Shenzhen Guoxin Artificial Intelligence Co.

STC8A8K64D4 Series Microcontrollers Technical Reference Manual

Technical support website:
www.STCAI.com资料更新日期:

2022/11/14

table of contents

1 Overview

The STC8A8K64D4 series microcontroller is a microcontroller that does not require an external crystal and an external reset, and is an 8051 microcontroller that targets super anti-interference/ultra-low cost/high speed/low power consumption. At the same operating frequency, the STC8A8K64D4 series microcontroller is approximately 12 times faster than the conventional 8051 (11.2~13.2 times faster), and it takes only 147 clocks to execute all 111 instructions sequentially, while the conventional 8051 requires 1944 clocks. The STC8A8K64D4 series microcontroller is a single clock/machine cycle (1T) microcontroller produced by STC, which is a new generation 8051 microcontroller with wide voltage/high speed/high reliability/low power consumption/strong anti-static/strong anti-interference and super encryption. Instruction code is fully compatible with traditional 8051.

MCU with internal high precision R/C clock (±0.3%, +25° C at room temperature), -1.38% to +1.42% temperature drift (-40° C to +85° C)
-0.88% to +1.05% temperature drift (-20° C to +65° C). 4MHz to 45MHz wide range configurable for ISP programming (Note: temperature range is -40° C)
The maximum frequency must be kept below 45MHz at ~+85° C), eliminating the need for an expensive external crystal and external reset circuitry (highly reliable reset circuitry has been integrated internally, and a 4-stage reset threshold voltage is optional for ISP programming).

There are three optional clock sources inside the MCU: internal high precision IRC clock (adjustable during ISP download), internal 32KHz low speed IRC, external 4M~33M crystal or external clock signal. Clock source can be freely selected in the user code, and the clock source can be divided by an 8-bit divider before providing the clock signal to the CPU and various peripherals (such as timer, serial port, SPI, etc.).

The MCU provides two low-power modes: IDLE mode and STOP mode. In IDLE mode, the MCU stops providing clock to the CPU, the CPU has no clock, and the CPU stops executing instructions, but all peripherals are still working, and the power consumption is about 1.0mA (6MHz operating frequency). The power consumption can be reduced to 0.6uA@Vcc=5.0V and 0.4uA@Vcc=3.3V.

掉电模式可以使用 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/P4.0)、RXD3(P0.0/P5.0)、RXD4(P0.2/P5.0)、CCP0(P1.7/P2.3/P7.0/P3.3)、CCP1 (P1.6/P2.4/P7.1/P3.2), CCP2 (P1.5/P2.5/P7.2/P3.1), CCP3 (P1.4/P2.6/P7.3/P3.0), I2C_SDA (P1.4/P2.4/P3.3), SPI_SS (P1.2/P2.2/P3.5), and all ports I/O interrupts, the
Comparator interrupt, low voltage detection interrupt, and power-down wake-up timer wake-up.

The MCU provides a rich set of digital peripherals (serial, timer, PCA, enhanced PWM, and I^2C, SPI) to interface with the analog peripherals
(12-bit*15-way ultra-high-speed ADC and comparator with a speed of up to 800K samples per second) can meet the design needs of a wide range of users.

The STC8A8K64D4 series microcontroller has an enhanced dual data pointer integrated inside. Through program control, the data pointer can be automatically incremented or

decremented and the two data pointers can be automatically switched.

product lines	I/O	UART	timers	ADC	enhanced PWM	PCA	CMP	SPI	I2C	MDU16	I/O disruptions	LCM	DMA
STC8A8K64D4 Series-64Pin/48Pin	59	4	5	15CH*12B	●	●	●	●	●	●	●	●	●

2 Features, Prices and Tubes feet

2.1 STC8A8K64D4-LQFP64/48/44, PDIP40 Series

2.1.1 Features and pricing (with 16-bit hardware multiplier and divider MDU16,quasi-16-bit microcontroller)

- Selection price (no external crystal, no external reset required)

Microcontroller Model	Operating Voltage (V)	xdata SRAM	Flash	EEPROM	I/O	8080/6800 LCM	DMA	I²C	SPI	PCA/CCP/PWM	15	Comparator	Hardware RTC	Touch Key	LCD	USB	RS485	Partial package and tax inclusive prices						2021 New Product Availability Information
																		PDIP40	LQFP44	QFN48 <6x6mm>	LQFP48	QFN64 <8x8mm>	LQFP64	
STC8A8K16D4	1.9-5.5	16K	8K	2	48K	59	4	Yes	Yes	Yes	Yes	Yes	Yes	Yes	5	8	4	Yes	Yes	Yes	Yes	Yes	4 levels	Yes	Yes	Yes	√	√	√	√	√	√		
STC8A8K32D4	1.9-5.5	32K	8K	2	32K	59	4	Yes	Yes	Yes	Yes	Yes	Yes	Yes	5	8	4	Yes	12 bits	Yes	Yes	Yes	4 levels	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes		
STC8A8K48D4	1.9-5.5	48K	8K	2	16K	59	4	Yes	Yes	Yes	Yes	Yes	Yes	Yes	5	8	4	Yes	12 bits	Yes	Yes	Yes	4 levels	Yes	Yes	Yes	Yes	Yes	Yes	¥4.5	√	¥3.9	√	¥3.9	
STC8A8K60D4	1.9-5.5	60K	8K	2	4K	59	4	Yes	Yes	Yes	Yes	Yes	Yes	Yes	5	8	4	Yes	12 bits	Yes	Yes	Yes	4 levels	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes		
STC8A8K64D4	1.9-5.5	64K	8K	2	IAP	59	4	Yes	Yes	Yes	Yes	Yes	Yes	Yes	5	8	4	Yes	12 bits	Yes	Yes	Yes	4 levels	Yes	Yes	Yes	Yes	Yes	Yes	¥4.5	√	¥3.9	√	¥3.9	

- ✓ Ultra-fast 8051 core (1T), approximately 12x faster than conventional 8051 kernels
- ✓ Instruction code is fully compatible with legacy 8051
- ✓ 43 interrupt sources, 4 levels of interrupt priority
- ✓ Support for online simulation

- Operating Voltage
  - ✓ 1.9V to 5.5V
  - ✓ Built-in LDO
- Operating temperature
  - ✓ -40° C to 85° C (for applications over the temperature range, please refer to the description in the Electrical Characteristics section)
- **Flash Memory**
  - ✓ Up to 64K bytes of FLASH program memory (ROM) for storing user code
  - ✓ User-configurable EEPROM size, 512 bytes single page erase, over 100,000 erase cycles
  - ✓ Supports updating user applications in the system programming mode (ISP) without a dedicated programmer
  - ✓ Support single-chip emulation, no special emulator required, no limit to the number of theoretical breakpoints
- **SRAM**

- ✓ 128 bytes of internal direct access RAM (DATA)
- ✓ 128 bytes of internal indirect access RAM (IDATA)

- ✓ 8192 bytes of internal extended RAM (internal XDATA)

➢ Clock control

- ✓ Internal high-precision IRC (4MHz to 45MHz, adjustable up or down during ISP programming, and user software dividable to lower frequencies, e.g. 100KHz)
  Error ±0.3% (25° C at room temperature)
  -1.38% to +1.42% temperature drift (full temperature range, -40° C to 85° C)
  -0.88% to +1.05% temperature drift (temperature range, -20° C to 65° C)
- ✓ Internal 32KHz low-speed IRC (large error)
- ✓ External crystal (4MHz to 45MHz) and external clock

➢ reset (a dislocated joint)

- ✓ Hardware Reset
  Power-on reset, measured voltage 1.69V to 1.82V. (Valid when the chip is not enabled for low-voltage reset)
  The power-on reset voltage consists of a voltage range of an upper limit voltage and a lower limit voltage. When the operating voltage drops down from 5V/3.3V to the lower threshold voltage of the power-on reset, the chip is in the reset state; when the voltage rises from 0V to the upper threshold voltage of the power-on reset, the chip is released from the reset state.
  Reset pin reset, factory default P5.4 is I/O port, ISP download can set P5.4 pin as reset pin (Note: when set P5.4
  (When the pin is a reset pin, the reset level is low)
  Watchdog overflow reset
  Low voltage detection reset, providing 4 levels of low voltage detection voltage: 2.0V (measured 1.90V to 2.04V),2.4V (measured 2.30V to 2.50V), and
  2.7V (measured 2.61V to 2.82V), 3.0V (measured 2.90V to 3.13V).
  Each low voltage detection voltage stage is a voltage range consisting of an upper voltage and a lower voltage, with low voltage detection taking effect when the operating voltage drops down from 5V/3.3V to the lower threshold voltage for low voltage detection, and when the voltage rises from 0V to the upper threshold voltage for low voltage detection.
- ✓ Software Reset
  Software method to write reset trigger register

➢ disruptions

- ✓ Provides 43 interrupt sources: INT0 (supports rising and falling edge interrupts), INT1 (supports rising and falling edge interrupts), INT2 (supports falling edge interrupts only), INT3 (supports falling edge interrupts only), INT4 (supports falling edge interrupts only), Timer 0, Timer 1, Timer 2, Timer 3, Timer 4, Serial 1, Serial 2, Serial 3, Serial 4 ADC Analog to Digital Conversion, LVD Low Voltage Detection, SPI, I2C, Comparator, PCA/CCP/PWM, Enhanced PWM, Enhanced PWM Exception Detection, all I/O interrupts (8 groups), LCD driver interrupts, DMA receive and transmit interrupts for serial port 1, DMA receive and transmit interrupts for serial port 2, DMA receive and transmit interrupts for serial port 3, DMA receive and transmit interrupts for serial port 4, DMA interrupts for SPI, DMA interrupts for ADC DMA interrupt for SPI, DMA interrupt for ADC, DMA interrupt for LCD driver, and memory-to-memory DMA interrupt.
- ✓ Provides 4 levels of interrupt priority

Interrupts that can wake up 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/P4.0), RXD3 (P0.0/P5.0), RXD4 (P0.2/P5.0), CCP0 (P1.7/P2.3/P7.0/P3.3), CCP1 (P1.6/P2.4/P7.1/P3.2), CCP2 (P1.5/P2.5/P7.2/P3.1), CCP3 (P1.4/P2.6/P7.3/P3.0), I2C_SDA (P1.4/P2.4/P3.3), SPI_SS (P1.2/P2.2/P3.5), and I/O interrupts for all ports, and Comparator interrupt, low voltage detection interrupt, power-down wake-up timer wake-up

- digital peripheral
  - ✓ 5 16-bit timers: Timer 0, Timer 1, Timer 2, Timer 3, Timer 4, with Mode 3 of Timer 0 having NMI (non-maskable interrupt) function, mode 0 of timer 0 and timer 1 is 16-bit auto-reload mode
  - ✓ 4 high-speed serial ports: serial port 1, serial port 2, serial port 3, serial port 4, baud rate clock source up to FOSC/4
  - ✓ Four 16-bit PCA modules: CCP0, CCP1, CCP2, CCP3, for capture, high-speed pulse output, and 6/7/8/10-bit PWM

exports

- ✓ 8 groups of 15-bit enhanced PWM for control signals with deadband and support for external anomaly detection, plus 4 groups of conventional PCA/CCP/PWM can be used as PWM
- ✓ SPI: Supports host and slave modes and automatic host/slave switching
- ✓ I^2C: Master and slave modes supported
- ✓ **MDU16:** Hardware 16-bit multiplier and divider (supports 32-bit divide by 16-bit, 16-bit divide by 16-bit, 16-bit by 16-bit, data shift, and data normalization)
- ✓ **I/O port interrupts:** all I/Os support interrupts, each I/O interrupt has a separate interrupt entry address, all I/O interrupts can support 4 interrupt modes: high interrupt, low interrupt, rising edge interrupt, falling edge interrupt

  (I/O port interrupts in this series are power-down wake-up capable and have 4 levels of interrupt priority)
- ✓ **LCD driver module:** supports both 8080 and 6800 interfaces and 8 and 16 bit data widths
- ✓ **DMA:** Support SPI shift to receive data to memory, SPI shift to send data to memory, serial port 1/2/3/4 to receive data to memory, and Serial port 1/2/3/4 sends data from memory, ADC automatically samples data to memory (while calculating average values), LCD driver sends data from memory, and memory-to-memory data copy
- ✓ **Hardware numeric ID:** 32 bytes supported

➢ Analog Peripherals

- ✓ Ultra-high-speed ADC supporting **12-bit high-precision** 15-channel (channel 0 to channel 14) analog-to-digital conversion **at speeds up to 800K (800,000 ADC conversions per second)**
- ✓ Channel 15 of the ADC is used to test the internal 1.19V reference source (the chip is shipped with the internal reference source adjusted to 1.19V)
- ✓ Comparator, a set of comparators (the positive side of the comparator selects the CMP+ port, CMP+_2, CMP+_3 and all ADC input ports, and the negative side of the comparator selects the CMP- port and the internal 1.19V reference source, so the comparator can be used as a multiplexer for time division)
- ✓ DACs: 8 enhanced PWM timers can be used as 8 DACs, 4 PCAs can be used as 4 DACs

➢ **GPIO**

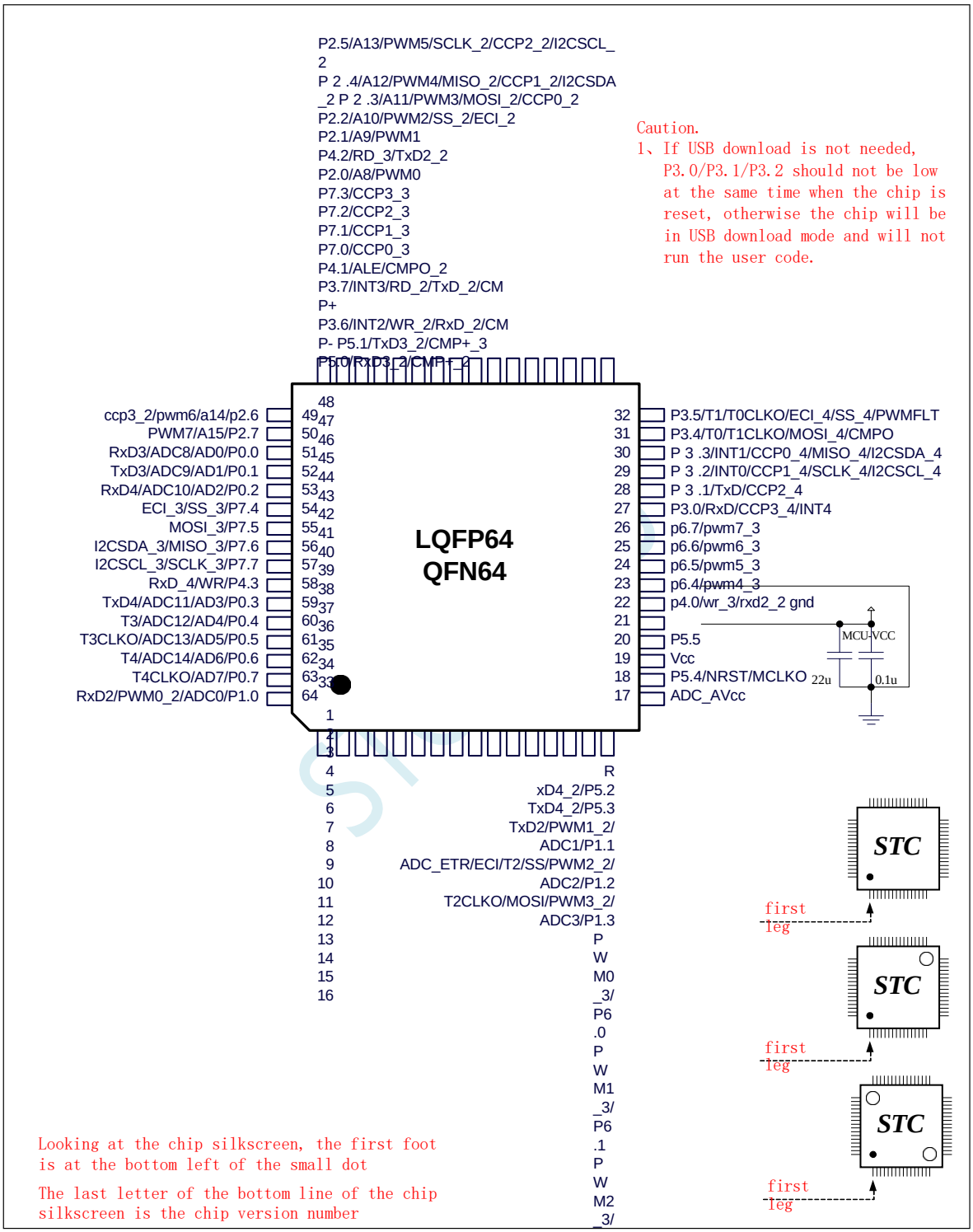
- ✓ Up to 59 GPIOs: P0.0~P0.7, P1.0~P1.7, P2.0~P2.7, P3.0~P3.7, P4.0~P4.4, P5.0~P5.5, P6.0~P6.7, P7.0~P7.7
- ✓ All GPIOs support the following 4 modes: quasi-bidirectional port mode, strong push-pull output mode, open-drain output mode, and high-resistance input mode
- ✓ **Except for P3.0 and P3.1, the status of all I/O ports after power up is high resistance input state, user must set I/O first when using I/O port**

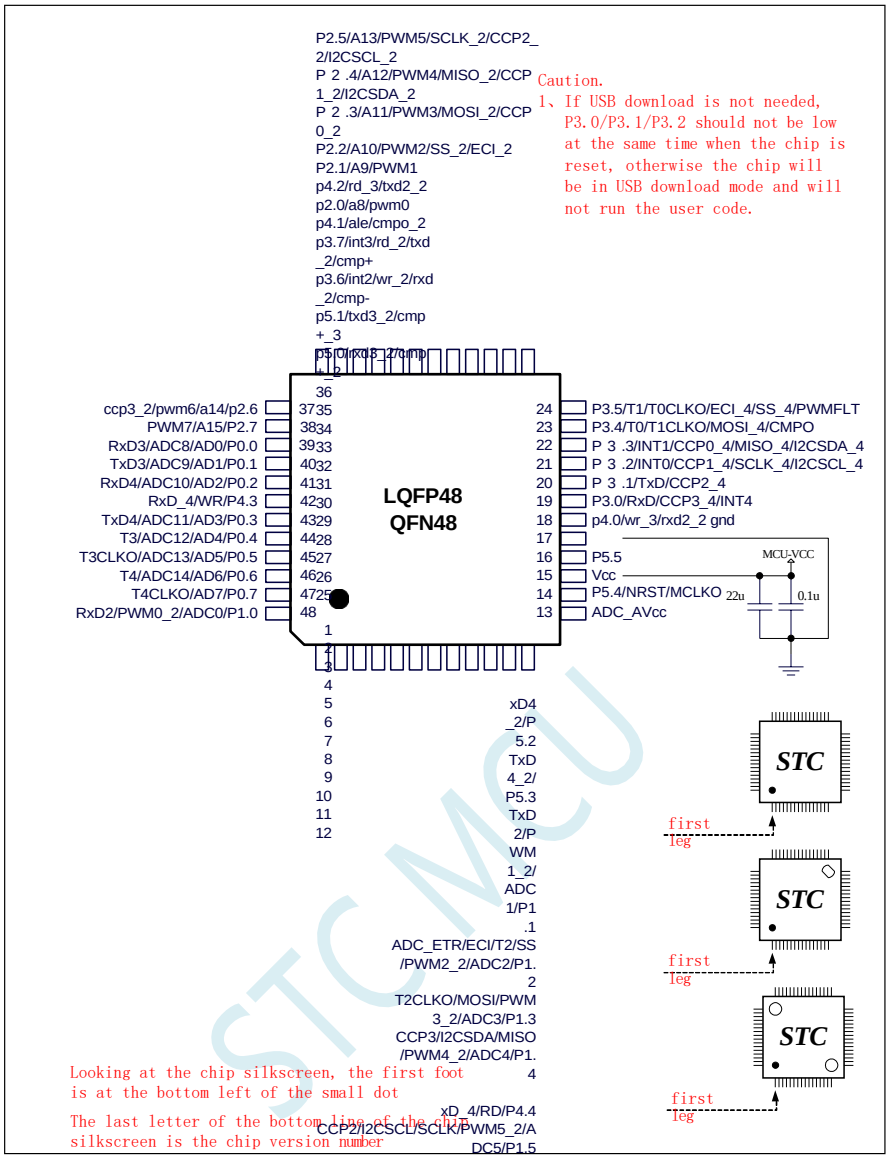
  In addition, each I/O can be independently enabled with an internal 4K pull-up resistor

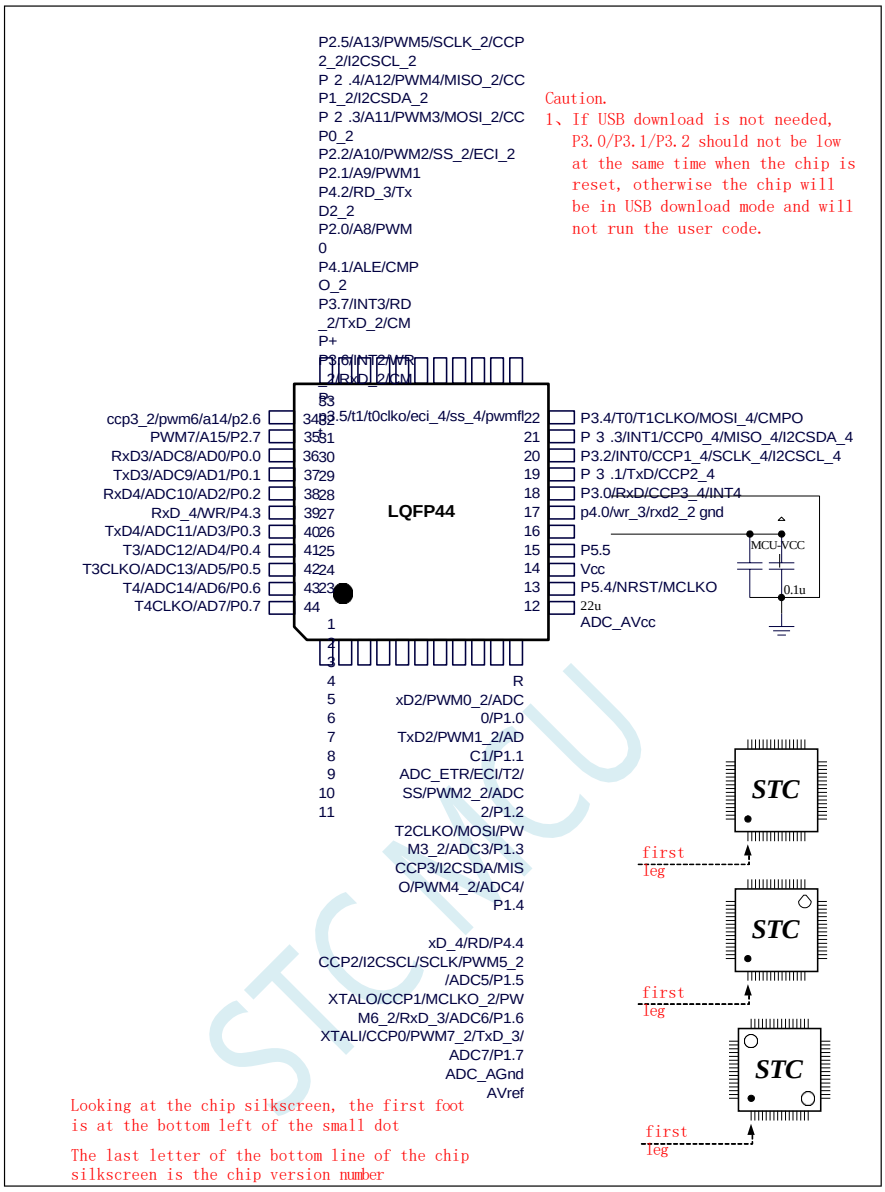
➢ package

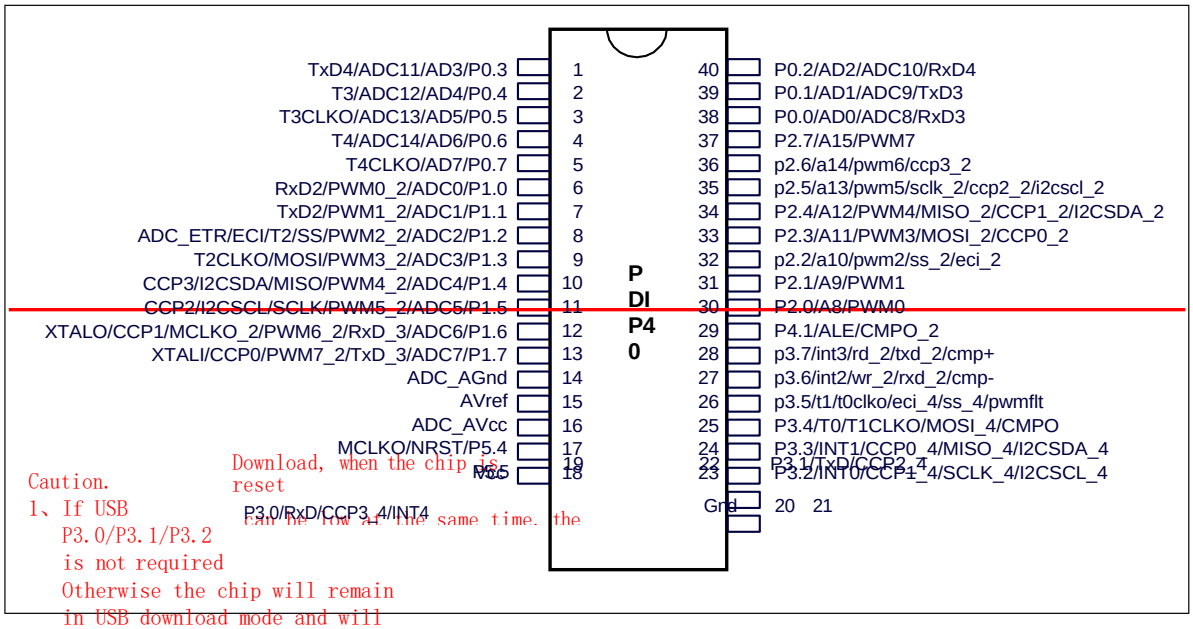
- ✓ LQFP64, LQFP48, LQFP44

2.1.2 Pinout Diagram, Minimum System




P2.5/A13/PWM5/SCLK_2/CCP2_2/I2CSCL_2
P 2 .4/A12/PWM4/MISO_2/CCP1_2/I2CSDA_2
P 2 .3/A11/PWM3/MOSI_2/CCP0_2
P2.2/A10/PWM2/SS_2/ECI_2
P2.1/A9/PWM1
p4.2/rd_3/txd2_2
p2.0/a8/pwm0
p4.1/ale/cmpo_2
p3.7/int3/rd_2/txd_2/cmp+
p3.6/int2/wr_2/rxd_2/cmp-
p5.1/txd3_2/cmp+_3
p5.0/rxd3_2/cmp+_2
Caution.
1、If USB download is not needed, P3.0/P3.1/P3.2 should not be low at the same time when the chip is reset, otherwise the chip will be in USB download mode and will not run the user code.
LQFP48
QFN48
ccp3_2/pwm6/a14/p2.6
PWM7/A15/P2.7
RxD3/ADC8/AD0/P0.0
TxD3/ADC9/AD1/P0.1
RxD4/ADC10/AD2/P0.2
RxD_4/WR/P4.3
TxD4/ADC11/AD3/P0.3
T3/ADC12/AD4/P0.4
T3CLKO/ADC13/AD5/P0.5
T4/ADC14/AD6/P0.6
T4CLKO/AD7/P0.7
RxD2/PWM0_2/ADC0/P1.0
P3.5/T1/T0CLKO/ECI_4/SS_4/PWMFLT
P3.4/T0/T1CLKO/MOSI_4/CMPO
P 3 .3/INT1/CCP0_4/MISO_4/I2CSDA_4
P 3 .2/INT0/CCP1_4/SCLK_4/I2CSCL_4
P 3 .1/TxD/CCP2_4
P3.0/RxD/CCP3_4/INT4
p4.0/wr_3/rxd2_2 gnd
P5.5
Vcc
P5.4/NRST/MCLKO
ADC_AVcc
MCU-VCC
22u
0.1u
xD4_2/P5.2
TxD4_2/P5.3
TxD2/PWM1_2/ADC1/P1.1
ADC_ETR/ECI/T2/SS/PWM2_2/ADC2/P1.2
T2CLKO/MOSI/PWM3_2/ADC3/P1.3
CCP3/I2CSDA/MISO/PWM4_2/ADC4/P1.4
xD_4/RD/P4.4
CCP2/I2CSCL/SCLK/PWM5_2/ADC5/P1.5
XTALO/CCP1/MCLKO_2/PWM6_2/RxD_3/ADC6/P1.6
XTALI/CCP0/PWM7_2/TxD_3/ADC7/P1.7
ADC_AGnd
AVref
Looking at the chip silkscreen, the first foot is at the bottom left of the small dot
The last letter of the bottom line of the chip silkscreen is the chip version number
STC
first leg
STC
first leg
STC
first leg


P2.5/A13/PWM5/SCLK_2/CCP2_2/I2CSCL_2
P 2 .4/A12/PWM4/MISO_2/CCP1_2/I2CSDA_2
P 2 .3/A11/PWM3/MOSI_2/CCP0_2
P2.2/A10/PWM2/SS_2/ECI_2
P2.1/A9/PWM1
P4.2/RD_3/Tx D2_2
P2.0/A8/PWM 0
P4.1/ALE/CMP O_2
P3.7/INT3/RD _2/TxD_2/CM P+
P3.6/INT2/WR _2/RxD_2/CM P-
P3.5/T1/T0CLKO/ECI_4/SS_4/PWMFLT
P3.4/T0/T1CLKO/MOSI_4/CMPO
P 3 .3/INT1/CCP0_4/MISO_4/I2CSDA_4
P3.2/INT0/CCP1_4/SCLK_4/I2CSCL_4
P 3 .1/TxD/CCP2_4
P3.0/RxD/CCP3_4/INT4
p4.0/wr_3/rxd2_2 gnd
P5.5
Vcc
P5.4/NRST/MCLKO
22u
ADC_AVcc
MCU-VCC
0.1u
LQFP44
ccp3_2/pwm6/a14/p2.6
PWM7/A15/P2.7
RxD3/ADC8/AD0/P0.0
TxD3/ADC9/AD1/P0.1
RxD4/ADC10/AD2/P0.2
RxD_4/WR/P4.3
TxD4/ADC11/AD3/P0.3
T3/ADC12/AD4/P0.4
T3CLKO/ADC13/AD5/P0.5
T4/ADC14/AD6/P0.6
T4CLKO/AD7/P0.7
R
xD2/PWM0_2/ADC 0/P1.0
TxD2/PWM1_2/AD C1/P1.1
ADC_ETR/ECI/T2/ SS/PWM2_2/ADC 2/P1.2
T2CLKO/MOSI/PW M3_2/ADC3/P1.3
CCP3/I2CSDA/MIS O/PWM4_2/ADC4/ P1.4
xD_4/RD/P4.4
CCP2/I2CSCL/SCLK/PWM5_2 /ADC5/P1.5
XTALO/CCP1/MCLKO_2/PW M6_2/RxD_3/ADC6/P1.6
XTALI/CCP0/PWM7_2/TxD_3/ ADC7/P1.7
ADC_AGnd
AVref
Caution.
1、If USB download is not needed, P3.0/P3.1/P3.2 should not be low at the same time when the chip is reset, otherwise the chip will be in USB download mode and will not run the user code.
STC
first leg
STC
first leg
STC
first leg
Looking at the chip silkscreen, the first foot is at the bottom left of the small dot
The last letter of the bottom line of the chip silkscreen is the chip version number



Caution.
1、If USB P3.0/P3.1/P3.2 is not required Otherwise the chip will remain in USB download mode and will not run user code

Download, when the chip is reset

Can be low at the same time, the

2.1.3 Pin Description

number				name (of a thing)	types	instructions
LQFP64	LQFP48	LQFP44	PDIP40			
1	1			P5.2	I/O	Standard IO port
				RxD4_2	I	Receive pin of serial port 4
2	2			P5.3	I/O	Standard IO port
				TxD4_2	O	Transmit pin of serial port 4
3	3	2	7	P1.1	I/O	Standard IO port
				ADC1	I	ADC Analog Input Channel 1
				PWM1_2	O	Enhanced PWM channel 1 output pin
				TxD2	O	Transmit pin of serial port 2
4	4	3	8	P1.2	I/O	Standard IO port
				ADC2	I	ADC analog input channel 2
				PWM2_2	O	Enhanced PWM channel 2 output pin
				SS	I/O	SPI Slave Selection
				T2	I	Timer 2 External clock input
				ECI	I	External pulse input for PCA
				ADC_ETR	I	ADC external trigger input
5	5	4	9	P1.3	I/O	Standard IO port
				ADC3	I	ADC analog input channel 3
				PWM3_2	O	Enhanced PWM channel 3 output pin
				MOSI	I/O	SPI host output slave input
				T2CLKO	O	Timer 2 clock division output
6				P6.0	I/O	Standard IO port
				PWM0_3	O	Enhanced PWM Channel 0 Output Pin
7				P6.1	I/O	Standard IO port
				PWM1_3	O	Enhanced PWM channel 1 output pin
8				P6.2	I/O	Standard IO port
				PWM2_3	O	Enhanced PWM channel 2 output pin
9				P6.3	I/O	Standard IO port
				PWM3_3	O	Enhanced PWM channel 3 output pin

number				name (of a thing)	types	instructions
LQFP64	LQFP48	LQFP44	PDIP40			
10	6	5	10	P1.4	I/O	Standard IO port
				ADC4	I	ADC analog input channel 4
				PWM4_2	O	Enhanced PWM channel 4 output pin
				MISO	I/O	SPI host input slave output
				SDA	I/O	Data line for I2C interface
				CCP3	I/O	Capture input and pulse output of PCA
11	7	6		P4.4	I/O	Standard IO port
				RD	O	Read signal line for external bus
				TxD_4	O	Transmit pin of serial port 1
12	8	7	11	P1.5	I/O	Standard IO port
				ADC5	I	ADC analog input channel 5
				PWM5_2	O	Enhanced PWM channel 5 output pin
				SCLK	I/O	Clock pin of SPI
				SCL	I/O	Clock line for I2C
				CCP2	I/O	Capture input and pulse output of PCA
13	9	8	12	P1.6	I/O	Standard IO port
				ADC6	I	ADC Analog Input Channel 6
				RxD_3	I	Receive pin of serial port 1
				PWM6_2	O	Enhanced PWM channel 6 output pin
				MCLKO_2	O	Master clock divider output
				CCP1	I/O	Capture input and pulse output of PCA
				XTALO	O	Output pin of the external crystal
14	10	9	13	P1.7	I/O	Standard IO port
				ADC7	I	ADC Analog Input Channel 7
				TxD_3	O	Transmit pin of serial port 1
				PWM7_2	O	Enhanced PWM channel 7 output pin
				CCP0	I/O	Capture input and pulse output of PCA
				XTALI	I	Input pin for external crystal/external clock
15	11	10	14	ADC_AGnd	GND	ADC Ground
16	12	11	15	AVref	I	Reference voltage pin of ADC
17	13	12	16	ADC_AVcc	VCC	ADC Power Pin
18	14	13	17	P5.4	I/O	Standard IO port
				NRST	I	Reset pin (low reset)

				MCLKO	O	Master clock divider output
19	15	14	18	Vcc	VCC	Power foot
20	16	15	19	P5.5	I/O	Standard IO port
21	17	16	20	Gnd	GND	earth (wire)

number				name (of a thing)	types	instructions
LQFP64	LQFP48	LQFP44	PDIP40			
22	18	17		P4.0	I/O	Standard IO port
				WR_3	O	Write signal line for external bus
				RxD2_2	I	Receive pin of serial port 2
23				P6.4	I/O	Standard IO port
				PWM4_3	O	Enhanced PWM channel 4 output pin
24				P6.5	I/O	Standard IO port
				PWM5_3	O	Enhanced PWM channel 5 output pin
25				P6.6	I/O	Standard IO port
				PWM6_3	O	Enhanced PWM channel 6 output pin
26				P6.7	I/O	Standard IO port
				PWM7_3	O	Enhanced PWM channel 7 output pin
27	19	18	21	P3.0	I/O	Standard IO port
				RxD	I	Receive pin of serial port 1
				CCP3_4	I/O	Capture input and pulse output of PCA
				INT4	I	External interrupts 4
28	20	19	22	P3.1	I/O	Standard IO port
				TxD	O	Transmit pin of serial port 1
				CCP2_4	I/O	Capture input and pulse output of PCA
29	21	20	23	P3.2	I/O	Standard IO port
				INT0	I	External interrupt 0
				CCP1_4	I/O	Capture input and pulse output of PCA
				SCLK_4	I/O	Clock pin of SPI
				SCL_4	I/O	Clock line for I2C
30	22	21	24	P3.3	I/O	Standard IO port
				INT1	I	External interrupt 1
				CCP0_4	I/O	Capture input and pulse output of PCA
				MISO_4	I/O	SPI host input slave output
				SDA_4	I/O	Data line for I2C interface
31	23	22	25	P3.4	I/O	Standard IO port
				T0	I	Timer 0 External clock input
				T1CLKO	O	Timer 1 Clock division output
				MOSI_4	I/O	SPI host output slave input
				CMPO	O	Comparator Output

number				name (of a thing)	types	instructions
LQFP64	LQFP48	LQFP44	PDIP40			
32	24	23	26	P3.5	I/O	Standard IO port
				T1	I	Timer 1 External clock input
				T0CLKO	O	Timer 0 Clock division output
				ECI_4	I	External pulse input for PCA
				SS_4	I	Slave select pin for SPI (host is output)
				PWMFLT	I	Enhanced external exception detection pin for PWM
33	25			P5.0	I/O	Standard IO port
				RxD3_2	I	Receive pin of serial port 3
				CMP+_2	I	Comparator positive input
34	26			P5.1	I/O	Standard IO port
				TxD3_2	O	Transmit pin of serial port 3
				CMP+_3	I	Comparator positive input
35	27	24	27	P3.6	I/O	Standard IO port
				INT2	I	External interrupts 2
				WR_2	O	Write signal line for external bus
				RxD_2	I	Receive pin of serial port 1
				CMP-	I	Comparator negative input
36	28	25	28	P3.7	I/O	Standard IO port
				INT3	I	External interrupts 3
				RD_2	O	Read signal line for external bus
				TxD_2	O	Transmit pin of serial port 1
				CMP+	I	Comparator positive input
37	29	26	29	P4.1	I/O	Standard IO port
				ALE	O	address latch signal
				CMPO_2	O	Comparator Output
38				P7.0	I/O	Standard IO port
				CCP0_3	I/O	Capture input and pulse output of PCA
39				P7.1	I/O	Standard IO port
				CCP1_3	I/O	Capture input and pulse output of PCA
40				P7.2	I/O	Standard IO port
				CCP2_3	I/O	Capture input and pulse output of PCA
41				P7.3	I/O	Standard IO port
				CCP3_3	I/O	Capture input and pulse output of PCA
42	30	27	30	P2.0	I/O	Standard IO port
				A8	I	address bus (computer)
				PWM0	O	Enhanced PWM Channel 0 Output Pin
43	31	28		P4.2	I/O	Standard IO port
				RD_3	O	Read signal line for external bus
				TxD2_2	O	Transmit pin of serial port 2

number				name (of a thing)	types	instructions
LQFP64	LQFP48	LQFP44	PDIP40			
44	32	29	31	P2.1	I/O	Standard IO port
				A9	I	address bus (computer)
				PWM1	O	Enhanced PWM channel 1 output pin
45	33	30	32	P2.2	I/O	Standard IO port
				A10	I	address bus (computer)
				PWM2	O	Enhanced PWM channel 2 output pin
				SS_2	I	Slave select pin for SPI (host is output)
				ECI_2	I	External pulse input for PCA
46	34	31	33	P2.3	I/O	Standard IO port
				A11	I	address bus (computer)
				PWM3	O	Enhanced PWM channel 3 output pin
				MOSI_2	I/O	SPI host output slave input
				CCP0_2	I/O	Capture input and pulse output of PCA
47	35	32	34	P2.4	I/O	Standard IO port
				A12	I	address bus (computer)
				PWM4	O	Enhanced PWM channel 4 output pin
				MISO_2	I/O	SPI host input slave output
				SDA_2	I/O	Data line for I2C interface
				CCP1_2	I/O	Capture input and pulse output of PCA
48	36	33	35	P2.5	I/O	Standard IO port
				A13	I	address bus (computer)
				PWM5	O	Enhanced PWM channel 5 output pin
				SCLK_2	I/O	Clock pin of SPI
				SCL_2	I/O	Clock line for I2C
				CCP2_2	I/O	Capture input and pulse output of PCA
49	37	34	36	P2.6	I/O	Standard IO port
				A14	I	address bus (computer)
				PWM6	O	Enhanced PWM channel 6 output pin
				CCP3_2	I/O	Capture input and pulse output of PCA
50	38	35	37	P2.7	I/O	Standard IO port
				A15	I	address bus (computer)
				PWM7	O	Enhanced PWM channel 7 output pin
51	39	36	38	P0.0	I/O	Standard IO port
				AD0	I	address bus (computer)
				ADC8	I	ADC Analog Input Channel 8
				RxD3	I	Receive pin of serial port 3

number				name (of a thing)	types	instructions
LQFP64	LQFP48	LQFP44	PDIP40			
52	40	37	39	P0.1	I/O	Standard IO port
				AD1	I	address bus (computer)
				ADC9	I	ADC Analog Input Channel 9
				TxD3	O	Transmit pin of serial port 3
53	41	38	40	P0.2	I/O	Standard IO port
				AD2	I	address bus (computer)
				ADC10	I	ADC Analog Input Channel 10
				RxD4	I	Receive pin of serial port 4
54				P7.4	I/O	Standard IO port
				SS_3	I	Slave select pin for SPI (host is output)
				ECI_3	I	External pulse input for PCA
55				P7.5	I/O	Standard IO port
				MOSI_3	I/O	SPI host output slave input
56				P7.6	I/O	Standard IO port
				MISO_3	I/O	SPI host input slave output
				SDA_3	I/O	Data line for I2C interface
57				P7.7	I/O	Standard IO port
				SCLK_3	I/O	Clock pin of SPI
				SCL_3	I/O	Clock line for I2C
58	42	39		P4.3	I/O	Standard IO port
				WR	O	Write signal line for external bus
				RxD_4	I	Receive pin of serial port 1
59	43	40	1	P0.3	I/O	Standard IO port
				AD3	I	address bus (computer)
				ADC11	I	ADC Analog Input Channel 11
				TxD4	O	Transmit pin of serial port 4
60	44	41	2	P0.4	I/O	Standard IO port
				AD4	I	address bus (computer)
				ADC12	I	ADC Analog Input Channel 12
				T3	I	Timer 3 External clock input

number				name (of a thing)	types	instructions
LQFP64	LQFP48	LQFP44	PDIP40			
61	45	42	3	P0.5	I/O	Standard IO port
				AD5	I	address bus (computer)
				ADC13	I	ADC Analog Input Channel 13
				T3CLKO	O	Timer 3 clock division output
62	46	43	4	P0.6	I/O	Standard IO port
				AD6	I	address bus (computer)
				ADC14	I	ADC Analog Input Channel 14
				T4	I	Timer 4 External clock input
63	47	44	5	P0.7	I/O	Standard IO port
				AD7	I	address bus (computer)
				T4CLKO	O	Timer 4 clock division output
64	48	1	6	P1.0	I/O	Standard IO port
				ADC0	I	ADC Analog Input Channel 0
				PWM0_2	O	Enhanced PWM Channel 0 Output Pin
				RxD2	I	Receive pin of serial port 2

3 Function foot cut change

The STC8A8K64D4 series microcontroller's special peripheral serial port, SPI, PCA, I²C, and bus control pins can be switched directly across multiple I/Os to enable time-sharing multiplexing of one peripheral as multiple devices.

3.1 Function pin switching related registers

symbolic	description	address	Bit Addresses and Symbols								reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
BUS_SPEED	Bus Speed Control Register	A1H	RW_S[1:0]						SPEED[1:0]		00xx,xx00
P_SW1	Peripheral port switching register 1	A2H	S1_S[1:0]		CCP_S[1:0]		SPI_S[1:0]		0	-	nn00,000x
P_SW2	Peripheral port switching register 2	BAH	EAXFR		I2C_S[1:0]		CMPO_S	S4_S	S3_S	S2_S	0x00,0000

symbolic	description	address	Bit Addresses and Symbols								reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
PWM0CR	PWM0 control register	FF14H	ENC0O	C0INI	-	C0_S[1:0]		EC0I	EC0T2SI	EC0T1SI	00x0,0000
PWM1CR	PWM1 control register	FF1CH	ENC1O	C1INI	-	C1_S[1:0]		EC1I	EC1T2SI	EC1T1SI	00x0,0000
PWM2CR	PWM2 control register	FF24H	ENC2O	C2INI	-	C2_S[1:0]		EC2I	EC2T2SI	EC2T1SI	00x0,0000
PWM3CR	PWM3 Control Register	FF2CH	ENC3O	C3INI	-	C3_S[1:0]		EC3I	EC3T2SI	EC3T1SI	00x0,0000
PWM4CR	PWM4 Control Register	FF34H	ENC4O	C4INI	-	C4_S[1:0]		EC4I	EC4T2SI	EC4T1SI	00x0,0000
PWM5CR	PWM5 control register	FF3CH	ENC5O	C5INI	-	C5_S[1:0]		EC5I	EC5T2SI	EC5T1SI	00x0,0000
PWM6CR	PWM6 Control Register	FF44H	ENC6O	C6INI	-	C6_S[1:0]		EC6I	EC6T2SI	EC6T1SI	00x0,0000
PWM7CR	PWM7 Control Register	FF4CH	ENC7O	C7INI	-	C7_S[1:0]		EC7I	EC7T2SI	EC7T1SI	00x0,0000
MCLKOCR	Master Clock Output Control Register	FE05H	MCLKO_S	MCLKODIV [6:0]							0000,0000
LCMIFCFG	LCM Interface Configuration Register	FE50H	LCMIFIE	-	LCMIFIP[1:0]		LCMIFPS[1:0]		D16_D8	M68_80	0x00,0000
LCMIFCFG2	LCM Interface Configuration Register 2	FE51H		LCMIFEXPS[1:0]		SETUPT[2:0]			HOLDT[1:0]		x000,0000

3.1.1 Bus Speed Control Register (BUS_SPEED)

symbolic	address	B7	B6	B5	B4	B3	B2	B1	B0
BUS_SPEED	A1H	RW_S[1:0]						SPEED[1:0]	

RW_S[1:0]: external bus RD/WR control line select bits

RW_S[1:0]	RD	WR
00	P4.4	P4.3
01	P3.7	P3.6
10	P4.2	P4.0
11	reta in	

3.1.2 Peripheral port switching control register 1 (P_SW1) serial port 1, CCP, SPI switch over

symbolic	address	B7	B6	B5	B4	B3	B2	B1	B0
P_SW1	A2H	S1_S[1:0]		CCP_S[1:0]		SPI_S[1:0]		0	-

S1_S[1:0]: Serial port 1 function pin select 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

CCP_S[1:0]: PCA function pin select bits

CCP_S[1:0]	ECI	CCP0	CCP1	CCP2	CCP3
00	P1.2	P1.7	P1.6	P1.5	P1.4
01	P2.2	P2.3	P2.4	P2.5	P2.6
10	P7.4	P7.0	P7.1	P7.2	P7.3
11	P3.5	P3.3	P3.2	P3.1	P3.0

SPI_S[1:0]: SPI function pin select bits

SPI_S[1:0]	SS	MOSI	MISO	SCLK
00	P1.2	P1.3	P1.4	P1.5
01	P2.2	P2.3	P2.4	P2.5
10	P7.4	P7.5	P7.6	P7.7
11	P3.5	P3.4	P3.3	P3.2

3.1.3 Peripheral port switching control register 2 (P_SW2) serial port 2/3/4, I2C, comparator output switching

symbolic	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: Enables access to XFR.

When access to the XFR is required, the EAXFR must be set to 1 before the XFR can be read or written to properly

I2C_S[1:0]: I² C function pin select 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 select bit

CMPO_S	CMPO
0	P3.4
1	P4.1

S4_S: Serial port 4 function pin selection bit

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

S3_S: Serial port 3 function pin selection bit

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

S2_S: Serial port 2 function pin select bit

S2_S	RxD2	TxD2
0	P1.0	P1.1
1	P4.0	P4.2

3.1.4 Clock Select Register (MCLKOCR)

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

MCLKO_S: Master clock output pin select bit

MCLKO_S	MCLKO
0	P5.4
1	P1.6

3.1.5 Enhanced PWM Control Register (PWMnCR)

symbolic	address	B7	B6	B5	B4	B3	B2	B1	B0
PWM0CR	FF14H	ENC0O	C0INI	-	C0_S[1:0]		EC0I	EC0T2SI	EC0T1SI
PWM1CR	FF1CH	ENC1O	C1INI	-	C1_S[1:0]		EC1I	EC1T2SI	EC1T1SI
PWM2CR	FF24H	ENC2O	C2INI	-	C2_S[1:0]		EC2I	EC2T2SI	EC2T1SI
PWM3CR	FF2CH	ENC3O	C3INI	-	C3_S[1:0]		EC3I	EC3T2SI	EC3T1SI

PWM4CR	FF34H	ENC4O	C4IFI	-	C4_S[1:0]	EC4I	EC4T2SI	EC4T1SI

PWM5CR	FF3CH	ENC5O	C5INI	-	C5_S[1:0]	EC5I	EC5T2SI	EC5T1SI
PWM6CR	FF44H	ENC6O	C6INI	-	C6_S[1:0]	EC6I	EC6T2SI	EC6T1SI
PWM7CR	FF4CH	ENC7O	C7INI	-	C7_S[1:0]	EC7I	EC7T2SI	EC7T1SI

C0_S[1:0]: Enhanced PWM channel 0 output pin select bits

C0_S[1:0]	PWM0
00	P2.0
01	P1.0
10	P6.0
11	retain

C1_S[1:0]: Enhanced PWM channel 1 output pin select bits

C1_S[1:0]	PWM1
00	P2.1
01	P1.1
10	P6.1
11	retain

C2_S[1:0]: Enhanced PWM channel 2 output pin select bits

C2_S[1:0]	PWM2
00	P2.2
01	P1.2
10	P6.2
11	retain

C3_S[1:0]: Enhanced PWM channel 3 output pin select bits

C3_S[1:0]	PWM3
00	P2.3
01	P1.3
10	P6.3
11	retain

C4_S[1:0]: Enhanced PWM channel 4 output pin select bits

C4_S[1:0]	PWM4
00	P2.4
01	P1.4
10	P6.4
11	retain

C5_S[1:0]: Enhanced PWM channel 5 output pin select bits

C5_S[1:0]	PWM5
00	P2.5
01	P1.5
10	P6.5
11	retain

C6_S[1:0]: Enhanced PWM channel 6 output pin select bits

C6_S[1:0]	PWM6
00	P2.6
01	P1.6
10	P6.6

11	retain

C7_S[1:0]: Enhanced PWM channel 7 output pin select bits

C7_S[1:0]	PWM7
00	P2.7
01	P1.7
10	P6.7
11	retain

3.1.6 LCM Interface Configuration Register (LCMIFCFG)

symbolic	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 select bits

LCMIFCPS [1:0]	RS	Read signal RD of I8080 Enable signal E of the M6800	Write signal WR of I8080 M6800 read/write signal RW
00	P4.1	P4.4	P4.3
01	P4.1	P3.7	P3.6
10	P4.1	P4.2	P4.0
11	P4.0	P3.7	P3.6

LCMIFDPS[1:0]: LCM interface data pin select bits

LCMIFDPS [1:0]	D16_D8	Data High Byte DB[15:8]	Data low byte DB[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]	P7 [7:0]
11	1	P6[7:0]	P7 [7:0]

3.2 paradigm procedure

3.2.1 Serial port 1 Switching

C code

```
// Tested operating frequency of 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 tested at 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 Serial port 2 Switching

C code

```
//test operating frequency of

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/P4.0, TXD2_2/P4.2

    while (1);
}
```

assembly code

Operating frequency tested at *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 Serial port 3 Switching

C code

```
//test operating frequency of 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 tested at 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 Serial port 4 Switching

C code

```
//test operating frequency of

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 tested at *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 toggle

C code

```
//test operating frequency of

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/P7.4 MOSI_3/P7.5, MISO_3/P7.6, SCLK_3/P7.7
```

```
    //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 tested at *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/P7.4 MOSI_3/P7.5, MISO_3/P7.6, SCLK_3/P7.7
;           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 PWM toggle

assembly code

```
P_SW2       DATA    0BAH
PWM0CR      EQU     0FF14H
```

```
PWM1CR      EQU     0FF1CH
PWM2CR      EQU     0FF24H
PWM3CR      EQU     0FF2CH
PWM4CR      EQU     0FF34H
PWM5CR      EQU     0FF3CH
PWM6CR      EQU     0FF44H
PWM7CR      EQU     0FF4CH

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,#3FH
            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          ;PWM0/P2.0
;           MOV     A,#08H          ;PWM0_2/P1.0
;           MOV     A,#10H          ;pwm0_3/p6.0
            MOV     DPTR,#PWM0CR
            MOVX    @DPTR,A
            MOV     A,#00H          ;PWM1/P2.1
;           MOV     A,#08H          ;pwm1_2/p1.1
;           MOV     A,#10H          ;pwm1_3/p6.1
            MOV     DPTR,#PWM1CR
            MOVX    @DPTR,A
            MOV     A,#00H          ;PWM2/P2.2
;           MOV     A,#08H          ;pwm2_2/p1.2
;           MOV     A,#10H          ;pwm2_3/p6.2
            MOV     DPTR,#PWM2CR
            MOVX    @DPTR,A
            MOV     A,#00H          ;PWM3/P2.3
;           MOV     A,#08H          ;pwm3_2/p1.3
;           MOV     A,#10H          ;pwm3_3/p6.3
```

```
        MOV     DPTR,#PWM3CR
        MOVX    @DPTR,A
        MOV     A,#00H          ;PWM4/P2.4
;       MOV     A,#08H          ;pwm4_2/p1.4
;       MOV     A,#10H          ;pwm4_3/p6.4
        MOV     DPTR,#PWM4CR
        MOVX    @DPTR,A
        MOV     A,#00H          ;PWM5/P2.5
;       MOV     A,#08H          ;pwm5_2/p1.5
;       MOV     A,#10H          ;pwm5_3/p6.5
        MOV     DPTR,#PWM5CR
        MOVX    @DPTR,A
        MOV     A,#00H          ;PWM6/P2.6
;       MOV     A,#08H          ;pwm6_2/p1.6
;       MOV     A,#10H          ;pwm6_3/p6.6
        MOV     DPTR,#PWM6CR
        MOVX    @DPTR,A
        MOV     A,#00H          ;PWM7/P2.7
;       MOV     A,#08H          ;pwm7_2/p1.7
;       MOV     A,#10H          ;pwm7_3/p6.7
        MOV     DPTR,#PWM7CR
        MOVX    @DPTR,A
        MOV     P_SW2,#00H

        SJMP    $

        END
```

C code

```
#include "reg51.h"

#define PWM0CR (*(unsigned char volatile xdata *)0xff14)
#define PWM1CR (*(unsigned char volatile xdata *)0xff1c)
#define PWM2CR (*(unsigned char volatile xdata *)0xff24)
#define PWM3CR (*(unsigned char volatile xdata *)0xff2c)
#define PWM4CR (*(unsigned char volatile xdata *)0xff34)
#define PWM5CR (*(unsigned char volatile xdata *)0xff3c)
#define PWM6CR (*(unsigned char volatile xdata *)0xff44)
#define PWM7CR (*(unsigned char volatile xdata *)0xff4c)

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;
    PWM0CR = 0x00;          //PWM0/P2.0
    //PWM0CR = 0x08;        //PWM0_2/P1.0
    //PWM0CR = 0x10;        //PWM0_3/P6.0
    PWM1CR = 0x00;          //PWM1/P2.1
    //PWM1CR = 0x08;        //PWM1_2/P1.1
    //PWM1CR = 0x10;        //PWM1_3/P6.1
    PWM2CR = 0x00;          //PWM2/P2.2
    //PWM2CR = 0x08;        //PWM2_2/P1.2
    //PWM2CR = 0x10;        //PWM2_3/P6.2
    PWM3CR = 0x00;          //PWM3/P2.3
    //PWM3CR = 0x08;        //PWM3_2/P1.3
    //PWM3CR = 0x10;        //PWM3_3/P6.3
    PWM4CR = 0x00;          //PWM4/P2.4
    //PWM4CR = 0x08;        //PWM4_2/P1.4
    //PWM4CR = 0x10;        //PWM4_3/P6.4
    PWM5CR = 0x00;          //PWM5/P2.5
    //PWM5CR = 0x08;        //PWM5_2/P1.5
    //PWM5CR = 0x10;        //PWM5_3/P6.5
    PWM6CR = 0x00;          //PWM6/P2.6
    //PWM6CR = 0x08;        //PWM6_2/P1.6
    //PWM6CR = 0x10;        //PWM6_3/P6.6
    PWM7CR = 0x00;          //PWM7/P2.7
    //PWM7CR = 0x08;        //PWM7_2/P1.7
    //PWM7CR = 0x10;        //PWM7_3/P6.7
    P_SW2 = 0x00;

    while (1);
}
```

3.2.7 PCA/CCP/PWM Switching

assembly code

```
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,#3FH
        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          ;ECI/P1.2, CCP0/P1.7, CCP1/P1.6, CCP2/P1.5,CCP3/P1.4
;       MOV     P_SW1,#10H          ;eci_2/p2.2, ccp0_2/p2.3, ccp1_2/p2.4, ccp2_2/p2.5,ccp3_2/p2.6
;       MOV     P_SW1,#20H          ;eci_3/p7.4, ccp0_3/p7.0, ccp1_3/p7.1, ccp2_3/p7.2,ccp3_3/p7.3
;       MOV     P_SW1,#30H          ;eci_4/p3.5, ccp0_4/p3.3, ccp1_4/p3.2, ccp2_4/p3.1,ccp3_4/p3.0

        SJMP    $

        END
```

C code

```
#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;           //ECI/P1.2, CCP0/P1.7, CCP1/P1.6, CCP2/P1.5,CCP3/P1.4
//      P_SW1 = 0x10;           //ECI_2/P2.2, CCP0_2/P2.3, CCP1_2/P2.4, CCP2_2/P2.5,CCP3_2/P2.6
//      P_SW1 = 0x20;           //ECI_3/P7.4, CCP0_3/P7.0, CCP1_3/P7.1, CCP2_3/P7.2,CCP3_3/P7.3
//      P_SW1 = 0x30;           //eci_4/p3.5, ccp0_4/p3.3, ccp1_4/p3.2, ccp2_4/p3.1,ccp3_4/p3.0

        while (1);
}
```

3.2.8 I2C Switching

C code

```
//test operating frequency of

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 tested at *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
```

3.2.9 Comparator output switching

C code

// Tested operating frequency of *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 tested at *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
```

3.2.10 Master clock output switching

C code

```
//test operating frequency of

11.0592MHz #include "reg51.h"

#define   MCLKOCR   (*(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;
    MCLKOCR = 0x04;                 //HIRC/4 output via MCLKO/P5.4
//  MCLKOCR = 0x84;                 //HIRC/4 output via MCLKO_2/P1.6
    P_SW2 = 0x00;

    while (1);
}
```

assembly code

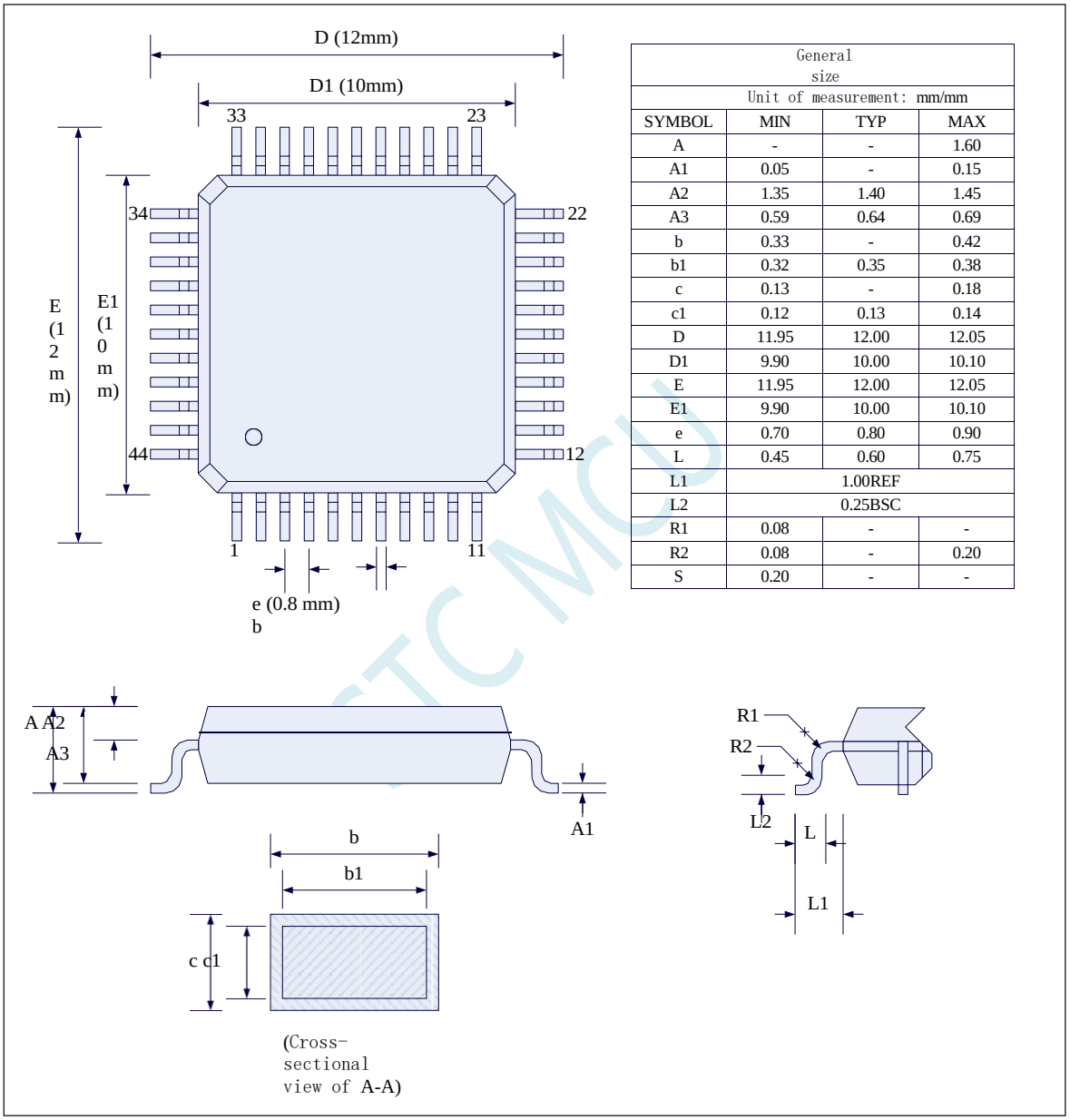
Operating frequency tested at *11.0592MHz*

```
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,#04H          ;HIRC/4 output via MCLKO/P5.4
;           MOV     A,#84H          ;HIRC/4 output via MCLKO_2/P1.6
            MOV     DPTR,#MCLKOCR
            MOVX    @DPTR,A
```

```
SJMP    $

END
```

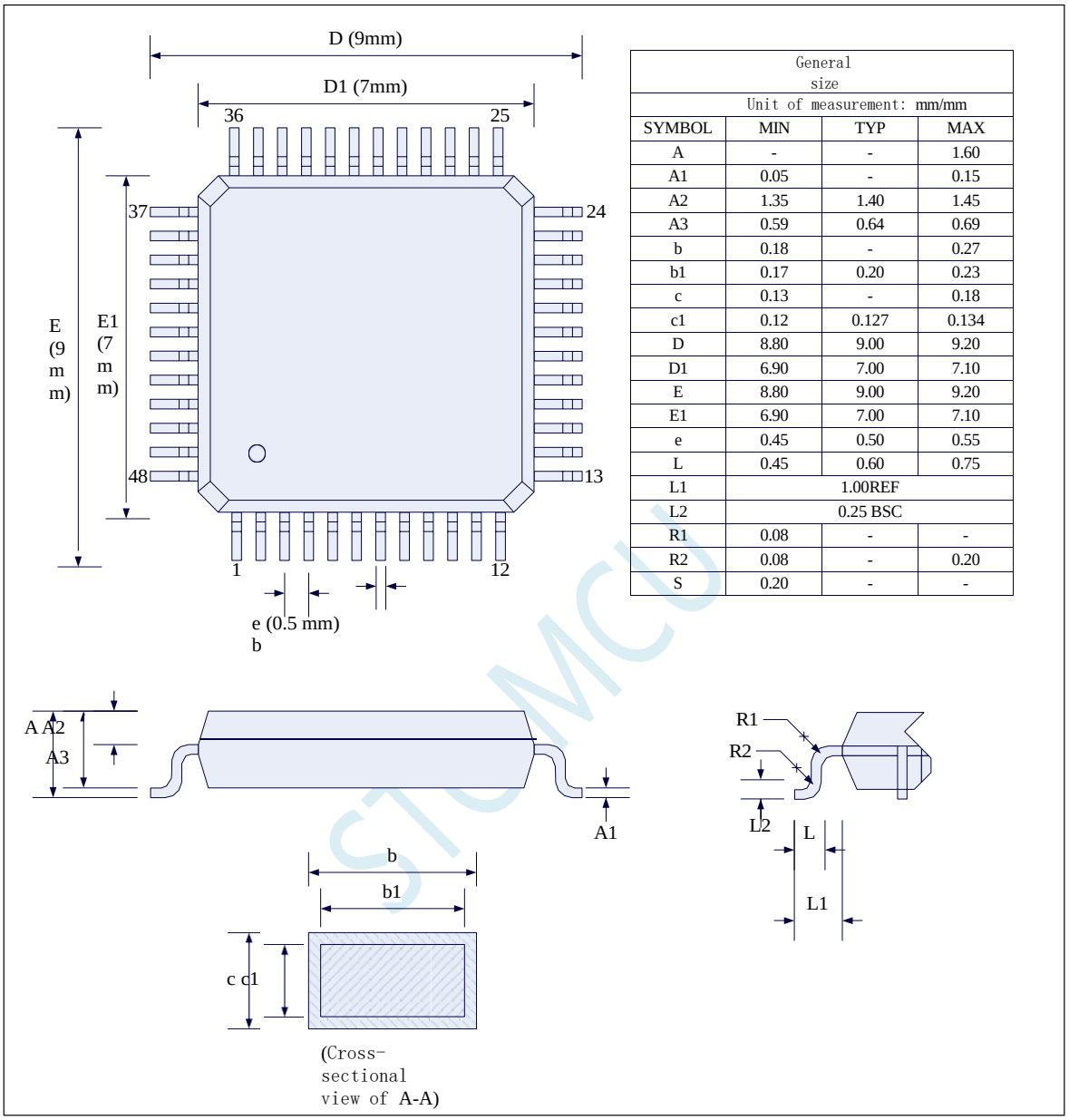
4 Package size Figure

4.1 LQFP44 package size diagram (12mm*12mm)



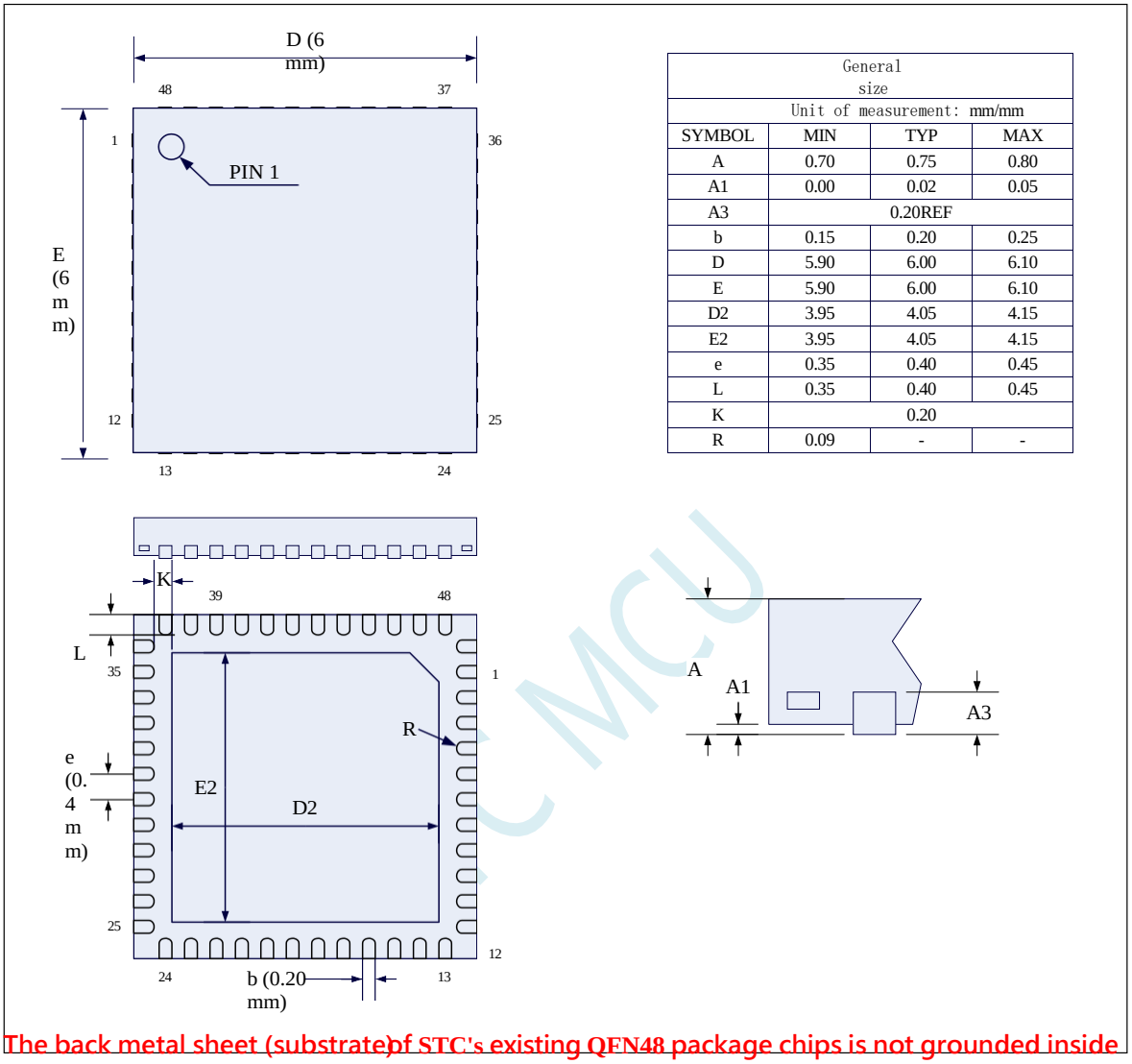
General size			
Unit of measurement: mm/mm			
SYMBOL	MIN	TYP	MAX
A	-	-	1.60
A1	0.05	-	0.15
A2	1.35	1.40	1.45
A3	0.59	0.64	0.69
b	0.33	-	0.42
b1	0.32	0.35	0.38
c	0.13	-	0.18
c1	0.12	0.13	0.14
D	11.95	12.00	12.05
D1	9.90	10.00	10.10
E	11.95	12.00	12.05
E1	9.90	10.00	10.10
e	0.70	0.80	0.90
L	0.45	0.60	0.75
L1	1.00REF		
L2	0.25BSC		
R1	0.08	-	-
R2	0.08	-	0.20
S	0.20	-	-

4.2 LQFP48 package size diagram (9mm*9mm)



General size			
Unit of measurement: mm/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	0.25 BSC		
R1	0.08	-	-
R2	0.08	-	0.20
S	0.20	-	-

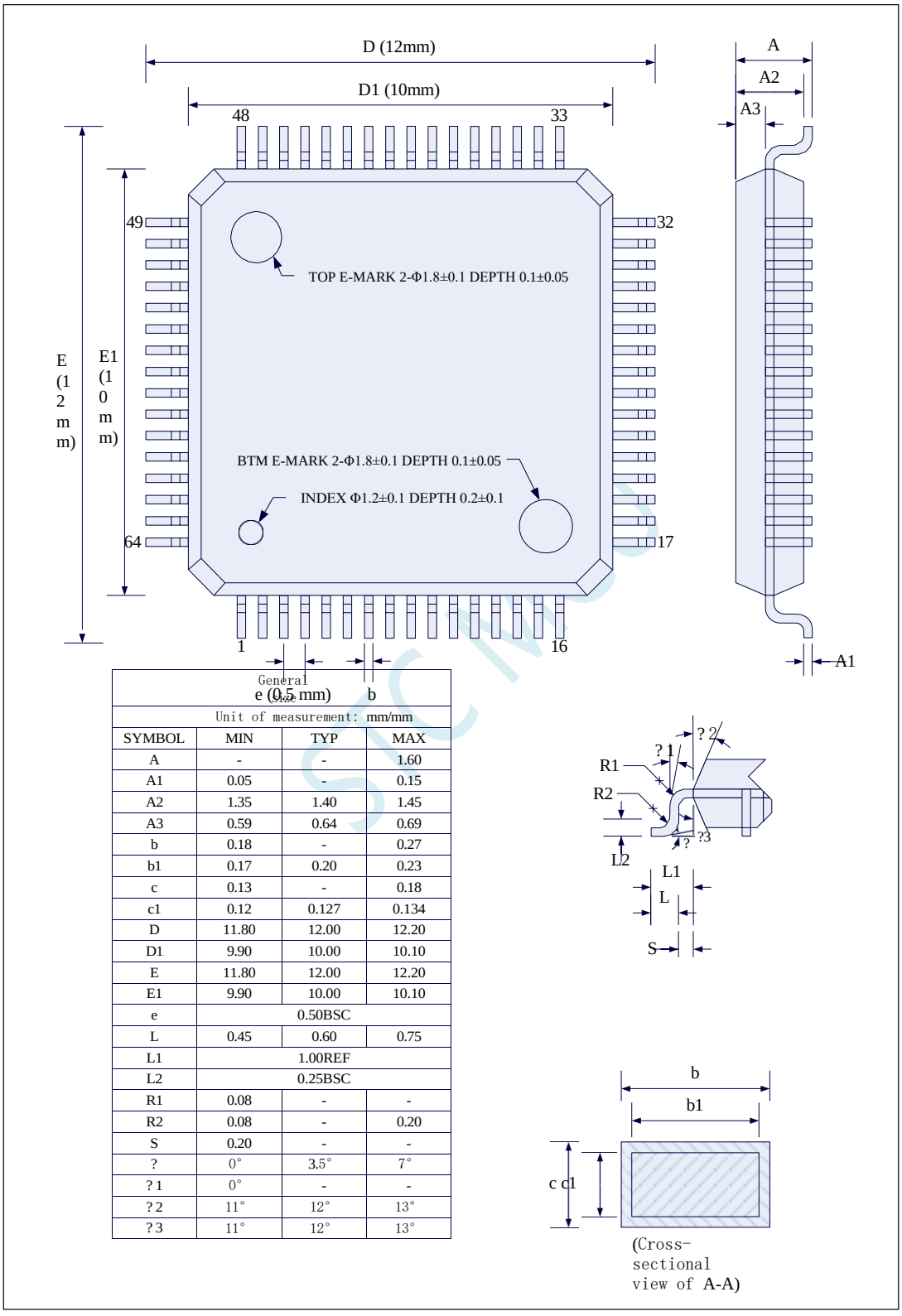
4.3 QFN48 package size diagram (6mm*6mm)



General size			
Unit of measurement: mm/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	-	-

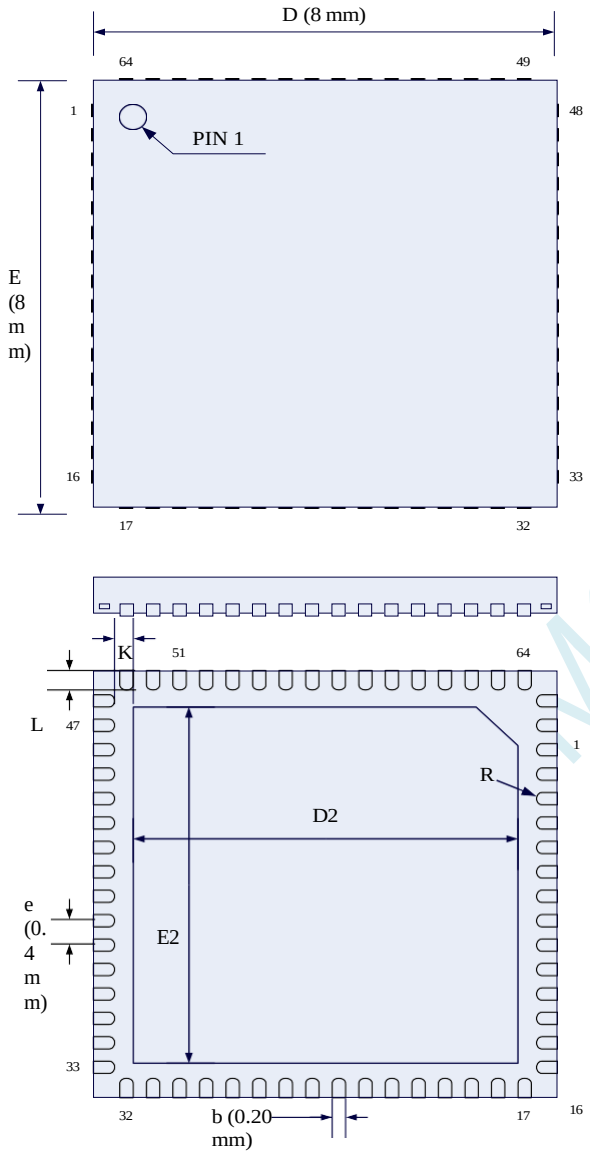
The back metal sheet (substrate) of STC's existing QFN48 package chips is not grounded inside the chip and may or may not be grounded on the user's PCB board without affecting chip performance

4.4 LQFP64 package size diagram (12mm*12mm)



General Size	e (0.5 mm)		b
Unit of measurement: mm/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°

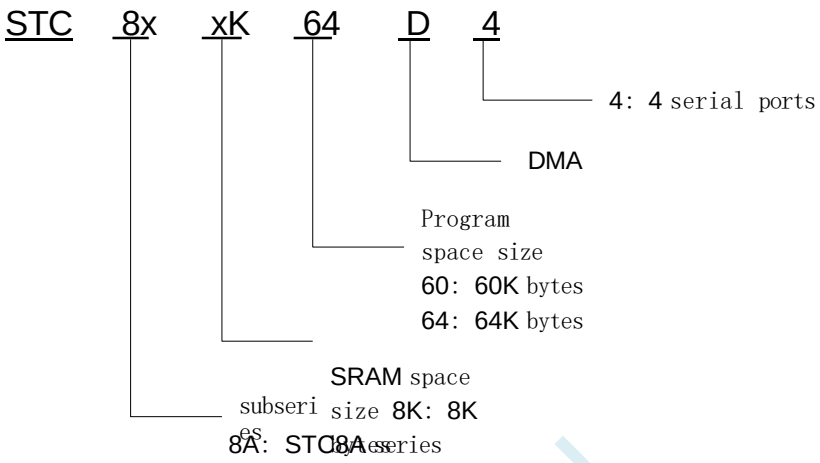
4.5 QFN64 package size diagram (8mm*8mm)



General size			
Unit of measurement: mm/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 QFN64 package chips is not grounded inside the chip and may or may not be grounded on the user's PCB board without affecting chip performance

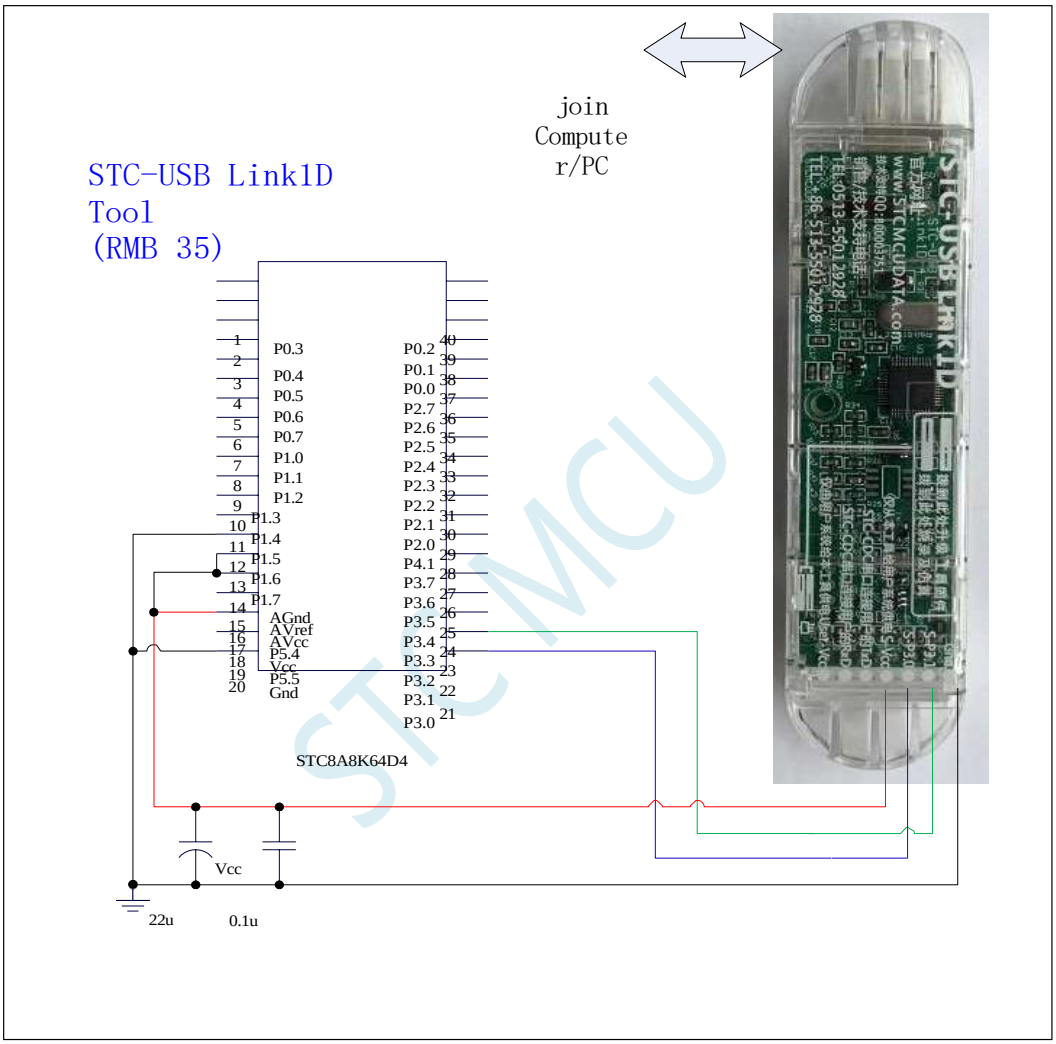
4.6 STC8A8K64D4 Series Microcontroller Naming Rules



5 ISP download and typical application lines diagram

5.1 STC8A8K64D4 Series ISP Download Application Wiring Diagram

5.1.1 Download using the STC-USB Link1D tool, which supports online and offline downloads



ISP Download steps.

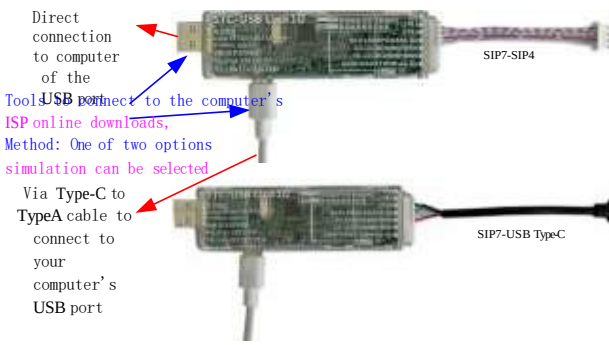
1. Connect the STC-USB Link1D to the target chip as shown in the figure
2. Click the "Download/Program" button in the STC-ISP download software
3. Start ISP Download

Note: If the STC-USB Link1D is used to power the target system, the total current of the

target system must not be greater than 200mA, otherwise the download will fail.

Note: It has been found that when using the USB cable for ISP downloading, the voltage drop on the USB cable is too thin, resulting in

The power supply is insufficient for ISP downloading, so be sure to use a USB-enhanced cable when using a USB cable for ISP downloading.



Direct connection to computer of the USB port

Tools to connect to the computer's ISP online downloads, Method: One of two options simulation can be selected

Connecting the Gnd of the target chip
Connect the target chip's P3.1 Connect the target chip's P3.0 tool to power the target system

This connection gives the target system for the tool power supply, the tool will automatically power down and power up the target system again when the ISP downloads.

Generally used for tool ISP offline download, online

Via Type-C to TypeA cable to connect to your computer's USB port

Connecting the Gnd of the target chip
Connect the target chip's P3.1 to the target chip's P3.0 tool to power the target system chip

These two types of connections

This type of connection is different from the previous type of connection

The function of the method is exactly the same, except that the target chip can be connected directly to the reserved Type-C connector

Connecting the Gnd of the target chip
Connect P3.1 of the target chip
Connect P3.0 of the target chip
Separate power supply for tools and target systems

This connection provides the tool and the target system

For each to be powered independently, ISP downloads require manual de-energization and reenergization of the target system.

Connecting the Gnd of the target chip
Connect P3.1 of the target chip Connect P3.0 of the target chip Target system to power the tool

This connection type is required when the power consumption of the target system is greater than 500mA

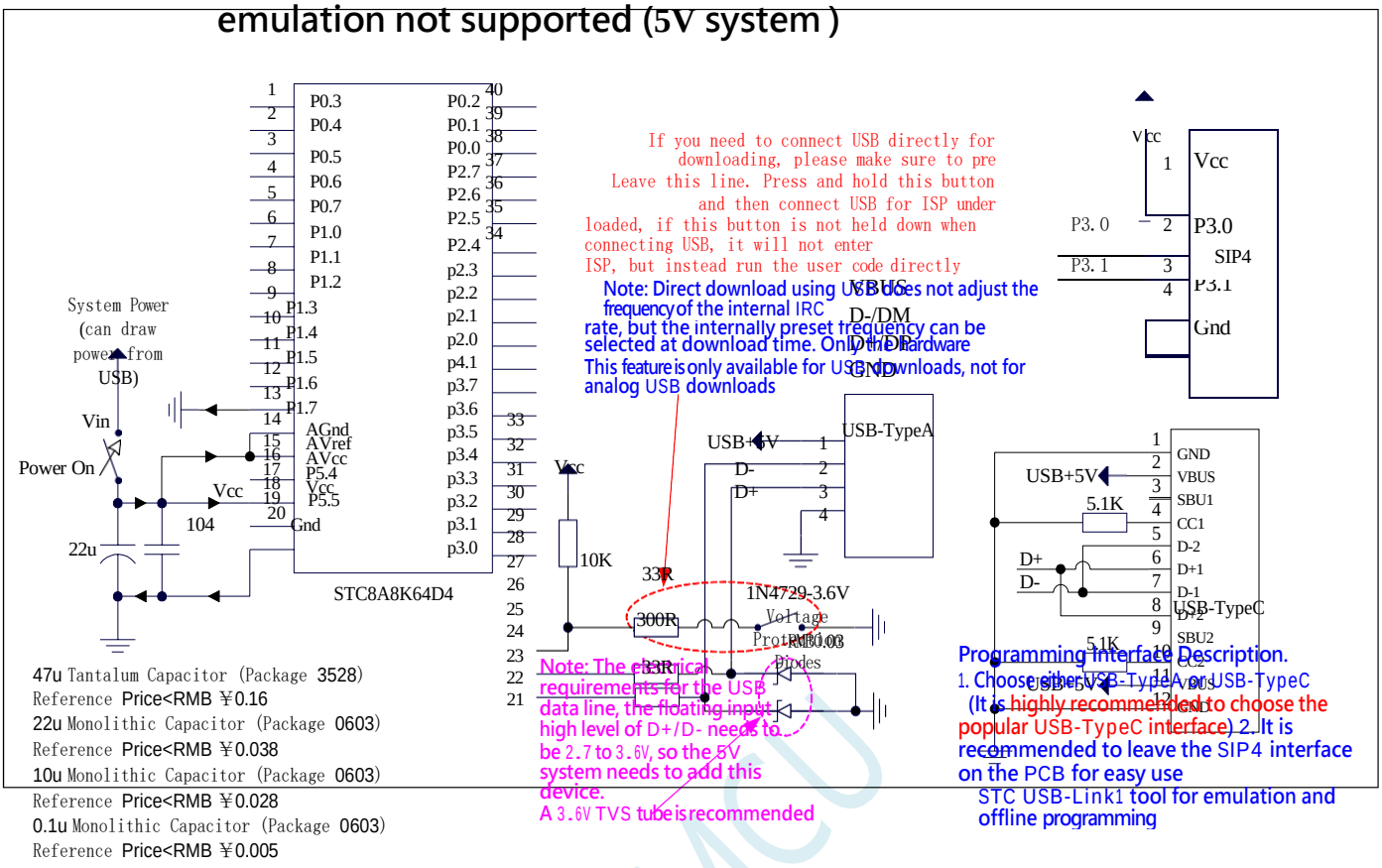
This connection gives the tool for the target system Electricity supply.

This connection is usually chosen when you need to use the tool for ISP offline download (the tool is not connected to the computer)

The STC provides this connection cable to facilitate the use of a cable that is only reserved for Direct hardware USB download or software emulated USB download from the target chip on the user board of the SIP4 interface

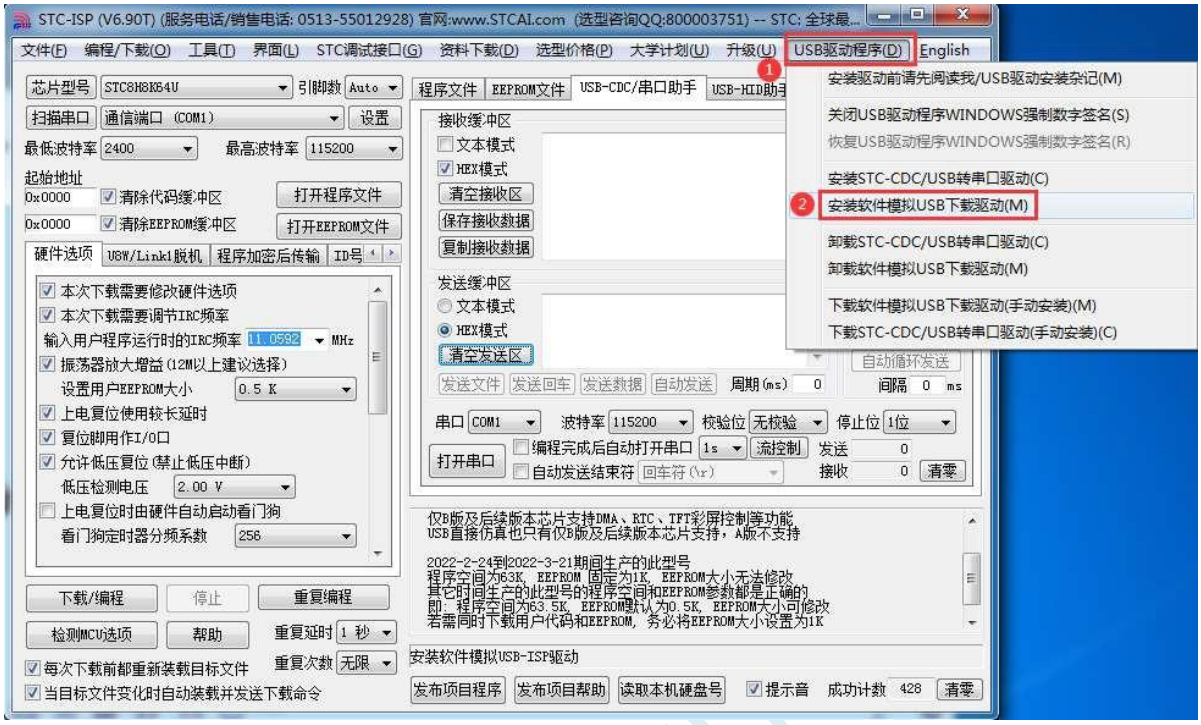
This cable with a male USB Type-A connector on both ends is standard The STC will also provide this connection cable for direct hardware USB download or software emulated USB download for target chips with USB Type-A female chassis reserved on the user's system board

5.1.2 Software emulation USB direct ISP download, recommended to try, emulation not supported (5V system)



47u Tantalum Capacitor (Package 3528)
Reference Price<RMB ￥0.16
22u Monolithic Capacitor (Package 0603)
Reference Price<RMB ￥0.038
10u Monolithic Capacitor (Package 0603)
Reference Price<RMB ￥0.028
0.1u Monolithic Capacitor (Package 0603)
Reference Price<RMB ￥0.005

Nowadays, STC8G/STC8H MCUs without hardware USB basically support software emulation of USB downloading user programs, because they use the USB-SCAN communication protocol, regardless of any version of the operating system, you have to install the driver. Install the driver in the place shown below in the STC-ISP download software.

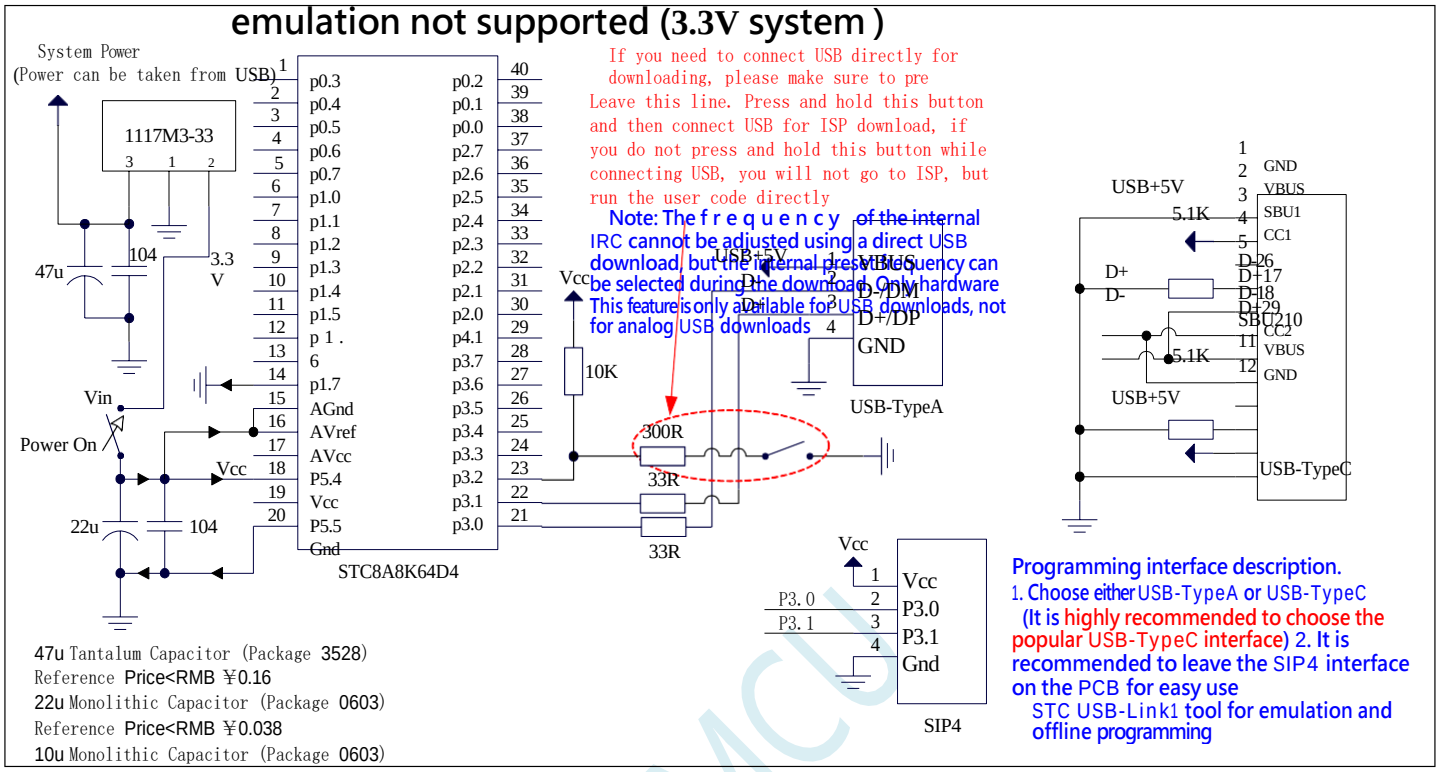


ISP Download steps.

1. D-/P3.0, D+/P3.1 connected to PC-USB port
2. the P3.2 and GND short, the experiment box board P3.2/INT0 button press
3. Re-energize the target chip. If the target chip has been powered down, it can be powered up directly; if the target chip is powered up, it needs to be powered down and powered up again (cold start) Wait for the "STC USB Writer (HID1)" to be recognized automatically in the STC-ISP download software, then it is not related to the P3.2 status (at this time, you don't need to keep pressing the P3.2 port, it doesn't matter if you are tired of pressing the handle all the time, it will be a problem if you press the button)
4. click the "Download/Program" button in the download software (Note: the order of operation of USB download and serial download is different, **do not**

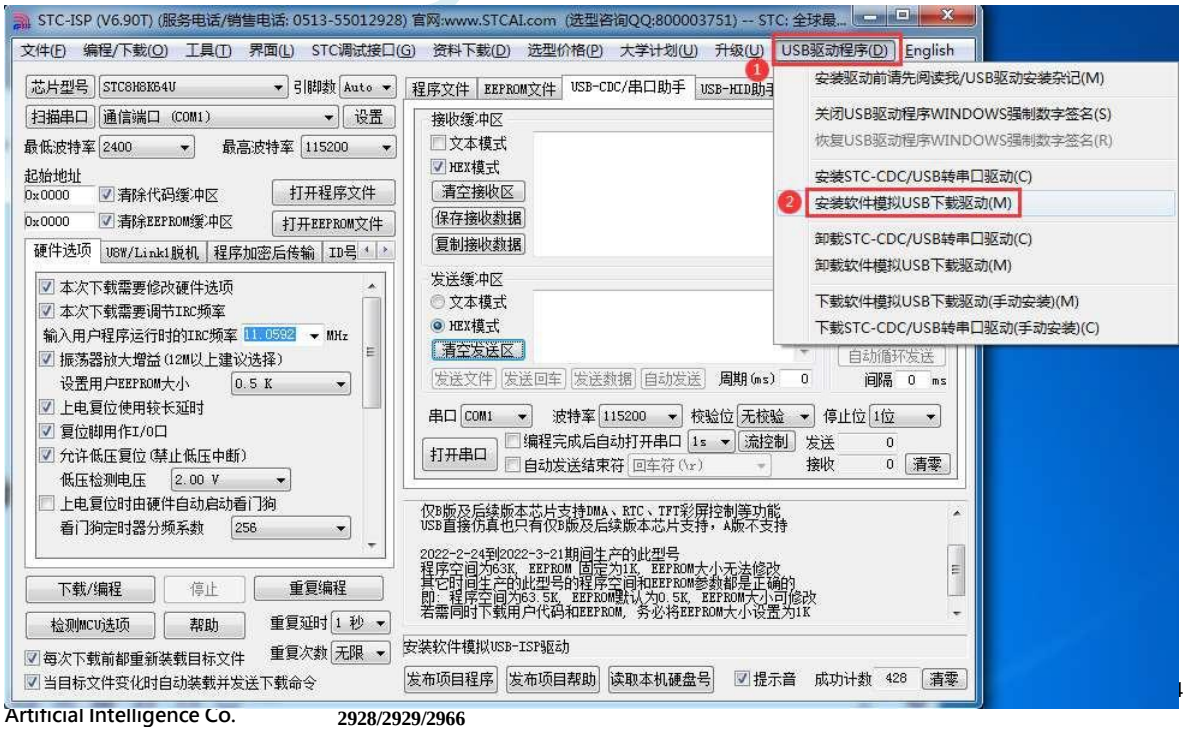
   (Click the Download button first, and wait until the computer recognizes the "STC USB Writer (HID1)" device before clicking the Download button to start downloading)

5.1.3 Software emulation USB direct ISP download, recommended to try, emulation not supported (3.3V system)



47u Tantalum Capacitor (Package 3528)
Reference Price<RMB ￥0.16
22u Monolithic Capacitor (Package 0603)
Reference Price<RMB ￥0.038
10u Monolithic Capacitor (Package 0603)
Reference Price<RMB ￥0.028
0.1u Monolithic Capacitor (Package 0603)
Reference Price<RMB ￥0.005

Nowadays, STC8G/STC8H MCUs without hardware USB basically support software emulation of USB downloading user programs, because they use the USB-SCAN communication protocol, regardless of any version of the operating system, you have to install the driver. Install the driver in the place shown below in the STC-ISP download software.

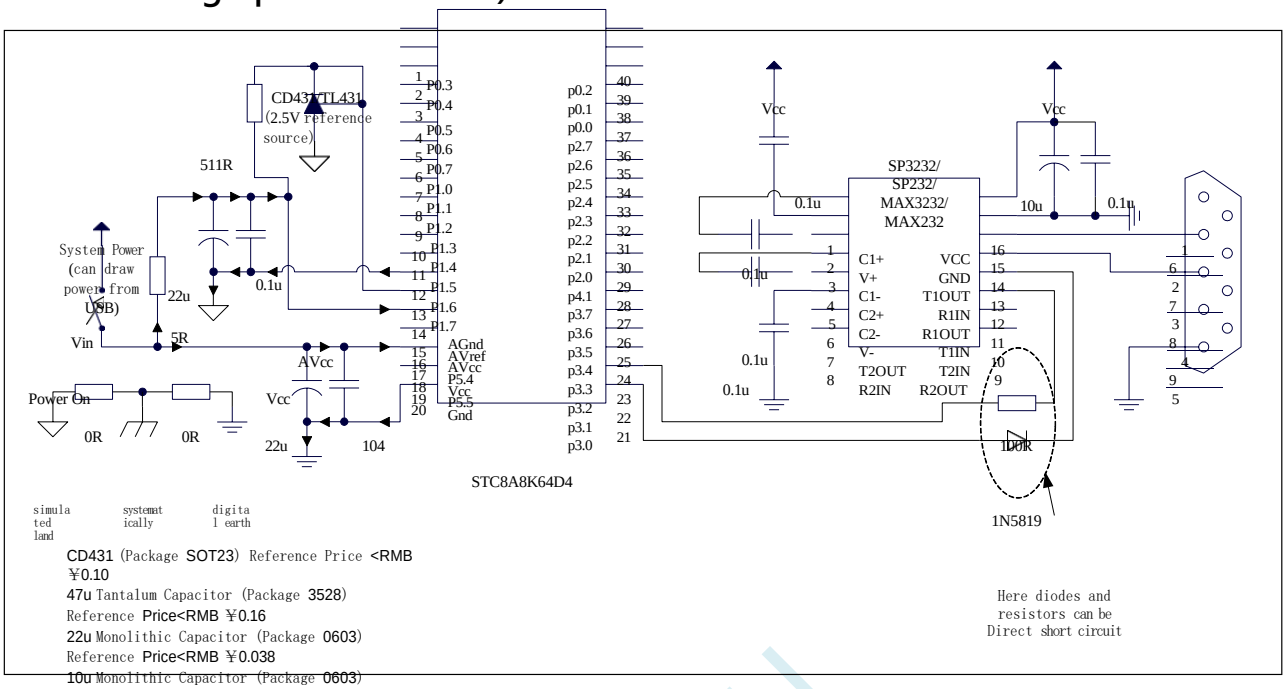


ISP Download steps.

1. D-/P3.0, D+/P3.1 connected to PC-USB port
2. the P3.2 and GND short, the experiment box board P3.2/INT0 button press
3. Re-energize the target chip. If the target chip has been powered down, it can be powered up directly; if the target chip is powered up, it needs to be powered down and powered up again (cold start) Wait for the "STC USB Writer (HID1)" to be recognized automatically in the STC-ISP download software, then it is not related to the P3.2 status (at this time, you don't need to keep pressing the P3.2 port, it doesn't matter if you are tired of pressing the handle all the time, it will be a problem if you press the button)
4. click the "Download/Program" button in the download software (Note: the order of operation of USB download and serial download is different, do not

(Click the Download button first, and wait until the computer recognizes the "STC USB Writer (HID1)" device before clicking the Download button to start downloading)

5.1.4 Download using RS-232 converter, also supports emulation (using high precision ADC)



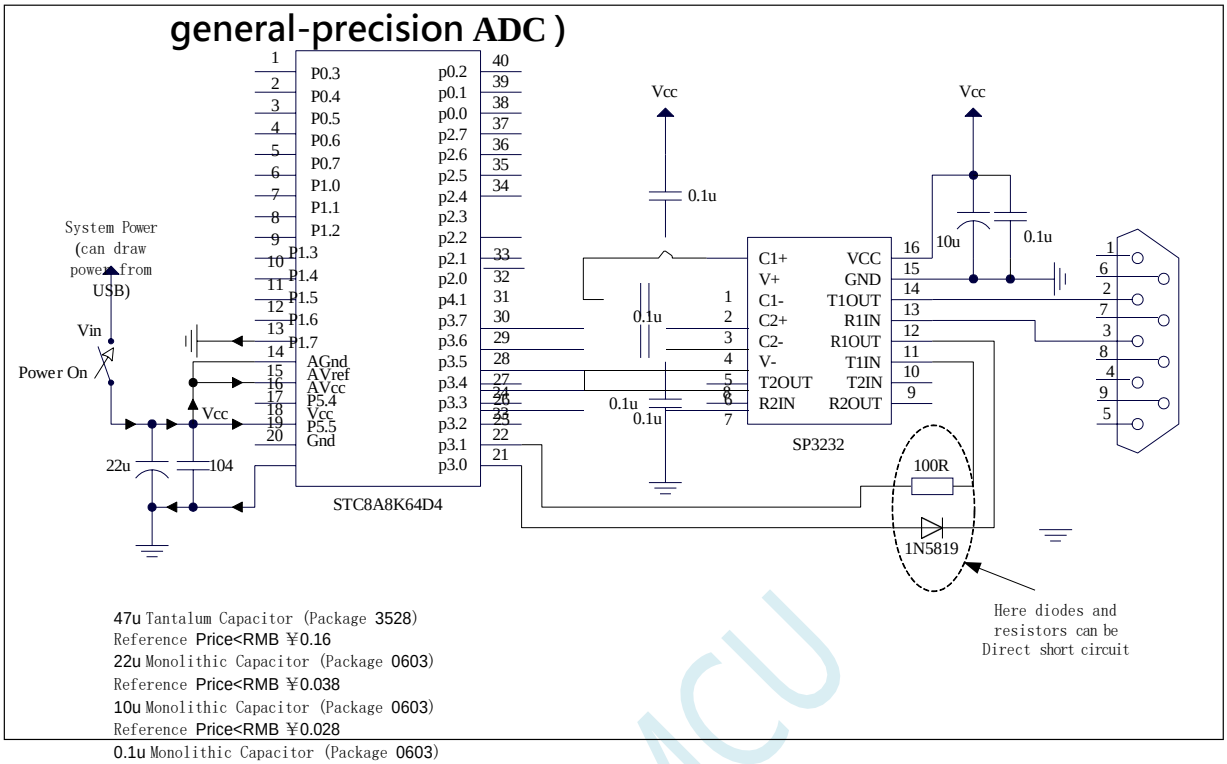
CD431 (Package SOT23) Reference Price <RMB ￥0.10
47u Tantalum Capacitor (Package 3528) Reference Price<RMB ￥0.16
22u Monolithic Capacitor (Package 0603) Reference Price<RMB ￥0.038
10u Monolithic Capacitor (Package 0603) Reference Price<RMB ￥0.028
0.1u Monolithic Capacitor (Package 0603) Reference Price<RMB ￥0.005

Here diodes and resistors can be Direct short circuit

ISP Download steps.

1. Powering down the target chip
2. Click the "Download/Program" button in the STC-ISP download software
3. Powering up the target chip
4. Start ISP Download

5.1.5 Download using RS-232 converter, also supports emulation (using general-precision ADC)

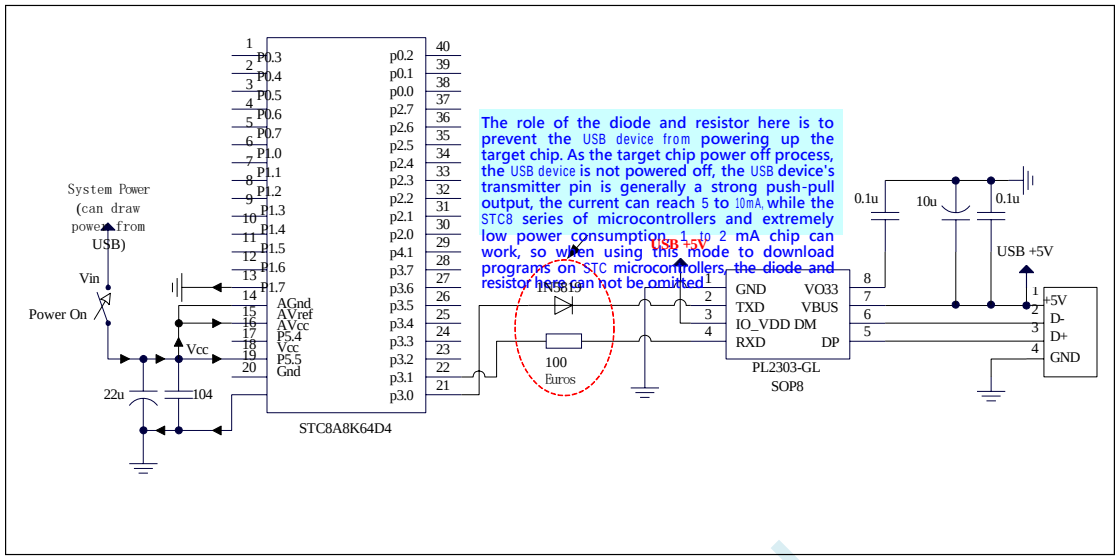


47u Tantalum Capacitor (Package 3528)
Reference Price<RMB ￥0.16
22u Monolithic Capacitor (Package 0603)
Reference Price<RMB ￥0.038
10u Monolithic Capacitor (Package 0603)
Reference Price<RMB ￥0.028
0.1u Monolithic Capacitor (Package 0603)
Reference Price<RMB ￥0.005

ISP Download steps.

1. Powering down the target chip
2. Click the "Download/Program" button in the STC-ISP download software
3. Powering up the target chip
4. Start ISP Download

5.1.6 Use the PL2303-GL download, which also supports emulation

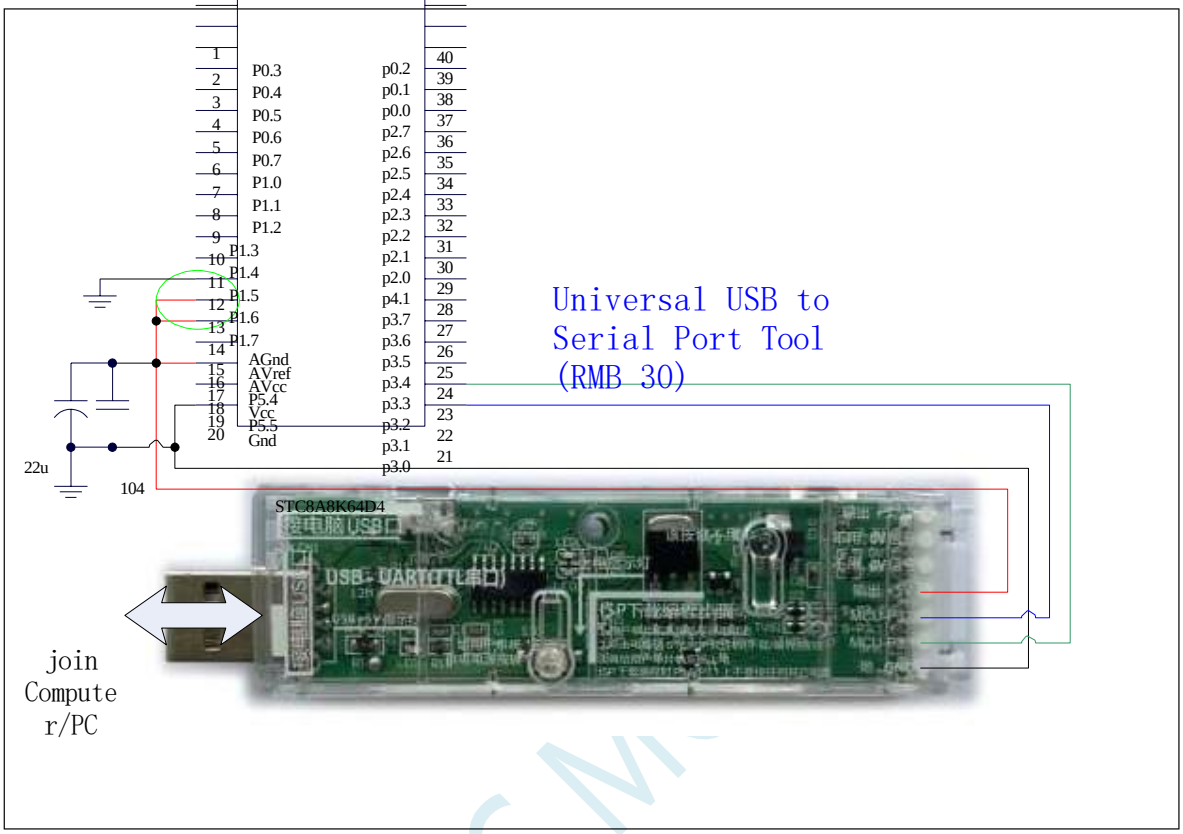


ISP Download steps.

1. Power down the target chip, note that you cannot power down the USB to serial chip (e.g. CH340, PL2303-GL, etc.) Note: The baud rate error of PL2303-SA is very large, we recommend using PL2303-GL.
2. as the USB to serial chip transmitter pin is generally strong push-pull output, you must connect a diode in series between the P3.0 port of the target chip and the transmitter pin of the USB to serial chip, otherwise the target chip can not be completely powered off, failing to achieve the goal of powering down the target chip.
3. Click the "Download/Program" button in the STC-ISP download software
4. Powering up the target chip
5. Start ISP Download

Note: It has been found that when using the USB cable for ISP downloading, the voltage drop on the USB cable is too thin, resulting in insufficient power supply during ISP downloading.

5.1.7 Download using the Universal USB to Serial Tool, which supports ISP online downloads and can also support emulation

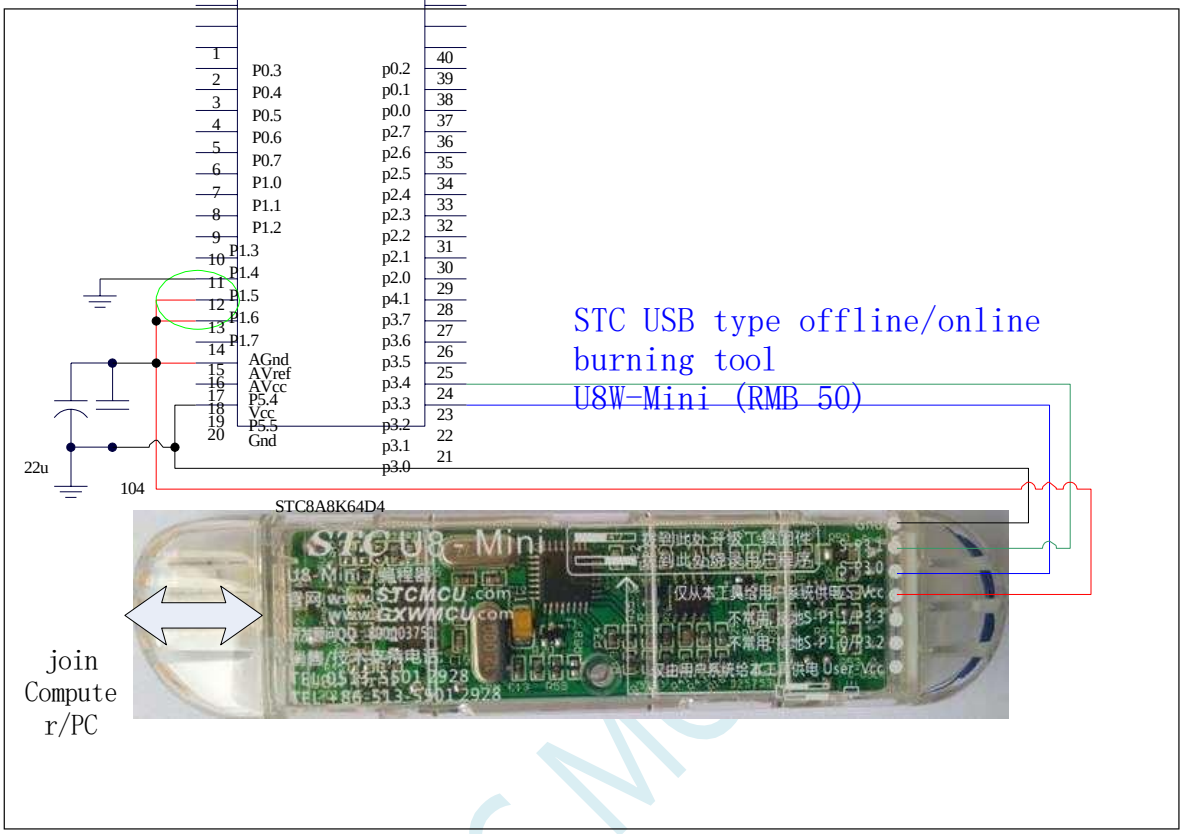


ISP Download steps.

1. Connect the Universal USB to Serial Tool to the target chip as shown in the diagram
2. Press the power button to make sure that the target chip is in a powered-down state (a state where the power-up indicator is off) Note: The tool is not powered externally the first time it is powered up, so you can skip this step if you are using this tool for the first time.
3. Click the "Download/Program" button in the STC-ISP download software
4. Press the power button again to power up the target chip (the power-on indicator is on)
5. Start ISP Download

Note: It has been found that when using the USB cable for ISP downloading, the voltage drop on the USB cable is too thin, resulting in insufficient power supply during ISP downloading.

5.1.8 Download using the U8-Mini tool, which supports ISP online and offline downloads, and also supports emulation



ISP Download steps.

1. Connect the U8-Mini to the target chip as shown in the diagram
2. Click the "Download/Program" button in the STC-ISP download software
3. Start ISP Download

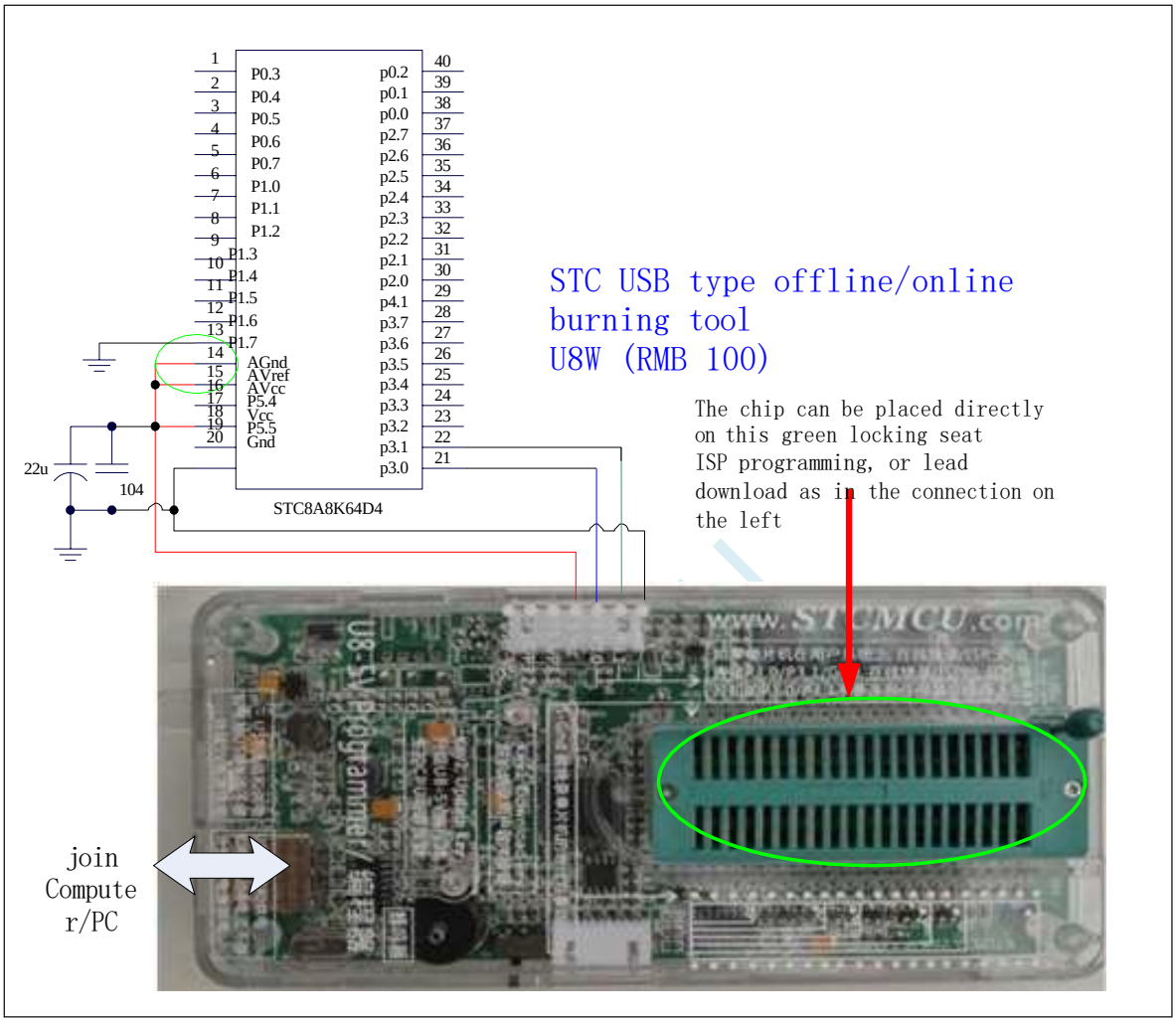
Note: If you use the U8-Mini to power the target system, the total current of the target system must not be greater than 200mA, otherwise the download will fail. **Note: Currently, it has been found that when using the USB cable to power the ISP download, the voltage drop on the USB cable is too thin, resulting in insufficient power supply during the ISP download, so please be sure to use a USB reinforced cable when using the USB cable to power the ISP download.**

To use the U8-Mini for emulation, the U8-Mini must first be set to passthrough mode.The U8W/U8W-Mini implements USB to serial passthrough mode as follows.

1. First, 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 normal download mode.At this time, press and hold the Key1 (Download) button on the tool, then press the Key2 (Power) button, then release the Key2 (Power) button, then release the Key1 (Download) button, the U8W/U8W-Mini will enter USB to serial port pass-through mode. (**Press Key1 Press Key2 Release Key2 Release Key1**)
3. The U8W/U8W-Mini tool in pass-through mode is simply a USB to serial port that does not have offline download function, if you need to restore

for the original function of the U8W/U8W-Mini, simply press the Key2 (power) button again separately.

5.1.9 Download using the U8W tool, which supports ISP online and offline downloads, and can also support imitation real



ISP Download Steps (Connected Method).

1. Connect the U8W to the target chip as shown in the diagram
2. Click the "Download/Program" button in the STC-ISP download software
3. Start ISP Download

Note: If the U8W is used to power the target system, the total current of the target system must not be greater than 200mA, otherwise the download will fail.

ISP download steps (on board approach)

1. Place the target chip in the direction of pin 1 near the locking wrench and pin down flush
2. Click the "Download/Program" button in the STC-ISP download software
3. Start ISP Download

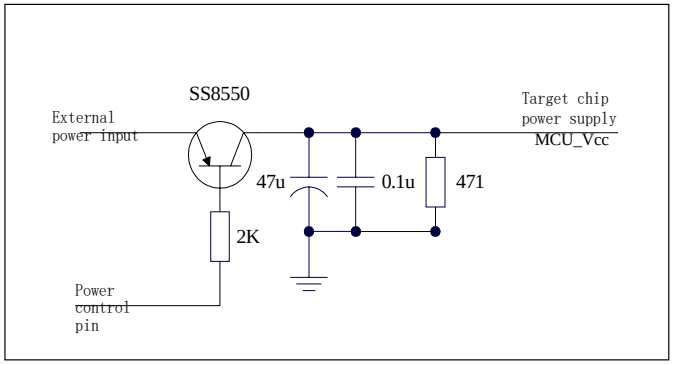
Note: It has been found that when using the USB cable for ISP downloading, the voltage drop on the USB cable is too thin, resulting in insufficient power supply during ISP downloading.

To use the U8W for emulation, the U8W must first be set to passthrough mode.The U8W/U8W-Mini implements USB to serial passthrough mode as follows.

1. First, 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 normal download mode.At this time, press and hold the Key1 (Download) button on the tool, then press the Key2 (Power) button, then release the Key2 (Power) button, then release the Key1 (Download) button, the U8W/U8W-Mini will enter USB to serial port pass-through mode. (**Press Key1 Press Key2 Release Key2 Release Key1**)
3. The U8W/U8W-Mini tool in pass-through mode is simply a USB to serial port that does not have offline download function, if you need to restore

   For the original function of the U8W/U8W-Mini, simply press the Key2 (power) button again separately.

5.1.10 Microcontroller Power Control Reference Electricity Road



5.2 STC-ISP Download Software Advanced Applications

5.2.1 Posting project procedures

The Publish Project program function is essentially the packaging of the user's program code with associated option settings into a super simple executable file with the user's own interface that can be programmed directly to the target chip for download.

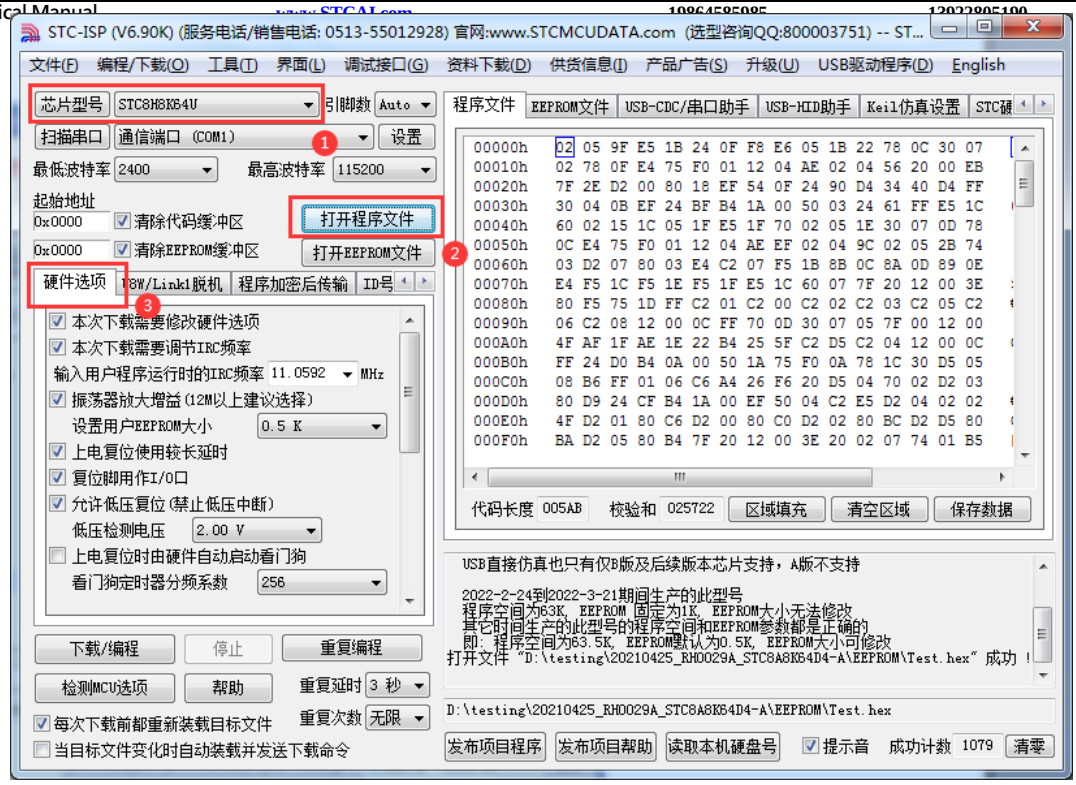
About the interface, the user can customize (the user can modify the title of the release project program, button names and help information) while the user can also specify the target computer's hard disk number and the target chip ID number, specify the target computer's hard disk number, then you can control the release application can only run on the specified computer (to prevent burners will easily steal the program from the computer, such as through the network sent away) (to prevent burners from easily stealing the program from the computer, such as through the network sent away, such as through the USB disk baked away, can not be prevented, of course, steal your computer that can not be done, so the STC offline download tool than computer burning security, can limit the number of burnable chips, so that the receptionist clerk lady burn, let the boss's wife can burn) copy to other computers, the application can not run. Similarly, when the ID number of the target chip is specified, then the user code can only be downloaded to the target chip with the corresponding ID number (especially useful for a device to sell tens of millions of products --- tanks, can be sent to customers to upgrade their own, without risking their lives to run to war-torn Iraq to upgrade the software) for other chips with inconsistent ID numbers, can not be downloaded for programming.

The detailed procedure for issuing a project is as follows.

1、First select the model of the target chip

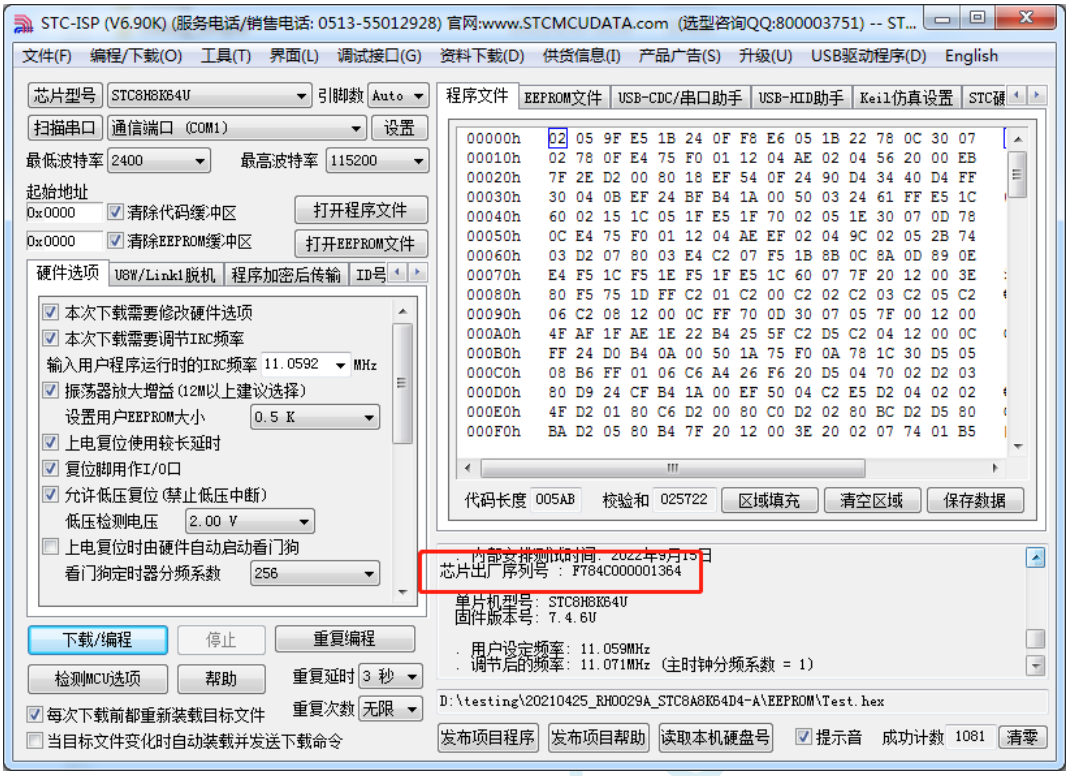
2、Open the program code file

3. Set the appropriate hardware options

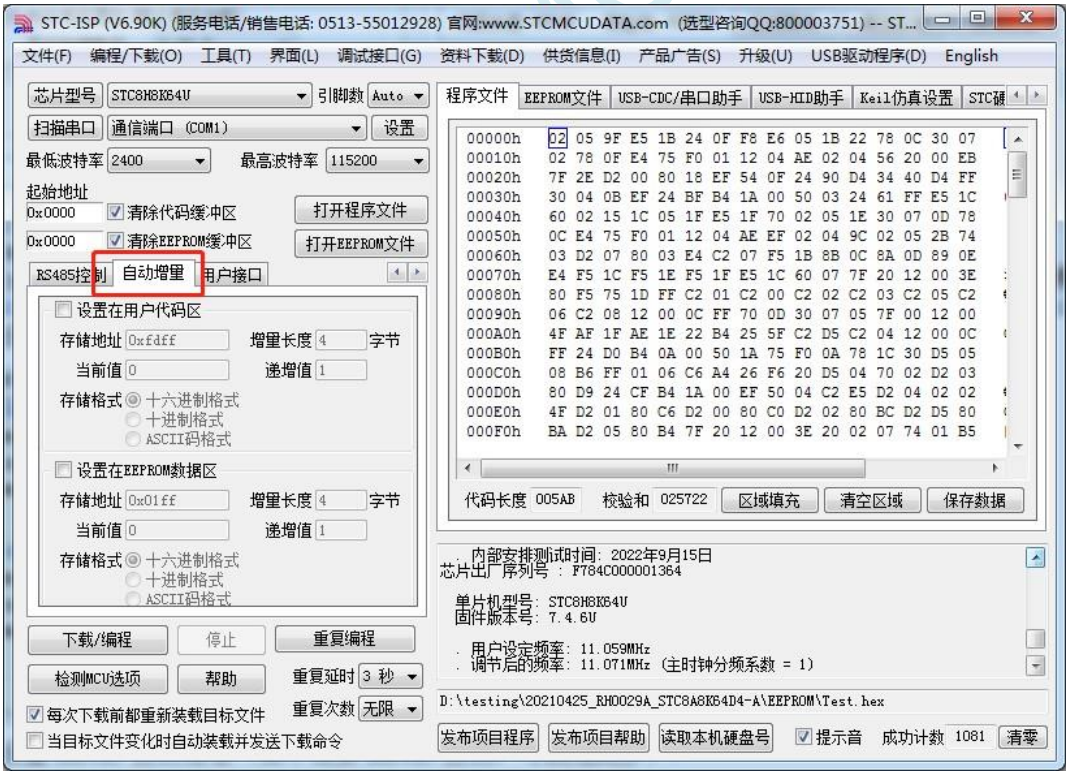


4, try to burn the chip and write down the ID number of the target chip, as shown below, the chip ID number is "F784C000001364"

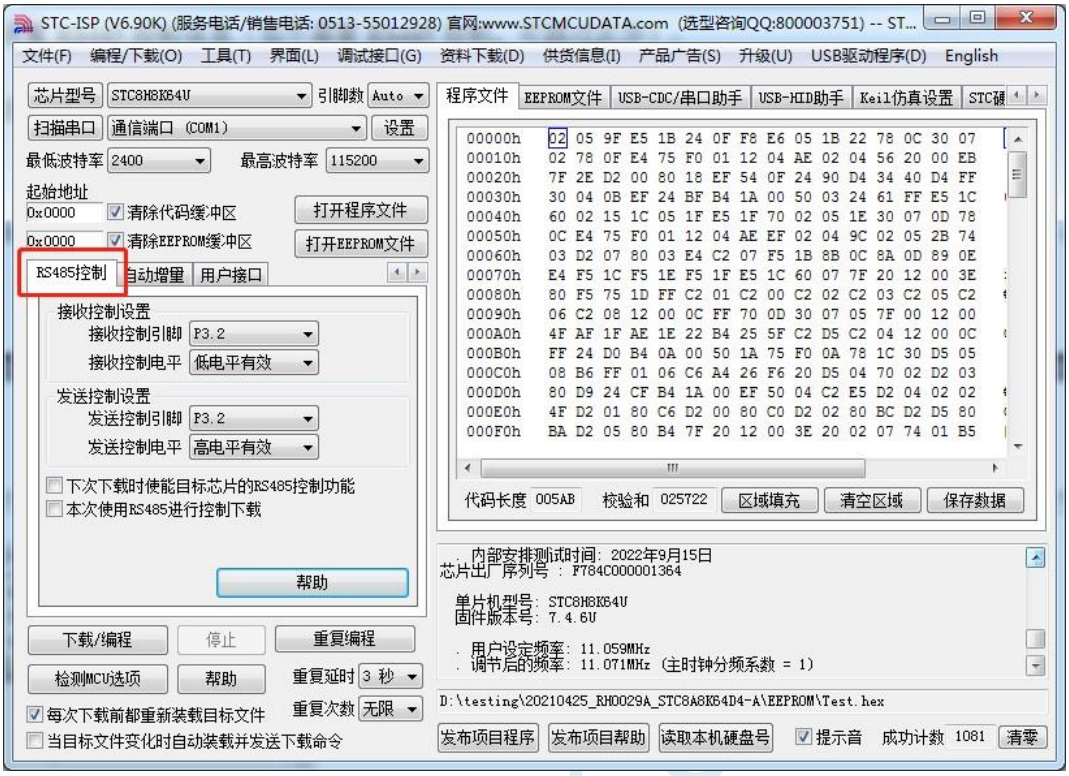
(Skip this step if you do not need to verify the ID number of the target chip)



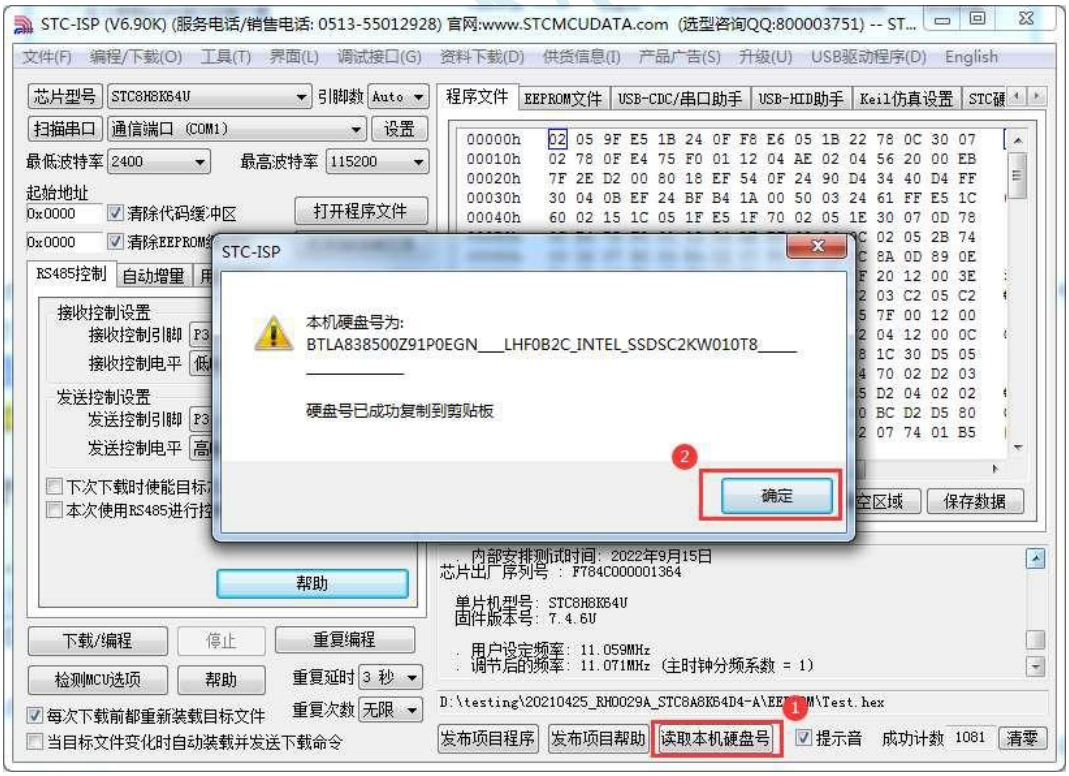
5, set the automatic increment (if you do not need automatic increment, you can skip this step)



6、Setting RS485 control information (if RS485 control is not needed, you can skip this step)



7. Click the "Read local hard disk number" button on the interface and write down the hard disk number of the target computer (if you do not need to verify the hard disk number



of the target computer, you can skip this step)

8. Click the "Publish Project Program" button to enter the setup interface for publishing applications.

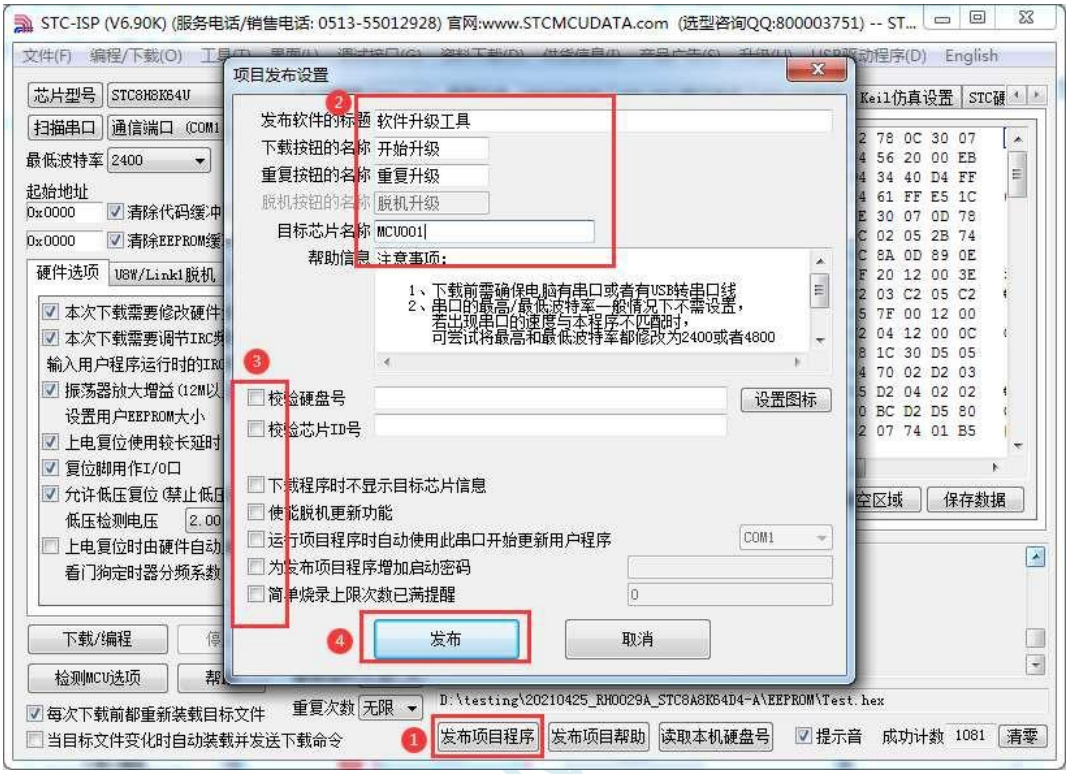
9. Modify the title of the release software, the name of the download button, the name of

the repeat download button, the name of the auto increment and the help information according to your respective needs

10、If you need to verify the hard disk number of the target computer,you need to check the "Verify hard disk number",and enter the text box behind the previous note

The hard drive number of the target computer under

11、If you need to verify the ID number of the target chip, you need to check "Verify chip ID number" and enter the ID number of the target chip noted in the text box at the



back.

12, and finally click the Publish button to save the project release program, you can get the corresponding executable file. The released project program hits the interface as follows

5.2.2 Program encrypted and transferred (to prevent serial port analysis of the program during burn-in)

At present, all ordinary serial port downloading and burning programming are using explicit communication (when the computer and the target chip communication, or offline downloading board and the target chip communication) the problem: If the burner analyzes the data of serial port communication when downloading and burning programming, the master can lead 2 lines out on the serial port when burning, and analyze the actual user program code by analyzing the data of serial port communication. Of course, using STC's offline download board to burn the program is always better than using a computer to burn the program (to prevent the burner will easily steal the program from the computer, such as through the network sent away, such as through the U disk baked away, can not prevent, of course, steal your computer that can not be done, so STC's offline download tool is safer than the computer burn, let the front desk clerk lady burn, let the boss's wife burn can be) Even the STC world's first offline download tool, for to prevent the genius of the unscrupulous in the offline download tool burning process by analyzing the serial communication data to analyze the actual user program code, there is no way to achieve the requirements, which requires the use of the latest STC microcontroller provided by the program encryption after the transfer function.

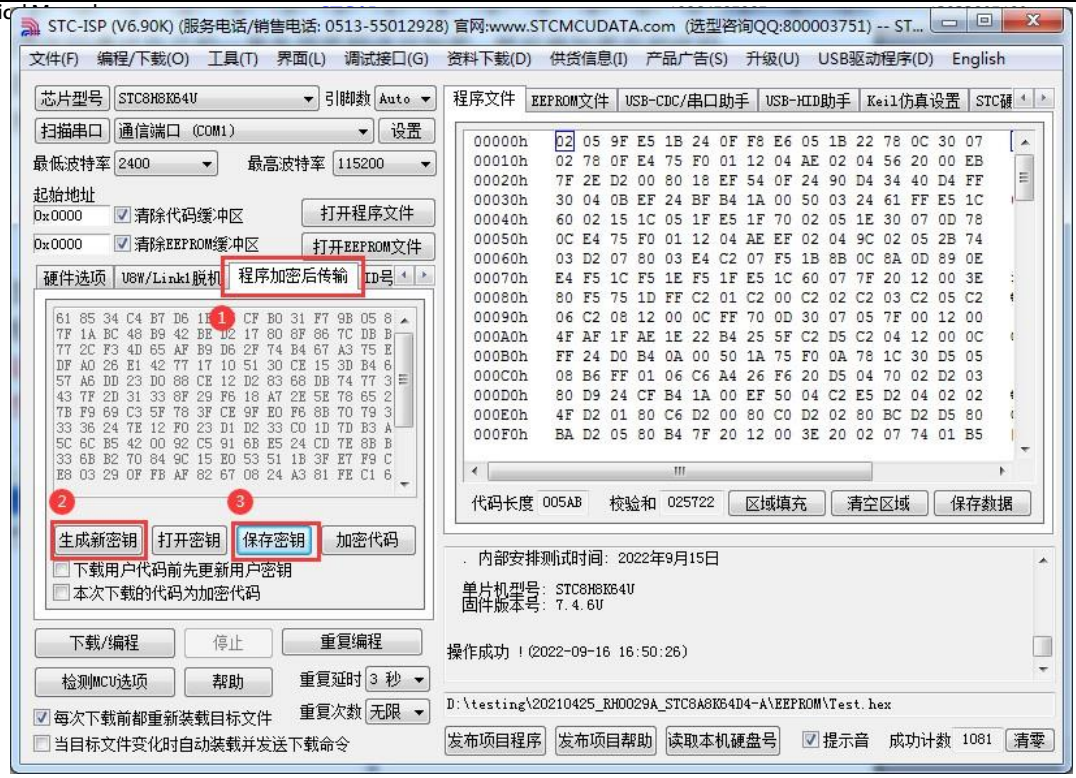
Program encryption after transmission download is the user will first encrypt the program code through their own set of special keys, and then the encrypted code and then download through the serial port, at this time the download transmission is encrypted file, through the serial port analysis out of the encrypted garbled code, such as not by sending someone to sneak into your company to steal your computer inside the encryption key, there is no value, it will play a role in preventing the burner program when the burner personnel The purpose of monitoring the serial port to analyze the code.

The use of the post-program encryption transfer function requires several steps as follows.

1. Generate and save the new key

   As shown below, go to the "Transfer after encryption" page and click the "Generate new key" button to display the newly generated 256-byte key in the buffer. Then click the "Save Key" button to save the new key as a key file with ". K" extension (**note: this key file must be saved, as all future code files need to be encrypted with this key, and the key generation is non-repeating, i.e. it is impossible to generate two identical keys at any time, so once**

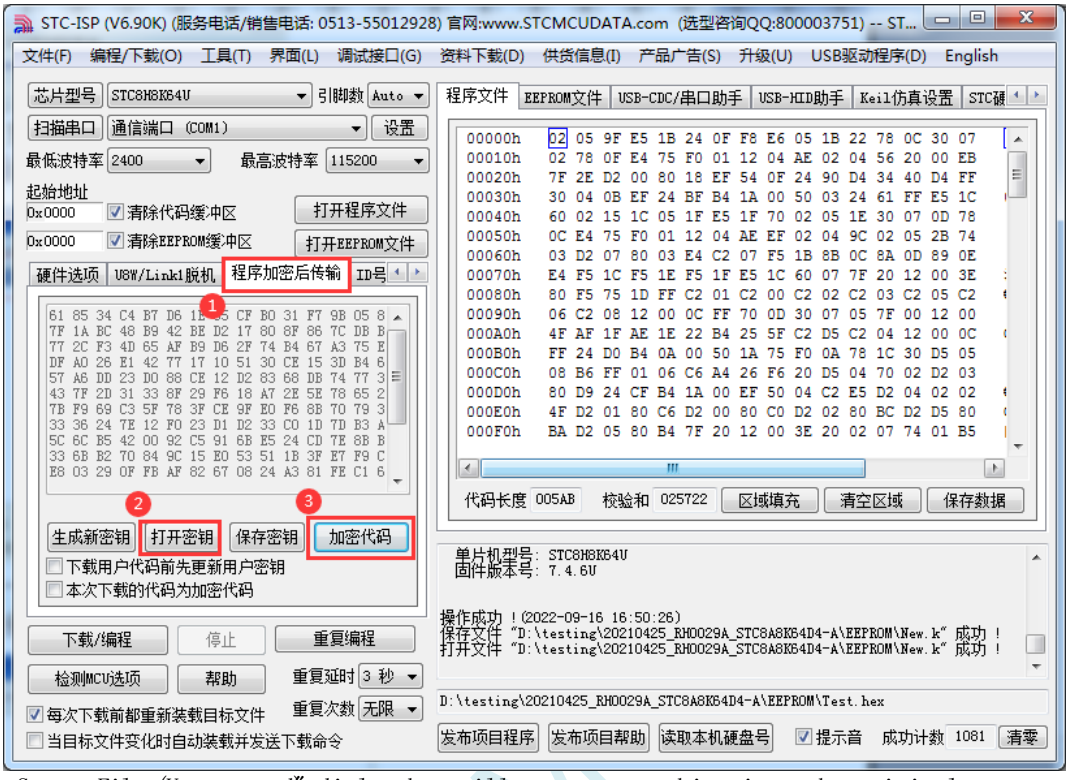
   Lost key file will not be regained) For example, we save the key as "New.k".



2. Encryption of code files

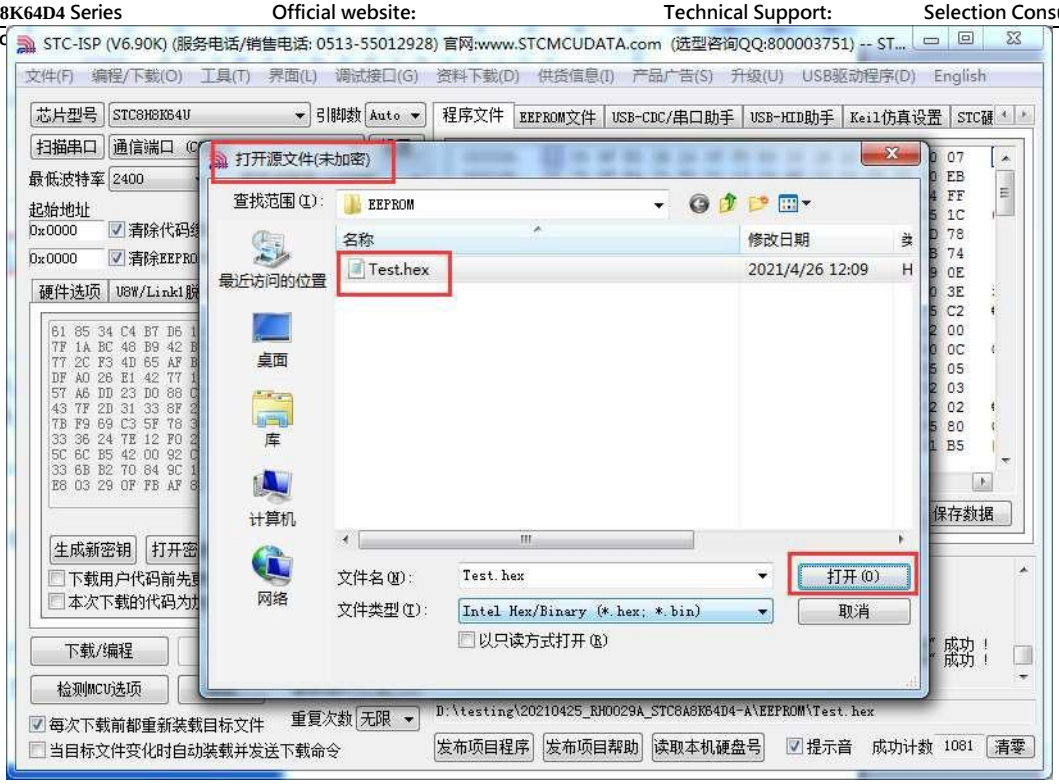
Before encrypting the file, we need to open our own key. If our key is already stored in the buffer, do not open it again. As shown below, click the "Open key" button on the "Transfer after encryption" page to open the key file we saved earlier.

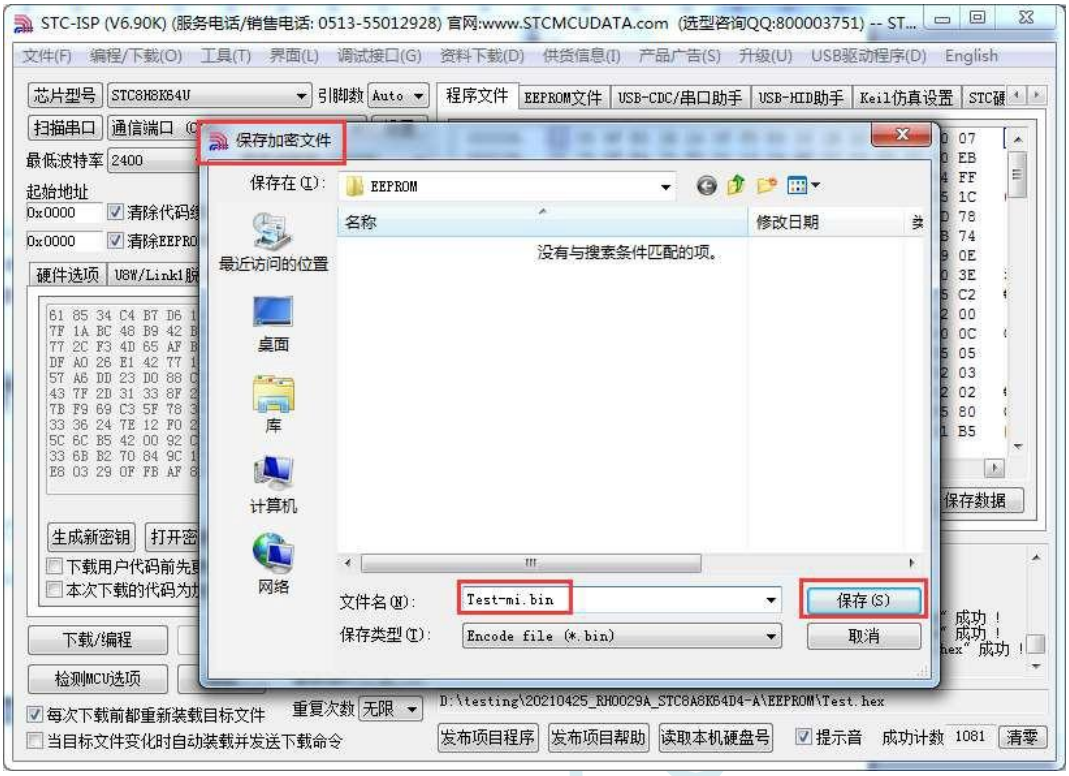
For example, "New.k", then go back to the "Transfer after encryption" page and click the "Encrypt code" button, as shown in the following figure, firstly, a pop-up will appear "Open



Source File (Unencrypted)" dialog box will pop up, at this time, the original unencrypted code file is selected.

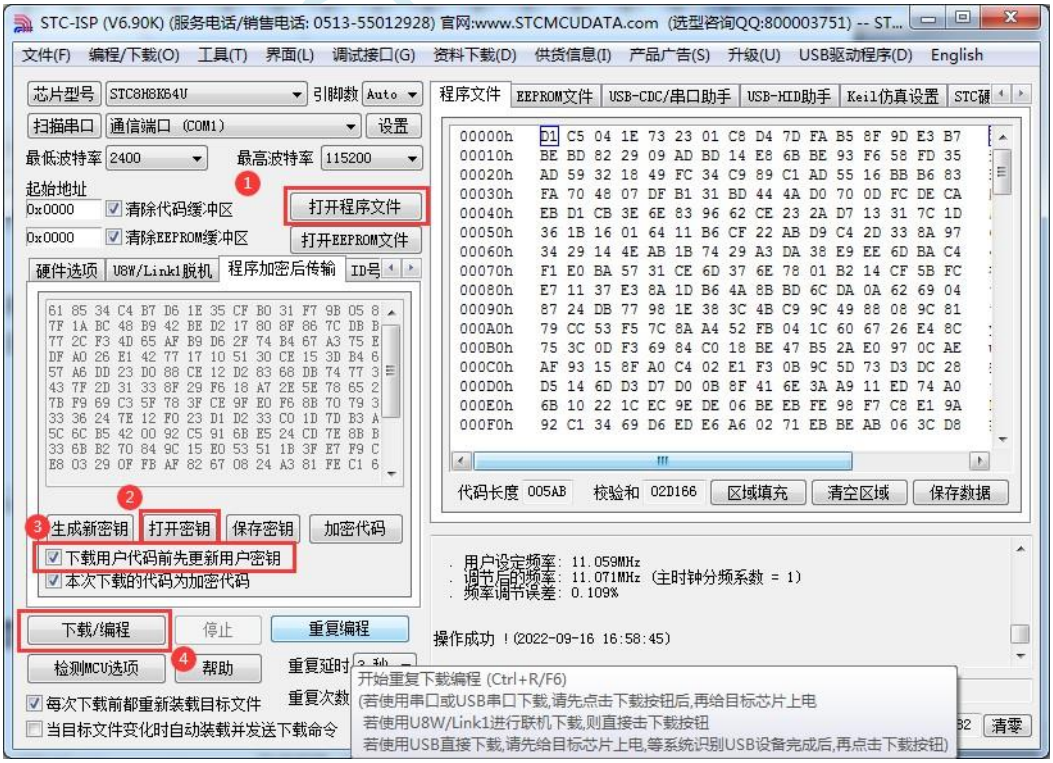
After clicking the Open button, a similar dialog box will pop up immediately, but this is a dialog box for saving the encrypted file. As shown in the figure below, click the Save button to save the encrypted file.


STC-ISP (V6.90K) (服务电话/销售电话: 0513-55012928) 官网:www.STCMCUDATA.com (选型咨询QQ:800003751) -- ST...
文件(F) 编程/下载(O) 工具(T) 界面(L) 调试接口(G) 资料下载(D) 供货信息(I) 产品广告(S) 升级(U) USB驱动程序(D) English
芯片型号 STC8H8K64U 引脚数 Auto
程序文件 EEPROM文件 USB-CDC/串口助手 USB-HID助手 Keil仿真设置
扫描串口 通信端口
最低波特率 2400
起始地址
0x0000 清除代码
0x0000 清除EEPRO
硬件选项
生成新密钥
下载用户代码前先
本次下载的代码为
下载/编程
检测MCU选项
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
重复次数 无限
保存数据
成功 !
成功 !
D:\testing\20210425_RH0029A_STC8A8K64D4-A\EEPROM\Test.hex
发布项目程序 发布项目帮助 读取本机硬盘号 提示音 成功计数 1081 清零
打开源文件(未加密)
查找范围(I): EEPROM
最近访问的位置
桌面
库
计算机
网络
名称
修改日期
Test.hex
2021/4/26 12:09
文件名(N): Test.hex
文件类型(T): Intel Hex/Binary (*.hex; *.bin)
以只读方式打开(R)
打开(O)
取消



3. Update the user key to the target chip

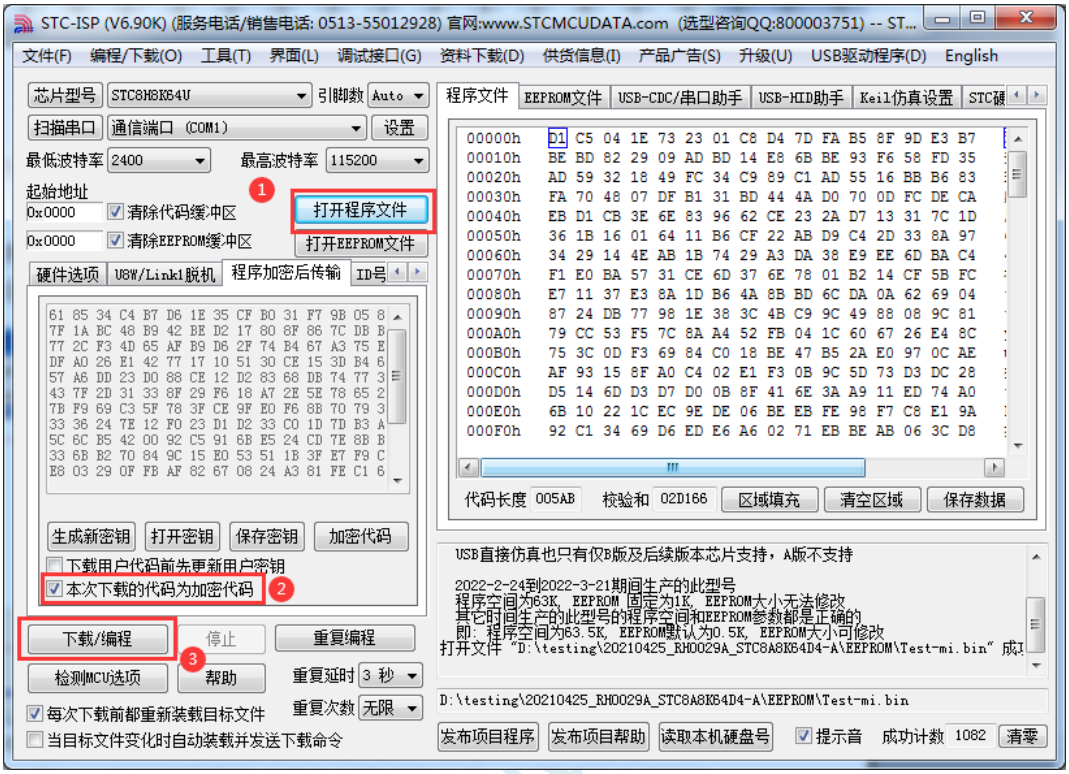
Before updating the key, we need to open our own key. If our key is already stored in the buffer, do not open it again. As shown below, click the "Open Key" button on the "Custom Encryption Download" page to open our previously saved key file, e.g. "New.k". After the key is opened, as shown below, check the "Update user key before downloading user code" option and "The downloaded code is encrypted code" option, then open our previously encrypted file and click the



"Download/Program" button at the bottom left corner of the interface. Click the "Download/Program" button in the bottom left corner of the interface, and download the target chip in the normal way to update the user key.

4. Encrypted update of user code

After the key is updated successfully, the target chip will have the function to receive the encrypted code and restore it. At this time, if you need to update/update



the code again, you only need to refer to the second step to encrypt the target code, and then the following figure

For a new STC microcontroller, you can combine step 3 and step 4, i.e. update the key to the target microcontroller and download the encrypted code to the microcontroller at the same time, if you have already performed step 3 (i.e. you have already updated the key to the target chip), only need to follow step 4 for the subsequent code update, just select the option "Transfer the encrypted code after the program" on the "Transfer the encrypted code" page. In the "Transfer after encryption" page, select the option of "The downloaded code is encrypted code" (no need to select the option of "Update user key before downloading user code"), then open the file we encrypted before, open it and click Click the "Download/Program" button in the lower left corner of the interface, and download the target chip in the normal way to complete the purpose of updating the user code with the user's own encrypted file (to prevent the code from being analyzed by the burner through the monitoring serial port when burning the program)

5.2.3 Release Project Program + Program encrypted for post-transmission use in combination

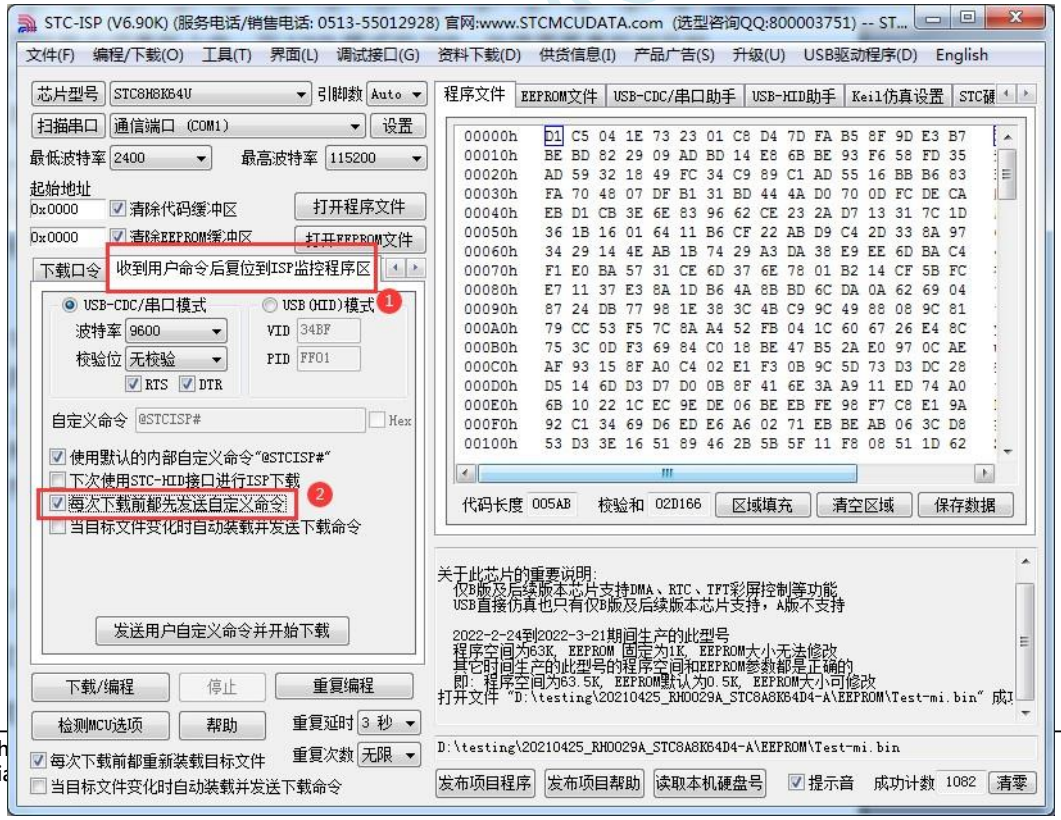
Two new special features can be used in conjunction with the release of the project program and the post-encryption transfer of the program. First of all, the post-encryption transmission of the program ensures the confidentiality of the user's code during the serial communication transfer during the programming burn-in, while the release of the project program allows the end-user to update the program remotely (without the need for the solution company's personnel to be physically present) So the combination of the two functions is very suitable for the solution company/manufacturer when the software needs to be updated, so that the end user can update the software of the end product by themselves, but also to ensure that the field burner can not analyze the useful program through the serial port, and is highly recommended for the solution company.

5.2.4 User-defined downloads (for non-stop downloads)

Downloading the user's target program to the STC microcontroller is accomplished by executing the microcontroller's internal ISP system code and the host computer for serial or USB communication. However, the internal ISP code of the STC microcontroller will only be executed every time the power is re-stopped and re-powered, which requires the user to re-power the target microcontroller every time he needs to update the program, while for the USB mode ISP, in addition to re-powering the target chip, the P3.2 port needs to be pulled down to GND at power-up. For projects in the development stage, the code needs to be modified and updated frequently, and it is very troublesome to re-power the target chip every time it is downloaded.

The STC microcontroller adds a soft reset register (IAP_CONTR) to the hardware design, allowing the user to set this register to determine whether the CPU resets and re-executes the user code or resets to the ISP area to execute the ISP system code. When 0x20 is written to the IAP_CONTR register, the CPU resets and re-executes the user code; when 0x60 is written to the IAP_CONTR register, the CPU resets and resets to the ISP area to execute the ISP system code.

To implement a non-stop ISP download, the user can design a piece of code in the program, such as detecting a special key, or monitoring the serial port waiting for a special serial command, and when the download condition is detected, the software triggers the soft reset register to reset to the ISP area to execute the ISP system code, thus achieving a non-stop ISP download. When the trigger condition is an external key, the key status can be monitored in real time in the user code. To fully synchronize the STC-ISP software with the user-triggered soft reset, use the "Reset to ISP monitor area after receiving user command" function provided in the STC-ISP software.



The steps to achieve a non-stop ISP download are as follows.

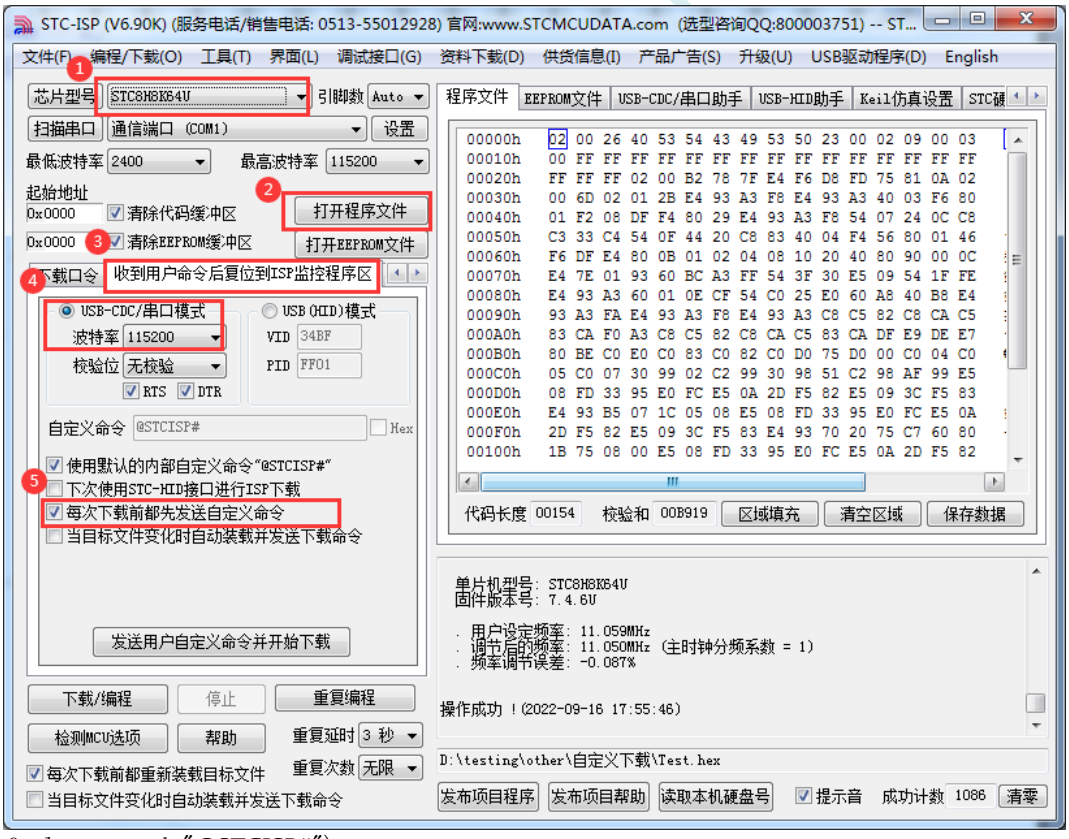
1. Write user code and add serial port command monitoring program to the user code (Reference code below, test microcontroller model is STC8H8K64U) #include "stc8h.h"

```
#define    FOSC        11059200UL
#define    BAUD        (65536 - FOSC/4/115200)

char code *STCISPCMD = "@STCISP#";          //custom download command char index;

void uart_isr() interrupt 4
{
    char dat;

    if (TI)
    {
        TI = 0;
    }

    if (RI)
    {
        RI = 0;
        dat = SBUF;                         //Receive data from serial port

        if (dat == STCISPCMD[index])        //determine if the received data and the current command character match
        {
            index++;                        //index+1 if match
            if (STCISPCMD[index] == '\0')   //judge if the command match is complete IAP_CONTR = 0x60;    //soft reset to ISP if matching is complete
        }
        else
        {

            index = 0;                      //If it doesn't match, you need to start from the beginning if (dat == STCISPCMD[index])
        }       index++;
    }
}

void main()
{
    P0M0 = 0x00; P0M1 = 0x00;
```

```
P1M0 = 0x00; P1M1 = 0x00;
```

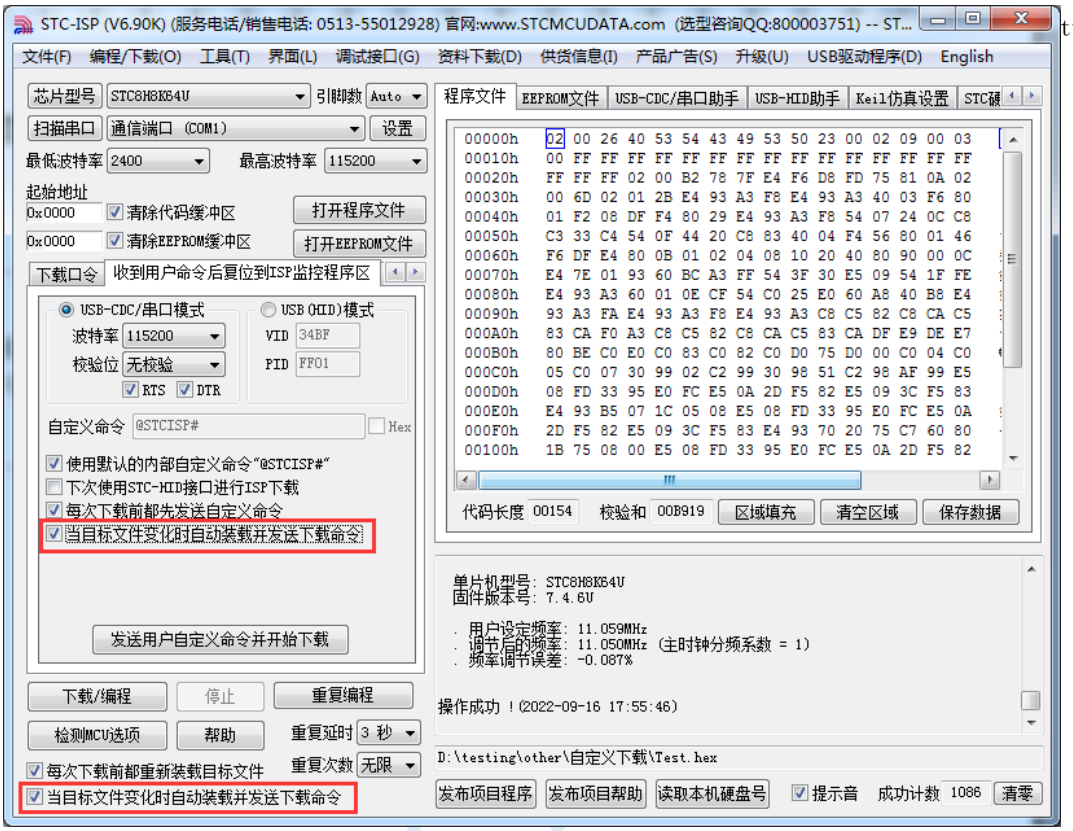
```
        P2M0 = 0x00; P2M1 = 0x00;
        P3M0 = 0x00; P3M1 = 0x00;

        SCON = 0x50;                        //Serial port initialization
        AUXR = 0x40;
        TMOD = 0x00;
        TH1 = BAUD >> 8;
        TL1 = BAUD;
        TR1 = 1;
        ES = 1;
        EA = 1;

        index = 0;                          //initialize command

        while (1);
    }
```

2. Set the custom download command as shown in the following diagram (the example uses the STC default command "@STCISP#")



3, the first download needs to re-power the target microcontroller, after each update only need to click the "Download/Program" button in the download software, the download software will automatically send the download command to the target microcontroller, the target microcontroller received the command automatically reset to the system ISP area, you can achieve non-stop update of user code.

4, STC-ISP can also realize the project development stage, fully automatic download function, that is, when the download software detects that the target code has been



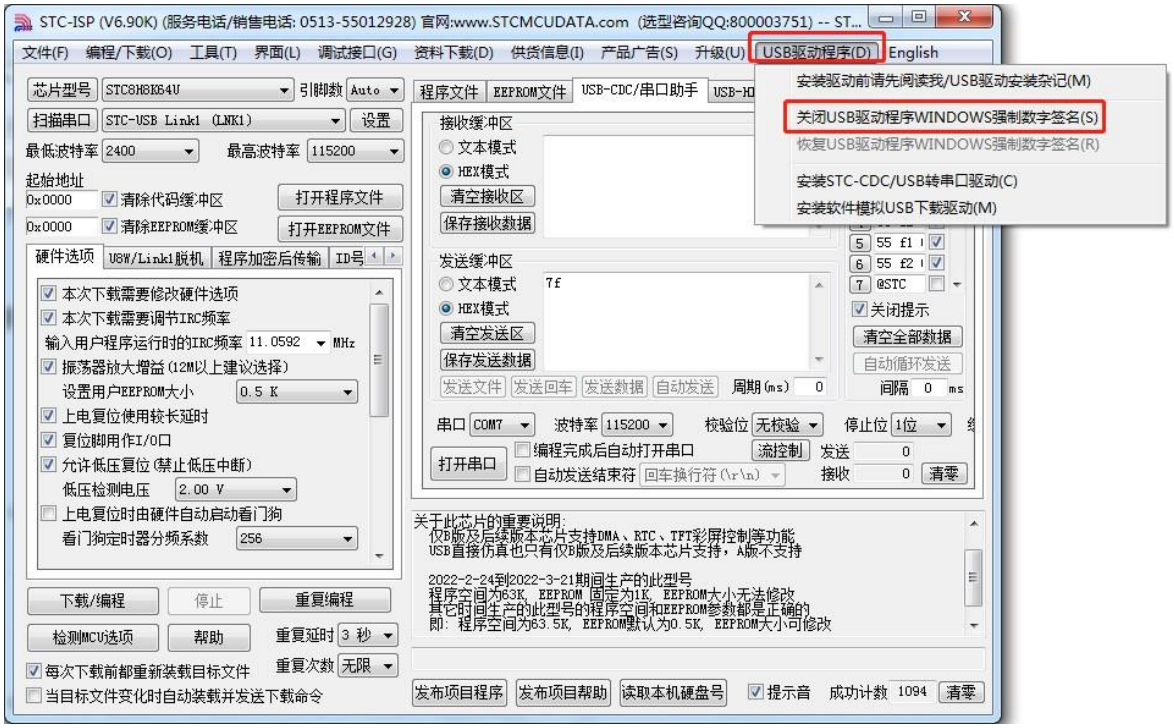
ture,

5.3 Instructions related to digital signature of drivers

5.3.1 About mandatory digital signatures for drivers

Windows systems starting with the Vista version of 64-bit Windows force the driver to be digitally signed for installation, otherwise the driver cannot be installed and the USB device cannot be used properly. Currently, the drivers provided by STC are not WHQL certified (Microsoft logo certified) at the moment, and may not be installed successfully on 64-bit Windows systems starting from Vista until the mandatory digital signature of the drivers is turned off.

The latest STC-ISP download software provides a simple disabling of the driver's forced digital signatureas shown in the menu directory below.

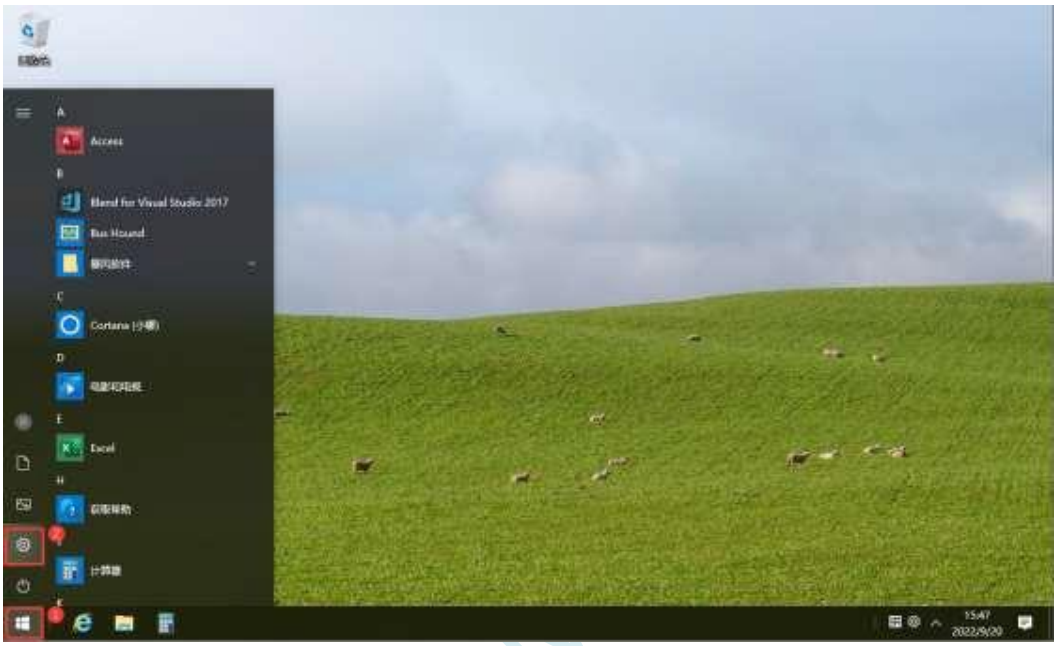


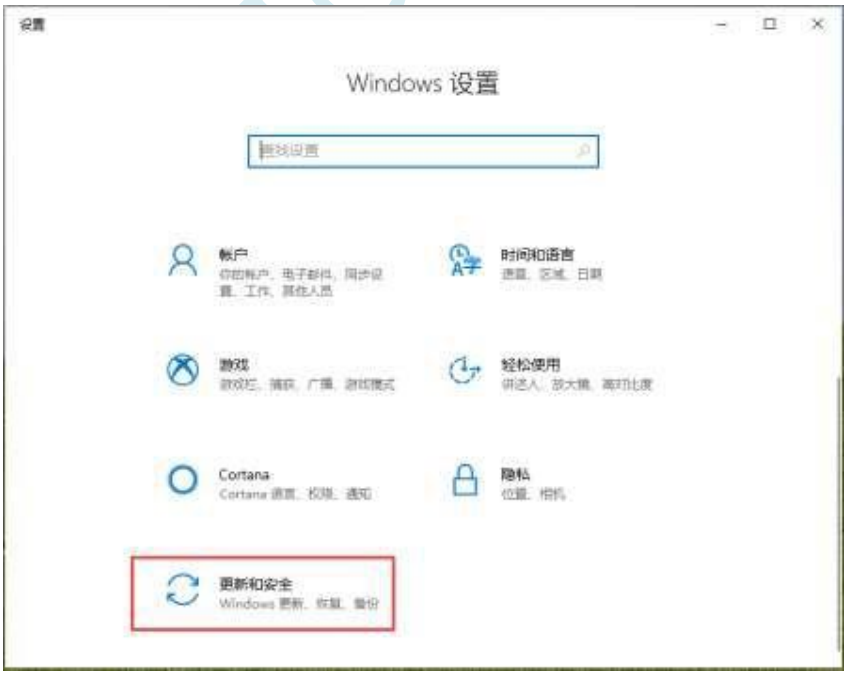
This menu item uses the TestSigning and NoIntegrityChecks parameters automatically set by the system's BCDEDIT utility to temporarily disable the driver's forced digital signature. For systems with Secure Boot enabled, the BCDEDIT utility will not work and Secure Boot needs to be disabled in the BIOS (set Secure Boot to Disable)

Currently tested WinXP/Win7-32/Win7-64 systems can disable digital signature normally, but Win10 and later systems do not work. To disable digital signatures on Win10 and later systems, you still need to disable forced digital signatures from the system boot menu. Please refer to the next section for detailed steps

5.3.2 Windows 10 Steps to turn off mandatory digital signatures for drivers

Select the "Settings" function in the "Start" menu





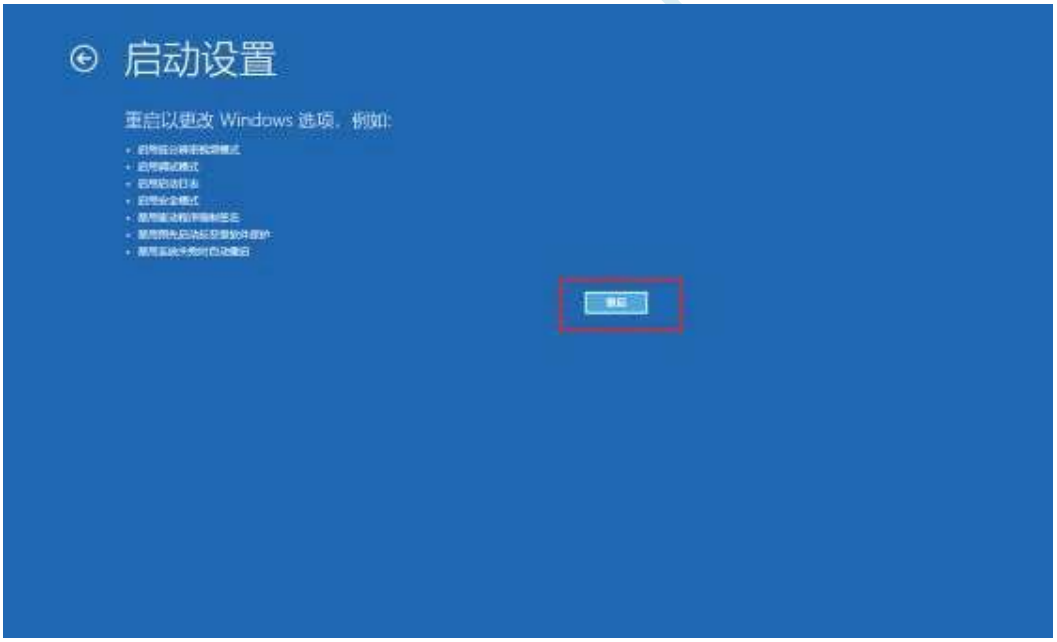

设置
主页
查找设置
更新和安全
Windows 更新
传递优化
Windows 安全中心
备份
疑难解答
恢复
激活
查找我的设备
开发者选项
恢复
重置此电脑
如果电脑未正常运行，重置电脑可能会解决问题。重置时，可以选择保留个人文件或删除个人文件，然后重新安装 Windows。
开始
高级启动
通过设备或磁盘(如 U 盘或 DVD)启动，更改你的电脑固件设置，更改 Windows 启动设置，或者从系统映像还原 Windows。 这将重新启动电脑。
立即重新启动
更多恢复选项
了解如何进行 Windows 的全新安装以便开始全新的体验
备份文件


选择一个选项
继续
关闭电脑
使用设备
疑难解答


疑难解答
重置此电脑
高级选项


高级选项
系统还原
启动修复
卸载更新
命令提示符
系统映像恢复
UEFI 固件设置
查看更多恢复选项


高级选项
启动设置
查看更多恢复选项


启动设置
重启以更改 Windows 选项，例如：

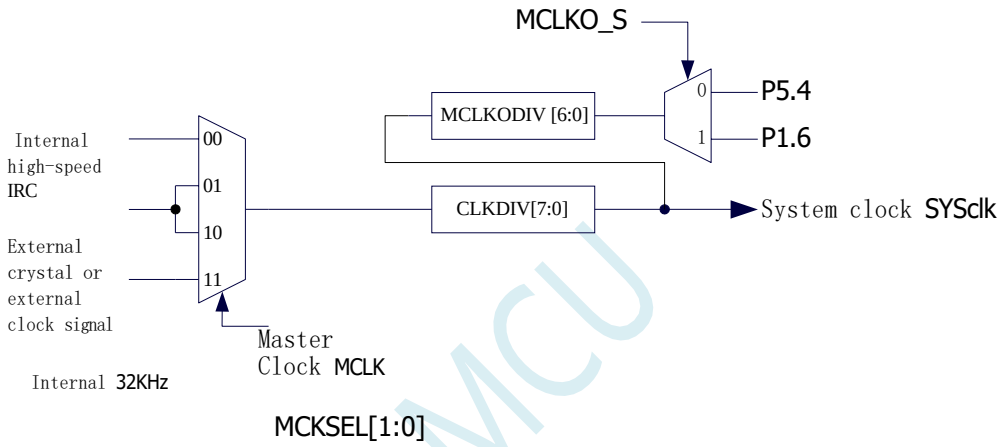
The driver's mandatory digital signature can be temporarily disabled after a system reboot

6 Clock, Reset, Power Save Mode and System Power Management Management

6.1 System clock control

The system clock controller provides the clock source for the microcontroller CPU and all peripheral systems. There are three clock sources available for the system clock: internal high precision IRC, internal 32KHz IRC (with large error) and external crystal. The user can programmatically enable and disable each clock source, as well as provide internal clock division to reduce power consumption.

After the microcontroller enters power-down mode, the clock controller will turn off all clock sources



System Clock Structure Diagram

Related Registers

sym bolic	description	addre ss	Bit Addresses and Symbols								reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
CLKSEL	Clock Select Register	FE00H	-						MCKSEL[1:0]		xxxx,xx00
CLKDIV	Clock division register	FE01H									Nnnn, nnnn
HIRCCR	Internal high-speed oscillator control register	FE02H	ENHIRC	-	-	-	-	-	-	HIRCST	1xxx,xxx0
XOSCCR	External crystal control register	FE03H	ENXOSC	XITYPE	XCFILTER[1:0]		NMXCG	-	-	XOSCST	0000,0xx0
IRC32KCR	Internal 32K oscillator control register	FE04H	ENIRC32K	-	-	-	-	-	-	IRC32KST	0xxx,xxx0
MCLKOCR	Master Clock Output Control Register	FE05H	MCLKO_S	MCLKODIV [6:0]							0000,0000
IRCDB	Internal IRC De-Shake Control	FE06H	IRCDB[7:0]								1000,0000

6.1.1 System Clock Select Register (CLKSEL)

symbolic	address	B7	B6	B5	B4	B3	B2	B1	B0
CLKSEL	FE00H	-						MCKSEL[1:0]	

MCKSEL[1:0]: master clock source selection

MCKSEL[1:0]	master clock source
00	Internal high precision IRC
01	External crystal oscillator or external input clock signal
10	-
11	Internal 32KHz low speed IRC

6.1.2 Clock division register (CLKDIV)

symbolic	address	B7	B6	B5	B4	B3	B2	B1	B0
CLKDIV	FE01H								

CLKDIV: Master clock dividing factor. The system clock SYSCLK is the clock signal after dividing the master clock MCLK.

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

Note: After a user program reset, the system will automatically set the initial value of this register based on the crossover factor required to set the operating frequency during the last ISP download

6.1.3 Internal High Speed High Precision IRC Control Register (HIRCCR)

symbolic	address	B7	B6	B5	B4	B3	B2	B1	B0
HIRCCR	FE02H	ENHIRC	-	-	-	-	-	-	HIRCST

ENHIRC: Internal high speed, high accuracy IRC enable bit

0: Turn off internal high-precision IRC

1: Enables internal high-precision IRC

HIRCST: Internal high precision IRC frequency stabilization flag bit.(read-only bit)

When the internal IRC is enabled from the stop state, a period of time must pass before the oscillator frequency stabilizes, and when the oscillator frequency stabilizes, the clock controller automatically sets the HIRCST flag bit to 1. So when the user program needs to switch the clock to use the internal IRC, it must first set ENHIRC=1 to enable the oscillator, and then keep querying the oscillator stabilization flag bit HIRCST until the flag bit changes to 1, and then the clock source switch.

6.1.4 External Oscillator Control Register (XOSCCR)

symbolic	address	B7	B6	B5	B4	B3	B2	B1	B0
XOSCCR	FE03H	ENXOSC	XITYPE	XCFILTER[1:0]		NMXCG	-	-	XOSCST

ENXOSC: External crystal oscillator enable bit 0:

Turn off external crystal

oscillator

1: Enables external crystal oscillator

XITYPE: External clock source type

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

1: The external clock source is a crystal oscillator. The signal source is connected to XTALI (P1.7) and XTALO (P1.6) of the microcontroller XOSCST: External crystal oscillator frequency stabilization flag bit.(read-only bit)

When an external crystal oscillator is enabled from a stopped state, a period of time must pass before the oscillator frequency stabilizes when the oscillator

Once the frequency is stable, the clock controller automatically sets the XOSCST flag to position 1. So when the user program needs to switch the clock to use an external crystal oscillator, it must first set ENXOSC=1 to enable the oscillator, and then keep querying the oscillator stability flag bit XOSCST until the flag bit changes to 1 before the clock source can be switched.

XCFILTER[1:0]: external crystal oscillator anti-interference control register

00: Selectable when the external crystal oscillator frequency is 48M or less

01: Selectable when the external crystal oscillator frequency is 24M or less

1x: Selectable for external crystal oscillator frequencies of 12M and below

NMXCG: External crystal

oscillator gain control 0:

Low gain

1: High gain

6.1.5 Internal 32KHz low-speed IRC control register (IRC32KCR)

symbolic	address	B7	B6	B5	B4	B3	B2	B1	B0
IRC32KCR	FE04H	ENIRC32K	-	-	-	-	-	-	IRC32KST

ENIRC32K: Internal 32K low-speed IRC enable bit

0: Turn off internal 32K low-speed IRC

1: Enables internal 32K low-speed IRC

IRC32KST: Internal 32K low-speed IRC frequency stabilization flag bit.(read-only bit)

When the internal 32K low-speed IRC is enabled from the stop state, a period of time must elapse before the oscillator frequency stabilizes, and when the oscillator frequency stabilizes, the clock controller automatically sets the IRC32KST flag bit to 1. So when the user program needs to switch the clock to use the internal 32K low-speed IRC, it must first set ENIRC32K=1 to enable the oscillator, and then keep querying the oscillator stability flag bit IRC32KST until the flag bit changes to 1, then it can The clock source switch is performed.

6.1.6 Master Clock Output Control Register (MCLKOCR)

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

MCLKODIV[6:0]: master clock output dividing factor

(Note: the clock source of the master clock divider output is the system clock after CLKDIV dividing)

MCLKODIV [6:0]	System clock divider output frequency
0000000	No clock output
0000001	SYSClk/1
0000010	SYSClk /2
0000011	SYSClk /3
...	...
1111110	SYSClk /126
1111111	SYSClk /127

MCLKO_S: System clock output pin selection 0: System clock divider output to P5.4 port
1: System clock divider output to P1.6 port

6.2 STC8A8K64D4 Series Internal IRC Frequency Tuning

The STC8A8K64D4 series microcontrollers all have a high precision internal IRC oscillator integrated inside. When users download using the ISP download software, the ISP download software will automatically adjust the frequency according to the user's selection/setting, and the general frequency value can be adjusted to ±0.3% or less, and the temperature drift of the adjusted frequency can reach -1.35% to 1.30% in the full temperature range (-40°C to 85° C).

The STC8A8K64D4 series has four internal IRC bands with center frequencies of 6MHz, 10MHz,27MHz, and 44MHz, respectively. The adjustment range for the 44M band is 29MHz to 55MHz (Note: there may be a manufacturing error of about 5% for different chips and different generation batches) It is recommended that users do not set the IRC frequency higher than 45MHz during ISP download.

Note: For general users, the internal IRC frequency adjustment can be left alone, because the frequency adjustment is done automatically when the ISP download is performed. Therefore, if the user does not need to adjust the frequency by himself, then the following 4 registers should not be modified arbitrarily, otherwise it may cause the operating frequency to change.

If you need to dynamically select the chip's preset frequencies in your own code, please refer to the list of preset frequencies and the "User-defined Internal

Example program for "IRC Frequency

The internal IRC frequency adjustment uses the

following 4 registers to adjust the relevant registers

symbolic	description	address	Bit Addresses and Symbols								reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
IRCBAND	IRC Band Selection	9DH	-	-	-	-	-	-	SEL[1:0]		0000,00nn
LIRTRIM	IRC Frequency Trim Register	9EH	-	-	-	-	-	-	-	LIRTRIM	0000,000n
IRTRIM	IRC Frequency Trim Register	9FH	IRTRIM[7:0]								Nnnn, nnnn
CLKDIV	Clock division Register	FE01H	CLKDIV[7:0]								Nnnn, nnnn

6.2.1 IRC Band Select Register (IRCBAND)

symbolic	address	B7	B6	B5	B4	B3	B2	B1	B0
IRCBAND	9DH	-	-	-	-	-	-	SEL[1:0]	

SEL[1:0]: band selection 00: selects the 6MHz band

01: Select 10MHz band

10: Select 27MHz band

11: Select 44MHz band

6.2.2 Internal IRC Frequency Adjustment Register (IRTRIM)

symbolic	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 adjusts the IRC frequency over 256 levels, and the frequency values adjusted at each level are linearly distributed overall, with local fluctuations. Macroscopically, the frequency adjusted at each level is about 0.24%, i.e., the frequency at IRTRIM (n+1) is about 0.24% faster than the frequency at IRTRIM (n). However, since the IRC frequency adjustment is not 0.24% at each level (the maximum value of the frequency adjusted at each level is about 0.55%, the minimum value is about 0.02%, and the overall average value is about 0.24%), it can cause local fluctuations.

6.2.3 Internal IRC Frequency Trim Register (LIRTRIM)

symbolic	address	B7	B6	B5	B4	B3	B2	B1	B0
LIRTRIM	9EH	-	-	-	-	-	-	-	LIRTRIM

LIRTRIM: Internal high precision IRC frequency trim register

6.2.4 Clock division register (CLKDIV)

symbolic	address	B7	B6	B5	B4	B3	B2	B1	B0
CLKDIV	FE01H								

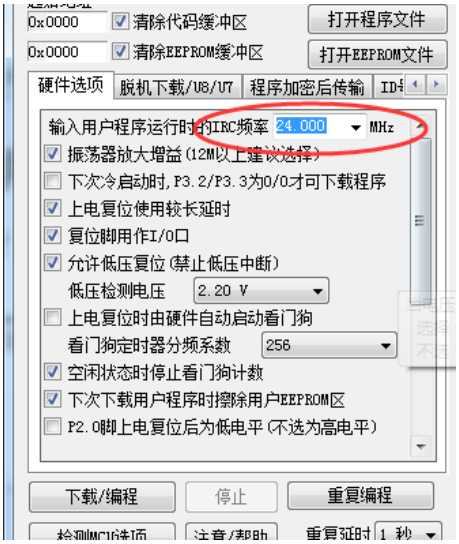
CLKDIV: Master clock dividing factor. The system clock SYSCLK is the clock signal after dividing the master clock MCLK.

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

6.2.5 Crossover out to 3MHz user operating frequency and user dynamically changing frequency chasing example

To obtain a frequency of 3 MHz, the method 24 MHz ÷ 8 can be used.

First select the internal IRC operating frequency of 24MHz when performing the ISP download, as shown in the following figure.



The clock source is then selected in the code to be the internal IRC and the CLKDIV register is used for 8 division.

C code

```
//Test operating frequency of 24MHz

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

#define    CLKSEL      (*(unsigned char volatile xdata *)0xfe00)
#define    CLKDIV      (*(unsigned char volatile xdata *)0xfe01)
#define    HIRCCR      (*(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;
    CLKSEL = 0x00;                  // Select internal IRC ( default )
    CLKDIV = 0x08; P_SW2 = 0x00;    /clock 8 division

    IRTRIM++;                       //IRC Frequency up 3‰ for fine tuning (note judging boundaries)
//  IRTRIM--;                       // IRC frequency down 3‰ for fine tuning (note judgment boundaries)

    while (1);
}
```

assembly code

```
; tested operating frequency of 24MHz

P_SW2       DATA    0BAH
IRTRIM      DATA    09FH

CLKSEL      EQU     0FE00H
CLKDIV      EQU     0FE01H
HIRCCR      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
```