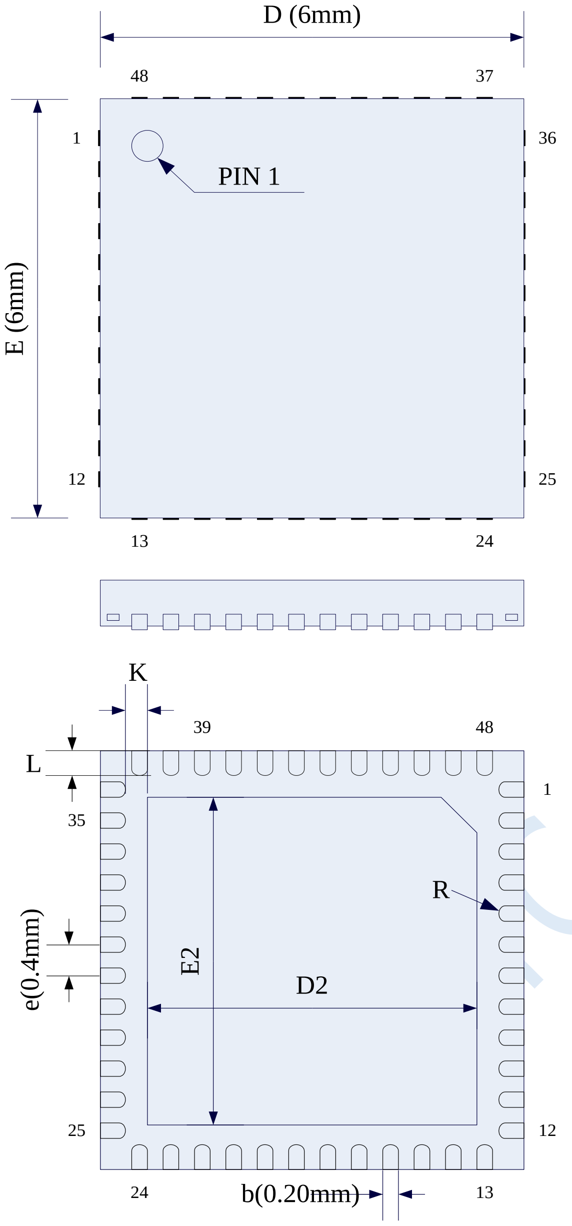
4.8 LQFP48 Package mechanical data（9mm*9mm）



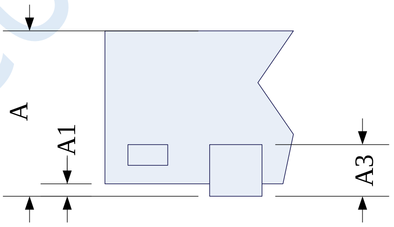
General size			
			mm
SYMBOL	MIN	TYP	MAX
A	-	-	1.60
A1	0.05	-	0.15
A2	1.35	1.40	1.45
A3	0.59	0.64	0.69
b	0.18	-	0.27
b1	0.17	0.20	0.23
c	0.13	-	0.18
c1	0.12	0.127	0.134
D	8.80	9.00	9.20
D1	6.90	7.00	7.10
E	8.80	9.00	9.20
E1	6.90	7.00	7.10
e	0.45	0.50	0.55
L	0.45	0.60	0.75
L1	1.00REF		
L2	0.25BSC		
R1	0.08	-	-
R2	0.08	-	0.20
S	0.20	-	-



4.9 QFN48 Package mechanical data（6mm*6mm）

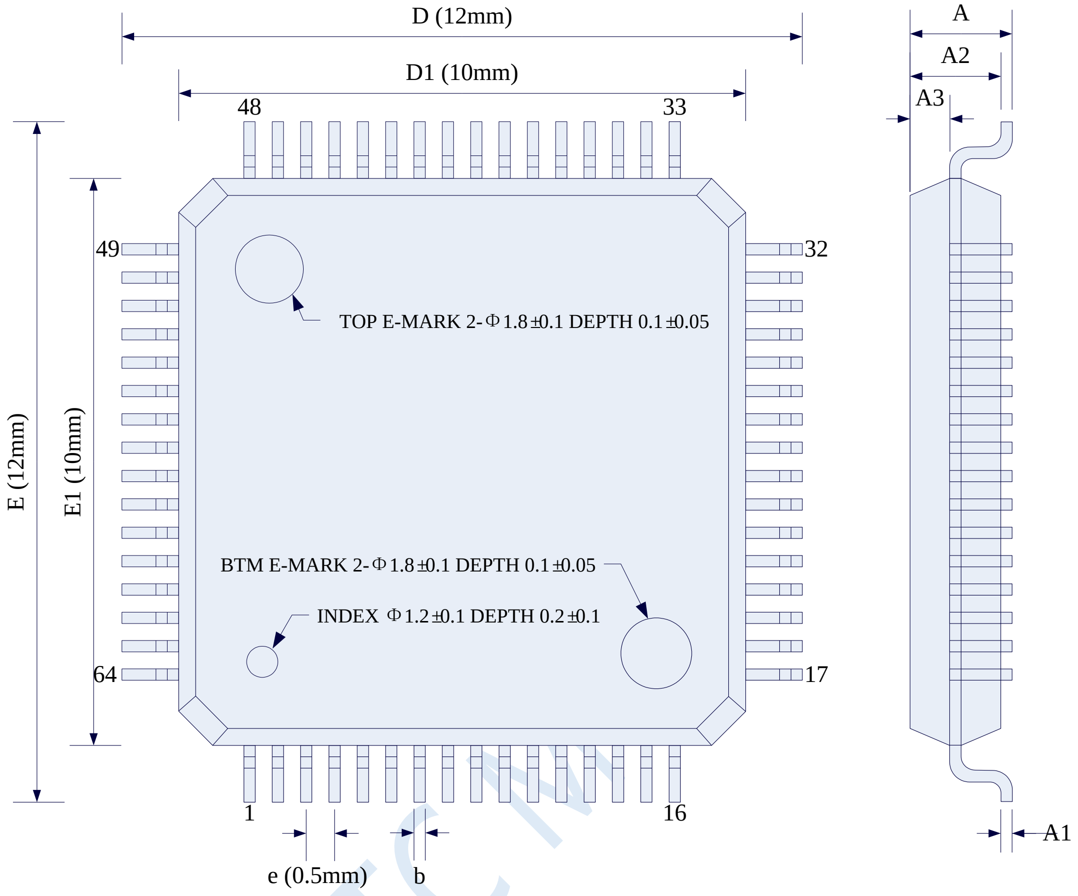


General size			
			mm
SYMBOL	MIN	TYP	MAX
A	0.70	0.75	0.80
A1	0.00	0.02	0.05
A3		0.20REF	
b	0.15	0.20	0.25
D	5.90	6.00	6.10
E	5.90	6.00	6.10
D2	3.95	4.05	4.15
E2	3.95	4.05	4.15
e	0.35	0.40	0.45
L	0.35	0.40	0.45
K		0.20	
R	0.09	-	-

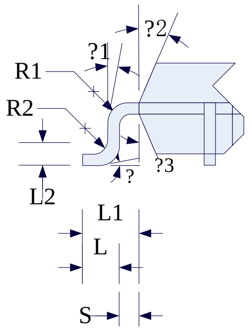


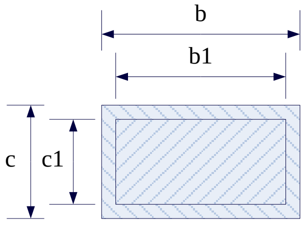
The back metal sheet (substrate) of STC's existing DFN8 packaged chip is not grounded inside the chip. It can be grounded or ungrounded on the user's PCB board, which will not affect the performance of the chip.

4.10 LQFP64S Package mechanical data（12mm*12mm）


D (12mm)
D1 (10mm)
48
33
49
32
64
17
1
16
E (12mm)
E1 (10mm)
TOP E-MARK 2-Φ1.8±0.1 DEPTH 0.1±0.05
BTM E-MARK 2-Φ1.8±0.1 DEPTH 0.1±0.05
INDEX Φ1.2±0.1 DEPTH 0.2±0.1
e (0.5mm)
b
A
A2
A3
A1

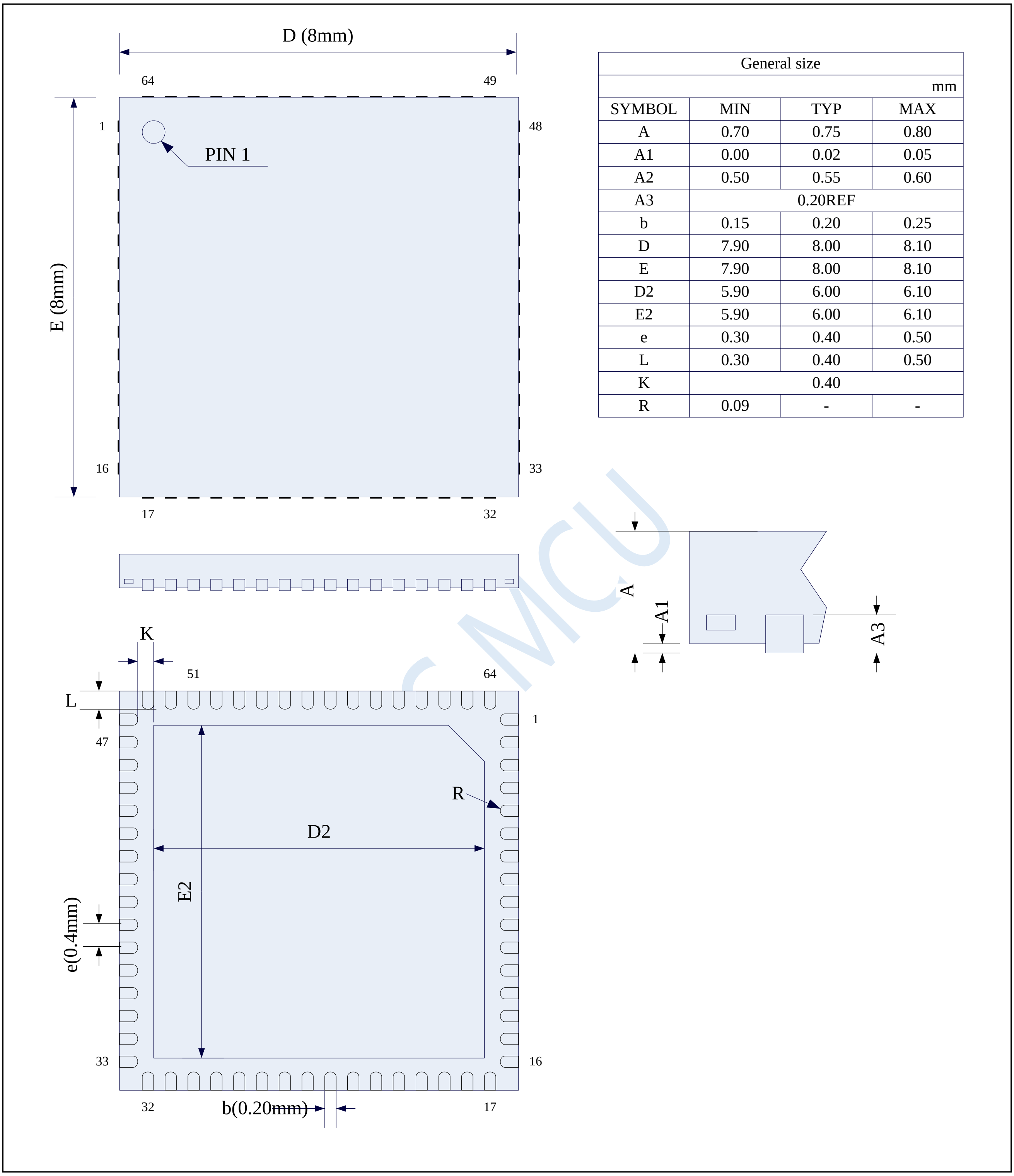
General size			
			mm
SYMBOL	MIN	TYP	MAX
A	-	-	1.60
A1	0.05	-	0.15
A2	1.35	1.40	1.45
A3	0.59	0.64	0.69
b	0.18	-	0.27
b1	0.17	0.20	0.23
c	0.13	-	0.18
c1	0.12	0.127	0.134
D	11.80	12.00	12.20
D1	9.90	10.00	10.10
E	11.80	12.00	12.20
E1	9.90	10.00	10.10
e	0.50BSC		
L	0.45	0.60	0.75
L1	1.00REF		
L2	0.25BSC		
R1	0.08	-	-
R2	0.08	-	0.20
S	0.20	-	-
?	0°	3.5°	7°
?1	0°	-	-
?2	11°	12°	13°
?3	11°	12°	13°





(A-A sectional view)

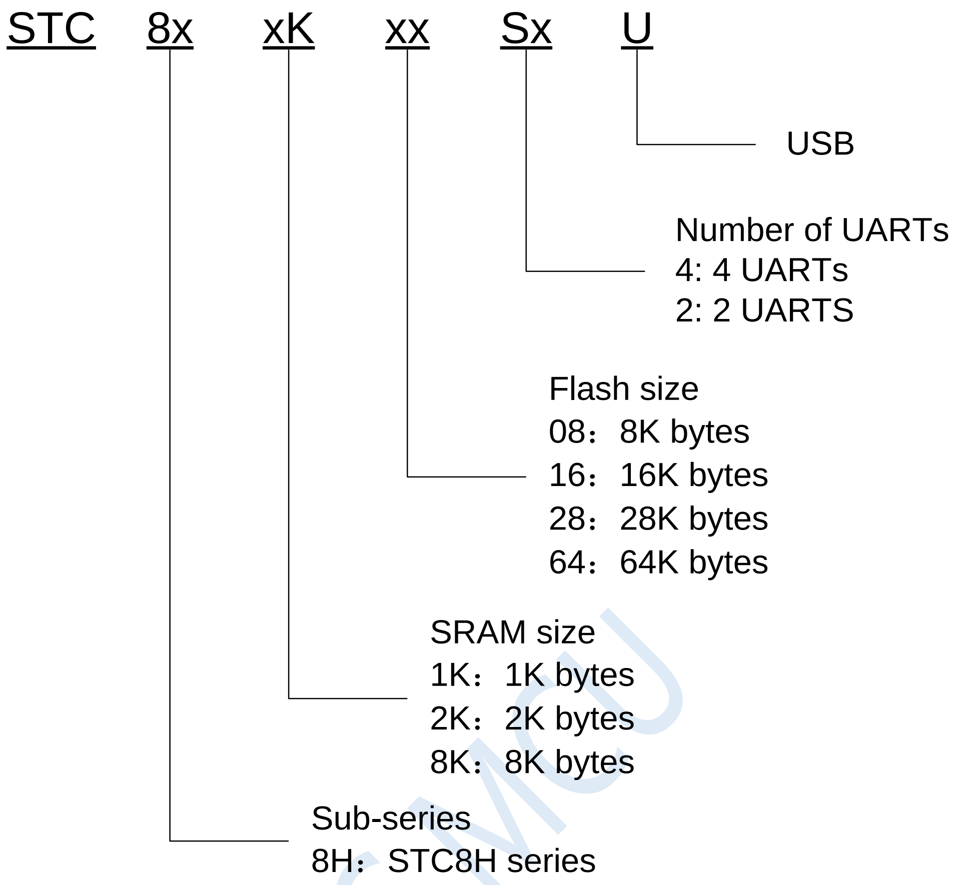
4.11 QFN64 Package mechanical data（8mm*8mm）



General size			
			mm
SYMBOL	MIN	TYP	MAX
A	0.70	0.75	0.80
A1	0.00	0.02	0.05
A2	0.50	0.55	0.60
A3		0.20REF	
b	0.15	0.20	0.25
D	7.90	8.00	8.10
E	7.90	8.00	8.10
D2	5.90	6.00	6.10
E2	5.90	6.00	6.10
e	0.30	0.40	0.50
L	0.30	0.40	0.50
K		0.40	
R	0.09	-	-

The back metal sheet (substrate) of STC's existing DFN8 packaged chip is not grounded inside the chip. It can be grounded or ungrounded on the user's PCB board, which will not affect the performance of the chip.

4.12 Naming rules of STC8 family



STC 8x xK xx Sx U

USB

Number of UARTs
4: 4 UARTs
2: 2 UARTS

Flash size
08：8K bytes
16：16K bytes
28：28K bytes
64：64K bytes

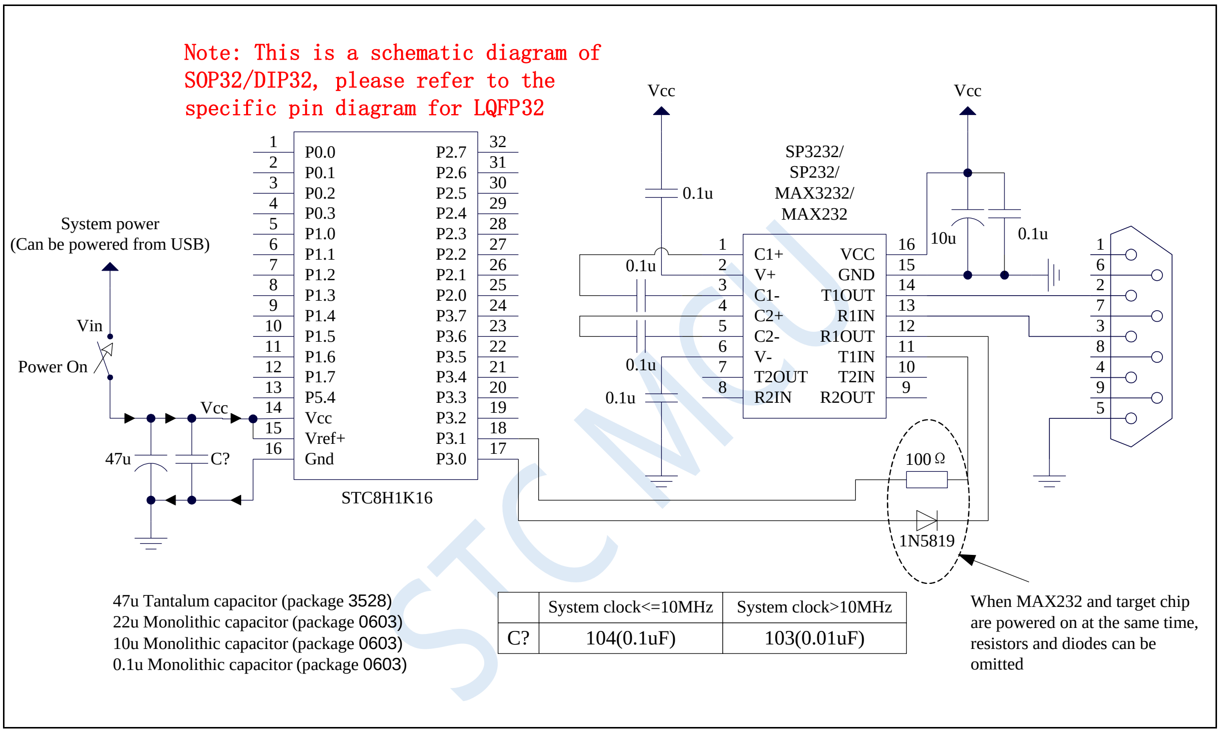
SRAM size
1K：1K bytes
2K：2K bytes
8K：8K bytes

Sub-series
8H：STC8H series

5 ISP Download and typical application circuit

5.1 STC8H series ISP download application circuit

5.1.1 Download using RS-232 converter (general precision ADC), Emulation supported



Note: This is a schematic diagram of SOP32/DIP32, please refer to the specific pin diagram for LQFP32

47u Tantalum capacitor (package 3528)
22u Monolithic capacitor (package 0603)
10u Monolithic capacitor (package 0603)
0.1u Monolithic capacitor (package 0603)

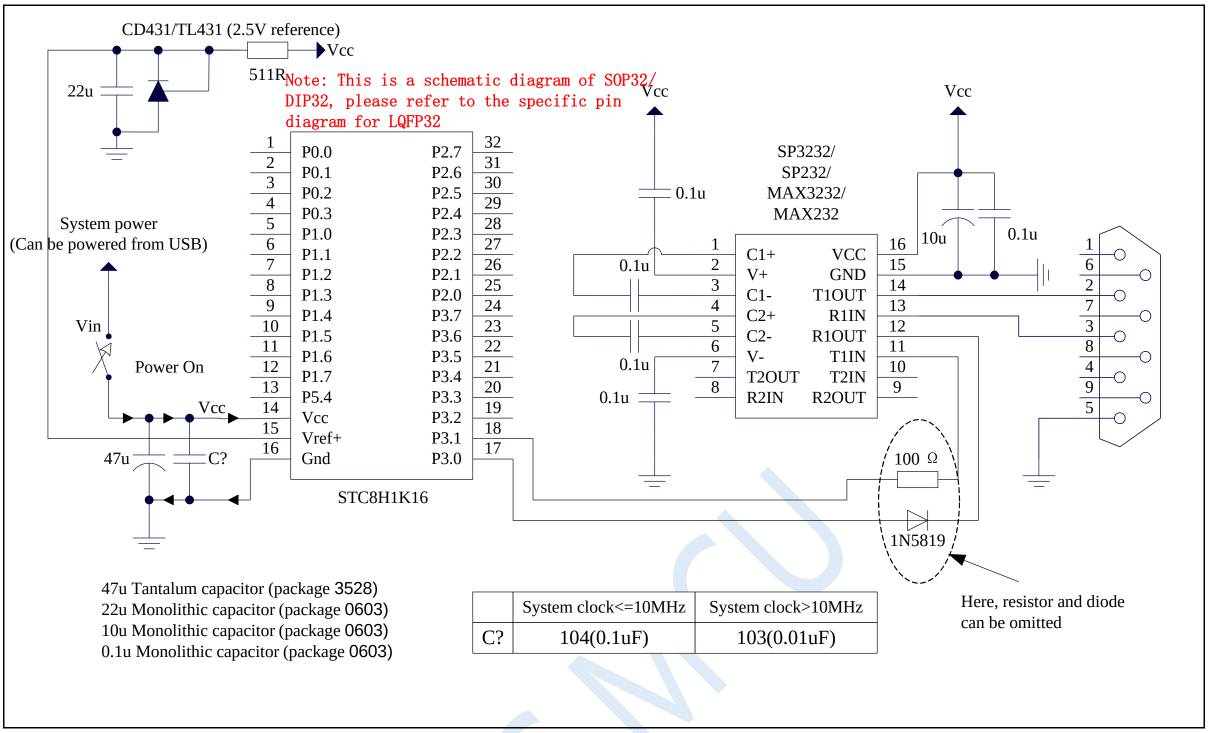
	System clock<=10MHz	System clock>10MHz
C?	104(0.1uF)	103(0.01uF)

When MAX232 and target chip are powered on at the same time, resistors and diodes can be omitted

ISP download steps:

1. Power off the target chip.
2. Click the "Download/Program" button in the STC-ISP download software.
3. Power on the target chip.
4. Start ISP download.

5.1.2 Download using RS-232 converter (High precision ADC), Emulation supported



Note: This is a schematic diagram of SOP32/DIP32, please refer to the specific pin diagram for LQFP32

47u Tantalum capacitor (package 3528)
22u Monolithic capacitor (package 0603)
10u Monolithic capacitor (package 0603)
0.1u Monolithic capacitor (package 0603)

	System clock<=10MHz	System clock>10MHz
C?	104(0.1uF)	103(0.01uF)

ISP download steps:

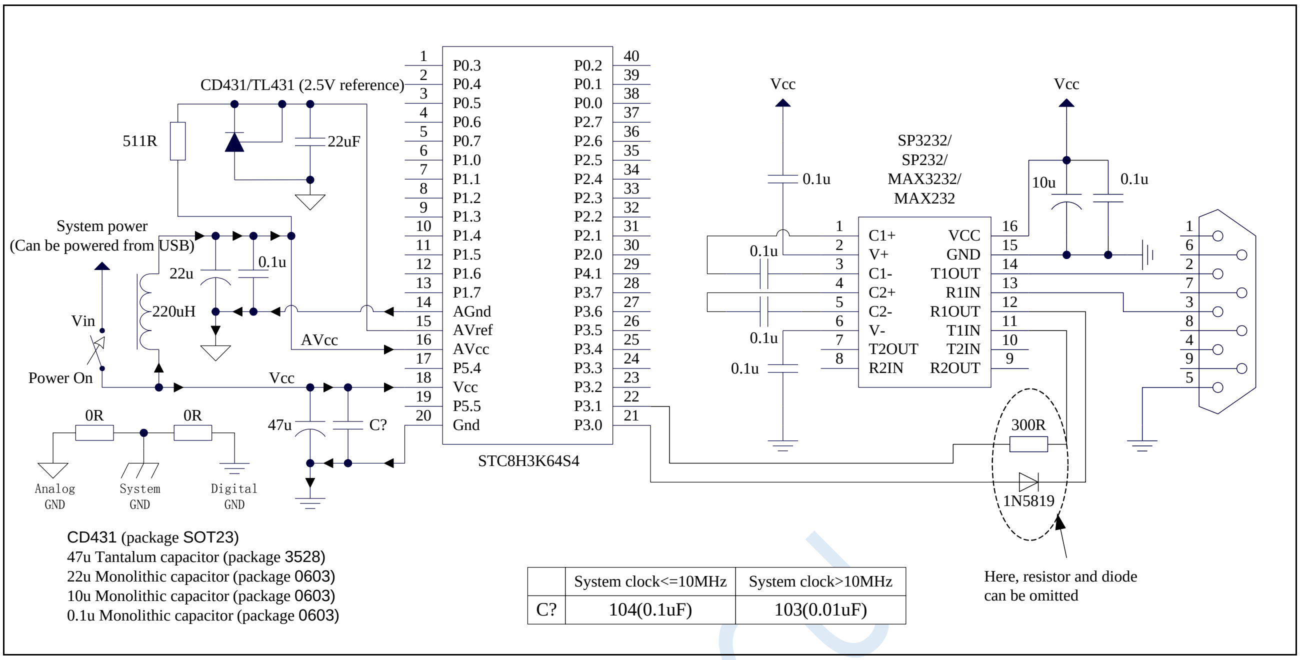
1. Power off the target chip.

2. Click the “Download/Program” button in the STC-ISP download software.

3. Power on the target chip.

4. Start ISP download.

5.1.3 High-precision ADC reference circuit diagram of STC8H3K64S4 series, Emulation supported



CD431 (package SOT23)
47u Tantalum capacitor (package 3528)
22u Monolithic capacitor (package 0603)
10u Monolithic capacitor (package 0603)
0.1u Monolithic capacitor (package 0603)

	System clock<=10MHz	System clock>10MHz
C?	104(0.1uF)	103(0.01uF)

Here, resistor and diode can be omitted

ISP download steps:

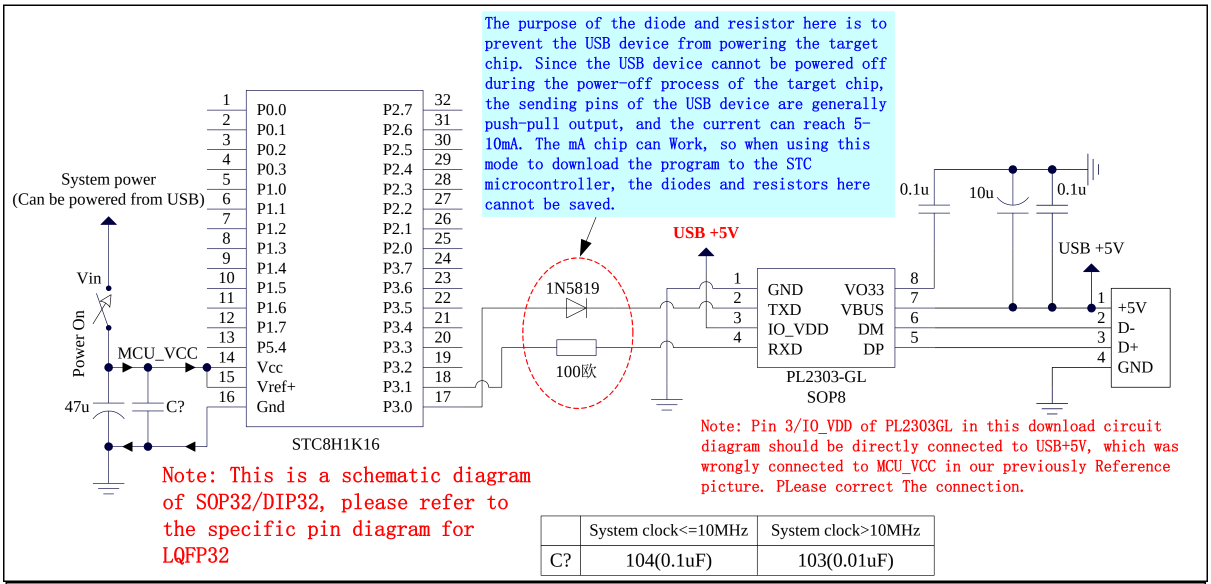
1. Power off the target chip.

2. Click the "Download/Program" button in the STC-ISP download software.

3. Power on the target chip.

4. Start ISP download.

5.1.4 Download using PL2303-GL, Emulation supported



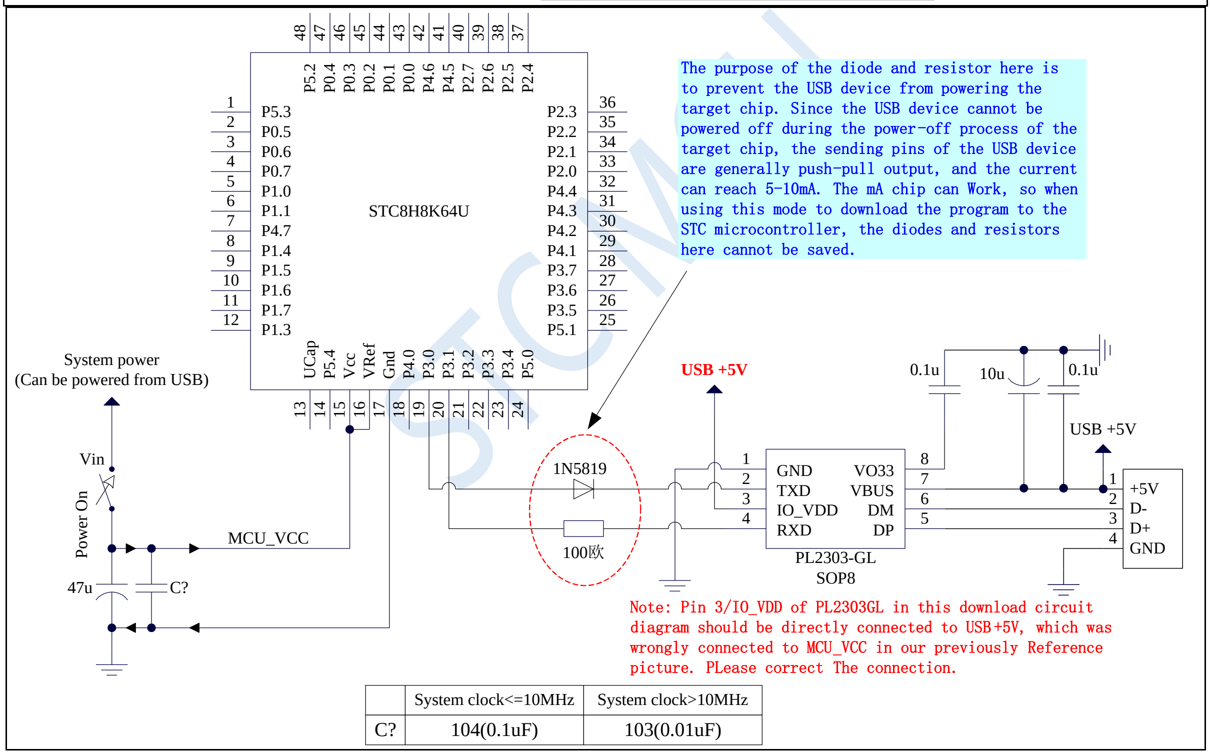
STC8H1K16
P0.0 P0.1 P0.2 P0.3 P1.0 P1.1 P1.2 P1.3 P1.4 P1.5 P1.6 P1.7 P5.4 Vcc Vref+ Gnd
P2.7 P2.6 P2.5 P2.4 P2.3 P2.2 P2.1 P2.0 P3.7 P3.6 P3.5 P3.4 P3.3 P3.2 P3.1 P3.0
System power
(Can be powered from USB)
Vin
Power On
MCU_VCC
47u
C?
1N5819
100欧
USB +5V
GND VO33
TXD VBUS
IO_VDD DM
RXD DP
PL2303-GL
SOP8
0.1u
10u
0.1u
USB +5V
+5V
D-
D+
GND

The purpose of the diode and resistor here is to prevent the USB device from powering the target chip. Since the USB device cannot be powered off during the power-off process of the target chip, the sending pins of the USB device are generally push-pull output, and the current can reach 5-10mA. The mA chip can Work, so when using this mode to download the program to the STC microcontroller, the diodes and resistors here cannot be saved.

Note: Pin 3/IO_VDD of PL2303GL in this download circuit diagram should be directly connected to USB+5V, which was wrongly connected to MCU_VCC in our previously Reference picture. PLease correct The connection.

Note: This is a schematic diagram of SOP32/DIP32, please refer to the specific pin diagram for LQFP32

	System clock<=10MHz	System clock>10MHz
C?	104(0.1uF)	103(0.01uF)



STC8H8K64U
P5.3 P0.5 P0.6 P0.7 P1.0 P1.1 P4.7 P1.4 P1.5 P1.6 P1.7 P1.3
P2.3 P2.2 P2.1 P2.0 P4.4 P4.3 P4.2 P4.1 P3.7 P3.6 P3.5 P5.1
P5.2 P0.4 P0.3 P0.2 P0.1 P0.0 P4.6 P4.5 P2.7 P2.6 P2.5 P2.4
UCap P5.4 Vcc VRef Gnd P4.0 P3.0 P3.1 P3.2 P3.3 P3.4 P5.0
System power
(Can be powered from USB)
Vin
Power On
MCU_VCC
47u
C?
1N5819
100欧
USB +5V
GND VO33
TXD VBUS
IO_VDD DM
RXD DP
PL2303-GL
SOP8
0.1u
10u
0.1u
USB +5V
+5V
D-
D+
GND

The purpose of the diode and resistor here is to prevent the USB device from powering the target chip. Since the USB device cannot be powered off during the power-off process of the target chip, the sending pins of the USB device are generally push-pull output, and the current can reach 5-10mA. The mA chip can Work, so when using this mode to download the program to the STC microcontroller, the diodes and resistors here cannot be saved.

Note: Pin 3/IO_VDD of PL2303GL in this download circuit diagram should be directly connected to USB+5V, which was wrongly connected to MCU_VCC in our previously Reference picture. PLease correct The connection.

	System clock<=10MHz	System clock>10MHz
C?	104(0.1uF)	103(0.01uF)

ISP download steps:

1. Power off the target chip, be careful not to power off the USB-to-serial chip (such as: CH340, PL2303-GL, etc.)

Note: Some baud rate errors of PL2303-SA are very large, it is recommended to use PL2303-GL

2. Since the sending pins of the USB-to-serial chip are generally strong push-pull outputs, a diode must be connected in series between the P3.0 port of the target chip and the sending pin of the USB-to-serial chip, otherwise the target chip cannot be completely powered off. The goal of powering down the target chip is not reached.

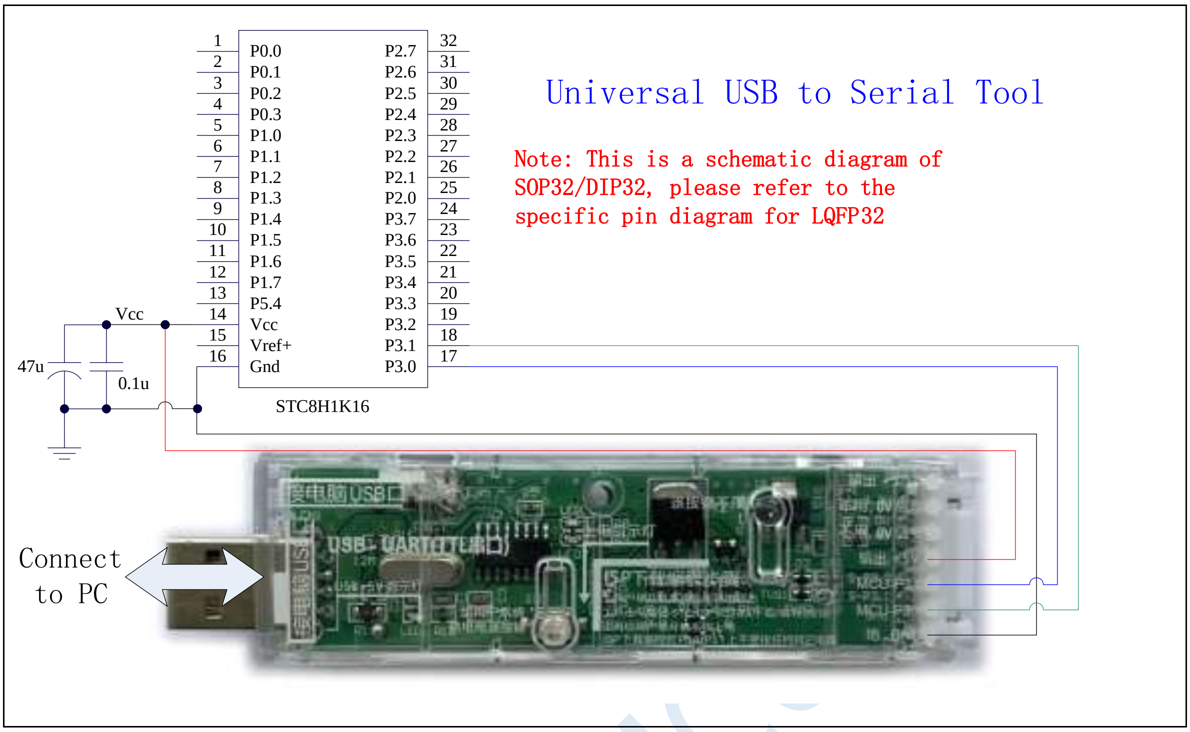
3. Click the "Download/Program" button in the STC-ISP download software

4. Power on the target chip

5. Start ISP download

Note: At present, it has been found that when using the USB cable for ISP download, the USB cable is too thin and the voltage drop on the USB cable is too large, resulting in insufficient power supply during the ISP download. Therefore, please make sure to use the booster USB cable for ISP download.

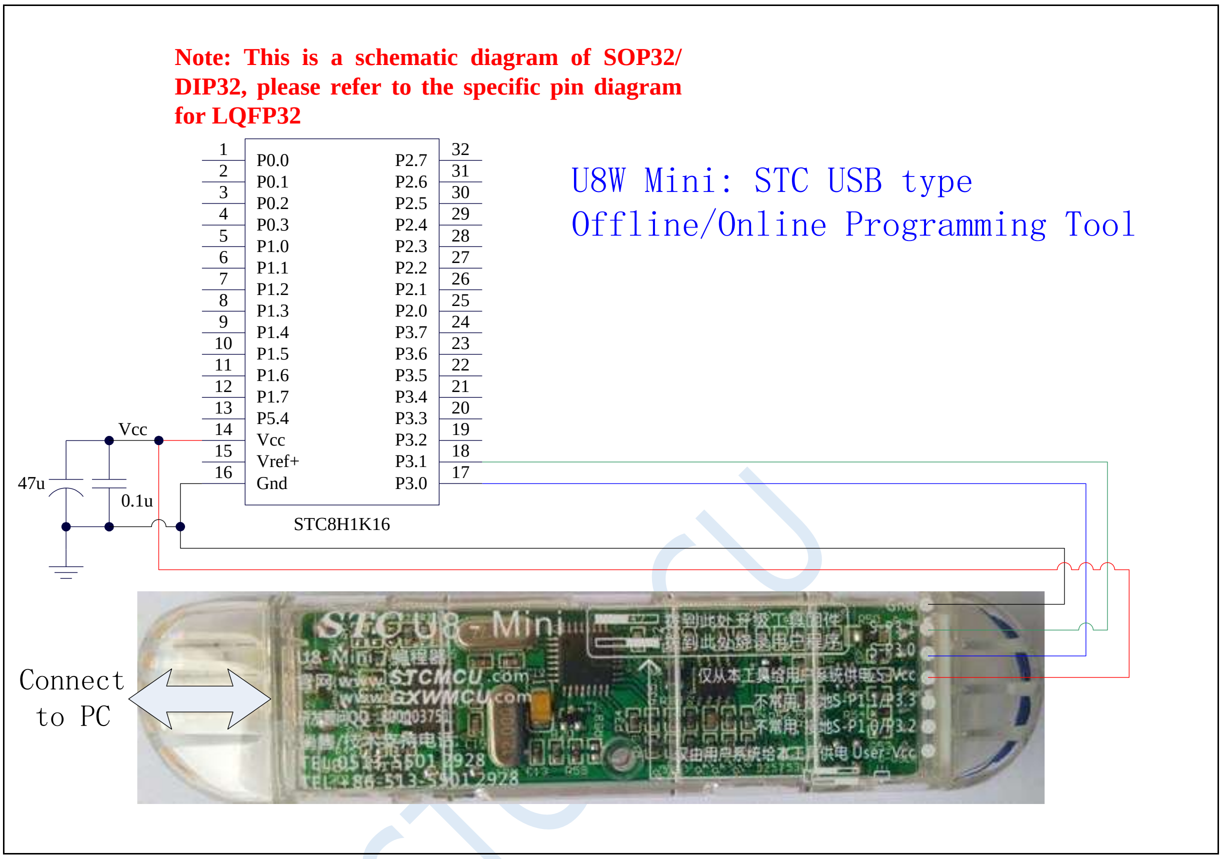
5.1.5 Download using universal USB to Serial Tool, Emulation supported



ISP download steps:
1. Connect the universal USB-to-serial tool to the target chip according to the connection method shown in the figure.
2. Press the power button to confirm that the target chip is in a power-off state (the power-on indicator is off). Note: When the tool is powered on for the first time, it will not supply power to the outside world, so if you use the tool for the first time, you can skip this step.
3. Click the "Download/Program" button in the STC-ISP download software
4. Press the power button again to power on the target chip (the power-on indicator is on)
5. Start ISP download.

Note: At present, it has been found that when using the USB cable for ISP download, the USB cable is too thin and the voltage drop on the USB cable is too large, resulting in insufficient power supply during the ISP download. Therefore, please make sure to use the booster USB cable for ISP download.

5.1.6 Download using U8-Mini tool, Support ISP online and offline download, also support emulation

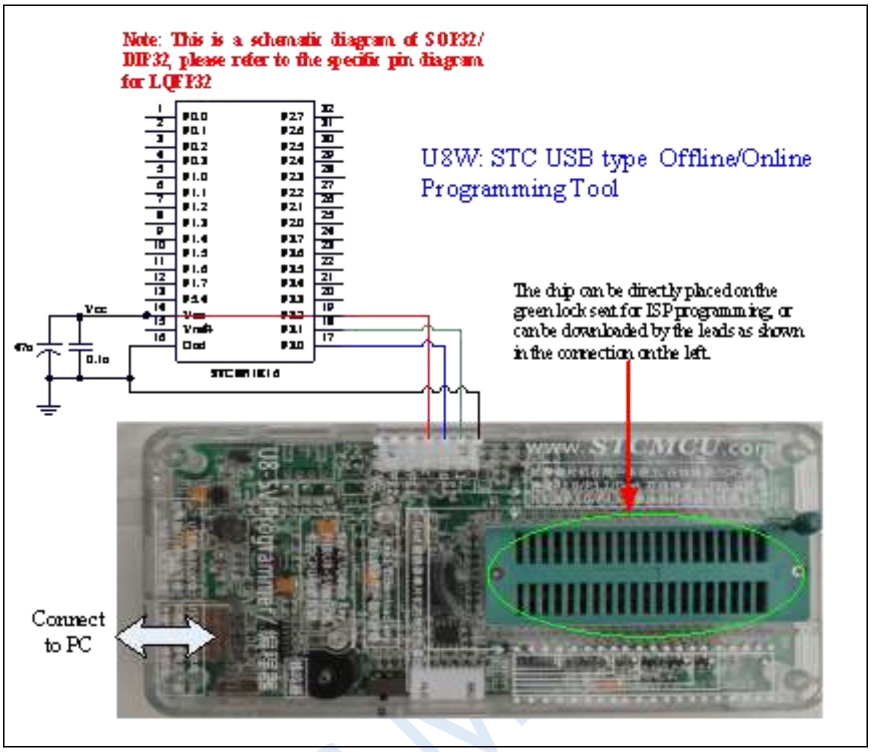


ISP download steps:
1. Connect the U8-Mini and the target chip according to the connection method shown in the figure.
2. Click the "Download/Program" button in the STC-ISP download software
3. Start ISP download
Note: If the U8-Mini is used to power the target system, the total current of the target system cannot be greater than 200mA, otherwise the download will fail.

Note: At present, it has been found that when using the USB cable for ISP download, the USB cable is too thin and the voltage drop on the USB cable is too large, resulting in insufficient power supply during the ISP download. Therefore, please make sure to use the booster USB cable for ISP download.

To emulate using the U8-Mini, you must set the U8-Mini to pass-through mode firstly. The method for U8W/U8W-Mini to realize the USB-to-serial pass-through mode is as follows:
1. Firstly, the U8W/U8W-Mini firmware must be upgraded to v1.37 and above.
2. After the U8W/U8W-Mini is powered on, it is in the normal download mode. At this time, press and hold the Key1 (download) button on the tool and do not release it, then press the Key2 (power) button again, and then release the Key2 (power) button. Release the Key1 (download) button again, and the U8W/U8W-Mini will enter the USB-to-serial pass-through mode. (Press Key1→Press Key2→Release Key2→Release Key1).
3. The U8W/U8W-Mini tool that enters the pass-through mode is just a simple USB to serial port and does not have the offline download function. If you need to restore the original function of the U8W/U8W-Mini, you only need to press the Key2 (power) button again.

5.1.7 Download using U8W tool, Support ISP online and offline download, also support emulation


Note: This is a schematic diagram of SOP32/ DIP32, please refer to the specific pin diagram for LQFP32
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
P0.0
P0.1
P0.2
P0.3
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P5.4
Vcc
Vref+
Gnd
P2.7
P2.6
P2.5
P2.4
P2.3
P2.2
P2.1
P2.0
P3.7
P3.6
P3.5
P3.4
P3.3
P3.2
P3.1
P3.0
32
31
30
29
28
27
26
25
24
23
22
21
20
19
18
17
STC8H1K16
Vcc
47u
0.1u
U8W: STC USB type Offline/Online Programming Tool
The chip can be directly placed on the green lock seat for ISP programming, or can be downloaded by the leads as shown in the connection on the left.
Connect to PC
www.STCMCU.com

ISP download steps(Connection method):

1. Connect the U8-Mini and the target chip according to the connection method shown in the figure.
2. Click the "Download/Program" button in the STC-ISP download software
3. Start ISP download

Note: If the U8-Mini is used to power the target system, the total current of the target system cannot be greater than 200mA, otherwise the download will fail.

ISP download steps (on-board mode):

1. Place the chip in the direction that pin 1 is close to the locking wrench and the pins are aligned downwards.
2. Click the "Download/Program" button in the STC-ISP download software
3. Start ISP download.

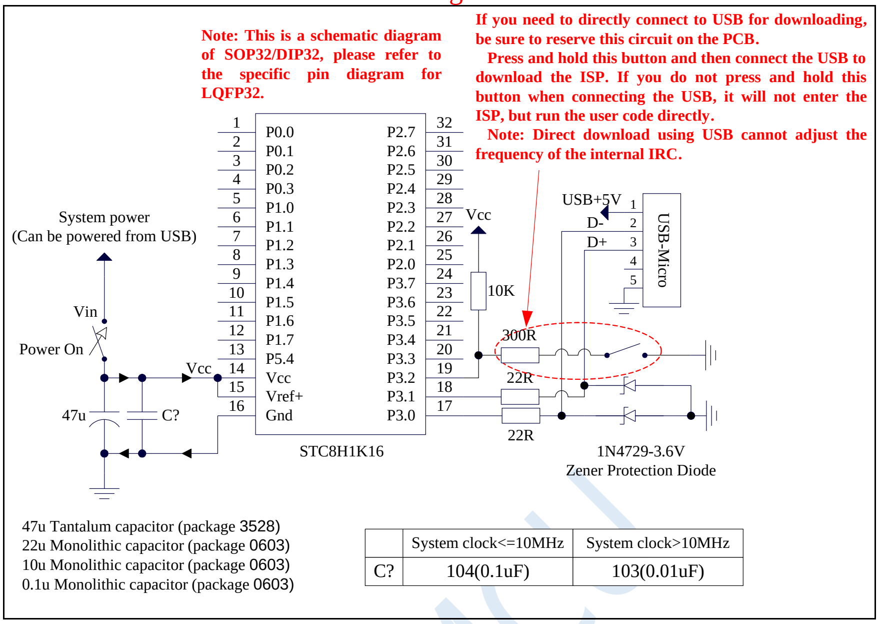
Note: At present, it has been found that when using the USB cable for ISP download, the USB cable is too thin and the voltage drop on the USB cable is too large, resulting in insufficient power supply during the ISP download. Therefore, please make sure to use the booster USB cable for ISP download.

To emulate using the U8-Mini, you must set the U8-Mini to pass-through mode firstly. The method for U8W/U8W-Mini to realize the USB-to-serial pass-through mode is as follows:

1. Firstly, the U8W/U8W-Mini firmware must be upgraded to v1.37 and above.
2. After the U8W/U8W-Mini is powered on, it is in the normal download mode. At this time, press and hold the Key1 (download) button on the tool and do not release it, then press the Key2 (power) button again, and then release the Key2 (power) button. Release the Key1 (download) button again, and the U8W/U8W-Mini will enter the USB-to-serial pass-through mode. (Press Key1→Press Key2→Release Key2→Release Key1).
3. The U8W/U8W-Mini tool that enters the pass-through mode is just a simple USB to serial port and does not have the offline download function. If you need to restore the original function of the U8W/U8W-Mini, you only need to press the Key2 (power) button again.

5.1.8 Software simulation USB ISP download directly, which is not recommended, and does not support simulation

Note 1: When using USB to download, you need to connect P3.2 to Gnd for normal download.
Note 2: If USB download is not required, P3.0/P3.1/P3.2 cannot be low at the same time when the chip is reset, otherwise the chip will always be in USB download mode without running user code.



Note: This is a schematic diagram of SOP32/DIP32, please refer to the specific pin diagram for LQFP32.

If you need to directly connect to USB for downloading, be sure to reserve this circuit on the PCB.
Press and hold this button and then connect the USB to download the ISP. If you do not press and hold this button when connecting the USB, it will not enter the ISP, but run the user code directly.
Note: Direct download using USB cannot adjust the frequency of the internal IRC.

47u Tantalum capacitor (package 3528)
22u Monolithic capacitor (package 0603)
10u Monolithic capacitor (package 0603)
0.1u Monolithic capacitor (package 0603)

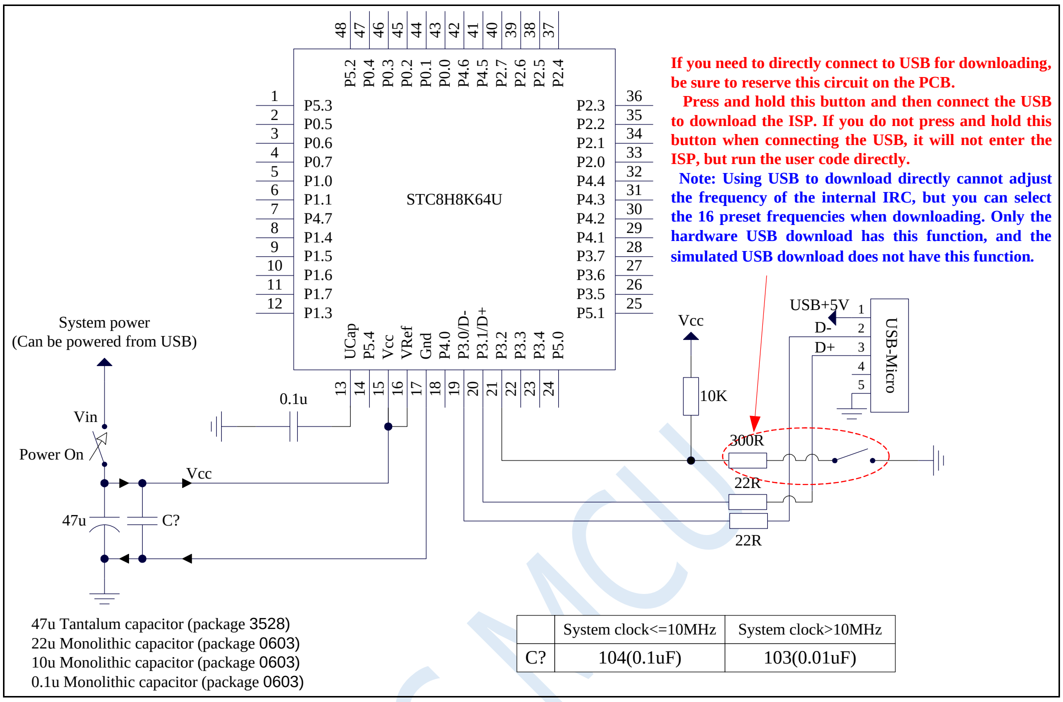
	System clock<=10MHz	System clock>10MHz
C?	104(0.1uF)	103(0.01uF)

ISP download steps:
1. Power off the target chip.
2. Connect P3.0/P3.1 to the USB port according to the connection method shown in the figure.
3. Short-circuit P3.2 with GND.
4. Power on the target chip and wait for "STC USB Writer (USB1)" to be automatically recognized in the STC-ISP download software.
5. Click the "Download/Program" button in the download software (note: the sequence of operations is different from serial download).
6. Start the ISP download.

Note: At present, it has been found that when using the USB cable for ISP download, the USB cable is too thin and the voltage drop on the USB cable is too large, resulting in insufficient power supply during the ISP download. Therefore, please make sure to use the booster USB cable for ISP download.

5.1.9 ISP Download using hardware USB directly

Note 1: STC8H8K64U-A version chip only supports ISP download in USB mode, STC8H8K64U-B version chip can support ISP download and USB emulation in USB mode.
Note 2: When using USB to download, you need to connect P3.2 to Gnd for normal download.
Note 3: If USB download is not required, P3.0/P3.1/P3.2 cannot be low at the same time when the chip is reset, otherwise the chip will always be in the USB download mode without running user code.



If you need to directly connect to USB for downloading, be sure to reserve this circuit on the PCB.
Press and hold this button and then connect the USB to download the ISP. If you do not press and hold this button when connecting the USB, it will not enter the ISP, but run the user code directly.
Note: Using USB to download directly cannot adjust the frequency of the internal IRC, but you can select the 16 preset frequencies when downloading. Only the hardware USB download has this function, and the simulated USB download does not have this function.

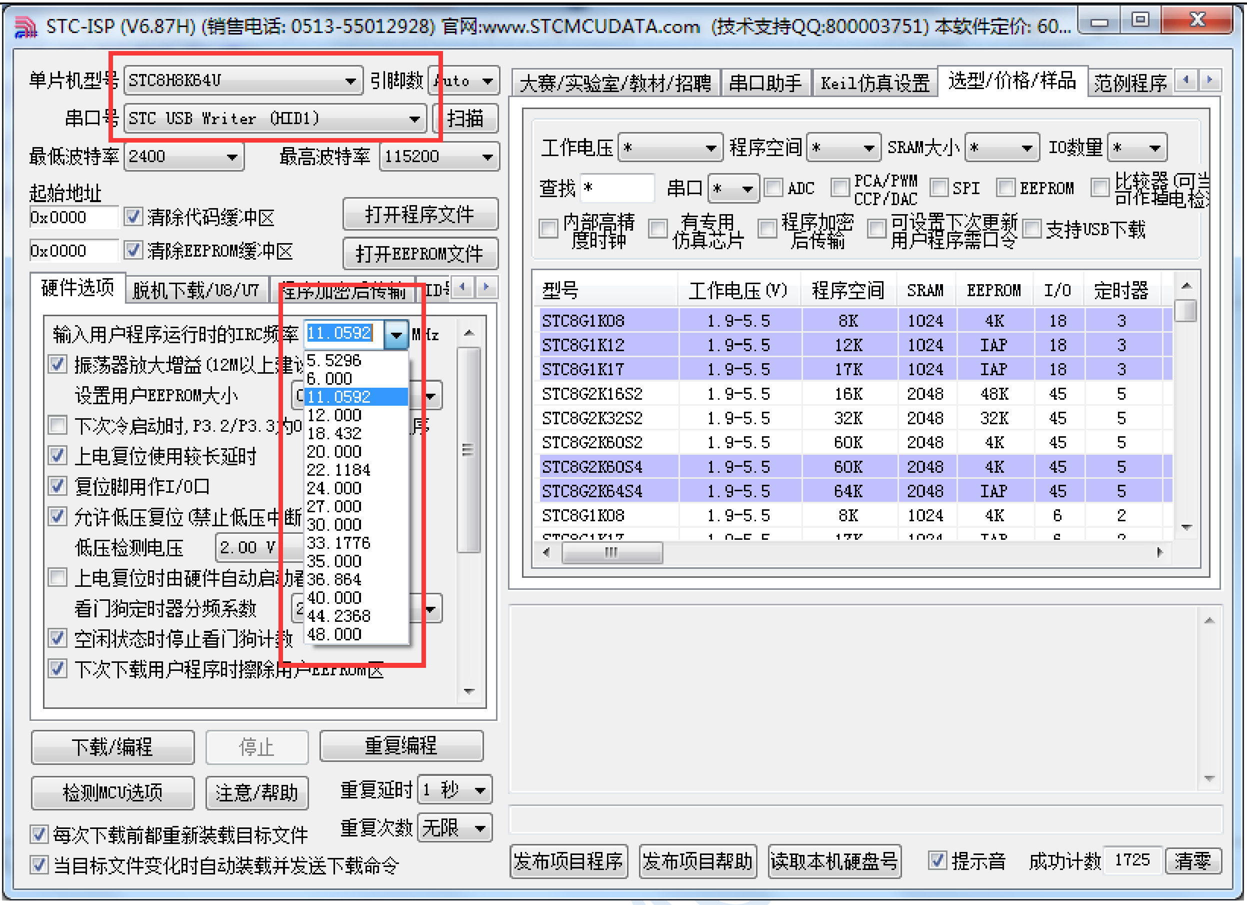
47u Tantalum capacitor (package 3528)
22u Monolithic capacitor (package 0603)
10u Monolithic capacitor (package 0603)
0.1u Monolithic capacitor (package 0603)

	System clock<=10MHz	System clock>10MHz
C?	104(0.1uF)	103(0.01uF)

ISP download steps:
1. Power off the target chip.
2. Connect P3.0/P3.1 to the USB port according to the connection method shown in the figure.
3. Short-circuit P3.2 with GND.
4. Power on the target chip and wait for "STC USB Writer (HID1)" to be automatically recognized in the STC-ISP download software.
5. Click the "Download/Program" button in the download software (note: the sequence of operations is different from serial download).
6. Start the ISP download.

Note: At present, it has been found that when using the USB cable for ISP download, the USB cable is too thin and the voltage drop on the USB cable is too large, resulting in insufficient power supply during the ISP download. Therefore, please make sure to use the booster USB cable for ISP download.

When the user uses the hardware USB to download the STC8H8K64U series to the ISP, the frequency of the internal IRC cannot be adjusted, but the user can select the 16 preset frequencies (5.5296M, 6M, 11.0592M, 12M, 18.432M, 20M, 22.1184M respectively) , 24M, 27M, 30M, 33.1776M, 35M, 36.864M, 40M, 44.2368M and 48M). When downloading, the user can only select one of the frequencies from the drop-down list, and cannot manually enter other frequencies. (Any frequency between 4M and 48M can be input when using the serial port to download). See the illustration on the next page for details.



STC-ISP (V6.87H) (销售电话: 0513-55012928) 官网:www.STCMCUDATA.com (技术支持QQ:800003751) 本软件定价: 60...

单片机型号 STC8H8K64U　引脚数 Auto

串口号 STC USB Writer (HID1)　扫描

最低波特率 2400　最高波特率 115200

起始地址

0x0000 ☑ 清除代码缓冲区　打开程序文件

0x0000 ☑ 清除EEPROM缓冲区　打开EEPROM文件

输入用户程序运行时的IRC频率 11.0592 MHz

```
5.5296
6.000
11.0592
12.000
18.432
20.000
22.1184
24.000
27.000
30.000
33.1776
35.000
36.864
40.000
44.2368
48.000
```

☑ 振荡器放大增益(12M以上建议
设置用户EEPROM大小
☐ 下次冷启动时,P3.2/P3.3为0
☑ 上电复位使用较长延时
☑ 复位脚用作I/O口
☑ 允许低压复位(禁止低压中断
低压检测电压 2.00 V
☐ 上电复位时由硬件自动启动看
看门狗定时器分频系数
☑ 空闲状态时停止看门狗计数
☑ 下次下载用户程序时擦除用户EEPROM区

工作电压 *　程序空间 *　SRAM大小 *　IO数量 *

查找 *　串口 *　☐ ADC　☐ PCA/PWM CCP/DAC　☐ SPI　☐ EEPROM　☐ 比较器(可当可作掉电检

☐ 内部高精度时钟　☐ 有专用仿真芯片　☐ 程序加密后传输　☐ 可设置下次更新用户程序需口令　☐ 支持USB下载

型号	工作电压(V)	程序空间	SRAM	EEPROM	I/O	定时器
STC8G1K08	1.9-5.5	8K	1024	4K	18	3
STC8G1K12	1.9-5.5	12K	1024	IAP	18	3
STC8G1K17	1.9-5.5	17K	1024	IAP	18	3
STC8G2K16S2	1.9-5.5	16K	2048	48K	45	5
STC8G2K32S2	1.9-5.5	32K	2048	32K	45	5
STC8G2K60S2	1.9-5.5	60K	2048	4K	45	5
STC8G2K60S4	1.9-5.5	60K	2048	4K	45	5
STC8G2K64S4	1.9-5.5	64K	2048	IAP	45	5
STC8G1K08	1.9-5.5	8K	1024	4K	6	2
STC8G1K17	1.9-5.5	17K	1024	IAP	6	2

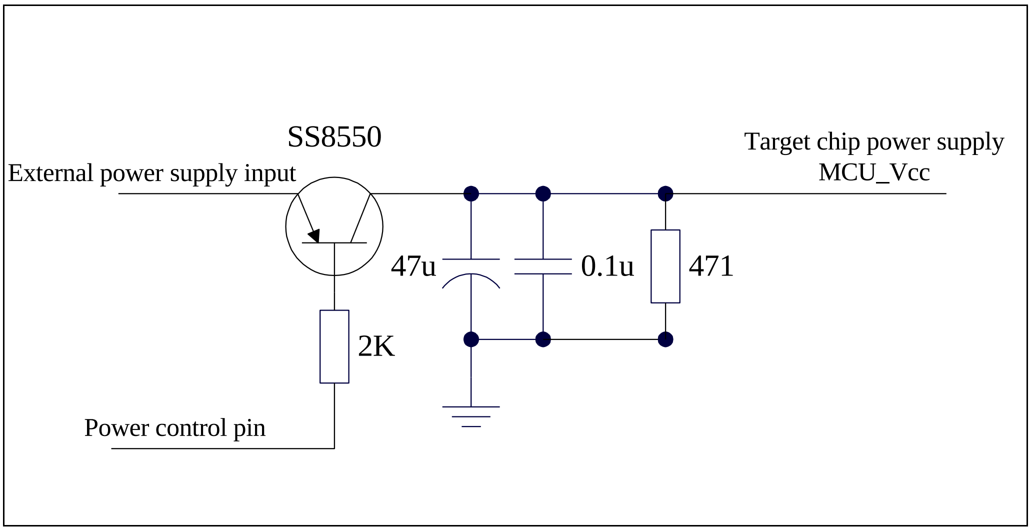
重复延时 1 秒　重复次数 无限

☑ 每次下载前都重新装载目标文件

☑ 当目标文件变化时自动装载并发送下载命令

☑ 提示音　成功计数 1725

5.1.10 Microcontroller Power supply Control Reference Circuit

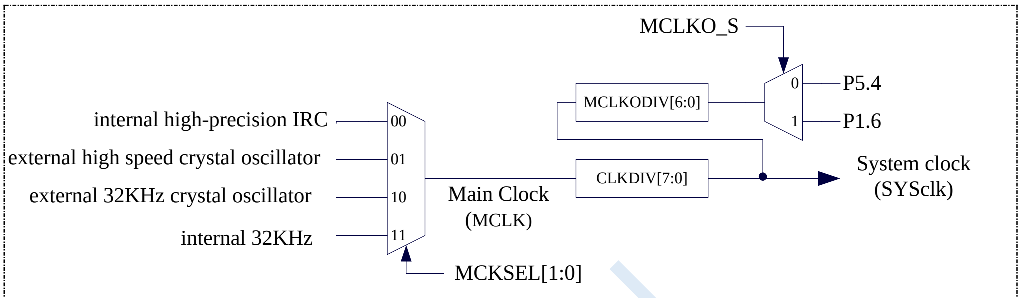


6. Clock, Reset and Power Management

6.1 System Clock Control

The system clock controller provides the clock sources for the microcontroller's CPU and all peripherals. One of the following three clock sources can be selected as the system clock: internal high-precision IRC, internal 32KHz IRC with large error, external crystal oscillator. Every clock source can be enabled or disabled respectively using programs, as well as internally provide clock divider for the purpose of reducing power consumption.

When the microcontroller enters Power-down mode, the clock controller will shut down all clock sources.



System clock structure diagram

Related registers

Symbol	Description	Address	Bit Address and Symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
CKSEL	Clock selection register	FE00H	-						MCKSEL[1:0]		xxxx,xx00
CLKDIV	Clock Division Register	FE01H									nnnn,nnnn
IRCCR	Internal Oscillator control register	FE02H	ENIRC	-	-	-	-	-	-	IRCST	1xxx,xxx0
XOSCCR	External Oscillator control register	FE03H	ENXOSC	XITYPE	XCFILTER[1:0]		GAIN	-	-	XOSCST	00xx,xxx0
IRC32KCR	Internal 32KHz Oscillator control register	FE04H	ENIRC32K	-	-	-	-	-	-	IRC32KST	0xxx,xxx0
MCLKOCR	Main clock output control register	FE05H	MCLKO_S	MCLKODIV[6:0]							0000,0000
X32KCR	External 32KHz Oscillator control register	FE08H	ENX32K	GAIN32K		-	-	-	-	X32KST	00xx,xxx0

6.1.1 System clock selection register (CKSEL)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
CKSEL	FE00H	-						MCKSEL[1:0]	

MCKSEL[1:0]: Main clock source selection

MCKSEL[1:0]	Main clock source
00	internal high speed high precision IRC
01	external high speed crystal oscillator
10	external 32KHz crystal oscillator
11	internal 32KHz low speed IRC

6.1.2 Clock Division register (CLKDIV)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
CLKDIV	FE01H								

CLKDIV: Main clock dividing factor. The system clock (SYSCLK) is the clock signal of main clock (MCLK) after being divided.

CLKDIV	System clock frequency
0	MCLK/1
1	MCLK/1
2	MCLK/2
3	MCLK/3
…	…
x	MCLK/x

…	…
255	MCLK/255

Note: After the user program is reset, the system will set the initial value of this register automatically according to the frequency division factor required for the operating frequency set during the last ISP download.

6.1.3 Internal high speed high precision IRC control register (IRCCR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
IRCCR	FE02H	ENIRC	-	-	-	-	-	-	IRCST

ENIRC: internal high speed high precision IRC enable bit

0: disable internal high-precision IRC

1: enable internal high-precision IRC

IRCST: internal high speed high precision IRC frequency stability flag (read-only)

After the internal IRC is enabled from the stopped state, it must take some time for the frequency of the oscillator to become stable. The clock controller will set the IRCST flag automatically after the internal oscillator frequency stabilizes. When the user program needs to switch the clock to internal IRC, ENIRC must be set at first to enable the oscillator and then keep polling the oscillator stable flag IRCST until the flag changes to 1.

6.1.4 External Oscillator control register (XOSCCR)

STC8H8K86U series **(Note: SFR assignment is not compatible with other STC8H series)**

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
XOSCCR	FE03H	ENXOSC	XITYPE	GAIN	-	XCFILTER[1:0]		-	XOSCST

STC8H other series

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
XOSCCR	FE03H	ENXOSC	XITYPE	XCFILTER[1:0]		GAIN	-	-	XOSCST

ENXOSC: external oscillator enable bit

0: disable external oscillator

1: enable external oscillator

XITYPE: external clock source type

0: The external clock source is the external clock signal (or active crystal). The signal source only needs to be connected to the XTALI(P1.7) of microcontroller. (At this time, P1.6 is fixed in high-impedance input mode, which can be used to read external digital signals or as ADC input, but it is generally not recommended to be used, because the high-frequency oscillation signal of the adjacent P1.7 will affect P1.6 signal.)

1: The external clock source is a crystal oscillator which is connected to XTALI (P1.7) and XTALO (P1.6) of microcontroller.

XCFILTER[1:0]: External crystal oscillator anti-interferencecontrol register.

00: When the external crystal oscillator frequency is 48M and below.

01: When the external crystal oscillator frequency is 24M and below

1x: When the external crystal oscillator frequency is 12M and below

Note: This register is currently only valid for B version chips of STC8H3K64S4 series, B version chips of STC8H3K64S2 series, STC8A8K64D4 series and STC8H8K64U series chips. It needs to be set carefully. Improper setting may cause the clock supplied by the external crystal oscillator to the internal MCU to be abnormal.

GAIN: External crystal oscillator oscillator gain control bits

0: disable oscillator gain (low gain)

1: enable oscillator gain (high gain)

XOSCST: external crystal oscillator frequency stability flag (read-only)

After the external crystal oscillator is enabled from the stopped state, it must take some time for the frequency of the oscillator to become stable. The clock controller will set the XOSCSTflag automatically after the oscillator frequency stabilizes. When the user program needs to switch the clock to external crystal oscillator, ENXOSC must be set at first to enable the oscillator and then keep polling the oscillator stable flag XOSCST until the flag changes to 1.

6.1.5 External 32KHz Oscillator control register (X32KCR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
X32KCR	FE08H	ENX32K	GAIN32K	-	-	-	-	-	X32KST

ENX32K: External 32KHz Oscillator enable bit
0: disable external 32KHz Oscillator
1: enable external 32KHz Oscillator
GAIN32K: External 32KHz Oscillator gain control bit
0: disable external 32KHz Oscillator (low gain)
1: enable external 32KHz Oscillator (high gain)
X32KST: external 32KHz Oscillator frequency stability flag (read-only)

6.1.6 Internal 32KHz low speed IRC control register (IRC32KCR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
IRC32KCR	FE04H	ENIRC32K	-	-	-	-	-	-	IRC32KST

ENIRC32K: internal 32KHz low speed IRC enable bit
0: disable internal 32KHz low speed IRC
1: enable internal 32KHz low speed IRC
IRC32KST: internal 32KHz low speed IRC frequency stability flag (read-only)
After the internal 32KHz low speed IRC is enabled from the stopped state, it must take some time for the frequency of the oscillator to become stable. The clock controller will set the IRC32KST flag automatically after the internal oscillator frequency stabilizes. When the user program needs to switch the clock to the internal 32KHz low speed IRC, ENIRC32K must be set at first to enable the oscillator and then keep polling the oscillator stable flag IRC32KST until the flag changes to 1.

6.1.7 Main clock output control register (MCLKOCR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
MCLKOCR	FE05H	MCLKO_S	MCLKODIV[6:0]						

MCLKODIV[6:0]: Main clock output division factor
(**Note: The clock source of main clock output is system clock divided by CLKDIV**)

MCLKODIV[6:0]	Divided system clock output frequency
0000000	No clock out
0000001	SYSClk/1
0000010	SYSClk /2
0000011	SYSClk /3
…	…
1111110	SYSClk /126
1111111	SYSClk /127

MCLKO_S: Main clock output pin selection
0: Main clock output to P5.4
1: Main clock output to P1.6

6.2 STC8H series internal IRC frequency adjustment

All STC8H series of microcontrollers integrate a high-precision internal IRC oscillator. The ISP download software will automatically adjust the frequency according to the frequency selected / set by the user when users download user program using ISP. The general frequency value can be adjusted below ±0.3%. The temperature drift of the adjusted frequency can be -1.35% ~ 1.30% within the full temperature range (-40℃~85℃).

The internal IRC of the STC8H series has two frequency bands whose center frequencies are 20MHz and 35MHz respectively. The adjustment range of the 20M band is about 15.5MHz to 27MHz. The adjustment range of the 35M band is about 27.5MHz to 47MHz. Note: Different chips or different manufacturing batches may have manufacturing errors of about 5%. After actual testing, the maximum operating frequency of some chips can only be 39.5MHz. For safety reasons, it is recommended that the IRC frequency is set not higher than 35MHz during ISP download.

Note: For general users, the adjustment of the internal IRC frequency can be ignored because the frequency adjustment is automatically completed when the ISP is downloaded. Therefore, if the user does not need to adjust the frequency by itself, the following four registers cannot be modified at will, otherwise the operating frequency may change.

If the user needs to dynamically select the preset frequency of the chip in his own code, please refer to the preset frequency list and the sample program of "User-Defined Internal IRC Frequency".

Internal IRC frequency adjustment is mainly adjusted using the following 4 registers.

Related registers

Symbol	Description	Address	Bit Address and Symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
IRCBAND	IRC band selection register	9DH	-	-	-	-	-	-	-	SEL	xxxx,xxxn
LIRTRIM	IRC frequency trim register	9EH	-	-	-	-	-	-	LIRTRIM[1:0]		xxxx,xxnn
IRTRIM	IRC frequency adjustment register	9FH	IRTRIM[7:0]								nnnn,nnnn
CLKDIV	Clock Divide Register	FE01H	CLKDIV[7:0]								nnnn,nnnn

6.2.1 IRC band selection register (IRCBAND)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
IRCBAND	9DH	-	-	-	-	-	-	-	SEL

SEL: band selection
0: Select 20MHz band
1: Select 35MHz band

6.2.2 Internal IRC Frequency Adjustment Register (IRTRIM)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
IRTRIM	9FH	IRTRIM[7:0]							

IRTRIM[7:0]: Internal high-precision IRC frequency adjustment register
IRTRIM can adjust 256 levels of IRC frequency. The frequency value adjusted by each level is linearly distributed as a whole, and there may be local fluctuations. Macroscopically, the frequency adjusted by each stage is about 0.24%, that is, the frequency when the IRTRIM is (n + 1) is about 0.24% faster than the frequency when the IRTRIM is (n). However, not every level of IRC frequency adjustment is 0.24% (the maximum value of the adjusted frequency of each level is about 0.55%, the minimum value is about 0.02%, and the overall average value is about 0.24%), so it will cause local fluctuations.

6.2.3 Internal IRC frequency trim register (LIRTRIM)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
LIRTRIM	9EH	-	-	-	-	-	-	IRTRIM[1:0]	

LIRTRIM[1:0]: Internal high precision IRC frequency trim register
LIRTRIM can adjust the IRC frequency in 3 levels. The frequency range adjusted by the 3 levels is shown in the table below.

LIRTRIM[1:0]	Adjusted frequency range
00	Not fine-tuned
01	Adjusted about 0.10%
10	Adjusted about 0.04％
11	Adjusted about 0.10％

6.2.4 Clock Divide Register (CLKDIV)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
CLKDIV	FE01H								

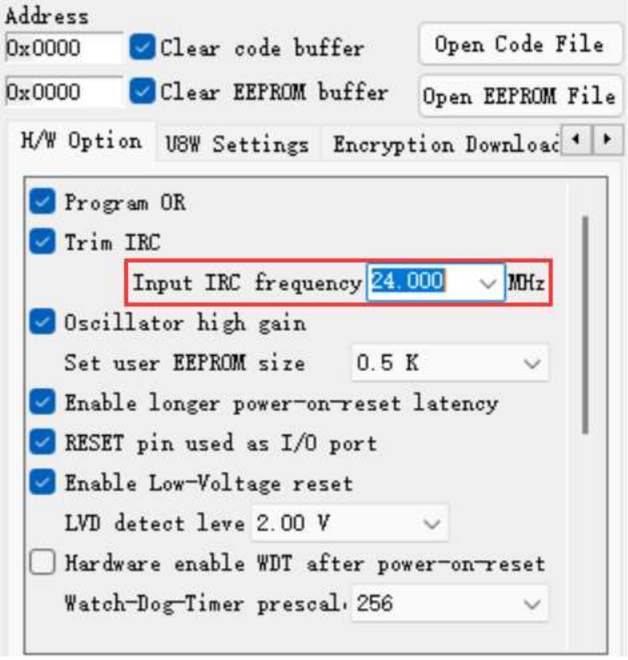
CLKDIV: Frequency division factor of the main clock. The system clock SYSCLK is a clock signal obtained by dividing the main clock MCLK.

CLKDIV	System clock frequency
0	MCLK/1
1	MCLK/1
2	MCLK/2
3	MCLK/3
…	…
x	MCLK/x
…	…
255	MCLK/255

The adjustable ranges of the two frequency bands within the STC8H series of microcontrollers are 15.5MHz to 27MHz and 25.3MHz to 43.6MHz respectively. Although the upper limit of the 35MHz frequency band can be adjusted to more than 40MHz, the internal Flash program memory of the chip cannot run at a speed of more than 40MHz. Therefore, you should set the internal IRC frequency not higher than 40MHz when using ISP download. It is generally recommended that the frequency should be set below 35MHz. If you need a lower operating frequency, you can use the CLKDIV register to divide the adjusted frequency. For example, if you need a frequency of 11.0592MHz, and this frequency cannot be obtained using the internal IRC direct adjustment, but the internal IRC can be adjusted to 22.1184MHz, you can get 11.0592MHz by dividing by 2 with CLKDIV.

6.2.5 Example of fine-tuning to get a user frequency of 3MHz

To get a frequency of 3MHz, you can use the method of 24MHz divided by 8.
Select the internal IRC operating frequency as 24MHz firstly when downloading the ISP, as shown in the figure below.



Address
0x0000 ☑ Clear code buffer
0x0000 ☑ Clear EEPROM buffer

☑ Program OR
☑ Trim IRC
Input IRC frequency 24.000 MHz
☑ Oscillator high gain
Set user EEPROM size 0.5 K
☑ Enable longer power-on-reset latency
☑ RESET pin used as I/O port
☑ Enable Low-Voltage reset
LVD detect leve 2.00 V
□ Hardware enable WDT after power-on-reset
Watch-Dog-Timer prescal 256

Then select the internal IRC as the clock source in the code and use the CLKDIV register to divide by 8.

C language code

```
// Operating frequency for test is 24MHz

#include "reg51.h"
#include "intrins.h"

#define     CKSEL       (*(unsigned char volatile xdata *)0xfe00)
#define     CLKDIV      (*(unsigned char volatile xdata *)0xfe01)
#define     IRCCR       (*(unsigned char volatile xdata *)0xfe02)
#define     XOSCCR      (*(unsigned char volatile xdata *)0xfe03)
#define     IRC32KCR    (*(unsigned char volatile xdata *)0xfe04)

sfr         P_SW2       =       0xba;
sfr         IRTRIM      =       0x9f;

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P_SW2 = 0x80;
        CKSEL = 0x00;                                   // Select internal IRC (default)
        CLKDIV = 0x08;                                  // Clock divided by 8
        P_SW2 = 0x00;

        IRTRIM++;    // Fine-tune the IRC frequency up 3 ‰ (pay attention to judging the boundary)
//      IRTRIM--;    // Fine-tune the IRC frequency down 3 ‰ (pay attention to judging the boundary)

        while (1);
}
```

Assembly code

```
; Operating frequency for test is 24MHz

P_SW2           DATA            0BAH
IRTRIM          DATA            09FH

CKSEL           EQU             0FE00H
CLKDIV          EQU             0FE01H
IRCCR           EQU             0FE02H
XOSCCR          EQU             0FE03H
IRC32KCR        EQU             0FE04H

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN

                ORG             0100H
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             P_SW2,#80H
                MOV             A,#00H                          ; Select internal IRC
                MOV             DPTR,#CKSEL
                MOVX            @DPTR,A
                MOV             A,#08H                          ; Clock divided by 8
                MOV             DPTR,#CLKDIV
                MOVX            @DPTR,A
                MOV             P_SW2,#00H

                INC             IRTRIM   ;IRC Fine-tune frequency up 3 ‰ (pay attention to judging boundaries)
;               DEC             IRTRIM   ;IRC Fine-tune frequency down 3 ‰ (pay attention to judging boundaries)

                JMP             $

                END
```

6.3 System reset

There are two types of resets in STC8H series of microcontrollers, hardware reset and software reset.

When hardware reset occurs, all registers are reset to their original values and the system rereads all hardware options. At the same time, after being powered on, the system will wait for some time according to the hardware power-on wait time option set. Hardware reset includes,

- Power-on reset, POR, about 1.7V
- Low-voltage detection reset, LVD-RESET (2.0V, 2.4V, 2.7V, 3.0V)
- RST pin reset (Low-level reset)
- Watch-Dog-Timer reset

When software reset occurs, all the registers values are reset to the initial value except that the clock-related registers remain unchanged. Software reset does not re-read all hardware options. Software reset mainly includes,

- Write SWRST bit in IAP_CONTR register to trigger reset

Related registers

Symbol	Description	Address	Bit Address and Symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
WDT_CONTR	Watchdog control register	C1H	WDT_FLAG	-	EN_WDT	CLR_WDT	IDL_WDT	WDT_PS[2:0]			0x00,0000
IAP_CONTR	IAP Control Register	C7H	IAPEN	SWBS	SWRST	CMD_FAIL	-	IAP_WT[2:0]			0000,x000
RSTCFG	Reset Configuration Registe	FFH	-	ENLVR	-	P54RST	-	-	LVDS[1:0]		x0x0,xx00

6.3.1 Watch dog timer reset（WDT_CONTR）

In industrial/automotive electronic control/aerospace need high reliability in the system, in order to prevent "system in exceptional cases, the disturbance of MCU/CPU program run fly, resulting in abnormal system for a long time work", is usually introduced watchdog, if the MCU/CPU is not within the stipulated time visit watchdog, according to the requirement as MCU/CPU in abnormal state, the watchdog will force MCU/CPU reset, enables the system to perform user program from the very beginning.

The STC8 series Guard Dog reset is one of the hardware reset in thermal boot reset. STC8 series SCM introduces this function, which makes the reliability design of SCM system more convenient and simple. After the reset state of STC8 series watchdog, the system is fixed to start from THE ISP monitor area, independent of the SWBS of IAP_CONTR register before the reset of watchdog (Note: This is different from STC15 series MCU).

WDT_CONTR (Watchdog control register)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
WDT_CONTR	C1H	WDT_FLAG	-	EN_WDT	CLR_WDT	IDL_WDT	WDT_PS[2:0]		

WDT_FLAG : WDT reset flag.

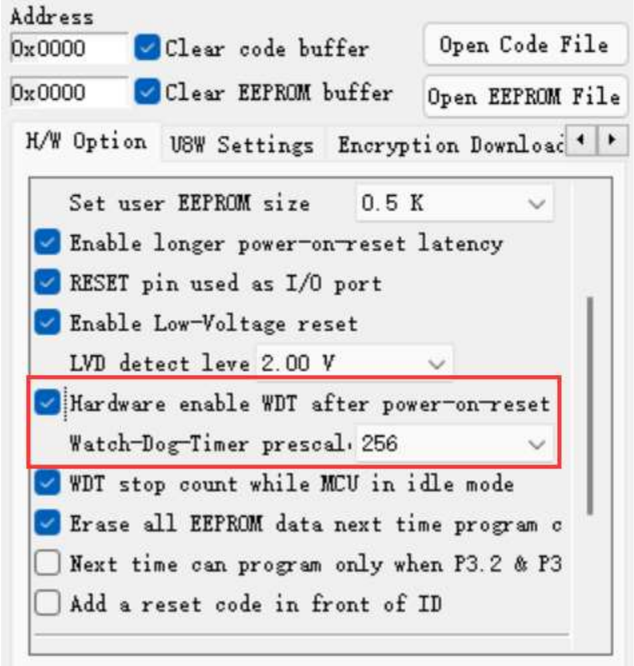
When WDT overflows, this bit is set by hardware automatically. This bit should be cleared by software.

EN_WDT: WDT enable bit.

0: No operation

1: WDT is started.

Note: The watchdog timer can be started by software or hardware automatically. Once the watchdog timer is started, the software cannot be shut down. The SCM must be recharged before it can be shut down. Software to start the watchdog only needs to write 1 to the EN_WDT bit. If you need the hardware to boot the watchdog, you need to set it up at your ISP when you download it, as shown in the figure below:



Address

0x0000 ☑ Clear code buffer | Open Code File

0x0000 ☑ Clear EEPROM buffer | Open EEPROM File

H/W Option | USW Settings | Encryption Downloa ◂ ▸

Set user EEPROM size 0.5 K

☑ Enable longer power-on-reset latency

☑ RESET pin used as I/O port

☑ Enable Low-Voltage reset

LVD detect leve 2.00 V

☑ Hardware enable WDT after power-on-reset

Watch-Dog-Timer prescal 256

☑ WDT stop count while MCU in idle mode

☑ Erase all EEPROM data next time program c

☐ Next time can program only when P3.2 & P3

☐ Add a reset code in front of ID

CLR_WDT: WDT clear bit.
0: No operation
1: WDT is cleared. This bit will be cleared by hardware automatically.
IDL_WDT: WDT control bit in IDLE mode.
0: WDT is disabled in IDLE mode.
1: WDT is enabled in IDLE mode, and the WDT will continue counting.
WDT_PS[2:0]: Watchdog timer clock division factor

WDT_PS[2:0]	division factor	Overflow time @12MHz	Overflow time @20MHz
000	2	≈ 65.5 ms	≈ 39.3 ms
001	4	≈ 131 ms	≈ 78.6 ms
010	8	≈ 262 ms	≈ 157 ms
011	16	≈ 524 ms	≈ 315 ms
100	32	≈ 1.05 s	≈ 629 ms
101	64	≈ 2.10 s	≈ 1.26 s
110	128	≈ 4.20 s	≈ 2.52 s
111	256	≈ 8.39 s	≈ 5.03 s

The WDT overflow time is determined by the following equation:

$$\text{WDT overflow time} = \frac{12 \times 32768 \times 2^{(WDT_{PS}+1)}}{SYSclk} \ (s)$$

6.3.2 Software reset（IAP_CONTR）

IAP_CONTR (IAP Control Register)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
IAP_CONTR	C7H	IAPEN	SWBS	SWRST	CMD_FAIL	-			

SWBS: Software boot selection bit
0: The microcontroller executes the code from user program space (main flash memory) after the software reset. The data in the user data space remains unchanged.
1: The microcontroller executes the code from ISP space after the software reset. The data in the user data space is initialized.
SWRST: Software reset trigger bit.
0: No operation
1: Trigger software reset.

6.3.3 Low voltage reset (RSTCFG)

RSTCFG (Reset Configuration Register)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
RSTCFG	FFH	-	ENLVR	-	P54RST	-	-	LVDS[1:0]	

ENLVR: Low voltage detection reset enable bit

0: Disable low voltage detection reset. When the system detects a low-voltage event, a low-voltage interrupt will occur.

1: Enable low voltage detection reset. When the system detects a low-voltage event, it will reset automatically.

P54RST: RST pin function selection bit

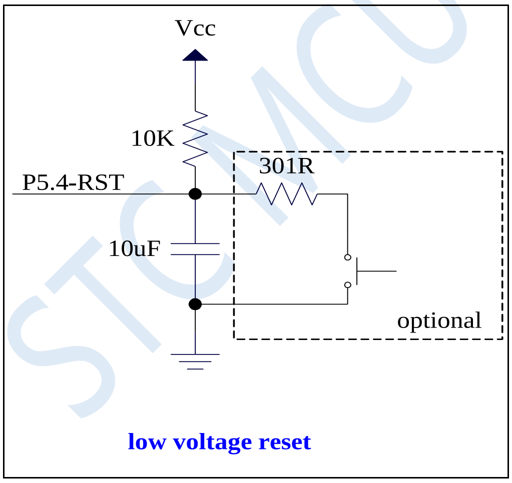
0: RST pin is used as common I/O (P5.4).

1: RST pin is used as reset pin. **(Low-level reset)**

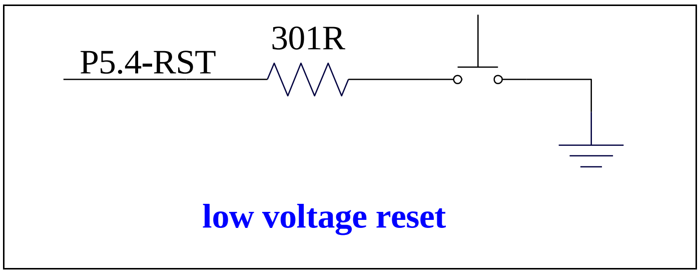
LVDS[1:0] : Low voltage detection threshold voltage setting bits

LVDS[1:0]	Low voltage detection threshold voltage
00	2.0V
01	2.4V
10	2.7V
11	3.0V

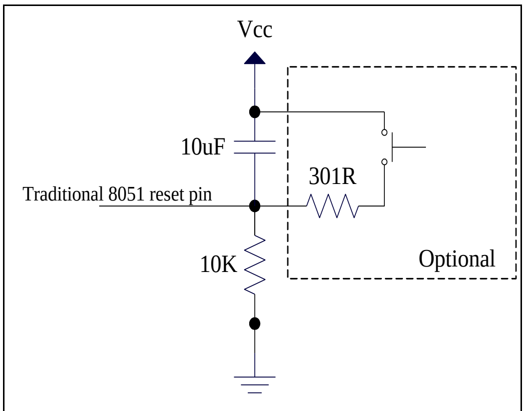
6.3.4 Low voltage power-on reset reference circuit (generally not required)



6.3.5 Low voltage button reset reference circuit



6.3.6 Traditional 8051 high voltage power-on reset reference circuit

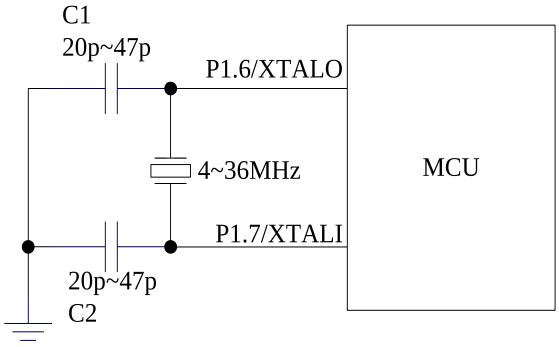


Traditional 8051 high voltage reset circuit

The picture above shows the high-level reset circuit of the traditional 8051. The reset of the STC8H is a low-level reset, which is different from the traditional reset circuit.

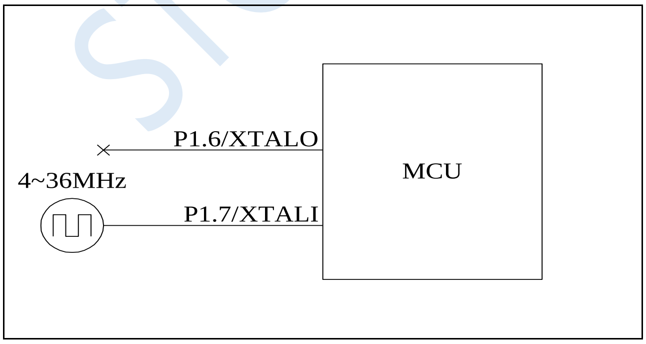
6.4 External crystal oscillator and external clock circuit

6.4.1 External crystal input circuit



Note: The two capacitors C1 and C2 must not be omitted. Without these two capacitors, the crystal oscillator may not oscillate. In addition, even if the crystal oscillator can start to oscillate, the MCU will increase the power consumption of 5~8mA.

6.4.2 External clock input circuit (P1.6 cannot be used as general I/O)



6.5 Clock stop / Power Saving Mode and System Power Management

Symbol	Description	Address	Bit Address and Symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
PCON	Power control register	87H	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000

6.5.1 Power control register (PCON)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
PCON	87H	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL

LVDF: Low voltage detection flag. When the system detects a low-voltage event, it is set by the hardware automatically and an interrupt request to the CPU ocuurs. It should be cleared by user software.

POF: Power-On reset flag. It is set by the hardware automatically when power-off-on action occurs.

PD: Clock stop mode / Power-Down mode / Power stop mode control bit

0: No operation.

1: Make the microcontroller entering Clock stop mode / Power-Down mode / Power stop mode. CPU and all peripherals stop working. It is cleared by hardware automatically after the microcontroller wakes up. (Note: In the clock stop mode, the CPU and all peripherals stop working, but the data in the SRAM and XRAM remain unchanged.)

IDL: IDLE mode control bit

0: No operation.

1: Make the microcontroller entering IDLE mode. CPU stops working and all peripherals keep working. It is cleared by hardware automatically after the microcontroller wakes up.

Note: In the power saving mode when the clock is stopped, it is not recommended to start the LVD and comparator, or hardware system will automatically start internal high precision 1.19 V reference source which has a corresponding temperature drift and calibration circuit, and leads to about 300uA extra power consumption. After MCU enters into the clock stopped vibration mode, the working voltage of 3.3V only takes about 0.4uA current, so it is not recommended to open the LVD and comparator when the MCU enters the clock stop mode. If it is really needed, it is recommended to turn on the power-down wake-up timer. The power-down wake-up timer will only increase the power consumption of about 1.4uA, which is acceptable for general systems. The MCU is awakened every 5 seconds by the power-down wake-up timer. After wake-up LVD, comparator and ADC can be used to detect the external battery voltage. The detection took about 1ms. Then it enters the clock shutdown/power-saving mode. In this way, the increased average current was less than 1uA, and the overall power consumption was about 2.8uA (0.4uA + 1.4uA +1uA).

Power-down mode can be woke up by one of the following interrupts, INT0(P3.2), INT1(P3.3), INT2(P3.6), INT3(P3.7), INT4(P3.0), T0(P3.4), T1(P3.5), T2(P1.2), T3(P0.4), T4(P0.6), RXD(P3.0/P3.6/P1.6/P4.3), RXD2(P1.4/P4.6), RXD3(P0.0/P5.0), RXD4(P0.2/P5.2), I2C_SDA(P1.4/P2.4/P3.3), Comparator, LVD and Power-down wake-up timer.

6.6 Power-down wake-up timer

The internal power-down wake-up timer is a 15-bit counter (composed of {WKTCH[6:0],WKTCL[7:0]}), which is used to wake up an MCU in power off mode.

6.6.1 Power-down wake-up timer count register (WKTCL,WKTCH)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
WKTCL	AAH								
WKTCH	ABH	WKTEN							

WKTEN: wake-up timer enable bit
 0: No operation
 1: WKT is started.

If the built-in power-down wake-up dedicated timer of STC8 series microcontrollers is enabled (set the WKTEN bit in the WKTCH register to 1 through software), when the MCU enters the power-down mode/stop mode, the power-down wake-up dedicated timer starts counting, when the count value is equal to When the values set by the user are equal, the power-down wake-up dedicated timer will wake up the MCU. After the MCU wakes up, the program executes from the statement next to the statement that set the MCU to enter the power-down mode last time. After waking up from power down, the sleep time of the MCU in power down mode can be obtained by reading the contents of WKTCH and WKTCL.

Please note here: The value written by the user in the register {WKTCH[6:0], WKTCL[7:0]} must be one less than the actual count value. If the user needs to count 10 times, write 9 into the register {WKTCH[6:0], WKTCL[7:0]}. Similarly, if the user needs to count 32767 times, he should write 7FFEH (ie 32766) in {WKTCH[6:0], WKTCL[7:0]}. (Count value 0 and count value 32767 are internal reserved values and cannot be used by users). The internal power-down wake-up timer has its own internal clock, and the time for the power-down wake-up timer to count once is determined by this clock. The clock frequency of the internal power-down wake-up timer is about 32KHz, and the error is relatively large. Users can read the contents of RAM area F8H and F9H (F8H stores the high byte of frequency, F9H stores the low byte) to obtain the clock frequency recorded by the internal power-down wake-up dedicated timer when it leaves the factory.

The formula for calculating the counting time of the dedicated timer for power-down wake-up is as follows: (Fwt is the clock frequency of the dedicated timer for internal power-down wake-up we obtained from RAM area F8H and F9H):

$$\text{Power down wakeup timer timing time} = \frac{106 \times 16 \times \text{count times}}{\text{Fwt}} \ (\text{us})$$

Assuming Fwt=32KHz, there are:

{WKTCH[6:0],WKTCL[7:0]}	Counting time of dedicated timer for wake-up after power failure
1	$10^6 \div 32K \times 16 \times (1+1) \approx 1ms$
9	$10^6 \div 32K \times 16 \times (1+9) \approx 5ms$
99	$10^6 \div 32K \times 16 \times (1+99) \approx 50ms$
999	$10^6 \div 32K \times 16 \times (1+999) \approx 0.5s$
4095	$10^6 \div 32K \times 16 \times (1+4095) \approx 2s$
32766	$10^6 \div 32K \times 16 \times (1+32767) \approx 16s$

6.7 Example Routines

6.7.1 System Clock Soure Selection

C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     CKSEL       (*(unsigned char volatile xdata *)0xfe00)
#define     CLKDIV      (*(unsigned char volatile xdata *)0xfe01)
#define     IRCCR       (*(unsigned char volatile xdata *)0xfe02)
#define     XOSCCR      (*(unsigned char volatile xdata *)0xfe03)
#define     IRC32KCR    (*(unsigned char volatile xdata *)0xfe04)

sfr         P_SW2       =       0xba;

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P_SW2 = 0x80;
        CKSEL = 0x00;                                   //Select internal IRC (default)
        P_SW2 = 0x00;

/*
        P_SW2 = 0x80;
        XOSCCR = 0xc0;                                  //Start external crystal
        while (!(XOSCCR & 1));                          //Waiting for the clock to stabilize
        CLKDIV = 0x00;                                  //Clock is not divided
        CKSEL = 0x01;                                           //Select external crystal
        P_SW2 = 0x00;
*/

/*
        P_SW2 = 0x80;
        IRC32KCR = 0x80;                                        //Start internal 32KHz IRC
        while (!(IRC32KCR & 1));                                //Waiting for the clock to stabilize
        CLKDIV = 0x00;                                          //Clock is not divided
        CKSEL = 0x03;                                           //Select internal 32KHz
        P_SW2 = 0x00;
*/
        while (1);
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz

P_SW2           DATA            0BAH

CKSEL           EQU             0FE00H
CLKDIV          EQU             0FE01H
IRCCR           EQU             0FE02H
XOSCCR          EQU             0FE03H
IRC32KCR        EQU             0FE04H

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN

                ORG             0100H
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             P_SW2,#80H
                MOV         A,#00H                  ;Select internal IRC (default)
                MOV         DPTR,#CKSEL
                MOVX        @DPTR,A
                MOV         P_SW2,#00H

;               MOV         P_SW2,#80H
;               MOV         A,#0C0H                 ;Start external crystal
;               MOV         DPTR,#XOSCCR
;               MOVX        @DPTR,A
;               MOVX        A,@DPTR
;               JNB         ACC.0,$-1               ;Waiting for the clock to stabilize
;               CLR         A                       ;Clock is not divided
;               MOV         DPTR,#CLKDIV
                MOVX        @DPTR,A
;               MOV         A,#01H                  ;Select external crystal
;               MOV         DPTR,#CKSEL
;               MOVX        @DPTR,A
;               MOV         P_SW2,#00H

;               MOV         P_SW2,#80H
;               MOV         A,#80H                  ;Start internal 32KHz IRC
;               MOV         DPTR,#IRC32KCR
;               MOVX        @DPTR,A
;               MOVX        A,@DPTR
;               JNB         ACC.0,$-1               ;Waiting for the clock to stabilize
;               CLR         A                       ;Clock is not divided
;               MOV         DPTR,#CLKDIV
;               MOVX        @DPTR,A
;               MOV         A,#03H                  ;Select internal 32KHz
;               MOV         DPTR,#CKSEL
;               MOVX        @DPTR,A
;               MOV         P_SW2,#00H

                JMP         $

                END
```

6.7.2 Main Clock Output

C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     MCLKOCR     (*(unsigned char volatile xdata *)0xfe05)

sfr         P_SW2       =       0xba;

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P_SW2 = 0x80;
//      MCLKOCR = 0x01;                             //Main clock output to P5.4
//      MCLKOCR = 0x02;                             //Divide the main clock by 2 and output to P5.4
        MCLKOCR = 0x04;                             //Divide the main clock by 4 and output to P5.4
//      MCLKOCR = 0x84;                             //Divide the main clock by 4 and output to P1.6
        P_SW2 = 0x00;

        while (1);
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz

P_SW2           DATA            0BAH

MCLKOCR         EQU             0FE05H

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN

                ORG             0100H
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             P_SW2,#80H
;               MOV             A,#01H                  ;Main clock output to P5.4
;               MOV             A,#02H                  ;Divide the main clock by 2 and output to P5.4
                MOV             A,#04H                  ;Divide the main clock by 4 and output to P5.4
;               MOV             A,#84H                  ;Divide the main clock by 4 and output to P1.6
                MOV             DPTR,#MCLKOCR
                MOVX            @DPTR,A
                MOV             P_SW2,#00H

                JMP             $

                END
```

6.7.3 Application of Watch-dog Timer

C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr     WDT_CONTR   =   0xc1;
sbit    P32         =   P3^2;

sfr     P0M1        =   0x93;
sfr     P0M0        =   0x94;
sfr     P1M1        =   0x91;
sfr     P1M0        =   0x92;
sfr     P2M1        =   0x95;
sfr     P2M0        =   0x96;
sfr     P3M1        =   0xb1;
sfr     P3M0        =   0xb2;
sfr     P4M1        =   0xb3;
sfr     P4M0        =   0xb4;
sfr     P5M1        =   0xc9;
sfr     P5M0        =   0xca;

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

//      WDT_CONTR = 0x23;                           //Watchdog enabled, overflow time is about 0.5s
        WDT_CONTR = 0x24;                           //Watchdog enabled, overflow time is about 1s
//      WDT_CONTR = 0x27;                           //Watchdog enabled, overflow time is about 8s
        P32 = 0;                                    //Test port

        while (1)
        {
//              WDT_CONTR = 0x33;                   // Clear watchdog timer, otherwise system reset
                WDT_CONTR = 0x34;                   // Clear watchdog timer, otherwise system reset
//              WDT_CONTR = 0x37;                   // Clear watchdog timer, otherwise system reset

                Display();                          //Call Display module
                Scankey();                          //Call Key scan module
                MotorDriver();                      //Call Motor drive module
        }
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz

WDT_CONTR   DATA        0C1H

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN

            ORG         0100H
MAIN:
            MOV         SP, #5FH
            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
            MOV         P3M1, #00H
            MOV         P4M0, #00H
            MOV         P4M1, #00H
            MOV         P5M0, #00H
            MOV         P5M1, #00H

;           MOV         WDT_CONTR,#23H          ;Watchdog enabled, overflow time is about 0.5s
            MOV         WDT_CONTR,#24H          ;Watchdog enabled, overflow time is about 1s
```

```
;           MOV         WDT_CONTR,#27H              ;Watchdog enabled, overflow time is about 8s
            CLR         P3.2                        ;Test port
LOOP:
;           MOV         WDT_CONTR,#33H              ; Clear watchdog timer, otherwise system reset
            MOV         WDT_CONTR,#34H              ; Clear watchdog timer, otherwise system reset
;           MOV         WDT_CONTR,#37H              ; Clear watchdog timer, otherwise system reset

            LCALL       DISPLAY                     ;Call Display module
            LCALL       SCANKEY                     ;Call Key scan module
            LCALL       MOTORDRIVER                 ;Call Motor drive module
            JMP         LOOP

            END
```

6.7.4 User Defined Downloading by Using Software Reset

C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr     IAP_CONTR   =   0xc7;
sbit    P32         =   P3^2;
sbit    P33         =   P3^3;

sfr     P0M1        =   0x93;
sfr     P0M0        =   0x94;
sfr     P1M1        =   0x91;
sfr     P1M0        =   0x92;
sfr     P2M1        =   0x95;
sfr     P2M0        =   0x96;
sfr     P3M1        =   0xb1;
sfr     P3M0        =   0xb2;
sfr     P4M1        =   0xb3;
sfr     P4M0        =   0xb4;
sfr     P5M1        =   0xc9;
sfr     P5M0        =   0xca;

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P32 = 1;                                        //Test port
    P33 = 1;                                        //Test port

    while (1)
```

```
        {
                if (!P32 && !P33)
                {
                        IAP_CONTR |= 0x60;                              //Reset to ISP when P3.2 and P3.3 are both 0
                }
        }
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz

IAP_CONTR       DATA            0C7H

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN

                ORG             0100H
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                SETB            P3.2
                SETB            P3.3
LOOP:
                JB              P3.2,LOOP
                JB              P3.3,LOOP
                MOV             IAP_CONTR,#60H          ;Reset to ISP when P3.2 and P3.3 are both 0
                JMP             $

                END
```

6.7.5 Low Voltage Detection

C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         RSTCFG      =       0xff;
#define     ENLVR               0x40                            //RSTCFG.6
#define     LVD2V0              0x00                            //LVD@2.0V
#define     LVD2V4              0x01                            //LVD@2.4V
#define     LVD2V7              0x02                            //LVD@2.7V
#define     LVD3V0              0x03                            //LVD@3.0V
sbit        ELVD        =       IE^6;
#define     LVDF                0x20                            //PCON.5
sbit        P32         =       P3^2;

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

void Lvd_Isr() interrupt 6
{
      PCON &= ~LVDF;                                            //Clear interrupt flag
      P32 = ~P32;                                               //Test port
}

void main()
{
      P0M0 = 0x00;
      P0M1 = 0x00;
      P1M0 = 0x00;
      P1M1 = 0x00;
      P2M0 = 0x00;
      P2M1 = 0x00;
      P3M0 = 0x00;
      P3M1 = 0x00;
      P4M0 = 0x00;
      P4M1 = 0x00;
      P5M0 = 0x00;
      P5M1 = 0x00;

      PCON &= ~LVDF;                                            //Test port
//    RSTCFG = ENLVR | LVD3V0;                                  //Low voltage reset when 3.0V is enabled, no LVD interrupt is generated
      RSTCFG = LVD3V0;                                          //Low voltage interrupt when 3.0V is enabled
      ELVD = 1;                                                 //Enable LVD interrupt
      EA = 1;

      while (1);
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz

RSTCFG      DATA        0FFH
ENLVR       EQU         40H                     ;RSTCFG.6
LVD2V0      EQU         00H                     ;LVD@2.0V
LVD2V4      EQU         01H                     ;LVD@2.4V
LVD2V7      EQU         02H                     ;LVD@2.7V
LVD3V0      EQU         03H                     ;LVD@3.0V

ELVD        BIT         IE.6
LVDF        EQU         20H                     ;PCON.5

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         0033H
            LJMP        LVDISR

            ORG         0100H
LVDISR:
            ANL         PCON,#NOT LVDF          ;Clear interrupt flag
            CPL         P3.2                    ;Test port
            RETI
MAIN:
            MOV         SP, #5FH
            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
            MOV         P3M1, #00H
            MOV         P4M0, #00H
            MOV         P4M1, #00H
            MOV         P5M0, #00H
            MOV         P5M1, #00H

            ANL         PCON,#NOT LVDF          ;LVDF flag needs to be cleared after power on
;           MOV         RSTCFG,#ENLVR | LVD3V0  ;Low voltage reset when 3.0V is enabled, no LVD interrupt is generated
            MOV         RSTCFG,#LVD3V0          ;Low voltage interrupt when 3.0V is enabled
            SETB        ELVD                    ;Enable LVD interrupt
            SETB        EA
            JMP         $
            END
```

6.7.6 Power Saving Mode

C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     IDL                 0x01                        //PCON.0
#define     PD                  0x02                        //PCON.1
sbit        P34         =       P3^4;
sbit        P35         =       P3^5;

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

void INT0_Isr() interrupt 0
{
        P34 = ~P34;                                         //Test port
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        EX0 = 1;                                            //Enable INT0 interrupt to wake up MCU
        EA = 1;
        _nop_();
        _nop_();
        _nop_();
        _nop_();
        PCON = IDL;                                         //MCU enters IDLE mode
//      PCON = PD;                                          //MCU enters power-down mode
        _nop_();
```

```
    _nop_();
    _nop_();
    _nop_();
    P35 = 0;

    while (1);
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz

IDL             EQU             01H                                 ;PCON.0
PD              EQU             02H                                 ;PCON.1

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0003H
                LJMP            INT0ISR

                ORG             0100H
INT0ISR:
                CPL             P3.4                                ;Test port
                RETI
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                SETB            EX0                                 ;Enable INT0 interrupt to wake up MCU
                SETB            EA
                NOP
                NOP
;               MOV             PCON,#IDL                           ;MCU enters IDLE mode
                MOV             PCON,#PD                            ;MCU enters power down mode
                NOP
                NOP
                NOP
                NOP
                CLR             P3.5                            ;Test port
                JMP             $

                END
```

6.7.7 Wake up MCU from Power Saving Mode using INT0/INT1/INT2/INT3/INT4 interrupts

C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         INTCLKO     =       0x8f;
#define     EX2                 0x10
#define     EX3                 0x20
#define     EX4                 0x40

sbit        P10         =       P1^0;
sbit        P11         =       P1^1;

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

void INT0_Isr() interrupt 0
{
    P10 = !P10;                                         //Test port
}

void INT1_Isr() interrupt 2
{
    P10 = !P10;                                         //Test port
}

void INT2_Isr() interrupt 10
{
    P10 = !P10;                                         //Test port
}

void INT3_Isr() interrupt 11
{
    P10 = !P10;                                         //Test port
}
```

```
void INT4_Isr() interrupt 16
{
        P10 = !P10;                                         //Test port
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        IT0 = 0;                                            //Enable INT0 rising edge and falling edge interrupts
//      IT0 = 1;                                            //Enable INT0 falling edge interrupt
        EX0 = 1;                                            //Enable INT0 interrupt

        IT1 = 0;                                            //Enable INT1 rising edge and falling edge interrupts
//      IT1 = 1;                                            //Enable INT1 falling edge interrupt
        EX1 = 1;                                            //Enable INT1 interrupt

        INTCLKO = EX2;                                      //Enable INT2 falling edge interrupt
        INTCLKO |= EX3;                                     //Enable INT3 falling edge interrupt
        INTCLKO |= EX4;                                     //Enable INT4 falling edge interrupt

        EA = 1;

        PCON = 0x02;                                        //MCU enters power down mode
        _nop_();                                            //Enter interrupt service routine immediately after wake-up from power
mode
        _nop_();
        _nop_();
        _nop_();

        while (1)
        {
                P11 = ~P11;
        }
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz

INTCLKO         DATA            8FH
EX2             EQU             10H
EX3             EQU             20H
EX4             EQU             40H

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
```

```
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN

                ORG             0003H
                LJMP            INT0ISR
                ORG             0013H
                LJMP            INT1ISR
                ORG             0053H
                LJMP            INT2ISR
                ORG             005BH
                LJMP            INT3ISR
                ORG             0083H
                LJMP            INT4ISR

                ORG             0100H
INT0ISR:
                CPL             P1.0                            ;Test port
                RETI
INT1ISR:
                CPL             P1.0                            ;Test port
                RETI
INT2ISR:
                CPL             P1.0                            ;Test port
                RETI
INT3ISR:
                CPL             P1.0                            ;Test port
                RETI
INT4ISR:
                CPL             P1.0                            ;Test port
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                CLR             IT0                             ;Enable INT0 rising edge and falling edge interrupts
;               SETB            IT0                             ;Enable INT0 falling edge interrupt
                SETB            EX0                             ;Enable INT0 interrupt
                CLR         IT1                     ;Enable INT1 rising edge and falling edge interrupts
;               SETB        IT1                     ;Enable INT1 falling edge interrupt
                SETB        EX1                     ;Enable INT1 interrupt

                MOV         INTCLKO,#EX2            ;Enable INT2 falling edge interrupt
                ORL         INTCLKO,#EX3            ;Enable INT3 falling edge interrupt
                ORL         INTCLKO,#EX4            ;Enable INT4 falling edge interrupt

                SETB        EA

                MOV         PCON,#02H               ;MCU enters power-down mode
                NOP                                 ;Enter interrupt service routine immediately after wake-up from power
mode
                NOP
                NOP
                NOP
LOOP:
                CPL         P1.1
                JMP         LOOP

                END
```

6.7.8 Wake up MCU from Power Saving Mode using T0/T1/T2/T3/T4 pin interrupts

C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         T2L         =       0xd7;
sfr         T2H         =       0xd6;
sfr         T3L         =       0xd5;
sfr         T3H         =       0xd4;
sfr         T4L         =       0xd3;
sfr         T4H         =       0xd2;
sfr         T4T3M       =       0xd1;
sfr         AUXR        =       0x8e;
sfr         IE2         =       0xaf;
#define     ET2                 0x04
#define     ET3                 0x20
#define     ET4                 0x40
sfr         AUXINTIF    =       0xef;
#define     T2IF                0x01
#define     T3IF                0x02
#define     T4IF                0x04

sbit        P10         =       P1^0;
sbit        P11         =       P1^1;

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

void TM0_Isr() interrupt 1
{
    P10 = !P10;                                 //Test port
}

void TM1_Isr() interrupt 3
{
    P10 = !P10;                                 //Test port
}

void TM2_Isr() interrupt 12
{
    P10 = !P10;                                 //Test port
 }

void TM3_Isr() interrupt 19
{
    P10 = !P10;                                 //Test port
}

void TM4_Isr() interrupt 20
{
    P10 = !P10;                                 //Test port
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    TMOD = 0x00;
    TL0 = 0x66;                                 //65536-11.0592M/12/1000
    TH0 = 0xfc;
    TR0 = 1;                                    //Start timer
    ET0 = 1;                                    //Enable timer interrupt

    TL1 = 0x66;                                 //65536-11.0592M/12/1000
    TH1 = 0xfc;
    TR1 = 1;                                    //Start timer
    ET1 = 1;                                    //Enable timer interrupt

    T2L = 0x66;                                 //65536-11.0592M/12/1000
```

```
        T2H = 0xfc;
        AUXR = 0x10;                                    //Start timer
        IE2 = ET2;                                      //Enable timer interrupt

        T3L = 0x66;                                     //65536-11.0592M/12/1000
        T3H = 0xfc;
        T4T3M = 0x08;                                   //Start timer
        IE2 |= ET3;                                     //Enable timer interrupt

        T4L = 0x66;                                     //65536-11.0592M/12/1000
        T4H = 0xfc;
        T4T3M |= 0x80;                                  //Start timer
        IE2 |= ET4;                                     //Enable timer interrupt

        EA = 1;

        PCON = 0x02;                                    //MCU enters power down mode
        _nop_();    //Does not enter the interrupt service routine immediately after wake-up from power down mode
                    //Instead, wait for the timer to overflow before entering the interrupt service routine.
        _nop_();
        _nop_();
        _nop_();

        while (1)
        {
                P11 = ~P11;
        }
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz

T2L             DATA            0D7H
T2H             DATA            0D6H
T3L             DATA            0D5H
T3H             DATA            0D4H
T4L             DATA            0D3H
T4H             DATA            0D2H
T4T3M           DATA            0D1H
AUXR            DATA            8EH

IE2             DATA            0AFH
ET2             EQU             04H
ET3             EQU             20H
ET4             EQU             40H

AUXINTIF        DATA            0EFH
T2IF            EQU             01H
T3IF            EQU             02H
T4IF            EQU             04H

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
```

```
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         000BH
            LJMP        TM0ISR
            ORG         001BH
            LJMP        TM1ISR
            ORG         0063H
            LJMP        TM2ISR
            ORG         009BH
            LJMP        TM3ISR
            ORG         00A3H
            LJMP        TM4ISR

            ORG         0100H
TM0ISR:
            CPL         P1.0                    ;Test port
            RETI
TM1ISR:
            CPL         P1.0                    ;Test port
            RETI
TM2ISR:
            CPL         P1.0                    ;Test port
            RETI
TM3ISR:
            CPL         P1.0                    ;Test port
            RETI
TM4ISR:
            CPL         P1.0                    ;Test port
            RETI

MAIN:
            MOV         SP, #5FH
            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
            MOV         P3M1, #00H
            MOV         P4M0, #00H
            MOV         P4M1, #00H
            MOV         P5M0, #00H
            MOV         P5M1, #00H

            MOV         TMOD,#00H
            MOV         TL0,#66H                ;65536-11.0592M/12/1000
            MOV         TH0,#0FCH
            SETB        TR0                     ;Start timer
            SETB        ET0                     ;Enable timer interrupt

            MOV         TL1,#66H                ;65536-11.0592M/12/1000
            MOV         TH1,#0FCH
            SETB        TR1                     ;Start timer
            SETB        ET1                     ;Enable timer interrupt
```

```
                MOV         T2L,#66H                ;65536-11.0592M/12/1000
                MOV         T2H,#0FCH
                MOV         AUXR,#10H               ;Start timer
                MOV         IE2,#ET2                ;Enable timer interrupt

                MOV         T3L,#66H                ;65536-11.0592M/12/1000
                MOV         T3H,#0FCH
                MOV         T4T3M,#08H              ;Start timer
                ORL         IE2,#ET3                ;Enable timer interrupt

                MOV         T4L,#66H                ;65536-11.0592M/12/1000
                MOV         T4H,#0FCH
                ORL         T4T3M,#80H              ;Start timer
                ORL         IE2,#ET4                ;Enable timer interrupt

                SETB        EA

                MOV         PCON,#02H               ;MCU enters power down mode
                NOP         ;Does not enter the interrupt service routine immediately after wake-up from power down mode
                            ;Instead, wait for the timer to overflow before entering the interrupt service routine.
                NOP
                NOP
                NOP
LOOP:
                CPL         P1.1
                JMP         LOOP

                END
```

6.7.9 Wake up MCU from Power Saving Mode using RxD/RxD2/RxD3/RxD4 pin interrupts

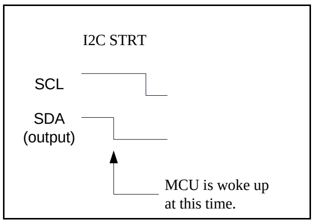
C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         IE2         =       0xaf;
#define     ES2                 0x01
#define     ES3                 0x08
#define     ES4                 0x10

sfr         P_SW1       =       0xa2;
sfr         P_SW2       =       0xba;

sbit        P11         =       P1^1;

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr     P3M0        =   0xb2;
sfr     P4M1        =   0xb3;
sfr     P4M0        =   0xb4;
sfr     P5M1        =   0xc9;
sfr     P5M0        =   0xca;

void UART1_Isr() interrupt 4
{
}

void UART2_Isr() interrupt 8
{
}

void UART3_Isr() interrupt 17
{
}

void UART4_Isr() interrupt 18
{
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P_SW1 = 0x00;                           //Wake up MCU on the falling edge of RXD / P3.0
//      P_SW1 = 0x40;                           //Wake up MCU on the falling edge of RXD_2/P3.6
//      P_SW1 = 0x80;                           //Wake up MCU on the falling edge of RXD_3/P1.6
//      P_SW1 = 0xc0;                           //Wake up MCU on the falling edge of RXD_4/P4.3

        P_SW2 = 0x00;                           //Wake up MCU on the falling edge of RXD2/P1.0
//      P_SW2 = 0x01;                           //Wake up MCU on the falling edge of RXD2_2/P4.6

        P_SW2 = 0x00;                           //Wake up MCU on the falling edge of RXD3/P0.0
//      P_SW2 = 0x02;                           //Wake up MCU on the falling edge of RXD3_2/P5.0

        P_SW2 = 0x00;                           //Wake up MCU on the falling edge of RXD4/P0.2
//      P_SW2 = 0x04;                           //Wake up MCU on the falling edge of RXD4_2/P5.2

        ES = 1;                                 //Enable UART interrupt
        IE2 = ES2;                              //Enable UART2 interrupt
        IE2 |= ES3;                             //Enable UART3 interrupt
        IE2| = ES4;                             //Enable UART4 interrupt
        EA = 1;

        PCON = 0x02;                            //MCU enters power-down mode
        _nop_();    //It will not enter the interrupt service routine after wake-up from power-down mode.
        _nop_();
        _nop_();
        _nop_();

        while (1)
        {
                P11 = ~P11;
        }
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz

IE2             DATA        0AFH
ES2             EQU         01H
ES3             EQU         08H
ES4             EQU         10H

P_SW1           DATA        0A2H
P_SW2           DATA        0BAH

P0M1            DATA        093H
P0M0            DATA        094H
P1M1            DATA        091H
P1M0            DATA        092H
P2M1            DATA        095H
P2M0            DATA        096H
P3M1            DATA        0B1H
P3M0            DATA        0B2H
P4M1            DATA        0B3H
P4M0            DATA        0B4H
P5M1            DATA        0C9H
P5M0            DATA        0CAH

                ORG         0000H
                LJMP        MAIN
                ORG         0023H
                LJMP        UART1ISR
                ORG         0043H
                LJMP        UART2ISR
                ORG         008BH
                LJMP        UART3ISR
                ORG         0093H
                LJMP        UART4ISR

                ORG         0100H
UART1ISR:
                RETI
UART2ISR:
                RETI
UART3ISR:
                RETI
UART4ISR:
                RETI

MAIN:
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                MOV         P_SW1,#00H                  ;Wake up MCU on the falling edge of RXD / P3.0
;               MOV         P_SW1,#40H                  ;Wake up MCU on the falling edge of RXD_2/P3.6
;               MOV         P_SW1,#80H                  ;Wake up MCU on the falling edge of RXD_3/P1.6
;               MOV         P_SW1,#0C0H                 ;Wake up MCU on the falling edge of RXD_4/P4.3

                MOV         P_SW2,#00H                  ;Wake up MCU on the falling edge of RXD2/P1.0
;               MOV         P_SW2,#01H                  ;Wake up MCU on the falling edge of RXD2_2/P4.6

                MOV         P_SW2,#00H                  ;Wake up MCU on the falling edge of RXD3/P0.0
;               MOV         P_SW2,#02H                  ;Wake up MCU on the falling edge of RXD3_2/P5.0

                MOV         P_SW2,#00H                  ;Wake up MCU on the falling edge of RXD4/P0.2
;               MOV         P_SW2,#04H                  ;Wake up MCU on the falling edge of RXD4_2/P5.2

                SETB        ES                          ;Enable UART interrupt
                MOV         IE2,#ES2                    ;Enable UART2 interrupt
                ORL         IE2,#ES3                    ;Enable UART3 interrupt
                ORL         IE2,#ES4                    ;Enable UART4 interrupt
                SETB        EA

                MOV         PCON,#02H                   ;MCU enters power down mode
                NOP         ;It will not enter the interrupt service routine after wake-up from power down mode.
                NOP
                NOP
                NOP
LOOP:
                CPL         P1.1
                JMP         LOOP

                END
```

6.7.10 Wake up MCU from Power Saving Mode using I2C SDA pin



C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P_SW2       =       0xba;
```

```
#define     I2CCFG              (*(unsigned char volatile xdata *)0xfe80)
#define     I2CSLCR             (*(unsigned char volatile xdata *)0xfe83)
#define     I2CSLST             (*(unsigned char volatile xdata *)0xfe84)

sbit        P11         =       P1^1;

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

void i2c_isr() interrupt 24
{
    P_SW2 |= 0x80;
    I2CSLST &= ~0x40;
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P_SW2 = 0x00;                               //Wake up MCU on the falling edge of SDA/P1.4
//  P_SW2 = 0x10;                               //Wake up MCU on the falling edge of SDA_2/P2.4
//  P_SW2 = 0x30;                               //Wake up MCU on the falling edge of SDA_4/P3.3
    P_SW2 |= 0x80;
    I2CCFG = 0x80;                              //Enable slave mode of I2C module
    I2CSLCR = 0x40;                             //Enable start signal interrupt
    EA = 1;

    PCON = 0x02;                                //MCU enters power-down mode, it will not enter the interrupt service
routine after power-down wake-up
    _nop_();
    _nop_();
    _nop_();
    _nop_();

    while (1)
    {
        P11 = ~P11;
    }
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz

P_SW2           DATA            0BAH

I2CCFG          XDATA           0FE80H
I2CSLCR         XDATA           0FE83H
I2CSLST         XDATA           0FE84H

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             00C3H
                LJMP            I2CISR

                ORG             0100H
I2CISR:
                PUSH            ACC
                PUSH            DPH
                PUSH            DPL
                ORL             PSW2,#80H
                MOV             DPTR,#I2CSLST
                MOVX            A,@DPTR
                ANL             A,#NOT 40H
                MOVX            @DPTR,A
                POP             DPL
                POP             DPH
                POP             ACC
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             P_SW2,#00H                  ;Wake up MCU on the falling edge of SDA/P1.4
//              MOV             P_SW2,#10H                  ;Wake up MCU on the falling edge of SDA_2/P2.4
//              MOV         P_SW2,#30H              ;Wake up MCU on the falling edge of SDA_4/P3.3
                ORL         P_SW2,#80H
                MOV         DPTR,#I2CCFG
                MOV         A,#80H
                MOVX        @DPTR,A                 ;Enable slave mode of I2C module
                MOV         DPTR,# I2CSLCR
                MOV         A,#40H                  ;Enable start signal interrupt
                SETB        EA

                MOV         PCON,#02H               ;MCU enters power down mode
                NOP         ;It will not enter the interrupt service routine after wake-up from power down mode.
                NOP
                NOP
                NOP
LOOP:
                CPL         P1.1
                JMP         LOOP

                END
```

6.7.11 Wake up MCU from Power Saving Mode using Power-down wake-up timer

C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         WKTCL       =       0xaa;
sfr         WKTCH       =       0xab;
sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sbit        P11         =       P1^1;

void main()
{
            P0M0 = 0x00;
            P0M1 = 0x00;
            P1M0 = 0x00;
            P1M1 = 0x00;
            P2M0 = 0x00;
            P2M1 = 0x00;
            P3M0 = 0x00;
            P3M1 = 0x00;
            P4M0 = 0x00;
            P4M1 = 0x00;
            P5M0 = 0x00;
            P5M1 = 0x00;
```

```
        WKTCL = 0xff;           // Set the power-down wake-up clock to be about 1 second
        WKTCH = 0x87;

        while (1)
        {
                _nop_();
                _nop_();
                PCON = 0x02;            //MCU enters power-down mode
                _nop_();
                _nop_();
                _nop_();
                _nop_();
                P11 = ~P11;
        }
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz
WKTCL   DATA            0AAH
WKTCH   DATA            0ABH

P0M1    DATA            093H
P0M0    DATA            094H
P1M1    DATA            091H
P1M0    DATA            092H
P2M1    DATA            095H
P2M0    DATA            096H
P3M1    DATA            0B1H
P3M0    DATA            0B2H
P4M1    DATA            0B3H
P4M0    DATA            0B4H
P5M1    DATA            0C9H
P5M0    DATA            0CAH

        ORG             0000H
                LJMP            MAIN

                ORG             0100H
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             WKTCL,#0FFH             ; Set the power-down wake-up clock to be about 1 second
                MOV             WKTCH,#87H

LOOP:
                NOP
                NOP
                MOV             PCON,#02H               ;MCU enters power-down mode
                NOP
                NOP
                NOP
                NOP
                CPL     P1.1
                JMP     LOOP

        END
```

6.7.12 Wake up MCU from Power Saving Mode using LVD interrupt (Recommended for use with power-down wake-up timer)

In the power saving mode when the clock is stopped, it is not recommended to start the LVD and comparator, or hardware system will automatically start internal high precision 1.19 V reference source which has a corresponding temperature drift and calibration circuit, and leads to about 300uA extra power consumption. After MCU enters into the clock stopped vibration mode, the working voltage of 3.3V only takes about 0.4uA current, so it is not recommended to open the LVD and comparator when the MCU enters the clock stop mode. If it is really needed, it is recommended to turn on the power-down wake-up timer. The power-down wake-up timer will only increase the power consumption of about 1.4uA, which is acceptable for general systems. The MCU is awakened every 5 seconds by the power-down wake-up timer. After wake-up LVD, comparator and ADC can be used to detect the external battery voltage. The detection took about 1ms. Then it enters the clock shutdown/power-saving mode. In this way, the increased average current was less than 1uA, and the overall power consumption was about 2.8uA (0.4uA + 1.4uA +1uA).

C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         RSTCFG      =     0xff;
#define     ENLVR             0x40                      //RSTCFG.6
#define     LVD2V0            0x00                      //LVD@2.0V
#define     LVD2V4            0x01                      //LVD@2.4V
#define     LVD2V7            0x02                      //LVD@2.7V
#define     LVD3V0            0x03                      //LVD@3.0V
sbit        ELVD        =     IE^6;
#define     LVDF              0x20                      //PCON.5

sbit        P10         =     P1^0;
sbit        P11         =     P1^1;

sfr         P0M1        =     0x93;
sfr         P0M0        =     0x94;
sfr         P1M1        =     0x91;
sfr         P1M0        =     0x92;
sfr         P2M1        =     0x95;
sfr         P2M0        =     0x96;
sfr         P3M1        =     0xb1;
sfr         P3M0        =     0xb2;
sfr         P4M1        =     0xb3;
sfr         P4M0        =     0xb4;
sfr         P5M1        =     0xc9;
sfr         P5M0        =     0xca;

void LVD_Isr() interrupt 6
{
    PCON &= ~LVDF;                                      //Clear interrupt flag
    P10 = !P10;                                         //Test port
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
```

```
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        PCON &= ~LVDF;                              //Interrupt flag needs to be cleared after power-on
        RSTCFG = LVD3V0;                            //Set the LVD voltage to 3.0V
        ELVD = 1;                                   //Enable LVD interrupt
        EA = 1;

        PCON = 0x02;                                //MCU enters power-down mode
        _nop_();    //Enter interrupt service routine immediately after wake-up from power mode
        _nop_();
        _nop_();
        _nop_();

        while (1)
        {
                P11 = ~P11;
        }
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz

RSTCFG          DATA            0FFH
ENLVR           EQU             40H                     ;RSTCFG.6
LVD2V0          EQU             00H                     ;LVD@2.0V
LVD2V4          EQU             01H                     ;LVD@2.4V
LVD2V7          EQU             02H                     ;LVD@2.7V
LVD3V0          EQU             03H                     ;LVD@3.0V

ELVD            BIT             IE.6
LVDF            EQU             20H                     ;PCON.5

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0033H
                LJMP            LVDISR
                ORG         0100H
LVDISR:
                ANL         PCON,#NOT LVDF              ;Clear interrupt flag
                CPL         P1.0                        ;Test port
                RETI

MAIN:
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                ANL         PCON,#NOT LVDF              ;Interrupt flag needs to be cleared after power-on
                MOV         RSTCFG,# LVD3V0             ;Set the LVD voltage to 3.0V
                SETB        ELVD                        ;Enable LVD interrupt
                SETB        EA

                MOV         PCON,#02H                   ;MCU enters power-down mode
                NOP         ;Enter interrupt service routine immediately after wake-up from power mode
                NOP
                NOP
                NOP
LOOP:
                CPL         P1.1
                JMP         LOOP

                END
```

6.7.13 Wake up MCU from Power Saving Mode using comparator interrupt (Recommended for use with power-down wake-up timer)

In the power saving mode when the clock is stopped, it is not recommended to start the LVD and comparator, or hardware system will automatically start internal high precision 1.19 V reference source which has a corresponding temperature drift and calibration circuit, and leads to about 300uA extra power consumption. After MCU enters into the clock stopped vibration mode, the working voltage of 3.3V only takes about 0.4uA current, so it is not recommended to open the LVD and comparator when the MCU enters the clock stop mode. If it is really needed, it is recommended to turn on the power-down wake-up timer. The power-down wake-up timer will only increase the power consumption of about 1.4uA, which is acceptable for general systems. The MCU is awakened every 5 seconds by the power-down wake-up timer. After wake-up LVD, comparator and ADC can be used to detect the external battery voltage. The detection took about 1ms. Then it enters the clock shutdown/power-saving mode. In this way, the increased average current was less than 1uA, and the overall power consumption was about 2.8uA (0.4uA + 1.4uA +1uA).

C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
```

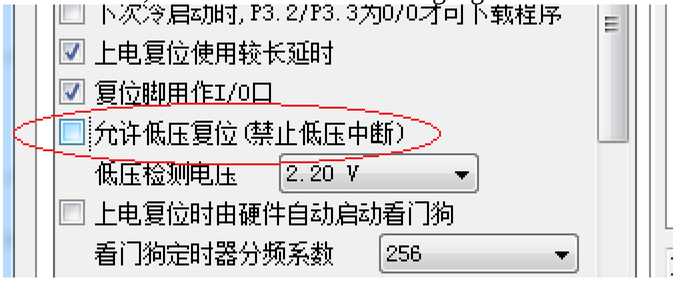
```
#include "intrins.h"

sfr         CMPCR1      =       0xe6;
sfr         CMPCR2      =       0xe7;

sbit        P10         =       P1^0;
sbit        P11         =       P1^1;

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

void CMP_Isr() interrupt 21
{
        CMPCR1 &= ~0x40;                                //Clear interrupt flag
        P10 = !P10;                                     //Test port
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        CMPCR2 = 0x00;
        CMPCR1 = 0x80;                                  //Enable comparator module
        CMPCR1 |= 0x30;                                 //Enable edge interrupt of comparator
        CMPCR1 &= ~0x08;                                //P3.6 is CMP + input pin
        CMPCR1 |= 0x04;                                 //P3.7 is CMP- input pin
        CMPCR1 |= 0x02;                                 //Enable Comparator output
        EA = 1;

        PCON = 0x02;                                    //MCU enters power-down mode
        _nop_();                     //Enter interrupt service routine immediately after wake-up from power mode
        _nop_();
        _nop_();
        _nop_();

        while (1)
        {
                P11 = ~P11;
        }
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz

CMPCR1          DATA            0E6H
CMPCR2          DATA            0E7H

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             00ABH
                LJMP            CMPISR

                ORG             0100H
CMPISR:
                ANL             CMPCR1,#NOT 40H             ;Clear interrupt flag
                CPL             P1.0                        ;Test port
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             CMPCR2,#00H
                MOV             CMPCR1,#80H                 ;Enable comparator module
                ORL             CMPCR1,#30H                 ;Enable edge interrupt of comparator
                ANL             CMPCR1,#NOT 08H             ;P3.6 is CMP + input pin
                ORL             CMPCR1,#04H                 ;P3.7 is CMP- input pin
                ORL             CMPCR1,#02H                 ;Enable Comparator output
                SETB            EA

                MOV             PCON,#02H                   ;MCU enters power-down mode
                NOP                         ;Enter interrupt service routine immediately after wake-up from power mode
                NOP
                NOP
                NOP

LOOP:
                CPL             P1.1
                JMP             LOOP

                END
```

6.7.14 Detect the Operating Voltage (Battery Voltage) using LVD

If you need to use LVD to detect the battery voltage, you need to remove the low-voltage reset function when downloading from the ISP, as shown in the following figure.



□ 下次冷启动时，P3.2/P3.3为0/0才可下载程序
☑ 上电复位使用较长延时
☑ 复位脚用作I/O口
□ 允许低压复位（禁止低压中断）
低压检测电压 2.20 V
□ 上电复位时由硬件自动启动看门狗
看门狗定时器分频系数 256

C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define FOSC                11059200UL
#define T1MS                (65536 - FOSC/4/100)

sfr         RSTCFG          =   0xff;
#define     LVD2V0              0x00                        //LVD@2.0V
#define     LVD2V4              0x01                        //LVD@2.4V
#define     LVD2V7              0x02                        //LVD@2.7V
#define     LVD3V0              0x03                        //LVD@3.0V

#define     LVDF                0x20                        //PCON.5

sfr         P0M1            =   0x93;
sfr         P0M0            =   0x94;
sfr         P1M1            =   0x91;
sfr         P1M0            =   0x92;
sfr         P2M1            =   0x95;
sfr         P2M0            =   0x96;
sfr         P3M1            =   0xb1;
sfr         P3M0            =   0xb2;
sfr         P4M1            =   0xb3;
sfr         P4M0            =   0xb4;
sfr         P5M1            =   0xc9;
sfr         P5M0            =   0xca;

void delay()
{
        int i;

        for (i=0; i<100; i++)
        {
                _nop_();
                _nop_();
                _nop_();
                _nop_();
        }
}

void main()
{
        unsigned char power;

        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        PCON &= ~LVDF;
        RSTCFG = LVD3V0;

        while (1)
        {
                power = 0x0f;

                RSTCFG = LVD3V0;
                delay();
                PCON &= ~LVDF;
                delay();
                if (PCON & LVDF)
                {
                        power >>= 1;
                        RSTCFG = LVD2V7;
                        delay();
                        PCON &= ~LVDF;
                        delay();
                        if (PCON & LVDF)
                        {
                                power >>= 1;
                                RSTCFG = LVD2V4;
                                delay();
                                PCON &= ~LVDF;
                                delay();
                                if (PCON & LVDF)
                                {
                                        power >>= 1;
                                        RSTCFG = LVD2V2;
                                        delay();
                                        PCON &= ~LVDF;
                                        delay();
                                        if (PCON & LVDF)
                                        {
                                                power >>= 1;
                                        }
                }
            }
        }
        RSTCFG = LVD3V0;
        P2 = ~power;      // P2.3 ~ P2.0 are used to display battery level
    }
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz

RSTCFG          DATA            0FFH
LVD2V0          EQU             00H                             ;LVD@2.0V
LVD2V4          EQU             01H                             ;LVD@2.4V
LVD2V7          EQU             02H                             ;LVD@2.7V
LVD3V0          EQU             03H                             ;LVD@3.0V

LVDF            EQU             20H                             ;PCON.5

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                JMP             MAIN

                ORG             0100H
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                ANL             PCON,#NOT LVDF
                MOV             RSTCFG,#LVD3V0
LOOP:
                MOV             B,#0FH

                MOV             RSTCFG,#LVD3V0
                CALL            DELAY
                ANL             PCON,#NOT LVDF
                CALL            DELAY
                MOV             A,PCON
                ANL             A,#LVDF
                JZ              SKIP
                MOV             A,B
                CLR             C
                RRC             A
                MOV             B,A

                MOV             RSTCFG,#LVD2V7
                CALL            DELAY
                ANL             PCON,#NOT LVDF
                CALL            DELAY
                MOV             A,PCON
                ANL             A,#LVDF
                JZ              SKIP
                MOV             A,B
                CLR             C
                RRC             A
                MOV             B,A

                MOV             RSTCFG,#LVD2V4
                CALL            DELAY
                ANL             PCON,#NOT LVDF
                CALL            DELAY
                MOV             A,PCON
                ANL             A,#LVDF
                JZ              SKIP
                MOV             A,B
                CLR             C
                RRC             A
                MOV             B,A

                MOV             RSTCFG,#LVD2V2
                CALL            DELAY
                ANL             PCON,#NOT LVDF
                CALL            DELAY
                MOV             A,PCON
                ANL             A,#LVDF
                JZ              SKIP
                MOV             A,B
                CLR             C
                RRC             A
                MOV             B,A

SKIP:
                MOV             A,B
                CPL             A
                MOV             P2,A                    ; P2.3 ~ P2.0 are used to display battery level
                JMP             LOOP

DELAY:
                MOV             R0,#100
NEXT:
                NOP
                NOP
                NOP
                NOP
                DJNZ            R0,NEXT
                RET

                END
```

7 Memory

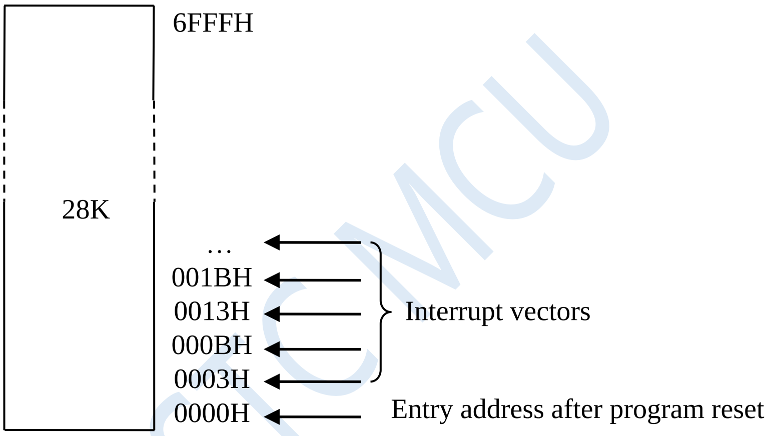
The STC8H series of microcontrollers have separate address spaces for Program Memory and Data Memory. Since no bus is provided for accessing external program memory, all program memory for all microcontrollers is on-chip Flash memory. The microcontrollers can not access external program memory.

Large-capacity data memory is integrated in STC8H series of microcontrollers. The data memory inside the STC8H series of microcontrollers is physically and logically separated into two address spaces: 256 bytes of internal RAM and internal extended RAM. The addresses of the high 128 bytes of internal RAM and special function registers (SFRs) overlap. They can be accessed through different addressing modes in actual use.

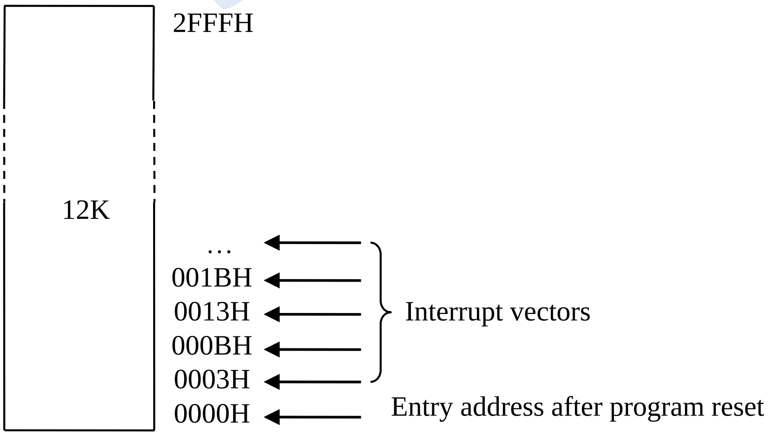
7.1 Program Memory

Program memory is used to store user programs, data, tables and other information.

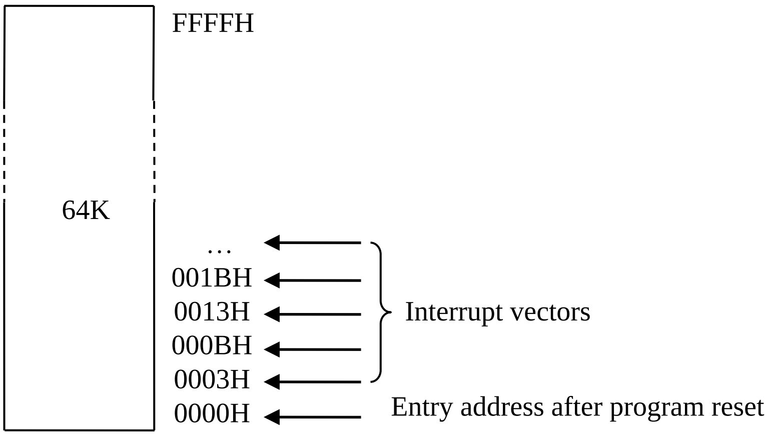
28K bytes of Flash program memory is integrated in STC8H1K28 family of microcontrollers.



12K bytes of Flash program memory is integrated in STC8H1K08 family of microcontrollers.



64K bytes of Flash program memory is integrated in STC8H3K64S4 family, STC8H3K64S2 family, STC8H8K64U family, STC8H2K64T family of microcontrollers.



After the microcontroller resets, the content of the Program Counter (PC) is 0000H, and the CPU begins to execute program from 0000H of Program Memory. The entry addresses of interrupt service routines, which are also called interrupt vectors, are also located in the program memory. Each interrupt has a fixed entry address in Program Memory. When an interrupt occurs and gets response, the microcontroller will automatically jump to its corresponding interrupt entry address to execute the service routine. The entry address of the interrupt service routine for the external interrupt 0 (INT0) is 0003H, the entry address for the timer / counter 0 (TIMER0) interrupt service routine is 000BH, and the entry address for the interrupt service routine for the external interrupt 1 (INT1) is 0013H. The counter/counter 1 (TIMER1) interrupt service routine's entry address is 001BH. More interrupt service routine entry address (interrupt vector), please refer to interrupt chapter.

The interval of adjacent interrupt entry addresses is only 8 bytes, which is not enough to save the complete interrupt service routine in general, so an unconditional jump instruction is stored in the the interrupt vector to jump to the space where the real interrupt service routine is stored, then execute interrupt service routine.

All STC8H series of microcontrollers integrate Flash data memory (EEPROM). The EEPROM is read or written in byte, and is erased in page of 512bytes. It can be repeatedly programmed and erased over 100,000 times, which improves the flexibility and convenience of use.

7.2 Data Memory

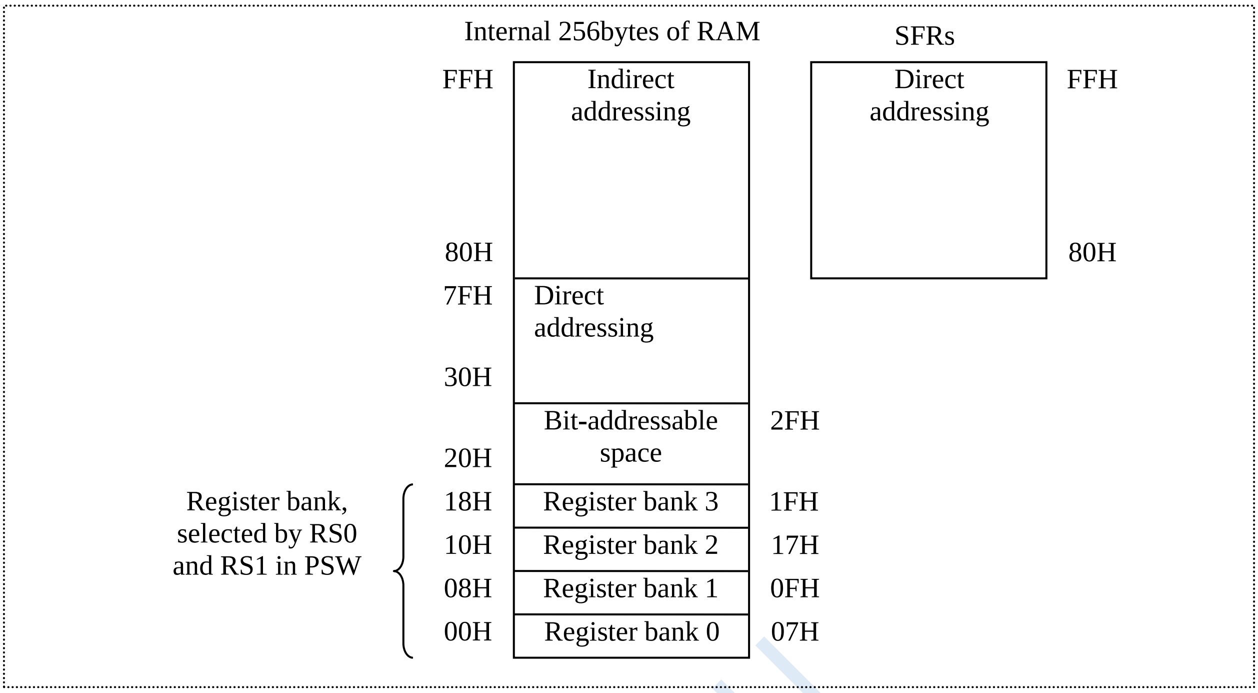
The RAM integrated in the STC8H series of microcontrollers can be used to store intermediate results and process data during program execution.

Family of microcontrollers	Internal direct access RAM (DATA)	Internal indirect access RAM (IDATA)	On-chip extended RAM (XDATA)
STC8H1K08 family	128 bytes	128 bytes	1024 bytes
STC8H1K28 family	128 bytes	128 bytes	1024 bytes
STC8H3K64S4 family	128 bytes	128 bytes	3072 bytes
STC8H3K64S2 family	128 bytes	128 bytes	3072 bytes
STC8H8K64U family	128 bytes	128 bytes	8192 bytes
STC8H2K64T family	128 bytes	128 bytes	2048 bytes
STC8H4K64TLR family	128 bytes	128 bytes	4096 bytes
STC8H4K64TLCD family	128 bytes	128 bytes	4096 bytes
STC8H4K64LCD family	128 bytes	128 bytes	4096 bytes

7.2.1 Internal RAM

A total of 256 bytes of internal RAM can be divided into two parts: Lower 128 bytes of RAM and Upper 128 bytes of RAM. The Lower 128 bytes of data memory are compatible with the traditional 8051 microcontroller, which can be access by either Direct addressing or Indirect addressing. The Upper 128 bytes of RAM (upper 128 bytes of RAM is extended in 8052) and special function registers, SFRs in short, occupy the same block of addresses, 80H to FFH, but they are physically separate entities and are accessed using different addressing modes. Upper 128 bytes of RAM can only be accessed by Indirect addressing, SFRs area can only be accessed by Direct addressing.

Internal RAM is mapped in the following figure.



The Lower 128 bytes of RAM are also called as general purpose RAM space. The general purpose RAM space can be divided into working register banks space, bit addressable space, user RAM space and stack space. Total of 32 bytes of working register bank space, 00H t0 1FH, are divided into 4 groups. Each group is called a register bank, which contains 8 8-bit working registers. All the numbers in different register bank are R0 through R7, but they belong to different Physical space. By using the working register registers, the operation speed can be increased. R0 ~ R7 are commonly used registers. Four bank sare provided because one bank is often not enough. The combination of RS1 and RS0 in the PSW register determines the working register bank currently used, see the introduction of PSW register below.

7.2.2 PSW (program status word register)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
PSW	D0H	CY	AC	F0	**RS1**	**RS0**	OV	F1	P

CY： Carry/borrow flag bit.

AC： Auxiliary carry/borrow flag bit.

F0: User flag bit 0。

RS1, RS0: Working register select bit

RS1	RS0	working register bank (R0~R7)
0	0	Bank 0 (00H~07H)
0	1	Bank 1 (08H~0FH)
1	0	Bank 2 (10H~17H)
1	1	Bank 3 (18H~1FH)

OV: Overflow flag bit。

F1: User flag bit 1。

P: Parity flag bit.

There are 16 bytes in the bit addressable space, 20H to 2FH. They can either be accessed by byte like ordinary RAM or be individually accessed by any one bit in the byte unit. There are totally 128 bits in this space, whose logic bit addresses are 00H to 7FH. From the appearance, bit addresses and the internal Lower 128 bytes RAM addresses are the same as 00H to 7FH, but in fact, they are essentially different: bit address points to a bit, and the byte address points to a byte unit. They are distinguished by using different instructions in programs.

The addresses 30H to FFH in the internal RAM are the user RAM and stack space. An 8-bit stack pointer, SP in short is used to point to the stack space. On reset, SP is 07H, which is R7 of register bank 0. Therefore, the initial value of SP should be set in the user 220nitialization codes. You would better to set the initial value of SP at 80H or higher.

SP is an 8-bit dedicated register. It indicates the top of the stack in the internal RAM. On reset, SP is initialized to 07H, which makes the stack space begin from 08H. The addresses 08H to 1FH are also the addresses of working register bank 1 through 3. It is better to change the SP value to a value of 80H or more if these spaces are used in user application. The stack of STC8H series of microcontrollers grows upward, which means that when a datum is pushed into the stack, the content of SP will increase.

7.2.3 On-chip extended RAM

In addition to 256 bytes of internal RAM, on-chip extended RAM is integrated in STC8H series of microcontrollers. The method of accessing the on-chip extended RAM is the same as that of the traditional 8051 MCU accessing the external extended RAM. However, the P0 port (data bus and low-order address bus), P2 port (high-order address bus), RD, WR and ALE are not affected.

In assembly language, the on-chip extended RAM is accessed through the MOVX instruction,

```
MOVX    A,@DPTR
MOVX    @DPTR,A
MOVX    A,@Ri
MOVX    @Ri,A
```

In C language, xdata / pdata can be used to declare the storage type, such as,

```
unsigned char xdata i;
unsigned int pdata j;
```

Note that pdata is the lower 256 bytes of xdata. After declaring a variable as the pdata type in C program, the compiler will automatically allocate the variables in the 0000H to 00FFH of XDATA and use MOVX @ Ri, A and MOVX A @ Ri to access.

The control bit EXTRAM located in AUXR register is used to control the access of on-chip extended RAM can be used or not.

7.2.4 Auxiliary register (AUXR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
AUXR	8EH	T0x12	T1x12	UART_M0x6	T2R	T2_C/T	T2x12	**EXTRAM**	S1ST2

EXTRAM: on-chip extended RAM access control bit

0: On-chip extended RAM is enabled or can be accessed.

1: On-chip extended RAM is disabled.

7.2.5 External extended RAM，XRAM，XDATA

The STC8H series of packages with a pin count of 40 and above have the ability to expand 64KB of external data memory. During access to the external data memory, the WR/RD/ALE signal must be valid. A new special function register BUS_SPEED for controlling the speed of the external 64K byte data bus has been added to the STC8H series of single-chip microcomputers. The description is as follows:

7.2.6 Bus speed control register（BUS_SPEED）

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
BUS_SPEED	A1H	RW_S[1:0]					SPEED[2:0]		

RW_S[1:0]：RD/WR control line selection bit

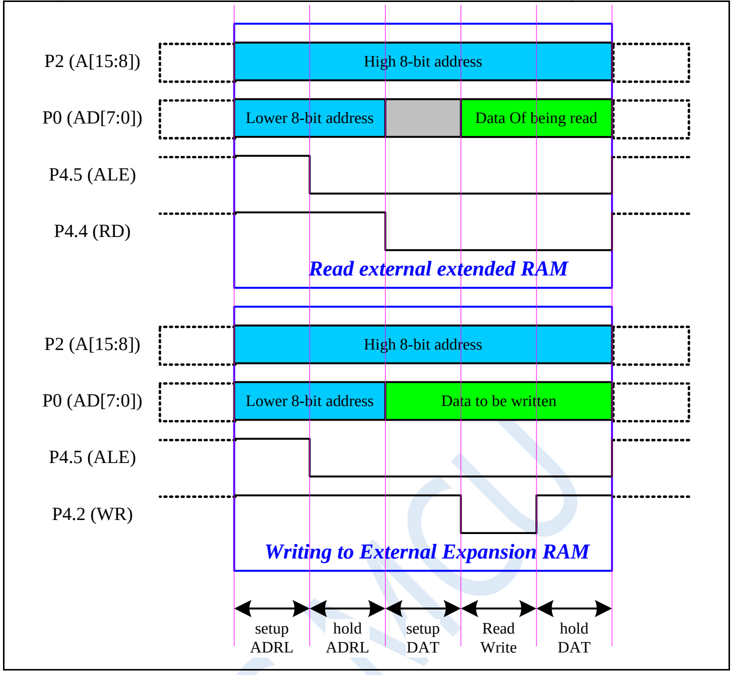
00：P4.4 is RD, P4.2 is WR

x1: Keep

SPEED[2:0]: Bus read and write speed control (preparation time and hold time of control signal and data signal when reading and writing data)

instruction	Number of clocks	
	Access internal expansion RAM	Access external expansion RAM
MOVX A,@Ri	3	3+5* (SPEED+1)
MOVX @Ri,A	3	3+5* (SPEED+1)
MOVX A,@DPTR	2	3+5* (SPEED+1)
MOVX @DPTR,A	2	3+5* (SPEED+1)

The timing of reading and writing external extended RAM is shown in the figure below:



7.2.7 Bit Addressable Data Memory in 8051

Bit addressable data memory integrated in 8051 single-chip includes two parts: the address range of the first part is 00H ~ 7FH, and the address range of the second part is 80H ~ FFH. The 00H ~ 7FH bit addressing area is a mapping of the 16 bytes of the data area 20H ~ 2FH, and the 80H ~ FFh bit addressing area is the 16 special function registers whose addresses are divisible by 8. (Including 80H, 88H, 90H, 98H, A0H, A8H, B0H, B8H, C0H, C8H, D0H, D8H, E0H, E8H, F0H, F8H).

Address of Data Memory	Address of Bit-addressable							
	B7	B6	B5	B4	B3	B2	B1	B0
F8H (P7)	FFH F8H.7	FEH F8H.6	FDH F8H.5	FCH F8H.4	FBH F8H.3	FAH F8H.2	F9H F8H.1	F8H F8H.0
F0H (B)	F7H F0H.7	F6H F0H.6	F5H F0H.5	F4H F0H.4	F3H F0H.3	F2H F0H.2	F1H F0H.1	F0H F0H.0
E8H (P6)	EFH E8H.7	EEH E8H.6	EDH E8H.5	ECH E8H.4	EBH E8H.3	EAH E8H.2	E9H E8H.1	E8H E8H.0
E0H (ACC)	E7H E0H.7	E6H E0H.6	E5H E0H.5	E4H E0H.4	E3H E0H.3	E2H E0H.2	E1H E0H.1	E0H E0H.0
D8H (CCON)	DFH D8H.7	DEH D8H.6	DDH D8H.5	DCH D8H.4	DBH D8H.3	DAH D8H.2	D9H D8H.1	D8H D8H.0
D0H (PSW)	D7H D0H.7	D6H D0H.6	D5H D0H.5	D4H D0H.4	D3H D0H.3	D2H D0H.2	D1H D0H.1	D0H D0H.0
C8H (P5)	CFH C8H.7	CEH C8H.6	CDH C8H.5	CCH C8H.4	CBH C8H.3	CAH C8H.2	C9H C8H.1	C8H C8H.0
C0H (P4)	C7H C0H.7	C6H C0H.6	C5H C0H.5	C4H C0H.4	C3H C0H.3	C2H C0H.2	C1H C0H.1	C0H C0H.0
B8H (IP)	BFH B8H.7	BEH B8H.6	BDH B8H.5	BCH B8H.4	BBH B8H.3	BAH B8H.2	B9H B8H.1	B8H B8H.0

B0H (P3)	B7H B0H.7	B6H B0H.6	B5H B0H.5	B4H B0H.4	B3H B0H.3	B2H B0H.2	B1H B0H.1	B0H B0H.0
A8H (IE)	AFH A8H.7	AEH A8H.6	ADH A8H.5	ACH A8H.4	ABH A8H.3	AAH A8H.2	A9H A8H.1	A8H A8H.0
A0H (P2)	A7H A0H.7	A6H A0H.6	A5H A0H.5	A4H A0H.4	A3H A0H.3	A2H A0H.2	A1H A0H.1	A0H A0H.0
98H (SCON)	9FH 98H.7	9EH 98H.6	9DH 98H.5	9CH 98H.4	9BH 98H.3	9AH 98H.2	99H 98H.1	98H 98H.0
90H (P1)	97H 90H.7	96H 90H.6	95H 90H.5	94H 90H.4	93H 90H.3	92H 90H.2	91H 90H.1	90H 90H.0
88H (TCON)	8FH 88H.7	8EH 88H.6	8DH 88H.5	8CH 88H.4	8BH 88H.3	8AH 88H.2	89H 88H.1	88H 88H.0
80H (P0)	87H 80H.7	86H 80H.6	85H 80H.5	84H 80H.4	83H 80H.3	82H 80H.2	81H 80H.1	80H 80H.0
2FH	7FH 2FH.7	7EH 2FH.6	7DH 2FH.5	7CH 2FH.4	7BH 2FH.3	7AH 2FH.2	79H 2FH.1	78H 2FH.0
2EH	77H 2EH.7	76H 2EH.6	75H 2EH.5	74H 2EH.4	73H 2EH.3	72H 2EH.2	71H 2EH.1	70H 2EH.0
2DH	6FH 2DH.7	6EH 2DH.6	6DH 2DH.5	6CH 2DH.4	6BH 2DH.3	6AH 2DH.2	69H 2DH.1	68H 2DH.0
2CH	67H 2CH.7	66H 2CH.6	65H 2CH.5	64H 2CH.4	63H 2CH.3	62H 2CH.2	61H 2CH.1	60H 2CH.0
2BH	5FH 2BH.7	5EH 2BH.6	5DH 2BH.5	5CH 2BH.4	5BH 2BH.3	5AH 2BH.2	59H 2BH.1	58H 2BH.0
2AH	57H 2AH.7	56H 2AH.6	55H 2AH.5	54H 2AH.4	53H 2AH.3	52H 2AH.2	51H 2AH.1	50H 2AH.0
29H	4FH 29H.7	4EH 29H.6	4DH 29H.5	4CH 29H.4	4BH 29H.3	4AH 29H.2	49H 29H.1	48H 29H.0
28H	47H 28H.7	46H 28H.6	45H 28H.5	44H 28H.4	43H 28H.3	42H 28H.2	41H 28H.1	40H 28H.0
27H	3FH 27H.7	3EH 27H.6	3DH 27H.5	3CH 27H.4	3BH 27H.3	3AH 27H.2	39H 27H.1	38H 27H.0
26H	37H 26H.7	36H 26H.6	35H 26H.5	34H 26H.4	33H 26H.3	32H 26H.2	31H 26H.1	30H 26H.0
25H	2FH 25H.7	2EH 25H.6	2DH 25H.5	2CH 25H.4	2BH 25H.3	2AH 25H.2	29H 25H.1	28H 25H.0
24H	27H 24H.7	26H 24H.6	25H 24H.5	24H 24H.4	23H 24H.3	22H 24H.2	21H 24H.1	20H 24H.0
23H	1FH 23H.7	1EH 23H.6	1DH 23H.5	1CH 23H.4	1BH 23H.3	1AH 23H.2	19H 23H.1	18H 23H.0
22H	17H 22H.7	16H 22H.6	15H 22H.5	14H 22H.4	13H 22H.3	12H 22H.2	11H 22H.1	10H 22H.0
21H	0FH 21H.7	0EH 21H.6	0DH 21H.5	0CH 21H.4	0BH 21H.3	0AH 21H.2	09H 21H.1	08H 21H.0
20H	07H 20H.7	06H 20H.6	05H 20H.5	04H 20H.4	03H 20H.3	02H 20H.2	01H 20H.1	00H 20H.0

7.3 Special parameters of memory

The data memory and program memory of the STC8H series of microcontrollers store some special parameters related to the chip, including the global unique ID, the frequency of the 32K power-down wake-up timer, the internal reference voltage value, and the IRC parameters.

Note: For chips with firmware version 7.4.4 and later, all special parameters can only be read from the read-only special function register (CHIPID).

The addresses of these parameters in the Flash program memory (ROM) are as follows:

Parameters	Addresses			Parameter Description
	STC8H1K08	STC8H1K12	STC8H1K17	
global unique ID	1FF9H~1FFFH	2FF9H~2FFFH	43F9H~43FFH	7 bytes
internal 1.19V reference voltage	1FF7H~1FF8H	2FF7H~2FF8H	43F7H~43F8H	mV (high byte first)
frequency of the 32K power-down wake-up timer	1FF5H~1FF6H	2FF5H~2FF6H	43F5H~43F6H	Hz (high byte first)
parameters of 22.1184MHz IRC(20M band)	1FF4H	2FF4H	43F4H	-
parameters of 24MHz IRC(20M band)	1FF3H	2FF3H	43F3H	-
parameters of 20MHz IRC(20M band)	1FF2H	2FF2H	43F2H	Valid for the version 7.3.12U or later firmware.
parameters of 27MHz IRC(35M band)	1FF1H	2FF1H	43F1H	
parameters of 30MHz IRC(35M band)	1FF0H	2FF0H	43F0H	
parameters of 33.1776MHz IRC(35M band)	1FEFH	2FEFH	43EFH	
parameters of 35MHz IRC(35M band)	1FEEH	2FEEH	43EEH	
parameters of 36.684MHz IRC(35M band)	1FEDH	2FEDH	43EDH	
Reserved	1FECH	2FECH	43ECH	
Reserved	1FEBH	2FEBH	43EBH	
parameters of VRTRIM in 20M band	1FEAH	2FEAH	43EAH	
parameters of VRTRIM in 35M band	1FE9H	2FE9H	43E9H	

Parameters	Addresses				Parameter Description
	STC8H1K16	STC8H1K24	STC8H1K28	STC8H1K33	
global unique ID	3FF9H~3FFFH	5FF9H~5FFFH	6FF9H~6FFFH	83F9H~83FFH	7 bytes
internal 1.19V reference voltage	3FF7H~3FF8H	5FF7H~5FF8H	6FF7H~6FF8H	83F7H~83F8H	mV (high byte first)
frequency of the 32K power-down wake-up timer	3FF5H~3FF6H	5FF5H~5FF6H	6FF5H~6FF6H	83F5H~83F6H	Hz (high byte first)
parameters of 22.1184MHz IRC(20M band)	3FF4H	5FF4H	6FF4H	83F4H	-
parameters of 24MHz IRC(20M band)	3FF3H	5FF3H	6FF3H	83F3H	-
parameters of 20MHz IRC(20M band)	3FF2H	5FF2H	6FF2H	83F2H	Valid for the version 7.3.12U or later firmware.
parameters of 27MHz IRC(35M band)	3FF1H	5FF1H	6FF1H	83F1H	
parameters of 30MHz IRC(35M band)	3FF0H	5FF0H	6FF0H	83F0H	
parameters of 33.1776MHz IRC(35M band)	3FEFH	5FEFH	6FEFH	83EFH	
parameters of 35MHz IRC(35M band)	3FEEH	5FEEH	6FEEH	83EEH	
parameters of 36.684MHz IRC(35M band)	3FEDH	5FEDH	6FEDH	83EDH	
Reserved	3FECH	5FECH	6FECH	83ECH	
Reserved	3FEBH	5FEBH	6FEBH	83EBH	
parameters of VRTRIM in 20M band	3FEAH	5FEAH	6FEAH	83EAH	
parameters of VRTRIM in 35M band	3FE9H	5FE9H	6FE9H	83E9H	

Parameters	Addresses				Parameter Description
	STC8H3K32S4 STC8H3K32S2 STC8H2K32T STC8H4K32TLR STC8H4K32TLCD STC8H4K32LCD	STC8H3K48S4 STC8H3K48S2 STC8H2K48T STC8H4K48TLR STC8H4K48TLCD STC8H4K48LCD	STC8H3K60S4 STC8H3K60S2 STC8H2K60T STC8H4K60TLR STC8H4K60TLCD STC8H4K60LCD	STC8H3K64S4 STC8H3K64S2 STC8H2K64T STC8H4K64TLR STC8H4K64TLCD STC8H4K64LCD	
global unique ID	7FF9H~7FFFH	BFF9H~BFFFH	EFF9H~EFFFH	FDF9H~FDFFH	7 bytes
internal 1.19V reference voltage	7FF7H~7FF8H	BFF7H~BFF8H	EFF7H~EFF8H	FDF7H~FDF8H	mV (high byte first)
frequency of the 32K power-down wake-up timer	7FF5H~7FF6H	BFF5H~BFF6H	EFF5H~EFF6H	FDF5H~FDF6H	Hz (high byte first)
parameters of 22.1184MHz IRC(20M band)	7FF4H	BFF4H	EFF4H	FDF4H	-
parameters of 24MHz IRC(20M band)	7FF3H	BFF3H	EFF3H	FDF3H	-
parameters of 20MHz IRC(20M band)	7FF2H	BFF2H	EFF2H	FDF2H	Valid for the version 7.3.12U or later firmware.
parameters of 27MHz IRC(35M band)	7FF1H	BFF1H	EFF1H	FDF1H	
parameters of 30MHz IRC(35M band)	7FF0H	BFF0H	EFF0H	FDF0H	
parameters of 33.1776MHz IRC(35M band)	7FEFH	BFEFH	EFEFH	FDEFH	
parameters of 35MHz IRC(35M band)	7FEEH	BFEEH	EFEEH	FDEEH	
parameters of 36.684MHz IRC(35M band)	7FEDH	BFEDH	EFEDH	FDEDH	
Reserved	7FECH	BFECH	EFECH	FDECH	
Reserved	7FEBH	BFEBH	EFEBH	FDEBH	

parameters of VRTRIM in 20M band	7FEAH	BFEAH	EFEAH	FDEAH	
parameters of VRTRIM in 35M band	7FE9H	BFE9H	EFE9H	FDE9H	

Parameters	Addresses				Parameter Description
	STC8H8K32U	STC8H8K48U	STC8H8K60U	STC8H8K64U	
global unique ID	7FF9H~7FFFH	BFF9H~BFFFH	EFF9H~EFFFH	FDF9H~FDFFH	7 bytes
internal 1.19V reference voltage	7FF7H~7FF8H	BFF7H~BFF8H	EFF7H~EFF8H	FDF7H~FDF8H	mV (high byte first)
frequency of the 32K power-down wake-up timer	7FF5H~7FF6H	BFF5H~BFF6H	EFF5H~EFF6H	FDF5H~FDF6H	Hz (high byte first)
parameters of 22.1184MHz IRC(20M band)	7FF4H	BFF4H	EFF4H	FDF4H	-
parameters of 24MHz IRC(20M band)	7FF3H	BFF3H	EFF3H	FDF3H	-
parameters of 27MHz IRC(35M band)	7FF2H	BFF2H	EFF2H	FDF2H	Valid for the version 7.3.12U or later firmware.
parameters of 30MHz IRC(40M band)	7FF1H	BFF1H	EFF1H	FDF1H	
parameters of 33.1776MHz IRC(40M band)	7FF0H	BFF0H	EFF0H	FDF0H	
parameters of 35MHz IRC(40M band)	7FEFH	BFEFH	EFEFH	FDEFH	
parameters of 36.684MHz IRC(40M band)	7FEEH	BFEEH	EFEEH	FDEEH	
parameters of 40MHz IRC(40M band)	7FEDH	BFEDH	EFEDH	FDEDH	
parameters of 44.2368MHz IRC(40M band)	7FECH	BFECH	EFECH	FDECH	
parameters of 48MHz IRC(40M band)	7FEBH	BFEBH	EFEBH	FDEBH	
parameters of VRTRIM in 20M band	7FEAH	BFEAH	EFEAH	FDEAH	
parameters of VRTRIM in 40M band	7FE9H	BFE9H	EFE9H	FDE9H	

Note: The extended IRC parameter list of STC8H8K64U series is not compatible with other series.

Note: For chips with firmware version 7.4.4 and later, all special parameters can only be read from the read-only special function register (CHIPID).

The addresses of these parameters in the data memory (RAM) are as follows:

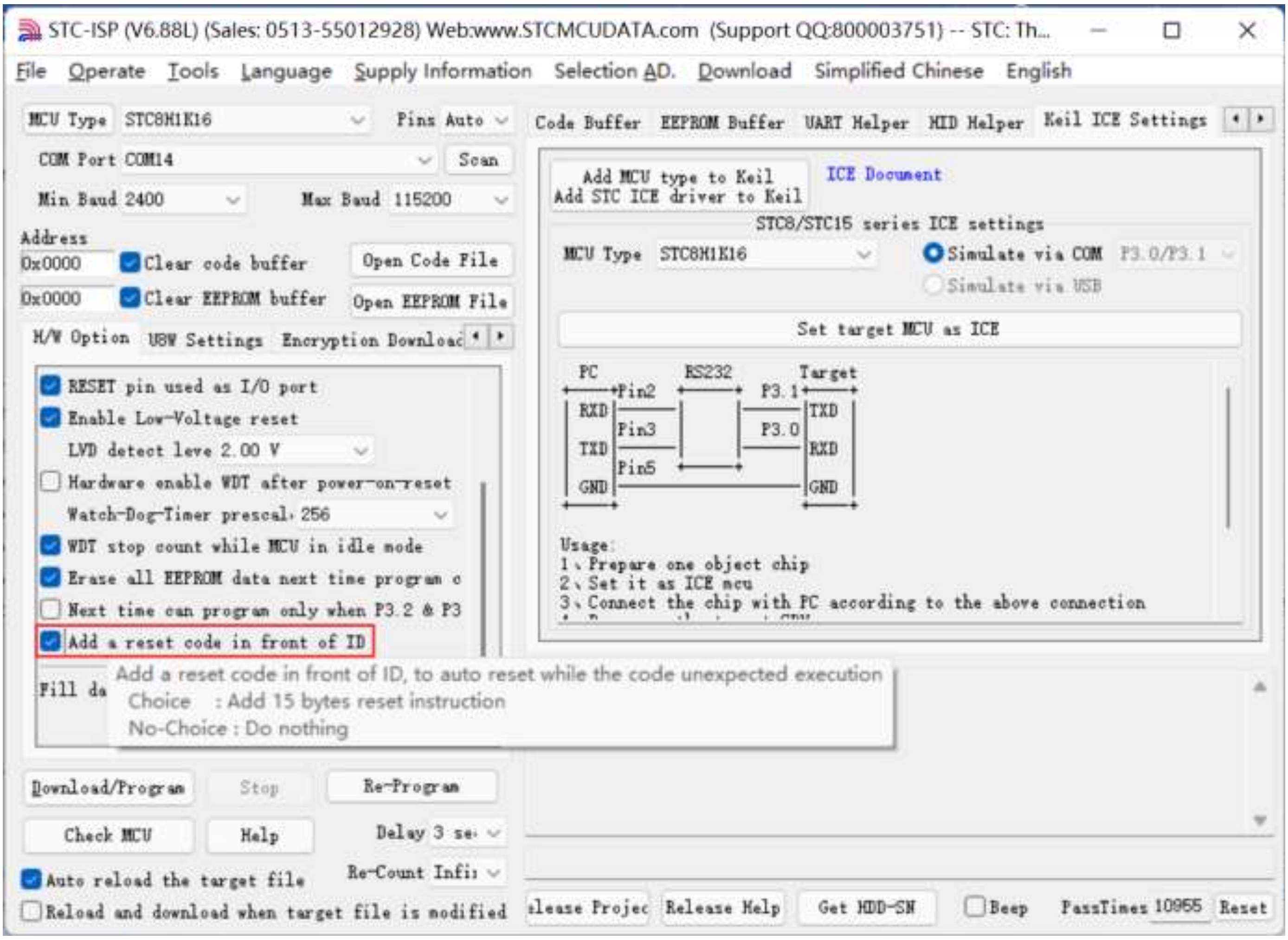
Parameters	Addresses	**Parameter Description**
global unique ID	idata: 0F1H~0F7H	7 bytes
internal reference voltage	idata: 0EFH~0F0H	mV (high byte first)
frequency of the 32K power-down wake-up timer	idata: 0F8H~0F9H	Hz (high byte first)
parameters of 22.1184MHz IRC	idata: 0FAH	-
parameters of 24MHz IRC	idata: 0FBH	-

Special Note

1. Since the parameters in RAM may be modified, it is generally not recommended to be used. Especially, it is strongly recommended to read ID data in FLASH program memory (ROM) when you use ID for encryption.

2. Due to the size of EEPROM in STC8H1K28, STC8H1K12, STC8H3K64S4, STC8H3K64S2, STC8H8K64U, STC8H2K64T, STC8H4K64TLR, STC8H4K64TLCD, STC8H4K64LCD can be set by user, important parameters stored in the FLASH program memory (ROM) space may be erased or modified when the FLASH is used as EEPROM. So this issue needs to be considered when using these microcontrollers for ID number encryption.

3. By default, only the global unique ID is in the Flash program memory (ROM), and the internal reference 1.19 voltage value, the frequency of the 32K power-down wake-up timer, and the IRC parameters are not available. You need to select the option in the following figure when downloading by ISP, and then the options shown are available.



STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...

MCU Type STC8H1K16 Pins Auto
COM Port COM14
Min Baud 2400 Max Baud 115200
Address
0x0000 ☑ Clear code buffer
0x0000 ☑ Clear EEPROM buffer

☑ RESET pin used as I/O port
☑ Enable Low-Voltage reset
LVD detect leve 2.00 V
□ Hardware enable WDT after power-on-reset
Watch-Dog-Timer prescal: 256
☑ WDT stop count while MCU in idle mode
☑ Erase all EEPROM data next time program c
□ Next time can program only when P3.2 & P3
☑ Add a reset code in front of ID
Fill da

Add a reset code in front of ID, to auto reset while the code unexpected execution
Choice : Add 15 bytes reset instruction
No-Choice : Do nothing

Delay 3 se
Re-Count Infi
☑ Auto reload the target file
□ Reload and download when target file is modified

STC8/STC15 series ICE settings
MCU Type STC8H1K16
● Simulate via COM P3.0/P3.1
○ Simulate via USB

Usage:
1、Prepare one object chip
2、Set it as ICE mcu
3、Connect the chip with PC according to the above connection

□ Beep PassTimes 10955

7.4 Unique ID number and important parameter (CHIPID) stored in read-only special function register

Note: For chips with firmware version 7.4.4 and later, all special parameters can only be read from the read-only special function register (CHIPID).

Product line	CHIPID
STC8H1K08 family	
STC8H1K28 family	
STC8H3K64S4 family	
STC8H3K64S2 family	
STC8H8K64U family version A	
STC8H8K64U family version B	●
STC8H2K64T family	
STC8H4K64TLR family	●
STC8H4K64TLCD family	●
STC8H4K64LCD family	●

Some STC8H series microcontrollers have built-in 32-byte read-only special function register CHIPID. The content in CHIPID can only be read by the user program and cannot be modified. Using the data in CHIPID to encrypt user programs is the optimal solution officially recommended by STC.

Related registers

Symbol	Description	Address	Bit address and symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
CHIPID00	hardware digital ID00	FDE0H	Globally Unique ID Number (0th byte)								nnnn,nnnn
CHIPID01	hardware digital ID01	FDE1H	Globally Unique ID Number (1st byte)								nnnn,nnnn
CHIPID02	hardware digital ID02	FDE2H	Globally Unique ID Number (2nd byte)								nnnn,nnnn
CHIPID03	hardware digital ID03	FDE3H	Globally Unique ID Number (3rd byte)								nnnn,nnnn
CHIPID04	hardware digital ID04	FDE4H	Globally Unique ID Number (4th byte)								nnnn,nnnn
CHIPID05	hardware digital ID05	FDE5H	Globally Unique ID Number (5th byte)								nnnn,nnnn
CHIPID06	hardware digital ID06	FDE6H	Globally Unique ID Number (6th byte)								nnnn,nnnn
CHIPID07	hardware digital ID07	FDE7H	Internal 1.19V reference signal source (high byte)								nnnn,nnnn
CHIPID08	hardware digital ID08	FDE8H	Internal 1.19V reference signal source (low byte)								nnnn,nnnn
CHIPID09	hardware digital ID09	FDE9H	32K Power-down wake-up timer frequency(high byte)								nnnn,nnnn
CHIPID10	hardware digital ID10	FDEAH	32K Power-down wake-up timer frequency (low byte)								nnnn,nnnn
CHIPID11	hardware digital ID11	FDEBH	**USB series (STC8H8K64U family)**				**Other series**				nnnn,nnnn
			IRC parameter of 22.1184MHz (27M band)				IRC parameter of 22.1184MHz (27M band)				
CHIPID12	hardware digital ID12	FDECH	IRC parameter of 24MHz (27M band)				IRC parameter of 24MHz (27M band)				nnnn,nnnn
CHIPID13	hardware digital ID13	FDEDH	IRC parameter of 27MHz (27M band)				IRC parameter of 20MHz (27M band)				nnnn,nnnn
CHIPID14	hardware digital ID14	FDEEH	IRC parameter of 30MHz (27M band)				IRC parameter of 27MHz (27M band)				nnnn,nnnn
CHIPID15	hardware digital ID15	FDEFH	IRC parameter of 33.1776MHz (27M band)				IRC parameter of 30MHz (27M band)				nnnn,nnnn
CHIPID16	hardware digital ID16	FDF0H	IRC parameter of 35MHz (44M band)				IRC parameter of 33.1776MHz (27M band)				nnnn,nnnn
CHIPID17	hardware digital ID17	FDF1H	IRC parameter of 36.864MHz (44M band)				IRC parameter of 35MHz (44M band)				nnnn,nnnn
CHIPID18	hardware digital ID18	FDF2H	IRC parameter of 40MHz (44M band)				IRC parameter of 36.864MHz(44M band)				nnnn,nnnn
CHIPID19	hardware digital ID19	FDF3H	IRC parameter of 44.2368MHz (44M band)				IRC parameter of 40MHz(44M band)				nnnn,nnnn
CHIPID20	hardware digital ID20	FDF4H	IRC parameter of 48MHz (44M band)				IRC parameter of 45MHz(44M band)				nnnn,nnnn
CHIPID21	hardware digital ID21	FDF5H	VRTRIM parameter of 6M band								nnnn,nnnn
CHIPID22	hardware digital ID22	FDF6H	VRTRIM parameter of 10M band								nnnn,nnnn
CHIPID23	hardware digital ID23	FDF7H	VRTRIM parameter of 27M band								nnnn,nnnn
CHIPID24	hardware digital ID24	FDF8H	VRTRIM parameter of 44M band								nnnn,nnnn
CHIPID25	hardware digital ID25	FDF9H	00H								nnnn,nnnn
CHIPID26	hardware digital ID26	FDFAH	User program space end address (high byte)								nnnn,nnnn
CHIPID27	hardware digital ID27	FDFBH	Chip test time (year)								nnnn,nnnn
CHIPID28	hardware digital ID28	FDFCH	Chip test time (month)								nnnn,nnnn
CHIPID29	hardware digital ID29	FDFDH	Chip test time (day)								nnnn,nnnn
CHIPID30	hardware digital ID30	FDFEH	Chip package form number								nnnn,nnnn
CHIPID31	hardware digital ID31	FDFFH	5AH								nnnn,nnnn

7.4.1 Interpretation of Global Unique ID Number in CHIP

Symbol	Description	Address	Bit address and symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
CHIPID00	hardware digital ID00	FDE0H	Globally Unique ID Number (0th byte)								nnnn,nnnn
CHIPID01	hardware digital ID01	FDE1H	Globally Unique ID Number (1st byte)								nnnn,nnnn
CHIPID02	hardware digital ID02	FDE2H	Globally Unique ID Number (2nd byte)								nnnn,nnnn
CHIPID03	hardware digital ID03	FDE3H	Globally Unique ID Number (3rd byte)								nnnn,nnnn
CHIPID04	hardware digital ID04	FDE4H	Globally Unique ID Number (4th byte)								nnnn,nnnn
CHIPID05	hardware digital ID05	FDE5H	Globally Unique ID Number (5th byte)								nnnn,nnnn
CHIPID06	hardware digital ID06	FDE6H	Globally Unique ID Number (6th byte)								nnnn,nnnn

[CHIPID0, CHIPID1]: 16-bit MCU ID, which is used to distinguish different MCU models (higher bit first).

The commonly used MCU IDs of STC8 series are shown in the following table:

STC8H4K16TLCD (F811)	STC8H4K16LCD (F821)	STC8H4K16TLR (F7C9)	STC8H3K16S4 (F741)	STC8H3K16S2 (F749)	STC8H8K16U (F781)
STC8H4K32TLCD (F812)	STC8H4K32LCD (F822)	STC8H4K32TLR (F7CA)	STC8H3K32S4 (F742)	STC8H3K32S2 (F74A)	STC8H8K32U (F782)
STC8H4K48TLCD (F815)	STC8H4K48LCD (F825)	STC8H4K48TLR (F7CD)	STC8H3K48S4 (F745)	STC8H3K48S2 (F74D)	STC8H8K48U (F785)
STC8H4K60TLCD (F813)	STC8H4K60LCD (F823)	STC8H4K60TLR (F7CB)	STC8H3K60S4 (F743)	STC8H3K60S2 (F74B)	STC8H8K60U (F783)
STC8H4K64TLCD (F814)	STC8H4K64LCD (F824)	STC8H4K64TLR (F7CC)	STC8H3K64S4 (F744)	STC8H3K64S2 (F74C)	STC8H8K64U (F784)
STC8H1K06 (F733)	STC8H1K16 (F721)	STC8G2K16S4 (F761)	STC8G2K16S2 (F769)	STC8G1K06A-8PIN (F793)	STC8G1K06-8PIN (F7A3)
STC8H1K08 (F734)	STC8H1K20 (F722)	STC8G2K32S4 (F762)	STC8G2K32S2 (F76A)	STC8G1K08A-8PIN (F794)	STC8G1K08-8PIN (F7A4)
STC8H1K10 (F735)	STC8H1K24 (F723)	STC8G2K48S4 (F765)	STC8G2K48S2 (F76D)	STC8G1K10A-8PIN (F795)	STC8G1K10-8PIN (F7A5)
STC8H1K12 (F736)	STC8H1K28 (F724)	STC8G2K60S4 (F763)	STC8G2K60S2 (F76B)	STC8G1K12A-8PIN (F796)	STC8G1K12-8PIN (F7A6)
STC8H1K17 (F737)	STC8H1K33 (F725)	STC8G2K64S4 (F764)	STC8G2K64S2 (F76C)	STC8G1K17A-8PIN (F797)	STC8G1K17-8PIN (F7A7)
STC8G1K06 (F753)	STC8C2K16S4 (F7D1)	STC8C2K16S2 (F7D9)	STC8A8K16D4 (F7F1)		
STC8G1K08 (F754)	STC8C2K32S4 (F7D2)	STC8C2K32S2 (F7DA)	STC8A8K32D4 (F7F2)		
STC8G1K10 (F755)	STC8C2K48S4 (F7D5)	STC8C2K48S2 (F7DD)	STC8A8K48D4 (F7F5)		
STC8G1K12 (F756)	STC8C2K60S4 (F7D3)	STC8C2K60S2 (F7DB)	STC8A8K60D4 (F7F3)		
STC8G1K17 (F757)	STC8C2K64S4 (F7D4)	STC8C2K64S2 (F7DC)	STC8A8K64D4 (F7F4)		

[CHIPID2, CHIPID3]: 16-bit test machine number (higher bit first).

[CHIPID4, CHIPID5, CHIPID6]: 24-bit test serial number (higher bit first).

7.4.2 Interpretation of the internal reference signal source in CHIP

Symbol	Description	Address	Bit address and symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
CHIPID07	hardware digital ID07	FDE7H	Internal 1.19V reference signal source (high byte)								nnnn,nnnn
CHIPID08	hardware digital ID08	FDE8H	Internal 1.19V reference signal source (low byte)								nnnn,nnnn

[CHIPID7, CHIPID8]: 16-bit internal reference signal source voltage value (higher bit first).

The standard value is 1190 (04A6H), and the unit is mV, which is 1.19V. But there is error in the actual chip due to manufacturing errors. The voltage value of the internal reference signal source is not affected by the working voltage VCC, so the internal reference signal source can be combined with the ADC to calibrate the ADC, and can also be combined with the comparator to detect the operating voltage.

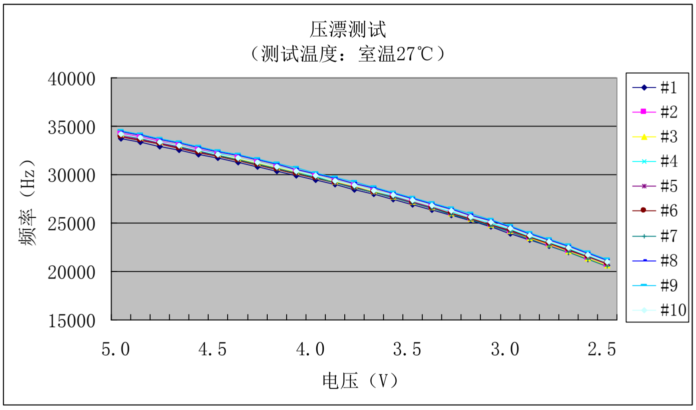
7.4.3 Interpretation of internal 32K IRC oscillation frequency in CHIP

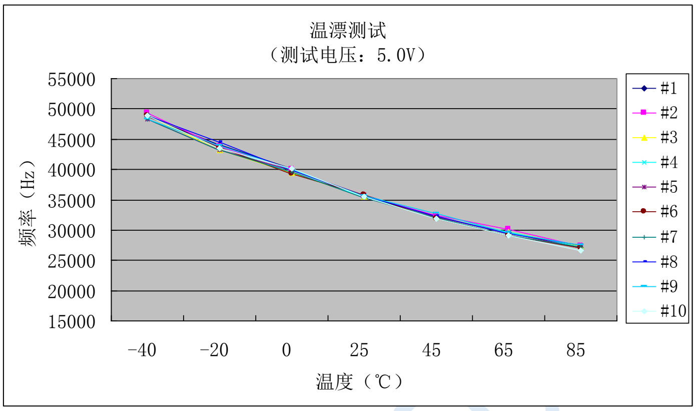
Symbol	Description	Address	Bit address and symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
CHIPID09	hardware digital ID09	FDE9H	32K Power-down wake-up timer frequency(high byte)								nnnn,nnnn
CHIPID10	hardware digital ID10	FDEAH	32K Power-down wake-up timer frequency (low byte)								nnnn,nnnn

[CHIPID9, CHIPID10]: 16-bit 32K IRC oscillator frequency value (higher bit first).

The standard value is 32768 (8000H), the unit is Hz, that is, 32.768KHz. However, the actual chip has manufacturing errors, and the temperature drift and pressure drift are relatively large.

The voltage drift test linear diagram and temperature drift linear diagram of the internal 32K oscillator are as follows:





7.4.4 Interpretation of high precision IRC parameters in CHIP

Symbol	Description	Address	Bit address and symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
CHIPID11	hardware digital ID11	FDEBH	**USB series (STC8H8K64U family)** IRC parameter of 22.1184MHz (27M band)				**Other series** IRC parameter of 22.1184MHz (27M band)				nnnn,nnnn
CHIPID12	hardware digital ID12	FDECH	IRC parameter of 24MHz (27M band)				IRC parameter of 24MHz (27M band)				nnnn,nnnn
CHIPID13	hardware digital ID13	FDEDH	IRC parameter of 27MHz (27M band)				IRC parameter of 20MHz (27M band)				nnnn,nnnn
CHIPID14	hardware digital ID14	FDEEH	IRC parameter of 30MHz (27M band)				IRC parameter of 27MHz (27M band)				nnnn,nnnn
CHIPID15	hardware digital ID15	FDEFH	IRC parameter of 33.1776MHz (27M band)				IRC parameter of 30MHz (27M band)				nnnn,nnnn
CHIPID16	hardware digital ID16	FDF0H	IRC parameter of 35MHz (44M band)				IRC parameter of 33.1776MHz (27M band)				nnnn,nnnn
CHIPID17	hardware digital ID17	FDF1H	IRC parameter of 36.864MHz (44M band)				IRC parameter of 35MHz (44M band)				nnnn,nnnn
CHIPID18	hardware digital ID18	FDF2H	IRC parameter of 40MHz (44M band)				IRC parameter of 36.864MHz(44M band)				nnnn,nnnn
CHIPID19	hardware digital ID19	FDF3H	IRC parameter of 44.2368MHz (44M band)				IRC parameter of 40MHz(44M band)				nnnn,nnnn
CHIPID20	hardware digital ID20	FDF4H	IRC parameter of 48MHz (44M band)				IRC parameter of 45MHz(44M band)				nnnn,nnnn
CHIPID21	hardware digital ID21	FDF5H	VRTRIM parameter of 6M band								
CHIPID22	hardware digital ID22	FDF6H	VRTRIM parameter of 10M band								
CHIPID23	hardware digital ID23	FDF7H	VRTRIM parameter of 27M band								
CHIPID24	hardware digital ID24	FDF8H	VRTRIM parameter of 44M band								

In the STC8H series MCUs that support the CHIPID function, the integrated high-precision IRC is divided into 4 frequency bands, and the reference voltage value corresponding to every frequency band has been calibrated at the factory. When selecting different frequency bands, you only need to fill the calibrated voltage value of the corresponding frequency band in the VRTRIM register. The center frequencies of the 4 frequency bands are 6MHz, 10MHz, 27MHz and 44MHz respectively. Due to manufacturing errors, the center frequency may generally have a deviation of ±5%. In order to obtain accurate user frequencies, IRTRIM can be used to fine-tune the frequency. When using the download software provided by STC to download the user program, the system will automatically set the VRTRIM and IRTRIM registers according to the frequency set by the user. At the same time, the IRTRIM value of 10 common frequencies and the calibration value of the reference voltage value of 4 frequency bands are preset in CHIPID, so that the user can dynamically modify the working frequency during the running of the program.

[CHIPID11 : CHIPID20]: the IRTRIM value of 10 common frequencies. The annotations in parentheses are the corresponding frequency bands.

[CHIPID21 : CHIPID24]: the calibration value of the reference voltage value of 4 frequency bands.

When the user modifies the frequency dynamically, he only needs to read out a certain frequency calibration value in [CHIPID11 : CHIPID20] and write it into the IRTRIM register, and at the same time, according to the frequency band corresponding to the frequency, set a certain voltage calibration value in [CHIPID21 : CHIPID24] Just read and write to the VRTRIM register. For detailed operations, please refer to the sample programs in the following chapters.

7.4.5 Interpretation of test time parameters in CHIP

Symbol	Description	Address	Bit address and symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
CHIPID27	hardware digital ID27	FDFBH	Chip test time (year)								nnnn,nnnn
CHIPID28	hardware digital ID28	FDFCH	Chip test time (month)								nnnn,nnnn
CHIPID29	hardware digital ID29	FDFDH	Chip test time (day)								nnnn,nnnn

The year, month, and day parameters of the test time are all BCD codes. For example, CHIPID27=0x21, CHIPID28=0x11, CHIPID29=0x18, then the production test date of the target chip is November 18, 2021.

7.4.6 Interpretation of chip package form number in CHIP

Symbol	Description	Address	Bit address and symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
CHIPID30	hardware digital ID30	FDFEH	Chip package form number								nnnn,nnnn

Chip package form number	Package		Chip package form number	Package
0x00	DIP8		0x50	SOP32
0x01	SOP8		0x51	LQFP32
0x02	DFN8		0x52	QFN32
0x10	DIP16		0x53	PLCC32
0x11	SOP16		0x54	QFN32S
0x20	DIP18		0x60	PDIP40
0x21	SOP18		0x70	LQFP44
0x30	DIP20		0x71	PLCC44
0x31	SOP20		0x72	PQFP44
0x32	TSSOP20		0x80	LQFP48
0x33	LSSOP20		0x81	QFN48
0x34	QFN20		0x90	LQFP64
0x40	SKDIP28		0x91	LQFP64S
0x41	SOP28		0x92	LQFP64L
0x42	TSSOP28		0x93	LQFP64M
0x43	QFN28		0x94	QFN64

7.5 Example Routines

7.5.1 Read Internal Referrence Voltage Value (Read from CHIPID)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (65536 - FOSC / 115200 / 4)

#define     USBCHIPID
#define     CPUIDBASE           0xfde0
#define     VREF_ADDR           (*(unsigned int volatile xdata *)(CPUIDBASE + 0x07))

sfr         AUXR            =   0x8e;
sfr         P_SW2           =   0xba;

sfr         P0M1            =   0x93;
sfr         P0M0            =   0x94;
sfr         P1M1            =   0x91;
sfr         P1M0            =   0x92;
sfr         P2M1            =   0x95;
sfr         P2M0            =   0x96;
sfr         P3M1            =   0xb1;
sfr         P3M0            =   0xb2;
sfr         P4M1            =   0xb3;
sfr         P4M0            =   0xb4;
sfr         P5M1            =   0xc9;
sfr         P5M0            =   0xca;

bit         busy;

void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
    }
}

void UartInit()
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
```

```
        TR1 = 1;
        AUXR = 0x40;
        busy = 0;
}

void UartSend(char dat)
{
        while (busy);
        busy = 1;
        SBUF = dat;
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        UartInit();
        ES = 1;
        EA = 1;
        P_SW2 = 0x80;
        UartSend(VREF_ADDR >> 8);                    // Read the high byte of the internal 1.19V reference signal source
        UartSend(VREF_ADDR);                         // Read the low byte of the internal 1.19V reference signal source

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

CPUIDBASE    EQU    0FDE0H
VREF_ADDR    EQU    CPUIDBASE + 07H

AUXR         DATA          8EH
P_SW2        DATA          0BAH

BUSY         BIT           20H.0

P0M1         DATA          093H
P0M0         DATA          094H
P1M1         DATA          091H
P1M0         DATA          092H
P2M1         DATA          095H
P2M0         DATA          096H
P3M1         DATA          0B1H
P3M0         DATA          0B2H
P4M1         DATA          0B3H
P4M0         DATA          0B4H
P5M1         DATA          0C9H
```

```
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0023H
                LJMP            UART_ISR

                ORG             0100H

UART_ISR:
                JNB             TI,CHKRI
                CLR             TI
                CLR             BUSY
CHKRI:
                JNB             RI,UARTISR_EXIT
                CLR             RI
UARTISR_EXIT:
                RETI

UART_INIT:
                MOV             SCON,#50H
                MOV             TMOD,#00H
                MOV             TL1,#0E8H                       ;65536-11059200/115200/4=0FFE8H
                MOV             TH1,#0FFH
                SETB            TR1
                MOV             AUXR,#40H
                CLR             BUSY
                RET

UART_SEND:
                JB              BUSY,$
                SETB            BUSY
                MOV             SBUF,A
                RET

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                LCALL           UART_INIT
                SETB            ES
                SETB            EA

                MOV             P_SW2,#80H
                MOV             DPTR,# VREF_ADDR
                CLR             A
                MOVX            A,@DPTR                         ; Read the high byte of the internal 1.19V reference signal source
                LCALL           UART_SEND
                INC             DPTR
                MOVX        A,@DPTR                 ; Read the low byte of the internal 1.19V reference signal source
                LCALL       UART_SEND

LOOP:
                JMP         LOOP

                END
```

7.5.2 Read Internal Referrence Voltage (from Flash)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (65536 - FOSC / 115200 / 4)

sfr         AUXR            =       0x8e;

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

bit         busy;
int         *BGV;

void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
    }
}

void UartInit()
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}

void UartSend(char dat)
{
        while (busy);
        busy = 1;
        SBUF = dat;
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        BGV = (int code *)0x3ff7;                          // STC8H1K16
        UartInit();
        ES = 1;
        EA = 1;
        UartSend(*BGV >> 8);                               //Read the high byte of the internal reference voltage
        UartSend(*BGV);                                    //Read the low byte of the internal reference voltage

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

AUXR        DATA        8EH
BGV         EQU         01FF7H                  ;STC8H1K16

BUSY        BIT         20H.0

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         0023H
            LJMP        UART_ISR
```

```
                ORG             0100H

UART_ISR:
                JNB             TI,CHKRI
                CLR             TI
                CLR             BUSY
CHKRI:
                JNB             RI,UARTISR_EXIT
                CLR             RI
UARTISR_EXIT:
                RETI

UART_INIT:
                MOV             SCON,#50H
                MOV             TMOD,#00H
                MOV             TL1,#0E8H               ;65536-11059200/115200/4=0FFE8H
                MOV             TH1,#0FFH
                SETB            TR1
                MOV             AUXR,#40H
                CLR             BUSY
                RET

UART_SEND:
                JB              BUSY,$
                SETB            BUSY
                MOV             SBUF,A
                RET

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                LCALL           UART_INIT
                SETB            ES
                SETB            EA

                MOV             DPTR,#BGV
                CLR             A
                MOVC            A,@A+DPTR               ;Read the high byte of the internal reference voltage
                LCALL           UART_SEND
                MOV             A,#1
                MOVC            A,@A+DPTR               ;Read the low byte of the internal reference voltage
                LCALL           UART_SEND

LOOP:
                JMP             LOOP

                END
```

~~7.5.3 Read Internal Referrence Voltage (Read from RAM)~~

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (65536 - FOSC / 115200 / 4)

sfr         AUXR            =       0x8e;

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

bit         busy;
int         *BGV;

void UartIsr() interrupt 4
{
        if (TI)
        {
                TI = 0;
                busy = 0;
        }
        if (RI)
        {
                RI = 0;
        }
}

void UartInit()
{
        SCON = 0x50;
        TMOD = 0x00;
        TL1 = BRT;
        TH1 = BRT >> 8;
        TR1 = 1;
        AUXR = 0x40;
        busy = 0;
}

void UartSend(char dat)
{
        while (busy);
        busy = 1;
        SBUF = dat;
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;

        P5M0 = 0x00;
        P5M1 = 0x00;

        BGV = (int idata *)0xef;
        UartInit();
        ES = 1;
        EA = 1;
        UartSend(*BGV >> 8);                            //Read the high byte of the internal reference voltage
        UartSend(*BGV);                                 //Read the low byte of the internal reference voltage

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

AUXR            DATA            8EH
BGV             DATA            0EFH

BUSY            BIT             20H.0

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0023H
                LJMP            UART_ISR

                ORG             0100H

UART_ISR:
                JNB             TI,CHKRI
                CLR             TI
                CLR             BUSY
```

```
CHKRI:
                JNB             RI,UARTISR_EXIT
                CLR             RI
UARTISR_EXIT:
                RETI

UART_INIT:
                MOV             SCON,#50H
                MOV             TMOD,#00H
                MOV             TL1,#0E8H                       ;65536-11059200/115200/4=0FFE8H
                MOV             TH1,#0FFH
                SETB            TR1
                MOV             AUXR,#40H
                CLR             BUSY
                RET

UART_SEND:
                JB              BUSY,$
                SETB            BUSY
                MOV             SBUF,A
                RET

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                LCALL           UART_INIT
                SETB            ES
                SETB            EA

                MOV             R0,#BGV
                MOV             A,@R0                           ;Read the high byte of the internal reference voltage
                LCALL           UART_SEND
                INC             R0
                MOV             A,@R0                           ;Read the low byte of the internal reference voltage
                LCALL           UART_SEND

LOOP:
                JMP             LOOP

                END
```

7.5.4 Read the Unique ID (Read from CHIPID)

C language code

```
//Operating frequency for test is 11.0592MHz
```

```
#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (65536 - FOSC / 115200 / 4)

#define     CPUIDBASE           0xfde0
#define     ID_ADDR             ((unsigned char volatile xdata *)(CPUIDBASE + 0x00))

sfr         AUXR            =   0x8e;
sfr         P_SW2           =   0xba;

sfr         P0M1            =   0x93;
sfr         P0M0            =   0x94;
sfr         P1M1            =   0x91;
sfr         P1M0            =   0x92;
sfr         P2M1            =   0x95;
sfr         P2M0            =   0x96;
sfr         P3M1            =   0xb1;
sfr         P3M0            =   0xb2;
sfr         P4M1            =   0xb3;
sfr         P4M0            =   0xb4;
sfr         P5M1            =   0xc9;
sfr         P5M0            =   0xca;

bit         busy;

void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
    }
}

void UartInit()
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}

void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}

void main()
```

```
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        char i;

        UartInit();
        ES = 1;
        EA = 1;

        P_SW2 = 0x80;
        for (i=0; i<7; i++)
        {
                UartSend(ID_ADDR[i]);
        }

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

CPUIDBASE     EQU   0FDE0H
ID_ADDR       EQU   CPUIDBASE + 00H

AUXR          DATA          8EH
P_SW2         DATA          0BAH

BUSY          BIT           20H.0

P0M1          DATA          093H
P0M0          DATA          094H
P1M1          DATA          091H
P1M0          DATA          092H
P2M1          DATA          095H
P2M0          DATA          096H
P3M1          DATA          0B1H
P3M0          DATA          0B2H
P4M1          DATA          0B3H
P4M0          DATA          0B4H
P5M1          DATA          0C9H
P5M0          DATA          0CAH

              ORG           0000H
              LJMP          MAIN
              ORG           0023H
              LJMP          UART_ISR

              ORG           0100H
```

```
UART_ISR:
                JNB             TI,CHKRI
                CLR             TI
                CLR             BUSY
CHKRI:
                JNB             RI,UARTISR_EXIT
                CLR             RI
UARTISR_EXIT:
                RETI

UART_INIT:
                MOV             SCON,#50H
                MOV             TMOD,#00H
                MOV             TL1,#0E8H                       ;65536-11059200/115200/4=0FFE8H
                MOV             TH1,#0FFH
                SETB            TR1
                MOV             AUXR,#40H
                CLR             BUSY
                RET

UART_SEND:
                JB              BUSY,$
                SETB            BUSY
                MOV             SBUF,A
                RET

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                LCALL           UART_INIT
                SETB            ES
                SETB            EA

                MOV             P_SW2,#80H
                MOV             DPTR,#ID_ADDR
                MOV             R1,#7
NEXT:           CLR             A
                MOVX            A,@DPTR
                LCALL           UART_SEND
                INC             DPTR
                DJNZ            R1,NEXT

LOOP:
                JMP             LOOP

                END
```

7.5.5 Read the Unique ID (Read from Flash)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (65536 - FOSC / 115200 / 4)

sfr         AUXR            =       0x8e;

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

bit         busy;
char        *ID;

void UartIsr() interrupt 4
{
        if (TI)
        {
                TI = 0;
                busy = 0;
        }
        if (RI)
        {
                RI = 0;
        }
}

void UartInit()
{
        SCON = 0x50;
        TMOD = 0x00;
        TL1 = BRT;
        TH1 = BRT >> 8;
        TR1 = 1;
        AUXR = 0x40;
        busy = 0;
}

void UartSend(char dat)
{
        while (busy);
        busy = 1;
        SBUF = dat;
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        char i;

        ID = (char code *)0x3ff9;                                   // STC8H1K16
        UartInit();
        ES = 1;
        EA = 1;

        for (i=0; i<7; i++)
        {
                UartSend(ID[i]);
        }

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

AUXR            DATA            8EH
ID              EQU             03FF9H                          ; STC8H1K16

BUSY            BIT             20H.0

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0023H
                LJMP            UART_ISR

                ORG             0100H
```

```
UART_ISR:
                JNB             TI,CHKRI
                CLR             TI
                CLR             BUSY
CHKRI:
                JNB             RI,UARTISR_EXIT
                CLR             RI
UARTISR_EXIT:
                RETI

UART_INIT:
                MOV             SCON,#50H
                MOV             TMOD,#00H
                MOV             TL1,#0E8H                       ;65536-11059200/115200/4=0FFE8H
                MOV             TH1,#0FFH
                SETB            TR1
                MOV             AUXR,#40H
                CLR             BUSY
                RET

UART_SEND:
                JB              BUSY,$
                SETB            BUSY
                MOV             SBUF,A
                RET

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                LCALL           UART_INIT
                SETB            ES
                SETB            EA

                MOV             DPTR,#ID
                MOV             R1,#7
NEXT:           CLR             A
                MOVC            A,@A+DPTR
                LCALL           UART_SEND
                INC             DPTR
                DJNZ            R1,NEXT

LOOP:
                JMP             LOOP

                END
```

~~7.5.6 Read the Unique ID (Read from RAM)~~

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (65536 - FOSC / 115200 / 4)

sfr         AUXR            =       0x8e;

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

bit         busy;
char        *ID;

void UartIsr() interrupt 4
{
        if (TI)
        {
                TI = 0;
                busy = 0;
        }
        if (RI)
        {
                RI = 0;
        }
}

void UartInit()
{
        SCON = 0x50;
        TMOD = 0x00;
        TL1 = BRT;
        TH1 = BRT >> 8;
        TR1 = 1;
        AUXR = 0x40;
        busy = 0;
}

void UartSend(char dat)
{
        while (busy);
        busy = 1;
        SBUF = dat;
}
```

```
void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        char i;

        ID = (char idata *)0xf1;
        UartInit();
        ES = 1;
        EA = 1;

        for (i=0; i<7; i++)
        {
                UartSend(ID[i]);
        }

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

AUXR            DATA            8EH
ID              DATA            0F1H

BUSY            BIT             20H.0

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0023H
                LJMP            UART_ISR

                ORG             0100H

UART_ISR:
```

```
                JNB             TI,CHKRI
                CLR             TI
                CLR             BUSY
CHKRI:
                JNB             RI,UARTISR_EXIT
                CLR             RI
UARTISR_EXIT:
                RETI

UART_INIT:
                MOV             SCON,#50H
                MOV             TMOD,#00H
                MOV             TL1,#0E8H                       ;65536-11059200/115200/4=0FFE8H
                MOV             TH1,#0FFH
                SETB            TR1
                MOV             AUXR,#40H
                CLR             BUSY
                RET

UART_SEND:
                JB              BUSY,$
                SETB            BUSY
                MOV             SBUF,A
                RET

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                LCALL           UART_INIT
                SETB            ES
                SETB            EA

                MOV             R0,#ID
                MOV             R1,#7
NEXT:           MOV             A,@R0
                LCALL           UART_SEND
                INC             R0
                DJNZ            R1,NEXT

LOOP:
                JMP             LOOP

                END
```

7.5.7 Read the Frequency of 32K Power-down Wake-up Timer (Read from CHIPID)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (65536 - FOSC / 115200 / 4)

#define     CPUIDBASE           0xfde0
#define     F32K_ADDR           (*(unsigned int volatile xdata *)(CPUIDBASE + 0x09))

sfr         AUXR            =   0x8e;
sfr         P_SW2           =   0xba;

sfr         P0M1            =   0x93;
sfr         P0M0            =   0x94;
sfr         P1M1            =   0x91;
sfr         P1M0            =   0x92;
sfr         P2M1            =   0x95;
sfr         P2M0            =   0x96;
sfr         P3M1            =   0xb1;
sfr         P3M0            =   0xb2;
sfr         P4M1            =   0xb3;
sfr         P4M0            =   0xb4;
sfr         P5M1            =   0xc9;
sfr         P5M0            =   0xca;

bit         busy;

void UartIsr() interrupt 4
{
        if (TI)
        {
                TI = 0;
                busy = 0;
        }
        if (RI)
        {
                RI = 0;
        }
}

void UartInit()
{
        SCON = 0x50;
        TMOD = 0x00;
        TL1 = BRT;
        TH1 = BRT >> 8;
        TR1 = 1;
        AUXR = 0x40;
        busy = 0;
}
```

```
void UartSend(char dat)
{
        while (busy);
        busy = 1;
        SBUF = dat;
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        F32K = (int code *)0xeff5;                          // STC8A8K60S4
        UartInit();
        ES = 1;
        EA = 1;

        P_SW2 = 0x80;
        UartSend(F32K_ADDR >> 8);                           // Read high byte of 32K frequency
        UartSend(F32K_ADDR);                                // Read low byte of 32K frequency

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

CPUIDBASE               EQU     0FDE0H
F32K_ADDR               EQU     CPUIDBASE + 09H

AUXR            DATA            8EH
P_SW2           DATA            0BAH

BUSY            BIT             20H.0

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0023H
                LJMP            UART_ISR

                ORG             0100H

UART_ISR:
                JNB             TI,CHKRI
                CLR             TI
                CLR             BUSY
CHKRI:
                JNB             RI,UARTISR_EXIT
                CLR             RI
UARTISR_EXIT:
                RETI

UART_INIT:
                MOV             SCON,#50H
                MOV             TMOD,#00H
                MOV             TL1,#0E8H                       ;65536-11059200/115200/4=0FFE8H
                MOV             TH1,#0FFH
                SETB            TR1
                MOV             AUXR,#40H
                CLR             BUSY
                RET

UART_SEND:
                JB              BUSY,$
                SETB            BUSY
                MOV             SBUF,A
                RET

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                LCALL           UART_INIT
                SETB            ES
                SETB            EA

                MOV             P_SW2,#80H
                MOV             DPTR,# F32K_ADDR
                CLR             A
                MOVX            A,@DPTR                         ; Read high byte of 32K frequency
                LCALL           UART_SEND
                INC             DPTR
                CLR             A
                MOVX            A,@ DPTR                        ; Read low byte of 32K frequency
                LCALL           UART_SEND
```

```
LOOP:
                JMP             LOOP

                END
```

7.5.8 Read the Frequency of 32K Power-down Wake-up Timer (Read from Flash)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (65536 - FOSC / 115200 / 4)

sfr         AUXR            =       0x8e;

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

bit         busy;
int         *F32K;

void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
    }
}

void UartInit()
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
```

```
        busy = 0;
}

void UartSend(char dat)
{
        while (busy);
        busy = 1;
        SBUF = dat;
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        F32K = (int code *)0x3ff5;                          // STC8H1K16
        UartInit();
        ES = 1;
        EA = 1;

        UartSend(*F32K >> 8);                               //Read high byte of 32K frequency
        UartSend(*F32K);                                    //Read low byte of 32K frequency

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

AUXR        DATA        8EH
F32K        EQU         03FF5H                      ; STC8H1K16

BUSY        BIT         20H.0

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         0023H
```

```
                LJMP            UART_ISR

                ORG             0100H

UART_ISR:
                JNB             TI,CHKRI
                CLR             TI
                CLR             BUSY
CHKRI:
                JNB             RI,UARTISR_EXIT
                CLR             RI
UARTISR_EXIT:
                RETI

UART_INIT:
                MOV             SCON,#50H
                MOV             TMOD,#00H
                MOV             TL1,#0E8H                       ;65536-11059200/115200/4=0FFE8H
                MOV             TH1,#0FFH
                SETB            TR1
                MOV             AUXR,#40H
                CLR             BUSY
                RET

UART_SEND:
                JB              BUSY,$
                SETB            BUSY
                MOV             SBUF,A
                RET

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                LCALL           UART_INIT
                SETB            ES
                SETB            EA

                MOV             DPTR,#F32K
                CLR             A
                MOVC            A,@A+DPTR                       ;Read high byte of 32K frequency
                LCALL           UART_SEND
                INC             DPTR
                CLR             A
                MOVC            A,@A+DPTR                       ;Read low byte of 32K frequency
                LCALL           UART_SEND

LOOP:
                JMP             LOOP
                END
```

~~7.5.9 Read the Frequency of 32K Power-down Wake-up Timer (Read from RAM)~~

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (65536 - FOSC / 115200 / 4)

sfr         AUXR            =       0x8e;

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

bit         busy;
int         *F32K;

void UartIsr() interrupt 4
{
        if (TI)
        {
                TI = 0;
                busy = 0;
        }
        if (RI)
        {
                RI = 0;
        }
}

void UartInit()
{
        SCON = 0x50;
        TMOD = 0x00;
        TL1 = BRT;
        TH1 = BRT >> 8;
        TR1 = 1;
        AUXR = 0x40;
        busy = 0;
}
```

```
void UartSend(char dat)
{
        while (busy);
        busy = 1;
        SBUF = dat;
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        F32K = (int idata *)0xf8;
        UartInit();
        ES = 1;
        EA = 1;

        UartSend(*F32K >> 8);                                   //Read high byte of 32K frequency
        UartSend(*F32K);                                        //Read low byte of 32K frequency

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

AUXR            DATA            8EH
F32K            DATA            0F8H

BUSY            BIT             20H.0

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0023H
                LJMP            UART_ISR
                ORG         0100H

UART_ISR:
                JNB         TI,CHKRI
                CLR         TI
                CLR         BUSY
CHKRI:
                JNB         RI,UARTISR_EXIT
                CLR         RI
UARTISR_EXIT:
                RETI

UART_INIT:
                MOV         SCON,#50H
                MOV         TMOD,#00H
                MOV         TL1,#0E8H                   ;65536-11059200/115200/4=0FFE8H
                MOV         TH1,#0FFH
                SETB        TR1
                MOV         AUXR,#40H
                CLR         BUSY
                RET

UART_SEND:
                JB          BUSY,$
                SETB        BUSY
                MOV         SBUF,A
                RET

MAIN:
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                LCALL       UART_INIT
                SETB        ES
                SETB        EA

                MOV         R0,#F32K
                MOV         A,@R0                       ;Read high byte of 32K frequency
                LCALL       UART_SEND
                INC         R0
                MOV         A,@R0                       ;Read low byte of 32K frequency
                LCALL       UART_SEND

LOOP:
                JMP         LOOP

                END
```

7.5.10 Read the User-defined internal IRC Frequency (Read from CHIPID)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     CLKSEL          (*(unsigned char volatile xdata *)0xfe00)
#define     CLKDIV          (*(unsigned char volatile xdata *)0xfe01)

#define     USBCHIPID
#define     CPUIDBASE           0xfde0

#define     T22M_ADDR           (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0b))       //22.1184MHz
#define     T24M_ADDR           (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0c))       //24MHz
#ifdef      USBCHIPID
#define     T27M_ADDR           (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0d))       //27MHz
#define     T30M_ADDR           (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0e))       //30MHz
#define     T33M_ADDR           (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0f))       //33.1776MHz
#define     T35M_ADDR           (*(unsigned char volatile xdata *)(CPUIDBASE + 0x10))       //35MHz
#define     T36M_ADDR           (*(unsigned char volatile xdata *)(CPUIDBASE + 0x11))       //36.864MHz
#define     T40M_ADDR           (*(unsigned char volatile xdata *)(CPUIDBASE + 0x12))       //40MHz
#define     T44M_ADDR           (*(unsigned char volatile xdata *)(CPUIDBASE + 0x13))       //44.2368MHz
#define     T48M_ADDR           (*(unsigned char volatile xdata *)(CPUIDBASE + 0x14))       //48MHz
#else
#define     T20M_ADDR           (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0d))       //20MHz
#define     T27M_ADDR           (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0e))       //27MHz
#define     T30M_ADDR           (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0f))       //30MHz
#define     T33M_ADDR           (*(unsigned char volatile xdata *)(CPUIDBASE + 0x10))       //33.1776MHz
#define     T35M_ADDR           (*(unsigned char volatile xdata *)(CPUIDBASE + 0x11))       //35MHz
#define     T36M_ADDR           (*(unsigned char volatile xdata *)(CPUIDBASE + 0x12))       //36.864MHz
#define     T40M_ADDR           (*(unsigned char volatile xdata *)(CPUIDBASE + 0x13))       //40MHz
#define     T45M_ADDR           (*(unsigned char volatile xdata *)(CPUIDBASE + 0x14))       //45MHz
#endif
#define     VRT6M_ADDR          (*(unsigned char volatile xdata *)(CPUIDBASE + 0x15))       //VRTRIM_6M
#define     VRT10M_ADDR         (*(unsigned char volatile xdata *)(CPUIDBASE + 0x16))       //VRTRIM_10M
#define     VRT27M_ADDR         (*(unsigned char volatile xdata *)(CPUIDBASE + 0x17))       //VRTRIM_27M
#define     VRT44M_ADDR         (*(unsigned char volatile xdata *)(CPUIDBASE + 0x18))       //VRTRIM_44M

sfr         P_SW2           =   0xba;
sfr         IRCBAND         =   0x9d;
sfr         IRTRIM          =   0x9f;
sfr         VRTRIM          =   0xa6;

sfr         P1M1            =   0x91;
sfr         P1M0            =   0x92;
sfr         P0M1            =   0x93;
sfr         P0M0            =   0x94;
sfr         P2M1            =   0x95;
sfr         P2M0            =   0x96;
sfr         P3M1            =   0xb1;
sfr         P3M0            =   0xb2;
sfr         P4M1            =   0xb3;
sfr         P4M0            =   0xb4;
sfr         P5M1            =   0xc9;
sfr         P5M0            =   0xca;
```

```
void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

//#ifndef    USBCHIPID
//      //Select 20MHz
//      P_SW2 = 0x80;
//      CLKDIV = 0x04;
//      IRTRIM = T20M_ADDR;
//      VRTRIM = VRT27M_ADDR;
//      IRCBAND = 0x02;
//      CLKDIV = 0x00;
//#endif

//      // Select 22.1184MHz
//      P_SW2 = 0x80;
//      CLKDIV = 0x04;
//      IRTRIM = T22M_ADDR;
//      VRTRIM = VRT27M_ADDR;
//      IRCBAND = 0x02;
//      CLKDIV = 0x00;

        // Select 24MHz
        P_SW2 = 0x80;
        CLKDIV = 0x04;
        IRTRIM = T24M_ADDR;
        VRTRIM = VRT27M_ADDR;
        IRCBAND = 0x02;
        CLKDIV = 0x00;

//      // Select 27MHz
//      P_SW2 = 0x80;
//      CLKDIV = 0x04;
//      IRTRIM = T27M_ADDR;
//      VRTRIM = VRT27M_ADDR;
//      IRCBAND = 0x02;
//      CLKDIV = 0x00;

//      // Select 30MHz
//      P_SW2 = 0x80;
//      CLKDIV = 0x04;
//      IRTRIM = T30M_ADDR;
//      VRTRIM = VRT27M_ADDR;
//      IRCBAND = 0x02;
//      CLKDIV = 0x00;

//      // Select 33.1776MHz
//      P_SW2 = 0x80;
//      CLKDIV = 0x04;
//      IRTRIM = T33M_ADDR;
//      VRTRIM = VRT27M_ADDR;
//      IRCBAND = 0x02;
//      CLKDIV = 0x00;

//      // Select 35MHz
//      P_SW2 = 0x80;
//      CLKDIV = 0x04;
//      IRTRIM = T35M_ADDR;
//      VRTRIM = VRT44M_ADDR;
//      IRCBAND = 0x03;
//      CLKDIV = 0x00;

//#ifdef    USBCHIPID
//      // Select 44.2368MHz
//      P_SW2 = 0x80;
//      CLKDIV = 0x04;
//      IRTRIM = T44M_ADDR;
//      VRTRIM = VRT44M_ADDR;
//      IRCBAND = 0x03;
//      CLKDIV = 0x00;

//      // Select 48MHz
//      P_SW2 = 0x80;
//      CLKDIV = 0x04;
//      IRTRIM = T48M_ADDR;
//      VRTRIM = VRT44M_ADDR;
//      IRCBAND = 0x03;
//      CLKDIV = 0x00;
//#else
//      // Select 40MHz
//      P_SW2 = 0x80;
//      CLKDIV = 0x04;
//      IRTRIM = T40M_ADDR;
//      VRTRIM = VRT44M_ADDR;
//      IRCBAND = 0x03;
//      CLKDIV = 0x00;

//      // Select 45MHz
//      P_SW2 = 0x80;
//      CLKDIV = 0x04;
//      IRTRIM = T45M_ADDR;
//      VRTRIM = VRT44M_ADDR;
//      IRCBAND = 0x03;
//      CLKDIV = 0x00;
//#endif

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

$SET(USBCHIPID = 1)
CPUIDBASE       EQU     0FDE0H

T22M_ADDR       EQU     CPUIDBASE + 0BH         ;22.1184MHz
T24M_ADDR       EQU     CPUIDBASE + 0CH         ;24MHz
$IF(USBCHIPID == 1)
```

```
T27M_ADDR         EQU     CPUIDBASE + 0DH         ;27MHz
T30M_ADDR         EQU     CPUIDBASE + 0EH         ;30MHz
T33M_ADDR         EQU     CPUIDBASE + 0FH         ;33.1776MHz
T35M_ADDR         EQU     CPUIDBASE + 10H         ;35MHz
T36M_ADDR         EQU     CPUIDBASE + 11H         ;36.864MHz
T40M_ADDR         EQU     CPUIDBASE + 12H         ;40MHz
T44M_ADDR         EQU     CPUIDBASE + 13H         ;44.2368MHz
T48M_ADDR         EQU     CPUIDBASE + 14H         ;20MHz
$ELSE
T20M_ADDR         EQU     CPUIDBASE + 0DH         ;20MHz
T27M_ADDR         EQU     CPUIDBASE + 0EH         ;27MHz
T30M_ADDR         EQU     CPUIDBASE + 0FH         ;30MHz
T33M_ADDR         EQU     CPUIDBASE + 10H         ;33.1776MHz
T35M_ADDR         EQU     CPUIDBASE + 11H         ;35MHz
T36M_ADDR         EQU     CPUIDBASE + 12H         ;36.864MHz
T40M_ADDR         EQU     CPUIDBASE + 13H         ;40MHz
T45M_ADDR         EQU     CPUIDBASE + 14H         ;45MHz
$ENDIF
VRT6M_ADDR        EQU     CPUIDBASE + 15H         ;VRTRIM_6M
VRT10M_ADDR       EQU     CPUIDBASE + 16H         ;VRTRIM_10M
VRT27M_ADDR       EQU     CPUIDBASE + 17H         ;VRTRIM_27M
VRT44M_ADDR       EQU     CPUIDBASE + 18H         ;VRTRIM_44M

P_SW2       DATA        0BAH
CLKSEL      EQU         0FE00H
CLKDIV      EQU         0FE01H

IRCBAND     DATA        09DH
IRCTRIM     DATA        09FH
VRTRIM      DATA        0A6H

P1M1        DATA        091H
P1M0        DATA        092H
P0M1        DATA        093H
P0M0        DATA        094H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN

            ORG         0100H
MAIN:
            MOV         SP, #5FH
            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
            MOV         P3M1, #00H
            MOV         P4M0, #00H
            MOV         P4M1, #00H
```

```
                    MOV         P5M0, #00H
                    MOV         P5M1, #00H

;$IF(USBCHIPID == 0)
;                   ; Select 20MHz
;                   MOV         P_SW2,#80H
;                   MOV         A,#4
;                   MOV         DPTR,#CLKDIV
;                   MOVX        @DPTR,A
;                   MOV         DPTR,#T20M_ADDR
;                   CLR         A
;                   MOVX        A,@DPTR
;                   MOV         IRTRIM,A
;                   MOV         DPTR,#VRT27M_ADDR
;                   CLR         A
;                   MOVX        A,@DPTR
;                   MOV         VRTRIM,A
;                   MOV         IRCBAND,#02H
;                   MOV         A,#0
;                   MOV         DPTR,#CLKDIV
;                   MOVX        @DPTR,A
;                   MOV         P_SW2,#00H
;$ENDIF

;                   ; Select 22.1184MHz
;                   MOV         P_SW2,#80H
;                   MOV         A,#4
;                   MOV         DPTR,#CLKDIV
;                   MOVX        @DPTR,A
;                   MOV         DPTR,#T22M_ADDR
;                   CLR         A
;                   MOVX        A,@DPTR
;                   MOV         IRTRIM,A
;                   MOV         DPTR,#VRT27M_ADDR
;                   CLR         A
;                   MOVX        A,@DPTR
;                   MOV         VRTRIM,A
;                   MOV         IRCBAND,#02H
;                   MOV         A,#0
;                   MOV         DPTR,#CLKDIV
;                   MOVX        @DPTR,A
;                   MOV         P_SW2,#00H

                    ; Select 24MHz
                    MOV         P_SW2,#80H
                    MOV         A,#4
                    MOV         DPTR,#CLKDIV
                    MOV         @DPTR,A
                    MOV         DPTR,#T24M_ADDR
                    CLR         A
                    MOVX        A,@DPTR
                    MOV         IRTRIM,A
                    MOV         DPTR,#VRT27M_ADDR
                    CLR         A
                    MOVX        A,@DPTR
                    MOV         VRTRIM,A
                    MOV         IRCBAND,#02H
                    MOV         A,#0
                    MOV         DPTR,#CLKDIV
                    MOV         @DPTR,A
```

```
                MOV             P_SW2,#00H

;               ; Select 27MHz
;               MOV             P_SW2,#80H
;               MOV             A,#4
;               MOV             DPTR,#CLKDIV
;               MOVX            @DPTR,A
;               MOV             DPTR,#T27M_ADDR
;               CLR             A
;               MOVX            A,@DPTR
;               MOV             IRTRIM,A
;               MOV             DPTR,#VRT27M_ADDR
;               CLR             A
;               MOVX            A,@DPTR
;               MOV             VRTRIM,A
;               MOV             IRCBAND,#02H
;               MOV             A,#0
;               MOV             DPTR,#CLKDIV
;               MOVX            @DPTR,A
;               MOV             P_SW2,#00H

;               ; Select 30MHz
;               MOV             P_SW2,#80H
;               MOV             A,#4
;               MOV             DPTR,#CLKDIV
;               MOVX            @DPTR,A
;               MOV             DPTR,#T30M_ADDR
;               CLR             A
;               MOVX            A,@DPTR
;               MOV             IRTRIM,A
;               MOV             DPTR,#VRT27M_ADDR
;               CLR             A
;               MOVX            A,@DPTR
;               MOV             VRTRIM,A
;               MOV             IRCBAND,#02H
;               MOV             A,#0
;               MOV             DPTR,#CLKDIV
;               MOVX            @DPTR,A
;               MOV             P_SW2,#00H

;               ; Select 33.1776MHz
;               MOV             P_SW2,#80H
;               MOV             A,#4
;               MOV             DPTR,#CLKDIV
;               MOVX            @DPTR,A
;               MOV             DPTR,#T33M_ADDR
;               CLR             A
;               MOVX            A,@DPTR
;               MOV             IRTRIM,A
;               MOV             DPTR,#VRT27M_ADDR
;               CLR             A
;               MOVX            A,@DPTR
;               MOV             VRTRIM,A
;               MOV             IRCBAND,#02H
;               MOV             A,#0
;               MOV             DPTR,#CLKDIV
;               MOVX            @DPTR,A
;               MOV             P_SW2,#00H

;               ; Select 35MHz
;                       MOV         P_SW2,#80H
;                       MOV         A,#4
;                       MOV         DPTR,#CLKDIV
;                       MOVX        @DPTR,A
;                       MOV         DPTR,#T35M_ADDR
;                       CLR         A
;                       MOVX        A,@DPTR
;                       MOV         IRTRIM,A
;                       MOV         DPTR,#VRT44M_ADDR
;                       CLR         A
;                       MOVX        A,@DPTR
;                       MOV         VRTRIM,A
;                       MOV         IRCBAND,#03H
;                       MOV         A,#0
;                       MOV         DPTR,#CLKDIV
;                       MOVX        @DPTR,A
;                       MOV         P_SW2,#00H

;                       ; Select 36.864MHz
;                       MOV         P_SW2,#80H
;                       MOV         A,#4
;                       MOV         DPTR,#CLKDIV
;                       MOVX        @DPTR,A
;                       MOV         DPTR,#T36M_ADDR
;                       CLR         A
;                       MOVX        A,@DPTR
;                       MOV         IRTRIM,A
;                       MOV         DPTR,#VRT44M_ADDR
;                       CLR         A
;                       MOVX        A,@DPTR
;                       MOV         VRTRIM,A
;                       MOV         IRCBAND,#03H
;                       MOV         A,#0
;                       MOV         DPTR,#CLKDIV
;                       MOVX        @DPTR,A
;                       MOV         P_SW2,#00H

;$IF(USBCHIPID == 1)
;                       ; Select 44.2368MHz
;                       MOV         P_SW2,#80H
;                       MOV         A,#4
;                       MOV         DPTR,#CLKDIV
;                       MOVX        @DPTR,A
;                       MOV         DPTR,#T44M_ADDR
;                       CLR         A
;                       MOVX        A,@DPTR
;                       MOV         IRTRIM,A
;                       MOV         DPTR,#VRT44M_ADDR
;                       CLR         A
;                       MOVX        A,@DPTR
;                       MOV         VRTRIM,A
;                       MOV         IRCBAND,#03H
;                       MOV         A,#0
;                       MOV         DPTR,#CLKDIV
;                       MOVX        @DPTR,A
;                       MOV         P_SW2,#00H

;                       ; Select 48MHz
;                       MOV         P_SW2,#80H
;                       MOV         A,#4
;               MOV         DPTR,#CLKDIV
;               MOVX        @DPTR,A
;               MOV         DPTR,#T48M_ADDR
;               CLR         A
;               MOVX        A,@DPTR
;               MOV         IRTRIM,A
;               MOV         DPTR,#VRT44M_ADDR
;               CLR         A
;               MOVX        A,@DPTR
;               MOV         VRTRIM,A
;               MOV         IRCBAND,#03H
;               MOV         A,#0
;               MOV         DPTR,#CLKDIV
;               MOVX        @DPTR,A
;               MOV         P_SW2,#00H
;$ELSE
;               ; Select 40MHz
;               MOV         P_SW2,#80H
;               MOV         A,#4
;               MOV         DPTR,#CLKDIV
;               MOVX        @DPTR,A
;               MOV         DPTR,#T40M_ADDR
;               CLR         A
;               MOVX        A,@DPTR
;               MOV         IRTRIM,A
;               MOV         DPTR,#VRT44M_ADDR
;               CLR         A
;               MOVX        A,@DPTR
;               MOV         VRTRIM,A
;               MOV         IRCBAND,#03H
;               MOV         A,#0
;               MOV         DPTR,#CLKDIV
;               MOVX        @DPTR,A
;               MOV         P_SW2,#00H

;               ; Select 45MHz
;               MOV         P_SW2,#80H
;               MOV         A,#4
;               MOV         DPTR,#CLKDIV
;               MOVX        @DPTR,A
;               MOV         DPTR,#T45M_ADDR
;               CLR         A
;               MOVX        A,@DPTR
;               MOV         IRTRIM,A
;               MOV         DPTR,#VRT44M_ADDR
;               CLR         A
;               MOVX        A,@DPTR
;               MOV         VRTRIM,A
;               MOV         IRCBAND,#03H
;               MOV         A,#0
;               MOV         DPTR,#CLKDIV
;               MOVX        @DPTR,A
;               MOV         P_SW2,#00H
;$ENDIF

                JMP         $

                END
```

7.5.11 Read the User-defined internal IRC Frequency (Read from Flash)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     CKSEL           (*(unsigned char volatile xdata *)0xfe00)
#define     CLKDIV          (*(unsigned char volatile xdata *)0xfe01)

// The following table is the parameter list of STC8H1K08-20Pin
#define     ID_ROMADDR              ((unsigned char code *)0x1ff9)
#define     VREF_ROMADDR            (*(unsigned int code *)0x1ff7)
#define     F32K_ROMADDR            (*(unsigned int code *)0x1ff5)
#define     T22M_ROMADDR            (*(unsigned char code *)0x1ff4)          //22.1184MHz
#define     T24M_ROMADDR            (*(unsigned char code *)0x1ff3)          //24MHz
#define     T20M_ROMADDR            (*(unsigned char code *)0x1ff2)          //20MHz
#define     T27M_ROMADDR            (*(unsigned char code *)0x1ff1)          //27MHz
#define     T30M_ROMADDR            (*(unsigned char code *)0x1ff0)          //30MHz
#define     T33M_ROMADDR            (*(unsigned char code *)0x1fef)          //33.1776MHz
#define     T35M_ROMADDR            (*(unsigned char code *)0x1fee)          //35MHz
#define     T36M_ROMADDR            (*(unsigned char code *)0x1fed)          //36.864MHz
#define     VRT20M_ROMADDR          (*(unsigned char code *)0x1fea)          //VRTRIM_20M
#define     VRT35M_ROMADDR          (*(unsigned char code *)0x1fe9)          //VRTRIM_35M

sfr         P_SW2           =       0xba;
sfr         IRCBAND         =       0x9d;
sfr         IRTRIM          =       0x9f;
sfr         VRTRIM          =       0xa6;

sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;
//      //Select 20MHz
//      P_SW2 = 0x80;
//      CLKDIV = 0x04;
//      IRTRIM = T20M_ROMADDR;
//      VRTRIM = VRT20M_ROMADDR;
//      IRCBAND = 0x00;
//      CLKDIV = 0x00;

//      // Select 22.1184MHz
//      P_SW2 = 0x80;
//      CLKDIV = 0x04;
//      IRTRIM = T22M_ROMADDR;
//      VRTRIM = VRT20M_ROMADDR;
//      IRCBAND = 0x00;
//      CLKDIV = 0x00;

        // Select 24MHz
        P_SW2 = 0x80;
        CLKDIV = 0x04;
        IRTRIM = T24M_ROMADDR;
        VRTRIM = VRT20M_ROMADDR;
        IRCBAND = 0x00;
        CLKDIV = 0x00;

//      // Select 27MHz
//      P_SW2 = 0x80;
//      CLKDIV = 0x04;
//      IRTRIM = T27M_ROMADDR;
//      VRTRIM = VRT35M_ROMADDR;
//      IRCBAND = 0x01;
//      CLKDIV = 0x00;

//      // Select 30MHz
//      P_SW2 = 0x80;
//      CLKDIV = 0x04;
//      IRTRIM = T30M_ROMADDR;
//      VRTRIM = VRT35M_ROMADDR;
//      IRCBAND = 0x01;
//      CLKDIV = 0x00;

//      // Select 33.1776MHz
//      P_SW2 = 0x80;
//      CLKDIV = 0x04;
//      IRTRIM = T33M_ROMADDR;
//      VRTRIM = VRT35M_ROMADDR;
//      IRCBAND = 0x01;
//      CLKDIV = 0x00;

//      // Select 35MHz
//      P_SW2 = 0x80;
//      CLKDIV = 0x04;
//      IRTRIM = T35M_ROMADDR;
//      VRTRIM = VRT35M_ROMADDR;
//      IRCBAND = 0x01;
//      CLKDIV = 0x00;

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

; The following table is the parameter list of STC8H1K08-20Pin
ID_ROMADDR          EQU     01FF9H
VREF_ROMADDR        EQU     01FF7H
F32K_ROMADDR        EQU     01FF5H
T22M_ROMADDR        EQU     01FF4H                  //22.1184MHz
T24M_ROMADDR        EQU     01FF3H                  //24MHz
T20M_ROMADDR        EQU     01FF2H                  //20MHz
T27M_ROMADDR        EQU     01FF1H                  //27MHz
T30M_ROMADDR        EQU     01FF0H                  //30MHz
T33M_ROMADDR        EQU     01FEFH                  //33.1776MHz
T35M_ROMADDR        EQU     01FEEH                  //35MHz
T36M_ROMADDR        EQU     01FEDH                  //36.864MHz
VRT20M_ROMADDR      EQU     01FEAH                  //VRTRIM_20M
VRT35M_ROMADDR      EQU     01FE9H                  //VRTRIM_35M

P_SW2           DATA        0BAH
CKSEL           EQU         0FE00H
CLKDIV          EQU         0FE01H

IRCBAND         DATA        09DH
IRCTRIM         DATA        09FH
VRTRIM          DATA        0A6H

P1M1            DATA        091H
P1M0            DATA        092H
P0M1            DATA        093H
P0M0            DATA        094H
P2M1            DATA        095H
P2M0            DATA        096H
P3M1            DATA        0B1H
P3M0            DATA        0B2H
P4M1            DATA        0B3H
P4M0            DATA        0B4H
P5M1            DATA        0C9H
P5M0            DATA        0CAH

                ORG         0000H
                LJMP        MAIN

                ORG         0100H
MAIN:
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

;               ;Select 20MHz
;               MOV         P_SW2,#80H
;                 MOV         A,#4
;                 MOV         DPTR,#CLKDIV
;                 MOV         DPTR,#T20M_ROMADDR
;                 CLR         A
;                 MOVC        A,@A+DPTR
;                 MOV         IRTRIM,A
;                 MOV         DPTR,#VRT20M_ROMADDR
;                 CLR         A
;                 MOVC        A,@A+DPTR
;                 MOV         VRTRIM,A
;                 MOV         IRCBAND,#00H
;                 MOV         A,#0
;                 MOV         DPTR,#CLKDIV
;                 MOV         P_SW2,#00H

;                 ; Select 22.1184MHz
;                 MOV         P_SW2,#80H
;                 MOV         A,#4
;                 MOV         DPTR,#CLKDIV
;                 MOV         DPTR,#T22M_ROMADDR
;                 CLR         A
;                 MOVC        A,@A+DPTR
;                 MOV         IRTRIM,A
;                 MOV         DPTR,#VRT20M_ROMADDR
;                 CLR         A
;                 MOVC        A,@A+DPTR
;                 MOV         VRTRIM,A
;                 MOV         IRCBAND,#00H
;                 MOV         A,#0
;                 MOV         DPTR,#CLKDIV
;                 MOV         P_SW2,#00H

                  ; Select 24MHz
                  MOV         P_SW2,#80H
                  MOV         A,#4
                  MOV         DPTR,#CLKDIV
                  MOV         DPTR,#T24M_ROMADDR
                  CLR         A
                  MOVC        A,@A+DPTR
                  MOV         IRTRIM,A
                  MOV         DPTR,#VRT20M_ROMADDR
                  CLR         A
                  MOVC        A,@A+DPTR
                  MOV         VRTRIM,A
                  MOV         IRCBAND,#00H
                  MOV         A,#0
                  MOV         DPTR,#CLKDIV
                  MOV         P_SW2,#00H

;                 ; Select 27MHz
;                 MOV         P_SW2,#80H
;                 MOV         A,#4
;                 MOV         DPTR,#CLKDIV
;                 MOV         DPTR,#T27M_ROMADDR
;                 CLR         A
;                 MOVC        A,@A+DPTR
;                 MOV         IRTRIM,A
;                 MOV         DPTR,#VRT35M_ROMADDR
;                 CLR         A
;                 MOVC        A,@A+DPTR
;               MOV         VRTRIM,A
;               MOV         IRCBAND,#01H
;               MOV         A,#0
;               MOV         DPTR,#CLKDIV
;               MOV         P_SW2,#00H

;               ; Select 30MHz
;               MOV         P_SW2,#80H
;               MOV         A,#4
;               MOV         DPTR,#CLKDIV
;               MOV         DPTR,#T30M_ROMADDR
;               CLR         A
;               MOVC        A,@A+DPTR
;               MOV         IRTRIM,A
;               MOV         DPTR,#VRT35M_ROMADDR
;               CLR         A
;               MOVC        A,@A+DPTR
;               MOV         VRTRIM,A
;               MOV         IRCBAND,#01H
;               MOV         A,#0
;               MOV         DPTR,#CLKDIV
;               MOV         P_SW2,#00H

;               ; Select 33.1776MHz
;               MOV         P_SW2,#80H
;               MOV         A,#4
;               MOV         DPTR,#CLKDIV
;               MOV         DPTR,#T33M_ROMADDR
;               CLR         A
;               MOVC        A,@A+DPTR
;               MOV         IRTRIM,A
;               MOV         DPTR,#VRT35M_ROMADDR
;               CLR         A
;               MOVC        A,@A+DPTR
;               MOV         VRTRIM,A
;               MOV         IRCBAND,#01H
;               MOV         A,#0
;               MOV         DPTR,#CLKDIV
;               MOV         P_SW2,#00H

;               ; Select 35MHz
;               MOV         P_SW2,#80H
;               MOV         A,#4
;               MOV         DPTR,#CLKDIV
;               MOV         DPTR,#T35M_ROMADDR
;               CLR         A
;               MOVC        A,@A+DPTR
;               MOV         IRTRIM,A
;               MOV         DPTR,#VRT35M_ROMADDR
;               CLR         A
;               MOVC        A,@A+DPTR
;               MOV         VRTRIM,A
;               MOV         IRCBAND,#01H
;               MOV         A,#0
;               MOV         DPTR,#CLKDIV
;               MOV         P_SW2,#00H

;               ; Select 36.864MHz
;               MOV         P_SW2,#80H
;               MOV         A,#4
;               MOV             DPTR,#CLKDIV
;               MOV             DPTR,#T36M_ROMADDR
;               CLR             A
;               MOVC            A,@A+DPTR
;               MOV             IRTRIM,A
;               MOV             DPTR,#VRT35M_ROMADDR
;               CLR             A
;               MOVC            A,@A+DPTR
;               MOV             VRTRIM,A
;               MOV             IRCBAND,#01H
;               MOV             A,#0
;               MOV             DPTR,#CLKDIV
;               MOV             P_SW2,#00H

                JMP             $

                END
```

~~7.5.12 Read the User-defined internal IRC Frequency (Read from RAM)~~

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     CLKDIV          (*(unsigned char volatile xdata *)0xfe01)

sfr         P_SW2           =       0xba;
sfr         IRTRIM          =       0x9f;

sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

char        *IRC22M;
char        *IRC24M;

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        IRC22M = (char idata *)0xfa;
        IRC24M = (char idata *) 0xfb;
//      IRCCR = *IRC22M;                                    //Load 22.1184MHz IRC parameters
        IRCCR = *IRC24M;                                    //Load 24MHz IRC parameters

        P_SW2 = 0x80;
        CLKDIV = 0;                                         //No division to main clock
        P_SW2 = 0x00;

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P_SW2           DATA            0BAH
CLKDIV          EQU             0FE01H

IRCCR           DATA            09FH

IRC22M          DATA            0FAH
IRC24M          DATA            0FBH

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN

                ORG             0100H
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H
```

```
;               MOV         R0,#IRC22M                  ;Load 22.1184MHz IRC parameters
;               MOV         IRCCR,@R0
                MOV         R0,#IRC24M                  ;Load 24MHz IRC parameters
                MOV         IRCCR,@R0

                MOV         P_SW2,#80H
                MOV         A,#0                        ;No division to main clock
                MOV         DPTR,#CLKDIV
                MOVX        @DPTR,A
                MOV         P_SW2,#00H

                JMP         $

                END
```

8 Special Function Registers

8.1 STC8H1K08 family

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8H								RSTCFG
F0H	B					IAP_TPS		
E8H								AUXINTIF
E0H	ACC			DPS	DPL1	DPH1	CMPCR1	CMPCR2
D8H							ADCCFG	
D0H	PSW						T2H	T2L
C8H	P5	P5M1	P5M0			SPSTAT	SPCTL	SPDAT
C0H		WDT_CONTR	IAP_DATA	IAP_ADDRH	IAP_ADDRL	IAP_CMD	IAP_TRIG	IAP_CONTR
B8H	IP	SADEN	P_SW2		ADC_CONTR	ADC_RES	ADC_RESL	
B0H	P3	P3M1	P3M0			IP2	IP2H	IPH
A8H	IE	SADDR	WKTCL	WKTCH			TA	IE2
A0H			P_SW1					
98H	SCON	SBUF	S2CON	S2BUF		IRCBAND	LIRTRIM	IRTRIM
90H	P1	P1M1	P1M0					
88H	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR	INTCLKO
80H		SP	DPL	DPH				PCON

Bit addressable

Not bit addressable

Note: The register address can be bit-addressable only if it is divisible by 8, and it cannot be bit-addressable if it is not divisible by 8.

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
FEF8H	PWMB_CCR6L	PWMB_CCR7H	PWMB_CCR7L	PWMB_CCR8H	PWMB_CCR8L	PWMB_BKR	PWMB_DTR	PWMB_OISR
FEF0H	PWMB_PSCRH	PWMB_PSCRL	PWMB_ARRH	PWMB_ARRL	PWMB_RCR	PWMB_CCR5H	PWMB_CCR5L	PWMB_CCR6H
FEE8H	PWMB_CCMR1	PWMB_CCMR2	PWMB_CCMR3	PWMB_CCMR4	PWMB_CCER1	PWMB_CCER2	PWMB_CNTRH	PWMB_CNTRL
FEE0H	PWMB_CR1	PWMB_CR2	PWMB_SMCR	PWMB_ETR	PWMB_IER	PWMB_SR1	PWMB_SR2	PWMB_EGR
FED8H	PWMA_CCR2L	PWMA_CCR3H	PWMA_CCR3L	PWMA_CCR4H	PWMA_CCR4L	PWMA_BKR	PWMA_DTR	PWMA_OISR
FED0H	PWMA_PSCRH	PWMA_PSCRL	PWMA_ARRH	PWMA_ARRL	PWMA_RCR	PWMA_CCR1H	PWMA_CCR1L	PWMA_CCR2H
FEC8H	PWMA_CCMR1	PWMA_CCMR2	PWMA_CCMR3	PWMA_CCMR4	PWMA_CCER1	PWMA_CCER2	PWMA_CNTRH	PWMA_CNTRL
FEC0H	PWMA_CR1	PWMA_CR2	PWMA_SMCR	PWMA_ETR	PWMA_IER	PWMA_SR1	PWMA_SR2	PWMA_EGR
FEB0H	PWMA_ETRPS	PWMA_ENO	PWMA_PS	PWMA_IOAUX	PWMB_ETRPS	PWMB_ENO	PWMB_PS	PWMB_IOAUX
FEA8H	ADCTIM							
FEA0H			TM2PS					
FE88H	I2CMSAUX							
FE80H	I2CCFG	I2CMSCR	I2CMSST	I2CSLCR	I2CSLST	I2CSLADR	I2CTxD	I2CRxD
FE30H		P1IE		P3IE				
FE28H		P1DR		P3DR		P5DR		
FE20H		P1SR		P3SR		P5SR		
FE18H		P1NCS		P3NCS		P5NCS		
FE10H		P1PU		P3PU		P5PU		
FE00H	CKSEL	CLKDIV	HIRCCR	XOSCCR	IRC32KCR	MCLKOCR	IRCDB	

8.2 STC8H1K28-32PIN family

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8H								RSTCFG
F0H	B					IAP_TPS		
E8H								AUXINTIF
E0H	ACC			DPS	DPL1	DPH1	CMPCR1	CMPCR2
D8H							ADCCFG	
D0H	PSW	T4T3M	T4H	T4L	T3H	T3L	T2H	T2L
C8H	P5	P5M1	P5M0			SPSTAT	SPCTL	SPDAT
C0H		WDT_CONTR	IAP_DATA	IAP_ADDRH	IAP_ADDRL	IAP_CMD	IAP_TRIG	IAP_CONTR
B8H	IP	SADEN	P_SW2		ADC_CONTR	ADC_RES	ADC_RESL	
B0H	P3	P3M1	P3M0			IP2	IP2H	IPH
A8H	IE	SADDR	WKTCL	WKTCH			TA	IE2
A0H	P2		P_SW1					
98H	SCON	SBUF	S2CON	S2BUF		IRCBAND	LIRTRIM	IRTRIM
90H	P1	P1M1	P1M0	P0M1	P0M0	P2M1	P2M0	
88H	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR	INTCLKO
80H	P0	SP	DPL	DPH				PCON

Bit addressable

Not bit addressable

Note: The register address can be bit-addressable only if it is divisible by 8, and it cannot be bit-addressable if it is not divisible by 8.

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
FEF8H	PWMB_CCR6L	PWMB_CCR7H	PWMB_CCR7L	PWMB_CCR8H	PWMB_CCR8L	PWMB_BKR	PWMB_DTR	PWMB_OISR
FEF0H	PWMB_PSCRH	PWMB_PSCRL	PWMB_ARRH	PWMB_ARRL	PWMB_RCR	PWMB_CCR5H	PWMB_CCR5L	PWMB_CCR6H
FEE8H	PWMB_CCMR1	PWMB_CCMR2	PWMB_CCMR3	PWMB_CCMR4	PWMB_CCER1	PWMB_CCER2	PWMB_CNTRH	PWMB_CNTRL
FEE0H	PWMB_CR1	PWMB_CR2	PWMB_SMCR	PWMB_ETR	PWMB_IER	PWMB_SR1	PWMB_SR2	PWMB_EGR
FED8H	PWMA_CCR2L	PWMA_CCR3H	PWMA_CCR3L	PWMA_CCR4H	PWMA_CCR4L	PWMA_BKR	PWMA_DTR	PWMA_OISR
FED0H	PWMA_PSCRH	PWMA_PSCRL	PWMA_ARRH	PWMA_ARRL	PWMA_RCR	PWMA_CCR1H	PWMA_CCR1L	PWMA_CCR2H
FEC8H	PWMA_CCMR1	PWMA_CCMR2	PWMA_CCMR3	PWMA_CCMR4	PWMA_CCER1	PWMA_CCER2	PWMA_CNTRH	PWMA_CNTRL
FEC0H	PWMA_CR1	PWMA_CR2	PWMA_SMCR	PWMA_ETR	PWMA_IER	PWMA_SR1	PWMA_SR2	PWMA_EGR
FEB0H	PWMA_ETRPS	PWMA_ENO	PWMA_PS	PWMA_IOAUX	PWMB_ETRPS	PWMB_ENO	PWMB_PS	PWMB_IOAUX
FEA8H	ADCTIM							
FEA0H			TM2PS	TM3PS	TM4PS			
FE88H	I2CMSAUX							
FE80H	I2CCFG	I2CMSCR	I2CMSST	I2CSLCR	I2CSLST	I2CSLADR	I2CTxD	I2CRxD
FE30H	P0IE	P1IE						
FE28H	P0DR	P1DR	P2DR	P3DR		P5DR		
FE20H	P0SR	P1SR	P2SR	P3SR		P5SR		
FE18H	P0NCS	P1NCS	P2NCS	P3NCS		P5NCS		
FE10H	P0PU	P1PU	P2PU	P3PU		P5PU		
FE00H	CKSEL	CLKDIV	HIRCCR	XOSCCR	IRC32KCR	MCLKOCR	IRCDB	

8.3 STC8H3K64S4 family

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8H	P7							RSTCFG
F0H	B					IAP_TPS		
E8H	P6						IP3H	AUXINTIF
E0H	ACC	P7M1	P7M0	DPS	DPL1	DPH1	CMPCR1	CMPCR2
D8H							ADCCFG	IP3
D0H	PSW	T4T3M	T4H	T4L	T3H	T3L	T2H	T2L
C8H	P5	P5M1	P5M0	P6M1	P6M0	SPSTAT	SPCTL	SPDAT
C0H	P4	WDT_CONTR	IAP_DATA	IAP_ADDRH	IAP_ADDRL	IAP_CMD	IAP_TRIG	IAP_CONTR
B8H	IP	SADEN	P_SW2		ADC_CONTR	ADC_RES	ADC_RESL	
B0H	P3	P3M1	P3M0	P4M1	P4M0	IP2	IP2H	IPH
A8H	IE	SADDR	WKTCL	WKTCH	S3CON	S3BUF	TA	IE2
A0H	P2	BUS_SPEED	P_SW1					
98H	SCON	SBUF	S2CON	S2BUF		IRCBAND	LIRTRIM	IRTRIM
90H	P1	P1M1	P1M0	P0M1	P0M0	P2M1	P2M0	
88H	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR	INTCLKO
80H	P0	SP	DPL	DPH	S4CON	S4BUF		PCON

Bit addressable

Not bit addressable

Note: The register address can be bit-addressable only if it is divisible by 8, and it cannot be bit-addressable if it is not divisible by 8.

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
FEF8H	PWMB_CCR6L	PWMB_CCR7H	PWMB_CCR7L	PWMB_CCR8H	PWMB_CCR8L	PWMB_BKR	PWMB_DTR	PWMB_OISR
FEF0H	PWMB_PSCRH	PWMB_PSCRL	PWMB_ARRH	PWMB_ARRL	PWMB_RCR	PWMB_CCR5H	PWMB_CCR5L	PWMB_CCR6H
FEE8H	PWMB_CCMR1	PWMB_CCMR2	PWMB_CCMR3	PWMB_CCMR4	PWMB_CCER1	PWMB_CCER2	PWMB_CNTRH	PWMB_CNTRL
FEE0H	PWMB_CR1	PWMB_CR2	PWMB_SMCR	PWMB_ETR	PWMB_IER	PWMB_SR1	PWMB_SR2	PWMB_EGR
FED8H	PWMA_CCR2L	PWMA_CCR3H	PWMA_CCR3L	PWMA_CCR4H	PWMA_CCR4L	PWMA_BKR	PWMA_DTR	PWMA_OISR
FED0H	PWMA_PSCRH	PWMA_PSCRL	PWMA_ARRH	PWMA_ARRL	PWMA_RCR	PWMA_CCR1H	PWMA_CCR1L	PWMA_CCR2H
FEC8H	PWMA_CCMR1	PWMA_CCMR2	PWMA_CCMR3	PWMA_CCMR4	PWMA_CCER1	PWMA_CCER2	PWMA_CNTRH	PWMA_CNTRL
FEC0H	PWMA_CR1	PWMA_CR2	PWMA_SMCR	PWMA_ETR	PWMA_IER	PWMA_SR1	PWMA_SR2	PWMA_EGR
FEB0H	PWMA_ETRPS	PWMA_ENO	PWMA_PS	PWMA_IOAUX	PWMB_ETRPS	PWMB_ENO	PWMB_PS	PWMB_IOAUX
FEA8H	ADCTIM				T3T4PIN			
FEA0H			TM2PS	TM3PS	TM4PS			
FE88H	I2CMSAUX							
FE80H	I2CCFG	I2CMSCR	I2CMSST	I2CSLCR	I2CSLST	I2CSLADR	I2CTxD	I2CRxD
FE30H	P0IE	P1IE						
FE28H	P0DR	P1DR	P2DR	P3DR	P4DR	P5DR	P6DR	P7DR
FE20H	P0SR	P1SR	P2SR	P3SR	P4SR	P5SR	P6SR	P7SR
FE18H	P0NCS	P1NCS	P2NCS	P3NCS	P4NCS	P5NCS	P6NCS	P7NCS
FE10H	P0PU	P1PU	P2PU	P3PU	P4PU	P5PU	P6PU	P7PU
FE08H	SPFUNC	RSTFLAG						
FE00H	CKSEL	CLKDIV	HIRCCR	XOSCCR	IRC32KCR	MCLKOCR	IRCDB	IRC48MCR
FD40H	P0WKUE	P1WKUE	P2WKUE	P3WKUE	P4WKUE	P5WKUE	P6WKUE	P7WKUE
FD30H	P0IM1	P1IM1	P2IM1	P3IM1	P4IM1	P5IM1	P6IM1	P7IM1
FD20H	P0IM0	P1IM0	P2IM0	P3IM0	P4IM0	P5IM0	P6IM0	P7IM0
FD10H	P0INTF	P1INTF	P2INTF	P3INTF	P4INTF	P5INTF	P6INTF	P7INTF
FD00H	P0INTE	P1INTE	P2INTE	P3INTE	P4INTE	P5INTE	P6INTE	P7INTE
FCF0H	MD3	MD2	MD1	MD0	MD5	MD4	ARCON	OPCON

8.4 STC8H3K64S2 family

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8H	P7							RSTCFG
F0H	B					IAP_TPS		
E8H	P6						IP3H	AUXINTIF
E0H	ACC	P7M1	P7M0	DPS	DPL1	DPH1	CMPCR1	CMPCR2
D8H							ADCCFG	IP3
D0H	PSW	T4T3M	T4H	T4L	T3H	T3L	T2H	T2L
C8H	P5	P5M1	P5M0	P6M1	P6M0	SPSTAT	SPCTL	SPDAT
C0H	P4	WDT_CONTR	IAP_DATA	IAP_ADDRH	IAP_ADDRL	IAP_CMD	IAP_TRIG	IAP_CONTR
B8H	IP	SADEN	P_SW2		ADC_CONTR	ADC_RES	ADC_RESL	
B0H	P3	P3M1	P3M0	P4M1	P4M0	IP2	IP2H	IPH
A8H	IE	SADDR	WKTCL	WKTCH			TA	IE2
A0H	P2	BUS_SPEED	P_SW1					
98H	SCON	SBUF	S2CON	S2BUF		IRCBAND	LIRTRIM	IRTRIM
90H	P1	P1M1	P1M0	P0M1	P0M0	P2M1	P2M0	
88H	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR	INTCLKO
80H	P0	SP	DPL	DPH				PCON

Bit addressable

Not bit addressable

Note: The register address can be bit-addressable only if it is divisible by 8, and it cannot be bit-addressable if it is not divisible by 8.

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
FEF8H	PWMB_CCR6L	PWMB_CCR7H	PWMB_CCR7L	PWMB_CCR8H	PWMB_CCR8L	PWMB_BKR	PWMB_DTR	PWMB_OISR
FEF0H	PWMB_PSCRH	PWMB_PSCRL	PWMB_ARRH	PWMB_ARRL	PWMB_RCR	PWMB_CCR5H	PWMB_CCR5L	PWMB_CCR6H
FEE8H	PWMB_CCMR1	PWMB_CCMR2	PWMB_CCMR3	PWMB_CCMR4	PWMB_CCER1	PWMB_CCER2	PWMB_CNTRH	PWMB_CNTRL
FEE0H	PWMB_CR1	PWMB_CR2	PWMB_SMCR	PWMB_ETR	PWMB_IER	PWMB_SR1	PWMB_SR2	PWMB_EGR
FED8H	PWMA_CCR2L	PWMA_CCR3H	PWMA_CCR3L	PWMA_CCR4H	PWMA_CCR4L	PWMA_BKR	PWMA_DTR	PWMA_OISR
FED0H	PWMA_PSCRH	PWMA_PSCRL	PWMA_ARRH	PWMA_ARRL	PWMA_RCR	PWMA_CCR1H	PWMA_CCR1L	PWMA_CCR2H
FEC8H	PWMA_CCMR1	PWMA_CCMR2	PWMA_CCMR3	PWMA_CCMR4	PWMA_CCER1	PWMA_CCER2	PWMA_CNTRH	PWMA_CNTRL
FEC0H	PWMA_CR1	PWMA_CR2	PWMA_SMCR	PWMA_ETR	PWMA_IER	PWMA_SR1	PWMA_SR2	PWMA_EGR
FEB0H	PWMA_ETRPS	PWMA_ENO	PWMA_PS	PWMA_IOAUX	PWMB_ETRPS	PWMB_ENO	PWMB_PS	PWMB_IOAUX
FEA8H	ADCTIM				T3T4PIN			
FEA0H			TM2PS	TM3PS	TM4PS			
FE88H	I2CMSAUX							
FE80H	I2CCFG	I2CMSCR	I2CMSST	I2CSLCR	I2CSLST	I2CSLADR	I2CTxD	I2CRxD
FE30H	P0IE	P1IE						
FE28H	P0DR	P1DR	P2DR	P3DR	P4DR	P5DR	P6DR	P7DR
FE20H	P0SR	P1SR	P2SR	P3SR	P4SR	P5SR	P6SR	P7SR
FE18H	P0NCS	P1NCS	P2NCS	P3NCS	P4NCS	P5NCS	P6NCS	P7NCS
FE10H	P0PU	P1PU	P2PU	P3PU	P4PU	P5PU	P6PU	P7PU
FE08H	SPFUNC	RSTFLAG						
FE00H	CKSEL	CLKDIV	HIRCCR	XOSCCR	IRC32KCR	MCLKOCR	IRCDB	IRC48MCR
FD40H	P0WKUE	P1WKUE	P2WKUE	P3WKUE	P4WKUE	P5WKUE	P6WKUE	P7WKUE
FD30H	P0IM1	P1IM1	P2IM1	P3IM1	P4IM1	P5IM1	P6IM1	P7IM1
FD20H	P0IM0	P1IM0	P2IM0	P3IM0	P4IM0	P5IM0	P6IM0	P7IM0
FD10H	P0INTF	P1INTF	P2INTF	P3INTF	P4INTF	P5INTF	P6INTF	P7INTF
FD00H	P0INTE	P1INTE	P2INTE	P3INTE	P4INTE	P5INTE	P6INTE	P7INTE
FCF0H	MD3	MD2	MD1	MD0	MD5	MD4	ARCON	OPCON

8.5 STC8H8K64U-64Pin/48Pin USB family

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8H	P7				USBADR			RSTCFG
F0H	B				USBCON	IAP_TPS		
E8H	P6				USBDAT		IP3H	AUXINTIF
E0H	ACC	P7M1	P7M0	DPS	DPL1	DPH1	CMPCR1	CMPCR2
D8H					USBCLK		ADCCFG	IP3
D0H	PSW	T4T3M	T4H	T4L	T3H	T3L	T2H	T2L
C8H	P5	P5M1	P5M0	P6M1	P6M0	SPSTAT	SPCTL	SPDAT
C0H	P4	WDT_CONTR	IAP_DATA	IAP_ADDRH	IAP_ADDRL	IAP_CMD	IAP_TRIG	IAP_CONTR
B8H	IP	SADEN	P_SW2		ADC_CONTR	ADC_RES	ADC_RESL	
B0H	P3	P3M1	P3M0	P4M1	P4M0	IP2	IP2H	IPH
A8H	IE	SADDR	WKTCL	WKTCH	S3CON	S3BUF	TA	IE2
A0H	P2	BUS_SPEED	P_SW1					
98H	SCON	SBUF	S2CON	S2BUF		IRCBAND	LIRTRIM	IRTRIM
90H	P1	P1M1	P1M0	P0M1	P0M0	P2M1	P2M0	
88H	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR	INTCLKO
80H	P0	SP	DPL	DPH	S4CON	S4BUF		PCON

Bit addressable

Not bit addressable

Note: The register address can be bit-addressable only if it is divisible by 8, and it cannot be bit-addressable if it is not divisible by 8.

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
FEF8H	PWMB_CCR6L	PWMB_CCR7H	PWMB_CCR7L	PWMB_CCR8H	PWMB_CCR8L	PWMB_BKR	PWMB_DTR	PWMB_OISR
FEF0H	PWMB_PSCRH	PWMB_PSCRL	PWMB_ARRH	PWMB_ARRL	PWMB_RCR	PWMB_CCR5H	PWMB_CCR5L	PWMB_CCR6H
FEE8H	PWMB_CCMR1	PWMB_CCMR2	PWMB_CCMR3	PWMB_CCMR4	PWMB_CCER1	PWMB_CCER2	PWMB_CNTRH	PWMB_CNTRL
FEE0H	PWMB_CR1	PWMB_CR2	PWMB_SMCR	PWMB_ETR	PWMB_IER	PWMB_SR1	PWMB_SR2	PWMB_EGR
FED8H	PWMA_CCR2L	PWMA_CCR3H	PWMA_CCR3L	PWMA_CCR4H	PWMA_CCR4L	PWMA_BKR	PWMA_DTR	PWMA_OISR
FED0H	PWMA_PSCRH	PWMA_PSCRL	PWMA_ARRH	PWMA_ARRL	PWMA_RCR	PWMA_CCR1H	PWMA_CCR1L	PWMA_CCR2H
FEC8H	PWMA_CCMR1	PWMA_CCMR2	PWMA_CCMR3	PWMA_CCMR4	PWMA_CCER1	PWMA_CCER2	PWMA_CNTRH	PWMA_CNTRL
FEC0H	PWMA_CR1	PWMA_CR2	PWMA_SMCR	PWMA_ETR	PWMA_IER	PWMA_SR1	PWMA_SR2	PWMA_EGR
FEB0H	PWMA_ETRPS	PWMA_ENO	PWMA_PS	PWMA_IOAUX	PWMB_ETRPS	PWMB_ENO	PWMB_PS	PWMB_IOAUX
FEA8H	ADCTIM				T3T4PIN			
FEA0H			TM2PS	TM3PS	TM4PS			
FE88H	I2CMSAUX							
FE80H	I2CCFG	I2CMSCR	I2CMSST	I2CSLCR	I2CSLST	I2CSLADR	I2CTxD	I2CRxD
FE30H	P0IE	P1IE						
FE28H	P0DR	P1DR	P2DR	P3DR	P4DR	P5DR	P6DR	P7DR
FE20H	P0SR	P1SR	P2SR	P3SR	P4SR	P5SR	P6SR	P7SR
FE18H	P0NCS	P1NCS	P2NCS	P3NCS	P4NCS	P5NCS	P6NCS	P7NCS
FE10H	P0PU	P1PU	P2PU	P3PU	P4PU	P5PU	P6PU	P7PU
FE08H	SPFUNC	RSTFLAG						
FE00H	CKSEL	CLKDIV	HIRCCR	XOSCCR	IRC32KCR	MCLKOCR	IRCDB	IRC48MCR
FCF0H	MD3	MD2	MD1	MD0	MD5	MD4	ARCON	OPCON

8.6 STC8H2K64T family

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8H								RSTCFG
F0H	B					IAP_TPS		
E8H							IP3H	AUXINTIF
E0H	ACC			DPS	DPL1	DPH1	CMPCR1	CMPCR2
D8H							ADCCFG	IP3
D0H	PSW	T4T3M	T4H	T4L	T3H	T3L	T2H	T2L
C8H	P5	P5M1	P5M0			SPSTAT	SPCTL	SPDAT
C0H	P4	WDT_CONTR	IAP_DATA	IAP_ADDRH	IAP_ADDRL	IAP_CMD	IAP_TRIG	IAP_CONTR
B8H	IP	SADEN	P_SW2		ADC_CONTR	ADC_RES	ADC_RESL	
B0H	P3	P3M1	P3M0	P4M1	P4M0	IP2	IP2H	IPH
A8H	IE	SADDR	WKTCL	WKTCH	S3CON	S3BUF	TA	IE2
A0H	P2	BUS_SPEED	P_SW1					
98H	SCON	SBUF	S2CON	S2BUF		IRCBAND	LIRTRIM	IRTRIM
90H	P1	P1M1	P1M0	P0M1	P0M0	P2M1	P2M0	
88H	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR	INTCLKO
80H	P0	SP	DPL	DPH	S4CON	S4BUF		PCON

Bit addressable

Not bit addressable

Note: The register address can be bit-addressable only if it is divisible by 8, and it cannot be bit-addressable if it is not divisible by 8.

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
FEF8H	PWMB_CCR6L	PWMB_CCR7H	PWMB_CCR7L	PWMB_CCR8H	PWMB_CCR8L	PWMB_BKR	PWMB_DTR	PWMB_OISR
FEF0H	PWMB_PSCRH	PWMB_PSCRL	PWMB_ARRH	PWMB_ARRL	PWMB_RCR	PWMB_CCR5H	PWMB_CCR5L	PWMB_CCR6H
FEE8H	PWMB_CCMR1	PWMB_CCMR2	PWMB_CCMR3	PWMB_CCMR4	PWMB_CCER1	PWMB_CCER2	PWMB_CNTRH	PWMB_CNTRL
FEE0H	PWMB_CR1	PWMB_CR2	PWMB_SMCR	PWMB_ETR	PWMB_IER	PWMB_SR1	PWMB_SR2	PWMB_EGR
FED8H	PWMA_CCR2L	PWMA_CCR3H	PWMA_CCR3L	PWMA_CCR4H	PWMA_CCR4L	PWMA_BKR	PWMA_DTR	PWMA_OISR
FED0H	PWMA_PSCRH	PWMA_PSCRL	PWMA_ARRH	PWMA_ARRL	PWMA_RCR	PWMA_CCR1H	PWMA_CCR1L	PWMA_CCR2H
FEC8H	PWMA_CCMR1	PWMA_CCMR2	PWMA_CCMR3	PWMA_CCMR4	PWMA_CCER1	PWMA_CCER2	PWMA_CNTRH	PWMA_CNTRL
FEC0H	PWMA_CR1	PWMA_CR2	PWMA_SMCR	PWMA_ETR	PWMA_IER	PWMA_SR1	PWMA_SR2	PWMA_EGR
FEB0H	PWMA_ETRPS	PWMA_ENO	PWMA_PS	PWMA_IOAUX	PWMB_ETRPS	PWMB_ENO	PWMB_PS	PWMB_IOAUX
FEA8H	ADCTIM				T3T4PIN			
FEA0H			TM2PS	TM3PS	TM4PS			
FE88H	I2CMSAUX							
FE80H	I2CCFG	I2CMSCR	I2CMSST	I2CSLCR	I2CSLST	I2CSLADR	I2CTxD	I2CRxD
FE70H	YEAR	MONTH	DAY	HOUR	MIN	SEC	SSEC	
FE68H	INIYEAR	INIMONTH	INIDAY	INIHOUR	INIMIN	INISEC	INISSEC	
FE60H	RTCCR	RTCCFG	RTCIEN	RTCIF	ALAHOUR	ALAMIN	ALASEC	ALASSEC
FE30H	P0IE	P1IE						
FE28H	P0DR	P1DR	P2DR	P3DR	P4DR	P5DR		
FE20H	P0SR	P1SR	P2SR	P3SR	P4SR	P5SR		
FE18H	P0NCS	P1NCS	P2NCS	P3NCS	P4NCS	P5NCS		
FE10H	P0PU	P1PU	P2PU	P3PU	P4PU	P5PU		
FE08H	X32KCR							
FE00H	CKSEL	CLKDIV	HIRCCR	XOSCCR	IRC32KCR	MCLKOCR	IRCDB	
FD30H	P0IM1	P1IM1	P2IM1	P3IM1	P4IM1	P5IM1		
FD20H	P0IM0	P1IM0	P2IM0	P3IM0	P4IM0	P5IM0		
FD10H	P0INTF	P1INTF	P2INTF	P3INTF	P4INTF	P5INTF		
FD00H	P0INTE	P1INTE	P2INTE	P3INTE	P4INTE	P5INTE		
FCF0H	MD3	MD2	MD1	MD0	MD5	MD4	ARCON	OPCON
FB68H	TSTH12H	TSTH12L	TSTH13H	TSTH13L	TSTH14H	TSTH14L	TSTH15H	TSTH15L
FB60H	TSTH08H	TSTH08L	TSTH09H	TSTH09L	TSTH10H	TSTH10L	TSTH11H	TSTH11L
FB58H	TSTH04H	TSTH04L	TSTH05H	TSTH05L	TSTH06H	TSTH06L	TSTH07H	TSTH07L
FB50H	TSTH00H	TSTH00L	TSTH01H	TSTH01L	TSTH02H	TSTH02L	TSTH03H	TSTH03L
FB48H	TSRT	TSDATH	TSDATL					
FB40H	TSCHEN1	TSCHEN2	TSCFG1	TSCFG2	TSWUTC	TSCTRL	TSSTA1	TSSTA2
FB28H	COM0_DC_H	COM1_DC_H	COM2_DC_H	COM3_DC_H	COM4_DC_H	COM5_DC_H	COM6_DC_H	COM7_DC_H
FB20H	COM0_DC_L	COM1_DC_L	COM2_DC_L	COM3_DC_L	COM4_DC_L	COM5_DC_L	COM6_DC_L	COM7_DC_L
FB18H	COM0_DA_H	COM1_DA_H	COM2_DA_H	COM3DA_H	COM4_DA_H	COM5_DA_H	COM6_DA_H	COM7_DA_H
FB10H	COM0_DA_L	COM1_DA_L	COM2_DA_L	COM3DA_L	COM4_DA_L	COM5_DA_L	COM6_DA_L	COM7_DA_L
FB00H	COMEN	SEGENL	SEGENH	LEDCTRL	LEDCKS			

8.7 STC8H4K64TLR family

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8H								RSTCFG
F0H	B					IAP_TPS		
E8H							IP3H	AUXINTIF
E0H	ACC			DPS	DPL1	DPH1	CMPCR1	CMPCR2
D8H							ADCCFG	IP3
D0H	PSW	T4T3M	T4H	T4L	T3H	T3L	T2H	T2L
C8H	P5	P5M1	P5M0			SPSTAT	SPCTL	SPDAT
C0H	P4	WDT_CONTR	IAP_DATA	IAP_ADDRH	IAP_ADDRL	IAP_CMD	IAP_TRIG	IAP_CONTR
B8H	IP	SADEN	P_SW2		ADC_CONTR	ADC_RES	ADC_RESL	
B0H	P3	P3M1	P3M0	P4M1	P4M0	IP2	IP2H	IPH
A8H	IE	SADDR	WKTCL	WKTCH	S3CON	S3BUF	TA	IE2
A0H	P2	BUS_SPEED	P_SW1					
98H	SCON	SBUF	S2CON	S2BUF		IRCBAND	LIRTRIM	IRTRIM
90H	P1	P1M1	P1M0	P0M1	P0M0	P2M1	P2M0	
88H	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR	INTCLKO
80H	P0	SP	DPL	DPH	S4CON	S4BUF		PCON

Bit addressable

Not bit addressable

Note: The register address can be bit-addressable only if it is divisible by 8, and it cannot be bit-addressable if it is not divisible by 8.

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
FEF8H	PWMB_CCR6L	PWMB_CCR7H	PWMB_CCR7L	PWMB_CCR8H	PWMB_CCR8L	PWMB_BKR	PWMB_DTR	PWMB_OISR
FEF0H	PWMB_PSCRH	PWMB_PSCRL	PWMB_ARRH	PWMB_ARRL	PWMB_RCR	PWMB_CCR5H	PWMB_CCR5L	PWMB_CCR6H
FEE8H	PWMB_CCMR1	PWMB_CCMR2	PWMB_CCMR3	PWMB_CCMR4	PWMB_CCER1	PWMB_CCER2	PWMB_CNTRH	PWMB_CNTRL
FEE0H	PWMB_CR1	PWMB_CR2	PWMB_SMCR	PWMB_ETR	PWMB_IER	PWMB_SR1	PWMB_SR2	PWMB_EGR
FED8H	PWMA_CCR2L	PWMA_CCR3H	PWMA_CCR3L	PWMA_CCR4H	PWMA_CCR4L	PWMA_BKR	PWMA_DTR	PWMA_OISR
FED0H	PWMA_PSCRH	PWMA_PSCRL	PWMA_ARRH	PWMA_ARRL	PWMA_RCR	PWMA_CCR1H	PWMA_CCR1L	PWMA_CCR2H
FEC8H	PWMA_CCMR1	PWMA_CCMR2	PWMA_CCMR3	PWMA_CCMR4	PWMA_CCER1	PWMA_CCER2	PWMA_CNTRH	PWMA_CNTRL
FEC0H	PWMA_CR1	PWMA_CR2	PWMA_SMCR	PWMA_ETR	PWMA_IER	PWMA_SR1	PWMA_SR2	PWMA_EGR
FEB0H	PWMA_ETRPS	PWMA_ENO	PWMA_PS	PWMA_IOAUX	PWMB_ETRPS	PWMB_ENO	PWMB_PS	PWMB_IOAUX
FEA8H	ADCTIM				T3T4PIN	ADCEXCFG	CMPEXCFG	
FEA0H			TM2PS	TM3PS	TM4PS			
FE88H	I2CMSAUX							
FE80H	I2CCFG	I2CMSCR	I2CMSST	I2CSLCR	I2CSLST	I2CSLADR	I2CTxD	I2CRxD
FE70H	YEAR	MONTH	DAY	HOUR	MIN	SEC	SSEC	
FE68H	INIYEAR	INIMONTH	INIDAY	INIHOUR	INIMIN	INISEC	INISSEC	
FE60H	RTCCR	RTCCFG	RTCIEN	RTCIF	ALAHOUR	ALAMIN	ALASEC	ALASSEC
FE50H	LCMIFCFG	LCMIFCFG2	LCMIFCR	LCMIFSTA	LCMIFDATL	LCMIFDATH		
FE30H	P0IE	P1IE				P5IE		
FE28H	P0DR	P1DR	P2DR	P3DR	P4DR	P5DR		
FE20H	P0SR	P1SR	P2SR	P3SR	P4SR	P5SR		
FE18H	P0NCS	P1NCS	P2NCS	P3NCS	P4NCS	P5NCS		
FE10H	P0PU	P1PU	P2PU	P3PU	P4PU	P5PU		
FE08H	X32KCR							
FE00H	CKSEL	CLKDIV	HIRCCR	XOSCCR	IRC32KCR	MCLKOCR	IRCDB	
FD60H	PINIPL	PINIPH						
FD40H	P0WKUE	P1WKUE	P2WKUE	P3WKUE	P4WKUE	P5WKUE		
FD30H	P0IM1	P1IM1	P2IM1	P3IM1	P4IM1	P5IM1		
FD20H	P0IM0	P1IM0	P2IM0	P3IM0	P4IM0	P5IM0		
FD10H	P0INTF	P1INTF	P2INTF	P3INTF	P4INTF	P5INTF		
FD00H	P0INTE	P1INTE	P2INTE	P3INTE	P4INTE	P5INTE		
FCF0H	MD3	MD2	MD1	MD0	MD5	MD4	ARCON	OPCON
FB68H	TSTH12H	TSTH12L	TSTH13H	TSTH13L	TSTH14H	TSTH14L	TSTH15H	TSTH15L
FB60H	TSTH08H	TSTH08L	TSTH09H	TSTH09L	TSTH10H	TSTH10L	TSTH11H	TSTH11L
FB58H	TSTH04H	TSTH04L	TSTH05H	TSTH05L	TSTH06H	TSTH06L	TSTH07H	TSTH07L
FB50H	TSTH00H	TSTH00L	TSTH01H	TSTH01L	TSTH02H	TSTH02L	TSTH03H	TSTH03L
FB48H	TSRT	TSDATH	TSDATL					
FB40H	TSCHEN1	TSCHEN2	TSCFG1	TSCFG2	TSWUTC	TSCTRL	TSSTA1	TSSTA2
FB28H	COM0_DC_H	COM1_DC_H	COM2_DC_H	COM3_DC_H	COM4_DC_H	COM5_DC_H	COM6_DC_H	COM7_DC_H
FB20H	COM0_DC_L	COM1_DC_L	COM2_DC_L	COM3_DC_L	COM4_DC_L	COM5_DC_L	COM6_DC_L	COM7_DC_L
FB18H	COM0_DA_H	COM1_DA_H	COM2_DA_H	COM3DA_H	COM4_DA_H	COM5_DA_H	COM6_DA_H	COM7_DA_H
FB10H	COM0_DA_L	COM1_DA_L	COM2_DA_L	COM3DA_L	COM4_DA_L	COM5_DA_L	COM6_DA_L	COM7_DA_L
FB00H	COMEN	SEGENL	SEGENH	LEDCTRL	LEDCKS			
FA78H	DMA_LCM_RXAL							
FA70H	DMA_LCM_CFG	DMA_LCM_CR	DMA_LCM_STA	DMA_LCM_AMT	DMA_LCM_DONE	DMA_LCM_TXAH	DMA_LCM_TXAL	DMA_LCM_RXAH
FA68H	DMA_UR4R_CFG	DMA_UR4R_CR	DMA_UR4R_STA	DMA_UR4R_AMT	DMA_UR4R_DONE	DMA_UR4R_RXAH	DMA_UR4R_RXAL	
FA60H	DMA_UR4T_CFG	DMA_UR4T_CR	DMA_UR4T_STA	DMA_UR4T_AMT	DMA_UR4T_DONE	DMA_UR4T_TXAH	DMA_UR4T_TXAL	
FA58H	DMA_UR3R_CFG	DMA_UR3R_CR	DMA_UR3R_STA	DMA_UR3R_AMT	DMA_UR3R_DONE	DMA_UR3R_RXAH	DMA_UR3R_RXAL	
FA50H	DMA_UR3T_CFG	DMA_UR3T_CR	DMA_UR3T_STA	DMA_UR3T_AMT	DMA_UR3T_DONE	DMA_UR3T_TXAH	DMA_UR3T_TXAL	

FA48H	DMA_UR2R_CFG	DMA_UR2R_CR	DMA_UR2R_STA	DMA_UR2R_AMT	DMA_UR2R_DONE	DMA_UR2R_RXAH	DMA_UR2R_RXAL	
FA40H	DMA_UR2T_CFG	DMA_UR2T_CR	DMA_UR2T_STA	DMA_UR2T_AMT	DMA_UR2T_DONE	DMA_UR2T_TXAH	DMA_UR2T_TXAL	
FA38H	DMA_UR1R_CFG	DMA_UR1R_CR	DMA_UR1R_STA	DMA_UR1R_AMT	DMA_UR1R_DONE	DMA_UR1R_RXAH	DMA_UR1R_RXAL	
FA30H	DMA_UR1T_CFG	DMA_UR1T_CR	DMA_UR1T_STA	DMA_UR1T_AMT	DMA_UR1T_DONE	DMA_UR1T_TXAH	DMA_UR1T_TXAL	
FA28H	DMA_SPI_RXAL	DMA_SPI_CFG2						
FA20H	DMA_SPI_CFG	DMA_SPI_CR	DMA_SPI_STA	DMA_SPI_AMT	DMA_SPI_DONE	DMA_SPI_TXAH	DMA_SPI_TXAL	DMA_SPI_RXAH
FA18H	DMA_ADC_RXAL	DMA_ADC_CFG2	DMA_ADC_CHSW0	DMA_ADC_CHSW1				
FA10H	DMA_ADC_CFG	DMA_ADC_CR	DMA_ADC_STA					DMA_ADC_RXAH
FA08H	DMA_M2M_RXAL							
FA00H	DMA_M2M_CFG	DMA_M2M_CR	DMA_M2M_STA	DMA_M2M_AMT	DMA_M2M_DONE	DMA_M2M_TXAH	DMA_M2M_TXAL	DMA_M2M_RXAH

8.8 STC8H4K64TLCD family

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8H	P7							RSTCFG
F0H	B					IAP_TPS		
E8H	P6						IP3H	AUXINTIF
E0H	ACC	P7M1	P7M0	DPS	DPL1	DPH1	CMPCR1	CMPCR2
D8H							ADCCFG	IP3
D0H	PSW	T4T3M	T4H	T4L	T3H	T3L	T2H	T2L
C8H	P5	P5M1	P5M0	P6M1	P6M0	SPSTAT	SPCTL	SPDAT
C0H	P4	WDT_CONTR	IAP_DATA	IAP_ADDRH	IAP_ADDRL	IAP_CMD	IAP_TRIG	IAP_CONTR
B8H	IP	SADEN	P_SW2		ADC_CONTR	ADC_RES	ADC_RESL	
B0H	P3	P3M1	P3M0	P4M1	P4M0	IP2	IP2H	IPH
A8H	IE	SADDR	WKTCL	WKTCH	S3CON	S3BUF	TA	IE2
A0H	P2	BUS_SPEED	P_SW1					
98H	SCON	SBUF	S2CON	S2BUF		IRCBAND	LIRTRIM	IRTRIM
90H	P1	P1M1	P1M0	P0M1	P0M0	P2M1	P2M0	
88H	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR	INTCLKO
80H	P0	SP	DPL	DPH	S4CON	S4BUF		PCON

↑ Bit addressable

Not bit addressable

Note: The register address can be bit-addressable only if it is divisible by 8, and it cannot be bit-addressable if it is not divisible by 8.

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
FEF8H	PWMB_CCR6L	PWMB_CCR7H	PWMB_CCR7L	PWMB_CCR8H	PWMB_CCR8L	PWMB_BKR	PWMB_DTR	PWMB_OISR
FEF0H	PWMB_PSCRH	PWMB_PSCRL	PWMB_ARRH	PWMB_ARRL	PWMB_RCR	PWMB_CCR5H	PWMB_CCR5L	PWMB_CCR6H
FEE8H	PWMB_CCMR1	PWMB_CCMR2	PWMB_CCMR3	PWMB_CCMR4	PWMB_CCER1	PWMB_CCER2	PWMB_CNTRH	PWMB_CNTRL
FEE0H	PWMB_CR1	PWMB_CR2	PWMB_SMCR	PWMB_ETR	PWMB_IER	PWMB_SR1	PWMB_SR2	PWMB_EGR
FED8H	PWMA_CCR2L	PWMA_CCR3H	PWMA_CCR3L	PWMA_CCR4H	PWMA_CCR4L	PWMA_BKR	PWMA_DTR	PWMA_OISR
FED0H	PWMA_PSCRH	PWMA_PSCRL	PWMA_ARRH	PWMA_ARRL	PWMA_RCR	PWMA_CCR1H	PWMA_CCR1L	PWMA_CCR2H
FEC8H	PWMA_CCMR1	PWMA_CCMR2	PWMA_CCMR3	PWMA_CCMR4	PWMA_CCER1	PWMA_CCER2	PWMA_CNTRH	PWMA_CNTRL
FEC0H	PWMA_CR1	PWMA_CR2	PWMA_SMCR	PWMA_ETR	PWMA_IER	PWMA_SR1	PWMA_SR2	PWMA_EGR
FEB0H	PWMA_ETRPS	PWMA_ENO	PWMA_PS	PWMA_IOAUX	PWMB_ETRPS	PWMB_ENO	PWMB_PS	PWMB_IOAUX
FEA8H	ADCTIM				T3T4PIN	ADCEXCFG	CMPEXCFG	
FEA0H			TM2PS	TM3PS	TM4PS			
FE88H	I2CMSAUX							
FE80H	I2CCFG	I2CMSCR	I2CMSST	I2CSLCR	I2CSLST	I2CSLADR	I2CTxD	I2CRxD
FE70H	YEAR	MONTH	DAY	HOUR	MIN	SEC	SSEC	
FE68H	INIYEAR	INIMONTH	INIDAY	INIHOUR	INIMIN	INISEC	INISSEC	
FE60H	RTCCR	RTCCFG	RTCIEN	RTCIF	ALAHOUR	ALAMIN	ALASEC	ALASSEC
FE30H	P0IE	P1IE	P2IE	P3IE	P4IE	P5IE	P6IE	P7IE
FE28H	P0DR	P1DR	P2DR	P3DR	P4DR	P5DR	P6DR	P7DR
FE20H	P0SR	P1SR	P2SR	P3SR	P4SR	P5SR	P6SR	P7SR
FE18H	P0NCS	P1NCS	P2NCS	P3NCS	P4NCS	P5NCS	P6NCS	P7NCS
FE10H	P0PU	P1PU	P2PU	P3PU	P4PU	P5PU	P6PU	P7PU
FE08H	X32KCR							
FE00H	CKSEL	CLKDIV	HIRCCR	XOSCCR	IRC32KCR	MCLKOCR	IRCDB	
FD60H	PINIPL	PINIPH						
FD40H	P0WKUE	P1WKUE	P2WKUE	P3WKUE	P4WKUE	P5WKUE	P6WKUE	P7WKUE
FD30H	P0IM1	P1IM1	P2IM1	P3IM1	P4IM1	P5IM1	P6IM1	P7IM1
FD20H	P0IM0	P1IM0	P2IM0	P3IM0	P4IM0	P5IM0	P6IM0	P7IM0
FD10H	P0INTF	P1INTF	P2INTF	P3INTF	P4INTF	P5INTF	P6INTF	P7INTF
FD00H	P0INTE	P1INTE	P2INTE	P3INTE	P4INTE	P5INTE	P6INTE	P7INTE
FCF0H	MD3	MD2	MD1	MD0	MD5	MD4	ARCON	OPCON
FBA8H	C3SEGV0	C3SEGV1	C3SEGV2	C3SEGV3	C3SEGV4			
FBA0H	C2SEGV0	C2SEGV1	C2SEGV2	C2SEGV3	C2SEGV4			
FB98H	C1SEGV0	C1SEGV1	C1SEGV2	C1SEGV3	C1SEGV4			
FB90H	C0SEGV0	C0SEGV1	C0SEGV2	C0SEGV3	C0SEGV4			
FB88H	COMON		SEGON1	SEGON2	SEGON3	SEGON4	SEGON5	
FB80H	LCDCFG	LCDCFG2	DBLEN	COMLENL	COMLENM	COMLENH	BLINKRATE	LCDCR
FB68H	TSTH12H	TSTH12L	TSTH13H	TSTH13L	TSTH14H	TSTH14L	TSTH15H	TSTH15L
FB60H	TSTH08H	TSTH08L	TSTH09H	TSTH09L	TSTH10H	TSTH10L	TSTH11H	TSTH11L
FB58H	TSTH04H	TSTH04L	TSTH05H	TSTH05L	TSTH06H	TSTH06L	TSTH07H	TSTH07L
FB50H	TSTH00H	TSTH00L	TSTH01H	TSTH01L	TSTH02H	TSTH02L	TSTH03H	TSTH03L
FB48H	TSRT	TSDATH	TSDATL					
FB40H	TSCHEN1	TSCHEN2	TSCFG1	TSCFG2	TSWUTC	TSCTRL	TSSTA1	TSSTA2
FA78H	DMA_LCM_RXAL							
FA70H	DMA_LCM_CFG	DMA_LCM_CR	DMA_LCM_STA	DMA_LCM_AMT	DMA_LCM_DONE	DMA_LCM_TXAH	DMA_LCM_TXAL	DMA_LCM_RXAH
FA68H	DMA_UR4R_CFG	DMA_UR4R_CR	DMA_UR4R_STA	DMA_UR4R_AMT	DMA_UR4R_DONE	DMA_UR4R_RXAH	DMA_UR4R_RXAL	
FA60H	DMA_UR4T_CFG	DMA_UR4T_CR	DMA_UR4T_STA	DMA_UR4T_AMT	DMA_UR4T_DONE	DMA_UR4T_TXAH	DMA_UR4T_TXAL	
FA58H	DMA_UR3R_CFG	DMA_UR3R_CR	DMA_UR3R_STA	DMA_UR3R_AMT	DMA_UR3R_DONE	DMA_UR3R_RXAH	DMA_UR3R_RXAL	

FA50H	DMA_UR3T_CFG	DMA_UR3T_CR	DMA_UR3T_STA	DMA_UR3T_AMT	DMA_UR3T_DONE	DMA_UR3T_TXAH	DMA_UR3T_TXAL	
FA48H	DMA_UR2R_CFG	DMA_UR2R_CR	DMA_UR2R_STA	DMA_UR2R_AMT	DMA_UR2R_DONE	DMA_UR2R_RXAH	DMA_UR2R_RXAL	
FA40H	DMA_UR2T_CFG	DMA_UR2T_CR	DMA_UR2T_STA	DMA_UR2T_AMT	DMA_UR2T_DONE	DMA_UR2T_TXAH	DMA_UR2T_TXAL	
FA38H	DMA_UR1R_CFG	DMA_UR1R_CR	DMA_UR1R_STA	DMA_UR1R_AMT	DMA_UR1R_DONE	DMA_UR1R_RXAH	DMA_UR1R_RXAL	
FA30H	DMA_UR1T_CFG	DMA_UR1T_CR	DMA_UR1T_STA	DMA_UR1T_AMT	DMA_UR1T_DONE	DMA_UR1T_TXAH	DMA_UR1T_TXAL	
FA28H	DMA_SPI_RXAL	DMA_SPI_CFG2						
FA20H	DMA_SPI_CFG	DMA_SPI_CR	DMA_SPI_STA	DMA_SPI_AMT	DMA_SPI_DONE	DMA_SPI_TXAH	DMA_SPI_TXAL	DMA_SPI_RXAH
FA18H	DMA_ADC_RXAL	DMA_ADC_CFG2	DMA_ADC_CHSW0	DMA_ADC_CHSW1				
FA10H	DMA_ADC_CFG	DMA_ADC_CR	DMA_ADC_STA					DMA_ADC_RXAH
FA08H	DMA_M2M_RXAL							
FA00H	DMA_M2M_CFG	DMA_M2M_CR	DMA_M2M_STA	DMA_M2M_AMT	DMA_M2M_DONE	DMA_M2M_TXAH	DMA_M2M_TXAL	DMA_M2M_RXAH

8.9 STC8H4K64LCD family

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8H	P7							RSTCFG
F0H	B					IAP_TPS		
E8H	P6						IP3H	AUXINTIF
E0H	ACC	P7M1	P7M0	DPS	DPL1	DPH1	CMPCR1	CMPCR2
D8H							ADCCFG	IP3
D0H	PSW	T4T3M	T4H	T4L	T3H	T3L	T2H	T2L
C8H	P5	P5M1	P5M0	P6M1	P6M0	SPSTAT	SPCTL	SPDAT
C0H	P4	WDT_CONTR	IAP_DATA	IAP_ADDRH	IAP_ADDRL	IAP_CMD	IAP_TRIG	IAP_CONTR
B8H	IP	SADEN	P_SW2		ADC_CONTR	ADC_RES	ADC_RESL	
B0H	P3	P3M1	P3M0	P4M1	P4M0	IP2	IP2H	IPH
A8H	IE	SADDR	WKTCL	WKTCH	S3CON	S3BUF	TA	IE2
A0H	P2	BUS_SPEED	P_SW1					
98H	SCON	SBUF	S2CON	S2BUF		IRCBAND	LIRTRIM	IRTRIM
90H	P1	P1M1	P1M0	P0M1	P0M0	P2M1	P2M0	
88H	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR	INTCLKO
80H	P0	SP	DPL	DPH	S4CON	S4BUF		PCON

Bit addressable

Not bit addressable

Note: The register address can be bit-addressable only if it is divisible by 8, and it cannot be bit-addressable if it is not divisible by 8.

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
FEF8H	PWMB_CCR6L	PWMB_CCR7H	PWMB_CCR7L	PWMB_CCR8H	PWMB_CCR8L	PWMB_BKR	PWMB_DTR	PWMB_OISR
FEF0H	PWMB_PSCRH	PWMB_PSCRL	PWMB_ARRH	PWMB_ARRL	PWMB_RCR	PWMB_CCR5H	PWMB_CCR5L	PWMB_CCR6H
FEE8H	PWMB_CCMR1	PWMB_CCMR2	PWMB_CCMR3	PWMB_CCMR4	PWMB_CCER1	PWMB_CCER2	PWMB_CNTRH	PWMB_CNTRL
FEE0H	PWMB_CR1	PWMB_CR2	PWMB_SMCR	PWMB_ETR	PWMB_IER	PWMB_SR1	PWMB_SR2	PWMB_EGR
FED8H	PWMA_CCR2L	PWMA_CCR3H	PWMA_CCR3L	PWMA_CCR4H	PWMA_CCR4L	PWMA_BKR	PWMA_DTR	PWMA_OISR
FED0H	PWMA_PSCRH	PWMA_PSCRL	PWMA_ARRH	PWMA_ARRL	PWMA_RCR	PWMA_CCR1H	PWMA_CCR1L	PWMA_CCR2H
FEC8H	PWMA_CCMR1	PWMA_CCMR2	PWMA_CCMR3	PWMA_CCMR4	PWMA_CCER1	PWMA_CCER2	PWMA_CNTRH	PWMA_CNTRL
FEC0H	PWMA_CR1	PWMA_CR2	PWMA_SMCR	PWMA_ETR	PWMA_IER	PWMA_SR1	PWMA_SR2	PWMA_EGR
FEB0H	PWMA_ETRPS	PWMA_ENO	PWMA_PS	PWMA_IOAUX	PWMB_ETRPS	PWMB_ENO	PWMB_PS	PWMB_IOAUX
FEA8H	ADCTIM				T3T4PIN	ADCEXCFG	CMPEXCFG	
FEA0H			TM2PS	TM3PS	TM4PS			
FE88H	I2CMSAUX							
FE80H	I2CCFG	I2CMSCR	I2CMSST	I2CSLCR	I2CSLST	I2CSLADR	I2CTxD	I2CRxD
FE70H	YEAR	MONTH	DAY	HOUR	MIN	SEC	SSEC	
FE68H	INIYEAR	INIMONTH	INIDAY	INIHOUR	INIMIN	INISEC	INISSEC	
FE60H	RTCCR	RTCCFG	RTCIEN	RTCIF	ALAHOUR	ALAMIN	ALASEC	ALASSEC
FE30H	P0IE	P1IE	P2IE	P3IE	P4IE	P5IE	P6IE	P7IE
FE28H	P0DR	P1DR	P2DR	P3DR	P4DR	P5DR	P6DR	P7DR
FE20H	P0SR	P1SR	P2SR	P3SR	P4SR	P5SR	P6SR	P7SR
FE18H	P0NCS	P1NCS	P2NCS	P3NCS	P4NCS	P5NCS	P6NCS	P7NCS
FE10H	P0PU	P1PU	P2PU	P3PU	P4PU	P5PU	P6PU	P7PU
FE08H	X32KCR							
FE00H	CKSEL	CLKDIV	HIRCCR	XOSCCR	IRC32KCR	MCLKOCR	IRCDB	
FD60H	PINIPL	PINIPH						
FD40H	P0WKUE	P1WKUE	P2WKUE	P3WKUE	P4WKUE	P5WKUE	P6WKUE	P7WKUE
FD30H	P0IM1	P1IM1	P2IM1	P3IM1	P4IM1	P5IM1	P6IM1	P7IM1
FD20H	P0IM0	P1IM0	P2IM0	P3IM0	P4IM0	P5IM0	P6IM0	P7IM0
FD10H	P0INTF	P1INTF	P2INTF	P3INTF	P4INTF	P5INTF	P6INTF	P7INTF
FD00H	P0INTE	P1INTE	P2INTE	P3INTE	P4INTE	P5INTE	P6INTE	P7INTE
FCF0H	MD3	MD2	MD1	MD0	MD5	MD4	ARCON	OPCON
FBA8H	C3SEGV0	C3SEGV1	C3SEGV2	C3SEGV3	C3SEGV4			
FBA0H	C2SEGV0	C2SEGV1	C2SEGV2	C2SEGV3	C2SEGV4			
FB98H	C1SEGV0	C1SEGV1	C1SEGV2	C1SEGV3	C1SEGV4			
FB90H	C0SEGV0	C0SEGV1	C0SEGV2	C0SEGV3	C0SEGV4			
FB88H	COMON		SEGON1	SEGON2	SEGON3	SEGON4	SEGON5	
FB80H	LCDCFG	LCDCFG2	DBLEN	COMLENL	COMLENM	COMLENH	BLINKRATE	LCDCR
FA78H	DMA_LCM_RXAL							
FA70H	DMA_LCM_CFG	DMA_LCM_CR	DMA_LCM_STA	DMA_LCM_AMT	DMA_LCM_DONE	DMA_LCM_TXAH	DMA_LCM_TXAL	DMA_LCM_RXAH
FA68H	DMA_UR4R_CFG	DMA_UR4R_CR	DMA_UR4R_STA	DMA_UR4R_AMT	DMA_UR4R_DONE	DMA_UR4R_RXAH	DMA_UR4R_RXAL	
FA60H	DMA_UR4T_CFG	DMA_UR4T_CR	DMA_UR4T_STA	DMA_UR4T_AMT	DMA_UR4T_DONE	DMA_UR4T_TXAH	DMA_UR4T_TXAL	
FA58H	DMA_UR3R_CFG	DMA_UR3R_CR	DMA_UR3R_STA	DMA_UR3R_AMT	DMA_UR3R_DONE	DMA_UR3R_RXAH	DMA_UR3R_RXAL	
FA50H	DMA_UR3T_CFG	DMA_UR3T_CR	DMA_UR3T_STA	DMA_UR3T_AMT	DMA_UR3T_DONE	DMA_UR3T_TXAH	DMA_UR3T_TXAL	
FA48H	DMA_UR2R_CFG	DMA_UR2R_CR	DMA_UR2R_STA	DMA_UR2R_AMT	DMA_UR2R_DONE	DMA_UR2R_RXAH	DMA_UR2R_RXAL	
FA40H	DMA_UR2T_CFG	DMA_UR2T_CR	DMA_UR2T_STA	DMA_UR2T_AMT	DMA_UR2T_DONE	DMA_UR2T_TXAH	DMA_UR2T_TXAL	
FA38H	DMA_UR1R_CFG	DMA_UR1R_CR	DMA_UR1R_STA	DMA_UR1R_AMT	DMA_UR1R_DONE	DMA_UR1R_RXAH	DMA_UR1R_RXAL	
FA30H	DMA_UR1T_CFG	DMA_UR1T_CR	DMA_UR1T_STA	DMA_UR1T_AMT	DMA_UR1T_DONE	DMA_UR1T_TXAH	DMA_UR1T_TXAL	
FA28H	DMA_SPI_RXAL	DMA_SPI_CFG2						

FA20H	DMA_SPI_CFG	DMA_SPI_CR	DMA_SPI_STA	DMA_SPI_AMT	DMA_SPI_DONE	DMA_SPI_TXAH	DMA_SPI_TXAL	DMA_SPI_RXAH
FA18H	DMA_ADC_RXAL	DMA_ADC_CFG2	DMA_ADC_CHSW0	DMA_ADC_CHSW1				
FA10H	DMA_ADC_CFG	DMA_ADC_CR	DMA_ADC_STA					DMA_ADC_RXAH
FA08H	DMA_M2M_RXAL							
FA00H	DMA_M2M_CFG	DMA_M2M_CR	DMA_M2M_STA	DMA_M2M_AMT	DMA_M2M_DONE	DMA_M2M_TXAH	DMA_M2M_TXAL	DMA_M2M_RXAH

8.10 List of Special Function Registers

Note: The register address can be bit-addressable only if it is divisible by 8, and it cannot be bit-addressable if it is not divisible by 8.

STC8H can be bit-addressable registers: P0 (80H), TCON (88H), P1 (90H), SCON (98H), P2 (A0H), IE (A8H), P3 (B0H), IP (B8H), P4 (C0H), P5 (C8H), PSW (D0H), ACC (E0H), P6 (E8H), B (F0H), P7 (F8H)

Symbol	Description	Address	Bit Address and Symbol								Value after Reset
			B7	B6	B5	B4	B3	B2	B1	B0	
P0	Port 0	80H	P07	P06	P05	P04	P03	P02	P01	P00	1111,1111
SP	Stack Pointer	81H									0000,0111
DPL	Data pointer low byte register	82H									0000,0000
DPH	Data pointer high byte register	83H									0000,0000
S4CON	UART 4 control register	84H	S4SM0	S4ST4	S4SM2	S4REN	S4TB8	S4RB8	S4TI	S4RI	0000,0000
S4BUF	UART 4 data buffer register	85H									0000,0000
PCON	Power control register	87H	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000
TCON	Timer 0 and 1 control register	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0000,0000
TMOD	Timer 0 and 1 mode register	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	0000,0000
TL0	Timer 0 low byte register	8AH									0000,0000
TL1	Timer 1 low byte register	8BH									0000,0000
TH0	Timer 0 high byte register	8CH									0000,0000
TH1	Timer 1 high byte register	8DH									0000,0000
AUXR	Auxiliary register 1	8EH	T0x12	T1x12	UART_M0x6	T2R	T2_C/T	T2x12	EXTRAM	S1ST2	0000,0001
INTCLKO	External interrupt and clock output control register	8FH	-	EX4	EX3	EX2	-	T2CLKO	T1CLKO	T0CLKO	x000,x000
P1	Port 1	90H	P17	P16	P15	P14	P13	P12	P11	P10	1111,1111
P1M1	Port 1 mode register 1	91H	P17M1	P16M1	P15M1	P14M1	P13M1	P12M1	P11M1	P10M1	1111,1111
P1M0	Port 1 mode register 0	92H	P17M0	P16M0	P15M0	P14M0	P13M0	P12M0	P11M0	P10M0	0000,0000
P0M1	Port 0 mode register 1	93H	P07M1	P06M1	P05M1	P04M1	P03M1	P02M1	P01M1	P00M1	1111,1111
P0M0	Port 0 mode register 0	94H	P07M0	P06M0	P05M0	P04M0	P03M0	P02M0	P01M0	P00M0	0000,0000
P2M1	Port 2 mode register 1	95H	P27M1	P26M1	P25M1	P24M1	P23M1	P22M1	P21M1	P20M1	1111,1111
P2M0	Port 2 mode register 0	96H	P27M0	P26M0	P25M0	P24M0	P23M0	P22M0	P21M0	P20M0	0000,0000
SCON	UART1 control register	98H	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	0000,0000
SBUF	UART1 data buffer register	99H									0000,0000
S2CON	UART2 control register	9AH	S2SM0	-	S2SM2	S2REN	S2TB8	S2RB8	S2TI	S2RI	0x00,0000
S2BUF	UART2 data buffer registe	9BH									0000,0000
IRCBAND	IRC band selection detection	9DH	-	-	-	-	-	-	-	SEL	xxxx,xxxn
LIRTRIM	IRC frequency trim register	9EH	-	-	-	-	-	-	LIRTRIM[1:0]		xxxx,xxnn
IRTRIM	IRC frequency adjustment register	9FH	IRTRIM[7:0]								nnnn,nnnn
P2	Port 2	A0H	P27	P26	P25	P24	P23	P22	P21	P20	1111,1111
BUS_SPEED	Bus speed control register	A1H	RW_S[1:0]					SPEED[2:0]			00xx,x000
P_SW1	Peripheral port switch register 1	A2H	S1_S[1:0]		CCP_S[1:0]		SPI_S[1:0]		0	-	nn00,000x
IE	Interrupt enable register	A8H	EA	ELVD	EADC	ES	ET1	EX1	ET0	EX0	0000,0000
SADDR	UART1 slave address register	A9H									0000,0000
WKTCL	Wake-up Timer Control Register Low Byte	AAH									1111,1111
WKTCH	Wake-up Timer Control Register High Byte	ABH	WKTEN								0111,1111
S3CON	UART3 control register	ACH	S3SM0	S3ST4	S3SM2	S3REN	S3TB8	S3RB8	S3TI	S3RI	0000,0000
S3BUF	UART3 data buffer register	ADH									0000,0000
TA	DPTR Timing control register	AEH									0000,0000
IE2	Interrupt enable register 2	AFH	EUSB	ET4	ET3	ES4	ES3	ET2	ESPI	ES2	x000,0000
P3	Port 3	B0H	P37	P36	P35	P34	P33	P32	P31	P30	1111,1111
P3M1	Port 3 mode register 1	B1H	P37M1	P36M1	P35M1	P34M1	P33M1	P32M1	P31M1	P30M1	1111,1100
P3M0	Port 3 mode register 0	B2H	P37M0	P36M0	P35M0	P34M0	P33M0	P32M0	P31M0	P30M0	0000,0000
P4M1	Port 4 mode register 1	B3H	P47M1	P46M1	P45M1	P44M1	P43M1	P42M1	P41M1	P40M1	1111,1111
P4M0	Port 4 mode register 0	B4H	P47M0	P46M0	P45M0	P44M0	P43M0	P42M0	P41M0	P40M0	0000,0000
IP2	2nd Interrupt Priority register low byte	B5H	PUSB PTKSU	PI2C	PCMP	PX4	PPWMB	PPWMA	PSPI	PS2	0000,0000
IP2H	2nd Interrupt Priority register high byte	B6H	PUSBH PTKSUH	PI2CH	PCMPH	PX4H	PPWMBH	PPWMAH	PSPIH	PS2H	0000,0000
IPH	Interrupt Priority High Byte	B7H	-	PLVDH	PADCH	PSH	PT1H	PX1H	PT0H	PX0H	x000,0000
IP	Interrupt Priority Low Byte	B8H	-	PLVD	PADC	PS	PT1	PX1	PT0	PX0	x000,0000
SADEN	UART1 slave address enable register	B9H									0000,0000
P_SW2	Peripheral port switch register 2	BAH	EAXFR	-	I2C_S[1:0]		CMPO_S	S4_S	S3_S	S2_S	0x00,0000
ADC_CONTR	ADC control register	BCH	ADC_POWER	ADC_START	ADC_FLAG	ADC_EPWMT	ADC_CHS[3:0]				0000,0000
ADC_RES	ADC Result High Byte	BDH									0000,0000
ADC_RESL	ADC Result Low Byte	BEH									0000,0000
P4	Port 4	C0H	P47	P46	P45	P44	P43	P42	P41	P40	1111,1111
WDT_CONTR	Watchdog control register	C1H	WDT_FLAG	-	EN_WDT	CLR_WDT	IDL_WDT	WDT_PS[2:0]			0x00,0000
IAP_DATA	IAP Flash Data Register	C2H									1111,1111
IAP_ADDRH	IAP Flash Address High Byte	C3H									0000,0000
IAP_ADDRL	IAP Flash Address Low Byte	C4H									0000,0000
IAP_CMD	IAP Flash Command Register	C5H	-	-	-	-	-	-	CMD[1:0]		xxxx,xx00
IAP_TRIG	IAP Flash Trigger register	C6H									0000,0000
IAP_CONTR	IAP Control Register	C7H	IAPEN	SWBS	SWRST	CMD_FAIL	-	-	-	-	0000,xxxx
P5	Port 5	C8H	-	-	P55	P54	P53	P52	P51	P50	xx11,1111
P5M1	Port 5 mode register 1	C9H	-	-	P55M1	P54M1	P53M1	P52M1	P51M1	P50M1	xx11,1111
P5M0	Port 5 mode register 0	CAH	-	-	P55M0	P54M0	P53M0	P52M0	P51M0	P50M0	xx00,0000

P6M1	Port 6 mode register 1	CBH	P67M1	P66M1	P65M1	P64M1	P63M1	P62M1	P61M1	P60M1	1111,1111
P6M0	Port 6 mode register 0	CCH	P67M0	P66M0	P65M0	P64M0	P63M0	P62M0	P61M0	P60M0	0000,0000
SPSTAT	SPI Status register	CDH	SPIF	WCOL	-	-	-	-	-	-	00xx,xxxx
SPCTL	SPI Control Register	CEH	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR[1:0]		0000,0100
SPDAT	SPI Data Register	CFH									0000,0000
PSW	Program Status Word Register	D0H	CY	AC	F0	RS1	RS0	OV	F1	P	0000,00x0
T4T3M	Timer4 and Timer 3 Control Register	D1H	T4R	T4_C/T	T4x12	T4CLKO	T3R	T3_C/T	T3x12	T3CLKO	0000,0000
T4H	Timer 4 high byte register	D2H									0000,0000
T4L	Timer 4 low byte register	D3H									0000,0000
T3H	Timer 3 high byte register	D4H									0000,0000
T3L	Timer 3 low byte register	D5H									0000,0000
T2H	Timer 2 high byte register	D6H									0000,0000
T2L	Timer 2 low byte register	D7H									0000,0000
USBCLK	USB clock control register	DCH	ENCKM	PCKI[1:0]		CRE	TST_USB	TST_PHY	PHYTST[1:0]		0010,0000
ADCCFG	ADC Configuration Register	DEH	-	-	RESFMT	-	SPEED[3:0]				xx0x,0000
IP3	3nd Interrupt Priority register low byte	DFH	-	-	-	-	-	PRTC	PS4	PS3	xxxx,x000
ACC	Accumulator	E0H									0000,0000
P7M1	Port 7 mode register 1	E1H	P77M1	P76M1	P75M1	P74M1	P73M1	P72M1	P71M1	P70M1	1111,1111
P7M0	Port 7 mode register 0	E2H	P77M0	P76M0	P75M0	P74M0	P73M0	P72M0	P71M0	P70M0	0000,0000
DPS	DPTR Selection Register	E3H	ID1	ID0	TSL	AU1	AU0	-	-	SEL	0000,0xx0
DPL1	2nd Data pointer low byte	E4H									0000,0000
DPH1	2nd Data pointer high byte	E5H									0000,0000
CMPCR1	Comparator Control Register 1	E6H	CMPEN	CMPIF	PIE	NIE	PIS	NIS	CMPOE	CMPRES	0000,0000
CMPCR2	Comparator Control Register 2	E7H	INVCMPO	DISFLT	LCDTY[5:0]						0000,0000
P6	Port 6	E8H	P67	P66	P65	P64	P63	P62	P61	P60	1111,1111
USBDAT	USB Data register	ECH									0000,0000
IP3H	3nd Interrupt Priority Register High Byte	EEH	-	-	-	-	-	PRTCH	PS4H	PS3H	xxxx,x000
AUXINTIF	Extended External Interrupt Flag Register	EFH	-	INT4IF	INT3IF	INT2IF	-	T4IF	T3IF	T2IF	x000,x000
B	B register	F0H									0000,0000
USBCON	USB Control Register	F4H	ENUSB	USBRST	PS2M	PUEN	PDEN	DFREC	DP	DM	0000,0000
IAP_TPS	IAP Waiting Time Control Register	F5H	-	-	IAPTPS[5:0]						xx00,0000
P7	Port 7	F8H	P77	P76	P75	P74	P73	P72	P71	P70	1111,1111
USBADR	USB Address register	FCH	BUSY	AUTORD	UADR[5:0]						0000,0000
RSTCFG	Reset Configuration Register	FFH	-	ENLVR	-	P54RST	-	-	LVDS[1:0]		xnxn,xxnn

The following special function registers are extended SFRs whose logical addresses are in the XDATA area. Before access them, the highest bit (EAXFR) of the P_SW2 (BAH) register needs to be set, and they can accessed by using the MOVX A, @DPTR and MOVX @ DPTR, A instructions.

Symbol	Description	Address	Bit Address and Symbol								Value after Reset
			B7	B6	B5	B4	B3	B2	B1	B0	
CKSEL	Clock Select Register	FE00H	-	-	-	-	-	-	MCKSEL[1:0]		xxxx,xx00
CLKDIV	Clock Divide Register	FE01H									nnnn,nnnn
HIRCCR	Internal high-speed oscillator Control Register	FE02H	ENHIRC	-	-	-	-	-	-	HIRCST	1xxx,xxx0
XOSCCR	External Oscillator Control Register	FE03H	ENXOSC	XITYPE	-	-	-	-	-	XOSCST	00xx,xxx0
IRC32KCR	Internal 32K Oscillator Control Register	FE04H	ENIRC32K	-	-	-	-	-	-	IRC32KST	0xxx,xxx0
MCLKOCR	Main Clock Output Control Register	FE05H	MCLKO_S	MCLKODIV[6:0]							0000,0000
IRCDB	Internal high-speed oscillator debounce control	FE06H	IRCDB_PAR[7:0]								1000,0000
IRC48MCR	Internal 48M Oscillator Control Register	FE07H	ENIRC48M	-	-	-	-	-	-	IRC48MST	1xxx,xxx0
SPFUNC	Special Function Control Register	FE08H								BKSWR	xxxx,xxx0
RSTFLAG	Reset flag register	FE09H						SWR	ROMOV	EXRST	xxxx,x000
P0PU	P0 Pull-up Resistor Control Register	FE10H	P07PU	P06PU	P05PU	P04PU	P03PU	P02PU	P01PU	P00PU	0000,0000
P1PU	P1 Pull-up Resistor Control Register	FE11H	P17PU	P16PU	P15PU	P14PU	P13PU	P12PU	P11PU	P10PU	0000,0000
P2PU	P2 Pull-up Resistor Control Register	FE12H	P27PU	P26PU	P25PU	P24PU	P23PU	P22PU	P21PU	P20PU	0000,0000
P3PU	P3 Pull-up Resistor Control Register	FE13H	P37PU	P36PU	P35PU	P34PU	P33PU	P32PU	P31PU	P30PU	0000,0000
P4PU	P4 Pull-up Resistor Control Register	FE14H	P47PU	P46PU	P45PU	P44PU	P43PU	P42PU	P41PU	P40PU	0000,0000
P5PU	P5 Pull-up Resistor Control Register	FE15H	-	-	-	P54PU	P53PU	P52PU	P51PU	P50PU	xxx0,0000
P6PU	P6 Pull-up Resistor Control Register	FE16H	P67PU	P66PU	P65PU	P64PU	P63PU	P62PU	P61PU	P60PU	0000,0000
P7PU	P7 Pull-up Resistor Control Register	FE17H	P77PU	P76PU	P75PU	P74PU	P73PU	P72PU	P71PU	P70PU	0000,0000
P0NCS	P0 Schmitt Trigger Control Register	FE18H	P07NCS	P06NCS	P05NCS	P04NCS	P03NCS	P02NCS	P01NCS	P00NCS	0000,0000
P1NCS	P1 Schmitt Trigger Control Register	FE19H	P17NCS	P16NCS	P15NCS	P14NCS	P13NCS	P12NCS	P11NCS	P10NCS	0000,0000
P2NCS	P2 Schmitt Trigger Control Register	FE1AH	P27NCS	P26NCS	P25NCS	P24NCS	P23NCS	P22NCS	P21NCS	P20NCS	0000,0000
P3NCS	P3 Schmitt Trigger Control	FE1BH	P37NCS	P36NCS	P35NCS	P34NCS	P33NCS	P32NCS	P31NCS	P30NCS	0000,0000

	Register										
P4NCS	P4 Schmitt Trigger Control Register	FE1CH	P47NCS	P46NCS	P45NCS	P44NCS	P43NCS	P42NCS	P41NCS	P40NCS	0000,0000
P5NCS	P5 Schmitt Trigger Control Register	FE1DH	-	-	P55NCS	P54NCS	P53NCS	P52NCS	P51NCS	P50NCS	xx00,0000
P6NCS	P6 Schmitt Trigger Control Register	FE1EH	P67NCS	P66NCS	P65NCS	P64NCS	P63NCS	P62NCS	P61NCS	P60NCS	0000,0000
P7NCS	P7 Schmitt Trigger Control Register	FE1FH	P77NCS	P76NCS	P75NCS	P74NCS	P73NCS	P72NCS	P71NCS	P70NCS	0000,0000
P0SR	P0 Level Shift Rate Register	FE20H	P07SR	P06SR	P05SR	P04SR	P03SR	P02SR	P01SR	P00SR	1111,1111
P1SR	P1 Level Shift Rate Register	FE21H	P17SR	P16SR	P15SR	P14SR	P13SR	P12SR	P11SR	P10SR	1111,1111
P2SR	P2 Level Shift Rate Register	FE22H	P27SR	P26SR	P25SR	P24SR	P23SR	P22SR	P21SR	P20SR	1111,1111
P3SR	P3 Level Shift Rate Register	FE23H	P37SR	P36SR	P35SR	P34SR	P33SR	P32SR	P31SR	P30SR	1111,1111
P4SR	P4 Level Shift Rate Register	FE24H	P47SR	P46SR	P45SR	P44SR	P43SR	P42SR	P41SR	P40SR	1111,1111
P5SR	P5 Level Shift Rate Register	FE25H	-	-	P55SR	P54SR	P53SR	P52SR	P51SR	P50SR	xx11,1111
P6SR	P6 Level Shift Rate Register	FE26H	P57SR	P66SR	P65SR	P64SR	P63SR	P62SR	P61SR	P60SR	1111,1111
P7SR	P7 Level Shift Rate Register	FE27H	P77SR	P76SR	P75SR	P74SR	P73SR	P72SR	P71SR	P70SR	1111,1111
P0DR	P0 Drive Current Control Register	FE28H	P07DR	P06DR	P05DR	P04DR	P03DR	P02DR	P01DR	P00DR	1111,1111
P1DR	P1 Drive Current Control Register	FE29H	P17DR	P16DR	P15DR	P14DR	P13DR	P12DR	P11DR	P10DR	1111,1111
P2DR	P2 Drive Current Control Register	FE2AH	P27DR	P26DR	P25DR	P24DR	P23DR	P22DR	P21DR	P20DR	1111,1111
P3DR	P3 Drive Current Control Register	FE2BH	P37DR	P36DR	P35DR	P34DR	P33DR	P32DR	P31DR	P30DR	1111,1111
P4DR	P4 Drive Current Control Register	FE2CH	P47DR	P46DR	P45DR	P44DR	P43DR	P42DR	P41DR	P40DR	1111,1111
P5DR	P5 Drive Current Control Register	FE2DH	-	-	P55DR	P54DR	P53DR	P52DR	P51DR	P50DR	xx11,1111
P6DR	P6 Drive Current Control Register	FE2EH	P67DR	P66DR	P65DR	P64DR	P63DR	P62DR	P61DR	P60DR	1111,1111
P7DR	P7 Drive Current Control Register	FE2FH	P77DR	P76DR	P75DR	P74DR	P73DR	P72DR	P71DR	P70DR	1111,1111
P0IE	P0 Input Enable Control Register	FE30H	P07IE	P06IE	P05IE	P04IE	P03IE	P02IE	P11IE	P00IE	1111,1111
P1IE	P1 Input Enable Control Register	FE31H	P17IE	P16IE	P15IE	P14IE	P13IE	P12IE	P11IE	P10IE	1111,1111
P2IE	P2 Input Enable Control Register	FE32H	P27IE	P26IE	P25IE	P24IE	P23IE	P22IE	P21IE	P20IE	1111,1111
P3IE	P3 Input Enable Control Register	FE33H	P37IE	P36IE	P35IE	P34IE	P33IE	P32IE	P31IE	P30IE	1111,1111
P4IE	P4 Input Enable Control Register	FE34H	P47IE	P46IE	P45IE	P44IE	P43IE	P42IE	P41IE	P40IE	1111,1111
P5IE	P5 Input Enable Control Register	FE35H	-	-	P55IE	P54IE	P53IE	P52IE	P51IE	P50IE	xx11,1111
P6IE	P6 Input Enable Control Register	FE36H	P67IE	P66IE	P65IE	P64IE	P63IE	P62IE	P61IE	P60IE	1111,1111
P7IE	P7 Input Enable Control Register	FE37H	P77IE	P76IE	P75IE	P74IE	P73IE	P72IE	P71IE	P70IE	1111,1111

LCMIFCFG	LCM Interface Configuration Register	FE50H	LCMIFIE	-	LCMIFIP[1:0]		LCMIFDPS[1:0]		D16_D8	M68_I80	0x00,0000
LCMIFCFG2	LCM Interface Configuration Register 2	FE51H	-	LCMIFCPS[1:0]		SETUPT[2:0]			HOLDT[1:0]		x000,0000
LCMIFCR	LCM Interface Control Register	FE52H	ENLCMIF	-	-	-	-	CMD[2:0]			0xxx,x000
LCMIFSTA	LCM Interface Status Register	FE53H	-	-	-	-	-	-	-	LCMIFIF	xxxx,xxx0
LCMIDDATL	LCM interface low byte data	FE54H	LCMIFDAT[7:0]								0000,0000
LCMIDDATH	LCM interface high byte data	FE55H	LCMIFDAT[15:8]								0000,0000
RTCCR	RTC Control Register	FE60H	-	-	-	-	-	-	-	RUNRTC	xxxx,xxx0
RTCCFG	RTC Configuration Register	FE61H	-	-	-	-	-	-	RTCCKS	SETRTC	xxxx,xx00
RTCIEN	RTC Interrupt Enable Register	FE62H	EALAI	EDAYI	EHOURI	EMINI	ESECI	ESEC2I	ESEC8I	ESEC32I	0000,0000
RTCIF	RTC Interrupt Request Register	FE63H	ALAIF	DAYIF	HOURIF	MINIF	SECIF	SEC2IF	SEC8IF	SEC32IF	0000,0000
ALAHOUR	the hour value of the RTC alarm	FE64H	-	-	-						xxx0,0000
ALAMIN	the minute value of the RTC alarm	FE65H	-	-							xx00,0000
ALASEC	the second value of the RTC alarm	FE66H	-	-							xx00,0000
ALASSEC	1/128 second value of RTC alarm	FE67H	-								x000,0000
INIYEAR	initialization of RTC year	FE68H	-								x000,0000
INIMONTH	initialization of RTC month	FE69H	-	-	-	-					xxxx,0000
INIDAY	initialization of RTC day	FE6AH	-	-	-						xxx0,0000
INIHOUR	initialization of RTC hour	FE6BH	-	-	-						xxx0,0000
INIMIN	initialization of RTC minute	FE6CH	-	-							xx00,0000
INISEC	initialization of RTC second	FE6DH	-	-							xx00,0000
INISSEC	initialization of RTC 1/128 second	FE6EH	-								x000,0000
YEAR	Year count value of RTC	FE70H	-								x000,0000
MONTH	Month count value of RTC	FE71H	-	-	-	-					xxxx,0000
DAY	Day count value of RTC	FE72H	-	-	-						xxx0,0000
HOUR	Hour count value of RTC	FE73H	-	-	-						xxx0,0000
MIN	Minute count value of RTC	FE74H	-	-							xx00,0000
SEC	Second count value of RTC	FE75H	-	-							xx00,0000
SSEC	1/128 second count value of RTC	FE76H	-								x000,0000
I2CCFG	I^{2}C Configuration Register	FE80H	ENI2C	MSSL	MSSPEED[6:1]						0000,0000
I2CMSCR	I^{2}C Master Control Register	FE81H	EMSI	-	-	-	MSCMD[3:0]				0xxx,0000
I2CMSST	I^{2}C Master Status Register	FE82H	MSBUSY	MSIF	-	-	-	-	MSACKI	MSACKO	00xx,xx00
I2CSLCR	I^{2}C Slave Control Register	FE83H	-	ESTAI	ERXI	ETXI	ESTOI	-	-	SLRST	x000,0xx0
I2CSLST	I^{2}C Slave Status Register	FE84H	SLBUSY	STAIF	RXIF	TXIF	STOIF	TXING	SLACKI	SLACKO	0000,0000
I2CSLADR	I^{2}C Slave Address Register	FE85H	SLADR[6:0]							MA	0000,0000
I2CTXD	I^{2}C Data Transmission Register	FE86H									0000,0000
I2CRXD	I^{2}C Data Receive Register	FE87H									0000,0000
I2CMSAUX	I^{2}C Master Auxiliary Control Register	FE88H	-	-	-	-	-	-	-	WDTA	xxxx,xxx0
TM2PS	Timer2 Clock Prescaler Register	FEA2H									0000,0000
TM3PS	Timer3 Clock Prescaler Register	FEA3H									0000,0000
TM4PS	Timer4 Clock Prescaler Register	FEA4H									0000,0000
ADCTIM	ADC Timing Control Register	FEA8H	CSSETUP	CSHOLD[1:0]		SMPDUTY[4:0]					0010,1010
ADCEXCFG	ADC Extended Configuration Register	FEADH	-	-	ADCETRS [1:0]		-	CVTIMESEL[2:0]			xx00,x000
CMPEXCFG	Comparator Extended Configuration Register	FEAEH	CHYS[1:0]		-	-	-	CMPNS	CMPPS[1:0]		00xx,x000
T3T4PIN	T3/T4 Select register	FEACH	-	-	-	-	-	-	-	T3T4SEL	xxxx,xxx0
PWMA_ETRPS	PWMA ETR Select register	FEB0H						BRKAPS	ETRAPS[1:0]		xxxx,x000
PWMA_ENO	PWMA Output enable control	FEB1H	ENO4N	ENO4P	ENO3N	ENO3P	ENO2N	ENO2P	ENO1N	ENO1P	0000,0000
PWMA_PS	PWMA Output pin selection register	FEB2H	C4PS[1:0]		C3PS[1:0]		C2PS[1:0]		C1PS[1:0]		0000,0000
PWMA_IOAUX	PWMAAuxiliary register	FEB3H	AUX4N	AUX4P	AUX3N	AUX3P	AUX2N	AUX2P	AUX1N	AUX1P	0000,0000
PWMB_ETRPS	PWMB ETR Select register	FEB4H						BRKBPS	ETRBPS[1:0]		xxxx,x000
PWMB_ENO	PWMB Output enable control	FEB5H	-	ENO8P	-	ENO7P	-	ENO6P	-	ENO5P	x0x0,x0x0
PWMB_PS	PWMB Output pin selection register	FEB6H	C8PS[1:0]		C7PS[1:0]		C6PS[1:0]		C5PS[1:0]		0000,0000
PWMB_IOAUX	PWM2 Auxiliary register	FEB7H	-	AUX8P	-	AUX7P	-	AUX6P	-	AUX5P	x0x0,x0x0
PWMA_CR1	PWMA Control register 1	FEC0H	ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN	0000,0000
PWMA_CR2	PWMA Control register 2	FEC1H	-	MMS[2:0]			-	COMS	-	CCPC	x000,x0x0
PWMA_SMCR	PWMA Slave mode control register	FEC2H	MSM	TS[2:0]			-	SMS[2:0]			0000,x000
PWMA_ETR	PWMA External trigger register	FEC3H	ETP	ECE	ETPS[1:0]		ETF[3:0]				0000,0000
PWMA_IER	PWMA Interrupt enable register	FEC4H	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE	0000,0000
PWMA_SR1	PWMA Status register1	FEC5H	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF	0000,0000
PWMA_SR2	PWMA Status register2	FEC6H	-	-	-	CC4OF	CC3OF	CC2OF	CC1OF	-	xxx0,000x
PWMA_EGR	PWMA Event occurrence register	FEC7H	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG	0000,0000
PWMA_CCMR1	PWMA Capture mode register1	FEC8H	OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]		0000,0000
	PWMA Compare mode register1		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]		0000,0000
PWMA_CCMR2	PWMA Capture mode register2	FEC9H	OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		0000,0000
	PWMA Compare mode register2		IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		0000,0000
PWMA_CCMR3	PWMA Capture mode register3	FECAH	OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]		0000,0000
	PWMA Compare mode register3		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]		0000,0000
PWMA_CCMR4	PWMA Capture mode register4	FECBH	OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		0000,0000
	PWMA Compare mode register4		IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		0000,0000
PWMA_CCER1	PWMA Capture compare enable register 1	FECCH	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E	0000,0000
PWMA_CCER2	PWMA Capture compare enable register 2	FECDH	CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	0000,0000
PWMA_CNTRH	PWMA Counter high byte	FECEH	CNT[15:8]								0000,0000
PWMA_CNTRL	PWMA Counter low byte	FECFH	CNT[7:0]								0000,0000

PWMA_PSCRH	PWMA Prescaler high byte	FED0H	PSC[15:8]								0000,0000
PWMA_PSCRL	PWMA Prescaler low byte	FED1H	PSC[7:0]								0000,0000
PWM1_ARRH	PWMA Auto reload register high byte	FED2H	ARR[15:8]								0000,0000
PWMA_ARRL	PWMA Auto reload register low byte	FED3H	ARR[7:0]								0000,0000
PWMA_RCR	PWMA Repeat counter register	FED4H	REP[7:0]								0000,0000
PWMA_CCR1H	PWMA Compare capture register1 high bit	FED5H	CCR1[15:8]								0000,0000
PWMA_CCR1L	PWMA Compare capture register1 low bit	FED6H	CCR1[7:0]								0000,0000
PWMA_CCR2H	PWMA Compare capture register2 high bit	FED7H	CCR2[15:8]								0000,0000
PWMA_CCR2L	PWMA Compare capture register2 low bit	FED8H	CCR2[7:0]								0000,0000
PWMA_CCR3H	PWMA Compare capture register3 high bit	FED9H	CCR3[15:8]								0000,0000
PWMA_CCR3L	PWMA Compare capture register3 low bit	FEDAH	CCR3[7:0]								0000,0000
PWMA_CCR4H	PWMA Compare capture register4 high bit	FEDBH	CCR4[15:8]								0000,0000
PWMA_CCR4L	PWMA Compare capture register4 low bit	FEDCH	CCR4[7:0]								0000,0000
PWMA_BKR	PWMA Brake register	FEDDH	MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		0000,000x
PWMA_DTR	PWMA Dead zone control register	FEDEH	DTG[7:0]								0000,0000
PWMA_OISR	PWMA Output idle status register	FEDFH	OIS4N	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	0000,0000
PWMB_CR1	PWMB Control register 1	FEEH	ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN	0000,0000
PWMB_CR2	PWMB Control register 2	FEE1H	-	MMS[2:0]			-	COMS	-	CCPC	x000,x0x0
PWMB_SMCR	PWMB Slave mode control register	FEE2H	MSM	TS[2:0]			-	SMS[2:0]			0000,x000
PWMB_ETR	PWMB External trigger register	FEE3H	ETP	ECE	ETPS[1:0]		ETF[3:0]				0000,0000
PWMB_IER	PWMB Interrupt enable register	FEE4H	BIE	TIE	COMIE	CC8IE	CC7IE	CC6IE	CC5IE	UIE	0000,0000
PWMB_SR1	PWMB Status register1	FEE5H	BIF	TIF	COMIF	CC8IF	CC7IF	CC6IF	CC5IF	UIF	0000,0000
PWMB_SR2	PWMB Status register2	FEE6H	-	-	-	CC8OF	CC7OF	CC6OF	CC5OF	-	xxx0,000x
PWMB_EGR	PWMB Event occurrence register	FEE7H	BG	TG	COMG	CC8G	CC7G	CC6G	CC5G	UG	0000,0000
PWMB_CCMR1	PWMB Capture mode register1	FEE8H	OC5CE	OC5M[2:0]			OC5PE	OC5FE	CC5S[1:0]		0000,0000
	PWMB Compare mode register1		IC5F[3:0]				IC5PSC[1:0]		CC5S[1:0]		0000,0000
PWMB_CCMR2	PWMB Capture mode register2	FEE9H	OC6CE	OC6M[2:0]			OC6PE	OC6FE	CC6S[1:0]		0000,0000
	PWMB Compare mode register2		IC6F[3:0]				IC6PSC[1:0]		CC6S[1:0]		0000,0000
PWMB_CCMR3	PWMB Capture mode register3	FEEAH	OC7CE	OC7M[2:0]			OC7PE	OC7FE	CC7S[1:0]		0000,0000
	PWMB Compare mode register3		IC7F[3:0]				IC7PSC[1:0]		CC7S[1:0]		0000,0000
PWMB_CCMR4	PWMB Capture mode register4	FEEBH	OC8CE	OC8M[2:0]			OC8PE	OC8FE	CC8S[1:0]		0000,0000
	PWMB Compare mode register4		IC8F[3:0]				IC8PSC[1:0]		CC8S[1:0]		0000,0000
PWMB_CCER1	PWMB Capture compare enable register 1	FEECH	-	-	CC6P	CC6E	-	-	CC5P	CC5E	xx00,xx00
PWMB_CCER2	PWMB Capture compare enable register 2	FEEDH	-	-	CC8P	CC8E	-	-	CC7P	CC7E	xx00,xx00
PWMB_CNTRH	PWMB Counter high byte	FEEEH	CNT[15:8]								0000,0000
PWMB_CNTRL	PWMB Counter low byte	FEEFH	CNT[7:0]								0000,0000
PWMB_PSCRH	PWMB Prescaler high byte	FEF0H	PSC[15:8]								0000,0000
PWMB_PSCRL	PWMB Prescaler low byte	FEF1H	PSC[7:0]								0000,0000
PWMB_ARRH	PWMB Auto reload register high byte	FEF2H	ARR[15:8]								0000,0000
PWMB_ARRL	PWMB Auto reload register low byte	FEF3H	ARR[7:0]								0000,0000
PWMB_RCR	PWMB Repeat counter register	FEF4H	REP[7:0]								0000,0000
PWMB_CCR5H	PWMB Compare capture register5 high bit	FEF5H	CCR1[15:8]								0000,0000
PWMB_CCR5L	PWMB Compare capture register5 low bit	FEF6H	CCR1[7:0]								0000,0000
PWMB_CCR6H	PWMB Compare capture register6 high bit	FEF7H	CCR2[15:8]								0000,0000
PWM2_CCR6L	PWM2 Compare capture register6 low bit	FEF8H	CCR2[7:0]								0000,0000
PWMB_CCR7H	PWMB Compare capture register7 high bit	FEF9H	CCR3[15:8]								0000,0000
PWMB_CCR7L	PWMB Compare capture register7 low bit	FEFAH	CCR3[7:0]								0000,0000
PWMB_CCR8H	PWMB Compare capture register8 high bit	FEFBH	CCR4[15:8]								0000,0000
PWMB_CCR8L	PWMB Compare capture register8 low bit	FEFCH	CCR4[7:0]								0000,0000
PWMB_BKR	PWMB Brake register	FEFDH	MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]	-	0000,000x
PWMB_DTR	PWMB2 Dead zone control register	FEFEH	DTG[7:0]								0000,0000
PWMB_OISR	PWMB Output idle status register	FEFFH	-	OIS8	-	OIS7	-	OIS6	-	OIS5	x0x0,x0x0
MD3	MDU Data register	FCF0H	MD3[7:0]								0000,0000
MD2	MDU Data register	FCF1H	MD2[7:0]								0000,0000
MD1	MDU Data register	FCF2H	MD1[7:0]								0000,0000
MD0	MDU Data register	FCF3H	MD0[7:0]								0000,0000
MD5	MDU Data register	FCF4H	MD5[7:0]								0000,0000
MD4	MDU Data register	FCF5H	MD4[7:0]								0000,0000
ARCON	MDU Mode Control Register	FCF6H	MODE[2:0]			SC[4:0]					0000,0000
OPCON	MDU Operation Control Register	FCF7H	-	MDOV	-	-	-	-	RST	ENOP	0000,0000
P0INTE	P0 Interrupt enable register	FD00H	P07INTE	P06INTE	P05INTE	P04INTE	P03INTE	P02INTE	P01INTE	P00INTE	0000,0000

P1INTE	P1 Interrupt enable register	FD01H	P17INTE	P16INTE	P15INTE	P14INTE	P13INTE	P12INTE	P11INTE	P10INTE	0000,0000
P2INTE	P2 Interrupt enable register	FD02H	P27INTE	P26INTE	P25INTE	P24INTE	P23INTE	P22INTE	P21INTE	P20INTE	0000,0000
P3INTE	P3 Interrupt enable register	FD03H	P37INTE	P36INTE	P35INTE	P34INTE	P33INTE	P32INTE	P31INTE	P30INTE	0000,0000
P4INTE	P4 Interrupt enable register	FD04H	P47INTE	P46INTE	P45INTE	P44INTE	P43INTE	P42INTE	P41INTE	P40INTE	0000,0000
P5INTE	P5 Interrupt enable register	FD05H	-	-	P55INTE	P54INTE	P53INTE	P52INTE	P51INTE	P50INTE	xx00,0000
P6INTE	P6 Interrupt enable register	FD06H	P67INTE	P66INTE	P65INTE	P64INTE	P63INTE	P62INTE	P61INTE	P60INTE	0000,0000
P7INTE	P7 Interrupt enable register	FD07H	P77INTE	P76INTE	P75INTE	P74INTE	P73INTE	P72INTE	P71INTE	P70INTE	0000,0000
P0INTF	P0 Interrupt flag register	FD10H	P07INTF	P06INTF	P05INTF	P04INTF	P03INTF	P02INTF	P01INTF	P00INTF	0000,0000
P1INTF	P1 Interrupt flag register	FD11H	P17INTF	P16INTF	P15INTF	P14INTF	P13INTF	P12INTF	P11INTF	P10INTF	0000,0000
P2INTF	P2 Interrupt flag register	FD12H	P27INTF	P26INTF	P25INTF	P24INTF	P23INTF	P22INTF	P21INTF	P20INTF	0000,0000
P3INTF	P3 Interrupt flag register	FD13H	P37INTF	P36INTF	P35INTF	P34INTF	P33INTF	P32INTF	P31INTF	P30INTF	0000,0000
P4INTF	P4 Interrupt flag register	FD14H	P47INTF	P46INTF	P45INTF	P44INTF	P43INTF	P42INTF	P41INTF	P40INTF	0000,0000
P5INTF	P5 Interrupt flag register	FD15H	-	-	P55INTF	P54INTF	P53INTF	P52INTF	P51INTF	P50INTF	xx00,0000
P6INTF	P6 Interrupt flag register	FD16H	P67INTF	P66INTF	P65INTF	P64INTF	P63INTF	P62INTF	P61INTF	P60INTF	0000,0000
P7INTF	P7 Interrupt flag register	FD17H	P77INTF	P76INTF	P75INTF	P74INTF	P73INTF	P72INTF	P71INTF	P70INTF	0000,0000
P0IM0	P0 Interrupt mode register0	FD20H	P07IM0	P06IM0	P05IM0	P04IM0	P03IM0	P02IM0	P01IM0	P00IM0	0000,0000
P1IM0	P1 Interrupt mode register0	FD21H	P17IM0	P16IM0	P15IM0	P14IM0	P13IM0	P12IM0	P11IM0	P10IM0	0000,0000
P2IM0	P2 Interrupt mode register0	FD22H	P27IM0	P26IM0	P25IM0	P24IM0	P23IM0	P22IM0	P21IM0	P20IM0	0000,0000
P3IM0	P3 Interrupt mode register0	FD23H	P37IM0	P36IM0	P35IM0	P34IM0	P33IM0	P32IM0	P31IM0	P30IM0	0000,0000
P4IM0	P4 Interrupt mode register0	FD24H	P47IM0	P46IM0	P45IM0	P44IM0	P43IM0	P42IM0	P41IM0	P40IM0	0000,0000
P5IM0	P5 Interrupt mode register0	FD25H	-	-	P55IM0	P54IM0	P53IM0	P52IM0	P51IM0	P50IM0	xx00,0000
P6IM0	P6 Interrupt mode register0	FD26H	P67IM0	P66IM0	P65IM0	P64IM0	P63IM0	P62IM0	P61IM0	P60IM0	0000,0000
P7IM0	P7 Interrupt mode register0	FD27H	P77IM0	P76IM0	P75IM0	P74IM0	P73IM0	P72IM0	P71IM0	P70IM0	0000,0000
P0IM1	P0 Interrupt mode register1	FD30H	P07IM1	P06IM1	P05IM1	P04IM1	P03IM1	P02IM1	P01IM1	P00IM1	0000,0000
P1IM1	P1 Interrupt mode register1	FD31H	P17IM1	P16IM1	P15IM1	P14IM1	P13IM1	P12IM1	P11IM1	P10IM1	0000,0000
P2IM1	P2 Interrupt mode register1	FD32H	P27IM1	P26IM1	P25IM1	P24IM1	P23IM1	P22IM1	P21IM1	P20IM1	0000,0000
P3IM1	P3 Interrupt mode register1	FD33H	P37IM1	P36IM1	P35IM1	P34IM1	P33IM1	P32IM1	P31IM1	P30IM1	0000,0000
P4IM1	P4 Interrupt mode register1	FD34H	P47IM1	P46IM1	P45IM1	P44IM1	P43IM1	P42IM1	P41IM1	P40IM1	0000,0000
P5IM1	P5 Interrupt mode register1	FD35H	-	-	P55IM1	P54IM1	P53IM1	P52IM1	P51IM1	P50IM1	xx00,0000
P6IM1	P6 Interrupt mode register1	FD36H	P67IM1	P66IM1	P65IM1	P64IM1	P63IM1	P62IM1	P61IM1	P60IM1	0000,0000
P7IM1	P7 Interrupt mode register1	FD37H	P77IM1	P76IM1	P75IM1	P74IM1	P73IM1	P72IM1	P71IM1	P70IM1	0000,0000
P0WKUE	P0 Interrupt Wake-Up Enable Register	FD40H	P07WKUE	P06WKUE	P05WKUE	P04WKUE	P03WKUE	P02WKUE	P01WKUE	P00WKUE	0000,0000
P1WKUE	P1 Interrupt Wake-Up Enable Register	FD41H	P17WKUE	P16WKUE	P15WKUE	P14WKUE	P13WKUE	P12WKUE	P11WKUE	P10WKUE	0000,0000
P2WKUE	P2 Interrupt Wake-Up Enable Register	FD42H	P27WKUE	P26WKUE	P25WKUE	P24WKUE	P23WKUE	P22WKUE	P21WKUE	P20WKUE	0000,0000
P3WKUE	P3 Interrupt Wake-Up Enable Register	FD43H	P37WKUE	P36WKUE	P35WKUE	P34WKUE	P33WKUE	P32WKUE	P31WKUE	P30WKUE	0000,0000
P4WKUE	P4 Interrupt Wake-Up Enable Register	FD44H	P47WKUE	P46WKUE	P45WKUE	P44WKUE	P43WKUE	P42WKUE	P41WKUE	P40WKUE	0000,0000
P5WKUE	P5 Interrupt Wake-Up Enable Register	FD45H	-	-	P55WKUE	P54WKUE	P53WKUE	P52WKUE	P51WKUE	P50WKUE	xx00,0000
P6WKUE	P6 Interrupt Wake-Up Enable Register	FD46H	P67WKUE	P66WKUE	P65WKUE	P64WKUE	P63WKUE	P62WKUE	P61WKUE	P60WKUE	0000,0000
P7WKUE	P7 Interrupt Wake-Up Enable Register	FD47H	P77WKUE	P76WKUE	P75WKUE	P74WKUE	P73WKUE	P72WKUE	P71WKUE	P70WKUE	0000,0000
PINIPL	I/O Interrupt Priority Low Register	FD60H	P7IP	P6IP	P5IP	P4IP	P3IP	P2IP	P1IP	P0IP	0000,0000
PINIPH	I/O Interrupt Priority High Register	FD61H	P7IPH	P6IPH	P5IPH	P4IPH	P3IPH	P2IPH	P1IPH	P0IPH	0000,0000
COMEN	COM Enable Register	FB00H	C7EN	C6EN	C5EN	C4EN	C3EN	C2EN	C1EN	C0EN	0000,0000
SEGENL	SEG Enable Register	FB01H	S7EN	S6EN	S5EN	S4EN	S3EN	S2EN	S1EN	S0EN	0000,0000
SEGENH	SEG Enable Register	FB02H	S15EN	S14EN	S13EN	S12EN	S11EN	S10EN	S9EN	S8EN	0000,0000
LEDCTRL	LED Control Register	FB03H	LEDON	-	LEDMODE[1:0]			LEDDUTY[2:0]			0000,0000
LEDCKS	LED Clock Divide Register	FB04H									0000,0001
COM0_DA_L	Common Anode Mode Disply	FB10H									0000,0000
COM1_DA_L	Common Anode Mode Disply	FB11H									0000,0000
COM2_DA_L	Common Anode Mode Disply	FB12H									0000,0000
COM3_DA_L	Common Anode Mode Disply	FB13H									0000,0000
COM4_DA_L	Common Anode Mode Disply	FB14H									0000,0000
COM5_DA_L	Common Anode Mode Disply	FB15H									0000,0000
COM6_DA_L	Common Anode Mode Disply	FB16H									0000,0000
COM7_DA_L	Common Anode Mode Disply	FB17H									0000,0000
COM0_DA_H	Common Anode Mode Disply	FB18H									0000,0000
COM1_DA_H	Common Anode Mode Disply	FB19H									0000,0000
COM2_DA_H	Common Anode Mode Disply	FB1AH									0000,0000
COM3_DA_H	Common Anode Mode Disply	FB1BH									0000,0000
COM4_DA_H	Common Anode Mode Disply	FB1CH									0000,0000
COM5_DA_H	Common Anode Mode Disply	FB1DH									0000,0000
COM6_DA_H	Common Anode Mode Disply	FB1EH									0000,0000
COM7_DA_H	Common Anode Mode Disply	FB1FH									0000,0000
COM0_DC_L	Common Cathode Mode Disply	FB20H									0000,0000
COM1_DC_L	Common Cathode Mode Disply	FB21H									0000,0000
COM2_DC_L	Common Cathode Mode Disply	FB22H									0000,0000
COM3_DC_L	Common Cathode Mode Disply	FB23H									0000,0000
COM4_DC_L	Common Cathode Mode Disply	FB24H									0000,0000
COM5_DC_L	Common Cathode Mode Disply	FB25H									0000,0000
COM6_DC_L	Common Cathode Mode Disply	FB26H									0000,0000
COM7_DC_L	Common Cathode Mode Disply	FB27H									0000,0000
COM0_DC_H	Common Cathode Mode Disply	FB28H									0000,0000
COM1_DC_H	Common Cathode Mode Disply	FB29H									0000,0000
COM2_DC_H	Common Cathode Mode Disply	FB2AH									0000,0000
COM3_DC_H	Common Cathode Mode Disply	FB2BH									0000,0000
COM4_DC_H	Common Cathode Mode Disply	FB2CH									0000,0000

COM5_DC_H	Common Cathode Mode Disply	FB2DH									0000,0000
COM6_DC_H	Common Cathode Mode Disply	FB2EH									0000,0000
COM7_DC_H	Common Cathode Mode Disply	FB2FH									0000,0000
TSCHEN1	Touch Key Enable Register 1	FB40H	TKEN7	TKEN6	TKEN5	TKEN4	TKEN3	TKEN2	TKEN1	TKEN0	0000,0000
TSCHEN2	Touch Key Enable Register 2	FB41H	TKEN15	TKEN14	TKEN13	TKEN12	TKEN11	TKEN10	TKEN9	TKEN8	0000,0000
TSCFG1	Touch Key Configuration Register 1	FB42H	-	SCR[2:0]			-	DT[2:0]			0000,0000
TSCFG2	Touch Key Configuration Register 2	FB43H	-	-	-	-	-	-	TSVR[1:0]		0000,0000
TSWUTC	Touch Key Wakeup Control Register	FB44H									0000,0000
TSCTRL	Touch Key Control Register	FB45H	TSGO	SINGLE	TSWAIT	TSWUCS	TSDCEN	TSWUEN	TSSAMP[1:0]		0000,0000
TSSTA1	Touch Key Status Register 1	FB46H	LEDWK	-	-	-	TSWKCHN[3:0]				0000,0000
TSSTA2	Touch Key Status Register 2	FB47H	TSIF	TSDOV	-	-	TSDNCHN[3:0]				0000,0000
TSRT	Touch Key Time Control Register	FB48H									0000,0000
TSDATH	Touch Key Data High Byte	FB49H									0000,0000
TSDATL	Touch Key Data Low Byte	FB4AH									0000,0000
TSTH00H	Touch Key0 Threshold High Byte	FB50H									0000,0000
TSTH00L	Touch Key0 Threshold Low Byte	FB51H									0000,0000
TSTH01H	Touch Key1 Threshold High Byte	FB52H									0000,0000
TSTH01L	Touch Key1 Threshold Low Byte	FB53H									0000,0000
TSTH02H	Touch Key2 Threshold High Byte	FB54H									0000,0000
TSTH02L	Touch Key2 Threshold Low Byte	FB55H									0000,0000
TSTH03H	Touch Key3 Threshold High Byte	FB56H									0000,0000
TSTH03L	Touch Key3 Threshold Low Byte	FB57H									0000,0000
TSTH04H	Touch Key4 Threshold High Byte	FB58H									0000,0000
TSTH04L	Touch Key4 Threshold Low Byte	FB59H									0000,0000
TSTH05H	Touch Key5 Threshold High Byte	FB5AH									0000,0000
TSTH05L	Touch Key5 Threshold Low Byte	FB5BH									0000,0000
TSTH06H	Touch Key6 Threshold High Byte	FB5CH									0000,0000
TSTH06L	Touch Key6 Threshold Low Byte	FB5DH									0000,0000
TSTH07H	Touch Key7 Threshold High Byte	FB5EH									0000,0000
TSTH07L	Touch Key7 Threshold Low Byte	FB5FH									0000,0000
TSTH08H	Touch Key8 Threshold High Byte	FB60H									0000,0000
TSTH08L	Touch Key8 Threshold Low Byte	FB61H									0000,0000
TSTH09H	Touch Key9 Threshold High Byte	FB62H									0000,0000
TSTH09L	Touch Key9 Threshold Low Byte	FB63H									0000,0000
TSTH10H	Touch Key10 Threshold High Byte	FB64H									0000,0000
TSTH10L	Touch Key10 Threshold Low Byte	FB65H									0000,0000
TSTH11H	Touch Key11 Threshold High Byte	FB66H									0000,0000
TSTH11L	Touch Key11 Threshold Low Byte	FB67H									0000,0000
TSTH12H	Touch Key12 Threshold High Byte	FB68H									0000,0000
TSTH12L	Touch Key12 Threshold Low Byte	FB69H									0000,0000
TSTH13H	Touch Key13 Threshold High Byte	FB6AH									0000,0000
TSTH13L	Touch Key13 Threshold Low Byte	FB6BH									0000,0000
TSTH14H	Touch Key14 Threshold High Byte	FB6CH									0000,0000
TSTH14L	Touch Key14 Threshold Low Byte	FB6DH									0000,0000
TSTH15H	Touch Key15 Threshold High Byte	FB6EH									0000,0000
TSTH15L	Touch Key15 Threshold Low Byte	FB6FH									0000,0000

LCDCFG	LCD configuration register	FB80H	CKSEL	-	-	-	VRLPSEL	VLCDSEL[2:0]			0xxx,0000
LCDCFG2	LCD configuration register 2	FB81H	-	-	-	-	SEG3PS	SEG2PS	SEG1PS	SEG0PS	xxxx,0000
DBLEN	Dead time length configuration	FB82H	-	-	-	-	-	DBLEN[2:0]			xxxx,x000
COMLENL	COM time length configuration low Bit	FB83H	COMLEN[7:0]								0000,0000
COMLENM	COM time length configuration median	FB84H	COMLEN[15:8]								0000,0000
COMLENH	COM time length configuration high bit	FB85H	-	-	-	-	COMLEN[19:16]				xxxx,0000
BLINKRATE	Flicker rate configuration register	FB86H	BLANKRATE[7:0]								1000,0000
LCDCR	LCD control register	FB87H	-	-	-	-	-	ACTMODE[1:0]		ENLCD	xxxx,x000
COMON	COM Line Enable Register	FB88H	-	-	-	-	COM3	COM2	COM1	COM0	xxxx,0000
SEGON1	SEG Line Enable Register 1	FB8AH	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0	0000,0000
SEGON2	SEG Line Enable Register 2	FB8BH	SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8	0000,0000
SEGON3	SEG Line Enable Register 3	FB8CH	SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16	0000,0000
SEGON4	SEG Line Enable Register 4	FB8DH	SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24	0000,0000
SEGON5	SEG Line Enable Register 5	FB8EH	SEG39	SEG38	SEG37	SEG36	SEG35	SEG34	SEG33	SEG32	0000,0000
C0SEGV0	C0SEG7_0 data register	FB90H	C0S7	C0S6	C0S5	C0S4	C0S3	C0S2	C0S1	C0S0	0000,0000
C0SEGV1	C0SEG15_8 data register	FB91H	C0S15	C0S14	C0S13	C0S12	C0S11	C0S10	C0S9	C0S8	0000,0000
C0SEGV2	C0SEG23_16 data register	FB92H	C0S23	C0S22	C0S21	C0S20	C0S19	C0S18	C0S17	C0S16	0000,0000
C0SEGV3	C0SEG31_24 data register	FB93H	C0S31	C0S30	C0S29	C0S28	C0S27	C0S26	C0S25	C0S24	0000,0000
C0SEGV4	C0SEG39_32 data register	FB94H	C0S39	C0S38	C0S37	C0S36	C0S35	C0S34	C0S33	C0S32	0000,0000
C1SEGV0	C1SEG7_0 data register	FB98H	C1S7	C1S6	C1S5	C1S4	C1S3	C1S2	C1S1	C1S0	0000,0000
C1SEGV1	C1SEG15_8 data register	FB99H	C1S15	C1S14	C1S13	C1S12	C1S11	C1S10	C1S9	C1S8	0000,0000
C1SEGV2	C1SEG23_16 data register	FB9AH	C1S23	C1S22	C1S21	C1S20	C1S19	C1S18	C1S17	C1S16	0000,0000
C1SEGV3	C1SEG31_24 data register	FB9BH	C1S31	C1S30	C1S29	C1S28	C1S27	C1S26	C1S25	C1S24	0000,0000
C1SEGV4	C1SEG39_32 data register	FB9CH	C1S39	C1S38	C1S37	C1S36	C1S35	C1S34	C1S33	C1S32	0000,0000
C2SEGV0	C2SEG7_0 data register	FBA0H	C2S7	C2S6	C2S5	C2S4	C2S3	C2S2	C2S1	C2S0	0000,0000
C2SEGV1	C2SEG15_8 data register	FBA1H	C2S15	C2S14	C2S13	C2S12	C2S11	C2S10	C2S9	C2S8	0000,0000
C2SEGV2	C2SEG23_16 data register	FBA2H	C2S23	C2S22	C2S21	C2S20	C2S19	C2S18	C2S17	C2S16	0000,0000
C2SEGV3	C2SEG31_24 data register	FBA3H	C2S31	C2S30	C2S29	C2S28	C2S27	C2S26	C2S25	C2S24	0000,0000
C2SEGV4	C2SEG39_32 data register	FBA4H	C2S39	C2S38	C2S37	C2S36	C2S35	C2S34	C2S33	C2S32	0000,0000
C3SEGV0	C3SEG7_0 data register	FBA8H	C3S7	C3S6	C3S5	C3S4	C3S3	C3S2	C3S1	C3S0	0000,0000
C3SEGV1	C3SEG15_8 data register	FBA9H	C3S15	C3S14	C3S13	C3S12	C3S11	C3S10	C3S9	C3S8	0000,0000
C3SEGV2	C3SEG23_16 data register	FBAAH	C3S23	C3S22	C3S21	C3S20	C3S19	C3S18	C3S17	C3S16	0000,0000
C3SEGV3	C3SEG31_24 data register	FBABH	C3S31	C3S30	C3S29	C3S28	C3S27	C3S26	C3S25	C3S24	0000,0000
C3SEGV4	C3SEG39_32 data register	FBACH	C3S39	C3S38	C3S37	C3S36	C3S35	C3S34	C3S33	C3S32	0000,0000
DMA_M2M_CFG	M2M_DMA configuration register	FA00H	M2MIE	-	TXACO	RXACO	M2MIP[1:0]		M2MPTY[1:0]		0x00,0000
DMA_M2M_CR	M2M_DMA control register	FA01H	ENM2M	TRIG	-	-	-	-	-	-	00xx,xxxx
DMA_M2M_STA	M2M_DMA status register	FA02H	-	-	-	-	-	-	-	M2MIF	xxxx,xxx0
DMA_M2M_AMT	M2M_DMA total bytes to be transferred	FA03H									0000,0000
DMA_M2M_DONE	M2M_DMA transfer completed bytes	FA04H									0000,0000
DMA_M2M_TXAH	M2M_DMA send address high byte	FA05H									0000,0000
DMA_M2M_TXAL	M2M_DMA send address low byte	FA06H									0000,0000
DMA_M2M_RXAH	M2M_DMA receive address high byte	FA07H									0000,0000
DMA_M2M_RXAL	M2M_DMA receive address low byte	FA08H									0000,0000
DMA_ADC_CFG	ADC_DMA configuration register	FA10H	ADCIE	-	-	-	ADCMIP[1:0]		ADCPTY[1:0]		0xxx,0000
DMA_ADC_CR	ADC_DMA control register	FA11H	ENADC	TRIG	-	-	-	-	-	-	00xx,xxxx
DMA_ADC_STA	ADC_DMA status register	FA12H	-	-	-	-	-	-	-	ADCIF	xxxx,xxx0
DMA_ADC_RXAH	ADC_DMA receive address high byte	FA17H									0000,0000
DMA_ADC_RXAL	ADC_DMA receive address low byte	FA18H									0000,0000
DMA_ADC_CFG2	ADC_DMA configuration register 2	FA19H	-	-	-	-	CVTIMESEL[3:0]				xxxx,0000
DMA_ADC_CHSW0	ADC_DMA channel enable 0	FA1AH	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	1000,0000
DMA_ADC_CHSW1	ADC_DMA channel enable 1	FA1BH	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0	0000,0001
DMA_SPI_CFG	SPI_DMA configuration register	FA20H	SPIIE	ACT_TX	ACT_RX	-	SPIIP[1:0]		SPIPTY[1:0]		000x,0000
DMA_SPI_CR	SPI_DMA control register	FA21H	ENSPI	TRIG_M	TRIG_S	-	-	-	-	CLRFIFO	000x,xxx0
DMA_SPI_STA	SPI_DMA status register	FA22H	-	-	-	-	-	TXOVW	RXLOSS	SPIIF	xxxx,x000
DMA_SPI_AMT	SPI_DMA total bytes to be transferred	FA23H									0000,0000
DMA_SPI_DONE	SPI_DMA transfer completed bytes	FA24H									0000,0000
DMA_SPI_TXAH	SPI_DMA send address high byte	FA25H									0000,0000
DMA_SPI_TXAL	SPI_DMA send address low byte	FA26H									0000,0000
DMA_SPI_RXAH	SPI_DMA receive address high byte	FA27H									0000,0000
DMA_SPI_RXAL	SPI_DMA receive address low byte	FA28H									0000,0000
DMA_SPI_CFG2	SPI_DMA configuration register 2	FA29H	-	-	-	-	-	WRPSS	SSS[1:0]		xxxx,x000
DMA_UR1T_CFG	UR1T_DMA configuration register	FA30H	UR1TIE	-	-	-	UR1TIP[1:0]		UR1TPTY[1:0]		0xxx,0000
DMA_UR1T_CR	UR1T_DMA control register	FA31H	ENUR1T	TRIG	-	-	-	-	-	-	00xx,xxxx
DMA_UR1T_STA	UR1T_DMA status register	FA32H	-	-	-	-	-	TXOVW	-	UR1TIF	xxxx,x0x0
DMA_UR1T_AMT	UR1T_DMA total bytes to be transferred	FA33H									0000,0000
DMA_UR1T_DONE	UR1T_DMA transfer completed bytes	FA34H									0000,0000
DMA_UR1T_TXAH	UR1T_DMA send address high	FA35H									0000,0000

	byte										
DMA_UR1T_TXAL	UR1T_DMA send address low byte	FA36H									0000,0000
DMA_UR1R_CFG	UR1R_DMA configuration register	FA38H	UR1RIE	-	-	-	UR1RIP[1:0]		UR1RPTY[1:0]		0xxx,0000
DMA_UR1R_CR	UR1R_DMA control register	FA39H	ENUR1R	-	TRIG	-	-	-	-	CLRFIFO	0x0x,xxx0
DMA_UR1R_STA	UR1R_DMA status register	FA3AH	-	-	-	-	-	-	RXLOSS	UR1RIF	xxxx,xx00
DMA_UR1R_AMT	UR1R_DMA total bytes to be transferred	FA3BH									0000,0000
DMA_UR1R_DONE	UR1R_DMA transfer completed bytes	FA3CH									0000,0000
DMA_UR1R_TXAH	UR1R_DMA send address high byte	FA3DH									0000,0000
DMA_UR1R_TXAL	UR1R_DMA send address low byte	FA3EH									0000,0000
DMA_UR2T_CFG	UR2T_DMA configuration register	FA40H	UR2TIE	-	-	-	UR2TIP[1:0]		UR2TPTY[1:0]		0xxx,0000
DMA_UR2T_CR	UR2T_DMA control register	FA41H	ENUR2T	TRIG	-	-	-	-	-	-	00xx,xxxx
DMA_UR2T_STA	UR2T_DMA status register	FA42H	-	-	-	-	-	TXOVW	-	UR2TIF	xxxx,x0x0
DMA_UR2T_AMT	UR2T_DMA total bytes to be transferred	FA43H									0000,0000
DMA_UR2T_DONE	UR2T_DMA transfer completed bytes	FA44H									0000,0000
DMA_UR2T_TXAH	UR2T_DMA send address high byte	FA45H									0000,0000
DMA_UR2T_TXAL	UR2T_DMA send address low byte	FA46H									0000,0000
DMA_UR2R_CFG	UR2R_DMA configuration register	FA48H	UR2RIE	-	-	-	UR2RIP[1:0]		UR2RPTY[1:0]		0xxx,0000
DMA_UR2R_CR	UR2R_DMA control register	FA49H	ENUR2R	-	TRIG	-	-	-	-	CLRFIFO	0x0x,xxx0
DMA_UR2R_STA	UR2R_DMA status register	FA4AH	-	-	-	-	-	-	RXLOSS	UR2RIF	xxxx,xx00
DMA_UR2R_AMT	UR2R_DMA total bytes to be transferred	FA4BH									0000,0000
DMA_UR2R_DONE	UR2R_DMA transfer completed bytes	FA4CH									0000,0000
DMA_UR2R_TXAH	UR2R_DMA send address high byte	FA4DH									0000,0000
DMA_UR2R_TXAL	UR2R_DMA send address low byte	FA4EH									0000,0000
DMA_UR3T_CFG	UR3T_DMA configuration register	FA50H	UR3TIE	-	-	-	UR3TIP[1:0]		UR3TPTY[1:0]		0xxx,0000
DMA_UR3T_CR	UR3T_DMA control register	FA51H	ENUR3T	TRIG	-	-	-	-	-	-	00xx,xxxx
DMA_UR3T_STA	UR3T_DMA status register	FA52H	-	-	-	-	-	TXOVW	-	UR3TIF	xxxx,x0x0
DMA_UR3T_AMT	UR3T_DMA total bytes to be transferred	FA53H									0000,0000
DMA_UR3T_DONE	UR3T_DMA transfer completed bytes	FA54H									0000,0000
DMA_UR3T_TXAH	UR3T_DMA send address high byte	FA55H									0000,0000
DMA_UR3T_TXAL	UR3T_DMA send address low byte	FA56H									0000,0000
DMA_UR3R_CFG	UR3R_DMA configuration register	FA58H	UR3RIE	-	-	-	UR3RIP[1:0]		UR3RPTY[1:0]		0xxx,0000
DMA_UR3R_CR	UR3R_DMA control register	FA59H	ENUR3R	-	TRIG	-	-	-	-	CLRFIFO	0x0x,xxx0
DMA_UR3R_STA	UR3R_DMA status register	FA5AH	-	-	-	-	-	-	RXLOSS	UR3RIF	xxxx,xx00
DMA_UR3R_AMT	UR3R_DMA total bytes to be transferred	FA5BH									0000,0000
DMA_UR3R_DONE	UR3R_DMA transfer completed bytes	FA5CH									0000,0000
DMA_UR3R_TXAH	UR3R_DMA send address high byte	FA5DH									0000,0000
DMA_UR3R_TXAL	UR3R_DMA send address low byte	FA5EH									0000,0000
DMA_UR4T_CFG	UR4T_DMA configuration register	FA60H	UR4TIE	-	-	-	UR4TIP[1:0]		UR4TPTY[1:0]		0xxx,0000
DMA_UR4T_CR	UR4T_DMA control register	FA61H	ENUR4T	TRIG	-	-	-	-	-	-	00xx,xxxx
DMA_UR4T_STA	UR4T_DMA status register	FA62H	-	-	-	-	-	TXOVW	-	UR4TIF	xxxx,x0x0
DMA_UR4T_AMT	UR4T_DMA total bytes to be transferred	FA63H									0000,0000
DMA_UR4T_DONE	UR4T_DMA transfer completed bytes	FA64H									0000,0000
DMA_UR4T_TXAH	UR4T_DMA send address high byte	FA65H									0000,0000
DMA_UR4T_TXAL	UR4T_DMA send address low byte	FA66H									0000,0000
DMA_UR4R_CFG	UR4R_DMA configuration register	FA68H	UR4RIE	-	-	-	UR4RIP[1:0]		UR4RPTY[1:0]		0xxx,0000
DMA_UR4R_CR	UR4R_DMA control register	FA69H	ENUR4R	-	TRIG	-	-	-	-	CLRFIFO	0x0x,xxx0
DMA_UR4R_STA	UR4R_DMA status register	FA6AH	-	-	-	-	-	-	RXLOSS	UR4RIF	xxxx,xx00
DMA_UR4R_AMT	UR4R_DMA total bytes to be transferred	FA6BH									0000,0000
DMA_UR4R_DONE	UR4R_DMA transfer completed bytes	FA6CH									0000,0000
DMA_UR4R_TXAH	UR4R_DMA send address high byte	FA6DH									0000,0000
DMA_UR4R_TXAL	UR4R_DMA send address low byte	FA6EH									0000,0000
DMA_LCM_CFG	LCM_DMA configuration register	FA70H	LCMIE	-	-	-	LCMIP[1:0]		LCMPTY[1:0]		0xxx,0000
DMA_LCM_CR	LCM_DMA control register	FA71H	ENLCM	TRIGWC	TRIGWD	TRIGRC	TRIGRD	-	-	-	0000,0xxx
DMA_LCM_STA	LCM_DMA status register	FA72H	-	-	-	-	-	-	TXOVW	LCMIF	xxxx,xx00
DMA_LCM_AMT	LCM_DMA total bytes to be transferred	FA73H									0000,0000
DMA_LCM_DONE	LCM_DMA transfer completed bytes	FA74H									0000,0000
DMA_LCM_TXAH	LCM_DMA send address high byte	FA75H									0000,0000
DMA_LCM_TXAL	LCM_DMA send address low byte	FA76H									0000,0000
DMA_LCM_RXAH	LCM_DMA receive address high	FA77H									0000,0000

	byte			
DMA_LCM_RXAL	LCM_DMA receive address low byte	FA78H		0000,0000

Note: The meaning of the initial value of the special function register:

0: The initial value is 0;

1: The initial value is 1;

n: The initial value is related to the hardware options when the ISP downloads;

x: This bit does not exist, the initial value is undefined.

9 I/O Ports

Product line	Maximum I/O lines
STC8H1K08 family	**17**
STC8H1K28 family	**29**
STC8H3K64S4 family	**45**
STC8H3K64S2 family	**45**
STC8H8K64U family	**60**
STC8H2K64T family	**44**
STC8H4K64TLR family	**44**
STC8H4K64TLCD family	**60**
STC8H4K64LCD family	**61**

There are 4 modes for all GPIOs of STC8H series MCU, quasi bidirectional or weak pull-up mode (standard 8051 output mode), push-pull output / strong pull-up mode, high-impedance input mode (where current can neither flow in nor out), open drain mode. It is easy to configure the I/O mode using software.

Note: All I/O ports except for P3.0 and P3.1 are in high-impedance input state after power-on. You must set the I/O port mode before using it.

9.1 Registers Related to I/O

Symbol	Description	Address	Bit Address and Symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
P0	Port 0	80H	P07	P06	P05	P04	P03	P02	P01	P00	1111,1111
P1	Port 1	90H	P17	P16	P15	P14	P13	P12	P11	P10	1111,1111
P2	Port 2	A0H	P27	P26	P25	P24	P23	P22	P21	P20	1111,1111
P3	Port 3	B0H	P37	P36	P35	P34	P33	P32	P31	P30	1111,1111
P4	Port 4	C0H	P47	P46	P45	P44	P43	P42	P41	P40	1111,1111
P5	Port 5	C8H	-	-	P55	P54	P53	P52	P51	P50	xx11,1111
P6	Port 6	E8H	P67	P66	P65	P64	P63	P62	P61	P60	1111,1111
P7	Port 7	F8H	P77	P76	P75	P74	P73	P72	P71	P70	1111,1111
P0M1	Port 0 mode register 1	93H	P07M1	P06M1	P05M1	P04M1	P03M1	P02M1	P01M1	P00M1	1111,1111
P0M0	Port 0 mode register 0	94H	P07M0	P06M0	P05M0	P04M0	P03M0	P02M0	P01M0	P00M0	0000,0000
P1M1	Port 1 mode register 1	91H	P17M1	P16M1	P15M1	P14M1	P13M1	P12M1	P11M1	P10M1	1111,1111
P1M0	Port 1 mode register 0	92H	P17M0	P16M0	P15M0	P14M0	P13M0	P12M0	P11M0	P10M0	0000,0000
P2M1	Port 2 mode register 1	95H	P27M1	P26M1	P25M1	P24M1	P23M1	P22M1	P21M1	P20M1	1111,1111
P2M0	Port 2 mode register 0	96H	P27M0	P26M0	P25M0	P24M0	P23M0	P22M0	P21M0	P20M0	0000,0000
P3M1	Port 3 mode register 1	B1H	P37M1	P36M1	P35M1	P34M1	P33M1	P32M1	P31M1	P30M1	n111,1100
P3M0	Port 3 mode register 0	B2H	P37M0	P36M0	P35M0	P34M0	P33M0	P32M0	P31M0	P30M0	n000,0000
P4M1	Port 4 mode register 1	B3H	P47M1	P46M1	P45M1	P44M1	P43M1	P42M1	P41M1	P40M1	1111,1111
P4M0	Port 4 mode register 0	B4H	P47M0	P46M0	P45M0	P44M0	P43M0	P42M0	P41M0	P40M0	0000,0000
P5M1	Port 5 mode register 1	C9H	-	-	P55M1	P54M1	P53M1	P52M1	P51M1	P50M1	xx11,1111
P5M0	Port 5 mode register 0	CAH	-	-	P55M0	P54M0	P53M0	P52M0	P51M0	P50M0	xx00,0000
P6M1	P6 mode registe 1	CBH	P67M1	P66M1	P65M1	P64M1	P63M1	P62M1	P61M1	P60M1	0000,0000
P6M0	P6 mode registe 0	CCH	P67M0	P66M0	P65M0	P64M0	P63M0	P62M0	P61M0	P60M0	0000,0000
P7M1	P7 mode registe 1	E1H	P77M1	P76M1	P75M1	P74M1	P73M1	P72M1	P71M1	P70M1	0000,0000
P7M0	P7 mode registe 0	E2H	P77M0	P76M0	P75M0	P74M0	P73M0	P72M0	P71M0	P70M0	0000,0000

Symbol	Description	Address	Bit Address and Symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
P0PU	P0 Pull-up resistor control register	FE10H	P07PU	P06PU	P05PU	P04PU	P03PU	P02PU	P01PU	P00PU	0000,0000
P1PU	P1 Pull-up resistor control register	FE11H	P17PU	P16PU	P15PU	P14PU	P13PU	P12PU	P11PU	P10PU	0000,0000
P2PU	P2 Pull-up resistor control register	FE12H	P27PU	P26PU	P25PU	P24PU	P23PU	P22PU	P21PU	P20PU	0000,0000
P3PU	P3 Pull-up resistor control register	FE13H	P37PU	P36PU	P35PU	P34PU	P33PU	P32PU	P31PU	P30PU	0000,0000
P4PU	P4 Pull-up resistor control register	FE14H	P47PU	P46PU	P45PU	P44PU	P43PU	P42PU	P41PU	P40PU	0000,0000
P5PU	P5 Pull-up resistor control register	FE15H	-	-	P55PU	P54PU	P53PU	P52PU	P51PU	P50PU	xx00,0000
P6PU	P6 Pull-up resistor control register	FE16H	P67PU	P66PU	P65PU	P64PU	P63PU	P62PU	P61PU	P60PU	0000,0000
P7PU	P7 Pull-up resistor control register	FE17H	P77PU	P76PU	P75PU	P74PU	P73PU	P72PU	P71PU	P70PU	0000,0000
P0NCS	P0 Schmitt trigger control register	FE18H	P07NCS	P06NCS	P05NCS	P04NCS	P03NCS	P02NCS	P01NCS	P00NCS	0000,0000
P1NCS	P1 Schmitt trigger control register	FE19H	P17NCS	P16NCS	P15NCS	P14NCS	P13NCS	P12NCS	P11NCS	P10NCS	0000,0000
P2NCS	P2 Schmitt trigger control registe	FE1AH	P27NCS	P26NCS	P25NCS	P24NCS	P23NCS	P22NCS	P21NCS	P20NCS	0000,0000
P3NCS	P3 Schmitt trigger control registe	FE1BH	P37NCS	P36NCS	P35NCS	P34NCS	P33NCS	P32NCS	P31NCS	P30NCS	0000,0000
P4NCS	P4 Schmitt trigger control registe	FE1CH	P47NCS	P46NCS	P45NCS	P44NCS	P43NCS	P42NCS	P41NCS	P40NCS	0000,0000
P5NCS	P5 Schmitt trigger control registe	FE1DH	-	-	P55NCS	P54NCS	P53NCS	P52NCS	P51NCS	P50NCS	xx00,0000
P6NCS	P6 Schmitt trigger control registe	FE1EH	P67NCS	P66NCS	P65NCS	P64NCS	P63NCS	P62NCS	P61NCS	P60NCS	0000,0000
P7NCS	P7 Schmitt trigger control registe	FE1FH	P77NCS	P76NCS	P75NCS	P74NCS	P73NCS	P72NCS	P71NCS	P70NCS	0000,0000
P0SR	Port0 Level Shift Rate Register	FE20H	P07SR	P06SR	P05SR	P04SR	P03SR	P02SR	P01SR	P00SR	1111,1111
P1SR	Port1 Level Shift Rate Register	FE21H	P17SR	P16SR	P15SR	P14SR	P13SR	P12SR	P11SR	P10SR	1111,1111

P2SR	Port2 Level Shift Rate Register	FE22H	P27SR	P26SR	P25SR	P24SR	P23SR	P22SR	P21SR	P20SR	1111,1111
P3SR	Port3 Level Shift Rate Register	FE23H	P37SR	P36SR	P35SR	P34SR	P33SR	P32SR	P31SR	P30SR	1111,1111
P4SR	Port4 Level Shift Rate Register	FE24H	P47SR	P46SR	P45SR	P44SR	P43SR	P42SR	P41SR	P40SR	1111,1111
P5SR	Port5 Level Shift Rate Register	FE25H	-	-	-	P54SR	P53SR	P52SR	P51SR	P50SR	xx11,1111
P6SR	Port6 Level Shift Rate Register	FE26H	P67SR	P66SR	P65SR	P64SR	P63SR	P62SR	P61SR	P60SR	1111,1111
P7SR	Port7 Level Shift Rate Register	FE27H	P77SR	P76SR	P75SR	P74SR	P73SR	P72SR	P71SR	P70SR	1111,1111
P0DR	P0 Drive Current Control Register	FE28H	P07DR	P06DR	P05DR	P04DR	P03DR	P02DR	P01DR	P00DR	1111,1111
P1DR	P1 Drive Current Control Register	FE29H	P17DR	P16DR	P15DR	P14DR	P13DR	P12DR	P11DR	P10DR	1111,1111
P2DR	P2 Drive Current Control Register	FE2AH	P27DR	P26DR	P25DR	P24DR	P23DR	P22DR	P21DR	P20DR	1111,1111
P3DR	P3 Drive Current Control Register	FE2BH	P37DR	P36DR	P35DR	P34DR	P33DR	P32DR	P31DR	P30DR	1111,1111
P4DR	P4 Drive Current Control Register	FE2CH	P47DR	P46DR	P45DR	P44DR	P43DR	P42DR	P41DR	P40DR	1111,1111
P5DR	P5 Drive Current Control Register	FE2DH	-	-	P55DR	P54DR	P53DR	P52DR	P51DR	P50DR	xx11,1111
P6DR	P6 Drive Current Control Register	FE2EH	P67DR	P66DR	P65DR	P64DR	P63DR	P62DR	P61DR	P60DR	1111,1111
P7DR	P7 Drive Current Control Register	FE2FH	P77DR	P76DR	P75DR	P74DR	P73DR	P72DR	P71DR	P70DR	1111,1111
P0IE	P0 Input Enable Control Register	FE30H	P07IE	P06IE	P05IE	P04IE	P03IE	P02IE	P01IE	P00IE	1111,1111
P1IE	P1 Input Enable Control Register	FE31H	P17IE	P16IE	P15IE	P14IE	P13IE	P12IE	P11IE	P10IE	1111,1111
P2IE	P2 Input Enable Control Register	FE32H	P27IE	P26IE	P25IE	P24IE	P23IE	P22IE	P21IE	P20IE	1111,1111
P3IE	P3 Input Enable Control Register	FE33H	P37IE	P36IE	P35IE	P34IE	P33IE	P32IE	P31IE	P30IE	1111,1111
P4IE	P4 Input Enable Control Register	FE34H	P47IE	P46IE	P45IE	P44IE	P43IE	P42IE	P41IE	P40IE	1111,1111
P5IE	P5 Input Enable Control Register	FE35H	-	-	P55IE	P54IE	P53IE	P52IE	P41IE	P50IE	xx11,1111
P6IE	P6 Input Enable Control Register	FE36H	P67IE	P66IE	P65IE	P64IE	P63IE	P62IE	P41IE	P60IE	1111,1111
P7IE	P7 Input Enable Control Register	FE37H	P77IE	P76IE	P75IE	P74IE	P73IE	P72IE	P41IE	P70IE	1111,1111

9.1.1 Port Data Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P0	80H	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
P1	90H	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
P2	A0H	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
P3	B0H	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0
P4	C0H	P4.7	P4.6	P4.5	P4.4	P4.3	P4.2	P4.1	P4.0
P5	C8H	-	-	P5.5	P5.4	P5.3	P5.2	P5.1	P5.0
P6	E8H	P6.7	P6.6	P6.5	P6.4	P6.3	P6.2	P6.1	P6.0
P7	F8H	P7.7	P7.6	P7.5	P7.4	P7.3	P7.2	P7.1	P7.0

Read and write port status

Write 0: Output low to port buffer.
Write 1: Output high to port buffer.
Read: Read the level on the port pin directly.

9.1.2 Ports Mode Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P0M0	94H	P07M1	P06M1	P05M1	P04M1	P03M1	P02M1	P01M1	P00M1
P0M1	93H	P07M0	P06M0	P05M0	P04M0	P03M0	P02M0	P01M0	P00M0
P1M0	92H	P17M1	P16M1	P15M1	P14M1	P13M1	P12M1	P11M1	P10M1
P1M1	91H	P17M0	P16M0	P15M0	P14M0	P13M0	P12M0	P11M0	P10M0
P2M0	96H	P27M1	P26M1	P25M1	P24M1	P23M1	P22M1	P21M1	P20M1
P2M1	95H	P27M0	P26M0	P25M0	P24M0	P23M0	P22M0	P21M0	P20M0
P3M0	B2H	P37M1	P36M1	P35M1	P34M1	P33M1	P32M1	P31M1	P30M1
P3M1	B1H	P37M0	P36M0	P35M0	P34M0	P33M0	P32M0	P31M0	P30M0
P4M0	B4H	P47M1	P46M1	P45M1	P44M1	P43M1	P42M1	P41M1	P40M1
P4M1	B3H	P47M0	P46M0	P45M0	P44M0	P43M0	P42M0	P41M0	P40M0
P5M0	CAH	-	-	P55M1	P54M1	P53M1	P52M1	P51M1	P50M1
P5M1	C9H	-	-	P55M0	P54M0	P53M0	P52M0	P51M0	P50M0
P6M0	CCH	P67M0	P66M0	P65M0	P64M0	P63M0	P62M0	P61M0	P60M0
P6M1	CBH	P67M1	P66M1	P65M1	P64M1	P63M1	P62M1	P61M1	P60M1
P7M0	E2H	P77M0	P76M0	P75M0	P74M0	P73M0	P72M0	P71M0	P70M0
P7M1	E1H	P77M1	P76M1	P75M1	P74M1	P73M1	P72M1	P71M1	P70M1

Configure the mode of the ports as shown below.

PnM1.x	PnM0.x	Pn.x mode
0	0	quasi bidirectional mode

0	1	push-pull output mode
1	0	high-impedance input mode
1	1	open drain mode

Note: When an I/O port is selected as the ADC input channel, the PxM0/PxM1 register must be set to set the I/O port mode to input mode. In addition, if the ADC channel still needs to be enabled after the MCU enters the power-down mode/clock stop mode, you need to set the PxIE register to close the digital input to ensure that there will be no additional power consumption.

9.1.3 Pull-up Resistor Control Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P0PU	FE10H	P07PU	P06PU	P05PU	P04PU	P03PU	P02PU	P01PU	P00PU
P1PU	FE11H	P17PU	P16PU	P15PU	P14PU	P13PU	P12PU	P11PU	P10PU
P2PU	FE12H	P27PU	P26PU	P25PU	P24PU	P23PU	P22PU	P21PU	P20PU
P3PU	FE13H	P37PU	P36PU	P35PU	P34PU	P33PU	P32PU	P31PU	P30PU
P4PU	FE14H	P47PU	P46PU	P45PU	P44PU	P43PU	P42PU	P41PU	P40PU
P5PU	FE15H	-	-	P55PU	P54PU	P53PU	P52PU	P51PU	P50PU
P6PU	FE16H	P67PU	P66PU	P65PU	P64PU	P63PU	P62PU	P61PU	P60PU
P7PU	FE17H	P77PU	P76PU	P75PU	P74PU	P73PU	P72PU	P71PU	P70PU

Internal 4.1K pull-up resistor control bit. (Note: The pull-up resistors on the P3.0 and P3.1 ports may be slightly smaller.)

0: Disable 4.1K pull-up resistor inside the port
1: Enable 4.1K pull-up resistor inside the port

9.1.4 Schmitt Trigger Control Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P0NCS	FE18H	P07NCS	P06NCS	P05NCS	P04NCS	P03NCS	P02NCS	P01NCS	P00NCS
P1NCS	FE19H	P17NCS	P16NCS	P15NCS	P14NCS	P13NCS	P12NCS	P11NCS	P10NCS
P2NCS	FE1AH	P27NCS	P26NCS	P25NCS	P24NCS	P23NCS	P22NCS	P21NCS	P20NCS
P3NCS	FE1BH	P37NCS	P36NCS	P35NCS	P34NCS	P33NCS	P32NCS	P31NCS	P30NCS
P4NCS	FE1CH	P47NCS	P46NCS	P45NCS	P44NCS	P43NCS	P42NCS	P41NCS	P40NCS
P5NCS	FE1DH	-	-	P55NCS	P54NCS	P53NCS	P52NCS	P51NCS	P50NCS
P6NCS	FE1EH	P67NCS	P66NCS	P65NCS	P64NCS	P63NCS	P62NCS	P61NCS	P60NCS
P7NCS	FE1FH	P77NCS	P76NCS	P75NCS	P74NCS	P73NCS	P72NCS	P71NCS	P70NCS

Schmitt trigger control bit:

0: Enable schmitt trigger function on the port. (Schmitt trigger is enabled by default after power-on reset.)
1: Disable schmitt trigger function on the port.

9.1.5 Level Shifting Speed Control Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0	Reset value
P0SR	FE20H	P07SR	P06SR	P05SR	P04SR	P03SR	P02SR	P01SR	P00SR	1111,1111
P1SR	FE21H	P17SR	P16SR	P15SR	P14SR	P13SR	P12SR	P11SR	P10SR	1111,1111
P2SR	FE22H	P27SR	P26SR	P25SR	P24SR	P23SR	P22SR	P21SR	P20SR	1111,1111
P3SR	FE23H	P37SR	P36SR	P35SR	P34SR	P33SR	P32SR	P31SR	P30SR	1111,1111
P4SR	FE24H	P47SR	P46SR	P45SR	P44SR	P43SR	P42SR	P41SR	P40SR	1111,1111
P5SR	FE25H	-	-	P55SR	P54SR	P53SR	P52SR	P51SR	P50SR	xx11,1111
P6SR	FE26H	P57SR	P66SR	P65SR	P64SR	P63SR	P62SR	P61SR	P60SR	1111,1111
P7SR	FE27H	P77SR	P76SR	P75SR	P74SR	P73SR	P72SR	P71SR	P70SR	1111,1111

Level shifting speed control bits:

0: Fast level shifting, and the corresponding up and down impact will be relatively large.
1: Slow level shifting, and the corresponding up and down impact will be relatively small.

9.1.6 Drive Current Control Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0	Reset value
P0DR	FE28H	P07DR	P06DR	P05DR	P04DR	P03DR	P02DR	P01DR	P00DR	1111,1111
P1DR	FE29H	P17DR	P16DR	P15DR	P14DR	P13DR	P12DR	P11DR	P10DR	1111,1111
P2DR	FE2AH	P27DR	P26DR	P25DR	P24DR	P23DR	P22DR	P21DR	P20DR	1111,1111
P3DR	FE2BH	P37DR	P36DR	P35DR	P34DR	P33DR	P32DR	P31DR	P30DR	1111,1111
P4DR	FE2CH	P47DR	P46DR	P45DR	P44DR	P43DR	P42DR	P41DR	P40DR	1111,1111
P5DR	FE2DH	-	-	P55DR	P54DR	P53DR	P52DR	P51DR	P50DR	xx11,1111
P6DR	FE2EH	P67DR	P66DR	P65DR	P64DR	P63DR	P62DR	P61DR	P60DR	1111,1111
P7DR	FE2FH	P77DR	P76DR	P75DR	P74DR	P73DR	P72DR	P71DR	P70DR	1111,1111

Drive capability control bit:

0: Enhanced drive ability

1: General drive ability

9.1.7 Port digital signal input enable control register (PxIE)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P0IE	FE30H	P07IE	P06IE	P05IE	P04IE	P03IE	P02IE	P11IE	P00IE
P1IE	FE31H	P17IE	P16IE	P15IE	P14IE	P13IE	P12IE	P11IE	P10IE
P2IE	FE32H	P27IE	P26IE	P25IE	P24IE	P23IE	P22IE	P21IE	P20IE
P3IE	FE33H	P37IE	P36IE	P35IE	P34IE	P33IE	P32IE	P31IE	P30IE
P4IE	FE34H	P47IE	P46IE	P45IE	P44IE	P43IE	P42IE	P41IE	P40IE
P5IE	FE35H	-	-	P55IE	P54IE	P53IE	P52IE	P41IE	P50IE
P6IE	FE36H	P67IE	P66IE	P65IE	P64IE	P63IE	P62IE	P41IE	P60IE
P7IE	FE37H	P77IE	P76IE	P75IE	P74IE	P73IE	P72IE	P41IE	P70IE

Digital signal input enable control:

0: Disable digital signal input. If the I/O is used as an analog port such as a comparator input port, ADC input port, touch key input port or external crystal oscillator input pin, it must be set to 0 before entering the clock stop mode, otherwise there will be additional power consumption.

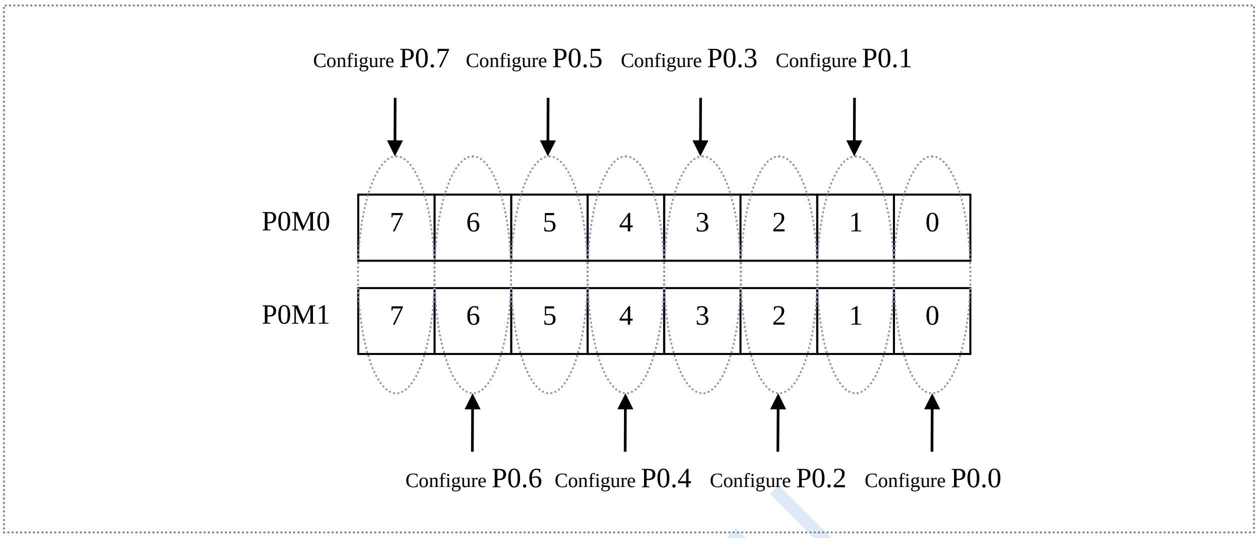
1: Enable digital signal input. If the I/O is used as a digital port, it must be set to 1, otherwise the MCU cannot read the level of the external port.

Special attention: For MCU with RTC function, when the clock source of RTC selects an external 32.768K crystal oscillator, it is necessary to close the digital channels of pins P1.6 and P1.7 which are connected with the crystal oscillator. Otherwise there will be additional leakage. (Set both bits 6 and 7 of register P1IE to 0 to close the digital channels of P1.6 and P1.7)

9.2 Configure I/O Ports

Two registers are used to configure each I/O mode.

Taking Port 0 as an example, two registers, P0M0 and P0M1, are used to configure Port 0, as shown in the following figure:



The combination of bit 0 of P0M0 and bit 0 of P0M1 is used to configure the mode of P0.0.
The combination of bit 1 of P0M0 and bit 1 of P0M1 is used to configure the mode of P0.1.
All other I/O lines configuration method is similar.

The combination of PnM0 and PnM1 to configure the I/O ports mode is as following.

PnM1	PnM0	I/O ports Mode
0	0	Quasi bidirectional (traditional 8051 I/O port, weak pull-up) Sink Current up to 20mA , Pull-up Current is 270~150μA (manufacturing error may be exist)
0	1	Push-pull output (strong pull-up output, current can be up to 20mA, resistors should be used to limit current)
1	0	high-impedance (where current can neither flow in nor out)
1	1	Open Drain mode. The internal pull-up resistors are disabled. The open drain mode can be used for both external status reading and output high or low. To read the external state correctly or output high level, the external pull-up resistors should be connected, otherwise the external state can not be read and the high level can not be output.

Note: n = 0,1,2,3,4,5,6,7

Note:

Any I/O port line can tolerate 20mA of sink current in weak pull-up mode (quasi-bidirectional mode) or strong push-pull output mode or open drain mode, and can output 20mA pull current in the strong push-pull output mode. Current limiting resistors should be connected in all I/O mode above, such as 1KΩ, 560Ω, 472Ω, etc. But the working current of the whole chip is recommended not to exceed 70mA, that is, the current flowing in from Vcc is not recommended to exceed 70mA, the current flowing from Gnd is not to exceed 70mA, and the overall current flowing in/out of it is not recommended to exceed 70mA.

9.3 I/O Ports Structure

9.3.1 Quasi-Bidirectional I/O (weak pull-up)

A quasi bidirectional port can be used as an input and output functions without the need to reconfigure the port. This is because the drive capability is weak when the port outputs a logic high level, allowing external devices to pull it low. When the pin outputs low, it has strong driving capability and able to sink a considerable current. There are three pull-up transistors in the quasi-bidirectional output to adapt different needs.

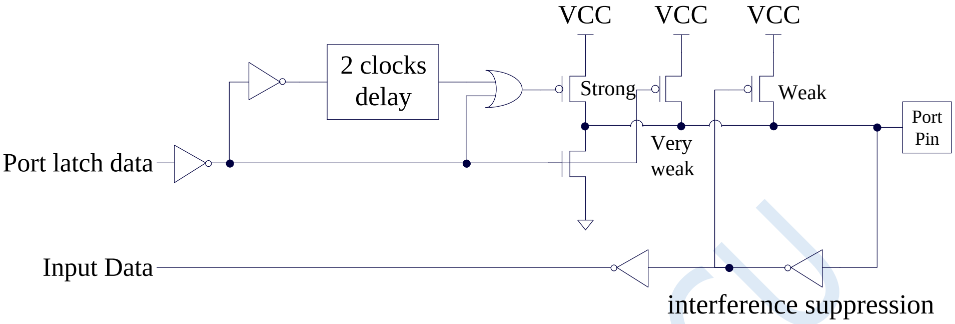
One of the three pull-up transistors, called "weak pull-up", is turned on when the port register is logic "1" and the pin itself is logic “1”. This pull-up transistor provides the basic drive current to make the quasi-bidirectional port output logic “1”. If one of the pin outputs logic “1” and the external device pulls it low, the weak pull-up transistor is off and the “very weak pull-up” maintains on. To pull the pin low, the external device must have sufficient sink capability to make the voltage on the pin drop below the threshold voltage. For a 5V microcontroller, the current of “weak pull-up” transistor is about 250uA; for a 3.3V microcontroller, the current of “weak pull-up” transistor is about 150uA.

The second pull-up transistor, called “very weak pull-up”, turns on when the port latch is “1”. When the pin is not connected, this very weak pull-up source produces a weak pull-up current that pulls the pin high. For a 5V microcontroller, the current of “weak pull-up” transistor is about 18uA; for 3.3V microcontrollers, the current of “weak pull-up” transistor is about 5uA.

The third pull-up transistor is called “strong pull-up”. This pull-up transistor is used to speed up the low-to-high transition for quasi-bidirectional port pin when the port latch changes from logic “0” to logic “1”. When this occurs, the strong pull-up transistor keeps on for about two clocks to quickly pull the pin high.

Quasi-bidirectional port (weak pull-up) has a Schmitt trigger and an interference suppression circuit. To read the correct external state, quasi-bidirectional port (weak pull-up) should latch to ‘1’ before reading.

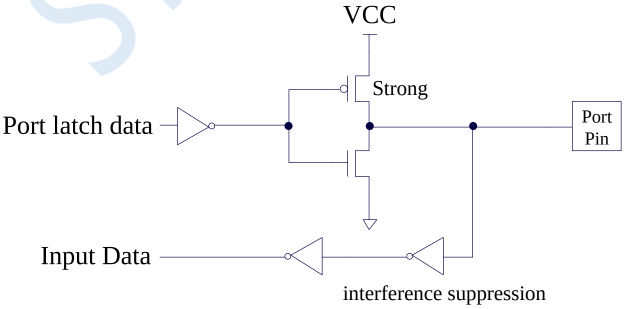
The structure of quasi-bidirectional port (weak pull-up) output is shown below:



9.3.2 Push-Pull Output

The pull-down structure of the strong push-pull output mode is the same as the pull-down structure of the open-drain output mode and quasi-bidirectional mode. However, the push-pull output mode can provide a sustained strong pull-up when the latch is logic “1”. Push-pull mode is generally used when more drive current is required.

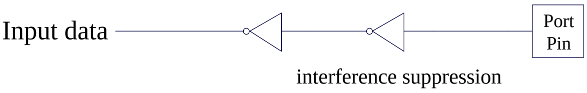
The structure of strong push-pull pin configuration is shown below:



9.3.3 High Impedance Input

The current can neither flow in nor flow out.
The input port has a Schmitt trigger input and an interference suppression circuit.

The structure of high impedance input pin configuration is shown below:



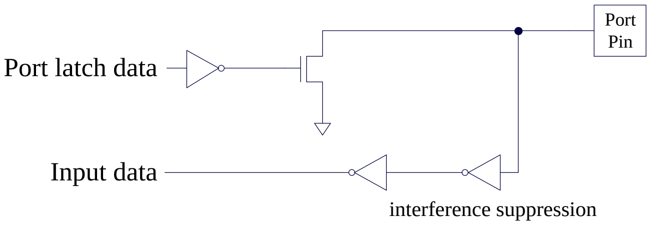
9.3.4 Open-Drain Output

The open-drain mode can be used for both reading external status and 300utputting high or low level. To read the external state correctly or output a high level, the external pull-up resistor should be connected.

All pull-up transistors are turned off in the open-drain output configuration when the port latch is logic “0”. There must be an external pull-up resistor in this configuration when the port outputs a logic high, typically the port pin is externally connected to VCC through a resistor. An open-drain I/O port pin can read the external state if the external pull-up resistor is connected, and the open-drain mode I/O port pin can be used as input mode. The pull-down structure in this way is the same as quasi-bidirectional mode.

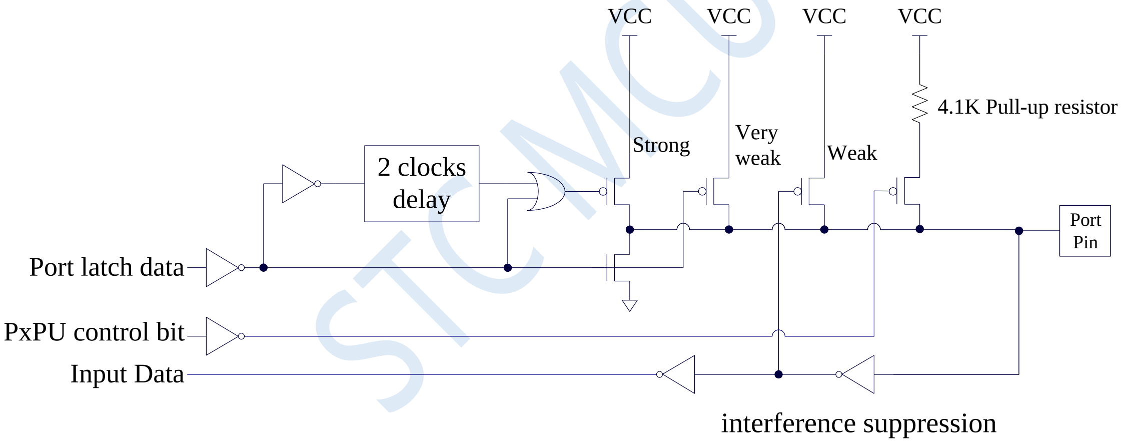
The open drain port has a Schmitt trigger input and an interference suppression circuit.

The structure of open drain port configuration is shown below:



9.3.5 4.1K Pull-up Resistor

A pull-up resistor of approximately 4.1K can be enabled internally in all I/O ports of the STC8 series (due to manufacturing errors, the range of the pull-up resistor may be 3K to 5K).



Pull-up Resistor Control Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P0PU	FE10H	P07PU	P06PU	P05PU	P04PU	P03PU	P02PU	P01PU	P00PU
P1PU	FE11H	P17PU	P16PU	P15PU	P14PU	P13PU	P12PU	P11PU	P10PU
P2PU	FE12H	P27PU	P26PU	P25PU	P24PU	P23PU	P22PU	P21PU	P20PU
P3PU	FE13H	P37PU	P36PU	P35PU	P34PU	P33PU	P32PU	P31PU	P30PU
P4PU	FE14H	P47PU	P46PU	P45PU	P44PU	P43PU	P42PU	P41PU	P40PU
P5PU	FE15H	-	-	-	P54PU	P53PU	P52PU	P51PU	P50PU
P6PU	FE16H	P67PU	P66PU	P65PU	P64PU	P63PU	P62PU	P61PU	P60PU
P7PU	FE17H	P77PU	P76PU	P75PU	P74PU	P73PU	P72PU	P71PU	P70PU

Internal 4.1K pull-up resistor control bit (Note: The pull-up resistors on the P3.0 and P3.1 ports may be slightly smaller)

0: Disable 4.1K pull-up resistor inside the port
1: Enable 4.1K pull-up resistor inside the port

9.3.6 How to set the external output speed of the I/O port

When users need the I/O port to output a faster frequency, they can increase the I/O port drive current and increase the I/O port level conversion speed to increase the I/O port output speed.

Setting the PxSR register can be used to control the I/O port level conversion speed. When it is set to 0, the corresponding I/O port is fast flipping, and when it is set to 1, it is slow flipping.

Set the PxDR register, which can be used to control the drive current of the I/O port. When set to 1, I/O output is general drive current, and when set to 0, it is strong drive current.

9.3.7 How to set I/O port current drive capability

If you need to change the current drive capability of the I/O port, you can do so by setting the PxDR register

Set the PxDR register, which can be used to control the drive current of the I/O port. When set to 1, I/O output is general drive current, when set to 0, it is strong drive current

9.3.8 How to reduce the external radiation of I/O ports

Because the PxSR register is set, it can be used to control the I/O port level conversion speed, and the PxDR register can be used to control the I/O port drive current.

When the external radiation of the I/O port needs to be reduced, the PxSR register needs to be set to 1 to reduce the I/O port level conversion speed, and the PxDR register needs to be set to 1 to reduce the I/O drive current, and finally reduce I /O port external radiation

9.4 Example Routines

9.4.1 Port Mode Setting

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr     P0M0        =   0x94;
sfr     P0M1        =   0x93;
sfr     P1M0        =   0x92;
sfr     P1M1        =   0x91;
sfr     P2M0        =   0x96;
sfr     P2M1        =   0x95;
sfr     P3M0        =   0xb2;
sfr     P3M1        =   0xb1;
sfr     P4M0        =   0xb4;
sfr     P4M1        =   0xb3;
sfr     P5M0        =   0xca;
sfr     P5M1        =   0xc9;
sfr     P6M0        =   0xcc;
sfr     P6M1        =   0xcb;
sfr     P7M0        =   0xe2;
sfr     P7M1        =   0xe1;

void main()
{
        P0M0 = 0x00;                                //Set P0.0 ~ P0.7 as bidirectional port mode
        P0M1 = 0x00;
        P1M0 = 0xff;                                //Set P1.0 ~ P1.7 as push-pull output mode
        P1M1 = 0x00;
        P2M0 = 0x00;                                //Set P2.0 ~ P2.7 as high impedance input mode
        P2M1 = 0xff;
        P3M0 = 0xff;                                //Set P3.0 ~ P3.7 as open-drain mode
        P3M1 = 0xff;

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P0M0        DATA        094H
P0M1        DATA        093H
P1M0        DATA        092H
P1M1        DATA        091H
P2M0        DATA        096H
P2M1        DATA        095H
P3M0        DATA        0B2H
P3M1        DATA        0B1H
P4M0        DATA        0B4H
P4M1        DATA        0B3H
P5M0        DATA        0CAH
P5M1        DATA        0C9H
P6M0        DATA        0CCH
```

```
P6M1            DATA            0CBH
P7M0            DATA            0E2H
P7M1            DATA            0E1H

                ORG             0000H
                LJMP            MAIN

                ORG             0100H
MAIN:
                MOV             SP, #5FH

                MOV             P0M0,#00H                   ;Set P0.0 ~ P0.7 as bidirectional port mode
                MOV             P0M1,#00H
                MOV             P1M0,#0FFH                  ;Set P1.0 ~ P1.7 as push-pull output mode
                MOV             P1M1,#00H
                MOV             P2M0,#00H                   ;Set P2.0 ~ P2.7 as high impedance input mode
                MOV             P2M1,#0FFH
                MOV             P3M0,#0FFH                  ;Set P3.0 ~ P3.7 as open-drain mode
                MOV             P3M1,#0FFH

                JMP             $

                END
```

9.4.2 Reading and Writing Operation of Bidirection Port

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P0M0        =       0x94;
sfr         P0M1        =       0x93;
sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;
sbit        P00         =       P0^0;

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
```

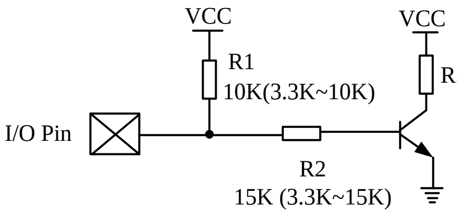
```
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P0M0 = 0x00;                                  //Set P0.0 ~ P0.7 as bidirectional port mode
    P0M1 = 0x00;

    P00 = 1;                                      //P0.0 output high level
    P00 = 0;                                      //P0.0 output low level

    P00 = 1;                                      //Enable the internal weak pull-up resistor before reading the port
    _nop_();                                      //Wait for two clocks
    _nop_();                                      //
    CY = P00;                                     //Read port status

    while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P0M0        DATA        094H
P0M1        DATA        093H
P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN

            ORG         0100H
MAIN:
            MOV         SP, #5FH
            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
            MOV         P3M1, #00H
            MOV         P4M0, #00H
            MOV         P4M1, #00H
            MOV         P5M0, #00H
            MOV         P5M1, #00H

            MOV         P0M0,#00H                 ;Set P0.0 ~ P0.7 as bidirectional port mode
            MOV         P0M1,#00H
```

```
        SETB        P0.0                    ;P0.0 output high level
        CLR         P0.0                    ;P0.0 output low level

        SETB        P0.0                    ;Enable the internal weak pull-up resistor before reading the port
        NOP                                 ;Wait for two clocks
        NOP
        MOV         C,P0.0                  ;Read port status

        JMP         $

        END
```

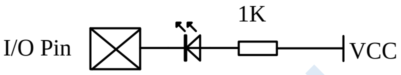
9.5 A Typical Circuit Controlled by Triode



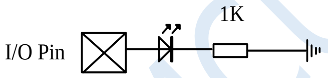
For pull-up control, it is recommended to add a pull-up resistor R1 (3.3K ～ 10K). If pull-up resistor R1 (3.3K ～ 10K) is not connected, it is recommended that the value of R2 be above 15K, or use strong push-pull output mode.

9.6 Typical Control Circuit of LED

For Quasi-Bidirectional (weak pull-up) I/O, you can drive the light-emitting diode using sink current mode, where the current limiting resistance should be greater than 1K oms, preferably not less than 470Ω.

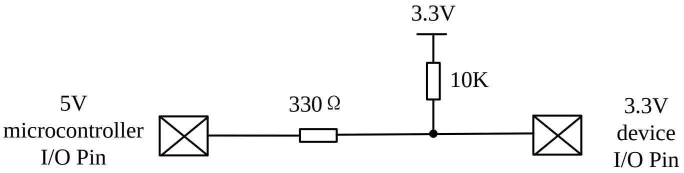


For push-pull (strong pull-up) I/O, you can drive the light-emitting diode with pull current mode.

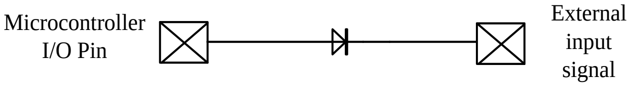


9.7 Interconnection of 3V/5V Devices in Mixed Voltage Power Supply System

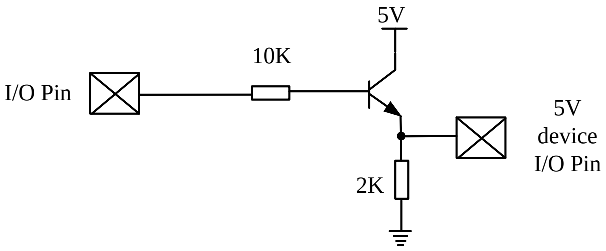
When STC's 5V microcontroller is connected to a 3.3V device, the corresponding I/O port of the 5V microcontroller can be connected with a 330Ω current limiting resistor to the 3.3V device I/O port in order to prevent the 3.3V device from withstanding 5V. The I/O port of the microcontroller is set to open-drain mode, and the internal pull-up resistor is disconnected. The corresponding 3.3V device I/O port is connected to 3.3V via with a 10K pull-up resistor. Then the high level is 3.3V, and the low level is 0V.



When STC's 3V microcontroller is connected to a 5V device and the corresponding I/O port is used as an input, an isolation diode can be connected in series to the I/O port to isolate the high voltage part in order to prevent the 3V microcontroller from bearing 5V. When the external signal voltage is higher than the microcontroller operating voltage, the isolation diode will cutoff, the state of the read I/O port is high because the I/O port is pulled up to a high level internally. When the external signal voltage is low, the isolation diode will turn on and the I/O port is clamped at 0.7V. The microcontroller reads I/O port low status as the voltage is less than 0.8V.



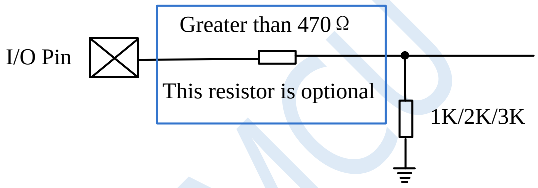
When STC's 3V microcontroller is connected to a 5V device and the corresponding I/O port is used as an output, it can be isolated with an NPN transistor in order to prevent the 3V microcontroller from bearing 5V. The circuit is as follows.



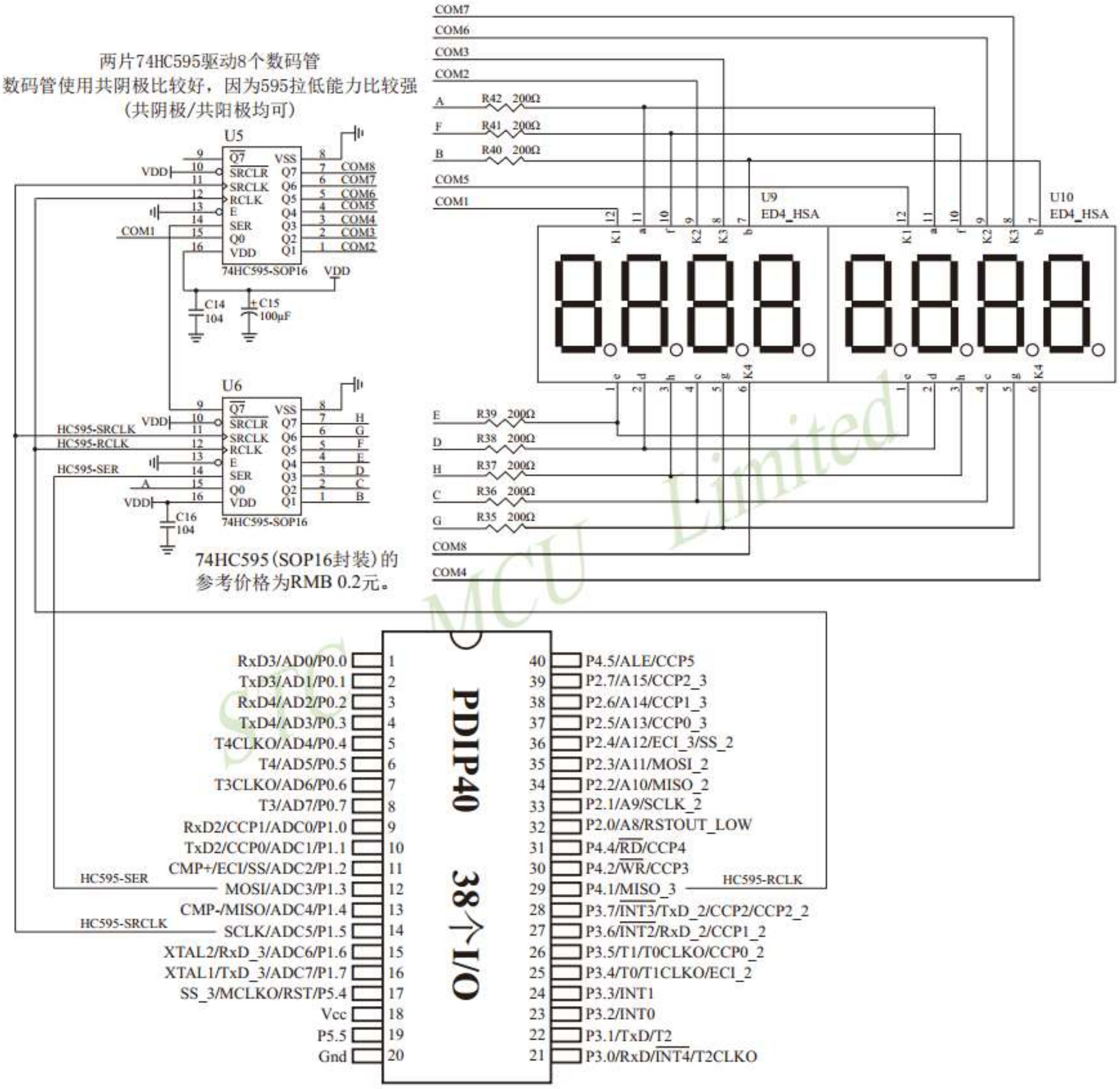
9.8 Make I/O Port Output Low When Power on Reset

The general I/O port which is a weak pull-up (quasi-bidirectional) port will output high level when the traditional 8051 microcontroller is powered on and reset. Many practical applications require that certain I/O ports be low level output when powered on, otherwise the system (such as the motor) controlled by the microcontroller will malfunction. The STC microcontroller’s I/O ports have weak pull-up output mode and strong push-pull output mode, which can easily solve this problem.

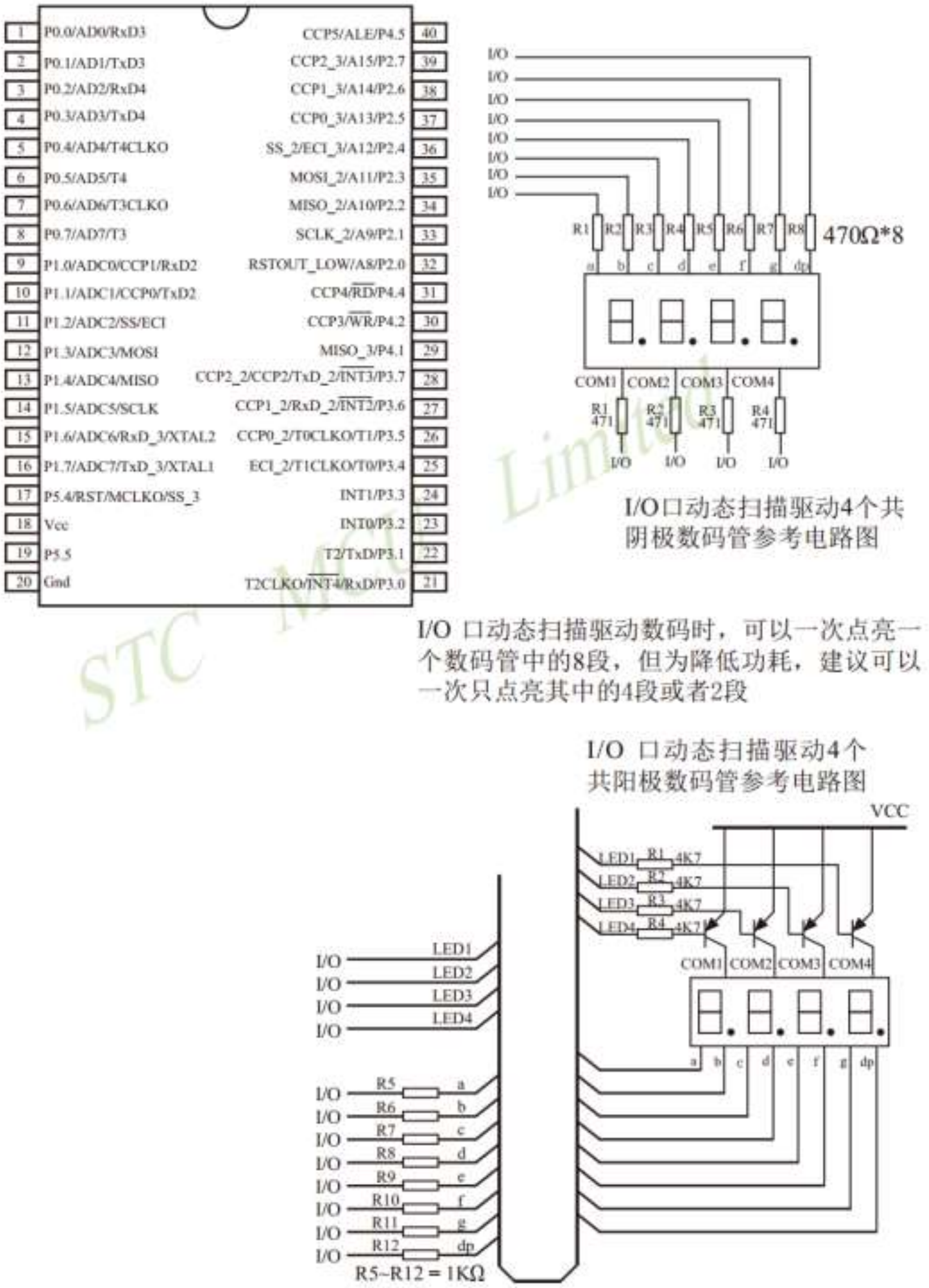
Now you can connect a pull-down resistor (1K, 2K or 3K) to the I/O port of STC microcontroller. The I/O port of the microcontroller is internal weak pull-up (quasi-bidirectional)/high-level output when the microcontroller is powered on reset. It cannot be pulled high because the internal pull-up capability is limited and the external pull-down resistor is small. Therefore the I/O port is externally low when power is reset. If you want to drive this I/O port to a high level, you can set this I/O port in strong push-pull output mode, under this circumstance, the I/O port drive current can reach 20mA, so the port can be definitely driven high.



9.9 Circuit Diagram of Driving 8 Digital LEDs using 74HC595



9.10 Digital LEDs Driven Directly by I/O Port Circuit



9.11 LCD Segment LCD Driven Directly by I/O Port Circuit

An external LCD driver IC is needed when you design a product having segment LCD display requirement using MCU without LCD driver is used, which will increase cost. In fact, many small projects, such as a large number of small appliances, do not need to display a lot of segment codes. There are usually four 8 colons with a decimal point or clock: ":", so if you use the IO port to scan and display directly, it will reduce costs and work more reliably.

However, this solution is not suitable for driving too many segments (occupying too many IOs), and it is not suitable for very low power consumption occasions (driving will have hundreds of uA current).

The simple principle of segment code LCD driving is shown in Figure 1.

LCD is a special kind of liquid crystal. The arrangement direction of the crystal will be reversed under the action of an electric field, which changes its light transmittance, so that the display content can be seen. LCD has a torsional voltage threshold. The content will be displayed if the voltage across the LCD is higher than this threshold, and the content will not be displayed if the voltage is lower than this threshold. There are usually 3 parameters in LCD: working voltage, DUTY (corresponding to COM number) and BIAS (ie bias, corresponding threshold). For example, ‘3.0V, 1/4 DUTY, 1/3 BIAS’ means that the LCD display voltage is 3.0V, 4 COM, the threshold is about 1.5V. The content will be displayed if the voltage across a certain LCD segment is 3.0V, and the content will not be displayed if the voltage is 1.0V. However, the LCD's response to the driving voltage is not very sensitive. For example, the display may be faint if the voltage is 2V. This is usually called a ‘ghost image’. Therefore, it is necessary to ensure that the voltage is larger than the threshold value if you want to display something, and the voltage is smaller than the threshold value if you do not display anything.

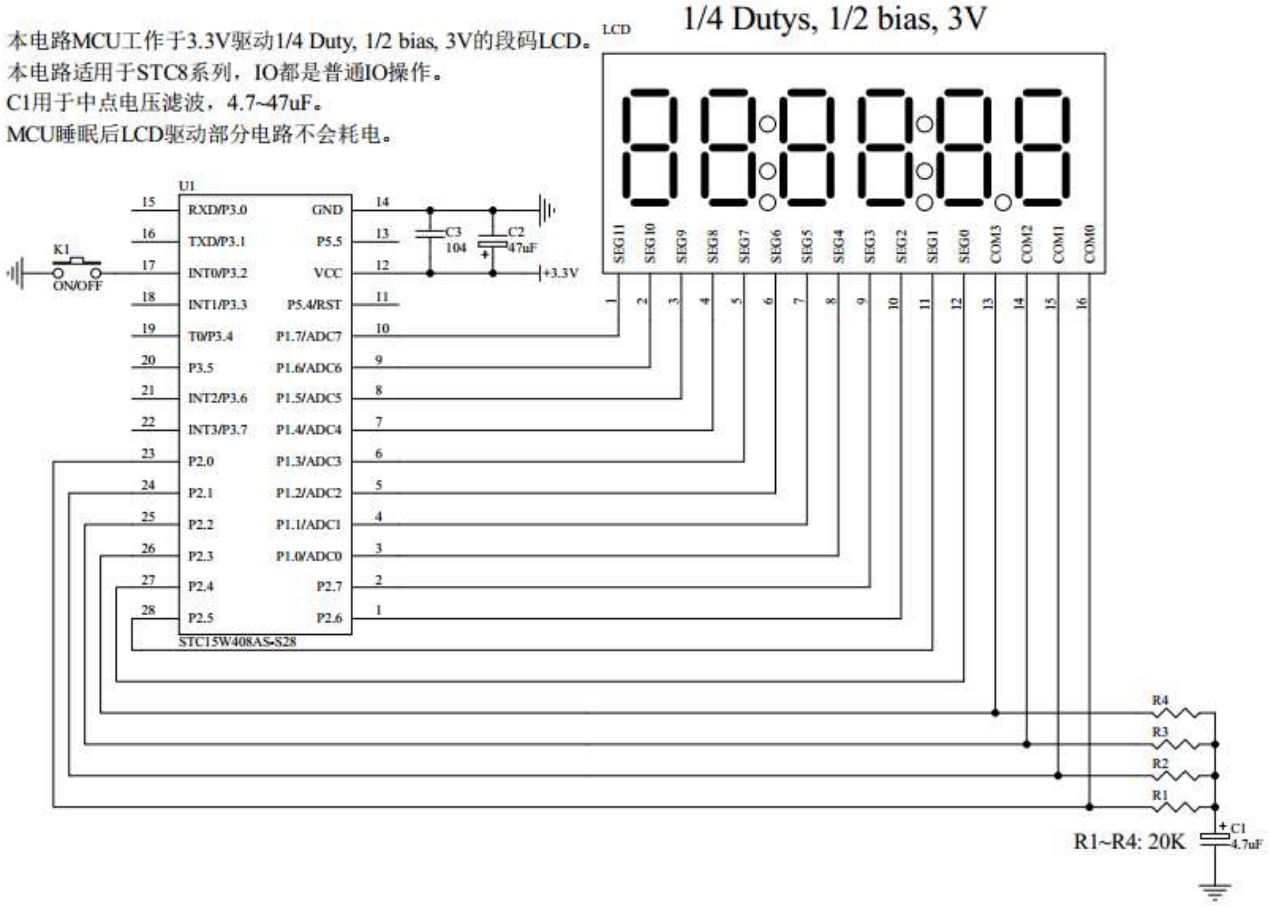
Note: The LCD should be driven by AC, and DC voltage cannot be applied to the two ends of the LCD. Otherwise it will be damaged for a long time DC applying. The average voltage of the driving voltage applied to the LCD must be 0. Time-sharing is used to LCD. At any time, if one COM scan is valid, the other COM is invalid.

The scheme circuit for driving ‘1/4Duty, 1/ 2BIAS, 3V’ is shown in Figure 1. The scanning principle of LCD is shown in Figure 3. The MCU works at 3.0V or 3.3V. Each COM is connected with a 20K resistor in series to a capacitor C1 aiming to obtain the midpoint voltage of 1/2VDD after RC filtering. When it is the turn of a COM scan, the connected IO is set to push-pull output mode, and the remaining COMs are set to high impedance. If the SEG connected to this COM is not used to display, the SEG output is in phase with the COM, and if it is not used to display, it is inverted. After scanning is finished, the I/O corresponding to this COM is set to high impedance. Each COM is connected to the 1/2VDD voltage on the capacitor C1 through a 20K resistor. The SEG outputs high or low level according to whether it is used to display or not. The voltage applied to the LCD segment is +-VDD when displayed, and +-1/2VDD when not displayed, which can ensure that the average DC voltage across the LCD is 0.

The circuit for driving the ‘1/4Duty, 1/3BIAS, 3V’ is shown in Figure 4. The scanning principle of LCD is shown in Figure 5. The MCU works at 5V. The IOs connected to SEG output 1.5V and 3.5V through the resistor divider. The IOs connected to COM output 0.5V, 2.5V (at high impedance), 4.5V. The common point of the voltage-dividing resistor is connected to a capacitor C1 to abtain a mid-point voltage of 1/2VDD after RC filtering. When it is the turn of a certain COM scan, the IO is set to push-pull output mode. If the SEG connected to this COM is not used to display, the SEG output is in phase with COM, and if it is used to display, it is inverted. After scanning is finished, the I/O corresponding to this COM is set to high impedance. This COM is connected to a 2.5V voltage through a 47K resistor. The SEG outputs high or low level according to whether it is used to display or not. The voltage applied to the LCD is +-3.0V when displayed, and +-1.0V when not displaying, which can meet the LCD scanning requirements.

When sleep of power saving is required, all IOs used to dirve COMs and SEGs output low level, and the extra current will not appear in LCD drive part.

Figure 1 Circuit for driving a ‘1/4Duty, 1/2BIAS, 3V’ LCD



本电路MCU工作于3.3V驱动1/4 Duty, 1/2 bias, 3V的段码LCD。
本电路适用于STC8系列，IO都是普通IO操作。
C1用于中点电压滤波，4.7~47uF。
MCU睡眠后LCD驱动部分电路不会耗电。

Figure 2 Segment name

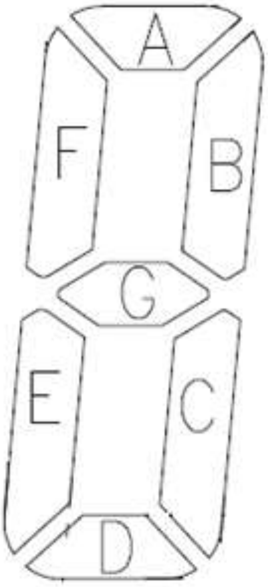


Figure 3 Principle of ‘1/4Duty, 1/2BIAS’ scanning

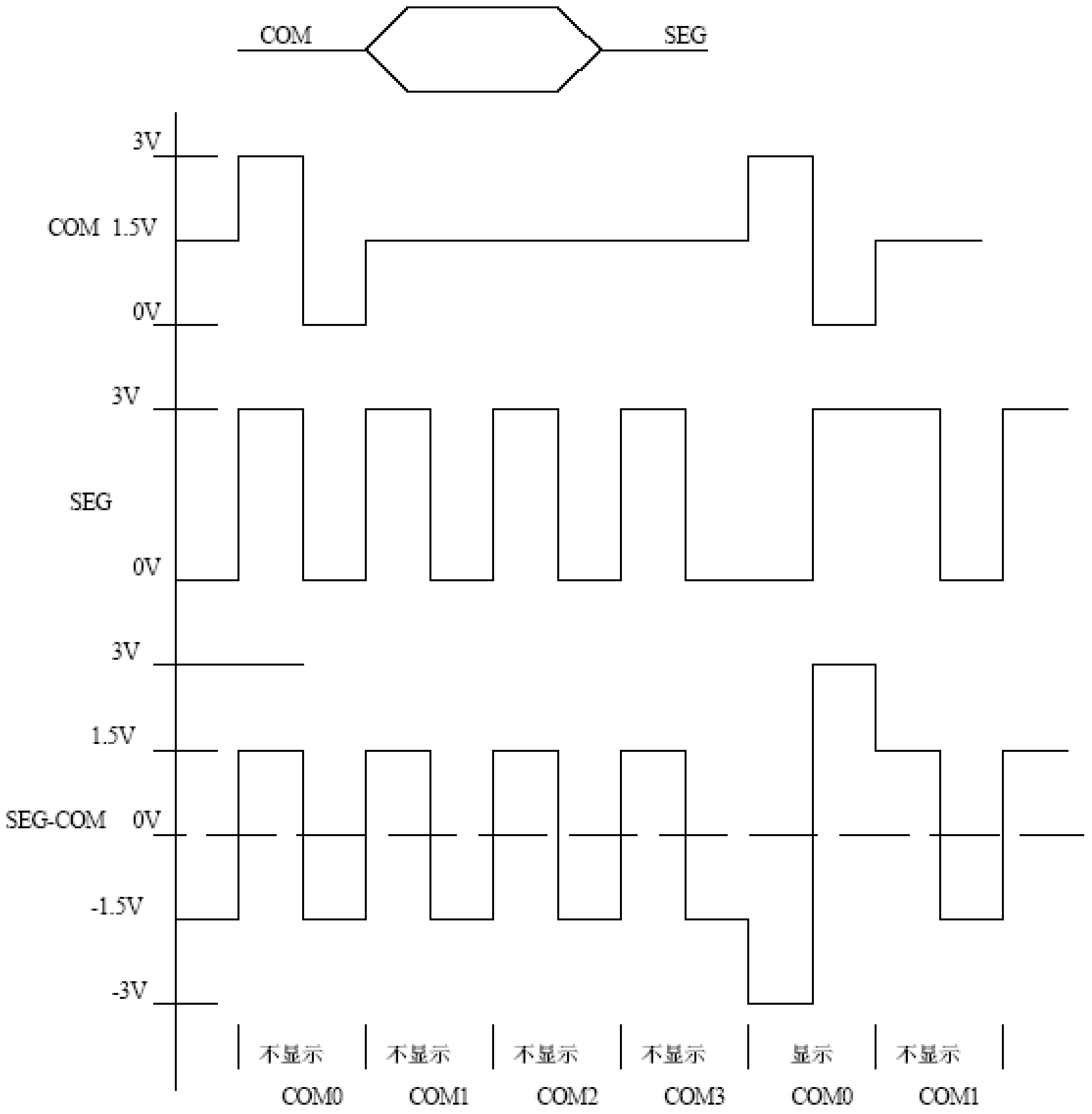
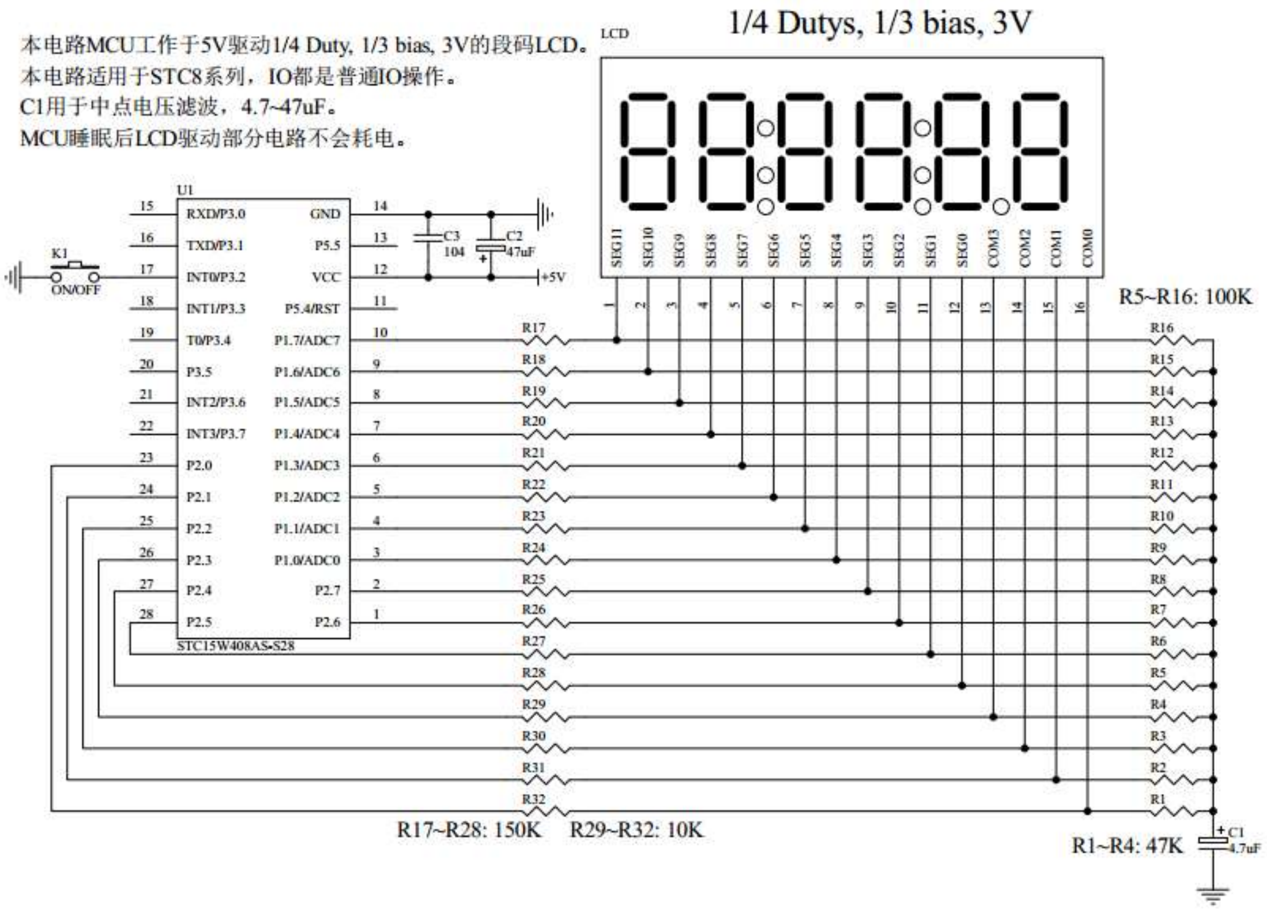
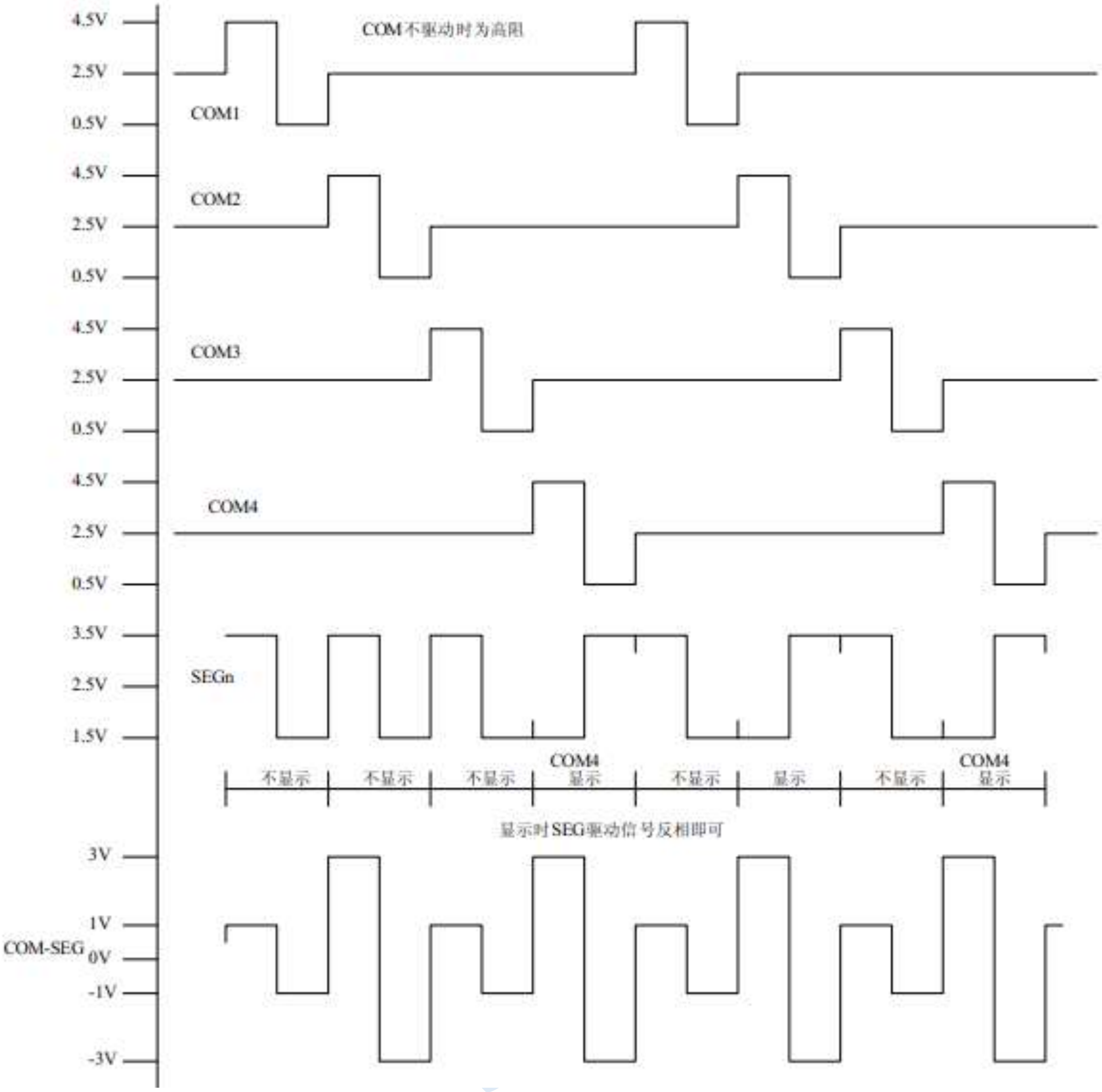


Figure 4 Driving circuit of ‘1/4Duty, 1/3BIAS, 3V’ LCD



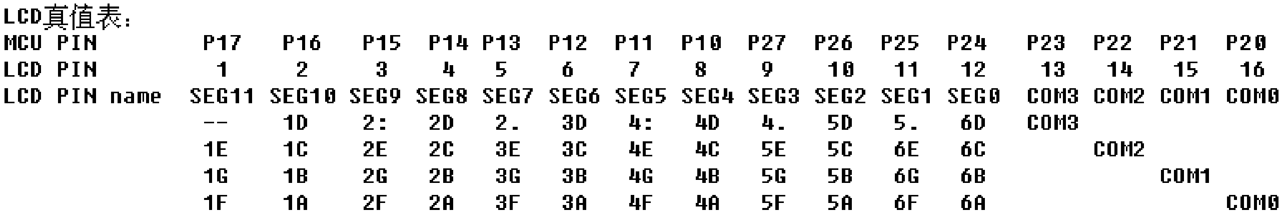
本电路MCU工作于5V驱动1/4 Duty, 1/3 bias, 3V的段码LCD。
本电路适用于STC8系列，IO都是普通IO操作。
C1用于中点电压滤波，4.7~47uF。
MCU睡眠后LCD驱动部分电路不会耗电。

Figure 5 Principle of ‘1/4Duty, 1/3BIAS’ scanning



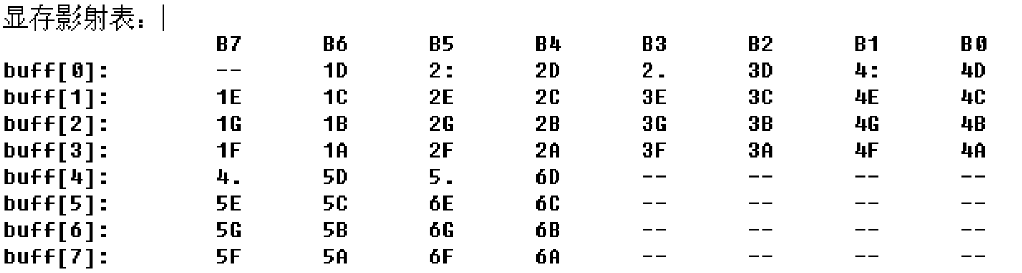
For ease of use, the display contents are stored in display memory where each bit corresponds to the LCD segment one by one, as shown in Figure 6.

Figure 6 LCD truth table and memory mapping table



LCD真值表：

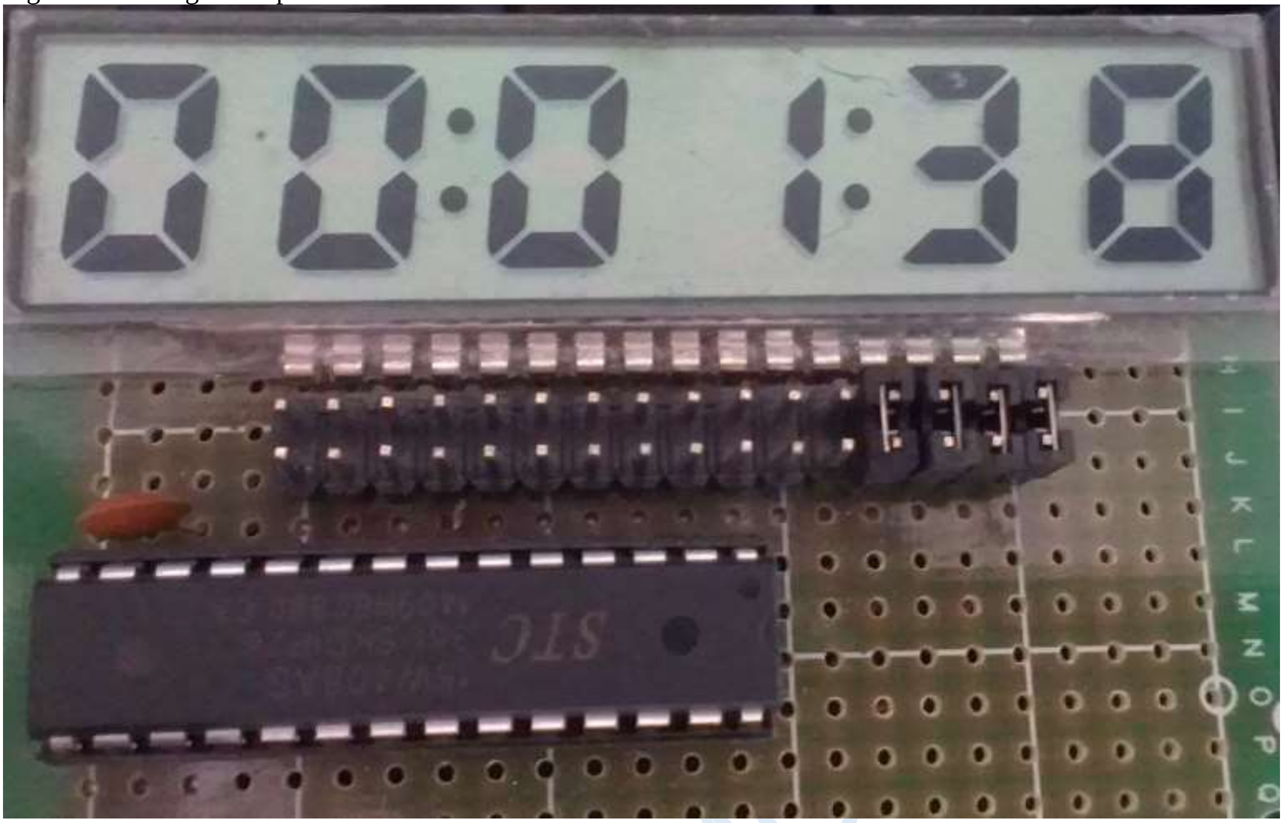
MCU PIN	P17	P16	P15	P14	P13	P12	P11	P10	P27	P26	P25	P24	P23	P22	P21	P20
LCD PIN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
LCD PIN name	SEG11	SEG10	SEG9	SEG8	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0	COM3	COM2	COM1	COM0
	--	1D	2:	2D	2.	3D	4:	4D	4.	5D	5.	6D	COM3			
	1E	1C	2E	2C	3E	3C	4E	4C	5E	5C	6E	6C		COM2		
	1G	1B	2G	2B	3G	3B	4G	4B	5G	5B	6G	6B			COM1	
	1F	1A	2F	2A	3F	3A	4F	4A	5F	5A	6F	6A				COM0



显存影射表：|

	B7	B6	B5	B4	B3	B2	B1	B0
buff[0]:	--	1D	2:	2D	2.	3D	4:	4D
buff[1]:	1E	1C	2E	2C	3E	3C	4E	4C
buff[2]:	1G	1B	2G	2B	3G	3B	4G	4B
buff[3]:	1F	1A	2F	2A	3F	3A	4F	4A
buff[4]:	4.	5D	5.	6D	--	--	--	--
buff[5]:	5E	5C	6E	6C	--	--	--	--
buff[6]:	5G	5B	6G	6B	--	--	--	--
buff[7]:	5F	5A	6F	6A	--	--	--	--

Figure 7: Driving effect photo



Only two functions are required in the LCD scanning program:
1. LCD segment code scan function
void LCD_scan(void)

The program calls this function at a certain interval, and it will display the contents of the LCD display buffer on the LCD. It takes 8 calling cycles to scan all of them. The calling interval is generally 1~ 2ms. The scanning cycle is 8ms if 1ms is used. The refresh rate is 125Hz.

2. LCD segment code display buffer loading function
void LCD_load(u8 n,u8 dat)

This function is used to put the displayed numbers or characters in the LCD display buffer. For example, LCD_load (1,6) is to display the number 6 at the first digit position. It supports the display of 0 ~ 9, A ~ F. You can add them by yourself if you need other characters.

In addition, macros are used to display, extinguish, or flash colons or decimal points.

C language code

```
/*****************Function description******************
STC15 series of microcontrollers are used to test segment LCD driven by I/O directly (6 8-word LCDs, 1/4 Dutys, 1/3 bias).
Time (hours, minutes and seconds) is displayed after power-on.
P3.2 is connected to ground via a switch to enter sleep or wake up.
******************************************/

#include "reg51.h"
#include "intrins.h"

typedef     unsigned char       u8;
typedef     unsigned int        u16;
typedef     unsigned long       u32;

sfr AUXR = 0x8e;
sfr P1M1 = 0x91;
sfr P1M0 = 0x92;
sfr P2M1 = 0x95;
sfr P2M0 = 0x96;

/********************Local constant declaration**************************/
#define MAIN_Fosc               11059200L                   //Define the main clock

#define DIS_BLACK               0x10
#define DIS_                    0x11
#define DIS_A                   0x0A
#define DIS_B                   0x0B
#define DIS_C                   0x0C
#define DIS_D                   0x0D
#define DIS_E                   0x0E
#define DIS_F                   0x0F

#define LCD_SET_DP2             LCD_buff[0] |=  0x08
#define LCD_CLR_DP2             LCD_buff[0] &= ~0x08
#define LCD_FLASH_DP2           LCD_buff[0] ^=  0x08

#define LCD_SET_DP4             LCD_buff[4] |=  0x80
#define LCD_CLR_DP4             LCD_buff[4] &= ~0x80
#define LCD_FLASH_DP4           LCD_buff[4] ^=  0x80

#define LCD_SET_2M              LCD_buff[0] |=  0x20
#define LCD_CLR_2M              LCD_buff[0] &= ~0x20
#define LCD_FLASH_2M            LCD_buff[0] ^=  0x20

#define LCD_SET_4M              LCD_buff[0] |=  0x02
#define LCD_CLR_4M              LCD_buff[0] &= ~0x02
#define LCD_FLASH_4M            LCD_buff[0] ^=  0x02

#define LCD_SET_DP5             LCD_buff[4] |=  0x20
#define LCD_CLR_DP5             LCD_buff[4] &= ~0x20
#define LCD_FLASH_DP5           LCD_buff[4] ^=  0x20

#define P1n_standard(bitn)      P1M1 &= ~(bitn), P1M0 &= ~(bitn)
#define P1n_push_pull(bitn)     P1M1 &= ~(bitn), P1M0 |= (bitn)
#define P1n_pure_input(bitn)    P1M1 |= (bitn), P1M0 &= ~(bitn)
#define P1n_open_drain(bitn)    P1M1 |= (bitn), P1M0 |= (bitn)

#define P2n_standard(bitn)      P2M1 &= ~(bitn), P2M0 &= ~(bitn)
#define P2n_push_pull(bitn)     P2M1 &= ~(bitn), P2M0 |= (bitn)
#define P2n_pure_input(bitn)    P2M1 |= (bitn), P2M0 &= ~(bitn)
#define P2n_open_drain(bitn)    P2M1 |= (bitn), P2M0 |= (bitn)

/********************Local variable declaration**************************/
u8 cnt_500ms;
u8 second,minute,hour;
bit B_Second;
bit B_2ms;
u8 LCD_buff[8];
u8 scan_index;

/********************Local function declaration**************************/
void LCD_load(u8 n,u8 dat);
void LCD_scan(void);
void LoadRTC(void);
void delay_ms(u8 ms);

/********************main function**************************/
void main(void)
```

```
{
        u8      i;

        AUXR = 0x80;
        TMOD = 0x00;
        TL0 = (65536 - (MAIN_Fosc / 500));
        TH0 = (65536 - (MAIN_Fosc / 500)) >> 8;
        TR0 = 1;
        ET0 = 1;
        EA = 1;

                                                        //Initialize LCD memory
        for(i=0; i<8; i++) LCD_buff[i] = 0;
        P2n_push_pull(0xf0);
        P1n_push_pull(0xff);                            //segment is set as push-pull output mode

        LCD_SET_2M;                                     //Display hour-minute division interval:
        LCD_SET_4M;                                     //Display minute-second division interval:
        LoadRTC();                                      //Display time

        while (1)
        {
                PCON |= 0x01;                           //Enter Idle mode, wake up and exit by Timer0 2ms
                _nop_();
                _nop_();
                _nop_();

                if(B_2ms)                               //2ms beat
                {
                        B_2ms = 0;

                        if(++cnt_500ms >= 250)          //reach to 500ms
                        {
                                cnt_500ms = 0;
                //              LCD_FLASH_2M;           //Flashing hour-minute interval:
                //              LCD_FLASH_4M;           //Flashing minute-second interval:

                                B_Second = ~B_Second;
                                if(B_Second)
                                {
                                        if(++second >= 60)      //reach to 1 minute
                                        {
                                                second = 0;
                                                if(++minute >= 60)      //reach to 1 hour
                                                {
                                                        minute = 0;
                                                        if(++hour >= 24) hour = 0;  //reach to 24 hours
                                                }
                                        }
                                        LoadRTC();      //Display time
                                }
                        }

                        if(!INT0)                       //key is pressed, ready to sleep
                        {
                                LCD_CLR_2M;             //Display hour-minute division interval:
                                LCD_CLR_4M;             //Display minute-second division interval:
                                LCD_load(1,DIS_BLACK);
                                LCD_load(2,DIS_BLACK);
                                LCD_load(3,0);
                                LCD_load(4,0x0F);
                                LCD_load(5,0x0F);
                                LCD_load(6,DIS_BLACK);

                                while(!INT0) delay_ms(10);              //Waiting for the key to be released
                                delay_ms(50);
                                while(!INT0) delay_ms(10);              //Waiting for the key to be released once more

                                TR0 = 0;                                //关闭定时器
                                IE0 = 0;                                //外中断 0 标志位
                                EX0 = 1;                                //INT0 Enable
                                IT0 = 1;                                //INT0 下降沿中断

                                P1n_push_pull(0xff);                    //com 和 seg 全部输出 0
                                P2n_push_pull(0xff);
                                P1 = 0;
                                P2 = 0;

                                PCON |= 0x02;                           //Sleep
                                _nop_();
                                _nop_();
                                _nop_();

                                LCD_SET_2M;                             //Display hour-minute division interval:
                                LCD_SET_4M;                             //Display minute-second division interval:
                                LoadRTC();                              //Display time
                                TR0 = 1;                                //Open the timer
                                while(!INT0) delay_ms(10);              //Waiting for the key to be released
                                delay_ms(50);
                                while(!INT0) delay_ms(10);              //Waiting for the key to be released once more
                            }
                }
        }
    }

    /********************delay function**************************/
    void delay_ms(u8 ms)
    {
        unsigned int i;
        do{
                i = MAIN_Fosc / 13000;
                while(--i);                                     //14T per loop
        }while(--ms);
    }

    /********************* Timer0 interrupt function***********************/
    void timer0_int (void) interrupt 1
    {
        LCD_scan();
        B_2ms = 1;
    }

    /********************* INT0 interrupt function ***********************/
    void INT0_int (void) interrupt 0
    {
        EX0 = 0;
        IE0  = 0;
    }

    /****************** LCD Segment code scan function *************************/
```

```
void  LCD_scan(void)                                       //5us @22.1184MHZ
{
        u8 code T_COM[4]={0x08,0x04,0x02,0x01};
        u8 j;

        j = scan_index >> 1;
        P2n_pure_input(0x0f);      //All COM outputs are high impedance, and COM’ s voltage is the midpoint
        if(scan_index & 1)                                 // Reverse scan
        {
                P1 = ~LCD_buff[j];
                P2 = ~(LCD_buff[j|4] & 0xf0);
        }
        else                                               // Normal phase scan
        {
                P1 = LCD_buff[j];
                P2 = LCD_buff[j|4] & 0xf0;
        }
        P2n_push_pull(T_COM[j]);                           //A COM is set as push-pull output mode
        if(++scan_index >= 8) scan_index = 0;
}

/****************** Load display functions for numbers 1 to 6 ***************************/
void  LCD_load(u8 n, u8 dat)                               // n is the number， dat is the number to be displayed
{
        u8 code t_display[]={                              // Standard font
                //0    1    2    3    4    5    6    7    8    9    A    B    C    D    E    F
                0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F,0x77,0x7C,0x39,0x5E,0x79,0x71,
                //black  -
                0x00,0x40
        };
        u8 code T_LCD_mask[4]  = {~0xc0,~0x30,~0x0c,~0x03};
        u8 code T_LCD_mask4[4] = {~0x40,~0x10,~0x04,~0x01};
        u8 i,k;
        u8 *p;

        if((n == 0) || (n > 6)) return;
        i = t_display[dat];

        if(n <= 4)                                         //1~4
        {
                n--;
                p = LCD_buff;
        }
        else
        {
                n = n - 5;
                p = &LCD_buff[4];
        }

        k = 0;
        if(i & 0x08) k |= 0x40;                            //D
        *p = (*p & T_LCD_mask4[n]) | (k>>2*n);
        p++;

        k = 0;
        if(i & 0x04) k |= 0x40;                            //C
        if(i & 0x10) k |= 0x80;                            //E
        *p = (*p & T_LCD_mask[n]) | (k>>2*n);
        p++;
        k = 0;
        if(i & 0x02) k |= 0x40;                                //B
        if(i & 0x40) k |= 0x80;                                //G
        *p = (*p & T_LCD_mask[n]) | (k>>2*n);
        p++;

        k = 0;
        if(i & 0x01) k |= 0x40;                                //A
        if(i & 0x20) k |= 0x80;                                //F
        *p = (*p & T_LCD_mask[n]) | (k>>2*n);
}

/********************Display time ***************************/
void LoadRTC(void)
{
        LCD_load(1,hour/10);
        LCD_load(2,hour%10);
        LCD_load(3,minute/10);
        LCD_load(4,minute%10);
        LCD_load(5,second/10);
        LCD_load(6,second%10);
}
```

Assembly code

```
;STC8 series of microcontrollers are used to test segment LCD driven by I/O directly (6 8-word LCDs, 1/4 Dutys, 1/3 bias).
;Time (hours, minutes and seconds) is displayed after power-on.

;*************************************************************
P0M1            DATA            0x93
P0M0            DATA            0x94
P1M1            DATA            0x91
P1M0            DATA            0x92
P2M1            DATA            0x95
P2M0            DATA            0x96
P3M1            DATA            0xB1
P3M0            DATA            0xB2
P4M1            DATA            0xB3
P4M0            DATA            0xB4
P5M1            DATA            0xC9
P5M0            DATA            0xC
P6M1            DATA            0xCB
P6M0            DATA            0xCC
P7M1            DATA            0xE1
P7M0            DATA            0xE2
AUXR            DATA            0x8E
INT_CLKO        DATA            0x8F
IE2             DATA            0xAF
P4              DATA            0xC0
T2H             DATA            0xD6
T2L             DATA            0xD7

;*************************************************************
DIS_BLACK       EQU             010H
DIS_            EQU             011H
DIS_A           EQU             00AH
DIS_B           EQU             00BH
DIS_C           EQU             00CH
```

```
DIS_D           EQU         00DH
DIS_E           EQU         00EH
DIS_F           EQU         00FH

B_2ms           BIT         20H.0                       ;2ms signal
B_Second        BIT         20H.1                       ;second signal
cnt_500ms       DATA        30H
second          DATA        31H
minute          DATA        32H
hour            DATA        33H
scan_index      DATA        34H

LCD_buff        DATA        40H                         ;40H~47H

;*************************************************************
                ORG         0000H
                LJMP        F_Main

                ORG         000BH
                LJMP        F_Timer0_Interrupt

;*************************************************************
                ORG         0100H
F_Main:
                CLR         A
                MOV         P3M1, A                     ;Set as a quasi-bidirectional port
                MOV         P3M0, A
                MOV         P5M1, A                     ;Set as a quasi-bidirectional port
                MOV         P5M0, A

                MOV         P1M1, #0                    ; segments are set as push-pull output mode
                MOV         P1M0, #0ffh
                ANL         P2M1, #NOT 0f0h             ; segments are set as push-pull output mode
                ORL         P2M0, #0f0h
                ORL         P2M1, #00fH  ; All COM outputs are high impedance, and COM’s voltage is the midpoint
                ANL         P2M0, #0f0H
                MOV         SP, #0D0H
                MOV         PSW, #0
                USING       0                           ;Select bank0 R0 ~ R7

;*************************************************************
                MOV         R2, #8
                MOV         R0, #LCD_buff
L_ClearLcdRam:
                MOV         @R0, #0
                INC         R0
                DJNZ        R2, L_ClearLcdRam

                LCALL       F_Timer0_init
                SETB        EA

;               ORL         LCD_buff, #020H             ;Display hour-minute division interval:
;               ORL         LCD_buff, #002H             ;Display minute-second division interval:

                MOV         hour, #12
                MOV         minute, #00
                MOV         second, #00
                LCALL       F_LoadRTC                   ;Display time

;*************************************************************
```

```
L_Main_Loop:
                JNB             B_2ms, L_Main_Loop              ;2ms beat
                CLR             B_2ms

                INC             cnt_500ms
                MOV             A, cnt_500ms
                CJNE            A, #250, L_Main_Loop
                                                                ;reach to 500ms
                MOV             cnt_500ms, #0;

                XRL             LCD_buff, #020H                 ;Flashing hour-minute interval:
                XRL             LCD_buff, #002H                 ;Flashing minute-second interval:

                CPL             B_Second
                JNB             B_Second, L_Main_Loop

                INC             second
                MOV             A, second
                CJNE            A, #60, L_Main_Load
                MOV             second, #0                      ; reach to 1 minute
                INC             minute
                MOV             A, minute
                CJNE            A, #60, L_Main_Load
                MOV             minute, #0;
                INC             hour
                MOV             A, hour
                CJNE            A, #24, L_Main_Load
                MOV             hour, #0                        ;reach to 24 hours
L_Main_Load:
                LCALL           F_LoadRTC                       ;Display time
                LJMP            L_Main_Loop

;**************************************************************

F_Timer0_init:
                CLR             TR0                             ; Stop counting
                ANL             TMOD, #0f0H
                SETB            ET0                             ; Enable interrupt
                ORL             TMOD, #0                        ; Working mode 0: 16-bit auto-reload
                ANL             INT_CLKO, #NOT 0x01             ; Does not output clock
                ORL             AUXR, #0x80                     ; 1T mode
                MOV             TH0, #HIGH (-22118)             ; 2ms
                MOV             TL0, #LOW  (-22118)             ;
                SETB            TR0                             ; Start operation
                RET

;**************************************************************
F_Timer0_Interrupt:             ;Timer0 1ms interrupt function
                PUSH            PSW                             ;push PSW into stack
                PUSH            ACC                             ;push ACC into stack
                PUSH            AR0
                PUSH            AR7
                PUSH            DPH
                PUSH            DPL

                LCALL           F_LCD_scan
                SETB            B_2ms

                POP             DPL
                POP             DPH
                POP             AR7
                POP             AR0
                POP             ACC                     ;pop ACC from stack
                POP             PSW                     ;pop PSW from stac
                RETI

;******************** Display time ***************************
F_LoadRTC:
                MOV             R6, #1                  ;LCD_load(1,hour/10);
                MOV             A, hour
                MOV             B, #10
                DIV             AB
                MOV             R7, A
                LCALL           F_LCD_load              ;R6 is the number, which is 1~6, R7 is the number to be displayed

                MOV             R6, #2                  ;LCD_load(2,hour%10);
                MOV             A, hour
                MOV             B, #10
                DIV             AB
                MOV             R7, B
                LCALL           F_LCD_load              ;R6 is the number, which is 1~6, R7 is the number to be displayed

                MOV             R6, #3                  ;LCD_load(3,minute/10);
                MOV             A, minute
                MOV             B, #10
                DIV             AB
                MOV             R7, A
                LCALL           F_LCD_load              ;R6 is the number, which is 1~6, R7 is the number to be displayed

                MOV             R6, #4                  ;LCD_load(4,minute%10);
                MOV             A, minute
                MOV             B, #10
                DIV             AB
                MOV             R7, B
                LCALL           F_LCD_load              ;R6 is the number, which is 1~6, R7 is the number to be displayed

                MOV             R6, #5                  ;LCD_load(5,second/10);
                MOV             A, second
                MOV             B, #10
                DIV             AB
                MOV             R7, A
                LCALL           F_LCD_load              ;R6 is the number, which is 1~6, R7 is the number to be displayed

                MOV             R6, #6                  ;LCD_load(6,second%10);
                MOV             A, second
                MOV             B, #10
                DIV             AB
                MOV             R7, B
                LCALL           F_LCD_load              ;R6 is the number, which is 1~6, R7 is the number to be displayed

                RET

;**************************************************************
T_COM:
                DB              008H, 004H, 002H, 001H

F_LCD_scan:
                MOV             A, scan_index           ;j = scan_index >> 1;
                CLR             C
                RRC             A
                MOV         R7, A                   ;R7 = j
                ADD         A, #LCD_buff
                MOV         R0, A                   ;R0 = LCD_buff[j]
                ORL         P2M1, #00fH             ;All COM outputs are high impedance, and COM’s voltage is the
midpoint
                ANL         P2M0, #0f0H

                MOV         A, scan_index
                JNB         ACC.0, L_LCD_Scan2      ;if(scan_index & 1)    // Reverse scan
                MOV         A, @R0                  ;P1 = ~LCD_buff[j];
                CPL         A
                MOV         P1, A
                MOV         A, R0                   ;P2 = ~(LCD_buff[j|4] & 0xf0);
                ADD         A, #4
                MOV         R0, A
                MOV         A, @R0
                ANL         A, #0f0H
                CPL         A
                MOV         P2, A
                SJMP        L_LCD_Scan3

L_LCD_Scan2:                                        ; Normal phase scan
                MOV         A, @R0                  ;P1 = LCD_buff[j];
                MOV         P1, A
                MOV         A, R0                   ;P2 = (LCD_buff[j|4] & 0xf0);
                ADD         A, #4
                MOV         R0, A
                MOV         A, @R0
                ANL         A, #0f0H
                MOV         P2, A

L_LCD_Scan3:
                MOV         DPTR, #T_COM            ;A COM is set as push-pull output mode
                MOV         A, R7
                MOVC        A, @A+DPTR
                ORL         P2M0, A
                CPL         A
                ANL         P2M1, A

                INC         scan_index              ;if(++scan_index == 8)      scan_index = 0;
                MOV         A, scan_index
                CJNE        A, #8, L_QuitLcdScan
                MOV         scan_index, #0

L_QuitLcdScan:
                RET

;****************** Standard font*************************
T_Display:
;                0   1   2   3   4   5   6   7   8   9   A   B   C   D   E   F
DB              03FH,006H,05BH,04FH,066H,06DH,07DH,007H,07FH,06FH,077H,07CH,039H,05EH,079H,071H
;               black  -
DB              000H,040H

;***************** Load display functions for numbers 1 to 6, and the algorithm is simple **************************
F_LCD_load:                                         ;R6 is the number, which is 1~6, R7 is the number to be displayed
                MOV         DPTR, #T_Display        ;i = t_display[dat];
                MOV         A, R7
                MOVC        A, @A+DPTR
                MOV         B, A                    ;the number to be displayed
                MOV         A, R6
                CJNE        A, #1, L_NotLoadChar1
                MOV         R0,                 #LCD_buff
                MOV         A, @R0
                MOV         C, B.3              ;D
                MOV         ACC.6, C
                MOV         @R0, A

                INC         R0
                MOV         A, @R0
                MOV         C, B.2              ;C
                MOV         ACC.6, C
                MOV         C, B.4              ;E
                MOV         ACC.7, C
                MOV         @R0, A

                INC         R0
                MOV         A, @R0
                MOV         C, B.1              ;B
                MOV         ACC.6, C
                MOV         C, B.6              ;G
                MOV         ACC.7, C
                MOV         @R0, A

                INC         R0
                MOV         A, @R0
                MOV         C, B.0              ;A
                MOV         ACC.6, C
                MOV         C, B.5              ;F
                MOV         ACC.7, C
                MOV         @R0, A
                RET

L_NotLoadChar1:
                CJNE        A, #2, L_NotLoadChar2
                MOV         R0,#LCD_buff
                MOV         A, @R0
                MOV         C, B.3              ;D
                MOV         ACC.4, C
                MOV         @R0, A

                INC         R0
                MOV         A, @R0
                MOV         C, B.2              ;C
                MOV         ACC.4, C
                MOV         C, B.4              ;E
                MOV         ACC.5, C
                MOV         @R0, A

                INC         R0
                MOV         A, @R0
                MOV         C, B.1              ;B
                MOV         ACC.4, C
                MOV         C, B.6              ;G
                MOV         ACC.5, C
                MOV         @R0, A

                INC         R0
                MOV         A, @R0
                MOV         C, B.0                  ;A
                MOV         ACC.4, C
                MOV         C, B.5                  ;F
                MOV         ACC.5, C
                MOV         @R0, A
                RET

L_NotLoadChar2:
                CJNE        A, #3, L_NotLoadChar3
                MOV         R0,#LCD_buff
                MOV         A, @R0
                MOV         C, B.3                  ;D
                MOV         ACC.2, C
                MOV         @R0, A

                INC         R0
                MOV         A, @R0
                MOV         C, B.2                  ;C
                MOV         ACC.2, C
                MOV         C, B.4                  ;E
                MOV         ACC.3, C
                MOV         @R0, A

                INC         R0
                MOV         A, @R0
                MOV         C, B.1                  ;B
                MOV         ACC.2, C
                MOV         C, B.6                  ;G
                MOV         ACC.3, C
                MOV         @R0, A

                INC         R0
                MOV         A, @R0
                MOV         C, B.0                  ;A
                MOV         ACC.2, C
                MOV         C, B.5                  ;F
                MOV         ACC.3, C
                MOV         @R0, A
                RET

L_NotLoadChar3:
                CJNE        A, #4, L_NotLoadChar4
                MOV         R0,#LCD_buff
                MOV         A, @R0
                MOV         C, B.3                  ;D
                MOV         ACC.0, C
                MOV         @R0, A

                INC         R0
                MOV         A, @R0
                MOV         C, B.2                  ;C
                MOV         ACC.0, C
                MOV         C, B.4                  ;E
                MOV         ACC.1, C
                MOV         @R0, A

                INC         R0
                MOV         A, @R0
                MOV         C, B.1                  ;B
                MOV         ACC.0, C
                MOV         C, B.6                  ;G
                MOV         ACC.1, C
                MOV         @R0, A

                INC         R0
                MOV         A, @R0
                MOV         C, B.0                  ;A
                MOV         ACC.0, C
                MOV         C, B.5                  ;F
                MOV         ACC.1, C
                MOV         @R0, A
                RET

L_NotLoadChar4:
                CJNE        A, #5, L_NotLoadChar5
                MOV         R0,#LCD_buff+4
                MOV         A, @R0
                MOV         C, B.3                  ;D
                MOV         ACC.6, C
                MOV         @R0, A

                INC         R0
                MOV         A, @R0
                MOV         C, B.2                  ;C
                MOV         ACC.6, C
                MOV         C, B.4                  ;E
                MOV         ACC.7, C
                MOV         @R0, A

                INC         R0
                MOV         A, @R0
                MOV         C, B.1                  ;B
                MOV         ACC.6, C
                MOV         C, B.6                  ;G
                MOV         ACC.7, C
                MOV         @R0, A

                INC         R0
                MOV         A, @R0
                MOV         C, B.0                  ;A
                MOV         ACC.6, C
                MOV         C, B.5                  ;F
                MOV         ACC.7, C
                MOV         @R0, A
                RET

L_NotLoadChar5:
                CJNE        A, #6, L_NotLoadChar6
                MOV         R0,#LCD_buff+4
                MOV         A, @R0
                MOV         C, B.3                  ;D
                MOV         ACC.4, C
                MOV         @R0, A

                INC         R0
                MOV         A, @R0
                MOV         C, B.2                  ;C
                MOV         ACC.4, C
                MOV         C, B.4                  ;E
                MOV         ACC.5, C
                MOV         @R0, A

                INC         R0
                MOV         A, @R0
                MOV         C, B.1                  ;B
                MOV         ACC.4, C
                MOV         C, B.6                  ;G
                MOV         ACC.5, C
                MOV         @R0, A

                INC         R0
                MOV         A, @R0
                MOV         C, B.0                  ;A
                MOV         ACC.4, C
                MOV         C, B.5                  ;F
                MOV         ACC.5, C
                MOV         @R0, A
                RET
L_NotLoadChar6:
                RET

                END
```

10 Instruction Set

Mnemonic		Description	Bytes	Cycle
ADD	A,Rn	Add register to Accumulator	1	1
ADD	A,direct	Add ditect byte to Accumulator	2	1
ADD	A,@Ri	Add indirect RAM to Accumulator	1	1
ADD	A,#data	Add immediate data to Accumulator	2	1
ADDC	A,Rn	Add register to Accumulator with Carry	1	1
ADDC	A,direct	Add ditect byte to Accumulator with Carry	2	1
ADDC	A,@Ri	Add indirect RAM to Accumulator with Carry	1	1
ADDC	A,#data	Add immediate data to Accumulator with Carry	2	1
SUBB	A,Rn	Subtract Register from Accumulator with borrow	1	1
SUBB	A,direct	Subtract direct byte from Accumulator with borrow	2	1
SUBB	A,@Ri	Subtract indirect RAM from Accumulator with borrow	1	1
SUBB	A,#data	Substract immediate data from Accumulator with borrow	2	1
INC	A	Increment Accumulator	1	1
INC	Rn	Increment register	1	1
INC	direct	Increment direct byte	2	1
INC	@Ri	Increment indirect RAM	1	1
DEC	A	Decrement Accumulator	1	1
DEC	Rn	Decrement Register	1	1
DEC	direct	Decrement direct byte	2	1
DEC	@Ri	Decrement indirect RAM	1	1
INC	DPTR	Increment Data Pointer	1	1
MUL	AB	Multiply A & B, high byte of result is in B, low byte in A	1	2
DIV	AB	Divde A by B, quotient is in A, remainder is in B.	1	6
DA	A	Decimal Adjust Accumulator	1	3
ANL	A,Rn	AND Register to Accumulator	1	1
ANL	A,direct	AND direct btye to Accumulator	2	1
ANL	A,@Ri	AND indirect RAM to Accumulator	1	1
ANL	A,#data	AND immediate data to Accumulator	2	1
ANL	direct,A	AND Accumulator to direct byte	2	1
ANL	direct,#data	AND immediate data to direct byte	3	1
ORL	A,Rn	OR register to Accumulator	1	1
ORL	A,direct	OR direct byte to Accumulator	2	1
ORL	A,@Ri	OR indirect RAM to Accumulator	1	1
ORL	A,#data	OR immediate data to Accumulator	2	1
ORL	direct,A	OR Accumulator to direct byte	2	1
ORL	direct,#data	OR immediate data to direct byte	3	1
XRL	A,Rn	Exclusive-OR register to Accumulator	1	1
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	1
XRL	A,@Ri	Exclusive-OR indirect RAM to Accumulator	1	1
XRL	A,#data	Exclusive-OR immediate data to Accumulator	2	1

XRL direct,A	Exclusive-OR Accumulator to direct byte	2	1
XRL direct,#data	Exclusive-OR immediate data to direct byte	3	1
CLR A	Clear Accumulator	1	1
CPL A	Complement Accumulator	1	1
RL A	Rotate Accumulator Left	1	1
RLC A	Rotate Accumulator Left through the Carry	1	1
RR A	Rotate Accumulator Right	1	1
RRC A	Rotate Accumulator Right through the Carry	1	1
SWAP A	Swap nibbles within the Accumulator	1	1
CLR C	Clear Carry	1	1
CLR bit	Clear direct bit	2	1
SETB C	Set Carry	1	1
SETB bit	Set direct bit	2	1
CPL C	Complement Carry	1	1
CPL bit	Complement direct bit	2	1
ANL C,bit	AND direct bit to Carry	2	1
ANL C,/bit	AND complement of direct bit to Carry	2	1
ORL C,bit	OR direct bit to Carry	2	1
ORL C,/bit	OR complement of direct bit to Carry	2	1
MOV C,bit	Move direct bit to Carry	2	1
MOV bit,C	Move Carry to direct bit	2	1
MOV A,Rn	Move register to Accumulator	1	1
MOV A,direct	Move direct byte to Accumulator	2	1
MOV A,@Ri	Move indirect RAM to Accumulator	1	1
MOV A,#data	Move immediate data to Accumulator	2	1
MOV Rn,A	Move Accumulator to register	1	1
MOV Rn,direct	Move direct byte to register	2	1
MOV Rn,#data	Move immediate data to register	2	1
MOV direct,A	Move Accumulator to direct byte	2	1
MOV direct,Rn	Move register to direct byte	2	1
MOV direct,direct	Move direct byte to direct	3	1
MOV direct,@Ri	Move indirect RAM to direct byte	2	1
MOV direct,#data	Move immediate data to direct byte	3	1
MOV @Ri,A	Move Accumulator to indirect RAM	1	1
MOV @Ri,direct	Move direct byte to indirect RAM	2	1
MOV @Ri,#data	Move immediate data to indirect RAM	2	1
MOV DPTR,#data16	Move 16-bit immdiate data to indirect RAM	3	1
MOVC A,@A+DPTR	Move Code byte relative to DPTR to Accumulator	1	4
MOVC A,@A+PC	Move Code byte relative to PC to Accumulator	1	3
MOVX A,@Ri	Move extended RAM(8-bit addr) to Accumulator (Read)	1	3[1]
MOVX A,@DPTR	Move extended RAM(16-bit addr) to Accumulator (Read)	1	2[1]
MOVX @Ri,A	Move Accumulator to extended RAM(8-bit addr) (Write)	1	3[1]
MOVX @DPTR,A	Move Accumulator to extended RAM(16-bit addr) (Write)	1	2[1]
PUSH direct	Push direct byte onto stack	2	1
POP direct	POP direct byte from stack	2	1
XCH A,Rn	Exchange register with Accumulator	1	1
XCH A,direct	Exchange direct byte with Accumulator	2	1
XCH A,@Ri	Exchange indirect RAM with Accumulator	1	1

XCHD A,@Ri	Exchange low-order Digit indirect RAM with Accumulator	1	1
ACALL addr11	Absolute Subroutine Call	2	3
LCALL addr16	Long Subroutine Call	3	3
RET	Return from Subroutine	1	3
RETI	Return from interrupt	1	3
AJMP addr11	Absolute Jump	2	3
LJMP addr16	Long Jump	3	3
SJMP rel	Short Jump (relative addr)	2	3
JMP @A+DPTR	Jump indirect relative to the DPTR	1	4
JZ rel	Jump if Accumulator is Zero	2	1/3[2]
JNZ rel	Jump if Accumulator is not Zero	2	1/3[2]
JC rel	Jump if Carry is set	2	1/3[2]
JNC rel	Jump if Carry not set	2	1/3[2]
JB bit,rel	Jump if direct bit is set	3	1/3[2]
JNB bit,rel	Jump if direct bit is not set	3	1/3[2]
JBC bit,rel	Jump if direct bit is set & clear bit	3	1/3[2]
CJNE A,direct,rel	Compare direct byte to Accumulator and jump if not equal	3	2/3[3]
CJNE A,#data,rel	Compare immediate data to Accumulator and Jump if not equal	3	1/3[2]
CJNE Rn,#data,rel	Compare immediate data to register and Jump if not equal	3	2/3[3]
CJNE @Ri,#data,rel	Compare immediate data to indirect and jump if not equal	3	2/3[3]
DJNZ Rn,rel	Decrement register and jump if not Zero	2	2/3[3]
DJNZ direct,rel	Decrement direct byte and Jump if not Zero	3	2/3[3]
NOP	No Operation	1	1

[1]: When accessing external extended RAM, the instruction execution cycle is related to the SPEED [1: 0] bits in the BUS_SPEED register.

[2]: For the conditional jump statement, the execution cycle will be different based on whether the conditions are met or not. When the conditions are not met, the jump will not occur and continue to execute the next instruction, then execution cycle of the conditional jump statement is 1 machine cycle. When the conditions are met, the jump will occur, the execution cycle of the conditional jump statement is 3 machine cycles.

[3]: For the conditional jump statement, the execution cycle will be different based on whether the conditions are met or not. When the conditions are not met, the jump will not occur and continue to execute the next instruction, then execution cycle of the conditional jump statement is 1 machine cycle. When the conditions are met, the jump will occur, the execution cycle of the conditional jump statement is 3 machine cycles.

11 Interrupt System

(An error will be reported when compiled in Keil when using an interrupt with an interrupt number greater than 31 in a C program. Please refer to Appendix for the solution.)

The interrupt system is set up to give the CPU real-time processing capabilities for external emergencies.

If an emergency request occurrs while CPU is dealing with something, the CPU is required to suspend the current work to handle the emergency. After the emergency processing is completed, the CPU returns to the place where it was interrupted and continues the original work. This process is called interrupt. The componet that implements this function is called the interrupt system. The request source that makes the CPU interrupt to suspend the current work is called the interrupt source. Microcontroller interrupt system generally allows multiple interrupt sources. When several interrupt sources simultaneously require the CPU to handle the requests, the CPU should response the interrupt source which has the highest priority. Usually the CPU handle the interrupt requests according to the priority of interrupt sources. The most urgent incidents have the highest priority. Each interrupt source has a priority level. The CPU always responds the highest priority interrupt request.

Another interrupt source request with a higher priority takes place while the CPU is processing an interrupt source request, that is, the CPU is executing the corresponding interrupt service routine, if the CPU can suspend the original interrupt service routine, and deal with the higher priority interrupt request source, and then return to the original low-level interrupt service routine after processing finished, this process is called interrupt nesting. Such an interrupt system is called a multi-level interrupt system, whereas an interrupt system without interrupt nesting is called a single-level interrupt system.

The corresponding interrupt request can be masked by turning off the general enable bit (EA / IE.7) or the corresponding interrupt enable bit. The CPU can be enabled to respond to the corresponding interrupt request by turning on the corresponding interrupt enable bit. Every interrupt source can be set or reset independently by software to interrupt enabled or disabled state. The priority of some interrupts can be set by software. Higher priority interrupt requests can interrupt lower priority interrupts, whereas lower priority interrupt requests can not interrupt higher priority interrupts. When two interrupts with the same priority ocuur simultaneously, the inquiry order determines which interrupt the system responds firstly.

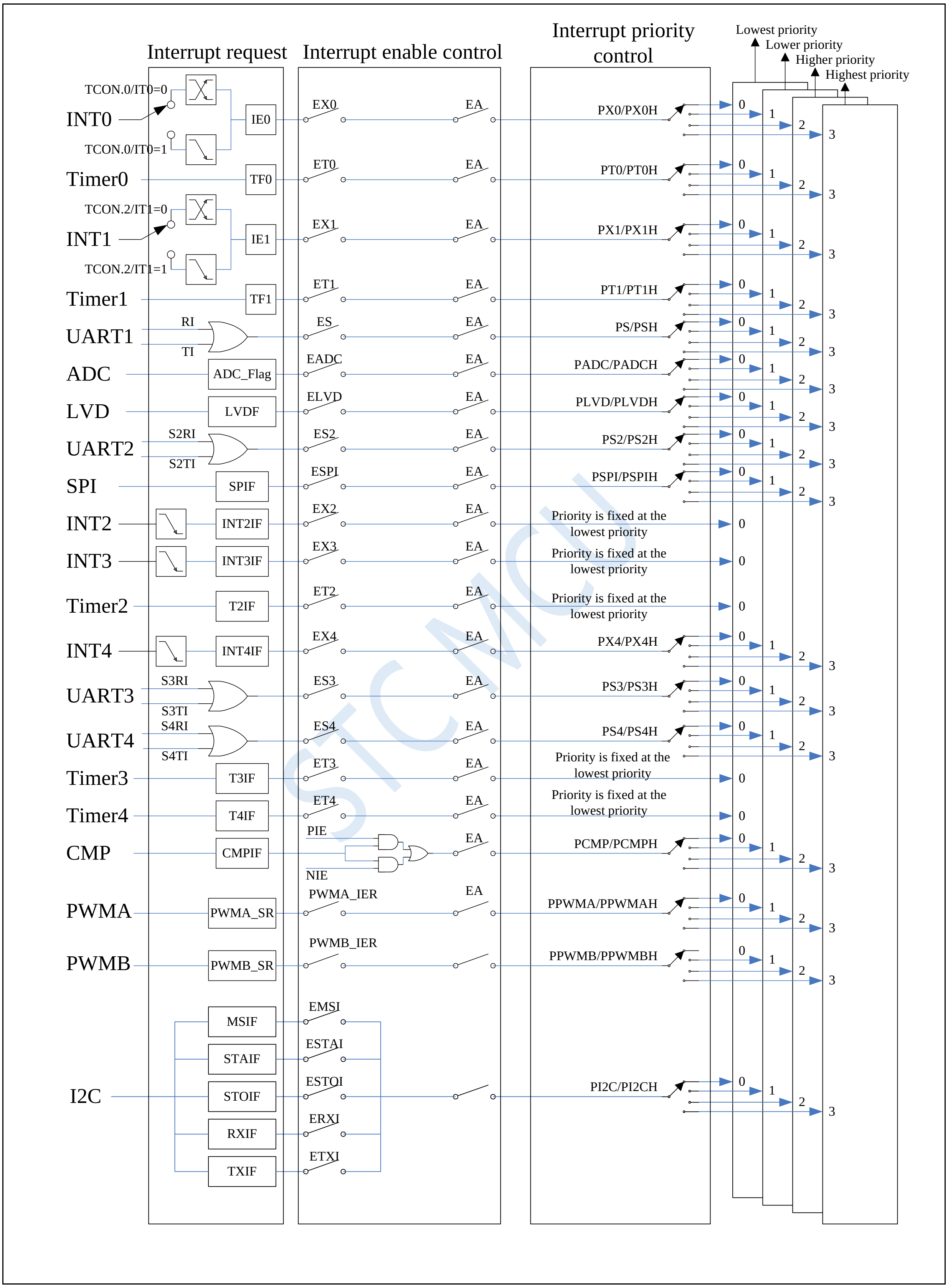
11.1 Interrupt sources of STC8H series

The √ in the following table indicates that the corresponding series have the corresponding interrupt source.

Interrupt sources	STC8H1K16 family	STC8H1K08 family	STC8H3K64S4 family	STC8H3K64S2 family	STC8H8K64U -A family	STC8H8K64U -B family	STC8H4K64TLR family	STC8H4K64TLCD family
External interrupt 0 (INT0) Supports falling and edges interrupts	√	√	√	√	√	√	√	√
Timer 0 interrrupt (Timer0)	√	√	√	√	√	√	√	√
External interrupt 1 (INT1) Supports falling and edges interrupts	√	√	√	√	√	√	√	√
Timer 1 interrrupt (Timer1)	√	√	√	√	√	√	√	√
UART1 interrupt (UART1)	√	√	√	√	√	√	√	√
ADC interrupt (ADC)	√	√	√	√	√	√	√	√
Low voltage detection interrupt (LVD)	√	√	√	√	√	√	√	√
CCP/PCA/PWM interrupt (CCP/PCA)	√	√	√	√	√	√	√	√
UART2 interrupt (UART2)	√	√	√	√	√	√	√	√
SPI interrupt (SPI)	√	√	√	√	√	√	√	√
External interrupt 2 (INT2) Supports falling edge interrupts	√	√	√	√	√	√	√	√
External interrupt 3 (INT3) Supports falling edge interrupts	√	√	√	√	√	√	√	√
Timer 2 interrupt (Timer2)	√	√	√	√	√	√	√	√

External interrupt 4 (INT4)	√	√	√	√	√	√	√	√
UART3 interrupt (UART3)			√		√	√	√	√
UART4 interrupt (UART4)			√		√	√	√	√
Timer 3 interrrupt (Timer3)	√		√	√	√	√	√	√
Timer 4 interrrupt (Timer4)	√		√	√	√	√	√	√
Comparator interrupt (CMP)	√	√	√	√	√	√	√	√
I2C interrupt	√	√	√	√	√	√	√	√
PWMA	√	√	√	√	√	√	√	√
PWMB	√	√	√	√	√	√	√	√
USB interrupt					√	√		
Touch Key interrupt							√	√
RTC interrupt Supports falling edge, rising edge, high level and low level interrupts						√	√	√
Port0 interrupt Supports falling edge, rising edge, high level and low level interrupts			√	√		√	√	√
Port1 interrupt Supports falling edge, rising edge, high level and low level interrupts			√	√		√	√	√
Port2 interrupt Supports falling edge, rising edge, high level and low level interrupts			√	√		√	√	√
Port3 interrupt Supports falling edge, rising edge, high level and low level interrupts			√	√		√	√	√
Port4 interrupt Supports falling edge, rising edge, high level and low level interrupts			√	√		√	√	√
Port5 interrupt Supports falling edge, rising edge, high level and low level interrupts			√	√		√	√	√
Port6 interrupt Supports falling edge, rising edge, high level and low level interrupts			√	√		√		
Port7 interrupt Supports falling edge, rising edge, high level and low level interrupts			√	√		√		
DMA_M2M interrupt						√	√	√
DMA_ADC interrupt						√	√	√
DMA_SPI interrupt						√	√	√
DMA_UR1T interrupt						√	√	√
DMA_UR1R interrupt						√	√	√
DMA_UR2T interrupt						√	√	√
DMA_UR2R interrupt						√	√	√
DMA_UR3T interrupt						√	√	√
DMA_UR3R interrupt						√	√	√
DMA_UR4T interrupt						√	√	√
DMA_UR4R interrupt						√	√	√
DMA_LCM interrupt						√	√	√
LCM interrupt						√	√	√

11.2 Structure of STC8H Interrupt



11.3 Interrupt List of STC8H Series

Interrupt source	Interrupt vector	Order	Priority level setup bit	Priority level	Interrupt request flag	Interrupt enable bit
INT0	0003H	0	PX0PX0H	0/1/2/3	IE0	EX0
Timer0	000BH	1	PT0,PT0H	0/1/2/3	TF0	ET0
INT1	0013H	2	PX1,PX1H	0/1/2/3	IE1	EX1
Timer1	001BH	3	PT1,PT1H	0/1/2/3	TF1	ET1
UART1	0023H	4	PS,PSH	0/1/2/3	RI || TI	ES
ADC	002BH	5	PADC,PADCH	0/1/2/3	ADC_FLAG	EADC
LVD	0033H	6	PLVD,PLVDH	0/1/2/3	LVDF	ELVD
PCA	003BH	7	PPCA,PPCAH	0/1/2/3	CF	ECF
					CCF0	ECCF0
					CCF1	ECCF1
					CCF2	ECCF2
					CCF3	ECCF3
UART2	0043H	8	PS2,PS2H	0/1/2/3	S2RI || S2TI	ES2
SPI	004BH	9	PSPI,PSPIH	0/1/2/3	SPIF	ESPI
INT2	0053H	10		0	INT2IF	EX2
INT3	005BH	11		0	INT3IF	EX3
Timer2	0063H	12		0	T2IF	ET2
INT4	0083H	16	PX4,PX4H	0/1/2/3	INT4IF	EX4
UART3	008BH	17	PS3,PS3H	0/1/2/3	S3RI || S3TI	ES3
UART4	0093H	18	PS4,PS4H	0/1/2/3	S4RI || S4TI	ES4
Timer3	009BH	19		0	T3IF	ET3
Timer4	00A3H	20		0	T4IF	ET4
CMP	00ABH	21	PCMP,PCMPH	0/1/2/3	CMPIF	PIE|NIE
I2C	00C3H	24	PI2C,PI2CH	0/1/2/3	MSIF	EMSI
					STAIF	ESTAI
					RXIF	ERXI
					TXIF	ETXI
					STOIF	ESTOI
USB	00CBH	25	PUSB,PUSBH	0/1/2/3	USB Events	EUSB
PWMA	00D3H	26	PPWMA,PPWMAH	0/1/2/3	PWMA_SR	PWMA_IER
PWMB	00DBH	27	PPWMB,PPWMBH	0/1/2/3	PWMB_SR	PWMB_IER
TKSU	011BH	35	PTKSU,PTKSUH	0/1/2/3	TKIF	ETKSUI
RTC	0123H	36	PRTC,PRTCH	0/1/2/3	ALAIF	EALAI
					DAYIF	EDAYI
					HOURIF	EHOURI
					MINIF	EMINI
					SECIF	ESECI
					SEC2IF	ESEC2I
					SEC8IF	ESEC8I
					SEC32IF	ESEC32I

Interrupt source	Interrupt vector	Order	Priority level setup bit	Priority level	Interrupt request flag	Interrupt enable bit
P0 interrupt	012BH	37	PINIPL[0], PINIPH[0]	0/1/2/3	P0INTF	P0INTE
P1 interrupt	0133H	38	PINIPL[1], PINIPH[1]	0/1/2/3	P1INTF	P1INTE
P2 interrupt	013BH	39	PINIPL[2], PINIPH[2]	0/1/2/3	P2INTF	P2INTE
P3 interrupt	0143H	40	PINIPL[3], PINIPH[3]	0/1/2/3	P3INTF	P3INTE
P4 interrupt	014BH	41	PINIPL[4], PINIPH[4]	0/1/2/3	P4INTF	P4INTE
P5 interrupt	0153H	42	PINIPL[5], PINIPH[5]	0/1/2/3	P5INTF	P5INTE
P6 interrupt	015BH	43	PINIPL[6], PINIPH[6]	0/1/2/3	P6INTF	P6INTE
P7 interrupt	0163H	44	PINIPL[7], PINIPH[7]	0/1/2/3	P7INTF	P7INTE
DMA_M2M interrupt	017BH	47	M2MIP[1:0]	0/1/2/3	M2MIF	M2MIE
DMA_ADC interrupt	0183H	48	ADCIP[1:0]	0/1/2/3	ADCIF	ADCIE
DMA_SPI interrupt	018BH	49	SPIIP[1:0]	0/1/2/3	SPIIF	SPIIE
DMA_UR1T interrupt	0193H	50	UR1TIP[1:0]	0/1/2/3	UR1TIF	UR1TIE
DMA_UR1R interrupt	019BH	51	UR1RIP[1:0]	0/1/2/3	UR1RIF	UR1RIE
DMA_UR2T interrupt	01A3H	52	UR2TIP[1:0]	0/1/2/3	UR2TIF	UR2TIE
DMA_UR2R interrupt	01ABH	53	UR2RIP[1:0]	0/1/2/3	UR2RIF	UR2RIE
DMA_UR3T interrupt	01B3H	54	UR3TIP[1:0]	0/1/2/3	UR3TIF	UR3TIE
DMA_UR3R interrupt	01BBH	55	UR3RIP[1:0]	0/1/2/3	UR3RIF	UR3RIE
DMA_UR4T interrupt	01C3H	56	UR4TIP[1:0]	0/1/2/3	UR4TIF	UR4TIE
DMA_UR4R interrupt	01CBH	57	UR4RIP[1:0]	0/1/2/3	UR4RIF	UR3RIE
DMA_LCM interrupt	01D3H	58	LCMIP[1:0]	0/1/2/3	LCMIF	LCMIE
LCM interrupt	01DBH	59	LCMIFIP[1:0]	0/1/2/3	LCMIFIF	LCMIFIE

Interrupt service routine may be declared in C language as the following,

```
void    INT0_Routine(void)        interrupt  0;
void    TM0_Rountine(void)        interrupt  1;
void    INT1_Routine(void)        interrupt  2;
void    TM1_Rountine(void)        interrupt  3;
void    UART1_Routine(void)       interrupt  4;
void    ADC_Routine(void)         interrupt  5;
void    LVD_Routine(void)         interrupt  6;
void    PCA_Routine(void)         interrupt  7;
void    UART2_Routine(void)       interrupt  8;
void    SPI_Routine(void)         interrupt  9;
void    INT2_Routine(void)        interrupt  10;
void    INT3_Routine(void)        interrupt  11;
void    TM2_Routine(void)         interrupt  12;
void    INT4_Routine(void)        interrupt  16;
void    UART3_Routine(void)       interrupt  17;
void    UART4_Routine(void)       interrupt  18;
void    TM3_Routine(void)         interrupt  19;
void    TM4_Routine(void)         interrupt  20;
void    CMP_Routine(void)         interrupt  21;
void    I2C_Routine(void)         interrupt  24;
void    USB_Routine(void)         interrupt  25;
void    PWMA_Routine(void)        interrupt  26;
void    PWMB_Routine(void)        interrupt  27;
//void TKSU_Routine(void) interrupt 35;
//void RTC_Routine(void) interrupt 36;
//void P0Int_Routine(void) interrupt 37;
//void P0Int_Routine(void) interrupt 37;
//void P0Int_Routine(void) interrupt 37;
//void P1Int_Routine(void) interrupt 38;
//void P2Int_Routine(void) interrupt 39;
//void P3Int_Routine(void) interrupt 40;
//void P4Int_Routine(void) interrupt 41;
//void P5Int_Routine(void) interrupt 42;
//void P6Int_Routine(void) interrupt 43;
 //void P7Int_Routine(void) interrupt 44;
```

Interrupt service routines with interrupt numbers greater than 31 cannot be directly declared in C language. Please refer to the processing method in "Appendix J". Assembly language is not affected

11.4 Registers Related to Interrupt

Symbol	Description	Address	Bit Address and Symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
IE	Interrupt enable register	A8H	EA	ELVD	EADC	ES	ET1	EX1	ET0	EX0	0000,0000
IE2	Interrupt enable register2	AFH	EUSB	ET4	ET3	ES4	ES3	ET2	ESPI	ES2	0000,0000
INTCLKO	External interrupt and clock output control register	8FH	-	EX4	EX3	EX2	-	T2CLKO	T1CLKO	T0CLKO	x000,x000
IP	Interrupt Priority Low Byte	B8H	-	PLVD	PADC	PS	PT1	PX1	PT0	PX0	x000,0000
IPH	Interrupt Priority High Byte	B7H	-	PLVDH	PADCH	PSH	PT1H	PX1H	PT0H	PX0H	x000,0000
IP2	2nd Interrupt Priority register low byte	B5H	PUSB PTKSU	PI2C	PCMP	PX4	PPWMB	PPWMA	PSPI	PS2	0000,0000
IP2H	2nd Interrupt Priority register high byte	B6H	PUSBH PTKSUH	PI2CH	PCMPH	PX4H	PPWMBH	PPWMAH	PSPIH	PS2H	0000,0000
IP3	3nd Interrupt Priority register low byte	DFH	-	-	-	-	-	PRTC	PS4	PS3	xxxx,x000
IP3H	3nd Interrupt Priority Register High Byte	EEH	-	-	-	-	-	PRTCH	PS4H	PS3H	xxxx,x000
TCON	Timer 0 and 1 control register	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0000,0000
AUXINTIF	Extended External Interrupt Flag Register	EFH	-	INT4IF	INT3IF	INT2IF	-	T4IF	T3IF	T2IF	x000,x000
SCON	UART1 control register	98H	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	0000,0000
S2CON	UART2 control register	9AH	S2SM0	-	S2SM2	S2REN	S2TB8	S2RB8	S2TI	S2RI	0100,0000
S3CON	UART3 control register	ACH	S3SM0	S3ST3	S3SM2	S3REN	S3TB8	S3RB8	S3TI	S3RI	0000,0000
S4CON	Serial port 4 control register	84H	S4SM0	S4ST4	S4SM2	S4REN	S4TB8	S4RB8	S4TI	S4RI	0000,0000
PCON	Power control register	87H	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000
ADC_CONTR	ADC control register	BCH	ADC_POWER	ADC_START	ADC_FLAG	ADC_EPWMT	ADC_CHS[3:0]				000x,0000
SPSTAT	SPI Status register	CDH	SPIF	WCOL	-	-	-	-	-	-	00xx,xxxx
CMPCR1	Comparator Control Register 1	E6H	CMPEN	CMPIF	PIE	NIE	PIS	NIS	CMPOE	CMPRES	0000,0000
LCMIFCFG	LCM Interface Configuration Register	FE50H	LCMIFIE	-	LCMIFIP[1:0]		LCMIFDPS[1:0]		D16_D8	M68_I80	0x00,0000
LCMIFSTA	LCM Interface Status Register	FE53H	-	-	-	-	-	-	-	LCMIFIF	xxxx,xxx0
I2CMSCR	I^{2}C Master Control Register	FE81H	EMSI	-	-	-	MSCMD[3:0]				0xxx,0000
I2CMSST	I^{2}C Master Status Register	FE82H	MSBUSY	MSIF	-	-	-	-	MSACKI	MSACKO	00xx,xx00
I2CSLCR	I^{2}C Slave Control Register	FE83H	-	ESTAI	ERXI	ETXI	ESTOI	-	-	SLRST	x000,0xx0
I2CSLST	I^{2}C Slave Status Register	FE84H	SLBUSY	STAIF	RXIF	TXIF	STOIF	TXING	SLACKI	SLACKO	0000,0000
PWMA_IER	PWMA Interrupt enable register	FEC4H	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE	0000,0000
PWMA_SR1	PWMA Status register 1	FEC5H	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF	0000,0000
PWMA_SR2	PWMA Status register 2	FEC6H	-	-	-	CC4OF	CC3OF	CC2OF	CC1OF	-	xxx0,000x
PWMB_IER	PWMB Interrupt enable register	FEE4H	BIE	TIE	COMIE	CC8IE	CC7IE	CC6IE	CC5IE	UIE	0000,0000
PWMB_SR1	PWMB Status register 1	FEE5H	BIF	TIF	COMIF	CC8IF	CC7IF	CC6IF	CC5IF	UIF	0000,0000
PWMB_SR2	PWMB Status register 2	FEE6H	-	-	-	CC8OF	CC7OF	CC6OF	CC5OF	-	xxx0,000x
P0INTE	P0 Interrupt enable register	FD00H	P07INTE	P06INTE	P05INTE	P04INTE	P03INTE	P02INTE	P01INTE	P00INTE	0000,0000
P1INTE	P1 Interrupt enable register	FD01H	P17INTE	P16INTE	P15INTE	P14INTE	P13INTE	P12INTE	P11INTE	P10INTE	0000,0000
P2INTE	P2 Interrupt enable register	FD02H	P27INTE	P26INTE	P25INTE	P24INTE	P23INTE	P22INTE	P21INTE	P20INTE	0000,0000
P3INTE	P3 Interrupt enable register	FD03H	P37INTE	P36INTE	P35INTE	P34INTE	P33INTE	P32INTE	P31INTE	P30INTE	0000,0000
P4INTE	P4 Interrupt enable register	FD04H	P47INTE	P46INTE	P45INTE	P44INTE	P43INTE	P42INTE	P41INTE	P40INTE	0000,0000
P5INTE	P5 Interrupt enable register	FD05H	-	-	P55INTE	P54INTE	P53INTE	P52INTE	P51INTE	P50INTE	xx00,0000
P6INTE	P6 Interrupt enable register	FD06H	P67INTE	P66INTE	P65INTE	P64INTE	P63INTE	P62INTE	P61INTE	P60INTE	0000,0000
P7INTE	P7 Interrupt enable register	FD07H	P77INTE	P76INTE	P75INTE	P74INTE	P73INTE	P72INTE	P71INTE	P70INTE	0000,0000
P0INTF	P0 Interrupt flag register	FD10H	P07INTF	P06INTF	P05INTF	P04INTF	P03INTF	P02INTF	P01INTF	P00INTF	0000,0000
P1INTF	P1 Interrupt flag register	FD11H	P17INTF	P16INTF	P15INTF	P14INTF	P13INTF	P12INTF	P11INTF	P10INTF	0000,0000
P2INTF	P2 Interrupt flag register	FD12H	P27INTF	P26INTF	P25INTF	P24INTF	P23INTF	P22INTF	P21INTF	P20INTF	0000,0000
P3INTF	P3 Interrupt flag register	FD13H	P37INTF	P36INTF	P35INTF	P34INTF	P33INTF	P32INTF	P31INTF	P30INTF	0000,0000
P4INTF	P4 Interrupt flag register	FD14H	P47INTF	P46INTF	P45INTF	P44INTF	P43INTF	P42INTF	P41INTF	P40INTF	0000,0000
P5INTF	P5 Interrupt flag register	FD15H	-	-	P55INTF	P54INTF	P53INTF	P52INTF	P51INTF	P50INTF	xx00,0000
P6INTF	P6 Interrupt flag register	FD16H	P67INTF	P66INTF	P65INTF	P64INTF	P63INTF	P62INTF	P61INTF	P60INTF	0000,0000
P7INTF	P7 Interrupt flag register	FD17H	P77INTF	P76INTF	P75INTF	P74INTF	P73INTF	P72INTF	P71INTF	P70INTF	0000,0000
PINIPL	I/O port interrupt priority low register	FD60H	P7IP	P6IP	P5IP	P4IP	P3IP	P2IP	P1IP	P0IP	0000,0000
PINIPH	I/O port interrupt priority high register	FD61H	P7IPH	P6IPH	P5IPH	P4IPH	P3IPH	P2IPH	P1IPH	P0IPH	0000,0000
DMA_M2M_CFG	M2M_DMA configuration register	FA00H	M2MIE	-	TXACO	RXACO	M2MIP[1:0]		M2MPTY[1:0]		0x00,0000
DMA_ADC_CFG	ADC_DMA configuration register	FA10H	ADCIE	-	-	-	ADCMIP[1:0]		ADCPTY[1:0]		0xxx,0000
DMA_SPI_CFG	SPI_DMA configuration register	FA20H	SPIIE	ACT_TX	ACT_RX	-	SPIIP[1:0]		SPIPTY[1:0]		000x,0000
DMA_UR1T_CFG	UR1T_DMA configuration register	FA30H	UR1TIE	-	-	-	UR1TIP[1:0]		UR1TPTY[1:0]		0xxx,0000
DMA_UR1R_CFG	UR1R_DMA configuration register	FA38H	UR1RIE	-	-	-	UR1RIP[1:0]		UR1RPTY[1:0]		0xxx,0000
DMA_UR2T_CFG	UR2T_DMA configuration register	FA40H	UR2TIE	-	-	-	UR2TIP[1:0]		UR2TPTY[1:0]		0xxx,0000
DMA_UR2R_CFG	UR2R_DMA configuration register	FA48H	UR2RIE	-	-	-	UR2RIP[1:0]		UR2RPTY[1:0]		0xxx,0000
DMA_UR3T_CFG	UR3T_DMA configuration register	FA50H	UR3TIE	-	-	-	UR3TIP[1:0]		UR3TPTY[1:0]		0xxx,0000
DMA_UR3R_CFG	UR3R_DMA configuration register	FA58H	UR3RIE	-	-	-	UR3RIP[1:0]		UR3RPTY[1:0]		0xxx,0000
DMA_UR4T_CFG	UR4T_DMA configuration register	FA60H	UR4TIE	-	-	-	UR4TIP[1:0]		UR4TPTY[1:0]		0xxx,0000
DMA_UR4R_CFG	UR4R_DMA configuration register	FA68H	UR4RIE	-	-	-	UR4RIP[1:0]		UR4RPTY[1:0]		0xxx,0000
DMA_LCM_CFG	LCM_DMA configuration register	FA70H	LCMIE	-	-	-	LCMIP[1:0]		LCMPTY[1:0]		0xxx,0000
DMA_M2M_STA	M2M_DMA status register	FA02H	-	-	-	-	-	-	-	M2MIF	xxxx,xxx0
DMA_ADC_STA	ADC_DMA status register	FA12H	-	-	-	-	-	-	-	ADCIF	xxxx,xxx0
DMA_SPI_STA	SPI_DMA status register	FA22H	-	-	-	-	-	TXOVW	RXLOSS	SPIIF	xxxx,x000
DMA_UR1T_STA	UR1T_DMA status register	FA32H	-	-	-	-	-	TXOVW	-	UR1TIF	xxxx,x0x0
DMA_UR1R_STA	UR1R_DMA status register	FA3AH	-	-	-	-	-	-	RXLOSS	UR1RIF	xxxx,xx00
DMA_UR2T_STA	UR2T_DMA status register	FA42H	-	-	-	-	-	TXOVW	-	UR2TIF	xxxx,x0x0
DMA_UR2R_STA	UR2R_DMA status register	FA4AH	-	-	-	-	-	-	RXLOSS	UR2RIF	xxxx,xx00
DMA_UR3T_STA	UR3T_DMA status register	FA52H	-	-	-	-	-	TXOVW	-	UR3TIF	xxxx,x0x0
DMA_UR3R_STA	UR3R_DMA status register	FA5AH	-	-	-	-	-	-	RXLOSS	UR3RIF	xxxx,xx00
DMA_UR4T_STA	UR4T_DMA status register	FA62H	-	-	-	-	-	TXOVW	-	UR4TIF	xxxx,x0x0
DMA_UR4R_STA	UR4R_DMA status register	FA6AH	-	-	-	-	-	-	RXLOSS	UR4RIF	xxxx,xx00

DMA_LCM_STA	LCM_DMA status register	FA72H	-	-	-	-	-	-	TXOVW	LCMIF	xxxx,xx00

11.4.1 Interrupt Enable Registers (Interrupt Enable bits)

IE (Interrupt Enable Rsgister)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
IE	A8H	EA	ELVD	EADC	ES	ET1	EX1	ET0	EX0

EA: The general or global interrupt enable control bit. The function of EA is to allow interrupts to be multi-level controlled. That is, every interrupt source is controlled by EA firstly and then by its own interrupt enable control bit.

0: All interrupts are masked.

1: Enable the CPU interrupt, every interrupt source would be individually enabled or disabled by setting or clearing its enable bit.

ELVD: Low volatge detection interrupt enable bit.

0: disable low voltage detection interrupt.

1: enable Low voltage detection interrupt.

EADC: ADC interrupt enable bit.

0: disable ADC interrupt.

1: enable ADC interrupt.

ES: UART1 interrupt enable bit.

0: disable UART1 interrupt.

1: enable UART1 interrupt.

ET1: Timer 1 interrupt enable bit.

0: disable Timer 1 interrupt.

1: enable Timer 1 interrupt.

EX1: External interrupt 1 enable bit.

0: disable external interrupt 1.

1: enable external interrupt 1.

ET0: Timer 0 interrupt enable bit.

0: disable Timer 0 interrupt.

1: enable Timer 0 interrupt.

EX0: External interrupt 0 enable bit.

0: disable external interrupt 0.

1: enable external interrupt 0.

IE2 (Interrupt Enable Rsgister 2)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
IE2	AFH	EUSB	ET4	ET3	ES4	ES3	ET2	ESPI	ES2

EUSB: USB interrupt enable bit.

0: disable USB interrupt.

1: enable USB interrupt.

ET4: Timer 4 interrupt enable bit.

0: disable Timer 4 interrupt.

1: enable Timer 4 interrupt.

ET3: Timer 3 interrupt enable bit.

0: disable Timer 3 interrupt.

1: enable Timer 3 interrupt.

ES4: UART4 interrupt enable bit.

0: disable UART4 interrupt.

1: enable UART4 interrupt.

ES3: UART3 interrupt enable bit.

0: disable UART3 interrupt.

1: enable UART3 interrupt.

ET2: Timer 2 interrupt enable bit.

0: disable Timer 2 interrupt.

1: enable Timer 2 interrupt.

ESPI: SPI interrupt enalbe bit.

0: disable SPI interrupt.

1: enable SPI interrupt.

ES2: UART2 interrupt enable bit.
0: disable UART2 interrupt.
1: enable UART2 interrupt.

INTCLKO (External interrupt and clock output control register)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
INTCLKO	8FH	-	EX4	EX3	EX2	-	T2CLKO	T1CLKO	T0CLKO

EX4: External Interrupt 4 enabel bit.
0: disable External Interrupt 4.
1: enable External Interrupt 4.
EX3: External Interrupt 3 enabel bit.
0: disable External Interrupt 3.
1: enable External Interrupt 3.
EX2: External Interrupt 2 enabel bit.
0: disable External Interrupt 2.
1: enable External Interrupt 2.

CMPCR1 (Comparator Control Register 1)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
CMPCR1	E6H	CMPEN	CMPIF	PIE	NIE	PIS	NIS	CMPOE	CMPRES

PIE: Comparator rising-edge interrupt enable bit.
0: disable comparator rising-edge interrupt.
1: enable comparator rising-edge interrupt.
NIE: Comparator falling-edge interrupt enable bit.
0: disable comparator falling-edge interrupt.
1: enable comparator falling-edge interrupt.

I2C Control Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
I2CMSCR	FE81H	EMSI	-	-	-	MSCMD[3:0]			
I2CSLCR	FE83H	-	ESTAI	ERXI	ETXI	ESTOI	-	-	SLRST

EMSI: I^{2}C master mode interrupt enable bit.
0: disable I^{2}C master mode interrupt.
1: enable I^{2}C master mode interrupt.
ESTAI: I^{2}C slave receives the START event interrupt enable bit.
0: disable I^{2}C slave receives the START event interrupt.
1: enable I^{2}C slave receives the START event interrupt.
ERXI: I^{2}C slave completes receiving data event interrupt enable bit.
0: disable I^{2}C slave completes receiving data event interrupt.
1: enable I^{2}C slave completes receiving data event interrupt.
ETXI: I^{2}C slave completes transmitting data event interrupt enable bit.
0: disable I^{2}C slave completes transmitting data event interrupt.
1: enable I^{2}C slave completes transmitting data event interrupt.
ESTOI: I^{2}C slave receives STOP event interrupt enable bit.
0: disable I^{2}C slave receives STOP event interrupt.
1: enable I^{2}C slave receives STOP event interrupt.

PWMA interrupt enable register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_IER	FEC4H	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE

BIE: PWMA brake interrupt enable bit.
0: Disable PWMA brake interrupt
1: Enable PWMA brake interrupt
TIE: PWMA trigger interrupt enable bit.
0: Disable PWMA trigger interrupt
1: Allow PWMA to trigger interrupt
COMIE: PWMA compare interrupt enable bit.
0: Disable PWMA compare interrupt

1: Enable PWMA compare interrupt

CC4IE: PWMA capture compare channel 4 interrupt enable bit.
0: Disable PWMA capture compare channel 4 interrupt
1: Allow PWMA to capture and compare channel 4 interrupt

CC3IE: PWMA capture compare channel 3 interrupt enable bit.
0: Disable PWMA capture compare channel 3 interrupt
1: Allow PWMA to capture and compare channel 3 interrupt

CC2IE: PWMA capture compare channel 2 interrupt enable bit.
0: Disable PWMA capture compare channel 2 interrupt
1: Allow PWMA to capture and compare channel 2 interrupt

CC1IE: PWMA capture compare channel 1 interrupt enable bit.
0: Disable PWMA capture compare channel 1 interrupt
1: Allow PWMA to capture and compare channel 1 interrupt

UIE: PWMA update interrupt enable bit.
0: Disable PWMA update interrupt
1: Enable PWMA update interrupt

PWMB interrupt enable register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
PWMB_IER	FEE4H	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE

BIE: PWMB brake interrupt enable bit.
0: Disable PWMB brake interrupt
1: Enable PWMB brake interrupt

TIE: PWMB trigger interrupt enable bit.
0: Disable PWMB trigger interrupt
1: Allow PWMB to trigger interrupt

COMIE: PWMB compare interrupt enable bit.
0: Disable PWMB compare interrupt
1: Enable PWMB compare interrupt

CC4IE: PWMB capture compare channel 4 interrupt enable bit.
0: Disable PWMB capture compare channel 4 interrupt
1: Allow PWMB to capture and compare channel 4 interrupt

CC3IE: PWMB capture compare channel 3 interrupt enable bit.
0: Disable PWMB capture compare channel 3 interrupt
1: Allow PWMB to capture and compare channel 3 interrupt

CC2IE: PWMB capture compare channel 2 interrupt enable bit.
0: Disable PWMB capture compare channel 2 interrupt
1: Allow PWMB to capture and compare channel 2 interrupt

CC1IE: PWMB capture compare channel 1 interrupt enable bit.
0: Disable PWMB capture compare channel 1 interrupt
1: Allow PWMB to capture and compare channel 1 interrupt

UIE: PWMB update interrupt enable bit.
0: Disable PWMB update interrupt
1: Enable PWMB update interrupt

Port interrupt enable registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P0INTE	FD00H	P07INTE	P06INTE	P05INTE	P04INTE	P03INTE	P02INTE	P01INTE	P00INTE
P1INTE	FD01H	P17INTE	P16INTE	P15INTE	P14INTE	P13INTE	P12INTE	P11INTE	P10INTE
P2INTE	FD02H	P27INTE	P26INTE	P25INTE	P24INTE	P23INTE	P22INTE	P21INTE	P20INTE
P3INTE	FD03H	P37INTE	P36INTE	P35INTE	P34INTE	P33INTE	P32INTE	P31INTE	P30INTE
P4INTE	FD04H	P47INTE	P46INTE	P45INTE	P44INTE	P43INTE	P42INTE	P41INTE	P40INTE
P5INTE	FD05H	-	-	P55INTE	P54INTE	P53INTE	P52INTE	P51INTE	P50INTE
P6INTE	FD06H	P67INTE	P66INTE	P65INTE	P64INTE	P63INTE	P62INTE	P61INTE	P60INTE
P7INTE	FD07H	P77INTE	P76INTE	P75INTE	P74INTE	P73INTE	P72INTE	P71INTE	P70INTE

PnINTE.x: Port interrupt enable control bit (n=0~7, x=0~7)
0: disable Pn.x port interrupt function
1: Enable Pn.x port interrupt function

LCM interface configuration register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
LCMIFCFG	FE50H	LCMIFIE	-	LCMIFIP[1:0]		LCMIFDPS[1:0]		D16_D8	M68_I80

LCMIFIE: LCM interface interrupt enable bit.

0: disable LCM interface interrupt
1: Enable LCM interface interrupt

DMA interrupt enable registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_M2M_CFG	FA00H	M2MIE	-	TXACO	RXACO	M2MIP[1:0]		M2MPTY[1:0]	
DMA_ADC_CFG	FA10H	ADCIE	-	-	-	ADCMIP[1:0]		ADCPTY[1:0]	
DMA_SPI_CFG	FA20H	SPIIE	ACT_TX	ACT_RX	-	SPIIP[1:0]		SPIPTY[1:0]	
DMA_UR1T_CFG	FA30H	UR1TIE	-	-	-	UR1TIP[1:0]		UR1TPTY[1:0]	
DMA_UR1R_CFG	FA38H	UR1RIE	-	-	-	UR1RIP[1:0]		UR1RPTY[1:0]	
DMA_UR2T_CFG	FA40H	UR2TIE	-	-	-	UR2TIP[1:0]		UR2TPTY[1:0]	
DMA_UR2R_CFG	FA48H	UR2RIE	-	-	-	UR2RIP[1:0]		UR2RPTY[1:0]	
DMA_UR3T_CFG	FA50H	UR3TIE	-	-	-	UR3TIP[1:0]		UR3TPTY[1:0]	
DMA_UR3R_CFG	FA58H	UR3RIE	-	-	-	UR3RIP[1:0]		UR3RPTY[1:0]	
DMA_UR3R_CFG	FA60H	UR4TIE	-	-	-	UR4TIP[1:0]		UR4TPTY[1:0]	
DMA_UR4R_CFG	FA68H	UR4RIE	-	-	-	UR4RIP[1:0]		UR4RPTY[1:0]	
DMA_LCM_CFG	FA70H	LCMIE	-	-	-	LCMIP[1:0]		LCMPTY[1:0]	

M2MIE: DMA_M2M (Memory-to-Memory DMA) interrupt enable bit.

0: disable DMA_M2M interrupt
1: enable DMA_M2M interrupt

ADCIE: DMA_ADC (ADC DMA) interrupt enable bit.

0: disable DMA_ADC interrupt
1: enable DMA_ADC interrupt

SPIIE: DMA_SPI (SPI DMA) interrupt enable bit.

0: disable DMA_SPI interrupt
1: enable DMA_SPI interrupt

UR1TIE: DMA_UR1T (UART 1 send DMA) interrupt enable bit.

0: disable DMA_UR1T interrupt
1: enable DMA_UR1T interrupt

UR1RIE: DMA_UR1R (UART 1 receive DMA) interrupt enable bit.

0: disable DMA_UR1R interrupt
1: enable DMA_UR1R interrupt

UR2TIE: DMA_UR2T (UART 2 send DMA) interrupt enable bit.

0: disable DMA_UR2T interrupt
1: enable DMA_UR2T interrupt

UR2RIE: DMA_UR2R (UART 2 receive DMA) interrupt enable bit.

0: disable DMA_UR2R interrupt
1: enable DMA_UR2R interrupt

UR3TIE: DMA_UR3T (UART 3 send DMA) interrupt enable bit.

0: disable DMA_UR3T interrupt
1: enable DMA_UR3T interrupt

UR3RIE: DMA_UR3R (UART 3 receive DMA) interrupt enable bit.

0: disable DMA_UR3R interrupt
1: enable DMA_UR3R interrupt

UR4TIE: DMA_UR4T (UART 4 send DMA) interrupt enable bit.

0: disable DMA_UR4T interrupt
1: enable DMA_UR4T interrupt

UR4RIE: DMA_UR4R (UART 4 receive DMA) interrupt enable bit.

0: disable DMA_UR4R interrupt
1: enable DMA_UR4R interrupt

LCMIE: DMA_LCM (LCM interface DMA) interrupt enable bit.

0: disable DMA_LCM interrupt
1: enable DMA_LCM interrupt

11.4.2 Interrupt Request Registers (Interrupt flags)

Timer 0 and 1 Control Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
TCON	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1: Timer/Counter 1 Overflow Flag. Set by hardware on Timer/Counter 1 overflow. It will be automatically cleared by the hardware when processor enters the Timer 1 interrupt service routine.

TF0: Timer/Counter 0 Overflow Flag. Set by hardware on Timer/Counter 0 overflow. It will be automatically cleared by the hardware when processor enters the Timer 1 interrupt service routine.

IE1: External Interrupt 1 request flag. It will be automatically cleared when the processor enters the external interrupt 1 service routine.

IE0: External Interrupt 0 request flag. It will be automatically cleared when the processor enters the external interrupt 0 service routine.

Auxiliary Interrupt Flag Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
AUXINTIF	EFH	-	INT4IF	INT3IF	INT2IF	-	T4IF	T3IF	T2IF

INT4IF: external interrupt 4 request flag, which must be cleared by software.

INT3IF: external interrupt 3 request flag, which must be cleared by software.

INT2IF: external interrupt 2 request flag, which must be cleared by software.

T4IF: timer 4 overflow interrupt flag, which must be cleared by software (Note: This bit is a write-only register, not readable).

T3IF: timer 3 overflow interrupt flag, which must be cleared by software (Note: This bit is a write-only register, not readable).

T2IF: timer 2 overflow interrupt flag, which must be cleared by software (Note: This bit is a write-only register, not readable).

Notice:

Ealy 1T 8051 MCU using 0.35um process, STC15 series added a 16-bit reload timer which was the world's first big innovation of 8051. Due to high manufacturing cost, STC 16-bit reloadable timer 2/3/4 did not design the interrupt request flag registers for users to access. The interrupt request flag register has only internal hidden flags. The method provided to the user software to clear the internal hidden flags is: when the user software disables the timer 2/3/4 interrupt, the hardware automatically clears the internal timer 2/3/4. Hide interrupt request flags.

For product consistency:

The STC8A/ STC8F and subsequent STC8G/STC8H/ STC8C/ STC12H series which adopt 0.18um process add an interrupt request flag register accessible by the timer 2/3/4 user, but when the timer 2/3/4 interrupt is disabled , the function of the internal hidden interrupt request flag bit of the hardware automatic clear timer 2/3/4 is still retained. Therefore, do not arbitrarily disable the timer 2/3/4 interrupt when the timer 2/3/4 does not stop counting, otherwise the hidden interrupt request flag that actually works will be cleared. It is possible that after the counter overflows again, there is also a case that after the hidden interrupt request flag is set to 1, and request an interrupt and wait, it is mistakenly cleared by the user.

This is different from the traditional INTEL8048, 8051, but INTEL has been discontinued, so the new design of STC does not consider the specifications compatible with traditional INTEL.

This is the further development of 8051 by China STC.

UARTs Control Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
SCON	98H	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI
S2CON	9AH	S2SM0	-	S2SM2	S2REN	S2TB8	S2RB8	S2TI	S2RI
S3CON	ACH	S3SM0	S3ST3	S3SM2	S3REN	S3TB8	S3RB8	S3TI	S3RI
S4CON	84H	S4SM0	S4ST4	S4SM2	S4REN	S4TB8	S4RB8	S4TI	S4RI

TI: Transmit interrupt flag of UART1, which must be cleared by software.

RI: Receive interrupt flag of UART1, which must be cleared by software.

S2TI: Transmit interrupt flag of UART2, which must be cleared by software.

S2RI: Receive interrupt flag of UART2, which must be cleared by software.

S3TI: Transmit interrupt flag of UART3, which must be cleared by software.

S3RI: Receive interrupt flag of UART3, which must be cleared by software.

S4TI: Transmit interrupt flag of UART4, which must be cleared by software.

S4RI: Receive interrupt flag of UART4, which must be cleared by software.

Power Control Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
PCON	87H	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL

LVDF: Low voltage detection interrupt flag, which must be cleared by software.

ADC Control Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
ADC_CONTR	BCH	ADC_POWER	ADC_START	ADC_FLAG	ADC_EPWMT	ADC_CHS[3:0]			

ADC_FLAG: ADC completes conversion interrupt request flag, which must be cleared by software.

SPI Status Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
SPSTAT	CDH	SPIF	WCOL	-	-	-	-	-	-

SPIF: SPI transmission completion interrupt request flag, which must be cleared by software.

Comparator Control Register 1

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
CMPCR1	E6H	CMPEN	CMPIF	PIE	NIE	PIS	NIS	CMPOE	CMPRES

CMPIF: Comparator interrupt request flag, which must be cleared by software.

I2C Status Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
I2CMSST	FE82H	MSBUSY	MSIF	-	-	-	-	MSACKI	MSACKO
I2CSLST	FE84H	SLBUSY	STAIF	RXIF	TXIF	STOIF	TXING	SLACKI	SLACKO

MSIF: I^{2}C master mode interrupt request flag, which must be cleared by software.

ESTAI: I^{2}C slave receives the START event interrupt request flag, which must be cleared by software.

ERXI: I^{2}C slave completes receiving data event interrupt request flag, which must be cleared by software.

ETXI: I^{2}C slave completes transmitting data event interrupt request flag, which must be cleared by software.

ESTOI: I^{2}C slave receives the STOP event interrupt request flag, which should be must by software.

PWMA Status Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_SR1	FEC5H	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
PWMA_SR2	FEC6H	-	-	-	CC4OF	CC3OF	CC2OF	CC1OF	-

BIF: PWMA brake interrupt request flag. Need to be cleared by software.
TIF: PWMA triggers the interrupt request flag. Need to be cleared by software.
COMIF: PWMA compare interrupt request flag. Need to be cleared by software.
CC4IF: A capture compare interrupt request flag occurred in PWMA channel 4. Need to be cleared by software.
CC3IF: A capture compare interrupt request flag occurred in PWMA channel 3. Need to be cleared by software.
CC2IF: A capture compare interrupt request flag occurred in PWMA channel 2. Need to be cleared by software.
CC1IF: A capture compare interrupt request flag occurred in PWMA channel 1. Need to be cleared by software.
TIF: PWMA update interrupt request flag. Need to be cleared by software.
CC4OF: PWMA channel 4 has repeated capture interrupt request flag. Need to be cleared by software.

CC3OF: PWMA channel 3 has repeated capture interrupt request flag. Need to be cleared by software.
CC2OF: PWMA channel 2 has repeated capture interrupt request flag. Need to be cleared by software.
CC1OF: PWMA channel 1 repeated capture interrupt request flag. Need to be cleared by software.

PWMB Status Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
PWMB_SR1	FEC5H	BIF	TIF	COMIF	CC8IF	CC7IF	CC6IF	CC5IF	UIF
PWMB_SR2	FEC6H	-	-	-	CC8OF	CC7OF	CC6OF	CC5OF	-

BIF: PWMB brake interrupt request flag. Need to be cleared by software.
TIF: PWMB triggers the interrupt request flag. Need to be cleared by software.
COMIF: PWMB compare interrupt request flag. Need to be cleared by software.
CC8IF: A capture compare interrupt request flag occurred in PWMB channel 8. Need to be cleared by software.
CC7IF: A capture compare interrupt request flag occurred in PWMB channel 7. Need to be cleared by software.
CC6IF: A capture compare interrupt request flag occurred in PWMB channel 6. Need to be cleared by software.
CC5IF: A capture compare interrupt request flag occurred in PWMB channel 5. Need to be cleared by software.
TIF: PWMB update interrupt request flag. Need to be cleared by software.
CC8OF: PWMB channel 8 has repeated capture interrupt request flag. Need to be cleared by software.
CC7OF: PWMB channel 7 has repeated capture interrupt request flag. Need to be cleared by software.
CC6OF: PWMB channel 6 has repeated capture interrupt request flag. Need to be cleared by software.
CC5OF: PWMB channel 5 repeated capture interrupt request flag. Need to be cleared by software.

Port interrupt flag registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P0INTF	FD10H	P07INTF	P06INTF	P05INTF	P04INTF	P03INTF	P02INTF	P01INTF	P00INTF
P1INTF	FD11H	P17INTF	P16INTF	P15INTF	P14INTF	P13INTF	P12INTF	P11INTF	P10INTF
P2INTF	FD12H	P27INTF	P26INTF	P25INTF	P24INTF	P23INTF	P22INTF	P21INTF	P20INTF
P3INTF	FD13H	P37INTF	P36INTF	P35INTF	P34INTF	P33INTF	P32INTF	P31INTF	P30INTF
P4INTF	FD14H	P47INTF	P46INTF	P45INTF	P44INTF	P43INTF	P42INTF	P41INTF	P40INTF
P5INTF	FD15H	-	-	P55INTF	P54INTF	P53INTF	P52INTF	P51INTF	P50INTF
P6INTF	FD16H	P67INTF	P66INTF	P65INTF	P64INTF	P63INTF	P62INTF	P61INTF	P60INTF
P7INTF	FD17H	P77INTF	P76INTF	P75INTF	P74INTF	P73INTF	P72INTF	P71INTF	P70INTF

PnINTF.x: Port interrupt request flag (n=0~7, x=0~7)

0: No interrupt request for Pn.x port

1: Pn.x port has an interrupt request, if the interrupt is enabled, it will enter the interrupt service routine. The flag bit needs to be cleared by software.

LCM Interface Status Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
LCMIFSTA	FE53H	-	-	-	-	-	-	-	LCMIFIF

LCMIFIF: LCM interface interrupt request flag. Need to be cleared by software.

DMA interrupt flag registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_M2M_STA	FA02H	-	-	-	-	-	-	-	M2MIF
DMA_ADC_STA	FA12H	-	-	-	-	-	-	-	ADCIF
DMA_SPI_STA	FA22H	-	-	-	-	-	TXOVW	RXLOSS	SPIIF
DMA_UR1T_STA	FA32H	-	-	-	-	-	TXOVW	-	UR1TIF
DMA_UR1R_STA	FA3AH	-	-	-	-	-	-	RXLOSS	UR1RIF
DMA_UR2T_STA	FA42H	-	-	-	-	-	TXOVW	-	UR2TIF
DMA_UR2R_STA	FA4AH	-	-	-	-	-	-	RXLOSS	UR2RIF
DMA_UR3T_STA	FA52H	-	-	-	-	-	TXOVW	-	UR3TIF
DMA_UR3R_STA	FA5AH	-	-	-	-	-	-	RXLOSS	UR3RIF
DMA_UR4T_STA	FA62H	-	-	-	-	-	TXOVW	-	UR4TIF
DMA_UR4R_STA	FA6AH	-	-	-	-	-	-	RXLOSS	UR4RIF
DMA_LCM_STA	FA72H	-	-	-	-	-	-	TXOVW	LCMIF

M2MIF: DMA_M2M(Memory-to-Memory DMA) interrupt request flag. Need to be cleared by software.

ADCIF: DMA_ADC(ADC DMA) interrupt request flag. Need to be cleared by software.

SPIIF: DMA_SPI(SPI DMA) interrupt request flag. Need to be cleared by software.。

UR1TIF: DMA_UR1T(UART1 send DMA) interrupt request flag. Need to be cleared by software.

UR1RIF: DMA_UR1R(UART1 receive DMA) interrupt request flag. Need to be cleared by software.
UR2TIF: DMA_UR2T(UART2 send DMA) interrupt request flag. Need to be cleared by software.
UR2RIF: DMA_UR2R(UART2 receive DMA) interrupt request flag. Need to be cleared by software.
UR3TIF: DMA_UR3T(UART3 send DMA) interrupt request flag. Need to be cleared by software.
UR3RIF: DMA_UR3R(UART3 receive DMA) interrupt request flag. Need to be cleared by software.
UR4TIF: DMA_UR4T(UART4 send DMA) interrupt request flag. Need to be cleared by software.
UR4RIF: DMA_UR4R(UART4 receive DMA) interrupt request flag. Need to be cleared by software.
LCMIF: DMA_LCM(LCM interface DMA) interrupt request flag. Need to be cleared by software.

11.4.3 Interrupt Priority Control Registers

Except INT2, INT3, Timer 2, Timer 3, Timer 4 and all port interrupts, all other interrupts have 4 levels of interrupt priority that can be set.

Interrupt Priority Control Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
IP	B8H	-	PLVD	PADC	PS	PT1	PX1	PT0	PX0
IPH	B7H	-	PLVDH	PADCH	PSH	PT1H	PX1H	PT0H	PX0H
IP2	B5H	PUSB PTKSU	PI2C	PCMP	PX4	PPWMB	PPWMA	PSPI	PS2
IP2H	B6H	PUSBH PTKSUH	PI2CH	PCMPH	PX4H	PPWMBH	PPWMAH	PSPIH	PS2H
IP3	DFH	-	-	-	-	-	PRTC	PS4	PS3
IP3H	EEH	-	-	-	-	-	PRTCH	PS4H	PS3H

PX0H,PX0: External interrupt 0 interrupt priority control bit.

00: INT0 interrupt priority level is 0 (lowest)
01: INT0 interrupt priority level is 1 (lower)
10: INT0 interrupt priority level is 2 (higher)
11: INT0 interrupt priority level is 3 (highest)

PT0H,PT0: Timer 0 interrupt priority control bit.

00: Timer 0 interrupt priority level is 0 (lowest)
01: Timer 0 interrupt priority level is 1 (lower)
10: Timer 0 interrupt priority level is 2 (higher)
11: Timer 0 interrupt priority level is 3 (highest)

PX1H,PX1: External interrupt 1 interrupt priority control bit.

00: INT1 interrupt priority level is 0 (lowest)
01: INT1 interrupt priority level is 1 (lower)
10: INT1 interrupt priority level is 2 (higher)
11: INT1 interrupt priority level is 3 (highest)

PT1H,PT1: Timer 1 interrupt priority control bit.

00: Timer 1 interrupt priority level is 0 (lowest)
01: Timer 1 interrupt priority level is 1 (lower)
10: Timer 1 interrupt priority level is 2 (higher)
11: Timer 1 interrupt priority level is 3 (highest)

PSH,PS: UART1 interrupt priority control bit.

00: UART1 interrupt priority level is 0 (lowest)
01: UART1 interrupt priority level is 1 (lower)
10: UART1 interrupt priority level is 2 (higher)
11: UART1 interrupt priority level is 3 (highest)

PADCH,PADC: ADC interrupt priority control bit.

00: ADC interrupt priority level is 0 (lowest)
01: ADC interrupt priority level is 1 (lower)
10: ADC interrupt priority level is 2 (higher)
11: ADC interrupt priority level is 3 (highest)

PLVDH,PLVD: Low voltage detection interrupt priority control bit.

00: LVD interrupt priority level is 0 (lowest)
01: LVD interrupt priority level is 1 (lower)
10: LVD interrupt priority level is 2 (higher)
11: LVD interrupt priority level is 3 (highest)

PS2H,PS2: UART2 interrupt priority control bit.

00: UART2 interrupt priority level is 0 (lowest)
01: UART2 interrupt priority level is 1 (lower)
10: UART2 interrupt priority level is 2 (higher)
11: UART2 interrupt priority level is 3 (highest)

PS3H,PS3: UART3 interrupt priority control bit.

00: UART3 interrupt priority level is 0 (lowest)
01: UART3 interrupt priority level is 1 (lower)
10: UART3 interrupt priority level is 2 (higher)
11: UART3 interrupt priority level is 3 (highest)

PS4H,PS4: UART4 interrupt priority control bit.

00: UART4 interrupt priority level is 0 (lowest)
01: UART4 interrupt priority level is 1 (lower)
10: UART4 interrupt priority level is 2 (higher)
11: UART4 interrupt priority level is 3 (highest)

PSPIH,PSPI: SPI interrupt priority control bit.

00: SPI interrupt priority level is 0 (lowest)
01: SPI interrupt priority level is 1 (lower)
10: SPI interrupt priority level is 2 (higher)
11: SPI interrupt priority level is 3 (highest)

PPWMAH,PPWMA: Advanced PWMA interrupt priority control bit.

00: Advanced PWMA interrupt priority level is 0 (lowest)
01: Advanced PWMA interrupt priority level is 1 (lower)
10: Advanced PWMA interrupt priority level is 2 (higher)
11: Advanced PWMA interrupt priority level is 3 (highest)

PPWMBH,PPWMB: Advanced PWMB interrupt priority control bit.

00: Advanced PWMB interrupt priority level is 0 (lowest)
01: Advanced PWMB interrupt priority level is 1 (lower)
10: Advanced PWMB interrupt priority level is 2 (higher)
11: Advanced PWMB interrupt priority level is 3 (highest)

PX4H,PX4: External interrupt 4 interrupt priority control bit.

00: INT4 interrupt priority level is 0 (lowest)
01: INT4 interrupt priority level is 1 (lower)
10: INT4 interrupt priority level is 2 (higher)
11: INT4 interrupt priority level is 3 (highest)

PCMPH,PCMP: Comparator interrupt priority control bit.

00: CMP interrupt priority level is 0 (lowest)
01: CMP interrupt priority level is 1 (lower)
10: CMP interrupt priority level is 2 (higher)
11: CMP interrupt priority level is 3 (highest)

PI2CH,PI2C: I2C interrupt priority control bit.

00: I2C interrupt priority level is 0 (lowest)
01: I2C interrupt priority level is 1 (lower)
10: I2C interrupt priority level is 2 (higher)
11: I2C interrupt priority level is 3 (highest)

PUSBH, PUSB: USB interrupt priority control bit

00: USB interrupt priority is level 0 (the lowest level)
01: USB interrupt priority is level 1 (lower level)
10: USB interrupt priority is level 2 (higher level)
11: USB interrupt priority is level 3 (the highest level)

PTKSUH, PTKSU: touch key interrupt priority control bit

00: Touch key interrupt priority is 0 (the lowest level)

01: Touch key interrupt priority is level 1 (lower level)
10: Touch button interrupt priority is level 2 (higher level)
11: Touch key interrupt priority is level 3 (the highest level)

PRTCH, PRTC: RTC interrupt priority control bit

00: RTC interrupt priority level is 0 (lowest level)
01: RTC interrupt priority is level 1 (lower level)
10: RTC interrupt priority is level 2 (higher level)
11: RTC interrupt priority level is 3 (the highest level)

LCM Interface Configuration Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
LCMIFCFG	FE50H	LCMIFIE	-	LCMIFIP[1:0]		LCMIFDPS[1:0]		D16_D8	M68_I80

LCMIFIP[1:0]: LCM interface interrupt priority control bits

00: LCM interface interrupt priority level is 0 (lowest level)
01: LCM interface interrupt priority is level 1 (lower level)
10: LCM interface interrupt priority is level 2 (higher level)
11: LCM interface interrupt priority level is 3 (the highest level)

Port Interrupt Priority Control Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
PINIPL	FD60H	P7IP	P6IP	P5IP	P4IP	P3IP	P2IP	P1IP	P0IP
PINIPH	FD61H	P7IPH	P6IPH	P5IPH	P4IPH	P3IPH	P2IPH	P1IPH	P0IPH

P0IPH,P0IP: P0 interrupt priority control bits

00: P0 interrupt priority level is 0 (lowest level)
01: P0 interrupt priority is level 1 (lower level)
10: P0 interrupt priority is level 2 (higher level)
11: P0 interrupt priority level is 3 (the highest level)

P1IPH,P1IP: P1 interrupt priority control bits

00: P1 interrupt priority level is 0 (lowest level)
01: P1 interrupt priority is level 1 (lower level)
10: P1 interrupt priority is level 2 (higher level)
11: P1 interrupt priority level is 3 (the highest level)

P2IPH,P2IP: P2 interrupt priority control bits

00: P2 interrupt priority level is 0 (lowest level)
01: P2 interrupt priority is level 1 (lower level)
10: P2 interrupt priority is level 2 (higher level)
11: P2 interrupt priority level is 3 (the highest level)

P3IPH,P3IP: P3 interrupt priority control bits

00: P3 interrupt priority level is 0 (lowest level)
01: P3 interrupt priority is level 1 (lower level)
10: P3 interrupt priority is level 2 (higher level)
11: P3 interrupt priority level is 3 (the highest level)

P4IPH,P4IP: P4 interrupt priority control bits

00: P4 interrupt priority level is 0 (lowest level)
01: P4 interrupt priority is level 1 (lower level)
10: P4 interrupt priority is level 2 (higher level)
11: P4 interrupt priority level is 3 (the highest level)

P5IPH,P5IP: P5 interrupt priority control bits

00: P5 interrupt priority level is 0 (lowest level)
01: P5 interrupt priority is level 1 (lower level)
10: P5 interrupt priority is level 2 (higher level)
11: P5 interrupt priority level is 3 (the highest level)

P6IPH,P6IP: P6 interrupt priority control bits

00: P6 interrupt priority level is 0 (lowest level)
01: P6 interrupt priority is level 1 (lower level)
10: P6 interrupt priority is level 2 (higher level)
11: P6 interrupt priority level is 3 (the highest level)

P7IPH,P7IP: P7 interrupt priority control bits

00: P7 interrupt priority level is 0 (lowest level)
01: P7 interrupt priority is level 1 (lower level)
10: P7 interrupt priority is level 2 (higher level)
11: P7 interrupt priority level is 3 (the highest level)

DMA Interrupt Priority Control Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_M2M_CFG	FA00H	M2MIE	-	TXACO	RXACO	M2MIP[1:0]		M2MPTY[1:0]	
DMA_ADC_CFG	FA10H	ADCIE	-	-	-	ADCMIP[1:0]		ADCPTY[1:0]	
DMA_SPI_CFG	FA20H	SPIIE	ACT_TX	ACT_RX	-	SPIIP[1:0]		SPIPTY[1:0]	
DMA_UR1T_CFG	FA30H	UR1TIE	-	-	-	UR1TIP[1:0]		UR1TPTY[1:0]	
DMA_UR1R_CFG	FA38H	UR1RIE	-	-	-	UR1RIP[1:0]		UR1RPTY[1:0]	
DMA_UR2T_CFG	FA40H	UR2TIE	-	-	-	UR2TIP[1:0]		UR2TPTY[1:0]	
DMA_UR2R_CFG	FA48H	UR2RIE	-	-	-	UR2RIP[1:0]		UR2RPTY[1:0]	
DMA_UR3T_CFG	FA50H	UR3TIE	-	-	-	UR3TIP[1:0]		UR3TPTY[1:0]	
DMA_UR3R_CFG	FA58H	UR3RIE	-	-	-	UR3RIP[1:0]		UR3RPTY[1:0]	
DMA_UR3R_CFG	FA60H	UR4TIE	-	-	-	UR4TIP[1:0]		UR4TPTY[1:0]	
DMA_UR4R_CFG	FA68H	UR4RIE	-	-	-	UR4RIP[1:0]		UR4RPTY[1:0]	
DMA_LCM_CFG	FA70H	LCMIE	-	-	-	LCMIP[1:0]		LCMPTY[1:0]	

M2MIP: DMA_M2M(Memory-to Memory DMA)interrupt priority control bits

00: DMA_M2M interrupt priority level is 0 (lowest level)
01: DMA_M2M interrupt priority is level 1 (lower level)
10: DMA_M2M interrupt priority is level 2 (higher level)
11: DMA_M2M interrupt priority level is 3 (the highest level)

ADCIP: DMA_ADC(ADC DMA)interrupt priority control bits

00: DMA_ADC interrupt priority level is 0 (lowest level)
01: DMA_ADC interrupt priority is level 1 (lower level)
10: DMA_ADC interrupt priority is level 2 (higher level)
11: DMA_ADC interrupt priority level is 3 (the highest level)

SPIIP: DMA_SPI(SPI DMA)interrupt priority control bits

00: DMA_SPI interrupt priority level is 0 (lowest level)
01: DMA_SPI interrupt priority is level 1 (lower level)
10: DMA_SPI interrupt priority is level 2 (higher level)
11: DMA_SPI interrupt priority level is 3 (the highest level)

UR1TIP: DMA_UR1T(UART 1 send DMA)interrupt priority control bits

00: DMA_UR1T interrupt priority level is 0 (lowest level)
01: DMA_UR1T interrupt priority is level 1 (lower level)
10: DMA_UR1T interrupt priority is level 2 (higher level)
11: DMA_UR1T interrupt priority level is 3 (the highest level)

UR1RIP: DMA_UR1R(UART 1 receive DMA)interrupt priority control bits

00: DMA_UR1R interrupt priority level is 0 (lowest level)
01: DMA_UR1R interrupt priority is level 1 (lower level)
10: DMA_UR1R interrupt priority is level 2 (higher level)
11: DMA_UR1R interrupt priority level is 3 (the highest level)

UR2TIP: DMA_UR2T(UART 2 send DMA)interrupt priority control bits

00: DMA_UR2T interrupt priority level is 0 (lowest level)
01: DMA_UR2T interrupt priority is level 1 (lower level)
10: DMA_UR2T interrupt priority is level 2 (higher level)
11: DMA_UR2T interrupt priority level is 3 (the highest level)

UR2RIP: DMA_UR2R(UART 2 receive DMA)interrupt priority control bits

00: DMA_UR2R interrupt priority level is 0 (lowest level)
01: DMA_UR2R interrupt priority is level 1 (lower level)
10: DMA_UR2R interrupt priority is level 2 (higher level)
11: DMA_UR2R interrupt priority level is 3 (the highest level)

UR3TIP: DMA_UR3T(UART 3 send DMA)interrupt priority control bits

00: DMA_UR3T interrupt priority level is 0 (lowest level)
01: DMA_UR3T interrupt priority is level 1 (lower level)

10: DMA_UR3T interrupt priority is level 2 (higher level)
11: DMA_UR3T interrupt priority level is 3 (the highest level)
UR3RIP: DMA_UR3R(UART 3 receive DMA)interrupt priority control bits

00: DMA_UR3R interrupt priority level is 0 (lowest level)
01: DMA_UR3R interrupt priority is level 1 (lower level)
10: DMA_UR3R interrupt priority is level 2 (higher level)
11: DMA_UR3R interrupt priority level is 3 (the highest level)
UR4TIP: DMA_UR4T(UART 4 send DMA)interrupt priority control bits

00: DMA_UR3R interrupt priority level is 0 (lowest level)
01: DMA_UR3R interrupt priority is level 1 (lower level)
10: DMA_UR3R interrupt priority is level 2 (higher level)
11: DMA_UR3R interrupt priority level is 3 (the highest level)
UR4RIP: DMA_UR4R(UART 4 receive DMA)interrupt priority control bits

00: DMA_UR4R interrupt priority level is 0 (lowest level)
01: DMA_UR4R interrupt priority is level 1 (lower level)
10: DMA_UR4R interrupt priority is level 2 (higher level)
11: DMA_UR4R interrupt priority level is 3 (the highest level)
LCMIP: DMA_LCM(LCM interfaceDMA)interrupt priority control bits

00: DMA_LCM interrupt priority level is 0 (lowest level)
01: DMA_LCM interrupt priority is level 1 (lower level)
10: DMA_LCM interrupt priority is level 2 (higher level)
11: DMA_LCM interrupt priority level is 3 (the highest level)

11.5 Example Routines

11.5.1 INT0 Interrupt (Rising and Falling Edges)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sbit        P10         =       P1^0;
sbit        P11         =       P1^1;

void INT0_Isr() interrupt 0
{
        if (INT0)                                           //Judging rising and falling edges
        {
                P10 = !P10;                                 //Test port
        }
        else
        {
                P11 = !P11;                                 //Test port
        }
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        IT0 = 0;                                            //Enable INT0 rising edge and falling edge interrupts
        EX0 = 1;                                            //Enable INT0 interrupt
        EA = 1;
```

```
    while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0003H
                LJMP            INT0ISR

                ORG             0100H
INT0ISR:
                JB              INT0,RISING                    ;Judging rising and falling edges
                CPL             P1.0                           ;Test port
                RETI
RISING:
                CPL             P1.1                           ;Test port
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                CLR             IT0                            ;Enable INT0 rising edge and falling edge interrupts
                SETB            EX0                            ;Enable INT0 interrupt
                SETB            EA
                JMP             $

                END
```

11.5.2 INT0 Interrupt (Falling Edge)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sbit        P10         =       P1^0;

void INT0_Isr() interrupt 0
{
        P10 = !P10;                                         //Test port
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        IT0 = 1;                                            //Enable INT0 falling edge interrupt
        EX0 = 1;                                            //Enable INT0 interrupt
        EA = 1;

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P0M1            DATA        093H
P0M0            DATA        094H
P1M1            DATA        091H
P1M0            DATA        092H
P2M1            DATA        095H
P2M0            DATA        096H
```

```
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0003H
                LJMP            INT0ISR

                ORG             0100H
INT0ISR:
                CPL             P1.0                    ;Test port
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                SETB            IT0                     ;Enable INT0 falling edge interrupt
                SETB            EX0                     ;Enable INT0 interrupt
                SETB            EA
                JMP             $

                END
```

11.5.3 INT1 Interrupt (Rising and Falling Edges)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sbit        P10         =       P1^0;
sbit        P11         =       P1^1;

void INT1_Isr() interrupt 2
{
        if (INT1)                                       //Judging rising and falling edges
        {
                P10 = !P10;                             //Test port
        }
        else
        {
                P11 = !P11;                             //Test port
        }
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        IT1 = 0;                                        //Enable INT1 rising edge and falling edge interrupts
        EX1 = 1;                                        //Enable INT1 interrupt
        EA = 1;

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0013H
                LJMP            INT1ISR
                ORG             0100H
INT1ISR:
                JB              INT1,RISING                     ;Judging rising and falling edges
                CPL             P1.0                            ;Test port
                RETI
RISING:
                CPL             P1.1                            ;Test port
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                CLR             IT1                             ;Enable INT1 rising edge and falling edge interrupts
                SETB            EX1                             ;Enable INT1 interrupt
                SETB            EA
                JMP             $

                END
```

11.5.4 INT1 Interrupt (Falling Edge)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sbit        P10         =       P1^0;

void INT1_Isr() interrupt 2
{
    P10 = !P10;                                     //Test port
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    IT1 = 1;                                        //Enable INT1 falling edge interrupt
    EX1 = 1;                                        //Enable INT1 interrupt
    EA = 1;

    while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0013H
                LJMP            INT1ISR

                ORG             0100H
INT1ISR:
                CPL             P1.0                            ;Test port
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                SETB        IT1                         ;Enable INT1 falling edge interrupt
                SETB        EX1                         ;Enable INT1 interrupt
                SETB        EA
                JMP         $

                END
```

11.5.5 INT2 Interrupt (Falling Edge)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sfr         INTCLKO     =       0x8f;
#define     EX2                 0x10
#define     EX3                 0x20
#define     EX4                 0x40
sbit        P10         =       P1^0;

void INT2_Isr() interrupt 10
{
        P10 = !P10;                                     //Test port
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        INTCLKO = EX2;                                      //Enable INT2 interrupt
        EA = 1;

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

INTCLKO         DATA            8FH
EX2             EQU             10H
EX3             EQU             20H
EX4             EQU             40H

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0053H
                LJMP            INT2ISR

                ORG             0100H
INT2ISR:
                CPL             P1.0                        ;Test port
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             INTCLKO,#EX2                ;Enable INT2 interrupt
                SETB            EA
                JMP             $

                END
```

11.5.6 INT3 Interrupt (Falling Edge)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sfr         INTCLKO     =       0x8f;
#define     EX2                 0x10
#define     EX3                 0x20
#define     EX4                 0x40
sbit        P10         =       P1^0;

void INT3_Isr() interrupt 11
{
        P10 = !P10;                                             //Test port
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        INTCLKO = EX3;                                          //Enable INT3 interrupt
        EA = 1;

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

INTCLKO         DATA            8FH
```

```
EX2             EQU             10H
EX3             EQU             20H
EX4             EQU             40H

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             005BH
                LJMP            INT3ISR

                ORG             0100H
INT3ISR:
                CPL             P1.0                            ;Test port
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             INTCLKO,#EX3                    ;Enable INT3 interrupt
                SETB            EA
                JMP             $

                  END
```

11.5.7 INT4 Interrupt (Falling Edge)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1        =     0x91;
sfr         P1M0        =     0x92;
sfr         P2M1        =     0x95;
sfr         P2M0        =     0x96;
sfr         P3M1        =     0xb1;
sfr         P3M0        =     0xb2;
sfr         P4M1        =     0xb3;
sfr         P4M0        =     0xb4;
sfr         P5M1        =     0xc9;
sfr         P5M0        =     0xca;

sfr         INTCLKO     =     0x8f;
#define     EX2               0x10
#define     EX3               0x20
#define     EX4               0x40
sbit        P10         =     P1^0;

void INT4_Isr() interrupt 16
{
        P10 = !P10;                                     //Test port
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        INTCLKO = EX4;                                  //Enable INT4 interrupt
        EA = 1;

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

INTCLKO         DATA        8FH
EX2             EQU         10H
EX3             EQU         20H
EX4             EQU         40H

P0M1            DATA        093H
P0M0            DATA        094H
P1M1            DATA        091H
P1M0            DATA        092H
P2M1            DATA        095H
P2M0            DATA        096H
P3M1            DATA        0B1H
P3M0            DATA        0B2H
```

```
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0083H
                LJMP            INT4ISR

                ORG             0100H
INT4ISR:
                CPL             P1.0                            ;Test port
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             INTCLKO,#EX4                    ;Enable INT4 interrupt
                SETB            EA
                JMP             $

                  END
```

11.5.8 Timer0 Interrupt

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sbit        P10         =       P1^0;
```

```
void TM0_Isr() interrupt 1
{
        P10 = !P10;                                     //Test port
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        TMOD = 0x00;
        TL0 = 0x66;                                     //65536-11.0592M/12/1000
        TH0 = 0xfc;
        TR0 = 1;                                        //Start timer
        ET0 = 1;                                        //Enable timer0 interrupt
        EA = 1;

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             000BH
                LJMP            TM0ISR

                ORG             0100H
TM0ISR:
                CPL             P1.0                    ;Test port
                RETI

MAIN:
                MOV             SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                MOV         TMOD,#00H
                MOV         TL0,#66H                    ;65536-11.0592M/12/1000
                MOV         TH0,#0FCH
                SETB        TR0                         ;Start timer
                SETB        ET0                         ;Enable timer0 interrupt
                SETB        EA

                JMP         $

                END
```

11.5.9 Timer1 Interrupt

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sbit        P10         =       P1^0;

void TM1_Isr() interrupt 3
{
    P10 = !P10;                                 //Test port
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
```

```
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        TMOD = 0x00;
        TL1 = 0x66;                                  //65536-11.0592M/12/1000
        TH1 = 0xfc;
        TR1 = 1;                                     //Start timer
        ET1 = 1;                                     //Enable timer1 interrupt
        EA = 1;

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             001BH
                LJMP            TM1ISR

                ORG             0100H
TM1ISR:
                CPL             P1.0                            ;Test port
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H
```

```
        MOV         TMOD,#00H
        MOV         TL1,#66H                    ;65536-11.0592M/12/1000
        MOV         TH1,#0FCH
        SETB        TR1                         ;Start timer
        SETB        ET1                         ;Enable timer1 interrupt
        SETB        EA

        JMP         $

        END
```

11.5.10 Timer2 Interrupt

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         T2L         =       0xd7;
sfr         T2H         =       0xd6;
sfr         AUXR        =       0x8e;
sfr         IE2         =       0xaf;
#define     ET2                 0x04
sfr         AUXINTIF    =       0xef;
#define     T2IF                0x01

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sbit        P10         =       P1^0;

void TM2_Isr() interrupt 12
{
    P10 = !P10;                                 //Test port
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    T2L = 0x66;                                 //65536-11.0592M/12/1000
    T2H = 0xfc;
    AUXR = 0x10;                                //Start timer
    IE2 = ET2;                                  //Enable timer2 interrupt
    EA = 1;

    while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

T2L         DATA        0D7H
T2H         DATA        0D6H
AUXR        DATA        8EH
IE2         DATA        0AFH
ET2         EQU         04H
AUXINTIF    DATA        0EFH
T2IF        EQU         01H

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         0063H
            LJMP        TM2ISR

            ORG         0100H
TM2ISR:
            CPL         P1.0                    ;Test port
            RETI

MAIN:
            MOV         SP, #5FH
            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
            MOV         P3M1, #00H
            MOV         P4M0, #00H
            MOV         P4M1, #00H
            MOV         P5M0, #00H
            MOV         P5M1, #00H

            MOV         T2L,#66H                    ;65536-11.0592M/12/1000
            MOV         T2H,#0FCH
            MOV         AUXR,#10H                   ;Start timer
            MOV         IE2,#ET2                    ;Enable timer2 interrupt
            SETB        EA

            JMP         $

            END
```

11.5.11 Timer3 Interrupt

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         T3L         =       0xd5;
sfr         T3H         =       0xd4;
sfr         T4T3M       =       0xd1;
sfr         IE2         =       0xaf;
#define     ET3                 0x20
sfr         AUXINTIF    =       0xef;
#define     T3IF                0x02

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sbit        P10         =       P1^0;

void TM3_Isr() interrupt 19
{
    P10 = !P10;                             //Test port
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    T3L = 0x66;                     //65536-11.0592M/12/1000
    T3H = 0xfc;
    T4T3M = 0x08;                   //Start timer
    IE2 = ET3;                      //Enable timer3 interrupt
    EA = 1;

    while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

T3L         DATA        0D5H
T3H         DATA        0D4H
T4T3M       DATA        0D1H
IE2         DATA        0AFH
ET3         EQU         20H
AUXINTIF    DATA        0EFH
T3IF        EQU         02H

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         009BH
            LJMP        TM3ISR

            ORG         0100H
TM3ISR:
            CPL         P1.0                    ;Test port
            RETI

MAIN:
            MOV         SP, #5FH
            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
```

```
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                MOV         T3L,#66H                    ;65536-11.0592M/12/1000
                MOV         T3H,#0FCH
                MOV         T4T3M,#08H                  ;Start timer
                MOV         IE2,#ET3                    ;Enable timer3 interrupt
                SETB        EA

                JMP         $

                END
```

11.5.12 Timer4 Interrupt

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         T3L         =       0xd5;
sfr         T3H         =       0xd4;
sfr         T4L         =       0xd3;
sfr         T4H         =       0xd2;
sfr         T4T3M       =       0xd1;
sfr         IE2         =       0xaf;
#define     ET3                 0x20
#define     ET4                 0x40
sfr         AUXINTIF    =       0xef;
#define     T3IF                0x02
#define     T4IF                0x04

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sbit        P10         =       P1^0;

void TM4_Isr() interrupt 20
{
    P10 = !P10;                             //Test port
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    T4L = 0x66;                     //65536-11.0592M/12/1000
    T4H = 0xfc;
    T4T3M = 0x80;                   //Start timer
    IE2 = ET4;                      //Enable timer4 interrupt
    EA = 1;

    while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

T3L         DATA        0D5H
T3H         DATA        0D4H
T4L         DATA        0D3H
T4H         DATA        0D2H
T4T3M       DATA        0D1H
IE2         DATA        0AFH
ET3         EQU         20H
ET4         EQU         40H
AUXINTIF    DATA        0EFH
T3IF        EQU         02H
T4IF        EQU         04H

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         00A3H
            LJMP        TM4ISR

            ORG         0100H
TM4ISR:
            CPL         P1.0                    ;Test port
            RETI
```

```
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             T4L,#66H                        ;65536-11.0592M/12/1000
                MOV             T4H,#0FCH
                MOV             T4T3M,#80H                      ;Start timer
                MOV             IE2,#ET4                        ;Enable timer4 interrupt
                SETB            EA

                JMP             $

                END
```

11.5.13 UART1 Interrupt

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         T2L             =       0xd7;
sfr         T2H             =       0xd6;
sfr         AUXR            =       0x8e;

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

sbit        P10             =       P1^0;
sbit        P11             =       P1^1;

void UART1_Isr() interrupt 4
{
        if (TI)
        {
                TI = 0;                                         //Clear interrupt flag
                P10 = !P10;                                     //Test port
        }
        if (RI)
        {
                RI = 0;                                         //Clear interrupt flag
                P11 = !P11;                                     //Test port
        }
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        SCON = 0x50;
        T2L = 0xe8;                                             //65536-11059200/115200/4=0FFE8H
        T2H = 0xff;
        AUXR = 0x15;                                            //Start timer
        ES = 1;                                                 //Enable UART1 interrupt
        EA = 1;
        SBUF = 0x5a;                                            // Send test data

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

T2L             DATA            0D7H
T2H             DATA            0D6H
AUXR            DATA            8EH

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0023H
                LJMP            UART1ISR

                ORG             0100H
UART1ISR:
                JNB             TI,CHECKRI
                CLR             TI                              ;Clear interrupt flag
                CPL             P1.0                            ;Test port
CHECKRI:
                JNB             RI,ISREXIT
                CLR             RI                              ;Clear interrupt flag
                CPL             P1.1                            ;Test port
ISREXIT:
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             SCON,#50H
                MOV             T2L,#0E8H                       ;65536-11059200/115200/4=0FFE8H
                MOV             T2H,#0FFH
                MOV             AUXR,#15H                       ;Start timer
                SETB            ES                              ;Enable UART1 interrupt
                SETB            EA
                MOV             SBUF,#5AH                       ; Send test data

                JMP             $

                END
```

11.5.14 UART2 Interrupt

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         T2L         =       0xd7;
sfr         T2H         =       0xd6;
sfr         AUXR        =       0x8e;
sfr         S2CON       =       0x9a;
sfr         S2BUF       =       0x9b;
sfr         IE2         =       0xaf;
#define     ES2                 0x01

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sbit        P12         =       P1^2;
sbit        P13         =       P1^3;

void UART2_Isr() interrupt 8
{
        if (S2CON & 0x02)
        {
                S2CON &= ~0x02;                         //Clear interrupt flag
                P12 = !P12;                             //Test port
        }
        if (S2CON & 0x01)
        {
                S2CON &= ~0x01;                         //Clear interrupt flag
                P13 = !P13;                             //Test port
        }
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        S2CON = 0x10;
        T2L = 0xe8;                                     //65536-11059200/115200/4=0FFE8H
        T2H = 0xff;
        AUXR = 0x14;                                    //Start timer
        IE2 = ES2;                                      //Enable UART2 interrupt
        EA = 1;
        S2BUF = 0x5a;                                   // Send test data

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

T2L             DATA            0D7H
```

```
T2H             DATA            0D6H
AUXR            DATA            8EH
S2CON           DATA            9AH
S2BUF           DATA            9BH
IE2             DATA            0AFH
ES2             EQU             01H

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0043H
                LJMP            UART2ISR

                ORG             0100H
UART2ISR:
                PUSH            ACC
                PUSH            PSW
                MOV             A,S2CON
                JNB             ACC.1,CHECKRI
                ANL             S2CON,#NOT 02H          ;Clear interrupt flag
                CPL             P1.2                    ;Test port
CHECKRI:
                MOV             A,S2CON
                JNB             ACC.0,ISREXIT
                ANL             S2CON,#NOT 01H          ;Clear interrupt flag
                CPL             P1.3                    ;Test port
ISREXIT:
                POP             PSW
                POP             ACC
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             S2CON,#10H
                MOV             T2L,#0E8H               ;65536-11059200/115200/4=0FFE8H
                MOV             T2H,#0FFH
                MOV             AUXR,#14H                       ;Start timer
                MOV             IE2,#ES2                        ;Enable UART2 interrupt
                SETB            EA
                MOV             S2BUF,#5AH                      ; Send test data

                JMP             $

                END
```

11.5.15 UART3 Interrupt

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         T2L         =       0xd7;
sfr         T2H         =       0xd6;
sfr         AUXR        =       0x8e;
sfr         S3CON       =       0xac;
sfr         S3BUF       =       0xad;
sfr         IE2         =       0xaf;
#define     ES3                 0x08

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sbit        P12         =       P1^2;
sbit        P13         =       P1^3;

void UART3_Isr() interrupt 17
{
        if (S3CON & 0x02)
        {
                S3CON &= ~0x02;                         //Clear interrupt flag
                P12 = !P12;                             //Test port
        }
        if (S3CON & 0x01)
        {
                S3CON &= ~0x01;                         //Clear interrupt flag
                P13 = !P13;                             //Test port
        }
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        S3CON = 0x10;
        T2L = 0xe8;                         //65536-11059200/115200/4=0FFE8H
        T2H = 0xff;
        AUXR = 0x14;                        //Start timer
        IE2 = ES3;                          //Enable UART3 interrupt
        EA = 1;
        S3BUF = 0x5a;                       //Send test data

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

T2L         DATA        0D7H
T2H         DATA        0D6H
AUXR        DATA        8EH
S3CON       DATA        0ACH
S3BUF       DATA        0ADH
IE2         DATA        0AFH
ES3         EQU         08H

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         008BH
            LJMP        UART3ISR

            ORG         0100H
UART3ISR:
            PUSH        ACC
            PUSH        PSW
            MOV         A,S3CON
            JNB         ACC.1,CHECKRI
```

```
                ANL         S3CON,#NOT 02H          ;Clear interrupt flag
                CPL         P1.2                    ;Test port
CHECKRI:
                MOV         A,S3CON
                JNB         ACC.0,ISREXIT
                ANL         S3CON,#NOT 01H          ;Clear interrupt flag
                CPL         P1.3                    ;Test port
ISREXIT:
                POP         PSW
                POP         ACC
                RETI

MAIN:
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                MOV         S3CON,#10H
                MOV         T2L,#0E8H               ;65536-11059200/115200/4=0FFE8H
                MOV         T2H,#0FFH
                MOV         AUXR,#14H               ;Start timer
                MOV         IE2,#ES3                ;Enable UART3 interrupt
                SETB        EA
                MOV         S3BUF,#5AH              ;Send test data

                JMP         $

                END
```

11.5.16 UART4 Interrupt

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         T2L         =       0xd7;
sfr         T2H         =       0xd6;
sfr         AUXR        =       0x8e;
sfr         S4CON       =       0x84;
sfr         S4BUF       =       0x85;
sfr         IE2         =       0xaf;
#define     ES4                 0x10

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sbit        P12         =       P1^2;
sbit        P13         =       P1^3;

void UART4_Isr() interrupt 18
{
        if (S4CON & 0x02)
        {
                S4CON &= ~0x02;                                 //Clear interrupt flag
                P12 = !P12;                                     //Test port
        }
        if (S4CON & 0x01)
        {
                S4CON &= ~0x01;                                 //Clear interrupt flag
                P13 = !P13;                                     //Test port
        }
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        S4CON = 0x10;
        T2L = 0xe8;                                             //65536-11059200/115200/4=0FFE8H
        T2H = 0xff;
        AUXR = 0x14;                                            //Start timer
        IE2 = ES4;                                              //Enable UART4 interrupt
        EA = 1;
        S4BUF = 0x5a;                                           //Send test data

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

T2L             DATA            0D7H
T2H             DATA            0D6H
AUXR            DATA            8EH
```

```
S4CON           DATA            84H
S4BUF           DATA            85H
IE2             DATA            0AFH
ES4             EQU             10H

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0093H
                LJMP            UART4ISR

                ORG             0100H
UART4ISR:
                PUSH            ACC
                PUSH            PSW
                MOV             A,S4CON
                JNB             ACC.1,CHECKRI
                ANL             S4CON,#NOT 02H          ;Clear interrupt flag
                CPL             P1.2                    ;Test port
CHECKRI:
                MOV             A,S4CON
                JNB             ACC.0,ISREXIT
                ANL             S4CON,#NOT 01H          ;Clear interrupt flag
                CPL             P1.3                    ;Test port
ISREXIT:
                POP             PSW
                POP             ACC
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             S4CON,#10H
                MOV             T2L,#0E8H               ;65536-11059200/115200/4=0FFE8H
                MOV             T2H,#0FFH
                MOV             AUXR,#14H               ;Start timer
                MOV         IE2,#ES4                ;Enable UART4 interrupt
                SETB        EA
                MOV         S4BUF,#5AH              ;Send test data

                JMP         $

                END
```

11.5.17 ADC Interrupt

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         ADC_CONTR   =       0xbc;
sfr         ADC_RES     =       0xbd;
sfr         ADC_RESL    =       0xbe;
sfr         ADCCFG      =       0xde;
sbit        EADC        =       IE^5;

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

void ADC_Isr() interrupt 5
{
        ADC_CONTR &= ~0x20;                             //Clear interrupt flag
        P0 = ADC_RES;                                   //Test port
        P2 = ADC_RESL;                                  //Test port
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        ADCCFG = 0x00;
```

```
    ADC_CONTR = 0xc0;                           //Enable and start the ADC module
    EADC = 1;                                   //Enable ADC interrupt
    EA = 1;

    while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

ADC_CONTR   DATA        0BCH
ADC_RES     DATA        0BDH
ADC_RESL    DATA        0BEH
ADCCFG      DATA        0DEH
EADC        BIT         IE.5

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         002BH
            LJMP        ADCISR

            ORG         0100H
ADCISR:
            ANL         ADC_CONTR,#NOT 20H      ;Clear interrupt flag
            MOV         P0,ADC_RES              ;Test port
            MOV         P2,ADC_RESL             ;Test port
            RETI

MAIN:
            MOV         SP, #5FH
            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
            MOV         P3M1, #00H
            MOV         P4M0, #00H
            MOV         P4M1, #00H
            MOV         P5M0, #00H
            MOV         P5M1, #00H

            MOV         ADCCFG,#00H
            MOV         ADC_CONTR,#0C0H         ;Enable and start the ADC module
            SETB        EADC                    ;Enable ADC interrupt
```

```
                SETB            EA

                JMP             $

                END
```

11.5.18 LVD Interrupt

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         RSTCFG          =       0xff;
#define     ENLVR                   0x40                    //RSTCFG.6
#define     LVD2V2                  0x00                    //LVD@2.2V
#define     LVD2V4                  0x01                    //LVD@2.4V
#define     LVD2V7                  0x02                    //LVD@2.7V
#define     LVD3V0                  0x03                    //LVD@3.0V
sbit        ELVD            =       IE^6;
#define     LVDF                    0x20                    //PCON.5

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;
sbit        P10             =       P1^0;

void LVD_Isr() interrupt 6
{
        PCON &= ~LVDF;                                      //Clear interrupt flag
        P10 = !P10;                                         //Test port
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;
```

```
    PCON &= ~LVDF;                          //Interrupt flag needs to be cleared at power-on
    RSTCFG = LVD3V0;                        //Set the LVD voltage to 3.0V
    ELVD = 1;                               //Enable LVD interrupt
    EA = 1;

    while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

RSTCFG      DATA        0FFH
ENLVR       EQU         40H                     ;RSTCFG.6
LVD2V2      EQU         00H                     ;LVD@2.2V
LVD2V4      EQU         01H                     ;LVD@2.4V
LVD2V7      EQU         02H                     ;LVD@2.7V
LVD3V0      EQU         03H                     ;LVD@3.0V
ELVD        BIT         IE.6
LVDF        EQU         20H                     ;PCON.5

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         0033H
            LJMP        LVDISR

            ORG         0100H
LVDISR:
            ANL         PCON,#NOT LVDF          ;Clear interrupt flag
            CPL         P1.0                    ;Test port
            RETI

MAIN:
            MOV         SP, #5FH
            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
            MOV         P3M1, #00H
            MOV         P4M0, #00H
            MOV         P4M1, #00H
            MOV         P5M0, #00H
            MOV         P5M1, #00H
```

```
                ANL             PCON,#NOT LVDF                  ;Interrupt flag needs to be cleared at power-on
                MOV             RSTCFG,# LVD3V0                 ;Set the LVD voltage to 3.0V
                SETB            ELVD                            ;Enable LVD interrupt
                SETB            EA
                JMP             $

                END
```

11.5.19 Comparator Interrupt

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         CMPCR1 = 0xe6;
sfr         CMPCR2 = 0xe7;

sfr         P0M1 = 0x93;
sfr         P0M0 = 0x94;
sfr         P1M1 = 0x91;
sfr         P1M0 = 0x92;
sfr         P2M1 = 0x95;
sfr         P2M0 = 0x96;
sfr         P3M1 = 0xb1;
sfr         P3M0 = 0xb2;
sfr         P4M1 = 0xb3;
sfr         P4M0 = 0xb4;
sfr         P5M1 = 0xc9;
sfr         P5M0 = 0xca;

sbit        P10 = P1^0;

void CMP_Isr() interrupt 21
{
            CMPCR1 &= ~0x40;                //clear interrupt flag
            P10 = !P10;                     //test port
}
void main()
{
            P0M0 = 0x00;
            P0M1 = 0x00;
            P1M0 = 0x00;
            P1M1 = 0x00;
            P2M0 = 0x00;
            P2M1 = 0x00;
            P3M0 = 0x00;
            P3M1 = 0x00;
            P4M0 = 0x00;
            P4M1 = 0x00;
            P5M0 = 0x00;
            P5M1 = 0x00;

            CMPCR2 = 0x00;
            CMPCR1 = 0x80;              // Enable comparator module
            CMPCR1 |= 0x30;             // Enable comparator edge interrupt
            CMPCR1 &= ~0x08;            //P3.6  is CMP+ input pin
            CMPCR1 |= 0x04;              //P3.7 is CMP- input pin
            CMPCR1 |= 0x02;             // Enable comparator output
            EA = 1;

            while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

CMPCR1          DATA 0E6H
CMPCR2          DATA 0E7H

P0M1            DATA 093H
P0M0            DATA 094H
P1M1            DATA 091H
P1M0            DATA 092H
P2M1            DATA 095H
P2M0            DATA 096H
P3M1            DATA 0B1H
P3M0            DATA 0B2H
P4M1            DATA 0B3H
P4M0            DATA 0B4H
P5M1            DATA 0C9H
P5M0            DATA 0CAH

            ORG             0000H
            LJMP            MAIN
            ORG             00ABH
            LJMP            CMPISR

            ORG             0100H
CMPISR:
            ANL         CMPCR1,#NOT 40H         ; clear interrupt flag
            CPL         P1.0                    ; test port
            RETI

MAIN:
        MOV         SP, #5FH
            MOV             P0M0, #00H
            MOV             P0M1, #00H
            MOV             P1M0, #00H
            MOV             P1M1, #00H
            MOV             P2M0, #00H
            MOV             P2M1, #00H
            MOV             P3M0, #00H
            MOV             P3M1, #00H
            MOV             P4M0, #00H
            MOV             P4M1, #00H
            MOV             P5M0, #00H
            MOV             P5M1, #00H

            MOV             CMPCR2,#00H
            MOV             CMPCR1,#80H             ; Enable comparator module
            ORL             CMPCR1,#30H             ; Enable comparator edge interrupt
            ANL             CMPCR1,#NOT 08H         ;P3.6  is CMP+ input pin
            ORL             CMPCR1,#04H             ; P3.7 is CMP- input pin
            ORL             CMPCR1,#02H             ; Enable comparator output
            SETB            EA

            JMP              $

            END
```

11.5.20 SPI Interrupt

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
```

```
#include "intrins.h"

sfr         SPSTAT          =       0xcd;
sfr         SPCTL           =       0xce;
sfr         SPDAT           =       0xcf;
sfr         IE2             =       0xaf;
#define     ESPI                    0x02

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

sbit        P10             =       P1^0;

void SPI_Isr() interrupt 9
{
        SPSTAT = 0xc0;                                          //Clear interrupt flag
        P10 = !P10;                                             //Test port
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        SPCTL = 0x50;                                           //Enable SPI master mode
        SPSTAT = 0xc0;                                          //Clear interrupt flag
        IE2 = ESPI;                                             //Enable SPI interrupt
        EA = 1;
        SPDAT = 0x5a;                                           //Send test data

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

SPSTAT          DATA            0CDH
SPCTL           DATA            0CEH
SPDAT           DATA            0CFH
```

```
IE2             DATA            0AFH
ESPI            EQU             02H

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             004BH
                LJMP            SPIISR

                ORG             0100H
SPIISR:
                MOV             SPSTAT,#0C0H                ;Clear interrupt flag
                CPL             P1.0                        ;Test port
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             SPCTL,#50H                  ;Enable SPI master mode
                MOV             SPSTAT,#0C0H                ;Clear interrupt flag
                MOV             IE2,#ESPI                   ;Enable SPI interrupt
                SETB            EA
                MOV             SPDAT,#5AH                  ;Send test data

                JMP             $

                END
```

11.5.21 I2C Interrupt

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
```

```
#include "intrins.h"

sfr         P_SW2           =       0xba;

#define     I2CCFG                  (*(unsigned char volatile xdata *)0xfe80)
#define     I2CMSCR                 (*(unsigned char volatile xdata *)0xfe81)
#define     I2CMSST                 (*(unsigned char volatile xdata *)0xfe82)
#define     I2CSLCR                 (*(unsigned char volatile xdata *)0xfe83)
#define     I2CSLST                 (*(unsigned char volatile xdata *)0xfe84)
#define     I2CSLADR                (*(unsigned char volatile xdata *)0xfe85)
#define     I2CTXD                  (*(unsigned char volatile xdata *)0xfe86)
#define     I2CRXD                  (*(unsigned char volatile xdata *)0xfe87)

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;
sbit        P10             =       P1^0;

void I2C_Isr() interrupt 24
{
        _push_(P_SW2);
        P_SW2 |= 0x80;
        if (I2CMSST & 0x40)
        {
                I2CMSST &= ~0x40;                                   //Clear interrupt flag
                P10 = !P10;                                         //Test port
        }
        _pop_(P_SW2);
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P_SW2 = 0x80;
        I2CCFG = 0xc0;                                              //Enable I2C master mode
        I2CMSCR = 0x80;                                             //Enable I2C interrupt;
        P_SW2 = 0x00;
        EA = 1;
```

```
    P_SW2 = 0x80;
    I2CMSCR = 0x81;                                         //Send start command
    P_SW2 = 0x00;

    while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P_SW2           DATA        0BAH

I2CCFG          XDATA       0FE80H
I2CMSCR         XDATA       0FE81H
I2CMSST         XDATA       0FE82H
I2CSLCR         XDATA       0FE83H
I2CSLST         XDATA       0FE84H
I2CSLADR        XDATA       0FE85H
I2CTXD          XDATA       0FE86H
I2CRXD          XDATA       0FE87H

P0M1            DATA        093H
P0M0            DATA        094H
P1M1            DATA        091H
P1M0            DATA        092H
P2M1            DATA        095H
P2M0            DATA        096H
P3M1            DATA        0B1H
P3M0            DATA        0B2H
P4M1            DATA        0B3H
P4M0            DATA        0B4H
P5M1            DATA        0C9H
P5M0            DATA        0CAH

                ORG         0000H
                LJMP        MAIN
                ORG         00C3H
                LJMP        I2CISR

                ORG         0100H
I2CISR:
                PUSH        ACC
                PUSH        DPL
                PUSH        DPH
                PUSH        P_SW2
                MOV         P_SW2,#80H
                MOV         DPTR,#I2CMSST
                MOVX        A,@DPTR
                ANL         A,#NOT 40H                  ;Clear interrupt flag
                MOVX        @DPTR,A
                CPL         P1.0                        ;Test port
                POP         P_SW2
                POP         DPH
                POP         DPL
                POP         ACC
                RETI

MAIN:
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                MOV         P_SW2,#80H
                MOV         A,#0C0H                     ;Enable I2C master mode
                MOV         DPTR,#I2CCFG
                MOVX        @DPTR,A
                MOV         A,#80H                      ;Enable I2C interrupt
                MOV         DPTR,#I2CMSCR
                MOVX        @DPTR,A
                MOV         P_SW2,#00H
                SETB        EA

                MOV         P_SW2,#80H
                MOV         A,#081H                     ;Send start command
                MOV         DPTR,#I2CMSCR
                MOVX        @DPTR,A
                MOV         P_SW2,#00H

                JMP         $

                END
```

12 I/O port interrupt

Product line	I/O interrupt	I/O interrupt priority	I/O interrupt wake-up function
STC8H1K08 family			
STC8H1K28 family			
STC8H3K64S4 family A version	●	1	
STC8H3K64S4 family A version	●	1	
STC8H3K64S2 family B version	●	1	●
STC8H3K64S2 family B version	●	1	●
STC8H8K64U family A version			
STC8H8K64U family B version	●	4	●
STC8H2K64T family	●	4	●
STC8H4K64TLR family	●	4	●
STC8H4K64TLCD family	●	4	●
STC8H4K64LCD family	●	4	●

The subsequent families of the STC8H series support all I/O interrupts, and support 4 interrupt modes: falling edge interrupt, rising edge interrupt, low-level interrupt, and high-level interrupt. Every group of I/O ports has an independent interrupt entry address, and each I/O can independently set the interrupt mode.

Note: The I/O interrupts of A-version chips of STC8H3K64S4 and STC8H3K64S2 series cannot wake up CPU from power-off, and only have level 1 interrupt priority. The I/O interrupts of B-version chips of STC8H3K64S4 and STC8H3K64S2 series can wake up CPU from power-down, and only have 1 level interrupt priority. For other series with I/O interrupts, I/O interrupts can wake up CPU from power-down, and there are 4 levels of interrupt priority.

12.1 I/O port interrupt related registers

Symbol	Description	Address	Bit Address and Symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
P0INTE	P0 interrupt enable register	FD00H	P07INTE	P06INTE	P05INTE	P04INTE	P03INTE	P02INTE	P01INTE	P00INTE	0000,0000
P1INTE	P1 interrupt enable register	FD01H	P17INTE	P16INTE	P15INTE	P14INTE	P13INTE	P12INTE	P11INTE	P10INTE	0000,0000
P2INTE	P2 interrupt enable register	FD02H	P27INTE	P26INTE	P25INTE	P24INTE	P23INTE	P22INTE	P21INTE	P20INTE	0000,0000
P3INTE	P3 interrupt enable register	FD03H	P37INTE	P36INTE	P35INTE	P34INTE	P33INTE	P32INTE	P31INTE	P30INTE	0000,0000
P4INTE	P4 interrupt enable register	FD04H	P47INTE	P46INTE	P45INTE	P44INTE	P43INTE	P42INTE	P41INTE	P40INTE	0000,0000
P5INTE	P5 interrupt enable register	FD05H	-	-	P55INTE	P54INTE	P53INTE	P52INTE	P51INTE	P50INTE	xx00,0000
P6INTE	P6 interrupt enable register	FD06H	P67INTE	P66INTE	P65INTE	P64INTE	P63INTE	P62INTE	P61INTE	P60INTE	0000,0000
P7INTE	P7 interrupt enable register	FD07H	P77INTE	P76INTE	P75INTE	P74INTE	P73INTE	P72INTE	P71INTE	P70INTE	0000,0000
P0INTF	P0 interrupt flag register	FD10H	P07INTF	P06INTF	P05INTF	P04INTF	P03INTF	P02INTF	P01INTF	P00INTF	0000,0000
P1INTF	P1 interrupt flag register	FD11H	P17INTF	P16INTF	P15INTF	P14INTF	P13INTF	P12INTF	P11INTF	P10INTF	0000,0000
P2INTF	P2 interrupt flag register	FD12H	P27INTF	P26INTF	P25INTF	P24INTF	P23INTF	P22INTF	P21INTF	P20INTF	0000,0000
P3INTF	P3 interrupt flag register	FD13H	P37INTF	P36INTF	P35INTF	P34INTF	P33INTF	P32INTF	P31INTF	P30INTF	0000,0000
P4INTF	P4 interrupt flag register	FD14H	P47INTF	P46INTF	P45INTF	P44INTF	P43INTF	P42INTF	P41INTF	P40INTF	0000,0000
P5INTF	P5 interrupt flag register	FD15H	-	-	P55INTF	P54INTF	P53INTF	P52INTF	P51INTF	P50INTF	xx00,0000
P6INTF	P6 interrupt flag register	FD16H	P67INTF	P66INTF	P65INTF	P64INTF	P63INTF	P62INTF	P61INTF	P60INTF	0000,0000
P7INTF	P7 interrupt flag register	FD17H	P77INTF	P76INTF	P75INTF	P74INTF	P73INTF	P72INTF	P71INTF	P70INTF	0000,0000

P0IM0	P0 Interrupt mode register 0	FD20H	P07IM0	P06IM0	P05IM0	P04IM0	P03IM0	P02IM0	P01IM0	P00IM0	0000,0000
P1IM0	P1 Interrupt mode register 0	FD21H	P17IM0	P16IM0	P15IM0	P14IM0	P13IM0	P12IM0	P11IM0	P10IM0	0000,0000
P2IM0	P2 Interrupt mode register 0	FD22H	P27IM0	P26IM0	P25IM0	P24IM0	P23IM0	P22IM0	P21IM0	P20IM0	0000,0000
P3IM0	P3 Interrupt mode register 0	FD23H	P37IM0	P36IM0	P35IM0	P34IM0	P33IM0	P32IM0	P31IM0	P30IM0	0000,0000
P4IM0	P4 Interrupt mode register 0	FD24H	P47IM0	P46IM0	P45IM0	P44IM0	P43IM0	P42IM0	P41IM0	P40IM0	0000,0000
P5IM0	P5 Interrupt mode register 0	FD25H	-	-	P55IM0	P54IM0	P53IM0	P52IM0	P51IM0	P50IM0	xx00,0000
P6IM0	P6 Interrupt mode register 0	FD26H	P67IM0	P66IM0	P65IM0	P64IM0	P63IM0	P62IM0	P61IM0	P60IM0	0000,0000
P7IM0	P7 Interrupt mode register 0	FD27H	P77IM0	P76IM0	P75IM0	P74IM0	P73IM0	P72IM0	P71IM0	P70IM0	0000,0000
P0IM1	P0 Interrupt mode register 1	FD30H	P07IM1	P06IM1	P05IM1	P04IM1	P03IM1	P02IM1	P01IM1	P00IM1	0000,0000
P1IM1	P1 Interrupt mode register 1	FD31H	P17IM1	P16IM1	P15IM1	P14IM1	P13IM1	P12IM1	P11IM1	P10IM1	0000,0000
P2IM1	P2 Interrupt mode register 1	FD32H	P27IM1	P26IM1	P25IM1	P24IM1	P23IM1	P22IM1	P21IM1	P20IM1	0000,0000
P3IM1	P3 Interrupt mode register 1	FD33H	P37IM1	P36IM1	P35IM1	P34IM1	P33IM1	P32IM1	P31IM1	P30IM1	0000,0000
P4IM1	P4 Interrupt mode register 1	FD34H	P47IM1	P46IM1	P45IM1	P44IM1	P43IM1	P42IM1	P41IM1	P40IM1	0000,0000
P5IM1	P5 Interrupt mode register 1	FD35H	-	-	P55IM1	P54IM1	P53IM1	P52IM1	P51IM1	P50IM1	xx00,0000
P6IM1	P6 Interrupt mode register 1	FD36H	P67IM1	P66IM1	P65IM1	P64IM1	P63IM1	P62IM1	P61IM1	P60IM1	0000,0000
P7IM1	P7 Interrupt mode register 1	FD37H	P77IM1	P76IM1	P75IM1	P74IM1	P73IM1	P72IM1	P71IM1	P70IM1	0000,0000
PINIPL	I/O port interrupt priority low register	FD60H	P7IP	P6IP	P5IP	P4IP	P3IP	P2IP	P1IP	P0IP	0000,0000
PINIPH	I/O port interrupt priority high register	FD61H	P7IPH	P6IPH	P5IPH	P4IPH	P3IPH	P2IPH	P1IPH	P0IPH	0000,0000
P0WKUE	P0 interrupt wake-up enable register	FD40H	P07WKUE	P06WKUE	P05WKUE	P04WKUE	P03WKUE	P02WKUE	P01WKUE	P00WKUE	0000,0000
P1WKUE	P1 interrupt wake-up enable register	FD41H	P17WKUE	P16WKUE	P15WKUE	P14WKUE	P13WKUE	P12WKUE	P11WKUE	P10WKUE	0000,0000
P2WKUE	P2 interrupt wake-up enable register	FD42H	P27WKUE	P26WKUE	P25WKUE	P24WKUE	P23WKUE	P22WKUE	P21WKUE	P20WKUE	0000,0000
P3WKUE	P3 interrupt wake-up enable register	FD43H	P37WKUE	P36WKUE	P35WKUE	P34WKUE	P33WKUE	P32WKUE	P31WKUE	P30WKUE	0000,0000
P4WKUE	P4 interrupt wake-up enable register	FD44H	P47WKUE	P46WKUE	P45WKUE	P44WKUE	P43WKUE	P42WKUE	P41WKUE	P40WKUE	0000,0000
P5WKUE	P5 interrupt wake-up enable register	FD45H	-	-	P55WKUE	P54WKUE	P53WKUE	P52WKUE	P51WKUE	P50WKUE	xx00,0000
P6WKUE	P6 interrupt wake-up enable register	FD46H	P67WKUE	P66WKUE	P65WKUE	P64WKUE	P63WKUE	P62WKUE	P61WKUE	P60WKUE	0000,0000
P7WKUE	P7 interrupt wake-up enable register	FD47H	P77WKUE	P76WKUE	P75WKUE	P74WKUE	P73WKUE	P72WKUE	P71WKUE	P70WKUE	0000,0000

12.1.1 Port interrupt enable registers (PxINTE)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P0INTE	FD00H	P07INTE	P06INTE	P05INTE	P04INTE	P03INTE	P02INTE	P01INTE	P00INTE
P1INTE	FD01H	P17INTE	P16INTE	P15INTE	P14INTE	P13INTE	P12INTE	P11INTE	P10INTE
P2INTE	FD02H	P27INTE	P26INTE	P25INTE	P24INTE	P23INTE	P22INTE	P21INTE	P20INTE
P3INTE	FD03H	P37INTE	P36INTE	P35INTE	P34INTE	P33INTE	P32INTE	P31INTE	P30INTE
P4INTE	FD04H	P47INTE	P46INTE	P45INTE	P44INTE	P43INTE	P42INTE	P41INTE	P40INTE
P5INTE	FD05H	-	-	P55INTE	P54INTE	P53INTE	P52INTE	P51INTE	P50INTE
P6INTE	FD06H	P67INTE	P66INTE	P65INTE	P64INTE	P63INTE	P62INTE	P61INTE	P60INTE
P7INTE	FD07H	P77INTE	P76INTE	P75INTE	P74INTE	P73INTE	P72INTE	P71INTE	P70INTE

PnINTE.x: Port interrupt enable control bit (n=0~7, x=0~7)

0: disable Pn.x port interrupt function

1: Enable Pn.x port interrupt function

12.1.2 Port interrupt flag registers (PxINTF)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P0INTF	FD10H	P07INTF	P06INTF	P05INTF	P04INTF	P03INTF	P02INTF	P01INTF	P00INTF
P1INTF	FD11H	P17INTF	P16INTF	P15INTF	P14INTF	P13INTF	P12INTF	P11INTF	P10INTF
P2INTF	FD12H	P27INTF	P26INTF	P25INTF	P24INTF	P23INTF	P22INTF	P21INTF	P20INTF
P3INTF	FD13H	P37INTF	P36INTF	P35INTF	P34INTF	P33INTF	P32INTF	P31INTF	P30INTF
P4INTF	FD14H	P47INTF	P46INTF	P45INTF	P44INTF	P43INTF	P42INTF	P41INTF	P40INTF
P5INTF	FD15H	-	-	P55INTF	P54INTF	P53INTF	P52INTF	P51INTF	P50INTF
P6INTF	FD16H	P67INTF	P66INTF	P65INTF	P64INTF	P63INTF	P62INTF	P61INTF	P60INTF
P7INTF	FD17H	P77INTF	P76INTF	P75INTF	P74INTF	P73INTF	P72INTF	P71INTF	P70INTF

PnINTF.x: Port interrupt request flag (n=0~7, x=0~7)

0: No interrupt request for Pn.x port

1: Pn.x port has an interrupt request, if the interrupt is enabled, it will enter the interrupt service routine. The flag bit needs to be cleared by software.

12.1.3 Port interrupt mode configuration registers (PxIM0，PxIM1)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P0IM0	FD20H	P07IM0	P06IM0	P05IM0	P04IM0	P03IM0	P02IM0	P01IM0	P00IM0
P0IM1	FD30H	P07IM1	P06IM1	P05IM1	P04IM1	P03IM1	P02IM1	P01IM1	P00IM1
P1IM0	FD21H	P17IM0	P16IM0	P15IM0	P14IM0	P13IM0	P12IM0	P11IM0	P10IM0
P1IM1	FD31H	P17IM1	P16IM1	P15IM1	P14IM1	P13IM1	P12IM1	P11IM1	P10IM1
P2IM0	FD22H	P27IM0	P26IM0	P25IM0	P24IM0	P23IM0	P22IM0	P21IM0	P20IM0
P2IM1	FD32H	P27IM1	P26IM1	P25IM1	P24IM1	P23IM1	P22IM1	P21IM1	P20IM1
P3IM0	FD23H	P37IM0	P36IM0	P35IM0	P34IM0	P33IM0	P32IM0	P31IM0	P30IM0

P3IM1	FD33H	P37IM1	P36IM1	P35IM1	P34IM1	P33IM1	P32IM1	P31IM1	P30IM1
P4IM0	FD24H	P47IM0	P46IM0	P45IM0	P44IM0	P43IM0	P42IM0	P41IM0	P40IM0
P4IM1	FD34H	P47IM1	P46IM1	P45IM1	P44IM1	P43IM1	P42IM1	P41IM1	P40IM1
P5IM0	FD25H	-	-	P55IM0	P54IM0	P53IM0	P52IM0	P51IM0	P50IM0
P5IM1	FD35H	-	-	P55IM1	P54IM1	P53IM1	P52IM1	P51IM1	P50IM1
P6IM0	FD26H	P67IM0	P66IM0	P65IM0	P64IM0	P63IM0	P62IM0	P61IM0	P60IM0
P6IM1	FD36H	P67IM1	P66IM1	P65IM1	P64IM1	P63IM1	P62IM1	P61IM1	P60IM1
P7IM0	FD27H	P77IM0	P76IM0	P75IM0	P74IM0	P73IM0	P72IM0	P71IM0	P70IM0
P7IM1	FD37H	P77IM1	P76IM1	P75IM1	P74IM1	P73IM1	P72IM1	P71IM1	P70IM1

Configure port mode

PnIM1.x	PnIM0.x	Pn.x Interrupt mode
0	0	Falling edge interrupt
0	1	Rising edge interrupt
1	0	Low level interrupt
1	1	High level interrupt

12.1.4 Port interrupt priority control registers (PINIPL, PINIPH)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
PINIPL	FD60H	P7IP	P6IP	P5IP	P4IP	P3IP	P2IP	P1IP	P0IP
PINIPH	FD61H	P7IPH	P6IPH	P5IPH	P4IPH	P3IPH	P2IPH	P1IPH	P0IPH

PxIPH,PxIP: Px interrupt priority control bit

00: Px interrupt priority level is 0 (lowest)
01: Px interrupt priority level is 1 (lower)
10: Px interrupt priority level is 2 (higher)
11: Px interrupt priority level is 3 (highest)

12.1.5 Port interrupt power-down wake-up enable registers (PxWKUE)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P0WKUE	FD40H	P07WKUE	P06WKUE	P05WKUE	P04WKUE	P03WKUE	P02WKUE	P01WKUE	P00WKUE
P1WKUE	FD41H	P17WKUE	P16WKUE	P15WKUE	P14WKUE	P13WKUE	P12WKUE	P11WKUE	P10WKUE
P2WKUE	FD42H	P27WKUE	P26WKUE	P25WKUE	P24WKUE	P23WKUE	P22WKUE	P21WKUE	P20WKUE
P3WKUE	FD43H	P37WKUE	P36WKUE	P35WKUE	P34WKUE	P33WKUE	P32WKUE	P31WKUE	P30WKUE
P4WKUE	FD44H	P47WKUE	P46WKUE	P45WKUE	P44WKUE	P43WKUE	P42WKUE	P41WKUE	P40WKUE
P5WKUE	FD45H	-	-	P55WKUE	P54WKUE	P53WKUE	P52WKUE	P51WKUE	P50WKUE
P6WKUE	FD46H	P67WKUE	P66WKUE	P65WKUE	P64WKUE	P63WKUE	P62WKUE	P61WKUE	P60WKUE
P7WKUE	FD47H	P77WKUE	P76WKUE	P75WKUE	P74WKUE	P73WKUE	P72WKUE	P71WKUE	P70WKUE

PnxWKUE: Port interrupt power-down wake-up enable bit(n=0~7, x=0~7)

0: disable Pn.x interrupt power-down wake-up function
1: enable Pn.x interrupt power-down wake-up function

12.2 Example Routines

12.2.1 P0 Falling edge interrupt

C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P0M0        =       0x94;
sfr         P0M1        =       0x93;
sfr         P1M0        =       0x92;
sfr         P1M1        =       0x91;
sfr         P2M0        =       0x96;
sfr         P2M1        =       0x95;
sfr         P3M0        =       0xb2;
sfr         P3M1        =       0xb1;
sfr         P4M0        =       0xb4;
sfr         P4M1        =       0xb3;
sfr         P5M0        =       0xca;
sfr         P5M1        =       0xc9;
sfr         P6M0        =       0xcc;
sfr         P6M1        =       0xcb;
sfr         P7M0        =       0xe2;
sfr         P7M1        =       0xe1;

sfr         P_SW2       =       0xba;

#define     P0INTE      (*(unsigned char volatile xdata *)0xfd00)
#define     P0INTF      (*(unsigned char volatile xdata *)0xfd10)
#define     P0IM0       (*(unsigned char volatile xdata *)0xfd20)
#define     P0IM1       (*(unsigned char volatile xdata *)0xfd30)

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P_SW2 |= 0x80;
    P0IM0 = 0x00;                               // Falling edge interrupt
    P0IM1 = 0x00;
    P0INTE = 0xff;                              // Enable P0 port interrupt
    P_SW2 &= ~0x80;

    EA = 1;

    while (1);
}
```

```
//Because the interrupt vector is greater than 31, it cannot be directly compiled in KEIL
//The 13th interrupt entry address must be borrowed
void common_isr() interrupt 13
{
        unsigned char psw2_st;
        unsigned char intf;

        psw2_st = P_SW2;
        P_SW2 |= 0x80;
        intf = P0INTF;
        if (intf)
        {
                P0INTF = 0x00;
                if (intf & 0x01)
                {
                                                                //P0.0 interrupt
                }
                if (intf & 0x02)
                {
                                                                //P0.1 interrupt
                }
                if (intf & 0x04)
                {
                                                                //P0.2 interrupt
                }
                if (intf & 0x08)
                {
                                                                //P0.3 interrupt
                }
                if (intf & 0x10)
                {
                                                                //P0.4 interrupt
                }
                if (intf & 0x20)
                {
                                                                //P0.5 interrupt
                }
                if (intf & 0x40)
                {
                                                                //P0.6 interrupt
                }
                if (intf & 0x80)
                {
                                                                //P0.7 interrupt
                }
        }
        P_SW2 = psw2_st;
}

// ISR.ASM
// Save the following code as ISP.ASM, and then add the file to the project

            CSEG        AT 012BH                ;P0 interrupt entry address
            JMP         P0INT_ISR
P0INT_ISR:
            JMP         006BH                   ; Borrow the entry address of the 13th interrupt
            END
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P0M0            DATA            094H
P0M1            DATA            093H
P1M0            DATA            092H
P1M1            DATA            091H
P2M0            DATA            096H
P2M1            DATA            095H
P3M0            DATA            0B2H
P3M1            DATA            0B1H
P4M0            DATA            0B4H
P4M1            DATA            0B3H
P5M0            DATA            0CAH
P5M1            DATA            0C9H
P6M0            DATA            0CCH
P6M1            DATA            0CBH
P7M0            DATA            0E2H
P7M1            DATA            0E1H

P_SW2           DATA            0BAH

P0INTE          XDATA           0FD00H
P0INTF          XDATA           0FD10H
P0IM0           XDATA           0FD20H
P0IM1           XDATA           0FD30H

                ORG             0000H
                LJMP            MAIN

                ORG             012BH                           ;P0 interrupt entry address
P0INT_ISR:
                PUSH            ACC
                PUSH            B
                PUSH            DPL
                PUSH            DPH
                PUSH            P_SW2

                MOV             DPTR,#P0INTF
                MOVX            A,@DPTR
                MOV             B,A
                CLR             A
                MOVX            @DPTR,A
                MOV             A,B
CHECKP00:
                JNB             ACC.0,CHECKP01
                NOP                                             ;P0.0 interrupt
CHECKP01:
                JNB             ACC.1,CHECKP02
                NOP                                             ;P0.1 interrupt
CHECKP02:
                JNB             ACC.2,CHECKP03
                NOP                                             ;P0.2 interrupt
CHECKP03
                JNB             ACC.3,CHECKP04
                NOP                                             ;P0.3 interrupt
CHECKP04:
                JNB             ACC.4,CHECKP05
                NOP                                             ;P0.4 interrupt
CHECKP05:
                JNB             ACC.5,CHECKP06
                NOP                                     ;P0.5 interrupt
CHECKP06:
                JNB             ACC.6,CHECKP07
                NOP                                     ;P0.6 interrupt
CHECKP07:
                JNB             ACC.7,P0ISREXIT
                NOP                                     ;P0.7 interrupt

P0ISREXIT:
                POP             P_SW2
                POP             DPH
                POP             DPL
                POP             B
                POP             ACC
                RETI

                ORG             0200H
MAIN:
                MOV             SP, #5FH

                MOV             P0M0,#00H
                MOV             P0M1,#00H
                MOV             P1M0,#00H
                MOV             P1M1,#00H
                MOV             P2M0,#00H
                MOV             P2M1,#00H
                MOV             P3M0,#00H
                MOV             P3M1,#00H

                ORL             P_SW2,#80H
                CLR             A
                MOV             DPTR,# P0IM0            ; Falling edge interrupt
                MOVX            @DPTR,A
                MOV             DPTR,# P0IM1
                MOVX            @DPTR,A
                MOV             DPTR,# P0INTE
                MOV             A,#0FFH
                MOVX            @DPTR,A                 ; Enable P0 interrupt
                ANL             P_SW2,#7FH

                SETB            EA

                JMP             $

                END
```

12.2.2 P1 rising edge interrupt

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr     P0M0        =       0x94;
sfr     P0M1        =       0x93;
sfr     P1M0        =       0x92;
sfr         P1M1            =       0x91;
sfr         P2M0            =       0x96;
sfr         P2M1            =       0x95;
sfr         P3M0            =       0xb2;
sfr         P3M1            =       0xb1;
sfr         P4M0            =       0xb4;
sfr         P4M1            =       0xb3;
sfr         P5M0            =       0xca;
sfr         P5M1            =       0xc9;
sfr         P6M0            =       0xcc;
sfr         P6M1            =       0xcb;
sfr         P7M0            =       0xe2;
sfr         P7M1            =       0xe1;

sfr         P_SW2           =       0xba;

#define     P1INTE          (*(unsigned char volatile xdata *)0xfd01)
#define     P1INTF          (*(unsigned char volatile xdata *)0xfd11)
#define     P1IM0           (*(unsigned char volatile xdata *)0xfd21)
#define     P1IM1           (*(unsigned char volatile xdata *)0xfd31)

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P_SW2 |= 0x80;
    P1IM0 = 0xff;                                   // Rising edge interrupt
    P1IM1 = 0x00;
    P1INTE = 0xff;                                  // Enable P1 interrupt
    P_SW2 &= ~0x80;

    EA = 1;

    while (1);
}

//Because the interrupt vector is greater than 31, it cannot be directly compiled in KEIL
//The 13th interrupt entry address must be borrowed
void common_isr() interrupt 13
{
    unsigned char psw2_st;
    unsigned char intf;

    psw2_st = P_SW2;
    P_SW2 |= 0x80;
    intf = P1INTF;
    if (intf)
    {
        P1INTF = 0x00;
```

```
                if (intf & 0x01)
                {
                                                        //P1.0 interrupt
                }
                if (intf & 0x02)
                {
                                                        //P1.1 interrupt
                }
                if (intf & 0x04)
                {
                                                        //P1.2 interrupt
                }
                if (intf & 0x08)
                {
                                                        //P1.3 interrupt
                }
                if (intf & 0x10)
                {
                                                        //P1.4 interrupt
                }
                if (intf & 0x20)
                {
                                                        //P1.5 interrupt
                }
                if (intf & 0x40)
                {
                                                        //P1.6 interrupt
                }
                if (intf & 0x80)
                {
                                                        //P1.7 interrupt
                }
        }
        P_SW2 = psw2_st;
}

// ISR.ASM
// Save the following code as ISP.ASM, and then add the file to the project

                CSEG        AT 0133H                    ;P1 interrupt entry address
                JMP         P1INT_ISR
P1INT_ISR:
                JMP         006BH                       ; Borrow the entry address of the 13th interrupt
                END
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P0M0            DATA        094H
P0M1            DATA        093H
P1M0            DATA        092H
P1M1            DATA        091H
P2M0            DATA        096H
P2M1            DATA        095H
P3M0            DATA        0B2H
P3M1            DATA        0B1H
P4M0            DATA        0B4H
P4M1            DATA        0B3H
```

```
P5M0            DATA            0CAH
P5M1            DATA            0C9H
P6M0            DATA            0CCH
P6M1            DATA            0CBH
P7M0            DATA            0E2H
P7M1            DATA            0E1H

P_SW2           DATA            0BAH

P1INTE          XDATA           0FD01H
P1INTF          XDATA           0FD11H
P1IM0           XDATA           0FD21H
P1IM1           XDATA           0FD31H

                ORG             0000H
                LJMP            MAIN

                ORG             0133H                           ;P1 interrupt entry address
P1INT_ISR:
                PUSH            ACC
                PUSH            B
                PUSH            DPL
                PUSH            DPH
                PUSH            P_SW2

                MOV             DPTR,#P1INTF
                MOVX            A,@DPTR
                MOV             B,A
                CLR             A
                MOVX            @DPTR,A
                MOV             A,B
CHECKP10:
                JNB             ACC.0,CHECKP11
                NOP                                             ;P1.0 interrupt
CHECKP11:
                JNB             ACC.1,CHECKP12
                NOP                                             ;P1.1 interrupt
CHECKP12:
                JNB             ACC.2,CHECKP13
                NOP                                             ;P1.2 interrupt
CHECKP13
                JNB             ACC.3,CHECKP14
                NOP                                             ;P1.3 interrupt
CHECKP14:
                JNB             ACC.4,CHECKP15
                NOP                                             ;P1.4 interrupt
CHECKP15:
                JNB             ACC.5,CHECKP16
                NOP                                             ;P1.5 interrupt
CHECKP16:
                JNB             ACC.6,CHECKP17
                NOP                                             ;P1.6 interrupt
CHECKP17:
                JNB             ACC.7,P1ISREXIT
                NOP                                             ;P1.7 interrupt

P1ISREXIT:
                POP             P_SW2
                POP             DPH
                POP             DPL
                POP             B
                POP             ACC
                RETI

                ORG             0200H
MAIN:
                MOV             SP, #5FH

                MOV             P0M0,#00H
                MOV             P0M1,#00H
                MOV             P1M0,#00H
                MOV             P1M1,#00H
                MOV             P2M0,#00H
                MOV             P2M1,#00H
                MOV             P3M0,#00H
                MOV             P3M1,#00H

                ORL             P_SW2,#80H
                CLR             A
                MOV             DPTR,# P1IM0                ; Falling edge interrupt
                MOVX            @DPTR,A
                MOV             DPTR,# P1IM1
                MOVX            @DPTR,A
                MOV             DPTR,# P1INTE
                MOV             A,#0FFH
                MOVX            @DPTR,A                     ; Enable P1 interrupt
                ANL             P_SW2,#7FH

                SETB            EA

                JMP             $

                END
```

12.2.3 P2 low level interrupt

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"


sfr         P0M0        =       0x94;
sfr         P0M1        =       0x93;
sfr         P1M0        =       0x92;
sfr         P1M1        =       0x91;
sfr         P2M0        =       0x96;
sfr         P2M1        =       0x95;
sfr         P3M0        =       0xb2;
sfr         P3M1        =       0xb1;
sfr         P4M0        =       0xb4;
sfr         P4M1        =       0xb3;
sfr         P5M0        =       0xca;
sfr         P5M1        =       0xc9;
sfr         P6M0        =       0xcc;
sfr         P6M1        =       0xcb;
sfr         P7M0        =       0xe2;
sfr         P7M1            =       0xe1;

sfr         P_SW2           =       0xba;

#define     P2INTE          (*(unsigned char volatile xdata *)0xfd02)
#define     P2INTF          (*(unsigned char volatile xdata *)0xfd12)
#define     P2IM0           (*(unsigned char volatile xdata *)0xfd22)
#define     P2IM1           (*(unsigned char volatile xdata *)0xfd32)

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P_SW2 |= 0x80;
        P2IM0 = 0x00;                                   // low level interrupt
        P2IM1 = 0xff;
        P2INTE = 0xff;                                  // Enable P2 port interrupt
        P_SW2 &= ~0x80;

        EA = 1;

        while (1);
}

//Because the interrupt vector is greater than 31, it cannot be directly compiled in KEIL
//The 13th interrupt entry address must be borrowed
void common_isr() interrupt 13
{
        unsigned char psw2_st;
        unsigned char intf;

        psw2_st = P_SW2;
        P_SW2 |= 0x80;
        intf = P2INTF;
        if (intf)
        {
                P2INTF = 0x00;
                if (intf & 0x01)
                {
                                                        //P2.0 interrupt
                }
                if (intf & 0x02)
                {
                                                        //P2.1 interrupt
                }
                if (intf & 0x04)
                {
                                                        //P2.2 interrupt
                }
                if (intf & 0x08)
                {
                                                        //P0.3 interrupt
                }
                if (intf & 0x10)
                {
                                                        //P2.4 interrupt
                }
                if (intf & 0x20)
                {
                                                        //P2.5 interrupt
                }
                if (intf & 0x40)
                {
                                                        //P2.6 interrupt
                }
                if (intf & 0x80)
                {
                                                        //P2.7 interrupt
                }
        }
        P_SW2 = psw2_st;
}

// ISR.ASM
// Save the following code as ISP.ASM, and then add the file to the project

            CSEG        AT 013BH                ;P2 interrupt entry address
            JMP         P2INT_ISR
P2INT_ISR:
            JMP         006BH                   ; Borrow the entry address of the 13th interrupt
            END
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P0M0        DATA        094H
P0M1        DATA        093H
P1M0        DATA        092H
P1M1        DATA        091H
P2M0        DATA        096H
P2M1        DATA        095H
P3M0        DATA        0B2H
P3M1        DATA        0B1H
P4M0        DATA        0B4H
P4M1        DATA        0B3H
P5M0        DATA        0CAH
P5M1        DATA        0C9H
P6M0        DATA        0CCH
P6M1        DATA        0CBH
P7M0        DATA        0E2H
P7M1        DATA        0E1H

P_SW2       DATA        0BAH
P2INTE      XDATA       0FD02H
P2INTF      XDATA       0FD12H
P2IM0       XDATA       0FD22H
P2IM1       XDATA       0FD32H

            ORG         0000H
            LJMP        MAIN
```

```
                ORG             013BH                   ;P2  interrupt entry address
P2INT_ISR:
                PUSH            ACC
                PUSH            B
                PUSH             DPL
                PUSH             DPH
                PUSH             P_SW2
                MOV              DPTR,#P2INTF
                MOVX            A,@DPTR
                MOV B,A
                CLR             A
                MOVX             @DPTR,A
                MOV             A,B
CHECKP20:
                JNB             ACC.0,CHECKP21
                NOP                                          ;P2.0 interrupt
CHECKP21:
                JNB             ACC.1,CHECKP22
                NOP                                          ;P2.1 interrupt
CHECKP22:
                JNB             ACC.2,CHECKP23
                NOP                                          ;P2.2 interrupt
CHECKP23
                JNB             ACC.3,CHECKP24
                NOP                                           ;P2.3 interrupt
CHECKP24:
                JNB             ACC.4,CHECKP25
                NOP                                          ;P2.4 interrupt
CHECKP25:
                JNB             ACC.5,CHECKP26
                NOP                                          ;P2.5 interrupt
CHECKP26:
                JNB             ACC.6,CHECKP27
                NOP                                          ;P2.6 interrupt
CHECKP27:
                JNB             ACC.7,P2ISREXIT
                NOP                                          ;P2.7  interrupt
P2ISREXIT:
                POP             P_SW2
                POP             DPH
                POP             DPL
                POP             B
                POP             ACC
                RETI

                ORG             0200H
MAIN:
                MOV             SP, #5FH

                MOV             P0M0,#00H
                MOV             P0M1,#00H
                MOV             P1M0,#00H
                MOV             P1M1,#00H
                MOV             P2M0,#00H
                MOV             P2M1,#00H
                MOV             P3M0,#00H
                MOV             P3M1,#00H

                ORL             P_SW2,#80H
                CLR             A
                MOV             DPTR,# P2IM0            ; low level interrupt
                MOVX            @DPTR,A
                MOV             DPTR,# P2IM1
                MOVX            @DPTR,A
                MOV             DPTR,# P2INTE
                MOV             A,#0FFH
                MOVX            @DPTR,A                 ;enable P2  interrupt
                ANL             P_SW2,#7FH

                SETB            EA
```

```
        JMP         $

        END
```

12.2.4 Port3 high level interrupt

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P0M0        =    0x94;
sfr         P0M1        =    0x93;
sfr         P1M0        =    0x92;
sfr         P1M1        =    0x91;
sfr         P2M0        =    0x96;
sfr         P2M1        =    0x95;
sfr         P3M0        =    0xb2;
sfr         P3M1        =    0xb1;
sfr         P4M0        =    0xb4;
sfr         P4M1        =    0xb3;
sfr         P5M0        =    0xca;
sfr         P5M1        =    0xc9;
sfr         P6M0        =    0xcc;
sfr         P6M1        =    0xcb;
sfr         P7M0        =    0xe2;
sfr         P7M1        =    0xe1;
sfr         P_SW2       =    0xba;

#define     P3INTE      (*(unsigned char volatile xdata *)0xfd03)
#define     P3INTF      (*(unsigned char volatile xdata *)0xfd13)
#define     P3IM0       (*(unsigned char volatile xdata *)0xfd23)
#define     P3IM1       (*(unsigned char volatile xdata *)0xfd33)

void main()
{
            P0M0 = 0x00;
            P0M1 = 0x00;
            P1M0 = 0x00;
            P1M1 = 0x00;
            P2M0 = 0x00;
            P2M1 = 0x00;
            P3M0 = 0x00;
            P3M1 = 0x00;
            P4M0 = 0x00;
            P4M1 = 0x00;
```

```
        P5M0 = 0x00;
        P5M1 = 0x00;

        P_SW2 |= 0x80;
        P3IM0 = 0xff;                               // high level interrupt
        P3IM1 = 0xff;
        P3INTE = 0xff;                              //Enable P3 interrupt
        P_SW2 &= ~0x80;

        EA = 1;
        while (1);
}
//Because the interrupt vector is greater than 31, it cannot be directly compiled in KEIL
//The 13th interrupt entry address must be borrowed
void common_isr() interrupt 13
{
    unsigned char psw2_st;
    unsigned char intf;

    psw2_st = P_SW2;
    P_SW2 |= 0x80;
    intf = P3INTF;
    if (intf)
    {
        P3INTF = 0x00;
        if (intf & 0x01)
        {
                                                    //P3.0 interrupt
        }
        if (intf & 0x02)
        {
                                                    //P3.1 interrupt
        }
        if (intf & 0x04)
        {
                                                    //P3.2 interrupt
        }
        if (intf & 0x08)
        {
                                                    //P3.3 interrupt
        }
```

```
            if (intf & 0x10)
            {
                                                                //P3.4 interrupt
            }
            if (intf & 0x20)
            {
                                                                //P3.5 interrupt
            }
            if (intf & 0x40)
            {
                                                                //P3.6 interrupt
            }
            if (intf & 0x80)
            {
                                                                //P3.7 interrupt
            }
        }
        P_SW2 = psw2_st;
}
// ISR.ASM
//Save the following code as ISP.ASM, and then add the file to the project
                CSEG        AT 0143H            ; P3 port interrupt entry address
                JMP     P3INT_ISR
P3INT_ISR:
                JMP     006BH                   ; Borrow the entry address of the 13th interrupt
                END
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P0M0            DATA        094H
P0M1            DATA        093H
P1M0            DATA        092H
P1M1            DATA        091H
P2M0            DATA        096H
P2M1            DATA        095H
P3M0            DATA        0B2H
P3M1            DATA        0B1H
P4M0            DATA        0B4H
P4M1            DATA        0B3H
P5M0            DATA        0CAH
P5M1            DATA        0C9H
P6M0            DATA        0CCH
```

```
P6M1            DATA            0CBH
P7M0            DATA            0E2H
P7M1            DATA            0E1H

P_SW2           DATA            0BAH
P3INTE          XDATA           0FD03H
P3INTF          XDATA           0FD13H
P3IM0           XDATA           0FD23H
P3IM1           XDATA           0FD33H

                ORG 0000H
                LJMP MAIN

                ORG 0143H                   ;P3  interrupt entry address
P3INT_ISR:
                PUSH            ACC
                PUSH            B
                PUSH            DPL
                PUSH            DPH
                PUSH             P_SW2

                MOV             DPTR,#P3INTF
                MOVX            A,@DPTR
                MOV             B,A
                CLR             A
                MOVX            @DPTR,A
                MOV             A,B
CHECKP30:
                JNB             ACC.0,CHECKP31
                NOP                                         ;P3.0  interrupt
CHECKP31:
                JNB             ACC.1,CHECKP32
                NOP                                         ;P3.1  interrupt
CHECKP32:
                JNB             ACC.2,CHECKP33
                NOP                                         ;P3.2  interrupt
CHECKP33
                JNB             ACC.3,CHECKP34
                NOP                                         ;P3.3  interrupt
CHECKP34:
                JNB             ACC.4,CHECKP35
                NOP                                         ;P3.4  interrupt
CHECKP35:
                JNB             ACC.5,CHECKP36
                NOP                                     ;P3.5  interrupt
CHECKP36:
                JNB             ACC.6,CHECKP37
                NOP                                     ;P3.6  interrupt
CHECKP37:
                JNB             ACC.7,P3ISREXIT
                NOP                                     ;P3.7  interrupt
P3ISREXIT:
                POP             P_SW2
                POP             DPH
                POP             DPL
                POP             B
                POP             ACC
                RETI

                ORG             0200H
MAIN:
                MOV             SP, #5FH

                MOV             P0M0,#00H
                MOV             P0M1,#00H
                MOV             P1M0,#00H
                MOV             P1M1,#00H
                MOV             P2M0,#00H
                MOV             P2M1,#00H
                MOV             P3M0,#00H
                MOV             P3M1,#00H

                ORL             P_SW2,#80H
                CLR A
                MOV             DPTR,# P3IM0                        ; high level interrupt
                MOVX            @DPTR,A
                MOV             DPTR,# P3IM1
                MOVX            @DPTR,A
                MOV             DPTR,# P3INTE
                MOV             A,#0FFH
                MOVX            @DPTR,A                             ;enable P3  interrupt
                ANL             P_SW2,#7FH
                SETB            EA
                JMP             $

                END
```

13 Timer/Counter

Product line	Number of timers
STC8H1K08 family	**3**
STC8H1K28 family	**5**
STC8H3K64S4 family	**5**
STC8H3K64S2 family	**5**
STC8H8K64U family	**5**
STC8H2K64T family	**5**
STC8H4K64TLR family	**5**
STC8H4K64TLCD family	**5**
STC8H4K64LCD family	**5**

Five 16-bit Timers/Counters are integrated in STC8H series of microcontrollers: T0, T1, T2, T3 and T4. All of them can be used as timer or counter. For T0 and T1, the 'timer' or 'counter' function is selected by the control bits C/T in the special function register TMOD. For T2, the 'timer' or 'counter' function is selected by the control bit T2_C/T in the special function register AUXR. For T3, the 'timer' or 'counter' function is selected by the control bit T3_C/T in the special function register T4T3M. For T4, the 'timer' or 'counter' function is selected by the control bit T4_C/T in the special function register T4T3M. The core of the timer/counter is a up counter, the essence of which is counting pulses. The only difference of 'timer' mode and 'counter' mode is the different counting pulses sources. If the counting pulse is from the system clock, the timer/counter runs in the timing mode, it counts once every 12 clocks or one clock. If the counting pulse is from the microcontroller external pins, the timer/counter runs in counting mode, it counts once every pulse.

When T0, T1 and T2 are operating in 'timer' mode, T0x12, T1x12 and T2x12 in AUXR register are used to determine the clocks of T0, T1 and T2 are system clock/12 or system clock/1. When T3 and T4 are operating in 'timer' mode, T3x12 and T4x12 in the T4T3M register determine the clocks of T3 and T4 are system clock/12 or system clock/1. When the timer/counters are operating in 'counter' mode, the frequency of the external pulse is not divided.

T0 has four operating modes which are selected by bit-pairs (M1, M0) in TMOD. The four modes are mode 0 (16-bit auto-reload mode), mode 1 (16-bit non-auto-reload mode), mode 2 (8-bit auto-reload mode) and mode 3 (16-bit auto-reload mode whose interrupt can not be disabled). And for T1, all modes except mode 3 are the same as T0. The mode 3 of T1 is invalid and stops counting. For T2, T3 and T4, they only have one mode: 16-bit auto-reload mode. Besides being used as timer/counters, T2, T3 and T4 can also be as the baud-rate generators of UARTs and programmable clock outputs.

13.1 Registers Related to Timers

Symbol	Description	Address	Bit Address and Symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
TCON	Timer 0 and 1 control register	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0000,0000
TMOD	Timer 0 and 1 mode register	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	0000,0000
TL0	Timer 0 low byte register	8AH									0000,0000
TL1	Timer 1 low byte register	8BH									0000,0000
TH0	Timer 0 high byte register	8CH									0000,0000
TH1	Timer 1 high byte register	8DH									0000,0000
AUXR	Auxiliary register 1	8EH	T0x12	T1x12	UART_M0x6	T2R	T2_C/T	T2x12	EXTRAM	S1ST2	0000,0001
INTCLKO	External interrupt and clock output control register	8FH	-	EX4	EX3	EX2	-	T2CLKO	T1CLKO	T0CLKO	x000,x000
WKTCL	Wake-up Timer Control Register low	AAH									1111,1111
WKTCH	Wake-up Timer Control Register high byte	ABH	WKTEN								0111,1111
T4T3M	Timer4 and Timer 3 Control Register	D1H	T4R	T4_C/T	T4x12	T4CLKO	T3R	T3_C/T	T3x12	T3CLKO	0000,0000
T4H	Timer 4 high byte register	D2H									0000,0000
T4L	Timer 4 low byte register	D3H									0000,0000
T3H	Timer 3 high byte register	D4H									0000,0000
T3L	Timer 3 low byte register	D5H									0000,0000

T2H	Timer 2 high byte register	D6H		0000,0000
T2L	Timer 2 low byte register	D7H		0000,0000

Symbol	Description	Address	Bit Address and Symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
TM2PS	Timer 2 clock prescaler register	FEA2H									0000,0000
TM3PS	Timer 3 clock prescaler register	FEA3H									0000,0000
TM4PS	Timer 4 clock prescaler register	FEA4H									0000,0000

13.2 Timer 0/1

13.2.1 Timer 0 and 1 Control Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
TCON	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1: T1 overflow flag. After T1 is enabled to count, it performs adding 1 count from the initial value. TF1 is set by hardware on T1 overflow and requests interrupt to CPU. It will keep the status until CPU responds the interrupt and is cleared by hardware automatically. It also can be cleared by software.

TR1: T1 run control bit. It is set or cleared by software to turn the timer on/off. If GATE (TMOD.7) = 0, T1 will start counting as soon as TR1=1 and stop counting when TR1=0. If GATE (TMOD.7) = 1, T1 is enabled to count only if TR1 = 1 and INT1 is high.

TF0: T0 overflow flag. After T0 is enabled to count, it performs adding 1 count from the initial value. TF0 is set by hardware on T0 overflow and requests interrupt to CPU. It will keep the status until CPU responds the interrupt and is cleared by hardware automatically. It also can be cleared by software.

TR0: T0 run control bit. It is set or cleared by software to turn the timer on/off. If GATE (TMOD.3) = 0, T0 will start counting as soon as TR0=1 and stop counting when TR0=0. If GATE (TMOD.0) = 1, T0 is enabled to count only if TR0 = 1 and INT0 is high.

IE1: External Interrupt 1 (INT1/P3.3) request flag. IE1=1 means external interrupt requests interrupt to CPU.It is cleared by hardware automatically when the CPU responds to the interrupt.

IT1: External Intenupt 1 trigger edge type control bit. If IT1 = 0, INT1 can be triggered by both rising and falling edges. If IT1 = 1, INT1 can be triggered only by falling edge.

IE0: External Interrupt 0 (INT0/P3.2) request flag. IE0=1 means external interrupt requests interrupt to CPU.It is cleared by hardware automatically when the CPU responds to the interrupt.

IT0: External Intenupt 0 trigger edge type control bit. If IT0 = 0, INT0 can be triggered by both rising and falling edges. If IT0 = 1, INT0 can be triggered only by falling edge.

13.2.2 Timer 0/1 Mode Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
TMOD	89H	T1_GATE	T1_C/T	T1_M1	T1_M0	T0_GATE	T0_C/T	T0_M1	T0_M0

T1_GATE: T1 gate control. If GATE/TMOD.7=1, T1 starts only when TR1 is set AND INT1 pin is high.

T0_GATE: T0 gate control. If GATE/TMOD.3 =1, T0 starts only when TR0 is set AND INT0 pin is high.

T1_C/T: T1 mode select bit. If it is reset, T1 is used as a timer (input pulse is from internal system clock). If it is set, T1 is used as a counter (input pulse is from external T1/P3.5 pin).

T0_C/T: T0 mode select bit. If it is reset, T0 is used as a timer (input pulse is from internal system clock). If it is set, T0 is used as a counter (input pulse is from external T0/P3.4 pin).

T1_M1/T1_M0: T1 mode select bits.

T1_M1	T1_M0	T1 operating mode
0	0	16-bit auto-reload mode.

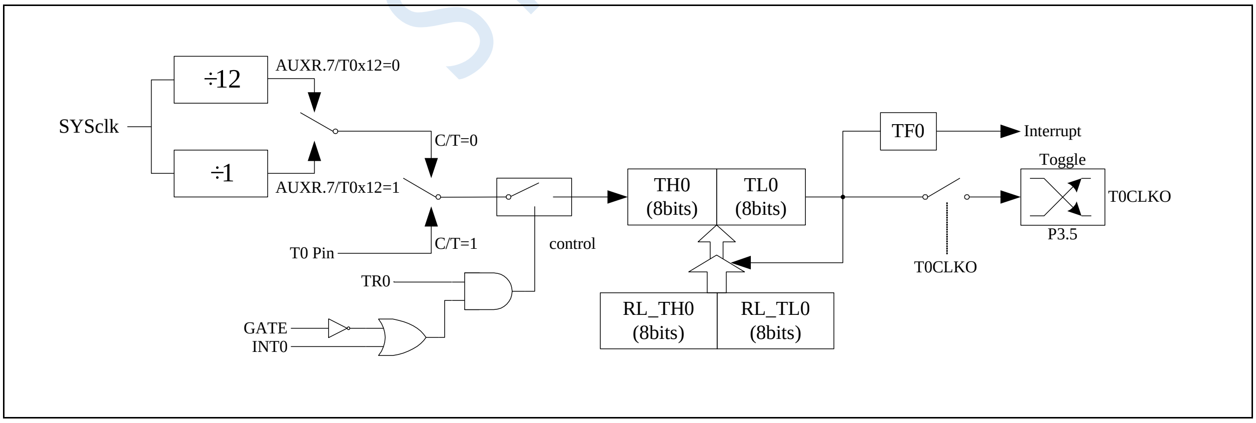
		When the 16-bit counter [TH1, TL1] overflows, the system loads the reload value in the internal 16-bit reload register into [TH1, TL1] automatically.
0	1	16-bit non-auto-reload mode. When the 16-bit counter [TH1, TL1] overflows, timer 1 will count from 0.
1	0	8-bit auto-reload mode. When the 8-bit counter TL1 overflows, the system loads the reload value in TH1 into TL1 automatically.
1	1	T1 stops working.

T0_M1/T0_M0: T0 mode select bits.

T0_M1	T0_M0	T0 operating mode
0	0	16-bit auto-reload mode. When the 16-bit counter [TH0, TL0] overflows, the system loads the reload value in the internal 16-bit reload register into [TH0, TL0] automatically.
0	1	16-bit non-auto-reload mode. When the 16-bit counter [TH1, TL1] overflows, timer 1 will count from 0.
1	0	8-bit auto-reload mode. When the 8-bit counter TL0 overflows, the system loads the reload value in TH0 into TL0 automatically.
1	1	16-bit auto-reload mode. It is similar to mode 0, whose interrupt can not be disabled. The interrupt has the highest priority, higher than the priority of all other interrupts, and cannot be turned off. It can be used as the system tick timer of the operating system or the system monitoring timer. The only way to stop is to turn off the TR0 bit in the TCON register and stop supplying the clock to Timer 0.

13.2.3 Timer0 mode 0 (16-bit auto-reloadable mode)

In this mode, Timer/Counter 0 is used as a 16-bit counter that can be automatically reloaded, as shown in the figure below.



Timer/Counter0 mode 0: 16-bit auto-reload mode

When GATE=0 (TMOD.3), the timer will count if TR0=1. When GATE=1, it is allowed to control timer0 by external input INT0, so that pulse width measurement can be realized. TR0 is the control bit in the TCON register. For the specific function description of each bit of the TCON register, see the introduction of the TCON register in the previous section.

When C/T=0, the multiplexer is connected to the frequency division output of the system clock. T0 counts the internal system clock, and works in timing mode. When C/T=1, the multiplexer is connected to the external pulse input P3.4/T0, and T0 works in counting mode.

Timer0 of STC microcontroller has two counting rates: one is 12T mode, which is increased by 1 for every 12 clocks, which is the same as traditional 8051 microcontroller, the other is 1T mode, which is increased by 1 for each clock, and the speed is 12 times of traditional 8051. The rate of T0 is determined by T0x12 in the special function register AUXR. If T0x12=0, T0 works in 12T mode, and if T0x12=1, T0 works in 1T mode.

Timer0 has two hidden registers RL_TH0 and RL_TL0. RL_TH0 and TH0 share the same address, and RL_TL0 and TL0 share the same address. When TR0=0, that is, when Timer/Counter0 is disabled, the content written to TL0 will be written to RL_TL0 at the same time, and the content written to TH0 will also be written to RL_TH0 at the same time. When TR0=1, that is, when Timer/Counter0 is allowed to work, writing content to TL0 is not actually written to the current register TL0, but written to the hidden register RL_TL0, and writing content to TH0 is actually also it is not written into the current register TH0, but into the hidden register RL_TH0, which can cleverly realize the 16-bit reload timer. When reading the contents of TH0 and TL0, the contents be read are the contents of TH0 and TL0, not the contents of RL_TH0 and RL_TL0.

When Timer0 is working in mode 0 (TMOD[1:0]/[M1,M0]=00B), the overflow of [TH0,TL0] not only sets TF0, but also automatically reloads the contents of [RL_TH0,RL_TL0] to [TH0,TL0].

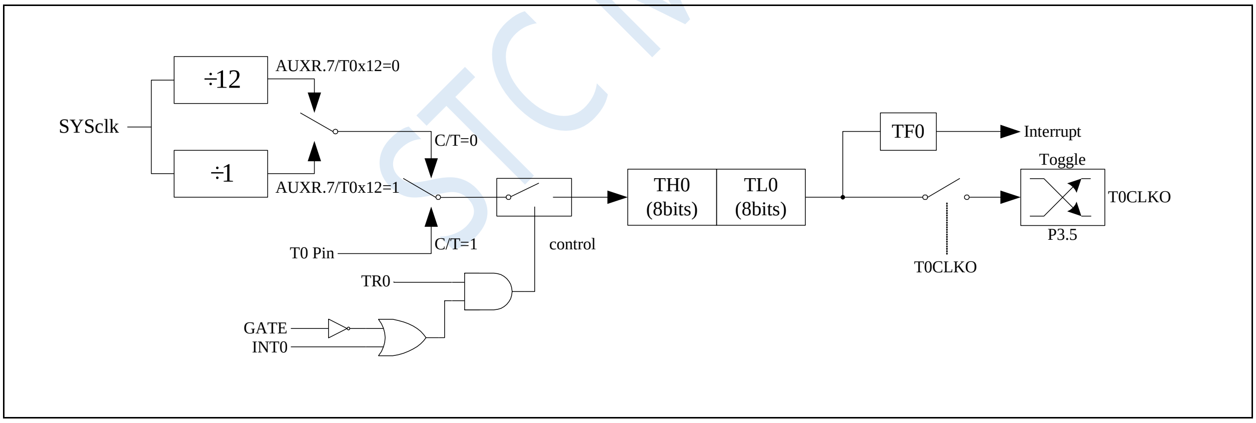
If T0CLKO/INT_CLKO.0=1, the P3.5/T1 pin is configured as timer 0's clock output T0CLKO. The output clock frequency is T0 overflow rate/2.

If C/T=0, the timer/counter 0 counts the internal system clock, then:
if T0 works in 1T mode (AUXR.7/T0x12=1), the output clock frequency = (SYSclk)/(65536-[RL_TH0, RL_TL0])/2
if T0 works in 12T mode (AUXR.7/T0x12=0), the output clock frequency = (SYSclk)/12/(65536-[RL_TH0, RL_TL0])/2

If C/T=1, the timer/counter 0 counts the external pulse input (P3.4/T0), then:
the output clock frequency = (T0_Pin_CLK) / (65536-[RL_TH0, RL_TL0])/2

13.2.4 Timer0 mode 1 (16-bit non-autoreloadable mode)

In this mode, Timer/Counter 0 works in 16-bit non-reloadable mode, as shown in the figure below.



Timer/counter 0 mode 1: 16-bit non-reloadable mode

In this mode, Timer/Counter 0 is configured as a 16-bit non-reloadable mode, which is composed of 8 bits of TL0 and 8 bits of TH0. The 8-bit overflow of TL0 carries over to TH0, and the overflow of TH0 counts the overflow flag TF0 in TCON.

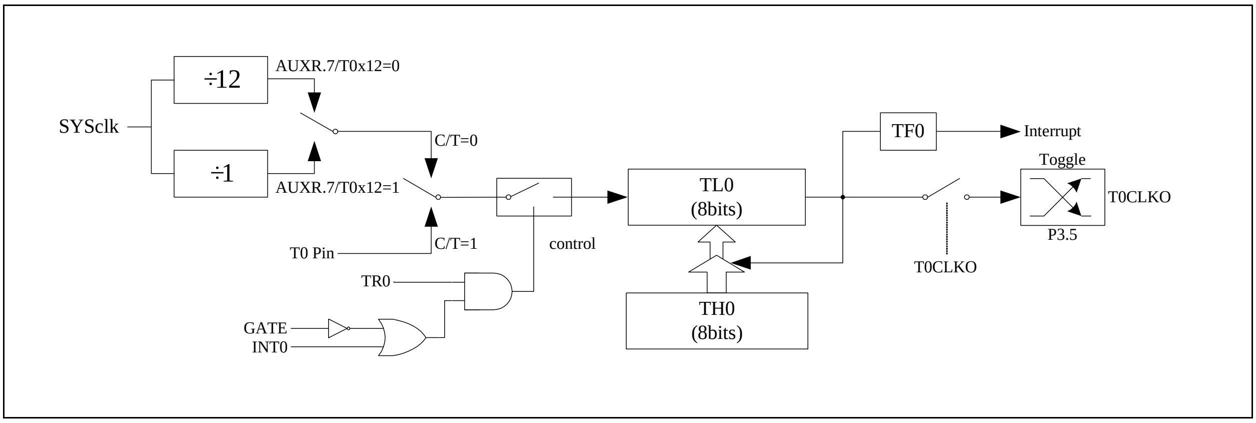
When GATE=0 (TMOD.3), the timer will count if TR0=1. When GATE=1, it is allowed to control timer 0 by external input INT0, so that pulse width measurement can be realized. TR0 is the control bit in the TCON register. For the specific function description of each bit of the TCON register, see the introduction of the TCON register in the previous section.

When C/T=0, the multiplexer is connected to the frequency division output of the system clock, T0 counts the internal system clock, and works in timing mode. When C/T=1, the multiplexer is connected to the external pulse input P3.4/T0, and T0 works in counting mode.

Timer0 of STC microcontroller has two counting rates: one is 12T mode, which is increased by 1 for every 12 clocks, which is the same as traditional 8051 microcontroller, the other is 1T mode, which is increased by 1 for each clock, and the speed is 12 times of traditional 8051. The rate of T0 is determined by T0x12 in the special function register AUXR. If T0x12=0, T0 works in 12T mode, and if T0x12=1, T0 works in 1T mode.

13.2.5 Timer 0 mode 2 (8-bit auto-reloadable mode)

In this mode, Timer/Counter 0 is an 8-bit counter that can be automatically reloaded, as shown in the figure below.



Timer/counter 0 mode 2: 8-bit auto-reloadable mode

The overflow of TL0 not only sets TF0, but also reloads the content of TH0 into TL0. The content of TH0 is preset by software, and its content remains unchanged during reloading.

When T0CLKO/INT_CLKO.0=1, the P3.5/T1 pin is configured as timer 0's clock output T0CLKO. The output clock frequency is T0 overflow rate/2.

If C/T=0, the timer/counter 0 counts the internal system clock, then:

if T0 works in 1T mode (AUXR.7/T0x12=1), the output clock frequency = (SYSclk)/(256-TH0)/2

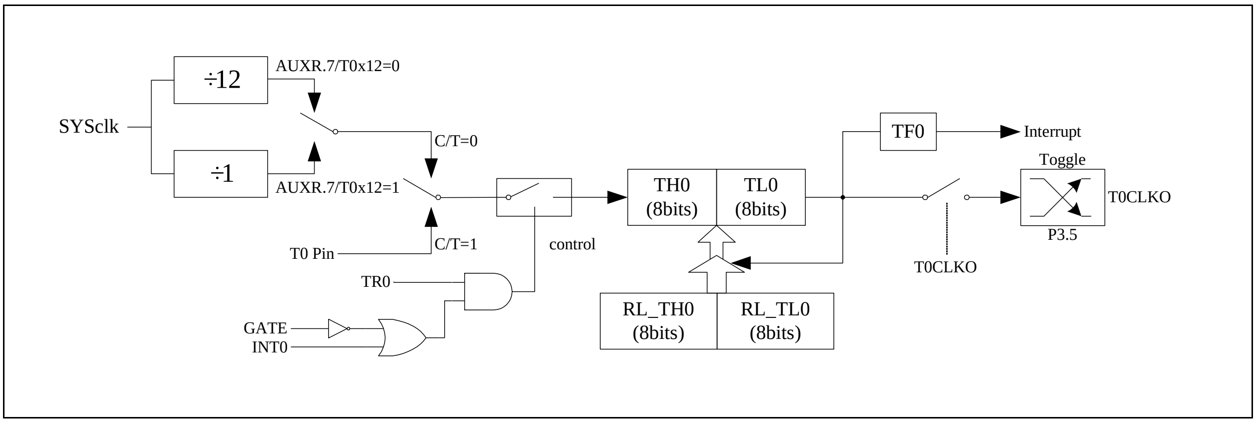
if T0 works in 12T mode (AUXR.7/T0x12=0), the output clock frequency = (SYSclk)/12/(256-TH0)/2

If C/T=1, the timer/counter T0 counts the external pulse input (P3.4/T0), then:

Output clock frequency = (T0_Pin_CLK) / (256-TH0)/2

13.2.6 Timer 0 mode 3 (16-bit auto-realoadable mode with non-maskable interrupt, which can be used as real-time operating system metronome)

For timer/counter 0, its working mode 3 is the same as working mode 0 (the schematic diagram of timer mode 3 in the figure below is the same as working mode 0). The only difference is: when timer/counter 0 is working in mode 3, its interrupt can be enabled just setting ET0/IE.1 (timer/counter 0 interrupt enable bit), and EA/IE.7 (total interrupt enable bit) is not required. The timer/counter 0 interrupt in this mode has nothing to do with the total interrupt enable bit EA. Once the timer/counter 0 interrupt working in mode 3 is enabled (ET0=1), then the interrupt is non-maskable, and the priority of the interrupt is the highest, that is, the interrupt cannot be interrupted by any interrupt, and the interrupt is neither controlled by EA/IE.7 nor controlled by ET0 after it is enabled, When EA=0 or ET0=0, this interrupt cannot be disabled. This mode is so called the 16-bit automatic reload mode with non-maskable interrupt.

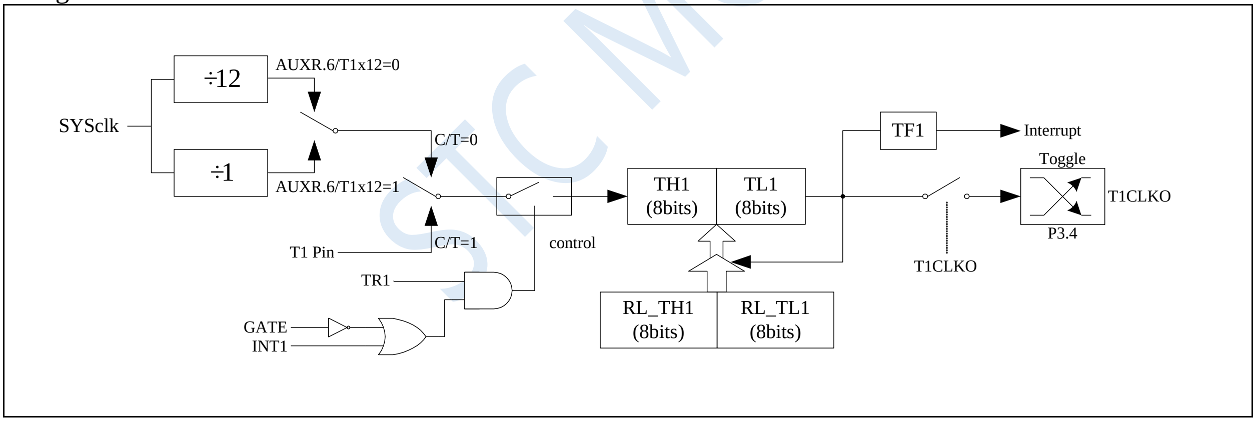


Timer/counter 0 mode 3: 16-bit auto-reload mode with non-maskable interrupt

Note: When Timer/Counter 0 works in mode 3 (16-bit auto-reload mode with non-maskable interrupt), it is not necessary to enable EA/IE.7 (total interrupt enable bit), only ET0/IE.1 is required. (Timer/counter 0 interrupt enable bit) can turn on the timer/counter 0 interrupt. The timer/counter 0 interrupt in this mode has nothing to do with the total interrupt enable bit EA. Once the timer/counter 0 interrupt in this mode is enabled, the timer/counter 0 interrupt priority is the highest, and it cannot be interrupted by any other interrupt (no matter it is lower than the timer/counter 0 interrupt priority). After the interrupt in this mode is enabled, it is neither controlled by EA/IE.7 nor controlled by ET0. Clearing EA nor ET0 can not disable this interrupt.

13.2.7 Timer 1 mode 0 (16-bit auto-reloadable mode)

In this mode, Timer/Counter 1 is used as a 16-bit counter that can be automatically reloaded, as shown in the figure below.



Timer/Counter 1 mode 0: 16-bit auto-reload mode

When GATE=0 (TMOD.7), the timer will count if TR1=1. When GATE=1, it is allowed to control timer1 by external input INT1, so that pulse width measurement can be realized. TR1 is the control bit in the TCON register. For the specific function description of each bit of the TCON register, see the introduction of the TCON register in the previous section.

When C/T=0, the multiplexer is connected to the frequency division output of the system clock. T1 counts the internal system clock, and works in timing mode. When C/T=1, the multiplexer is connected to the external pulse input P3.5/T1, and T1 works in counting mode.

Timer1 of STC microcontroller has two counting rates: one is 12T mode, which is increased by 1 for every 12 clocks, which is the same as traditional 8051 microcontroller, the other is 1T mode, which is increased by 1 for each clock, and the speed is 12 times of traditional 8051. The rate of T1 is determined by T1x12 in the special function register AUXR. If T1x12=0, T1 works in 12T mode, and if T1x12=1, T1 works in 1T mode.

Timer1 has two hidden registers RL_TH1 and RL_TL1. RL_TH1 and TH1 share the same address, and RL_TL1 and TL1 share the same address. When TR1=0, that is, when Timer/Counter1 is disabled, the content written to TL1 will be written to RL_TL1 at the same time, and the content written to TH1 will also be written to RL_TH1 at the same time. When TR1=1, that is, when Timer/Counter1 starts to work, writing content to TL1 is not actually written to the current register TL1, but written to the hidden register RL_TL1, and writing content to TH1 is actually also it is not written into the current register TH1, but into the hidden register RL_TH1, which can cleverly realize the 16-bit reload timer. When reading the contents of TH1 and TL1, the contents be read are the contents of TH1 and TL1, not the contents of RL_TH1 and RL_TL1.

When Timer1 is working in mode 0 (TMOD[5:4]/[M1,M0]=00B), the overflow of [TH1,TL1] not only sets TF1, but also automatically reloads the contents of [RL_TH1,RL_TL1] to [TH1,TL1].

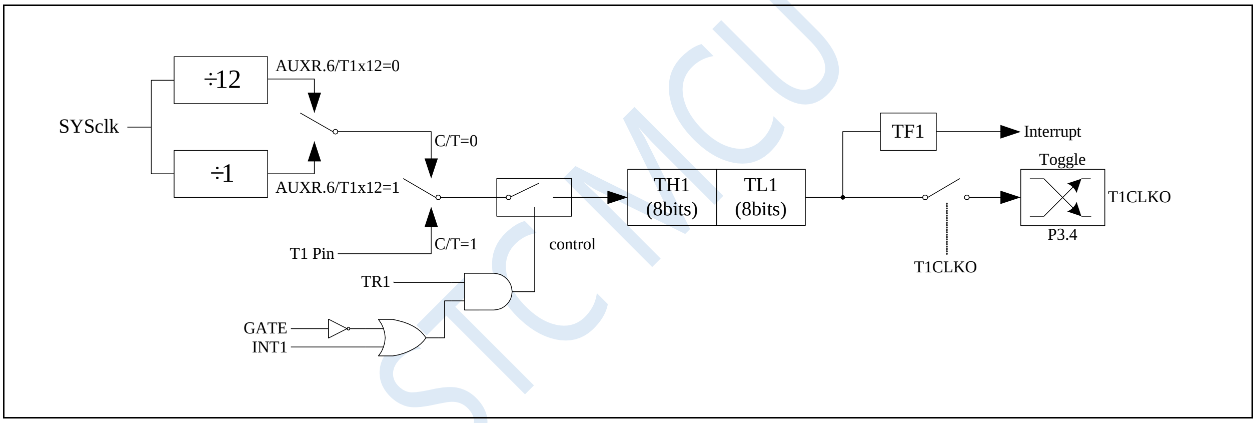
If T1CLKO/INT_CLKO.1=1, the P3.4/T1 pin is configured as timer 1's clock output T1CLKO. The output clock frequency is T1 overflow rate/2.

If C/T=0, the timer/counter 1 counts the internal system clock, then:
if T1 works in 1T mode (AUXR.6/T1x12=1), the output clock frequency = (SYSclk)/(65536-[RL_TH1, RL_TL1])/2
if T1 works in 12T mode (AUXR.6/T1x12=0), the output clock frequency = (SYSclk)/12/(65536-[RL_TH1, RL_TL1])/2

If C/T=1, the timer/counter 1 counts the external pulse input (P3.5/T1), then:
the output clock frequency = (T1_Pin_CLK) / (65536-[RL_TH1, RL_TL1])/2

13.2.8 Timer1 mode 1 (16-bit non-autoreloadable mode)

In this mode, Timer/Counter 1 works in 16-bit non-reloadable mode, as shown in the figure below.



Timer/counter 1 mode 1: 16-bit non-reloadable mode

In this mode, Timer/Counter 1 is configured as a 16-bit non-reloadable mode, which is composed of 8 bits of TL1 and 8 bits of TH1. The 8-bit overflow of TL1 carries over to TH1, and the overflow of TH1 counts the overflow flag TF1 in TCON.

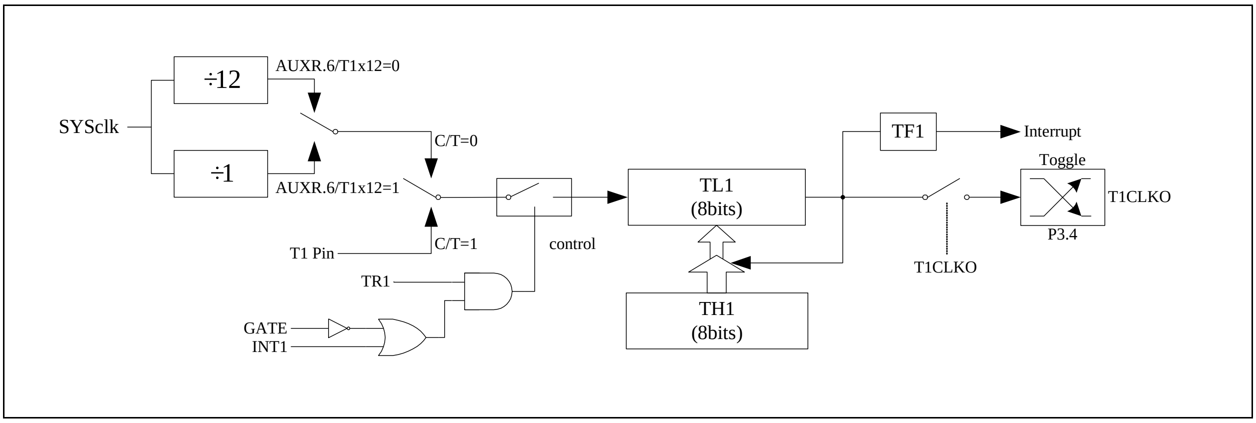
When GATE=0 (TMOD.7), the timer will count if TR1=1. When GATE=1, it is allowed to control timer 1 by external input INT1, so that pulse width measurement can be realized. TR1 is the control bit in the TCON register. For the specific function description of each bit of the TCON register, see the introduction of the TCON register in the previous section.

When C/T=0, the multiplexer is connected to the frequency division output of the system clock, T1 counts the internal system clock, and works in timing mode. When C/T=1, the multiplexer is connected to the external pulse input P3.5/T1, and T1 works in counting mode.

Timer1 of STC microcontroller has two counting rates: one is 12T mode, which is increased by 1 for every 12 clocks, which is the same as traditional 8051 microcontroller, the other is 1T mode, which is increased by 1 for each clock, and the speed is 12 times of traditional 8051. The rate of T1 is determined by T1x12 in the special function register AUXR. If T1x12=0, T1 works in 12T mode, and if T1x12=1, T1 works in 1T mode.

13.2.9 Timer 1 mode 2 (8-bit auto-reloadable mode)

In this mode, Timer/Counter 1 is an 8-bit counter that can be automatically reloaded, as shown in the figure below.



Timer/counter 1 mode 2: 8-bit auto-reloadable mode

The overflow of TL1 not only sets TF1, but also reloads the content of TH1 into TL1. The content of TH1 is preset by software, and its content remains unchanged during reloading.

When T1CLKO/INT_CLKO.1=1, the P3.4/T0 pin is configured as timer 1's clock output T1CLKO. The output clock frequency is T1 overflow rate/2.

If C/T=0, the timer/counter 1 counts the internal system clock, then:

if T1 works in 1T mode (AUXR.6/T1x12=1), the output clock frequency = (SYSclk)/(256-TH1)/2

if T1 works in 12T mode (AUXR.6/T1x12=0), the output clock frequency = (SYSclk)/12/(256-TH1)/2

If C/T=1, the timer/counter T1 counts the external pulse input (P3.5/T1), then:

Output clock frequency = (T1_Pin_CLK) / (256-TH1)/2

13.2.10 Timer 0 Counting Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
TL0	8AH								
TH0	8CH								

When T0 is operating in 16-bit mode (Mode 0, Mode 1, Mode 3), TL0 and TH0 combine into a 16-bit register with TL0 as the low byte and TH0 as the high byte. For 8-bit mode (mode 2), TL0 and TH0 are two independent 8-bit registers.

13.2.11 Timer 1 Counting Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
TL1	8BH								
TH1	8DH								

When T1 is operating in 16-bit mode (Mode 0, Mode 1, Mode 3), TL1 and TH1 combine into a 16-bit register with TL1 as the low byte and TH1 as the high byte. For 8-bit mode (mode 2), TL1 and TH1 are two independent 8-bit registers.

13.2.12 Auxiliary Register 1 (AUXR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
AUXR	8EH	T0x12	T1x12	UART_M0x6	T2R	T2_C/T	T2x12	EXTRAM	S1ST2

T0x12: T0 speed control bit.

0: The clock source of T0 is SYSclk/12.
1: The clock source of T0 is SYSclk/1.

T1x12: T1 speed control bit.

0: The clock source of T1 is SYSclk/12.
1: The clock source of T1 is SYSclk/1.

13.2.13 External Interrupt and Clock Output Control Register (INTCLKO)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
INTCLKO	8FH	-	EX4	EX3	EX2	-	T2CLKO	T1CLKO	T0CLKO

T0CLKO: T0 clock out control bit.

0: Turn off the clock output.
1: P3.5 is configured for T0 clock output pin. When T0 overflows, P3.5 will flip automatically.

T1CLKO: T0 clock out control bit.

0: Turn off the clock output.
1: P3.4 is configured for T1 clock output pin. When T1 overflows, P3.4 will flip automatically.

13.2.14 Timer 0 calculation formula

Mode of Timer	Speed	Period calculation formula
Mode 0/3 (16-bit automatic reload)	1T	$\text{Timer period} = \frac{65536 - [TH0, TL0]}{SYSclk}$ (Auto-reload)
	12T	(Auto-reload)
Mode 1 (16-bit does not automatically reload)	1T	$\text{Timer period} = \frac{65536 - [TH0\ TL0]}{SYSclk}$ (Soft reload)
	12T	(Soft reload)
Mode 2 (8-bit automatic reload)	1T	$\text{Timer period} = \frac{256 - TH0}{SYSclk}$

	12T	(Auto-reload)

13.2.15 Timer 1 calculation formula

Mode of Timer	Speed	Period calculation formula
Mode 0 (16-bit automatic reload)	1T	$\text{Timer period} = \frac{65536 - TH1, TL1}{SYSclk}$ (Auto-reload)
	12T	(Auto-reload)
Mode 1 (16-bit does not automatically reload)	1T	$\text{Timer period} = \frac{65536 - TH1\ \ TL1}{SYSclk}$ (Soft reload)
	12T	(Soft reload)
Mode 2 (8-bit automatic reload)	1T	$\text{Timer period} = \frac{256 - TH1}{SYSclk}$ (Auto-reload)
	12T	(Auto-reload)

13.3 Timer 2

13.3.1 Auxiliary Register 1 (AUXR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
AUXR	8EH	T0x12	T1x12	UART_M0x6	T2R	T2_C/T	T2x12	EXTRAM	S1ST2

TR2: T2 run control bit.

0: T2 stops counting.

1: T2 start counting.

T2_C/T: T2 mode select bit.

0: T2 is used as a timer (input pulse is from internal system clock);

1: T2 is used as a counter (input pulse is from external T2/P1.2 pin).

T2x12: T2 speed control bit.

0: The clock source of T2 is SYSclk/12.

1: The clock source of T2 is SYSclk/1.

13.3.2 External Interrupt and Clock Output Control Register (INTCLKO)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
INTCLKO	8FH	-	EX4	EX3	EX2	-	T2CLKO	T1CLKO	T0CLKO

T2CLKO: T2 clock out control bit.

0: Turn off the clock output.

1: P1.3 is configured for T2 clock output pin. When T2 overflows, P1.3 will flip automatically.

13.3.3 Timer 2 Counting Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
T2L	D7H								
T2H	D6H								

T2 operates in 16-bit auto-reload mode. T2L and T2H combine into a 16-bit register with T2L as the low byte and T2H as the high byte. When the 16-bit counter [T2H, T2L] overflows, the system loads the reload value in the internal 16-bit reload register into [T2H, T2L] automatically.

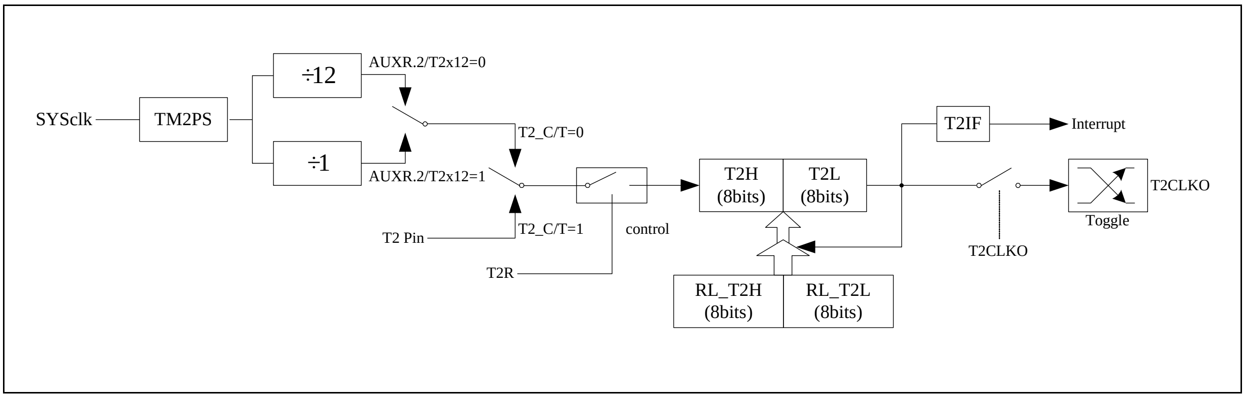
13.3.4 Timer 2 8-bit Prescaler Register (TM2PS)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
TM2PS	FEA2H								

Timer 2 clock ＝ SYSclk ÷（ TM2PS + 1 ）

13.3.5 Timer 2 working mode

The functional block diagram of Timer/Counter 2 is as follows.



Timer/counter 2 working mode: 16-bit auto-reload mode

T2R/AUXR.4 is the control bit in the AUXR register. For the specific function description of each bit of the AUXR register, see the introduction of the AUXR register in the previous section.

When T2_C/T=0, the multiplexer is connected to the frequency division output of the system clock, T2 counts the internal system clock, and works in timing mode. When T2_C/T=1, the multiplexer is connected to the external pulse input T2, and T2 works in counting mode.

Timer2 of STC microcontroller has two counting rates: one is 12T mode, which is increased by 1 for every 12 clocks, which is the same as traditional 8051 microcontroller, the other is 1T mode, which is increased by 1 for each clock, and the speed is 12 times of traditional 8051. The rate of T2 is determined by T2x12 in the special function register AUXR. If T2x12=0, T2 works in 12T mode, and if T2x12=1, T1 works in 1T mode.

Timer2 has two hidden registers RL_T2H and RL_T2L. RL_T2H and T2H share the same address, and RL_T2L and T2L share the same address. When T2R=0, that is, when Timer/Counter2 is disabled, the content written to T2L will be written to RL_T2L at the same time, and the content written to T2H will also be written to RL_T2H at the same time. When T2R=1, that is, when Timer/Counter2 starts to work, writing content to T2L is not actually written to the current register T2L, but written to the hidden register RL_T2L, and writing content to T2H is actually also it is not written into the current register T2H, but into the hidden register RL_T2H, which can cleverly realize the 16-bit reload timer. When reading the contents of T2H and T2L, the contents be read are the contents of T2H and T2L, not the contents of RL_T2H and RL_T2L.

The overflow of [T2H, T2L] not only sets the interrupt request flag (T2IF), which causes the CPU to switch to the timer 2 interrupt routine, but also automatically reloads the contents of [RL_T2H, RL_T2L] into [T2H, T2L].

13.3.6 Timer 2 calculation formula

Speed	Period calculation formula
1T	$\text{Timer period} = \frac{65536 \mid T2H, T2L}{SYSclk}$ (Auto-reload)
12T	(Auto-reload)

13.4 Timer 3/4

13.4.1 Timer4 and Timer 3 Control Register (T4T3M)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
T4T3M	D1H	T4R	T4_C/T	T4x12	T4CLKO	T3R	T3_C/T	T3x12	T3CLKO

TR4: T4 run control bit.

0: T4 stops counting.

1: T4 start counting.

T4_C/T: T4 mode select bit.

0: T4 is used as a timer (input pulse is from internal system clock);

1: T4 is used as a counter (input pulse is from external T4/P0.6 pin).

T4x12: T4 speed control bit.

0: The clock source of T4 is SYSclk/12.

1: The clock source of T4 is SYSclk/1.

T4CLKO: T4 clock out control bit.

0: Turn off the clock output.

1: P0.7 is configured for T4 clock output pin. When T4 overflows, P0.7 will flip automatically.

TR3: T3 run control bit.

0: T3 stops counting.

1: T3 start counting.

T3_C/T: T3 mode select bit.

0: T3 is used as a timer (input pulse is from internal system clock);

1: T3 is used as a counter (input pulse is from external T3/P0.4 pin).

T3x12: T3 speed control bit.

0: The clock source of T3 is SYSclk/12.

1: The clock source of T3 is SYSclk/1.

T3CLKO: T3 clock out control bit.

0: Turn off the clock output.

1: P0.5 is configured for T3 clock output pin. When T3 overflows, P0.5 will flip automatically.

13.4.2 Timer 3 Counting Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
T3L	D5H								
T3H	D4H								

T3 operates in 16-bit auto-reload mode. T3L and T3H combine into a 16-bit register with T3L as the low byte and T3H as the high byte. When the 16-bit counter [T3H, T3L] overflows, the system loads the reload value in the internal 16-bit reload register into [T3H, T3L] automatically.

13.4.3 Timer 4 Counting Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
T4L	D3H								
T4H	D2H								

T4 operates in 16-bit auto-reload mode. T4L and T4H combine into a 16-bit register with T4L as the low byte and T4H as the high byte. When the 16-bit counter [T4H, T4L] overflows, the system loads the reload value in the internal 16-bit reload register into [T4H, T4L] automatically.

13.4.4 Timer 3 8-bit Prescaler Register (TM3PS)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
TM3PS	FEA3H								

Timer 3 clock = SYSclk ÷ (TM3PS + 1)

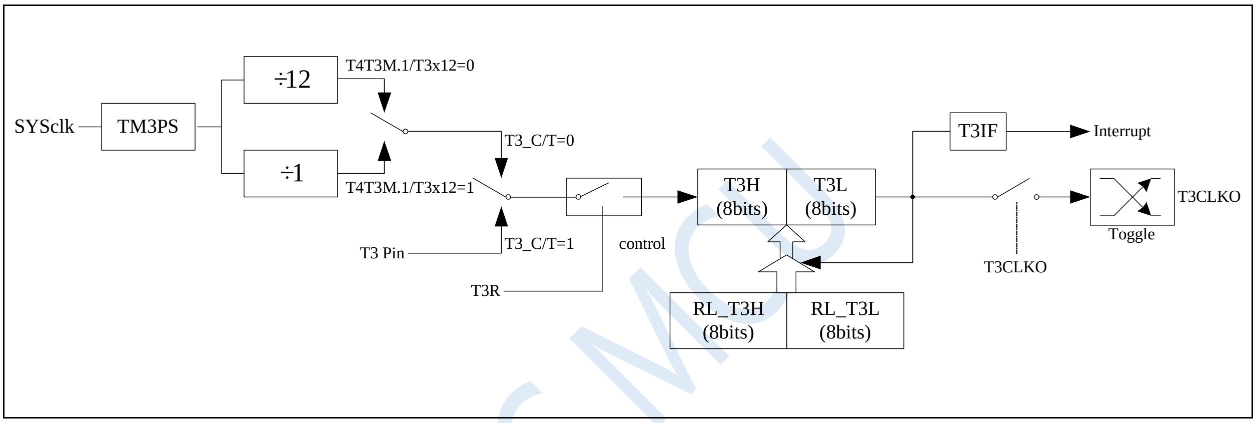
13.4.5 Timer 4 8-bit Prescaler Register (TM4PS)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
TM4PS	FEA4H								

Timer 4 clock = SYSclk ÷ (TM4PS + 1)

13.4.6 Timer 3 working mode

The functional block diagram of Timer/Counter 3 is as follows.



Timer/counter 3 working mode: 16-bit auto-reload mode

T3R/T4T3M.3 is the control bit in the T4T3M register. For the specific function description of each bit of the T4T3M register, see the introduction of the T4T3M register in the previous section.

When T3_C/T=0, the multiplexer is connected to the frequency division output of the system clock, T3 counts the internal system clock, and works in timing mode. When T3_C/T=1, the multiplexer is connected to the external pulse input T3, and T3 works in counting mode.

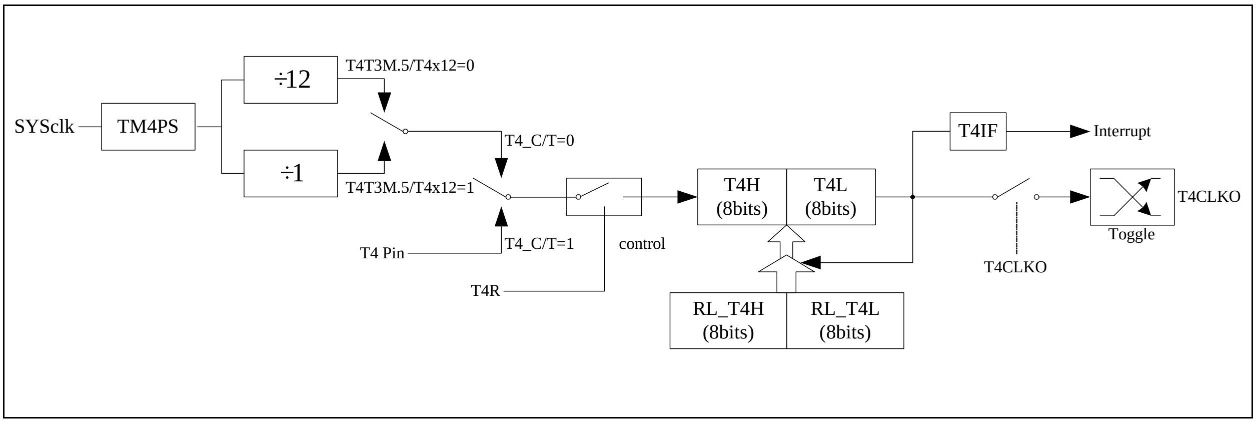
Timer3 of STC microcontroller has two counting rates: one is 12T mode, which is increased by 1 for every 12 clocks, which is the same as traditional 8051 microcontroller, the other is 1T mode, which is increased by 1 for each clock, and the speed is 12 times of traditional 8051. The rate of T3 is determined by T3x12 in the special function register T4T3M. If T3x12=0, T3 works in 12T mode, and if T3x12=1, T3 works in 1T mode.

Timer3 has two hidden registers RL_T3H and RL_T3L. RL_T3H and T3H share the same address, and RL_T3L and T3L share the same address. When T3R=0, that is, when Timer/Counter3 is disabled, the content written to T3L will be written to RL_T3L at the same time, and the content written to T3H will also be written to RL_T3H at the same time. When T3R=1, that is, when Timer/Counter3 starts to work, writing content to T3L is not actually written to the current register T3L, but written to the hidden register RL_T3L, and writing content to T3H is actually also it is not written into the current register T3H, but into the hidden register RL_T3H, which can cleverly realize the 16-bit reload timer. When reading the contents of T3H and T3L, the contents be read are the contents of T3H and T3L, not the contents of RL_T3H and RL_T3L.

The overflow of [T3H, T3L] not only sets the interrupt request flag (T3IF), which causes the CPU to switch to the timer 3 interrupt routine, but also automatically reloads the contents of [RL_T3H, RL_T3L] into [T3H, T3L].

13.4.7 Timer 4 working mode

The functional block diagram of Timer/Counter 4 is as follows.



Timer/counter 4 working mode: 16-bit auto-reload mode

T4R/T4T3M.7 is the control bit in the T4T3M register. For the specific function description of each bit of the T4T3M register, see the introduction of the T4T3M register in the previous section.

When T4_C/T=0, the multiplexer is connected to the frequency division output of the system clock, T4 counts the internal system clock, and works in timing mode. When T4_C/T=1, the multiplexer is connected to the external pulse input T4, and T4 works in counting mode.

Timer4 of STC microcontroller has two counting rates: one is 12T mode, which is increased by 1 for every 12 clocks, which is the same as traditional 8051 microcontroller, the other is 1T mode, which is increased by 1 for each clock, and the speed is 12 times of traditional 8051. The rate of T4 is determined by T4x12 in the special function register T4T3M. If T4x12=0, T4 works in 12T mode, and if T4x12=1, T4 works in 1T mode.

Timer4 has two hidden registers RL_T4H and RL_T4L. RL_T4H and T4H share the same address, and RL_T4L and T4L share the same address. When T4R=0, that is, when Timer/Counter4 is disabled, the content written to T4L will be written to RL_T4L at the same time, and the content written to T4H will also be written to RL_T4H at the same time. When T4R=1, that is, when Timer/Counter4 starts to work, writing content to T4L is not actually written to the current register T4L, but written to the hidden register RL_T4L, and writing content to T4H is actually also it is not written into the current register T4H, but into the hidden register RL_T4H, which can cleverly realize the 16-bit reload timer. When reading the contents of T4H and T4L, the contents be read are the contents of T4H and T4L, not the contents of RL_T4H and RL_T4L.

The overflow of [T4H, T4L] not only sets the interrupt request flag (T4IF), which causes the CPU to switch to the timer 4 interrupt routine, but also automatically reloads the contents of [RL_T4H, RL_T4L] into [T4H, T4L].

13.4.8 Timer 3 calculation formula

Speed of Timer	Period calculation formula
1T	$\text{Timer period} = \frac{65536 \mid T3H, T3L}{SYSclk}$ (Auto-reload)
12T	(Auto- reload)

13.4.9 Timer 4 calculation formula

Speed of Timer	Period calculation formula
1T	$\text{Timer period} = \frac{65536 - T4H, T4L}{SYSclk}$ (Auto-reload)
12T	

13.5 Example Routines

13.5.1 Timer 0 (Mode 0 – 16-bit auto reload)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sbit        P10         =       P1^0;

void TM0_Isr() interrupt 1
{
        P10 = !P10;                                         //Test port
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        TMOD = 0x00;                                        //Mode 0
        TL0 = 0x66;                                         //65536-11.0592M/12/1000
        TH0 = 0xfc;
        TR0 = 1;                                            //Start timer
        ET0 = 1;                                            //Enable timer interrupt
        EA = 1;

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             000BH
                LJMP            TM0ISR

                ORG             0100H
TM0ISR:
                CPL             P1.0                    ;Test port
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             TMOD,#00H               ;Mode 0
                MOV             TL0,#66H                ;65536-11.0592M/12/1000
                MOV             TH0,#0FCH
                SETB            TR0                     ;Start timer
                SETB            ET0                     ;Enable timer interrupt
                SETB            EA

                JMP             $

                END
```

13.5.2 Timer 0 (Mode 1 – 16-bit non-auto reload)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
```

```
#include "intrins.h"

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sbit        P10         =       P1^0;

void TM0_Isr() interrupt 1
{
    TL0 = 0x66;                                         //Reset parameters
    TH0 = 0xfc;
    P10 = !P10;                                         //Test port
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    TMOD = 0x01;                                        //Mode 1
    TL0 = 0x66;                                         //65536-11.0592M/12/1000
    TH0 = 0xfc;
    TR0 = 1;                                            //Start timer
    ET0 = 1;                                            //Enable timer interrupt
    EA = 1;

    while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
```

```
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             000BH
                LJMP            TM0ISR

                ORG             0100H
TM0ISR:
                MOV             TL0,#66H                        ;Reset parameters
                MOV             TH0,#0FCH
                CPL             P1.0                            ;Test port
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             TMOD,#01H                       ;Mode 1
                MOV             TL0,#66H                        ;65536-11.0592M/12/1000
                MOV             TH0,#0FCH
                SETB            TR0                             ;Start timer
                SETB            ET0                             ;Enable timer interrupt
                SETB            EA

                JMP             $

                END
```

13.5.3 Timer 0 (Mode 2 - 8-bit auto reload)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

sbit        P10             =       P1^0;

void TM0_Isr() interrupt 1
{
        P10 = !P10;                                     //Test port
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        TMOD = 0x02;                                    //Mode 2
        TL0 = 0xf4;                                     //256-11.0592M/12/76K
        TH0 = 0xf4;
        TR0 = 1;                                        //Start timer
        ET0 = 1;                                        //Enable timer interrupt
        EA = 1;

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             000BH
```

```
                    LJMP        TM0ISR

                    ORG         0100H
TM0ISR:
                    CPL         P1.0                    ;Test port
                    RETI

MAIN:
                    MOV         SP, #5FH
                    MOV         P0M0, #00H
                    MOV         P0M1, #00H
                    MOV         P1M0, #00H
                    MOV         P1M1, #00H
                    MOV         P2M0, #00H
                    MOV         P2M1, #00H
                    MOV         P3M0, #00H
                    MOV         P3M1, #00H
                    MOV         P4M0, #00H
                    MOV         P4M1, #00H
                    MOV         P5M0, #00H
                    MOV         P5M1, #00H

                    MOV         TMOD,#02H               ;Mode 2
                    MOV         TL0,#0F4H               ;256-11.0592M/12/76K
                    MOV         TH0,#0F4H
                    SETB        TR0                     ;Start timer
                    SETB        ET0                     ;Enable timer interrupt
                    SETB        EA

                    JMP         $

                    END
```

13.5.4 Timer 0 (Mode 3 - 16-bit auto reload with non-maskable interrupt)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;
```

```
sbit        P10              =      P1^0;

void TM0_Isr() interrupt 1
{
        P10 = !P10;                                          //Test port
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        TMOD = 0x03;                                         //Mode 3
        TL0 = 0x66;                                          //65536-11.0592M/12/1000
        TH0 = 0xfc;
        TR0 = 1;                                             //Start timer
        ET0 = 1;                                             //Enable timer interrupt
//      EA = 1;                                              // Not controlled by EA

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             000BH
                LJMP            TM0ISR

                ORG             0100H
TM0ISR:
                CPL             P1.0                    ;Test port
                RETI

MAIN:
```

```
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                MOV         TMOD,#03H                   ;Mode 3
                MOV         TL0,#66H                    ;65536-11.0592M/12/1000
                MOV         TH0,#0FCH
                SETB        TR0                         ;Start timer
                SETB        ET0                         ;Enable timer interrupt
;               SETB        EA                          ; Not controlled by EA

                JMP         $

                END
```

13.5.5 Timer 0 (External count - T0 is extended for external falling edge interrupt)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr     P0M1        =       0x93;
sfr     P0M0        =       0x94;
sfr     P1M1        =       0x91;
sfr     P1M0        =       0x92;
sfr     P2M1        =       0x95;
sfr     P2M0        =       0x96;
sfr     P3M1        =       0xb1;
sfr     P3M0        =       0xb2;
sfr     P4M1        =       0xb3;
sfr     P4M0        =       0xb4;
sfr     P5M1        =       0xc9;
sfr     P5M0        =       0xca;

sbit    P10         =       P1^0;

void TM0_Isr() interrupt 1
{
    P10 = !P10;                                 //Test port
}

void main()
```

```
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        TMOD = 0x04;                                            //External counting mode
        TL0 = 0xff;
        TH0 = 0xff;
        TR0 = 1;                                                //Start timer
        ET0 = 1;                                                //Enable timer interrupt
        EA = 1;

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             000BH
                LJMP            TM0ISR

                ORG             0100H
TM0ISR:
                CPL             P1.0                    ;Test port
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                MOV         TMOD,#04H                   ;External counting mode
                MOV         TL0,#0FFH
                MOV         TH0,#0FFH
                SETB        TR0                         ;Start timer
                SETB        ET0                         ;Enable timer interrupt
                SETB        EA

                JMP         $

                END
```

13.5.6 Timer 0 (Pulse width measurement for high-level width of INT0)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         AUXR        =       0x8e;

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

void INT0_Isr() interrupt 0
{
        P0 = TL0;                                       //TL0 is the low byte of the measured value
        P1 = TH0;                                       //TH0 is the high byte of the measured value
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        AUXR = 0x80;                                    //1T mode
        TMOD = 0x08;                                    //Enable GATE, and enable timing when INT0 is 1
        TL0 = 0x00;
        TH0 = 0x00;
        while (INT0);                                   //Wait for INT0 to be low
        TR0 = 1;                                        //Start timer
        IT0 = 1;                                        //Enable INT0 falling edge interrupt
        EX0 = 1;
        EA = 1;

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

AUXR        DATA        8EH
P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         0003H
            LJMP        INT0ISR

            ORG         0100H
INT0ISR:
            MOV         P0,TL0                  ;TL0 is the low byte of the measured value
            MOV         P1,TH0                  ;TH0 is the high byte of the measured value
            RETI

MAIN:
            MOV         SP, #5FH
            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
            MOV         P3M1, #00H
            MOV         P4M0, #00H
```

```
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                MOV         AUXR,#80H                   ;1T mode
                MOV         TMOD,#08H                   ;Enable GATE, and enable timing when INT0 is 1
                MOV         TL0,#00H
                MOV         TH0,#00H
                JB          INT0,$                      ;Wait for INT0 to be low
                SETB        TR0                         ;Start timer
                SETB        IT0                         ;Enable INT0 falling edge interrupt
                SETB        EX0
                SETB        EA

                JMP         $

                END
```

13.5.7 Timer 0 (Mode 0, Divided clock output)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr     INTCLKO     =   0x8f;

sfr     P0M1        =   0x93;
sfr     P0M0        =   0x94;
sfr     P1M1        =   0x91;
sfr     P1M0        =   0x92;
sfr     P2M1        =   0x95;
sfr     P2M0        =   0x96;
sfr     P3M1        =   0xb1;
sfr     P3M0        =   0xb2;
sfr     P4M1        =   0xb3;
sfr     P4M0        =   0xb4;
sfr     P5M1        =   0xc9;
sfr     P5M0        =   0xca;

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    TMOD = 0x00;                                //Mode 0
    TL0 = 0x66;                                         //65536-11.0592M/12/1000
    TH0 = 0xfc;
    TR0 = 1;                                            //Start timer
    INTCLKO = 0x01;                                     //Enable clock output

    while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

INTCLKO     DATA        8FH
P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN

            ORG         0100H
MAIN:
            MOV         SP, #5FH
            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
            MOV         P3M1, #00H
            MOV         P4M0, #00H
            MOV         P4M1, #00H
            MOV         P5M0, #00H
            MOV         P5M1, #00H

            MOV         TMOD,#00H               ;Mode 0
            MOV         TL0,#66H                ;65536-11.0592M/12/1000
            MOV         TH0,#0FCH
            SETB        TR0                     ;Start timer
            MOV         INTCLKO,#01H            ;Enable clock output

            JMP         $

            END
```

13.5.8 Timer 1 (Mode 0 - 16-bit auto reload)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

sbit        P10             =       P1^0;

void TM1_Isr() interrupt 3
{
    P10 = !P10;                                         //Test port
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    TMOD = 0x00;                                        //Mode 0
    TL1 = 0x66;                                         //65536-11.0592M/12/1000
    TH1 = 0xfc;
    TR1 = 1;                                            //Start timer
    ET1 = 1;                                            //Enable timer interrupt
    EA = 1;

    while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
```

```
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             001BH
                LJMP            TM1ISR

                ORG             0100H
TM1ISR:
                CPL             P1.0                    ;Test port
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             TMOD,#00H               ;Mode 0
                MOV             TL1,#66H                ;65536-11.0592M/12/1000
                MOV             TH1,#0FCH
                SETB            TR1                     ;Start timer
                SETB            ET1                     ;Enable timer interrupt
                SETB            EA

                JMP             $

                END
```

13.5.9 Timer 1 (Mode 1 - 16-bit non-auto reload)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

sbit        P10             =       P1^0;

void TM1_Isr() interrupt 3
{
        TL1 = 0x66;                                         //Reset parameters
        TH1 = 0xfc;
        P10 = !P10;                                         //Test port
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        TMOD = 0x10;                                        //Mode 1
        TL1 = 0x66;                                         //65536-11.0592M/12/1000
        TH1 = 0xfc;
        TR1 = 1;                                            //Start timer
        ET1 = 1;                                            //Enable timer interrupt
        EA = 1;

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             001BH
                LJMP            TM1ISR

                ORG             0100H
TM1ISR:
                MOV             TL1,#66H                        ;Reset parameters
                MOV             TH1,#0FCH
                CPL             P1.0                            ;Test port
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             TMOD,#10H                       ;Mode 1
                MOV             TL1,#66H                        ;65536-11.0592M/12/1000
                MOV             TH1,#0FCH
                SETB            TR1                             ;Start timer
                SETB            ET1                             ;Enable timer interrupt
                SETB            EA

                JMP             $

                END
```

13.5.10 Timer 1 (Mode 2 - 8-bit auto reload)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;
```

```
sbit        P10             =       P1^0;

void TM1_Isr() interrupt 3
{
        P10 = !P10;                                             //Test port
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        TMOD = 0x20;                                            //Mode 2
        TL1 = 0xf4;                                             //256-11.0592M/12/76K
        TH1 = 0xf4;
        TR1 = 1;                                                //Start timer
        ET1 = 1;                                                //Enable timer interrupt
        EA = 1;

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P0M1            DATA        093H
P0M0            DATA        094H
P1M1            DATA        091H
P1M0            DATA        092H
P2M1            DATA        095H
P2M0            DATA        096H
P3M1            DATA        0B1H
P3M0            DATA        0B2H
P4M1            DATA        0B3H
P4M0            DATA        0B4H
P5M1            DATA        0C9H
P5M0            DATA        0CAH

                ORG         0000H
                LJMP        MAIN
                ORG         001BH
                LJMP        TM1ISR

                ORG         0100H
TM1ISR:
                CPL         P1.0                        ;Test port
                RETI

MAIN:
```

```
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                MOV         TMOD,#20H                   ;Mode 2
                MOV         TL1,#0F4H                   ;256-11.0592M/12/76K
                MOV         TH1,#0F4H
                SETB        TR1                         ;Start timer
                SETB        ET1                         ;Enable timer interrupt
                SETB        EA

                JMP         $

                END
```

13.5.11 Timer 1 (External count – T1 is extended for external falling edge interrupt)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

sbit        P10             =       P1^0;

void TM1_Isr() interrupt 3
{
    P10 = !P10;                                 //Test port
}

void main()
```

```
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        TMOD = 0x40;                                //External counting mode
        TL1 = 0xff;
        TH1 = 0xff;
        TR1 = 1;                                    //Start timer
        ET1 = 1;                                    //Enable timer interrupt
        EA = 1;

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         001BH
            LJMP        TM1ISR

            ORG         0100H
TM1ISR:
            CPL         P1.0                    ;Test port
            RETI

MAIN:
            MOV         SP, #5FH
            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
```

```
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                MOV         TMOD,#40H                   ;External counting mode
                MOV         TL1,#0FFH
                MOV         TH1,#0FFH
                SETB        TR1                         ;Start timer
                SETB        ET1                         ;Enable timer interrupt
                SETB        EA

                JMP         $

                 END
```

13.5.12 Timer 1 (Pulse width measurement for high-level width of INT1)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sfr         AUXR        =       0x8e;

void INT1_Isr() interrupt 2
{
    P0 = TL1;                                   //TL1 is the low byte of the measured value
    P1 = TH1;                                   //TH1 is the high byte of the measured value
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    AUXR = 0x40;                              //1T mode
    TMOD = 0x80;                              //Enable GATE, and enable timing when INT1 is 1
    TL1 = 0x00;
    TH1 = 0x00;
    while (INT1);                             //Wait for INT1 to be low
    TR1 = 1;                                  //Start timer
    IT1 = 1;                                  //Enable INT1 falling edge interrupt
    EX1 = 1;
    EA = 1;

    while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

AUXR        DATA        8EH
P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         0013H
            LJMP        INT1ISR

            ORG         0100H
INT1ISR:
            MOV         P0,TL1                  ;TL1 is the low byte of the measured value
            MOV         P1,TH1                  ;TH1 is the high byte of the measured value
            RETI

MAIN:
            MOV         SP, #5FH
            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
            MOV         P3M1, #00H
            MOV         P4M0, #00H
```

```
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             AUXR,#40H                   ;1T mode
                MOV             TMOD,#80H                   ;Enable GATE, and enable timing when INT1 is 1
                MOV             TL1,#00H
                MOV             TH1,#00H
                JB              INT1,$                      ;Wait for INT1 to be low
                SETB            TR1                         ;Start timer
                SETB            IT1                         ;Enable INT1 falling edge interrupt
                SETB            EX1
                SETB            EA

                JMP             $

                END
```

13.5.13 Timer 1 (Mode 0, Divided clock output)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr     INTCLKO     =   0x8f;

sfr     P0M1        =   0x93;
sfr     P0M0        =   0x94;
sfr     P1M1        =   0x91;
sfr     P1M0        =   0x92;
sfr     P2M1        =   0x95;
sfr     P2M0        =   0x96;
sfr     P3M1        =   0xb1;
sfr     P3M0        =   0xb2;
sfr     P4M1        =   0xb3;
sfr     P4M0        =   0xb4;
sfr     P5M1        =   0xc9;
sfr     P5M0        =   0xca;

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        TMOD = 0x00;                                //Mode 0
```

```
    TL1 = 0x66;                                     //65536-11.0592M/12/1000
    TH1 = 0xfc;
    TR1 = 1;                                        //Start timer
    INTCLKO = 0x02;                                 //Enable clock output

    while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

INTCLKO     DATA        8FH
P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN

            ORG         0100H
MAIN:
            MOV         SP, #5FH
            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
            MOV         P3M1, #00H
            MOV         P4M0, #00H
            MOV         P4M1, #00H
            MOV         P5M0, #00H
            MOV         P5M1, #00H

            MOV         TMOD,#00H               ;Mode 0
            MOV         TL1,#66H                ;65536-11.0592M/12/1000
            MOV         TH1,#0FCH
            SETB        TR1                     ;Start timer
            MOV         INTCLKO,#02H            ;Enable clock output

            JMP         $

            END
```

13.5.14 Timer 1 (Mode 0) is used as baud rate generator of UART1

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (65536 - FOSC / 115200 / 4)

sfr         AUXR            =       0x8e;

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

bit         busy;
char        wptr;
char        rptr;
char        buffer[16];

void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
        buffer[wptr++] = SBUF;
        wptr &= 0x0f;
    }
}

void UartInit()
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}

void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}

void UartSendStr(char *p)
{
    while (*p)
    {
        UartSEND(*p++);
    }
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    UartInit();
    ES = 1;
    EA = 1;
    UartSENDStr("Uart Test !\r\n");

    while (1)
    {
        if (rptr != wptr)
        {
            UartSEND(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

AUXR        DATA        8EH

BUSY        BIT         20H.0
WPTR        DATA        21H
RPTR        DATA        22H
BUFFER      DATA        23H                         ;16 bytes

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
```

```
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0023H
                LJMP            UART_ISR

                ORG             0100H

UART_ISR:
                PUSH            ACC
                PUSH            PSW
                MOV             PSW,#08H

                JNB             TI,CHKRI
                CLR             TI
                CLR             BUSY
CHKRI:
                JNB             RI,UARTISR_EXIT
                CLR             RI
                MOV             A,WPTR
                ANL             A,#0FH
                ADD             A,#BUFFER
                MOV             R0,A
                MOV             @R0,SBUF
                INC             WPTR
UARTISR_EXIT:
                POP             PSW
                POP             ACC
                RETI

UART_INIT:
                MOV             SCON,#50H
                MOV             TMOD,#00H
                MOV             TL1,#0E8H                       ;65536-11059200/115200/4=0FFE8H
                MOV             TH1,#0FFH
                SETB            TR1
                MOV             AUXR,#40H
                CLR             BUSY
                MOV             WPTR,#00H
                MOV             RPTR,#00H
                RET

UART_SEND:
                JB              BUSY,$
                SETB            BUSY
                MOV             SBUF,A
                RET

UART_SENDSTR:
                CLR             A
                MOVC            A,@A+DPTR
                JZ              SENDEND
                LCALL           UART_SEND
                INC             DPTR
                JMP             UART_SENDSTR
SENDEND:
                RET

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                LCALL           UART_INIT
                SETB            ES
                SETB            EA

                MOV             DPTR,#STRING
                LCALL           UART_SENDSTR

LOOP:
                MOV             A,RPTR
                XRL             A,WPTR
                ANL             A,#0FH
                JZ              LOOP
                MOV             A,RPTR
                ANL             A,#0FH
                ADD             A,#BUFFER
                MOV             R0,A
                MOV             A,@R0
                LCALL           UART_SEND
                INC             RPTR
                JMP             LOOP

STRING:         DB              'Uart Test !',0DH,0AH,00H

                END
```

13.5.15 Timer 1 (Mode 2) is used as baud rate generator of UART1

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (256 - FOSC / 115200 / 32)

sfr         AUXR            =       0x8e;
sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

bit         busy;
char        wptr;
char        rptr;
char        buffer[16];

void UartIsr() interrupt 4
{
        if (TI)
        {
                TI = 0;
                busy = 0;
        }
        if (RI)
        {
                RI = 0;
                buffer[wptr++] = SBUF;
                wptr &= 0x0f;
        }
}

void UartInit()
{
        SCON = 0x50;
        TMOD = 0x20;
        TL1 = BRT;
        TH1 = BRT;
        TR1 = 1;
        AUXR = 0x40;
        wptr = 0x00;
        rptr = 0x00;
        busy = 0;
}

void UartSend(char dat)
{
        while (busy);
        busy = 1;
        SBUF = dat;
}

void UartSendStr(char *p)
{
        while (*p)
        {
                UartSEND(*p++);
        }
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        UartInit();
        ES = 1;
        EA = 1;
        UartSENDStr("Uart Test !\r\n");

        while (1)
        {
                if (rptr != wptr)
                {
                        UartSEND(buffer[rptr++]);
                        rptr &= 0x0f;
                }
        }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

AUXR            DATA            8EH

BUSY            BIT             20H.0
WPTR            DATA            21H
RPTR            DATA            22H
BUFFER          DATA            23H                             ;16 bytes

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0023H
                LJMP            UART_ISR
                ORG             0100H

UART_ISR:
                PUSH            ACC
                PUSH            PSW
                MOV             PSW,#08H

                JNB             TI,CHKRI
                CLR             TI
                CLR             BUSY
CHKRI:
                JNB             RI,UARTISR_EXIT
                CLR             RI
                MOV             A,WPTR
                ANL             A,#0FH
                ADD             A,#BUFFER
                MOV             R0,A
                MOV             @R0,SBUF
                INC             WPTR
UARTISR_EXIT:
                POP             PSW
                POP             ACC
                RETI

UART_INIT:
                MOV             SCON,#50H
                MOV             TMOD,#20H
                MOV             TL1,#0FDH               ;256-11059200/115200/32=0FDH
                MOV             TH1,#0FDH
                SETB            TR1
                MOV             AUXR,#40H
                CLR             BUSY
                MOV             WPTR,#00H
                MOV             RPTR,#00H
                RET

UART_SEND:
                JB              BUSY,$
                SETB            BUSY
                MOV             SBUF,A
                RET

UART_SENDSTR:
                CLR             A
                MOVC            A,@A+DPTR
                JZ              SENDEND
                LCALL           UART_SEND
                INC             DPTR
                JMP             UART_SENDSTR
SENDEND:
                RET

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                LCALL           UART_INIT
                SETB            ES
                SETB            EA

                MOV             DPTR,#STRING
                LCALL           UART_SENDSTR

LOOP:
                MOV             A,RPTR
                XRL             A,WPTR
                ANL             A,#0FH
                JZ              LOOP
                MOV             A,RPTR
                ANL             A,#0FH
                ADD             A,#BUFFER
                MOV             R0,A
                MOV             A,@R0
                LCALL           UART_SEND
                INC             RPTR
                JMP             LOOP

STRING:         DB              'Uart Test !',0DH,0AH,00H

                END
```

13.5.16 Timer 2 (16-bit auto reload)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         T2L             =       0xd7;
sfr         T2H             =       0xd6;
sfr         AUXR            =       0x8e;
sfr         IE2             =       0xaf;
#define     ET2                     0x04
sfr         AUXINTIF        =       0xef;
#define     T2IF                    0x01

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sbit        P10         =       P1^0;

void TM2_Isr() interrupt 12
{
        P10 = !P10;                                     //Test port
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        T2L = 0x66;                                     //65536-11.0592M/12/1000
        T2H = 0xfc;
        AUXR = 0x10;                                    //Start timer
        IE2 = ET2;                                      //Enable timer interrupt
        EA = 1;

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

T2L             DATA            0D7H
T2H             DATA            0D6H
AUXR            DATA            8EH
IE2             DATA            0AFH
ET2             EQU             04H
AUXINTIF        DATA            0EFH
T2IF            EQU             01H

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
```

```
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0063H
                LJMP            TM2ISR

                ORG             0100H
TM2ISR:
                CPL             P1.0                                    ;Test port
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             T2L,#66H                                ;65536-11.0592M/12/1000
                MOV             T2H,#0FCH
                MOV             AUXR,#10H                               ;Start timer
                MOV             IE2,#ET2                                ;Enable timer interrupt
                SETB            EA

                JMP             $

                END
```

13.5.17 Timer 2 (External count – T2 is extended for external falling edge interrupt)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         T2L             =       0xd7;
sfr         T2H             =       0xd6;
sfr         AUXR            =       0x8e;
sfr         IE2             =       0xaf;
#define     ET2                     0x04
sfr         AUXINTIF        =       0xef;
#define     T2IF                    0x01

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

sbit        P10             =       P1^0;

void TM2_Isr() interrupt 12
{
        P10 = !P10;                                     //Test port
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        T2L = 0xff;
        T2H = 0xff;
        AUXR = 0x18;                                    //Set external counting mode and start timer
        IE2 = ET2;                                      //Enable timer interrupt
        EA = 1;

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

T2L             DATA            0D7H
T2H             DATA            0D6H
AUXR            DATA            8EH
IE2             DATA            0AFH
ET2             EQU             04H
AUXINTIF        DATA            0EFH
T2IF            EQU             01H

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
```

```
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0063H
                LJMP            TM2ISR

                ORG             0100H
TM2ISR:
                CPL             P1.0                    ;Test port
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             T2L,#0FFH
                MOV             T2H,#0FFH
                MOV             AUXR,#18H               ;Set external counting mode and start timer
                MOV             IE2,#ET2                ;Enable timer interrupt
                SETB            EA

                JMP             $

                END
```

13.5.18 Timer 2 (Divided clock output)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr     T2L         =   0xd7;
sfr     T2H         =   0xd6;
sfr     AUXR        =   0x8e;
sfr     INTCLKO     =   0x8f;

sfr     P0M1        =   0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        T2L = 0x66;                                     //65536-11.0592M/12/1000
        T2H = 0xfc;
        AUXR = 0x10;                                    //Start timer
        INTCLKO = 0x04;                                 //Enable clock output

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

T2L             DATA            0D7H
T2H             DATA            0D6H
AUXR            DATA            8EH
INTCLKO         DATA            8FH

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0100H
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             T2L,#66H                        ;65536-11.0592M/12/1000
                MOV             T2H,#0FCH
                MOV             AUXR,#10H                       ;Start timer
                MOV             INTCLKO,#04H                    ;Enable clock output

                JMP             $

                END
```

13.5.19 Timer 2 is used as baud rate generator of UART1

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (65536 - FOSC / 115200 / 4)

sfr         AUXR            =       0x8e;
sfr         T2H             =       0xd6;
sfr         T2L             =       0xd7;

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

bit         busy;
char        wptr;
char        rptr;
char        buffer[16];
```

```
void UartIsr() interrupt 4
{
        if (TI)
        {
                TI = 0;
                busy = 0;
        }
        if (RI)
        {
                RI = 0;
                buffer[wptr++] = SBUF;
                wptr &= 0x0f;
        }
}

void UartInit()
{
        SCON = 0x50;
        T2L = BRT;
        T2H = BRT >> 8;
        AUXR = 0x15;
        wptr = 0x00;
        rptr = 0x00;
        busy = 0;
}

void UartSend(char dat)
{
        while (busy);
        busy = 1;
        SBUF = dat;
}

void UartSendStr(char *p)
{
        while (*p)
        {
                UartSEND(*p++);
        }
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        UartInit();
        ES = 1;
        EA = 1;
        UartSENDStr("Uart Test !\r\n");

        while (1)
        {
                if (rptr != wptr)
                {
                        UartSEND(buffer[rptr++]);
                        rptr &= 0x0f;
                }
        }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

AUXR            DATA            8EH
T2H             DATA            0D6H
T2L             DATA            0D7H

BUSY            BIT             20H.0
WPTR            DATA            21H
RPTR            DATA            22H
BUFFER          DATA            23H                             ;16 bytes

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0023H
                LJMP            UART_ISR

                ORG             0100H

UART_ISR:
                PUSH            ACC
                PUSH            PSW
                MOV             PSW,#08H

                JNB             TI,CHKRI
                CLR             TI
                CLR             BUSY
CHKRI:
                JNB             RI,UARTISR_EXIT
                CLR             RI
                MOV             A,WPTR
                ANL             A,#0FH
                ADD             A,#BUFFER
                MOV         R0,A
                MOV         @R0,SBUF
                INC         WPTR
UARTISR_EXIT:
                POP         PSW
                POP         ACC
                RETI

UART_INIT:
                MOV         SCON,#50H
                MOV         T2L,#0E8H                       ;65536-11059200/115200/4=0FFE8H
                MOV         T2H,#0FFH
                MOV         AUXR,#15H
                CLR         BUSY
                MOV         WPTR,#00H
                MOV         RPTR,#00H
                RET

UART_SEND:
                JB          BUSY,$
                SETB        BUSY
                MOV         SBUF,A
                RET

UART_SENDSTR:
                CLR         A
                MOVC        A,@A+DPTR
                JZ          SENDEND
                LCALL       UART_SEND
                INC         DPTR
                JMP         UART_SENDSTR
SENDEND:
                RET

MAIN:
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                LCALL       UART_INIT
                SETB        ES
                SETB        EA

                MOV         DPTR,#STRING
                LCALL       UART_SENDSTR

LOOP:
                MOV         A,RPTR
                XRL         A,WPTR
                ANL         A,#0FH
                JZ          LOOP
                MOV         A,RPTR
                ANL         A,#0FH
                ADD         A,#BUFFER
                MOV         R0,A
                MOV         A,@R0
                LCALL       UART_SEND
                INC         RPTR
                JMP         LOOP

STRING:         DB          'Uart Test !',0DH,0AH,00H

                END
```

13.5.20 Timer 2 is used as baud rate generator of UART2

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (65536 - FOSC / 115200 / 4)

sfr         AUXR            =       0x8e;
sfr         T2H             =       0xd6;
sfr         T2L             =       0xd7;
sfr         S2CON           =       0x9a;
sfr         S2BUF           =       0x9b;
sfr         IE2             =       0xaf;

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

bit         busy;
char        wptr;
char        rptr;
char        buffer[16];

void Uart2Isr() interrupt 8
{
        if (S2CON & 0x02)
        {
                S2CON &= ~0x02;
                busy = 0;
        }
        if (S2CON & 0x01)
        {
                S2CON &= ~0x01;
                buffer[wptr++] = S2BUF;
                wptr &= 0x0f;
        }
}

void Uart2Init()
{
        S2CON = 0x10;
        T2L = BRT;
        T2H = BRT >> 8;
        AUXR = 0x14;
        wptr = 0x00;
        rptr = 0x00;
        busy = 0;
}

void Uart2Send(char dat)
{
        while (busy);
        busy = 1;
        S2BUF = dat;
}

void Uart2SendStr(char *p)
{
        while (*p)
        {
                Uart2SEND(*p++);
        }
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        Uart2Init();
        IE2 = 0x01;
        EA = 1;
        Uart2SENDStr("Uart Test !\r\n");

        while (1)
        {
                if (rptr != wptr)
                {
```

```
                Uart2SEND(buffer[rptr++]);
                rptr &= 0x0f;
            }
        }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

AUXR        DATA        8EH
T2H         DATA        0D6H
T2L         DATA        0D7H
S2CON       DATA        9AH
S2BUF       DATA        9BH
IE2         DATA        0AFH

BUSY        BIT         20H.0
WPTR        DATA        21H
RPTR        DATA        22H
BUFFER      DATA        23H                         ;16 bytes

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         0043H
            LJMP        UART2_ISR

            ORG         0100H

UART2_ISR:
            PUSH        ACC
            PUSH        PSW
            MOV         PSW,#08H

            MOV         A,S2CON
            JNB         ACC.1,CHKRI
            ANL         S2CON,#NOT 02H
            CLR         BUSY
CHKRI:
            JNB         ACC.0,UART2ISR_EXIT
            ANL         S2CON,#NOT 01H
            MOV         A,WPTR
            ANL         A,#0FH
            ADD         A,#BUFFER
            MOV         R0,A
            MOV         @R0,S2BUF
            INC         WPTR
```

```
UART2ISR_EXIT:
                POP             PSW
                POP             ACC
                RETI

UART2_INIT:
                MOV             S2CON,#10H
                MOV             T2L,#0E8H                       ;65536-11059200/115200/4=0FFE8H
                MOV             T2H,#0FFH
                MOV             AUXR,#14H
                CLR             BUSY
                MOV             WPTR,#00H
                MOV             RPTR,#00H
                RET

UART2_SEND:
                JB              BUSY,$
                SETB            BUSY
                MOV             S2BUF,A
                RET

UART2_SENDSTR:
                CLR             A
                MOVC            A,@A+DPTR
                JZ              SEND2END
                LCALL           UART2_SEND
                INC             DPTR
                JMP             UART2_SENDSTR
SEND2END:
                RET

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                LCALL           UART2_INIT
                MOV             IE2,#01H
                SETB            EA

                MOV             DPTR,#STRING
                LCALL           UART2_SENDSTR

LOOP:
                MOV             A,RPTR
                XRL             A,WPTR
                ANL             A,#0FH
                JZ              LOOP
                MOV             A,RPTR
                ANL             A,#0FH
                ADD             A,#BUFFER
                MOV             R0,A
                MOV             A,@R0
                LCALL           UART2_SEND
                INC             RPTR
                JMP             LOOP

STRING:         DB              'Uart Test !',0DH,0AH,00H

                END
```

13.5.21 Timer 2 is used as baud rate generator of UART3

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (65536 - FOSC / 115200 / 4)

sfr         AUXR            =       0x8e;
sfr         T2H             =       0xd6;
sfr         T2L             =       0xd7;
sfr         S3CON           =       0xac;
sfr         S3BUF           =       0xad;
sfr         IE2             =       0xaf;

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

bit         busy;
char        wptr;
char        rptr;
char        buffer[16];

void Uart3Isr() interrupt 17
{
    if (S3CON & 0x02)
    {
        S3CON &= ~0x02;
        busy = 0;
    }
    if (S3CON & 0x01)
    {
```

```
                S3CON &= ~0x01;
                buffer[wptr++] = S3BUF;
                wptr &= 0x0f;
        }
}

void Uart3Init()
{
        S3CON = 0x10;
        T2L = BRT;
        T2H = BRT >> 8;
        AUXR = 0x14;
        wptr = 0x00;
        rptr = 0x00;
        busy = 0;
}

void Uart3Send(char dat)
{
        while (busy);
        busy = 1;
        S3BUF = dat;
}

void Uart3SendStr(char *p)
{
        while (*p)
        {
                Uart3SEND(*p++);
        }
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        Uart3Init();
        IE2 = 0x08;
        EA = 1;
        Uart3SENDStr("Uart Test !\r\n");

        while (1)
        {
                if (rptr != wptr)
                {
                        Uart3SEND(buffer[rptr++]);
                        rptr &= 0x0f;
                }
        }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

AUXR            DATA            8EH
T2H             DATA            0D6H
T2L             DATA            0D7H
S3CON           DATA            0ACH
S3BUF           DATA            0ADH
IE2             DATA            0AFH

BUSY            BIT             20H.0
WPTR            DATA            21H
RPTR            DATA            22H
BUFFER          DATA            23H                             ;16 bytes

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             008BH
                LJMP            UART3_ISR

                ORG             0100H

UART3_ISR:
                PUSH            ACC
                PUSH            PSW
                MOV             PSW,#08H

                MOV             A,S3CON
                JNB             ACC.1,CHKRI
                ANL             S3CON,#NOT 02H
                CLR             BUSY
CHKRI:
                JNB             ACC.0,UART3ISR_EXIT
                ANL             S3CON,#NOT 01H
                MOV             A,WPTR
                ANL             A,#0FH
                ADD             A,#BUFFER
                MOV             R0,A
                MOV             @R0,S3BUF
                INC             WPTR
UART3ISR_EXIT:
                POP             PSW
                POP             ACC
                RETI

UART3_INIT:
                MOV             S3CON,#10H
                MOV             T2L,#0E8H                   ;65536-11059200/115200/4=0FFE8H
                MOV             T2H,#0FFH
                MOV             AUXR,#14H
                CLR             BUSY
                MOV             WPTR,#00H
                MOV             RPTR,#00H
                RET

UART3_SEND:
                JB              BUSY,$
                SETB            BUSY
                MOV             S3BUF,A
                RET

UART3_SENDSTR:
                CLR             A
                MOVC            A,@A+DPTR
                JZ              SEND3END
                LCALL           UART3_SEND
                INC             DPTR
                JMP             UART3_SENDSTR
SEND3END:
                RET

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                LCALL           UART3_INIT
                MOV             IE2,#08H
                SETB            EA

                MOV             DPTR,#STRING
                LCALL           UART3_SENDSTR

LOOP:
                MOV             A,RPTR
                XRL             A,WPTR
                ANL             A,#0FH
                JZ              LOOP
                MOV             A,RPTR
                ANL             A,#0FH
                ADD             A,#BUFFER
                MOV             R0,A
                MOV         A,@R0
                LCALL       UART3_SEND
                INC         RPTR
                JMP         LOOP

STRING:         DB          'Uart Test !',0DH,0AH,00H

                END
```

13.5.22 Timer 2 is used as baud rate generator of UART4

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (65536 - FOSC / 115200 / 4)

sfr         AUXR            =       0x8e;
sfr         T2H             =       0xd6;
sfr         T2L             =       0xd7;
sfr         S4CON           =       0x84;
sfr         S4BUF           =       0x85;
sfr         IE2             =       0xaf;

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

bit         busy;
char        wptr;
char        rptr;
char        buffer[16];

void Uart4Isr() interrupt 18
{
        if (S4CON & 0x02)
        {
                S4CON &= ~0x02;
                busy = 0;
        }
        if (S4CON & 0x01)
        {
                S4CON &= ~0x01;
                buffer[wptr++] = S4BUF;
                wptr &= 0x0f;
        }
}

void Uart4Init()
{
        S4CON = 0x10;
        T2L = BRT;
        T2H = BRT >> 8;
        AUXR = 0x14;
        wptr = 0x00;
        rptr = 0x00;
        busy = 0;
}

void Uart4Send(char dat)
{
        while (busy);
        busy = 1;
        S4BUF = dat;
}

void Uart4SendStr(char *p)
{
        while (*p)
        {
                Uart4SEND(*p++);
        }
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        Uart4Init();
        IE2 = 0x10;
        EA = 1;
        Uart4SENDStr("Uart Test !\r\n");

        while (1)
        {
                if (rptr != wptr)
                {
                        Uart4SEND(buffer[rptr++]);
                        rptr &= 0x0f;
                }
        }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

AUXR            DATA            8EH
T2H             DATA            0D6H
T2L             DATA            0D7H
S4CON           DATA            84H
S4BUF           DATA            85H
IE2             DATA            0AFH

BUSY            BIT             20H.0
WPTR            DATA            21H
RPTR            DATA            22H
BUFFER          DATA            23H                             ;16 bytes

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0093H
                LJMP            UART4_ISR

                ORG             0100H

UART4_ISR:
                PUSH            ACC
                PUSH            PSW
                MOV             PSW,#08H

                MOV             A,S4CON
                JNB             ACC.1,CHKRI
                ANL             S4CON,#NOT 02H
                CLR             BUSY
CHKRI:
                JNB             ACC.0,UART4ISR_EXIT
                ANL             S4CON,#NOT 01H
                MOV             A,WPTR
                ANL             A,#0FH
                ADD             A,#BUFFER
                MOV             R0,A
                MOV             @R0,S4BUF
                INC             WPTR
UART4ISR_EXIT:
                POP             PSW
                POP             ACC
                RETI

UART4_INIT:
```

```
                MOV         S4CON,#10H
                MOV         T2L,#0E8H                   ;65536-11059200/115200/4=0FFE8H
                MOV         T2H,#0FFH
                MOV         AUXR,#14H
                CLR         BUSY
                MOV         WPTR,#00H
                MOV         RPTR,#00H
                RET

UART4_SEND:
                JB          BUSY,$
                SETB        BUSY
                MOV         S4BUF,A
                RET

UART4_SENDSTR:
                CLR         A
                MOVC        A,@A+DPTR
                JZ          SEND4END
                LCALL       UART4_SEND
                INC         DPTR
                JMP         UART4_SENDSTR
SEND4END:
                RET

MAIN:
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                LCALL       UART4_INIT
                MOV         IE2,#10H
                SETB        EA

                MOV         DPTR,#STRING
                LCALL       UART4_SENDSTR

LOOP:
                MOV         A,RPTR
                XRL         A,WPTR
                ANL         A,#0FH
                JZ          LOOP
                MOV         A,RPTR
                ANL         A,#0FH
                ADD         A,#BUFFER
                MOV         R0,A
                MOV         A,@R0
                LCALL       UART4_SEND
                INC         RPTR
                JMP             LOOP

STRING:         DB              'Uart Test !',0DH,0AH,00H

                END
```

13.5.23 Timer 3 (16-bit auto reload)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         T4T3M       =       0xd1;
sfr         T4L         =       0xd3;
sfr         T4H         =       0xd2;
sfr         T3L         =       0xd5;
sfr         T3H         =       0xd4;
sfr         T2L         =       0xd7;
sfr         T2H         =       0xd6;
sfr         AUXR        =       0x8e;
sfr         IE2         =       0xaf;
#define     ET2                 0x04
#define     ET3                 0x20
#define     ET4                 0x40
sfr         AUXINTIF    =       0xef;
#define     T2IF                0x01
#define     T3IF                0x02
#define     T4IF                0x04

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sbit        P10         =       P1^0;

void TM3_Isr() interrupt 19
{
    P10 = !P10;                         //Test port
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
```

```
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        T3L = 0x66;                         //65536-11.0592M/12/1000
        T3H = 0xfc;
        T4T3M = 0x08;                       //Start timer
        IE2 = ET3;                          //Enable timer interrupt
        EA = 1;

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

T4T3M           DATA        0D1H
T4L             DATA        0D3H
T4H             DATA        0D2H
T3L             DATA        0D5H
T3H             DATA        0D4H
T2L             DATA        0D7H
T2H             DATA        0D6H
AUXR            DATA        8EH
IE2             DATA        0AFH
ET2             EQU         04H
ET3             EQU         20H
ET4             EQU         40H
AUXINTIF        DATA        0EFH
T2IF            EQU         01H
T3IF            EQU         02H
T4IF            EQU         04H

P0M1            DATA        093H
P0M0            DATA        094H
P1M1            DATA        091H
P1M0            DATA        092H
P2M1            DATA        095H
P2M0            DATA        096H
P3M1            DATA        0B1H
P3M0            DATA        0B2H
P4M1            DATA        0B3H
P4M0            DATA        0B4H
P5M1            DATA        0C9H
P5M0            DATA        0CAH

                ORG         0000H
                LJMP        MAIN
                ORG         009BH
                LJMP        TM3ISR

                ORG         0100H
TM3ISR:
                CPL         P1.0                        ;Test port
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             T3L,#66H                        ;65536-11.0592M/12/1000
                MOV             T3H,#0FCH
                MOV             T4T3M,#08H                      ;Start timer
                MOV             IE2,#ET3                        ;Enable timer interrupt
                SETB            EA

                JMP             $

                END
```

13.5.24 Timer 3 (External count – T3 is extended for external falling edge interrupt)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         T4T3M       =       0xd1;
sfr         T4L         =       0xd3;
sfr         T4H         =       0xd2;
sfr         T3L         =       0xd5;
sfr         T3H         =       0xd4;
sfr         T2L         =       0xd7;
sfr         T2H         =       0xd6;
sfr         AUXR        =       0x8e;
sfr         IE2         =       0xaf;
#define     ET2                 0x04
#define     ET3                 0x20
#define     ET4                 0x40
sfr         AUXINTIF    =       0xef;
#define     T2IF                0x01
#define     T3IF                0x02
#define     T4IF                0x04

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

sbit        P10             =       P1^0;

void TM3_Isr() interrupt 19
{
        P10 = !P10;                             //Test port
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        T3L = 0x66;                             //65536-11.0592M/12/1000
        T3H = 0xfc;
        T4T3M = 0x0c;                           //Set external counting mode and start timer
        IE2 = ET3;                              //Enable timer interrupt
        EA = 1;

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz
T4T3M             DATA        0D1H
T4L             DATA        0D3H
T4H             DATA        0D2H
T3L             DATA        0D5H
T3H             DATA        0D4H
T2L             DATA        0D7H
T2H             DATA        0D6H
AUXR            DATA        8EH
IE2             DATA        0AFH
ET2             EQU         04H
ET3             EQU         20H
ET4             EQU         40H
AUXINTIF        DATA        0EFH
T2IF            EQU         01H
T3IF            EQU         02H
```

```
T4IF        EQU         04H

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         009BH
            LJMP        TM3ISR

            ORG         0100H
TM3ISR:
            CPL         P1.0                    ;Test port
            RETI

MAIN:
            MOV         SP, #5FH
            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
            MOV         P3M1, #00H
            MOV         P4M0, #00H
            MOV         P4M1, #00H
            MOV         P5M0, #00H
            MOV         P5M1, #00H

            MOV         T3L,#66H                ;65536-11.0592M/12/1000
            MOV         T3H,#0FCH
            MOV         T4T3M,#0CH              ;Set external counting mode and start timer
            MOV         IE2,#ET3                ;Enable timer interrupt
            SETB        EA

            JMP         $

            END
```

13.5.25 Timer 3 (Divided clock output)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"
```

```
sfr         T4T3M       =       0xd1;
sfr         T4L         =       0xd3;
sfr         T4H         =       0xd2;
sfr         T3L         =       0xd5;
sfr         T3H         =       0xd4;
sfr         T2L         =       0xd7;
sfr         T2H         =       0xd6;

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    T3L = 0x66;                                     //65536-11.0592M/12/1000
    T3H = 0xfc;
    T4T3M = 0x09;                                   //Enable clock output and start timer

    while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

T4T3M           DATA            0D1H
T4L             DATA            0D3H
T4H             DATA            0D2H
T3L             DATA            0D5H
T3H             DATA            0D4H
T2L             DATA            0D7H
T2H             DATA            0D6H

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
```

```
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN

                ORG             0100H
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             T3L,#66H                        ;65536-11.0592M/12/1000
                MOV             T3H,#0FCH
                MOV             T4T3M,#09H                      ;Enable clock output and start timer

                JMP             $

                END
```

13.5.26 Timer 3 is used as baud rate generator of UART3

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (65536 - FOSC / 115200 / 4)

sfr         T4T3M           =       0xd1;
sfr         T4L             =       0xd3;
sfr         T4H             =       0xd2;
sfr         T3L             =       0xd5;
sfr         T3H             =       0xd4;
sfr         T2L             =       0xd7;
sfr         T2H             =       0xd6;
sfr         S3CON           =       0xac;
sfr         S3BUF           =       0xad;
sfr         IE2             =       0xaf;
sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

bit         busy;
char        wptr;
char        rptr;
char        buffer[16];

void Uart3Isr() interrupt 17
{
        if (S3CON & 0x02)
        {
                S3CON &= ~0x02;
                busy = 0;
        }
        if (S3CON & 0x01)
        {
                S3CON &= ~0x01;
                buffer[wptr++] = S3BUF;
                wptr &= 0x0f;
        }
}

void Uart3Init()
{
        S3CON = 0x50;
        T3L = BRT;
        T3H = BRT >> 8;
        T4T3M = 0x0a;
        wptr = 0x00;
        rptr = 0x00;
        busy = 0;
}

void Uart3Send(char dat)
{
        while (busy);
        busy = 1;
        S3BUF = dat;
}

void Uart3SendStr(char *p)
{
        while (*p)
        {
                Uart3SEND(*p++);
        }
}
```

```
void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        Uart3Init();
        IE2 = 0x08;
        EA = 1;
        Uart3SENDStr(“Uart Test !\r\n”);

        while (1)
        {
                if (rptr != wptr)
                {
                        Uart3SEND(buffer[rptr++]);
                        rptr &= 0x0f;
                }
        }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

T4T3M           DATA            0D1H
T4L             DATA            0D3H
T4H             DATA            0D2H
T3L             DATA            0D5H
T3H             DATA            0D4H
T2L             DATA            0D7H
T2H             DATA            0D6H
S3CON           DATA            0ACH
S3BUF           DATA            0ADH
IE2             DATA            0AFH

BUSY            BIT             20H.0
WPTR            DATA            21H
RPTR            DATA            22H
BUFFER          DATA            23H                             ;16 bytes

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
```

```
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             008BH
                LJMP            UART3_ISR

                ORG             0100H

UART3_ISR:
                PUSH            ACC
                PUSH            PSW
                MOV             PSW,#08H

                MOV             A,S3CON
                JNB             ACC.1,CHKRI
                ANL             S3CON,#NOT 02H
                CLR             BUSY
CHKRI:
                JNB             ACC.0,UART3ISR_EXIT
                ANL             S3CON,#NOT 01H
                MOV             A,WPTR
                ANL             A,#0FH
                ADD             A,#BUFFER
                MOV             R0,A
                MOV             @R0,S3BUF
                INC             WPTR
UART3ISR_EXIT:
                POP             PSW
                POP             ACC
                RETI

UART3_INIT:
                MOV             S3CON,#50H
                MOV             T3L,#0E8H                       ;65536-11059200/115200/4=0FFE8H
                MOV             T3H,#0FFH
                MOV             T4T3M,#0AH
                CLR             BUSY
                MOV             WPTR,#00H
                MOV             RPTR,#00H
                RET

UART3_SEND:
                JB              BUSY,$
                SETB            BUSY
                MOV             S3BUF,A
                RET

UART3_SENDSTR:
                CLR             A
                MOVC            A,@A+DPTR
                JZ              SEND3END
                LCALL           UART3_SEND
                INC             DPTR
                JMP             UART3_SENDSTR
SEND3END:
                RET
```

```
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                LCALL           UART3_INIT
                MOV             IE2,#08H
                SETB            EA

                MOV             DPTR,#STRING
                LCALL           UART3_SENDSTR

LOOP:
                MOV             A,RPTR
                XRL             A,WPTR
                ANL             A,#0FH
                JZ              LOOP
                MOV             A,RPTR
                ANL             A,#0FH
                ADD             A,#BUFFER
                MOV             R0,A
                MOV             A,@R0
                LCALL           UART3_SEND
                INC             RPTR
                JMP             LOOP

STRING:         DB              'Uart Test !',0DH,0AH,00H

                END
```

13.5.27 Timer 4 (16-bit auto reload)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         T4T3M           =       0xd1;
sfr         T4L             =       0xd3;
sfr         T4H             =       0xd2;
sfr         T3L             =       0xd5;
sfr         T3H             =       0xd4;
sfr         T2L             =       0xd7;
sfr         T2H             =       0xd6;
sfr         AUXR            =       0x8e;
sfr         IE2             =       0xaf;
#define     ET2                     0x04
#define     ET3                     0x20
#define     ET4                     0x40
sfr         AUXINTIF        =       0xef;
#define     T2IF                    0x01
#define     T3IF                    0x02
#define     T4IF                    0x04

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

sbit        P10             =       P1^0;

void TM4_Isr() interrupt 20
{
        P10 = !P10;                             //Test port
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        T4L = 0x66;                             //65536-11.0592M/12/1000
        T4H = 0xfc;
        T4T3M = 0x80;                           //Start timer
        IE2 = ET4;                              //Enable timer interrupt
        EA = 1;

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

T4T3M           DATA            0D1H
T4L             DATA            0D3H
T4H             DATA            0D2H
```

```
T3L             DATA            0D5H
T3H             DATA            0D4H
T2L             DATA            0D7H
T2H             DATA            0D6H
AUXR            DATA            8EH
IE2             DATA            0AFH
ET2             EQU             04H
ET3             EQU             20H
ET4             EQU             40H
AUXINTIF        DATA            0EFH
T2IF            EQU             01H
T3IF            EQU             02H
T4IF            EQU             04H

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             00A3H
                LJMP            TM4ISR

                ORG             0100H
TM4ISR:
                CPL             P1.0                    ;Test port
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             T4L,#66H                ;65536-11.0592M/12/1000
                MOV             T4H,#0FCH
                MOV             T4T3M,#80H              ;Start timer
                MOV             IE2,#ET4                ;Enable timer interrupt
                SETB            EA

                JMP             $
                END
```

13.5.28 Timer 4 (External count – T4 is extended for external falling edge interrupt)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         T4T3M       =       0xd1;
sfr         T4L         =       0xd3;
sfr         T4H         =       0xd2;
sfr         T3L         =       0xd5;
sfr         T3H         =       0xd4;
sfr         T2L         =       0xd7;
sfr         T2H         =       0xd6;
sfr         AUXR        =       0x8e;
sfr         IE2         =       0xaf;
#define     ET2                 0x04
#define     ET3                 0x20
#define     ET4                 0x40
sfr         AUXINTIF    =       0xef;
#define     T2IF                0x01
#define     T3IF                0x02
#define     T4IF                0x04

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sbit        P10         =       P1^0;

void TM4_Isr() interrupt 20
{
    P10 = !P10;                             //Test port
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    T4L = 0x66;                     //65536-11.0592M/12/1000
    T4H = 0xfc;
    T4T3M = 0xc0;                   //Set external counting mode and start timer
    IE2 = ET4;                      //Enable timer interrupt
    EA = 1;

    while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

T4T3M       DATA        0D1H
T4L         DATA        0D3H
T4H         DATA        0D2H
T3L         DATA        0D5H
T3H         DATA        0D4H
T2L         DATA        0D7H
T2H         DATA        0D6H
AUXR        DATA        8EH
IE2         DATA        0AFH
ET2         EQU         04H
ET3         EQU         20H
ET4         EQU         40H
AUXINTIF    DATA        0EFH
T2IF        EQU         01H
T3IF        EQU         02H
T4IF        EQU         04H

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         00A3H
            LJMP        TM4ISR

            ORG         0100H
TM4ISR:
            CPL         P1.0                        ;Test port
```

```
                RETI

MAIN:
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                MOV         T4L,#66H                    ;65536-11.0592M/12/1000
                MOV         T4H,#0FCH
                MOV         T4T3M,#0C0H                 ;Set external counting mode and start timer
                MOV         IE2,#ET4                    ;Enable timer interrupt
                SETB        EA

                JMP         $

                END
```

13.5.29 Timer 4 (Divided clock output)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         T4T3M       =       0xd1;
sfr         T4L         =       0xd3;
sfr         T4H         =       0xd2;
sfr         T3L         =       0xd5;
sfr         T3H         =       0xd4;
sfr         T2L         =       0xd7;
sfr         T2H         =       0xd6;

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

void main()
```

```
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P3M1 = 0x00;

        T4L = 0x66;                                         //65536-11.0592M/12/1000
        T4H = 0xfc;
        T4T3M = 0x90;                                       //Enable clock output and start timer

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

T4T3M           DATA            0D1H
T4L             DATA            0D3H
T4H             DATA            0D2H
T3L             DATA            0D5H
T3H             DATA            0D4H
T2L             DATA            0D7H
T2H             DATA            0D6H

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN

                ORG             0100H
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                MOV         T4L,#66H                    ;65536-11.0592M/12/1000
                MOV         T4H,#0FCH
                MOV         T4T3M,#90H                  ;Enable clock output and start timer

                JMP         $

                END
```

13.5.30 Timer 4 is used as baud rate generator of UART4

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (65536 - FOSC / 115200 / 4)

sfr         T4T3M           =       0xd1;
sfr         T4L             =       0xd3;
sfr         T4H             =       0xd2;
sfr         T3L             =       0xd5;
sfr         T3H             =       0xd4;
sfr         T2L             =       0xd7;
sfr         T2H             =       0xd6;
sfr         S4CON           =       0x84;
sfr         S4BUF           =       0x85;
sfr         IE2             =       0xaf;

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

bit         busy;
char        wptr;
char        rptr;
char        buffer[16];

void Uart4Isr() interrupt 18
{
    if (S4CON & 0x02)
    {
```

```
                S4CON &= ~0x02;
                busy = 0;
        }
        if (S4CON & 0x01)
        {
                S4CON &= ~0x01;
                buffer[wptr++] = S4BUF;
                wptr &= 0x0f;
        }
}

void Uart4Init()
{
        S4CON = 0x50;
        T4L = BRT;
        T4H = BRT >> 8;
        T4T3M = 0xa0;
        wptr = 0x00;
        rptr = 0x00;
        busy = 0;
}

void Uart4Send(char dat)
{
        while (busy);
        busy = 1;
        S4BUF = dat;
}

void Uart4SendStr(char *p)
{
        while (*p)
        {
                Uart4SEND(*p++);
        }
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        Uart4Init();
        IE2 = 0x10;
        EA = 1;
        Uart4SENDStr("Uart Test !\r\n");

        while (1)
        {
                if (rptr != wptr)
                {
                        Uart4SEND(buffer[rptr++]);
                        rptr &= 0x0f;
                }
        }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

T4T3M           DATA            0D1H
T4L             DATA            0D3H
T4H             DATA            0D2H
T3L             DATA            0D5H
T3H             DATA            0D4H
T2L             DATA            0D7H
T2H             DATA            0D6H
S4CON           DATA            84H
S4BUF           DATA            85H
IE2             DATA            0AFH

BUSY            BIT             20H.0
WPTR            DATA            21H
RPTR            DATA            22H
BUFFER          DATA            23H                             ;16 bytes

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0093H
                LJMP            UART4_ISR

                ORG             0100H

UART4_ISR:
                PUSH            ACC
                PUSH            PSW
                MOV             PSW,#08H

                MOV             A,S4CON
                JNB             ACC.1,CHKRI
                ANL             S4CON,#NOT 02H
                CLR             BUSY
CHKRI:
                JNB             ACC.0,UART4ISR_EXIT
                ANL             S4CON,#NOT 01H
                MOV         A,WPTR
                ANL         A,#0FH
                ADD         A,#BUFFER
                MOV         R0,A
                MOV         @R0,S4BUF
                INC         WPTR
UART4ISR_EXIT:
                POP         PSW
                POP         ACC
                RETI

UART4_INIT:
                MOV         S4CON,#50H
                MOV         T4L,#0E8H                  ;65536-11059200/115200/4=0FFE8H
                MOV         T4H,#0FFH
                MOV         T4T3M,#0A0H
                CLR         BUSY
                MOV         WPTR,#00H
                MOV         RPTR,#00H
                RET

UART4_SEND:
                JB          BUSY,$
                SETB        BUSY
                MOV         S4BUF,A
                RET

UART4_SENDSTR:
                CLR         A
                MOVC        A,@A+DPTR
                JZ          SEND4END
                LCALL       UART4_SEND
                INC         DPTR
                JMP         UART4_SENDSTR
SEND4END:
                RET

MAIN:
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                LCALL       UART4_INIT
                MOV         IE2,#10H
                SETB        EA

                MOV         DPTR,#STRING
                LCALL       UART4_SENDSTR
```

```
LOOP:
                MOV             A,RPTR
                XRL             A,WPTR
                ANL             A,#0FH
                JZ              LOOP
                MOV             A,RPTR
                ANL             A,#0FH
                ADD             A,#BUFFER
                MOV             R0,A
                MOV             A,@R0
                LCALL           UART4_SEND
                INC             RPTR
                JMP             LOOP

STRING:         DB              'Uart Test !',0DH,0AH,00H

                END
```

14 UART Communication

Product line	Number of UART
STC8H1K08 family	**2**
STC8H1K28 family	**2**
STC8H3K64S4 family	**4**
STC8H3K64S2 family	**2**
STC8H8K64U family	**4**
STC8H2K64T family	**4**
STC8H4K64TLR family	**4**
STC8H4K64TLCD family	**4**
STC8H4K64LCD family	**4**

There are 4 full duplex asynchronous serial communication ports (UART in short) in STC8H series of microcontrollers. Each UART consists of two data buffers, a shift register, a serial control register and a baud rate generator. Each UART data buffer consists of two independent receive and transmit buffers, which can transmit and receive data simultaneously.

There are 4 modes for UART1 of STC8H series of microcontrollers, the baud rates of two modes of them are variable, and the baud rates of the other two modes are fixed. They can be chosen for different applications. There are only two modes in UART2, UART3 and UART4, and their baud rates are variable. Different baud rates and different modes can be set by software. It is flexible for the host to query the receiving or sending process, or use the interrupt method.

All the pins of UART1, UART2, UART3 and UART4 can be switched amoung multiple groups of ports using the pin switching function, so that a serial port can be multiplexed into severial serial ports in a time-sharing manner.

14.1 Registers Related to UARTs

Symbol	Description	Address	Bit Address and Symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
SCON	UART1 control register	98H	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	0000,0000
SBUF	UART1 data buffer register	99H									0000,0000
S2CON	UART2 control register	9AH	S2SM0	-	S2SM2	S2REN	S2TB8	S2RB8	S2TI	S2RI	0100,0000
S2BUF	UART2 data buffer registe	9BH									0000,0000
S3CON	UART3 control register	ACH	S3SM0	S3ST3	S3SM2	S3REN	S3TB8	S3RB8	S3TI	S3RI	0000,0000
S3BUF	UART3 data buffer register	ADH									0000,0000
S4CON	Serial port 4 control register	84H	S4SM0	S4ST4	S4SM2	S4REN	S4TB8	S4RB8	S4TI	S4RI	0000,0000
S4BUF	Serial port 4 data buffer register	85H									0000,0000
PCON	Power control register	87H	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000
AUXR	Auxiliary register 1	8EH	T0x12	T1x12	UART_M0x6	T2R	T2_C/T	T2x12	EXTRAM	S1ST2	0000,0001
SADDR	UART1 slave address register	A9H									0000,0000
SADEN	UART1 slave address enable register	B9H									0000,0000

14.2 UART1

14.2.1 UART1 control register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
SCON	98H	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI

SM0/FE: If the SMOD0 bit in the PCON register is 1, this bit is the frame error detection flag. When the UART detects an invalid stop bit during reception, it is set by the UART receiver and must be cleared by software. If SMOD0 bit in PCON register is 0, this bit and SM1 specify the communication mode of UART1 as shown in the following table:

SM0	SM1	Mode of UART1	Function description
0	0	Mode 0	synchronous shift serial mode
0	1	Mode 1	8-bit UART, whose baud-rate is variable
1	0	Mode 2	9-bit UART, whose baud-rate is fixed
1	1	Mode 3	9-bit UART, whose baud-rate is variable

SM2: Mode 2 or mode 3 multi-machine communication enable control bit. When UART1 adopts mode 2 or mode 3, if the SM2 bit is 1 and the REN bit is 1, the receiver is in the Address Frame Filter state. In this case, the received 9th bit (RB8) can be used to filter the address frame. If RB8 = 1, it indicates that the frame is an address frame, the address information can enter SBUF and set RI bit. The address information is compared in the interrupt service routine. If RB8 = 0, it indicates that the frame is not an address frame, which should be discarded and keep RI = 0. In mode 2 or mode 3, if the SM2 bit is 0 and the REN bit is 1, the receiver is in a state where the address frame filtering is disabled. The received message can enter SBUF regardless of whether RB8 is 0 or 1, and make RI = 1. Here, RB8 is usually used as a check bit. Mode 1 and mode 0 are non-multi-machine communication modes. In these two modes, SM2 should be set to 0.

REN: Receive enable control bit.

0: disable UART1 receive data.

1: enable UART1 receive data.

TB8: The 9th bit be transmitted for UART1 in mode 2 and 3. It can be set or cleared by software. It is not used in mode 0 and mode 1.

RB8: The 9th bit received for UART1 in mode 2 and 3 which is usually used as a check bit or address frame/data frame flag. It is not used in mode 0 and mode 1.

TI: Transmit interrupt request flag of UART1. In mode 0, when the ransmission of the 8th bit completes, TI is set by the hardware automatically and requests the interrupt to the CPU. After the CPU responds the interrupt, TI must be cleared by software. In other modes, TI is set by the hardware automatically at the start of the stop bit transmittion and requests interrupts to the CPU. TI must be cleared by software after the interrupt is responded.

RI: Receive interrupt request flag of UART1. In mode 0, when the serial port receives the 8th bit of datum, RI is set by the hardware automatically and requests interrupt to the CPU. After the interrupt is responded, RI must be cleared by software. In other modes, RI is set by hardware automatically at the middle of stop bit the serial port received, and requests the interrupt to the CPU. After the interrupt is responded, RI must be cleared by software.

14.2.2 UART1 data register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
SBUF	99H								

SBUF: It is used as the buffer in transmission and receiving of UART1. SBUF is actually two buffers, reading buffer and writing buffer. Two operations correspond to two different registers, one is write-only register (writing buffer), the other is read-only register (reading buffer). In fact, the CPU reads serial receive buffer when reads SBUF. When CPU writes to the SBUF will trigger the serial port to start sending data.

14.2.3 Power control register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
PCON	87H	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL

SMOD: double Baud rate of UART1 control bit.
0: disable double baud rate of the UART1.
1: enable double baud rate of the UART1.

SMOD0: Frame error detection control bit.
0: No frame error detection function, SCON.7 is SM0 function.
1: enable frame error detection function. The function of SM0/FE is FE.

14.2.4 Auxiliary register 1

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
AUXR	8EH	T0x12	T1x12	UART_M0x6	T2R	T2_C/T	T2x12	EXTRAM	S1ST2

UART_M0x6: Baud rate select bit of UART1 while it works in mode 0.
0: The baud-rate of UART in mode 0 is SYSclk/12.
1: The baud-rate of UART in mode 0 is SYSclk/2.

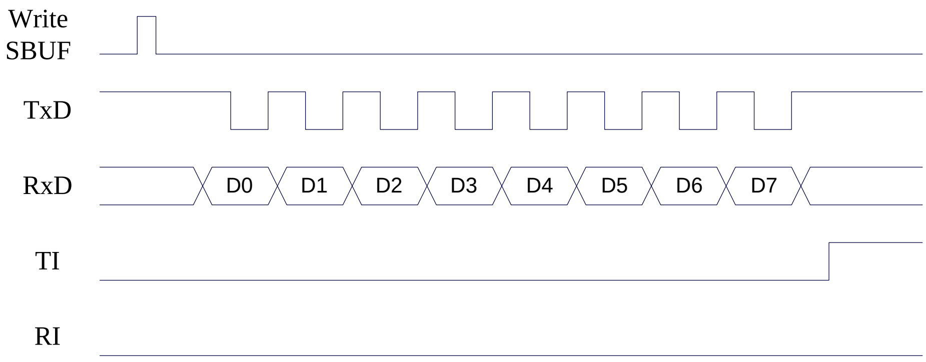
S1ST2: UART1 baud rate generator select bit.
0: Select Timer 1 as the baud-rate generator of UART1.
1: Select Timer 2 as the baud-rate generator of UART1.

14.2.5 UART1 Mode 0

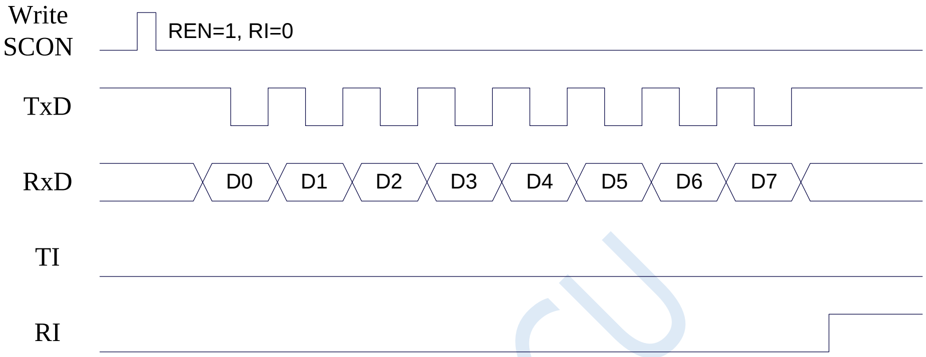
When mode 0 is selected for UART1, UART1 operates in synchronous shift register mode. When the serial port mode 0 communication speed setting bit UART_M0x6 is 0, the baud rate is fixed to SYSclk/12. When UART_M0x6 is 1, the baud rate is fixed to SYSclk/2. RxD is used as serial communication data pin, TxD is used as synchronous shift pulse output pin. 8-bit data are transmitted and received, LSB first.

Transmission process of mode 0: Transmission is initiated by any instruction that write data to SBUF. The 8-bit datum is output from the RxD pin at the baud rate of SYSclk/12 or SYSclk/2 (determined by the UART_M0x6 divided by 12 or 2), from LSB to MSB. The TxD pin outputs the synchronous shift pulse signal. The interrupt flag TI will be set when transmittion is completed. When the write signal is valid, the transmit control signal SEND is active (high) one clock apart, allowing RxD to send data while allowing the TxD output the synchronous shift pulse. When a frame (8 bits) of datum is sent, all control signals are reset to the original status, and only TI keeps high level and keeps the interrupt request status. TI must be cleared by software before sending data again.

Receiving process of mode 0: Receiving is initiated by setting REN and the receive interrupt request flag RI=0. After starting the receive process, RxD is the serial data input pin and TxD is the synchronous pulse output pin. The serial receiving baud rate is SYSclk/12 or SYSclk/2 (determined by UART_M0x6 is 12 or 2). After receiving a frame of datum (8 bits), the control signal is reset and the interrupt flag RI is set to 1, the interrupt requet status appears. RI must be cleared by software for the next receiving data.



Transmitting data (UART1 mode 0)



Receiving data (UART1 mode 0)

In mode 0, SM2 must be cleared so that TB8 and RB8 bits are not affected. Since the baud rate is fixed at SYSclk/12 or SYSclk/2, no timer is required and the clock of the microcontroller is used as the synchronous shift pulse directly.

The baud rates of UART1 mode 0 are shown in the following table, where SYSclk is the system operating frequency:

UART_M0x6	Baud rate calculation formula
0	$\text{Baud rate} = \frac{\text{SYSclk}}{12}$
1	$\text{Baud rate} = \frac{\text{SYSclk}}{2}$

14.2.6 UART1 Mode 1

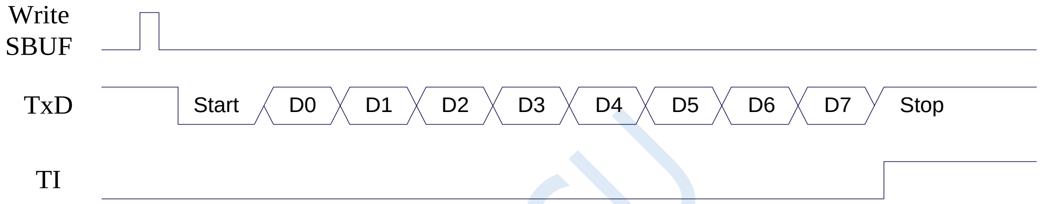
If SM0 and SM1 of SCON are set to ‘01’ by the software, UART1 will work in mode 1, which is a 8-bit UART mode. In mode 1, a frame of information consists 10 bits: 1 start bit, 8 data bits (LSB first) and 1 stop bit. The baud rate is variable, which can be set by the software as needed. TxD is the data transmitting pin, and RxD is the data receiving pin, the UART is a full duplex receiver/transmitter.

Transmission process of mode 1: TxD is used as data output pin when transmitting a datum. Transmission is initiated by writing SBUF. "1" is also written into the 9th bit of transmission shift register by the writing "SBUF" signal, and the TX control unit is notified to start sending. The shift register shifts the data right to TxD to send, and shifts "0" in the left to supplement. When the highest bit of data is shifted to the output of the shift register, it is followed by the 9th bit "1", and all bits to the left of it are "0". This state causes the TX control unit to make the last shift output, and then disables the transmission signal "SEND" to complete the transmission of a frame and sets TI, and requests interrupt processing to CPU.

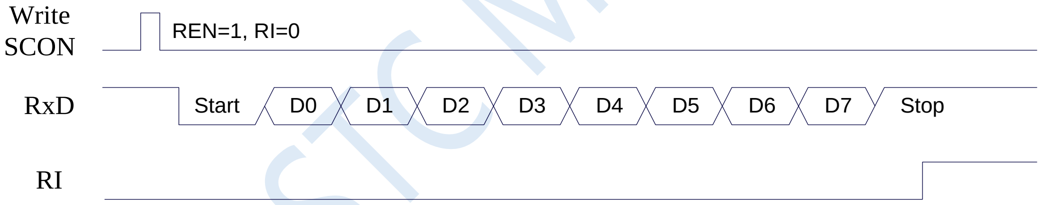
Receiving process of mode 1: After the software sets the reception enable flag REN, i.e. REN = 1, the receiver will detect the RxD pin signal. The receiver is ready to receive data when a "1" → "0" falling edge is detected at RxD pin, and resets the receiving counter of the baud rate generator immediately, loads 1FFH into the shift register. The received datum is shifted in from the right of the receiving shift register, the loaded 1FFH is shifted out to the left. When the start bit "0" is shifted to the far left of the shift register, the RX controller shifts for the last time and completes a frame receiving. The received datum is valid only if the following two conditions are met:

- RI=0;
- SM2=0 or the stop bit received is 1.

The datum received is loaded into SBUF, the stop bit is loaded into RB8, RI flag is set to request interrupt to CPU. If the two conditions can not be met at the same time, the received data is invalid and is discarded. Regardless of the conditions are met or not, the receiver will re-test RxD pin of the "1" → "0" edge, and continue to receive the next frame. If the received datum is valid, the RI flag must be cleared by software in the interrupt service routine. Usually, SM2 is set to "0" when serial port is operating in mode 1.



Transmitting data (UART1 mode 1)



Receiving data (UART1 mode 1)

The baud rate of UART1 is variable. It can be generated by T1 or T2. If the timer is in 1T mode (12x speed), the corresponding baud rate is increased by 12 times.

The baud rate of UART1 mode 1 is calculated as follows, where SYSclk is the system operating frequency.

Timer selected	Speed of timer	Baud rate calculation formula
T2	1T	$\text{reload value □f T2} = 65536 - \frac{\text{SYSclk}}{4 \times \text{baud rate}}$
	12T	$\text{reload value of □imer 2} = 65536 - \frac{\text{SYSclk}}{12 \times 4 \times \text{baud rate}}$
T1 mode 0	1T	$\text{reload value □f T1} = 65536 - \frac{\text{SYSclk}}{4 \times \text{baud rate}}$
	12T	$\text{reload value □f T1} = 65536 - \frac{\text{SYSclk}}{12 \times 4 \times \text{baud rate}}$
T1 mode 2	1T	$\text{reload value of T1} = 256 - \frac{2^{SMOD} \times \text{SYSclk}}{32 \times \text{baud rate}}$
	12T	$\text{reload value of T1} = 256 - \frac{2^{SMOD} \times \text{SYSclk}}{12 \times 32 \times \text{baud rate}}$

The reload value of the timers corresponding to the common frequency and the common baud rate are as following.

Frequency (MHz)	Baud rate	T2		T1 mode 0		T1 mode 2			
		1T mode	12T mode	1T mode	12T mode	SMOD=1		1T mode	
						1T mode	12T mode	1T mode	12T mode
11.0592	115200	FFE8H	FFFEH	FFE8H	FFFEH	FAH	-	FDH	-
	57600	FFD0H	FFFCH	FFD0H	FFFCH	F4H	FFH	FAH	-
	38400	FFB8H	FFFAH	FFB8H	FFFAH	EEH	-	F7H	-
	19200	FF70H	FFF4H	FF70H	FFF4H	DCH	FDH	EEH	-
	9600	FEE0H	FFE8H	FEE0H	FFE8H	B8H	FAH	DCH	FDH
18.432	115200	FFD8H	-	FFD8H	-	F6H	-	FBH	-
	57600	FFB0H	-	FFB0H	-	ECH	-	F6H	-
	38400	FF88H	FFF6H	FF88H	FFF6H	E2H	-	F1H	-
	19200	FF10H	FFECH	FF10H	FFECH	C4H	FBH	E2H	-
	9600	FE20H	FFD8H	FE20H	FFD8H	88H	F6H	C4H	FBH
22.1184	115200	FFD0H	FFFCH	FFD0H	FFFCH	F4H	FFH	FAH	-
	57600	FFA0H	FFF8H	FFA0H	FFF8H	E8H	FEH	F4H	FFH
	38400	FF70H	FFF4H	FF70H	FFF4H	DCH	FDH	EEH	-
	19200	FEE0H	FFE8H	FEE0H	FFE8H	B8H	FAH	DCH	FDH
	9600	FDC0H	FFD0H	FDC0H	FFD0H	70H	F4H	B8H	FAH

14.2.7 UART1 Mode 2

If the two bits of SM0 and SM1 are ‘10’, UART1 operates in mode 2. UART1 operating in mode 2 is a 9-bit data asynchronous communication UART. One frame of data consists of 11 bits: 1 start bit, 8 data bits (LSB first), 1 programmable bit (9th bit) and 1 stop bit. The transmitted programmable bit (9th bit) is supplied by TB8 in SCON, which can be confugred as either 1 or 0 by software. Or, the odd/even parity bit P in the PSW can be loaded into TB8. Not only can TB8 be used as a multi-machine communication address/data flag, but also it can be used as datum parity check bit. The 9th bit is received into RB8 of SCON. TxD is the transmitting pin, and RxD is the receiving pin, the serial port is a full duplex receiver/transmitter.

The baud rate of mode 2 is fixed to the system clock divided by 64 or 32 depending on the value of SMOD in PCON.

The baud rate of UART1 mode 2 is shown in the following table, where SYSclk is the system operating frequency.

SMOD	Baud rate calculation formula
0	$\text{baud rate} = \frac{\text{SYSclk}}{64}$
1	$\text{baud rate} = \frac{\text{SYSclk}}{32}$

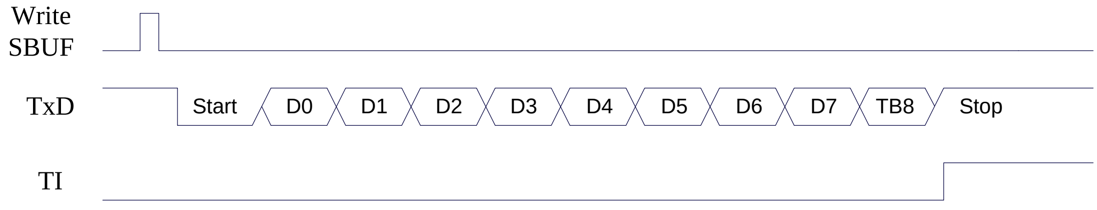
Except that the source of the baud rate is slightly different, and the 9th bit of the shift register supplied by TB8 while being sent is different, the functional and structure of mode 2 and mode 1 are basically the same, the receiving / sending operation and timing of mode 2 and mode 1are also basically the same.

After the receiver receives a frame of information, the following conditions must be met at the same time.

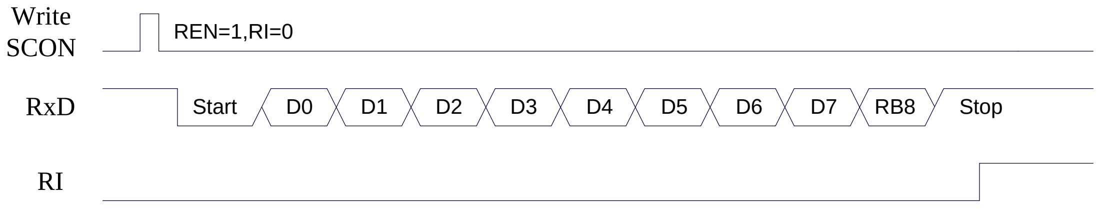
- RI=0
- SM2=0 or SM2=1 and the 9th bit received RB8=1.

Only when the two conditions above are met at the same time, the data received in shift register is loaded into SBUF and RB8. The RI flag is set to 1, and the interrupt is requested to CPU. If one of the above conditions is not met, the data received in the shift register is invalid and is discarded, and RI is not set. Regardless of the above conditions are met or not, the receiver begins to detect the RxD pin hopping information again to receive the next frame of information. In mode 2, the received stop bit is not related to SBUF, RB8 and RI.

It provides for the convenience of multi-machine communication by setting SM2, TB8 of SCON and communication protocol using the software.



Transmitting data (UART1 mode 2)



Receiving data (UART1 mode 2)

14.2.8 UART1 Mode 3

If the two bits of SM0 and SM1 are ‘11’, UART1 operates in mode 3. UART1 operating in mode 3 is a 9-bit data asynchronous communication UART. One frame of data consists of 11 bits: 1 start bit, 8 data bits (LSB first), 1 programmable bit (9th bit) and 1 stop bit. The transmitted programmable bit (9th bit) is supplied by TB8 in SCON, which can be confugred as either 1 or 0 by software. Or, the odd/even parity bit P in the PSW can be loaded into TB8. Not only can TB8 be used as a multi-machine communication address/data flag, but also it can be used as datum parity check bit. The 9th bit is received into RB8 of SCON. TxD is the transmitting pin, and RxD is the receiving pin, the serial port is a full duplex receiver/transmitter.

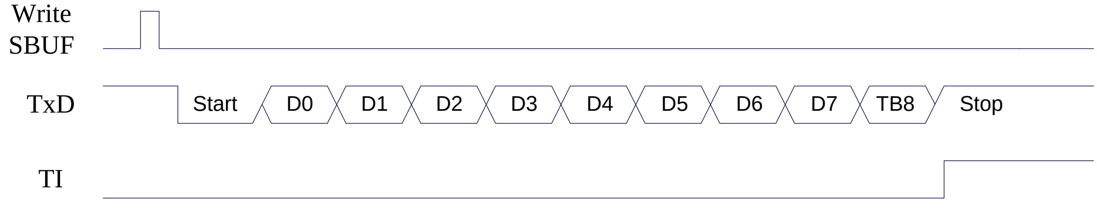
Except that the 9th bit of the shift register supplied by TB8 while being sent is different, the functional and structure of mode 3 and mode 1 are basically the same, the receiving / sending operation and timing of mode 3 and mode 1 are also basically the same.

After the receiver receives a frame of information, the following conditions must be met at the same time.

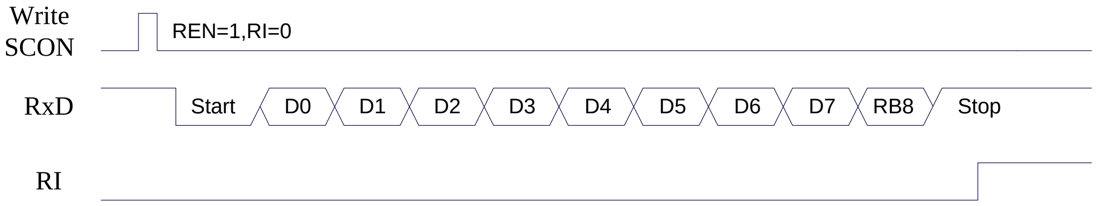
- RI=0
- SM2=0 or SM2=1 and the 9th bit received RB8=1.

Only when the two conditions above are met at the same time, the data received in shift register is loaded into SBUF and RB8. The RI flag is set to 1, and the interrupt is requested to CPU. If one of the above conditions is not met, the data received in the shift register is invalid and is discarded, and RI is not set. Regardless of the above conditions are met or not, the receiver begins to detect the RxD pin hopping information again to receive the next frame of information. In mode 3, the received stop bit is not related to SBUF, RB8 and RI.

It provides for the convenience of multi-machine communication by setting SM2, TB8 of SCON and communication protocol using the software.



Transmitting data (UART1 mode 3)



Receiving data (UART1 mode 3)

The baud rate calculation formula of UART1 mode 3 is exactly the same as that of mode 1. Please refer to the mode 1 baud rate calculation formula.

14.2.9 Automatic Address Recognition

14.2.10 UART1 slave address control registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
SADDR	A9H								
SADEN	B9H								

SADDR: Slave address register
SADEN: Slave address mask register

The automatic address recognition function is typically used in the field of multi-machine communications. Its main principle is that the slave system identifies the address information from the master serial port data stream through the hardware comparison function. The address of the slave is set by the registers SADDR and SADEN. The hardware filters the slave address automatically. The hardware will generate a serial port interrupt when the slave address information from the master matches the slave address set by the slave. Otherwise, the hardware will discard the serial port data automatically without any interruption. When a number of slaves in Idle mode are connected together, only the slave that matches the slave address will wake up from Idle mode. Then the power consumption of the slave MCU reduces greatly. Constantly entering the serial port interrupt which reduces the system execution efficiency can be avoided even if the slave is in normal operation.

To use the automatic address recognition feature of the serial port, mode 2 or mode 3 of the serial port of the MCU that participates in communication is selected. Usually the mode 3 with variable baud rate is selected because the baud rate of mode 2 is fixed, and it is inconvenient to adjust. SM2 bit of slave in SCON is set to 1. The 9th bit which is stored in RB8 is the address/data flag in mode 2 or 3. When the 9th bit is 1, it indicates the previous 8-bit datum stored in SBUF is the address information. If SM2 is set to 1, the slave MCU will filter out non-address data whose 9th bit is 0 automatically while the address data whose 9th bit is 1 in SBUF will automatically be matched with the address set in SADDR and SADEN. If the address matches, RI will be set to "1" and an interrupt will occur. Otherwise the received data is discarded.

The slave address is set by two registers, SADDR and SADEN. SADDR is the slave address register, where the slave address is stored. SADEN is the slave address mask register, which is used to set the ignore bit in the address information. The setting method is as follows.

For example
SADDR = 11001010
SADEN = 10000001
Then the matched address is 1xxxxxx0
That is, as long as bit 0 is 0 and bit 7 is 1 in the address data sent by the master, the address can be matched with the local address.
Another example
SADDR = 11001010
SADEN = 00001111
Then the matched address is xxxx1010
That is, as long as the low 4 bits are 1010 in the address data sent by the master, the address can be

matched with the local address. The high 4 bits can be any value and are ignored.

The Broadcast Address (FFH) can be used by the master to select all the slaves simultaneously for communication.

14.3 UART2

14.3.1 UART2 control register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
S2CON	9AH	S2SM0	-	S2SM2	S2REN	S2TB8	S2RB8	S2TI	S2RI

S2SM0: Serial port 2 mode select bit.

S2SM0	UART2 mode	Function description
0	Mode 0	8-bit UART, whose baud-rate is variable
1	Mode 1	9-bit UART, whose baud-rate is variable

S2SM2: UART2 multi-machine communication control enable bit. In mode 1, if the S2SM2 bit is 1 and the S2REN bit is 1, the receiver is in the address frame filter state. In this case, the received 9th bit (S2RB8) can be used to filter the address frame. If S2RB8 = 1, the frame is the address frame, address information can enter S2BUF, S2RI becomes 1, and then address can be compared in the interrupt service routine. If S2RB8 = 0, it indicates that the frame is not an address frame and should be discarded and keep S2RI = 0. In mode 1, if the S2SM2 bit is 0 and the S2REN bit is 1, the receiver is in the address frame filter disabled state. Regardless of the received S2RB8 is 0 or 1, the information received can enter into the S2BUF, and make S2RI = 1. Here, S2RB8 is usually used as check bit. Mode 0 is non-multi-machine communication mode, where S2SM2 should be 0.

S2REN: Receive enable control bit.

0: disable UART2 receive data.

1: enable UART2 receive data.

S2TB8: S2TB8 is the 9th bit of datum to be sent when UART2 is in mode 1, which is usually used as a parity check bit or an address frame / data frame flag. It can be set or cleared by software as required. In mode 0, this bit is not used.

S2RB8: S2RB8 is the 9th bit of datum recieved when UART2 is in mode 1, which is usually used as a parity check bit or an address frame / data frame flag. It can be set or cleared by software as required. In mode 0, this bit is not used.

S2TI: Transmit interrupt request flag of UART2. S2TI is set by the hardware automatically at the beginning of the stop bit transmittion and requests interrupts to the CPU. S2TI must be cleared by software after the interrupt is responded.

S2RI: Receive interrupt request flag of UART2. S2RI is set by hardware automatically at the middle of stop bit received, and requests the interrupt to the CPU. After the interrupt is responded, S2RI must be cleared by software.

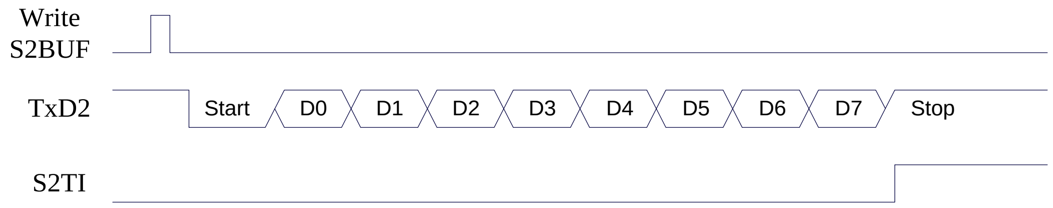
14.3.2 UART2 data register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
S2BUF	9BH								

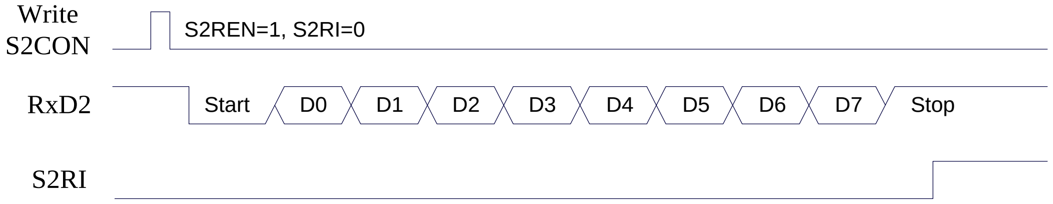
S2BUF: It is used as the buffer in transmission and receiving for UART2. S2BUF is actually two buffers, reading buffer and writing buffer. Two operations correspond to two different registers, one is write-only register (writing buffer), the other is read-only register (reading buffer). The CPU reads serial receiving buffer when reads S2BUF, and writes to the S2BUF will trigger the serial port to start sending data.

14.3.3 UART2 Mode 0

Serial port 2 mode 0 is 8-bit UART mode with ariable baud rate. In this mode, a frame of data consists 10 bits: 1 start bit, 8 data bits (LSB first) and 1 stop bit. The baud rate is variable, which can be set by the software as needed. TxD2 is the data transmitting pin, and RxD2 is the data receiving pin, the serial port is a full duplex receiver/transmitter.



Transmitting data (UART2 mode 0)



Receiving data (UART2 mode 0)

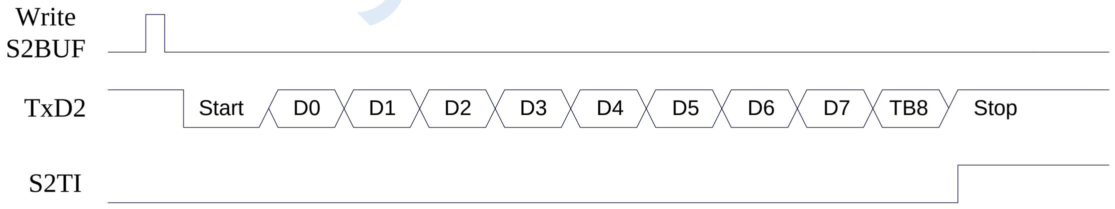
The baud rate of UART2 is variable. It is generated by T2. If the timer is in 1T mode (12x speed), the corresponding baud rate is increased by 12 times.

The baud rate of UART2 mode 0 is calculated as follows, where SYSclk is the system operating frequency.

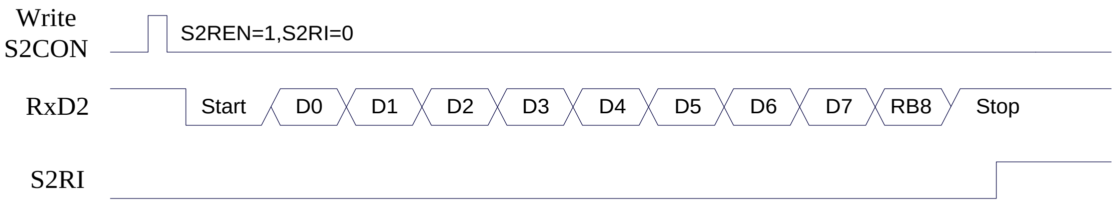
Timer selected	Speed of timer	Baud rate calculation formula
T2	1T	$\text{reload value of timer 2} = 65536 - \frac{SYSclk}{4 \times \text{baud rate}}$
	12T	$\text{reload value of timer 2} = 65536 - \frac{SYSclk}{12 \times 4 \times \text{baud rate}}$

14.3.4 UART2 Mode 1

UART2 operating in mode 1 is a 9-bit data UART mode with variable baud rate. One frame data consists of 11 bits: 1 start bit, 8 data bits (LSB first), 1 programmable bit (9th bit) and 1 stop bit. The baud rate is variable, which can be set by the software as needed. TxD2 is the data transmitting pin, and RxD2 is the data receiving pin, the serial port is a full duplex receiver/transmitter.



Transmitting data (UART2 mode 1)



Receiving data (UART2 mode 1)

The baud rate calculation formula of UART2 mode 1 is exactly the same as that of mode 0. Please refer to the mode 0 baud rate calculation formula.

14.4 UART3

14.4.1 UART3 control register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
S3CON	ACH	S3SM0	S3ST3	S3SM2	S3REN	S3TB8	S3RB8	S3TI	S3RI

S3SM0: UART3 mode select bit.

S3SM0	UART3 mode	Function description
0	Mode 0	8-bit UART, whose baud-rate is variable
1	Mode 1	9-bit UART, whose baud-rate is variable

S3ST3: UART3 baud rate generator select bit.

0: Select T2 as the baud-rate generator of UART3.

1: Select T3 as the baud-rate generator of UART3.

S3SM2: UART3 multi-machine communication control bit. In mode 1, if the S3SM2 bit is 1 and the S3REN bit is 1, the receiver is in the address frame filter state. In this case, the received 9th bit (S3RB8) can be used to filter the address frame. If S3RB8 = 1, the frame is the address frame, address information can enter S3BUF, S3RI becomes 1, and then the address is compared with the slave address in the interrupt service routine. If S3RB8 = 0, it indicates that the frame is not an address frame and should be discarded and keep S3RI = 0. In mode 1, if the S3SM2 bit is 0 and the S3REN bit is 1, the receiver is in the address frame filter disabled state. Regardless of the received S3RB8 is 0 or 1, the information received can enter into the S3BUF, and make S3RI = 1. Here, S3RB8 is usually used as parity check bit. Mode 0 is non-multi-machine communication mode, where S3SM2 should be 0.

S3REN: Receive enable control bit.

0: disable UART3 receive data.

1: enable UART3 receive data.

S3TB8: S3TB8 is the 9th bit of datum to be sent when UART3 is in mode 1, which is usually used as a parity check bit or an address frame / data frame flag. It can be set or cleared by software as required. In mode 0, this bit is not used.

S3RB8: S3RB8 is the 9th bit of datum recieved when UART3 is in mode 1, which is usually used as a parity check bit or an address frame / data frame flag. It can be set or cleared by software as required. In mode 0, this bit is not used.

S3TI: Transmit interrupt request flag of UART3. S3TI is set by the hardware automatically at the beginning of the stop bit transmittion and requests interrupt to the CPU. S3TI must be cleared by software after the interrupt is responded.

S3RI: Receive interrupt request flag of UART3. S3RI is set by hardware automatically at the middle of stop bit the serial port received, and requests interrupt to the CPU. After the interrupt is responded, S3RI must be cleared by software.

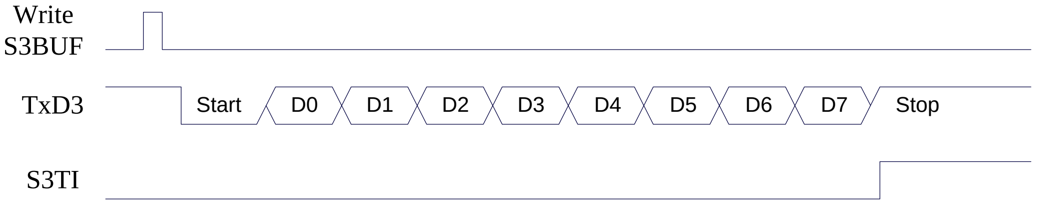
14.4.2 UART3 data register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
S3BUF	ADH								

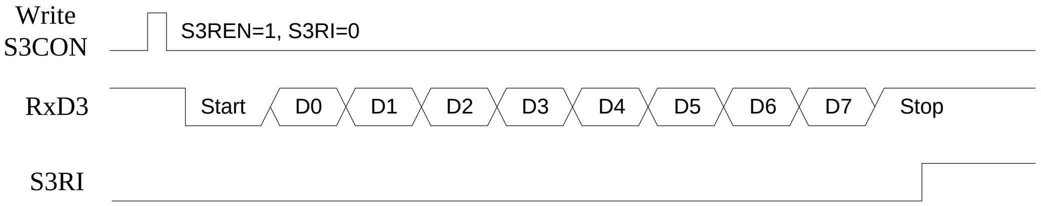
S3BUF: It is used as the buffer in transmission and receiving for UART3. S3BUF is actually two buffers, reading buffer and writing buffer. Two operations correspond to two different registers, one is write-only register (writing buffer), the other is read-only register (reading buffer). The CPU reads serial receive buffer when reads S3BUF, and writes to the S3BUF will trigger the serial port to start sending data.

14.4.3 UART3 Mode 0

UART3 mode 0 is a 8-bit UART mode with variable baud rate, where a frame of data consists 10 bits: 1 start bit, 8 data bits (LSB first) and 1 stop bit. The baud rate is variable, which can be set by the software as needed. TxD3 is the data transmitting pin, and RxD3 is the data receiving pin, the serial port is a full duplex receiver/transmitter.



Transmitting data (UART3 mode 0)



Receiving data (UART3 mode 0)

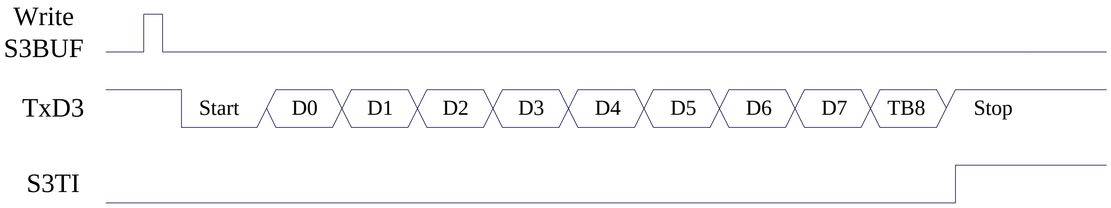
The baud rate of UART3 is variable. It is generated by T2 or T3. If the timer is in 1T mode (12x speed), the corresponding baud rate is increased by 12 times.

The baud rate of UART3 mode 0 is calculated as follows, where SYSclk is the system operating frequency.

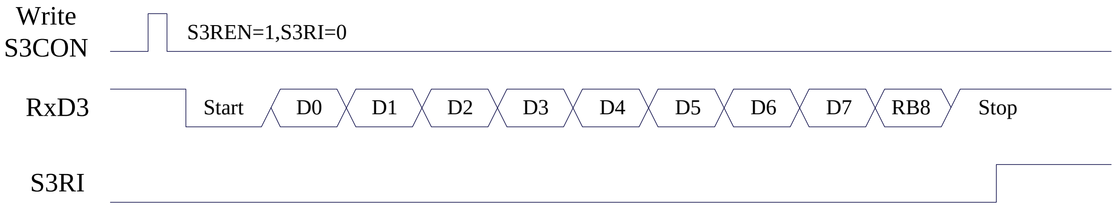
Timer selected	Speed of timer	Baud rate calculation formula
T2	1T	$\text{reload value of T2} = 65536 - \frac{SYSclk}{4 \times \text{baud rate}}$
	12T	$\text{reload value of T2} = 65536 - \frac{SYSclk}{12 \times 4 \times \text{baud rate}}$
T3	1T	$\text{reload value of T3} = 65536 - \frac{SYSclk}{4 \times \text{baud rate}}$
	12T	$\text{reload value of T3} = 65536 - \frac{SYSclk}{12 \times 4 \times \text{baud rate}}$

14.4.4 UART3 Mode 1

UART3 operating in mode 1 is a 9-bit data UART mode with variable baud rate. One frame data consists of 11 bits: 1 start bit, 8 data bits (LSB first), 1 programmable bit (9th bit) and 1 stop bit. The baud rate is variable, which can be set by the software as needed. TxD3 is the data transmitting pin, and RxD3 is the data receiving pin, the serial port is a full duplex receiver/transmitter.



Transmitting data (UART3 mode 1)



Receiving data (UART3 mode 1)

The baud rate calculation formula of UART3 mode 1 is exactly the same as that of mode 0. Please refer to the mode 0 baud rate calculation formula.

14.5 UART4

14.5.1 UART4 control register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
S4CON	84H	S4SM0	S4ST4	S4SM2	S4REN	S4TB8	S4RB8	S4TI	S4RI

S4SM0: UART4 mode select bit.

S4SM0	UART4 mode	Function description
0	Mode 0	8-bit UART, whose baud-rate is variable
1	Mode 1	9-bit UART, whose baud-rate is variable

S4ST4: UART4 baud rate generator select bit.

0: Select T2 as the baud-rate generator of UART4.

1: Select T4 as the baud-rate generator of UART4.

S4SM2: UART4 multi-machine communication control bit. In mode 1, if the S4SM2 bit is 1 and the S4REN bit is 1, the receiver is in the address frame filter state. In this case, the received 9th bit (S4RB8) can be used to filter the address frame. If S4RB8 = 1, the frame is the address frame, address information can enter S4BUF, S4RI becomes 1, and then address can be compared with the slave address in the interrupt service routine. If S4RB8 = 0, it indicates that the frame is not an address frame and should be discarded and keep S4RI = 0. In mode 1, if the S4SM2 bit is 0 and the S4REN bit is 1, the receiver is in the address frame filter disabled state. Regardless of the received S4RB8 is 0 or 1, the information received can enter into the S4BUF, and make S4RI = 1. Here, S4RB8 is usually used as parity check bit. Mode 0 is non-multi-machine communication mode, where S4SM2 should be 0.

S4REN: Receive enable control bit.

0: disable UART4 receive data.

1: enable UART4 receive data.

S4TB8: S4TB8 is the 9th bit of datum to be sent when UART4 is in mode 1, which is usually used as a parity check bit or an address frame / data frame flag. It can be set or cleared by software as required. In mode 0, this bit is not used.

S4RB8: S4RB8 is the 9th bit of datum recieved when UART4 is in mode 1, which is usually used as a parity check bit or an address frame / data frame flag. It can be set or cleared by software as required. In mode 0, this bit is not used.

S4TI: Transmit interrupt request flag of UART4. S4TI is set by the hardware automatically at the beginning of the stop bit transmittion and requests interrupt to the CPU. S4TI must be cleared by software after the interrupt is responded.

S4RI: Receive interrupt request flag of UART4. S4RI is set by hardware automatically at the middle of stop bit the serial port received, and requests interrupt to the CPU. After the interrupt is responded, S4RI must be cleared by software.

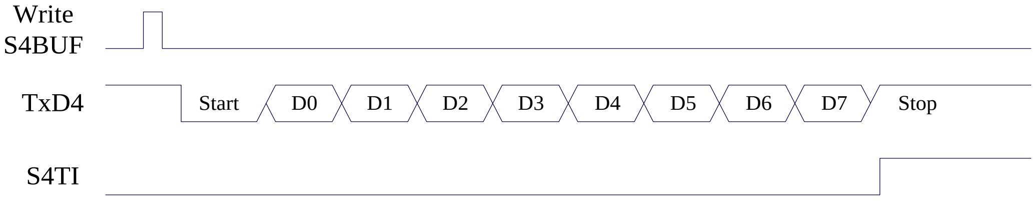
14.5.2 UART4 data register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
S4BUF	85H								

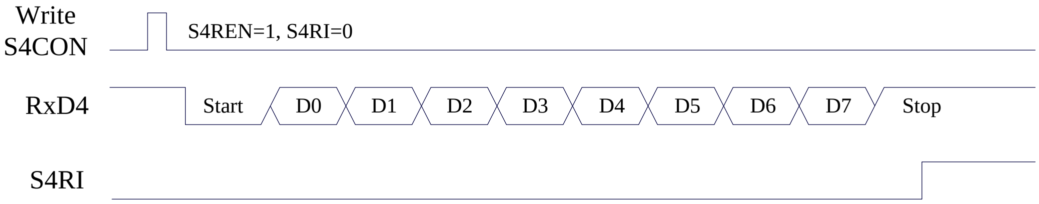
S4BUF: It is used as the buffer in transmission and receiving for UART4. S4BUF is actually two buffers, reading buffer and writing buffer. Two operations correspond to two different registers, one is write-only register (writing buffer), the other is read-only register (reading buffer). The CPU reads serial receive buffer when reads S4BUF, and writes to the S4BUF will trigger the serial port to start sending data.

14.5.3 UART4 Mode 0

UART4 mode 0 is an 8-bit UART mode with variable baud rate, where a frame of data consists 10 bits: 1 start bit, 8 data bits (LSB first) and 1 stop bit. The baud rate is variable, which can be set by the software as needed. TxD4 is the data transmitting pin, and RxD4 is the data receiving pin, the serial port is a full duplex receiver/transmitter.



Transmitting data (UART4 mode 0)



Receiving data (UART4 mode 0)

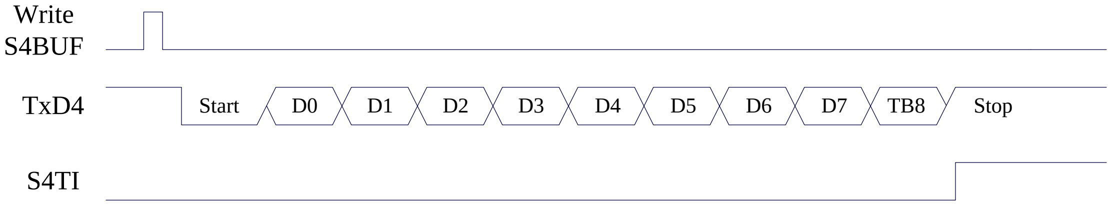
The baud rate of UART4 is variable. It is generated by T2 or T4. If the timer is in 1T mode (12x speed), the corresponding baud rate is increased by 12 times.

The baud rate of UART4 mode 0 is calculated as follows, where SYSclk is the system operating frequency.

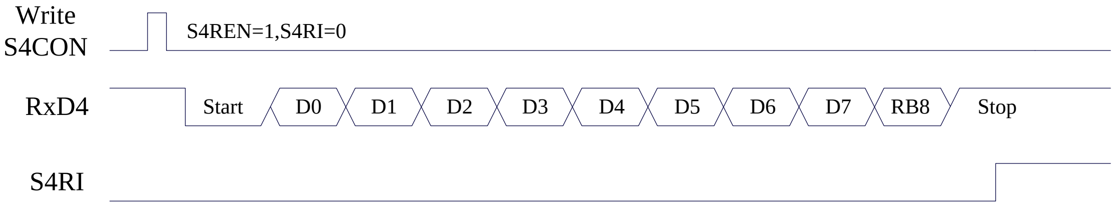
Timer selected	Speed of timer	Baud rate calculation formula
T2	1T	$\text{reload value of T2} = 65536 - \frac{\text{SYSclk}}{4 \times \text{baud rate}}$
	12T	$\text{reload value of T2} = 65536 - \frac{\text{SYSclk}}{12 \times 4 \times \text{baud rate}}$
T4	1T	$\text{reload value of T4} = 65536 - \frac{\text{SYSclk}}{4 \times \text{baud rate}}$
	12T	$\text{reload value of T4} = 65536 - \frac{\text{SYSclk}}{12 \times 4 \times \text{baud rate}}$

14.5.4 UART4 Mode 1

UART4 operating in mode 1 is a 9-bit data UART mode with variable baud rate. One frame data consists of 11 bits: 1 start bit, 8 data bits (LSB first), 1 programmable bit (9th bit) and 1 stop bit. The baud rate is variable, which can be set by the software as needed. TxD4 is the data transmitting pin, and RxD4 is the data receiving pin, the serial port is a full duplex receiver/transmitter.



Transmitting data (UART4 mode 1)



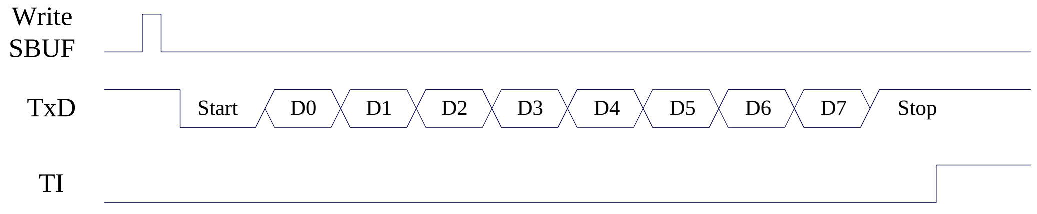
Receiving data (UART4 mode 1)

The baud rate calculation formula of UART4 mode 1 is exactly the same as that of mode 0. Please refer to the mode 0 baud rate calculation formula.

14.6 Precautions of UARTs

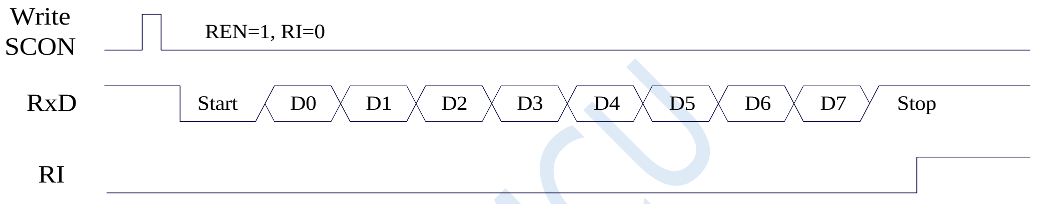
Regarding the UART interrupts requests, the following issues need to be noted. UART1, UART2, UART3, and UART4 are all similar, and serial port 1 is used as an example below.

In 8-bit data mode, TI interrupt request is generated after the entire stop bit is transmitted, as shown in the following figure:



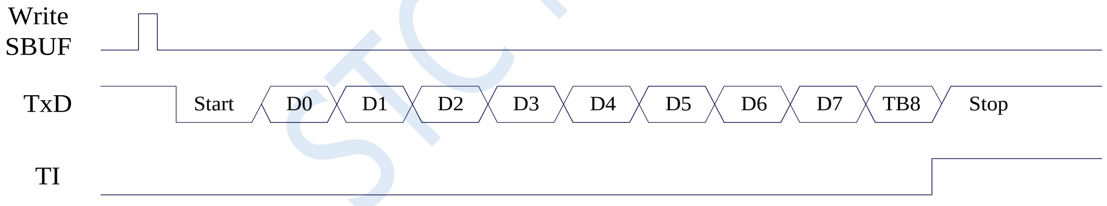
Transmission data (8-bit data)

In 8-bit data mode, RI interrupt request is generated after half of the stop bit is received, as shown in the following figure:



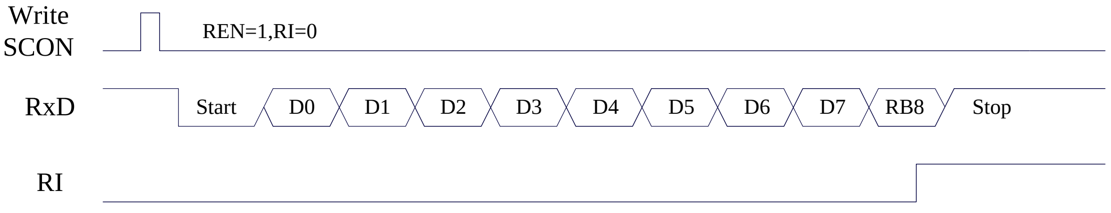
Receiving data (8-bit data)

In 9-bit data mode, TI interrupt request is generated after the entire 9th data bit is transmitted, as shown in the following figure:



Transmission data (9-bit data)

In 9-bit data mode, RI interrupt request is generated after receiving half of the 9th bit, as shown in the following figure:



Receiving data (9-bit data)

14.7 Example Routines

14.7.1 UART1 using T2 as baud rate generator

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (65536 - FOSC / 115200 / 4)

sfr         AUXR            =       0x8e;
sfr         T2H             =       0xd6;
sfr         T2L             =       0xd7;

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

bit         busy;
char        wptr;
char        rptr;
char        buffer[16];

void UartIsr() interrupt 4
{
        if (TI)
        {
                TI = 0;
                busy = 0;
        }
        if (RI)
        {
                RI = 0;
                buffer[wptr++] = SBUF;
                wptr &= 0x0f;
        }
}

void UartInit()
{
        SCON = 0x50;
        T2L = BRT;
        T2H = BRT >> 8;
        AUXR = 0x15;
        wptr = 0x00;
```

```
    rptr = 0x00;
    busy = 0;
}

void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}

void UartSendStr(char *p)
{
    while (*p)
    {
        UartSEND(*p++);
    }
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    UartInit();
    ES = 1;
    EA = 1;
    UartSENDStr(“Uart Test !\r\n”);

    while (1)
    {
        if (rptr != wptr)
        {
            UartSEND(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

AUXR        DATA        8EH
T2H         DATA        0D6H
T2L         DATA        0D7H

BUSY        BIT         20H.0
WPTR        DATA        21H
RPTR        DATA        22H
```

```
BUFFER          DATA            23H                             ;16 bytes

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0023H
                LJMP            UART_ISR

                ORG             0100H

UART_ISR:
                PUSH            ACC
                PUSH            PSW
                MOV             PSW,#08H

                JNB             TI,CHKRI
                CLR             TI
                CLR             BUSY
CHKRI:
                JNB             RI,UARTISR_EXIT
                CLR             RI
                MOV             A,WPTR
                ANL             A,#0FH
                ADD             A,#BUFFER
                MOV             R0,A
                MOV             @R0,SBUF
                INC             WPTR
UARTISR_EXIT:
                POP             PSW
                POP             ACC
                RETI

UART_INIT:
                MOV             SCON,#50H
                MOV             T2L,#0E8H                       ;65536-11059200/115200/4=0FFE8H
                MOV             T2H,#0FFH
                MOV             AUXR,#15H
                CLR             BUSY
                MOV             WPTR,#00H
                MOV             RPTR,#00H
                RET

UART_SEND:
                JB              BUSY,$
                SETB            BUSY
                MOV             SBUF,A
                RET
```

```
UART_SENDSTR:
                CLR             A
                MOVC            A,@A+DPTR
                JZ              SENDEND
                LCALL           UART_SEND
                INC             DPTR
                JMP             UART_SENDSTR
SENDEND:
                RET

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                LCALL           UART_INIT
                SETB            ES
                SETB            EA

                MOV             DPTR,#STRING
                LCALL           UART_SENDSTR

LOOP:
                MOV             A,RPTR
                XRL             A,WPTR
                ANL             A,#0FH
                JZ              LOOP
                MOV             A,RPTR
                ANL             A,#0FH
                ADD             A,#BUFFER
                MOV             R0,A
                MOV             A,@R0
                LCALL           UART_SEND
                INC             RPTR
                JMP             LOOP

STRING:         DB              'Uart Test !',0DH,0AH,00H

                END
```

14.7.2 UART1 using T1 (Mode 0) as baud rate generator

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
```

```
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (65536 - FOSC / 115200 / 4)

sfr         AUXR            =       0x8e;

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

bit         busy;
char        wptr;
char        rptr;
char        buffer[16];

void UartIsr() interrupt 4
{
        if (TI)
        {
                TI = 0;
                busy = 0;
        }
        if (RI)
        {
                RI = 0;
                buffer[wptr++] = SBUF;
                wptr &= 0x0f;
        }
}

void UartInit()
{
        SCON = 0x50;
        TMOD = 0x00;
        TL1 = BRT;
        TH1 = BRT >> 8;
        TR1 = 1;
        AUXR = 0x40;
        wptr = 0x00;
        rptr = 0x00;
        busy = 0;
}

void UartSend(char dat)
{
        while (busy);
        busy = 1;
        SBUF = dat;
}
```

```
void UartSendStr(char *p)
{
        while (*p)
        {
                UartSEND(*p++);
        }
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        UartInit();
        ES = 1;
        EA = 1;
        UartSENDStr("Uart Test !\r\n");

        while (1)
        {
                if (rptr != wptr)
                {
                        UartSEND(buffer[rptr++]);
                        rptr &= 0x0f;
                }
        }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

AUXR            DATA            8EH

BUSY            BIT             20H.0
WPTR            DATA            21H
RPTR            DATA            22H
BUFFER          DATA            23H                             ;16 bytes

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
```

```
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0023H
                LJMP            UART_ISR

                ORG             0100H

UART_ISR:
                PUSH            ACC
                PUSH            PSW
                MOV             PSW,#08H

                JNB             TI,CHKRI
                CLR             TI
                CLR             BUSY
CHKRI:
                JNB             RI,UARTISR_EXIT
                CLR             RI
                MOV             A,WPTR
                ANL             A,#0FH
                ADD             A,#BUFFER
                MOV             R0,A
                MOV             @R0,SBUF
                INC             WPTR
UARTISR_EXIT:
                POP             PSW
                POP             ACC
                RETI

UART_INIT:
                MOV             SCON,#50H
                MOV             TMOD,#00H
                MOV             TL1,#0E8H                       ;65536-11059200/115200/4=0FFE8H
                MOV             TH1,#0FFH
                SETB            TR1
                MOV             AUXR,#40H
                CLR             BUSY
                MOV             WPTR,#00H
                MOV             RPTR,#00H
                RET

UART_SEND:
                JB              BUSY,$
                SETB            BUSY
                MOV             SBUF,A
                RET

UART_SENDSTR:
                CLR             A
                MOVC            A,@A+DPTR
                JZ              SENDEND
                LCALL           UART_SEND
                INC             DPTR
                JMP             UART_SENDSTR
SENDEND:
                RET
```

```
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                LCALL           UART_INIT
                SETB            ES
                SETB            EA

                MOV             DPTR,#STRING
                LCALL           UART_SENDSTR

LOOP:
                MOV             A,RPTR
                XRL             A,WPTR
                ANL             A,#0FH
                JZ              LOOP
                MOV             A,RPTR
                ANL             A,#0FH
                ADD             A,#BUFFER
                MOV             R0,A
                MOV             A,@R0
                LCALL           UART_SEND
                INC             RPTR
                JMP             LOOP

STRING:         DB              'Uart Test !',0DH,0AH,00H

                END
```

14.7.3 UART1 using T1 (Mode 2) as baud rate generator

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (256 - FOSC / 115200 / 32)

sfr         AUXR            =       0x8e;

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

bit         busy;
char        wptr;
char        rptr;
char        buffer[16];

void UartIsr() interrupt 4
{
        if (TI)
        {
                TI = 0;
                busy = 0;
        }
        if (RI)
        {
                RI = 0;
                buffer[wptr++] = SBUF;
                wptr &= 0x0f;
        }
}

void UartInit()
{
        SCON = 0x50;
        TMOD = 0x20;
        TL1 = BRT;
        TH1 = BRT;
        TR1 = 1;
        AUXR = 0x40;
        wptr = 0x00;
        rptr = 0x00;
        busy = 0;
}

void UartSend(char dat)
{
        while (busy);
        busy = 1;
        SBUF = dat;
}

void UartSendStr(char *p)
{
        while (*p)
        {
                UartSEND(*p++);
        }
}

void main()
```

```
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        UartInit();
        ES = 1;
        EA = 1;
        UartSENDStr("Uart Test !\r\n");

        while (1)
        {
                if (rptr != wptr)
                {
                        UartSEND(buffer[rptr++]);
                        rptr &= 0x0f;
                }
        }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

AUXR            DATA            8EH

BUSY            BIT             20H.0
WPTR            DATA            21H
RPTR            DATA            22H
BUFFER          DATA            23H                             ;16 bytes

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0023H
                LJMP            UART_ISR

                ORG             0100H
```

```
UART_ISR:
                PUSH        ACC
                PUSH        PSW
                MOV         PSW,#08H

                JNB         TI,CHKRI
                CLR         TI
                CLR         BUSY
CHKRI:
                JNB         RI,UARTISR_EXIT
                CLR         RI
                MOV         A,WPTR
                ANL         A,#0FH
                ADD         A,#BUFFER
                MOV         R0,A
                MOV         @R0,SBUF
                INC         WPTR
UARTISR_EXIT:
                POP         PSW
                POP         ACC
                RETI

UART_INIT:
                MOV         SCON,#50H
                MOV         TMOD,#20H
                MOV         TL1,#0FDH                       ;256-11059200/115200/32=0FDH
                MOV         TH1,#0FDH
                SETB        TR1
                MOV         AUXR,#40H
                CLR         BUSY
                MOV         WPTR,#00H
                MOV         RPTR,#00H
                RET

UART_SEND:
                JB          BUSY,$
                SETB        BUSY
                MOV         SBUF,A
                RET

UART_SENDSTR:
                CLR         A
                MOVC        A,@A+DPTR
                JZ          SENDEND
                LCALL       UART_SEND
                INC         DPTR
                JMP         UART_SENDSTR
SENDEND:
                RET

MAIN:
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                LCALL           UART_INIT
                SETB            ES
                SETB            EA

                MOV             DPTR,#STRING
                LCALL           UART_SENDSTR

LOOP:
                MOV             A,RPTR
                XRL             A,WPTR
                ANL             A,#0FH
                JZ              LOOP
                MOV             A,RPTR
                ANL             A,#0FH
                ADD             A,#BUFFER
                MOV             R0,A
                MOV             A,@R0
                LCALL           UART_SEND
                INC             RPTR
                JMP             LOOP

STRING:         DB              'Uart Test !',0DH,0AH,00H

                END
```

14.7.4 UART2 using T2 as baud rate generator

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (65536 - FOSC / 115200 / 4)

sfr         AUXR            =       0x8e;
sfr         T2H             =       0xd6;
sfr         T2L             =       0xd7;
sfr         S2CON           =       0x9a;
sfr         S2BUF           =       0x9b;
sfr         IE2             =       0xaf;

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

bit         busy;
char        wptr;
char        rptr;
char        buffer[16];

void Uart2Isr() interrupt 8
{
        if (S2CON & 0x02)
        {
                S2CON &= ~0x02;
                busy = 0;
        }
        if (S2CON & 0x01)
        {
                S2CON &= ~0x01;
                buffer[wptr++] = S2BUF;
                wptr &= 0x0f;
        }
}

void Uart2Init()
{
        S2CON = 0x10;
        T2L = BRT;
        T2H = BRT >> 8;
        AUXR = 0x14;
        wptr = 0x00;
        rptr = 0x00;
        busy = 0;
}

void Uart2Send(char dat)
{
        while (busy);
        busy = 1;
        S2BUF = dat;
}

void Uart2SendStr(char *p)
{
        while (*p)
        {
                Uart2SEND(*p++);
        }
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        Uart2Init();
        IE2 = 0x01;
        EA = 1;
        Uart2SENDStr("Uart Test !\r\n");

        while (1)
        {
                if (rptr != wptr)
                {
                        Uart2SEND(buffer[rptr++]);
                        rptr &= 0x0f;
                }
        }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

AUXR            DATA            8EH
T2H             DATA            0D6H
T2L             DATA            0D7H
S2CON           DATA            9AH
S2BUF           DATA            9BH
IE2             DATA            0AFH

BUSY            BIT             20H.0
WPTR            DATA            21H
RPTR            DATA            22H
BUFFER          DATA            23H                             ;16 bytes

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0043H
                LJMP            UART2_ISR

                ORG             0100H

UART2_ISR:
                PUSH            ACC
                PUSH        PSW
                MOV         PSW,#08H

                MOV         A,S2CON
                JNB         ACC.1,CHKRI
                ANL         S2CON,#NOT 02H
                CLR         BUSY
CHKRI:
                JNB         ACC.0,UART2ISR_EXIT
                ANL         S2CON,#NOT 01H
                MOV         A,WPTR
                ANL         A,#0FH
                ADD         A,#BUFFER
                MOV         R0,A
                MOV         @R0,S2BUF
                INC         WPTR
UART2ISR_EXIT:
                POP         PSW
                POP         ACC
                RETI

UART2_INIT:
                MOV         S2CON,#10H
                MOV         T2L,#0E8H                   ;65536-11059200/115200/4=0FFE8H
                MOV         T2H,#0FFH
                MOV         AUXR,#14H
                CLR         BUSY
                MOV         WPTR,#00H
                MOV         RPTR,#00H
                RET

UART2_SEND:
                JB          BUSY,$
                SETB        BUSY
                MOV         S2BUF,A
                RET

UART2_SENDSTR:
                CLR         A
                MOVC        A,@A+DPTR
                JZ          SEND2END
                LCALL       UART2_SEND
                INC         DPTR
                JMP         UART2_SENDSTR
SEND2END:
                RET

MAIN:
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                LCALL       UART2_INIT
                MOV         IE2,#01H
                SETB        EA

                MOV         DPTR,#STRING
                LCALL       UART2_SENDSTR

LOOP:
                MOV         A,RPTR
                XRL         A,WPTR
                ANL         A,#0FH
                JZ          LOOP
                MOV         A,RPTR
                ANL         A,#0FH
                ADD         A,#BUFFER
                MOV         R0,A
                MOV         A,@R0
                LCALL       UART2_SEND
                INC         RPTR
                JMP         LOOP

STRING:         DB          'Uart Test !',0DH,0AH,00H

                END
```

14.7.5 UART3 using T2 as baud rate generator

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (65536 - FOSC / 115200 / 4)

sfr         AUXR            =       0x8e;
sfr         T2H             =       0xd6;
sfr         T2L             =       0xd7;
sfr         S3CON           =       0xac;
sfr         S3BUF           =       0xad;
sfr         IE2             =       0xaf;

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

bit         busy;
char        wptr;
char        rptr;
char        buffer[16];

void Uart3Isr() interrupt 17
{
        if (S3CON & 0x02)
        {
                S3CON &= ~0x02;
                busy = 0;
        }
        if (S3CON & 0x01)
        {
                S3CON &= ~0x01;
                buffer[wptr++] = S3BUF;
                wptr &= 0x0f;
        }
}

void Uart3Init()
{
        S3CON = 0x10;
        T2L = BRT;
        T2H = BRT >> 8;
        AUXR = 0x14;
        wptr = 0x00;
        rptr = 0x00;
        busy = 0;
}

void Uart3Send(char dat)
{
        while (busy);
        busy = 1;
        S3BUF = dat;
}

void Uart3SendStr(char *p)
{
        while (*p)
        {
                Uart3SEND(*p++);
        }
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        Uart3Init();
        IE2 = 0x08;
        EA = 1;
        Uart3SENDStr("Uart Test !\r\n");

        while (1)
        {
                if (rptr != wptr)
                {
                        Uart3SEND(buffer[rptr++]);
                        rptr &= 0x0f;
                }
        }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

AUXR        DATA        8EH
T2H         DATA        0D6H
T2L         DATA        0D7H
S3CON       DATA        0ACH
S3BUF       DATA        0ADH
IE2         DATA        0AFH

BUSY        BIT         20H.0
WPTR        DATA        21H
RPTR        DATA        22H
BUFFER      DATA        23H                     ;16 bytes

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         008BH
            LJMP        UART3_ISR

            ORG         0100H

UART3_ISR:
            PUSH        ACC
            PUSH        PSW
            MOV         PSW,#08H
```

```
                MOV             A,S3CON
                JNB             ACC.1,CHKRI
                ANL             S3CON,#NOT 02H
                CLR             BUSY
CHKRI:
                JNB             ACC.0,UART3ISR_EXIT
                ANL             S3CON,#NOT 01H
                MOV             A,WPTR
                ANL             A,#0FH
                ADD             A,#BUFFER
                MOV             R0,A
                MOV             @R0,S3BUF
                INC             WPTR
UART3ISR_EXIT:
                POP             PSW
                POP             ACC
                RETI

UART3_INIT:
                MOV             S3CON,#10H
                MOV             T2L,#0E8H                   ;65536-11059200/115200/4=0FFE8H
                MOV             T2H,#0FFH
                MOV             AUXR,#14H
                CLR             BUSY
                MOV             WPTR,#00H
                MOV             RPTR,#00H
                RET

UART3_SEND:
                JB              BUSY,$
                SETB            BUSY
                MOV             S3BUF,A
                RET

UART3_SENDSTR:
                CLR             A
                MOVC            A,@A+DPTR
                JZ              SEND3END
                LCALL           UART3_SEND
                INC             DPTR
                JMP             UART3_SENDSTR
SEND3END:
                RET

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H
                LCALL           UART3_INIT
                MOV             IE2,#08H
                SETB            EA

                MOV             DPTR,#STRING
                LCALL           UART3_SENDSTR

LOOP:
                MOV             A,RPTR
                XRL             A,WPTR
                ANL             A,#0FH
                JZ              LOOP
                MOV             A,RPTR
                ANL             A,#0FH
                ADD             A,#BUFFER
                MOV             R0,A
                MOV             A,@R0
                LCALL           UART3_SEND
                INC             RPTR
                JMP             LOOP

STRING:         DB              'Uart Test !',0DH,0AH,00H

                END
```

14.7.6 UART3 using T3 as baud rate generator

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (65536 - FOSC / 115200 / 4)

sfr         T4T3M           =       0xd1;
sfr         T4L             =       0xd3;
sfr         T4H             =       0xd2;
sfr         T3L             =       0xd5;
sfr         T3H             =       0xd4;
sfr         T2L             =       0xd7;
sfr         T2H             =       0xd6;
sfr         S3CON           =       0xac;
sfr         S3BUF           =       0xad;
sfr         IE2             =       0xaf;

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

bit         busy;
char        wptr;
char        rptr;
char        buffer[16];

void Uart3Isr() interrupt 17
{
        if (S3CON & 0x02)
        {
                S3CON &= ~0x02;
                busy = 0;
        }
        if (S3CON & 0x01)
        {
                S3CON &= ~0x01;
                buffer[wptr++] = S3BUF;
                wptr &= 0x0f;
        }
}

void Uart3Init()
{
        S3CON = 0x50;
        T3L = BRT;
        T3H = BRT >> 8;
        T4T3M = 0x0a;
        wptr = 0x00;
        rptr = 0x00;
        busy = 0;
}

void Uart3Send(char dat)
{
        while (busy);
        busy = 1;
        S3BUF = dat;
}

void Uart3SendStr(char *p)
{
        while (*p)
        {
                Uart3SEND(*p++);
        }
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
```

```
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    Uart3Init();
    IE2 = 0x08;
    EA = 1;
    Uart3SENDStr("Uart Test !\r\n");

    while (1)
    {
        if (rptr != wptr)
        {
            Uart3SEND(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

T4T3M       DATA        0D1H
T4L         DATA        0D3H
T4H         DATA        0D2H
T3L         DATA        0D5H
T3H         DATA        0D4H
T2L         DATA        0D7H
T2H         DATA        0D6H
S3CON       DATA        0ACH
S3BUF       DATA        0ADH
IE2         DATA        0AFH

BUSY        BIT         20H.0
WPTR        DATA        21H
RPTR        DATA        22H
BUFFER      DATA        23H                     ;16 bytes

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         008BH
            LJMP        UART3_ISR

            ORG         0100H
```

```
UART3_ISR:
                PUSH        ACC
                PUSH        PSW
                MOV         PSW,#08H

                MOV         A,S3CON
                JNB         ACC.1,CHKRI
                ANL         S3CON,#NOT 02H
                CLR         BUSY
CHKRI:
                JNB         ACC.0,UART3ISR_EXIT
                ANL         S3CON,#NOT 01H
                MOV         A,WPTR
                ANL         A,#0FH
                ADD         A,#BUFFER
                MOV         R0,A
                MOV         @R0,S3BUF
                INC         WPTR
UART3ISR_EXIT:
                POP         PSW
                POP         ACC
                RETI

UART3_INIT:
                MOV         S3CON,#50H
                MOV         T3L,#0E8H                   ;65536-11059200/115200/4=0FFE8H
                MOV         T3H,#0FFH
                MOV         T4T3M,#0AH
                CLR         BUSY
                MOV         WPTR,#00H
                MOV         RPTR,#00H
                RET

UART3_SEND:
                JB          BUSY,$
                SETB        BUSY
                MOV         S3BUF,A
                RET

UART3_SENDSTR:
                CLR         A
                MOVC        A,@A+DPTR
                JZ          SEND3END
                LCALL       UART3_SEND
                INC         DPTR
                JMP         UART3_SENDSTR
SEND3END:
                RET

MAIN:
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                LCALL       UART3_INIT
                MOV         IE2,#08H
                SETB        EA

                MOV         DPTR,#STRING
                LCALL       UART3_SENDSTR

LOOP:
                MOV         A,RPTR
                XRL         A,WPTR
                ANL         A,#0FH
                JZ          LOOP
                MOV         A,RPTR
                ANL         A,#0FH
                ADD         A,#BUFFER
                MOV         R0,A
                MOV         A,@R0
                LCALL       UART3_SEND
                INC         RPTR
                JMP         LOOP

STRING:         DB          'Uart Test !',0DH,0AH,00H

                END
```

14.7.7 UART4 using T2 as baud rate generator

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC        11059200UL
#define     BRT         (65536 - FOSC / 115200 / 4)

sfr         AUXR        =   0x8e;
sfr         T2H         =   0xd6;
sfr         T2L         =   0xd7;
sfr         S4CON       =   0x84;
sfr         S4BUF       =   0x85;
sfr         IE2         =   0xaf;

sfr         P0M1        =   0x93;
sfr         P0M0        =   0x94;
sfr         P1M1        =   0x91;
sfr         P1M0        =   0x92;
sfr         P2M1        =   0x95;
sfr         P2M0        =   0x96;
sfr         P3M1        =   0xb1;
sfr         P3M0        =   0xb2;
sfr         P4M1        =   0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

bit         busy;
char        wptr;
char        rptr;
char        buffer[16];

void Uart4Isr() interrupt 18
{
        if (S4CON & 0x02)
        {
                S4CON &= ~0x02;
                busy = 0;
        }
        if (S4CON & 0x01)
        {
                S4CON &= ~0x01;
                buffer[wptr++] = S4BUF;
                wptr &= 0x0f;
        }
}

void Uart4Init()
{
        S4CON = 0x10;
        T2L = BRT;
        T2H = BRT >> 8;
        AUXR = 0x14;
        wptr = 0x00;
        rptr = 0x00;
        busy = 0;
}

void Uart4Send(char dat)
{
        while (busy);
        busy = 1;
        S4BUF = dat;
}

void Uart4SendStr(char *p)
{
        while (*p)
        {
                Uart4SEND(*p++);
        }
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
```

```
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    Uart4Init();
    IE2 = 0x10;
    EA = 1;
    Uart4SENDStr(“Uart Test !\r\n”);

    while (1)
    {
        if (rptr != wptr)
        {
            Uart4SEND(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

AUXR            DATA            8EH
T2H             DATA            0D6H
T2L             DATA            0D7H
S4CON           DATA            84H
S4BUF           DATA            85H
IE2             DATA            0AFH

BUSY            BIT             20H.0
WPTR            DATA            21H
RPTR            DATA            22H
BUFFER          DATA            23H                             ;16 bytes

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0093H
                LJMP            UART4_ISR

                ORG             0100H

UART4_ISR:
                PUSH            ACC
                PUSH            PSW
                MOV             PSW,#08H

                MOV             A,S4CON
                JNB             ACC.1,CHKRI
                ANL             S4CON,#NOT 02H
                CLR             BUSY
CHKRI:
                JNB             ACC.0,UART4ISR_EXIT
                ANL             S4CON,#NOT 01H
                MOV             A,WPTR
                ANL             A,#0FH
                ADD             A,#BUFFER
                MOV             R0,A
                MOV             @R0,S4BUF
                INC             WPTR
UART4ISR_EXIT:
                POP             PSW
                POP             ACC
                RETI

UART4_INIT:
                MOV             S4CON,#10H
                MOV             T2L,#0E8H                       ;65536-11059200/115200/4=0FFE8H
                MOV             T2H,#0FFH
                MOV             AUXR,#14H
                CLR             BUSY
                MOV             WPTR,#00H
                MOV             RPTR,#00H
                RET

UART4_SEND:
                JB              BUSY,$
                SETB            BUSY
                MOV             S4BUF,A
                RET

UART4_SENDSTR:
                CLR             A
                MOVC            A,@A+DPTR
                JZ              SEND4END
                LCALL           UART4_SEND
                INC             DPTR
                JMP             UART4_SENDSTR
SEND4END:
                RET

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV         P5M1, #00H

                LCALL       UART4_INIT
                MOV         IE2,#10H
                SETB        EA

                MOV         DPTR,#STRING
                LCALL       UART4_SENDSTR

LOOP:
                MOV         A,RPTR
                XRL         A,WPTR
                ANL         A,#0FH
                JZ          LOOP
                MOV         A,RPTR
                ANL         A,#0FH
                ADD         A,#BUFFER
                MOV         R0,A
                MOV         A,@R0
                LCALL       UART4_SEND
                INC         RPTR
                JMP         LOOP

STRING:         DB          'Uart Test !',0DH,0AH,00H

                END
```

14.7.8 UART4 using T4 as baud rate generator

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC        11059200UL
#define     BRT         (65536 - FOSC / 115200 / 4)

sfr         T4T3M       =   0xd1;
sfr         T4L         =   0xd3;
sfr         T4H         =   0xd2;
sfr         T3L         =   0xd5;
sfr         T3H         =   0xd4;
sfr         T2L         =   0xd7;
sfr         T2H         =   0xd6;
sfr         S4CON       =   0x84;
sfr         S4BUF       =   0x85;
sfr         IE2         =   0xaf;

sfr         P0M1        =   0x93;
sfr         P0M0        =   0x94;
sfr         P1M1        =   0x91;
sfr         P1M0        =   0x92;
sfr         P2M1        =   0x95;
sfr         P2M0        =   0x96;
sfr         P3M1        =   0xb1;
sfr         P3M0        =   0xb2;
sfr         P4M1            =     0xb3;
sfr         P4M0            =     0xb4;
sfr         P5M1            =     0xc9;
sfr         P5M0            =     0xca;

bit         busy;
char        wptr;
char        rptr;
char        buffer[16];

void Uart4Isr() interrupt 18
{
        if (S4CON & 0x02)
        {
                S4CON &= ~0x02;
                busy = 0;
        }
        if (S4CON & 0x01)
        {
                S4CON &= ~0x01;
                buffer[wptr++] = S4BUF;
                wptr &= 0x0f;
        }
}

void Uart4Init()
{
        S4CON = 0x50;
        T4L = BRT;
        T4H = BRT >> 8;
        T4T3M = 0xa0;
        wptr = 0x00;
        rptr = 0x00;
        busy = 0;
}

void Uart4Send(char dat)
{
        while (busy);
        busy = 1;
        S4BUF = dat;
}

void Uart4SendStr(char *p)
{
        while (*p)
        {
                Uart4SEND(*p++);
        }
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        Uart4Init();
        IE2 = 0x10;
        EA = 1;
        Uart4SENDStr("Uart Test !\r\n");

        while (1)
        {
                if (rptr != wptr)
                {
                        Uart4SEND(buffer[rptr++]);
                        rptr &= 0x0f;
                }
        }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

T4T3M           DATA            0D1H
T4L             DATA            0D3H
T4H             DATA            0D2H
T3L             DATA            0D5H
T3H             DATA            0D4H
T2L             DATA            0D7H
T2H             DATA            0D6H
S4CON           DATA            84H
S4BUF           DATA            85H
IE2             DATA            0AFH

BUSY            BIT             20H.0
WPTR            DATA            21H
RPTR            DATA            22H
BUFFER          DATA            23H                             ;16 bytes

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0093H
                LJMP            UART4_ISR
```

```
                    ORG                 0100H

UART4_ISR:
                    PUSH                ACC
                    PUSH                PSW
                    MOV                 PSW,#08H

                    MOV                 A,S4CON
                    JNB                 ACC.1,CHKRI
                    ANL                 S4CON,#NOT 02H
                    CLR                 BUSY
CHKRI:
                    JNB                 ACC.0,UART4ISR_EXIT
                    ANL                 S4CON,#NOT 01H
                    MOV                 A,WPTR
                    ANL                 A,#0FH
                    ADD                 A,#BUFFER
                    MOV                 R0,A
                    MOV                 @R0,S4BUF
                    INC                 WPTR
UART4ISR_EXIT:
                    POP                 PSW
                    POP                 ACC
                    RETI

UART4_INIT:
                    MOV                 S4CON,#50H
                    MOV                 T4L,#0E8H                           ;65536-11059200/115200/4=0FFE8H
                    MOV                 T4H,#0FFH
                    MOV                 T4T3M,#0A0H
                    CLR                 BUSY
                    MOV                 WPTR,#00H
                    MOV                 RPTR,#00H
                    RET

UART4_SEND:
                    JB                  BUSY,$
                    SETB                BUSY
                    MOV                 S4BUF,A
                    RET

UART4_SENDSTR:
                    CLR                 A
                    MOVC                A,@A+DPTR
                    JZ                  SEND4END
                    LCALL               UART4_SEND
                    INC                 DPTR
                    JMP                 UART4_SENDSTR
SEND4END:
                    RET

MAIN:
                    MOV                 SP, #5FH
                    MOV                 P0M0, #00H
                    MOV                 P0M1, #00H
                    MOV                 P1M0, #00H
                    MOV                 P1M1, #00H
                    MOV                 P2M0, #00H
                    MOV                 P2M1, #00H
```

```
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                LCALL           UART4_INIT
                MOV             IE2,#10H
                SETB            EA

                MOV             DPTR,#STRING
                LCALL           UART4_SENDSTR

LOOP:
                MOV             A,RPTR
                XRL             A,WPTR
                ANL             A,#0FH
                JZ              LOOP
                MOV             A,RPTR
                ANL             A,#0FH
                ADD             A,#BUFFER
                MOV             R0,A
                MOV             A,@R0
                LCALL           UART4_SEND
                INC             RPTR
                JMP             LOOP

STRING:         DB              'Uart Test !',0DH,0AH,00H

                END
```

14.7.9 Serial multi-MCUs communication

Now refer to the STC15 series data sheet, which will be supplemented later.

14.7.10 UART to LIN bus

C language code

```
// Operating frequency for test is 22.1184MHz

/************* Function Description **************
This routine is based on the experiment box 8 to program and test, whose main control chip is STC8H8K64U.
It is can be used for general reference when using STC8G and STC8H series chips.
Connect the LIN transceiver through the UART interface to realize the LIN bus signal transceiver test routine. UART1 is
connected to the computer through the serial port tool.
UART2 is connected to an external LIN transceiver (TJA1020/1) and connected to the LIN bus.
Forward the data sent by the computer serial port to the LIN bus; forward the data received from the LIN bus to the computer
serial port.
Default transmission rate: 9600 baud rate, switch baud rate before sending LIN data, send 13 dominant interval signals.
When downloading, select the clock 22.1184MHz (users can modify the frequency by themselves).
******************************************/

#include "reg51.h"
#include "intrins.h"

#define     MAIN_Fosc               22118400L

typedef     unsigned char           u8;
typedef     unsigned int            u16;
typedef     unsigned long           u32;

sfr         AUXR            =       0x8E;
sfr         S2CON           =       0x9A;
sfr         S2BUF           =       0x9B;
sfr         TH2             =       0xD6;
sfr         TL2             =       0xD7;
sfr         IE2             =       0xAF;
sfr         INT_CLKO        =       0x8F;
sfr         P_SW1           =       0xA2;
sfr         P_SW2           =       0xBA;

sfr         P4              =       0xC0;
sfr         P5              =       0xC8;
sfr         P6              =       0xE8;
sfr         P7              =       0xF8;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xB1;
sfr         P3M0            =       0xB2;
sfr         P4M1            =       0xB3;
sfr         P4M0            =       0xB4;
sfr         P5M1            =       0xC9;
sfr         P5M0            =       0xCA;
sfr         P6M1            =       0xCB;
sfr         P6M0            =       0xCC;
sfr         P7M1            =       0xE1;
sfr         P7M0            =       0xE2;
```

```
sbit        P00             =       P0^0;
sbit        P01             =       P0^1;
sbit        P02             =       P0^2;
sbit        P03             =       P0^3;
sbit        P04             =       P0^4;
sbit        P05             =       P0^5;
sbit        P06             =       P0^6;
sbit        P07             =       P0^7;
sbit        P10             =       P1^0;
sbit        P11             =       P1^1;
sbit        P12             =       P1^2;
sbit        P13             =       P1^3;
sbit        P14             =       P1^4;
sbit        P15             =       P1^5;
sbit        P16             =       P1^6;
sbit        P17             =       P1^7;
sbit        P20             =       P2^0;
sbit        P21             =       P2^1;
sbit        P22             =       P2^2;
sbit        P23             =       P2^3;
sbit        P24             =       P2^4;
sbit        P25             =       P2^5;
sbit        P26             =       P2^6;
sbit        P27             =       P2^7;
sbit        P30             =       P3^0;
sbit        P31             =       P3^1;
sbit        P32             =       P3^2;
sbit        P33             =       P3^3;
sbit        P34             =       P3^4;
sbit        P35             =       P3^5;
sbit        P36             =       P3^6;
sbit        P37             =       P3^7;
sbit        P40             =       P4^0;
sbit        P41             =       P4^1;
sbit        P42             =       P4^2;
sbit        P43             =       P4^3;
sbit        P44             =       P4^4;
sbit        P45             =       P4^5;
sbit        P46             =       P4^6;
sbit        P47             =       P4^7;
sbit        P50             =       P5^0;
sbit        P51             =       P5^1;
sbit        P52             =       P5^2;
sbit        P53             =       P5^3;
sbit        P54             =       P5^4;
sbit        P55             =       P5^5;
sbit        P56             =       P5^6;
sbit        P57             =       P5^7;

sbit        SLP_N           =       P2^4;                       //0: Sleep

/****************************** user-defined macro ***********************************/

#define     Baudrate1               (65536UL - (MAIN_Fosc /  4) / 9600UL)
#define     Baudrate2               (65536UL - (MAIN_Fosc /  4) / 9600UL)

#define     Baudrate_Break          (65536UL - (MAIN_Fosc /  4) / 6647UL)     // Baud Rate when Transmitting Dominant
Interval Signal
```

```
#define     UART1_BUF_LENGTH     32
#define     UART2_BUF_LENGTH     32

#define     LIN_ID           0x31

u8 TX1_Cnt;                                                   // count of sendding
u8 RX1_Cnt;                                                   //count of recieving
u8 TX2_Cnt;                                                   // count of sendding
u8 RX2_Cnt;                                                   // count of recieving
bit B_TX1_Busy;                                               // busy flag of sendding
bit B_TX2_Busy;                                               // busy flag of sendding
u8 RX1_TimeOut;
u8 RX2_TimeOut;

u8 xdata RX1_Buffer[UART1_BUF_LENGTH];                        //buffer if recieving
u8 xdata RX2_Buffer[UART2_BUF_LENGTH];                        // buffer if recieving

void UART1_config(u8 brt);
void UART2_config(u8 brt);
void PrintString1(u8 *puts);
void delay_ms(u8 ms);
void UART1_TxByte(u8 dat);
void UART2_TxByte(u8 dat);
void Lin_Send(u8 *puts);
void SetTimer2Baudraye(u16 dat);

//========================================================================
// function: void main(void)
// description: main function
// parameters: none.
// return: none.
// version: VER1.0
// date: 2014-11-28
// remark:
//========================================================================
void main(void)
{
        u8 i;

        P0M1 = 0; P0M0 = 0;                                   //set as quasi-bidirectional port
        P1M1 = 0; P1M0 = 0;                                   //set as quasi-bidirectional port
        P2M1 = 0; P2M0 = 0;                                   //set as quasi-bidirectional port
        P3M1 = 0; P3M0 = 0;                                   //set as quasi-bidirectional port
        P4M1 = 0; P4M0 = 0;                                   //set as quasi-bidirectional port
        P5M1 = 0; P5M0 = 0;                                   //set as quasi-bidirectional port
        P6M1 = 0; P6M0 = 0;                                   //set as quasi-bidirectional port
        P7M1 = 0; P7M0 = 0;                                   //set as quasi-bidirectional port

        UART1_config(1);
        UART2_config(2);
        EA = 1;                                               // Enable global interrupt
        SLP_N =   1;

        PrintString1("STC8H8K64U UART1 Test Programme!\r\n"); //UART1 sends a string

        while (1)
        {
                delay_ms(1);
                if(RX1_TimeOut > 0)
                {
                        if(--RX1_TimeOut == 0)                          // If it times out, the serial port reception ends
                        {
                                if(RX1_Cnt > 0)
                                {
                                        Lin_Send(RX1_Buffer);           // Send the data received by UART1 to the LIN bus
                                }
                                RX1_Cnt = 0;
                        }
                }

                if(RX2_TimeOut > 0)
                {
                        if(--RX2_TimeOut == 0)                          // If it times out, the serial port reception ends
                        {
                                if(RX2_Cnt > 0)
                                {
                                        for (i=0; I < RX2_Cnt; i++)     // End with stop 0
                                        {
                                                UART1_TxByte(RX2_Buffer[i]); // Send data received from LIN bus to UART1
                                        }
                                }
                                RX2_Cnt = 0;
                        }
                }
        }
}

//========================================================================
// function: void delay_ms(unsigned      char ms)
// description: delay function
// parameters: ms, number of ms to delay, only support 1~255ms, and automatically adapt to master clock.
// return: none.
// version: VER1.0
// date: 2013-4-1
// remark:
//========================================================================
void delay_ms(u8 ms)
{
        u16 i;
        do{
                i = MAIN_Fosc / 10000;
                while(--i);                                             //10T per loop
        }while(--ms);
}

//========================================================================
// function: u8                    Lin_CheckPID(u8       id)
// description: The ID code plus the check character is converted into a PID code.
// parameters: IDcode.
// return: PIDcode.
// version: VER1.0
// date: 2020-12-2
// remark:
//========================================================================
u8 Lin_CheckPID(u8 id)
{
        u8 returnpid ;
        u8 P0 ;
        u8 P1 ;

        P0 = (((id)^(id>>1)^(id>>2)^(id>>4))&0x01)<<6;
        P1 = ((~((id>>1)^(id>>3)^(id>>4)^(id>>5)))&0x01)<<7;

        returnpid = id|P0|P1 ;

        return returnpid ;
}

//========================================================================
// function: u8 LINCalcChecksum(u8 *dat)
// description: Calculate the checksum.
// parameters: The data transmitted by the data field.
// return: checksum.
// version: VER1.0
// date: 2020-12-2
// remark:
//========================================================================
static u8 LINCalcChecksum(u8 *dat)
{
        u16 sum = 0;
        u8 i;

        for(I = 0; i < 8; i++)
        {
        sum += dat[i];
        if(sum & 0xFF00)
        {
                sum = (sum & 0x00FF) + 1;
        }
        }
        sum ^= 0x00FF;
        return (u8)sum;
}

//========================================================================
// function: void Lin_SendBreak(void)
// description: Send a dominant interval signal.
// parameters: none.
// return: none.
// version: VER1.0
// date: 2020-12-2
// remark:
//========================================================================
void Lin_SendBreak(void)
{
        SetTimer2Baudraye(Baudrate_Break);
        UART2_TxByte(0);
        SetTimer2Baudraye(Baudrate2);
}

//========================================================================
// function: void Lin_Send(u8 *puts)
// description: Send LIN bus message.
// parameters: The content of the data field to be sent.
// return: none.
// version: VER1.0
// date: 2020-12-2
// remark:
//========================================================================
void Lin_Send(u8 *puts)
{
        u8 i;

        Lin_SendBreak();                                        //Break
        UART2_TxByte(0x55);                                     //SYNC
        UART2_TxByte(Lin_CheckPID(LIN_ID));                     //LIN ID
        for(i=0;i<8;i++)
        {
                UART2_TxByte(puts[i]);
        }
        UART2_TxByte(LINCalcChecksum(puts));
}

//========================================================================
// function: void UART1_TxByte(u8 dat)
// description: Send a byte.
// parameters: none.
// return: none..
// version: V1.0, 2014-6-30
//========================================================================
void UART1_TxByte(u8 dat)
{
        SBUF = dat;
        B_TX1_Busy = 1;
        while(B_TX1_Busy);
}

//========================================================================
// function: void UART2_TxByte(u8 dat)
// description: Send a byte.
// parameters: none.
// return: none.
// version: V1.0, 2014-6-30
//========================================================================
void UART2_TxByte(u8 dat)
{
        S2BUF = dat;
        B_TX2_Busy = 1;
        while(B_TX2_Busy);
}

//========================================================================
// function: void PrintString1(u8 *puts)
// description: UART1 sends a string function
// parameters: puts:            String pointer.
// return: none.
// version: VER1.0
// date: 2014-11-28
// remark:
//========================================================================
void PrintString1(u8 *puts)
{
        for     (; *puts != 0; puts++)                          // End with stop 0
        {
                SBUF = *puts;
                B_TX1_Busy = 1;
                while(B_TX1_Busy);
        }
}

//========================================================================
// function: void PrintString2(u8 *puts)
// description: UART1 sends a string function
// parameters: puts:            String pointer.
// return: none.
// version: VER1.0
// date: 2014-11-28
// remark:
//========================================================================
//void PrintString2(u8 *puts)
//{
//      for (; *puts != 0;  puts++)                             //End with stop 0
//      {
//              S2BUF = *puts;
//              B_TX2_Busy = 1;
//              while(B_TX2_Busy);
//      }
//}

//========================================================================
// function: SetTimer2Baudraye(u16 dat)
// description: Set Timer2 as baud rate generator.
// parameters: dat: Reload value of Timer2
// return: none.
// version: VER1.0
// date: 2014-11-28
// remark:
//========================================================================
void SetTimer2Baudraye(u16 dat)
{
        AUXR &= ~(1<<4);                                        //Timer stop
        AUXR &= ~(1<<3);                                        //Timer2 set As Timer
        AUXR |= (1<<2);                                         //Timer2 set as 1T mode
        TH2  = dat / 256;
        TL2 = dat % 256;
        IE2 &= ~(1<<2);                                         //Disable interrupt
        AUXR |= (1<<4);                                         //Timer run enable
}

//========================================================================
// function: void UART1_config(u8 brt)
// description: UART1 initialization function
// parameters: brt: baud rate selected, 2:  select Timer2 as baud rate generator, other values: select Timer1 as baud rate generator
// return: none.
// version: VER1.0
// date: 2014-11-28
// remark:
//========================================================================
void UART1_config(u8 brt)
{
        /*********** select Timer2 as baud rate generator ****************/
        if(brt == 2)
        {
                AUXR |= 0x01;                               //S1 BRT Use Timer2;
                SetTimer2Baudraye(Baudrate1);
        }

        /*********** select Timer1 as baud rate generator ****************/
        else
        {
                TR1  = 0;
                AUXR &= ~0x01;                              //S1 BRT Use Timer1;
                AUXR |= (1<<6);                             //Timer1 set as   1T mode
                TMOD &= ~(1<<6);                            //Timer1 set As   Timer
                TMOD &= ~0x30;                              //Timer1_16bitAutoReload;
                TH1 = (u8)(Baudrate1 / 256);
                TL1 = (u8)(Baudrate1 % 256);
                ET1 = 0;                                    //diable interrupt
                INT_CLKO &= ~0x02;                          //does not output clock
                TR1 = 1;
        }
        /*************************************************/

        SCON = (SCON & 0x3f) | 0x40;                        //UART1 mode:  0x00: Synchronous shift output,
                                                            //             0x40: 8-bit data, variable baud rate,
                                                            //             0x80: 9-bit data, fixed baud rate,
                                                            //             0xc0: 9-bit data, variable baud rate
//      PS    = 1;                                          //High priority
        ES    = 1;                                          //enable interrupt
        REN = 1;                                            //enable recieving
        P_SW1 &= 0x3f;
//      P_SW1 |= 0x80;                                      //UART1switch to:     0x00: P3.0 P3.1,
                                                            //                    0x40: P3.6 P3.7,
                                                            //                    0x80: P1.6 P1.7,
                                                            //                    0xC0: P4.3 P4.4

        B_TX1_Busy = 0;
        TX1_Cnt = 0;
        RX1_Cnt = 0;
}

//========================================================================
// function: void UART2_config(u8 brt)
// description: UART2 initialization function
// parameters: brt: baud rate selected, 2: select Timer2 as baud rate generator, other values: not valid.
// return: none.
// version: VER1.0
// date: 2014-11-28
// remark:
//========================================================================
void UART2_config(u8 brt)
{
        if(brt == 2)
        {
                SetTimer2Baudraye(Baudrate2);

                S2CON &= ~(1<<7);                           //8-bit data, 1 start bit, 1 stop bit, no checking
                IE2 |= 1;                                   //enable interrupt
                S2CON |= (1<<4);                            //enable recieving
                P_SW2 &= ~0x01;
//              P_SW2      |= 1;                            //UART2 switch to: 0: P1.0/P1.1, 1: P4.6/P4.7
```

```
        B_TX2_Busy = 0;
        TX2_Cnt = 0;
        RX2_Cnt = 0;
    }
}

//========================================================================
// function: void UART1_int(void) interrupt UART1_VECTOR
// description: UART1 interrupt function。
// parameters: nine.
// return: none.
// version: VER1.0
// date: 2014-11-28
// remark:
//========================================================================
void UART1_int (void) interrupt   4
{
    if(RI)
    {
        RI = 0;
        if(RX1_Cnt >= UART1_BUF_LENGTH) RX1_Cnt = 0;
        RX1_Buffer[RX1_Cnt] = SBUF;
        RX1_Cnt++;
        RX1_TimeOut = 5;
    }

    if(TI)
    {
        TI = 0;
        B_TX1_Busy = 0;
    }
}

//========================================================================
// function: void UART2_int(void) interrupt UART2_VECTOR
// description: UART2 interrupt function
// parameters: nine.
// return: none.
// version: VER1.0
// date: 2014-11-28
// remark:
//========================================================================
void UART2_int (void) interrupt   8
{
    if((S2CON & 1) != 0)
    {
        S2CON &= ~1;                                    //Clear Rx flag
        if(RX2_Cnt >= UART2_BUF_LENGTH) RX2_Cnt = 0;
        RX2_Buffer[RX2_Cnt] = S2BUF;
        RX2_Cnt++;
        RX2_TimeOut = 5;
    }

    if((S2CON &     2) != 0)
    {
        S2CON &= ~2;                                    //Clear Tx flag
        B_TX2_Busy = 0;
    }
}
```

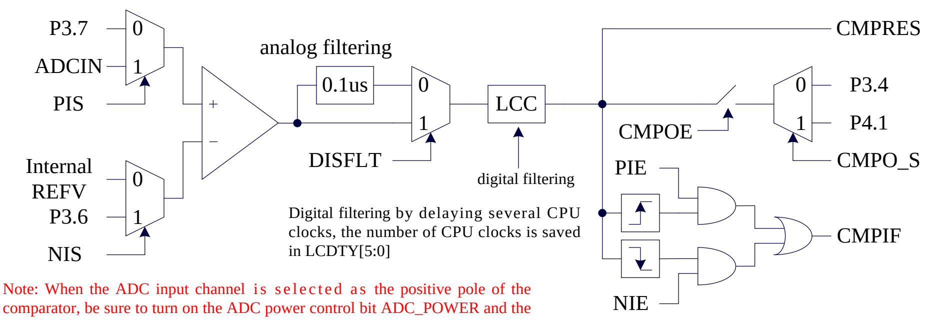
15 Comparator, Power-down Detection, Internal Reference Voltage

Product line	Comparator	
	Old version(2P+2N)	New version(4P+2N)
STC8H1K08 family	●	
STC8H1K28 family	●	
STC8H3K64S4 family	●	
STC8H3K64S2 family	●	
STC8H8K64U family A version	●	
STC8H8K64U family version		●
STC8H2K64T family	●	
STC8H4K64TLR family		●
STC8H4K64TLCD family		●
STC8H4K64LCD family		●

A comparator is integrated in STC8H series of microcontrollers. The positive terminal of the comparator can be P3.7 or ADC analog input (The positive pole of the new version of the comparator can be the P3.7 port, the P5.0 port, the P5.1 port or the analog input channel of the ADC), and the negative can be P3.6 or the REFV voltage of the internal BandGap after amplified. The application of multiple comparators can be realized through multiplexer and time division multiplexing.

There are two stage programmable filterings inside the comparator: analog filtering and digital filtering. Analog filtering can filter out glitches in the input signal, and digital filtering can wait for the input signal to stabilize before making a comparison. The result of the comparison can be obtained directly by reading the internal register bits or output the result of the comparator forward or reverse to the external port. Outputting the comparison result to the external port can be used as the trigger signal of external events and the feedback signal to expand the scope of application.

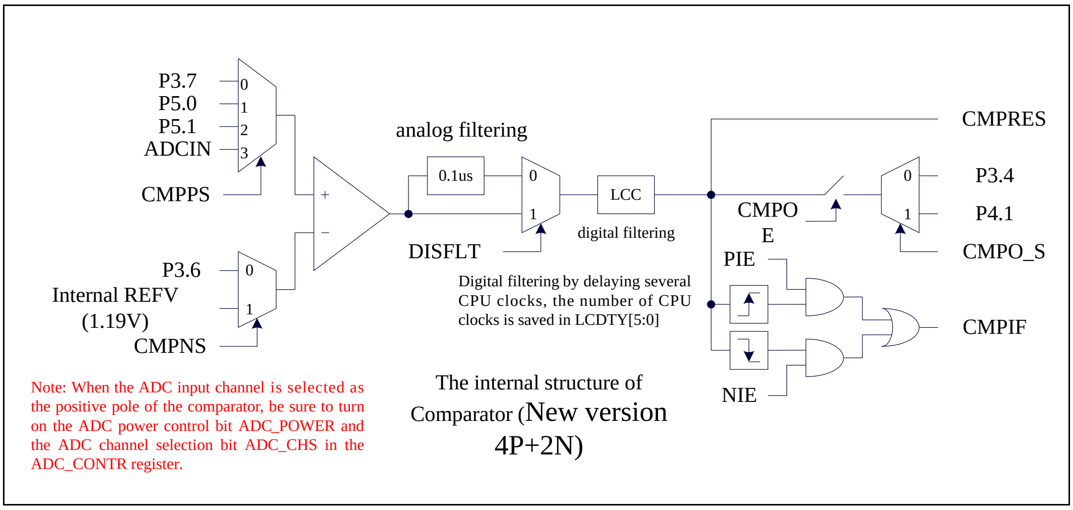
15.1 Internal Structure of Comparator



Digital filtering by delaying several CPU clocks, the number of CPU clocks is saved in LCDTY[5:0]

Note: When the ADC input channel is selected as the positive pole of the comparator, be sure to turn on the ADC power control bit ADC_POWER and the ADC channel selection bit ADC_CHS in the ADC_CONTR register.

The internal structure of Comparator (Old version 2P+2N)



Note: When the ADC input channel is selected as the positive pole of the comparator, be sure to turn on the ADC power control bit ADC_POWER and the ADC channel selection bit ADC_CHS in the ADC_CONTR register.

The internal structure of Comparator (New version 4P+2N)

15.2 Registers Related to Comparator

Symbol	Description	Address	Bit Address and Symbol								Reset Value
			B7	B6	B5	B4	B3	B2	B1	B0	
CMPCR1	Comparator Control Register 1	E6H	CMPEN	CMPIF	PIE	NIE	PIS	NIS	CMPOE	CMPRES	0000,0000
CMPCR2	Comparator Control Register 2	E7H	INVCMPO	DISFLT	LCDTY[5:0]						0000,0000
CMPEXCFG	Comparator Extended Configuration Register	FEAEH	CHYS[1:0]		-	-	-	CMPNS	CMPPS[1:0]		00xx,x000

15.2.1 Comparator control register 1

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
CMPCR1	E6H	CMPEN	CMPIF	PIE	NIE	PIS	NIS	CMPOE	CMPRES

CMPEN: Comparator enable bit

0: disable comparator

1: enable comparator

CMPIF: Comparator interrupt flag. When PIE or NIE is enabled, if the corresponding interrupt signal is generated, the hardware will automatically set CMPIF and request interrupt to CPU. This flag must be cleared by software. **(Note: When the comparator interrupt is not enabled, this interrupt flag will not be set by the hardware, that is, this interrupt flag cannot be queried when the comparator is accessed in query mode.)**

PIE: Comparator rising edge interrupt enable bit

0: disable comparator rising edge interrupt

1: enable comparator rising edge interrupt. Enable the interrupt request when the compare result of the comparator changes from 0 to 1.

NIE: Comparator falling edge interrupt enable bit

0: disable comparator falling edge interrupt

1: enable comparator falling edge interrupt. Enable the interrupt request when the compare result of the comparator changes from 1 to 0.

PIS: Positive of comparator selection bit

0: Select P3.7 as the comparator positive input source.

1: The analog input of the ADC selected by the ADC_CHS bits in ADC_CONTR is selected as the comparator positive input source.

(Note 1: When the ADC input channel is selected as the positive of the comparator, please make sure to turn on the ADC power control bit ADC_POWER and ADC channel selection bit ADC_CHS in the ADC_CONTR register)

(Note 2: When the comparator interrupt needs to be used to wake up the CPU in power-down mode/clock stop mode, we must select P3.7 as the positive of the comparator, and the ADC input channel cannot be used)

NIS: Negative of comparator selection bit

0: The REFV voltage of the internal BandGap after amplified is selected as the comparator negative input

source. (When the chip is shipped, the internal reference voltage is adjusted to 1.19V)
1: Select P3.6 as the comparator negative input source.

CMPOE: Comparator result output control bit
0: disable comparator result output.
1: enable comparator result output. The comparator result is output to P3.4 or P4.1, which is selected by CMPO_S in P_SW2.

CMPRES: Flag bit of comparator result. (Read-only)
0: the level of CMP+ is lower than CMP-.
1: the level of CMP+ is higher than CMP-.
CMPRES is the digitally filtered output signal, not the comparator's direct output.

15.2.2 Comparator control register 2

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
CMPCR2	E7H	INVCMPO	DISFLT	LCDTY[5:0]					

INVCMPO: Inverse comparator output control bit
0: Normal output the result of comparator. If CMPRES is 0, P3.4 / P4.1 output low, and vice versa output high.
1: Output the result of comparator after it is inversed. If CMPRES is 0, P3.4 / P4.1 output high, and vice versa output low.

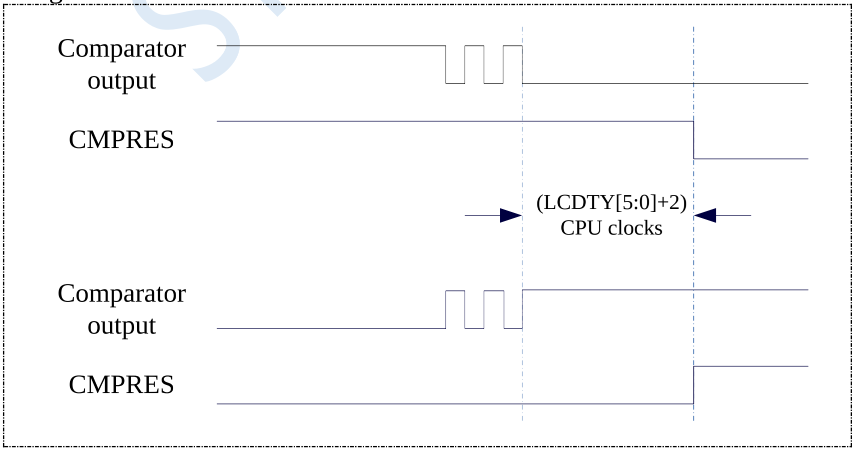
DISFLT: Analog filtering function control bit
0: enable 0.1us analog filtering function
1: disable 0.1us analog filtering function, which can speed up the comparator slightly.

LCDTY[5:0]: Digital filtering function control bit

Digital filtering is the debouncing function of the digital signal. When the comparison result changes at the rising edge or falling edge, the data changing is considered be valid only if the signal the comparator detected does not change and maintains the number of CPU clocks set in LCDTY. Otherwise, the signal will be treated as no change.

Note: When the digital filtering function is enabled, the actual waiting clock inside the chip needs to add two additional state machine switching times, that is, if LCDTY is set to 0, the digital filtering function is turned off; if LCDTY is set to a non-zero value n (n =1~63), the actual digital filtering time is (n+2) system clocks

The digital filtering function is disabled if LCDTY is set to 0.



15.2.3 Comparator Extended Configuration Register (CMPEXCFG)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
CMPEXCFG	FEAEH	CHYS[1:0]		-	-	-	CMPNS	CMPPS[1:0]	

CHYS[1:0]：Comparator DC Hysteresis Input Selection

CHYS [1:0]	Comparator DC Hysteresis Input Selection
00	0mV
01	10mV

10	20mV
11	30mV

CMPNS：Comparator negative input selection

0：P3.6

1：The REFV voltage of the internal BandGap after amplified is selected as the comparator negative input source. (When the chip is shipped, the internal reference voltage is adjusted to 1.19V)

CMPPS[1:0]：Positive of comparator selection bit

CMPPS[1:0]	Positive of comparator
00	P3.7
01	P5.0
10	P5.1
11	ADCIN

15.3 Example Routines

15.3.1 Using Old Version Comparator (Interrupt Mode)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         CMPCR1      =       0xe6;
sfr         CMPCR2      =       0xe7;

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sbit        P10         =       P1^0;
sbit        P11         =       P1^1;

void CMP_Isr() interrupt 21
{
    CMPCR1 &= ~0x40;                        //Clear interrupt flag
    if (CMPCR1 & 0x01)
    {
        P10 = !P10;                         //Falling edge interrupt test port
    }
    else
    {
        P11 = !P11;                         //Rising edge interrupt test port
    }
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;
```

```
        CMPCR2 = 0x00;
        CMPCR2 &= ~0x80;                                    //Comparator forward output
//      CMPCR2 |= 0x80;                                     //Comparator inverted output
        CMPCR2 &= ~0x40;                                    //Disable 0.1us filtering
//      CMPCR2 |= 0x40;                                     //Enable 0.1us filtering
//      CMPCR2 &= ~0x3f;                                    //Output comparator result directly
        CMPCR2 |= 0x10;                                     //Output comparator result after 16 debounce clocks
        CMPCR1 = 0x00;
        CMPCR1 |= 0x30;                                     //Enable edge interrupt of comparator
//      CMPCR1 &= ~0x20;                                    //Disable comparator rising edge interrupt
//      CMPCR1 |= 0x20;                                     //Enable comparator rising edge interrupt
//      CMPCR1 &= ~0x10;                                    //Disable comparator falling edge interrupt
//      CMPCR1 |= 0x10;                                     //Enable comparator falling edge interrupt
        CMPCR1 &= ~0x08;                                    //P3.7 is CMP+ input pin
//      CMPCR1 |= 0x08;                                     //ADC input pin is CMP+ input pin
//      CMPCR1 &= ~0x04;                                    //Internal reference voltage is CMP- input pin
        CMPCR1 |= 0x04;                                     //P3.6 is CMP- input pin
//      CMPCR1 &= ~0x02;                                    //Disable comparator output
        CMPCR1 |= 0x02;                                     //Enable Comparator output
        CMPCR1 |= 0x80;                                     //Enable comparator module

        EA = 1;

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

CMPCR1          DATA            0E6H
CMPCR2          DATA            0E7H

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             00ABH
                LJMP            CMPISR

                ORG             0100H
CMPISR:
                PUSH            ACC
                ANL             CMPCR1,#NOT 40H             ;Clear interrupt flag
                MOV             A,CMPCR1
                JB              ACC.0,RSING
FALLING:
                CPL             P1.0                        ;Falling edge interrupt test port
                POP             ACC
                RETI
RSING:
                CPL             P1.1                    ;Rising edge interrupt test port
                POP             ACC
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             CMPCR2,#00H
                ANL             CMPCR2,#NOT 80H         ;Comparator forward output
;               ORL             CMPCR2,#80H             ;Comparator inverted output
                ANL             CMPCR2,#NOT 40H         ;Disable 0.1us filtering
;               ORL             CMPCR2,#40H             ;Enable 0.1us filtering
;               ANL             CMPCR2,#NOT 3FH         ;Output comparator result directly
                ORL             CMPCR2,#10H             ;Output comparator result after 16 debounce clocks
                MOV             CMPCR1,#00H
                ORL             CMPCR1,#30H             ;Enable edge interrupt of comparator
;               ANL             CMPCR1,#NOT 20H         ;Disable comparator rising edge interrupt
;               ORL             CMPCR1,#20H             ;Enable comparator rising edge interrupt
;               ANL             CMPCR1,#NOT 10H         ;Disable comparator falling edge interrupt
;               ORL             CMPCR1,#10H             ;Enable comparator falling edge interrupt
                ANL             CMPCR1,#NOT 08H         ;P3.7 is CMP+ input pin
;               ORL             CMPCR1,#08H             ;ADC input pin is CMP+ input pin
;               ANL             CMPCR1,#NOT 04H         ;Internal reference voltage is CMP- input pin
                ORL             CMPCR1,#04H             ;P3.6 is CMP- input pin
;               ANL             CMPCR1,#NOT 02H         ;Disable comparator output
                ORL             CMPCR1,#02H             ;Enable Comparator output
                ORL             CMPCR1,#80H             ;Enable comparator module
                SETB            EA

                JMP             $

                END
```

15.3.2 Using Old Version Comparator (Polling Mode)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         CMPCR1      =       0xe6;
sfr         CMPCR2      =       0xe7;
sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

sbit        P10             =       P1^0;
sbit        P11             =       P1^1;

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        CMPCR2 = 0x00;
        CMPCR2 &= ~0x80;                                //Comparator forward output
//      CMPCR2 |= 0x80;                                 //Comparator inverted output
        CMPCR2 &= ~0x40;                                //Disable 0.1us filtering
//      CMPCR2 |= 0x40;                                 //Enable 0.1us filtering
//      CMPCR2 &= ~0x3f;                                //Output comparator result directly
        CMPCR2 |= 0x10;                                 //Output comparator result after 16 debounce clocks
        CMPCR1 = 0x00;
        CMPCR1 |= 0x30;                                 //Enable edge interrupt of comparator
//      CMPCR1 &= ~0x20;                                //Disable comparator rising edge interrupt
//      CMPCR1 |= 0x20;                                 //Enable comparator rising edge interrupt
//      CMPCR1 &= ~0x10;                                //Disable comparator falling edge interrupt
//      CMPCR1 |= 0x10;                                 //Enable comparator falling edge interrupt
        CMPCR1 &= ~0x08;                                //P3.7 is CMP+ input pin
//      CMPCR1 |= 0x08;                                 //ADC input pin is CMP+ input pin
//      CMPCR1 &= ~0x04;                                //Internal reference voltage is CMP- input pin
        CMPCR1 |= 0x04;                                 //P3.6 is CMP- input pin
//      CMPCR1 &= ~0x02;                                //Disable comparator output
        CMPCR1 |= 0x02;                                 //Enable Comparator output
        CMPCR1 |= 0x80;                                 //Enable comparator module

        while (1)
        {
                P10 = CMPCR1 & 0x01;                    //Read comparator comparison result
        }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

CMPCR1      DATA        0E6H
CMPCR2      DATA        0E7H

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN

            ORG         0100H
MAIN:
            MOV         SP, #5FH
            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
            MOV         P3M1, #00H
            MOV         P4M0, #00H
            MOV         P4M1, #00H
            MOV         P5M0, #00H
            MOV         P5M1, #00H

            MOV         CMPCR2,#00H
            ANL         CMPCR2,#NOT 80H         ;Comparator forward output
;           ORL         CMPCR2,#80H             ;Comparator inverted output
            ANL         CMPCR2,#NOT 40H         ;Disable 0.1us filtering
;           ORL         CMPCR2,#40H             ;Enable 0.1us filtering
;           ANL         CMPCR2,#NOT 3FH         ;Output comparator result directly
            ORL         CMPCR2,#10H             ;Output comparator result after 16 debounce clocks
            MOV         CMPCR1,#00H
            ORL         CMPCR1,#30H             ;Enable edge interrupt of comparator
;           ANL         CMPCR1,#NOT 20H         ;Disable comparator rising edge interrupt
;           ORL         CMPCR1,#20H             ;Enable comparator rising edge interrupt
;           ANL         CMPCR1,#NOT 10H         ;Disable comparator falling edge interrupt
;           ORL         CMPCR1,#10H             ;Enable comparator falling edge interrupt
            ANL         CMPCR1,#NOT 08H         ;P3.7 is CMP+ input pin
;           ORL         CMPCR1,#08H             ;ADC input pin is CMP+ input pin
;           ANL         CMPCR1,#NOT 04H         ;Internal reference voltage is CMP- input pin
            ORL         CMPCR1,#04H             ;P3.6 is CMP- input pin
;           ANL         CMPCR1,#NOT 02H         ;Disable comparator output
            ORL         CMPCR1,#02H             ;Enable Comparator output
            ORL         CMPCR1,#80H             ;Enable comparator module
```

```
LOOP:
                MOV             A,CMPCR1
                MOV             C,ACC.0
                MOV             P1.0,C                          ;Read comparator comparison result
                JMP             LOOP

                END
```

15.3.3 Using New Version Comparator (Interrupt Mode)

C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P_SW2           =       0xba;
sfr         CMPCR1          =       0xe6;
sfr         CMPCR2          =       0xe7;

sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

sbit        P10             =       P1^0;
sbit        P11             =       P1^1;

#define     CMPEXCFG   (*(unsigned char volatile xdata *)0xfeae)

void CMP_Isr() interrupt 21
{
        CMPCR1 &= ~0x40;                                        // Clear interrupt flag
        if (CMPCR1 & 0x01)
        {
                P10 = !P10;                                     // Rising edge interrupt test port
        }
        else
        {
                P11 = !P11;                                     // Falling edge interrupt test port
        }
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P_SW2 |= 0x80;                                      // Enable XFR access
        CMPEXCFG = 0x00;
//      CMPEXCFG |= 0x40;                                   // Comparator DC Hysteresis Input Selection
                                                            //0:0mV; 0x40:10mV; 0x80:20mV; 0xc0:30mV

        CMPEXCFG &= ~0x03;                                  //P3.7 is CMP+ input pin
//      CMPEXCFG |= 0x01;                                   //P5.0 is CMP+ input pin
//      CMPEXCFG |= 0x02;                                   //P5.1 is CMP+ input pin
//      CMPEXCFG |= 0x03;                                   // ADC input pin is CMP+ input pin
        CMPEXCFG &= ~0x04;                                  //P3.6 is CMP- input pin
//      CMPEXCFG |= 0x04;                                   // The internal 1.19V reference voltage is the CMP- input
pin
        P_SW2 &= 0x7f;                                      // Disable XFR access

        CMPCR2 = 0x00;
        CMPCR2 &= ~0x80;                                    // Comparator forward output
//      CMPCR2 |= 0x80;                                     // Comparator inverted output
        CMPCR2 &= ~0x40;                                    // Enable 0.1us filtering
//      CMPCR2 |= 0x40;                                     // Disable 0.1us filtering
//      CMPCR2 &= ~0x3f;                                    // Output comparator result directly
        CMPCR2 |= 0x10;                                     // Output comparator result after 16 debounce clocks
        CMPCR1 = 0x00;
        CMPCR1 |= 0x30;                                     // Enable edge interrupt of comparator
//      CMPCR1 &= ~0x20;                                    // Disable comparator rising edge interrupt
//      CMPCR1 |= 0x20;                                     // Enable comparator rising edge interrupt
//      CMPCR1 &= ~0x10;                                    // Disable comparator falling edge interrupt
//      CMPCR1 |= 0x10;                                     // Enable comparator falling edge interrupt
//      CMPCR1 &= ~0x02;                                    // Disable comparator output
        CMPCR1 |= 0x02;                                     // Enable Comparator output
        CMPCR1 |= 0x80;                                     // Enable comparator module

        EA = 1;

        while (1);
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz

P_SW2           DATA            0BAH
CMPCR1          DATA            0E6H
CMPCR2          DATA            0E7H

P1M1            DATA            091H
P1M0            DATA            092H
P0M1            DATA            093H
P0M0            DATA            094H
P2M1            DATA            095H
P2M0            DATA            096H
```

```
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

CMPEXCFG    XDATA       0FEAEH

            ORG         0000H
            LJMP        MAIN
            ORG         00ABH
            LJMP        CMPISR

            ORG         0100H
CMPISR:
            PUSH        ACC
            ANL         CMPCR1,#NOT 40H         ; Clear interrupt flag
            MOV         A,CMPCR1
            JB          ACC.0,RSING
FALLING:
            CPL         P1.0                    ; Falling edge interrupt test port
            POP         ACC
            RETI
RSING:
            CPL         P1.1                    ; Rising edge interrupt test port
            POP         ACC
            RETI

MAIN:
            MOV         SP, #5FH
            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
            MOV         P3M1, #00H
            MOV         P4M0, #00H
            MOV         P4M1, #00H
            MOV         P5M0, #00H
            MOV         P5M1, #00H

            MOV         P_SW2,#80H
            MOV         DPTR,# CMPEXCFG
            CLR         A
            ANL         A,#NOT 03H              ; P3.7 is CMP+ input pin
;           ORL         A,#01H                  ; P5.0 is CMP+ input pin
;           ORL         A,#02H                  ; P5.1 is CMP+ input pin
;           ORL         A,#03H                  ; ADC input pin is CMP+ input pin
            ANL         A,#NOT 04H              ; P3.6 is CMP- input pin
;           ORL         A,# 04H                 ; Internal reference voltage is CMP- input pin
            MOVX        @DPTR,A
            MOV         P_SW2,#00H

            MOV         CMPCR2,#00H
            ANL         CMPCR2,#NOT 80H         ; Comparator forward output
;           ORL         CMPCR2,#80H             ; Comparator inverted output
```

```
                ANL         CMPCR2,#NOT 40H         ; Enable 0.1us filtering
;               ORL         CMPCR2,#40H             ; Disable 0.1us filtering
;               ANL         CMPCR2,#NOT 3FH         ; Output comparator result directly
                ORL         CMPCR2,#10H             ; Output comparator result after 16 debounce clocks
                MOV         CMPCR1,#00H
                ORL         CMPCR1,#30H             ; Enable edge interrupt of comparator
;               ANL         CMPCR1,#NOT 20H         ; Disable comparator rising edge interrupt
;               ORL         CMPCR1,#20H             ; Enable comparator rising edge interrupt
;               ANL         CMPCR1,#NOT 10H         ; Disable comparator falling edge interrupt
;               ORL         CMPCR1,#10H             ; Enable comparator falling edge interrupt
;               ANL         CMPCR1,#NOT 02H         ; Disable comparator output
                ORL         CMPCR1,#02H             ; Enable Comparator output
                ORL         CMPCR1,#80H             ; Enable comparator module
                SETB        EA

                JMP         $

                END
```

15.3.4 Using New Version Comparator (Polling Mode)

C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P_SW2           =       0xba;
sfr         CMPCR1          =       0xe6;
sfr         CMPCR2          =       0xe7;

sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

sbit        P10             =       P1^0;
sbit        P11             =       P1^1;

#define     CMPEXCFG   (*(unsigned char volatile xdata *)0xfeae)

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P_SW2 |= 0x80;                                  // Enable XFR access
        CMPEXCFG = 0x00;
//      CMPEXCFG |= 0x40;           // Comparator DC Hysteresis Input Selection
                                        //0:0mV; 0x40:10mV; 0x80:20mV; 0xc0:30mV

        CMPEXCFG &= ~0x03;                              //P3.7 is CMP+ input pin
//      CMPEXCFG |= 0x01;                               //P5.0 is CMP+ input pin
//      CMPEXCFG |= 0x02;                               //P5.1 is CMP+ input pin
//      CMPEXCFG |= 0x03;                               // ADC input pin is CMP+ input pin
        CMPEXCFG &= ~0x04;                              //P3.6 is CMP- input pin
//      CMPEXCFG |= 0x04;                               // Internal reference voltage is CMP- input pin
        P_SW2 &= 0x7f;                                  // Disable XFR access

        CMPCR2 = 0x00;
        CMPCR2 &= ~0x80;                                // Comparator forward output
//      CMPCR2 |= 0x80;                                 // Comparator inverted output
        CMPCR2 &= ~0x40;                                // Enable 0.1us filtering
//      CMPCR2 |= 0x40;                                 // Disable 0.1us filtering
//      CMPCR2 &= ~0x3f;                                // Output comparator result directly
        CMPCR2 |= 0x10;                                 // Output comparator result after 16 debounce clocks
        CMPCR1 = 0x00;
        CMPCR1 |= 0x30;                                 // Enable edge interrupt of comparator
//      CMPCR1 &= ~0x20;                                // Disable comparator rising edge interrupt
//      CMPCR1 |= 0x20;                                 // Enable comparator rising edge interrupt
//      CMPCR1 &= ~0x10;                                // Disable comparator falling edge interrupt
//      CMPCR1 |= 0x10;                                 // Enable comparator falling edge interrupt
//      CMPCR1 &= ~0x02;                                // Disable comparator output
        CMPCR1 |= 0x02;                                 // Enable Comparator output
        CMPCR1 |= 0x80;                                 // Enable comparator module

        while (1)
        {
                P10 = CMPCR1 & 0x01;                    // Read comparator comparison result
        }
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz

P_SW2           DATA            0BAH
CMPCR1          DATA            0E6H
CMPCR2          DATA            0E7H

P1M1            DATA            091H
P1M0            DATA            092H
P0M1            DATA            093H
P0M0            DATA            094H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
```

```
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

CMPEXCFG    XDATA       0FEAEH

            ORG         0000H
            LJMP        MAIN

            ORG         0100H
MAIN:
            MOV         SP, #5FH
            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
            MOV         P3M1, #00H
            MOV         P4M0, #00H
            MOV         P4M1, #00H
            MOV         P5M0, #00H
            MOV         P5M1, #00H

            MOV         P_SW2,#80H
            MOV         DPTR,# CMPEXCFG
            CLR         A
            ANL         A,#NOT 03H              ; P3.7 is CMP+ input pin
;           ORL         A,#01H                  ; P5.0 is CMP+ input pin
;           ORL         A,#02H                  ; P5.1 is CMP+ input pin
;           ORL         A,#03H                  ; ADC input pin is CMP+ input pin
            ANL         A,#NOT 04H              ; P3.6 is CMP- input pin
;           ORL         A,# 04H                 ; Internal reference voltage is CMP- input pin
            MOVX        @DPTR,A
            MOV         P_SW2,#00H

            MOV         CMPCR2,#00H
            ANL         CMPCR2,#NOT 80H         ; Comparator forward output
;           ORL         CMPCR2,#80H             ; Comparator inverted output
            ANL         CMPCR2,#NOT 40H         ; Enable 0.1us filtering
;           ORL         CMPCR2,#40H             ; Disable 0.1us filtering
;           ANL         CMPCR2,#NOT 3FH         ; Output comparator result directly
            ORL         CMPCR2,#10H             ; Output comparator result after 16 debounce clocks
            MOV         CMPCR1,#00H
            ORL         CMPCR1,#30H             ; Enable edge interrupt of comparator
;           ANL         CMPCR1,#NOT 20H         ; Disable comparator rising edge interrupt
;           ORL         CMPCR1,#20H             ; Enable comparator rising edge interrupt
;           ANL         CMPCR1,#NOT 10H         ; Disable comparator falling edge interrupt
;           ORL         CMPCR1,#10H             ; Enable comparator falling edge interrupt
;           ANL         CMPCR1,#NOT 02H         ; Disable comparator output
            ORL         CMPCR1,#02H             ; Enable Comparator output
            ORL         CMPCR1,#80H             ; Enable comparator module

LOOP:
            MOV         A,CMPCR1
            MOV         C,ACC.0
            MOV         P1.0,C                  ; Read comparator comparison result
            JMP         LOOP

            END
```

15.3.5 Multiplexing Application of Old Version Comparator (Comparator+ADC input channel)

Since the analog input channel of the ADC can be seleceted as the positive of the comparator, the application of multiple comparators can be realized through the multiplexer and time-division multiplexing.

Note: When the ADC input channel is selected as the positive of the comparator, please make sure to turn on the ADC power control bit ADC_POWER and the ADC channel selection bit ADC_CHS in the ADC_CONTR register.

C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         CMPCR1      =       0xe6;
sfr         CMPCR2      =       0xe7;

sfr         ADC_CONTR   =       0xbc;

sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sbit        P10         =       P1^0;
sbit        P11         =       P1^1;

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;
```

```
    P1M0 &= 0xfe;                                   //Set P1.0 to input mode
    P1M1 |= 0x01;
    ADC_CONTR = 0x80;                               // Enable ADC module and select P1.0 as ADC input pin

    CMPCR2 = 0x00;
    CMPCR1 = 0x00;

    CMPCR1 |= 0x08;                                 // Select ADC input pin as CMP+ input pin
    CMPCR1 |= 0x04;                                 //P3.6 is CMP- input pin
    CMPCR1 |= 0x02;                                 // Enable Comparator output
    CMPCR1 |= 0x80;                                 // Enable comparator module

    while (1);
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz

CMPCR1      DATA        0E6H
CMPCR2      DATA        0E7H
ADC_CONTR   DATA        0BCH

P1M1        DATA        091H
P1M0        DATA        092H
P0M1        DATA        093H
P0M0        DATA        094H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN

            ORG         0100H
MAIN:
            MOV         SP, #5FH
            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
            MOV         P3M1, #00H
            MOV         P4M0, #00H
            MOV         P4M1, #00H
            MOV         P5M0, #00H
            MOV         P5M1, #00H

            ANL         P1M0,#0FEH                  ; Set P1.0 to input mode
            ORL         P1M1,#01H
            MOV         ADC_CONTR,#80H              ; Enable ADC module and select P1.0 as ADC input pin

            MOV         CMPCR2,#00H
```

```
                MOV         CMPCR1,#00H

                ORL         CMPCR1,#08H              ; Select ADC input pin as CMP+ input pin
                ORL         CMPCR1,#04H              ;P3.6 is CMP- input pin
                ORL         CMPCR1,#02H              ; Enable Comparator output
                ORL         CMPCR1,#80H              ; Enable comparator module

LOOP:
                JMP         LOOP

                END
```

15.3.6 Multiplexing Application of New Version Comparator (Comparator+ADC input channel)

Since the analog input channel of the ADC can be seleceted as the positive of the comparator, the application of multiple comparators can be realized through the multiplexer and time-division multiplexing.

Note: When the ADC input channel is selected as the positive of the comparator, please make sure to turn on the ADC power control bit ADC_POWER and the ADC channel selection bit ADC_CHS in the ADC_CONTR register.

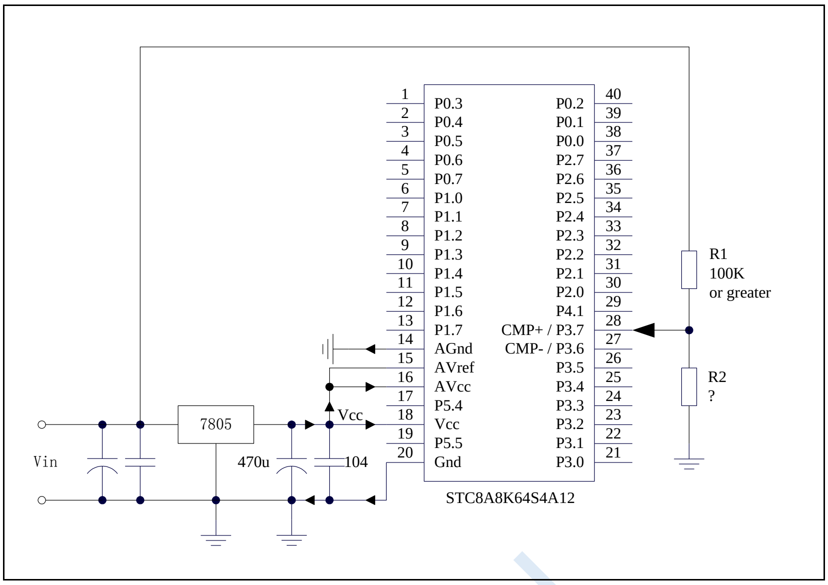
C language code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P_SW2           =       0xba;
sfr         CMPCR1          =       0xe6;
sfr         CMPCR2          =       0xe7;

sfr         ADC_CONTR       =       0xbc;

sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

sbit        P10             =       P1^0;
sbit        P11             =       P1^1;

#define     CMPEXCFG    (*(unsigned char volatile xdata *)0xfeae)

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P1M0 &= 0xfe;                                       // Set P1.0 to input mode
        P1M1 |= 0x01;
        ADC_CONTR = 0x80;                                   // Enable ADC module and select P1.0 as ADC input pin

        P_SW2 |= 0x80;                                      // Enable XFR access
        CMPEXCFG = 0x00;
//      CMPEXCFG &= ~0x03;                                  //P3.7 is CMP+ input pin
//      CMPEXCFG |= 0x01;                                   //P5.0 is CMP+ input pin
//      CMPEXCFG |= 0x02;                                   //P5.1 is CMP+ input pin
        CMPEXCFG |= 0x03;                                   //ADC input pin is CMP+ input pin
        CMPEXCFG &= ~0x04;                                  //P3.6 is CMP- input pin
//      CMPEXCFG |= 0x04;                                   // The internal 1.19V reference voltage is the CMP- input
pin
        P_SW2 &= 0x7f;                                      // Disable XFR access

        CMPCR2 = 0x00;
        CMPCR1 = 0x00;
        CMPCR1 |= 0x02;                                     // Enable Comparator output
        CMPCR1 |= 0x80;                                     // Enable comparator module

        while (1);
}
```

Assembly code

```
; Operating frequency for test is 11.0592MHz

P_SW2           DATA            0BAH
CMPCR1          DATA            0E6H
CMPCR2          DATA            0E7H
ADC_CONTR       DATA            0BCH

P1M1            DATA            091H
P1M0            DATA            092H
P0M1            DATA            093H
P0M0            DATA            094H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

CMPEXCFG        XDATA           0FEAEH

                ORG             0000H
                LJMP            MAIN
                ORG             0100H
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                ANL             P1M0,#0FEH                  ; Set P1.0 to input mode
                ORL             P1M1,#01H
                MOV             ADC_CONTR,#80H              ; Enable ADC module and select P1.0 as ADC input pin

                MOV             P_SW2,#80H
                MOV             DPTR,# CMPEXCFG
                CLR             A
;               ANL             A,#NOT 03H                  ; P3.7 is CMP+ input pin
;               ORL             A,#01H                      ; P5.0 is CMP+ input pin
;               ORL             A,#02H                      ; P5.1 is CMP+ input pin
                ORL             A,#03H                      ;ADC input pin is CMP+ input pin
                ANL             A,#NOT 04H                  ; P3.6 is CMP- input pin
;               ORL             A,# 04H                     ; The internal 1.19V reference voltage is the CMP- input
pin
                MOVX            @DPTR,A
                MOV             P_SW2,#00H

                MOV             CMPCR2,#00H
                MOV             CMPCR1,#00H
                ORL             CMPCR1,#02H                 ; Enable Comparator output
                ORL             CMPCR1,#80H                 ; Enable comparator module

LOOP:
                JMP             LOOP

                END
```

15.3.7 Comparator is Used for External Power-down Detection (User data should be saved to EEPROM in time during power down)

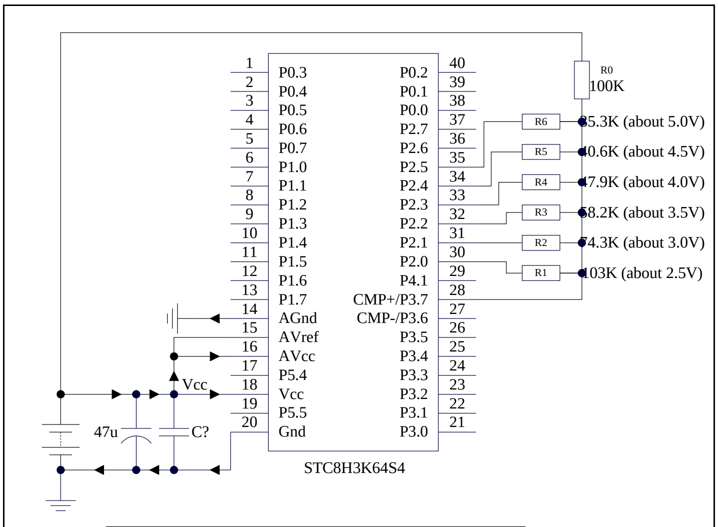


In the figure above, the resistor R1 and R2 divide the front-end voltage of the voltage regulator 7805. The divided voltage is used as the external input of the comparator CMP+ to compare with the internal reference voltage.

When the AC power is at 220V, the DC voltage at the front end of the voltage regulator block 7805 is 11V, and when the AC voltage drops to 160V, the DC voltage at the front end of the voltage regulator 7805 is 8.5V. When the dc voltage at the front end of the voltage regulator 7805 is lower than or equal to 8.5V, the dc voltage at the front end is divided by the resistors R1 and R2, and added to the comparator positive input terminal CMP+. The input voltage at the CMP+ terminal is lower than the internal reference voltage. A comparator interrupt can be generated at this time, so that there is sufficient time to save the data to the EEPROM during power-down detection. When the DC voltage of the front end of the voltage regulator 7805 is higher than 8.5V, the DC voltage input by the front end is divided by the resistors R1 and R2, and connected to the comparator positive input terminal CMP+. The input voltage of the CMP+ terminal is higher than the internal reference voltage. At this time, the CPU Can continue to work normally.

The internal reference voltage is the REPV of the internal BandGap after the amplified (the internal reference voltage is adjusted to 1.19V when the chip is shipped from the factory). The specific value should be obtained by reading the value occupied by the internal reference voltage in the internal RAM area or the Flash program memory (ROM) area. For STC8 series, the storage address of the internal reference voltage value in RAM and Flash program memory (ROM), please refer to the Chapter of "Special Parameters in Memory".

15.3.8 Comparator is Used to Detect the Operation Voltage (Battery Voltage)



	System clock<=10MHz	System clock>10MHz
C?	104(0.1uF)	103(0.01uF)

In the figure above, the working voltage of the MCU can be approximately measured using the principle of resistance voltage division. The I/O port of selected channel outputs low level, which is close to GND, and the I/O port of unselected channel working in open-drain mode outputs high. Other channels are not affected.

The negative terminal of the comparator selects the internal reference voltage, and the positive terminal selects the voltage value got from the voltage divided by a resistor as the input to the CMP+ pin.

In initialization, P2.5 ~ P2.0 are set to open-drain mode and output high. Firstly, P2.0 outputs a low level. At this time, if the VCC voltage is lower than 2.5V, the comparison value of the comparator is 0. Otherwise, if the VCC voltage is higher than 2.5V, the comparison value of the comparator is 1.

If you make sure that VCC is higher than 2.5V, then make the output of P2.0 high and the output of P2.1 low. At this time, if the VCC voltage is lower than 3.0V, the comparison value of the comparator is 0. Otherwise, if the VCC voltage is higher than 3.0V, the comparison value of the comparator is 1.

If you make sure that VCC is higher than 3.0V, then make the output of P2.1 high and the output of P2.2 low. At this time, if the VCC voltage is lower than 3.5V, the comparison value of the comparator is 0. Otherwise, if the VCC voltage is higher than 3.5V, the comparison value of the comparator is 1.

If you make sure that VCC is higher than 3.5V, then make the output of P2.2 high and the output of P2.3 low. At this time, if the VCC voltage is lower than 4.0V, the comparison value of the comparator is 0. Otherwise, if the VCC voltage is higher than 4.0V, the comparison value of the comparator is 1.

If you make sure that VCC is higher than 4.0V, then make the output of P2.3 high and the output of P2.4 low. At this time, if the VCC voltage is lower than 4.5V, the comparison value of the comparator is 0. Otherwise, if the VCC voltage is higher than 4.5V, the comparison value of the comparator is 1.

If you make sure that VCC is higher than 4.5V, then make the output of P2.4 high and the output of P2.5 low. At this time, if the VCC voltage is lower than 5.0V, the comparison value of the comparator is 0. Otherwise, if the VCC voltage is higher than 5.0V, the comparison value of the comparator is 1.

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         CMPCR1      =       0xe6;
sfr         CMPCR2      =       0xe7;

sfr         P0M1        =       0x93;
sfr         P0M0          =       0x94;
sfr         P1M1          =       0x91;
sfr         P1M0          =       0x92;
sfr         P2M1          =       0x95;
sfr         P2M0          =       0x96;
sfr         P3M1          =       0xb1;
sfr         P3M0          =       0xb2;
sfr         P4M1          =       0xb3;
sfr         P4M0          =       0xb4;
sfr         P5M1          =       0xc9;
sfr         P5M0          =       0xca;

sfr         P2M0          =       0x96;
sfr         P2M1          =       0x95;

void delay ()
{
        char i;

        for (i=0; i<20; i++);
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        unsigned char v;

        P2M0 = 0x3f;                                            //P2.5 ~ P2.0 are initialized to open-drain mode
        P2M1 = 0x3f;
        P2 = 0xff;

        CMPCR2 = 0x10;                                          //Output comparator result after 16 debounce clocks
        CMPCR1 = 0x00;
        CMPCR1 &= ~0x08;                                        //P3.7 is CMP + input pin
        CMPCR1 &= ~0x04;                                        //Internal reference voltage is CMP- input pin
        CMPCR1 &= ~0x02;                                        //Disable comparator output
        CMPCR1 |= 0x80;                                         //Enable comparator module

        while (1)
        {
                v = 0x00;                                       //Voltage <2.5V
                P2 = 0xfe;                                      //P2.0 outputs 0
                delay();
                if (!(CMPCR1 & 0x01)) goto ShowVol;
                v = 0x01;                                       //Voltage>2.5V
                P2 = 0xfd;                                      //P2.1 outputs 0
                delay();
                if (!(CMPCR1 & 0x01)) goto ShowVol;
                v = 0x03;                                               //Voltage>3.0V
                P2 = 0xfb;                                              //P2.2 outputs 0
                delay();
                if (!(CMPCR1 & 0x01)) goto ShowVol;
                v = 0x07;                                               //Voltage>3.5V
                P2 = 0xf7;                                              //P2.3 outputs 0
                delay();
                if (!(CMPCR1 & 0x01)) goto ShowVol;
                v = 0x0f;                                               //Voltage>4.0V
                P2 = 0xef;                                              //P2.4 outputs 0
                delay();
                if (!(CMPCR1 & 0x01)) goto ShowVol;
                v = 0x1f;                                               //Voltage>4.5V
                P2 = 0xdf;                                              //P2.5 outputs 0
                delay();
                if (!(CMPCR1 & 0x01)) goto ShowVol;
                v = 0x3f;                                               //Voltage>5.0V
ShowVol:
                P2 = 0xff;
                P0 = ~v;
        }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

CMPCR1          DATA            0E6H
CMPCR2          DATA            0E7H

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M0            DATA            096H
P2M1            DATA            095H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN

                ORG             0100H
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                MOV         P2M0,#00111111B         ;P2.5 ~ P2.0 are initialized to open-drain mode
                MOV         P2M1,#00111111B
                MOV         P2,#0FFH
                MOV         CMPCR2,#10H             ;Output comparator result after 16 debounce clocks
                MOV         CMPCR1,#00H
                ANL         CMPCR1,#NOT 08H         ;P3.7 is CMP+ input pin
                ANL         CMPCR1,#NOT 04H         ;Internal reference voltage is CMP- input pin
                ANL         CMPCR1,#NOT 02H         ;Disable comparator output
                ORL         CMPCR1,#80H             ;Enable comparator module
LOOP:
                MOV         R0,#00000000B           ;Voltage <2.5V
                MOV         P2,#11111110B           ;P2.0 outputs 0
                CALL        DELAY
                MOV         A,CMPCR1
                JNB         ACC.0,SKIP
                MOV         R0,#00000001B           ;Voltage>2.5V
                MOV         P2,#11111101B           ;P2.1 outputs 0
                CALL        DELAY
                MOV         A,CMPCR1
                JNB         ACC.0,SKIP
                MOV         R0,#00000011B           ;Voltage>3.0V
                MOV         P2,#11111011B           ;P2.2 outputs 0
                CALL        DELAY
                MOV         A,CMPCR1
                JNB         ACC.0,SKIP
                MOV         R0,#00000111B           ;Voltage>3.5V
                MOV         P2,#11110111B           ;P2.3 outputs 0
                CALL        DELAY
                MOV         A,CMPCR1
                JNB         ACC.0,SKIP
                MOV         R0,#00001111B           ;Voltage>4.0V
                MOV         P2,#11101111B           ;P2.4 outputs 0
                CALL        DELAY
                MOV         A,CMPCR1
                JNB         ACC.0,SKIP
                MOV         R0,#00011111B           ;Voltage>4.5V
                MOV         P2,#11011111B           ;P2.5 outputs 0
                CALL        DELAY
                MOV         A,CMPCR1
                JNB         ACC.0,SKIP
                MOV         R0,#00111111B           ;Voltage>5.0V
SKIP:
                MOV         P2,#11111111B
                MOV         A,R0
                CPL         A
                MOV         P0,A                    ;P0.5 ~ P0.0 display voltage
                JMP         LOOP

DELAY:
                MOV         R0,#20
                DJNZ        R0,$
                RET

                END
```

16 IAP/EEPROM/DATA-FLASH

Large capacity of internal EEPROM is integrated in STC8H series of microcontrollers. The internal Data Flash can be used as EEPROM by using ISP / IAP technology. And it can be repeatedly erased more than 100,000 times. EEPROM can be divided into several sectors, each sector contains 512 bytes.

Note: The EEPROM write operation can only write the 1 in the byte as 0. When the 0 in the byte needs to be written as 1, the sector erase operation must be performed. The read/write operation of EEPROM is performed in units of 1 byte, while the erase operation of EEPROM is performed in units of 1 sector (512 bytes). During the erasing operation, if there is something that needs to be reserved in the target sector Data, these data must be read into RAM for temporary storage in advance, and then the saved data and the data to be updated will be written back to EEPROM/DATA-FLASH after erasing is completed.

When EEPROM is used, it is recommended that the data modified at the same time be stored in the same sector, and data modified at different time be stored in different sectors, and not necessarily full. Data memory is erased sector by sector.

EEPROM can be used to save some parameters which need to be modified in the application and need be kept when power down takes place. In the user program, byte read / byte programming / sector erase can be performed to the EEPROM. When the operating voltage is low, it is recommended not to carry out EEPROM operation to avoid data loss.

16.1 EEPROM operation time

- Read 1 byte: 4 system clocks (use MOVC instruction to read more convenient and fast)
- Programming 1 byte: about 30～40us (the actual programming time is 6～7.5us, but state conversion time and various control SETUP and HOLD time of the control signal)
- Erase 1 sector (512 bytes): about 4～6ms

The time required for EEPROM operation is automatically controlled by the hardware, and the user only needs to set the IAP_TPS register correctly.
IAP_TPS=System operating frequency/1000000 (the decimal part is rounded to the nearest whole number)
For example: the operating frequency of the system is 12MHz, then IAP_TPS is set to 12
Another example: the system operating frequency is 22.1184MHz, then IAP_TPS is set to 22
Another example: the system operating frequency is 5.5296MHz, then IAP_TPS is set to 6

16.2 Registers Related to EEPROM

Symbol	Description	Address	Bit Address and Symbol								Reset Value
			B7	B6	B5	B4	B3	B2	B1	B0	
IAP_DATA	IAP Flash Data Register	C2H									1111,1111
IAP_ADDRH	IAP Flash Address High Byte	C3H									0000,0000
IAP_ADDRL	IAP Flash Address Low Byte	C4H									0000,0000
IAP_CMD	IAP Flash Command Register	C5H	-	-	-	-	-	-	CMD[1:0]		xxxx,xx00
IAP_TRIG	IAP Flash Trigger register	C6H									0000,0000
IAP_CONTR	IAP Control Register	C7H	IAPEN	SWBS	SWRST	CMD_FAIL	-	-	-	-	0000,xxxx
IAP_TPS	IAP Waiting Time Control Register	F5H	-	-	IAPTPS[5:0]						xx00,0000

16.2.1 EEPROM data register (IAP_DATA)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
IAP_DATA	C2H								

During EEPROM being read operation, the EEPROM data be read after the command execution is completed is stored in the IAP_DATA register. When writing the EEPROM, the data to be written must be stored in the IAP_DATA register before the write command is sent. The erase EEPROM command is not related to the IAP_DATA register.

16.2.2 EEPROM address registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
IAP_ADDRH	C3H								
IAP_ADDRL	C4H								

The target address register of EEPROM for reading, writing and erasing operation. IAP_ADDRH is the high byte address, and IAP_ADDRL is the low byte of the address.

16.2.3 EEPROM command register (IAP_CMD)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
IAP_CMD	C5H	-	-	-	-	-	-	CMD[1:0]	

CMD[1:0]: EEPROM operation command to be sent.

00: No operation.

01: EEPROM reading command. Read one byte from the destination address. Note: Writing operations can only write 1 as 0 in the destination byte, not 0 as 1. Generally, when the target byte is not FFH, it must be erased first.

10: EEPROM writing command. Write one byte from the destination address.

11: EEPROM erasing command. Write one sector from the destination address. Note: The erase operation will erase 1 sector (512 bytes) at a time, and the content of the entire sector will become FFH.

16.2.4 EEPROM trigger register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
IAP_TRIG	C6H								

After setting the command register, address register, data register and control register of EEPROM for reading, writing and erasing operation, 5AH and A5H are written to the trigger register IAP_TRIG sequentially to trigger the corresponding operation. The order of 5AH and A5H can not be changed. After the operation is completed, the contents of the EEPROM address registers IAP_ADDRH, IAP_ADDRL and the EEPROM command register IAP_CMD do not change. The value of the IAP_ADDRH and IAP_ADDRL registers must be updated manually if the datum of the next address needs to be operated.

Note: For every EEPROM operation, 5AH should be written to IAP_TRIG firstly and then A5H to take effect the corresponding command. After the trigger command has been written, the CPU is in IDLE state until the corresponding IAP operation completes. And then the CPU will return to the normal state from the IDLE and resume executing the CPU instructions.

16.2.5 EEPROM control register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
IAP_CONTR	C7H	IAPEN	SWBS	SWRST	CMD_FAIL	-	-	-	-

IAPEN: EEPROM operation enable control bit.

0: disable EEPROM operation.

1: Enable EEPROM operation.

SWBS: Software reset selection control bit, which should be used with SWRST.

0: Execute the program from the user code area after the software reset.

1: Execute the program from the ISP memory area after the software reset.

SWRST: Software reset control bit.

0: No operation.

1: Generate software reset.

CMD_FAIL: Command fail status bit for EEPROM operation which should be cleared by software.

0: EEPROM operation is right.

1: EEPROMoperation fails.

16.2.6 EEPROM erase wait time control register

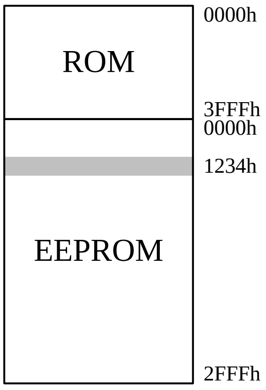
Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
IAP_TPS	F5H	-	-	IAPTPS[5:0]					

Need to be set according to the operating frequency. If the working frequency is 12MHz, IAP_TPS needs to be set to 12; if the working frequency is 24MHz, IAP_TPS needs to be set to 24, and so on for other frequencies.

16.3 EEPROM Size and Address

There is EEPROM for saving user data in all STC8H series of microcontrollers. There are three operation modes for the internal EEPROM: reading, writing, and erasing. The erasing operation is performed in sectors. Each sector has 512 bytes. That is, each time an erasing command will erase a sector when it executes. The reading and writing operations are in bytes, that is, only one byte can be read or written each time when a reading or writing command is executed.

There are two ways to access the internal EEPROM of STC8H series microcontrollers: IAP mode and MOVC mode. The IAP mode can perform reading, writing and erasing operations on the EEPROM. MOVC can only perform reading operations on the EEPROM, cannot perform writing and erasing operations. Regardless of whether IAP or MOVC is used to access the EEPROM, the correct target address must be set firstly. In IAP mode, the target address is the same as the actual physical address of the EEPROM. Both of them are accessed from address 0000H. However, when using MOVC instruction to read EEPROM data, the target address must be the actual physical address of the EEPROM plus a program size offset. STC8H1K16 is used as an example to describe the target address in detail as following:



STC8C1K16

The program space of STC8H1K16 is 16K bytes (0000h ~ 3FFFh), and the EEPROM space is 12K bytes (0000h ~ 2FFFh). When you need to read, write, and erase the unit with EEPROM physical address 1234h, if you use IAP to access, set the target address to 1234h, that is, IAP_ADDRH is set to 12h, IAP_ADDRL is set to 34h, and then the corresponding trigger command can be set and the 1234h can be operated correctly. However, if the 1234h unit of the EEPROM is read by MOVC, the flash program memory (ROM) space must be added in addition to 4000h. That is, the DPTR must be set to 5234h before the MOVC instruction can be used for reading.

Note: Because the erasing is performed in 512-byte units, the lower 9 bits of the target address set when performing the erasing operation are meaningless. For example, if the target address is set to 1234H, 1200H, 1300H or 13FFH when executing the erasing command, the final erasing operation is the same, and the 512 bytes of 1200H ~ 13FFH are erased.

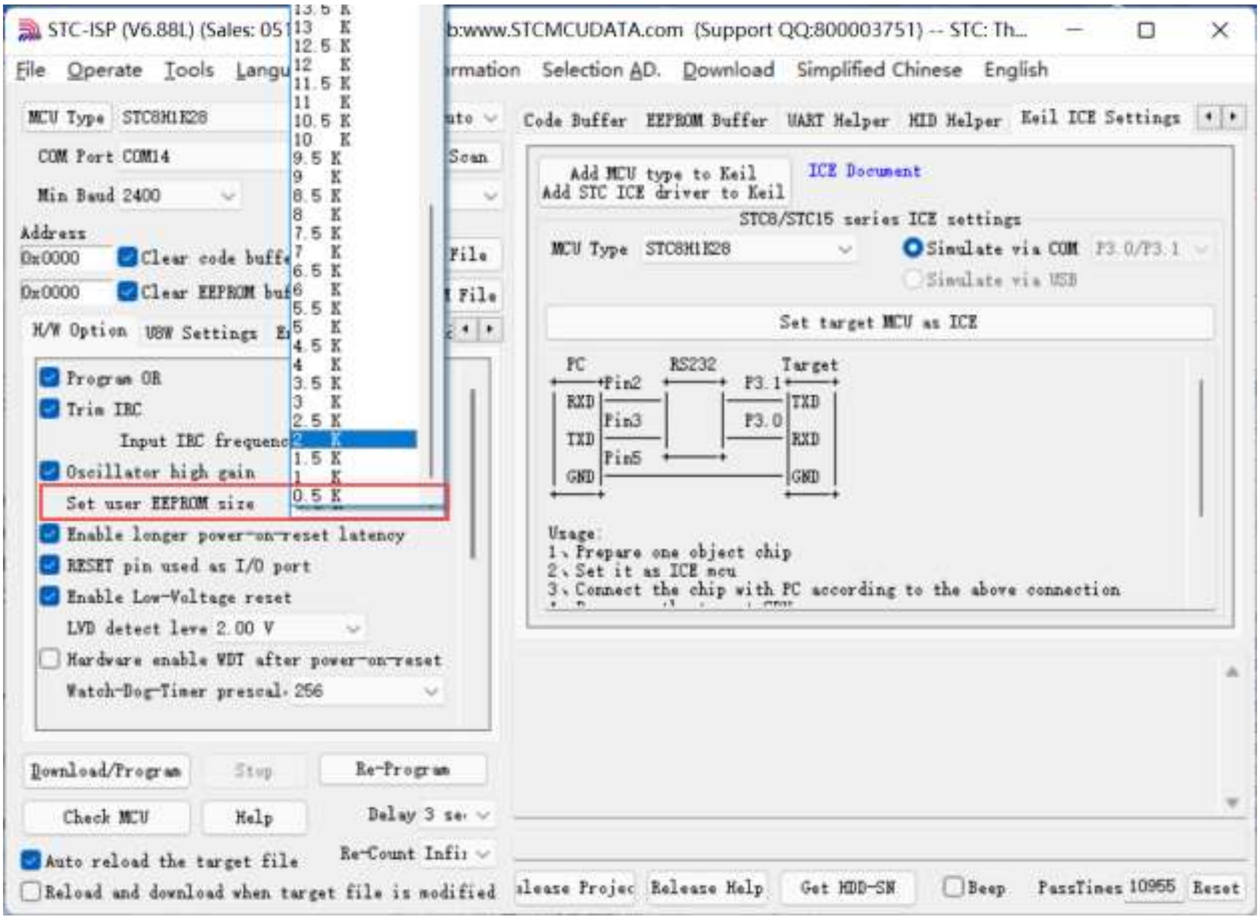
The size and access address of the internal EEPROM are different for different models. The size and address of EEPROM of each model are listed in the table below.

Model	Size	Sectors	Reading, writing, erasing in IAP mode		Reding using MOVC	
			Beginning adress	End address	Beginning adress	End address
STC8H1K16	12K	24	0000h	2FFFh	4000h	6FFFh
STC8H1K24	4K	8	0000h	0FFFh	6000h	6FFFh
STC8H1K28	User defined[1]					
STC8H1K33	User defined[1]					
STC8H1K08	4K	8	0000h	0FFFh	2000h	2FFFh
STC8H1K12	User defined[1]					
STC8H1K17	User defined[1]					
STC8H3K32S4	32K	64	0000h	7FFFh	8000h	FFFFh
STC8H3K48S4	16K	32	0000h	3FFFh	C000h	FFFFh
STC8H3K60S4	4K	8	0000h	0FFFh	F000h	FFFFh
STC8H3K64S4	User defined[1]					
STC8H3K32S2	32K	64	0000h	7FFFh	8000h	FFFFh
STC8H3K48S2	16K	32	0000h	3FFFh	C000h	FFFFh
STC8H3K60S2	4K	8	0000h	0FFFh	F000h	FFFFh
STC8H3K64S2	User defined[1]					
STC8H8K32U	32K	64	0000h	7FFFh	8000h	FFFFh
STC8H8K48U	16K	32	0000h	3FFFh	C000h	FFFFh
STC8H8K60U	4K	8	0000h	0FFFh	F000h	FFFFh
STC8H8K64U	User defined[1]					
STC8H2K32T	32K	64	0000h	7FFFh	8000h	FFFFh
STC8H2K48T	16K	32	0000h	3FFFh	C000h	FFFFh
STC8H2K60T	4K	8	0000h	0FFFh	F000h	FFFFh
STC8H2K64T	User defined[1]					
STC8H4K32TLR	32K	64	0000h	7FFFh	8000h	FFFFh
STC8H4K48TLR	16K	32	0000h	3FFFh	C000h	FFFFh
STC8H4K60TLR	4K	8	0000h	0FFFh	F000h	FFFFh
STC8H4K64TLR	User defined[1]					
STC8H4K32TLCD	32K	64	0000h	7FFFh	8000h	FFFFh
STC8H4K48TLCD	16K	32	0000h	3FFFh	C000h	FFFFh
STC8H4K60TLCD	4K	8	0000h	0FFFh	F000h	FFFFh
STC8H4K64TLCD	User defined[1]					
STC8H4K32LCD	32K	64	0000h	7FFFh	8000h	FFFFh
STC8H4K48LCD	16K	32	0000h	3FFFh	C000h	FFFFh
STC8H4K60LCD	4K	8	0000h	0FFFh	F000h	FFFFh
STC8H4K64LCD	User defined[1]					

[1]: This is a special model. The EEPROM size of this model can be set by the user when downloading by the ISP, as shown below:

Users can plan any EEPROM space in the entire FLASH space provided that the size does not exceed the FLASH size according to their own needs. It should be noted that the EEPROM is always planned from the back to the front.

For example, the size of FLASH in STC8H1K28 is 28K. If user wants to allocate 8K of it as EEPROM, the physical address of EEPROM is the last 8K of 28K, and the physical address is 5000h ~ 6FFFh. Of course, if the user uses IAP to access, the target address still starts from 0000h and ends at 1FFFh. When using MOVC to read, the target address is in the range from 5000h to 6FFFh.


STC-ISP (V6.88L) (Sales: 05 … b:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...

MCU Type STC8H1K28
COM Port COM14
Min Baud 2400
Address
0x0000 ☑ Clear code buffe
0x0000 ☑ Clear EEPROM buf

☑ Program OR
☑ Trim IRC
Input IRC frequenc
☑ Oscillator high gain
Set user EEPROM size 0.5 K
☑ Enable longer power-on-reset latency
☑ RESET pin used as I/O port
☑ Enable Low-Voltage reset
LVD detect leve 2.00 V
□ Hardware enable WDT after power-on-reset
Watch-Dog-Timer prescal: 256

☑ Auto reload the target file
□ Reload and download when target file is modified

STC8/STC15 series ICE settings
MCU Type STC8H1K28
● Simulate via COM P3.0/P3.1
○ Simulate via USB

Usage:
1、Prepare one object chip
2、Set it as ICE mcu
3、Connect the chip with PC according to the above connection

□ Beep PassTimes 10955

16.4 Example Routines

16.4.1 EEPROM Basic Operation

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sfr         IAP_DATA    =       0xC2;
sfr         IAP_ADDRH   =       0xC3;
sfr         IAP_ADDRL   =       0xC4;
sfr         IAP_CMD     =       0xC5;
sfr         IAP_TRIG    =       0xC6;
sfr         IAP_CONTR   =       0xC7;
sfr         IAP_TPS     =       0xF5;

void IapIdle()
{
    IAP_CONTR = 0;                          //Disable IAP function
    IAP_CMD = 0;                            //Clear command register
    IAP_TRIG = 0;                           //Clear trigger register
    IAP_ADDRH = 0x80;                       //Set the address to a non-IAP area
    IAP_ADDRL = 0;
}

char IapRead(int addr)
{
    char dat;

    IAP_CONTR = 0x80;                       //Enable IAP
    IAP_TPS = 12;                           //Set the erasing wait parameter of 12MHz
    IAP_CMD = 1;                            //Set IAP read command
    IAP_ADDRL = addr;                       //Set IAP low address
    IAP_ADDRH = addr >> 8;                  //Set IAP high address
    IAP_TRIG = 0x5a;                        //Write trigger command (0x5a)
    IAP_TRIG = 0xa5;                        //Write trigger command (0xa5)
    _nop_();
    dat = IAP_DATA;                         //Read IAP data
    IapIdle();                              //Disable IAP function

    return dat;
}
```

```
void IapProgram(int addr, char dat)
{
        IAP_CONTR = 0x80;                               //Enable IAP
        IAP_TPS = 12;                                   //Set the erasing wait parameter of 12MHz
        IAP_CMD = 2;                                    //Set IAP writing command
        IAP_ADDRL = addr;                               //Set IAP low address
        IAP_ADDRH = addr >> 8;                          //Set IAP high address
        IAP_DATA = dat;                                 //Write IAP data
        IAP_TRIG = 0x5a;                                //Write trigger command (0x5a)
        IAP_TRIG = 0xa5;                                //Write trigger command (0xa5)
        _nop_();
        IapIdle();                                      //Disable IAP function
}

void IapErase(int addr)
{
        IAP_CONTR = 0x80;                               //Enable IAP
        IAP_TPS = 12;                                   //Set the erasing wait parameter of 12MHz
        IAP_CMD = 3;                                    //Set IAP erasing command
        IAP_ADDRL = addr;                               //Set IAP low address
        IAP_ADDRH = addr >> 8;                          //Set IAP high address
        IAP_TRIG = 0x5a;                                //Write trigger command (0x5a)
        IAP_TRIG = 0xa5;                                //Write trigger command (0xa5)
        _nop_();                                        //
        IapIdle();                                      //Disable IAP function
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        IapErase(0x0400);
        P0 = IapRead(0x0400);                           //P0=0xff
        IapProgram(0x0400, 0x12);
        P1 = IapRead(0x0400);                           //P1=0x12

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

IAP_DATA        DATA            0C2H
IAP_ADDRH       DATA            0C3H
IAP_ADDRL       DATA            0C4H
IAP_CMD         DATA            0C5H
IAP_TRIG        DATA            0C6H
```

```
IAP_CONTR       DATA        0C7H
IAP_TPS         DATA        0F5H

P0M1            DATA        093H
P0M0            DATA        094H
P1M1            DATA        091H
P1M0            DATA        092H
P2M1            DATA        095H
P2M0            DATA        096H
P3M1            DATA        0B1H
P3M0            DATA        0B2H
P4M1            DATA        0B3H
P4M0            DATA        0B4H
P5M1            DATA        0C9H
P5M0            DATA        0CAH

                ORG         0000H
                LJMP        MAIN

                ORG         0100H

IAP_IDLE:
                MOV         IAP_CONTR,#0            ;Disable IAP function
                MOV         IAP_CMD,#0              ;Clear command register
                MOV         IAP_TRIG,#0             ;Clear trigger register
                MOV         IAP_ADDRH,#80H          ;Set the address to a non-IAP area
                MOV         IAP_ADDRL,#0
                RET

IAP_READ:
                MOV         IAP_CONTR,#80H          ;Enable IAP
                MOV         IAP_TPS,#12             ;Set the erasing wait parameter of 12MHz
                MOV         IAP_CMD,#1              ;Set IAP read command
                MOV         IAP_ADDRL,DPL           ;Set IAP low address
                MOV         IAP_ADDRH,DPH           ;Set IAP high address
                MOV         IAP_TRIG,#5AH           ;Write trigger command (0x5a)
                MOV         IAP_TRIG,#0A5H          ;Write trigger command (0xa5)
                NOP
                MOV         A,IAP_DATA              ;Read IAP data
                LCALL       IAP_IDLE                ;Disable IAP function
                RET

IAP_PROGRAM:
                MOV         IAP_CONTR,#80H          ;Enable IAP
                MOV         IAP_TPS,#12             ;Set the erasing wait parameter of 12MHz
                MOV         IAP_CMD,#2              ;Set IAP writing command
                MOV         IAP_ADDRL,DPL           ;Set IAP low address
                MOV         IAP_ADDRH,DPH           ;Set IAP high address
                MOV         IAP_DATA,A              ;Write IAP data
                MOV         IAP_TRIG,#5AH           ;Write trigger command (0x5a)
                MOV         IAP_TRIG,#0A5H          ;Write trigger command (0xa5)
                NOP
                LCALL       IAP_IDLE                ;Disable IAP function
                RET

IAP_ERASE:
                MOV         IAP_CONTR,#80H          ;Enable IAP
                MOV         IAP_TPS,#12             ;Set the erasing wait parameter of 12MHz
                MOV         IAP_CMD,#3              ;Set IAP erasing command
                MOV         IAP_ADDRL,DPL           ;Set IAP low address
                MOV         IAP_ADDRH,DPH           ;Set IAP high address
                MOV         IAP_TRIG,#5AH           ;Write trigger command (0x5a)
                MOV         IAP_TRIG,#0A5H          ;Write trigger command (0xa5)
                NOP
                LCALL       IAP_IDLE                ;Disable IAP function
                RET

MAIN:
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                MOV         DPTR,#0400H
                LCALL       IAP_ERASE
                MOV         DPTR,#0400H
                LCALL       IAP_READ
                MOV         P0,A                    ;P0=0FFH
                MOV         DPTR,#0400H
                MOV         A,#12H
                LCALL       IAP_PROGRAM
                MOV         DPTR,#0400H
                LCALL       IAP_READ
                MOV         P1,A                    ;P1=12H

                SJMP        $

                END
```

16.4.2 Read EEPROM using MOVC

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

sfr         IAP_DATA        =       0xC2;
sfr         IAP_ADDRH       =       0xC3;
sfr         IAP_ADDRL       =       0xC4;
sfr         IAP_CMD         =       0xC5;
sfr         IAP_TRIG        =       0xC6;
sfr         IAP_CONTR       =       0xC7;
sfr         IAP_TPS         =       0xF5;

#define     IAP_OFFSET              0x4000H                         //STC8H1K16

void IapIdle()
{
        IAP_CONTR = 0;                                              //Disable IAP function
        IAP_CMD = 0;                                                //Clear command register
        IAP_TRIG = 0;                                               //Clear trigger register
        IAP_ADDRH = 0x80;                                           //Set the address to a non-IAP area
        IAP_ADDRL = 0;
}

char IapRead(int addr)
{
        addr += IAP_OFFSET;                                         //Using MOVC to read the EEPROM needs to add the
corresponding offset
        return *(char code *)(addr);                                //Read data using MOVC
}

void IapProgram(int addr, char dat)
{
        IAP_CONTR = 0x80;                                           //Enable IAP
        IAP_TPS = 12;                                               //Set the erasing wait parameter of 12MHz
        IAP_CMD = 2;                                                //Set IAP writing command
        IAP_ADDRL = addr;                                           //Set IAP low address
        IAP_ADDRH = addr >> 8;                                      //Set IAP high address
        IAP_DATA = dat;                                             //Write IAP data
        IAP_TRIG = 0x5a;                                            //Write trigger command (0x5a)
        IAP_TRIG = 0xa5;                                            //Write trigger command (0xa5)
        _nop_();
        IapIdle();                                                  //Disable IAP function
}

void IapErase(int addr)
{
        IAP_CONTR = 0x80;                                           //Enable IAP
        IAP_TPS = 12;   //Set the erasing wait parameter of 12MHz
        IAP_CMD = 3;    //Set IAP erasing command
        IAP_ADDRL = addr;                                           //Set IAP low address
        IAP_ADDRH = addr >> 8;                                      //Set IAP high address
        IAP_TRIG = 0x5a;                                            //Write trigger command (0x5a)
        IAP_TRIG = 0xa5;                                            //Write trigger command (0xa5)
        _nop_();                                                    //
        IapIdle();                                                  //Disable IAP function
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        IapErase(0x0400);
        P0 = IapRead(0x0400);                           //P0=0xff
        IapProgram(0x0400, 0x12);
        P1 = IapRead(0x0400);                           //P1=0x12

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

IAP_DATA        DATA        0C2H
IAP_ADDRH       DATA        0C3H
IAP_ADDRL       DATA        0C4H
IAP_CMD         DATA        0C5H
IAP_TRIG        DATA        0C6H
IAP_CONTR       DATA        0C7H
IAP_TPS         DATA        0F5H

IAP_OFFSET      EQU         4000H                   ;STC8H1K16

P0M1            DATA        093H
P0M0            DATA        094H
P1M1            DATA        091H
P1M0            DATA        092H
P2M1            DATA        095H
P2M0            DATA        096H
P3M1            DATA        0B1H
P3M0            DATA        0B2H
P4M1            DATA        0B3H
P4M0            DATA        0B4H
P5M1            DATA        0C9H
P5M0            DATA        0CAH

                ORG         0000H
                LJMP        MAIN

                ORG         0100H

IAP_IDLE:
                MOV         IAP_CONTR,#0            ;Disable IAP function
                MOV         IAP_CMD,#0              ;Clear command register
                MOV         IAP_TRIG,#0             ;Clear trigger register
                MOV         IAP_ADDRH,#80H          ;Set the address to a non-IAP area
                MOV         IAP_ADDRL,#0
                RET
```

```
IAP_READ:
                MOV         A,#LOW IAP_OFFSET          ;Using MOVC to read the EEPROM needs to add the
corresponding offset
                ADD         A,DPL
                MOV         DPL,A
                MOV         A,@HIGH IAP_OFFSET
                ADDC        A,DPH
                MOV         DPH,A
                CLR         A
                MOVC        A,@A+DPTR                  ;Read data using MOVC
                RET

IAP_PROGRAM:
                MOV         IAP_CONTR,#80H             ;Enable IAP
                MOV         IAP_TPS,#12                ;Set the erasing wait parameter of 12MHz
                MOV         IAP_CMD,#2                 ;Set IAP writing command
                MOV         IAP_ADDRL,DPL              ;Set IAP low address
                MOV         IAP_ADDRH,DPH              ;Set IAP high address
                MOV         IAP_DATA,A                 ;Write IAP data
                MOV         IAP_TRIG,#5AH              ;Write trigger command (0x5a)
                MOV         IAP_TRIG,#0A5H             ;Write trigger command (0xa5)
                NOP
                LCALL       IAP_IDLE                   ;Disable IAP function
                RET

IAP_ERASE:
                MOV         IAP_CONTR,#80H             ;Enable IAP
                MOV         IAP_TPS,#12                ;Set the erasing wait parameter of 12MHz
                MOV         IAP_CMD,#3                 ;Set IAP erasing command
                MOV         IAP_ADDRL,DPL              ;Set IAP low address
                MOV         IAP_ADDRH,DPH              ;Set IAP high address
                MOV         IAP_TRIG,#5AH              ;Write trigger command (0x5a)
                MOV         IAP_TRIG,#0A5H             ;Write trigger command (0xa5)
                NOP
                LCALL       IAP_IDLE                   ;Disable IAP function
                RET

MAIN:
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                MOV         DPTR,#0400H
                LCALL       IAP_ERASE
                MOV         DPTR,#0400H
                LCALL       IAP_READ
                MOV         P0,A                       ;P0=0FFH
                MOV         DPTR,#0400H
                MOV         A,#12H
                LCALL           IAP_PROGRAM
                MOV             DPTR,#0400H
                LCALL           IAP_READ
                MOV             P1,A                            ;P1=12H

                SJMP            $

                END
```

16.4.3 Send Out the Data in EEPROM Using UART

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (65536 - FOSC / 115200 / 4)

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

sfr         AUXR            =       0x8e;
sfr         T2H             =       0xd6;
sfr         T2L             =       0xd7;

sfr         IAP_DATA        =       0xC2;
sfr         IAP_ADDRH       =       0xC3;
sfr         IAP_ADDRL       =       0xC4;
sfr         IAP_CMD         =       0xC5;
sfr         IAP_TRIG        =       0xC6;
sfr         IAP_CONTR       =       0xC7;
sfr         IAP_TPS         =       0xF5;

void UartInit()
{
        SCON = 0x5a;
        T2L = BRT;
        T2H = BRT >> 8;
        AUXR = 0x15;
}

void UartSend(char dat)
{
        while (!TI);
        TI = 0;
        SBUF = dat;
}

void IapIdle()
{
        IAP_CONTR = 0;                                  //Disable IAP function
        IAP_CMD = 0;                                    //Clear command register
        IAP_TRIG = 0;                                   //Clear trigger register
        IAP_ADDRH = 0x80;                               //Set the address to a non-IAP area
        IAP_ADDRL = 0;
}

char IapRead(int addr)
{
        char dat;

        IAP_CONTR = 0x80;                               //Enable IAP
        IAP_TPS = 12;                                   //Set the erasing wait parameter of 12MHz
        IAP_CMD = 1;                                    //Set IAP read command
        IAP_ADDRL = addr;                               //Set IAP low address
        IAP_ADDRH = addr >> 8;                          //Set IAP high address
        IAP_TRIG = 0x5a;                                //Write trigger command (0x5a)
        IAP_TRIG = 0xa5;                                //Write trigger command (0xa5)
        _nop_();
        dat = IAP_DATA;                                 //Read IAP data
        IapIdle();                                      //Disable IAP function

        return dat;
}

void IapProgram(int addr, char dat)
{
        IAP_CONTR = 0x80;                               //Enable IAP
        IAP_TPS = 12;                                   //Set the erasing wait parameter of 12MHz
        IAP_CMD = 2;                                    //Set IAP writing command
        IAP_ADDRL = addr;                               //Set IAP low address
        IAP_ADDRH = addr >> 8;                          //Set IAP high address
        IAP_DATA = dat;                                 //Write IAP data
        IAP_TRIG = 0x5a;                                //Write trigger command (0x5a)
        IAP_TRIG = 0xa5;                                //Write trigger command (0xa5)
        _nop_();
        IapIdle();                                      //Disable IAP function
}

void IapErase(int addr)
{
        IAP_CONTR = 0x80;                               //Enable IAP
        IAP_TPS = 12;                                   //Set the erasing wait parameter of 12MHz
        IAP_CMD = 3;                                    //Set IAP erasing command
        IAP_ADDRL = addr;                               //Set IAP low address
        IAP_ADDRH = addr >> 8;                          //Set IAP high address
        IAP_TRIG = 0x5a;                                //Write trigger command (0x5a)
        IAP_TRIG = 0xa5;                                //Write trigger command (0xa5)
        _nop_();                                        //
        IapIdle();                                      //Disable IAP function
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        UartInit();
        IapErase(0x0400);
        UartSEND(IapRead(0x0400));
        IapProgram(0x0400, 0x12);
        UartSEND(IapRead(0x0400));

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

AUXR        DATA        8EH
T2H         DATA        0D6H
T2L         DATA        0D7H

IAP_DATA    DATA        0C2H
IAP_ADDRH   DATA        0C3H
IAP_ADDRL   DATA        0C4H
IAP_CMD     DATA        0C5H
IAP_TRIG    DATA        0C6H
IAP_CONTR   DATA        0C7H
IAP_TPS     DATA        0F5H

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN

            ORG         0100H

UART_INIT:
            MOV         SCON,#5AH
            MOV         T2L,#0E8H                   ;65536-11059200/115200/4=0FFE8H
            MOV         T2H,#0FFH
```

```
                MOV             AUXR,#15H
                RET

UART_SEND:
                JNB             TI,$
                CLR             TI
                MOV             SBUF,A
                RET

IAP_IDLE:
                MOV             IAP_CONTR,#0                    ;Disable IAP function
                MOV             IAP_CMD,#0                      ;Clear command register
                MOV             IAP_TRIG,#0                     ;Clear trigger register
                MOV             IAP_ADDRH,#80H                  ;Set the address to a non-IAP area
                MOV             IAP_ADDRL,#0
                RET

IAP_READ:
                MOV             IAP_CONTR,#80H                  ;Enable IAP
                MOV             IAP_TPS,#12                     ;Set the erasing wait parameter of 12MHz
                MOV             IAP_CMD,#1                      ;Set IAP read command
                MOV             IAP_ADDRL,DPL                   ;Set IAP low address
                MOV             IAP_ADDRH,DPH                   ;Set IAP high address
                MOV             IAP_TRIG,#5AH                   ;Write trigger command (0x5a)
                MOV             IAP_TRIG,#0A5H                  ;Write trigger command (0xa5)
                NOP
                MOV             A,IAP_DATA                      ;Read IAP data
                LCALL           IAP_IDLE                        ;Disable IAP function
                RET

IAP_PROGRAM:
                MOV             IAP_CONTR,#80H                  ;Enable IAP
                MOV             IAP_TPS,#12                     ;Set the erasing wait parameter of 12MHz
                MOV             IAP_CMD,#2                      ;Set IAP writing command
                MOV             IAP_ADDRL,DPL                   ;Set IAP low address
                MOV             IAP_ADDRH,DPH                   ;Set IAP high address
                MOV             IAP_DATA,A                      ;Write IAP data
                MOV             IAP_TRIG,#5AH                   ;Write trigger command (0x5a)
                MOV             IAP_TRIG,#0A5H                  ;Write trigger command (0xa5)
                NOP
                LCALL           IAP_IDLE                        ;Disable IAP function
                RET

IAP_ERASE:
                MOV             IAP_CONTR,#80H                  ;Enable IAP
                MOV             IAP_TPS,#12                     ;Set the erasing wait parameter of 12MHz
                MOV             IAP_CMD,#3                      ;Set IAP erasing command
                MOV             IAP_ADDRL,DPL                   ;Set IAP low address
                MOV             IAP_ADDRH,DPH                   ;Set IAP high address
                MOV             IAP_TRIG,#5AH                   ;Write trigger command (0x5a)
                MOV             IAP_TRIG,#0A5H                  ;Write trigger command (0xa5)
                NOP
                LCALL           IAP_IDLE                        ;Disable IAP function
                RET

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                LCALL           UART_INIT
                MOV             DPTR,#0400H
                LCALL           IAP_ERASE
                MOV             DPTR,#0400H
                LCALL           IAP_READ
                LCALL           UART_SEND
                MOV             DPTR,#0400H
                MOV             A,#12H
                LCALL           IAP_PROGRAM
                MOV             DPTR,#0400H
                LCALL           IAP_READ
                LCALL           UART_SEND

                SJMP            $

                END
```

16.4.4 UART1 reads and writes EEPROM - Read using MOVC

C language code (main.c)

```
//Operating frequency for test is 11.0592MHz

/*      This program is completely normal after testing, and does not provide telephone technical support. If you cannot
understand it, please supplement the relevant basis yourself.  */

/************* 本程序功能说明      **************
STC8G family MCUs EEPROM general test program.
Please do not modify the program firstly, and download the "UART-EEPROM.hex" directly to test "02-UART 1 reads and
writes EEPROM- using MOVC to read". Select the frequency 11.0592MHZ when downloading.
PC serial port setting: baud rate 115200,8,n,1.
Do sector erase, write 64 bytes, and read 64 bytes of EEPROM.

Command example:
E 0   Perform sector erasing operation on EEPROM, E means erasing, the number 0 is 0 sector (decimal, 0~126, see the
specific IC).
W 0  Write operation to EEPROM, W means write, number 0 is 0 sector (decimal, 0~126, see the specific IC). Write 64 bytes
continuously from the start address of the sector.
R 0  Perform IAP read operation on EEPROM, R means read out, the number 0 is 0 sector (decimal, 0~126, see the specific IC).
Read 64 bytes continuously from the start address of the sector.
M 0  Perform MOVC read operation on EEPROM (operation address is sector*512+offset address), number 0 is sector 0
(decimal, 0~126, see the specific IC). Read 64 bytes continuously from the start address of the sector .


Note: For general purpose, the program does not recognize whether the sector is valid or not, and the user decides according to
the specific model.
```

```
Date: 2019-6-10
******************************************/

#include    "config.H"
#include    "EEPROM.h"

#define     Baudrate1                   115200L
#define     UART1_BUF_LENGTH            10
#define     EEADDR_OFFSET               (8 * 1024)      //Define the offset added when EEPROM is accessed using
MOVC
                                                        // Equal to the size of the FLASH ROM.
                                                        // For IAP or IRC at the beginning, the offset must be 0
#define     TimeOutSet1                 5

/************* local constant declaration    **************/
u8    code  T_Strings[]={"去年今日此门中，人面桃花相映红。人面不知何处去，桃花依旧笑春风。"};

/************* local variable declaration    **************/
u8    xdatatmp[70];
u8    xdataRX1_Buffer[UART1_BUF_LENGTH];
u8    RX1_Cnt;
u8    RX1_TimeOut;
bit   B_TX1_Busy;

/************* local function declaration    **************/
void  UART1_config(void);
void  TX1_write2buff(u8 dat);                           //write send buffer
void  PrintString1(u8 *puts);                           //send a string

/*************  External function and variable declarations ****************/

/*********************************************/

u8    CheckData(u8 dat)
{
      if((dat >= '0') && (dat <= '9'))      return (dat-'0');
      if((dat >= 'A') && (dat <= 'F'))      return (dat-'A'+10);
      if((dat >= 'a') && (dat <= 'f'))      return (dat-'a'+10);
      return 0xff;
}

u16   GetAddress(void)
{
      u16   address;
      u8    i;

      address = 0;
      if(RX1_Cnt <  3)return 65535;                   //error
      if(RX1_Cnt <= 5)                                  //Within 5 bytes is sector operation, decimal,
                                                        //Command supported:        E 0, E 12, E 120
                                                        //                      W 0, W 12, W 120
                                                        //                      R 0, R 12, R 120
      {
            for(i=2; i<RX1_Cnt; i++)
            {
                  if(CheckData(RX1_Buffer[i]) > 9)
                        return 65535;                   //error
                  address = address * 10 + CheckData(RX1_Buffer[i]);
```

```
        }
        if(address < 124)                           //Limited to sectors 0~123
        {
            address <<= 9;
            return (address);
        }
    }
    else if(RX1_Cnt == 8)                           //8 bytes direct address operation, hex,
                                                    //Command supported: E 0x1234, W 0x12b3, R 0x0A00
    {
        if((RX1_Buffer[2] == '0') && ((RX1_Buffer[3] == 'x') || (RX1_Buffer[3] == 'X')))
        {
            for(i=4; i<8; i++)
            {
                if(CheckData(RX1_Buffer[i]) > 0x0F)
                    return 65535;                   //error
                address = (address << 4) + CheckData(RX1_Buffer[i]);
            }
            if(address < 63488)
                return (address);                   //Limited to sectors 0~123
        }
    }

    return  65535;                                  //error
}

//========================================================================
// Function: void  delay_ms(u8 ms)
// Description: delay function
// Parameters: ms, the number of ms to delay, here only supports 1~255ms. Automatically adapt to the main clock.
// Return: none.
// Version: VER1.0
// Date: 2013-4-1
// Remark:
//========================================================================
void  delay_ms(u8 ms)
{
    u16 i;
    do
    {
        i = MAIN_Fosc / 10000;
        while(--i)   ;
    }while(--ms);
}

//Read EEPROM using MOVC
void EEPROM_MOVC_read_n(u16 EE_address, u8 *DataAddress, u16 number)
{
    u8   code  *pc;

    pc = EE_address + EEADDR_OFFSET;
    do
    {
        *DataAddress = *pc;                         //Data be read
        DataAddress++;
        pc++;
    }while(--number);
}
```

```
/******************** main function ***********************/
void main(void)
{
    u8   i;
    u16 addr;

    UART1_config();                             // select baud rate, 2: Use Timer2 as baud rate generator,
                                                //Other values: Use Timer1 as baud rate generator,
    EA = 1;                                     //Enable CPU interrupt

    PrintString1("STC8 familyMCU 用串口 1 测试 EEPROM 程序!\r\n");   //UART1send a string

    while(1)
    {
        delay_ms(1);
        if(RX1_TimeOut > 0)                     //timeout count
        {
            if(--RX1_TimeOut == 0)
            {
                if(RX1_Buffer[1] == ' ')
                {
                    addr = GetAddress();
                    if(addr < 63488)            //Limited to sectors 0~123
                    {
                        if(RX1_Buffer[0] == 'E')    //PC requests to erase a sector
                        {
                            EEPROM_SectorErase(addr);
                            PrintString1("扇区擦除完成!\r\n");
                        }

                        else if(RX1_Buffer[0] == 'W')       //PC request to write 64 bytes data to EEPROM
                        {
                            EEPROM_write_n(addr,T_Strings,64);
                            PrintString1("写入操作完成!\r\n");
                        }

                        else if(RX1_Buffer[0] == 'R')       //PC requests to return 64 bytes of EEPROM data
                        {
                            PrintString1("IAPData be read 如下：\r\n");
                            EEPROM_read_n(addr,tmp,64);
                            for(i=0; i<64; i++)
                                TX1_write2buff(tmp[i]);     // Return data to serial port
                            TX1_write2buff(0x0d);
                            TX1_write2buff(0x0a);
                        }
                        else if(RX1_Buffer[0] == 'M')       //PC requests to return 64 bytes of EEPROM data
                        {
                            PrintString1("MOVCData be read 如下：\r\n");
                            EEPROM_MOVC_read_n(addr,tmp,64);
                            for(i=0; i<64; i++)
                                TX1_write2buff(tmp[i]);     // Return data to serial port
                            TX1_write2buff(0x0d);
                            TX1_write2buff(0x0a);
                        }
                        else   PrintString1("命令错误!\r\n");
                    }
                    else   PrintString1("命令错误!\r\n");
                }
```

```
                        RX1_Cnt = 0;
                    }
                }
            }
}
/*********************************************************/

/*************** send a byte *******************************/
void TX1_write2buff(u8 dat)                         //write send buffer
{
        B_TX1_Busy = 1;                             //Set sending busy flag
        SBUF = dat;                                 //send a byte
        while(B_TX1_Busy);                          //wait for sending complish
}

//========================================================================
// Function: void PrintString1(u8 *puts)
// Description: UART1 send string function.
// Parameters:puts:  String pointer.
// Return: none.
// Version: VER1.0
// Date: 2014-11-28
// Remark:
//========================================================================
void PrintString1(u8 *puts)                         //send a string
{
    for (; *puts != 0;      puts++)             //End with stop 0
        {
                TX1_write2buff(*puts);
        }
}

//========================================================================
// Function: voidUART1_config(void)
// Description: UART1 initialization function.
// Parameters:none.
// Return: none.
// Version: VER1.0
// Date: 2014-11-28
// Remark:
//========================================================================
void  UART1_config(void)
{
        TR1 = 0;
        AUXR &= ~0x01;                              //S1 BRT Use Timer1;
        AUXR |=  (1<<6);                            //Timer1 set as 1T mode
        TMOD &= ~(1<<6);                            //Timer1 set As Timer
        TMOD &= ~0x30;                              //Timer1_16bitAutoReload;
        TH1 = (u8)((65536L-(MAIN_Fosc / 4) / Baudrate1) >> 8);
        TL1 = (u8)(65536L-(MAIN_Fosc / 4) / Baudrate1);
        ET1 = 0;                                    // Diable Timer1 interrupt
        INT_CLKO &= ~0x02;                          // Timer1 does not output high-speed clock
        TR1  = 1;                                   // Start Timer1

        S1_USE_P30P31();    P3n_standard(0x03);     //Switch to  P3.0 P3.1
        //S1_USE_P36P37();  P3n_standard(0xc0);     //Switch to  P3.6 P3.7
        //S1_USE_P16P17();  P1n_standard(0xc0);     //Switch to  P1.6 P1.7

        SCON = (SCON & 0x3f) | 0x40;                //UART1 mode,   0x00: synchronized shift output,
                                                    //                  0x40: 8-bit data, variable baud rate,
                                                    //                  0x80: 9-bit data, fixed baud rate,
                                                    //                  0xc0: 9-bit data, variable baud rate
//      PS  = 1;                                    //high priority interrupt
        ES  = 1;                                    //enable interrupt
        REN = 1;                                    //enable receiving

        B_TX1_Busy = 0;
        RX1_Cnt = 0;
}

//========================================================================
// Function: void UART1_int (void) interrupt UART1_VECTOR
// Description: UART1 interrupt function
// Parameters:nine.
// Return: none.
// Version: VER1.0
// Date: 2014-11-28
// Remark:
//========================================================================
void UART1_int (void) interrupt 4
{
        if(RI)
        {
                RI = 0;
                if(RX1_Cnt >= UART1_BUF_LENGTH)
                        RX1_Cnt = 0;                //avoid overflow
                RX1_Buffer[RX1_Cnt++] = SBUF;
                RX1_TimeOut = TimeOutSet1;
        }

        if(TI)
        {
                TI = 0;
                B_TX1_Busy = 0;
        }
}
```

C language code (EEPROM.c)

```
//Operating frequency for test is 11.0592MHz

//      This program is the built-in EEPROM read and write program of STC series.

#include "config.h"
#include "eeprom.h"

//========================================================================
// Function: voidISP_Disable(void)
// Description: Disable access ISP/IAP.
// Parameters:non.
// Return: non.
// Version: V1.0, 2012-10-22
//========================================================================
void  DisableEEPROM(void)
{
        ISP_CONTR = 0;                              //Disable ISP/IAP operation
        IAP_TPS   = 0;
        ISP_CMD   = 0;                      //Remove ISP/IAP commands
        ISP_TRIG  = 0;                      //Prevent false triggering of ISP/IAP commands
```

```
    ISP_ADDRH = 0xff;                           //Clear address high byte
    ISP_ADDRL = 0xff;                           //Clear address low byte, point to non-EEPROM area to prevent
misoperation
}

//========================================================================
// Function: void EEPROM_read_n(u16 EE_address,u8 *DataAddress,u16 number)
// Description: Read n bytes from the specified EEPROM first address and put them in the specified buffer.
// Parameters:EE_address:  The first address of the EEPROM to read.
//      DataAddress: The first address of the data buffer.
//      number:      The length of bytes to read.
// Return: non.
// Version: V1.0, 2012-10-22
//========================================================================
void EEPROM_read_n(u16 EE_address,u8 *DataAddress,u16 number)
{
    EA = 0;                                     //Disable interrupts
    ISP_CONTR = ISP_EN;                         //Allow ISP/IAP operation
    IAP_TPS = (u8)(MAIN_Fosc / 1000000L);       //Working frequency setting
    ISP_READ();                                 //Send byte read command, when the command does not need to
be changed, there is no need to send the command again
    do
    {
        ISP_ADDRH = EE_address / 256;           //Send the high byte of the address (the address needs to be re-
sent when the address needs to be changed)
        ISP_ADDRL = EE_address % 256;           //Send the low byte of the address
        ISP_TRIG();                             //Send 5AH firstly, then send A5H to the ISP/IAP trigger register,
                                                //Do this every time
                                                //After sending A5H, the ISP/IAP command is triggered to start
immediately
                                                //The CPU waits for the IAP to complete before continuing to
execute the program.
        _nop_();
        _nop_();
        _nop_();
        *DataAddress = ISP_DATA;                //Data be read
        EE_address++;
        DataAddress++;
    }while(--number);

    DisableEEPROM();
    EA = 1;                                     //re-enable interrupt
}

/******************* Sector Erase Function ****************/
//========================================================================
// Function: void EEPROM_SectorErase(u16 EE_address)
// Description: Erase the EEPROM sector at the specified address.
// Parameters:EE_address:  The address of the sector EEPROM to be erased.
// Return: non.
// Version: V1.0, 2013-5-10
//========================================================================
void EEPROM_SectorErase(u16 EE_address)
{
    EA = 0;                                     //Disable interrupts
                                                //Only sector erase, no byte erase, 512 bytes per sector.
                                                //Any byte address in a sector is sector address.
    ISP_ADDRH = EE_address / 256;               //Send the high byte of the sector address (the address needs to be
re-sent when the address needs to be changed)
```

```
        ISP_ADDRL = EE_address % 256;                  //Send the low byte of the sector address
        ISP_CONTR = ISP_EN;                            //Allow ISP/IAP operation
        IAP_TPS = (u8)(MAIN_Fosc / 1000000L);          //Working frequency setting
        ISP_ERASE();                                   //Send sector erase command. When the command does not need
to be changed, it is not necessary to send the command again
        ISP_TRIG();
        _nop_();
        _nop_();
        _nop_();
        DisableEEPROM();
        EA = 1;                                        //re-enable interrupt
}

//========================================================================
// Function: void EEPROM_write_n(u16 EE_address,u8 *DataAddress,u16 number)
// Description: Write the buffered n bytes into the EEPROM of the specified first address.
// Parameters:EE_address:  Write the first address of the EEPROM.
//      DataAddress: The first address of the buffer where the source data is written.
//      number:      The length of bytes written.
// Return: non.
// Version: V1.0, 2012-10-22
//========================================================================
void EEPROM_write_n(u16 EE_address,u8 *DataAddress,u16 number)
{
        EA = 0;                                        //Disable interrupts

        ISP_CONTR = ISP_EN;                            //Allow ISP/IAP operation
        IAP_TPS = (u8)(MAIN_Fosc / 1000000L);          //Working frequency setting
        ISP_WRITE();                                   //Send byte write command. When the command does not need to
be changed, no need to send the command again
        do
        {
                ISP_ADDRH = EE_address / 256;          //Send the high byte of the address (the address needs to be re-
sent when the address needs to be changed)
                ISP_ADDRL = EE_address % 256;          //Send the low byte of the address
                ISP_DATA  = *DataAddress;              //Send data to ISP_DATA, and only need to send it again when
the data changes.
                ISP_TRIG();
                _nop_();
                _nop_();
                _nop_();
                EE_address++;
                DataAddress++;
        }while(--number);

        DisableEEPROM();
        EA = 1;                                        //re-enable interrupt
}
```

16.4.5 Password erasing and writing - multi-sector backup - UART1 operation

C language code (main.c)

```
//Operating frequency for test is 11.0592MHz
```

```
/*      This program is completely normal after testing, and does not provide telephone technical support. If you cannot understand it, please supplement the relevant basis yourself.   */

/************* Function description of this program     **************
STC8G family, STC8H family and STC8C family’s EEPROM general test program to demonstrate multi-sector backup, writing with correct sector data if there is a sector error, and writing the default value for all sector errors (such as the first time the program is run).

Each writting writes 3 sectors, that is, redundant backup.
Write a record in each sector, after the writing is completed, read the saved data and check value and compare it with the source data and check value, and return the result (correct or wrong) from UART1 (P3.0 P3.1) .
Each record is self-checked, 64-byte data, 2-byte check value, check value = 64 bytes data’s cumulative sum ^ 0x5555. ^0x5555 is to ensure that the written 66 data are not all 0.
If there is a sector error, the data of the correct sector will be written to the wrong sector, and if all three sectors are wrong, the default value will be written.
A password needs to be set before erasing, writing, and reading operations. If the password is incorrect, the operation will be exited, and the password will be cleared each time when an exit operation is performed.

Please do not modify the program firstly, and download the "UART-EEPROM.hex" directly to test "03-Password erasing and writing-multi-sector backup-UART 1 operation ". Select the frequency 11.0592MHZ when downloading.

PC serial port setting: baud rate 115200,8,n,1.
Do sector erase, write 64 bytes, and read 64 bytes of EEPROM.

Command example:
Use the serial port assistant to send a single character, both upper and lower case.

E or e:  Perform sector erase operation on EEPROM, E means erase, it will erase sectors 0, 1, 2.

W or w:  Write operation to EEPROM, W means write, will write to sectors 0, 1, 2, each sector writes 64 bytes continuously, sector 0 writes 0x0000~0x003f, sector 1 writes 0x0200~0x023f , write 0x0400~0x043f in sector 0.

R or r:  Read data from the EEPROM, R means read, it will read sectors 0, 1, 2, each sector reads 64 bytes continuously, sector 0 reads 0x0000~0x003f, sector 1 reads 0x0200~0x023f , Sector 0 reads 0x0400~0x043f.

Note: For general purpose, the program does not recognize whether the sector is valid or not, and the user decides according to the specific model.

Date: 2021-11-5
******************************************/

#include    "config.H"
#include    "EEPROM.h"

#define             Baudrate1               115200L

/************* local constant declaration    **************/
u8    code  T_StringD[]={"去年今日此门中，人面桃花相映红。人面不知何处去，桃花依旧笑春风。"};
u8    code  T_StringW[]={"横看成岭侧成峰，远近高低各不同。不识庐山真面目，只缘身在此山中。"};


/************* local variable declaration    **************/
u8    xdatatmp[70];                                 //General data
u8    xdataSaveTmp[70];                             // array to write

bit   B_TX1_Busy;
u8    cmd;                                          // single character command of UART
```

```
/************* local function declaration    **************/
void  UART1_config(void);
void  TX1_write2buff(u8 dat);                           //write send buffer
void  PrintString1(u8 *puts);                           //send a string


/*************  External function and variable declarations *****************/

/*************    Read  the  EEPROM  record,  and  verify,  return  the  verification  result,  0  is  correct,  1  is
wrong*****************/
u8    ReadRecord(u16 addr)
{
        u8    i;
        u16   ChckSum;                                  //Calculated cumulative sum
        u16   j;                                        //Accumulated sum of reading

        for(i=0; i<66; i++)      tmp[i] = 0;            //clear buffer
        PassWord = D_PASSWORD;                          //given password
        EEPROM_read_n(addr,tmp,66);                     //read sector 0
        for(ChckSum=0, i=0; i<64; i++)
                ChckSum += tmp[i];                      //Calculate the cumulative sum
        j = ((u16)tmp[64]<<8) + (u16)tmp[65];           //Read cumulative sum of records
        j ^= 0x5555;                                    //Invert every other bit, avoid all 0s
        if(ChckSum != j)        return 1;               //If the cumulative sum is wrong, return 1
        return       0;                                 // If the cumulative sum is right, return 0
}

/*************   Write  the  EEPROM  record,  and  verify,  return  the  verification  result,  0  is  correct,  1  is  wrong
*****************/
u8    SaveRecord(u16 addr)
{
        u8    i;
        u16   ChckSum;                                  //Calculated cumulative sum

        for(ChckSum=0, i=0; i<64; i++)
                ChckSum += SaveTmp[i];                  //Calculate the cumulative sum
        ChckSum ^= 0x5555;                              //Invert every other bit, avoid all 0s
        SaveTmp[64] = (u8)(ChckSum >> 8);
        SaveTmp[65] = (u8)ChckSum;

        PassWord = D_PASSWORD;                          //given password
        EEPROM_SectorErase(addr);                       //Erase a sector
        PassWord = D_PASSWORD;                          //given password
        EEPROM_write_n(addr, SaveTmp, 66);              //write a sector

        for(i=0; i<66; i++)
                tmp[i] = 0;                             //clear buffer
        PassWord = D_PASSWORD;                          //given password
        EEPROM_read_n(addr,tmp,66);                     //read sector 0
        for(i=0; i<66; i++)                             //data comparison
        {
                if(SaveTmp[i] != tmp[i])
                        return 1;                       //If there is an error in the data, return 1
        }
        return 0;                                       // If the cumulative sum is right, return 0
}

/******************** main function ***********************/
void main(void)
```

```
{
        u8      i;
        u8      status;                                         //Statuc

        UART1_config();                                         // select baud rate, 2: Use Timer2 as baud rate generator,
                                                                //Other values: Use Timer1 as baud rate generator,
        EA = 1;                                                 //Enable CPU interrupt

        PrintString1("STC8G-8H-8C familyMCU 用串口 1 测试 EEPROM 程序!\r\n");      //UART1send a string

                                                                // Power on and read 3 sectors and verify, if there is a sector error,
write the correct sector into the wrong sector, if all 3 sectors are wrong, write the default value.
        status = 0;
        if(ReadRecord(0x0000) == 0)                             //read sector 0
        {
                status |= 0x01;                                 //If it is correct, mark status.0=1
                for(i=0; i<64; i++)
                        SaveTmp[i] = tmp[i];                    //save data in write buffer
        }
        if(ReadRecord(0x0200) == 0)                             //Read sector 1
        {
                status |= 0x02;                                 //If it is correct, mark status.1=1
                for(i=0; i<64; i++)
                        SaveTmp[i] = tmp[i];                    //save data in write buffer
        }
        if(ReadRecord(0x0400) == 0)                             //Read sector 2
        {
                status |= 0x04;                                 //If it is correct, mark status.2=1
                for(i=0; i<64; i++)
                        SaveTmp[i] = tmp[i];                    //save data in write buffer
        }

        if(status == 0)                                         //If all sectors are wrong, write default value
        {
                for(i=0; i<64; i++)
                        SaveTmp[i] = T_StringD[i];              //Read the default value
        }
        else PrintString1("上电读取 3 个扇区数据均正确!\r\n");          //UART1 send a string prompt

        if((status & 0x01) == 0)                                // If sector 0 is wrong, write default value
        {
                if(SaveRecord(0x0000) == 0)
                        PrintString1("write a sector0 正确!\r\n");          // Writing record 0 sector is correct
                else
                        PrintString1("write a sector0 错误!\r\n");          // Writing record 0 sector is wrong
        }
        if((status & 0x02) == 0)                                // If sector 1 is wrong, write default value
        {
                if(SaveRecord(0x0200) == 0)
                        PrintString1("write a sector1 正确!\r\n");          // Writing record 1 sector is correct
                else
                        PrintString1("write a sector1 错误!\r\n");          // Writing record 1 sector is wrong
        }
        if((status & 0x04) == 0)                                // If sector 2 is wrong, write default value
        {
                if(SaveRecord(0x0400) == 0)
                        PrintString1("write a sector2 正确!\r\n");          // Writing record 2 sector is correct
                else
                        PrintString1("write a sector2 错误!\r\n");          // Writing record 2 sector is wrong
```

```
    }


    while(1)
    {
        if(cmd != 0)                                // There is UART commands
        {
            if((cmd >= 'a') && (cmd <= 'z'))
                cmd -= 0x20;                        // lowercase to uppercase

            if(cmd == 'E')                          //PC requests to erase a sector
            {
                PassWord = D_PASSWORD;              //given password
                EEPROM_SectorErase(0x0000);         //Erase a sector
                PassWord = D_PASSWORD;              //given password
                EEPROM_SectorErase(0x0200);         //Erase a sector
                PassWord = D_PASSWORD;              //given password
                EEPROM_SectorErase(0x0400);         //Erase a sector
                PrintString1("扇区擦除完成!\r\n");
            }

            else if(cmd == 'W')                                 //PC request to write 64 bytes data to EEPROM
            {
                for(i=0; i<64; i++)
                    SaveTmp[i] = T_StringW[i];                  //write value
                if(SaveRecord(0x0000) == 0)
                    PrintString1("write a sector0 正确!\r\n");          // Writing record 0 sector is correct
                else
                    PrintString1("write a sector0 错误!\r\n");          // Writing record 0 sector is wrong
                if(SaveRecord(0x0200) == 0)
                    PrintString1("write a sector1 正确!\r\n");          // Writing record 1 sector is correct
                else
                    PrintString1("write a sector1 错误!\r\n");          // Writing record 1 sector is wrong
                if(SaveRecord(0x0400) == 0)
                    PrintString1("write a sector2 正确!\r\n");          // Writing record 2 sector is correct
                else
                    PrintString1("write a sector2 错误!\r\n");          // Writing record 2 sector is wrong
            }

            else if(cmd == 'R')                                 //PC requests to return 64 bytes of EEPROM data
            {
                if(ReadRecord(0x0000) == 0)                     //read data of sector 0
                {
                    PrintString1("read sector 0 的数据如下：\r\n");
                    for(i=0; i<64; i++)
                        TX1_write2buff(tmp[i]);                 // Return data to UART
                    TX1_write2buff(0x0d);                       // carriage return and line feed
                    TX1_write2buff(0x0a);
                }
                else    PrintString1("read sector 0 的数据错误!\r\n");

                if(ReadRecord(0x0200) == 0)                     // Read data of sector 1
                {
                    PrintString1("读出扇区1的数据如下：\r\n");
                    for(i=0; i<64; i++)
                        TX1_write2buff(tmp[i]);                 // Return data to UART
                    TX1_write2buff(0x0d);                       // carriage return and line feed
                    TX1_write2buff(0x0a);
                }
                else  PrintString1("读出扇区 1 的数据错误!\r\n");

                if(ReadRecord(0x0400) == 0)                 // Read data of sector 2
                {
                    PrintString1("读出扇区 2 的数据如下：\r\n");
                    for(i=0; i<64; i++)
                        TX1_write2buff(tmp[i]);             // Return data to UART
                    TX1_write2buff(0x0d);                   // carriage return and line feed
                    TX1_write2buff(0x0a);
                }
                else  PrintString1("读出扇区 2 的数据错误!\r\n");
            }
            else  PrintString1("命令错误!\r\n");
            cmd = 0;
        }
    }
}
/**********************************************/

/*************** send a byte ******************************/
void TX1_write2buff(u8 dat)                         //write send buffer
{
    B_TX1_Busy = 1;                                 //Set sending busy flag
    SBUF = dat;                                     //send a byte
    while(B_TX1_Busy);                              //wait for sending complish
}

//========================================================================
// Function: void PrintString1(u8 *puts)
// Description: UART1 send string function.
// Parameters:puts:  String pointer.
// Return: none.
// Version: VER1.0
// Date: 2014-11-28
// Remark:
//========================================================================
void PrintString1(u8 *puts)                         //send a string
{
  for (; *puts != 0;     puts++)                    //End with stop 0
    {
        TX1_write2buff(*puts);
    }
}

//========================================================================
// Function: voidUART1_config(void)
// Description: UART1 initialization function.
// Parameters:none.
// Return: none.
// Version: VER1.0
// Date: 2014-11-28
// Remark:
//========================================================================
void  UART1_config(void)
{
    TR1 = 0;
    AUXR &= ~0x01;                              //S1 BRT Use Timer1;
    AUXR |=  (1<<6);                            //Timer1 set as 1T mode
    TMOD &= ~(1<<6);                            //Timer1 set As Timer
```

```
        TMOD &= ~0x30;                                  //Timer1_16bitAutoReload;
        TH1 = (u8)((65536L-(MAIN_Fosc / 4) / Baudrate1) >> 8);
        TL1 = (u8)(65536L-(MAIN_Fosc / 4) / Baudrate1);
        ET1 = 0;                                        // Disable Timer1 interrupt
        INT_CLKO &= ~0x02;                              // Timer1 does not output high-speed clock
        TR1  = 1;                                       // Start Timer1

        S1_USE_P30P31();    P3n_standard(0x03);         //Switch to  P3.0 P3.1
        //S1_USE_P36P37();  P3n_standard(0xc0);         //Switch to  P3.6 P3.7
        //S1_USE_P16P17();  P1n_standard(0xc0);         //Switch to  P1.6 P1.7

        SCON = (SCON & 0x3f) | 0x40;                    //UART1 mode,  0x00: synchronized shift output,
                                                        //             0x40: 8-bit data, variable baud rate,
                                                        //             0x80: 9-bit data, fixed baud rate,
                                                        //             0xc0: 9-bit data, variable baud rate
//      PS  = 1;                                        //high priority interrupt
        ES  = 1;                                        //enable interrupt
        REN = 1;                                        //enable receiving

        B_TX1_Busy = 0;
}

//========================================================================
// Function: void UART1_int (void) interrupt UART1_VECTOR
// Description: UART1 interrupt function
// Parameters:nine.
// Return: none.
// Version: VER1.0
// Date: 2014-11-28
// Remark:
//========================================================================
void UART1_int (void) interrupt 4
{
        if(RI)
        {
                RI = 0;
                cmd = SBUF;
        }

        if(TI)
        {
                TI = 0;
                B_TX1_Busy = 0;
        }
}
```

C language code (EEPROM.c)

```
//Operating frequency for test is 11.0592MHz

//      This program is the built-in EEPROM read and write program of STC series.

#include    "config.h"
#include    "EEPROM.h"

u32         PassWord;                                   // Password required for erasing and writing

//========================================================================
// Function: voidISP_Disable(void)
// Description: Disable access ISP/IAP.
// Parameters:non.
// Return: non.
// Version: V1.0, 2012-10-22
//========================================================================
void  DisableEEPROM(void)
{
        ISP_CONTR = 0;                                          //Disable ISP/IAP operation
        IAP_TPS   = 0;
        ISP_CMD   = 0;                                  //Remove ISP/IAP commands
        ISP_TRIG  = 0;                                  //Prevent false triggering of ISP/IAP commands
        ISP_ADDRH = 0xff;                                       //Clear address high byte
        ISP_ADDRL = 0xff;                                       //Clear address low byte, point to non-EEPROM area to prevent
misoperation
}

//========================================================================
// Function: void EEPROM_read_n(u16 EE_address,u8 *DataAddress,u16 number)
// Description: Read n bytes from the specified EEPROM first address and put them in the specified buffer.
// Parameters:EE_address:  The first address of the EEPROM to read.
//      DataAddress: The first address of the data buffer.
//      number:      The length of bytes to read.
// Return: non.
// Version: V1.0, 2012-10-22
//========================================================================
void EEPROM_read_n(u16 EE_address,u8 *DataAddress,u16 number)
{
        if(PassWord == D_PASSWORD)                              // Only the correct password is allowed to operate the EEPROM
        {
                EA = 0;                                         //Disable interrupts
                ISP_CONTR = ISP_EN;                             //Allow ISP/IAP operation
                IAP_TPS = (u8)(MAIN_Fosc / 1000000L);           //Working frequency setting
                ISP_READ();                                     //Send byte read command, when the command does not need to
be changed, there is no need to send the command again
                do
                {
                        ISP_ADDRH = EE_address / 256;           //Send the high byte of the address (the address needs to be re-
sent when the address needs to be changed)
                        ISP_ADDRL = EE_address % 256;           //Send the low byte of the address
                        if(PassWord == D_PASSWORD)              // If the password is correct, trigger the operation
                        {
                                ISP_TRIG = 0x5A;                //Send 5AH firstly, then send A5H to the ISP/IAP trigger register,
                                                                //Do this every time
                                ISP_TRIG = 0xA5;                //After sending A5H, the ISP/IAP command is triggered to start
immediately
                        }                                       //The CPU waits for the IAP to complete before continuing to
execute the program.
                        _nop_();
                        _nop_();
                        _nop_();
                        *DataAddress = ISP_DATA;                //Data be read
                        EE_address++;
                        DataAddress++;
                }while(--number);

                DisableEEPROM();
                EA = 1;                                         //re-enable interrupt
        }
        PassWord = 0;                                           //clear password
}
```

```
/******************* Sector Erase Function ****************/
//========================================================================
// Function: void EEPROM_SectorErase(u16 EE_address)
// Description: Erase the EEPROM sector at the specified address.
// Parameters:EE_address:  The address of the sector EEPROM to be erased.
// Return: non.
// Version: V1.0, 2013-5-10
//========================================================================
void EEPROM_SectorErase(u16 EE_address)
{
        if(PassWord == D_PASSWORD)                      // Only the password is correct, the EEPROM will be operated
        {
                EA = 0;                                 //Disable interrupts
                                                        //Only sector erase, no byte erase, 512 bytes per sector.
                                                        //Any byte address in a sector is sector address.
                ISP_ADDRH = EE_address / 256;           //Send the high byte of the sector address (the address needs to be
re-sent when the address needs to be changed)
                ISP_ADDRL = EE_address % 256;           //Send the low byte of the sector address
                ISP_CONTR = ISP_EN;                     //Allow ISP/IAP operation
                IAP_TPS = (u8)(MAIN_Fosc / 1000000L);   //Working frequency setting
                ISP_ERASE();                            //Send sector erase command. When the command does not need
to be changed, it is not necessary to send the command again
                        if(PassWord == D_PASSWORD)      // If the password is correct, trigger the operation
                        {
                                ISP_TRIG = 0x5A;        //Send 5AH firstly, then send A5H to the ISP/IAP trigger register,
                                                        //Do this every time
                                ISP_TRIG = 0xA5;        //After sending A5H, the ISP/IAP command is triggered to start
immediately
                        }                               //The CPU waits for the IAP to complete before continuing to
execute the program.
                _nop_();
                _nop_();
                _nop_();
                DisableEEPROM();
                EA = 1;                                 //re-enable interrupt
        }
        PassWord = 0;                                   //Clear password
}

//========================================================================
// Function: void EEPROM_write_n(u16 EE_address,u8 *DataAddress,u16 number)
// Description: Write the buffered n bytes into the EEPROM of the specified first address.
// Parameters:EE_address:  Write the first address of the EEPROM.
//      DataAddress: The first address of the buffer where the source data is written.
//      number:      The length of bytes written.
// Return: non.
// Version: V1.0, 2012-10-22
//========================================================================
void EEPROM_write_n(u16 EE_address,u8 *DataAddress,u16 number)
{
        if(PassWord == D_PASSWORD)                      // Only the password is correct, the EEPROM will be operated
        {
                EA = 0;                                 //Disable interrupts

                ISP_CONTR = ISP_EN;                     //Allow ISP/IAP operation
                IAP_TPS = (u8)(MAIN_Fosc / 1000000L);   //Working frequency setting
                ISP_WRITE();                            //Send byte write command. When the command does not need to
be changed, no need to send the command again
```

```
        do
        {
            ISP_ADDRH = EE_address / 256;           //Send the high byte of the address (the address needs to be re-
sent when the address needs to be changed)
            ISP_ADDRL = EE_address % 256;           //Send the low byte of the address
            ISP_DATA  = *DataAddress;               //Send data to ISP_DATA, and only need to send it again when
the data changes.
            if(PassWord == D_PASSWORD)              // If the password is correct, trigger the operation
            {
                ISP_TRIG = 0x5A;                    //Send 5AH firstly, then send A5H to the ISP/IAP trigger register,
                                                    //Do this every time
                ISP_TRIG = 0xA5;                    //After sending A5H, the ISP/IAP command is triggered to start
immediately
            }                                       //The CPU waits for the IAP to complete before continuing to
execute the program.
            _nop_();
            _nop_();
            _nop_();
            EE_address++;
            DataAddress++;
        }while(--number);

        DisableEEPROM();
        EA = 1;                                     //re-enable interrupt
    }
    PassWord = 0;                                   // Clear password
}
```

17 ADC, Internal Reference Voltage

Product line	Resolution of ADC	Channels of ADC
STC8H1K08 family	**10 bit**	**9 channels**
STC8H1K28 family	**10 bit**	**12 channels**
STC8H3K64S4 family	**12 bit**	**12 channels**
STC8H3K64S2 family	**12 bit**	**12 channels**
STC8H8K64U family	**12 bit**	**15 channels**
STC8H2K64T family	**12 bit**	**15 channels**
STC8H4K64TLR family	**12 bit**	**15 channels**
STC8H4K64TLCD family	**12 bit**	**15 channels**
STC8H4K64LCD family	**12 bit**	**15 channels**

A 10-bit/12-bit high-speed Analog to Digital Converter is integrated in STC8H family of microcontrollers. The system frequency is divided by 2 and then divided again by the user-set division ratio as the clock frequency of the ADC. The range of ADC clock frequency is SYSclk/2/1 ~ SYSclk/2/16.

The fastest ADC speed of STC8H series: 12-bit ADC is 800K (800,000 ADC conversions per second), 10-bit ADC is 500K (500,000 ADC conversions per second).

There are two data formats for ADC conversion results: Align left and Align right. It is convenient for user program to read and reference.

Note: The 15th channel of the ADC can only be used to detect the internal reference voltage. The reference voltage value is calibrated to 1.19V at the factory. Due to the manufacturing errors and measurement errors, the actual internal reference voltage has about ±1% error compared to 1.19V. If you want to know the exact internal reference voltage of each chip, you can connect an accurate reference voltage and then use the 15th channel of the ADC to measure the calibration.

If the chip has ADC external reference power supply pin ADC_Vref+, it must not be floating, it must be connected to an external reference power supply or directly connected to VCC

17.1 Registers Related to ADC

Symbol	Description	Address	Bit Address and Symbol								Reset Value
			B7	B6	B5	B4	B3	B2	B1	B0	
ADC_CONTR	ADC control register	BCH	ADC_POWER	ADC_START	ADC_FLAG	ADC_EPWMT	ADC_CHS[3:0]				000x,0000
ADC_RES	ADC Result High Byte	BDH									0000,0000
ADC_RESL	ADC Result Low Byte	BEH									0000,0000
ADCCFG	ADC Configuration Register	DEH	-	-	RESFMT	-	SPEED[3:0]				xx0x,0000
ADCTIM	ADC Timing Control Register	FEA8H	CSSETUP	CSHOLD[1:0]		SMPDUTY[4:0]					0010,1010

17.1.1 ADC control register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
ADC_CONTR	BCH	ADC_POWER	ADC_START	ADC_FLAG	ADC_EPWMT	ADC_CHS[3:0]			

ADC_POWER: ADC power supply control bit.

0: turn off the power supply of ADC.

1: turn on the power supply of ADC.

It is recommended to turn off the ADC before entering Idle mode and Power-down mode to reduce the power consumption.

Pay attention:

1. After the power supply to the internal ADC module of the MCU is turned on, wait for about 1ms, and wait for the ADC power supply inside the MCU to stabilize before allowing the ADC to work;

2. Properly lengthening the sampling time of the external signal is the charging or discharging time of the internal sampling and holding capacitor of the ADC. If the time is enough, the internal can be equal to the

external potential.

ADC_START: ADC start bit. ADC conversion will start after write 1 to this bit. It is cleared automatically by the hardware after A/D conversion completes.

0: no effect. Writing 0 to this bit will not stop the A/D conversion if the ADC has already started.

1: start the A/D conversion. It is cleared automatically by the hardware after A/D conversion completes.

ADC_FLAG: ADC conversion completement flag. It is set by the hardware after the ADC conversion hasfinished, and requests interrupt to CPU. It must be cleared by software.

ADC_EPWMT: enable PWM synchronous trigger ADC function.

ADC_CHS[3:0]: ADC anolog channel selection bits.

(Note: The I/O port selected as the ADC input channel must be set to the PxM0/PxM1 register to set the I/O port mode to high-impedance input mode. In addition, if the MCU enters the power-down mode/clock stop mode, it still needs To enable the ADC channel, you need to set the PxIE register to close the digital input channel to prevent the external analog input signal from fluctuating high and low and causing additional power consumption)

(STC8H1K28family)

ADC_CHS[3:0]	ADC channel
0000	P1.0/ADC0
0001	P1.1/ADC1
0010	P1.2/ADC2
0011	P1.3/ADC3
0100	P1.4/ADC4
0101	P1.5/ADC5
0110	P1.6/ADC6
0111	P1.7/ADC7
1000	P0.0/ADC8
1001	P0.1ADC9
1010	P0.2/ADC10
1011	P0.3/ADC11
1100	No such channel
1101	No such channel
1110	No such channel
1111	Test internal 1.19V

(STC8H1K08 family)

ADC_CHS[3:0]	ADC channel
0000	P1.0/ADC0
0001	P1.1/ADC1
0010	No such channel
0011	No such channel
0100	No such channel
0101	No such channel
0110	No such channel
0111	No such channel
1000	P3.0/ADC8
1001	P3.1ADC9
1010	P3.2/ADC10
1011	P3.3/ADC11
1100	P3.4/ADC12
1101	P3.5/ADC13
1110	P3.6/ADC14
1111	Test internal 1.19V

(STC8H3K64S4/STC8H3K64S2 family)

ADC_CHS[3:0]	ADC channel
0000	P1.0/ADC0
0001	P1.1/ADC1
0010	P1.2/ADC2

0011	No such channel
0100	No such channel
0101	No such channel
0110	P1.6/ADC6
0111	P1.7/ADC7
1000	P0.0/ADC8
1001	P0.1ADC9
1010	P0.2/ADC10
1011	P0.3/ADC11
1100	P0.4/ADC12
1101	P0.5/ADC13
1110	P0.6/ADC14
1111	Test internal 1.19V

(STC8H8K64U, C8H2K64T, STC8H4K64TLR family)

ADC_CHS[3:0]	ADC channel
0000	P1.0/ADC0
0001	P1.1/ADC1
0010	P5.4/ADC2
0011	P1.3/ADC3
0100	P1.4/ADC4
0101	P1.5/ADC5
0110	P1.6/ADC6
0111	P1.7/ADC7
1000	P0.0/ADC8
1001	P0.1ADC9
1010	P0.2/ADC10
1011	P0.3/ADC11
1100	P0.4/ADC12
1101	P0.5/ADC13
1110	P0.6/ADC14
1111	Test internal 1.19V

(TC8H4K64TLCD, C8H4K64LCD family)

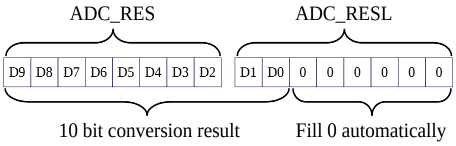
ADC_CHS[3:0]	ADC channel
0000	P1.0/ADC0
0001	P1.1/ADC1
0010	P5.4/ADC2
0011	P1.3/ADC3
0100	P1.4/ADC4
0101	P1.5/ADC5
0110	P6.2/ADC6
0111	P6.3/ADC7
1000	P0.0/ADC8
1001	P0.1/ADC9
1010	P0.2/ADC10
1011	P0.3/ADC11
1100	P0.4/ADC12
1101	P0.5/ADC13
1110	P0.6/ADC14
1111	Test internal 1.19V

17.1.2 ADC configuration register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
ADCCFG	DEH	-	-	RESFMT	-	SPEED[3:0]			

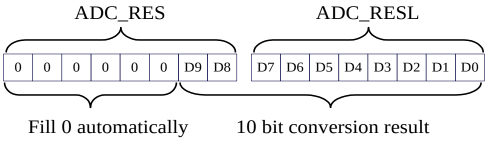
RESFMT: ADC conversion result format control bit (STC8H1K28family, C8H1K08family)

0: The conversion result aligns left. ADC_RES is used to save the upper 8 bits of the result and ADC_RESL is used to save the lower 2 bits of the result. The format is as follows:



RESFMT=0

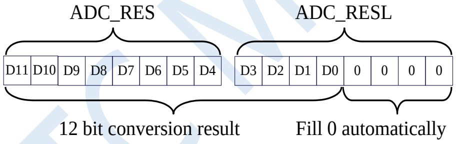
1: The conversion result aligns right. ADC_RES is used to save the upper 2 bits of the result and ADC_RESL is used to save the lower 8 bits of the result. The format is as follows:



RESFMT=1

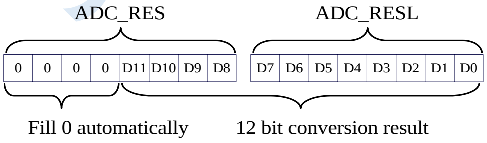
RESFMT: ADC conversion result format control bit (STC8H3K64S4 family, C8H3K64S2 family, STC8H8K64U family, STC8H2K64T family, **STC8H4K64TLR** family**, STC8H4K64TLCD** family**, STC8H4K64LCD** family)

0: The conversion result aligns left. ADC_RES is used to save the upper 8 bits of the result and ADC_RESL is used to save the lower 4 bits of the result. The format is as follows:



RESFMT=0

1: The conversion result aligns right. ADC_RES is used to save the upper 4 bits of the result and ADC_RESL is used to save the lower 8 bits of the result. The format is as follows:



RESFMT=1

SPEED[3:0]: ADC clock control bits {F_{ADC}=SYSclk/2/(SPEED+1)}

SPEED[3:0]	ADC clock frequency
0000	SYSclk/2/1
0001	SYSclk/2/2
0010	SYSclk/2/3
...	. . .
1101	SYSclk/2/14
1110	SYSclk/2/15
1111	SYSclk/2/16

17.1.3 ADC result registers (ADC_RES, ADC_RESL)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
ADC_RES	BDH								
ADC_RESL	BEH								

After the A/D conversion is completed, the 10-bit/12-bit conversion result is automatically saved to ADC_RES and ADC_RESL. Please refer to the RESFMT setting in the ADC_CFG register to see the result's data format.

17.1.4 ADC timing control register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
ADCTIM	FEA8H	CSSETUP	CSHOLD[1:0]		SMPDUTY[4:0]				

CSSETUP: ADC channel selection time control T_{setup}

CSSETUP	ADC number of clocks
0	**1 (default)**
1	2

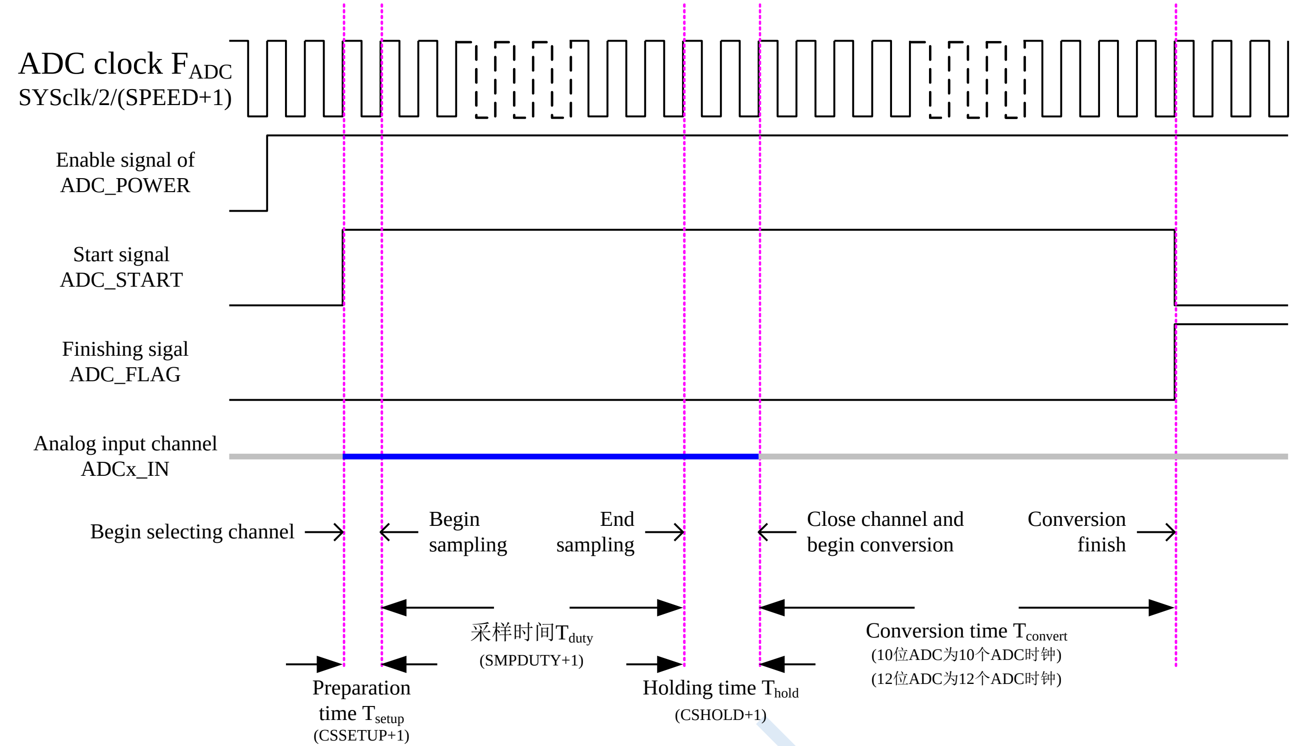
CSHOLD[1:0]: ADC Channel selection hold time control T_{hold}

CSHOLD[1:0]	ADC number of clocks
00	1
01	**2 (default)**
10	3
11	4

SMPDUTY[4:0]: ADC analog signal sampling time control T_{duty} (Note: SMPDUTY must not be set less than 01010B)

SMPDUTY[4:0]	ADC number of clocks
00000	1
00001	2
...	...
01010	**11 (default)**
...	...
11110	31
11111	32

ADC digital-to-analog conversion time: $T_{convert}$
The conversion time of 10-bit ADC is fixed at 10 ADC working clocks
The conversion time of 12-bit ADC is fixed at 12 ADC working clocks
A complete ADC conversion time is: Tsetup + Tduty + Thold + Tconvert, as shown in the figure below



The whole timing Diagram Of ADC

17.2 ADC related calculation formula

17.2.1 ADC speed calculation formula

The ADC conversion speed is controlled by the SPEED and ADCTIM registers in the ADCCFG register. The calculation formula of the conversion speed is as follows:

$$\mathbf{10bit\ ADC\ conversion\ speed} = \frac{\mathit{MCU\ operating\ frequency\ SYSclk}}{2 \times (\mathbf{SPEED}[3:0] + 1) \times [(\mathbf{CSSETUP} + 1) + (\mathbf{CSHOLD} + 1) + (\mathbf{SMPDUTY} + 1) + 10]}$$

$$\mathbf{12bit\ ADC\ conversion\ speed} = \frac{\mathit{MCU\ operating\ frequency\ SYSclk}}{2 \times (\mathbf{SPEED}[3:0] + 1) \times [(\mathbf{CSSETUP} + 1) + (\mathbf{CSHOLD} + 1) + (\mathbf{SMPDUTY} + 1) + 10]}$$

Note:

- ● The speed of 10-bit ADC cannot be higher than 500KHz
- ■ The speed of 12-bit ADC cannot be higher than 800KHz
- ■ The value of SMPDUTY cannot be less than 10, it is recommended to set to 15
- ■ CSSETUP can use power-on default value 0
- ■ CHOLD can use the power-on default value 1 (ADCTIM is recommended to be set to 3FH)

17.2.2 ADC conversion result calculation formula

$$\text{10bit ADC conversion result} = 1024 \,|\, \frac{\text{The input voltage Vin of the ADC converted channel}}{(\text{MCU working voltage Vcc})\,(no\ ADC_{Vref}\ +)}$$

$$\text{10bit ADC conversion result} = 1024 \,|\, \frac{\text{The input voltage Vin of the ADC converted channel}}{(\text{ADC external reference source voltage})\,(ADC_{Vref}\ +)}$$

$$\text{12bit ADC conversion result} = 4096 \,|\, \frac{\text{The input voltage Vin of the ADC converted channel}}{(\text{MCU working voltage Vcc})\,(no\ ADC_{Vref}\ +)}$$

$$\text{12bit ADC conversion result} = 4096 \,|\, \frac{\text{The input voltage Vin of the ADC converted channel}}{(\text{ADC external reference source voltage})\,(ADC_{Vref}\ +)}$$

17.2.3 Reverse calculation formula for ADC input voltage

$$\text{input voltage Vin of the ADC converted channel} = \text{MCU working voltage Vcc} \,|\, \frac{\text{10bit ADC conversion r}}{1024\text{ɲo } ADC\text{ref} \quad +}$$

$$\text{input voltage Vin of the ADC converted channel} = \text{ADC external reference source voltage} \,|\, \frac{\text{10bit ADC}}{1024}$$

$$\text{input voltage Vin of the ADC converted channel} = \text{MCU working voltage Vcc} \,|\, \frac{\text{12bit ADC conversion r}}{4096\text{ɲo } ADC\text{ref}}$$

$$\text{input voltage Vin of the ADC converted channel} = \text{ADC external reference source voltage} \,|\, \frac{\text{12bit ADC}}{4096}$$

17.2.4 Reverse working voltage calculation formula

When you need to use the ADC input voltage and ADC conversion results to reverse the working voltage, if the target chip does not have an independent ADC_Vref+ pin, you can directly measure and use the following formula. If the target chip has an independent ADC_Vref+ pin, you must connect the ADC_Vref+ tube The pin is connected to the Vcc pin.

$$\text{MCU working voltage Vcc} = 1024 \,|\, \frac{\text{input voltage Vin of the ADC converted channel}}{\text{10bit ADC conversion result}}$$

$$\text{MCU working voltage Vcc} = 4096 \,|\, \frac{\text{input voltage Vin of the ADC converted channel}}{\text{12bit ADC conversion result}}$$

17.3 10 BIT ADC Static Characteristics

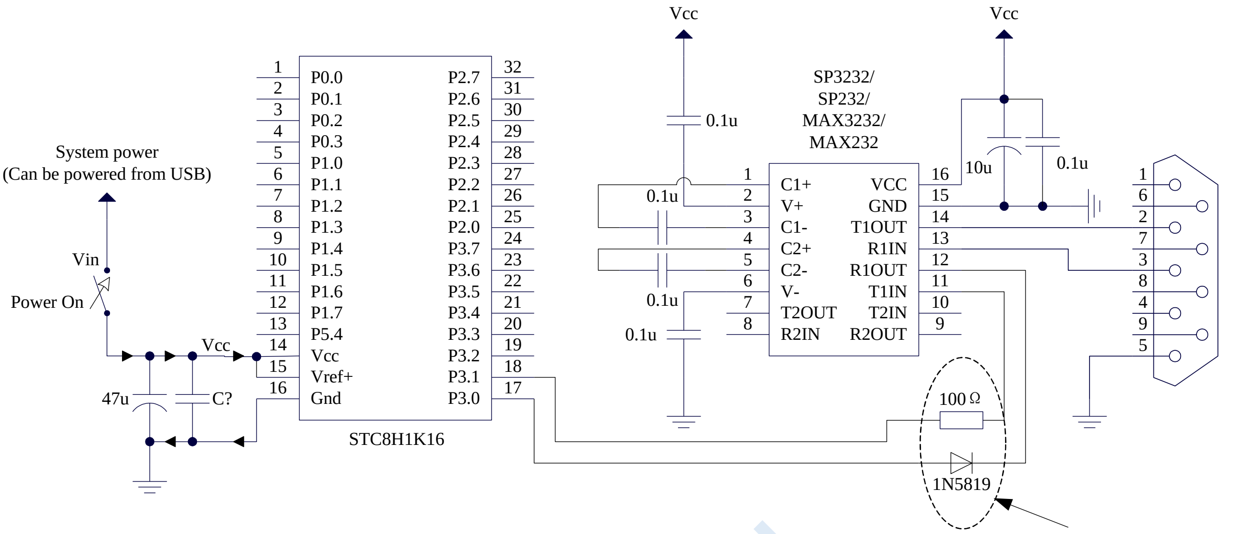
Symbol	Description	Minimum	Typical	Max	Unit
RES	Resolution	-	10	-	Bits
E_T	Overall error	-	1.3	3	LSB
E_O	Offset error	-	0.3	1	LSB
E_G	Gain error	-	0	1	LSB
E_D	Differential nonlinear error	-	0.7	1.5	LSB
E_I	Integral nonlinear error	-	1	2	LSB
R_{AIN}	Channel equivalent resistance	-	∞	-	Ohm
R_{ESD}	Antistatic resistance connected in series before the sample and hold capacitor	-	700	-	Ohm
C_{ADC}	Internal sample and hold capacitor	-	16.5	-	pF

17.4 12 BIT ADC Static Characteristics

Symbol	Description	Minimum	Typical	Max	Unit
RES	Resolution	-	12	-	Bits
E_T	Overall error	-	0.5	1	LSB
E_O	Offset error	-	-0.1	1	LSB
E_G	Gain error	-	0	1	LSB
E_D	Differential nonlinear error	-	0.7	1.5	LSB
E_I	Integral nonlinear error	-	1	2	LSB
R_{AIN}	Channel equivalent resistance	-	∞	-	Ohm
R_{ESD}	Antistatic resistance connected in series before the sample and hold capacitor	-	700	-	Ohm
C_{ADC}	Internal sample and hold capacitor	-	16.5	-	pF

17.5 ADC application reference circuit diagram

17.5.1 General precision ADC reference circuit diagram

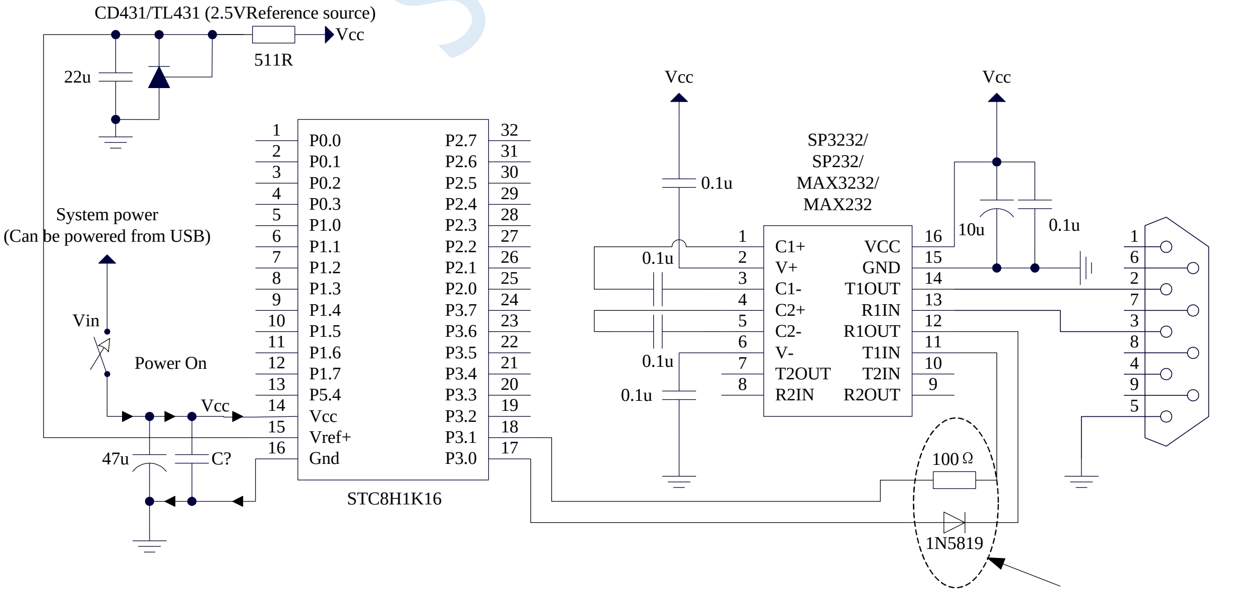


Here, resistors and diodes can be omitted

47u Tantalum capacitor (package 3528)
22u Monolithic capacitor (package 0603)
10u Monolithic capacitor (package 0603)
0.1u Monolithic capacitor (package 0603)

	System clock<=10MHz	System clock>10MHz
C?	104(0.1uF)	103(0.01uF)

17.5.2 High-precision ADC reference circuit diagram



Here, resistors and diodes can be omitted

47u Tantalum capacitor (package 3528)
22u Monolithic capacitor (package 0603)
10u Monolithic capacitor (package 0603)
0.1u Monolithic capacitor (package 0603)

	System clock<=10MHz	System clock>10MHz
C?	104(0.1uF)	103(0.01uF)

17.6 Example Routines

17.6.1 ADC Basic Operation (Polling Mode)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         ADC_CONTR   =           0xbc;
sfr         ADC_RES     =           0xbd;
sfr         ADC_RESL    =           0xbe;
sfr         ADCCFG      =           0xde;

sfr         P_SW2       =           0xba;
#define     ADCTIM      (*(unsigned cha volatile xdata *)0xfea8)

sfr         P0M1        =           0x93;
sfr         P0M0        =           0x94;
sfr         P1M1        =           0x91;
sfr         P1M0        =           0x92;
sfr         P2M1        =           0x95;
sfr         P2M0        =           0x96;
sfr         P3M1        =           0xb1;
sfr         P3M0        =           0xb2;
sfr         P4M1        =           0xb3;
sfr         P4M0        =           0xb4;
sfr         P5M1        =           0xc9;
sfr         P5M0        =           0xca;

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P1M0 = 0x00;                                    //Set P1.0 as ADC input
    P1M1 = 0x01;
    P_SW2 |= 0x80;
    ADCTIM = 0x3f;                                  // Set ADC internal timing
    P_SW2 &= 0x7f;
    ADCCFG = 0x0f;                                  //Set the ADC clock to the system clock/2/16
    ADC_CONTR = 0x80;                               //Enable ADC module
```

```
        while (1)
        {
                ADC_CONTR |= 0x40;                              //Start AD conversion
                _nop_();
                _nop_();
                while (!(ADC_CONTR & 0x20));                    //Query ADC completion flag
                ADC_CONTR &= ~0x20;                             //Clear completion flag
                P2 = ADC_RES;                                   //Read ADC results
        }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

ADC_CONTR   DATA        0BCH
ADC_RES     DATA        0BDH
ADC_RESL    DATA        0BEH
ADCCFG      DATA        0DEH

P_SW2       DATA        0BAH
ADCTIM      XDATA       0FEA8H

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN

            ORG         0100H
MAIN:
            MOV         SP, #5FH
            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
            MOV         P3M1, #00H
            MOV         P4M0, #00H
            MOV         P4M1, #00H
            MOV         P5M0, #00H
            MOV         P5M1, #00H

            MOV         P1M0,#00H                       ;Set P1.0 as ADC input
            MOV         P1M1,#01H
            MOV         P_SW2,#80H
            MOV         DPTR,#ADCTIM                    ; Set ADC internal timing
```

```
                MOV         A,#3FH
                MOVX        @DPTR,A
                MOV         P_SW2,#00H
                MOV         ADCCFG,#0FH                 ;Set the ADC clock to the system clock/2/16
                MOV         ADC_CONTR,#80H              ;Enable ADC module
LOOP:
                ORL         ADC_CONTR,#40H              ;Start AD conversion
                NOP
                NOP
                MOV         A,ADC_CONTR                 ;Query ADC completion flag
                JNB         ACC.5,$-2
                ANL         ADC_CONTR,#NOT 20H          ;Clear completion flag
                MOV         P2,ADC_RES                  ;Read ADC results

                SJMP        LOOP

                END
```

17.6.2 ADC Basic Operation (Interrupt Mode)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         ADC_CONTR   =       0xbc;
sfr         ADC_RES     =       0xbd;
sfr         ADC_RESL    =       0xbe;
sfr         ADCCFG      =       0xde;

sfr         P_SW2       =       0xba;
#define     ADCTIM              (*(unsigned char volatile xdata *)0xfea8)

sbit        EADC        =       IE^5;

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

void ADC_Isr() interrupt 5
{
    ADC_CONTR &= ~0x20;                             //Clear interrupt flag
    P2 = ADC_RES;                                   //Read ADC results
    ADC_CONTR |= 0x40;                              //Continue AD conversion
}

void main()
```

```
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P1M0 = 0x00;                                        //Set P1.0 as ADC input
        P1M1 = 0x01;
        P_SW2 |= 0x80;
        ADCTIM = 0x3f;                                      // Set ADC internal timing
        P_SW2 &= 0x7f;
        ADCCFG = 0x0f;                                      //Set the ADC clock to the system clock/2/16
        ADC_CONTR = 0x80;                                   //Enable ADC module
        EADC = 1;                                           //Enable ADC interrupt
        EA = 1;
        ADC_CONTR |= 0x40;                                  //Start AD conversion

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

ADC_CONTR   DATA        0BCH
ADC_RES     DATA        0BDH
ADC_RESL    DATA        0BEH
ADCCFG      DATA        0DEH

P_SW2       DATA        0BAH
ADCTIM      XDATA       0FEA8H

EADC        BIT         IE.5

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         002BH
            LJMP        ADCISR
```

```
                ORG             0100H
ADCISR:
                ANL             ADC_CONTR,#NOT 20H      ;Clear completion flag
                MOV             P2,ADC_RES              ;Read ADC results
                ORL             ADC_CONTR,#40H          ;Continue AD conversion
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             P1M0,#00H               ;Set P1.0 as ADC input
                MOV             P1M1,#01H
                MOV             P_SW2,#80H
                MOV             DPTR,#ADCTIM            ; Set ADC internal timing
                MOV             A,#3FH
                MOVX            @DPTR,A
                MOV             P_SW2,#00H
                MOV             ADCCFG,#0FH             ;Set the ADC clock to the system clock/2/16
                MOV             ADC_CONTR,#80H          ;Enable ADC module
                SETB            EADC                    ;Enable ADC interrupt
                SETB            EA
                ORL             ADC_CONTR,#40H          ;Start AD conversion

                SJMP            $

                END
```

17.6.3 Format ADC Conversion Result

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         ADC_CONTR   =       0xbc;
sfr         ADC_RES     =       0xbd;
sfr         ADC_RESL    =       0xbe;
sfr         ADCCFG      =       0xde;

sfr         P_SW2       =       0xba;
#define     ADCTIM              (*(unsigned char volatile xdata *)0xfea8)

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P1M0 = 0x00;                                        //Set P1.0 as ADC input
        P1M1 = 0x01;
        P_SW2 |= 0x80;
        ADCTIM = 0x3f;                                      //Set ADC internal timing
        P_SW2 &= 0x7f;
        ADCCFG = 0x0f;                                      //Set the ADC clock to the system clock/2/16/16
        ADC_CONTR = 0x80;                                   //Enable ADC module
        ADC_CONTR |= 0x40;                                  //Start AD conversion
        _nop_();
        _nop_();
        while (!(ADC_CONTR & 0x20));                        //Query ADC completion flag
        ADC_CONTR &= ~0x20;                                 //Clear completion flag

        ADCCFG = 0x00;                                      //Set result to align left
        ACC = ADC_RES;                                      //A stores the upper 8 bits of the ADC’ s 10-bit result
        B = ADC_RESL;                   //B[7: 6] stores the lower 2 bits of the 10-bit ADC result, B [5: 0] is 0

//      ADCCFG = 0x20;                                      //Set result to align right
//      ACC = ADC_RES;          // A [1: 0] stores the upper 2 bits of the 10-bit result of the ADC, and A [7: 2] is 0
//      B = ADC_RESL;                                       //B stores the lower 8 bits of the ADC’s 10-bit result

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

ADC_CONTR   DATA        0BCH
ADC_RES     DATA        0BDH
ADC_RESL    DATA        0BEH
ADCCFG      DATA        0DEH

P_SW2       DATA        0BAH
```

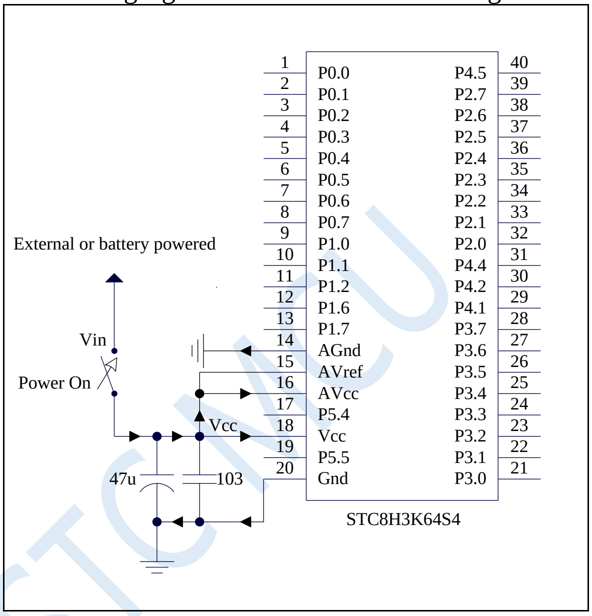
```
ADCTIM          XDATA           0FEA8H

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN

                ORG             0100H
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             P1M0,#00H                   ;Set P1.0 as ADC input
                MOV             P1M1,#01H
                MOV             P_SW2,#80H
                MOV             DPTR,#ADCTIM                ; Set ADC internal timing
                MOV             A,#3FH
                MOVX            @DPTR,A
                MOV             P_SW2,#00H
                MOV             ADCCFG,#0FH                 ;Set the ADC clock to the system clock/2/16
                MOV             ADC_CONTR,#80H              ;Enable ADC module

                ORL             ADC_CONTR,#40H              ;Start AD conversion
                NOP
                NOP
                MOV             A,ADC_CONTR                 ;Query ADC completion flag
                JNB             ACC.5,$-2
                ANL             ADC_CONTR,#NOT 20H          ;Clear completion flag

                MOV             ADCCFG,#00H                 ;Set result to align left
                MOV             A,ADC_RES                   ;A stores the upper 8 bits of the ADC’s 10-bit result
                MOV             B,ADC_RESL   ;B[7: 6] stores the lower 2 bits of the 10-bit ADC result, B [5: 0] is 0

;               MOV             ADCCFG,#20H                 ;Set result to align right
;               MOV             A,ADC_RES ;A [3: 0] stores the upper 2 bits of the 10-bit result of the ADC, and A [7: 2] is 0
;               MOV             B,ADC_RESL                  ;B stores the lower 8 bits of the ADC’s 10-bit result
```

```
                SJMP        $

                END
```

17.6.4 Detect External Voltage or Battery Voltage using ADC 15th Channel

The 15th channel of ADC in the STC8H family of microcontrollers is used to measure the internal reference voltage. The internal reference voltage is stable, about 1.19V, and does not change with the chip's working voltage. So you can measure the internal reference voltage and use it to deduce the external voltage or external battery voltage through the value of the ADC.

The following figure is a reference circuit diagram:



C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BRT             (65536 - FOSC / 115200 / 4)

sfr         AUXR        =   0x8e;

sfr         ADC_CONTR   =   0xbc;
sfr         ADC_RES     =   0xbd;
sfr         ADC_RESL    =   0xbe;
sfr         ADCCFG      =   0xde;

sfr         P_SW2       =   0xba;
#define     ADCTIM          (*(unsigned char volatile xdata *)0xfea8)

sfr         P0M1        =   0x93;
sfr         P0M0        =   0x94;
sfr         P1M1        =   0x91;
sfr         P1M0        =   0x92;
sfr         P2M1          =     0x95;
sfr         P2M0          =     0x96;
sfr         P3M1          =     0xb1;
sfr         P3M0          =     0xb2;
sfr         P4M1          =     0xb3;
sfr         P4M0          =     0xb4;
sfr         P5M1          =     0xc9;
sfr         P5M0          =   0xca;

int         *BGV;                                       //The internal reference voltage value is stored in idata
                                                        //The high byte is stored in idata's EFH address
                                                        //The low byte is stored in idata's F0H address
                                                        //Voltage unit is millivolt (mV)
bit         busy;

void UartIsr() interrupt 4
{
        if (TI)
        {
                TI = 0;
                busy = 0;
        }
        if (RI)
        {
                RI = 0;
        }
}

void UartInit()
{
        SCON = 0x50;
        TMOD = 0x00;
        TL1 = BRT;
        TH1 = BRT >> 8;
        TR1 = 1;
        AUXR = 0x40;
        busy = 0;
}

void UartSend(char dat)
{
        while (busy);
        busy = 1;
        SBUF = dat;
}

void ADCInit()
{
        P_SW2 |= 0x80;
        ADCTIM = 0x3f;                                  // Set ADC internal timing
        P_SW2 &= 0x7f;

        ADCCFG = 0x2f;                                  //Set the ADC clock to the system clock/2/16
        ADC_CONTR = 0x8f;                               //Enable ADC module, and select channel 15
}

int     ADCRead()
{
        int res;

        ADC_CONTR |= 0x40;                              //Start AD conversion
        _nop_();
        _nop_();
        while (!(ADC_CONTR & 0x20));                          //Query ADC completion flag
        ADC_CONTR &= ~0x20;                                   //Clear completion flag
        res = (ADC_RES << 8) | ADC_RESL;                      //Read ADC results

        return res;
}

void main()
{
        int res;
        int vcc;
        int i;

        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        BGV = (int idata *)0xef;
        ADCInit();                                            //ADC initialization
        UartInit();                                           //UART initialization

        ES = 1;
        EA = 1;

//      ADCRead();
//      ADCRead();                                            //Discard the first two data

        res = 0;
        for (i=0; i<8; i++)
        {
                res += ADCRead();                             //Read data 8 times
        }
        res >>= 3;                                            //take the average

        vcc = (int)(4096L * *BGV / res);                      // (12-bit ADC algorithm)Calculate VREF pin voltage, i.e.
battery voltage
//      vcc = (int)(1024L * *BGV / res);                      // (10-bit ADC algorithm)Calculate VREF pin voltage, i.e.
battery voltage
                                                              //Note that this voltage is in millivolts (mV)
        UartSend(vcc >> 8);                                   //Output voltage value to UART
        UartSend(vcc);

        while (1);
}
```

The method above uses the 15th channel of the ADC to invert the external battery voltage. In the ADC measurement range, the external measurement voltage of the ADC is proportional to the measurement value of the ADC, so the 15th channel of the ADC can also be used to reverse the input voltage of the external channel. Assuming that the current internal reference signal source voltage is BGV, the ADC measurement value of the internal reference signal source is res_{bg}, and the ADC measurement value of the external channel input voltage is res_x, then the external channel input voltage $Vx=BGV / res_{bg} * res_x$;

17.6.5 Using ADC as Capacitive Sensing Touch Keys

Key is one of the most commonly used parts in the circuit, and it is an important input method for the human-machine interface. We are most familiar with mechanical keys. The mechanical keys have a disadvantage of limited contact life especially for the cheap keys. And they are easy to appear poor contact and failure. Non-contact keys have no mechanical contacts, long life and easy to use.

There are various solutions for non-contact keys. Capacitive-sensing keys are low-cost solutions. Specialized ICs were used to implement capacitive-sensing keys many years ago. With the enhancement of MCU functions and the practical experience of users, MCUs were used to implement capacitive-sensing keys directly. The technology of capacitive sensing keys is mature. The most typical and reliable one is the solution using ADC.

The solution of using STC series MCUs with ADC is described in detail in this document. Any MCU with ADC function can be used to implement the scheme. The first three diagrams below are the most commonly used methods. The principles are the same. The second diagram is used.

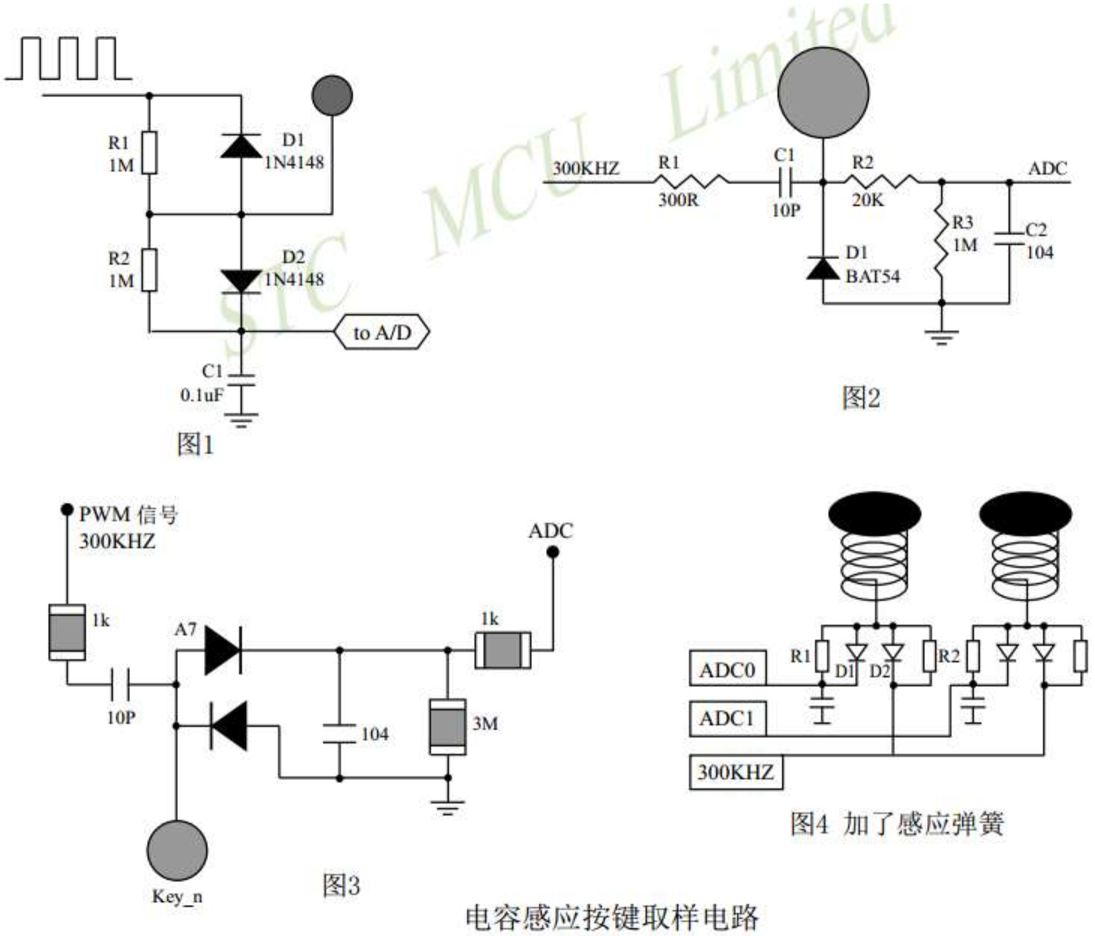


图1

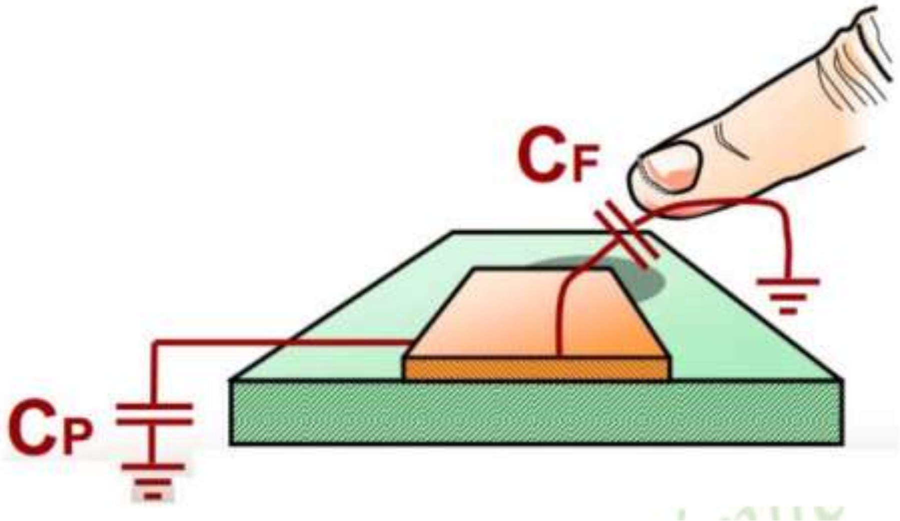
图2

图3

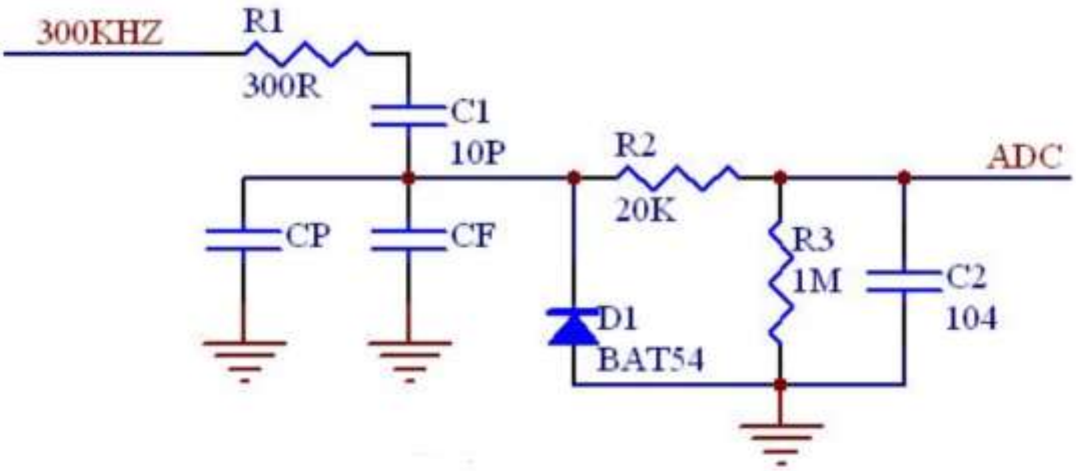
图4 加了感应弹簧

电容感应按键取样电路

In general applications, the induction spring shown in Figure 4 is used to increase the area pressed by a finger. The induction spring is equivalent to a metal plate to the ground. There is a capacitor CP to the ground. After pressing the finger, a capacitor CF is connected in parallel to the ground, as shown in the figure below.



The following is the description of the circuit diagram. CP is the distributed capacitance of metal plate and ground, CF is the finger capacitance, they are connected in parallel and connected with C1 to divide the input 300KHZ square wave. After being rectified by D1 and filtered by R2 and C2, the wave is sent to ADC. After pressing the finger, the voltage sent to the ADC decreases, and the program can detect the key action.



C language code

```
//Operating frequency for testing is 24MHz

#include "reg51.h"
#include "intrins.h"

#define     MAIN_Fosc      24000000UL                        //Define the main clock
#define     Timer0_Reload  (65536UL -(MAIN_Fosc / 600000))   //Timer 0 reload value corresponds to 300KHz

typedef     unsigned char  u8;
typedef     unsigned int   u16;
typedef     unsigned long  u32;

sfr         P0M1           =    0x93;
sfr         P0M0           =    0x94;
sfr         P1M1           =    0x91;
sfr         P1M0           =    0x92;
sfr         P2M1           =    0x95;
sfr         P2M0           =    0x96;
sfr         P3M1           =    0xb1;
sfr         P3M0           =    0xb2;
sfr         P4M1           =    0xb3;
sfr     P4M0        =   0xb4;
sfr     P5M1        =   0xc9;
sfr     P5M0        =   0xca;

sfr     ADC_CONTR   =   0xBC;                   // microcontrollers with ADC
sfr     ADC_RES     =   0xBD;                   / microcontrollers with ADC
sfr     ADC_RESL    =   0xBE;                   // microcontrollers with ADC
sfr     AUXR        =   0x8E;
sfr     AUXR2       =   0x8F;

#define CHANNEL     8                           //ADC channel numbers
#define ADC_90T     (3<<5)                      //ADC conversion time 90T
#define ADC_180T    (2<<5)                      //ADC conversion time 180T
#define ADC_360T    (1<<5)                      //ADC conversion time 360T
#define ADC_540T    0                           //ADC conversion time 540T
#define ADC_FLAG    (1<<4)                      //Cleared by software
#define ADC_START   (1<<3)                      //Cleared automatically

sbit    P_LED7      =   P2^7;
sbit    P_LED6      =   P2^6;
sbit    P_LED5      =   P2^5;
sbit    P_LED4      =   P2^4;
sbit    P_LED3      =   P2^3;
sbit    P_LED2      =   P2^2;
sbit    P_LED1      =   P2^1;
sbit    P_LED0      =   P2^0;

u16 idata adc[TOUCH_CHANNEL];                   //Current ADC value
u16 idata adc_prev[TOUCH_CHANNEL];              //Previous ADC value
u16 idata TouchZero[TOUCH_CHANNEL];             //ADC value of 0-point
u8 idata TouchZeroCnt[TOUCH_CHANNEL];           //Automatic tracking count for 0-point
u8 cnt_250ms;

void delay_ms(u8 ms);
void ADC_init(void);
u16 Get_ADC10bitResult(u8 channel);
void AutoZero(void);
u8 check_adc(u8 index);
void ShowLED(void);

void main(void)
{
    u8 i;

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    delay_ms(50);
    ET0 = 0;                                    //Initialize Timer0 to output a 300KHz clock
```

```
        TR0 = 0;
        AUXR |= 0x80;                                       //Timer0 set as 1T mode
        AUXR2 |= 0x01;                                      //Enable clock output
        TMOD = 0;                                           //Timer0 set as Timer, 16 bits Auto Reload.
        TH0 = (u8)(Timer0_Reload >> 8);
        TL0 = (u8)Timer0_Reload;
        TR0 = 1;
        ADC_init();                                         //ADC initialization
        delay_ms(50);                                       //Delay 50ms
        for (i=0; i<TOUCH_CHANNEL; i++)  // Initialize the 0-point, the previous value and the 0-point auto-tracking count
        {
                adc_prev[i] = 1023;
                TouchZero[i] = 1023;
                TouchZeroCnt[i] = 0;
        }
        cnt_250ms = 0;
        while (1)
        {
                delay_ms(50);                               //Process key once every 50ms
                ShowLED();
                if (++cnt_250ms >= 5)
                {
                        cnt_250ms = 0;
                        AutoZero();                         //Process 0-point auto-tracking every 250ms
                }
        }
}

void delay_ms(u8 ms)
{
        unsigned int i;

        do
        {
                i = MAIN_Fosc / 13000;
                while(--i) ;
        } while(--ms);
}

void ADC_init(void)
{
        P1M0 = 0x00;                                        //8 channels ADC
        P1M1 = 0xff;
        ADC_CONTR = 0x80;                                   //Enable ADC
}

u16 Get_ADC10bitResult(u8 channel)
{
        ADC_RES = 0;
        ADC_RESL = 0;
        ADC_CONTR = 0x80 | ADC_90T | ADC_START | channel;   //Trigger ADC
        _nop_();
        _nop_();
        _nop_();
        _nop_();
        while((ADC_CONTR & ADC_FLAG) == 0) ;                //Wait for ADC conversion complement
        ADC_CONTR = 0x80;                                   //Clear flag
        return(((u16)ADC_RES << 2) | ((u16)ADC_RESL & 3));  //Return ADC result
}
```

```
void AutoZero(void)                                             //Call once every 250ms
                    // This is detected using the sum of the absolute values of the differences between two adjacent
samples.
{
    u8 i;
    u16 j,k;

    for(i=0; i<TOUCH_CHANNEL; i++)                              //Process 8 channels
    {
        j = adc[i];
        k = j - adc_prev[i];                                    // Subtract previous reading
        F0 = 0;                                                 //Pressed
        if(k & 0x8000) F0 = 1, k = 0 - k;                       //Release, get the difference between two samples
        if(k >= 20)                                             // Big change
        {
            TouchZeroCnt[i] = 0;                                // If the change is large, clear the counter
            if(F0) TouchZero[i] = j;    // If it is released, and the change is relatively large, then directly replace
        }
        else                            // If the change is relatively small, then creep, track 0-point automatically
        {
            if(++TouchZeroCnt[i] >= 20)                         // Continuously detect small changes 20 times/4 = 5
seconds.
            {
                TouchZeroCnt[i] = 0;
                TouchZero[i] = adc_prev[i];                     // Use slowly changing values as 0 points           }
        }
        adc_prev[i] = j;                                        // Save this time's sample value
    }
}

u8 check_adc(u8 index)                                          // Get touch information function, called every 50ms
                                                                // Judge key is pressed or released with hysteresis control
{
    u16 delta;

    adc[index] = 1023 - Get_ADC10bitResult(index);      // Get ADC value, convert to press the key, ADC value increases
    if(adc[index] < TouchZero[index]) return 0;                 // A value smaller than 0-point is considered a key release
    delta = adc[index] - TouchZero[index];
    if(delta >= 40) return 1;                                   //Key pressed
    if(delta <= 20) return 0;                                   //Key released
    return 2;                                                   // Keep the original state
}

void ShowLED(void)
{
    u8 i;

    i = check_adc(0);
    if(i == 0) P_LED0 = 1;                                      //Light off
    if(i == 1) P_LED0 = 0;                                      //Light on
    i = check_adc(1);
    if(i == 0) P_LED1 = 1;                                      //Light off
    if(i == 1) P_LED1 = 0;                                      //Light on
    i = check_adc(2);
    if(i == 0) P_LED2 = 1;                                      //Light off
    if(i == 1) P_LED2 = 0;                                      //Light on
    i = check_adc(3);
    if(i == 0) P_LED3 = 1;                                      //Light off
```

```
        if(i == 1) P_LED3 = 0;                                      //Light on
        i = check_adc(4);
        if(i == 0) P_LED4 = 1;                                      //Light off
        if(i == 1) P_LED4 = 0;                                      //Light on
        i = check_adc(5);
        if(i == 0) P_LED5 = 1;                                      //Light off
        if(i == 1) P_LED5 = 0;                                      //Light on
        i = check_adc(6);
        if(i == 0) P_LED6 = 1;                                      //Light off
        if(i == 1) P_LED6 = 0;                                      //Light on
        i = check_adc(7);
        if(i == 0) P_LED7 = 1;                                      //Light off
        if(i == 1) P_LED7 = 0;                                      //Light on
}
```

Assembly code

```
;Operating frequency for testing is 24MHz

Fosc_KHZ        EQU         24000                       ;Define the main clock KHZ
Reload          EQU         (65536 - Fosc_KHZ/600)      ;Timer 0 reload value, corresponding to 300KHz

ADC_CONTR       DATA        0xBC                        ; microcontrollers with ADC
ADC_RES         DATA        0xBD                        ; microcontrollers with ADC
ADC_RESL        DATA        0xBE                        ; microcontrollers with ADC
AUXR            DATA        0x8E
AUXR2           DATA        0x8F

P0M1            DATA        093H
P0M0            DATA        094H
P1M1            DATA        091H
P1M0            DATA        092H
P2M1            DATA        095H
P2M0            DATA        096H
P3M1            DATA        0B1H
P3M0            DATA        0B2H
P4M1            DATA        0B3H
P4M0            DATA        0B4H
P5M1            DATA        0C9H
P5M0            DATA        0CAH

CHANNEL         EQU         8                           ;ADC channel numbers
ADC_90T         EQU         (3 SHL 5)                   ;ADC conversion time 90T
ADC_180T        EQU         (2 SHL 5)                   ;ADC conversion time 180T
ADC_360T        EQU         (1 SHL 5)                   ;ADC conversion time 360T
ADC_540T        EQU         0                           ;ADC conversion time 540T
ADC_FLAG        EQU         (1 SHL 4)                   ;Cleared by software
ADC_START       EQU         (1 SHL 3)                   ;Cleared automatically

P_LED7          BIT         P2.7;
P_LED6          BIT         P2.6;
P_LED5          BIT         P2.5;
P_LED4          BIT         P2.4;
P_LED3          BIT         P2.3;
P_LED2          BIT         P2.2;
P_LED1          BIT         P2.1;
P_LED0          BIT         P2.0;
adc             EQU         30H     ; Current ADC value in 30H ~ 3FH, two bytes constitute one value
```

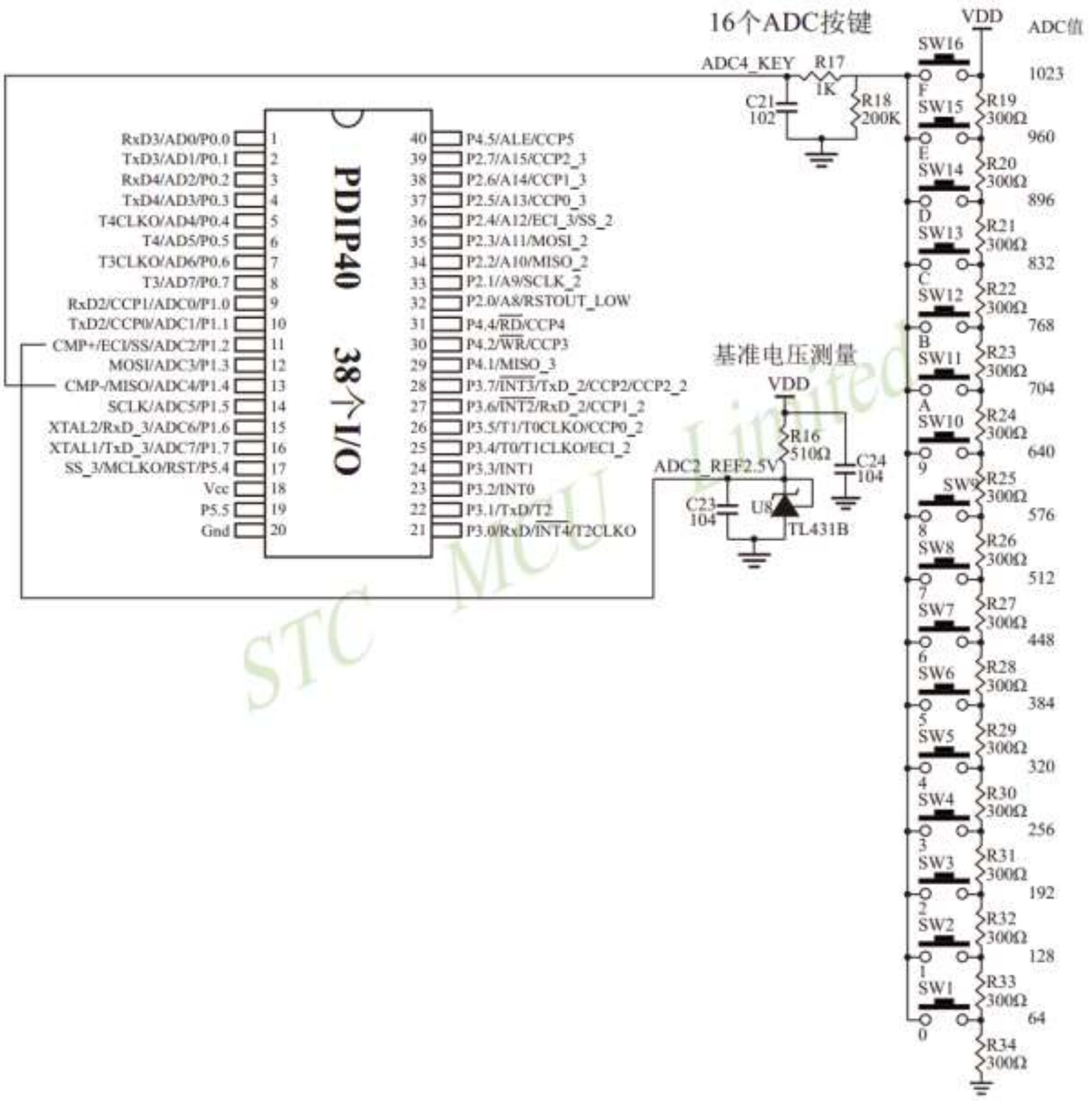
```
adc_prev        EQU         40H     ; Previous ADC value in 40H ~ 4FH, two bytes constitute a value
TouchZero       EQU         50H                         ; ADC 0 value in 50H~5FH, two bytes constitute a value
TouchZeroCnt    EQU         60H                         ; 0-point automatic tracking count in 60H~67H
cnt_250ms       DATA        68H

                ORG         0000H
                LJMP        MAIN

                ORG         0100H
MAIN:
                MOV         SP,#0D0H
                MOV         P0M0,#00H
                MOV         P0M1,#00H
                MOV         P1M0,#00H
                MOV         P1M1,#00H
                MOV         P2M0,#00H
                MOV         P2M1,#00H
                MOV         P3M0,#00H
                MOV         P3M1,#00H
                MOV         P4M0,#00H
                MOV         P4M1,#00H
                MOV         P5M0,#00H
                MOV         P5M1,#00H

                MOV         R7,#50
                LCALL       F_delay_ms
                CLR         ET0                         ;Initialize Timer0 to output a 300KHz clock
                CLR         TR0
                ORL         AUXR,#080H                  ;Timer0 set as 1T mode
                ORL         AUXR2,#01H                  ;Enable clock output
                MOV         TMOD,#0                     ;Timer0 set as Timer,16 bits Auto Reload.
                MOV         TH0,#HIGH Reload
                MOV         TL0,#LOW Reload
                SETB        TR0
                LCALL       F_ADC_init
                MOV         R7,#50
                LCALL       F_delay_ms
                MOV         R0,#adc_prev                ;Initialize the previous ADC value
L_Init_Loop1:
                MOV         @R0,#03H
                INC         R0
                MOV         @R0,#0FFH
                INC         R0
                MOV         A,R0
                CJNE        A,#(adc_prev + CHANNEL * 2),L_Init_Loop1
                MOV         R0,#TouchZero               ;Initialize the ADC 0-point value
L_Init_Loop2:
                MOV         @R0,#03H
                INC         R0
                MOV         @R0,#0FFH
                INC         R0
                MOV         A,R0
                CJNE        A,#(TouchZero+CHANNEL * 2),L_Init_Loop2
                MOV         R0,#TouchZeroCnt            ;Initialize the automatic tracking count value
L_Init_Loop3:
                MOV         @R0,#0
                INC         R0
                MOV         A,R0
                CJNE        A,#(TouchZeroCnt + CHANNEL),L_Init_Loop3
                MOV             cnt_250ms,#5
L_MainLoop:
                MOV             R7,#50                          ;Delay 50ms
                LCALL           F_delay_ms
                LCALL           F_ShowLED                       ;Handle key value once
                DJNZ            cnt_250ms,L_MainLoop
                MOV             cnt_250ms,#5        ;Processing once 0-point automatic tracking value every 250ms
                LCALL           F_AutoZero                      ; Zero tracking
                SJMP            L_MainLoop

F_ADC_init:
                MOV             P1M0,#00H                       ;8 channels ADC
                MOV             P1M1,#0FFH
                MOV             ADC_CONTR,#080H                 ;Enable ADC
                RET

F_Get_ADC10bitResult:
                MOV             ADC_RES,#0
                MOV             ADC_RESL,#0
                MOV             A,R7
                ORL             A,#0E8H                         ;Trigger ADC
                MOV             ADC_CONTR,A
                NOP
                NOP
                NOP
                NOP
L_10bitADC_Loop1:
                MOV             A,ADC_CONTR
                JNB             ACC.4,L_10bitADC_Loop1          ;Wait for the ADC conversion complement
                MOV             ADC_CONTR,#080H                 ;Clear flag
                MOV             A,ADC_RES
                MOV             B,#04H
                MUL             AB
                MOV             R7,A
                MOV             R6,B
                MOV             A,ADC_RESL
                ANL             A,#03H
                ORL             A,R7
                MOV             R7,A
                RET

F_AutoZero:                                                     ; Call once every 250ms
                ; This is detected using the sum of the absolute values of the differences between two adjacent samples.
                CLR             A
                MOV             R5,A
L_AutoZero_Loop:
                MOV             A,R5
                ADD             A,ACC
                ADD             A,#LOW (adc)
                MOV             R0,A
                MOV             A,@R0
                MOV             R6,A
                INC             R0
                MOV             A,@R0
                MOV             R7,A
                MOV             A,R5
                ADD             A,ACC
                ADD             A,#LOW (adc_prev+01H)
                MOV             R0,A
                CLR         C
                MOV         A,R7
                SUBB        A,@R0
                MOV         R3,A
                MOV         A,R6
                DEC         R0
                SUBB        A,@R0
                MOV         R2,A
                CLR         F0 ;按下
                JNB         ACC.7,L_AutoZero_1
                SETB        F0
                CLR         C
                CLR         A
                SUBB        A,R3
                MOV         R3,A
                MOV         A,R3
                CLR         A
                SUBB        A,R2
                MOV         R2,A
L_AutoZero_1:
                CLR         C                       ;Calculate [R2 R3] -    #20,if(k >= 20)
                MOV         A,R3
                SUBB        A,#20
                MOV         A,R2
                SUBB        A,#00H
                JC          L_AutoZero_2            ;[R2 R3] ,20, Jump
                MOV         A,#LOW (TouchZeroCnt)  ; If the change is large, clear the counter TouchZeroCnt[i] =
0;
                ADD         A,R5
                MOV         R0,A
                MOV         @R0,#0
                JNB         F0,L_AutoZero_3
                MOV         A,R5
                ADD         A,ACC
                ADD         A,#LOW (TouchZero)
                MOV         R0,A
                MOV         @R0,6
                INC         R0
                MOV         @R0,7
                SJMP        L_AutoZero_3
L_AutoZero_2:                     ; If the change is relatively small, then creep, track 0-point automatically
                                            ; Continuously detect small changes 20 times/4 = 5 seconds.
                MOV         A,#LOW (TouchZeroCnt)
                ADD         A,R5
                MOV         R0,A
                INC         @R0
                MOV         A,@R0
                CLR         C
                SUBB        A,#20
                JC          L_AutoZero_3            ;if(TouchZeroCnt[i] < 20), jump
                MOV         @R0,#0                  ;TouchZeroCnt[i]= 0;
                MOV         A,R5                    ; Use slowly changing values as 0 points
                ADD         A,ACC
                ADD         A,#LOW (adc_prev)
                MOV         R0,A
                MOV         A,@R0
                MOV         R2,A
                INC         R0
                MOV         A,@R0
```

```
                    MOV         R3,A
                    MOV         A,R5
                    ADD         A,ACC
                    ADD         A,#LOW (TouchZero)
                    MOV         R0,A
                    MOV         @R0,2
                    INC         R0
                    MOV         @R0,3
L_AutoZero_3:                                           ; Save the sampled value adc_prev[i] = j;
                    MOV         A,R5
                    ADD         A,ACC
                    ADD         A,#LOW (adc_prev)
                    MOV         R0,A
                    MOV         @R0,6
                    INC         R0
                    MOV         @R0,7
                    INC         R5
                    MOV         A,R5
                    XRL         A,#08H
                    JZ          $ + 5H
                    LJMP        L_AutoZero_Loop
                    RET

F_check_adc:                                            ; Judge key is pressed or released, with hysteresis control
                    MOV R4,7
                    LCALL       F_Get_ADC10bitResult    ; The ADC value returned is [R6 R7]
                    CLR         C
                    MOV         A,#0FFH
                    SUBB        A,R7
                    MOV         R7,A
                    MOV         A,#03H
                    SUBB        A,R6
                    MOV         R6,A
                    MOV         A,R4                    ;Save adc[index]
                    ADD         A,ACC
                    ADD         A,#LOW (adc)
                    MOV         R0,A
                    MOV         @R0,6
                    INC         R0
                    MOV         @R0,7
                    MOV         A,R4
                    ADD         A,ACC
                    ADD         A,#LOW (TouchZero+01H)
                    MOV         R1,A
                    MOV         A,R4
                    ADD         A,ACC
                    ADD         A,#LOW (adc)
                    MOV         R0,A
                    MOV         A,@R0
                    MOV         R6,A
                    INC         R0
                    MOV         A,@R0
                    CLR         C
                    SUBB        A,@R1                   ;Calculate adc[index] - TouchZero[index]
                    MOV         A,R6
                    DEC         R1
                    SUBB        A,@R1
                    JNC         L_check_adc_1
                    MOV         R7,#00H
```

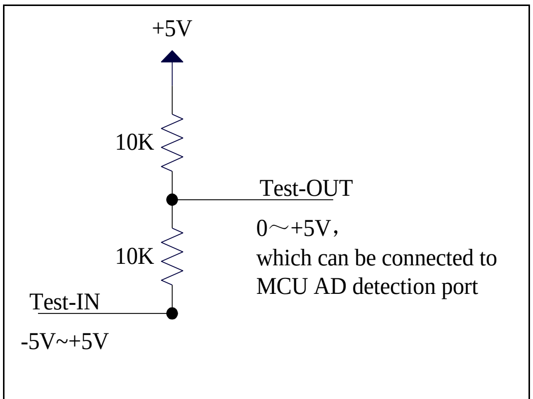
```
                RET
L_check_adc_1:
                MOV             A,R4
                ADD             A,ACC
                ADD             A,#LOW (TouchZero+01H)
                MOV             R1,A
                MOV             A,R4
                ADD             A,ACC
                ADD             A,#LOW (adc+01H)
                MOV             R0,A
                CLR             C
                MOV             A,@R0
                SUBB            A,@R1
                MOV             R7,A
                DEC             R0
                MOV             A,@R0
                DEC             R1
                SUBB            A,@R1
                MOV             R6,A
                CLR             C
                MOV             A,R7
                SUBB            A,#40
                MOV             A,R6
                SUBB            A,#00H
                JC              L_check_adc_2                   ;if(delta < 40), jump
                MOV             R7,#1                           ;if(delta >= 40) return 1; //Key pressed, return 1
                RET
L_check_adc_2:
                SETB            C
                MOV             A,R7
                SUBB            A,#20
                MOV             A,R6
                SUBB            A,#00H
                JNC             L_check_adc_3
                MOV             R7,#0
                RET
L_check_adc_3:
                MOV             R7,#2
                RET

F_ShowLED:
                MOV             R7,#0
                LCALL           F_check_adc
                MOV             A,R7
                ANL             A,#0FEH
                JNZ             L_QuitCheck0
                MOV             A,R7
                MOV             C,ACC.0
                CPL             C
                MOV             P_LED0,C
L_QuitCheck0:
                MOV             R7,#1
                LCALL           F_check_adc
                MOV             A,R7
                ANL             A,#0FEH
                JNZ             L_QuitCheck1
                MOV             A,R7
                MOV             C,ACC.0
                CPL             C
                MOV         P_LED1,C
L_QuitCheck1:
                MOV         R7,#2
                LCALL       F_check_adc
                MOV         A,R7
                ANL         A,#0FEH
                JNZ         L_QuitCheck2
                MOV         A,R7
                MOV         C,ACC.0
                CPL         C
                MOV         P_LED2,C
L_QuitCheck2:
                MOV         R7,#3
                LCALL       F_check_adc
                MOV         A,R7
                ANL         A,#0FEH
                JNZ         L_QuitCheck3
                MOV         A,R7
                MOV         C,ACC.0
                CPL         C
                MOV         P_LED3,C
L_QuitCheck3:
                MOV         R7,#4
                LCALL       F_check_adc
                MOV         A,R7
                ANL         A,#0FEH
                JNZ         L_QuitCheck4
                MOV         A,R7
                MOV         C,ACC.0
                CPL         C
                MOV         P_LED4,C
L_QuitCheck4:
                MOV         R7,#5
                LCALL       F_check_adc
                MOV         A,R7
                ANL         A,#0FEH
                JNZ         L_QuitCheck5
                MOV         A,R7
                MOV         C,ACC.0
                CPL         C
                MOV         P_LED5,C
L_QuitCheck5:
                MOV         R7,#6
                LCALL       F_check_adc
                MOV         A,R7
                ANL         A,#0FEH
                JNZ         L_QuitCheck6
                MOV         A,R7
                MOV         C,ACC.0
                CPL         C
                MOV         P_LED6,C
L_QuitCheck6:
                MOV         R7,#7
                LCALL       F_check_adc
                MOV         A,R7
                ANL         A,#0FEH
                JNZ         L_QuitCheck7
                MOV         A,R7
                MOV         C,ACC.0
                CPL             C
                MOV             P_LED7,C
L_QuitCheck7:
                RET

F_delay_ms:
                PUSH            3
                PUSH            4
L_delay_ms_1:
                MOV             R3,#HIGH (Fosc_KHZ / 13)
                MOV             R4,#LOW (Fosc_KHZ / 13)
L_delay_ms_2:
                MOV             A,R4
                DEC             R4
                JNZ             L_delay_ms_3
                DEC             R3
L_delay_ms_3:
                DEC             A
                ORL             A,R3
                JNZ             L_delay_ms_2
                DJNZ            R7,L_delay_ms_1
                POP             4
                POP             3
                RET

                END
```

17.6.6 Key-scan Application Circuit Diagram using ADC

Method for reading the ADC key: Read the ADC value every 10ms or so and save the last 3 readings. If the change is relatively small, judge the key. When the key is judged be valid, a certain deviation is allowed, such as a deviation of ±16 words.

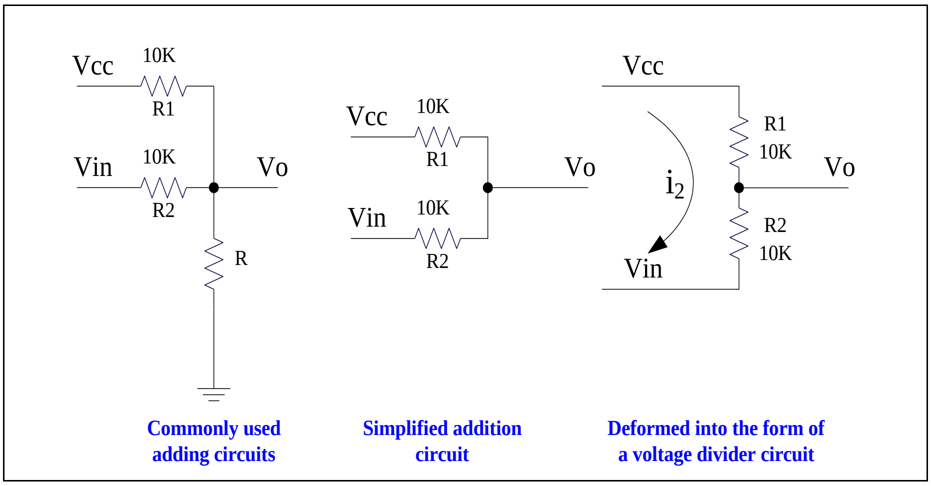


17.6.7 Reference circuit diagram for detecting negative voltage



Negative Voltage Conversion Circuit

17.6.8 The application of common addition circuit in ADC


Vcc
10K
R1
Vin
10K
R2
Vo
R
Commonly used adding circuits
Vcc
10K
R1
Vin
10K
R2
Vo
Simplified addition circuit
Vcc
R1
10K
Vo
i_2
R2
10K
Vin
Deformed into the form of a voltage divider circuit

Refer to the voltage divider circuit to get formula 1
Formula 1: Vo = Vin + i2 * R2
Formula 2: i2 = (Vcc-Vin) / (R1 + R2) {Condition: the current flowing to Vo ≈0}

Substituting R1=R2 into formula 2 gives formula 3
Formula 3: i2 = (Vcc-Vin) / 2R2

Substituting formula 3 into formula 1 gives formula 4
Formula 4: Vo = (Vcc + Vin) / 2
According to formula 4, the above circuit can be regarded as an addition circuit.

In the analog-to-digital conversion measurement of the microcontroller, the measured voltage is required to be greater than 0 and less than VCC. If the measured voltage is less than 0V, an addition circuit can be used to increase the measured voltage to above 0V. At this time, there are certain requirements for the variation range of the measured voltage:

Substituting the above conditions into formula 4, the following formula 2 can be obtained
(Vcc + Vin) / 2> 0 means Vin> -Vcc
(Vcc + Vin) / 2 <Vcc means Vin <Vcc
The above 2 formulas can be combined: -Vcc <Vin <Vcc

18 Sysnchronous Serial Peripheral Interface (SPI)

Product line	SPI	High speed SPI (the SPI clock is the system clock/2 if SPR=11B)
STC8H1K08 family	●	
STC8H1K28 family	●	
STC8H3K64S4 family A version	●	
STC8H3K64S4 family A version	●	
STC8H3K64S2 family B version		●
STC8H3K64S4 family B version		●
STC8H8K64U family A version	●	
STC8H8K64U family B version		●
STC8H2K64T family	●	
STC8H4K64TLR family		●
STC8H4K64TLCD family		●
STC8H4K64LCD family		●

A high-speed serial communication interface, SPI, is integrated in STC8H series of microcontrollers. SPI is a full-duplex high-speed synchronous communication bus. SPI interface integrated in the STC8H series of microcontrollers offers two operation modes: master mode and slave mode.

18.1 Registers Related to SPI

Symbol	Description	Address	Bit Address and Symbol								Reset Value
			B7	B6	B5	B4	B3	B2	B1	B0	
SPSTAT	SPI Status register	CDH	SPIF	WCOL	-	-	-	-	-	-	00xx,xxxx
SPCTL	SPI Control Register	CEH	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR[1:0]		0000,0100
SPDAT	SPI Data Register	CFH									0000,0000

18.1.1 SPI Status register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
SPSTAT	CDH	SPIF	WCOL	-	-	-	-	-	-

SPIF: SPI transfer completion flag.

When SPI completes sending / receiving 1 byte of data, the hardware will automatically set this bit and request interrupt to CPU. If the SSIG bit is set to 0, this flag will also be automatically set by hardware to indicate a mode change of device when the master / slave mode of the device changes due to changes in the SS pin level.

Note: This bit must be cleared using software writing 1 to it.

WCOL: SPI write collision flag bit.

This bit is set by hardware when the SPI is writing to the SPDAT register during data transfer.

Note: This bit must be cleared using software by writing 1 to it.

18.1.2 SPI Control Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
SPCTL	CEH	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR[1:0]	

SSIG: Control bit of whether SS pin is ignored or not.

0: the SS pin decides whether the device is a master or slave.

1: the function of SS pin is ignored. MSTR decides whether the device is a master or slave.

SPEN: SPI enable bit.

0: the SPI is disabled.

1: the SPI is enabled.

DORD: Set the transmitted or received SPI data order.

0: The MSB of the data is transmitted firstly.

1: The LSB of the data is transmitted firstly.

MSTR: Master/Slave mode select bit.

To set the mastert mode:

If SSIG = 0, the SS pin must be high and set MSTR to 1.

If SSIG = 1, it only needs to set MSTR to 1 (ignoring the SS pin level).

To set the slave mode:

If SSIG = 0, the SS pin must be low (regardless of the MSTR bit).

If SSIG = 1, it only needs to set MSTR to 0 (ignoring the SS pin level).

CPOL: SPI clock polarity select bit.

0：SCLK is low when idle. The leading edge of SCLK is the rising edge and the trailing edge is the falling edge.

1：SCLK is high when idle. The leading edge of SCLK is the falling edge and the trailing edge is the rising edge.

CPHA: SPI clock phase select bit.

0: The first bit of datum is driven when SS pin is low. The datum changes on the trailing edge of SCLK and is sampled on the leading edge of SCLK. (SSIG must be 0.)

1: The datum is driven on the leading edge of SCLK, and is sampled on the trailing edge.

SPR[1:0]: SPI clock frequency select bits

SPR[1:0]	SCLK frequency	SCLK frequency of high speed SPI
00	SYSclk/4	SYSclk/4
01	SYSclk/8	SYSclk/8
10	SYSclk/16	SYSclk/16
11	SYSclk/32	SYSclk/2

18.1.3 SPI Data Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
SPDAT	CFH								

The SPDAT holds the data to be transmitted or the data received.

18.2 SPI Communication Modes

There are three SPI communication modes: single master and single slave mode, dual devices configuration mode (any one of them can be a master or slave), single master and multiple slaves mode.

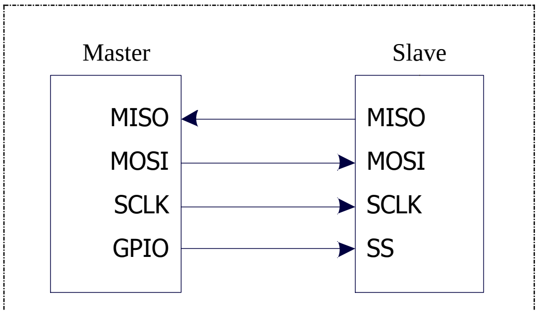
18.2.1 Single Master and Single Slave Mode

Two devices are connected, one of which is fixed as a master and the other as a slave.

Master settings: SSIG set to 1, MSTR set to 1, fixed to be master mode. The master can use any port to connect the slave SS pin, pull down the slave SS pin to enable the slave.

Slave settings: SSIG is set to 0, SS pin as the chip select signal of the slave.

Single master single slave connection configuration diagram is shown as follows:



Single master and single slave configuration

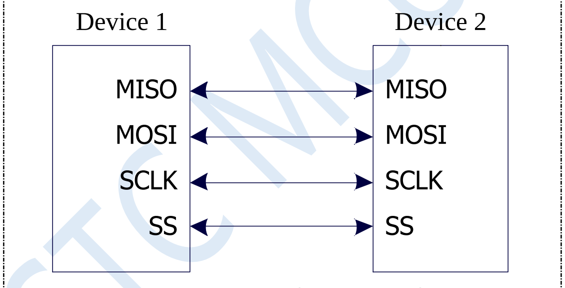
18.2.2 Dual Devices Configuration Mode

Two devices are connected, the master and the slave are not fixed.

Setting Method 1: Both devices are initialized with SSIG set to 0, MSTR set to 1, and SS pin set to bi-directional mode and output high. Now the both devices are in master mode with not ignoring SS. When one of the devices needs to initiate a transfer, set its own SS pin to output mode and output low to pull down the other device's SS pin so that the other device is forcibly set to slave mode.

Set Method 2: Both devices are initialized as slave mode with ignoring SS, where SIG is set to 1 and MSTR is set to 0. When one of the devices needs to initiate a transfer, detect the SS pin's level firstly. If SS is high, the device sets itself to master mode with ignoring SS, then starts the data transfer.

The connection configuration of dual devices configuration mode is shown as follows:



The connection configuration of dual devices configuration mode

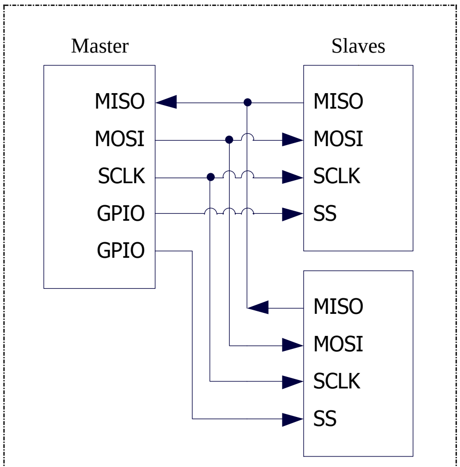
18.2.3 Single Master and Multiple Slaves Mode

Multiple devices are connected, one of which is fixed as a master and others are fixed as slaves.

Master settings: SSIG set to 1, MSTR set to 1, fixed to master mode. The master can use any port to connect with the SS pins of each slave respectively, and pull down the SS pin of one slave to enable the corresponding slave device.

Slave settings: SSIG is set to 0, SS pin is used as the chip select signal of the slave.

The configuration diagram of single master multiple slaves is as follows:



The configuration diagram of single master multiple slaves

18.3 Configure SPI

Control bits			Communication port pins				Descriptions
SPEN	SSIG	MSTR	SS	MISO	MOSI	SCLK	
0	x	x	x	input	input	input	SPI is disabled, SS/MOSI/MISO/SCLK are used as general I/O ports
1	0	0	0	output	input	input	**Selected as slave**
1	0	0	1	high impedance	input	input	**Selected as slave,** not selected.
1	0	1→0	0	output	input	input	**Slavce mode**, master mode with notignoring SS and MSTR is 1. When SS pin is pulled low, MSTR will be automatically cleared by hardware and the operating mode will be passively set to slave mode.
1	0	1	1	input	high impedance	high impedance	**Master mode,** idle state
					output	output	**Master mode, ative state**
1	1	0	x	output	input	input	**Slave mode**
1	1	1	x	input	output	output	**Master mode**

Additional Considerations for a Slave

When CPHA = 0, SSIG must be 0 (i.e. SS pin can not be ignored). The SS pin must be pulled low before each serial byte begins transfer and must be reset to high after the transfer completes. The SPDAT register can not be written while the SS pin is low, otherwise a write collision error will occur. Operation with CPHA = 0 and SSIG = 1 is undefined.

When CPHA = 1, SSIG may be set to 1 (i.e. the SS pin can be ignored). If SSIG = 0, the SS pin may remain active low (i.e., stay low all the way) for consecutive transfers. This method is suitable for fixed single master single slave system.

Additional Considerations for a Master

In SPI, transfers are always initiated by the master. If the SPI is enabled (SPEN = 1) and selected as the master, the master will initiate SPI clock generator and data transfer by writing to SPI data register, SPDAT. The data will appear on the MOSI pin a half to one SPI bit-time later after the data is written to SPDAT. The data written to the SPDAT register of the master is shifted out from the MOSI pin and sent to the MOSI pin of the slave. And, at the same time the data in SPDAT register of the selected slave is shifted out on MISO pin to the MISO pin of the master.

After one byte has been transmitted, the SPI clock generator is stopped, the transfer completion flag (SPIF) is set, and an SPI interrupt is generated if the SPI interrupt is enabled. The two shift registers for the master and slave CPUs can be considered as a 16-bit cyclic shift register. As data is shifted from the master to the slave, data is also shifted in the opposite direction simultaneously. This means that the data of the master and the slave are exchanged with each other in one shift period.

Change Mode using SS pin

If SPEN = 1, SSIG = 0 and MSTR = 1, SPI is enabled in master mode and the SS pin can be configured for input mode or quasi-bidirectional port mode. In this case, another master can drive this pin low to select the device as an SPI slave and send data to it. To avoid bus contention, the SPI system clears the slave's MSTR, forces MOSI and SCLK to be input mode, and MISO changes to output mode. The SPIF flag in SPSTAT is set, and if the SPI interrupt is enabled, an SPI interrupt will occur.

The user software must always detect the MSTR bit. If this bit is cleared by a slave selection action and the user wants to continue using the SPI as a master, the MSTR bit must be set again, otherwise it will remain in slave mode.

Write Collision

The SPI is single buffered in the transmition process and double buffered in receiving process. New data for transmission can not be written to the shift register until the previous transmission is complete. The WCOL bit will be set to indicate that a data write collision error has occurred when the data register SPDAT is written during transmission. In this case, the data currently being transmitted will continue to be transmitted, and the newly written data will be lost.

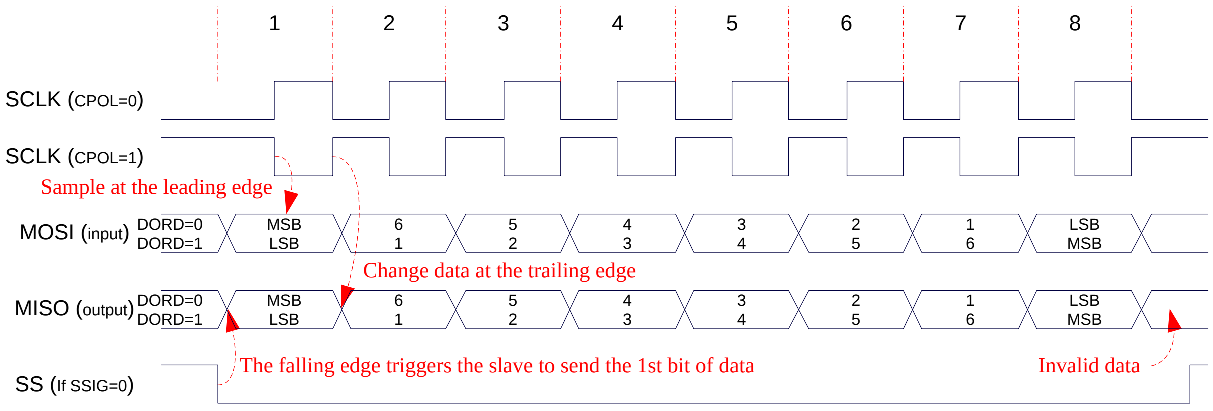
A write collision condition on the master is rare when write collision detection is performed on a master or slave because the master has full control of the data transfer. However, a write collision may occur on the slave because the slave can not control it when the master initiates the transfer.

When receiving data, the received data is transferred to a parallel read data buffer, which will release the shift register for the next data reception. However, the received data must be read from the data register before the next character is completely shifted in. Otherwise, the previous received data will be lost.

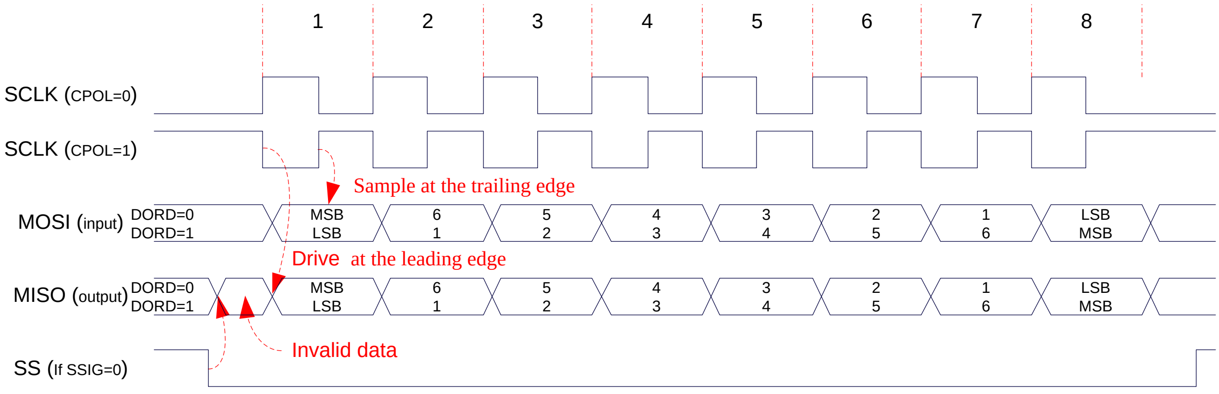
WCOL can be cleared by software by writing "1" to it.

18.4 Data Format

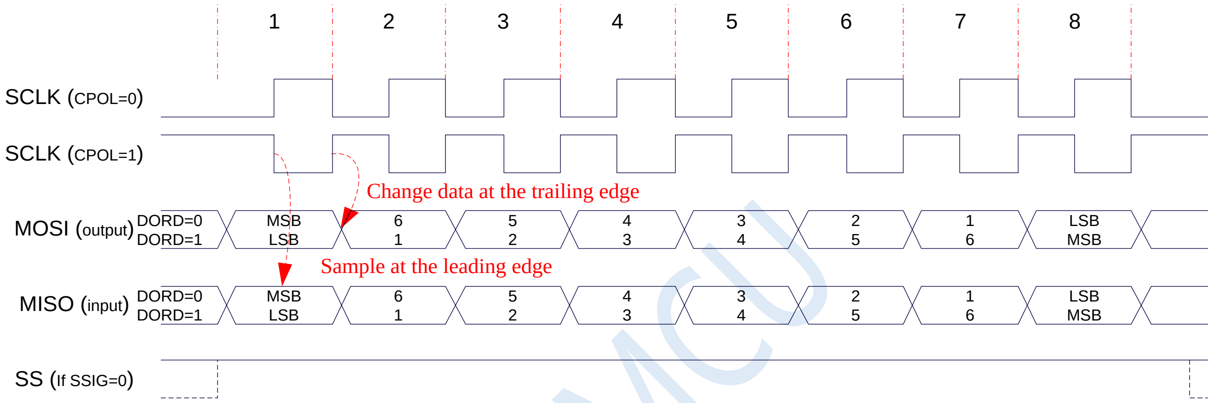
The clock phase control bit, CPHA, of the SPI allows the user to set the clock edge when the data is sampled and changed. The clock polarity bit, CPOL, allows the user to set the clock polarity. The following illustrations show the SPI communication timing under different clock phases and polarity settings.



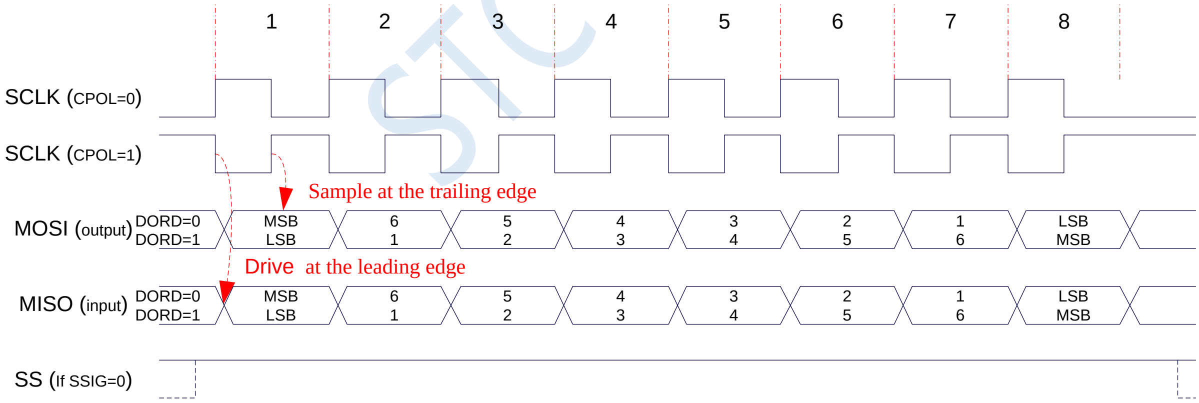
SPI slave transfer format with CPHA=0



SPI slave transfer format with CPHA=1



SPI master transfer format with CPHA=0



SPI master transfer format with CPHA=1

18.5 Example Routines

18.5.1 Master Routine of Single Master Single Slave Mode (Interrupt Mode)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         SPSTAT          =       0xcd;
sfr         SPCTL           =       0xce;
sfr         SPDAT           =       0xcf;
sfr         IE2             =       0xaf;
#define     ESPI                    0x02

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

sbit        SS              =       P1^0;
sbit        LED             =       P1^1;

bit         busy;

void SPI_Isr() interrupt 9
{
    SPSTAT = 0xc0;                                      //Clear interrupt flag
    SS = 1;                                             //Pull up the SS pin of the slave
    busy = 0;
    LED = !LED;                                         //Test port
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
```

```
        P5M0 = 0x00;
        P5M1 = 0x00;

        LED = 1;
        SS = 1;
        busy = 0;

        SPCTL = 0x50;                                   //Enable SPI master mode
        SPSTAT = 0xc0;                                  //Clear interrupt flag
        IE2 = ESPI;                                     //Enable SPI interrupt
        EA = 1;

        while (1)
        {
                while (busy);
                busy = 1;
                SS = 0;                                 //Pull down the slave SS pin
                SPDAT = 0x5a;                           //Send test data
        }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

SPSTAT          DATA            0CDH
SPCTL           DATA            0CEH
SPDAT           DATA            0CFH
IE2             DATA            0AFH
ESPI            EQU             02H

BUSY            BIT             20H.0
SS              BIT             P1.0
LED             BIT             P1.1

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             004BH
                LJMP            SPIISR

                ORG             0100H
SPIISR:
                MOV             SPSTAT,#0C0H            ;Clear interrupt flag
                SETB            SS                      ;Pull up the SS pin of the slave
                CLR             BUSY
                CPL             LED
                RETI
```

```
MAIN:
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                SETB        LED
                SETB        SS
                CLR         BUSY

                MOV         SPCTL,#50H                  ;Enable SPI master mode
                MOV         SPSTAT,#0C0H                ;Clear interrupt flag
                MOV         IE2,#ESPI                   ;Enable SPI interrupt
                SETB        EA

LOOP:
                JB          BUSY,$
                SETB        BUSY
                CLR         SS                          ;Pull down the slave SS pin
                MOV         SPDAT,#5AH                  ;Send test data
                JMP         LOOP

                END
```

18.5.2 Slave Routine of Single Master Single Slave Mode (Interrupt Mode)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         SPSTAT      =       0xcd;
sfr         SPCTL       =       0xce;
sfr         SPDAT       =       0xcf;
sfr         IE2         =       0xaf;
#define     ESPI                0x02

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

sbit        LED             =       P1^1;

void SPI_Isr() interrupt 9
{
        SPSTAT = 0xc0;                                  //Clear interrupt flag
        SPDAT = SPDAT;                                  //Transmit the received data back to the master
        LED = !LED;                                     //Test port
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        SPCTL = 0x40;                                   //Enable SPI slave mode
        SPSTAT = 0xc0;                                  //Clear interrupt flag
        IE2 = ESPI;                                     //Enable SPI interrupt
        EA = 1;

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

SPSTAT          DATA            0CDH
SPCTL           DATA            0CEH
SPDAT           DATA            0CFH
IE2             DATA            0AFH
ESPI            EQU             02H

LED             BIT             P1.1

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
```

```
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             004BH
                LJMP            SPIISR

                ORG             0100H
SPIISR:
                MOV             SPSTAT,#0C0H                    ;Clear interrupt flag
                MOV             SPDAT,SPDAT                     ;Transmit the received data back to the master
                CPL             LED
                RETI

MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             SPCTL,#40H                      ;Enable SPI slave mode
                MOV             SPSTAT,#0C0H                    ;Clear interrupt flag
                MOV             IE2,#ESPI                       ;Enable SPI interrupt
                SETB            EA

                JMP             $

                END
```

18.5.3 Master Routine of Single Master Single Slave Mode (Polling Mode)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

sfr         SPSTAT          =       0xcd;
sfr         SPCTL           =       0xce;
sfr         SPDAT           =       0xcf;
sfr         IE2             =       0xaf;
#define     ESPI                    0x02

sbit        SS              =       P1^0;
sbit        LED             =       P1^1;

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        LED = 1;
        SS = 1;

        SPCTL = 0x50;                                   //Enable SPI master mode
        SPSTAT = 0xc0;                                  //Clear interrupt flag

        while (1)
        {
                SS = 0;                                 //Pull down the slave SS pin
                SPDAT = 0x5a;                           //Send test data
                while (!(SPSTAT & 0x80));               //Query completion flag
                SPSTAT = 0xc0;                          //Clear interrupt flag
                SS = 1;                                 //Pull up the SS pin of the slave
                LED = !LED;                             //Test port
        }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

SPSTAT          DATA            0CDH
SPCTL           DATA            0CEH
SPDAT           DATA            0CFH
IE2             DATA            0AFH
ESPI            EQU             02H

SS              BIT             P1.0
LED             BIT             P1.1
```

```
P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN

            ORG         0100H
MAIN:
            MOV         SP, #5FH
            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
            MOV         P3M1, #00H
            MOV         P4M0, #00H
            MOV         P4M1, #00H
            MOV         P5M0, #00H
            MOV         P5M1, #00H

            SETB        LED
            SETB        SS

            MOV         SPCTL,#50H              ;Enable SPI master mode
            MOV         SPSTAT,#0C0H            ;Clear interrupt flag

LOOP:
            CLR         SS                      ;Pull down the slave SS pin
            MOV         SPDAT,#5AH              ;Send test data
            MOV         A,SPSTAT                ;Query completion flag
            JNB         ACC.7,$-2
            MOV         SPSTAT,#0C0H            ;Clear interrupt flag
            SETB        SS
            CPL         LED
            JMP         LOOP

            END
```

18.5.4 Slave Routine of Single Master Single Slave Mode (Polling Mode)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         SPSTAT          =       0xcd;
sfr         SPCTL           =       0xce;
sfr         SPDAT           =       0xcf;
sfr         IE2             =       0xaf;
#define     ESPI                    0x02

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

sbit        LED             =       P1^1;

void SPI_Isr() interrupt 9
{
    SPSTAT = 0xc0;                                  //Clear interrupt flag
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    SPCTL = 0x40;                                   //Enable SPI slave mode
    SPSTAT = 0xc0;                                  //Clear interrupt flag

    while (1)
    {
        while (!(SPSTAT & 0x80));                   //Query completion flag
        SPSTAT = 0xc0;                              //Clear interrupt flag
        SPDAT = SPDAT;                              //Transmit the received data back to the master
        LED = !LED;                                 //Test port
    }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

SPSTAT          DATA            0CDH
SPCTL           DATA            0CEH
SPDAT           DATA            0CFH
IE2             DATA            0AFH
ESPI            EQU             02H

LED             BIT             P1.1

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN

                ORG             0100H
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             SPCTL,#40H                      ;Enable SPI slave mode
                MOV             SPSTAT,#0C0H                    ;Clear interrupt flag

LOOP:
                MOV             A,SPSTAT                        ;Query completion flag
                JNB             ACC.7,$-2
                MOV             SPSTAT,#0C0H                    ;Clear interrupt flag
                MOV             SPDAT,SPDAT                     ;Transmit the received data back to the master
                CPL             LED
                JMP             LOOP

                END
```

18.5.5 Routine of Mutual Master-Slave Mode (Interrupt Mode)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         SPSTAT          =       0xcd;
sfr         SPCTL           =       0xce;
sfr         SPDAT           =       0xcf;
sfr         IE2             =       0xaf;
#define     ESPI                    0x02

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

sbit        SS              =       P1^0;
sbit        LED             =       P1^1;
sbit        KEY             =       P0^0;

void SPI_Isr() interrupt 9
{
        SPSTAT = 0xc0;                                  //Clear interrupt flag
        if (SPCTL & 0x10)
        {                                               //Master mode
                SS = 1;                                 //Pull up the SS pin of the slave
                SPCTL = 0x40;                           //Reset to slave and standby
        }
        else
        {                                               //Slave mode
                SPDAT = SPDAT;                          //Transmit the received data back to the master
        }
        LED = !LED;                                     //Test port
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
```

```
    P5M0 = 0x00;
    P5M1 = 0x00;

    LED = 1;
    KEY = 1;
    SS = 1;

    SPCTL = 0x40;                          //Enable SPI slave modeand standby
    SPSTAT = 0xc0;                         //Clear interrupt flag
    IE2 = ESPI;                            //Enable SPI interrupt
    EA = 1;

    while (1)
    {
        if (!KEY)                          //Wait for the key to trigger
        {
            SPCTL = 0x50;                  //Enable SPI master mode
            SS = 0;                        //Pull down the slave SS pin
            SPDAT = 0x5a;                  //Send test data
            while (!KEY);                  //Wait for the keys to be released
        }
    }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

SPSTAT      DATA        0CDH
SPCTL       DATA        0CEH
SPDAT       DATA        0CFH
IE2         DATA        0AFH
ESPI        EQU         02H

SS          BIT         P1.0
LED         BIT         P1.1
KEY         BIT         P0.0

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         004BH
            LJMP        SPIISR

            ORG         0100H
SPIISR:
            PUSH        ACC
            MOV         SPSTAT,#0C0H            ;Clear interrupt flag
```

```
                MOV         A,SPCTL
                JB          ACC.4,MASTER
SLAVE:
                MOV         SPDAT,SPDAT                 ;Transmit the received data back to the master
                JMP         ISREXIT
MASTER:
                SETB        SS                          ;Pull up the SS pin of the slave
                MOV         SPCTL,#40H                  ;Reset to slave and standby
ISREXIT:
                CPL         LED
                POP         ACC
                RETI

MAIN:
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                SETB        SS
                SETB        LED
                SETB        KEY

                MOV         SPCTL,#40H                  ;Enable SPI slave mode and standby
                MOV         SPSTAT,#0C0H                ;Clear interrupt flag
                MOV         IE2,#ESPI                   ;Enable SPI interrupt
                SETB        EA

LOOP:
                JB          KEY,LOOP                    ;Wait for the key to trigger
                MOV         SPCTL,#50H                  ;Enable SPI master mode
                CLR         SS                          ;Pull down the slave SS pin
                MOV         SPDAT,#5AH                  ;Send test data
                JNB         KEY,$                       ;Wait for the keys to be released
                JMP         LOOP

                END
```

18.5.6 Routine of Mutual Master-Slave Mode (Polling Mode)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         SPSTAT      =       0xcd;
sfr         SPCTL       =       0xce;
sfr         SPDAT          =       0xcf;
sfr         IE2            =       0xaf;
#define     ESPI                   0x02

sfr         P0M1           =       0x93;
sfr         P0M0           =       0x94;
sfr         P1M1           =       0x91;
sfr         P1M0           =       0x92;
sfr         P2M1           =       0x95;
sfr         P2M0           =       0x96;
sfr         P3M1           =       0xb1;
sfr         P3M0           =       0xb2;
sfr         P4M1           =       0xb3;
sfr         P4M0           =       0xb4;
sfr         P5M1           =       0xc9;
sfr         P5M0           =       0xca;

sbit        SS             =       P1^0;
sbit        LED            =       P1^1;
sbit        KEY            =       P0^0;

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        LED = 1;
        KEY = 1;
        SS = 1;

        SPCTL = 0x40;                                   //Enable SPI slave mode and standby
        SPSTAT = 0xc0;                                  //Clear interrupt flag

        while (1)
        {
                if (!KEY)                               //Wait for the key to trigger
                {
                        SPCTL = 0x50;                   //Enable SPI master mode
                        SS = 0;                         //Pull down the slave SS pin
                        SPDAT = 0x5a;                   //Send test data
                        while (!KEY);                   //Wait for the keys to be released
                }
                if (SPSTAT & 0x80)
                {
                        SPSTAT = 0xc0;                  //Clear interrupt flag
                        if (SPCTL & 0x10)
                        {                               //Master mode
                                SS = 1;                 //Pull up the SS pin of the slave
                                SPCTL = 0x40;           //Reset to slave and standby
                        }
                        else
                        {                                                   //Slave mode
                                SPDAT = SPDAT;                              //Transmit the received data back to the master
                        }
                        LED = !LED;                                         //Test port
                }
        }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

SPSTAT          DATA            0CDH
SPCTL           DATA            0CEH
SPDAT           DATA            0CFH
IE2             DATA            0AFH
ESPI            EQU             02H

SS              BIT             P1.0
LED             BIT             P1.1
KEY             BIT             P0.0

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN

                ORG             0100H
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                SETB            SS
                SETB            LED
                SETB            KEY
                MOV         SPCTL,#40H              ;Enable SPI slave mode and standby
                MOV         SPSTAT,#0C0H            ;Clear interrupt flag

LOOP:
                JB          KEY,SKIP                ;Wait for the key to trigger
                MOV         SPCTL,#50H              ;Enable SPI master mode
                CLR         SS                      ;Pull down the slave SS pin
                MOV         SPDAT,#5AH              ;Send test data
                JNB         KEY,$                   ;Wait for the keys to be released
SKIP:
                MOV         A,SPSTAT
                JNB         ACC.7,LOOP
                MOV         SPSTAT,#0C0H            ;Clear interrupt flag
                MOV         A,SPCTL
                JB          ACC.4,MASTER
SLAVE:
                MOV         SPDAT,SPDAT             ;Transmit the received data back to the master
                CPL         LED
                JMP         LOOP
MASTER:
                SETB        SS                      ;Pull up the SS pin of the slave
                MOV         SPCTL,#40H              ;Reset to slave and standby
                CPL         LED
                JMP         LOOP

                END
```

19 I2C Bus

Product line	I2C
STC8H1K08 family	●
STC8H1K28 family	●
STC8H3K64S4 family	●
STC8H3K64S2 family	●
STC8H8K64U family	●
STC8H2K64T family	●
STC8H4K64TLR family	●
STC8H4K64TLCD family	●
STC8H4K64LCD family	●

An I^{2}C serial bus controller is integrated in the STC8H series of microcontrollers. I^{2}C is a high-speed synchronous communication bus, which uses SCL (clock line) and SDA (data line) to carry out two-wire synchronous communication. For the port allocation of SCL and SDA, STC8H series of microcontrollers provide pin switch mode that can switch SCL and SDA to different I/O ports. Therefor, it is convenience to use a set of I^{2}C as multiple sets of I^{2}C buses through time sharing.

Compared with the standard I^{2}C protocol, the following two mechanisms are ignored:

- No arbitration will be performed after the start signal (START) is sent.
- No timeout detection when the clock signal (SCL) stays at low level.

The I^{2}C bus of the STC8H series of microcontrollers offer two modes of operation: master mode (SCL is the output port, which is used to transmit synchronous clock signal) and slave mode (SCL is the input port, which is used to receive the synchronous clock signal).

STC innovation: When the I^{2}C serial bus controller of STC works in slave mode, the falling edge signal of SDA pin can wake up the MCU which is in power-down mode. (Note: Due to the fast I^{2}C transmission speed, the first packet of data after the MCU wakes up is generally incorrect.)

19.1 Registers Related to I^{2}C

Symbol	Description	Address	Bit Address and Symbol								Reset Value
			B7	B6	B5	B4	B3	B2	B1	B0	
I2CCFG	I2C Configuration Register	FE80H	ENI2C	MSSL	MSSPEED[6:1]						0000,0000
I2CMSCR	I^{2}C Master Control Register	FE81H	EMSI	-	-	-	MSCMD[3:0]				0xxx,0000
I2CMSST	I^{2}C Master Status Register	FE82H	MSBUSY	MSIF	-	-	-	-	MSACKI	MSACKO	00xx,xx00
I2CSLCR	I^{2}C Slave Control Register	FE83H	-	ESTAI	ERXI	ETXI	ESTOI	-	-	SLRST	x000,0xx0
I2CSLST	I^{2}C Slave Status Register	FE84H	SLBUSY	STAIF	RXIF	TXIF	STOIF	TXING	SLACKI	SLACKO	0000,0000
I2CSLADR	I^{2}C Slave Address Register	FE85H	SLADR[6:0]							MA	0000,0000
I2CTXD	I^{2}C Data Transmission Register	FE86H									0000,0000
I2CRXD	I^{2}C Data Receive Register	FE87H									0000,0000
I2CMSAUX	I^{2}C Master Auxiliary Control Register	FE88H	-	-	-	-	-	-	-	WDTA	xxxx,xxx0

19.2 I^{2}C Master Mode

19.2.1 I2C Configuration Register (I2CCFG)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
I2CCFG	FE80H	ENI2C	MSSL	MSSPEED[5:0]					

ENI2C: I^{2}C function enable bit

0: disable I^{2}C function

1: enable I^{2}Cfunction

MSSL: I^{2}C mode selection bit

0: Salve mode

1: Master mode

MSSPEED[5:0]: I^{2}C bus speed control bits (clocks to wait), **I2C bus speed＝F_{OSC} / 2 / (MSSPEED * 2 + 4)**

MSSPEED[5:0]	Corresponding clocks
0	4
1	6
2	8
…	…
x	2x+4
…	…
62	128
63	130

The waiting parameter set by the MSSPEED is valid only when the I^{2}C module is operating in the master mode. The waiting parameter is mainly used for the following signals in master mode:

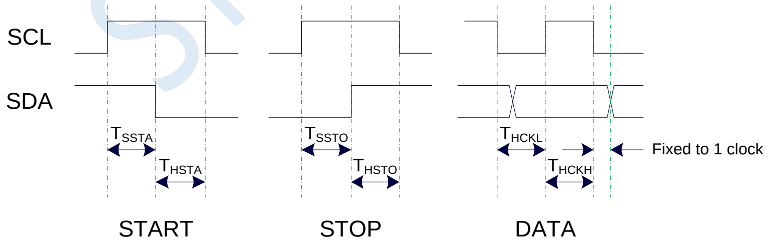
T_{SSTA}: Setup Time of START

T_{HSTA}: Hold Time of START

T_{SSTO}: Setup Time of STOP

T_{HSTO}: Hold Time of STOP

T_{HCKL}: Hold Time of SCL Low



Example 1: When MSSPEED＝10, T_{SSTA}＝T_{HSTA}＝T_{SSTO}＝T_{HSTO}＝T_{HCKL}＝24/FOSC

Example 2: When 400K I2C bus speed is required at 24MHz operating frequency, MSSPEED＝(24M / 400K / 2 - 4) / 2＝13

19.2.2 I^{2}C Master Control Register (I2CMSCR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
I2CMSCR	FE81H	EMSI	-	-	-	MSCMD[3:0]			

EMSI: Master mode interrupt enable control bit

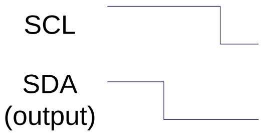
0: disable master mode interrupt

1: enable master mode interrupt

MSCMD[3:0]: master command bits

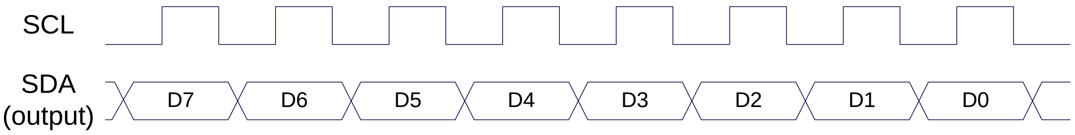
0000: Standby, no action

0001: START command. Send a START signal. If the I^{2}C controller is in idle state currently, i.e. MSBUSY (I2CMSST.7) is 0, writing this command will make the controller enter the busy status, and the hardware will set the MSBUSY status bit automatically and start sending START signal. **If the I^{2}C controller is busy currently**, writing this command will triger to send the START signal. Sending the START signal waveform is shown below:



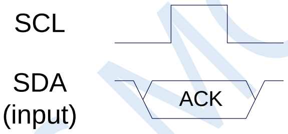
0010: Send data command.

After writing this command, the I^{2}C bus controller will generate 8 clocks on the SCL pin and send the data in the I2CTXD register bit by bit to the SDA pin (send MSB firstly). The waveform of the transmitting data is shown in the following figure:



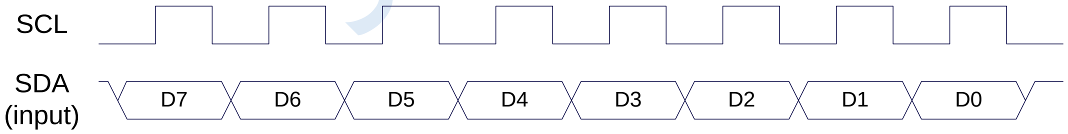
0011: Receive ACK command.

After writing this command, the I^{2}C bus controller will generate a clock on the SCL pin and save the data bit read from SDA to MSACKI (I2CMSST.1). The waveform of the receiving ACK is shown below:



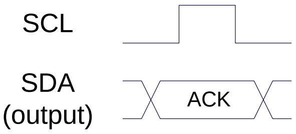
0100: Receive data command.

After writing this command, the I^{2}C bus controller will generate 8 clocks on the SCL pin, and sequentially shift the data bit read from SDA to the I2CRXD register (receiving MSB firstly). The waveform of the receiving data is as shown in the figure below:



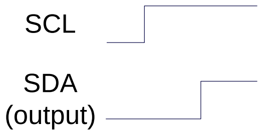
0101: Send ACK command.

After writing this command, the I^{2}C bus controller will generate a clock on the SCL pin and send the data bit in MSACKO (I2CMSST.0) to SDA. The waveform of sending ACK is shown below:



0110: Send STOP signal.

After writing this command, the I^{2}C bus controller starts sending STOP signal. After the signal is sent, the hardware clears the MSBUSY status bit automatically. The waveform of STOP signal is shown below:



0111: Reserved.

1000: Reserved.

1001: Start command + send data command + receive ACK command.

This command is a combination of command 0001, command 0010 and command 0011. After wrting this command, the controller will execute these three commands in sequence.

1010: Send data command + receive ACK command.

This command is a combination of command 0010 and command 0011. After writing this command, the controller will execute these two commands in sequence.

1011: Receive data command + send ACK (0) command.

This command is a combination of command 0100 and command 0101. After writing this command, the controller will execute these two commands in sequence.

Note: The response signal returned by this command is fixed as ACK (0) and is not affected by the MSACKO bit.

1100: Receive data command + send NAK (1) command.

This command is a combination of command 0100 and command 0101. After writing this command, the controller will execute these two commands in sequence.

Note: The response signal returned by this command is fixed to NAK (1), and is not affected by the MSACKO bit.

19.2.3 I^{2}C Master Auxiliary Control Register (I2CMSAUX)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
I2CMSAUX	FE88H	-	-	-	-	-	-	-	WDTA

WDTA: I2C data automatic transmission enable bit in master mode.

0: disable automatic transmission

1: enable automatic transmission

If the automatic transmission function is enabled, when the MCU finishes writing to the I2CTXD data register, the I^{2}C controller will trigger the "1010" command automatically, that is, it will send data automatically and receive the ACK signal.

19.2.4 I^{2}C Master Status Register (I2CMSST)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
I2CMSST	FE82H	MSBUSY	MSIF	-	-	-	-	MSACKI	MSACKO

MSBUSY: status bit of I^{2}C controller in master mode. (Read-only)

0: the controller is in idle state.

1: the controller is in busy state.

When the I^{2}C controller is in master mode, the controller will enter the busy state after sending the START signal in the idle state. The busy state will be maintained until the STOP signal is successfully transmitted, and the state will return to the idle.

MSIF: master mode interrupt request bit (interrupt flag bit). When the I^{2}C controller in the master mode executes the MSCMD command in the completion register I2CMSCR, it generates an interrupt signal. This bit is set to 1 by hardware automatically to request an interrupt to CPU. The MSIF bit must be cleared by software after responding to the interrupt.

MSACKI: In master mode, it is the ACK datum received after sending the “011” command to the MSCMD bit in I2CMSCR.

MSACKO: In master mode, it is the ACK signal ready to be transmitted. When the “101” command is sent to the MSCMD bit of I2CMSCR, the controller will read the datum of this bit automatically and send it as

ACK to SDA.

19.3 I²C Slave Mode

19.3.1 I²C Slave Control Register (I2CSLCR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
I2CSLCR	FE83H	-	ESTAI	ERXI	ETXI	ESTOI	-	-	SLRST

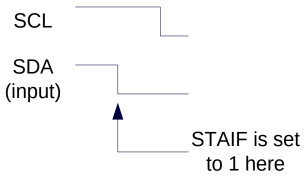
ESTAI: interrupt enable bit when receiving START signal in slave mode.
0: disable interrupt when receiving START signal in slave mode.
1: enable interrupt when receiving START signal in slave mode.
ERXI: interrupt enable bit after 1 byte datum is received in slave mode
0: disable interrupt after a datum is received in slave mode.
1: enable interrupt after 1 byte datum is reveived in slave mode.
ERXI: interrupt enable bit after 1 byte datum is sent in slave mode
0: disable interrupt after a datum is sent in slave mode.
1: enable interrupt after 1 byte datum is sent in slave mode.
ESTOI: interrupt enable bit after STOP signal is received in slave mode.
0: disable interrupt after STOP signal is received in slave mode.
1: enable interrupt after STOP signal is received in slave mode.
SLRST: reset slave mode

19.3.2 I²C Slave Status Register (I2CSLST)

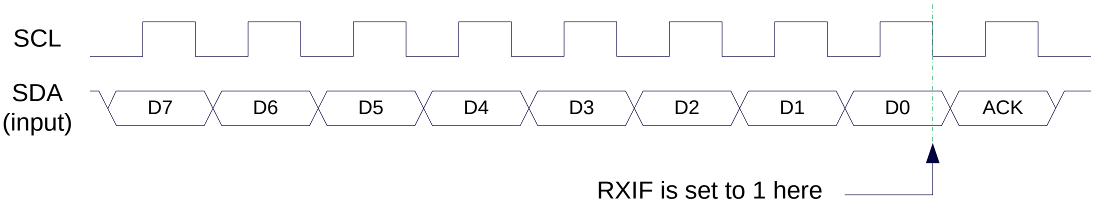
Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
I2CSLST	FE84H	SLBUSY	STAIF	RXIF	TXIF	STOIF	-	SLACKI	SLACKO

SLBUSY: status bit of I²C controller in slave mode. (Read-only)
0: the controller is in idle state.
1: the controller is in busy state.
When the I²C controller is in slave mode, the controller will continue to detect the subsequent device address data when it receives the START signal from the master in idle state. If the device address matches the slave address set in the current I2CSLADR register, the controller will enter the busy state. And the busy state will be maintained until receives a STOP signal sent by the master successfully, and then the state will return to idle.

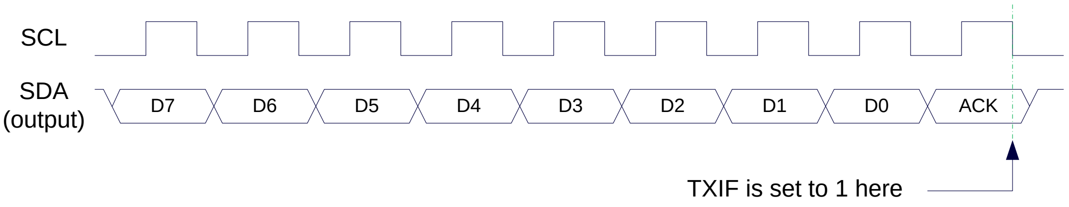
STAIF: interrupt request bit after START signal is received in slave mode. After the I²C controller in slave mode receives the START signal, this bit is set by hardware automatically and requests interrupt to CPU. The STAIF bit must be cleared by software after the interrupt is responded. The time point of STAIF being set is shown below:



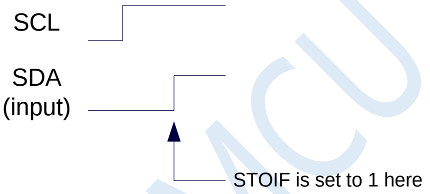
RXIF: interrupt request bit after 1-byte datum is received in slave mode. After the I²C controller in slave mode receives a 1-byte datum, this bit is set by hardware automatically at the falling edge of the 8th clock and will request interrupt to CPU. The RXIF bit must be cleared by software after the interrupt is responded. The time point of RXIF being set is shown in the figure below:



TXIF: interrupt request bit after 1-byte datum transmission is completed in slave mode. After the I^{2}C controller in slave mode completes sending 1 byte of datum and receives a 1-bit ACK signal successfully, this bit is set by hardware automatically at the falling edge of the 9th clock and requests an interrupt to CPU. TXIF bit must be cleared by software after the interrupt is responded. The time point of TXIF being set is shown below:

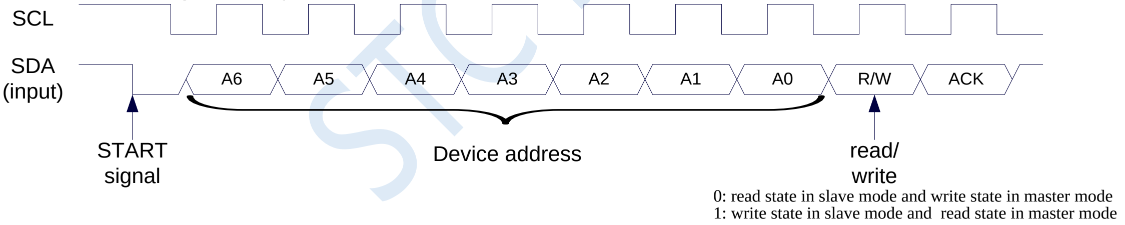


STOIF: interrupt request bit after STOP signal is received in slave mode. After the I^{2}C controller in slave mode receives the STOP signal, this bit is set by hardware automatically and requests interrupt to CPU. The STOIF bit must be cleared by software after the interrupt is serviced. The time point of STOIF being set is shown below:



SLACKI: ACK data received in slave mode

SLACKO: the ACK signal ready to send out in slave mode.


SCL
SDA
(input)
A6
A5
A4
A3
A2
A1
A0
R/W
ACK
START
signal
Device address
read/
write
0: read state in slave mode and write state in master mode
1: write state in slave mode and read state in master mode

19.3.3 I^{2}C Slave Address Register (I2CSLADR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
I2CSLADR	FE85H	I2CSLADR[7:1]							MA

I2CSLADR[7:1]: the slave device address

When the I^{2}C controller is in slave mode, the controller will continue to detect the device address and read / write signals sent by the master after it receives the START signal. If the device address sent by the master matches the slave device address set in SLADR[6: 0], the controller will request an interrupt to CPU to process the I^{2}C event. Otherwise, if the device address does not match, the I^{2}C controller continues to monitor, wait for the next START signal, and match the next device address.

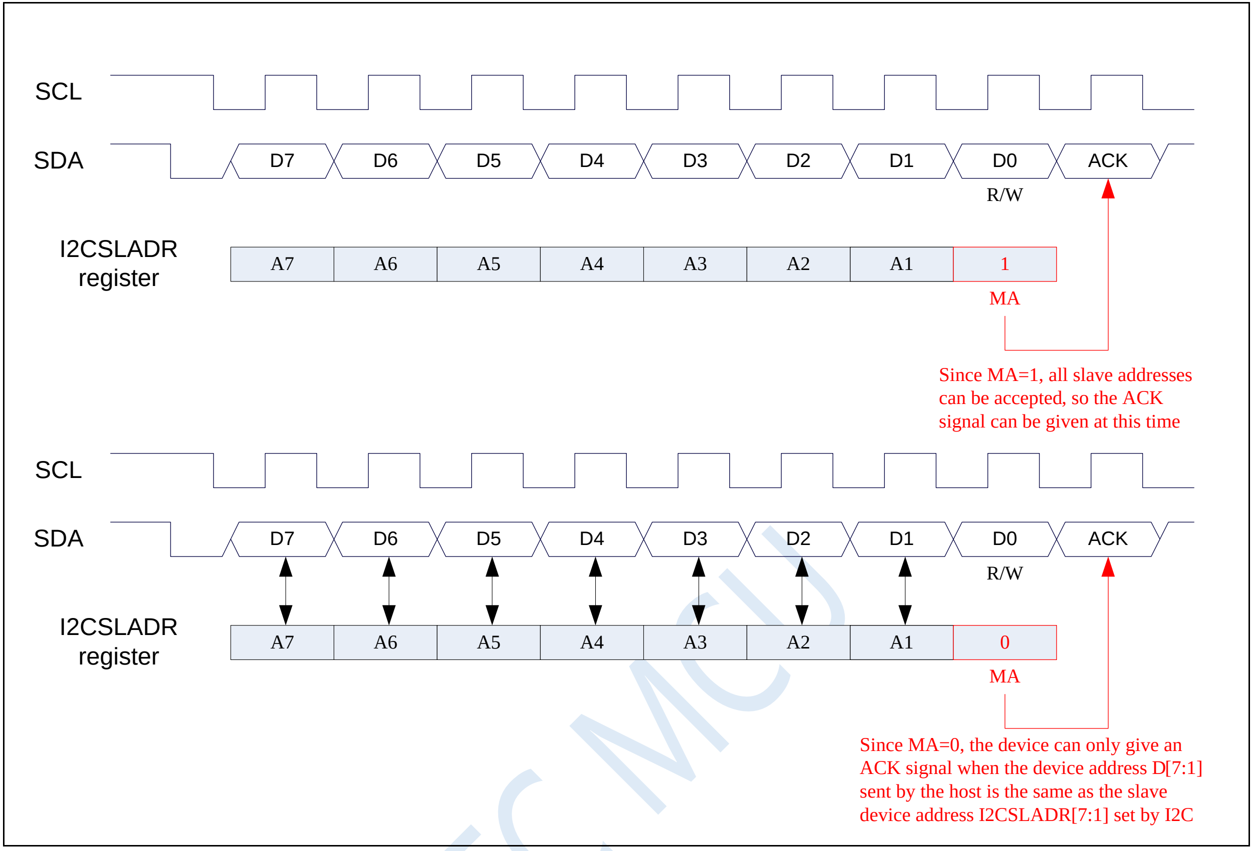
MA: Slave device address matching control bit

0: The device address must be the same as I2CSLADR[7:1].

1: Ignore the settings in I2CSLADR[7:1] and match all device addresses.

Note: The I2C bus protocol stipulates that a maximum of 128 I2C devices (theoretical value) can be mounted on the I2C bus, and different I2C devices are identified by different I2C slave device addresses. After the I2C master sends the start signal, the upper 7 bits of the first data (DATA0) sent are the slave device address (DATA0[7:1] is the I2C device address), and the lowest bit is the read and

write signal. When the I2C device slave address register MA (I2CSLADR.0) is 1, it means that the I2C slave can accept all device addresses. At this time, any device address sent by the host, that is, DATA0[7:1] is any value, the slave Can respond. When I2C device slave address register MA (I2CSLADR.0)
When it is 0, the device address DATA0[7:1] sent by the host must be the same as the device address I2CSLADR[7:1] of the slave to access this slave device


SCL
SDA
D7
D6
D5
D4
D3
D2
D1
D0
ACK
R/W
I2CSLADR register
A7
A6
A5
A4
A3
A2
A1
1
MA
Since MA=1, all slave addresses can be accepted, so the ACK signal can be given at this time
SCL
SDA
D7
D6
D5
D4
D3
D2
D1
D0
ACK
R/W
I2CSLADR register
A7
A6
A5
A4
A3
A2
A1
0
MA
Since MA=0, the device can only give an ACK signal when the device address D[7:1] sent by the host is the same as the slave device address I2CSLADR[7:1] set by I2C

19.3.4 I^{2}C data registers (I2CTXD, I2CRXD)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
I2CTXD	FE86H								
I2CRXD	FE87H								

I2CTXD is the I^{2}C transmit data register that holds the I^{2}C data to be transmitted.
I2CRXD is the I^{2}C receive data register that holds the I^{2}C data received.

19.4 Example Routines

19.4.1 I^{2}C is Used to Access AT24C256 in Master Mode (Interrupt Mode)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P_SW2           =       0xba;

#define     I2CCFG                  (*(unsigned char volatile xdata *)0xfe80)
#define     I2CMSCR                 (*(unsigned char volatile xdata *)0xfe81)
#define     I2CMSST                 (*(unsigned char volatile xdata *)0xfe82)
#define     I2CSLCR                 (*(unsigned char volatile xdata *)0xfe83)
#define     I2CSLST                 (*(unsigned char volatile xdata *)0xfe84)
#define     I2CSLADR                (*(unsigned char volatile xdata *)0xfe85)
#define     I2CTXD                  (*(unsigned char volatile xdata *)0xfe86)
#define     I2CRXD                  (*(unsigned char volatile xdata *)0xfe87)

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

sbit        SDA             =       P1^4;
sbit        SCL             =       P1^5;

bit         busy;

void I2C_Isr() interrupt 24
{
        _push_(P_SW2);
        P_SW2 |= 0x80;
        if (I2CMSST & 0x40)
        {
                I2CMSST &= ~0x40;                               //Clear interrupt flag
                busy = 0;
        }
        _pop_(P_SW2);
}

void Start()
{
        busy = 1;
        I2CMSCR = 0x81;                                         //Send START command
        while (busy);
}

void SendData(char dat)
{
        I2CTXD = dat;                                           //Write data to the data buffer
        busy = 1;
        I2CMSCR = 0x82;                                         //Send a SEND command
        while (busy);
}

void RecvACK()
{
        busy = 1;
        I2CMSCR = 0x83;                                         //Send read ACK command
        while (busy);
}

char RecvData()
{
        busy = 1;
        I2CMSCR = 0x84;                                         //Send RECV command
        while (busy);
        return I2CRXD;
}

void SendACK()
{
        I2CMSST = 0x00;                                         //Setup the ACK signal
        busy = 1;
        I2CMSCR = 0x85;                                         //Send ACK command
        while (busy);
}

void SendNAK()
{
        I2CMSST = 0x01;                                         //Setup the NAK signal
        busy = 1;
        I2CMSCR = 0x85;                                         //Send ACK command
        while (busy);
}

void Stop()
{
        busy = 1;
        I2CMSCR = 0x86;                                         //Send STOP command
        while (busy);
}

void Delay()
{
        int i;

        for (i=0; i<3000; i++)
        {
                _nop_();
                _nop_();
                _nop_();
```

```
            _nop_();
        }
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P_SW2 = 0x80;

        I2CCFG = 0xe0;                                  //Enable I2C master mode
        I2CMSST = 0x00;
        EA = 1;

        Start();     //Send start command
        SendData(0xa0);                                 //Send device address + write command
        RecvACK();
        SendData(0x00);                                 //Send storage address high byte
        RecvACK();
        SendData(0x00);                                 //Send storage address low byte
        RecvACK();
        SendData(0x12);                                 //Write test data 1
        RecvACK();
        SendData(0x78);                                 //Write test data 2
        RecvACK();
        Stop();                                         //Send stop command

        Delay();                                        //Waiting for the device to write data

        Start();                                        //Send start command
        SendData(0xa0);                                 //Send device address + write command
        RecvACK();
        SendData(0x00);                                 //Send storage address high byte
        RecvACK();
        SendData(0x00);                                 //Send storage address low byte
        RecvACK();
        Start();                                        //Send start command
        SendData(0xa1);                                 //Send device address + read command
        RecvACK();
        P0 = RecvData();                                //Read data 1
        SendACK();
        P2 = RecvData();                                //Read data 2
        SendNAK();
        Stop();                                         //Send stop command

        P_SW2 = 0x00;

        while (1);
```

```
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P_SW2       DATA        0BAH

I2CCFG      XDATA       0FE80H
I2CMSCR     XDATA       0FE81H
I2CMSST     XDATA       0FE82H
I2CSLCR     XDATA       0FE83H
I2CSLST     XDATA       0FE84H
I2CSLADR    XDATA       0FE85H
I2CTXD      XDATA       0FE86H
I2CRXD      XDATA       0FE87H

SDA         BIT         P1.4
SCL         BIT         P1.5

BUSY        BIT         20H.0

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         00C3H
            LJMP        I2CISR

            ORG         0100H
I2CISR:
            PUSH        ACC
            PUSH        DPL
            PUSH        DPH

            MOV         DPTR,#I2CMSST           ;Clear interrupt flag
            MOVX        A,@DPTR
            ANL         A,#NOT 40H
            MOV         DPTR,#I2CMSST
            MOVX        @DPTR,A
            CLR         BUSY                    ;Reset busy flag

            POP         DPH
            POP         DPL
            POP         ACC
            RETI

START:
            SETB        BUSY
```

```
                MOV         A,#10000001B               ;Send START command
                MOV         DPTR,#I2CMSCR
                MOVX        @DPTR,A
                JMP         WAIT
SENDDATA:
                MOV         DPTR,#I2CTXD               ;Write data to the data buffer
                MOVX        @DPTR,A
                SETB        BUSY
                MOV         A,#10000010B               ;Send a SEND command
                MOV         DPTR,#I2CMSCR
                MOVX        @DPTR,A
                JMP         WAIT
RECVACK:
                SETB        BUSY
                MOV         A,#10000011B               ;Send read ACK command
                MOV         DPTR,#I2CMSCR
                MOVX        @DPTR,A
                JMP         WAIT
RECVDATA:
                SETB        BUSY
                MOV         A,#10000100B               ;Send RECV command
                MOV         DPTR,#I2CMSCR
                MOVX        @DPTR,A
                CALL        WAIT
                MOV         DPTR,#I2CRXD               ;Read data from the data buffer
                MOVX        A,@DPTR
                RET
SENDACK:
                MOV         A,#00000000B               ;Setup the ACK signal
                MOV         DPTR,#I2CMSST
                MOVX        @DPTR,A
                SETB        BUSY
                MOV         A,#10000101B               ;Send ACK command
                MOV         DPTR,#I2CMSCR
                MOVX        @DPTR,A
                JMP         WAIT
SENDNAK:
                MOV         A,#00000001B               ;Setup the NAK signal
                MOV         DPTR,#I2CMSST
                MOVX        @DPTR,A
                SETB        BUSY
                MOV         A,#10000101B               ;Send ACK command
                MOV         DPTR,#I2CMSCR
                MOVX        @DPTR,A
                JMP         WAIT
STOP:
                SETB        BUSY
                MOV         A,#10000110B               ;Send STOP command
                MOV         DPTR,#I2CMSCR
                MOVX        @DPTR,A
                JMP         WAIT
WAIT:
                JB          BUSY,$                     ;Wait for the command to be sent
                RET

DELAY:
                MOV         R0,#0
                MOV         R1,#0
DELAY1:
```

```
                NOP
                NOP
                NOP
                NOP
                DJNZ        R1,DELAY1
                DJNZ        R0,DELAY1
                RET

MAIN:
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                MOV         P_SW2,#80H

                MOV         A,#11100000B                ;Set the I2C module as master mode
                MOV         DPTR,#I2CCFG
                MOVX        @DPTR,A
                MOV         A,#00000000B
                MOV         DPTR,#I2CMSST
                MOVX        @DPTR,A
                SETB        EA

                CALL        START                       ;Send start command
                MOV         A,#0A0H
                CALL        SENDDATA                    ;Send device address + write command
                CALL        RECVACK
                MOV         A,#000H                     ;Send storage address high byte
                CALL        SENDDATA
                CALL        RECVACK
                MOV         A,#000H                     ;Send storage address low byte
                CALL        SENDDATA
                CALL        RECVACK
                MOV         A,#12H                      ;Write test data 1
                CALL        SENDDATA
                CALL        RECVACK
                MOV         A,#78H                      ;Write test data 2
                CALL        SENDDATA
                CALL        RECVACK
                CALL        STOP                        ;Send stop command

                CALL        DELAY                       ;Waiting for the device to write data

                CALL        START                       ;Send start command
                MOV         A,#0A0H                     ;Send device address + write command
                CALL        SENDDATA
                CALL        RECVACK
                MOV         A,#000H                     ;Send storage address high byte
                CALL        SENDDATA
                CALL        RECVACK
                MOV         A,#000H                 ;Send storage address low byte
                CALL        SENDDATA
                CALL        RECVACK
                CALL        START                   ;Send start command
                MOV         A,#0A1H                 ;Send device address + read command
                CALL        SENDDATA
                CALL        RECVACK
                CALL        RECVDATA                ;Read data 1
                MOV         P0,A
                CALL        SENDACK
                CALL        RECVDATA                ;Read data 2
                MOV         P2,A
                CALL        SENDNAK
                CALL        STOP                    ;Send stop command

                JMP         $

                END
```

19.4.2 I^{2}C is Used to Access AT24C256 in Master Mode AT24C256 (Polling Mode)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P_SW2       =       0xba;

#define     I2CCFG              (*(unsigned char volatile xdata *)0xfe80)
#define     I2CMSCR             (*(unsigned char volatile xdata *)0xfe81)
#define     I2CMSST             (*(unsigned char volatile xdata *)0xfe82)
#define     I2CSLCR             (*(unsigned char volatile xdata *)0xfe83)
#define     I2CSLST             (*(unsigned char volatile xdata *)0xfe84)
#define     I2CSLADR            (*(unsigned char volatile xdata *)0xfe85)
#define     I2CTXD              (*(unsigned char volatile xdata *)0xfe86)
#define     I2CRXD              (*(unsigned char volatile xdata *)0xfe87)

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sbit        SDA         =       P1^4;
sbit        SCL         =       P1^5;
```

```
void Wait()
{
    while (!(I2CMSST & 0x40));
    I2CMSST &= ~0x40;
}

void Start()
{
    I2CMSCR = 0x01;                                     //Send START command
    Wait();
}

void SendData(char dat)
{
    I2CTXD = dat;                                       //Write data to the data buffer
    I2CMSCR = 0x02;                                     //Send a SEND command
    Wait();
}

void RecvACK()
{
    I2CMSCR = 0x03;                                     //Send read ACK command
    Wait();
}

char RecvData()
{
    I2CMSCR = 0x04;                                     //Send RECV command
    Wait();
    return I2CRXD;
}

void SendACK()
{
    I2CMSST = 0x00;                                     //Setup the ACK signal
    I2CMSCR = 0x05;                                     //Send ACK command
    Wait();
}

void SendNAK()
{
    I2CMSST = 0x01;                                     //Setup the NAK signal
    I2CMSCR = 0x05;                                     //Send ACK command
    Wait();
}

void Stop()
{
    I2CMSCR = 0x06;                                     //Send STOP command
    Wait();
}

void Delay()
{
    int i;

    for (i=0; i<3000; i++)
    {
```

```
                _nop_();
                _nop_();
                _nop_();
                _nop_();
        }
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P_SW2 = 0x80;

        I2CCFG = 0xe0;                          //Enable I2C master mode
        I2CMSST = 0x00;

        Start();                                //Send start command
        SendData(0xa0);                         //Send device address + write command
        RecvACK();
        SendData(0x00);                         //Send storage address high byte
        RecvACK();
        SendData(0x00);                         //Send storage address low byte
        RecvACK();
        SendData(0x12);                         //Write test data 1
        RecvACK();
        SendData(0x78);                         //Write test data 2
        RecvACK();
        Stop();                                 //Send stop command

        Delay();                                //Waiting for the device to write data

        Start();                                //Send start command
        SendData(0xa0);                         //Send device address + write command
        RecvACK();
        SendData(0x00);                         //Send storage address high byte
        RecvACK();
        SendData(0x00);                         //Send storage address low byte
        RecvACK();
        Start();                                //Send start command
        SendData(0xa1);                         //Send device address + read command
        RecvACK();
        P0 = RecvData();                        //Read data 1
        SendACK();
        P2 = RecvData();                        //Read data 2
        SendNAK();
        Stop();                                 //Send stop command

        P_SW2 = 0x00;
        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P_SW2       DATA        0BAH

I2CCFG      XDATA       0FE80H
I2CMSCR     XDATA       0FE81H
I2CMSST     XDATA       0FE82H
I2CSLCR     XDATA       0FE83H
I2CSLST     XDATA       0FE84H
I2CSLADR    XDATA       0FE85H
I2CTXD      XDATA       0FE86H
I2CRXD      XDATA       0FE87H

SDA         BIT         P1.4
SCL         BIT         P1.5

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN

            ORG         0100H
START:
            MOV         A,#00000001B                ;Send START command
            MOV         DPTR,#I2CMSCR
            MOVX        @DPTR,A
            JMP         WAIT
SENDDATA:
            MOV         DPTR,#I2CTXD                ;Write data to the data buffer
            MOVX        @DPTR,A
            MOV         A,#00000010B                ;Send a SEND command
            MOV         DPTR,#I2CMSCR
            MOVX        @DPTR,A
            JMP         WAIT
RECVACK:
            MOV         A,#00000011B                ;Send read ACK command
            MOV         DPTR,#I2CMSCR
            MOVX        @DPTR,A
            JMP         WAIT
RECVDATA:
            MOV         A,#00000100B                ;Send RECV command
            MOV         DPTR,#I2CMSCR
            MOVX        @DPTR,A
```

```
                CALL        WAIT
                MOV         DPTR,#I2CRXD                ;Read data from the data buffer
                MOVX        A,@DPTR
                RET
SENDACK:
                MOV         A,#00000000B                ;Setup the ACK signal
                MOV         DPTR,#I2CMSST
                MOVX        @DPTR,A
                MOV         A,#00000101B                ;Send ACK command
                MOV         DPTR,#I2CMSCR
                MOVX        @DPTR,A
                JMP         WAIT
SENDNAK:
                MOV         A,#00000001B                ;Setup the NAK signal
                MOV         DPTR,#I2CMSST
                MOVX        @DPTR,A
                MOV         A,#00000101B                ;Send ACK command
                MOV         DPTR,#I2CMSCR
                MOVX        @DPTR,A
                JMP         WAIT
STOP:
                MOV         A,#00000110B                ;Send STOP command
                MOV         DPTR,#I2CMSCR
                MOVX        @DPTR,A
                JMP         WAIT
WAIT:
                MOV         DPTR,#I2CMSST               ;Clear interrupt flag
                MOVX        A,@DPTR
                JNB         ACC.6,WAIT
                ANL         A,#NOT 40H
                MOVX        @DPTR,A
                RET

DELAY:
                MOV         R0,#0
                MOV         R1,#0
DELAY1:
                NOP
                NOP
                NOP
                NOP
                DJNZ        R1,DELAY1
                DJNZ        R0,DELAY1
                RET

MAIN:
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H
```

```
        MOV     P_SW2,#80H

        MOV     A,#11100000B            ;Set the I2C module as master mode
        MOV     DPTR,#I2CCFG
        MOVX    @DPTR,A
        MOV     A,#00000000B
        MOV     DPTR,#I2CMSST
        MOVX    @DPTR,A

        CALL    START                   ;Send start command
        MOV     A,#0A0H
        CALL    SENDDATA                ;Send device address + write command
        CALL    RECVACK
        MOV     A,#000H                 ;Send storage address high byte
        CALL    SENDDATA
        CALL    RECVACK
        MOV     A,#000H                 ;Send storage address low byte
        CALL    SENDDATA
        CALL    RECVACK
        MOV     A,#12H                  ;Write test data 1
        CALL    SENDDATA
        CALL    RECVACK
        MOV     A,#78H                  ;Write test data 2
        CALL    SENDDATA
        CALL    RECVACK
        CALL    STOP                    ;Send stop command

        CALL    DELAY                   ;Waiting for the device to write data

        CALL    START                   ;Send start command
        MOV     A,#0A0H                 ;Send device address + write command
        CALL    SENDDATA
        CALL    RECVACK
        MOV     A,#000H                 ;Send storage address high byte
        CALL    SENDDATA
        CALL    RECVACK
        MOV     A,#000H                 ;Send storage address low byte
        CALL    SENDDATA
        CALL    RECVACK
        CALL    START                   ;Send start command
        MOV     A,#0A1H                 ;Send device address + read command
        CALL    SENDDATA
        CALL    RECVACK
        CALL    RECVDATA                ;Read data 1
        MOV     P0,A
        CALL    SENDACK
        CALL    RECVDATA                ;Read data 2
        MOV     P2,A
        CALL    SENDNAK
        CALL    STOP                    ;Send stop command

        JMP     $

        END
```

19.4.3 I^{2}C is Used to Access PCF8563 in Master Mode

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P_SW2           =       0xba;

#define     I2CCFG                  (*(unsigned char volatile xdata *)0xfe80)
#define     I2CMSCR                 (*(unsigned char volatile xdata *)0xfe81)
#define     I2CMSST                 (*(unsigned char volatile xdata *)0xfe82)
#define     I2CSLCR                 (*(unsigned char volatile xdata *)0xfe83)
#define     I2CSLST                 (*(unsigned char volatile xdata *)0xfe84)
#define     I2CSLADR                (*(unsigned char volatile xdata *)0xfe85)
#define     I2CTXD                  (*(unsigned char volatile xdata *)0xfe86)
#define     I2CRXD                  (*(unsigned char volatile xdata *)0xfe87)

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

sbit        SDA             =       P1^4;
sbit        SCL             =       P1^5;

void Wait()
{
    while (!(I2CMSST & 0x40));
    I2CMSST &= ~0x40;
}

void Start()
{
    I2CMSCR = 0x01;                                     //Send START command
    Wait();
}

void SendData(char dat)
{
    I2CTXD = dat;                                       //Write data to the data buffer
    I2CMSCR = 0x02;                                     //Send a SEND command
    Wait();
}

void RecvACK()
{
    I2CMSCR = 0x03;                                     //Send read ACK command
    Wait();
```

```
}

char RecvData()
{
        I2CMSCR = 0x04;                                         //Send RECV command
        Wait();
        return I2CRXD;
}

void SendACK()
{
        I2CMSST = 0x00;                                         //Setup the ACK signal
        I2CMSCR = 0x05;                                         //Send ACK command
        Wait();
}

void SendNAK()
{
        I2CMSST = 0x01;                                         //Setup the NAK signal
        I2CMSCR = 0x05;                                         //Send ACK command
        Wait();
}

void Stop()
{
        I2CMSCR = 0x06;                                         //Send STOP command
        Wait();
}

void Delay()
{
        int i;

        for (i=0; i<3000; i++)
        {
                _nop_();
                _nop_();
                _nop_();
                _nop_();
        }
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P_SW2 = 0x80;
        I2CCFG = 0xe0;                                              //Enable I2C master mode
        I2CMSST = 0x00;

        Start();                                                    //Send start command
        SendData(0xa2);                                             //Send device address + write command
        RecvACK();
        SendData(0x02);                                             //Send storage address
        RecvACK();
        SendData(0x00);                                             //Set second value
        RecvACK();
        SendData(0x00);                                             //Set minute value
        RecvACK();
        SendData(0x12);                                             //Set hour value
        RecvACK();
        Stop();                                                     //Send stop command

        while (1)
        {
                Start();                                            //Send start command
                SendData(0xa2);                                     //Send device address + write command
                RecvACK();
                SendData(0x02);                                     //Send storage address
                RecvACK();
                Start();                                            //Send start command
                SendData(0xa3);                                     //Send device address + read command
                RecvACK();
                P0 = RecvData();                                    //Read second value
                SendACK();
                P2 = RecvData();                                    //Read minute value
                SendACK();
                P3 = RecvData();                                    //Read hour value
                SendNAK();
                Stop();                                             //Send stop command

                Delay();
        }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P_SW2           DATA            0BAH

I2CCFG          XDATA           0FE80H
I2CMSCR         XDATA           0FE81H
I2CMSST         XDATA           0FE82H
I2CSLCR         XDATA           0FE83H
I2CSLST         XDATA           0FE84H
I2CSLADR        XDATA           0FE85H
I2CTXD          XDATA           0FE86H
I2CRXD          XDATA           0FE87H

SDA             BIT             P1.4
SCL             BIT             P1.5

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
```

```
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN

            ORG         0100H
START:
            MOV         A,#00000001B                ;Send START command
            MOV         DPTR,#I2CMSCR
            MOVX        @DPTR,A
            JMP         WAIT
SENDDATA:
            MOV         DPTR,#I2CTXD                ;Write data to the data buffer
            MOVX        @DPTR,A
            MOV         A,#00000010B                ;Send a SEND command
            MOV         DPTR,#I2CMSCR
            MOVX        @DPTR,A
            JMP         WAIT
RECVACK:
            MOV         A,#00000011B                ;Send read ACK command
            MOV         DPTR,#I2CMSCR
            MOVX        @DPTR,A
            JMP         WAIT
RECVDATA:
            MOV         A,#00000100B                ;Send RECV command
            MOV         DPTR,#I2CMSCR
            MOVX        @DPTR,A
            CALL        WAIT
            MOV         DPTR,#I2CRXD                ;Read data from the data buffer
            MOVX        A,@DPTR
            RET
SENDACK:
            MOV         A,#00000000B                ;Setup the ACK signal
            MOV         DPTR,#I2CMSST
            MOVX        @DPTR,A
            MOV         A,#00000101B                ;Send ACK command
            MOV         DPTR,#I2CMSCR
            MOVX        @DPTR,A
            JMP         WAIT
SENDNAK:
            MOV         A,#00000001B                ;Setup the NAK signal
            MOV         DPTR,#I2CMSST
            MOVX        @DPTR,A
            MOV         A,#00000101B                ;Send ACK command
            MOV         DPTR,#I2CMSCR
            MOVX        @DPTR,A
            JMP         WAIT
STOP:
            MOV         A,#00000110B                ;Send STOP command
            MOV         DPTR,#I2CMSCR
            MOVX        @DPTR,A
            JMP         WAIT
```

```
WAIT:
            MOV         DPTR,#I2CMSST               ;Clear interrupt flag
            MOVX        A,@DPTR
            JNB         ACC.6,WAIT
            ANL         A,#NOT 40H
            MOVX        @DPTR,A
            RET

DELAY:
            MOV         R0,#0
            MOV         R1,#0
DELAY1:
            NOP
            NOP
            NOP
            NOP
            DJNZ        R1,DELAY1
            DJNZ        R0,DELAY1
            RET

MAIN:
            MOV         SP, #5FH
            MOV         P0M0, #00H
            MOV         P0M1, #00H
            MOV         P1M0, #00H
            MOV         P1M1, #00H
            MOV         P2M0, #00H
            MOV         P2M1, #00H
            MOV         P3M0, #00H
            MOV         P3M1, #00H
            MOV         P4M0, #00H
            MOV         P4M1, #00H
            MOV         P5M0, #00H
            MOV         P5M1, #00H

            MOV         P_SW2,#80H

            MOV         A,#11100000B                ;Set the I2C module as master mode
            MOV         DPTR,#I2CCFG
            MOVX        @DPTR,A
            MOV         A,#00000000B
            MOV         DPTR,#I2CMSST
            MOVX        @DPTR,A

            CALL        START                       ;Send start command
            MOV         A,#0A2H
            CALL        SENDDATA                    ;Send device address + write command
            CALL        RECVACK
            MOV         A,#002H                     ;Send storage address
            CALL        SENDDATA
            CALL        RECVACK
            MOV         A,#00H                      ;Set second value
            CALL        SENDDATA
            CALL        RECVACK
            MOV         A,#00H                      ;Set minute value
            CALL        SENDDATA
            CALL        RECVACK
            MOV         A,#12H                      ;Set hour value
            CALL        SENDDATA
```

```
                CALL        RECVACK
                CALL        STOP                    ;Send stop command
LOOP:
                CALL        START                   ;Send start command
                MOV         A,#0A2H                 ;Send device address + write command
                CALL        SENDDATA
                CALL        RECVACK
                MOV         A,#002H                 ;Send storage address
                CALL        SENDDATA
                CALL        RECVACK
                CALL        START                   ;Send start command
                MOV         A,#0A3H                 ;Send device address + read command
                CALL        SENDDATA
                CALL        RECVACK
                CALL        RECVDATA                ;Read second value
                MOV         P0,A
                CALL        SENDACK
                CALL        RECVDATA                ;Read minute value
                MOV         P2,A
                CALL        SENDACK
                CALL        RECVDATA                ;Read hour value
                MOV         P3,A
                CALL        SENDNAK
                CALL        STOP                    ;Send stop command

                CALL        DELAY

                JMP         LOOP

                END
```

19.4.4 I^{2}C Slave Mode (Polling Mode)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P_SW2           =       0xba;

#define     I2CCFG                  (*(unsigned char volatile xdata *)0xfe80)
#define     I2CMSCR                 (*(unsigned char volatile xdata *)0xfe81)
#define     I2CMSST                 (*(unsigned char volatile xdata *)0xfe82)
#define     I2CSLCR                 (*(unsigned char volatile xdata *)0xfe83)
#define     I2CSLST                 (*(unsigned char volatile xdata *)0xfe84)
#define     I2CSLADR                (*(unsigned char volatile xdata *)0xfe85)
#define     I2CTXD                  (*(unsigned char volatile xdata *)0xfe86)
#define     I2CRXD                  (*(unsigned char volatile xdata *)0xfe87)

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

sbit        SDA             =       P1^4;
sbit        SCL             =       P1^5;

bit         isda;                                           //Device address flag
bit         isma;                                           //Storage address flag
unsigned char               addr;
unsigned char pdata         buffer[256];

void I2C_Isr() interrupt 24
{
        _push_(P_SW2);
        P_SW2 |= 0x80;

        if (I2CSLST & 0x40)
        {
                I2CSLST &= ~0x40;                           //Handle the START event
        }
        else if (I2CSLST & 0x20)
        {
                I2CSLST &= ~0x20;                           //Handle the RECV event
                if (isda)
                {
                        isda = 0;                           //Handle the RECV event (RECV DEVICE ADDR)
                }
                else if (isma)
                {
                        isma = 0;                           //Handle the RECV event (RECV MEMORY ADDR)
                        addr = I2CRXD;
                        I2CTXD = buffer[addr];
                }
                else
                {
                        buffer[addr++] = I2CRXD;            //Handle the RECV event (RECV DATA)
                }
        }
        else if (I2CSLST & 0x10)
        {
                I2CSLST &= ~0x10;                           //Handle the SEND event
                if (I2CSLST & 0x02)
                {
                        I2CTXD = 0xff;                      //Stop receiving data when receiving NAK
                }
                else
                {
                        I2CTXD = buffer[++addr];            //Continue reading data when receiving ACK
                }
        }
        else if (I2CSLST & 0x08)
        {
                I2CSLST &= ~0x08;                           //Handle the STOP event
                isda = 1;
                isma = 1;
        }
        _pop_(P_SW2);
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P_SW2 = 0x80;

        I2CCFG = 0x81;                                      //Enable I2C slave mode
        I2CSLADR = 0x5a;                                    //Set the slave device address to 5A
                                                            //That is, I2CSLADR[7:1]=010_1101B, MA=0B.
                                                            //Since MA is 0, the device address sent by the host must be
                                                            // the same as I2CSLADR[7:1] to access this I2C slave
device.
                                                            //If the host needs to write data, it will send
5AH(0101_1010B)
        I2CSLST = 0x00;
        I2CSLCR = 0x78;                                     //Enable interrupt of slave mode
        EA = 1;

        isda = 1;                                           //User variable initialization
        isma = 1;
        addr = 0;
        I2CTXD = buffer[addr];

        while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P_SW2           DATA            0BAH

I2CCFG          XDATA           0FE80H
I2CMSCR         XDATA           0FE81H
I2CMSST         XDATA           0FE82H
I2CSLCR         XDATA           0FE83H
I2CSLST         XDATA           0FE84H
I2CSLADR        XDATA           0FE85H
I2CTXD          XDATA           0FE86H
I2CRXD          XDATA           0FE87H

SDA             BIT             P1.4
SCL             BIT             P1.5
ISDA            BIT             20H.0                   ;Device address flag
ISMA            BIT             20H.1                   ;Storage address flag
```

```
ADDR            DATA            21H

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             00C3H
                LJMP            I2CISR

                ORG             0100H
I2CISR:
                PUSH            ACC
                PUSH            PSW
                PUSH            DPL
                PUSH            DPH
                MOV             DPTR,#I2CSLST               ;Detect slave status
                MOVX            A,@DPTR
                JB              ACC.6,STARTIF
                JB              ACC.5,RXIF
                JB              ACC.4,TXIF
                JB              ACC.3,STOPIF
ISREXIT:
                POP             DPH
                POP             DPL
                POP             PSW
                POP             ACC
                RETI
STARTIF:
                ANL             A,#NOT 40H                  ;Handle the START event
                MOVX            @DPTR,A
                JMP             ISREXIT
RXIF:
                ANL             A,#NOT 20H                  ;Handle the RECV event
                MOVX            @DPTR,A
                MOV             DPTR,#I2CRXD
                MOVX            A,@DPTR
                JBC             ISDA,RXDA
                JBC             ISMA,RXMA
                MOV             R0,ADDR                     ;Handle the RECV event (RECV DATA)
                MOVX            @R0,A
                INC             ADDR
                JMP             ISREXIT
RXDA:
                JMP             ISREXIT                     ;Handle the RECV event (RECV DEVICE ADDR)
RXMA:
                MOV             ADDR,A                      ;Handle the RECV event (RECV MEMORY ADDR)
                MOV             R0,A
                MOVX        A,@R0
                MOV         DPTR,#I2CTXD
                MOVX        @DPTR,A
                JMP         ISREXIT
TXIF:
                ANL         A,#NOT 10H                  ;Handle the SEND event
                MOVX        @DPTR,A
                JB          ACC.1,RXNAK
                INC         ADDR
                MOV         R0,ADDR
                MOVX        A,@R0
                MOV         DPTR,#I2CTXD
                MOVX        @DPTR,A
                JMP         ISREXIT
RXNAK:
                MOVX        A,#0FFH
                MOV         DPTR,#I2CTXD
                MOVX        @DPTR,A
                JMP         ISREXIT
STOPIF:
                ANL         A,#NOT 08H                  ;Handle the STOP event
                MOVX        @DPTR,A
                SETB        ISDA
                SETB        ISMA
                JMP         ISREXIT

MAIN:
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                MOV         P_SW2,#80H

                MOV         A,#10000001B                ;Enable I2C slave mode
                MOV         DPTR,#I2CCFG
                MOVX        @DPTR,A
                MOV         A,#01011010B                ;Set the slave device address to 5A
                                                        ;That is, I2CSLADR[7:1]=010_1101B, MA=0B.
                                                        ;Since MA is 0, the device address sent by the host must be
                                                        ; the same as I2CSLADR[7:1] to access this I2C slave
device.
                                                        ;If the host needs to write data, it will send
5AH(0101_1010B)

                MOV         DPTR,#I2CSLADR
                MOVX        @DPTR,A
                MOV         A,#00000000B
                MOV         DPTR,#I2CSLST
                MOVX        @DPTR,A
                MOV         A,#01111000B            ;Enable interrupt of slave mode
                MOV         DPTR,#I2CSLCR
                MOVX        @DPTR,A

                SETB        ISDA                    ;User variable initialization
                SETB        ISMA
                CLR         A
                MOV         ADDR,A
                MOV         R0,A
                MOVX        A,@R0
                MOV         DPTR,#I2CTXD
                MOVX        @DPTR,A

                SETB        EA

                SJMP        $

                END
```

19.4.5 I^{2}C Slave Mode (Polling Mode)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P_SW2       =       0xba;

#define     I2CCFG              (*(unsigned char volatile xdata *)0xfe80)
#define     I2CMSCR             (*(unsigned char volatile xdata *)0xfe81)
#define     I2CMSST             (*(unsigned char volatile xdata *)0xfe82)
#define     I2CSLCR             (*(unsigned char volatile xdata *)0xfe83)
#define     I2CSLST             (*(unsigned char volatile xdata *)0xfe84)
#define     I2CSLADR            (*(unsigned char volatile xdata *)0xfe85)
#define     I2CTXD              (*(unsigned char volatile xdata *)0xfe86)
#define     I2CRXD              (*(unsigned char volatile xdata *)0xfe87)

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sbit        SDA         =       P1^4;
sbit        SCL         =       P1^5;

bit         isda;                                   //Device address flag
bit         isma;                                   //Storage address flag
unsigned char           addr;
unsigned char pdata         buffer[256];

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P_SW2 = 0x80;

        I2CCFG = 0x81;//Enable I2C slave mode
        I2CSLADR = 0x5a;                            //Set the slave device address to 5A
                                                    //That is, I2CSLADR[7:1]=010_1101B, MA=0B.
                                                    //Since MA is 0, the device address sent by the host must be
                                                    // the same as I2CSLADR[7:1] to access this I2C slave
device.
                                                    //If the host needs to write data, it will send
5AH(0101_1010B)
                                                    //If the host needs to read data, it will send 5BH
(0101_1011B)
        I2CSLST = 0x00;
        I2CSLCR = 0x00;                             //Disable interrupt of slave mode

        isda = 1;                                   //User variable initialization
        isma = 1;
        addr = 0;
        I2CTXD = buffer[addr];

        while (1)
        {
                if (I2CSLST & 0x40)
                {
                        I2CSLST &= ~0x40;           //Handle the START event
                }
                else if (I2CSLST & 0x20)
                {
                        I2CSLST &= ~0x20;           //Handle the RECV event
                        if (isda)
                        {
                                isda = 0;           //Handle the RECV event (RECV DEVICE ADDR)
                        }
                        else if (isma)
                        {
                                isma = 0;           //Handle the RECV event (RECV MEMORY ADDR)
                                addr = I2CRXD;
                                I2CTXD = buffer[addr];
                        }
                        else
                        {
                                buffer[addr++] = I2CRXD;    //Handle the RECV event (RECV DATA)
                        }
                }
                else if (I2CSLST & 0x10)
                {
                        I2CSLST &= ~0x10;                               //Handle the SEND event
                        if (I2CSLST & 0x02)
                        {
                                I2CTXD = 0xff;                          //Stop receiving data when receiving NAK
                        }
                        else
                        {
                                I2CTXD = buffer[++addr];                //Continue reading data when receiving ACK
                        }
                }
                else if (I2CSLST & 0x08)
                {
                        I2CSLST &= ~0x08;                               //Handle the STOP event
                        isda = 1;
                        isma = 1;
                }
        }
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P_SW2           DATA            0BAH

I2CCFG          XDATA           0FE80H
I2CMSCR         XDATA           0FE81H
I2CMSST         XDATA           0FE82H
I2CSLCR         XDATA           0FE83H
I2CSLST         XDATA           0FE84H
I2CSLADR        XDATA           0FE85H
I2CTXD          XDATA           0FE86H
I2CRXD          XDATA           0FE87H

SDA             BIT             P1.4
SCL             BIT             P1.5
ISDA            BIT             20H.0                           ;Device address flag
ISMA            BIT             20H.1                           ;Storage address flag

ADDR            DATA            21H

P0M1            DATA            093H
P0M0            DATA            094H
P1M1            DATA            091H
P1M0            DATA            092H
P2M1            DATA            095H
P2M0            DATA            096H
P3M1            DATA            0B1H
P3M0            DATA            0B2H
P4M1            DATA            0B3H
P4M0            DATA            0B4H
P5M1            DATA            0C9H
P5M0            DATA            0CAH

                ORG             0000H
                LJMP            MAIN
                ORG             0100H
MAIN:
                MOV             SP, #5FH
                MOV             P0M0, #00H
                MOV             P0M1, #00H
                MOV             P1M0, #00H
                MOV             P1M1, #00H
                MOV             P2M0, #00H
                MOV             P2M1, #00H
                MOV             P3M0, #00H
                MOV             P3M1, #00H
                MOV             P4M0, #00H
                MOV             P4M1, #00H
                MOV             P5M0, #00H
                MOV             P5M1, #00H

                MOV             P_SW2,#80H

                MOV             A,#10000001B                ;Enable I2C slave mode
                MOV             DPTR,#I2CCFG
                MOVX            @DPTR,A
                MOV             A,#01011010B                ;Set the slave device address to 5A
                                                            ;/That is, I2CSLADR[7:1]=010_1101B, MA=0B.
                                                            ;Since MA is 0, the device address sent by the host must be
                                                            ; the same as I2CSLADR[7:1] to access this I2C slave
device.
                                                            ;If the host needs to write data, it will send
5AH(0101_1010B)
                                                            ;If the host needs to read data, it will send 5BH
(0101_1011B)

                MOV             DPTR,#I2CSLADR
                MOVX            @DPTR,A
                MOV             A,#00000000B
                MOV             DPTR,#I2CSLST
                MOVX            @DPTR,A
                MOV             A,#00000000B                ;Disable interrupt of slave mode
                MOV             DPTR,#I2CSLCR
                MOVX            @DPTR,A

                SETB            ISDA                        ;User variable initialization
                SETB            ISMA
                CLR             A
                MOV             ADDR,A
                MOV             R0,A
                MOVX            A,@R0
                MOV             DPTR,#I2CTXD
                MOVX            @DPTR,A

LOOP:
                MOV             DPTR,#I2CSLST               ;Detect slave status
                MOVX            A,@DPTR
                JB              ACC.6,STARTIF
                JB              ACC.5,RXIF
                JB              ACC.4,TXIF
                JB              ACC.3,STOPIF
                JMP             LOOP
STARTIF:
```

```
                ANL         A,#NOT 40H                  ;Handle the START event
                MOVX        @DPTR,A
                JMP         LOOP
RXIF:
                ANL         A,#NOT 20H                  ;Handle the RECV event
                MOVX        @DPTR,A
                MOV         DPTR,#I2CRXD
                MOVX        A,@DPTR
                JBC         ISDA,RXDA
                JBC         ISMA,RXMA
                MOV         R0,ADDR                     ;Handle the RECV event (RECV DATA)
                MOVX        @R0,A
                INC         ADDR
                JMP         LOOP
RXDA:
                JMP         LOOP                        ;Handle the RECV event (RECV DEVICE ADDR)
RXMA:
                MOV         ADDR,A                      ;Handle the RECV event (RECV MEMORY ADDR)
                MOV         R0,A
                MOVX        A,@R0
                MOV         DPTR,#I2CTXD
                MOVX        @DPTR,A
                JMP         LOOP
TXIF:
                ANL         A,#NOT 10H                  ;Handle the SEND event
                MOVX        @DPTR,A
                JB          ACC.1,RXNAK
                INC         ADDR
                MOV         R0,ADDR
                MOVX        A,@R0
                MOV         DPTR,#I2CTXD
                MOVX        @DPTR,A
                JMP         LOOP
RXNAK:
                MOVX        A,#0FFH
                MOV         DPTR,#I2CTXD
                MOVX        @DPTR,A
                JMP         LOOP
STOPIF:
                ANL         A,#NOT 08H                  ;Handle the STOP event
                MOVX        @DPTR,A
                SETB        ISDA
                SETB        ISMA
                JMP         LOOP

                END
```

19.4.6 Master Codes for testing I^{2}C Slave Mode

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P_SW2           =       0xba;
```

```
#define     I2CCFG              (*(unsigned char volatile xdata *)0xfe80)
#define     I2CMSCR             (*(unsigned char volatile xdata *)0xfe81)
#define     I2CMSST             (*(unsigned char volatile xdata *)0xfe82)
#define     I2CSLCR             (*(unsigned char volatile xdata *)0xfe83)
#define     I2CSLST             (*(unsigned char volatile xdata *)0xfe84)
#define     I2CSLADR            (*(unsigned char volatile xdata *)0xfe85)
#define     I2CTXD              (*(unsigned char volatile xdata *)0xfe86)
#define     I2CRXD              (*(unsigned char volatile xdata *)0xfe87)

sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xb1;
sfr         P3M0        =       0xb2;
sfr         P4M1        =       0xb3;
sfr         P4M0        =       0xb4;
sfr         P5M1        =       0xc9;
sfr         P5M0        =       0xca;

sbit        SDA         =       P1^4;
sbit        SCL         =       P1^5;

void Wait()
{
    while (!(I2CMSST & 0x40));
    I2CMSST &= ~0x40;
}

void Start()
{
    I2CMSCR = 0x01;                             //Send START command
    Wait();
}

void SendData(char dat)
{
    I2CTXD = dat;                               //Write data to the data buffer
    I2CMSCR = 0x02;                             //Send a SEND command
    Wait();
}

void RecvACK()
{
    I2CMSCR = 0x03;                             //Send read ACK command
    Wait();
}

char RecvData()
{
    I2CMSCR = 0x04;                             //Send RECV command
    Wait();
    return I2CRXD;
}

void SendACK()
{
        I2CMSST = 0x00;                                     //Setup the ACK signal
        I2CMSCR = 0x05;                                     //Send ACK command
        Wait();
}
void SendNAK()
{
        I2CMSST = 0x01;                                     //Setup the NAK signal
        I2CMSCR = 0x05;                                     //Send ACK command
        Wait();
}
void Stop()
{
        I2CMSCR = 0x06;                                     //Send STOP command
        Wait();
}
void Delay()
{
        int i;

        for (i=0; i<3000; i++)
        {
                _nop_();
                _nop_();
                _nop_();
                _nop_();
        }
}

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P_SW2 = 0x80;

        I2CCFG = 0xe0;                                      //Enable I2C master mode
        I2CMSST = 0x00;

        Start();                                            //Send start command
        SendData(0x5a);                                     //Send device address (010_1101B) + write command (0b)
        RecvACK();
        SendData(0x00);                                     //Send storage address
        RecvACK();
        SendData(0x12);                                     //Write test data 1
        RecvACK();
        SendData(0x78);                                     //Write test data 2
        RecvACK();
        Stop();                                             //Send stop command
```

```
    Start();                                    //Send start command
    SendData(0x5a);                             //Send device address (010_1101B) + write command (0b)
    RecvACK();
    SendData(0x00);                             //Send storage address high byte
    RecvACK();
    Start();                                    //Send start command
    SendData(0x5b);                             //Send device address (010_1101B) + read command (1b)
    RecvACK();
    P0 = RecvData();                            //Read data 1
    SendACK();
    P2 = RecvData();                            //Read data 2
    SendNAK();
    Stop();                                     //Send stop command

    P_SW2 = 0x00;

    while (1);
}
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P_SW2       DATA        0BAH

I2CCFG      XDATA       0FE80H
I2CMSCR     XDATA       0FE81H
I2CMSST     XDATA       0FE82H
I2CSLCR     XDATA       0FE83H
I2CSLST     XDATA       0FE84H
I2CSLADR    XDATA       0FE85H
I2CTXD      XDATA       0FE86H
I2CRXD      XDATA       0FE87H

SDA         BIT         P1.4
SCL         BIT         P1.5

P0M1        DATA        093H
P0M0        DATA        094H
P1M1        DATA        091H
P1M0        DATA        092H
P2M1        DATA        095H
P2M0        DATA        096H
P3M1        DATA        0B1H
P3M0        DATA        0B2H
P4M1        DATA        0B3H
P4M0        DATA        0B4H
P5M1        DATA        0C9H
P5M0        DATA        0CAH

            ORG         0000H
            LJMP        MAIN

            ORG         0100H
START:
            MOV         A,#00000001B            ;Send START command
            MOV         DPTR,#I2CMSCR
            MOVX        @DPTR,A
            JMP         WAIT
SENDDATA:
```

```
                MOV         DPTR,#I2CTXD                ;Write data to the data buffer
                MOVX        @DPTR,A
                MOV         A,#00000010B                ;Send a SEND command
                MOV         DPTR,#I2CMSCR
                MOVX        @DPTR,A
                JMP         WAIT
RECVACK:
                MOV         A,#00000011B                ;Send read ACK command
                MOV         DPTR,#I2CMSCR
                MOVX        @DPTR,A
                JMP         WAIT
RECVDATA:
                MOV         A,#00000100B                ;Send RECV command
                MOV         DPTR,#I2CMSCR
                MOVX        @DPTR,A
                CALL        WAIT
                MOV         DPTR,#I2CRXD                ;Read data from the data buffer
                MOVX        A,@DPTR
                RET
SENDACK:
                MOV         A,#00000000B                ;Setup the ACK signal
                MOV         DPTR,#I2CMSST
                MOVX        @DPTR,A
                MOV         A,#00000101B                ;Send ACK command
                MOV         DPTR,#I2CMSCR
                MOVX        @DPTR,A
                JMP         WAIT
SENDNAK:
                MOV         A,#00000001B                ;Setup the NAK signal
                MOV         DPTR,#I2CMSST
                MOVX        @DPTR,A
                MOV         A,#00000101B                ;Send ACK command
                MOV         DPTR,#I2CMSCR
                MOVX        @DPTR,A
                JMP         WAIT
STOP:
                MOV         A,#00000110B                ;Send STOP command
                MOV         DPTR,#I2CMSCR
                MOVX        @DPTR,A
                JMP         WAIT
WAIT:
                MOV         DPTR,#I2CMSST               ;Clear interrupt flag
                MOVX        A,@DPTR
                JNB         ACC.6,WAIT
                ANL         A,#NOT 40H
                MOVX        @DPTR,A
                RET

DELAY:
                MOV         R0,#0
                MOV         R1,#0
DELAY1:
                NOP
                NOP
                NOP
                NOP
                DJNZ        R1,DELAY1
                DJNZ        R0,DELAY1
                RET
```

```
MAIN:
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                MOV         P_SW2,#80H

                MOV         A,#11100000B                ;Set the I2C module as master mode
                MOV         DPTR,#I2CCFG
                MOVX        @DPTR,A
                MOV         A,#00000000B
                MOV         DPTR,#I2CMSST
                MOVX        @DPTR,A

                CALL        START                       ;Send start command
                MOV         A,#5AH                      ;Slave address is 5A
                CALL        SENDDATA                    ;Send device address (010_1101B) + write command (0b)
                CALL        RECVACK
                MOV         A,#000H                     ;Send storage address
                CALL        SENDDATA
                CALL        RECVACK
                MOV         A,#12H                      ;Write test data 1
                CALL        SENDDATA
                CALL        RECVACK
                MOV         A,#78H                      ;Write test data 2
                CALL        SENDDATA
                CALL        RECVACK
                CALL        STOP                        ;Send stop command

                CALL        DELAY                       ;Waiting for the device to write data

                CALL        START                       ;Send start command
                MOV         A,#5AH                      ;Send device address (010_1101B) + write command (0b)
                CALL        SENDDATA
                CALL        RECVACK
                MOV         A,#000H                     ;Send storage address
                CALL        SENDDATA
                CALL        RECVACK
                CALL        START                       ;Send start command
                MOV         A,#5BH                      ;Send device address (010_1101B) + read command (1b)
                CALL        SENDDATA
                CALL        RECVACK
                CALL        RECVDATA                    ;Read data 1
                MOV         P0,A
                CALL        SENDACK
                CALL        RECVDATA                    ;Read data 2
                MOV         P2,A
                CALL        SENDNAK
```

```
        CALL        STOP                    ;Send stop command
        JMP         $
        END
```

20 16-bit advanced PWM timer, support quadrature encoder

Product line	Advanced PWM
STC8H1K08 family	●
STC8H1K28 family	●
STC8H3K64S4 family	●
STC8H3K64S2 family	●
STC8H8K64U family	●
STC8H2K64T family	●
STC8H4K64TLR family	●
STC8H4K64TLCD family	●
STC8H4K64LCD family	●

The STC8H series of microcomputers integrate 8-channel 16-bit advanced PWM timers, which are divided into two groups of PWMs with different periods, named PWMA and PWMB (the previous data sheet used to be named PWM1 and PWM2, but it is easy to match the chip pin name for confusion, it is changed to PWMA and PWMB), which can be set separately. The first group of PWM/PWMA can be configured as 4 groups of complementary/symmetrical/dead-zone controlled PWM or capture external signals, and the second group of PWM/PWMB can be configured as 4 channels of PWM output or capture external signals.

The clock frequency of the first set of PWM/PWMA can be the system clock divided by the registers PWMA_PSCRH and PWMA_PSCRL. The frequency division value can be any value between 1~65535. The clock frequency of the second group of PWM/PWMB can be the system clock divided by the PWMB_PSCRH PWMB_PSCRL register, and the divided value can be any value between 1 and 65535. The clock frequencies of the two sets of PWM can be set independently.

The first set of PWM timers/PWMA has 4 channels (PWM1P/PWM1N, PWM2P/PWM2N, PWM3P/PWM3N, PWM4P/PWM4N), each channel can independently realize PWM output (complementary symmetrical PWM output with dead zone can be set), capture and compare functions; the second group of PWM timer/PWMB has 4 channels (PWM5, PWM6, PWM7, PWM8), and each channel can also independently implement PWM output, capture and compare functions. The only difference between the two sets of PWM timers is that the first set can output complementary symmetrical PWM with dead zone, while the second set can only output single-ended PWM, and other functions are exactly the same. The following description of advanced PWM timers only takes the first group as an example.

When using the first set of PWM timers to output PWM waveforms, the PWM1P/PWM2P/PWM3P/PWM4P output can be individually enabled, or the PWM1N/PWM2N/PWM3N/PWM4N output can be individually enabled. For example: if PWM1P output is enabled separately, PWM1N can no longer output independently, unless PWM1P and PWM1N form a set of complementary symmetrical output. The 4 outputs of PWMA can be set independently, for example: PWM1P and PWM2N output can be individually enabled, or PWM2N and PWM3N output can be individually enabled. If you need to use the first set of PWM timers for capture function or pulse width measurement, the input signal can only be input from the positive end of each channel, that is, only PWM1P/PWM2P/PWM3P/PWM4P have the capture function and pulse width measurement function.

When the two sets of advanced PWM timers capture external signals, you can select rising edge capture or falling edge capture. If you need to capture the rising and falling edges at the same time, you can connect the input signal to two PWMs at the same time, and enable one of them to capture the rising edge and the other to capture the falling edge. What's more powerful is that when the external input signal is connected to two PWMs at the same time, the period value and duty ratio value of the signal can be captured at the same time.

Comparison of STC three hardware PWMs:

Compatible with traditional 8051 PCA/CCP/PWM: It can output PWM waveforms, capture external input signals and output high-speed pulses. It can output 6-bit/7-bit/8-bit/10-bit PWM waveform externally. The frequency of the 6-bit PWM waveform is the PCA module clock source frequency/64; the frequency of the 7-bit PWM waveform is the PCA module clock source frequency/128; 8-bit The frequency of the PWM waveform is the PCA module clock source frequency/256; the frequency of the 10-bit PWM waveform is the PCA module clock source frequency/1024. Capture the external input signal, you can capture the rising edge, the falling edge, or both the rising edge and the falling edge.

15-bit enhanced PWM of STC8G series: It can only output PWM waveform externally, without input capture function. The frequency and duty cycle of external output PWM can be set arbitrarily. Through software intervention, multi-channel complementary/symmetrical/with dead zone PWM waveform can be realized. It has external abnormality detection function and real-time trigger ADC conversion function.

The 16-bit advanced PWM timer of the STC8H series: It is the PWM with the strongest STC function at present, and can output PWM waveforms of any frequency and any duty cycle. It can output complementary/symmetrical/with dead zone PWM waveform without software intervention. It can capture the external input signal, can capture the rising edge, the falling edge or the rising edge and the falling edge at the same time, when measuring the external waveform, the period value and the duty ratio value of the waveform can be measured at the same time. There are quadrature encoding function, external anomaly detection function and real-time trigger ADC conversion function.

In the following description, PWMA represents the first group of PWM timers, and PWMB represents the second group of PWM timers

The first group of advanced PWM timer/PWMB internal signal description

TI1: External clock input signal 1 (PWM1P pin signal or the signal after the exclusive OR of PWM1P/PWM2P/PWM3P)
TI1F: TI1 signal after IC1F digital filtering
TI1FP: TI1F signal after CC1P/CC2P edge detector
TI1F_ED: TI1F edge signal
TI1FP1: TI1F signal after CC1P edge detector
TI1FP2: TI1F signal after CC2P edge detector
IC1: Capture input signal of channel 1 selected by CC1S
OC1REF: output reference waveform of channel 1 (middle waveform)
OC1: Main output signal of channel 1 (OC1REF signal after CC1P polarity processing)
OC1N: Complementary output signal of channel 1 (OC1REF signal after CC1NP polarity processing)

TI2: External clock input signal 2 (PWM2P pin signal)
TI2F: TI2 signal after IC2F digital filtering
TI2F_ED: TI2F edge signal
TI2FP: TI2F signal after CC1P/CC2P edge detector
TI2FP1: TI2F signal after CC1P edge detector
TI2FP2: TI2F signal after CC2P edge detector
IC2: Capture input signal of channel 2 selected by CC2S
OC2REF: output reference waveform of channel 2 (middle waveform)
OC2: Main output signal of channel 2 (OC2REF signal after CC2P polarity processing)
OC2N: Complementary output signal of channel 2 (OC2REF signal after CC2NP polarity processing)

TI3: External clock input signal 3 (PWM3P pin signal)
TI3F: TI3 signal after IC3F digital filtering
TI3F_ED: TI3F edge signal
TI3FP: TI3F signal after CC3P/CC4P edge detector
TI3FP3: TI3F signal after CC3P edge detector
TI3FP4: TI3F signal after CC4P edge detector
IC3: Capture input signal of channel 3 selected by CC3S
OC3REF: output reference waveform of channel 3 (middle waveform)
OC3: Main output signal of channel 3 (OC3REF signal after CC3P polarity processing)

OC3N: Complementary output signal of channel 3 (OC3REF signal after CC3NP polarity processing)

TI4: External clock input signal 4 (PWM4P pin signal)
TI4F: TI4 signal after IC4F digital filtering
TI4F_ED: TI4F edge signal
TI4FP: TI4F signal after CC3P/CC4P edge detector
TI4FP3: TI4F signal after CC3P edge detector
TI4FP4: TI4F signal after CC4P edge detector
IC4: The capture input signal of channel 4 selected by CC4S
OC4REF: output reference waveform of channel 4 (middle waveform)
OC4: Main output signal of channel 4 (OC4REF signal after CC4P polarity processing)
OC4N: Complementary output signal of channel 4 (OC4REF signal after CC4NP polarity processing)
ITR1: Internal trigger input signal 1
ITR2: Internal trigger input signal 2

TRC: fixed to TI1_ED
TRGI: Trigger input signal after TS multiplexer
TRGO: Trigger output signal after MMS multiplexer

ETR: External trigger input signal (PWMETI1 pin signal)
ETRP: ETR signal after passing ETP edge detector and ETPS divider
ETRF: ETRP signal after ETF digital filtering

BRK: Brake input signal (PWMFLT)

CK_PSC: prescaler clock, PWMA_PSCR prescaler input clock
CK_CNT: PWMA_PSCR prescaler output clock, PWM timer clock

The second group of advanced PWM timer/PWMB internal signal description

TI5: External clock input signal 5 (PWM5 pin signal or signal after the exclusive OR of PWM5/PWM6/PWM7)
TI5F: TI5 signal after IC5F digital filtering
TI5FP: TI5F signal after CC5P/CC6P edge detector
TI5F_ED: TI5F edge signal
TI5FP5: TI5F signal after CC5P edge detector
TI5FP6: TI5F signal after CC6P edge detector
IC5: The capture input signal of channel 5 selected by CC5S
OC5REF: output reference waveform of channel 5 (middle waveform)
OC5: Main output signal of channel 5 (OC5REF signal after CC5P polarity processing)

TI6: External clock input signal 6 (PWM6 pin signal)
TI6F: TI6 signal after IC6F digital filtering
TI6F_ED: TI6F edge signal
TI6FP: TI6F signal after CC5P/CC6P edge detector
TI6FP5: TI6F signal after CC5P edge detector
TI6FP6: TI6F signal after CC6P edge detector
IC6: The capture input signal of channel 6 selected by CC6S
OC6REF: output reference waveform of channel 6 (middle waveform)
OC6: Main output signal of channel 6 (OC6REF signal after CC6P polarity processing)

TI7: External clock input signal 7 (PWM7 pin signal)
TI7F: TI7 signal after IC7F digital filtering
TI7F_ED: TI7F edge signal
TI7FP: TI7F signal after CC7P/CC8P edge detector
TI7FP7: TI7F signal after CC7P edge detector
TI7FP8: TI7F signal after CC8P edge detector
IC7: Capture input signal of channel 7 selected by CC7S
OC7REF: output reference waveform of channel 7 (middle waveform)
OC7: Main output signal of channel 7 (OC7REF signal after CC7P polarity processing)

TI8: External clock input signal 8 (PWM8 pin signal)
TI8F: TI8 signal after IC8F digital filtering
TI8F_ED: TI8F edge signal
TI8FP: TI8F signal after CC7P/CC8P edge detector
TI8FP7: TI8F signal after CC7P edge detector
TI8FP8: TI8F signal after CC8P edge detector
IC8: The capture input signal of channel 8 selected by CC8S
OC8REF: output reference waveform of channel 8 (middle waveform)
OC8: Main output signal of channel 8 (OC8REF signal after CC8P polarity processing)

20.1 Introduction

PWMA consists of a 16-bit auto-load counter, which is driven by a programmable prescaler.
PWMA is suitable for many different purposes:

- Basic timing
- Measure the pulse width of the input signal (input capture)
- Generate output waveforms (output compare, PWM and single pulse mode)
- Corresponding to interrupts of different events (capture, compare, overflow, brake, trigger)
- Synchronize with PWMB or external signals (external clock, reset signal, trigger and enable signal)

PWMA is widely used in a variety of control applications, including those that require mid-aligned mode PWM, which supports complementary output and dead time control. The clock source of PWMA can be an internal clock or an external signal, which can be selected through the configuration register.

20.2 Main features

Features of PWMA include:

- 16-bit up, down, up/down auto load counter
- Allows to update the repeat counter of the timer register after a specified number of counter cycles
- 16-bit programmable (can be modified in real time) prescaler, the division coefficient of the counter clock frequency is any value between 1～65535
- Synchronization circuit, used to control timer and timer interconnection with external signal
- Up to 4 independent channels can be configured as:
  — Input capture
  — Output comparison
  — PWM output (edge or center alignment mode)
  — Six-step PWM output
  — Single pulse mode output
  — Supports complementary output on 4 channels with programmable dead time
- The brake input signal (PWMFLT) can put the timer output signal in a reset state or a certain state
- External trigger input pin (PWMETI)
- Events that generate interrupts include:
  — Update: Counter overflow/downflow, counter initialization (by software or internal/external trigger)
  — Trigger event (counter start, stop, initialization or count by internal/external trigger)
  — Input capture, measure pulse width

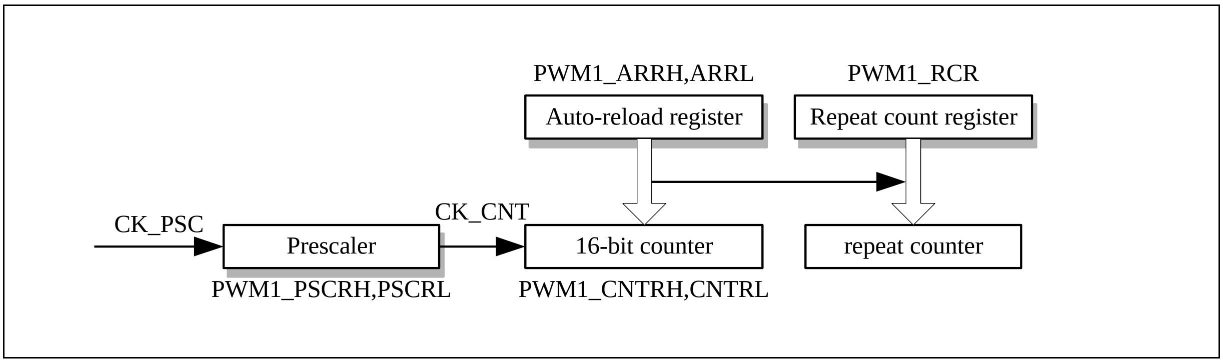
— External Interrupt
— Output comparison
— Brake signal input

20.3 Time base unit

The time base unit of PWMA includes:

- 16-bit up/down counter
- 16-bit automatic reload register
- Repeat counter
- Prescaler

PWMA time base unit



The 16-bit counter, prescaler, auto-reload register and repeat counter register can all be read and written by software. The auto-reload register is composed of preload register and shadow register.

The automatic reload register can be written in two modes:

- Auto preloading is enabled (ARPE bit of PWMA_CR1 register is 1). In this mode, the data written into the auto-reload register will be saved in the preload register and transferred to the shadow register at the next update event (UEV).
- Auto preloading is disabled (ARPE bit of PWMA_CR1 register is 0). In this mode, the data written to the auto-reload register will be written to the shadow register immediately.

Conditions for generating update events:

- The counter overflows up or down.
- The software sets the UG bit in the PWMA_EGR register.
- The clock/trigger controller generates a trigger event.

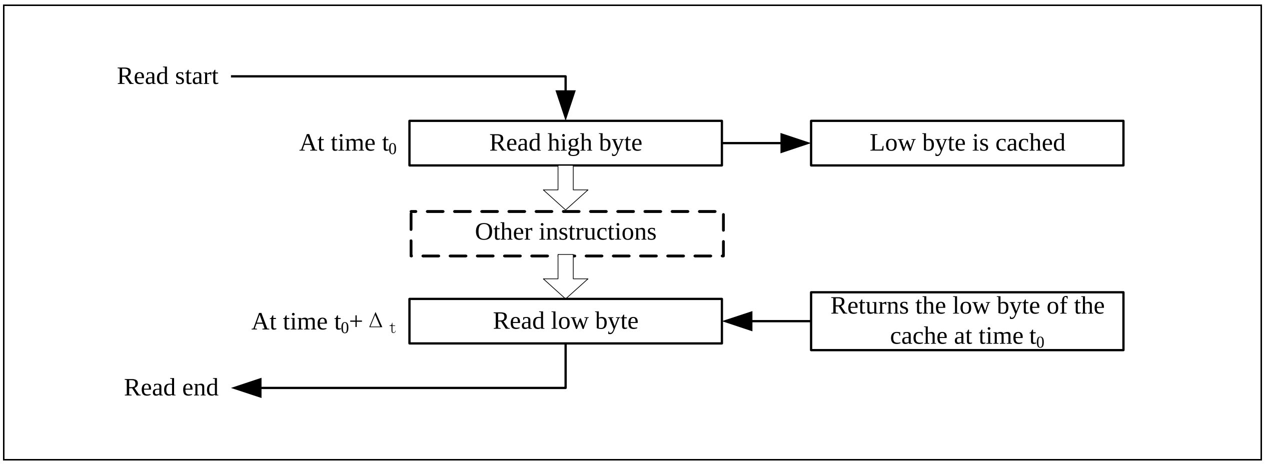
When the preload is enabled (ARPE=1), if an update event occurs, the value in the preload register (PWMA_ARR) will be written into the shadow register, and the value in the PWMA_PSCR register will be written into the prescaler . Setting the UDIS bit in the PWMA_CR1 register will disable the update event (UEV). The output of the prescaler CK_CNT drives the counter, and CK_CNT is only valid when the counter enable bit (CEN) of the PWMA_CR1 register is set.

Note: The actual counter does not start counting until one clock cycle after the CEN bit is enabled.

20.3.1 Reading and writing 16-bit counter

There is no buffer for writing counter operations, and PWMA_CNTRH and PWMA_CNTRL registers can be written at any time. Therefore, in order to avoid writing wrong values, it is generally recommended not to write new values while the counter is running.

The operation of the read counter has an 8-bit buffer. The user must first read the high byte of the timer. After the user reads the high byte, the low byte will be automatically buffered, and the buffered data will be kept until the 16-bit data read operation is completed.



20.3.2 Writing operation of 16-bit PWMA_ARR register

The value in the preload register will be written into the 16-bit PWMA_ARR register. This operation is completed by two instructions, each of which writes 1 byte. The high byte must be written first, followed by the low byte.

The shadow register is locked when the high byte is written, and remains until the low byte is written.

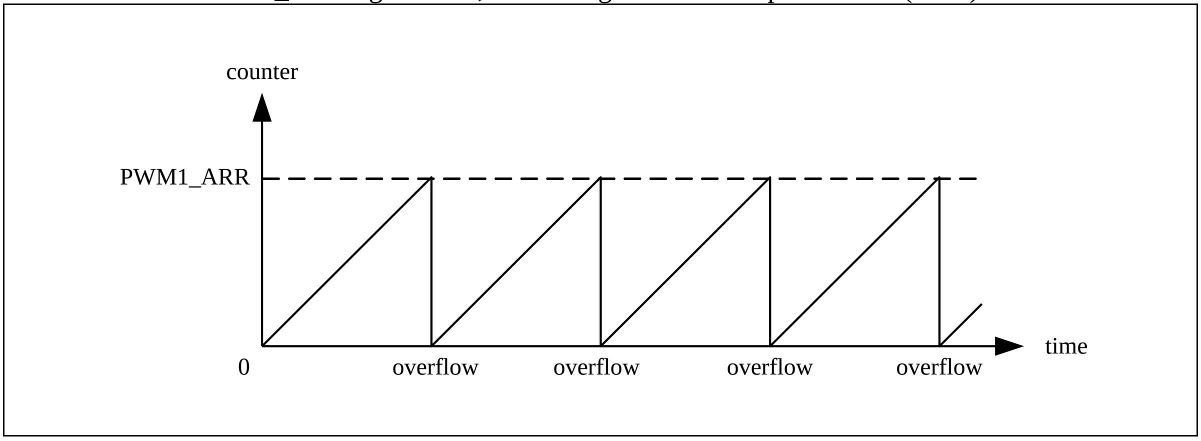
20.3.3 Prescaler

The prescaler of PWMA is based on a 16-bit counter controlled by a 16-bit register (PWMA_PSCR). Since this control register has a buffer, it can be changed at runtime. The prescaler can divide the counter clock frequency by any value between 1 and 65536. The value of the prescaler is written by the preload register, and the shadow register holding the current value is loaded when the low byte is written. Since two separate write operations are required to write 16-bit registers, it is necessary to ensure that the high byte is written first. The new prescaler value is used when the next update event arrives. The read operation of the PWMA_PSCR register is completed through the preload register.

Frequency calculation formula of counter: $f_{CK_{CNT}} = f_{CK_{PSC}} \frac{\square}{PSCR[15:0]\ +\ 1}$

20.3.4 Up counting mode

In the up-counting mode, the counter counts from 0 to the user-defined comparison value (the value of the PWMA_ARR register), and then restarts counting from 0 and generates a counter overflow event. At this time, if the UDIS bit of the PWMA_CR1 register is 0, it will be generated An update event (UEV).



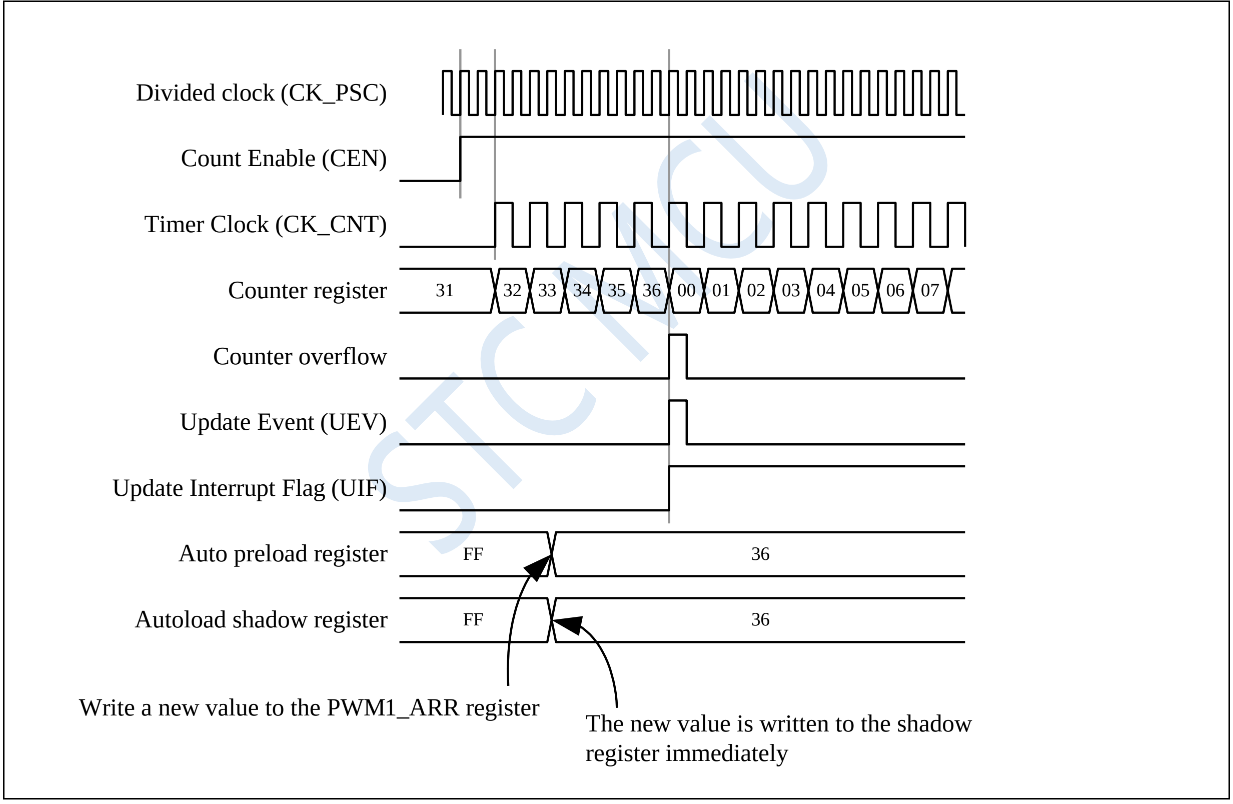
An update event can also be generated by software or by using the trigger controller to set the UG bit of the PWMA_EGR register. Use software to set the UDIS bit of the PWMA_CR1 register to disable the update event, which can avoid updating the shadow register when updating the preload register. Until the UDIS bit is cleared, no update event will be generated. But when an update event should occur, the counter will still be cleared, and the count of the prescaler will also be cleared (but the value of the prescaler will not change). In addition, if the URS bit in the PWMA_CR1 register (select update request) is set, setting the UG bit will generate an update event UEV, but the hardware does not set the UIF flag (that is, no interrupt request is generated). This is to avoid the update and capture interrupts when the counter is cleared in capture mode.

When an update event occurs, all registers are updated, and the hardware simultaneously sets the update flag (the UIF bit of the PWMA_SR register) according to the URS bit:

- The autoload shadow register is reset to the value of the preload register (PWMA_ARR).
- The buffer of the prescaler is set into the value of the preload register (PWMA_PSC).

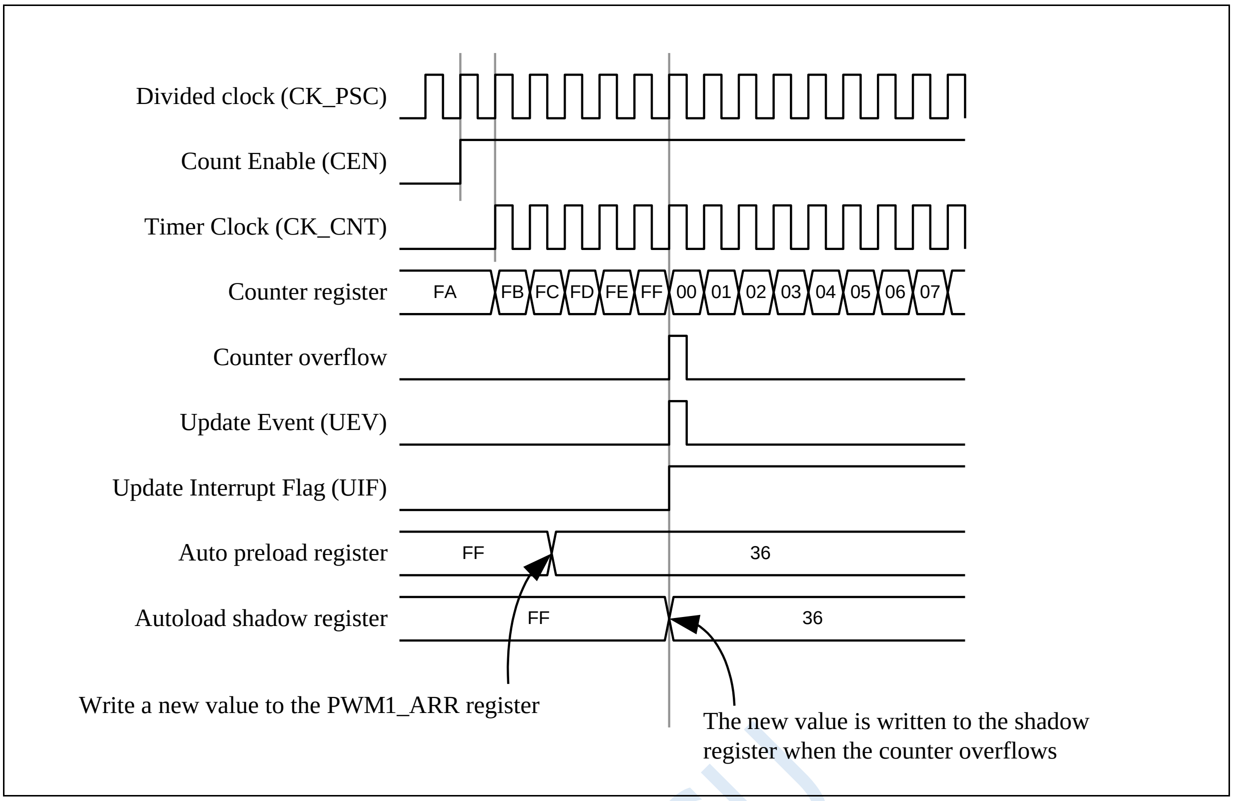
The following figure gives some examples to illustrate the actions of the counter at different clock frequencies when PWMA_ARR=0x36. The prescaler in the figure is 2, so the counter clock (CK_CNT) frequency is half of the prescaler clock (CK_PSC) frequency. In the figure, the automatic loading function is disabled (ARPE=0), so when the counter reaches 0x36, the counter overflows, the shadow register is updated immediately, and an update event is generated at the same time.

When ARPE=0 (ARR is not preloaded), the counter is updated when the prescaler is 2:



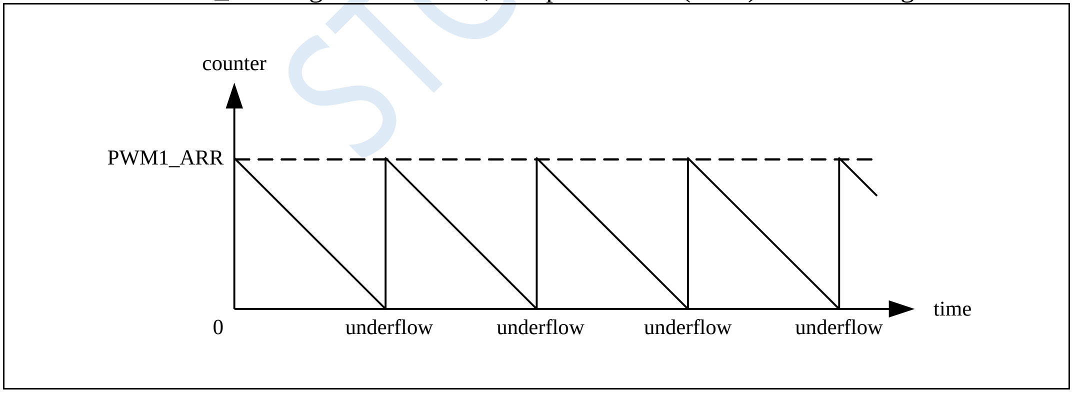
The prescaler in the figure below is 1, so the frequency of CK_CNT is the same as CK_PSC. In the figure, auto reload is enabled (ARPE=1), so an overflow occurs when the counter reaches 0xFF. 0x36 will be written on overflow and an update event will be generated at the same time.

When ARPE=1 (PWMA_ARR preload), the counter update when the prescale is 1:



20.3.5 Down counting mode

In the down mode, the counter starts counting down from the auto-loaded value (PWMA_ARR register value) to 0, and then restarts counting from the auto-loaded value, and a counter overflow event is generated. If the UDIS bit of the PWMA_CR1 register is cleared, an update event (UEV) will also be generated.



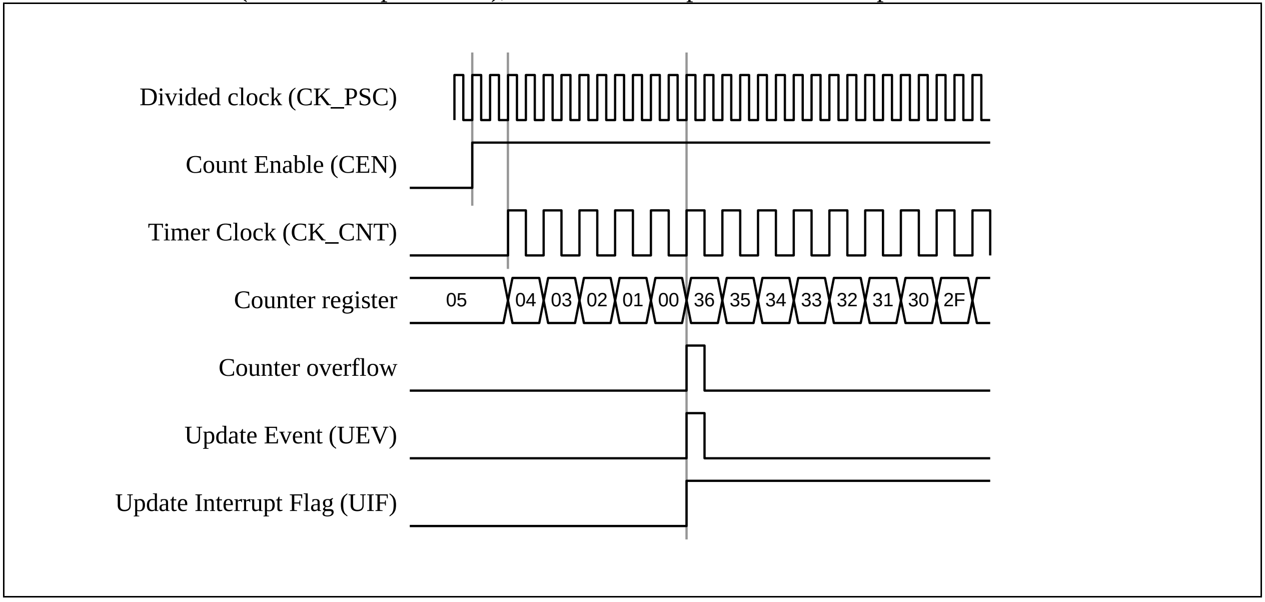
An update event can also be generated by software or by using the trigger controller to set the UG bit of the PWMA_EGR register. The UEV event can be disabled by setting the UDIS bit in the PWMA_CR1 register. This avoids updating the shadow register when updating the preload register. Therefore, no update event will be generated before the UDIS bit is cleared. However, the counter will still start counting from the current autoload value, and the counter of the prescaler will restart from 0 (but the prescaler cannot be modified). In addition, if the URS bit (select update request) in the PWMA_CR1 register is set, setting the UG bit will generate an update event UEV without setting the UIF flag (so no interrupt is generated). This is to avoid the occurrence of a capture event and clearing the counter , Simultaneously generate update and capture interrupts.

When an update event occurs, all registers are updated, and the hardware sets the update flag (the UIF bit of the PWMA_SR register) according to the URS bit at the same time:

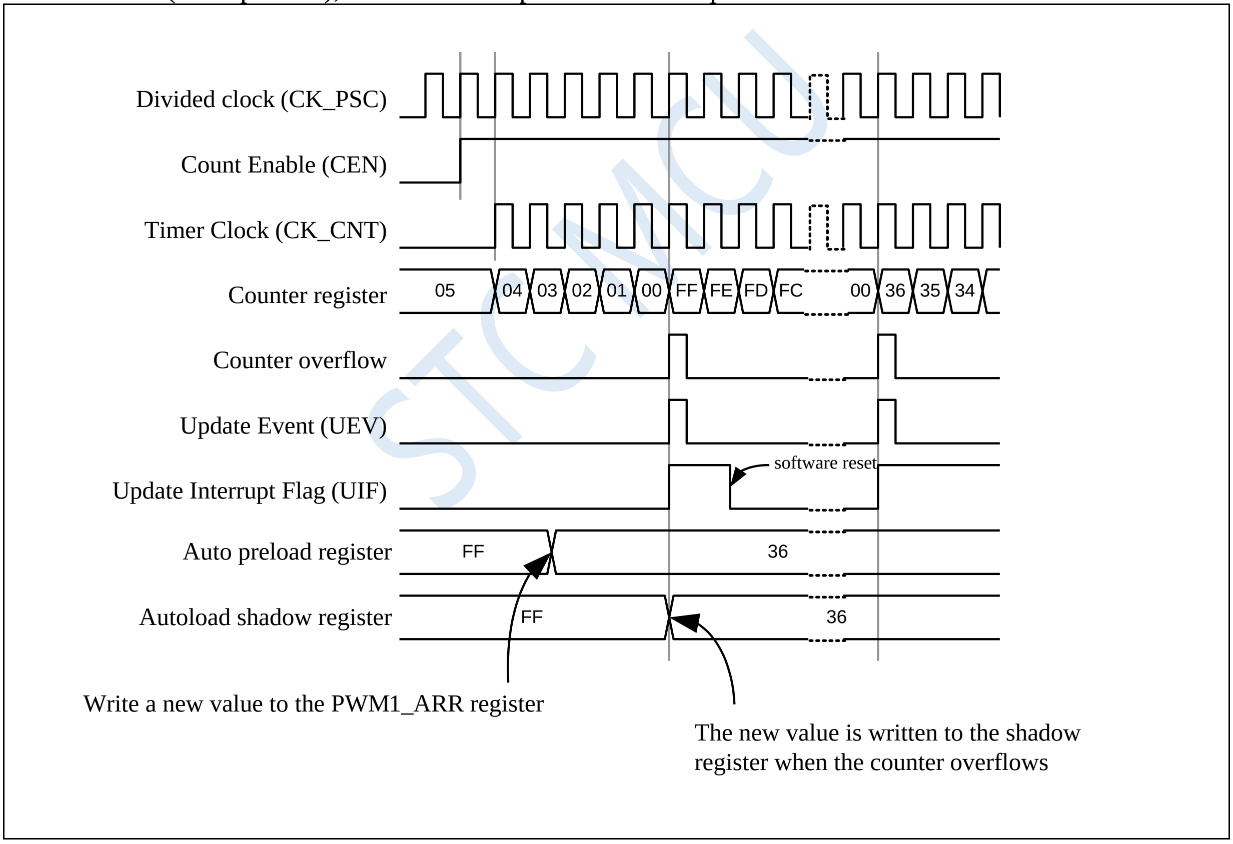
- The autoload shadow register is reset to the value of the preload register (PWMA_ARR).
- The buffer of the prescaler is set to the value of the preload register (PWMA_PSC).

The following are some charts of the counter at different clock frequencies when PWMA_ARR=0x36. The following figure depicts that in the down-counting mode, the new value is written in the next cycle when preloading is disabled.

When ARPE=0 (ARR is not preloaded), the counter is updated when the prescaler is 2:

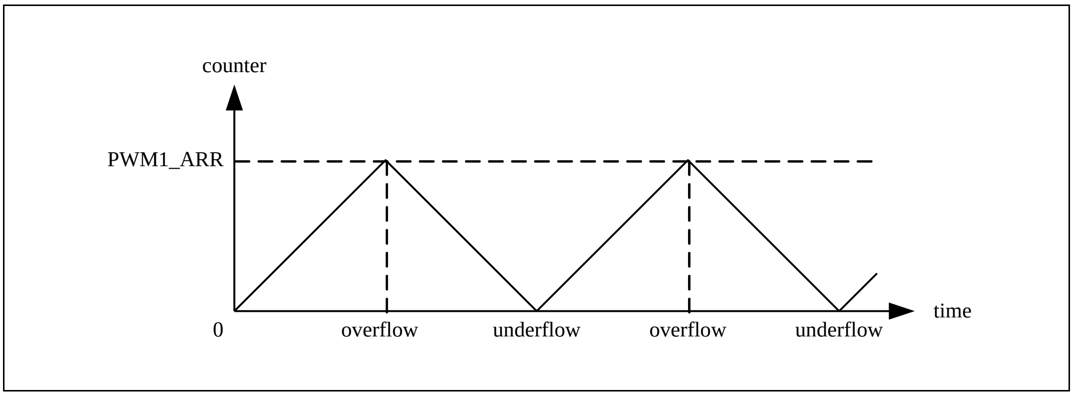


When ARPE=1 (ARR preload), the counter is updated when the prescaler is 1



20.3.6 Center alignment mode (up/down count)

In the center-aligned mode, the counter starts counting from 0 to the value of the PWMA_ARR register, generates a counter overflow event, then counts down from the value of the PWMA_ARR register to 0 and generates a counter underflow event; then restarts counting from 0. In this mode, the DIR direction bit in PWMA_CR1 cannot be written. It is updated by hardware and indicates the current counting direction.



If the timer has a repetition counter, an update event (UEV) will be generated after the up and down overflow of the specified number of times (the value of PWMA_RCR) is repeated. Otherwise, each upward and downward overflow will generate an update event. An update event can also be generated by software or by using the trigger controller to set the UG bit of the PWMA_EGR register. At this time, the counter starts counting from 0 again, and the prescaler also starts counting from 0 again. The UEV event can be disabled by setting the UDIS bit in the PWMA_CR1 register. This avoids updating the shadow register when updating the preload register. Therefore, no update event will be generated before the UDIS bit is cleared to 0. However, the counter will continue to count up or down according to the current auto-reload value. If the timer has a repeat counter, since the repeat register is not double-buffered, the new repeat value will take effect immediately, so be careful when modifying it. In addition, if the URS bit (select update request) in the PWMA_CR1 register is set, setting the UG bit will generate an update event UEV but does not set the UIF flag (so no interrupt is generated). This is to avoid the occurrence of a capture event and clear the counter , Simultaneously generate update and capture interrupts.

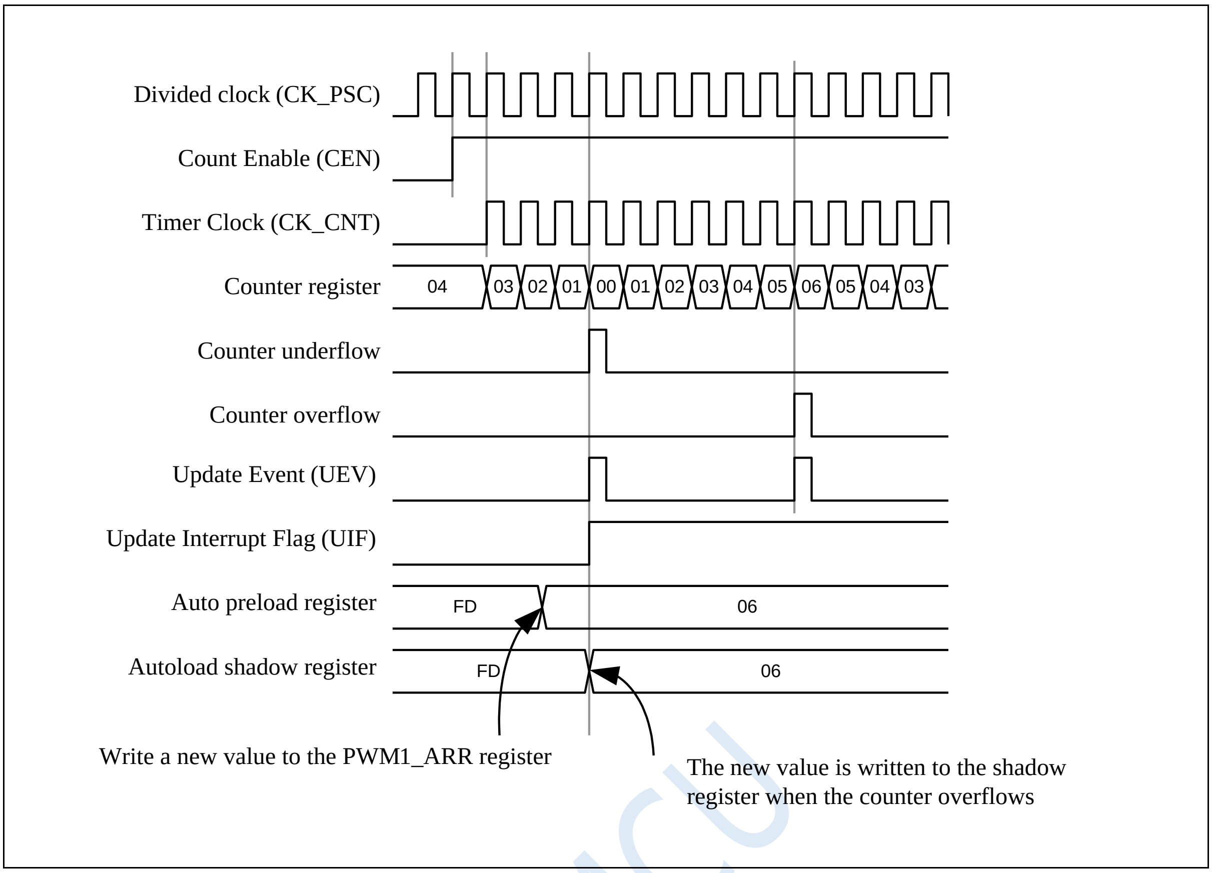
When an update event occurs, all registers are updated, and the hardware updates the flag bit according to the URS bit (the UIF bit in the PWMA_SR register):

- The register of the prescaler is loaded with the preloaded value (PWMA_PSCR).
- The current autoload register is updated to the preload value (PWMA_ARR).

It should be noted that if an update occurs due to a counter overflow, the auto-reload register will be updated before the counter is reloaded, so the next cycle is the expected value (the counter is loaded with the new value).

The following are some examples of counter operations at different clock frequencies:

The internal clock frequency division factor is 1, PWMA_ARR=0x6, ARPE=1



Tips for using center alignment mode:

- When the center alignment mode is activated, the counter will count according to the original up/down configuration. In other words, the DIR bit in the PWMA_CR1 register will determine whether the counter counts up or down. In addition, the software cannot modify the value of DIR bit and CMS bit at the same time.
- It is not recommended to write the counter value when the counter is counting in the center-aligned mode. This will cause unpredictable consequences. Specifically:

  — When a value larger than the auto-load value is written to the counter (PWMA_CNT>PWMA_ARR), the counting direction of the counter does not change. For example, the counter has overflowed, but the counter is still counting up.

  — Write 0 or the value of PWMA_ARR to the counter, but the update event does not occur.
- The safe way to use the counter in center-aligned mode is to use software (set the UG bit of the PWMA_EGR register) to generate an update event before starting the counter, and not to modify the counter value when the counter is counting.

20.3.7 Repeat counter

The time base unit explains how the update event (UEV) is generated when the counter overflows/downflows, but in fact it can only be generated when the value of the repeat counter reaches 0. This feature is very useful for generating PWM signals.

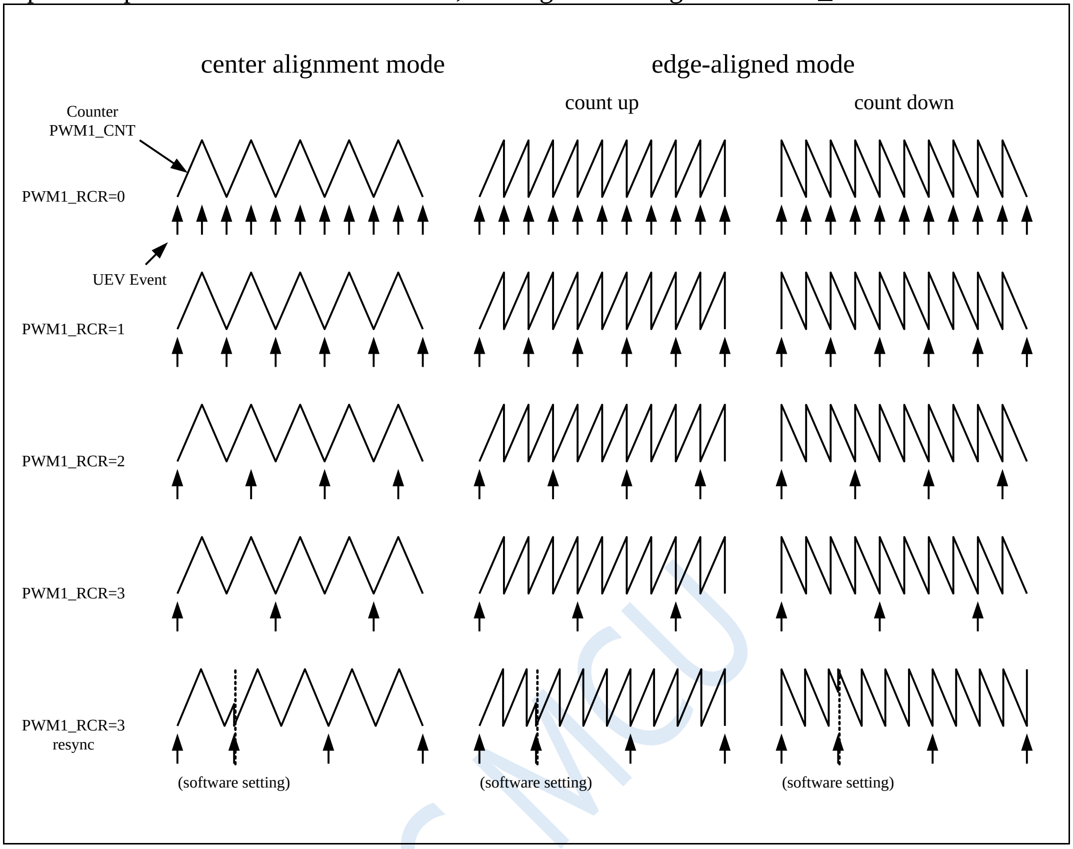
This means that every N counts overflow or underflow, the data is transferred from the preload register to the shadow register (PWMA_ARR auto-reload register, PWMA_PSCR preload register, and capture/compare register in compare mode PWMA_CCRx), N is the value in the PWMA_RCR repeat count register.

The repeat counter is decremented when any of the following conditions are met:

- Each time the counter overflows in the up-counting mode
- Every time the counter overflows in down counting mode
- At each overflow and each underflow in the center-aligned mode. Although this limits the maximum PWM cycle period to 128, it can update the duty cycle twice in each PWM cycle. In the center-aligned mode, because the waveform is symmetrical, if the compare register is refreshed only once in each PWM cycle, the maximum resolution is 2*tCK_PSC.

The repetition counter is automatically loaded, and the repetition rate is defined by the value of the PWMA_RCR register. When the update event is generated by software or by the hardware clock/trigger controller, no matter what the value of the repeat counter is, the update event occurs immediately, and the content in the PWMA_RCR register is reloaded into the repeat counter.

Examples of update rates in different modes, and register settings of PWMA_RCR



20.4 Clock/Trigger Controller

The clock/trigger controller allows users to select the counter clock source, input trigger signal and output signal.

20.4.1 Prescaler Clock (CK_PSC)

The prescaler clock (CK_PSC) of the time base unit can be provided by the following sources:

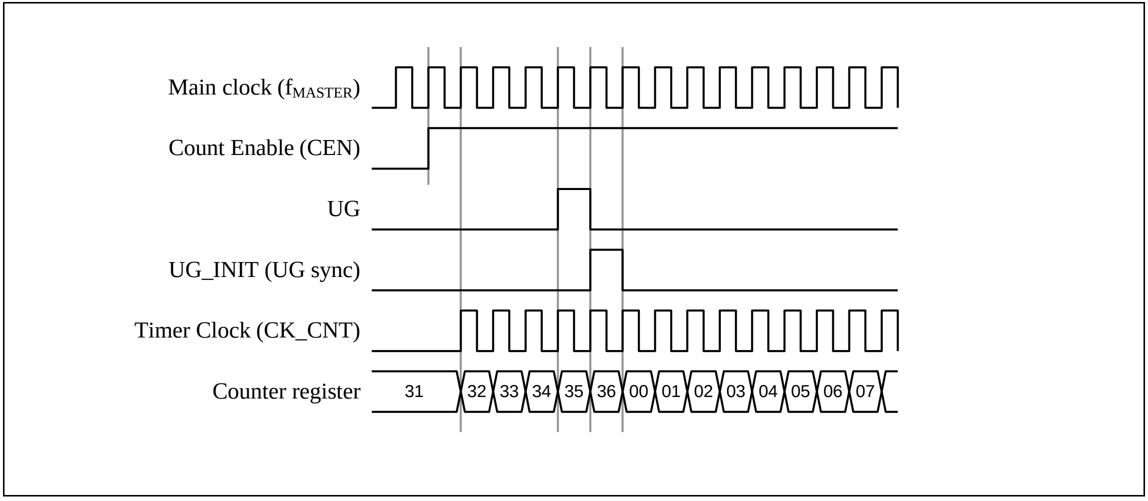
- Internal clock (fMASTER)
- External clock mode 1: External clock input (TIx)
- External clock mode 2: External trigger input ETR
- Internal trigger input (ITRx): Use TRGO of one PWM as the prescaler clock of another PWM.

20.4.2 Internal clock source (f_{MASTER})

If the clock/trigger mode controller and external trigger input are disabled at the same time (SMS=000 in the PWMA_SMCR register, ECE=0 in the PWMA_ETR register), the CEN, DIR and UG bits are the actual control bits and can only be modified by software (The UG bit is still automatically cleared). Once the CEN bit is written as 1, the prescaler clock is provided by the internal clock.

The following figure describes the operation of the control circuit and up-counter in normal mode without prescaler.

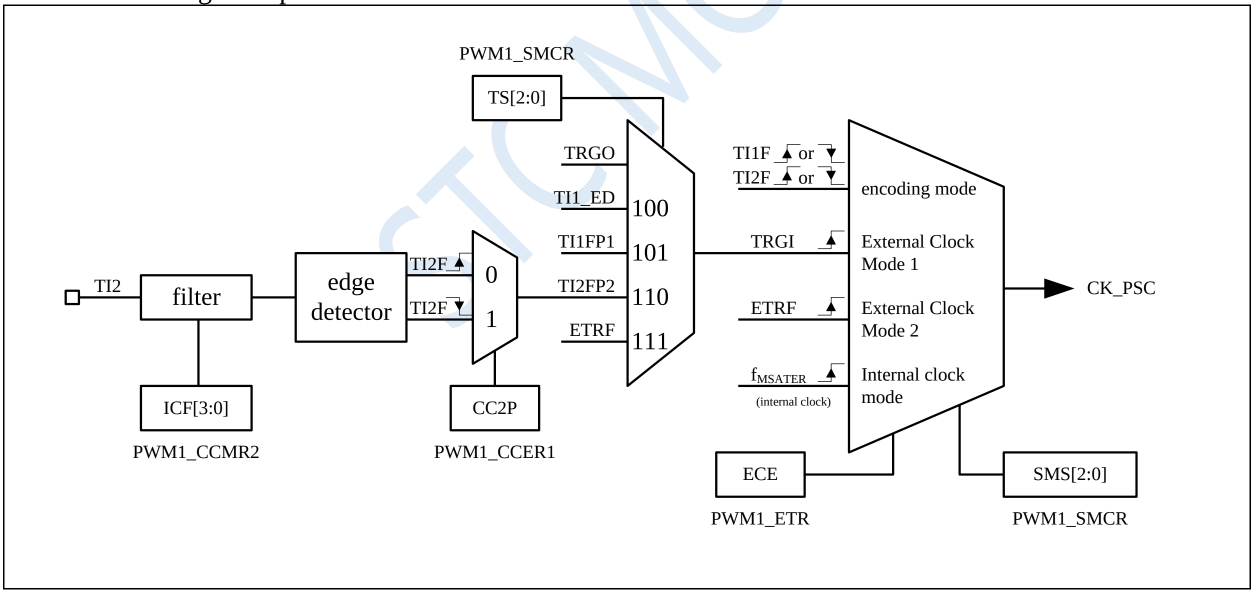
Control circuit in normal mode, fMASTER division factor is 1



20.4.3 External clock source mode 1

When SMS=111 in PWMA_SMCR register, this mode is selected. Then select the signal source of TRGI through TS of PWMA_SMCR register. The counter can count on every rising or falling edge of the selected input.

The following example uses TI2 as the external clock



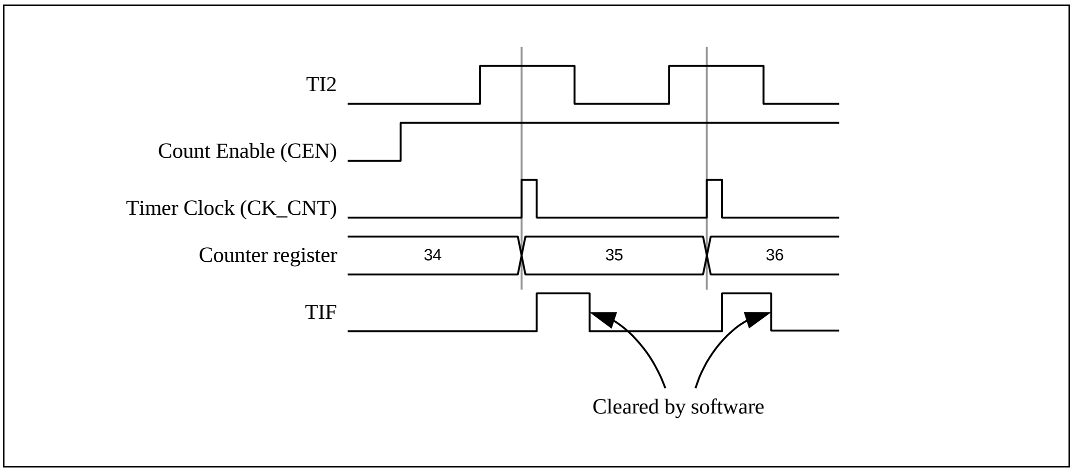
For example, to configure the up counter to count on the rising edge of the TI2 input, use the following steps:

1. Configure CC2S=01 in PWMA_CCMR2 register, use channel 2 to detect the rising edge of TI2 input
2. Configure the IC2F[3:0] bits of the PWMA_CCMR2 register to select the input filter bandwidth
3. Configure CC2P=0 in PWMA_CCER1 register, select the rising edge polarity
4. Configure SMS=111 in the PWMA_SMCR register, configure the counter to use external clock mode 1
5. Configure TS=110 in PWMA_SMCR register and select TI2 as input source
6. Set CEN=1 in PWMA_CR1 register to start the counter

When the rising edge occurs at TI2, the counter counts once, and the trigger flag (TIF bit of the PWMA_SR1 register) is set to 1, if the interrupt is enabled (configured in the PWMA_IER register), an interrupt request will be generated.

The delay between the rising edge of TI2 and the actual counter clock depends on the resynchronization circuit at the input of TI2.

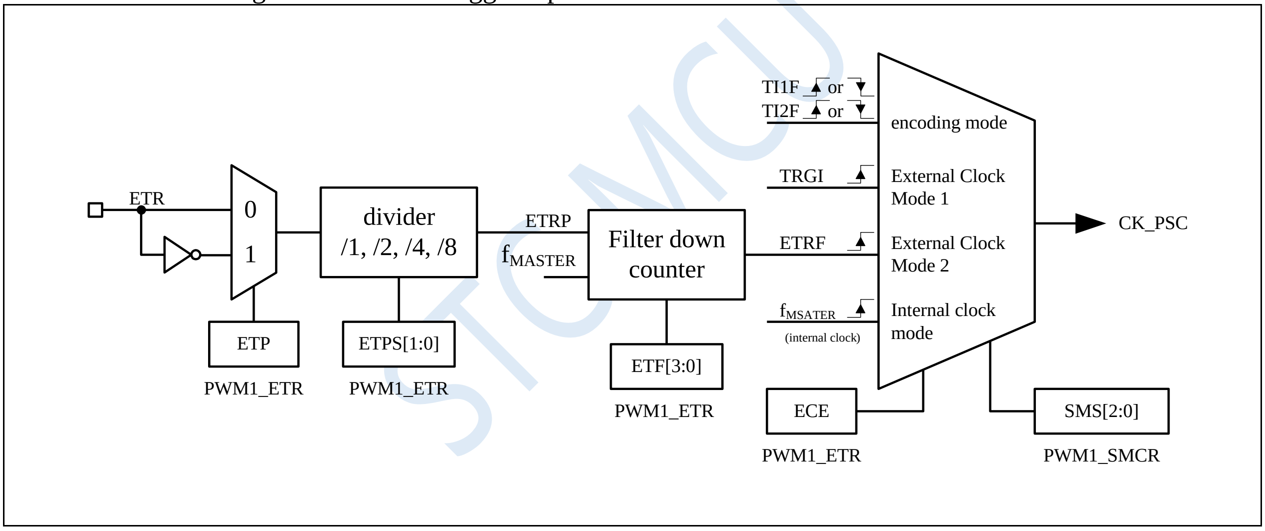
Control circuit in external clock mode 1



20.4.4 External clock source mode 2

The counter can count on every rising or falling edge of the external trigger input ETR signal. Write 1 to the ECE bit of the PWMA_ETR register to select this mode. (When SMS=111 in PWMA_SMCR register and TS=111 in PWMA_SMCR register, this mode can also be selected)

Overall block diagram of external trigger input:

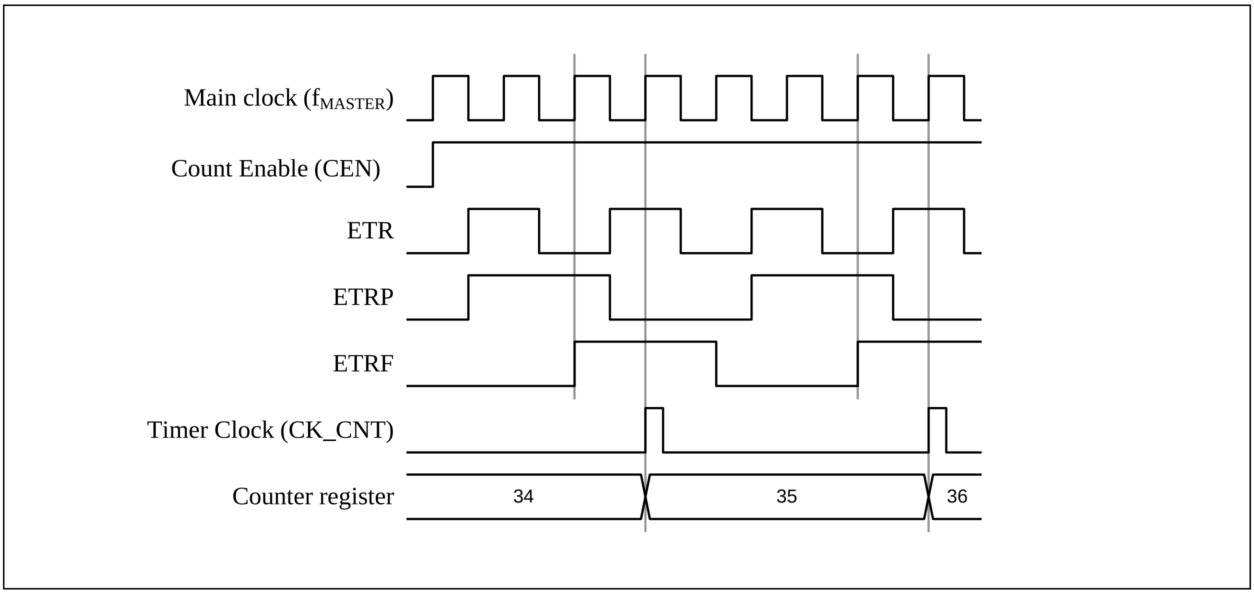


For example, to configure the counter to count up every 2 rising edges of the ETR signal, use the following steps:

1. No filter is needed in this example, configure ETF[3:0]=0000 in PWMA_ETR register
2. Set the prescaler, configure ETPS[1:0]=01 in the PWMA_ETR register
3. Select the rising edge detection of ETR, configure ETP=0 in the PWMA_ETR register
4. Turn on external clock mode 2, configure ECE=1 in the PWMA_ETR register
5. Start the counter, write CEN=1 in the PWMA_CR1 register

The counter counts once every 2 ETR rising edges.

Control circuit in external clock mode 2



20.4.5 Trigger synchronization

The PWMA counter uses three modes to synchronize with the external trigger signal:

- Standard trigger mode
- Reset trigger mode
- Gated trigger mode

Standard trigger mode

The enable of the counter (CEN) depends on the event on the selected input.

In the following example, the counter starts counting up on the rising edge of the TI2 input:

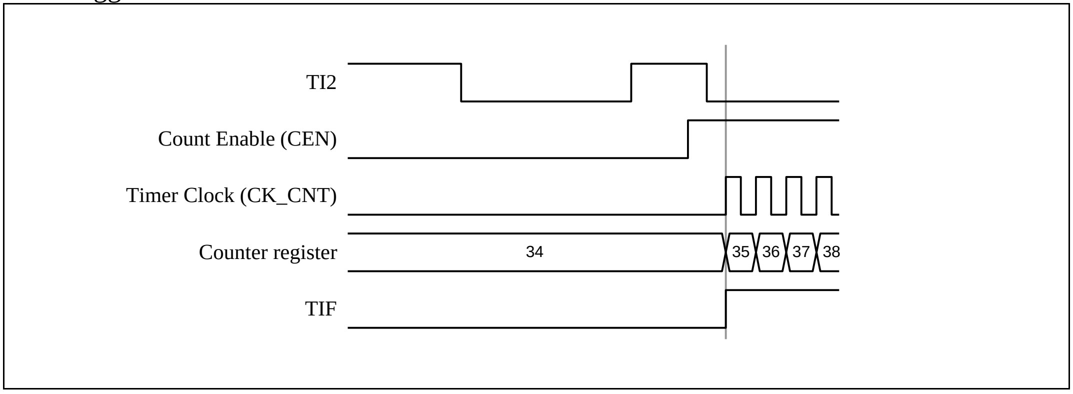
1. Configure CC2P=0 in the PWMA_CCER1 register and select the rising edge of TI2 as the trigger condition.

2. Configure SMS=110 in the PWMA_SMCR register and select the counter as trigger mode. Configure the PWMA_SMCR register

TS=110, select TI2 as the input source.

When TI2 has a rising edge, the counter starts to count under the internal clock drive and the TIF flag is set at the same time. The delay between the rising edge of TI2 and the counter starting to count depends on the resynchronization circuit at the TI2 input.

Standard trigger mode control circuit



Reset trigger mode

When a trigger input event occurs, the counter and its prescaler can be reinitialized. At the same time, if the URS bit of the PWMA_CR1 register is low, an update event UEV is also generated, and then all preload registers (PWMA_ARR, PWMA_CCRx) will be updated.

In the following example, the rising edge of the TI1 input causes the up counter to be cleared:

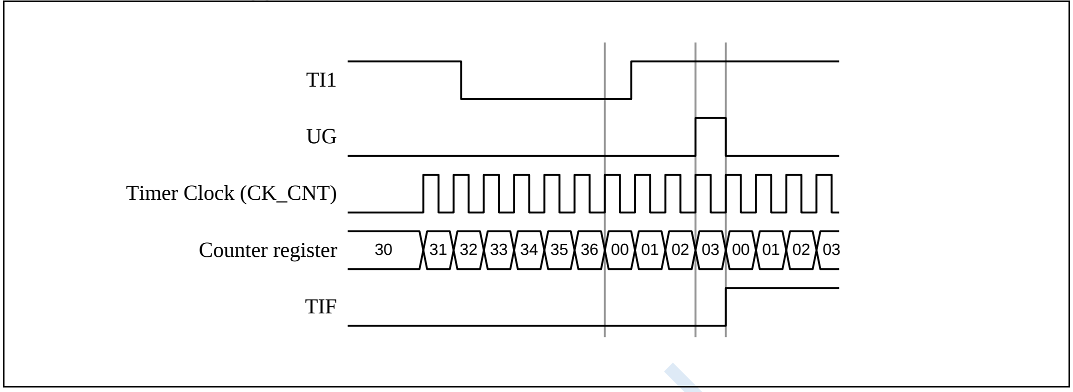
1. Configure CC1P=0 in the PWMA_CCER1 register to select the polarity of TI1 (only detect the rising edge of TI1).

2. Configure SMS=100 in the PWMA_SMCR register and select the timer as the reset trigger mode. Configure TS=101 in PWMA_SMCR register and select TI1 as input source.
3. Configure CEN=1 in the PWMA_CR1 register to start the counter.

The counter starts to count according to the internal clock, and then counts normally until TI1 has a rising edge. At this time, the counter is cleared and then restarts counting from 0. At the same time, the trigger flag (TIF bit of PWMA_SR1 register) is set. If the interrupt is enabled (TIE bit of PWMA_IER register), an interrupt request is generated.

The following figure shows the action when the auto reload register PWMA_ARR=0x36. The delay between the rising edge of TI1 and the actual reset of the counter depends on the resynchronization circuit at the input of TI1.

Control circuit in reset trigger mode



Gated trigger mode

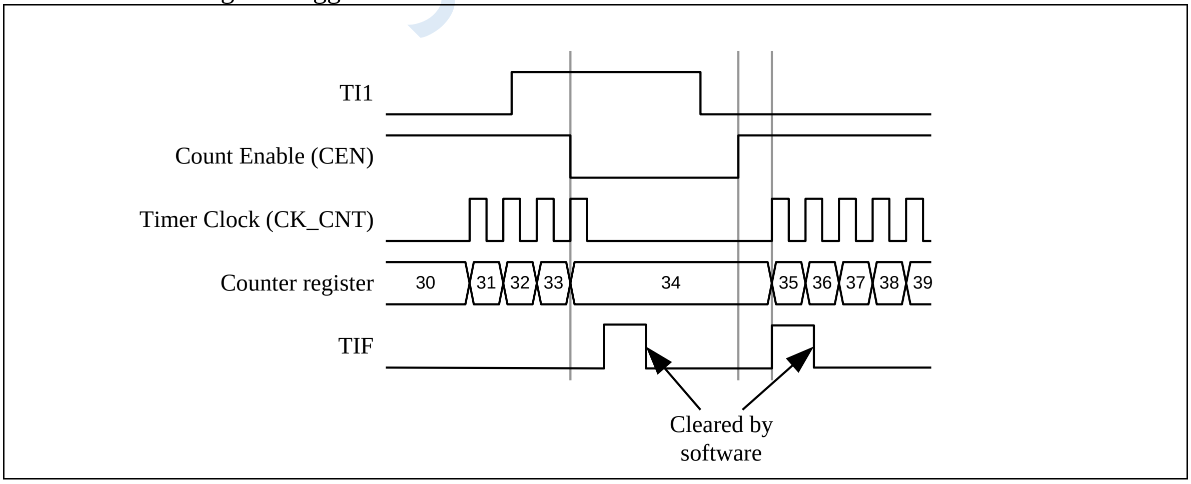
The counter is enabled by the level of the selected input signal.

In the following example, the counter only counts up when TI1 is low:

1. Configure CC1P=1 in the PWMA_CCER1 register to determine the polarity of TI1 (only detect the low level on TI1).
2. Configure SMS=101 in the PWMA_SMCR register, select the timer as gated trigger mode, configure TS=101 in the PWMA_SMCR register, and select TI1 as the input source.
3. Configure CEN=1 in the PWMA_CR1 register to start the counter (in gating mode, if CEN=0, the counter cannot be started, regardless of the trigger input level).

As long as TI1 is low, the counter starts counting according to the internal clock, and stops counting once TI1 goes high. The TIF flag will be set when the counter starts or stops. The delay between the rising edge of TI1 and the actual stop of the counter depends on the resynchronization circuit at the input of TI1.

Control circuit in gated trigger mode



External clock mode 2 Joint trigger mode

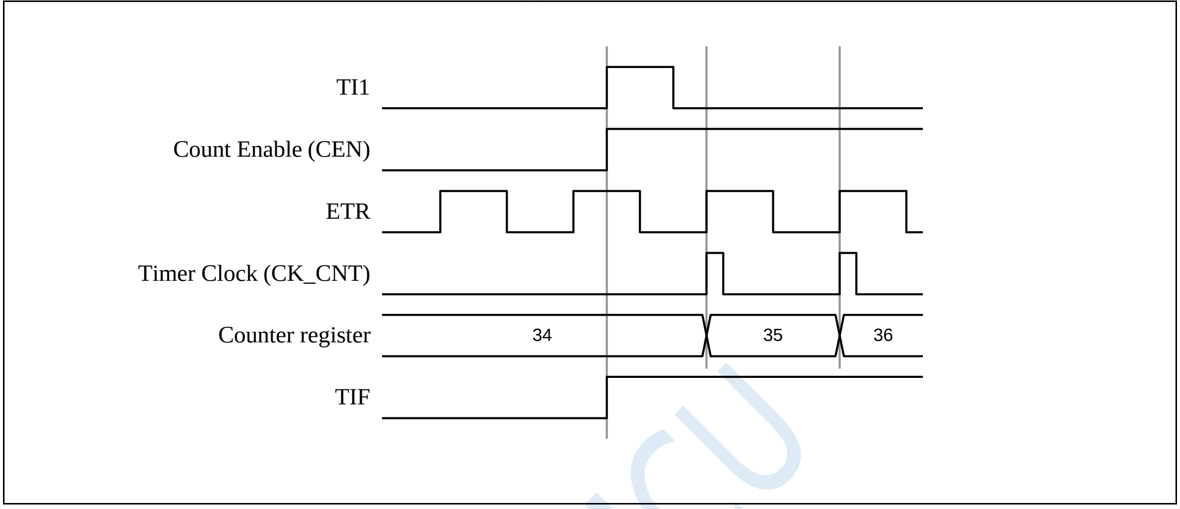
External clock mode 2 can be used with another input signal trigger mode. For example, the ETR signal is used as the input of an external clock, and another input signal can be used as the trigger input (supports standard trigger mode, reset trigger mode and gated trigger mode). Note that ETR cannot be configured as TRGI through the TS bit of the PWMA_SMCR register.

In the following example, once a rising edge occurs on TI1, the counter counts up once on each rising edge of ETR:

1. Configure the external trigger input circuit through the PWMA_ETR register. Configure ETPS=00 to disable prescaler, configure ETP=0 to monitor the rising edge of the ETR signal, configure ECE=1 to enable external clock mode 2.
2. Configure CC1P=0 in the PWMA_CCER1 register to select the rising edge trigger of TI1.
3. Configure SMS=110 in the PWMA_SMCR register to select the timer as trigger mode. Configure TS=101 in PWMA_SMCR register to select TI1 as the input source.

When a rising edge occurs on TI1, the TIF flag is set and the counter starts counting on the rising edge of ETR. The delay between the rising edge of the TI1 signal and the actual counter clock depends on the resynchronization circuit at the TI1 input. The delay between the rising edge of the ETR signal and the actual clock of the counter depends on the resynchronization circuit at the ETRP input.

External clock mode 2 + control circuit in trigger mode



20.4.6 Synchronize with PWMB

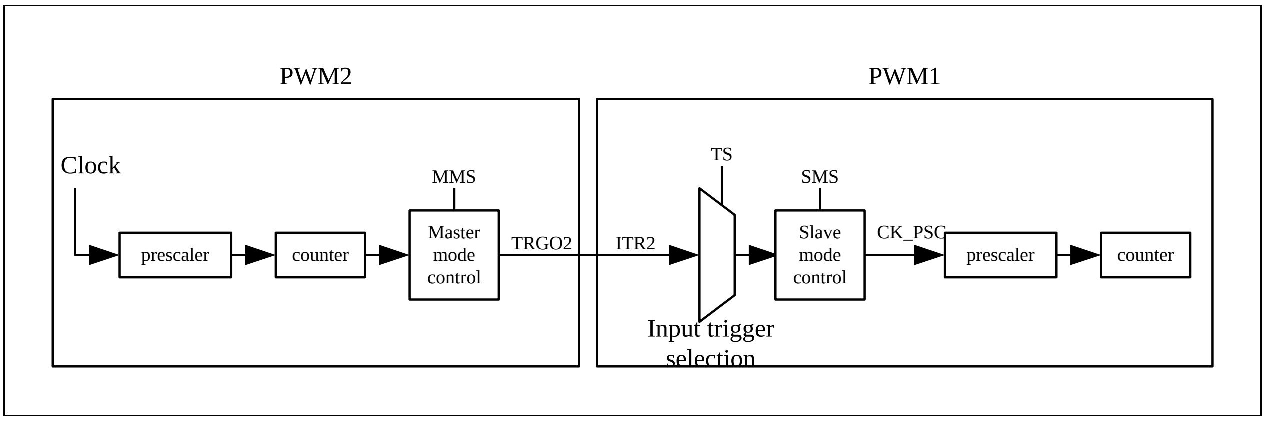
In the chip, the timers are interconnected internally for synchronization or linking of the timers. When a timer is configured as master mode, it can output trigger signal (TRGO) to those timers configured as slave mode to complete reset operation, start operation, stop operation or as the driving clock of those timers.

Use the TRGO of PWMB as the prescaler clock of PWMA

For example, users can configure PWMB as the prescaler clock of PWMA, and the following configuration is required:

1. Configure PWMB as the main mode, so that a periodic trigger signal is output at each update event (UEV). Configure MMS=010 in the PWMB_CR2 register so that TRGO can output a rising edge during each update event.
2. The TRGO signal output by PWMB is linked to PWMA. PWMA needs to be configured to trigger slave mode, using ITR2 as the input trigger signal. The above operations can be achieved by configuring TS=010 in the PWMA_SMCR register.
3. Configure SMS=111 in the PWMA_SMCR register to set the clock/trigger controller to external clock mode 1. This operation will cause the rising edge of the periodic trigger signal TRGO output by PWMB to drive the PWMA clock.
4. Finally, set the CEN bit of PWMB (in the PWMB_CR1 register) to enable two PWMs.

Example of master trigger slave mode



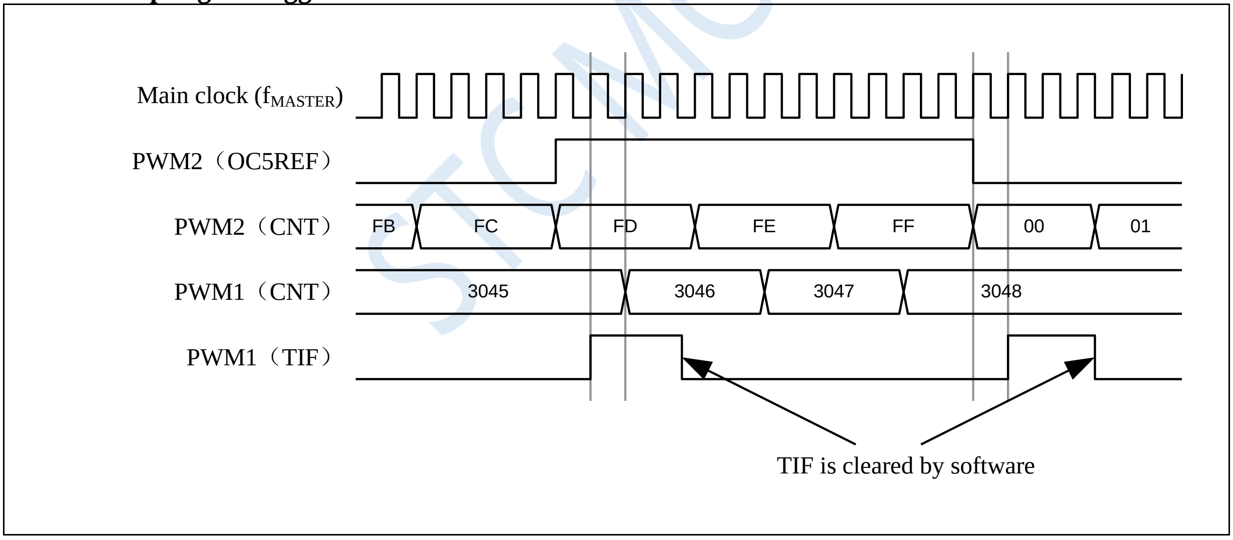
Use PWMB to enable PWMA

In this example, we use the compare output of PWMB to enable PWMA. PWMA counts according to its own driving clock only when the OC1REF signal of PWMB is high. Both PWMs use f_{MASTER} divided by 4 as the clock ($f_{CK\_CNT} = f_{MASTER}/4$).

1. Configure PWMB as the main mode, and output the comparison output signal (OC5REF) as the trigger signal. (Configure MMS=100 in PWMB_CR2 register).
2. Configure the waveform of the OC5REF signal of PWMB (PWMB_CCMR1 register).
3. Configure PWMA to use the output of PWMB as its own trigger input signal (configure TS=010 in PWMA_SMCR register).
4. Configure PWMA as the gated trigger mode (configure SMS=101 in the PWMA_SMCR register).
5. Set the CEN bit (PWMA_CR1 register) to enable PWMA.
6. Set the CEN bit (PWMB_CR1 register) to enable PWMB.

Note: The clocks of the two PWMs are not synchronized, but only affect the enable signal of PWMA.

PWMB output gate trigger PWMA

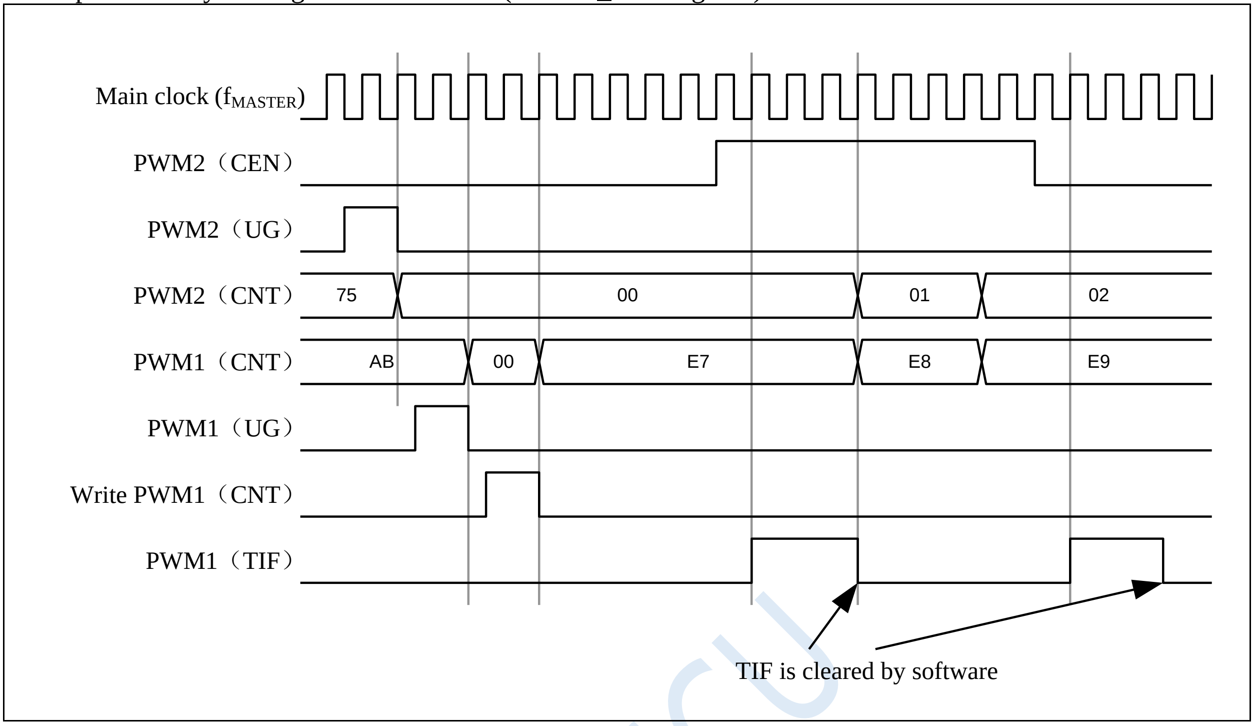


In the figure above, the PWMA counter and prescaler are not initialized before starting, so they start counting from the existing value. If the two timers are reset before starting PWMB, the user can write the desired value to the PWMA counter to start counting from the specified value. The reset operation of PWMA can be realized by software writing the UG bit of PWMA_EGR register.

In the following example, we synchronize PWMB and PWMA. PWMB is the main mode and starts counting from 0. PWMA is the trigger slave mode and starts counting from 0xE7. The two PWMs use the same frequency division factor. When the CEN bit in the PWMB_CR1 register is cleared, PWMB is disabled and PWMA stops counting.

1. Configure PWMB as the main mode, and output the comparison output signal (OC5REF) as the trigger signal. (Configure MMS=100 in PWMB_CR2 register).
2. Configure the waveform of the OC5REF signal of PWMB (PWMB_CCMR1 register).
3. Configure PWMA to use the output of PWMB as its own trigger input signal (configure TS=010 in PWMA_SMCR register).
4. Configure PWMA as the gated trigger mode (configure SMS=101 in the PWMA_SMCR register).
5. Reset PWMB by writing 1 to the UG bit (PWMB_EGR register).
6. Reset PWMA by writing 1 to the UG bit (PWMA_EGR register).

7. Write 0xE7 into the counter of PWMA (PWMA_CNTRL) to initialize PWMA.
8. Enable PWMA by writing 1 to the CEN bit (PWMA_CR1 register).
9. Start PWMB by writing 1 to the CEN bit (PWMB_CR1 register).
10. Stop PWMB by writing 0 to the CEN bit (PWMB_CR1 register).



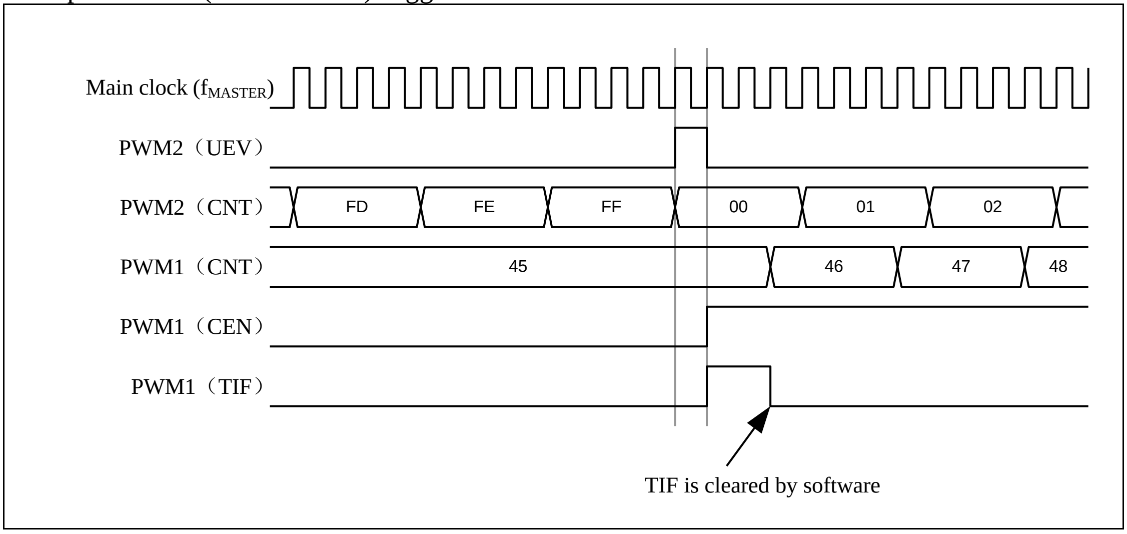
Use PWMB to start PWMA

In this example, we use the update event of PWMB to start PWMA.

When the PWMB update event occurs, PWMA starts counting from its existing value according to PWMA's own drive clock (it can be a non-zero value). After PWMA receives the trigger signal, it automatically enables the CEN bit and starts counting until the user writes 0 to the CEN bit of the PWMA_CR1 register. Both PWMs use fMASTER divided by 4 as the driving clock (fCK_CNT = fMASTER/4).

1. Configure PWMB as the main mode and output the update signal (UEV). (Configure MMS=010 in PWMB_CR2 register).
2. Configure the period of PWMB (PWMB_ARR register).
3. Configure PWMA to use the output of PWMB as the input trigger signal (configure TS=010 in PWMA_SMCR register).
4. Configure PWMA as trigger mode (configure SMS=110 in PWMA_SMCR register).
5. Set the CEN bit (PWMB_CR1 register) to start PWMB.

PWMB update event (PWMB-UEV) triggers PWMA



As in the previous example, the user can also initialize them before starting the counter.

Trigger two PWMs synchronously with external signals

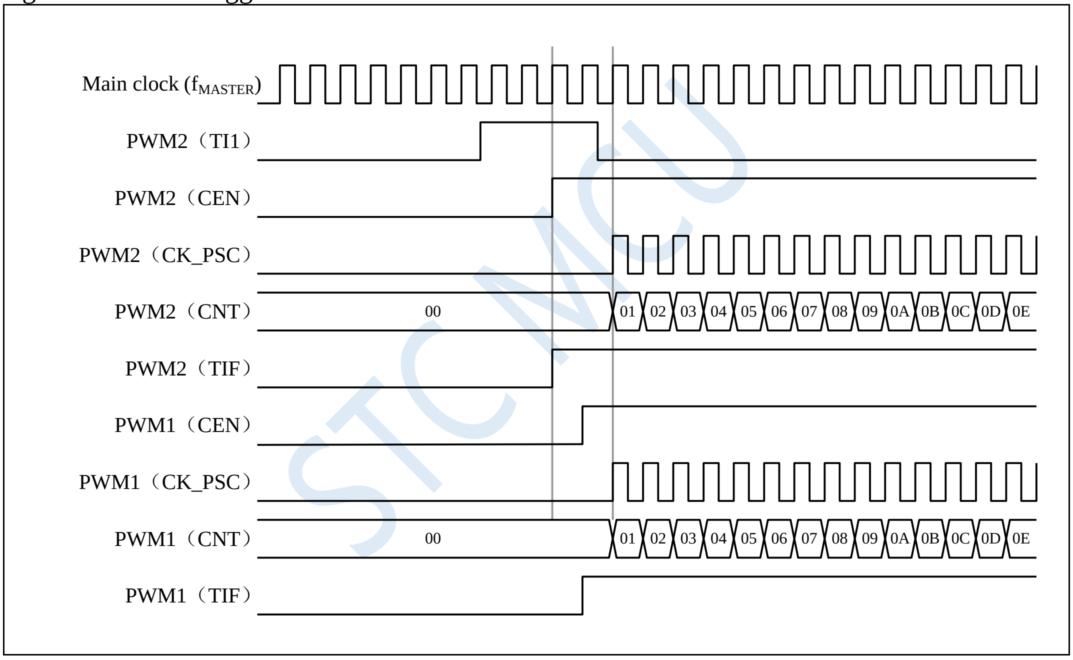
In this example, the rising edge of TI1 is used to enable PWMB and enable PWMA at the same time. In order to maintain timer alignment, PWMB needs to be configured in master/slave mode (slave mode for TI1 signal and master mode for PWMA).

1. Configure PWMB as the main mode, and use the output enable signal as the trigger of PWMA (configure MMS=001 in the PWMB_CR2 register).
2. Configure PWMB as slave mode, and use TI1 signal as the input trigger signal (configure TS=100 in PWMB_SMCR register).
3. Configure the trigger mode of PWMB (configure SMS=110 in PWMB_SMCR register).
4. Configure PWMB as master/slave mode (configure MSM=1 in PWMB_SMCR register).
5. Configure PWMA to use the output of PWMB as the input trigger signal (configure TS=010 in the PWMA_SMCR register).
6. Configure the trigger mode of PWMA (configure SMS=110 in the PWMA_SMCR register).

When a rising edge occurs on TI1, the two timers start counting synchronously, and the TIF bit is set.

Note: In this example, both timers are initialized (setting the UG bit) before starting, so they both start counting from 0, but the user can also insert an offset by modifying the counter register (PWMA_CNT), In this case, a delay will be inserted between the CK_PSC signal and the CNT_EN signal of PWMB.

The TI1 signal of PWMB triggers PWMB and PWMA

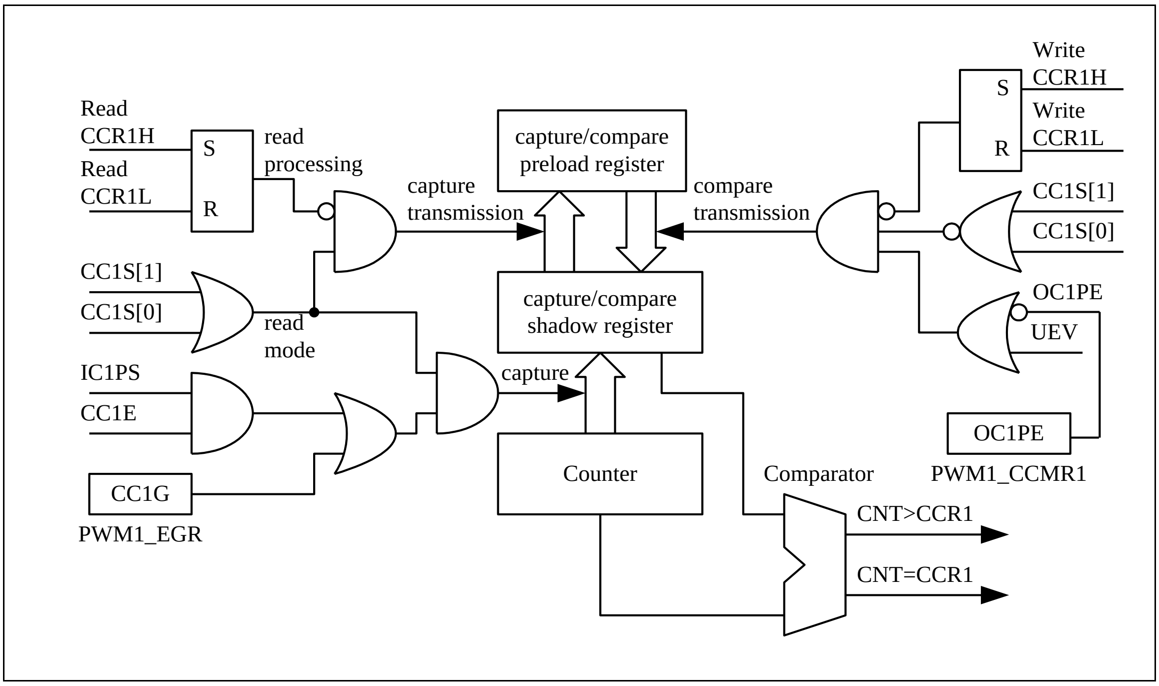


20.5 Capture/Compare Channel

PWM1P, PWM2P, PWM3P, PWM4P can be used as input capture, PWM1P/PWM1N, PWM2P/PWM2N, PWM3P/PWM3N, PWM4P/PWM4N can output comparison. This function can be realized by configuring the CCiS channel selection bit of the capture/compare channel mode register (PWMA_CCMRi), where i represents the number of channels from 1 to 4.

Each capture/compare channel is built around a capture/compare register (including shadow registers), including the input part of the capture (digital filtering, multiplexing and prescaler) and the output part (comparator and Output control).

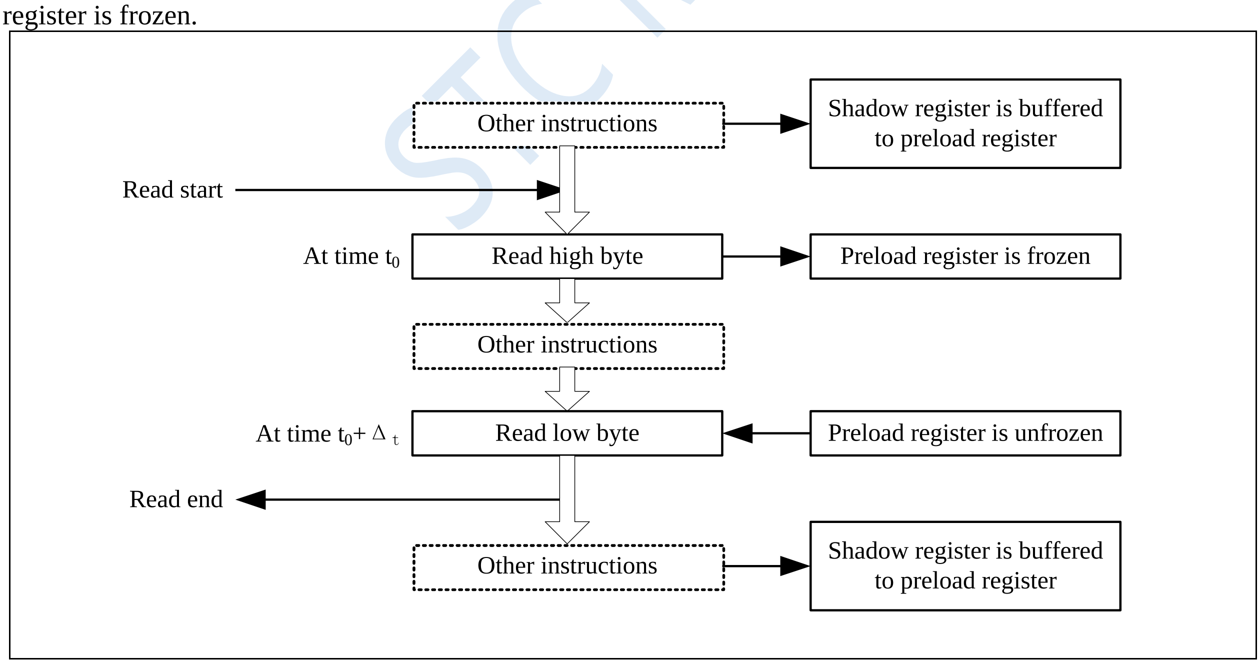
The main circuit of capture/compare channel 1 (other channels are similar to this)



The capture/compare module consists of a preload register and a shadow register. The read and write process only operates the preload register. In capture mode, the capture occurs on the shadow register and then copied to the preload register. In the compare mode, the contents of the preload register are copied to the shadow register, and then the contents of the shadow register are compared with the counter.

When the channel is configured in output mode, the PWMA_CCRi register can be accessed at any time.

When the channel is configured as input mode, the read operation of the PWMA_CCRi register is similar to the read operation of the counter. When the capture occurs, the contents of the counter are captured to the PWMA_CCRi shadow register, and then copied to the preload register. During the read operation, the preload register is frozen.


Other instructions
Shadow register is buffered to preload register
Read start
At time t_0
Read high byte
Preload register is frozen
Other instructions
At time t_0+ Δ t
Read low byte
Preload register is unfrozen
Read end
Other instructions
Shadow register is buffered to preload register

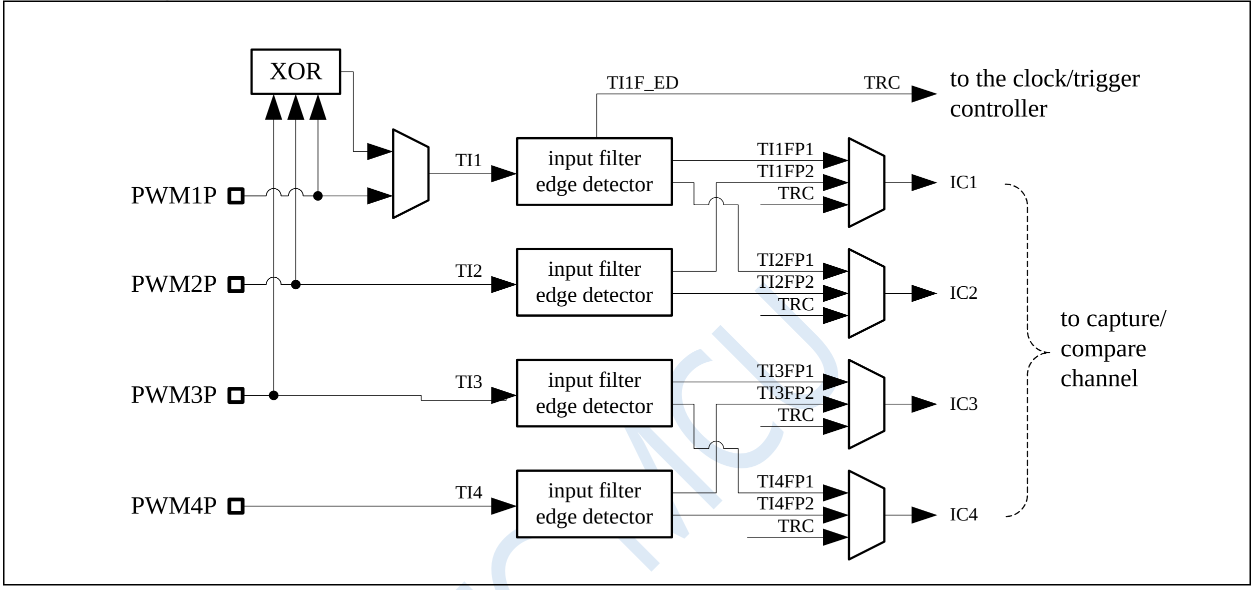
The figure above describes the read operation flow of the 16-bit CCRi register. The data buffered will remain unchanged until the end of the read flow. After the entire reading process is over, if only the PWMA_CCRiL register is read, the low bit of the counter value is returned. If the upper data is read after reading the lower data, the same lower data will not be returned.

20.5.1 Writing process of 16-bit PWMA_CCRi register

The writing operation of the 16-bit PWMA_CCRi register is completed through the preload register. Two instructions must be used to complete the entire process, one instruction corresponds to one byte. The upper byte must be written first. When writing the upper byte, the shadow register update is prohibited until the write operation of the lower byte is completed.

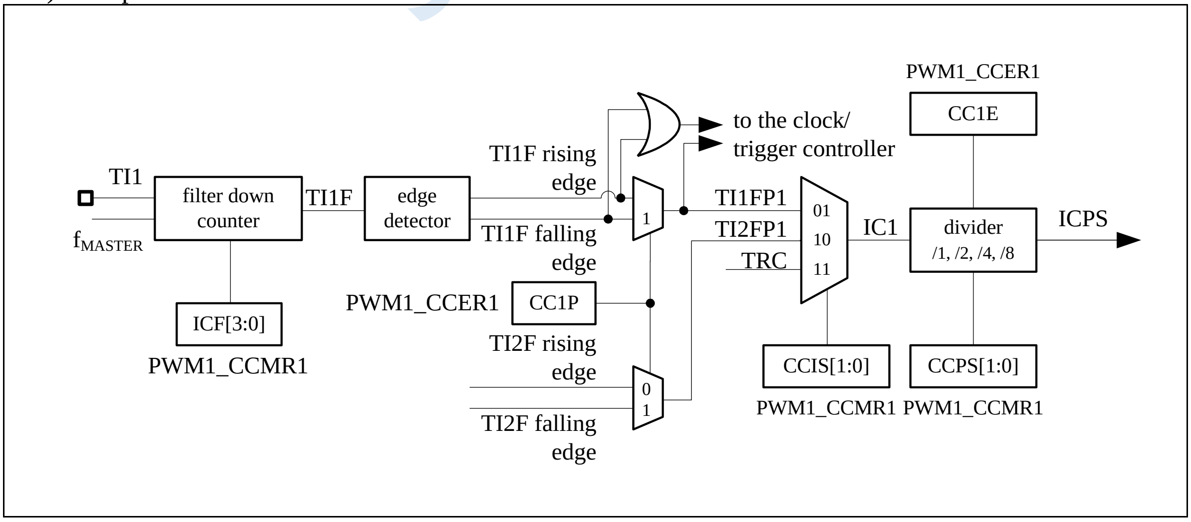
20.5.2 Input Module

Block diagram of the input module



XOR
TI1F_ED
TRC
to the clock/trigger controller
PWM1P
TI1
input filter edge detector
TI1FP1
TI1FP2
TRC
IC1
PWM2P
TI2
input filter edge detector
TI2FP1
TI2FP2
TRC
IC2
to capture/ compare channel
PWM3P
TI3
input filter edge detector
TI3FP1
TI3FP2
TRC
IC3
PWM4P
TI4
input filter edge detector
TI4FP1
TI4FP2
TRC
IC4

As shown in the figure, the input part samples the corresponding TIx input signal and generates a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx), which can be used as an input trigger for the trigger mode controller or as a capture control. This signal enters the capture register (ICxPS) after prescaler.



20.5.3 Input Capture Mode

In the input capture mode, when the corresponding edge on the ICi signal is detected, the current value of the counter is latched into the capture/compare register (PWMA_CCRx). When a capture event occurs, the corresponding CCiIF flag (PWMA_SR register) is set. If the CCiIE bit of the PWMA_IER register is set, that is, the interrupt is enabled, an interrupt request will be generated. If the CCiIF flag is already high when the capture event occurs, the repeated capture flag CCiOF (PWMA_SR2 register) is set. Writing CCiIF=0 or reading the captured data stored in the PWMA_CCRiL register can clear CCiIF. Write CCiOF=0 to clear CCiOF.

Capture on rising edge of PWM input signal

The following example shows how to capture the counter value to the PWMA_CCR1 register at the rising edge of TI1 input. The steps are as follows:

1. Select a valid input terminal and set CC1S=01 in the PWMA_CCMR1 register. At this time, the channel is configured as an input, and the PWMA_CCR1 register becomes read-only.
2. According to the characteristics of the input signal TIi, the filter time of the corresponding input filter can be set by configuring the IC1F bit in the PWMA_CCMR1 register. Assuming that the input signal jitters within a maximum of 5 clock cycles, we need to configure the filter bandwidth to be longer than 5 clock cycles; therefore, we can sample 8 times continuously to confirm the last real edge transition at TI1, that is, in the PWMA_CCMR1 register Write IC1F=0011 in the middle. At this time, only if 8 consecutively sampled TI1 signals valid (sampling frequency is f_{MASTER}).
3. Select the effective conversion edge of the TI1 channel and write CC1P=0 (rising edge) in the PWMA_CCER1 register.
4. Configure the input prescaler. In this example, we want the capture to occur at every valid level transition moment, so the prescaler is disabled (write IC1PS=00 in the PWMA_CCMR1 register).
5. Set CC1E of the PWMA_CCER1 register to 1, allowing the value of the counter to be captured into the capture register.
6. If necessary, enable related interrupt requests by setting the CC1IE bit in the PWMA_IER register.

When an input capture occurs:

- When a valid level conversion occurs, the value of the counter is transferred to the PWMA_CCR1 register.
- The CC1IF flag is set. When at least 2 consecutive captures have occurred and CC1IF has not been cleared, CC1OF is also set.
- If the CC1IE bit is set, an interrupt will be generated.

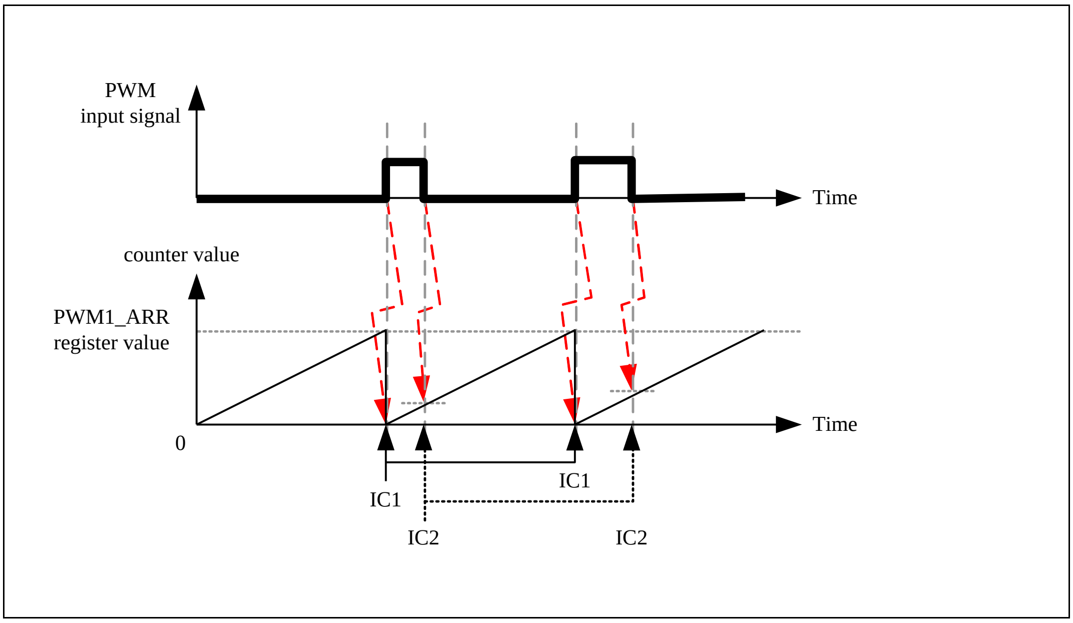
In order to handle the capture overflow event (CC1OF bit), it is recommended to read the data before reading the repeated capture flag. This is to avoid losing the repeated capture information that may be generated after the capture overflow flag is read and before the data is read.

Note: Setting the corresponding CCiG bit in the PWMA_EGR register can generate an input capture interrupt by software.

PWM input signal measurement

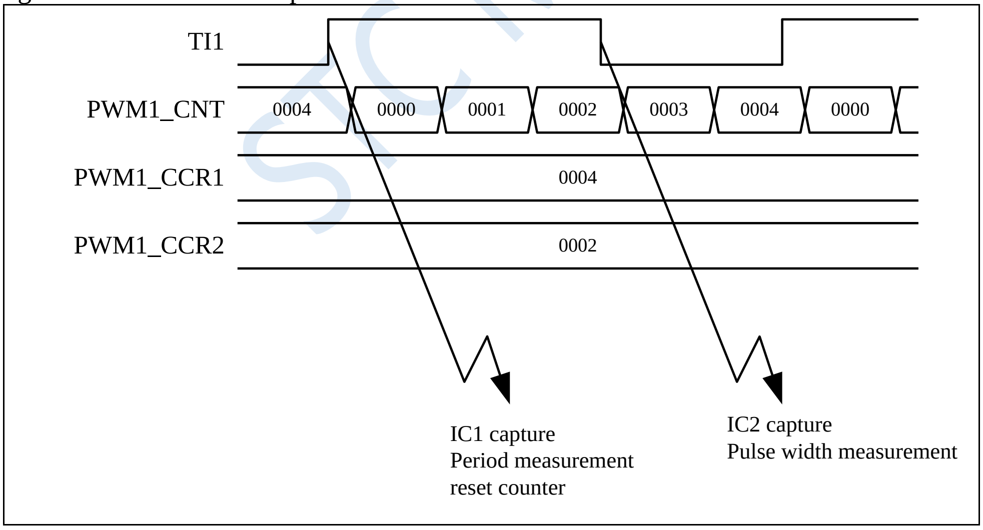
This mode is a special case of the input capture mode, except for the following differences, the operation is the same as the input capture mode:

- Two ICi signals are mapped to the same TIi input.
- The valid edges of the two ICI signals have opposite polarities.
- One of the TIiFP signals is used as the trigger input signal, and the trigger mode controller is configured to reset the trigger mode.



For example, you can measure the period (PWMA_CCR1 register) and duty cycle (PWMA_CCR2 register) of the PWM signal input on TI1 in the following way.
1. Select the effective input of PWMA_CCR1: set CC1S=01 in the PWMA_CCMR1 register (select TI1FP1).
2. Select the valid polarity of TI1FP1: set CC1P=0 (valid at rising edge).
3. Select the valid input of PWMA_CCR2: set CC2S=10 in the PWMA_CCMR2 register (select TI1FP2).
4. Select the valid polarity of TI1FP2 (capture data to PWMA_CCR2): set CC2P=1 (falling edge valid).
5. Select a valid trigger input signal: set TS=101 in the PWMA_SMCR register (select TI1FP1).
6. Configure the trigger mode controller to reset trigger mode: set SMS=100 in PWMA_SMCR.
7. Enable capture: set CC1E=1 and CC2E=1 in the PWMA_CCER1 register.

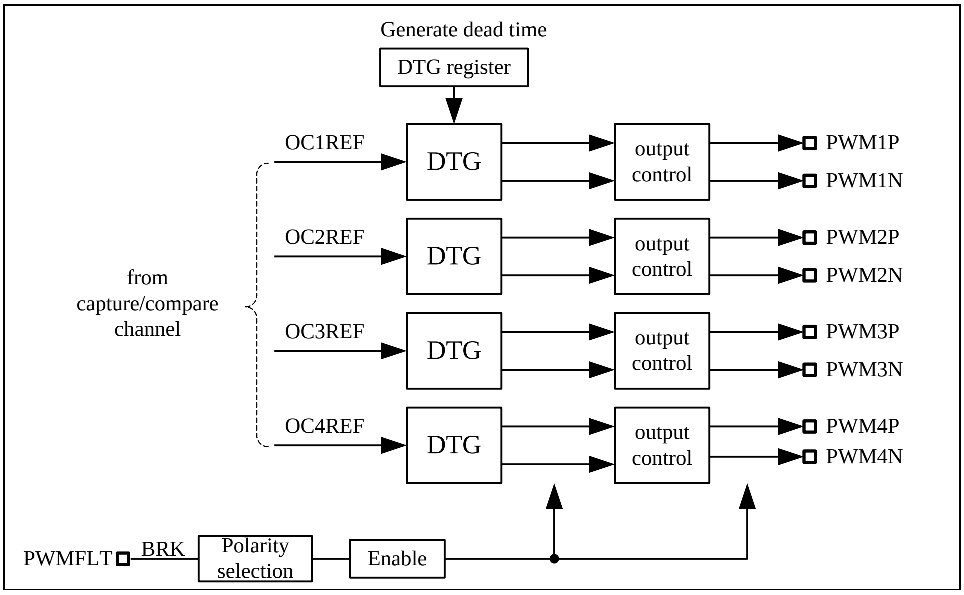
PWM input signal measurement example



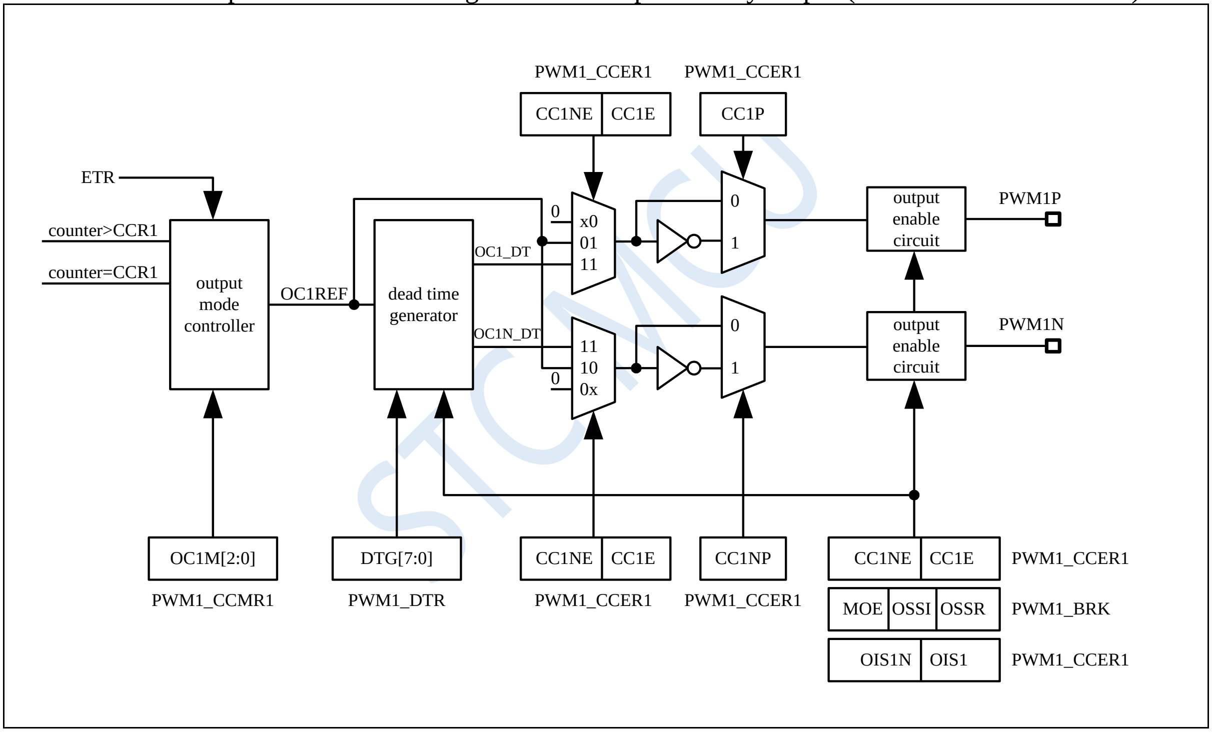
20.5.4 Output Module

The output module will generate an intermediate waveform used as a reference, called OCIREF (high effective). The processing of the brake function and polarity is at the end of the module.

Output module block diagram



Channel 1 detailed output module block diagram with complementary output (similar to other channels)



20.5.5 Forced output mode

In the output mode, the output comparison signal can be directly forced to a high or low state by software, instead of relying on the comparison result between the output comparison register and the counter.

Set OCIM=101 in the PWMA_CCMRi register to force the OCiREF signal to be low.

Set OCIM=100 in PWMA_CCMRi register to force the OCiREF signal to be low.

Whether the output of OCi/OCiN is high or low depends on the CCiP/CCiNP polarity flag.

In this mode, the comparison between the PWMA_CCRi shadow register and the counter is still in progress, the corresponding flag will be modified, and the corresponding interrupt will still be generated.

20.5.6 Output Compare Mode

This mode is used to control an output waveform or indicate that a given period of time has been reached. When the counter matches the content of the capture/compare register, the following operations are performed:

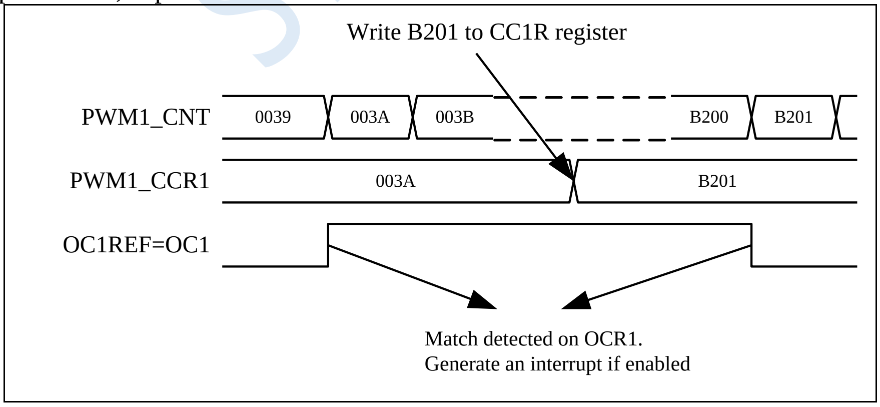
- According to different output comparison modes, the corresponding OCI output signal:
  - Keep unchanged (OCiM=000)
  - Set to effective level (OCiM=001)
  - Set to invalid level (OCiM=010)
  - Flip (OCiM=011)
- Set the flag bit in the interrupt status register (CCiIF bit in the PWMA_SR1 register).
- If the corresponding interrupt enable bit (CCiIE bit in the PWMA_IER register) is set, an interrupt is generated.

The OPiM bit of the PWMA_CCMRi register is used to select the output compare mode, and the CCiP bit of the PWMA_CCMRi register is used to select the valid and invalid level polarity. The OPiPE bit of the PWMA_CCMRi register is used to select whether the PWMA_CCRi register needs to use the preload register. In the output compare mode, the update event UEV has no effect on the OCIREF and OCI outputs. The time accuracy is one counting period of the counter. The output compare mode can also be used to output a single pulse.

Configuration steps of output comparison mode:
1. Select the counter clock (internal, external or prescaler).
2. Write the corresponding data into the PWMA_ARR and PWMA_CCRi registers.
3. To generate an interrupt request, set the CCiIE bit.
4. Steps to select output mode:
    1. Set OCIM=011, when the counter matches CCRi, flip the output of OCIM pin
    2. Set OPiPE = 0, disable preload register
    3. Set CCiP = 0, select high level as active level
    4. Set CCiE = 1, enable output
    5. Set the CEN bit of the PWMA_CR1 register to start the counter

The PWMA_CCRi register can be updated by software at any time to control the output waveform, provided that the preload register is not used (OCiPE=0), otherwise the shadow register of PWMA_CCRi can only be updated when the next update event occurs.

Output compare mode, flip OC1



20.5.7 PWM Mode

Pulse width modulation (PWM) mode can generate a signal whose frequency is determined by the PWMA_ARR register and the duty cycle is determined by the PWMA_CCRi register.

Write 110 (PWM mode 1) or 111 (PWM mode 2) in the OCIM bit in the PWMA_CCMRi register to independently set each OCI output channel to generate a PWM. The OPiPE bit of the PWMA_CCMRi register must be set to enable the corresponding preload register, and the ARPE bit of the PWMA_CR1 register can also be set to enable the preload register for auto-reload (in up-counting mode or central symmetric mode).

Since the preload register can only be transferred to the shadow register when an update event occurs, all registers must be initialized by setting the UG bit of the PWMA_EGR register before the counter starts counting.

The polarity of OCi can be set by software in the CCiP bit in the PWMA_CCERi register, and it can be set as active high or active low. The output enable of OCi is controlled by the combination of CCiE, MOE, OISi, OSSR and OSSI bits in the PWMA_CCERi and PWMA_BKR registers.

In PWM mode (mode 1 or mode 2), PWMA_CNT and PWMA_CCRi are always being compared (according to the counting direction of the counter) to determine whether it meets PWMA_CCRi ≤ PWMA_CNT or PWMA_CNT≤PWMA_CCRi.

According to the state of the CMS bit field in the PWMA_CR1 register, the timer can generate edge-aligned PWM signals or center-aligned PWM signals.

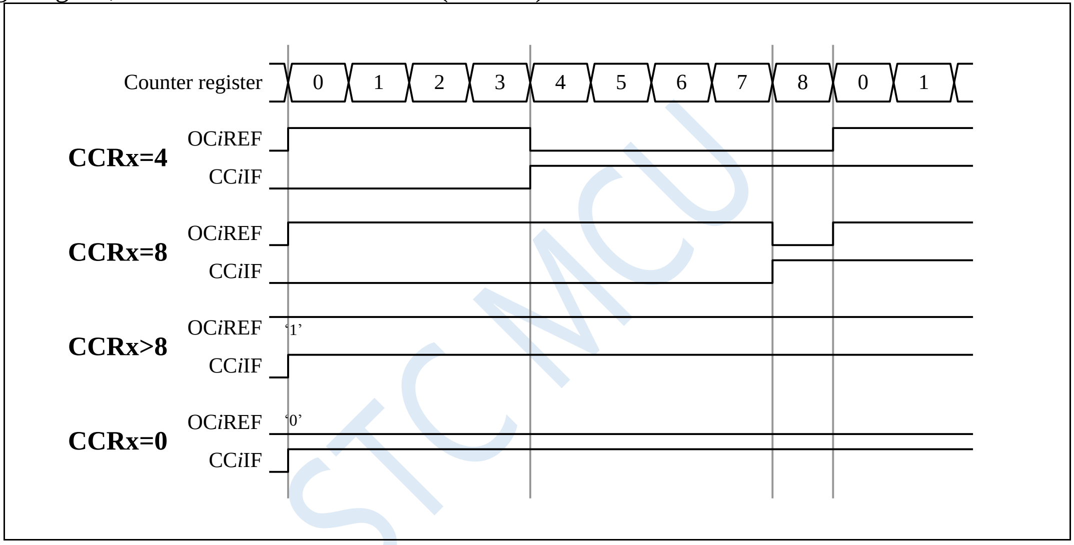
PWM edge alignment mode

Up counting configuration

When the DIR bit in the PWMA_CR1 register is 0, count up is executed.

The following is an example of PWM mode 1. When PWMA_CNT<PWMA_CCRi, the PWM reference signal OCiREF is high, otherwise it is low. If the comparison value in PWMA_CCRi is greater than the auto-reload value (PWMA_ARR), then OCiREF remains high. If the comparison value is 0, then OCiREF remains low.

Edge aligned, waveform of PWM mode 1 (ARR=8)



Countdown configuration

When the DIR bit of the PWMA_CR1 register is 1, the down count is executed.

In PWM mode 1, when PWMA_CNT>PWMA_CCRi, the reference signal OCiREF is low, otherwise it is high. If the comparison value in PWMA_CCRi is greater than the auto-reload value in PWMA_ARR, then OCiREF remains high. In this mode, a PWM waveform with a duty cycle of 0% cannot be generated.

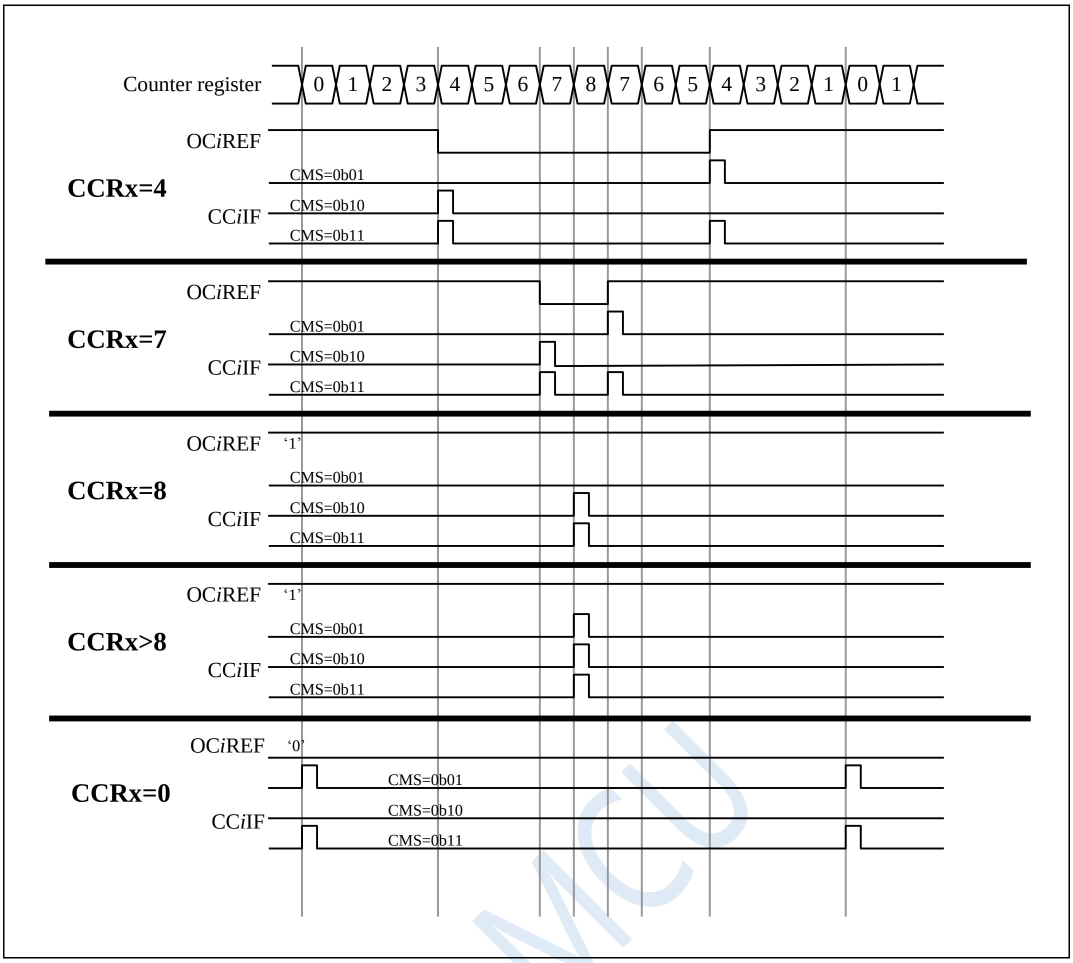
PWM center aligned mode

When the CMS bit in the PWMA_CR1 register is not '00', it is the center-aligned mode (all other configurations have the same effect on the OCiREF/OCi signal).

According to different CMS bit settings, the compare flag can be set when the counter counts up, counts down, or counts up and down. The counting direction bit (DIR) in the PWMA_CR1 register is updated by hardware. Do not modify it by software.

Some examples of center-aligned PWM waveforms are given below:

- PWMA_ARR=8
- PWM mode 1
- The flag bit is set in the following three situations:
  - — Only when the counter counts down (CMS=01)
  - — Only when the counter is counting up (CMS=10)
  - — When the counter is counting up and down (CMS=11)
- Center-aligned PWM waveform (ARR=8)



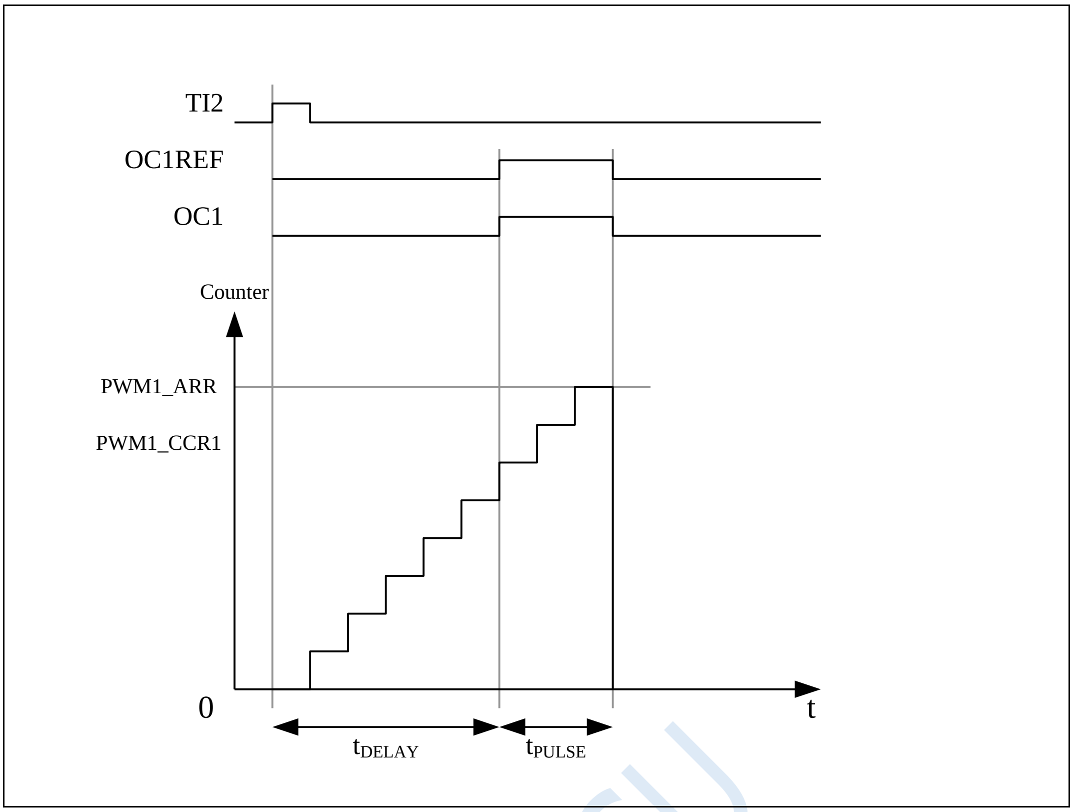
Single pulse mode

Single pulse mode (OPM) is a special case of many of the aforementioned modes. This mode allows the counter to respond to a stimulus and generate a pulse with a controllable pulse width after a programmable delay.

The counter can be started by the clock/trigger controller, and the waveform can be generated in output comparison mode or PWM mode. Setting the OPM bit of the PWMA_CR1 register will select the single pulse mode, and the counter will automatically stop at the next update event UEV. Only when the comparison value is different from the initial value of the counter can a pulse be generated. Before starting (when the timer is waiting to be triggered), the following must be configured:

- Up counting mode: counter CNT <CCRi ≤ ARR,
- Counting down mode: counter CNT> CCRi.

Single pulse mode legend



For example, delay tDELAY after detecting a rising edge on the TI2 input pin, and generate a positive pulse of tPULSE width on OC1: (assuming IC2 is used as the trigger source for triggering channel 1)

- Set CC2S=01 in PWMA_CCMR2 register to map IC2 to TI2.
- Set CC2P=0 in PWMA_CCER1 register to enable IC2 to detect the rising edge.
- Set TS=110 in the PWMA_SMCR register to make IC2 the trigger source (TRGI) of the clock/trigger controller.
- Set SMS=110 (trigger mode) in PWMA_SMCR register, IC2 is used to start the counter. The OPM waveform is determined by the value written in the compare register (the clock frequency and counter prescaler should be considered).
- tDELAY is defined by the value in the PWMA_CCR1 register.
- tPULSE is defined by the difference between the autoload value and the comparison value (PWMA_ARR – PWMA_CCR1).
- Suppose that a waveform from 0 to 1 is to be generated when a comparison match occurs, and a waveform from 1 to 0 is generated when the counter reaches the preload value. First, set the OCIM of the PWMA_CCMR1 register to 111 and enter PWM mode 2. According to You need to selectively set OC1PE=1 in the PWMA_CCMR1 register, set the ARPE in the PWMA_CR1 register, enable the preload register, then fill in the comparison value in the PWMA_CCR1 register, fill in the autoload value in the PWMA_ARR register, and set the UG bit to generate An update event, and then wait for an external trigger event on TI2.

  In this example, the DIR and CMS bits in the PWMA_CR1 register should be set low.

  Because only one pulse is needed, set OPM=1 in the PWMA_CR1 register to stop counting at the next update event (when the counter rolls over from the auto-load value to 0).

OCx fast enable (special case)

In the single pulse mode, the edge detection of the TIi input pin will set the CEN bit to start the counter, and then the comparison operation between the counter and the comparison value produces a single pulse output. But these operations require certain clock cycles, so it limits the minimum delay t_{DELAY} that can be obtained.

If you want to output the waveform with the minimum delay, you can set the OCiFE bit in the PWMA_CCMRi register. At this time, OCiREF (and OCx) is forced to respond directly to the excitation without relying on the comparison result. The output waveform is the same as the waveform when the comparison matches. OCiFE only works when the channel is configured in PWMA and PWMB mode.

Complementary output and dead zone insertion

PWMA can output two complementary signals, and can manage the instantaneous turn-off and turn-on of the output. This period is usually called the dead zone. The user should base on the connected output devices and their characteristics (delay of level conversion, power supply). Switch delay, etc.) to adjust the dead time.

Configure the CCiP and CCiNP bits in the PWMA_CCERi register to independently select the polarity (main output OCI or complementary output OCIN) for each output. The complementary signals OCI and OCIN are controlled by the following control bit combinations: CCiE and CCiNE bits in PWMA_CCERi register, MOE, OISi, OISiN, OSSI and OSSR bits in PWMA_BKR register. In particular, the dead zone control is activated when transitioning to the IDLE state (MOE drops to 0).

Setting the CCiE and CCiNE bits at the same time will insert the dead zone. If there is a brake circuit, the MOE bit should also be set. Each channel has an 8-bit dead zone generator.

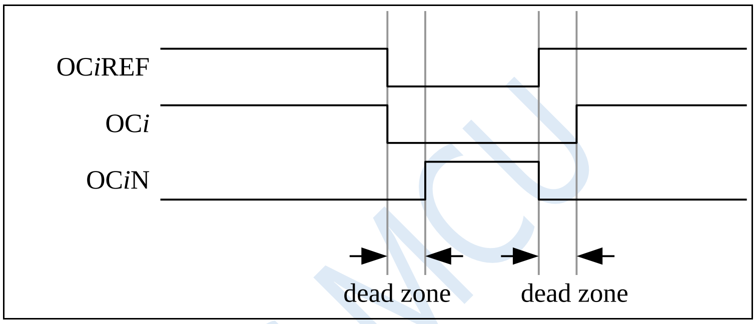
If OSi and OSiN are high effective:

- The OCi output signal is the same as OCiREF, except that its rising edge has a delay relative to the rising edge of OCiREF.
- The output signal of OCiN is opposite to OciREF, except that its rising edge has a delay relative to the falling edge of OCiREF.

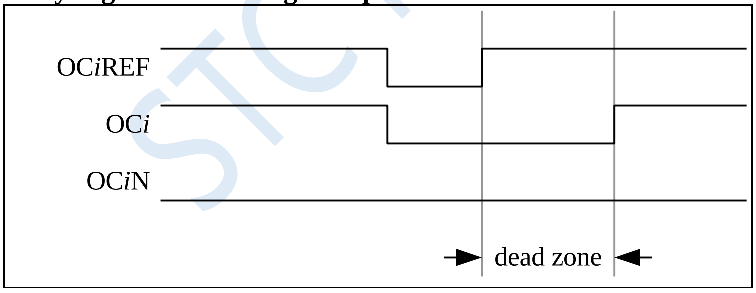
If the delay is greater than the currently effective output width (OCi or OCIN), the corresponding pulse will not be generated.

The following figures show the relationship between the output signal of the dead zone generator and the current reference signal OSiREF. (Assuming CCiP=0, CCiNP=0, MOE=1, CCiE=1 and CCiNE=1)

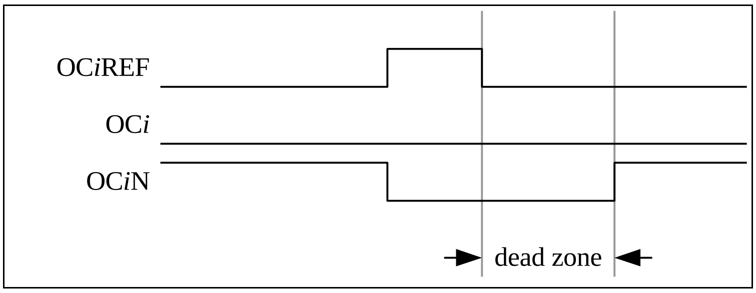
Complementary output with dead zone insertion



Dead zone waveform delay is greater than negative pulse



Dead zone waveform delay is greater than positive pulse



The dead time delay of each channel is the same, which is programmed and configured by the DTG bit in the PWMA_DTR register.

Redirect OCiREF to OCI or OCIN

In the output mode (forced output, output compare or PWM output), by configuring the CCiE and CCiNE bits of the PWMA_CCERi register, OCiREF can be redirected to the output of OCI or OCIN.

This function can send a special waveform (such as PWM or static active level) on an output when the complementary output is at an invalid level. Another function is to make the two outputs be at the inactive level at the same time, or at the active level at the same time (this is still the complementary output with dead zone).

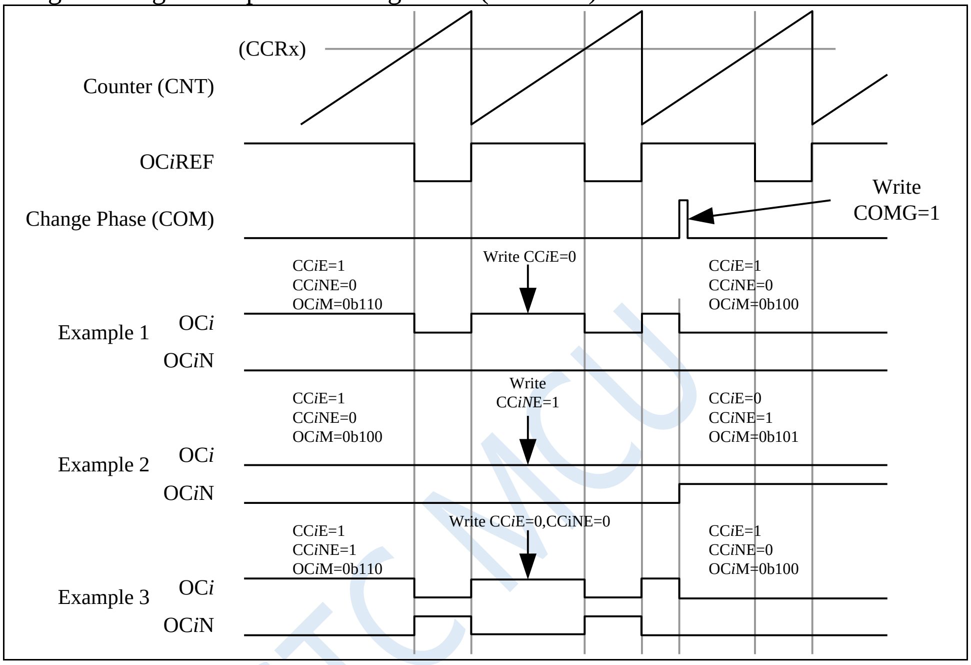
Note: When only OCiN is enabled (CCiE=0, CCiNE=1), it will not be inverted, and it will be effective immediately when OCiREF goes high. For example, if CCiNP=0, then OSiN=OCiREF. On the other hand, when both OCI and OCIN are enabled (CCiE=CCiNE=1), OCI is valid when OCiREF is high; on the contrary, when OCIREF is low, OCIN becomes valid.

Six-step PWM output for motor control

When complementary output is required on a channel, the preload bits are OPiM, CCiE and CCiNE. When a COM commutation event occurs, these preload bits are transferred to the shadow register bits. In this way, you can set the next step configuration in advance, and modify the configuration of all channels at the same time. COM can be generated by software by setting the COMG bit in the PWMA_EGR register, or by hardware on the rising edge of TRGI.

The following figure shows the output of OCx and OCxN in three different configurations when a COM event occurs.

Example of generating six-step PWM using COM (OSSR=1)



20.5.8 Using the brake function (PWMFLT)

The brake function is often used in motor control. When using the brake function, according to the corresponding control bits (MOE, OSSI and OSSR bits in the PWMA_BKR register), the output enable signal and invalid level will be modified.

After the system is reset, the brake circuit is disabled and the MOE bit is low. Setting the BKE bit in the PWMA_BKR register can enable the brake function. The polarity of the brake input signal can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time.

The falling edge of MOE can be asynchronous with respect to the clock module, so a resynchronization circuit is set between the actual signal (acting on the output) and the synchronization control bit (in the PWMA_BKR register). This resynchronization circuit creates a delay between the asynchronous signal and the synchronous signal. In particular, if you write MOE=1 when it is low, you must insert a delay (null instruction) before reading it to read the correct value. This is because asynchronous signals are written and synchronous signals are read.

When a brake occurs (the selected level appears at the brake input), the following actions are taken:
The MOE bit is cleared asynchronously, putting the output in an inactive state, an idle state or a reset state (selected by the OSSI bit). This feature is still valid when the MCU's oscillator is turned off.

- Once MOE=0, each output channel outputs the level set by the OISi bit of the PWMA_OISR register. If OSSI=0, the timer no longer controls the output enable signal, otherwise the output enable signal is always high.
- When using complementary output:
  - The output is first placed in a reset state, that is, an invalid state (depending on polarity). This is an asynchronous operation. This function is valid even when the timer does not have a clock.

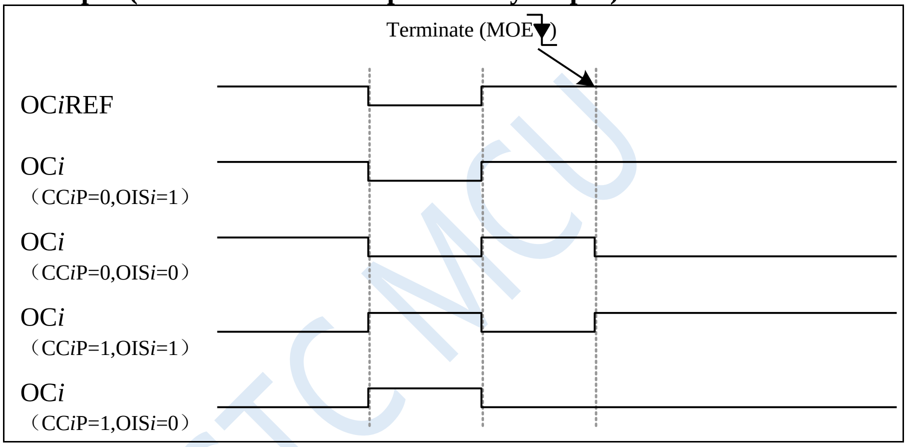
— If the timer clock still exists, the dead-band generator will take effect again and drive the output port according to the level indicated by the OISi and OISiN bits after the dead-band. Even in this case, OSi and OSiN cannot be driven to a valid level at the same time. Note: Because of the resynchronization of the MOE, the dead time is longer than usual (about 2 clock cycles).

- If the BIE bit of the PWMA_IER register is set, when the brake status flag (the BIF bit in the PWMA_SR1 register) is 1, an interrupt is generated.
- If the AOE bit in the PWMA_BKR register is set, the MOE bit will be automatically set in the next update event UEV. For example, this can be used for waveform control, otherwise, MOE will always remain low until it is set to 1 again. This feature can be used in safety, you can connect the brake input to the power-driven alarm output, thermal sensor or other safety devices.

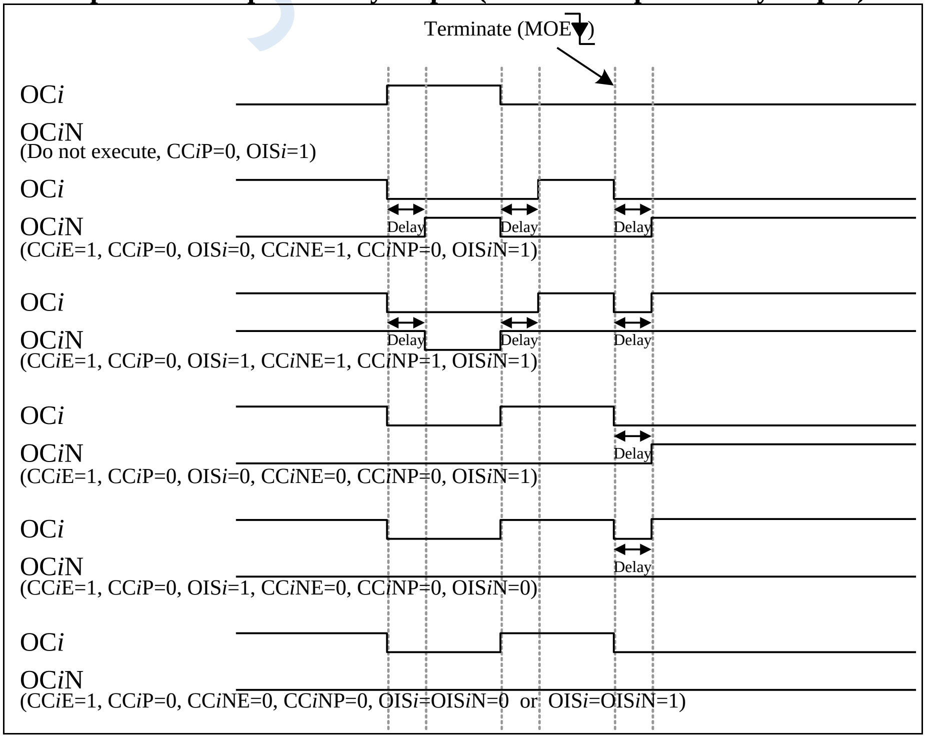
Note: The brake input is level effective. Therefore, when the brake input is valid, MOE cannot be set at the same time (automatically or through software). At the same time, the status flag BIF cannot be cleared.

The brake is generated by the BRK input, its effective polarity is programmable, and it is enabled or disabled by the BKE bit of the PWMA_BKR register. In addition to brake input and output management, write protection is also implemented in the brake circuit to ensure the safety of the application. It allows the user to freeze several configuration parameters (OCi polarity and state when disabled, OCIM configuration, brake enable and polarity). The user can select one of three levels of protection through the LOCK bit of the PWMA_BKR register. The LOCK bit field can only be modified once after the MCU is reset.

Brake response output (channel without complementary output)



Brake response output with complementary output (PWMA complementary output)


Terminate (MOE↓)
OCi
OCiN
(Do not execute, CCiP=0, OISi=1)
OCi
OCiN
Delay
Delay
Delay
(CCiE=1, CCiP=0, OISi=0, CCiNE=1, CCiNP=0, OISiN=1)
OCi
OCiN
Delay
Delay
Delay
(CCiE=1, CCiP=0, OISi=1, CCiNE=1, CCiNP=1, OISiN=1)
OCi
OCiN
Delay
(CCiE=1, CCiP=0, OISi=0, CCiNE=0, CCiNP=0, OISiN=1)
OCi
OCiN
Delay
(CCiE=1, CCiP=0, OISi=1, CCiNE=0, CCiNP=0, OISiN=0)
OCi
OCiN
(CCiE=1, CCiP=0, CCiNE=0, CCiNP=0, OISi=OISiN=0 or OISi=OISiN=1)

20.5.9 Clear the OCIREF signal when an external event occurs

For a given channel, a high level at the ETRF input terminal (set the corresponding OCiCE bit in the PWMA_CCMRi register to '1') can pull the OCiREF signal low, and the OCiREF signal will remain low until the next update event UEV occurs . This function can only be used in output comparison mode and PWM mode, but not in forced mode.

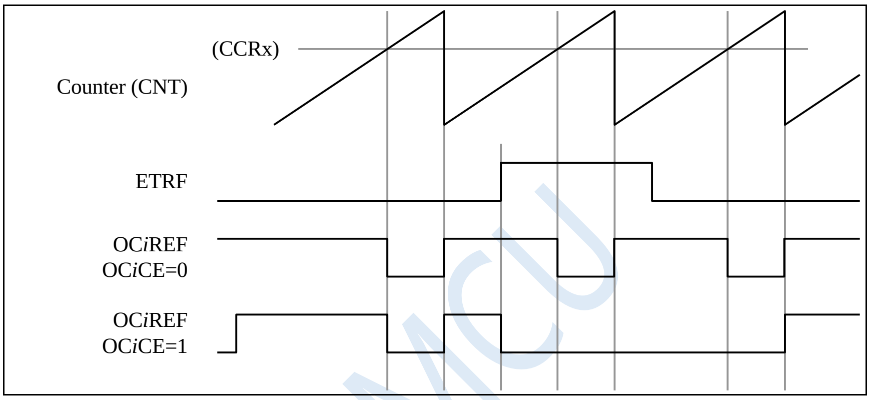
For example, the OCiREF signal can be connected to the output of a comparator to control the current. At this time, ETR must be configured as follows:

1. The external trigger prescaler must be turned off: ETPS[1:0]=00 in the PWMA_ETR register.
2. The external clock mode 2: ECE=0 in the PWMA_ETR register must be disabled.
3. External trigger polarity (ETP) and external trigger filter (ETF) can be configured as required.

The following figure shows the action of the OCiREF signal corresponding to different OCiCE values when the ETRF input goes high.

In this example, the timer PWMA is placed in PWM mode.

ETR clear OCIREF of PWMA



20.5.10 Encoder interface mode

The encoder interface mode is generally used for motor control.

The method to select the encoder interface mode is:

- If the counter only counts on the edge of TI2, set SMS=001 in the PWMA_SMCR register;
- If counting only on the edge of TI1, set SMS=010;
- If the counter counts on both TI1 and TI2 edges, set SMS=011.

By setting the CC1P and CC2P bits in the PWMA_CCER1 register, the TI1 and TI2 polarity can be selected; if necessary, the input filter can also be programmed.

Two inputs TI1 and TI2 are used as the interface of incremental encoder. Assuming that the counter has been started (CEN=1 in the PWMA_CR1 register), the counter counts every time TI1FP1 or TI2FP2 has a valid transition. TI1FP1 and TI2FP2 are the signals of TI1 and TI2 after passing the input filter and polarity control. If there is no filtering and polarity conversion, then TI1FP1=TI1, TI2FP2=TI2. According to the jump sequence of the two input signals, count pulses and direction signals are generated. According to the transition sequence of the two input signals, the counter counts up or down, and the hardware sets the DIR bit of the PWMA_CR1 register accordingly. Regardless of whether the counter counts on TI1, TI2, or both TI1 and TI2, a transition on either input (TI1 or TI2) will recalculate the DIR bit.

The encoder interface mode is basically equivalent to using an external clock with direction selection. This means that the counter only counts continuously from 0 to the autoload value of the PWMA_ARR register (according to the direction, either 0 to ARR or ARR to 0). Therefore, PWMA_ARR must be configured before counting. In this mode, the capturer, comparator, prescaler, repeat counter, trigger output feature, etc. still work as usual. Encoder mode and external clock mode 2 are not compatible, so they cannot be operated at the same time.

In the encoder interface mode, the counter is automatically modified according to the speed and direction of the incremental encoder, so the content of the counter always indicates the position of the encoder, and the counting direction corresponds to the direction of rotation of the connected sensor.

The following table lists all possible combinations (assuming that TI1 and TI2 do not change at the same time).

The relationship between counting direction and encoder signal

Active edge	Relative signal level (TI1FP1 corresponds to TI2, TI2FP2 corresponds to TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Count only at TI1	High	count down	count up	does not count	does not count
	Low	count up	count down	does not count	does not count
Count only at TI2	High	does not count	does not count	count up	count down
	Low	does not count	does not count	count down	count up
Count at TI1 and TI2	High	count down	count up	count up	count down
	Low	count up	count down	count down	count up

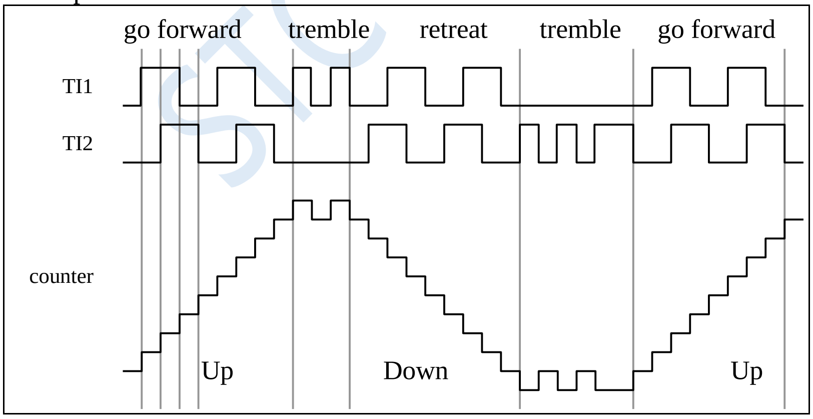
An external incremental encoder can be directly connected to the MCU without external interface logic. However, a comparator is generally used to edit

The differential output of the encoder is converted into a digital signal, which greatly increases the ability to resist noise interference. The third signal output by the encoder represents the mechanical zero point, which can be connected to an external interrupt input and trigger a counter reset.

The following is an example of counter operation, showing the count signal generation and direction control. It also shows how input jitter is suppressed when both edges are selected; jitter may occur when the sensor is close to a transition point. In this example, we assume the following configuration:

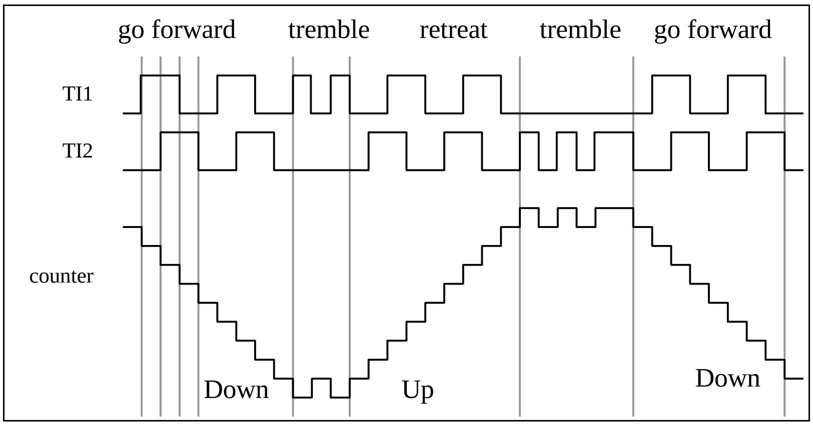
- CC1S=01 (PWMA_CCMR1 register, IC1FP1 is mapped to TI1)
- CC2S=01 (PWMA_CCMR2 register, IC2FP2 is mapped to TI2)
- CC1P=0 (PWMA_CCER1 register, IC1 is not inverted, IC1=TI1)
- CC2P=0 (PWMA_CCER1 register, IC2 is not inverted, IC2=TI2)
- SMS=011 (PWMA_SMCR register, all inputs are valid on rising and falling edges).
- CEN=1 (PWMA_CR1 register, counter enable)

Example of counter operation in encoder mode



The following figure shows the operation example of the counter when the polarity of IC1 is reversed (CC1P=1, other configurations are the same as the above example)

IC1 inverted encoder interface mode example



When the timer is configured in the encoder interface mode, it provides information about the current position of the sensor. Use another timer configured in capture mode to measure the interval between two encoder events to obtain dynamic information (speed, acceleration, deceleration). The encoder output indicating the mechanical zero point can be used for this purpose. According to the interval between two events, the counter can be read at a certain time interval. If possible, you can latch the value of the counter into the third input capture register (the capture signal must be periodic and can be generated by another timer).

20.6 Interrupt

PWMA/PWMB each have 8 interrupt request sources:

- The brake is interrupted
- Trigger interrupt
- COM event interrupt
- Input capture/output compare 4 interrupt
- Input capture/output compare 3 interrupt
- Input capture/output compare 2 interrupt
- Input capture/output compare 1 interrupt
- Update event interrupt (such as: counter overflow, underflow and initialization)

In order to use the interrupt feature, for each interrupt channel used, set the corresponding interrupt enable bit in the PWM_IER/PWMB_IER register: BIE, TIE, COMIE, CCiIE, UIE bit. By setting the corresponding bits in the PWMA_EGR/PWMB_EGR register, the above-mentioned interrupt sources can also be generated by software.

20.7 PWMA/PWMB registers description

20.7.1 Output Enable Registers (PWMx_ENO)

Symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_ENO	FEB1H	ENO4N	ENO4P	ENO3N	ENO3P	ENO2N	ENO2P	ENO1N	ENO1P
PWMB_ENO	FEB5H	-	ENO8P	-	ENO7P	-	ENO6P	-	ENO5P

ENO8P: PWM8 output control bit
    0: Disable PWM8 output
    1: Enable PWM8 output
ENO7P: PWM7 output control bit
    0: Disable PWM7 output
    1: Enable PWM7 output
ENO6P: PWM6 output control bit
    0: Disable PWM6 output

1: Enable PWM6 output
ENO5P: PWM5 output control bit
0: Disable PWM5 output
1: Enable PWM5 output
ENO4N: PWM4N output control bit
0: Disable PWM4N output
1: Enable PWM4N output
ENO4P: PWM4P output control bit
0: Disable PWM4P output
1: Enable PWM4P output
ENO3N: PWM3N output control bit
0: Disable PWM3N output
1: Enable PWM3N output
ENO3P: PWM3P output control bit
0: Disable PWM3P output
1: Enable PWM3P output
ENO2N: PWM2N output control bit
0: Disable PWM2N output
1: Enable PWM2N output
ENO2P: PWM2P output control bit
0: Disable PWM2P output
1: Enable PWM2P output
ENO1N: PWM1N output control bit
0: Disable PWM1N output
1: Enable PWM1N output
ENO1P: PWM1P output control bit
0: Disable PWM1P output
1: Enable PWM1P output

20.7.2 Output Additional Enable Registers (PWMx_IOAUX)

Symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_IOAUX	FEB3H	AUX4N	AUX4P	AUX3N	AUX3P	AUX2N	AUX2P	AUX1N	AUX1P
PWMB_IOAUX	FEB7H	-	AUX8P	-	AUX7P	-	AUX6P	-	AUX5P

AUX8P: PWM8 output additional control bit
0: PWM8 output is directly controlled by ENO8P
1: The output of PWM8 is controlled by ENO8P and PWMB_BKR
AUX7P: PWM7 output additional control bit
0: The output of PWM7 is directly controlled by ENO7P
1: The output of PWM7 is controlled by ENO7P and PWMB_BKR
AUX6P: PWM6 output additional control bit
0: The output of PWM6 is directly controlled by ENO6P
1: The output of PWM6 is jointly controlled by ENO6P and PWMB_BKR
AUX5P: PWM5 output additional control bit
0: The output of PWM5 is directly controlled by ENO5P
1: The output of PWM5 is controlled by ENO5P and PWMB_BKR
AUX4N: PWM4N output additional control bit
0: The output of PWM4N is directly controlled by ENO4N
1: The output of PWM4N is controlled by ENO4N and PWMA_BKR
AUX4P: PWM4P output additional control bit
0: The output of PWM4P is directly controlled by ENO4P
1: The output of PWM4P is jointly controlled by ENO4P and PWMA_BKR
AUX3N: PWM3N output additional control bit
0: The output of PWM3N is directly controlled by ENO3N
1: The output of PWM3N is controlled by ENO3N and PWMA_BKR
AUX3P: PWM3P output additional control bit

0: The output of PWM3P is directly controlled by ENO3P
1: The output of PWM3P is jointly controlled by ENO3P and PWMA_BKR

AUX2N: PWM2N output additional control bit
0: PWM2N output is directly controlled by ENO2N
1: The output of PWM2N is controlled by ENO2N and PWMA_BKR

AUX2P: PWM2P output additional control bit
0: PWM2P output is directly controlled by ENO2P
1: The output of PWM2P is jointly controlled by ENO2P and PWMA_BKR

AUX1N: PWM1N output additional control bit
0: PWM1N output is directly controlled by ENO1N
1: The output of PWM1N is jointly controlled by ENO1N and PWMA_BKR

AUX1P: PWM1P output additional control bit
0: The output of PWM1P is directly controlled by ENO1P
1: The output of PWM1P is jointly controlled by ENO1P and PWMA_BKR

20.7.3 Control Registers 1 (PWMx_CR1)

Symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_CR1	FEC0H	ARPEA	CMSA[1:0]		DIRA	OPMA	URSA	UDISA	CENA
PWMB_CR1	FEE0H	ARPEB	CMSB[1:0]		DIRB	OPMB	URSB	UDISB	CENB

ARPEn: Automatic preloading permission bit (n=A,B)
0: PWMn_ARR register is not buffered, it can be written directly
1: PWMn_ARR register is buffered by the preload buffer

CMSn[1:0]: select alignment mode (n=A,B)

CMSn[1:0]	Alignment mode	Description
00	Edge alignment mode	The counter counts up or down according to the direction bit (DIR)
01	Center alignment mode 1	The counter counts up and down alternately. The output compare interrupt flag bit of the channel configured as an output is only set when the counter counts down.
10	Center alignment mode2	The counter counts up and down alternately. The output compare interrupt flag bit of the channel configured as an output is only set when the counter is counting up.
11	Center alignment mode3	The counter counts up and down alternately. The output compare interrupt flag bit of the channel configured as output is set to 1 when the counter is counting up and down.

Note 1: When the counter is turned on (CEN=1), it is not allowed to switch from edge-aligned mode to center-aligned mode.

Note 2: In the center-aligned mode, the encoder mode (SMS=001, 010, 011) must be disabled.

DIRn: counting direction of the counter (n= A, B)
0: The counter counts up;
1: The counter counts down.
Note: When the counter is configured in center aligned mode or encoder mode, this bit is read-only.

OPMn: Single pulse mode (n= A, B)
0: When an update event occurs, the counter does not stop;
1: When the next update event occurs, the CEN bit is cleared and the counter stops.

URSn: Update request source (n= A, B)
0: If UDIS allows the generation of update events, any of the following events will generate an update interrupt:
− Register is updated (counter overflow/underflow)
− Software setting UG bit
− Updates generated by the clock/trigger controller

1: If UDIS allows the generation of update events, the update interrupt will only be generated when the following events occur, and UIF is set to 1:

− Register is updated (counter overflow/underflow)

UDISn: Update is prohibited (n= A, B)

0: Once the following events occur, an update (UEV) event occurs:

− Counter overflow/underflow

− Generate software update events

− Hardware reset generated by the clock/trigger mode controller The cached registers are loaded with their preload values.

1: No update event is generated, and the shadow registers (ARR, PSC, CCRx) maintain their values. If the UG bit is set or the clock/trigger controller issues a hardware reset, the counter and prescaler are reinitialized.

CENn: Allow counter (n= A, B)

0: disable the counter;

1: Enable the counter.

Note: The external clock, gating mode and encoder mode can only work after the CEN bit is set by the software. However, the trigger mode can automatically set the CEN bit by hardware.

20.7.4 Control Registers 2 (PWMx_CR2), and trigger ADC in real time

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_CR2	FEC1H	TI1S	MMSA[2:0]			-	COMSA	-	CCPCA
PWMB_CR2	FEE1H	TI5S	MMSB[2:0]			-	COMSB	-	CCPCB

TI1S: TI1 selection of the first group of PWM/PWMA

0: PWM1P input pin is connected to TI1 (digital filter input);

1: PWM1P, PWM2P and PWM3P pins are connected to TI1 of the first group of PWM after exclusive OR.

TI5S: TI5 selection of the second set of PWM/PWMB

0: PWM5 input pin is connected to TI5 (digital filter input);

1: PWM5, PWM6 and PWM7 pins are connected to TI5 of the second group of PWM after exclusive OR.

MMSA[2:0]: Main mode selection

MMSA[2:0]	Master mode	Description
000	Reset	The UG bit of the PWMA_EGR register is used as a trigger output (TRGO). If the trigger input (the clock/trigger controller is configured in reset mode) generates a reset, the signal on TRGO will have a delay relative to the actual reset
001	Enable	The counter enable signal is used as a trigger output (TRGO). It is used to start the ADC in order to control the enabling of the ADC within a period of time. The counter enable signal is generated by the logical OR of the CEN control bit and the trigger input signal in the gating mode. Unless the master/slave mode is selected, there will be a delay on TRGO when the counter enable signal is controlled by the trigger input. Note: When you need to use PWM to trigger ADC conversion, you need to set ADC_POWER, ADC_CHS and ADC_EPWMT in ADC_CONTR register first. When PWM generates TRGO internal signal, the system will automatically set ADC_START to start AD conversion. For detailed usage, please refer to the sample program. "Use PWM CEN to start PWMA timer and trigger ADC in real time"
010	Update	Update event is selected as trigger output (TRGO)
011	Comparison pulse	Once a capture or a comparison is successful, when the CC1IF flag is set to 1, the trigger output sends a positive pulse (TRGO)
100	Compare	OC1REF signal is used as trigger output (TRGO)
101	Compare	OC2REF signal is used as trigger output (TRGO)

110	Compare	OC3REF signal is used as trigger output (TRGO)
111	Compare	OC4REF signal is used as trigger output (TRGO)

MMSB[2:0]: Main mode selection

MMSB[2:0]	Master mode	Description
000	Reset	The UG bit of the PWMB_EGR register is used as a trigger output (TRGO). If the trigger input (the clock/trigger controller is configured in reset mode) generates a reset, the signal on TRGO will have a delay relative to the actual reset.
001	Enable	The counter enable signal is used as a trigger output (TRGO). It is used to start multiple PWMs in order to control to enable slave PWM for a period of time. The counter enable signal is generated by the logical OR of the CEN control bit and the trigger input signal in the gating mode. Unless the master/slave mode is selected, there will be a delay on TRGO when the counter enable signal is controlled by the trigger input.
010	Enable	Update event is selected as trigger output (TRGO)
011	Update	Once a capture occurs or a comparison is successful, when the CC5IF flag is set to 1, the trigger output sends a positive pulse (TRGO)
100	Comparison pulse	OC5REF signal is used as trigger output (TRGO)
101	Compare	OC6REF signal is used as trigger output (TRGO)
110	Compare	OC7REF signal is used as trigger output (TRGO)
111	Compare	OC8REF signal is used as trigger output (TRGO)

Note: Only the TRGO of the first set of PWM can be used to trigger the start of ADC

Note: Only the TRGO of the second group of PWM can be used for the ITR2 of the first group of PWM

COMSn: Update control selection of capture/compare control bit (n=A, B)

0: When CCPCn=1, these control bits are only updated when the COMG bit is set to 1.

1: When CCPCn=1, these control bits will be updated only when the COMG bit is 1 or TRGI has a rising edge

CCPCn: capture/compare preload control bit (n=A,B)

0: CCIE, CCINE, CCiP, CCiNP and OCIM bits are not preloaded

1: CCIE, CCINE, CCiP, CCiNP and OCIM bits are pre-loaded; after setting this bit, they will only be updated after setting the COMG bit.

Note: This bit only works on channels with complementary outputs.

20.7.5 Slave Mode Control Registers (PWMx_SMCR)

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_SMCR	FEC2H	MSMA	TSA[2:0]			-	SMSA[2:0]		
PWMB_SMCR	FEE2H	MSMB	TSB[2:0]			-	SMSB[2:0]		

MSMn: Master/Slave mode (n= A, B)

0: No effect

1: The event on the trigger input (TRGI) is delayed to allow perfect synchronization between PWMn and its slave PWM (via TRGO)

TSA[2:0]: trigger source selection

TSA[2:0]	Trigger source
000	-
001	-
010	Internal trigger ITR2
011	-
100	TI1 edge detector (TI1F_ED)
101	Filtered timer input 1 (TI1FP1)

110	Filtered timer input 2 (TI2FP2)
111	External trigger input (ETRF)

TSB[2:0]: trigger source selection

TSB[2:0]	Trigger source
000	-
001	-
010	-
011	-
100	TI5 edge detector (TI5F_ED)
101	Filtered timer input 1 (TI5FP5)
110	Filtered timer input 2 (TI5FP6)
111	External trigger input (ETRF)

Note: These bits can only be changed when SMS=000, to avoid false edge detection when changing.

SMSA[2:0]: clock/trigger/slave mode selection

SMSA[2:0]	function	Description
000	Internal clock mode	If CEN=1, the prescaler is directly driven by the internal clock
001	Encoder mode 1	According to the level of TI1FP1, the counter counts up/down on the edge of TI2FP2
010	Encoder mode 2	According to the level of TI2FP2, the counter counts up/down on the edge of TI1FP1
011	Encoder mode 3	According to the level of another input, the counter counts up/down on the edge of TI1FP1 and TI2FP2
100	Reset mode	Reinitialize the counter on the rising edge of the selected trigger input (TRGI) and generate a signal to update the register
101	Gating mode	When the trigger input (TRGI) is high, the counter clock is turned on. Once the trigger input goes low, the counter stops (but does not reset). The start and stop of the counter are controlled)
110	Trigger mode	The counter is started (but not reset) on the rising edge of the trigger input TRGI, only the start of the counter is controlled
111	External clock mode 1	The rising edge of the selected trigger input (TRGI) drives the counter. Note: If TI1F_ED is selected as the trigger input (TS=100), do not use the gated mode. This is because TI1F_ED only outputs a pulse every time TI1F changes, but the gate control mode is to check the level of the trigger input

SMSB[2:0]: clock/trigger/slave mode selection

SMSB[2:0]	function	Description
000	Internal clock mode	If CEN=1, the prescaler is directly driven by the internal clock
001	Encoder mode 1	According to the level of TI5FP5, the counter counts up/down on the edge of TI6FP6
010	Encoder mode 2	According to the level of TI6FP6, the counter counts up/down on the edge of TI5FP5
011	Encoder mode 3	According to the level of another input, the counter counts up/down on the edge of TI5FP5 and TI6FP6
100	Reset mode	Reinitialize the counter on the rising edge of the selected trigger input (TRGI) and generate a signal to update the register
101	Gating mode	When the trigger input (TRGI) is high, the counter clock is turned on. Once the trigger input goes low, the counter stops (but does not reset). The start and stop of the counter are controlled)
110	Trigger mode	The counter is started (but not reset) on the rising edge of the trigger input TRGI, only the start of the counter is controlled
111	External clock mode 1	The rising edge of the selected trigger input (TRGI) drives the counter. Note: If TI5F_ED is selected as the trigger input (TS=100), do not use the gated mode. This is because TI5F_ED only outputs a pulse every time TI5F changes, but the gate control mode is to check the level of the trigger input.

20.7.6 External Trigger Registers (PWMx_ETR)

Symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_ETR	FEC3H	ETP1	ECEA	ETPSA[1:0]		ETFA[3:0]			
PWMB_ETR	FEE3H	ETP2	ECEB	ETPSB[1:0]		ETFB[3:0]			

ETPn: Polarity of external trigger ETR (n= A, B)

0: high level or rising edge valid

1: Low level or falling edge valid

ECEn: External clock enable (n= A, B)

0: Disable external clock mode 2

1: Enable external clock mode 2, the counter clock is the valid edge of ETRF.

Note 1: The effect of setting ECE to 1 is the same as selecting external clock mode 1 connecting TRGI to ETRF (in the PWMn_SMCR register, SMS=111, TS=111).

Note 2: External clock mode 2 can be used simultaneously with the following modes: trigger standard mode; trigger reset mode; trigger gated mode. However, at this time, TRGI must not be connected to ETRF (in the PWMn_SMCR register, TS cannot be 111).

Note 3: External clock mode 1 and external clock mode 2 are enabled at the same time, and the external clock input is ETRF.

ETPSn: The frequency of the external trigger signal EPRP of the external trigger prescaler cannot exceed fMASTER/4. You can use a prescaler to reduce low ETRP frequency, it is very useful when EPRP frequency is high: (n=A,B)

00: prescaler is off

01: EPRP frequency/2

02: EPRP frequency/4

03: EPRP frequency/8

ETFn[3:0]: External trigger filter selection, this bit field defines the ETRP sampling frequency and digital filter length. (N=A,B)

ETFn[3:0]	Number of clocks	ETF[3:0]	Number of clocks
0000	1	1000	48
0001	2	1001	64
0010	4	1010	80
0011	8	1011	96
0100	12	1100	128
0101	16	1101	160
0110	24	1110	192
0111	32	1111	256

20.7.7 Interrupt Enable Registers (PWMx_IER)

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_IER	FEC4H	BIEA	TIEA	COMIEA	CC4IE	CC3IE	CC2IE	CC1IE	UIEA
PWMB_IER	FEE4H	BIEB	TIEB	COMIEB	CC8IE	CC7IE	CC6IE	CC5IE	UIEB

BIEn: Allow brake interruption (n= A, B)

0: Prohibit brake interruption;

1: Allow brake interruption.

TIE: trigger interrupt enable (n= A, B)
0: Prohibit triggering interrupts;
1: Enable trigger interrupt.
COMIE: Allow COM interruption (n= A, B)
0: Disable COM interrupt;
1: Enable COM interrupt.
CCnIE: Allow capture/compare n interrupts (n=1,2,3,4,5,6,7,8)
0: Disable capture/compare n interrupt;
1: Allow capture/compare n interrupts.
UIEn: Allow update interruption (n= A, B)
0: Update interrupt is prohibited;
1: Allow update interruption.

20.7.8 Status Registers 1 (PWMx_SR1)

symbol	sddress	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_SR1	FEC5H	BIFA	TIFA	COMIFA	CC4IF	CC3IF	CC2IF	CC1IF	UIFA
PWMB_SR1	FEE5H	BIFB	TIFB	COMIFB	CC8IF	CC7IF	CC6IF	CC5IF	UIFB

BIFn: Brake interruption mark. Once the brake input is valid, the bit is set by the hardware. If the brake input is invalid, this bit can be cleared by software. (N=A,B)
0: No brake event occurs
1: Valid level detected on brake input
TIFn: Trigger interrupt mark. The bit is set by hardware when a trigger event occurs. Cleared by software. (N=A,B)
0: No trigger event is generated
1: Trigger interrupt waiting for response
COMIFn: COM interrupt flag. Once a COM event occurs, this bit is set by hardware. Cleared by software. (N=A,B)
0: No COM event is generated
1: COM interrupt waiting for response
CC8IF: capture/compare 8 interrupt flag, refer to CC1IF description
CC7IF: Capture/Compare 7 interrupt flag, refer to CC1IF description
CC6IF: Capture/compare 6 interrupt flag, refer to CC1IF description
CC5IF: Capture/compare 5 interrupt flag, refer to CC1IF description
CC4IF: capture/compare 4 interrupt flag, refer to CC1IF description
CC3IF: Capture/Compare 3 interrupt flag, refer to CC1IF description
CC2IF: Capture/Compare 2 interrupt flag, refer to CC1IF description
CC1IF: Capture/Compare 1 interrupt flag.

If channel CC1 is configured as output mode:

This bit is set by hardware when the counter value matches the comparison value, except in the center symmetric mode. It is cleared by software.
0: No match occurred;
1: The value of PWMA_CNT matches the value of PWMA_CCR1.

Note: In the center symmetry mode, when the counter value is 0, count up, when the counter value is ARR, count down (it counts up from 0 to ARR-1, and then counts down from ARR to 1). Therefore, for all SMS bit values, these two values are not marked. However, if CCR1>ARR, when CNT reaches the ARR value, CC1IF is set.

If channel CC1 is configured as input mode:

This bit is set by hardware when a capture event occurs, and it is cleared by software or cleared by reading PWMA_CCR1L.
0: No input capture is generated
1: The counter value has been captured to PWMA_CCR1

UIFn: Update interrupt flag This bit is set by hardware when an update event is generated. It is cleared by software. (N=A,B)
0: No update event is generated
1: Update event waiting for response. This bit is set by hardware when the register is updated

− If UDIS=0 in the PWMn_CR1 register, when the counter overflows or underflows
− If UDIS=0 and URS=0 in the PWMn_CR1 register, when the UG bit of the PWMn_EGR register is set by the software to reinitialize the counter CNT
− If UDIS=0 and URS=0 in the PWMn_CR1 register, when the counter CNT is reinitialized by a trigger event

20.7.9 Status Registers 2 (PWMx_SR2)

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_SR2	FEC6H	-	-	-	CC4OF	CC3OF	CC2OF	CC1OF	-
PWMB_SR2	FEE6H	-	-	-	CC8OF	CC7OF	CC6OF	CC5OF	-

CC8OF: capture/compare 8 repeat capture flag. See CC1OF description.
CC7OF: Capture/Compare 7 repeat capture flag. See CC1OF description.
CC6OF: Capture/Compare 6 Repeat Capture Mark. See CC1OF description.
CC5OF: Capture/Compare 5 repeat capture flag. See CC1OF description.
CC4OF: Capture/Compare 4 repeat capture flag. See CC1OF description.
CC3OF: Capture/Compare 3 repeat capture flag. See CC1OF description.
CC2OF: Capture/Compare 2 repeat capture flag. See CC1OF description.
CC1OF: Capture/Compare 1 repeat capture flag. This flag can be set by hardware only when the corresponding channel is configured as input capture. Write 0 to clear this bit.
0: No repeated capture is generated;
1: When the counter value is captured to the PWMA_CCR1 register, the state of CC1IF is already 1.

20.7.10 Event Generation Registers (PWMx_EGR)

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_EGR	FEC7H	BGA	TGA	COMGA	CC4G	CC3G	CC2G	CC1G	UGA
PWMB_EGR	FEE7H	BGB	TGB	COMGB	CC8G	CC7G	CC6G	CC5G	UGB

BGn: A brake event is generated. This bit is set by software to generate a brake event and is automatically cleared by hardware (n = A, B)
0: No action
1: Generate a brake event. At this moment, MOE=0, BIF=1, if the corresponding interrupt is turned on (BIE=1), the corresponding interrupt will be generated
TGn: Generate a trigger event. This bit is set by software to generate a trigger event and is automatically cleared by hardware (n = A, B)
0: No action
1: TIF=1, if the corresponding interrupt is turned on (TIE=1), the corresponding interrupt will be generated
COMGn: Capture/compare events and generate control updates. This bit is set by software and cleared by hardware automatically (n= A, B)
0: No action
1: CCPC=1, allow to update CCIE, CCINE, CCiP, CCiNP, OCIM bits.
Note: This bit is only valid for channels with complementary outputs
CC8G: Generate capture/compare 8 events. Refer to CC1G description
CC7G: Generate capture/compare 7 events. Refer to CC1G description
CC6G: Generate capture/compare 6 events. Refer to CC1G description
CC5G: Generate capture/compare 5 events. Refer to CC1G description
CC4G: Generate capture/compare 4 events. Refer to CC1G description
CC3G: Generate capture/compare 3 events. Refer to CC1G description
CC2G: Generate capture/compare 2 event. Refer to CC1G description
CC1G: Generate capture/compare 1 event. Generate capture/compare 1 event. This bit is set by software to generate a capture/compare event and is automatically cleared by hardware.

0: No action;

1: Generate a capture/compare event on channel CC1.

If channel CC1 is configured as output: set CC1IF=1, if the corresponding interrupt is turned on, the corresponding interrupt will be generated.

If channel CC1 is configured as input: the current counter value is captured to the PWMA_CCR1 register, set CC1IF=1, if the corresponding interrupt is enabled, the corresponding interrupt will be generated. If CC1IF is already 1, set CC1OF=1.

UGn: Update event generated. This bit is set by software and cleared by hardware automatically. (N=A,B)

0: No action;

1: Reinitialize the counter and generate an update event.

Note that the counter of the prescaler is also cleared to 0 (but the prescaler coefficient remains unchanged). If in center symmetric mode or DIR=0 (counting up), the counter will be cleared; if DIR=1 (counting down), the counter will take the value of PWMn_ARR..

20.7.11 Capture/Compare Mode Register 1 (PWMx_CCMR1)

The channel can be used to capture input mode or compare output mode, and the direction of the channel is defined by the corresponding CCnS bit. The functions of other bits of this register are different in input and output modes. OCxx describes the function of the channel in output mode, and ICxx describes the function of the channel in input mode. Therefore, it must be noted that the function of the same bit in output mode and input mode is different.

Channel is configured to compare output mode

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_CCMR1	FEC8H	OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
PWMB_CCMR1	FEE8H	OC5CE	OC5M[2:0]			OC5PE	OC5FE	CC5S[1:0]	

OCnCE: Output compare n clear enable. This bit is used to enable the use of external events on the PWMETI pin to clear the output signal of channel n (OCnREF) (n=1,5)

0: OCnREF is not affected by ETRF input;

1: Once the ETRF input high level is detected, OCnREF=0.

OCnM[2:0]: Output compare n mode. These 3 bits define the action of the output reference signal OCnREF, and OCnREF determines the value of OCn. OCnREF is active high, and the active level of OCn depends on the CCnP bit. (N=1,5)

OCnM[2:0]	mode	Description
000	freeze	The comparison between PWMn_CCR1 and PWMn_CNT has no effect on OCnREF
001	Set channel n when matching Output is valid level	When PWMn_CCR1=PWMn_CNT, OCnREF output high
010	Set channel n when matching Output is invalid level	When PWMn_CCR1=PWMn_CNT, OCnREF output low
011	Flip	When PWMn_CCR1=PWMn_CNT, flip OCnREF
100	Forced to invalid level	Force OCnREF to low
101	Forced to active level	Force OCnREF to be high
110	PWM mode 1	When counting up, when PWMn_CNT<PWMn_CCR1, OCnREF outputs high, otherwise OCnREF outputs low When counting down, when PWMn_CNT>PWMn_CCR1 OCnREF output low, otherwise OCnREF output high
111	PWM mode 2	When counting up, when PWMn_CNT<PWMn_CCR1OCnREF outputs low, otherwise OCnREF outputs high When counting down, when PWMn_CNT>PWMn_CCR1 OCnREF output high, otherwise OCnREF output low

Note 1: Once the LOCK level is set to 3 (LOCK bit in the PWMn_BKR register) and CCnS=00 (the channel is configured as an output), this bit cannot be modified.

Note 2: In PWM mode 1 or PWM mode 2, the OCnREF level only changes when the comparison result is changed or when switching from freeze mode to PWM mode in output comparison mode.

Note 3: On channels with complementary outputs, these bits are pre-loaded. If the CCPC of the PWMn_CR2 register is 1, the OCM bit will take a new value from the preload bit only when a COM event occurs.

OCnPE: output compare n preload enable (n=1,5)

0: Disable the preload function of the PWMn_CCR1 register, and the PWMn_CCR1 register can be written at any time, and the newly written value will take effect immediately.

1: Enable the preload function of the PWMn_CCR1 register. Read and write operations only operate on the preload register. The preload value of PWMn_CCR1 is loaded into the current register when the update event arrives.

Note 1: Once the LOCK level is set to 3 (LOCK bit in the PWMn_BKR register) and CCnS=00 (the channel is configured as an output), this bit cannot be modified.

Note 2: In order to operate correctly, the preload function must be enabled in the PWM mode. But in single pulse mode (OPM=1 in PWMn_CR1 register), it is not necessary.

OCnFE: Output compare n fast enable. This bit is used to speed up the response of the CC output to the trigger input event. (N=1,5)

0: According to the value of the counter and CCRn, CCn operates normally, even if the trigger is turned on. When the flip-flop input has a valid edge, the minimum delay for activating the CCn output is 5 clock cycles.

1: The effect of the valid edge input to the trigger is like a comparison match. Therefore, OC is set to the comparison level regardless of the comparison result. The delay between the valid edge of the sampling flip-flop and the CC1 output is shortened to 3 clock cycles. OCFE only works when the channel is configured in PWMA or PWMB mode.

CC1S[1:0]: Capture/compare 1 selection. These two bits define the direction of the channel (input/output), and the selection of input pins

CC5S[1:0]	direction	Input pin
00	Output	
01	Input	IC1 is mapped on TI1FP1
10	Input	IC1 is mapped on TI2FP1
11	Input	IC1 is mapped on TRC. This mode only works when the internal trigger input is selected (selected by the TS bit of the PWMA_SMCR register)

CC5S[1:0]: Capture/compare 5 selection. These two bits define the direction of the channel (input/output), and the selection of input pins

CC5S[1:0]	direction	Input pin
00	Output	
01	Input	IC5 is mapped on TI5FP5
10	Input	IC5 is mapped on TI6FP5
11	Input	IC5 is mapped on TRC. This mode only works when the internal trigger input is selected (selected by the TS bit of the PWM5_SMCR register)

Note: CC1S is writable only when the channel is closed (CC1E=0 in PWMA_CCER1 register).

Note: CC5S is writable only when the channel is closed (CC5E=0 in PWM5_CCER1 register).

Channel is configured to capture input mode

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_CCMR1	FEC8H	IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
PWMB_CCMR1	FEE8H	IC5F[3:0]				IC5PSC[1:0]		CC5S[1:0]	

ICnF[3:0]: Input capture n filter selection, this bit field defines the sampling frequency of TIn and the length of the digital filter. (N=1,5)

ICnF[3:0]	Number of clocks	ICnF[3:0]	Number of clocks
0000	1	1000	48
0001	2	1001	64
0010	4	1010	80

0011	8	1011	96
0100	12	1100	128
0101	16	1101	160
0110	24	1110	192
0111	32	1111	256

Note: Even for channels with complementary outputs, this bit field is not preloaded, and the value of CCPC (PWMn_CR2 register) will not be considered

ICnPSC[1:0]: Input/capture n prescaler. These two bits define the prescaler coefficient of CCn input (IC1). (N=1,5)

00: No prescaler, every edge detected on the capture input port triggers a capture

01: Trigger a capture every 2 events

10: Trigger a capture every 4 events

11: Trigger a capture every 8 events

CC1S[1:0]: Capture/compare 1 selection.

These two bits define the direction of the channel (input/output), and the selection of input pins

CC1S[1:0]	direction	input pin
00	Output	
01	Input	IC1 is mapped on TI1FP1
10	Input	IC1 is mapped on TI2FP1
11	Input	IC1 is mapped on TRC. This mode only works when the internal trigger input is selected (selected by the TS bit of the PWMA_SMCR register)

CC5S[1:0]: Capture/compare 5 selection.

These two bits define the direction of the channel (input/output), and the selection of input pins

CC5S[1:0]	direction	input pin
00	Output	
01	Input	IC5 is mapped on TI5FP5
10	Input	IC5 is mapped on TI6FP5
11	Input	IC5 is mapped on TRC. This mode only works when the internal trigger input is selected (selected by the TS bit of the PWM5_SMCR register)

Note: CC1S is writable only when the channel is closed (CC1E=0 in PWMA_CCER1 register).

Note: CC5S is writable only when the channel is closed (CC5E=0 in PWM5_CCER1 register).

20.7.12 Capture/compare mode registers 2 (PWMx_CCMR2)

Channel is configured to compare output mode

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_CCMR2	FEC9H	OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]	
PWMB_CCMR2	FEE9H	OC6CE	OC6M[2:0]			OC6PE	OC6FE	CC6S[1:0]	

OCnCE: Output compare n clear enable. This bit is used to enable the use of external events on the PWMETI pin to clear the output signal of channel n (OCnREF) (n=2,6)

0: OCnREF is not affected by ETRF input;

1: Once the ETRF input high level is detected, OCnREF=0.

OCnM[2:0]: Output compare 2 mode, refer to OC1M. (N=2,6)

OCnPE: Output compare 2 preload enable, refer to OP1PE. (N=2,6)

CC2S[1:0]: Capture/compare 2 selection.

These two bits define the direction of the channel (input/output), and the selection of input pins

CC2S[1:0]	direction	Input pin
00	Output	
01	Input	IC2 is mapped on TI2FP2
10	Input	IC2 is mapped on TI1FP2
11	Input	IC2 is mapped on TRC

CC6S[1:0]: Capture/compare 6 selection.

These two bits define the direction of the channel (input/output), and the selection of input pins

CC6S[1:0]	direction	Input pin
00	Output	
01	Input	IC6 is mapped on TI6FP6
10	Input	IC6 is mapped on TI5FP6
11	Input	IC6 is mapped on TRC

Channel is configured to capture input mode

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_CCMR2	FEC9H	IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]	
PWMB_CCMR2	FEE9H	IC6F[3:0]				IC6PSC[1:0]		CC6S[1:0]	

ICnF[3:0]: Input capture n filter selection, refer to IC1F. (N=2,6)

ICnPSC[1:0]: Input/capture n prescaler, refer to IC1PSC. (N=2,6)

CC2S[1:0]: Capture/compare 2 selection.

These two bits define the direction of the channel (input/output), and the selection of input pins

CC2S[1:0]	direction	Input pin
00	Output	
01	Input	IC2 is mapped on TI2FP2
10	Input	IC2 is mapped on TI1FP2
11	Input	IC2 is mapped on TRC。

CC6S[1:0]: Capture/compare 6 selection.

These two bits define the direction of the channel (input/output), and the selection of input pins

CC6S[1:0]	direction	Input pin
00	Output	
01	Input	IC6 is mapped on TI6FP6
10	Input	IC6 is mapped on TI5FP6
11	Input	IC6 is mapped on TRC

20.7.13 Capture/Compare Mode Registers 3 (PWMx_CCMR3)

Channel is configured to compare output mode

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_CCMR3	FECAH	OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
PWMB_CCMR3	FEEAH	OC7CE	OC7M[2:0]			OC7PE	OC7FE	CC7S[1:0]	

OCnCE: Output compare n clear enable. This bit is used to enable the use of external events on the PWMETI pin to clear the output signal of channel n (OCnREF) (n=3,7)

0: OCnREF is not affected by ETRF input;

1: Once the ETRF input high level is detected, OCnREF=0.

OCnM[2:0]: Output compare 3 mode, refer to OC1M. (N=3,7) OCnPE: Output compare 3 preload enable, refer to OP1PE. (N=3,7)

CC3S[1:0]: Capture/compare 3 selection. These two bits define the direction of the channel (input/output), and the selection of input pins

CC3S[1:0]	direction	Input pin
00	Output	
01	Input	IC3 is mapped on TI3FP3
10	Input	IC3 is mapped on TI4FP3
11	Input	IC3 is mapped on TRC

CC7S[1:0]: Capture/compare 7 selection. These two bits define the direction of the channel (input/output), and the selection of input pins

CC7S[1:0]	direction	Input pin
00	Output	
01	Input	IC7 is mapped on TI7FP7
10	Input	IC7 is mapped on TI8FP7
11	Input	IC7 is mapped on TRC

Channel is configured to capture input mode

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_CCMR3	FECAH	IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
PWMB_CCMR3	FEEAH	IC7F[3:0]				IC7PSC[1:0]		CC7S[1:0]	

ICnF[3:0]: Input capture n filter selection, refer to IC1F. (N=3,7)

ICnPSC[1:0]: Input/capture n prescaler, refer to IC1PSC. (N=3,7)

CC3S[1:0]: Capture/compare 3 selection. These two bits define the direction of the channel (input/output), and the selection of input pins

CC3S[1:0]	direction	Input pin
00	Output	
01	Input	IC3 is mapped on TI3FP3
10	Input	IC3 is mapped on TI4FP3
11	Input	IC3 is mapped on TRC

CC7S[1:0]: Capture/compare 7 selection. These two bits define the direction of the channel (input/output), and the selection of input pins

CC7S[1:0]	direction	Input pin
00	Output	
01	Input	IC7 is mapped on TI7FP7
10	Input	IC7 is mapped on TI8FP7
11	Input	IC7 is mapped on TRC

20.7.14 Capture/Compare Mode Registers 4 (PWMx_CCMR4)

Channel is configured to compare output mode

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_CCMR4	FECBH	OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]	
PWMB_CCMR4	FEEBH	OC8CE	OC8M[2:0]			OC8PE	OC8FE	CC8S[1:0]	

OCnCE: Output compare n clear enable. This bit is used to enable the use of external events on the PWMETI pin to clear the output signal of channel n (OCnREF) (n=4,8)

0: OCnREF is not affected by ETRF input;

1: Once the ETRF input high level is detected, OCnREF=0.

OCnM[2:0]: Output compare n mode, refer to OC1M. (N=4,8) OCnPE: output compare n preload enable, refer to OP1PE. (N=4,8)

CC4S[1:0]: Capture/compare 4 selection. These two bits define the direction of the channel (input/output), and the selection of input pins

CC4S[1:0]	direction	Input pin
00	Output	
01	Input	IC4 is mapped on TI4FP4
10	Input	IC4 is mapped on TI3FP4
11	Input	IC4 is mapped on TRC。

CC8S[1:0]: Capture/compare 8 selection. These two bits define the direction of the channel (input/output), and the selection of input pins

CC8S[1:0]	direction	Input pin
00	Output	
01	Input	IC8 is mapped on TI8FP8
10	Input	IC8 is mapped on TI7FP8
11	Input	IC8 is mapped on TRC。

Channel is configured to capture input mode

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_CCMR4	FECBH	IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]	
PWMB_CCMR4	FEEBH	IC8F[3:0]				IC8PSC[1:0]		CC8S[1:0]	

ICnF[3:0]: Input capture n filter selection, refer to IC1F. (N=4,8)

ICnPSC[1:0]: Input/capture n prescaler, refer to IC1PSC. (N=4,8)

CC4S[1:0]: Capture/compare 4 selection. These two bits define the direction of the channel (input/output), and the selection of input pins

CC4S[1:0]	direction	Input pin
00	Output	
01	Input	IC4 is mapped on TI4FP4
10	Input	IC4 is mapped on TI3FP4
11	Input	IC4 is mapped on TRC。

CC8S[1:0]: Capture/compare 8 selection. These two bits define the direction of the channel (input/output), and the selection of input pins

CC8S[1:0]	direction	Input pin
00	Output	
01	Input	IC8 is mapped on TI8FP8
10	Input	IC8 is mapped on TI7FP8
11	Input	IC8 is mapped on TRC。

20.7.15 Capture/Compare Enable Registers 1 (PWMx_CCER1)

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_CCER1	FECCH	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
PWMB_CCER1	FEECH	-	-	CC6P	CC6E	-	-	CC5P	CC5E

CC6P: OC6 input capture/compare output polarity. Reference CC1P

CC6E: OC6 input capture/compare output enable. Reference CC1E

CC5P: OC5 input capture/compare output polarity. Reference CC1P

CC5E: OC5 input capture/compare output enable. Reference CC1E

CC2NP: OC2N compare output polarity. Reference CC1NP

CC2NE: OC2N compare output enable. Reference CC1NE

CC2P: OC2 input capture/compare output polarity. Reference CC1P

CC2E: OC2 input capture/compare output enable. Reference CC1E

CC1NP: OC1N compare output polarity

0: High level is active;

1: Low level is active.

Note 1: Once the LOCK level (LOCK bit in the PWMA_BKR register) is set to 3 or 2 and CC1S=00 (the channel is configured as an output), this bit cannot be modified.

Note 2: For channels with complementary outputs, this bit is pre-loaded. If CCPC=1 (PWMA_CR2 register), only when a COM event occurs, the CC1NP bit will take a new value from the preload bit.

CC1NE: OC1N compare output enable

0: Turn off the comparison output.

1: Turn on the comparison output. The output level depends on the value of the MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Note: For channels with complementary outputs, this bit is preloaded. If CCPC=1 (PWMA_CR2 register), only when a COM event occurs, the CC1NE bit will take a new value from the preload bit.

CC1P: OC1 input capture/compare output polarity

CC1 channel is configured as output:

0: high level is active

1: low level is active

CC1 channel is configured as input or capture:

0: Capture occurs on the rising edge of TI1F or TI2F;

1: Capture occurs on the falling edge of TI1F or TI2F.

CC1E: OC1 input capture/compare output enable

0: Turn off input capture/comparison output;

1: Turn on input capture/comparison output.

Note 1: Once the LOCK level (LOCK bit in PWMA_BKR register) is set to 3 or 2, this bit cannot be modified.

Note 2: For channels with complementary outputs, this bit is pre-loaded. If CCPC=1 (PWMA_CR2 register), the CC1P bit will take a new value from the preload bit only when a COM event occurs.

Control bits of complementary output channels OCi and OCIN with brake function

Control bit					output status	
MOE	OSSI	OSSR	CCiE	CCiNE	OCi output status	OCiN output status
1	X	0	0	0	Output prohibited	Output prohibited
		0	0	1	Output prohibited	OCiREF with polarity
		0	1	0	OCiREF with polarity	Output prohibited
		0	1	1	OCiREF with polarity and dead zone	Reverse OCiREF with polarity and dead zone
		1	0	0	Output prohibited	Output prohibited

		1	0	1	Disabled (The output is enabled and invalid level) OCi=CCiP	OCiREF with polarity
		1	1	0	OCiREF with polarity	Disabled (The output is enabled and invalid level) OCi=CCiP
		1	1	1	OCiREF with polarity and dead zone	Reverse OCiREF with polarity and dead zone
0	0	X	X	X	Output prohibited	
	1				Closed state (output enabled and at an invalid level) asynchronously: OCI=CCiP, OCIN=CCiNP; Then, if the clock exists: OCI=OISi, OCIN=OISiN after a dead time, assuming that OISi and OISiN are not both Corresponds to the effective level of OCI and OCIN。	

Note: The status of the external I/O pins connected to the complementary OCi and OCiN channels depends on the status of the OCi and OCiN channels and the GPIO register.

20.7.16 Capture/Compare Enable Registers 2 (PWMx_CCER2)

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_CCER2	FECDH	CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E
PWMB_CCER2	FEEDH	-	-	CC8P	CC8E	-	-	CC7P	CC7E

CC8P: OC8 input capture/compare output polarity. Reference CC1P
CC8E: OC8 input capture/compare output enable. Reference CC1E
CC7P: OC7 input capture/compare output polarity. Reference CC1P
CC7E: OC7 input capture/compare output enable. Reference CC1E
CC4NP: OC4N compare output polarity. Reference CC1NP
CC4NE: OC4N compare output enable. Reference CC1NE
CC4P: OC4 input capture/compare output polarity. Reference CC1P
CC4E: OC4 input capture/compare output enable. Reference CC1E
CC3NP: OC3N compare output polarity. Reference CC1NP
CC3NE: OC3N compare output enable. Reference CC1NE
CC3P: OC3 input capture/compare output polarity. Reference CC1P
CC3E: OC3 input capture/compare output enable. Reference CC1E

20.7.17 Counters upper 8 bits (PWMx_CNTRH)

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_CNTRH	FECEH	CNT1[15:8]							
PWMB_CNTRH	FEEEH	CNT2[15:8]							

CNTn[15:8]: the high 8-bit value of the counter (n= A, B)

20.7.18 Counters low 8 bits (PWMx_CNTRL)

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_CNTRL	FECFH	CNT1[7:0]							
PWMB_CNTRL	FEEFH	CNT2[7:0]							

CNTn[7:0]: the high 8-bit value of the counter (n= A, B)

20.7.19 Prescalers high 8 bits (PWMx_PSCRH), output frequency calculation formula

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_PSCRH	FED0H	PSC1[15:8]							
PWMB_PSCRH	FEF0H	PSC2[15:8]							

PSCn[15:8]: The high 8-bit value of the prescaler. (N=A,B)

The prescaler is used to divide the frequency of CK_PSC. The counter clock frequency (fCK_CNT) is equal to fCK_PSC/(PSCR[15:0]+1).

PSCR contains the value loaded into the current prescaler register when an update event occurs (update events include the counter being cleared by the UG bit of TIM_EGR or cleared by the slave controller working in reset mode). This means that in order for the new value to take effect, an update event must be generated.

PWM output frequency calculation formula

The output frequency calculation formula of the PWMA and PWMB two groups of PWM is the same, and each group can set a different frequency.

Alignment mode	PWM output frequency calculation formula
Edge alignment	PWM output frequency $= \frac{\text{System clock SYSclk}}{(\text{PWMx\_PSCR} + 1) \times (\text{PWMx\_ARR} + 1)}$
Center aligned	PWM output frequency $= \frac{\text{System clock SYSclk}}{(\text{PWMx\_PSCR} + 1) \times \text{PWMx\_ARR} \times 2}$

20.7.20 Prescalers low 8 bits (PWMx_PSCRL)

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_PSCRL	FED1H	PSC1[7:0]							
PWMB_PSCRL	FEF1H	PSC2[7:0]							

PSCn[7:0]: The lower 8-bit value of the prescaler. (N=A,B)

20.7.21 The upper 8 bits of the auto-reload registers (PWMx_ARRH)

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_ARRH	FED2H	ARR1[15:8]							
PWMB_ARRH	FEF2H	ARR2[15:8]							

ARRn[15:8]: automatically reload the high 8-bit value (n= A, B)

ARR contains the value to be loaded into the actual auto-reload register. When the value of auto reload is 0, the counter does not work.

20.7.22 The lower 8 bits of the auto-reload registers (PWMx_ARRL)

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_ARRL	FED3H	ARR1[7:0]							
PWMB_ARRL	FEF3H	ARR2[7:0]							

ARRn[7:0]: Automatic reloading of the lower 8-bit value (n= A, B)

20.7.23 Repeat Counter Registers (PWMx_RCR)

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_RCR	FED4H	REP1[7:0]							
PWMB_RCR	FEF4H	REP2[7:0]							

REPn[7:0]: Repeat counter value (n= A, B)

After enabling the preload function, these bits allow the user to set the update rate of the compare register (that is, periodically transfer from the preload register to the current register); if the update interrupt is allowed, it will also affect the rate of the update interrupt. Every time the down counter REP_CNT reaches 0, an update event is generated and the counter REP_CNT restarts counting from the REP value. Since REP_CNT only reloads the REP value when the period update event U_RC occurs, the new value written to the PWMn_RCR register will only take effect when the next period update event occurs. This means that in the PWM mode, (REP+1) corresponds to:

— In edge-aligned mode, the number of PWM cycles;

— In the center symmetric mode, the number of PWM half cycles;

20.7.24 Capture/Compare Registers 1/5 High 8 bits (PWMx_CCR1H)

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_CCR1H	FED5H	CCR1[15:8]							
PWMB_CCR5H	FEF5H	CCR5[15:8]							

CCRn[15:8]: Capture/compare the high 8-bit value of n (n=1,5)

If the CCn channel is configured as output: CCRn contains the loaded current comparison value (preload value). If the preload function is not selected in the PWMn_CCMR1 register (OCnPE bit), the written value will be transferred to the current register immediately. Otherwise, only when an update event occurs, the preload value will be transferred to the current capture/compare n register. The current comparison value is compared with the value of the counter PWMn_CNT, and an output signal is generated on the OCn port.

If the CCn channel is configured as an input: CCRn contains the counter value when the last input capture event occurred (this register is read-only at this time).

20.7.25 Capture/compare registers 1/5 lower 8 bits (PWMx_CCR1L)

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_CCR1L	FED6H	CCR1[7:0]							
PWMB_CCR5L	FEF6H	CCR5[7:0]							

CCRn[7:0]: Capture/compare the low 8-bit value of n (n=1,5)

20.7.26 Capture/Compare Registers 2/6 High 8-bit (PWMx_CCR2H)

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_CCR2H	FED7H	CCR2[15:8]							
PWMB_CCR6H	FEF7H	CCR6[15:8]							

CCRn[15:8]: Capture/compare the high 8-bit value of n (n=2,6)

20.7.27 Capture/compare registers 2/6 lower 8 bits (PWMx_CCR2L)

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_CCR2L	FED8H	CCR2[7:0]							
PWMB_CCR6L	FEF8H	CCR6[7:0]							

CCRn[7:0]: Capture/compare the low 8-bit value of n (n=2,6)

20.7.28 Capture/Compare Registers 3/7 High 8 bits (PWMx_CCR3H)

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_CCR3H	FED9H	CCR3[15:8]							
PWMB_CCR7H	FEF9H	CCR7[15:8]							

CCRn[15:8]: Capture/compare the high 8-bit value of n (n=3,7)

20.7.29 Capture/compare registers 3/7 lower 8 bits (PWMx_CCR3L)

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_CCR3L	FEDAH	CCR3[7:0]							
PWMB_CCR7L	FEFAH	CCR7[7:0]							

CCRn[7:0]: Capture/compare the low 8-bit value of n (n=3,7)

20.7.30 Capture/compare registers 4/8 high 8 bits (PWMx_CCR4H)

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_CCR4H	FEDBH	CCR4[15:8]							
PWMB_CCR8H	FEFBH	CCR8[15:8]							

CCRn[15:8]: Capture/compare the high 8-bit value of n (n=4,8)

20.7.31 Capture/compare registers 4/8 lower 8 bits (PWMx_CCR4L)

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_CCR4L	FEDCH	CCR4[7:0]							
PWMB_CCR8L	FEFCH	CCR8[7:0]							

CCRn[7:0]: Capture/compare the low 8-bit value of n (n=4,8)

20.7.32 Brake Registers (PWMx_BKR)

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_BRK	FEDDH	MOEA	AOEA	BKPA	BKEA	OSSRA	OSSIA	LOCKA[1:0]	
PWMB_BRK	FEFDH	MOEB	AOEB	BKPB	BKEB	OSSRB	OSSIB	LOCKB[1:0]	

MOEn: Main output enable. Once the brake input is valid, this bit is asynchronously cleared by hardware. According to the setting value of the AOE bit, this bit can be set by software or be set automatically. It is only valid for channels configured as output. (N=A,B)

0: Disable OC and OCN output or force to idle state

1: If the corresponding enable bit (CCIE bit of PWMn_CCERX register) is set, OC and OCN output are enabled.

AOEn: automatic output enable (n= A, B)

0: MOE can only be set by software;

1: MOE can be set to 1 by the software or automatically set to 1 in the next update event (if the brake input is invalid).

Note: Once the LOCK level (LOCK bit in the PWMn_BKR register) is set to 1, this bit cannot be modified

BKPn: Brake input polarity (n= A, B)

0: The brake input is active at low level

1: The brake input is active at high level

Note: Once the LOCK level (LOCK bit in the PWMn_BKR register) is set to 1, this bit cannot be modified

BKEn: Brake function enable (n= A, B)

0: Forbid brake input (BRK)

1: Turn on the brake input (BRK)

Note: Once the LOCK level (LOCK bit in PWMn_BKR register) is set to 1, this bit cannot be modified.

OSSRn: "Off state" selection in operating mode. This bit is valid when MOE=1 and the channel is set to output (n= A, B)

0: When PWM is not working, disable OC/OCN output (OC/OCN enable output signal=0);

1: When PWM is not working, once CCiE=1 or CCiNE=1, first turn on OC/OCN and output an invalid level, and then set OC/OCN enable output signal=1.

Note: Once the LOCK level (LOCK bit in PWMn_BKR register) is set to 2, this bit cannot be modified.

OSSIn: "Off state" selection in idle mode. This bit is valid when MOE=0 and the channel is set to output. (N=A,B)

0: When PWM is not working, disable OC/OCN output (OC/OCN enable output signal=0);

1: When the PWM is not working, once CCiE=1 or CCiNE=1, OC/OCN outputs its idle level first, and then OC/OCN

Enable output signal=1.

Note: Once the LOCK level (LOCK bit in PWMn_BKR register) is set to 2, this bit cannot be modified.

LOCKn[1:0]: Lock settings. This bit provides write protection measures to prevent software errors (n= A, B)

LOCKn[1:0]	Protection level	Protect content
00	No protection	Register is not write protected
01	Lock level 1	Cannot write to the BKE, BKP, and AOE bits of the PWMn_BKR register,OISI bit of PWMn_OISR register
10	Lock level 2	You cannot write to the bits in lock level 1, Also cannot write CC polarity bit and OSSR/OSSI bit
11	Lock level 3	You cannot write to the bits in lock level 2, Can not write CC control bit

Note: Because the BKE, BKP, AOE, OSSR, OSSI bits can be locked (depending on the LOCK bit), they must be set when writing to the PWMn_BKR register for the first time.

20.7.33 Dead Zone Registers (PWMx_DTR)

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_DTR	FEDEH	DTGA[7:0]							
PWMB_DTR	FEFEH	DTGB[7:0]							

DTGn[7:0]: Dead zone generator setting. (N=A,B)

These bits define the duration of the dead zone inserted between complementary outputs. (TCK_PSC is the clock pulse of PWMn)

DTGn[7:5]	Dead time
000	DTGn[7:0] * t_{CK_PSC}
001	
010	
011	
100	(64 + DTGn[6:0]) * 2 * t_{CK_PSC}
101	
110	(32 + DTGn[5:0]) * 8 * t_{CK_PSC}
111	(32 + DTGn[4:0]) * 16 * t_{CK_PSC}

20.7.34 Output Idle Status Registers (PWMx_OISR)

symbol	address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_OISR	FEDFH	OIS4N	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1
PWMB_OISR	FEFFH	-	OIS8	-	OIS7	-	OIS6	-	OIS5

OIS8: OC8 output level in idle state
OIS7: OC7 output level in idle state
OIS6: OC6 output level in idle state
OIS5: OC5 output level in idle state
OIS4N: OC4N output level in idle state
OIS4: idle state OC4 output level
OIS3N: OC3N output level in idle state
OIS3: OC3 output level in idle state
OIS2N: OC2N output level in idle state
OIS2: OC2 output level in idle state
OIS1N: OC1N output level in idle state

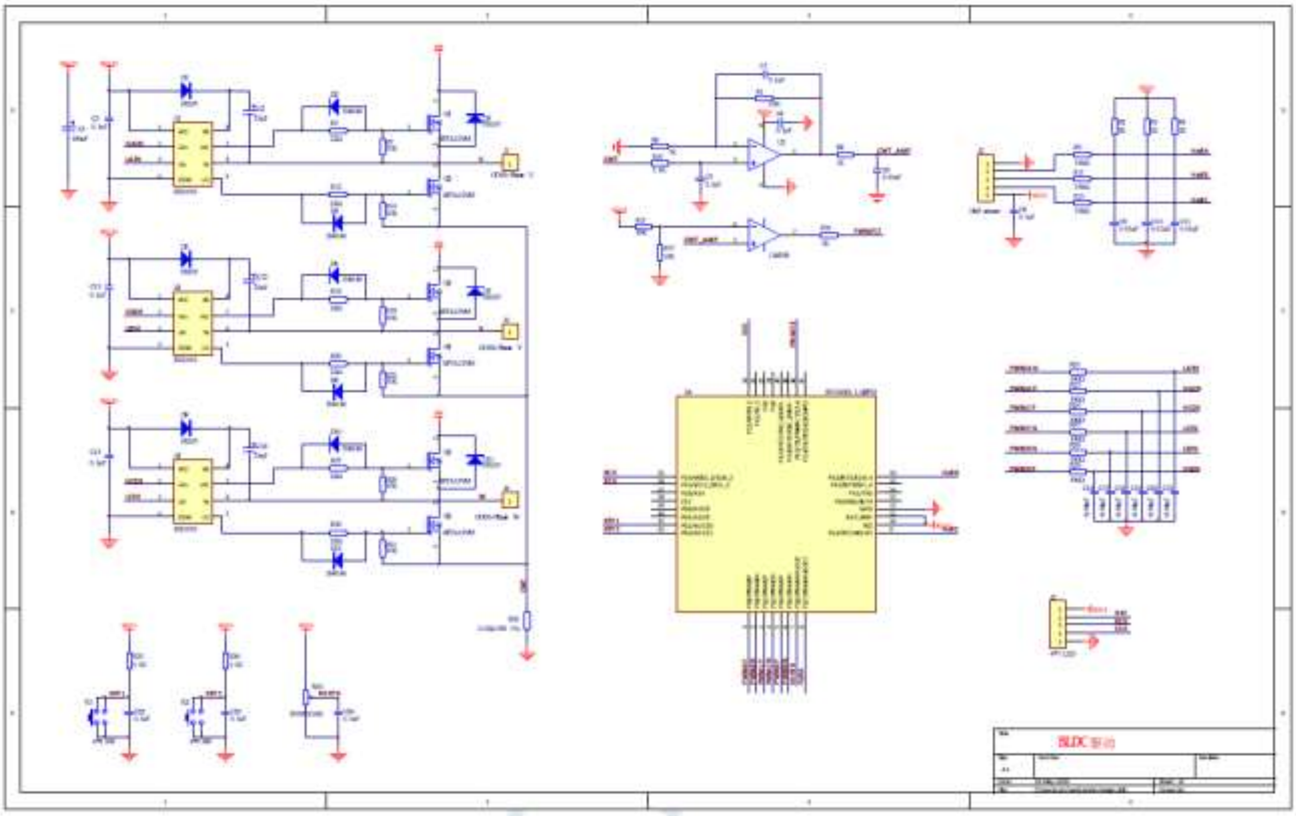
0: When MOE=0, after a dead time, OC1N=0;
1: When MOE=0, after a dead time, OC1N=1.

OIS1: OC1 output level in idle state

0: When MOE=0, if OC1N is enabled, after a dead zone, OC1=0;
1: When MOE=0, if OC1N is enabled, after a dead zone, OC1=1.

20.8 Sample Routines

20.8.1 Six-step BLDC brushless DC motor drive (with HALL)



C language code

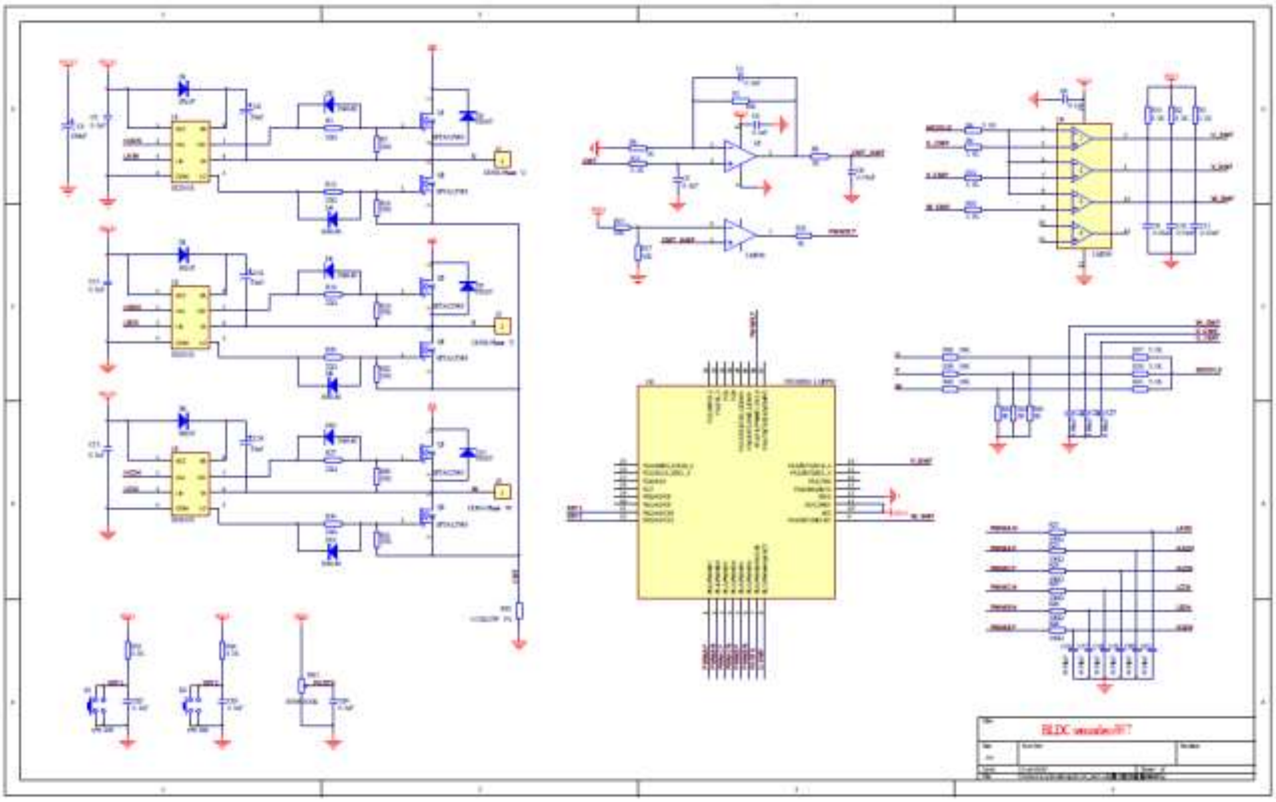
```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"
#include "reg51.h"

typedef     unsigned char    u8;
typedef     unsigned int     u16;

typedef struct TIM1_struct
{
        volatile unsigned char CR1;                      /*!< control register 1 */
        volatile unsigned char CR2;                      /*!< control register 2 */
        volatile unsigned char SMCR;                     /*!< Synchro mode control register */
        volatile unsigned char ETR;                      /*!< external trigger register */
        volatile unsigned char IER;                      /*!< interrupt enable register*/
        volatile unsigned char SR1;                      /*!< status register 1 */
        volatile unsigned char SR2;                      /*!< status register 2 */
        volatile unsigned char EGR;                      /*!< event generation register */
        volatile unsigned char CCMR1;                    /*!< CC mode register 1 */
        volatile unsigned char CCMR2;                    /*!< CC mode register 2 */
        volatile unsigned char CCMR3;                    /*!< CC mode register 3 */
        volatile unsigned char CCMR4;                    /*!< CC mode register 4 */
        volatile unsigned char CCER1;                    /*!< CC enable register 1 */
        volatile unsigned char CCER2;                    /*!< CC enable register 2 */
        volatile unsigned char CNTRH;                    /*!< counter high */
        volatile unsigned char CNTRL;                    /*!< counter low */
        volatile unsigned char PSCRH;                    /*!< prescaler high */
        volatile unsigned char PSCRL;                    /*!< prescaler low */
        volatile unsigned char ARRH;                     /*!< auto-reload register high */
        volatile unsigned char ARRL;                     /*!< auto-reload register low */
        volatile unsigned char RCR;                      /*!< Repetition Counter register */
        volatile unsigned char CCR1H;                    /*!< capture/compare register 1 high */
        volatile unsigned char CCR1L;                    /*!< capture/compare register 1 low */
        volatile unsigned char CCR2H;                    /*!< capture/compare register 2 high */
        volatile unsigned char CCR2L;                    /*!< capture/compare register 2 low */
        volatile unsigned char CCR3H;                    /*!< capture/compare register 3 high */
        volatile unsigned char CCR3L;                    /*!< capture/compare register 3 low */
        volatile unsigned char CCR4H;                    /*!< capture/compare register 3 high */
        volatile unsigned char CCR4L;                    /*!< capture/compare register 3 low */
        volatile unsigned char BKR;                      /*!< Break Register */
        volatile unsigned char DTR;                      /*!< dead-time register */
        volatile unsigned char OISR;                     /*!< Output idle register */
}TIM1_TypeDef;

#define     TIM1_BaseAddress    0xFEC0
#define     TIM2_BaseAddress    0xFEE0

#define     TIM1                ((TIM1_TypeDef xdata*) TIM1_BaseAddress)
#define     TIM2                ((TIM1_TypeDef xdata*) TIM2_BaseAddress)

#define     PWMA_ETRPS          (*(unsigned char volatile xdata *) 0xFEB0)
#define     PWMA_ENO            (*(unsigned char volatile xdata *) 0xFEB1)
#define     PWMA_PS             (*(unsigned char volatile xdata *) 0xFEB2)
#define     PWMB_ENO            (*(unsigned char volatile xdata *) 0xFEB5)
#define     PWMB_PS             (*(unsigned char volatile xdata *) 0xFEB6)

sfr         ADC_CONTR   =       0xbc;
sfr         ADC_RES     =       0xbd;
sfr         ADC_RESL    =       0xbe;
sfr         ADCCFG      =       0xde;
sfr         CMPCR1      =       0xe6;
sfr         CMPCR2      =       0xe7;

sfr         P0M0        =       0x94;
sfr         P0M1        =       0x93;
sfr         P1M0        =       0x92;
sfr         P1M1        =       0x91;
sfr         P2M0        =       0x96;
sfr         P2M1        =       0x95;
sfr         P3M0        =       0xb2;
sfr         P3M1        =       0xb1;
sfr         P5M0        =       0xca;
sfr         P5M1        =       0xc9;
sfr         P5          =       0xc8;
sfr         P_SW2       =       0xba;

sbit        P00         =       P0^0;
sbit        P01         =       P0^1;
sbit        P02         =       P0^2;
sbit        P03             =       P0^3;
sbit        P04             =       P0^4;
sbit        P05             =       P0^5;
sbit        P06             =       P0^6;
sbit        P07             =       P0^7;

sbit        P10             =       P1^0;
sbit        P11             =       P1^1;
sbit        P12             =       P1^2;
sbit        P13             =       P1^3;
sbit        P14             =       P1^4;
sbit        P15             =       P1^5;
sbit        P16             =       P1^6;
sbit        P17             =       P1^7;

sbit        P20             =       P2^0;
sbit        P21             =       P2^1;
sbit        P22             =       P2^2;
sbit        P23             =       P2^3;
sbit        P24             =       P2^4;
sbit        P25             =       P2^5;
sbit        P26             =       P2^6;
sbit        P27             =       P2^7;

sbit        P30             =       P3^0;
sbit        P31             =       P3^1;
sbit        P32             =       P3^2;
sbit        P33             =       P3^3;
sbit        P34             =       P3^4;
sbit        P35             =       P3^5;
sbit        P36             =       P3^6;
sbit        P37             =       P3^7;

sbit        P50             =       P5^0;
sbit        P51             =       P5^1;
sbit        P52             =       P5^2;
sbit        P53             =       P5^3;
sbit        P54             =       P5^4;
sbit        P55             =       P5^5;

#define     TRUE                    1
#define     FALSE                   0

#define     RV09_CH                 6

#define     TIM1_Period             ((u16)0x0180)
#define     TIM1_STPulse            ((u16)342)

#define     START                   0x1A
#define     RUN                     0x1B
#define     STOP                    0x1C
#define     IDLE                    0x1D

#define     TIM1_OCMODE_MASK                ((u8)0x70)
#define     TIM1_OCCE_ENABLE                ((u8)0x80)
#define     TIM1_OCCE_DISABLE               ((u8)0x00)
#define     TIM1_OCMODE_TIMING              ((u8)0x00)
#define     TIM1_OCMODE_ACTIVE              ((u8)0x10)
```

```
#define    TIM1_OCMODE_INACTIVE    ((u8)0x20)
#define    TIM1_OCMODE_TOGGLE      ((u8)0x30)
#define    TIM1_FORCE_INACTIVE     ((u8)0x40)
#define    TIM1_FORCE_ACTIVE       ((u8)0x50)
#define    TIM1_OCMODE_PWMA        ((u8)0x60)
#define    TIM1_OCMODE_PWMB        ((u8)0x70)
#define    CC1_POLARITY_HIGH       ((u8)0x02)
#define    CC1N_POLARITY_HIGH      ((u8)0x08)
#define    CC2_POLARITY_HIGH       ((u8)0x20)
#define    CC2N_POLARITY_HIGH      ((u8)0x80)
#define    CC1_POLARITY_LOW        ((u8)~0x02)
#define    CC1N_POLARITY_LOW       ((u8)~0x08)
#define    CC2_POLARITY_LOW        ((u8)~0x20)
#define    CC2N_POLARITY_LOW       ((u8)~0x80)
#define    CC1_OCENABLE            ((u8)0x01)
#define    CC1N_OCENABLE           ((u8)0x04)
#define    CC2_OCENABLE            ((u8)0x10)
#define    CC2N_OCENABLE           ((u8)0x40)
#define    CC1_OCDISABLE           ((u8)~0x01)
#define    CC1N_OCDISABLE          ((u8)~0x04)
#define    CC2_OCDISABLE           ((u8)~0x10)
#define    CC2N_OCDISABLE          ((u8)~0x40)
#define    CC3_POLARITY_HIGH       ((u8)0x02)
#define    CC3N_POLARITY_HIGH      ((u8)0x08)
#define    CC4_POLARITY_HIGH       ((u8)0x20)
#define    CC4N_POLARITY_HIGH      ((u8)0x80)
#define    CC3_POLARITY_LOW        ((u8)~0x02)
#define    CC3N_POLARITY_LOW       ((u8)~0x08)
#define    CC4_POLARITY_LOW        ((u8)~0x20)
#define    CC4N_POLARITY_LOW       ((u8)~0x80)
#define    CC3_OCENABLE            ((u8)0x01)
#define    CC3N_OCENABLE           ((u8)0x04)
#define    CC4_OCENABLE            ((u8)0x10)
#define    CC4N_OCENABLE           ((u8)0x40)
#define    CC3_OCDISABLE           ((u8)~0x01)
#define    CC3N_OCDISABLE          ((u8)~0x04)
#define    CC4_OCDISABLE           ((u8)~0x10)
#define    CC4N_OCDISABLE          ((u8)~0x40)

void LED_OUT(u8 X);                                        //LED single-byte serial shift function

unsigned char code LED_0F[] =
{
        0xC0,0xF9,0xA4,0xB0,
        0x99,0x92,0x82,0xF8,
        0x80,0x90,0x8C,0xBF,
        0xC6,0xA1,0x86,0xFF,
        0xbf
};

#define    DIO                P23                          // Serial data input
#define    RCLK               P24                          // Clock pulse signal - valid on rising edge
#define    SCLK               P25                          //input signal - valid on rising edge

void DelayXus(unsigned char delayTime);
void DelayXms( unsigned char delayTime);
unsigned int ADC_Convert(u8 ch);
void PWM_Init(void);
void SPEED_ADJ();
unsigned char RD_HALL();
void MOTOR_START();
void MOTOR_STOP();
unsigned char KEY_detect();
void LED4_Display (unsigned int dat,unsigned char num);

unsigned char Display_num=1;
unsigned int Display_dat=0;
unsigned int Motor_speed;
unsigned char Motor_sta = IDLE;
unsigned char BRK_occur=0;
unsigned int TIM2_CAP1_v=0;
unsigned int CAP1_avg=0;
unsigned char CAP1_cnt=0;
unsigned long CAP1_sum=0;

void main(void)
{
        P_SW2 = 0x80;

        P1 = 0x00;
        P0M1 = 0x0C;
        P0M0 = 0x01;
        P1M1 = 0xc0;
        P1M0 = 0x3F;
        P2M1 = 0x00;
        P2M0 = 0x38;
        P3M1 = 0x28;
        P3M0 = 0x00;

        ET0=1;
        TR0=1;

        ADCCFG = 0x0f;
        ADC_CONTR = 0x80;

        PWMA_ENO = 0x3F;                                  //PWMA output enable
        PWMB_ENO = 0x00;                                  //PWMB output enable
        PWMA_PS = 0x00;                                   //PWMA pin selection
        PWMB_PS = 0xd5;                                   //PWMB pin selection

/************************************************************
output compare mode PWMx_duty = [CCRx/(ARR + 1)]*100
************************************************************/
/************PWMB connect with hall sensor***************/
/////////// time base unit ///////////
        TIM2-> PSCRL = 15;
        TIM2-> ARRH = 0xff;                               //Auto-reload registers, counter overflow points
        TIM2-> ARRL = 0xff;
        TIM2-> CCR4H = 0x00;
        TIM2-> CCR4L = 0x05;

///////////Channel configuration///////////
        TIM2-> CCMR1 = 0x43;                              //Channel Mode Configuration
        TIM2-> CCMR2 = 0x41;
        TIM2-> CCMR3 = 0x41;
        TIM2-> CCMR4 = 0x70;
```

```
    TIM2-> CCER1 = 0x11;
    TIM2-> CCER2 = 0x11;

///////////Mode Configuration///////////
    TIM2-> CR2 = 0xf0;
    TIM2-> CR1 = 0x81;
    TIM2-> SMCR = 0x44;

///////////Enable & Interrupt Configuration///////////
    TIM2-> BKR = 0x80;                          //main output enable
    TIM2-> IER = 0x02;                          //enable interrupt


/************PWMA  Controls motor to change phase ****************/
///////////time base unit ///////////
    TIM1-> PSCRH = 0x00;                        //Prescaler register
    TIM1-> PSCRL = 0x00;
    TIM1-> ARRH = (u8)(TIM1_Period >> 8);
    TIM1-> ARRL = (u8)(TIM1_Period);

///////////Channel configuration///////////
    TIM1-> CCMR1 = 0x70;                        //Channel Mode Configuration
    TIM1-> CCMR2 = 0x70;
    TIM1-> CCMR3 = 0x70;
    TIM1-> CCER1 = 0x11;                        //Config channel output enable and polarity
    TIM1-> CCER2 = 0x01;                        //Config channel output enable and polarity
    TIM1-> OISR = 0xAA;                         //Configure the output level of each channel when MOE=0

///////////Mode Configuration///////////
    TIM1-> CR1 = 0xA0;
    TIM1-> CR2 = 0x24;
    TIM1-> SMCR = 0x20;

///////////Enable & Interrupt Configuration///////////
    TIM1-> BKR = 0x1c;
    TIM1-> CR1 |= 0x01;                         //enable counter

    EA = 1;
    while (1)
    {
        P22=~P22;
        Display_dat = Motor_speed;              //Motor_speed

        switch(Motor_sta)
        {
        case START:
            MOTOR_START();
            Motor_sta = RUN;
            break;
        case RUN:
            SPEED_ADJ();
            if((KEY_detect() == 2)||(BRK_occur == TRUE))
                Motor_sta = STOP;
            break;
        case STOP:
            MOTOR_STOP();
            Motor_sta = IDLE;
            break;
```

```
                case IDLE:
                        if(KEY_detect()==1)
                                Motor_sta = START;
                        BRK_occur = FALSE;
                        Motor_speed = 0;
                        CAP1_avg = 0;
                        CAP1_cnt = 0;
                        CAP1_sum = 0;
                        break;
                }
        }
}

void TIM0_ISR() interrupt 1
{
        TH0=0xf0;
        if(Display_num>8)
                Display_num=1;
        LED4_Display(Display_dat,Display_num);
        Display_num=(Display_num<<1);
}

void PWMA_ISR() interrupt 26
{
        if((TIM1->SR1 & 0x20))
        {
                switch(RD_HALL())
                {
                case 3:
                        TIM1-> CCMR3 &= ~TIM1_OCMODE_MASK;
                        TIM1-> CCMR3 |= TIM1_FORCE_INACTIVE;
                        TIM1-> CCMR1 &= ~TIM1_OCMODE_MASK;
                        TIM1-> CCMR1 |= TIM1_OCMODE_PWMB;
                        break;
                case 2:
                        TIM1-> CCER1 &= CC2N_POLARITY_LOW;
                        TIM1-> CCER2 |= CC3N_POLARITY_HIGH;
                        break;
                case 6:
                        TIM1-> CCMR1 &= ~TIM1_OCMODE_MASK;
                        TIM1-> CCMR1 |= TIM1_FORCE_INACTIVE;
                        TIM1-> CCMR2 &= ~TIM1_OCMODE_MASK;
                        TIM1-> CCMR2 |= TIM1_OCMODE_PWMB;
                        break;
                case 4:
                        TIM1-> CCER1 |= CC1N_POLARITY_HIGH;
                        TIM1-> CCER2 &= CC3N_POLARITY_LOW;
                        break;
                case 5:
                        TIM1-> CCMR2 &= ~TIM1_OCMODE_MASK;
                        TIM1-> CCMR2 |= TIM1_FORCE_INACTIVE;
                        TIM1-> CCMR3 &= ~TIM1_OCMODE_MASK;
                        TIM1-> CCMR3 |= TIM1_OCMODE_PWMB;
                        break;
                case 1:
                        TIM1-> CCER1 &= CC1N_POLARITY_LOW;
                        TIM1-> CCER1 |= CC2N_POLARITY_HIGH;
                        break;
                }

                CAP1_sum += TIM2_CAP1_v;
                CAP1_cnt++;
                if(CAP1_cnt==128)
                {
                        CAP1_cnt=0;
                        CAP1_avg = (CAP1_sum>>7);
                        CAP1_sum = 0;
                        Motor_speed = 5000000/CAP1_avg;
                }

                TIM1->SR1 &=~0x20;                              //clear
        }
        if((TIM1->SR1 & 0x80))                                  //BRK
        {
                BRK_occur = TRUE;
                TIM1->SR1 &=~0x80;                              //clear
        }
}

void PWMB_ISR() interrupt 27
{
        if((TIM2->SR1 & 0x02))
        {
                TIM2_CAP1_v = TIM2-> CCR1H;
                TIM2_CAP1_v = (TIM2_CAP1_v<<8) + TIM2-> CCR1L;
                TIM2->SR1 &=~0x02;
        }
}

void DelayXus(unsigned char delayTime)
{
        int i = 0;
        while( delayTime--)
        {
                for( i = 0 ; i < 1 ; i++);
        }
}

void DelayXms( unsigned char delayTime )
{
        int i = 0;
        while( delayTime--)
        {
                for( i = 0 ; i < 2 ; i++)
                {
                        DelayXus(100);
                }
        }
}

unsigned int ADC_Convert(u8 ch)
{
        u16 res=0;

        ADC_CONTR &= ~0x0f;
        ADC_CONTR |= ch;
        ADC_CONTR |= 0x40;
        DelayXus(1);
        while (!(ADC_CONTR & 0x20));
        ADC_CONTR &= ~0x20;

        res = ADC_RES;
        res = (res<<2)+(ADC_RESL>>6);
        return res;
}

void SPEED_ADJ()
{
        u16 ADC_result;

        ADC_result = (ADC_Convert(RV09_CH)/3);
        TIM1-> CCR1H = (u8)(ADC_result >> 8);                  //Counter comparison value
        TIM1-> CCR1L = (u8)(ADC_result);
        TIM1-> CCR2H = (u8)(ADC_result >> 8);
        TIM1-> CCR2L = (u8)(ADC_result);
        TIM1-> CCR3H = (u8)(ADC_result >> 8);
        TIM1-> CCR3L = (u8)(ADC_result);
}

unsigned char RD_HALL()
{
        unsigned char Hall_sta = 0;

        (P17)? (Hall_sta|=0x01) : (Hall_sta&=~0x01);
        (P54)? (Hall_sta|=0x02) : (Hall_sta&=~0x02);
        (P33)? (Hall_sta|=0x04) : (Hall_sta&=~0x04);

        return Hall_sta;
}

void MOTOR_START()
{
        u16 temp;
        u16 ADC_result;

        TIM1-> CCR1H = (u8)(TIM1_STPulse >> 8);                //Counter comparison value
        TIM1-> CCR1L = (u8)(TIM1_STPulse);
        TIM1-> CCR2H = (u8)(TIM1_STPulse >> 8);
        TIM1-> CCR2L = (u8)(TIM1_STPulse);
        TIM1-> CCR3H = (u8)(TIM1_STPulse >> 8);
        TIM1-> CCR3L = (u8)(TIM1_STPulse);
        TIM1-> BKR |= 0x80;                                    //enable main output, equivalent to a master switch
        TIM1-> IER |= 0xA0;                                    //enable interrupt

        switch(RD_HALL())
        {
        case 1:
                TIM1-> CCER1 &= CC1N_POLARITY_LOW;
                TIM1-> CCER1 |= CC2N_POLARITY_HIGH;
                TIM1-> CCER2 &= CC3N_POLARITY_LOW;
                TIM1-> CCMR3 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR3 |= TIM1_FORCE_INACTIVE;
                TIM1-> CCMR2 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR2 |= TIM1_FORCE_INACTIVE;
                TIM1-> CCMR1 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR1 |= TIM1_OCMODE_PWMB;
                break;
        case 3:
                TIM1-> CCMR3 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR3 |= TIM1_FORCE_INACTIVE;
                TIM1-> CCMR2 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR2 |= TIM1_FORCE_INACTIVE;
                TIM1-> CCMR1 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR1 |= TIM1_OCMODE_PWMB;
                TIM1-> CCER1 &= CC1N_POLARITY_LOW;
                TIM1-> CCER1 &= CC2N_POLARITY_LOW;
                TIM1-> CCER2 |= CC3N_POLARITY_HIGH;
                break;
        case 2:
                TIM1-> CCER1 &= CC1N_POLARITY_LOW;
                TIM1-> CCER1 &= CC2N_POLARITY_LOW;
                TIM1-> CCER2 |= CC3N_POLARITY_HIGH;
                TIM1-> CCMR1 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR1 |= TIM1_FORCE_INACTIVE;
                TIM1-> CCMR2 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR2 |= TIM1_OCMODE_PWMB;
                TIM1-> CCMR3 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR3 |= TIM1_FORCE_INACTIVE;
                break;
        case 6:
                TIM1-> CCMR1 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR1 |= TIM1_FORCE_INACTIVE;
                TIM1-> CCMR2 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR2 |= TIM1_OCMODE_PWMB;
                TIM1-> CCMR3 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR3 |= TIM1_FORCE_INACTIVE;
                TIM1-> CCER1 |= CC1N_POLARITY_HIGH;
                TIM1-> CCER1 &= CC2N_POLARITY_LOW;
                TIM1-> CCER2 &= CC3N_POLARITY_LOW;
                break;
        case 4:
                TIM1-> CCER1 |= CC1N_POLARITY_HIGH;
                TIM1-> CCER1 &= CC2N_POLARITY_LOW;
                TIM1-> CCER2 &= CC3N_POLARITY_LOW;
                TIM1-> CCMR1 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR1 |= TIM1_FORCE_INACTIVE;
                TIM1-> CCMR2 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR2 |= TIM1_FORCE_INACTIVE;
                TIM1-> CCMR3 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR3 |= TIM1_OCMODE_PWMB;
                break;
        case 5:
                TIM1-> CCMR1 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR1 |= TIM1_FORCE_INACTIVE;
                TIM1-> CCMR2 &= ~TIM1_OCMODE_MASK;
                 TIM1-> CCMR2 |= TIM1_FORCE_INACTIVE;
                TIM1-> CCMR3 &= ~TIM1_OCMODE_MASK;
                 TIM1-> CCMR3 |= TIM1_OCMODE_PWMB;
                TIM1-> CCER1 &= CC1N_POLARITY_LOW;
                TIM1-> CCER1 |= CC2N_POLARITY_HIGH;
                TIM1-> CCER2 &= CC3N_POLARITY_LOW;
                break;
        }
        ADC_result = (ADC_Convert(RV09_CH)/3);

        for(temp = TIM1_STPulse; temp > ADC_result; temp--)
        {
                TIM1-> CCR1H = (u8)(temp >> 8);                 //Counter comparison value
                TIM1-> CCR1L = (u8)(temp);
                TIM1-> CCR2H = (u8)(temp >> 8);
                TIM1-> CCR2L = (u8)(temp);
                TIM1-> CCR3H = (u8)(temp >> 8);
                TIM1-> CCR3L = (u8)(temp);
                DelayXms(10);
        }
}

void MOTOR_STOP()
{
        TIM1-> BKR &= ~0x80;
        TIM1-> IER &= ~0xA0;
}

void LED4_Display (u16 dat,u8 num)
{
        switch(num)
        {
        case 0x01:
                LED_OUT(LED_0F[(dat/1)%10]);
                LED_OUT(0x01);
                RCLK = 0;
                RCLK = 1;
                break;
        case 0x02:
                LED_OUT(LED_0F[(dat/10)%10]
                LED_OUT(0x02);
                RCLK = 0;
                RCLK = 1;
                break;
        case 0x04:
                LED_OUT(LED_0F[(dat/100)%10]);
                LED_OUT(0x04);
                RCLK = 0;
                RCLK = 1;
                break;
        case 0x08:
                LED_OUT(LED_0F[(dat/1000)%10]);
                LED_OUT(0x08);
                RCLK = 0;
                RCLK = 1;
                break;
        }
}

void LED_OUT(u8 X)
{
        u8 i;

        for(i=8;i>=1;i--)
        {
                if (X&0x80) DIO=1;
                else DIO=0;
                X<<=1;
                SCLK = 0;
                SCLK = 1;
        }
}

unsigned char KEY_detect()
{
        if(!P02)
        {
                DelayXms(10);
                if(!P02)
                {
                        return 1;
                }
                else return 0;
        }
        else if(!P03)
        {
                DelayXms(10);
                if(!P03)
                {
                        return 2;
                }
                else return 0;
        }
        else return 0;
}
```

20.8.2 BLDC brushless DC motor drive (without HALL)



C language code

```
//Operating frequency for test is 11.0592MHz
//This routine implements the following functions: Control the operation of the Hallless motor through 3 sets of PWM channels
//This example is only applicable to the demonstration of 57BL02 motor under 24V no-load condition
#include "reg51.h"
#include "intrins.h"
#include "reg51.h"
typedef unsigned char u8;
typedef unsigned int u16;

typedef struct TIM1_struct
{
        volatile unsigned char CR1; /*!< control register 1 */
        volatile unsigned char CR2; /*!< control register 2 */
        volatile unsigned char SMCR; /*!< Synchro mode control register */
        volatile unsigned char ETR; /*!< external trigger register */
        volatile unsigned char IER; /*!< interrupt enable register*/
        volatile unsigned char SR1; /*!< status register 1 */
        volatile unsigned char SR2; /*!< status register 2 */
        volatile unsigned char EGR; /*!< event generation register */
        volatile unsigned char CCMR1; /*!< CC mode register 1 */
        volatile unsigned char CCMR2; /*!< CC mode register 2 */
        volatile unsigned char CCMR3; /*!< CC mode register 3 */
        volatile unsigned char CCMR4; /*!< CC mode register 4 */
        volatile unsigned char CCER1; /*!< CC enable register 1 */
        volatile unsigned char CCER2; /*!< CC enable register 2 */
        volatile unsigned char CNTRH; /*!< counter high */
        volatile unsigned char CNTRL; /*!< counter low */
        volatile unsigned char PSCRH; /*!< prescaler high */
        volatile unsigned char PSCRL; /*!< prescaler low */
        volatile unsigned char ARRH; /*!< auto-reload register high */
        volatile unsigned char ARRL; /*!< auto-reload register low */
        volatile unsigned char RCR; /*!< Repetition Counter register */
        volatile unsigned char CCR1H; /*!< capture/compare register 1 high */
        volatile unsigned char CCR1L; /*!< capture/compare register 1 low */
        volatile unsigned char CCR2H; /*!< capture/compare register 2 high */
        volatile unsigned char CCR2L; /*!< capture/compare register 2 low */
        volatile unsigned char CCR3H; /*!< capture/compare register 3 high */
        volatile unsigned char CCR3L; /*!< capture/compare register 3 low */
        volatile unsigned char CCR4H; /*!< capture/compare register 3 high */
        volatile unsigned char CCR4L; /*!< capture/compare register 3 low */
        volatile unsigned char BKR; /*!< Break Register */
        volatile unsigned char DTR; /*!< dead-time register */
        volatile unsigned char OISR; /*!< Output idle register */
}TIM1_TypeDef;

#define     TIM1_BaseAddress    0xFEC0
#define     TIM2_BaseAddress    0xFEE0
#define     TIM1                ((TIM1_TypeDef xdata*) TIM1_BaseAddress)
#define     TIM2                ((TIM1_TypeDef xdata*) TIM2_BaseAddress)
#define     PWMA_ETRPS (*(unsigned char volatile xdata *) 0xFEB0)
#define     PWMA_ENO            (*(unsigned char volatile xdata *) 0xFEB1)
#define     PWMA_PS             (*(unsigned char volatile xdata *) 0xFEB2)
#define     PWMB_ENO            (*(unsigned char volatile xdata *) 0xFEB5)
#define     PWMB_PS             (*(unsigned char volatile xdata *) 0xFEB6)

sfr         ADC_CONTR   =       0xbc;
sfr         ADC_RES     =       0xbd;
sfr         ADC_RESL    =       0xbe;
sfr         ADCCFG      =       0xde;
sfr         CMPCR1      =       0xe6;
sfr         CMPCR2      =       0xe7;
sfr         AUXR        =       0x8e;
sfr         P0M0        =       0x94;
sfr         P0M1        =       0x93;
sfr         P1M0        =       0x92;
sfr         P1M1        =       0x91;
sfr         P2M0        =       0x96;
sfr         P2M1        =       0x95;
sfr         P3M0        =       0xb2;
sfr         P3M1        =       0xb1;
sfr         P5M0        =       0xca;
sfr         P5M1        =       0xc9;
sfr         P5          =       0xc8;
sfr         P_SW2       =       0xba;

sbit        P00         =       P0^0;
sbit        P01         =       P0^1;
sbit        P02         =       P0^2;
sbit        P03         =       P0^3;
sbit        P04         =       P0^4;
sbit        P05         =       P0^5;
sbit        P06         =       P0^6;
sbit        P07         =       P0^7;

sbit        P10         =       P1^0;
sbit        P11         =       P1^1;
sbit        P12         =       P1^2;
sbit        P13         =       P1^3;
sbit        P14         =       P1^4;
sbit        P15         =       P1^5;
sbit        P16         =       P1^6;
sbit        P17         =       P1^7;

sbit        P20         =       P2^0;
sbit        P21             =       P2^1;
sbit        P22             =       P2^2;
sbit        P23             =       P2^3;
sbit        P24             =       P2^4;
sbit        P25             =       P2^5;
sbit        P26             =       P2^6;
sbit        P27             =       P2^7;

sbit        P30             =       P3^0;
sbit        P31             =       P3^1;
sbit        P32             =       P3^2;
sbit        P33             =       P3^3;
sbit        P34             =       P3^4;
sbit        P35             =       P3^5;
sbit        P36             =       P3^6;
sbit        P37             =       P3^7;

sbit        P50             =       P5^0;
sbit        P51             =       P5^1;
sbit        P52             =       P5^2;
sbit        P53             =       P5^3;
sbit        P54             =       P5^4;
sbit        P55             =       P5^5;

#define     TRUE                    1
#define     FALSE                   0

#define     RV09_CH                 6

#define     TIM1_Period             ((u16)280)
#define     TIM1_STPulse            ((u16)245)

#define     START                   0x1A
#define     RUN                     0x1B
#define     STOP                    0x1C
#define     IDLE                    0x1D

#define     TIM1_OCMODE_MASK                ((u8)0x70)
#define     TIM1_OCCE_ENABLE                ((u8)0x80)
#define     TIM1_OCCE_DISABLE               ((u8)0x00)
#define     TIM1_OCMODE_TIMING              ((u8)0x00)
#define     TIM1_OCMODE_ACTIVE              ((u8)0x10)
#define     TIM1_OCMODE_INACTIVE            ((u8)0x20)
#define     TIM1_OCMODE_TOGGLE              ((u8)0x30)
#define     TIM1_FORCE_INACTIVE             ((u8)0x40)
#define     TIM1_FORCE_ACTIVE               ((u8)0x50)
#define     TIM1_OCMODE_PWMA                ((u8)0x60)
#define     TIM1_OCMODE_PWMB                ((u8)0x70)
#define     CC1_POLARITY_HIGH               ((u8)0x02)
#define     CC1N_POLARITY_HIGH              ((u8)0x08)
#define     CC2_POLARITY_HIGH               ((u8)0x20)
#define     CC2N_POLARITY_HIGH              ((u8)0x80)
#define     CC1_POLARITY_LOW                ((u8)~0x02)
#define     CC1N_POLARITY_LOW               ((u8)~0x08)
#define     CC2_POLARITY_LOW                ((u8)~0x20)
#define     CC2N_POLARITY_LOW               ((u8)~0x80)
#define     CC1_OCENABLE                    ((u8)0x01)
#define     CC1N_OCENABLE                   ((u8)0x04)
```

```
#define     CC2_OCENABLE                ((u8)0x10)
#define     CC2N_OCENABLE               ((u8)0x40)
#define     CC1_OCDISABLE               ((u8)~0x01)
#define     CC1N_OCDISABLE              ((u8)~0x04)
#define     CC2_OCDISABLE               ((u8)~0x10)
#define     CC2N_OCDISABLE              ((u8)~0x40)
#define     CC3_POLARITY_HIGH           ((u8)0x02)
#define     CC3N_POLARITY_HIGH          ((u8)0x08)
#define     CC4_POLARITY_HIGH           ((u8)0x20)
#define     CC4N_POLARITY_HIGH          ((u8)0x80)
#define     CC3_POLARITY_LOW            ((u8)~0x02)
#define     CC3N_POLARITY_LOW           ((u8)~0x08)
#define     CC4_POLARITY_LOW            ((u8)~0x20)
#define     CC4N_POLARITY_LOW           ((u8)~0x80)
#define     CC3_OCENABLE                ((u8)0x01)
#define     CC3N_OCENABLE               ((u8)0x04)
#define     CC4_OCENABLE                ((u8)0x10)
#define     CC4N_OCENABLE               ((u8)0x40)
#define     CC3_OCDISABLE               ((u8)~0x01)
#define     CC3N_OCDISABLE              ((u8)~0x04)
#define     CC4_OCDISABLE               ((u8)~0x10)
#define     CC4N_OCDISABLE              ((u8)~0x40)

void UART_INIT();
void DelayXus(unsigned char delayTime);
void DelayXms( unsigned char delayTime);
unsigned int ADC_Convert(u8 ch);
void PWM_Init(void);
void SPEED_ADJ();
unsigned char RD_HALL();
void MOTOR_START();
void MOTOR_STOP();
unsigned char KEY_detect();

unsigned char Timer0_cnt=0xb0;
unsigned int HA=0;
unsigned int Motor_speed;
unsigned char Motor_sta =  IDLE;
unsigned char BRK_occur=0;
unsigned int TIM2_CAP1_v=0;
unsigned int CAP1_avg=0;
unsigned char CAP1_cnt=0;
unsigned long CAP1_sum=0;

void main(void)
{
        unsigned int temp=0;
        unsigned int ADC_result=0;

        P_SW2= 0x80;
        P1 = 0x00;
        P0M1 = 0x0C;
        P0M0 = 0x01;
        P1M1 = 0xc0;
        P1M0 = 0x3F;
        P2M1 = 0x00;
        P2M0 = 0x38;
        P3M1 = 0x88;
        P3M0 = 0x02;
```

```
    ET0=1;
    TR0=0;
    ADCCFG = 0x0f;
    ADC_CONTR = 0x80;
    PWMA_ENO = 0x3F;                         //PWMA output enable
    PWMB_ENO = 0x00;                         //PWMB output enable
    PWMA_PS = 0x00;                         /PWMA pin selection
    PWMB_PS = 0xD5;                         //PWMB pin selection
/***************************************************************
output compare mode  PWMx_duty = [CCRx/(ARR + 1)]*100
***************************************************************/
/************PWMB BMF input   ****************/
/////////// time base unit   ///////////
    TIM2-> PSCRL = 15;
    TIM2-> ARRH = 0xff;                     // Auto-reload register, counter overflow point
    TIM2-> ARRL = 0xff;
    TIM2-> CCR4H = 0x00;
    TIM2-> CCR4L = 0x05;
    /////////// Channel configuration  ///////////
    TIM2-> CCMR1 = 0xf3;                    //Channel Mode Configuration
    TIM2-> CCMR2 = 0xf1;
    TIM2-> CCMR3 = 0xf1;
    TIM2-> CCMR4 = 0x70;
    TIM2-> CCER1 = 0x11;
    TIM2-> CCER2 = 0x11;
    /////////// Mode Configuration  ///////////
    TIM2-> CR2 = 0xf0;
    TIM2-> CR1 = 0x81;
    TIM2-> SMCR = 0x44;
    /////////// Enable & Interrupt Configuration  ///////////
    TIM2-> BKR = 0x80;                  //enable output
    TIM2-> IER = 0x02;                      //enable interrupt
    /************PWMA   Controls motor to change phase   ****************/
    /////////// time base unit   ///////////
    TIM1-> PSCRH = 0x00;                    //Prescaler register
    TIM1-> PSCRL = 0x00;
    TIM1-> ARRH = (u8)(TIM1_Period >> 8);
    TIM1-> ARRL = (u8)(TIM1_Period);
    /////////// Channel configuration  ///////////
    TIM1-> CCMR1 = 0x70;                    //Channel Mode Configuration
    TIM1-> CCMR2 = 0x70;
    TIM1-> CCMR3 = 0x70;
    TIM1-> CCER1 = 0x11;                     //Config channel output enable and polarity
    TIM1-> CCER2 = 0x01;                    //Config channel output enable and polarity
    TIM1-> OISR = 0xAA;                         //Configure the output level of each channel when MOE=0
    /////////// Mode Configuration  ///////////
    TIM1-> CR1 = 0xA0;
    TIM1-> CR2 = 0x24;
    TIM1-> SMCR = 0x20;
    TIM1-> BKR = 0x0c;
    /////////// Enable & Interrupt Configuration  ///////////
    TIM1-> CR1 |= 0x01;                          //enable counter
    EA = 1;

    UART_INIT();

    while (1)
    {
        switch(Motor_sta)
        {
            case START:
                MOTOR_START();
                Motor_sta = RUN;
                for(temp = TIM1_STPulse; temp > ADC_result; temp--) // Open loop start
                {
                    ADC_result = (ADC_Convert(RV09_CH)/4);
                    TIM1-> CCR1H = (u8)(temp >> 8);
                    TIM1-> CCR1L = (u8)(temp);
                    TIM1-> CCR2H = (u8)(temp >> 8);
                    TIM1-> CCR2L = (u8)(temp);
                    TIM1-> CCR3H = (u8)(temp >> 8);
                    TIM1-> CCR3L = (u8)(temp);
                    DelayXms(10);
                }
                break;
            case RUN:
                SPEED_ADJ();                // Motor speed adjustment
                if((BRK_occur == TRUE))
                Motor_sta = STOP;
                break;
            case STOP:
                MOTOR_STOP();
                Motor_sta = IDLE;
                break;
            case IDLE:
                if(KEY_detect()==1)
                Motor_sta = START;                  //Start motor
                BRK_occur = FALSE;
                Motor_speed = 0;
                CAP1_avg = 0;
                CAP1_cnt = 0;
                CAP1_sum = 0;
                break;
        }
    }
}
void TIM0_ISR() interrupt 1
{
    if(Motor_sta == START)
    {
        if(Timer0_cnt<0xe0) Timer0_cnt++;
        TH0=Timer0_cnt;

        switch(HA%6)
        {
            case 0:
                TIM1-> CCMR3 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR3 |= TIM1_FORCE_INACTIVE;
                TIM1-> CCMR1 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR1 |= TIM1_OCMODE_PWMB;
                break;
            case 1:
                TIM1-> CCER1 &= CC2N_POLARITY_LOW;
                TIM1-> CCER2 |= CC3N_POLARITY_HIGH;
                break;
            case 2:
                TIM1-> CCMR1 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR1 |= TIM1_FORCE_INACTIVE;
                TIM1-> CCMR2 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR2 |= TIM1_OCMODE_PWMB;
                break;
            case 3:
                TIM1-> CCER1 |= CC1N_POLARITY_HIGH;
                TIM1-> CCER2 &= CC3N_POLARITY_LOW;
                break;
            case 4:
                TIM1-> CCMR2 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR2 |= TIM1_FORCE_INACTIVE;
                TIM1-> CCMR3 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR3 |= TIM1_OCMODE_PWMB;
                break;
            case 5:
                TIM1-> CCER1 &= CC1N_POLARITY_LOW;
                TIM1-> CCER1 |= CC2N_POLARITY_HIGH;
                break;
        }
        HA++;
    }
    if(Motor_sta == RUN)
    {
        TR0=0;
        switch(RD_HALL())
        {
            case 3:
                TIM1-> CCMR3 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR3 |= TIM1_FORCE_INACTIVE;
                TIM1-> CCMR1 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR1 |= TIM1_OCMODE_PWMB;
                break;
            case 1:
                TIM1-> CCER1 &= CC2N_POLARITY_LOW;
                TIM1-> CCER2 |= CC3N_POLARITY_HIGH;
                break;
            case 5:
                TIM1-> CCMR1 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR1 |= TIM1_FORCE_INACTIVE;
                TIM1-> CCMR2 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR2 |= TIM1_OCMODE_PWMB;
                break;
            case 4:
                TIM1-> CCER1 |= CC1N_POLARITY_HIGH;
                TIM1-> CCER2 &= CC3N_POLARITY_LOW;
                break;
            case 6:
                TIM1-> CCMR2 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR2 |= TIM1_FORCE_INACTIVE;
                TIM1-> CCMR3 &= ~TIM1_OCMODE_MASK;
                TIM1-> CCMR3 |= TIM1_OCMODE_PWMB;
                break;
            case 2:
                TIM1-> CCER1 &= CC1N_POLARITY_LOW;
                TIM1-> CCER1 |= CC2N_POLARITY_HIGH;
                break;
        }
    }
}
void PWMA_ISR() interrupt 26
{
    if((TIM1->SR1 & 0x20))
    {
        P00=0;
        CAP1_sum += TIM2_CAP1_v;
        CAP1_cnt++;
        if(CAP1_cnt==128)
        {
            CAP1_cnt=0;
            CAP1_avg = (CAP1_sum>>7);
            CAP1_sum = 0;
            Motor_speed = 5000000/CAP1_avg;
        }
        TIM1->SR1 &=~0x20;                      //clear
    }
    if((TIM1->SR1 & 0x80))                      //BRK
    {
        BRK_occur = TRUE;
        TIM1->SR1 &=~0x80;                      //clear
    }
}

void PWMB_ISR() interrupt 27
```

```
{
        unsigned char ccr_tmp=0;

        if((TIM2->SR1 & 0X02))
        {
                ccr_tmp = TIM2-> CCR1H;
                if(ccr_tmp>1)                                   // software filtering
                {
                    TIM2_CAP1_v = ccr_tmp;
                    TIM2_CAP1_v = (TIM2_CAP1_v<<8) + TIM2->CCR1L;
                    if(Motor_sta == RUN)                        //phase change delay timing
                    {
                            TR0=1;
                            TH0 = 256-(TIM2_CAP1_v>>9);
                    }
                }
                TIM2->SR1 &=~0X02;
        }
}
void UART_INIT()
{
        SCON = 0x50;                                    //8-bit, variable baud rate
        AUXR = 0x40;                                     // Timer 1 is in 1T mode
        TMOD = 0x20;                                     // Timer 1 works in Mode 0 (16-bit auto-reload)
        TL1 = 254;
        TH1 = 254;
        // ET1 = 0;
        TR1 = 1;
}
void DelayXus(unsigned char delayTime)
{
        int i = 0;
        while( delayTime--)
        {
                for( i = 0 ; i < 1 ; i++);
        }
}
void DelayXms( unsigned char delayTime )
{
        int i = 0;
        while( delayTime--)
        {
                for( i = 0 ; i < 2 ; i++)
                {
                        DelayXus(100);
                }
        }
}
unsigned int ADC_Convert(u8 ch)
{
        u16 res=0;

        ADC_CONTR &= ~0x0f;
        ADC_CONTR |= ch;
        ADC_CONTR |= 0x40;
        DelayXus(1);
        while (!(ADC_CONTR & 0x20));
        ADC_CONTR &= ~0x20;

        res = ADC_RES;
        res = (res<<2)+(ADC_RESL>>6);

        if (res < 360) res=360;
        if (res > 900) res=900;
        return res;
}
void SPEED_ADJ()
```

```
{
        u16 ADC_result;

        ADC_result = (ADC_Convert(RV09_CH)/4);          // Speed knob ADC sampling
        TIM1-> CCR1H = (u8)(ADC_result >> 8);           //Counter comparison value
        TIM1-> CCR1L = (u8)(ADC_result);
        TIM1-> CCR2H = (u8)(ADC_result >> 8);
        TIM1-> CCR2L = (u8)(ADC_result);
        TIM1-> CCR3H = (u8)(ADC_result >> 8);
        TIM1-> CCR3L = (u8)(ADC_result);
}
unsigned char RD_HALL()                                 // read hall sensor
{
        unsigned char Hall_sta = 0;

        DelayXus(40);
        (P17)? (Hall_sta|=0x01) : (Hall_sta&=~0x01);
        (P54)? (Hall_sta|=0x02) : (Hall_sta&=~0x02);
        (P33)? (Hall_sta|=0x04) : (Hall_sta&=~0x04);

        return Hall_sta;
}
void MOTOR_START()
{
        TIM1-> CCR1H = (u8)(TIM1_STPulse >> 8);         //Counter comparison value
        TIM1-> CCR1L = (u8)(TIM1_STPulse);
        TIM1-> CCR2H = (u8)(TIM1_STPulse >> 8);
        TIM1-> CCR2L = (u8)(TIM1_STPulse);
        TIM1-> CCR3H = (u8)(TIM1_STPulse >> 8);
        TIM1-> CCR3L = (u8)(TIM1_STPulse);
        TIM1-> BKR |= 0x80;                             //enable main output, equivalent to a master switch
        TIM1-> IER = 0x00;                              //enable interrupt
        TR0 = 1;

        while (HA < 6*20);

        TIM1-> IER = 0xa0;                              //enable interrupt
}
void MOTOR_STOP()
{
        TIM1-> BKR &= ~0x80;
        TIM1-> IER &= ~0x20;
}
unsigned char KEY_detect()
{
        if(!P37)
        {
                DelayXms(10);
                if(!P37)
                {
                        return 1;
                }
                else return 0;
        }
        else if(!P03)
        {
                DelayXms(10);
                if(!P03)
                {
                        return 2;
                }
                else return 0;
        }
        else return 0;
}
```

20.8.3 Quadrature encoder mode

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

typedef struct TIM1_struct
{
volatile unsigned char CR1;                              /*!< control register 1 */
        volatile unsigned char CR2;                      /*!< control register 2 */
        volatile unsigned char SMCR;                     /*!< Synchro mode control register */
        volatile unsigned char ETR;                      /*!< external trigger register */
        volatile unsigned char IER;                      /*!< interrupt enable register*/
        volatile unsigned char SR1;                      /*!< status register 1 */
        volatile unsigned char SR2;                      /*!< status register 2 */
        volatile unsigned char EGR;                      /*!< event generation register */
        volatile unsigned char CCMR1;                    /*!< CC mode register 1 */
        volatile unsigned char CCMR2;                    /*!< CC mode register 2 */
        volatile unsigned char CCMR3;                    /*!< CC mode register 3 */
        volatile unsigned char CCMR4;                    /*!< CC mode register 4 */
        volatile unsigned char CCER1;                    /*!< CC enable register 1 */
        volatile unsigned char CCER2;                    /*!< CC enable register 2 */
        volatile unsigned char CNTRH;                    /*!< counter high */
        volatile unsigned char CNTRL;                    /*!< counter low */
        volatile unsigned char PSCRH;                    /*!< prescaler high */
        volatile unsigned char PSCRL;                    /*!< prescaler low */
        volatile unsigned char ARRH;                     /*!< auto-reload register high */
        volatile unsigned char ARRL;                     /*!< auto-reload register low */
        volatile unsigned char RCR;                      /*!< Repetition Counter register */
        volatile unsigned char CCR1H;                    /*!< capture/compare register 1 high */
        volatile unsigned char CCR1L;                    /*!< capture/compare register 1 low */
        volatile unsigned char CCR2H;                    /*!< capture/compare register 2 high */
        volatile unsigned char CCR2L;                    /*!< capture/compare register 2 low */
        volatile unsigned char CCR3H;                    /*!< capture/compare register 3 high */
        volatile unsigned char CCR3L;                    /*!< capture/compare register 3 low */
        volatile unsigned char CCR4H;                    /*!< capture/compare register 3 high */
        volatile unsigned char CCR4L;                    /*!< capture/compare register 3 low */
        volatile unsigned char BKR;                      /*!< Break Register */
        volatile unsigned char DTR;                      /*!< dead-time register */
        volatile unsigned char OISR;                     /*!< Output idle register */
}TIM1_TypeDef;

#define     TIM1_BaseAddress    0xFEC0

#define     TIM1                ((TIM1_TypeDef xdata*)TIM1_BaseAddress)
#define     PWMA_ENO            (*(unsigned char volatile xdata *)0xFEB1)
#define     PWMA_PS             (*(unsigned char volatile xdata *)0xFEB2)

sfr         P0M0        =       0x94;
sfr         P0M1        =       0x93;
sfr         P1M0        =       0x92;
sfr         P1M1        =       0x91;
sfr         P_SW2           =       0xba;

sbit        P03             =       P0^3;

unsigned char cnt_H, cnt_L;

void main(void)
{
P_SW2 = 0x80;

        P1M1 = 0x0f;
        P1M0 = 0x00;

        PWMA_ENO = 0x00;                        // The pin configured as TRGI needs to turn off the corresponding bit of ENO and configure it as input
        PWMA_PS = 0x00;                         //00:PWM at P1

        TIM1-> PSCRH = 0x00;                    // Prescaler register
        TIM1-> PSCRL = 0x00;

        TIM1-> CCMR1  = 0x21;                   // Channel mode is configured as input, connected to encoder, filter 4 clocks
        TIM1-> CCMR2  = 0x21;                   // Channel mode is configured as input, connected to encoder, filter 4 clocks

        TIM1-> SMCR   = 0x03;                   // encoder mode 3

        TIM1-> CCER1  = 0x55;                   // Configure Channel Enable and Polarity
        TIM1-> CCER2  = 0x55;                   // Configure Channel Enable and Polarity

        TIM1-> IER  = 0x02;                     // enable interrupt

        TIM1-> CR1 |= 0x01;                     // enable counter

        EA = 1;

        while (1);
}

/******************** PWM interrupt to read encoder count value *************************/
void PWMA_ISR() interrupt 26
{
        if (TIM1->SR1 & 0X02)
        {
                P03 = ~P03;
                cnt_H = TIM1->CCR1H;
                cnt_L = TIM1->CCR1L;
                TIM1->SR1 & = ~0X02;
        }
}
```

20.8.4 Single pulse mode (trigger control pulse output)

C language code

```
//Operating frequency for test is 11.0592MHz
```

```
#include "reg51.h"
#include "intrins.h"

typedef struct TIM1_struct
{
        volatile unsigned char CR1;                             /*!< control register 1 */
        volatile unsigned char CR2;                             /*!< control register 2 */
        volatile unsigned char SMCR;                            /*!< Synchro mode control register */
        volatile unsigned char ETR;                             /*!< external trigger register */
        volatile unsigned char IER;                             /*!< interrupt enable register*/
        volatile unsigned char SR1;                             /*!< status register 1 */
        volatile unsigned char SR2;                             /*!< status register 2 */
        volatile unsigned char EGR;                             /*!< event generation register */
        volatile unsigned char CCMR1;                           /*!< CC mode register 1 */
        volatile unsigned char CCMR2;                           /*!< CC mode register 2 */
        volatile unsigned char CCMR3;                           /*!< CC mode register 3 */
        volatile unsigned char CCMR4;                           /*!< CC mode register 4 */
        volatile unsigned char CCER1;                           /*!< CC enable register 1 */
        volatile unsigned char CCER2;                           /*!< CC enable register 2 */
        volatile unsigned char CNTRH;                           /*!< counter high */
        volatile unsigned char CNTRL;                           /*!< counter low */
        volatile unsigned char PSCRH;                           /*!< prescaler high */
        volatile unsigned char PSCRL;                           /*!< prescaler low */
        volatile unsigned char ARRH;                            /*!< auto-reload register high */
        volatile unsigned char ARRL;                            /*!< auto-reload register low */
        volatile unsigned char RCR;                             /*!< Repetition Counter register */
        volatile unsigned char CCR1H;                           /*!< capture/compare register 1 high */
        volatile unsigned char CCR1L;                           /*!< capture/compare register 1 low */
        volatile unsigned char CCR2H;                           /*!< capture/compare register 2 high */
        volatile unsigned char CCR2L;                           /*!< capture/compare register 2 low */
        volatile unsigned char CCR3H;                           /*!< capture/compare register 3 high */
        volatile unsigned char CCR3L;                           /*!< capture/compare register 3 low */
        volatile unsigned char CCR4H;                           /*!< capture/compare register 3 high */
        volatile unsigned char CCR4L;                           /*!< capture/compare register 3 low */
        volatile unsigned char BKR;                             /*!< Break Register */
        volatile unsigned char DTR;                             /*!< dead-time register */
        volatile unsigned char OISR;                            /*!< Output idle register */
}TIM1_TypeDef;

#define     TIM1_BaseAddress    0xFEC0

#define     TIM1                ((TIM1_TypeDef xdata*)TIM1_BaseAddress)
#define     PWMA_ENO            (*(unsigned char volatile xdata *)0xFEB1)
#define     PWMA_PS             (*(unsigned char volatile xdata *)0xFEB2)

sfr         P0M0        =       0x94;
sfr         P0M1        =       0x93;
sfr         P1M0        =       0x92;
sfr         P1M1        =       0x91;
sfr         P_SW2       =       0xba;

sbit        P03         =       P0^3;

void main(void)
{
        P_SW2 = 0x80;
        P0M1 = 0x00;
        P0M0 = 0xFF;
        P1M1 = 0x0c;
        P1M0 = 0xF3;

        PWMA_ENO = 0xF3;                                        //IO outputs PWM
        PWMA_PS = 0x00;                                         //00:PWM at P1

        /***************************************************************
        PWMx_duty = [CCRx/(ARR + 1)]*100
        ***************************************************************/
        // The pin configured as TRGI needs to turn off the corresponding bit of ENO and configure it as input
        TIM1-> PSCRH = 0x00;                                    // Prescaler register
        TIM1-> PSCRL = 0x00;
        TIM1-> DTR = 0x00;                                      // Dead time configuration

        TIM1-> CCMR1 = 0x68;                                    // Channel Mode Configuration
        TIM1-> CCMR2 = 0x01;                                    // Configured as an input channel
        TIM1-> CCMR3 = 0x68;
        TIM1-> CCMR4 = 0x68;

        TIM1-> SMCR  = 0x66;

        TIM1-> ARRH = 0x08;                                     // Auto-reload register, counter overflow point
        TIM1-> ARRL = 0x00;

        TIM1-> CCR1H = 0x04;                                    // Counter comparison value
        TIM1-> CCR1L = 0x00;
        TIM1-> CCR2H = 0x02;
        TIM1-> CCR2L = 0x00;
        TIM1-> CCR3H = 0x01;
        TIM1-> CCR3L = 0x00;
        TIM1-> CCR4H = 0x01;
        TIM1-> CCR4L = 0x00;

        TIM1-> CCER1 = 0x55;                                    // Configure Channel Output Enable and Polarity
        TIM1-> CCER2 = 0x55;                                    // Configure Channel Output Enable and Polarity

        TIM1-> BKR = 0x80;                                      // Main output enable, equivalent to main switch
        TIM1-> IER = 0x02;                                      // enable interrupt
        TIM1-> CR1 = 0x08;                                      // Single pulse mode
        TIM1-> CR1 |= 0x01;                                     // enable counter

        EA = 1;
        while (1);
}

void PWMA_ISR() interrupt 26
{
        if (TIM1->SR1 & 0X02)
        {
                P03 = ~P03;
                TIM1->SR1 & = ~0X02;
        }
}
```

20.8.5 Gating mode (input level enable counter)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

typedef struct TIM1_struct
{
        volatile unsigned char CR1;                     /*!< control register 1 */
        volatile unsigned char CR2;                     /*!< control register 2 */
        volatile unsigned char SMCR;                    /*!< Synchro mode control register */
        volatile unsigned char ETR;                     /*!< external trigger register */
        volatile unsigned char IER;                     /*!< interrupt enable register*/
        volatile unsigned char SR1;                     /*!< status register 1 */
        volatile unsigned char SR2;                     /*!< status register 2 */
        volatile unsigned char EGR;                     /*!< event generation register */
        volatile unsigned char CCMR1;                   /*!< CC mode register 1 */
        volatile unsigned char CCMR2;                   /*!< CC mode register 2 */
        volatile unsigned char CCMR3;                   /*!< CC mode register 3 */
        volatile unsigned char CCMR4;                   /*!< CC mode register 4 */
        volatile unsigned char CCER1;                   /*!< CC enable register 1 */
        volatile unsigned char CCER2;                   /*!< CC enable register 2 */
        volatile unsigned char CNTRH;                   /*!< counter high */
        volatile unsigned char CNTRL;                   /*!< counter low */
        volatile unsigned char PSCRH;                   /*!< prescaler high */
        volatile unsigned char PSCRL;                   /*!< prescaler low */
        volatile unsigned char ARRH;                    /*!< auto-reload register high */
        volatile unsigned char ARRL;                    /*!< auto-reload register low */
        volatile unsigned char RCR;                     /*!< Repetition Counter register */
        volatile unsigned char CCR1H;                   /*!< capture/compare register 1 high */
        volatile unsigned char CCR1L;                   /*!< capture/compare register 1 low */
        volatile unsigned char CCR2H;                   /*!< capture/compare register 2 high */
        volatile unsigned char CCR2L;                   /*!< capture/compare register 2 low */
        volatile unsigned char CCR3H;                   /*!< capture/compare register 3 high */
        volatile unsigned char CCR3L;                   /*!< capture/compare register 3 low */
        volatile unsigned char CCR4H;                   /*!< capture/compare register 3 high */
        volatile unsigned char CCR4L;                   /*!< capture/compare register 3 low */
        volatile unsigned char BKR;                     /*!< Break Register */
        volatile unsigned char DTR;                     /*!< dead-time register */
        volatile unsigned char OISR;                    /*!< Output idle register */
}TIM1_TypeDef;

#define     TIM1_BaseAddress    0xFEC0

#define     TIM1                ((TIM1_TypeDef xdata*)TIM1_BaseAddress)
#define     PWMA_ENO            (*(unsigned char volatile xdata *)0xFEB1)
#define     PWMA_PS             (*(unsigned char volatile xdata *)0xFEB2)

sfr         P0M0        =       0x94;
sfr         P0M1        =       0x93;
sfr         P1M0        =       0x92;
sfr         P1M1        =       0x91;
sfr         P3M0        =       0xb2;
sfr         P3M1        =       0xb1;
sfr         P_SW2           =       0xba;

sbit        P03             =       P0^3;

void main(void)
{
        P_SW2 = 0x80;

        P0M1 = 0x00;
        P0M0 = 0xFF;
        P1M1 = 0x00;
        P1M0 = 0xFF;
        P3M1 = 0x04;
        P3M0 = 0x00;

        PWMA_ENO = 0xFF;                                    //IO outputs PWM
        PWMA_PS = 0x00;                                     //00:PWM at P1

/**************************************************************
PWMx_duty = [CCRx/(ARR + 1)]*100
***************************************************************/
// The pin configured as TRGI needs to turn off the corresponding bit of ENO and configure it as input
        TIM1-> PSCRH = 0x00;                                // Prescaler register
        TIM1-> PSCRL = 0x00;
        TIM1-> DTR = 0x00;                                  // Dead time configuration

        TIM1-> CCMR1 = 0x68;                                // Channel Mode Configuration
        TIM1-> CCMR2 = 0x68;                                // Configured as an input channel
        TIM1-> CCMR3 = 0x68;
        TIM1-> CCMR4 = 0x68;

        TIM1-> SMCR = 0x75;                                 // Gated trigger mode ETRF input

        TIM1-> ARRH = 0x08;                                 // Auto-reload register, counter overflow point
        TIM1-> ARRL = 0x00;

        TIM1-> CCR1H = 0x04;                                // Counter comparison value
        TIM1-> CCR1L = 0x00;                                //
        TIM1-> CCR2H = 0x02;                                //
        TIM1-> CCR2L = 0x00;                                //
        TIM1-> CCR3H = 0x01;                                //
        TIM1-> CCR3L = 0x00;                                //
        TIM1-> CCR4H = 0x01;                                //
        TIM1-> CCR4L = 0x00;                                //

        TIM1-> CCER1 = 0x55;                                // Configure Channel Output Enable and Polarity
        TIM1-> CCER2 = 0x55;                                // Configure Channel Output Enable and Polarity

        TIM1-> BKR = 0x80;                                  // Main output enable, equivalent to main switch
        TIM1-> IER = 0x02;                                  // enable interrupt

        TIM1-> CR1 |= 0x01;                                 // enable counter

        EA = 1;
        while (1) ;
}

void PWMA_ISR() interrupt 26
```

```
{
        if(TIM1->SR1 & 0X02)
        {
                P03 = ~P03;
                TIM1->SR1 &=~0X02;
        }
}
```

20.8.6 External clock mode

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

typedef struct TIM1_struct
{
        volatile unsigned char CR1;                     /*!< control register 1 */
        volatile unsigned char CR2;                     /*!< control register 2 */
        volatile unsigned char SMCR;                    /*!< Synchro mode control register */
        volatile unsigned char ETR;                     /*!< external trigger register */
        volatile unsigned char IER;                     /*!< interrupt enable register*/
        volatile unsigned char SR1;                     /*!< status register 1 */
        volatile unsigned char SR2;                     /*!< status register 2 */
        volatile unsigned char EGR;                     /*!< event generation register */
        volatile unsigned char CCMR1;                   /*!< CC mode register 1 */
        volatile unsigned char CCMR2;                   /*!< CC mode register 2 */
        volatile unsigned char CCMR3;                   /*!< CC mode register 3 */
        volatile unsigned char CCMR4;                   /*!< CC mode register 4 */
        volatile unsigned char CCER1;                   /*!< CC enable register 1 */
        volatile unsigned char CCER2;                   /*!< CC enable register 2 */
        volatile unsigned char CNTRH;                   /*!< counter high */
        volatile unsigned char CNTRL;                   /*!< counter low */
        volatile unsigned char PSCRH;                   /*!< prescaler high */
        volatile unsigned char PSCRL;                   /*!< prescaler low */
        volatile unsigned char ARRH;                    /*!< auto-reload register high */
        volatile unsigned char ARRL;                    /*!< auto-reload register low */
        volatile unsigned char RCR;                     /*!< Repetition Counter register */
        volatile unsigned char CCR1H;                   /*!< capture/compare register 1 high */
        volatile unsigned char CCR1L;                   /*!< capture/compare register 1 low */
        volatile unsigned char CCR2H;                   /*!< capture/compare register 2 high */
        volatile unsigned char CCR2L;                   /*!< capture/compare register 2 low */
        volatile unsigned char CCR3H;                   /*!< capture/compare register 3 high */
        volatile unsigned char CCR3L;                   /*!< capture/compare register 3 low */
        volatile unsigned char CCR4H;                   /*!< capture/compare register 3 high */
        volatile unsigned char CCR4L;                   /*!< capture/compare register 3 low */
        volatile unsigned char BKR;                     /*!< Break Register */
        volatile unsigned char DTR;                     /*!< dead-time register */
        volatile unsigned char OISR;                    /*!< Output idle register */
}TIM1_TypeDef;

#define     TIM1_BaseAddress    0xFEC0
```

```
#define     TIM1                ((TIM1_TypeDef xdata*)TIM1_BaseAddress)
#define     PWMA_ENO            (*(unsigned char volatile xdata *)0xFEB1)
#define     PWMA_PS             (*(unsigned char volatile xdata *)0xFEB2)

sfr         P0M0        =       0x94;
sfr         P0M1        =       0x93;
sfr         P1M0        =       0x92;
sfr         P1M1        =       0x91;
sfr         P3M0        =       0xb2;
sfr         P3M1        =       0xb1;
sfr         P_SW2       =       0xba;

sbit        P03         =       P0^3;

void main(void)
{
        P_SW2 = 0x80;

        P0M1 = 0x00;
        P0M0 = 0xFF;
        P1M1 = 0x00;
        P1M0 = 0xFF;
        P3M1 = 0x04;
        P3M0 = 0x00;

        PWMA_ENO = 0xFF;                        //IO outputs PWM
        PWMA_PS = 0x00;                         //00:PWM at P1

/*************************************************************
PWMx_duty = [CCRx/(ARR + 1)]*100
*************************************************************/
// The pin configured as TRGI needs to turn off the corresponding bit of ENO and configure it as input
        TIM1-> PSCRH = 0x00;                    // Prescaler register
        TIM1-> PSCRL = 0x00;
        TIM1-> DTR = 0x00;                      // Dead time configuration

        TIM1-> CCMR1 = 0x68;                    // Channel Mode Configuration
        TIM1-> CCMR2 = 0x68;                    // Configured as an input channel
        TIM1-> CCMR3 = 0x68;
        TIM1-> CCMR4 = 0x68;

        TIM1-> SMCR = 0x77;                     //ETRF input

        TIM1-> ARRH = 0x08;                     // Auto-reload register, counter overflow point
        TIM1-> ARRL = 0x00;

        TIM1-> CCR1H = 0x04;                    // Counter comparison value
        TIM1-> CCR1L = 0x00;
        TIM1-> CCR2H = 0x02;
        TIM1-> CCR2L = 0x00;
        TIM1-> CCR3H = 0x01;
        TIM1-> CCR3L = 0x00;
        TIM1-> CCR4H = 0x01;
        TIM1-> CCR4L = 0x00;

        TIM1-> CCER1 = 0x55;                    // Configure Channel Output Enable and Polarity
        TIM1-> CCER2 = 0x55;                    // Configure Channel Output Enable and Polarity
        TIM1-> BKR = 0x80;                              // enable main output, equivalent to main switch
        TIM1-> IER = 0x02;                              // enable interrupt
        TIM1-> CR1 |= 0x01;                             // enable counter

        EA = 1;
        while (1);
}

void PWMA_ISR() interrupt 26
{
        if(TIM1->SR1 & 0X02)
        {
                P03 = ~P03;
                TIM1->SR1 &=~0X02;
        }
}
```

20.8.7 Input capture mode to measure the pulse period (capture rising edge to rising edge or falling edge to falling edge)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

typedef struct TIM1_struct
{
        volatile unsigned char CR1;                     /*!< control register 1 */
        volatile unsigned char CR2;                     /*!< control register 2 */
        volatile unsigned char SMCR;                    /*!< Synchro mode control register */
        volatile unsigned char ETR;                     /*!< external trigger register */
        volatile unsigned char IER;                     /*!< interrupt enable register*/
        volatile unsigned char SR1;                     /*!< status register 1 */
        volatile unsigned char SR2;                     /*!< status register 2 */
        volatile unsigned char EGR;                     /*!< event generation register */
        volatile unsigned char CCMR1;                   /*!< CC mode register 1 */
        volatile unsigned char CCMR2;                   /*!< CC mode register 2 */
        volatile unsigned char CCMR3;                   /*!< CC mode register 3 */
        volatile unsigned char CCMR4;                   /*!< CC mode register 4 */
        volatile unsigned char CCER1;                   /*!< CC enable register 1 */
        volatile unsigned char CCER2;                   /*!< CC enable register 2 */
        volatile unsigned char CNTRH;                   /*!< counter high */
        volatile unsigned char CNTRL;                   /*!< counter low */
        volatile unsigned char PSCRH;                   /*!< prescaler high */
        volatile unsigned char PSCRL;                   /*!< prescaler low */
        volatile unsigned char ARRH;                    /*!< auto-reload register high */
        volatile unsigned char ARRL;                    /*!< auto-reload register low */
        volatile unsigned char RCR;                     /*!< Repetition Counter register */
        volatile unsigned char CCR1H;                   /*!< capture/compare register 1 high */
        volatile unsigned char CCR1L;                   /*!< capture/compare register 1 low */
        volatile unsigned char CCR2H;                   /*!< capture/compare register 2 high */
        volatile unsigned char CCR2L;                   /*!< capture/compare register 2 low */
        volatile unsigned char CCR3H;                       /*!< capture/compare register 3 high */
        volatile unsigned char CCR3L;                       /*!< capture/compare register 3 low */
        volatile unsigned char CCR4H;                       /*!< capture/compare register 3 high */
        volatile unsigned char CCR4L;                       /*!< capture/compare register 3 low */
        volatile unsigned char BKR;                         /*!< Break Register */
        volatile unsigned char DTR;                         /*!< dead-time register */
        volatile unsigned char OISR;                        /*!< Output idle register */
}TIM1_TypeDef;

#define     TIM1_BaseAddress    0xFEC0

#define     TIM1                ((TIM1_TypeDef xdata*)TIM1_BaseAddress)
#define     PWMA_ENO            (*(unsigned char volatile xdata *)0xFEB1)
#define     PWMA_PS             (*(unsigned char volatile xdata *)0xFEB2)

sfr         P0M0        =       0x94;
sfr         P0M1        =       0x93;
sfr         P1M0        =       0x92;
sfr         P1M1        =       0x91;
sfr         P3M0        =       0xb2;
sfr         P3M1        =       0xb1;
sfr         P_SW2       =       0xba;

sbit        P03         =       P0^3;

int         cap;

void main(void)
{
        P_SW2 = 0x80;

        P0M1 = 0x00;
        P0M0 = 0xFF;
        P1M1 = 0x0c;
        P1M0 = 0xF3;

        PWMA_ENO = 0xF3;                                    //IO outputs PWM
        PWMA_PS = 0x00;                                     //00:PWM at P1

/* The pin configured as TRGI needs to turn off the corresponding bit of ENO and configure it as input */
        TIM1-> PSCRH = 0x00;                                // Prescaler register
        TIM1-> PSCRL = 0x00;
        TIM1-> DTR = 0x00;                                  // Dead time configuration

        TIM1-> CCMR1 = 0x68;                                // Channel Mode Configuration
        TIM1-> CCMR2 = 0x01;                                // Configured as an input channel
        TIM1-> CCMR3 = 0x68;
        TIM1-> CCMR4 = 0x68;

        TIM1-> SMCR = 0x66;

        TIM1-> CCER1 = 0x55;                                // Configure Channel Output Enable and Polarity
        TIM1-> CCER2 = 0x55;                                // Configure Channel Output Enable and Polarity

        TIM1-> IER = 0x04;                                  // enable interrupt

        TIM1-> CR1 |= 0x01;                                 // enable counter
        EA = 1;
        while (1);
}

/* Channel 2 input, capture data is read through TIM1->CCR2H / TIM1->CCR2L*/
void PWMA_ISR() interrupt 26
{
        if(TIM1->SR1 & 0X02)
        {
                P03 = ~P03;
                TIM1->SR1 &=~0X02;
        }
        if(TIM1->SR1 & 0X04)
        {
                P03 = ~P03;
                cap = TIM1-> CCR2H;                     //read CCR2H
                cap = (cap << 8) + TIM1-> CCR2L;        //read CCR2L
                TIM1->SR1 &=~0X04;
        }
}
```

20.8.8 Input capture mode to measure the pulse high level width (capture rising edge to falling edge)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr     P_SW2       =   0xba;
sfr     P1M0        =   0x92;
sfr     P1M1        =   0x91;
sfr     P3M0        =   0xb2;
sfr     P3M1        =   0xb1;
sfr     P5M0        =   0xca;
sfr     P5M1        =   0xc9;

#define PWMA_CR1        (*(unsigned char volatile xdata *)0xfec0)
#define PWMA_IER        (*(unsigned char volatile xdata *)0xfec4)
#define PWMA_SR1        (*(unsigned char volatile xdata *)0xfec5)
#define PWMA_CCMR1      (*(unsigned char volatile xdata *)0xfec8)
#define PWMA_CCMR2      (*(unsigned char volatile xdata *)0xfec9)
#define PWMA_CCER1      (*(unsigned char volatile xdata *)0xfecc)
#define PWMA_CCR1       (*(unsigned int volatile xdata *)0xfed5)
#define PWMA_CCR2       (*(unsigned int volatile xdata *)0xfed7)

void main()
{
        P1M0 = 0x00;
        P1M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
```

```
    P5M0 = 0x00;
    P5M1 = 0x00;

    P_SW2 = 0x80;

                                            //( CC1 captures the rising edge of TI1, CC2 captures the falling
edge of TI1)
    PWMA_CCER1 = 0x00;
    PWMA_CCMR1 = 0x01;                      // CC1 is in the input mode and is mapped to TI1FP1
    PWMA_CCMR2 = 0x02;                      //CC2 is in the input mode and is mapped to TI1FP2
    PWMA_CCER1 = 0x11;                      // Enable capture function on CC1/CC2
    PWMA_CCER1 |= 0x00;                     // Set the capture polarity to the rising edge of CC1
    PWMA_CCER1 |= 0x20;                     // Set capture polarity to falling edge of CC2
    PWMA_CR1 = 0x01;

    PWMA_IER = 0x04;                        // Enable CC2 capture interrupt
    EA = 1;

    while (1);
}

void PWMA_ISR() interrupt 26
{
    unsigned int cnt;

    if (PWMA_SR1 & 0x04)
    {
        PWMA_SR1 &= ~0x04;

        cnt = PWMA_CCR2 - PWMA_CCR1;        // The difference is the high level width
    }
}
```

20.8.9 Input capture mode to measure the pulse low-level width (capture falling edge to rising edge)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr     P_SW2       =   0xba;
sfr     P1M0        =   0x92;
sfr     P1M1        =   0x91;
sfr     P3M0        =   0xb2;
sfr     P3M1        =   0xb1;
sfr     P5M0        =   0xca;
sfr     P5M1        =   0xc9;

#define PWMA_CR1        (*(unsigned char volatile xdata *)0xfec0)
#define PWMA_IER        (*(unsigned char volatile xdata *)0xfec4)
#define PWMA_SR1        (*(unsigned char volatile xdata *)0xfec5)
```

```
#define     PWMA_CCMR1      (*(unsigned char volatile xdata *)0xfec8)
#define     PWMA_CCMR2      (*(unsigned char volatile xdata *)0xfec9)
#define     PWMA_CCER1      (*(unsigned char volatile xdata *)0xfecc)
#define     PWMA_CCR1       (*(unsigned int volatile xdata *)0xfed5)
#define     PWMA_CCR2       (*(unsigned int volatile xdata *)0xfed7)

void main()
{
        P1M0 = 0x00;
        P1M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P_SW2 = 0x80;

                                                //( CC1 captures the rising edge of TI1, CC2 captures the falling
edge of TI1)
        PWMA_CCER1 = 0x00;
        PWMA_CCMR1 = 0x01;                      //CC1 is in the input mode and is mapped to TI1FP1
        PWMA_CCMR2 = 0x02;                      //CC2 is in the input mode and is mapped to TI1FP2
        PWMA_CCER1 = 0x11;                      // Enable capture function on CC1/CC2
        PWMA_CCER1 |= 0x00;                     // Set the capture polarity to the rising edge of CC1
        PWMA_CCER1 |= 0x20;                     // Set capture polarity to falling edge of CC2
        PWMA_CR1 = 0x01;

        PWMA_IER = 0x02;                        // Enable CC1 capture interrupt
        EA = 1;

        while (1);
}

void PWMA_ISR() interrupt 26
{
        unsigned int cnt;

        if (PWMA_SR1 & 0x02)
        {
                PWMA_SR1 &= ~0x02;

                cnt = PWMA_CCR1 - PWMA_CCR2;    // The difference is the low level width
        }
}
```

20.8.10 Simultaneous measurement of pulse period and duty cycle in input capture mode

Note: Only PWM1P, PWM2P, PWM5, PWM6 can measure the period and duty cycle at the same time

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
```

```
#include "intrins.h"

sfr         P_SW2           =       0xba;
sfr         P1M0            =       0x92;
sfr         P1M1            =       0x91;
sfr         P3M0            =       0xb2;
sfr         P3M1            =       0xb1;
sfr         P5M0            =       0xca;
sfr         P5M1            =       0xc9;

#define     PWMA_CR1        (*(unsigned char volatile xdata *)0xfec0)
#define     PWMA_SMCR       (*(unsigned char volatile xdata *)0xfec2)
#define     PWMA_IER        (*(unsigned char volatile xdata *)0xfec4)
#define     PWMA_SR1        (*(unsigned char volatile xdata *)0xfec5)
#define     PWMA_CCMR1      (*(unsigned char volatile xdata *)0xfec8)
#define     PWMA_CCMR2      (*(unsigned char volatile xdata *)0xfec9)
#define     PWMA_CCER1      (*(unsigned char volatile xdata *)0xfecc)
#define     PWMA_CCR1       (*(unsigned int volatile xdata *)0xfed5)
#define     PWMA_CCR2       (*(unsigned int volatile xdata *)0xfed7)

void main()
{
        P1M0 = 0x00;
        P1M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        P_SW2 = 0x80;

                                        //( CC1 captures the rising edge of TI1, CC2 captures the falling edge of TI1)
                                        // CC1 captures the period, CC2 captures the high level width

        PWMA_CCER1 = 0x00;
        PWMA_CCMR1 = 0x01;              //CC1 is in the input mode and is mapped to TI1FP1
        PWMA_CCMR2 = 0x02;              //CC2 is in the input mode and is mapped to TI1FP2
        PWMA_CCER1 = 0x11;              // Enable capture function on CC1/CC2
        PWMA_CCER1 |= 0x00;             // Set the capture polarity to the rising edge of CC1
        PWMA_CCER1 |= 0x20;             // Set capture polarity to falling edge of CC2
        PWMA_SMCR = 0x54;               //TS=TI1FP1,SMS=TI1, rising edge reset mode
        PWMA_CR1 = 0x01;

        PWMA_IER = 0x06;                // Enable CC1/CC2 capture interrupt
        EA = 1;

        while (1);
}

void PWMA_ISR() interrupt 26
{
        unsigned int cnt;

        if (PWMA_SR1 & 0x02)
        {
                PWMA_SR1 &= ~0x02;

                cnt = PWMA_CCR1;        //CC1 captures the period
```

```
    }
    if (PWMA_SR1 & 0x04)
    {
        PWMA_SR1 &= ~0x04;

        cnt = PWMA_CCR2;                            //CC2 captures duty cycle (high width)
    }
}
```

20.8.11 PWM complementary output with dead zone control

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

typedef struct TIM1_struct
{
    volatile unsigned char CR1;                 /*!< control register 1 */
    volatile unsigned char CR2;                 /*!< control register 2 */
    volatile unsigned char SMCR;                /*!< Synchro mode control register */
    volatile unsigned char ETR;                 /*!< external trigger register */
    volatile unsigned char IER;                 /*!< interrupt enable register*/
    volatile unsigned char SR1;                 /*!< status register 1 */
    volatile unsigned char SR2;                 /*!< status register 2 */
    volatile unsigned char EGR;                 /*!< event generation register */
    volatile unsigned char CCMR1;               /*!< CC mode register 1 */
    volatile unsigned char CCMR2;               /*!< CC mode register 2 */
    volatile unsigned char CCMR3;               /*!< CC mode register 3 */
    volatile unsigned char CCMR4;               /*!< CC mode register 4 */
    volatile unsigned char CCER1;               /*!< CC enable register 1 */
    volatile unsigned char CCER2;               /*!< CC enable register 2 */
    volatile unsigned char CNTRH;               /*!< counter high */
    volatile unsigned char CNTRL;               /*!< counter low */
    volatile unsigned char PSCRH;               /*!< prescaler high */
    volatile unsigned char PSCRL;               /*!< prescaler low */
    volatile unsigned char ARRH;                /*!< auto-reload register high */
    volatile unsigned char ARRL;                /*!< auto-reload register low */
    volatile unsigned char RCR;                 /*!< Repetition Counter register */
    volatile unsigned char CCR1H;               /*!< capture/compare register 1 high */
    volatile unsigned char CCR1L;               /*!< capture/compare register 1 low */
    volatile unsigned char CCR2H;               /*!< capture/compare register 2 high */
    volatile unsigned char CCR2L;               /*!< capture/compare register 2 low */
    volatile unsigned char CCR3H;               /*!< capture/compare register 3 high */
    volatile unsigned char CCR3L;               /*!< capture/compare register 3 low */
    volatile unsigned char CCR4H;               /*!< capture/compare register 3 high */
    volatile unsigned char CCR4L;               /*!< capture/compare register 3 low */
    volatile unsigned char BKR;                 /*!< Break Register */
    volatile unsigned char DTR;                 /*!< dead-time register */
    volatile unsigned char OISR;                /*!< Output idle register */
}TIM1_TypeDef;
```

```
#define     TIM1_BaseAddress     0xFEC0

#define     TIM1                 ((TIM1_TypeDef xdata*)TIM1_BaseAddress)
#define     PWMA_ENO             (*(unsigned char volatile xdata *)0xFEB1)
#define     PWMA_PS              (*(unsigned char volatile xdata *)0xFEB2)

sfr         P0M0        =        0x94;
sfr         P0M1        =        0x93;
sfr         P1M0        =        0x92;
sfr         P1M1        =        0x91;
sfr         P3M0        =        0xb2;
sfr         P3M1        =        0xb1;
sfr         P_SW2       =        0xba;

sbit        P03         =        P0^3;

void main(void)
{
        P_SW2 = 0x80;

        P0M1 = 0x00;
        P0M0 = 0xFF;
        P1M1 = 0x00;
        P1M0 = 0xFF;

        PWMA_ENO = 0xFF;                                //IO outputs PWM
        PWMA_PS = 0x00;                                 //00:PWM at P1

/*************************************************************
PWMx_duty = [CCRx/(ARR + 1)]*100
**************************************************************/
        TIM1-> PSCRH = 0x00;                            // Prescaler register
        TIM1-> PSCRL = 0x00;
        TIM1-> DTR = 0x00;                              // Dead time configuration

        TIM1-> CCMR1 = 0x68;                            // Channel Mode Configuration
        TIM1-> CCMR2 = 0x68;
        TIM1-> CCMR3 = 0x68;
        TIM1-> CCMR4 = 0x68;

        TIM1-> ARRH = 0x08;                             // Auto-reload register, counter overflow point
        TIM1-> ARRL = 0x00;

        TIM1-> CCR1H = 0x04;                            // Counter comparison value
        TIM1-> CCR1L = 0x00;
        TIM1-> CCR2H = 0x02;
        TIM1-> CCR2L = 0x00;
        TIM1-> CCR3H = 0x01;
        TIM1-> CCR3L = 0x00;
        TIM1-> CCR4H = 0x01;
        TIM1-> CCR4L = 0x00;

        TIM1-> CCER1 = 0x55;                            // Configure Channel Output Enable and Polarity
        TIM1-> CCER2 = 0x55;                            // Configure Channel Output Enable and Polarity

        TIM1-> BKR = 0x80;                              // enable main output, equivalent to main switch
        TIM1-> IER = 0x02;                              // enable interrupt
        TIM1-> CR1 = 0x01;                              // enable counter
        EA = 1;
        while (1);
}

void PWMA_ISR() interrupt 26
{
        if(TIM1->SR1 & 0X02)
        {
                P03 = ~P03;
                TIM1->SR1 &=~0X02;
        }
}
```

20.8.12 PWM port as external interrupt (falling edge interrupt or rising edge interrupt)

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     PWMA_CR1            (*(unsigned char volatile xdata *)0xfec0)
#define     PWMA_IER            (*(unsigned char volatile xdata *)0xfec4)
#define     PWMA_SR1            (*(unsigned char volatile xdata *)0xfec5)
#define     PWMA_CCMR1          (*(unsigned char volatile xdata *)0xfec8)
#define     PWMA_CCER1          (*(unsigned char volatile xdata *)0xfecc)

sfr         P0M0        =       0x94;
sfr         P0M1        =       0x93;
sfr         P1M0        =       0x92;
sfr         P1M1        =       0x91;
sfr         P3M0        =       0xb2;
sfr         P3M1        =       0xb1;

sfr         P_SW2       =       0xba;

sbit        P37         =       P3^7;

void main(void)
{
        P_SW2 = 0x80;

        P1M1 = 0x00;
        P1M0 = 0x00;
        P3M1 = 0x00;
        P3M0 = 0x00;

        P_SW2 = 0x80;
                                                    //( Capture PWM1P rising/falling edge)

        PWMA_CCER1 = 0x00;
        PWMA_CCMR1 = 0x01;                          //CC1 is in the input mode and is mapped to TI1FP1
        PWMA_CCER1 = 0x01;                              // Enable capture function on CC1
        PWMA_CCER1 |= 0x00;                             // Set the capture polarity to the rising edge of CC1
//      PWMA_CCER1 |= 0x02;                             // Set the capture polarity to the falling edge of CC1
        PWMA_CR1 = 0x01;
        PWMA_IER = 0x02;
        EA = 1;

        while (1);
}

void PWMA_ISR() interrupt 26
{
        if(PWMA_SR1 & 0X02)
        {
                P37 = ~P37;
                PWMA_SR1 &=~0X02;
        }
}
```

20.8.13 Output waveforms with any period and any duty cycle

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P_SW2           =       0xba;

#define     PWMA_CCER1              (*(unsigned char volatile xdata *)0xfecc)
#define     PWMA_CCMR1              (*(unsigned char volatile xdata *)0xfec8)
#define     PWMA_ENO                (*(unsigned char volatile xdata *)0xfeb1)
#define     PWMA_BKR                (*(unsigned char volatile xdata *)0xfedd)
#define     PWMA_CCR1               (*(unsigned int volatile xdata *)0xfed5)
#define     PWMA_ARR                (*(unsigned int volatile xdata *)0xfed2)
#define     PWMA_CR1                (*(unsigned char volatile xdata *)0xfec0)

sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

void main()
{
        P0M0 = 0x00;
```

```
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P_SW2 = 0x80;
    PWMA_CCER1 = 0x00;          // CCERx must be cleared to close the channel before writing to CCMRx
    PWMA_CCMR1 = 0x60;                              // Set CC1 to PWMA output mode
    PWMA_CCER1 = 0x01;                              // Enable CC1 channel
    PWMA_CCR1 = 100;                                // Set duty cycle time
    PWMA_ARR = 500;                                 // Set period time
    PWMA_ENO = 0x01;                                // Enable PWM1P port output
    PWMA_BKR = 0x80;                                // enable main output
    PWMA_CR1 = 0x01;                                // start timing

    while (1);
}
```

20.8.14 Use PWM CEN to start PWMA timer and trigger ADC in real time

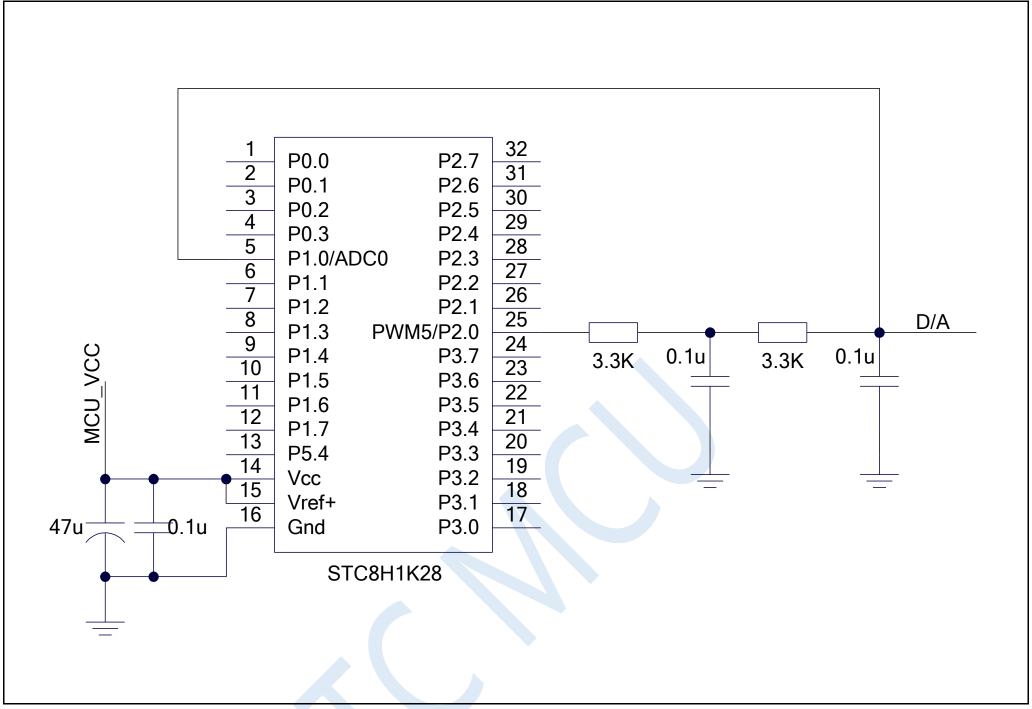
C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     PWMA_CR1        (*(unsigned char volatile xdata *)0xfec0)
#define     PWMA_CR2        (*(unsigned char volatile xdata *)0xfec1)
#define     PWMA_IER        (*(unsigned char volatile xdata *)0xfec4)
#define     PWMA_SR1        (*(unsigned char volatile xdata *)0xfec5)
#define     PWMA_CCMR1      (*(unsigned char volatile xdata *)0xfec8)
#define     PWMA_CCER1      (*(unsigned char volatile xdata *)0xfecc)
#define     PWMA_ARR        (*(unsigned int volatile xdata *)0xfed2)

sfr         P0M0        =       0x94;
sfr         P0M1        =       0x93;
sfr         P1M0        =       0x92;
sfr         P1M1        =       0x91;
sfr         P3M0        =       0xb2;
sfr         P3M1        =       0xb1;

sfr         P_SW2       =       0xba;

sfr         ADC_CONTR   =       0xbc;
```

```
#define     ADC_POWER           0x80
#define     ADC_START           0x40
#define     ADC_FLAG            0x20
#define     ADC_EPWMT           0x10
sfr         ADC_RES     =       0xbd;
sfr         ADC_RESL    =       0xbe;

sbit        EADC        =       IE^5;

void delay()
{
        int i;
        for (i=0; i<100; i++);
}

void main()
{
        P1M0 = 0x00;
        P1M1 = 0x01;
        P3M0 = 0x00;
        P3M1 = 0x00;

        P_SW2 |= 0x80;

        ADC_CONTR = ADC_POWER | ADC_EPWMT | 0;          // Select P1.0 as ADC input channel
        delay();                                        // Wait for ADC power supply to stabilize
        EADC = 1;

        PWMA_CR2 = 0x10;                // The CEN signal is TRGO, which can be used to trigger the ADC
        PWMA_ARR = 5000;
        PWMA_IER = 0x01;
        PWMA_CR1 = 0x01;                // Set CEN to start the PWMA timer and trigger the ADC in real time
        EA = 1;

        while (1);
}

void ADC_ISR() interrupt 5
{
        ADC_CONTR &= ~ADC_FLAG;
}

void PWMA_ISR() interrupt 26
{
        if(PWMA_SR1 & 0x01)
        {
                PWMA_SR1 &=~0x01;
        }
}
```

20.8.15 Reference circuit diagram for implementing 16-bit DAC using PWM

The advanced PWM timer of STC8H series MCU can output 16-bit PWM waveform, and then after two-stage low-pass filtering, 16-bit DAC signal can be generated. The DAC signal can be changed by adjusting the high-level duty cycle of the PWM waveform. . The application circuit diagram is shown in the figure below. The output DAC signal can be input to the MCU's ADC for feedback measurement.



20.8.16 Using PWM to realize complementary SPWM

Advanced PWM timers PWM1P/PWM1N, PWM2P/PWM2N, PWM3P/PWM3N, PWM4P/PWM4N Each channel can realize PWM output independently, or two-by-two complementary symmetrical output. Demonstrate the use of PWM1P and PWM1N to generate complementary SPWM. The main clock is 24MHZ, the PWM clock is 1T, the PWM period is 2400, the dead zone is 12 clocks (0.5us), and the sine wave table uses 200 points.
Output sine wave frequency = 24000000/2400/200 = 50 HZ.
This program is just a demonstration program of SPWM. Users can modify the PWM period and the number and amplitude of the sine wave through the above calculation method. The output frequency of this program is fixed. If frequency conversion is required, please design your own frequency conversion scheme.

C language code

```
//Operating frequency for test is 24MHz

#include "reg51.h"
#include "intrins.h"

#define     MAIN_Fosc           24000000L                       // define the main clock

typedef     unsigned char       u8;
typedef     unsigned int        u16;
typedef     unsigned long           u32;

sfr         TH2             =       0xD6;
sfr         TL2             =       0xD7;
sfr         IE2             =       0xAF;
sfr         INT_CLKO        =       0x8F;
sfr         AUXR            =       0x8E;
sfr         P_SW1           =       0xA2;
sfr         P_SW2           =       0xBA;

sfr         P4              =       0xC0;
sfr         P5              =       0xC8;
sfr         P6              =       0xE8;
sfr         P7              =       0xF8;
sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xB1;
sfr         P3M0            =       0xB2;
sfr         P4M1            =       0xB3;
sfr         P4M0            =       0xB4;
sfr         P5M1            =       0xC9;
sfr         P5M0            =       0xCA;
sfr         P6M1            =       0xCB;
sfr         P6M0            =       0xCC;
sfr         P7M1            =       0xE1;
sfr         P7M0            =       0xE2;

/****************************** user-defined macro ***********************************/

#define     PWMA_ENO                (*(unsigned char  volatile xdata *)  0xFEB1)
#define     PWMA_PS                 (*(unsigned char  volatile xdata *)  0xFEB2)
#define     PWMB_ENO                (*(unsigned char  volatile xdata *)  0xFEB5)
#define     PWMB_PS                 (*(unsigned char  volatile xdata *)  0xFEB6)

#define     PWMA_CR1                (*(unsigned char  volatile xdata *)  0xFEC0)
#define     PWMA_CR2                (*(unsigned char  volatile xdata *)  0xFEC1)
#define     PWMA_SMCR               (*(unsigned char  volatile xdata *)  0xFEC2)
#define     PWMA_ETR                (*(unsigned char  volatile xdata *)  0xFEC3)
#define     PWMA_IER                (*(unsigned char  volatile xdata *)  0xFEC4)
#define     PWMA_SR1                (*(unsigned char  volatile xdata *)  0xFEC5)
#define     PWMA_SR2                (*(unsigned char  volatile xdata *)  0xFEC6)
#define     PWMA_EGR                (*(unsigned char  volatile xdata *)  0xFEC7)
#define     PWMA_CCMR1              (*(unsigned char  volatile xdata *)  0xFEC8)
#define     PWMA_CCMR2              (*(unsigned char  volatile xdata *)  0xFEC9)
#define     PWMA_CCMR3              (*(unsigned char  volatile xdata *)  0xFECA)
#define     PWMA_CCMR4              (*(unsigned char  volatile xdata *)  0xFECB)
#define     PWMA_CCER1              (*(unsigned char  volatile xdata *)  0xFECC)
#define     PWMA_CCER2              (*(unsigned char  volatile xdata *)  0xFECD)
#define     PWMA_CNTRH              (*(unsigned char  volatile xdata *)  0xFECE)
#define     PWMA_CNTRL              (*(unsigned char  volatile xdata *)  0xFECF)
#define     PWMA_PSCRH              (*(unsigned char  volatile xdata *)  0xFED0)
#define     PWMA_PSCRL              (*(unsigned char  volatile xdata *)  0xFED1)
#define     PWMA_ARRH               (*(unsigned char  volatile xdata *)  0xFED2)
#define     PWMA_ARRL               (*(unsigned char  volatile xdata *)  0xFED3)
```

```
#define     PWMA_RCR          (*(unsigned char  volatile xdata *)  0xFED4)
#define     PWMA_CCR1H        (*(unsigned char  volatile xdata *)  0xFED5)
#define     PWMA_CCR1L        (*(unsigned char  volatile xdata *)  0xFED6)
#define     PWMA_CCR2H        (*(unsigned char  volatile xdata *)  0xFED7)
#define     PWMA_CCR2L        (*(unsigned char  volatile xdata *)  0xFED8)
#define     PWMA_CCR3H        (*(unsigned char  volatile xdata *)  0xFED9)
#define     PWMA_CCR3L        (*(unsigned char  volatile xdata *)  0xFEDA)
#define     PWMA_CCR4H        (*(unsigned char  volatile xdata *)  0xFEDB)
#define     PWMA_CCR4L        (*(unsigned char  volatile xdata *)  0xFEDC)
#define     PWMA_BKR          (*(unsigned char  volatile xdata *)  0xFEDD)
#define     PWMA_DTR          (*(unsigned char  volatile xdata *)  0xFEDE)
#define     PWMA_OISR         (*(unsigned char  volatile xdata *)  0xFEDF)

/*****************************************************************************/

#define     PWMA_1            0x00                        //P:P1.0  N:P1.1
#define     PWMA_2            0x01                        //P:P2.0  N:P2.1
#define     PWMA_3            0x02                        //P:P6.0  N:P6.1

#define     PWMB_1            0x00                        //P:P1.2/P5.4  N:P1.3
#define     PWMB_2            0x04                        //P:P2.2  N:P2.3
#define     PWMB_3            0x08                        //P:P6.2  N:P6.3

#define     PWM3_1            0x00                        //P:P1.4  N:P1.5
#define     PWM3_2            0x10                        //P:P2.4  N:P2.5
#define     PWM3_3            0x20                        //P:P6.4  N:P6.5

#define     PWM4_1            0x00                        //P:P1.6  N:P1.7
#define     PWM4_2            0x40                        //P:P2.6  N:P2.7
#define     PWM4_3            0x80                        //P:P6.6  N:P6.7
#define     PWM4_4            0xC0                        //P:P3.4  N:P3.3

#define     ENO1P             0x01
#define     ENO1N             0x02
#define     ENO2P             0x04
#define     ENO2N             0x08
#define     ENO3P             0x10
#define     ENO3N             0x20
#define     ENO4P             0x40
#define     ENO4N             0x80

/*************  local variable declaration    **************/

unsigned int code T_SinTable[]=
{
    1220, 1256, 1292, 1328, 1364, 1400, 1435, 1471,
    1506, 1541, 1575, 1610, 1643, 1677, 1710, 1742,
    1774, 1805, 1836, 1866, 1896, 1925, 1953, 1981,
    2007, 2033, 2058, 2083, 2106, 2129, 2150, 2171,
    2191, 2210, 2228, 2245, 2261, 2275, 2289, 2302,
    2314, 2324, 2334, 2342, 2350, 2356, 2361, 2365,
    2368, 2369, 2370, 2369, 2368, 2365, 2361, 2356,
    2350, 2342, 2334, 2324, 2314, 2302, 2289, 2275,
    2261, 2245, 2228, 2210, 2191, 2171, 2150, 2129,
    2106, 2083, 2058, 2033, 2007, 1981, 1953, 1925,
    1896, 1866, 1836, 1805, 1774, 1742, 1710, 1677,
    1643, 1610, 1575, 1541, 1506, 1471, 1435, 1400,
    1364, 1328, 1292, 1256, 1220, 1184, 1148, 1112,
        1076, 1040, 1005,  969,  934,  899,  865,  830,
         797,  763,  730,  698,  666,  635,  604,  574,
         544,  515,  487,  459,  433,  407,  382,  357,
         334,  311,  290,  269,  249,  230,  212,  195,
         179,  165,  151,  138,  126,  116,  106,   98,
          90,   84,   79,   75,   72,   71,   70,   71,
          72,   75,   79,   84,   90,   98,  106,  116,
         126,  138,  151,  165,  179,  195,  212,  230,
         249,  269,  290,  311,  334,  357,  382,  407,
         433,  459,  487,  515,  544,  574,  604,  635,
         666,  698,  730,  763,  797,  830,  865,  899,
         934,  969, 1005, 1040, 1076, 1112, 1148, 1184,
};

u16 PWMA_Duty;
u8 PWM_Index;                                           // SPWM lookup table index

/******************* main function ************************/
void main(void)
{
        P0M1 = 0;   P0M0 = 0;   //set as quasi-bidirectional port
        P1M1 = 0;   P1M0 = 0;   //set as quasi-bidirectional port
        P2M1 = 0;   P2M0 = 0;   //set as quasi-bidirectional port
        P3M1 = 0;   P3M0 = 0;   //set as quasi-bidirectional port
        P4M1 = 0;   P4M0 = 0;   //set as quasi-bidirectional port
        P5M1 = 0;   P5M0 = 0;   //set as quasi-bidirectional port
        P6M1 = 0;   P6M0 = 0;   //set as quasi-bidirectional port
        P7M1 = 0;   P7M0 = 0;   //set as quasi-bidirectional port

        PWMA_Duty = 1220;

        P_SW2 |= 0x80;

        PWMA_CCER1 = 0x00;                              // CCxE must be cleared before writing to CCMRx to close the
channel
        PWMA_CCER2 = 0x00;
        PWMA_CCMR1 = 0x60;                              // Channel Mode Configuration
//      PWMA_CCMR2 = 0x60;
//      PWMA_CCMR3 = 0x60;
//      PWMA_CCMR4 = 0x60;
        PWMA_CCER1 = 0x05;                              // Configure Channel Output Enable and Polarity
//      PWMA_CCER2 = 0x55;

        PWMA_ARRH = 0x09;                                       // Set period time
        PWMA_ARRL = 0x60;

        PWMA_CCR1H = (u8)(PWMA_Duty >> 8);              // Set duty cycle time
        PWMA_CCR1L = (u8)(PWMA_Duty);

        PWMA_DTR = 0x0C;                                // Set dead time

        PWMA_ENO = 0x00;
        PWMA_ENO |= ENO1P;                              //enable output
        PWMA_ENO |= ENO1N;                              //enable output
//      PWMA_ENO |= ENO2P;                              //enable output
//      PWMA_ENO |= ENO2N;                              //enable output
//      PWMA_ENO |= ENO3P;                              //enable output
//      PWMA_ENO |= ENO3N;                              //enable output
//      PWMA_ENO |= ENO4P;                                  //enable output
//      PWMA_ENO |= ENO4N;                                  //enable output

        PWMA_PS = 0x00;                                     // Advanced PWM Channel Output Pin Selection Bits
        PWMA_PS |= PWMA_3;                                  // Select PWMA_3 channel
//      PWMA_PS |= PWMB_3;                                  // Select PWMB_3 channel
//      PWMA_PS |= PWM3_3;                                  // Select PWM3_3 channel
//      PWMA_PS |= PWM4_3;                                  // Select PWM4_3 channel

        PWMA_BKR = 0x80;                                    // enable main output
        PWMA_IER = 0x01;                                    // enable interrupt
        PWMA_CR1 |= 0x01;                                   //start timing

        P_SW2 &= 0x7f;

        EA = 1;

        while (1)
        {
        }
}

/******************** interrupt function **************************/
void PWMA_ISR() interrupt 26
{
        P_SW2 |= 0x80;
        if (PWMA_SR1 & 0x01)
        {
                PWMA_SR1 &=~0x01;
                PWMA_Duty = T_SinTable[PWM_Index];
                if (++PWM_Index >= 200)
                        PWM_Index = 0;

                PWMA_CCR1H = (u8)(PWMA_Duty >> 8);          // Set duty cycle time
                PWMA_CCR1L = (u8)(PWMA_Duty);
        }
        PWMA_SR1 = 0;
        P_SW2 &= 0x7f;
}
```

20.8.17 Advanced PWM Output - Frequency Adjustable - Pulse Count

C language code

```
//Operating frequency for test is 24MHz

/*************              Function Description              **************
This routine is based on STC8H8K64U as the main control chip to write and test on the experiment box 8. STC8H series chips can be
used as a general reference.
Advanced PWM timer realizes high-speed PWM pulse output.
The period/duty cycle is adjustable, and the number of pulses is counted through compare/capture interrupts.
Demonstrate output through P6 port, output PWM every 10ms, stop output after counting 10 pulses.
The timer adjusts the PWM period every 1ms.
When downloading, select the clock 24MHZ (users can modify the frequency by themselves).
******************************************/
```

```
#include    "reg51.h"
#include    "intrins.h"

#define     MAIN_Fosc           24000000L

typedef     unsigned char       u8;
typedef     unsigned int        u16;
typedef     unsigned long       u32;

sfr         TH2         =       0xD6;
sfr         TL2         =       0xD7;
sfr         IE2         =       0xAF;
sfr         INT_CLKO    =       0x8F;
sfr         AUXR        =       0x8E;
sfr         P_SW1       =       0xA2;
sfr         P_SW2       =       0xBA;

sfr         P4          =       0xC0;
sfr         P5          =       0xC8;
sfr         P6          =       0xE8;
sfr         P7          =       0xF8;
sfr         P1M1        =       0x91;
sfr         P1M0        =       0x92;
sfr         P0M1        =       0x93;
sfr         P0M0        =       0x94;
sfr         P2M1        =       0x95;
sfr         P2M0        =       0x96;
sfr         P3M1        =       0xB1;
sfr         P3M0        =       0xB2;
sfr         P4M1        =       0xB3;
sfr         P4M0        =       0xB4;
sfr         P5M1        =       0xC9;
sfr         P5M0        =       0xCA;
sfr         P6M1        =       0xCB;
sfr         P6M0        =       0xCC;
sfr         P7M1        =       0xE1;
sfr         P7M0        =       0xE2;

sbit        P00         =       P0^0;
sbit        P01         =       P0^1;
sbit        P02         =       P0^2;
sbit        P03         =       P0^3;
sbit        P04         =       P0^4;
sbit        P05         =       P0^5;
sbit        P06         =       P0^6;
sbit        P07         =       P0^7;
sbit        P10         =       P1^0;
sbit        P11         =       P1^1;
sbit        P12         =       P1^2;
sbit        P13         =       P1^3;
sbit        P14         =       P1^4;
sbit        P15         =       P1^5;
sbit        P16         =       P1^6;
sbit        P17         =       P1^7;
sbit        P20         =       P2^0;
sbit        P21         =       P2^1;
sbit        P22         =       P2^2;
sbit        P23         =       P2^3;
sbit        P24             =       P2^4;
sbit        P25             =       P2^5;
sbit        P26             =       P2^6;
sbit        P27             =       P2^7;
sbit        P30             =       P3^0;
sbit        P31             =       P3^1;
sbit        P32             =       P3^2;
sbit        P33             =       P3^3;
sbit        P34             =       P3^4;
sbit        P35             =       P3^5;
sbit        P36             =       P3^6;
sbit        P37             =       P3^7;
sbit        P40             =       P4^0;
sbit        P41             =       P4^1;
sbit        P42             =       P4^2;
sbit        P43             =       P4^3;
sbit        P44             =       P4^4;
sbit        P45             =       P4^5;
sbit        P46             =       P4^6;
sbit        P47             =       P4^7;
sbit        P50             =       P5^0;
sbit        P51             =       P5^1;
sbit        P52             =       P5^2;
sbit        P53             =       P5^3;
sbit        P54             =       P5^4;
sbit        P55             =       P5^5;
sbit        P56             =       P5^6;
sbit        P57             =       P5^7;

/****************************** user-defined macro ***********************************/

#define     Timer0_Reload           (65536UL -(MAIN_Fosc / 1000))           // Timer0 interrupt frequency, 1000 times/second

#define     PWMA_ENO                (*(unsigned char  volatile xdata *)  0xFEB1)
#define     PWMA_PS                 (*(unsigned char  volatile xdata *)  0xFEB2)
#define     PWMB_ENO                (*(unsigned char  volatile xdata *)  0xFEB5)
#define     PWMB_PS                 (*(unsigned char  volatile xdata *)  0xFEB6)

#define     PWMA_CR1                 (*(unsigned char  volatile xdata *)  0xFEC0)
#define     PWMA_CR2                 (*(unsigned char  volatile xdata *)  0xFEC1)
#define     PWMA_SMCR                (*(unsigned char  volatile xdata *)  0xFEC2)
#define     PWMA_ETR                 (*(unsigned char  volatile xdata *)  0xFEC3)
#define     PWMA_IER                 (*(unsigned char  volatile xdata *)  0xFEC4)
#define     PWMA_SR1                 (*(unsigned char  volatile xdata *)  0xFEC5)
#define     PWMA_SR2                 (*(unsigned char  volatile xdata *)  0xFEC6)
#define     PWMA_EGR                 (*(unsigned char  volatile xdata *)  0xFEC7)
#define     PWMA_CCMR1               (*(unsigned char  volatile xdata *)  0xFEC8)
#define     PWMA_CCMR2               (*(unsigned char  volatile xdata *)  0xFEC9)
#define     PWMA_CCMR3               (*(unsigned char  volatile xdata *)  0xFECA)
#define     PWMA_CCMR4               (*(unsigned char  volatile xdata *)  0xFECB)
#define     PWMA_CCER1               (*(unsigned char  volatile xdata *)  0xFECC)
#define     PWMA_CCER2               (*(unsigned char  volatile xdata *)  0xFECD)
#define     PWMA_CNTR                (*(unsigned int   volatile xdata *)  0xFECE)
#define     PWMA_CNTRH               (*(unsigned char  volatile xdata *)  0xFECE)
#define     PWMA_CNTRL               (*(unsigned char  volatile xdata *)  0xFECF)
#define     PWMA_PSCRH               (*(unsigned char  volatile xdata *)  0xFED0)
#define     PWMA_PSCRL               (*(unsigned char  volatile xdata *)  0xFED1)
#define     PWMA_ARR                 (*(unsigned int   volatile xdata *)  0xFED2)
```

```
#define     PWMA_ARRH           (*(unsigned char  volatile xdata *)  0xFED2)
#define     PWMA_ARRL           (*(unsigned char  volatile xdata *)  0xFED3)
#define     PWMA_RCR            (*(unsigned char  volatile xdata *)  0xFED4)
#define     PWMA_CCR1           (*(unsigned int   volatile xdata *)  0xFED5)
#define     PWMA_CCR1H          (*(unsigned char  volatile xdata *)  0xFED5)
#define     PWMA_CCR1L          (*(unsigned char  volatile xdata *)  0xFED6)
#define     PWMA_CCR2           (*(unsigned int   volatile xdata *)  0xFED7)
#define     PWMA_CCR2H          (*(unsigned char  volatile xdata *)  0xFED7)
#define     PWMA_CCR2L          (*(unsigned char  volatile xdata *)  0xFED8)
#define     PWMA_CCR3           (*(unsigned int   volatile xdata *)  0xFED9)
#define     PWMA_CCR3H          (*(unsigned char  volatile xdata *)  0xFED9)
#define     PWMA_CCR3L          (*(unsigned char  volatile xdata *)  0xFEDA)
#define     PWMA_CCR4           (*(unsigned int   volatile xdata *)  0xFEDB)
#define     PWMA_CCR4H          (*(unsigned char  volatile xdata *)  0xFEDB)
#define     PWMA_CCR4L          (*(unsigned char  volatile xdata *)  0xFEDC)
#define     PWMA_BKR            (*(unsigned char  volatile xdata *)  0xFEDD)
#define     PWMA_DTR            (*(unsigned char  volatile xdata *)  0xFEDE)
#define     PWMA_OISR           (*(unsigned char  volatile xdata *)  0xFEDF)

/*****************************************************************************/

#define     PWM1_1              0x00                        //P:P1.0  N:P1.1
#define     PWM1_2              0x01                        //P:P2.0  N:P2.1
#define     PWM1_3              0x02                        //P:P6.0  N:P6.1

#define     PWM2_1              0x00                        //P:P1.2/P5.4  N:P1.3
#define     PWM2_2              0x04                        //P:P2.2  N:P2.3
#define     PWM2_3              0x08                        //P:P6.2  N:P6.3

#define     PWM3_1              0x00                        //P:P1.4  N:P1.5
#define     PWM3_2              0x10                        //P:P2.4  N:P2.5
#define     PWM3_3              0x20                        //P:P6.4  N:P6.5

#define     PWM4_1              0x00                        //P:P1.6  N:P1.7
#define     PWM4_2              0x40                        //P:P2.6  N:P2.7
#define     PWM4_3              0x80                        //P:P6.6  N:P6.7
#define     PWM4_4              0xC0                        //P:P3.4  N:P3.3

#define     ENO1P               0x01
#define     ENO1N               0x02
#define     ENO2P               0x04
#define     ENO2N               0x08
#define     ENO3P               0x10
#define     ENO3N               0x20
#define     ENO4P               0x40
#define     ENO4N               0x80

/************* local variable declaration **************/
bit     B_1ms;                                              //1ms flag
bit     PWM1_Flag;

u16   Period;
u8 Counter;
u8 msSecond;

void UpdatePwm(void);
void TxPulse(void);
```

```
/******************* main function ************************/
void main(void)
{
        P0M1 = 0x00;   P0M0 = 0x00;                        //set as quasi-bidirectional port
        P1M1 = 0x00;   P1M0 = 0x00;                        //set as quasi-bidirectional port
        P2M1 = 0x00;   P2M0 = 0x00;                        //set as quasi-bidirectional port
        P3M1 = 0x00;   P3M0 = 0x00;                        //set as quasi-bidirectional port
        P4M1 = 0x00;   P4M0 = 0x00;                        //set as quasi-bidirectional port
        P5M1 = 0x00;   P5M0 = 0x00;                        //set as quasi-bidirectional port
        P6M1 = 0x00;   P6M0 = 0x00;                        //set as quasi-bidirectional port
        P7M1 = 0x00;   P7M0 = 0x00;                        //set as quasi-bidirectional port

        PWM1_Flag = 0;
        Counter = 0;
        Period = 0x1000;

        // Timer0 initialization
        AUXR = 0x80;                                       //Timer0 set as 1T,16 bits timer auto-reload,
        TH0 = (u8)(Timer0_Reload / 256);
        TL0 = (u8)(Timer0_Reload % 256);
        ET0 = 1;                                           //Timer0 interrupt enable
        TR0 = 1;                                           //Tiner0 run

        P_SW2 |= 0x80;                                     // Enable XFR access

        PWMA_ENO = 0x00;
        PWMA_ENO |= ENO1P;                                 // enable output

        PWMA_PS = 0x00;                                    // Advanced PWM Channel Output Pin Selection Bits
        PWMA_PS |= PWM1_3;                                 // Select PWM1_3 channel

        UpdatePwm();
        PWMA_BKR = 0x80;                                   // enable main output
        PWMA_CR1 |= 0x01;                                  // start timing

        P40 = 0;                                           // Powering the LED
        EA = 1;                                            //Enable CPU interrupt

        while (1)
        {
                if(B_1ms)
                {
                        B_1ms = 0;
                        msSecond++;
                        if(msSecond >= 10)
                        {
                                msSecond = 0;
                                TxPulse();                 //10ms Start a PWM output
                        }
                }
        }
}

/************* send pulse function **************/
void TxPulse(void)
{
        PWMA_CCER1 = 0x00;                                 // CCxE must be cleared before writing to CCMRx to close the
channel
```

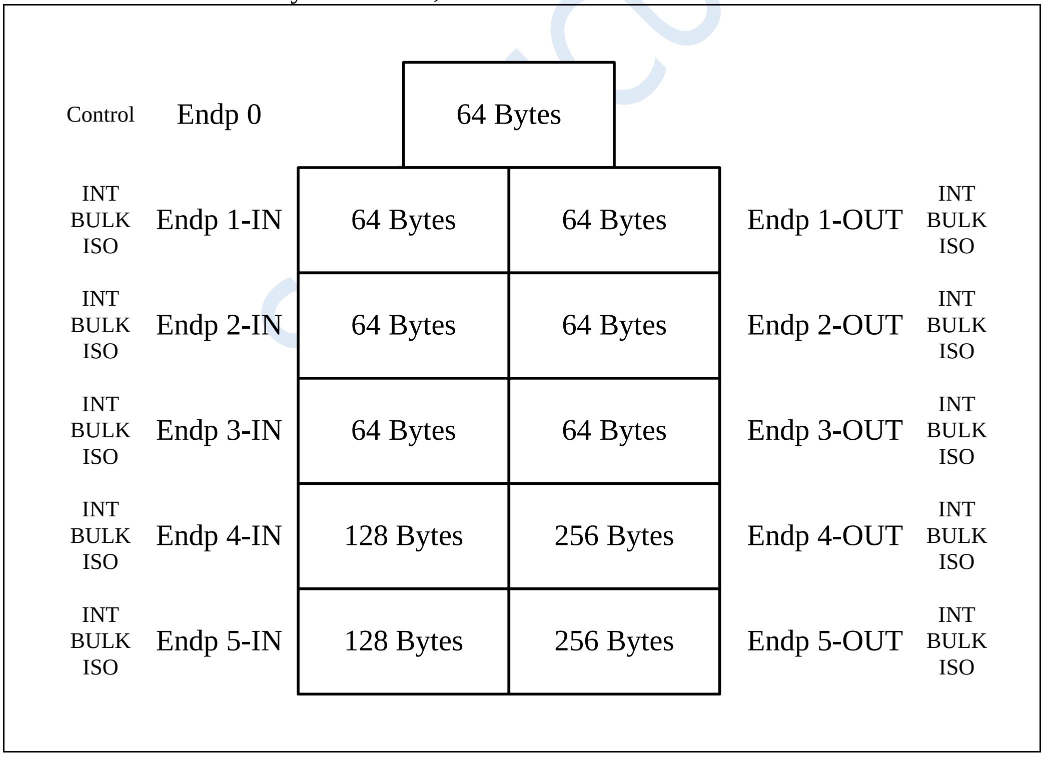
```
        PWMA_CCMR1 = 0x60;                                      // Set PWM1 Mode 1 Output
        PWMA_CCER1 = 0x01;                                      // Enable CC1E channel, active high
        PWMA_SR1 = 0;                                           //clear flag
        PWMA_CNTR = 0;                                          //clear counter
        PWMA_IER = 0x02;                                        // Enable capture/compare 1 interrupt
}

/**********************          Timer0 1ms interrupt function ***********************/
void timer0(void) interrupt 1
{
        B_1ms = 1;
        if(PWM1_Flag)
        {
                Period++;                                       // Period increment
                if(Period >= 0x1000) PWM1_Flag = 0;
        }
        else
        {
                Period--;                                       // period decrement
                if(Period <= 0x0100) PWM1_Flag = 1;
        }
        UpdatePwm();                                            // Set period, duty cycle
}

/****************** PWM interrupt function *******************/
void PWMA_ISR() interrupt 26
{
        if(PWMA_SR1 & 0X02)
        {
                PWMA_SR1 &=~0X02;                               //clear flag

                Counter++;
                if(Counter >= 10)                               // Turn off the PWM counter after counting 10 pulses
                {
                        Counter = 0;
                        PWMA_CCER1 = 0x00;                      // CCxE must be cleared before writing to CCMRx to close the
channel
                        PWMA_CCMR1 = 0x40;                      // Set PWM1 to force inactive level
                        PWMA_CCER1 = 0x01;                      // Enable CC1E channel, active high
                        PWMA_IER = 0x00;                        //disable interrupt
                }
        }
}

//========================================================================
// function: UpdatePwm(void)
// description: 更新 PWM 周期占空比.
// parameters: none.
// return: none.
// version: V1.0, 2012-11-22
//========================================================================
void UpdatePwm(void)
{
         PWMA_ARR = Period;
         PWMA_CCR1 = (Period >> 1);                             // Set duty cycle time: Period/2
}
```

21 USB Universal Serial Bus

Product line	USB
STC8H1K08 family	
STC8H1K28 family	
STC8H3K64S4 family	
STC8H3K64S2 family	
STC8H8K64U family	●
STC8H2K64T family	
STC8H4K64TLR family	
STC8H4K64TLCD family	
STC8H4K64LCD family	

STC8H series microcomputer integrates USB2.0/USB1.1 compatible full-speed USB, 6 bidirectional endpoints, supports 4 endpoint transmission modes (control transmission, interrupt transmission, batch transmission and synchronous transmission), each endpoint has a 64-byte buffer Area.

The USB module has a total of 1280 bytes of FIFO, the structure is as follows:



21.1 USB related registers

Symbol	Description	Address	Bit Address and Symbol								Reset Value
			B7	B6	B5	B4	B3	B2	B1	B0	

USBCLK	USB clock control register	DCH	ENCKM	PCKI[1:0]		CRE	TST_USB	TST_PHY	PHYTST[1:0]		0010,0000
USBDAT	USB Data register	ECH									0000,0000
USBCON	USB Control register	F4H	ENUSB	USBRST	PS2M	PUEN	PDEN	DFREC	DP	DM	0000,0000
USBADR	USB Address register	FCH	BUSY	AUTORD	UADR[5:0]						0000,0000

21.1.1 USB Control Register (USBCON)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
USBCON	DFH	ENUSB	USBRST	PS2M	PUEN	PDEN	DFREC	DP	DM

ENUSB: USB function and USB clock control bit
 0: Turn off USB function and USB clock
 1: Enable USB function and USB clock
ENRST: USB reset setting control bit
 0: Turn off USB reset settings
 1: Enable USB reset
PS2M: PS2 mode function control bit 0: disable PS2 mode function 1: enable PS2 mode function
PUEN: 1.5K pull-up resistor control bit on DP/DM port
 0: Disable the pull-up resistor
 1: Enable pull-up resistor
PDEN: 500K pull-down resistor control bit on DP/DM port
 0: Disable pull-down resistor
 1: Enable pull-down resistor
DFREC: Differential receive status bit (read only)
 0: The current DP/DM differential status is "0"
 1: The current DP/DM differential status is "1"
DP: D+ port status (read only when PS2 is 0, read and write when PS2 is 1)
 0: The current D+ is logic 0 level
 1: The current D+ is logic 1 level
DM: D-port status (read only when PS2 is 0, read and write when PS2 is 1)
 0: The current D- is logic 0 level
 1: The current D- is logic 1 level

21.1.2 USB clock control register (USBCLK)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
USBCLK	DEH	ENCKM	PCKI[1:0]		CRE	TST_USB	TST_PHY	PHYTST[1:0]	

ENCKM: PLL frequency multiplication control
 0: Disable PLL frequency multiplication
 1: Enable PLL frequency multiplication
PCKI[1:0]: PLL clock selection

PCKI[1:0]	PLL clock source
00	6M
01	12M(default)
10	24M
11	IRC/2

CRE: Clock chasing control bit
 0: Clock chasing prohibited

1: Enable clock tracking

TST_USB: USB test mode

0: Disable USB test mode

1: Enable USB test mode

TST_PHY: PHY test mode

0: Disable PHY test mode

1: Enable PHY test mode

PHYTST[1:0]: USB PHY test

PHYTST[1:0]	Mode	DP	DM
00	Mode 0: Normal	x	x
01	Mode 1: Force to “1”	1	0
10	Mode 2: Force to “0”	0	1
11	Mode 3: Force to single-ended “0”	0	0

21.1.3 USB Indirect Address Address Register (USBADDR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
USBADDR	EEH	BUSY	AUTORD	UADR[5:0]					

BUSY: USB register read busy flag bit

write 0: meaningless

Write 1: Start the read operation of USB indirect register, the address is set by USBADDR

Read 0: The data in USBDATA register is valid

Reading 1: The data in the USBDATA register is invalid, and the USB is reading the indirect register

AUTORD: USB register automatic reading flag, used for USB FIFO block reading and writing

0: Every time the indirect USB register is read, the BUSY flag must be written first

Write 1: When the software reads USBDATA, the next USB indirect register reading will start automatically (USBADDR remains unchanged)

UADR[5:0]: USB indirect register address

21.1.4 USB indirect address data register (USBDATA)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
USBDATA	EFH	UDAT[7:0]							

UDAT[7:0]: used to read and write USB registers indirectly

21.2 USB Controller Registers (SIE)

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
30H	UTRKCTL	UTRKSTS						
28H								
20H	FIFO0	FIFO1	FIFO2	FIFO3	FIFO4	FIFO5		
18H								
10H	INMAXP	CSR0 INCSR1	INCSR2	OUTMAXP	OUTCSR1	OUTCSR2	COUNT0 OUTCOUNT1	OUTCOUNT2
08H		INTROUT1E		INTRUSBE	FRAME1	FRAME2	INDEX	
00H	FADDR	POWER	INTRIN1	-	INTROUT1	-	INTRUSB	INTRIN1E

Symbol	Description	Addre	Bit Address and Symbol	Reset

		ss	B7	B6	B5	B4	B3	B2	B1	B0	Value
FADDR	USB function address register	00H	UPDATE	UADDR[6:0]							0000,0000
POWER	USB Power Management Register	01H	ISOUD	-	-	-	USBRST	USBRSU	USBSUS	ENSUS	0xxx,0000
INTRIN1	USB endpoint IN interrupt flag	02H	-	-	EP5INIF	EP4INIF	EP3INIF	EP2INIF	EP1INIF	EP0IF	xx00,0000
INTROUT1	USB endpoint OUT interrupt flag	04H	-	-	EP5OUTIF	EP4OUTIF	EP3OUTIF	EP2OUTIF	EP1OUTIF	-	xx00,000x
INTRUSB	USB power interrupt flag	06H	-	-	-	-	SOFIF	RSTIF	RSUIF	SUSIF	xxxx,0000
INTRIN1E	USB endpoint IN interrupt enable bit	07H	-	-	EP5INIE	EP4INIE	EP3INIE	EP2INIE	EP1INIE	EP0IE	xx11,1111
INTROUT1E	USB endpoint OUT interrupt enable bit	09H	-	-	EP5OUTIE	EP4OUTIE	EP3OUTIE	EP2OUTIE	EP1OUTIE	-	xx11,111x
INTRUSBE	USB power interruption enable bit	0BH	-	-	-	-	SOFIE	RSTIE	RSUIE	SUSIE	xxxx,0110
FRAME1	USB data frame number low byte	0CH	FRAME[7:0]								0000,0000
FRAME2	USB data frame number high byte	0DH	-	-	-	-	-	FRAME[10:8]			xxxx,x000
INDEX	USB Endpoint Index Register	0EH	-	-	-	-	-	INDEX[2:0]			xxxx,x000
INMAXP	Maximum packet size of IN endpoint	10H	INMAXP[7:0]								0000,0000
CSR0	Endpoint 0 control status register	11H	SSUEND	SOPRDY	SDSTL	SUEND	DATEND	STSTL	IPRDY	OPRDY	0000,0000
INCSR1	IN endpoint control status register 1	11H	CLRDT	STSTL	SDSTL	FLUSH	-	UNDRUN	FIFONE	IPRDY	0000,x000
INCSR2	IN endpoint control status register 2	12H	AUTOSET	ISO	MODE	ENDMA	FCDT	-	-	-	0010,0xxx
OUTMAXP	Maximum packet size of OUT endpoint	13H	OUTMAXP[7:0]								0000,0000
OUTCSR1	OUT endpoint control status register 1	14H	CLRDT	STSTL	SDSTL	FLUSH	DATERR	OVRRUN	FIFOFUL	OPRDY	0000,0000
OUTCSR2	OUT endpoint control status register 2	15H	AUTOCLR	ISO	ENDMA	DMAMD	-	-	-	-	0000,xxxx
COUNT0	OUT length of endpoint 0	16H	-	OUTCNT0[6:0]							x000,0000
OUTCOUNT1	USB endpoint OUT length low byte	16H	OUTCNT[7:0]								0000,0000
OUTCOUNT2	USB endpoint OUT length high byte	17H	-	-	-	-	-	OUTCNT[10:8]			xxxx,x000
FIFO0	FIFO access register of endpoint 0	20H	FIFO0[7:0]								0000,0000
FIFO1	FIFO access register of endpoint 1	21H	FIFO1[7:0]								0000,0000
FIFO2	FIFO access register of endpoint 2	22H	FIFO2[7:0]								0000,0000
FIFO3	FIFO access register of endpoint 3	23H	FIFO3[7:0]								0000,0000
FIFO4	FIFO access register of endpoint 4	24H	FIFO4[7:0]								0000,0000
FIFO5	FIFO access register of endpoint 5	25H	FIFO5[7:0]								0000,0000
UTRKCTL	USB tracking control register	30H	FTM1	FTM0	INTV[1:0]		ENST5	RES[2:0]			1011,1011
UTRKSTS	USB trace status register	31H	INTVCNT[3:0]				STS[1:0]		TST_UTRK	UTRK_RDY	1111,00x0

21.2.1 USB Function Address Register (FADDR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
FADDR	00H	UPDATE	UADDR[6:0]						

UPDATE: Update the USB function address
0: The last UADDR address has taken effect
1: The last UADDR address has not yet taken effect
UADDR[6:0]: Save the 7-bit function address of the USB

21.2.2 USB Power Control Register (POWER)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
POWER	01H	ISOUD	-	-	-	USBRST	USBRSU	USBSUS	ENSUS

ISOUD (ISO Update): ISO update
0: When the software writes "1" to IPRDY, the USB will send data packets after receiving the next IN token
1: When the software writes "1" to IPRDY, the USB sends a data packet after receiving the SOF token. If the IN token is received before the SOF token, the USB sends a data packet with a length of 0
USBRST (USB Reset): USB reset control bit
Writing "1" to this bit can force an asynchronous USB reset. Read this bit to get the reset status information on the current bus
0: No reset signal is detected on the bus
1: A reset signal is detected on the bus
USBRSU (USB Resume): USB recovery control bit
To force a resume signal to be generated on the bus in software mode to remotely wake up the USB device from the suspend mode. When USB is in suspend mode (USBSUS=1), writing "1" to this bit will force a resume signal to be generated on the USB bus. The software should write "0" to this bit after 10-15ms to end the resume signal . After the software writes "0" to USBRSU, a USB recovery interrupt will be generated, and the hardware will automatically clear USBSUS to "0"
USBSUS (USB Suspend): USB suspend control bit
When USB enters suspend mode, this bit is set to "1" by hardware. When the recovery signal is forced to be generated on the bus by software or the recovery signal is detected on the bus and the INTRUSB register is read, the hardware automatically clears this bit to "0".
ENSUS (Enable Suspend Detection): enable USB suspend detection
0: Suspend detection is disabled, USB will ignore the suspend signal on the bus
1: Enable suspend detection, when the suspend signal on the bus is detected, the USB will enter the suspend mode

21.2.3 USB endpoint IN interrupt flag Register (INTRIN1)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
INTRIN1	02H	-	-	EP5INIF	EP4INIF	EP3INIF	EP2INIF	EP1INIF	EP0IF

EP5INIF: IN interrupt flag bit of endpoint 5
0: IN interrupt of endpoint 5 is invalid
1: IN interrupt of endpoint 5 is valid
EP4INIF: IN interrupt flag bit of endpoint 4
0: IN interrupt of endpoint 4 is invalid
1: IN interrupt of endpoint 4 is valid
EP3INIF: Endpoint 3 IN interrupt flag bit
0: IN interrupt of endpoint 3 is invalid
1: The IN interrupt of endpoint 3 is valid

EP2INIF: Endpoint 2 IN interrupt flag bit
0: IN interrupt of endpoint 2 is invalid
1: IN interrupt of endpoint 2 is valid
EP1INIF: IN interrupt flag bit of endpoint 1
0: IN interrupt of endpoint 1 is invalid
1: IN interrupt of endpoint 1 is valid
EP0IF: IN/OUT interrupt flag bit of endpoint 0
0: IN/OUT interrupt of endpoint 0 is invalid
1: IN/OUT interrupt of endpoint 0 is valid
After the software reads the INTRIN1 register, the hardware will automatically clear all interrupt flags in INTRIN1

21.2.4 USB endpoint OUT interrupt flag Register (INTROUT1)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
INTROUT1	04H	-	-	EP5OUTIF	EP4OUTIF	EP3OUTIF	EP2OUTIF	EP1OUTIF	-

EP5OUTIF: Endpoint 5 OUT interrupt flag bit
0: The OUT interrupt of endpoint 5 is invalid
1: The OUT interrupt of endpoint 5 is valid
EP4OUTIF: Endpoint 4 OUT interrupt flag bit
0: The OUT interrupt of endpoint 4 is invalid
1: The OUT interrupt of endpoint 4 is valid
EP3OUTIF: Endpoint 3 OUT interrupt flag bit
0: The OUT interrupt of endpoint 3 is invalid
1: The OUT interrupt of endpoint 3 is valid
EP2OUTIF: Endpoint 2 OUT interrupt flag bit
0: The OUT interrupt of endpoint 2 is invalid
1: The OUT interrupt of endpoint 2 is valid
EP1OUTIF: Endpoint 1 OUT interrupt flag bit
0: The OUT interrupt of endpoint 1 is invalid
1: The OUT interrupt of endpoint 1 is valid
After the software reads the INTROUT1 register, the hardware will automatically clear all interrupt flags in INTROUT1

21.2.5 USB power interruption flag (INTRUSB)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
INTRUSB	06H	-	-	-	-	SOFIF	RSTIF	RSUIF	SUSIF

SOFIF: USB frame start signal interrupt flag
0: USB frame start signal interrupt is invalid
1: USB frame start signal interrupt is valid
RSTIF: USB reset signal interrupt flag
0: USB reset signal interrupt is invalid
1: USB reset signal interrupt is valid
RSUIF: USB recovery signal interruption flag
0: USB recovery signal interruption is invalid
1: USB recovery signal interruption is valid
SUSIF: USB suspend signal interrupt flag
0: USB suspend signal interrupt is invalid
1: USB suspend signal interrupt is valid
After the software reads the INTRUSB register, the hardware will automatically clear all interrupt flags in the INTRUSB

21.2.6 USB endpoint IN interrupt enable register (INTRIN1E)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
INTRIN1E	07H	-	-	EP5INIE	EP4INIE	EP3INIE	EP2INIE	EP1INIE	EP0IE

EP5INIE: IN interrupt control bit of endpoint 5
0: Disable the IN interrupt of endpoint 5
1: Enable IN interrupt of endpoint 5
EP4INIE: IN interrupt control bit of endpoint 4
0: Disable the IN interrupt of endpoint 4
1: Enable IN interrupt of endpoint 4
EP3INIE: IN interrupt control bit of endpoint 3
0: Disable the IN interrupt of endpoint 3
1: Enable IN interrupt of endpoint 3
EP2INIE: IN interrupt control bit of endpoint 2
0: Disable the IN interrupt of endpoint 2
1: Enable IN interrupt of endpoint 2
EP1INIE: IN interrupt control bit of endpoint 1
0: Disable the IN interrupt of endpoint 1
1: Enable IN interrupt of endpoint 1
EP0IE: IN/OUT interrupt control bit of endpoint 0
0: Disable the IN/OUT interrupt of endpoint 0
1: Enable IN/OUT interrupt of endpoint 0

21.2.7 USB endpoint OUT interrupt enable register (INTROUT1E)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
INTROUT1E	09H	-	-	EP5OUTIE	EP4OUTIE	EP3OUTIE	EP2OUTIE	EP1OUTIE	-

EP5OUTIE: OUT interrupt control bit of endpoint 5
0: Disable the OUT interrupt of endpoint 5
1: Enable OUT interrupt of endpoint 5
EP4OUTIE: OUT interrupt control bit of endpoint 4
0: Disable the OUT interrupt of endpoint 4
1: Enable OUT interrupt of endpoint 4
EP3OUTIE: Endpoint 3 OUT interrupt control bit
0: Disable the OUT interrupt of endpoint 3
1: Enable OUT interrupt of endpoint 3
EP2OUTIE: Endpoint 2 OUT interrupt control bit
0: Disable the OUT interrupt of endpoint 2
1: Enable OUT interrupt of endpoint 2
EP1OUTIE: OUT interrupt control bit of endpoint 1
0: Disable the OUT interrupt of endpoint 1
1: Enable OUT interrupt of endpoint 1

21.2.8 USB power interruption enable register (INTRUSB)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
INTRUSB	0BH	-	-	-	-	SOFIE	RSTIE	RSUIE	SUSIE

SOFIE: USB frame start signal interrupt control bit
0: Disable USB frame start signal interrupt
1: Allow USB frame start signal interrupt

RSTIE: USB reset signal interrupt control bit
 0: Disable USB reset signal interrupt
 1: Allow USB reset signal interrupt
RSUIE: USB recovery signal interrupt control bit
 0: Disable USB recovery signal interrupt
 1: Allow USB recovery signal interruption
SUSIE: USB suspend signal interrupt control bit
 0: Disable USB suspend signal interrupt
 1: Allow USB suspend signal interrupt

21.2.9 USB data frame number registers (FRAMEn)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
FRAME1	0CH	FRAME[7:0]							
FRAME2	0DH	-	-	-	-	-	FRAME[10:8]		

FRAME[10:0]: used to save the 11-bit frame number of the last received data frame

21.2.10 USB Endpoint Index Register (INDEX)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
INDEX	0EH	-	-	-	-	-	INDEX[2:0]		

INDEX[2:0]: select USB endpoint

INDEX[2:0]	Target endpoint
000	endpoint 0
001	endpoint 1
010	endpoint 2
011	endpoint 3
100	endpoint 4
101	endpoint 5

21.2.11 Maximum packet size of IN endpoint (INMAXP)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
INMAXP	10H	INMAXP[7:0]							

INMAXP[7:0]: Set the maximum data packet size of the USB IN endpoint
When you need to get/set this information, you must first use INDEX to select the target endpoint 0~5

21.2.12 USB Endpoint 0 Control Status Register (CSR0)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
CSR0	11H	SSUEND	SOPRDY	SDSTL	SUEND	DATEND	STSTL	IPRDY	OPRDY

SSUEND (Serviced Setup End)
 SETUP End event processing completed flag. After processing the SETUP end event (SUEND), the software needs to set the SSUEND flag bit, and the hardware will automatically clear the SUEND bit to “0” when it detects that SSUEND is written to “1”.

SOPRDY (Serviced OPRDY)

OPRDY event processing complete flag. After processing the data packet received from endpoint 0, the software needs to set the SOPRDY flag bit, and the hardware will automatically clear the OPRDY bit to "0" when it detects that SOPRDY is written to "1".

SDSTL (Send Stall)

When receiving wrong conditions or unsupported requests, you can write "1" to this bit to end the current data transmission. When the STALL signal is sent, the hardware automatically clears this bit to "0".

SUEND (Setup End)

nstallation package end flag. When a control transfer ends before the software writes "1" to the DATAEND bit, the hardware will set this read-only bit to "1". When the software writes a "1" to SSUEND, the hardware clears this bit to "0".

DATEND (Data End)

End of data. The software should write "1" to this bit in the following situations:
1. After sending the last data packet, the firmware writes "1" to IPRDY;
2. After sending a zero-length data packet, the firmware writes "1" to IPRDY;
3. After receiving the last data packet, the firmware writes "1" to SOPRDY;
This bit will be automatically cleared to "0" by hardware

STSTL (Sent Stall)

STALL signal transmission completed flag. After sending the STALL signal, the hardware will set this bit to "1". This bit must be cleared to "0" by software.

IPRDY (IN Packet Ready)

IN Data packet preparation complete flag. The software should "1" the bit after packing a data to be sent into the FIFO of endpoint 0.

When one of the following conditions occurs, the hardware clears this bit to "0":
1. When the data packet has been sent;
2. When the data packet is covered by a SETUP packet;
3. When the data packet is covered by an OUT packet;

OPRDY (OUT Packet Ready)

OUT packet preparation complete flag. When an OUT packet is received, the hardware will set the read-only bit to "1" and generate an interrupt. This bit is cleared to “0” only when the software writes “1” to the SOPRDY bit.

When you need to get/set this information, you must first use INDEX to select the target endpoint 0

21.2.13 IN endpoint control status register 1 (INCSR1)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
INCSR1	11H	CLRDT	STSTL	SDSTL	FLUSH	-	UNDRUN	FIFONE	IPRDY

CLRDT (Clear Data Toggle): Reset the IN data toggle bit.

When the IN endpoint needs to reset the data switch bit to "0" due to reconfiguration or STALL, the software needs to write "1" to this data bit.

STSTL (Sent Stall): STALL signal transmission completion flag.

After the STALL signal is sent, the hardware will set this bit to “1” (the FIFO is cleared at this time, and the IPRDY bit is cleared to “0”). This flag must be cleared to "0" by software.

SDSTL (Send Stall): STALL signal sending request bit.

Software should write "1" to this bit to generate STALL signal as a response to an IN token. Software should write "0" to this bit to end the STALL signal. This bit has no effect on ISO mode.

FLUSH (FIFO Flush): Clear the next data packet of the FIFO of the IN endpoint.

Writing "1" to this bit will clear the next data packet to be sent from the IN endpoint FIFO. The FIFO pointer is reset and the IPRDY bit is cleared to "0". If the FIFO contains multiple data packets, the software must write "1" to FLUSH for each data packet. When the FIFO is cleared, the hardware will clear the FLUSH bit to “0”.

UNDRUN (Data Underrun): insufficient data.

The function of this bit depends on the mode of the IN endpoint:

ISO mode: When IPRDY is "0" and a zero-length data packet is sent after receiving an IN token, this bit is set to "1".

Interrupt/Batch mode: When NAK is used as a response to an IN token, this bit is set to "1".

This bit must be cleared to "0" by software.

FIFONE (FIFO Not Empty): FIFO not empty flag of IN endpoint

0: FIFO of IN endpoint is empty

1: The FIFO of the IN endpoint contains one or more data packets

IPRDY (IN Packet Ready): IN packet preparation completion flag.

The software should "1" the bit after packing a data to be sent into the endpoint's FIFO. When one of the following conditions occurs, the hardware clears this bit to "0":

1. When the data packet has been sent;

2. Automatic setting is enabled (AUTOSET = ‘1’) and the FIFO data packet of endpoint IN reaches the value set by INMAXP;

3. If the endpoint is in synchronous mode and ISOUD is "1", the read value of IPRDY is always 0 before receiving the next SOF.

When you need to get/set this information, you must first use INDEX to select the target endpoint 1~5

21.2.14 IN endpoint control status register 2 (INCSR2)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
INCSR2	12H	AUTOSET	ISO	MODE	ENDMA	FCDT	-	-	-

AUTOSET: Automatically set the IPRDY flag control bit.

0: prohibit automatic setting of IPRDY flag

1: Enable automatic setting of IPRDY (the data loaded into the IN FIFO must reach the value set by INMAXP, otherwise the IPRDY flag must be manually set)

ISO: Synchronous transmission enable.

0: The endpoint is configured for bulk/interrupt transmission

1: The endpoint is configured for synchronous transmission

MODE: Endpoint direction selection bit.

0: Select the endpoint direction as OUT

1: Select the end direction as IN

ENDMA: DMA control of IN endpoint

0: Disable DMA request of IN endpoint

1: Enable DMA request for IN endpoint

FCDT: Force DATA0/DATA1 data switching settings.

0: Endpoint data is only switched after sending a data packet and receiving ACK.

1: Endpoint data is forced to switch after sending a data packet, regardless of whether it receives ACK.

When you need to get/set this information, you must first use INDEX to select the target endpoint 1~5

21.2.15 Maximum packet size of OUT endpoint (OUTMAXP)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
OUTMAXP	13H	OUTMAXP[7:0]							

OUTMAXP[7:0]: Set the maximum data packet size of the USB OUT endpoint

When you need to get/set this information, you must first use INDEX to select the target endpoint 1~5

21.2.16 OUT endpoint control status register 1 (OUTCSR1)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
OUTCSR1	14H	CLRDT	STSTL	SDSTL	FLUSH	DATERR	OVRRUN	FIFOFUL	OPRDY

CLRDT (Clear Data Toggle): Reset the OUT data toggle bit.

When the OUT endpoint needs to reset the data switch bit to "0" due to reconfiguration or STALL, the software needs to write "1" to this data bit.

STSTL (Sent Stall): STALL signal transmission completion flag.

When the STALL signal is sent, the hardware will set this bit to "1". This flag must be cleared to "0" by software.

SDSTL (Send Stall): STALL signal sending request bit.

The software should write "1" to this bit to generate STALL signal as a response to an OUT token. Software should write "0" to this bit to end the STALL signal. This bit has no effect on ISO mode.

FLUSH (FIFO Flush): Clear the next data packet of the FIFO of the OUT endpoint.

Writing "1" to this bit will clear the next data packet from the OUT endpoint FIFO. The FIFO pointer is reset and the OPRDY bit is cleared to "0". If the FIFO contains multiple data packets, the software must write "1" to FLUSH for each data packet. When the FIFO is empty,

The hardware clears the FLUSH bit to "0".

DATAERR (Data Error): data error.

In ISO mode, this bit is set to '1' by hardware if the received packet has a CRC or bit stuffing error. When software clears OPRDY, this bit is cleared to '0'. This bit is only valid in ISO mode.

OVRRUN (Data Overrun): Data overflow.

When an input packet cannot be loaded into the OUT endpoint FIFO, this bit is set to "1" by hardware. This bit is only valid in ISO mode. This bit must be cleared to "0" by software.

0: No data overflow

1: Since the flag was cleared for the last time, data packets have been lost due to full FIFO

FIFOFUL (FIFO Full): FIFO data full flag of OUT endpoint.

0: The FIFO of the OUT endpoint is not full

1: FIFO of OUT endpoint is full

OPRDY (OUT Packet Ready): OUT packet receiving completion flag.

The hardware sets this bit to "1" when there are data packets available. Software should clear this bit to '0' after unloading each data packet from the OUT endpoint FIFO.

When you need to get/set this information, you must first use INDEX to select the target endpoint 1~5

21.2.17 OUT endpoint control status register 2 (OUTCSR2)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
OUTCSR2	15H	AUTOCLR	ISO	~~ENDMA~~	~~DMAMD~~	-	-	-	-

AUTOCLR: Automatically clear the OPRDY flag control bit.

0: Prohibit automatic clearing of OPRDY flag

1: Enable automatic clearing of OPRDY (the data downloaded from OUT FIFO must reach the value set by OUTMAXP, otherwise the OPRDY flag must be cleared manually)

ISO: Synchronous transmission enable.

0: The endpoint is configured for bulk/interrupt transmission

1: The endpoint is configured for synchronous transmission

~~ENDMA: DMA control of OUT endpoint~~

~~0: Disable DMA request of OUT endpoint~~

~~1: Enable DMA request for OUT endpoint~~

~~DMAMD: Set the DMA mode of the OUT endpoint~~

When you need to get/set this information, you must first use INDEX to select the target endpoint 1~5

21.2.18 OUT length of USB endpoint 0 (COUNT0)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
COUNT0	16H	-	OUTCNT0[6:0]						

OUTCNT0[6:0]: OUT byte length of endpoint 0

COUNT0 is dedicated to save the data length of the last OUT packet received by endpoint 0 (because the longest packet of endpoint 0 can only be 64 bytes, only 7 bits are needed). This length value is only valid when the OPRDY bit of endpoint 0 is "1".

When you need to obtain this length information, you must first use INDEX to select the target endpoint 0

21.2.19 OUT length of USB endpoint (OUTCOUNTn))

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
OUTCOUNT1	16H	OUTCNT[7:0]							
OUTCOUNT2	17H	-	-	-	-	-	OUTCNT[10:8]		

OUTCNT[10:0]: OUT byte length of the endpoint

OUTCOUNT1 and OUTCOUNT2 are combined to form an 11-bit number, which saves the data length of the last OUT packet, which is suitable for endpoints 1~5. This length value is only valid when the OPRDY bit of endpoint 1~5 is "1".

When you need to obtain this length information, you must first use INDEX to select the target endpoint 1~5

21.2.20 FIFO Data Access Registers (FIFOn) of USB Endpoint

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
FIFO0	20H	FIFO0[7:0]							
FIFO1	21H	FIFO1[7:0]							
FIFO2	22H	FIFO2[7:0]							
FIFO3	23H	FIFO3[7:0]							
FIFO4	24H	FIFO4[7:0]							
FIFO5	25H	FIFO5[7:0]							

FIFOn[7:0]: Indirect access register of IN/OUT data of each endpoint of USB

21.3 USB Product Development Considerations

Each USB product must have its own unique VID&PID combination in order to be correctly recognized by the computer. If the VID&PID combinations corresponding to two different USB products are the same, the computer may not recognize the USB product abnormally, so that the USB product cannot be used normally. To avoid this situation, both VID and PID need to be uniformly planned and assigned through formal ways.

At present, STC has obtained the VID number 13503 (hexadecimal: 34BF) of STC's dedicated USB device through the USB-IF organization. When customers use STC's USB chip to develop their own USB products, if you have obtained your own VID through other ways, the corresponding PID can be planned by yourself. If your USB product needs to use the official VID of STC, you must apply to STC for the PID of the product.

21.4 Sample Routines

21.4.1 Example of HID man-machine interface equipment

C Language Code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

typedef     unsigned char           BYTE;
typedef     unsigned int            WORD;
typedef     unsigned long           DWORD;

sfr         IE2             =       0xAF;

sfr         USBCLK          =       0xDC;
sfr         USBDAT          =       0xEC;
sfr         USBCON          =       0xF4;
sfr         USBADR          =       0xFC;

sfr         P_SW2           =       0xBA;

sfr         P1M1            =       0x91;
sfr         P1M0            =       0x92;
sfr         P0M1            =       0x93;
sfr         P0M0            =       0x94;
sfr         P2M1            =       0x95;
sfr         P2M0            =       0x96;
sfr         P3M1            =       0xb1;
sfr         P3M0            =       0xb2;
sfr         P4M1            =       0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

#define     IRC48MCR                (*(unsigned char volatile xdata *)0xfe07)

#define     FADDR                   0
#define     POWER                   1
#define     INTRIN1                 2
#define     EP5INIF                 0x20
#define     EP4INIF                 0x10
#define     EP3INIF                 0x08
#define     EP2INIF                 0x04
#define     EP1INIF                 0x02
#define     EP0IF                   0x01
#define     INTROUT1                4
#define     EP5OUTIF                0x20
#define     EP4OUTIF                0x10
#define     EP3OUTIF                0x08
#define     EP2OUTIF                0x04
```

```
#define     EP1OUTIF        0x02
#define     INTRUSB         6
#define     SOFIF           0x08
#define     RSTIF           0x04
#define     RSUIF           0x02
#define     SUSIF           0x01
#define     INTRIN1E        7
#define     EP5INIE         0x20
#define     EP4INIE         0x10
#define     EP3INIE         0x08
#define     EP2INIE         0x04
#define     EP1INIE         0x02
#define     EP0IE           0x01
#define     INTROUT1E       9
#define     EP5OUTIE        0x20
#define     EP4OUTIE        0x10
#define     EP3OUTIE        0x08
#define     EP2OUTIE        0x04
#define     EP1OUTIE        0x02
#define     INTRUSBE        11
#define     SOFIE           0x08
#define     RSTIE           0x04
#define     RSUIE           0x02
#define     SUSIE           0x01
#define     FRAME1          12
#define     FRAME2          13
#define     INDEX           14
#define     INMAXP          16
#define     CSR0            17
#define     SSUEND          0x80
#define     SOPRDY          0x40
#define     SDSTL           0x20
#define     SUEND           0x10
#define     DATEND          0x08
#define     STSTL           0x04
#define     IPRDY           0x02
#define     OPRDY           0x01
#define     INCSR1          17
#define     INCLRDT         0x40
#define     INSTSTL         0x20
#define     INSDSTL         0x10
#define     INFLUSH         0x08
#define     INUNDRUN        0x04
#define     INFIFONE        0x02
#define     INIPRDY         0x01
#define     INCSR2          18
#define     INAUTOSET       0x80
#define     INISO           0x40
#define     INMODEIN        0x20
#define     INMODEOUT       0x00
#define     INENDMA         0x10
#define     INFCDT          0x08
#define     OUTMAXP         19
#define     OUTCSR1         20
#define     OUTCLRDT        0x80
#define     OUTSTSTL        0x40
#define     OUTSDSTL        0x20
#define     OUTFLUSH        0x10
```

```
#define     OUTDATERR           0x08
#define     OUTOVRRUN           0x04
#define     OUTFIFOFUL          0x02
#define     OUTOPRDY            0x01
#define     OUTCSR2             21
#define     OUTAUTOCLR          0x80
#define     OUTISO              0x40
#define     OUTENDMA            0x20
#define     OUTDMAMD            0x10
#define     COUNT0              22
#define     OUTCOUNT1           22
#define     OUTCOUNT2           23
#define     FIFO0               32
#define     FIFO1               33
#define     FIFO2               34
#define     FIFO3               35
#define     FIFO4               36
#define     FIFO5               37
#define     UTRKCTL             48
#define     UTRKSTS             49

#define     EPIDLE              0
#define     EPSTATUS            1
#define     EPDATAIN            2
#define     EPDATAOUT           3
#define     EPSTALL             -1

#define     GET_STATUS                  0x00
#define     CLEAR_FEATURE               0x01
#define     SET_FEATURE                 0x03
#define     SET_ADDRESS                 0x05
#define     GET_DESCRIPTOR              0x06
#define     SET_DESCRIPTOR              0x07
#define     GET_CONFIG                  0x08
#define     SET_CONFIG                  0x09
#define     GET_INTERFACE               0x0A
#define     SET_INTERFACE               0x0B
#define     SYNCH_FRAME                 0x0C

#define     GET_REPORT                  0x01
#define     GET_IDLE                    0x02
#define     GET_PROTOCOL                0x03
#define     SET_REPORT                  0x09
#define     SET_IDLE                    0x0A
#define     SET_PROTOCOL                0x0B

#define     DESC_DEVICE                 0x01
#define     DESC_CONFIG                 0x02
#define     DESC_STRING                 0x03
#define     DESC_HIDREPORT              0x22

#define     STANDARD_REQUEST            0x00
#define     CLASS_REQUEST               0x20
#define     VENDOR_REQUEST              0x40
#define     REQUEST_MASK                0x60

typedef struct
{
        BYTE        bmRequestType;
        BYTE        bRequest;
        BYTE        wValueL;
        BYTE        wValueH;
        BYTE        wIndexL;
        BYTE        wIndexH;
        BYTE        wLengthL;
        BYTE        wLengthH;
}SETUP;

typedef struct
{
        BYTE        bStage;
        WORD        wResidue;
        BYTE        *pData;
}EP0STAGE;

void UsbInit();
BYTE ReadReg(BYTE addr);
void WriteReg(BYTE addr, BYTE dat);
BYTE ReadFifo(BYTE fifo, BYTE *pdat);
void WriteFifo(BYTE fifo, BYTE *pdat, BYTE cnt);

char code DEVICEDESC[18];
char code CONFIGDESC[41];
char code HIDREPORTDESC[27];
char code LANGIDDESC[4];
char code MANUFACTDESC[8];
char code PRODUCTDESC[30];

SETUP Setup;
EP0STAGE Ep0Stage;
BYTE xdata HidFreature[64];
BYTE xdata HidInput[64];
BYTE xdata HidOutput[64];

void main()
{
        P0M0 = 0x00;
        P0M1 = 0x00;
        P1M0 = 0x00;
        P1M1 = 0x00;
        P2M0 = 0x00;
        P2M1 = 0x00;
        P3M0 = 0x00;
        P3M1 = 0x00;
        P4M0 = 0x00;
        P4M1 = 0x00;
        P5M0 = 0x00;
        P5M1 = 0x00;

        UsbInit();

        IE2 = 0x80;
        EA = 1;

        while (1);
}
```

```
BYTE ReadReg(BYTE addr)
{
        BYTE dat;

        while (USBADR & 0x80);
        USBADR = addr | 0x80;
        while (USBADR & 0x80);
        dat = USBDAT;
        return dat;
}

void WriteReg(BYTE addr, BYTE dat)
{
        while (USBADR & 0x80);
        USBADR = addr & 0x7f;
        USBDAT = dat;
}

BYTE ReadFifo(BYTE fifo, BYTE *pdat)
{
        BYTE cnt;
        BYTE ret;

        ret = cnt = ReadReg(COUNT0);
        while (cnt--)
        {
                *pdat++ = ReadReg(fifo);
        }
        return ret;
}

void WriteFifo(BYTE fifo, BYTE *pdat, BYTE cnt)
{
        while (cnt--)
        {
                WriteReg(fifo, *pdat++);
        }
}

void UsbInit()
{
        P3M0 = 0x00;
        P3M1 = 0x03;

        P_SW2 |= 0x80;
        IRC48MCR = 0x80;
        while (!(IRC48MCR & 0x01));
        P_SW2 &= ~0x80;
        USBCLK = 0x00;
        USBCON = 0x90;

        WriteReg(FADDR, 0x00);
        WriteReg(POWER, 0x08);
        WriteReg(INTRIN1E, 0x3f);
        WriteReg(INTROUT1E, 0x3f);
        WriteReg(INTRUSBE, 0x00);
        WriteReg(POWER, 0x01);
        Ep0Stage.bStage = EPIDLE;
}

void usb_isr() interrupt 25
{
        BYTE intrusb;
        BYTE intrin;
        BYTE introut;
        BYTE csr;
        BYTE cnt;
        WORD len;

        intrusb = ReadReg(INTRUSB);
        intrin = ReadReg(INTRIN1);
        introut = ReadReg(INTROUT1);

        if (intrusb  & RSTIF)
        {
                WriteReg(INDEX, 1);
                WriteReg(INCSR1, INCLRDT);
                WriteReg(INDEX, 1);
                WriteReg(OUTCSR1, OUTCLRDT);
                Ep0Stage.bStage = EPIDLE;
        }

        if (intrin & EP0IF)
        {
                WriteReg(INDEX, 0);
                csr = ReadReg(CSR0);
                if (csr & STSTL)
                {
                        WriteReg(CSR0, csr & ~STSTL);
                        Ep0Stage.bStage = EPIDLE;
                }
                if (csr & SUEND)
                {
                        WriteReg(CSR0, csr | SSUEND);
                }

                switch (Ep0Stage.bStage)
                {
                case EPIDLE:
                        if (csr & OPRDY)
                        {
                                Ep0Stage.bStage = EPSTATUS;
                                ReadFifo(FIFO0, (BYTE *)&Setup);
                                ((BYTE *)&Ep0Stage.wResidue)[0] = Setup.wLengthH;
                                ((BYTE *)&Ep0Stage.wResidue)[1]= Setup.wLengthL;
                                switch (Setup.bmRequestType & REQUEST_MASK)
                                {
                                case STANDARD_REQUEST:
                                        switch (Setup.bRequest)
                                        {
                                        case SET_ADDRESS:
                                                WriteReg(FADDR, Setup.wValueL);
                                                break;
                                        case SET_CONFIG:
```

```
        WriteReg(INDEX, 1);
        WriteReg(INCSR2, INMODEIN);
        WriteReg(INMAXP, 8);
        WriteReg(INDEX, 1);
        WriteReg(INCSR2, INMODEOUT);
        WriteReg(OUTMAXP, 8);
        WriteReg(INDEX, 0);
        break;
    case GET_DESCRIPTOR:
        Ep0Stage.bStage = EPDATAIN;
        switch (Setup.wValueH)
        {
        case DESC_DEVICE:
            Ep0Stage.pData = DEVICEDESC;
            len = sizeof(DEVICEDESC);
            break;
        case DESC_CONFIG:
            Ep0Stage.pData = CONFIGDESC;
            len = sizeof(CONFIGDESC);
            break;
        case DESC_STRING:
            switch (Setup.wValueL)
            {
            case 0:
                Ep0Stage.pData = LANGIDDESC;
                len = sizeof(LANGIDDESC);
                break;
            case 1:
                Ep0Stage.pData = MANUFACTDESC;
                len = sizeof(MANUFACTDESC);
                break;
            case 2:
                Ep0Stage.pData = PRODUCTDESC;
                len = sizeof(PRODUCTDESC);
                break;
            default:
                Ep0Stage.bStage = EPSTALL;
                break;
            }
            break;
        case DESC_HIDREPORT:
            Ep0Stage.pData = HIDREPORTDESC;
            len = sizeof(HIDREPORTDESC);
            break;
        default:
            Ep0Stage.bStage = EPSTALL;
            break;
        }
        if (len < Ep0Stage.wResidue)
        {
            Ep0Stage.wResidue = len;
        }
        break;
    default:
        Ep0Stage.bStage = EPSTALL;
        break;
    }
    break;
```

```
                case CLASS_REQUEST:
                    switch (Setup.bRequest)
                    {
                    case GET_REPORT:
                        Ep0Stage.pData = HidFreature;
                        Ep0Stage.bStage = EPDATAIN;
                        break;
                    case SET_REPORT:
                        Ep0Stage.pData = HidFreature;
                        Ep0Stage.bStage = EPDATAOUT;
                        break;
                    case SET_IDLE:
                        break;
                    case GET_IDLE:
                    case GET_PROTOCOL:
                    case SET_PROTOCOL:
                    default:
                        Ep0Stage.bStage = EPSTALL;
                        break;
                    }
                    break;
                default:
                    Ep0Stage.bStage = EPSTALL;
                    break;
                }

                switch (Ep0Stage.bStage)
                {
                case EPDATAIN:
                    WriteReg(CSR0, SOPRDY);
                    goto L_Ep0SendData;
                    break;
                case EPDATAOUT:
                    WriteReg(CSR0, SOPRDY);
                    break;
                case EPSTATUS:
                    WriteReg(CSR0, SOPRDY | DATEND);
                    Ep0Stage.bStage = EPIDLE;
                    break;
                case EPSTALL:
                    WriteReg(CSR0, SOPRDY | SDSTL);
                    Ep0Stage.bStage = EPIDLE;
                    break;
                }
            }
            break;
        case EPDATAIN:
            if (!(csr & IPRDY))
            {
L_Ep0SendData:
                cnt = Ep0Stage.wResidue > 64 ? 64 : Ep0Stage.wResidue;
                WriteFifo(FIFO0, Ep0Stage.pData, cnt);
                Ep0Stage.wResidue -= cnt;
                Ep0Stage.pData += cnt;
                if (Ep0Stage.wResidue == 0)
                {
                    WriteReg(CSR0, IPRDY | DATEND);
                    Ep0Stage.bStage = EPIDLE;
                }
                else
                {
                    WriteReg(CSR0, IPRDY);
                }
            }
            break;
        case EPDATAOUT:
            if (csr & OPRDY)
            {
                cnt = ReadFifo(FIFO0, Ep0Stage.pData);
                Ep0Stage.wResidue -= cnt;
                Ep0Stage.pData += cnt;
                if (Ep0Stage.wResidue == 0)
                {
                    WriteReg(CSR0, SOPRDY | DATEND);
                    Ep0Stage.bStage = EPIDLE;
                }
                else
                {
                    WriteReg(CSR0, SOPRDY);
                }
            }
            break;
        }
    }

    if (intrin & EP1INIF)
    {
        WriteReg(INDEX, 1);
        csr = ReadReg(INCSR1);
        if (csr & INSTSTL)
        {
            WriteReg(INCSR1, INCLRDT);
        }
        if (csr & INUNDRUN)
        {
            WriteReg(INCSR1, 0);
        }
    }

    if (introut & EP1OUTIF)
    {
        WriteReg(INDEX, 1);
        csr = ReadReg(OUTCSR1);
        if (csr & OUTSTSTL)
        {
            WriteReg(OUTCSR1, OUTCLRDT);
        }
        if (csr & OUTOPRDY)
        {
            ReadFifo(FIFO1, HidOutput);
            WriteReg(OUTCSR1, 0);

            WriteReg(INDEX, 1);
            WriteFifo(FIFO1, HidOutput, 64);
            WriteReg(INCSR1, INIPRDY);
        }
```

```
    }
}

char code DEVICEDESC[18] =
{
    0x12,                                   //bLength(18);
    0x01,                                   //bDescriptorType(Device);
    0x00,0x02,                              //bcdUSB(2.00);
    0x00,                                   //bDeviceClass(0);
    0x00,                                   //bDeviceSubClass0);
    0x00,                                   //bDeviceProtocol(0);
    0x40,                                   //bMaxPacketSize0(64);
    0xbf,0x34,                              //idVendor(34bf);
    0x01,0xf0,                              //idProduct(f001);
    0x00,0x01,                              //bcdDevice(1.00);
    0x01,                                   //iManufacturer(1);
    0x02,                                   //iProduct(2);
    0x00,                                   //iSerialNumber(0);
    0x01,                                   //bNumConfigurations(1);
};

char code CONFIGDESC[41] =
{
    0x09,                                   //bLength(9);
    0x02,                                   //bDescriptorType(Configuration);
    0x29,0x00,                              //wTotalLength(41);
    0x01,                                   //bNumInterfaces(1);
    0x01,                                   //bConfigurationValue(1);
    0x00,                                   //iConfiguration(0);
    0x80,                                   //bmAttributes(BUSPower);
    0x32,                                   //MaxPower(100mA);

    0x09,                                   //bLength(9);
    0x04,                                   //bDescriptorType(Interface);
    0x00,                                   //bInterfaceNumber(0);
    0x00,                                   //bAlternateSetting(0);
    0x02,                                   //bNumEndpoints(2);
    0x03,                                   //bInterfaceClass(HID);
    0x00,                                   //bInterfaceSubClass(0);
    0x00,                                   //bInterfaceProtocol(0);
    0x00,                                   //iInterface(0);

    0x09,                                   //bLength(9);
    0x21,                                   //bDescriptorType(HID);
    0x01,0x01,                              //bcdHID(1.01);
    0x00,                                   //bCountryCode(0);
    0x01,                                   //bNumDescriptors(1);
    0x22,                                   //bDescriptorType(HID Report);
    0x1b,0x00,                              //wDescriptorLength(27);

    0x07,                                   //bLength(7);
    0x05,                                   //bDescriptorType(Endpoint);
    0x81,                                   //bEndpointAddress(EndPoint1 as     IN);
    0x03,                                   //bmAttributes(Interrupt);
    0x40,0x00,                              //wMaxPacketSize(64);
    0x01,                                   //bInterval(10ms);

    0x07,                                   //bLength(7);
```

```
        0x05,                                           //bDescriptorType(Endpoint);
        0x01,                                           //bEndpointAddress(EndPoint1 as      OUT);
        0x03,                                           //bmAttributes(Interrupt);
        0x40,0x00,                                      //wMaxPacketSize(64);
        0x01,                                           //bInterval(10ms);
};

char code HIDREPORTDESC[27] =
{
        0x05,0x0c,                                      //USAGE_PAGE(Consumer);
        0x09,0x01,                                      //USAGE(Consumer Control);
        0xa1,0x01,                                      //COLLECTION(Application);
        0x15,0x00,                                      //      LOGICAL_MINIMUM(0);
        0x25,0xff,                                      //      LOGICAL_MAXIMUM(255);
        0x75,0x08,                                      //      REPORT_SIZE(8);
        0x95,0x40,                                      //      REPORT_COUNT(64);
        0x09,0x01,                                      //      USAGE(Consumer Control);
        0xb1,0x02,                                      //      FEATURE(Data,Variable);
        0x09,0x01,                                      //      USAGE(Consumer Control);
        0x81,0x02,                                      //      INPUT(Data,Variable);
        0x09,0x01,                                      //      USAGE(Consumer Control);
        0x91,0x02,                                      //      OUTPUT(Data,Variable);
        0xc0,                                           //END_COLLECTION;
};

char code LANGIDDESC[4] =
{
        0x04,0x03,
        0x09,0x04,
};

char code MANUFACTDESC[8] =
{
        0x08,0x03,
        'S',0,
        'T',0,
        'C',0,
};

char code PRODUCTDESC[30] =
{
        0x1e,0x03,
        'S',0,
        'T',0,
        'C',0,
        ' ',0,
        'U',0,
        'S',0,
        'B',0,
        ' ',0,
        'D',0,
        'e',0,
        'v',0,
        'i',0,
        'c',0,
        'e',0,
};
```

22 Touch Key Controller

Product line	Touch key
STC8H1K08 family	
STC8H1K28 family	
STC8H3K64S4 family	
STC8H3K64S2 family	
STC8H8K64U family	
STC8H2K64T family	●
STC8H4K64TLR family	●
STC8H4K64TLCD family	●
STC8H4K64LCD family	

A touch key controller or touch sensor unit (TSU in short) is integrated in the STC8H series of microcontrollers, which can connect up to 16 keys. It can detect the small capacitance changes caused by a finger touching the key electrode and quantify the capacitance as a 16-bit number. In principle, the TSU module is similar to a 16-bit ADC. The only difference is that the ADC detects and quantifies the simulated voltage or current, while the TSU detects and quantifies the capacitor. For touch-key sensing, a fixed capacitor of about 10nF to 47nF needs to be added for application as a reference. If the temperature of the environment changes rapidly and intensely, a low-temperature drift capacitor should be used to maintain the TSU output data no too much change, so as to avoid software misjudgment.

A description of continuous sensing of a certain key is shown in the following figure. The counter output value is near 2572 if there is no finger touch. The counter output becomes near 2443 if there is a finger touch. The difference of counter output value with or without finger touch is about 5%.

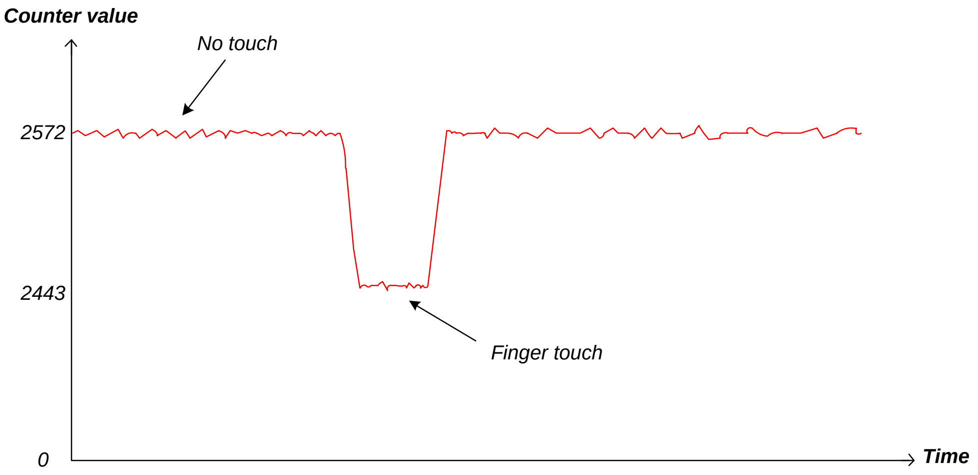


Fig. counter value for touch-sensing

The TSU module can connect up to 16 keys. Each touch key port can be independently enabled through the two 8-bit registers TSCHEN1 and TSCHEN2. The I/O port not used as touch key function can still maintain its original GPIO or special functions such as LED driver. The frequency of the switched capacitor circuit is selected through SCR [2: 0]. It is recommended not to exceed 12.5MHz to avoid large errors. There are four reference voltage segments of the internal comparator of the TSU module for choosing, which is selected through TSVR [1:0]. Changing the reference voltage will change the touch sensing time and sensitivity.

The touch key scan can be configured to scan continuously or to stop after one round of the enabled key. This function is controlled by the SINGLE bit. The configuration register TSSAMP [1: 0] allows the TSU module to scan four times continuously to one channel to sample and calculate the average value of the data. To do this has the effect of hardware filtering. The WAIT bit allows the TSU module to enter the wait mode when the TSIF flag is 1. TSU will continue to perform key scans until the software clears the WAIT bit to 0, which helps the heavy CPU to have enough time to process other things. The value of the external capacitor Cref is recommended to be in the range of 10nF to 47nF. The time from the initial value of Cref discharging to zero must be sufficient. This can be adjusted flexibly through DT [2: 0].

When a key scan is completed, the output value of the 16-bit counter will be written to TSDATAH and TSDATAL. The flag bit TSIF is set to 1 by hardware, and the scanned touch channel number will be written to TSDNCHN [3: 0]. If the module’s external interrupt controller is enabled, TSIF can request interrupt to the CPU. The software may read the TSDNCHN [3: 0] register content to determine which touch channel requested the TSIF interrupt. You can get the status of the TSU module being scanned and the number of keys being scanned in real time by reading TSWKCHN [3: 0] and TSGO using software. If the 16-bit counter overflows, the TSDOV flag will be set to 1 by hardware.

The TSU module can use I/O port in time-sharing multiplexing with the LED driver circuit. It is necessary to enable the LED driving circuit together and time-multiplex the I/O port when the content of the TSRT register is not zero, which means that TSGO enables TSU.Therefore, the relevant registers of the TSU module and the LED driving circuit must be configured in order firstly before TSGO is enabled. In TSU / LED time-sharing multi-tasking mode, in order to maintain the fixed frame rate of the LED, if the time allocated to the key scan has arrived but the key being scanned has not been completed, the incomplete key will be 851escanned to start a new key scan.

22.1 Internal Structure Diagram of Touch Key Controller

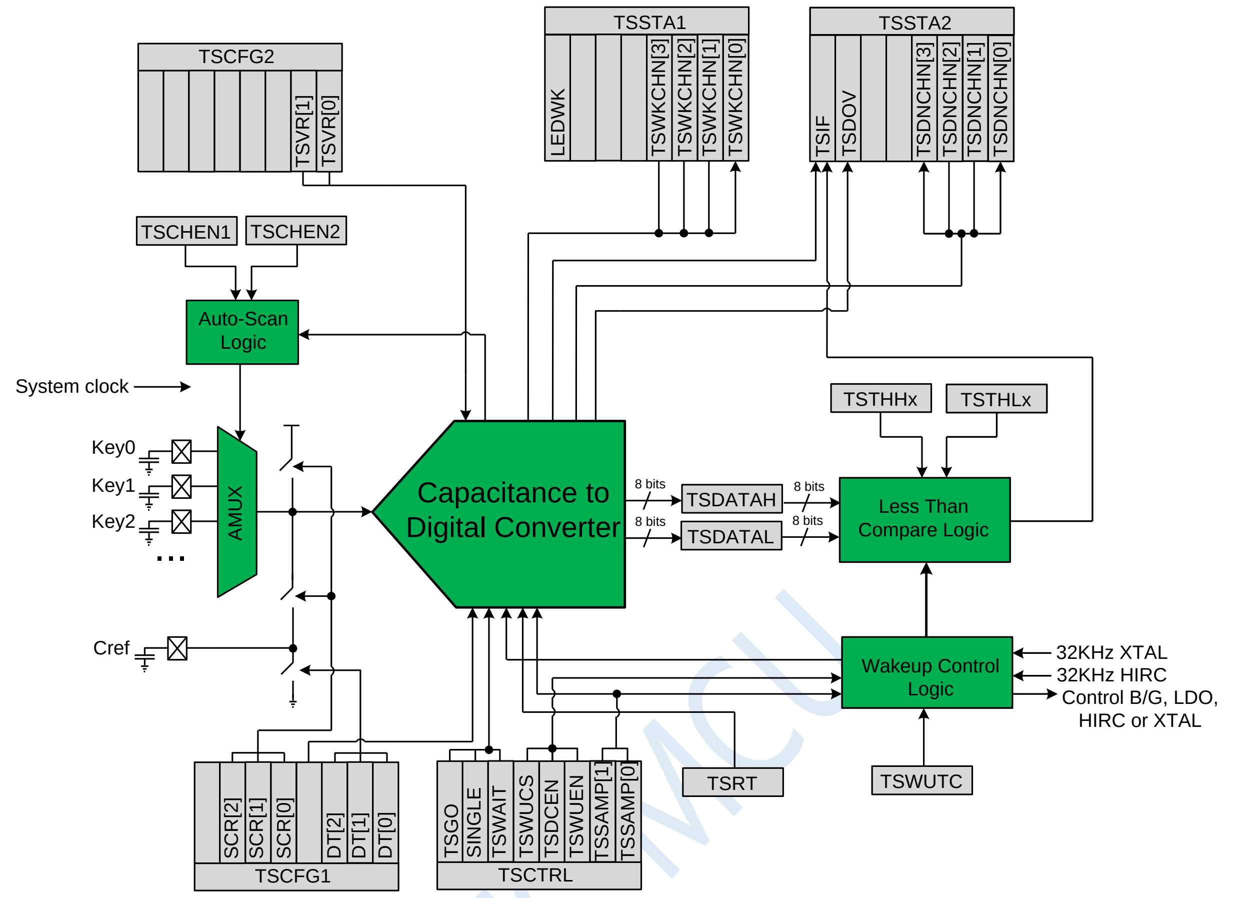


Fig. Touch-Sense Unit (TSU) block diagram

22.2 Wake-up from Low Power Mode using Touch Key

The TSU module has a dedicated timing and control circuit, which can accept an external 32KHz crystal or internal 32KHz RC oscillation circuit as a clock source, and wake up the TSU module at regular intervals to perform key scans, and realize the low-power touch wake-up function by duty control. There is a dedicated 16-bit threshold register {TSTHHx, TSTHLx} in each touch channel. If the wake-up enable bit TSWUEN and the digital comparator enable bit TSDCEN are set, MCU will enter the power-down state, and the entire chip enters a low-power state. In touch wake-up mode, the TSU module can repeatedly and regularly self-wake for key scans to wake up the CPU. If the data (or average result) of the key scan is less than the set threshold, the hardware will set TSIF to 1 and wake up the CPU out of power-down state. There is a hardware averaging circuit inside the TSU module, which can average the maximum of four consecutive scans of the same channel. The TSSAMP [1: 0] register is used to configure the number of samples. {TSDATAH, TSDATAL} stores average result.

22.3 Operation Steps When Touch Key Function is Used only

1. Turn on the TSU power switch TSPD = 0, select the channel to be scanned, and the registers are TSCHEN1 and TSCHEN2.
2. Set the TSRT content to 0x00, which means that the LED driver time-sharing multi-tasking function is not enabled.
3. Configure the switching frequency SCR [2: 0] and discharge time DT [2: 0] according to the value of Cref and the touch key capacitor, and select the internal comparator reference voltage TSVR [1: 0] according to the required scan time and sensitivity.

4. Configure the SINGLE bit to determine whether the scan is stopped automatically or continuously. Configure TSSAMP [1: 0] to allow a channel to be resampled up to four times. If the CPU task is heavy, configure TSWAIT to use the TSIF status to delay scanning of the next channel.
5. Configure TSDCEN to enable internal digital comparison function if necessary.
6. Set TSGO=1 to start the touch key scanning. You can read TSWKCHN [3: 0] to know which channel is currently being scanned using software. After finish scanning a channel, the hardware will set TSIF to 1, and the completed channel number will be written into TSDNCHN [3: 0]. If an overflow occurs, TSOV will also be set to 1. You should read these registers to decide what to do next in your software. TSIF and TSOV can only be set by hardware and cleared by software.
7. If SINGLE = 1, the hardware will clear TSGO automatically and end scanning after one round of scanning, otherwise TSGO will remain at 1 and continue the new scanning.
8. If you want to stop the touch key scanning, you can set TSGO to 0 at any time. If you want to reduce power consumption, you must set TSPD to 1.

22.4 Operation Steps for Wake-up from Low Power Mode using Touch Key

1. Select the channel to be scanned, the registers are TSCHEN1 and TSCHEN2.
2. Be sure to set the TSRT content to 0x00. At this time, you cannot turn on the LED driver time-sharing multitasking function.
3. Configure switching frequency SCR [2: 0], discharge time DT [2: 0] and select internal comparator reference voltage TSVR [1: 0].
4. Set the SINGLE bit to 0 for continuous scanning and set TSWAIT to 0.
5. Configure TSWUCS to determine which clock source is used to wake up the controller, external 32KHz crystal or internal 32KHz IRC.
6. Configure TSWUTC to determine how often TSU needs to work and enter the power saving mode automatically after work.
7. Configure TSSAMP [1: 0] to determine the number of scan samples for each channel, configure TSDCEN = 1 to enable the internal digital comparison function.
8. Configure the wake-up threshold {TSTHHx, TSTHLx} for each channel, which will be compared with the average of the scan results.
9. Enable TSWUEN = 1, set TSPD = 1 to turn off the analog power of the TSU module, enable TSIF to wake up the CPU, then let the MCU enter the power-down state. Once the MCU enters the power-down state, the wake-up controller inside the TSU starts to work, and controls the power switch, key scan, data comparison, and so on of the TSU module periodically.
10. If the data result is lower than the set threshold, the hardware will set TSIF to 1, write the key number in TSDNCHN [3: 0], and the CPU will be woken up, the low power wake-up process ends.
11. After the CPU is awakened, in addition to directly reading TSDNCHN [3: 0] to determine which key is touched, you can also perform a key scan in the normal working mode to confirm whether the wake-up is caused by noise interference.

22.5 Registers Related to Touch Key Controller

Symbol	Description	Address	Bit Address and Symbol								Reset Value
			B7	B6	B5	B4	B3	B2	B1	B0	
TSCHEN1	Touch Key Enable Register 1	FB40H	TKEN7	TKEN6	TKEN5	TKEN4	TKEN3	TKEN2	TKEN1	TKEN0	0000,0000
TSCHEN2	Touch Key Enable Register 2	FB41H	TKEN15	TKEN14	TKEN13	TKEN12	TKEN11	TKEN10	TKEN9	TKEN8	0000,0000
TSCFG1	Touch Key Configuration Register 1	FB42H	-	SCR[2:0]			-	DT[2:0]			0000,0000
TSCFG2	Touch Key Configuration Register 2	FB43H	-	-	-	-	-	-	TSVR[1:0]		0000,0000
TSWUTC	Touch key power-down mode wake-up time control register	FB44H									0000,0000
TSCTRL	Touch Key Control Register	FB45H	TSGO	SINGLE	TSWAIT	TSWUCS	TSDCEN	TSWUEN	TSSAMP[1:0]		0000,0000
TSSTA1	Touch Key Status Register 1	FB46H	LEDWK	-	-	-	TSWKCHN[3:0]				0000,0000
TSSTA2	Touch Key Status Register 2	FB47H	TSIF	TSDOV	-	-	TSDNCHN[3:0]				0000,0000

TSRT	Touch Key Time Control Register	FB48H		0000,0000
TSDATH	Touch Key Data High Byte	FB49H		0000,0000
TSDATL	Touch Key Data Low Byte	FB4AH		0000,0000
TSTH00H	Touch Key0 Threshold High Byte	FB50H		0000,0000
TSTH00L	Touch Key0 Threshold Low Byte	FB51H		0000,0000
TSTH01H	Touch Key1 Threshold High Byte	FB52H		0000,0000
TSTH01L	Touch Key1 Threshold Low Byte	FB53H		0000,0000
TSTH02H	Touch Key2 Threshold High Byte	FB54H		0000,0000
TSTH02L	Touch Key2 Threshold Low Byte	FB55H		0000,0000
TSTH03H	Touch Key3 Threshold High Byte	FB56H		0000,0000
TSTH03L	Touch Key3 Threshold Low Byte	FB57H		0000,0000
TSTH04H	Touch Key4 Threshold High Byte	FB58H		0000,0000
TSTH04L	Touch Key4 Threshold Low Byte	FB59H		0000,0000
TSTH05H	Touch Key5 Threshold High Byte	FB5AH		0000,0000
TSTH05L	Touch Key5 Threshold Low Byte	FB5BH		0000,0000
TSTH06H	Touch Key6 Threshold High Byte	FB5CH		0000,0000
TSTH06L	Touch Key6 Threshold Low Byte	FB5DH		0000,0000
TSTH07H	Touch Key7 Threshold High Byte	FB5EH		0000,0000
TSTH07L	Touch Key7 Threshold Low Byte	FB5FH		0000,0000
TSTH08H	Touch Key8 Threshold High Byte	FB60H		0000,0000
TSTH08L	Touch Key8 Threshold Low Byte	FB61H		0000,0000
TSTH09H	Touch Key9 Threshold High Byte	FB62H		0000,0000
TSTH09L	Touch Key9 Threshold Low Byte	FB63H		0000,0000
TSTH10H	Touch Key10 Threshold High Byte	FB64H		0000,0000
TSTH10L	Touch Key10 Threshold Low Byte	FB65H		0000,0000
TSTH11H	Touch Key11 Threshold High Byte	FB66H		0000,0000
TSTH11L	Touch Key11 Threshold Low Byte	FB67H		0000,0000
TSTH12H	Touch Key12 Threshold High Byte	FB68H		0000,0000
TSTH12L	Touch Key12 Threshold Low Byte	FB69H		0000,0000
TSTH13H	Touch Key13 Threshold High Byte	FB6AH		0000,0000
TSTH13L	Touch Key13 Threshold Low Byte	FB6BH		0000,0000
TSTH14H	Touch Key14 Threshold High Byte	FB6CH		0000,0000
TSTH14L	Touch Key14 Threshold Low Byte	FB6DH		0000,0000
TSTH15H	Touch Key15 Threshold High Byte	FB6EH		0000,0000
TSTH15L	Touch Key15 Threshold Low Byte	FB6FH		0000,0000

22.5.1 Touch Key Enable Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
TSCHEN1	FB40H	TKEN7	TKEN6	TKEN5	TKEN4	TKEN3	TKEN2	TKEN1	TKEN0
TSCHEN2	FB41H	TKEN15	TKEN14	TKEN13	TKEN12	TKEN11	TKEN10	TKEN9	TKEN8

TKENn: Touch key enable bit (n＝0~15)

0: Corresponding TKn pin is GPIO

1: Corresponding TKn pin is a touch key

22.5.2 Touch Key Configuration Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
TSCFG1	FB42H	-	SCR[2:0]			-	DT[2:0]		
TSCFG2	FB43H	-	-	-	-	-	-	TSVR[1:0]	

SCR: Configure the switching capacitor working frequency inside the touch key controller (the higher the frequency, the shorter the charging time)

$$\text{Working frequency of the switching capacitor} = \frac{\text{System working frequency}}{2 * (\text{SCR[2:0]} + 1)}$$

DT[2:0]: Configure the initial discharge time of Cref inside the touch key controller to ground

DT[2:0]	discharge time
000	125 system clocks
001	250 system clocks
010	500 system clocks
011	1000 system clocks
100	2000 system clocks
101	2500 system clocks
110	5000 system clocks
111	7500 system clocks

TSCV[1:0]: Configure the reference voltage inside the touch key controller

TSCV[1:0]	reference voltage
00	1/4 AVCC
01	1/2 AVCC
10	5/8 AVCC
11	3/4 AVCC

22.5.3 Touch key power-down mode wake-up time control register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
TSWUTC	FB44H								

TSWUTC register is used to configure how often the touch key controller is woken up.

$$\text{Wake-up frequency} = \frac{F_{32K}}{32 * 8 * \text{TSWUTC[7:0]}}$$

For example: if a external 32.768KHz crystal is used and TSWUTC = 0x80,
Then the wake-up frequency of the touch key controller is 32768 / (32 * 8 * 0x80) = 1Hz, that is to wake up once every 1 second.

Note: If the wake-up frequency is set too fast, the wake-up time is not enough to complete a round of key scan, and the touch key controller will scan continuously and cannot enter the power saving mode.

22.5.4 Touch Key Control Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
TSCTRL	FB45H	TSGO	SINGLE	TSWAIT	TSWUCS	TSDCEN	TSWUEN	TSSAMP[1:0]	

TSGO: Touch key controller start control bit

0: Touch key controller does not work.
1: Touch key controller start to work.

SINGLE: Single scan mode control bit

0: Repeat scan mode
1: Single scan mode. If TSGO=1, the hardware will clear TSGO to 0 automatically to terminate the scan after completing a round of key scan.

Note: IF TSGO = 1 and TSRT is not 0, it means that the touch button controller and LED driver share GPIO and time-sharing and multiplexing. At this time, the SINGLE control bit is invalid.

TSWAIT: Touch key controller waiting for control

0: touch key controller repeats scanning automatically

1: After completing one round of scanning, the TSIF is set to 1 by the hardware. At this time, the touch key controller will pause scanning until the TSIF flag is cleared to 0 and start the next round of scanning.

TSWUCS: clock source selection of touch key controller in low power mode

0: The clock source of the touch key controller in the low power mode is the internal 32K IRC.

1: The clock source of touch key controller in low power mode is external 32K crystal.

TSDCEN: 16-bit digital comparator inside the touch key controller control bit

0: disable 16-bit digital comparator inside the touch key controller

1: enable 16-bit digital comparator inside the touch key controller

Note: If the internal digital comparator of TSU is enabled, the TSIF will be set to 1 only when the touch sensing data result {TSDATAH, TSDATAL} is less than the threshold {TSTHHx, TSTHLx} set for the corresponding channel. This function is used to wake up the CPU by touch in low power mode.

TSWUEN: touch key controller low power wake up enable bit

0: disable touch key controller low power wake up function

1: enable touch key controller low power wake up function. After enabled, when the MCU enters the power-down state, it enters the touch-key low-power wake-up MCU mode immediately. In this mode, the low-power timing control circuit inside the touch key controller will enable the TSU periodically to perform key touch scanning. The duty control is used to maintain extremely low average current. TSWUEN is only effective when the MCU enters power-down mode and TSIF is 0.

Note: There are two 32K oscillators in this chip, one is an external 32K crystal, and the other is an internal IRC32K oscillator. In the case of non-STOP-mode, the internal IRC32K enable mechanism is simply using XFR: IRC32KCR [7], the external X32K enable mechanism is simply using XFR: X2KCR [7]. In the case of STOP-mode, the internal IRC32K enable mechanism is that in addition to XFR: IRC32KCR [7] must be set to 1, SFR: ENWKT or XFR: TSWUEN must also be set to 1, the external X32K enable mechanism is that in addition to XFR : X32KCR [7] must be set to 1, XFR: TSWUEN must also be set to 1. The key point to emphasize is that TSWUEN also plays the role of enabling or disabling the 32K oscillator in the case of STOP-mode.

TSSAMP[1:0]: single touch channel repeats scan times setting bits

TSSAMP [1:0]	Repeat scan times
00	once
01	2 times
10	3 times
11	4 times

Note: The interrupt flag TSIF will only be set when the number of scans of the same key reaches the configuration of TSSAMP. At this time, the average value of the results is written in {TSDATAH, TSDATAL}. However, if any overflow occurs, the hardware will set TSDOV to 1. With TSWKEN enabled, the average value must be less than the threshold to cause the interrupt flag TSIF to be set to 1 to wake the CPU.

22.5.5 Touch Key Status Register 1

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
TSSTA1	FB46H	LEDWK	-	-	-	TSWKCHN[3:0]			

LEDWK: working state of touch key controller and LED driver in time-sharing operation

0: The LED driver is in waiting state and the touch key controller is in working state.

1: The LED driver is in working state and the touch key controller is in waiting state.

TSWKCHN [3:0]: touch channel scan status

TSWKCHN [3:0]	Touch channel scan status
0000	Touch channel 0 is being scanned
0001	Touch channel 1 is being scanned
0010	Touch channel 2 is being scanned

0011	Touch channel 3 is being scanned
0100	Touch channel 4 is being scanned
0101	Touch channel 5 is being scanned
0110	Touch channel 6 is being scanned
0111	Touch channel 7 is being scanned
1000	Touch channel 8 is being scanned
1001	Touch channel 9 is being scanned
1010	Touch channel 10 is being scanned
1011	Touch channel 11 is being scanned
1100	Touch channel 12 is being scanned
1101	Touch channel 13 is being scanned
1110	Touch channel 14 is being scanned
1111	Touch channel 15 is being scanned

22.5.6 Touch Key Status Register 2

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
TSSTA2	FB47H	TSIF	TSDOV	-	-	TSDNCHN[3:0]			

TSIF: touch key channel scanning completion flag (cleared by writing 1 using software)

0: Scan has not completed.

1: When the number of scans set by TSSAMP is completed, TSIF is set by the hardware, and TSIF can request an interrupt to the CPU. If it is in the low-power wake-up mode, the scan data result value is lower than the set threshold at the same time, then TSIF will be set to 1.

Note: TSIF can only be set by hardware. Software cannot set TSIF to 1. It is important to note that writing 1 to TSIF in software clears TSIF to 0, and writing 0 to TSIF in software has no effect. If TSWAIT = 1 and TSIF is 1, the TSU is in the pause and wait state. The next key scan will be continued after the CPU is busy finished and clears the TSIF to 0.

TSDOV: Key scan data overflow flag (cleared by writing 1 using software)

0: The key scan data does not overflow, the scan data is less than or equal to 0xFFFF.

1: The key scan data overflows, and the scan data is greater than 0xFFFF. At this time, extreme software configurations (such as TSVR) or system hardware must be adjusted to avoid overflow. TSDOV can only be set to 1 by hardware. It is cleared by software writing 1 to it. There is no effect on it if software writing 0 to it.

TSDNCHN [3:0]: Touch channel completion status

TSDNCHN [3:0]	Touch channel completion status
0000	Scanning of touch channel 0 is completed
0001	Scanning of touch channel 1 is completed
0010	Scanning of touch channel 2 is completed
0011	Scanning of touch channel 3 is completed
0100	Scanning of touch channel 4 is completed
0101	Scanning of touch channel 5 is completed
0110	Scanning of touch channel 6 is completed
0111	Scanning of touch channel 7 is completed
1000	Scanning of touch channel 8 is completed
1001	Scanning of touch channel 9 is completed
1010	Scanning of touch channel 10 is completed
1011	Scanning of touch channel 11 is completed
1100	Scanning of touch channel 12 is completed
1101	Scanning of touch channel 13 is completed

1110	Scanning of touch channel 14 is completed
1111	Scanning of touch channel 15 is completed

22.5.7 Touch Key Time Control Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
TSRT	FB48H								

The TSRT register is used to configure the touch key controller and the LED driver to work in time-sharing.
If TSRT is not 00, the touch keycontroller and LED driver are in the time-sharing mode. The length of working time for the touch key controller is TSRT * T_{LED}. (Please refer to the LED driver description section for T_{LED})

22.5.8 Touch key data registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
TSDATH	FB49H	TSDAT[15:8]							
TSDATL	FB4AH	TSDAT[7:0]							

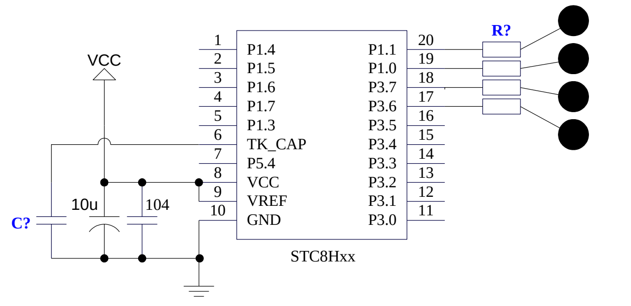
TSDAT[15:0]: Data scanned by touch key

22.5.9 Touch key threshold register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
TSTH00H	FB50H	TSTH00[15:8]							
TSTH00L	FB51H	TSTH00[7:0]							
TSTH01H	FB52H	TSTH01[15:8]							
TSTH01L	FB53H	TSTH01[7:0]							
TSTH02H	FB54H	TSTH02[15:8]							
TSTH02L	FB55H	TSTH02[7:0]							
TSTH03H	FB56H	TSTH03[15:8]							
TSTH03L	FB57H	TSTH03[7:0]							
TSTH04H	FB58H	TSTH04[15:8]							
TSTH04L	FB59H	TSTH04[7:0]							
TSTH05H	FB5AH	TSTH05[15:8]							
TSTH05L	FB5BH	TSTH05[7:0]							
TSTH06H	FB5CH	TSTH06[15:8]							
TSTH06L	FB5DH	TSTH06[7:0]							
TSTH07H	FB5EH	TSTH07[15:8]							
TSTH07L	FB5FH	TSTH07[7:0]							
TSTH08H	FB60H	TSTH08[15:8]							
TSTH08L	FB61H	TSTH08[7:0]							
TSTH09H	FB62H	TSTH09[15:8]							
TSTH09L	FB63H	TSTH09[7:0]							
TSTH10H	FB64H	TSTH10[15:8]							
TSTH10L	FB65H	TSTH10[7:0]							
TSTH11H	FB66H	TSTH11[15:8]							
TSTH11L	FB67H	TSTH11[7:0]							
TSTH12H	FB68H	TSTH12[15:8]							
TSTH12L	FB69H	TSTH12[7:0]							
TSTH13H	FB6AH	TSTH13[15:8]							
TSTH13L	FB6BH	TSTH13[7:0]							
TSTH14H	FB6CH	TSTH14[15:8]							
TSTH14L	FB6DH	TSTH14[7:0]							
TSTH15H	FB6EH	TSTH15[15:8]							
TSTH15L	FB6FH	TSTH15[7:0]							

TSTHn[15:0]: Touch key scan data threshold. After the digital comparator is enabled, TSIF will be set to 1 by hardware only when the scan data is below this threshold.

22.6 Basic Reference Circuit and Precautions



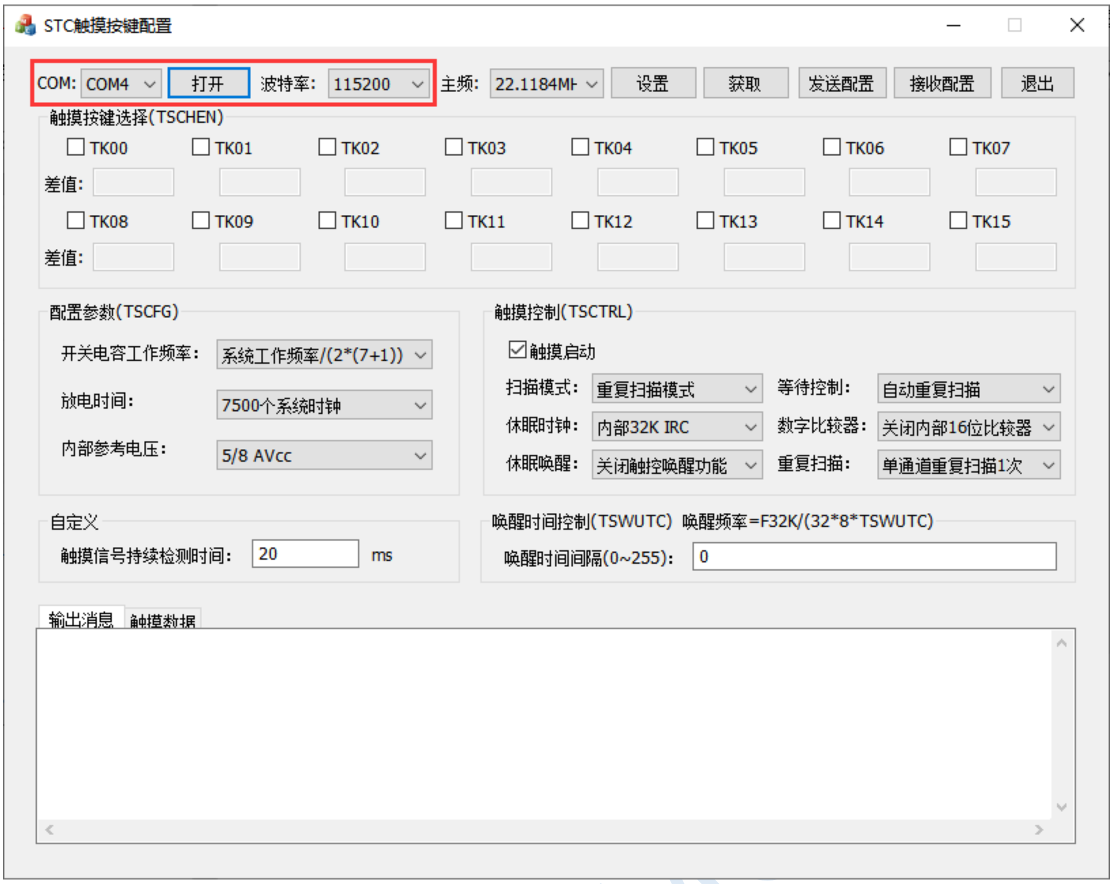
	Recommended value	Recommended package
C?	10nF ~ 47nF	0603
R?	470ohm ~ 1Kohm	0603

Note: In the reference circuit diagram, C? is the sensitivity adjustment capacitor for touch keys, and R? is the ESD protection resistor. In PCB layout, C? and R? must be as close as possible to the IC pins.

22.7 Example Routines

22.7.1 Introduction of touch key configuration software

1. Serial port settings
"COM": select the serial port number;
"Baud rate": Set the baud rate of serial communication, the default is '115200, N, 8, 1'. If the main frequency of the MCU system is set below 5.5296MHz, the baud rate needs to be reduced for normal communication, and 9600 is recommended. The MCU code needs to modify the definition of "Baudrate" synchronously.
"Open": After setting the serial port parameters, click this button to open the serial port, and the button name becomes "Close"; click it again to close the serial port, and the button name becomes "Open".



STC触摸按键配置

COM: COM4 打开 波特率: 115200 主频: 22.1184MH 设置 获取 发送配置 接收配置 退出

触摸按键选择(TSCHEN)

□ TK00 □ TK01 □ TK02 □ TK03 □ TK04 □ TK05 □ TK06 □ TK07

差值:

□ TK08 □ TK09 □ TK10 □ TK11 □ TK12 □ TK13 □ TK14 □ TK15

差值:

配置参数(TSCFG)

开关电容工作频率: 系统工作频率/(2*(7+1))

放电时间: 7500个系统时钟

内部参考电压: 5/8 AVcc

触摸控制(TSCTRL)

☑ 触摸启动

扫描模式: 重复扫描模式 等待控制: 自动重复扫描

休眠时钟: 内部32K IRC 数字比较器: 关闭内部16位比较器

休眠唤醒: 关闭触控唤醒功能 重复扫描: 单通道重复扫描1次

自定义

触摸信号持续检测时间: 20 ms

唤醒时间控制(TSWUTC) 唤醒频率=F32K/(32*8*TSWUTC)

唤醒时间间隔(0~255): 0

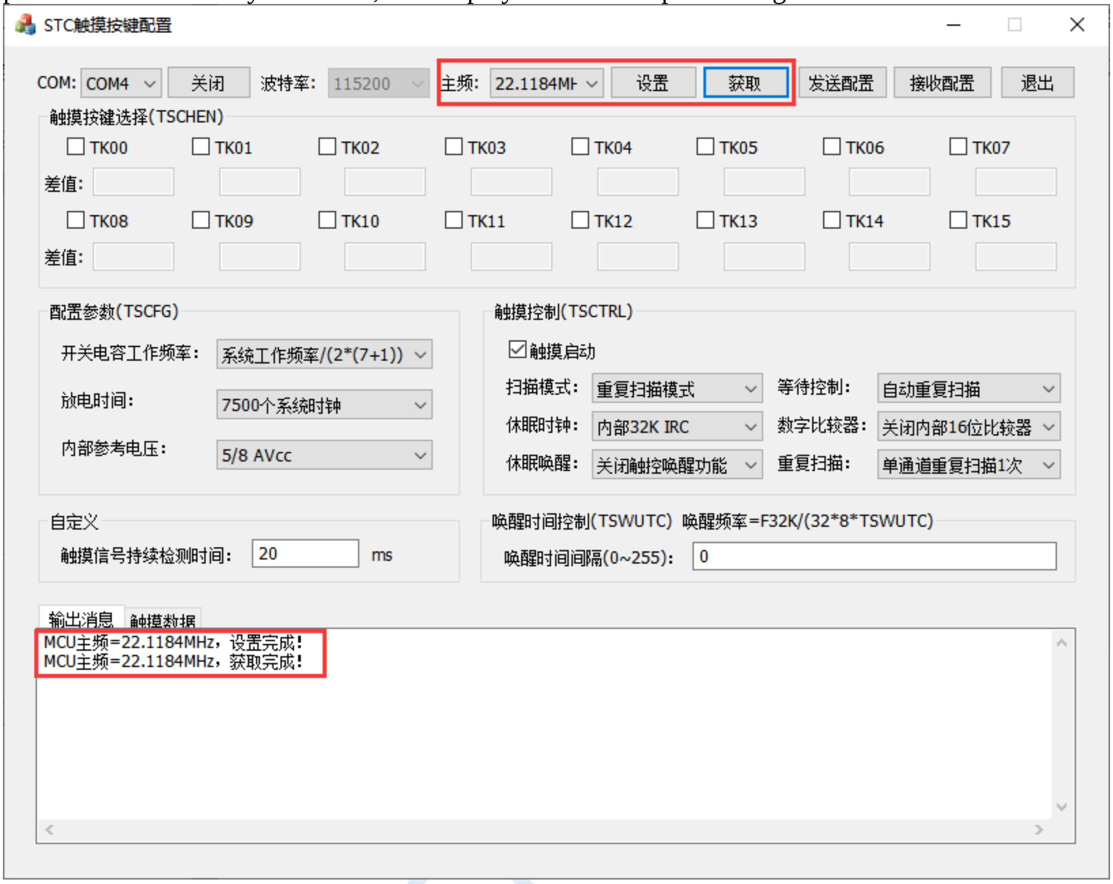
输出消息 触摸数据

2. MCU main frequency setting

"Frequency": drop-down box to select the MCU main frequency to be set.

"Set": The button sends the main frequency setting command to the MCU, and at the same time outputs the message box to display the message of setting the main frequency.

"Get": The button sends the main frequency acquisition command to the MCU, receives the main frequency parameters returned by the MCU, and displays it in the output message box.



COM: COM4

波特率: 115200

主频: 22.1184MH

☐ TK00 ☐ TK01 ☐ TK02 ☐ TK03 ☐ TK04 ☐ TK05 ☐ TK06 ☐ TK07

差值:

☐ TK08 ☐ TK09 ☐ TK10 ☐ TK11 ☐ TK12 ☐ TK13 ☐ TK14 ☐ TK15

差值:

开关电容工作频率: 系统工作频率/(2*(7+1))

放电时间: 7500个系统时钟

内部参考电压: 5/8 AVcc

☑触摸启动

扫描模式: 重复扫描模式

等待控制: 自动重复扫描

休眠时钟: 内部32K IRC

数字比较器: 关闭内部16位比较器

休眠唤醒: 关闭触控唤醒功能

重复扫描: 单通道重复扫描1次

触摸信号持续检测时间: 20 ms

唤醒时间间隔(0~255): 0

```
MCU主频=22.1184MHz，设置完成！
MCU主频=22.1184MHz，获取完成！
```

3. Touch Configuration Settings

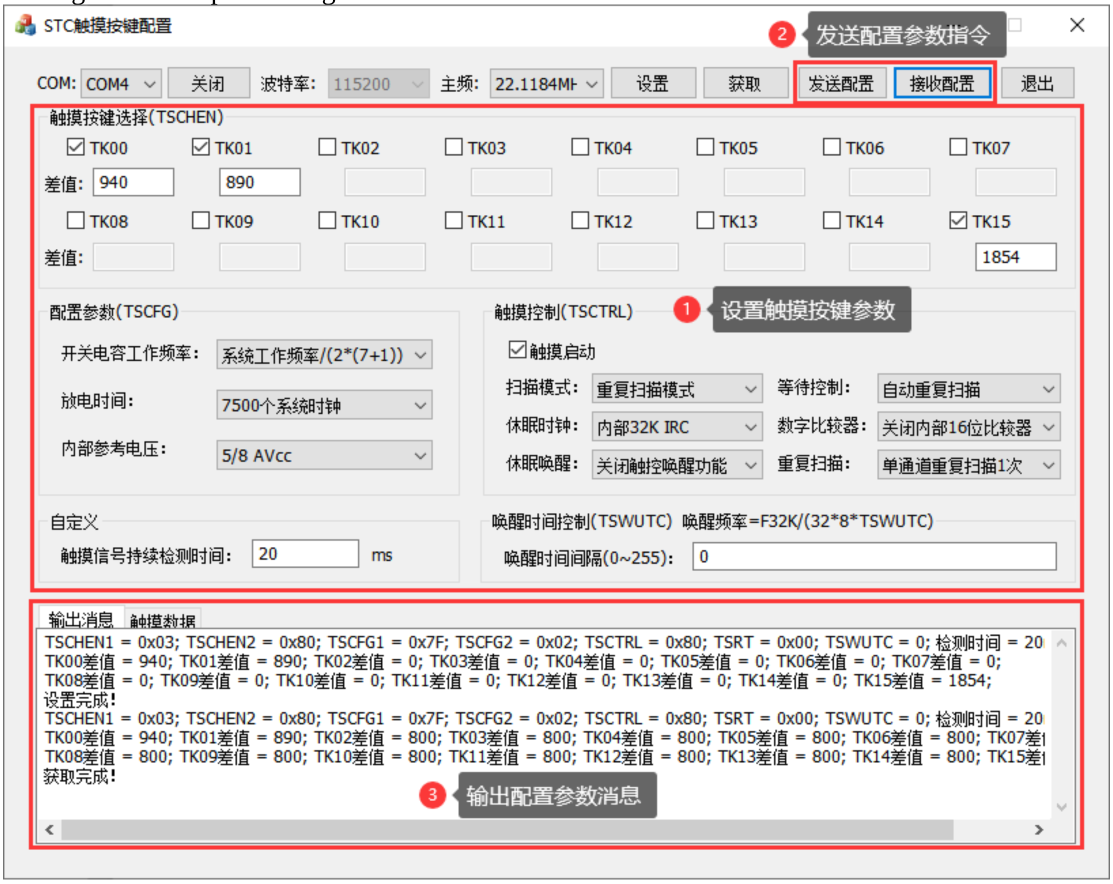
"Touch key selection": Check the channel that needs to be set as the touch key function, "Difference" is the key value difference before and after the touch key is pressed, and the MCU uses half of the difference as the threshold to judge whether the touch key is pressed.

"Configuration Parameters", "Touch Control": Set the corresponding register parameters through the drop-down box and check box.

"Custom": You can set the continuous detection time of the touch signal and the wake-up time interval.

"Send Configuration": Click this button to send the touch key configuration parameters to the MCU, and at the same time output the message box to display the touch key register parameter configuration message.

"Receive configuration": Click this button to send the obtained touch key configuration to the MCU, receive the configuration parameters returned by the MCU, and display the obtained touch key register parameter configuration message in the output message box.



COM: COM4 波特率: 115200 主频: 22.1184MH

② 发送配置参数指令

触摸按键选择(TSCHEN)

	☑ TK00	☑ TK01	□ TK02	□ TK03	□ TK04	□ TK05	□ TK06	□ TK07
差值:	940	890						
	□ TK08	□ TK09	□ TK10	□ TK11	□ TK12	□ TK13	□ TK14	☑ TK15
差值:								1854

配置参数(TSCFG)

开关电容工作频率: 系统工作频率/(2*(7+1))

放电时间: 7500个系统时钟

内部参考电压: 5/8 AVcc

触摸控制(TSCTRL)

① 设置触摸按键参数

☑ 触摸启动

扫描模式: 重复扫描模式 等待控制: 自动重复扫描

休眠时钟: 内部32K IRC 数字比较器: 关闭内部16位比较器

休眠唤醒: 关闭触控唤醒功能 重复扫描: 单通道重复扫描1次

自定义

触摸信号持续检测时间: 20 ms

唤醒时间控制(TSWUTC) 唤醒频率=F32K/(32*8*TSWUTC)

唤醒时间间隔(0~255): 0

```
TSCHEN1 = 0x03; TSCHEN2 = 0x80; TSCFG1 = 0x7F; TSCFG2 = 0x02; TSCTRL = 0x80; TSRT = 0x00; TSWUTC = 0; 检测时间 = 20
TK00差值 = 940; TK01差值 = 890; TK02差值 = 0; TK03差值 = 0; TK04差值 = 0; TK05差值 = 0; TK06差值 = 0; TK07差值 = 0;
TK08差值 = 0; TK09差值 = 0; TK10差值 = 0; TK11差值 = 0; TK12差值 = 0; TK13差值 = 0; TK14差值 = 0; TK15差值 = 1854;
设置完成!
TSCHEN1 = 0x03; TSCHEN2 = 0x80; TSCFG1 = 0x7F; TSCFG2 = 0x02; TSCTRL = 0x80; TSRT = 0x00; TSWUTC = 0; 检测时间 = 20
TK00差值 = 940; TK01差值 = 890; TK02差值 = 800; TK03差值 = 800; TK04差值 = 800; TK05差值 = 800; TK06差值 = 800; TK07差
TK08差值 = 800; TK09差值 = 800; TK10差值 = 800; TK11差值 = 800; TK12差值 = 800; TK13差值 = 800; TK14差值 = 800; TK15差
获取完成!
```

③ 输出配置参数消息

23 LED Driver

Product line	LED driver
STC8H1K08 family	
STC8H1K28 family	
STC8H3K64S4 family	
STC8H3K64S2 family	
STC8H8K64U family	
STC8H2K64T family	●
STC8H4K64TLR family	●
STC8H4K64TLCD family	
STC8H4K64LCD family	

An LED driver is integrated in STC8H series of microcontrollers.

The LED driver circuit includes a timing controller, 8 COM output pins and 8 SEGMENT output pins. Each pin has a corresponding register enable bit, which can independently control whether the pin is enabled or not. Pins that are not enabled can be used as pins for GPIO or other functions.

The LED driver supports three modes: common cathode, common anode, common cathode / common anode. At the same time, it can select 1/8 ~ 8/8 duty/cycle to adjust the gray scale. So only software is needed to adjust the LED and digital LED brightness.

After power-on reset, the enable bit LEDON is 0 and the LED driver is turned off. Set LEDON to 1 to enable the LED driver. If LEDMODE= 00, the driver works in the common cathode mode. At this time, the selected COM outputs a low level, the selected SEGMENTs to light LED output high level. Therefore, the forward bias of the LED between the two points of SEGMENT and COM turns on and lights up. Similarly, if LEDMODE = 01, the driver works in the common anode mode. At this time, the selected COM outputs a high level, and the selected SEGMENTs to light the LED output low level. The forward bias of the LED lights up. If LEDMODE = 10, the driver works in the common cathode / common anode time-sharing drive mode. The COM level is low and high-level time-sharing. The principle of LED lighting is the same as common cathod and common anode.

In the common cathode mode and the common anode mode, the display RAM address is independent. The location of display RAM address in the common cathode / common anode time-sharing mode is also read independently.

23.1 Internal Structure Diagram of LED Driver

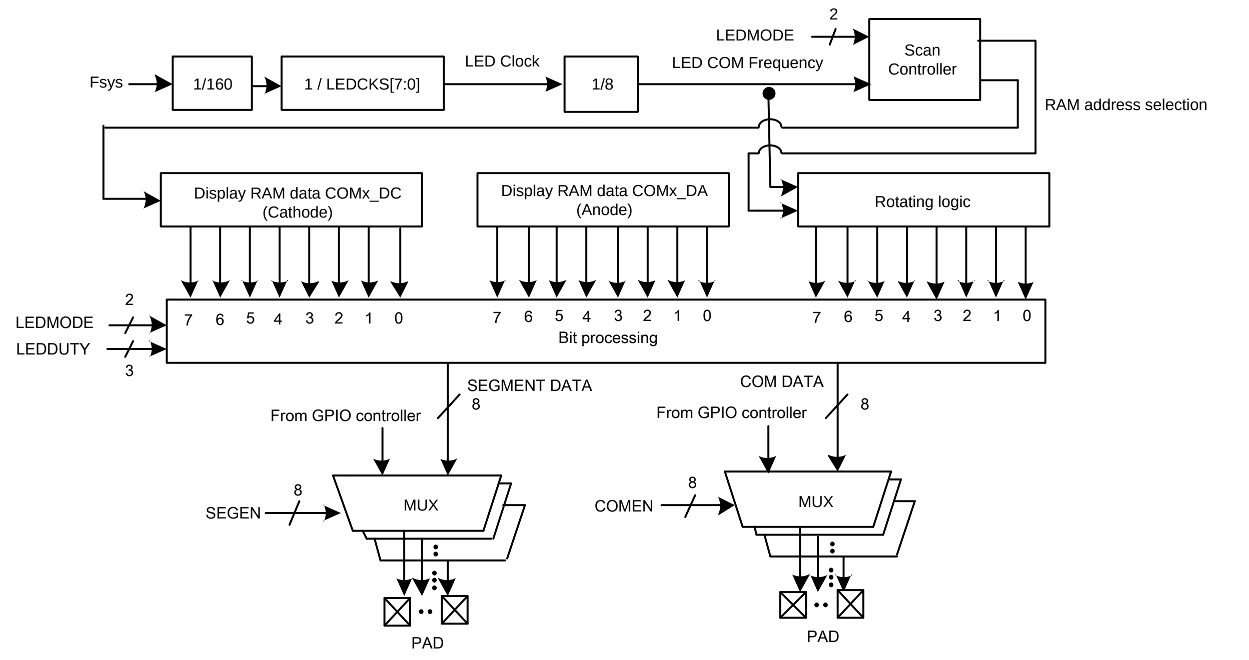


Fig. LED driver block diagram

23.2 Registers Related to LED Driver

Symbol	Description	Address	Bit Address and Symbol								Reset Value
			B7	B6	B5	B4	B3	B2	B1	B0	
COMEN	COM Enable Register	FB00H	C7EN	C6EN	C5EN	C4EN	C3EN	C2EN	C1EN	C0EN	0000,0000
SEGENL	SEG Enable Register	FB01H	S7EN	S6EN	S5EN	S4EN	S3EN	S2EN	S1EN	S0EN	0000,0000
SEGENH	SEG Enable Register	FB02H	S15EN	S14EN	S13EN	S12EN	S11EN	S10EN	S9EN	S8EN	0000,0000
LEDCTRL	LED Control Register	FB03H	LEDON	-	LEDMODE[1:0]		-	LEDDUTY[2:0]			0000,0000
LEDCKS	LED Clock Divide Register	FB04H									0000,0001
COM0_DA_L	Common Anode Mode Disply	FB10H									0000,0000
COM1_DA_L	Common Anode Mode Disply	FB11H									0000,0000
COM2_DA_L	Common Anode Mode Disply	FB12H									0000,0000
COM3_DA_L	Common Anode Mode Disply	FB13H									0000,0000
COM4_DA_L	Common Anode Mode Disply	FB14H									0000,0000
COM5_DA_L	Common Anode Mode Disply	FB15H									0000,0000
COM6_DA_L	Common Anode Mode Disply	FB16H									0000,0000
COM7_DA_H	Common Anode Mode Disply	FB17H									0000,0000
COM0_DA_H	Common Anode Mode Disply	FB18H									0000,0000
COM1_DA_H	Common Anode Mode Disply	FB19H									0000,0000
COM2_DA_H	Common Anode Mode Disply	FB1AH									0000,0000
COM3_DA_H	Common Anode Mode Disply	FB1BH									0000,0000
COM4_DA_H	Common Anode Mode Disply	FB1CH									0000,0000
COM5_DA_H	Common Anode Mode Disply	FB1DH									0000,0000
COM6_DA_H	Common Anode Mode Disply	FB1EH									0000,0000
COM7_DA_H	Common Anode Mode Disply	FB1FH									0000,0000
COM0_DC_L	Common Cathode Mode Disply	FB20H									0000,0000
COM1_DC_L	Common Cathode Mode Disply	FB21H									0000,0000
COM2_DC_L	Common Cathode Mode Disply	FB22H									0000,0000
COM3_DC_L	Common Cathode Mode Disply	FB23H									0000,0000
COM4_DC_L	Common Cathode Mode Disply	FB24H									0000,0000
COM5_DC_L	Common Cathode Mode Disply	FB25H									0000,0000

COM6_DC_L	Common Cathode Mode Disply	FB26H		0000,0000
COM7_DC_L	Common Cathode Mode Disply	FB27H		0000,0000
COM0_DC_H	Common Cathode Mode Disply	FB28H		0000,0000
COM1_DC_H	Common Cathode Mode Disply	FB29H		0000,0000
COM2_DC_H	Common Cathode Mode Disply	FB2AH		0000,0000
COM3_DC_H	Common Cathode Mode Disply	FB2BH		0000,0000
COM4_DC_H	Common Cathode Mode Disply	FB2CH		0000,0000
COM5_DC_H	Common Cathode Mode Disply	FB2DH		0000,0000
COM6_DC_H	Common Cathode Mode Disply	FB2EH		0000,0000
COM7_DC_H	Common Cathode Mode Disply	FB2FH		0000,0000

23.2.1 COM Enable Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
COMEN	FB00H	C7EN	C6EN	C5EN	C4EN	C3EN	C2EN	C1EN	C0EN

CnEN: COMn enable control bit (n＝0~7)

0: disable COMn, keep GPIO function
1: enable COMn, the corresponding I/O outputs the driving waveform of COM when LEDON = 1.

23.2.2 SEG Enable Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
SEGENL	FB01H	S7EN	S6EN	S5EN	S4EN	S3EN	S2EN	S1EN	S0EN
SEGENH	FB02H	S15EN	S14EN	S13EN	S12EN	S11EN	S10EN	S9EN	S8EN

SnEN: SEGn enable control bit (n＝0~7)

0: disable SEGn, keep GPIO function
1: enable SEGn, the corresponding I/O outputs the driving waveform of SEG when LEDON = 1.

23.2.3 LED Control Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
LEDCTRL	FB03H	LEDON	-	LEDMODE[1:0]		-	LEDDUTY[2:0]		

LEDON: LED driver enable control bit

0: disable LED driver
1: enable LED driver.

LEDMODE[1:0]: LED drive mode

LEDMODE[1:0]	Drive mode
00	Common cathod mode
01	Common anode mode
10	Common cathod / common anode mode
11	Reserved

LEDDUTY[2:0]: LED grayscale adjustment

LEDDUTY[2:0]	LED Duty/cycle	LED brightness
000	8/8	100%
001	7/8	87.5%
010	6/8	75%
011	5/8	62.5%
100	4/8	50%

101	3/8	37.5%
110	2/8	25%
111	1/8	12.5%

23.2.4 LED Clock Divide Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
LEDCKS	FB04H								

LEDCKS: LED clock division control

$$\text{LED working frequency} = \frac{\text{SYSclk}}{160 * \text{LEDCKS}[7:0]}$$

If the value of the register LEDCKS is set too large, it will cause the LED to flicker. Generally, if the LED refresh frequency is greater than or equal to 75Hz, there will be no obvious flicker.

$$\frac{\text{SYSclk}}{160 * 8 * \text{Ncom} * \text{LEDCKS}[7:0]} \geqslant 75\text{HZ}$$

NCOM: The number of COMs enabled, if it is a common cathode/common anode mode, it is twice the number of COMs

For example: if the operating frequency of the single-chip microcomputer is 11.0592MHz, the operating mode of the LED is common cathode/common anode mode, COMEN is set to 0FFH, even if 8 COMs are available, 11059200/160/8/16 / LEDCKS ⩾ 75, LEDCKS ⩽ 7.2, so LEDCKS is recommended to be set to 7.

23.2.5 LED data registers of common anode mode

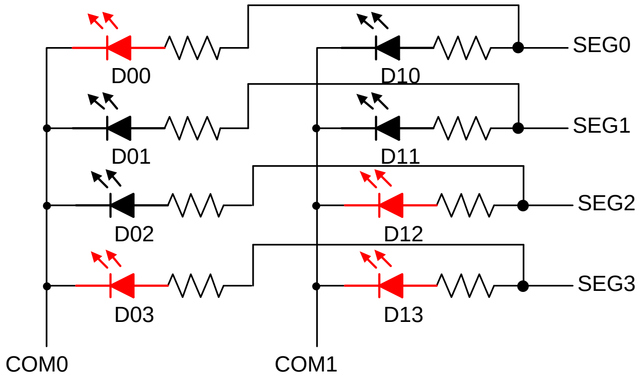
Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
COM0_DA_L	FB10H								
COM1_DA_L	FB11H								
COM2_DA_L	FB12H								
COM3_DA_L	FB13H								
COM4_DA_L	FB14H								
COM5_DA_L	FB15H								
COM6_DA_L	FB16H								
COM7_DA_L	FB17H								
COM0_DA_H	FB18H								
COM1_DA_H	FB19H								
COM2_DA_H	FB1AH								
COM3_DA_H	FB1BH								
COM4_DA_H	FB1CH								
COM5_DA_H	FB1DH								
COM6_DA_H	FB1EH								
COM7_DA_H	FB1FH								

23.2.6 LED data registers of common cathod mode

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0

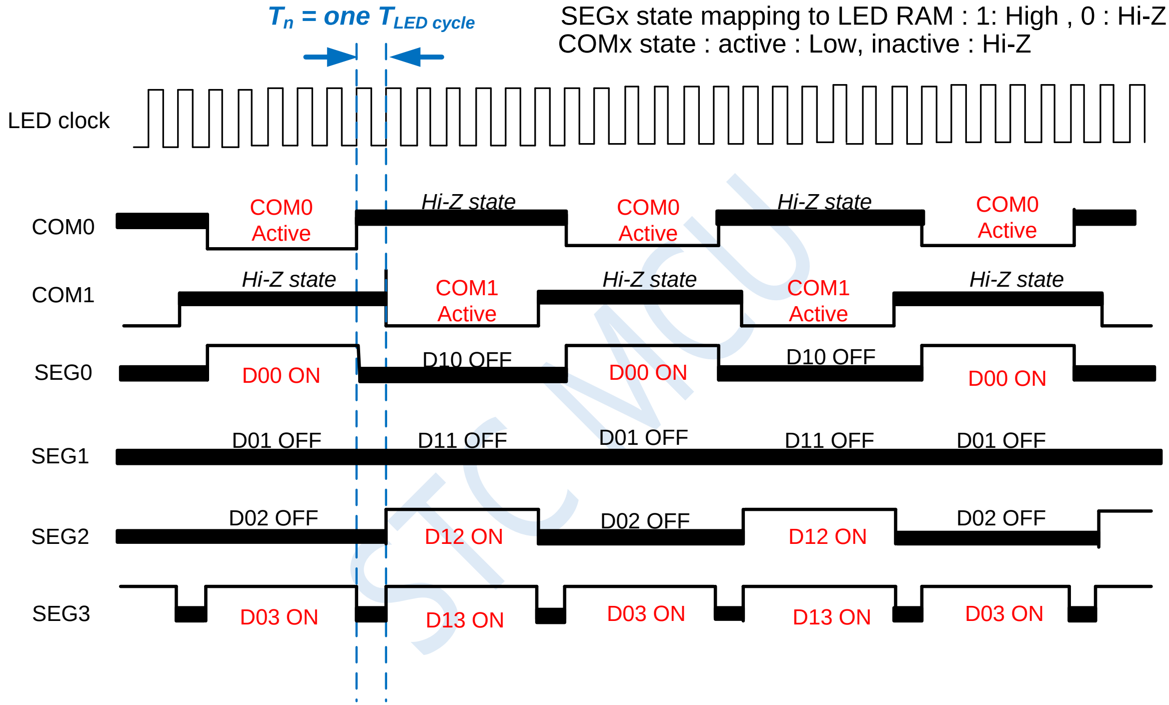
COM0_DC_L	FB20H	
COM1_DC_L	FB21H	
COM2_DC_L	FB22H	
COM3_DC_L	FB23H	
COM4_DC_L	FB24H	
COM5_DC_L	FB25H	
COM6_DC_L	FB26H	
COM7_DC_L	FB27H	
COM0_DC_H	FB28H	
COM1_DC_H	FB29H	
COM2_DC_H	FB2AH	
COM3_DC_H	FB2BH	
COM4_DC_H	FB2CH	
COM5_DC_H	FB2DH	
COM6_DC_H	FB2EH	
COM7_DC_H	FB2FH	

23.3 LED Common Cathod Mode (LEDMODE = 00)



LED RAM 1 : ON 0: OFF

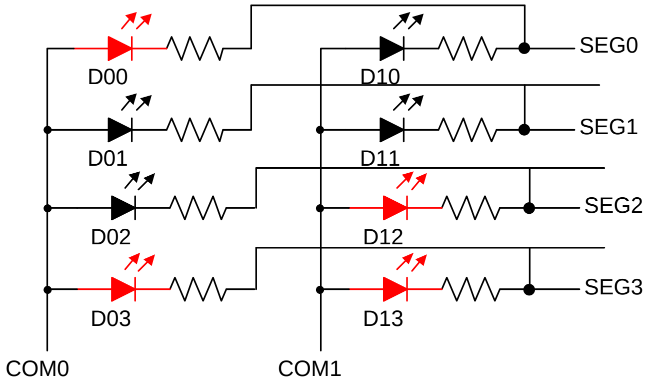
	D00	D01	D02	D03
COM0	1	0	0	1
	D10	D11	D12	D13
COM1	0	0	1	1



SEGx state mapping to LED RAM : 1: High , 0 : Hi-Z
COMx state : active : Low, inactive : Hi-Z

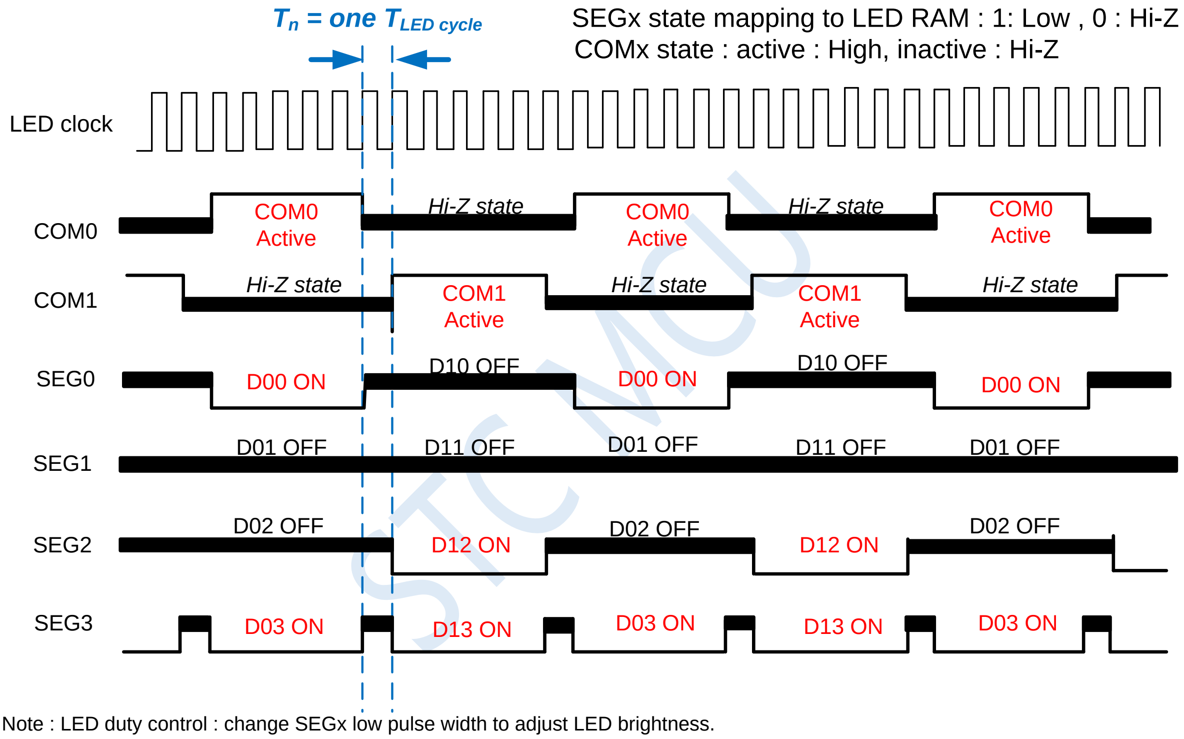
Note : LED duty control : change SEGx low pulse width to adjust LED brightness.

23.4 LED Common Anode Mode (LEDMODE = 01)



LED RAM 1 : ON 0: OFF

	D00	D01	D02	D03
COM0	1	0	0	1
	D10	D11	D12	D13
COM1	0	0	1	1



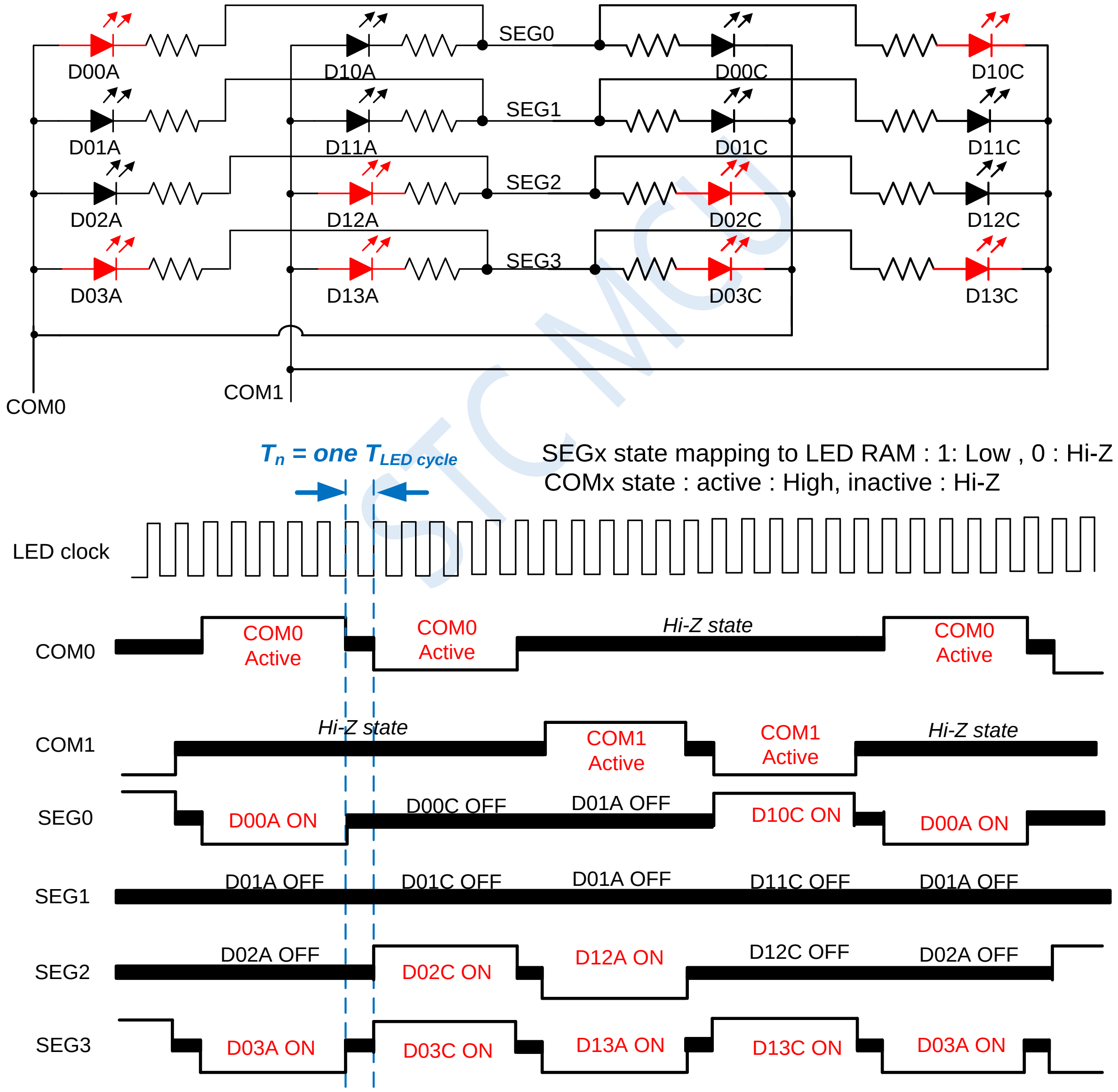
SEGx state mapping to LED RAM : 1: Low , 0 : Hi-Z
COMx state : active : High, inactive : Hi-Z

Note : LED duty control : change SEGx low pulse width to adjust LED brightness.

23.5 LED Common Cathode/ Common Anode Mode (LEDMODE = 10)

LED RAM 1 : ON 0: OFF

	D00A	D01A	D02A	D03A
COM0	1	0	0	1
	D00C	D01C	D02C	D03C
COM0	0	0	1	1
	D10A	D11A	D12A	D13A
COM1	0	0	1	1
	D10C	D11C	D12C	D13C
COM1	1	0	0	1



SEGx state mapping to LED RAM : 1: Low , 0 : Hi-Z
COMx state : active : High, inactive : Hi-Z

Note : LED duty control : change SEGx low pulse width to adjust LED brightness.

23.6 Touch Keys and LED Driver Share I/O

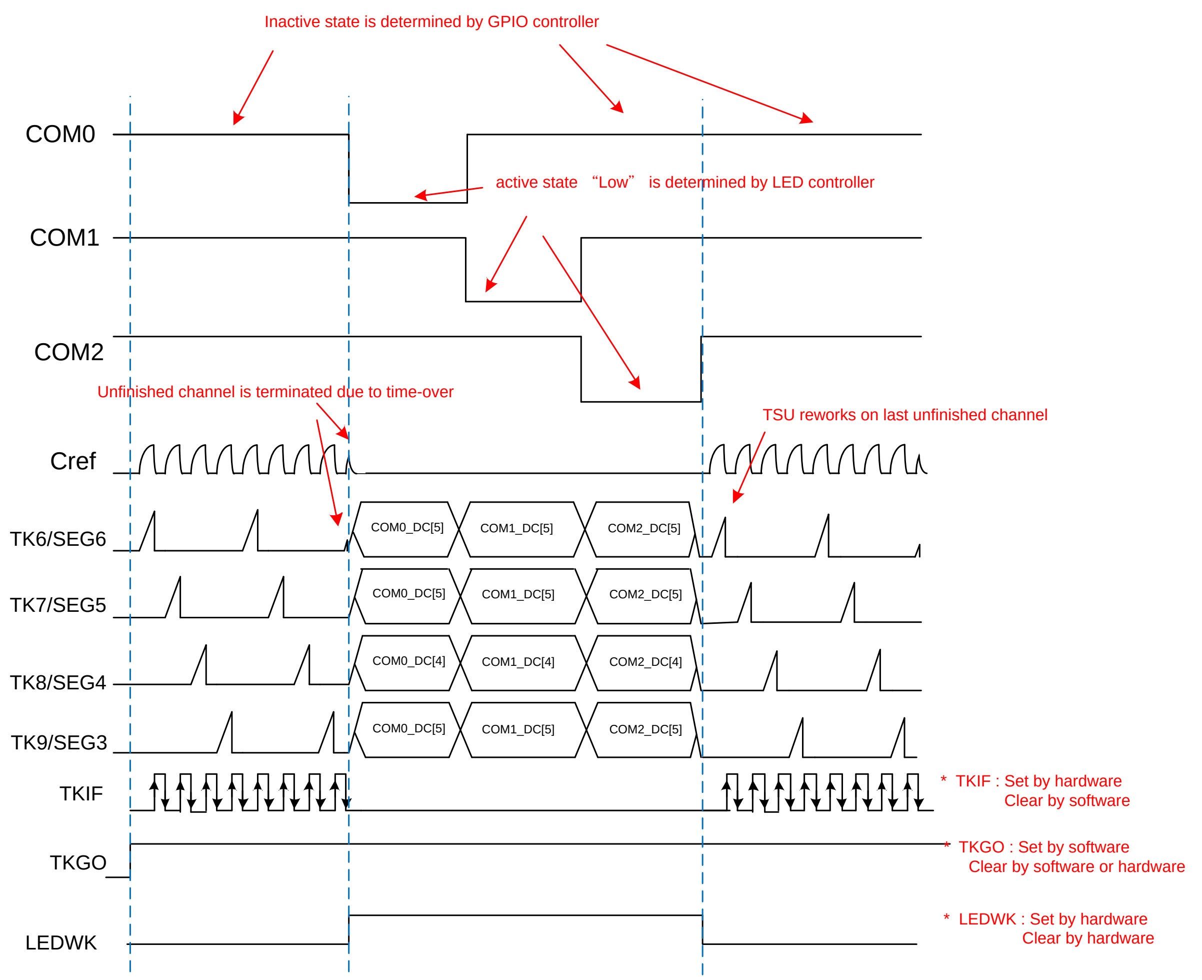


Fig. Touch/LED share timing diagram

(Common Cathode, only COM 0~2 and TK6~TK9 are enabled)

Steps:

1. Select the touch key channel to be scanned. The registers are TSCHEN1 and TSCHEN2.
2. Configure switching frequency SCR [2: 0], discharge time DT [2: 0] and select internal comparator reference voltage TSVR [1: 0].
3. Configure TSSAMP [1: 0] to determine the number of repeated scans of the same channel. If the CPU task is heavy, configure TSWAIT to use the TSIF state to delay the next channel scan.
4. If necessary, configure TSDCEN to enable the internal digital comparison function.
5. Set the TSRT content. If the TSRT content is not 0x00, the LED driver time sharing multitasking function is not enabled.
6. Configure the SEGEN and COMEN registers.
7. Configure LEDCKS to determine the time length of each COM action. This needs to be considered with the TKRT register to calculate the LED refresh rate.
8. Configure LEDMODE [1: 0] and LEDDUTY according to the operating mode and brightness of the LED required.
9. Write data to the data registers COMx_DC and COM_DA of the LED.
10. Set TSGO=1, the touch key controller starts scanning.
    a) You can read TSWKCHN [3: 0] using software to know which channel is currently being scanned. After a channel is scanned, the hardware will set TSIF to 1 and the completed channel number will

be written into TSDNCHN [3: 0]. If an overflow occurs, TSOV will also be set to 1. Software should read these registers to decide what to do next. TSIF and TSOV can only be set by hardware and cleared by software.

b) When switching to the LED working time, the LEDWK bit is read as 1, which is used to determine whether the touch key controller is working or the LED driver is working.

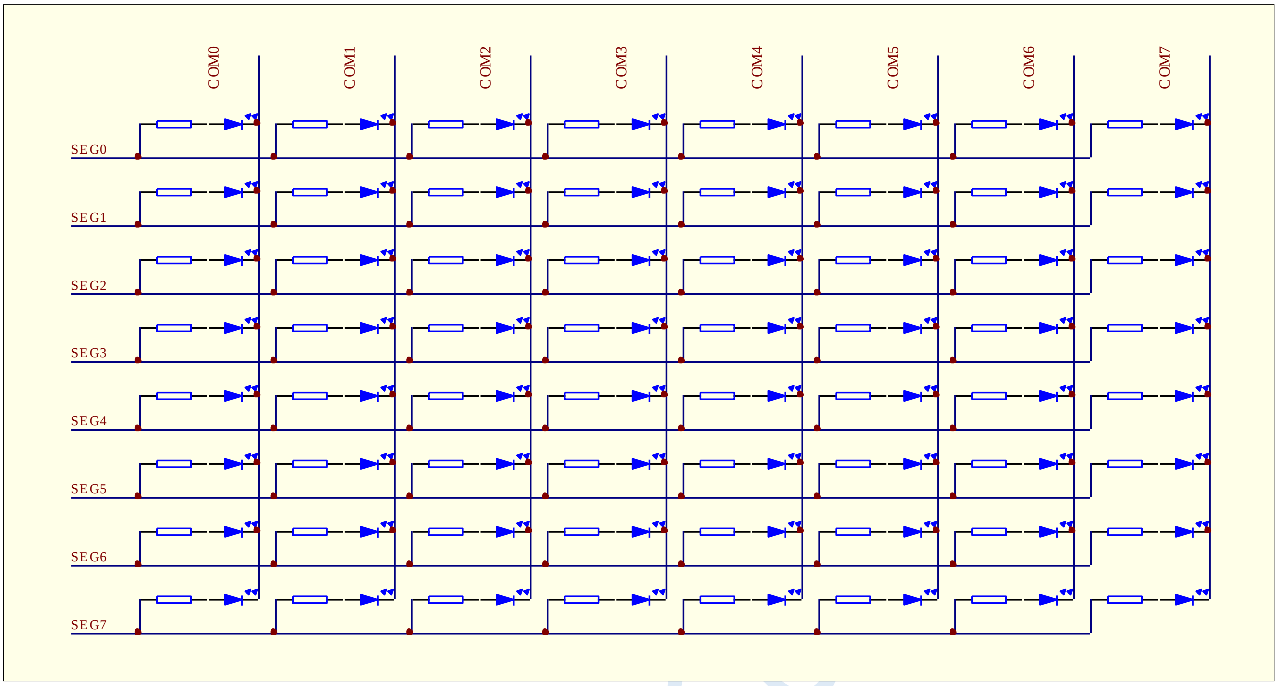
c) Software continuously updates LED data register according to actual needs.

11. If you want to terminate the touch key and LED time-sharing multi-tasking mode, you need to write TSGO = 0, then the multi-task mode is terminated, the touch key and LED are not working. And the control right of the I/O port returns to the GPIO controller.

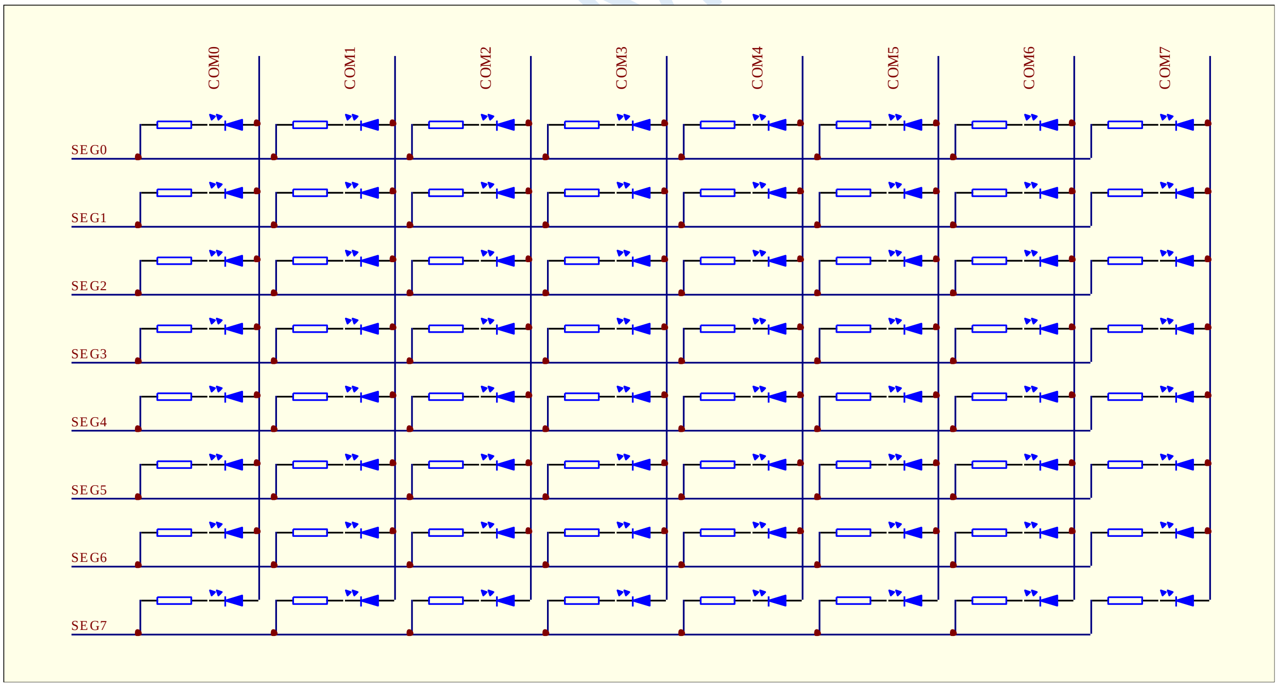
Note:

1. The SINGLE control bit is invalid in touch key and LED time-sharing multi-tasking mode. Only software writing TSGO can control the module on and off.
2. When the touch key scan time is terminated and turned to LED action time, the last key is almost incomplete. At this time, the hardware processing will not produce TSIF, and not update the registers related to the touch data, but the hardware will remember this channel number. The incomplete channel will be re-scanned and start a new round of scanning after the LED action period ends and turn to the touch key scanning time.
3. The circuit diagram of LED and touch multiplexing is as above. It should be noted that the LED's light-emitting color is different, and the equivalent capacitance of the LED will be different. The larger the capacitance, the more unfavorable the touch button, and the zero sensitivity will decrease. Generally speaking, the capacitance value of the red LED may be 35pF, but the yellow light will be as high as 100pF. At this time, if you want to increase the sensitivity of the touch button, you can connect a 1N4148 diode in series. The capacitance of 1N4148 is only 4pF, and 1N4148 string a 100pF yellow LED, the parasitic capacitance of the key will be slightly smaller than 4pF.

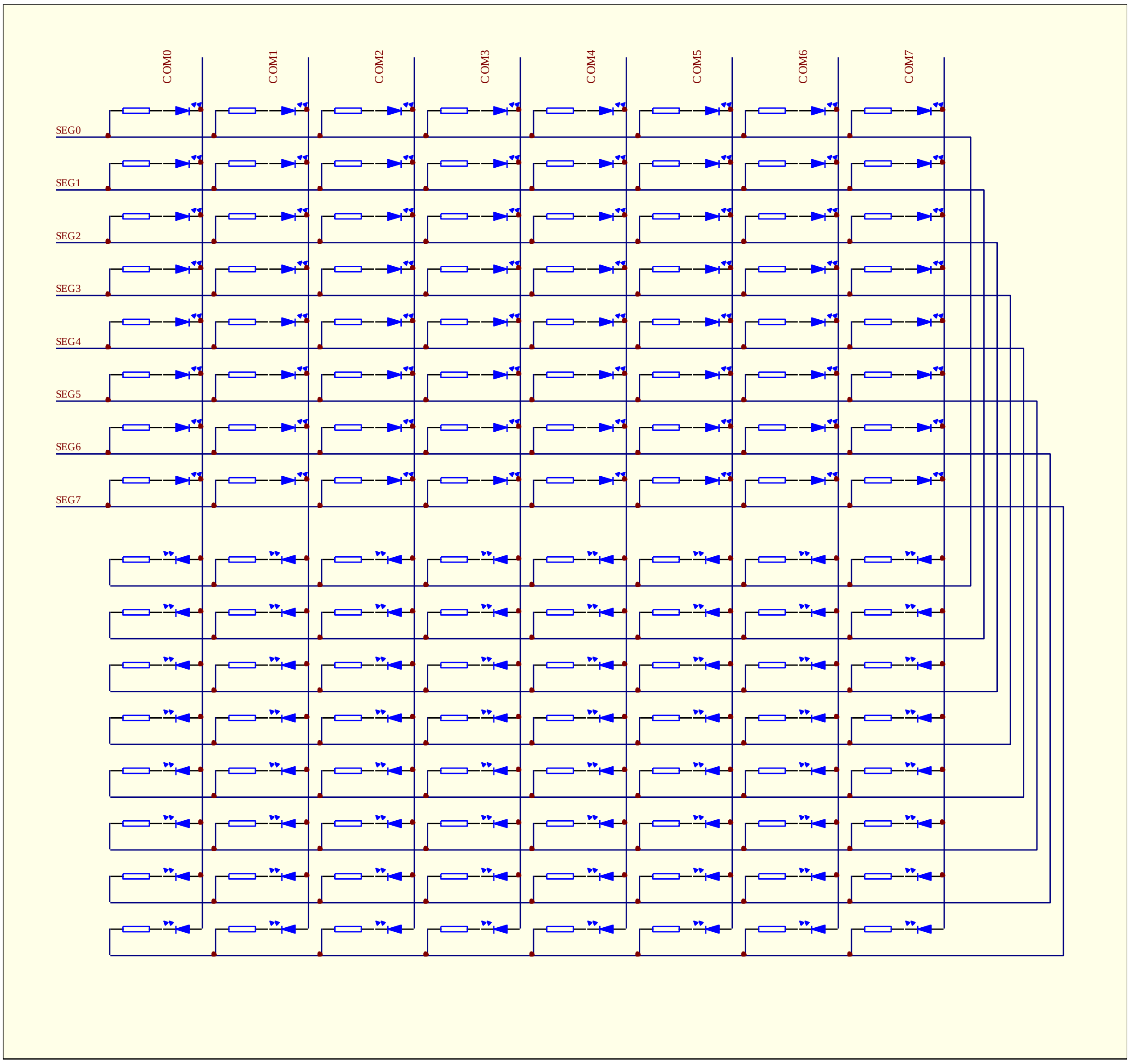
23.7 Reference Circuit of Common Cathode Mode



23.8 Reference Circuit of Common Anode Mode

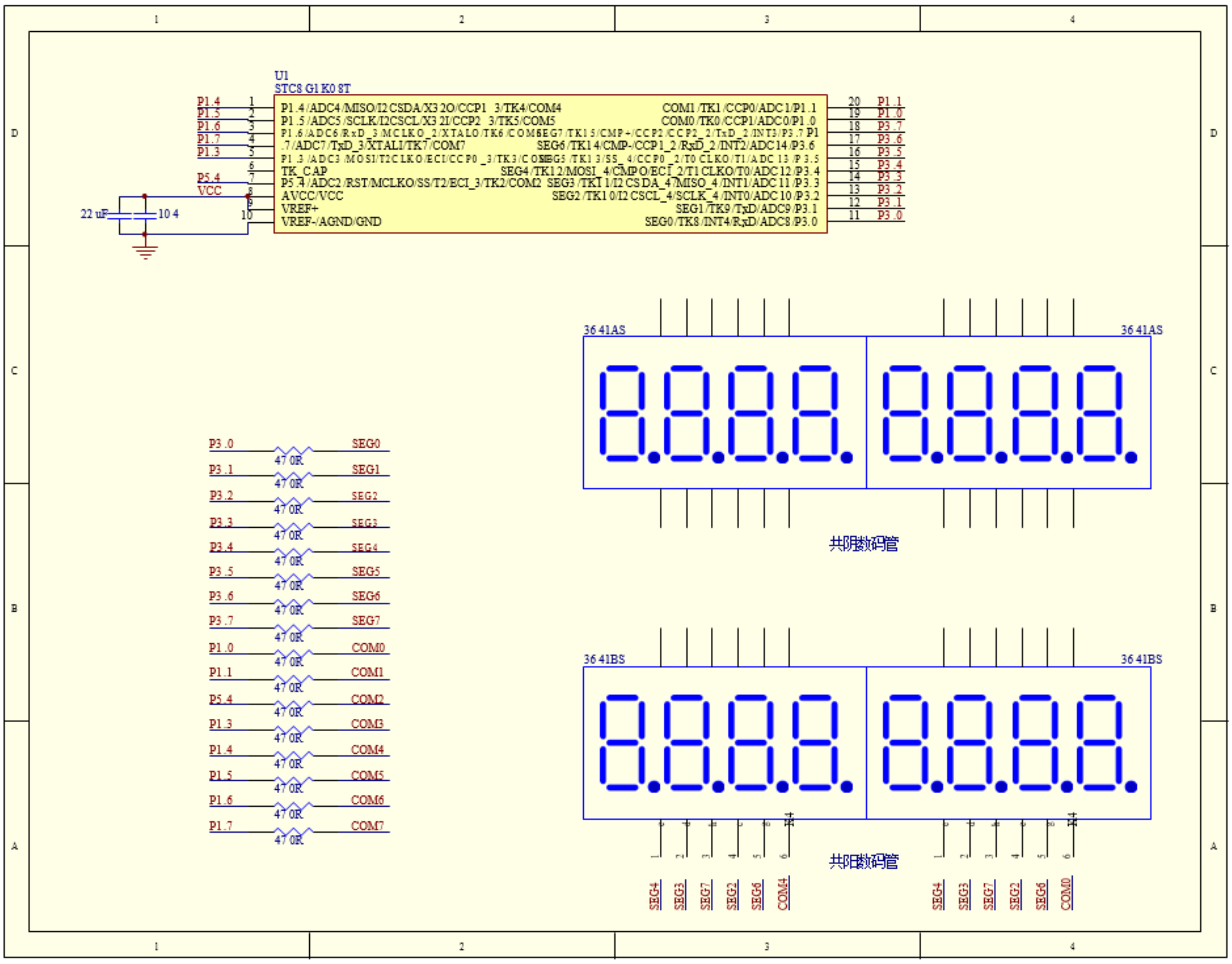


23.9 Reference Circuit of Common Cathode/Common Anode Mode



23.10 Example Routines

23.10.1 Common cathode/common anode mode drives 16 7-segment digital tubes


U1
STC8 G1 K0 8T
P1.4/ADC4/MISO/I2CSDA/X32O/CCP1_3/TK4/COM4
P1.5/ADC5/SCLK/I2CSCL/X32I/CCP2_3/TK5/COM5
P1.6/ADC6/RxD_3/MCLKO_2/XTALO/TK6/COM6
P1.7/ADC7/TxD_3/XTALI/TK7/COM7
P1.3/ADC3/MOSI/T2CLKO/ECI/CCP0_3/TK3/COM3
TK_CAP
P5.4/ADC2/RST/MCLKO/SS/T2/ECI_3/TK2/COM2
AVCC/VCC
VREF+
VREF-/AGND/GND
COM1/TK1/CCP0/ADC1/P1.1
COM0/TK0/CCP1/ADC0/P1.0
SEG7/TK15/CMP+/CCP2/CCP2_2/TxD_2/INT3/P3.7
SEG6/TK14/CMP-/CCP1_2/RxD_2/INT2/ADC14/P3.6
SEG5/TK13/SS_4/CCP0_2/T0CLKO/T1/ADC13/P3.5
SEG4/TK12/MOSI_4/CMPO/ECI_2/T1CLKO/T0/ADC12/P3.4
SEG3/TK11/I2CSDA_4/MISO_4/INT1/ADC11/P3.3
SEG2/TK10/I2CSCL_4/SCLK_4/INT0/ADC10/P3.2
SEG1/TK9/TxD/ADC9/P3.1
SEG0/TK8/INT4/RxD/ADC8/P3.0
P1.4
P1.5
P1.6
P1.7
P1.3
P5.4
VCC
22 uF
104
P1.1
P1.0
P3.7
P3.6
P3.5
P3.4
P3.3
P3.2
P3.1
P3.0
3641AS
3641AS
共阴数码管
3641BS
3641BS
共阳数码管
P3.0 470R SEG0
P3.1 470R SEG1
P3.2 470R SEG2
P3.3 470R SEG3
P3.4 470R SEG4
P3.5 470R SEG5
P3.6 470R SEG6
P3.7 470R SEG7
P1.0 470R COM0
P1.1 470R COM1
P5.4 470R COM2
P1.3 470R COM3
P1.4 470R COM4
P1.5 470R COM5
P1.6 470R COM6
P1.7 470R COM7
SEG4
SEG3
SEG7
SEG2
SEG6
COM4
SEG4
SEG3
SEG7
SEG2
SEG6
COM0

C language code

```
//Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         P0M1        =     0x93;
sfr         P0M0        =     0x94;
sfr         P1M1        =     0x91;
sfr         P1M0        =     0x92;
sfr         P2M1        =     0x95;
sfr         P2M0        =     0x96;
sfr         P3M1        =     0xb1;
sfr         P3M0        =     0xb2;
sfr         P4M1        =     0xb3;
sfr         P4M0            =       0xb4;
sfr         P5M1            =       0xc9;
sfr         P5M0            =       0xca;

#define COMEN               (*(unsigned char volatile xdata *)0xfb00)
#define SEGENL              (*(unsigned char volatile xdata *)0xfb01)
#define LEDCTR              (*(unsigned char volatile xdata *)0xfb03)
#define LEDCKS              (*(unsigned char volatile xdata *)0xfb04)
#define COM0_DA             (*(unsigned char volatile xdata *)0xfb10)
#define COM1_DA             (*(unsigned char volatile xdata *)0xfb11)
#define COM2_DA             (*(unsigned char volatile xdata *)0xfb12)
#define COM3_DA             (*(unsigned char volatile xdata *)0xfb13)
#define COM4_DA             (*(unsigned char volatile xdata *)0xfb14)
#define COM5_DA             (*(unsigned char volatile xdata *)0xfb15)
#define COM6_DA             (*(unsigned char volatile xdata *)0xfb16)
#define COM7_DA             (*(unsigned char volatile xdata *)0xfb17)
#define COM0_DC             (*(unsigned char volatile xdata *)0xfb20)
#define COM1_DC             (*(unsigned char volatile xdata *)0xfb21)
#define COM2_DC             (*(unsigned char volatile xdata *)0xfb22)
#define COM3_DC             (*(unsigned char volatile xdata *)0xfb23)
#define COM4_DC             (*(unsigned char volatile xdata *)0xfb24)
#define COM5_DC             (*(unsigned char volatile xdata *)0xfb25)
#define COM6_DC             (*(unsigned char volatile xdata *)0xfb26)
#define COM7_DC             (*(unsigned char volatile xdata *)0xfb27)

char code PATTERN[16] =
{
        0x3f,           //0
        0x06,           //1
        0x5b,           //2
        0x4f,           //3
        0x66,           //4
        0x6d,           //5
        0x7d,           //6
        0x27,           //7
        0x7f,           //8
        0x6f,           //9
        0x77,           //A
        0x7c,           //b
        0x39,           //C
        0x5E,           //d
        0x79,           //E
        0x71,           //F
};

void main()
{
        P1M0 = 0xff;
        P1M1 = 0x00;
        P3M0 = 0xff;
        P3M1 = 0x00;
        P5M0 = 0x10;
        P5M1 = 0x00;

        P_SW2 = 0x80;

        COMEN = 0xff;                                   //enable COM0~COM7
        SEGENL = 0xff;                                  //enable SEG0~SEG7
```

```
    LEDCTRL = 0x20;                                        // Common cathode/common anode mode for LED driver
    LEDCKS = 7;                                            // Set the LED refresh rate

    COM0_DA = PATTERN[0];                                  // Set the LED display content
    COM1_DA = PATTERN[1];
    COM2_DA = PATTERN[2];
    COM3_DA = PATTERN[3];
    COM4_DA = PATTERN[4];
    COM5_DA = PATTERN[5];
    COM6_DA = PATTERN[6];
    COM7_DA = PATTERN[7];

    COM0_DC = PATTERN[8];
    COM1_DC = PATTERN[9];
    COM2_DC = PATTERN[10];
    COM3_DC = PATTERN[11];
    COM4_DC = PATTERN[12];
    COM5_DC = PATTERN[13];
    COM6_DC = PATTERN[14];
    COM7_DC = PATTERN[15];

    LEDCTRL |= 0x80;                                       // Start LED driver

    P_SW2 &= ~0x80;

    while (1);
}
```

24 RTC real time clock

Product line	RTC
STC8H1K08 family	
STC8H1K28 family	
STC8H3K64S4 family	
STC8H3K64S2 family	
STC8H8K64U family A version	
STC8H8K64U family B version	●
STC8H2K64T family	●
STC8H4K64TLR family	●
STC8H4K64TLCD family	●
STC8H4K64LCD family	●

The STC8H2K64T series single-chip microcomputer integrates a real-time clock control circuit, which mainly has the following characteristics:

- Low power consumption: RTC module working current is as low as 10uA
- Long time span: support from 2000 to 2099, and automatically judge leap years
- Alarm clock: support a group of alarm clock settings
- Support multiple interrupts: alarm interrupt, day interrupt, hour interrupt, minute interrupt, second interrupt, 1/2 second interrupt, 1/8 second interrupt, 1/32 second interrupt
- Support power-down wake-up

24.1 RTC related registers

Symbol	Description	Address	Bit Address and Symbol								Reset Value
			B7	B6	B5	B4	B3	B2	B1	B0	
RTCCR	RTC Control register	FE60H	-	-	-	-	-	-	-	RUNRTC	0000,0000
RTCCFG	RTC Configuration register	FE61H	-	-	-	-	-	-	RTCCKS	SETRTC	0000,0000
RTCIEN	RTC Interrupt enable register	FE62H	EALAI	EDAYI	EHOURI	EMINI	ESECI	ESEC2I	ESEC8I	ESEC32I	0000,0000
RTCIF	RTC Interrupt request register	FE63H	ALAIF	DAYIF	HOURIF	MINIF	SECIF	SEC2IF	SEC8IF	SEC32IF	0000,0000
ALAHOUR	RTC alarm hour value	FE64H	-	-	-						0000,0000
ALAMIN	RTC alarm minute value	FE65H	-	-							0000,0000
ALASEC	RTC alarm second value	FE66H	-	-							0000,0000
ALASSEC	RTC alarm 1/128 second value	FE67H	-								0000,0000
INIYEAR	RTC year initialization	FE68H	-								0000,0000
INIMONTH	RTC month initialization	FE69H	-	-	-	-					0000,0000
INIDAY	RTC day initialization	FE6AH	-	-	-						0000,0000
INIHOUR	RTC hour initialization	FE6BH	-	-	-						0000,0000
INIMIN	RTC minute initialization	FE6CH	-	-							0000,0000
INISEC	RTC second initialization	FE6DH	-	-							0000,0000
INISSEC	RTC 1/128 second	FE6EH	-								0000,0000

	initialization										
YEAR	RTC year count value	FE70H	-								0000,0000
MONTH	RTC month count value	FE71H	-	-	-	-					0000,0000
DAY	RTC day count value	FE72H	-	-	-						0000,0000
HOUR	RTC hour count value	FE73H	-	-	-						0000,0000
MIN	RTC minute count value	FE74H	-	-							0000,0000
SEC	RTC second count value	FE75H	-	-							0000,0000
SSEC	RTC 1/128 second count value	FE76H	-								0000,0000

24.1.1 RTC control register (RTCCR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
RTCCR	FE60H	-	-	-	-	-	-	-	RUNRTC

RUNRTC: RTC module control bit

0: Turn off RTC, RTC stops counting

1: Enable RTC and start RTC counting

24.1.2 RTC Configuration Register (RTCCFG)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
RTCCFG	FE61H	-	-	-	-	-	-	RTCCKS	SETRTC

RTCCKS: RTC clock source selection

0: select external 32.768KHz clock source (The software needs to start the external 32K crystal oscillator firstly)

1: Select internal 32K clock source (The software needs to start the internal 32K crystal oscillator firstly)

SETRTC: Set the initial value of RTC

0: meaningless

1: Trigger RTC register initialization. When SETRTC is set to 1, the hardware will automatically copy the values in the registers INIYEAR, INIMONTH, INIDAY, INIHOUR, INIMIN, INISEC, INISSEC to the registers YEAR, MONTH, DAY, HOUR, MIN, SEC, SSEC. After the initial completion, the hardware will automatically clear the SETRTC bit to 0.

24.1.3 RTC Interrupt Enable Register (RTCIEN)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
RTCIEN	FE62H	EALAI	EDAYI	EHOURI	EMINI	ESECI	ESEC2I	ESEC8I	ESEC32I

EALAI: Alarm interrupt enable bit

0: Turn off the alarm interrupt

1: Enable alarm interrupt

EDAYI: One day (24 hours) interrupt enable bit

0: Turn off one-day interrupt

1: Enable one-day interrupt

EHOURI: One hour (60 minutes) interrupt enable bit

0: Turn off hourly interrupt

1: Enable hourly interrupt

EMINI: One minute (60 seconds) interrupt enable bit

0: Turn off hourly interrupt

1: Enable hourly interrupt

ESECI: One second interrupt enable bit
0: Turn off the second interrupt
1: Enable second interrupt

ESEC2I: 1/2 second interrupt enable bit
0: Turn off the 1/2 second interrupt
1: Enable 1/2 second interrupt

ESEC8I: 1/8 second interrupt enable bit
0: Turn off 1/8 second interrupt
1: Enable 1/8 second interrupt

ESEC32I: 1/32 second interrupt enable bit
0: Disable 1/32 second interrupt
1: Enable 1/32 second interrupt

24.1.4 RTC Interrupt Request Register (RTCIF)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
RTCIF	FE63H	ALAIF	DAYIF	HOURIF	MINIF	SECIF	SEC2IF	SEC8IF	SEC32IF

ALAIF: Alarm interrupt request bit. Need software to clear, software write 1 is invalid.
DAYIF: One day (24 hours) interrupt request bit. Need software to clear, software write 1 is invalid.
HOURIF: One hour (60 minutes) interrupt request bit. Need software to clear, software write 1 is invalid.
MINIF: One minute (60 seconds) interrupt request bit. Need software to clear, software write 1 is invalid.
SECIF: One-second interrupt request bit. Need software to clear, software write 1 is invalid.
SEC2IF: 1/2 second interrupt request bit. Need software to clear, software write 1 is invalid.
SEC8IF: 1/8 second interrupt request bit. Need software to clear, software write 1 is invalid.
SEC32IF: 1/32 second interrupt request bit. Need software to clear, software write 1 is invalid.

24.1.5 RTC Alarm Setting Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
ALAHOUR	FE64H	-	-	-					
ALAMIN	FE65H	-	-						
ALASEC	FE66H	-	-						
ALASSEC	FE67H	-							

ALAHOUR: Set the hour value of the daily alarm.

Note: The value set is not BCD code, but HEX code. For example, if you need to set the hour value from 20 to ALAHOUR, you need to use the following code to set

```
MOV     DPTR,#ALAHOUR
MOV     A,#14H
MOVX    @@DPTR,A
```

ALAMIN: Set the minute value of the daily alarm. The digital code is the same as ALAHOUR.
ALASEC: Set the second value of the daily alarm. The digital code is the same as ALAHOUR.
ALASSEC: Set the 1/128 second value of the daily alarm. The digital code is the same as ALAHOUR.

24.1.6 RTC real-time clock initial value setting registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
INIYEAR	FE68H	-							
INIMONTH	FE69H								

INIDAY	FE6AH								
INIHOUR	FE6BH	-	-	-					
INIMIN	FE6CH	-	-						
INISEC	FE6DH	-	-						
INISSEC	FE6EH	-							

INIYEAR: Set the year value of the current real-time time. The valid range is 00~99. Corresponding to 2000～2099

Note: The set value is not BCD code, but HEX code. For example, if you need to set 20 to INIYEAR, you need to use the following code to set

```
MOV     DPTR,#INIYEAR
MOV     A,#14H
MOVX    @@DPTR,A
```

INIMONTH: Set the monthly value of the current real-time time. The valid range is 1-12. The number code is the same as INIYEAR.

INIDAY: Set the daily value of the current real-time time. The valid range is 1~31. The number code is the same as INIYEAR.

INIHOUR: Set the hour value of the current real-time time. The valid range is 00~23. The number code is the same as INIYEAR.

INIMIN: Set the minute value of the current real-time time. The valid range is 00~59. The number code is the same as INIYEAR.

INISEC: Set the second value of the current real-time time. The valid range is 00~59. The number code is the same as INIYEAR.

INISSEC: Set 1/128 second value of current real time. The valid range is 00~127. The number code is the same as INIYEAR.

After the user has set the above initial value register, the user also needs to write 1 to the SETRTC bit (RTCCFG.0) to trigger the hardware to load the initial value into the RTC real-time counter

Also note: the hardware will not check the validity of the initialized data. When setting the initial value, the user must ensure the validity of the data and cannot exceed its valid range.

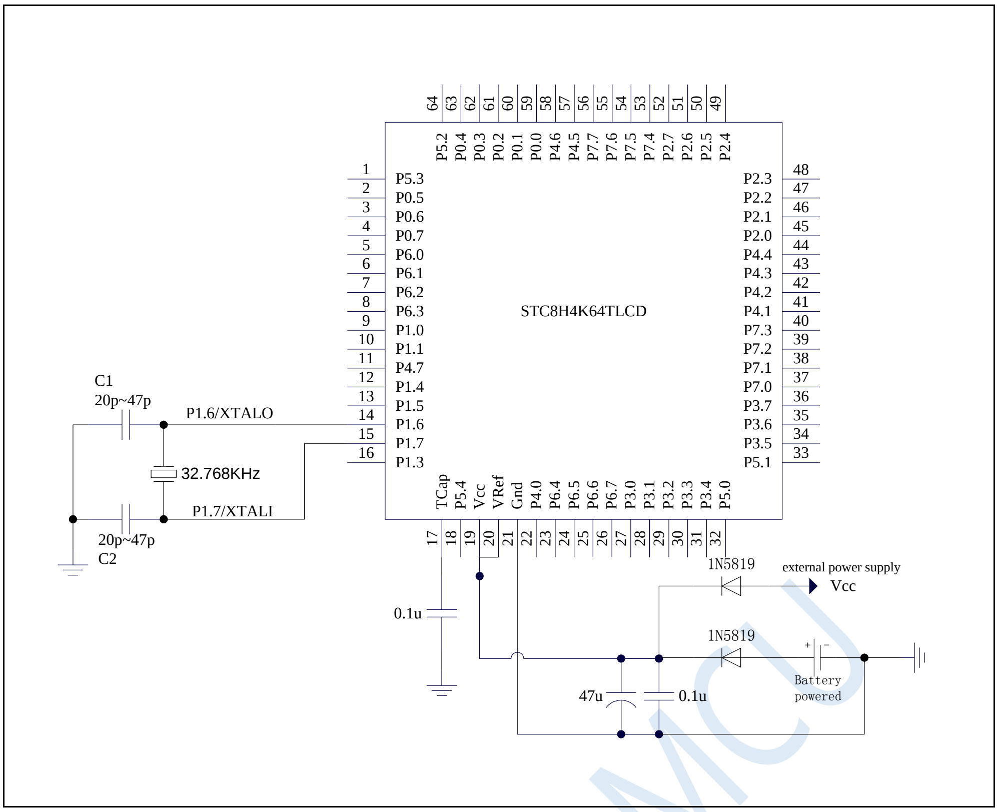
24.1.7 RTC real-time clock count registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
YEAR	FE70H	-							
MONTH	FE71H								
DAY	FE72H								
HOUR	FE73H	-	-	-					
MIN	FE74H	-	-						
SEC	FE75H	-	-						
SSEC	FE76H	-							

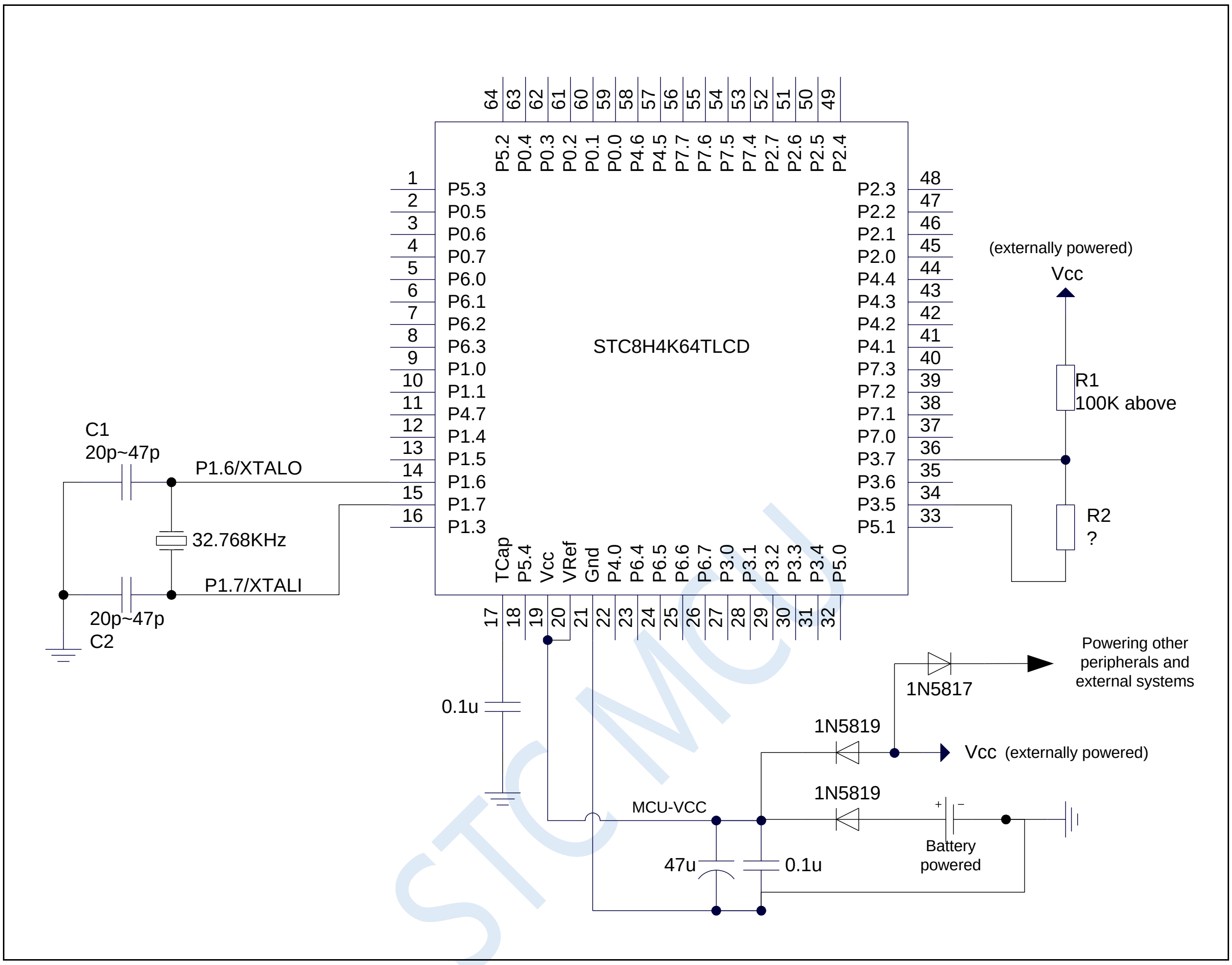
YEAR: The year value of the current real time. Note: The value of the register is not BCD code, but HEX code
MONTH: The monthly value of the current real-time time. The number code is the same as YEAR.
DAY: The daily value of the current real time. The number code is the same as YEAR.
HOUR: The hour value of the current real time. The number code is the same as YEAR.
MIN: The minute value of the current real time. The number code is the same as YEAR.
SEC: The second value of the current real time. The number code is the same as YEAR.
SSEC: 1/128 second value of current real time. The number code is the same as YEAR.

Note: YEAR, MONTH, DAY, HOUR, MIN, SEC, and SSEC are all read-only registers. If you need to write to these registers, you must use the registers INIYEAR, INIMONTlH, INIDAT, INIHOU, INIMIN, INISEC, INISSEC and SETRTC are implemented.

24.2 RTC reference circuit diagram (without VBAT pin)



24.3 RTC practical circuit diagram



24.4 Example Routines

24.4.1 Serial port printing RTC clock example

C language code

```
//Operating frequency for test is 22.1184MHz. It is necessary to load the C language code file and the following assembly code file into the same project for use

#include "reg51.h"
#include "intrins.h"
#include "stdio.h"

sfr         TH2             =   0xD6;
sfr         TL2             =   0xD7;
sfr         AUXR            =   0x8E;
sfr         P_SW2           =   0xBA;

sfr         P1M1            =   0x91;
sfr         P1M0            =   0x92;
sfr         P0M1            =   0x93;
sfr         P0M0            =   0x94;
sfr         P2M1            =   0x95;
sfr         P2M0            =   0x96;
sfr         P3M1            =   0xb1;
sfr         P3M0            =   0xb2;
sfr         P4M1            =   0xb3;
sfr         P4M0            =   0xb4;
sfr         P5M1            =   0xc9;
sfr         P5M0            =   0xca;

#define     RTCCR               (*(unsigned char volatile xdata *)0xfe60)
#define     RTCCFG              (*(unsigned char volatile xdata *)0xfe61)
#define     RTCIEN              (*(unsigned char volatile xdata *)0xfe62)
#define     RTCIF               (*(unsigned char volatile xdata *)0xfe63)
#define     ALAHOUR             (*(unsigned char volatile xdata *)0xfe64)
#define     ALAMIN              (*(unsigned char volatile xdata *)0xfe65)
#define     ALASEC              (*(unsigned char volatile xdata *)0xfe66)
#define     ALASSEC             (*(unsigned char volatile xdata *)0xfe67)
#define     INIYEAR             (*(unsigned char volatile xdata *)0xfe68)
#define     INIMONTH            (*(unsigned char volatile xdata *)0xfe69)
#define     INIDAY              (*(unsigned char volatile xdata *)0xfe6a)
#define     INIHOUR             (*(unsigned char volatile xdata *)0xfe6b)
#define     INIMIN              (*(unsigned char volatile xdata *)0xfe6c)
#define     INISEC              (*(unsigned char volatile xdata *)0xfe6d)
#define     INISSEC             (*(unsigned char volatile xdata *)0xfe6e)
#define     YEAR                (*(unsigned char volatile xdata *)0xfe70)
#define     MONTH               (*(unsigned char volatile xdata *)0xfe71)
#define     DAY                 (*(unsigned char volatile xdata *)0xfe72)
#define     HOUR                (*(unsigned char volatile xdata *)0xfe73)
#define     MIN                 (*(unsigned char volatile xdata *)0xfe74)
#define     SEC                 (*(unsigned char volatile xdata *)0xfe75)
#define     SSEC                (*(unsigned char volatile xdata *)0xfe76)

#define     MAIN_Fosc           22118400L
#define     Baudrate            115200L
```

```
#define     TM                      (65536 -(MAIN_Fosc/Baudrate/4))

bit         B1S_Flag;

void RTC_config(void);

void UartInit(void)
{
    SCON = (SCON & 0x3f) | 0x40;
    TL2  = TM;
    TH2  = TM>>8;
    AUXR |= 0x15;
}

void UartPutc(unsigned char dat)
{
    SBUF = dat;
    while(TI==0);
    TI = 0;
}

char putchar(char c)
{
    UartPutc(c);
    return c;
}

void RTC_Isr() interrupt 13
{
    char store;

    store = P_SW2;
    P_SW2 |= 0x80;          // Enable XFR access

    if(RTCIF & 0x08)            // Determine whether it is the second interrupt
    {
        RTCIF &= ~0x08;                             // clear interrupt flag
        B1S_Flag = 1;
    }

    P_SW2 = store;
}

void main(void)
{
    P0M1 = 0;   P0M0 = 0;   //set as quasi-bidirectional port
    P1M1 = 0;   P1M0 = 0;   //set as quasi-bidirectional port
    P2M1 = 0;   P2M0 = 0;   //set as quasi-bidirectional port
    P3M1 = 0;   P3M0 = 0;   //set as quasi-bidirectional port
    P4M1 = 0;   P4M0 = 0;   //set as quasi-bidirectional port
    P5M1 = 0;   P5M0 = 0;   //set as quasi-bidirectional port

    UartInit();
    RTC_config();
    EA = 1;
    printf("RTC Test Programme!\r\n");                  // UART sends a string

    while (1)
```

```
        {
                if(B1S_Flag)
                {
                        B1S_Flag = 0;
                        P_SW2 |= 0x80;                              // Enable XFR access

                        printf("Year=20%bd  ", YEAR);
                        printf("Month=%bd  ", MONTH);
                        printf("Day=%bd  ", DAY);
                        printf("Hour=%bd  ", HOUR);
                        printf("Minute=%bd  ", MIN);
                        printf("Second=%bd  ", SEC);
                        printf("\r\n");

                        P_SW2 &= ~0x80;                             //Disable XFR access
                }
        }
}

void RTC_config(void)
{
        P_SW2 |= 0x80;                                              //Enable XFR access

        //Select Internal 32K
        IRC32KCR = 0x80;                                            //Start the internal 32K oscillator
        while (!(IRC32KCR & 0x01));                                 //Wait for the clock to stabilize
        RTCCFG |= 0x02;                                             //Select Internal 32K as RTC clock source

//      //Select external 32K
//      X32KCR = 0xc0;                                              //Start the external 32K crystal oscillator
//      while (!(X32KCR & 0x01));                                   //Wait for the clock to stabilize
//      RTCCFG &= ~0x02;                                            //Select external 32K as RTC clock source

        INIYEAR = 21;                                               //Y:2021
        INIMONTH = 12;                                              //M:12
        INIDAY = 31;                                                //D:31
        INIHOUR = 23;                                               //H:23
        INIMIN = 59;                                                //M:59
        INISEC = 50;                                                //S:50
        INISSEC = 0;                                                //S/128:0
        RTCCFG |= 0x01;                                             //Trigger RTC register initialization

        RTCIF = 0;                                                  //clear interrupt flag
        RTCIEN = 0x08;                                              //Enable RTC second interrupt
        RTCCR = 0x01;                                               // Enable RTC
        P_SW2 &= ~0x80;                                             //Disable XFR access
}
```

Assembly code

```
; Save the following code as an ASM format file and load it into the project together, for example: isr.asm

                CSEG    AT  0123H
                JMP             006BH
                END
```

25 LCD driver (Traditional segment/stroke LCD driver)

Product line	LCD driver
STC8H1K08 family	
STC8H1K28 family	
STC8H3K64S4 family	
STC8H3K64S2 family	
STC8H8K64U family	
STC8H2K64T family	
STC8H4K64TLR family	
STC8H4K64TLCD family	●
STC8H4K64LCD family	●

Some microcontrollers of the STC8H series integrate an LCD driver, which can be used to drive the LCD. It can drive up to 4COM*40SEG dot matrix LCD.

Ports corresponding to the COM line:

COM7	COM6	COM5	COM4	COM3	COM2	COM1	COM0
-	-	-	-	P3.6	P3.5	P5.1	P5.0

Ports corresponding to SEG lines:

SEG39	SEG38	SEG37	SEG36	SEG35	SEG34	SEG33	SEG32
P6.3	P6.2	P6.1	P6.0	P7.0	P7.1	P7.2	P7.3
SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24
P3.7	P4.1	P4.2	P4.3	P4.4	P2.0	P2.1	P2.2
SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16
P2.3	P2.4	P2.5	P2.6	P2.7	P4.5	P4.6	P0.0
SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8
P0.1	P0.2	P0.3	P0.4	P5.2	P5.3	P0.5	P0.6
SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0
P0.7	P1.0	P1.1	P4.7	P1.2/P7.4	P1.3/P7.5	P1.4/P7.6	P1.5/P7.7

25.1 LCD function pins switch

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
LCDCFG2	FB81H	-	-	-	-	SEG3PS	SEG2PS	SEG1PS	SEG0PS

SEG3PS: SEG3 control pin selection bit

SEG3PS	SEG3
0	P1.2
1	P7.4

SEG2PS: SEG2 control pin selection bit

SEG2PS	SEG2
0	P1.3
1	P7.5

SEG1PS: SEG1 control pin selection bit

SEG1PS	SEG1
0	P1.4

1	P7.6

SEG0PS: SEG0 control pin selection bit

SEG0PS	SEG0
0	P1.5
1	P7.7

25.2 LCD related registers

Symbol	Description	Address	Bit Address and Symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
LCDCFG	LCD configuration register	FB80H	CKSEL	-	-	-	VRLPSEL	VLCDSEL[2:0]			0xxx,0000
LCDCFG2	LCD configuration register 2	FB81H	-	-	-	-	SEG3PS	SEG2PS	SEG1PS	SEG0PS	xxxx,0000
DBLEN	Dead time length configuration	FB82H	-	-	-	-	-	DBLEN[2:0]			xxxx,x000
COMLENL	COM time length configuration low bits	FB83H	COMLEN[7:0]								0000,0000
COMLENM	COM time length configuration medium bits	FB84H	COMLEN[15:8]								0000,0000
COMLENH	COM time length configuration high bits	FB85H	-	-	-	-	COMLEN[19:16]				xxxx,0000
BLINKRATE	Flicker Rate Configuration Register	FB86H	BLANKRATE[7:0]								1000,0000
LCDCR	LCD control register	FB87H	-	-	-	-	-	ACTMODE[1:0]		ENLCD	xxxx,x000
COMON	COM Line Enable Register	FB88H	-	-	-	-	COM3	COM2	COM1	COM0	xxxx,0000
SEGON1	SEG Line Enable Register 1	FB8AH	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0	0000,0000
SEGON2	SEG Line Enable Register 2	FB8BH	SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8	0000,0000
SEGON3	SEG Line Enable Register 3	FB8CH	SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16	0000,0000
SEGON4	SEG Line Enable Register 4	FB8DH	SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24	0000,0000
SEGON5	SEG Line Enable Register 5	FB8EH	SEG39	SEG38	SEG37	SEG36	SEG35	SEG34	SEG33	SEG32	0000,0000
C0SEGV0	C0SEG7_0 Data Register	FB90H	C0S7	C0S6	C0S5	C0S4	C0S3	C0S2	C0S1	C0S0	0000,0000
C0SEGV1	C0SEG15_8 Data Register	FB91H	C0S15	C0S14	C0S13	C0S12	C0S11	C0S10	C0S9	C0S8	0000,0000
C0SEGV2	C0SEG23_16 Data Register	FB92H	C0S23	C0S22	C0S21	C0S20	C0S19	C0S18	C0S17	C0S16	0000,0000
C0SEGV3	C0SEG31_24 Data Register	FB93H	C0S31	C0S30	C0S29	C0S28	C0S27	C0S26	C0S25	C0S24	0000,0000
C0SEGV4	C0SEG39_32 Data Register	FB94H	C0S39	C0S38	C0S37	C0S36	C0S35	C0S34	C0S33	C0S32	0000,0000
C1SEGV0	C1SEG7_0 Data Register	FB98H	C1S7	C1S6	C1S5	C1S4	C1S3	C1S2	C1S1	C1S0	0000,0000
C1SEGV1	C1SEG15_8 Data Register	FB99H	C1S15	C1S14	C1S13	C1S12	C1S11	C1S10	C1S9	C1S8	0000,0000
C1SEGV2	C1SEG23_16 Data Register	FB9AH	C1S23	C1S22	C1S21	C1S20	C1S19	C1S18	C1S17	C1S16	0000,0000
C1SEGV3	C1SEG31_24 Data Register	FB9BH	C1S31	C1S30	C1S29	C1S28	C1S27	C1S26	C1S25	C1S24	0000,0000
C1SEGV4	C1SEG39_32 Data Register	FB9CH	C1S39	C1S38	C1S37	C1S36	C1S35	C1S34	C1S33	C1S32	0000,0000
C2SEGV0	C2SEG7_0 Data Register	FBA0H	C2S7	C2S6	C2S5	C2S4	C2S3	C2S2	C2S1	C2S0	0000,0000
C2SEGV1	C2SEG15_8 Data Register	FBA1H	C2S15	C2S14	C2S13	C2S12	C2S11	C2S10	C2S9	C2S8	0000,0000
C2SEGV2	C2SEG23_16 Data Register	FBA2H	C2S23	C2S22	C2S21	C2S20	C2S19	C2S18	C2S17	C2S16	0000,0000
C2SEGV3	C2SEG31_24 Data Register	FBA3H	C2S31	C2S30	C2S29	C2S28	C2S27	C2S26	C2S25	C2S24	0000,0000
C2SEGV4	C2SEG39_32 Data Register	FBA4H	C2S39	C2S38	C2S37	C2S36	C2S35	C2S34	C2S33	C2S32	0000,0000
C3SEGV0	C3SEG7_0 Data Register	FBA8H	C3S7	C3S6	C3S5	C3S4	C3S3	C3S2	C3S1	C3S0	0000,0000
C3SEGV1	C3SEG15_8 Data Register	FBA9H	C3S15	C3S14	C3S13	C3S12	C3S11	C3S10	C3S9	C3S8	0000,0000
C3SEGV2	C3SEG23_16 Data Register	FBAAH	C3S23	C3S22	C3S21	C3S20	C3S19	C3S18	C3S17	C3S16	0000,0000
C3SEGV3	C3SEG31_24 Data Register	FBABH	C3S31	C3S30	C3S29	C3S28	C3S27	C3S26	C3S25	C3S24	0000,0000
C3SEGV4	C3SEG39_32 Data Register	FBACH	C3S39	C3S38	C3S37	C3S36	C3S35	C3S34	C3S33	C3S32	0000,0000

25.2.1 LCD configuration register (LCDCFG)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
LCDCFG	FB80H	CKSEL	-	-	-	VRLPSEL	VLCDSEL[2:0]		

CKSEL: LCD clock source selection

0: Select CPU clock as LCD clock source

1: Select external 32K crystal oscillator as LCD clock source

VRLPSEL: not used temporarily, it is recommended to set it to 0

VLCDSEL: VLCD voltage select bit

VLCDSEL[2:0]	VLCD
000	0.65*VCC
001	0.70*VCC
010	0.75*VCC
011	0.80*VCC
100	0.85*VCC
101	0.90*VCC
110	0.95*VCC
111	1.00*VCC

For example:

If the VCC is 3.3V and the LCD is 3.0V powered, VLCDSEL can be set to 101B, that is, the VLCD voltage can be set to 3.3V*0.90=2.97V;

If the VCC is 5.0V and the LCD is 3.0V powered, VLCDSEL can be set to 000B, that is, set the VLCD voltage to 5.0V*0.65=3.25V;

25.2.2 LCD configuration register 2 (LCDCFG2)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
LCDCFG2	FB81H	-	-	-	-	SEG3PS	SEG2PS	SEG1PS	SEG0PS

SEG3PS: SEG3 control pin selection

SEG3PS	SEG3
0	P1.2
1	P7.4

SEG2PS: SEG2 control pin selection

SEG2PS	SEG2
0	P1.3
1	P7.5

SEG1PS: SEG1 control pin selection

SEG1PS	SEG1
0	P1.4
1	P7.6

SEG0PS: SEG0 control pin selection

SEG0PS	SEG0
0	P1.5
1	P7.7

25.2.3 Dead Time Length Configuration Register (DBLEN)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DBLEN	FB82H	-	-	-	-	-	DBLEN[2:0]		

DBLEN[2:0]: Set the length of dead time when LCD display

25.2.4 COM Time Length Configuration Registers (COMLENx)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
COMLENL	FB83H	COMLEN[7:0]							
COMLENM	FB84H	COMLEN[15:8]							
COMLENH	FB85H	-	-	-	-	COMLEN[19:16]			

COMLEN[19:0]: Set the COM time length for LCD display

25.2.5 Flicker Rate Configuration Register (BLANKRATE)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
BLANKRATE	FB86H	BLANKRATE[7:0]							

BLANKRATE[7:0]: Set the blinking rate of LCD in blinking mode

25.2.6 LCD Control Register (LCDCR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
LCDCR	FB87H	-	-	-	-	-	ACTMODE[1:0]		ENLCD

ACTMODE[1:0]: Set LCD display mode

ACTMODE[1:0]	Mode
00	normal display mode
01	Always light mode
10	Always dark mode
11	Blink mode

ENLCD: LCD module enable control bit

0: disable LCD module

1: enable LCD module

25.2.7 COM Line Enable Register (COMON)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
COMON	FB88H	-	-	-	-	COM3	COM2	COM1	COM0

COMn: COMn enable control bit (n=0~3)

0: disable COMn(The corresponding port is normal IO)

1: enable COMn

25.2.8 SEG Line Enable Register (SEGONx)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
SEGON1	FB8AH	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0
SEGON2	FB8BH	SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8
SEGON3	FB8CH	SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16
SEGON4	FB8DH	SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24
SEGON5	FB8EH	SEG39	SEG38	SEG37	SEG36	SEG35	SEG34	SEG33	SEG32

SEGn: SEGn enable control bit (n=0~39)

0: disable SEGn (The corresponding port is normal IO)

1: enable SEGn

25.2.9 LCD Data Registers (CxSEGVx)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
C0SEGV0	FB90H	C0S7	C0S6	C0S5	C0S4	C0S3	C0S2	C0S1	C0S0
C0SEGV1	FB91H	C0S15	C0S14	C0S13	C0S12	C0S11	C0S10	C0S9	C0S8
C0SEGV2	FB92H	C0S23	C0S22	C0S21	C0S20	C0S19	C0S18	C0S17	C0S16

C0SEGV3	FB93H	C0S31	C0S30	C0S29	C0S28	C0S27	C0S26	C0S25	C0S24
C0SEGV4	FB94H	C0S39	C0S38	C0S37	C0S36	C0S35	C0S34	C0S33	C0S32
C1SEGV0	FB98H	C1S7	C1S6	C1S5	C1S4	C1S3	C1S2	C1S1	C1S0
C1SEGV1	FB99H	C1S15	C1S14	C1S13	C1S12	C1S11	C1S10	C1S9	C1S8
C1SEGV2	FB9AH	C1S23	C1S22	C1S21	C1S20	C1S19	C1S18	C1S17	C1S16
C1SEGV3	FB9BH	C1S31	C1S30	C1S29	C1S28	C1S27	C1S26	C1S25	C1S24
C1SEGV4	FB9CH	C1S39	C1S38	C1S37	C1S36	C1S35	C1S34	C1S33	C1S32
C2SEGV0	FBA0H	C2S7	C2S6	C2S5	C2S4	C2S3	C2S2	C2S1	C2S0
C2SEGV1	FBA1H	C2S15	C2S14	C2S13	C2S12	C2S11	C2S10	C2S9	C2S8
C2SEGV2	FBA2H	C2S23	C2S22	C2S21	C2S20	C2S19	C2S18	C2S17	C2S16
C2SEGV3	FBA3H	C2S31	C2S30	C2S29	C2S28	C2S27	C2S26	C2S25	C2S24
C2SEGV4	FBA4H	C2S39	C2S38	C2S37	C2S36	C2S35	C2S34	C2S33	C2S32
C3SEGV0	FBA8H	C3S7	C3S6	C3S5	C3S4	C3S3	C3S2	C3S1	C3S0
C3SEGV1	FBA9H	C3S15	C3S14	C3S13	C3S12	C3S11	C3S10	C3S9	C3S8
C3SEGV2	FBAAH	C3S23	C3S22	C3S21	C3S20	C3S19	C3S18	C3S17	C3S16
C3SEGV3	FBABH	C3S31	C3S30	C3S29	C3S28	C3S27	C3S26	C3S25	C3S24
C3SEGV4	FBACH	C3S39	C3S38	C3S37	C3S36	C3S35	C3S34	C3S33	C3S32

CmSn: Display data of COMm-SEGn in LCD matrix (m=0~3, n=0~39)

25.3 LCD display related configuration

25.3.1 Configure LCD refresh rate (frame rate)

The configuration of the refresh rate of the LCD refers to setting the time to scan all the dots of the entire LCD. Generally, the refresh rate of the LCD is configured to be 55 to 60 Hz, and the display effect is the best.

In the LCD module of STC8H, the refresh rate is mainly set by DBLEN, COMLENL, COMLENM, COMLENH and COMON registers.

$$\text{LCD refresh rate} = \frac{\text{LCD Clock frequency}}{(\text{DBLEN[2:0]} + \text{COMLEN[19:0]} + 1) * \text{number of COM}} \quad (\text{Hz})$$

For example: CPU operating frequency is 24MHz, select CPU clock as LCD clock source, enable COM0~COM3, then DBLEN can be set to 2, COMLEN can be set to 99997 (ie COMLENH=0x01, COMLENM=0x86, COMLENL=0x9D), thus The LCD refresh rate can be obtained as 24000000/(4*(2+99997+1))=60Hz

25.3.2 Configure LCD Flicker Rate

When the display mode of the LCD is configured as blinking mode, the blinking rate of the LCD is set by the BLANKRATE register.

$$\text{LCD blink rate} = \frac{\text{LCD refresh rate}}{\text{BLANKRATE[7:0]} * 2} \quad (\text{Hz})$$

26 LCM interface (8/16-bit color screen module I8080/M6800 interface)

Product line	LCM
STC8H1K08 family	
STC8H1K28 family	
STC8H3K64S4 family	
STC8H3K64S2 family	
STC8H8K64U family A version	
STC8H8K64U family B version	●
STC8H2K64T family	
STC8H4K64TLR family	●
STC8H4K64TLCD family	●
STC8H4K64LCD family	●

Some microcontrollers of the STC8H series integrate an LCM interface controller, which can be used to drive the current popular LCD modules. It can drive I8080 interface and M6800 interface color screen, support 8-bit and 16-bit data width.

26.1 LCM interface function pin switch

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
LCMIFCFG	FE50H	LCMIFIE	-	LCMIFIP[1:0]		LCMIFDPS[1:0]		D16_D8	M68_I80
LCMIFCFG2	FE51H		LCMIFCPS[1:0]		SETUPT[2:0]			HOLDT[1:0]	

LCMIFCPS[1:0]: LCM interface control pin selection bit

LCMIFCPS [1:0]	RS	RD signal of I8080 E signal of M6800	WR signal of I8080 RW signal of M6800
00	P4.5	P4.4	P4.2
01	P4.5	P3.7	P3.6
10	P4.0	P4.4	P4.2
11	P4.0	P3.7	P3.6

LCMIFDPS[1:0]: 8-bit data LCM interface data pin selection bit

LCMIFDPS [1:0]	D16_D8	Data byte DAT[7:0]
00	0	P2[7:0]
01	0	P6[7:0]
10	0	P2[7:0]
11	0	P6[7:0]

LCMIFDPS[1:0]: 16-bit data LCM interface data pin selection bit

LCMIFDPS [1:0]	D16_D8	High byte DAT[15:8]	Low byte DAT[7:0]
00	1	P2[7:0]	P0[7:0]
01	1	P6[7:0]	P2[7:0]
10	1	P2[7:0]	{P0[7:4],P4[7],P4[6],P4[3],P4[1]}
11	1	P6[7:0]	P7[7:0]

26.2 Registers Related to LCM

Symbol	Description	Address	Bit Address and Symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
LCMIFCFG	LCM Interface Configuration Register	FE50H	LCMIFIE	-	LCMIFIP[1:0]		LCMIFDPS[1:0]		D16_D8	M68_I80	0x00,0000
LCMIFCFG2	LCM Interface Configuration Register 2	FE51H	-	LCMIFCPS[1:0]		SETUPT[2:0]			HOLDT[1:0]		x000,0000
LCMIFCR	LCM Interface Control Register	FE52H	ENLCMIF	-	-	-	-	CMD[2:0]			0xxx,x000
LCMIFSTA	LCM Interface Status Register	FE53H	-	-	-	-	-	-	-	LCMIFIF	xxxx,xxx0
LCMIDDATL	LCM Interface low byte data	FE54H	LCMIFDAT[7:0]								0000,0000
LCMIDDATH	LCM Interface high byte data	FE55H	LCMIFDAT[15:8]								0000,0000

● LCM Interface Configuration Register (LCMIFCFG)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
LCMIFCFG	FE50H	LCMIFIE	-	LCMIFIP[1:0]		LCMIFDPS[1:0]		D16_D8	M68_I80

LCMIFIE: LCM interrupt enable control bit

0: disable LCM interface interrupt

1: enable LCM interface interrupt

LCMIFIP[1:0]: LCM interface interrupt priority control bits

LCMIFIP[1:0]	interrupt priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

LCMIFDPS[1:0]: LCM interface data pin selection bit

LCMIFDPS [1:0]	D16_D8	High byte DAT[15:8]	Low byte DAT[7:0]
00	0	N/A	P2[7:0]
01	0	N/A	P6[7:0]

10	0	N/A	P2[7:0]
11	0	N/A	P6[7:0]
00	1	P2[7:0]	P0[7:0]
01	1	P6[7:0]	P2[7:0]
10	1	P2[7:0]	{P0[7:4],P4[7],P4[6],P4[3],P4[1]}
11	1	P6[7:0]	P7[7:0]

D16_D8: LCM interface data width control bit
  0: 8-bit data width
  1: 16-bit data width
M68_I80: LCM interface mode selection bit
  0: I8080 mode
  1: M6800 mode

26.2.2 LCM Interface Configuration Register 2 (LCMIFCFG2)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
LCMIFCFG2	FE51H	-	LCMIFCPS[1:0]		SETUPT[2:0]			HOLDT[1:0]	

LCMIFCPS[1:0]: LCM interface control pin selection bit

LCMIFCPS [1:0]	RS	RD signal of I8080 E signal of M6800	WR signal of I8080 RW signal of M6800
00	P4.5	P4.4	P4.2
01	P4.5	P3.7	P3.6
10	P4.0	P4.4	P4.2
11	P4.0	P3.7	P3.6

SETUPT[2:0]: Data setup time control bit for LCM interface communication (see timing diagrams in subsequent chapters for details)
HOLDT[1:0]: Data hold time control bit for LCM interface communication (see the timing diagram in the subsequent chapters for details)

26.2.3 LCM Interface Control Register (LCMIFCR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
LCMIFCR	FE52H	ENLCMIF	-	-	-	-	CMD[2:0]		

ELCMIF: LCM interface enable control bit
  0: Disable LCM interface function
  1: Enable LCM interface function
CMD[2:0]: LCM interface trigger command

CMD[2:0]	trigger command
100	Write command
101	Write data
110	Read command/status
111	Read data

26.2.4 LCM Interface Status Register (LCMIFSTA)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
LCMIFSTA	FE53H	-	-	-	-	-	-	-	LCMIFIF

LCMIFIF: LCM interface interrupt request flag, needs to be cleared by software

26.2.5 LCM interface Data Registers (LCMIFDATL， LCMIFDATH)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
LCMIFDATL	FE54H	LCMIFDAT[7:0]							
LCMIFDATH	FE55H	LCMIFDAT[15:8]							

LCMIFDAT: LCM interface data register

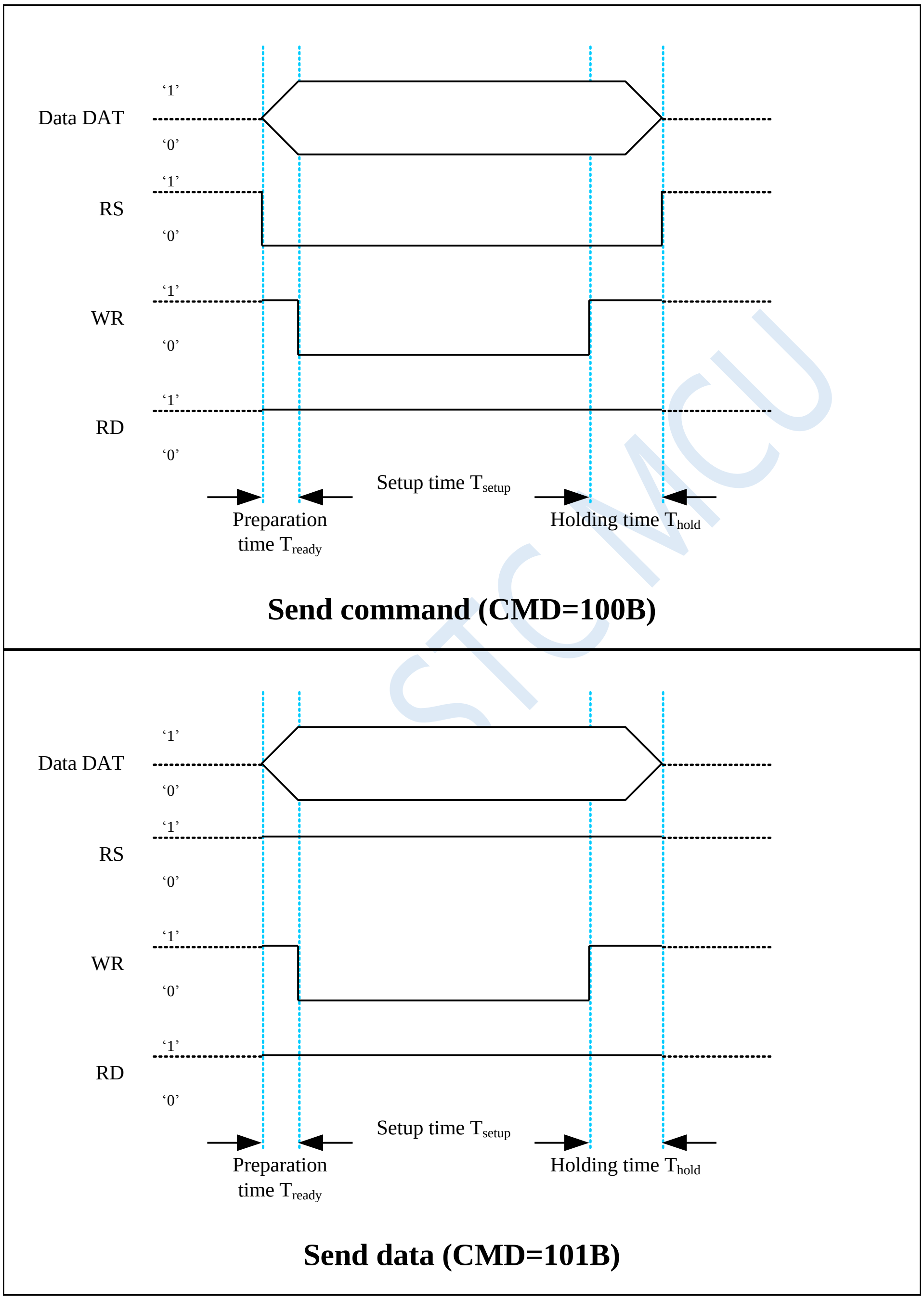
When the data width is 8 bits, only the LCMDATL data is valid.

When the data width is 16 bits, it is combined into 16-bit data by LCMDATL and LCMDATH.

26.3 I8080/M6800 mode LCM interface timing diagram

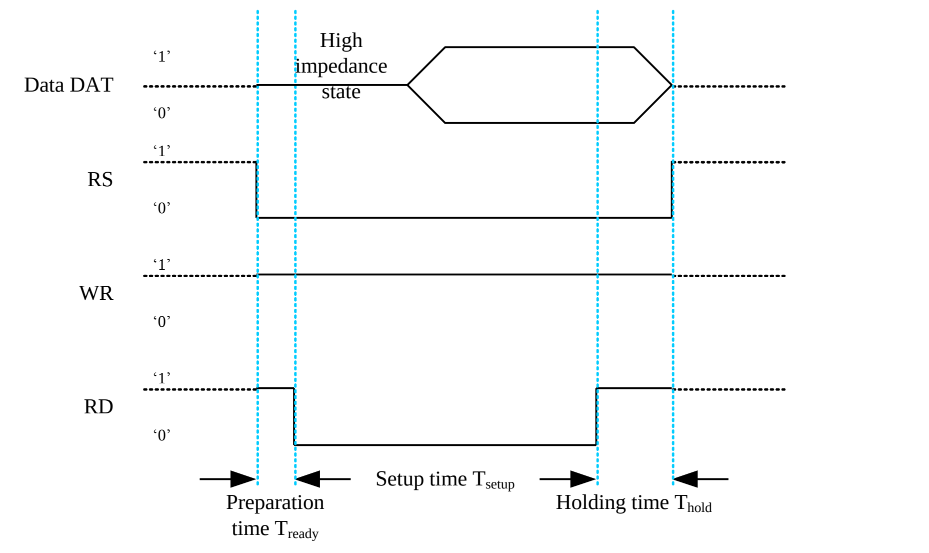
Note: T_{ready} =1 system clock
T_{setup} = (SETUPT +1) system clocks
T_{hold} = (HOLDT+1) system clocks

26.3.1 I8080 mode

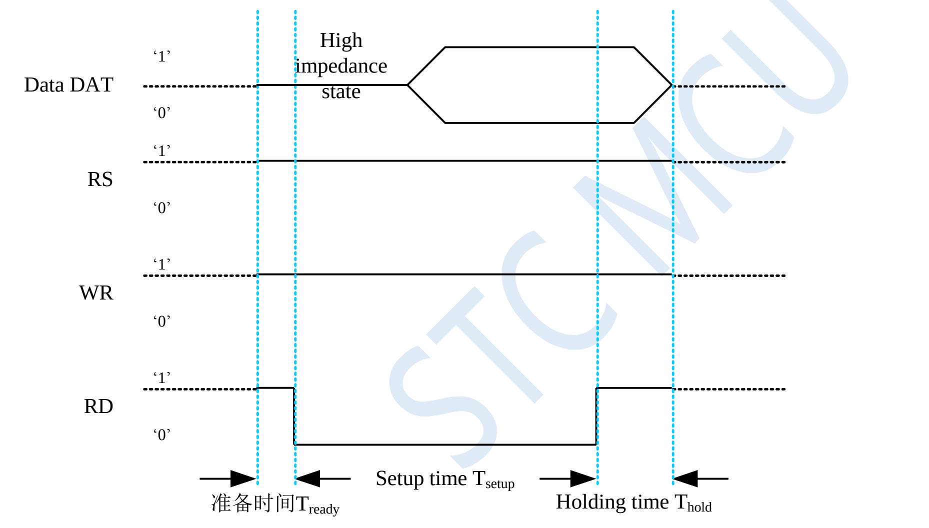


Send command (CMD=100B)

Send data (CMD=101B)

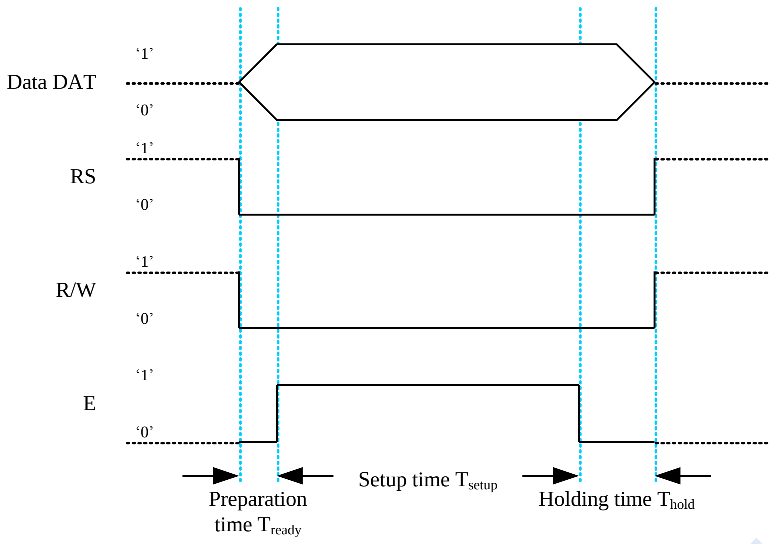


Read command (CMD=110B)

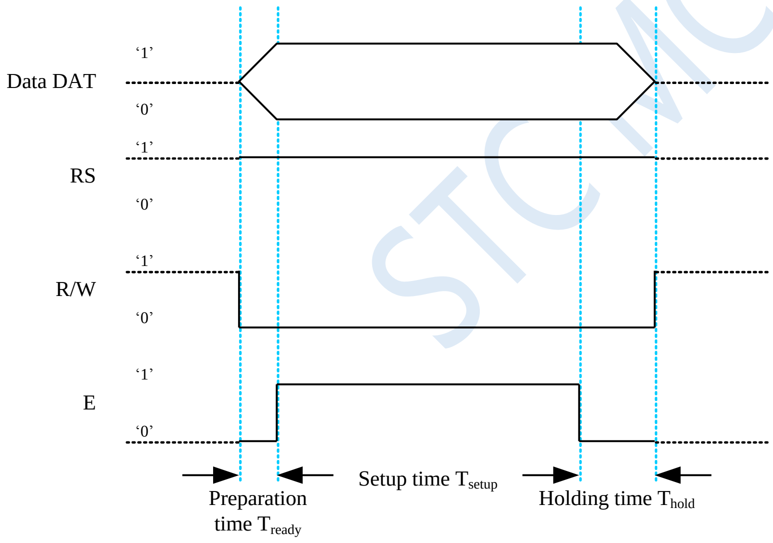


Read data (CMD=111B)

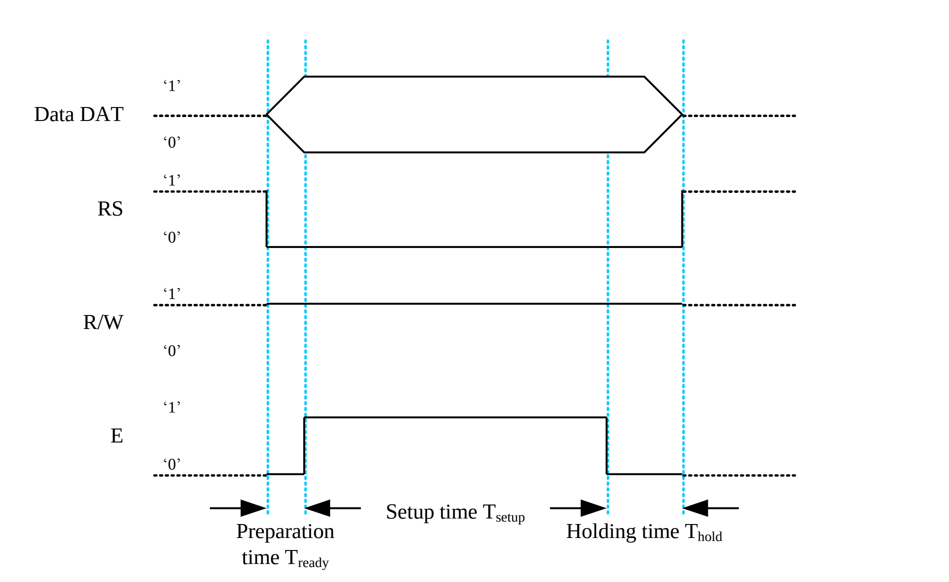
26.3.2 M6800 mode



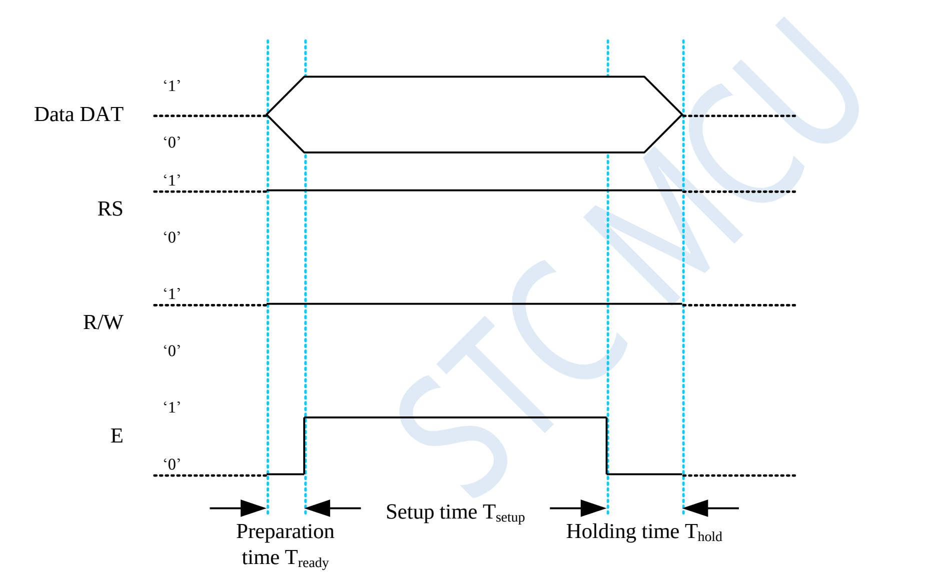
Send command (CMD=100B)



Send data (CMD=101B)



Read command (CMD=110B)



Read data (CMD=111B)

27 DMA (batch data transfer)

Product line	DMA
STC8H1K08 family	
STC8H1K28 family	
STC8H3K64S4 family	
STC8H3K64S2 family	
STC8H8K64U family A version	
STC8H8K64U family B version	●
STC8H2K64T family	
STC8H4K64TLR family	●
STC8H4K64TLCD family	●
STC8H4K64LCD family	●

Some microcomputers of the STC8H series support the function of batch data storage, that is, traditional DMA.
The following DMA operations are supported:

- M2M_DMA: read and write data from XRAM memory to XRAM memory
- ADC_DMA: automatically scan the enabled ADC channels and automatically store the converted ADC data into XRAM
- SPI_DMA: automatically exchange data between XRAM data and SPI peripherals
- UR1T_DMA: automatically send the data in XRAM through UART1
- UR1R_DMA: automatically store the data received from UART1 into XRAM
- UR2T_DMA: automatically send the data in XRAM through UART2
- UR2R_DMA: automatically store the data received from UART2 into XRAM
- UR3T_DMA: automatically send the data in XRAM through UART3
- UR3R_DMA: automatically store the data received from UART3 into XRAM
- UR4T_DMA: automatically send the data in XRAM through UART 4
- UR4R_DMA: automatically store the data received from UART4 into XRAM
- LCM_DMA: automatically exchange data between the data in XRAM and the LCM device

The maximum size of each DMA data transfer is 256 bytes.
Each DMA read and write operation to XRAM can be set to 4-level access priority, and the hardware will automatically perform the access arbitration of the XRAM bus, which will not affect CPU access to XRAM. Under the same priority, the access order of different DMAs to XRAM is as follows: SPI_DMA, UR1R_DMA, UR1T_DMA, UR2R_DMA, UR2T_DMA, UR3R_DMA, UR3T_DMA, UR4R_DMA, UR4T_DMA, LCM_DMA, M2M_DMA, ADC_DMA

27.1 Registers Related to DMA

Symbol	Description	Address	Bit Address and Symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
DMA_M2M_CFG	M2M_DMA Configuration Register	FA00H	M2MIE	-	TXACO	RXACO	M2MIP[1:0]		M2MPTY[1:0]		0x00,0000
DMA_M2M_CR	M2M_DMA Control Register	FA01H	ENM2M	TRIG	-	-	-	-	-	-	00xx,xxxx
DMA_M2M_STA	M2M_DMA Status Register	FA02H	-	-	-	-	-	-	-	M2MIF	xxxx,xxx0
DMA_M2M_AMT	M2M_DMA Total Bytes Need to be Transferred	FA03H									0000,0000
DMA_M2M_DONE	M2M_DMA Transfer Completed Bytes	FA04H									0000,0000
DMA_M2M_TXAH	M2M_DMA Send High Address	FA05H									0000,0000
DMA_M2M_TXAL	M2M_DMA Send Low Address	FA06H									0000,0000
DMA_M2M_RXAH	M2M_DMA Receive High Address	FA07H									0000,0000
DMA_M2M_RXAL	M2M_DMA Receive Low Address	FA08H									0000,0000
DMA_ADC_CFG	ADC_DMA Configuration Register	FA10H	ADCIE	-	-	-	ADCMIP[1:0]		ADCPTY[1:0]		0xxx,0000
DMA_ADC_CR	ADC_DMA Control Register	FA11H	ENADC	TRIG	-	-	-	-	-	-	00xx,xxxx
DMA_ADC_STA	ADC_DMA Status Register	FA12H	-	-	-	-	-	-	-	ADCIF	xxxx,xxx0
DMA_ADC_RXAH	ADC_DMA Receive High Address	FA17H									0000,0000
DMA_ADC_RXAL	ADC_DMA Receive Low Address	FA18H									0000,0000

DMA_ADC_CFG2	ADC_DMA Configuration Register2	FA19H	-	-	-	-	CVTIMESEL[3:0]				xxxx,0000
DMA_ADC_CHSW0	ADC_DMA Channel Enable	FA1AH	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	1000,0000
DMA_ADC_CHSW1	ADC_DMA Channel Enable	FA1BH	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0	0000,0001
DMA_SPI_CFG	SPI_DMA Configuration Register	FA20H	SPIIE	ACT_TX	ACT_RX	-	SPIIP[1:0]		SPIPTY[1:0]		000x,0000
DMA_SPI_CR	SPI_DMA Control Register	FA21H	ENSPI	TRIG_M	TRIG_S	-	-	-	-	CLRFIFO	000x,xxx0
DMA_SPI_STA	SPI_DMA Status Register	FA22H	-	-	-	-	-	TXOVW	RXLOSS	SPIIF	xxxx,x000
DMA_SPI_AMT	SPI_DMA Total Bytes Need to be Transferred	FA23H									0000,0000
DMA_SPI_DONE	SPI_DMA Transfer Completed Bytes	FA24H									0000,0000
DMA_SPI_TXAH	SPI_DMA Send High Address	FA25H									0000,0000
DMA_SPI_TXAL	SPI_DMA Send Low Address	FA26H									0000,0000
DMA_SPI_RXAH	SPI_DMA Receive High Address	FA27H									0000,0000
DMA_SPI_RXAL	SPI_DMA Receive Low Address	FA28H									0000,0000
DMA_SPI_CFG2	SPI_DMA Configuration Register2	FA29H	-	-	-	-	-	WRPSS	SSS[1:0]		xxxx,x000
DMA_UR1T_CFG	UR1T_DMA Configuration Register	FA30H	UR1TIE	-	-	-	UR1TIP[1:0]		UR1TPTY[1:0]		0xxx,0000
DMA_UR1T_CR	UR1T_DMA Control Register	FA31H	ENUR1T	TRIG	-	-	-	-	-	-	00xx,xxxx
DMA_UR1T_STA	UR1T_DMA Status Register	FA32H	-	-	-	-	-	TXOVW	-	UR1TIF	xxxx,x0x0
DMA_UR1T_AMT	UR1T_DMA Total Bytes Need to be Transferred	FA33H									0000,0000
DMA_UR1T_DONE	UR1T_DMA Transfer Completed Bytes	FA34H									0000,0000
DMA_UR1T_TXAH	UR1T_DMA Send High Address	FA35H									0000,0000
DMA_UR1T_TXAL	UR1T_DMA Send Low Address	FA36H									0000,0000
DMA_UR1R_CFG	UR1R_DMA Configuration Register	FA38H	UR1RIE	-	-	-	UR1RIP[1:0]		UR1RPTY[1:0]		0xxx,0000
DMA_UR1R_CR	UR1R_DMA Control Register	FA39H	ENUR1R	-	TRIG	-	-	-	-	CLRFIFO	0x0x,xxx0
DMA_UR1R_STA	UR1R_DMA Status Register	FA3AH	-	-	-	-	-	-	RXLOSS	UR1RIF	xxxx,xx00
DMA_UR1R_AMT	UR1R_DMA Total Bytes Need to be Transferred	FA3BH									0000,0000
DMA_UR1R_DONE	UR1R_DMA Transfer Completed Bytes	FA3CH									0000,0000
DMA_UR1R_TXAH	UR1R_DMA Send High Address	FA3DH									0000,0000
DMA_UR1R_TXAL	UR1R_DMA Send Low Address	FA3EH									0000,0000
DMA_UR2T_CFG	UR2T_DMA Configuration Register	FA40H	UR2TIE	-	-	-	UR2TIP[1:0]		UR2TPTY[1:0]		0xxx,0000
DMA_UR2T_CR	UR2T_DMA Control Register	FA41H	ENUR2T	TRIG	-	-	-	-	-	-	00xx,xxxx
DMA_UR2T_STA	UR2T_DMA Status Register	FA42H	-	-	-	-	-	TXOVW	-	UR2TIF	xxxx,x0x0
DMA_UR2T_AMT	UR2T_DMA Total Bytes Need to be Transferred	FA43H									0000,0000
DMA_UR2T_DONE	UR2T_DMA Transfer Completed Bytes	FA44H									0000,0000
DMA_UR2T_TXAH	UR2T_DMA Send High Address	FA45H									0000,0000
DMA_UR2T_TXAL	UR2T_DMA Send Low Address	FA46H									0000,0000
DMA_UR2R_CFG	UR2R_DMA Configuration Register	FA48H	UR2RIE	-	-	-	UR2RIP[1:0]		UR2RPTY[1:0]		0xxx,0000
DMA_UR2R_CR	UR2R_DMA Control Register	FA49H	ENUR2R	-	TRIG	-	-	-	-	CLRFIFO	0x0x,xxx0
DMA_UR2R_STA	UR2R_DMA Status Register	FA4AH	-	-	-	-	-	-	RXLOSS	UR2RIF	xxxx,xx00
DMA_UR2R_AMT	UR2R_DMA Total Bytes Need to be Transferred	FA4BH									0000,0000
DMA_UR2R_DONE	UR2R_DMA Transfer Completed Bytes	FA4CH									0000,0000
DMA_UR2R_TXAH	UR2R_DMA Send High Address	FA4DH									0000,0000
DMA_UR2R_TXAL	UR2R_DMA Send Low Address	FA4EH									0000,0000
DMA_UR3T_CFG	UR3T_DMA Configuration Register	FA50H	UR3TIE	-	-	-	UR3TIP[1:0]		UR3TPTY[1:0]		0xxx,0000
DMA_UR3T_CR	UR3T_DMA Control Register	FA51H	ENUR3T	TRIG	-	-	-	-	-	-	00xx,xxxx
DMA_UR3T_STA	UR3T_DMA Status Register	FA52H	-	-	-	-	-	TXOVW	-	UR3TIF	xxxx,x0x0
DMA_UR3T_AMT	UR3T_DMA Total Bytes Need to be Transferred	FA53H									0000,0000
DMA_UR3T_DONE	UR3T_DMA Transfer Completed Bytes	FA54H									0000,0000
DMA_UR3T_TXAH	UR3T_DMA Send High Address	FA55H									0000,0000
DMA_UR3T_TXAL	UR3T_DMA Send Low Address	FA56H									0000,0000
DMA_UR3R_CFG	UR3R_DMA Configuration Register	FA58H	UR3RIE	-	-	-	UR3RIP[1:0]		UR3RPTY[1:0]		0xxx,0000
DMA_UR3R_CR	UR3R_DMA Control Register	FA59H	ENUR3R	-	TRIG	-	-	-	-	CLRFIFO	0x0x,xxx0
DMA_UR3R_STA	UR3R_DMA Status Register	FA5AH	-	-	-	-	-	-	RXLOSS	UR3RIF	xxxx,xx00
DMA_UR3R_AMT	UR3R_DMA Total Bytes Need to be Transferred	FA5BH									0000,0000
DMA_UR3R_DONE	UR3R_DMA Transfer Completed Bytes	FA5CH									0000,0000
DMA_UR3R_TXAH	UR3R_DMA Send High Address	FA5DH									0000,0000
DMA_UR3R_TXAL	UR3R_DMA Send Low Address	FA5EH									0000,0000
DMA_UR4T_CFG	UR4T_DMA Configuration Register	FA60H	UR4TIE	-	-	-	UR4TIP[1:0]		UR4TPTY[1:0]		0xxx,0000
DMA_UR4T_CR	UR4T_DMA Control Register	FA61H	ENUR4T	TRIG	-	-	-	-	-	-	00xx,xxxx
DMA_UR4T_STA	UR4T_DMA Status Register	FA62H	-	-	-	-	-	TXOVW	-	UR4TIF	xxxx,x0x0
DMA_UR4T_AMT	UR4T_DMA Total Bytes Need to be Transferred	FA63H									0000,0000

DMA_UR4T_DONE	UR4T_DMA Transfer Completed Bytes	FA64H									0000,0000
DMA_UR4T_TXAH	UR4T_DMA Send High Address	FA65H									0000,0000
DMA_UR4T_TXAL	UR4T_DMA Send Low Address	FA66H									0000,0000
DMA_UR4R_CFG	UR4R_DMA Configuration Register	FA68H	UR4RIE	-	-	-	UR4RIP[1:0]		UR4RPTY[1:0]		0xxx,0000
DMA_UR4R_CR	UR4R_DMA Control Register	FA69H	ENUR4R	-	TRIG	-	-	-	-	CLRFIFO	0x0x,xxx0
DMA_UR4R_STA	UR4R_DMA Status Register	FA6AH	-	-	-	-	-	-	RXLOSS	UR4RIF	xxxx,xx00
DMA_UR4R_AMT	UR4R_DMA Total Bytes Need to be Transferred	FA6BH									0000,0000
DMA_UR4R_DONE	UR4R_DMA Transfer Completed Bytes	FA6CH									0000,0000
DMA_UR4R_TXAH	UR4R_DMA Send High Address	FA6DH									0000,0000
DMA_UR4R_TXAL	UR4R_DMA Send Low Address	FA6EH									0000,0000
DMA_LCM_CFG	LCM_DMA Configuration Register	FA70H	LCMIE	-	-	-	LCMIP[1:0]		LCMPTY[1:0]		0xxx,0000
DMA_LCM_CR	LCM_DMA Control Register	FA71H	ENLCM	TRIGWC	TRIGWD	TRIGRC	TRIGRD	-	-	-	0000,0xxx
DMA_LCM_STA	LCM_DMA Status Register	FA72H	-	-	-	-	-	-	TXOVW	LCMIF	xxxx,xx00
DMA_LCM_AMT	LCM_DMA Total Bytes Need to be Transferred	FA73H									0000,0000
DMA_LCM_DONE	LCM_DMA Transfer Completed Bytes	FA74H									0000,0000
DMA_LCM_TXAH	LCM_DMA Send High Address	FA75H									0000,0000
DMA_LCM_TXAL	LCM_DMA Send Low Address	FA76H									0000,0000
DMA_LCM_RXAH	LCM_DMA Receive High Address	FA77H									0000,0000
DMA_LCM_RXAL	LCM_DMA Receive Low Address	FA78H									0000,0000

27.2 Data read and write between memory and memory (M2M_DMA)

27.2.1 M2M_DMA Configuration Register (DMA_M2M_CFG)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_M2M_CFG	FA00H	M2MIE	-	TXACO	RXACO	M2MIP[1:0]		M2MPTY[1:0]	

M2MIE: M2M_DMA interrupt enable control bit

0: Disable M2M_DMA interrupt

1: Enable M2M_DMA interrupt

TXACO: M2M_DMA source address (read address) changes direction

0: The address is automatically incremented after the data read is completed

1: The address is automatically decremented after the data read is completed

RXACO: M2M_DMA target address (write address) changed direction

0: The address is automatically incremented after data writing is completed

1: The address is automatically decremented after data writing is completed

M2MIP[1:0]: M2M_DMA interrupt priority control bits

M2MIP[1:0]	Interrupt priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

M2MPTY[1:0]: M2M_DMA Data bus access priority control bits

M2MPTY [1:0]	Bus access priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

27.2.2 M2M_DMA Control Register (DMA_M2M_CR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_M2M_CR	FA01H	ENM2M	TRIG	-	-	-	-	-	-

ENM2M: M2M_DMA function enable control bit

0: Disable M2M_DMA function

1: Enable M2M_DMA function

TRIG: M2M_DMA data read and write trigger control bit

0: Write 0 is invalid

1: Write 1 to start M2M_DMA operation.

27.2.3 M2M_DMA Status Register (DMA_M2M_STA)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_M2M_STA	FA02H	-	-	-	-	-	-	-	M2MIF

M2MIF: M2M_DMA interrupt request flag bit. When the M2M_DMA operation is completed, the hardware automatically sets M2MIF to 1, and if the M2M_DMA interrupt is enabled, the interrupt service routine is entered. The flag bit needs to be cleared by software.

27.2.4 M2M_DMA transfer total byte register (DMA_M2M_AMT)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_M2M_AMT	FA03H								

DMA_M2M_AMT: Set the number of bytes that need to read and write data. **Note: The actual number of bytes read and written is (DMA_M2M_AMT+1), that is, when DMA_M2M_AMT is set to 0, 1 byte is read and written, and when DMA_M2M_AMT is set to 255, 256 bytes are read and written.**

27.2.5 M2M_DMA transfer complete byte register (DMA_M2M_DONE)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_M2M_DONE	FA04H								

DMA_M2M_DONE: The number of bytes that have been read and written currently.

27.2.6 M2M_DMA Send Address Registers (DMA_M2M_TXAx)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_M2M_TXAH	FA05H	ADDR[15:8]							
DMA_M2M_TXAL	FA06H	ADDR[7:0]							

DMA_M2M_TXA: Set the source address when reading and writing data. Data will be read from this address when the M2M_DMA operation is performed.

27.2.7 M2M_DMA Receive Address Registers (DMA_M2M_RXAx)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_M2M_RXAH	FA07H	ADDR[15:8]							
DMA_M2M_RXAL	FA08H	ADDR[7:0]							

DMA_M2M_RXA: Set the target address when reading and writing data. Data will be written from this address when the M2M_DMA operation is performed.

27.3 ADC Automatic Data Storage (ADC_DMA)

27.3.1 ADC_DMA Configuration Register (DMA_ADC_CFG)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_ADC_CFG	FA10H	ADCIE	-			ADCIP[1:0]		ADCPTY[1:0]	

ADCIE: ADC_DMA interrupt enable control bit

0: Disable ADC_DMA interrupt

1: Enable ADC_DMA interrupt

ADCIP[1:0]: ADC_DMA interrupt priority control bits

ADCIP[1:0]	Interrupt priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

ADCPTY[1:0]：ADC_DMA Data bus access priority control bits

ADCPTY [1:0]	Bus access priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

27.3.2 ADC_DMA Control Register (DMA_ADC_CR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_ADC_CR	FA11H	ENADC	TRIG	-	-	-	-	-	-

ENADC: ADC_DMA function enable control bit

0: Disable ADC_DMA function

1: Enable ADC_DMA function

TRIG: ADC_DMA operation trigger control bit

0: Write 0 is invalid

1: Write 1 to start ADC_DMA operation.

27.3.3 ADC_DMA Status Register (DMA_ADC_STA)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_ADC_STA	FA12H	-	-	-	-	-	-	-	ADCIF

ADCIF: ADC_DMA interrupt request flag bit. After ADC_DMA completes scanning all enabled ADC channels, the hardware automatically sets ADCIF to 1. If the ADC_DMA interrupt is enabled, the interrupt service routine is entered. The flag bit needs to be cleared by software.

27.3.4 ADC_DMA Receive Address Registers (DMA_ADC_RXAx)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_ADC_RXAH	FA17H	ADDR[15:8]							
DMA_ADC_RXAL	FA18H	ADDR[7:0]							

DMA_ADC_RXA: Set the storage address of ADC conversion data during ADC_DMA operation.

27.3.5 ADC_DMA Configuration Register 2 (DMA_ADC_CFG2)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_ADC_CFG2	FA19H	-	-	-	-	CVTIMESEL[3:0]			

CVTIMESEL[3:0]: Set the number of ADC conversions for each ADC channel during ADC_DMA operation.

CVTIMESEL[3:0]	Number of ADC conversions
0xxx	1
1000	2
1001	4
1010	8
1011	16
1100	32
1101	64
1110	128
1111	256

27.3.6 ADC_DMA Channel Enable Registers (DMA_ADC_CHSWx)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_ADC_CHSW0	FA1AH	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8
DMA_ADC_CHSW1	FA1BH	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

CHn: The ADC channel to be scanned automatically when setting ADC_DMA operation. Channel scanning always starts from the lower-numbered channel.

27.3.7 Data storage format of ADC_DMA

Note: ADC conversion speed and conversion result alignment are set by ADC related registers.
XRAM[DMA_ADC_RXA+0] = high byte of the 1st ADC conversion result of the 1st enabled channel;
XRAM[DMA_ADC_RXA+1] = low byte of the 1st ADC conversion result of the 1st enabled channel;
XRAM[DMA_ADC_RXA+2] = high byte of the 2nd ADC conversion result of the 1st enabled channel;
XRAM[DMA_ADC_RXA+3] = low byte of the 2nd ADC conversion result of the 1st enabled channel;
...
XRAM[DMA_ADC_RXA+2n-2] = high byte of the nth ADC conversion result of the 1st enabled channel;
XRAM[DMA_ADC_RXA+2n-1] = low byte of the nth ADC conversion result of the 1st enabled channel;
XRAM[DMA_ADC_RXA+2n] = ADC channel number of 1st channel;
XRAM[DMA_ADC_RXA+2n+1] = remainder after the average value of the n ADC conversion results of the 1st channel;
XRAM[DMA_ADC_RXA+2n+2] = high byte of the average value of the n ADC conversion results of the 1st channel;
XRAM[DMA_ADC_RXA+2n+3] = low byte of the average value of the n ADC conversion results of the 1st channel;

XRAM[DMA_ADC_RXA+(2n+3)+0] = high byte of the 1st ADC conversion result of the 2nd enabled channel;
XRAM[DMA_ADC_RXA+(2n+3)+1] = low byte of the 1st ADC conversion result of the 2nd enabled channel;
XRAM[DMA_ADC_RXA+(2n+3)+2] = high byte of the 2nd ADC conversion result of the 2nd enabled channel;
XRAM[DMA_ADC_RXA+(2n+3)+3] = low byte of the 2nd ADC conversion result of the 2nd enabled channel;
...
XRAM[DMA_ADC_RXA+(2n+3)+2n-2] = high byte of the nth ADC conversion result of the enabled 2nd channel;
XRAM[DMA_ADC_RXA+(2n+3)+2n-1] = low byte of the nth ADC conversion result of the enabled 2nd channel;
XRAM[DMA_ADC_RXA+(2n+3)+2n] = ADC channel number of 2nd channel;
XRAM[DMA_ADC_RXA+(2n+3)+2n+1] = remainder after the average value of the n ADC conversion results of 2nd channel;
XRAM[DMA_ADC_RXA+(2n+3)+2n+2] = high byte of the average value of n ADC conversion results of 2nd channel;
XRAM[DMA_ADC_RXA+(2n+3)+2n+3] = low byte of the average value of n ADC conversion results of 2nd channel;
...
XRAM[DMA_ADC_RXA+(m-1)(2n+3)+0] = high byte of the 1st ADC conversion result of the enabled mth channel;
XRAM[DMA_ADC_RXA+(m-1)(2n+3)+1] = low byte of the 1st ADC conversion result of the enabled mth channel;
XRAM[DMA_ADC_RXA+(m-1)(2n+3)+2] = high byte of the 2nd ADC conversion result of the enabled mth channel;
XRAM[DMA_ADC_RXA+(m-1)(2n+3)+3] = low byte of the 2nd ADC conversion result of the enabled mth channel;
...
XRAM[DMA_ADC_RXA+(m-1)(2n+3)+2n-2] = high byte of the nth ADC conversion result of the enabled mth channel;
XRAM[DMA_ADC_RXA+(m-1)(2n+3)+2n-1] = low byte of the nth ADC conversion result of the enabled mth channel;
XRAM[DMA_ADC_RXA+(m-1)(2n+3)+2n] = ADC channel number of the mth channel;
XRAM[DMA_ADC_RXA+(m-1)(2n+3)+2n+1] = remainder after the average value of the n ADC conversion results of the mth channel;
XRAM[DMA_ADC_RXA+(m-1)(2n+3)+2n+2] = high byte of the average value of the n ADC conversion results of the mth channel;
XRAM[DMA_ADC_RXA+(m-1)(2n+3)+2n+3] = low byte of the average value of the n ADC conversion results of the mth channel;

The form is as follows:

ADC channel	Offset address	Data
1st channel	0	high byte of the 1st ADC conversion result of the 1st enabled channel
	1	low byte of the 1st ADC conversion result of the 1st enabled channel
	2	high byte of the 2nd ADC conversion result of the 1st enabled channel
	3	low byte of the 2nd ADC conversion result of the 1st enabled channel
	...	...
	2n-2	high byte of the nth ADC conversion result of the 1st enabled channel
	2n-1	low byte of the nth ADC conversion result of the 1st enabled channel
	2n	ADC channel number of 1st channel
	2n+1	remainder after the average value of the n ADC conversion results of the 1st channel
	2n+2	high byte of the average value of the n ADC conversion results of the 1st channel
	2n+3	low byte of the average value of the n ADC conversion results of the 1st channel
2nd channel	(2n+3) + 0	high byte of the 1st ADC conversion result of the 2nd enabled channel
	(2n+3) + 1	low byte of the 1st ADC conversion result of the 2nd enabled channel
	(2n+3) + 2	high byte of the 2nd ADC conversion result of the 2nd enabled channel
	(2n+3) + 3	low byte of the 2nd ADC conversion result of the 2nd enabled channel
	...	...
	(2n+3) + 2n-2	high byte of the nth ADC conversion result of the enabled 2nd channel
	(2n+3) + 2n-1	low byte of the nth ADC conversion result of the enabled 2nd channel
	(2n+3) + 2n	ADC channel number of 2nd channel
	(2n+3) + 2n+1	remainder after the average value of the n ADC conversion results of 2nd channel
	(2n+3) + 2n+2	high byte of the average value of n ADC conversion results of 2nd channel
	(2n+3) + 2n+3	low byte of the average value of n ADC conversion results of 2nd channel
	...	...
mth channel	(m-1)(2n+3) + 0	high byte of the 1st ADC conversion result of the enabled mth channel
	(m-1)(2n+3) + 1	low byte of the 1st ADC conversion result of the enabled mth channel
	(m-1)(2n+3) + 2	high byte of the 2nd ADC conversion result of the enabled mth channel
	(m-1)(2n+3) + 3	low byte of the 2nd ADC conversion result of the enabled mth channel
	...	...
	(m-1)(2n+3) + 2n-2	high byte of the nth ADC conversion result of the enabled mth channel
	(m-1)(2n+3) + 2n-1	low byte of the nth ADC conversion result of the enabled mth channel
	(m-1)(2n+3) + 2n	ADC channel number of the mth channel
	(m-1)(2n+3) + 2n+1	remainder after the average value of the n ADC conversion results of the mth channel
	(m-1)(2n+3) + 2n+2	high byte of the average value of the n ADC conversion results of the mth channel
	(m-1)(2n+3) + 2n+3	low byte of the average value of the n ADC conversion results of the mth channel

27.4 Data exchange between SPI and memory (SPI_DMA)

27.4.1 SPI_DMA Configuration Register（DMA_SPI_CFG）

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_SPI_CFG	FA20H	SPIIE	ACT_TX	ACT_RX	-	SPIIP[1:0]		SPIPTY[1:0]	

SPIIE: SPI_DMA interrupt enable control bit

0: Disable SPI_DMA interrupt

1: Enable SPI_DMA interrupt

ACT_TX: SPI_DMA transmit data control bit

0: Disable SPI_DMA to send data. In master mode, SPI only sends clock to SCLK port, but does not read data from XRAM, nor send data to MOSI port; in slave mode, SPI does not read data from XRAM, nor send data to MISO port.

1: Enable SPI_DMA to send data. In master mode, SPI sends clock to SCLK port, and reads data from XRAM and sends data to MOSI port; in slave mode, SPI reads data from XRAM and sends data to MISO port.

ACT_RX: SPI_DMA receive data control bit

0: Disable SPI_DMA to receive data. In master mode, SPI only sends clock to SCLK port, but does not read data from MISO port, nor write data to XRAM; in slave mode, SPI does not read data from MOSI port, nor write data to XRAM.

1: Enable SPI_DMA to receive data. In master mode, SPI sends clock to SCLK port, and reads data from MISO port and writes data to XRAM; in slave mode, SPI reads data from MOSI port and writes XRAM.

SPIIP[1:0]: SPI_DMA interrupt priority control bits

SPIIP[1:0]	Interrupt priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

SPIPTY[1:0]: SPI_DMA Data bus access priority control bits

SPIPTY [1:0]	Bus access priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

27.4.2 SPI_DMA Control Register (DMA_SPI_CR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_SPI_CR	FA21H	ENSPI	TRIG_M	TRIG_S	-	-	-	-	CLRFIFO

ENSPI: SPI_DMA function enable control bit

0: Disable SPI_DMA function

1: Enable SPI_DMA function

TRIG_M: SPI_DMA master mode trigger control bit

0: Write 0 is invalid

1: Write 1 to start SPI_DMA master mode operation.

TRIG_S: SPI_DMA slave mode trigger control bit

0: Write 0 is invalid

1: Write 1 to start SPI_DMA slave mode operation.

CLRFIFO: Clear SPI_DMA receive FIFO control bit

0: Write 0 is invalid

1: Before starting the SPI_DMA operation, clear the built-in FIFO of SPI_DMA firstly.

27.4.3 SPI_DMA Status Register (DMA_SPI_STA)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_SPI_STA	FA22H	-	-	-	-	-	TXOVW	RXLOSS	SPIIF

SPIIF: SPI_DMA interrupt request flag bit. After the SPI_DMA data exchange is completed, the hardware automatically sets SPIIF to 1. If the SPI_DMA interrupt is enabled, the interrupt service routine is entered. The flag bit needs to be cleared by software.

RXLOSS: SPI_DMA receive data discard flag. During the SPI_DMA operation, when the XRAM bus is too busy to clear the receive FIFO of the SPI_DMA and the data received by the SPI_DMA is automatically discarded, the hardware automatically sets RXLOSS to 1. The flag bit needs to be cleared by software.

TXOVW: SPI_DMA data coverage flag. During the data transfer process of SPI_DMA, when the host mode SPI writes the SPDAT register to trigger the SPI data transfer again, the data transfer will fail, and the hardware will automatically set TXOVW to 1. The flag bit needs to be cleared by software.

27.4.4 SPI_DMA transfer total byte register (DMA_SPI_AMT)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_SPI_AMT	FA23H								

DMA_SPI_AMT: Set the number of bytes that need to read and write data. **Note: The actual number of bytes read and written is (DMA_SPI_AMT+1), that is, when DMA_SPI_AMT is set to 0, 1 byte is transferred, and when DMA_SPI_AMT is set to 255, 256 bytes are transferred.**

27.4.5 SPI_DMA transfer complete byte register (DMA_SPI_DONE)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_SPI_DONE	FA24H								

DMA_SPI_DONE: The number of bytes that have been read and written currently.

27.4.6 SPI_DMA Send Address Registers (DMA_SPI_TXAx)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_SPI_TXAH	FA25H	ADDR[15:8]							
DMA_SPI_TXAL	FA26H	ADDR[7:0]							

DMA_SPI_TXA: Set the source address when reading and writing data. Data will be read from this address when the SPI_DMA operation is performed.

27.4.7 SPI_DMA Receive Address Registers (DMA_SPI_RXAx)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_SPI_RXAH	FA27H	ADDR[15:8]							
DMA_SPI_RXAL	FA28H	ADDR[7:0]							

DMA_SPI_RXA: Set the target address when reading and writing data. Data will be written from this address when the SPI_DMA operation is performed.

27.4.8 SPI_DMA Configuration Register 2 (DMA_SPI_CFG2)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_SPI_CFG2	FA29H	-	-	-	-	-	WRPSS	SSS[1:0]	

WRPSS: Enable SS pin control bit during SPI_DMA process

0: During the SPI_DMA transfer process, the SS pin is not automatically controlled

1: During the SPI_DMA transfer process, the SS pin is automatically pulled down, and the original state is automatically restored after the transfer is completed.

SSS[1:0]: During the SPI_DMA process, control the SS selection bit automatically

SSS[1:0]	SS pin
00	P1.2
01	P2.2
10	P7.4
11	P3.5

27.5 Data exchange between UART1 and memory (UR1T_DMA, UR1R_DMA)

27.5.1 UR1T_DMA Configuration Register (DMA_UR1T_CFG)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR1T_CFG	FA30H	UR1TIE	-	-	-	UR1TIP[1:0]		UR1TPTY[1:0]	

UR1TIE: UR1T_DMA interrupt enable control bit
0: Disable UR1T_DMA interrupt
1: Enable UR1T_DMA interrupt

UR1TIP[1:0]: UR1T_DMA interrupt priority control bits

UR1TIP[1:0]	Interrupt priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

UR1TPTY[1:0]: UR1T_DMA Data bus access priority control bits

UR1TPTY [1:0]	Bus access priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

27.5.2 UR1T_DMA Control Register (DMA_UR1T_CR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR1T_CR	FA31H	ENUR1T	TRIG	-	-	-	-	-	-

ENUR1T: UR1T_DMA function enable control bit
0: Disable UR1T_DMA function
1: Enable UR1T_DMA function

TRIG: UR1T_DMA UART1 transmit trigger control bit
0: Write 0 is invalid
1: Write 1 to start UR1T_DMA automatically sending data.

27.5.3 UR1T_DMA Status Register (DMA_UR1T_STA)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR1T_STA	FA32H	-	-	-	-	-	TXOVW	-	UR1TIF

UR1TIF: UR1T_DMA interrupt request flag bit. When the UR1T_DMA data transmission is completed, the hardware automatically sets UR1TIF to 1, and if the UR1T_DMA interrupt is enabled, the interrupt service routine is entered. The flag bit needs to be cleared by software.

TXOVW: UR1T_DMA data coverage flag. When UR1T_DMA is in the process of data transmission, and the UART writes the SBUF register to trigger the UART to send data again, the data transmission will fail. At this time, the hardware will automatically set TXOVW to 1. The flag bit needs to be cleared by software.

27.5.4 UR1T_DMA transfer total byte register (DMA_UR1T_AMT)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR1T_AMT	FA33H								

DMA_UR1T_AMT: Set the number of bytes of data that needs to be automatically sent.
Note: The actual number of bytes is (DMA_UR1T_AMT+1), that is, when DMA_UR1T_AMT is set to 0, 1

byte is transferred, and when DMA_UR1T_AMT is set to 255, 256 bytes are transferred.

27.5.5 UR1T_DMA transfer complete byte register (DMA_UR1T_DONE)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR1T_DONE	FA34H								

DMA_UR1T_DONE: The number of bytes that have been sent so far.

27.5.6 UR1T_DMA Send Address Registers (DMA_UR1T_TXAx)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR1T_TXAH	FA35H	ADDR[15:8]							
DMA_UR1T_TXAL	FA36H	ADDR[7:0]							

DMA_UR1T_TXA: Set the source address of automatic data transmission. Data is read from this address when performing a UR1T_DMA operation.

27.5.7 UR1R_DMA Configuration Register (DMA_UR1R_CFG)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR1R_CFG	FA38H	UR1RIE	-	-	-	UR1RIP[1:0]		UR1RPTY[1:0]	

UR1RIE: UR1R_DMA interrupt enable control bit

0: Disable UR1R_DMA interrupt

1: Enable UR1R_DMA interrupt

UR1RIP[1:0]: UR1R_DMA interrupt priority control bits

UR1RIP[1:0]	Interrupt priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

UR1RPTY[1:0]: UR1R_DMA Data bus access priority control bits

UR1RPTY [1:0]	Bus access priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

27.5.8 UR1R_DMA Control Register (DMA_UR1R_CR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR1R_CR	FA39H	ENUR1R	-	TRIG	-	-	-	-	CLRFIFO

ENUR1R: UR1R_DMA function enable control bit

0: Disable UR1R_DMA function

1: Enable UR1R_DMA function

TRIG: UR1R_DMA UART1 receive trigger control bit

0: Write 0 is invalid

1: Write 1 to start UR1R_DMA receiving data automatically

CLRFIFO: Clear UR1R_DMA receive FIFO control bit

0: Write 0 is invalid

1: Before starting the UR1R_DMA operation, clear the built-in FIFO of the UR1R_DMA firstly

27.5.9 UR1R_DMA Status Register (DMA_UR1R_STA)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR1R_STA	FA3AH	-	-	-	-	-	-	RXLOSS	UR1RIF

UR1RIF: UR1R_DMA interrupt request flag bit. When UR1R_DMA receives data, the hardware will automatically set UR1RIF to 1. If the UR1R_DMA interrupt is enabled, it will enter the interrupt service routine. The flag bit needs to be cleared by software

RXLOSS: UR1R_DMA receive data discard flag. During the UR1R_DMA operation, when the XRAM bus is too busy to clear the receive FIFO of the UR1R_DMA and the data received by the UR1R_DMA is automatically discarded, the hardware will automatically set RXLOSS to 1. The flag bit needs to be cleared by software

27.5.10 UR1R_DMA transfer total byte register (DMA_UR1R_AMT)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR1R_AMT	FA3BH								

DMA_UR1R_AMT: Set the number of data bytes that need to automatically receive. **Note: The actual number of bytes is (DMA_UR1R_AMT+1), that is, when DMA_UR1R_AMT is set to 0, 1 byte is transferred, and when DMA_UR1R_AMT is set to 255, 256 bytes are transferred.**

27.5.11 UR1R_DMA transfer complete byte register (DMA_UR1R_DONE)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR1R_DONE	FA3CH								

DMA_UR1R_DONE: The number of bytes that have been received currently.

27.5.12 UR1R_DMA Receive Address Registers (DMA_UR1T_RXAx)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR1R_RXAH	FA3DH	ADDR[15:8]							
DMA_UR1R_RXAL	FA3EH	ADDR[7:0]							

DMA_UR1R_RXA: Set the target address for automatically receiving data. Data will be written from this address when performing a UR1R_DMA operation.

27.6 Data exchange between UART2 and memory (UR2T_DMA, UR2R_DMA)

27.6.1 UR2T_DMA Configuration Register (DMA_UR2T_CFG)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR2T_CFG	FA40H	UR2TIE	-	-	-	UR2TIP[1:0]		UR2TPTY[1:0]	

UR2TIE: UR2T_DMA interrupt enable control bit

0: Disable UR2T_DMA interrupt

1: Enable UR2T_DMA interrupt

UR2TIP[1:0]: UR2T_DMA interrupt priority control bits

UR2TIP[1:0]	Interrupt priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

UR2TPTY[1:0]: UR2T_DMA Data bus access priority control bits

UR2TPTY [1:0]	Bus access priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

27.6.2 UR2T_DMA Control Register (DMA_UR2T_CR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR2T_CR	FA41H	ENUR2T	TRIG	-	-	-	-	-	-

ENUR2T: UR2T_DMA function enable control bit

0: Disable UR2T_DMA function

1: Enable UR2T_DMA function

TRIG: UR2T_DMA UART1 transmit trigger control bit

0: Write 0 is invalid

1: Write 1 to start UR2T_DMA automatically sending data.

27.6.3 UR2T_DMA Status Register (DMA_UR2T_STA)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR2T_STA	FA42H	-	-	-	-	-	TXOVW	-	UR2TIF

UR2TIF: UR2T_DMA interrupt request flag bit. When the UR2T_DMA data transmission is completed, the hardware automatically sets UR2TIF to 1, and if the UR2T_DMA interrupt is enabled, the interrupt service routine is entered. The flag bit needs to be cleared by software.

TXOVW: UR2T_DMA data coverage flag. When UR2T_DMA is in the process of data transmission, and the UART writes the SBUF register to trigger the UART to send data again, the data transmission will fail. At this time, the hardware will automatically set TXOVW to 1. The flag bit needs to be cleared by software.

27.6.4 UR2T_DMA transfer total byte register (DMA_UR2T_AMT)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR2T_AMT	FA43H								

DMA_UR2T_AMT: Set the number of bytes of data that needs to be automatically sent.

Note: The actual number of bytes is (DMA_UR2T_AMT+1), that is, when DMA_UR2T_AMT is set to 0, 1

byte is transferred, and when DMA_UR2T_AMT is set to 255, 256 bytes are transferred.

27.6.5 UR2T_DMA transfer complete byte register (DMA_UR2T_DONE)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR2T_DONE	FA44H								

DMA_UR2T_DONE: The number of bytes that have been sent so far.

27.6.6 UR2T_DMA Send Address Registers (DMA_UR2T_TXAx)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR2T_TXAH	FA45H	ADDR[15:8]							
DMA_UR2T_TXAL	FA46H	ADDR[7:0]							

DMA_UR2T_TXA: Set the source address of automatic data transmission. Data is read from this address when performing a UR2T_DMA operation.

27.6.7 UR2R_DMA Configuration Register (DMA_UR2R_CFG)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR2R_CFG	FA48H	UR2RIE	-	-	-	UR2RIP[1:0]		UR2RPTY[1:0]	

UR2RIE: UR2R_DMA interrupt enable control bit

0: Disable UR2R_DMA interrupt

1: Enable UR2R_DMA interrupt

UR2RIP[1:0]: UR2R_DMA interrupt priority control bits

UR2RIP[1:0]	Interrupt priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

UR2RPTY[1:0]: UR2R_DMA Data bus access priority control bits

UR2RPTY [1:0]	Bus access priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

27.6.8 UR2R_DMA Control Register (DMA_UR2R_CR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR2R_CR	FA49H	ENUR2R	-	TRIG	-	-	-	-	CLRFIFO

ENUR2R: UR2R_DMA function enable control bit

0: Disable UR2R_DMA function

1: Enable UR2R_DMA function

TRIG: UR2R_DMA UART1 receive trigger control bit

0: Write 0 is invalid

1: Write 1 to start UR2R_DMA receiving data automatically

CLRFIFO: Clear UR2R_DMA receive FIFO control bit

0: Write 0 is invalid

1: Before starting the UR2R_DMA operation, clear the built-in FIFO of the UR2R_DMA firstly

27.6.9 UR2R_DMA Status Register (DMA_UR2R_STA)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0

DMA_UR2R_STA	FA4AH	-	-	-	-	-	-	RXLOSS	UR2RIF

UR2RIF: UR2R_DMA interrupt request flag bit. When UR2R_DMA receives data, the hardware will automatically set UR2RIF to 1. If the UR2R_DMA interrupt is enabled, it will enter the interrupt service routine. The flag bit needs to be cleared by software

RXLOSS: UR2R_DMA receive data discard flag. During the UR2R_DMA operation, when the XRAM bus is too busy to clear the receive FIFO of the UR2R_DMA and the data received by the UR2R_DMA is automatically discarded, the hardware will automatically set RXLOSS to 1. The flag bit needs to be cleared by software

27.6.10 UR2R_DMA transfer total byte register (DMA_UR2R_AMT)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR2R_AMT	FA4BH								

DMA_UR2R_AMT: Set the number of data bytes that need to automatically receive. **Note: The actual number of bytes is (DMA_UR2R_AMT+1), that is, when DMA_UR2R_AMT is set to 0, 1 byte is transferred, and when DMA_UR2R_AMT is set to 255, 256 bytes are transferred.**

27.6.11 UR2R_DMA transfer complete byte register (DMA_UR2R_DONE)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR2R_DONE	FA4CH								

DMA_UR2R_DONE: The number of bytes that have been received currently.

27.6.12 UR2R_DMA Receive Address Registers (DMA_UR2T_RXAx)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR2R_RXAH	FA4DH	ADDR[15:8]							
DMA_UR2R_RXAL	FA4EH	ADDR[7:0]							

DMA_UR2R_RXA: Set the target address for automatically receiving data. Data will be written from this address when performing a UR2R_DMA operation.

27.7 Data exchange between UART3 and memory (UR3T_DMA, UR3R_DMA)

27.7.1 UR3T_DMA Configuration Register (DMA_UR3T_CFG)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR3T_CFG	FA50H	UR3TIE	-	-	-	UR3TIP[1:0]		UR3TPTY[1:0]	

UR3TIE: UR3T_DMA interrupt enable control bit

0: Disable UR3T_DMA interrupt

1: Enable UR3T_DMA interrupt

UR3TIP[1:0]: UR3T_DMA interrupt priority control bits

UR3TIP[1:0]	Interrupt priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

UR3TPTY[1:0]: UR3T_DMA Data bus access priority control bits

UR3TPTY [1:0]	Bus access priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

27.7.2 UR3T_DMA Control Register (DMA_UR3T_CR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR3T_CR	FA51H	ENUR3T	TRIG	-	-	-	-	-	-

ENUR3T: UR3T_DMA function enable control bit

0: Disable UR3T_DMA function

1: Enable UR3T_DMA function

TRIG: UR3T_DMA UART1 transmit trigger control bit

0: Write 0 is invalid

1: Write 1 to start UR3T_DMA automatically sending data.

27.7.3 UR3T_DMA Status Register (DMA_UR3T_STA)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR3T_STA	FA52H	-	-	-	-	-	TXOVW	-	UR3TIF

UR3TIF: UR3T_DMA interrupt request flag bit. When the UR3T_DMA data transmission is completed, the hardware automatically sets UR3TIF to 1, and if the UR3T_DMA interrupt is enabled, the interrupt service routine is entered. The flag bit needs to be cleared by software.

TXOVW: UR3T_DMA data coverage flag. When UR3T_DMA is in the process of data transmission, and the UART writes the SBUF register to trigger the UART to send data again, the data transmission will fail. At this time, the hardware will automatically set TXOVW to 1. The flag bit needs to be cleared by software.

27.7.4 UR3T_DMA transfer total byte register (DMA_UR3T_AMT)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR3T_AMT	FA53H								

DMA_UR3T_AMT: Set the number of bytes of data that needs to be automatically sent.

Note: The actual number of bytes is (DMA_UR3T_AMT+1), that is, when DMA_UR3T_AMT is set to 0, 1

byte is transferred, and when DMA_UR3T_AMT is set to 255, 256 bytes are transferred.

27.7.5 UR3T_DMA transfer complete byte register (DMA_UR3T_DONE)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR3T_DONE	FA54H								

DMA_UR3T_DONE: The number of bytes that have been sent so far.

27.7.6 UR3T_DMA Send Address Registers (DMA_UR3T_TXAx)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR3T_TXAH	FA55H	ADDR[15:8]							
DMA_UR3T_TXAL	FA56H	ADDR[7:0]							

DMA_UR3T_TXA: Set the source address of automatic data transmission. Data is read from this address when performing a UR3T_DMA operation.

27.7.7 UR3R_DMA Configuration Register (DMA_UR3R_CFG)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR3R_CFG	FA58H	UR3RIE	-	-	-	UR3RIP[1:0]		UR3RPTY[1:0]	

UR3RIE: UR3R_DMA interrupt enable control bit

0: Disable UR3R_DMA interrupt

1: Enable UR3R_DMA interrupt

UR3RIP[1:0]: UR3R_DMA interrupt priority control bits

UR3RIP[1:0]	Interrupt priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

UR3RPTY[1:0]: UR3R_DMA Data bus access priority control bits

UR3RPTY [1:0]	Bus access priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

27.7.8 UR3R_DMA Control Register (DMA_UR3R_CR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR3R_CR	FA59H	ENUR3R	-	TRIG	-	-	-	-	CLRFIFO

ENUR3R: UR3R_DMA function enable control bit

0: Disable UR3R_DMA function

1: Enable UR3R_DMA function

TRIG: UR3R_DMA UART1 receive trigger control bit

0: Write 0 is invalid

1: Write 1 to start UR3R_DMA receiving data automatically

CLRFIFO: Clear UR3R_DMA receive FIFO control bit

0: Write 0 is invalid

1: Before starting the UR3R_DMA operation, clear the built-in FIFO of the UR3R_DMA firstly

27.7.9 UR3R_DMA Status Register (DMA_UR3R_STA)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR3R_STA	FA5AH	-	-	-	-	-	-	RXLOSS	UR3RIF

UR3RIF: UR3R_DMA interrupt request flag bit. When UR3R_DMA receives data, the hardware will automatically set UR3RIF to 1. If the UR3R_DMA interrupt is enabled, it will enter the interrupt service routine. The flag bit needs to be cleared by software

RXLOSS: UR3R_DMA receive data discard flag. During the UR3R_DMA operation, when the XRAM bus is too busy to clear the receive FIFO of the UR3R_DMA and the data received by the UR3R_DMA is automatically discarded, the hardware will automatically set RXLOSS to 1. The flag bit needs to be cleared by software

27.7.10 UR3R_DMA transfer total byte register (DMA_UR3R_AMT)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR3R_AMT	FA5BH								

DMA_UR3R_AMT: Set the number of data bytes that need to automatically receive. **Note: The actual number of bytes is (DMA_UR3R_AMT+1), that is, when DMA_UR3R_AMT is set to 0, 1 byte is transferred, and when DMA_UR3R_AMT is set to 255, 256 bytes are transferred.**

27.7.11 UR3R_DMA transfer complete byte register (DMA_UR3R_DONE)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR3R_DONE	FA5CH								

DMA_UR3R_DONE: The number of bytes that have been received currently.

27.7.12 UR3R_DMA Receive Address Registers (DMA_UR3T_RXAx)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR3R_RXAH	FA5DH	ADDR[15:8]							
DMA_UR3R_RXAL	FA5EH	ADDR[7:0]							

DMA_UR3R_RXA: Set the target address for automatically receiving data. Data will be written from this address when performing a UR3R_DMA operation.

27.8 Data exchange between UART4 and memory (UR4T_DMA, UR4R_DMA)

27.8.1 UR4T_DMA Configuration Register (DMA_UR4T_CFG)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR4T_CFG	FA50H	UR4TIE	-	-	-	UR4TIP[1:0]		UR4TPTY[1:0]	

UR4TIE: UR4T_DMA interrupt enable control bit

0: Disable UR4T_DMA interrupt

1: Enable UR4T_DMA interrupt

UR4TIP[1:0]: UR4T_DMA interrupt priority control bits

UR4TIP[1:0]	Interrupt priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

UR4TPTY[1:0]: UR4T_DMA Data bus access priority control bits

UR4TPTY [1:0]	Bus access priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

27.8.2 UR4T_DMA Control Register (DMA_UR4T_CR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR4T_CR	FA51H	ENUR4T	TRIG	-	-	-	-	-	-

ENUR4T: UR4T_DMA function enable control bit

0: Disable UR4T_DMA function

1: Enable UR4T_DMA function

TRIG: UR4T_DMA UART1 transmit trigger control bit

0: Write 0 is invalid

1: Write 1 to start UR4T_DMA automatically sending data.

27.8.3 UR4T_DMA Status Register (DMA_UR4T_STA)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR4T_STA	FA52H	-	-	-	-	-	TXOVW	-	UR4TIF

UR4TIF: UR4T_DMA interrupt request flag bit. When the UR4T_DMA data transmission is completed, the hardware automatically sets UR4TIF to 1, and if the UR4T_DMA interrupt is enabled, the interrupt service routine is entered. The flag bit needs to be cleared by software.

TXOVW: UR4T_DMA data coverage flag. When UR4T_DMA is in the process of data transmission, and the UART writes the SBUF register to trigger the UART to send data again, the data transmission will fail. At this time, the hardware will automatically set TXOVW to 1. The flag bit needs to be cleared by software.

27.8.4 UR4T_DMA transfer total byte register (DMA_UR4T_AMT)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR4T_AMT	FA53H								

DMA_UR4T_AMT: Set the number of bytes of data that needs to be automatically sent.

Note: The actual number of bytes is (DMA_UR4T_AMT+1), that is, when DMA_UR4T_AMT is set to 0, 1

byte is transferred, and when DMA_UR4T_AMT is set to 255, 256 bytes are transferred.

27.8.5 UR4T_DMA transfer complete byte register (DMA_UR4T_DONE)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR4T_DONE	FA54H								

DMA_UR4T_DONE: The number of bytes that have been sent so far.

27.8.6 UR4T_DMA Send Address Registers (DMA_UR4T_TXAx)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR4T_TXAH	FA55H	ADDR[15:8]							
DMA_UR4T_TXAL	FA56H	ADDR[7:0]							

DMA_UR4T_TXA: Set the source address of automatic data transmission. Data is read from this address when performing a UR4T_DMA operation.

27.8.7 UR4R_DMA Configuration Register (DMA_UR4R_CFG)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR4R_CFG	FA58H	UR4RIE	-	-	-	UR4RIP[1:0]		UR4RPTY[1:0]	

UR4RIE: UR4R_DMA interrupt enable control bit

0: Disable UR4R_DMA interrupt

1: Enable UR4R_DMA interrupt

UR4RIP[1:0]: UR4R_DMA interrupt priority control bits

UR4RIP[1:0]	Interrupt priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

UR4RPTY[1:0]: UR4R_DMA Data bus access priority control bits

UR4RPTY [1:0]	Bus access priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

27.8.8 UR4R_DMA Control Register (DMA_UR4R_CR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR4R_CR	FA59H	ENUR4R	-	TRIG	-	-	-	-	CLRFIFO

ENUR4R: UR4R_DMA function enable control bit

0: Disable UR4R_DMA function

1: Enable UR4R_DMA function

TRIG: UR4R_DMA UART1 receive trigger control bit

0: Write 0 is invalid

1: Write 1 to start UR4R_DMA receiving data automatically

CLRFIFO: Clear UR4R_DMA receive FIFO control bit

0: Write 0 is invalid

1: Before starting the UR4R_DMA operation, clear the built-in FIFO of the UR4R_DMA firstly

27.8.9 UR4R_DMA Status Register (DMA_UR4R_STA)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR4R_STA	FA5AH	-	-	-	-	-	-	RXLOSS	UR4RIF

UR4RIF: UR4R_DMA interrupt request flag bit. When UR4R_DMA receives data, the hardware will automatically set UR4RIF to 1. If the UR4R_DMA interrupt is enabled, it will enter the interrupt service routine. The flag bit needs to be cleared by software

RXLOSS: UR4R_DMA receive data discard flag. During the UR4R_DMA operation, when the XRAM bus is too busy to clear the receive FIFO of the UR4R_DMA and the data received by the UR4R_DMA is automatically discarded, the hardware will automatically set RXLOSS to 1. The flag bit needs to be cleared by software

27.8.10 UR4R_DMA transfer total byte register (DMA_UR4R_AMT)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR4R_AMT	FA5BH								

DMA_UR4R_AMT: Set the number of data bytes that need to automatically receive. **Note: The actual number of bytes is (DMA_UR4R_AMT+1), that is, when DMA_UR4R_AMT is set to 0, 1 byte is transferred, and when DMA_UR4R_AMT is set to 255, 256 bytes are transferred.**

27.8.11 UR4R_DMA transfer complete byte register (DMA_UR4R_DONE)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR4R_DONE	FA5CH								

DMA_UR4R_DONE: The number of bytes that have been received currently.

27.8.12 UR4R_DMA Receive Address Registers (DMA_UR4T_RXAx)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_UR4R_RXAH	FA5DH	ADDR[15:8]							
DMA_UR4R_RXAL	FA5EH	ADDR[7:0]							

DMA_UR4R_RXA: Set the target address for automatically receiving data. Data will be written from this address when performing a UR4R_DMA operation.

27.9 Data exchange between LCM and memory (LCM_DMA)

27.9.1 LCM_DMA Configuration Register (DMA_LCM_CFG)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_LCM_CFG	FA70H	LCMIE	ACT_TX	ACT_RX	-	LCMIP[1:0]		LCMPTY[1:0]	

LCMIE: LCM_DMA interrupt enable control bit

0: Disable LCM_DMA interrupt

1: Enable LCM_DMA interrupt

LCMIP [1:0]: LCM_DMA interrupt priority control bits

LCMIP[1:0]	Interrupt priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

LCMPTY[1:0]: LCM_DMA Data bus access priority control bits

LCMPTY [1:0]	Bus access priority
00	Lowest (0)
01	Lower (1)
10	Higher (2)
11	Highest (3)

27.9.2 LCM_DMA Control Register (DMA_LCM_CR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_LCM_CR	FA71H	ENLCM	TRIGWC	TRIGWD	TRIGRC	TRIGRD	-	-	CLRFIFO

ENLCM: LCM_DMA function enable control bit

0: Disable LCM_DMA function

1: Enable LCM_DMA function

TRIGWC: LCM_DMA send command mode trigger control bit

0: Write 0 is invalid

1: Write 1 to start LCM_DMA send command mode operation

TRIGWD: LCM_DMA send data mode trigger control bit

0: Write 0 is invalid

1: Write 1 to start LCM_DMA send data mode operation

TRIGRC: LCM_DMA read command mode trigger control bit

0: Write 0 is invalid

1: Write 1 to start LCM_DMA read command mode operation

TRIGRD: LCM_DMA read data mode trigger control bit

0: Write 0 is invalid

1: Write 1 to start LCM_DMA read data mode operation

27.9.3 LCM_DMA Status Register (DMA_LCM_STA)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_LCM_STA	FA72H	-	-	-	-	-	-	TXOVW	LCMIF

LCMIF: LCM_DMA interrupt request flag bit. After the LCM_DMA data exchange is completed, the hardware automatically sets LCMIF to 1. If the LCM_DMA interrupt is enabled, the interrupt service routine is entered. The flag bit needs to be cleared by software

TXOVW: LCM_DMA data coverage flag. When LCM_DMA is in the process of data transmission, and LCMIF writes the LCMIFDATL and LCMIDDATH registers, the data transmission will fail. At this time, the hardware will automatically set TXOVW to 1. The flag bit needs to be cleared by software

27.9.4 LCM_DMA transfer total byte register (DMA_LCM_AMT)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_LCM_AMT	FA73H								

DMA_LCM_AMT: Set the number of bytes that need to be read and written. **Note: The actual number of bytes read and written is (DMA_LCM_AMT+1), that is, when DMA_LCM_AMT is set to 0, 1 byte is transferred, and when DMA_LCM_AMT is set to 255, 256 bytes are transferred.**

27.9.5 LCM_DMA transfer complete byte register (DMA_LCM_DONE)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_LCM_DONE	FA74H								

DMA_LCM_DONE: The number of bytes that have been transferred so far.

27.9.6 LCM_DMA Send Address Registers (DMA_LCM_TXAx)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_LCM_TXAH	FA75H	ADDR[15:8]							
DMA_LCM_TXAL	FA76H	ADDR[7:0]							

DMA_LCM_TXA: Set the source address of automatic data transmission. Data is read from this address when performing an LCM_DMA operation.

27.9.7 LCM_DMA Receive Address Registers (DMA_LCM_RXAx)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_LCM_RXAH	FA77H	ADDR[15:8]							
DMA_LCM_RXAL	FA78H	ADDR[7:0]							

DMA_LCM_RXA: Set the target address for data transfer. Data is written from this address when performing an LCM_DMA operation.

27.10 Example Routines

27.10.1 UART1 interrupt mode and computer transceiver test - DMA receive timeout interrupt

C language code

```
//Operating frequency for test is 11.0592MHz

/************* Function Description **************
UART1 works in full-duplex interrupt mode to send and receive data. PC sends data to the MCU, and the MCU will
automatically store the received data in the DMA space. When the content received at one time is full of the set DMA
space, the data in the storage space are output through the DMA automatic sending function of UART 1. Use UART
receive interrupt to judge timeout, if no new data is received and timeout, it means that a string of data has been
received, then outputs the received content, and clear the DMA space. If a timer is uased as baud rate generator, it is
recommended to use 1T mode (unless 12T is used for low baud rates), and select a clock frequency that is divisible by
the baud rate to improve accuracy.
When downloading, select the clock 22.1184MHz (users can modify the frequency by themselves).
******************************************/

#include "stdio.h"
#include "stc8h.h"

#define     MAIN_Fosc           22118400L                   // Define the main clock (accurately calculate 115200 baud rate)
#define     Baudrate1           115200L
#define     Timer0_Reload       (65536UL -(MAIN_Fosc / 1000))

#define     DMA_AMT_LEN         255                         //Set total bytes to be transferred(0~255) : DMA_AMT_LEN+1

bit         B_1ms;                                          //1ms flag
bit         DMATxFlag;
bit         DMARxFlag;
bit         BusyFlag;
u8          Rx_cnt;
u8          RX1_TimeOut;

u8          xdata DMABuffer[256];

void UART1_config(u8 brt);
void DMA_Config(void);

void UartPutc(unsigned char dat)
{
        BusyFlag = 1;
        SBUF = dat;
        while(BusyFlag);
}

char putchar(char c)
{
        UartPutc(c);
        return c;
}
```

```
void main(void)
{
    u16 i;

    P0M1 = 0x00;   P0M0 = 0x00;                        //set as quasi-bidirectional port
    P1M1 = 0x00;   P1M0 = 0x00;                        //set as quasi-bidirectional port
    P2M1 = 0x00;   P2M0 = 0x00;                        //set as quasi-bidirectional port
    P3M1 = 0x00;   P3M0 = 0x00;                        //set as quasi-bidirectional port
    P4M1 = 0x00;   P4M0 = 0x00;                        //set as quasi-bidirectional port
    P5M1 = 0x00;   P5M0 = 0x00;                        //set as quasi-bidirectional port
    P6M1 = 0x00;   P6M0 = 0x00;                        //set as quasi-bidirectional port
    P7M1 = 0x00;   P7M0 = 0x00;                        //set as quasi-bidirectional port

    for(i=0; i<256; i++)
    {
        DMABuffer[i] = i;
    }

    AUXR = 0x80;                                       //Timer0 set as 1T, 16 bits timer auto-reload,
    TH0 = (u8)(Timer0_Reload / 256);
    TL0 = (u8)(Timer0_Reload % 256);
    ET0 = 1;                                           //Timer0 interrupt enable
    TR0 = 1;                                           //Tiner0 run

    UART1_config(1);                                   //Use Timer1 as baud rate generator.
    DMA_Config();
    EA = 1;                  //enable CPU interrupt

    printf("UART1 DMA Timeout Programme!\r\n");        //UART1 sends a string
    DMATxFlag = 0;
    DMARxFlag = 0;

    while (1)
    {
        if((DMATxFlag) && (DMARxFlag))                 //Judging the send completion flag and the reception completion
flag
        {
            Rx_cnt = 0;
            RX1_TimeOut = 0;
            printf("\r\nUART1 DMA FULL!\r\n");         //UART1 sends a string
            DMATxFlag = 0;
            DMA_UR1T_CR = 0xc0;                        //bit7 1:Enable UART1_DMA,
                                                       //bit6 1:Start UART1_DMA automatic transmission

            DMARxFlag = 0;
            DMA_UR1R_CR = 0xa1;                        //bit7 1:Enable UART1_DMA,
                                                       //bit5 1:Start UART1_DMA automatic reception,
                                                       //bit0 1:clear FIFO

        }

        if(B_1ms)                                      //reach 1ms
        {
            B_1ms = 0;
            if(RX1_TimeOut > 0)                        //timeout count
            {
                if(--RX1_TimeOut == 0)
                {
                    DMA_UR1R_CR = 0x00;                //Disable UART1_DMA
                    printf("\r\nUART1 Timeout!\r\n");   //UART1 sends a string

                    for(i=0;i<Rx_cnt;i++) UartPutc(DMABuffer[i]);
                    printf("\r\n");

                    Rx_cnt = 0;
                    DMA_UR1R_CR = 0xa1;                 //bit7 1:Enable UART1_DMA,
                                                        //bit5 1:Start UART1_DMA automatic reception,
                                                        //bit0 1:clear FIFO
                }
            }
        }
    }
}

void DMA_Config(void)
{
    P_SW2 = 0x80;
    DMA_UR1T_CFG = 0x80;                                //bit7 1:Enable Interrupt
    DMA_UR1T_STA = 0x00;
    DMA_UR1T_AMT = DMA_AMT_LEN;                         //Set total bytes to be transferred：n+1
    DMA_UR1T_TXA = DMABuffer;
    DMA_UR1T_CR = 0xc0;                                 //bit7 1:Enable UART1_DMA,
                                                        //bit6 1:Start UART1_DMA automatic transmission

    DMA_UR1R_CFG = 0x80;                                //bit7 1:Enable Interrupt
    DMA_UR1R_STA = 0x00;
    DMA_UR1R_AMT = DMA_AMT_LEN;                         //Set total bytes to be transferred：n+1
    DMA_UR1R_RXA = DMABuffer;
    DMA_UR1R_CR = 0xa1;                                 //bit7 1:Enable UART1_DMA,
                                                        //bit5 1:Start UART1_DMA automatic reception,
                                                        //bit0 1:clear FIFO
}

void SetTimer2Baudraye(u16 dat)
{
    AUXR &= ~(1<<4);                                    //Timer stop
    AUXR &= ~(1<<3);                                    //Timer2 set As Timer
    AUXR |=  (1<<2);                                    //Timer2 set as 1T mode
    T2H = dat / 256;
    T2L = dat % 256;
    IE2  &= ~(1<<2);                                    //Disable interrupts
    AUXR |=  (1<<4);                                    //Timer run enable
}

void UART1_config(u8 brt)                               //select baud rate:
                                                        //2: Use Timer2 as baud rate generator,
                                                        //Other values: Use Timer1 as baud rate generator.
{
    /***********  Use Timer2 as baud rate generator *****************/
    if(brt == 2)
    {
        AUXR |= 0x01;                                   //S1 BRT Use Timer2;
        SetTimer2Baudraye(65536UL - (MAIN_Fosc / 4) / Baudrate1);
    }

    /*********** Use Timer1 as baud rate generator *****************/
    else
    {
        TR1 = 0;
        AUXR &= ~0x01;                                      //S1 BRT Use Timer1;
        AUXR |=  (1<<6);                                    //Timer1 set as 1T mode
        TMOD &= ~(1<<6);                                    //Timer1 set As Timer
        TMOD &= ~0x30;                                      //Timer1_16bitAutoReload;
        TH1 = (u8)((65536UL - (MAIN_Fosc / 4) / Baudrate1) / 256);
        TL1 = (u8)((65536UL - (MAIN_Fosc / 4) / Baudrate1) % 256);
        ET1 = 0;                                            //Disable interrupts
        INTCLKO &= ~0x02;                                   //Does not output clock
        TR1  = 1;
    }
    /*************************************************/

    SCON = (SCON & 0x3f) | 0x40;                            //UART1 mode:
                                                            //0x00: Synchronous shift output,
                                                            //0x40: 8-bit data, variable baud rate,
                                                            //0x80: 9-bit data, fixed baud rate,
                                                            //0xc0: 9-bit data, variable baud rate
//  PS = 1;                                                 //high priority interrupt
    ES = 1;                                                 //enable interrupt
    REN = 1;                                                //enable to receive
    P_SW1 &= 0x3f;
    P_SW1 |= 0x00;                                          //UART1 switch to:
                                                            //0x00: P3.0 P3.1,
                                                            //0x40: P3.6 P3.7,
                                                            //0x80: P1.6 P1.7,
                                                            //0xC0: P4.3 P4.4

    RX1_TimeOut = 0;
}

void UART1_int (void) interrupt 4
{
    if(RI)
    {
        RI = 0;
        Rx_cnt++;
        if(Rx_cnt >= DMA_AMT_LEN) Rx_cnt = 0;
        RX1_TimeOut = 5;                                    //If no new data is received within 5ms, it is determined that a
string of data has been received.
    }

    if(TI)
    {
        TI = 0;
        BusyFlag = 0;
    }
}

void timer0 (void) interrupt 1
{
    B_1ms = 1;                                              //1ms flag
}

void UART1_DMA_Interrupt(void) interrupt 13
{
    if (DMA_UR1T_STA & 0x01)                                //send completed
    {
        DMA_UR1T_STA &= ~0x01;
        DMATxFlag = 1;
    }
    if (DMA_UR1T_STA & 0x04)                            //data coverage
    {
        DMA_UR1T_STA &= ~0x04;
    }

    if (DMA_UR1R_STA & 0x01)                            //Receive complete
    {
        DMA_UR1R_STA &= ~0x01;
        DMARxFlag = 1;
    }
    if (DMA_UR1R_STA & 0x02)                            //data is discarded
    {
        DMA_UR1R_STA &= ~0x02;
    }
}
```

```
//File: ISR.ASM
//Interrupts with interrupt numbers greater than 31 require interrupt entry address remapping processing
            CSEG    AT      012BH                       ;P0INT_VECTOR
            JMP             P0INT_ISR
            CSEG    AT      0133H                       ;P1INT_VECTOR
            JMP             P1INT_ISR
            CSEG    AT      013BH                       ;P2INT_VECTOR
            JMP             P2INT_ISR
            CSEG    AT      0143H                       ;P3INT_VECTOR
            JMP             P3INT_ISR
            CSEG    AT      014BH                       ;P4INT_VECTOR
            JMP             P4INT_ISR
            CSEG    AT      0153H                       ;P5INT_VECTOR
            JMP             P5INT_ISR
            CSEG    AT      015BH                       ;P6INT_VECTOR
            JMP             P6INT_ISR
            CSEG    AT      0163H                       ;P7INT_VECTOR
            JMP             P7INT_ISR
            CSEG    AT      016BH                       ;P8INT_VECTOR
            JMP             P8INT_ISR
            CSEG    AT      0173H                       ;P9INT_VECTOR
            JMP             P9INT_ISR
            CSEG    AT      017BH                       ;M2MDMA_VECTOR
            JMP             M2MDMA_ISR
            CSEG    AT      0183H                       ;ADCDMA_VECTOR
            JMP             ADCDMA_ISR
            CSEG    AT      018BH                       ;SPIDMA_VECTOR
            JMP             SPIDMA_ISR
            CSEG    AT      0193H                       ;U1TXDMA_VECTOR
            JMP             U1TXDMA_ISR
            CSEG    AT      019BH                       ;U1RXDMA_VECTOR
            JMP             U1RXDMA_ISR
            CSEG    AT      01A3H                       ;U2TXDMA_VECTOR
            JMP             U2TXDMA_ISR
            CSEG    AT      01ABH                       ;U2RXDMA_VECTOR
            JMP             U2RXDMA_ISR
            CSEG    AT      01B3H                       ;U3TXDMA_VECTOR
```

```
                    JMP             U3TXDMA_ISR
                    CSEG    AT      01BBH                   ;U3RXDMA_VECTOR
                    JMP             U3RXDMA_ISR
                    CSEG    AT      01C3H                   ;U4TXDMA_VECTOR
                    JMP             U4TXDMA_ISR
                    CSEG    AT      01CBH                   ;U4RXDMA_VECTOR
                    JMP             U4RXDMA_ISR
                    CSEG    AT      01D3H                   ;LCMDMA_VECTOR
                    JMP             LCMDMA_ISR
                    CSEG    AT      01DBH                   ;LCMIF_VECTOR
                    JMP             LCMIF_ISR

P0INT_ISR:
P1INT_ISR:
P2INT_ISR:
P3INT_ISR:
P4INT_ISR:
P5INT_ISR:
P6INT_ISR:
P7INT_ISR:
P8INT_ISR:
P9INT_ISR:
M2MDMA_ISR:
ADCDMA_ISR:
SPIDMA_ISR:
U1TXDMA_ISR:
U1RXDMA_ISR:
U2TXDMA_ISR:
U2RXDMA_ISR:
U3TXDMA_ISR:
U3RXDMA_ISR:
U4TXDMA_ISR:
U4RXDMA_ISR:
LCMDMA_ISR:
LCMIF_ISR:

                    JMP             006BH

                    END
```

27.10.2 UART1 interrupt mode and computer transceiver test - DMA data check

C language code

```
// Operating frequency for test is 11.0592MHz

/*************  Function Description    **************
UART1 works in full-duplex interrupt mode to send and receive data. PC sends data to the MCU, and the MCU will
automatically store the received data in the DMA space. The last two bytes of the data packet are used as the check
digit, and the routine performs the check using the crc16_ccitt algorithm. When the DMA space is full of the content
of the set size, the valid data is checked and calculated, and then compared with the last two check digits. The data in
the storage space is output through the DMA automatic sending function of UART1. If a timer is uased as baud rate
generator, it is recommended to use 1T mode (unless 12T is used for low baud rates), and select a clock frequency
```

```
that is divisible by the baud rate to improve accuracy.
When downloading, select the clock 22.1184MHz (users can modify the frequency by themselves).
******************************************/

#include "stdio.h"
#include "stc8h.h"
#include "crc16.h"

#define     MAIN_Fosc          22118400L                // Define the main clock (accurately calculate 115200 baud rate)
#define     Baudrate1          115200L

#define     DMA_AMT_LEN        255                      ///Set total bytes to be transferred(0~255)：DMA_AMT_LEN+1

bit         DMATxFlag;
bit         DMARxFlag;

u8          xdata DMABuffer[256];

void UART1_config(u8 brt);
void DMA_Config(void);

void UartPutc(unsigned char dat)
{
        SBUF = dat;
        while(TI == 0);
        TI = 0;
}

char putchar(char c)
{
        UartPutc(c);
        return c;
}

/*****CRC calculation function********/
u16 crc16_ccitt(u8 *pbuf, u16 len)
{
        unsigned short code crc16_ccitt_table[256] =
        {
                0x0000,  0x1021,  0x2042,  0x3063,  0x4084,  0x50A5,  0x60C6,  0x70E7,
                0x8108,  0x9129,  0xA14A,  0xB16B,  0xC18C,  0xD1AD,  0xE1CE,  0xF1EF,
                0x1231,  0x0210,  0x3273,  0x2252,  0x52B5,  0x4294,  0x72F7,  0x62D6,
                0x9339,  0x8318,  0xB37B,  0xA35A,  0xD3BD,  0xC39C,  0xF3FF,  0xE3DE,
                0x2462,  0x3443,  0x0420,  0x1401,  0x64E6,  0x74C7,  0x44A4,  0x5485,
                0xA56A,  0xB54B,  0x8528,  0x9509,  0xE5EE,  0xF5CF,  0xC5AC,  0xD58D,
                0x3653,  0x2672,  0x1611,  0x0630,  0x76D7,  0x66F6,  0x5695,  0x46B4,
                0xB75B,  0xA77A,  0x9719,  0x8738,  0xF7DF,  0xE7FE,  0xD79D,  0xC7BC,
                0x48C4,  0x58E5,  0x6886,  0x78A7,  0x0840,  0x1861,  0x2802,  0x3823,
                0xC9CC,  0xD9ED,  0xE98E,  0xF9AF,  0x8948,  0x9969,  0xA90A,  0xB92B,
                0x5AF5,  0x4AD4,  0x7AB7,  0x6A96,  0x1A71,  0x0A50,  0x3A33,  0x2A12,
                0xDBFD,  0xCBDC,  0xFBBF,  0xEB9E,  0x9B79,  0x8B58,  0xBB3B,  0xAB1A,
                0x6CA6,  0x7C87,  0x4CE4,  0x5CC5,  0x2C22,  0x3C03,  0x0C60,  0x1C41,
                0xEDAE,  0xFD8F,  0xCDEC,  0xDDCD,  0xAD2A,  0xBD0B,  0x8D68,  0x9D49,
                0x7E97,  0x6EB6,  0x5ED5,  0x4EF4,  0x3E13,  0x2E32,  0x1E51,  0x0E70,
                0xFF9F,  0xEFBE,  0xDFDD,  0xCFFC,  0xBF1B,  0xAF3A,  0x9F59,  0x8F78,
                0x9188,  0x81A9,  0xB1CA,  0xA1EB,  0xD10C,  0xC12D,  0xF14E,  0xE16F,
                0x1080,  0x00A1,  0x30C2,  0x20E3,  0x5004,  0x4025,  0x7046,  0x6067,
                0x83B9,  0x9398,  0xA3FB,  0xB3DA,  0xC33D,  0xD31C,  0xE37F,  0xF35E,
                0x02B1,  0x1290,  0x22F3,  0x32D2,  0x4235,  0x5214,  0x6277,  0x7256,
                0xB5EA,  0xA5CB,  0x95A8,  0x8589,  0xF56E,  0xE54F,  0xD52C,  0xC50D,
                0x34E2,  0x24C3,  0x14A0,  0x0481,  0x7466,  0x6447,  0x5424,  0x4405,
                0xA7DB,  0xB7FA,  0x8799,  0x97B8,  0xE75F,  0xF77E,  0xC71D,  0xD73C,
                0x26D3,  0x36F2,  0x0691,  0x16B0,  0x6657,  0x7676,  0x4615,  0x5634,
                0xD94C,  0xC96D,  0xF90E,  0xE92F,  0x99C8,  0x89E9,  0xB98A,  0xA9AB,
                0x5844,  0x4865,  0x7806,  0x6827,  0x18C0,  0x08E1,  0x3882,  0x28A3,
                0xCB7D,  0xDB5C,  0xEB3F,  0xFB1E,  0x8BF9,  0x9BD8,  0xABBB,  0xBB9A,
                0x4A75,  0x5A54,  0x6A37,  0x7A16,  0x0AF1,  0x1AD0,  0x2AB3,  0x3A92,
                0xFD2E,  0xED0F,  0xDD6C,  0xCD4D,  0xBDAA,  0xAD8B,  0x9DE8,  0x8DC9,
                0x7C26,  0x6C07,  0x5C64,  0x4C45,  0x3CA2,  0x2C83,  0x1CE0,  0x0CC1,
                0xEF1F,  0xFF3E,  0xCF5D,  0xDF7C,  0xAF9B,  0xBFBA,  0x8FD9,  0x9FF8,
                0x6E17,  0x7E36,  0x4E55,  0x5E74,  0x2E93,  0x3EB2,  0x0ED1,  0x1EF0
        };

        u16 crc16 = 0x0000;
        u16 crc_h8, crc_l8;

        while( len-- ) {
                crc_h8 = (crc16 >> 8);
                crc_l8 = (crc16 << 8);
                crc16 = crc_l8 ^ crc16_ccitt_table[crc_h8 ^ *pbuf];
                pbuf++;
        }

        return crc16;
}

void main(void)
{
        u16 i;
        u16 CheckSum;

        P0M1 = 0x00;   P0M0 = 0x00;                          //set as quasi-bidirectional port
        P1M1 = 0x00;   P1M0 = 0x00;                          //set as quasi-bidirectional port
        P2M1 = 0x00;   P2M0 = 0x00;                          //set as quasi-bidirectional port
        P3M1 = 0x00;   P3M0 = 0x00;                          //set as quasi-bidirectional port
        P4M1 = 0x00;   P4M0 = 0x00;                          //set as quasi-bidirectional port
        P5M1 = 0x00;   P5M0 = 0x00;                          //set as quasi-bidirectional port
        P6M1 = 0x00;   P6M0 = 0x00;                          //set as quasi-bidirectional port
        P7M1 = 0x00;   P7M0 = 0x00;                          //set as quasi-bidirectional port

        for(i=0; i<256; i++)
        {
                DMABuffer[i] = i;
        }

        P_SW2 = 0x80;
        DMA_UR1T_STA = 0x00;
        UART1_config(1);
        printf("UART1 DMA CRC Programme!\r\n");

        DMA_Config();
        EA = 1;                                              //enable CPU interrupt

        DMATxFlag = 0;
        DMARxFlag = 0;
    while (1)
    {
        if((DMATxFlag) && (DMARxFlag))
        {
            CheckSum = crc16_ccitt(DMABuffer,DMA_AMT_LEN-1);
            if(((u8)CheckSum == DMABuffer[DMA_AMT_LEN-1]) &&
            ((u8)(CheckSum>>8) == DMABuffer[DMA_AMT_LEN]))
            {
                printf("\r\nOK! CheckSum = %04x\r\n",CheckSum);
            }
            else
            {
                printf("\r\nERROR! CheckSum = %04x\r\n",CheckSum);
            }
            DMATxFlag = 0;
            DMA_UR1T_CR = 0xc0;                     //bit7 1:Enable UART1_DMA,
                                                    //bit6 1:Start UART1_DMA automatic transmission
            DMARxFlag = 0;
            DMA_UR1R_CR = 0xa1;                     //bit7 1:Enable UART1_DMA,
                                                    //bit5 1:Start UART1_DMA automatic reception,
                                                    //bit0 1:clear FIFO
        }
    }
}

void DMA_Config(void)
{
    P_SW2 = 0x80;
    DMA_UR1T_CFG = 0x80;                            //bit7 1:Enable Interrupt
    DMA_UR1T_STA = 0x00;
    DMA_UR1T_AMT = DMA_AMT_LEN;                     // Set total bytes to be transferred：n+1
    DMA_UR1T_TXA = DMABuffer;
    DMA_UR1T_CR = 0xc0;                             //bit7 1:Enable UART1_DMA,
                                                    //bit6 1:Start UART1_DMA automatic transmission

    DMA_UR1R_CFG = 0x80;                            //bit7 1:Enable Interrupt
    DMA_UR1R_STA = 0x00;
    DMA_UR1R_AMT = DMA_AMT_LEN;                     // Set total bytes to be transferred：n+1
    DMA_UR1R_RXA = DMABuffer;
    DMA_UR1R_CR = 0xa1;                             //bit7 1:Enable UART1_DMA,
                                                    //bit5  1:Start  UART1_DMA  automatic  reception,  bit0  1:clear
FIFO
}

void SetTimer2Baudraye(u16 dat)                     //select baud rate:
                                                    //2: Use Timer2 as baud rate generator,
                                                    //Other values: Use Timer1 as baud rate generator.
{
    AUXR &= ~(1<<4);                                //Timer stop
    AUXR &= ~(1<<3);                                //Timer2 set As Timer
    AUXR |=  (1<<2);                                //Timer2 set as 1T mode
    T2H = dat / 256;
    T2L = dat % 256;
    IE2  &= ~(1<<2);                                //Disable interrupts
    AUXR |=  (1<<4);                                //Timer run enable
}

void UART1_config(u8 brt)                           //select baud rate:
```

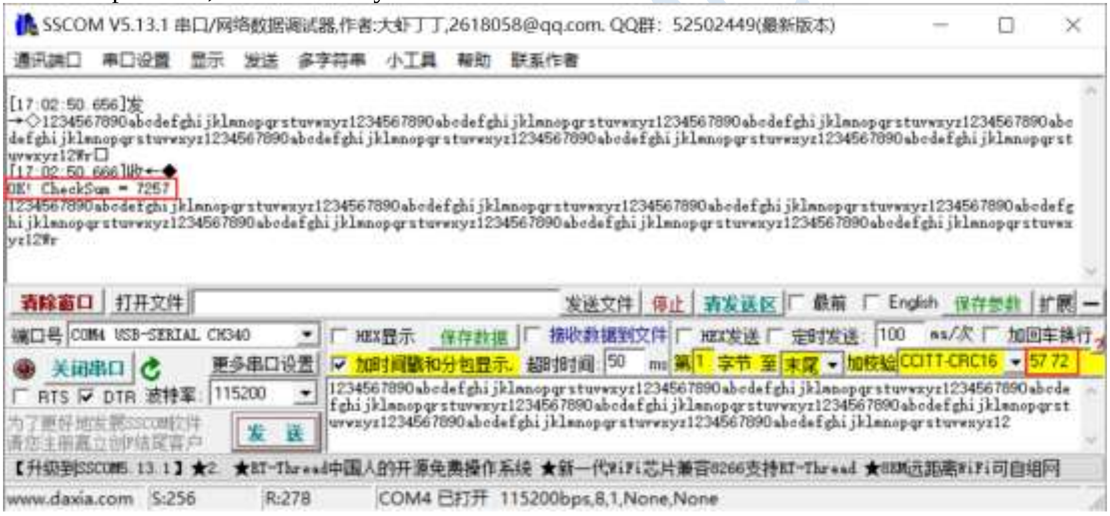
```
                                                //2: Use Timer2 as baud rate generator
                                                //Other values: Use Timer1 as baud rate generator.
{
    /*********** Use Timer2 as baud rate generator *****************/
    if(brt == 2)
    {
        AUXR |= 0x01;                           //S1 BRT Use Timer2;
        SetTimer2Baudraye(65536UL - (MAIN_Fosc / 4) / Baudrate1);
    }

    /*********** Use Timer1 as baud rate generator *****************/
    else
    {
        TR1 = 0;
        AUXR &= ~0x01;                          //S1 BRT Use Timer1;
        AUXR |=  (1<<6);                        //Timer1 set as 1T mode
        TMOD &= ~(1<<6);                        //Timer1 set As Timer
        TMOD &= ~0x30;                          //Timer1_16bitAutoReload;
        TH1 = (u8)((65536UL - (MAIN_Fosc / 4) / Baudrate1) / 256);
        TL1 = (u8)((65536UL - (MAIN_Fosc / 4) / Baudrate1) % 256);
        ET1 = 0;                                //Disable interrupts
        INTCLKO &= ~0x02;                       //Does not output clock
        TR1  = 1;
    }
    /*************************************************/

    SCON = (SCON & 0x3f) | 0x40;                //UART1 mode,
                                                //0x00: Synchronous shift output,
                                                //0x40: 8-bit data, variable baud rate,
                                                //0x80: 9-bit data, fixed baud rate,
                                                //0xc0: 9-bit data, variable baud rate
//  PS  = 1;                                    //high priority interrupt
//  ES  = 1;                                    //enable interrupt
    REN = 1;                                    //enable to receive
    P_SW1 &= 0x3f;
    P_SW1 |= 0x00;
}

void UART1_DMA_Interrupt(void) interrupt 13
{
    if (DMA_UR1T_STA & 0x01)                    //send completed
    {
        DMA_UR1T_STA &= ~0x01;
        DMATxFlag = 1;
    }
    if (DMA_UR1T_STA & 0x04)                    //data coverage
    {
        DMA_UR1T_STA &= ~0x04;
    }

    if (DMA_UR1R_STA & 0x01)                    //Receive complete
    {
        DMA_UR1R_STA &= ~0x01;
        DMARxFlag = 1;
    }
    if (DMA_UR1R_STA & 0x02)                    //data is discarded
    {
        DMA_UR1R_STA &= ~0x02;
        }
}
```

```
//File: ISR.ASM
//Interrupts with interrupt numbers greater than 31 require interrupt entry address remapping processing
                CSEG    AT      012BH                   ;P0INT_VECTOR
                JMP             P0INT_ISR
                CSEG    AT      0133H                   ;P1INT_VECTOR
                JMP             P1INT_ISR
                CSEG    AT      013BH                   ;P2INT_VECTOR
                JMP             P2INT_ISR
                CSEG    AT      0143H                   ;P3INT_VECTOR
                JMP             P3INT_ISR
                CSEG    AT      014BH                   ;P4INT_VECTOR
                JMP             P4INT_ISR
                CSEG    AT      0153H                   ;P5INT_VECTOR
                JMP             P5INT_ISR
                CSEG    AT      015BH                   ;P6INT_VECTOR
                JMP             P6INT_ISR
                CSEG    AT      0163H                   ;P7INT_VECTOR
                JMP             P7INT_ISR
                CSEG    AT      016BH                   ;P8INT_VECTOR
                JMP             P8INT_ISR
                CSEG    AT      0173H                   ;P9INT_VECTOR
                JMP             P9INT_ISR
                CSEG    AT      017BH                   ;M2MDMA_VECTOR
                JMP             M2MDMA_ISR
                CSEG    AT      0183H                   ;ADCDMA_VECTOR
                JMP             ADCDMA_ISR
                CSEG    AT      018BH                   ;SPIDMA_VECTOR
                JMP             SPIDMA_ISR
                CSEG    AT      0193H                   ;U1TXDMA_VECTOR
                JMP             U1TXDMA_ISR
                CSEG    AT      019BH                   ;U1RXDMA_VECTOR
                JMP             U1RXDMA_ISR
                CSEG    AT      01A3H                   ;U2TXDMA_VECTOR
                JMP             U2TXDMA_ISR
                CSEG    AT      01ABH                   ;U2RXDMA_VECTOR
                JMP             U2RXDMA_ISR
                CSEG    AT      01B3H                   ;U3TXDMA_VECTOR
                JMP             U3TXDMA_ISR
                CSEG    AT      01BBH                   ;U3RXDMA_VECTOR
                JMP             U3RXDMA_ISR
                CSEG    AT      01C3H                   ;U4TXDMA_VECTOR
                JMP             U4TXDMA_ISR
                CSEG    AT      01CBH                   ;U4RXDMA_VECTOR
                JMP             U4RXDMA_ISR
                CSEG    AT      01D3H                   ;LCMDMA_VECTOR
                JMP             LCMDMA_ISR
                CSEG    AT      01DBH                   ;LCMIF_VECTOR
                JMP             LCMIF_ISR

P0INT_ISR:
P1INT_ISR:
P2INT_ISR:
P3INT_ISR:
P4INT_ISR:
```

```
P5INT_ISR:
P6INT_ISR:
P7INT_ISR:
P8INT_ISR:
P9INT_ISR:
M2MDMA_ISR:
ADCDMA_ISR:
SPIDMA_ISR:
U1TXDMA_ISR:
U1RXDMA_ISR:
U2TXDMA_ISR:
U2RXDMA_ISR:
U3TXDMA_ISR:
U3RXDMA_ISR:
U4TXDMA_ISR:
U4RXDMA_ISR:
LCMDMA_ISR:
LCMIF_ISR:

                JMP         006BH

                END
```

Code testing method

According to the predefined DMA packet length (for example: 256 bytes), send a packet of data (254 bytes) through the serial port tool, and add a 2-byte CCITT-CRC16 check code at the end:


SSCOM V5.13.1 串口/网络数据调试器,作者:大虾丁丁,2618058@qq.com. QQ群： 52502449(最新版本)
通讯端口
串口设置
显示
发送
多字符串
小工具
帮助
联系作者
清除窗口
打开文件
发送文件
停止
清发送区
最前
English
保存参数
扩展
保存数据
接收数据到文件
关闭串口
更多串口设置
发 送
为了更好地发展SSCOM软件
请您注册嘉立创P结尾客户
【升级到SSCOM5.13.1】★2. ★RT-Thread中国人的开源免费操作系统 ★新一代WiFi芯片兼容8266支持RT-Thread ★8KM远距离WiFi可自组网

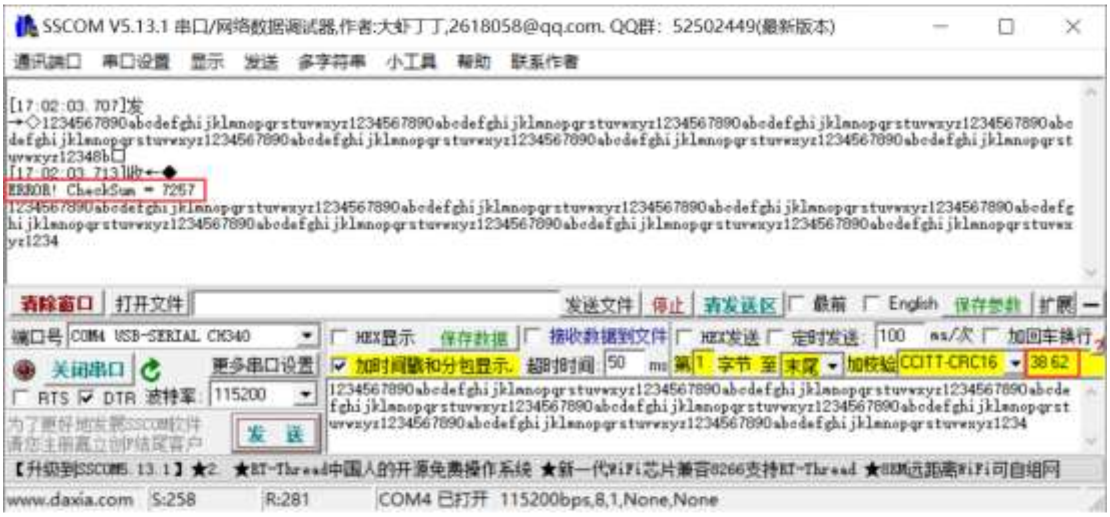
```
[17:02:50.656]发→◇1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567890abc
defghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrst
uvwxyz12Wr□
[17:02:50.666]收←◆
OK! CheckSum = 7257
1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefg
hijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwx
yz12Wr
```

端口号 COM4 USB-SERIAL CH340 ☐ HEX显示 ☐ HEX发送 ☐ 定时发送 100 ms/次 ☐ 加回车换行
☑ 加时间戳和分包显示. 超时时间 50 ms 第 1 字节 至 末尾 加校验 CCITT-CRC16 57 72
☐ RTS ☑ DTR 波特率 115200

```
1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567890abcde
fghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrst
uvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz12
```

www.daxia.com S:256 R:278 COM4 已打开 115200bps,8,1,None,None

After the MCU receives the entire packet of data (256 bytes), it performs CRC16 check on the first 254 bytes of data, and the obtained check code is compared with the last two bytes. If the values are equal, print "OK!" and calculate the check code, and then output the content read from the DMA space.


SSCOM V5.13.1 串口/网络数据调试器,作者:大虾丁丁,2618058@qq.com. QQ群: 52502449(最新版本)
通讯端口 串口设置 显示 发送 多字符串 小工具 帮助 联系作者
清除窗口 打开文件 发送文件 停止 清发送区 ☐ 最前 ☐ English 保存参数 扩展 —
☐ HEX显示 保存数据 ☐ 接收数据到文件 ☐ HEX发送 ☐ 定时发送: 100 ms/次 ☐ 加回车换行
关闭串口 更多串口设置
为了更好地发展SSCOM软件
请您注册嘉立创P结尾客户
发 送
【升级到SSCOM5.13.1】★2. ★RT-Thread中国人的开源免费操作系统 ★新一代WiFi芯片兼容8266支持RT-Thread ★8KM远距离WiFi可自组网

```
[17:02:03.707]发→◇1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567890abc
defghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrst
uvwxyz12348b□
[17:02:03.713]收←◆
ERROR! CheckSum = 7257
1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefg
hijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwx
yz1234
```

端口号 COM4 USB-SERIAL CH340

☑ 加时间戳和分包显示. 超时时间: 50 ms 第 1 字节 至 末尾 加校验 CCITT-CRC16 38 62

☐ RTS ☑ DTR 波特率: 115200

```
1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567890abcde
fghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrst
uvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrstuvwxyz1234
```

www.daxia.com S:258 R:281 COM4 已打开 115200bps,8,1,None,None

If the checksum values are not equal, print "ERROR!" and the calculated checksum.

28 Enhanced Dual Data Pointer

Two 16-bit data pointers are integrated in STC8H series of microcontrollers. The data pointers can be increased or decreased automatically by the program control, and they can be switched automatically.

28.1 Related special function registers

Symbol	Description	Address	Bit Address and Symbol								Reset Value
			B7	B6	B5	B4	B3	B2	B1	B0	
DPL	Data pointer low byte register	82H									0000,0000
DPH	Data pointer high byte register	83H									0000,0000
DPL1	2nd Data pointer low byte	E4H									0000,0000
DPH1	2nd Data pointer high byte	E5H									0000,0000
DPS	DPTR Selection Register	E3H	ID1	ID0	TSL	AU1	AU0	-	-	SEL	0000,0xx0
TA	DPTR Timing control register	AEH									0000,0000

28.1.1 1st 16-bit Data Pointer Registers (DPTR0)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DPL	82H								
DPH	83H								

DPL is Data pointer 0 low byte.

DPH is Data pointer 0 high byte.

The combination of DPL and DPH is the first 16-bit data pointer register DPTR0.

28.1.2 2nd 16-bit Data Pointer Registers (DPTR1)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DPL1	E4H								
DPH1	E5H								

DPL1 is Data pointer 1 low byte.

DPH1 is Data pointer 1 low byte.

The combination of DPL1 and DPH1 is the second 16-bit data pointer register DPTR1.

28.1.3 DPTR control register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DPS	E3H	ID1	ID0	TSL	AU1	AU0	-	-	SEL

ID1: DPTR1 auto-increment or auto-decrement mode control bit
    0: DPTR1 auto-increment mode
    1: DPTR1 auto-decrement mode
ID0: DPTR0 auto-increment or auto-decrement mode control bit
    0: DPTR0 auto-increment mode
    1: DPTR0 auto-decrement mode
TSL: DPTR0/DPTR1 auto-switch control bit (invert SEL automatically)
    0: DPTR0/DPTR1 auto switch is disabled.
    1: DPTR0/DPTR1 auto switch is enabled.

If the TSL bit is set, the SEL bit will be inverted automatically after the relevant instruction is executed.

Instructions related to TSL include:

MOV DPTR,#data16

```
INC      DPTR
MOVC     A,@A+DPTR
MOVX     A,@DPTR
MOVX     @DPTR,A
```

AU1/AU0: Enable DPTR1 / DPTR0 Automatic increment / decrement control bit

0: disable Automatic increment / decrement function

1: enable Automatic increment / decrement function

Note: In write-protect mode, AU0 and AU1 can not be enabled individually. AU0 will be enabled automatically if AU1 is enabled. If AU0 is enabled alone, there is no effect to AU1. To enable AU1 or AU0 independently, the TA register must be used to trigger the DPS protection mechanism. For more detail, please refer to the description of the TA register. In addition, DPTR0 / DPTR1 will be incremented / decremented automatically only after executing the following three instructions.

```
MOVC     A,@A+DPTR
MOVX     A,@DPTR
MOVX     @DPTR,A
```

SEL: DPTR register select bit.

0: Default. DPTR0 is selected as current Data pointer.

1: DPTR1 is selected as current Data pointer.

The selection of current DPTR using SEL is valid for the following instructions,

```
MOV      DPTR,#data16
INC      DPTR
MOVC     A,@A+DPTR
MOVX     A,@DPTR
MOVX     @DPTR,A
JMP      @A+DPTR
```

28.1.4 Data Pointer control register (TA)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
TA	AEH								

AU1 and AU0 in the DPS register is write-protected by TA register. Since the program can not write AU1 and AU0 separately, TA register must be used to trigger enabling AU1 or AU0 independently. TA is a write-only register.

The following steps must be executed if you need to enable AU1 or AU0 separately.

```
CLR      EA              ; disable interrupt (it is necessary to disable interrupt)
MOV      TA,#0AAH        ; write the trigger command sequence 1
                         ; any other instructions can not be here
MOV      TA,#55H         ; write the trigger command sequence 2
                         ; any other instructions can not be here
MOV      DPS,#xxH        ; Write-protection is temporarily disabled,
                         ; and any value can be written to the DPS
                         ; DSP enters the write-protected mode again
SETB     EA              ; enable interrupt if necessary
```

28.2 Example Routines

28.2.1 Example Routine 1

Copy 4 bytes of data stored in program space 1000H to 1003H in reverse to 0100H to 0103H of the extended RAM, that is,

```
C:1000H -> X:0103H
C:1001H -> X:0102H
C:1002H -> X:0101H
C:1003H -> X:0100H
```

Assembly code

```
;Operating frequency for test is 11.0592MHz

P0M1            DATA        093H
P0M0            DATA        094H
P1M1            DATA        091H
P1M0            DATA        092H
P2M1            DATA        095H
P2M0            DATA        096H
P3M1            DATA        0B1H
P3M0            DATA        0B2H
P4M1            DATA        0B3H
P4M0            DATA        0B4H
P5M1            DATA        0C9H
P5M0            DATA        0CAH

                ORG         0000H
                LJMP        MAIN

                ORG         0100H
MAIN:
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                MOV         DPS,#00100000B          ; enable TSL and select DPTR0
                MOV         DPTR,#1000H      ; write 1000H to DPTR0, and then select DPTR1 as current DPTR
                MOV         DPTR,#0103H             ; write 0103H to DPTR1
                MOV         DPS,#10111000B          ;set DPTR1 in auto decreasing mode,
                                                  ;DPTR0 in auto increasing mode,enable TSL, AU0 and AU1,
                                                    ;select DPTR0 as the current DPTR
                MOV         R7,#4                   ; set the counter of copies
COPY_NEXT:
                CLR         A                       ;
                MOVC        A,@A+DPTR               ; Read data from the program space indicated by DPTR0,
                            ; When done, DPTR0 increments automatically and select DPTR1 as the current DPTR
                MOVX        @DPTR,A                 ; write the content of ACC to XDARA indicated by DPTR1,
                              ; When done, DPTR1 decrements automatically and select DPTR0 as the current DPTR
                DJNZ        R7,COPY_NEXT            ;

                SJMP        $

                END
```

28.2.2 Example Routine 2

Send the data stored in the extended RAM 0100H to 0103H to P0 port successively.

Assembly code

```
;Operating frequency for test is 11.0592MHz

P0M1            DATA        093H
P0M0            DATA        094H
P1M1            DATA        091H
P1M0            DATA        092H
P2M1            DATA        095H
P2M0            DATA        096H
P3M1            DATA        0B1H
P3M0            DATA        0B2H
P4M1            DATA        0B3H
P4M0            DATA        0B4H
P5M1            DATA        0C9H
P5M0            DATA        0CAH

                ORG         0000H
                LJMP        MAIN

                ORG         0100H
MAIN:
                MOV         SP, #5FH
                MOV         P0M0, #00H
                MOV         P0M1, #00H
                MOV         P1M0, #00H
                MOV         P1M1, #00H
                MOV         P2M0, #00H
                MOV         P2M1, #00H
                MOV         P3M0, #00H
                MOV         P3M1, #00H
                MOV         P4M0, #00H
                MOV         P4M1, #00H
                MOV         P5M0, #00H
                MOV         P5M1, #00H

                CLR         EA                      ; disable interrupt
                MOV         TA,#0AAH                ; Write DPS write-protection trigger command 1
                MOV         TA,#55H                 ; Write DPS write-protection trigger command 2
                MOV         DPS,#00001000B          ; set DPTR0 in increasing mode,enable AU0 independently,
                                                    ;and select DPTR0

                SETB        EA                      ; enable interrupt
                MOV         DPTR,#0100H             ; write 0100H to DPTR0
                MOVX        A,@DPTR                 ; Read data from XRAM indicated by DPTR0,
                                                    ;and then DPTR0 increments automatically
                MOV         P0,A                    ; output the datum to Port0
```

```
        MOVX        A,@DPTR                 ; Read data from XRAM indicated by DPTR0,
                                            ;and then DPTR0 increments automatically
        MOV         P0,A                    ; output the datum to Port0
        MOVX        A,@DPTR                 ; Read data from XRAM indicated by DPTR0,
                                            ;and then DPTR0 increments automatically
        MOV         P0,A                    ; output the datum to Port0
        MOVX        A,@DPTR                 ; Read data from XRAM indicated by DPTR0,
                                            ;and then DPTR0 increments automatically
        MOV         P0,A                    ; output the datum to Port0

        SJMP        $

        END
```

29 MDU16 Hardware 16-bit Multiplier and Divider

Product line	MDU16
STC8H1K08 family	
STC8H1K28 family	
STC8H3K64S4 family	●
STC8H3K64S2 family	●
STC8H8K64U family	●
STC8H2K64T family	●
STC8H4K64TLR family	●
STC8H4K64TLCD family	●
STC8H4K64LCD family	●

A 16-bit hardware multiply / divide unit MDU16 is integrated in some microcontrollers of the STC8H series.

The following data operations are supported:

- Data standardization (need 3-20 clocks of computing time)
- Logic left shift (need 3～18 clocks of operation time)
- Logic shift right (need 3～18 clocks of operation time)
- 16 bits multiplied by 16 bits (it takes 10 clocks of operation time)
- 16 bits divided by 16 bits (need 9 clocks of operation time)
- 32 bits divided by 16 bits (requires 17 clocks of operation time)

All operations are based on unsigned integer data types.

29.1 Registers Related to MDU16

Symbol	Description	Address	Bit Address and Symbol								Reset Value
			B7	B6	B5	B4	B3	B2	B1	B0	
MD3	MDU Data Register	FCF0H	MD3[7:0]								0000,0000
MD2	MDU Data Register	FCF1H	MD2[7:0]								0000,0000
MD1	MDU Data Register	FCF2H	MD1[7:0]								0000,0000
MD0	MDU Data Register	FCF3H	MD0[7:0]								0000,0000
MD5	MDU Data Register	FCF4H	MD5[7:0]								0000,0000
MD4	MDU Data Register	FCF5H	MD4[7:0]								0000,0000
ARCON	MDU Mode Control Register	FCF6H	MODE[2:0]			SC[4:0]					0000,0000
OPCON	MDU Operation Control Register	FCF7H	-	MDOV	-	-	-	-	RST	ENOP	0000,0000

29.1.1 Operand 1 Data Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
MD3	FCF0H	MD3[7:0]							
MD2	FCF1H	MD2[7:0]							
MD1	FCF2H	MD1[7:0]							
MD0	FCF3H	MD0[7:0]							

29.1.2 Operand 2 Data Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
MD5	FCF4H	MD5[7:0]							
MD4	FCF5H	MD4[7:0]							

32-bit division by 16-bit division:

Dividend: {MD3,MD2,MD1,MD0}
Divisor: {MD5,MD4}
Quotient: {MD3,MD2,MD1,MD0}
Remainder: {MD5,MD4}

16-bit division by 16-bit division:

Dividend: {MD1,MD0}
Divisor: {MD5,MD4}
Quotient: {MD1,MD0}
Remainder: {MD5,MD4}

16-bit multiplication by 16-bit multiplication:

Multiplicand: {MD1,MD0}
Multiplier: {MD5,MD4}
Product: {MD3,MD2,MD1,MD0}

32-bit logical shift left / logical shift right

Operand: {MD3,MD2,MD1,MD0}

32-bit data normalization:

Operand: {MD3,MD2,MD1,MD0}

29.1.3 MDU Mode Control Register (ARCON)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
ARCON	FCF6H	MODE[2:0]			SC[4:0]				

MODE[2:0]: MDU mode selection

MODE[2:0]	Mode	Clocks	Desciption
1	Logical right shift	3～18	Shift the data in {MD3, MD2, MD1, MD0} right by SC [4: 0] bits, MD3 high-order complement 0
2	Logical left shift	3～18	Shift the data in {MD3, MD2, MD1, MD0} SC [4: 0] bits to the left, low-order complement of MD0
3	Data normalization	3～20	Perform logical left shift on the data in {MD3, MD2, MD1, MD0}, shift out all the high-order 0s of the data, make the highest bit of MD3 be 1, and the number of logical left shifts is recorded in SC [4: 0].
4	16-bit multiplication	10	{MD1,MD0}×{MD5,MD4}＝{MD3,MD2,MD1,MD0}
5	16-bit division	9	{MD1,MD0}÷{MD5,MD4}＝ {MD1,MD0}…{MD5,MD4}
6	32-bit division	17	{MD3,MD2,MD1,MD0}÷{MD5,MD4}＝ {MD3,MD2,MD1,MD0}…{MD5,MD4}
Others	Invalid		

SC[4:0]: Data shift bits

When the MDU is in shift mode, SC is used to set the number of bits for left/right shift

When MDU is in data normalization mode, SC is the actual number of bits moved by the data after data normalization

29.1.4 MDU Operation Control Register (OPCON)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
OPCON	FCF7H	-	MDOV	-	-	-	-	RST	ENOP

MDOV: MDU Overflow flag (read-only flag)

MDOV is set by hardware automatically in the following situations:

1. When the divisor is 0;

2. When the product of multiplication is greater than 0FFFFH;

When software writes OPCON.0 (EN) or writes ARCON, MDOV is cleared by the hardware automatically.

RST: Software resets the MDU multiplication and division unit. Writing 1 to it will trigger a software reset. It is cleared by the hardware automatically after the MDU reset is complete.

Note: The value of the ARCON register is cleared when software resets the MDU multiply and divide unit.

ENOP: MDU enable bit.

Writing 1 to this bit will trigger the MDU module to start calculation. When the MDU calculation is completed, ENOP is cleared to 0 by the hardware automatically. After setting ENOP to 1, the software can query ENOP cyclically. When ENOP changes from 1 to 0, the calculation is completed.

29.2 Example Routines

C Language code

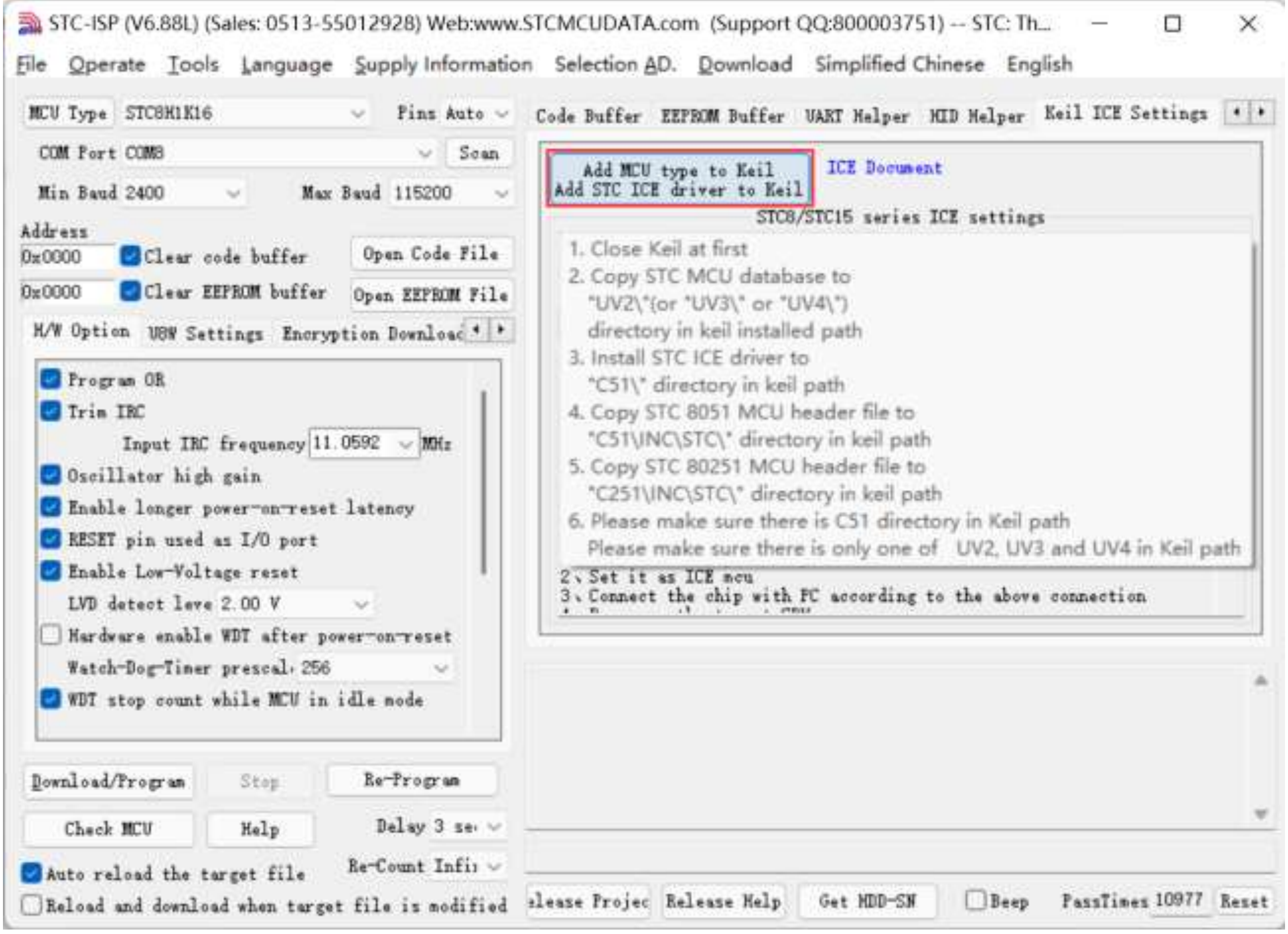
```
;Operating frequency for test is 11.0592MHz

#include  "reg51.h"
#include  "intrins.h"

#define     MD3U32              (*(unsigned long volatile xdata *)0xfcf0)
#define     MD3U16              (*(unsigned int volatile xdata *)0xfcf0)
#define     MD1U16              (*(unsigned int volatile xdata *)0xfcf2)
#define     MD5U16              (*(unsigned int volatile xdata *)0xfcf4)

#define     MD3                 (*(unsigned char volatile xdata *)0xfcf0)
#define     MD2                 (*(unsigned char volatile xdata *)0xfcf1)
#define     MD1                 (*(unsigned char volatile xdata *)0xfcf2)
#define     MD0                 (*(unsigned char volatile xdata *)0xfcf3)
#define     MD5                 (*(unsigned char volatile xdata *)0xfcf4)
#define     MD4                 (*(unsigned char volatile xdata *)0xfcf5)
#define     ARCON               (*(unsigned char volatile xdata *)0xfcf6)
#define     OPCON               (*(unsigned char volatile xdata *)0xfcf7)

sfr         P_SW2       =       0xBA;
///////////////////////////////////////////////////////////////////////////////
//16 bits by 16 bits
///////////////////////////////////////////////////////////////////////////////
unsigned  long  res;
unsigned int dat1, dat2;
P_SW2 |= 0x80;                          // Access the extension register xsfr
MD1U16 = dat1;                              //dat1 User given
MD5U16 = dat2;                              //dat2 User given
ARCON = 4 << 5;                             //16 bits*16 bits, multiplication mode
OPCON = 1;                                  // Start calculation
while((OPCON & 1) != 0);                    // Wait for the calculation to complete
res = MD3U32;                               //32-bit result
///////////////////////////////////////////////////////////////////////////////
//32 bits divided by 16 bits
///////////////////////////////////////////////////////////////////////////////
unsigned long res;
unsigned long dat1;
unsigned int dat2;
P_SW2 |= 0x80;                          // Access the extension register xsfr
MD3U32 = dat1;                              //dat1 User given
MD5U16 = data2;                             //dat2 User given
ARCON = 6 << 5;                             //32-bit/16-bit, division mode
OPCON = 1;                                  // Start calculation
while((OPCON & 1) != 0);                    // Wait for the calculation to complete
res = MD3U32;                               //32-bit quotient, 16-bit remainder in MD5U16
unsigned long res;
unsigned long dat1;
unsigned char num;                          // The number of bits to shift, User given
MD3U32 = dat1;                              //dat1 User given
ARCON = (2 << 5) + num;                     //32-bit left shift mode
//ARCON = (1 << 5) + num;                   //32-bit right shift mode
OPCON = 1;                                  // Start calculation
while((OPCON & 1) != 0);                    // Wait for the calculation to complete
res = MD3U32;                               //32-bit result
```

Appendix A STC Emulator User Guide

A: What kind of compiler/assembler should be used for STC MCU?
Q: Any old 8051 compiler/assembler can support it. Keil C51 is popular now

A: How to include header files in Keil environment
Q: After installing the driver and header files according to the steps shown below, select the STC corresponding MCU model when creating a new project, and directly use "#include <stc8h.h>" in the source file to complete the inclusion of the header file. If you select Intel’s 8052/87C52/87C54/87C58 or Philips P87C52/P87C54/P87C58 compile, the header file contains <reg51.h>, but the new STC special function register needs to be declared by the user.

1.Install Keil version of emulation driver.



MCU Type STC8H1K16 Pins Auto

COM Port COM8

Min Baud 2400 Max Baud 115200

Address
0x0000 ☑ Clear code buffer
0x0000 ☑ Clear EEPROM buffer

☑ Program OR
☑ Trim IRC
Input IRC frequency 11.0592 MHz
☑ Oscillator high gain
☑ Enable longer power-on-reset latency
☑ RESET pin used as I/O port
☑ Enable Low-Voltage reset
LVD detect leve 2.00 V
□ Hardware enable WDT after power-on-reset
Watch-Dog-Timer prescal: 256
☑ WDT stop count while MCU in idle mode

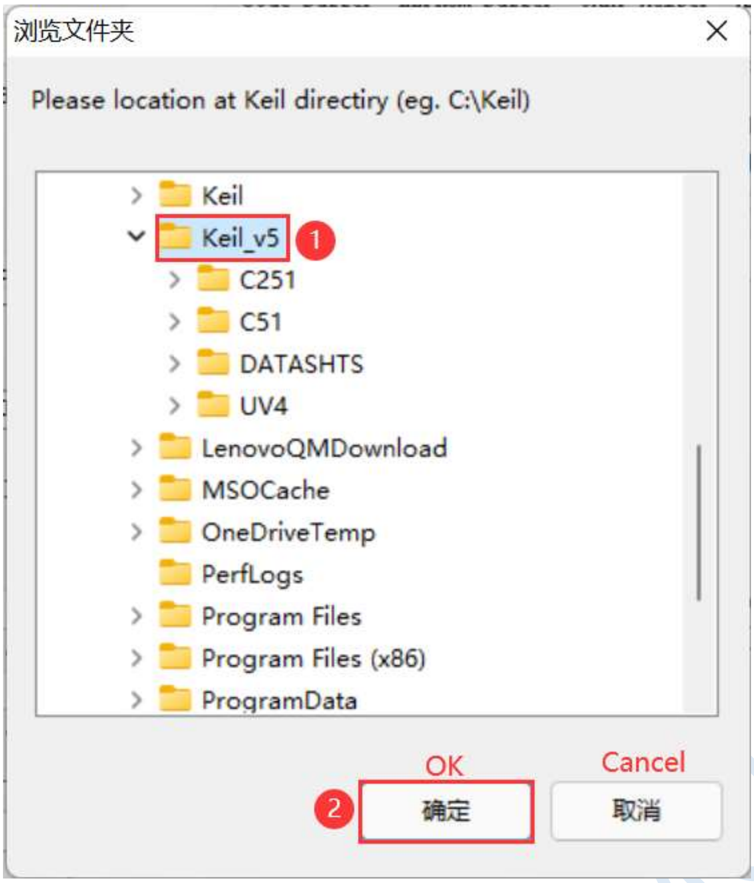
Delay 3 sec
Re-Count Infi
☑ Auto reload the target file
□ Reload and download when target file is modified
□ Beep PassTimes 10977

STC8/STC15 series ICE settings

1. Close Keil at first
2. Copy STC MCU database to "UV2\"(or "UV3\" or "UV4\") directory in keil installed path
3. Install STC ICE driver to "C51\" directory in keil path
4. Copy STC 8051 MCU header file to "C51\INC\STC\" directory in keil path
5. Copy STC 80251 MCU header file to "C251\INC\STC\" directory in keil path
6. Please make sure there is C51 directory in Keil path
Please make sure there is only one of UV2, UV3 and UV4 in Keil path

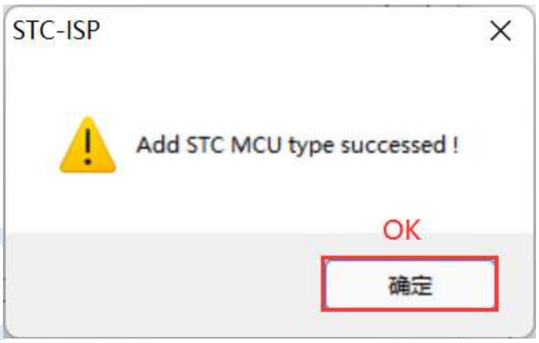
2、Set it as ICE mcu
3、Connect the chip with PC according to the above connection

As shown above, select the "Keil Simulation Settings" page firstly, click "Add MCU model to Keil", and in the following directory selection window that appears, navigate to the installation directory of Keil (usually "C: \ Keil \") , After press "OK" button, the prompt message shown on the right in the following figure appears, indicating that the installation was successful. The STC Monitor51 emulation driver STCMON51.DLL will also be installed when adding the header file. The installation directory of the driver and header file is shown above.



浏览文件夹

Please location at Keil directiry (eg. C:\Keil)

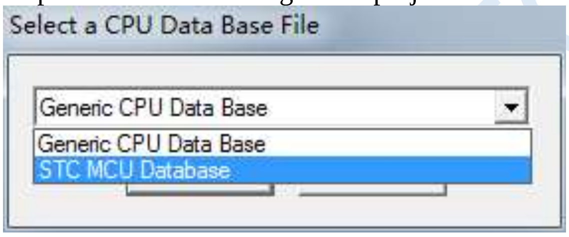


STC-ISP

Add STC MCU type successed !

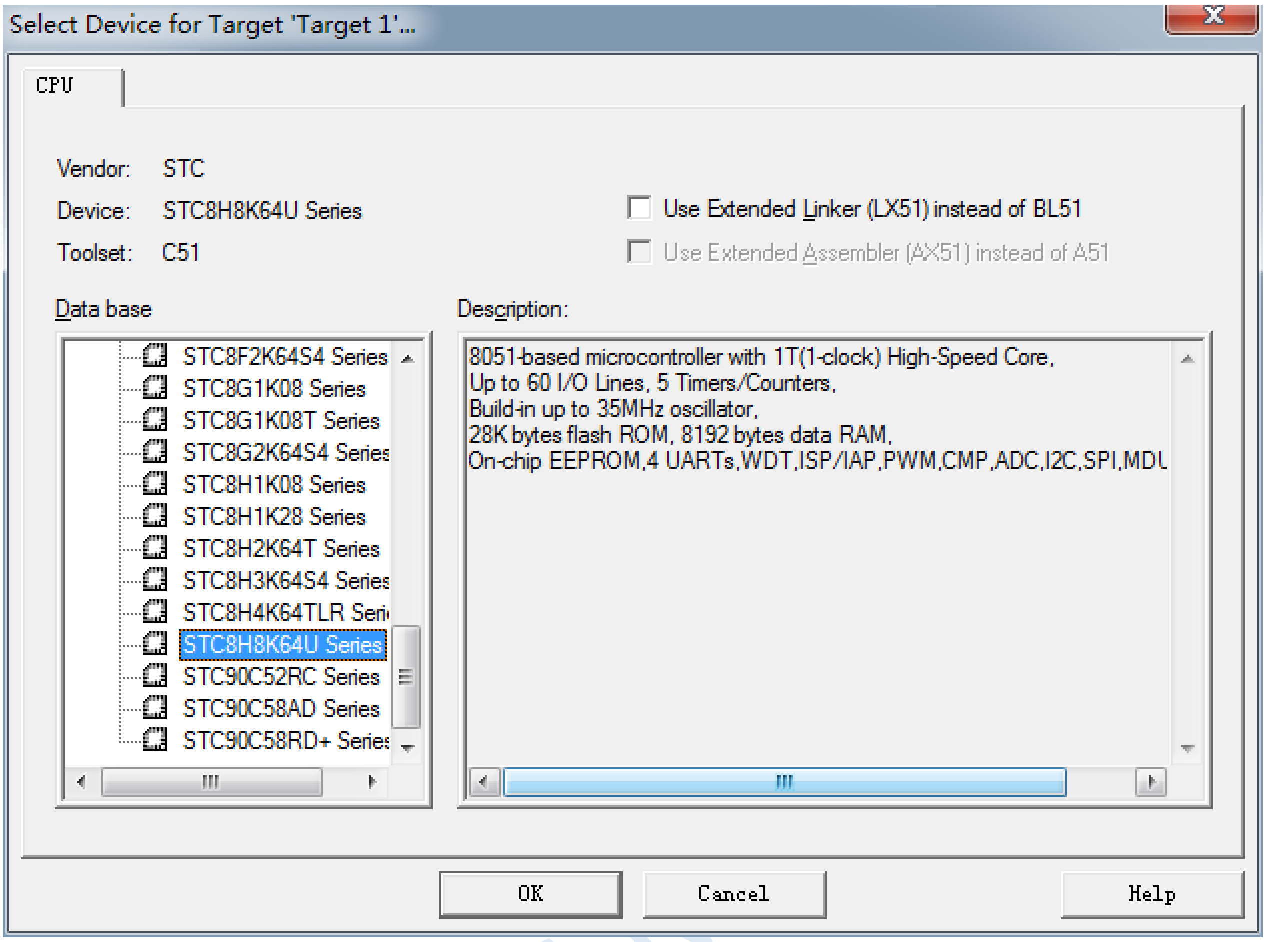
2. Create a project in Keil

If the driver installation is successful in the first step, there will be an option of "STC MCU Database" in selecting the chip model when creating a new project in Keil as shown below.



Select a CPU Data Base File

Then select the responding MCU model from the list. Here we select the model of "STC8A8K64S4A12" and click "OK" to complete the selection.



Vendor: STC
Device: STC8H8K64U Series
Toolset: C51

□ Use Extended Linker (LX51) instead of BL51
□ Use Extended Assembler (AX51) instead of A51

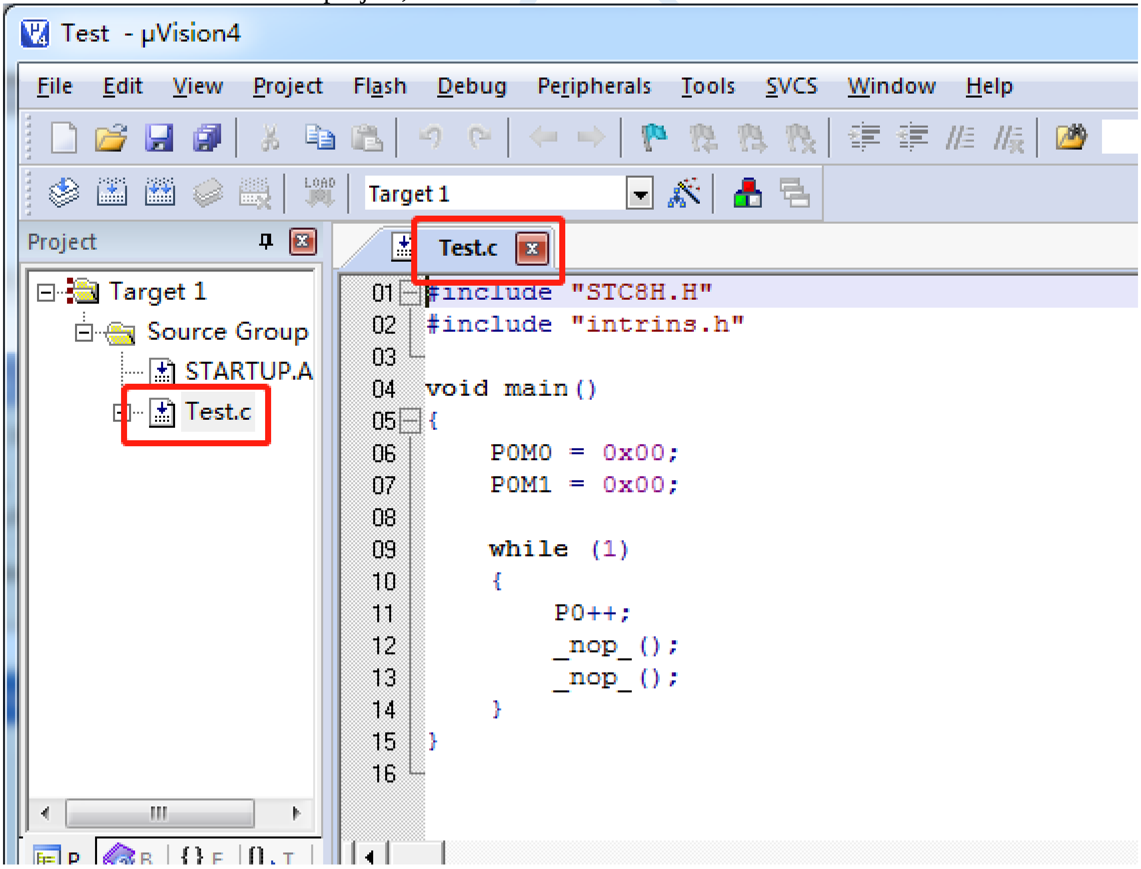
Data base

- STC8F2K64S4 Series
- STC8G1K08 Series
- STC8G1K08T Series
- STC8G2K64S4 Series
- STC8H1K08 Series
- STC8H1K28 Series
- STC8H2K64T Series
- STC8H3K64S4 Series
- STC8H4K64TLR Seri
- STC8H8K64U Series
- STC90C52RC Series
- STC90C58AD Series
- STC90C58RD+ Series

Description:

8051-based microcontroller with 1T(1-clock) High-Speed Core,
Up to 60 I/O Lines, 5 Timers/Counters,
Build-in up to 35MHz oscillator,
28K bytes flash ROM, 8192 bytes data RAM,
On-chip EEPROM,4 UARTs,WDT,ISP/IAP,PWM,CMP,ADC,I2C,SPI,MDU

Add source code files to the project, as shown below:

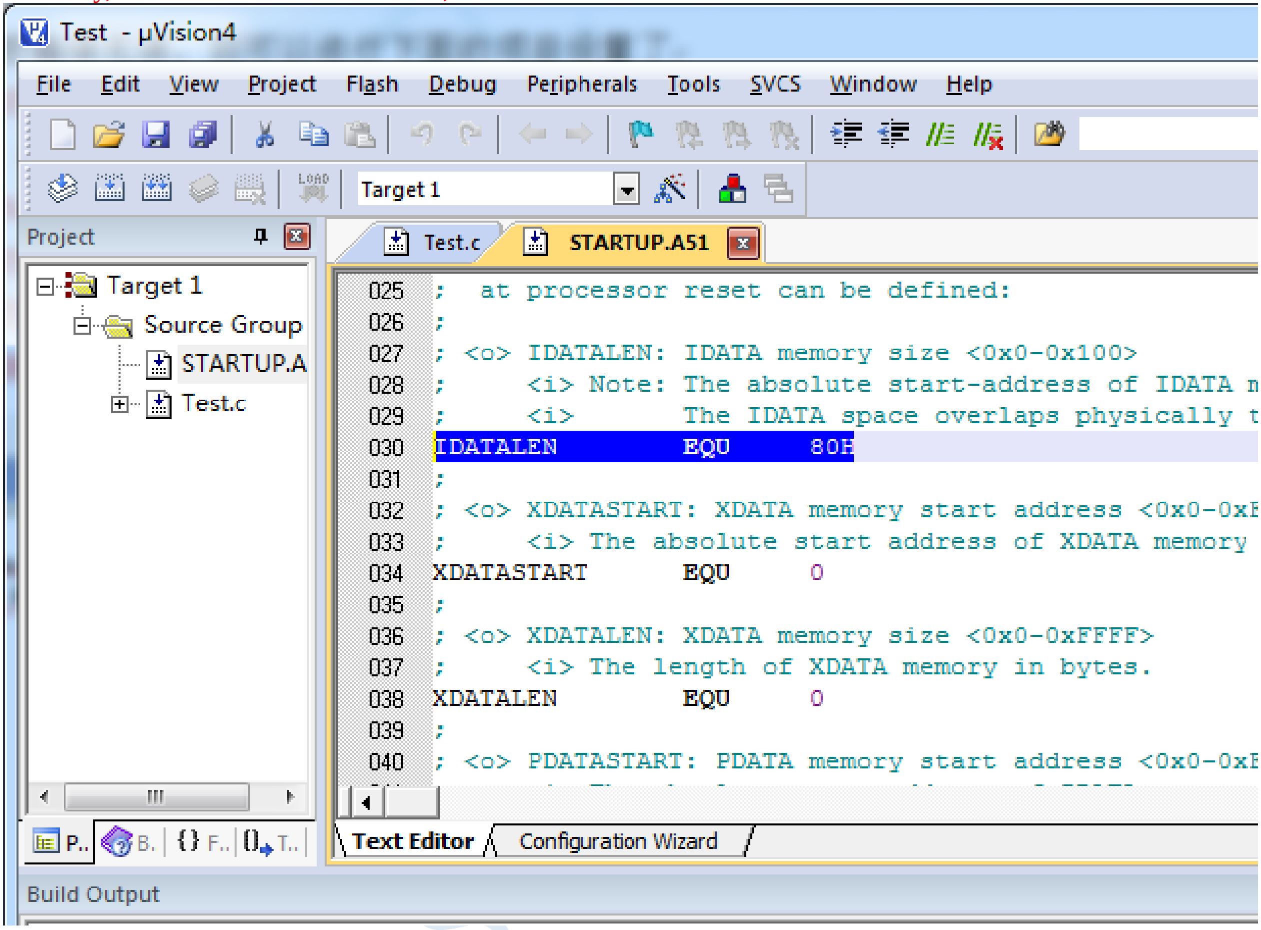


```
#include "STC8H.H"
#include "intrins.h"

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;

    while (1)
    {
        P0++;
        _nop_();
        _nop_();
    }
}

```

Save the project. If there is no error while compiling the project, you can set the following projects.

One additional note:

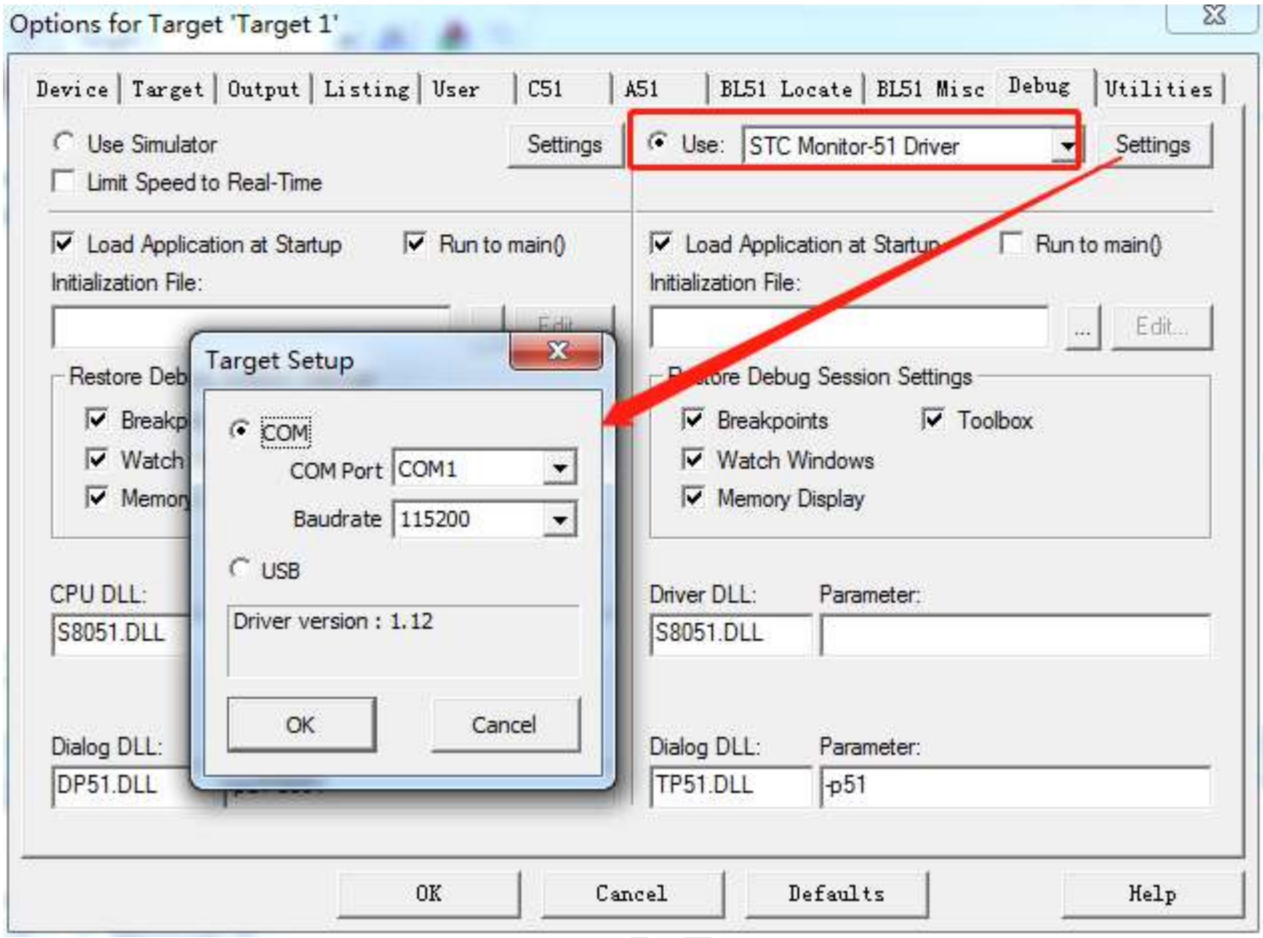
When a C language project is created and a startup file "STARTUP.A51" is added to the project, there is a macro definition named "IDATALEN", which is a macro used to define the size of the IDATA. The default value is 128, which is 80H in hexadecimal, and it is also the size of IDATA in the startup file that needs to be initialized to 0. Therefor if IDATA is defined as 80H, the code in STARTUP.A51 will initialize the RAM of 00-7F of IDATA to 0. Similarly, if IDATA is defined as 0FFH, the RAM of 00-FF of IDATA will be initialized to 0.



```
;  at processor reset can be defined:
;
; <o> IDATALEN: IDATA memory size <0x0-0x100>
;      <i> Note: The absolute start-address of IDATA m
;      <i>           The IDATA space overlaps physically t
IDATALEN          EQU       80H
;
; <o> XDATASTART: XDATA memory start address <0x0-0xF
;      <i> The absolute start address of XDATA memory
XDATASTART        EQU       0
;
; <o> XDATALEN: XDATA memory size <0x0-0xFFFF>
;      <i> The length of XDATA memory in bytes.
XDATALEN          EQU       0
;
; <o> PDATASTART: PDATA memory start address <0x0-0xF
```

The IDATA size of the STC8 series of microcontrollers is 256 bytes (DATA of 00-7F and IDATA of 80H-FFH). Because the last 17 bytes of RAM have the ID number and related test parameters, if you need to use this part of the data in the program, you must not define IDATALEN as 256.

3. Project settings, select STC simulation driver.



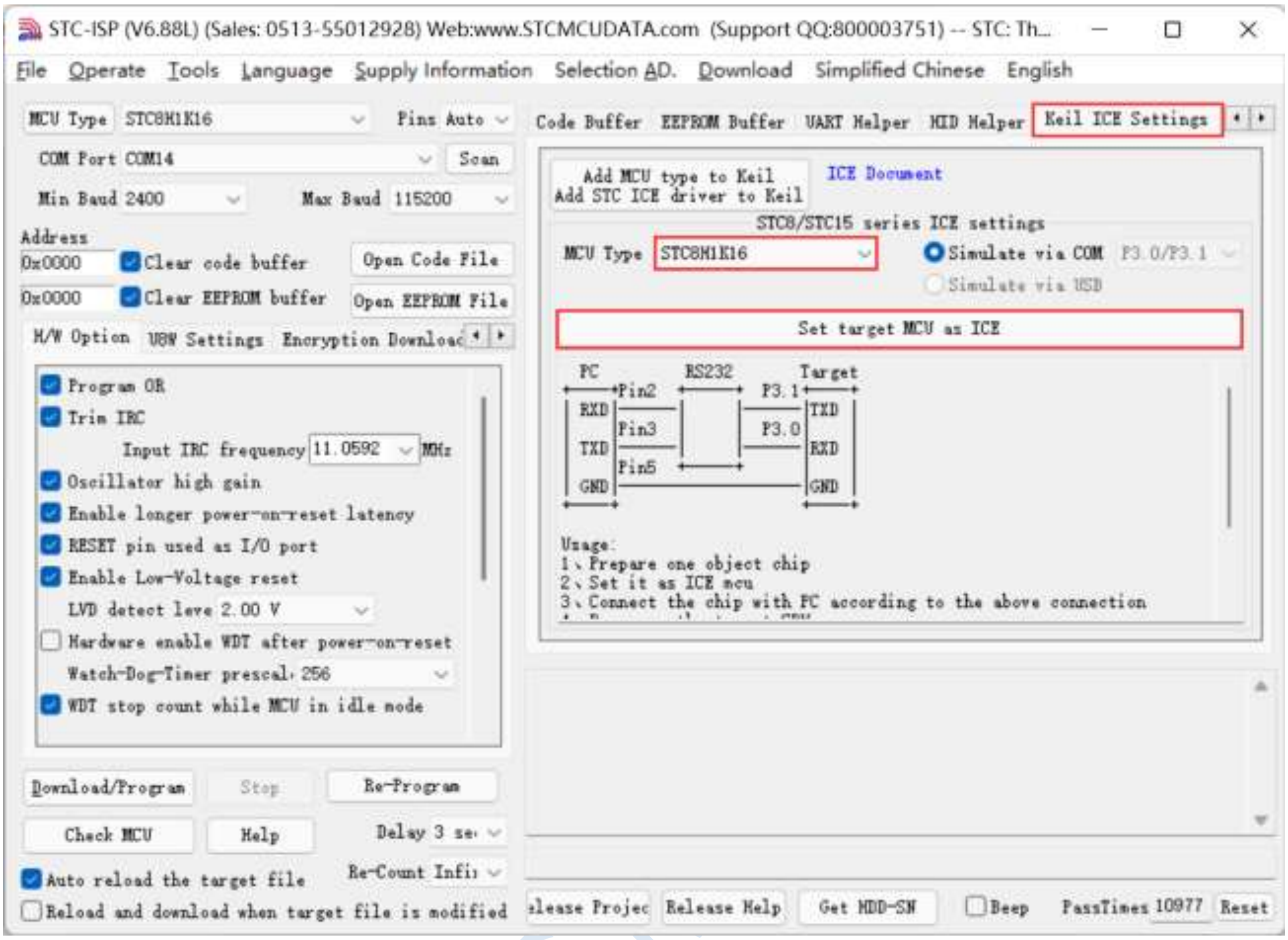
○ Use Simulator
□ Limit Speed to Real-Time
☑ Load Application at Startup ☑ Run to main()
Initialization File:
Restore Debug Session Settings
☑ Breakpoints
☑ Watch Windows
☑ Memory Display
CPU DLL: S8051.DLL
Dialog DLL: DP51.DLL

● Use: STC Monitor-51 Driver
☑ Load Application at Startup □ Run to main()
Initialization File:
Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watch Windows
☑ Memory Display
Driver DLL: S8051.DLL Parameter:
Dialog DLL: TP51.DLL Parameter: -p51

Target Setup
● COM
COM Port: COM1
Baudrate: 115200
○ USB
Driver version : 1.12

As shown above, enter the project setting page firstly, select the "Debug" setting page, select the hardware emulation "Use…" on the right, and select "STC Monitor-51 Driver" in the emulation driver drop-down list. And then click the "Settings" button to enter the following setting screen. Set the port number and baud rate of the serial port. The baud rate is generally 115200. Then complete the setup.

4.Create simulation chip



MCU Type STC8H1K16 Pins Auto
COM Port COM14
Min Baud 2400 Max Baud 115200
Address
0x0000 ☑ Clear code buffer
0x0000 ☑ Clear EEPROM buffer

☑ Program OR
☑ Trim IRC
Input IRC frequency 11.0592 MHz
☑ Oscillator high gain
☑ Enable longer power-on-reset latency
☑ RESET pin used as I/O port
☑ Enable Low-Voltage reset
LVD detect leve 2.00 V
□ Hardware enable WDT after power-on-reset
Watch-Dog-Timer prescal: 256
☑ WDT stop count while MCU in idle mode

Delay 3 sec
Re-Count Infinite
☑ Auto reload the target file
□ Reload and download when target file is modified

STC8/STC15 series ICE settings
MCU Type STC8H1K16
● Simulate via COM P3.0/P3.1
○ Simulate via USB

Usage:
1、Prepare one object chip
2、Set it as ICE mcu
3、Connect the chip with PC according to the above connection

□ Beep PassTimes 10977

Prepare a STC8H series chip, and connect it to the serial port of the computer through the download board. Then select the correct chip model as shown above, and enter the "Keil simulation settings" page, click the button of the corresponding model. After the program downloading completes, the simulator is ready for use.

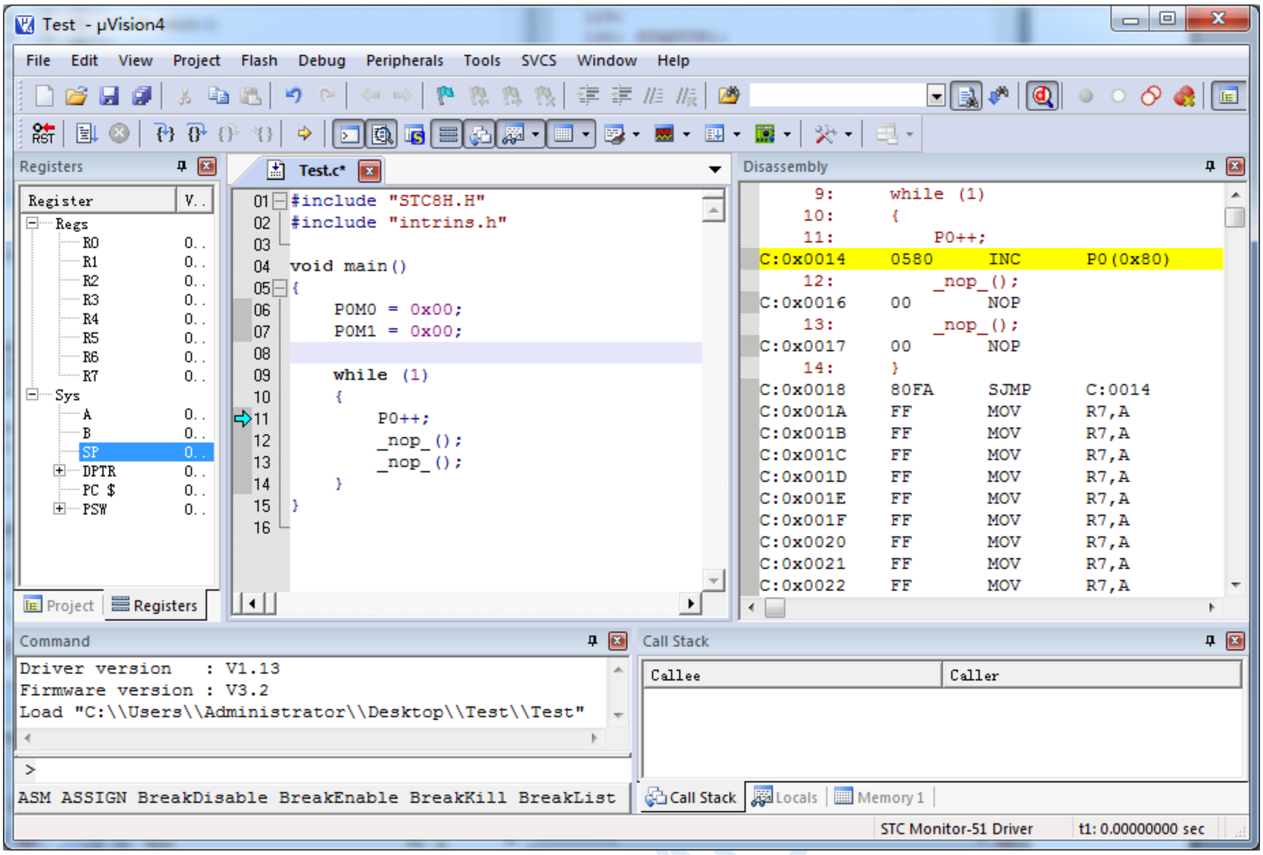
5.Start simulation

Connect the completed simulation chip to the computer through the serial port.

After compiling the project we created before without errors, press "Ctrl + F5" to start debugging.

If the hardware connection is correct, you will enter a debugging interface similar to the following, and display the current simulation driver version number and the current simulation monitoring code firmware version number in the command output window.

The current maximum number of breakpoints is 20 (in theory, any number of breakpoints can be set, but too many breakpoints will affect the speed of debugging).



```
Register  V..
Regs
  R0      0..
  R1      0..
  R2      0..
  R3      0..
  R4      0..
  R5      0..
  R6      0..
  R7      0..
Sys
  A       0..
  B       0..
  SP      0..
  DPTR    0..
  PC $    0..
  PSW     0..
```

```
#include "STC8H.H"
#include "intrins.h"

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;

    while (1)
    {
        P0++;
        _nop_();
        _nop_();
    }
}
```

```
     9:     while (1)
    10:     {
    11:         P0++;
C:0x0014    0580     INC      P0(0x80)
    12:         _nop_();
C:0x0016    00       NOP
    13:         _nop_();
C:0x0017    00       NOP
    14:     }
C:0x0018    80FA     SJMP     C:0014
C:0x001A    FF       MOV      R7,A
C:0x001B    FF       MOV      R7,A
C:0x001C    FF       MOV      R7,A
C:0x001D    FF       MOV      R7,A
C:0x001E    FF       MOV      R7,A
C:0x001F    FF       MOV      R7,A
C:0x0020    FF       MOV      R7,A
C:0x0021    FF       MOV      R7,A
C:0x0022    FF       MOV      R7,A
```

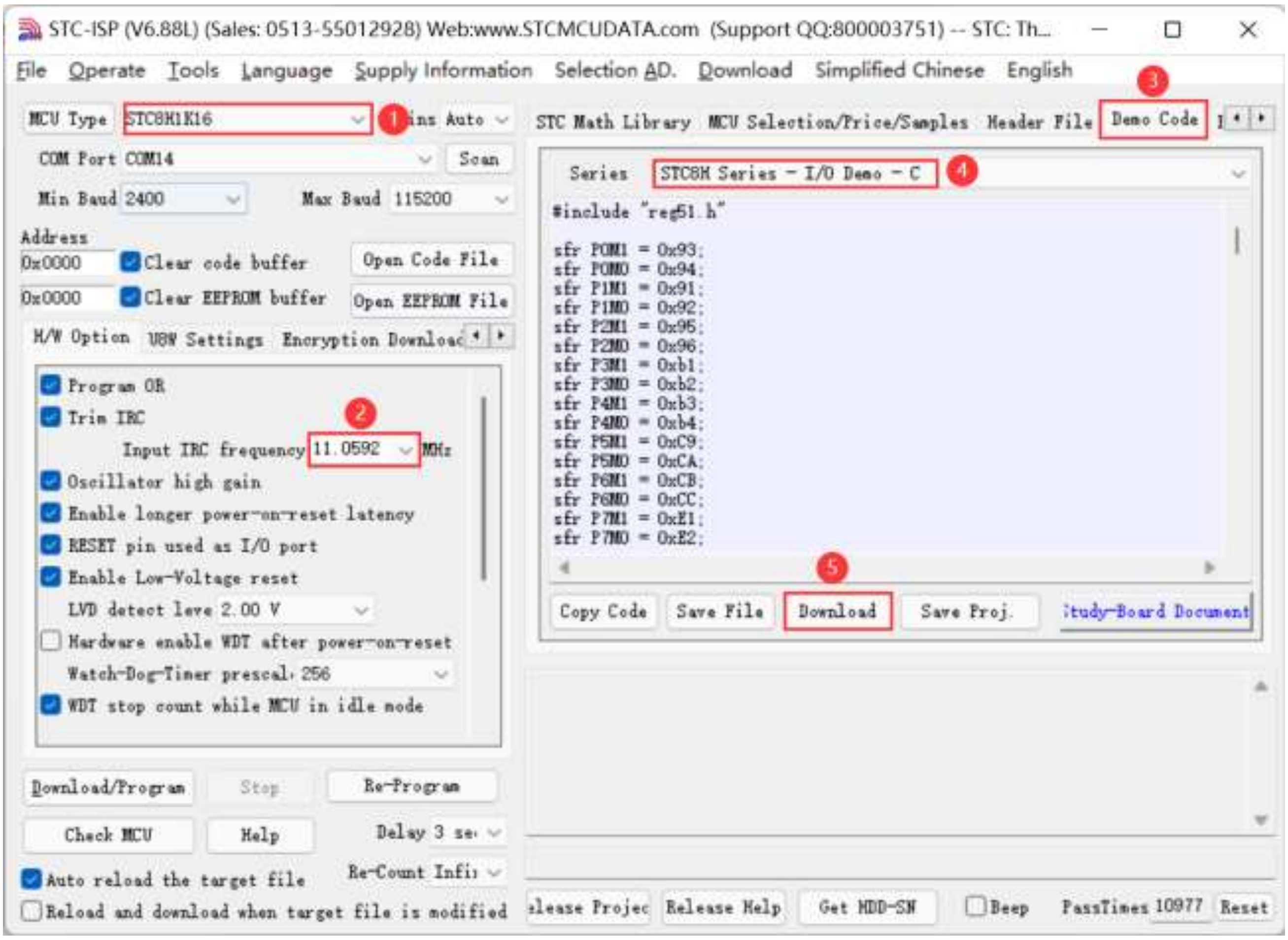
```
Driver version   : V1.13
Firmware version : V3.2
Load "C:\\Users\\Administrator\\Desktop\\Test\\Test"
>
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

Simulation considerations:

1. The simulation monitoring program occupies the two ports P3.0/P3.1, but does not occupy the serial port 1. The user can switch the serial port 1 to P3.6/P3.7 or P1.6/P1.7 to be used.

2. The simulation monitoring program occupies the last 768 bytes of the internal extended RAM (XDATA), and the user cannot write to XDATA in this area .

Appendix B How to Test I/O Ports


STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...

MCU Type STC8H1K16 ① Pins Auto

COM Port COM14

Min Baud 2400 Max Baud 115200

Address

0x0000 ☑ Clear code buffer

0x0000 ☑ Clear EEPROM buffer

☑ Program OR

☑ Trim IRC

Input IRC frequency 11.0592 MHz ②

☑ Oscillator high gain

☑ Enable longer power-on-reset latency

☑ RESET pin used as I/O port

☑ Enable Low-Voltage reset

LVD detect leve 2.00 V

☐ Hardware enable WDT after power-on-reset

Watch-Dog-Timer prescal 256

☑ WDT stop count while MCU in idle mode

Delay 3 sec

Re-Count Infinite

☑ Auto reload the target file

☐ Reload and download when target file is modified

Series STC8H Series - I/O Demo - C ④

```
#include "reg51.h"

sfr P0M1 = 0x93;
sfr P0M0 = 0x94;
sfr P1M1 = 0x91;
sfr P1M0 = 0x92;
sfr P2M1 = 0x95;
sfr P2M0 = 0x96;
sfr P3M1 = 0xb1;
sfr P3M0 = 0xb2;
sfr P4M1 = 0xb3;
sfr P4M0 = 0xb4;
sfr P5M1 = 0xC9;
sfr P5M0 = 0xCA;
sfr P6M1 = 0xCB;
sfr P6M0 = 0xCC;
sfr P7M1 = 0xE1;
sfr P7M0 = 0xE2;
```

☐ Beep PassTimes 10977

Test I/O port steps:

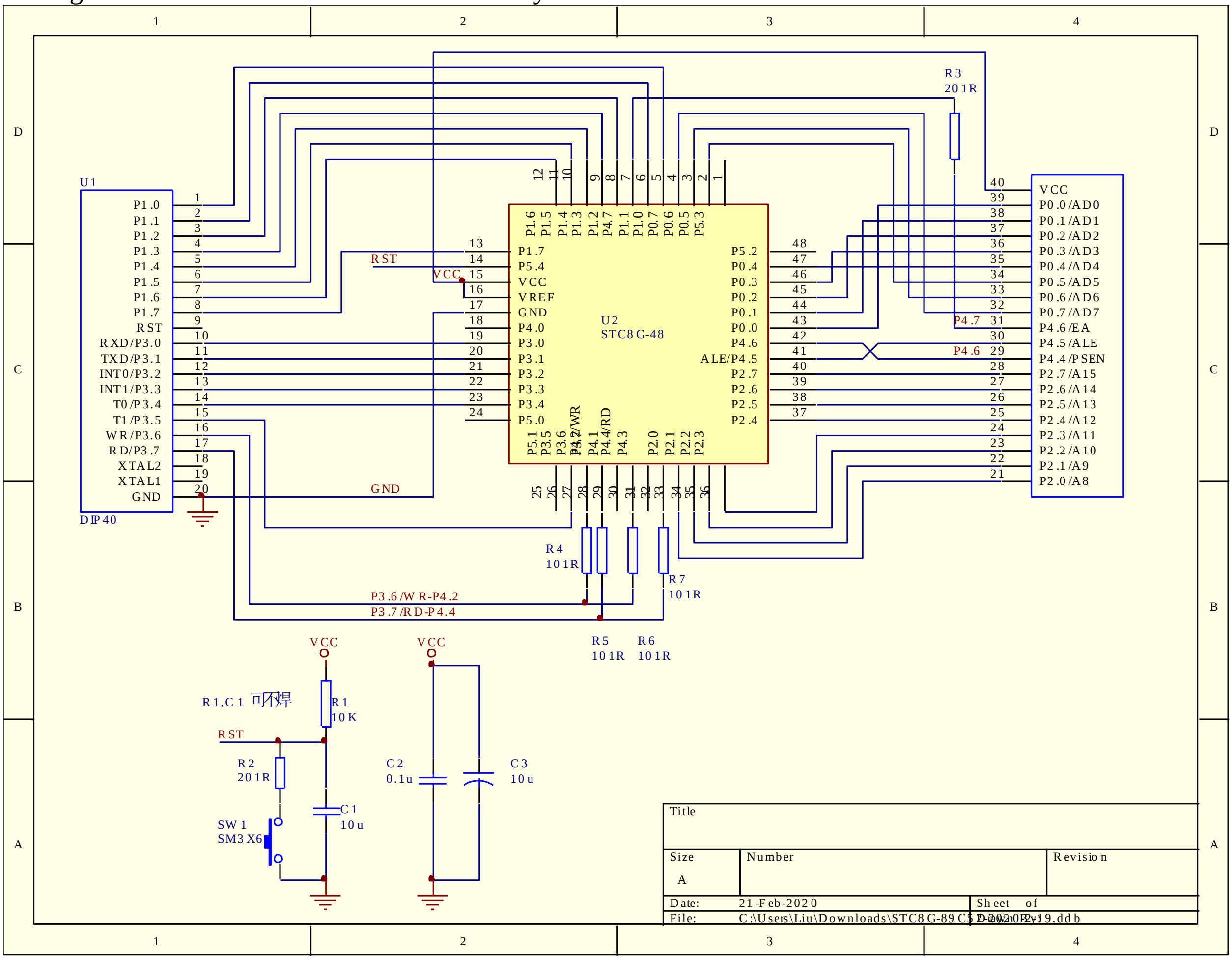
1. Select the microcontroller model
2. Set the operating frequency of the test program
3. Open the "Example Program" page
4. Select the "I/O port test" program of STC8G or STC8H series
5. Click "Download Hex Directly" on the page

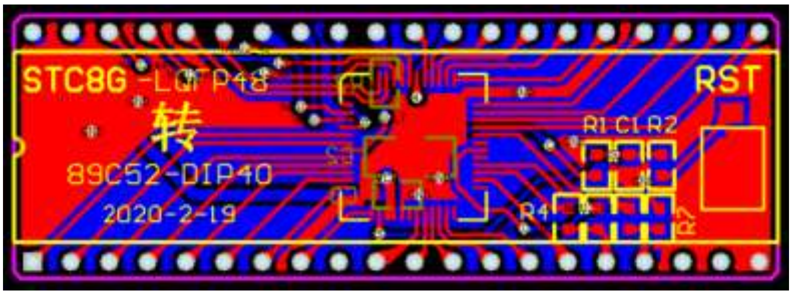
After the download is completed, the running water light program will be executed on all I/O ports. At this time, you can connect LEDs to the I/O ports or use an oscilloscope to see the waveform.

Appendix C How to Make the Traditional 8051 MCU EVB Emulatable

The traditional 8051 microcontroller EVB does not have simulation function. To enable the traditional 8051 microcontroller EVB to be simulated, a conversion board is needed. The physical picture of the conversion board is shown below. The converted pin arrangement is basically the same as that of the traditional 8051. Therefore, the simulation function of the standard 8051 learning board can be realized.

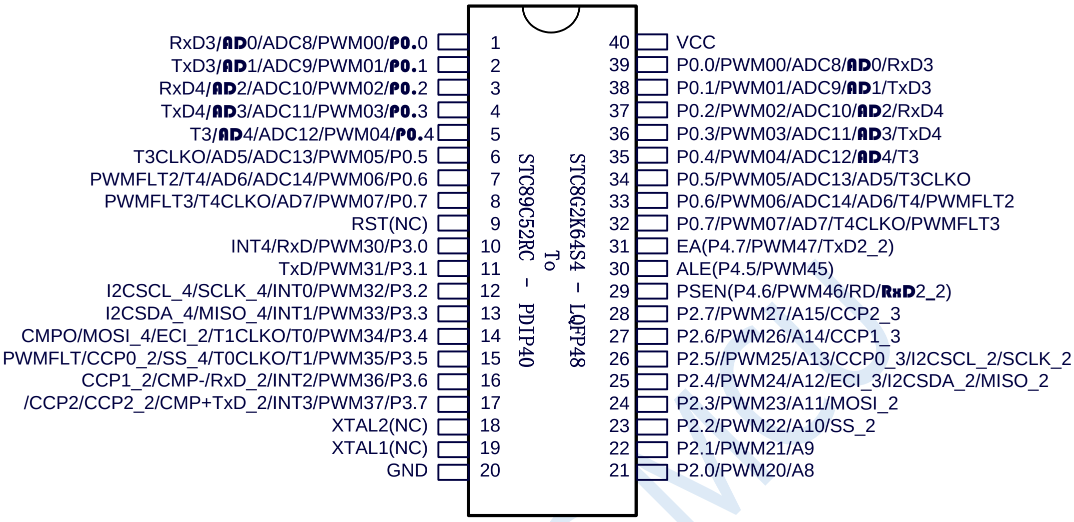
The figures below are the schematic and PCB layout of the converter board.





This conversion board can be used for STC8H series LQFP48 to STC89C52RC / STC89C58RD + series simulation.

The following figure is a functional diagram of the conversion board.



STC8G2K64S4 – LQFP48
To
STC89C52RC – PDIP40
RxD3/AD0/ADC8/PWM00/P0.0 1
TxD3/AD1/ADC9/PWM01/P0.1 2
RxD4/AD2/ADC10/PWM02/P0.2 3
TxD4/AD3/ADC11/PWM03/P0.3 4
T3/AD4/ADC12/PWM04/P0.4 5
T3CLKO/AD5/ADC13/PWM05/P0.5 6
PWMFLT2/T4/AD6/ADC14/PWM06/P0.6 7
PWMFLT3/T4CLKO/AD7/PWM07/P0.7 8
RST(NC) 9
INT4/RxD/PWM30/P3.0 10
TxD/PWM31/P3.1 11
I2CSCL_4/SCLK_4/INT0/PWM32/P3.2 12
I2CSDA_4/MISO_4/INT1/PWM33/P3.3 13
CMPO/MOSI_4/ECI_2/T1CLKO/T0/PWM34/P3.4 14
PWMFLT/CCP0_2/SS_4/T0CLKO/T1/PWM35/P3.5 15
CCP1_2/CMP-/RxD_2/INT2/PWM36/P3.6 16
/CCP2/CCP2_2/CMP+TxD_2/INT3/PWM37/P3.7 17
XTAL2(NC) 18
XTAL1(NC) 19
GND 20
40 VCC
39 P0.0/PWM00/ADC8/AD0/RxD3
38 P0.1/PWM01/ADC9/AD1/TxD3
37 P0.2/PWM02/ADC10/AD2/RxD4
36 P0.3/PWM03/ADC11/AD3/TxD4
35 P0.4/PWM04/ADC12/AD4/T3
34 P0.5/PWM05/ADC13/AD5/T3CLKO
33 P0.6/PWM06/ADC14/AD6/T4/PWMFLT2
32 P0.7/PWM07/AD7/T4CLKO/PWMFLT3
31 EA(P4.7/PWM47/TxD2_2)
30 ALE(P4.5/PWM45)
29 PSEN(P4.6/PWM46/RD/RxD2_2)
28 P2.7/PWM27/A15/CCP2_3
27 P2.6/PWM26/A14/CCP1_3
26 P2.5//PWM25/A13/CCP0_3/I2CSCL_2/SCLK_2
25 P2.4/PWM24/A12/ECI_3/I2CSDA_2/MISO_2
24 P2.3/PWM23/A11/MOSI_2
23 P2.2/PWM22/A10/SS_2
22 P2.1/PWM21/A9
21 P2.0/PWM20/A8

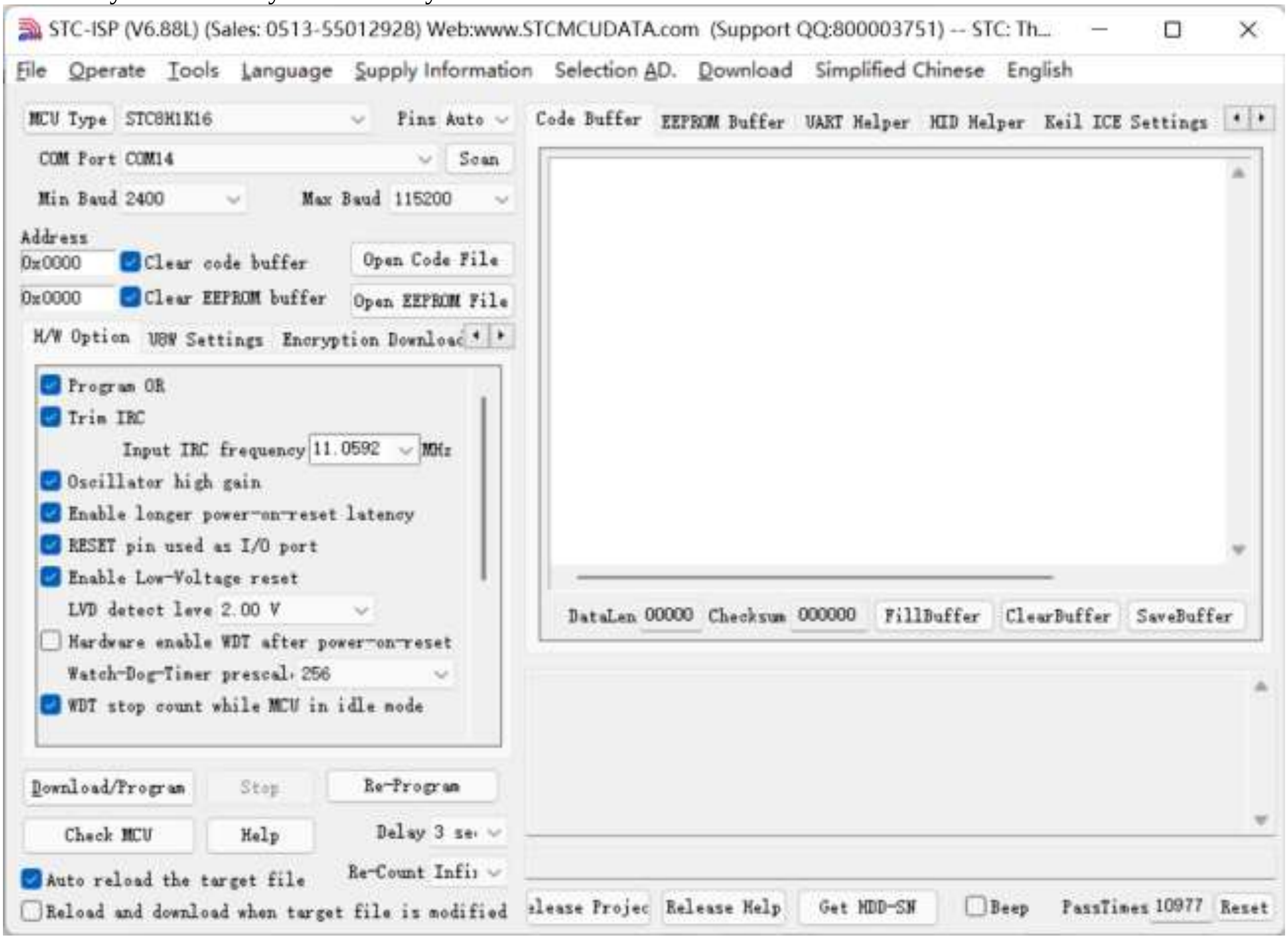
Note:

- ✓ Due to the built-in high-precision R/C clock, no external crystal is needed, XTAL1 and XTAL2 can be empty.
- ✓ WR and RD are P4.2/ WR and P4.4/ RD respectively, not traditional WR/P3.6 and RD/P3.7. **(In the conversion board, P4.2 and P3.6 are connected together, and P4.4 and P3.7 are connected together. When this conversion board is used to access the external bus, P3.6 and P3.7 should be set to high-impedance input mode, so that P4.2 and P4.4 can normally output the bus read and write signals. If the external bus is not needed to be accessed, P4.2 and P4.4 should be set to high-impedance input mode, and P3.6 and P3.7 are ordinary I/O.)**
- ✓ The STC8H series MCUs are low-level reset, it is not compatible with the high-level reset of the traditional 8051, so the RST pin is left floating, and replaced by the reset button and reset circuit on the conversion board.

Appendix D STC-USB Driver Installation Instructions

Installation Instructions in Windows XP

Open the STC-ISP download software of V6.79 (or later). The download software will copy the driver files to the relevant system directory automatically.



STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...

MCU Type STC8H1K16　Pins Auto

COM Port COM14

Min Baud 2400　Max Baud 115200

Address

0x0000 ☑ Clear code buffer

0x0000 ☑ Clear EEPROM buffer

☑ Program OR

☑ Trim IRC

Input IRC frequency 11.0592 MHz

☑ Oscillator high gain

☑ Enable longer power-on-reset latency

☑ RESET pin used as I/O port

☑ Enable Low-Voltage reset

LVD detect leve 2.00 V

☐ Hardware enable WDT after power-on-reset

Watch-Dog-Timer prescal 256

☑ WDT stop count while MCU in idle mode

Delay 3 se

Re-Count Infi

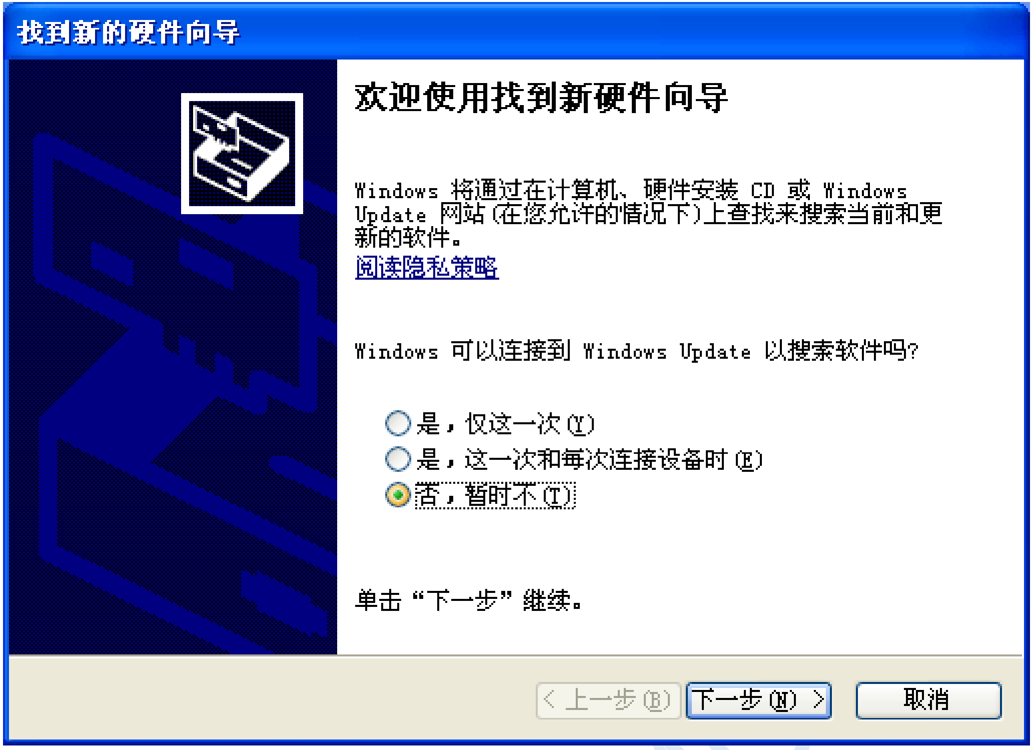
☑ Auto reload the target file

☐ Reload and download when target file is modified

DataLen 00000　Checksum 000000

☐ Beep　PassTimes 10977

Plug in the USB device, the system will pop up the following dialog box automatically after finding the device, select "No, not this time".



找到新的硬件向导

欢迎使用找到新硬件向导

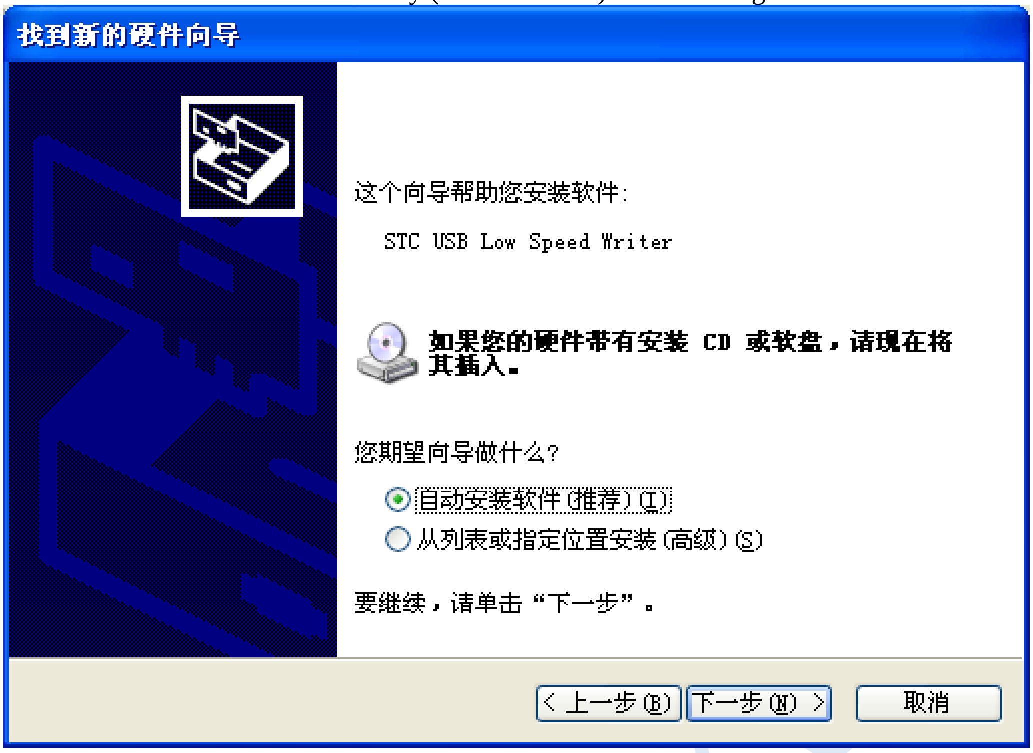
Windows 将通过在计算机、硬件安装 CD 或 Windows Update 网站(在您允许的情况下)上查找来搜索当前和更新的软件。
阅读隐私策略

Windows 可以连接到 Windows Update 以搜索软件吗?

○ 是，仅这一次(Y)
○ 是，这一次和每次连接设备时(E)
● 否，暂时不(T)

单击“下一步”继续。

Select "Install software automatically (recommended)" in the dialog below.



找到新的硬件向导

这个向导帮助您安装软件:

STC USB Low Speed Writer

如果您的硬件带有安装 CD 或软盘，请现在将其插入。

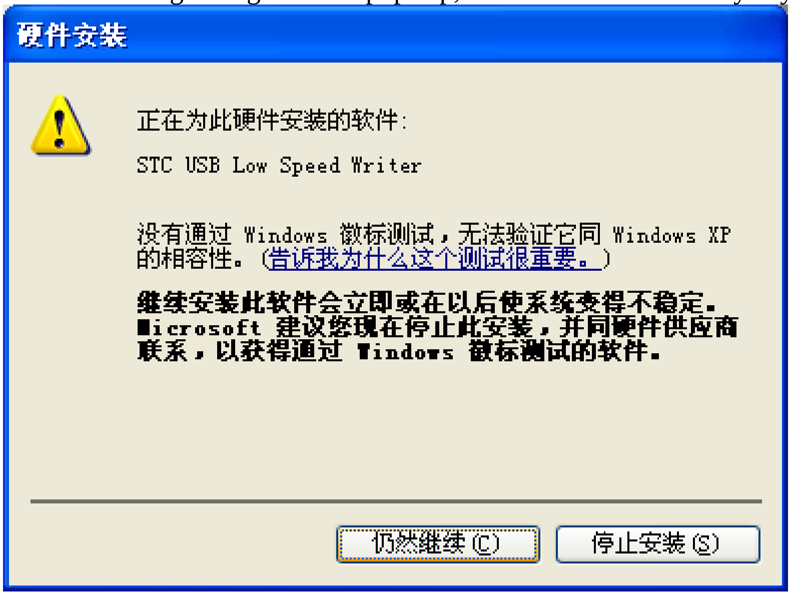
您期望向导做什么?

● 自动安装软件(推荐)(I)

○ 从列表或指定位置安装(高级)(S)

要继续，请单击“下一步”。

In the following dialog box that pops up, select the "Continue Anyway" button.



硬件安装

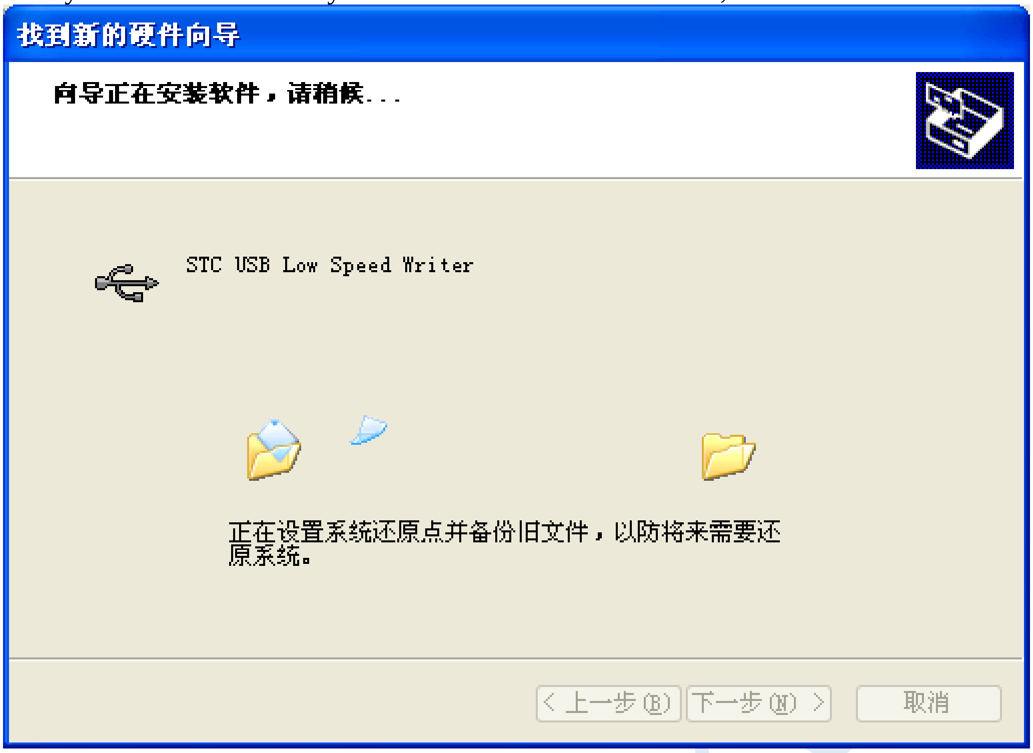
正在为此硬件安装的软件:

STC USB Low Speed Writer

没有通过 Windows 徽标测试，无法验证它同 Windows XP 的相容性。(告诉我为什么这个测试很重要。)

继续安装此软件会立即或在以后使系统变得不稳定。Microsoft 建议您现在停止此安装，并同硬件供应商联系，以获得通过 Windows 徽标测试的软件。

The system will automatically install the driver when connected, as shown below



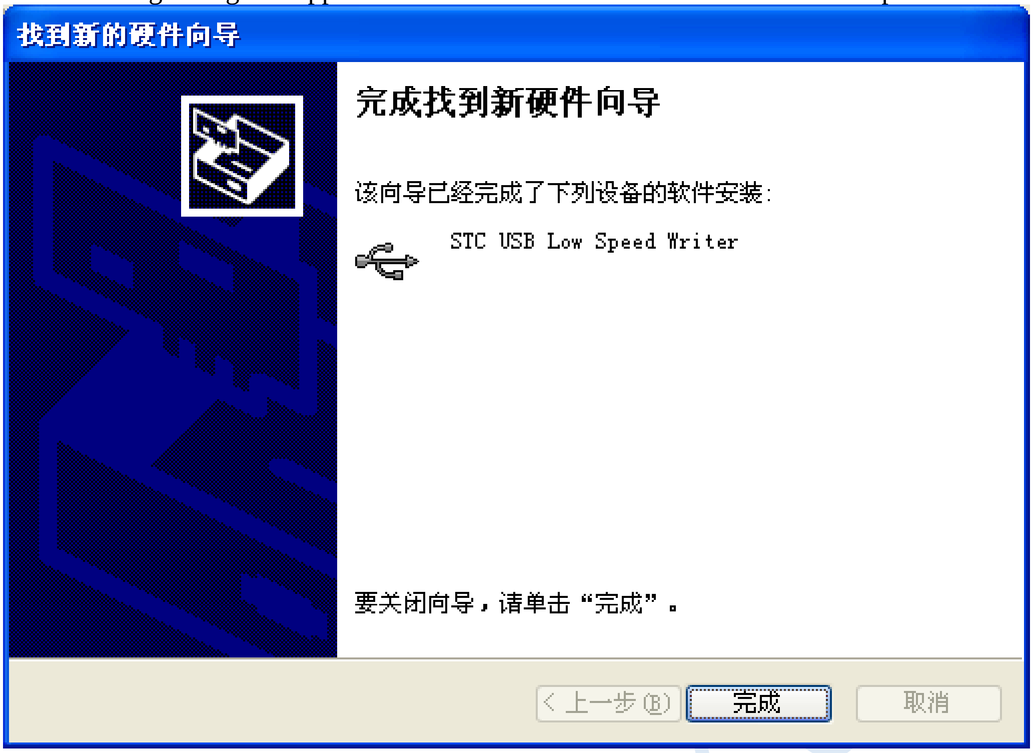
找到新的硬件向导

向导正在安装软件，请稍候...

STC USB Low Speed Writer

正在设置系统还原点并备份旧文件，以防将来需要还原系统。

The following dialog box appears to indicate that the driver installation is complete.



找到新的硬件向导

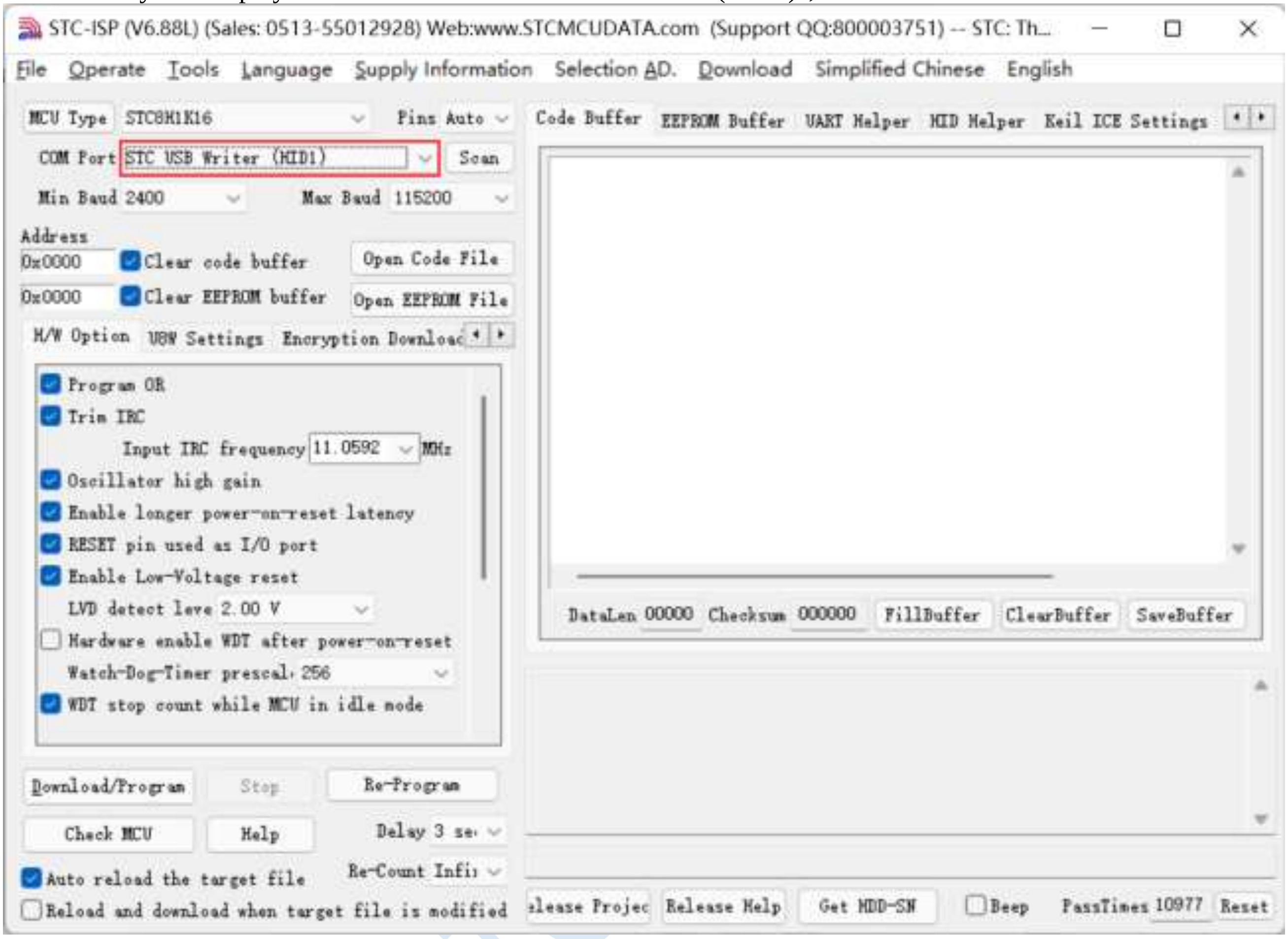
完成找到新硬件向导

该向导已经完成了下列设备的软件安装：

STC USB Low Speed Writer

要关闭向导，请单击“完成”。

Now, the serial number list in the previously opened STC-ISP download software will select the inserted USB device automatically and display the device name as "STC USB Writer (HID1)", as shown below.



STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...

MCU Type STC8H1K16 Pins Auto

COM Port STC USB Writer (HID1)

Min Baud 2400 Max Baud 115200

Address

0x0000 ☑ Clear code buffer

0x0000 ☑ Clear EEPROM buffer

☑ Program OR

☑ Trim IRC

Input IRC frequency 11.0592 MHz

☑ Oscillator high gain

☑ Enable longer power-on-reset latency

☑ RESET pin used as I/O port

☑ Enable Low-Voltage reset

LVD detect leve 2.00 V

☐ Hardware enable WDT after power-on-reset

Watch-Dog-Timer prescal: 256

☑ WDT stop count while MCU in idle mode

Delay 3 se

Re-Count Infi

☑ Auto reload the target file

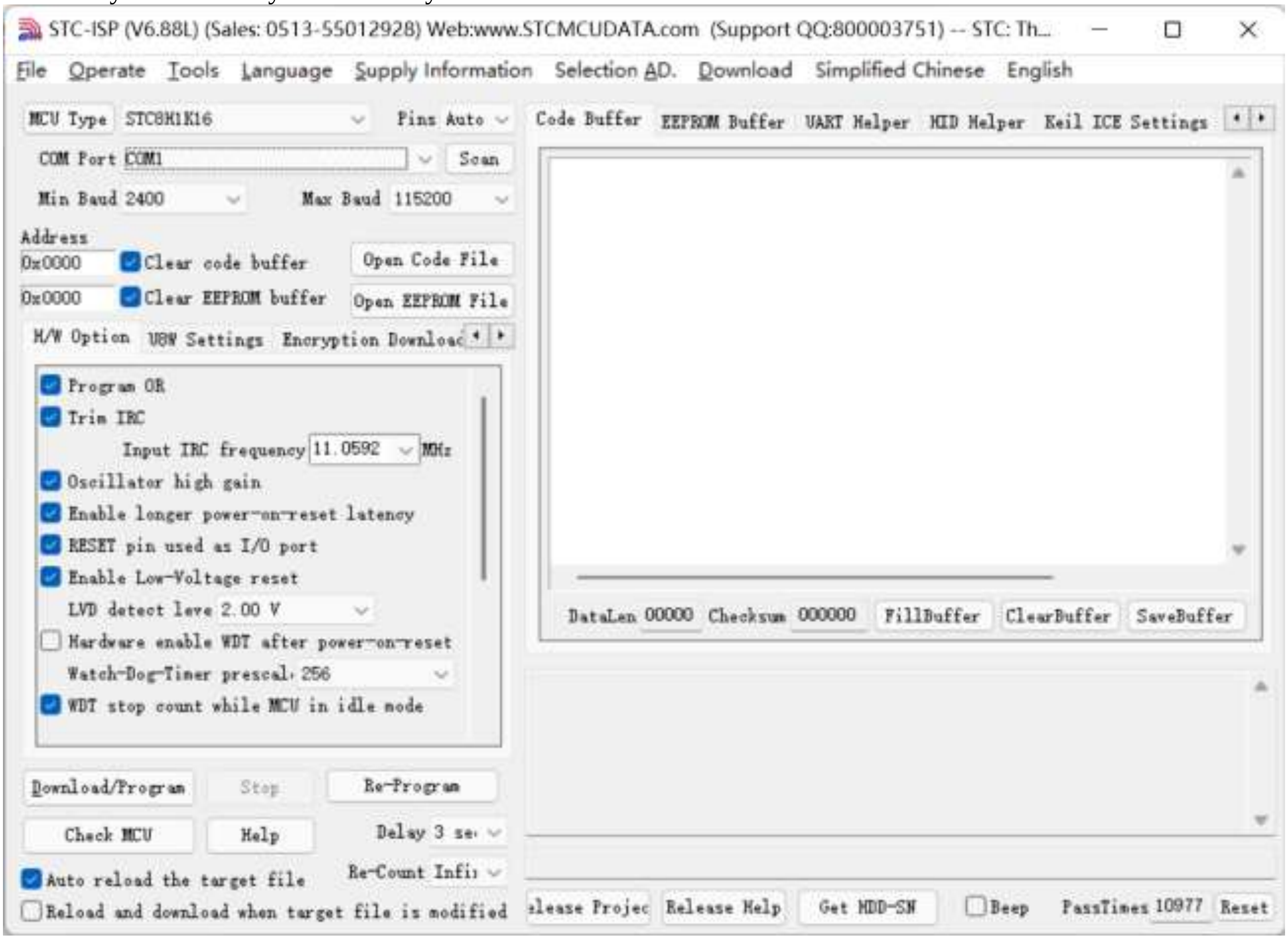
☐ Reload and download when target file is modified

DataLen 00000 Checksum 000000

☐ Beep PassTimes 10977

Installation Instructions in Windows 7 (32-bit)

Open the STC-ISP download software of V6.79 (or later). The download software will copy the driver files to the relevant system directory automatically.



STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...

MCU Type STC8H1K16 Pins Auto

COM Port COM1

Min Baud 2400 Max Baud 115200

Address

0x0000 ☑ Clear code buffer

0x0000 ☑ Clear EEPROM buffer

☑ Program OR

☑ Trim IRC

Input IRC frequency 11.0592 MHz

☑ Oscillator high gain

☑ Enable longer power-on-reset latency

☑ RESET pin used as I/O port

☑ Enable Low-Voltage reset

LVD detect leve 2.00 V

□ Hardware enable WDT after power-on-reset

Watch-Dog-Timer prescal 256

☑ WDT stop count while MCU in idle mode

Delay 3 se

Re-Count Infi

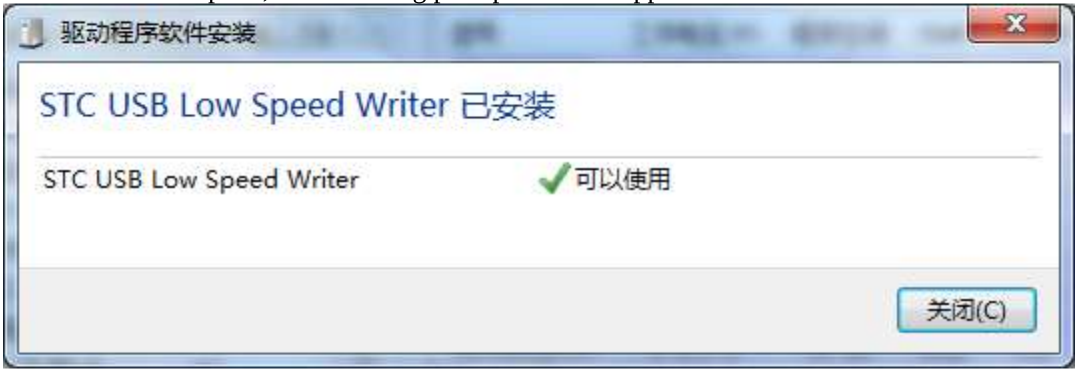
☑ Auto reload the target file

□ Reload and download when target file is modified

DataLen 00000 Checksum 000000

□ Beep PassTimes 10977

Plug in the USB device, and the system will install the driver automatically when it finds the device. After the installation is complete, the following prompt box will appear.

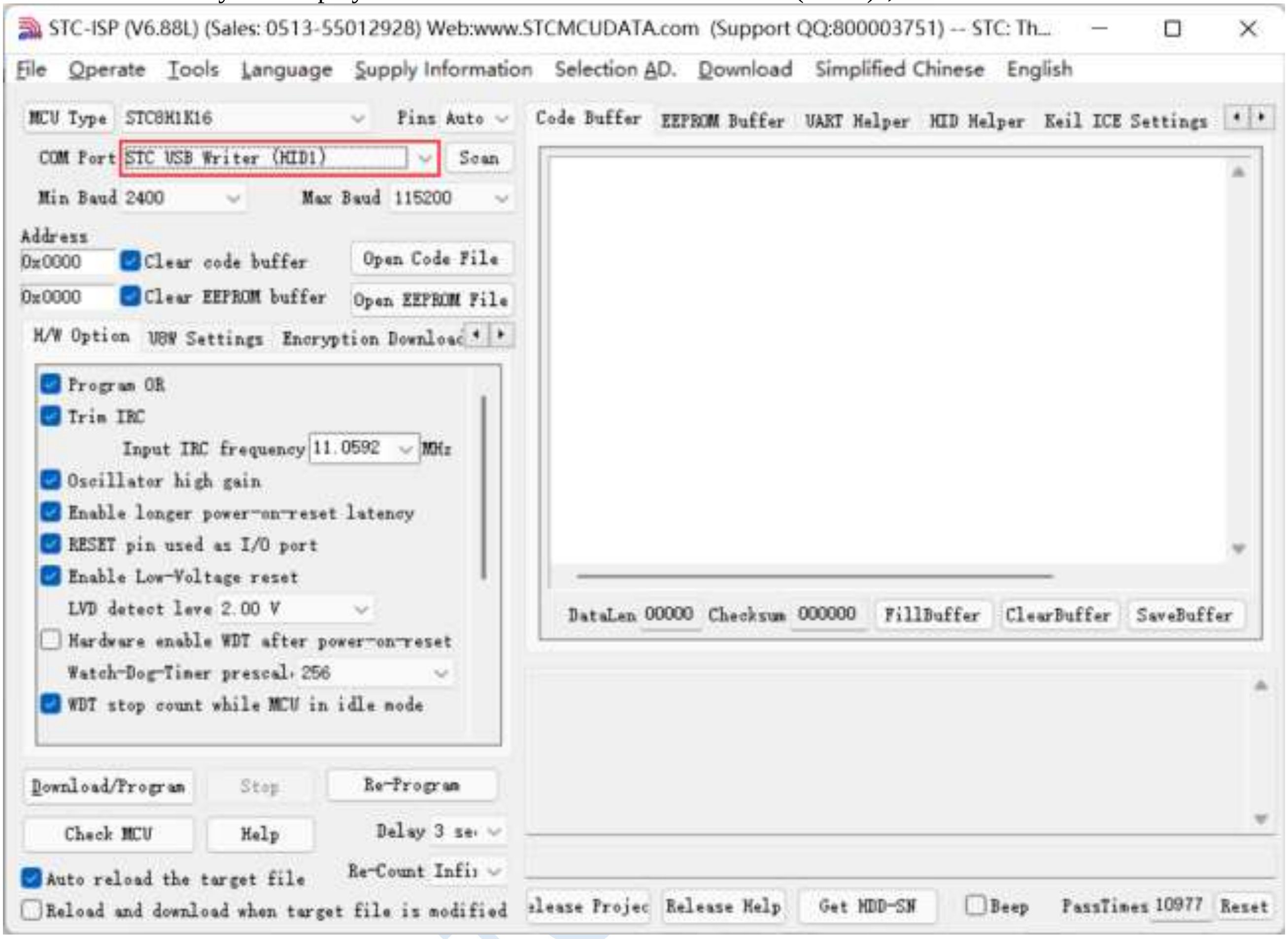


驱动程序软件安装

STC USB Low Speed Writer 已安装

STC USB Low Speed Writer ✓可以使用

Now, the serial port number list in the previously opened STC-ISP download software will select the inserted USB device automatically and display the device name as "STC USB Writer (HID1)", as shown below.


STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...

MCU Type STC8H1K16 Pins Auto

COM Port STC USB Writer (HID1)

Min Baud 2400 Max Baud 115200

Address

0x0000 ☑ Clear code buffer

0x0000 ☑ Clear EEPROM buffer

☑ Program OR

☑ Trim IRC

Input IRC frequency 11.0592 MHz

☑ Oscillator high gain

☑ Enable longer power-on-reset latency

☑ RESET pin used as I/O port

☑ Enable Low-Voltage reset

LVD detect leve 2.00 V

☐ Hardware enable WDT after power-on-reset

Watch-Dog-Timer prescal 256

☑ WDT stop count while MCU in idle mode

Delay 3 se

Re-Count Infi

☑ Auto reload the target file

☐ Reload and download when target file is modified

DataLen 00000 Checksum 000000

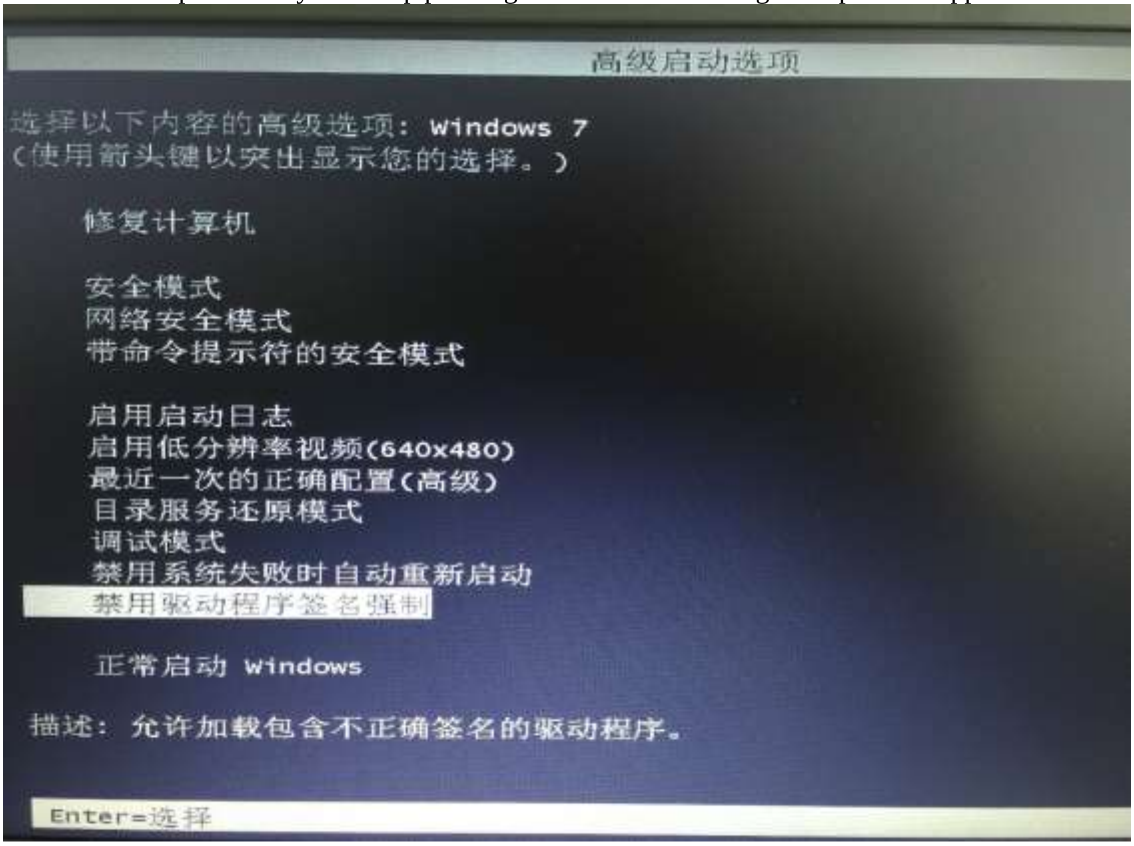
☐ Beep PassTimes 10977

Note: If the system does not install the driver automatically in Windows 7, please refer to the installation method of Windows 8 (32-bit) for the driver installation method.

Installation Instructions in Windows 7 (64-bit)

By default, the driver without digital signature cannot be successfully installed in Windows 7 64-bit operating system. So, you need to follow the steps below before installing the STC-USB driver, skip the digital signature temporarily, and the installation will be successful.

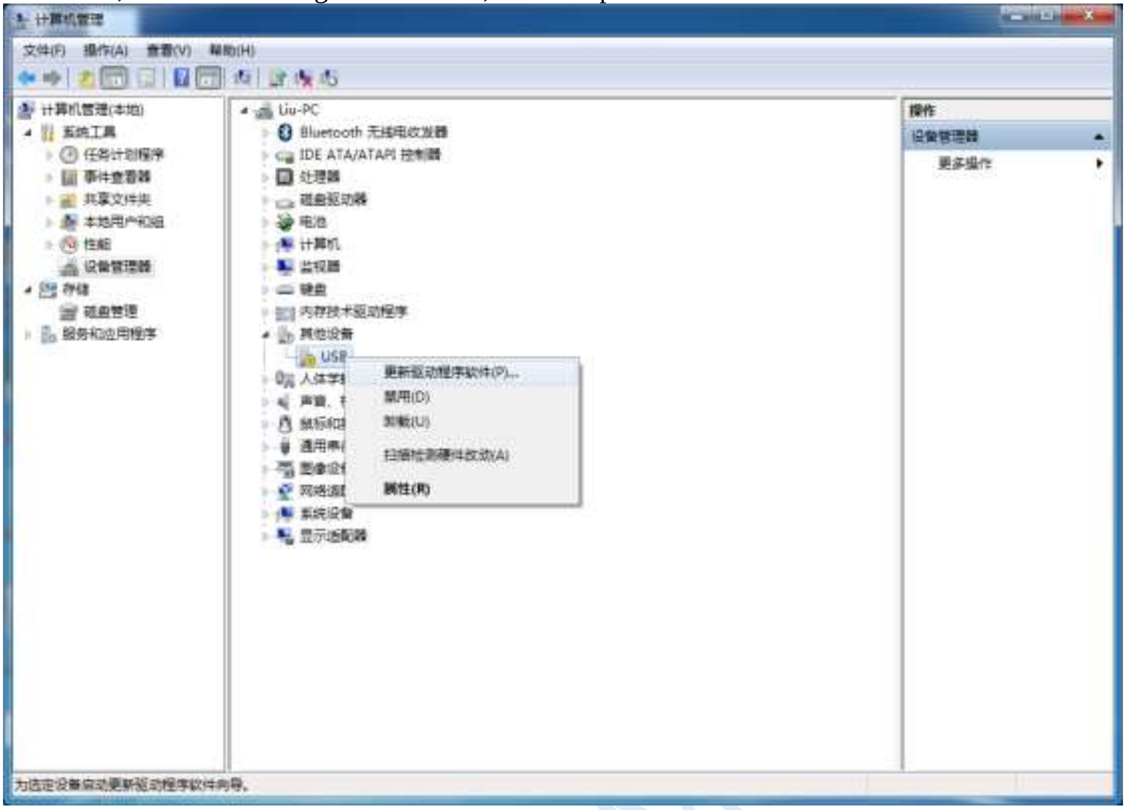
Restart the computer firstly and keep pressing F8 until the following startup screen appears.



```
高级启动选项

选择以下内容的高级选项：Windows 7
（使用箭头键以突出显示您的选择。）

    修复计算机

    安全模式
    网络安全模式
    带命令提示符的安全模式

    启用启动日志
    启用低分辨率视频(640x480)
    最近一次的正确配置(高级)
    目录服务还原模式
    调试模式
    禁用系统失败时自动重新启动
    禁用驱动程序签名强制

    正常启动 Windows

描述：允许加载包含不正确签名的驱动程序。

Enter=选择
```

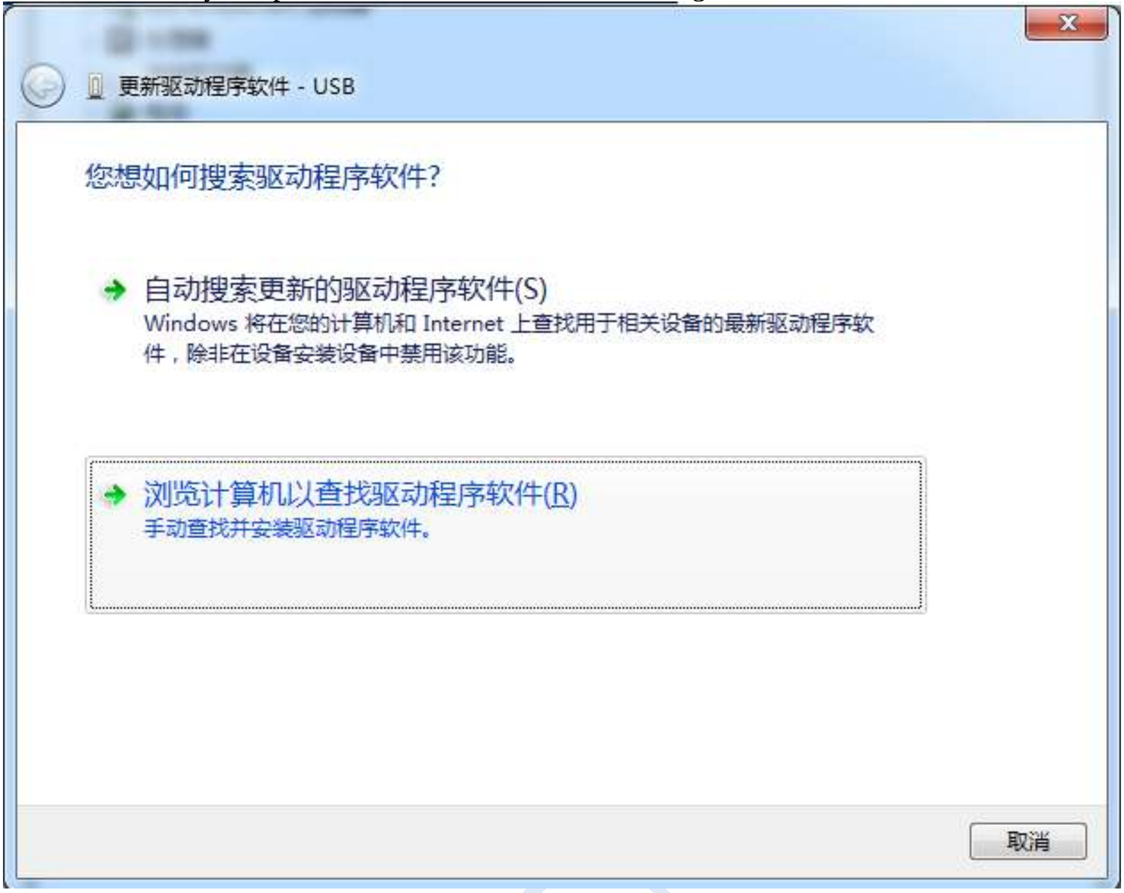
Select ‘Disable Driver Signature Enforcement’. The digital signature verification function is temporarily turn off after startup.

Plug in the USB device and open the Device Manager. Find the USB device with a yellow exclamation mark in the device list, in the device's right-click menu, select "Update Driver Software".



为选定设备启动更新驱动程序软件向导。

Select "Browse my computer for driver software" in the dialog below.



您想如何搜索驱动程序软件？

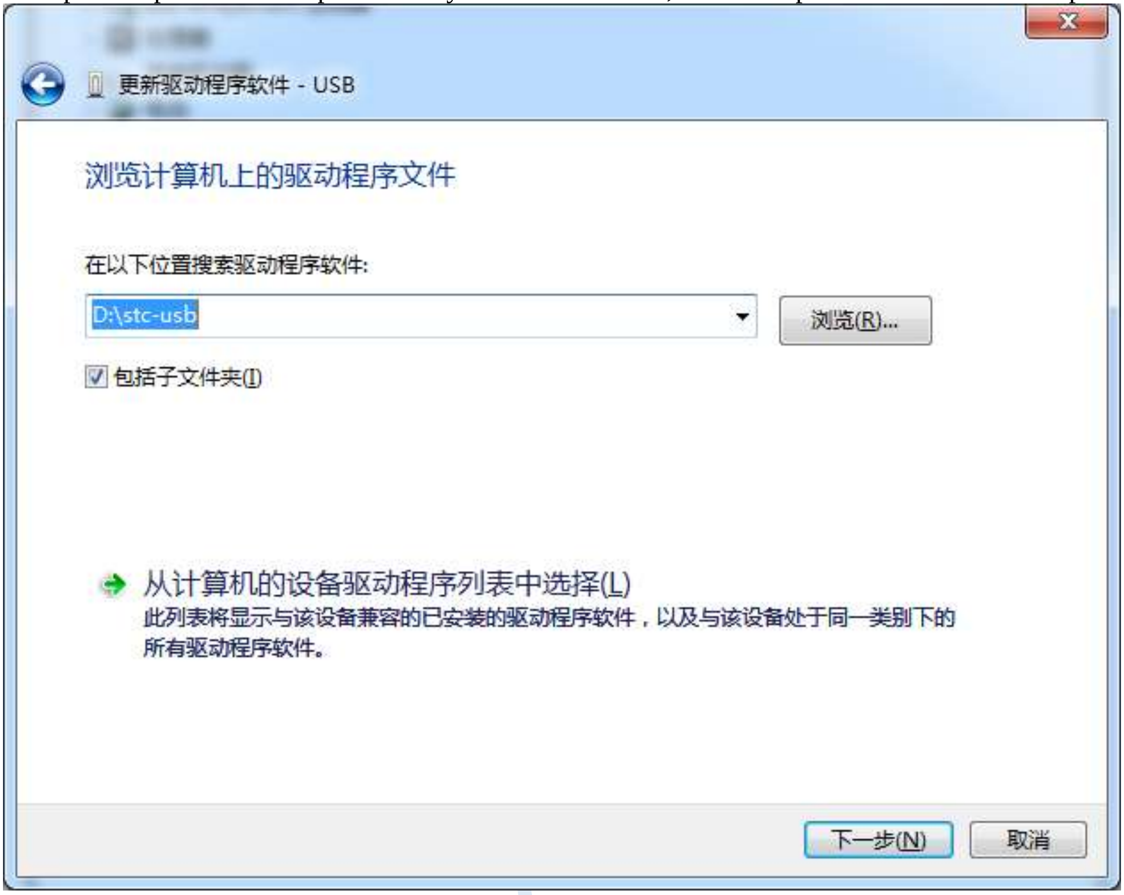
自动搜索更新的驱动程序软件(S)

Windows 将在您的计算机和 Internet 上查找用于相关设备的最新驱动程序软件，除非在设备安装设备中禁用该功能。

浏览计算机以查找驱动程序软件(R)

手动查找并安装驱动程序软件。

Click the "Browse" button in the dialog below to find the directory of the previous STC-USB driver stored (for example: the previous example directory is "D:\STC-USB", locate the path to the actual decompression directory).



浏览计算机上的驱动程序文件

在以下位置搜索驱动程序软件:

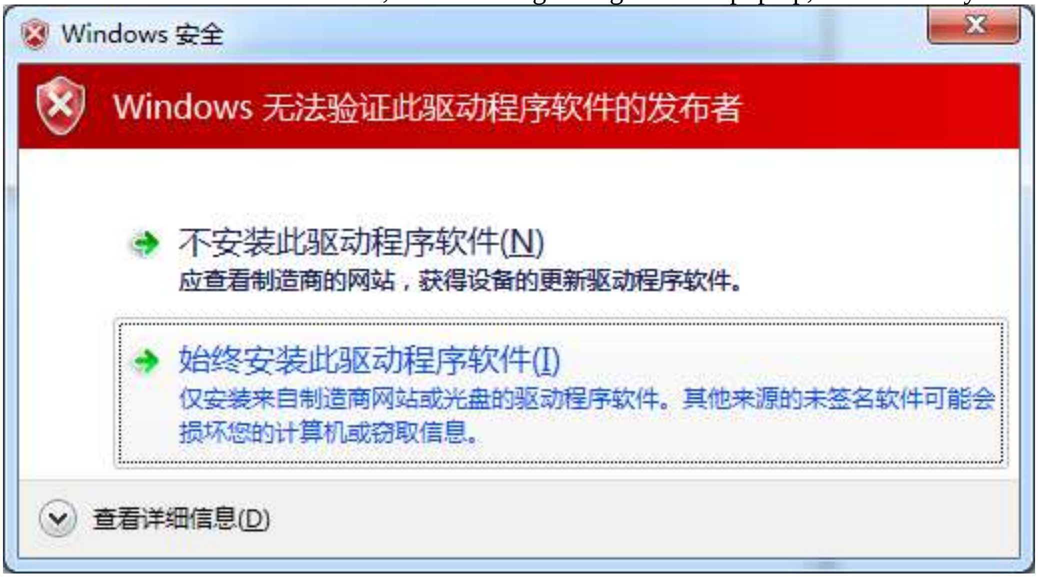
D:\stc-usb

☑ 包括子文件夹(I)

从计算机的设备驱动程序列表中选择(L)

此列表将显示与该设备兼容的已安装的驱动程序软件，以及与该设备处于同一类别下的所有驱动程序软件。

When the driver installation starts, the following dialog box will pop up, select "Always install this driver software".



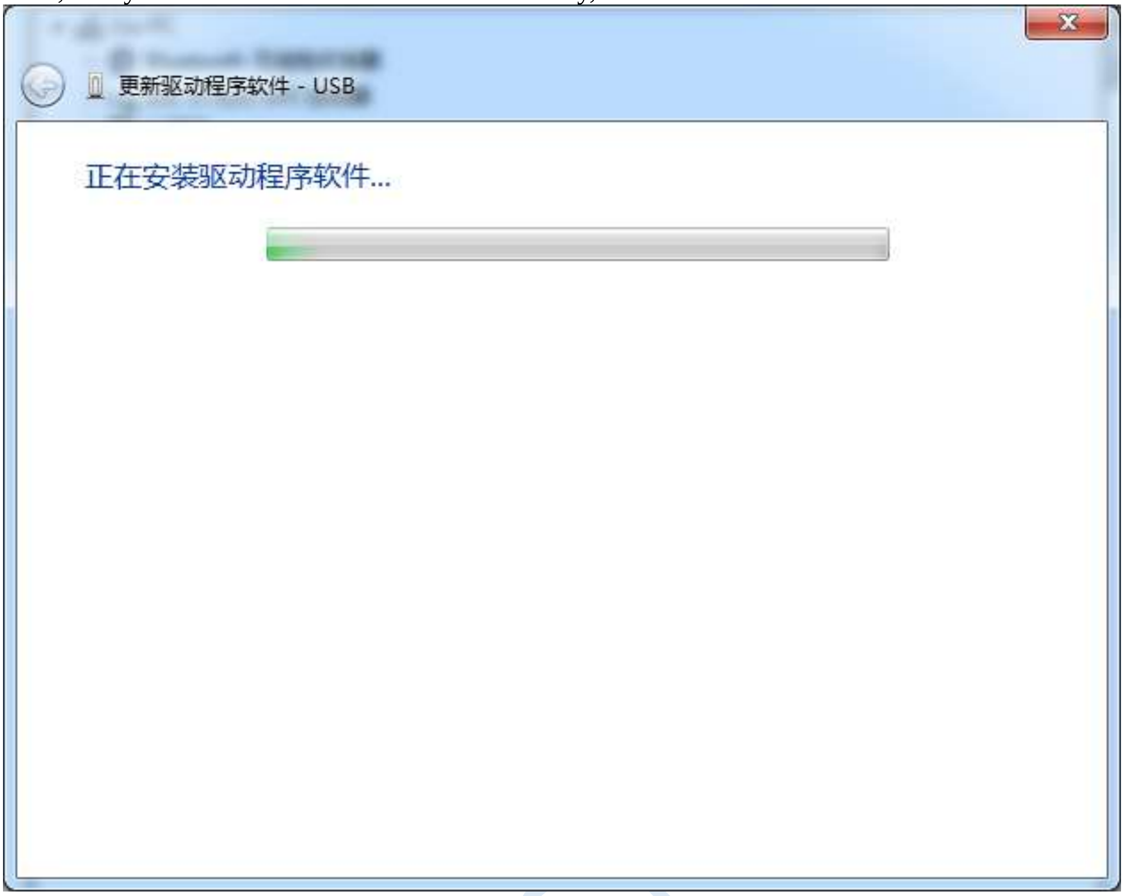
Windows 安全

Windows 无法验证此驱动程序软件的发布者

不安装此驱动程序软件(N)
应查看制造商的网站，获得设备的更新驱动程序软件。

始终安装此驱动程序软件(I)
仅安装来自制造商网站或光盘的驱动程序软件。其他来源的未签名软件可能会损坏您的计算机或窃取信息。

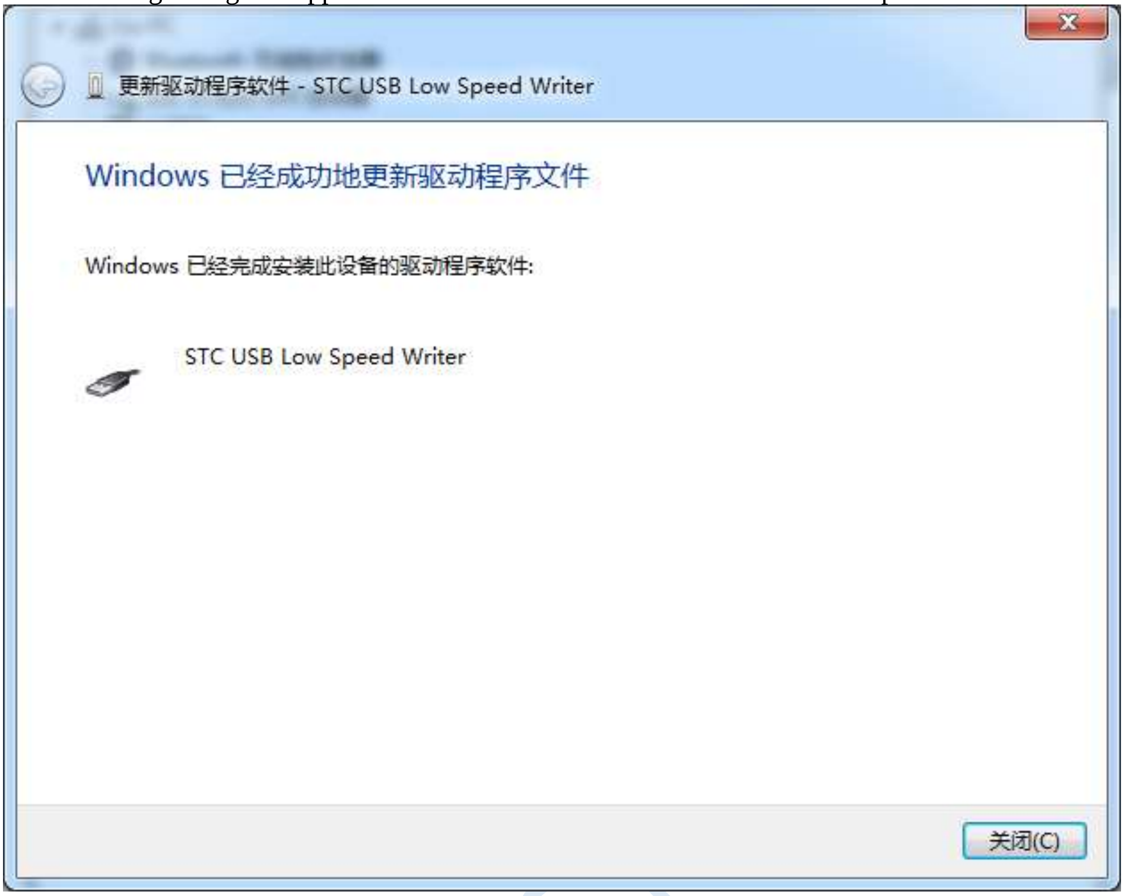
Next, the system will install the driver automatically, as shown below.



更新驱动程序软件 - USB

正在安装驱动程序软件...

The following dialog box appears to indicate that the driver installation is complete.



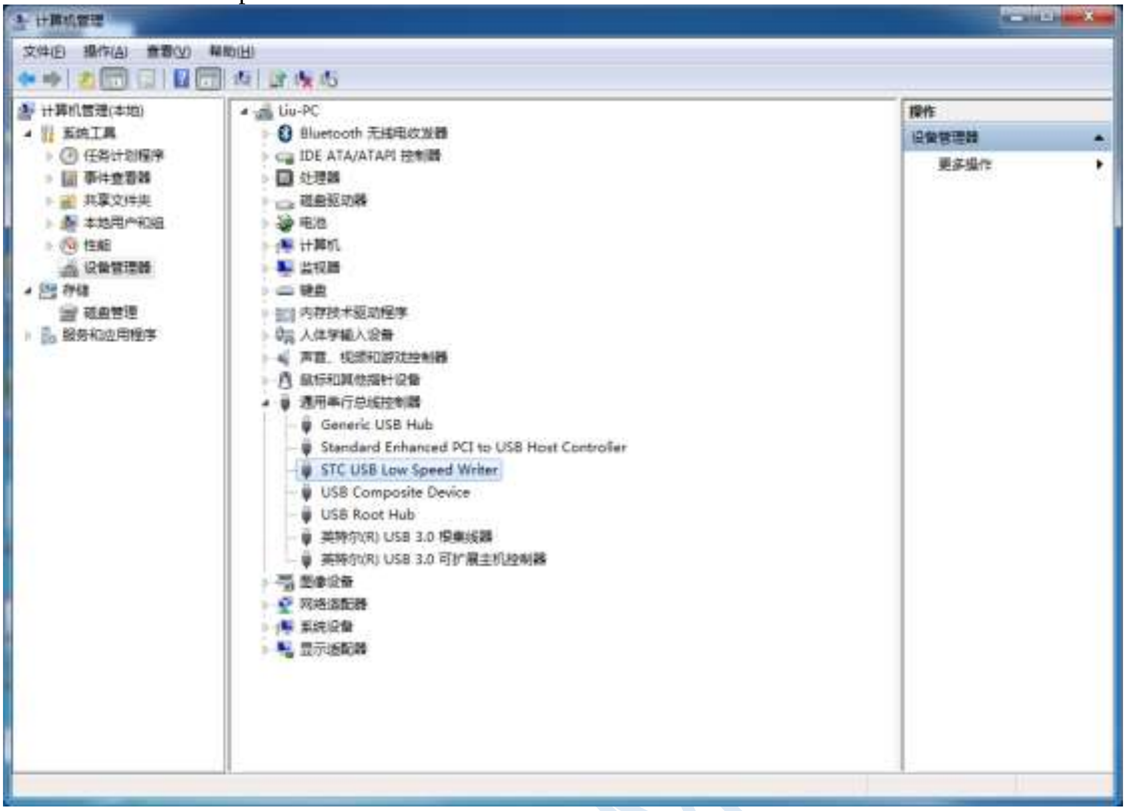
更新驱动程序软件 - STC USB Low Speed Writer

Windows 已经成功地更新驱动程序文件

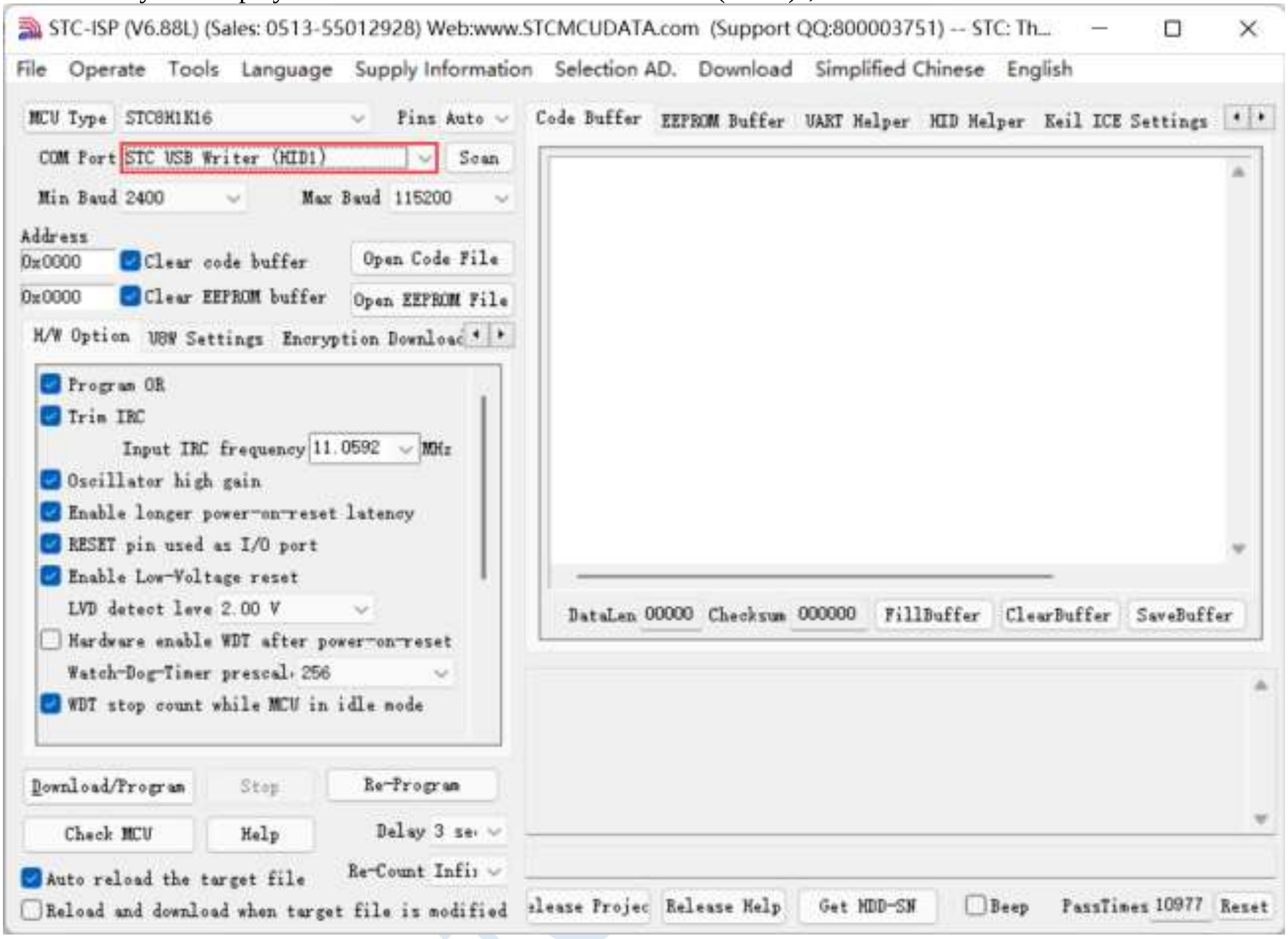
Windows 已经完成安装此设备的驱动程序软件:

STC USB Low Speed Writer

Now in the device manager, the device with the yellow exclamation mark before will be displayed as the device name of "STC USB Low Speed Writer".



The serial number list in the previously downloaded STC-ISP download software will select the inserted USB device automatically and display the device name as "STC USB Writer (HID1)", as shown below.



STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...

MCU Type STC8H1K16

Pins Auto

COM Port STC USB Writer (HID1)

Min Baud 2400

Max Baud 115200

Address

0x0000 ☑ Clear code buffer

0x0000 ☑ Clear EEPROM buffer

☑ Program OR

☑ Trim IRC

Input IRC frequency 11.0592 MHz

☑ Oscillator high gain

☑ Enable longer power-on-reset latency

☑ RESET pin used as I/O port

☑ Enable Low-Voltage reset

LVD detect leve 2.00 V

☐ Hardware enable WDT after power-on-reset

Watch-Dog-Timer prescal 256

☑ WDT stop count while MCU in idle mode

Delay 3 se

Re-Count Infi

☑ Auto reload the target file

☐ Reload and download when target file is modified

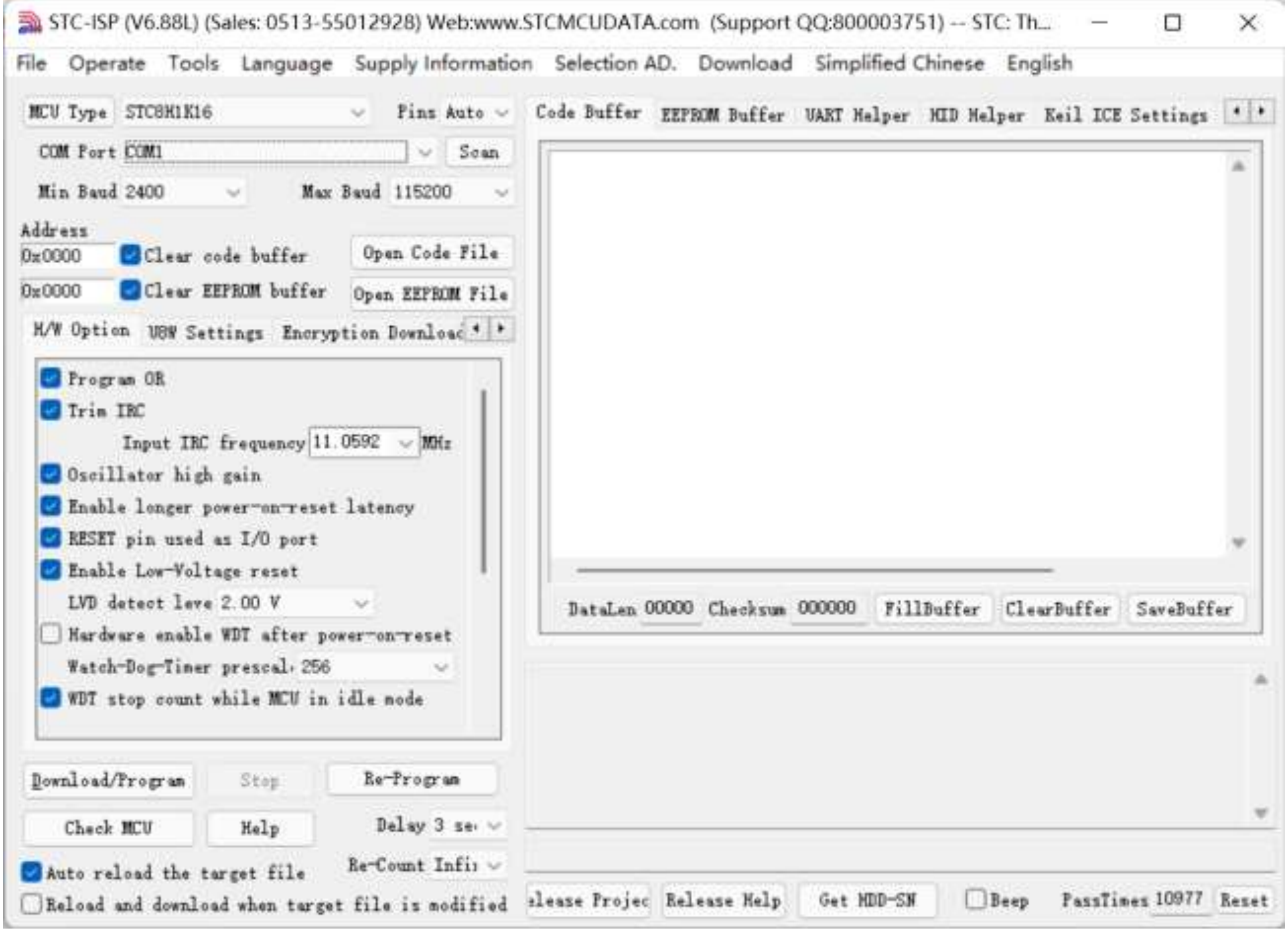
DataLen 00000 Checksum 000000

☐ Beep

PassTimes 10977

Installation Instructions in Windows 8 (32-bit)

Open the STC-ISP download software of V6.79 (or newer version) (Due to permission reasons, downloading the software in Windows 8 will not copy the driver files to the relevant system directory. It requires manual installation by the user. Firstly, download "stc-isp-15xx-v6.79.zip" (or newer version) from the STC official website, and decompress it to the local disk after downloading, then the STC-USB driver file will also be decompressed to the "STC-USB Driver" folder of the current folder. (For example, decompress the downloaded compressed file" stc-isp-15xx-v6.79.zip "to" F:\", then the STC-USB driver is in the" F:\ STC-USB Driver "directory))



STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...

MCU Type STC8H1K16

Pins Auto

COM Port COM1

Min Baud 2400

Max Baud 115200

Address

0x0000 ☑ Clear code buffer

0x0000 ☑ Clear EEPROM buffer

☑ Program OR

☑ Trim IRC

Input IRC frequency 11.0592 MHz

☑ Oscillator high gain

☑ Enable longer power-on-reset latency

☑ RESET pin used as I/O port

☑ Enable Low-Voltage reset

LVD detect leve 2.00 V

☐ Hardware enable WDT after power-on-reset

Watch-Dog-Timer prescal. 256

☑ WDT stop count while MCU in idle mode

Delay 3 se

Re-Count Infi

☑ Auto reload the target file

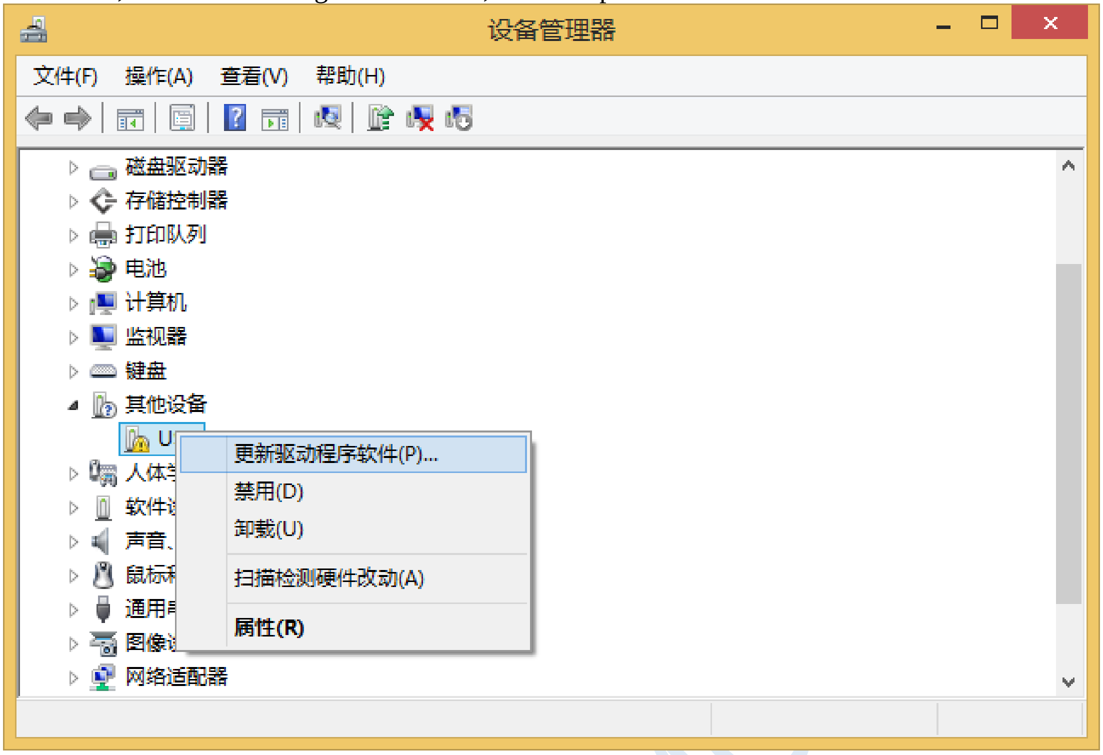
☐ Reload and download when target file is modified

DataLen 00000 Checksum 000000

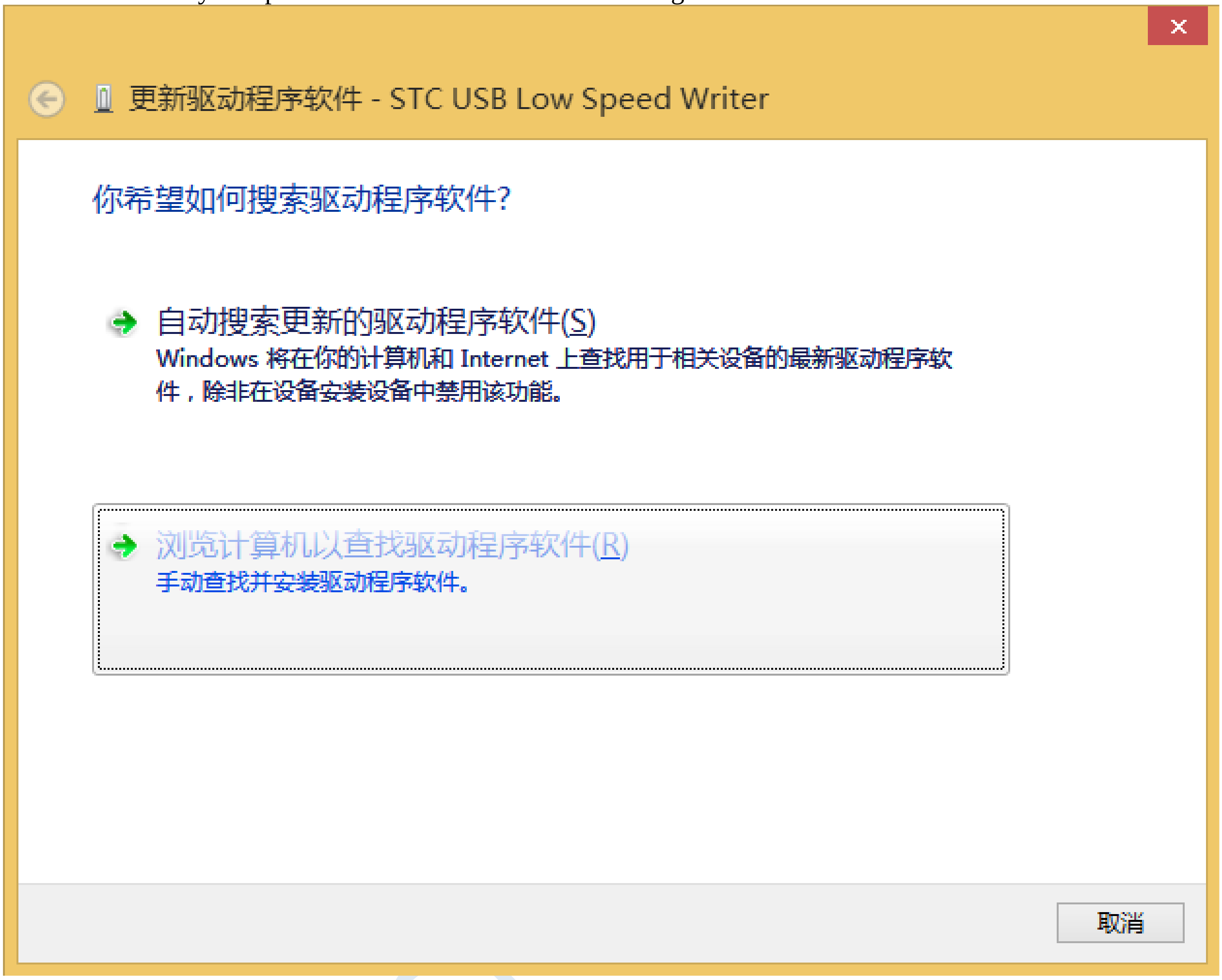
☐ Beep

PassTimes 10977

Plug in the USB device and open the Device Manager. Find the USB device with a yellow exclamation mark in the device list, in the device's right-click menu, select "Update Driver Software".



Select "Browse my computer for driver software" in the dialog below.



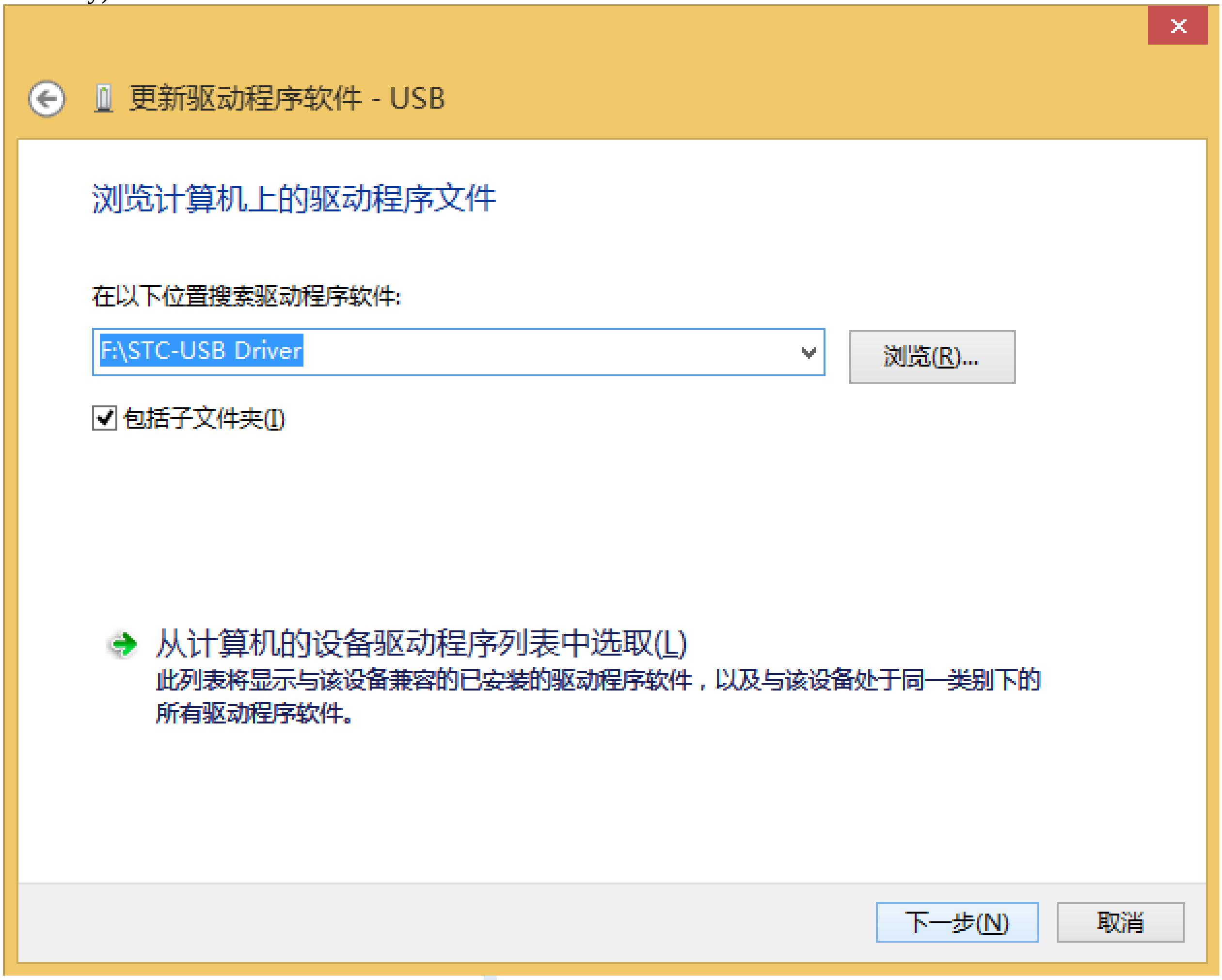
更新驱动程序软件 - STC USB Low Speed Writer

你希望如何搜索驱动程序软件?

自动搜索更新的驱动程序软件(S)
Windows 将在你的计算机和 Internet 上查找用于相关设备的最新驱动程序软件，除非在设备安装设备中禁用该功能。

浏览计算机以查找驱动程序软件(R)
手动查找并安装驱动程序软件。

Click the "Browse" button in the dialog below to find the directory where the STC-USB driver was stored (for example: the previous example directory is "F: \ STC-USB Driver", locate the path to the actual decompression directory) .



更新驱动程序软件 - USB

浏览计算机上的驱动程序文件

在以下位置搜索驱动程序软件:

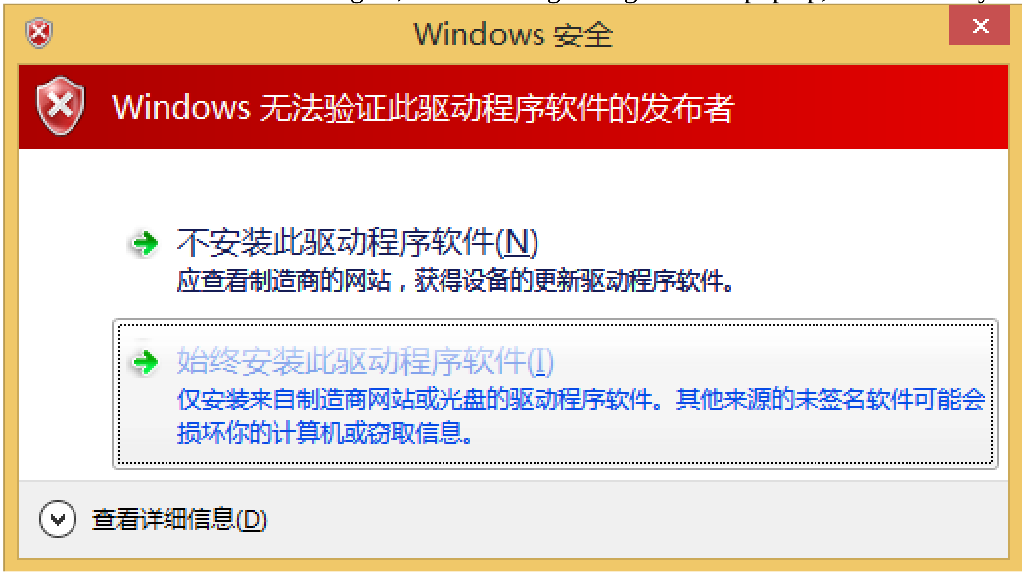
F:\STC-USB Driver

☑ 包括子文件夹(I)

从计算机的设备驱动程序列表中选取(L)

此列表将显示与该设备兼容的已安装的驱动程序软件，以及与该设备处于同一类别下的所有驱动程序软件。

When the driver installation begins, the following dialog box will pop up, select "Always install this driver software".


Windows 安全

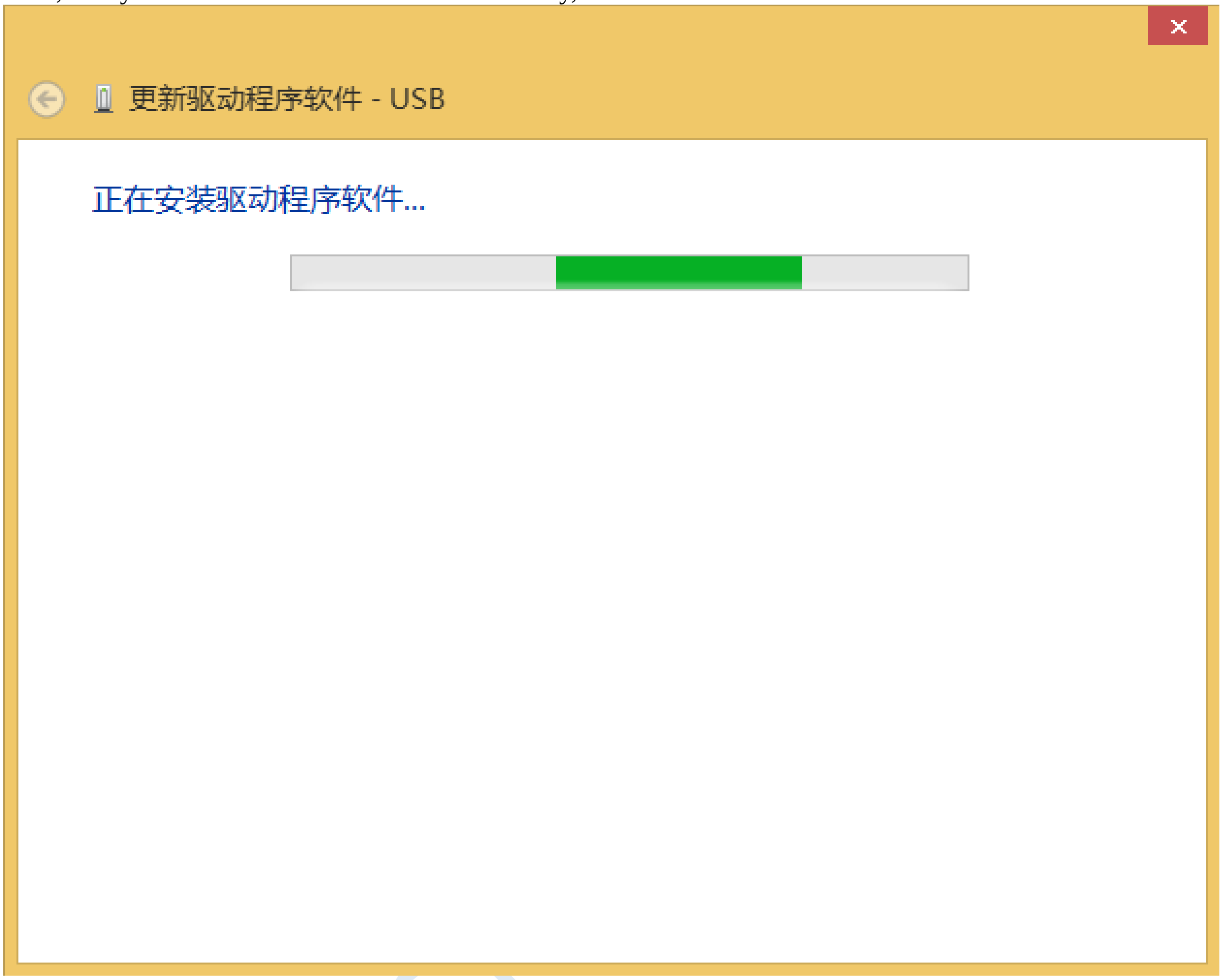
Windows 无法验证此驱动程序软件的发布者

不安装此驱动程序软件(N)
应查看制造商的网站，获得设备的更新驱动程序软件。

始终安装此驱动程序软件(I)
仅安装来自制造商网站或光盘的驱动程序软件。其他来源的未签名软件可能会损坏你的计算机或窃取信息。

查看详细信息(D)

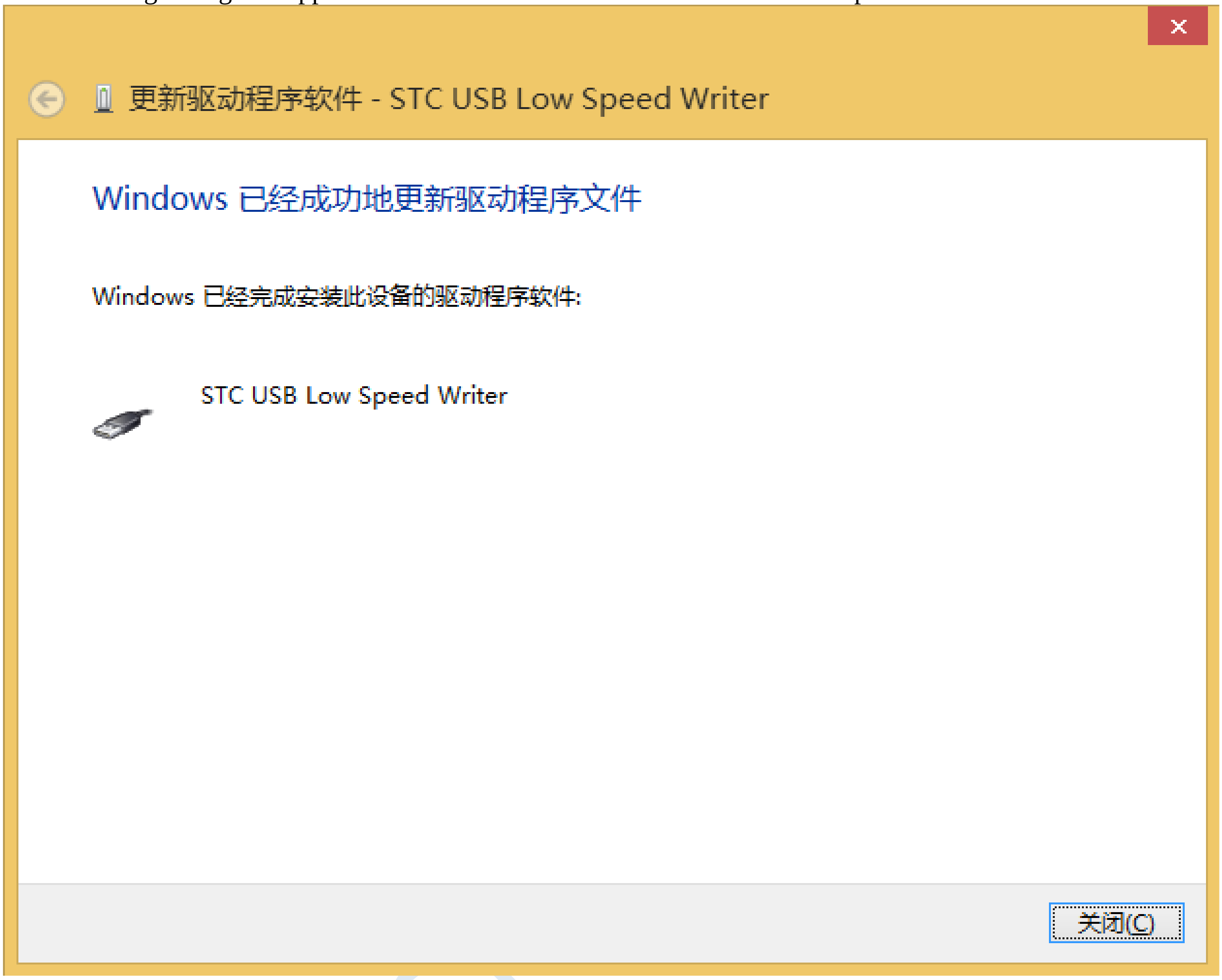
Next, the system will install the driver automatically, as shown below.



更新驱动程序软件 - USB

正在安装驱动程序软件...

The following dialog box appears to indicate that the driver installation is complete.

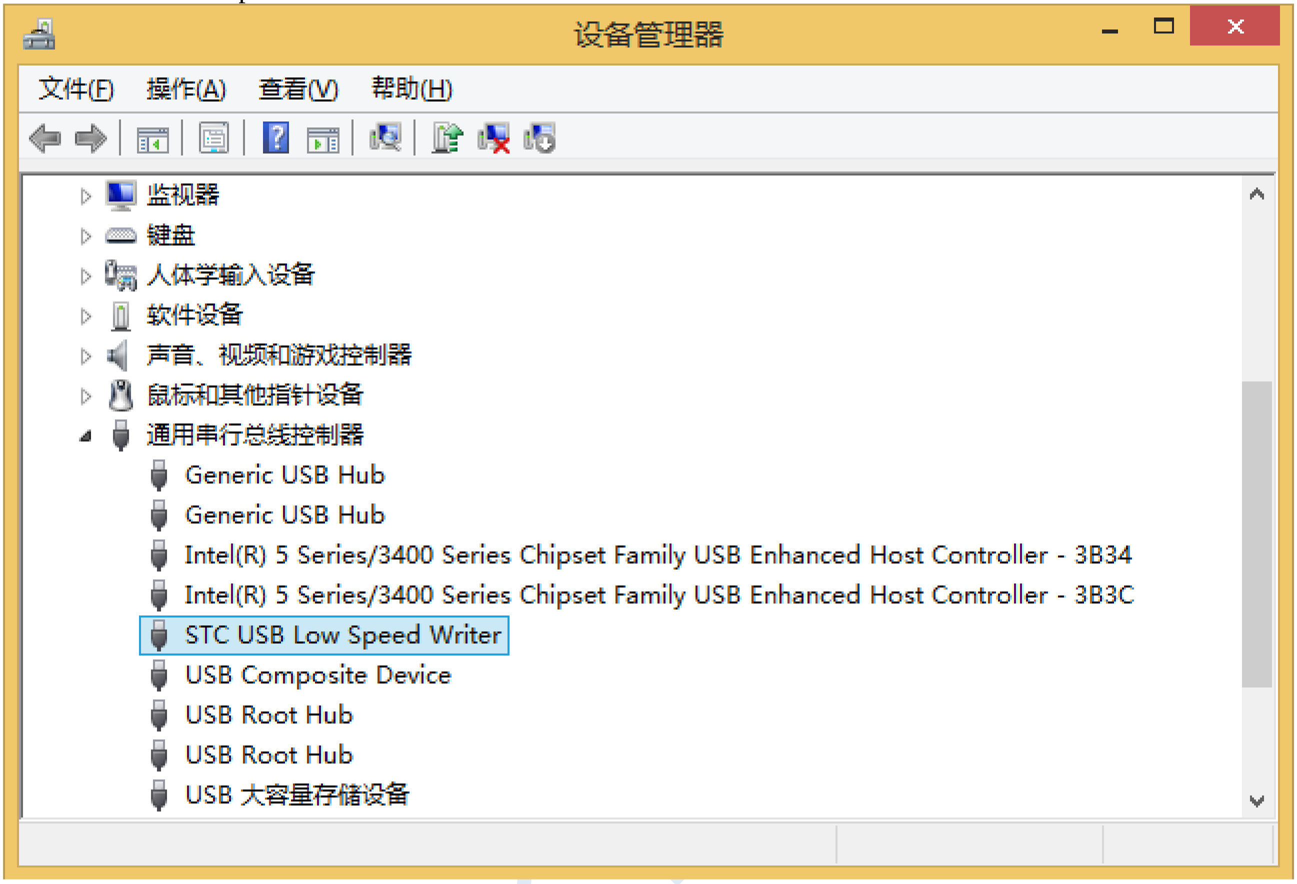


Windows 已经成功地更新驱动程序文件

Windows 已经完成安装此设备的驱动程序软件:

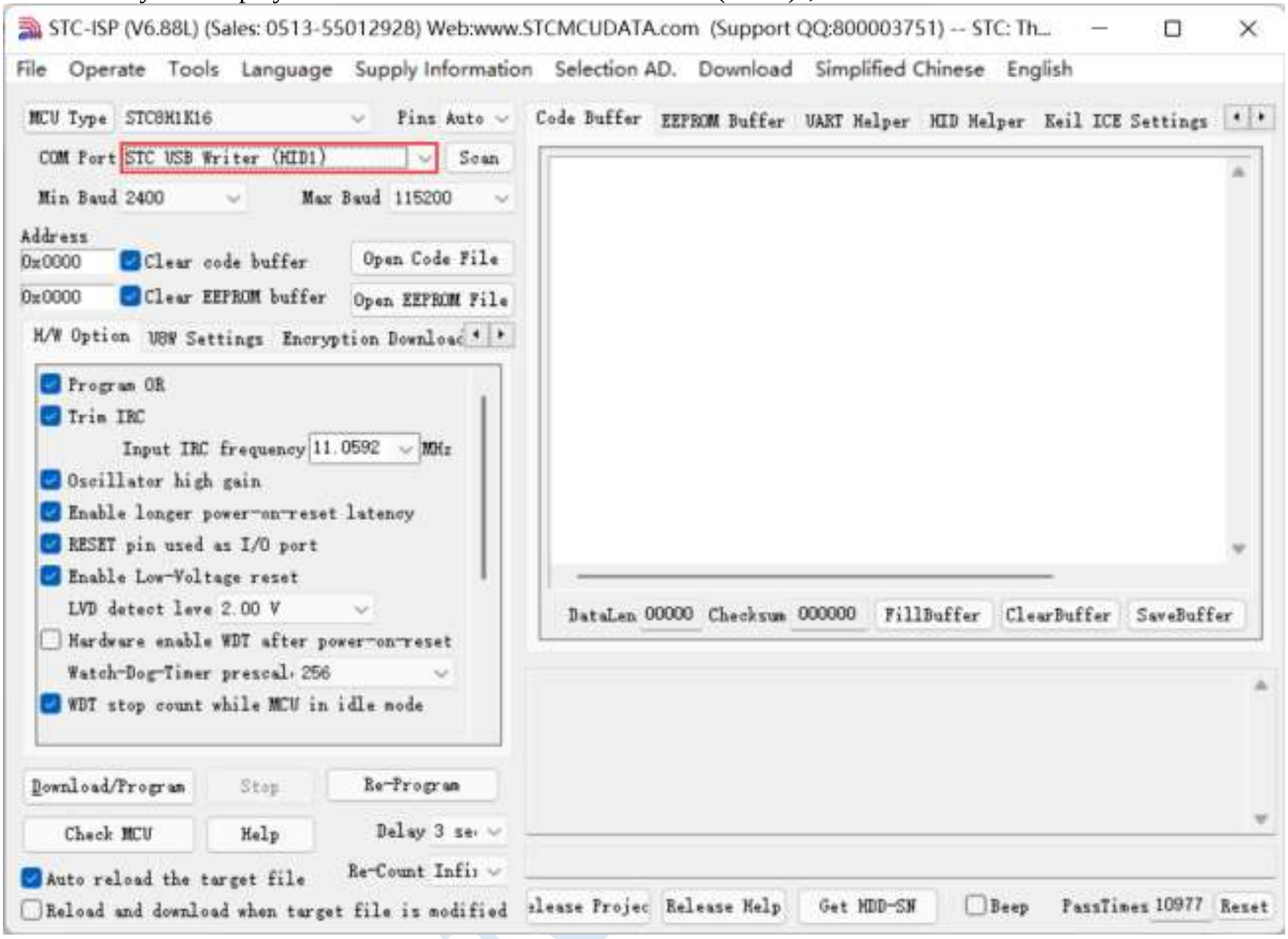
STC USB Low Speed Writer

Now in the device manager, the device with the yellow exclamation mark before will be displayed as the device name of "STC USB Low Speed Writer".



- 监视器
- 键盘
- 人体学输入设备
- 软件设备
- 声音、视频和游戏控制器
- 鼠标和其他指针设备
- 通用串行总线控制器
  - Generic USB Hub
  - Generic USB Hub
  - Intel(R) 5 Series/3400 Series Chipset Family USB Enhanced Host Controller - 3B34
  - Intel(R) 5 Series/3400 Series Chipset Family USB Enhanced Host Controller - 3B3C
  - STC USB Low Speed Writer
  - USB Composite Device
  - USB Root Hub
  - USB Root Hub
  - USB 大容量存储设备

The serial number list in the previously downloaded STC-ISP download software will select the inserted USB device automatically and display the device name as "STC USB Writer (HID1)", as shown below:



MCU Type STC8H1K16 Pins Auto

COM Port STC USB Writer (HID1)

Min Baud 2400 Max Baud 115200

Address

0x0000 ☑ Clear code buffer

0x0000 ☑ Clear EEPROM buffer

☑ Program OR

☑ Trim IRC

Input IRC frequency 11.0592 MHz

☑ Oscillator high gain

☑ Enable longer power-on-reset latency

☑ RESET pin used as I/O port

☑ Enable Low-Voltage reset

LVD detect leve 2.00 V

☐ Hardware enable WDT after power-on-reset

Watch-Dog-Timer prescal 256

☑ WDT stop count while MCU in idle mode

DataLen 00000 Checksum 000000

Delay 3 se

Re-Count Infi

☑ Auto reload the target file

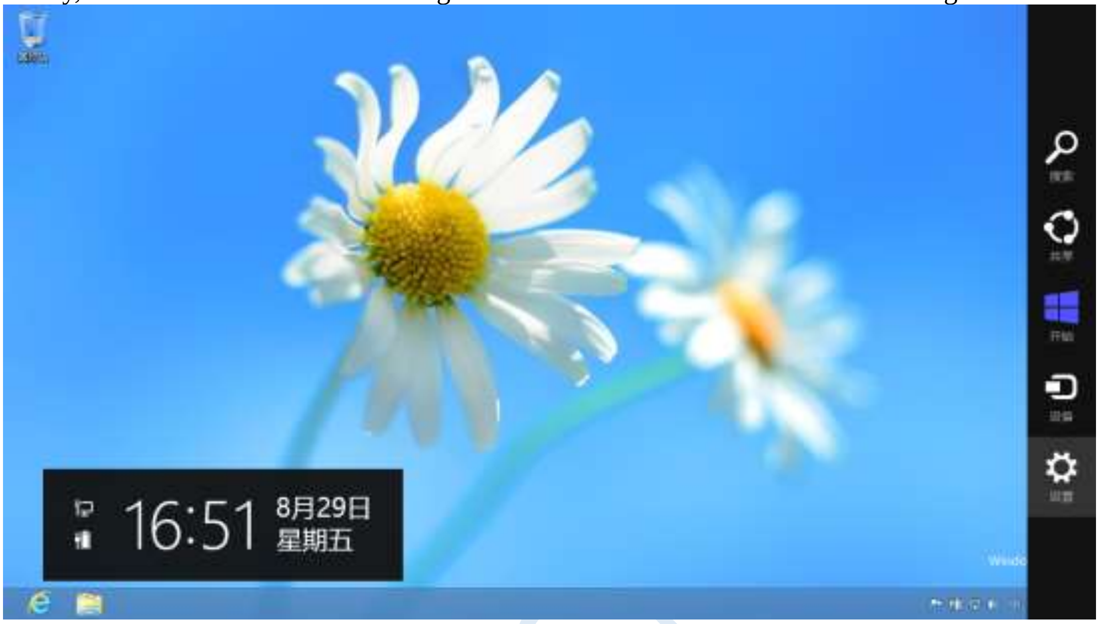
☐ Reload and download when target file is modified

☐ Beep PassTimes 10977

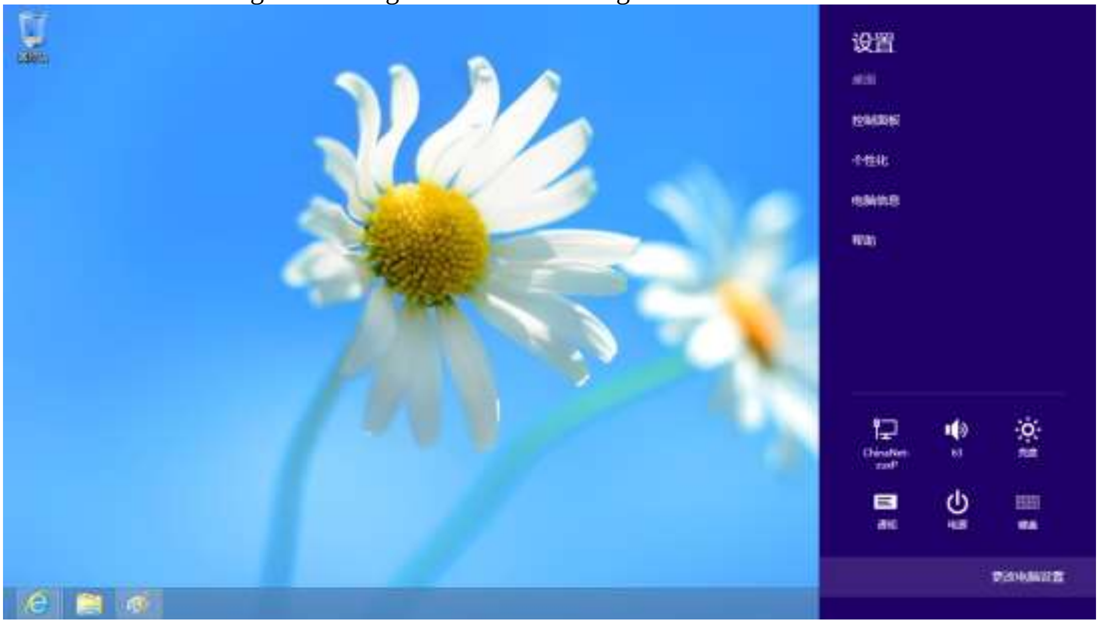
Installation Instructions in Windows 8 (64-bit)

By default, the driver without digital signature cannot be successfully installed in Windows 8 64-bit operating system. So, you need to follow the steps below before installing the STC-USB driver, skip the digital signature temporarily, and the installation will be successful.

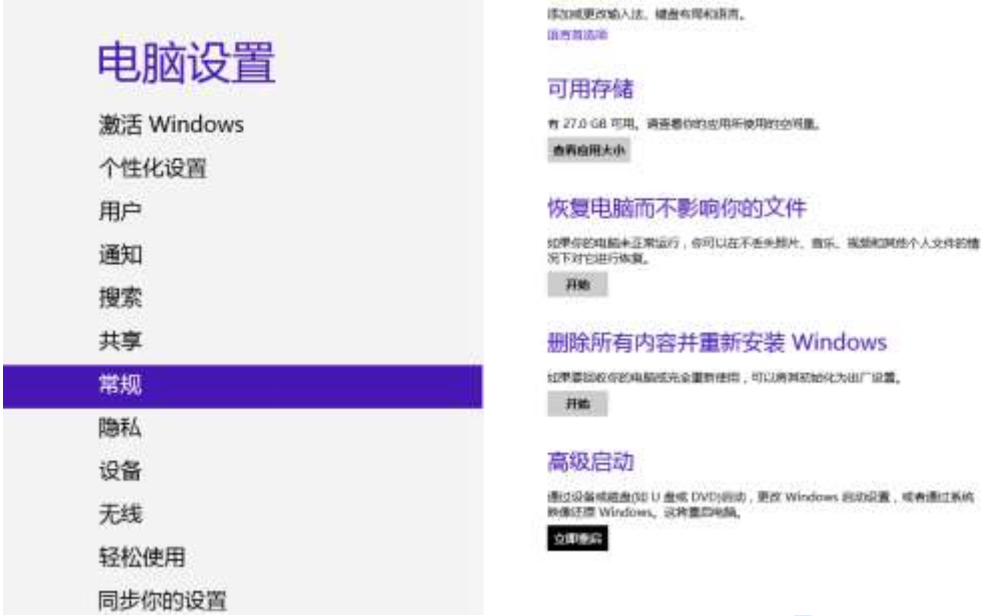
Firstly, move the mouse to the lower right corner of the screen and select the "Settings" button.



Then select the "Change PC settings" item in the settings window.



In the computer settings, select the "Start Now" button under the "Advanced Startup" item in the "General" property page.



添加或更改输入法、键盘布局和语言。

可用存储

有 27.0 GB 可用。请查看你的应用所使用的空间量。

恢复电脑而不影响你的文件

如果你的电脑未正常运行，你可以在不丢失照片、音乐、视频和其他个人文件的情况下对它进行恢复。

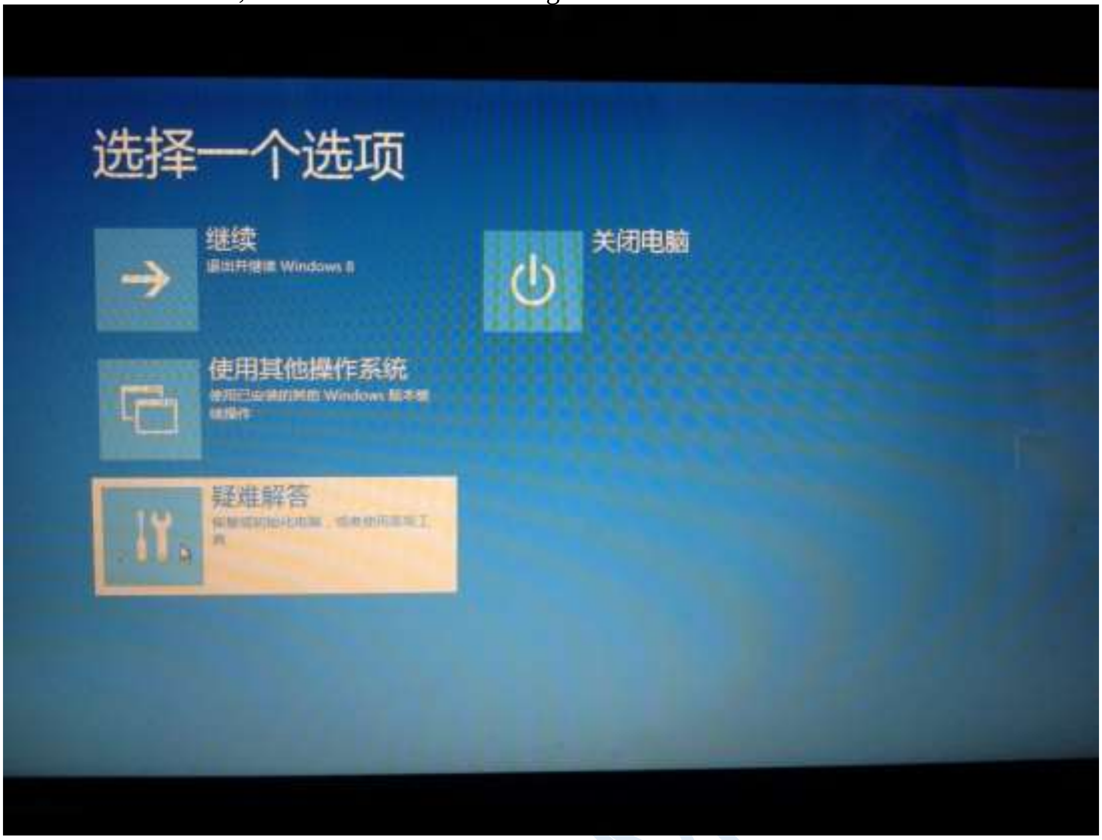
删除所有内容并重新安装 Windows

如果要回收你的电脑或完全重新使用，可以将其初始化为出厂设置。

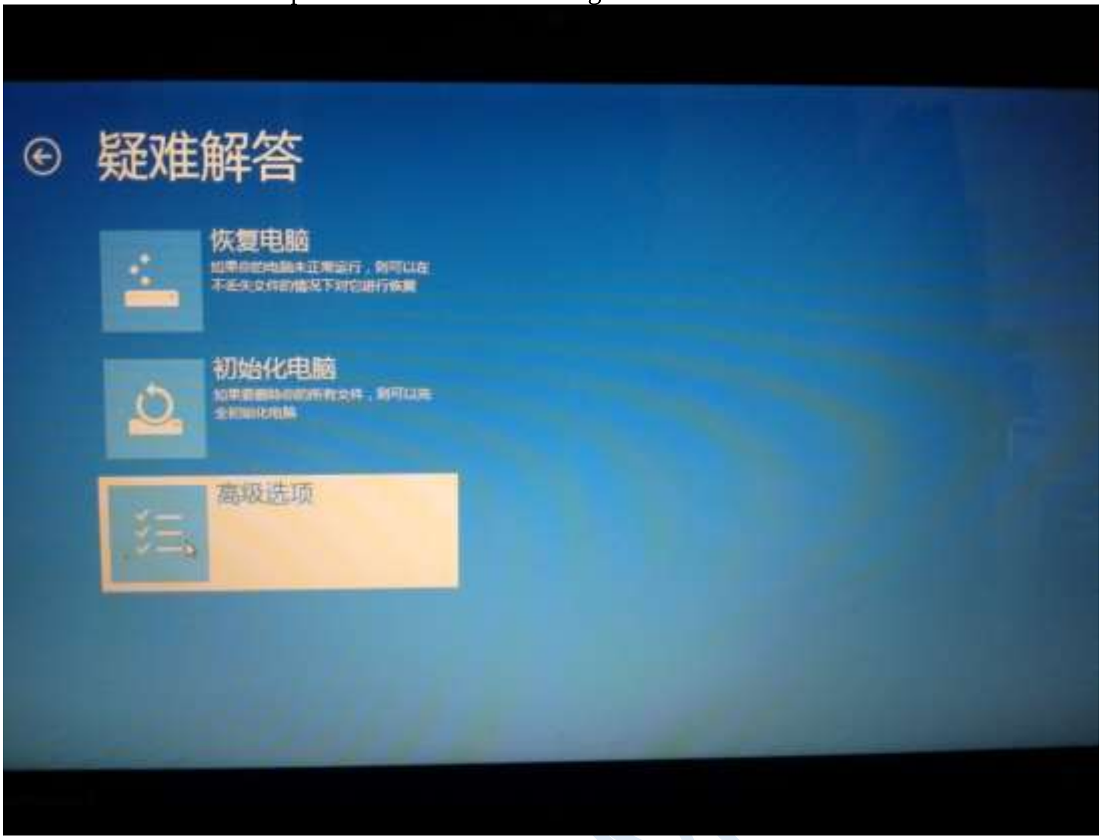
高级启动

通过设备或磁盘(如 U 盘或 DVD)启动，更改 Windows 启动设置，或者通过系统映像还原 Windows。这将重启电脑。

In the window below, select the "Troubleshooting" item.


选择一个选项
继续
退出并继续 Windows 8
关闭电脑
使用其他操作系统
使用已安装的其他 Windows 版本继续操作
疑难解答
恢复或初始化电脑，或者使用高级工具

Then select "Advanced Options" in "Troubleshooting".



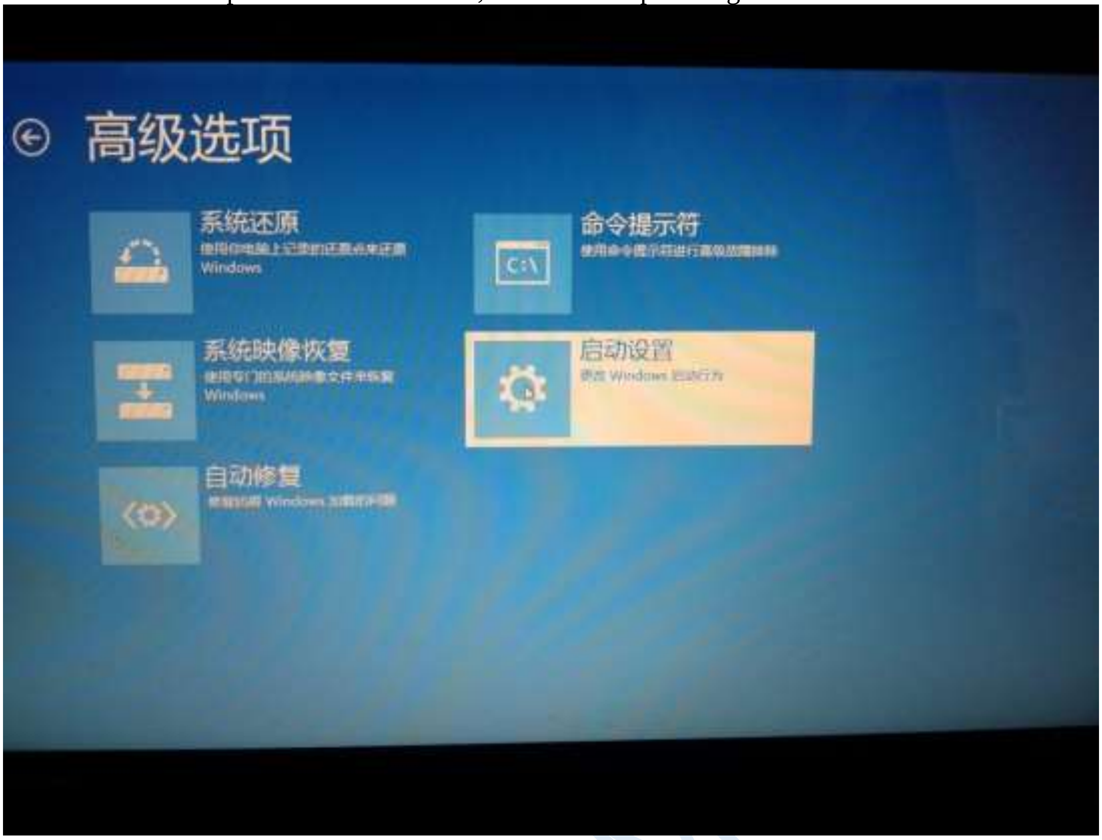
疑难解答

恢复电脑
如果你的电脑未正常运行，则可以在不丢失文件的情况下对它进行恢复

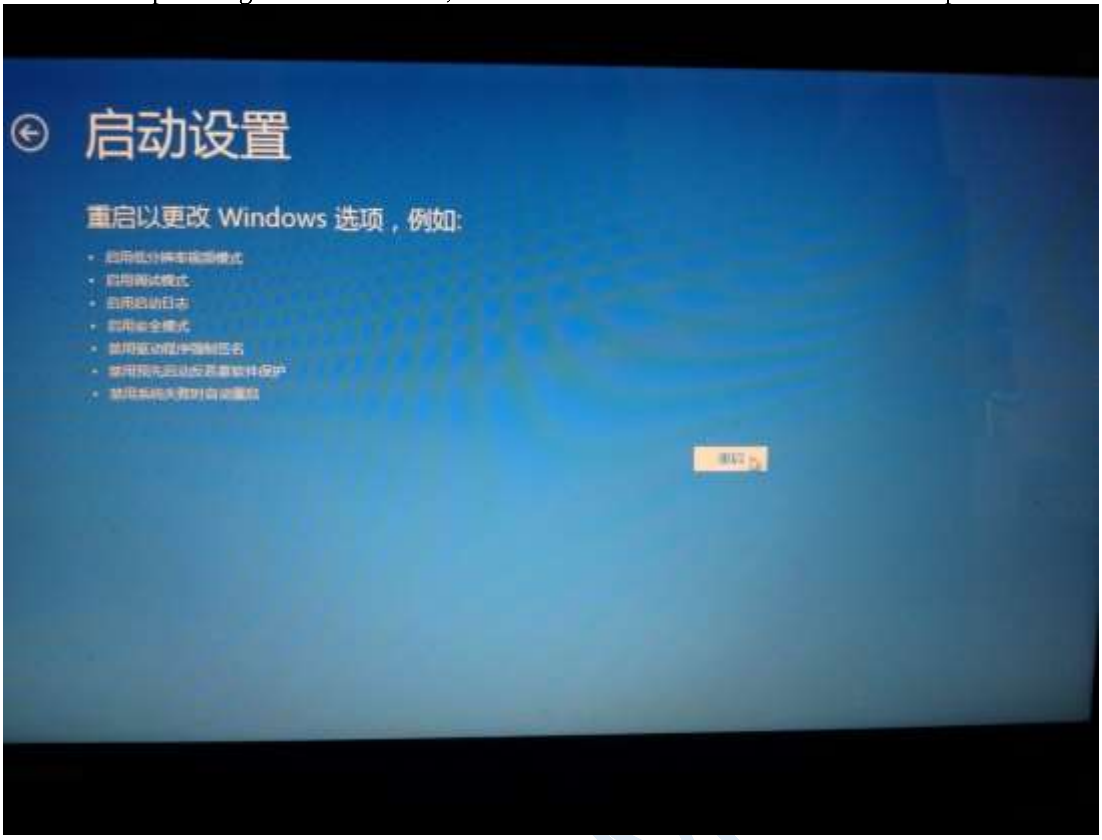
初始化电脑
如果要删除你的所有文件，则可以完全初始化电脑

高级选项

In the "Advanced Options" window below, select "Startup Settings".


高级选项
系统还原
使用你电脑上记录的还原点来还原 Windows
命令提示符
使用命令提示符进行高级故障排除
系统映像恢复
使用专门的系统映像文件来恢复 Windows
启动设置
更改 Windows 启动行为
自动修复
修复阻止 Windows 加载的问题
C:\

In the "Startup Settings" window below, click the "Restart" button to restart the computer.

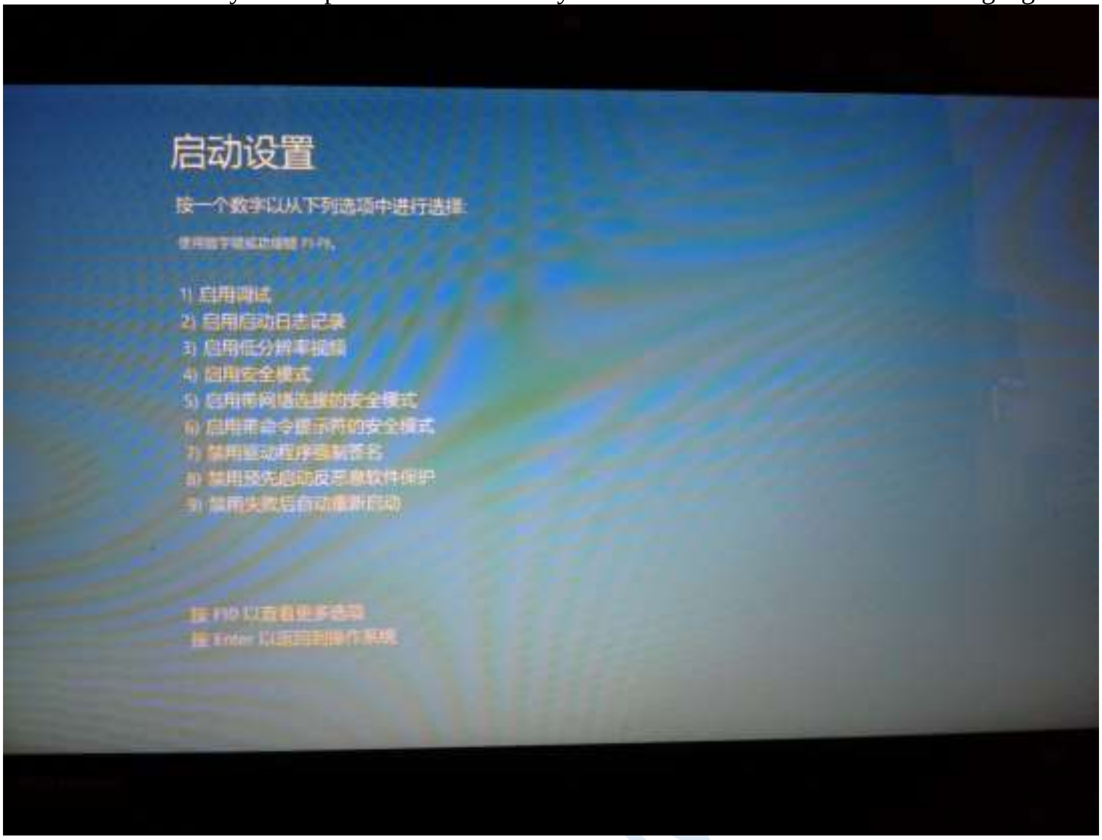


启动设置

重启以更改 Windows 选项，例如:

- 启用低分辨率视频模式
- 启用调试模式
- 启用启动日志
- 启用安全模式
- 禁用驱动程序强制签名
- 禁用预先启动反恶意软件保护
- 禁用系统失败时自动重启

After the computer restarts, it will enter the "Startup Settings" window automatically as shown in the figure below. Press the number key "7" or press the function key "F7" to select "Disable driver forcing signature" to start.



启动设置

按一个数字以从下列选项中进行选择:

使用数字键或功能键 F1-F9。

1) 启用调试
2) 启用启动日志记录
3) 启用低分辨率视频
4) 启用安全模式
5) 启用带网络连接的安全模式
6) 启用带命令提示符的安全模式
7) 禁用驱动程序强制签名
8) 禁用预先启动反恶意软件保护
9) 禁用失败后自动重新启动

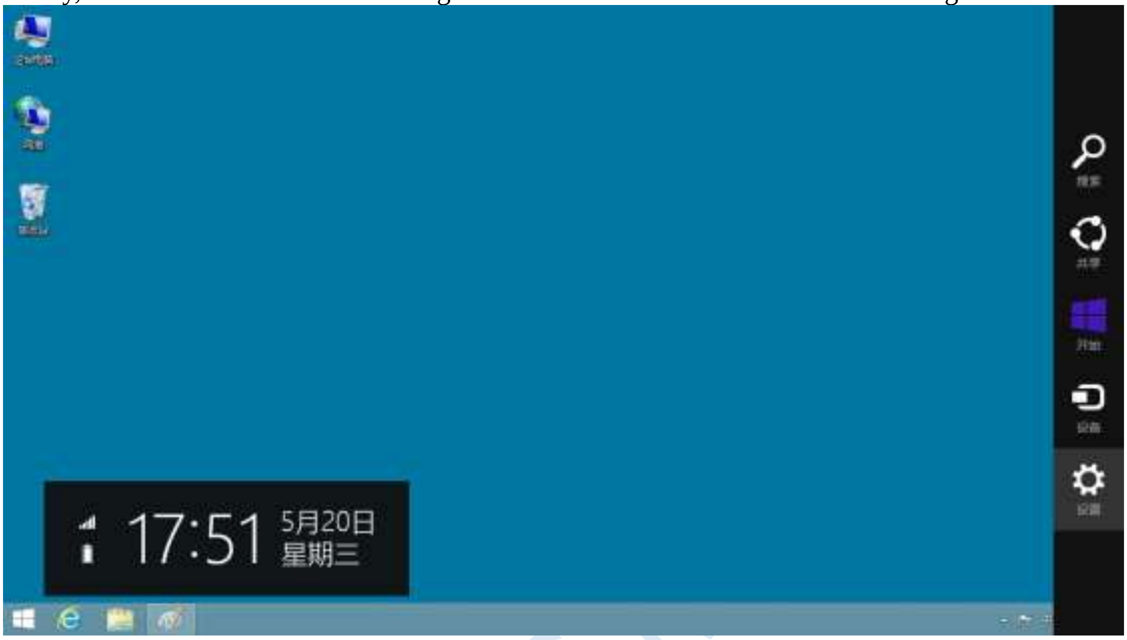
按 F10 以查看更多选项
按 Enter 以返回到操作系统

After booting to Windows 8, follow the Windows 8 (32-bit) installation method to complete the driver installation.

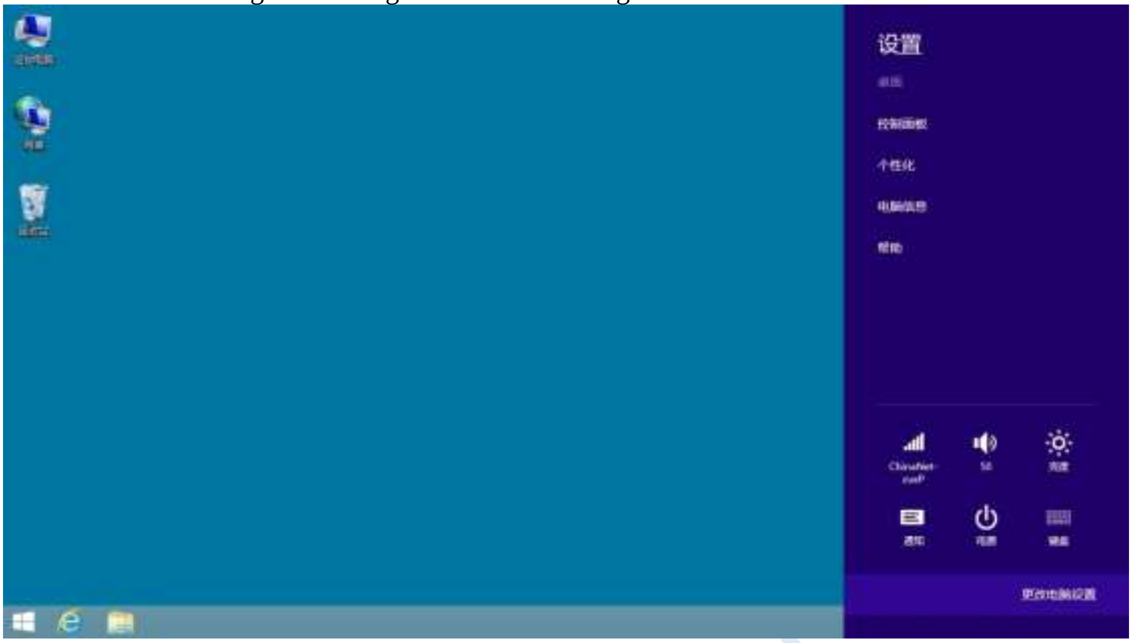
Installation Instructions in Windows 8.1 (64-bit)

Windows 8.1 has different method for entering the advanced boot menu with respect to Windows 8, which is specifically explained here.

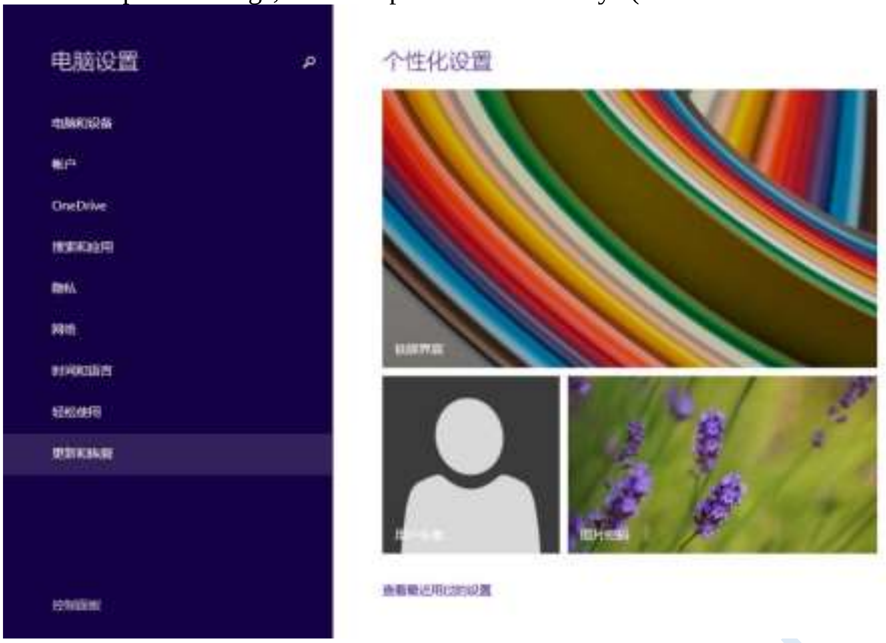
Firstly, move the mouse to the lower right corner of the screen and select the "Settings" button.



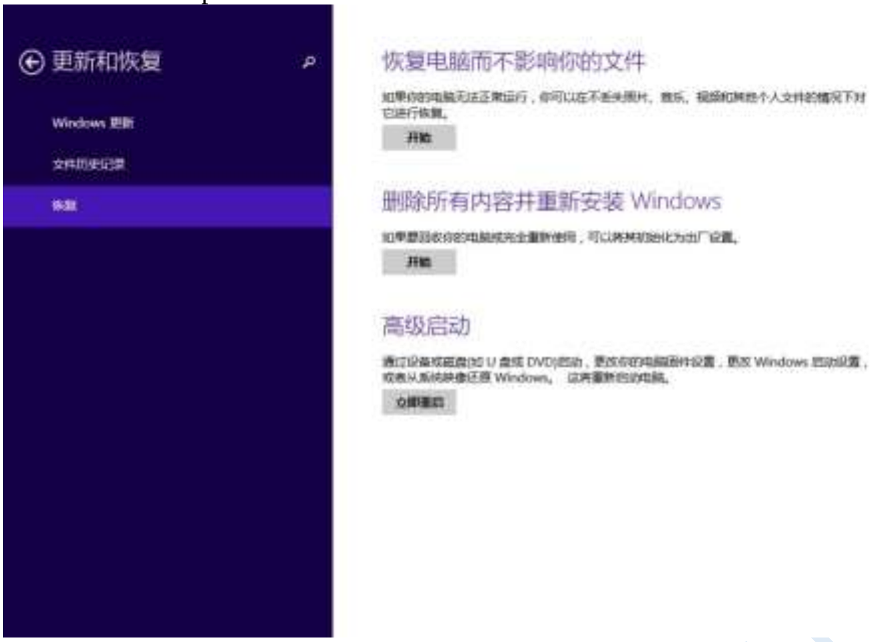
Then select the "Change PC settings" item in the settings window.



In the computer settings, select "Update and Recovery" (this is not the same as Windows 8, which is "General").



Select the "Restore" property page in the update and recovery page, and click the "Start Now" button under the "Advanced Startup" item.



恢复电脑而不影响你的文件

如果你的电脑无法正常运行，你可以在不丢失照片、音乐、视频和其他个人文件的情况下对它进行恢复。

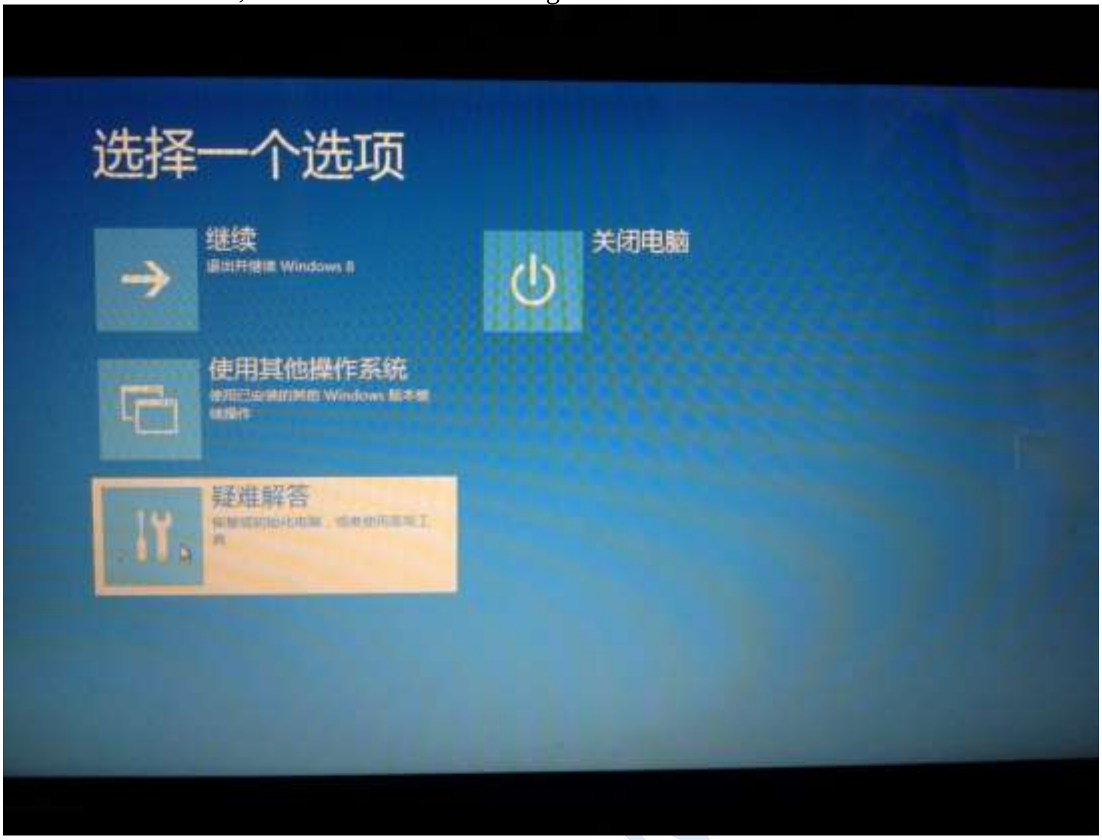
删除所有内容并重新安装 Windows

如果要回收你的电脑或完全重新使用，可以将其初始化为出厂设置。

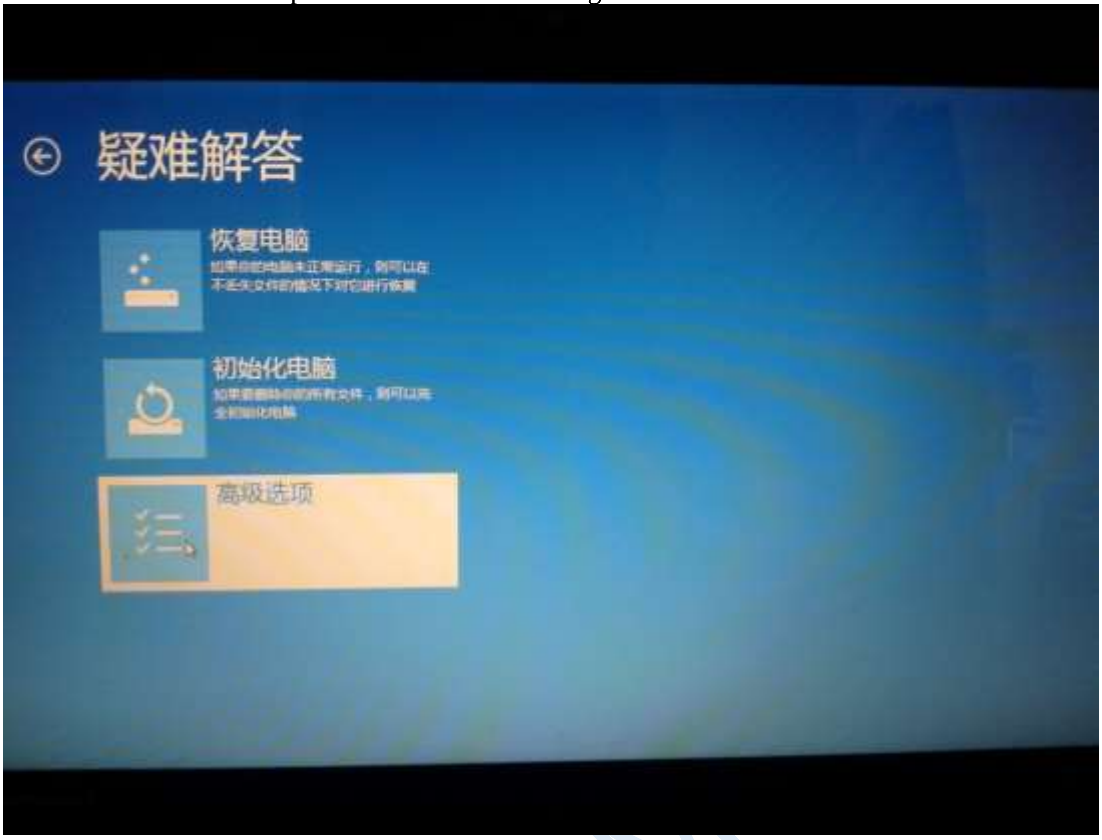
高级启动

通过设备或磁盘(如 U 盘或 DVD)启动，更改你的电脑固件设置，更改 Windows 启动设置，或者从系统映像还原 Windows。 这将重新启动电脑。

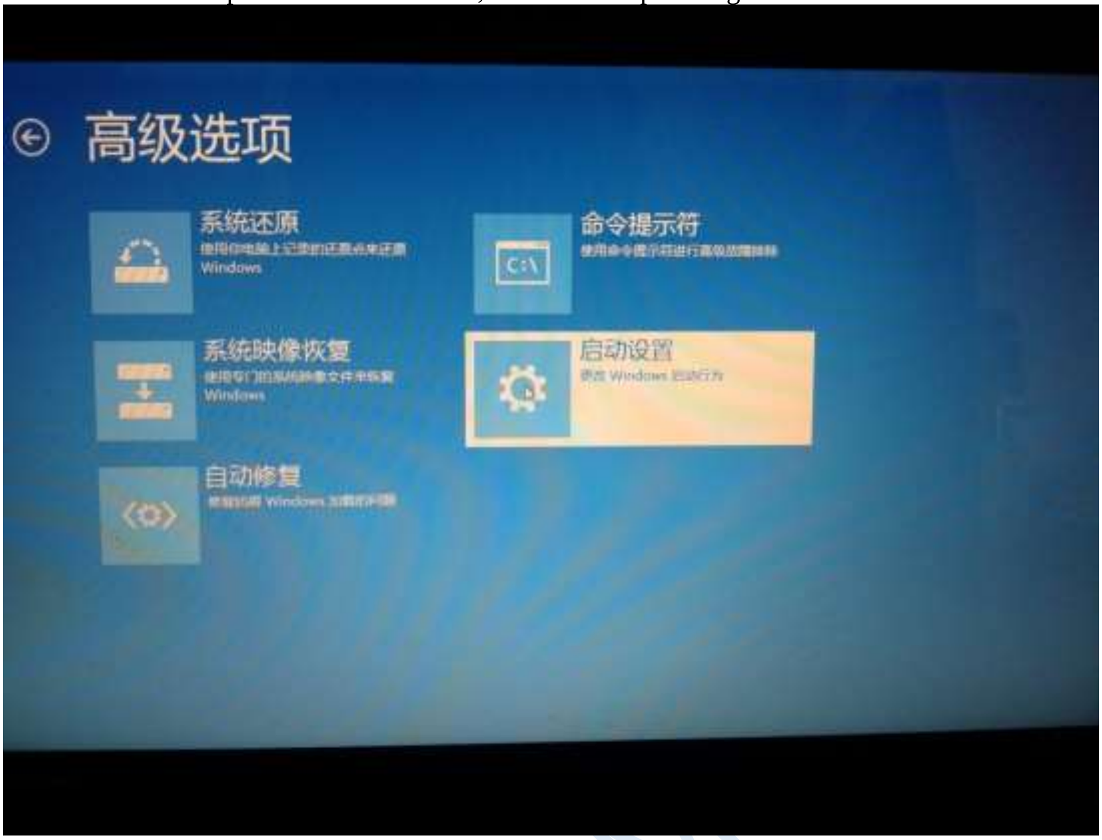
The following steps are the same as those of Window 8.
In the window below, select the "Troubleshooting" item.


选择一个选项
继续
退出并继续 Windows 8
关闭电脑
使用其他操作系统
使用已安装的其他 Windows 版本继续操作
疑难解答
恢复或初始化电脑，或者使用高级工具

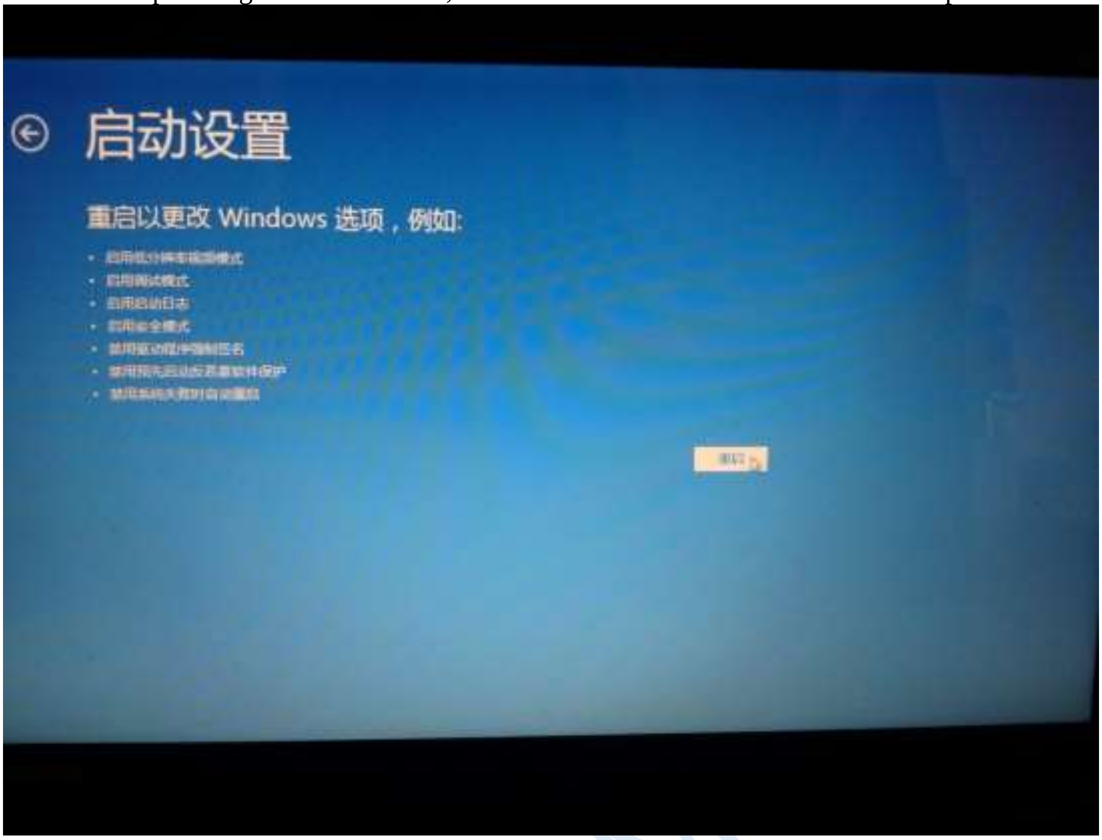
Then select "Advanced Options" in "Troubleshooting".


疑难解答
恢复电脑
如果你的电脑未正常运行，则可以在不丢失文件的情况下对它进行恢复
初始化电脑
如果要删除你的所有文件，则可以完全初始化电脑
高级选项

In the "Advanced Options" window below, select "Startup Settings".


高级选项
系统还原
使用你电脑上记录的还原点来还原 Windows
命令提示符
使用命令提示符进行高级故障排除
系统映像恢复
使用专门的系统映像文件来恢复 Windows
启动设置
更改 Windows 启动行为
自动修复
修复妨碍 Windows 加载的问题

In the "Startup Settings" window below, click the "Restart" button to restart the computer.

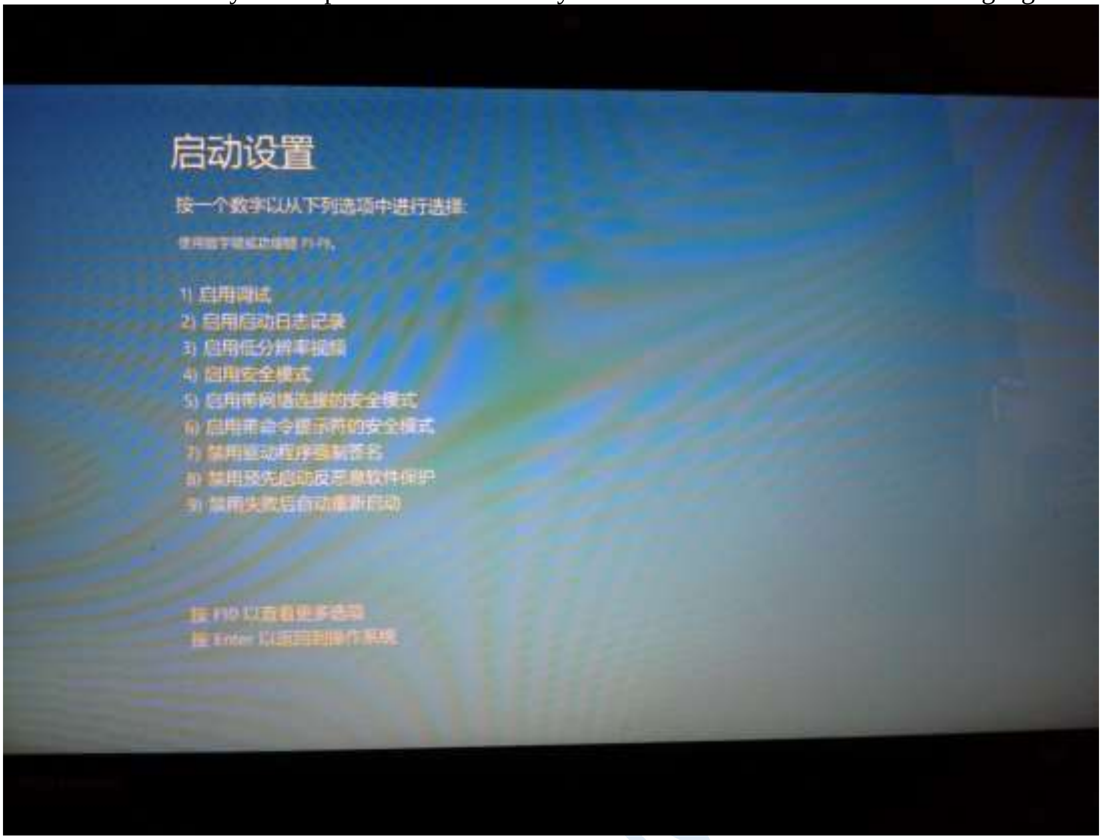


启动设置

重启以更改 Windows 选项，例如:

- 启用低分辨率视频模式
- 启用调试模式
- 启用启动日志
- 启用安全模式
- 禁用驱动程序强制签名
- 禁用预先启动反恶意软件保护
- 禁用系统失败时自动重启

After the computer restarts, it will enter the "Startup Settings" window automatically as shown in the figure below. Press the number key "7" or press the function key "F7" to select "Disable driver forcing signature" to start.



启动设置

按一个数字以从下列选项中进行选择:

使用数字键或功能键 F1-F9。

1) 启用调试
2) 启用启动日志记录
3) 启用低分辨率视频
4) 启用安全模式
5) 启用带网络连接的安全模式
6) 启用带命令提示符的安全模式
7) 禁用驱动程序强制签名
8) 禁用预先启动反恶意软件保护
9) 禁用失败后自动重新启动

按 F10 以查看更多选项
按 Enter 以返回到操作系统

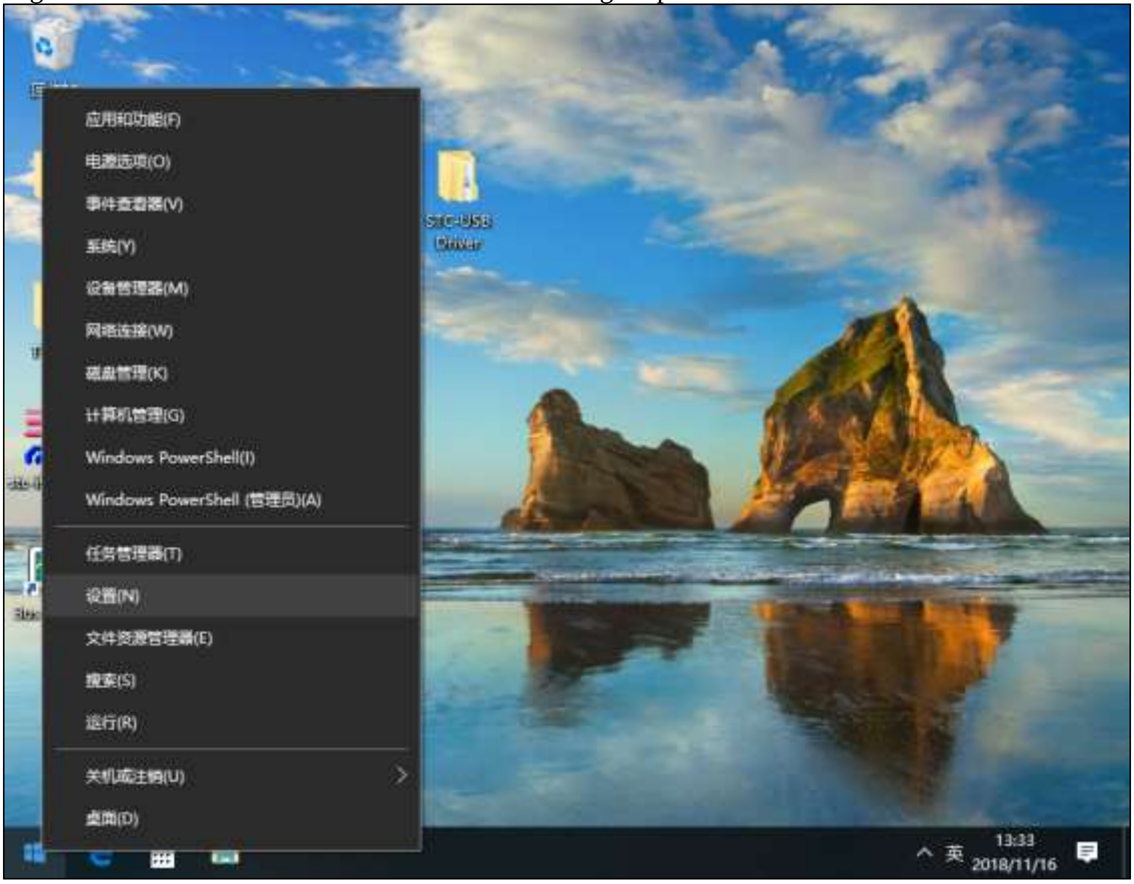
After booting to Windows 8, follow the Windows 8 (32-bit) installation method to complete the driver installation.

Installation Instructions in Windows 10 (64-bit)

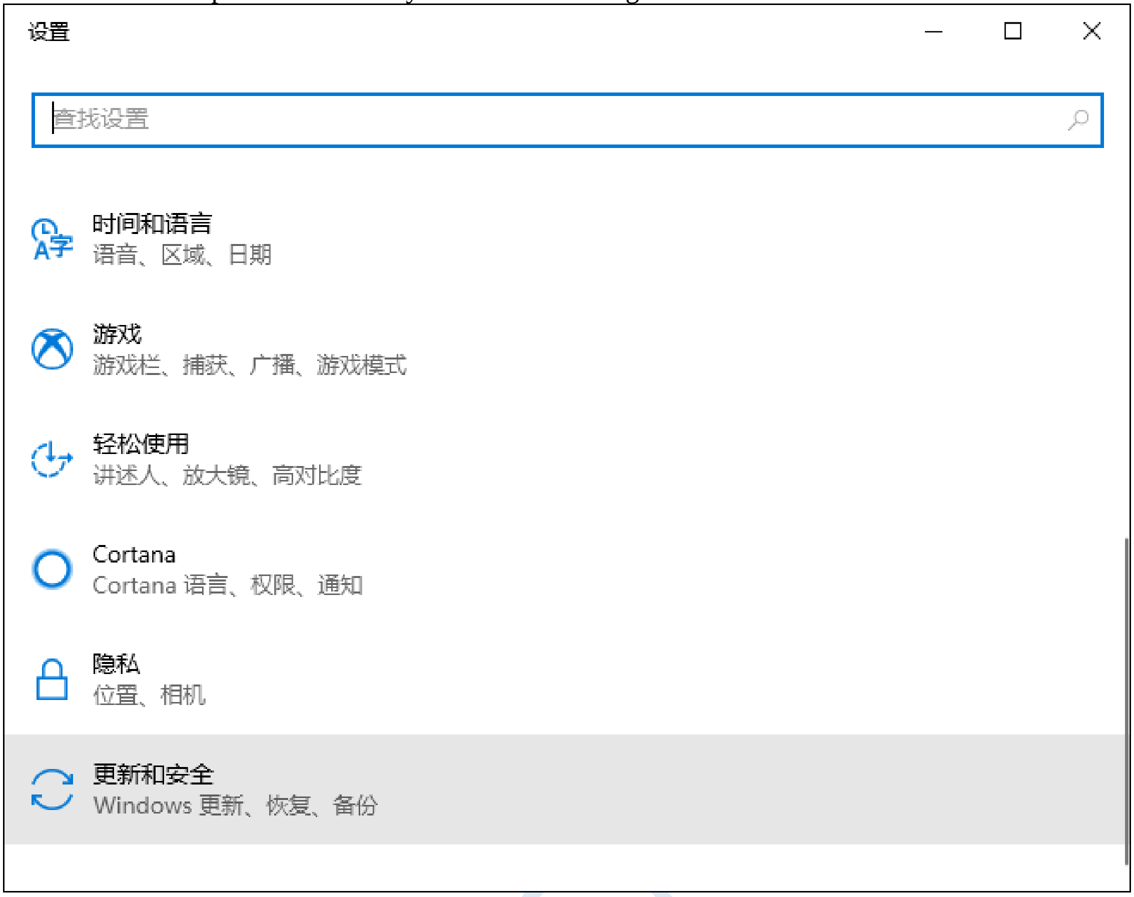
By default, the driver without digital signature cannot be successfully installed in Windows 10 64-bit operating system. So, you need to follow the steps below before installing the STC-USB driver, skip the digital signature temporarily, and the installation will be successful.

Before installing the driver, you need to extract the "STC-USB Driver" folder to the hard disk from the STC-ISP download software package downloaded from the STC official website. Prepare the chip with USB download function, but don't connect the computer firstly.

Right-click on the "Start" menu and select the "Settings" option.



Then select the "Update and Security" item in the settings window.


设置

查找设置

时间和语言
语音、区域、日期

游戏
游戏栏、捕获、广播、游戏模式

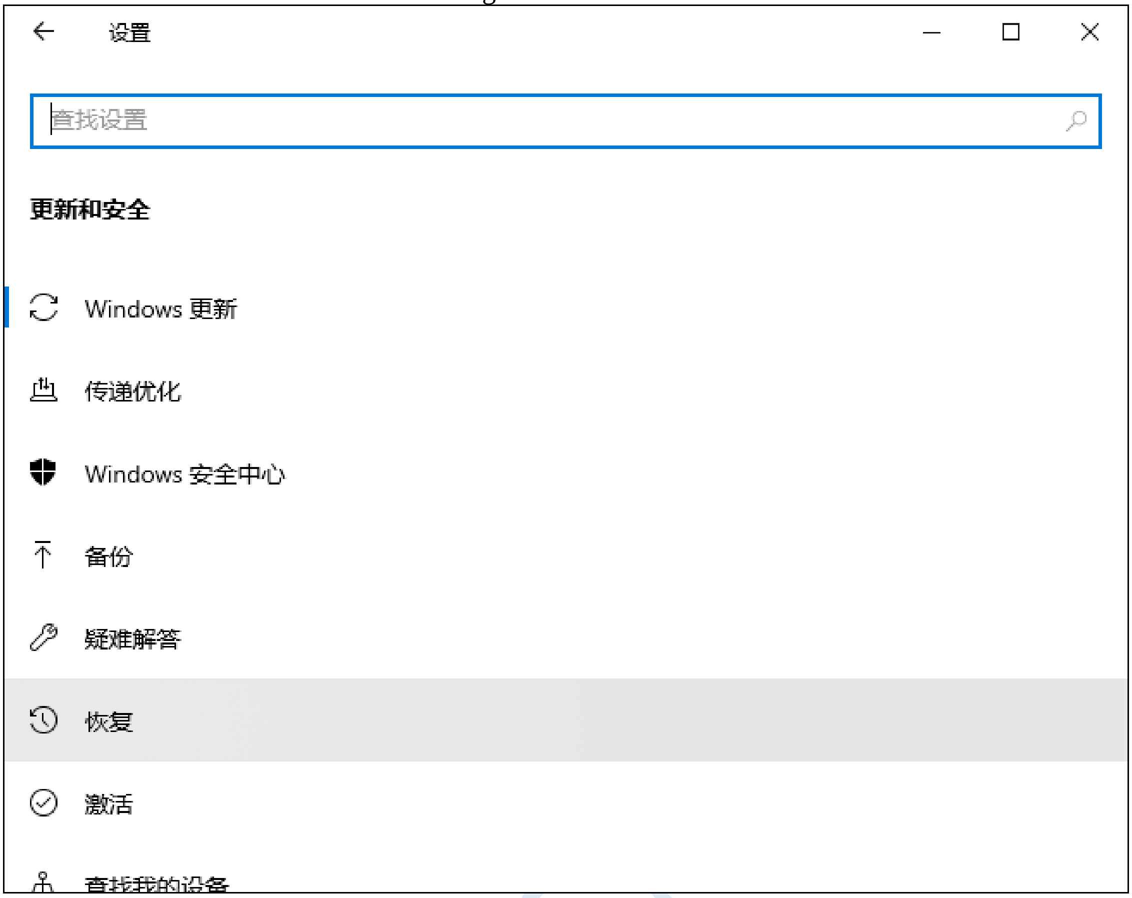
轻松使用
讲述人、放大镜、高对比度

Cortana
Cortana 语言、权限、通知

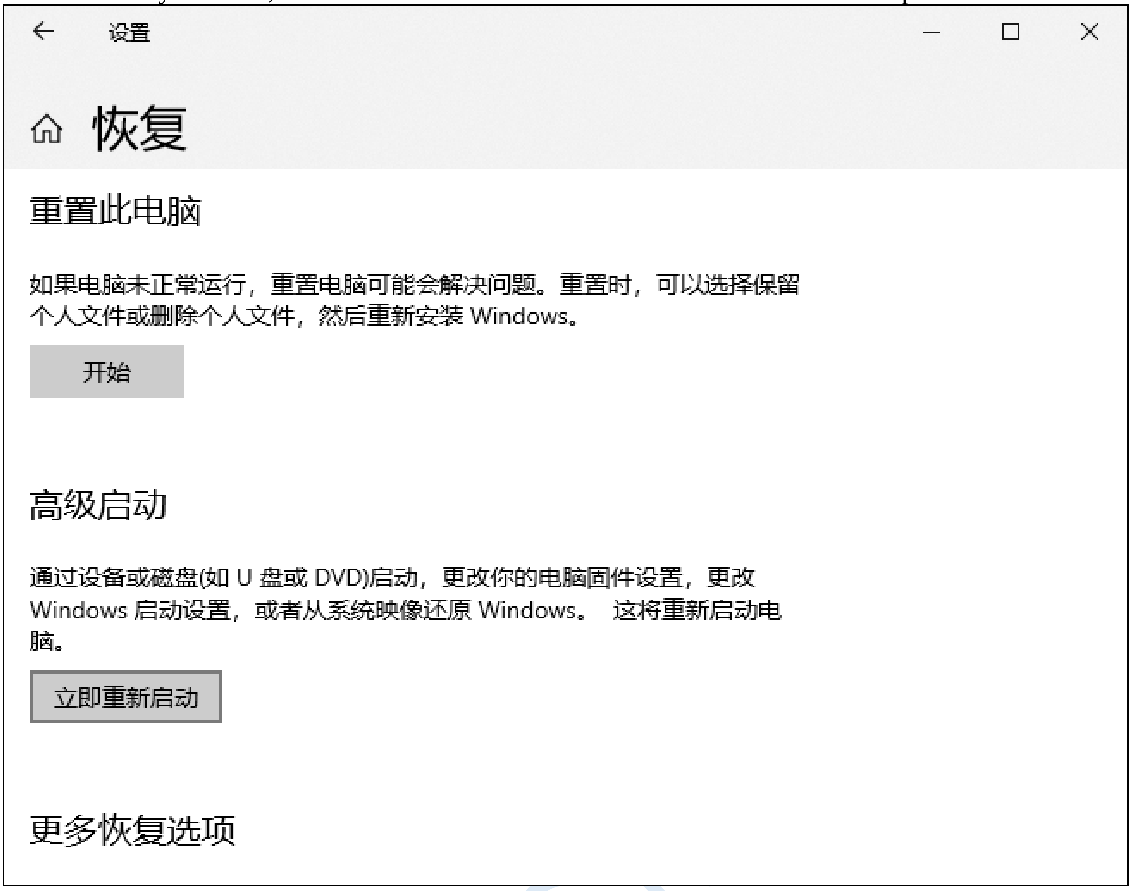
隐私
位置、相机

更新和安全
Windows 更新、恢复、备份

Then select the "Restore" item in the settings window.



In the recovery window, click the "Restart Now" button in the "Advanced Startup" item.



重置此电脑

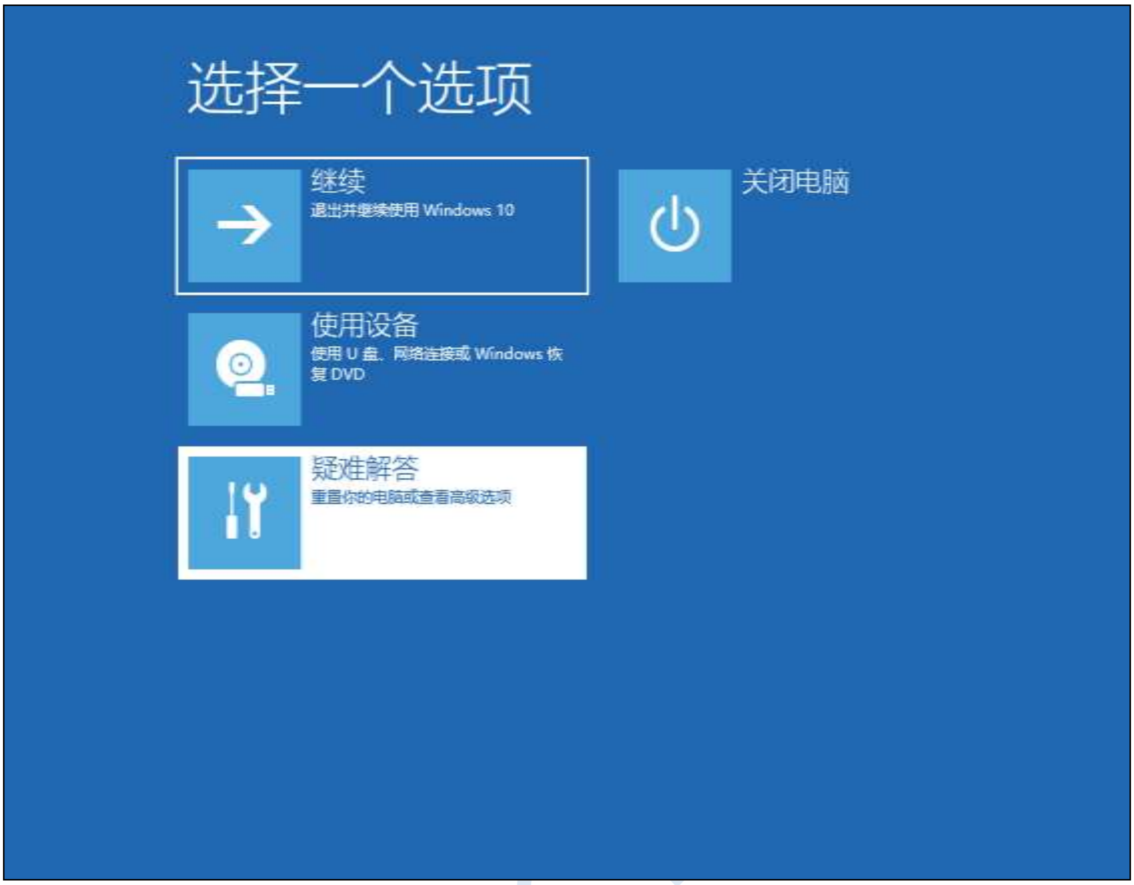
如果电脑未正常运行，重置电脑可能会解决问题。重置时，可以选择保留个人文件或删除个人文件，然后重新安装 Windows。

高级启动

通过设备或磁盘(如 U 盘或 DVD)启动，更改你的电脑固件设置，更改 Windows 启动设置，或者从系统映像还原 Windows。 这将重新启动电脑。

更多恢复选项

Before the computer restarts, the system will enter the following boot menu firstly and select the "Troubleshooting" item.



选择一个选项

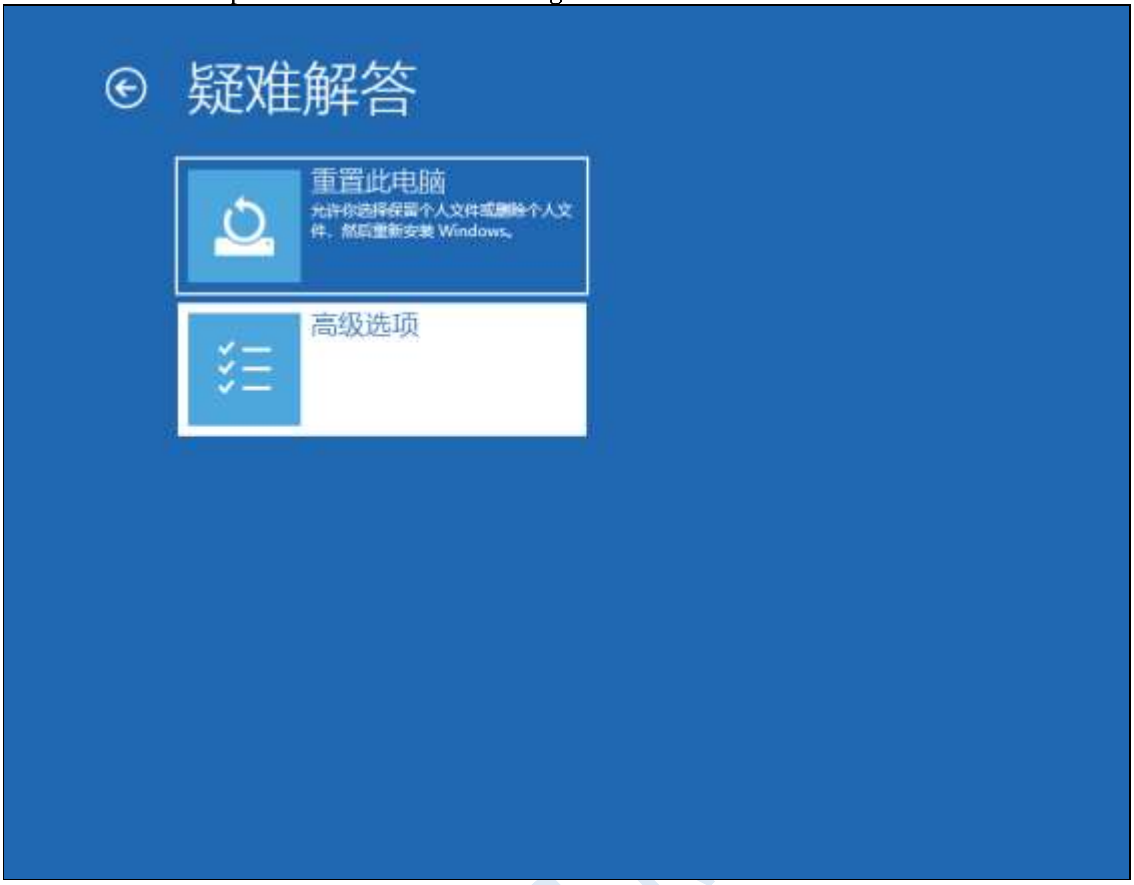
继续
退出并继续使用 Windows 10

关闭电脑

使用设备
使用 U 盘、网络连接或 Windows 恢复 DVD

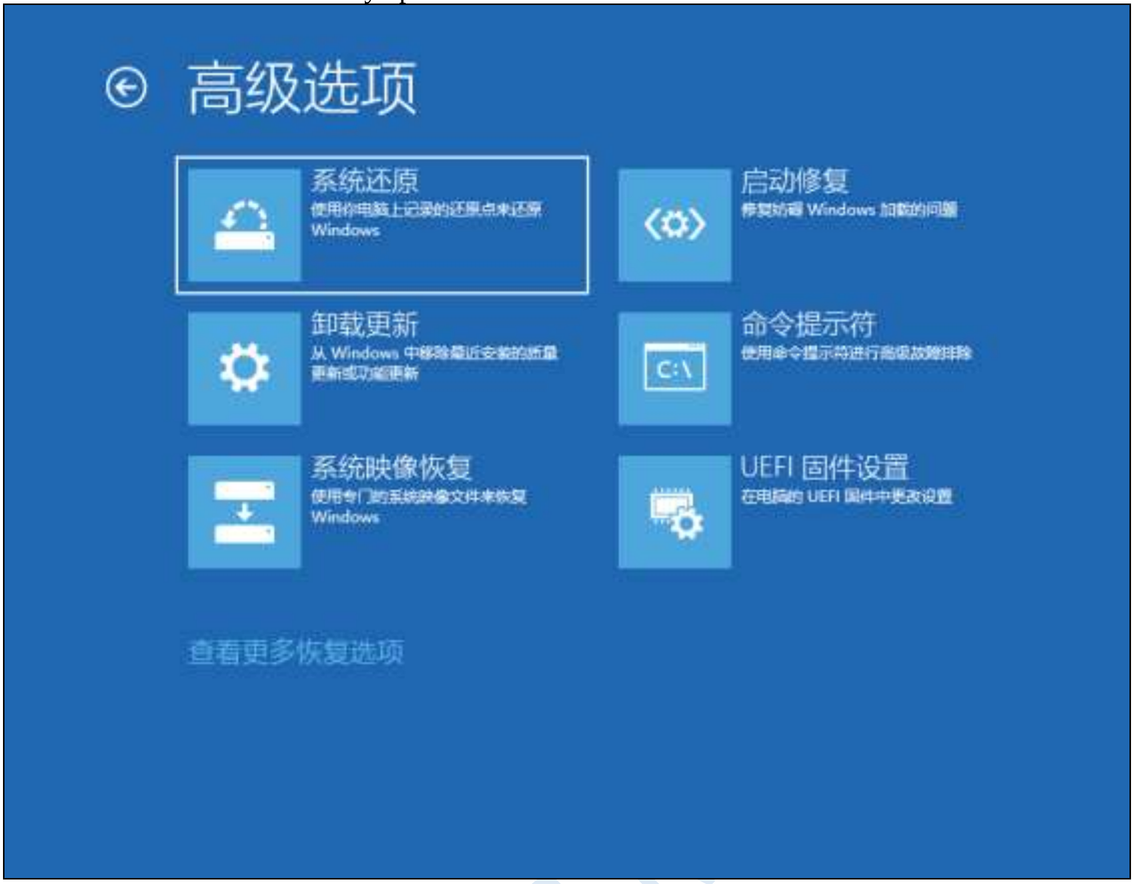
疑难解答
重置你的电脑或查看高级选项

Select "Advanced Options" in the troubleshooting window.

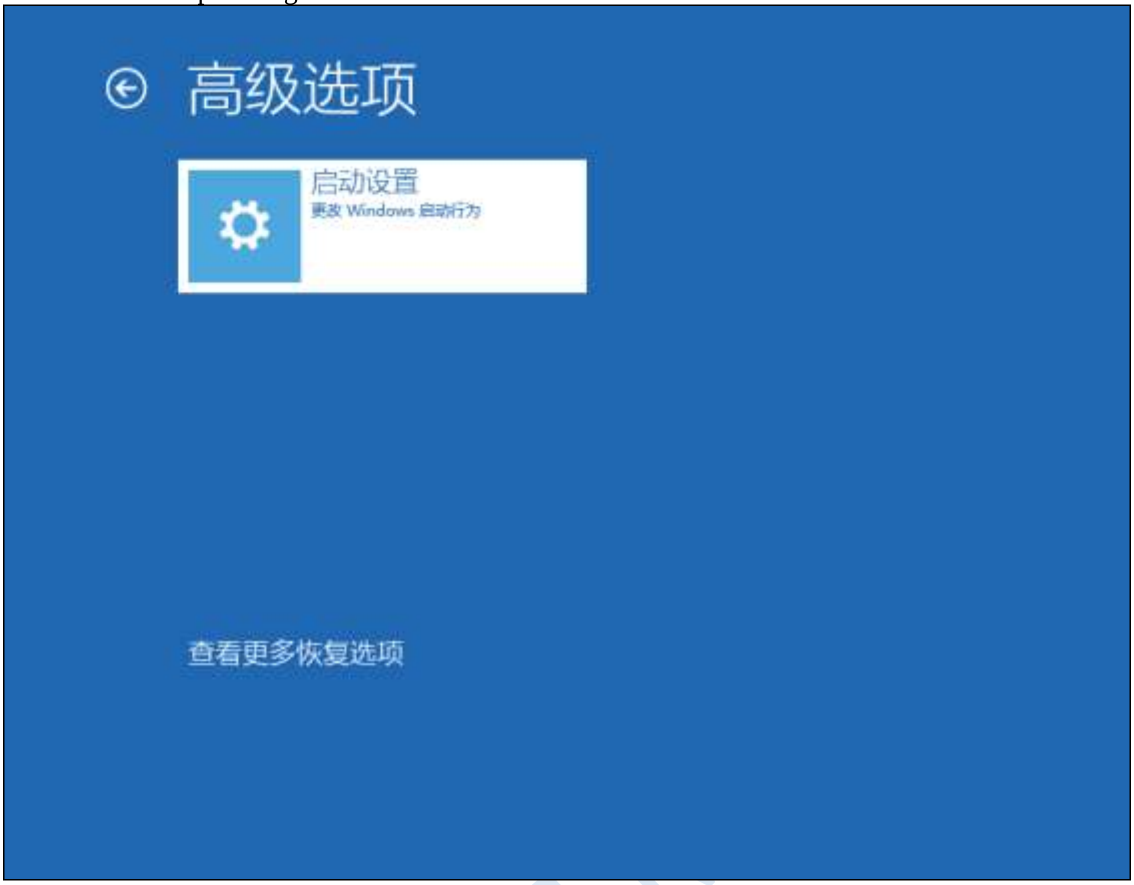


允许你选择保留个人文件或删除个人文件，然后重新安装 Windows。

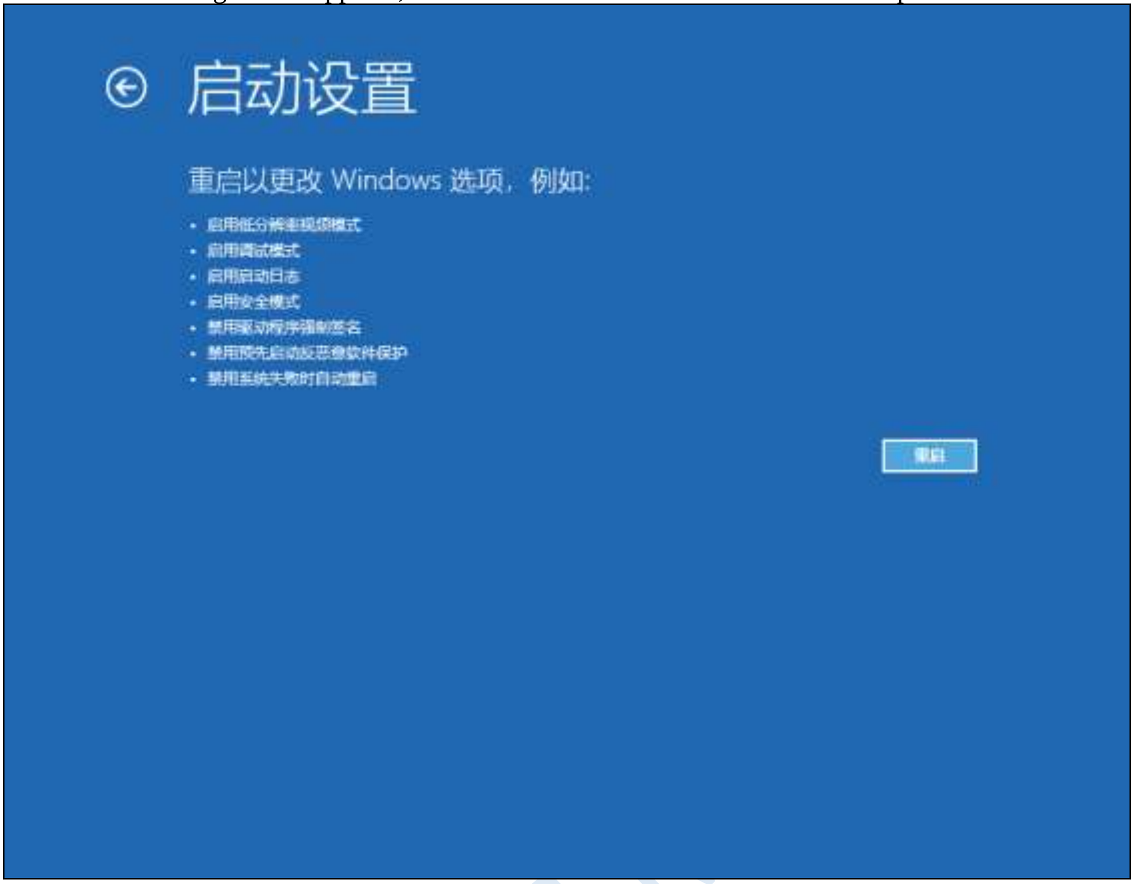
Then select "View more recovery options".


高级选项
系统还原
使用你电脑上记录的还原点来还原 Windows
启动修复
修复妨碍 Windows 加载的问题
卸载更新
从 Windows 中移除最近安装的质量更新或功能更新
命令提示符
使用命令提示符进行高级故障排除
C:\
系统映像恢复
使用专门的系统映像文件来恢复 Windows
UEFI 固件设置
在电脑的 UEFI 固件中更改设置
查看更多恢复选项

Select the "Startup Settings" item.



When the following screen appears, click the "Restart" button to restart the computer.

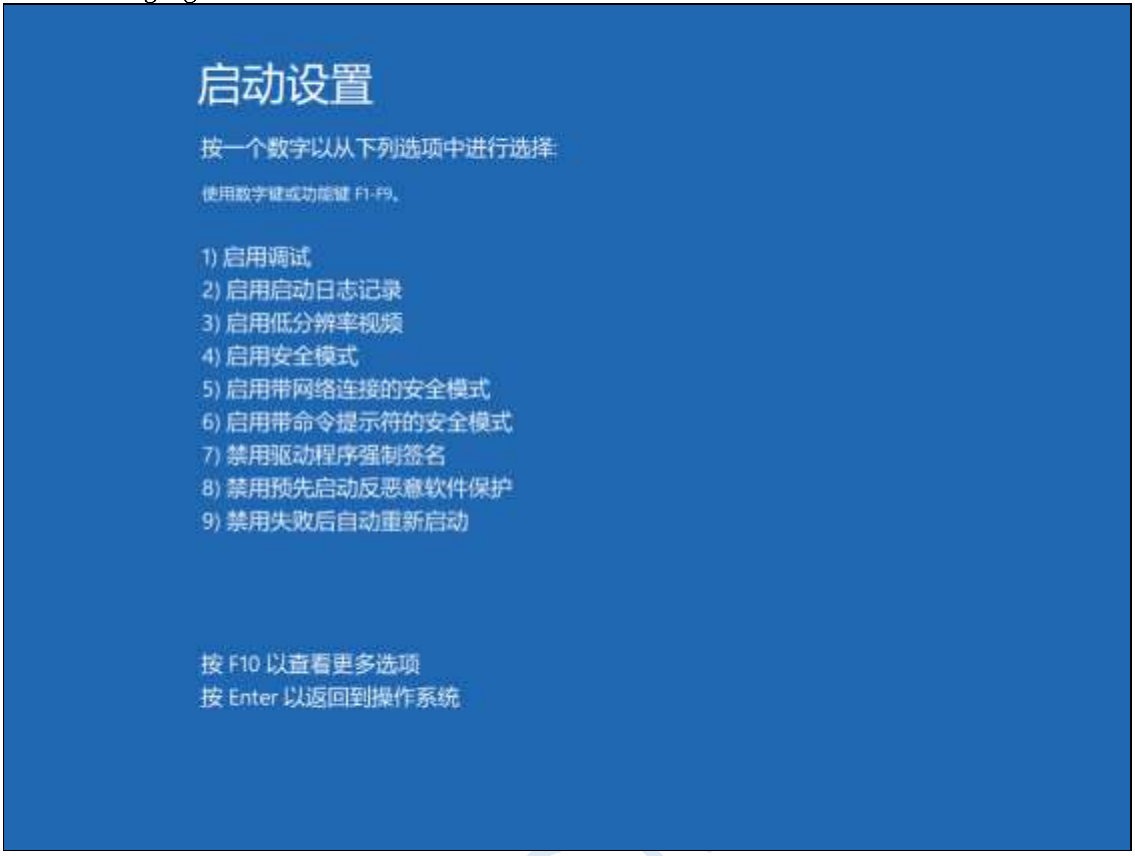


启动设置

重启以更改 Windows 选项，例如:

- 启用低分辨率视频模式
- 启用调试模式
- 启用启动日志
- 启用安全模式
- 禁用驱动程序强制签名
- 禁用预先启动反恶意软件保护
- 禁用系统失败时自动重启

After the computer restarts, the "Startup Settings" window will pop up. Press the "F7" button to select the "Prohibit driver forcing signature" item.



启动设置

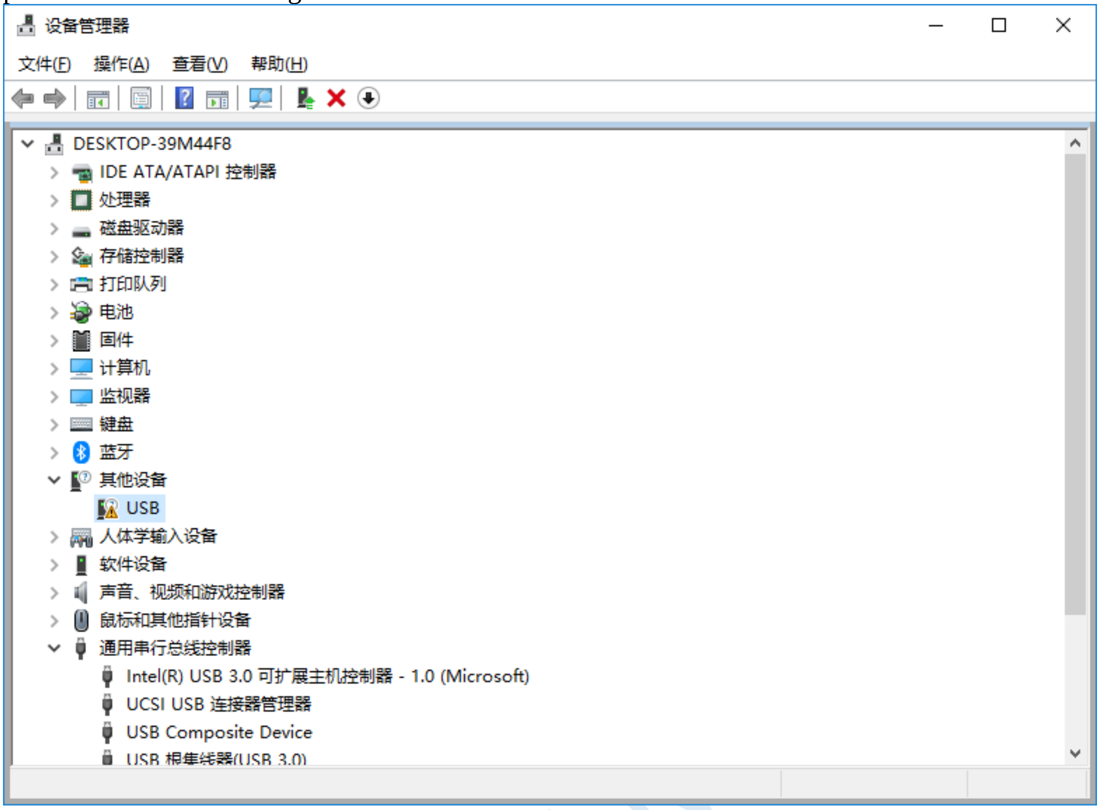
按一个数字以从下列选项中进行选择:

使用数字键或功能键 F1-F9。

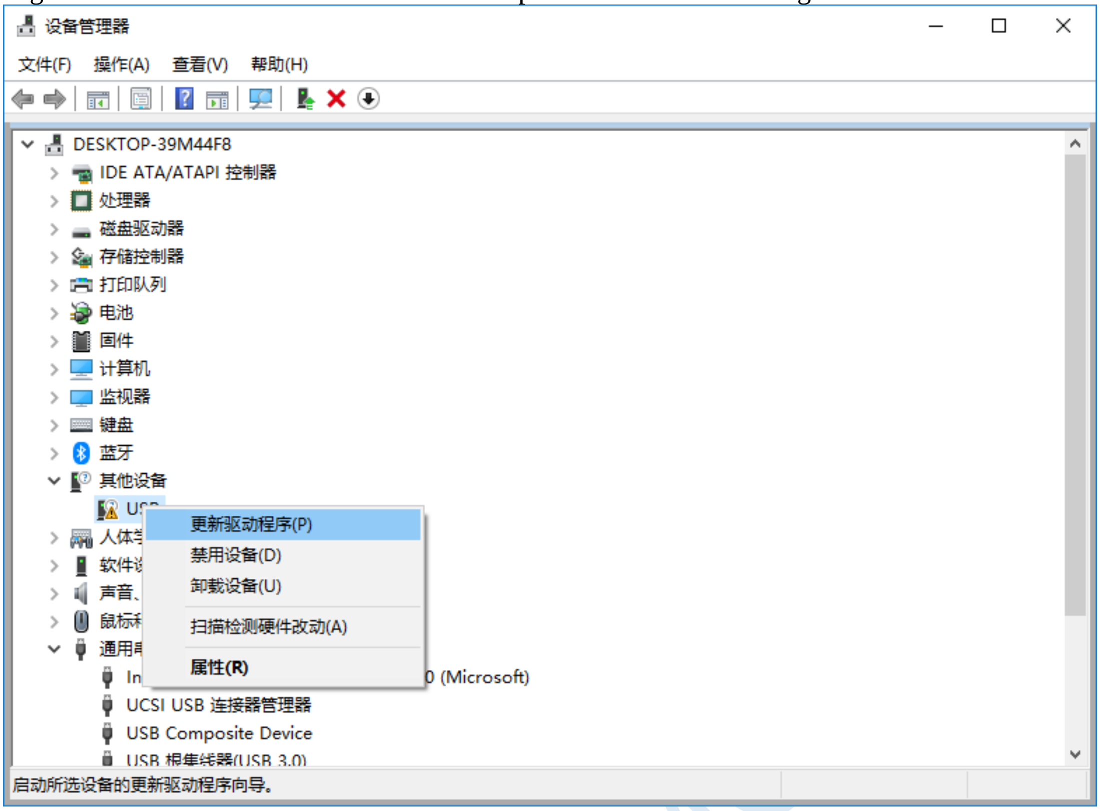
1) 启用调试
2) 启用启动日志记录
3) 启用低分辨率视频
4) 启用安全模式
5) 启用带网络连接的安全模式
6) 启用带命令提示符的安全模式
7) 禁用驱动程序强制签名
8) 禁用预先启动反恶意软件保护
9) 禁用失败后自动重新启动

按 F10 以查看更多选项
按 Enter 以返回到操作系统

After the computer is started, connect the prepared chip to the computer with a USB cable and open the "Device Manager". Now, the driver has not yet been installed, so it will appear as an unknown device with an exclamation point in the Device Manager.

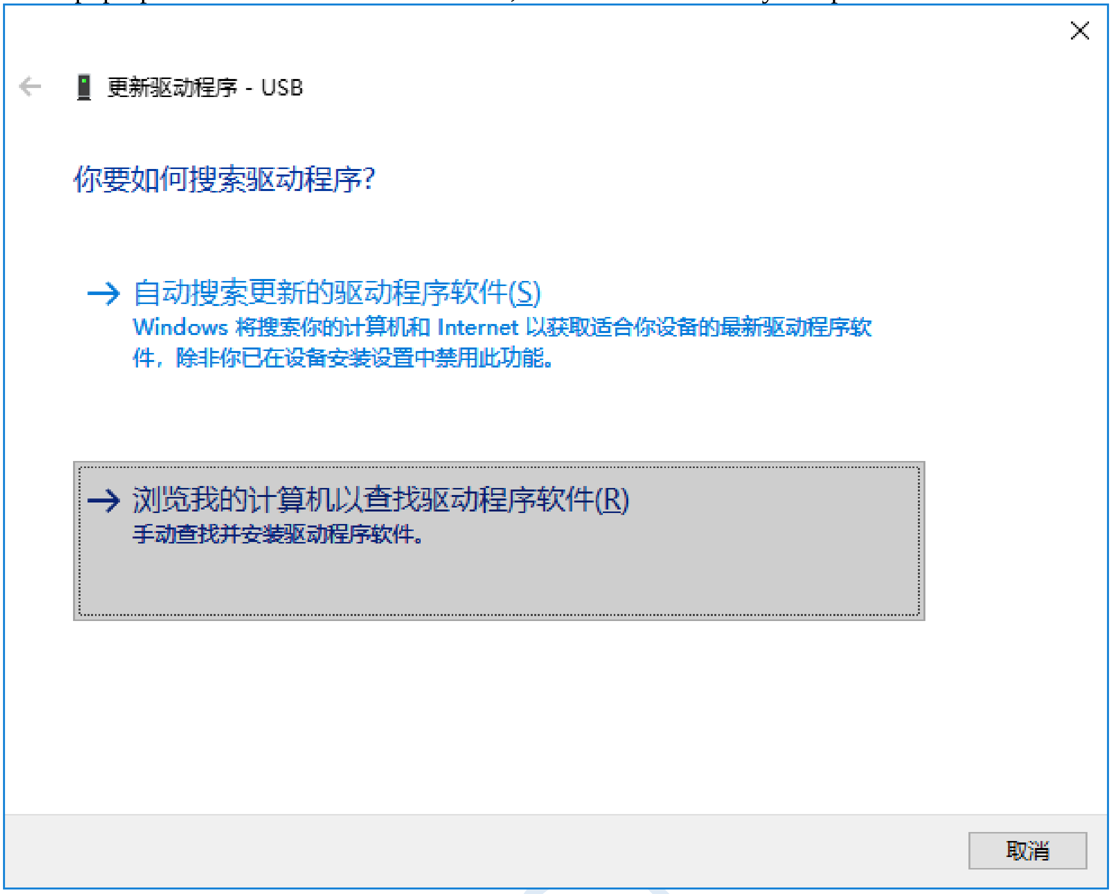


Right-click the unknown device and select "Update Driver" from the right-click menu.



启动所选设备的更新驱动程序向导。

In the pop-up driver installer selection screen, select the "Browse my computer for driver software" item.



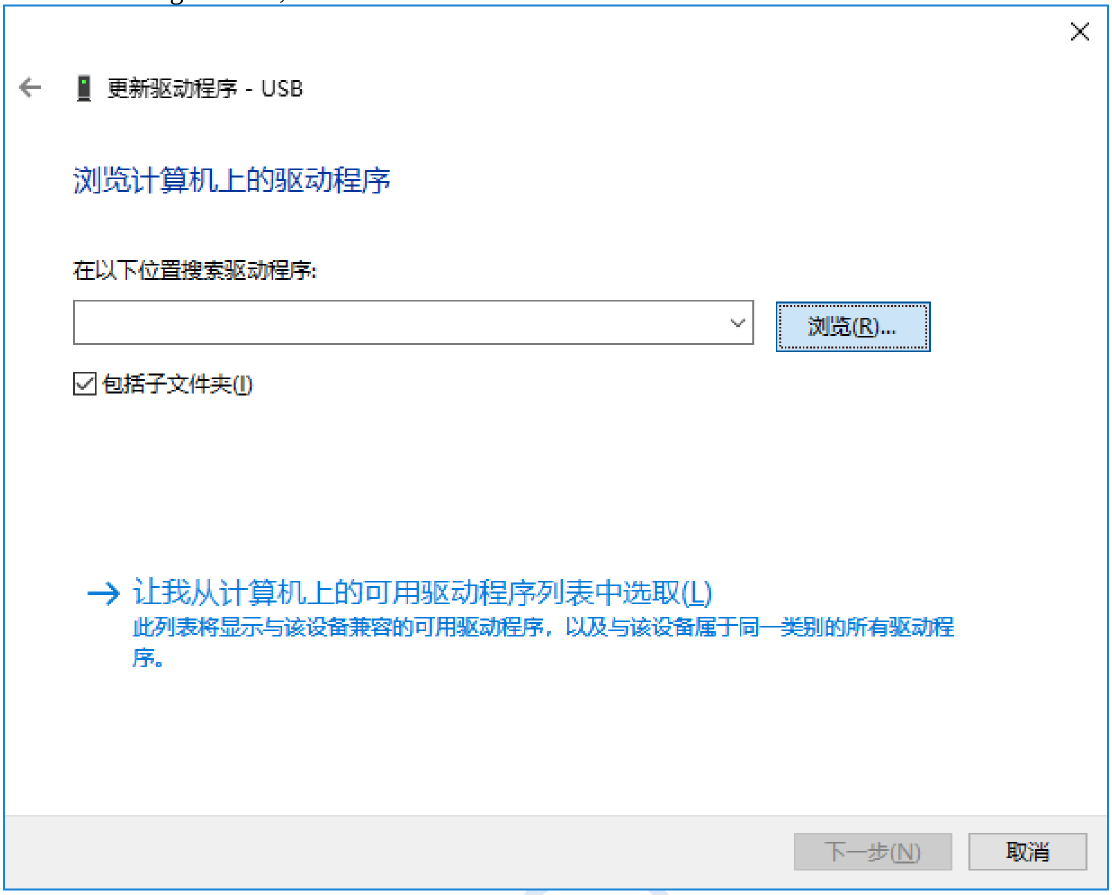
更新驱动程序 - USB

你要如何搜索驱动程序?

→ 自动搜索更新的驱动程序软件(S)
Windows 将搜索你的计算机和 Internet 以获取适合你设备的最新驱动程序软件，除非你已在设备安装设置中禁用此功能。

→ 浏览我的计算机以查找驱动程序软件(R)
手动查找并安装驱动程序软件。

In the following window, click the "Browse" button.



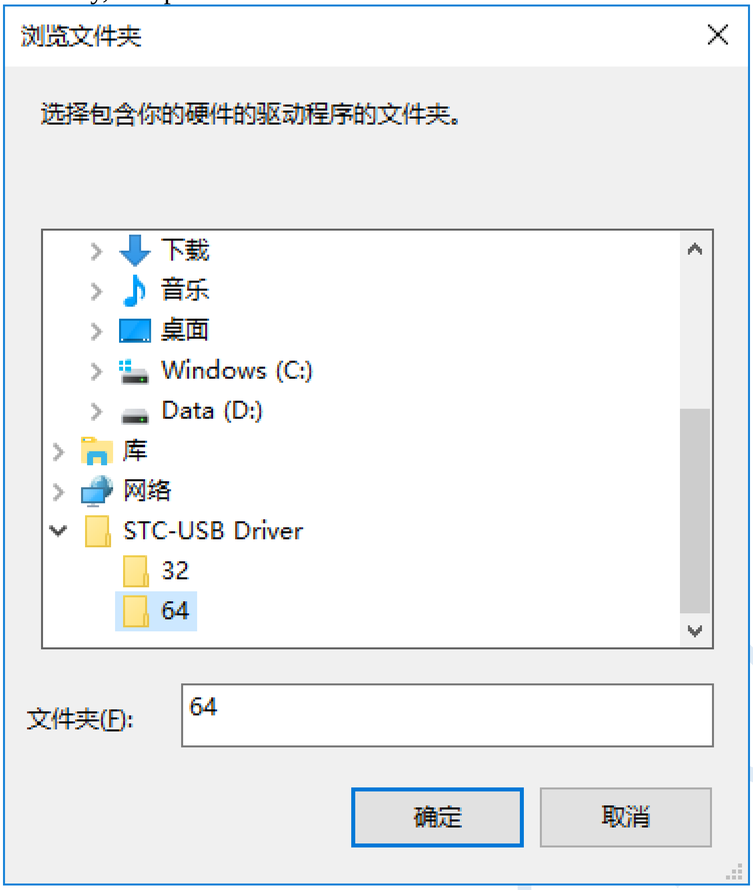
更新驱动程序 - USB

浏览计算机上的驱动程序

在以下位置搜索驱动程序:

此列表将显示与该设备兼容的可用驱动程序，以及与该设备属于同一类别的所有驱动程序。

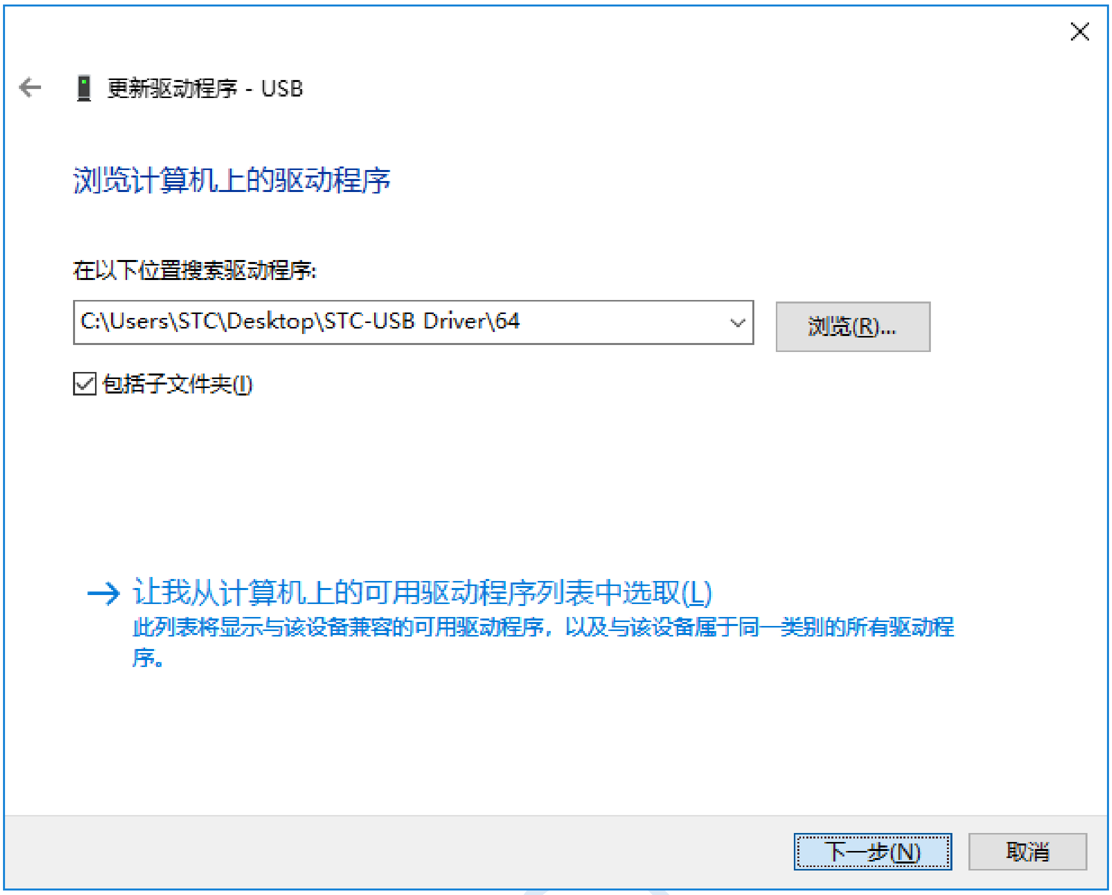
Find the "STC-USB Driver" directory that was previously extracted to your hard disk, select the "64" directory in the directory, and press "OK" button.



选择包含你的硬件的驱动程序的文件夹。

文件夹(F): 64

Click "Next" to start the driver installation.



更新驱动程序 - USB

浏览计算机上的驱动程序

在以下位置搜索驱动程序:

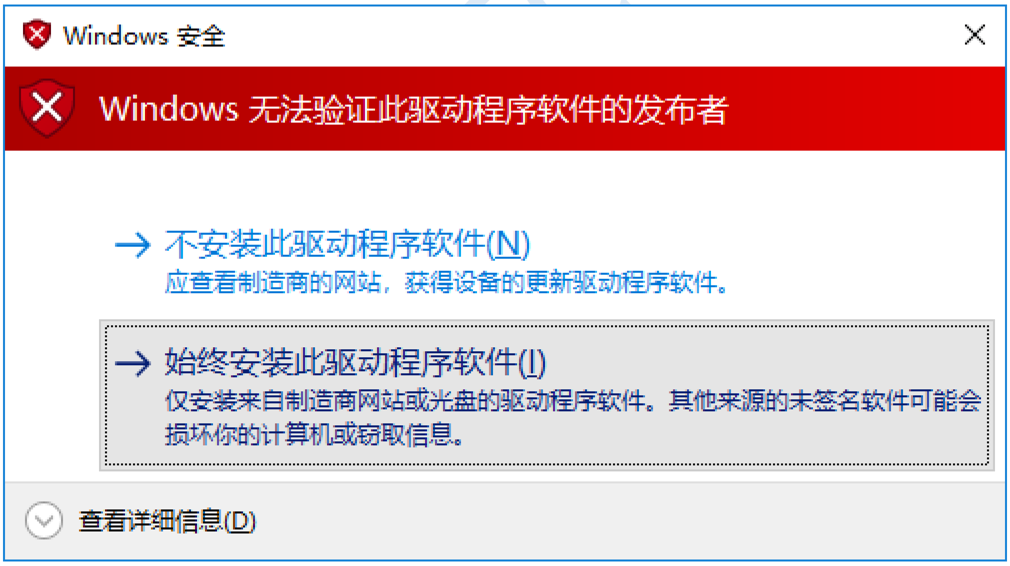
C:\Users\STC\Desktop\STC-USB Driver\64

☑ 包括子文件夹(I)

→ 让我从计算机上的可用驱动程序列表中选取(L)

此列表将显示与该设备兼容的可用驱动程序，以及与该设备属于同一类别的所有驱动程序。

During the driver installation process, the following warning screen will pop up, select "Always install this driver software".



Windows 安全

Windows 无法验证此驱动程序软件的发布者

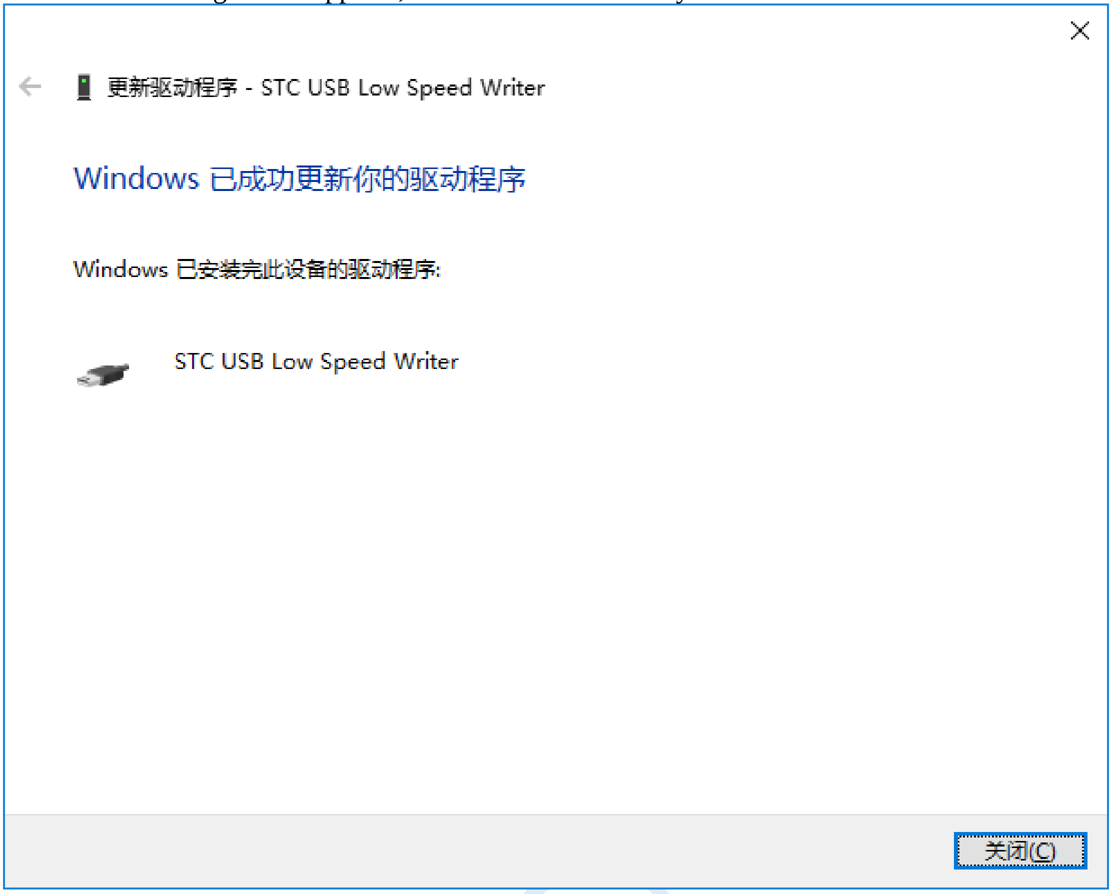
→ 不安装此驱动程序软件(N)

应查看制造商的网站，获得设备的更新驱动程序软件。

→ 始终安装此驱动程序软件(I)

仅安装来自制造商网站或光盘的驱动程序软件。其他来源的未签名软件可能会损坏你的计算机或窃取信息。

When the following screen appears, the driver is successfully installed.



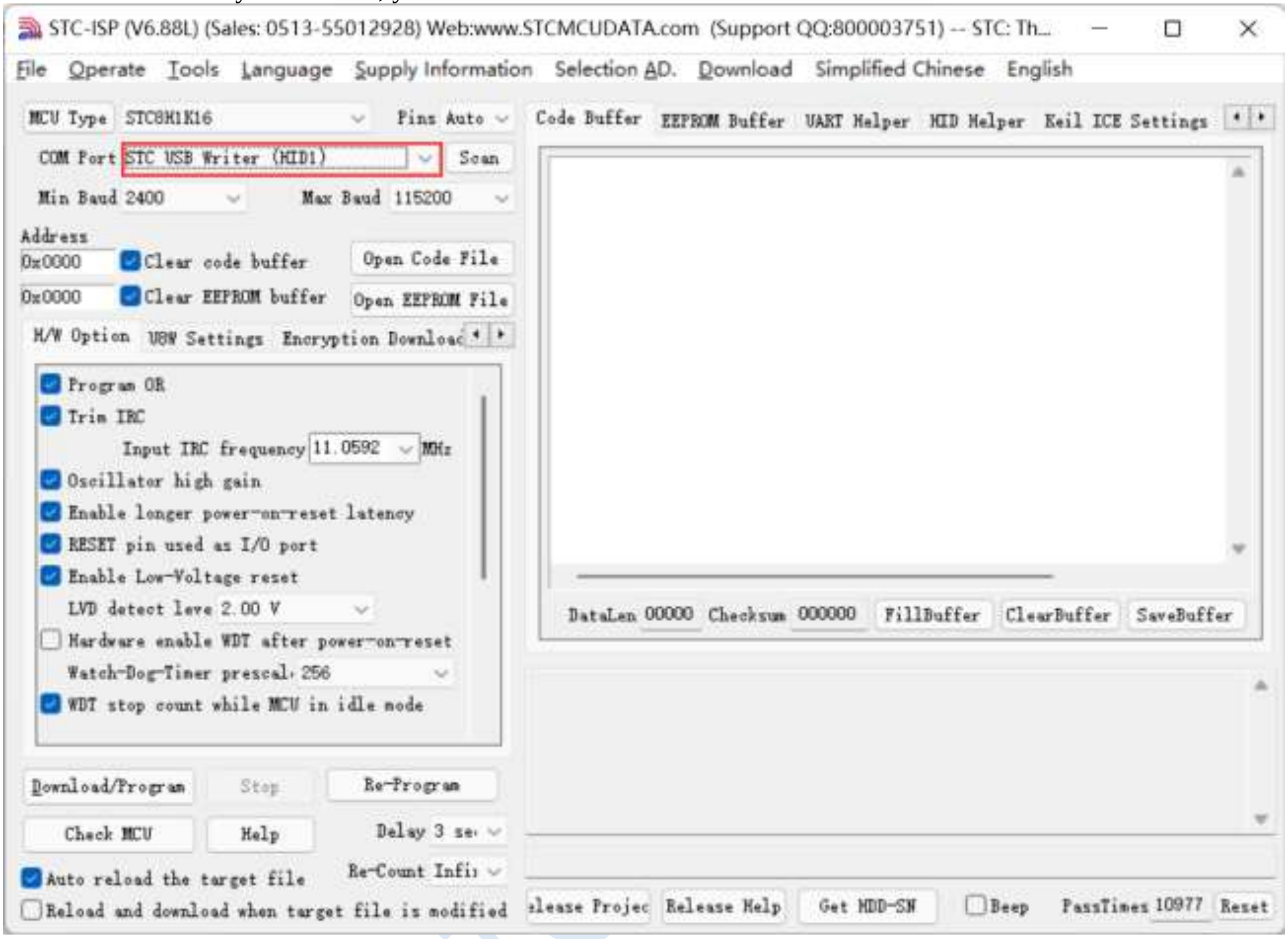
更新驱动程序 - STC USB Low Speed Writer

Windows 已成功更新你的驱动程序

Windows 已安装完此设备的驱动程序:

STC USB Low Speed Writer

Back to the download software of STC-ISP, "STC USB Writer (HID1)" in the "Serial Port Number" drop-down list is selected automatically at this time, you can use USB for ISP download.



STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...

MCU Type STC8H1K16

Pins Auto

COM Port STC USB Writer (HID1)

Min Baud 2400

Max Baud 115200

Address

0x0000 ☑ Clear code buffer

0x0000 ☑ Clear EEPROM buffer

☑ Program OR

☑ Trim IRC

Input IRC frequency 11.0592 MHz

☑ Oscillator high gain

☑ Enable longer power-on-reset latency

☑ RESET pin used as I/O port

☑ Enable Low-Voltage reset

LVD detect leve 2.00 V

□ Hardware enable WDT after power-on-reset

Watch-Dog-Timer prescal 256

☑ WDT stop count while MCU in idle mode

Delay 3 se

Re-Count Infi

☑ Auto reload the target file

□ Reload and download when target file is modified

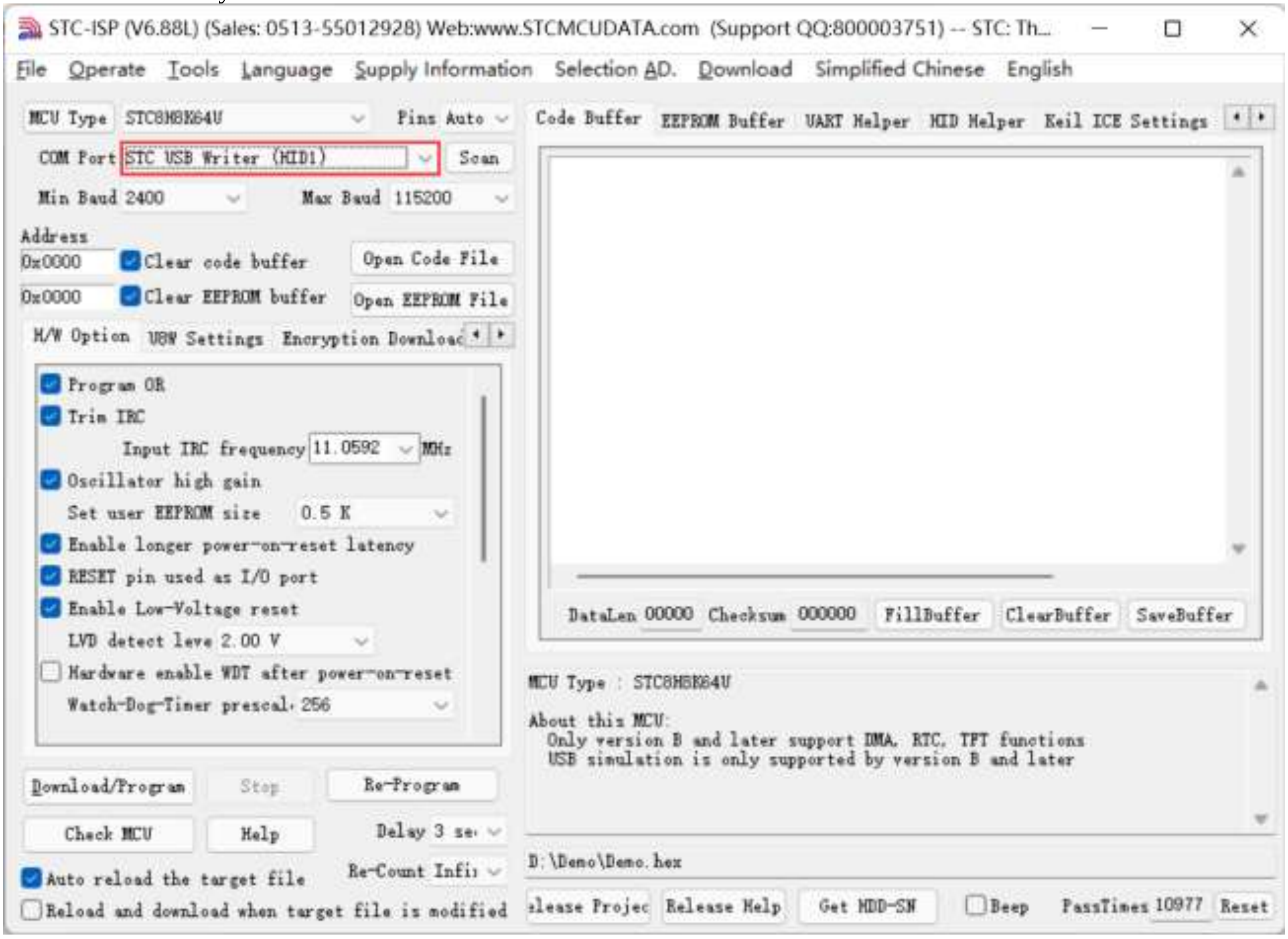
DataLen 00000 Checksum 000000

□ Beep

PassTimes 10977

Appendix E Download Step Demo using USB

1. Refer to the application circuit diagram in P5.1.5 to connect the microcontroller firstly, and connect the P3.2 port of the target chip to GND, and then connect the system to the USB port on the PC side. Open the ISP to download the software, and the serial number of the downloaded software will search for the "STC USB Writer (HID1)" USB device automatically.



STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...

MCU Type STC8H8K64U Pins Auto

COM Port STC USB Writer (HID1)

Min Baud 2400 Max Baud 115200

Address 0x0000 ☑ Clear code buffer

0x0000 ☑ Clear EEPROM buffer

☑ Program OR

☑ Trim IRC

Input IRC frequency 11.0592 MHz

☑ Oscillator high gain

Set user EEPROM size 0.5 K

☑ Enable longer power-on-reset latency

☑ RESET pin used as I/O port

☑ Enable Low-Voltage reset

LVD detect leve 2.00 V

☐ Hardware enable WDT after power-on-reset

Watch-Dog-Timer prescal 256

Delay 3 se

☑ Auto reload the target file

Re-Count Infi

☐ Reload and download when target file is modified

DataLen 00000 Checksum 000000

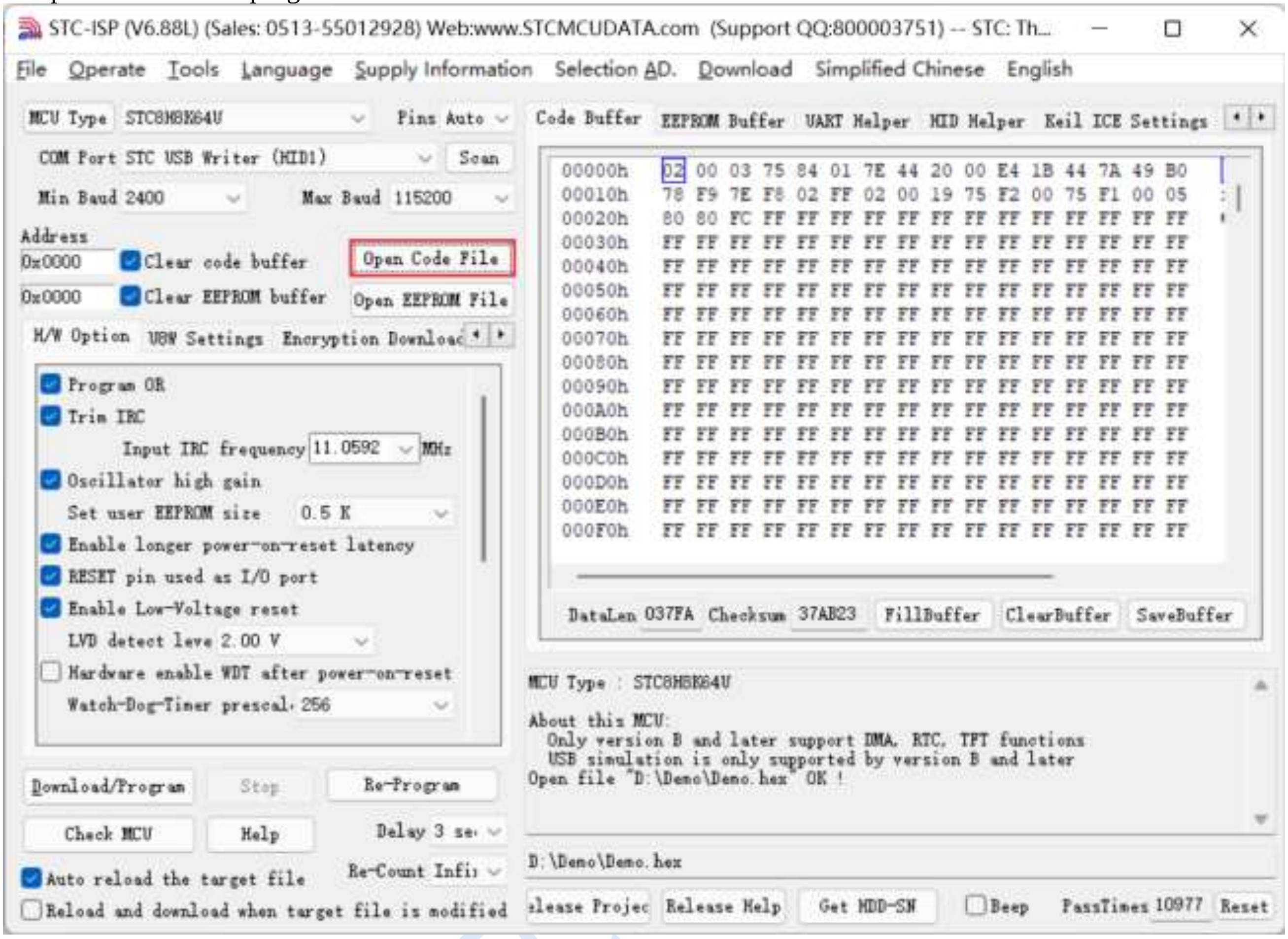
MCU Type : STC8H8K64U

About this MCU:
  Only version B and later support DMA, RTC, TFT functions
  USB simulation is only supported by version B and later

D:\Demo\Demo.hex

☐ Beep PassTimes 10977

2. Open the user code program.


STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...
File Operate Tools Language Supply Information Selection AD. Download Simplified Chinese English
Scan
Open Code File
Open EEPROM File
H/W Option USB Settings Encryption Download
Code Buffer EEPROM Buffer UART Helper HID Helper Keil ICE Settings
FillBuffer ClearBuffer SaveBuffer
Download/Program Stop Re-Program
Check MCU Help
Release Projec Release Help Get HDD-SN Reset

MCU Type STC8H8K64U
Pins Auto
COM Port STC USB Writer (HID1)
Min Baud 2400
Max Baud 115200
Address
0x0000 ☑ Clear code buffer
0x0000 ☑ Clear EEPROM buffer

☑ Program OR
☑ Trim IRC
Input IRC frequency 11.0592 MHz
☑ Oscillator high gain
Set user EEPROM size 0.5 K
☑ Enable longer power-on-reset latency
☑ RESET pin used as I/O port
☑ Enable Low-Voltage reset
LVD detect leve 2.00 V
☐ Hardware enable WDT after power-on-reset
Watch-Dog-Timer prescal 256

Delay 3 sec
Re-Count Infi
☑ Auto reload the target file
☐ Reload and download when target file is modified

```
00000h  02 00 03 75 84 01 7E 44 20 00 E4 1B 44 7A 49 B0
00010h  78 F9 7E F8 02 FF 02 00 19 75 F2 00 75 F1 00 05
00020h  80 80 FC FF FF FF FF FF FF FF FF FF FF FF FF FF
00030h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00040h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00050h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00060h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00070h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00080h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00090h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000A0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000B0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000E0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000F0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

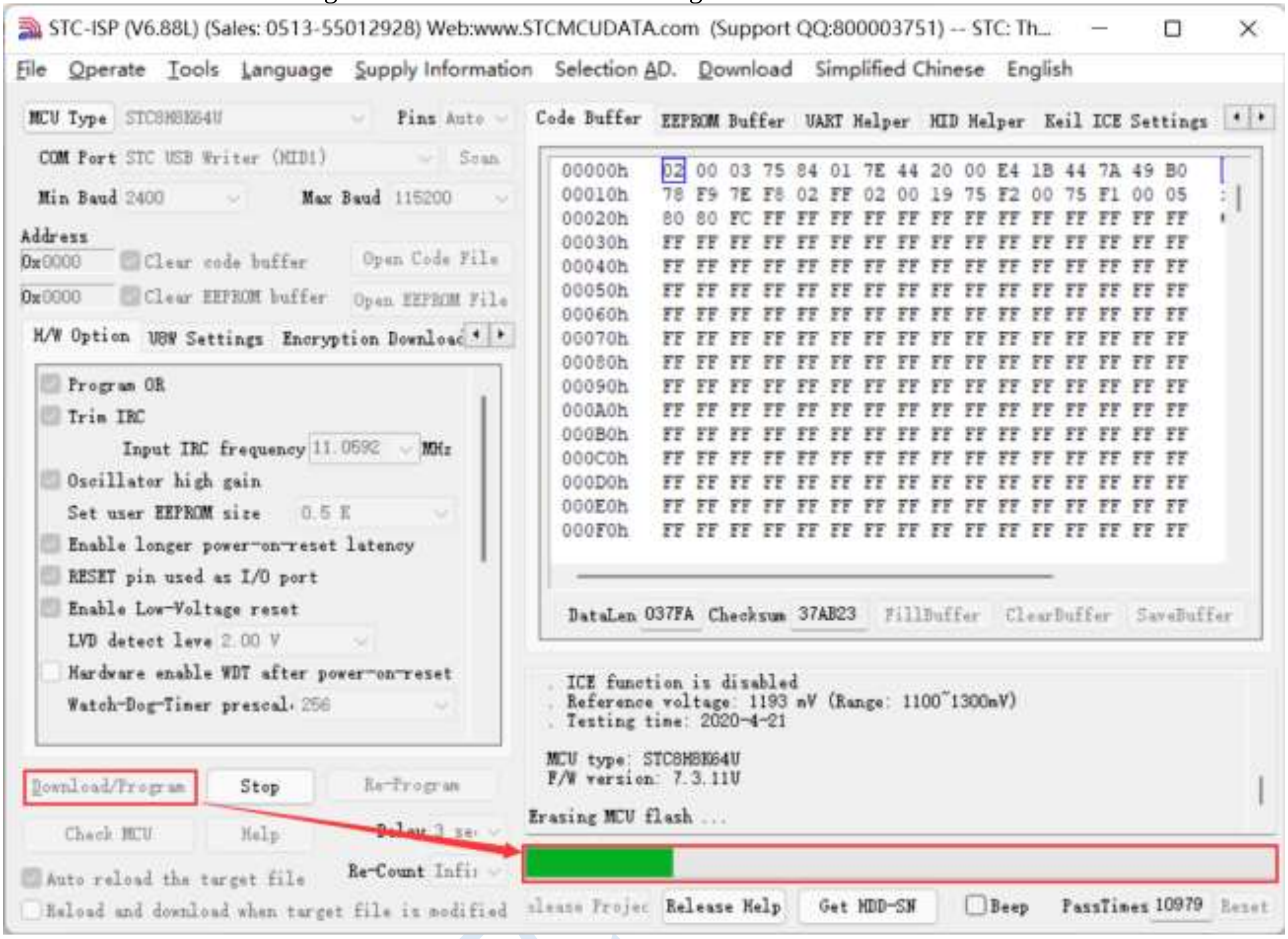
DataLen 037FA Checksum 37AB23

```
MCU Type : STC8H8K64U
About this MCU:
  Only version B and later support DMA, RTC, TFT functions
  USB simulation is only supported by version B and later
Open file "D:\Demo\Demo.hex" OK !
```

D:\Demo\Demo.hex

☐ Beep
PassTimes 10977

3. Click the "Download / Program" button to start downloading the user code.



MCU Type STC8H8K64U Pins Auto

COM Port STC USB Writer (HID1)

Min Baud 2400 Max Baud 115200

Address

0x0000 ☑ Clear code buffer

0x0000 ☑ Clear EEPROM buffer

☑ Program OR

☑ Trim IRC

Input IRC frequency 11.0592 MHz

☑ Oscillator high gain

Set user EEPROM size 0.5 K

☑ Enable longer power-on-reset latency

☑ RESET pin used as I/O port

☑ Enable Low-Voltage reset

LVD detect leve 2.00 V

□ Hardware enable WDT after power-on-reset

Watch-Dog-Timer prescal 256

Delay 3 se

☑ Auto reload the target file Re-Count Infi

□ Reload and download when target file is modified

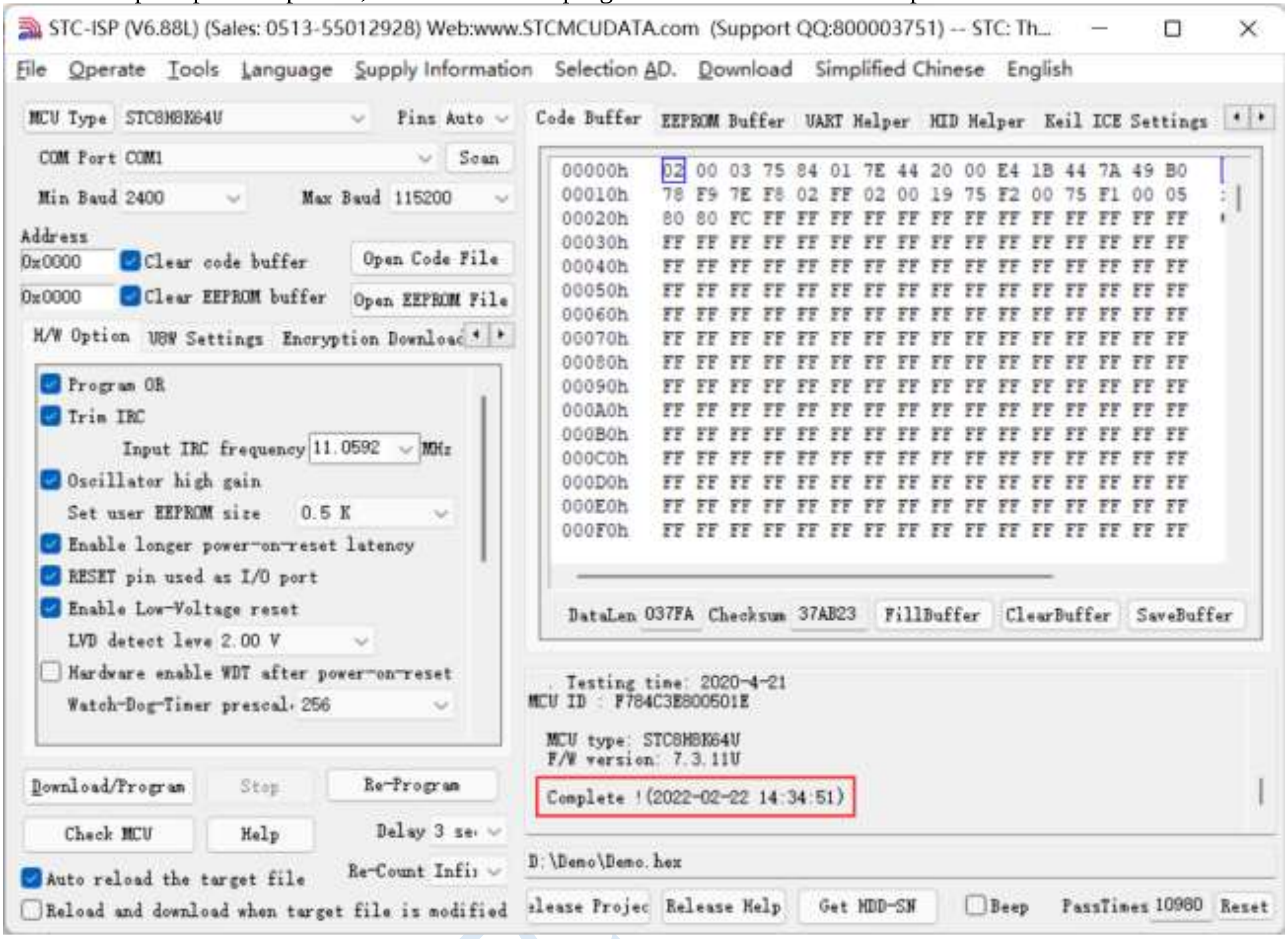
```
00000h  02 00 03 75 84 01 7E 44 20 00 E4 1B 44 7A 49 B0
00010h  78 F9 7E F8 02 FF 02 00 19 75 F2 00 75 F1 00 05
00020h  80 80 FC FF FF FF FF FF FF FF FF FF FF FF FF FF
00030h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00040h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00050h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00060h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00070h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00080h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00090h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000A0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000B0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000E0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000F0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

DataLen 037FA Checksum 37AB23

```
 . ICE function is disabled
 . Reference voltage: 1193 mV (Range: 1100~1300mV)
 . Testing time: 2020-4-21

 MCU type: STC8H8K64U
 F/W version: 7.3.11U

Erasing MCU flash ...
```

□ Beep PassTimes 10979

4. Until the prompt "Complete!", it means that the program code download is complete.



STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...
File Operate Tools Language Supply Information Selection AD. Download Simplified Chinese English
Scan
Open Code File
Open EEPROM File
H/W Option USB Settings Encryption Downloac
Code Buffer EEPROM Buffer UART Helper HID Helper Keil ICE Settings
FillBuffer ClearBuffer SaveBuffer
Download/Program Stop Re-Program
Check MCU Help
elease Projec Release Help Get HDD-SN Reset

MCU Type STC8H8K64U Pins Auto
COM Port COM1
Min Baud 2400 Max Baud 115200
Address
0x0000 ☑ Clear code buffer
0x0000 ☑ Clear EEPROM buffer

☑ Program OR
☑ Trim IRC
Input IRC frequency 11.0592 MHz
☑ Oscillator high gain
Set user EEPROM size 0.5 K
☑ Enable longer power-on-reset latency
☑ RESET pin used as I/O port
☑ Enable Low-Voltage reset
LVD detect leve 2.00 V
☐ Hardware enable WDT after power-on-reset
Watch-Dog-Timer prescal 256

Delay 3 se
Re-Count Infi
☑ Auto reload the target file
☐ Reload and download when target file is modified

```
00000h  02 00 03 75 84 01 7E 44 20 00 E4 1B 44 7A 49 B0
00010h  78 F9 7E F8 02 FF 02 00 19 75 F2 00 75 F1 00 05
00020h  80 80 FC FF FF FF FF FF FF FF FF FF FF FF FF FF
00030h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00040h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00050h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00060h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00070h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00080h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00090h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000A0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000B0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000E0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000F0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

DataLen 037FA Checksum 37AB23

```
  Testing time: 2020-4-21
MCU ID : F784C3E800501E

 MCU type: STC8H8K64U
 F/W version: 7.3.11U

 Complete !(2022-02-22 14:34:51)
```

D:\Demo\Demo.hex

☐ Beep PassTimes 10980

Appendix F USB emulation step demonstration

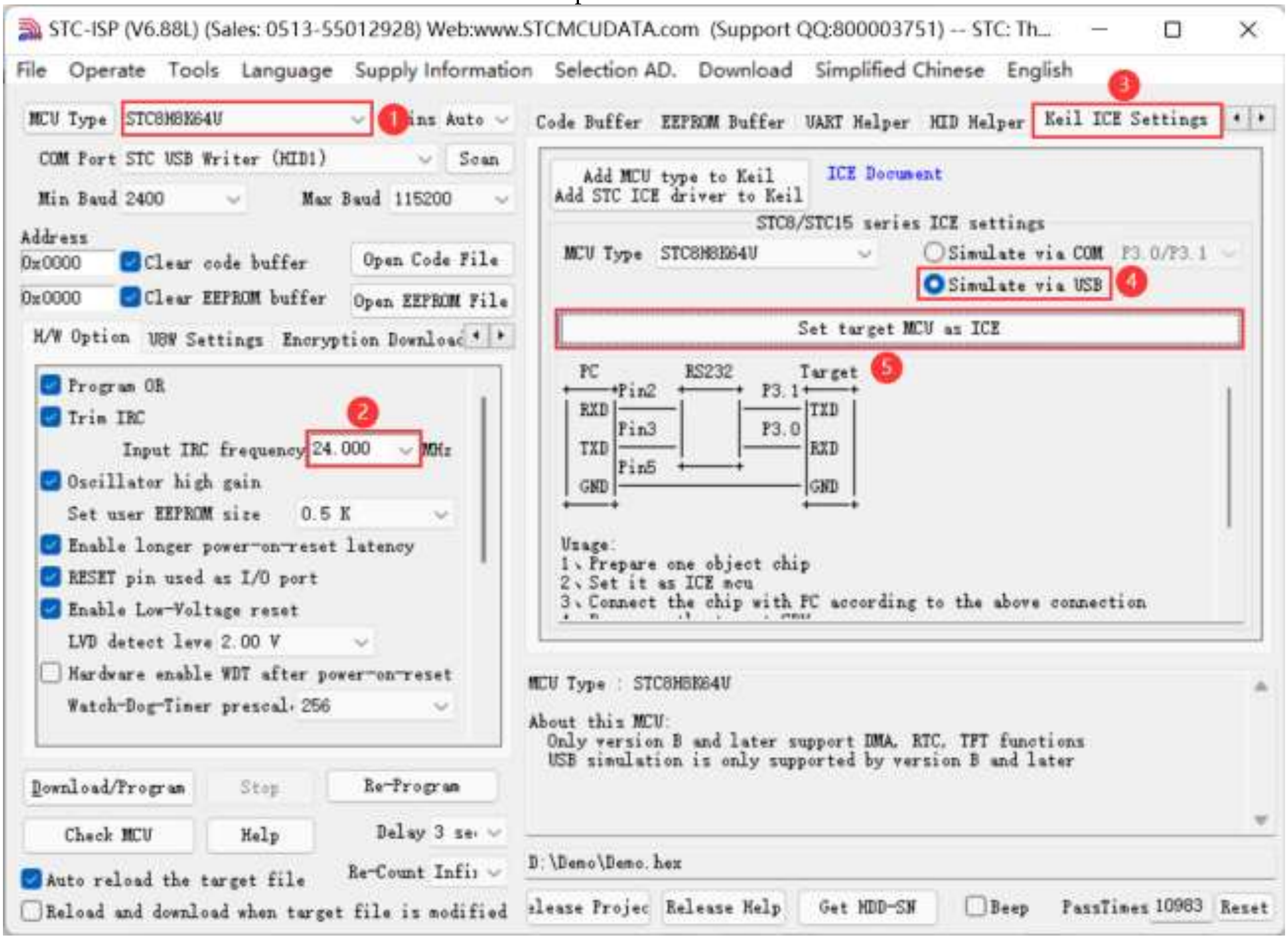
All of the B version chips of the STC8H8K64U series and subsequent versions support the USB direct emulation function.

The specific operation steps are as follows:

1. Install the latest STC simulation driver in the Keil environment

Download the latest STC-ISP download software from the official website (www.STCMCUDATA.com), and follow the steps in Appendix A to install the STC emulation driver

2. Use serial ISP or USB ISP to make emulation chips



STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...

MCU Type STC8H8K64U

COM Port STC USB Writer (HID1)

Min Baud 2400 Max Baud 115200

Address 0x0000 ☑ Clear code buffer

0x0000 ☑ Clear EEPROM buffer

☑ Program OR

☑ Trim IRC

Input IRC frequency 24.000 MHz

☑ Oscillator high gain

Set user EEPROM size 0.5 K

☑ Enable longer power-on-reset latency

☑ RESET pin used as I/O port

☑ Enable Low-Voltage reset

LVD detect leve 2.00 V

□ Hardware enable WDT after power-on-reset

Watch-Dog-Timer prescal: 256

Delay 3 se

☑ Auto reload the target file

Re-Count Infi

□ Reload and download when target file is modified

STC8/STC15 series ICE settings

MCU Type STC8H8K64U

○ Simulate via COM P3.0/P3.1

● Simulate via USB

Usage:

1、Prepare one object chip

2、Set it as ICE mcu

3、Connect the chip with PC according to the above connection

MCU Type : STC8H8K64U

About this MCU:

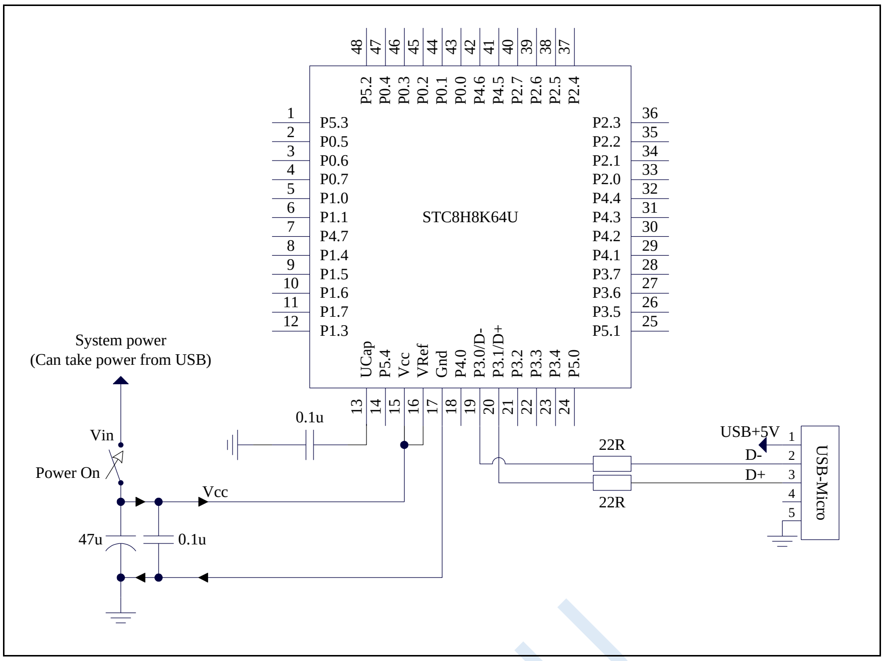
Only version B and later support DMA, RTC, TFT functions

USB simulation is only supported by version B and later

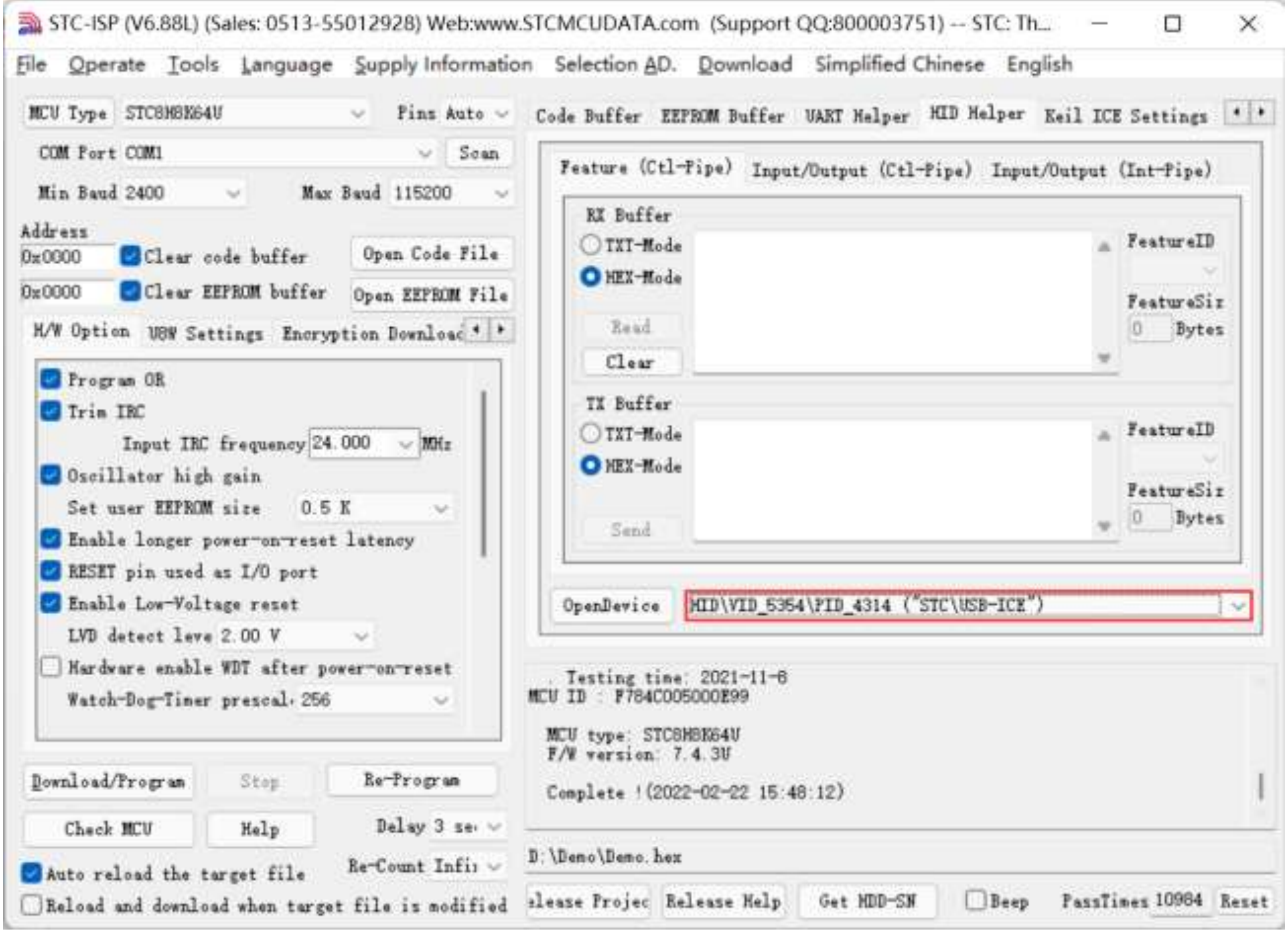
D:\Demo\Demo.hex

□ Beep PassTimes 10983

3. Referring to the connection method in the figure below, connect the target MCU with the emulated chip to the USB port of the computer.



When the "STC\USB-ICE" device can be correctly displayed in the HID assistant in the downloaded software, it means the hardware is connected correctly.


STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...
File Operate Tools Language Supply Information Selection AD. Download Simplified Chinese English
Scan
Open Code File
Open EEPROM File
H/W Option USB Settings Encryption Downloac
Code Buffer EEPROM Buffer UART Helper HID Helper Keil ICE Settings
Feature (Ctl-Pipe) Input/Output (Ctl-Pipe) Input/Output (Int-Pipe)
RX Buffer
Read
Clear
TX Buffer
Send
OpenDevice
Download/Program
Stop
Re-Program
Check MCU
Help
lease Projec
Release Help
Get MDD-SN
Reset

MCU Type STC8H8K64U Pins Auto
COM Port COM1
Min Baud 2400 Max Baud 115200
Address
0x0000 ☑ Clear code buffer
0x0000 ☑ Clear EEPROM buffer

☑ Program OR
☑ Trim IRC
Input IRC frequency 24.000 MHz
☑ Oscillator high gain
Set user EEPROM size 0.5 K
☑ Enable longer power-on-reset latency
☑ RESET pin used as I/O port
☑ Enable Low-Voltage reset
LVD detect leve 2.00 V
□ Hardware enable WDT after power-on-reset
Watch-Dog-Timer prescal 256

Delay 3 se
Re-Count Infi
☑ Auto reload the target file
□ Reload and download when target file is modified

○ TXT-Mode
● HEX-Mode
FeatureID
FeatureSiz 0 Bytes

○ TXT-Mode
● HEX-Mode
FeatureID
FeatureSiz 0 Bytes

HID\VID_5354\PID_4314 ("STC\USB-ICE")

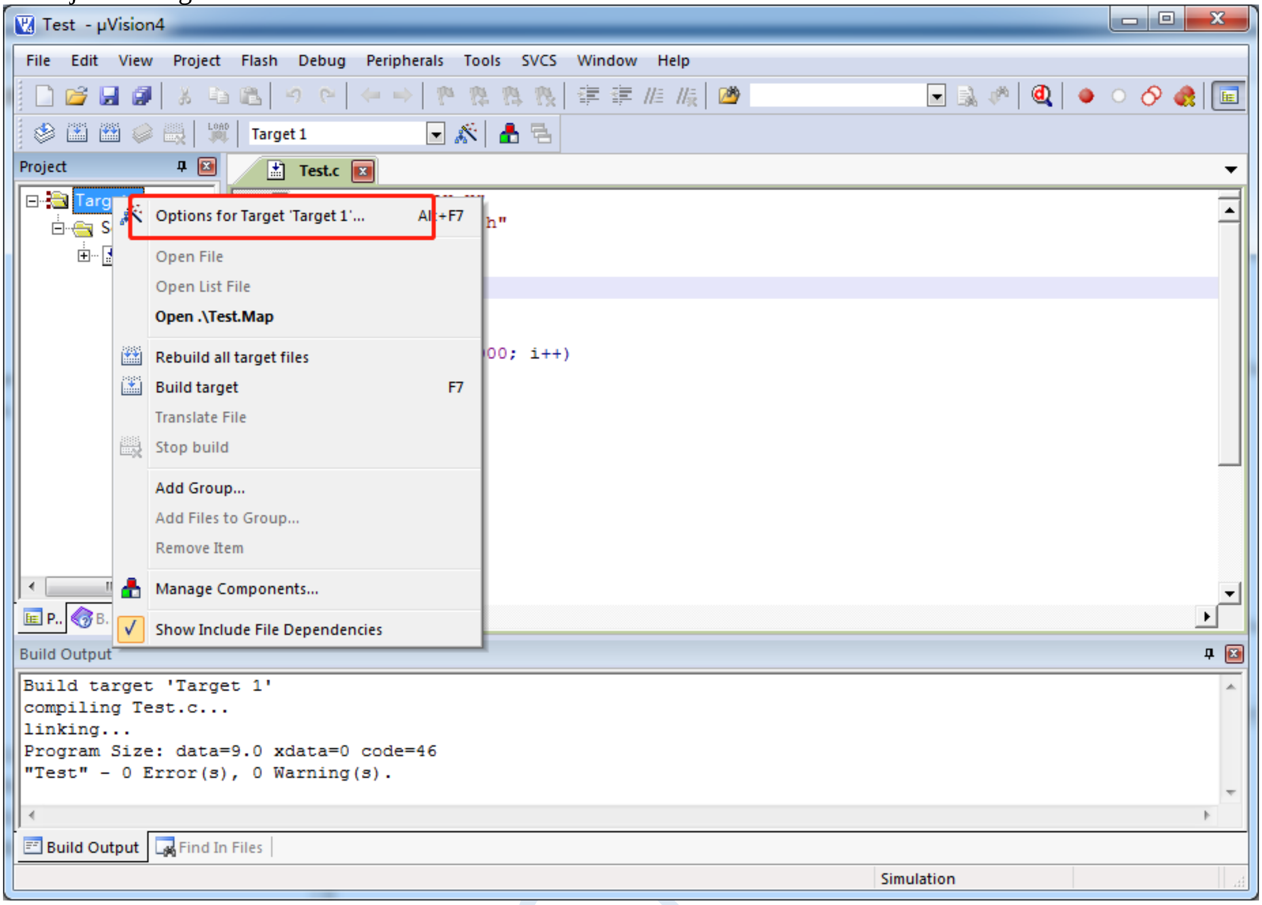
```
  Testing time: 2021-11-6
MCU ID : F784C005000E99

MCU type: STC8H8K64U
F/W version: 7.4.3U

Complete !(2022-02-22 15:48:12)
```

D:\Demo\Demo.hex

□ Beep PassTimes 10984

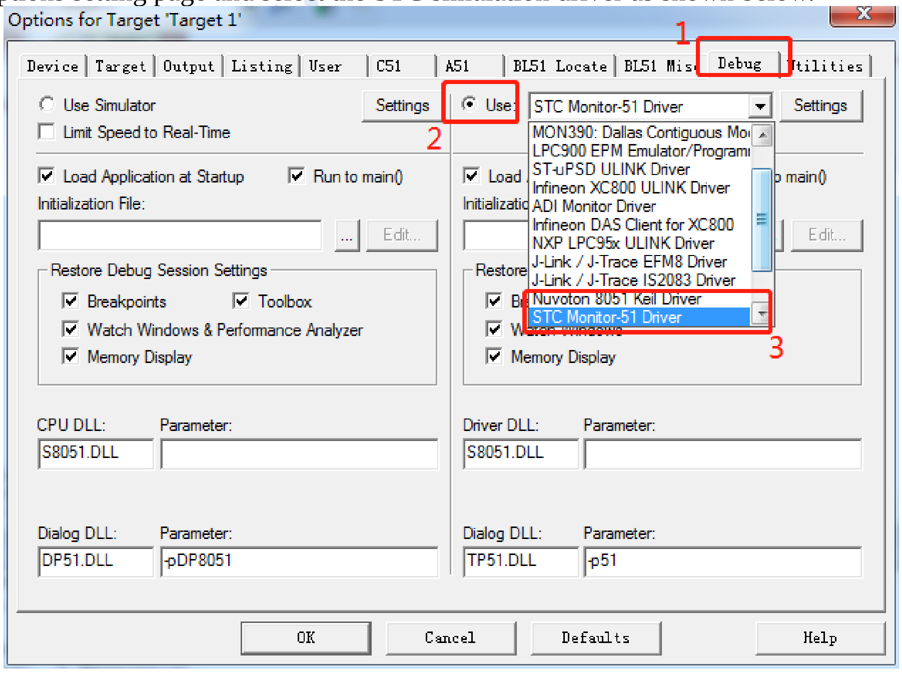
Note: When the displayed device name is "STC\USB-ISP", it means that the target chip is in the USB-ISP download mode, please make sure that the P3.2 port is at a high level and reconnect to the USB.

4. Project settings in Keil software



```
Build target 'Target 1'
compiling Test.c...
linking...
Program Size: data=9.0 xdata=0 code=46
"Test" - 0 Error(s), 0 Warning(s).
```

Open the project options setting page and select the STC simulation driver as shown below.



○ Use Simulator
□ Limit Speed to Real-Time
● Use: STC Monitor-51 Driver

☑ Load Application at Startup ☑ Run to main()
Initialization File:

Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watch Windows & Performance Analyzer
☑ Memory Display

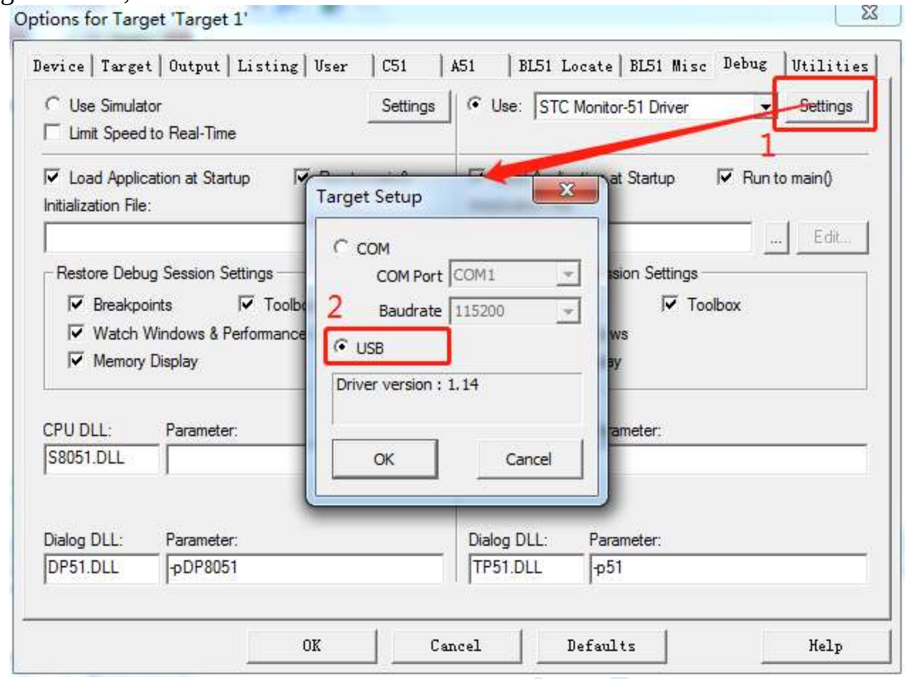
CPU DLL: S8051.DLL Parameter:
Dialog DLL: DP51.DLL Parameter: -pDP8051

☑ Load ... main()
Initialization File:

Restore ...
☑ Br...
☑ W...
☑ Memory Display

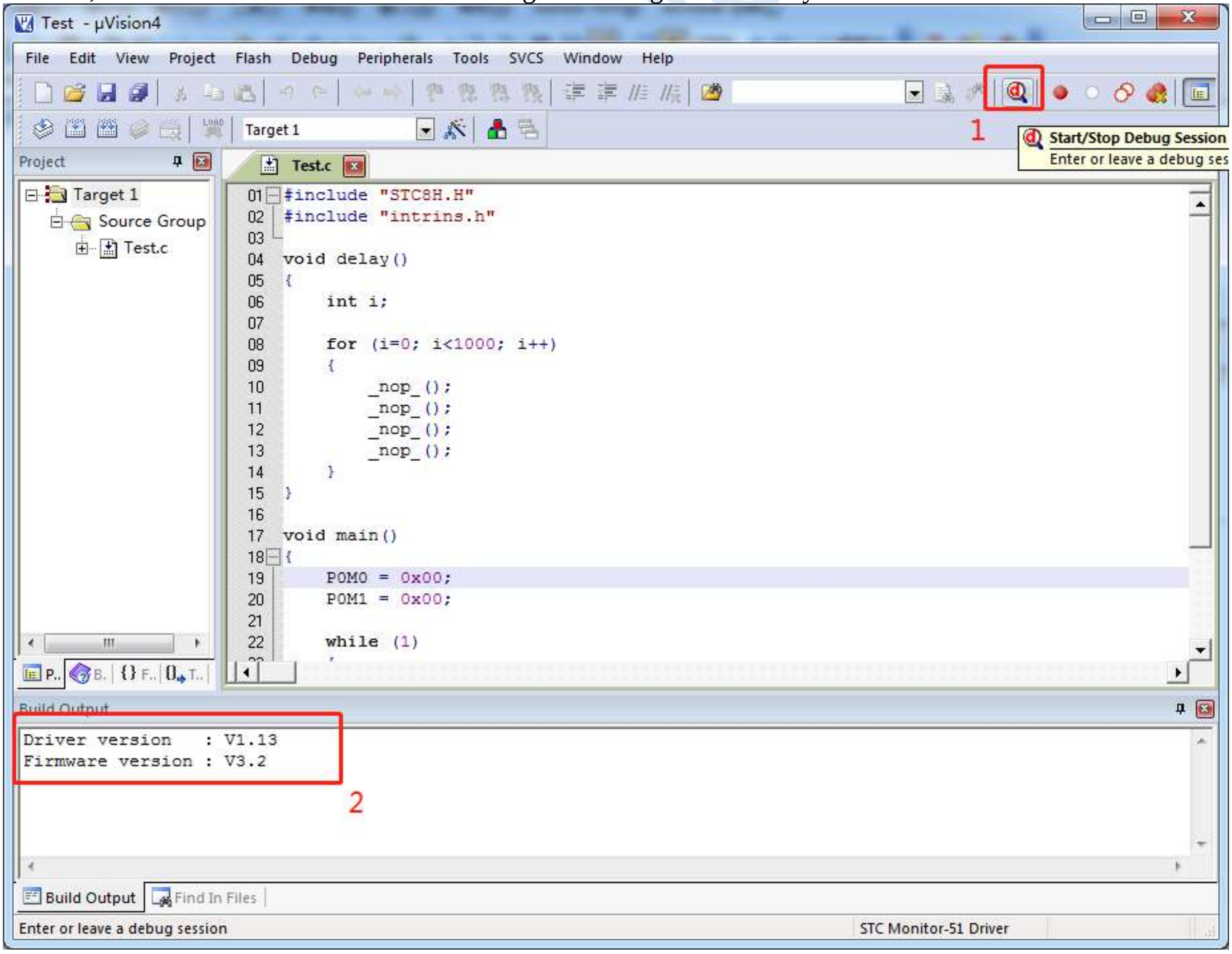
Driver DLL: S8051.DLL Parameter:
Dialog DLL: TP51.DLL Parameter: -p51

As shown in the figure below, select the USB emulation interface of STC.



5. After the setting is completed, start the simulation

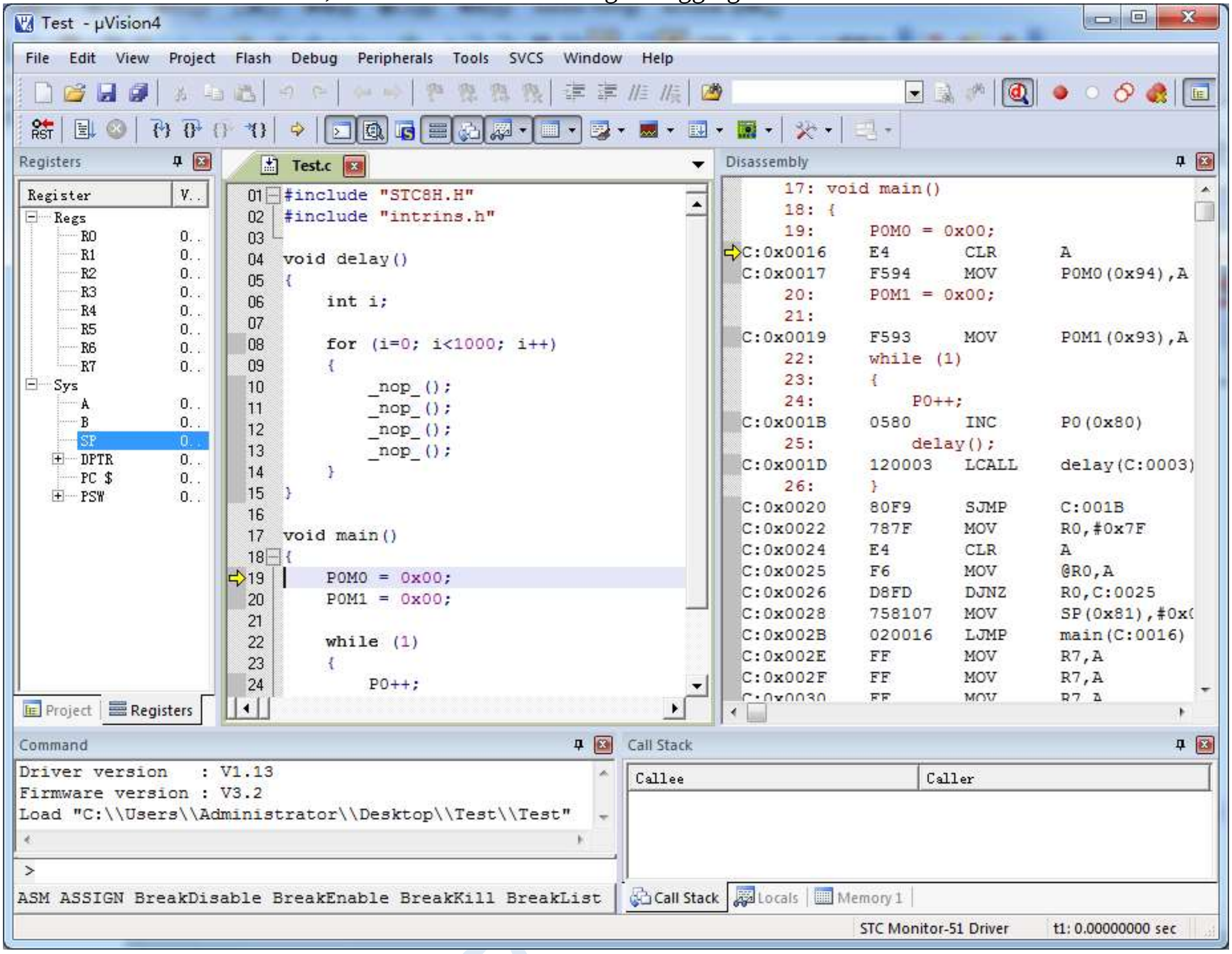
Click the Start Simulation button in the Keil software, if the version number is displayed correctly in the output window, it means that the connection and settings are configured correctly



```
#include "STC8H.H"
#include "intrins.h"

void delay()
{
    int i;

    for (i=0; i<1000; i++)
    {
        _nop_();
        _nop_();
        _nop_();
        _nop_();
    }
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;

    while (1)
```

```
Driver version   : V1.13
Firmware version : V3.2
```

After the code is downloaded, it will enter the following debugging screen



```
Register    V..
Regs
  R0        0..
  R1        0..
  R2        0..
  R3        0..
  R4        0..
  R5        0..
  R6        0..
  R7        0..
Sys
  A         0..
  B         0..
  SP        0..
  DPTR      0..
  PC $      0..
  PSW       0..
```

```
#include "STC8H.H"
#include "intrins.h"

void delay()
{
    int i;

    for (i=0; i<1000; i++)
    {
        _nop_();
        _nop_();
        _nop_();
        _nop_();
    }
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;

    while (1)
    {
        P0++;
```

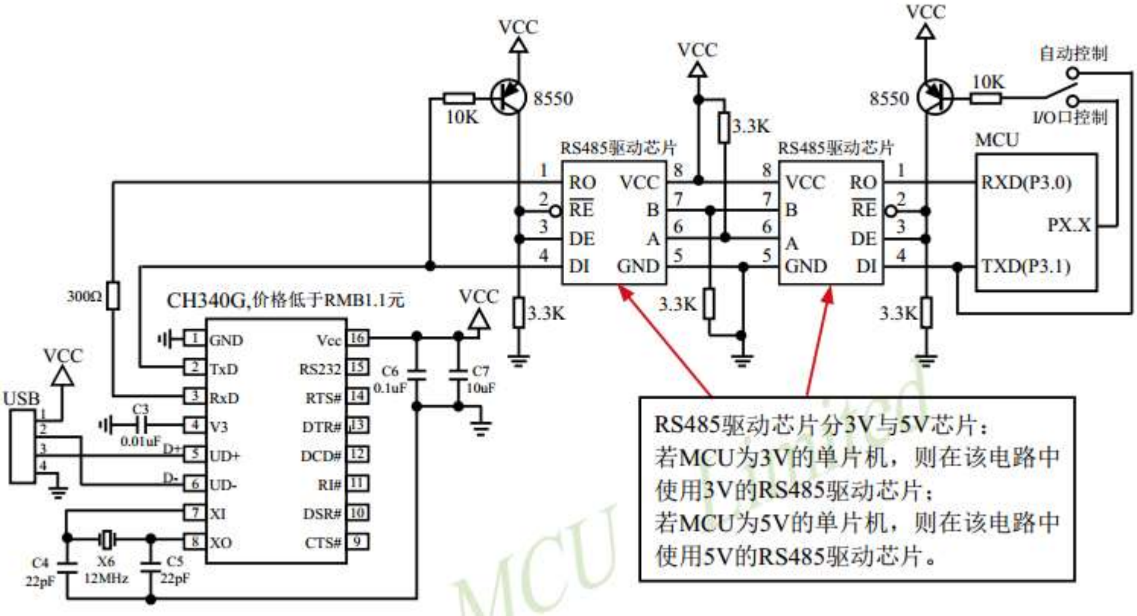
```
    17: void main()
    18: {
    19:     P0M0 = 0x00;
C:0x0016    E4       CLR      A
C:0x0017    F594     MOV      P0M0(0x94),A
    20:     P0M1 = 0x00;
    21:
C:0x0019    F593     MOV      P0M1(0x93),A
    22:     while (1)
    23:     {
    24:         P0++;
C:0x001B    0580     INC      P0(0x80)
    25:         delay();
C:0x001D    120003   LCALL    delay(C:0003)
    26:     }
C:0x0020    80F9     SJMP     C:001B
C:0x0022    787F     MOV      R0,#0x7F
C:0x0024    E4       CLR      A
C:0x0025    F6       MOV      @R0,A
C:0x0026    D8FD     DJNZ     R0,C:0025
C:0x0028    758107   MOV      SP(0x81),#0x(
C:0x002B    020016   LJMP     main(C:0016)
C:0x002E    FF       MOV      R7,A
C:0x002F    FF       MOV      R7,A
C:0x0030    FF       MOV      R7,A
```

```
Driver version   : V1.13
Firmware version : V3.2
Load "C:\\Users\\Administrator\\Desktop\\Test\\Test"
>
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

```
Callee          Caller
```

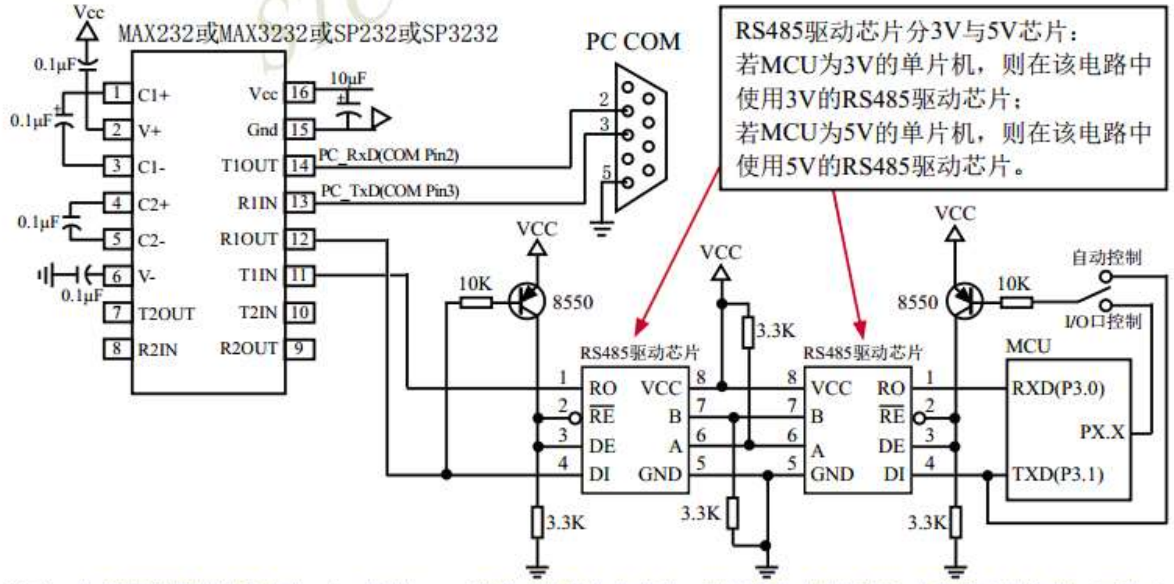
Appendix G RS485 Automatic control or I/O port control circuit diagram

1. Use the USB to serial port to connect the computer's RS485 to control the download circuit diagram (automatic control or I/O port control)



RS485驱动芯片分3V与5V芯片：
若MCU为3V的单片机，则在该电路中使用3V的RS485驱动芯片；
若MCU为5V的单片机，则在该电路中使用5V的RS485驱动芯片。

2. Use RS232 to serial port to connect the computer's RS485 to control the download circuit diagram (automatic control or I/O port control)



RS485驱动芯片分3V与5V芯片：
若MCU为3V的单片机，则在该电路中使用3V的RS485驱动芯片；
若MCU为5V的单片机，则在该电路中使用5V的RS485驱动芯片。

注意：如果要设置单片机某个I/O口控制RS485发送或接收命令有效，则必须将单片机焊入电路板之前先用U8下载工具结合电脑ISP软件对该单片机进行“RS485控制”设置并烧录一下（如上节所述），否则将单片机实现不了RS485控制功能。

建议用户将本节所述“RS485控制下载线路图(自动控制或I/O口控制)”设计到您的用户板上

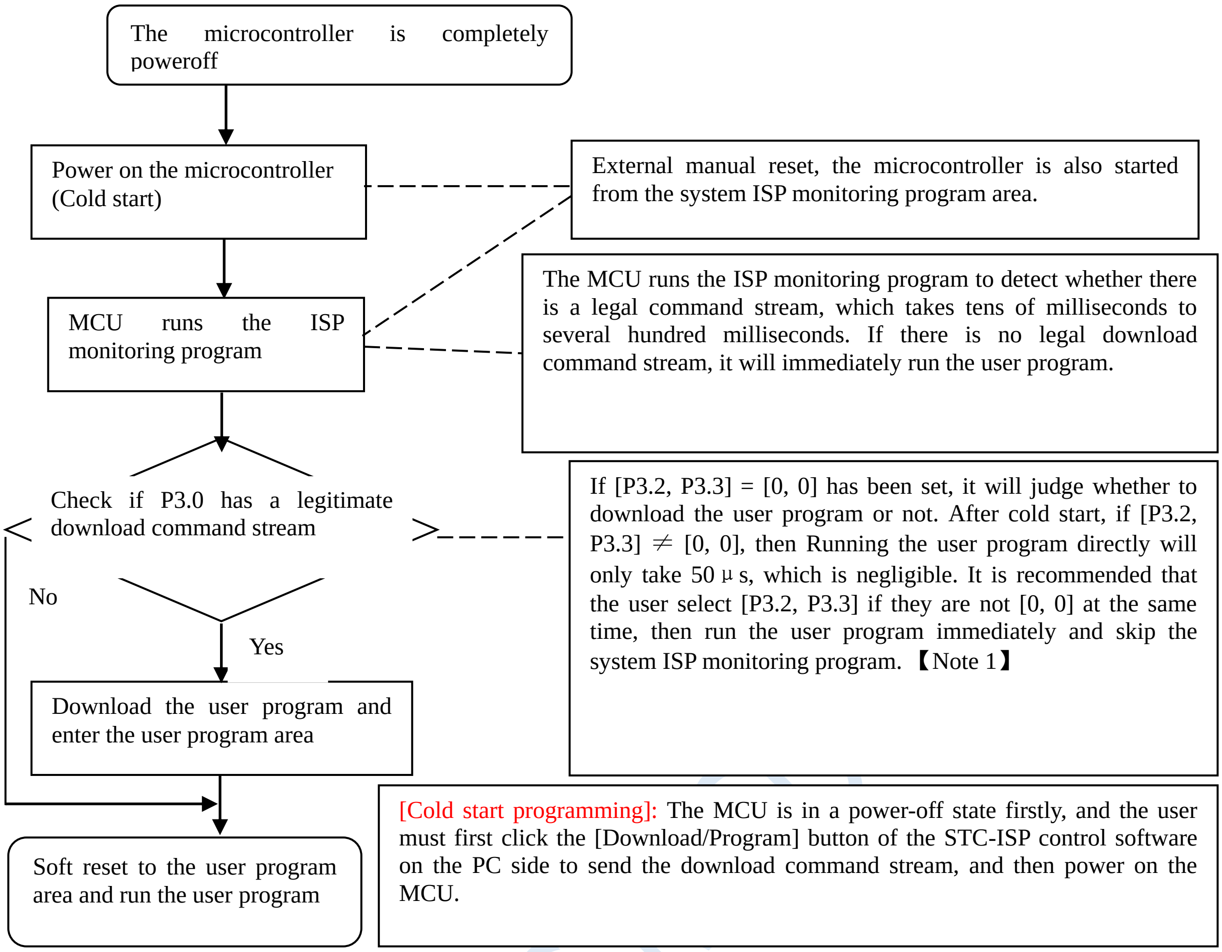
Appendix H STC tool instruction manual

H.1 Overview

U8W/U8W-Mini is a series of programming tools that integrates online download and offline download. STC Universal USB to Serial Port Tool is a programming tool that supports online download and online simulation.

Tool type	Online Download	Offline download	Burner download	Online simulation	Price(CNY)
U8W	support	support	support	Need to set pass-through mode	100 yuan
U8W-Mini	support	support	not support	Need to set pass-through mode	50 yuan
Universal USB to serial port	support	not support	not support	support	30 yuan

H.2 System Programmable (ISP) Process Description


The microcontroller is completely poweroff
Power on the microcontroller (Cold start)
MCU runs the ISP monitoring program
Check if P3.0 has a legitimate download command stream
No
Yes
Download the user program and enter the user program area
Soft reset to the user program area and run the user program

External manual reset, the microcontroller is also started from the system ISP monitoring program area.

The MCU runs the ISP monitoring program to detect whether there is a legal command stream, which takes tens of milliseconds to several hundred milliseconds. If there is no legal download command stream, it will immediately run the user program.

If [P3.2, P3.3] = [0, 0] has been set, it will judge whether to download the user program or not. After cold start, if [P3.2, P3.3] ≠ [0, 0], then Running the user program directly will only take 50μs, which is negligible. It is recommended that the user select [P3.2, P3.3] if they are not [0, 0] at the same time, then run the user program immediately and skip the system ISP monitoring program.【Note 1】

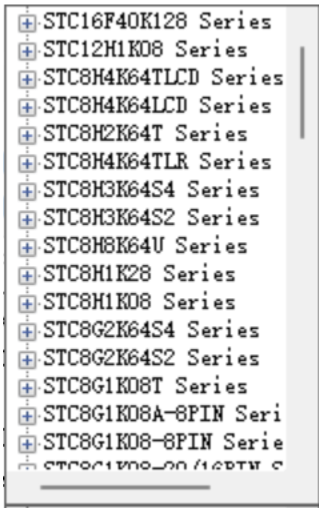
[Cold start programming]: The MCU is in a power-off state firstly, and the user must first click the [Download/Program] button of the STC-ISP control software on the PC side to send the download command stream, and then power on the MCU.

Note: Because [P3.0, P3.1] is used for download/simulation (download/simulation interface is only available [P3.0, P3.1]), it is recommended that users put serial port 1 on P3.6/P3.7 Or P1.6/P1.7, if the user does not want to switch and insists on using P3.0/P3.1 to work or communicate as the serial port 1, be sure to check the "Next cold start, when downloading the program, The program can be downloaded only when P3.2/P3.3 is 0/0".【Note 1】

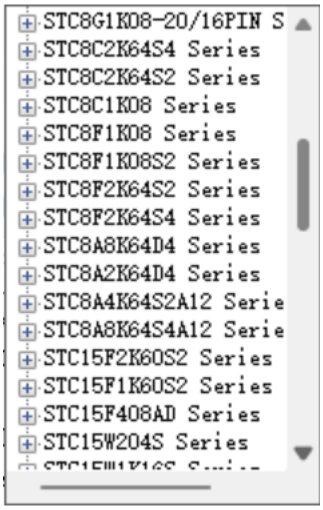
[Note 1]: The programming protection pins of STC15, STC8 and later new chips are P3.2/P3.3, and the programming protection pins of earlier chips are P1.0/P1.1.

H.3 USB type online/offline download tool U8W/U8W-Mini

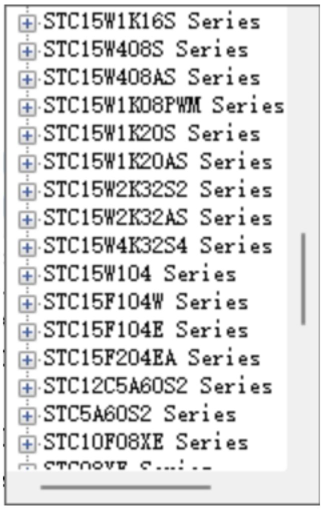
The application range of U8W/U8W-Min can support all current MCU series of STC, and the Flash program space and EEPROM data space are not restricted. Support includes the following and upcoming STC full series chips:



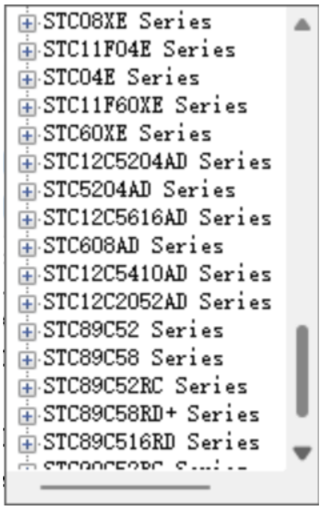
STC16F40K128 Series
STC12H1K08 Series
STC8H4K64TLCD Series
STC8H4K64LCD Series
STC8H2K64T Series
STC8H4K64TLR Series
STC8H3K64S4 Series
STC8H3K64S2 Series
STC8H8K64U Series
STC8H1K28 Series
STC8H1K08 Series
STC8G2K64S4 Series
STC8G2K64S2 Series
STC8G1K08T Series
STC8G1K08A-8PIN Seri
STC8G1K08-8PIN Serie



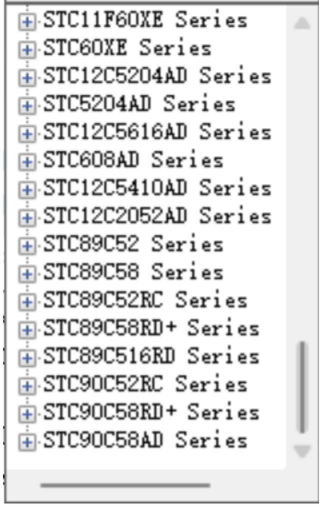
STC8G1K08-20/16PIN S
STC8C2K64S4 Series
STC8C2K64S2 Series
STC8C1K08 Series
STC8F1K08 Series
STC8F1K08S2 Series
STC8F2K64S2 Series
STC8F2K64S4 Series
STC8A8K64D4 Series
STC8A2K64D4 Series
STC8A4K64S2A12 Serie
STC8A8K64S4A12 Serie
STC15F2K60S2 Series
STC15F1K60S2 Series
STC15F408AD Series
STC15W204S Series



STC15W1K16S Series
STC15W408S Series
STC15W408AS Series
STC15W1K08PWM Series
STC15W1K20S Series
STC15W1K20AS Series
STC15W2K32S2 Series
STC15W2K32AS Series
STC15W4K32S4 Series
STC15W104 Series
STC15F104W Series
STC15F104E Series
STC15F204EA Series
STC12C5A60S2 Series
STC5A60S2 Series
STC10F08XE Series



STC08XE Series
STC11F04E Series
STC04E Series
STC11F60XE Series
STC60XE Series
STC12C5204AD Series
STC5204AD Series
STC12C5616AD Series
STC608AD Series
STC12C5410AD Series
STC12C2052AD Series
STC89C52 Series
STC89C58 Series
STC89C52RC Series
STC89C58RD+ Series
STC89C516RD Series



STC11F60XE Series
STC60XE Series
STC12C5204AD Series
STC5204AD Series
STC12C5616AD Series
STC608AD Series
STC12C5410AD Series
STC12C2052AD Series
STC89C52 Series
STC89C58 Series
STC89C52RC Series
STC89C58RD+ Series
STC89C516RD Series
STC90C52RC Series
STC90C58RD+ Series
STC90C58AD Series

The offline download tool can be used for downloading without the computer, and can be used for mass production and remote upgrades. The offline download board can support multiple functions such as automatic increment, download limit, and encrypted transmission of user programs.

The following picture shows the front and back views of U8W tools and the front and back views of U8W-Mini:

U8W正面图 U8W反面图 U8W-Mini正反面图

U8W-Mini工具的体积仅有一个U盘大小，其功能与U8W相同，但无锁紧座，价格仅为RMB 50元，欢迎来电订购！

In addition, some wires and tools are used together as follows, such as:

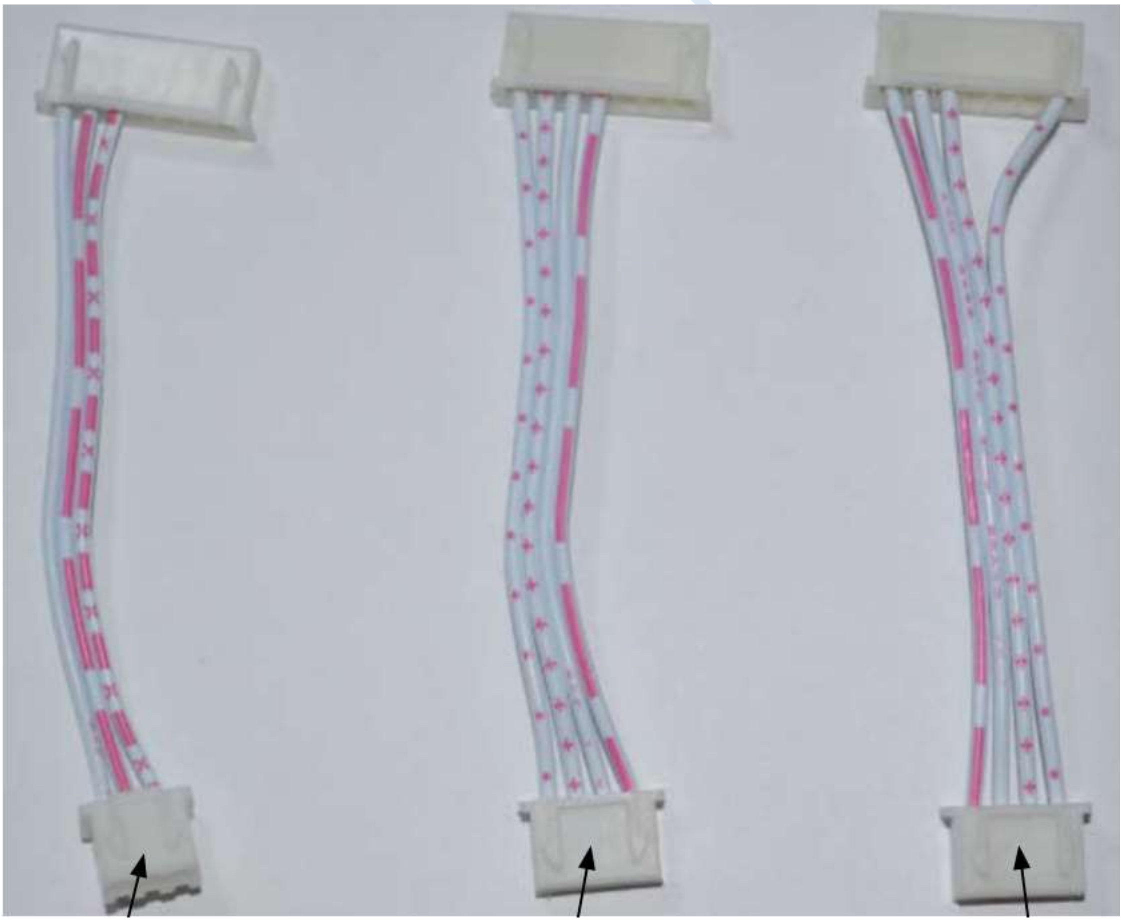
(1) Two-end male USB cable (shown on the left in the figure below) and USB-Micro cable (shown on the right in the

figure below):

Note: This USB cable is a USB enhanced cable specially customized by our company, which can ensure that the download can be successful when directly powered by USB. On the market, some relatively low-quality two-end male USB cables have too much internal resistance and cause a large voltage drop (for example, the voltage of the USB when it is empty is about 5.0V. When using a low-quality USB cable to connect U8W/U8W-Mini/U8 /U8-Mini, the voltage to our download board may drop to 4.2V or lower, causing the chip to be in a reset state and fail to download successfully).

(2) The download cable connecting U8W/U8W-Mini to the user system (ie the connecting cable between U8W/U8W-Mini and the target MCU on the user board), such as

As shown below:



U8W/U8W-Mini and
User systems are independent
Power supply cable

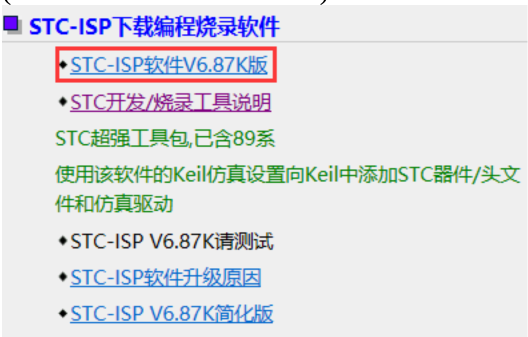
U8W/U8W-Mini to
The user's system power supply
wiring

User system gives
U8W/U8W-Mini
Power supply cable

H3.1 Install U8W/U8W-Mini driver

The U8W/U8W-Mini download board uses a CH340 USB-to-serial universal chip. This saves the trouble that some computers without a serial port must buy a USB to serial port tool to download. But CH340 is the same as other USB-to-serial tools, the driver must be installed before use.

Obtain the driver by downloading the STC-ISP software package
The following is the download location of the STC-ISP software package provided on the STC official website (www.STCMCUDATA.com):



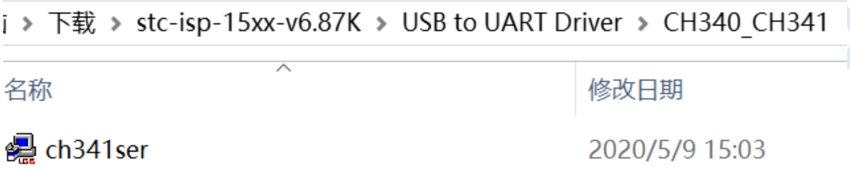
STC-ISP下载编程烧录软件
- STC-ISP软件V6.87K版
- STC开发/烧录工具说明

STC超强工具包,已含89系

使用该软件的Keil仿真设置向Keil中添加STC器件/头文件和仿真驱动

- STC-ISP V6.87K请测试
- STC-ISP软件升级原因
- STC-ISP V6.87K简化版

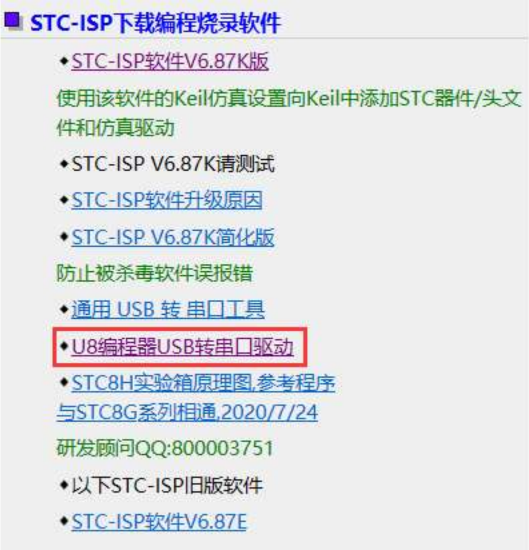
After downloading, decompress, the path of CH340 driver installation package is stc-isp-15xx-v6.87K\USB to UART Driver\CH340_CH341:



i › 下载 › stc-isp-15xx-v6.87K › USB to UART Driver › CH340_CH341

名称	修改日期
ch341ser	2020/5/9 15:03

Download the driver manually through STC's official website or in the latest STC-ISP download software
Download the driver manually on the official website of STC or in the latest STC-ISP download software. The download link of the driver is: U8 programmer USB to serial port driver (http://www.stcmcu.com/STCISP/CH341SER.exe). The driver address on the website and STC-ISP download software is shown in the figure below:



STC-ISP下载编程烧录软件
- STC-ISP软件V6.87K版

使用该软件的Keil仿真设置向Keil中添加STC器件/头文件和仿真驱动

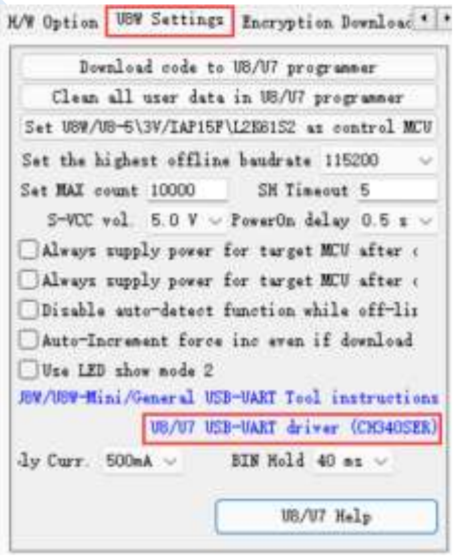
- STC-ISP V6.87K请测试
- STC-ISP软件升级原因
- STC-ISP V6.87K简化版

防止被杀毒软件误报错

- 通用 USB 转 串口工具
- U8编程器USB转串口驱动
- STC8H实验箱原理图,参考程序与STC8G系列相通,2020/7/24

研发顾问QQ:800003751

- 以下STC-ISP旧版软件
- STC-ISP软件V6.87E

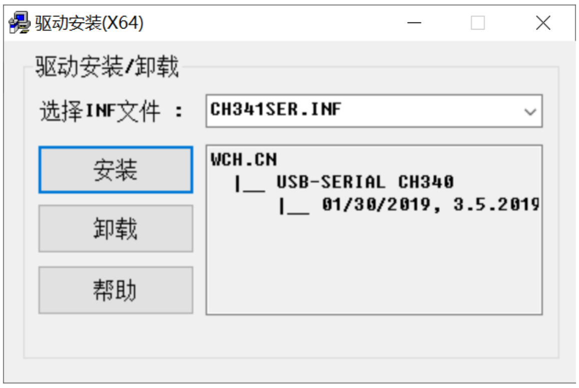


H/W Option
U8W Settings
Encryption
Downloac
Download code to U8/U7 programmer
Clean all user data in U8/U7 programmer
Set U8W/U8-5\3V/IAP15F\L2K61S2 as control MCU
U8/U7 Help

Set the highest offline baudrate 115200
Set MAX count 10000 SN Timeout 5
S-VCC vol 5.0 V PowerOn delay 0.5 s
☐ Always supply power for target MCU after (
☐ Always supply power for target MCU after (
☐ Disable auto-detect function while off-li
☐ Auto-Increment force inc even if download
☐ Use LED show mode 2
J8W/U8W-Mini/General USB-UART Tool instructions
U8/U7 USB-UART driver (CH340SER)
ly Curr. 500mA BIN Hold 40 ms

Install U8W/U8W-Mini driver

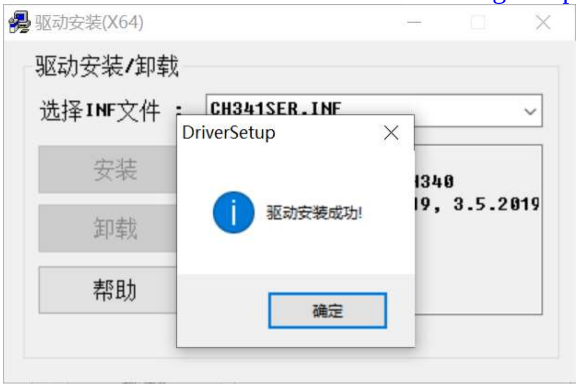
After the driver is downloaded to the machine, double-click the executable program and run it. The interface shown in the figure below appears, click the "Install" button to start the automatic driver installation:



CH341SER.INF

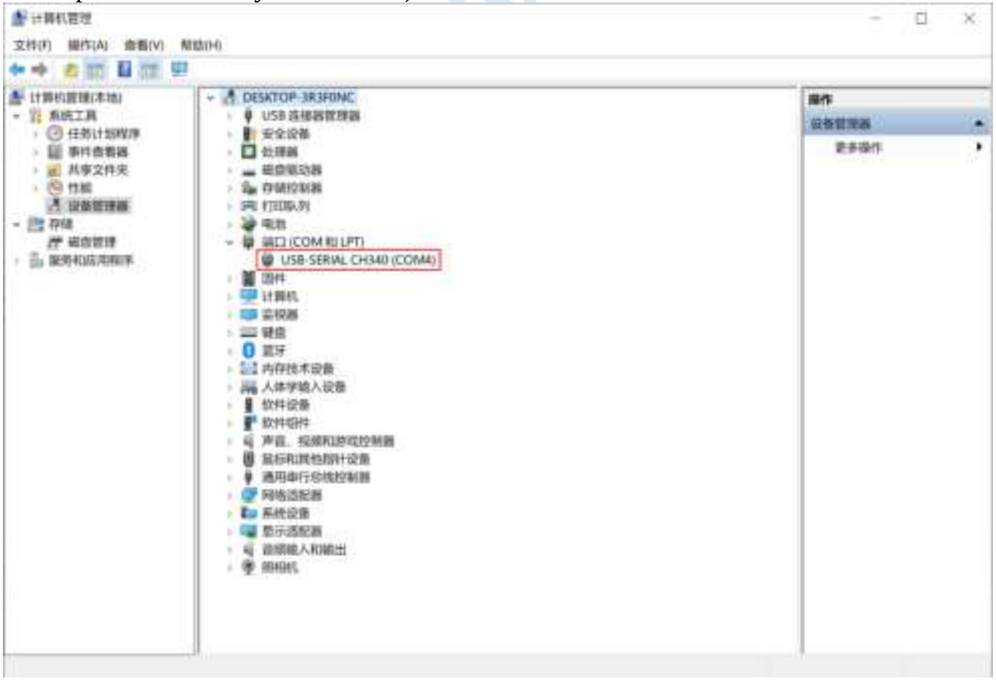
```
WCH.CN
  |__ USB-SERIAL CH340
      |__ 01/30/2019, 3.5.2019
```

Then the driver installation successful dialog box pops up, click the "OK" button to complete the installation:

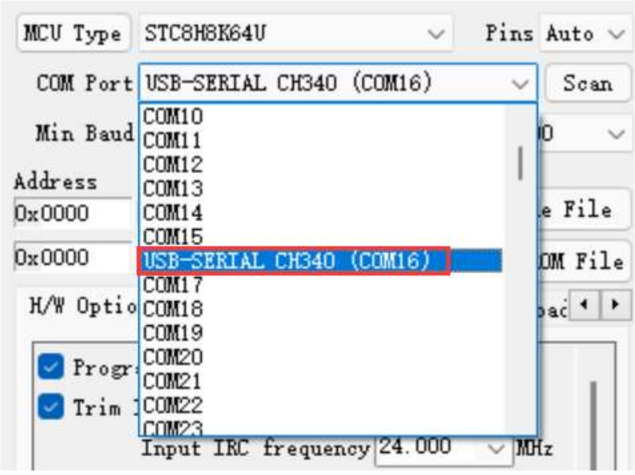


驱动安装成功!

Then use the USB cable provided by STC to connect the U8W/U8W-Mini download board to the computer, open the device manager of the computer, and under the port device category, if there is a device similar to "USB-SERIAL CH340 (COMx)", it means U8W/U8W-Mini can be used normally. As shown in the figure below (different computers, the serial port number may be different):



Note: When using STC-ISP to download software later, the selected serial port number must be the corresponding serial port number, as shown in the figure below:

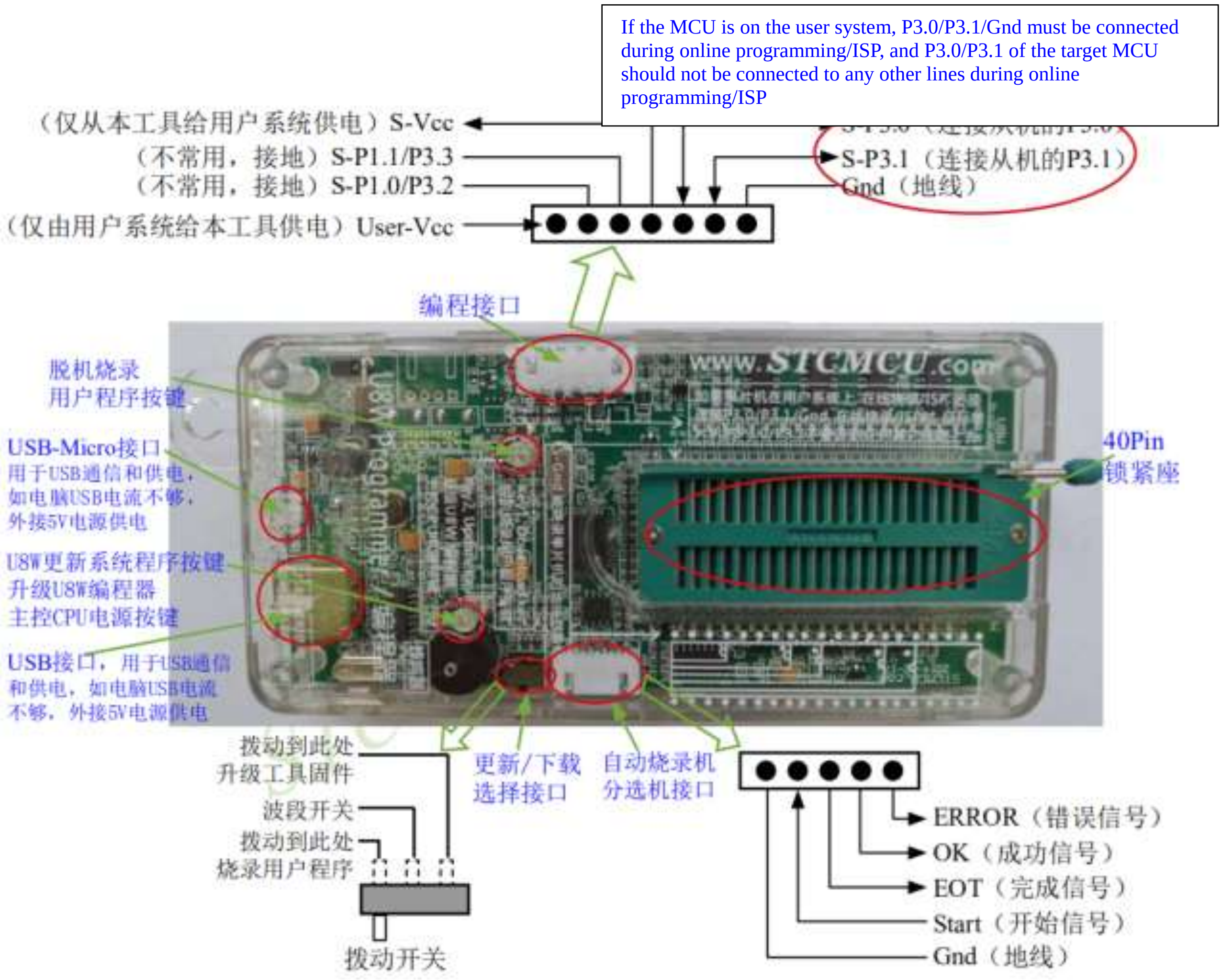


MCU Type: STC8H8K64U
Pins: Auto
COM Port: USB-SERIAL CH340 (COM16)

COM10
COM11
COM12
COM13
COM14
COM15
USB-SERIAL CH340 (COM16)
COM17
COM18
COM19
COM20
COM21
COM22
COM23

H3.2 U8W function introduction

The main interfaces and functions of U8W tools are described in detail below:



If the MCU is on the user system, P3.0/P3.1/Gnd must be connected during online programming/ISP, and P3.0/P3.1 of the target MCU should not be connected to any other lines during online programming/ISP

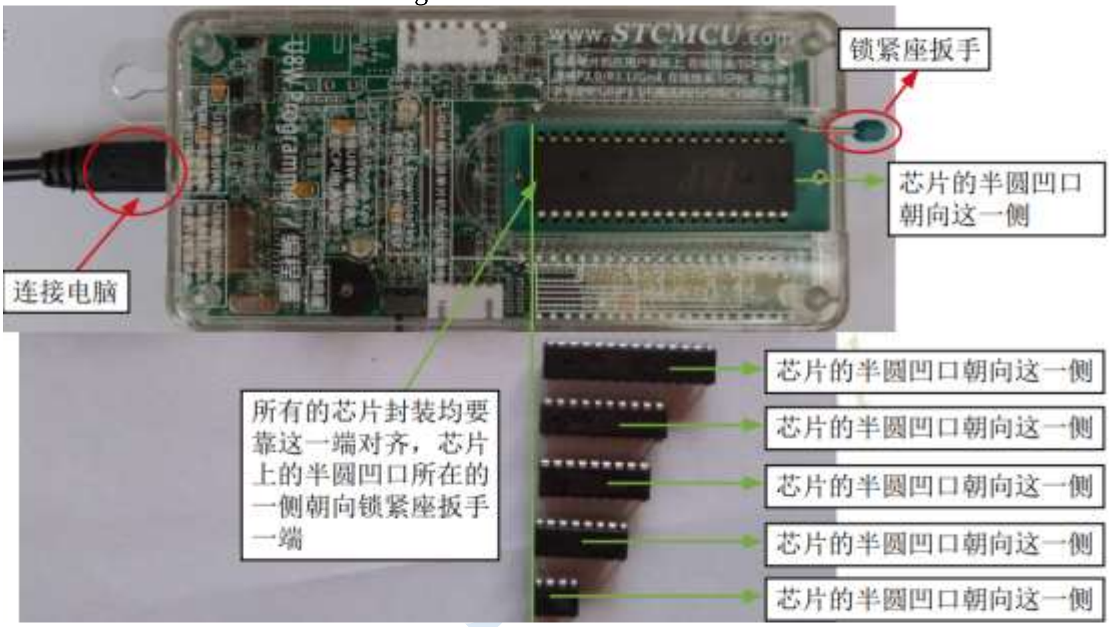
Programming interface: Use different download cables to connect the U8W download board and the user system according to different power supply methods.

U8W update system program button: used to update U8W tools. When there is a new version of U8W firmware, you need to press this button to update the U8W main control chip (note: you must first set the toggle switch on the update/download selection interface Toggle to upgrade tool firmware).

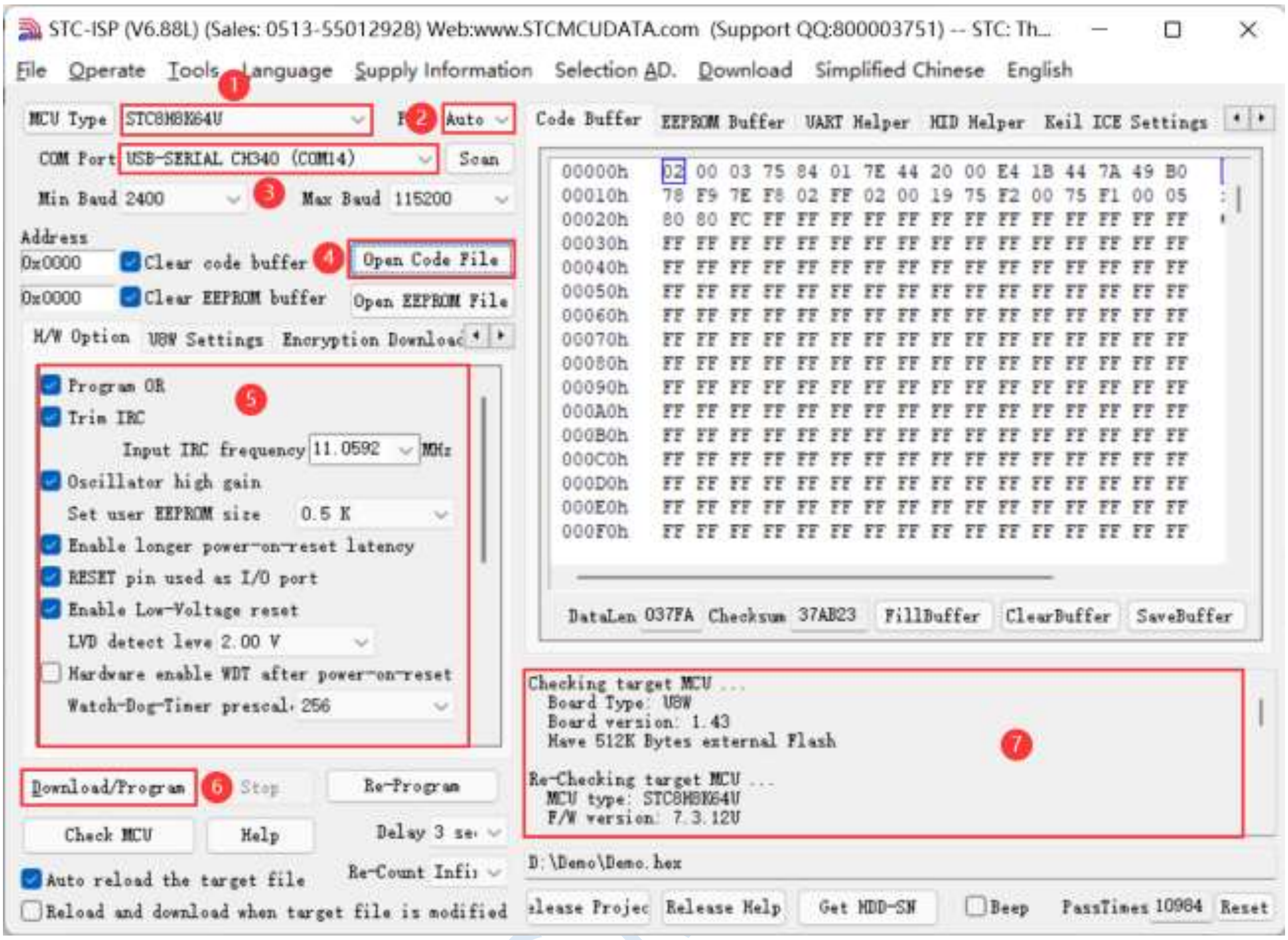
Offline download user program button: Start offline download button. First, the PC downloads the offline code to the U8W board, and then uses the download cable to connect the user system to the U8W, and then press this button to start the offline download (the user code will also start to download every time the power is turned on).
Update/download selection interface: When you need to upgrade the underlying firmware of U8W, you need to switch this toggle switch to the firmware upgrade tool. When you need to program the target chip through U8W, you need to turn the toggle switch to Burn the user program.
(Please refer to the figure above for the connection of the toggle switch)
Automatic burner/sorter interface: It is a control interface used to control the automatic burner/sorter for automatic production.

H3.3 U8W online download instructions

The target chip is installed on the U8W locking base and connected to the computer by U8W for online download
First use the USB cable provided by STC to connect the U8W to the computer, and then install the target MCU on the U8W in the direction shown in the figure below:



Then use STC-ISP to download the software to download the program, the steps are as follows:



MCU Type STC8H8K64U Pins Auto

COM Port USB-SERIAL CH340 (COM14)

Min Baud 2400 Max Baud 115200

Address

0x0000 ☑ Clear code buffer

0x0000 ☑ Clear EEPROM buffer

☑ Program OR

☑ Trim IRC

Input IRC frequency 11.0592 MHz

☑ Oscillator high gain

Set user EEPROM size 0.5 K

☑ Enable longer power-on-reset latency

☑ RESET pin used as I/O port

☑ Enable Low-Voltage reset

LVD detect leve 2.00 V

☐ Hardware enable WDT after power-on-reset

Watch-Dog-Timer prescal: 256

Delay 3 sec

Re-Count Infinite

☑ Auto reload the target file

☐ Reload and download when target file is modified

```
00000h  02 00 03 75 84 01 7E 44 20 00 E4 1B 44 7A 49 B0
00010h  78 F9 7E F8 02 FF 02 00 19 75 F2 00 75 F1 00 05
00020h  80 80 FC FF FF FF FF FF FF FF FF FF FF FF FF FF
00030h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00040h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00050h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00060h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00070h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00080h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00090h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000A0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000B0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000E0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000F0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

DataLen 037FA Checksum 37AB23

```
Checking target MCU ...
  Board Type: U8W
  Board version: 1.43
  Have 512K Bytes external Flash

Re-Checking target MCU ...
  MCU type: STC8H8K64U
  F/W version: 7.3.12U
```

D:\Demo\Demo.hex

☐ Beep PassTimes 10984

1 Select the MCU model;

2 Select the number of pins. When the chip is directly installed on the U8W to download, be sure to select the correct number of pins, otherwise the download will fail;

3 Select the serial port number corresponding to U8W;

4 Open the target file (HEX format or BIN format);

5 Set the hardware options;

6 Click the "Download/Program" button to start burning;

7The step information of the programming process is displayed, and the message "Completed!" is displayed when the programming is completed.

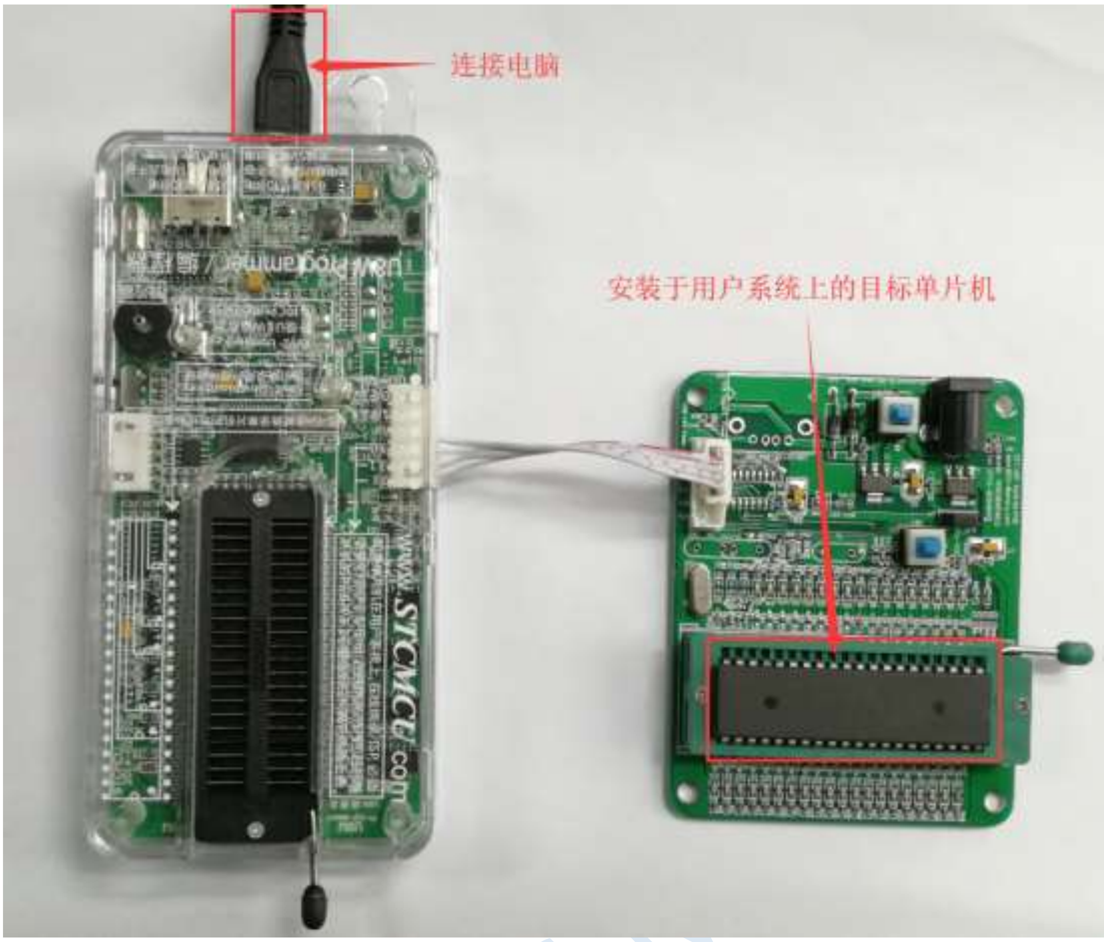
When there is the version number information of the output download board and the corresponding information of the external Flash in the information box, it means that the U8W download tool has been correctly detected.

During the downloading process, the 4 LEDs on the U8W download tool will be displayed in marquee mode. After the download is complete, if the download is successful, All 4 LEDs will be on and off at the same time; if the download fails, all 4 LEDs will be off.

It is recommended that users use the latest version of STC-ISP to download the software (please pay attention to the updates of the STC-ISP download software on the STC official website http://www.STCMCUDATA.com. It is strongly recommended that users download the software from the official website http://www.STCMCUDATA.com).

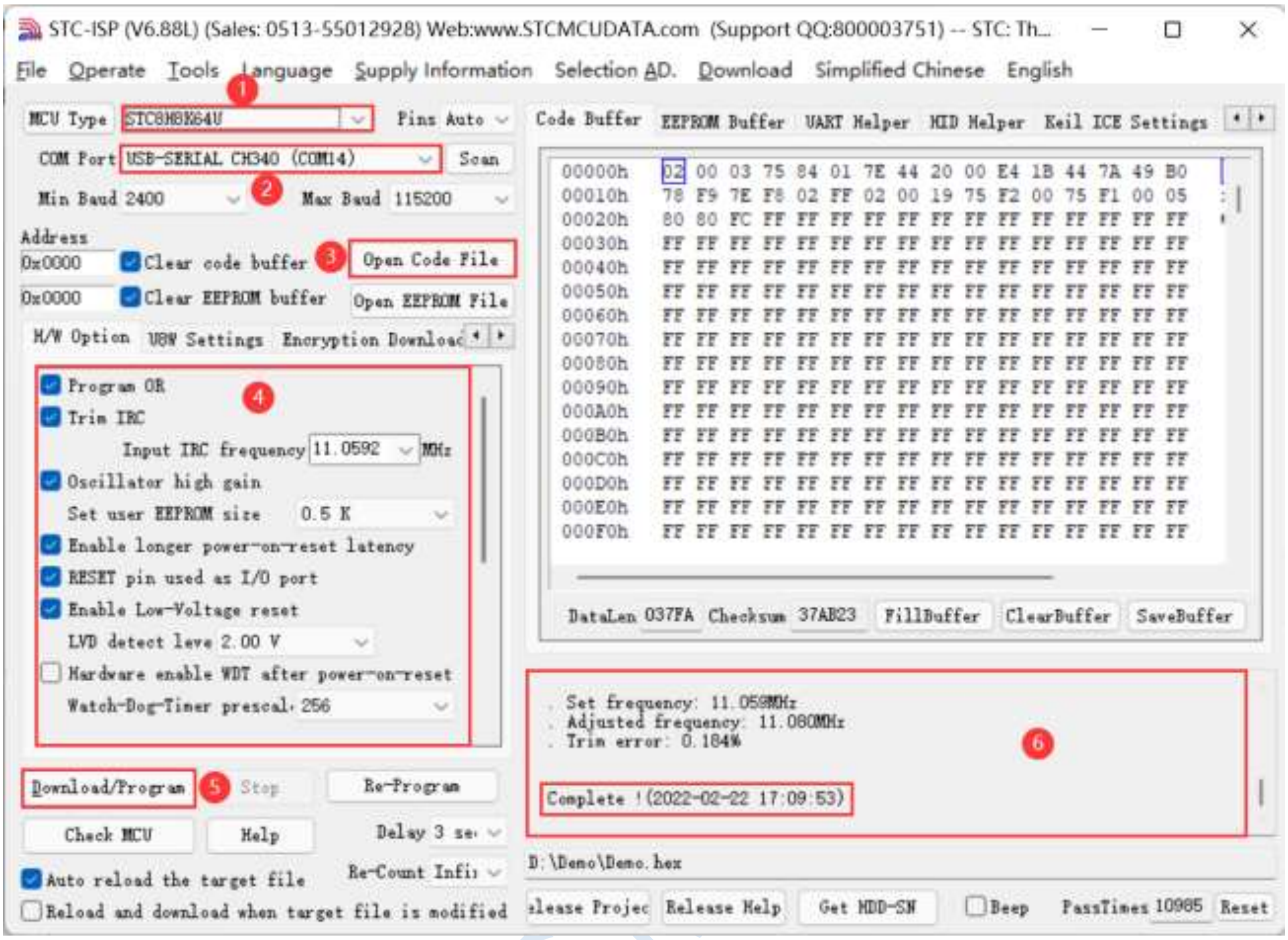
The target chip is connected to U8W through the user system leads and U8W is connected to the computer for online download

Firstly, use the USB cable provided by STC to connect U8W to the computer, and then connect U8W to the target MCU of the user system through the download line. The connection method is shown in the following figure:



Then use STC-ISP to download the software to download the program, the steps are as follows:

1. Select the MCU model;
2. Select the serial port number corresponding to U8W;
3. Open the target file (HEX format or BIN format);
4. Set hardware options;
5. Click the "Download/Program" button to start burning;
6. The step information of the programming process is displayed, and the message "Completed!" is displayed when the programming is completed.



MCU Type STC8H8K64U Pins Auto

COM Port USB-SERIAL CH340 (COM14)

Min Baud 2400 Max Baud 115200

Address

0x0000 ☑ Clear code buffer

0x0000 ☑ Clear EEPROM buffer

☑ Program OR

☑ Trim IRC

Input IRC frequency 11.0592 MHz

☑ Oscillator high gain

Set user EEPROM size 0.5 K

☑ Enable longer power-on-reset latency

☑ RESET pin used as I/O port

☑ Enable Low-Voltage reset

LVD detect leve 2.00 V

☐ Hardware enable WDT after power-on-reset

Watch-Dog-Timer prescal: 256

Delay 3 sec

☑ Auto reload the target file

☐ Reload and download when target file is modified

Re-Count Infinite

```
00000h  02 00 03 75 84 01 7E 44 20 00 E4 1B 44 7A 49 B0
00010h  78 F9 7E F8 02 FF 02 00 19 75 F2 00 75 F1 00 05
00020h  80 80 FC FF FF FF FF FF FF FF FF FF FF FF FF FF
00030h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00040h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00050h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00060h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00070h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00080h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00090h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000A0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000B0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000E0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000F0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

DataLen 037FA Checksum 37AB23

```
. Set frequency: 11.059MHz
. Adjusted frequency: 11.080MHz
. Trim error: 0.184%

Complete !(2022-02-22 17:09:53)
```

D:\Demo\Demo.hex

☐ Beep PassTimes 10985

When there is the version number information of the output download board and the corresponding information of the external Flash in the information box, it means that the U8W download tool has been correctly detected.

During the download process, the 4 LEDs on the U8W download tool will be displayed in marquee mode. After the download is complete, if the download is successful, the 4 LEDs will be on and off at the same time; if the download fails, all the 4 LEDs will be off.

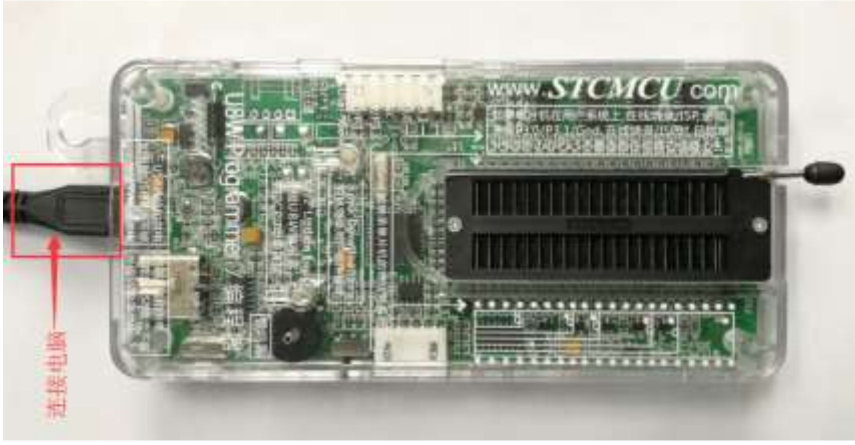
It is recommended that users use the latest version of STC-ISP to download the software (please always pay attention to the updates of the STC-ISP download software on the STC official website http://www.STCMCUDATA.com. It is strongly recommended that users download the software from the official website http://www.STCMCUDATA.com).

F.3.4 U8W offline download instructions

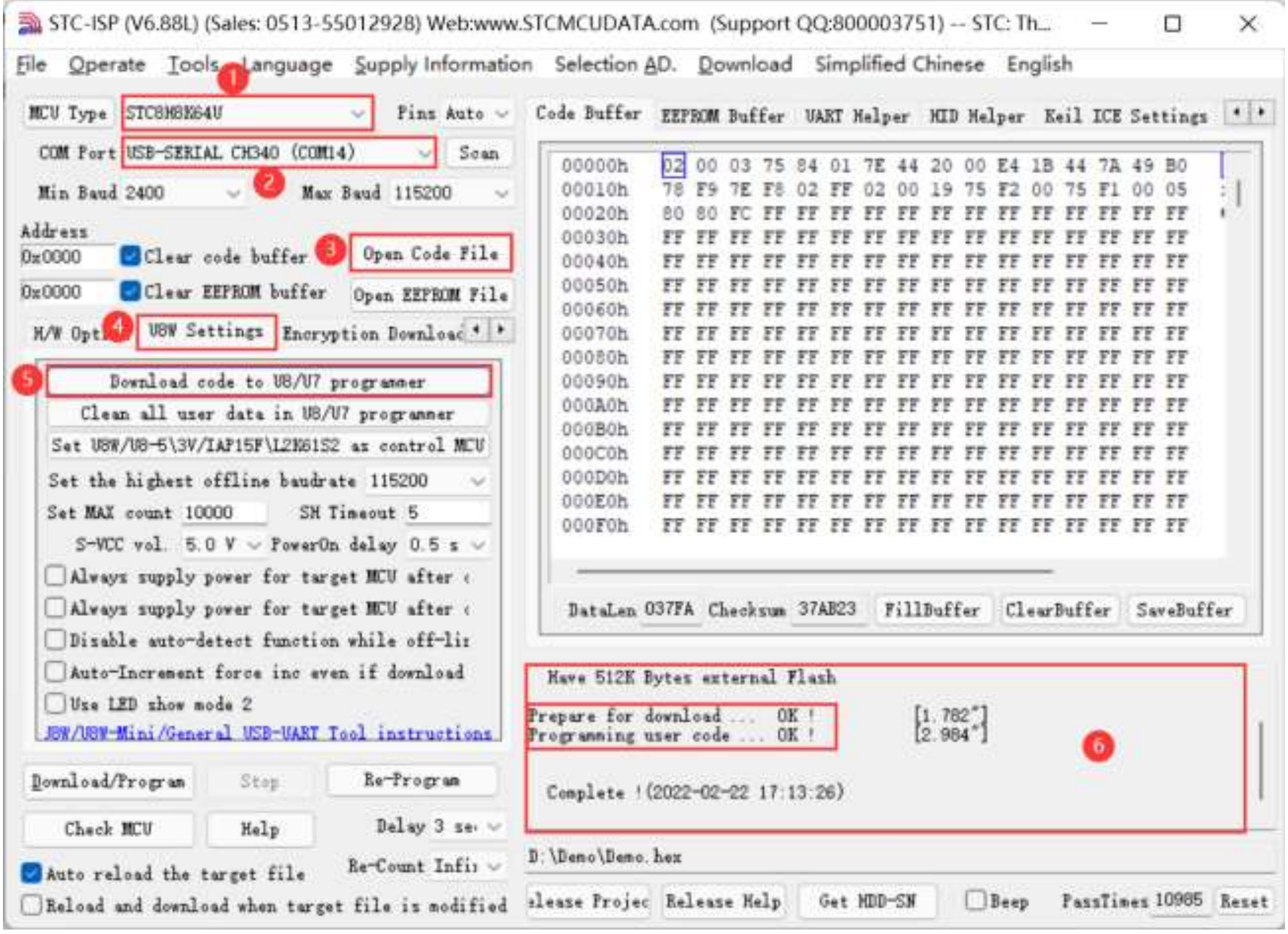
The target chip is installed on the U8W socket and locked and connected to the computer via USB to power the U8W for offline download

The steps to use USB to power U8W for offline download are as follows:

(1) Use the USB cable provided by STC to connect the U8W download board to the computer, as shown below:



(2) Set up in the STC-ISP download software according to the steps shown in the figure below:


STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...
File Operate Tools Language Supply Information Selection AD. Download Simplified Chinese English
Code Buffer EEPROM Buffer UART Helper HID Helper Keil ICE Settings
Open Code File
Open EEPROM File
H/W Opt
U8W Settings
Encryption Download
Download code to U8/U7 programmer
Clean all user data in U8/U7 programmer
Set U8W/U8-5\3V/IAP15F\L2K61S2 as control MCU
JBW/U8W-Mini/General USB-UART Tool instructions
Download/Program
Stop
Re-Program
Check MCU
Help
FillBuffer
ClearBuffer
SaveBuffer
Release Projec
Release Help
Get HDD-SN
Reset
1
2
3
4
5
6

MCU Type STC8H8K64U Pins Auto
COM Port USB-SERIAL CH340 (COM14)
Min Baud 2400 Max Baud 115200
Address
0x0000 ☑ Clear code buffer
0x0000 ☑ Clear EEPROM buffer
Set the highest offline baudrate 115200
Set MAX count 10000 SN Timeout 5
S-VCC vol. 5.0 V PowerOn delay 0.5 s
☐ Always supply power for target MCU after ‹
☐ Always supply power for target MCU after ‹
☐ Disable auto-detect function while off-li
☐ Auto-Increment force inc even if download
☐ Use LED show mode 2
Delay 3 se
Re-Count Infi
☑ Auto reload the target file
☐ Reload and download when target file is modified

```
00000h  02 00 03 75 84 01 7E 44 20 00 E4 1B 44 7A 49 B0
00010h  78 F9 7E F8 02 FF 02 00 19 75 F2 00 75 F1 00 05
00020h  80 80 FC FF FF FF FF FF FF FF FF FF FF FF FF FF
00030h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00040h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00050h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00060h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00070h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00080h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00090h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000A0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000B0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000E0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000F0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

DataLen 037FA Checksum 37AB23

```
Have 512K Bytes external Flash
Prepare for download ...   OK !        [1.782"]
Programming user code ...  OK !        [2.984"]

Complete !(2022-02-22 17:13:26)
```

D:\Demo\Demo.hex

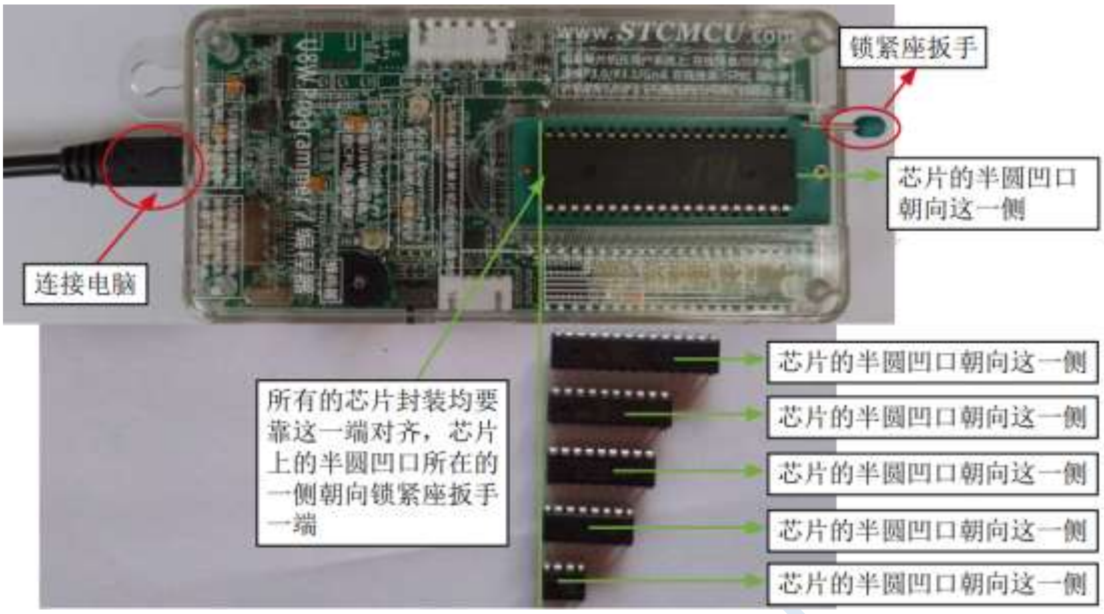
☐ Beep PassTimes 10985

1. Select the MCU model;
2. Select the serial port number corresponding to U8W;
3. Open the target file (HEX format or BIN format);
4. Set the hardware options;
5. Select the "U8W Offline/Online" tab, set the offline programming options, pay attention to the S-VCC output voltage matches the target chip working voltage; click the "Download user program to U8/U7 programmer for offline download" button；
6. The step information of the setting process is displayed, and the prompt "Completed!" is displayed after the setting is completed.

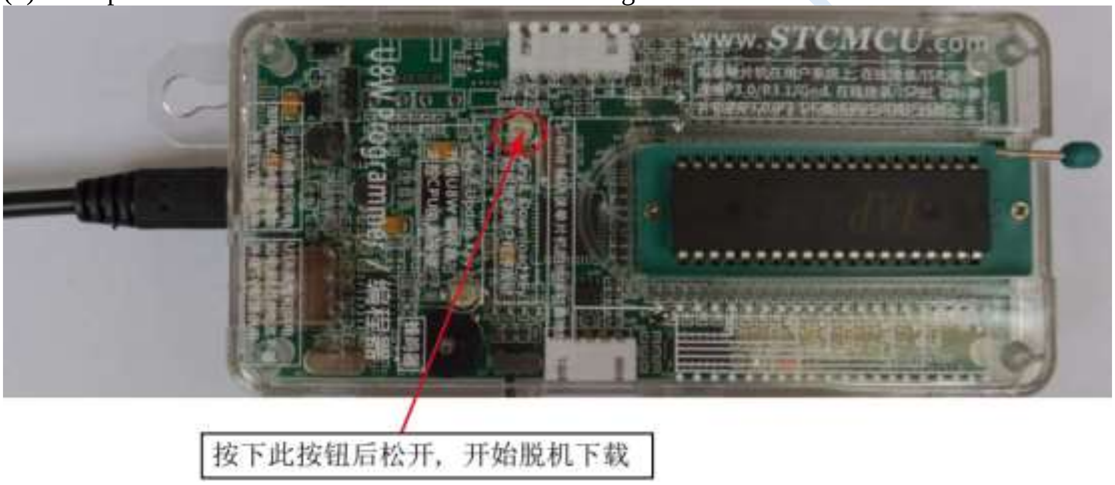
According to the steps in the above figure, after the operation is completed, if the download is successful, it means that the user code and related setting options have been downloaded to the U8W download tool.

It is recommended that users use the latest version of STC-ISP to download the software (please always pay attention to the updates of the STC-ISP download software on the STC official website http://www.STCMCUDATA.com. It is

strongly recommended that users download the software from the official website http://www.STCMCUDATA.com).
(3) Place the target MCU in the U8W download tool in the direction shown in the figure below, as shown in the figure below:



(4) Then press and release the button as shown in the figure below to start offline download:

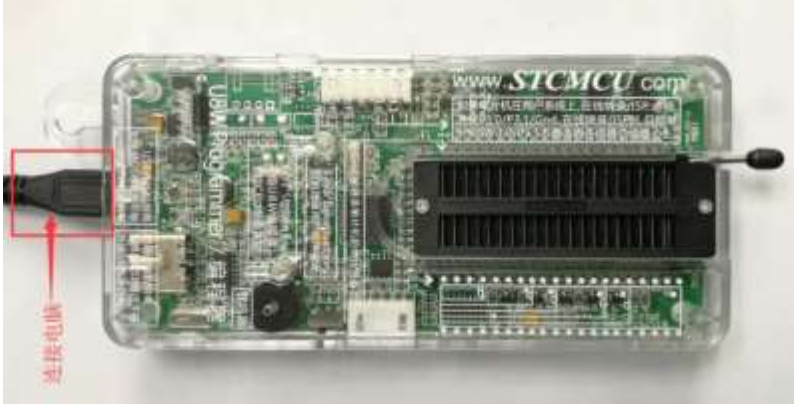


During the downloading process, the 4 LEDs on the U8W download tool will be displayed in marquee mode. After the download is complete, if the download is successful, the 4 LEDs will be on and off at the same time; if the download fails, all the 4 LEDs will be off.

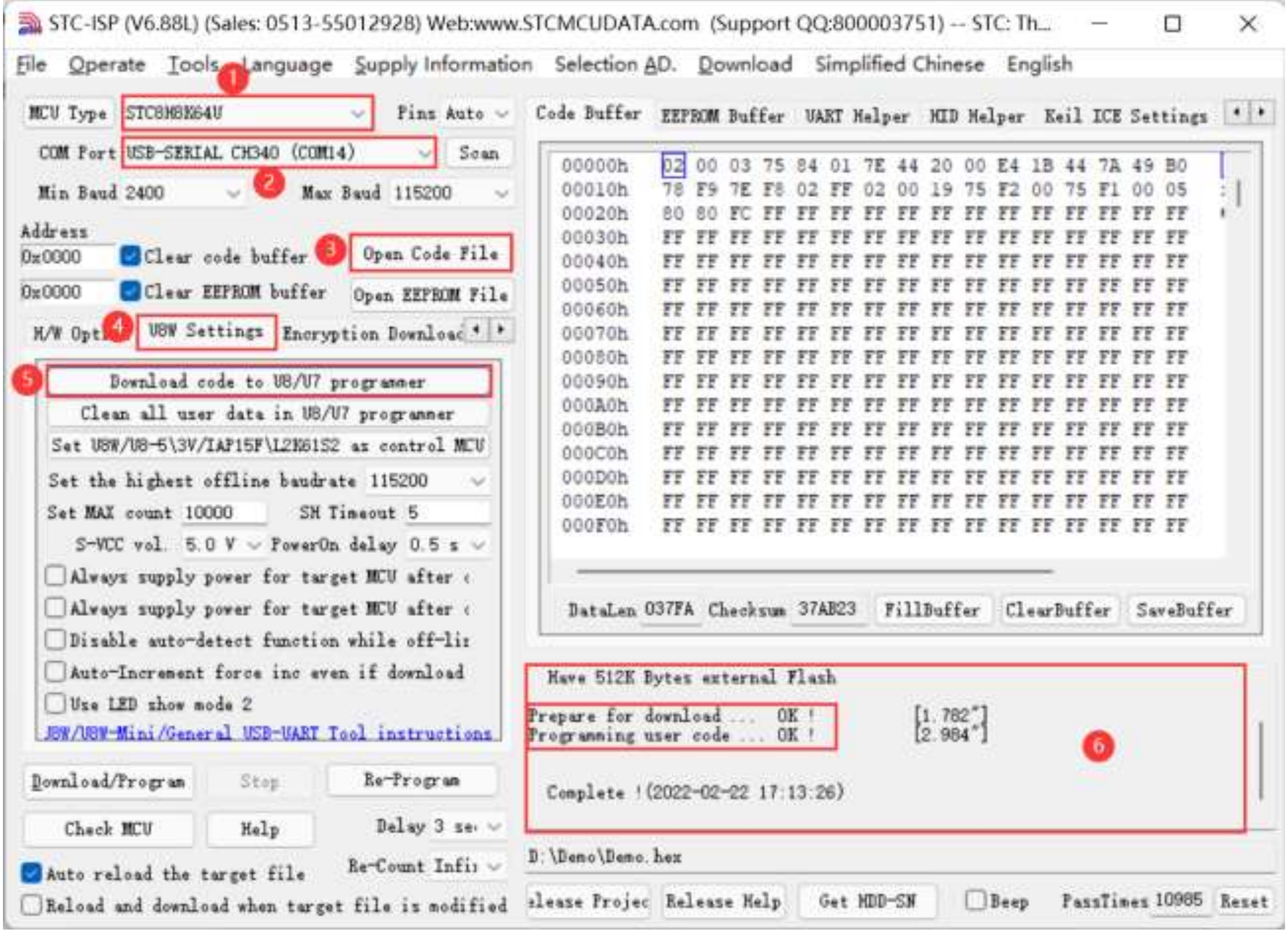
Offline download plug and play burning function introduction:
1. After completing the above steps (1) and (2), U8W is in the plug-and-play programming state by default when it is connected to the computer and powered on;
2. Put the chip into the programming socket according to the instructions in step (3). While tightening the socket wrench, U8W will automatically start programming;
3. Display the burning process and burning result through the indicator light;
4. After programming is completed, loosen the wrench and take out the chip;
5. Repeat steps 2, 3 and 4 for continuous programming, eliminating the need to press the button to trigger the programming action.

The target chip is connected to U8W by the user system lead and connected to the computer via USB to supply power to U8W for offline download.

The steps for offline download using USB to supply power to U8W are as follows:
(1) Use the USB cable provided by STC to connect the U8W download board to the computer, as shown below:



(2) Set up in the STC-ISP download software according to the steps shown in the figure below:
It is recommended that users use the latest version of STC-ISP to download the software (please always pay attention to the updates of the STC-ISP download software on the STC official website http://www.STCMCUDATA.com. It is strongly recommended that users download the software from the official website http://www.STCMCUDATA.com).



STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...

MCU Type STC8H8K64U Pins Auto
COM Port USB-SERIAL CH340 (COM14)
Min Baud 2400 Max Baud 115200
Address
0x0000 ☑ Clear code buffer
0x0000 ☑ Clear EEPROM buffer
H/W Opt
Set the highest offline baudrate 115200
Set MAX count 10000 SN Timeout 5
S-VCC vol. 5.0 V PowerOn delay 0.5 s
☐ Always supply power for target MCU after (
☐ Always supply power for target MCU after (
☐ Disable auto-detect function while off-lir
☐ Auto-Increment force inc even if download
☐ Use LED show mode 2
Delay 3 se
☑ Auto reload the target file
Re-Count Infi
☐ Reload and download when target file is modified

```
00000h  02 00 03 75 84 01 7E 44 20 00 E4 1B 44 7A 49 B0
00010h  78 F9 7E F8 02 FF 02 00 19 75 F2 00 75 F1 00 05
00020h  80 80 FC FF FF FF FF FF FF FF FF FF FF FF FF FF
00030h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00040h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00050h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00060h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00070h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00080h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00090h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000A0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000B0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000E0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000F0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

DataLen 037FA Checksum 37AB23

```
Have 512K Bytes external Flash
Prepare for download ... OK !        [1.782"]
Programming user code ... OK !       [2.984"]

Complete !(2022-02-22 17:13:26)
```

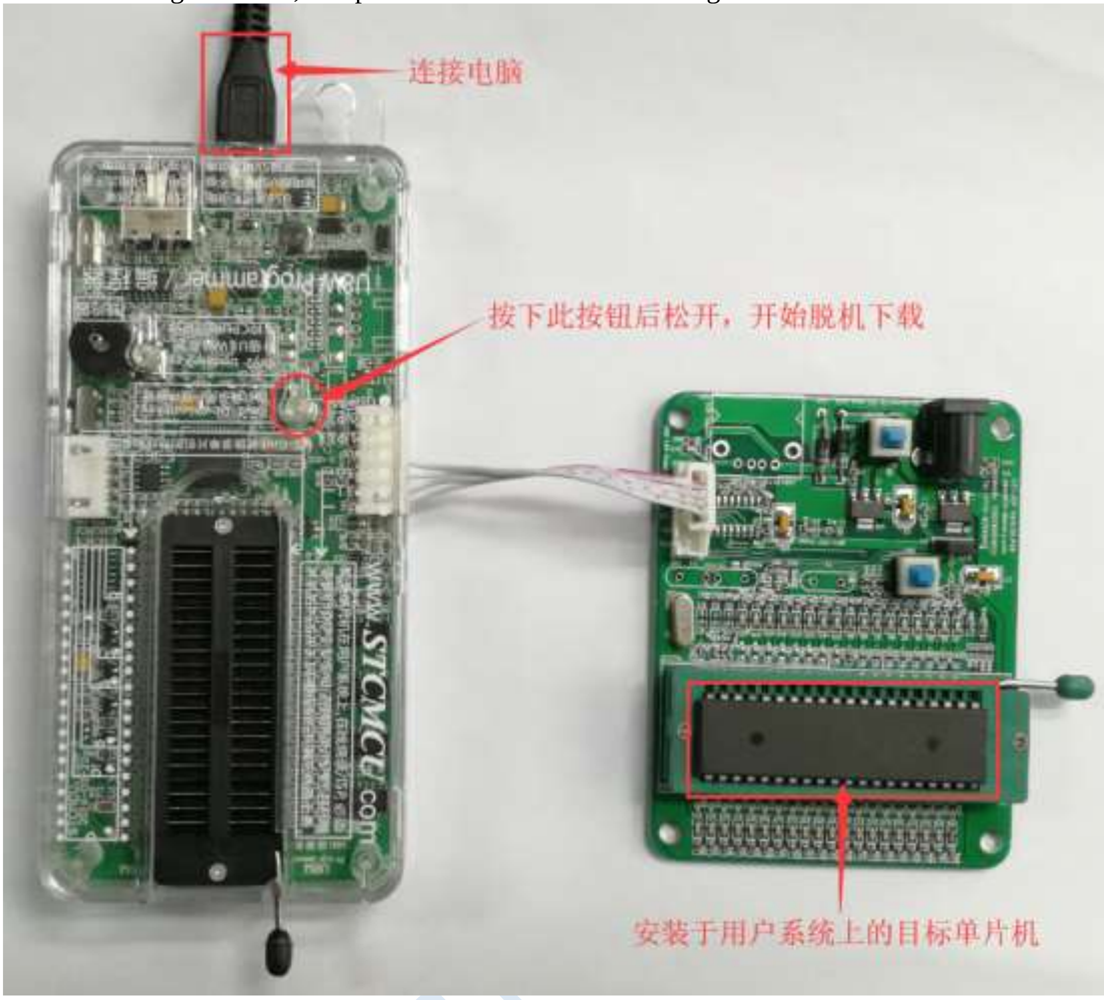
D:\Demo\Demo.hex

☐ Beep PassTimes 10985

1. Select the MCU model;
2. Select the serial port number corresponding to U8W;
3. Open the target file (HEX format or BIN format);
4. Set the hardware options;
5. Select the "U8W Offline/Online" tab, set the offline programming options, pay attention to the S-VCC output voltage matches the target chip operating voltage;
Click the "Download user program to U8/U7 programmer for offline download" button;
6. The step information of the setting process is displayed, and the prompt "Completed!" is displayed after the setting is completed.

According to the steps in the above figure, after the operation is completed, if the download is successful, it means that the user code and related setting options have been downloaded to the U8W download tool.

(3) Then use the cable to connect the computer, connect the U8W download tool and the user system (target MCU) as shown in the figure below, and press the button shown in the figure and release it to start offline downloading:



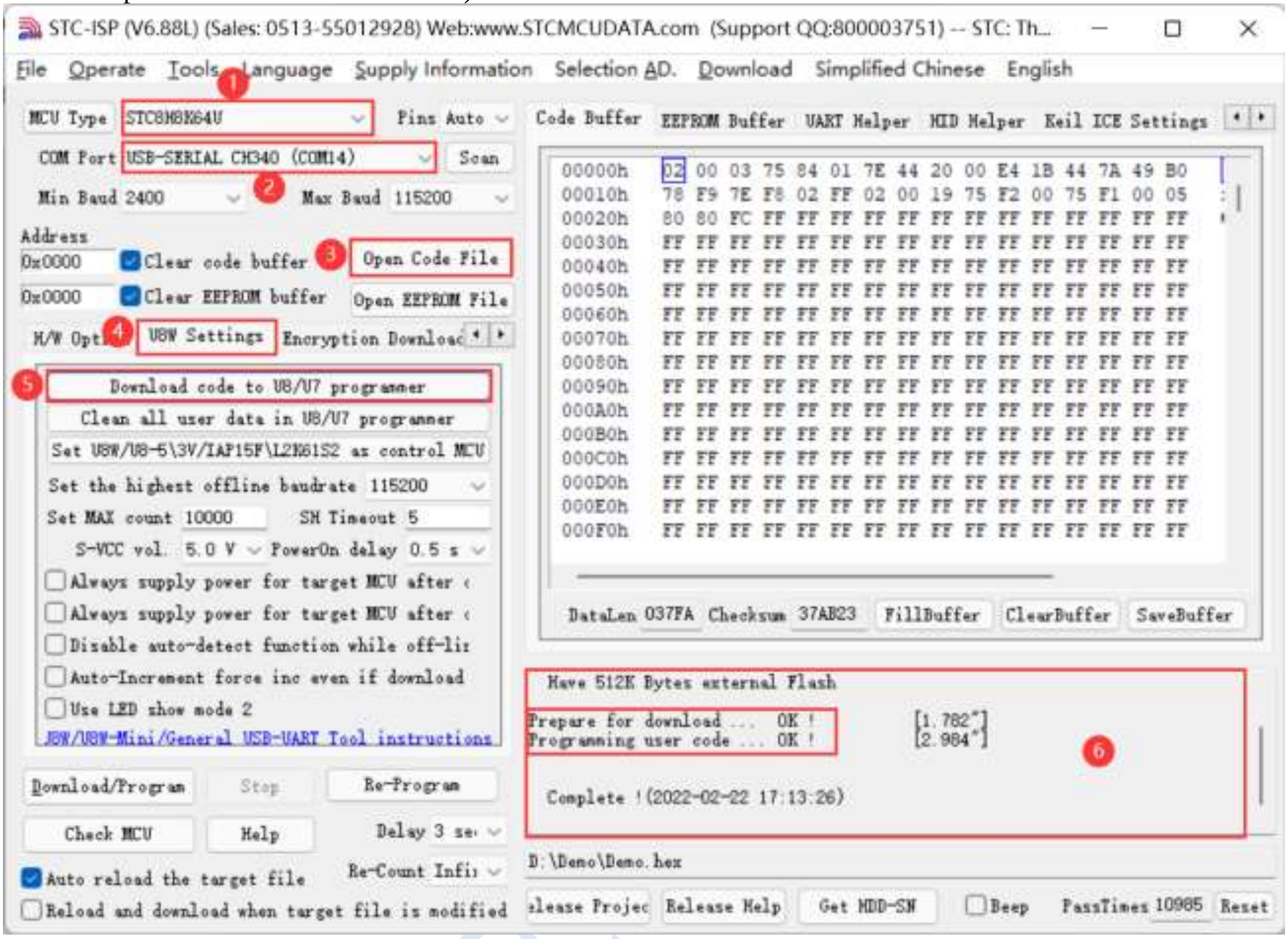
During the downloading process, the 4 LEDs on the U8W download tool will be displayed in marquee mode. After the download is complete, if the download is successful, the 4 LEDs will be on and off at the same time; if the download fails, all the 4 LEDs will be off.

The target chip is connected to U8W by the user system lead, and the user system supplies power to U8W for offline download

(1) Firstly, use the USB cable provided by STC to connect the U8W download board to the computer, as shown below:

(2) Set up in the STC-ISP download software according to the steps shown in the figure below:

It is recommended that users use the latest version of STC-ISP to download the software (please always pay attention to the updates of the STC-ISP download software on the STC official website http://www.STCMCUDATA.com. It is strongly recommended that users download the software from the official website http://www.STCMCUDATA.com).



MCU Type STC8H8K64U Pins Auto

COM Port USB-SERIAL CH340 (COM14)

Min Baud 2400 Max Baud 115200

Address

0x0000 ☑ Clear code buffer

0x0000 ☑ Clear EEPROM buffer

Set U8W/U8-5\3V/IAP15F\L2K61S2 as control MCU

Set the highest offline baudrate 115200

Set MAX count 10000 SN Timeout 5

S-VCC vol. 5.0 V PowerOn delay 0.5 s

☐ Always supply power for target MCU after (

☐ Always supply power for target MCU after (

☐ Disable auto-detect function while off-lir

☐ Auto-Increment force inc even if download

☐ Use LED show mode 2

U8W/U8W-Mini/General USB-UART Tool instructions

Delay 3 sec

☑ Auto reload the target file Re-Count Infi

☐ Reload and download when target file is modified

```
00000h  02 00 03 75 84 01 7E 44 20 00 E4 1B 44 7A 49 B0
00010h  78 F9 7E F8 02 FF 02 00 19 75 F2 00 75 F1 00 05
00020h  80 80 FC FF FF FF FF FF FF FF FF FF FF FF FF FF
00030h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00040h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00050h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00060h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00070h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00080h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00090h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000A0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000B0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000E0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000F0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

DataLen 037FA Checksum 37AB23

```
Have 512K Bytes external Flash
Prepare for download ...  OK !      [1.782"]
Programming user code ... OK !      [2.984"]

Complete !(2022-02-22 17:13:26)
```

D:\Demo\Demo.hex

☐ Beep PassTimes 10985

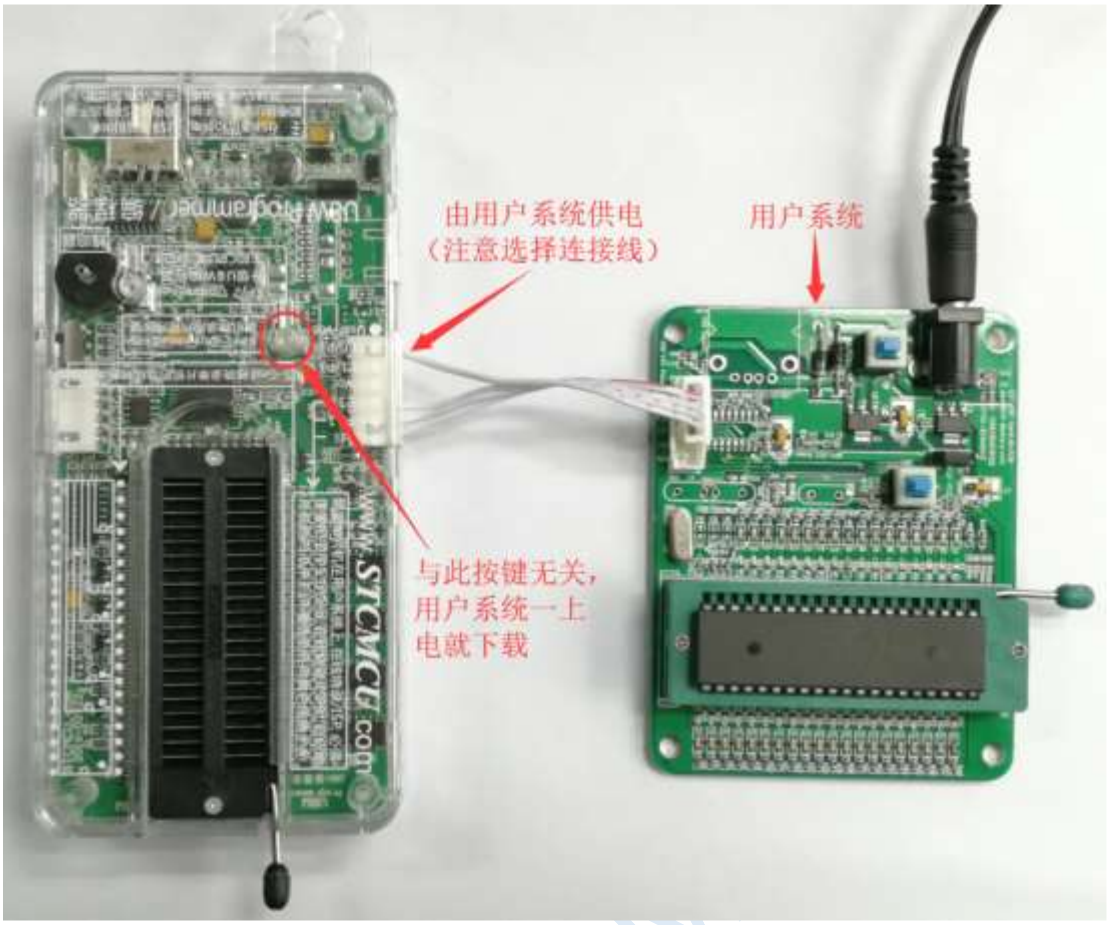
1. Select the MCU model;
2. Select the serial port number corresponding to U8W;
3. Open the target file (HEX format or BIN format);
4. Set the hardware options;
5. Select the "U8W Offline/Online" tab, set the offline programming options, pay attention to the S-VCC output voltage matches the target chip operating voltage;

Click the "Download user program to U8/U7 programmer for offline download" button;

6. The step information of the setting process is displayed, and the prompt "Completed!" is displayed after the setting is completed.

According to the steps in the above figure, after the operation is completed, if the download is successful, it means that the user code and related setting options have been downloaded to the U8W download tool.

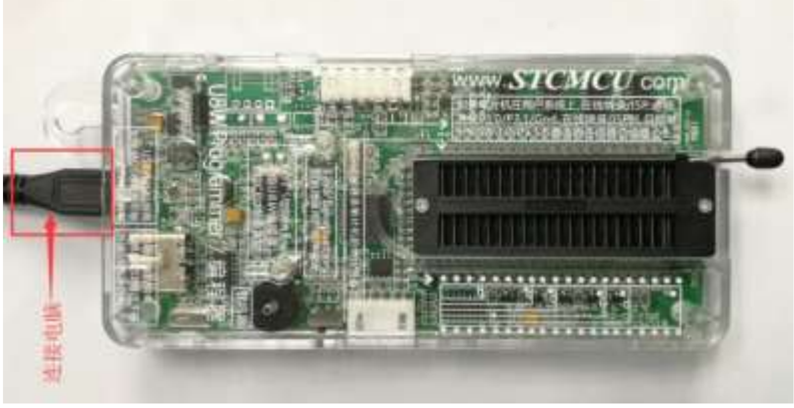
(3) Then connect U8W to the user system as shown in the figure below, supply power to the user system, and then start offline downloading:



During the downloading process, the 4 LEDs on the U8W download tool will be displayed in marquee mode. After the download is complete, if the download is successful, the 4 LEDs will be on and off at the same time; if the download fails, all the 4 LEDs will be off.

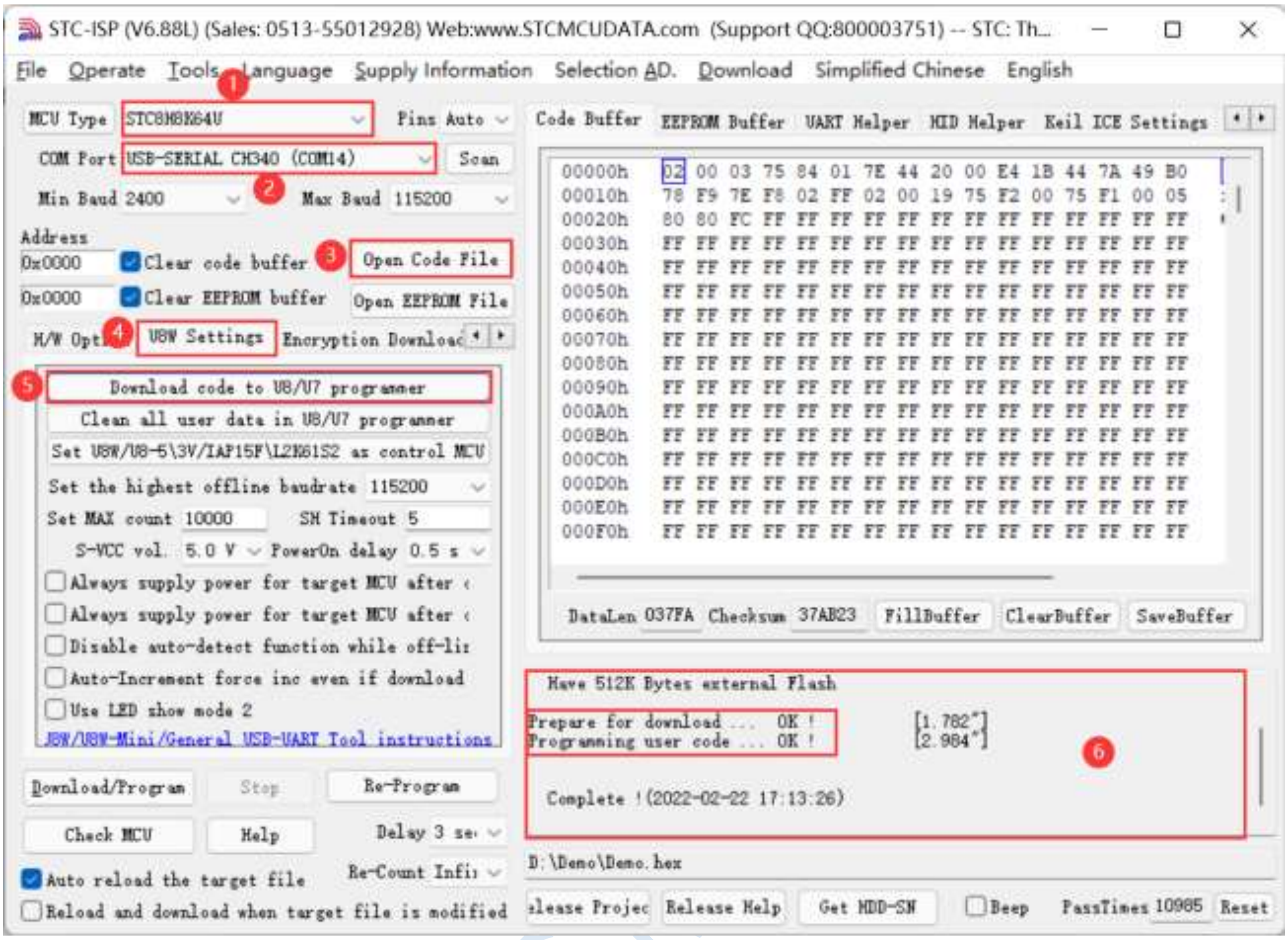
The target chip is connected to U8W by the user system lead, and U8W and the user system are independently powered for offline download

(1) Firstly, use the USB cable provided by STC to connect the U8W download board to the computer, as shown below:



(2) Set up in the STC-ISP download software according to the steps shown in the figure below:

It is recommended that users use the latest version of STC-ISP to download the software (please always pay attention to the updates of the STC-ISP download software on the STC official website http://www.STCMCUDATA.com. It is strongly recommended that users download the software from the official website http://www.STCMCUDATA.com).



MCU Type STC8H8K64U Pins Auto

COM Port USB-SERIAL CH340 (COM14)

Min Baud 2400 Max Baud 115200

Address
0x0000 ☑ Clear code buffer
0x0000 ☑ Clear EEPROM buffer

Set the highest offline baudrate 115200

Set MAX count 10000 SN Timeout 5

S-VCC vol. 5.0 V PowerOn delay 0.5 s

□ Always supply power for target MCU after (

□ Always supply power for target MCU after (

□ Disable auto-detect function while off-lin

□ Auto-Increment force inc even if download

□ Use LED show mode 2

Delay 3 sec

☑ Auto reload the target file Re-Count Infi

□ Reload and download when target file is modified

```
00000h  02 00 03 75 84 01 7E 44 20 00 E4 1B 44 7A 49 B0
00010h  78 F9 7E F8 02 FF 02 00 19 75 F2 00 75 F1 00 05
00020h  80 80 FC FF FF FF FF FF FF FF FF FF FF FF FF FF
00030h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00040h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00050h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00060h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00070h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00080h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00090h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000A0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000B0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000E0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000F0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

DataLen 037FA Checksum 37AB23

```
Have 512K Bytes external Flash
Prepare for download ...  OK !     [1.782"]
Programming user code ... OK !     [2.984"]

Complete !(2022-02-22 17:13:26)
```

D:\Demo\Demo.hex

□ Beep PassTimes 10985

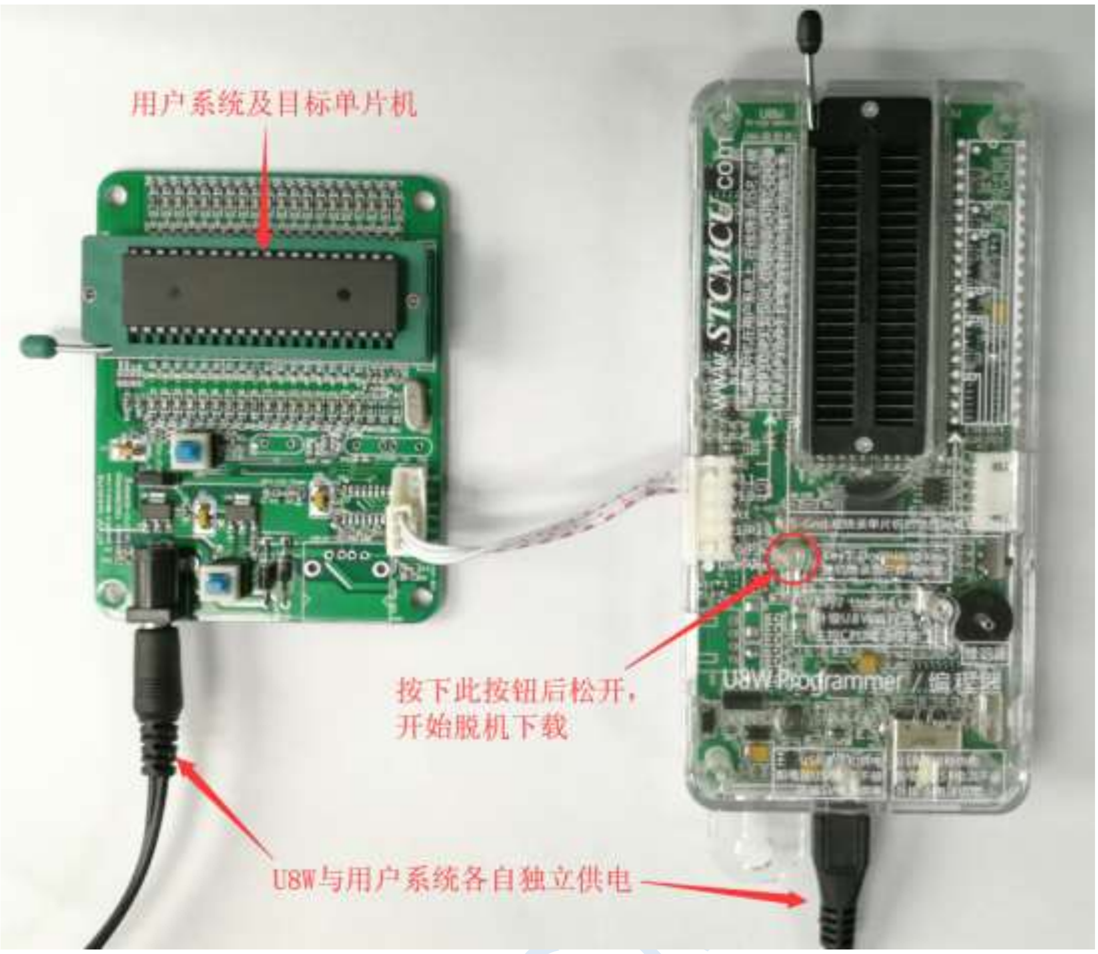
1. Select the MCU model;
2. Select the serial port number corresponding to U8W;
3. Open the target file (HEX format or BIN format);
4. Set the hardware options;
5. Select the "U8W Offline/Online" tab, set the offline programming options, pay attention to the S-VCC output voltage matches the target chip operating voltage;

Click the "Download user program to U8/U7 programmer for offline download" button;

6. The step information of the setting process is displayed, and the prompt "Completed!" is displayed after the setting is completed.

According to the steps in the above figure, after the operation is completed, if the download is successful, it means that the user code and related setting options have been downloaded to the U8W download tool.

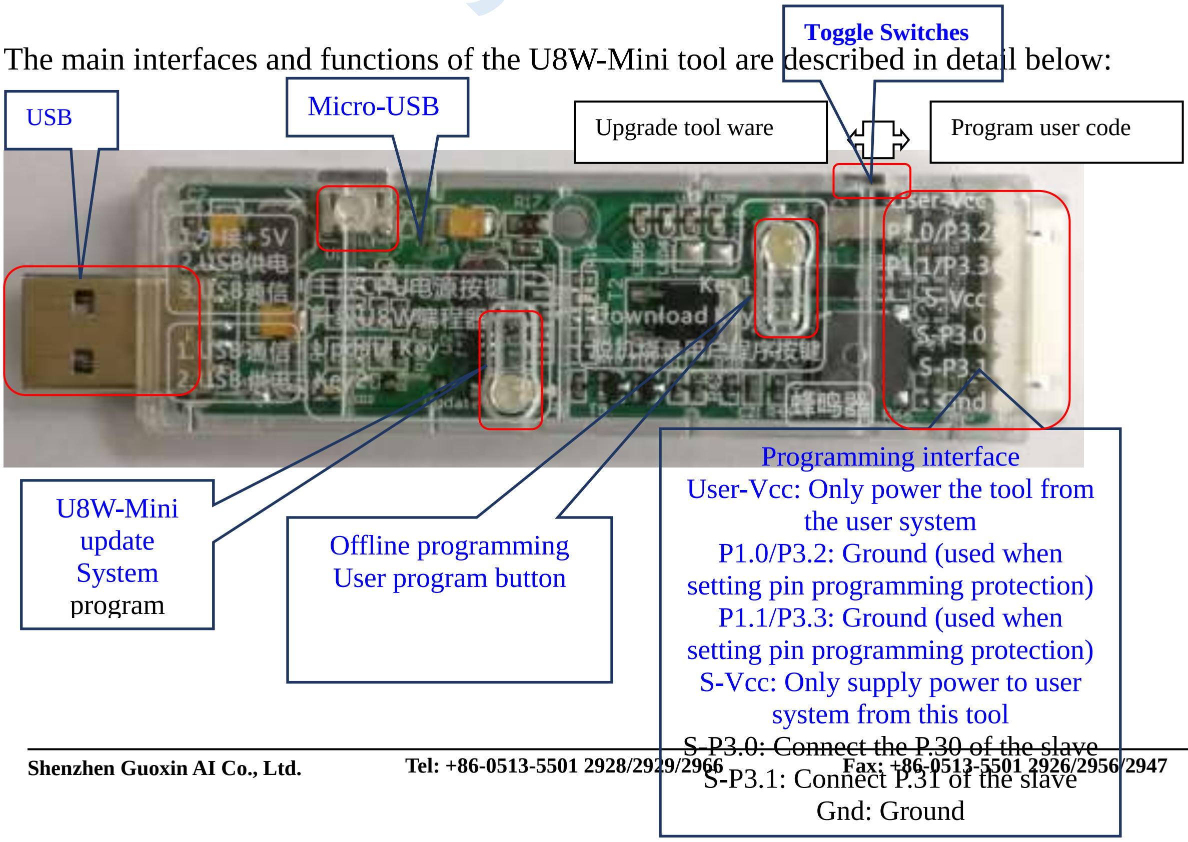
(3) Then connect U8W to the user system as shown in the figure below, and press the button shown in the figure first and then release it, ready to start offline download, and finally power on/on the user system to download the user program Officially begin:



During the downloading process, the 4 LEDs on the U8W download tool will be displayed in marquee mode. After the download is complete, if the download is successful, the 4 LEDs will be on and off at the same time; if the download fails, all the 4 LEDs will be off.

H.3.5 U8W-Mini's function introduction

The main interfaces and functions of the U8W-Mini tool are described in detail below:


Toggle Switches
USB
Micro-USB
Upgrade tool ware
Program user code
U8W-Mini update System program
Offline programming User program button

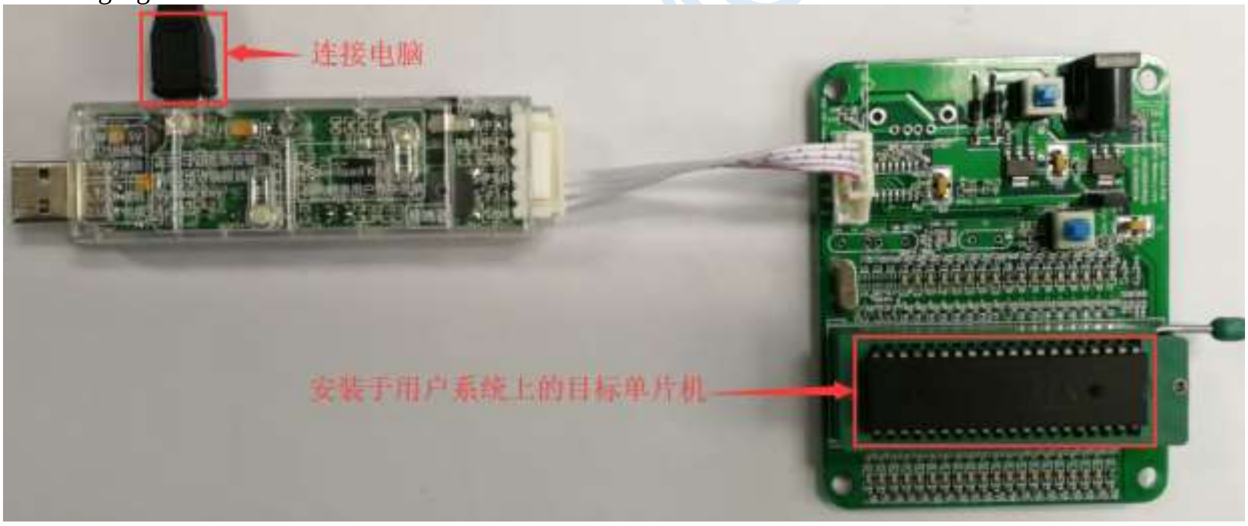
Programming interface
User-Vcc: Only power the tool from the user system
P1.0/P3.2: Ground (used when setting pin programming protection)
P1.1/P3.3: Ground (used when setting pin programming protection)
S-Vcc: Only supply power to user system from this tool
S-P3.0: Connect the P.30 of the slave
S-P3.1: Connect P.31 of the slave
Gnd: Ground

Programming interface: According to different power supply methods, use different download cables to connect the U8W-Mini download board and the user system.
U8W-Mini update system program button: used to update U8W-Mini tools. When there is a new version of U8W firmware, you need to press this button to update the U8W-Mini main control chip (note: you must first select update/download The toggle switch on the interface is moved to the upgrade tool firmware).
Offline download user program button: Start offline download button. First, the PC downloads the offline code to the U8W-Mini, and then uses the download cable to connect the user system to the U8W-Mini, and then press this button to start the offline download (the user will also start downloading immediately every time the power is turned on Code).
Update/download selection interface: When you need to upgrade the underlying firmware of U8W-Mini, you need to move this toggle switch to the firmware of the upgrade tool. When you need to program the target chip through U8W-Mini, you need to Toggle the switch to burn the user program. (Please refer to the figure above for the connection of the toggle switch)
USB interface: The USB interface has the same function as the Micro-USB interface. Users can connect one of them to the computer as needed.

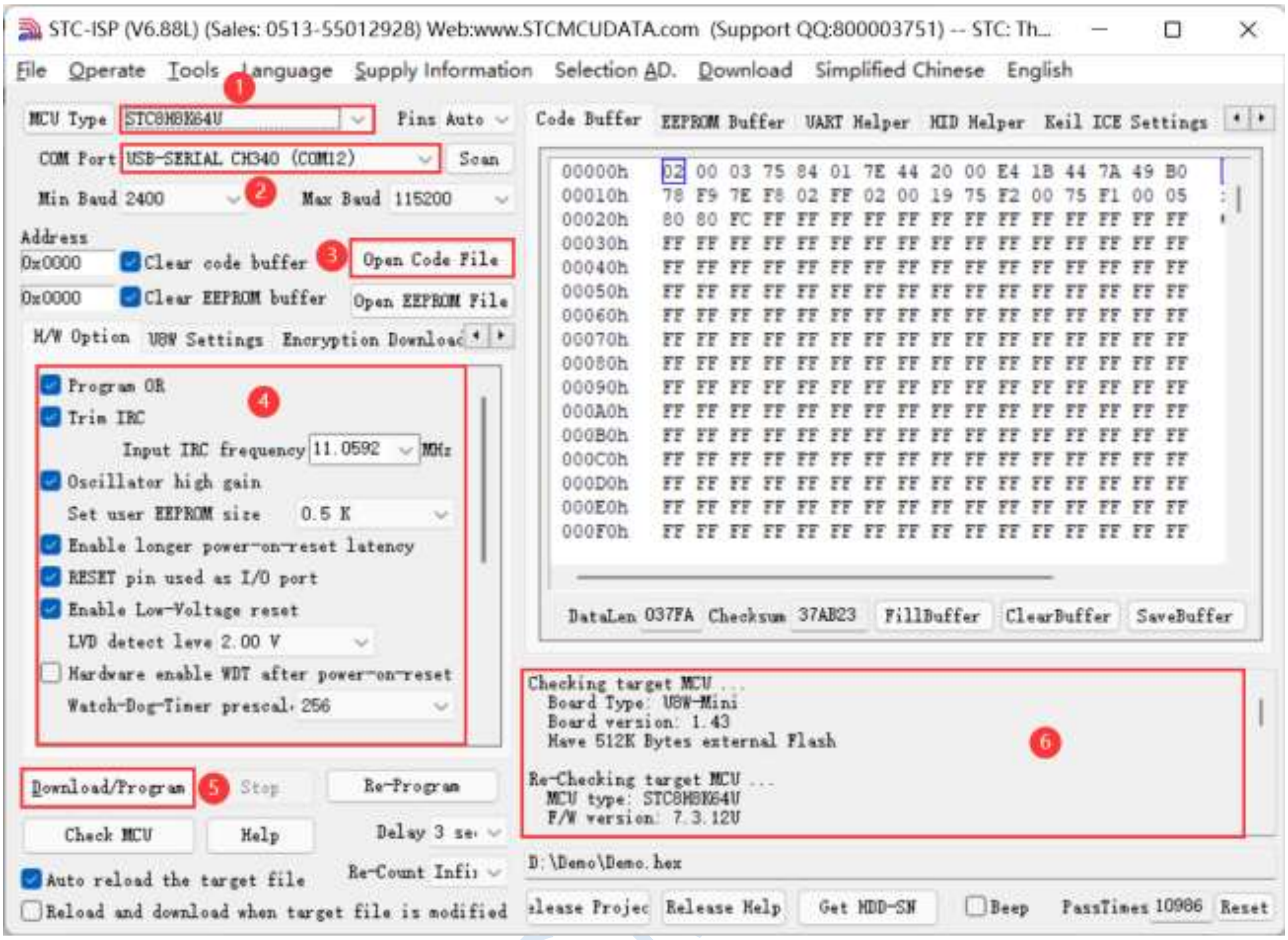
H.3.6 U8W-Mini online download instructions

The target chip is connected to the U8W-Mini through the user system lead, and the U8W-Mini is connected to the computer for online download

Firstly, use the USB cable provided by STC to connect the U8W-Mini to the computer, and then connect the U8W-Mini to the target MCU of the user system through the download cable. The connection method is shown in the following figure:



Then use STC-ISP to download the software to download the program, the steps are as follows:



MCU Type STC8H8K64U
Pins Auto
COM Port USB-SERIAL CH340 (COM12)
Min Baud 2400
Max Baud 115200
Address
0x0000 ☑ Clear code buffer
0x0000 ☑ Clear EEPROM buffer

☑ Program OR
☑ Trim IRC
Input IRC frequency 11.0592 MHz
☑ Oscillator high gain
Set user EEPROM size 0.5 K
☑ Enable longer power-on-reset latency
☑ RESET pin used as I/O port
☑ Enable Low-Voltage reset
LVD detect leve 2.00 V
☐ Hardware enable WDT after power-on-reset
Watch-Dog-Timer prescal 256

Delay 3 se
Re-Count Infi
☑ Auto reload the target file
☐ Reload and download when target file is modified

```
00000h   02 00 03 75 84 01 7E 44 20 00 E4 1B 44 7A 49 B0
00010h   78 F9 7E F8 02 FF 02 00 19 75 F2 00 75 F1 00 05
00020h   80 80 FC FF FF FF FF FF FF FF FF FF FF FF FF FF
00030h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00040h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00050h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00060h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00070h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00080h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00090h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000A0h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000B0h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C0h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D0h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000E0h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000F0h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

DataLen 037FA Checksum 37AB23

```
Checking target MCU ...
  Board Type:  U8W-Mini
  Board version:  1.43
  Have 512K Bytes external Flash

Re-Checking target MCU ...
  MCU type:  STC8H8K64U
  F/W version:  7.3.12U
```

D:\Demo\Demo.hex

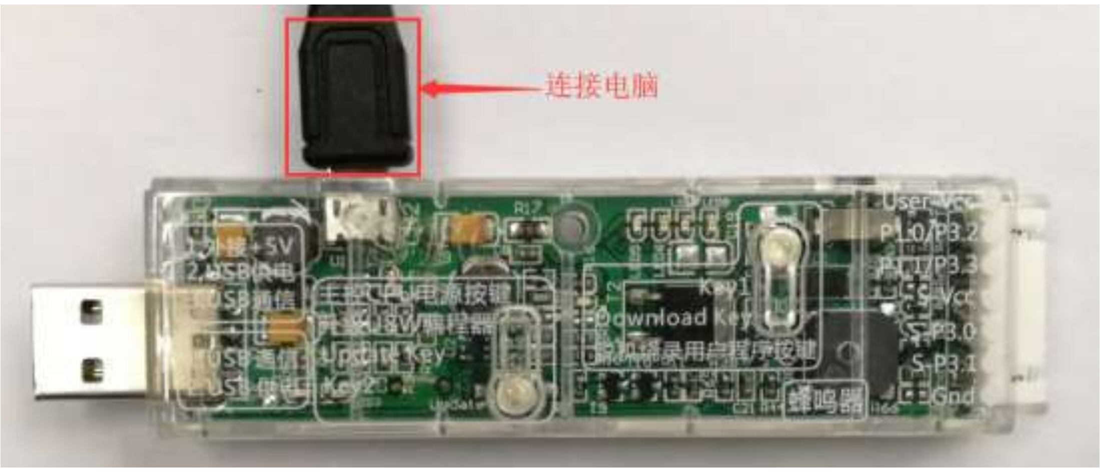
☐ Beep PassTimes 10986

1. Select the MCU model;
2. Select the number of pins. When the chip is directly installed on the U8W-Mini to download, be sure to select the correct number of pins, otherwise the download will fail;
3. Select the serial port number corresponding to U8W-Mini;
4. Open the target file (HEX format or BIN format);
5. Set the hardware options;
6. Click the "Download/Program" button to start burning;
7. The step information of the burning process is displayed, and the message "Completed!" is displayed when the burning is completed.

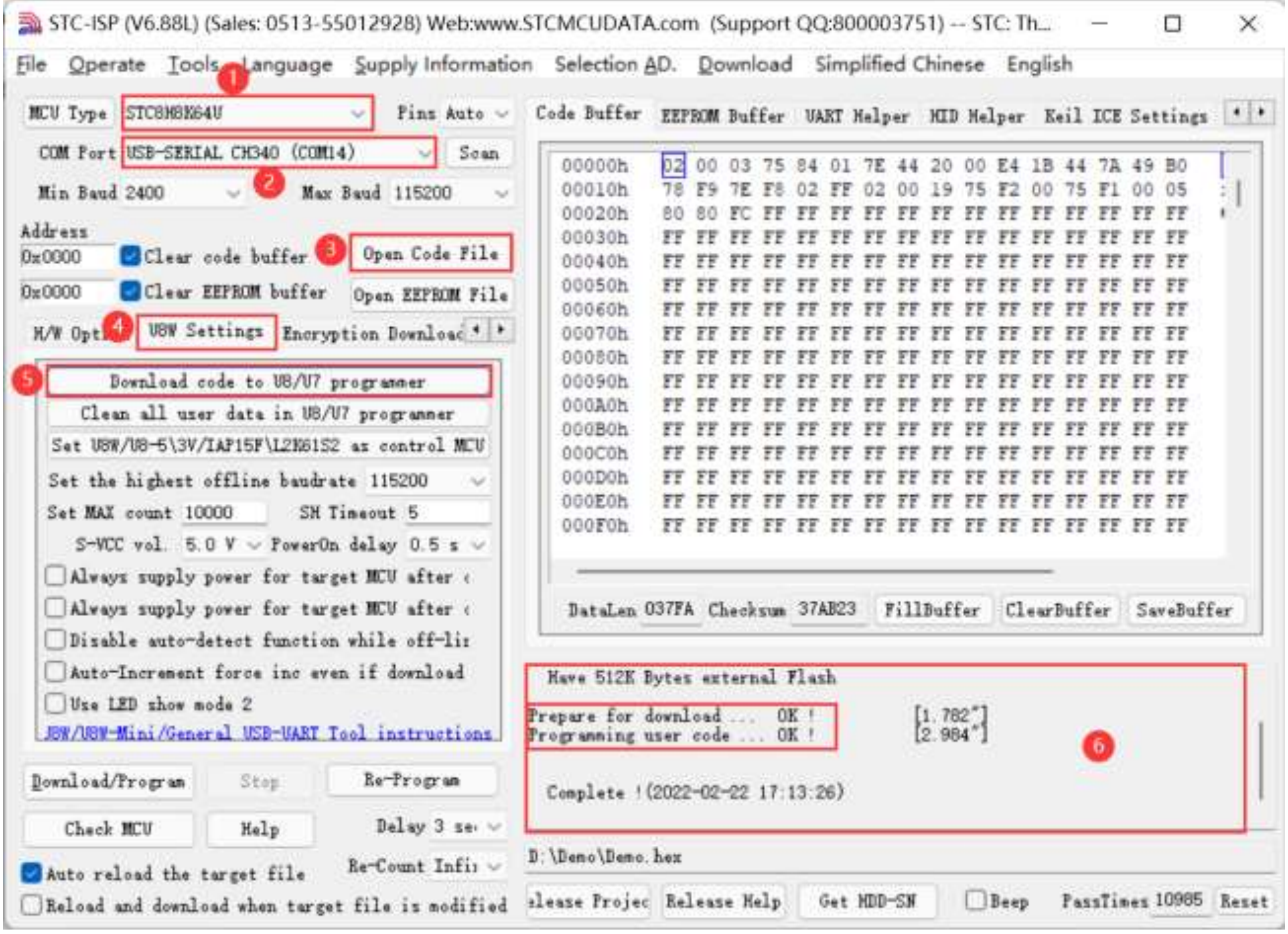
When there is the version number information of the output download board and the corresponding information of the external Flash in the information box, it means that the U8W-Mini download tool has been correctly detected.
During the downloading process, the 4 LEDs on the U8W-Mini download tool will be displayed in marquee mode. After the download is complete, if the download is successful, the 4 LEDs will be on and off at the same time; if the download fails, all the 4 LEDs will be off.
It is recommended that users use the latest version of STC-ISP to download the software (please pay attention to the updates of the STC-ISP download software on the STC official website http://www.STCMCUDATA.com. It is strongly recommended that users download the software on the official website http://www.STCMCUDATA.com).

H.3.7 U8W-Mini offline download instructions

The target chip is connected to the U8W-Mini by the user system lead and connected to the computer via USB to power the U8W-Mini for offline download.
The steps for offline download using USB to power the U8W-Mini are as follows:
(1) Use the USB cable provided by STC to connect the U8W-Mini download board to the computer, as shown below:



(2) Set up in the STC-ISP download software according to the steps shown in the figure below:


STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...
File Operate Tools Language Supply Information Selection AD. Download Simplified Chinese English
Code Buffer EEPROM Buffer UART Helper HID Helper Keil ICE Settings
1
2
3
4
5
6

MCU Type STC8H8K64U Pins Auto

COM Port USB-SERIAL CH340 (COM14) Scan

Min Baud 2400 Max Baud 115200

Address

0x0000 ☑Clear code buffer Open Code File

0x0000 ☑Clear EEPROM buffer Open EEPROM File

H/W Opti U8W Settings Encryption Download

Download code to U8/U7 programmer

Clean all user data in U8/U7 programmer

Set U8W/U8-5\3V/IAP15F\L2K61S2 as control MCU

Set the highest offline baudrate 115200

Set MAX count 10000 SN Timeout 5

S-VCC vol. 5.0 V PowerOn delay 0.5 s

□Always supply power for target MCU after (

□Always supply power for target MCU after (

□Disable auto-detect function while off-lin

□Auto-Increment force inc even if download

□Use LED show mode 2

U8W/U8W-Mini/General USB-UART Tool instructions

Download/Program Stop Re-Program

Check MCU Help Delay 3 se

☑Auto reload the target file Re-Count Infi

□Reload and download when target file is modified

```
00000h  02 00 03 75 84 01 7E 44 20 00 E4 1B 44 7A 49 B0
00010h  78 F9 7E F8 02 FF 02 00 19 75 F2 00 75 F1 00 05
00020h  80 80 FC FF FF FF FF FF FF FF FF FF FF FF FF FF
00030h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00040h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00050h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00060h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00070h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00080h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00090h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000A0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000B0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000E0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000F0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

DataLen 037FA Checksum 37AB23 FillBuffer ClearBuffer SaveBuffer

```
Have 512K Bytes external Flash
Prepare for download ...   OK !      [1.782"]
Programming user code ...  OK !      [2.984"]

Complete !(2022-02-22 17:13:26)
```

D:\Demo\Demo.hex

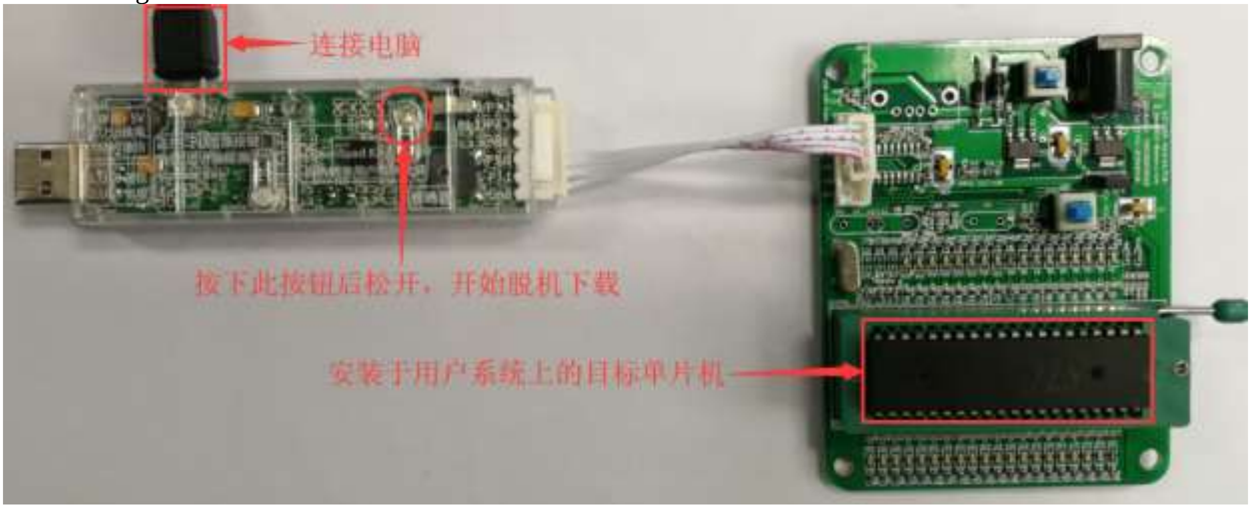
lease Projec Release Help Get HDD-SN □Beep PassTimes 10985 Reset

1. Select the MCU model;
2. Select the serial port number corresponding to U8W-Mini;
3. Open the target file (HEX format or BIN format);
4. Set the hardware options;
5. Select the "U8W Offline/Online" tab, set the offline programming options, pay attention to the S-VCC output voltage matches the target chip working voltage; click the "Download user program to U8/U7 programmer for offline download" button；
6. The step information of the setting process is displayed, and the prompt "Completed!" is displayed after the setting is completed.

Follow the steps in the above figure, after the operation is completed, if the download is successful, it means that the user code and related setting options have been downloaded to the U8W-Mini download tool.

It is recommended that users use the latest version of STC-ISP to download the software (please pay attention to the updates of the STC-ISP download software on the STC official website http://www.STCMCUDATA.com. It is strongly recommended that users download the software from the official website http://www.STCMCUDATA.com).

(3) Then use the cable to connect the computer, connect the U8W-Mini download tool and the user system (target MCU) as shown in the figure below, and press the button shown in the figure and release it to start offline downloading :

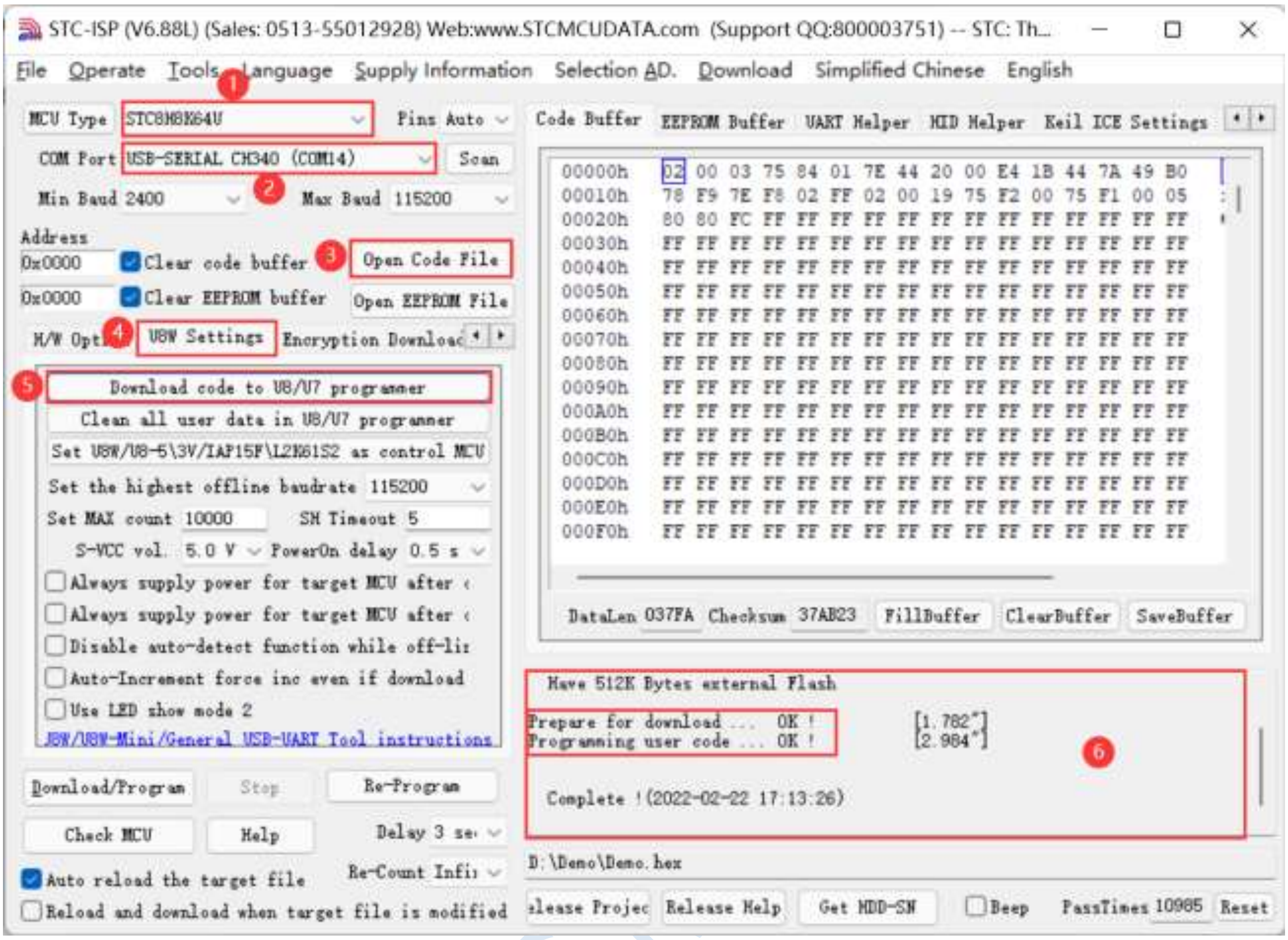


During the downloading process, the 4 LEDs on the U8W-Mini download tool will be displayed in marquee mode. After the download is complete, if the download is successful, the 4 LEDs will be on and off at the same time; if the download fails, all the 4 LEDs will be off.

The target chip is connected to the U8W-Mini by the user system lead, and the U8W-Mini is powered by the user system for offline download.

(1) Firstly, use the USB cable provided by STC to connect the U8W-Mini download board to the computer, as shown below:

(2) Set up in the STC-ISP download software according to the steps shown in the figure below:


STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...
File Operate Tools Language Supply Information Selection AD. Download Simplified Chinese English
1
2
3
4
5
6
Scan
Open Code File
Open EEPROM File
H/W Opt
U8W Settings
Encryption Download
Download code to U8/U7 programmer
Clean all user data in U8/U7 programmer
Set U8W/U8-5\3V/IAP15F\L2K61S2 as control MCU
J8W/U8W-Mini/General USB-UART Tool instructions
Download/Program
Stop
Re-Program
Check MCU
Help
Code Buffer
EEPROM Buffer
UART Helper
HID Helper
Keil ICE Settings
FillBuffer
ClearBuffer
SaveBuffer
Release Project
Release Help
Get HDD-SN
Reset

MCU Type STC8H8K64U Pins Auto

COM Port USB-SERIAL CH340 (COM14)

Min Baud 2400 Max Baud 115200

Address

0x0000 ☑ Clear code buffer

0x0000 ☑ Clear EEPROM buffer

Set the highest offline baudrate 115200

Set MAX count 10000 SN Timeout 5

S-VCC vol. 5.0 V PowerOn delay 0.5 s

☐ Always supply power for target MCU after

☐ Always supply power for target MCU after

☐ Disable auto-detect function while off-li

☐ Auto-Increment force inc even if download

☐ Use LED show mode 2

Delay 3 se

☑ Auto reload the target file Re-Count Infi

☐ Reload and download when target file is modified

```
00000h  02 00 03 75 84 01 7E 44 20 00 E4 1B 44 7A 49 B0
00010h  78 F9 7E F8 02 FF 02 00 19 75 F2 00 75 F1 00 05
00020h  80 80 FC FF FF FF FF FF FF FF FF FF FF FF FF FF
00030h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00040h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00050h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00060h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00070h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00080h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00090h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000A0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000B0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000E0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000F0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

DataLen 037FA Checksum 37AB23

```
Have 512K Bytes external Flash
Prepare for download ...    OK !        [1.782"]
Programming user code ...   OK !        [2.984"]

Complete !(2022-02-22 17:13:26)
```

D:\Demo\Demo.hex

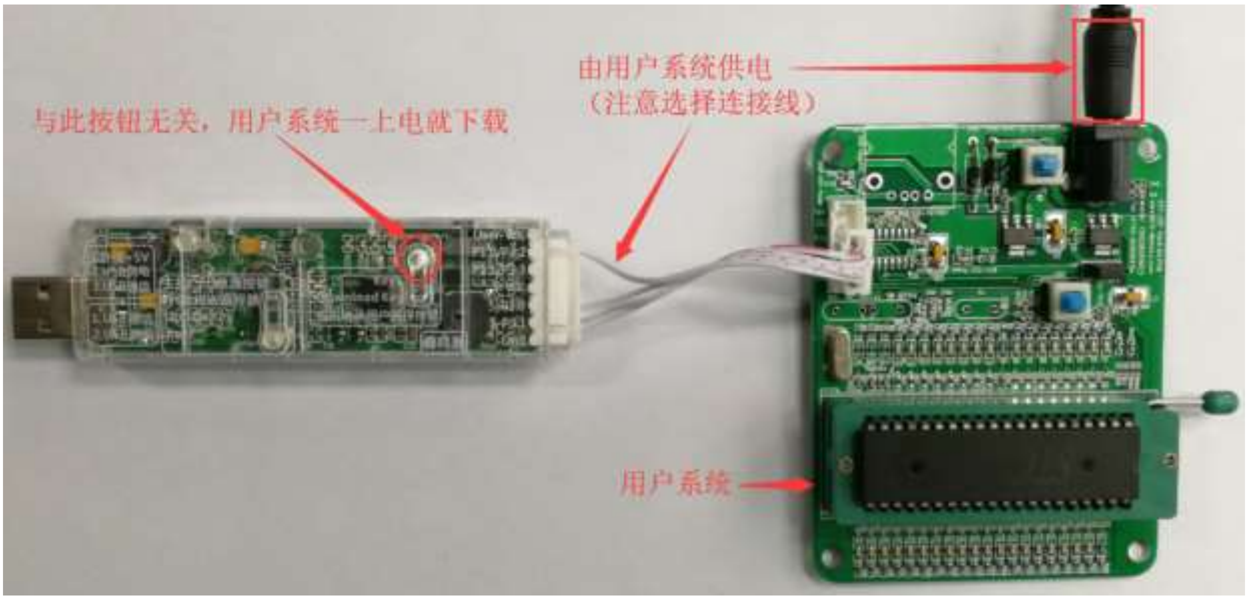
☐ Beep PassTimes 10985

1. Select the MCU model;
2. Select the serial port number corresponding to U8W-Mini;
3. Open the target file (HEX format or BIN format);
4. Set the hardware options;
5. Select the "U8W Offline/Online" tab, set the offline programming options, pay attention to the S-VCC output voltage matches the target chip working voltage; click the "Download user program to U8/U7 programmer for offline download" button ;
6. The step information of the setting process is displayed, and the prompt "Completed!" is displayed after the setting is completed.

Follow the steps in the above figure, after the operation is completed, if the download is successful, it means that the user code and related setting options have been downloaded to the U8W-Mini download tool.

It is recommended that users use the latest version of STC-ISP to download the software (please pay attention to the updates of the STC-ISP download software on the STC official website http://www.STCMCUDATA.com. It is strongly recommended that users download the software from the official website http://www.STCMCUDATA.com) Software used).

(3) Then connect U8W-Mini to the user system as shown in the figure below. Once the user system is powered on, it will start offline downloading:

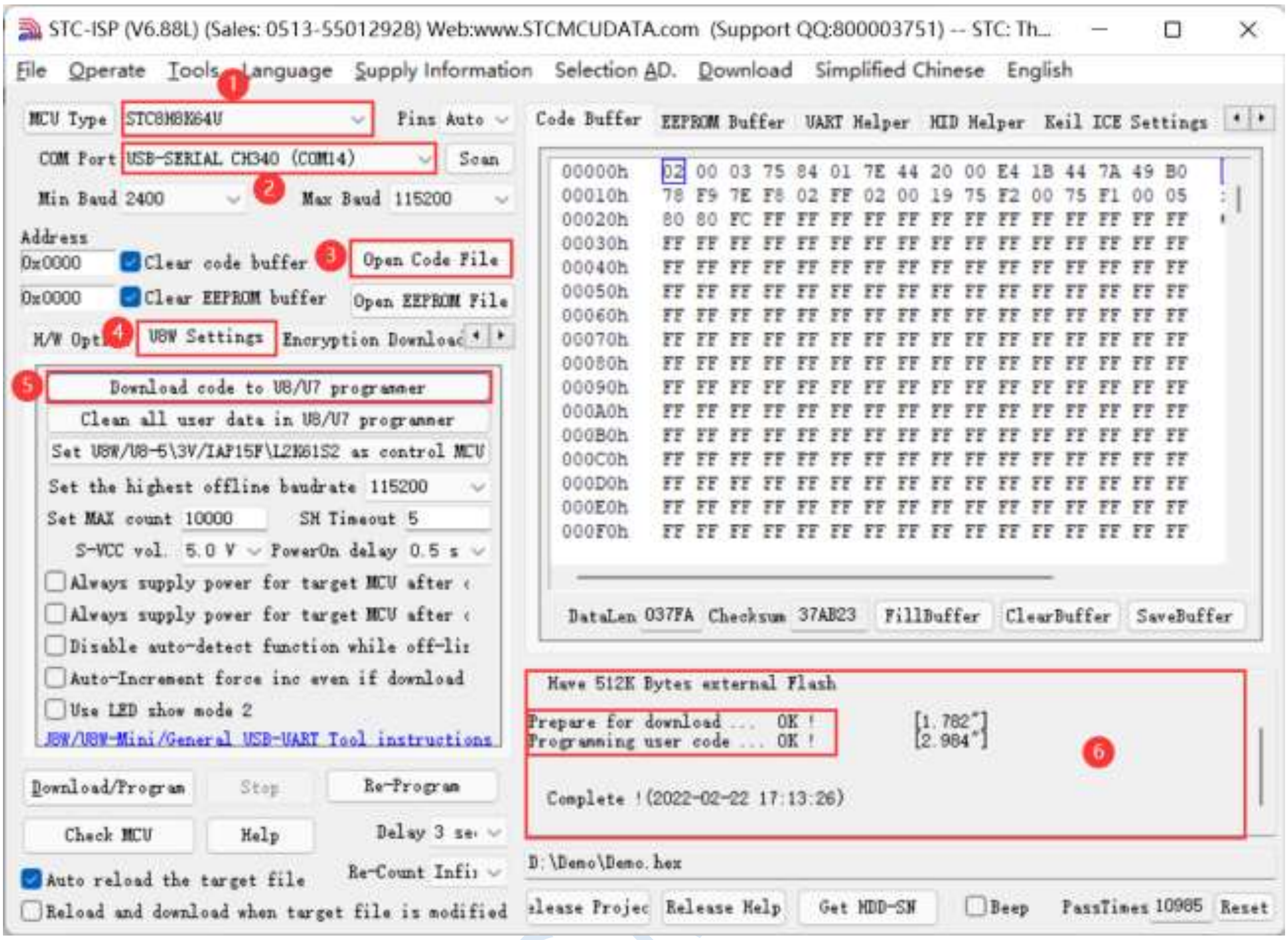


During the downloading process, the 4 LEDs on the U8W-Mini download tool will be displayed in marquee mode. After the download is complete, if the download is successful, the 4 LEDs will be on and off at the same time; if the download fails, all the 4 LEDs will be off.

The target chip is connected to U8W-Mini by the user system lead, and U8W-Mini and the user system are independently powered for offline download.

(1) Firstly, use the USB cable provided by STC to connect the U8W-Mini download board to the computer, as shown below:

(2) Set up in the STC-ISP download software according to the steps shown in the figure below:


STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...
File Operate Tools Language Supply Information Selection AD. Download Simplified Chinese English
Code Buffer EEPROM Buffer UART Helper HID Helper Keil ICE Settings
Scan
Open Code File
Open EEPROM File
H/W Opt
U8W Settings
Encryption Download
Download code to U8/U7 programmer
Clean all user data in U8/U7 programmer
Set U8W/U8-5\3V/IAP15F\L2K61S2 as control MCU
U8W/U8W-Mini/General USB-UART Tool instructions
Download/Program
Stop
Re-Program
Check MCU
Help
FillBuffer
ClearBuffer
SaveBuffer
Release Project
Release Help
Get HDD-SN
Reset
1
2
3
4
5
6

MCU Type STC8H8K64U

COM Port USB-SERIAL CH340 (COM14)

Pins Auto

Min Baud 2400 Max Baud 115200

Address
0x0000 ☑ Clear code buffer
0x0000 ☑ Clear EEPROM buffer

Set the highest offline baudrate 115200
Set MAX count 10000 SN Timeout 5
S-VCC vol. 5.0 V PowerOn delay 0.5 s
☐ Always supply power for target MCU after (
☐ Always supply power for target MCU after (
☐ Disable auto-detect function while off-lir
☐ Auto-Increment force inc even if download
☐ Use LED show mode 2

Delay 3 sec
☑ Auto reload the target file Re-Count Infinity
☐ Reload and download when target file is modified

```
00000h   02 00 03 75 84 01 7E 44 20 00 E4 1B 44 7A 49 B0
00010h   78 F9 7E F8 02 FF 02 00 19 75 F2 00 75 F1 00 05
00020h   80 80 FC FF FF FF FF FF FF FF FF FF FF FF FF FF
00030h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00040h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00050h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00060h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00070h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00080h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00090h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000A0h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000B0h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C0h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D0h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000E0h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000F0h   FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

DataLen 037FA Checksum 37AB23

```
Have 512K Bytes external Flash
Prepare for download ...   OK !        [1.782"]
Programming user code ...  OK !        [2.984"]

Complete !(2022-02-22 17:13:26)
```

D:\Demo\Demo.hex

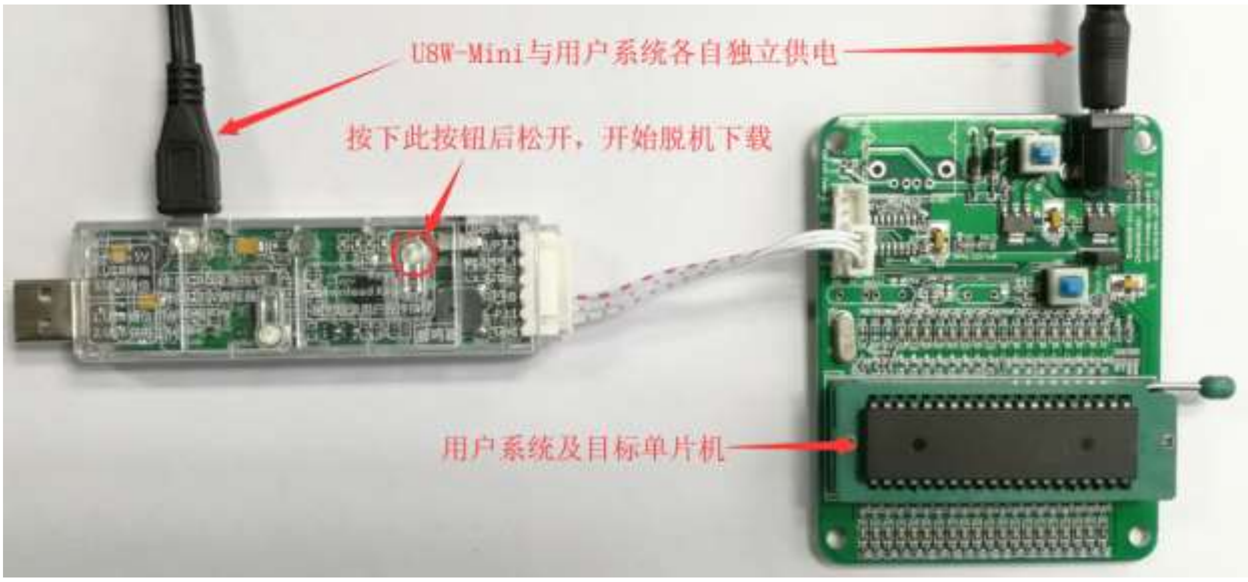
☐ Beep PassTimes 10985

1. Select the MCU model;
2. Select the serial port number corresponding to U8W-Mini;
3. Open the target file (HEX format or BIN format);
4. Set the hardware options;
5. Select the "U8W Offline/Online" tab, set the offline programming options, pay attention to the S-VCC output voltage matches the target chip working voltage; click the "Download user program to U8/U7 programmer for offline download" button ;
6. The step information of the setting process is displayed, and the prompt "Completed!" is displayed after the setting is completed.

Follow the steps in the above figure, after the operation is completed, if the download is successful, it means that the user code and related setting options have been downloaded to the U8W-Mini download tool.

It is recommended that users use the latest version of STC-ISP to download the software (please pay attention to the updates of the STC-ISP download software on the STC official website http://www.STCMCUDATA.com. It is strongly recommended that users download the software from the official website http://www.STCMCUDATA.com).

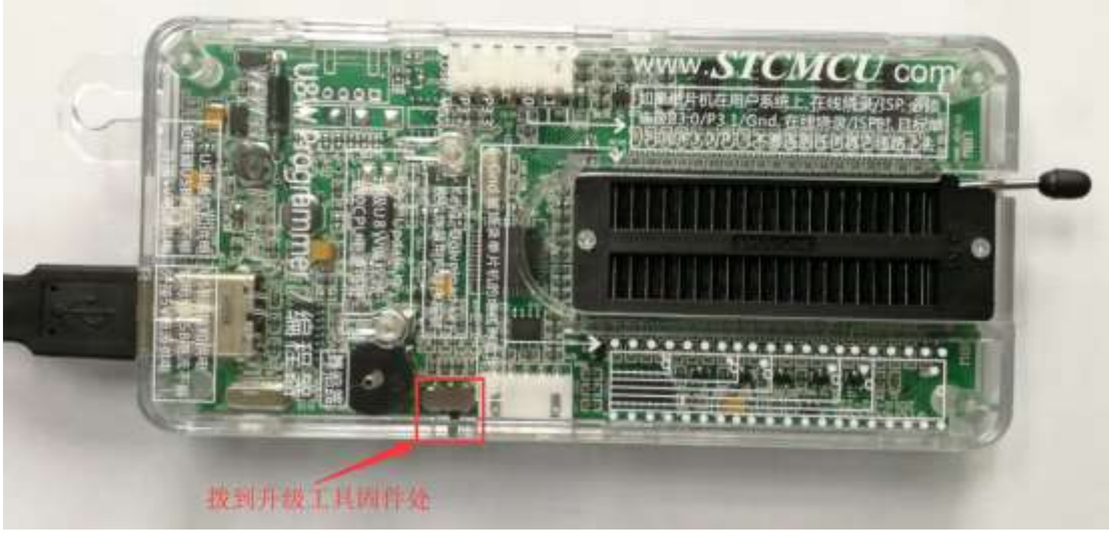
(3) Then connect U8W-Mini to the user system as shown in the figure below, and press the button shown in the figure first and then release it, ready to start offline download, and finally power on/on the user system to download The user program officially starts:



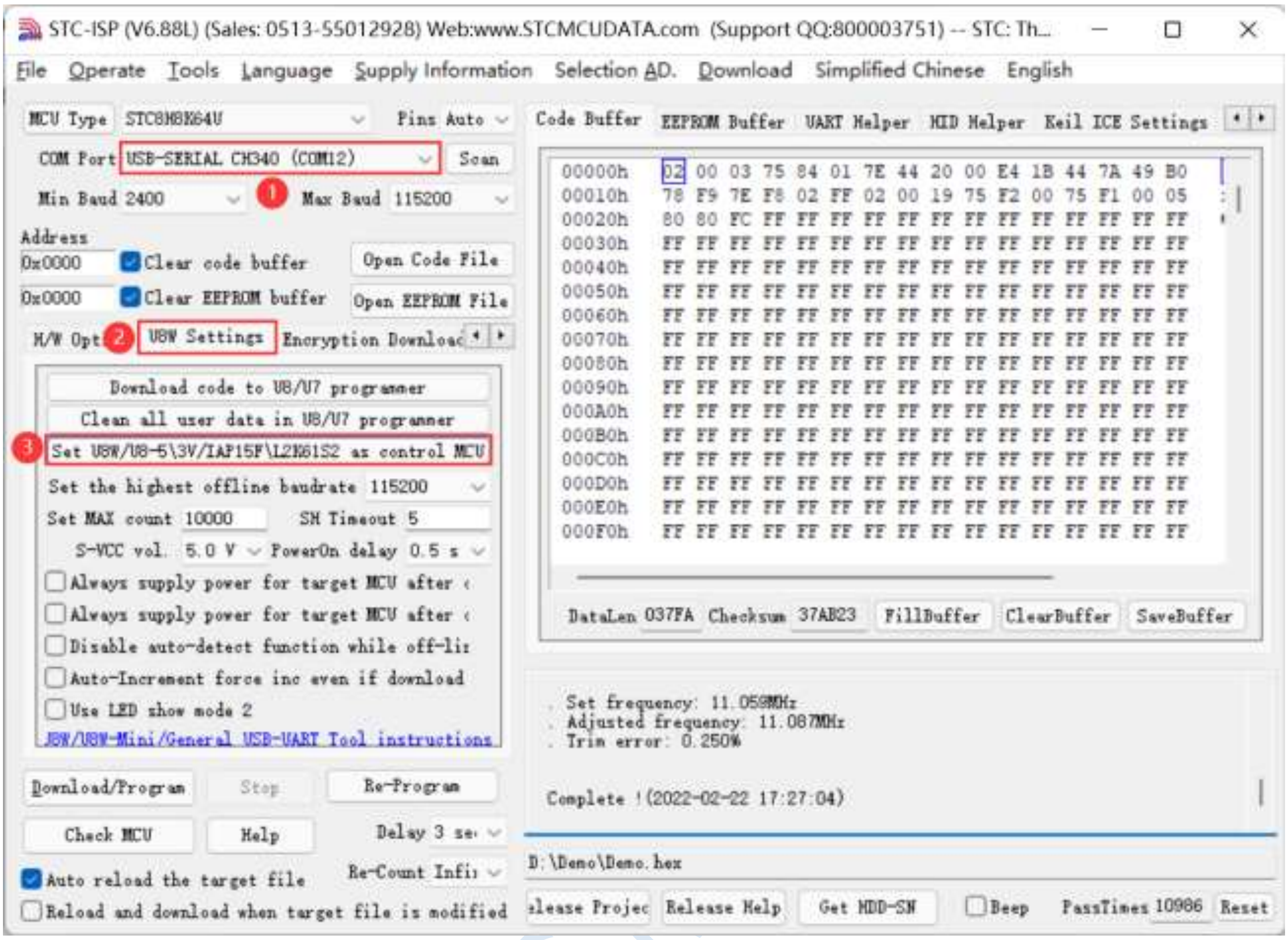
During the downloading process, the 4 LEDs on the U8W-Mini download tool will be displayed in marquee mode. After the download is complete, if the download is successful, the 4 LEDs will be on and off at the same time; if the download fails, all the 4 LEDs will be off.

H.3.8 Make/Update U8W/U8W-Mini

The process of making U8W/U8W-Mini download master is similar. To save space, the following uses U8W as an example to describe how to make U8W download master. Before making the U8W download master, you need to dial the "Update/Download Selection Interface" of the U8W download board to "Upgrade Tool Firmware", as shown in the figure below:



Then click the "Set U8W/U8-5V/U8-3V as the offline download master chip" button on the "U8W Offline/Online" page in the STC-ISP download program, as shown in the figure below: (Note: Be sure to select The serial port corresponding to U8W)



MCU Type STC8H8K64U Pins Auto

COM Port USB-SERIAL CH340 (COM12)

Min Baud 2400 Max Baud 115200

Address

0x0000 ☑ Clear code buffer

0x0000 ☑ Clear EEPROM buffer

Set the highest offline baudrate 115200

Set MAX count 10000 SN Timeout 5

S-VCC vol. 5.0 V PowerOn delay 0.5 s

□ Always supply power for target MCU after (

□ Always supply power for target MCU after (

□ Disable auto-detect function while off-li

□ Auto-Increment force inc even if download

□ Use LED show mode 2

Delay 3 se

☑ Auto reload the target file Re-Count Infi

□ Reload and download when target file is modified

```
00000h  02 00 03 75 84 01 7E 44 20 00 E4 1B 44 7A 49 B0
00010h  78 F9 7E F8 02 FF 02 00 19 75 F2 00 75 F1 00 05
00020h  80 80 FC FF FF FF FF FF FF FF FF FF FF FF FF FF
00030h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00040h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00050h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00060h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00070h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00080h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00090h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000A0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000B0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000E0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000F0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

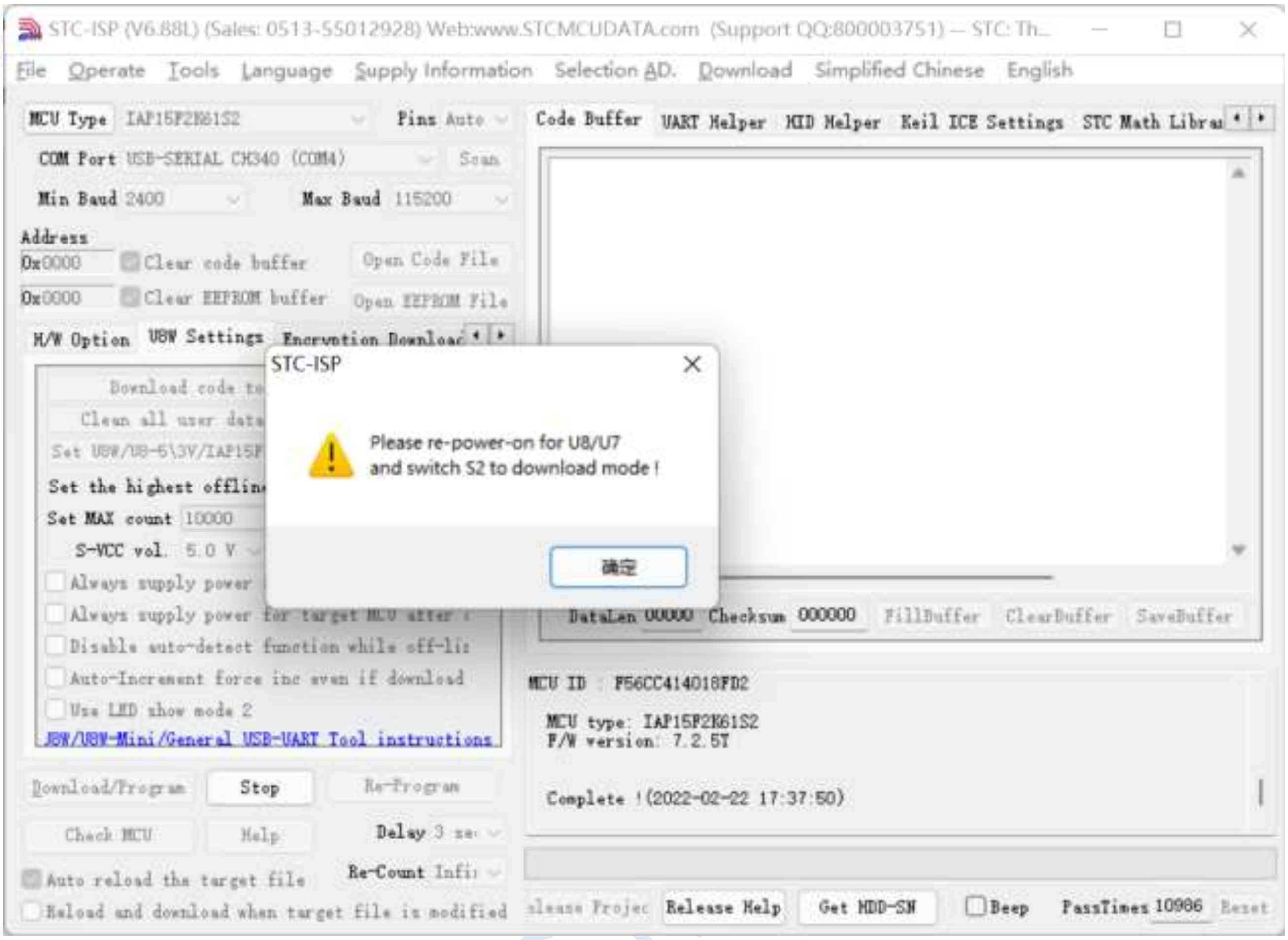
DataLen 037FA Checksum 37AB23

```
. Set frequency: 11.059MHz
. Adjusted frequency: 11.087MHz
. Trim error: 0.250%

Complete !(2022-02-22 17:27:04)
```

D:\Demo\Demo.hex

□ Beep PassTimes 10986

When the following screen appears, it indicates that the U8W control chip is made:


STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...
File Operate Tools Language Supply Information Selection AD. Download Simplified Chinese English
Code Buffer UART Helper HID Helper Keil ICE Settings STC Math Librai
H/W Option U8W Settings Encryption Download
Download code to
Clean all user data
Set U8W/U8-5\3V/IAP15F
Open Code File
Open EEPROM File
Scan
FillBuffer ClearBuffer SaveBuffer
Download/Program Stop Re-Program
Check MCU Help
Release Projec Release Help Get HDD-SN Reset
U8W/U8W-Mini/General USB-UART Tool instructions
确定

MCU Type IAP15F2K61S2 Pins Auto

COM Port USB-SERIAL CH340 (COM4)

Min Baud 2400 Max Baud 115200

Address
0x0000 ☑ Clear code buffer
0x0000 ☑ Clear EEPROM buffer

Set the highest offline

Set MAX count 10000

S-VCC vol. 5.0 V

□ Always supply power
□ Always supply power for target MCU after
□ Disable auto-detect function while off-li
□ Auto-Increment force inc even if download
□ Use LED show mode 2

Delay 3 se

Re-Count Infi

☑ Auto reload the target file
□ Reload and download when target file is modified

DataLen 00000 Checksum 000000

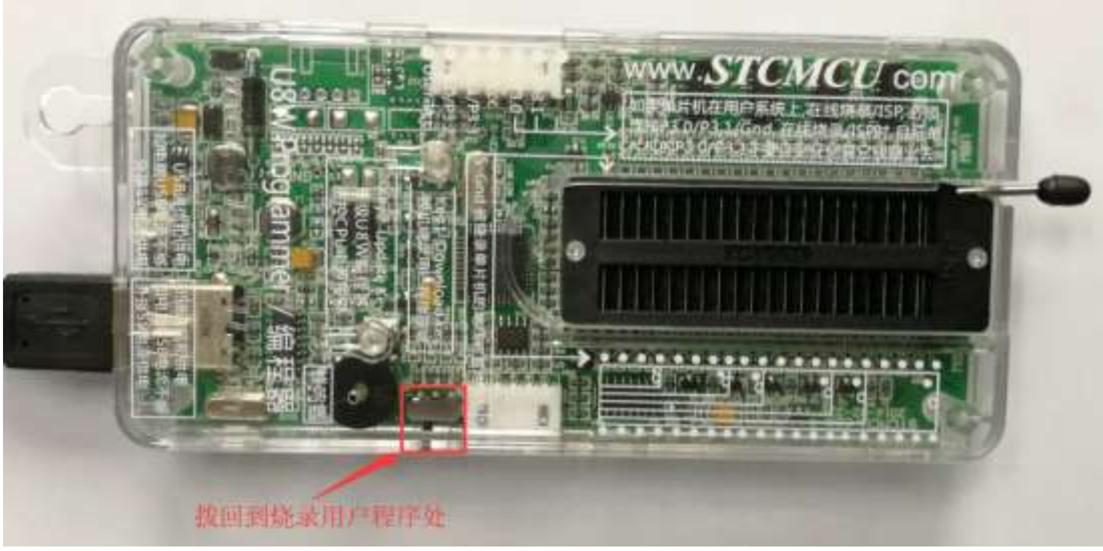
STC-ISP

Please re-power-on for U8/U7
and switch S2 to download mode !

```
MCU ID : F56CC414018FD2

 MCU type: IAP15F2K61S2
 F/W version: 7.2.5T


 Complete !(2022-02-22 17:37:50)
```

□ Beep PassTimes 10986

After the production is completed, do not forget to dial the "Update/Download Selection Interface" of U8W back to the "Burn User Program" mode, and power on the U8W download tool again, as shown in the figure below: (Otherwise, programming will not be performed normally)



H.3.9 U8W/U8W-Mini set through mode (can be used for simulation)

To use U8W/U8-Mini for simulation, you must first set U8W/U8-Mini to pass-through mode. The method of U8W/U8W-Mini to realize USB to serial port pass-through mode is as follows:

1. First, the U8W/U8W-Mini firmware must be upgraded to v1.37 and above;

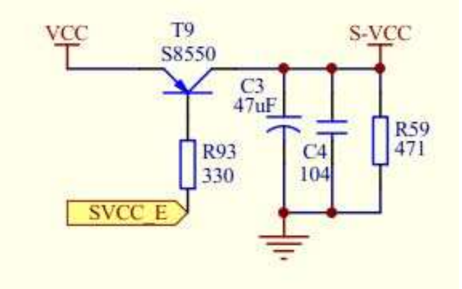
2. After U8W/U8W-Mini is powered on, it is in normal download mode. At this time, press and hold the Key1 (download) button on the tool and do not release it. Press the Key2 (power) button again, and then release the Key2 (power) button. Then release the Key1 (download) button, U8W/U8W-Mini will enter the USB to serial port pass-through mode. (Press Key1→ Press Key2 → Release Key2 → Release Key1);
3. The U8W/U8W-Mini tool that enters the pass-through mode is just a simple USB to serial port and does not have the offline download function. If you need to restore the original functions of U8W/U8W-Mini only need to press the Key2 (power) button separately again.

H.3.10 Reference circuit of U8W/U8W-Mini

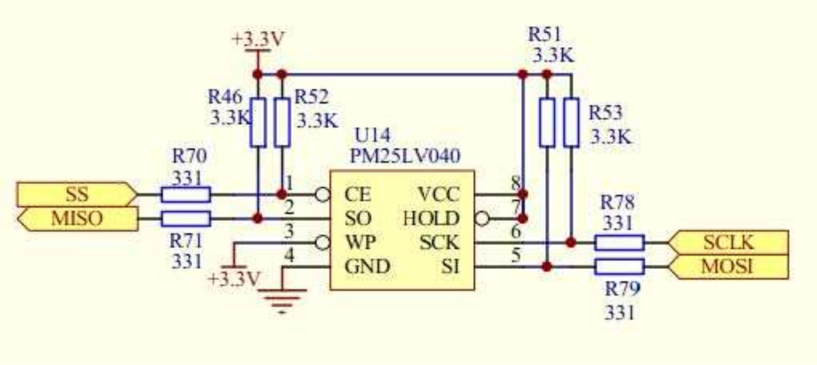
USB online/offline download board U8W/U8W-Mini provides users with the following common control interfaces:

Pin function	Port	Function description
Power control pin	P2.6	Low effective
Download communication pin	P1.0	Serial port RXD, connect to the TXD of the target chip (P3.1)
	P1.1	Serial port TXD, connect to RXD of target chip (P3.0)
Programming button	P3.6	Low effective
display	P3.2	LED1
	P3.3	LED2
	P3.4	LED3
	P5.5	LED4
External serial Flash control pin	P2.4	Flash CE Pin
	P2.2	Flash SO Pin
	P2.3	Flash SI Pin
	P2.1	Flash SCLK Pin
Automatic burning tool Sorter signal	P3.6	Start signal
	P1.5	Completion signal
	P5.4	OK signal (good signal)
	P3.7	ERROR signal (defective product signal)
Buzzer (BEEP) control	P2.5	High effective (sound at high level)

Reference circuit diagram for power control part:

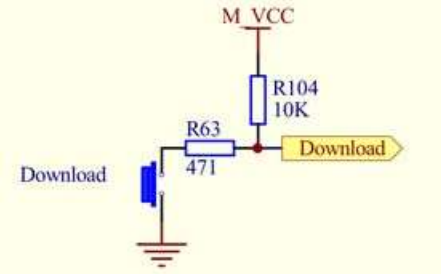


Reference circuit diagram of Flash control part:

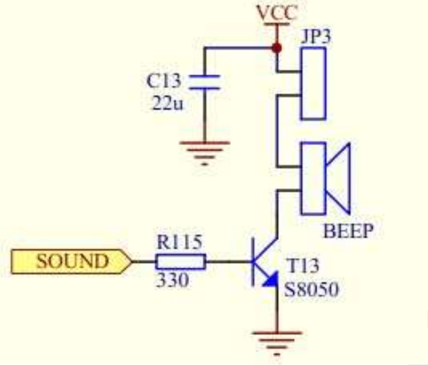


This Flash memory is required when the user program is larger than 41K

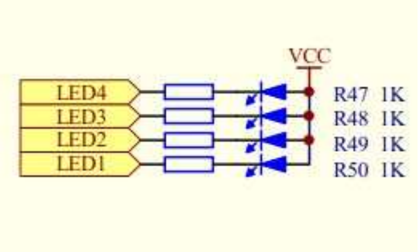
The reference circuit diagram of the button part:



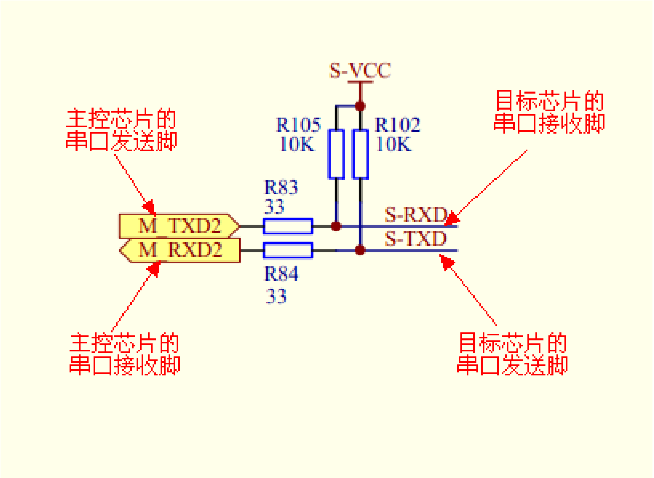
Reference circuit diagram of the buzzer part:



LED display part reference circuit diagram:



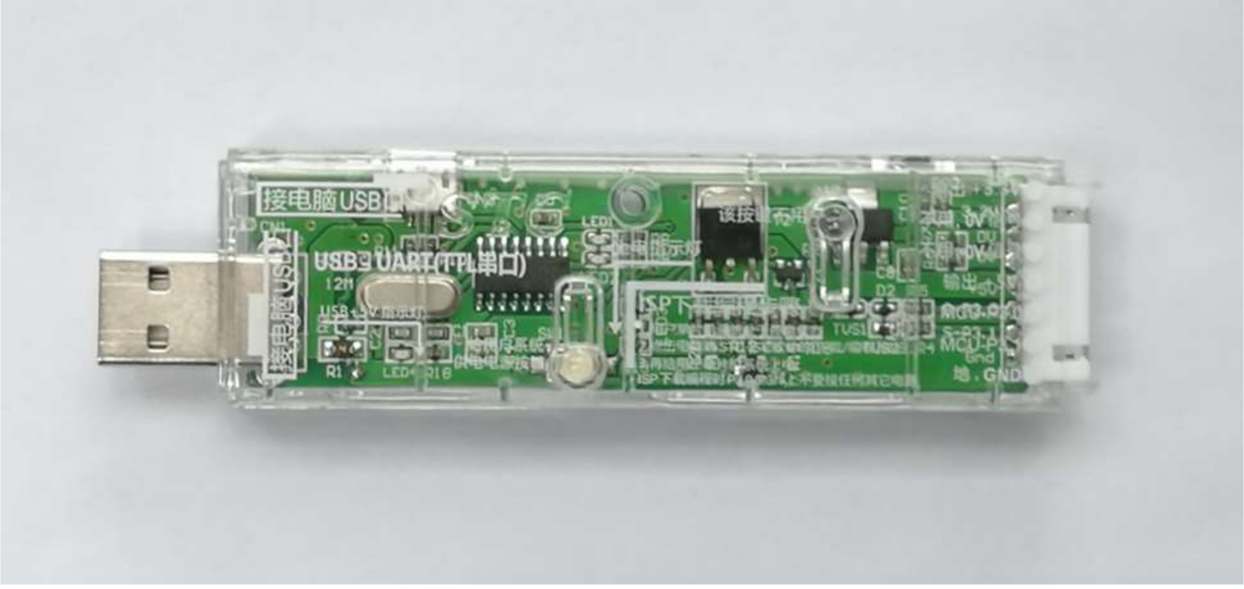
Reference circuit diagram of serial port communication pin connection part:



H.4 STC Universal USB to Serial Tool

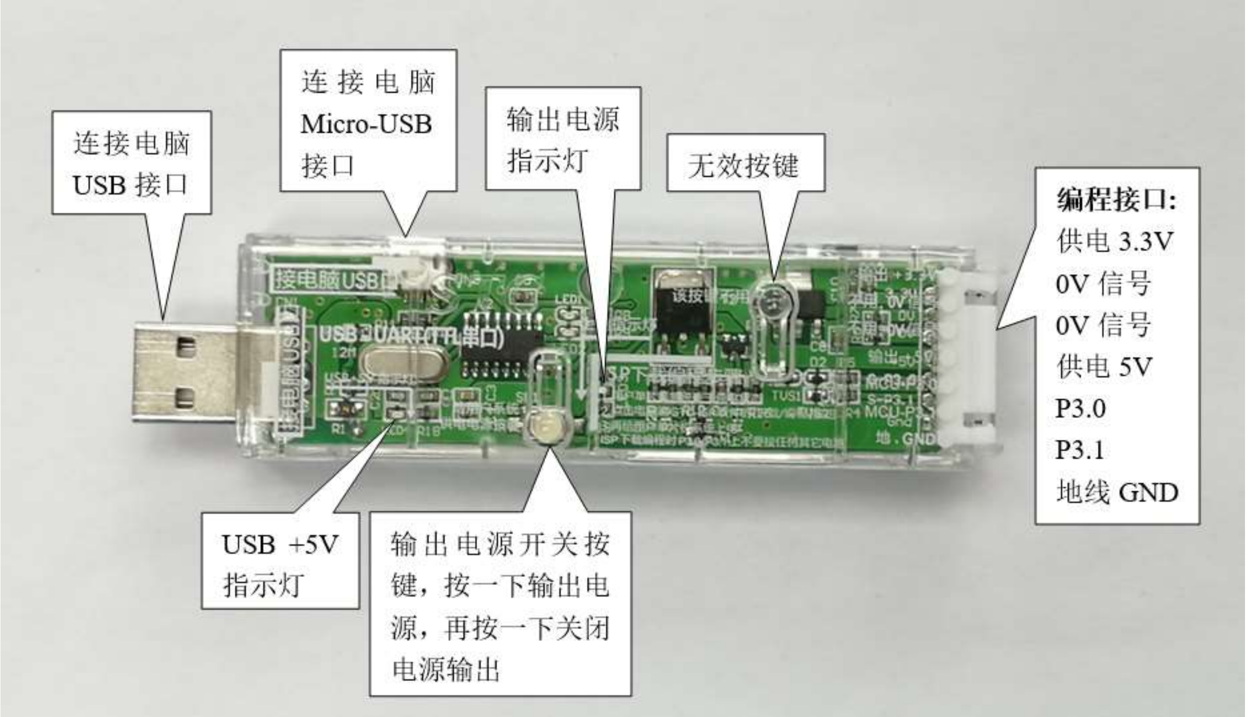
H.4.1 Appearance of STC Universal USB to Serial Tool

Front:



Back:

H.4.2 STC general USB to serial tool layout diagram



Here, the "power switch" needs to be explained:

The function of this button is the same as the self-locking switch. When the switch button is pressed for the first time, the switch is turned on and held, that is, self-locking. When the switch button is pressed for the second time, the switch turns off the power. In view of the characteristics of self-locking switches that are easily damaged during use, we have designed a set of circuits that use light touch switches to replace self-locking switches to increase the service life of the tools.

For STC microcontrollers, if you want to perform ISP download, you must receive the serial port command at power-on reset to start executing the ISP program, so the correct steps to download the program to the MCU using the STC universal USB to serial tool are:

1. Use STC universal USB to serial port tool to connect the MCU to be burned with the computer;
2. Open STC's ISP to download the software;
3. Select the MCU model;
4. Select the serial port corresponding to the STC Universal USB to Serial Tool;
5. Open the target file (HEX format or BIN format);
6. Click the "download/program" button in the ISP download software;
7. Press the "power switch" on the STC Universal USB to Serial Tool to power the MCU, and the download can start.

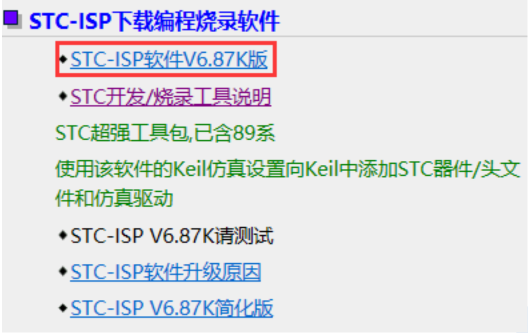
【Cold start burning】

In addition, the USB interface has the same function as the Micro-USB interface, and the user can connect one of the interfaces to the computer as needed.

The 0V signal pin of the programming interface has a 470 ohm resistor grounded. It can be downloaded only when P1.0/P1.1=0/0 or P3.2/P3.3=0/0 is set. You can set P1.0, P1 .1 or P3.2, P3.3 are connected to the 0V signal pin.

H.4.3 STC Universal USB to Serial Tool Driver Installation

STC general USB to serial port tool uses CH340 USB to serial chip (can plug in crystal oscillator, more accurate), just download the general CH340 serial driver and install it. The following is the CH341SER serial driver provided by STC official website (www.STCMCUDATA.com) Download location:



STC-ISP下载编程烧录软件

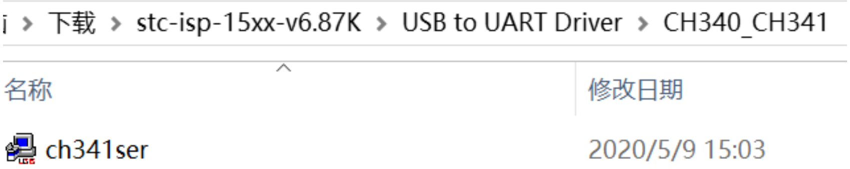
- STC-ISP软件V6.87K版
- STC开发/烧录工具说明

STC超强工具包,已含89系

使用该软件的Keil仿真设置向Keil中添加STC器件/头文件和仿真驱动

- STC-ISP V6.87K请测试
- STC-ISP软件升级原因
- STC-ISP V6.87K简化版

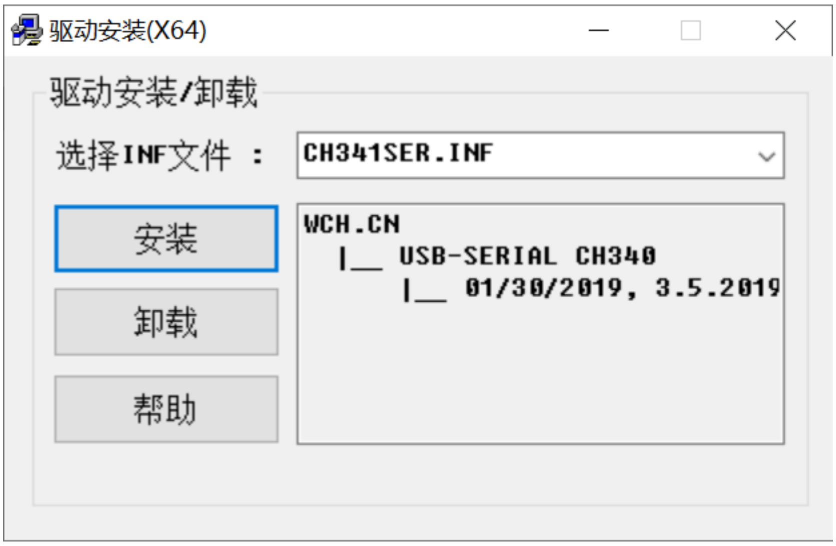
After downloading, decompress, the path of CH340 driver installation package is stc-isp-15xx-v6.87K\USB to UART Driver\CH340_CH341:



į › 下载 › stc-isp-15xx-v6.87K › USB to UART Driver › CH340_CH341

名称	修改日期
ch341ser	2020/5/9 15:03

Take the CH341SER serial port driver provided by STC official website as an example, double-click the "CH341SER.exe" installation package, and click the "Install" button on the pop-up main interface to start installing the driver:



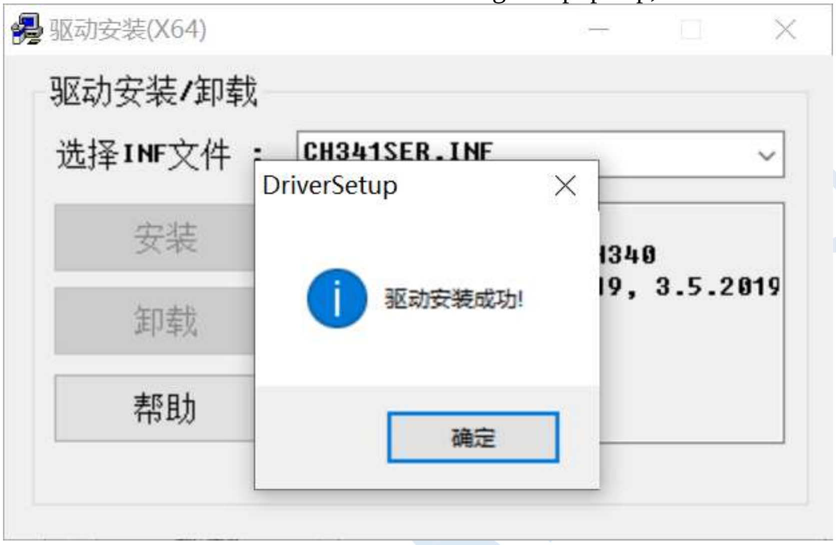
驱动安装(X64)

驱动安装/卸载

选择INF文件 ： CH341SER.INF

```
WCH.CN
  |__ USB-SERIAL CH340
      |__ 01/30/2019, 3.5.2019
```

Then the driver installation successful dialog box pops up, click the "OK" button to complete the installation:

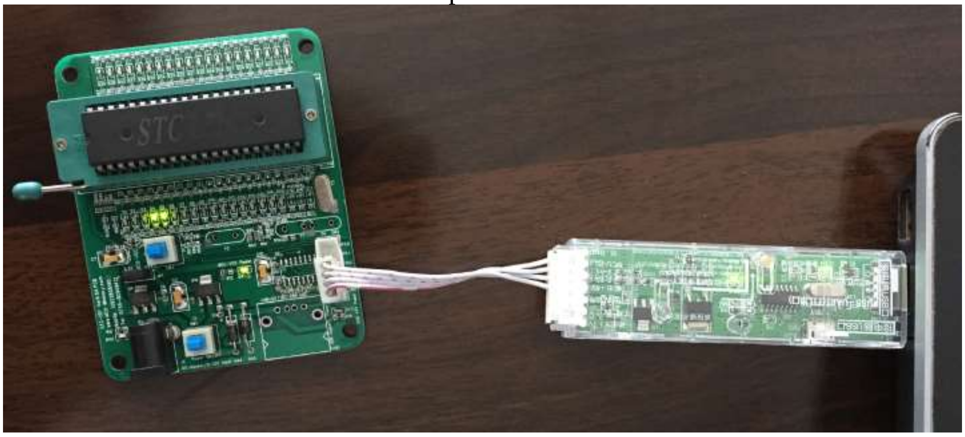


DriverSetup

驱动安装成功!

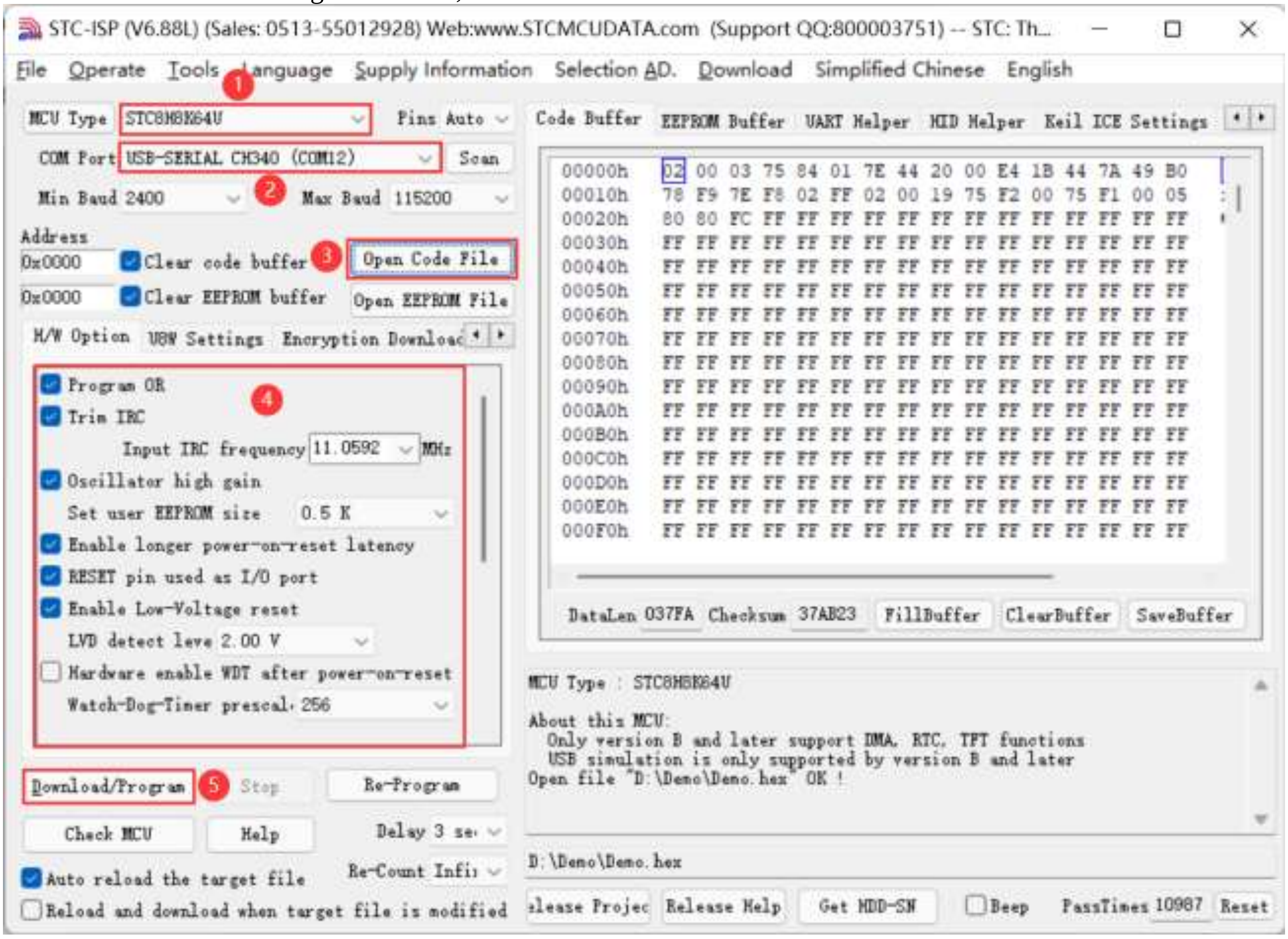
H.4.4 Use STC universal USB to serial port tool to download program to MCU

1. Use the STC universal USB to serial port tool to connect the MCU to be burned to the computer:



2. Open the STC-ISP software;

3. Select the model corresponding to the burning chip;
4. Select the serial port number recognized by the STC universal USB to serial tool (when the STC universal USB to serial tool is correctly connected to the computer, the software will automatically scan and identify the serial port named "USB-SERIAL CH340 (COMx)", the specific COM The number will vary from computer to computer). When multiple USB-to-serial cables are connected to the computer, they must be selected manually;
5. Load the burning program;
6. Set burning options;
7. Click the "Download/Program" button;



MCU Type STC8H8K64U Pins Auto
COM Port USB-SERIAL CH340 (COM12)
Min Baud 2400 Max Baud 115200
Address
0x0000 ☑ Clear code buffer
0x0000 ☑ Clear EEPROM buffer

☑ Program OR
☑ Trim IRC
Input IRC frequency 11.0592 MHz
☑ Oscillator high gain
Set user EEPROM size 0.5 K
☑ Enable longer power-on-reset latency
☑ RESET pin used as I/O port
☑ Enable Low-Voltage reset
LVD detect leve 2.00 V
☐ Hardware enable WDT after power-on-reset
Watch-Dog-Timer prescal 256

Delay 3 se
Re-Count Infi
☑ Auto reload the target file
☐ Reload and download when target file is modified

```
00000h  02 00 03 75 84 01 7E 44 20 00 E4 1B 44 7A 49 B0
00010h  78 F9 7E F8 02 FF 02 00 19 75 F2 00 75 F1 00 05
00020h  80 80 FC FF FF FF FF FF FF FF FF FF FF FF FF FF
00030h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00040h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00050h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00060h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00070h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00080h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00090h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000A0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000B0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000E0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000F0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

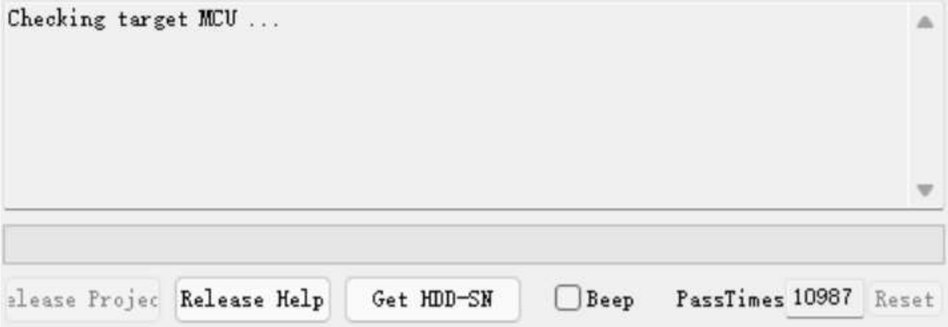
DataLen 037FA Checksum 37AB23

```
MCU Type : STC8H8K64U
About this MCU:
  Only version B and later support DMA, RTC, TFT functions
  USB simulation is only supported by version B and later
Open file "D:\Demo\Demo.hex" OK !
```

D:\Demo\Demo.hex

☐ Beep PassTimes 10987

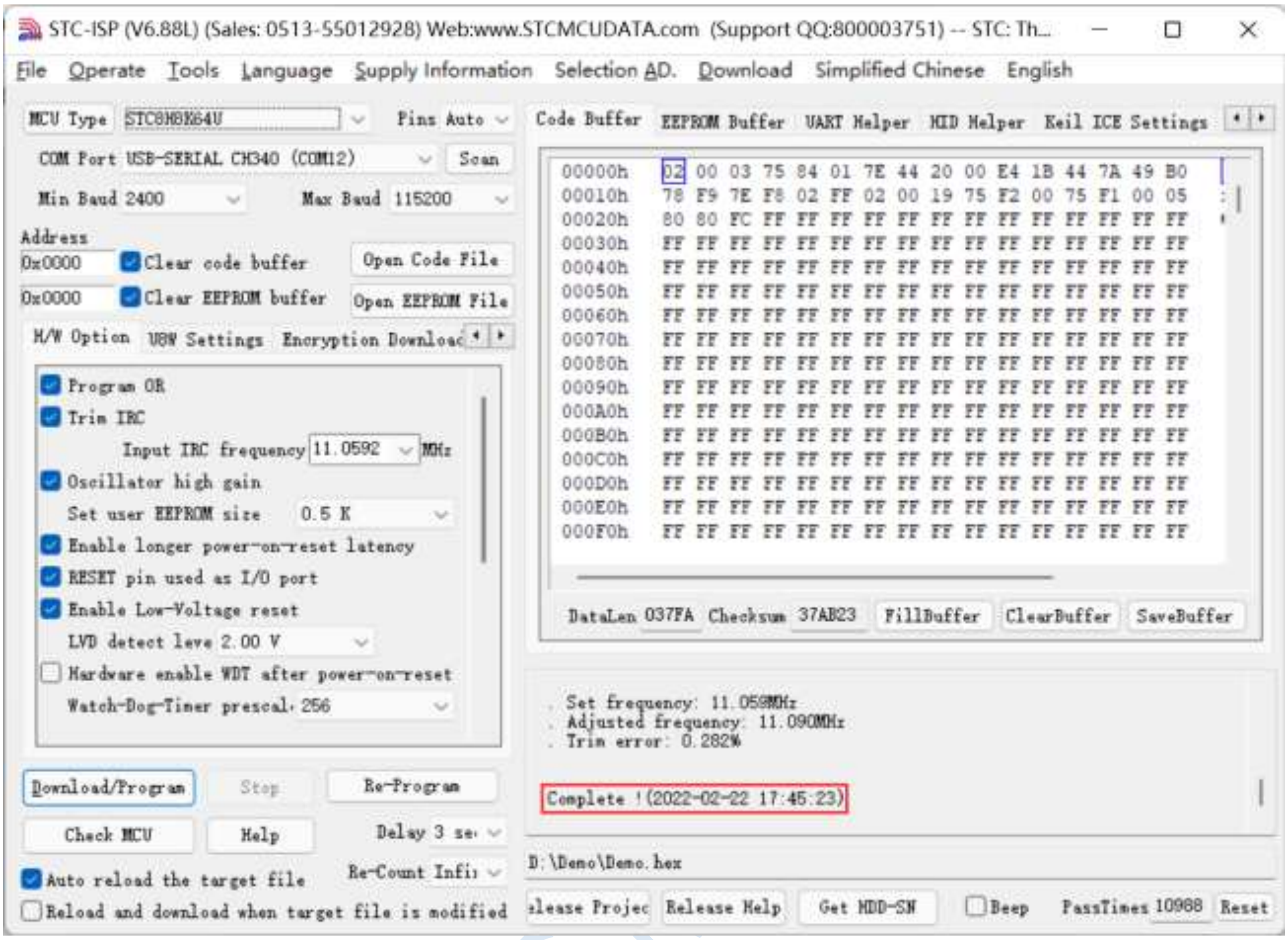
8. When the prompt box in the lower right corner displays "Checking target MCU...", press the "power switch" on the STC universal USB to serial port tool to power on the MCU, and then start downloading [Cold Start Programming];



```
Checking target MCU ...
```

☐ Beep PassTimes 10987

9. Wait for the download to end. If the download is successful, the prompt box in the lower right corner will display "Complete!"



STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...

MCU Type STC8H8K64U
Pins Auto
COM Port USB-SERIAL CH340 (COM12)
Min Baud 2400
Max Baud 115200
Address
0x0000 ☑ Clear code buffer
0x0000 ☑ Clear EEPROM buffer

☑ Program OR
☑ Trim IRC
Input IRC frequency 11.0592 MHz
☑ Oscillator high gain
Set user EEPROM size 0.5 K
☑ Enable longer power-on-reset latency
☑ RESET pin used as I/O port
☑ Enable Low-Voltage reset
LVD detect leve 2.00 V
☐ Hardware enable WDT after power-on-reset
Watch-Dog-Timer prescal 256

Delay 3 se
Re-Count Infi
☑ Auto reload the target file
☐ Reload and download when target file is modified

```
00000h  02 00 03 75 84 01 7E 44 20 00 E4 1B 44 7A 49 B0
00010h  78 F9 7E F8 02 FF 02 00 19 75 F2 00 75 F1 00 05
00020h  80 80 FC FF FF FF FF FF FF FF FF FF FF FF FF FF
00030h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00040h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00050h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00060h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00070h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00080h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00090h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000A0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000B0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000E0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000F0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

DataLen 037FA Checksum 37AB23

```
. Set frequency: 11.059MHz
. Adjusted frequency: 11.090MHz
. Trim error: 0.282%

Complete !(2022-02-22 17:45:23)
```

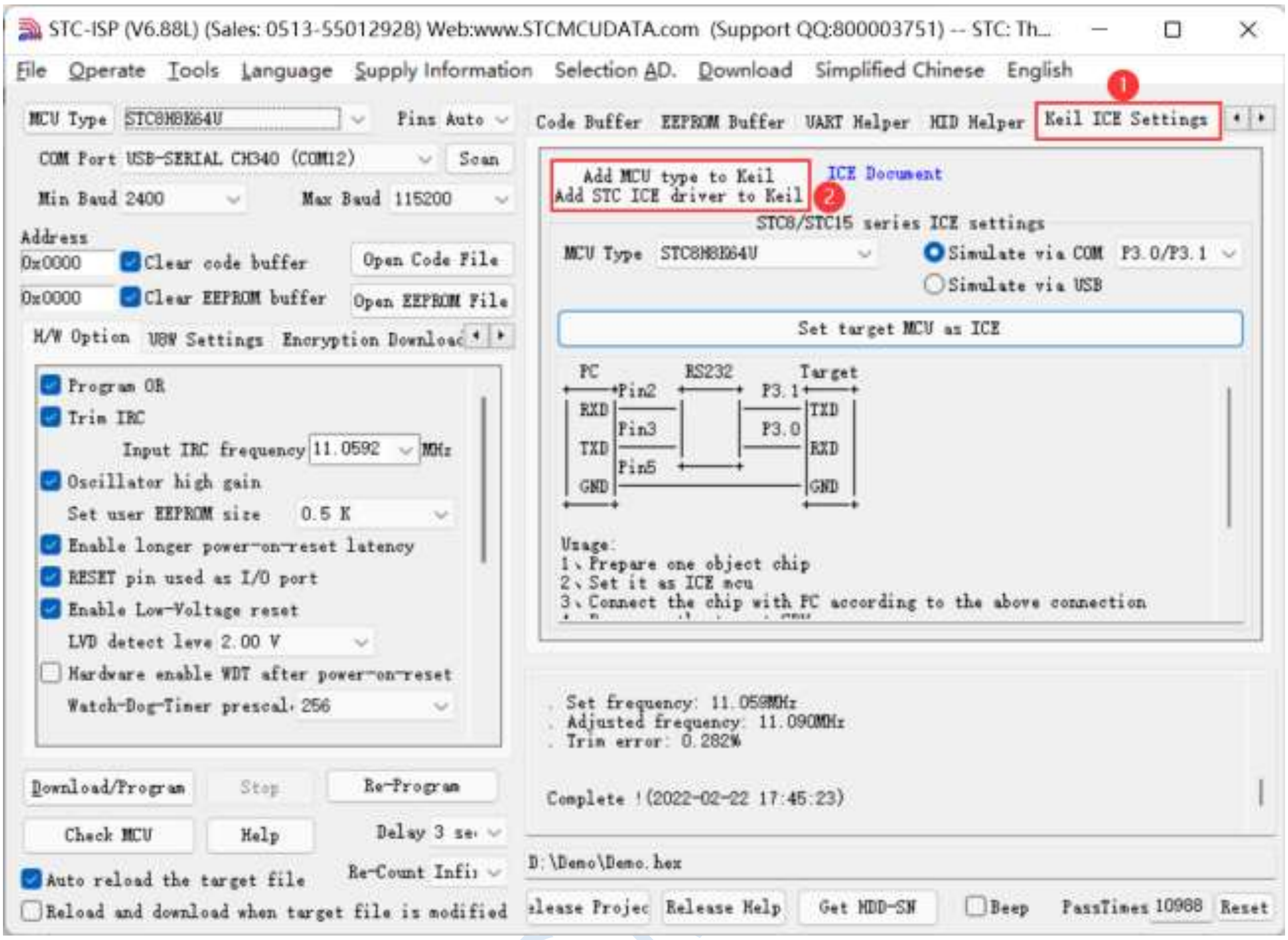
D:\Demo\Demo.hex

☐ Beep PassTimes 10988

H.4.5 Use STC universal USB to serial port tool to simulate user code

The current STC simulation is based on the Keil environment, so if you need to use the STC universal USB to serial port tool to simulate user code, you must install the Keil software.

After the Keil software is installed, you also need to install the STC simulation driver. The installation steps of STC's simulation driver are as follows:
Firstly, open STC-ISP to download the software;
Then click the "Add MCU Type to Keil / Add STC ICE driver to Keil" on the "Keil ICE settings" page in the functional area on the right side of the software:



STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...

File Operate Tools Language Supply Information Selection AD. Download Simplified Chinese English
1
2
Code Buffer EEPROM Buffer UART Helper HID Helper Keil ICE Settings
Add MCU type to Keil
Add STC ICE driver to Keil
ICE Document
Scan
Open Code File
Open EEPROM File
H/W Option USB Settings Encryption Download
Set target MCU as ICE
PC RS232 Target
Pin2 Pin3 Pin5 P3.1 P3.0
RXD TXD GND TXD RXD GND
Download/Program Stop Re-Program
Check MCU Help
Release Project Release Help Get HDD-SN Reset

MCU Type STC8H8K64U
Pins Auto
COM Port USB-SERIAL CH340 (COM12)
Min Baud 2400
Max Baud 115200
Address
0x0000 ☑ Clear code buffer
0x0000 ☑ Clear EEPROM buffer

☑ Program OR
☑ Trim IRC
Input IRC frequency 11.0592 MHz
☑ Oscillator high gain
Set user EEPROM size 0.5 K
☑ Enable longer power-on-reset latency
☑ RESET pin used as I/O port
☑ Enable Low-Voltage reset
LVD detect leve 2.00 V
☐ Hardware enable WDT after power-on-reset
Watch-Dog-Timer prescal 256

Delay 3 sec
☑ Auto reload the target file
Re-Count Infinite
☐ Reload and download when target file is modified

STC8/STC15 series ICE settings
MCU Type STC8H8K64U
● Simulate via COM P3.0/P3.1
○ Simulate via USB

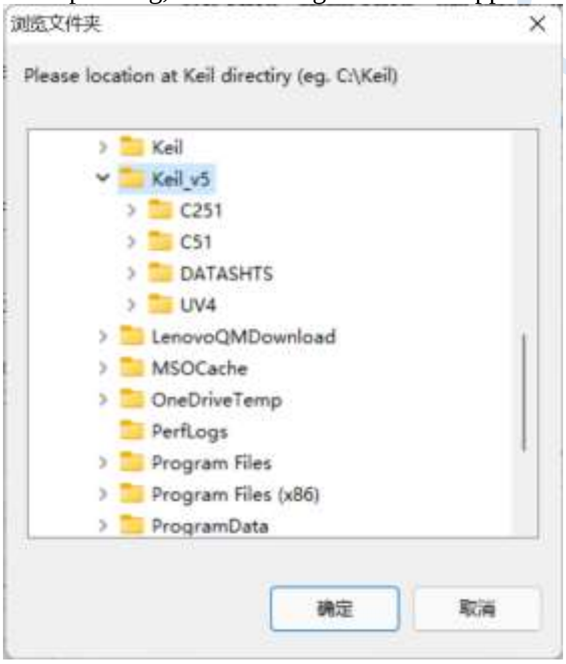
Usage:
1、Prepare one object chip
2、Set it as ICE mcu
3、Connect the chip with PC according to the above connection

```
 Set frequency: 11.059MHz
 Adjusted frequency: 11.090MHz
 Trim error: 0.282%

Complete !(2022-02-22 17:45:23)
```

D:\Demo\Demo.hex

☐ Beep
PassTimes 10988

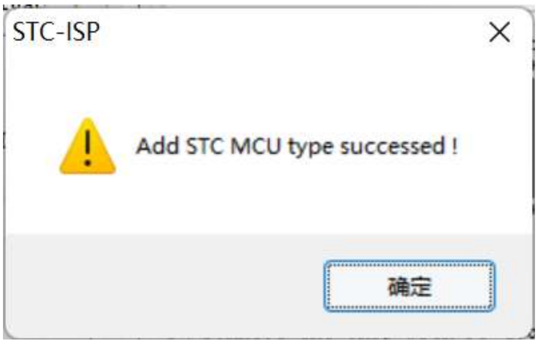
After pressing, the following screen will appear:



浏览文件夹

Please location at Keil directiry (eg. C:\Keil)

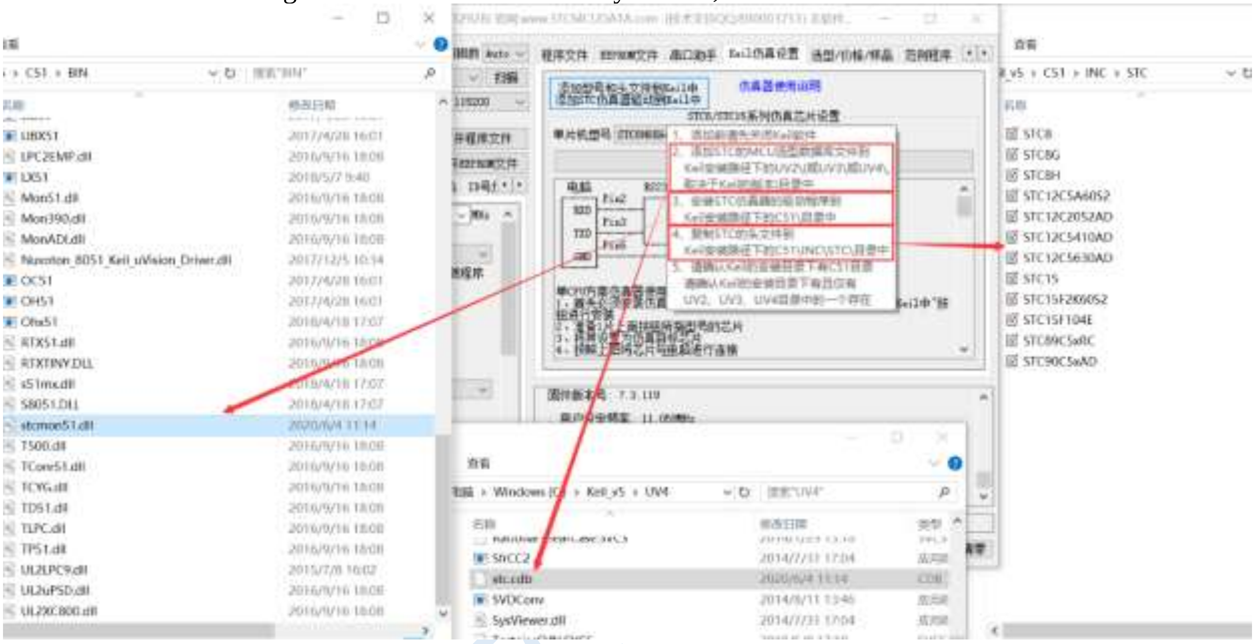
Locate the directory to the installation directory of the Keil software, and then confirm.
After the installation is successful, the following prompt box will pop up:



STC-ISP

Add STC MCU type successed !

You can see the following files in the relevant directory of Keil, which means that the driver is installed correctly.



C51 › BIN

名称	修改日期
UBX51	2017/4/28 16:01
LPC2EMP.dll	2016/9/16 18:08
LX51	2018/5/7 9:40
Mon51.dll	2016/9/16 18:08
Mon390.dll	2016/9/16 18:08
MonADI.dll	2016/9/16 18:08
Nuvoton_8051_Keil_uVision_Driver.dll	2017/12/5 10:14
OC51	2017/4/28 16:01
OH51	2017/4/28 16:01
Ohx51	2018/4/18 17:07
RTX51.dll	2016/9/16 18:08
RTXTINY.DLL	2016/9/16 18:08
s51mx.dll	2018/4/18 17:07
S8051.DLL	2018/4/18 17:07
stcmon51.dll	2020/6/4 11:14
T500.dll	2016/9/16 18:08
TCore51.dll	2016/9/16 18:08
TCYG.dll	2016/9/16 18:08
TDS1.dll	2016/9/16 18:08
TLPC.dll	2016/9/16 18:08
TP51.dll	2016/9/16 18:08
UL2LPC9.dll	2015/7/8 16:02
UL2uPSD.dll	2016/9/16 18:08
UL2XC800.dll	2016/9/16 18:08

Keil_v5 › C51 › INC › STC

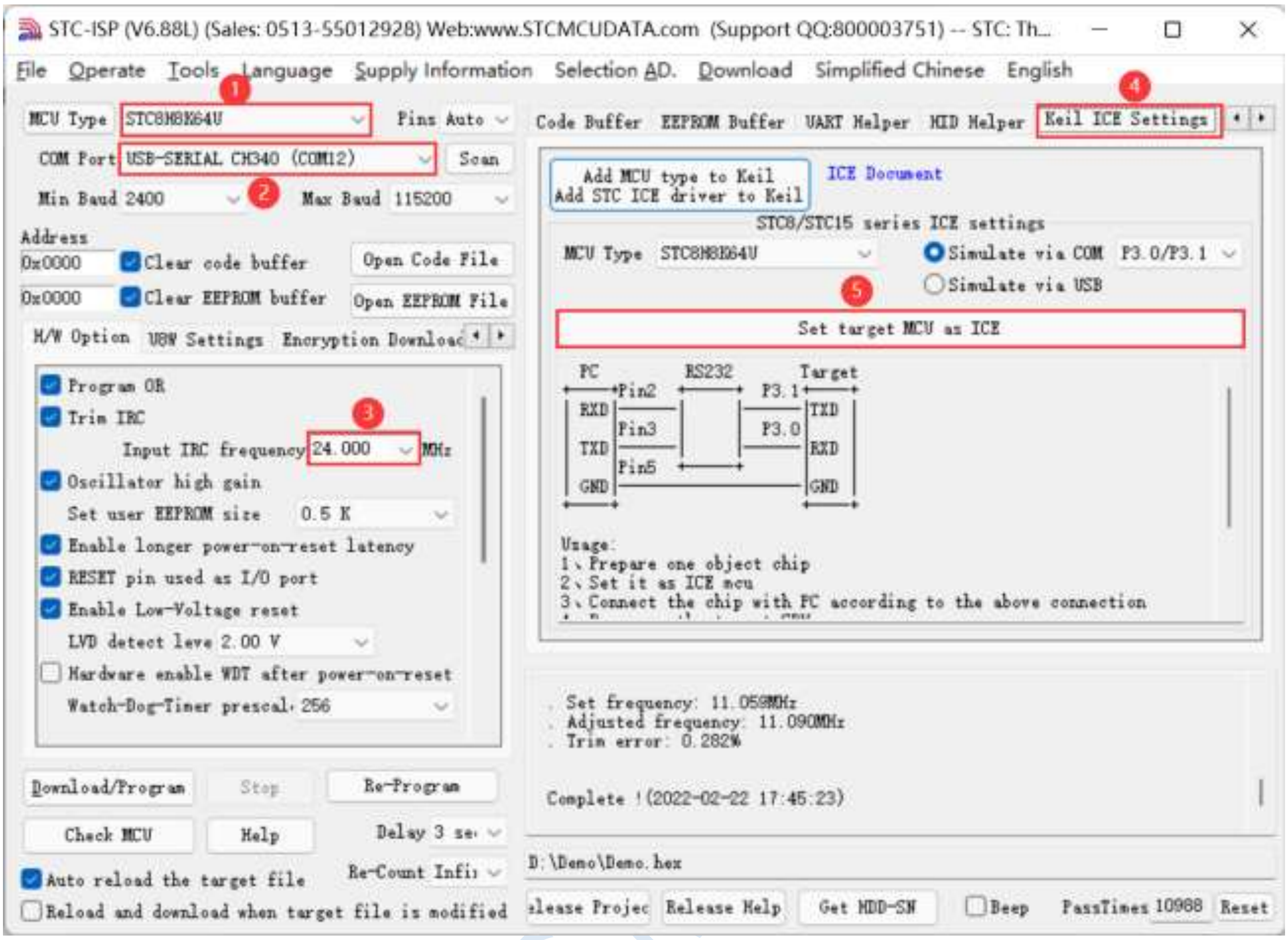
- STC8
- STC8G
- STC8H
- STC12C5A60S2
- STC12C2052AD
- STC12C5410AD
- STC12C5630AD
- STC15
- STC15F2K60S2
- STC15F104E
- STC89C5xRC
- STC90C5xAD

Windows (C:) › Keil_v5 › UV4

名称	修改日期
SnCC2	2014/7/31 17:04
stc.cdb	2020/6/4 11:14
SVDConv	2014/8/11 13:46
SysViewer.dll	2014/7/31 17:04

Since in the default state, the main control chip of STC is not an emulation chip and has no emulation function, if simulation is needed, the main control chip of STC needs to be set as an emulation chip.
The steps of making a simulation chip are as follows:

Firstly, use STC universal USB to serial port tool to connect the MCU to the computer;
Then open STC's ISP to download the software, and select the serial port number corresponding to the serial port tool in the serial port number drop-down list;
Select the MCU model;
Select the IRC frequency of the user program to run, and the frequency selected when making the simulation chip is consistent with the frequency set by the simulated user program, in order to achieve the real running effect.


STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...
File Operate Tools Language Supply Information Selection AD. Download Simplified Chinese English
1
2
3
4
5
Scan
Open Code File
Open EEPROM File
H/W Option USB Settings Encryption Downloac
Code Buffer EEPROM Buffer UART Helper HID Helper Keil ICE Settings
Add MCU type to Keil
Add STC ICE driver to Keil
ICE Document
Set target MCU as ICE
PC RS232 Target
RXD TXD GND Pin2 Pin3 Pin5 P3.1 P3.0 TXD RXD GND
Download/Program Stop Re-Program
Check MCU Help
Release Projec Release Help Get HID-SN
Reset

MCU Type STC8H8K64U
Pins Auto
COM Port USB-SERIAL CH340 (COM12)
Min Baud 2400
Max Baud 115200
Address
0x0000 ☑ Clear code buffer
0x0000 ☑ Clear EEPROM buffer

☑ Program OR
☑ Trim IRC
Input IRC frequency 24.000 MHz
☑ Oscillator high gain
Set user EEPROM size 0.5 K
☑ Enable longer power-on-reset latency
☑ RESET pin used as I/O port
☑ Enable Low-Voltage reset
LVD detect leve 2.00 V
□ Hardware enable WDT after power-on-reset
Watch-Dog-Timer prescal 256

STC8/STC15 series ICE settings
MCU Type STC8H8K64U
● Simulate via COM P3.0/P3.1
○ Simulate via USB

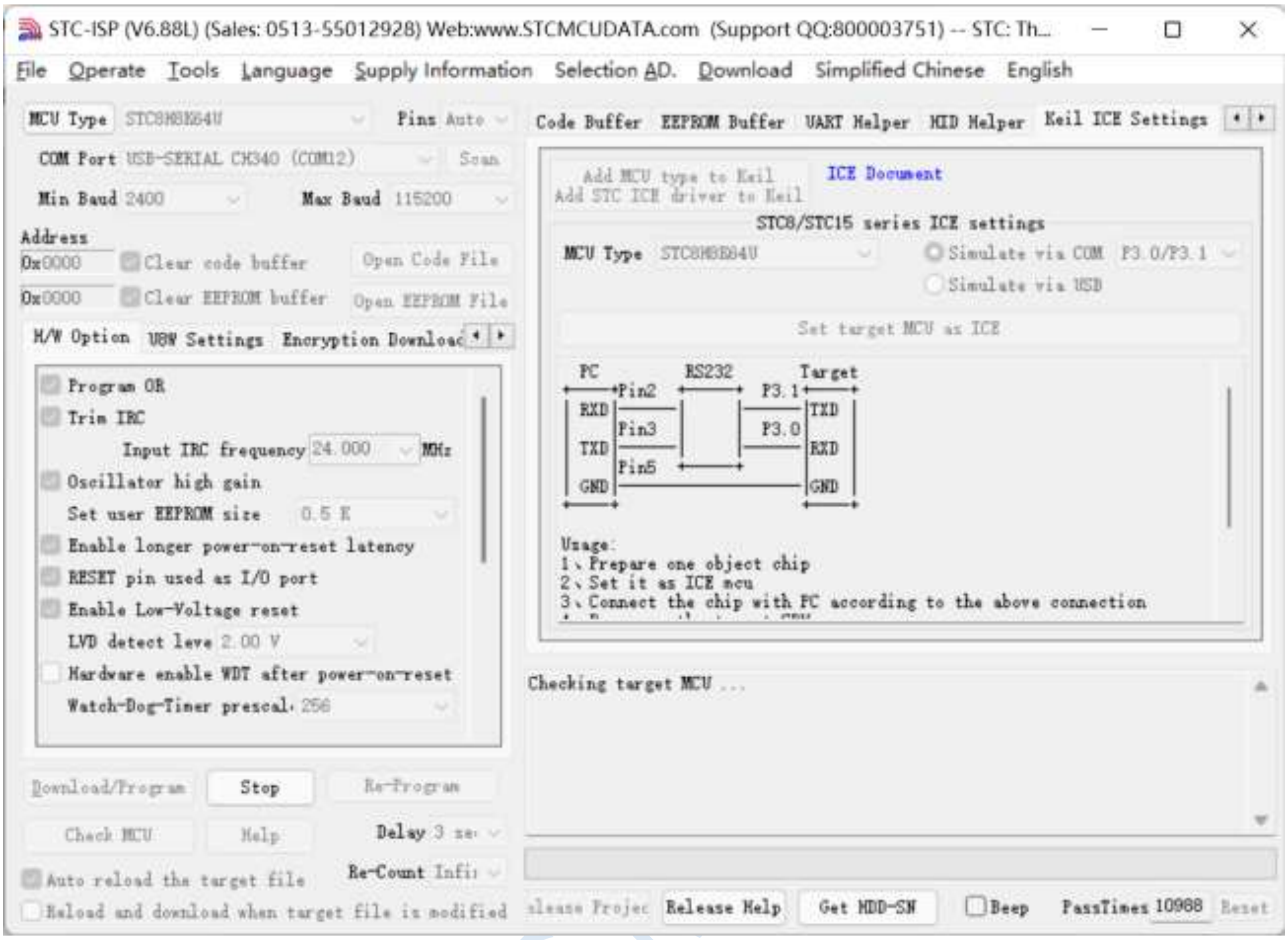
Usage:
1、Prepare one object chip
2、Set it as ICE mcu
3、Connect the chip with PC according to the above connection

Set frequency: 11.059MHz
Adjusted frequency: 11.090MHz
Trim error: 0.282%

Complete !(2022-02-22 17:45:23)

Delay 3 sec
☑ Auto reload the target file
Re-Count Infi
□ Reload and download when target file is modified
D:\Demo\Demo.hex
□ Beep
PassTimes 10988

Then click the "Set target MCU as ICE" button on the "Keil ICE Settings" page in the right functional area of the software,After pressing, the following screen will appear:



STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...

MCU Type STC8H8K64U Pins Auto

COM Port USB-SERIAL CH340 (COM12)

Min Baud 2400 Max Baud 115200

Address

0x0000 □ Clear code buffer

0x0000 □ Clear EEPROM buffer

□ Program OR

□ Trim IRC

Input IRC frequency 24.000 MHz

□ Oscillator high gain

Set user EEPROM size 0.5 K

□ Enable longer power-on-reset latency

□ RESET pin used as I/O port

□ Enable Low-Voltage reset

LVD detect leve 2.00 V

□ Hardware enable WDT after power-on-reset

Watch-Dog-Timer prescal 256

Delay 3 sec

□ Auto reload the target file

Re-Count Infinite

□ Reload and download when target file is modified

STC8/STC15 series ICE settings

MCU Type STC8H8K64U ○ Simulate via COM P3.0/P3.1

○ Simulate via USB

Usage:

1、Prepare one object chip

2、Set it as ICE mcu

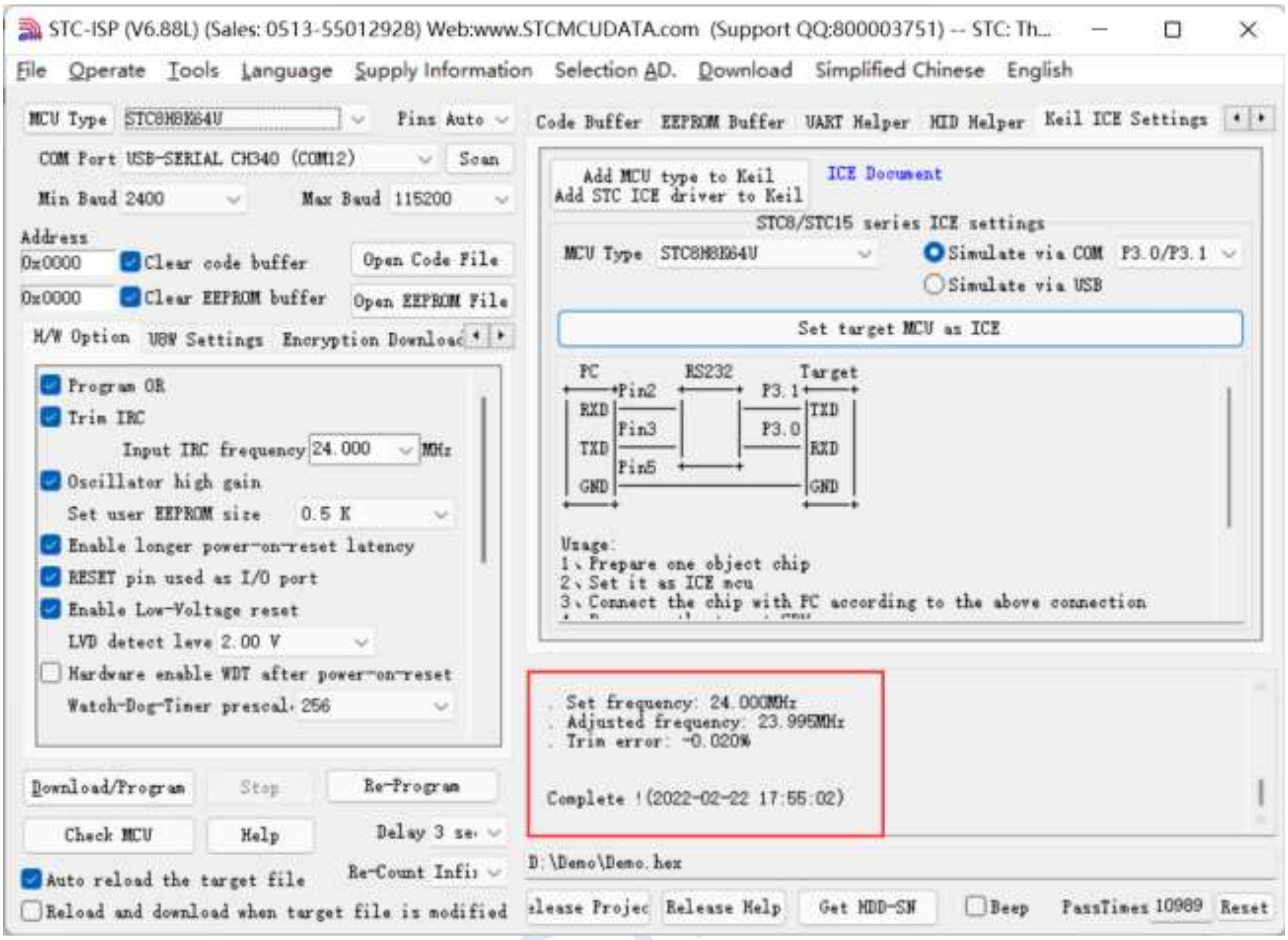
3、Connect the chip with PC according to the above connection

Checking target MCU ...

□ Beep PassTimes 10988

Next, you need to press the "power switch" on the STC Universal USB to Serial Tool to supply power to the MCU [cold start], and you can start to make the simulation chip.

If the setting is successful, the following screen will appear:

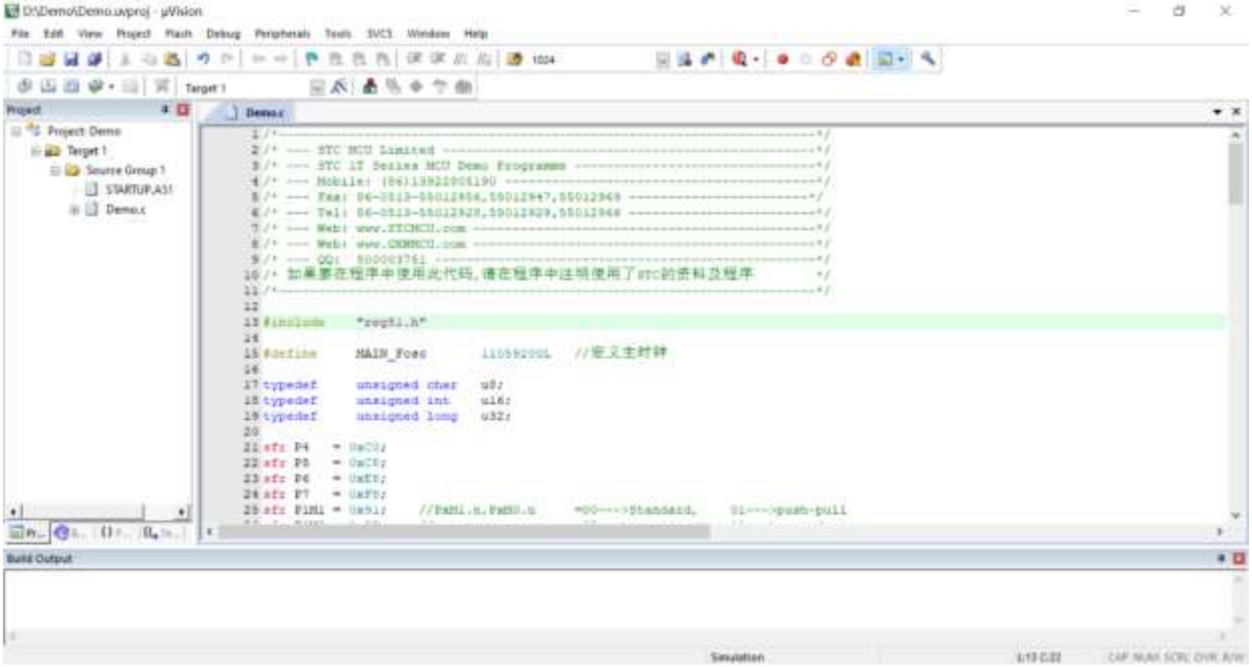

STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...
File Operate Tools Language Supply Information Selection AD. Download Simplified Chinese English
MCU Type STC8H8K64U
Pins Auto
COM Port USB-SERIAL CH340 (COM12)
Scan
Min Baud 2400
Max Baud 115200
Address
0x0000 ☑ Clear code buffer
Open Code File
0x0000 ☑ Clear EEPROM buffer
Open EEPROM File
H/W Option U8W Settings Encryption Download
☑ Program OR
☑ Trim IRC
Input IRC frequency 24.000 MHz
☑ Oscillator high gain
Set user EEPROM size 0.5 K
☑ Enable longer power-on-reset latency
☑ RESET pin used as I/O port
☑ Enable Low-Voltage reset
LVD detect leve 2.00 V
□ Hardware enable WDT after power-on-reset
Watch-Dog-Timer prescal. 256
Download/Program
Stop
Re-Program
Check MCU
Help
Delay 3 sec
☑ Auto reload the target file
Re-Count Infi
□ Reload and download when target file is modified
Code Buffer EEPROM Buffer UART Helper HID Helper Keil ICE Settings
Add MCU type to Keil
Add STC ICE driver to Keil
ICE Document
STC8/STC15 series ICE settings
MCU Type STC8H8K64U
● Simulate via COM P3.0/P3.1
○ Simulate via USB
Set target MCU as ICE
PC
RS232
Target
Pin2
P3.1
RXD
Pin3
P3.0
TXD
TXD
RXD
Pin5
GND
GND
Usage:
1、Prepare one object chip
2、Set it as ICE mcu
3、Connect the chip with PC according to the above connection
Release Project
Release Help
Get HDD-SN
□ Beep
PassTimes 10989
Reset

```
Set frequency: 24.000MHz
Adjusted frequency: 23.995MHz
Trim error: -0.020%

Complete !(2022-02-22 17:55:02)
```

D:\Demo\Demo.hex

At this point, the simulation chip has been made successfully.

Next we open a project for simulation:

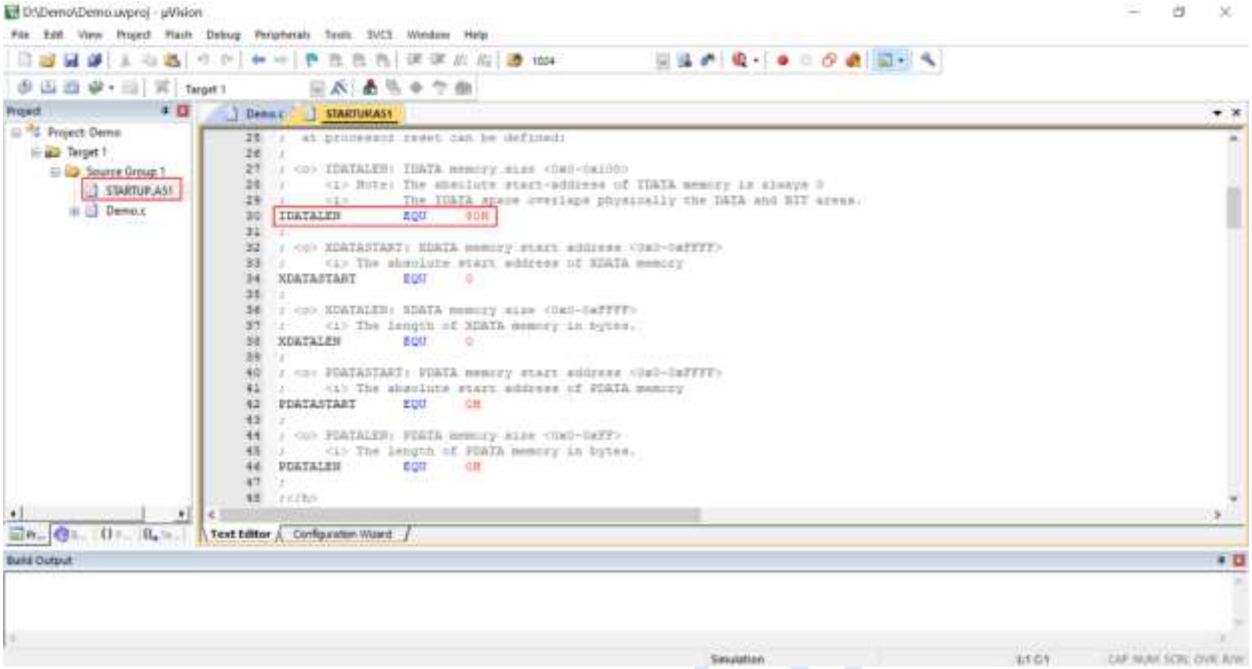


```
/*---------------------------------------------------------------------*/
/* --- STC MCU Limited ------------------------------------------------*/
/* --- STC 1T Series MCU Demo Programme -------------------------------*/
/* --- Mobile: (86)13922805190 ----------------------------------------*/
/* --- Fax: 86-0513-55012956,55012947,55012969 ------------------------*/
/* --- Tel: 86-0513-55012928,55012929,55012966 ------------------------*/
/* --- Web: www.STCMCU.com --------------------------------------------*/
/* --- Web: www.GXWMCU.com --------------------------------------------*/
/* --- QQ:  800003751 -------------------------------------------------*/
/* 如果要在程序中使用此代码,请在程序中注明使用了STC的资料及程序        */
/*---------------------------------------------------------------------*/

#include    "reg51.h"

#define     MAIN_Fosc       11059200L   //定义主时钟

typedef     unsigned char   u8;
typedef     unsigned int    u16;
typedef     unsigned long   u32;

sfr P4   = 0xC0;
sfr P5   = 0xC8;
sfr P6   = 0xE8;
sfr P7   = 0xF8;
sfr P1M1 = 0x91;    //PxM1.n,PxM0.n     =00--->Standard,    01--->push-pull
```

Then make the following project settings:

An additional note:

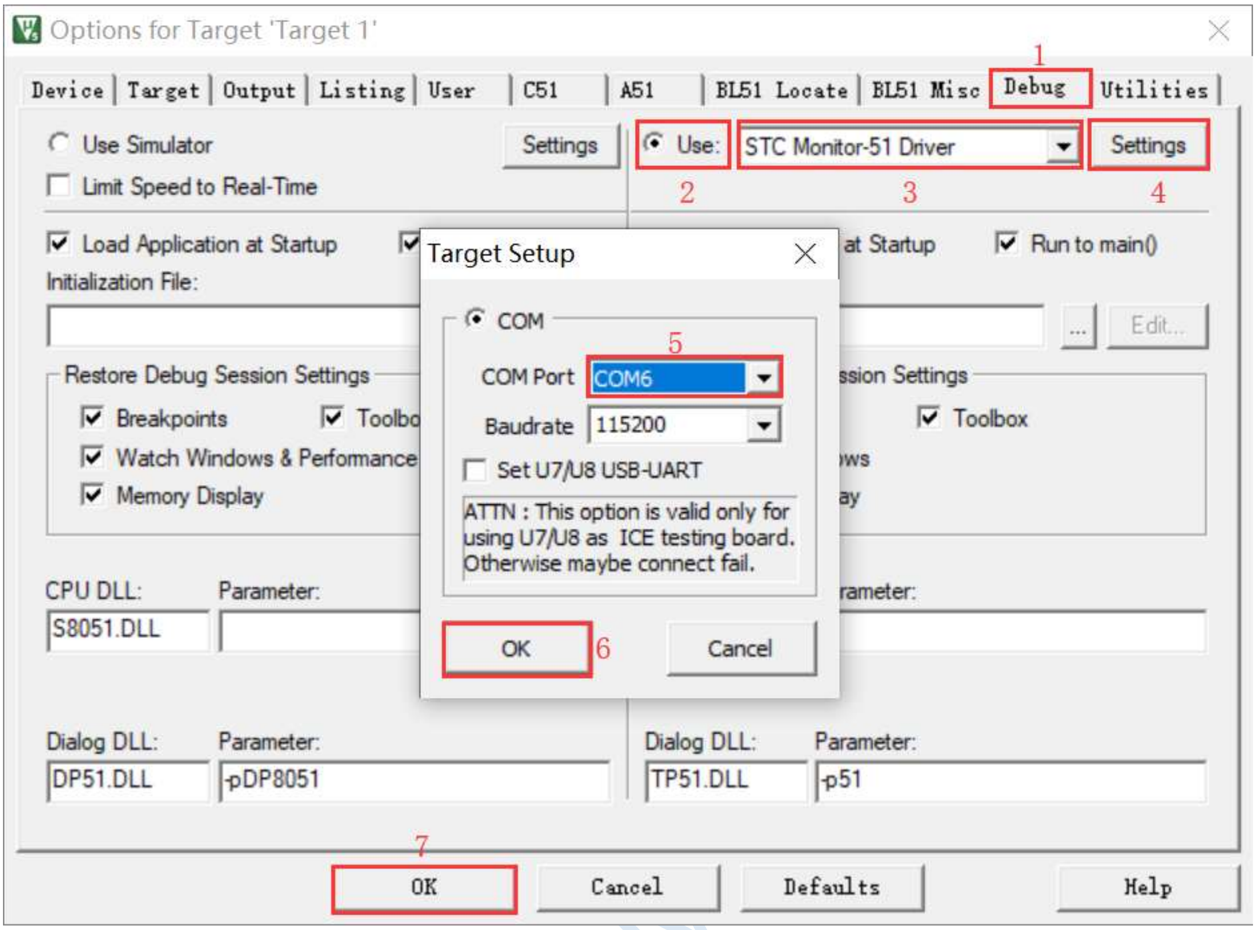
When a C language project is created and the startup file "STARTUP.A51" is added to the project, there is a macro definition named "IDATALEN", which is used to define the size of IDATA. The default value is 128, which is 80H in hexadecimal, and it is also the size of IDATA that needs to be initialized to 0 in the startup file. So when IDATA is defined as 80H, the code in STARTUP.A51 will initialize the RAM of 00-7F of IDATA to 0; similarly, if IDATA is defined as 0FFH, it will initialize the RAM of 00-FF of IDATA to 0.



```
;   at processor reset can be defined:
;
; <o> IDATALEN: IDATA memory size <0x0-0x100>
;     <i> Note: The absolute start-address of IDATA memory is always 0
;     <i>       The IDATA space overlaps physically the DATA and BIT areas.
IDATALEN        EQU     80H
;
; <o> XDATASTART: XDATA memory start address <0x0-0xFFFF>
;     <i> The absolute start address of XDATA memory
XDATASTART      EQU     0
;
; <o> XDATALEN: XDATA memory size <0x0-0xFFFF>
;     <i> The length of XDATA memory in bytes.
XDATALEN        EQU     0
;
; <o> PDATASTART: PDATA memory start address <0x0-0xFFFF>
;     <i> The absolute start address of PDATA memory
PDATASTART      EQU     0H
;
; <o> PDATALEN: PDATA memory size <0x0-0xFF>
;     <i> The length of PDATA memory in bytes.
PDATALEN        EQU     0H
;
;</h>
```

The IDATA size of the STC8H series microcontroller we selected is 256 bytes (00-7F DATA and 80H-FFH IDATA), but because the last 17 bytes of RAM have written ID numbers and related test parameters, If users need to use this part of data in the program, they must not define IDATALEN as 256.

Press the shortcut key "Alt+F7" or select "Option for Target ‘Target1’" in the menu "Project" to configure the project in the "Option for Target ‘Target1’" dialog box:
Step 1. Enter the project setting page and select the "Debug" setting page;
Step 2. Select the hardware emulation "Use …" on the right;
Step 3. Select "STC Monitor-51 Driver" item in the simulation driver drop-down list;
Step 4. Click the "Settings" button to enter the serial port settings screen;
Step 5: Set the port number and baud rate of the serial port. The serial port number should be the serial port corresponding to the STC universal USB to serial port tool. The baud rate is generally 115200 or 57600.
Make sure to complete the simulation settings.
The detailed steps are shown in the figure below:



Options for Target 'Target 1'

○ Use Simulator
☐ Limit Speed to Real-Time
☑ Load Application at Startup
Initialization File:

Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watch Windows & Performance
☑ Memory Display

CPU DLL: S8051.DLL Parameter:
Dialog DLL: DP51.DLL Parameter: -pDP8051

● Use: STC Monitor-51 Driver
☑ Load Application at Startup ☑ Run to main()
Restore Debug Session Settings
☑ Toolbox
Dialog DLL: TP51.DLL Parameter: -p51

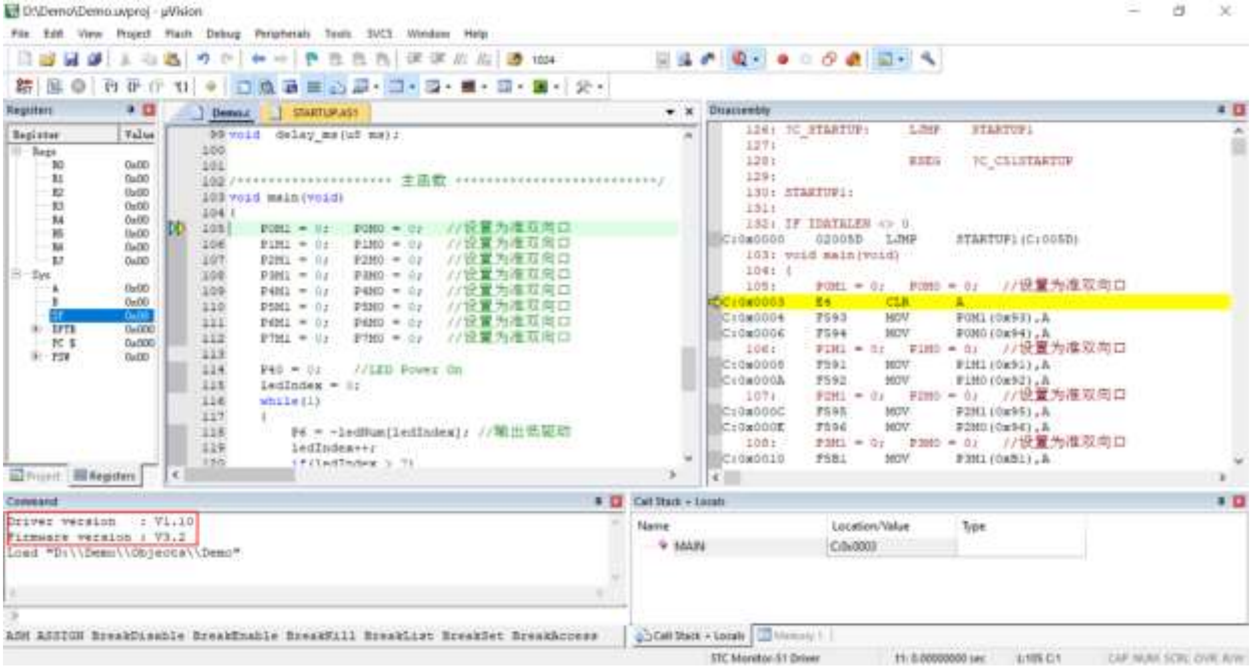
Target Setup

● COM
COM Port: COM6
Baudrate: 115200
☐ Set U7/U8 USB-UART

ATTN : This option is valid only for using U7/U8 as ICE testing board. Otherwise maybe connect fail.

After finishing all the above work, you can press "Ctrl+F5" in Keil software to start simulation debugging.

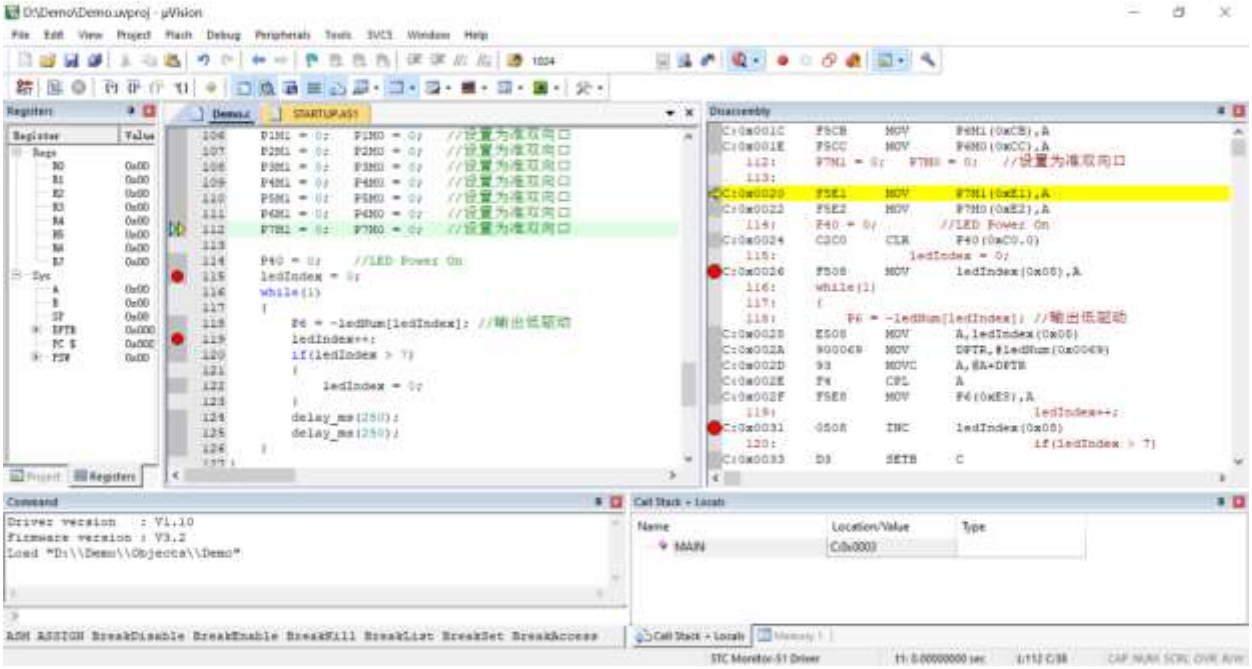
If the hardware connection is correct, you will enter a debugging interface similar to the following, and display the current simulation driver version number and the current simulation monitoring code firmware version number in the command output window, as shown in the following figure:



```
void  delay_ms(u8 ms);


/********************* 主函数 *************************/
void main(void)
{
    P0M1 = 0;   P0M0 = 0;   //设置为准双向口
    P1M1 = 0;   P1M0 = 0;   //设置为准双向口
    P2M1 = 0;   P2M0 = 0;   //设置为准双向口
    P3M1 = 0;   P3M0 = 0;   //设置为准双向口
    P4M1 = 0;   P4M0 = 0;   //设置为准双向口
    P5M1 = 0;   P5M0 = 0;   //设置为准双向口
    P6M1 = 0;   P6M0 = 0;   //设置为准双向口
    P7M1 = 0;   P7M0 = 0;   //设置为准双向口

    P40 = 0;    //LED Power On
    ledIndex = 0;
    while(1)
    {
        P6 = ~ledNum[ledIndex]; //输出低驱动
        ledIndex++;
        if(ledIndex > 7)
```

```
Driver version   : V1.10
Firmware version : V3.2
Load "D:\\Demo\\Objects\\Demo"
```

During the simulation debugging process, you can perform multiple operations such as resetting, running at full speed, single stepping, and setting breakpoints.



```
        P1M1 = 0;   P1M0 = 0;   //设置为准双向口
        P2M1 = 0;   P2M0 = 0;   //设置为准双向口
        P3M1 = 0;   P3M0 = 0;   //设置为准双向口
        P4M1 = 0;   P4M0 = 0;   //设置为准双向口
        P5M1 = 0;   P5M0 = 0;   //设置为准双向口
        P6M1 = 0;   P6M0 = 0;   //设置为准双向口
        P7M1 = 0;   P7M0 = 0;   //设置为准双向口

        P40 = 0;        //LED Power On
        ledIndex = 0;
        while(1)
        {
            P6 = ~ledNum[ledIndex];  //输出低驱动
            ledIndex++;
            if(ledIndex > 7)
            {
                ledIndex = 0;
            }
            delay_ms(250);
            delay_ms(250);
        }
}
```

```
C:0x001C    F5CB     MOV      P6M1(0xCB),A
C:0x001E    F5CC     MOV      P6M0(0xCC),A
   112:     P7M1 = 0;   P7M0 = 0;   //设置为准双向口
   113:
C:0x0020    F5E1     MOV      P7M1(0xE1),A
C:0x0022    F5E2     MOV      P7M0(0xE2),A
   114:     P40 = 0;          //LED Power On
C:0x0024    C2C0     CLR      P40(0xC0.0)
   115:              ledIndex = 0;
C:0x0026    F508     MOV      ledIndex(0x08),A
   116:     while(1)
   117:     {
   118:         P6 = ~ledNum[ledIndex]; //输出低驱动
C:0x0028    E508     MOV      A,ledIndex(0x08)
C:0x002A    900069   MOV      DPTR,#ledNum(0x0069)
C:0x002D    93       MOVC     A,@A+DPTR
C:0x002E    F4       CPL      A
C:0x002F    F5E8     MOV      P6(0xE8),A
   119:                                 ledIndex++;
C:0x0031    0508     INC      ledIndex(0x08)
   120:                                 if(ledIndex > 7)
C:0x0033    D3       SETB     C
```

```
Driver version   : V1.10
Firmware version : V3.2
Load "D:\\Demo\\Objects\\Demo"
```

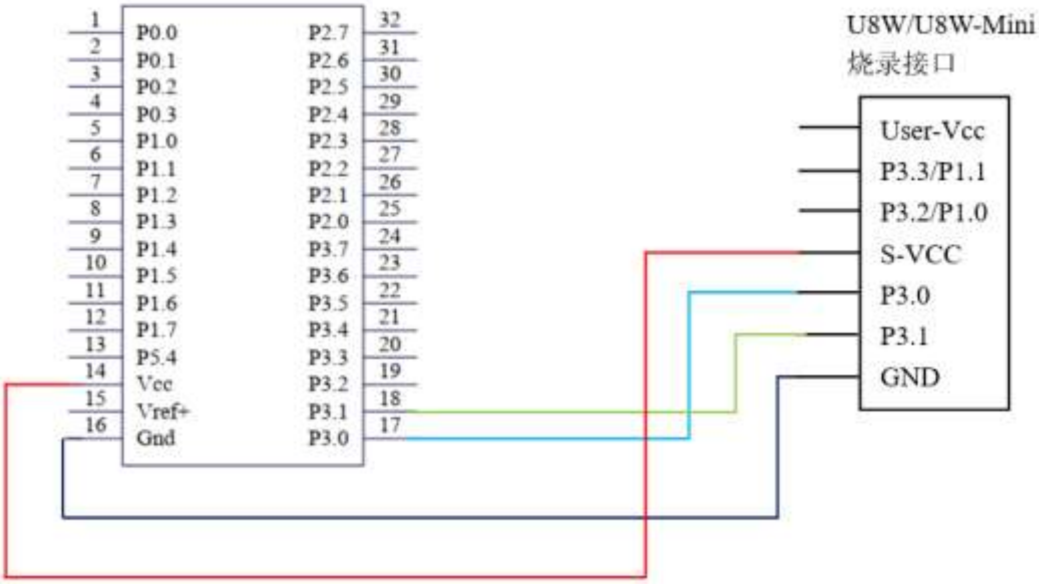
```
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess
```

Name	Location/Value	Type
MAIN	C:0x0003	

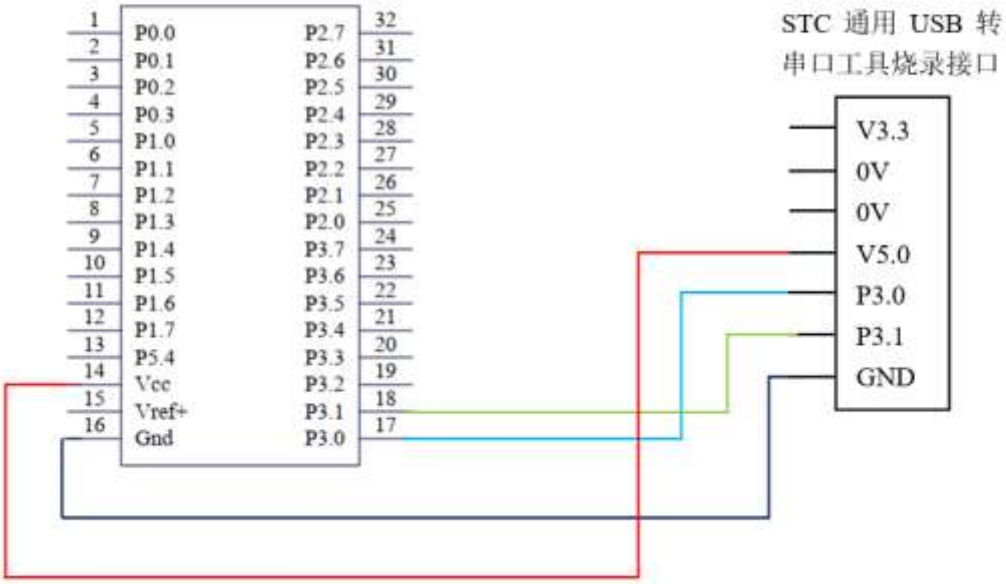
As shown in the figure above, multiple breakpoints can be set in the program, and the maximum number of breakpoint settings currently allowed is 20 (in theory, any number can be set, but setting too many breakpoints will affect the speed of debugging).

H.5 Application circuit diagram

H.5.1 U8W tool application reference circuit diagram



H.5.2 STC Universal USB to Serial Tool Application Reference Circuit Diagram



Appendix I STC Emulation Instruction Manual

I.1 Overview

All STC8G/8H series microcontrollers support online emulation, including downloading user code, chip reset, full-speed operation, single-step operation, setting breakpoints (the number of theoretical breakpoints is unlimited, but in order to improve the emulation efficiency, the current limit is up to 20 breakpoints), viewing variables, etc. The emulation operation is convenient for users to debug the code and find logical errors in the code, thereby shortening the project development cycle.

The emulation interface can be USB or serial port, the microcontroller itself is an emulator, and all emulation functions can be realized without an additional emulator. The corresponding USB port or serial port is originally a dedicated port for emulation, but when the emulation function is turned off, the user can freely use the emulation interface as GPIO, USB or serial port.

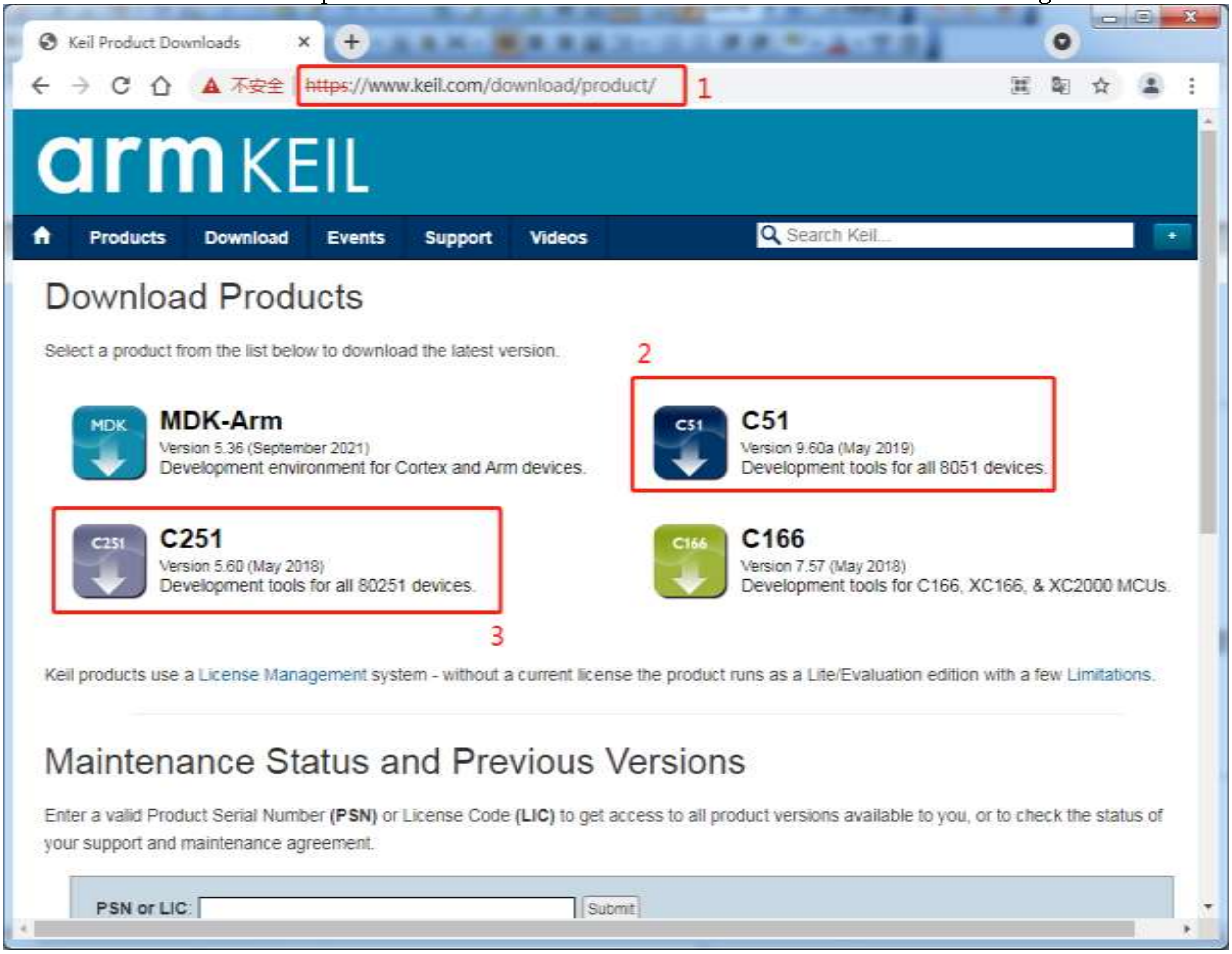
At present, the emulation mode of all MCUs is the software monitoring emulation mode, which will occupy part of the system resources. The resources occupied by each series of MCU simulation are shown in the following table (**NO Flash memory is occupied**):

MCU Series	Emulation Port	Resource occupied	
		Port	Data memory (XDATA)
STC8H8K64U family-B version	USB	D+，D-	768 bytes (1D00H-1FFFH)
	UART	P3.0，P3.1	768 bytes (1D00H-1FFFH)
		P3.6，P3.7	
		P1.6，P1.7	
		P4.3，P4.4	
STC8H8K64U family-A version	UART	P3.0，P3.1	768 bytes (1D00H-1FFFH)
		P3.6，P3.7	
		P1.6，P1.7	
		P4.3，P4.4	
STC8H4K64T family	UART	P3.0，P3.1	768 bytes (0D00H-0FFFH)
STC8H3K64S4 family	UART	P3.0，P3.1	768 bytes (0900H-0BFFH)
STC8H1K16 family	UART	P3.0，P3.1	768 bytes (0100H-03FFH)
STC8H1K08 family	UART	P3.0，P3.1	768 bytes (0100H-03FFH)
STC8G2K64S4 family	UART	P3.0，P3.1	768 bytes (0500H-07FFH)
STC8G1K08 family	UART	P3.0，P3.1	768 bytes (0100H-03FFH)
STC8C2K64S4 family	UART	P3.0，P3.1	768 bytes (0500H-07FFH)
STC8A8K64D4 family	UART	P3.0，P3.1	768 bytes (1D00H-1FFFH)
STC8A8K64S4A12 family	UART	P3.0，P3.1	768 bytes (1D00H-1FFFH)
STC8F2K64S4 family	UART	P3.0，P3.1	768 bytes (0500H-07FFH)
STC8F1K08S2 family	UART	P3.0，P3.1	768 bytes (0100H-03FFH)
IAP15W4K58S4	UART	P3.0，P3.1	768 bytes (0D00H-0FFFH)
IAP15F2K61S2	UART	P3.0，P3.1	768 bytes (0500H-07FFH)

I.2 Install Keil software

The emulation of STC microcontroller is based on the Keil development environment, so before the emulation, the Keil software must be installed.

The C51 and C251 development kits can be downloaded from the addresses shown in the figure below



Download Products

Select a product from the list below to download the latest version.

MDK-Arm
Version 5.36 (September 2021)
Development environment for Cortex and Arm devices.

C51
Version 9.60a (May 2019)
Development tools for all 8051 devices.

C251
Version 5.60 (May 2018)
Development tools for all 80251 devices.

C166
Version 7.57 (May 2018)
Development tools for C166, XC166, & XC2000 MCUs.

Keil products use a License Management system - without a current license the product runs as a Lite/Evaluation edition with a few Limitations.

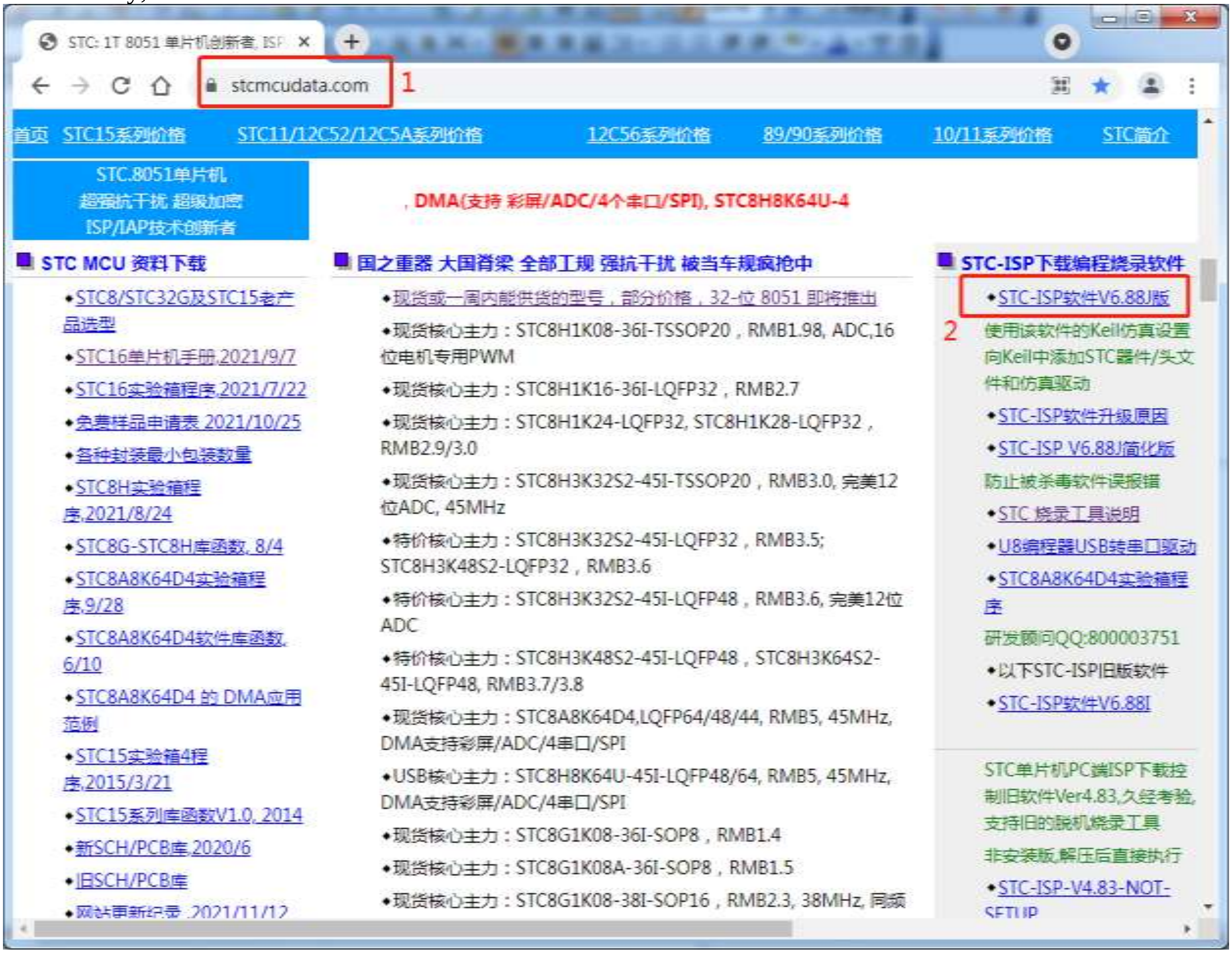
Maintenance Status and Previous Versions

Enter a valid Product Serial Number (**PSN**) or License Code (**LIC**) to get access to all product versions available to you, or to check the status of your support and maintenance agreement.

Note: The latest Keil-UV5 software does not include 8051 and 80251 toolkits by default, and must be downloaded and installed manually.

I.3 Install the emulation driver

After the Keil development environment is installed, you also need to install the STC-specific emulation driver. Proceed as follows:
Firstly, download the latest STC-ISP download software from the STC official website


STC: 1T 8051 单片机创新者, ISP
stcmcudata.com
1
2
首页 STC15系列价格 STC11/12C52/12C5A系列价格 12C56系列价格 89/90系列价格 10/11系列价格 STC简介

STC.8051单片机
超强抗干扰 超级加密
ISP/IAP技术创新者

，DMA(支持 彩屏/ADC/4个串口/SPI), STC8H8K64U-4

STC MCU 资料下载

- STC8/STC32G及STC15老产品选型
- STC16单片机手册,2021/9/7
- STC16实验箱程序,2021/7/22
- 免费样品申请表 2021/10/25
- 各种封装最小包装数量
- STC8H实验箱程序,2021/8/24
- STC8G-STC8H库函数, 8/4
- STC8A8K64D4实验箱程序,9/28
- STC8A8K64D4软件库函数, 6/10
- STC8A8K64D4 的 DMA应用范例
- STC15实验箱4程序,2015/3/21
- STC15系列库函数V1.0, 2014
- 新SCH/PCB库,2020/6
- 旧SCH/PCB库
- 网站更新纪录 2021/11/12

国之重器 大国脊梁 全部工规 强抗干扰 被当车规疯抢中

- 现货或一周内能供货的型号，部分价格，32-位 8051 即将推出
- 现货核心主力：STC8H1K08-36I-TSSOP20，RMB1.98, ADC,16位电机专用PWM
- 现货核心主力：STC8H1K16-36I-LQFP32，RMB2.7
- 现货核心主力：STC8H1K24-LQFP32, STC8H1K28-LQFP32，RMB2.9/3.0
- 现货核心主力：STC8H3K32S2-45I-TSSOP20，RMB3.0, 完美12位ADC, 45MHz
- 特价核心主力：STC8H3K32S2-45I-LQFP32，RMB3.5; STC8H3K48S2-LQFP32，RMB3.6
- 特价核心主力：STC8H3K32S2-45I-LQFP48，RMB3.6, 完美12位ADC
- 特价核心主力：STC8H3K48S2-45I-LQFP48，STC8H3K64S2-45I-LQFP48, RMB3.7/3.8
- 现货核心主力：STC8A8K64D4,LQFP64/48/44, RMB5, 45MHz, DMA支持彩屏/ADC/4串口/SPI
- USB核心主力：STC8H8K64U-45I-LQFP48/64, RMB5, 45MHz, DMA支持彩屏/ADC/4串口/SPI
- 现货核心主力：STC8G1K08-36I-SOP8，RMB1.4
- 现货核心主力：STC8G1K08A-36I-SOP8，RMB1.5
- 现货核心主力：STC8G1K08-38I-SOP16，RMB2.3, 38MHz, 同频

STC-ISP下载编程烧录软件

- STC-ISP软件V6.88J版

使用该软件的Keil仿真设置向Keil中添加STC器件/头文件和仿真驱动

- STC-ISP软件升级原因
- STC-ISP V6.88J简化版

防止被杀毒软件误报错

- STC 烧录工具说明
- U8编程器USB转串口驱动
- STC8A8K64D4实验箱程序

研发顾问QQ:800003751

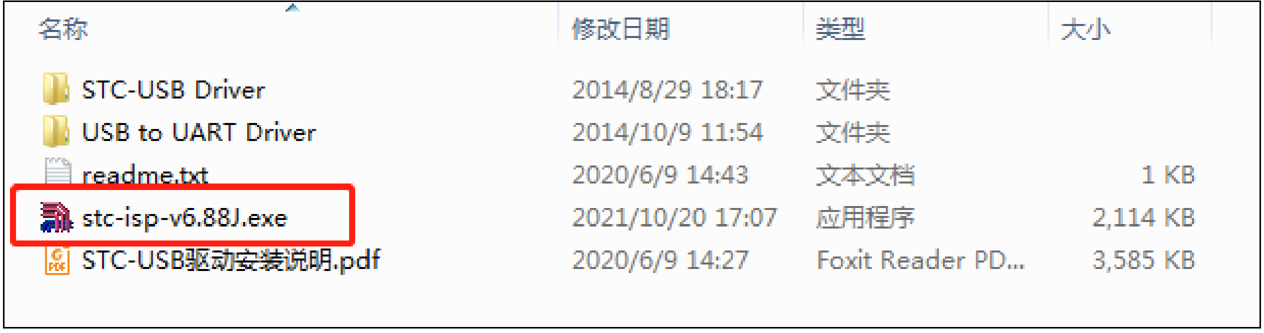
- 以下STC-ISP旧版软件
- STC-ISP软件V6.88I

STC单片机PC端ISP下载控制旧软件Ver4.83,久经考验,支持旧的脱机烧录工具

非安装版,解压后直接执行

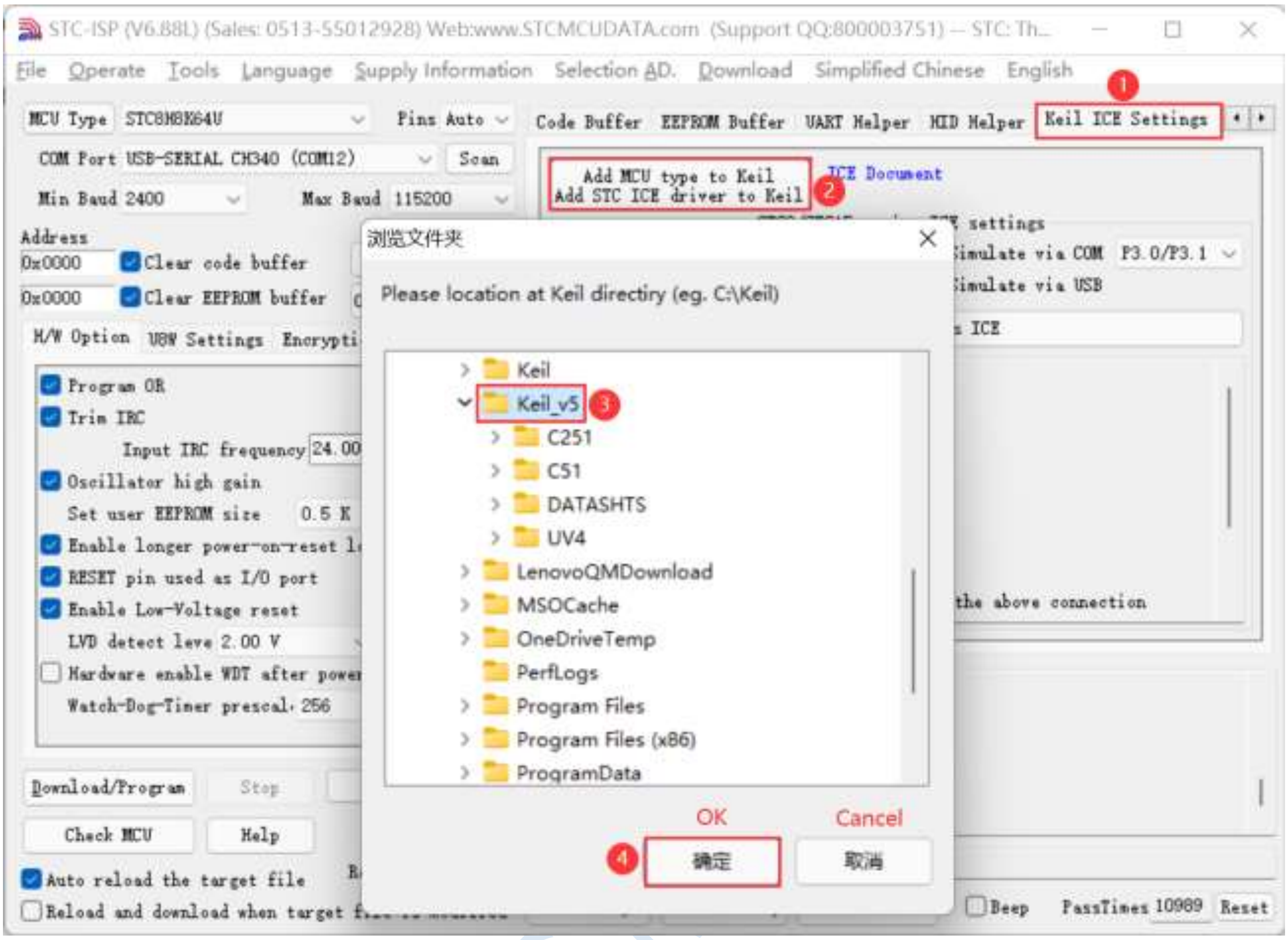
- STC-ISP-V4.83-NOT-SETUP

After downloading and unzipping, open the "stc-isp-vxx.exe" executable in the package.



名称	修改日期	类型	大小
STC-USB Driver	2014/8/29 18:17	文件夹	
USB to UART Driver	2014/10/9 11:54	文件夹	
readme.txt	2020/6/9 14:43	文本文档	1 KB
stc-isp-v6.88J.exe	2021/10/20 17:07	应用程序	2,114 KB
STC-USB驱动安装说明.pdf	2020/6/9 14:27	Foxit Reader PD...	3,585 KB

Click the "Add MCU Type to Keil..." button in the "Keil ICE Settings" page of the download software (Figure below)



MCU Type STC8H8K64U

Pins Auto

COM Port USB-SERIAL CH340 (COM12)

Min Baud 2400

Max Baud 115200

Address

0x0000 ☑ Clear code buffer

0x0000 ☑ Clear EEPROM buffer

☑ Program OR

☑ Trim IRC

Input IRC frequency 24.00

☑ Oscillator high gain

Set user EEPROM size 0.5 K

☑ Enable longer power-on-reset l

☑ RESET pin used as I/O port

☑ Enable Low-Voltage reset

LVD detect leve 2.00 V

☐ Hardware enable WDT after power

Watch-Dog-Timer prescal 256

☑ Auto reload the target file

☐ Reload and download when target f

Please location at Keil directiry (eg. C:\Keil)

- Keil
- Keil_v5
  - C251
  - C51
  - DATASHTS
  - UV4
- LenovoQMDownload
- MSOCache
- OneDriveTemp
- PerfLogs
- Program Files
- Program Files (x86)
- ProgramData

settings

Simulate via COM P3.0/P3.1

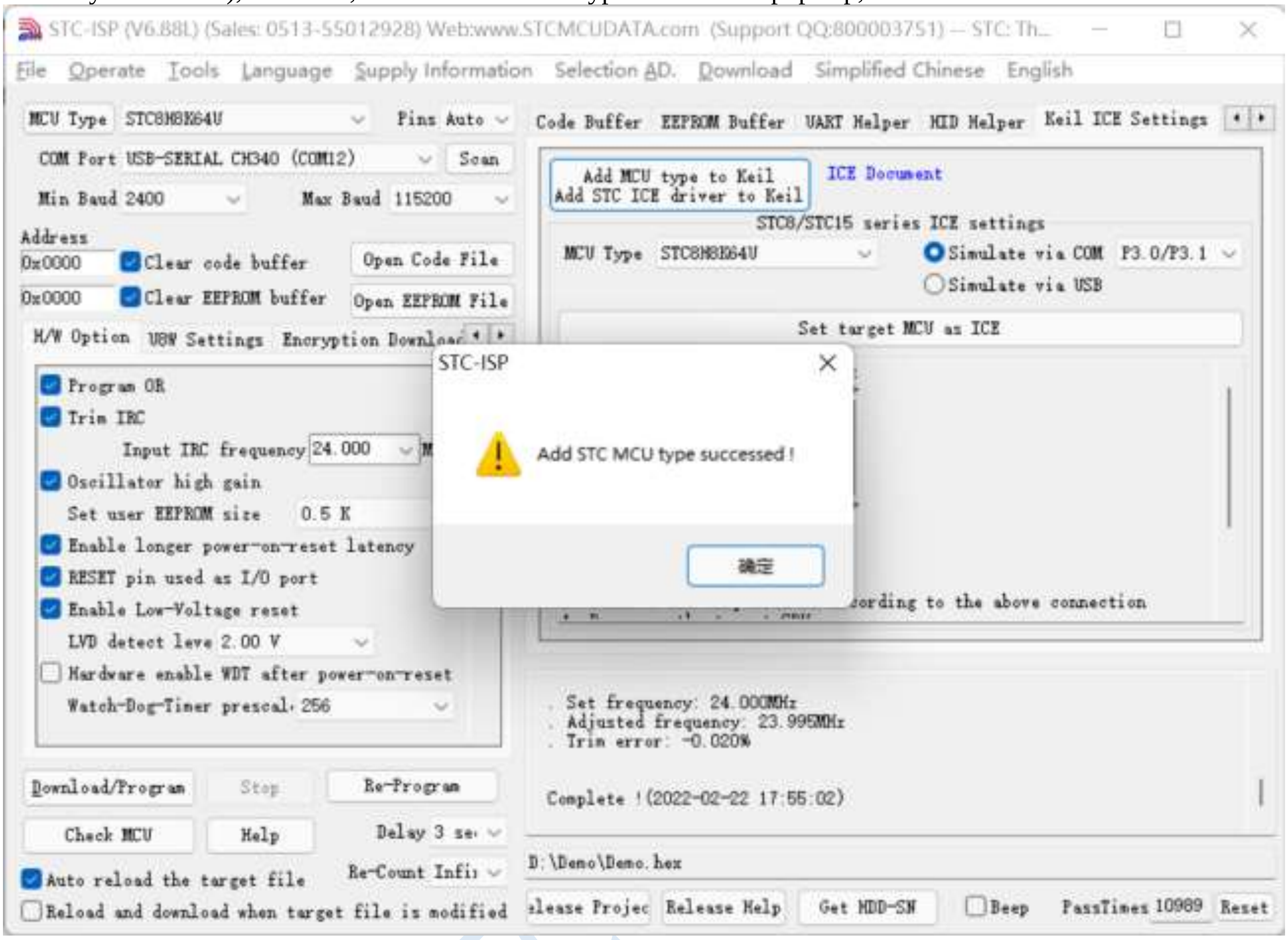
Simulate via USB

ICE

the above connection

☐ Beep PassTimes 10989

In the pop-up "Browse Folder" window, select the Keil installation directory (usually Keil's installation directory is "c:\keil"), click OK, if "Add STC MCU type successed!" pops up, it means the driver has been installed. .



STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...

MCU Type STC8H8K64U Pins Auto

COM Port USB-SERIAL CH340 (COM12)

Min Baud 2400 Max Baud 115200

Address 0x0000 ☑ Clear code buffer

0x0000 ☑ Clear EEPROM buffer

☑ Program OR

☑ Trim IRC

Input IRC frequency 24.000 M

☑ Oscillator high gain

Set user EEPROM size 0.5 K

☑ Enable longer power-on-reset latency

☑ RESET pin used as I/O port

☑ Enable Low-Voltage reset

LVD detect leve 2.00 V

□ Hardware enable WDT after power-on-reset

Watch-Dog-Timer prescal 256

Delay 3 se Re-Count Infi

☑ Auto reload the target file

□ Reload and download when target file is modified

STC8/STC15 series ICE settings

MCU Type STC8H8K64U ● Simulate via COM P3.0/P3.1 ○ Simulate via USB

STC-ISP

Add STC MCU type successed !

确定

...cording to the above connection

Set frequency: 24.000MHz
Adjusted frequency: 23.995MHz
Trim error: -0.020%

Complete !(2022-02-22 17:55:02)

D:\Demo\Demo.hex

□ Beep PassTimes 10989

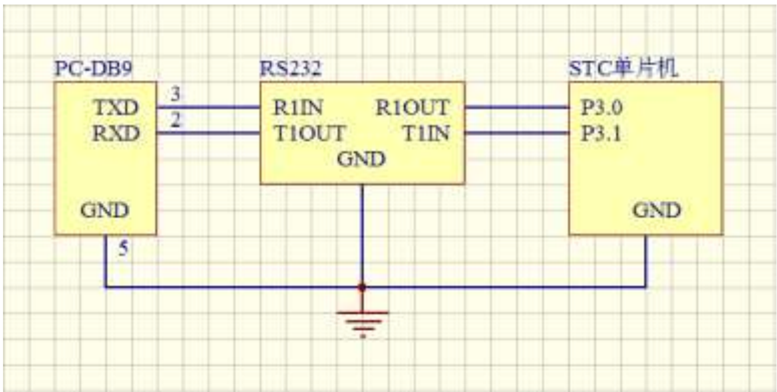
I.4 Serial port emulation directly

I.4.1 Make serial port emulation chip

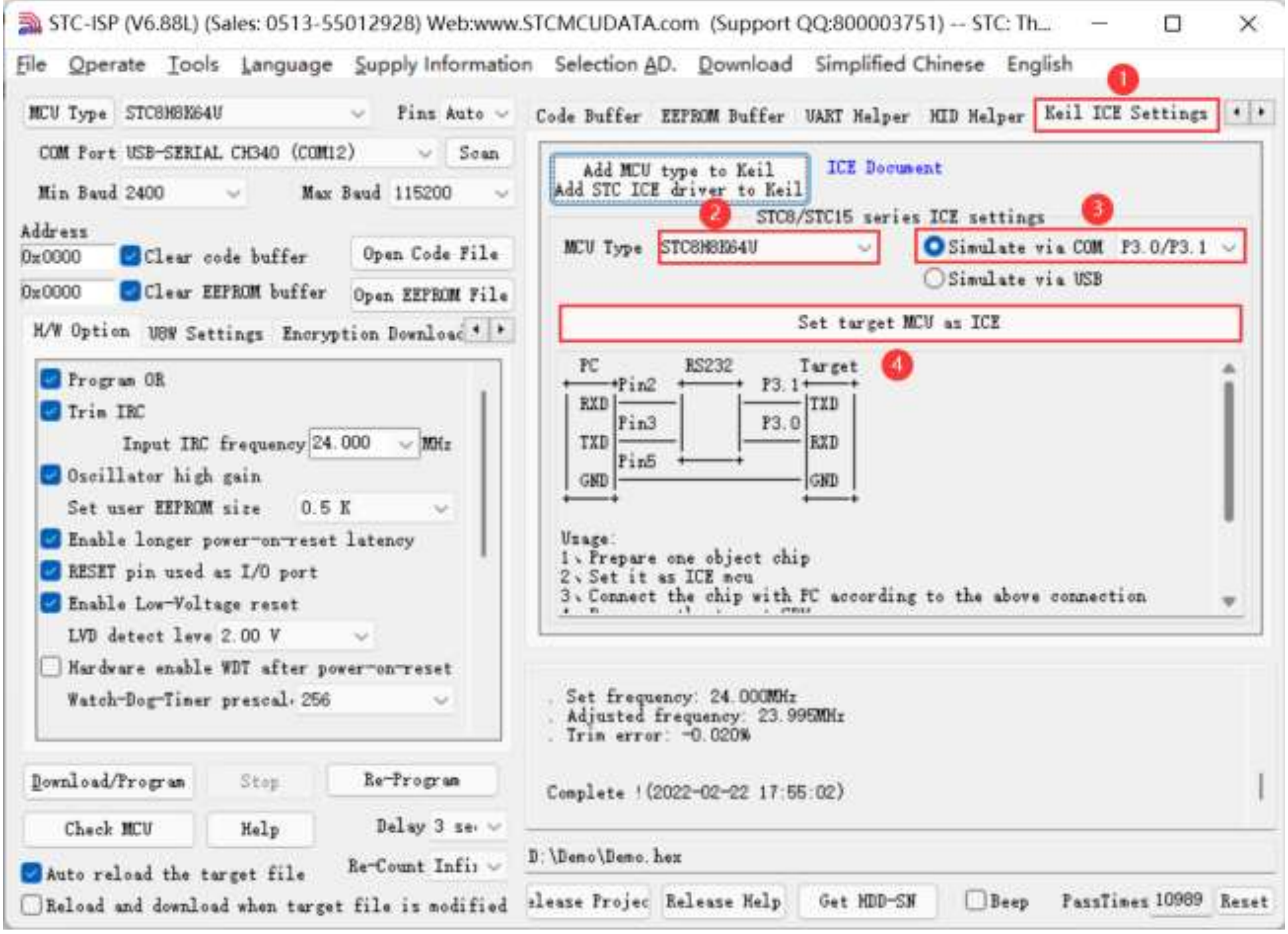
When the STC microcontroller is shipped, the emulation function is disabled by default. If you want to use the emulation function, you need to use the STC-ISP download software to set the target microcontroller as the emulation chip.

The setting steps are as follows:

Firstly, connect the target chip to the serial port of the computer as shown in the figure below, and power off the microcontroller



Open the STC-ISP download software and follow the steps shown in the figure below to set the emulation chip.



MCU Type STC8H8K64U Pins Auto

COM Port USB-SERIAL CH340 (COM12)

Min Baud 2400 Max Baud 115200

Address

0x0000 ☑ Clear code buffer

0x0000 ☑ Clear EEPROM buffer

☑ Program OR

☑ Trim IRC

Input IRC frequency 24.000 MHz

☑ Oscillator high gain

Set user EEPROM size 0.5 K

☑ Enable longer power-on-reset latency

☑ RESET pin used as I/O port

☑ Enable Low-Voltage reset

LVD detect leve 2.00 V

□ Hardware enable WDT after power-on-reset

Watch-Dog-Timer prescal 256

Delay 3 sec

Re-Count Infinite

☑ Auto reload the target file

□ Reload and download when target file is modified

STC8/STC15 series ICE settings

MCU Type STC8H8K64U

● Simulate via COM P3.0/P3.1

○ Simulate via USB

Usage:
1、Prepare one object chip
2、Set it as ICE mcu
3、Connect the chip with PC according to the above connection

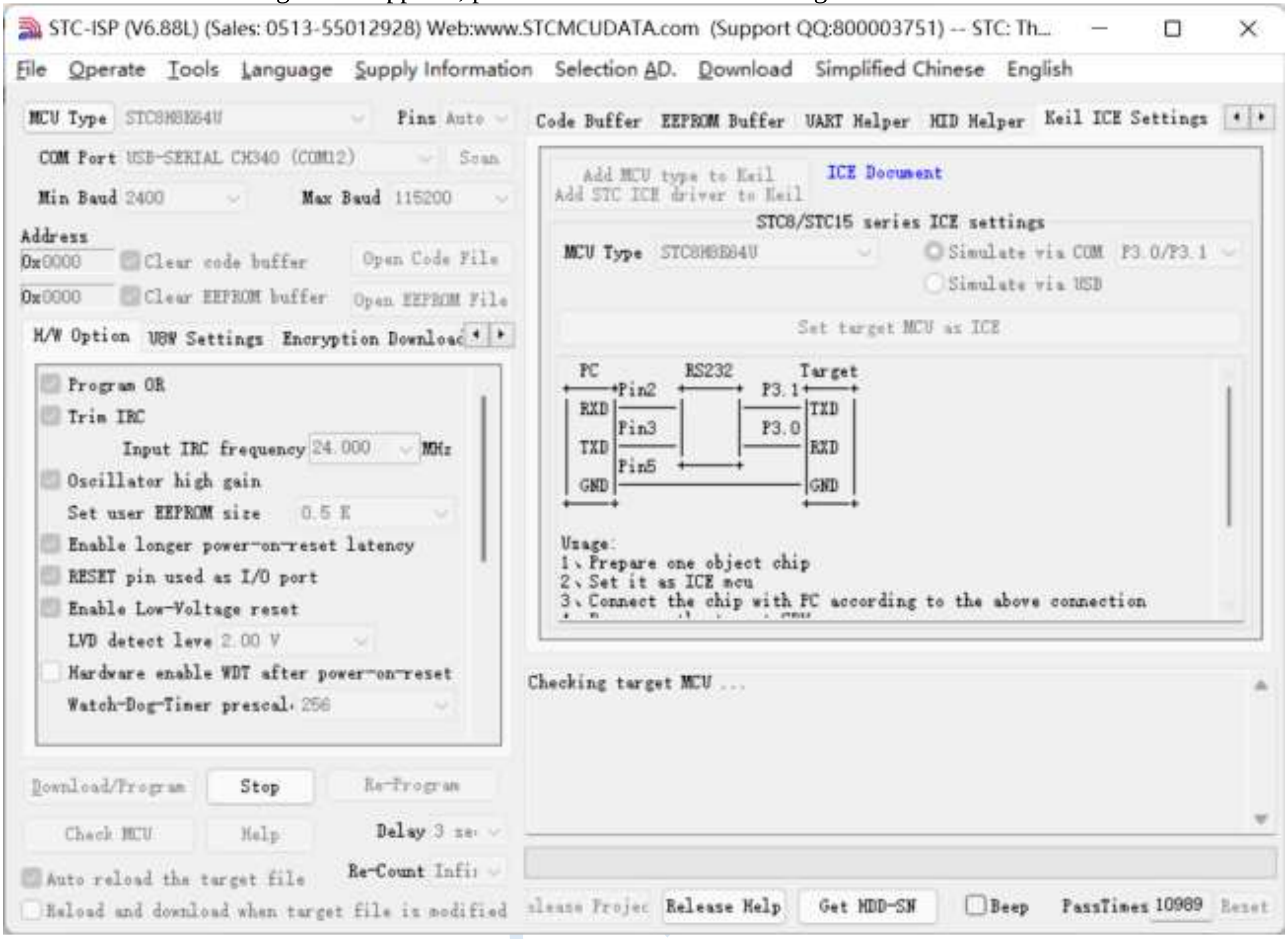
Set frequency: 24.000MHz
Adjusted frequency: 23.995MHz
Trim error: -0.020%

Complete !(2022-02-22 17:55:02)

D:\Demo\Demo.hex

□ Beep PassTimes 10989

When the following screen appears, power on the microcontroller again.



STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...

MCU Type STC8H8K64U Pins Auto

COM Port USB-SERIAL CH340 (COM12)

Min Baud 2400 Max Baud 115200

Address

0x0000 □ Clear code buffer

0x0000 □ Clear EEPROM buffer

H/W Option

- ☐ Program OR
- ☐ Trim IRC
- Input IRC frequency 24.000 MHz
- ☐ Oscillator high gain
- Set user EEPROM size 0.5 K
- ☐ Enable longer power-on-reset latency
- ☐ RESET pin used as I/O port
- ☐ Enable Low-Voltage reset
- LVD detect leve 2.00 V
- ☐ Hardware enable WDT after power-on-reset
- Watch-Dog-Timer prescal 256

Delay 3 se

Re-Count Infi

☐ Auto reload the target file

☐ Reload and download when target file is modified

Keil ICE Settings

STC8/STC15 series ICE settings

MCU Type STC8H8K64U ○ Simulate via COM P3.0/P3.1

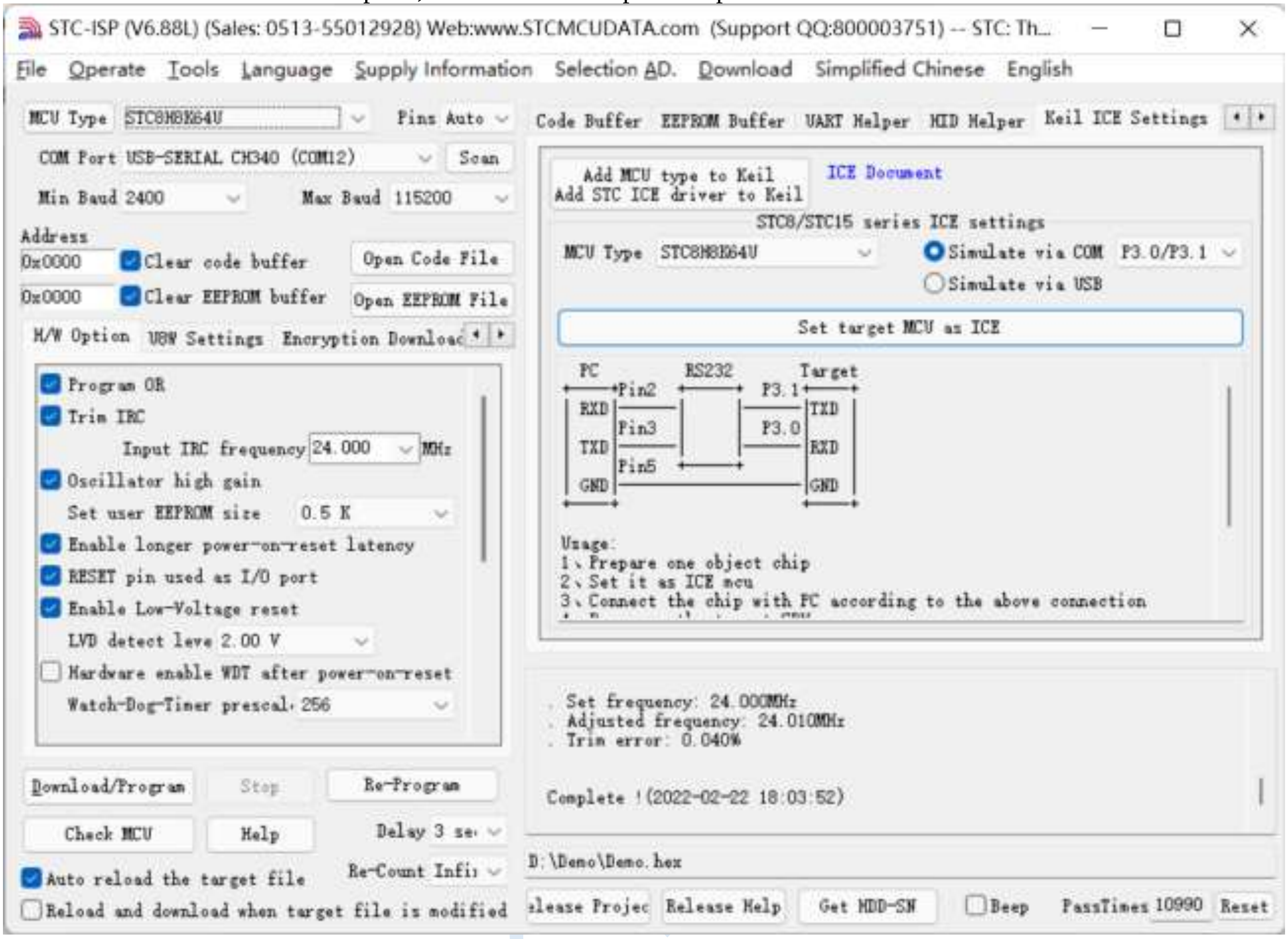
○ Simulate via USB

Usage:
1、Prepare one object chip
2、Set it as ICE mcu
3、Connect the chip with PC according to the above connection

Checking target MCU ...

☐ Beep PassTimes 10989

After the download is complete, the emulation chip is completed.


STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...
File Operate Tools Language Supply Information Selection AD. Download Simplified Chinese English
Scan
Open Code File
Open EEPROM File
H/W Option U8W Settings Encryption Download
Code Buffer EEPROM Buffer UART Helper HID Helper Keil ICE Settings
Add MCU type to Keil
Add STC ICE driver to Keil
ICE Document
Set target MCU as ICE
PC
RS232
Target
Pin2
Pin3
Pin5
RXD
TXD
GND
P3.1
P3.0
TXD
RXD
GND
Download/Program Stop Re-Program
Check MCU Help
elease Projec Release Help Get HID-SN Reset

MCU Type STC8H8K64U
Pins Auto
COM Port USB-SERIAL CH340 (COM12)
Min Baud 2400
Max Baud 115200

Address
0x0000 ☑ Clear code buffer
0x0000 ☑ Clear EEPROM buffer

☑ Program OR
☑ Trim IRC
Input IRC frequency 24.000 MHz
☑ Oscillator high gain
Set user EEPROM size 0.5 K
☑ Enable longer power-on-reset latency
☑ RESET pin used as I/O port
☑ Enable Low-Voltage reset
LVD detect leve 2.00 V
☐ Hardware enable WDT after power-on-reset
Watch-Dog-Timer prescal 256

Delay 3 se
Re-Count Infi
☑ Auto reload the target file
☐ Reload and download when target file is modified

STC8/STC15 series ICE settings
MCU Type STC8H8K64U
● Simulate via COM P3.0/P3.1
○ Simulate via USB

Usage:
1、Prepare one object chip
2、Set it as ICE mcu
3、Connect the chip with PC according to the above connection

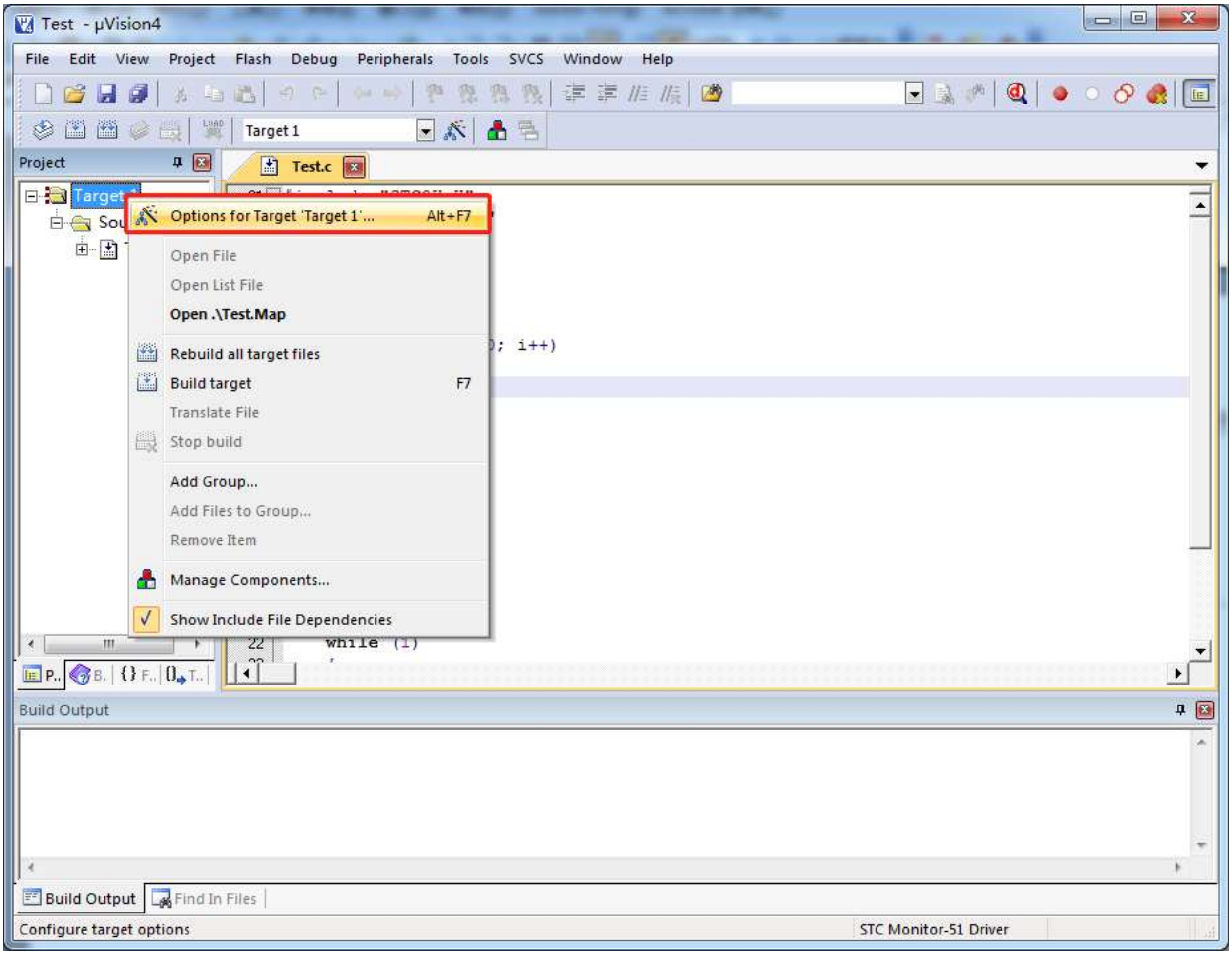
```
. Set frequency: 24.000MHz
. Adjusted frequency: 24.010MHz
. Trim error: 0.040%

Complete !(2022-02-22 18:03:52)
```

D:\Demo\Demo.hex

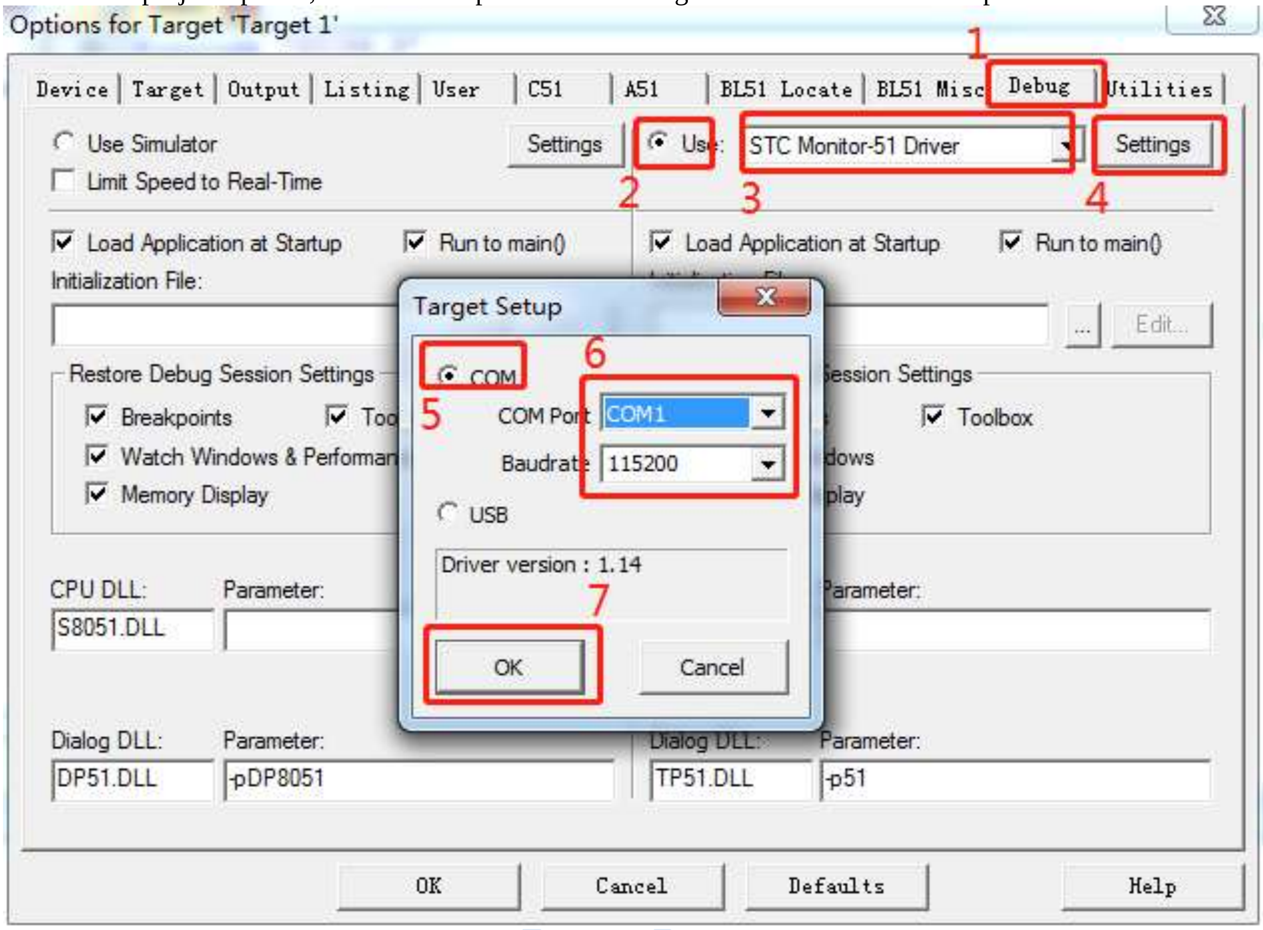
☐ Beep
PassTimes 10990

I.4.2 Serial port emulation settings in Keil software

Open the project file in Keil software, and click "Options for ..." in the right-click menu as shown in the figure below.



In the project options, follow the steps shown in the figure below to set the serial port emulation.


Options for Target 'Target 1'

Target Setup

○ Use Simulator

☐ Limit Speed to Real-Time

● Use: STC Monitor-51 Driver

☑ Load Application at Startup ☑ Run to main()

Initialization File:

Restore Debug Session Settings

☑ Breakpoints ☑ Toolbox

☑ Watch Windows & Performance

☑ Memory Display

● COM

COM Port: COM1

Baudrate: 115200

○ USB

Driver version : 1.14

CPU DLL: S8051.DLL Parameter:

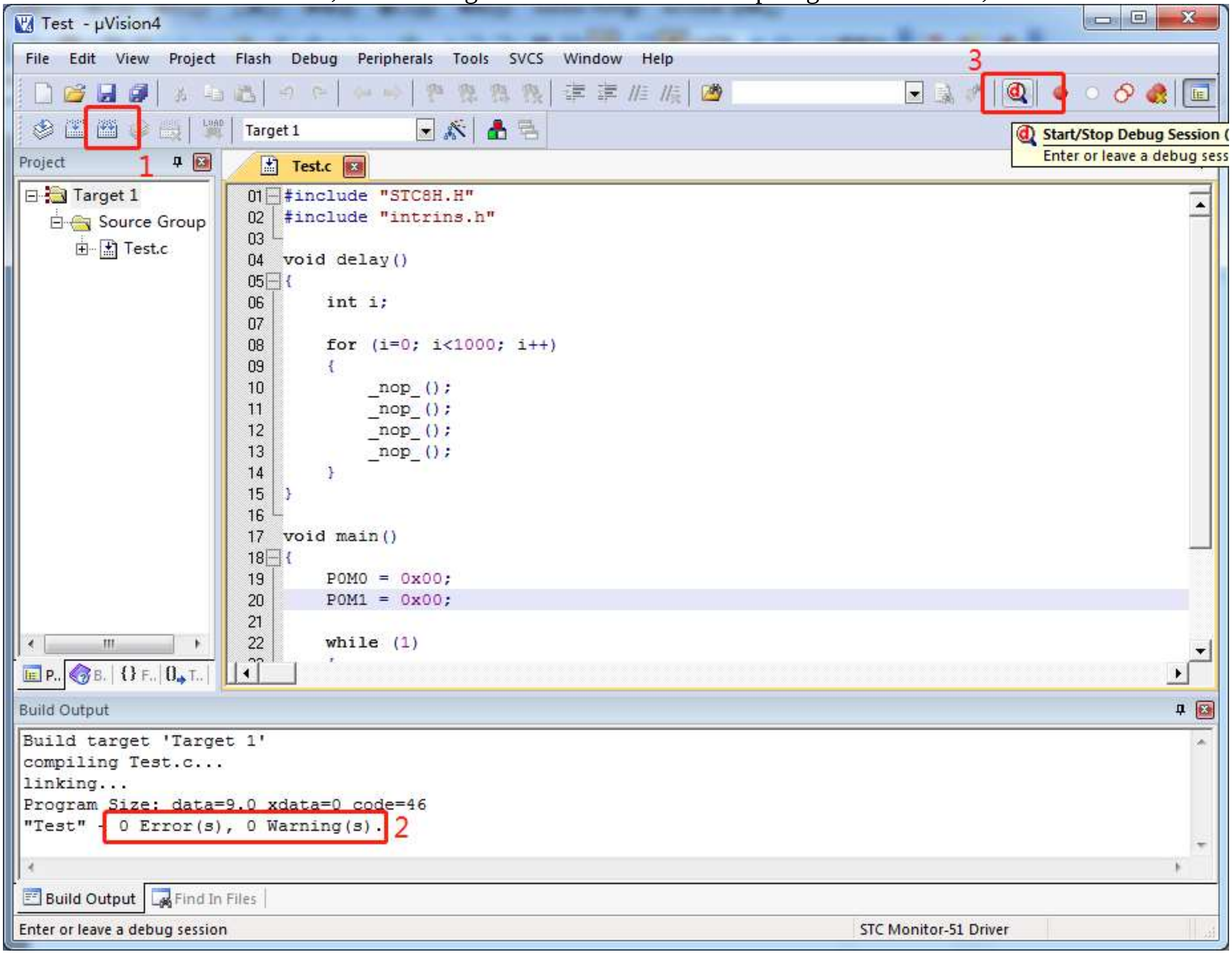
Dialog DLL: DP51.DLL Parameter: -pDP8051

Dialog DLL: TP51.DLL Parameter: -p51

Note: Please select the serial port according to the actual connection, and the baud rate is generally 115200.

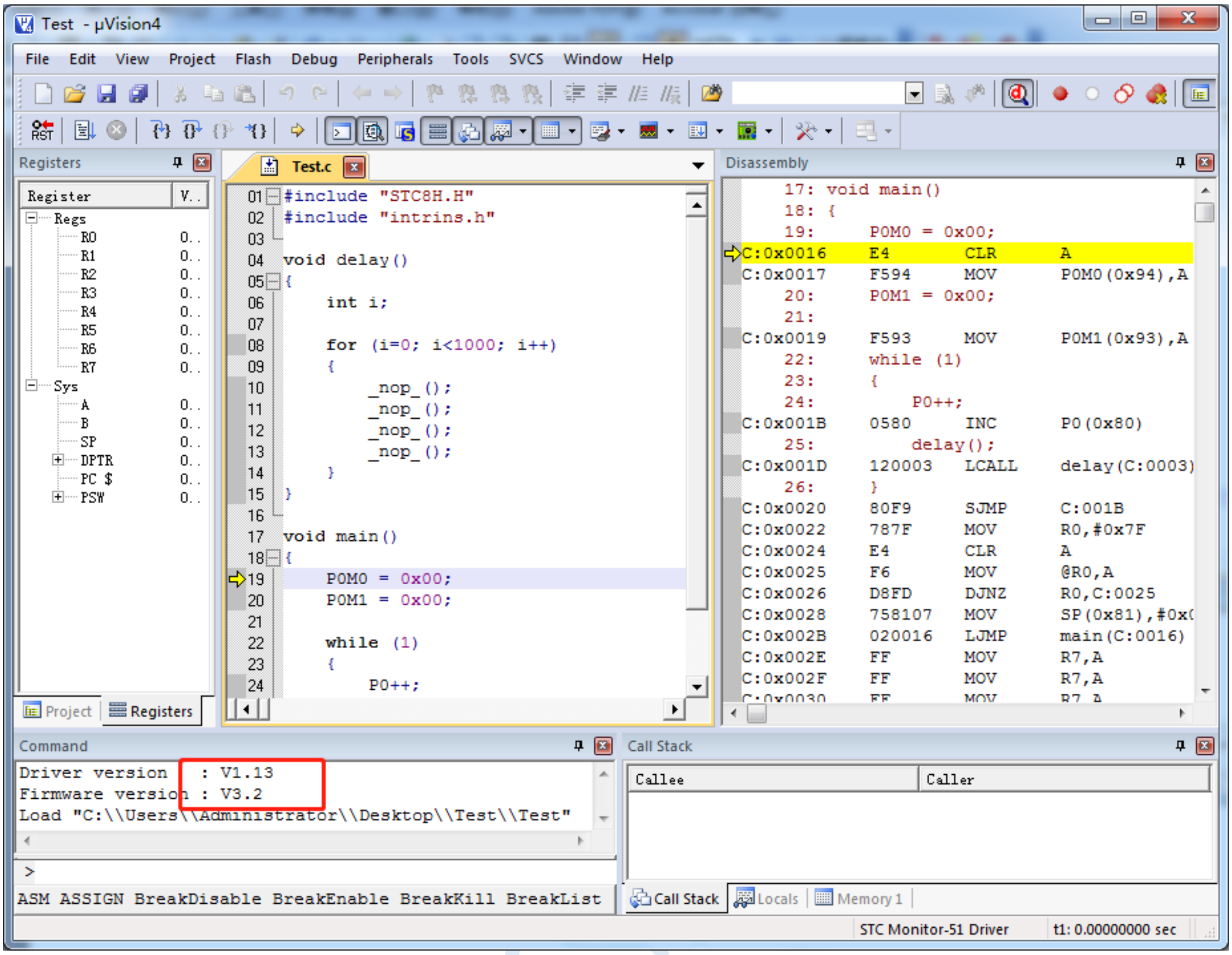
I.4.3 Emulation using serial port in Keil software

In the Keil environment, after editing the source code and compiling without errors, the emulationcan be started.



```
#include "STC8H.H"
#include "intrins.h"

void delay()
{
    int i;

    for (i=0; i<1000; i++)
    {
        _nop_();
        _nop_();
        _nop_();
        _nop_();
    }
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;

    while (1)
```

```
Build target 'Target 1'
compiling Test.c...
linking...
Program Size: data=9.0 xdata=0 code=46
"Test" - 0 Error(s), 0 Warning(s).
```



```
#include "STC8H.H"
#include "intrins.h"

void delay()
{
    int i;

    for (i=0; i<1000; i++)
    {
        _nop_();
        _nop_();
        _nop_();
        _nop_();
    }
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;

    while (1)
    {
        P0++;
```

```
    17: void main()
    18: {
    19:     P0M0 = 0x00;
C:0x0016    E4       CLR      A
C:0x0017    F594     MOV      P0M0(0x94),A
    20:     P0M1 = 0x00;
    21:
C:0x0019    F593     MOV      P0M1(0x93),A
    22:     while (1)
    23:     {
    24:         P0++;
C:0x001B    0580     INC      P0(0x80)
    25:         delay();
C:0x001D    120003   LCALL    delay(C:0003)
    26:     }
C:0x0020    80F9     SJMP     C:001B
C:0x0022    787F     MOV      R0,#0x7F
C:0x0024    E4       CLR      A
C:0x0025    F6       MOV      @R0,A
C:0x0026    D8FD     DJNZ     R0,C:0025
C:0x0028    758107   MOV      SP(0x81),#0x(
C:0x002B    020016   LJMP     main(C:0016)
C:0x002E    FF       MOV      R7,A
C:0x002F    FF       MOV      R7,A
```

```
Driver version   : V1.13
Firmware version : V3.2
Load "C:\\Users\\Administrator\\Desktop\\Test\\Test"
>
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

If the chip making and connection are correct, the emulation driver version will be displayed as shown in the figure above, and the user code can be downloaded to the microcontroller correctly, and then debugging functions such as running, single step, and breakpoint can be performed.

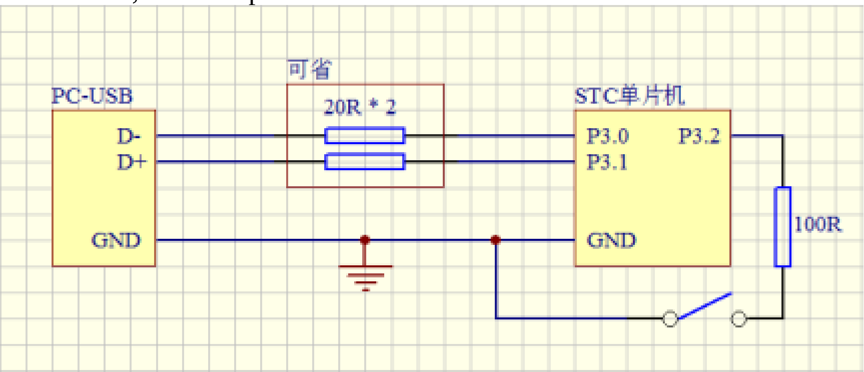
I.5 USB direct emulation (currently only supported by STC8H8K64U-B version chip)

I.5.1 Making a USB emulation chip

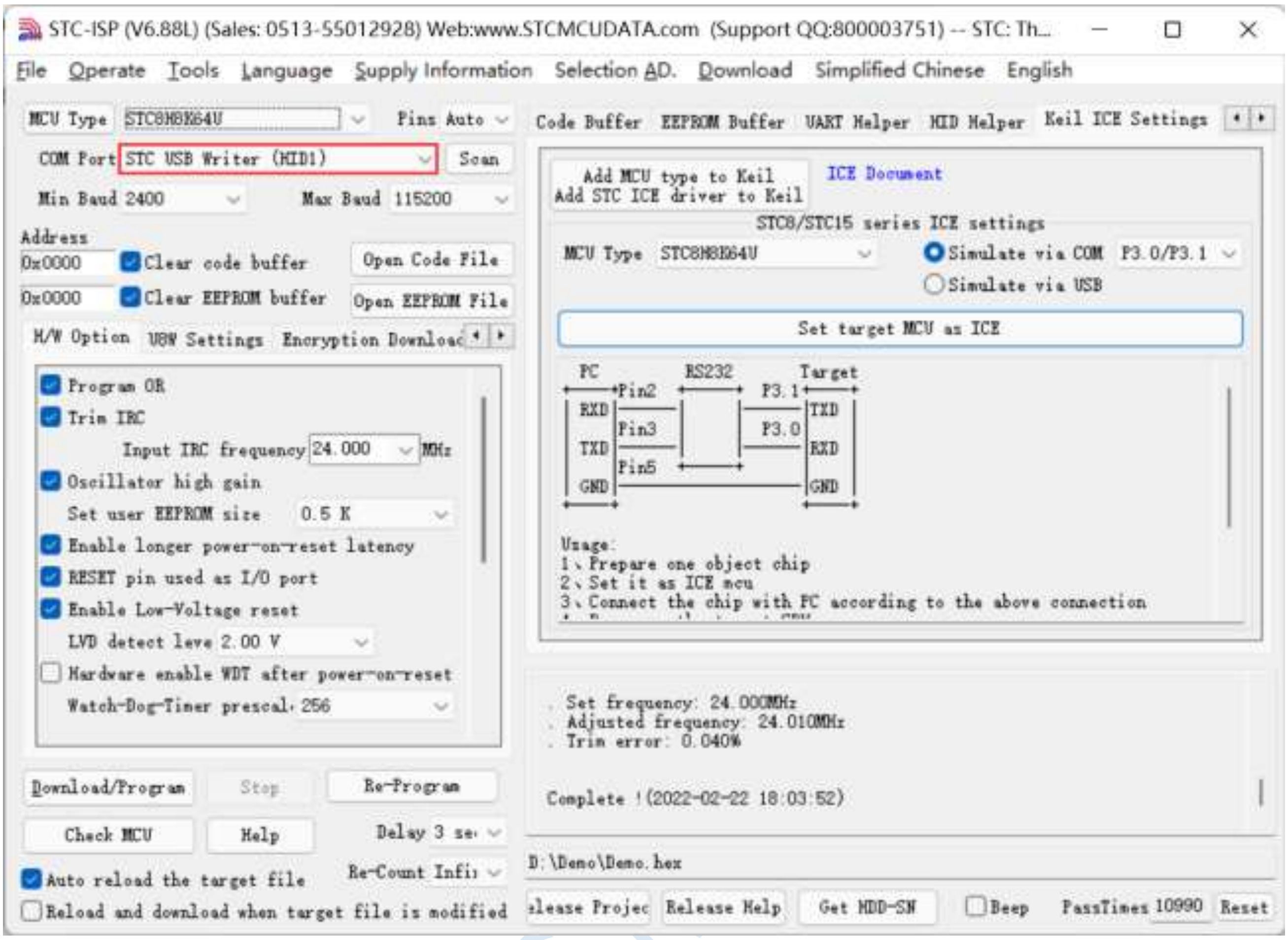
To make a USB emulation chip, you can use the serial ISP to make it according to the steps in Section 4.1, or use the USB-ISP method. This section will introduce how to use the USB-ISP to make it.

The setting steps are as follows:

Firstly, connect the target chip to the serial port of the computer as shown in the figure below, and short-circuit P3.2 to GND through the switch, and then power on the microcontroller.



If "STC USB Writer (HID1)" can be automatically scanned in the ISP software, it means the connection is correct.



STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...

MCU Type STC8H8K64U Pins Auto
COM Port STC USB Writer (HID1) Scan
Min Baud 2400 Max Baud 115200
Address
0x0000 ☑ Clear code buffer Open Code File
0x0000 ☑ Clear EEPROM buffer Open EEPROM File

☑ Program OR
☑ Trim IRC
Input IRC frequency 24.000 MHz
☑ Oscillator high gain
Set user EEPROM size 0.5 K
☑ Enable longer power-on-reset latency
☑ RESET pin used as I/O port
☑ Enable Low-Voltage reset
LVD detect leve 2.00 V
□ Hardware enable WDT after power-on-reset
Watch-Dog-Timer prescal 256

Delay 3 se
☑ Auto reload the target file Re-Count Infi
□ Reload and download when target file is modified

STC8/STC15 series ICE settings
MCU Type STC8H8K64U ● Simulate via COM P3.0/P3.1
○ Simulate via USB

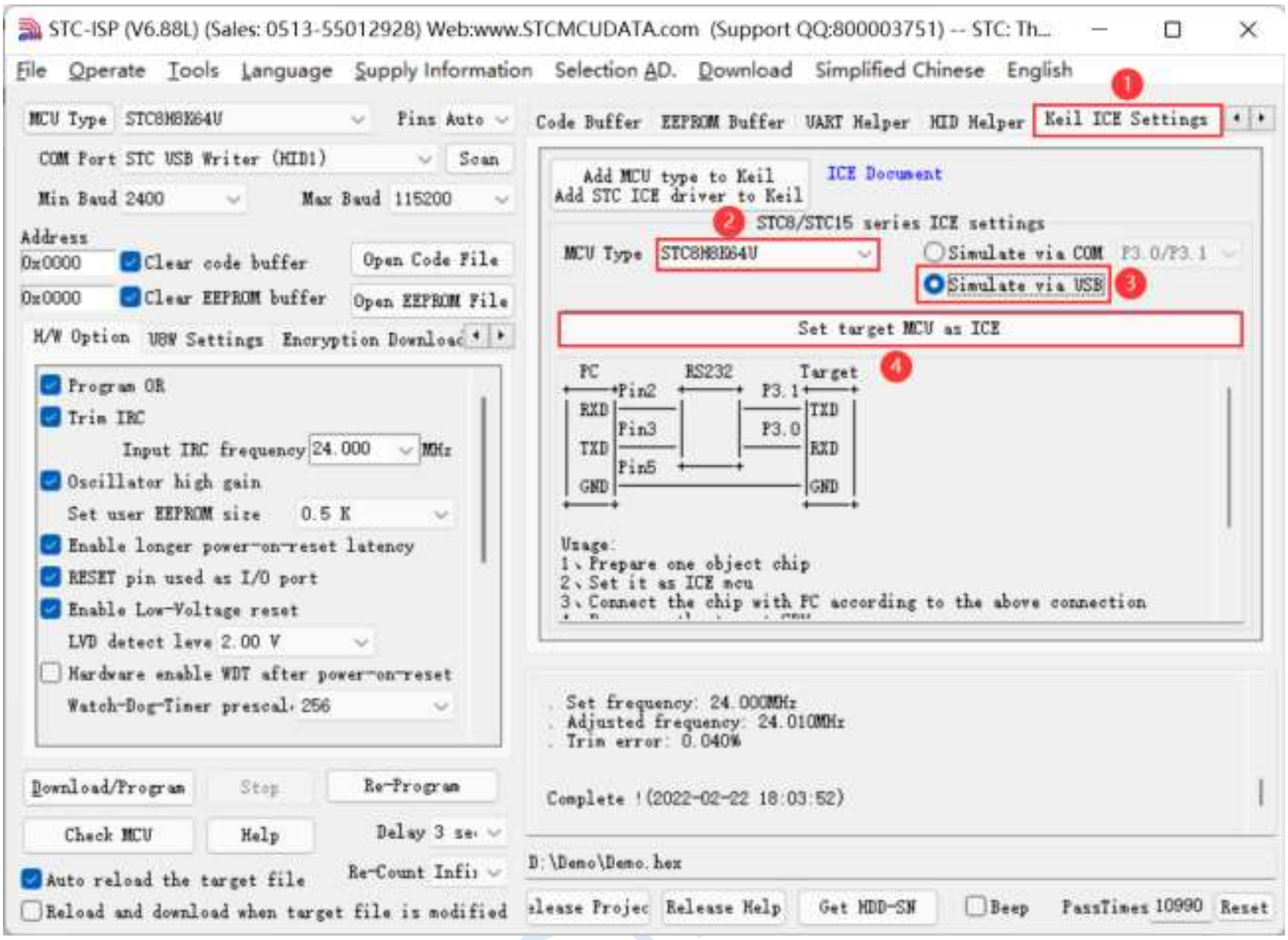
Usage:
1、Prepare one object chip
2、Set it as ICE mcu
3、Connect the chip with PC according to the above connection

```
. Set frequency: 24.000MHz
. Adjusted frequency: 24.010MHz
. Trim error: 0.040%

Complete !(2022-02-22 18:03:52)
```

D:\Demo\Demo.hex

□ Beep PassTimes 10990 Reset

Next, in the STC-ISP download software, follow the steps shown in the figure below to set up the emulation chip.



MCU Type STC8H8K64U

Pins Auto

COM Port STC USB Writer (HID1)

Min Baud 2400

Max Baud 115200

Address

0x0000 ☑ Clear code buffer

0x0000 ☑ Clear EEPROM buffer

☑ Program OR

☑ Trim IRC

Input IRC frequency 24.000 MHz

☑ Oscillator high gain

Set user EEPROM size 0.5 K

☑ Enable longer power-on-reset latency

☑ RESET pin used as I/O port

☑ Enable Low-Voltage reset

LVD detect leve 2.00 V

☐ Hardware enable WDT after power-on-reset

Watch-Dog-Timer prescal 256

Delay 3 sec

☑ Auto reload the target file

Re-Count Infini

☐ Reload and download when target file is modified

MCU Type STC8H8K64U

○ Simulate via COM P3.0/P3.1

● Simulate via USB

Usage:
1、Prepare one object chip
2、Set it as ICE mcu
3、Connect the chip with PC according to the above connection

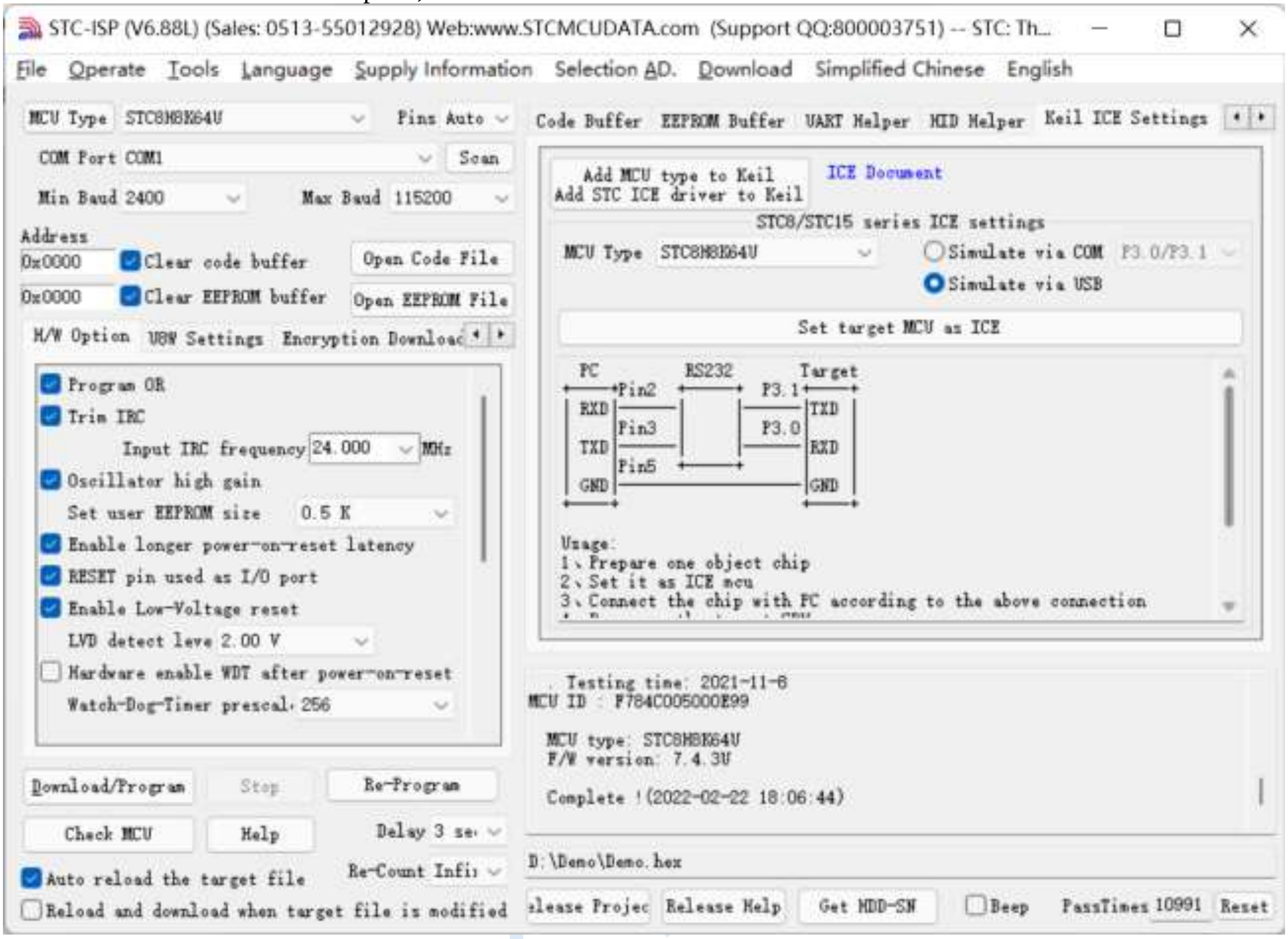
```
. Set frequency: 24.000MHz
. Adjusted frequency: 24.010MHz
. Trim error: 0.040%

Complete !(2022-02-22 18:03:52)
```

D:\Demo\Demo.hex

☐ Beep

PassTimes 10990

After the download is complete, it will be as shown below.



STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...

MCU Type STC8H8K64U   Pins Auto

COM Port COM1   Scan

Min Baud 2400   Max Baud 115200

Address

0x0000 ☑ Clear code buffer   Open Code File

0x0000 ☑ Clear EEPROM buffer   Open EEPROM File

☑ Program OR

☑ Trim IRC

Input IRC frequency 24.000 MHz

☑ Oscillator high gain

Set user EEPROM size 0.5 K

☑ Enable longer power-on-reset latency

☑ RESET pin used as I/O port

☑ Enable Low-Voltage reset

LVD detect leve 2.00 V

□ Hardware enable WDT after power-on-reset

Watch-Dog-Timer prescal 256

Delay 3 se

☑ Auto reload the target file   Re-Count Infi

□ Reload and download when target file is modified

STC8/STC15 series ICE settings

MCU Type STC8H8K64U   ○ Simulate via COM P3.0/P3.1

● Simulate via USB

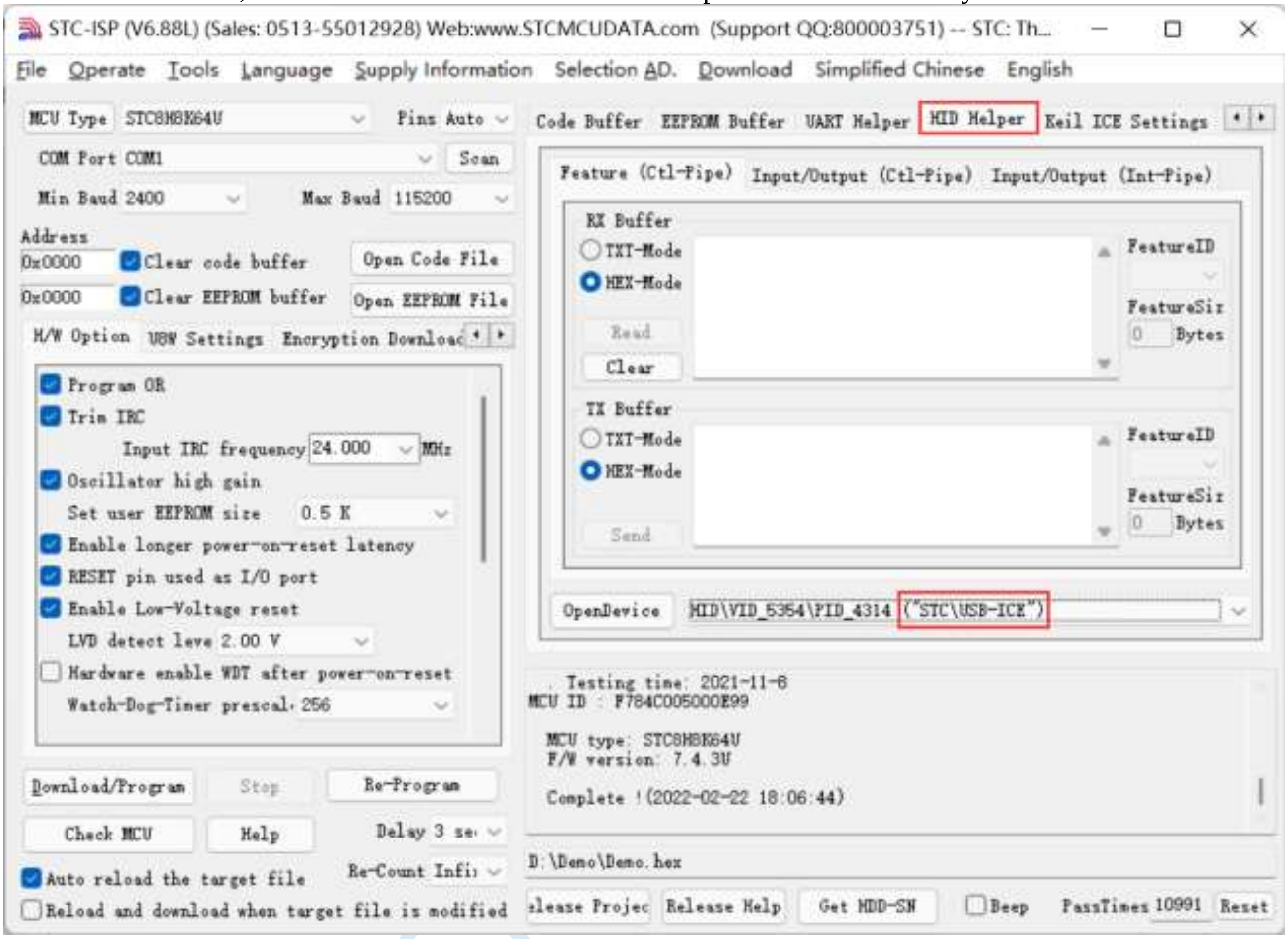
Usage:
1、Prepare one object chip
2、Set it as ICE mcu
3、Connect the chip with PC according to the above connection

```
 Testing time: 2021-11-6
MCU ID : F784C005000E99

 MCU type: STC8H8K64U
 F/W version: 7.4.3U

 Complete !(2022-02-22 18:06:44)
```

D:\Demo\Demo.hex

PassTimes 10991

After the production is completed, the grounding switch of the P3.2 port needs to be disconnected, and the MCU needs to be powered on again. If the "STC\USB-ICE" device can be detected in the "HID Helper" in the downloaded software, it means that the USB emulation The chip was made successfully.



MCU Type STC8H8K64U Pins Auto

COM Port COM1

Min Baud 2400 Max Baud 115200

Address

0x0000 ☑ Clear code buffer

0x0000 ☑ Clear EEPROM buffer

- ☑ Program OR
- ☑ Trim IRC
- Input IRC frequency 24.000 MHz
- ☑ Oscillator high gain
- Set user EEPROM size 0.5 K
- ☑ Enable longer power-on-reset latency
- ☑ RESET pin used as I/O port
- ☑ Enable Low-Voltage reset
- LVD detect leve 2.00 V
- □ Hardware enable WDT after power-on-reset
- Watch-Dog-Timer prescal 256

Delay 3 sec

Re-Count Infinite

☑ Auto reload the target file

□ Reload and download when target file is modified

RX Buffer: ○ TXT-Mode ● HEX-Mode FeatureID FeatureSize 0 Bytes

TX Buffer: ○ TXT-Mode ● HEX-Mode FeatureID FeatureSize 0 Bytes

HID\VID_5354\PID_4314 ("STC\USB-ICE")

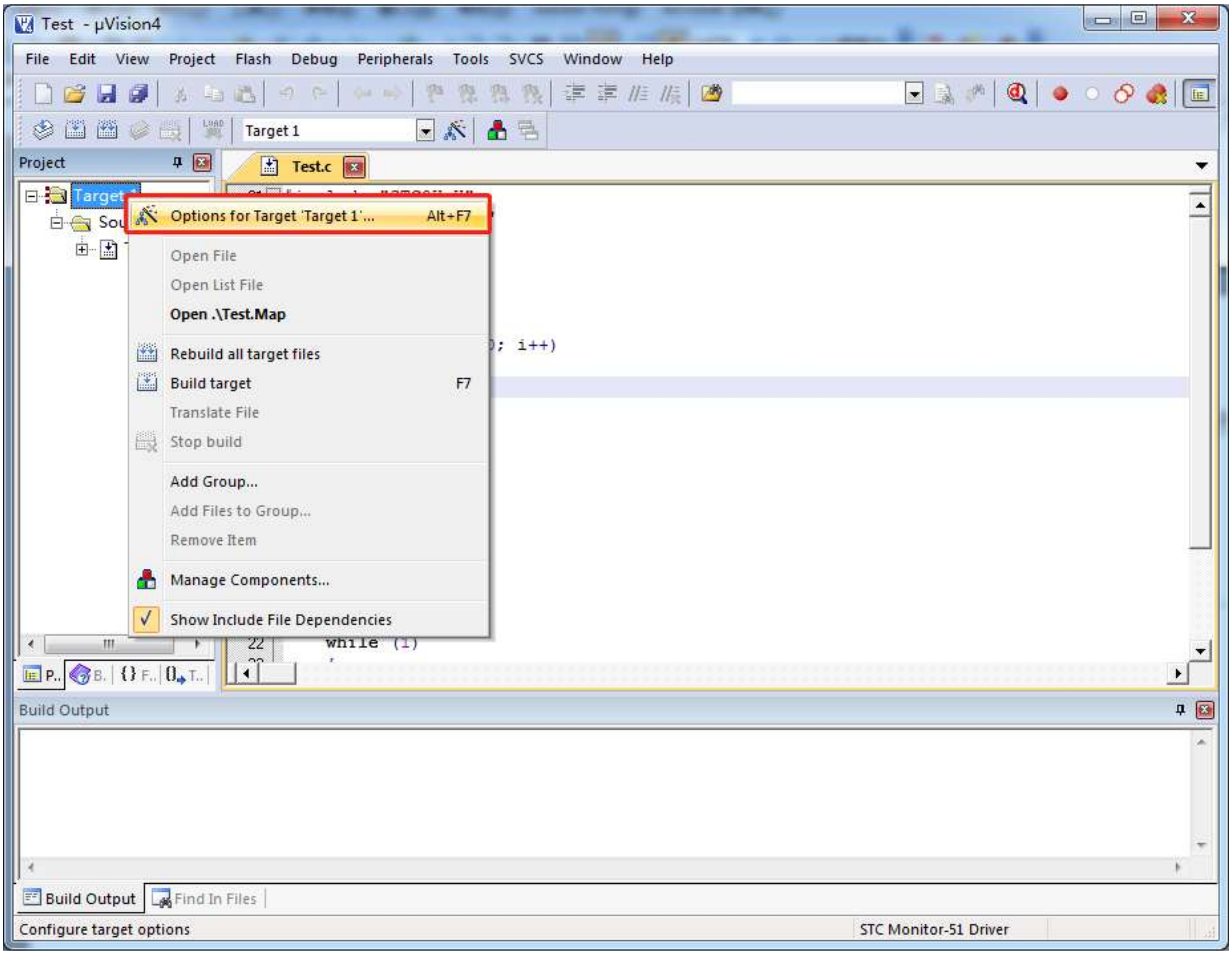
```
   Testing time: 2021-11-6
MCU ID : F784C005000E99

  MCU type: STC8H8K64U
  F/W version: 7.4.3U

  Complete !(2022-02-22 18:06:44)
```

D:\Demo\Demo.hex

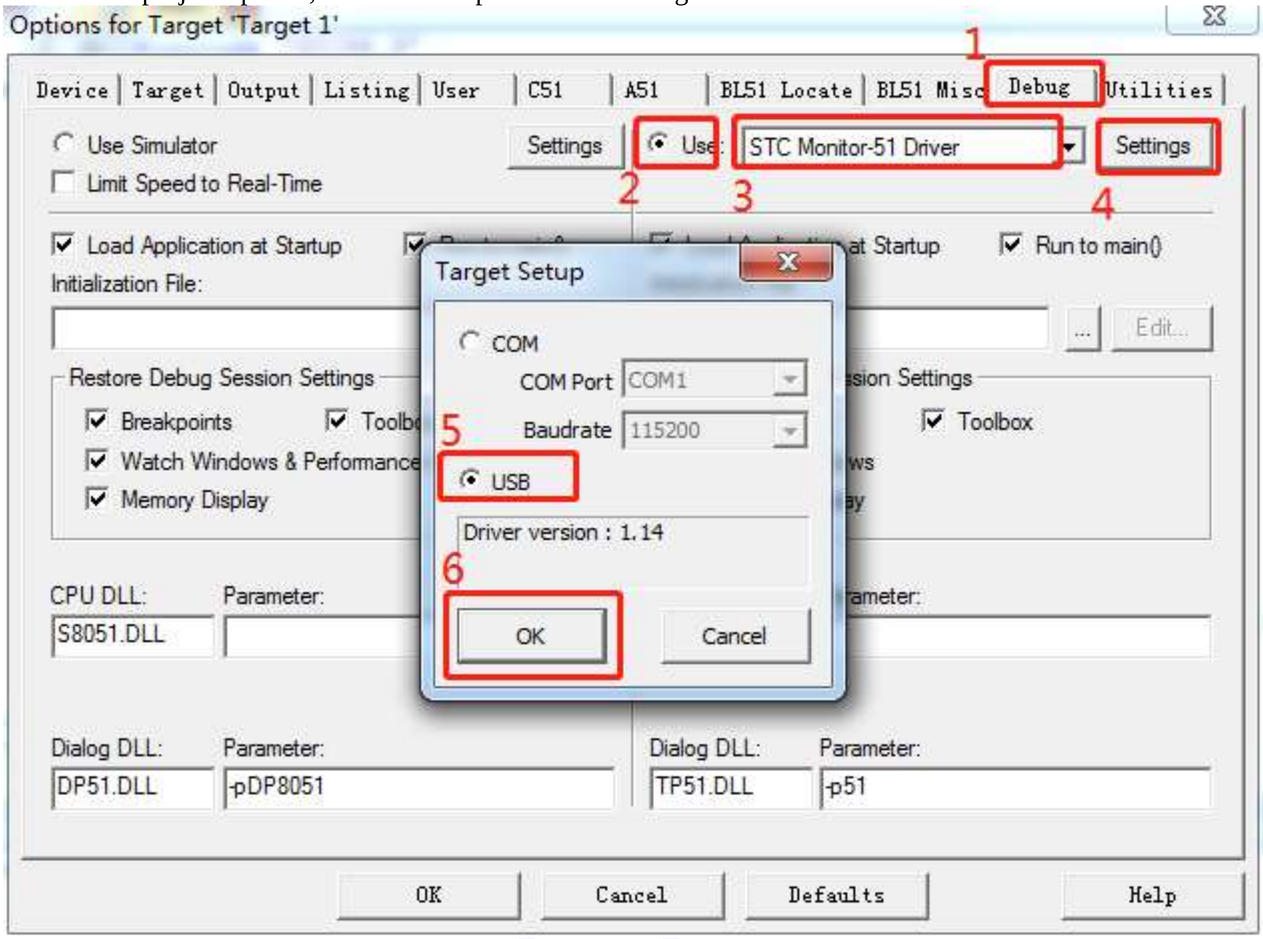
□ Beep PassTimes 10991

I.5.2 USB emulation settings in Keil software

Open the project file in Keil software, and click "Options for ..." in the right-click menu as shown in the figure below.



In the project options, follow the steps shown in the figure below to set the USB emulation.



Options for Target 'Target 1'

1 Debug

○ Use Simulator

□ Limit Speed to Real-Time

2 ● Use:

3 STC Monitor-51 Driver

4 Settings

☑ Load Application at Startup ☑ Run to main()

Initialization File:

Restore Debug Session Settings

☑ Breakpoints ☑ Toolbox

☑ Watch Windows & Performance

☑ Memory Display

CPU DLL: S8051.DLL Parameter:

Dialog DLL: DP51.DLL Parameter: -pDP8051

Dialog DLL: TP51.DLL Parameter: -p51

Target Setup

○ COM

COM Port COM1

Baudrate 115200

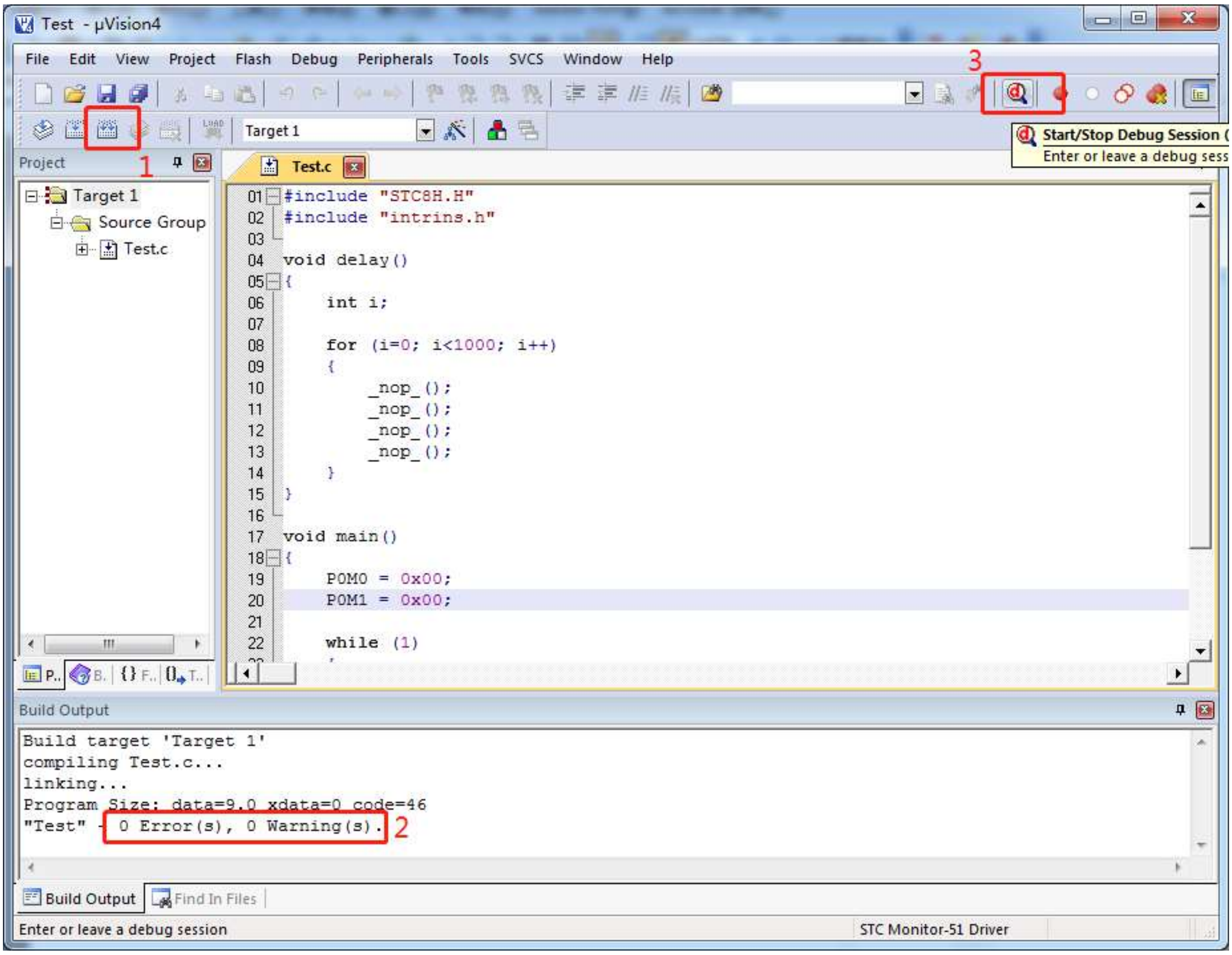
5 ● USB

Driver version : 1.14

6 OK

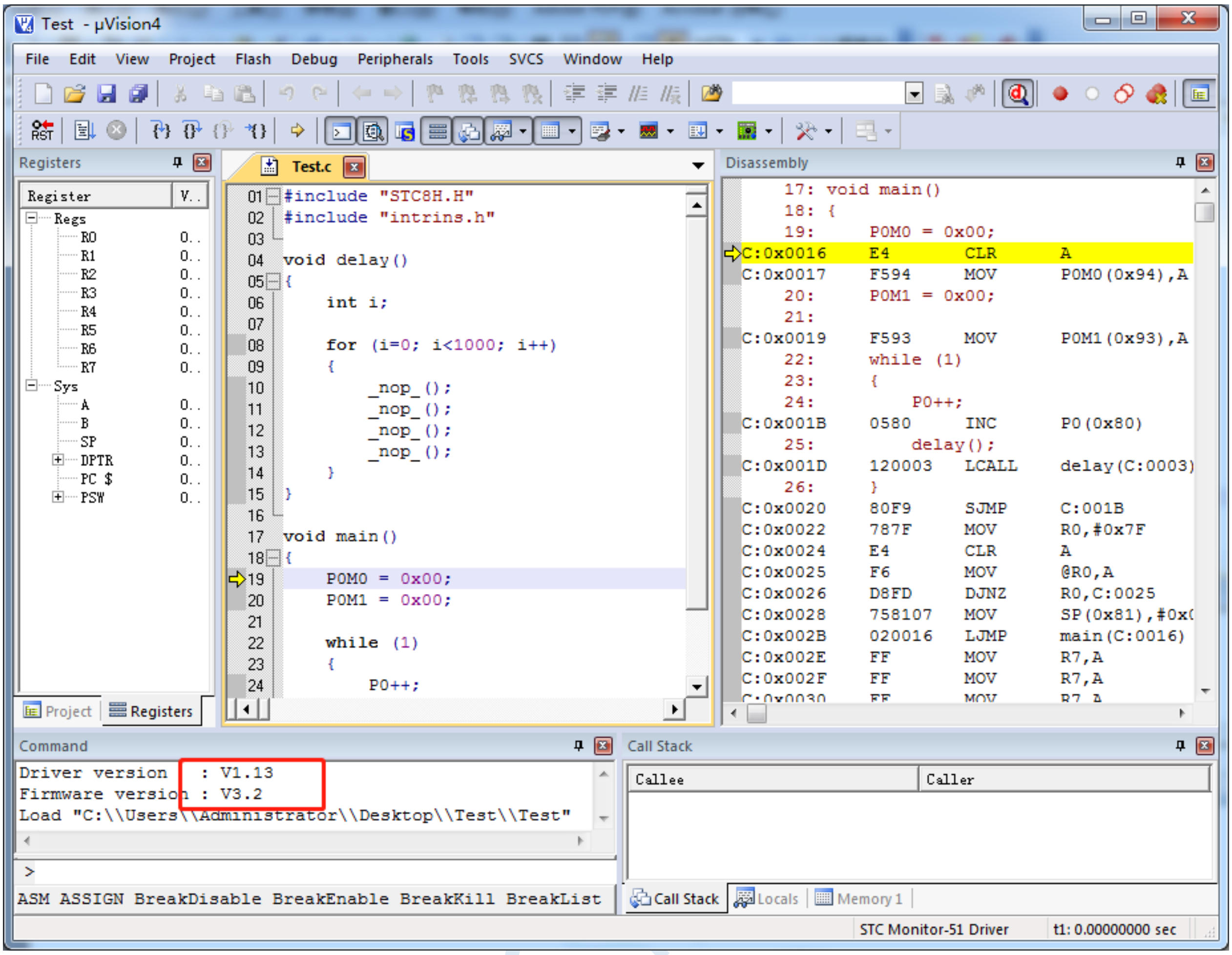
I.5.3 Emulation using USB in Keil software

In the Keil environment, after editing the source code and compiling without errors, the emulation can be started.


Test - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
1
Target 1
Source Group
Test.c
3
Start/Stop Debug Session (
Enter or leave a debug sess
2
Build Output
Find In Files
Enter or leave a debug session
STC Monitor-51 Driver

```
#include "STC8H.H"
#include "intrins.h"

void delay()
{
    int i;

    for (i=0; i<1000; i++)
    {
        _nop_();
        _nop_();
        _nop_();
        _nop_();
    }
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;

    while (1)
```

```
Build target 'Target 1'
compiling Test.c...
linking...
Program Size: data=9.0 xdata=0 code=46
"Test" - 0 Error(s), 0 Warning(s).
```



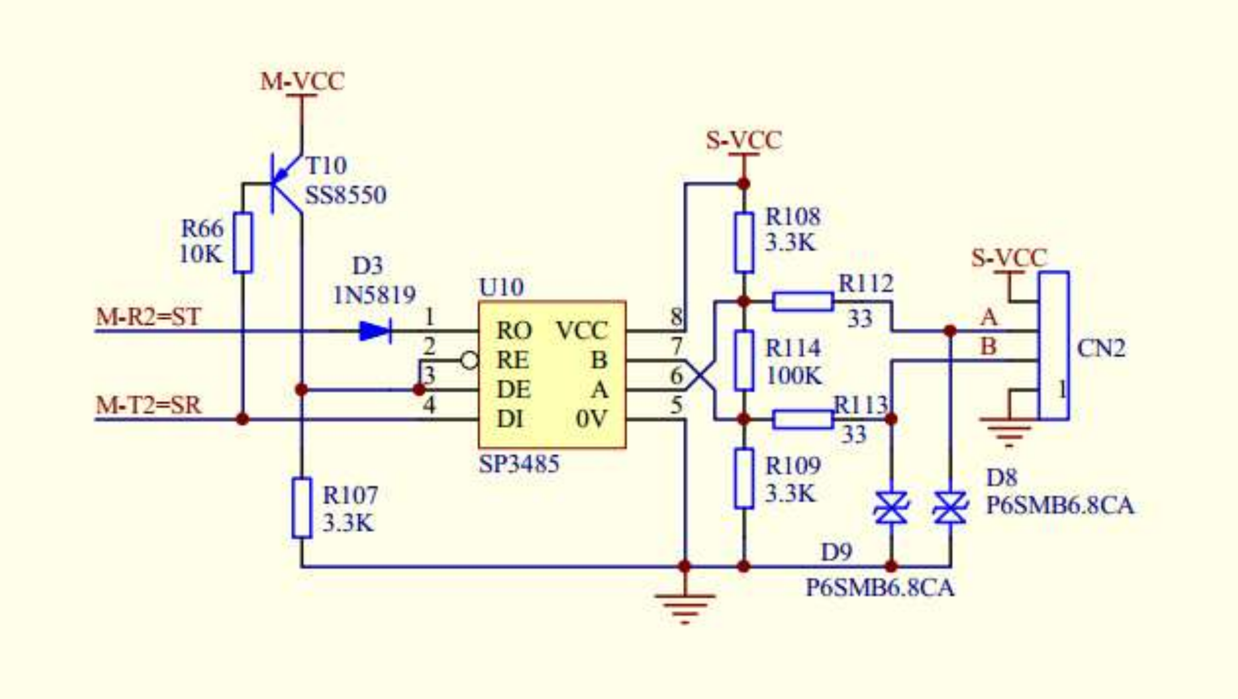
```
#include "STC8H.H"
#include "intrins.h"

void delay()
{
    int i;

    for (i=0; i<1000; i++)
    {
        _nop_();
        _nop_();
        _nop_();
        _nop_();
    }
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;

    while (1)
    {
        P0++;
```

```
    17: void main()
    18: {
    19:     P0M0 = 0x00;
C:0x0016    E4       CLR      A
C:0x0017    F594     MOV      P0M0(0x94),A
    20:     P0M1 = 0x00;
    21:
C:0x0019    F593     MOV      P0M1(0x93),A
    22:     while (1)
    23:     {
    24:         P0++;
C:0x001B    0580     INC      P0(0x80)
    25:         delay();
C:0x001D    120003   LCALL    delay(C:0003)
    26:     }
C:0x0020    80F9     SJMP     C:001B
C:0x0022    787F     MOV      R0,#0x7F
C:0x0024    E4       CLR      A
C:0x0025    F6       MOV      @R0,A
C:0x0026    D8FD     DJNZ     R0,C:0025
C:0x0028    758107   MOV      SP(0x81),#0x(
C:0x002B    020016   LJMP     main(C:0016)
C:0x002E    FF       MOV      R7,A
C:0x002F    FF       MOV      R7,A
C:0x0030    FF       MOV      R7,A
```

```
Driver version   : V1.13
Firmware version : V3.2
Load "C:\\Users\\Administrator\\Desktop\\Test\\Test"

>
ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList
```

If the chip making and connection are correct, the emulation driver version will be displayed as shown in the figure above, and the user code can be downloaded to the microcontroller correctly, and then debugging functions such as running, single step, and breakpoint can be performed.

Appendix J Partial Circuit of RS485 in U8W Download Tool



BOM list:

Label	Model	Package	Note
U10	SP3485EN	SOP8	RS485 chip
R66	10K	0603	Resistor
R107	3.3K	0603	Resistor
R108	3.3K	0603	Resistor
R109	3.3K	0603	Resistor
R112	33R	0603	Resistor
R113	33R	0603	Resistor
R114	100K	0603	Resistor
T10	SS8550	SOT-23	PNP Triode
D3	1N5819	0603	Schottky diode
D8	P6SMB6.8CA	DO-214AA	TVS diode
D9	P6SMB6.8CA	DO-214AA	TVS diode
CN2		SIP4	Communication Interface

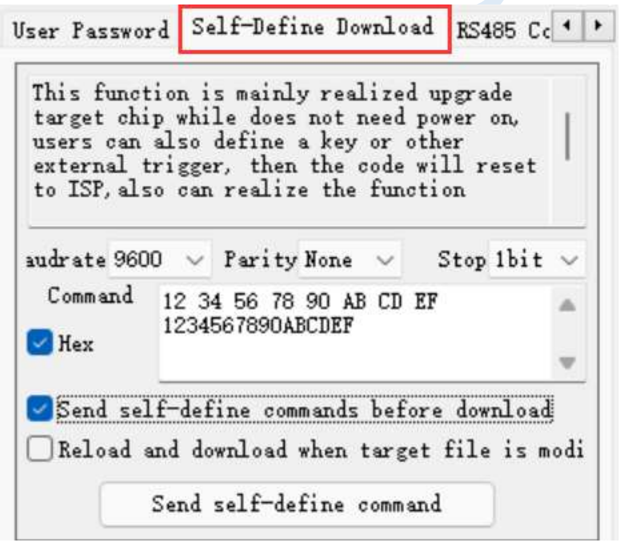
Appendix K ISP Download Starts Automatically After Receiving User Command While Running User Program (no Power-down)

"User-defined download" and "user-defined encrypted download" are two completely different functions. Compared with the function of user-defined encrypted download, the function of user-defined download is simpler.

The specific functions is: Before the computer or offline download board starts to send the real ISP download programming handshake command, it first sends a user-defined string of commands (for this string of serial commands, user can set the baud rate, parity, and stop bits), and then immediately sends the ISP download programming handshake command.

The function of "user-defined download" is mainly used in the early development stage of the project, which can download user code without power-off (without re-power-on to the target chip). The specific implementation method is: User needs to add a piece of code to detect the custom command in user program. When the command is detected, execute the assembly code of "MOV IAP_CONTR, # 60H" or the C language code of "IAP_CONTR = 0x60;" , MCU will reset to ISP area to execute ISP code automatically.

As shown in the figure below, set the custom command sequence with a baud rate of 115200, no parity bit, and one stop bit: 0x12, 0x34, 0x56, 0xAB, 0xCD, 0xEF, 0x12. When the option "Send custom commands before each download" is checked, the user-defined download function can be implemented.



This function is mainly realized upgrade target chip while does not need power on, users can also define a key or other external trigger, then the code will reset to ISP, also can realize the function

audrate 9600 Parity None Stop 1bit

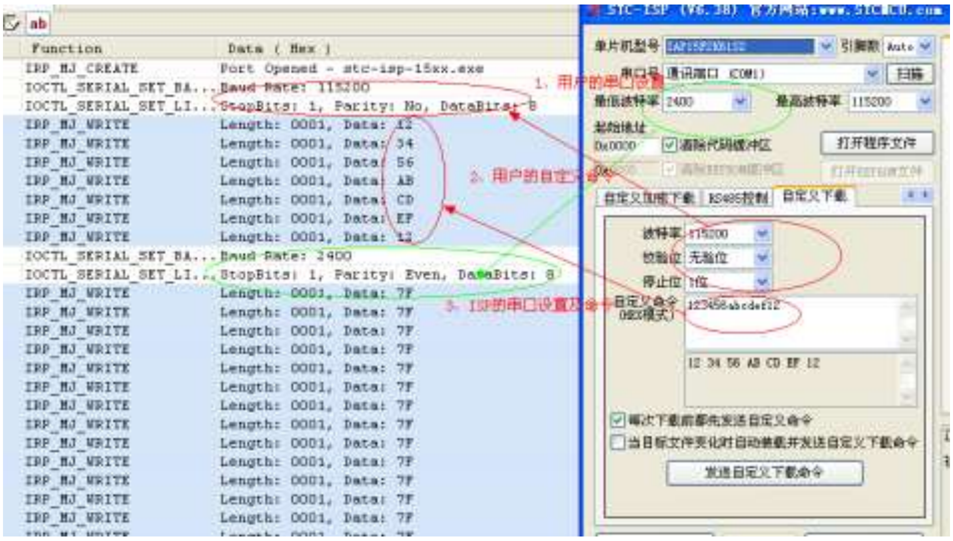
Command
12 34 56 78 90 AB CD EF
1234567890ABCDEF

☑ Hex

☑ Send self-define commands before download

□ Reload and download when target file is modi

Click "Send user-defined download command" or click the "Download / Program" button in the lower left corner of the window, the application will send the serial data as shown below.



```
Function                    Data ( Hex )
IRP_MJ_CREATE               Port Opened - stc-isp-15xx.exe
IOCTL_SERIAL_SET_BA...      Baud Rate: 115200
IOCTL_SERIAL_SET_LI...      StopBits: 1, Parity: No, DataBits: 8
IRP_MJ_WRITE                Length: 0001, Data: 12
IRP_MJ_WRITE                Length: 0001, Data: 34
IRP_MJ_WRITE                Length: 0001, Data: 56
IRP_MJ_WRITE                Length: 0001, Data: AB
IRP_MJ_WRITE                Length: 0001, Data: CD
IRP_MJ_WRITE                Length: 0001, Data: EF
IRP_MJ_WRITE                Length: 0001, Data: 12
IOCTL_SERIAL_SET_BA...      Baud Rate: 2400
IOCTL_SERIAL_SET_LI...      StopBits: 1, Parity: Even, DataBits: 8
IRP_MJ_WRITE                Length: 0001, Data: 7F
IRP_MJ_WRITE                Length: 0001, Data: 7F
IRP_MJ_WRITE                Length: 0001, Data: 7F
IRP_MJ_WRITE                Length: 0001, Data: 7F
IRP_MJ_WRITE                Length: 0001, Data: 7F
IRP_MJ_WRITE                Length: 0001, Data: 7F
IRP_MJ_WRITE                Length: 0001, Data: 7F
IRP_MJ_WRITE                Length: 0001, Data: 7F
IRP_MJ_WRITE                Length: 0001, Data: 7F
IRP_MJ_WRITE                Length: 0001, Data: 7F
IRP_MJ_WRITE                Length: 0001, Data: 7F
IRP_MJ_WRITE                Length: 0001, Data: 7F
IRP_MJ_WRITE                Length: 0001, Data: 7F
```

STC-ISP (V6.38) 官方网站:www.STCMCU.com

单片机型号 IAP15F2K61S2 引脚数 Auto

串口号 通讯端口 (COM1) 扫描

最低波特率 2400 最高波特率 115200

起始地址 0x0000 ☑ 清除代码缓冲区 打开程序文件

0x0000 ☑ 清除EEPROM缓冲区 打开EEPROM文件

自定义加密下载 | RS485控制 | 自定义下载

波特率 115200

校验位 无校验

停止位 1位

自定义命令 (HEX模式) 123456abcdef12

12 34 56 AB CD EF 12

☑ 每次下载前都先发送自定义命令

☐ 当目标文件变化时自动装载并发送自定义下载命令

发送自定义下载命令

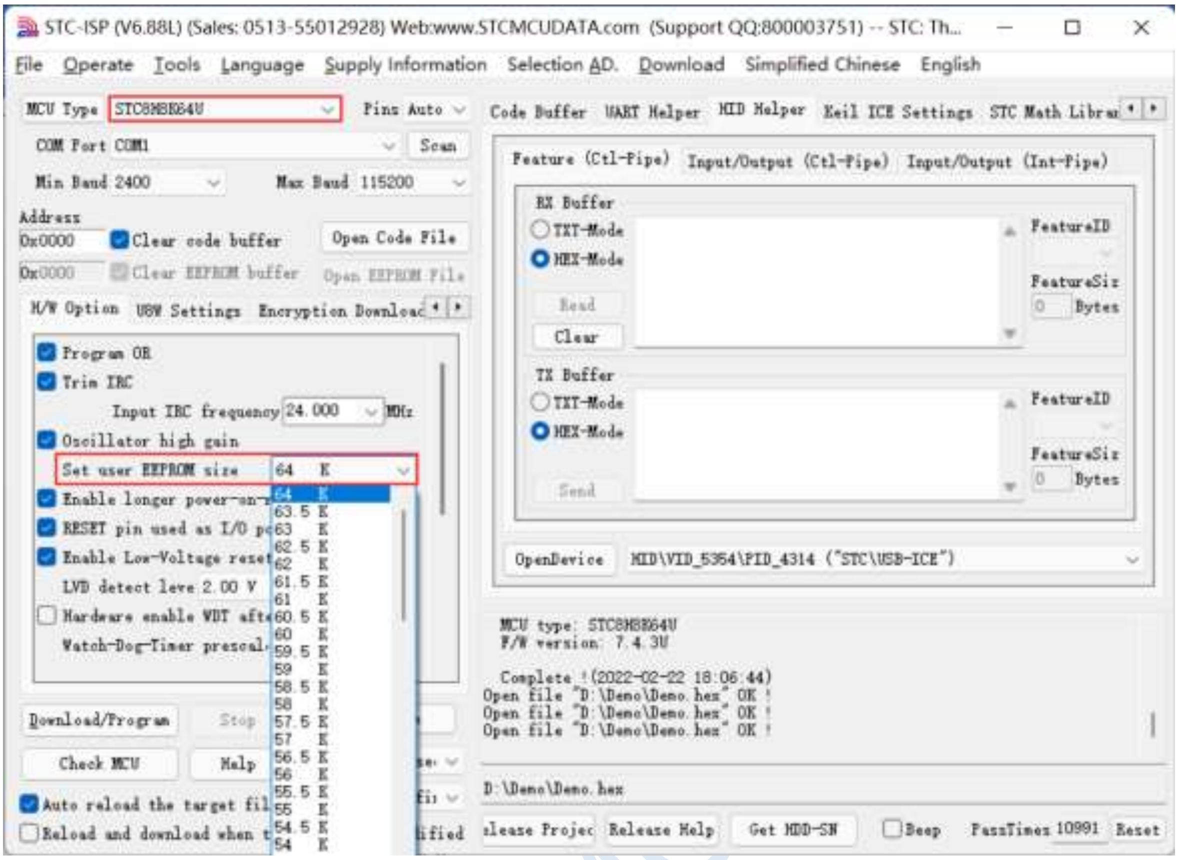
Appendix L Use STC's IAP series MCU to develop your own ISP program

With the continuous development of IAP (In-Application-Programming) technology in the field of single-chip microcomputers, it has brought great convenience to the application system program code upgrade. STC's serial ISP (In-System-Programming) program uses the IAP function to upgrade the user's program online, but for the sake of user code safety, neither the underlying code nor the upper application is open source. For this reason, STC launched With the IAP series single-chip microcomputer, that is, the Flash space of the entire MCU, users can rewrite in their own programs, so that the idea that users need to develop their own ISP programs can be realized.

All the models in the STC8H series microcontrollers that can customize the EEPROM size during ISP download are IAP series.Film machine. At present, the STC8H series have the following types of microcontrollers as the IAP series: STC8H1K12, STC8H1K17, STC8H1K28, STC8H1K33, STC8H3K64S2, STC8H3K64S4, STC8H8K64U, STC8H2K64T. This article uses STC8H8K64U,As an example, the method of using STC's IAP MCU to develop user's own ISP program is explained in detail, and the assembly and C source code based on Keil environment are given.

The first step: internal FLASH planning

Since the EEPROM of the IAP microcontroller of the STC8H series is set by the user during ISP download, if the user needs to implement his own ISP, when downloading the user's own ISP program, the user needs to follow the figure below to set all 64K Set it to EEPROM, so that the user program space and EEPROM space are completely overlapped, so that users can modify and update their own program space.


STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...
File Operate Tools Language Supply Information Selection AD. Download Simplified Chinese English
Scan
Open Code File
Open EEPROM File
H/W Option USB Settings Encryption Downloa
Code Buffer UART Helper HID Helper Keil ICE Settings STC Math Librar
Feature (Ctl-Pipe) Input/Output (Ctl-Pipe) Input/Output (Int-Pipe)
Read
Clear
Send
OpenDevice
Download/Program
Stop
Check MCU
Help
Release Projec Release Help
Get HDD-SN
Reset

MCU Type STC8H8K64U
Pins Auto
COM Port COM1
Min Baud 2400
Max Baud 115200
Address
0x0000 ☑ Clear code buffer
0x0000 ☑ Clear EEPROM buffer

☑ Program OR
☑ Trim IRC
Input IRC frequency 24.000 MHz
☑ Oscillator high gain
Set user EEPROM size 64 K
☑ Enable longer power-on-r
☑ RESET pin used as I/O p
☑ Enable Low-Voltage reset
LVD detect leve 2.00 V
☐ Hardware enable WDT aft
Watch-Dog-Timer prescal

64 K
63.5 K
63 K
62.5 K
62 K
61.5 K
61 K
60.5 K
60 K
59.5 K
59 K
58.5 K
58 K
57.5 K
57 K
56.5 K
56 K
55.5 K
55 K
54.5 K
54 K

☑ Auto reload the target fil
☐ Reload and download when t

RX Buffer
○ TXT-Mode
● HEX-Mode
FeatureID
FeatureSiz 0 Bytes

TX Buffer
○ TXT-Mode
● HEX-Mode
FeatureID
FeatureSiz 0 Bytes

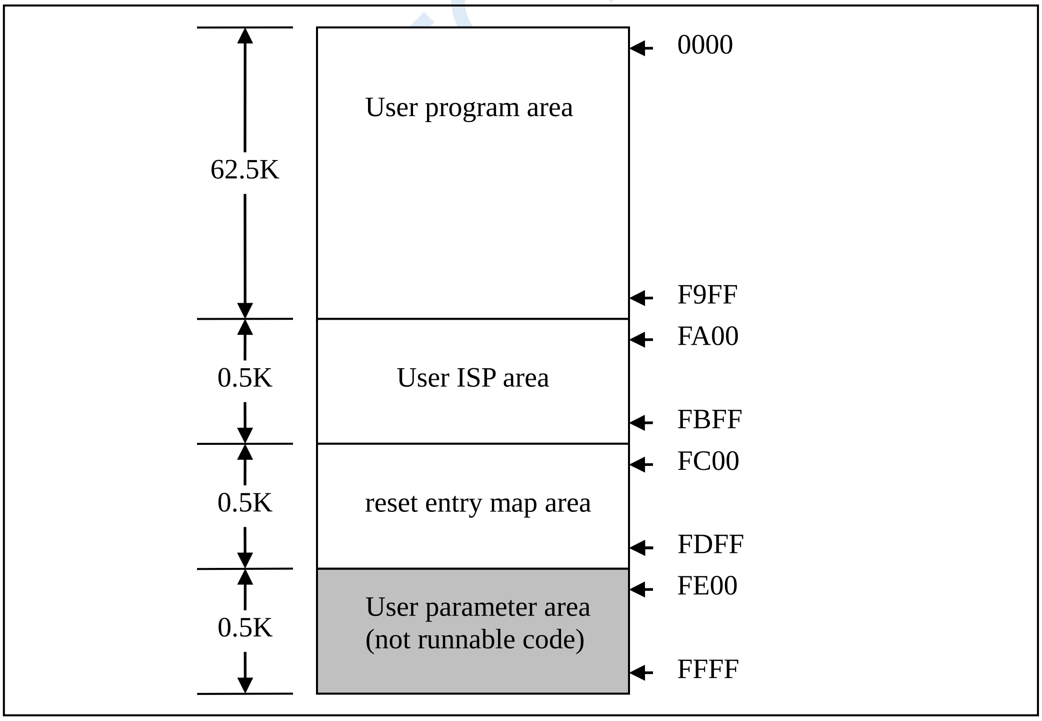
HID\VID_5354\PID_4314 ("STC\USB-ICE")

```
MCU type: STC8H8K64U
F/W version: 7.4.3U

Complete !(2022-02-22 18:06:44)
Open file "D:\Demo\Demo.hex" OK !
Open file "D:\Demo\Demo.hex" OK !
Open file "D:\Demo\Demo.hex" OK !
```

D:\Demo\Demo.hex

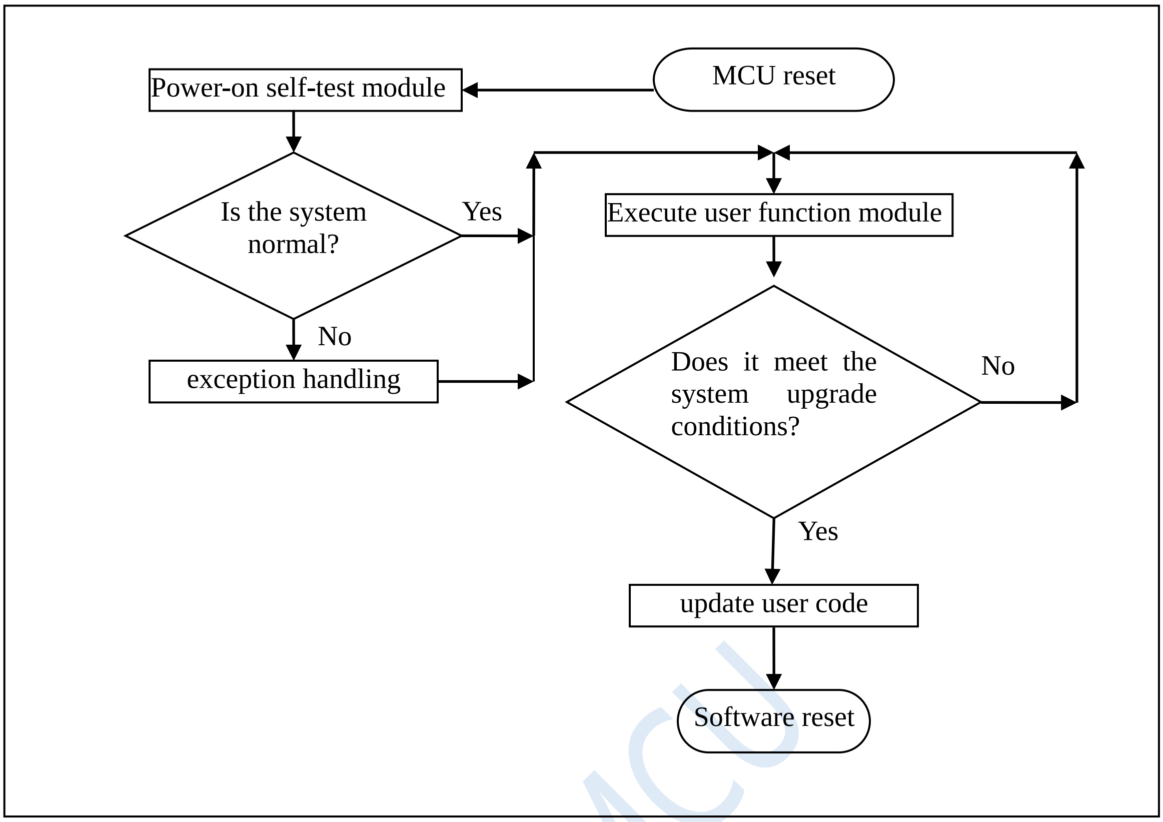
☐ Beep PassTimes 10991

The following assumes that the user has set the entire 64K program space as EEPROM. Now the entire 64K program space is divided as follows:



In the FLASH space, the continuous 62.5K bytes of space starting from address 0000H is the user program area. When the specific download conditions are met, the user is required to jump the PC to the user ISP program area. At this time, the user program area can be erased and rewritten to achieve the purpose of updating the user program

The second step, the basic framework of the program


MCU reset
Power-on self-test module
Is the system normal?
Yes
No
exception handling
Execute user function module
Does it meet the system upgrade conditions?
No
Yes
update user code
Software reset

The third step, the firmware program description of the lower computer

The firmware program of the lower computer includes two parts: ISP (ISP code) and AP (user code)

ISP Code (assembly code)

```
; Operating frequency for test is 11.0592MHz

UARTBAUD    EQU     0FFE8H              ;Define the serial port baud rate   (65536-11059200/4/115200)

AUXR        DATA    08EH                ;Additional Function Control Register
WDT_CONTR   DATA    0C1H                ;Watchdog Control Register
IAP_DATA    DATA    0C2H                ;IAP data register
IAP_ADDRH   DATA    0C3H                ;IAP High Address Register
IAP_ADDRL   DATA    0C4H                ;IAP Low Address Register
IAP_CMD     DATA    0C5H                ;IAP Command Register
IAP_TRIG    DATA    0C6H                ;IAP Command Trigger Register
IAP_CONTR   DATA    0C7H                ;IAP Control Register
IAP_TPS     DATA    0F5H                ;IAP latency control register

ISPCODE     EQU     0FA00H              ;ISP module entry address (1 page), also external interface
address
APENTRY     EQU     0FC00H              ; Application entry address data (1 page)

            ORG     0000H

            LJMP    ISP_ENTRY           ;System reset entry

RESET:
```

```
                MOV         SCON,#50H               ;Set serial port mode (8 data bits, no parity bit)
                MOV         AUXR,#40H               ;Timer 1 is in 1T mode
                MOV         TMOD,#00H               ;Timer 1 works in mode 0 (16-bit reload)
                MOV         TH1,#HIGH UARTBAUD      ;set reload value
                MOV         TL1,#LOW UARTBAUD
                SETB        TR1                     ;start timer 1
NEXT1:
                MOV         R0,#16
NEXT2:
                JNB         RI,$                    ;Waiting for serial data
                CLR         RI
                MOV         A,SBUF
                CJNE        A,#7FH,NEXT1            ;Determine whether it is 7F
                DJNZ        R0,NEXT2
                LJMP        ISP_DOWNLOAD            ;Jump to download interface

                ORG         ISPCODE

ISP_DOWNLOAD:
                CLR         A
                MOV         PSW,A                   ;ISP module uses group 0 registers
                MOV         IE,A                    ;Disable all interrupts
                CLR         RI                      ;Clear serial port receive flag
                SETB        TI                      ; Set serial port send flag
                CLR         TR0
                MOV         SP,#5FH                 ;set stack pointer

                MOV         A,#5AH                  ;Return 5A 55 to PC, indicating ISP erase module is ready
                LCALL       ISP_SENDUART
                MOV         A,#055H
                LCALL       ISP_SENDUART
                LCALL       ISP_RECVACK             ;Receive response data

                MOV         IAP_ADDRL,#0            ;First write the "LJMP ISP_ENTRY" instruction at the starting
address of page 2
                MOV         IAP_ADDRH,#02H
                LCALL       ISP_ERASEIAP
                MOV         A,#02H
                LCALL       ISP_PROGRAMIAP          ;Programming User Code Reset Vector Code
                MOV         A,#HIGH                 ISP_ENTRY
                LCALL       ISP_PROGRAMIAP          ;Programming User Code Reset Vector Code
                MOV         A,#LOW ISP_ENTRY
                LCALL       ISP_PROGRAMIAP          ;Programming User Code Reset Vector Code

                MOV         IAP_ADDRL,#0            ;User code address starts from 0
                MOV         IAP_ADDRH,#0
                LCALL       ISP_ERASEIAP
                MOV         A,#02H
                LCALL       ISP_PROGRAMIAP          ;Programming User Code Reset Vector Code
                MOV         A,#HIGH                 ISP_ENTRY
                LCALL       ISP_PROGRAMIAP          ;Programming User Code Reset Vector Code
                MOV         A,#LOW ISP_ENTRY
                LCALL       ISP_PROGRAMIAP          ;Programming User Code Reset Vector Code

                MOV         IAP_ADDRL,#0            ;new code buffer address
                MOV         IAP_ADDRH,#02H
                MOV         R7,#124                 ;Erase 62.5K bytes
ISP_ERASEAP:
```

```
                LCALL       ISP_ERASEIAP
                INC         IAP_ADDRH                   ;target address+512
                INC         IAP_ADDRH
                DJNZ        R7,ISP_ERASEAP              ;Determine if erasing is complete

                MOV         IAP_ADDRL,#LOW APENTRY
                MOV         IAP_ADDRH,#HIGH APENTRY
                LCALL       ISP_ERASEIAP

                MOV         A,#5AH                      ;Return 5A A5 to PC, indicating that the ISP programming
module is ready
                LCALL       ISP_SENDUART
                MOV         A,#0A5H
                LCALL       ISP_SENDUART
                LCALL       ISP_RECVACK                 ;Receive response data

                LCALL       ISP_RECVUART                ;Receive length high byte
                MOV         R0,A
                LCALL       ISP_RECVUART                ;Receive length low byte
                MOV         R1,A
                CLR         C                           ; total length -3
                MOV         A,#03H
                SUBB        A,R1
                MOV         DPL,A
                CLR         A
                SUBB        A,R0
                MOV         DPH,A                       ;Total length complement stored in DPTR

                LCALL       ISP_RECVUART                ;Map user code reset entry code to map area
                LCALL       ISP_PROGRAMIAP              ;0000
                LCALL       ISP_RECVUART
                LCALL       ISP_PROGRAMIAP              ;0001
                LCALL       ISP_RECVUART
                LCALL       ISP_PROGRAMIAP              ;0002

                MOV         IAP_ADDRL,#03H              ;User code start address
                MOV         IAP_ADDRH,#00H
ISP_PROGRAMNEXT:
                LCALL       ISP_RECVUART                ;receive code data
                LCALL       ISP_PROGRAMIAP              ;Program the data into the user code area
                INC         DPTR
                MOV         A,DPL
                ORL         A,DPH
                JNZ         ISP_PROGRAMNEXT             ;length detection

ISP_SOFTRESET:
                MOV         IAP_CONTR,#20H              ;Software reset system
                SJMP        $


ISP_ENTRY:
                MOV         WDT_CONTR,#17H              ;clear watchdog
                MOV         IAP_CONTR,#80H              ;Enable IAP function
                MOV         IAP_TPS,#11                 ;Set the IAP wait time parameter
                MOV         IAP_ADDRL,#LOW ISP_DOWNLOAD
                MOV         IAP_ADDRH,#HIGH ISP_DOWNLOAD
                MOV         IAP_DATA,#00H               ;Test data 1
                MOV         IAP_CMD,#1                  ;Read command
                MOV         IAP_TRIG,#5AH           ;Triger ISP command
                MOV         IAP_TRIG,#0A5H
                MOV         A,IAP_DATA
                CJNE        A,#0E4H,ISP_ENTRY       ; If the data cannot be read, wait for the voltage to stabilize
                INC         IAP_ADDRL               ;Test address FC01H
                MOV         IAP_DATA,#45H           ;Test data 2
                MOV         IAP_CMD,#1              ;Read command
                MOV         IAP_TRIG,#5AH           ;Triger ISP command
                MOV         IAP_TRIG,#0A5H
                MOV         A,IAP_DATA
                CJNE        A,#0F5H,ISP_ENTRY       ; If the data cannot be read, wait for the voltage to stabilize

                MOV         SCON,#50H               ;Set serial port mode (8 data bits, no parity bit)
                MOV         AUXR,#40H               ;Timer 1 is in 1T mode
                MOV         TMOD,#00H               ;Timer 1 works in mode 0 (16-bit reload)
                MOV         TH1,#HIGH UARTBAUD      ;set reload value
                MOV         TL1,#LOW UARTBAUD
                SETB        TR1                     ;start timer 1
                SETB        TR0

                LCALL       ISP_RECVUART            ; Check if there is serial data
                JC          GOTOAP
                MOV         R0,#16
ISP_CHECKNEXT:
                LCALL       ISP_RECVUART            ; receive sync data
                JC          GOTOAP
                CJNE        A,#7FH,GOTOAP           ;Determine whether it is 7F
                DJNZ        R0,ISP_CHECKNEXT
                MOV         A,#5AH                  ; Return 5A 69 to PC, indicating ISP module is ready
                LCALL       ISP_SENDUART
                MOV         A,#69H
                LCALL       ISP_SENDUART
                LCALL       ISP_RECVACK             ;Receive response data
                LJMP        ISP_DOWNLOAD            ;Jump to download interface
GOTOAP:
                CLR         A                       ; Reset SFR to reset value
                MOV         TCON,A
                MOV         TMOD,A
                MOV         TL0,A
                MOV         TH0,A
                MOV         TL1,A
                MOV         TH1,A
                MOV         SCON,A
                MOV         AUXR,A
                LJMP        APENTRY                 ; Run the user program normally

ISP_RECVACK:
                LCALL       ISP_RECVUART
                JC          GOTOAP
                XRL         A,#7FH
                JZ          ISP_RECVACK             ; skip sync data
                CJNE        A,#25H,GOTOAP           ; Response data 1 detection
                LCALL       ISP_RECVUART
                JC          GOTOAP
                CJNE        A,#69H,GOTOAP           ; Response data 2 detection
                RET

ISP_RECVUART:
```

```
                CLR         A
                MOV         TL0,A                       ; Initialize timeout timer
                MOV         TH0,A
                CLR         TF0
                MOV         WDT_CONTR,#17H              ;clear watchdog
ISP_RECVWAIT:
                JBC         TF0,ISP_RECVTIMEOUT         ; Timeout detection
                JNB         RI,ISP_RECVWAIT             ; wait for receive to complete
                MOV         A,SBUF                      ; Read serial data
                CLR         RI                          ;Clear flag
                CLR         C                           ; Receive serial data correctly
                RET
ISP_RECVTIMEOUT:
                SETB        C                           ; timeout
                RET

ISP_SENDUART:
                MOV         WDT_CONTR,#17H              ;clear watchdog
                JNB         TI,ISP_SENDUART             ; Wait for the previous data transmission to complete
                CLR         TI                          ; Clear flag
                MOV         SBUF,A                      ; send current data
                RET

ISP_ERASEIAP:
                MOV         WDT_CONTR,#17H              ;clear watchdog
                MOV         IAP_CMD,#3                  ; Erase command
                MOV         IAP_TRIG,#5AH               ; Trigger ISP command
                MOV         IAP_TRIG,#0A5H
                NOP
                NOP
                NOP
                NOP
                RET

ISP_PROGRAMIAP:
                MOV         WDT_CONTR,#17H              ;clear watchdog
                MOV         IAP_CMD,#2                  ; programming command
                MOV         IAP_DATA,A                  ; send the current data to IAP data register
                MOV         IAP_TRIG,#5AH               ; Trigger ISP command
                MOV         IAP_TRIG,#0A5H
                NOP
                NOP
                NOP
                NOP
                MOV         A,IAP_ADDRL                 ;IAP address +1
                ADD         A,#01H
                MOV         IAP_ADDRL,A
                MOV         A,IAP_ADDRH
                ADDC        A,#00H
                MOV         IAP_ADDRH,A
                RET


                ORG         APENTRY
                LJMP        RESET
                END
```

The ISP code includes the following external interface modules:
ISP_DOWNLOAD: program download entry address, absolute address FA00H
ISP_ENTRY: Power-on system self-check program (automatically called by the system)
For the user program, the user only needs to jump the PC value to ISPPROGRAM (ie FA00H, Absolute address), you can update the code.

User codes (C language code)

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"

#define     FOSC            11059200L               // system clock frequency
#define     BAUD            (65536 - FOSC/4/115200) //Define the serial port baud rate
#define     ISPPROGRAM      0xfa00                  //ISP download program entry address

sfr         AUXR        =   0x8e;                   // Baud Rate Generator Control Register
sfr         P1M0        =   0x92;
sfr         P1M1        =   0x91;

void (*IspProgram)() = ISPPROGRAM;                  // define pointer function
char cnt7f;                                         //Isp_Check Internally used variables

void uart() interrupt 4                             // UART Interrupt Service Routine
{
        if (TI) TI = 0;                             // send complete interrupt
        if (RI)                                     // Receive complete interrupt
        {
                if (SBUF == 0x7f)
                {
                        cnt7f++;
                        if (cnt7f >= 16)
                        {
                                IspProgram();       // Invoke the download module (****important statement ****)
                        }
                }
                else
                {
                        cnt7f = 0;
                }
                RI = 0;                             // Clear the reception completion flag
        }
}

void main()
{
        SCON = 0x50;                                // Define the serial port mode as 8-bit, variable baud rate, no
parity bit
        AUXR = 0x40;
        TH1  = BAUD >> 8;
        TL1  = BAUD;
        TR1  = 1;
        ES = 1;                                     //Enable UART interrupt
        EA = 1;                                     //Enable CPU interrupt

        P1M0 = 0;
```

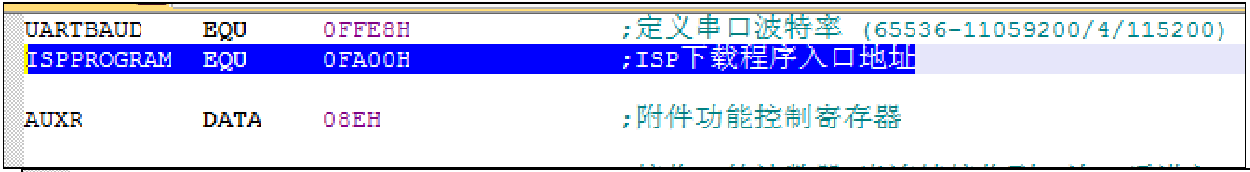
```
    P1M1 = 0;

    while (1)
    {
        P1++;
    }
}
```

User codes (assembly code)

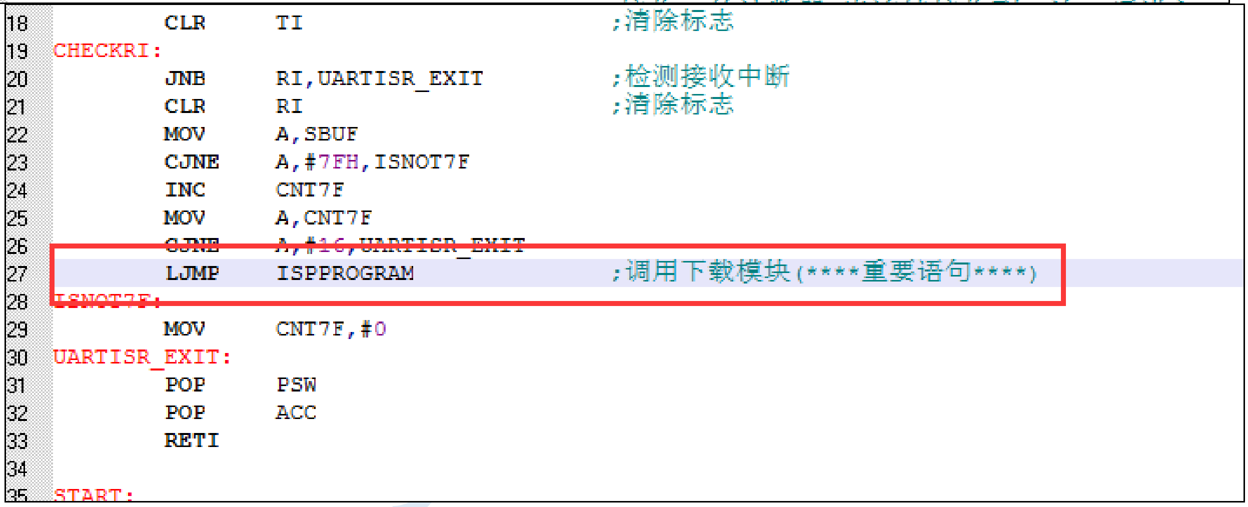
```
; Operating frequency for test is 11.0592MHz

UARTBAUD    EQU         0FFE8H                      ;Define the serial port baud rate   (65536-11059200/4/115200)
ISPPROGRAM  EQU         0FA00H                      ; ISP download program entry address

AUXR        DATA        08EH                        ; Additional Function Control Register

CNT7F       DATA        60H                         ; Receive 7F counter

            ORG         0000H
            LJMP        START                       ;System reset entry

            ORG         0023H
            LJMP        UART_ISR                    ; UART interrupt entry

UART_ISR:
            PUSH        ACC
            PUSH        PSW
            JNB         TI,CHECKRI                  ; Detect transmission interruption
            CLR         TI                          ; clear flag
CHECKRI:
            JNB         RI,UARTISR_EXIT             ; Detect receive interruption
            CLR         RI                          ; clear flag
            MOV         A,SBUF
            CJNE        A,#7FH,ISNOT7F
            INC         CNT7F
            MOV         A,CNT7F
            CJNE        A,#16,UARTISR_EXIT
            LJMP        ISPPROGRAM                  ; Invoke the download module (****important statement ****)
ISNOT7F:
            MOV         CNT7F,#0
UARTISR_EXIT:
            POP         PSW
            POP         ACC
            RETI

START:
            MOV         R0,#7FH                     ;Clear RAM
            CLR         A
            MOV         @R0,A
            DJNZ        R0,$-1
            MOV         SP,#7FH                     ; Initialize SP

            MOV         SCON,#50H                   ; Set UART mode (8-bit data, variable baud rate, no parity bit)
            MOV         AUXR,#15H                   ; BRT works in 1T mode, start BRT
            MOV         TMOD,#00H                   ;Timer 1 works in mode 0 (16-bit reload)
            MOV         TH1,#HIGH UARTBAUD          ;set reload value
            MOV         TL1,#LOW UARTBAUD
```

```
                SETB        TR1                         ;start timer 1
                SETB        ES                          ; Enable UART interrupt
                SETB        EA                          ; Enable CPU interrupt

MAIN:
                INC         P0
                SJMP        MAIN

                END
```

User code can be written in C or assembly language, but one thing to note about assembly code: the instruction at the reset entry address of 0000H must be a long jump statement (similar to LJMP START). In the user code, the serial port needs to be set up, and when the download conditions are met, the PC value is jumped to ISPPROGRAM (that is, the absolute address of FA00H) to achieve code update. For assembly code, we can use the "LJMP 0FA00H" instruction to call, as shown below

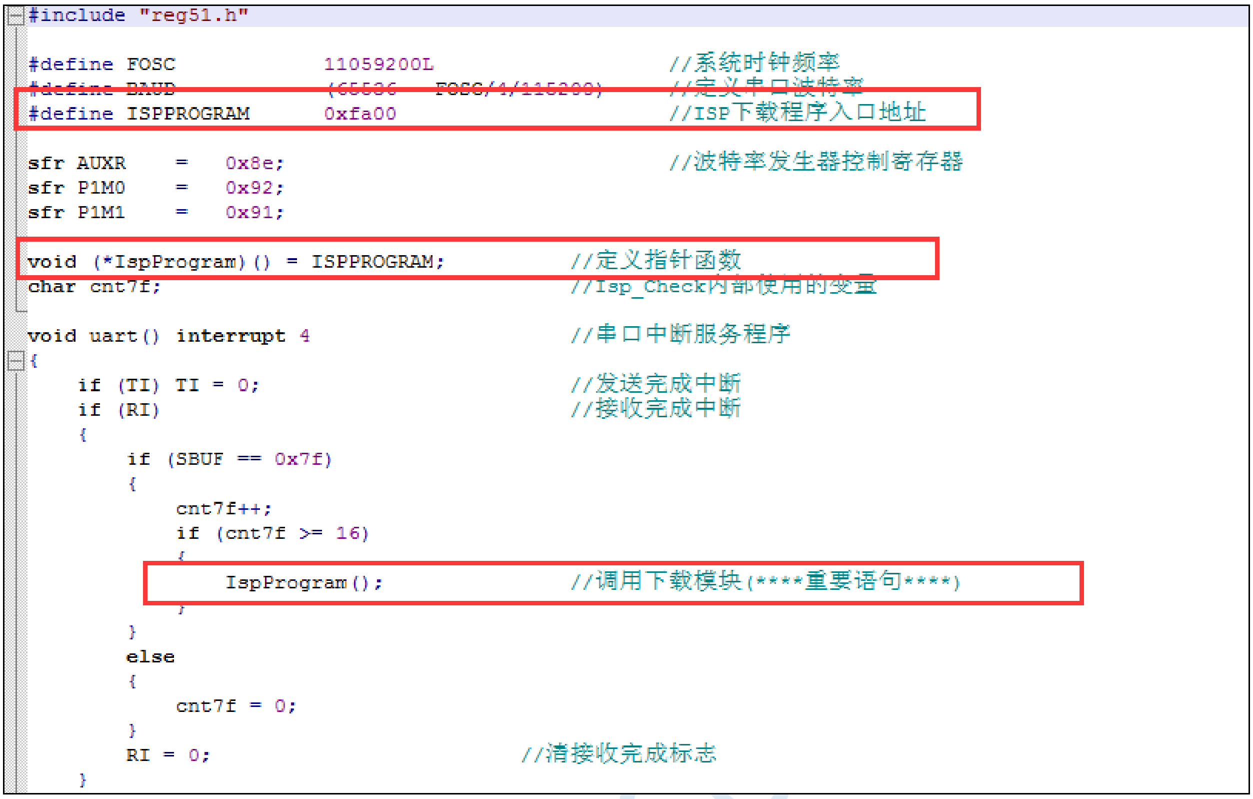


```
UARTBAUD      EQU      0FFE8H          ;定义串口波特率 (65536-11059200/4/115200)
ISPPROGRAM    EQU      0FA00H          ;ISP下载程序入口地址

AUXR          DATA     08EH            ;附件功能控制寄存器
```



```
                CLR      TI                      ;清除标志
CHECKRI:
                JNB      RI,UARTISR_EXIT         ;检测接收中断
                CLR      RI                      ;清除标志
                MOV      A,SBUF
                CJNE     A,#7FH,ISNOT7F
                INC      CNT7F
                MOV      A,CNT7F
                CJNE     A,#16,UARTISR_EXIT
                LJMP     ISPPROGRAM              ;调用下载模块(****重要语句****)
ISNOT7F:
                MOV      CNT7F,#0
UARTISR_EXIT:
                POP      PSW
                POP      ACC
                RETI

START:
```

In the C code, you must define a function pointer variable, and assign this variable to 0xFA00, and then call, as shown below:



```
#include "reg51.h"

#define FOSC            11059200L                   //系统时钟频率
#define BAUD            (65536 - FOSC/4/115200)     //定义串口波特率
#define ISPPROGRAM      0xfa00                      //ISP下载程序入口地址

sfr AUXR    =   0x8e;                               //波特率发生器控制寄存器
sfr P1M0    =   0x92;
sfr P1M1    =   0x91;

void (*IspProgram)() = ISPPROGRAM;          //定义指针函数
char cnt7f;                                 //Isp_Check内部使用的变量

void uart() interrupt 4                     //串口中断服务程序
{
    if (TI) TI = 0;                         //发送完成中断
    if (RI)                                 //接收完成中断
    {
        if (SBUF == 0x7f)
        {
            cnt7f++;
            if (cnt7f >= 16)
            {
                IspProgram();               //调用下载模块(****重要语句****)
            }
        }
        else
        {
            cnt7f = 0;
        }
        RI = 0;                         //清接收完成标志
    }
```

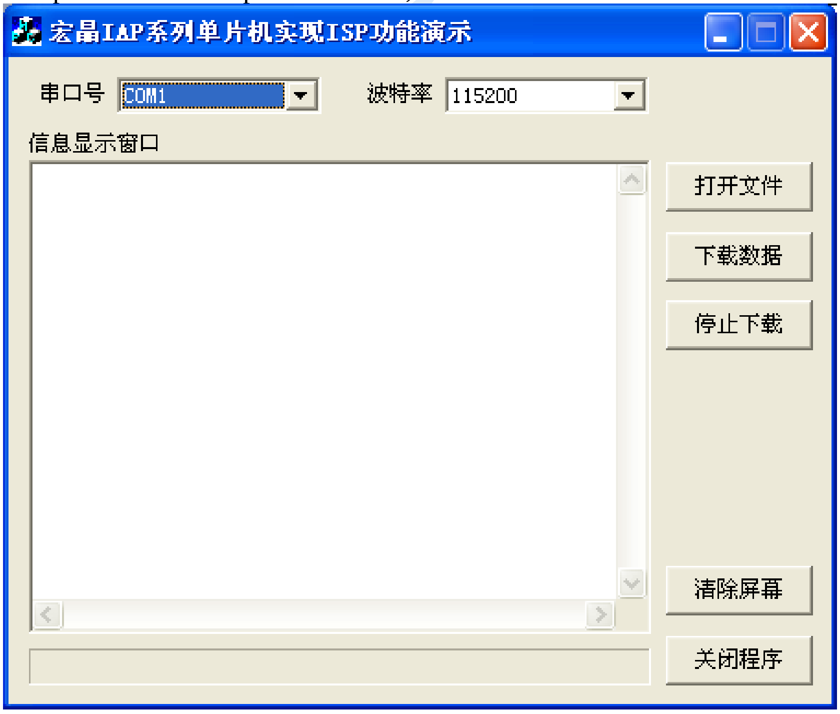
Step 4, the host computer application program description

The program of the upper computer is a dialog box project based on MFC. The access to the serial port is to directly call the API function of Windows without using the serial port control, which saves the registration of the control and many problems of system version incompatibility. The interface is relatively simple, but it provides a framework for the realization of this function. Other functions and requirements can be added in the past.

The core module of the host computer program is a friend function "UINT Download(LPVOID pParam);" based on CISPDlg,This function is responsible for communicating with the lower computer and sending various communication commands to complete the update of the user program. Users can add commands according to their different needs.

Step 5: How to use the host computer application

1. Open the host computer interface, as shown below



宏晶IAP系列单片机实现ISP功能演示

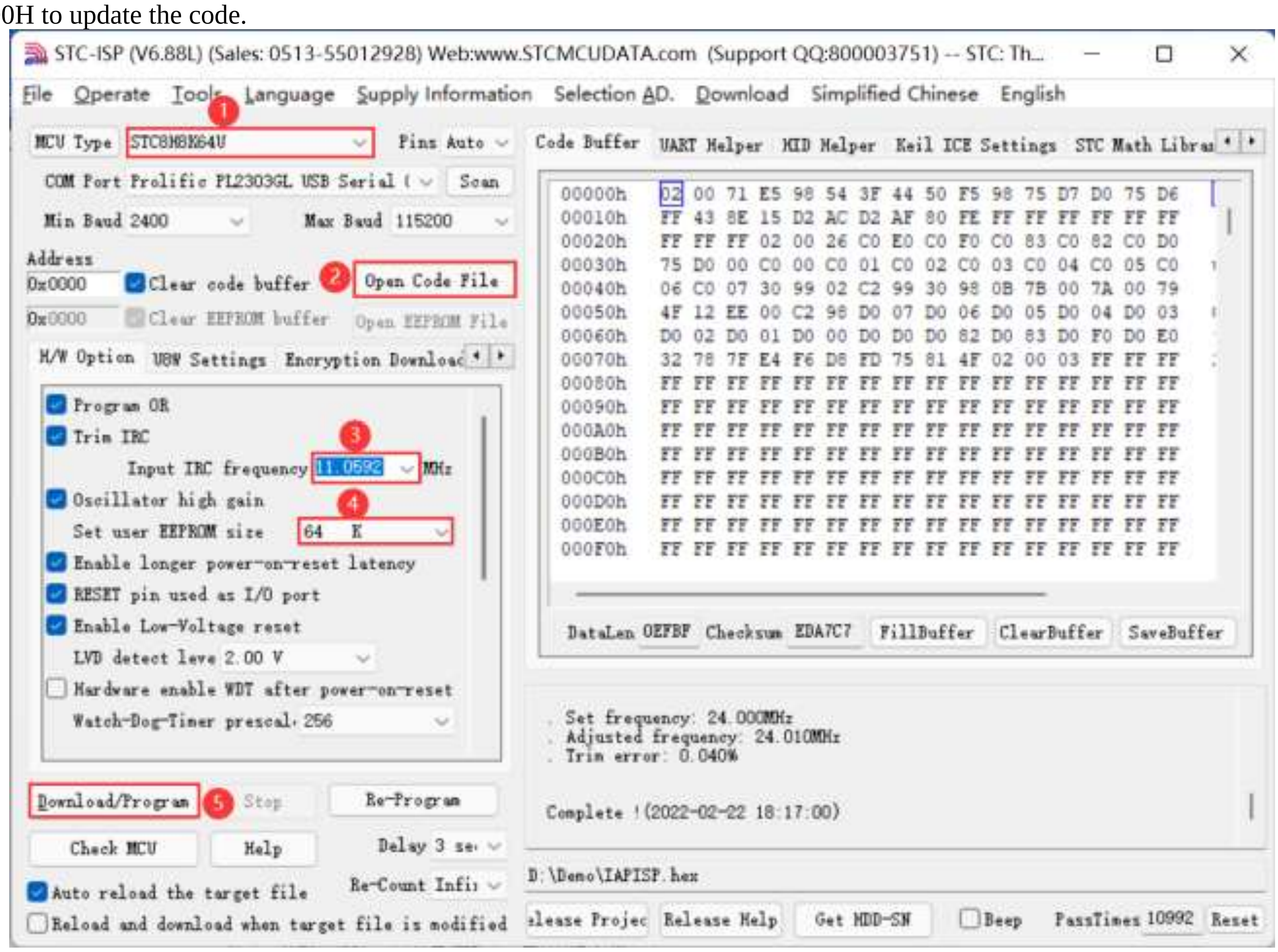
串口号 COM1

波特率 115200

2. Select the serial port number, set the same serial port baud rate as the lower computer
3. Open the source data file to be downloaded, either in Bin or Intel hex format
4. Click the "download data" button to start downloading data

The sixth step, how to use the firmware of the lower computer

The target file of the lower computer has two "IAPISP.hex" and "AP.hex". For a new single-chip computer, for the first time, you must use the ISP download tool of Acer Technology to write "IAPISP.hex" into the chip. As shown below. No need to write after updating "IAPISP.hex" is the file. The "AP.hex" in the attachment is just a template for the user program. When the download conditions are met, the user only needs to jump the PC value to the address of FA00H to update the code.



STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...

File Operate Tools Language Supply Information Selection AD. Download Simplified Chinese English
Code Buffer UART Helper HID Helper Keil ICE Settings STC Math Librar
H/W Option USB Settings Encryption Download
Scan
FillBuffer ClearBuffer SaveBuffer
Stop Re-Program Check MCU Help
Release Project Release Help Get HDD-SN Reset
1
2
3
4
5

1. MCU Type: STC8H8K64U
2. Open Code File
3. Input IRC frequency: 11.0592 MHz
4. Set user EEPROM size: 64 K
5. Download/Program

Pins: Auto

COM Port: Prolific PL2303GL USB Serial (

Min Baud: 2400 Max Baud: 115200

Address: 0x0000 ☑ Clear code buffer

0x0000 ☐ Clear EEPROM buffer Open EEPROM File

☑ Program OR
☑ Trim IRC
☑ Oscillator high gain
☑ Enable longer power-on-reset latency
☑ RESET pin used as I/O port
☑ Enable Low-Voltage reset
LVD detect leve: 2.00 V
☐ Hardware enable WDT after power-on-reset
Watch-Dog-Timer prescal: 256

Delay: 3 sec

☑ Auto reload the target file

☐ Reload and download when target file is modified

Re-Count: Infinite

☐ Beep PassTimes: 10992

```
00000h  02 00 71 E5 98 54 3F 44 50 F5 98 75 D7 D0 75 D6
00010h  FF 43 8E 15 D2 AC D2 AF 80 FE FF FF FF FF FF FF
00020h  FF FF FF 02 00 26 C0 E0 C0 F0 C0 83 C0 82 C0 D0
00030h  75 D0 00 C0 00 C0 01 C0 02 C0 03 C0 04 C0 05 C0
00040h  06 C0 07 30 99 02 C2 99 30 98 0B 7B 00 7A 00 79
00050h  4F 12 EE 00 C2 98 D0 07 D0 06 D0 05 D0 04 D0 03
00060h  D0 02 D0 01 D0 00 D0 D0 D0 82 D0 83 D0 F0 D0 E0
00070h  32 78 7F E4 F6 D8 FD 75 81 4F 02 00 03 FF FF FF
00080h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00090h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000A0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000B0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000E0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000F0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

DataLen: 0EFBF Checksum: EDA7C7

```
Set frequency: 24.000MHz
Adjusted frequency: 24.010MHz
Trim error: 0.040%

Complete !(2022-02-22 18:17:00)
```

D:\Demo\IAPISP.hex

Appendix J The method of resetting the user program to the system area for ISP download (without power off)

When the project is in the development stage, it is necessary to repeatedly download the user code to the target chip for code verification, and the STC microcontroller

For normal ISP downloads, the target chip needs to be re-powered, which will make the project development phase more cumbersome. For this reason, STC MCU has added a special function register IAP_CONTR. When the user writes 0x60 to this register, the software can be reset to the system area, and then ISP download can be performed without power failure.

But how do users judge whether ISP download is in progress? When to write 0x60 to register IAP_CONTR to trigger a soft reset? Regarding these two issues, four methods of judgment are introduced below:

Use P3.0 port to detect serial port start signal

The serial port ISP of the STC microcontroller uses P3.0 and P3.1. When the ISP download software starts to download, it will send a handshake command to the P3.0 port of the microcontroller. If the user's P3.0 and P3.1 are only used for ISP download, you can use the P3.0 port to detect the start signal of the serial port to judge the ISP download.

C Language Code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr     IAP_CONTR   =   0xc7;
sfr     P3M0        =   0xb2;
sfr     P3M1        =   0xb1;
sbit    P30         =   P3^0;

void main()
{
    P3M0 = 0x00;
    P3M1 = 0x00;
    P30 = 1;

    while (1)
    {
        if (!P30) IAP_CONTR = 0x60;                // The low level of P3.0 is the serial port start signal
                                                   // Software reset to system area

        ...                                        //User codes
    }
}
```

Use the falling edge interrupt of P3.0/INT4 to detect the serial port start signal

Method B is similar to method A, except that method A uses query mode, and method B uses interrupt mode. Because the P3.0 port of the STC single-chip computer is the interrupt port of INT4.

C Language Code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

sfr         IAP_CONTR       =     0xc7;
sfr         INTCLKO         =     0x8f;
sfr         P3M0            =     0xb2;
sfr         P3M1            =     0xb1;

void Int4Isr() interrupt 16                         //INT4 interrupt service routine
{
        IAP_CONTR = 0x60;                           // UART start signal triggers INT4 interrupt
                                                    // Software reset to system area
}

void main()
{
        P3M0 = 0x00;
        P3M1 = 0x00;

        INTCLKO |= 0x40;                            // Enable INT4 interrupt
        EA = 1;

        while (1)
        {
                ...                                 //User codes
        }
}
```

Use P3.0/RxD to receive and detect the 7F sent by the ISP download software

Method A and Method B are very simple, but easy to be interfered. If there is any interference signal on P3.0 port, it will trigger the software Reset, so method C is to verify the serial port data.

When STC's ISP download software performs ISP download, it will first use the lowest baud rate (usually 2400) + even parity 9+1 stop bit to continuously send the handshake command 7F, so the user can set the serial port to 9 in the program Bit data bit + 2400 baud rate, and then continue to detect 7F. For example, if 8 7Fs are continuously detected, it means that ISP download is required, and then a software reset is triggered.

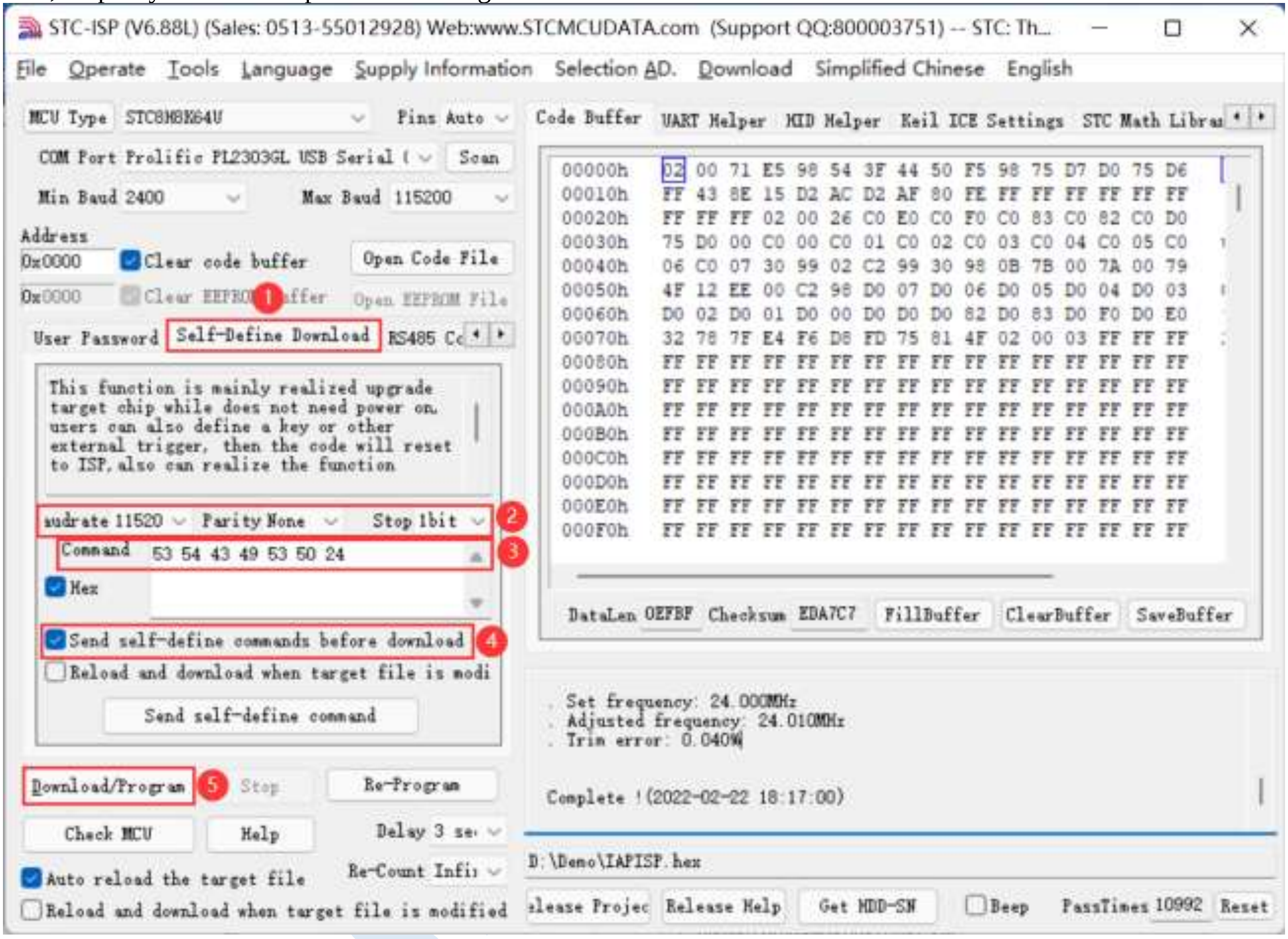
C Language Code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BR2400          (65536 - FOSC / 4 / 2400)

sfr         IAP_CONTR       =     0xc7;
sfr     AUXR        =   0x8e;
sfr     P3M0        =   0xb2;
sfr     P3M1        =   0xb1;

char cnt7f;

void UartIsr() interrupt 4                          // UART Interrupt Service Routine
{
    if (TI)
    {
        TI = 0;
    }

    if (RI)
    {
        RI = 0;
        if ((SBUF == 0x7f) && (RB8 == 1))           // Handshake command 7F sent by ISP download software
                                                    //The even parity bit of 7F is 1
        {
            if (++cnt7f == 8)                       // When 8 consecutive 7Fs are detected
                IAP_CONTR = 0x60;                   // reset to system area
        }
        else
        {
            cnt7f = 0;
        }
    }
}

void main()
{
    P3M0 = 0x00;
    P3M1 = 0x00;

    SCON = 0xd0;                                    // Set the UART to 9 data bits
    TMOD = 0x00;
    AUXR = 0x40;
    TH1 = BR2400 >> 8;                              // Set the UART baud rate to 2400
    TL1 = BR2400;
    TR1 = 1;
    ES = 1;
    EA = 1;

    cnt7f = 0;

    while (1)
    {
        ...                                         //User codes
    }
}
```

Use P3.0/RxD to receive and detect user download commands sent by ISP download software

If the user code needs to use the serial port for communication, the above 3 methods may not be applicable. At this time, you can use the interface provided by the STC ISP download software to customize a set of dedicated user download commands (you can specify the baud rate, Check bit and stop bit). If this function is enabled, the ISP download software will use the user-specified baud rate, check bit and stop bit to send the user download command before ISP download, and then send the handshake command. The user only needs to monitor the serial port command sequence in his own code. When the correct user download command is detected, the software is reset to the system area to realize the ISP function without power failure.

The following assumes that the user download command is the string "STCISP$", the serial port is set to 115200 baud rate, no parity bit and 1 stop bit. The settings in the ISP download software are as follows:



STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...

MCU Type STC8H8K64U Pins Auto
COM Port Prolific PL2303GL USB Serial (
Min Baud 2400 Max Baud 115200
Address 0x0000 ☑ Clear code buffer
0x0000 ☐ Clear EEPROM buffer

This function is mainly realized upgrade target chip while does not need power on. users can also define a key or other external trigger, then the code will reset to ISP, also can realize the function

Baudrate 11520 Parity None Stop 1bit
Command 53 54 43 49 53 50 24
☑ Hex
☑ Send self-define commands before download
☐ Reload and download when target file is modi

☑ Auto reload the target file
☐ Reload and download when target file is modified
Delay 3 sec
Re-Count Infi

```
00000h  02 00 71 E5 98 54 3F 44 50 F5 98 75 D7 D0 75 D6
00010h  FF 43 8E 15 D2 AC D2 AF 80 FE FF FF FF FF FF FF
00020h  FF FF FF 02 00 26 C0 E0 C0 F0 C0 83 C0 82 C0 D0
00030h  75 D0 00 C0 00 C0 01 C0 02 C0 03 C0 04 C0 05 C0
00040h  06 C0 07 30 99 02 C2 99 30 98 0B 7B 00 7A 00 79
00050h  4F 12 EE 00 C2 98 D0 07 D0 06 D0 05 D0 04 D0 03
00060h  D0 02 D0 01 D0 00 D0 D0 D0 82 D0 83 D0 F0 D0 E0
00070h  32 78 7F E4 F6 D8 FD 75 81 4F 02 00 03 FF FF FF
00080h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00090h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000A0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000B0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000C0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000D0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000E0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000F0h  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

DataLen 0EFBF Checksum EDA7C7

```
Set frequency: 24.000MHz
Adjusted frequency: 24.010MHz
Trim error: 0.040%

Complete !(2022-02-22 18:17:00)
```

D:\Demo\IAPISP.hex

☐ Beep PassTimes 10992

User sample code is as follows:

C Language Code

```
// Operating frequency for test is 11.0592MHz

#include "reg51.h"
#include "intrins.h"

#define     FOSC            11059200UL
#define     BR115200        (65536 - FOSC / 4 / 115200)

sfr         IAP_CONTR   =   0xc7;
sfr         AUXR        =   0x8e;
sfr         P3M0        =   0xb2;
sfr         P3M1        =   0xb1;

char stage;

void UartIsr() interrupt 4                                      // UART Interrupt Service Routine
{
        char dat;
```

```
    if (TI)
    {
        TI = 0;
    }

    if (RI)
    {
        RI = 0;

        dat = SBUF;
        switch (stage)
        {
        case 0:
        default:
L_Check1st:
            if (dat == 'S') stage = 1;
            else stage = 0;
            break;
        case 1:
            if (dat == 'T') stage = 2;
            else goto L_Check1st;
            break;
        case 2:
            if (dat == 'C') stage = 3;
            else goto L_Check1st;
            break;
        case 3:
            if (dat == 'I') stage = 4;
            else goto L_Check1st;
            break;
        case 4:
            if (dat == 'S') stage = 5;
            else goto L_Check1st;
            break;
        case 5:
            if (dat == 'P') stage = 6;
            else goto L_Check1st;
            break;
        case 6:
            if (dat == '$')                     // When the correct user download command is detected
                IAP_CONTR = 0x60;               // reset to system area
            else goto L_Check1st;
        break;
        }
    }
}

void main()
{
    P3M0 = 0x00;
    P3M1 = 0x00;

    SCON = 0x50;                                // Set the UART to 8 data bits
    TMOD = 0x00;
    AUXR = 0x40;
    TH1 = BR2400 >> 8;                          // Set the UART baud rate to 115200
    TL1 = BR2400;
    TR1 = 1;
```

```
        ES = 1;
        EA = 1;

        stage = 0;

        while (1)
        {
                ...                                                     // User codes
        }
}
```

Appendix N Example Routine of ISP download for STC8H series MCUs using third-party MCU

C language code

```
/*Note: When using this code to download the STC8H series of microcontrollers, you must execute the Download code before powering on the target chip, otherwise the target chip will not download correctly.*/

#include "reg51.h"

typedef     bit                 BOOL;
typedef     unsigned char       BYTE;
typedef     unsigned short      WORD;

//Macro and constant definition
#define     FALSE               0
#define     TRUE                1
#define     LOBYTE(w)           ((BYTE)(WORD)(w))
#define     HIBYTE(w)           ((BYTE)((WORD)(w) >> 8))

#define     MINBAUD             2400L
#define     MAXBAUD             115200L

#define     FOSC                11059200L               //Main chip working frequency
#define     BR(n)               (65536 - FOSC/4/(n))    // Calculation formula of serial port baud rate of main chip
#define     T1MS                (65536 - FOSC/1000)     // 1ms timing initial value of main chip

#define     FUSER               24000000L               //STC8H Series target chip operating frequency
#define     RL(n)               (65536 - FUSER/4/(n))   //STC8H Serial target chip baud rate calculation formula

sfr         AUXR = 0x8e;
sfr         P3M1 = 0xB1;
sfr         P3M0 = 0xB2;

// Variable definitions
BOOL f1ms;                          //1ms flag
BOOL UartBusy;                      // Serial transmit busy flag
BOOL UartReceived;                  // Serial data receiving completion flag
BYTE UartRecvStep;                  // Serial data receiving control
BYTE TimeOut;                       // Serial communication timeout counter
BYTE xdata TxBuffer[256];           // Serial data transmission buffer
BYTE xdata RxBuffer[256];           // Serial data receiving buffer
char code DEMO[256];                // Demo code data
```

```
// Functions declarations
void Initial(void);
void DelayXms(WORD x);
BYTE UartSend(BYTE dat);
void CommInit(void);
void CommSend(BYTE size);
BOOL Download(BYTE *pdat, long size);

// Main function entry
void main(void)
{
        P3M0 = 0x00;
        P3M1 = 0x00;

        Initial();
        if (Download(DEMO, 256))
        {
                // download successfully
                P3 = 0xff;
                DelayXms(500);
                P3 = 0x00;
                DelayXms(500);
                P3 = 0xff;
                DelayXms(500);
                P3 = 0x00;
                DelayXms(500);
                P3 = 0xff;
                DelayXms(500);
                P3 = 0x00;
                DelayXms(500);
                P3 = 0xff;
        }
        else
        {
                // download failed
                P3 = 0xff;
                DelayXms(500);
                P3 = 0xf3;
                DelayXms(500);
                P3 = 0xff;
                DelayXms(500);
                P3 = 0xf3;
                DelayXms(500);
                P3 = 0xff;
                DelayXms(500);
                P3 = 0xf3;
                DelayXms(500);
                P3 = 0xff;
        }

        while (1);
}

//1ms Timer interrupt service routine
void tm0(void) interrupt 1
{
        static BYTE Counter100;
        f1ms = TRUE;
        if (Counter100-- == 0)
        {
                Counter100 = 100;
                if (TimeOut) TimeOut--;
        }
}

// Serial port interrupt service routine
void uart(void) interrupt 4
{
        static WORD RecvSum;
        static BYTE RecvIndex;
        static BYTE RecvCount;
        BYTE dat;

        if (TI)
        {
                TI = 0;
                UartBusy = FALSE;
        }

        if (RI)
        {
                RI = 0;
                dat    = SBUF;
                switch (UartRecvStep)
                {
                case 1:
                        if (dat != 0xb9) goto L_CheckFirst;
                        UartRecvStep++;
                        break;
                case 2:
                        if (dat != 0x68) goto L_CheckFirst;
                        UartRecvStep++;
                        break;
                case 3:
                        if (dat != 0x00) goto L_CheckFirst;
                        UartRecvStep++;
                        break;
                case 4:
                        RecvSum = 0x68 + dat;
                        RecvCount = dat - 6;
                        RecvIndex = 0;
                        UartRecvStep++;
                        break;
                case 5:
                        RecvSum += dat;
                        RxBuffer[RecvIndex++] = dat;
                        if (RecvIndex == RecvCount)     UartRecvStep++;
                        break;
                case 6:
                        if (dat != HIBYTE(RecvSum))     goto L_CheckFirst;
                        UartRecvStep++;
                        break;
                case 7:
                        if (dat != LOBYTE(RecvSum))     goto L_CheckFirst;
                        UartRecvStep++;
                        break;
                case 8:
                        if (dat != 0x16) goto L_CheckFirst;
                        UartReceived = TRUE;
                        UartRecvStep++;
                        break;
L_CheckFirst:
                case 0:
                default:
                        CommInit();
                        UartRecvStep = (dat == 0x46      ? 1 : 0);
                        break;
                }
        }
}

// system initialization
void Initial(void)
{
        UartBusy = FALSE;

        SCON = 0xd0;                            // Serial data format must be 8-bit data + 1-bit even check
        AUXR = 0xc0;
        TMOD = 0x00;
        TH0 = HIBYTE(T1MS);
        TL0 = LOBYTE(T1MS);
        TR0 = 1;
        TH1 = HIBYTE(BR(MINBAUD));
        TL1 = LOBYTE(BR(MINBAUD));
        TR1 = 1;
        ET0 = 1;
        ES = 1;
        EA = 1;
}

//Xms Delay program
void DelayXms(WORD x)
{
        do
        {
                f1ms = FALSE;
                while (!f1ms);
        } while (x--);
}

// Serial data sending program
BYTE UartSend(BYTE dat)
{
        while (UartBusy);

        UartBusy = TRUE;
        ACC = dat;
        TB8 = P;
        SBUF = ACC;

        return dat;
}
```

```
// Serial communication initialization
void CommInit(void)
{
    UartRecvStep = 0;
    TimeOut = 20;
    UartReceived = FALSE;
}

// Send serial communication packets
void CommSend(BYTE size)
{
    WORD sum;
    BYTE i;

    UartSend(0x46);
    UartSend(0xb9);
    UartSend(0x6a);
    UartSend(0x00);
    sum = size + 6 + 0x6a;
    UartSend(size + 6);
    for (i=0; i<size; i++)
    {
        sum  += UartSend(TxBuffer[i]);
    }
    UartSend(HIBYTE(sum));
    UartSend(LOBYTE(sum));
    UartSend(0x16);
    while (UartBusy);

    CommInit();
}

//对 STC15H Series of chips for ISP download
BOOL Download(BYTE *pdat, long size)
{
    BYTE arg;
    BYTE offset;
    BYTE cnt;
    WORD addr;

    // Shake hands
    CommInit();
    while (1)
    {
        if (UartRecvStep == 0)
        {
            UartSend(0x7f);
            DelayXms(10);
        }
        if (UartReceived)
        {
            arg = RxBuffer[4];
            if (RxBuffer[0] == 0x50) break;
            return FALSE;
        }
    }

    // Set parameters (set the parameters such as the highest baud rate used by the slave chip and erase wait time)
    TxBuffer[0] = 0x01;
    TxBuffer[1] = arg;
    TxBuffer[2] = 0x40;
    TxBuffer[3] = HIBYTE(RL(MAXBAUD));
    TxBuffer[4] = LOBYTE(RL(MAXBAUD));
    TxBuffer[5] = 0x00;
    TxBuffer[6] = 0x00;
    TxBuffer[7] = 0x97;
    CommSend(8);
    while (1)
    {
        if (TimeOut == 0) return FALSE;
        if (UartReceived)
        {
            if (RxBuffer[0] == 0x01) break;
            return FALSE;
        }
    }

    //prepare
    TH1 = HIBYTE(BR(MAXBAUD));
    TL1 = LOBYTE(BR(MAXBAUD));
    DelayXms(10);
    TxBuffer[0] = 0x05;
    TxBuffer[1] = 0x00;
    TxBuffer[2] = 0x00;
    TxBuffer[3] = 0x5a;
    TxBuffer[4] = 0xa5;
    CommSend(5);
    while (1)
    {
        if (TimeOut == 0) return FALSE;
        if (UartReceived)
        {
            if (RxBuffer[0] == 0x05) break;
            return FALSE;
        }
    }

    // Erase
    DelayXms(10);
    TxBuffer[0] = 0x03;
    TxBuffer[1] = 0x00;
    TxBuffer[2] = 0x00;
    TxBuffer[3] = 0x5a;
    TxBuffer[4] = 0xa5;
    CommSend(5);
    TimeOut = 100;
    while (1)
    {
        if (TimeOut == 0) return FALSE;
        if (UartReceived)
        {
            if (RxBuffer[0] == 0x03) break;
            return FALSE;
        }
    }
```

```
// Write user code
DelayXms(10);
addr = 0;
TxBuffer[0] = 0x22;
TxBuffer[3] = 0x5a;
TxBuffer[4] = 0xa5;
offset = 5;
while (addr < size)
{
        TxBuffer[1] = HIBYTE(addr);
        TxBuffer[2] = LOBYTE(addr);
        cnt    = 0;
        while (addr < size)
        {
                TxBuffer[cnt+offset] = pdat[addr];
                addr++;
                cnt++;
                if (cnt >= 128) break;
        }
        CommSend(cnt + offset);
        while (1)
        {
                if (TimeOut == 0) return FALSE;
                if (UartReceived)
                {
                        if ((RxBuffer[0] == 0x02) && (RxBuffer[1] == 'T')) break;
                        return FALSE;
                }
        }
        TxBuffer[0] = 0x02;
}

//// Write hardware options
//// If you do not need to modify the hardware options, this step can be skipped directly.At this time, all the hardware options
//// remain unchanged, the frequency of the MCU is the last adjusted frequency
//// If you write the hardware option, the MCU's internal IRC frequency will be fixed to 24MHz,
//// and other options will be restored to the factory settings.
////Suggestion: Set the hardware options of the slave chip when you use STC-ISP download software the first time.

//// Do not write hardware options when downloading programs from the master chip to the slave chip.
//DelayXms(10);
//for (cnt=0; cnt<128; cnt++)
//{
//      TxBuffer[cnt] = 0xff;
//}
//TxBuffer[0] = 0x04;
//TxBuffer[1] = 0x00;
//TxBuffer[2] = 0x00;
//TxBuffer[3] = 0x5a;
//TxBuffer[4] = 0xa5;
//TxBuffer[33] = arg;
//TxBuffer[34] = 0x00;
//TxBuffer[35] = 0x01;
//TxBuffer[41] = 0xbf;
//TxBuffer[42] = 0xbd;                      //P5.4 is I/O port
////TxBuffer[42] = 0xad;                    //P5.4 is reset pin
//TxBuffer[43] = 0xf7;
//TxBuffer[44] = 0xff;
```

```
        //CommSend(45);
        //while (1)
        //{
        //      if (TimeOut == 0) return FALSE;
        //      if (UartReceived)
        //      {
        //              if ((RxBuffer[0] == 0x04) && (RxBuffer[1] == 'T')) break;
        //              return FALSE;
        //      }
        //}

        // Download completed
        return TRUE;
}

char code DEMO[256] =
{
        0x80,0x00,0x75,0xB2,0xFF,0x75,0xB1,0x00,0x05,0xB0,0x11,0x0E,0x80,0xFA,0xD8,0xFE,
        0xD9,0xFC,0x22,
};
```

Note: If user needs to set different working frequencies, please refer to the example codes in chapters 7.3.7 and 7.3.8.

Appendix O Use a third-party application program to call the STC release project program to download the ISP of the MCU

The release project program generated by STC's ISP download software is an executable EXE format file. The user can directly double-click the released project program to run it for ISP download, or call the release project program in a third-party application for ISP download. Two methods of calling are introduced below.

Simple call

In the third-party application, it is only a simple process of creating and publishing the project program. All other download operations are carried out in the publishing project program. The third-party application only needs to wait for the completion of the publishing project program and clean the scene.

VC code

```
BOOL IspProcess()
{
        // define related variables
        STARTUPINFO si;
        PROCESS_INFORMATION pi;
        CString path;

        // Full path of publishing project program
        path = _T("D:\\Work\\Upgrade.exe");

        // variable initialization
        memset(&si, 0, sizeof(STARTUPINFO));
        memset(&pi, 0, sizeof(PROCESS_INFORMATION));

        // Set startup variables
        si.cb = sizeof(STARTUPINFO);
        GetStartupInfo(&si);
        si.wShowWindow = SW_SHOWNORMAL;
        si.dwFlags = STARTF_USESHOWWINDOW;

        // Create Publish Project Program Process
        if (CreateProcess(NULL, (LPTSTR)(LPCTSTR)path, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
        {
                // Wait for the publish project program operation to complete
                // Since the main process will be blocked here, it is recommended to create a new working process and wait in the worker process
                WaitForSingleObject(pi.hProcess,INFINITE);
        // clean up
        CloseHandle(pi.hThread);
        CloseHandle(pi.hProcess);

        return TRUE;
    }
    else
    {
        AfxMessageBox(_T("创建进程失败!"));

        return FALSE;
    }
}
```

Advanced call

The process of creating and publishing project programs in third-party applications, including selecting serial ports and starting ISP in third-party applications

All ISP download operations such as programming, ISP programming with vibration stopped, and closing the release project program, do not require interface interaction in the release project program.

VC 代码

```
// A data structure that defines the parameters of the callback function
struct CALLBACK_PARAM
{
    DWORD dwProcessId;                          // main process id
    HWND hMainWnd;                              // main window handle
};

// Callback function for enumerating windows to get the main window handle
BOOL CALLBACK EnumWindowCallBack(HWND hWnd, LPARAM lParam)
{
    CALLBACK_PARAM *pcp = (CALLBACK_PARAM *)lParam;
    DWORD id;

    GetWindowThreadProcessId(hWnd, &id);
    if ((pcp->dwProcessId == id) && (GetParent(hWnd) == NULL))
    {
        pcp->hMainWnd = hWnd;

        return FALSE;
    }

    return TRUE;
}

BOOL IspProcess()
{
    // define related variables
    STARTUPINFO si;
    PROCESS_INFORMATION pi;
    CALLBACK_PARAM cp;
    CString path;

    // the IDs of some controls in the publishing project program
    const UINT ID_PROGRAM       = 1046;
    const UINT ID_STOP          = 1044;
    const UINT ID_COMPORT       = 1009;
    const UINT ID_PROGRESS      = 1044;

    // Full path of publishing project program
    path = _T("D:\\Work\\Upgrade.exe");

    // variable initialization
    memset(&si, 0, sizeof(STARTUPINFO));
    memset(&pi, 0, sizeof(PROCESS_INFORMATION));
    memset(&cp, 0, sizeof(CALLBACK_PARAM));

    // Set startup variables
    si.cb = sizeof(STARTUPINFO);
    GetStartupInfo(&si);
    si.wShowWindow = SW_SHOWNORMAL;                 // If it is set to SW_HIDE here, the operation interface for
publishing the project program will not be displayed, and all ISP operations can be performed in the background

    si.dwFlags = STARTF_USESHOWWINDOW;

    // Create a process for publishing a project program
    if (CreateProcess(NULL, (LPTSTR)(LPCTSTR)path, NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
    {
        // Wait for the release project program process initialization to complete
        WaitForInputIdle(pi.hProcess, 5000);

        // Get the main window handle of the publishing project program
        cp.dwProcessId = pi.dwProcessId;
        cp.hMainWnd = NULL;
        EnumWindows(EnumWindowCallBack, (LPARAM)&cp);

        if (cp.hMainWnd != NULL)
        {
            HWND hProgram;
            HWND hStop;
            HWND hPort;

            // Get the handle of some controls in the main window of the publishing project program
            hProgram = ::GetDlgItem(cp.hMainWnd, ID_PROGRAM);
            hStop = ::GetDlgItem(cp.hMainWnd, ID_STOP);
            hPort = ::GetDlgItem(cp.hMainWnd, ID_COMPORT);

            // Set the serial port number in the release project program, the third parameter is 0:COM1, 1:COM2, 2:COM3, ...
            ::SendMessage(hPort, CB_SETCURSEL, 0, 0);

            // Trigger the programming button to start ISP programming
            ::SendMessage(hProgram, BM_CLICK, 0, 0);

            // wait for programming to complete,
            // Since the main process will be blocked here, it is recommended to create a new working process and wait in the
worker process
            while (!::IsWindowEnabled(hProgram));

            // Close the release project program after programming is complete
            ::SendMessage(cp.hMainWnd, WM_CLOSE, 0, 0);
        }

        // wait for the process to end
        WaitForSingleObject(pi.hProcess,INFINITE);
```

```
                // clean up
                CloseHandle(pi.hThread);
                CloseHandle(pi.hProcess);

                return TRUE;
        }
        else
        {
                AfxMessageBox(_T("创建进程失败!"));

                return FALSE;
        }
}
```

Appendix P STC8H series orthogonal decoding example (courtesy of Chengdu Zhufei Technology)

Entrusted by STC, this article intends to share the use of the enhanced PWM module of the STC8H series of single-chip microcomputers to realize the orthogonal decoding function, so as to further realize the two-way speed measurement of the encoder output by the orthogonal code signal.

Hardware platform: Zhufei STC8H8K-64 pin core board + Zhufei 1024 line mini quadrature encoder

Compilation environment: keil V9.60

Host computer: serial port assistant

From the previous open source library of Zhufei Technology, we can understand that there is no orthogonal decoding routine in the open source library. The main reason is that STC8H only has two PWM modules. If we recommend using an orthogonal encoder, it means one encoding. However, this year’s energy-saving group requires a balanced car to be made, which means that there are two motors, so two encoders are needed. Then the two PWM modules of the microcontroller will be occupied. However, the motor of the car Control also requires PWM function, so it is not recommended that you use the PWM module to achieve quadrature decoding. Instead, it is recommended that you use an encoder with directional output, so that the PWM module can be used by the motor.

Of course, there can also be another idea. It is feasible to use a PWM module to realize the speed measurement of an encoder with quadrature encoding, another motor uses an encoder with a direction signal, and an ordinary timer to capture pulses. Yes, but there is no need to be so troublesome.

One more thing to note is that the method of using the PWM module to count and using the timer module to capture pulse count is different. The PWM module captures the encoder data by counting the edges, which means that this module counts when a rising or falling edge occurs, and the timer captures pulses to obtain the number of high and low level flips. Here we will find through experiments that using the same quadrature encoder to rotate 360 °, the encoder data collected by the PWM module is twice the pulse data captured by the timer, but this data does not increase the accuracy, but the count of the single-chip microcomputer. The way led to double the result.

The following is an example program of using STC8H8K64U to collect quadrature coded signal output encoder:

C Language Code

```
#include "headfile.h"

int16 encoder_data;

//-------------------------------------------------------------------
//@brief          PWMA Module Orthogonal Decoding Initialization
//@param          void
//@return         void
//@since          v1.0
//Sample usage:  PWMA_encoder_init();                     // Initialize Orthogonal Decoding
//@note
//-------------------------------------------------------------------
void PWMA_encoder_init(void)
{
        P_SW2 |= 1<<7;                                    // Enable access to XFR
        PWMA_ARR = 0xFFFF;                                // Set the auto-reload value. When the auto-reload value is 0, the
counter does not work.

        PWMA_CCMR1 |= 1<<0;                               // IC1 is mapped on TI1FP1, i.e. use the P10 pin to get the
direction
        PWMA_CCMR2 |= 1<<0;                               // IC2 is mapped on TI2FP2, that is, using the P22 pin to capture
edge transitions
        PWMA_SMCR |= 1<<0;                                // Encoder mode 1, according to the level of TI1FP1, the counter
counts up/down on the edge of TI2FP2
        PWMA_CR1 |= 1<<0;                                 // Enable PWMA counter
        PWMA_PS |= 1<<2;                                  //Channel 1 of PWMA uses P10, and channel 2 of PWMB uses
P22.
}

//-------------------------------------------------------------------
// @brief         PWMA module obtains the quadrature decoding value
// @param         void
// @return        void
// @since         v1.0
// Sample usage: encoder_data = PWMA_get_encoder();       // Get the quadrature decoding value
// @note
//-------------------------------------------------------------------
int16 PWMA_get_encoder(void)
{
        int16 res;

        res = PWMA_CNTR;                                  // Save the current counter value
        PWMA_CNTR = 0;                                    // clear counter
        return res;
}

//-------------------------------------------------------------------
// @brief         Interrupt service function of timer 0 5ms
// @param         void
// @return        void
// @since         v1.0
// Sample usage:
```

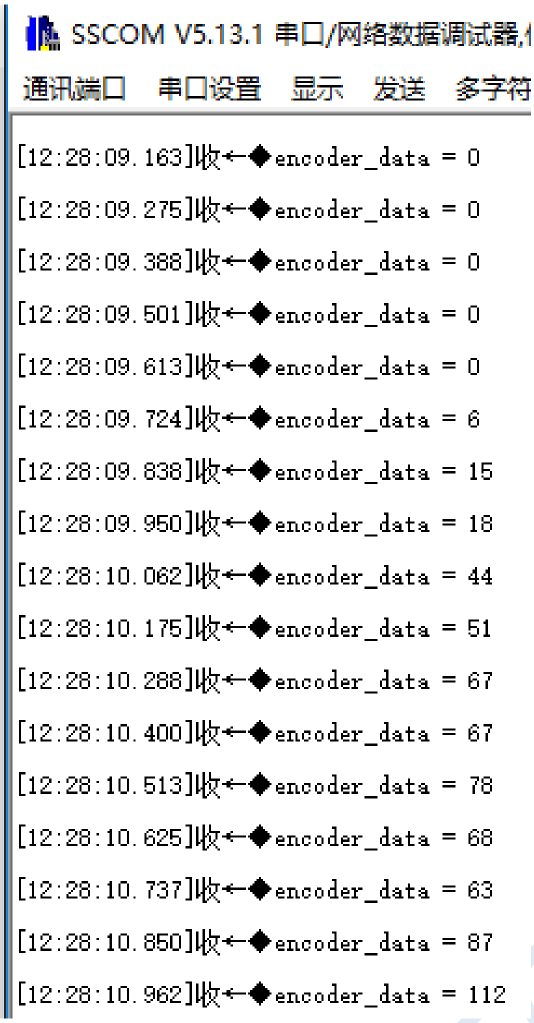
```
// @note
//-------------------------------------------------------------------
void TM0_Isr() interrupt 1
{
        encoder_data = PWMA_get_encoder();                  // Get the quadrature decoding encoder value
}

void main()
{
        DisableGlobalIRQ();                                  // Disable CPU interrupt
        board_init();                                        // Initialize internal registers, do not delete this code.
        pit_timer_ms(TIM_0, 5);                              // Initialize the timer, execute once in 5ms
        PWMA_encoder_init();                                 // The PWMA module is initialized to the quadrature decoding
function
        EnableGlobalIRQ();                                   //Enable CPU interrupt
        while(1)
        {
                delay_ms(100);                               // Output printing information every 100ms
                printf("encoder_data = %d \r\n" ,encoder_data);   // UART 1 prints encoder data
        }
}
```

Demo video link: https://www.bilibili.com/video/BV1zT4y177Ht

Video description: We compile the written routine, then download it to the microcontroller, open the serial port assistant to receive the print data of the microcontroller, rotate the encoder to observe the data changes, we find that the output data is 0 when the encoder is not rotating, and when the encoder faces Two kinds of values can be output when rotating in different directions. The faster the rotation, the greater the absolute value of the value. Positive and negative are used to indicate the two rotation directions. Which direction is positive and which direction is negative can be defined by yourself. At the same time, we also see from the program example that the print data is 100ms once, and the data collection is 5ms once, so the printed data is equivalent to intermittent. At the same time, because the encoder has a high precision of 1024 lines, it is observed that the data changes relatively large. , But if the motor is driven with no-load fixed PWM duty cycle, you can see that the encoder's data output is very stable.

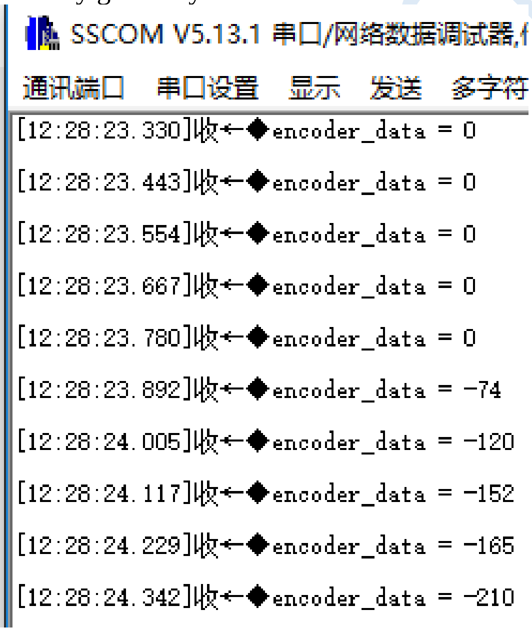
Screenshot of serial port assistant receiving data:

The figure below is the data of the quadrature-encoded encoder rotating clockwise and the angular velocity gradually increasing



```
[12:28:09.163]收←◆encoder_data = 0
[12:28:09.275]收←◆encoder_data = 0
[12:28:09.388]收←◆encoder_data = 0
[12:28:09.501]收←◆encoder_data = 0
[12:28:09.613]收←◆encoder_data = 0
[12:28:09.724]收←◆encoder_data = 6
[12:28:09.838]收←◆encoder_data = 15
[12:28:09.950]收←◆encoder_data = 18
[12:28:10.062]收←◆encoder_data = 44
[12:28:10.175]收←◆encoder_data = 51
[12:28:10.288]收←◆encoder_data = 67
[12:28:10.400]收←◆encoder_data = 67
[12:28:10.513]收←◆encoder_data = 78
[12:28:10.625]收←◆encoder_data = 68
[12:28:10.737]收←◆encoder_data = 63
[12:28:10.850]收←◆encoder_data = 87
[12:28:10.962]收←◆encoder_data = 112
```

The following figure shows the data when the quadrature-encoded encoder rotates counterclockwise and the angular velocity gradually increases:

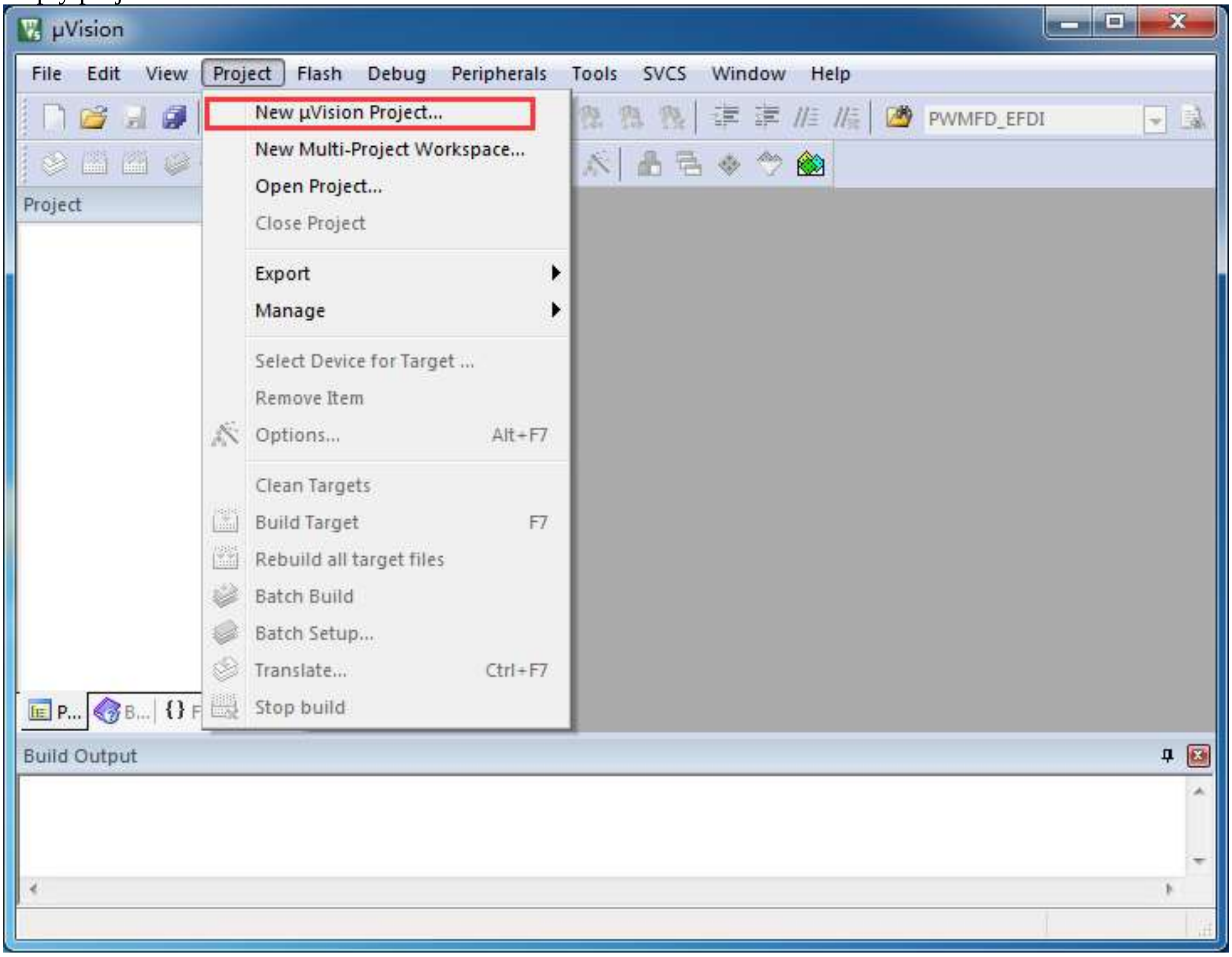


```
[12:28:23.330]收←◆encoder_data = 0
[12:28:23.443]收←◆encoder_data = 0
[12:28:23.554]收←◆encoder_data = 0
[12:28:23.667]收←◆encoder_data = 0
[12:28:23.780]收←◆encoder_data = 0
[12:28:23.892]收←◆encoder_data = -74
[12:28:24.005]收←◆encoder_data = -120
[12:28:24.117]收←◆encoder_data = -152
[12:28:24.229]收←◆encoder_data = -165
[12:28:24.342]收←◆encoder_data = -210
```

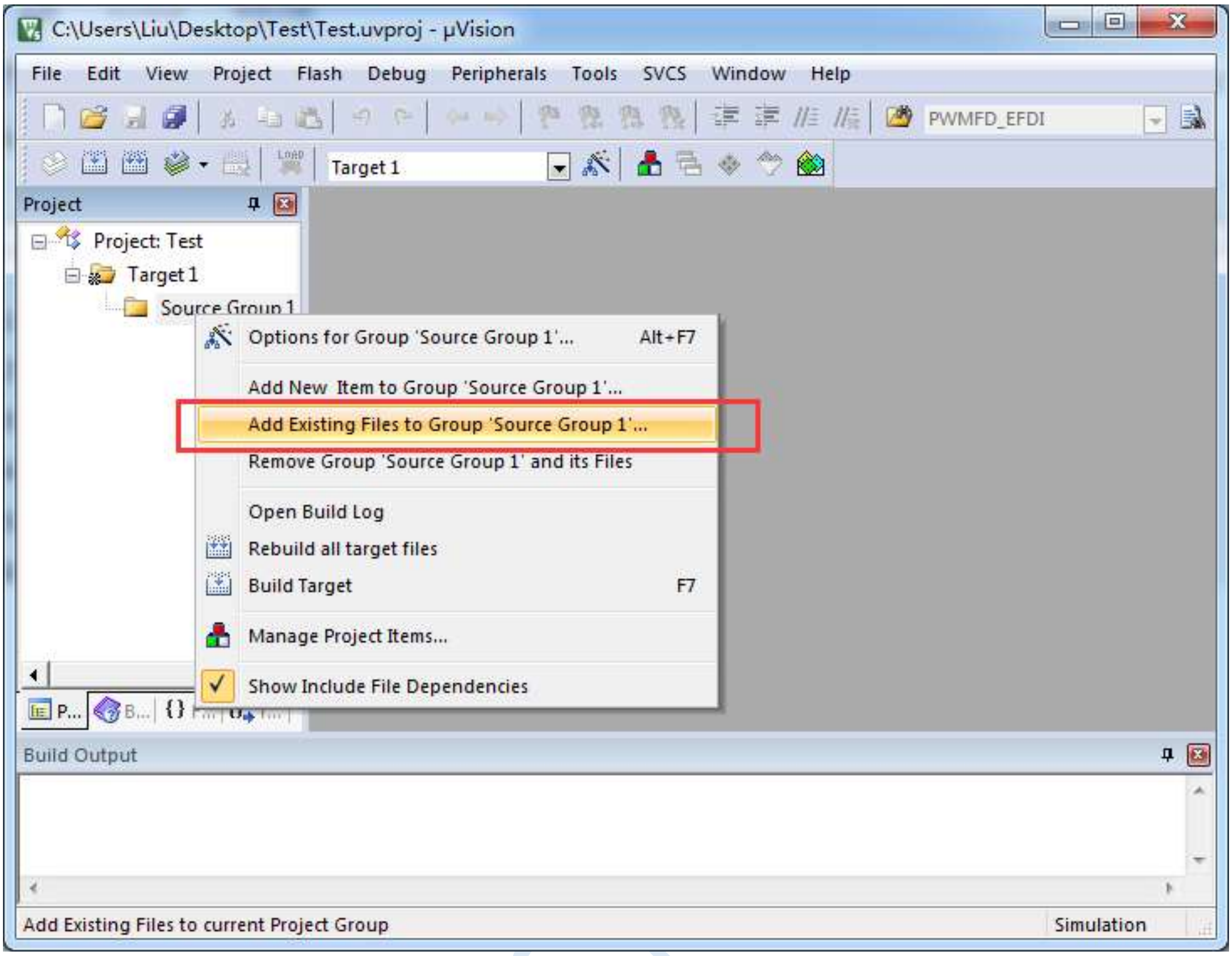
Appendix Q Method for Creating Multi-file Projects in Keil

In Keil, relatively small projects generally have only one source file, but for some slightly more complex projects, multiple source files are often required. Here's how to set up a multi-file project:

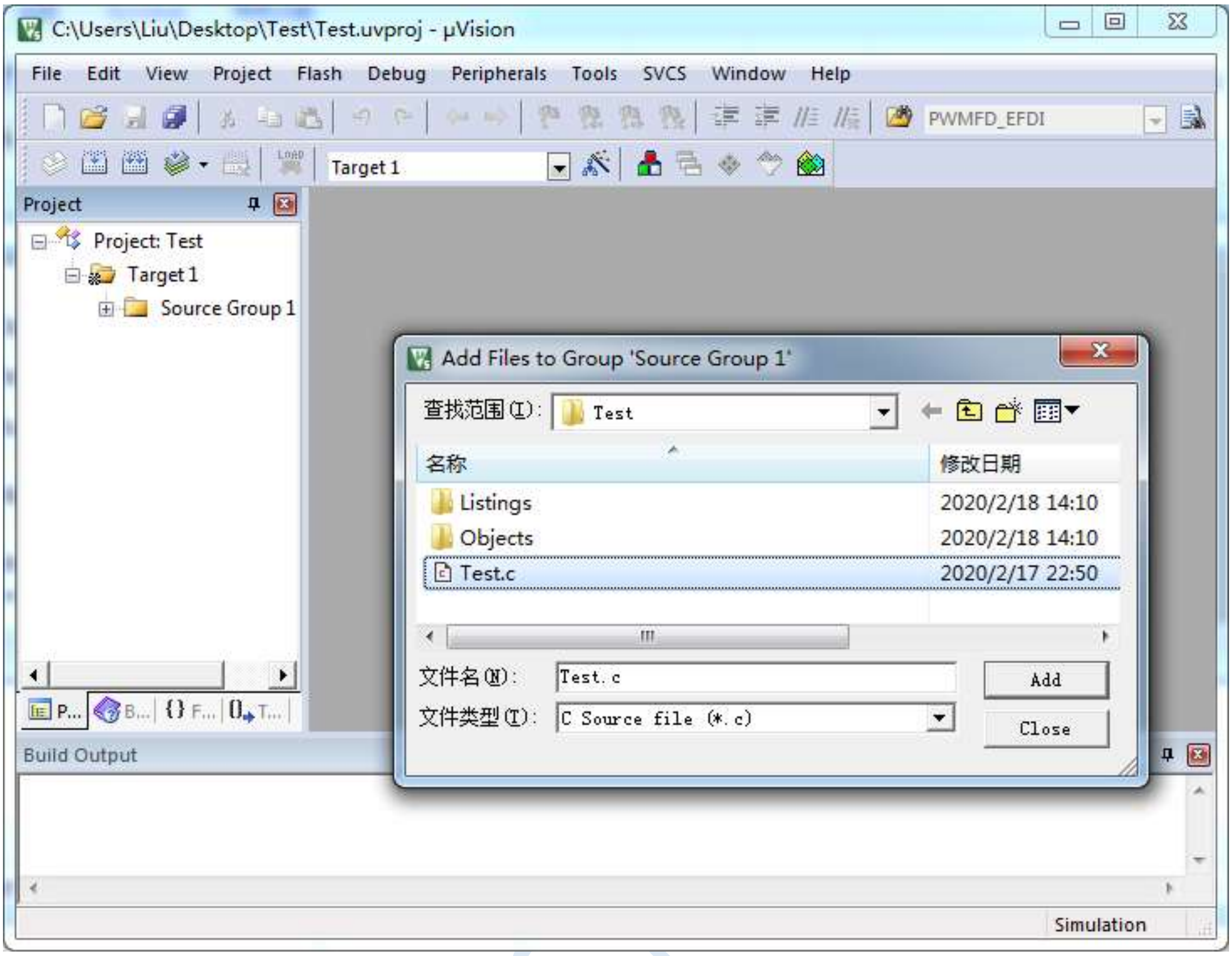
1. Open Keil firstly and select "New uVision Project ..." from the "Project" menu to complete the creation of an empty project.



2. In the project tree of the empty project, right-click "Source Group 1" and select "Add Existing Files to Group" Source Group 1 "..." from the right-click menu.


C:\Users\Liu\Desktop\Test\Test.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
PWMFD_EFDI
Target 1
Project
Project: Test
Target 1
Source Group 1
Options for Group 'Source Group 1'... Alt+F7
Add New Item to Group 'Source Group 1'...
Add Existing Files to Group 'Source Group 1'...
Remove Group 'Source Group 1' and its Files
Open Build Log
Rebuild all target files
Build Target F7
Manage Project Items...
Show Include File Dependencies
P...
B...
{} F...
0+ T...
Build Output
Add Existing Files to current Project Group
Simulation

3. In the file dialog that pops up, add the source file multiple times.


C:\Users\Liu\Desktop\Test\Test.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
PWMFD_EFDI
Target 1
Project
Project: Test
Target 1
Source Group 1
P... B... {} F... 0→ T...
Build Output
Simulation

Add Files to Group 'Source Group 1'

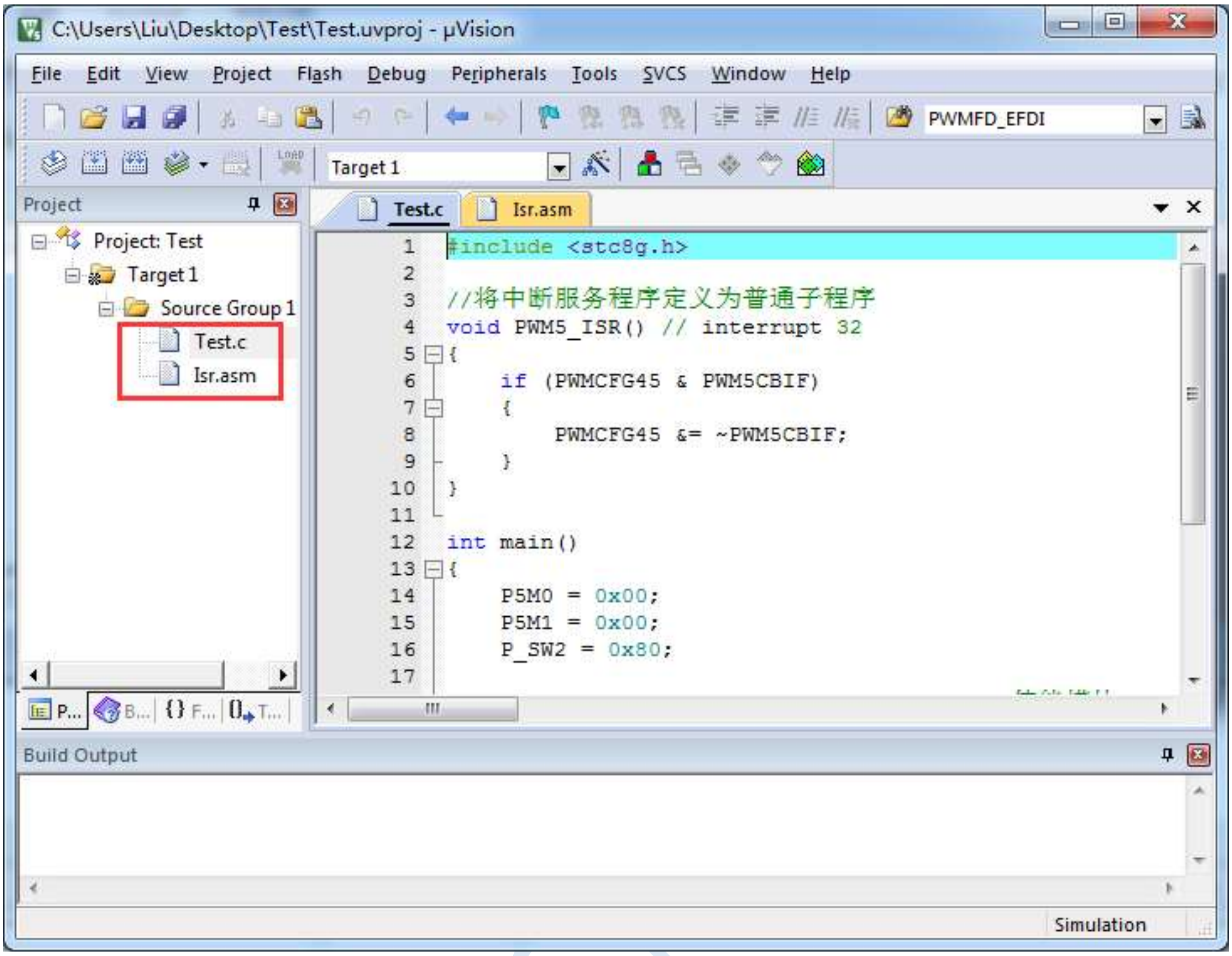
查找范围(I): Test

名称	修改日期
Listings	2020/2/18 14:10
Objects	2020/2/18 14:10
Test.c	2020/2/17 22:50

文件名(N): Test.c

文件类型(T): C Source file (*.c)

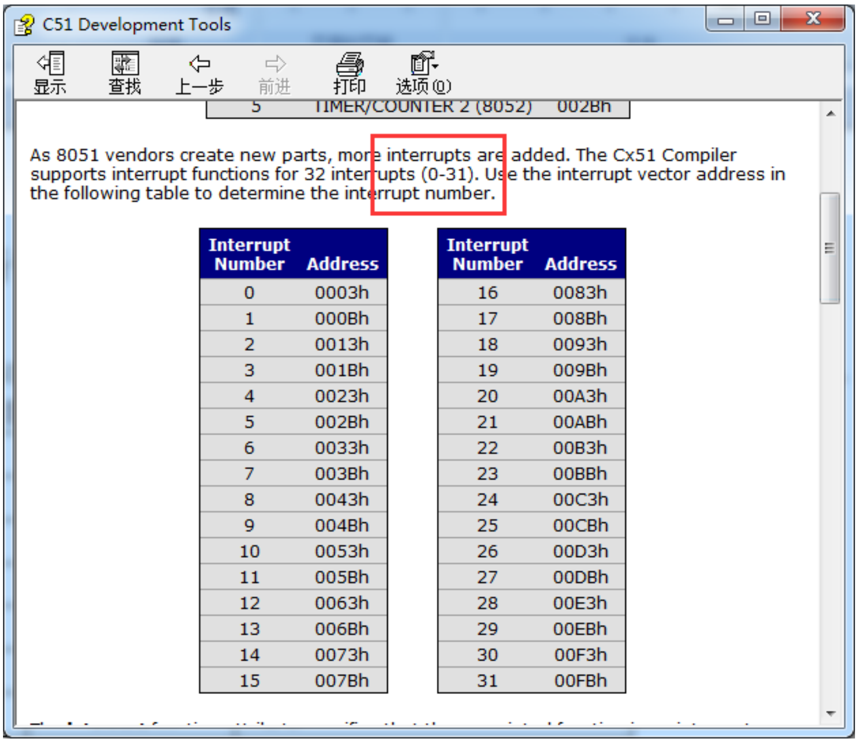
Complete the creation of the multi-file project as shown in the figure below.



```
#include <stc8g.h>

//将中断服务程序定义为普通子程序
void PWM5_ISR() // interrupt 32
{
    if (PWMCFG45 & PWM5CBIF)
    {
        PWMCFG45 &= ~PWM5CBIF;
    }
}

int main()
{
    P5M0 = 0x00;
    P5M1 = 0x00;
    P_SW2 = 0x80;

```

Appendix R Handling of Compilation Error in Keil with Interrupt Numbers Greater Than 31

In Keil's C51 compilation environment, only 0 ~ 31 of the interrupt number are supported, that is, the interrupt vector must be less than 0100H.



5	TIMER/COUNTER 2 (8052)	002Bh

As 8051 vendors create new parts, more interrupts are added. The Cx51 Compiler supports interrupt functions for 32 interrupts (0-31). Use the interrupt vector address in the following table to determine the interrupt number.

Interrupt Number	Address
0	0003h
1	000Bh
2	0013h
3	001Bh
4	0023h
5	002Bh
6	0033h
7	003Bh
8	0043h
9	004Bh
10	0053h
11	005Bh
12	0063h
13	006Bh
14	0073h
15	007Bh

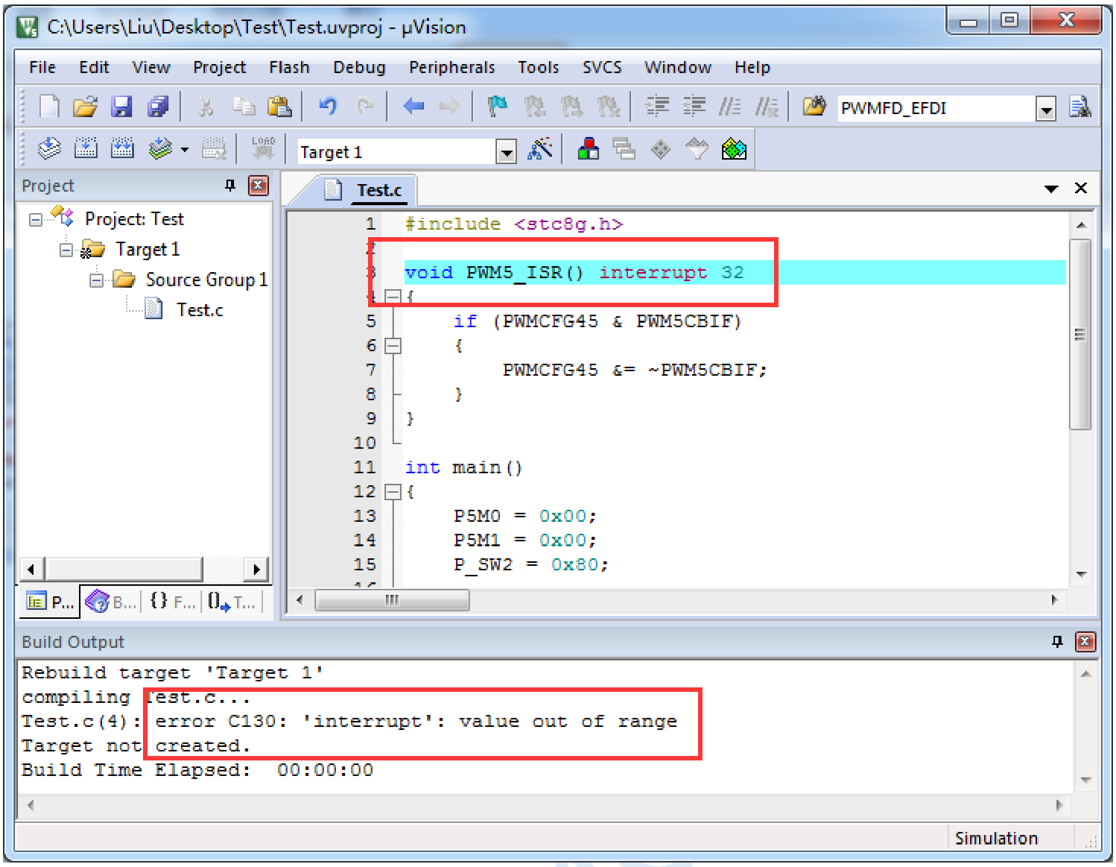
Interrupt Number	Address
16	0083h
17	008Bh
18	0093h
19	009Bh
20	00A3h
21	00ABh
22	00B3h
23	00BBh
24	00C3h
25	00CBh
26	00D3h
27	00DBh
28	00E3h
29	00EBh
30	00F3h
31	00FBh

The following table is a list of interrupts for all current STC series:

Interrupt number	Interrupt vector	Interrupt type
0	0003 H	INT0
1	000B H	Timer 0
2	0013 H	INT1
3	001B H	Timer 1
4	0023 H	UART 1
5	002B H	ADC
6	0033 H	LVD
7	003B H	PCA
8	0043 H	UART 2
9	004B H	SPI
10	0053 H	INT2

11	005B H	INT3
12	0063 H	Timer 2
13	**006B H**	
14	0073 H	System internal interrupt
15	007B H	System internal interrupt
16	0083 H	INT4
17	008B H	UART 3
18	0093 H	UART 4
19	009B H	Timer 3
20	00A3 H	Timer 4
21	00AB H	Comparator
22	00B3 H	Waveform generator 0
23	00BB H	Waveform generator fault 0
24	00C3 H	I2C
25	00CB H	USB
26	00D3 H	PWMA
27	00DB H	PWMB
28	00E3 H	Waveform generator 1
29	00EB H	Waveform generator 2
30	00F3 H	Waveform generator 3
31	00FB H	Waveform generator 4
32	**0103 H**	**Waveform generator 5**
33	**010B H**	**Waveform generator fault 2**
34	**0113 H**	**Waveform generator fault 4**
35	**011B H**	**Touch Key**
36	**0123 H**	**RTC**
37	**012B H**	**P0 interrupt**
38	**0133 H**	**P1 interrupt**
39	**013B H**	**P2 interrupt**
40	**0143 H**	**P3 interrupt**
41	**014B H**	**P4 interrupt**
42	**0153 H**	**P5 interrupt**
43	**015B H**	**P6 interrupt**
44	**0163 H**	**P7 interrupt**
45	**016B H**	**P8 interrupt**
46	**0173 H**	**P9 interrupt**

It is not difficult to find that starting from the interrupt of the waveform generator 5, there will be errors when all subsequent interrupt service routines be compiled in keil, as shown in the following figure:


C:\Users\Liu\Desktop\Test\Test.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
PWMFD_EFDI
Target 1
Project
Project: Test
Target 1
Source Group 1
Test.c
Test.c
Build Output
Simulation

```
#include <stc8g.h>

void PWM5_ISR() interrupt 32
{
    if (PWMCFG45 & PWM5CBIF)
    {
        PWMCFG45 &= ~PWM5CBIF;
    }
}

int main()
{
    P5M0 = 0x00;
    P5M1 = 0x00;
    P_SW2 = 0x80;
```

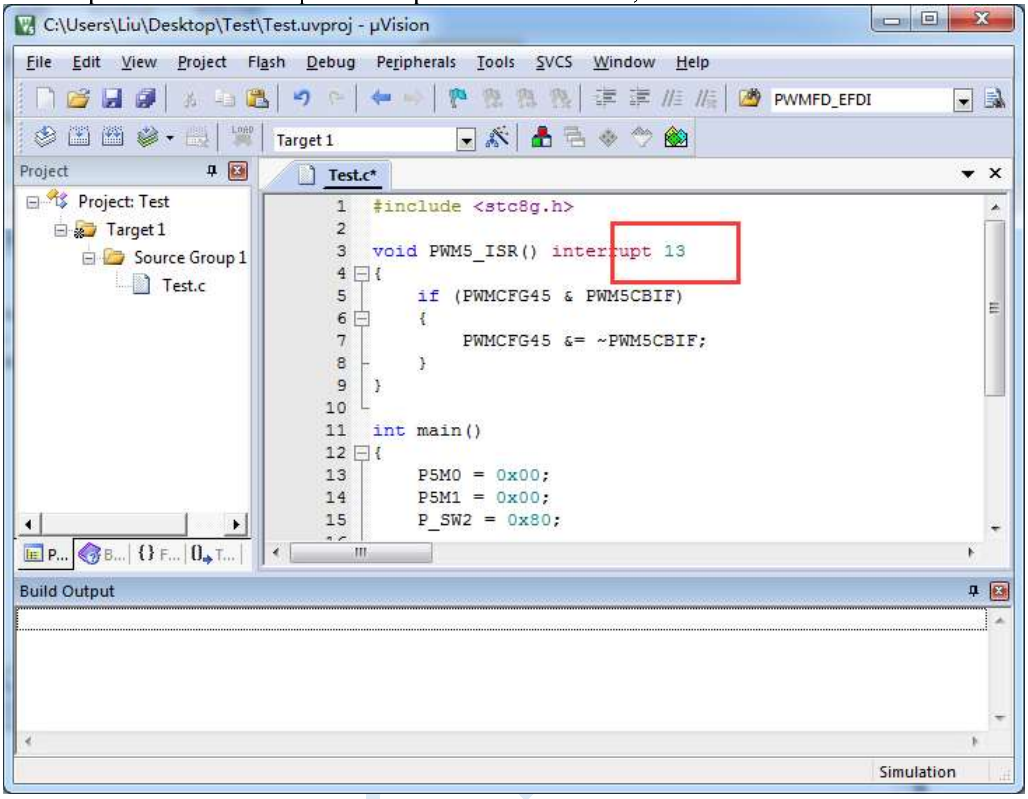
```
Rebuild target 'Target 1'
compiling Test.c...
Test.c(4): error C130: 'interrupt': value out of range
Target not created.
Build Time Elapsed:  00:00:00
```

There are three ways to deal with this kind of error: (All of them need the help of assembly code, the first method is recommended)

Method 1: Borrow Interrupt Vector 13

Among interrupts 0 ~ 31, the 13th is a reserved interrupt number, we can borrow this interrupt number.
The steps are as follows:

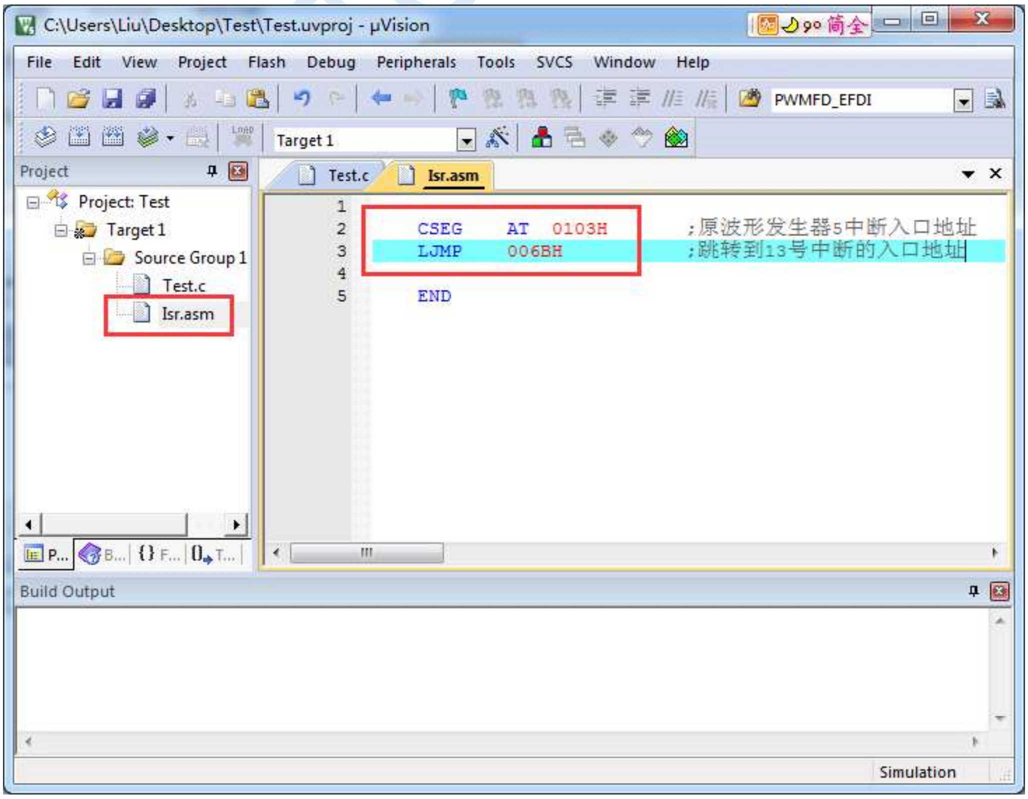
1. Change the interrupt number the compilation reported error to "13", as shown below:



```
#include <stc8g.h>

void PWM5_ISR() interrupt 13
{
    if (PWMCFG45 & PWM5CBIF)
    {
        PWMCFG45 &= ~PWM5CBIF;
    }
}

int main()
{
    P5M0 = 0x00;
    P5M1 = 0x00;
    P_SW2 = 0x80;
```

C:\Users\Liu\Desktop\Test\Test.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
PWMFD_EFDI
Target 1
Project
Project: Test
Target 1
Source Group 1
Test.c
Test.c*
Build Output
Simulation

2. Create a new assembly language file, such as "isr.asm", add it to the project, and add "LJMP 006BH" at the address "0103H", as shown below:

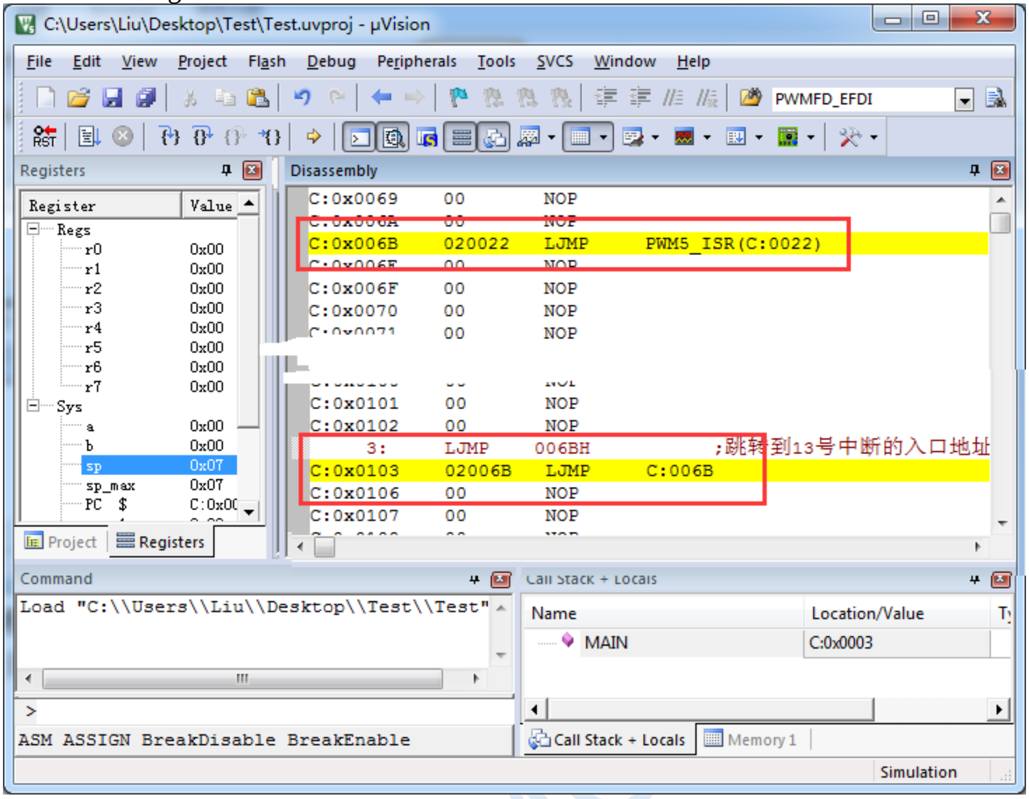


```

    CSEG    AT   0103H          ;原波形发生器5中断入口地址
    LJMP    006BH               ;跳转到13号中断的入口地址

    END
```

C:\Users\Liu\Desktop\Test\Test.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
PWMFD_EFDI
Target 1
Project
Project: Test
Target 1
Source Group 1
Test.c
Isr.asm
Test.c
Isr.asm
Build Output
Simulation

3. Compile successfully.

Now, after being compiled by Keil's C51 compiler, there is an "LJMP PWM5_ISR" at 006BH and an "LJMP 006BH" at 0103H, as shown in the figure below:


C:\Users\Liu\Desktop\Test\Test.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
PWMFD_EFDI
Registers
Disassembly
Project
Registers
Command
Call Stack + Locals
Memory 1
Simulation

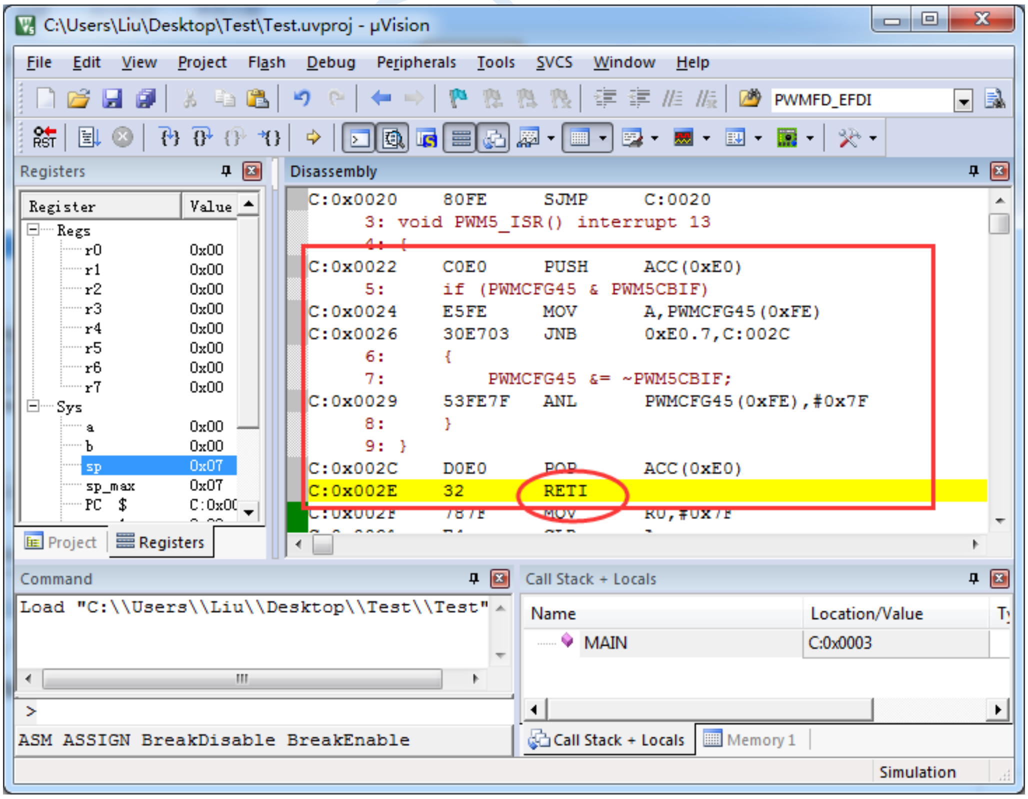
```
Register    Value
Regs
  r0        0x00
  r1        0x00
  r2        0x00
  r3        0x00
  r4        0x00
  r5        0x00
  r6        0x00
  r7        0x00
Sys
  a         0x00
  b         0x00
  sp        0x07
  sp_max    0x07
  PC  $     C:0x00
```

```
C:0x0069    00       NOP
C:0x006A    00       NOP
C:0x006B    020022   LJMP     PWM5_ISR(C:0022)
C:0x006E    00       NOP
C:0x006F    00       NOP
C:0x0070    00       NOP
C:0x0071    00       NOP
C:0x0101    00       NOP
C:0x0102    00       NOP
     3:     LJMP    006BH              ;跳转到13号中断的入口地址
C:0x0103    02006B   LJMP     C:006B
C:0x0106    00       NOP
C:0x0107    00       NOP
```

```
Load "C:\\Users\\Liu\\Desktop\\Test\\Test"
>
ASM ASSIGN BreakDisable BreakEnable
```

Name	Location/Value
MAIN	C:0x0003

When the PWM5 interrupt occurs, the hardware will jump to the 0103H address automatically to execute "LJMP 006BH", and then execute "LJMP PWM5_ISR" at 006BH to jump to the real interrupt service routine, as shown in the figure below:


C:\Users\Liu\Desktop\Test\Test.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
PWMFD_EFDI
Registers
Disassembly
Project
Registers
Command
Call Stack + Locals
Memory 1
Simulation

```
Register    Value
Regs
  r0        0x00
  r1        0x00
  r2        0x00
  r3        0x00
  r4        0x00
  r5        0x00
  r6        0x00
  r7        0x00
Sys
  a         0x00
  b         0x00
  sp        0x07
  sp_max    0x07
  PC  $     C:0x00
```

```
C:0x0020    80FE     SJMP     C:0020
     3: void PWM5_ISR() interrupt 13
     4: {
C:0x0022    C0E0     PUSH     ACC(0xE0)
     5:     if (PWMCFG45 & PWM5CBIF)
C:0x0024    E5FE     MOV      A,PWMCFG45(0xFE)
C:0x0026    30E703   JNB      0xE0.7,C:002C
     6:     {
     7:         PWMCFG45 &= ~PWM5CBIF;
C:0x0029    53FE7F   ANL      PWMCFG45(0xFE),#0x7F
     8:     }
     9: }
C:0x002C    D0E0     POP      ACC(0xE0)
C:0x002E    32       RETI
C:0x002F    787F     MOV      R0,#0x7F
```

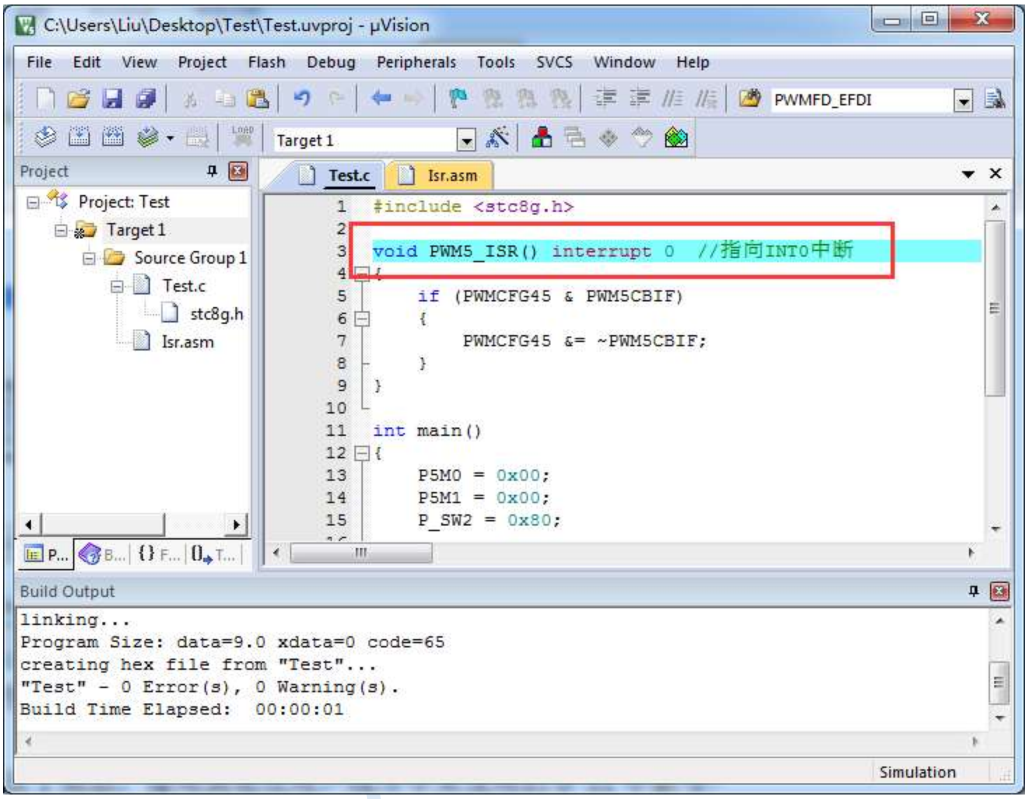
```
Load "C:\\Users\\Liu\\Desktop\\Test\\Test"
>
ASM ASSIGN BreakDisable BreakEnable
```

Name	Location/Value
MAIN	C:0x0003

After the execution of the interrupt service routine is completed, it returns through the RETI instruction. The entire interrupt response process just executed an additional LJMP statement.

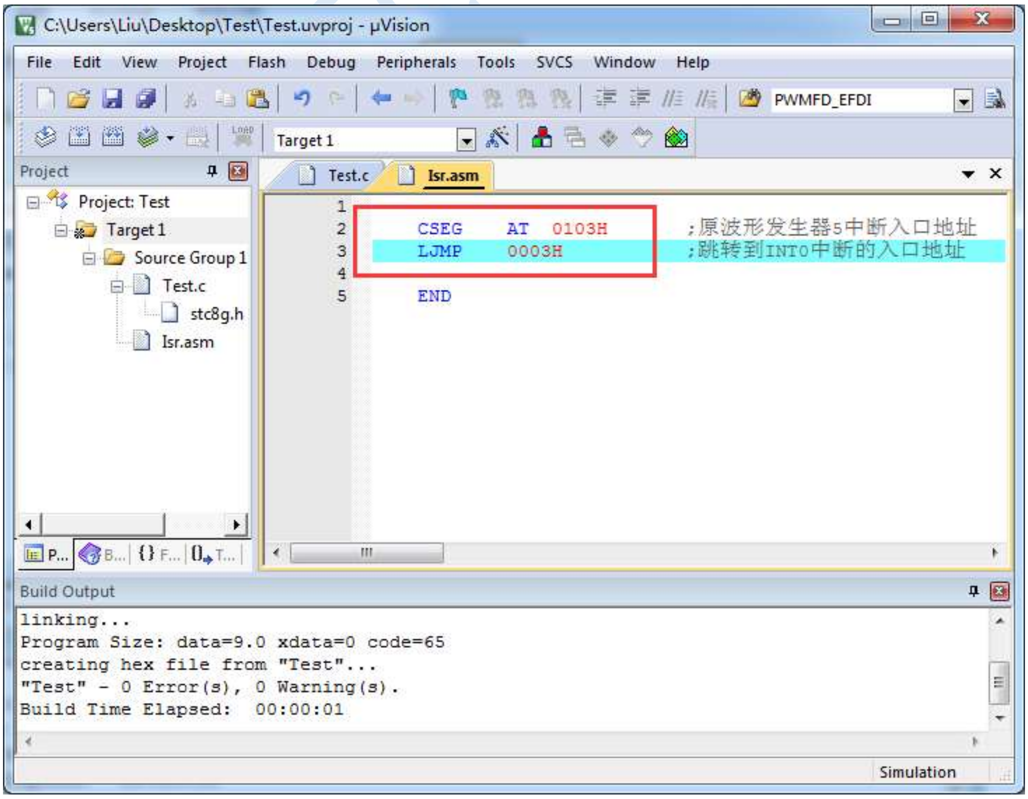
Method 2: Similar to method 1, borrow unused interrupt numbers from 0 to 31 in user program.

For example, in the user's code, if the INT0 interrupt is not used, the above code can be modified similarly to method 1:


C:\Users\Liu\Desktop\Test\Test.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
PWMFD_EFDI
Target 1
Project
Project: Test
Target 1
Source Group 1
Test.c
stc8g.h
Isr.asm
Test.c
Isr.asm
Build Output
Simulation

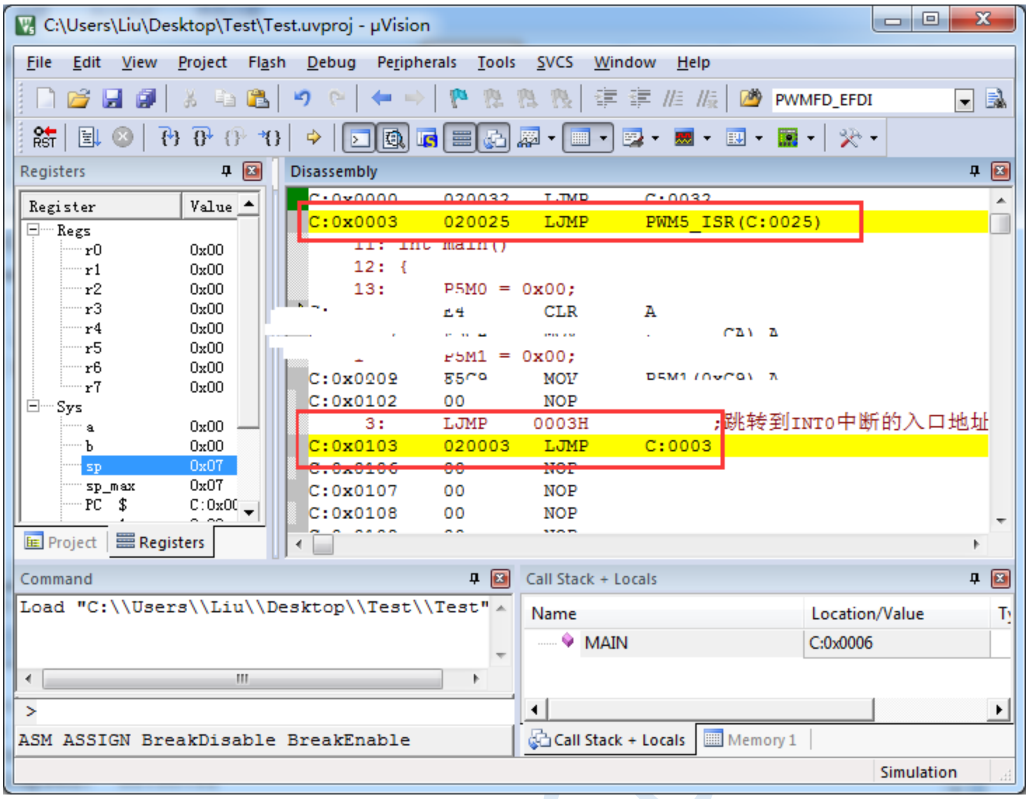
```
#include <stc8g.h>

void PWM5_ISR() interrupt 0   //指向INT0中断
{
    if (PWMCFG45 & PWM5CBIF)
    {
        PWMCFG45 &= ~PWM5CBIF;
    }
}

int main()
{
    P5M0 = 0x00;
    P5M1 = 0x00;
    P_SW2 = 0x80;
```

```
linking...
Program Size: data=9.0 xdata=0 code=65
creating hex file from "Test"...
"Test" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:01
```


C:\Users\Liu\Desktop\Test\Test.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
PWMFD_EFDI
Target 1
Project
Project: Test
Target 1
Source Group 1
Test.c
stc8g.h
Isr.asm
Test.c
Isr.asm
Build Output
Simulation

```

    CSEG    AT  0103H           ;原波形发生器5中断入口地址
    LJMP    0003H               ;跳转到INT0中断的入口地址

    END
```

```
linking...
Program Size: data=9.0 xdata=0 code=65
creating hex file from "Test"...
"Test" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:01
```


C:\Users\Liu\Desktop\Test\Test.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
PWMFD_EFDI
Registers
Disassembly
Project
Registers
Command
Call Stack + Locals
Memory 1
Simulation

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
PC $	C:0x00

```
C:0x0000    020032    LJMP      C:0032
C:0x0003    020025    LJMP      PWM5_ISR(C:0025)
    11: int main()
    12: {
    13:       P5M0 = 0x00;
            E4        CLR       A
            P5M1 = 0x00;
C:0x0009    E5C9      MOV       P5M1(0xC9),A
C:0x0102    00        NOP
      3:    LJMP     0003H                  ;跳转到INT0中断的入口地址
C:0x0103    020003    LJMP      C:0003
C:0x0106    00        NOP
C:0x0107    00        NOP
C:0x0108    00        NOP
```

```
Load "C:\\Users\\Liu\\Desktop\\Test\\Test"
>
ASM ASSIGN BreakDisable BreakEnable
```

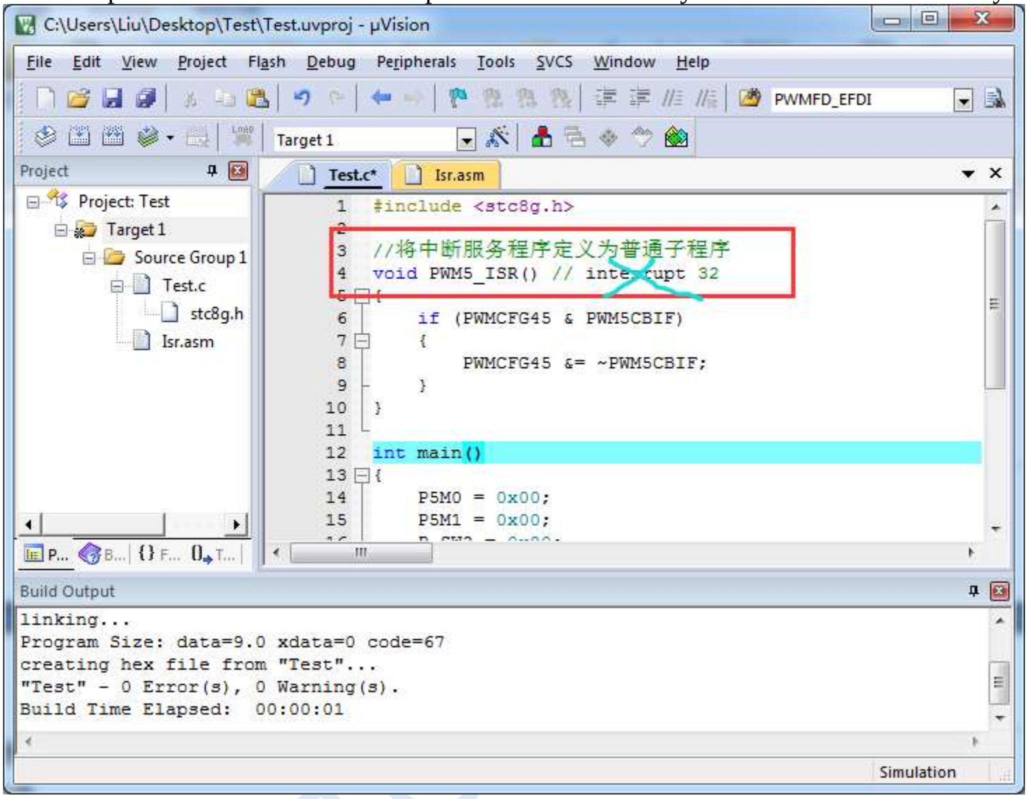
Name	Location/Value
MAIN	C:0x0006

The execution effect is the same as Method 1. This method is applicable to the situation where multiple interrupt numbers greater than 31 need to be remapped.

Method 3: Define the interrupt service routine as a subroutine, and then use the LCALL instruction in the interrupt entry address in the assembly code to execute the service routine.

The steps are as follows:

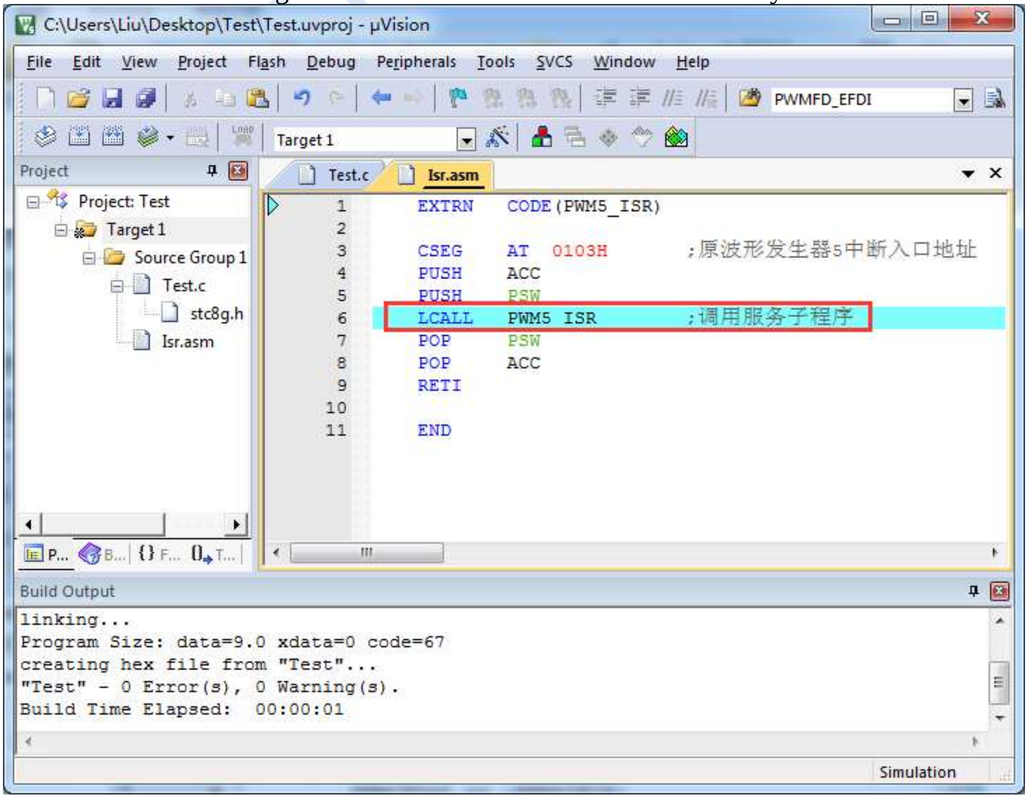
1. Remove the "interrupt" attribute from the interrupt service routine firstly and define it as an ordinary subroutine.


C:\Users\Liu\Desktop\Test\Test.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
PWMFD_EFDI
Target 1
Project
Project: Test
Target 1
Source Group 1
Test.c
stc8g.h
Isr.asm
Test.c*
Isr.asm
Build Output
Simulation

```
#include <stc8g.h>

//将中断服务程序定义为普通子程序
void PWM5_ISR() // interrupt 32
{
    if (PWMCFG45 & PWM5CBIF)
    {
        PWMCFG45 &= ~PWM5CBIF;
    }
}

int main()
{
    P5M0 = 0x00;
    P5M1 = 0x00;
```

```
linking...
Program Size: data=9.0 xdata=0 code=67
creating hex file from "Test"...
"Test" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:01
```

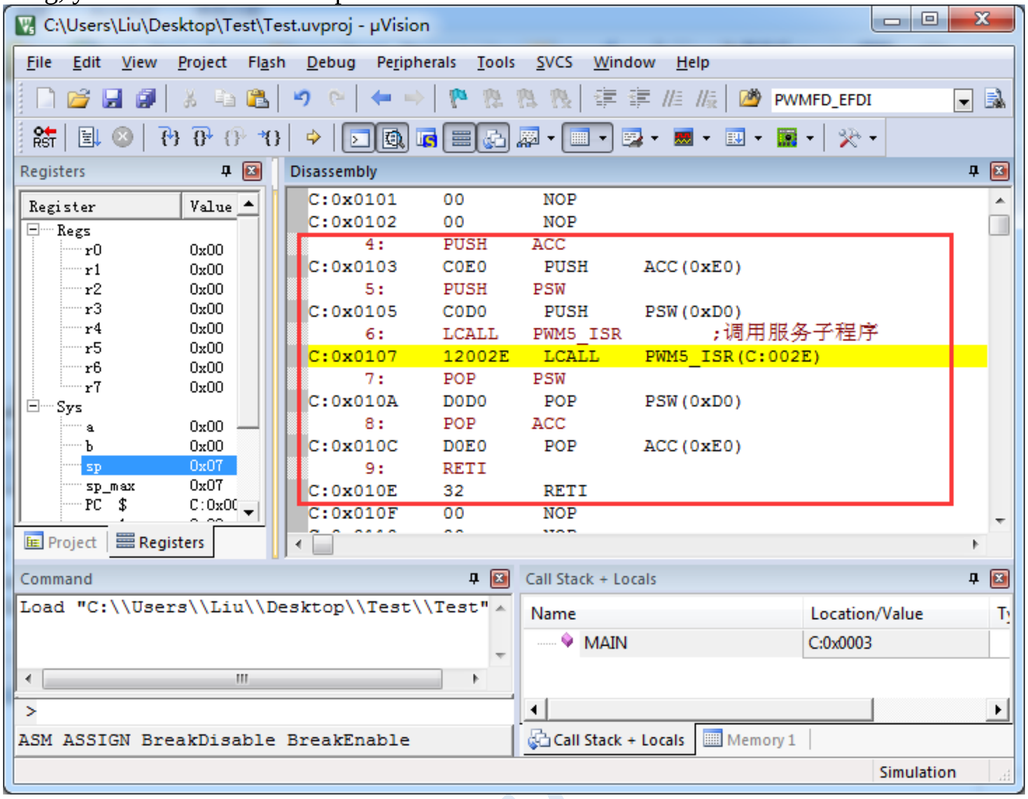
2. Then enter the code shown in the figure below at address 0103H of the assembly file.


C:\Users\Liu\Desktop\Test\Test.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
PWMFD_EFDI
Target 1
Project
Project: Test
Target 1
Source Group 1
Test.c
stc8g.h
Isr.asm
Test.c
Isr.asm
Build Output
Simulation

```
        EXTRN   CODE(PWM5_ISR)

        CSEG    AT  0103H       ;原波形发生器5中断入口地址
        PUSH    ACC
        PUSH    PSW
        LCALL   PWM5_ISR        ;调用服务子程序
        POP     PSW
        POP     ACC
        RETI

        END
```

```
linking...
Program Size: data=9.0 xdata=0 code=67
creating hex file from "Test"...
"Test" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:01
```

3. After compiling, you can find the interrupt service routine at the address of 0103H.


C:\Users\Liu\Desktop\Test\Test.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
PWMFD_EFDI
RST
Registers
Disassembly
Project
Registers
Command
Call Stack + Locals
Memory 1
Simulation

Register	Value
Regs	
r0	0x00
r1	0x00
r2	0x00
r3	0x00
r4	0x00
r5	0x00
r6	0x00
r7	0x00
Sys	
a	0x00
b	0x00
sp	0x07
sp_max	0x07
PC $	C:0x00

```
C:0x0101    00        NOP
C:0x0102    00        NOP
      4:    PUSH     ACC
C:0x0103    C0E0      PUSH     ACC(0xE0)
      5:    PUSH     PSW
C:0x0105    C0D0      PUSH     PSW(0xD0)
      6:    LCALL    PWM5_ISR          ;调用服务子程序
C:0x0107    12002E    LCALL    PWM5_ISR(C:002E)
      7:    POP      PSW
C:0x010A    D0D0      POP      PSW(0xD0)
      8:    POP      ACC
C:0x010C    D0E0      POP      ACC(0xE0)
      9:    RETI
C:0x010E    32        RETI
C:0x010F    00        NOP
```

```
Load "C:\\Users\\Liu\\Desktop\\Test\\Test"
>
ASM ASSIGN BreakDisable BreakEnable
```

Name	Location/Value
MAIN	C:0x0003

This method does not need to remap interrupt entries. But there is a problem with this method. It requires the user to check the disassembly code of the C program to determine which registers need to be pushed onto the stack in the assembly file. PSW, ACC, B, DPL, DPH and R0 ~ R7 are included generally. In addition to the PSW must be pushed onto the stack, registers which are used in the user subroutine must be pushed onto the stack.

Appendix S Electrical Characteristics

S.1 Absolute Maximum Rating

Parameters	Minimum	Maximum	UNIT	Description
Storage temperature	-55	+125	℃	
Operating temperature	-40	+85	℃	If the operating temperature is higher than 85 ℃ (such as around 125 ℃), due to the large temperature drift of the internal IRC clock frequency at high temperature, it is recommended to use an external high temperature clock or crystal oscillator. In addition, when the temperature is high, the frequency does not run fast, and it is recommended to use a working frequency below 24M; if the system must run at a higher temperature, please use an external high-reliability low-frequency active clock. If the working temperature is around -55 ℃, the working voltage should not be too low. It is strongly recommended that the MCU-VCC voltage should not be lower than 3.0V. In addition, the rising speed of the power supply must also be as fast as possible, preferably at the millisecond level.
Operating Voltage	1.9	5.5	V	
VDD to ground voltage	-0.3	+5.5	V	
I/O port to ground voltage	-0.3	VDD+0.3	V	

S.2 DC ELECTRICAL CHARACTERISTICS (3.3V)

(VSS=0V, VDD=5.0V, test temperature =25℃)

Symbol	Parameter	Limits				Test Conditions
		MIN	TYP	MAX	UNIT	
I_{PD}	Power-down mode current	-	0.4	-	uA	
I_{WKT}	Power-down Wake-up timer	-	1.5	-	uA	
I_{LVD}	Low-voltage detection module	-	10	-	uA	
I_{CMP}	Comparator power consumption	-	90	-	uA	
I_{IDL}	Idle mode current (6MHz)	-	0.88	-	mA	
	Idle mode current (12MHz)	-	1.0	-	mA	
	Idle mode current (24MHz)	-	1.16	-	mA	

	Idle mode current (internal 32KHz)	-	0.48	-	mA	
I_{NOR}	Normal mode current（internal 32KHz）	-	0.48	-	mA	Equivalent to 0.5M of traditional 8051
	Normal mode current（500KHz）	-	0.88	-	mA	Equivalent to 7M of traditional 8051
	Normal mode current（600KHz）	-	0.88	-	mA	Equivalent to 8M of traditional 8051
	Normal mode current（700KHz）	-	0.90	-	mA	Equivalent to 9M of traditional 8051
	Normal mode current（800KHz）	-	0.91	-	mA	Equivalent to 11M of traditional 8051
	Normal mode current（900KHz）	-	0.91	-	mA	Equivalent to 12M of traditional 8051
	Normal mode current（1MHz）	-	0.94	-	mA	Equivalent to 13M of traditional 8051
	Normal mode current（2MHz）	-	1.05	-	mA	Equivalent to 26M of traditional 8051
	Normal mode current（3MHz）	-	1.17	-	mA	Equivalent to 40M of traditional 8051
	Normal mode current（4MHz）	-	1.26	-	mA	Equivalent to 53M of traditional 8051
	Normal mode current（5MHz）	-	1.40	-	mA	Equivalent to 66M of traditional 8051
	Normal mode current（6MHz）	-	1.49	-	mA	Equivalent to 79M of traditional 8051
	Normal mode current（12MHz）	-	2.09	-	mA	Equivalent to 158M of traditional 8051
	Normal mode current（24MHz）	-	3.16	-	mA	Equivalent to 317M of traditional 8051
V_{IL1}	Input low voltage	-	-	0.99	V	Turn on Schmitt trigger
		-	-	1.07	V	Turn off Schmitt trigger
V_{IH1}	Input high voltage1(general I/O)	1.18	-	-	V	Turn on Schmitt trigger
		1.09	-	-	V	Turn off Schmitt trigger
V_{IH2}	Input high voltage2(RST pin)	1.18	-	0.99	V	
I_{OL1}	Output low-level sink current	-	20	-	mA	Port voltage 0.45V
I_{OH1}	Output high level current (bi-direction mode)	200	270	-	uA	
I_{OH2}	Output high level current (Push-pull mode)	-	20	-	mA	Port voltage 2.4V
I_{IL}	Logic 0 input current	-	-	50	uA	Port voltage 0V
I_{TL}	Logical 1 to 0 transition current	100	270	600	uA	Port voltage 2.0V
R_{PU}	I/O port pull-up resistor	5.8	5.9	6.0	KΩ	
I/O speed	I/O high current drive, I/O fast conversion		25		MHz	PxDR=0, PxSR=0
	I/O high current drive, I/O fast conversion		22		MHz	PxDR=1, PxSR=0
	I/O low current drive, I/O slow conversion		16		MHz	PxDR=0, PxSR=1
	I/O low current drive, I/O slow		12		MHz	PxDR=1,

	conversion					PxSR=1
Comparators	Fastest speed		10		MHz	Turn off all analog and digital filtering
	Analog filter time		0.1		us	
	Digital filter time		0		system clock	LCDTY＝0
			n+2			LCDTY=n (n=1~63)
I_{PD2}	Power-down mode power consumption when the comparator is enabled	-	400	-	uA	
I_{PD3}	Power-down mode power consumption when LVD is enabled	-	470	-	uA	

S.3 DC ELECTRICAL CHARACTERISTICS (5V)

(VSS=0V, VDD=5.0V, test temperature =25℃)

Symbol	Parameter	Limits				Test Conditions
		MIN	TYP	MAX	UNIT	
I_{PD}	Power-down mode current	-	0.6	-	uA	
I_{WKT}	Power-down Wake-up timer	-	4.4	-	uA	
I_{LVD}	Low-voltage detection module	-	30	-	uA	
I_{CMP}	Comparator power consumption	-	90	-	uA	
I_{IDL}	Idle mode current (6MHz)	-	0.58	-	mA	
	Idle mode current (12MHz)	-	0.98	-	mA	
	Idle mode current (24MHz)	-	1.10	-	mA	
	Idle mode current (internal 32KHz)	-	1.25	-	mA	
I_{NOR}	Normal mode current（internal 32KHz）	-	0.58	-	mA	Equivalent to 0.5M of traditional 8051
	Normal mode current（500KHz）		0.97		mA	Equivalent to 7M of traditional 8051
	Normal mode current（600KHz）		0.97		mA	Equivalent to 8M of traditional 8051
	Normal mode current（700KHz）		1.00		mA	Equivalent to 9M of traditional 8051
	Normal mode current（800KHz）		1.01		mA	Equivalent to 11M of traditional 8051
	Normal mode current（900KHz）		1.01		mA	Equivalent to 12M of traditional 8051
	Normal mode current（1MHz）		1.03		mA	Equivalent to 13M of traditional 8051
	Normal mode current（2MHz）		1.15		mA	Equivalent to 26M of traditional 8051
	Normal mode current（3MHz）		1.27		mA	Equivalent to 40M of traditional 8051
	Normal mode current（4MHz）		1.35		mA	Equivalent to 53M of traditional 8051
	Normal mode current（5MHz）		1.49		mA	Equivalent to 66M of traditional 8051
	Normal mode current（6MHz）	-	1.59	-	mA	Equivalent to 79M of traditional 8051

	Normal mode current（12MHz）	-	2.19	-	mA	Equivalent to 158M of traditional 8051
	Normal mode current（24MHz）	-	3.27	-	mA	Equivalent to 317M of traditional 8051
V_{IL1}	Input low voltage	-	-	1.32	V	Turn on Schmitt trigger
		-	-	1.48	V	Turn off Schmitt trigger
V_{IH1}	Input high voltage1(general I/O)	1.60	-	-	V	Turn on Schmitt trigger
		1.54	-	-	V	Turn off Schmitt trigger
V_{IH2}	Input high voltage2(RST pin)	1.60	-	1.32	V	
I_{OL1}	Output low-level sink current	-	20	-	mA	Port voltage 0.45V
I_{OH1}	Output high level current (bi-direction mode)	200	270	-	uA	
I_{OH2}	Output high level current (Push-pull mode)	-	20	-	mA	Port voltage 2.4V
I_{IL}	Logic 0 input current	-	-	50	uA	Port voltage 0V
I_{TL}	Logical 1 to 0 transition current	100	270	600	uA	Port voltage 2.0V
R_{PU}	I/O port pull-up resistor	4.1	4.2	4.4	KΩ	
I/O speed	I/O high current drive, I/O fast conversion		36		MHz	PxDR=0, PxSR=0
	I/O high current drive, I/O fast conversion		32		MHz	PxDR=1, PxSR=0
	I/O low current drive, I/O slow conversion		26		MHz	PxDR=0, PxSR=1
	I/O low current drive, I/O slow conversion		22		MHz	PxDR=1, PxSR=1
Comparators	Fastest speed		10		MHz	Turn off all analog and digital filtering
	Analog filter time		0.1		us	
	Digital filter time		0		system clock	LCDTY＝0
			n+2			LCDTY=n (n=1~63)
I_{PD2}	Power-down mode power consumption when the comparator is enabled	-	460	-	uA	
I_{PD3}	Power-down mode power consumption when LVD is enabled	-	520	-	uA	

S.4 Internal IRC temperature drift characteristic (reference temperature 25 ℃)

Temperature	Range		
	MIN	TYP	MAX

-40℃～85℃		−1.38%～+1.42%	
-20℃～65℃		−0.88%～+1.05%	

S.5 Low voltage reset threshold voltage (test temperature 25 ℃)

Level	Voltage		
	MIN	TYP	MAX
POR		（1.69V~1.82V）	
LVR0		2.0V（1.88V~1.99V）	
LVR1		2.4V（2.28V~2.45V）	
LVR2		2.7V（2.58V~2.76V）	
LVR3		3.0V（2.86V~3.06V）	

Appendix T Application Considerations

T.1 Notes on STC8H series IO ports

1. For the IO ports of STC8H series chips, except for the ISP download ports P3.0 and P3.1, the initial modes of the remaining IO ports after power-on are high-impedance input mode, and the user cannot directly output the level, so the user initializes the program It is necessary to use the PxM0 and PxM1 registers to initialize the corresponding IO mode in order to be used normally.
2. All I/O ports of STC8H series chips can be set to quasi-bidirectional port mode, strong push-pull output mode, open-drain output mode or high-impedance input mode. In addition, each I/O can independently enable internal 4K Pull resistance
3. STC8H series chips will not automatically set the IO port mode for special IO, such as ADC port, serial port, I2C port, and SPI port. The user must set the corresponding port to the appropriate mode.
4. If the enable P5.4 pin is a reset pin, the reset level is low
5. Special attention: Since all I/Os of STC8H series (except ISP download port P3.0/P3.1) are in high-impedance input mode after power-on, the external level of I/O is not fixed. The MCU directly enters the power-down mode/stop mode, which will cause extra power consumption of the I/O. Before the MCU enters the power-down mode/stop mode, all I/O ports must be set according to the actual situation. mode, all unused external floating I/Os need to be set as quasi-bidirectional ports, and output a high level fixed. Especially for chips with some pins, since some I/O ports inside the chip are not wired to external pins, these I/Os are also in a floating state, and these I/Os also need to be set as quasi-bidirectional ports. And fixed output high level.

T.2 Notes on STC8H2K64T series

1. STC8H2K64T series A version of the chip, when setting the P0, P1, P5 port I/O port interrupt enable bit register P0INTE,P1INTE and P5INTE will affect the digital input enable registers P0IE, P1IE, P5IE of P0, P1, and P5.
2. When testing the ADC function of the A version of the STC8H2K64T series chip, do not set the P0IE, P1IE and P5IE registers, because the settings are invalid.
3. The RTC function of version A chips of STC8H2K64T series has yet to be improved, please do not use
4. Special attention: Since all I/Os of STC8H series (except ISP download port P3.0/P3.1) are in high-impedance input mode after power-on, the external level of I/O is not fixed. The MCU directly enters the power-down mode/stop mode, which will cause extra power consumption of the I/O. Before the MCU enters the power-down mode/stop mode, all I/O ports must be set according to the actual situation. mode, all unused external floating I/Os need to be set as quasi-bidirectional ports, and output a high level fixed. Especially for chips with some pins, since some I/O ports inside the chip are not wired to external pins, these I/Os are also in a floating state, and these I/Os also need to be set as quasi-bidirectional ports. And fixed output high level.

T.3 Notes on STC8H4K64TLR series

1. When the clock source of the RTC selects an external 32.768K crystal oscillator, it is necessary to turn off the digital channel of the crystal oscillator connected to pins P1.6 and P1.7, otherwise there will be additional leakage after entering the STOP mode. (Set both bits 6 and 7 of register P1IE to 0 to close the digital channels of P1.6 and P1.7)

T.4 Notes on STC8H8K64U series

1. For the B version chip of STC8H8K64U, the RTC cannot use the internal 32K as the clock source.
2. For the B version chip of STC8H8K64U, the comparator cannot select the ADC channel as the positive

input.

3. The B version chip of STC8H8K64U will consume about 3uA after entering the power saving mode.

For the above 3 problems, the C version of STC8H8K64U will all be modified correctly

Appendix U PCB design guidance for touch keys

The touch key has strict requirements on PCB design, otherwise its effect will be greatly reduced or even fail.It is recommended that users follow the following principles when designing PCB:

1. Follow the basic principles of common digital-analog hybrid circuit design.

The capacitive touch button module integrates an analog circuit for precise capacitance measurement, so it should be treated as an independent analog circuit when designing the PCB. Follow the basic principles of common digital-analog hybrid circuit design.

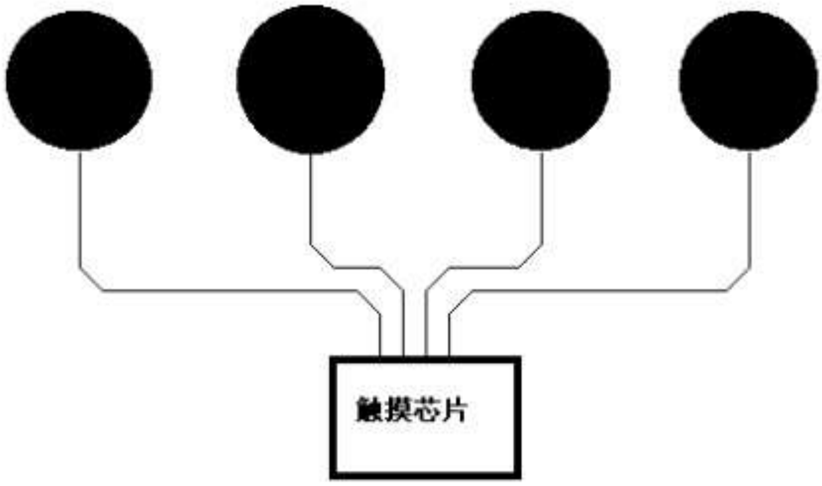
2. Use star grounding

The ground wire of the touch chip should not be shared with other circuits. It should be connected to the ground point of the board's power input separately, which is usually called "star grounding".

3. The impact of noise generated on the power supply on the touch chip

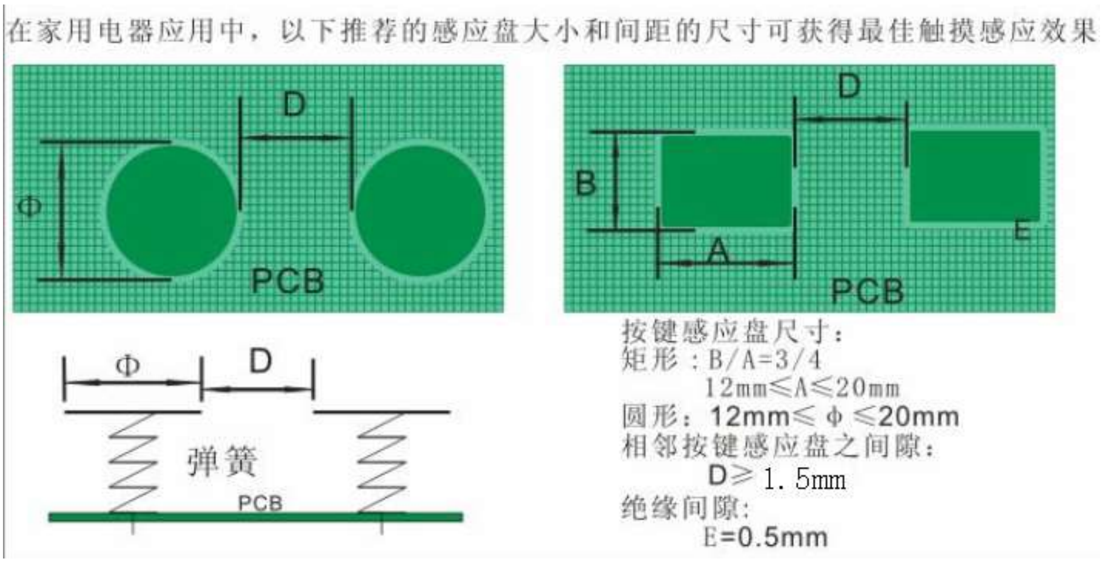
The power ripple and noise should be as small as possible. It is best to use an independent trace to take power from the power supply point of the board and add filtering measures. Do not share the power circuit with other circuits.

4. The connection between the IC and the sensing board should be as long as possible, so that it has an approximate distributed capacitance, as shown in the figure below.



5. The size and gap of the key induction plate (capacitive sensor)

In the case of meeting the aesthetic design requirements of the panel, the optimal touch sensing effect must be obtained through a reasonable arrangement of the size of the sensing plate and the interval size. The induction plate is placed on the bottom layer, and the IC is also placed on the bottom layer. There should be no vias between the induction plate and the IC. The distance between the edges of adjacent sensing plates is preferably at least 1.5mm (dimension D in the figure below). If the PCB area allows, try to use a larger distance. The distance between the copper paving and the induction plate is 0.5mm (dimension E in the figure below).



在家用电器应用中，以下推荐的感应盘大小和间距的尺寸可获得最佳触摸感应效果

按键感应盘尺寸：
矩形：B/A=3/4
12mm≤A≤20mm
圆形：12mm≤ φ ≤20mm
相邻按键感应盘之间隙：
D≥ 1. 5mm
绝缘间隙:
E=0.5mm

6. Copper plating

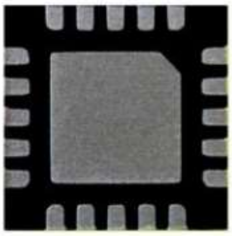
The bottom layer can be covered with grid copper or solid copper. Note that the distance between the copper and the induction plate is 0.5mm. The silk screen information of the top layer is printed on the button. The frame shape of the silk screen is the same as the bottom sensor plate. The top layer corresponding to the bottom sensor plate should not be covered with copper, otherwise the touch action will be shielded. The copper on the top layer is the same as the copper on the bottom layer.

7. Wiring processing

It is better to use a smaller line width for the connection between the sensor plate and the IC, such as between 10 and 15 mils. The connection between the induction plate and the touch chip should not cross the lines with strong interference, high frequency, and high current. Do not run other signal lines within 1.5mm of the connection between the sensor pad and the touch chip, the farther away the better. The top layer corresponds to the bottom sensor plate and connecting line, it is best not to put any line.

Appendix V QFN/DFN packaged components welding method

In the package form of STC products, the more popular QFN and DFN packages have been added. Since the pins of the chip in this package are at the bottom of the chip, manual soldering is difficult. There are small companies on the market that specialize in welding engineering samples, which can undertake engineering sample proofing. If users need to weld by themselves, please refer to the following welding method.



1. Firstly, you need to prepare the following tools: electric soldering iron, hot air gun, tweezers, fixing frame and other tools

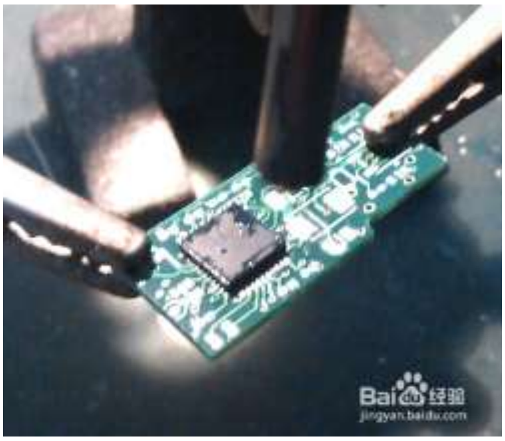
2. The PCB boards and chips that need to be soldered are as follows:

3. Tin the pads of the chip on the board:

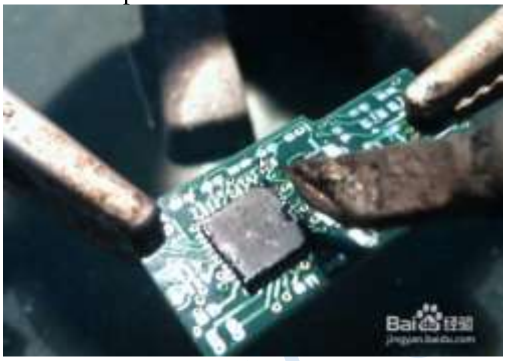
4. Then apply tin to the bottom of the chip. After the tin is applied, it should be flattened to reduce the tin as much as possible, but it cannot be eliminated.

5.Adjust the temperature of the hot air gun, the actual air output is about 240 degrees, because the quality of the air gun is different, adjust it according to the actual situation.

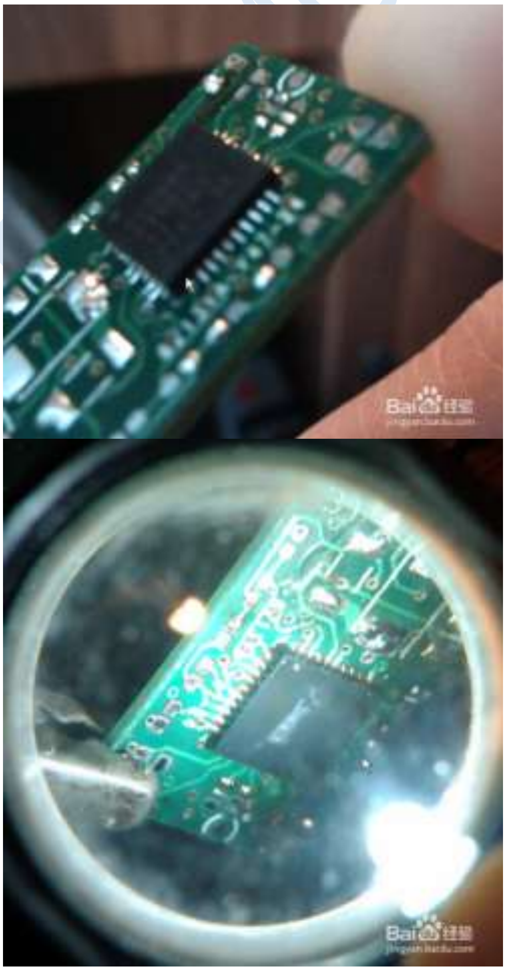
6. Put the chip on the pad, be sure to place it straight, and then blow it with a hot air gun at a uniform speed until the tin melts, usually within 20 seconds.



7. Use a soldering iron to tin the pins on the chip side



8. The effect after welding



Appendix W About whether to bake before reflow soldering

According to the requirements of the International Moisture Sensitivity Level 3 (MSL3) specification, the SMD components must be reflowed within 168 hours and 7 days after the vacuum packaging is unpacked. If not, they must be baked again at high temperature.

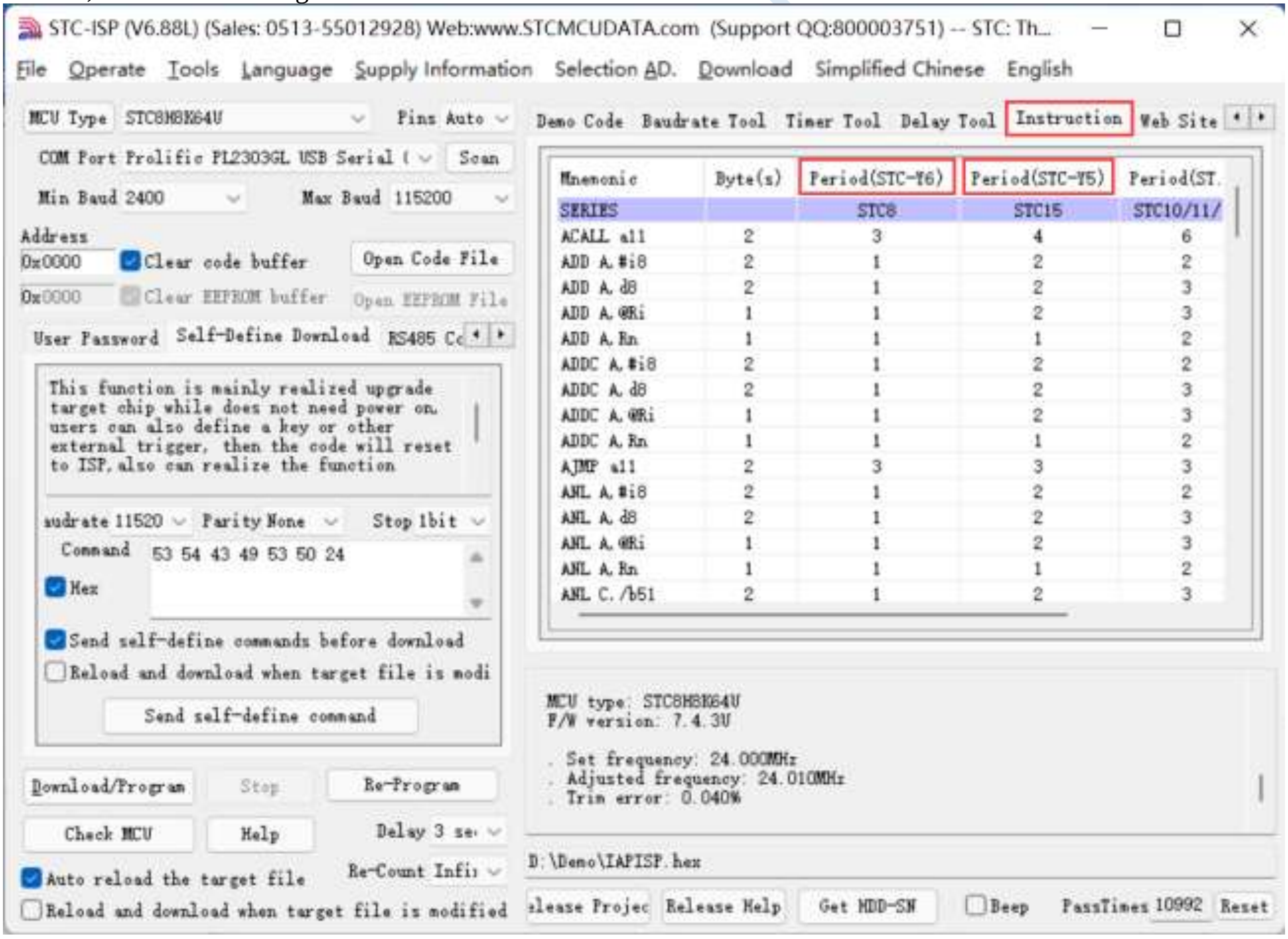
SOP/TSSOP plastic tubes cannot withstand high temperatures above 100 degrees, and must be reflow soldered within 7 days after unpacking. Otherwise, remove the plastic tubes that cannot withstand high temperatures above 100 degrees before reflow soldering, put them in metal trays, and re-bake them. Baking: 110～125℃, 4～8 hours

LQFP/QFN/DFN trays can withstand high temperatures above 100 degrees, and must be reflowed within 7 days after unpacking.

Appendix X Precautions for STC8H series MCU to replace STC15 series

■ MCU instructions

The instruction code of STC8H series is completely consistent with STC15 series, so the code of STC15 series is transplanted to STC8H,The operation is still correct, but the instruction speed of the STC8H series is faster than that of the STC15 series. The instruction system of the STC15 series belongs to the STC-Y5 series of instructions, while the instructions of the STC8H series belong to the STC-Y6 series of instructions. Most of the instructions of the STC-Y6 series are executed. Only one CPU clock is required. If there is a command delay code in the user code, it needs to be adjusted. For the comparison of each instruction, please refer to the instruction table of the STC download software, as shown in the figure below:



STC-ISP (V6.88L) (Sales: 0513-55012928) Web:www.STCMCUDATA.com (Support QQ:800003751) -- STC: Th...

MCU Type STC8H8K64U Pins Auto

COM Port Prolific PL2303GL USB Serial (

Min Baud 2400 Max Baud 115200

Address

0x0000 ☑ Clear code buffer

0x0000 □ Clear EEPROM buffer

This function is mainly realized upgrade target chip while does not need power on, users can also define a key or other external trigger, then the code will reset to ISP, also can realize the function

audrate 11520 Parity None Stop 1bit

Command 53 54 43 49 53 50 24

☑ Hex

☑ Send self-define commands before download

□ Reload and download when target file is modi

Delay 3 se

☑ Auto reload the target file Re-Count Infi

□ Reload and download when target file is modified

Mnenonic	Byte(s)	Period(STC-Y6)	Period(STC-Y5)	Period(ST.
SERIES		STC8	STC15	STC10/11/
ACALL a11	2	3	4	6
ADD A,#i8	2	1	2	2
ADD A,d8	2	1	2	3
ADD A,@Ri	1	1	2	3
ADD A,Rn	1	1	1	2
ADDC A,#i8	2	1	2	2
ADDC A,d8	2	1	2	3
ADDC A,@Ri	1	1	2	3
ADDC A,Rn	1	1	1	2
AJMP a11	2	3	3	3
ANL A,#i8	2	1	2	2
ANL A,d8	2	1	2	3
ANL A,@Ri	1	1	2	3
ANL A,Rn	1	1	1	2
ANL C,/b51	2	1	2	3

```
MCU type: STC8H8K64U
F/W version: 7.4.3U

. Set frequency: 24.000MHz
. Adjusted frequency: 24.010MHz
. Trim error: 0.040%
```

D:\Demo\IAPISP.hex

□ Beep PassTimes 10992

■ I/O port

After the STC8H series MCU is powered on, the I/O mode is different from that of the STC15 series. All I/O ports of STC15 series single-chip microcomputers are in 8051 quasi-bidirectional port mode after power-on. In the I/O of STC8H series single-chip microcomputers, except for ISP download pins P3.0/P3.1 which are quasi-bidirectional port modes, all the other I/O ports are in high-impedance input mode after power on. The traditional

8051 and STC 15 series single-chip microcomputers are in quasi-bidirectional port mode and output high level after power-on. Often customers use I/O to drive motors or LED lights in their systems, so there will be moments when the single-chip microcomputer is powered on. Move it once or the LED will flash once. The I/O of the STC8H series is in high-impedance input mode after power-on, which can avoid this kind of malfunction of the motor and LED.

Because in the I/O of STC8H series single-chip microcomputer, except ISP download pin P3.0/P3.1 which is quasi-bidirectional port mode, all other I/O ports are in high-impedance input mode after power-on, so when users need Before the I/O ports of STC8H series output signals, the two registers PxM0 and PxM1 must be used to set the I/O working mode.

■ Reset foot

The P5.4 port of the STC8H series and STC15 series is generally used as a normal I/O port. When the user sets P5.4 as the reset pin function during ISP download, the P5.4 port is the reset of the microcontroller Pin (RESET pin). For the STC15H series, when the reset pin is high, the microcontroller is in the reset state, and when the reset pin is low, the microcontroller is released from the reset state. The reset levels of STC8 series and STC15H series are reversed, that is, for STC8H series, when the reset pin is low, the microcontroller is in the reset state, and when the reset pin is high, the microcontroller is released from the reset state.

Therefore, when the user enables the reset pin function of port P5.4, it is necessary to pay attention to the reset level.

■ ADC

The addresses of the ADC_CONTR, ADC_RES and ADC_RESL 3 registers of STC8H series and STC15 series are the same.But the STC8H series adds two new registers: ADCCFG and ADCTIM.

STC15 series start ADC conversion bit ADC_START is located at BIT3 of register ADC_CONTR, while STC8H series is located at BIT6 of ADC_CONTR

The STC15 series ADC conversion complete flag ADC_FLAG is located at BIT4 of the register ADC_CONTR, while the STC8H series is located at BIT5 of ADC_CONTR

STC15 series ADC speed control is ADC_SPEED located at BIT6-BIT5 of register ADC_CONTR, and STC8H series is located at BIT3-BIT0 of ADCCFG

The alignment control bit ADRJ of the STC15 series ADC conversion result is located at BIT5 of the register CLK_DIV, while the alignment control bit RESFMT of the STC8H series is located at BIT5 of ADCCFG

The STC8H series adds a more precise ADC conversion timing control mechanism, which can be set through the register ADCTIM

■ EEPROM

The waiting time for EEPROM erasing and programming of STC15 series is set by Bit2-Bit0 of the register IAP_CONTR. The setting is only an approximate frequency range value. The STC8H series adds a new register IAP_TPS (SFR address: 0F5H), dedicated to setting EEPROM erasing In addition to the waiting time for programming, and the user does not need to calculate, just fill in IAP_TPS directly according to the current CPU working frequency, and the hardware will automatically calculate the waiting time. (For example: the current CPU operating frequency is 24MHz, you only need to fill in 24 to IAP_TPS).

Appendix Y Precautions for STC8H series single-chip microcomputer to replace STC8A/8F series

■ I/O port

After the STC8H series MCU is powered on, the I/O mode is different from that of the STC8A/8F series. All I/O ports of STC8A/8F series single-chip microcomputers are in 8051 quasi-bidirectional port mode after power-on. In the I/O of STC8H series single-chip microcomputers, except for ISP download pins P3.0/P3.1 which are quasi-bidirectional port modes, the rest All of the I/O ports are in high impedance input mode after power on. The traditional 8051 and STC 15/8A/8F series single-chip microcomputers are in quasi-bidirectional port mode and output high level after power-on. Often customers' systems use I/O to drive motors or LED lights, so the single-chip microcomputer will be powered on. The motor will move or the LED will flash. The I/O of the STC8H series is in high-impedance input mode after power-on, which can avoid this kind of malfunction of the motor and LED.

Because in the I/O of STC8H series single-chip microcomputer, except ISP download pin P3.0/P3.1 which is quasi-bidirectional port mode, all other I/O ports are in high-impedance input mode after power-on, so when users need Before the I/O ports of STC8H series output signals, the two registers PxM0 and PxM1 must be used to set the I/O working mode.

■ Reset foot

The P5.4 port of the STC8H series and STC8A/8F series is generally used as a normal I/O port. When the user sets P5.4 as the reset pin function during ISP download, the P5.4 port is a microcontroller The reset pin (RESET pin). For the STC8A/8F series, when the reset pin is high, the microcontroller is in the reset state, and when the reset pin is low, the microcontroller is released from the reset state. The reset levels of STC8H series and STC8A/8F series are reversed, that is, for STC8H series, when the reset pin is low, the microcontroller is in the reset state, and when the reset pin is high, the microcontroller is released from the reset state.

Therefore, when the user enables the reset pin function of port P5.4, it is necessary to pay attention to the reset level.

■ EEPROM

The waiting time for erasing and programming of EEPROM of STC8A/8F series is set by Bit2-Bit0 of the register IAP_CONTR. The setting is only a rough frequency range value. The STC8H series adds a new register IAP_TPS (SFR address: 0F5H), dedicated to setting The waiting time of EEPROM erasing and programming, and the user does not need to calculate, just fill in IAP_TPS directly according to the current CPU operating frequency, and the hardware will automatically calculate the waiting time. (For example: the current CPU operating frequency is 24MHz, you only need to fill in 24 to IAP_TPS)

Appendix Z Internally tested models

Z.1 STC8H2K64T-35I-LQFP48/QFN48

Z.1.1 Features and Price

- **Selection and price (No external crystal and external reset required with 15 channels 12-bit ADC)**

MCU	Operating voltage（V）	Flash Code Memory (100 thousand times) (Byte)	idata Internal DATA RAM(Byte)	xdata Internal extended SRAM (Byte)	Enhanced Dual DPTR increasing or decreasing	EEPROM 100 thousand times) (Byte)	Maximum I/O Lines	Traditional I/O interrupt(INT0/INT1/INT2/INT3/INT4) (can wake-up CPU)	All I/O ports support interrupts and can wake up MCU	UARTs which can wake-up CPU	SPI which can wake-up CPU	I^{2}C which can wake-up CPU	Touch key	LED driver	RTC	MDU16 (Hardware 16-bit Multiplier and Divider)	Timers/Counters (T0-T4 Pin can wake-up CPU)	16-bit advanced PWM timer with Complementary symmetrical dead-time	Power-down Wake-up timer	15 channels high speed ADC (8 PWMs can be used as 8 DACs)	Comparator (May be used as ADC to detect external power-down)	Internal LVD interrupt (can wake-up CPU)	Watch-dog Timer	Internal high reliable reset circuit with 4 levels optional reset threshold voltage	Internal high precision Clock (adjustbal under 36MHz)	Clock output and Reset	Program encrypted transmission (Anti-blocking)	Password can be set for next update	Support RS485 download	Support software USB download directly	Online debug itself	Price & Package		products supply information
																																LQFP48 <9mm*9mm>	QFN48 <6mm*6mm>	
STC8H2K32T	1.9-5.5	32K	256	2K	2	32K	44	Y	Y	4	Y	Y	Y	Y	Y	Y	5	8	Y	12-bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y			Some available
STC8H2K60T	1.9-5.5	60K	256	2K	2	4K	44	Y	Y	4	Y	Y	Y	Y	Y	Y	5	8	Y	12-bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y			
STC8H2K64T	1.9-5.5	64K	256	2K	2	IAP	44	Y	Y	4	Y	Y	Y	Y	Y	Y	5	8	Y	12-bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y			

- **Core**
  - ✓ Ultra-high speed 8051 Core with single clock per machine cycle, which is called 1T and the speed is about 12 times faster than traditional 8051
  - ✓ Fully compatible instruction set with traditional 8051
  - ✓ 29 interrupt sources and 4 interrupt priority levels
  - ✓ Online debugging is supported

- **Operating voltage**
  - ✓ 1.9V～5.5V

- **Operating temperature**
  - ✓ -40℃~85℃(The chip is produced in -40℃～125℃ process. Please refer to the description of the electrical characteristics chapter for applications beyond the temperature range)

- **Flash memory**

- ✓ Up to 64Kbytes of Flash memory to be used for storing user code
- ✓ Configurable EEPROM size, 512bytes single page for being erased, which can be repeatedly erased more than 100 thousand times.
- ✓ In-System-Programming, ISP in short, can be used to update the application code. No special programmer is needed.
- ✓ Online debugging with single chip is supported, and no special emulator is needed. The number of breakpoints is unlimited theoretically.

➢ **SRAM**

- ✓ 128 bytes internal direct access RAM (DATA, use keyword *data* to declare in C language program)
- ✓ 128 bytes internal indirect access RAM (IDATA, use keyword *idata* to declare in C language program)
- ✓ 2048 bytes internal extended RAM (internal XDATA, use keyword *xdata* to declare in C language program)

➢ **Clock**

- ✓ Internal high precise RC clock IRC(IRC for short, ranges from 4MHz to 36MHz), adjustable while ISP and can be divided to lower frequency by user software, 100KHz for instance.
  - ✓ Error:±0.3% (at the temperature 25℃)
  - ✓ -1.35%～+1.30% temperature drift (at the temperature range of -40℃ to +85℃)
  - ✓ -0.76%～+0.98% temperature drift (at the temperature range of -20℃ to 65℃)
- ✓ Internal 32KHz low speed IRC with large error
- ✓ External crystal (4MHz～33MHz) and external clock

  Users can freely choose the above 3 clock sources

➢ **Reset**

- ✓ Hardware reset
  - ✓ Power-on reset. Measured voltage is 1.69V~1.82V. (Effective when the chip does not enable the low voltage reset function)

    The power-on reset voltage is a voltage range consisting of an upper limit voltage and a lower limit voltage. When the operating voltage drops from 5V / 3.3V to the lower limit threshold voltage of the power-on reset, the chip is in a reset state; when the voltage rises from 0V to the upper threshold voltage of power-on reset, the chip is released from the reset state.
  - ✓ Reset by reset pin. The default function of P5.4 is the I/O port. The P5.4 pin can be set as the reset pin while ISP download. (Note: When the P5.4 pin is set as the reset pin, the reset level is low.)
  - ✓ Watch dog timer reset
  - ✓ Low voltage detection reset. 4 low voltage detection levels are provided, 1.9V, 2.3V, 2.8V, 3.0V. Each level of low-voltage detection voltage is a voltage range consisting of an upper limit voltage and a lower limit voltage. When the operating voltage drops from 5V / 3.3V to the lower limit threshold voltage of low-voltage detection, the low-voltage detection takes effect. When the voltage rises from 0V to the upper threshold voltage, the low voltage detection becomes effective.
- ✓ Software reset
  - ✓ Writing the reset trigger register using software

➢ **Interrupts**

- ✓ 29 interrupt sources: INT0(Supports rising edge and falling edge interrupt), INT1(Supports rising edge and falling edge interrupt), INT2(Supports falling edge interrupt only), INT3(Supports falling edge interrupt only), INT4(Supports falling edge interrupt only), timer 0, timer 1, timer 2, timer 3, timer 4, UART 1, UART 2, UART 3, UART 4, ADC, LVD, SPI, I^{2}C, comparator, PWMA, PWMB, RTC, TKS, EXP0, EXP1, EXP2, EXP3, EXP4, EXP5.
- ✓ 4 interrupt priority levels
- ✓ Interrupts that can wake up the CPU in clock stop mode: INT0(P3.2), INT1(P3.3), INT2(P3.6), INT3(P3.7), INT4(P3.0), T0(P3.4), T1(P3.5), T2 (P1.2), T3 (P0.4), T4 (P0.6), RXD (P3.0/P3.6/P1.6/P4.3), RXD2 (P1.0/P4.6), RXD3(P0.0/P5.0), RXD4(P0.2/P5.2), I2C_SDA(P1.4/P2.4/P3.3) and comparator interrupt, low voltage detection interrupt, power-down wake-up timer.

➢ **Digital peripherals**

- ✓ 5 16-bit timers: timer0, timer1, timer2, timer 3, timer 4, where the mode 3 of timer 0 has the Non-Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
- ✓ 4 high speed UARTs: UART1, UART2, UART3, UART4, whose maximum baudrate clock may be FOSC/4
- ✓ 8 channels/2 groups of enhanced PWM, which can realize control signals with dead time, and support external fault detection function. In addition, supports 16-bit timers, 8 external interrupts, 8 external captures and pulse width measurement functions.
- ✓ SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
- ✓ I^{2}C: Master mode or slave mode are supported.
- ✓ MDU16: Hardware 16-bit Multiplier and Divider which supports 32-bit divided by 16-bit, 16-bit divided by 16-bit, 16-bit multiplied by 16-bit, data shift, and data normalization operations.
- ✓ I/O port interrupt: All I/Os support interrupts, each group of I/O interrupts has an independent interrupt entry address, all I/O interrupts can support 4 types interrupt mode: high level interrupt, low level interrupt, rising edge interrupt, falling edge interrupt.

➢ **Analog peripherals**

- ✓ Ultra high speed ADC which supports 12-bit precision 15 channels (channel 0 to channel 14) analog-to-digital conversion. The maximum speed can be 800K(800K ADC conversions per second)

- ✓ ADC channel 15 is used to test the internal reference voltage. (The default internal reference voltage is 1.19V when the chip is shipped)
- ✓ Comparator. A set of comparator (The CMP+ port and all ADC input ports can be selected as the positive terminal of the comparator. So the comparator can be used as a multi-channel comparator for time division multiplexing)
- ✓ Touch key: The microcontroller supports up to 16 touch keys. Every touch key can be enabled independently. The internal reference voltage is adjustable with 4 levels. Charge and discharge time settings and internal working frequency settings are flexible. The touch key supports wake-up CPU from low-power mode.
- ✓ LED driver: can drive up to 256 (8*16*2) LEDs; support common cathode mode, common anode mode and common cathode/common anode mode; support 8-level grayscale adjustment (brightness adjustment)
- ✓ DAC: 8 channels advanced PWM timer can be used as 8 channels DAC

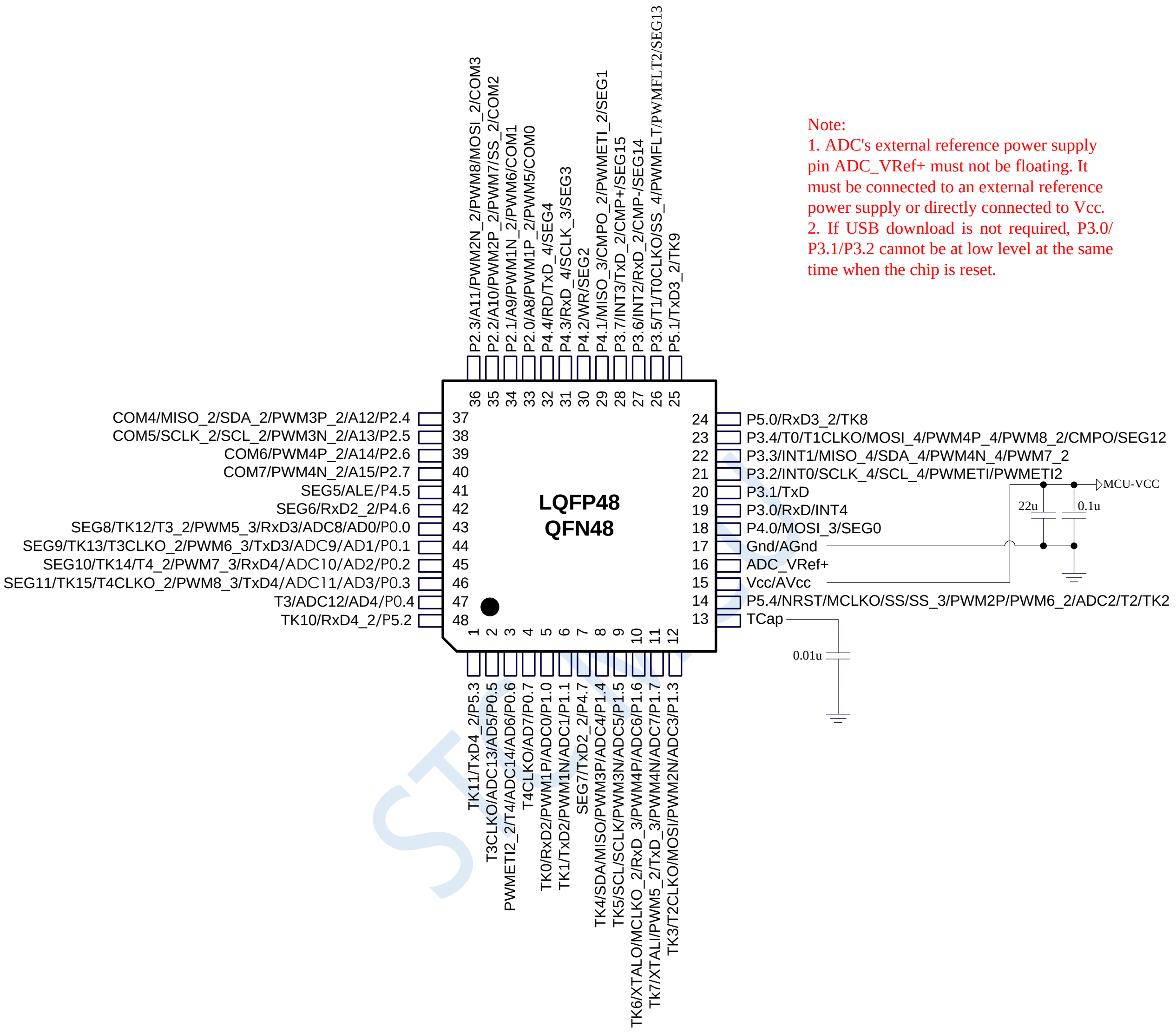
➢ **GPIO**

- ✓ Up to 44 GPIOs: P0.0~P0.7, P1.0~ P1.7 (No P1.2), P2.0~P2.7, P3.0~P3.7, P4.0~P4.7, P5.0~P5.4
- ✓ 4 modes for all GPIOs: quasi_bidirectional mode, push-pull outputmode, open drain mode, high-impedance input mode
- ✓ Except for P3.0 and P3.1, all other I/O ports are in a high-impedance state after power-on. User must set the I/O ports mode before using them. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.

➢ **Package**

- ✓ LQFP48 <9mm*9mm>, QFN48 <6mm*6mm>

Z.1.2 Pinouts


P2.3/A11/PWM2N_2/PWM8/MOSI_2/COM3
P2.2/A10/PWM2P_2/PWM7/SS_2/COM2
P2.1/A9/PWM1N_2/PWM6/COM1
P2.0/A8/PWM1P_2/PWM5/COM0
P4.4/RD/TxD_4/SEG4
P4.3/RxD_4/SCLK_3/SEG3
P4.2/WR/SEG2
P4.1/MISO_3/CMPO_2/PWMETI_2/SEG1
P3.7/INT3/TxD_2/CMP+/SEG15
P3.6/INT2/RxD_2/CMP-/SEG14
P3.5/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2/SEG13
P5.1/TxD3_2/TK9
36
35
34
33
32
31
30
29
28
27
26
25
COM4/MISO_2/SDA_2/PWM3P_2/A12/P2.4 37
COM5/SCLK_2/SCL_2/PWM3N_2/A13/P2.5 38
COM6/PWM4P_2/A14/P2.6 39
COM7/PWM4N_2/A15/P2.7 40
SEG5/ALE/P4.5 41
SEG6/RxD2_2/P4.6 42
SEG8/TK12/T3_2/PWM5_3/RxD3/ADC8/AD0/P0.0 43
SEG9/TK13/T3CLKO_2/PWM6_3/TxD3/ADC9/AD1/P0.1 44
SEG10/TK14/T4_2/PWM7_3/RxD4/ADC10/AD2/P0.2 45
SEG11/TK15/T4CLKO_2/PWM8_3/TxD4/ADC11/AD3/P0.3 46
T3/ADC12/AD4/P0.4 47
TK10/RxD4_2/P5.2 48
LQFP48
QFN48
24 P5.0/RxD3_2/TK8
23 P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO/SEG12
22 P3.3/INT1/MISO_4/SDA_4/PWM4N_4/PWM7_2
21 P3.2/INT0/SCLK_4/SCL_4/PWMETI/PWMETI2
20 P3.1/TxD
19 P3.0/RxD/INT4
18 P4.0/MOSI_3/SEG0
17 Gnd/AGnd
16 ADC_VRef+
15 Vcc/AVcc
14 P5.4/NRST/MCLKO/SS/SS_3/PWM2P/PWM6_2/ADC2/T2/TK2
13 TCap
MCU-VCC
22u
0.1u
0.01u
1
2
3
4
5
6
7
8
9
10
11
12
TK11/TxD4_2/P5.3
T3CLKO/ADC13/AD5/P0.5
PWMETI2_2/T4/ADC14/AD6/P0.6
T4CLKO/AD7/P0.7
TK0/RxD2/PWM1P/ADC0/P1.0
TK1/TxD2/PWM1N/ADC1/P1.1
SEG7/TxD2_2/P4.7
TK4/SDA/MISO/PWM3P/ADC4/P1.4
TK5/SCL/SCLK/PWM3N/ADC5/P1.5
TK6/XTALO/MCLKO_2/RxD_3/PWM4P/ADC6/P1.6
Tk7/XTALI/PWM5_2/TxD_3/PWM4N/ADC7/P1.7
TK3/T2CLKO/MOSI/PWM2N/ADC3/P1.3

Note:
1. ADC's external reference power supply pin ADC_VRef+ must not be floating. It must be connected to an external reference power supply or directly connected to Vcc.
2. If USB download is not required, P3.0/P3.1/P3.2 cannot be at low level at the same time when the chip is reset.

Z.2 STC8H4K64LCD-45I-LQFP64/QFN64/LQFP48/QFN48

Z.2.1 Features and Price(Quasi 16-bit MCU with 16-bit hardware multiplier and divider MDU16)

- **Selection and price (No external crystal and external reset required with 15 channels 12-bit ADC)**

MCU	Operating voltage（V）	Flash Code Memory (100 thousand times) (Byte)	idata Internal DATA RAM(Byte)	xdata Internal extended SRAM (Byte)	Enhanced Dual DPTR increasing or decreasing	EEPROM 100 thousand times) (Byte)	Maximum I/O Lines	Traditional I/O interrupt(INT0/INT1/INT2/INT3/INT4) (can wake-up CPU)	All I/O ports support interrupts and can wake up MCU	DMA UARTs which can wake-up CPU	DMA 8080/6800 interface/ LCM driver(8-bit and 16-bit)	LCD driver (4COM*40SEG)	Touch key	RTC	DMA SPI which can wake-up CPU	I^{2}C which can wake-up CPU (No DMA)	MDU16 (Hardware 16-bit Multiplier and Divider)	Timers/Counters (T0-T4 Pin can wake-up CPU)	16-bit advanced PWM timer with Complementary symmetrical dead-time	Power-down Wake-up timer	DMA 15 channels high speed ADC (8 PWMs can be used as 8 DACs)	Comparator (May be used as ADC to detect external power-down)	Internal LVD interrupt (can wake-up CPU)	Watch-dog Timer	Internal high reliable reset circuit with 4 levels optional reset threshold voltage	Internal high precision Clock (adjustbal under 45MHz)	Clock output and Reset	Program encrypted transmission (Anti-blocking)	Password can be set for next update	Support RS485 download	Support software USB download directly	Online debug itself	Price & Package				products supply information
																																	LQFP64 <12mm*12mm>	QFN64 <8mm*8mm>	LQFP48 <9mm*9mm>	QFN48 <6mm*6mm>	
STC8H4K32TLCD	1.9-5.5	32K	256	4K	2	32K	60	Y	Y	4	Y	Y	Y	Y	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	√				Samples
STC8H4K48TLCD	1.9-5.5	48K	256	4K	2	16K	60	Y	Y	4	Y	Y	Y	Y	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	√				
STC8H4K64TLCD	1.9-5.5	64K	256	4K	2	IAP	60	Y	Y	4	Y	Y	Y	Y	Y	Y	Y	5	8	Y	12bit	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	√				

- **Core**
  - ✓ Ultra-high speed 8051 Core with single clock per machine cycle, which is called 1T and the speed is about 12 times faster than traditional 8051
  - ✓ Fully compatible instruction set with traditional 8051
  - ✓ 42 interrupt sources and 4 interrupt priority levels
  - ✓ Online debugging is supported
- **Operating voltage**
  - ✓ 1.9V～5.5V
- **Operating temperature**
  - ✓ -40℃~85℃(The chip is produced in -40℃～125℃ process. Please refer to the description of the electrical characteristics chapter for applications beyond the temperature range)
- **Flash memory**
  - ✓ Up to 64Kbytes of Flash memory to be used to store user code
  - ✓ Configurable EEPROM size, 512bytes single page for being erased, which can be repeatedly erased more than 100 thousand times.
  - ✓ In-System-Programming, ISP in short, can be used to update the application code. No special programmer is needed.
  - ✓ Online debugging with single chip is supported, and no special emulator is needed. The number of breakpoints is unlimited theoratically.

- **SRAM**
  - ✓ 128 bytes internal direct access RAM (DATA, use keyword *data* to declare in C language program)
  - ✓ 128 bytes internal indirect access RAM (IDATA, use keyword *idata* to declare in C language program)
  - ✓ 4096 bytes internal extended RAM (internal XDATA, use keyword *xdata* to declare in C language program)
- **Clock**
  - ✓ Internal high precise RC clock IRC(IRC for short, ranges from 4MHz to 45MHz), adjustable while ISP and can be divided to lower frequency by user software, 100KHz for instance.
    - ✓ Error:±0.3% (at the temperature 25℃)
    - ✓ -1.35%～+1.30% temperature drift (at the temperature range of -40 ℃ to +85 ℃)
    - ✓ -0.76%～+0.98% temperature drift (at the temperature range of -20℃ to 65℃)
  - ✓ Internal 32KHz low speed IRC with large error
  - ✓ External crystal (4MHz～33MHz) and external clock

Users can freely choose the above 3 clock sources

- **Reset**
  - ✓ Hardware reset
    - ✓ Power-on reset. Measured voltage is 1.69V~1.82V. (Effective when the chip does not enable the low voltage reset function)
      The power-on reset voltage is a voltage range consisting of an upper limit voltage and a lower limit voltage. When the operating voltage drops from 5V / 3.3V to the lower limit threshold voltage of the power-on reset, the chip is in a reset state; when the voltage rises from 0V to the upper threshold voltage of power-on reset, the chip is released from the reset state.
    - ✓ Reset by reset pin. The default function of P5.4 is the I/O port. The P5.4 pin can be set as the reset pin while ISP download. (Note: When the P5.4 pin is set as the reset pin, the reset level is low.)
    - ✓ Watch dog timer reset
    - ✓ Low voltage detection reset. 4 low voltage detection levels are provided, 1.9V, 2.3V, 2.8V, 3.0V. Each level of low-voltage detection voltage is a voltage range consisting of an upper limit voltage and a lower limit voltage. When the operating voltage drops from 5V / 3.3V to the lower limit threshold voltage of low-voltage detection, the low-voltage detection takes effect. When the voltage rises from 0V to the upper threshold voltage, the low voltage detection becomes effective.
  - ✓ Software reset
    - ✓ Writing the reset trigger register using software
- **Interrupts**
  - ✓ 42 interrupt sources: INT0(Supports rising edge and falling edge interrupt), INT1(Supports rising edge and falling edge interrupt), INT2(Supports falling edge interrupt only), INT3(Supports falling edge interrupt only), INT4(Supports falling edge interrupt only), timer 0, timer 1, timer 2, timer 3, timer 4, UART 1, UART 2, UART 3, UART 4, ADC, LVD, SPI, I^{2}C, comparator, PWMA, PWMB, RTC, TKS, P1, P2, P3, P4, P5, P6, P7, LCM driver, DMA receive and transmit interrupts of UART 1, DMA receive and transmit interrupts of UART 2, DMA receive and transmit interrupts of UART 3, DMA receive and transmit interrupts of UART 4, DMA interrupt of SPI, DMA interrupt of ADC, DMA interrupt of LCM driver and DMA interrupt of memory-to-memory.
  - ✓ 4 interrupt priority levels
  - ✓ Interrupts that can wake up the CPU in clock stop mode: INT0(P3.2), INT1(P3.3), INT2(P3.6), INT3(P3.7), INT4(P3.0), T0(P3.4), T1(P3.5), T2(P1.2), T3(P0.4), T4(P0.6), RXD(P3.0/P3.6/P1.6/P4.3), RXD2(P1.0/P4.6), RXD3(P0.0/P5.0), RXD4(P0.2/P5.2), I2C_SDA(P1.4/P2.4/P3.3), SPI_SS(P5.4/P2.2/P3.5), Comparator interrupt, LVD interrupt, Power-down wake-up timer and interrupts of all I/O ports.
- **Digital peripherals**
  - ✓ 5 16-bit timers: timer0, timer1, timer2, timer3, timer4, where the mode 3 of timer 0 has the Non-Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
  - ✓ 4 high speed UARTs: UART1, UART2, UART3, UART4, whose maximum baudrate clock may be FOSC/4
  - ✓ 8 channels/2 groups of enhanced PWM, which can realize control signals with dead time, and support external fault detection function. In addition, supports 16-bit timers, 8 external interrupts, 8 external captures and pulse width measurement functions.
  - ✓ SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
  - ✓ I^{2}C: Master mode or slave mode are supported.

- ✓ MDU16: Hardware 16-bit Multiplier and Divider which supports 32-bit divided by 16-bit, 16-bit divided by 16-bit, 16-bit multiplied by 16-bit, data shift, and data normalization operations.
- ✓ RTC: Support year, month, day, hour, minute, second, sub-second (1/128 second). And supports clock interrupt and a set of alarm clocks (Note: A version of the chip does not have this function)
- ✓ I/O port interrupt: All I/Os support interrupts, each group of I/O interrupts has an independent interrupt entry address, all I/O interrupts can support 4 types interrupt mode: high level interrupt, low level interrupt, rising edge interrupt, falling edge interrupt. Provides 4 levels of interrupt priority and supports power-down wake-up function.
- ✓ DMA: support Memory-To-Memory, SPI, UART1TX/UART1RX, UART2TX/UART2RX, UART3TX/UART3RX, UART4TX/UART4RX, ADC(Automatically calculates the average of multiple ADC results), LCM
- ✓ LCM (TFT color screen) dirver: support 8080 and 6800 interface, and support 8-bit and 16-bit data width (Note: A version of the chip does not have this function)
  - ✓ 8 bits 8080 data bus: 8 bits data lines (TD0~TD7), READ signael (TRD)c WRITE signal (TWR), RS line (TRS)
  - ✓ 16 bits 8080 bus: 16 bits data lines (TD0~TD15), READ signael (TRD)c WRITE signal (TWR), RS line (TRS)
  - ✓ 8 bits 6800 bus: 8 bits data lines (TD0~TD7), enable signal (TE) , READ and WRITE signal (TRW) , RS line (TRS)
  - ✓ 16 bits 6800 bus: 16 bits data lines (TD0~TD15), enable signal (TE) , READ and WRITE signal (TRW) , RS line (TRS)
  - ✓ Note: If you use 8-bit data lines to control the TFT screen, you generally need TD0~D7, TRD/TWR/TRS, 11 data and control lines, plus 2 common I/Os to control chip selection and reset (many TFT color screen chip selections and reset manufacturer has carried out automatic processing, does not need software control)
- ✓ LCD dirver: support up to 4COM*40 SEGs and 8 levels grayscale adjustment

➢ **Analog peripherals**

- ✓ Ultra high speed ADC which supports 12-bit precision 15 channels (channel 0 to channel 14) analog-to-digital conversion. The maximum speed can be 800K(800K ADC conversions per second)
- ✓ ADC channel 15 is used to test the internal reference voltage. (The default internal reference voltage is 1.19V when the chip is shipped)
- ✓ Comparator. A set of comparator (The CMP+ port and all ADC input ports can be selected as the positive terminal of the comparator. So the comparator can be used as a multi-channel comparator for time division multiplexing)
- ✓ DAC: 8 channels advanced PWM timer can be used as 8 channels DAC

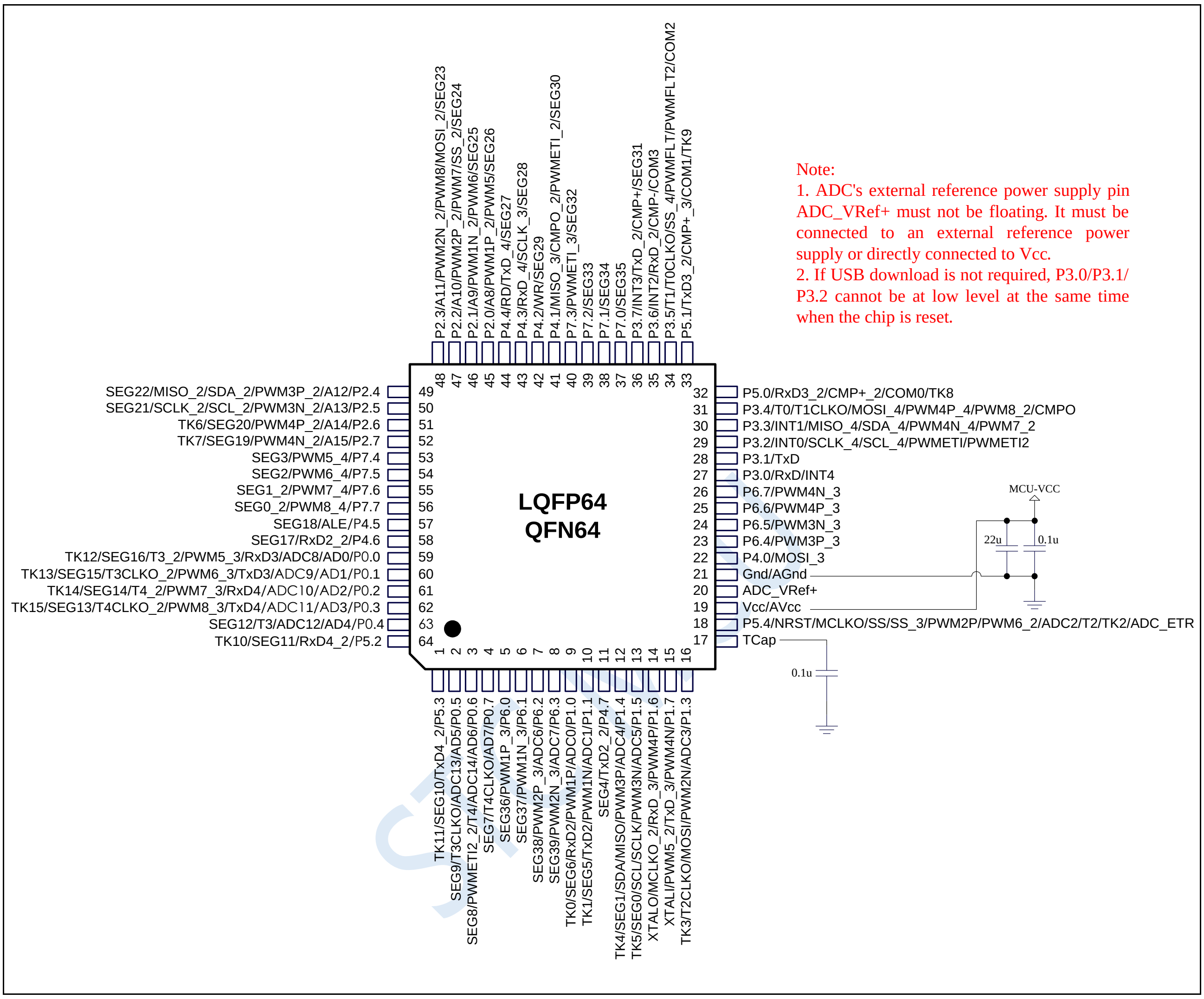
➢ **GPIO**

- ✓ Up to 61 GPIOs: P0.0~P0.7, P1.0~P1.7, P2.0~P2.7, P3.0~P3.7, P4.0~P4.7, P5.0~P5.4, P6.0~P6.7, P7.0~P7.7
- ✓ 4 modes for all GPIOs: quasi_bidirectional mode, push-pull outputmode, open drain mode, high-impedance input mode
- ✓ Except for P3.0 and P3.1, all other I/O ports are in a high-impedance state after power-on. User must set the I/O ports mode before using them. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.

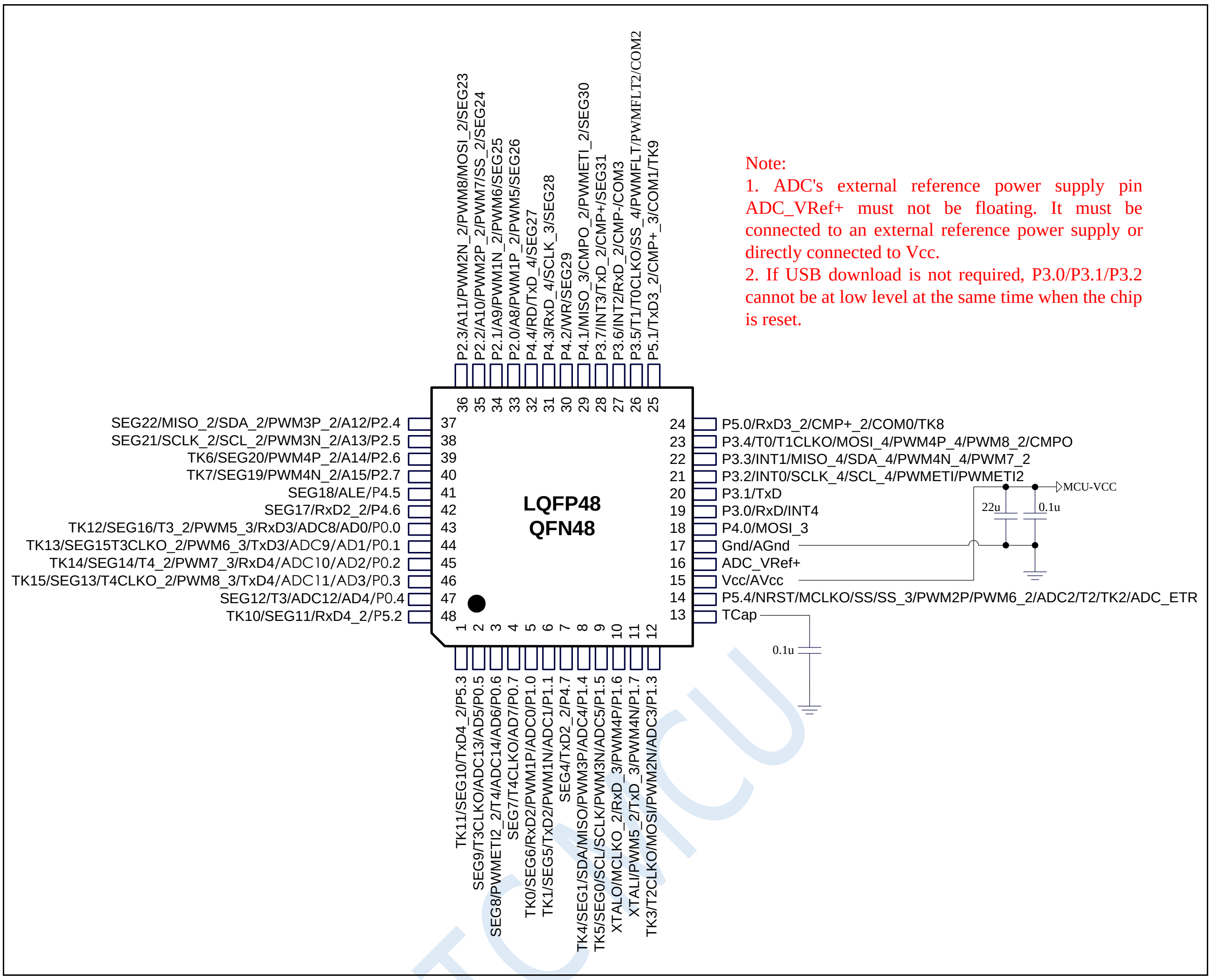
➢ **Package**

- ✓ LQFP64 <12mm*12mm>, QFN64 <8mm*8mm>, LQFP48 <9mm*9mm>, QFN48 <6mm*6mm>

Z.2.2 Pinouts


LQFP64
QFN64
1 TK11/SEG10/TxD4_2/P5.3
2 SEG9/T3CLKO/ADC13/AD5/P0.5
3 SEG8/PWMETI2_2/T4/ADC14/AD6/P0.6
4 SEG7/T4CLKO/AD7/P0.7
5 SEG36/PWM1P_3/P6.0
6 SEG37/PWM1N_3/P6.1
7 SEG38/PWM2P_3/ADC6/P6.2
8 SEG39/PWM2N_3/ADC7/P6.3
9 TK0/SEG6/RxD2/PWM1P/ADC0/P1.0
10 TK1/SEG5/TxD2/PWM1N/ADC1/P1.1
11 SEG4/TxD2_2/P4.7
12 TK4/SEG1/SDA/MISO/PWM3P/ADC4/P1.4
13 TK5/SEG0/SCL/SCLK/PWM3N/ADC5/P1.5
14 XTALO/MCLKO_2/RxD_3/PWM4P/P1.6
15 XTALI/PWM5_2/TxD_3/PWM4N/P1.7
16 TK3/T2CLKO/MOSI/PWM2N/ADC3/P1.3
17 TCap
18 P5.4/NRST/MCLKO/SS/SS_3/PWM2P/PWM6_2/ADC2/T2/TK2/ADC_ETR
19 Vcc/AVcc
20 ADC_VRef+
21 Gnd/AGnd
22 P4.0/MOSI_3
23 P6.4/PWM3P_3
24 P6.5/PWM3N_3
25 P6.6/PWM4P_3
26 P6.7/PWM4N_3
27 P3.0/RxD/INT4
28 P3.1/TxD
29 P3.2/INT0/SCLK_4/SCL_4/PWMETI/PWMETI2
30 P3.3/INT1/MISO_4/SDA_4/PWM4N_4/PWM7_2
31 P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO
32 P5.0/RxD3_2/CMP+_2/COM0/TK8
33 P5.1/TxD3_2/CMP+_3/COM1/TK9
34 P3.5/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2/COM2
35 P3.6/INT2/RxD_2/CMP-/COM3
36 P3.7/INT3/TxD_2/CMP+/SEG31
37 P7.0/SEG35
38 P7.1/SEG34
39 P7.2/SEG33
40 P7.3/PWMETI_3/SEG32
41 P4.1/MISO_3/CMPO_2/PWMETI_2/SEG30
42 P4.2/WR/SEG29
43 P4.3/RxD_4/SCLK_3/SEG28
44 P4.4/RD/TxD_4/SEG27
45 P2.0/A8/PWM1P_2/PWM5/SEG26
46 P2.1/A9/PWM1N_2/PWM6/SEG25
47 P2.2/A10/PWM2P_2/PWM7/SS_2/SEG24
48 P2.3/A11/PWM2N_2/PWM8/MOSI_2/SEG23
49 SEG22/MISO_2/SDA_2/PWM3P_2/A12/P2.4
50 SEG21/SCLK_2/SCL_2/PWM3N_2/A13/P2.5
51 TK6/SEG20/PWM4P_2/A14/P2.6
52 TK7/SEG19/PWM4N_2/A15/P2.7
53 SEG3/PWM5_4/P7.4
54 SEG2/PWM6_4/P7.5
55 SEG1_2/PWM7_4/P7.6
56 SEG0_2/PWM8_4/P7.7
57 SEG18/ALE/P4.5
58 SEG17/RxD2_2/P4.6
59 TK12/SEG16/T3_2/PWM5_3/RxD3/ADC8/AD0/P0.0
60 TK13/SEG15/T3CLKO_2/PWM6_3/TxD3/ADC9/AD1/P0.1
61 TK14/SEG14/T4_2/PWM7_3/RxD4/ADC10/AD2/P0.2
62 TK15/SEG13/T4CLKO_2/PWM8_3/TxD4/ADC11/AD3/P0.3
63 SEG12/T3/ADC12/AD4/P0.4
64 TK10/SEG11/RxD4_2/P5.2
MCU-VCC
22u
0.1u
0.1u

Note:
1. ADC's external reference power supply pin ADC_VRef+ must not be floating. It must be connected to an external reference power supply or directly connected to Vcc.
2. If USB download is not required, P3.0/P3.1/ P3.2 cannot be at low level at the same time when the chip is reset.


LQFP48
QFN48
P2.3/A11/PWM2N_2/PWM8/MOSI_2/SEG23 36
P2.2/A10/PWM2P_2/PWM7/SS_2/SEG24 35
P2.1/A9/PWM1N_2/PWM6/SEG25 34
P2.0/A8/PWM1P_2/PWM5/SEG26 33
P4.4/RD/TxD_4/SEG27 32
P4.3/RxD_4/SCLK_3/SEG28 31
P4.2/WR/SEG29 30
P4.1/MISO_3/CMPO_2/PWMETI_2/SEG30 29
P3.7/INT3/TxD_2/CMP+/SEG31 28
P3.6/INT2/RxD_2/CMP-/COM3 27
P3.5/T1/T0CLKO/SS_4/PWMFLT/PWMFLT2/COM2 26
P5.1/TxD3_2/CMP+_3/COM1/TK9 25
SEG22/MISO_2/SDA_2/PWM3P_2/A12/P2.4 37
SEG21/SCLK_2/SCL_2/PWM3N_2/A13/P2.5 38
TK6/SEG20/PWM4P_2/A14/P2.6 39
TK7/SEG19/PWM4N_2/A15/P2.7 40
SEG18/ALE/P4.5 41
SEG17/RxD2_2/P4.6 42
TK12/SEG16/T3_2/PWM5_3/RxD3/ADC8/AD0/P0.0 43
TK13/SEG15T3CLKO_2/PWM6_3/TxD3/ADC9/AD1/P0.1 44
TK14/SEG14/T4_2/PWM7_3/RxD4/ADC10/AD2/P0.2 45
TK15/SEG13/T4CLKO_2/PWM8_3/TxD4/ADC11/AD3/P0.3 46
SEG12/T3/ADC12/AD4/P0.4 47
TK10/SEG11/RxD4_2/P5.2 48
24 P5.0/RxD3_2/CMP+_2/COM0/TK8
23 P3.4/T0/T1CLKO/MOSI_4/PWM4P_4/PWM8_2/CMPO
22 P3.3/INT1/MISO_4/SDA_4/PWM4N_4/PWM7_2
21 P3.2/INT0/SCLK_4/SCL_4/PWMETI/PWMETI2
20 P3.1/TxD
19 P3.0/RxD/INT4
18 P4.0/MOSI_3
17 Gnd/AGnd
16 ADC_VRef+
15 Vcc/AVcc
14 P5.4/NRST/MCLKO/SS/SS_3/PWM2P/PWM6_2/ADC2/T2/TK2/ADC_ETR
13 TCap
MCU-VCC
22u
0.1u
0.1u
1 TK11/SEG10/TxD4_2/P5.3
2 SEG9/T3CLKO/ADC13/AD5/P0.5
3 SEG8/PWMETI2_2/T4/ADC14/AD6/P0.6
4 SEG7/T4CLKO/AD7/P0.7
5 TK0/SEG6/RxD2/PWM1P/ADC0/P1.0
6 TK1/SEG5/TxD2/PWM1N/ADC1/P1.1
7 SEG4/TxD2_2/P4.7
8 TK4/SEG1/SDA/MISO/PWM3P/ADC4/P1.4
9 TK5/SEG0/SCL/SCLK/PWM3N/ADC5/P1.5
10 XTALO/MCLKO_2/RxD_3/PWM4P/P1.6
11 XTALI/PWM5_2/TxD_3/PWM4N/P1.7
12 TK3/T2CLKO/MOSI/PWM2N/ADC3/P1.3

Note:

1. ADC's external reference power supply pin ADC_VRef+ must not be floating. It must be connected to an external reference power supply or directly connected to Vcc.

2. If USB download is not required, P3.0/P3.1/P3.2 cannot be at low level at the same time when the chip is reset.

Appendix AA Update Records

- **2022/3/9**
  1. Update data sheet
- **2021/12/21**
  1. STC8H8K64U adds LQFP32 and TSSOP20 pin diagram
  2. Correct typos in the document
  3. Add new product MCU notice information
- **2021/12/17**
  1. Revise the description of automatic switching from SPI master mode to slave mode
  2. Update RTC sample program
  3. Change the reset pin name to NRST in all pin diagrams
  4. Revise the timing calculation formula of timer 2/3/4
  5. Update the selection price list
  6. Update the pin diagram and pin description of STC8H8K64U
- **2021/11/23**
  1. Retype the EEPROM application example program
  2. Add appendix "STC Simulation User Manual" chapter
  3. Add detailed description to read-only special function register (CHIPID)
  4. Increase the use of VID and the allocation of PID in USB product development
  5. Add the pin diagram of LQFP32 and TSSOP20 of STC8H4K64TLR series
- **2021/11/16**
  1. Add the chapter "Unique ID Number and Important Parameters Stored in Read-Only Special Function Registers"
  2. Add a sample program for reading important parameters from read-only special function registers
  3. Add the appendix chapter of "USB Emulation Step Demonstration"
  4. Add EEPROM application example program
- **2021/10/27**
  1. Correct the description of LCD operating voltage setting register
  2. Update the interrupt structure diagram
  3. Correct typos in the document
  4. All BMMs are renamed to DMA
  5. Update the supply information of STC8H3K64S2 series and STC8H3K64S2 series
  6. Add the application precautions of using external crystal oscillator for RTC of STC8H4K64TLR system
- **2021/10/8**
  1. Update the port switching table for 8-bit data and 16-bit data in the LCM chapter
  2. Correct typos found in the document
  3. Add the description of external interrupts to the interrupt source table in the Interrupt System chapter
  4. Take screenshots using the new version of the software in the Emulator chapter in the appendix
  5. Added the chapter "How to Test I/O Ports" in the appendix
- **2021/9/26**
  1. Add a sample program for serial port DMA timeout processing and data verification
- **2021/8/30**
  1. Modify the title of some chapters
  2. Add the chapter "Notes on Baking Before Reflow Soldering" in the appendix

- **2020/8/26**

1. Add the chapter of timer calculation formula
2. Add the chapter of serial port baud rate calculation formula
3. Added 16-bit advanced PWM output frequency calculation formula chapter
4. Add ADC related calculation formula chapter
5. Add 12-bit ADC static parameter reference data
6. Add the parameter of the number of clocks required for MDU16 operation
7. Increase the time parameter required for EEPROM operation
8. The special function registers in all chapters are listed separately as directory subsections for easy searching
9. Precautions for adding STC8H series MCU to replace STC8A/8F series

- **2020/8/21**

1. Modify some errors in the description of the document
2. Added description to the 16-bit advanced PWM timer chapter
3. The first set of 16-bit advanced PWM timer PWM1 is renamed PWMA
4. The second group of 16-bit advanced PWM timer PWM2 is renamed to PWMB
5. Add "Using PWM to realize complementary SPWM" sample program

- **2020/8/10**

1. Add watchdog timer chapter
2. Organize the chapter on wake-up timer
3. Add the appendix chapter about STC download tool usage instructions

- **2020/8/6**

1. Explain the working temperature
2. Add a sample program of 16-bit advanced PWM timer for measuring period, duty cycle, high level and low level width
3. Added the description of the external 32.768KHz crystal oscillator control register X32KCR
4. Add the application circuit diagram downloaded using the universal USB to serial tool
5. Update application notes

- **2020/7/16**

1. Add description of BUS_SPEED register
2. Added soldering instructions for QFN/DFN packaged chips
3. Add reference circuit diagram of BLDC brushless DC motor drive (without HALL)
4. Add quadrature encoder mode sample program
5. Add the orthogonal decoding example provided by Chengdu Zhufei Technology, see Appendix L
6. Add EEPROM programming instructions
7. Add the method of setting U8W/U8-Mini to pass-through mode in the chapter of downloading ` application circuit diagram

- **2020/7/3**

1. Modify the problem of disordered layout of some text in the file

- **2020/7/2**

1. Add STC8H2K64T series
2. Add the appendix chapter, "How to reset the user program to the system area for ISP download without power failure"
3. Add the appendix chapter, "Develop your own ISP program using STC's IAP series MCU"
4. Add the appendix chapter, "Precautions for STC8H series MCU to replace STC15 series"
5. Add appendix chapter, "Official website description"
6. Add LED driver description chapter
7. Add touch button description chapter
8. Added RTC real-time clock description chapter
9. Add ADC conversion timing diagram in ADC chapter

- **2020/6/15**

1. Add ADC_VRef+ pin description
2. Added instructions for using diodes and resistors in the USB to serial reference circuit
3. Modify the description of the I/O port drive current control register PxDR (1: normal drive current; 0: strong drive current)
4. Add description of I2C slave device address

- **2020/6/8**

1. Add the description of the fastest ADC conversion speed
2. Detailed description of I2C bus speed setting
3. Update analog USB and hardware USB mode ISP download reference circuit diagram
4. Add the preset user-selectable internal IRC frequency description during hardware USB download
5. Add DFN8, QFN20, QFN32, QFN48, QFN64 substrate description in the package drawing
6. Add sample program for comparator multiplexing (comparator + ADC input) application

- **2020/5/29**

1. Add adding circuit application to ADC chapter
2. Add the description of register EAXFR
3. Correct the errors in the DFN8 package dimension drawing
4. Add the method of using a third-party application to call the release project program

- **2020/5/25**

1. Add negative voltage detection circuit in ADC chapter
2. Fix garbled characters in some pictures

- **2020/5/20**

1. Update the power consumption parameters of the clock stop mode when the low-voltage detection wake-up function is enabled in the electrical characteristics
2. Update the power consumption parameters of the clock stop mode when the comparator power-down wake-up function is enabled in the electrical characteristics
3. The ADCTIM register is added to the ADC sample program to control the ADC internal timing
4. Correct some clerical errors in the document
5. Add an interrupt that can be used to wake up from clock stop mode in the features of each microcontroller series
6. Add sample program for I/O port interrupt
7. Added ISP download step guide in the ISP download application circuit diagram
8. Add power-down wake-up timer to wake up the power saving mode sample program

- **2020/5/14**

9. Add description of comparator multiplexing
10. Add PWM trigger ADC sample program
11. Added ADC working clock frequency description in ADC chapter
12. Add ADC reference circuit diagram in ADC chapter
13. Update the power consumption parameters of low voltage detection and comparator in electrical characteristics
14. Update the reference price of STC8H1K08 series
15. Add reference circuit diagram of PWM as DAC
16. Update the pin diagram and the description of the PWM external trigger pin PWMETI in the pin description
17. Add power-on reset and button reset reference circuit diagram

- **2020/4/29**

1. Change the serial port download reference circuit diagram, the series resistance on the TxD pin of the MCU is changed from 300 ohms to 100 ohms
2. Fix the error in the power supply part of the reference circuit diagram using PL2303GL for ISP download

- **2020/4/26**

1. Update I/O speed parameters in electrical characteristics
2. Add the reference pin diagram of PDIP40 of STC8H8K64U series
3. Update the speed parameter of the comparator in the electrical characteristics
4. Correct the timing of setting TI and RI in Chapter 14.6 Serial Notes
5. Added the description about analog filtering and digital filtering in the chapter of comparator
6. In Appendix E, the connection error between MAX232 and RS485 is corrected

- **2020/4/8**

1. Add the description of the power-down wake-up timer register
2. Update the content about the overall drive current in the I/O port chapter

- **2020/3/27**

1. Delete STC8H8K64S2U series
2. STC8H8K64S4U series was renamed STC8H8K64U series
3. The IRC24MCR register is renamed HIRCCR

4. Add STC8H8K64S4U model and use PL2303GL to download the reference circuit diagram
5. Add STC8H8K64S4U model direct hardware USB download reference circuit diagram
6. Rename the power-related pin names of all chips in a unified style
7. Update the power consumption of the chip in the DC characteristics at different operating frequencies
8. Added a description at the beginning of the advanced PWM timer chapter
9. Update STC8H8K64U series selection price list
10. Correct the formula for calculating voltage in the chapter "Using ADC channel 15 to measure external voltage or battery voltage"

- **2020/3/13**

1. Re-calibrate the content in the advanced PWM timer chapter
2. Add the sample program of advanced PWM timer used as external interrupt

- **2020/3/6**

1. Correct the incorrect description of the internal reference signal source in the document
2. Add application circuit diagrams of general precision ADC and high precision ADC
3. Add static parameters of ADC module
4. Added STC8H8K64S4U series LQFP48 pin diagram and pin description
5. Added STC8H8K64S2U series LQFP48 pin diagram and pin description
6. Delete STC8H3K64S4 series
7. Delete STC8H3K64S2 series
8. Added I/O Port Interrupt Chapter
9. Add the description of I/O interrupt in the interrupt system chapter
10. Add USB sample program (HID interface)
11. Reorganized the chapter structure of the pin diagram
12. Correct the chip characteristics of STC8H1K17 model

- **2020/1/20**

1. Add sample code related to advanced PWM
2. Add "a typical triode control" circuit
3. Add "typical LED control" circuit
4. Add the reference circuit of "I/O port interconnection of 3V/5V devices in mixed voltage power supply system"
5. Add the reference circuit of "How to make the I/O port be low when power-on reset"
6. Add "Using 74HC595 to drive 8 digital tubes (serial expansion, 3 lines)" reference circuit
7. Add "I/O port directly drive LED digital tube" reference circuit
8. Added the description of "Automatically start ISP download after receiving user command when running user program"

- **2020/1/17**

1. Add MDU16 operation clock description
2. Added STC8H1K08 series QFN20 pin diagram

- **2020/1/15**

1. Add "ADC as capacitive sensing touch button" chapter
2. Add "ADC as key scan application circuit diagram" chapter
3. Add appendix "RS485 automatic control or I/O port control circuit diagram"
4. Add appendix "RS485 partial circuit diagram in U8W download tool"

- **2019/12/31**

1. Modified the number of I/O ports of STC8H8K64S2U series and STC8H8K64S4U series, the actual number is 60 I/O at most

- **2020/12/30**

1. Create STC8H series MCU technical reference manual document
2. Add STC8H1K28 series
3. Add STC8H1K08 series
4. Add STC8H3K64S4 series
5. Add STC8H3K64S2 series
6. Add STC8H8K64S4U series
7. Add STC8H8K64S2U series
8. Add MDU16 multiplication and division unit description

Appendix BB STC8 series naming tidbits

STC8A: The letter "A" stands for ADC, which is the starting product of STC 12-bit ADC

STC8F: No ADC, PWM and PCA functions, the current version of the STC8F chip is fully compatible with the original STC8F pins, but the internal design has been optimized and updated, the user needs to modify the program, so it is named STC8C

STC8C: The letter "C" stands for revision, which is a revised chip of STC8F

STC8G: The letter "G" was originally a typo when the chip was produced. Later, the G series was defined as the "GOOD" series. The STC8G series is easy to learn.

STC8H: The letter "H" is taken from the first letter of the English word "High", and "High" means "16-bit advanced PWM"