```
    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
```

```
        MOV     P2M0, #00H
        MOV     P2M1, #00H
        MOV     P3M0, #00H
        MOV     P3M1, #00H
        MOV     P4M0, #00H
        MOV     P4M1, #00H
        MOV     P5M0, #00H
        MOV     P5M1, #00H

        MOV     P_SW2,#80H
        MOV     A,#00H          ; Select internal IRC
        MOV     DPTR,#CLKSEL
        MOVX    @DPTR,A
        MOV     A,#08H          ; Clock 8 division
        MOV     DPTR,#CLKDIV
        MOVX    @DPTR,A
        MOV     P_SW2,#00H

        INC     IRTRIM          ;IRC Frequency up 3‰ for fine tuning (watch for boundaries)
;       DEC     IRTRIM          ;IRC frequency trimmed down 3‰ (watch the boundaries)

        JMP     $

        END
```

6.3 System reset

The reset of the STC8A8K64D4 series microcontroller is divided into two types: hardware reset and software reset.

When the hardware is reset, the values of all registers are reset to their initial values, and the system re-reads all hardware options. Also the power-up wait is performed according to the power-up wait time set by the hardware options. A hardware reset consists mainly of.

- Power-On Reset, POR, 1.7V Nearby
- Low Voltage Reset, LVD-RESET (2.0V, 2.4V, 2.7V, 3.0V nearby)
- Reset pin reset **(low level reset)**
- Watchdog reset

During a software reset, the values of all registers are reset to their initial values, except for the registers associated with the clock, which remain unchanged, and a software reset does not re-read all hardware options. Software resets consist mainly of.

- Write the **reset-related register**

triggered by SWRST of IAP_CONTR

symbolic	description	address	Bit Addresses and Symbols								reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
WDT_CONTR	Watchdog control register	C1H	WDT_FLAG	-	EN_WDT	CLR_WDT	IDL_WDT	WDT_PS[2:0]			0x00,0000
IAP_CONTR	IAP control register	C7H	IAPEN	SWBS	SWRST	CMD_FAIL	-				0000,xxxx
RSTCFG	Reset Configuration Register	FFH	-	ENLVR	-	P54RST	-	-	LVDS[1:0]		0000,0000

6.3.1 Watchdog reset (WDT_CONTR)

In industrial control/automotive electronics/aerospace systems that require high reliability, a watchdog is usually introduced to prevent the system from operating abnormally for a long period of time due to interference with the MCU/CPU program in abnormal situations. The watchdog will force the MCU/CPU to reset and cause the system to start executing the user program from scratch again.

The STC8 series watchdog reset is one of the hardware resets in the hot-start reset, and the STC8 series microcontroller introduces this function to make the microcontroller system reliability design more convenient and concise.

SWBS of IAP_CONTR register before dog reset is irrelevant (**note: different here from STC15 series MCUs**)

WDT_CONTR (watchdog control register)

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

WDT_FLAG: Watchdog overflow flag

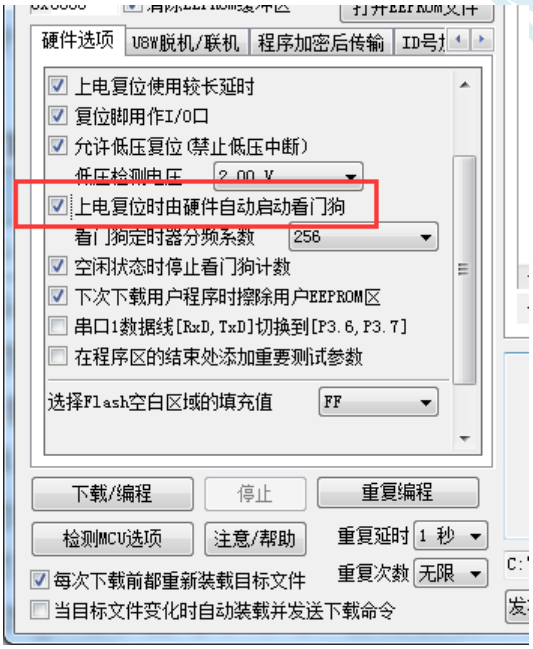
Hardware automatically sets this bit to 1 when a watchdog overflow occurs, requiring a software clear.

EN_WDT: Watchdog enable bit

0: No effect on the microcontroller

1: Start the watchdog timer.

Note: The watchdog timer can be started in software or automatically in hardware. Once the watchdog timer is started, it cannot be shut down in software and must be re-powered on the microcontroller before it can be shut down. Once the watchdog timer is started,



it cannot be turned off by software and must be powered up again. If the watchdog needs to be started in hardware, it needs to be set up during ISP download as shown in the following figure.

CLR_WDT: Watchdog timer clear 0: no effect on microcontroller

1: Clear the watchdog timer, hardware automatically resets this bit IDL_WDT: Watchdog control bit when in IDLE mode

0: Watchdog stops counting in IDLE mode

1: Watchdog continues counting during IDLE mode

WDT_PS[2:0]: watchdog timer clock division factor

WDT_PS[2:0]	crossover coefficient	Overflow time at 12M mains	Overflow time at 20M mains
000	2	≈ 65.5 ms	≈ 39.3 ms
001	4	≈ 131 ms	≈ 78.6 ms
010	8	≈ 262 ms	≈ 157 ms
011	16	≈ 524 ms	≈ 315 ms
100	32	≈ 1.05 seconds	≈ 629 ms
101	64	≈ 2.10 seconds	≈ 1.26 seconds
110	128	≈ 4.20 seconds	≈ 2.52 seconds
111	256	≈ 8.39 seconds	≈ 5.03 seconds

The watchdog overflow time is calculated by the following formula.

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

6.3.2 Software reset (IAP_CONTR)

IAP_CONTR (IAP Control Register) Write 60H to the IAP control register to achieve the effect of cold start for the microcontroller

symbolic	address	B7	B6	B5	B4	B3	B2	B1	B0
IAP_CONTR	C7H	IAPEN	SWBS	SWRST	CMD_FAIL	-			

SWBS: Software Reset Start Selection

0: Code execution starts from the user program area after a software reset. The data in the user data area remains unchanged.

1: Code execution starts from the system ISP area after a software reset. The data in the user data area is initialized.

SWRST: Software reset

trigger bit 0: No effect on the microcontroller

1: Trigger software reset

6.3.3 Low Pressure Reset (RSTCFG)

RSTCFG (Reset Configuration Register)

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

ENLVR: Low voltage reset control bit

0: Low voltage reset is disabled. When the system detects a low voltage event, a low voltage interrupt is generated

1: Enables low voltage reset. Automatically resets

when the system detects a low-voltage event P54RST:

RST pin function selection

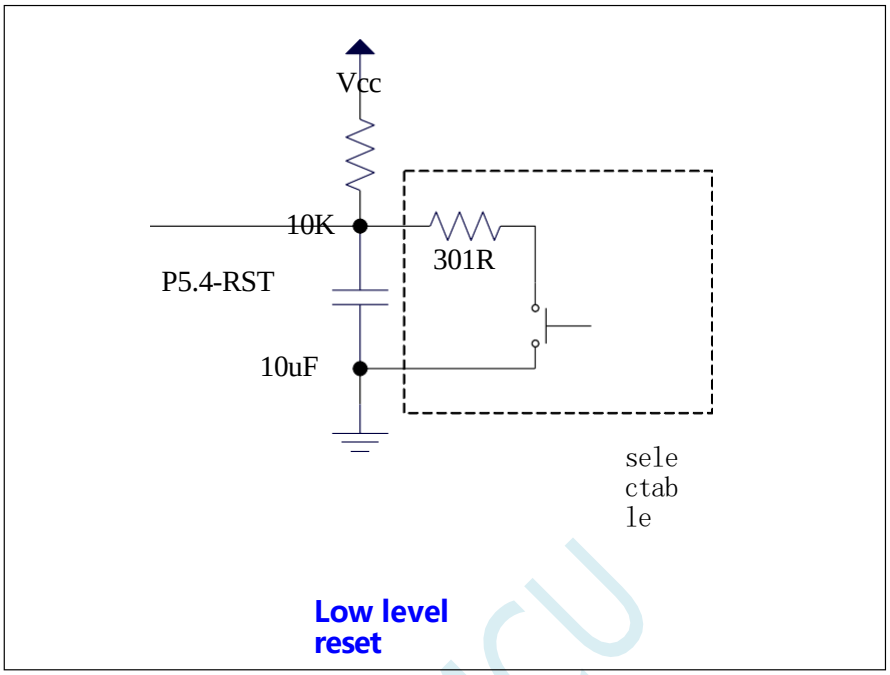
0: RST pin used as normal I/O port (P5.4)

1: The RST pin is used as a reset pin **(low reset)**

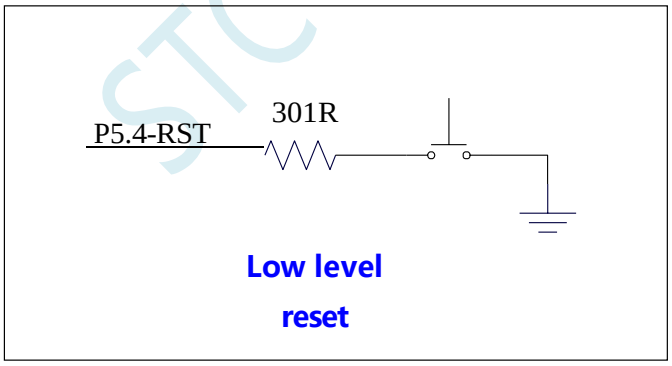
LVDS[1:0]: low voltage detection threshold voltage setting

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

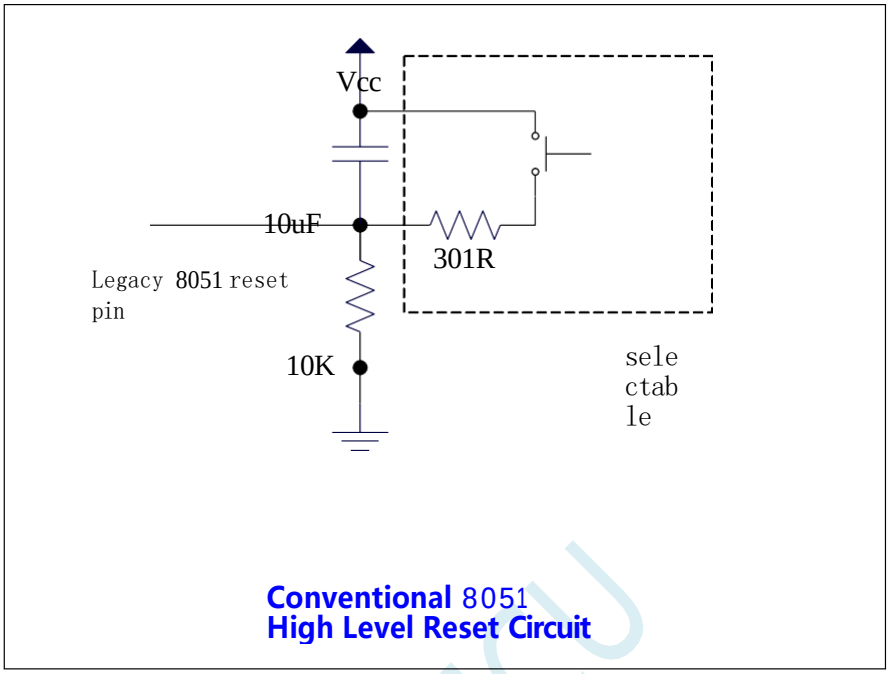
6.3.4 Low-level power-on reset reference circuit (not normally required)



6.3.5 Low Level Key Manual Reset Reference Circuit



6.3.6 Legacy 8051 High-Level Power-Up Reset Reference Circuit

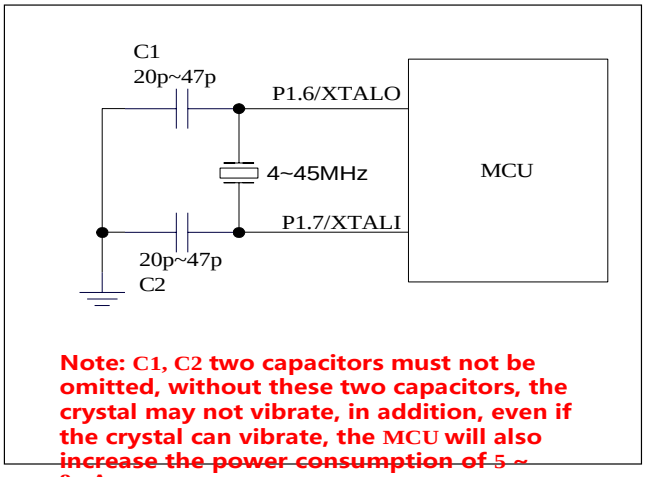


Conventional 8051
High Level Reset Circuit

The above figure shows the high reset circuit of traditional 8051, the reset of STC8A8K64D4 is low reset, different from the traditional reset circuit

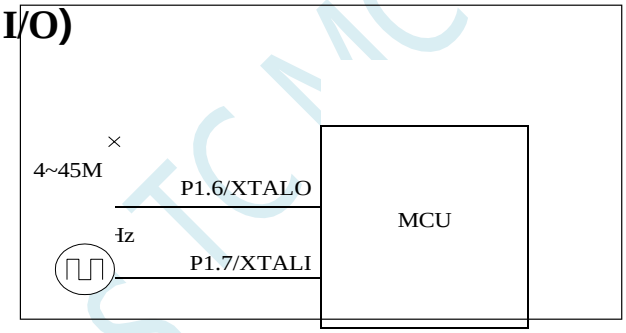
6.4 External Crystal and External Clock Circuit

6.4.1 External Crystal Input Circuit



Note: C1, C2 two capacitors must not be omitted, without these two capacitors, the crystal may not vibrate, in addition, even if the crystal can vibrate, the MCU will also increase the power consumption of 5 ~ 8mA

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



6.5 Clock stop/power saving mode and system power management

symbolic	description	address	Bit Addresses and Symbols								reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
PCON	Power Control Register	87H	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000

6.5.1 Power Control Register (PCON)

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

LVDF: Low voltage detection flag bit. When the system detects a low voltage event, the hardware automatically sets this bit to 1 and requests an interrupt to the CPU. This bit needs to be cleared by user software.

POF: Power-on reset flag bit. hardware automatically sets this bit to 1 each time the MCU is re-powered. software can clear this bit. PD: Clock stop mode/power down mode/power down mode control bit

0: No effect

1: The microcontroller enters clock stop mode/power down mode/power out mode, the CPU and all peripherals stop working. The hardware will be cleared automatically after waking up. (**Note: In clock stop mode, the CPU and all peripherals stop working, but the data in SRAM and XRAM are kept unchanged**)

IDL: IDLE (idle) mode control

bit 0: no effect

1: The microcontroller enters IDLE mode, only the CPU stops working, other peripherals are still running. The hardware is automatically cleared after waking up

Note: Although both LVD and comparator can wake up the clock stop mode, but the clock stop power saving mode, it is not recommended to start the LVD and comparator, otherwise the hardware system will automatically start the internal 1.19V high precision reference source, this high precision reference source has the corresponding anti-temperature drift and adjustment line, about 300uA additional power consumption, while the MCU into the clock stop mode, 3.3V operating voltage only about 0.4uA current consumption, so It is not recommended to turn on the LVD and comparator when entering the clock stop mode. If you do need to use it, it is recommended to turn on the power-down wake-up timer, which will only increase the power consumption by about 1.4uA, which is generally acceptable to the system. Let the power-down wake-up timer wake up the MCU every 5 seconds, and use LVD, comparator and ADC to detect the external battery voltage after waking up, and then enter the clock stop/power saving mode after the detection work takes about 1mS.

6.6 Power-down wake-up timer

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

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

symbolic	address	B7	B6	B5	B4	B3	B2	B1	B0
WKTCL	AAH								
WKTCH	ABH	WKTEN							

WKTEN: Enable control bit of the power-down wake-up timer 0: Disable the power-down wake-up timer

1: Enable power-down wake-up timer

If the built-in power-down wake-up dedicated timer of STC8 series microcontroller is allowed (WKTEN in WKTCH register is set to position 1 by software), when the MCU enters the power-down mode/shutdown mode, the power-down wake-up dedicated timer starts counting and wakes up the MCU when the count value is equal to the value set by the user. After the MCU wakes up, the program starts from the next statement of the last statement that set the MCU into the power-down mode After the MCU wakes up, the program is executed from the next statement of the last statement that set the MCU to power-down mode. After the power-down wake-up, you can read the contents of WKTCH and WKTCL to get the sleep time of the MCU in the power-down mode.

Note here that the user must write 1 less than the actual count value in registers {WKTCH[6:0],WKTCL[7:0]}. e.g. if the user needs to count 10 times, then write 9 to register {WKTCH[6:0],WKTCL[7:0]}. Similarly, if the user needs to count 32767 times, then {WKTCH[6:0],WKTCL[7:0]} should be written to 7FFEH (i.e. 32766)(Count **value 0 and count value 32767 are internal reserved values and cannot be used by the user**) The internal power-down wake-up timer has its own internal clock, and the time for the power-down wake-up timer to count once is determined by this clock. The clock frequency of the internal power-down wake-up timer is about 32KHz with a large error. The user can read the contents of RAM area F8H and F9H (F8H stores the high byte of frequency, F9H stores the low byte) to get the clock frequency recorded by the internal power-down wake-up dedicated timer at the factory.

The formula for calculating the power-down wake-up dedicated timer count time is shown below:(F_{wt} is the clock frequency of the internal power-down wake-up dedicated timer that we get from RAM area F8H and F9H)

$$\frac{16 \times \text{number of counts}}{F_{wt}}$$

$$\text{Power-down wake-up timer timing time} = \frac{\text{Fwt} \times (\text{seconds})}{\text{Fwt}}$$

Assuming F_{wt} = 32 KHz, we have.

{WKTCH[6:0],WKTCL[7:0]}	Power-down wake-up dedicated timer count time
~~0 (internal reservations)~~	
1	$10^6 \div 32K \times 16 \times (1+1) \approx 1$ ms
9	$10^6 \div 32K \times 16 \times (1+9) \approx 5$ ms
99	$10^6 \div 32K \times 16 \times (1+99) \approx 50$ ms
999	$10^6 \div 32K \times 16 \times (1+999) \approx 0.5$ sec
4095	$10^6 \div 32K \times 16 \times (1+4095) \approx 2$ seconds
32766	$10^6 \div 32K \times 16 \times (1+32766) \approx 16$ seconds
~~32767 (internal reservation)~~	

6.7 paradigm procedure

6.7.1 Selecting the system clock source

C code

```
// Tested operating frequency of 11.0592MHz

#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       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 )
    P_SW2 = 0x00;
```

```
    P_SW2 = 0x80;
    XOSCCR = 0xc0;                  //Start external crystal
    while (! (XOSCCR & 1));         // wait for the clock to stabilize
    CLKDIV = 0x00;                  //Clock is not frequency divided
```

```
    CLKSEL = 0x01;                          //Select external crystal
    P_SW2 = 0x00;
*/

/*
    P_SW2 = 0x80;
    IRC32KCR = 0x80;                        //start internal 32K IRC
    while (! (IRC32KCR & 1));               // wait for the clock to stabilize
    CLKDIV = 0x00;                          //Clock is not frequency divided
    CLKSEL = 0x03;                          //Select internal 32K
    P_SW2 = 0x00;
*/
    while (1);
}
```

assembly code

Operating frequency tested at 11.0592MHz

```
P_SW2       DATA    0BAH

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
            MOV     P2M0, #00H
            MOV     P2M1, #00H
            MOV     P3M0, #00H
            MOV     P3M1, #00H
            MOV     P4M0, #00H
            MOV     P4M1, #00H
            MOV     P5M0, #00H
            MOV     P5M1, #00H
```

```
                MOV     P_SW2,#80H
                MOV     A,#00H                  Select Internal IRC (default)
                MOV     DPTR,#CLKSEL
                MOVX    @DPTR,A
                MOV     P_SW2,#00H

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

;               MOV     P_SW2,#80H
;               MOV     A,#80H                  ; Start internal 32K IRC
;               MOV     DPTR,#IRC32KCR
;               MOVX    @DPTR,A
;               MOVX    A,@DPTR
;               JNB     ACC.0,$-1               Wait for the clock to stabilize
;               CLR     A                       Clock not crossover
;               MOV     DPTR,#CLKDIV
;               MOVX    @DPTR,A
;               MOV     A,#03H                  ; select internal 32K
;               MOV     DPTR,#CLKSEL
;               MOVX    @DPTR,A
;               MOV     P_SW2,#00H

                JMP     $

                END
```

6.7.2 Master clock divider output

C code

```
// Tested operating frequency of 11.0592MHz

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

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

sfr   P_SW2   =   0xba;

sfr   P0M1    =   0x93;
sfr   P0M0    =   0x94;
sfr   P1M1    =   0x91;
sfr   P1M0    =   0x92;
sfr   P2M1    =   0x95;
```

```
sfr     P2M0    =   0x96;
sfr     P3M1    =   0xb1;
sfr     P3M0    =   0xb2;
sfr     P4M1    =   0xb3;
sfr     P4M0    =   0xb4;
sfr     P5M1    =   0xc9;
sfr     P5M0    =   0xca;

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

    P_SW2 = 0x80;
//  MCLKOCR = 0x01;     // Master clock output to P5.4 port
//  MCLKOCR = 0x02;     //Master clock 2 division output to P5.4 port
    MCLKOCR = 0x04;     // Master clock 4-way output to P5.4 port
//  MCLKOCR = 0x84;     // Master clock 4-division output to P1.6 port
    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
```

```
DEMO:   DATA    0CAH

        ORG     0000H
        LJMP    MAIN

        ORG     0100H
MAIN:
```

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

        MOV     P_SW2,#80H
;       MOV     A,#01H          Master clock output to P5.4 port
;       MOV     A,#02H          Master clock 2-division output to P5.4 port
        MOV     A,#04H          Master clock 4-division output to P5.4 port
;       MOV     A,#84H          Master clock 4-division output to P1.6 port
        MOV     DPTR,#MCLKOCR
        MOVX    @DPTR,A
        MOV     P_SW2,#00H

        JMP     $

        END
```

6.7.3 Watchdog Timer Applications

C code

```
// Tested operating frequency of 11.0592MHz

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

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

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

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

```
    P1M0 = 0x00; P1M1 = 0x00;
    P2M0 = 0x00; P2M1 = 0x00;
    P3M0 = 0x00; P3M1 = 0x00;
    P4M0 = 0x00; P4M1 = 0x00;
    P5M0 = 0x00; P5M1 = 0x00;

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

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

        Display();          // Display module
        Scankey();          /Key Scan Module
        MotorDriver();      /Motor Drive Module
    }
}
```

assembly code

Operating frequency tested at 11.0592MHz

```
WDT_CONTR   DATA    0C1H

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

            ORG     0000H
            LJMP    MAIN

            ORG     0100H
MAIN:
            MOV     SP,#5FH
            MOV     P0M0,#00H
```

```
MOV     P0M1, #00H
MOV     P0M0, #00H
MOV     P1M0, #00H
MOV     P1M1, #00H
MOV     P2M0, #00H
MOV     P2M1, #00H
```

```
        MOV     P3M0, #00H
        MOV     P3M1, #00H
        MOV     P4M0, #00H
        MOV     P4M1, #00H
        MOV     P5M0, #00H
        MOV     P5M1, #00H

;       MOV     WDT_CONTR,#23H      Enable watchdog, overflow time is about 0.5s
        MOV     WDT_CONTR,#24H      Enable watchdog, overflow time is about 1s
;       MOV     WDT_CONTR,#27H      Enable watchdog, overflow time is about 8s
        CLR     P3.2                ; Test port
LOOP:
;       MOV     WDT_CONTR,#33H      Clear the watchdog, otherwise the system resets.
        MOV     WDT_CONTR,#34H      Clear the watchdog, otherwise the system resets.
;       MOV     WDT_CONTR,#37H      Clear the watchdog, otherwise the system resets.

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

        END
```

6.7.4 Soft reset for custom downloads

C code

```
// Tested operating frequency of 11.0592MHz

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

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

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

void main()
{
    P0M0 = 0x00; P0M1 = 0x00;
    P1M0 = 0x00; P1M1 = 0x00;
```

```
P3M1 = 0x00;
```

```
    P2M1 = 0x00;   P3M0 = 0x00;
    P3M1 = 0x00;   P4M0 = 0x00;
    P4M1 = 0x00;   P5M0 = 0x00;
    P5M1 = 0x00;

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

    while (1)
    {
        if (!P32 && !P33)
        {
            IAP_CONTR |= 0x60;                  //check to reset to ISP when P3.2 and P3.3 are both 0
        }
    }
}
```

assembly code

Operating frequency tested at 11.0592MHz

```
IAP_CONTR   DATA    0C7H

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

            ORG     0000H
            LJMP    MAIN

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

```
MOV     P3M1,#00H
SETB    P3.2
```

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

        END
```

6.7.5 Low voltage detection

C code

```
// Tested operating frequency of 11.0592MHz

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

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

sfr     p0m1    =   0x93;
sfr     p0m0    =   0x94;
sfr     p1m1    =   0x91;
sfr     p1m0    =   0x92;
sfr     p2m1    =   0x95;
sfr     p2m0    =   0x96;
sfr     p3m1    =   0xb1;
sfr     p3m0    =   0xb2;
sfr     p4m1    =   0xb3;
sfr     p4m0    =   0xb4;
sfr     p5m1    =   0xc9;
sfr     p5m0    =   0xca;
sfr

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

void main()
{
    P0M0    =
    0x00;   P0M1
    =       0x00;
    P1M0    =
    0x00;   P1M1
    =       0x00;
    P2M0    =
```

```
    P0M1 = 0x00;
    P0M0 = 0x00;
    P1M1 = 0x00;
    P1M0 = 0x00;
    P2M1 = 0x00;
    P2M0 = 0x00;
    P3M1 = 0x00;
    P3M0 = 0x00;
```

```
    P3M1 = 0x00; P4M0 = 0x00;
    P4M1 = 0x00; P5M0 = 0x00;
    P5M1 = 0x00;

    PCON &= ~LVDF;                  // Test port
//  RSTCFG = ENLVR | LVD3V0;        // Enables low voltage reset at 3.0V, no LVD interrupt is generated
    RSTCFG = LVD3V0;
    ELVD = 1;                       // Enables low voltage interrupt at 3.0V
    EA = 1;                         // Enables LVD interrupt

    while (1);
}
```

assembly code

Operating frequency tested at 11.0592MHz

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

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

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

        ORG   0000H
        LJMP  MAIN
        ORG   0033H
        LJMP  LVDISR

        ORG   0100H
LVDISR:
        ANL   PCON,#NOT LVDF    ; Clear the interrupt flag
        CPL   P3.2              ; Test port
        RETI
MAIN:
        MOV   SP, #5FH
        MOV   P0M0, #00H
        MOV   P0M1, #00H
```

```
MOV     P1M0, #00H
MOV     P1M1, #00H
MOV     P2M0, #00H
```

```
        MOV     P2M1, #00H
        MOV     P3M0, #00H
        MOV     P3M1, #00H
        MOV     P4M0, #00H
        MOV     P4M1, #00H
        MOV     P5M0, #00H
        MOV     P5M1, #00H

        ANL     PCON,#NOT LVDF      The LVDF logo needs to be cleared after power-up.
;       MOV     RSTCFG,#ENLVR | LVD3V0  Low voltage reset on 3.0V enable, no LVD interrupt
        MOV     RSTCFG,#LVD3V0      Enable low voltage interrupt at 3.0V
        SETB    ELVD                Enable LVD interrupt
        SETB    EA
        JMP     $

        END
```

6.7.6 power saving mode

C code

```
// Tested operating frequency of 11.0592MHz

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

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

sfr         p0m1        =   0x93;
sfr         p0m0        =   0x94;
sfr         p1m1        =   0x91;
sfr         p1m0        =   0x92;
sfr         p2m1        =   0x95;
sfr         p2m0        =   0x96;
sfr         p3m1        =   0xb1;
sfr         p3m0        =   0xb2;
sfr         p4m1        =   0xb3;
sfr         p4m0        =   0xb4;
sfr         p5m1        =   0xc9;
sfr         p5m0        =   0xca;
sfr

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

void main()
{
    P0M0        =
    0x00;   P0M1
    =       0x00;
    P1M0        =
```

```
    P2M1 = 0x00;
    P2M0 = 0x00;
```

```
    P2M1 = 0x00; P3M0 = 0x00;
    P3M1 = 0x00; P4M0 = 0x00;
    P4M1 = 0x00; P5M0 = 0x00;
    P5M1 = 0x00;

    EX0 = 1;                    //Enable INT0 interrupt to wake up MCU
    EA = 1;
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    PCON = IDL;                 //MCU enters IDLE mode
//  PCON = PD;                  //MCU enters power-down mode
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    P35 = 0;

    while (1);
}
```

assembly code

Operating frequency tested at 11.0592MHz

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

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

        ORG     0000H
        LJMP    MAIN
        ORG     0003H
        LJMP    INT0ISR

        ORG     0100H
INT0ISR:
        CPL     P3.4        ; Test port
        RETI
MAIN:
        MOV     SP,#5FH
```

```
MOV     P0M0, #00H
MOV     P0M1, #00H
MOV     P1M0, #00H
```

```
        MOV     P1M1, #00H
        MOV     P2M0, #00H
        MOV     P2M1, #00H
        MOV     P3M0, #00H
        MOV     P3M1, #00H
        MOV     P4M0, #00H
        MOV     P4M1, #00H
        MOV     P5M0, #00H
        MOV     P5M1, #00H

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

        END
```

6.7.7 Wake-up power saving mode using INT0/INT1/INT2/INT3/INT4 pin interrupts

C code

```
// Tested operating frequency of 11.0592MHz

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

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

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

sfr     P0M1    =   0x93;
sfr     P0M0    =   0x94;
sfr     P1M1    =   0x91;
sfr     P1M0    =   0x92;
sfr     P2M1    =   0x95;
sfr     P2M0    =   0x96;
sfr     P3M1    =   0xb1;
sfr     P3M0    =   0xb2;
sfr     P4M1    =   0xb3;
```

```
sfr  P4M0  =  0xb4;
sfr  P5M1  =  0xc9;
sfr  P5M0  =  0xca;
```

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

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

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

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

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

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

    IT0 = 0;                                    //enables INT0 rising and falling edge interrupts
//  IT0 = 1;                                    // Enables INT0 falling edge interrupt
    EX0 = 1;                                    //enable INT0 interrupt

    IT1 = 0;                                    //enables INT1 rising and falling edge interrupts
//  IT1 = 1;                                    // Enables INT1 falling edge interrupt
    EX1 = 1;                                    //enable INT1 interrupt

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

    EA = 1;
```

```
    PCON = 0x02;                    //MCU enters power-down mode
    _nop_();                        //The MCU will first execute this statement when it is
    woken up in power-down mode.

                                    // Then enter the interrupt service program again
    _nop_();
```

```
    _nop_();
    _nop_();

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

assembly code

Operating frequency tested at *11.0592MHz*

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

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

            ORG     0000H
            LJMP    MAIN

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

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

```
            CPL     P1.0                ; Test port
            RETI

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

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

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

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

            SETB    EA

            MOV     PCON,#02H           ;MCU enters power-down mode
            NOP                         After waking up in power-down mode, the MCU
                                        will first execute this statement
            NOP                         Then enter the interrupt service program.
            NOP
            NOP
LOOP:
            CPL     P1.1
            JMP     LOOP

            END
```

6.7.8 Wake up MCU power saving mode using T0/T1/T2/T3/T4 pin interrupts

C code

```
// Tested operating frequency of 11.0592MHz

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

sfr         T2L         =   0xd7;
```

```
sfr T2H = 0xd6;
sfr T3L = 0xd5;
```

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

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

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

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

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

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

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

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

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

```
    P1M0 = 0x00;   P1M1 = 0x00;
    P2M0 = 0x00;   P2M1 = 0x00;
    P3M0 = 0x00;   P3M1 = 0x00;
    P4M0 = 0x00;   P4M1 = 0x00;
    P5M0 = 0x00;   P5M1 = 0x00;

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

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

    T2L = 0x66;                     //65536-11.0592M/12/1000
    T2H = 0xfc;
    AUXR = 0x10;                    //Start timer
    IE2 = ET2;                      //enable timer interrupt

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

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

    EA = 1;

    PCON = 0x02;                    //MCU enters power-down mode
    _nop_();                        // The interrupt service program will not be entered
    immediately after power-down wake-up,
                                    // Instead, it waits until the timer overflows before
                                    entering the interrupt service program
    _nop_();
    _nop_();
    _nop_();

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

Operating frequency tested at *11.0592MHz*

```
T2L     DATA    0D7H
T2H     DATA    0D6H
T3L     DATA    0D5H
```

```
T3H         DATA    0D4H
T4L         DATA    0D3H
T4H         DATA    0D2H
T4T3M       DATA    0D1H
AUXR        DATA    8EH

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

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

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

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

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

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

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

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

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

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

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

        SETB    EA

        MOV     PCON,#02H               ;MCU enters power-down mode
        NOP                             After the external pin of ;T0/T1/T2/T3/T4 wakes up, the
                                        ; does not enter the interrupt service program, but
        NOP                             simply continues to execute the program.
                                        ; Not the same as power-down wake-up for
                                        INT0/INT1/INT2/INT3/INT4
                                        It is recommended to add more NOP commands,
                                        e.g. 3 or more
        NOP
        NOP
LOOP:
        CPL     P1.1
        JMP     LOOP

        END
```

6.7.9 Wake up MCU power saving mode using RxD/RxD2/RxD3/RxD4 pin interrupts

C code

```
// Tested operating frequency of 11.0592MHz

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

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

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

sbit    P11     =   P1^1;

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

void UART1_Isr() interrupt 4
{
}

void UART2_Isr() interrupt 8
{
}

void UART3_Isr() interrupt 17
{
}

void UART4_Isr() interrupt 18
{
}

void main()
{
    P0M0 = 0x00; P0M1 = 0x00;
    P1M0 = 0x00; P1M1 = 0x00;
```

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

```
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

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

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

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

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

    ES = 1;                         // Enables serial interrupts
    IE2 = ES2;                      // Enables serial interrupts
    IE2 |= ES3; IE2 |= ES4; EA = 1; // Enables serial interrupts
                                    // Enables serial interrupts

    PCON = 0x02;                    // MCU enters power-down mode
    _nop_();                        // No interrupt service program will be entered after power-down wake-up,
    _nop_();
    _nop_();
    _nop_();

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

assembly code

Operating frequency tested at *11.0592MHz*

```
TE2         DATA    0AEH
ES2         EQU     01H
ES3         EQU     08H
ES4         EQU     10H

P_SW1       DATA    0A2H
P_SW2       DATA    0BAH

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

```
P5M1    DATA    0C9H
P5M0    DATA    0CAH

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

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

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

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

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

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

        MOV     P_SW2,#00H      ;RXD4/P0.2 Wake-up on falling edge
```

```
;               MOV     P_SW3,#04H          ;RXD4_2/P5.2 Wake-up on falling edge

                SETB    ES                  ; Enable serial interrupt
                MOV     IE2,#ES2            ; Enable serial interrupt
                ORL     IE2,#ES3            ; Enable serial interrupt
                ORL     IE2,#ES4            ; Enable serial interrupt
                SETB    EA
```

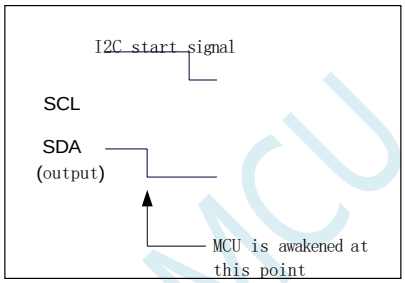
```
                NOP                                 ;RxD/RxD2/RxD3/RxD4 and corresponding switchable
                sets of pins down

                                                    After waking up along, instead of entering the
                                                    interrupt service program, the program simply
                                                    continues to execute.
                                                    ; Not the same as power-down wake-up for
                                                    INT0/INT1/INT2/INT3/INT4
                NOP                                 ; it is recommended to add more NOP commands,
                e.g. 3 or more
LOOP:           MOV     PCON,#02H                   ;MCU enters power-down mode
                NOP                                 ; No interrupt service program will be entered after
                power-down wake-up,
                NOP
                NOP
                NOP

                CPL     P1.1
                JMP     LOOP

                END
```

6.7.10 Wake up MCU power saving mode using SDA pin of I2C



C code

```
// Tested operating frequency of 11.0592MHz

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

sfr         P_SW2       =   0xba;

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

sbit        P11         =   P1^1;

sfr         P0M1        =   0x93;
sfr         P0M0        =   0x94;
sfr         P1M1        =   0x91;
sfr         P1M0        =   0x92;
sfr         P2M1        =   0x95;
sfr         P2M0        =   0x96;
sfr         P3M1        =   0xb1;
```

```
sfr     P3M0    =   0xb2;
sfr     P4M1    =   0xb3;
sfr     P4M0    =   0xb4;
sfr     P5M1    =   0xc9;
sfr     P5M0    =   0xca;

void i2c_isr() interrupt 24
```

```
{
    P_SW2 |= 0x80;
    I2CSLST &= ~0x40;
}

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

    P_SW2 = 0x00;                   //SDA/P1.4 Wake-up on falling edge
//  P_SW2 = 0x10;                   //SDA_2/P2.4 Wake-up on falling edge
//  P_SW2 = 0x30;                   //SDA_4/P3.3 Wake-up on falling edge
    P_SW2 |= 0x80;
    I2CCFG = 0x80;                  //Slave mode of the /enable I2C module

    I2CSLCR = 0x40; EA = 1;         //Enables start signal interrupt

    PCON = 0x02;                    //MCU enters power-down mode
    _nop_();
    _nop_();
    _nop_();
    _nop_();

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

assembly code

Operating frequency tested at 11.0592MHz

```
P_SW2       DATA    0BAH

I2CCFG      XDATA   0FE80H
I2CSLCR     XDATA   0FE83H
```

I2CSLST	*XDATA*	*0FE84H*
P0M1	*DATA*	*093H*
P0M0	*DATA*	*094H*
P1M1	*DATA*	*091H*
P1M0	*DATA*	*092H*
P2M1	*DATA*	*095H*
P2M0	*DATA*	*096H*
P3M1	*DATA*	*0B1H*
P3M0	*DATA*	*0B2H*
P4M1	*DATA*	*0B3H*

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

            ORG     0000H
            LJMP    MAIN
            ORG     00C3H
            LJMP    I2CISR

            ORG     0100H
I2CISR:
            PUSH    ACC
            PUSH    DPH
            PUSH    DPL
            ORL     PSW2,#80H
            MOV     DPTR,#I2CSLST
            MOVX    A,@DPTR
            ANL     A,#NOT 40H
            MOVX    @DPTR,A
            POP     DPL
            POP     DPH
            POP     ACC
            RETI

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

            MOV     P_SW2,#00H      ;SDA/P1.4 Wake-up on falling edge
//          MOV     P_SW2,#10H      ;SDA_2/P2.4 Wake-up on falling
                                    edge
//          MOV     P_SW2,#30H      ;SDA_4/P3.3 Wake-up on falling
                                    edge
            ORL     P_SW2,#80H
            MOV     DPTR,#I2CCFG
            MOV     A,#80H
            MOVX    @DPTR,A         Enables slave mode of the I2C
                                    module
            MOV     DPTR,# I2CSLCR
            MOV     A,#40H          ;Enable start signal interrupt
            SETB    EA

            MOV     PCON,#02H       ;MCU enters power-down mode
            NOP                     No interrupt service program
            NOP                     will be entered after power-
            NOP                     down wake-up,
            NOP
LOOP:       NOP
            P1.1
```

JMP *LOOP*

END

6.7.11 Wake-up power saving mode using the power-down wake-up timer

C code

```
// Tested operating frequency of 11.0592MHz

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

sfr     WKTCL     =   0xaa;
sfr     WKTCH     =   0xab;;

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

sbit    P11       =   P1^1;

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

    WKTCL = 0xff;                   // Set the power-down wake-up clock to approximately 1 second
    WKTCH = 0x87;
```

```
void main(void)
{
    _nop_();
    _nop_();
    PCON = 0x02;                    //MCU enters power-down mode
    _nop_();
    _nop_();
    _nop_();
    _nop_();
```

```
        P11 = ~P11;
    }
}
```

assembly code

Operating frequency tested at *11.0592MHz*

```
WKTCL   DATA    0AAH
WKTCH   DATA    0ABH

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

        ORG     0000H
        LJMP    MAIN

        ORG     0100H

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

        MOV     WKTCL,#0FFH     Set the power-down wake-up clock to approximately 1 second
        MOV     WKTCH,#87H

LOOP:
        NOP
        NOP
        MOV     PCON,#02H       ;MCU enters power-down mode
        NOP
        NOP
        NOP
        NOP
        NOP     P1.1
        CPL
        JMP     LOOP
```

END

6.7.12 LVD interrupt wake-up power saving mode, recommended to use with power-down wake-up timer

In clock stop power saving mode, it is not recommended to start the LVD and comparator, otherwise the hardware system will automatically start the internal 1.19V high precision reference source, this high precision reference source has the corresponding anti-temperature drift and tuning line, which will increase the power consumption of about 300uA, while the MCU into the clock stop mode, the 3.3V operating voltage only consumes about 0.4uA current, so it is not recommended to enter the clock stop mode to turn on the LVD and comparator. If you do need to use it, it is recommended to turn on the power-down wake-up timer, which will only increase the power consumption by about 1.4uA, which is generally acceptable to the system. Let the power-down wake-up timer wake up the MCU every 5 seconds, and use LVD, comparator and ADC to detect the external battery voltage after waking up, and then enter the clock stop/power saving mode after the detection work takes about 1mS.

C code

```
// Tested operating frequency of 11.0592MHz

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

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

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

sfr     p0m1        =   0x93;
sfr     p0m0        =   0x94;
sfr     p1m1        =   0x91;
sfr     p1m0        =   0x92;
sfr     p2m1        =   0x95;
sfr     p2m0        =   0x96;
sfr     p3m1        =   0xb1;
sfr     p3m0        =   0xb2;
sfr     p4m1        =   0xb3;
sfr     p4m0        =   0xb4;
sfr     p5m1        =   0xc9;
sfr     p5m0        =   0xca;
sfr

void LVD_Isr() interrupt 6
{
```

```
        PCON &= ~LVDF;
        P10 = !P10;
}

void main()
{
        P0M0 = 0x00;  P0M1 = 0x00;
        P1M0 = 0x00;  P1M1 = 0x00;
```

// Clear the interrupt flag
//test port

```
    P2M0 = 0x00; P2M1 = 0x00;
    P3M0 = 0x00; P3M1 = 0x00;
    P4M0 = 0x00; P4M1 = 0x00;
    P5M0 = 0x00; P5M1 = 0x00;

    PCON &= ~LVDF;                  // Power up requires clearing the interrupt flag
    RSTCFG = LVD3V0;                // Set LVD voltage to 3.0V
    ELVD = 1;                       //enable LVD interrupt
    EA = 1;

    PCON = 0x02;                    //MCU enters power-down mode
    _nop_();                        //Wake up immediately after power down and enter
    the interrupt service program
    _nop_();
    _nop_();
    _nop_();

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

assembly code

Operating frequency tested at 11.0592MHz

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

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

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

        ORG   0000H
        LJMP  MAIN
```

```
        ORG     0000H
        LJMP    LVDISR

        ORG     0100H
```

```
LVDISR:
        ANL     PCON,#NOT LVDF      ; Clear the interrupt flag
        CPL     P1.0                ; Test port
        RETI

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

        ANL     PCON,#NOT LVDF      Power-up requires clearing the interrupt flag
        MOV     RSTCFG,#LVD3V0      Set LVD voltage to 3.0V
        SETB    ELVD                Enable LVD interrupt
        SETB    EA

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

        END
```

6.7.13 Comparator interrupt wake-up power saving mode, recommended to use with power-down wake-up timer

In clock stop power saving mode, it is not recommended to start the LVD and comparator, otherwise the hardware system will automatically start the internal 1.19V high precision reference source, this high precision reference source has the corresponding anti-temperature drift and tuning line, which will increase the power consumption of about 300uA, while the MCU into the clock stop mode, the 3.3V operating voltage only consumes about 0.4uA current, so it is not recommended to enter the clock stop mode to turn on the LVD and comparator. If you do need to use it, it is recommended to turn on the power-down wake-up timer, which will only increase the power consumption by about 1.4uA, which is generally acceptable to the system. Let the power-down wake-up timer wake up the MCU every 5 seconds, and use LVD, comparator and ADC to detect the external battery voltage after waking up, and then enter the clock stop/power saving mode after the detection work takes about 1mS.

C code

```
// Tested operating frequency of 11.0592MHz

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

sfr     CMPCR1    =    0xe6;
sfr     CMPCR2    =    0xe7;
```

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

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

void CMP_Isr() interrupt 21
{
    CMPCR1 &= ~0x40;                    // Clear the interrupt flag
    P10 = !P10;                         //test port
}

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

    CMPCR2 = 0x00;
    CMPCR1 = 0x80;                      //Enable comparator module
    CMPCR1 |= 0x30;                     //enable comparator edge interrupt
    CMPCR1 &= ~0x08;                    //P3.6 is the CMP+ input pin
    CMPCR1 |= 0x04;                     //P3.7 is the CMP-input pin
    CMPCR1 |= 0x02;                     //enable comparator output
    EA = 1;

    PCON = 0x02;                        //MCU enters power-down mode
    _nop_();                            //Wake up immediately after power down and enter
    the interrupt service program
    _nop_();
    _nop_();
    _nop_();

    while (1)
```

```
    {
        P11 = ~P11;
    }
}
```

Operating frequency tested at *11.0592MHz*

```
CMPCR1      DATA    0E6H
CMPCR2      DATA    0E7H

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

            ORG     0000H
            LJMP    MAIN
            ORG     00ABH
            LJMP    CMPISR

            ORG     0100H
CMPISR:
            ANL     CMPCR1,#NOT 40H     ; Clear the interrupt flag
            CPL     P1.0                ; Test port
            RETI

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

            MOV     CMPCR2,#00H
            MOV     CMPCR1,#80H         Enabling comparator module
            ORL     CMPCR1,#30H         ; Enable comparator edge interrupt
            ANL     CMPCR1,#NOT 08H     ;P3.6 is the CMP+ input pin
            ORL     CMPCR1,#04H         ;P3.7 is the CMP-input pin
            ORL     CMPCR1,#02H         ; Enable comparator output
            SETB    EA

            MOV     PCON,#02H           ;MCU enters power-down mode
            NOP                         Enter interrupt service program
            NOP                         immediately after power-down
            NOP                         wake-up
            NOP
```

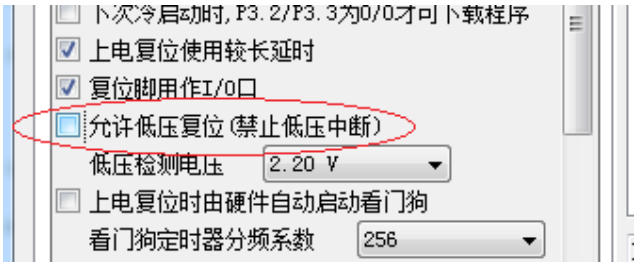
```
LOOP:
        CPL     P1.1
        JMP     LOOP

        END
```

6.7.14 Operating voltage detection (battery voltage) using the LVD function

If you need to use the LVD function to detect the battery voltage, you need to remove the low voltage reset function during the ISP download, as shown in the following figure "Allow low voltage reset (disable low voltage interrupt)" hardware option's checkbox needs to be removed

(It is recommended to use channel 15 of the ADC to detect the battery voltage, see the ADC section)



C code

```
// Tested operating frequency of 11.0592MHz

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

#define FOSC            11059200UL
#define T1MS            (65536 - FOSC/4/100)

sfr                     =
RSTCFG #define           0xff;           //LVD@2.0V
LVD2V0                  0x00
#define     LVD2V4      0x01            //LVD@2.4V
#define     LVD2V7      0x02            //LVD@2.7V
#define     LVD3V0      0x03            //LVD@3.0V

#define     LVDF        0x20            //PCON.5
```

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

void delay()
{
    int i;
```

```
    for (i=0; i<100; i++)
    {
        _nop_();
        _nop_();
        _nop_();
        _nop_();
    }
}

void main()
{
    unsigned char power;

    P0M0 =
    0x00; P0M1
    = 0x00;
    P1M0 =
    0x00; P1M1
    = 0x00;
    P2M0 =
    0x00; P2M1
    = 0x00;
    P3M0 =
    0x00; P3M1
    = 0x00;
    P4M0 =
    0x00; P4M1
    = 0x00;
    P5M0 =
    0x00; P5M1
    = 0x00;

    PCON &= ~LVDF;
    RSTCFG = LVD3V0;

    while (1)
    {
        power = 0x0f;

        RSTCFG = LVD3V0;
        delay();
        PCON &= ~LVDF;
        delay();
        if (PCON & LVDF)
        {
            power >>= 1;
            RSTCFG = LVD2V7;
            delay();
            PCON &= ~LVDF;
            delay();
            if (PCON & LVDF)
            {
                power >>= 1;
                RSTCFG = LVD2V4;
                delay();
                PCON &= ~LVDF;
                delay();
                if (PCON & LVDF)
                {
```

```
power >> 5;
RSTCFG = LVD2V2;
delay();
PCON &= ~LVDF;
delay();
```

```
                    if (PCON & LVDF)
                    {
                        power >>= 1;
                    }
                }
            }
        }
        RSTCFG = LVD3V0;
        P2 = ~power;      //P2.3~P2.0 Display battery level
    }
}
```

assembly code

Operating frequency tested at 11.0592MHz

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

LVDF        EQU     20H         ;PCON.5

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

            ORG     0000H
            JMP     MAIN

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

            ANL     PCON,#NOT LVDF
            MOV     RSTCFG,#LVD3V0
```

```
LOOP:
        MOV     B,#0FH

        MOV     RSTCFG,#LVD3V0
        CALL    DELAY
        ANL     PCON,#NOT LVDF
        CALL    DELAY
        MOV     A, PCON
        ANL     A,#LVDF
        JZ      SKIP
        MOV     A,B
        CLR     C
        RRC     A
        MOV     B,A

        MOV     RSTCFG,#LVD2V7
        CALL    DELAY
        ANL     PCON,#NOT LVDF
        CALL    DELAY
        MOV     A, PCON
        ANL     A,#LVDF
        JZ      SKIP
        MOV     A,B
        CLR     C
        RRC     A
        MOV     B,A

        MOV     RSTCFG,#LVD2V4
        CALL    DELAY
        ANL     PCON,#NOT LVDF
        CALL    DELAY
        MOV     A, PCON
        ANL     A,#LVDF
        JZ      SKIP
        MOV     A,B
        CLR     C
        RRC     A
        MOV     B,A

        MOV     RSTCFG,#LVD2V2
        CALL    DELAY
        ANL     PCON,#NOT LVDF
        CALL    DELAY
        MOV     A, PCON
        ANL     A,#LVDF
        JZ      SKIP
        MOV     A,B
        CLR     C
        RRC     A
        MOV     B,A

SKIP:
        MOV     A,B
        CPL     A
        MOV     P2,A            ;P2.3~P2.0 Display battery level
        JMP     LOOP

DELAY:
        MOV     R0,#100
```

```
NEXT:
        NOP
        NOP
        NOP
        NOP
        NOP
        DJNZ
        R0,NEXT RET

        END
```

7 memory (unit)

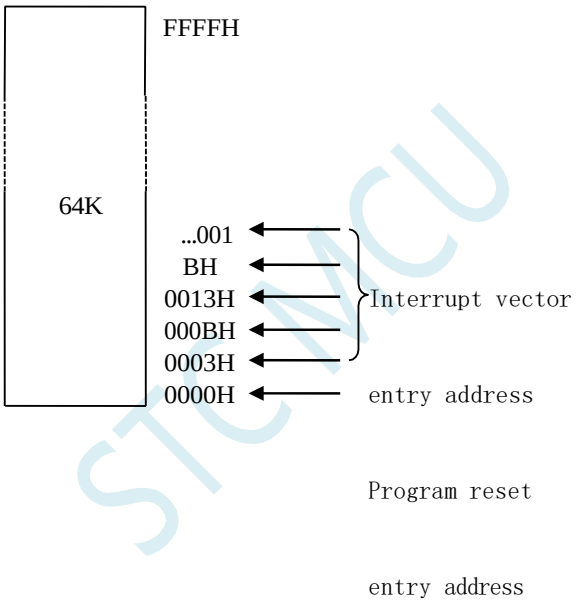
The program memory and data memory of the STC8A8K64D4 series microcontroller are independently addressed. Since no bus is provided to access external program memory, all program memory of the microcontroller is on-chip Flash memory and external program memory cannot be accessed.

The STC8A8K64D4 series microcontroller has a large internal data memory, which is physically and logically divided into two address spaces: the internal RAM (256 bytes) and the internal extended RAM, where the high 128 bytes of the internal RAM overlaps with the addresses of the special function registers (SFRs), which are differentiated by different addressing methods for practical use.

7.1 program memory

Program memory is used to store information such as user programs, data, and tables.

The STC8A8K64D4-64Pin/48Pin series monoliths have an integrated 64K bytes of Flash program memory (ROM) inside.



After a microcontroller reset, the program counter (PC) is 0000H and program execution starts from unit 0000H. Also the entry address of the interrupt service program (also known as the interrupt vector) is located in the program memory cell. In program memory, each interrupt has a fixed entry address, and when an interrupt occurs and is responded to, the microcontroller automatically jumps to the corresponding interrupt entry address to execute the program. The entry address of the interrupt service program for external interrupt 0(INT0) is 0003H, the entry address of the interrupt service program for timer/counter 0 (TIMER0) is 000BH, the entry address of the interrupt service program for external interrupt 1 (INT1) is 0013H, the entry address of the interrupt service program for timer/counter 1 (TIMER1) is 001BH, etc. For more interrupt service program entry addresses (interrupt vectors), please refer to the Interrupt Introduction chapter.

Since the interval between adjacent interrupt entry addresses is only 8 bytes, it is generally not possible to store the complete interrupt service program, so an unconditional transfer instruction is stored in the address area of the interrupt response, pointing to the space where the real interrupt service program is stored for execution.

Flash data memory (EEPROM) is included in all STC8A8K64D4 series microcontrollers. Data is read/written in byte units, and erased in 512-byte page units, and can be repeatedly programmed and erased online more than 100,000 times, improving flexibility and convenience of use.

7.2 data memory

The internal RAM integrated in the STC8A8K64D4 series microcontroller can be used to store intermediate results of program execution and process data.

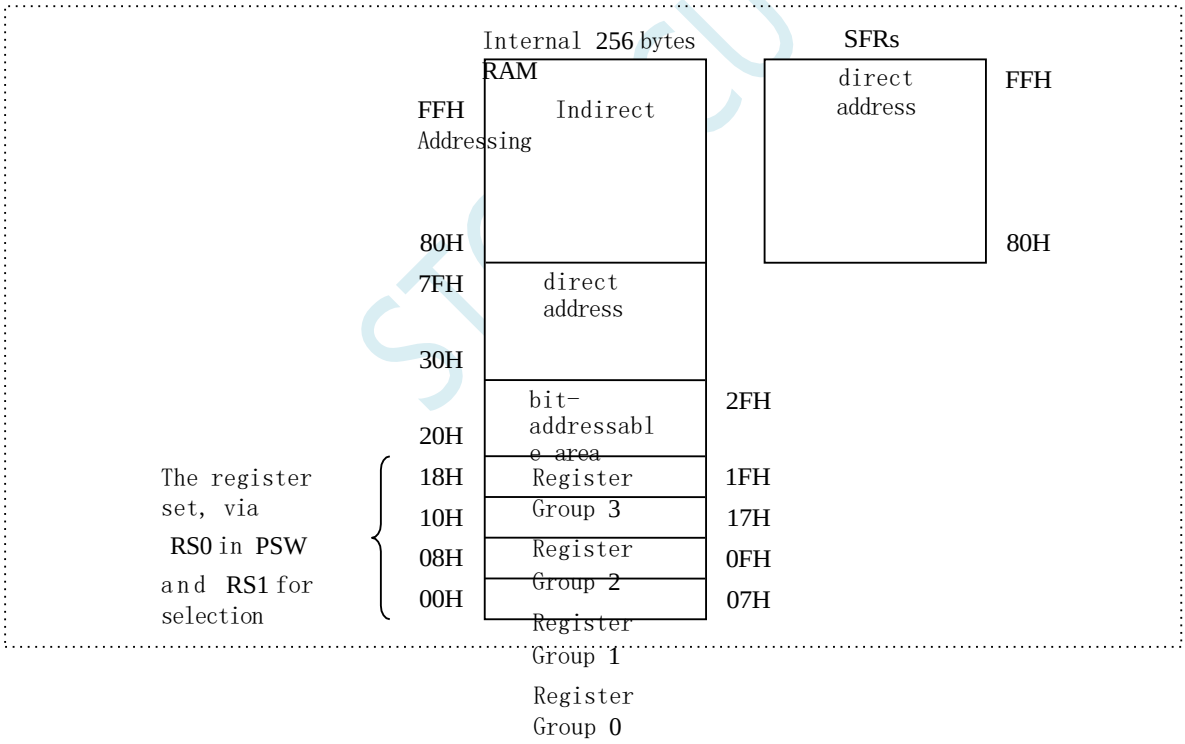
Microcontroller Series	Internal direct access to RAM (DATA)	Internal indirect access to RAM (IDATA)	Internal Expansion RAM (XDATA)
STC8A8K64D4-64Pin/48Pin Series	128 bytes	128 bytes	8192 bytes

7.2.1 Internal RAM

Internal RAM is 256 bytes in total and can be divided into 2 parts: low 128 bytes RAM and high 128 bytes RAM. low 128 bytes of data storage

The 8052 is compatible with the conventional 8051, and can be addressed both directly and indirectly. The high 128-byte RAM (extended by the high 128-byte RAM in the 8052) and the special function register area share the same logical address, using 80H to FFH, but are physically separate and are distinguished by different addressing methods when used. The high 128-byte RAM can only be indirectly addressed, and the special function register area can only be directly addressed.

The structure of the internal RAM is shown in the following figure.



The low 128-byte RAM is also known as the general-purpose RAM area. The general purpose RAM

area can be divided into working register group area, bit addressable area, user RAM area and stack area. The working register group area address is 32 bytes from 00H to 1FH, divided into 4 groups, each group is called a register group, each group contains 8 working registers of 8 bits, all numbered R0 ~ R7, but belong to different physical space. R0~R7 are commonly used registers, and four groups are provided because one group is often not enough. The combination of RS1 and RS0 in the program status word PSW register determines the working register group currently in use, see the description of PSW registers below.

7.2.2 Program Status Register (PSW)

symbolic	address	B7	B6	B5	B4	B3	B2	B1	B0
PSW	D0H	CY	AC	F0	**RS1**	**RS0**	OV	F1	P

CY: In/borrow flag bit.

AC: Auxiliary group in/borrow flag bit. F0: User flag bit 0. RS1, RS0: Working register select bits

RS1	RS0	Working register set (R0~R7)
0	0	Group 0 (00H to 07H)
0	1	Group 1 (08H to 0FH)
1	0	Group 2 (10H to 17H)
1	1	Group 3 (18H to 1FH)

OV: Overflow flag bit. F1: User flag bit 1. P: Parity flag bit.

The address of the bit addressable area is 16 byte cells from 20H to 2FH. 20H~2FH cells can be accessed by byte like ordinary RAM cells, or any bit in the cell can be accessed individually. 128 bits in total, corresponding to a logical bit address range of 00H~7FH. The bit address range is 00H~7FH, and the address of the lower 128 bytes of internal RAM is also 00H~7FH. In fact, they are fundamentally different; a bit address points to a bit, while a byte address points to a byte cell, which is distinguished by using different instructions in the program.

Units 30H~FFH in the internal RAM are the user RAM and the stack area. An 8-bit stack pointer (SP) is used to point to the stack area. After the MCU reset, the stack pointer SP is 07H, which points to R7 in working register group 0. Therefore, the user initialization program should set the initial value of SP, which is generally set to a cell after 80H.

The stack pointer is an 8-bit dedicated register. It indicates the location of the top of the stack in the internal RAM block. After system reset, the SP initialization bit 07H makes the stack in fact start from cell 08H. Considering that cells 08H~1FH belong to working register group 1~3 respectively, if these areas are used in programming, it is better to change the SP value to 80H or larger. the stack of STC8 series microcontroller grows upwards, i.e., the SP content increases after data is pressed into the stack.

7.2.3 Internal Expansion RAM, XRAM, XDATA

The STC8A8K64D4 series microcontrollers integrate internal expansion RAM in addition to 256 bytes of internal RAM. accessing the internal expansion RAM is done in the same way as the conventional 8051 microcontroller accesses the external expansion RAM, but does not affect the signals on the P0 port (data bus and high octet address bus), the P2 port (low octet address bus), and the RD, WR, and ALE ports.

In assembly language, the internal extended RAM is accessed via the MOVX instruction.

```
MOVX    A,@DPTR
```

```
MOVX    @DPTR,A
MOVX    A,@Ri
MOVX    @Ri,A
```

In C, you can just declare the storage type using xdata. For example.

```
unsigned char xdata i;
```

The accessibility of the internal expansion RAM of the microcontroller is controlled by the EXTRAM bit in the auxiliary register AUXR.

7.2.4 Auxiliary Register (AUXR)

symbolic	address	B7	B6	B5	B4	B3	B2	B1	B0
AUXR	8EH	T0x12	T1x12	UART_M0x6	T2R	T2_C/T	T2x12	**EXTRAM**	S1ST2

EXTRAM: Extended RAM Access Control

0: Access to internal expansion RAM.

1: Internal extended RAM is disabled.

7.2.5 External Expansion RAM, XRAM, XDATA

The STC8A8K64D4 series microcontrollers with package pin count 40 and above have the ability to expand 64KB of external data memory. The WR/RD/ALE signals are to be active during access to the external data memory. The STC8A8K64D4 series microcontrollers have a new special function register BUS_SPEED to control the speed of the external 64K byte data bus, described as follows.

7.2.6 Bus Speed Control Register (BUS_SPEED)

symbolic	address	B7	B6	B5	B4	B3	B2	B1	B0
BUS_SPEED	A1H	RW_S[1:0]					SPEED[2:0]		

RW_S[1:0]: RD/WR control line selection bits

00: P4.4 for RD, P4.3 for WR

01: P3.7 for RD, P3.6 for WR

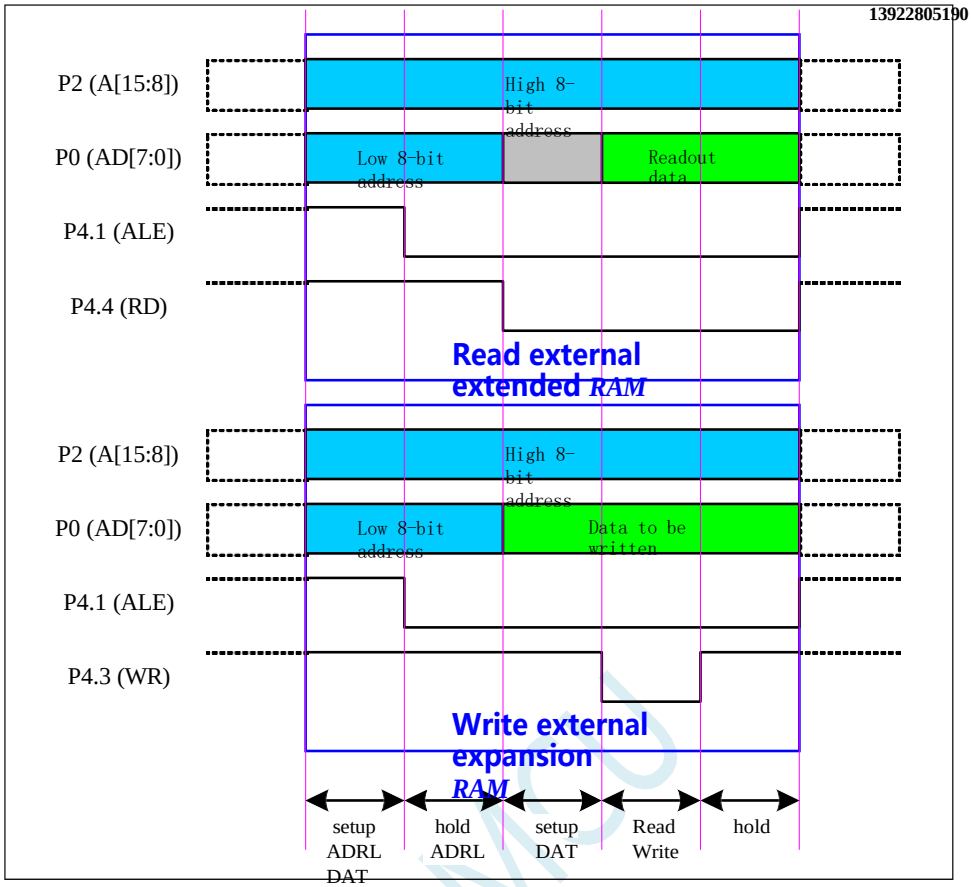
10: RD for P4.2, WR for P4.0

11: Reservations

SPEED[2:0]: bus read/write speed control (ready time and hold time of control signal and data signal when reading and writing data)

command	Number of clocks	
	Accessing Internal Expansion RAM	Accessing External Expansion RAM
MOVX A ,@Ri	3	3+5* (SPEED+1)
MOVX @Ri,A	3	3+5* (SPEED+1)
MOVX A ,@DPTR	2	2+5* (SPEED+1)
MOVX @DPTR,A	2	2+5* (SPEED+1)

The read and write external expansion RAM timings are shown in the following diagram.


P2 (A[15:8])
P0 (AD[7:0])
P4.1 (ALE)
P4.4 (RD)
High 8-bit address
Low 8-bit address
Readout data
Read external extended RAM
P2 (A[15:8])
P0 (AD[7:0])
P4.1 (ALE)
P4.3 (WR)
High 8-bit address
Low 8-bit address
Data to be written
Write external expansion RAM
setup ADRL DAT
hold ADRL
setup DAT
Read Write
hold

7.2.7 Bit-addressable data memory in 8051

The 8051 microcontroller's internal bit-addressable data memory consists of two parts: the first part has an address range of 00H~7FH, and the second part has an address range of 80H~FFH. H, 98H, A0H, A8H, B0H, B8H, C0H, C8H, D0H, D8H, E0H, E8H, F0H, F8H).

Data memory address	bit-addressable address							
	B7	B6	B5	B4	B3	B2	B1	B0
F8H (P7)	FFH F8H.7	FEH F8H.6	FDH F8H.5	FCH F8H.4	FBH F8H.3	FAH F8H.2	F9H F8H.1	F8H F8H.0
F0H(B)	F7H F0H.7	F6H F0H.6	F5H F0H.5	F4H F0H.4	F3H F0H.3	F2H F0H.2	F1H F0H.1	F0H F0H.0
E8H (P6)	EFH E8H.7	EEH E8H.6	EDH E8H.5	ECH E8H.4	EBH E8H.3	EAH E8H.2	E9H E8H.1	E8H E8H.0
E0H (ACC)	E7H E0H.7	E6H E0H.6	E5H E0H.5	E4H E0H.4	E3H E0H.3	E2H E0H.2	E1H E0H.1	E0H E0H.0
D8H (CCON)	DFH D8H.7	DEH D8H.6	DDH D8H.5	DCH D8H.4	DBH D8H.3	DAH D8H.2	D9H D8H.1	D8H D8H.0
D0H (PSW)	D7H D0H.7	D6H D0H.6	D5H D0H.5	D4H D0H.4	D3H D0H.3	D2H D0H.2	D1H D0H.1	D0H D0H.0
C8H (P5)	CFH C8H.7	CEH C8H.6	CDH C8H.5	CCH C8H.4	CBH C8H.3	CAH C8H.2	C9H C8H.1	C8H C8H.0
C0H (P4)	C7H C0H.7	C6H C0H.6	C5H C0H.5	C4H C0H.4	C3H C0H.3	C2H C0H.2	C1H C0H.1	C0H C0H.0
B8H (IP)	BFH B8H.7	BEH B8H.6	BDH B8H.5	BCH B8H.4	BBH B8H.3	BAH B8H.2	B9H B8H.1	B8H B8H.0
B0H (P3)	B7H B0H.7	B6H B0H.6	B5H B0H.5	B4H B0H.4	B3H B0H.3	B2H B0H.2	B1H B0H.1	B0H B0H.0
A8H (IE)	AFH A8H.7	AEH A8H.6	ADH A8H.5	ACH A8H.4	ABH A8H.3	AAH A8H.2	A9H A8H.1	A8H A8H.0
A0H (P2)	A7H A0H.7	A6H A0H.6	A5H A0H.5	A4H A0H.4	A3H A0H.3	A2H A0H.2	A1H A0H.1	A0H A0H.0
98H (SCON)	9FH 98H.7	9EH 98H.6	9DH 98H.5	9CH 98H.4	9BH 98H.3	9AH 98H.2	99H 98H.1	98H 98H.0
90H (P1)	97H 90H.7	96H 90H.6	95H 90H.5	94H 90H.4	93H 90H.3	92H 90H.2	91H 90H.1	90H 90H.0
88H (TCON)	8FH 88H.7	8EH 88H.6	8DH 88H.5	8CH 88H.4	8BH 88H.3	8AH 88H.2	89H 88H.1	88H 88H.0
80H (P0)	87H 80H.7	86H 80H.6	85H 80H.5	84H 80H.4	83H 80H.3	82H 80H.2	81H 80H.1	80H 80H.0
2FH	7FH 2FH.7	7EH 2FH.6	7DH 2FH.5	7CH 2FH.4	7BH 2FH.3	7AH 2FH.2	79H 2FH.1	78H 2FH.0
2EH	77H 2EH.7	76H 2EH.6	75H 2EH.5	74H 2EH.4	73H 2EH.3	72H 2EH.2	71H 2EH.1	70H 2EH.0
2DH	6FH 2DH.7	6EH 2DH.6	6DH 2DH.5	6CH 2DH.4	6BH 2DH.3	6AH 2DH.2	69H 2DH.1	68H 2DH.0
2CH	67H 2CH.7	66H 2CH.6	65H 2CH.5	64H 2CH.4	63H 2CH.3	62H 2CH.2	61H 2CH.1	60H 2CH.0
2BH	5FH 2BH.7	5EH 2BH.6	5DH 2BH.5	5CH 2BH.4	5BH 2BH.3	5AH 2BH.2	59H 2BH.1	58H 2BH.0

2AH	2AH. 7	2AH. 6	55H 2AH. 5	54H 2AH. 4	53H 2AH. 3	52H 2AH. 2	51H 2AH. 1	50H 2AH. 0

29H	4FH 29H. 7	4EH 29H. 6	4DH 29H. 5	4CH 29H. 4	4BH 29H. 3	4AH 29H. 2	49H 29H. 1	48H 29H. 0
28H	47H 28H. 7	46H 28H. 6	45H 28H. 5	44H 28H. 4	43H 28H. 3	42H 28H. 2	41H 28H. 1	40H 28H. 0
27H	3FH 27H. 7	3EH 27H. 6	3DH 27H. 5	3CH 27H. 4	3BH 27H. 3	3AH 27H. 2	39H 27H. 1	38H 27H. 0
26H	37H 26H. 7	36H 26H. 6	35H 26H. 5	34H 26H. 4	33H 26H. 3	32H 26H. 2	31H 26H. 1	30H 26H. 0
25H	2FH 25H. 7	2EH 25H. 6	2DH 25H. 5	2CH 25H. 4	2BH 25H. 3	2AH 25H. 2	29H 25H. 1	28H 25H. 0
24H	27H 24H. 7	26H 24H. 6	25H 24H. 5	24H 24H. 4	23H 24H. 3	22H 24H. 2	21H 24H. 1	20H 24H. 0
23H	1FH 23H. 7	1EH 23H. 6	1DH 23H. 5	1CH 23H. 4	1BH 23H. 3	1AH 23H. 2	19H 23H. 1	18H 23H. 0
22H	17H 22H. 7	16H 22H. 6	15H 22H. 5	14H 22H. 4	13H 22H. 3	12H 22H. 2	11H 22H. 1	10H 22H. 0
21H	0FH 21H. 7	0EH 21H. 6	0DH 21H. 5	0CH 21H. 4	0BH 21H. 3	0AH 21H. 2	09H 21H. 1	08H 21H. 0
20H	07H 20H. 7	06H 20H. 6	05H 20H. 5	04H 20H. 4	03H 20H. 3	02H 20H. 2	01H 20H. 1	00H 20H. 0

7.3 Special parameters in memory that can be burned into the program during ISP download FLASH

The STC8A8K64D4 series microcontroller has some special parameters related to the chip stored in the internal data memory and program memory, including: the globally unique ID number, the frequency of the 32K power-down wake-up timer, the internal 1.19V reference source value, and the IRC parameters.

These parameters are stored in the Flash program memory (ROM) at the following addresses.

Parameter Name	Save Address				Parameter Description
	STC8A8K16D4	STC8A8K32D4	STC8A8K60D4	STC8A8K64D4	
Global Unique ID Number	3FF9H~3FFFH	7FF9H~7FFFH	0EFF9H~0EFFFH	0FDF9H~0FDFFH	7 bytes
Internal 1.19V Reference Signal Source	3FF7H~3FF8H	7FF7H~7FF8H	0EFF7H~0EFF8H	0FDF7H~0FDF8H	Millivolts (high byte first)
32K Power-down wake-up timer frequency	3FF5H~3FF6H	7FF5H~7FF6H	0EFF5H~0EFF6H	0FDF5H~0FDF6H	Hz (high byte first)
IRC parameters for 22.1184 MHz	3FF4H	7FF4H	0EFF4H	0FDF4H	-
24MHz IRC Parameters	3FF3H	7FF3H	0EFF3H	0FDF3H	-
IRC parameters at 20MHz	3FF2H	7FF2H	EFF2H	FDF2H	-
27MHz IRC Parameters	3FF1H	7FF1H	EFF1H	FDF1H	
IRC parameters at 30MHz	3FF0H	7FF0H	EFF0H	FDF0H	
IRC parameters for 33.1776 MHz	3FEFH	7FEFH	EFEFH	FDEFH	
35MHz IRC Parameters	3FEEH	7FEEH	EFEEH	FDEEH	
IRC parameters for 36.864 MHz	3FEDH	7FEDH	EFEDH	FDEDH	
IRC parameters at 40MHz	3FECH	7FECH	EFECH	FDECH	
45MHz IRC Parameters	3FEBH	7FEBH	EFEBH	FDEBH	
VRTRIM parameters for the 6M band	3FEAH	7FEAH	EFEAH	FDEAH	
VRTRIM parameters for the 10M band	3FE9H	7FE9H	EFE9H	FDE9H	
VRTRIM parameters for the 27M band	3FE8H	7FE8H	EFE8H	FDE8H	
VRTRIM parameters for the 44M band	3FE7H	7FE7H	EFE7H	FDE7H	

These parameters are stored in the data memory (RAM) at the following addresses, respectively.

Parameter Name	Save Address	Parameter Description
Internal 1.19V Reference Signal Source	idata: 0EFH~0F0H	Millivolts (high byte first)
Global Unique ID Number	idata: 0F1H~0F7H	7 bytes
32K Power-down wake-up timer frequency	idata: 0F8H~0F9H	Hz (high byte first)

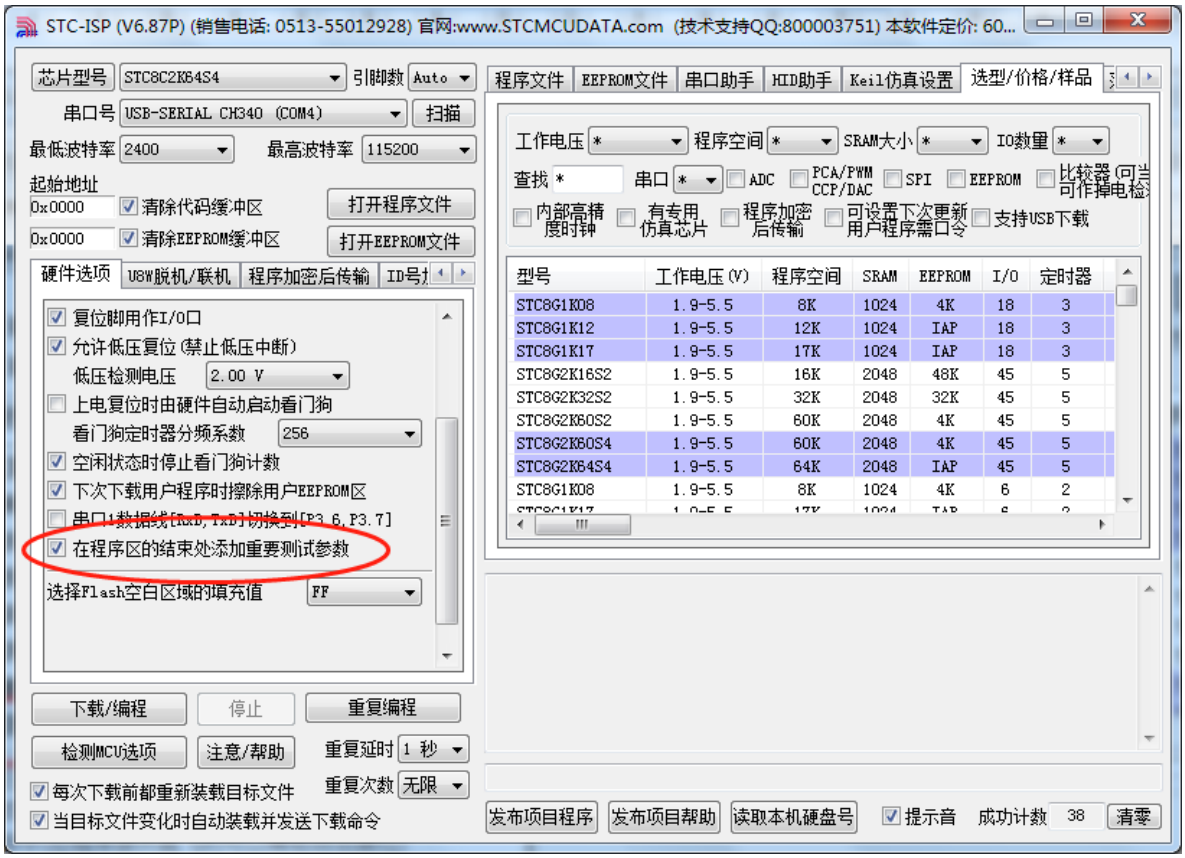
Technical Manual IRC parameters for 22.1184 MHz	www.STCAI.com idata: 0FAH	19864585985 Consultant: 13922805190
24MHz IRC Parameters	idata: 0FBH	-

special note

1. Since the parameters in RAM may be modified, it is generally not recommended for users, especially when users use ID numbers for encryption, and is strongly recommended for reading ID data in FLASH program memory (ROM).

2. Since the size of EEPROM of STC8A8K64S4 and STC8A8K64S2 models can be set by users, it is possible that the FLASH program memory (ROM) space where important parameters are stored is set to EEPROM and important parameters are erased or modified artificially, so this issue may need to be considered when using this model for ID number encryption.

3. By default, the Flash program memory (ROM) contains only data with a globally unique ID number, while the internal 1.19V reference source value, the 32K power-down wake-up timer frequency, and the IRC parameters are not available and need to be selected during ISP download as shown in the following figure to be available.



7.4 Read-only unique ID numbers and important parameters (CHIPID) stored in special function registers

Some STC8A8K64D4 series microcontrollers have built-in 32-byte read-only special function register CHIPID, the contents of which can only be read and not modified by the user program. Using the data in CHIPID to encrypt the user program is the best solution recommended by STC.

Related Registers

symbolic	description	address	Bit Addresses and Symbols								reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
CHIPID00	Hardware number ID00	FDE0H	Globally unique ID number (byte 0)								Nnnn, nnnn
CHIPID01	Hardware number ID01	FDE1H	Globally unique ID number (1st byte)								Nnnn, nnnn
CHIPID02	Hardware number ID02	FDE2H	Globally unique ID number (byte 2)								Nnnn, nnnn
CHIPID03	Hardware number ID03	FDE3H	Globally unique ID number (3rd byte)								Nnnn, nnnn
CHIPID04	Hardware number ID 04	FDE4H	Globally unique ID number (byte 4)								Nnnn, nnnn
CHIPID05	Hardware number ID05	FDE5H	Globally unique ID number (byte 5)								Nnnn, nnnn
CHIPID06	Hardware number ID06	FDE6H	Globally unique ID number (byte 6)								Nnnn, nnnn
CHIPID07	Hardware number ID 07	FDE7H	Internal 1.19V reference signal source (high byte)								Nnnn, nnnn
CHIPID08	Hardware number ID08	FDE8H	Internal 1.19V reference signal source (low byte)								Nnnn, nnnn
CHIPID09	Hardware number ID09	FDE9H	32K Power-down wake-up timer frequency (high byte)								Nnnn, nnnn
CHIPID10	Hardware Digital ID 10	FDEAH	32K Power-down wake-up timer frequency (low byte)								Nnnn, nnnn
CHIPID11	Hardware number ID 11	FDEBH	IRC parameters for 22.1184 MHz (27M band)								Nnnn, nnnn
CHIPID12	Hardware Digital ID 12	FDECH	IRC parameters for 24MHz (27M band)								Nnnn, nnnn
CHIPID13	Hardware number ID 13	FDEDH	IRC parameters for 20MHz (27M band)								Nnnn, nnnn
CHIPID14	Hardware number ID 14	FDEEH	IRC parameters for 27MHz (27M band)								Nnnn, nnnn
CHIPID15	Hardware number ID 15	FDEFH	IRC parameters for 30MHz (27M band)								Nnnn, nnnn
CHIPID16	Hardware numeric ID 16	FDF0H	IRC parameters for 33.1776 MHz (27M band)								Nnnn, nnnn
CHIPID17	Hardware number ID	FDF1H	IRC parameters for 35MHz (44M band)								Nnnn, nnnn

7.4.1 CHIP's Global Unique ID Number Explained

symbolic	description	address	Bit Addresses and Symbols								reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
CHIPID00	Hardware number ID00	FDE0H	Globally unique ID number (byte 0)								Nnnn, nnnn
CHIPID01	Hardware number ID01	FDE1H	Globally unique ID number (1st byte)								Nnnn, nnnn
CHIPID02	Hardware number ID02	FDE2H	Globally unique ID number (byte 2)								Nnnn, nnnn
CHIPID03	Hardware number ID03	FDE3H	Globally unique ID number (3rd byte)								Nnnn, nnnn
CHIPID04	Hardware number ID 04	FDE4H	Globally unique ID number (byte 4)								Nnnn, nnnn
CHIPID05	Hardware number ID05	FDE5H	Globally unique ID number (byte 5)								Nnnn, nnnn
CHIPID06	Hardware number ID06	FDE6H	Globally unique ID number (byte 6)								Nnnn, nnnn

[CHIPID0, CHIPID1]: 16-bit MCU IDs, used to distinguish different MCU models (higher bit first) The MCU IDs commonly used for the STC8A8K64D4 series are shown in the following table.

stc8a8k16d4 (f7f1)
stc8a8k32d4 (f7f2)
stc8a8k48d4 (f7f5)
stc8a8k60d4 (f7f3)
stc8a8k64d4 (f7f4)

[CHIPID2, CHIPID3]: 16-bit test machine number (higher position first)

[CHIPID4, CHIPID5, CHIPID6]: 24-bit test flow number (higher position first)

7.4.2 CHIP's Internal Reference Source Interpretation

symbolic	description	address	Bit Addresses and Symbols								reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
CHIPID07	Hardware number ID 07	FDE7H	Internal 1.19V reference signal source (high byte)								Nnnn, nnnn
CHIPID08	Hardware number ID08	FDE8H	Internal 1.19V reference signal source (low byte)								Nnnn, nnnn

[CHIPID7, CHIPID8]: 16-bit internal reference signal source voltage value (high bit first)

The standard value is 1190 (04A6H)in mV, i.e. 1.19 V. However, the actual chip is subject to manufacturing errors. The voltage value of the internal reference source is not affected by the operating voltage VCC, so the internal reference source can be used in combination with an ADC to calibrate the ADC, or with a comparator to detect the operating voltage.

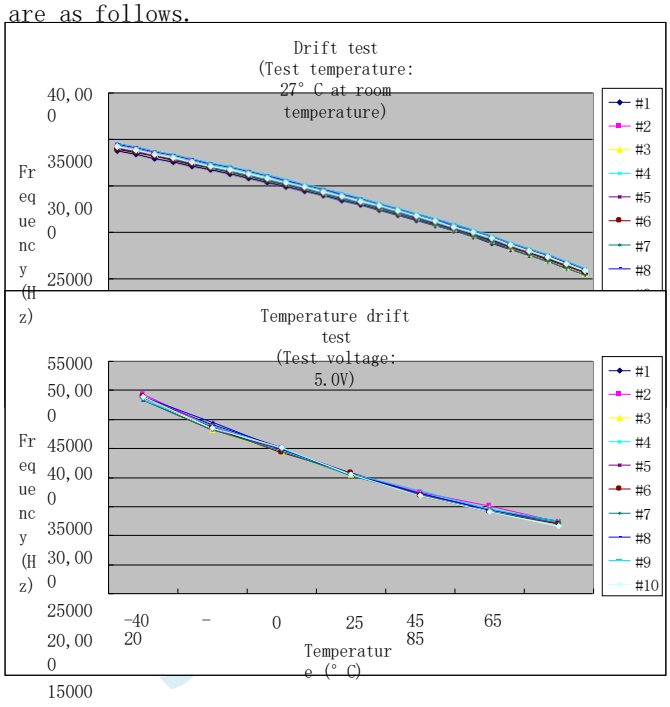
7.4.3 CHIP's internal 32K IRC oscillation frequency interpretation

symbolic	description	address	Bit Addresses and Symbols								reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
CHIPID09	Hardware number ID09	FDE9H	32K Power-down wake-up timer frequency (high byte)								Nnnn, nnnn
CHIPID10	Hardware Digital ID 10	FDEAH	32K Power-down wake-up timer frequency (low byte)								Nnnn, nnnn

[CHIPID9, CHIPID10]: 16-bit 32K IRC oscillator frequency value (higher bit first)

The standard value is 32768 (8000H) in Hz, or 32.768 KHz, but the actual chip is subject to manufacturing errors and has a large temperature and voltage drift.

The linearity plot of the internal 32K oscillator for the voltage drift test and the temperature drift linearity plot are as follows.



7.4.4 CHIP's High Precision IRC Parameter Interpretation

symbolic	description	address	Bit Addresses and Symbols								reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
CHIPID11	Hardware number ID 11	FDEBH	IRC parameters for 22.1184 MHz (27M band)								Nnnn, nnnn
CHIPID12	Hardware Digital ID 12	FDECH	IRC parameters for 24MHz (27M band)								Nnnn, nnnn
CHIPID13	Hardware number ID 13	FDEDH	IRC parameters for 20MHz (27M band)								Nnnn, nnnn
CHIPID14	Hardware number ID 14	FDEEH	IRC parameters for 27MHz (27M band)								Nnnn, nnnn
CHIPID15	Hardware number ID 15	FDEFH	IRC parameters for 30MHz (27M band)								Nnnn, nnnn
CHIPID16	Hardware number ID 16	FDF0H	IRC parameters for 33.1776 MHz (27M band)								Nnnn, nnnn
CHIPID17	Hardware number ID 17	FDF1H	IRC parameters for 35MHz (44M band)								Nnnn, nnnn
CHIPID18	Hardware number ID 18	FDF2H	IRC parameters for 36.864 MHz (44M band)								Nnnn, nnnn
CHIPID19	Hardware number ID 19	FDF3H	IRC parameters for 40MHz (44M band)								Nnnn, nnnn
CHIPID20	Hardware Digital ID 20	FDF4H	IRC parameters for 45MHz (44M band)								Nnnn, nnnn
CHIPID21	Hardware number ID 21	FDF5H	VRTRIM parameters for the 6M band								Nnnn, nnnn
CHIPID22	Hardware number ID 22	FDF6H	VRTRIM parameters for the 10M band								Nnnn, nnnn
CHIPID23	Hardware number ID 23	FDF7H	VRTRIM parameters for the 27M band								Nnnn, nnnn
CHIPID24	Hardware number ID 24	FDF8H	VRTRIM parameters for the 44M band								Nnnn, nnnn

STC8A8K64D4 series microcontroller, the internal integrated high precision IRC is divided into 4 frequency bands, each band corresponding to the reference voltage value has been calibrated at the factory, when selecting different frequency bands, only need to fill the corresponding frequency band voltage calibration value into the VRTRIM register. The center frequencies of the 4 bands are 6MHz, 10MHz, 27MHz and 44MHz respectively. Due to manufacturing errors, the center frequencies may deviate by ±5% in general. When the user program is downloaded using the official download software provided by STC, the system automatically sets the VRTRIM and IRTRIM registers according to the user set frequency. The IRTRIM values for 10 common frequencies and the reference voltage calibration values for 4 frequency bands are also preset internally in CHIPID, allowing the user to dynamically modify the operating frequency during program operation.

[CHIPID11 : CHIPID20]: IRTRIM values for 10 common frequencies. The notes inside the parentheses are the corresponding frequency bands

[CHIPID21 : CHIPID24]: Calibration value of the reference voltage value for the 4 bands.

To dynamically modify the frequency, user only needs to read out a frequency calibration value from [CHIPID11：CHIPID20] and write it to the IRTRIM register, and read out a voltage calibration value from [CHIPID21：CHIPID24] and write it to the VRTRIM register according to the frequency band corresponding to that frequency. Please refer to the sample program in the following section for details.

7.4.5 Interpretation of CHIP's test time parameters

symbolic	description	address	Bit Addresses and Symbols								reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
CHIPID27	Hardware number ID 27	FDFBH	Chip test time (years)								Nnnn, nnnn
CHIPID28	Hardware number ID 28	FDFCH	Chip test time (months)								Nnnn, nnnn
CHIPID29	Hardware number ID 29	FDFDH	Chip test time (days)								Nnnn, nnnn

The year, month and day parameters of the test time are BCD codes (e.g. CHIPID27 = 0x21, CHIPID28 = 0x11, CHIPID29 = 0x18, the production test date of the target chip is November 18, 2021)

7.4.6 CHIP's chip package form number interpretation

symbolic	description	address	Bit Addresses and Symbols								reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
CHIPID30	Hardware Digital ID 30	FDFEH	Chip Package Form Number								Nnnn, nnnn

Package Number	Package form		Package Number	Package form
0x00	DIP8		0x50	SOP32
0x01	SOP8		0x51	LQFP32
0x02	DFN8		0x52	QFN32
0x10	DIP16		0x53	PLCC32
0x11	SOP16		0x54	QFN32S
0x20	DIP18		0x60	PDIP40
0x21	SOP18		0x70	LQFP44
0x30	DIP20		0x71	PLCC44
0x31	SOP20		0x72	PQFP44
0x32	TSSOP20		0x80	LQFP48
0x33	LSSOP20		0x81	QFN48
0x34	QFN20		0x90	LQFP64
0x40	SKDIP28		0x91	LQFP64S
0x41	SOP28		0x92	LQFP64L
0x42	TSSOP28		0x93	LQFP64M
0x43	QFN28		0x94	QFN64

7.5 paradigm procedure

7.5.1 Read internal 1.19V reference source value (from CHIPID)

C code

```
// Tested operating frequency of 11.0592MHz

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

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

#define   CPUIDBASE       0xfde0

#define   ID_ADDR         ((unsigned char volatile xdata *)(CPUIDBASE + 0x00))
#define   VREF_ADDR       (*(unsigned int volatile xdata *)(CPUIDBASE + 0x07))
#define   F32K_ADDR       (*(unsigned int volatile xdata *)(CPUIDBASE + 0x09))
#define   T22M_ADDR       (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0b))   //22.1184MHz
#define   T24M_ADDR       (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0c))   //24MHz
#define   T20M_ADDR       (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0d))   //20MHz
#define   T27M_ADDR       (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0e))   //27MHz
#define   T30M_ADDR       (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0f))   //30MHz
#define   T33M_ADDR       (*(unsigned char volatile xdata *)(CPUIDBASE + 0x10))   //33.1776MHz
#define   T35M_ADDR       (*(unsigned char volatile xdata *)(CPUIDBASE + 0x11))   //35MHz
#define   T36M_ADDR       (*(unsigned char volatile xdata *)(CPUIDBASE + 0x12))   //36.864MHz
#define   T40M_ADDR       (*(unsigned char volatile xdata *)(CPUIDBASE + 0x13))   //40MHz
#define   T45M_ADDR       (*(unsigned char volatile xdata *)(CPUIDBASE + 0x14))   //45MHz
#define   VRT6M_ADDR      (*(unsigned char volatile xdata *)(CPUIDBASE + 0x15))   //VRTRIM_6M
#define   VRT10M_ADDR     (*(unsigned char volatile xdata *)(CPUIDBASE + 0x16))   //VRTRIM_10M
#define   VRT27M_ADDR     (*(unsigned char volatile xdata *)(CPUIDBASE + 0x17))   //VRTRIM_27M
#define   VRT44M_ADDR     (*(unsigned char volatile xdata *)(CPUIDBASE + 0x18))   //VRTRIM_44M

sfr       AUXR      =     0x8e;
sfr       P_SW2     =     0xba;

sfr       p0m1      =     0x93;
sfr       p0m0      =     0x94;
sfr       p1m1      =     0x91;
sfr       p1m0      =     0x92;
sfr       p2m1      =     0x95;
sfr       p2m0      =     0x96;
sfr       p3m1      =     0xb1;
sfr       p3m0      =     0xb2;
sfr       p4m1      =     0xb3;
sfr       p4m0      =     0xb4;
sfr       p5m1      =     0xc9;
sfr       p5m0      =     0xca;
sfr

bit       busy;

void UartIsr() interrupt 4
```

if (TI)

```
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
    }
}

void UartInit()
{
    SCON = 0x50;
    TMOD =
    0x00;  TL1  =
    BRT;
    TH1  =  BRT  >>
    8; TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}

void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}

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

    UartInit();
    ES = 1;
    EA = 1;
    P_SW2 =
    0x80;
    UartSend(VREF_ADDR >> 8);          //read the high byte of the internal 1.19V reference
    signal source
    UartSend(VREF_ADDR);               //read the low byte of the internal 1.19V reference
    signal source
```

```
    while (1);
}
```

assembly code

The test operating frequency is *11.0592MHz*

```
CPUIDBASE       EQU     0FDE0H
```

```
ID_ADDR         EQU     CPUIDBASE + 00H
VREF_ADDR       EQU     CPUIDBASE + 07H
F32K_ADDR       EQU     CPUIDBASE + 09H
T22M_ADDR       EQU     CPUIDBASE + 0BH         ;22.1184MHz
T24M_ADDR       EQU     CPUIDBASE + 0CH         ;24MHz
T20M_ADDR       EQU     CPUIDBASE + 0DH         ;20MHz
T27M_ADDR       EQU     CPUIDBASE + 0EH         ;27MHz
T30M_ADDR       EQU     CPUIDBASE + 0FH         ;30MHz
T33M_ADDR       EQU     CPUIDBASE + 10H         ;33.1776MHz
T35M_ADDR       EQU     CPUIDBASE + 11H         ;35MHz
T36M_ADDR       EQU     CPUIDBASE + 12H         ;36.864MHz
T40M_ADDR       EQU     CPUIDBASE + 13H         ;40MHz
T45M_ADDR       EQU     CPUIDBASE + 14H         ;45MHz
VRT6M_ADDR      EQU     CPUIDBASE + 15H         ;VRTRIM_6M
VRT10M_ADDR     EQU     CPUIDBASE + 16H         ;VRTRIM_10M
VRT27M_ADDR     EQU     CPUIDBASE + 17H         ;VRTRIM_27M
VRT44M_ADDR     EQU     CPUIDBASE + 18H         ;VRTRIM_44M

AUXR            DATA    8EH
P_SW2           DATA    0BAH

BUSY            BIT     20H.0

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

                ORG     0000H
                LJMP    MAIN
                ORG     0023H
                LJMP    UART_ISR

                ORG     0100H

UART_ISR:
                JNB     TI, CHKRI
                CLR     TI
                CLR     BUSY
CHKRI:
                JNB     RI,UARTISR_EXIT
                CLR     RI
UARTISR_EXIT:
                RETI

UART_INIT:
                MOV     SCON,#50H
                MOV     TMOD,#00H
                MOV     TL1,#0E8H               65536-11059200/115200/4=0FFE8H
                MOV     TH1,#0FFH
                SETB    TR1
```

```
        MOV     AUXR,#40H
        CLR     BUSY
        RET

UART_SEND:
        JB      BUSY,$
        SETB    BUSY
        MOV     SBUF,A
        RET

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

        LCALL   UART_INIT
        SETB    ES
        SETB    EA

        MOV     P_SW2,#80H
        MOV     DPTR,# VREF_ADDR
        CLR     A
        MOVX    A,@DPTR         Read the high byte of the internal 1.19V reference signal source
        LCALL   UART_SEND
        INC     DPTR
        MOVX    A,@DPTR         Read the low byte of the internal 1.19V reference signal source
        LCALL   UART_SEND

LOOP:
        JMP     LOOP

        END
```

7.5.2 Read internal 1.19V reference source value (from Flash program memory (ROM)) read in)

C code

```
// Tested operating frequency of 11.0592MHz

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

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

```
sfr     AUXR    =   0x8e;

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

bit     busy;
int     *BGV;

void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
    }
}

void UartInit()
{
    SCON = 0x50;
    TMOD =
    0x00; TL1 =
    BRT;
    TH1 = BRT >>
    8; TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}

void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}

void main()
{
    P0M0 =
    0x00; P0M1
    = 0x00;
    P1M0 =
    0x00; P1M1
    = 0x00;
    P2M0 =
    0x00; P2M1
```

= 0x00;
P3M0 = 0x00; P3M1 = 0x00;

```
    P4M0 = 0x00; P4M1 = 0x00;
    P5M0 = 0x00; P5M1 = 0x00;

    BGV = (int code *)0xeff7;                   // STC8A8K60S4
    UartInit();
    ES = 1;
    EA = 1;
    UartSend(*BGV >> 8);                        // Read the high byte of the internal 1.19V reference signal source
    UartSend(*BGV);                             // Read the low byte of the internal 1.19V reference signal source

    while (1);
}
```

assembly code

Operating frequency tested at 11.0592MHz

```
AUXR        DATA    8EH
BGV         EQU     0EFF7H          ;STC8A8K60S4

BUSY        BIT     20H.0

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

            ORG     0000H
            LJMP    MAIN
            ORG     0023H
            LJMP    UART_ISR

            ORG     0100H

UART_ISR:
            JNB     TI, CHKRI
            CLR     TI
            CLR     BUSY
CHKRI:
            JNB     RI,UARTISR_EXIT
            CLR     RI
UARTISR_EXIT:
            RETI

UART_INIT:
```

```
MOV     SCON,#50H
MOV     TMOD,#00H
MOV     TL1,#0E8H       65536-11059200/115200/4=0FFE8H
MOV     TH1,#0FFH
```

```
                SETB    TR1
                MOV     AUXR,#40H
                CLR     BUSY
                RET

UART_SEND:
                JB      BUSY,$
                SETB    BUSY
                MOV     SBUF,A
                RET

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

                LCALL   UART_INIT
                SETB    ES
                SETB    EA

                MOV     DPTR,#BGV
                CLR     A
                MOVC    A,@A+DPTR       Read the high byte of the internal 1.19V reference signal source
                LCALL   UART_SEND
                MOV     A,#1
                MOVC    A,@A+DPTR       Read the low byte of the internal 1.19V reference signal source
                LCALL   UART_SEND

LOOP:
                JMP     LOOP

                END
```

7.5.3 ~~Read internal 1.19V reference source value (from RAM)~~

C code

```
// Tested operating frequency of 11.0592MHz

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

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

```
sfr  AUXR  =  0x8e;
```

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

bit     busy;
int     *BGV;

void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
    }
}

void UartInit()
{
    SCON = 0x50;
    TMOD =
    0x00; TL1 =
    BRT;
    TH1 = BRT >>
    8; TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}

void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}

void main()
{
    P0M0 =
    0x00; P0M1
    = 0x00;
    P1M0 =
    0x00; P1M1
    = 0x00;
    P2M0 =
    0x00; P2M1
    = 0x00;
```

```
    P3M0 = 0x00;    P3M1 = 0x00;
```

```
    P5M0 = 0x00;
    P5M1 = 0x00;

    BGV = (int idata *)0xef;
    UartInit();
    ES = 1;
    EA = 1;
    UartSend(*BGV >> 8);                    // Read the high byte of the internal 1.19V reference signal source
    UartSend(*BGV);                         // Read the low byte of the internal 1.19V reference signal source

    while (1);
}
```

assembly code

Operating frequency tested at 11.0592MHz

```
AUXR        DATA    8EH
BGV         DATA    0EFH

BUSY        BIT     20H.0

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

            ORG     0000H
            LJMP    MAIN
            ORG     0023H
            LJMP    UART_ISR

            ORG     0100H

UART_ISR:
            JNB     TI, CHKRI
            CLR     TI
            CLR     BUSY
CHKRI:
            JNB     RI,UARTISR_EXIT
            CLR     RI
UARTISR_EXIT:
            RETI

UART_INIT:
            MOV     SCON,#50H
            MOV     TMOD,#00H
            MOV     TL1,#0E8H           ;65536-
```

1
0
5
9
2
0
0
0
/
1
1
5
2
0
0
0
/
4
=
0
F
F
E
8
H

M
O
V

T
H
1
,
#
0
F
F
H
SETB TR1
MOV AUXR,#40H

```
                CLR     BUSY
                RET

UART_SEND:
                JB      BUSY,$
                SETB    BUSY
                MOV     SBUF,A
                RET

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

                LCALL   UART_INIT
                SETB    ES
                SETB    EA

                MOV     R0,#BGV
                MOV     A,@R0           Read the high byte of the internal 1.19V reference signal source

                LCALL   UART_SEND
                INC     R0
                MOV     A,@R0           Read the low byte of the internal 1.19V reference signal source

                LCALL   UART_SEND

LOOP:
                JMP     LOOP

                END
```

7.5.4 Read globally unique ID number (from CHIPID)

C code

```
// Tested operating frequency of 11.0592MHz

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

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

#define   CPUIDBASE     0xfde0

#define   ID_ADDR       ((unsigned char volatile xdata *)(CPUIDBASE + 0x00))
```

```
#define VREF_ADDR (*(unsigned long volatile xdata *)(CPUIDBASE+0x07))
```

```
#define     F32K_ADDR       (*(unsigned int volatile xdata *)(CPUIDBASE + 0x09))
#define     T22M_ADDR       (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0b))   //22.1184MHz
#define     T24M_ADDR       (*(unsigned char volatile xdata *)(CPUIDBASE +          //24MHz
#define     T20M_ADDR       0x0c)) (*(unsigned char volatile xdata *)(CPUIDBASE      //20MHz
#define     T27M_ADDR       + 0x0d)) (*(unsigned char volatile xdata                 //27MHz
#define     T30M_ADDR       *)(CPUIDBASE + 0x0e)) (*(unsigned char volatile         //30MHz
#define     T33M_ADDR       xdata *)(CPUIDBASE + 0x0f)) (*(unsigned char volatile   //33.1776MHz
#define     T35M_ADDR       xdata *)(CPUIDBASE + 0x10)) (*(unsigned char            //35MHz
#define     T36M_ADDR       volatile xdata *)(CPUIDBASE + 0x11)) (*(unsigned        //36.864MHz
#define     T40M_ADDR       char volatile xdata *)(CPUIDBASE + 0x12))               //40MHz
#define     T45M_ADDR       (*(unsigned char volatile xdata *)(CPUIDBASE +          //45MHz
#define     VRT6M_ADDR      0x13)) (*(unsigned char volatile xdata *)(CPUIDBASE      //VRTRIM_6M
#define     VRT10M_ADDR     + 0x14)) (*(unsigned char volatile xdata                 //VRTRIM_10M
#define     VRT27M_ADDR     *)(CPUIDBASE + 0x15)) (*(unsigned char volatile          //VRTRIM_27M
#define     VRT44M_ADDR     xdata *)(CPUIDBASE + 0x16)) (*(unsigned char             //VRTRIM_44M
#define                     volatile xdata *)(CPUIDBASE + 0x17)) (*(unsigned
#define                     char volatile xdata *)(CPUIDBASE + 0x18))
#define
#define
#define
#define
#define
#define
#define

sfr         AUXR        =   0x8e;
sfr         P_SW2       =   0xba;

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

bit         busy;

void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
    }
}

void UartInit()
{
    SCON = 0x50;
    TMOD =
```

```
    0x10;                   TL1 =
    BRT;
    TH1 = BRT >>
    8; TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}

void UartSend(char dat)
```

```
{
    while (busy);
    busy = 1;
    SBUF = dat;
}

void main()
{
    char i;

    P0M0 = 0x00; P0M1 = 0x00;
    P1M0 = 0x00; P1M1 = 0x00;
    P2M0 = 0x00; P2M1 = 0x00;
    P3M0 = 0x00; P3M1 = 0x00;
    P4M0 = 0x00; P4M1 = 0x00;
    P5M0 = 0x00; P5M1 = 0x00;

    UartInit();
    ES = 1;
    EA = 1;

    P_SW2 = 0x80;
    for (i=0; i<7; i++)
    {
        UartSend(ID_ADDR[i]);
    }

    while (1);
}
```

assembly code

Operating frequency tested at *11.0592MHz*

```
CPUIDBASE       EQU     0FDE0H

ID_ADDR         EQU     CPUIDBASE + 00H
VREF_ADDR       EQU     CPUIDBASE + 07H
F32K_ADDR       EQU     CPUIDBASE + 09H
T22M_ADDR       EQU     CPUIDBASE + 0BH     ;22.1184MHz
T24M_ADDR       EQU     CPUIDBASE + 0CH     ;24MHz
T20M_ADDR       EQU     CPUIDBASE + 0DH     ;20MHz
T27M_ADDR       EQU     CPUIDBASE + 0EH     ;27MHz
T30M_ADDR       EQU     CPUIDBASE + 0FH     ;30MHz
T33M_ADDR       EQU     CPUIDBASE + 10H     ;33.1776MHz
T35M_ADDR       EQU     CPUIDBASE + 11H     ;35MHz
T36M_ADDR       EQU     CPUIDBASE + 12H     ;36.864MHz
T40M_ADDR       EQU     CPUIDBASE + 13H     ;40MHz
```

```
T45M_ADDR       EQU     CPUIDBASE + 14H     ;45MHz
VRT6M_ADDR      EQU     CPUIDBASE + 15H     ;VRTRIM_6M
VRT10M_ADDR     EQU     CPUIDBASE + 16H     ;VRTRIM_10M
VRT27M_ADDR     EQU     CPUIDBASE + 17H     ;VRTRIM_27M
VRT44M_ADDR     EQU     CPUIDBASE + 18H     ;VRTRIM_44M
```

```
AUXR        DATA    8EH
P_SW2       DATA    0BAH

BUSY        BIT     20H.0

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

            ORG     0000H
            LJMP    MAIN
            ORG     0023H
            LJMP    UART_ISR

            ORG     0100H

UART_ISR:
            JNB     TI, CHKRI
            CLR     TI
            CLR     BUSY
CHKRI:
            JNB     RI,UARTISR_EXIT
            CLR     RI
UARTISR_EXIT:
            RETI

UART_INIT:
            MOV     SCON,#50H
            MOV     TMOD,#00H
            MOV     TL1,#0E8H           ;65536-
11059200/115200/4=0FFE8H MOV            TH1,#0FFH
            SETB    TR1
            MOV     AUXR,#40H
            CLR     BUSY
            RET

UART_SEND:
            JB      BUSY,$
            SETB    BUSY
            MOV
            SBUF,A RET

MAIN:
            MOV     SP, #5FH
            MOV     P0M0, #00H
            MOV     P0M1, #00H
            MOV     P1M0, #00H
            MOV     P1M1, #00H
            MOV     P2M0, #00H
```

```
                MOV     P2M1, #00H
                MOV     P3M0, #00H
                MOV     P3M1, #00H
                MOV     P4M0, #00H
                MOV     P4M1, #00H
                MOV     P5M0, #00H
                MOV     P5M1, #00H

                LCALL   UART_INIT
                SETB    ES
                SETB    EA

                MOV     P_SW2,#80H
                MOV     DPTR,#ID_ADDR
                MOV     R1,#7
NEXT:           CLR     A
                MOVX    A,@DPTR
                LCALL   UART_SEND
                INC     DPTR
                DJNZ    R1,NEXT

LOOP:
                JMP     LOOP

                END
```

7.5.5 Reads globally unique ID number (from Flash program memory (ROM))

C code

```
// Tested operating frequency of 11.0592MHz

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

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

sfr       AUXR        =     0x8e;

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

bit       busy;
char      *ID;
```

```
void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
    }
}

void UartInit()
{
    SCON = 0x50;
    TMOD =
    0x00; TL1 =
    BRT;
    TH1 = BRT >>
    8; TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}

void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}

void main()
{
    char i;

    P0M0 =
    0x00; P0M1
    = 0x00;
    P1M0 =
    0x00; P1M1
    = 0x00;
    P2M0 =
    0x00; P2M1
    = 0x00;
    P3M0 =
    0x00; P3M1
    = 0x00;
    P4M0 =
    0x00; P4M1
    = 0x00;
    P5M0 =
    0x00; P5M1
    = 0x00;

    ID = (char code *)0xeff9;                    // STC8A8K60S4
    UartInit();
    ES = 1;
    EA = 1;
```

```
    for (i=0; i<7; i++)
    {
        UartSend(ID[i]);
    }

    while (1);
```

assembly code

Operating frequency tested at *11.0592MHz*

```
AUXR    DATA    8EH
ID      EQU     0EFF9H          STC8A8K60S4

BUSY    BIT     20H.0

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

        ORG     0000H
        LJMP    MAIN
        ORG     0023H
        LJMP    UART_ISR

        ORG     0100H

UART_ISR:
        JNB     TI, CHKRI
        CLR     TI
        CLR     BUSY
CHKRI:
        JNB     RI,UARTISR_EXIT
        CLR     RI
UARTISR_EXIT:
        RETI

UART_INIT:
        MOV     SCON,#50H
        MOV     TMOD,#00H
        MOV     TL1,#0E8H       ;65536-
11059200/115200/4=0FFE8H MOV    TH1,#0FFH
        SETB    TR1
        MOV     AUXR,#40H
        CLR     BUSY
        RET

UART_SEND:
        JB      BUSY,$
        SETB    BUSY
        MOV
        SBUF,A RET

MAIN:
        MOV     SP, #5FH
```

```
        MOV     P0M0, #00H
        MOV     P0M1, #00H
        MOV     P1M0, #00H
        MOV     P1M1, #00H
        MOV     P2M0, #00H
        MOV     P2M1, #00H
        MOV     P3M0, #00H
        MOV     P3M1, #00H
        MOV     P4M0, #00H
        MOV     P4M1, #00H
        MOV     P5M0, #00H
        MOV     P5M1, #00H

        LCALL   UART_INIT
        SETB    ES
        SETB    EA

        MOV     DPTR,#ID
        MOV     R1,#7
NEXT:   CLR     A
        MOVC    A,@A+DPTR
        LCALL   UART_SEND
        INC     DPTR
        DJNZ    R1,NEXT

LOOP:
        JMP     LOOP

        END
```

7.5.6 Read globally unique ID number (from RAM)

C code

```
// Tested operating frequency of 11.0592MHz

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

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

sfr   AUXR  =  0x8e;

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

```
bit         busy;
char        *ID;

void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
    }
}

void UartInit()
{
    SCON = 0x50;
    TMOD =
    0x00; TL1 =
    BRT;
    TH1 = BRT >>
    8; TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}

void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}

void main()
{
    char i;

    P0M0 =
    0x00; P0M1
    = 0x00;
    P1M0 =
    0x00; P1M1
    = 0x00;
    P2M0 =
    0x00; P2M1
    = 0x00;
    P3M0 =
    0x00; P3M1
    = 0x00;
    P4M0 =
    0x00; P4M1
    = 0x00;
    P5M0 =
    0x00; P5M1
    = 0x00;

    ID = (char idata *)0xf1;
```

```
    Uart_Init();
    ES = 1;
    EA = 1;

    for (i=0; i<7; i++)
    {
```

```
        UartSend(ID[i]);
    }

    while (1);
}
```

assembly code

Operating frequency tested at *11.0592MHz*

```
AUXR    DATA    8EH
ID      DATA    0F1H

BUSY    BIT     20H.0

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

        ORG     0000H
        LJMP    MAIN
        ORG     0023H
        LJMP    UART_ISR

        ORG     0100H

UART_ISR:
        JNB     TI, CHKRI
        CLR     TI
        CLR     BUSY
CHKRI:
        JNB     RI,UARTISR_EXIT
        CLR     RI
UARTISR_EXIT:
        RETI

UART_INIT:
        MOV     SCON,#50H
        MOV     TMOD,#00H
        MOV     TL1,#0E8H       ;65536-11059200/115200/4=0FFE8H MOV     TH1,#0FFH
        SETB    TR1
        MOV     AUXR,#40H
        CLR     BUSY
        RET

UART_SEND:
        JB      BUSY,$
        SETB    BUSY
        MOV     SBUF,A
```

```
                RET

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

                LCALL   UART_INIT
                SETB    ES
                SETB    EA

                MOV     R0,#ID
                MOV     R1,#7
NEXT:           MOV     A,@R0
                LCALL   UART_SEND
                INC     R0
                DJNZ    R1,NEXT

LOOP:
                JMP     LOOP

                END
```

7.5.7 Read 32K power-down wake-up timer frequency (from CHIPID)

C code

```
// Tested operating frequency of 11.0592MHz

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

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

#define   CPUIDBASE     0xfde0

#define   ID_ADDR       ((unsigned char volatile xdata *)(CPUIDBASE +
#define   VREF_ADDR     0x00)) (*(unsigned int volatile xdata *)(CPUIDBASE +
#define   F32K_ADDR     0x07)) (*(unsigned int volatile xdata *)(CPUIDBASE +
#define   T22M_ADDR     0x09))                                          //22.1184MHz
                        (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0b))
#define   T24M_ADDR     (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0c))  //24MHz
#define   T20M_ADDR     (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0d))  //20MHz
#define   T27M_ADDR     (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0e))  //27MHz
```

```
#define   T30M_ADDR   (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0f))   //30MHz
#define   T33M_ADDR   (*(unsigned char volatile xdata *)(CPUIDBASE + 0x10))   //33.1776MHz
```

```
#define   T35M_ADDR   (*(unsigned char volatile xdata *)(CPUIDBASE + 0x11))   //35MHz
#define   T36M_ADDR   (*(unsigned char volatile xdata *)(CPUIDBASE + 0x12))   //36.864MHz
#define   T40M_ADDR   (*(unsigned char volatile xdata *)(CPUIDBASE + 0x13))   //40MHz
#define   T45M_ADDR   (*(unsigned char volatile xdata *)(CPUIDBASE + 0x14))   //45MHz
#define   VRT6M_ADDR  (*(unsigned char volatile xdata *)(CPUIDBASE + 0x15))   //VRTRIM_6M
#define   VRT10M_ADDR (*(unsigned char volatile xdata *)(CPUIDBASE + 0x16))   //VRTRIM_10M
#define   VRT27M_ADDR (*(unsigned char volatile xdata *)(CPUIDBASE + 0x17))   //VRTRIM_27M
#define   VRT44M_ADDR (*(unsigned char volatile xdata *)(CPUIDBASE + 0x18))   //VRTRIM_44M

sfr   AUXR   = 0x8e;
sfr   P_SW2  = 0xba;

sfr   p0m1   = 0x93;
sfr   p0m0   = 0x94;
sfr   p1m1   = 0x91;
sfr   p1m0   = 0x92;
sfr   p2m1   = 0x95;
sfr   p2m0   = 0x96;
sfr   p3m1   = 0xb1;
sfr   p3m0   = 0xb2;
sfr   p4m1   = 0xb3;
sfr   p4m0   = 0xb4;
sfr   p5m1   = 0xc9;
sfr   p5m0   = 0xca;
sfr

bit   busy;

void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
    }
}

void UartInit()
{
    SCON = 0x50;
    TMOD =
    0x00;  TL1 =
    BRT;
    TH1 = BRT >>
    8; TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}

void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}
```

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

    F32K = (int code *)0xeff5;                  // STC8A8K60S4
    UartInit();
    ES = 1;
    EA = 1;

    P_SW2 = 0x80;
    UartSend(F32K_ADDR >> 8);                   //Read high byte of 32K frequency
    UartSend(F32K_ADDR);                        //Read low byte of 32K frequency

    while (1);
}
```

assembly code

Operating frequency tested at 11.0592MHz

```
CPUIDBASE       EQU     0FDE0H

ID_ADDR         EQU     CPUIDBASE + 00H
VREF_ADDR       EQU     CPUIDBASE + 07H
F32K_ADDR       EQU     CPUIDBASE + 09H
T22M_ADDR       EQU     CPUIDBASE + 0BH     ;22.1184MHz
T24M_ADDR       EQU     CPUIDBASE + 0CH     ;24MHz
T20M_ADDR       EQU     CPUIDBASE + 0DH     ;20MHz
T27M_ADDR       EQU     CPUIDBASE + 0EH     ;27MHz
T30M_ADDR       EQU     CPUIDBASE + 0FH     ;30MHz
T33M_ADDR       EQU     CPUIDBASE + 10H     ;33.1776MHz
T35M_ADDR       EQU     CPUIDBASE + 11H     ;35MHz
T36M_ADDR       EQU     CPUIDBASE + 12H     ;36.864MHz
T40M_ADDR       EQU     CPUIDBASE + 13H     ;40MHz
T45M_ADDR       EQU     CPUIDBASE + 14H     ;45MHz
VRT6M_ADDR      EQU     CPUIDBASE + 15H     ;VRTRIM_6M
VRT10M_ADDR     EQU     CPUIDBASE + 16H     ;VRTRIM_10M
VRT27M_ADDR     EQU     CPUIDBASE + 17H     ;VRTRIM_27M
VRT44M_ADDR     EQU     CPUIDBASE + 18H     ;VRTRIM_44M

AUXR        DATA    8EH
P_SW2       DATA    0BAH
```

BUSY	*BIT*	*20H.0*
P0M1	*DATA*	*093H*
P0M0	*DATA*	*094H*

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

            ORG     0000H
            LJMP    MAIN
            ORG     0023H
            LJMP    UART_ISR

            ORG     0100H

UART_ISR:
            JNB     TI, CHKRI
            CLR     TI
            CLR     BUSY
CHKRI:
            JNB     RI,UARTISR_EXIT
            CLR     RI
UARTISR_EXIT:
            RETI

UART_INIT:
            MOV     SCON,#50H
            MOV     TMOD,#00H
            MOV     TL1,#0E8H       ;65536-
11059200/115200/4=0FFE8H MOV        TH1,#0FFH
            SETB    TR1
            MOV     AUXR,#40H
            CLR     BUSY
            RET

UART_SEND:
            JB      BUSY,$
            SETB    BUSY
            MOV
            SBUF,A RET

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

```
                LCALL   UART_INIT
                SETB    ES
                SETB    EA

                MOV     P_SW2,#80H
                MOV     DPTR,# F32K_ADDR
                CLR     A
                MOVX    A,@DPTR                 Read the high byte of the 32K frequency
                LCALL   UART_SEND
                INC     DPTR
                CLR     A
                MOVX    A,@DPTR                 Read the low byte of the 32K frequency
                LCALL   UART_SEND

LOOP:
                JMP     LOOP

                END
```

7.5.8 Read the 32K power-down wake-up timer frequency (from Flash program memory) (read in ROM)

C code

```
// Tested operating frequency of 11.0592MHz

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

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

sfr       AUXR        =   0x8e;

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

bit       busy;
int       *F32K;

void UartIsr() interrupt 4
{
    if (TI)
```

```
            TI = 0;
            busy = 0;
        }
        if (RI)
        {
            RI = 0;
        }
}

void UartInit()
{
        SCON = 0x50;
        TMOD =
        0x00;  TL1  =
        BRT;
        TH1  =  BRT  >>
        8; TR1 = 1;
        AUXR = 0x40;
        busy = 0;
}

void UartSend(char dat)
{
        while (busy);
        busy = 1;
        SBUF = dat;
}

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

        F32K = (int code *)0xeff5;                  // STC8A8K60S4
        UartInit();
        ES = 1;
        EA = 1;

        UartSend(*F32K >> 8);                       // Read the high byte of the 32K frequency
        UartSend(*F32K);                            // Read the low byte of the 32K frequency

        while (1);
}
```

assembly code

Operating frequency tested at *11.0592MHz*

```
AUXR    DATA    8EH
F32K    EQU     0EFF5H          ; STC8A8K60S4
```

```
BUSY        BIT     20H.0

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

            ORG     0000H
            LJMP    MAIN
            ORG     0023H
            LJMP    UART_ISR

            ORG     0100H

UART_ISR:
            JNB     TI, CHKRI
            CLR     TI
            CLR     BUSY
CHKRI:
            JNB     RI,UARTISR_EXIT
            CLR     RI
UARTISR_EXIT:
            RETI

UART_INIT:
            MOV     SCON,#50H
            MOV     TMOD,#00H
            MOV     TL1,#0E8H           ;65536-
            11059200/115200/4=0FFE8H MOV    TH1,#0FFH
            SETB    TR1
            MOV     AUXR,#40H
            CLR     BUSY
            RET

UART_SEND:
            JB      BUSY,$
            SETB    BUSY
            MOV
            SBUF,A RET

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

```
        MOV     p4m0,  #00h
        MOV     p4m1,  #00h
        MOV     p5m0,  #00h
        MOV     p5m1, #00h

        LCALL   UART_INIT
        SETB    ES
        SETB    EA

        MOV     DPTR,#F32K
        CLR     A
        MOVC    A,@A+DPTR           Read the high byte of the 32K frequency
        LCALL   UART_SEND
        INC     DPTR
        CLR     A
        MOVC    A,@A+DPTR           Read the low byte of the 32K frequency
        LCALL   UART_SEND

LOOP:
        JMP     LOOP

        END
```

7.5.9 ~~Read 32K power-down wake-up timer frequency (from RAM)~~

C code

```
// Tested operating frequency of 11.0592MHz

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

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

sfr       AUXR      =   0x8e;

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

bit       busy;
int       *F32K;

void UartIsr() interrupt 4
{
```

```
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
    }
}

void UartInit()
{
    SCON = 0x50;
    TMOD =
    0x00;  TL1  =
    BRT;
    TH1  =  BRT  >>
    8; TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}

void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}

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

    F32K = (int idata *)0xf8;
    UartInit();
    ES = 1;
    EA = 1;

    UartSend(*F32K >> 8);          // Read the high byte of the 32K frequency
    UartSend(*F32K);               // Read the low byte of the 32K frequency

    while (1);
}
```

assembly code

Operating frequency tested at *11.0592MHz*

```
AUXR    DATA    8EH
```

```
F32K        DATA    0F8H

BUSY        BIT     20H.0

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

            ORG     0000H
            LJMP    MAIN
            ORG     0023H
            LJMP    UART_ISR

            ORG     0100H

UART_ISR:
            JNB     TI, CHKRI
            CLR     TI
            CLR     BUSY
CHKRI:
            JNB     RI,UARTISR_EXIT
            CLR     RI
UARTISR_EXIT:
            RETI

UART_INIT:
            MOV     SCON,#50H
            MOV     TMOD,#00H
            MOV     TL1,#0E8H               ;65536-
            11059200/115200/4=0FFE8H MOV    TH1,#0FFH
            SETB    TR1
            MOV     AUXR,#40H
            CLR     BUSY
            RET

UART_SEND:
            JB      BUSY,$
            SETB    BUSY
            MOV
            SBUF,A RET

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

```
                MOV     P3M1, #00H
                MOV     P4M0, #00H
                MOV     P4M1, #00H
                MOV     P5M0, #00H
                MOV     P5M1, #00H

                LCALL   UART_INIT
                SETB    ES
                SETB    EA

                MOV     R0,#F32K
                MOV     A,@R0                   Read the high byte of the 32K frequency
                LCALL   UART_SEND
                INC     R0
                MOV     A,@R0                   Read the low byte of the 32K frequency
                LCALL   UART_SEND

LOOP:
                JMP     LOOP

                END
```

7.5.10 User-defined internal IRC frequency (read from CHIPID)

C code

```
// Tested operating frequency of 11.0592MHz

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

#define     CLKSEL      (*(unsigned char volatile xdata *)0xfe00)
#define     CLKDIV      (*(unsigned char volatile xdata *)0xfe01)

#define     CPUIDBASE       0xfde0

#define     ID_ADDR         ((unsigned char volatile xdata *)(CPUIDBASE + 0x00))
#define     VREF_ADDR       (*(unsigned int volatile xdata *)(CPUIDBASE + 0x07))
#define     F32K_ADDR       (*(unsigned int volatile xdata *)(CPUIDBASE + 0x09))
#define     T22M_ADDR       (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0b))    //22.1184MHz
#define     T24M_ADDR       (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0c))    //24MHz
#define     T20M_ADDR       (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0d))    //20MHz
#define     T27M_ADDR       (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0e))    //27MHz
#define     T30M_ADDR       (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0f))    //30MHz
#define     T33M_ADDR       (*(unsigned char volatile xdata *)(CPUIDBASE + 0x10))    //33.1776MHz
#define     T35M_ADDR       (*(unsigned char volatile xdata *)(CPUIDBASE + 0x11))    //35MHz
#define     T36M_ADDR       (*(unsigned char volatile xdata *)(CPUIDBASE + 0x12))    //36.864MHz
#define     T40M_ADDR       (*(unsigned char volatile xdata *)(CPUIDBASE + 0x13))    //40MHz
#define     T45M_ADDR       (*(unsigned char volatile xdata *)(CPUIDBASE + 0x14))    //45MHz
#define     VRT6M_ADDR      (*(unsigned char volatile xdata *)(CPUIDBASE + 0x15))    //VRTRIM_6M
#define     VRT10M_ADDR     (*(unsigned char volatile xdata *)(CPUIDBASE + 0x16))    //VRTRIM_10M
#define     VRT27M_ADDR     (*(unsigned char volatile xdata *)(CPUIDBASE + 0x17))    //VRTRIM_27M
#define     VRT44M_ADDR     (*(unsigned char volatile xdata *)(CPUIDBASE + 0x18))    //VRTRIM_44M
```

```
sfr  P_SW2   =  0xba;
sfr  IRCBAND =  0x9d;
```

```
sfr     IRTRIM  =   0x9f;
sfr     VRTRIM  =   0xa6;

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

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

// // Select 20MHz
// P_SW2 = 0x80;
    //CLKDIV = 0x04;
    //IRTRIM = T20M_ADDR;
    //VRTRIM = VRT27M_ADDR;
    //IRCBAND = 0x02;
    //CLKDIV = 0x00;

// // Select 22.1184MHz
// P_SW2 = 0x80;
    //CLKDIV = 0x04;
    //IRTRIM = T22M_ADDR;
    //VRTRIM = VRT27M_ADDR;
    //IRCBAND = 0x02;
    //CLKDIV = 0x00;

    //      Select
    24MHz P_SW2
    = 0x80;
    CLKDIV = 0x04;
    IRTRIM = T24M_ADDR;
    VRTRIM = VRT27M_ADDR;
    IRCBAND = 0x02;
```

```
    CLKDIV = 0x00;

// // Select 27MHz
// P_SW2 = 0x80;
      //CLKDIV = 0x04;
      //IRTRIM = T27M_ADDR;
```

```
    //VRTRIM = VRT27M_ADDR;
    //IRCBAND = 0x02;
    //CLKDIV = 0x00;

//  //Select 30MHz
    //P_SW2 = 0x80;
    //CLKDIV = 0x04;
    //IRTRIM = T30M_ADDR;
    //VRTRIM = VRT27M_ADDR;
    //IRCBAND = 0x02;
    //CLKDIV = 0x00;

//  //Select 33.1776MHz
    //P_SW2 = 0x80;
    //CLKDIV = 0x04;
    //IRTRIM = T33M_ADDR;
    //VRTRIM = VRT27M_ADDR;
    //IRCBAND = 0x02;
    //CLKDIV = 0x00;

//  //Select 35MHz
    //P_SW2 = 0x80;
    //CLKDIV = 0x04;
    //IRTRIM = T35M_ADDR;
    //VRTRIM = VRT44M_ADDR;
    //IRCBAND = 0x03;
    //CLKDIV = 0x00;

//  //Select 40MHz
    //P_SW2 = 0x80;
    //CLKDIV = 0x04;
    //IRTRIM = T40M_ADDR;
    //VRTRIM = VRT44M_ADDR;
    //IRCBAND = 0x03;
    //CLKDIV = 0x00;

//  //Select 45MHz
    //P_SW2 = 0x80;
    //CLKDIV = 0x04;
    //IRTRIM = T45M_ADDR;
    //VRTRIM = VRT44M_ADDR;
    //IRCBAND = 0x03;
    //CLKDIV = 0x00;

    while (1);
}
```

assembly code

Operating frequency tested at 11.0592MHz

```
CPUIDBASE       EQU     0FDE0H

ID_ADDR         EQU     CPUIDBASE + 00H
VREF_ADDR       EQU     CPUIDBASE + 07H
F32K_ADDR       EQU     CPUIDBASE + 09H
T22M_ADDR       EQU     CPUIDBASE + 0BH         ;22.1184MHz
T24M_ADDR       EQU     CPUIDBASE + 0CH         ;24MHz
T20M_ADDR       EQU     CPUIDBASE + 0DH         ;20MHz
```

```
T27M_ADDR       EQU     CPUIDBASE + 0EH     ;27MHz
T30M_ADDR       EQU     CPUIDBASE + 0FH     ;30MHz
T33M_ADDR       EQU     CPUIDBASE + 10H     ;33.1776MHz
T35M_ADDR       EQU     CPUIDBASE + 11H     ;35MHz
T36M_ADDR       EQU     CPUIDBASE + 12H     ;36.864MHz
T40M_ADDR       EQU     CPUIDBASE + 13H     ;40MHz
T45M_ADDR       EQU     CPUIDBASE + 14H     ;45MHz
VRT6M_ADDR      EQU     CPUIDBASE + 15H     ;VRTRIM_6M
VRT10M_ADDR     EQU     CPUIDBASE + 16H     ;VRTRIM_10M
VRT27M_ADDR     EQU     CPUIDBASE + 17H     ;VRTRIM_27M
VRT44M_ADDR     EQU     CPUIDBASE + 18H     ;VRTRIM_44M

P_SW2           DATA    0BAH
CLKSEL          EQU     0FE00H
CLKDIV          EQU     0FE01H

IRCBAND         DATA    09DH
IRCTRIM         DATA    09FH
VRTRIM          DATA    0A6H

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

                ORG     0000H
                LJMP    MAIN

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

;               ; Select 20MHz
;               MOV     P_SW2,#80H
;               MOV     A,#4
;               MOV     DPTR,#CLKDIV
;               MOVX    @DPTR,A
;               MOV     DPTR,#T20M_ADDR
;               CLR     A
```

```
;               MOVX    A,@DPTR
;               MOV     IRTRIM,A
;               MOV     DPTR,#VRT27M_ADDR
;               CLR     A
;               MOVX    A,@DPTR
;               MOV     VRTRIM,A
;               MOV     IRCBAND,#02H
;               MOV     A,#0
;               MOV     DPTR,#CLKDIV
;               MOVX    @DPTR,A
;               MOV     P_SW2,#00H

;               Select 22.1184MHz
;               MOV     P_SW2,#80H
;               MOV     A,#4
;               MOV     DPTR,#CLKDIV
;               MOVX    @DPTR,A
;               MOV     DPTR,#T22M_ADDR
;               CLR     A
;               MOVX    A,@DPTR
;               MOV     IRTRIM,A
;               MOV     DPTR,#VRT27M_ADDR
;               CLR     A
;               MOVX    A,@DPTR
;               MOV     VRTRIM,A
;               MOV     IRCBAND,#02H
;               MOV     A,#0
;               MOV     DPTR,#CLKDIV
;               MOVX    @DPTR,A
;               MOV     P_SW2,#00H

            ; Select 24MHz
                MOV     P_SW2,#80H
                MOV     A,#4
                MOV     DPTR,#CLKDIV
                MOVX    @DPTR,A
                MOV     DPTR,#T24M_ADDR
                CLR     A
                MOVX    A,@DPTR
                MOV     IRTRIM,A
                MOV     DPTR,#VRT27M_ADDR
                CLR     A
                MOVX    A,@DPTR
                MOV     VRTRIM,A
                MOV     IRCBAND,#02H
                MOV     A,#0
                MOV     DPTR,#CLKDIV
                MOVX    @DPTR,A
                MOV     P_SW2,#00H

;               Select 27MHz
;               MOV     P_SW2,#80H
;               MOV     A,#4
;               MOV     DPTR,#CLKDIV
;               MOVX    @DPTR,A
;               MOV     DPTR,#T27M_ADDR
;               CLR     A
;               MOVX    A,@DPTR
;               MOV     IRTRIM,A
```

```
;       MOV     DPTR,#VRT27M_ADDR
;       CLR     A
;       MOVX    A,@DPTR
;       MOV     VRTRIM,A
;       MOV     IRCBAND,#02H
;       MOV     A,#0
;       MOV     DPTR,#CLKDIV
;       MOVX    @DPTR,A
;       MOV     P_SW2,#00H

;       ; Select 30MHz
;       MOV     P_SW2,#80H
;       MOV     A,#4
;       MOV     DPTR,#CLKDIV
;       MOVX    @DPTR,A
;       MOV     DPTR,#T30M_ADDR
;       CLR     A
;       MOVX    A,@DPTR
;       MOV     IRTRIM,A
;       MOV     DPTR,#VRT27M_ADDR
;       CLR     A
;       MOVX    A,@DPTR
;       MOV     VRTRIM,A
;       MOV     IRCBAND,#02H
;       MOV     A,#0
;       MOV     DPTR,#CLKDIV
;       MOVX    @DPTR,A
;       MOV     P_SW2,#00H

;       Select 33.1776 MHz
;       MOV     P_SW2,#80H
;       MOV     A,#4
;       MOV     DPTR,#CLKDIV
;       MOVX    @DPTR,A
;       MOV     DPTR,#T33M_ADDR
;       CLR     A
;       MOVX    A,@DPTR
;       MOV     IRTRIM,A
;       MOV     DPTR,#VRT27M_ADDR
;       CLR     A
;       MOVX    A,@DPTR
;       MOV     VRTRIM,A
;       MOV     IRCBAND,#02H
;       MOV     A,#0
;       MOV     DPTR,#CLKDIV
;       MOVX    @DPTR,A
;       MOV     P_SW2,#00H

;       ; Select 35MHz
;       MOV     P_SW2,#80H
;       MOV     A,#4
;       MOV     DPTR,#CLKDIV
;       MOVX    @DPTR,A
;       MOV     DPTR,#T35M_ADDR
;       CLR     A
;       MOVX    A,@DPTR
;       MOV     IRTRIM,A
;       MOV     DPTR,#VRT44M_ADDR
```

```
;       MOVX    A,@DPTR
;       MOV     VRTRIM,A
;       MOV     IRCBAND,#03H
;       MOV     A,#0
;       MOV     DPTR,#CLKDIV
;       MOVX    @DPTR,A
;       MOV     P_SW2,#00H

;       ; Select 36.864MHz
;       MOV     P_SW2,#80H
;       MOV     A,#4
;       MOV     DPTR,#CLKDIV
;       MOVX    @DPTR,A
;       MOV     DPTR,#T36M_ADDR
;       CLR     A
;       MOVX    A,@DPTR
;       MOV     IRTRIM,A
;       MOV     DPTR,#VRT44M_ADDR
;       CLR     A
;       MOVX    A,@DPTR
;       MOV     VRTRIM,A
;       MOV     IRCBAND,#03H
;       MOV     A,#0
;       MOV     DPTR,#CLKDIV
;       MOVX    @DPTR,A
;       MOV     P_SW2,#00H

;       Select 40MHz
;       MOV     P_SW2,#80H
;       MOV     A,#4
;       MOV     DPTR,#CLKDIV
;       MOVX    @DPTR,A
;       MOV     DPTR,#T40M_ADDR
;       CLR     A
;       MOVX    A,@DPTR
;       MOV     IRTRIM,A
;       MOV     DPTR,#VRT44M_ADDR
;       CLR     A
;       MOVX    A,@DPTR
;       MOV     VRTRIM,A
;       MOV     IRCBAND,#03H
;       MOV     A,#0
;       MOV     DPTR,#CLKDIV
;       MOVX    @DPTR,A
;       MOV     P_SW2,#00H

;       Select 45MHz
;       MOV     P_SW2,#80H
;       MOV     A,#4
;       MOV     DPTR,#CLKDIV
;       MOVX    @DPTR,A
;       MOV     DPTR,#T36M_ADDR
;       CLR     A
;       MOVX    A,@DPTR
;       MOV     IRTRIM,A
;       MOV     DPTR,#VRT45M_ADDR
;       CLR     A
;       MOVX    A,@DPTR
;       MOV     VRTRIM,A
```