STC8F family of Micro-controllers
Reference Manual

Technical support website:  www.STCMCU.com / www.GXWMCU.com
Date of update:  2018/3/20
Contents

1 Overview ...........................................................................................................................................................1
2 Features..................................................................................................................................................................2
  2.1 Features and Prices of STC8A8K64S4A12 family.............................................................................................2
  2.2 Features and Prices of STC8A4K64S2A12 family..........................................................................................4
  2.3 Features and Prices of STC8F2K64S4 family .................................................................................................6
  2.4 Features and Prices of STC8F2K64S4 family .................................................................................................8
  2.5 Features and Prices of STC8F1K08S2 family ...............................................................................................10
  2.6 Features and Prices of STC8H1K08S2A10 family .........................................................................................11
  2.7 Advance notice of STC8H1K64S2A10 family ..............................................................................................14
  2.8 Advance notice of STC8H1K08S2 family .................................................................................................14
  2.9 Advance notice of STC8H04A10 family ......................................................................................................15
  2.10 Advance notice of STC8H04 family ........................................................................................................15
3 Pinouts and pin descriptions ..............................................................................................................................16
  3.1 Pinouts .........................................................................................................................................................16
    3.1.1 STC8A8K64S4A12 family pinouts ..........................................................................................................16
    3.1.2 STC8A4K64S2A12 family pinouts ..........................................................................................................19
    3.1.3 STC8F2K64S4 family pinouts ..................................................................................................................22
    3.1.4 STC8F2K64S2 family pinouts ..................................................................................................................24
    3.1.5 STC8F1K08S2 family pinouts ..................................................................................................................26
    3.1.6 STC8H1K08S2A10 family pinouts ..........................................................................................................26
    3.1.7 GX8S003 family pinouts ......................................................................................................................27
    3.1.8 STC8H1K08S2 family pinouts ..................................................................................................................27
  3.2 Pin descriptions .............................................................................................................................................28
    3.2.1 STC8A8K64S4A12 family pin descriptions ..............................................................................................28
    3.2.2 STC8A4K64S2A12 family pin descriptions ..............................................................................................34
    3.2.3 STC8F2K64S4 family pin descriptions ......................................................................................................40
    3.2.4 STC8F2K64S2 family pin descriptions ......................................................................................................45
  3.3 Function Pin Switch ......................................................................................................................................48
    3.3.1 Function pin related register ..................................................................................................................48
  3.4 Sample Program ..........................................................................................................................................52
    3.4.1 Serial 1 switch .........................................................................................................................................52
    3.4.2 Serial 2 switch .........................................................................................................................................53
    3.4.3 Serial 3 switch .........................................................................................................................................53
    3.4.4 Serial 4 switch .........................................................................................................................................54
    3.4.5 SPI switch ...............................................................................................................................................54
    3.4.6 PWM switch ...........................................................................................................................................55
    3.4.7 PCA/CCP/PWM switch ..........................................................................................................................57
    3.4.8 I2C switch ...............................................................................................................................................58
    3.4.9 Comparator output switch ......................................................................................................................58
3.4.10 Master clock output switching ........................................................................................................... 59

4 Package characteristics........................................................................................................................................ 61
  4.1 LQFP64S package mechanical data (12mm*12mm) ......................................................................................... 61
  4.2 LQFP64L package mechanical data (16mm*16mm) .............................................................................................. 62
  4.3 LQFP48 package mechanical data (9mm*9mm) .................................................................................................. 63
  4.4 LQFP44 package mechanical data (12mm*12mm) .............................................................................................. 66
  4.5 LQFP32 package mechanical data (9mm*9mm) .................................................................................................. 69
  4.6 QFN32 package mechanical data (4mm*4mm) .................................................................................................. 70
  4.7 PDIP40 package mechanical data .................................................................................................................. 71
  4.8 TSSOP20 package mechanical data ............................................................................................................... 72
  4.9 SOP16 package mechanical data .................................................................................................................. 73

5 STC8 series microcontroller selection price list.................................................................................................. 74
  5.1 STC8 series of microcontroller package price list ............................................................................................... 75
  5.2 STC8 series microcontroller name rule ........................................................................................................... 76

6 ISP download and typical application circuit diagram......................................................................................... 77
  6.1 STC8F series ISP download application circuit diagram....................................................................................... 77
  6.1.1 Using RS-232 transferor download ................................................................................................................. 77
  6.1.2 Using PL2303-SA download .......................................................................................................................... 78
  6.1.3 Using U8-Mini tool download .......................................................................................................................... 78
  6.1.4 Using U8W tool download ............................................................................................................................ 79
  6.1.5 USB directly ISP download ........................................................................................................................... 80
  6.2 STC8A series ISP download application circuit diagram................................................................................... 81
  6.2.1 Using RS-232 transfer download(using high-precision ADC)........................................................................ 81
  6.2.2 Using RS-232 transfer download (ADC general application) ........................................................................ 82
  6.2.3 Using PL2303 download ................................................................................................................................. 83
  6.2.4 Using U8-Mini tool download ........................................................................................................................ 84
  6.2.5 Using U8W tool download ............................................................................................................................ 85
  6.2.6 USB directly ISP download ........................................................................................................................... 86

7 clock, reset and power management.................................................................................................................. 87
  7.1 System clock control ......................................................................................................................................... 87
  7.2 System reset ...................................................................................................................................................... 89
  7.3 System power management ............................................................................................................................ 92
  7.4 Sample program ............................................................................................................................................... 93
    7.4.1 Select system clock source ........................................................................................................................... 93
    7.4.2 Master clock frequency division output ....................................................................................................... 95
    7.4.3 Watchdog timer application ......................................................................................................................... 96
    7.4.4 Soft reset to implement custom Downloads ............................................................................................... 96
    7.4.5 Low voltage detection ................................................................................................................................ 97
    7.4.6 Power saving mode ..................................................................................................................................... 98
    7.4.7 Wake-up MCU with INT0/INT1/INT2/INT3/INT4 interrupt ....................................................................... 100
    7.4.8 Wake-up MCU with T0/T1/T2/T3/T4 interrupts ......................................................................................... 102
    7.4.9 Wake-up MCU with RxD/RxD2/RxD3/RxD4 interrupt .................................................................................. 106
    7.4.10 Wake-up MCU with LVD interrupt ........................................................................................................... 108
    7.4.11 Wake-up MCU with CCP0/CCP1/CCP2/CCP3 interrupt ........................................................................... 110
8 Memory .................................................................................................................................................. 118
  8.1 Program Memory .................................................................................................................................. 118
  8.2 Data Memory ....................................................................................................................................... 119
    8.2.1 Internal RAM .................................................................................................................................. 119
    8.2.2 Internal extended RAM ................................................................................................................... 121
    8.2.3 External extended RAM .................................................................................................................. 122
  8.3 Special Parameters in Memory ............................................................................................................. 123
    8.3.1 Read the Bandgap voltage (Read from ROM) ................................................................................. 125
    8.3.2 Read the Bandgap voltage (Read from RAM) ............................................................................... 127
    8.3.3 Read the global unique ID number (Read from ROM) ................................................................. 129
    8.3.4 Read the global unique ID number (Read from RAM) ................................................................. 131
    8.3.5 Read the frequency of the 32K power down wake-up timer (Read from ROM) ....................... 134
    8.3.6 Read the frequency of the 32K power down wake-up timer (Read from RAM) ....................... 136
    8.3.7 Manually set the internal IRC frequency (Read from ROM) ....................................................... 138
    8.3.8 Manually set the internal IRC frequency (Read from RAM) ....................................................... 140

9 Special Function Register ...................................................................................................................... 142
  9.1 Series of STC8A8K64S4A12 .................................................................................................................. 142
  9.2 Series of STC8A4K64S2A12 .................................................................................................................. 143
  9.3 Series of STC8F2K64S4 ....................................................................................................................... 144
  9.4 Series of STC8F2K64S2 ....................................................................................................................... 145
  9.5 List of Special Function Registers .................................................................................................... 146

10 I/O Ports .............................................................................................................................................. 153
  10.1 I/O port related registers .................................................................................................................... 153
  10.2 I/O Ports Configurations .................................................................................................................... 156
  10.3 I/O ports structure ............................................................................................................................. 158
    10.3.1 Quasi-Bidirectional I/O (weak pull-up) ......................................................................................... 158
    10.3.2 Push-Pull Output ......................................................................................................................... 158
    10.3.3 High-Impedance .......................................................................................................................... 159
    10.3.4 Open-Drain Output ...................................................................................................................... 159
  10.4 Instructions about special I/O ports ................................................................................................... 160
    10.4.1 P2.0 / RSTCV ............................................................................................................................... 160
    10.4.2 I/O ports related to PWM ........................................................................................................... 160
  10.5 Sample programs ............................................................................................................................... 160
    10.5.1 Mode configure of the ports ........................................................................................................ 160
    10.5.2 Read and write operations on bidirectional port ....................................................................... 162

11 Instruction Set .................................................................................................................................... 163

12 Interrupt System ................................................................................................................................ 167
  12.1 Interrupt Sources of STC8F family ................................................................................................. 167
    12.1.1 Interrupt Sources of STC8F8K64S4A10 series ......................................................................... 168
    12.1.2 Interrupt Sources of STC8A8K64S4A12 series ........................................................................ 169
    12.1.3 Interrupt Sources of STC8F2K64S4 series .............................................................................. 169
    12.1.4 Interrupt Sources of STC8F2K64S4 series .............................................................................. 169
12.2 Interrupt Structure Diagrams of STC8F family ................................................................. 170
12.3 Interrupt List of STC8F family microcontrollers .............................................................. 171
12.4 Interrupt Related Registers .............................................................................................. 173
   12.4.1 Interrupt Enable Control Registers (interrupt Enable bits) ................................. 174
   12.4.2 Interrupt Request Registers (interrupt flags) ......................................................... 178
   12.4.3 Interrupt Priority Control Registers ......................................................................... 180
12.5 Demo codes ...................................................................................................................... 182
   12.5.1 INTO interrupt (rising and falling edges) ................................................................ 182
   12.5.2 INTO interrupt (falling edge) .................................................................................. 183
   12.5.3 INT1 interrupt (rising and falling edges) ................................................................. 184
   12.5.4 INT1 interrupt (falling edge) .................................................................................. 185
   12.5.5 INT2 interrupt (falling edge) .................................................................................. 186
   12.5.6 INT3 interrupt (falling edge) .................................................................................. 187
   12.5.7 INT4 interrupt (falling edge) .................................................................................. 188
   12.5.8 timer0 interrupt ...................................................................................................... 189
   12.5.9 Timer1 interrupt ...................................................................................................... 190
   12.5.10 Timer2 interrupt .................................................................................................... 191
   12.5.11 Timer3 interrupt .................................................................................................... 192
   12.5.12 Timer4 interrupt .................................................................................................... 193
   12.5.13 UART1 interrupt .................................................................................................. 194
   12.5.14 UART2 interrupt .................................................................................................. 196
   12.5.15 UART3 interrupt .................................................................................................. 197
   12.5.16 UART4 interrupt .................................................................................................. 199
   12.5.17 ADC interrupt ....................................................................................................... 201
   12.5.18 LVD interrupt ....................................................................................................... 202
   12.5.19 PCA interrupt ....................................................................................................... 203
   12.5.20 SPI interrupt ......................................................................................................... 205
   12.5.21 CMP interrupt ....................................................................................................... 206
13 Timer/Counter ...................................................................................................................... 213
   13.1 Timer Related Registers ............................................................................................... 213
   13.2 Timer 0/1 ...................................................................................................................... 214
   13.3 Timer 2 ......................................................................................................................... 217
   13.4 Timer 3/4 ...................................................................................................................... 217
   13.5 Power-Down Wake-Up Special Timer .......................................................................... 219
   13.6 Demo code .................................................................................................................. 220
      13.6.1 Timer 0 (Automatic reloading of mode 0—16 bit) .................................................. 220
      13.6.2 Timer 0 (Non automatic reloading of mode 1—16 bit) ......................................... 221
      13.6.3 Timer 0 (Automatic reloading of mode 2—8 bit) .................................................. 222
      13.6.4 Timer 0 (Automatic reloading of non-maskable interrupt for mode 2—8 bit) ....... 223
      13.6.5 Timer 0 (External counting—set T0 as external interrupt of falling edge) ............ 224
      13.6.6 Timer 0 (Test pulse width—the width of high level for INTO) ............................... 225
      13.6.7 Timer 0 (Clock divider output) ............................................................................. 226
      13.6.8 Timer 1 (Automatic reloading of mode 0—16 bit) .................................................. 227
      13.6.9 Timer 1 (Non automatic reloading of mode 1—16 bit) ......................................... 228
13.6.10 Timer 1(Automatic reloading of mode 2—8 bit) ................................................................. 229
13.6.11 Timer 1(External counting—set T1 as external interrupt of falling edge) .................. 230
13.6.12 Timer 1(Test pulse width—the width of high level for INT1) ........................................ 231
13.6.13 Timer 1(clock divider output) ....................................................................................... 232
13.6.14 Configure Timer 1(mode 0) as Baud Rate Generator of serial port 1 .................... 233
13.6.15 Configure Timer 1(mode 2) as Baud Rate Generator of serial port 1 .................... 236
13.6.16 Timer 2 (Automatic reloading for 16 bits) ................................................................. 239
13.6.17 Timer 2 (External counting—set T2 as external interrupt of falling edge) ............ 240
13.6.18 Timer 2(clock divider output) ....................................................................................... 241
13.6.19 Configure Timer 2 as Baud Rate Generator of serial port 1 ...................................... 242
13.6.20 Configure Timer 2 as Baud Rate Generator of serial port 2 ...................................... 245
13.6.21 Configure Timer 2 as Baud Rate Generator of serial port 3 ...................................... 248
13.6.22 Configure Timer 2 as Baud Rate Generator of serial port 4 ...................................... 251
13.6.23 Timer 3(Automatic reloading for 16 bits) .................................................................... 255
13.6.24 Timer 3(External counting—set T3 as the external interrupt for falling edge) ......... 256
13.6.25 Timer 3(clock divider output) ....................................................................................... 257
13.6.26 Configure Timer 3 as Baud Rate Generator of serial port 3 ...................................... 258
13.6.27 Timer 4(Automatic reloading for 16 bits) .................................................................... 261
13.6.28 Timer 4(External counting—set T4 as the external interrupt for falling edge) ......... 262
13.6.29 Timer 4(clock divider output) ....................................................................................... 264
13.6.30 Configure Timer 4 as Baud Rate Generator of serial port 4 ....................................... 264

14 Serial Port (UART) Communication .................................................................................. 268
14.1 Serial Port Related Registers ......................................................................................... 268
14.2 Serial Port 1 .................................................................................................................. 268
14.2.1 Serial Port 1 Mode 0 ................................................................................................. 270
14.2.2 Serial Port 1 Mode 1 ................................................................................................. 272
14.2.3 Serial Port 1 Mode 2 ................................................................................................. 274
14.2.4 Serial Port 1 Mode 3 ................................................................................................. 275
14.2.5 Automatic Address Recognition ............................................................................. 276
14.3 Serial Port 2 .................................................................................................................. 277
14.3.1 Serial Port 2 Mode 0 ................................................................................................. 278
14.3.2 Serial Port 2 Mode 1 ................................................................................................. 279
14.4 Serial Port 3 .................................................................................................................. 279
14.4.1 Serial Port 3 Mode 0 ................................................................................................. 280
14.4.2 Serial Port 3 Mode 1 ................................................................................................. 281
14.5 Serial Port 4 .................................................................................................................. 282
14.5.1 Serial Port 4 Mode 0 ................................................................................................. 283
14.5.2 Serial Port 4 Mode 1 ................................................................................................. 284
14.6 Precautions for Serial Port ............................................................................................ 285
14.7 Demo code .................................................................................................................... 286
14.7.1 Serial port 1 use timer 2 as Baud Rate Generator ...................................................... 286
14.7.2 Serial port 1 use timer 1 as Baud Rate Generator(MODE 0) ...................................... 289
14.7.3 Serial port 1 use timer 1 as Baud Rate Generator(MODE 2) ...................................... 292
14.7.4 Serial port 2 use timer 2 as Baud Rate Generator ...................................................... 295
14.7.5 Serial port 3 use timer 2 as Baud Rate Generator ......................................................... 298
14.7.6 Serial port 3 use timer 3 as Baud Rate Generator .......................................................... 301
14.7.7 Serial port 4 use timer 2 as Baud Rate Generator .......................................................... 304
14.7.8 Serial port 4 use timer 4 as Baud Rate Generator .......................................................... 307

15 Comparator,brown-out detection,internal fixed comparison voltage ........................................... 311
  15.1 Internal Structure of Comparator .................................................................................. 311
  15.2 Registers related to comparator ..................................................................................... 311
  15.3 Demo code .................................................................................................................... 313
    15.3.1 Use of comparators(interrupt way) ............................................................................ 313
    15.3.2 Use of comparators(search way) ............................................................................... 315
    15.3.3 Configure Comparator as External Brown-out Detection ......................................... 317
    15.3.4 Comparator detects operating voltage (battery voltage) ............................................. 319

16 IAP/EEPROM ..................................................................................................................... 323
  16.1 EEPROM Related Registers .......................................................................................... 323
  16.2 Important Notes on EEPROM Programming and Erase Waiting Time ......................... 325
  16.3 Demo code .................................................................................................................... 326
    16.3.1 Basic operation for EEPROM ................................................................................. 326
    16.3.2 Using the Serial Port to Send EEPROM Data ......................................................... 329

17 Analog to Digital Converter (ADC) ........................................................................................ 333
  17.1 ADC Related Registers ................................................................................................ 333
  17.2 ADC Typical application circuit diagram ........................................................................ 335
    17.2.1 High precision ADC application ............................................................................ 335
    17.2.2 ADC General Application (Applications with Low Accuracy ADC Requirements) .... 336
  17.3 Sample program ............................................................................................................ 336
    17.3.1 ADC basic operation (Query Mode) ........................................................................ 336
    17.3.2 ADC basic operation(Interrupt Mode) ................................................................. 338
    17.3.3 Format the ADC conversion result ........................................................................ 339
    17.3.4 Using ADC Channel 16 to Measure External Voltage or Battery Voltage ............... 341

18 Application of PCA/CCP/PWM application .......................................................................... 344
  18.1 PCA Related register .................................................................................................. 344
  18.2 PCA Operation Mode ................................................................................................ 347
    18.2.1 Capture Mode ....................................................................................................... 347
    18.2.2 Software Timer Mode ........................................................................................... 348
    18.2.3 High Speed pulse Output Mode ............................................................................. 348
    18.2.4 Pulse Width Modulator Mode (PWM mode) ......................................................... 349
  18.3 Sample program .......................................................................................................... 352
    18.3.1 PCA outputs PWM(6/7/8/10 bit) ......................................................................... 352
    18.3.2 PCA Capture measurement pulse width ............................................................... 354
    18.3.3 PCA implements 16-bit software timing ............................................................... 357
    18.3.4 PCA Output high speed pulse .............................................................................. 360
    18.3.5 PCA extends external interrupt ............................................................................ 362

19 Enhanced PWM .................................................................................................................. 365
  19.1 PWM Related Registers ............................................................................................. 365
  19.2 Sample program ......................................................................................................... 371
19.2.1 Output waveforms of any period and arbitrary duty cycle ....................................................... 371
19.2.2 Two PWMs Complementary Symmetric Waveform with Dead-time Control ................. 374
19.2.3 PWM to achieve gradient light (breath light) ................................................................. 377

20 Synchro nous Serial Peripheral Interface (SPI) .............................................................................. 383
  20.1 SPI Related Registers ........................................................................................................... 383
  20.2 SPI Communication Modes ............................................................................................... 384
    20.2.1 Single Master and Single Slave Mode ................................................................. 384
    20.2.2 Dual Devices Configuration Mode ............................................................................. 385
    20.2.3 Single Master and Multiple Slaves Mode .............................................................. 385
  20.3 SPI Configuration ................................................................................................................. 386
  20.4 Data Pattern ......................................................................................................................... 388
  20.5 Sample program .................................................................................................................. 389
    20.5.1 SPI Single Master and Single Slave System host program(interrupt mode) .............. 389
    20.5.2 SPI Single Master and Single Slave System slave program(Single Master and Single Slave) ................................................................. 391
    20.5.3 SPI Single Master and Single Slave System host program(Query mode) ............... 392
    20.5.4 SPI Single Master and Single Slave System host program(Query mode) ............... 393
    20.5.5 SPI Mutual master-slave system program(interrupt mode) ....................................... 394
    20.5.6 SPI Mutual master-slave system program(Query mode) ........................................... 396

21 I²C Bus ........................................................................................................................................... 399
  21.1 I²C Related Registers .......................................................................................................... 399
  21.2 I²C Master Mode .................................................................................................................. 399
  21.3 I²C Slave Mode ..................................................................................................................... 403
  21.4 Sample program .................................................................................................................. 405
    21.4.1 I²C master mode access AT24C256(interrupt mode) .............................................. 405
    21.4.2 I²C master mode access AT24C256(Query mode) .................................................. 411
    21.4.3 I²C master mode access PCF8563 ............................................................................ 415
    21.4.4 I²C Slave Mode(interrupt mode) ............................................................................ 420
    21.4.5 I²C Slave Mode(Query mode) ................................................................................. 424
    21.4.6 Host code for testing I2C slave mode code ............................................................... 427

22 Enhanced double data pointer ...................................................................................................... 432
  22.1 Sample program .................................................................................................................. 434
    22.1.1 Example code 1 ........................................................................................................ 434
    22.1.2 Example code 2 ........................................................................................................ 434

Appendix A Application Considerations ...................................................................................... 436
Appendix B STC Guide to the use of simulators .............................................................................. 445
Appendix C STC-USB Driver installation instructions ..................................................................... 449
Appendix D Electrical Character ................................................................................................... 495
Appendix E Update Log ............................................................................................................... 498
1 Overview

STC8F family of MCUs are single clock/machine cycle (which is also called 1T) microcontrollers produced by STC Co. Ltd. It is a new generation of 8051 core MCU with wide voltage range, high speed, high reliability, low power and super strong anti-interference. STC8F family of MCUs use STC ninth generation encryption technology so that they can not be decrypted. They have a fully compatible instruction set with traditional 8051 family of microcontroller. With the enhanced kernel, STC8F family of MCUs are faster than the traditional 8051 MCU at about 11.2~13.2 times.

High precision of ±0.3% R/C clock is integrated in MCU with ±1% temperature drift under the temperature range of -40°C to +85°C, and ±0.6% temperature drift under normal temperature range from -20°C to +65°C. The frequency of RC clock can be set from 5MHz to 30MHz when programming a MCU using ISP. Moreover, high reliable reset circuit with 4 level optional reset threshold voltage is integrated in MCU. So, external expensive crystal and the external reset circuit can be eliminated completely.

There are three optional clock sources inside the MCU, internal 24MHz high precision IRC, internal 32KHz low speed IRC, external 4MHz~33MHz oscillator or external clock signal. The clock source can be freely chosen in the user code. After the clock source is selected, it can be 8-bit divided freely, and then be supplied to the CPU and the peripherals.

Two low power modes are provided in MCU: the IDLE mode and the STOP mode. In IDLE mode, CPU stops executing instructions, but all peripherals are still working. At this moment, the power consumption is about 1.5mA at 6MHz working frequency. The STOP mode is the power off mode. At this moment, the CPU and all peripherals stop working, and the power consumption can be reduced to about 0.1μA.

Rich digital peripherals and analog peripherals are provided in MCU, including 4 serial ports, 5 timers, 4 sets of PCA, 8 groups of enhanced PWM and I2C, SPI, 16 channels 12 bit ADC and comparator, which can meet almost all the needs of users when designing a product.

The enhanced dual data pointers are integrated in the STC8F family of microcontrollers. Using program control, the function of automatic increasing or decreasing of data pointer and automatic switching of two sets of data pointers can be realized.

<table>
<thead>
<tr>
<th>Product</th>
<th>UART</th>
<th>Timers</th>
<th>ADC</th>
<th>Enhanced PWM</th>
<th>PCA</th>
<th>Comparator</th>
<th>I²C</th>
<th>SPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>STC8F8K64S4A10</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>STC8A8K64S4A12</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>STC8F2K64S4</td>
<td>●</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>
2 Features

2.1 Features and Prices of STC8A8K64S4A12 family

<table>
<thead>
<tr>
<th>Microcontroller Model</th>
<th>Operating Voltage(V)</th>
<th>Operating Temperature°C</th>
<th>Flash Program Memory (bytes)</th>
<th>Large Capacity Expansion SRAM (bytes)</th>
<th>I/O maximum number</th>
<th>Serial Ports</th>
<th>Power-down wake-up</th>
<th>Timer/Counter(External Power-down Wake-up)</th>
<th>16-bit Enhanced PWM(Dual compare, external interrupt)</th>
<th>PC/ACC/PWM(Max in external interrupt)</th>
<th>Power Down wake-up timer</th>
<th>Open mode</th>
<th>Support USB download</th>
<th>Support RS485 download</th>
<th>ISP</th>
<th>Online simulation</th>
<th>OPN48 (8mm)</th>
<th>LQFP48</th>
<th>LQFP44</th>
<th>PDIP40</th>
</tr>
</thead>
<tbody>
<tr>
<td>STC8A8K16S4A12</td>
<td>2.0-5.5</td>
<td>-40°C~85°C</td>
<td>16K</td>
<td>2</td>
<td>4</td>
<td>Yes</td>
<td>Yes</td>
<td>5</td>
<td>8</td>
<td>4</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>93.4</td>
<td>93.2</td>
<td>93.2</td>
<td>-</td>
</tr>
<tr>
<td>STC8A8K32S4A12</td>
<td>2.0-5.5</td>
<td>-40°C~85°C</td>
<td>32K</td>
<td>2</td>
<td>4</td>
<td>Yes</td>
<td>Yes</td>
<td>5</td>
<td>8</td>
<td>4</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>93.6</td>
<td>93.3</td>
<td>93.3</td>
<td>-</td>
</tr>
<tr>
<td>STC8A8K60S4A12</td>
<td>2.0-5.5</td>
<td>-40°C~85°C</td>
<td>60K</td>
<td>2</td>
<td>4</td>
<td>Yes</td>
<td>Yes</td>
<td>5</td>
<td>8</td>
<td>4</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>93.8</td>
<td>93.4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>STC8A8K64S4A12</td>
<td>2.0-5.5</td>
<td>-40°C~85°C</td>
<td>64K</td>
<td>2</td>
<td>AP</td>
<td>Yes</td>
<td>Yes</td>
<td>5</td>
<td>8</td>
<td>4</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>93.8</td>
<td>93.4</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- Core
  - Enhanced 8051 Core with single clock per machine cycle (1T)
  - Fully compatible instruction set with traditional 8051
  - 22 interrupt sources and 4 interrupt priority levels
  - Online debugging is supported

- Operating voltage
  - 2.0 to 5.5V
  - Built-in LDO

- Operating temperature
  - -40°C~85°C

- Flash memory
  - Up to 64Kbytes of Flash memory to be used to store user code
  - Configurable EEPROM size, 512bytes single page erased, can be repeatedly erased more than 100 thousand times.
  - In-System-Programming, ISP in short, can be used to update the application code, no need for programmer.
  - Online debugging with single chip is supported, and no emulator is needed. The number of breakpoints is unlimited theoretically.
✓ SRAM
   - 128 bytes internal direct access RAM
   - 128 bytes internal indirect access RAM
   - 8192 bytes internal extended RAM
   - RAM expandable externally up to 64 Kbytes

✓ Clock
   - Internal 24MHz high precise R/C clock IRC
     - Error: ±0.3%
     - Temperature drift: ±1.0% at the temperature range of -40°C to 85°C and ±0.6% at the temperature range of -20°C to 65°C
   - Internal 32KHz low speed IRC with large error
   - External 4MHz–33MHz oscillator or external clock
     The three clock source above can be selected freely by used code.

✓ Reset
   - Hardware reset
     - Power-on reset
     - Reset by reset pin with high reset pulse
     - Watch dog timer reset
     - Low voltage detection reset. 4 low voltage detection levels are provided, 2.2V, 2.4V, V2.7, V3.0
   - Software reset
     - Writing the reset trigger register using software

✓ Interrupts
   - 22 interrupt sources: INT0, INT1, INT2, INT3, INT4, timer0, timer1, timer2, timer3, timer4, uart1, uart2, uart3, uart4, ADC, LVD, PCA/CCP, SPI, I^2C, comparator, enhanced PWM, enhanced PWM fault detection
   - 4 interrupt priority levels

✓ Digital peripherals
   - 5 16-bit timers: timer0, timer1, timer2, timer3, timer4. Where the mode 3 of timer0 has the Non Maskable Interrupt (NMI in short) function. Mode 0 of timer0 and timer1 is 16-bit Auto-reload mode.
   - 4 high speed UARTs: uart1, uart2,uart3, uart4, whose baud rate clock source may be fast as FOSEC/4
   - 4 groups of PCA: CCP0, CCP1, CCP2, CCP3, which can be used as capture, high speed output and 6-bits, 7-bits, 8-bits or 10-bits PWM
   - 8 groups of 15 bit enhanced PWM. Control signal with dead zone can be realized, and external fault detection function is supported.
   - SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
   - I^2C: Master mode or slave mode are supported.

✓ Analog peripherals
   - ADC: 16 channels 10 bit ADC
   - Comparator

✓ GPIO
   - Up to 62 GPIOs: P0.0–P0.7, P1.0–P1.7, P2.0–P2.7, P3.0–P3.7, P4.0–P4.7, P5.0–P5.5,
P6.0~P6.7, P7.0~P7.7

- 4 modes for all GPIOs: quasi-bidirectional mode, push-pull output mode, open drain mode, high-impedance input mode

- Package
  - LQFP64, LQFP48, LQFP44

### 2.2 Features and Prices of STC8A4K64S2A12 family

- **Features**
  - Enhanced 8051 Core with single clock per machine cycle (1T)
  - Fully compatible instruction set with traditional 8051
  - 22 interrupt sources and 4 interrupt priority levels
  - Online debugging is supported
  - Operating voltage
    - 2.0 to 5.5V
  - Built-in LDO
  - Operating temperature
    - -40°C~85°C
  - Flash memory
    - Up to 64Kbytes of Flash memory to be used to store user code
    - Configurable EEPROM size, 512bytes single page erased, can be repeatedly erased more than 100 thousand times.
    - In-System-Programming, ISP in short, can be used to update the application code, no need for programmer.

<table>
<thead>
<tr>
<th>Microcontroller Model</th>
<th>Operating Voltage(V)</th>
<th>Flash Program Memory</th>
<th>Large Capacity Expansion SRAM</th>
<th>Powerful Dual Timer</th>
<th>Watch Dog Timer</th>
<th>16Bit Enhanced PWM</th>
<th>15 High Speed ADC</th>
<th>Internal Brownout Detection</th>
<th>Power Down Wake-Up Timer</th>
<th>Power Down Wake-Up</th>
<th>External Clock Output and Reset</th>
<th>IAP Support</th>
<th>Large Supply</th>
<th>Online Simulation</th>
<th>Support USB download</th>
<th>Modern Chips Type</th>
<th>LQFP64/S</th>
<th>LQFP48</th>
<th>LQFP44</th>
<th>PDIP40</th>
</tr>
</thead>
<tbody>
<tr>
<td>STC8A4K16S2A12</td>
<td>2.0-5.5</td>
<td>16K</td>
<td>4K</td>
<td>2</td>
<td>Yes</td>
<td>Yes</td>
<td>5</td>
<td>-</td>
<td>8</td>
<td>4</td>
<td>Yes</td>
<td>Yes</td>
<td>3.1</td>
<td>82.9</td>
<td>83.3</td>
<td>OPA4-6Pin</td>
<td>OPN4-6Pin</td>
<td>LQFP48</td>
<td>LQFP44</td>
<td>PDIP40</td>
</tr>
<tr>
<td>STC8A4K32S2A12</td>
<td>2.0-5.5</td>
<td>32K</td>
<td>4K</td>
<td>2</td>
<td>Yes</td>
<td>Yes</td>
<td>5</td>
<td>-</td>
<td>8</td>
<td>4</td>
<td>Yes</td>
<td>Yes</td>
<td>3.3</td>
<td>83.0</td>
<td>83.0</td>
<td>OPN4-8Pin</td>
<td>LQFP48</td>
<td>LQFP44</td>
<td>PDIP40</td>
<td></td>
</tr>
<tr>
<td>STC8A4K60S2A12</td>
<td>2.0-5.5</td>
<td>60K</td>
<td>4K</td>
<td>2</td>
<td>Yes</td>
<td>Yes</td>
<td>5</td>
<td>-</td>
<td>8</td>
<td>4</td>
<td>Yes</td>
<td>Yes</td>
<td>3.6</td>
<td>83.2</td>
<td>83.2</td>
<td>OPN4-8Pin</td>
<td>LQFP48</td>
<td>LQFP44</td>
<td>PDIP40</td>
<td></td>
</tr>
<tr>
<td>STC8A4K64S2A12</td>
<td>2.0-5.5</td>
<td>64K</td>
<td>4K</td>
<td>2</td>
<td>Yes</td>
<td>Yes</td>
<td>5</td>
<td>-</td>
<td>8</td>
<td>4</td>
<td>Yes</td>
<td>Yes</td>
<td>3.6</td>
<td>83.2</td>
<td>83.2</td>
<td>OPN4-8Pin</td>
<td>LQFP48</td>
<td>LQFP44</td>
<td>PDIP40</td>
<td></td>
</tr>
</tbody>
</table>
Online debugging with single chip is supported, and no emulator is needed. The number of breakpoints is unlimited theoretically.

**SRAM**
- 128 bytes internal direct access RAM
- 128 bytes internal indirect access RAM
- 8192 bytes internal extended RAM
- RAM expandable externally up to 64 Kbytes

**Clock**
- Internal 24MHz high precise R/C clock IRC
  - Error: ±0.3%
  - Temperature drift: ±1.0% at the temperature range of -40°C to 85°C and ±0.6% at the temperature range of -20°C to 65°C
- Internal 32KHz low speed IRC with large error
- External 4MHz–33MHz oscillator or external clock
  The three clock source above can be selected freely by used code.

**Hardware reset**
- Power-on reset
- Reset by reset pin with high reset pulse
- Watch dog timer reset
- Low voltage detection reset. 4 low voltage detection levels are provided, 2.2V, 2.4V, V2.7, V3.0

**Software reset**
- Writing the reset trigger register using software

**Interrupts**
- 22 interrupt sources: INT0, INT1, INT2, INT3, INT4, timer0, timer1, timer2, timer3, timer4, uart1, uart2, uart3, uart4, ADC, LVD, PCA/CCP, SPI, I²C, comparator, enhanced PWM, enhanced PWM fault detection
- 4 interrupt priority levels

**Digital peripherals**
- 5 16-bit timers: timer0, timer1, timer2, timer3, timer4. Where the mode 3 of timer0 has the Non Maskable Interrupt (NMI in short) function. Mode 0 of timer0 and timer1 is 16-bit Auto-reload mode.
- 4 high speed UARTs: uart1, uart2, uart3, uart4, whose baud rate clock source may be fast as FOSC/4
- 4 groups of PCA: CCP0, CCP1, CCP2, CCP3, which can be used as capture, high speed output and 6-bits, 7-bits, 8-bits or 10-bits PWM
- 8 groups of 15 bit enhanced PWM. Control signal with dead zone can be realized, and external fault detection function is supported.
- SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
- I²C: Master mode or slave mode are supported.

**Analog peripherals**
- ADC: 16 channels 12 bit ADC
- Comparator

**GPIO**
- Up to 59 GPIOs: P0.0~P0.7, P1.0~P1.7, P2.0~P2.7, P3.0~P3.7, P4.0~P4.4, P5.0~P5.5, P6.0~P6.7, P7.0~P7.7
- 4 modes for all GPIOs: quasi-bidirectional mode, push-pull output mode, open drain mode, high-impedance input mode

- **Package**
  - LQFP64, LQFP48, LQFP44

## 2.3 Features and Prices of STC8F2K64S4 family

### Prices of different selections

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>STC8F2K16S4</td>
<td>2.0-5.5</td>
<td>16K</td>
<td>2K</td>
<td>48K</td>
<td>42</td>
<td>4</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>LQFP48</td>
<td></td>
</tr>
<tr>
<td>STC8F2K32S4</td>
<td>2.0-5.5</td>
<td>32K</td>
<td>2K</td>
<td>32K</td>
<td>42</td>
<td>4</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>LQFP48</td>
<td></td>
</tr>
<tr>
<td>STC8F2K60S4</td>
<td>2.0-5.5</td>
<td>60K</td>
<td>2K</td>
<td>4K</td>
<td>42</td>
<td>4</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>LQFP48</td>
<td></td>
</tr>
<tr>
<td>STC8F2K64S4</td>
<td>2.0-5.5</td>
<td>64K</td>
<td>2K</td>
<td>1AP</td>
<td>42</td>
<td>4</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>LQFP48</td>
<td></td>
</tr>
</tbody>
</table>

### Core
- Enhanced 8051 Core with single clock per machine cycle (1T)
- Fully compatible instruction set with traditional 8051
- 19 interrupt sources and 4 interrupt priority levels
- Online debugging is supported

### Operating voltage
- 2.0 to 5.5V
- Built-in LDO

### Operating temperature
- -40°C~85°C

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

**SRAM**
- 128 bytes internal direct access RAM
- 128 bytes internal indirect access RAM
- 2048 bytes internal extended RAM
- RAM expandable externally up to 64 Kbytes

**Clock**
- Internal 24MHz high precise R/C clock IRC
  - Error: ±0.3%
  - Temperature drift: ±1.0% at the temperature range of -40°C to 85°C and ±0.6% at the temperature range of -20°C to 65°C
- Internal 32KHz low speed IRC with large error
- External 4MHz~33MHz oscillator or external clock

The three clock source above can be selected freely by used code.

**Reset**
- Hardware reset
  - Power-on reset
  - Reset by reset pin with high reset pulse
  - Watch dog timer reset
  - Low voltage detection reset. 4 low voltage detection levels are provided, 2.2V, 2.4V, V2.7, V3.0
- Software reset
  - Writing the reset trigger register using software

**Interrupts**
- 19 interrupt sources: INT0, INT1, INT2, INT3, INT4, timer0, timer1, timer2, timer3, timer4, uart1, uart2, uart3, uart4, LVD, PCA/CCP, SPI, I²C, comparator
- 4 interrupt priority levels

**Digital peripherals**
- 5 16-bit timers: timer0, timer1, timer2, timer3, timer4. Where the mode 3 of timer0 has the Non Maskable Interrupt (NMI in short) function. Mode 0 of timer0 and timer1 is 16-bit Auto-reload mode.
- 4 high speed UARTs: uart1, uart2,uart3, uart4, whose baud rate clock source may be fast as FOSC/4
- 4 groups of PCA: CCP0, CCP1, CCP2, CCP3, which can be used as capture, high speed output and 6-bits, 7-bits, 8-bits or 10-bits PWM
- SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
- I²C: Master mode or slave mode are supported.

**Analog peripherals**
- Comparator

**GPIO**
- Up to 42 GPIOs: P0.0–P0.7, P1.0–P1.7, P2.0–P2.7, P3.0–P3.7, P4.0–P4.7, P5.4–P5.5
- 4 modes for all GPIOs: quasi-bidirectional mode, push-pull output mode, open drain mode,
high-impedance input mode

- Package
- LQFP44, PDIP40

## 2.4 Features and Prices of STC8F2K64S4 family

### Prices of different selections

| Microcontroller Model | Operating Voltage(V) | Flash Program Memory | Large Capacity Expansion SRAM bytes | Powerful dual DPTR Increase or Decrease | Enhanced SPI | 10 Enhanced PWM(Dead Zone Control) | 16 bits advanced PWM Timers | PC, AC, CCP PWM can be external interrupt | 15 high-speed ADC(PWM can be dual A/D) | Comparator(14-bit, A-D, external brownout detection) | 15 High-speed ADC(PWM can be dual A/D) | Power-down wake-up timer | Timer/Counter(External Pow-down Wake-up) | 16 bits Enhanced PWM(Dead Zone Control) | Watchdog Reset timer | Internal Reset(optional reset threshold vol) | Internal Low-vol Detection interrupt Pow-wk | Internal Clock(24MHz Adjustable) | External clock output and reset | Program encrypted transmission | Set password for next update procedure | Support RS485 download | Support USB download | Online simulation | Start Supply |
|------------------------|----------------------|----------------------|-------------------------------------|----------------------------------------|-------------|-----------------------------------|-------------------------------|-----------------------------------------|----------------------------------------|------------------------------------------------|----------------------------------------|----------------------------------------|----------------------------------------|---------------------------------|-------------------------------|----------------------------------------|----------------------------------------|----------------------------------------|----------------------------------------|----------------------------------------|----------------------------------------|----------------------------------------|----------------------------------------|----------------------------------------|
| STC8F2K16S4            | 2.0-5.5              | 16K                  | 2                                  | 48K                                    | 4           | Yes                               | Yes                          | Yes                                     | 5                                      | -                                      | -                                    | -                                    | -                                    | -                                | -                              | No                            | No                                    | No                                    | No                                    | No                                    | No                                    | No                                    | No                                    | No                                    | No                                    | No                                    |
| STC8F2K32S4            | 2.0-5.5              | 32K                  | 2                                  | 32K                                    | 4           | Yes                               | Yes                          | Yes                                     | 5                                      | -                                      | -                                    | -                                    | -                                    | -                                | -                              | No                            | No                                    | No                                    | No                                    | No                                    | No                                    | No                                    | No                                    | No                                    | No                                    | No                                    |
| STC8F2K60S4            | 2.0-5.5              | 60K                  | 2                                  | 4K                                     | 4           | Yes                               | Yes                          | Yes                                     | 5                                      | -                                      | -                                    | -                                    | -                                    | -                                | -                              | No                            | No                                    | No                                    | No                                    | No                                    | No                                    | No                                    | No                                    | No                                    | No                                    | No                                    |
| STC8F2K64S4            | 2.0-5.5              | 64K                  | 2                                  | 1AP                                    | 4           | Yes                               | Yes                          | Yes                                     | 5                                      | -                                      | -                                    | -                                    | -                                    | -                                | -                              | No                            | No                                    | No                                    | No                                    | No                                    | No                                    | No                                    | No                                    | No                                    | No                                    | Yes                                   |

- Core
  - Enhanced 8051 Core with single clock per machine cycle (1T)
  - Fully compatible instruction set with traditional 8051
  - 19 interrupt sources and 4 interrupt priority levels
  - Online debugging is supported

- Operating voltage
  - 2.0 to 5.5V
  - Built-in LDO

- Operating temperature
  - -40°C~85°C

- Flash memory
  - Up to 64Kbytes of Flash memory to be used to store user code
  - Configurable EEPROM size, 512bytes single page erased, can be repeatedly erased more than 100 thousand times.
  - In-System-Programming, ISP in short, can be used to update the application code, no need for programmer.
  - Online debugging with single chip is supported, and no emulator is needed. The number of breakpoints is unlimited theoretically.
✓ SRAM
  ✓ 128 bytes internal direct access RAM
  ✓ 128 bytes internal indirect access RAM
  ✓ 2048 bytes internal extended RAM
  ✓ RAM expandable externally up to 64 Kbytes

✓ Clock
  ✓ Internal 24MHz high precise R/C clock IRC
    ✴ Error: ±0.3%
    ✴ Temperature drift: ±1.0% at the temperature range of -40°C to 85°C and ±0.6% at the temperature range of -20°C to 65°C
  ✓ Internal 32KHz low speed IRC with large error
  ✓ External 4MHz–33MHz oscillator or external clock
  The three clock source above can be selected freely by used code.

✓ Reset
  ✓ Hardware reset
    ✴ Power-on reset
    ✴ Reset by reset pin with high reset pulse
    ✴ Watch dog timer reset
    ✴ Low voltage detection reset. 4 low voltage detection levels are provided, 2.2V, 2.4V, V2.7, V3.0
  ✓ Software reset
    ✴ Writing the reset trigger register using software

✓ Interrupts
  ✓ 19 interrupt sources: INT0, INT1, INT2, INT3, INT4, timer0, timer1, timer2, timer3, timer4, uart1, uart2, uart3, uart4, LVD, PCA/CCP, SPI, I2C, comparator
  ✓ 4 interrupt priority levels

✓ Digital peripherals
  ✓ 5 16-bit timers: timer0, timer1, timer2, timer3, timer4. Where the mode 3 of timer0 has the Non Maskable Interrupt (NMI in short) function. Mode 0 of timer0 and timer1 is 16-bit Auto-reload mode.
  ✓ 4 high speed UARTs: uart1, uart2,uart3, uart4, whose baud rate clock source may be fast as FOSC/4
  ✓ 4 groups of PCA: CCP0, CCP1, CCP2, CCP3, which can be used as capture, high speed output and 6-bits, 7-bits, 8-bits or 10-bits PWM
  ✓ SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
  ✓ I2C: Master mode or slave mode are supported.

✓ Analog peripherals
  ✓ Comparator

✓ GPIO
  ✓ Up to 42 GPIOs: P0.0–P0.7, P1.0–P1.7, P2.0–P2.7, P3.0–P3.7, P4.0–P4.7, P5.4–P5.5
  ✓ 4 modes for all GPIOs: quasi-bidirectional mode, push-pull output mode, open drain mode, high-impedance input mode

✓ Package
  ✓ LQFP44, PDIP40
## 2.5 Features and Prices of STC8F1K08S2 family

### Prices of different selections

<table>
<thead>
<tr>
<th>Microcontroller Model</th>
<th>Operating Voltage(V)</th>
<th>Flash Program Memory</th>
<th>Large Capacity Expansion SRAM(Direct)</th>
<th>Powerful dual DPTR increase or Decrease</th>
<th>EEPROM</th>
<th>I/O maximum number</th>
<th>Timer/Counter(External POW-down Wake-up)</th>
<th>15 High-speed PWM(Timers)</th>
<th>15 High-speed PWM(Enhanced PWM, Cycle Control)</th>
<th>Watchdog Reset timer</th>
<th>Internal Clock(24MHz Adjustable)</th>
<th>Program encrypted transmission</th>
<th>Support RS485 download</th>
<th>Support USB download</th>
<th>Support online simulation</th>
<th>Footprint</th>
</tr>
</thead>
<tbody>
<tr>
<td>STC8F1K08S2 2.0-5.5 5</td>
<td>2.0-5.5</td>
<td>8K</td>
<td>2K</td>
<td>3K</td>
<td>2</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>TSSOP20</td>
</tr>
<tr>
<td>STC8F1K08S 2.0-5.5 5</td>
<td>2.0-5.5</td>
<td>8K</td>
<td>2K</td>
<td>3K</td>
<td>2</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>SOP16</td>
</tr>
</tbody>
</table>

### Core
- Enhanced 8051 Core with single clock per machine cycle (1T)
- Fully compatible instruction set with traditional 8051
- 19 interrupt sources and 4 interrupt priority levels
- Online debugging is supported

### Operating voltage
- 2.0 to 5.5V
- Built-in LDO

### Operating temperature
- −40°C~85°C

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

### SRAM
- 128 bytes internal direct access RAM
- 128 bytes internal indirect access RAM
- 2048 bytes internal extended RAM
- RAM expandable externally up to 64 Kbytes

### Clock
- Internal 24MHz high precise R/C clock IRC
  - Error: ±0.3%

---

Nantong guoxin Microelectronics Co., Ltd.  Tel: 0513-5501 2928/2929/2966  Fax: 0513-5501 2926/2956/2947
Temperature drift: ±1.0% at the temperature range of -40°C to 85°C and ±0.6% at the temperature range of -20°C to 65°C

- Internal 32KHz low speed IRC with large error
- External 4MHz–33MHz oscillator or external clock

The three clock source above can be selected freely by used code.

**Reset**
- Hardware reset
  - Power-on reset
  - Reset by reset pin with high reset pulse
  - Watch dog timer reset
  - Low voltage detection reset. 4 low voltage detection levels are provided, 2.2V, 2.4V, V2.7, V3.0
- Software reset
  - Writing the reset trigger register using software

**Interrupts**
- 19 interrupt sources: INT0, INT1, INT2, INT3, INT4, timer0, timer1, timer2, timer3, timer4, uart1, uart2, uart3, uart4, LVD, PCA/CCP, SPI, I²C, comparator
- 4 interrupt priority levels

**Digital peripherals**
- 5 16-bit timers: timer0, timer1, timer2, timer3, timer4. Where the mode 3 of timer0 has the Non Maskable Interrupt (NMI in short) function. Mode 0 of timer0 and timer1 is 16-bit Auto-reload mode.
- 4 high speed UARTs: uart1, uart2, uart3, uart4, whose baud rate clock source may be fast as FOSC/4
- 4 groups of PCA: CCP0, CCP1, CCP2, CCP3, which can be used as capture, high speed output and 6-bits, 7-bits, 8-bits or 10-bits PWM
- SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
- I²C: Master mode or slave mode are supported.

**Analog peripherals**
- Comparator

**GPIO**
- Up to 42 GPIOs: P0.0–P0.7, P1.0–P1.7, P2.0–P2.7, P3.0–P3.7, P4.0–P4.7, P5.4–P5.5
- 4 modes for all GPIOs: quasi-bidirectional mode, push-pull output mode, open drain mode, high-impedance input mode

**Package**
- LQFP44, PDIP40

### 2.6 Features and Prices of STC8H1K08S2A10 family

- Prices of different selections
**Core**
- Enhanced 8051 Core with single clock per machine cycle (1T)
- Fully compatible instruction set with traditional 8051
- 19 interrupt sources and 4 interrupt priority levels
- Online debugging is supported

**Operating voltage**
- 2.0 to 5.5V
- Built-in LDO

**Operating temperature**
- -40°C~85°C

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

**SRAM**
- 128 bytes internal direct access RAM
- 128 bytes internal indirect access RAM
- 2048 bytes internal extended RAM
- RAM expandable externally up to 64 Kbytes

**Clock**
- Internal 24MHz high precise R/C clock IRC
  - Error: ±0.3%
  - Temperature drift: ±1.0% at the temperature range of -40°C to 85°C and ±0.6% at the temperature range of -20°C to 65°C
- Internal 32KHz low speed IRC with large error
- External 4MHz~33MHz oscillator or external clock
  The three clock source above can be selected freely by used code.

**Reset**
Hardware reset
  ✷ Power-on reset
  ✷ Reset by reset pin with high reset pulse
  ✷ Watch dog timer reset
  ✷ Low voltage detection reset. 4 low voltage detection levels are provided, 2.2V, 2.4V, V2.7, V3.0
Software reset
  ✷ Writing the reset trigger register using software

Interrupts
  ✷ 19 interrupt sources: INT0, INT1, INT2, INT3, INT4, timer0, timer1, timer2, timer3, timer4, uart1, uart2, uart3, uart4, LVD, PCA/CCP, SPI, I2C, comparator
  ✷ 4 interrupt priority levels

Digital peripherals
  ✷ 5 16-bit timers: timer0, timer1, timer2, timer3, timer4. Where the mode 3 of timer0 has the Non Maskable Interrupt (NMI in short) function. Mode 0 of timer0 and timer1 is 16-bit Auto-reload mode.
  ✷ 4 high speed UARTs: uart1, uart2,uart3, uart4, whose baud rate clock source may be fast as FOSC/4
  ✷ 4 groups of PCA: CCP0, CCP1, CCP2, CCP3, which can be used as capture, high speed output and 6-bits, 7-bits, 8-bits or 10-bits PWM
  ✷ SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
  ✷ I2C: Master mode or slave mode are supported.

Analog peripherals
  ✷ Comparator

GPIO
  ✷ Up to 42 GPIOs: P0.0~P0.7, P1.0~P1.7, P2.0~P2.7, P3.0~P3.7, P4.0~P4.7, P5.4~P5.5
  ✷ 4 modes for all GPIOs: quasi-bidirectional mode, push-pull output mode, open drain mode, high-impedance input mode

Package
  ✷ LQFP44, PDIP40
## 2.7 Advance notice of STC8H1K64S2A10 family

### Prices of different selections

<table>
<thead>
<tr>
<th>Microcontroller Model</th>
<th>Operating Voltage(V)</th>
<th>Flash Program Memory</th>
<th>Large Capacity Expansion SRAM (K bytes)</th>
<th>EEPROM (K bytes)</th>
<th>I/O maximum number</th>
<th>Serial ports Power-down wake-up</th>
<th>SPI</th>
<th>SoP</th>
<th>Timer/Counter(External Power-down Wake-up)</th>
<th>16 bits Enhanced PWM Timers</th>
<th>15 bits Enhanced PWM Timers</th>
<th>Power-down wake-up timer</th>
<th>Internal Low-vol Detection interrupt</th>
<th>Internal Reset (optional reset threshold)</th>
<th>Program encrypted transmission</th>
<th>Set password for next update procedure</th>
<th>Support RS485 download</th>
<th>Support USB download</th>
<th>Online simulation</th>
<th>Footprint</th>
<th>2018 new product list</th>
</tr>
</thead>
<tbody>
<tr>
<td>STC8H1K64S2A10</td>
<td>1.7-5.5</td>
<td>64K</td>
<td>2</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>LQFP48</td>
<td></td>
</tr>
<tr>
<td>STC8H1K32S2A10</td>
<td>1.7-5.5</td>
<td>32K</td>
<td>2</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>LQFP48</td>
<td></td>
</tr>
<tr>
<td>STC8H1K16S2A10</td>
<td>1.7-5.5</td>
<td>16K</td>
<td>2</td>
<td>Yes</td>
<td>Yes</td>
<td>4</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>LQFP48</td>
<td></td>
</tr>
</tbody>
</table>

## 2.8 Advance notice of STC8H1K08S2 family

### Prices of different selections

<table>
<thead>
<tr>
<th>Microcontroller Model</th>
<th>Operating Voltage(V)</th>
<th>Flash Program Memory</th>
<th>Large Capacity Expansion SRAM (K bytes)</th>
<th>EEPROM (K bytes)</th>
<th>I/O maximum number</th>
<th>Serial ports Power-down wake-up</th>
<th>SPI</th>
<th>SoP</th>
<th>Timer/Counter(External Power-down Wake-up)</th>
<th>16 bits Enhanced PWM Timers</th>
<th>15 bits Enhanced PWM Timers</th>
<th>Power-down wake-up timer</th>
<th>Internal Low-vol Detection interrupt</th>
<th>Internal Reset (optional reset threshold)</th>
<th>Program encrypted transmission</th>
<th>Set password for next update procedure</th>
<th>Support RS485 download</th>
<th>Support USB download</th>
<th>Online simulation</th>
<th>Footprint</th>
<th>2018 new product list</th>
</tr>
</thead>
<tbody>
<tr>
<td>STC8H1K08S2</td>
<td>1.7-5.5</td>
<td>8K</td>
<td>2</td>
<td>Yes</td>
<td>Yes</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>SOIC16</td>
<td></td>
</tr>
</tbody>
</table>

July Sample delivery.
## 2.9 Advance notice of STC8H04A10 family

### Prices of different selections

<table>
<thead>
<tr>
<th>Microcontroller Model</th>
<th>Operating Voltage(V)</th>
<th>Flash Program Memory (100K times bytes)</th>
<th>Large Capacity Expansion SRAM bytes</th>
<th>Timer/Counter (External Power-down Wake-up)</th>
<th>Power-down wake-up timer</th>
<th>PCU/CPU (PWM mode or 2 external interrupt to compare edges)</th>
<th>16-bit Enhanced PWM Timers</th>
<th>15 High-speed ADC (PWM as 8D/A use)</th>
<th>Internal Reset (optional reset threshold Vol)</th>
<th>Internal Clock (24MHz Adaptable)</th>
<th>External clock output and reset</th>
<th>Power-down Wake-up Timer</th>
<th>15 High-speed ADC (D/A use)</th>
<th>Watchdog Reset timer</th>
<th>Support for RS485 download</th>
<th>Support for USB download</th>
<th>Online simulation</th>
<th>Support for before download</th>
<th>Footprint</th>
<th>2018 new product list</th>
</tr>
</thead>
<tbody>
<tr>
<td>STC8H04A10</td>
<td>1.7-5.5</td>
<td>4k</td>
<td>256</td>
<td>1</td>
<td>6</td>
<td>1</td>
<td>Yes</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>SOP8</td>
<td>July</td>
</tr>
</tbody>
</table>

## 2.10 Advance notice of STC8H04 family

### Prices of different selections

<table>
<thead>
<tr>
<th>Microcontroller Model</th>
<th>Operating Voltage(V)</th>
<th>Flash Program Memory (100K times bytes)</th>
<th>Large Capacity Expansion SRAM bytes</th>
<th>Timer/Counter (External Power-down Wake-up)</th>
<th>Power-down wake-up timer</th>
<th>PCU/CPU (PWM mode or 2 external interrupt to compare edges)</th>
<th>16-bit Enhanced PWM Timers</th>
<th>15 High-speed ADC (PWM as 8D/A use)</th>
<th>Internal Reset (optional reset threshold Vol)</th>
<th>Internal Clock (24MHz Adaptable)</th>
<th>External clock output and reset</th>
<th>Power-down Wake-up Timer</th>
<th>15 High-speed ADC (D/A use)</th>
<th>Watchdog Reset timer</th>
<th>Support for RS485 download</th>
<th>Support for USB download</th>
<th>Online simulation</th>
<th>Support for before download</th>
<th>Footprint</th>
<th>2018 new product list</th>
</tr>
</thead>
<tbody>
<tr>
<td>STC8H04</td>
<td>1.7-5.5</td>
<td>4k</td>
<td>256</td>
<td>1</td>
<td>6</td>
<td>1</td>
<td>Yes</td>
<td>-</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>SOP8</td>
<td>January</td>
</tr>
</tbody>
</table>

Nantong guoxin Microelectronics Co., Ltd.  Tel: 0513-5501 2928/2929/2966  Fax: 0513-5501 2926/2956/2947
3 Pinouts and pin descriptions

3.1 Pinouts

3.1.1 STC8A8K64S4A12 family pinouts

---

the last letter of the bottom line of the chip silk screen is the chip version number
STC8A4K64S2A12
LQFP48

the last letter of the bottom line of the chip silk screen is the chip version number.
3.1.2 STC8A4K64S2A12 family pinouts

the last letter of the bottom line of the chip silk screen is the chip version number
STC8A4K64S2A12
LQFP48

the last letter of the bottom line of the chip
silk screen is the chip version number
the last letter of the bottom line of the chip silk screen is the chip version number
3.1.3 STC8F2K64S4 family pinouts

First Pin

the last letter of the bottom line of the chip silk screen is the chip version number

- 22 -
First pin

the last letter of the bottom line of the chip silk screen is the chip version number
3.1.4 STC8F2K64S2 family pinouts
### 3.1.5 STC8F1K08S2 family pinouts

<table>
<thead>
<tr>
<th>TSSOP20/SOP20</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1.1/TxD2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1.0/RxD2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.7/TINT/TxD_2/CMP-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.6/TINT/RxD_2/CMP-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.5/T1/T0CLKO/SS_4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.4/T1/T0CLKO/MOSI_4/CMPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.3/T1/INT/MISO_4/I2CSDA_4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.2/T1/INT/SCLK_4/I2CSCL_4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.1/TxD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vcc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P5.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gnd</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1.1/TxD2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1.0/RxD2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.7/TINT/TxD_2/CMP-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.6/TINT/RxD_2/CMP-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.5/T1/T0CLKO/SS_4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.4/T1/T0CLKO/MOSI_4/CMPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.3/T1/INT/MISO_4/I2CSDA_4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.2/T1/INT/SCLK_4/I2CSCL_4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.1/TxD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vcc/ADC_AVcc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I2CSCL_2/P5.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gnd/ADC_AGnd</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The P1.0 and P1.1 pins are not the same as the STC8F2K series.

### 3.1.6 STC8H1K08S2A10 family pinouts

<table>
<thead>
<tr>
<th>SOP8</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>P3.7/TINT/TxD_2/CMP-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.6/TINT/RxD_2/CMP-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.5/T1/T0CLKO/SS_4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.4/T1/T0CLKO/MOSI_4/CMPO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.3/T1/INT/MISO_4/I2CSDA_4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.2/T1/INT/SCLK_4/I2CSCL_4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.1/TxD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vcc/ADC_AVcc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I2CSCL_2/P5.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gnd/ADC_AGnd</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TSSOP9</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1.1/ADC1/TxD2/CCP0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1.0/ADC0/RxD2/CCP1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.7/TINT/TxD_2/2CCP2/2CCP3/CMP-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.6/TINT/RxD_2/2CCP2/2CCP3/CMP-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.5/T1/T0CLKO/CCP0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.4/T1/T0CLKO/ECI_2/CMP0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.3/T1/INT/SLCLK_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.2/T1/INT0/MISO_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.1/TxD/T2/MOSI_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.0/ADC8/RxD/T2CLKO/INT4/SS_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SOP8</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>P3.7/TINT/TxD_2/CCP2/CCP3/CMP-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.6/ADC14/INT2/RxD_2/2CCP1/2CCP2/2CCP3/CMP-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.5/ADC13/T1/T0CLKO/CCP0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.4/ADC12/T0/SLCLK/ECI/2/CMP0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.3/ADC11/INT1/SLCLK_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.2/ADC10/INT0/MISO_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.1/ADC9/TxD/T2/MOSI_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3.0/ADC8/RxD/T2CLKO/INT4/SS_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SOP16</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>TxD2/CCP0/ADC1/P1.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RxD2/CCP1/ADC0/P1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCLKO_2/RxD_3/ADC6/P1.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I2CSCL_2/P5.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gnd/ADC_AGnd</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SOP16</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>TxD2/CCP0/ADC1/P1.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RxD2/CCP1/ADC0/P1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCLKO_2/RxD_3/ADC6/P1.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I2CSCL_2/P5.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gnd/ADC_AGnd</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
3.1.7 GX8S003 family pinouts

Special pin package for customer needs

3.1.8 STC8H1K08S2 family pinouts
## 3.2 Pin descriptions

### 3.2.1 STC8A8K64S4A12 family pin descriptions

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Class</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>P5.2</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>2</td>
<td>RxD4_2</td>
<td>I</td>
<td>Serial Port 4 Receive Pin</td>
</tr>
<tr>
<td>2</td>
<td>P5.3</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>2</td>
<td>TxD4_2</td>
<td>O</td>
<td>Serial Port 4 Transport Pin</td>
</tr>
<tr>
<td>3</td>
<td>P1.1</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>3</td>
<td>ADC1</td>
<td>I</td>
<td>ADC analog input channel 1</td>
</tr>
<tr>
<td>3</td>
<td>PWM1_2</td>
<td>O</td>
<td>Enhanced PWM channel 1 output pin</td>
</tr>
<tr>
<td>3</td>
<td>TxD2</td>
<td>O</td>
<td>Serial Port 2 Transport Pin</td>
</tr>
<tr>
<td>4</td>
<td>P1.2</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>4</td>
<td>ADC2</td>
<td>I</td>
<td>ADC analog input channel 2</td>
</tr>
<tr>
<td>4</td>
<td>PWM2_2</td>
<td>O</td>
<td>Enhanced PWM channel 2 output pin</td>
</tr>
<tr>
<td>4</td>
<td>SS</td>
<td>I/O</td>
<td>SPI Slave selection</td>
</tr>
<tr>
<td>4</td>
<td>T2</td>
<td>I</td>
<td>Timer 2 external clock input</td>
</tr>
<tr>
<td>4</td>
<td>ECI</td>
<td>I</td>
<td>PCA external pulse input</td>
</tr>
<tr>
<td>5</td>
<td>P1.3</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>5</td>
<td>ADC3</td>
<td>I</td>
<td>ADC analog input channel 3</td>
</tr>
<tr>
<td>5</td>
<td>PWM3_2</td>
<td>O</td>
<td>Enhanced PWM channel 3 output pin</td>
</tr>
<tr>
<td>5</td>
<td>MOSI</td>
<td>I/O</td>
<td>SPI master output slave input</td>
</tr>
<tr>
<td>5</td>
<td>T2CLKO</td>
<td>O</td>
<td>Timer 2 clock frequency output</td>
</tr>
<tr>
<td>6</td>
<td>P6.0</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>6</td>
<td>PWM0_3</td>
<td>O</td>
<td>Enhanced PWM channel 0 output pin</td>
</tr>
<tr>
<td>7</td>
<td>P6.1</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>7</td>
<td>PWM1_3</td>
<td>O</td>
<td>Enhanced PWM channel 1 output pin</td>
</tr>
<tr>
<td>8</td>
<td>P6.2</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>8</td>
<td>PWM2_3</td>
<td>O</td>
<td>Enhanced PWM channel 2 output pin</td>
</tr>
<tr>
<td>9</td>
<td>P6.3</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>9</td>
<td>PWM3_3</td>
<td>O</td>
<td>Enhanced PWM channel 3 output pin</td>
</tr>
<tr>
<td>Number</td>
<td>LQFP64</td>
<td>LQFP48</td>
<td>LQFP44</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>7</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>8</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>13</td>
<td>9</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>9</td>
<td>13</td>
</tr>
<tr>
<td>15</td>
<td>11</td>
<td>10</td>
<td>14</td>
</tr>
<tr>
<td>16</td>
<td>12</td>
<td>11</td>
<td>15</td>
</tr>
<tr>
<td>17</td>
<td>13</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>18</td>
<td>14</td>
<td>13</td>
<td>17</td>
</tr>
<tr>
<td>19</td>
<td>15</td>
<td>14</td>
<td>18</td>
</tr>
<tr>
<td>20</td>
<td>16</td>
<td>15</td>
<td>19</td>
</tr>
<tr>
<td>21</td>
<td>17</td>
<td>16</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number</th>
<th>LQFP64</th>
<th>LQFP48</th>
<th>LQFP44</th>
<th>PDIP40</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>6</td>
<td>5</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>7</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>8</td>
<td>7</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>9</td>
<td>8</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>9</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>11</td>
<td>10</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>12</td>
<td>11</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>13</td>
<td>12</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>14</td>
<td>13</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>15</td>
<td>14</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>16</td>
<td>15</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>17</td>
<td>16</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1.4</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>ADC4</td>
<td>I</td>
<td>ADC analog input channel 4</td>
</tr>
<tr>
<td>PWM4_2</td>
<td>O</td>
<td>Enhanced PWM channel 4 output pin</td>
</tr>
<tr>
<td>MISO</td>
<td>I/O</td>
<td>SPI master input slave output</td>
</tr>
<tr>
<td>SDA</td>
<td>I/O</td>
<td>I2C interface data line</td>
</tr>
<tr>
<td>CCP3</td>
<td>I/O</td>
<td>PCA external pulse input</td>
</tr>
<tr>
<td>P4.4</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>RD</td>
<td>O</td>
<td>External bus read signal line</td>
</tr>
<tr>
<td>TxD_4</td>
<td>O</td>
<td>Serial Port 1 Transport Pin</td>
</tr>
<tr>
<td>P1.5</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>ADC5</td>
<td>I</td>
<td>ADC analog input channel 5</td>
</tr>
<tr>
<td>PWM5_2</td>
<td>O</td>
<td>Enhanced PWM channel 5 output pin</td>
</tr>
<tr>
<td>SCLK</td>
<td>I/O</td>
<td>SPI Clock line</td>
</tr>
<tr>
<td>SCL</td>
<td>I/O</td>
<td>I2C Clock line</td>
</tr>
<tr>
<td>CCP2</td>
<td>I/O</td>
<td>PCA capture input and pulse output</td>
</tr>
<tr>
<td>P1.6</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>ADC6</td>
<td>I</td>
<td>ADC analog input channel 6</td>
</tr>
<tr>
<td>RxD_3</td>
<td>I</td>
<td>Serial Port 1 Receive Pin</td>
</tr>
<tr>
<td>PWM6_2</td>
<td>O</td>
<td>Enhanced PWM channel 6 output pin</td>
</tr>
<tr>
<td>MCLKO_2</td>
<td>O</td>
<td>Main clock frequency output</td>
</tr>
<tr>
<td>CCP1</td>
<td>I/O</td>
<td>PCA capture input and pulse output</td>
</tr>
<tr>
<td>XTALO</td>
<td>O</td>
<td>Output pin of external crystal</td>
</tr>
<tr>
<td>P1.7</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>ADC7</td>
<td>I</td>
<td>ADC analog input channel 7</td>
</tr>
<tr>
<td>TxD_3</td>
<td>O</td>
<td>Serial Port 1 Transport Pin</td>
</tr>
<tr>
<td>PWM7_2</td>
<td>O</td>
<td>Enhanced PWM channel 7 output pin</td>
</tr>
<tr>
<td>CCP0</td>
<td>I/O</td>
<td>PCA capture input and pulse output</td>
</tr>
<tr>
<td>XTALI</td>
<td>I</td>
<td>External crystal/external clock input pin</td>
</tr>
<tr>
<td>ADC_AGnd</td>
<td>GND</td>
<td>ADC GND</td>
</tr>
<tr>
<td>ARef</td>
<td>I</td>
<td>ADC reference voltage pin</td>
</tr>
<tr>
<td>ADC_AVcc</td>
<td>VCC</td>
<td>ADC VCC</td>
</tr>
<tr>
<td>P5.4</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>RST</td>
<td>I</td>
<td>Reset Pin</td>
</tr>
<tr>
<td>MCLKO</td>
<td>O</td>
<td>Main clock frequency output</td>
</tr>
<tr>
<td>Vcc</td>
<td>VCC</td>
<td>Source Pin</td>
</tr>
<tr>
<td>P5.5</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
</tbody>
</table>

Nantong guoxin Microelectronics Co., Ltd.  Tel: 0513-5501 2928/2929/2966  Fax: 0513-5501 2926/2956/2947
<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Class</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>P4.0</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>WR_3</td>
<td>O</td>
<td>External bus write signal line</td>
</tr>
<tr>
<td></td>
<td>RxD2_2</td>
<td>I</td>
<td>Serial Port 2 Receive Pin</td>
</tr>
<tr>
<td>23</td>
<td>P6.4</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>PWM4_3</td>
<td>O</td>
<td>Enhanced PWM channel 4 output pin</td>
</tr>
<tr>
<td>24</td>
<td>P6.5</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>PWM5_3</td>
<td>O</td>
<td>Enhanced PWM channel 5 output pin</td>
</tr>
<tr>
<td>25</td>
<td>P6.6</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>PWM6_3</td>
<td>O</td>
<td>Enhanced PWM channel 6 output pin</td>
</tr>
<tr>
<td>26</td>
<td>P6.7</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>PWM7_3</td>
<td>O</td>
<td>Enhanced PWM channel 7 output pin</td>
</tr>
<tr>
<td>27</td>
<td>P3.0</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>RxD</td>
<td>I</td>
<td>Serial Port 1 Receive Pin</td>
</tr>
<tr>
<td></td>
<td>CCP3_4</td>
<td>I/O</td>
<td>PCA CAPTURE INPUT AND PULSE OUTPUT</td>
</tr>
<tr>
<td></td>
<td>INT4</td>
<td>I</td>
<td>External interrupt 4</td>
</tr>
<tr>
<td>28</td>
<td>P3.1</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>TxD</td>
<td>O</td>
<td>Serial Port 1 Transport Pin</td>
</tr>
<tr>
<td></td>
<td>CCP2_4</td>
<td>I/O</td>
<td>PCA capture input and pulse output</td>
</tr>
<tr>
<td>29</td>
<td>P3.2</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>INT0</td>
<td>I</td>
<td>External interrupt 0</td>
</tr>
<tr>
<td></td>
<td>CCP1_4</td>
<td>I/O</td>
<td>PCA capture input and pulse output</td>
</tr>
<tr>
<td></td>
<td>SCLK_4</td>
<td>I/O</td>
<td>SPI CLOCK LINE</td>
</tr>
<tr>
<td></td>
<td>SCL_4</td>
<td>I/O</td>
<td>I2C CLOCK LINE</td>
</tr>
<tr>
<td>30</td>
<td>P3.3</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>INT1</td>
<td>I</td>
<td>External interrupt 1</td>
</tr>
<tr>
<td></td>
<td>CCP0_4</td>
<td>I/O</td>
<td>PCA capture input and pulse output</td>
</tr>
<tr>
<td></td>
<td>MISO_4</td>
<td>I/O</td>
<td>SPI master input slave output</td>
</tr>
<tr>
<td></td>
<td>SDA_4</td>
<td>I/O</td>
<td>I2C INTERFACE DATA LINE</td>
</tr>
<tr>
<td>31</td>
<td>P3.4</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>T0</td>
<td>I</td>
<td>Timer 0 external clock input</td>
</tr>
<tr>
<td></td>
<td>T1CLKO</td>
<td>O</td>
<td>Timer 1 clock frequency output</td>
</tr>
<tr>
<td></td>
<td>MOSI_4</td>
<td>I/O</td>
<td>SPI master output slave input</td>
</tr>
<tr>
<td></td>
<td>CMPO</td>
<td>O</td>
<td>Comparator output</td>
</tr>
<tr>
<td>Number</td>
<td>Name</td>
<td>Class</td>
<td>Instruction</td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
<td>-------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>32</td>
<td>P3.5</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>T1</td>
<td>I</td>
<td>Timer 1 external clock input</td>
</tr>
<tr>
<td></td>
<td>T0CLKO</td>
<td>O</td>
<td>Timer 0 clock divider output</td>
</tr>
<tr>
<td></td>
<td>ECI_4</td>
<td>I</td>
<td>PCA external pulse input</td>
</tr>
<tr>
<td></td>
<td>SS_4</td>
<td>I</td>
<td>SPI slave select pin (host output)</td>
</tr>
<tr>
<td></td>
<td>PWMFLT</td>
<td>I</td>
<td>Enhanced PWM external anomaly detection pin</td>
</tr>
<tr>
<td>33</td>
<td>P5.0</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>RxD3_2</td>
<td>I</td>
<td>Serial Port 3 Receive Pin</td>
</tr>
<tr>
<td>34</td>
<td>P5.1</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>TxD3_2</td>
<td>O</td>
<td>Serial Port 3 Transport Pin</td>
</tr>
<tr>
<td>35</td>
<td>P3.6</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>INT2</td>
<td>I</td>
<td>External interrupt 2</td>
</tr>
<tr>
<td></td>
<td>WR_2</td>
<td>O</td>
<td>External bus write signal line</td>
</tr>
<tr>
<td></td>
<td>RxD_2</td>
<td>I</td>
<td>Serial Port 1 Receive Pin</td>
</tr>
<tr>
<td></td>
<td>CMP-</td>
<td>I</td>
<td>Comparator negative input</td>
</tr>
<tr>
<td>36</td>
<td>P3.7</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>INT3</td>
<td>I</td>
<td>External interrupt 3</td>
</tr>
<tr>
<td></td>
<td>RD_2</td>
<td>O</td>
<td>External bus read signal line</td>
</tr>
<tr>
<td></td>
<td>TxD_2</td>
<td>O</td>
<td>Serial Port 1 Transport Pin</td>
</tr>
<tr>
<td></td>
<td>CMP+</td>
<td>I</td>
<td>Comparator positive input</td>
</tr>
<tr>
<td>37</td>
<td>P4.1</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>ALE</td>
<td>O</td>
<td>Address latch signal</td>
</tr>
<tr>
<td></td>
<td>CMP0_2</td>
<td>O</td>
<td>Comparator output</td>
</tr>
<tr>
<td>38</td>
<td>P7.0</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>CCP0_3</td>
<td>I/O</td>
<td>PCA capture input and pulse output</td>
</tr>
<tr>
<td>39</td>
<td>P7.1</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>CCP1_3</td>
<td>I/O</td>
<td>PCA capture input and pulse output</td>
</tr>
<tr>
<td>40</td>
<td>P7.2</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>CCP2_3</td>
<td>I/O</td>
<td>PCA capture input and pulse output</td>
</tr>
<tr>
<td>41</td>
<td>P7.3</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>CCP3_3</td>
<td>I/O</td>
<td>PCA capture input and pulse output</td>
</tr>
<tr>
<td>42</td>
<td>P2.0</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>A8</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>PWM0</td>
<td>O</td>
<td>Enhanced PWM channel 0 output pin</td>
</tr>
<tr>
<td></td>
<td>RSTSV</td>
<td>-</td>
<td>the port can be configured during ISP download</td>
</tr>
<tr>
<td>43</td>
<td>P4.2</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>RD_3</td>
<td>O</td>
<td>External bus read signal line</td>
</tr>
<tr>
<td></td>
<td>TxD2_2</td>
<td>O</td>
<td>Serial Port 2 Transport Pin</td>
</tr>
<tr>
<td>Number</td>
<td>Name</td>
<td>Class</td>
<td>Instruction</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>-------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>LQFP64S</td>
<td>LQFP48</td>
<td>LQFP44</td>
<td>PDIP40</td>
</tr>
<tr>
<td>44</td>
<td>P2.1</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>A9</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>PWM1</td>
<td>O</td>
<td>Enhanced PWM channel 1 output pin</td>
</tr>
<tr>
<td>45</td>
<td>P2.2</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>A10</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>PWM2</td>
<td>O</td>
<td>Enhanced PWM channel 2 output pin</td>
</tr>
<tr>
<td></td>
<td>SS_2</td>
<td>I</td>
<td>SPI slave select pin (host output)</td>
</tr>
<tr>
<td></td>
<td>ECI_2</td>
<td>I</td>
<td>PCA external pulse input</td>
</tr>
<tr>
<td>46</td>
<td>P2.3</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>A11</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>PWM3</td>
<td>O</td>
<td>Enhanced PWM channel 3 output pin</td>
</tr>
<tr>
<td></td>
<td>MOSI_2</td>
<td>I/O</td>
<td>SPI master output slave input</td>
</tr>
<tr>
<td></td>
<td>CCP0</td>
<td>I/O</td>
<td>PCA capture input and pulse output</td>
</tr>
<tr>
<td>47</td>
<td>P2.4</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>A12</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>PWM4</td>
<td>O</td>
<td>Enhanced PWM channel 4 output pin</td>
</tr>
<tr>
<td></td>
<td>MISO_2</td>
<td>I/O</td>
<td>SPI master input slave output</td>
</tr>
<tr>
<td></td>
<td>SDA_2</td>
<td>I/O</td>
<td>I2C INTERFACE DATA LINE</td>
</tr>
<tr>
<td></td>
<td>CCP1</td>
<td>I/O</td>
<td>PCA capture input and pulse output</td>
</tr>
<tr>
<td>48</td>
<td>P2.5</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>A13</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>PWM5</td>
<td>O</td>
<td>Enhanced PWM channel 5 output pin</td>
</tr>
<tr>
<td></td>
<td>SCLK_2</td>
<td>I/O</td>
<td>SPI CLOCK LINE</td>
</tr>
<tr>
<td></td>
<td>SCL_2</td>
<td>I/O</td>
<td>I2C CLOCK LINE</td>
</tr>
<tr>
<td></td>
<td>CCP2</td>
<td>I/O</td>
<td>PCA capture input and pulse output</td>
</tr>
<tr>
<td>49</td>
<td>P2.6</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>A14</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>PWM6</td>
<td>O</td>
<td>Enhanced PWM channel 6 output pin</td>
</tr>
<tr>
<td></td>
<td>CCP3</td>
<td>I/O</td>
<td>PCA capture input and pulse output</td>
</tr>
<tr>
<td>50</td>
<td>P2.7</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>A15</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>PWM7</td>
<td>O</td>
<td>Enhanced PWM channel 7 output pin</td>
</tr>
<tr>
<td>51</td>
<td>P0.0</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>AD0</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>ADC8</td>
<td>I</td>
<td>ADC analog input channel 8</td>
</tr>
<tr>
<td></td>
<td>RxD3</td>
<td>I</td>
<td>Serial Port 3 Receive Pin</td>
</tr>
<tr>
<td>Number</td>
<td>Name</td>
<td>Class</td>
<td>Instruction</td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
<td>-------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td></td>
<td>LQFP64S</td>
<td>LQFP48</td>
<td>LQFP44</td>
</tr>
<tr>
<td>52</td>
<td>P0.1</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td>53</td>
<td>AD1</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>ADC9</td>
<td>I</td>
<td>ADC analog input channel 9</td>
</tr>
<tr>
<td></td>
<td>TxD3</td>
<td>O</td>
<td>Serial Port 3 Transport Pin</td>
</tr>
<tr>
<td>52</td>
<td>P0.2</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td>53</td>
<td>AD2</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>ADC10</td>
<td>I</td>
<td>ADC analog input channel 10</td>
</tr>
<tr>
<td></td>
<td>RxD4</td>
<td>I</td>
<td>Serial Port 4 Receive Pin</td>
</tr>
<tr>
<td>54</td>
<td>P7.4</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>SS_3</td>
<td>I</td>
<td>SPI slave select pin (host output)</td>
</tr>
<tr>
<td></td>
<td>ECI_3</td>
<td>I</td>
<td>PCA external pulse input</td>
</tr>
<tr>
<td>55</td>
<td>P7.5</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>MOSI_3</td>
<td>I/O</td>
<td>SPI master output slave input</td>
</tr>
<tr>
<td>56</td>
<td>P7.6</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>MISO_3</td>
<td>I/O</td>
<td>SPI master input slave output</td>
</tr>
<tr>
<td></td>
<td>SDA_3</td>
<td>I/O</td>
<td>I2C INTERFACE DATA LINE</td>
</tr>
<tr>
<td>57</td>
<td>P7.7</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>SCLK_3</td>
<td>I/O</td>
<td>SPI CLOCK LINE</td>
</tr>
<tr>
<td></td>
<td>SCL_3</td>
<td>I/O</td>
<td>I2C CLOCK LINE</td>
</tr>
<tr>
<td>58</td>
<td>P4.3</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>WR</td>
<td>O</td>
<td>External bus write signal line</td>
</tr>
<tr>
<td></td>
<td>RxD_4</td>
<td>I</td>
<td>Serial Port 1 Receive Pin</td>
</tr>
<tr>
<td>59</td>
<td>P0.3</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>AD3</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>ADC11</td>
<td>I</td>
<td>ADC analog input channel 11</td>
</tr>
<tr>
<td></td>
<td>TxD4</td>
<td>O</td>
<td>Serial Port 4 Transport Pin</td>
</tr>
<tr>
<td>60</td>
<td>P0.4</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>AD4</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>ADC12</td>
<td>I</td>
<td>ADC analog input channel 12</td>
</tr>
<tr>
<td></td>
<td>T3</td>
<td>I</td>
<td>Timer 3 external clock input</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Class</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LQFP64S</td>
<td>LQFP48</td>
<td>LQFP44</td>
</tr>
<tr>
<td>61</td>
<td>P0.5</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>AD5</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>ADC13</td>
<td>I</td>
<td>ADC analog input channel 13</td>
</tr>
<tr>
<td></td>
<td>T3CLKO</td>
<td>O</td>
<td>Timer 3 clock frequency output</td>
</tr>
<tr>
<td>62</td>
<td>P0.6</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>AD6</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>ADC14</td>
<td>I</td>
<td>ADC analog input channel 14</td>
</tr>
</tbody>
</table>
### 3.2.2 STC8A4K64S2A12 family pin descriptions

<table>
<thead>
<tr>
<th>Number</th>
<th>LQFP64S</th>
<th>LQFP48</th>
<th>LQFP44</th>
<th>PDIP40</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>T4</td>
<td>I</td>
<td>Timer 4 external clock input</td>
</tr>
<tr>
<td>P0.7</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td>AD7</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td>T4CLKO</td>
<td>O</td>
<td>Timer 4 clock frequency output</td>
</tr>
<tr>
<td>P1.0</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td>ADC0</td>
<td>I</td>
<td>ADC analog input channel 0</td>
</tr>
<tr>
<td>PWM0_2</td>
<td>O</td>
<td>Enhanced PWM channel 0 output pin</td>
</tr>
<tr>
<td>RxD2</td>
<td>I</td>
<td>Serial Port 2 Receive Pin</td>
</tr>
<tr>
<td>P5.2</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>P5.3</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>P1.1</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>ADC1</td>
<td>I</td>
<td>ADC analog input channel 1</td>
</tr>
<tr>
<td>PWM1_2</td>
<td>O</td>
<td>Enhanced PWM channel 1 output pin</td>
</tr>
<tr>
<td>TxD2</td>
<td>O</td>
<td>Serial Port 2 Transport Pin</td>
</tr>
<tr>
<td>P1.2</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>ADC2</td>
<td>I</td>
<td>ADC analog input channel 2</td>
</tr>
<tr>
<td>PWM2_2</td>
<td>O</td>
<td>Enhanced PWM channel 2 output pin</td>
</tr>
<tr>
<td>SS</td>
<td>I/O</td>
<td>SPI Slave selection</td>
</tr>
<tr>
<td>T2</td>
<td>I</td>
<td>Timer 2 external clock input</td>
</tr>
<tr>
<td>ECI</td>
<td>I</td>
<td>PCA external pulse input</td>
</tr>
<tr>
<td>P1.3</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>ADC3</td>
<td>I</td>
<td>ADC analog input channel 3</td>
</tr>
<tr>
<td>PWM3_2</td>
<td>O</td>
<td>Enhanced PWM channel 3 output pin</td>
</tr>
<tr>
<td>MOSI</td>
<td>I/O</td>
<td>SPI master output slave input</td>
</tr>
<tr>
<td>T2CLKO</td>
<td>O</td>
<td>Timer 2 clock frequency output</td>
</tr>
<tr>
<td>P6.0</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>PWM0_3</td>
<td>O</td>
<td>Enhanced PWM channel 0 output pin</td>
</tr>
<tr>
<td>P6.1</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>PWM1_3</td>
<td>O</td>
<td>Enhanced PWM channel 1 output pin</td>
</tr>
<tr>
<td>P6.2</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>PWM2_3</td>
<td>O</td>
<td>Enhanced PWM channel 2 output pin</td>
</tr>
<tr>
<td>P6.3</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>PWM3_3</td>
<td>O</td>
<td>Enhanced PWM channel 3 output pin</td>
</tr>
<tr>
<td>Number</td>
<td>LQFP64S</td>
<td>LQFP48</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
<td>--------</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>11</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>12</td>
</tr>
<tr>
<td>17</td>
<td>17</td>
<td>13</td>
</tr>
<tr>
<td>18</td>
<td>18</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>19</td>
<td>15</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td>21</td>
<td>21</td>
<td>17</td>
</tr>
<tr>
<td>Number</td>
<td>LQFP64S</td>
<td>LQFP48</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
<td>--------</td>
</tr>
<tr>
<td>22</td>
<td>18</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>19</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>20</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>21</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>22</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>23</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>LQFP64</td>
<td>LQFP48</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>32</td>
<td>24</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>27</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>28</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>29</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>30</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>31</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>LQFP64S</td>
<td>LQFP48</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
<td>--------</td>
</tr>
<tr>
<td>44</td>
<td>32</td>
<td>29</td>
</tr>
<tr>
<td>45</td>
<td>33</td>
<td>30</td>
</tr>
<tr>
<td>46</td>
<td>34</td>
<td>31</td>
</tr>
<tr>
<td>47</td>
<td>35</td>
<td>32</td>
</tr>
<tr>
<td>48</td>
<td>36</td>
<td>33</td>
</tr>
<tr>
<td>49</td>
<td>37</td>
<td>34</td>
</tr>
<tr>
<td>50</td>
<td>38</td>
<td>35</td>
</tr>
<tr>
<td>51</td>
<td>39</td>
<td>36</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2.1</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td>A9</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td>PWM1</td>
<td>O</td>
<td>Enhanced PWM channel 1 output pin</td>
</tr>
<tr>
<td>P2.2</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td>A10</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td>PWM2</td>
<td>O</td>
<td>Enhanced PWM channel 2 output pin</td>
</tr>
<tr>
<td>SS_2</td>
<td>I</td>
<td>SPI slave select pin (host output)</td>
</tr>
<tr>
<td>ECI_2</td>
<td>I</td>
<td>PCA EXTERNAL PULSE INPUT</td>
</tr>
<tr>
<td>P2.3</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td>A11</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td>PWM3</td>
<td>O</td>
<td>Enhanced PWM channel 3 output pin</td>
</tr>
<tr>
<td>MOSI_2</td>
<td>I/O</td>
<td>SPI master output slave input</td>
</tr>
<tr>
<td>CCP0_2</td>
<td>I/O</td>
<td>PCA capture input and pulse output</td>
</tr>
<tr>
<td>P2.4</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td>A12</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td>PWM4</td>
<td>O</td>
<td>Enhanced PWM channel 4 output pin</td>
</tr>
<tr>
<td>MISO_2</td>
<td>I/O</td>
<td>SPI master input slave output</td>
</tr>
<tr>
<td>SDA_2</td>
<td>I/O</td>
<td>I2C INTERFACE DATA LINE</td>
</tr>
<tr>
<td>CCP1_2</td>
<td>I/O</td>
<td>PCA CAPTURE INPUT AND PULSE OUTPUT</td>
</tr>
<tr>
<td>P2.5</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td>A13</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td>PWM5</td>
<td>O</td>
<td>Enhanced PWM channel 5 output pin</td>
</tr>
<tr>
<td>SCLK_2</td>
<td>I/O</td>
<td>SPI CLOCK LINE</td>
</tr>
<tr>
<td>SCL_2</td>
<td>I/O</td>
<td>I2C CLOCK LINE</td>
</tr>
<tr>
<td>CCP2_2</td>
<td>I/O</td>
<td>PCA capture input and pulse output</td>
</tr>
<tr>
<td>P2.6</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td>A14</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td>PWM6</td>
<td>O</td>
<td>Enhanced PWM channel 6 output pin</td>
</tr>
<tr>
<td>CCP3_2</td>
<td>I/O</td>
<td>PCA capture input and pulse output</td>
</tr>
<tr>
<td>P2.7</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td>A15</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td>PWM7</td>
<td>O</td>
<td>Enhanced PWM channel 7 output pin</td>
</tr>
<tr>
<td>P0.0</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td>AD0</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td>ADC8</td>
<td>I</td>
<td>ADC analog input channel 8</td>
</tr>
<tr>
<td>Number</td>
<td>Name</td>
<td>Class</td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>52</td>
<td>P0.1</td>
<td>I/O</td>
</tr>
<tr>
<td></td>
<td>AD1</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>ADC9</td>
<td>I</td>
</tr>
<tr>
<td>53</td>
<td>P0.2</td>
<td>I/O</td>
</tr>
<tr>
<td></td>
<td>AD2</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>ADC10</td>
<td>I</td>
</tr>
<tr>
<td>54</td>
<td>P7.4</td>
<td>I/O</td>
</tr>
<tr>
<td></td>
<td>SS_3</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>ECI_3</td>
<td>I</td>
</tr>
<tr>
<td>55</td>
<td>P7.5</td>
<td>I/O</td>
</tr>
<tr>
<td></td>
<td>MOSI_3</td>
<td>I/O</td>
</tr>
<tr>
<td>56</td>
<td>P7.6</td>
<td>I/O</td>
</tr>
<tr>
<td></td>
<td>MISO_3</td>
<td>I/O</td>
</tr>
<tr>
<td></td>
<td>SDA_3</td>
<td>I/O</td>
</tr>
<tr>
<td>57</td>
<td>P7.7</td>
<td>I/O</td>
</tr>
<tr>
<td></td>
<td>SCLK_3</td>
<td>I/O</td>
</tr>
<tr>
<td></td>
<td>SCL_3</td>
<td>I/O</td>
</tr>
<tr>
<td>58</td>
<td>P4.3</td>
<td>I/O</td>
</tr>
<tr>
<td></td>
<td>WR</td>
<td>O</td>
</tr>
<tr>
<td></td>
<td>RxD_4</td>
<td>I</td>
</tr>
<tr>
<td>59</td>
<td>P0.3</td>
<td>I/O</td>
</tr>
<tr>
<td></td>
<td>AD3</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>ADC11</td>
<td>I</td>
</tr>
<tr>
<td>60</td>
<td>P0.4</td>
<td>I/O</td>
</tr>
<tr>
<td></td>
<td>AD4</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>ADC12</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td>T3</td>
<td>I</td>
</tr>
</tbody>
</table>
### STC8F2K64S4 family pin descriptions

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Class</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>LQFP64S LQFP48 LQFP44 PDIP40</td>
</tr>
<tr>
<td>61</td>
<td>P0.5</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td>62</td>
<td>P0.6</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td>63</td>
<td>P0.7</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td>64</td>
<td>P1.0</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>AD5</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>AD6</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>AD7</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>ADC13</td>
<td>I</td>
<td>ADC analog input channel 13</td>
</tr>
<tr>
<td></td>
<td>ADC14</td>
<td>I</td>
<td>ADC analog input channel 14</td>
</tr>
<tr>
<td></td>
<td>T3CLKO</td>
<td>O</td>
<td>Timer 3 clock frequency output</td>
</tr>
<tr>
<td></td>
<td>T4</td>
<td>I</td>
<td>Timer 4 external clock input</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LQFP44 PDIP40 LQFP32</td>
</tr>
<tr>
<td>2</td>
<td>P1.1</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td></td>
<td>TxD2</td>
<td>O</td>
<td>Serial Port 2 Transport Pin</td>
</tr>
<tr>
<td>3</td>
<td>P1.2</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td></td>
<td>SS</td>
<td>I</td>
<td>SPI Slave selection</td>
</tr>
<tr>
<td></td>
<td>T2</td>
<td>I</td>
<td>Timer 2 external clock input</td>
</tr>
<tr>
<td>4</td>
<td>P1.3</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td></td>
<td>MOSI</td>
<td>I/O</td>
<td>SPI master output slave input</td>
</tr>
<tr>
<td></td>
<td>T2CLKO</td>
<td>O</td>
<td>Timer 2 clock frequency output</td>
</tr>
<tr>
<td>5</td>
<td>P1.4</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td></td>
<td>MISO</td>
<td>I/O</td>
<td>SPI master input slave output</td>
</tr>
<tr>
<td></td>
<td>SDA</td>
<td>I/O</td>
<td>I2C Data Interface Line</td>
</tr>
<tr>
<td>6</td>
<td>P4.4</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td></td>
<td>RD</td>
<td>O</td>
<td>External bus read signal line</td>
</tr>
<tr>
<td></td>
<td>TxD_4</td>
<td>O</td>
<td>Serial Port 1 Transport Pin</td>
</tr>
<tr>
<td>7</td>
<td>P1.5</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td></td>
<td>SCLK</td>
<td>I/O</td>
<td>SPI Clock line</td>
</tr>
<tr>
<td></td>
<td>SCL</td>
<td>I/O</td>
<td>I2C Clock line</td>
</tr>
<tr>
<td>8</td>
<td>P1.6</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>Pin</td>
<td>Port</td>
<td>Function</td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>RxD_3</td>
<td>I</td>
<td>Serial Port 1 Receive Pin</td>
<td></td>
</tr>
<tr>
<td>XTALO</td>
<td>O</td>
<td>Output pin of external crystal</td>
<td></td>
</tr>
<tr>
<td>MCLKO_2</td>
<td>O</td>
<td>Main clock frequency output</td>
<td></td>
</tr>
<tr>
<td>P1.7</td>
<td>I/O</td>
<td>Standard IO Pins</td>
<td></td>
</tr>
<tr>
<td>TxD_3</td>
<td>O</td>
<td>Serial Port 1 Transport Pin</td>
<td></td>
</tr>
<tr>
<td>XTALI</td>
<td>I</td>
<td>External crystal/external clock input pin</td>
<td></td>
</tr>
<tr>
<td>P4.5</td>
<td>I/O</td>
<td>Standard IO Pins</td>
<td></td>
</tr>
<tr>
<td>P4.6</td>
<td>I/O</td>
<td>Standard IO Pins</td>
<td></td>
</tr>
<tr>
<td>P4.7</td>
<td>I/O</td>
<td>Standard IO Pins</td>
<td></td>
</tr>
<tr>
<td>P5.4</td>
<td>I/O</td>
<td>Standard IO Pins</td>
<td></td>
</tr>
<tr>
<td>RST</td>
<td>I</td>
<td>Reset Pine</td>
<td></td>
</tr>
<tr>
<td>MCLKO</td>
<td>O</td>
<td>Main clock frequency output</td>
<td></td>
</tr>
<tr>
<td>Vcc</td>
<td>VCC</td>
<td>Source Pin</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>Name</td>
<td>Class</td>
<td>Instruction</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>-------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>15</td>
<td>P5.5</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td>16</td>
<td>Gnd</td>
<td>GND</td>
<td>GND</td>
</tr>
<tr>
<td>17</td>
<td>P4.0</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td></td>
<td>WR_3</td>
<td>O</td>
<td>External bus write signal line</td>
</tr>
<tr>
<td></td>
<td>RxD2_2</td>
<td>I</td>
<td>Serial Port 2 Receive Pin</td>
</tr>
<tr>
<td>18</td>
<td>P3.0</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td></td>
<td>RxD</td>
<td>I</td>
<td>Serial Port 1 Receive Pin</td>
</tr>
<tr>
<td></td>
<td>INT4</td>
<td>I</td>
<td>External interrupt 4</td>
</tr>
<tr>
<td>19</td>
<td>P3.1</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td></td>
<td>TxD</td>
<td>O</td>
<td>Serial Port 1 Transport Pin</td>
</tr>
<tr>
<td>20</td>
<td>P3.2</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td></td>
<td>INT0</td>
<td>I</td>
<td>External interrupt 0</td>
</tr>
<tr>
<td></td>
<td>SCL_4</td>
<td>I/O</td>
<td>I2C Clock line</td>
</tr>
<tr>
<td></td>
<td>SCLK_4</td>
<td>I/O</td>
<td>SPI Clock pin</td>
</tr>
<tr>
<td>21</td>
<td>P3.3</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td></td>
<td>INT1</td>
<td>I</td>
<td>External interrupt 1</td>
</tr>
<tr>
<td></td>
<td>SDA_4</td>
<td>I/O</td>
<td>I2C interface data line</td>
</tr>
<tr>
<td></td>
<td>MISO_4</td>
<td>I/O</td>
<td>SPI master input slave output</td>
</tr>
<tr>
<td>22</td>
<td>P3.4</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td></td>
<td>T0</td>
<td>I</td>
<td>Timer 0 external clock input</td>
</tr>
<tr>
<td></td>
<td>T1CLKO</td>
<td>O</td>
<td>Timer 1 clock frequency output</td>
</tr>
<tr>
<td></td>
<td>MOSI_4</td>
<td>I/O</td>
<td>SPI master output slave input</td>
</tr>
<tr>
<td></td>
<td>CMPO</td>
<td>O</td>
<td>Comparator output</td>
</tr>
<tr>
<td>23</td>
<td>P3.5</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td></td>
<td>T1</td>
<td>I</td>
<td>Timer 1 external clock input</td>
</tr>
<tr>
<td></td>
<td>T0CLKO</td>
<td>O</td>
<td>Timer 0 clock divider output</td>
</tr>
<tr>
<td></td>
<td>SS_4</td>
<td>I</td>
<td>SPI slave select pin (host output)</td>
</tr>
<tr>
<td>24</td>
<td>P3.6</td>
<td>I/O</td>
<td>Standard IO Pins</td>
</tr>
<tr>
<td></td>
<td>INT2</td>
<td>I</td>
<td>External interrupt 2</td>
</tr>
<tr>
<td></td>
<td>WR_2</td>
<td>O</td>
<td>External bus write signal line</td>
</tr>
<tr>
<td></td>
<td>RxD_2</td>
<td>I</td>
<td>Receiver 1 of serial port 1</td>
</tr>
<tr>
<td></td>
<td>CMP-</td>
<td>I</td>
<td>Comparator negative input</td>
</tr>
<tr>
<td>Number</td>
<td>LQFP44</td>
<td>PDIP40</td>
<td>LQFP32</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>25</td>
<td>28</td>
<td>20</td>
<td>LQFP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>29</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>30</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>31</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>32</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>33</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>34</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>35</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>36</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>37</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>Name</td>
<td>Class</td>
<td>Instruction</td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
<td>-------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>36</td>
<td>P0.0</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>AD0</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>RxD3</td>
<td>I</td>
<td>Serial Port 3 Receive Pin</td>
</tr>
<tr>
<td>37</td>
<td>P0.1</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>AD1</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>TxD3</td>
<td>O</td>
<td>Serial Port 3 Transport Pin</td>
</tr>
<tr>
<td>38</td>
<td>P0.2</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>AD2</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>RxD4</td>
<td>I</td>
<td>Serial Port 4 Receive Pin</td>
</tr>
<tr>
<td>39</td>
<td>P4.3</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>WR</td>
<td>O</td>
<td>External bus write signal line</td>
</tr>
<tr>
<td></td>
<td>RxD_4</td>
<td>I</td>
<td>Serial Port 1 Receive Pin</td>
</tr>
<tr>
<td>40</td>
<td>P0.3</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>AD3</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>TxD4</td>
<td>O</td>
<td>Serial Port 4 Transport Pin</td>
</tr>
<tr>
<td>41</td>
<td>P0.4</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>AD4</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>T3</td>
<td>I</td>
<td>Timer 3 external clock input</td>
</tr>
<tr>
<td>42</td>
<td>P0.5</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>AD5</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>T3CLKO</td>
<td>O</td>
<td>Timer 3 clock frequency output</td>
</tr>
<tr>
<td>43</td>
<td>P0.6</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>AD6</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>T4</td>
<td>I</td>
<td>Timer 4 external clock input</td>
</tr>
<tr>
<td>44</td>
<td>P0.7</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>AD7</td>
<td>I</td>
<td>Address bus</td>
</tr>
<tr>
<td></td>
<td>T4CLKO</td>
<td>O</td>
<td>Timer 4 clock frequency output</td>
</tr>
<tr>
<td>1</td>
<td>P1.0</td>
<td>I/O</td>
<td>Standard IO port</td>
</tr>
<tr>
<td></td>
<td>RxD2</td>
<td>I</td>
<td>Serial Port 2 Receive Pin</td>
</tr>
</tbody>
</table>
### 3.2.4 STC8F2K64S2 family pin descriptions

<table>
<thead>
<tr>
<th>Number</th>
<th>LQFP44</th>
<th>PDIP40</th>
<th>LQFP32</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>7</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Class</td>
<td>Instruction</td>
<td></td>
</tr>
<tr>
<td>P1.1</td>
<td>I/O</td>
<td>Standard IO Pins</td>
<td></td>
</tr>
<tr>
<td>TxD2</td>
<td>O</td>
<td>Serial Port 2 Transport Pin</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Class</td>
<td>Instruction</td>
<td></td>
</tr>
<tr>
<td>P1.2</td>
<td>I/O</td>
<td>Standard IO Pins</td>
<td></td>
</tr>
<tr>
<td>SS</td>
<td>I</td>
<td>SPI Host output slave input</td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>I</td>
<td>Timer 2 external clock input</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Class</td>
<td>Instruction</td>
<td></td>
</tr>
<tr>
<td>P1.3</td>
<td>I/O</td>
<td>Standard IO Pins</td>
<td></td>
</tr>
<tr>
<td>MOSI</td>
<td>I/O</td>
<td>SPI master output slave input</td>
<td></td>
</tr>
<tr>
<td>T2CLKO</td>
<td>O</td>
<td>Timer 2 clock frequency output</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Class</td>
<td>Instruction</td>
<td></td>
</tr>
<tr>
<td>MISO</td>
<td>I/O</td>
<td>SPI master input slave output</td>
<td></td>
</tr>
<tr>
<td>SDA</td>
<td>I/O</td>
<td>I2C interface data line</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Class</td>
<td>Instruction</td>
<td></td>
</tr>
<tr>
<td>RD</td>
<td>O</td>
<td>External bus read signal line</td>
<td></td>
</tr>
<tr>
<td>TxD_4</td>
<td>O</td>
<td>Serial Port 1 Transport Pin</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>11</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Class</td>
<td>Instruction</td>
<td></td>
</tr>
<tr>
<td>P1.5</td>
<td>I/O</td>
<td>Standard IO Pins</td>
<td></td>
</tr>
<tr>
<td>SCLK</td>
<td>I/O</td>
<td>SPI Clock pin</td>
<td></td>
</tr>
<tr>
<td>SCL</td>
<td>I/O</td>
<td>I2C Clock pin</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>12</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Class</td>
<td>Instruction</td>
<td></td>
</tr>
<tr>
<td>RxD_3</td>
<td>I</td>
<td>Serial Port 1 Receive Pin</td>
<td></td>
</tr>
<tr>
<td>XTALO</td>
<td>O</td>
<td>Output pin of external crystal</td>
<td></td>
</tr>
<tr>
<td>MCLKO_2</td>
<td>O</td>
<td>Main clock frequency output</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>13</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Class</td>
<td>Instruction</td>
<td></td>
</tr>
<tr>
<td>P1.7</td>
<td>I/O</td>
<td>Standard IO Pins</td>
<td></td>
</tr>
<tr>
<td>TxD_3</td>
<td>O</td>
<td>Serial Port 1 Transport Pin</td>
<td></td>
</tr>
<tr>
<td>XTAL1</td>
<td>I</td>
<td>External crystal/external clock input pin</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Class</td>
<td>Instruction</td>
<td></td>
</tr>
<tr>
<td>P4.5</td>
<td>I/O</td>
<td>Standard IO Pins</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Class</td>
<td>Instruction</td>
<td></td>
</tr>
<tr>
<td>P4.6</td>
<td>I/O</td>
<td>Standard IO Pins</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Class</td>
<td>Instruction</td>
<td></td>
</tr>
<tr>
<td>P4.7</td>
<td>I/O</td>
<td>Standard IO Pins</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>17</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Class</td>
<td>Instruction</td>
<td></td>
</tr>
<tr>
<td>P5.4</td>
<td>I/O</td>
<td>Standard IO Pins</td>
<td></td>
</tr>
<tr>
<td>RST</td>
<td>I</td>
<td>Reset pin</td>
<td></td>
</tr>
<tr>
<td>MCLKO</td>
<td>O</td>
<td>Main clock frequency output</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>18</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Class</td>
<td>Instruction</td>
<td></td>
</tr>
<tr>
<td>Vcc</td>
<td>VCC</td>
<td>VCC</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>LQFP44</td>
<td>PDIP40</td>
<td>LQFP32</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>15</td>
<td>19</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>20</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>21</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>22</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>23</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>24</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>25</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>26</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>27</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>LQFP44</td>
<td>PDIP40</td>
<td>LQFP32</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>25</td>
<td>28</td>
<td>20</td>
<td>P3.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INT3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RD_2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TxD_2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CMP+</td>
</tr>
<tr>
<td>26</td>
<td>29</td>
<td></td>
<td>P4.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ALE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CMPO_2</td>
</tr>
<tr>
<td>27</td>
<td>30</td>
<td>21</td>
<td>P2.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RSTSV</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td>P4.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RD_3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TxD2</td>
</tr>
<tr>
<td>29</td>
<td>31</td>
<td>22</td>
<td>P2.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A9</td>
</tr>
<tr>
<td>30</td>
<td>32</td>
<td>23</td>
<td>P2.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SS_2</td>
</tr>
<tr>
<td>31</td>
<td>33</td>
<td>24</td>
<td>P2.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MOSI_2</td>
</tr>
<tr>
<td>32</td>
<td>34</td>
<td>25</td>
<td>P2.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MISO_2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SDA_2</td>
</tr>
<tr>
<td>33</td>
<td>35</td>
<td>26</td>
<td>P2.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A13</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SCLK_2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SCL_2</td>
</tr>
<tr>
<td>34</td>
<td>36</td>
<td>27</td>
<td>P2.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A14</td>
</tr>
<tr>
<td>35</td>
<td>37</td>
<td>28</td>
<td>P2.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A15</td>
</tr>
</tbody>
</table>
### 3.3 Function Pin Switch

#### 3.3.1 Function pin related register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>addr</th>
<th>Bit address and symbol</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUS_SPEED</td>
<td>Bus speed control register</td>
<td>A1H</td>
<td>RW_S[1:0]</td>
<td>SPEED[1:0]</td>
</tr>
<tr>
<td>P_SW1</td>
<td>Peripheral Port Switch Register 1</td>
<td>A2H</td>
<td>S1_S[1:0]</td>
<td>0</td>
</tr>
<tr>
<td>P_SW2</td>
<td>Peripheral Port Switch Register 2</td>
<td>BAH</td>
<td>I2C_S[1:0]</td>
<td>S2_S</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Nantong guoxin Microelectronics Co., Ltd. Tel: 0513-5501 2928/2929/2966 Fax: 0513-5501 2926/2956/2947
### Bit address and symbol

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>addr</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWM0CR</td>
<td>PWM0 control register</td>
<td>FF04H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td>PWM1CR</td>
<td>PWM1 control register</td>
<td>FF14H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td>PWM2CR</td>
<td>PWM2 control register</td>
<td>FF24H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td>PWM3CR</td>
<td>PWM3 control register</td>
<td>FF34H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td>PWM4CR</td>
<td>PWM4 control register</td>
<td>FF44H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td>PWM5CR</td>
<td>PWM5 control register</td>
<td>FF54H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td>PWM6CR</td>
<td>PWM6 control register</td>
<td>FF64H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td>PWM7CR</td>
<td>PWM7 control register</td>
<td>FF74H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000000</td>
</tr>
<tr>
<td>CKSEL</td>
<td>Clock select register</td>
<td>FE00H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10000000</td>
</tr>
</tbody>
</table>

### Bus speed control register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>addr</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUS_SPEED</td>
<td>A1H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SPEED[1:0]</td>
</tr>
</tbody>
</table>

### RW_S[1:0]: External bus RD/WR controlling choosing bit

<table>
<thead>
<tr>
<th>RW_S[1:0]</th>
<th>RD</th>
<th>WR</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>P4.4</td>
<td>P4.3</td>
</tr>
<tr>
<td>01</td>
<td>P3.7</td>
<td>P3.6</td>
</tr>
<tr>
<td>10</td>
<td>P4.2</td>
<td>P4.0</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Periphery Port Switch Control Register 1

<table>
<thead>
<tr>
<th>Symbol</th>
<th>addr</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_SW1</td>
<td>A2H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

### S1_S[1:0]: Serial Port 1 Function pin select bit

<table>
<thead>
<tr>
<th>S1_S[1:0]</th>
<th>RxD</th>
<th>TxD</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>P3.0</td>
<td>P3.1</td>
</tr>
<tr>
<td>01</td>
<td>P3.6</td>
<td>P3.7</td>
</tr>
<tr>
<td>10</td>
<td>P1.6</td>
<td>P1.7</td>
</tr>
<tr>
<td>11</td>
<td>P4.3</td>
<td>P4.4</td>
</tr>
</tbody>
</table>

### CCP_S[1:0]: PCA Function pin select bit

<table>
<thead>
<tr>
<th>CCP_S[1:0]</th>
<th>EC1</th>
<th>CCP0</th>
<th>CCP1</th>
<th>CCP2</th>
<th>CCP3</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>P1.2</td>
<td>P1.7</td>
<td>P1.6</td>
<td>P1.5</td>
<td>P1.4</td>
</tr>
<tr>
<td>01</td>
<td>P2.2</td>
<td>P2.3</td>
<td>P2.4</td>
<td>P2.5</td>
<td>P2.6</td>
</tr>
<tr>
<td>10</td>
<td>P7.4</td>
<td>P7.0</td>
<td>P7.1</td>
<td>P7.2</td>
<td>P7.3</td>
</tr>
<tr>
<td>11</td>
<td>P3.5</td>
<td>P3.3</td>
<td>P3.2</td>
<td>P3.1</td>
<td>P3.0</td>
</tr>
</tbody>
</table>

### SPI_S[1:0]: SPI Function pin select bit
### Periphery Port Switch Control Register 2

<table>
<thead>
<tr>
<th>Symbol</th>
<th>addr</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Periphery Port Switch Control Register 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### I2C_S[1:0]: I2C Function pin select bit

<table>
<thead>
<tr>
<th>I2C_S[1:0]</th>
<th>SCL</th>
<th>SDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>P1.5</td>
<td>P1.4</td>
</tr>
<tr>
<td>01</td>
<td>P2.5</td>
<td>P2.4</td>
</tr>
<tr>
<td>10</td>
<td>P7.7</td>
<td>P7.6</td>
</tr>
<tr>
<td>11</td>
<td>P3.2</td>
<td>P3.3</td>
</tr>
</tbody>
</table>

#### CMPO_S: Comparator output Function pin select bit

<table>
<thead>
<tr>
<th>CMPO_S</th>
<th>CMPO</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>P3.4</td>
</tr>
<tr>
<td>1</td>
<td>P4.1</td>
</tr>
</tbody>
</table>

#### S4_S: Serial Port 4 Function pin select bit

<table>
<thead>
<tr>
<th>S4_S</th>
<th>RxD4</th>
<th>TxD4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>P0.2</td>
<td>P0.3</td>
</tr>
<tr>
<td>1</td>
<td>P5.2</td>
<td>P5.3</td>
</tr>
</tbody>
</table>

#### S3_S: Serial Port 1 Function pin select bit

<table>
<thead>
<tr>
<th>S3_S</th>
<th>RxD3</th>
<th>TxD3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>P0.0</td>
<td>P0.1</td>
</tr>
<tr>
<td>1</td>
<td>P5.0</td>
<td>P5.1</td>
</tr>
</tbody>
</table>

#### S2_S: Serial Port 2 Function pin select bit

<table>
<thead>
<tr>
<th>S2_S</th>
<th>RxD2</th>
<th>TxD2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>P1.0</td>
<td>P1.1</td>
</tr>
<tr>
<td>1</td>
<td>P4.0</td>
<td>P4.2</td>
</tr>
</tbody>
</table>

### Clock select register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>addr</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock select register</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### MCLKO_S: Main clock output pin select bit

<table>
<thead>
<tr>
<th>MCLKO_S</th>
<th>MCLKO</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>P5.4</td>
</tr>
<tr>
<td>1</td>
<td>P1.6</td>
</tr>
</tbody>
</table>

### Enhanced PWM control register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>addr</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enhanced PWM control register</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM0CR</td>
<td>FF04H</td>
<td>ENC0O</td>
<td>C0INI</td>
<td>-</td>
<td>C0_S[1:0]</td>
<td>EC0I</td>
<td>EC0T2SI</td>
<td>EC0T1SI</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>---</td>
<td>---------</td>
<td>------</td>
<td>---------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>PWM1CR</td>
<td>FF14H</td>
<td>ENC1O</td>
<td>C1INI</td>
<td>-</td>
<td>C1_S[1:0]</td>
<td>EC1I</td>
<td>EC1T2SI</td>
<td>EC1T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM2CR</td>
<td>FF24H</td>
<td>ENC2O</td>
<td>C2INI</td>
<td>-</td>
<td>C2_S[1:0]</td>
<td>EC2I</td>
<td>EC2T2SI</td>
<td>EC2T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM3CR</td>
<td>FF34H</td>
<td>ENC3O</td>
<td>C3INI</td>
<td>-</td>
<td>C3_S[1:0]</td>
<td>EC3I</td>
<td>EC3T2SI</td>
<td>EC3T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM4CR</td>
<td>FF44H</td>
<td>ENC4O</td>
<td>C4INI</td>
<td>-</td>
<td>C4_S[1:0]</td>
<td>EC4I</td>
<td>EC4T2SI</td>
<td>EC4T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM5CR</td>
<td>FF54H</td>
<td>ENC5O</td>
<td>C5INI</td>
<td>-</td>
<td>C5_S[1:0]</td>
<td>EC5I</td>
<td>EC5T2SI</td>
<td>EC5T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM6CR</td>
<td>FF64H</td>
<td>ENC6O</td>
<td>C6INI</td>
<td>-</td>
<td>C6_S[1:0]</td>
<td>EC6I</td>
<td>EC6T2SI</td>
<td>EC6T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM7CR</td>
<td>FF74H</td>
<td>ENC7O</td>
<td>C7INI</td>
<td>-</td>
<td>C7_S[1:0]</td>
<td>EC7I</td>
<td>EC7T2SI</td>
<td>EC7T1SI</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>C0_S[1:0]</th>
<th>PWM0</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>P2.0</td>
</tr>
<tr>
<td>01</td>
<td>P1.0</td>
</tr>
<tr>
<td>10</td>
<td>P6.0</td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>C1_S[1:0]</th>
<th>PWM1</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>P2.1</td>
</tr>
<tr>
<td>01</td>
<td>P1.1</td>
</tr>
<tr>
<td>10</td>
<td>P6.1</td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>C2_S[1:0]</th>
<th>PWM2</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>P2.2</td>
</tr>
<tr>
<td>01</td>
<td>P1.2</td>
</tr>
<tr>
<td>10</td>
<td>P6.2</td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>C3_S[1:0]</th>
<th>PWM3</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>P2.3</td>
</tr>
<tr>
<td>01</td>
<td>P1.3</td>
</tr>
<tr>
<td>10</td>
<td>P6.3</td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>C4_S[1:0]</th>
<th>PWM4</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>P2.4</td>
</tr>
<tr>
<td>01</td>
<td>P1.4</td>
</tr>
<tr>
<td>10</td>
<td>P6.4</td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>C5_S[1:0]</th>
<th>PWM5</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>P2.5</td>
</tr>
<tr>
<td>01</td>
<td>P1.5</td>
</tr>
<tr>
<td>C6_S[1:0]</td>
<td>PWM6</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C7_S[1:0]</th>
<th>PWM7</th>
<th>00</th>
<th>P2.7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>01</td>
<td>P1.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>P6.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

3.4 Sample Program

3.4.1 Serial 1 switch

Assembly code

```
P_SW1   DATA  0A2H

ORG 0000H
LJMP MAIN

ORG 0100H

MAIN:

MOV SP, #3FH

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

SJMP $

END
```

C Code

```
#include "reg51.h"

sfr P_SW1 = 0xa2;

void main()
{
    P_SW1 = 0x00; //RXD/P3.0, TXD/P3.1
    // P_SW1 = 0x40; //RXD_2/P3.6, TXD_2/P3.7
    // P_SW1 = 0x80; //RXD_3/P1.6, TXD_3/P1.7
    // P_SW1 = 0xC0; //RXD_4/P4.3, TXD_4/P4.4
}
```
3.4.2 Serial 2 switch

Assembly code

```
P_SW2 DATA 0BAH

ORG 0000H
LJMP MAIN

ORG 0100H
MAIN:
    MOV SP, #3FH
    MOV P_SW2, #00H ; RXD2/P1.0, TXD2/P1.1
    MOV P_SW2, #01H ; RXD2_2/P4.0, TXD2_2/P4.2
    SJMP $

END
```

C Code

```
#include "reg51.h"

sfr P_SW2 = 0xba;

void main()
{
    P_SW2 = 0x00; // RXD2/P1.0, TXD2/P1.1
    // P_SW2 = 0x01; // RXD2_2/P4.0, TXD2_2/P4.2
    while (1);
}
```

3.4.3 Serial 3 switch

Assembly code

```
P_SW2 DATA 0BAH

ORG 0000H
LJMP MAIN

ORG 0100H
MAIN:
    MOV SP, #3FH
    MOV P_SW2, #00H ; RXD3/P0.0, TXD3/P0.1
    MOV P_SW2, #02H ; RXD3_2/P5.0, TXD3_2/P5.1
    SJMP $

END
```

C Code

```
#include "reg51.h"

sfr P_SW2 = 0xba;

void main()
{
    P_SW2 = 0x00; // RXD3/P0.0, TXD3/P0.1
    // P_SW2 = 0x02; // RXD3_2/P5.0, TXD3_2/P5.1
    while (1);
}
```
3.4.4 Serial 4 switch

Assembly code

<table>
<thead>
<tr>
<th>P_SW2</th>
<th>DATA</th>
<th>0BAH</th>
</tr>
</thead>
</table>

end

mov sp , #3fh
mov p_sw2,#00h ;rxd4/p0.2, txd4/p0.3
;mov p_sw2,#04h ;rxd4_2/p5.2, txd4_2/p5.3

SJMP $

END

C CODE

#include "reg51.h"
sfr P_SW2 = 0xba;

void main()
{
    P_SW2 = 0x00; //RXD3/P0.0, TXD3/P0.1
    // P_SW2 = 0x02; //RXD3_2/P5.0, TXD3_2/P5.1

    while (1);
}

3.4.5 SPI switch

Assembly code

| P_SW1   | DATA | 0A2H |

end

mov sp , #3fh
mov p_sw1,#00h ;rxd4/p0.2, txd4/p0.3
;mov p_sw1,#04h ;rxd4_2/p5.2, txd4_2/p5.3

SJMP $

END

C CODE

#include "reg51.h"
sfr P_SW2 = 0xba;

void main()
{
    P_SW2 = 0x00; //RXD4/P0.2, TXD4/P0.3
    // P_SW2 = 0x04; //RXD4_2/P5.2, TXD4_2/P5.3

    while (1);
}
ORG 0000H
LJMP MAIN

ORG 0100H

MAIN:
MOV SP, #3FH

MOV P_SW1,#00H ;SS/P1.2, MOSI/P1.3, MISO/P1.4, SCLK/P1.5
; MOV P_SW1,#04H ;SS_2/P2.2, MOSI_2/P2.3, MISO_2/P2.4, SCLK_2/P2.5
; MOV P_SW1,#08H ;SS_3/P7.4, MOSI_3/P7.5, MISO_3/P7.6, SCLK_3/P7.7
; MOV P_SW1,#0CH ;SS_4/P3.5, MOSI_4/P3.4, MISO_4/P3.3, SCLK_4/P3.2

SJMP $

END

C CODE

#include "reg51.h"

sfr P_SW1 = 0xa2;

void main()
{
    P_SW1 = 0x00; //SS/P1.2, MOSI/P1.3, MISO/P1.4, SCLK/P1.5
    // P_SW1 = 0x04; //SS_2/P2.2, MOSI_2/P2.3, MISO_2/P2.4, SCLK_2/P2.5
    // P_SW1 = 0x08; //SS_3/P7.4, MOSI_3/P7.5, MISO_3/P7.6, SCLK_3/P7.7
    // P_SW1 = 0x0c; //SS_4/P3.5, MOSI_4/P3.4, MISO_4/P3.3, SCLK_4/P3.2

    while (1);
}

3.4.6 PWM switch

Assembly code

P_SW2 DATA 0BAH
PWM0CR EQU 0FF04H
PWM1CR EQU 0FF14H
PWM2CR EQU 0FF24H
PWM3CR EQU 0FF34H
PWM4CR EQU 0FF44H
PWM5CR EQU 0FF54H
PWM6CR EQU 0FF64H
PWM7CR EQU 0FF74H

ORG 0000H
LJMP MAIN

ORG 0100H

MAIN:
MOV SP, #3FH

MOV P_SW2,#80H
MOV A,#00H ;PWM0/P2.0
; MOV A,#08H ;PWM0_2/P1.0
; MOV A,#10H ;PWM0_3/P6.0
MOV DPTR,#PWM0CR
MOVX @DPTR,A
MOV A,#00H ;PWM1/P2.1
; MOV A,#08H ;PWM1_2/P1.1
; MOV A,#10H ;PWM1_3/P6.1
MOV DPTR,#PWM1CR
MOVX @DPTR,A
MOV A,#00H ;PWM2/P2.2
; MOV A,#08H ;PWM2_2/P1.2
; MOV A,#10H ;PWM2_3/P6.2
MOV DPTR,#PWM2CR
MOVX @DPTR,A
MOV A,#00H ;PWM3/P2.3
; MOV A,#08H ;PWM3_2/P1.3
; MOV A,#10H ;PWM3_3/P6.3
MOV DPTR,#PWM3CR
MOVX @DPTR,A
MOV A,#00H ;PWM4/P2.4
; MOV A,#08H ;PWM4_2/P1.4
; MOV A,#10H ;PWM4_3/P6.4
MOV DPTR,#PWM4CR
MOVX @DPTR,A
MOV A,#00H ;PWM5/P2.5
; MOV A,#08H ;PWM5_2/P1.5
; MOV A,#10H ;PWM5_3/P6.5
MOV DPTR,#PWM5CR
MOVX @DPTR,A
MOV A,#00H ;PWM6/P2.6
; MOV A,#08H ;PWM6_2/P1.6
; MOV A,#10H ;PWM6_3/P6.6
MOV DPTR,#PWM6CR
MOVX @DPTR,A
MOV A,#00H ;PWM7/P2.7
; MOV A,#08H ;PWM7_2/P1.7
; MOV A,#10H ;PWM7_3/P6.7
MOV DPTR,#PWM7CR
MOVX @DPTR,A
MOV P_SW2,#00H
SJMP $
END

C CODE
#include "reg51.h"
#define PWM0CR (*(unsigned char volatile xdata *)0xff04)
#define PWM1CR (*(unsigned char volatile xdata *)0xff14)
#define PWM2CR (*(unsigned char volatile xdata *)0xff24)
#define PWM3CR (*(unsigned char volatile xdata *)0xff34)
#define PWM4CR (*(unsigned char volatile xdata *)0xff44)
#define PWM5CR (*(unsigned char volatile xdata *)0xff54)
#define PWM6CR (*(unsigned char volatile xdata *)0xff64)
#define PWM7CR (*(unsigned char volatile xdata *)0xff74)
sfr P_SW2 = 0xba;
void main()
{
    P_SW2 = 0x80;
    // PWM0CR = 0x00; //PWM0/P2.0
    // PWM0CR = 0x08; //PWM0_2/P1.0
    // PWM0CR = 0x10; //PWM0_3/P6.0
    PWM1CR = 0x00; //PWM1/P2.1
    // PWM1CR = 0x08; //PWM1_2/P1.1
    // PWM1CR = 0x10; //PWM1_3/P6.1
    PWM2CR = 0x00; //PWM2/P2.2
    // PWM2CR = 0x08; //PWM2_2/P1.2
    // PWM2CR = 0x10; //PWM2_3/P6.2
    PWM3CR = 0x00; //PWM3/P2.3
    // PWM3CR = 0x08; //PWM3_2/P1.3
    // PWM3CR = 0x10; //PWM3_3/P6.3
    PWM4CR = 0x00; //PWM4/P2.4
    // PWM4CR = 0x08; //PWM4_2/P1.4
    // PWM4CR = 0x10; //PWM4_3/P6.4
    PWM5CR = 0x00; //PWM5/P2.5
    // PWM5CR = 0x08; //PWM5_2/P1.5
    // PWM5CR = 0x10; //PWM5_3/P6.5
    PWM6CR = 0x00; //PWM6/P2.6
    // PWM6CR = 0x08; //PWM6_2/P1.6
    // PWM6CR = 0x10; //PWM6_3/P6.6
    PWM7CR = 0x00; //PWM7/P2.7
    // PWM7CR = 0x08; //PWM7_2/P1.7
    // PWM7CR = 0x10; //PWM7_3/P6.7
    P_SW2 = 0x00;
    while (1);
}

3.4.7 PCA/CCP/PWM switch

Assembly code

C CODE

#include "reg51.h"

Nantong guoxin Microelectronics Co., Ltd. Tel: 0513-5501 2928/2929/2966 Fax: 0513-5501 2926/2956/2947 - 57 -
sfr P_SW1 = 0xa2;

void main()
{
    P_SW1 = 0x00;  //ECI/P1.2, CCP0/P1.7, CCP1/P1.6, CCP2/P1.5, CCP3/P1.4
    // P_SW1 = 0x10;  //ECI_2/P2.2, CCP0_2/P2.3, CCP1_2/P2.4, CCP2_2/P2.5, CCP3_2/P2.6
    // P_SW1 = 0x20;  //ECI_3/P7.4, CCP0_3/P7.0, CCP1_3/P7.1, CCP2_3/P7.2, CCP3_3/P7.3
    // P_SW1 = 0x30;  //ECI_4/P3.5, CCP0_4/P3.3, CCP1_4/P3.2, CCP2_4/P3.1, CCP3_4/P3.0
    while (1);
}

3.4.8 I2C switch

Assembly code

P_SW2 DATA 0BAH

ORG 0000H
LJMP MAIN

ORG 0100H

MAIN:

MOV SP, #3FH

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

SJMP $

END

C CODE

#include "reg51.h"

sfr P_SW2 = 0xba;

void main()
{
    P_SW2 = 0x00;  //SCL/P1.5, SDA/P1.4
    // P_SW2 = 0x10;  //SCL_2/P2.5, SDA_2/P2.4
    // P_SW2 = 0x20;  //SCL_3/P7.7, SDA_3/P7.6
    // P_SW2 = 0x30;  //SCL_4/P3.2, SDA_4/P3.3
    while (1);
}

3.4.9 Comparator output switch

Assembly code

P_SW2 DATA 0BAH
ORG 0000H
LJMP MAIN

ORG 0100H

MAIN:
MOV SP, #3FH

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

SJMP $  

END

C CODE

#include "reg51.h"

sfr P_SW2 = 0xba;

void main()
{
P_SW2 = 0x00; //CMPO/P3.4
// P_SW2 = 0x08; //CMPO_2/P4.1
while (1);
}

3.4.10 Master clock output switching

Assembly code

<table>
<thead>
<tr>
<th>DATA</th>
<th>0BAH</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQU</td>
<td>0FE00H</td>
</tr>
</tbody>
</table>

ORG 0000H
LJMP MAIN

ORG 0100H

MAIN:
MOV SP, #3FH

MOV P_SW2, #80H ;IRC24M/4 output via MCLKO/P5.4
MOV A, #40H ;IRC24M/4 output via MCLKO_2/P1.6
MOV A, #48H ;IRC24M/128 output via MCLKO_2/P1.6
MOV DPTR, #CKSEL
MOVX @DPTR, A

SJMP $  

END

C CODE

#include "reg51.h"
#define CKSEL (*(unsigned char volatile xdata *)0xfe00)

sfr P_SW2 = 0xba;

void main()
{
    P_SW2 = 0x80;
    CKSEL = 0x40;  // IRC24M/4 output via MCLKO/P5.4
    // CKSEL = 0x48;  // IRC24M/4 output via MCLKO_2/P1.6
    // CKSEL = 0xe8;  // IRC24M/128 output via MCLKO_2/P1.6
    P_SW2 = 0x00;

    while (1);
}
4 Package characteristics

4.1 LQFP64S package mechanical data (12mm*12mm)
4.2 LQFP64L package mechanical data (16mm*16mm)

STC8 series does not have this package.

---

### General Size

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-</td>
<td>-</td>
<td>1.60</td>
</tr>
<tr>
<td>A1</td>
<td>0.05</td>
<td>-</td>
<td>0.15</td>
</tr>
<tr>
<td>A2</td>
<td>1.35</td>
<td>1.40</td>
<td>1.45</td>
</tr>
<tr>
<td>A3</td>
<td>0.59</td>
<td>0.64</td>
<td>0.69</td>
</tr>
<tr>
<td>b</td>
<td>0.31</td>
<td>-</td>
<td>0.44</td>
</tr>
<tr>
<td>b1</td>
<td>0.30</td>
<td>0.35</td>
<td>0.40</td>
</tr>
<tr>
<td>c</td>
<td>0.13</td>
<td>-</td>
<td>0.18</td>
</tr>
<tr>
<td>c1</td>
<td>0.12</td>
<td>0.127</td>
<td>0.134</td>
</tr>
<tr>
<td>D</td>
<td>15.80</td>
<td>16.00</td>
<td>16.20</td>
</tr>
<tr>
<td>D1</td>
<td>13.90</td>
<td>14.00</td>
<td>14.10</td>
</tr>
<tr>
<td>E</td>
<td>15.80</td>
<td>16.00</td>
<td>16.20</td>
</tr>
<tr>
<td>E1</td>
<td>13.90</td>
<td>14.00</td>
<td>14.10</td>
</tr>
<tr>
<td>e</td>
<td>0.70</td>
<td>0.80</td>
<td>0.90</td>
</tr>
<tr>
<td>L</td>
<td>0.45</td>
<td>0.60</td>
<td>0.75</td>
</tr>
<tr>
<td>L1</td>
<td>1.00REF</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>L2</td>
<td>0.25BSC</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>R1</td>
<td>0.08</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>R2</td>
<td>0.08</td>
<td>-</td>
<td>0.20</td>
</tr>
<tr>
<td>S</td>
<td>0.20</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>a</td>
<td>0°</td>
<td>3.5°</td>
<td>7°</td>
</tr>
<tr>
<td>a1</td>
<td>11°</td>
<td>12°</td>
<td>13°</td>
</tr>
<tr>
<td>a2</td>
<td>11°</td>
<td>12°</td>
<td>13°</td>
</tr>
</tbody>
</table>

---

At the bottom left of the chip silk screen is the first foot.
4.3 LQFP48 package mechanical data (9mm*9mm)

At the bottom left of the chip silk screen is the first foot.
D (9mm)  D1 (7mm)

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-</td>
<td>-</td>
<td>1.60</td>
</tr>
<tr>
<td>A1</td>
<td>0.05</td>
<td>-</td>
<td>0.15</td>
</tr>
<tr>
<td>A2</td>
<td>1.35</td>
<td>1.40</td>
<td>1.45</td>
</tr>
<tr>
<td>A3</td>
<td>0.59</td>
<td>0.64</td>
<td>0.69</td>
</tr>
<tr>
<td>b</td>
<td>0.18</td>
<td>-</td>
<td>0.27</td>
</tr>
<tr>
<td>b1</td>
<td>0.30</td>
<td>0.35</td>
<td>0.40</td>
</tr>
<tr>
<td>c</td>
<td>0.13</td>
<td>-</td>
<td>0.18</td>
</tr>
<tr>
<td>c1</td>
<td>0.12</td>
<td>0.127</td>
<td>0.134</td>
</tr>
<tr>
<td>D</td>
<td>8.80</td>
<td>9.00</td>
<td>9.20</td>
</tr>
<tr>
<td>D1</td>
<td>6.90</td>
<td>7.00</td>
<td>7.10</td>
</tr>
<tr>
<td>E</td>
<td>8.80</td>
<td>9.00</td>
<td>9.20</td>
</tr>
<tr>
<td>E1</td>
<td>6.90</td>
<td>7.00</td>
<td>7.10</td>
</tr>
<tr>
<td>e</td>
<td>0.45</td>
<td>0.50</td>
<td>0.55</td>
</tr>
<tr>
<td>L</td>
<td>0.45</td>
<td>0.60</td>
<td>0.75</td>
</tr>
<tr>
<td>L1</td>
<td>1.00REF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L2</td>
<td>0.25BSC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R1</td>
<td>0.08</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>R2</td>
<td>0.08</td>
<td>-</td>
<td>0.20</td>
</tr>
<tr>
<td>S</td>
<td>0.20</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

(units of measurements: mm)

(interface diagram A-A)
### general size

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-</td>
<td>-</td>
<td>1.60</td>
</tr>
<tr>
<td>A1</td>
<td>0.05</td>
<td>-</td>
<td>0.15</td>
</tr>
<tr>
<td>A2</td>
<td>1.35</td>
<td>1.40</td>
<td>1.45</td>
</tr>
<tr>
<td>A3</td>
<td>0.59</td>
<td>0.64</td>
<td>0.69</td>
</tr>
<tr>
<td>b</td>
<td>0.18</td>
<td>-</td>
<td>0.27</td>
</tr>
<tr>
<td>b1</td>
<td>0.30</td>
<td>0.35</td>
<td>0.40</td>
</tr>
<tr>
<td>c</td>
<td>0.13</td>
<td>-</td>
<td>0.18</td>
</tr>
<tr>
<td>c1</td>
<td>0.12</td>
<td>0.127</td>
<td>0.134</td>
</tr>
<tr>
<td>D</td>
<td>8.80</td>
<td>9.00</td>
<td>9.20</td>
</tr>
<tr>
<td>D1</td>
<td>6.90</td>
<td>7.00</td>
<td>7.10</td>
</tr>
<tr>
<td>E</td>
<td>8.80</td>
<td>9.00</td>
<td>9.20</td>
</tr>
<tr>
<td>E1</td>
<td>6.90</td>
<td>7.00</td>
<td>7.10</td>
</tr>
<tr>
<td>e</td>
<td>0.45</td>
<td>0.50</td>
<td>0.55</td>
</tr>
<tr>
<td>L</td>
<td>0.45</td>
<td>0.60</td>
<td>0.75</td>
</tr>
<tr>
<td>L1</td>
<td>1.00REF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L2</td>
<td>0.25BSC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R1</td>
<td>0.08</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>R2</td>
<td>0.08</td>
<td>-</td>
<td>0.20</td>
</tr>
<tr>
<td>S</td>
<td>0.20</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**units of measurement:** mm
4.4 LQFP44 package mechanical data (12mm*12mm)

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-</td>
<td>-</td>
<td>1.80</td>
</tr>
<tr>
<td>A1</td>
<td>0.05</td>
<td>-</td>
<td>0.15</td>
</tr>
<tr>
<td>A2</td>
<td>1.35</td>
<td>1.40</td>
<td>1.45</td>
</tr>
<tr>
<td>A3</td>
<td>0.59</td>
<td>0.64</td>
<td>0.69</td>
</tr>
<tr>
<td>b</td>
<td>0.25</td>
<td>0.30</td>
<td>0.35</td>
</tr>
<tr>
<td>c1</td>
<td>0.09</td>
<td>-</td>
<td>0.16</td>
</tr>
<tr>
<td>D</td>
<td>11.80</td>
<td>12.00</td>
<td>12.20</td>
</tr>
<tr>
<td>D1</td>
<td>9.90</td>
<td>10.00</td>
<td>10.10</td>
</tr>
<tr>
<td>E</td>
<td>11.80</td>
<td>12.00</td>
<td>12.20</td>
</tr>
<tr>
<td>E1</td>
<td>9.90</td>
<td>10.00</td>
<td>10.10</td>
</tr>
<tr>
<td>e</td>
<td>0.70</td>
<td>0.80</td>
<td>0.90</td>
</tr>
<tr>
<td>L</td>
<td>0.45</td>
<td>0.60</td>
<td>0.75</td>
</tr>
<tr>
<td>L1</td>
<td>1.00</td>
<td>REF</td>
<td></td>
</tr>
<tr>
<td>L2</td>
<td>0.25</td>
<td>BSC</td>
<td></td>
</tr>
<tr>
<td>R1</td>
<td>0.08</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>R2</td>
<td>0.08</td>
<td>-</td>
<td>0.20</td>
</tr>
</tbody>
</table>

Notes:
- The general size is 12mm x 12mm.
- Units of measurement: mm.
- At the bottom left of the chip silk screen is the first foot.
at the bottom left of the chip silk screen is the first foot

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-</td>
<td>-</td>
<td>1.60</td>
</tr>
<tr>
<td>A1</td>
<td>0.05</td>
<td>-</td>
<td>0.15</td>
</tr>
<tr>
<td>A2</td>
<td>1.35</td>
<td>1.40</td>
<td>1.45</td>
</tr>
<tr>
<td>A3</td>
<td>0.59</td>
<td>0.64</td>
<td>0.69</td>
</tr>
<tr>
<td>b</td>
<td>0.25</td>
<td>0.30</td>
<td>0.35</td>
</tr>
<tr>
<td>c1</td>
<td>0.09</td>
<td>-</td>
<td>0.16</td>
</tr>
<tr>
<td>D</td>
<td>11.80</td>
<td>12.00</td>
<td>12.20</td>
</tr>
<tr>
<td>D1</td>
<td>9.90</td>
<td>10.00</td>
<td>10.10</td>
</tr>
<tr>
<td>E</td>
<td>11.80</td>
<td>12.00</td>
<td>12.20</td>
</tr>
<tr>
<td>E1</td>
<td>9.90</td>
<td>10.00</td>
<td>10.10</td>
</tr>
<tr>
<td>e</td>
<td>0.70</td>
<td>0.80</td>
<td>0.90</td>
</tr>
<tr>
<td>L</td>
<td>0.45</td>
<td>0.60</td>
<td>0.75</td>
</tr>
<tr>
<td>L1</td>
<td>1.00REF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L2</td>
<td>0.25BSC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R1</td>
<td>0.08</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>R2</td>
<td>0.08</td>
<td>-</td>
<td>0.20</td>
</tr>
</tbody>
</table>
at the bottom left of the chip silk screen is the first foot

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-</td>
<td>-</td>
<td>1.60</td>
</tr>
<tr>
<td>A1</td>
<td>0.05</td>
<td>-</td>
<td>0.15</td>
</tr>
<tr>
<td>A2</td>
<td>1.35</td>
<td>1.40</td>
<td>1.45</td>
</tr>
<tr>
<td>A3</td>
<td>0.59</td>
<td>0.64</td>
<td>0.69</td>
</tr>
<tr>
<td>b</td>
<td>0.25</td>
<td>0.30</td>
<td>0.35</td>
</tr>
<tr>
<td>c1</td>
<td>0.09</td>
<td>-</td>
<td>0.16</td>
</tr>
<tr>
<td>D</td>
<td>11.80</td>
<td>12.00</td>
<td>12.20</td>
</tr>
<tr>
<td>D1</td>
<td>9.90</td>
<td>10.00</td>
<td>10.10</td>
</tr>
<tr>
<td>E</td>
<td>11.80</td>
<td>12.00</td>
<td>12.20</td>
</tr>
<tr>
<td>E1</td>
<td>9.90</td>
<td>10.00</td>
<td>10.10</td>
</tr>
<tr>
<td>e</td>
<td>0.70</td>
<td>0.80</td>
<td>0.90</td>
</tr>
<tr>
<td>L</td>
<td>0.45</td>
<td>0.60</td>
<td>0.75</td>
</tr>
<tr>
<td>L1</td>
<td>1.00REF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L2</td>
<td>0.25BSC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R1</td>
<td>0.08</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>R2</td>
<td>0.08</td>
<td>-</td>
<td>0.20</td>
</tr>
</tbody>
</table>

units of measurement: mm

*general size*
4.5 LQFP32 package mechanical data (9mm*9mm)

![Diagram of LQFP32 package]

**General Size**

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1.45</td>
<td>1.55</td>
<td>1.65</td>
</tr>
<tr>
<td>A1</td>
<td>0.01</td>
<td>-</td>
<td>0.21</td>
</tr>
<tr>
<td>A2</td>
<td>1.35</td>
<td>1.40</td>
<td>1.45</td>
</tr>
<tr>
<td>A3</td>
<td>-</td>
<td>0.254</td>
<td>-</td>
</tr>
<tr>
<td>b</td>
<td>0.30</td>
<td>0.35</td>
<td>0.40</td>
</tr>
<tr>
<td>b1</td>
<td>0.31</td>
<td>0.37</td>
<td>0.43</td>
</tr>
<tr>
<td>c</td>
<td>-</td>
<td>0.127</td>
<td>-</td>
</tr>
<tr>
<td>D</td>
<td>8.80</td>
<td>9.00</td>
<td>9.20</td>
</tr>
<tr>
<td>D1</td>
<td>6.90</td>
<td>7.00</td>
<td>7.10</td>
</tr>
<tr>
<td>E</td>
<td>8.80</td>
<td>9.00</td>
<td>9.20</td>
</tr>
<tr>
<td>E1</td>
<td>6.90</td>
<td>7.00</td>
<td>7.10</td>
</tr>
<tr>
<td>e</td>
<td>0.70</td>
<td>0.80</td>
<td>0.90</td>
</tr>
<tr>
<td>L</td>
<td>0.43</td>
<td>-</td>
<td>0.71</td>
</tr>
<tr>
<td>L1</td>
<td>1.00REF</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>R</td>
<td>0.1</td>
<td>-</td>
<td>0.25</td>
</tr>
<tr>
<td>R1</td>
<td>0.1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>θ</td>
<td>0°</td>
<td>-</td>
<td>10°</td>
</tr>
</tbody>
</table>

**Units of Measurement:** mm

*At the bottom left of the chip silk screen is the first foot.*
4.6 QFN32 package mechanical data (4mm*4mm)

![Diagram of QFN32 package]

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.70</td>
<td>0.75</td>
<td>0.80</td>
</tr>
<tr>
<td>A1</td>
<td>0</td>
<td>0.02</td>
<td>0.05</td>
</tr>
<tr>
<td>A2</td>
<td>0.50</td>
<td>0.55</td>
<td>0.60</td>
</tr>
<tr>
<td>A3</td>
<td>-</td>
<td>0.20REF</td>
<td>-</td>
</tr>
<tr>
<td>b</td>
<td>0.15</td>
<td>0.20</td>
<td>0.25</td>
</tr>
<tr>
<td>D</td>
<td>3.90</td>
<td>4.00</td>
<td>4.10</td>
</tr>
<tr>
<td>E</td>
<td>3.90</td>
<td>4.00</td>
<td>4.10</td>
</tr>
<tr>
<td>D2</td>
<td>2.60</td>
<td>2.70</td>
<td>2.80</td>
</tr>
<tr>
<td>E2</td>
<td>2.60</td>
<td>2.70</td>
<td>2.80</td>
</tr>
<tr>
<td>e</td>
<td>0.30</td>
<td>0.40</td>
<td>0.50</td>
</tr>
<tr>
<td>L</td>
<td>0.35</td>
<td>0.40</td>
<td>0.45</td>
</tr>
<tr>
<td>K</td>
<td>-</td>
<td>0.25REF</td>
<td>-</td>
</tr>
<tr>
<td>R</td>
<td>0.09</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C1</td>
<td>-</td>
<td>0.16</td>
<td>-</td>
</tr>
<tr>
<td>C2</td>
<td>-</td>
<td>0.16</td>
<td>-</td>
</tr>
</tbody>
</table>
4.7 PDIP40 package mechanical data

![Diagram of PDIP40 package mechanical data]

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td>190</td>
</tr>
<tr>
<td>A1</td>
<td>15</td>
<td>-</td>
<td>20</td>
</tr>
<tr>
<td>A2</td>
<td>150</td>
<td>155</td>
<td>160</td>
</tr>
<tr>
<td>b</td>
<td>45</td>
<td>-</td>
<td>67</td>
</tr>
<tr>
<td>b1</td>
<td>15</td>
<td>-</td>
<td>21</td>
</tr>
<tr>
<td>C</td>
<td>8</td>
<td>-</td>
<td>15</td>
</tr>
<tr>
<td>D</td>
<td>2025</td>
<td>2060</td>
<td>2070</td>
</tr>
<tr>
<td>E</td>
<td>600 BSC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E1</td>
<td>5.40</td>
<td>5.45</td>
<td>5.50</td>
</tr>
<tr>
<td>ε0</td>
<td>0.630</td>
<td>0.650</td>
<td>0.690</td>
</tr>
<tr>
<td>L</td>
<td>120</td>
<td>130</td>
<td>140</td>
</tr>
<tr>
<td>θ</td>
<td>0°</td>
<td>7°</td>
<td>15°</td>
</tr>
</tbody>
</table>
4.8 TSSOP20 package mechanical data

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-</td>
<td>-</td>
<td>1.20</td>
</tr>
<tr>
<td>A1</td>
<td>0.05</td>
<td>-</td>
<td>0.15</td>
</tr>
<tr>
<td>A2</td>
<td>0.90</td>
<td>1.00</td>
<td>1.05</td>
</tr>
<tr>
<td>A3</td>
<td>0.34</td>
<td>0.44</td>
<td>0.54</td>
</tr>
<tr>
<td>b</td>
<td>0.20</td>
<td>0.24</td>
<td>0.28</td>
</tr>
<tr>
<td>D</td>
<td>6.40</td>
<td>6.50</td>
<td>6.60</td>
</tr>
<tr>
<td>E</td>
<td>6.20</td>
<td>6.50</td>
<td>6.60</td>
</tr>
<tr>
<td>E1</td>
<td>4.30</td>
<td>4.40</td>
<td>4.50</td>
</tr>
<tr>
<td>e</td>
<td>0.65BSC</td>
<td>0.65BSC</td>
<td></td>
</tr>
<tr>
<td>L</td>
<td>0.45</td>
<td>0.60</td>
<td>0.75</td>
</tr>
<tr>
<td>L1</td>
<td>1.00REF</td>
<td>1.00REF</td>
<td></td>
</tr>
<tr>
<td>L2</td>
<td>0.25BSC</td>
<td>0.25BSC</td>
<td></td>
</tr>
<tr>
<td>R1</td>
<td>0.09</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>R2</td>
<td>0.09</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

units of measurement: mm
4.9 SOP16 package mechanical data

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1.35</td>
<td>1.60</td>
<td>1.75</td>
</tr>
<tr>
<td>A1</td>
<td>0.10</td>
<td>0.15</td>
<td>0.25</td>
</tr>
<tr>
<td>A2</td>
<td>1.25</td>
<td>1.45</td>
<td>1.65</td>
</tr>
<tr>
<td>A3</td>
<td>0.55</td>
<td>0.65</td>
<td>0.75</td>
</tr>
<tr>
<td>b</td>
<td>0.35</td>
<td>0.40</td>
<td>0.45</td>
</tr>
<tr>
<td>D</td>
<td>9.80</td>
<td>9.90</td>
<td>10.00</td>
</tr>
<tr>
<td>E</td>
<td>5.80</td>
<td>6.00</td>
<td>6.20</td>
</tr>
<tr>
<td>E1</td>
<td>3.80</td>
<td>3.90</td>
<td>4.00</td>
</tr>
<tr>
<td>e</td>
<td></td>
<td></td>
<td>1.27BSC</td>
</tr>
<tr>
<td>L</td>
<td>0.45</td>
<td>0.60</td>
<td>0.80</td>
</tr>
<tr>
<td>L1</td>
<td>0.04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L2</td>
<td></td>
<td>0.25BSC</td>
<td></td>
</tr>
<tr>
<td>R1</td>
<td>0.07</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R2</td>
<td>0.07</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

units of measurement: mm
## 5  STC8 series microcontroller selection price list

<table>
<thead>
<tr>
<th>Type</th>
<th>Working Voltage(V)</th>
<th>Flash program memory</th>
<th>I/O port maximum number</th>
<th>SPI</th>
<th>PCF</th>
<th>Time out (Time-out period + power off)</th>
<th>Internal low voltage detection interrupt and power down wake-up timer</th>
<th>Watch dog reset timer</th>
<th>16-bit enhanced PWM (PWM1 had one channel)</th>
<th>15-road high level PWM can be</th>
<th>15-road high level PWM can be</th>
<th>Mass extension SRAM (byte)</th>
<th>Online simulation</th>
<th>Encapsulation</th>
<th>Implementation Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>STC8F2K08S2</td>
<td>2.0-5.5</td>
<td>8K</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8F2K16S2</td>
<td>2.0-5.5</td>
<td>16K</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8F2K32S2</td>
<td>2.0-5.5</td>
<td>32K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8F2K60S2</td>
<td>2.0-5.5</td>
<td>60K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8F2K64S2</td>
<td>2.0-5.5</td>
<td>64K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8F2K16S4</td>
<td>2.0-5.5</td>
<td>16K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8F2K32S4</td>
<td>2.0-5.5</td>
<td>32K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8F2K60S4</td>
<td>2.0-5.5</td>
<td>60K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8F2K64S4</td>
<td>2.0-5.5</td>
<td>64K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8A4K16S2A12</td>
<td>2.0-5.5</td>
<td>16K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8A4K32S2A12</td>
<td>2.0-5.5</td>
<td>32K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8A4K60S2A12</td>
<td>2.0-5.5</td>
<td>60K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8A4K64S2A12</td>
<td>2.0-5.5</td>
<td>64K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8A8K16S4A12</td>
<td>2.0-5.5</td>
<td>16K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8A8K32S4A12</td>
<td>2.0-5.5</td>
<td>32K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8A8K64S4A12</td>
<td>2.0-5.5</td>
<td>64K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8H1K64S2A10</td>
<td>3.0-5.5</td>
<td>64K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8H1K128S2A10</td>
<td>3.0-5.5</td>
<td>128K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8H1K256S2A10</td>
<td>3.0-5.5</td>
<td>256K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8H1K512S2A10</td>
<td>3.0-5.5</td>
<td>512K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8H1K1024S2A10</td>
<td>3.0-5.5</td>
<td>1024K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8H1K2048S2A10</td>
<td>3.0-5.5</td>
<td>2048K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8H1K4096S2A10</td>
<td>3.0-5.5</td>
<td>4096K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8H1K8192S2A10</td>
<td>3.0-5.5</td>
<td>8192K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8H1K16384S2A10</td>
<td>3.0-5.5</td>
<td>16384K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8H1K32768S2A10</td>
<td>3.0-5.5</td>
<td>32768K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8H1K65536S2A10</td>
<td>3.0-5.5</td>
<td>65536K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8H1K131072S2A10</td>
<td>3.0-5.5</td>
<td>131072K</td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Nantong guoxin Microelectronics Co., Ltd.  Tel: 0513-5501 2928/2929/2966  Fax: 0513-5501 2956/2947  - 74 -
# 5.1 STC8 series of microcontroller package price list

<table>
<thead>
<tr>
<th>Type</th>
<th>LQFP64S</th>
<th>QFN64</th>
<th>QFN48</th>
<th>QFN48</th>
<th>LQFP44</th>
<th>LQFP32</th>
<th>QFN32</th>
<th>TSSOP20</th>
<th>SOP16</th>
<th>SOP8</th>
<th>PDIP40</th>
</tr>
</thead>
<tbody>
<tr>
<td>STC8F2K08S2</td>
<td>¥1.2</td>
<td>¥1.15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8F2K16S2</td>
<td>¥1.8</td>
<td>¥1.65</td>
<td>¥1.4</td>
<td>¥1.35</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8F2K32S2</td>
<td>1.99</td>
<td>¥1.8</td>
<td>¥1.85</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8F2K60S2</td>
<td>¥2.2</td>
<td>¥2.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8F2K64S2</td>
<td>¥2.2</td>
<td>¥2.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8F2K16S4</td>
<td>¥2.7</td>
<td>¥2.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8F2K32S4</td>
<td>¥2.8</td>
<td>¥2.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8F2K60S4</td>
<td>¥2.9</td>
<td>¥2.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8F2K64S4</td>
<td>¥2.9</td>
<td>¥2.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8A4K16S2A12</td>
<td>¥3.1</td>
<td>¥2.9</td>
<td>¥2.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8A4K32S2A12</td>
<td>¥3.3</td>
<td>¥3.0</td>
<td>¥3.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8A4K60S2A12</td>
<td>¥3.6</td>
<td>¥3.2</td>
<td>¥3.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8A4K64S2A12</td>
<td>¥3.6</td>
<td>¥3.2</td>
<td>¥3.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8A8K16S4A12</td>
<td>¥3.4</td>
<td>¥3.2</td>
<td>¥3.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8A8K32S4A12</td>
<td>¥3.6</td>
<td>¥3.3</td>
<td>¥3.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8A8K60S4A12</td>
<td>¥3.8</td>
<td>¥3.4</td>
<td>¥3.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8A8K64S4A12</td>
<td>¥3.8</td>
<td>¥3.4</td>
<td>¥3.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8H1K32S2</td>
<td>¥2.3</td>
<td>¥2.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8H1K32S2</td>
<td>¥1.9</td>
<td>¥1.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8H1K16S2</td>
<td>¥1.7</td>
<td>¥1.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8H1K08S2A10</td>
<td>¥1.1</td>
<td>¥1.05</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8H1K08S2</td>
<td>¥0.95</td>
<td>¥0.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8H04A10</td>
<td>¥0.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STC8H04</td>
<td>¥0.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5.2 STC8 series microcontroller name rule

- **STC**
- **8x**
- **xK**
- **64**
- **Sx**
- **Ax**

**ADC accuracy**
- A12: 12bits ADC
- A10: 10bits ADC

**Number of independent serial ports**
- S4: 4 Independent serial port
- S2: 2 Independent serial port
- S : 1 Independent serial port

**Program space size**
- 64: 64K bytes
- 32: 32K bytes
- 16: 16K bytes

**SRAM space size**
- 8K: 8K bytes
- 2K: 2K bytes

**Sub-series**
- **8F**: STC8F series (without AVcc, AGnd, AVRef pins)
- **8A**: STC8A series (with AVcc, AGnd, AVRef pins)
- **8H**: STC8H series
6 ISP download and typical application circuit diagram

6.1 STC8F series ISP download application circuit diagram

6.1.1 Using RS-232 transferor download

![Circuit Diagram](image)

**System Clock <= 10MHz**

System Clock <= 10MHz | System Clock >= 10MHz
--- | ---
104(0.1uF) | 103(0.01uF)

Diodes and resistors can be shorted directly.
6.1.2 Using PL2303-SA download

47μF Tantalum capacitors (footprint 3528) prices:<RMB ¥ 0.16
22μF Monolithic capacitors (footprint 0603) prices:<RMB ¥ 0.038
10μF Monolithic capacitors (footprint 0603) prices:<RMB ¥ 0.026
0.1μF Monolithic (footprint 0603) prices:<RMB ¥ 0.005

<table>
<thead>
<tr>
<th>System clock &lt;=10MHz</th>
<th>System clock&gt;10MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>C7 104(0.1uF)</td>
<td>103(0.01uF)</td>
</tr>
</tbody>
</table>

6.1.3 Using U8-Mini tool download

Connect PC
6.1.4 Using U8W tool download

The chip can be placed directly on this green locking seat for ISP programming, or it can be downloaded as a left connection.
6.1.5 USB directly ISP download

Notice: you need to connect p3.2 to GND to download it normally when you use it with USB.
6.2 STC8A series ISP download application circuit diagram

6.2.1 Using RS-232 transfer download (using high-precision ADC)

- System clock <= 10MHz
- System clock > 10MHz

- 47u Tantalum capacitors (footprint 3528) prices<RMB ¥ 0.16
- 22u Monolithic capacitors (footprint 0603) prices<RMB ¥ 0.038
- 10u Monolithic capacitors (footprint 0603) prices<RMB ¥ 0.028
- 0.1u Monolithic (footprint 0603) prices<RMB ¥ 0.005

Here diodes and resistors can be shorted directly.
6.2.2 Using RS-232 transfer download (ADC general application)

System Power (Power from USB)

- Connect PowerOn to Vin
- Connect Vcc to 5V

Components:
- 47u Tantalum capacitors (footprint 3528) prices<RMB ¥0.16
- 22u Monolithic capacitors (footprint 0603) prices<RMB ¥0.038
- 10u Monolithic capacitors (footprint 0603) prices<RMB ¥0.028
- 0.1u Monolithic (footprint 0603) prices<RMB ¥0.005

<table>
<thead>
<tr>
<th>Component</th>
<th>Values</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>C7</td>
<td>0.1u (0.1μF)</td>
<td>¥0.005</td>
</tr>
<tr>
<td>C1+</td>
<td>10u</td>
<td>¥0.028</td>
</tr>
<tr>
<td>C1-</td>
<td>10u</td>
<td>¥0.028</td>
</tr>
<tr>
<td>C2+</td>
<td>10u</td>
<td>¥0.028</td>
</tr>
<tr>
<td>C2-</td>
<td>10u</td>
<td>¥0.028</td>
</tr>
<tr>
<td>R1IN</td>
<td>300Ω</td>
<td>¥0.04</td>
</tr>
<tr>
<td>R1OUT</td>
<td>300Ω</td>
<td>¥0.04</td>
</tr>
<tr>
<td>R2IN</td>
<td>300Ω</td>
<td>¥0.04</td>
</tr>
<tr>
<td>R2OUT</td>
<td>300Ω</td>
<td>¥0.04</td>
</tr>
</tbody>
</table>

Here diodes and resistors can be shorted directly.
6.2.3 Using PL2303 download

System Power (Power from USB)

- 47u Tantalum capacitors (footprint 3528) prices<RMB ¥0.16
- 22u Monolithic capacitors (footprint 0603) prices<RMB ¥0.038
- 10u Monolithic capacitors (footprint 0603) prices<RMB ¥0.028
- 0.1u Monolithic (footprint 0603) prices<RMB ¥0.005

System clock <=10MHz

C?= 104(0.1uF)  103(0.01uF)

System clock >10MHz

C?= 104(0.1uF)  103(0.01uF)
6.2.4 Using U8-Mini tool download

If you just use the tool for ISP download, please refer to the reference line in 6.2.1.
6.2.5 Using U8W tool download

If you only use the tool for ISP download, AGnd, AVref, and AVcc pins can reference this connection. If you need to use a high-precision ADC, refer to 6.2.1 Reference Lines.

The chip can be placed directly on this green locking seat for ISP programming, or it can be downloaded as a left connection.
6.2.6 USB directly ISP download

Notice: you need to connect P3.2 to GND to download it normally when you use it with USB.

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

Hold down this button and connect USB to download ISP.

If you do not press this button while connected to USB, you will not enter the ISP but run the user code directly.

Note: The use of USB direct download cannot adjust the internal IRC frequency.
7 Clock, Reset and Power management

7.1 System clock control

The system clock controller provides a clock source for the CPU of the single chip microcomputer and all peripheral systems. The system clock has three clock sources to choose from: an internal high-precision IRC of 24 MHz, an internal IRC of 32 k Hz (large error), an external crystal oscillator, or an external clock signal. Each clock source can be individually enabled and turned off by a program, and clock division is internally provided to achieve the purpose of reducing power consumption.

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

System Clock Structure

Correlation register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>addr</th>
<th>Bit addr and symbol</th>
<th>Res value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CKSEL</td>
<td>Clock selection register</td>
<td>FE00H</td>
<td>MCLKODIV[3:0]</td>
<td>0000_0000</td>
</tr>
<tr>
<td>CLKDIV</td>
<td>Clock frequency division register</td>
<td>FE01H</td>
<td>MCLKO_S, MCKSEL[1:0]</td>
<td>0000_0100</td>
</tr>
<tr>
<td>IRC24MCR</td>
<td>Internal 24M oscillator control register</td>
<td>FE02H</td>
<td>ENIRC24M, IRC24MST</td>
<td>xxx_xxx0</td>
</tr>
<tr>
<td>XOSCCR</td>
<td>External oscillator control register</td>
<td>FE03H</td>
<td>XITYPE, XOSCST</td>
<td>xxx_xxx0</td>
</tr>
<tr>
<td>IRC32KCR</td>
<td>Internal 32K oscillator control register</td>
<td>FE04H</td>
<td>ENIRC32K, IRC32KST</td>
<td>xxx_xxx0</td>
</tr>
</tbody>
</table>

CKSEL(System clock select register)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CKSEL</td>
<td>FE00H</td>
<td>MCLKODIV[3:0]</td>
<td>MCLKO_S</td>
<td>MCKSEL[1:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MCLKODIV[3:0]: Frequency division coefficient of system clock output

(Note: the clock source for the system clock division output is the system clock after the main clock MCLK is divided by CLKDIV)
<table>
<thead>
<tr>
<th>Symbol</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>No output</td>
</tr>
<tr>
<td>0001</td>
<td>SYSClk/1</td>
</tr>
<tr>
<td>001x</td>
<td>SYSClk/2</td>
</tr>
<tr>
<td>010x</td>
<td>SYSClk/4</td>
</tr>
<tr>
<td>011x</td>
<td>SYSClk/8</td>
</tr>
<tr>
<td>100x</td>
<td>SYSClk/16</td>
</tr>
<tr>
<td>101x</td>
<td>SYSClk/32</td>
</tr>
<tr>
<td>110x</td>
<td>SYSClk/64</td>
</tr>
<tr>
<td>111x</td>
<td>SYSClk/128</td>
</tr>
</tbody>
</table>

- **MCLKO_S**: System clock output pin selection
  - 0: System clock frequency division output to P5.4
  - 1: System clock frequency division output to P1.6

- **MCKSEL[1:0]**: Main clock source selection

<table>
<thead>
<tr>
<th>MCKSEL[1:0]</th>
<th>Main clock source</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Internal 24MHz high precision IRC</td>
</tr>
<tr>
<td>01</td>
<td>External crystal oscillator or External input clock signal</td>
</tr>
<tr>
<td>10</td>
<td>Internal 32KHz low speed IRC</td>
</tr>
</tbody>
</table>

**CLKDIV(Clock divider register)**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKDIV</td>
<td>FE01H</td>
</tr>
</tbody>
</table>

- **CLKDIV**: Main clock division factor. The system clock SYSLK is a clock signal obtained by dividing the main clock MCLK.

<table>
<thead>
<tr>
<th>CLKDIV</th>
<th>System clock frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>MCLK/1</td>
</tr>
<tr>
<td>1</td>
<td>MCLK/1</td>
</tr>
<tr>
<td>2</td>
<td>MCLK/2</td>
</tr>
<tr>
<td>3</td>
<td>MCLK/3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>x</td>
<td>MCLK/x</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>255</td>
<td>MCLK/255</td>
</tr>
</tbody>
</table>

**IRC24MCR(Internal 24m high precision IRC control register)**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRC24MCR</td>
<td>FE02H</td>
</tr>
</tbody>
</table>

- **ENIRC24M**: Internal 24m high precision IRC enable bit
  - 0: close the internal 24m high precision IRC
  - 1: enable internal 24m high precision IRC
IRC 24 MST: internal 24m high precision IRC frequency stability flag bit. (read - only bit)
When the internal 24m IRC is enabled from the stop state, it must be some time before the oscillator frequency can be stabilized. When the oscillator frequency is stabilized, the clock controller will automatically set IRC 24 MST flag position 1. So when the user program needs to switch the clock to IRC using the internal 24m, must first set the en IRC 24 m = 1 enable oscillator, and then always query the oscillator stability flag IRC 24 MST, until the flag bit becomes 1, to switch the clock source.

**XOSCCR (External oscillator control register)**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>XOSCCR</td>
<td>FE03H</td>
<td>ENXOSC</td>
<td>XITYPE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>XOSCST</td>
</tr>
</tbody>
</table>

ENXOSC: External crystal oscillator enable bit
0: turn off the external crystal oscillator
1: enable external crystal oscillator
Xi type: external clock source type
0: the external clock source is an external clock signal (or active crystal oscillator). The signal source only needs to be connected to XTALI (p 1.7) of the single chip microcomputer
1: the external clock source is a crystal oscillator. The signal source is connected with XTALI (p 1.7) and XTALO (p 1.6) of the singlechip
XOS CST: external crystal oscillator frequency stability flag bit. (read - only bit)
When the external crystal oscillator is enabled from the stop state, it must take a period of time for the oscillator frequency to stabilize. When the oscillator frequency stabilizes, the clock controller will automatically set XOSCST flag position 1. So when the user program needs to switch the clock to use the external crystal oscillator, must first set ENSOSC = 1 enable oscillator, and then always query the oscillator stability flag XOSCST, until the flag bit becomes 1, to switch the clock source.

**IRC32KCR (Internal 32kHz low speed IRC control register)**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRC32KCR</td>
<td>FE04H</td>
<td>ENIRC32K</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>IRC32KST</td>
</tr>
</tbody>
</table>

En IRC 32k: internal 32k low speed IRC enable bit
0: turn off the internal 32k low speed IRC
1: enable internal 32k low speed IRC
Irc 32k ST: internal 32kHz low speed IRC frequency stability flag bit. (read - only bit)
When the internal 32kHz low-speed IRC is enabled from the stop state, it must take a period of time for the oscillator frequency to stabilize. When the oscillator frequency stabilizes, the clock controller will automatically mark the IRC 32kHz ST position 1. So when the user program needs to switch the clock to use the internal 32kHz low speed IRC, must first set the en IRC 32k = 1 enable oscillator, and then always query the oscillator stability flag IRC 32k ST, until the flag bit becomes 1, to switch the clock source.

### 7.2 System reset

The reset of STC8 series singlechip is divided into two kinds: hardware reset and software reset.

- When the hardware is reset, the values of all registers are reset to the initial values, and all hardware options are re-read. At the same time according to the hardware options set by the power-on wait time
to power up wait. Hardware reset mainly includes:

- Power-on reset
- Low pressure reset
- Reduction of foot
- Watchdog reset

When the software is reset, the value of all the registers will be reset to the initial value except for the register associated with the clock, and the software reset will not re-read all the hardware options. Software reset mainly includes:

The reset triggered by writing IAP CONTR’s SWRST

### Correlation register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>description</th>
<th>addr ess</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
<th>reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDT_CONTR</td>
<td>Watchdog control register</td>
<td>C1H</td>
<td>WDT_FLAG</td>
<td>EN_WDT</td>
<td>CLR_WDT</td>
<td>IDL_WDT</td>
<td>WDT_PS[2:0]</td>
<td>0x00,0000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IAP_CONTR</td>
<td>IAP control register</td>
<td>C7H</td>
<td>IAPEN</td>
<td>SWBS</td>
<td>SWRST</td>
<td>CMD_FAIL</td>
<td>-</td>
<td>IAP_WT[2:0]</td>
<td>0000,x000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSTCFG</td>
<td>Reset configuration register</td>
<td>FFH</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>P54RST</td>
<td>-</td>
<td>-</td>
<td>LVDS[1:0]</td>
<td>0000,0000</td>
<td></td>
</tr>
</tbody>
</table>

### WDT_CONTR(Watchdog control register)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDT_CONTR</td>
<td>C1H</td>
<td>WDT_FLAG</td>
<td>-</td>
<td>EN_WDT</td>
<td>CLR_WDT</td>
<td>IDL_WDT</td>
<td>WDT_PS[2:0]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

WDTFLAG: watchdog overflow sign
Watchdog overflow occurs, the hardware automatically will this position 1, the need for software clearance.
ENWD: watchdog enable
0: no effect on single Chip Microcomputer
1: start watchdog timer
CLRWDTT: watchdog timer zero
0: no effect on single Chip Microcomputer
1: Clear watchdog timer and reset this bit automatically
Watchdog control bit in IDL_WDT:IDLE mode
Watchdog stop counting in 0:IDLE mode
Watchdog continues counting when in 1:IDLE mode
WDT_PS[2:0]: watchdog timer clock frequency division coefficient WDTFLAG: watchdog overflow sign
Watchdog overflow occurs, the hardware automatically will this position 1, the need for software clearance.
Enwd: watchdog enable
0: no effect on single Chip Microcomputer
1: start watchdog timer
CLRWDTT: watchdog timer zero
0: no effect on single Chip Microcomputer
1: Clear watchdog timer and reset this bit automatically
Watchdog control bit in IDL_WDT:IDLE mode
Watchdog stop counting in 0:IDLE mode
Watchdog continues counting when in 1:IDLE mode

**WDT_PS [2:0]: watchdog timer clock frequency division coefficient**

<table>
<thead>
<tr>
<th>WDT_PS[2:0]</th>
<th>division factor</th>
<th>The overflow time of 12M in the main frequency</th>
<th>The overflow time of 20M in the main frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>2</td>
<td>≈ 65.5 MS</td>
<td>≈ 39.3 MS</td>
</tr>
<tr>
<td>001</td>
<td>4</td>
<td>≈ 131 MS</td>
<td>≈ 78.6 MS</td>
</tr>
<tr>
<td>010</td>
<td>8</td>
<td>≈ 262 MS</td>
<td>≈ 157 MS</td>
</tr>
<tr>
<td>011</td>
<td>16</td>
<td>≈ 524 MS</td>
<td>≈ 315 MS</td>
</tr>
<tr>
<td>100</td>
<td>32</td>
<td>≈ 1.05 S</td>
<td>≈ 629 MS</td>
</tr>
<tr>
<td>101</td>
<td>64</td>
<td>≈ 2.10 S</td>
<td>≈ 1.26 S</td>
</tr>
<tr>
<td>110</td>
<td>128</td>
<td>≈ 4.20 S</td>
<td>≈ 2.52 S</td>
</tr>
<tr>
<td>111</td>
<td>256</td>
<td>≈ 8.39 S</td>
<td>≈ 5.03 S</td>
</tr>
</tbody>
</table>

The formula for calculating the overflow time of the watchdog is as follows:

\[
\text{Overflow time of watchdog timer} = \frac{12 \times 32768 \times 2^{(WDT\_PS+1)}}{\text{SYSclk}}
\]

**IAP_CONTR(IAP control register)**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAP_CONTR</td>
<td>C7H</td>
<td>IAPEN</td>
<td>SWBS</td>
<td>SWRST</td>
<td>CMD_FAIL</td>
<td>-</td>
<td></td>
<td></td>
<td>IAP_CMD[2:0]</td>
</tr>
</tbody>
</table>

SWBS: software reset startup selection
0: after the software is reset, execute the code from the user program area. The data in the user data area remains the same.
1: after the software reset, the code is executed from the system ISP area. The data in the user data area is initialized.

SWRST: the program for calculating the spillover time of the watchdog is as follows:
The formula for calculating the spillover time of the watchdog is as follows:
The formula for calculating the spillover time of the watchdog is as follows:
0: No effect on single Chip Microcomputer
1: Trigger software reset

**RSTCFG(Reset configuration register)**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSTCFG</td>
<td>FFH</td>
<td>-</td>
<td>ENLVR</td>
<td>-</td>
<td>P54RST</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>LVDS[1:0]</td>
</tr>
</tbody>
</table>

ENLVR: Low voltage reset control bit
0: low-voltage reset is prohibited. Low voltage interruptions occur when the system detects a low voltage event
1: enable low pressure reset. Automatic reset when a low voltage event is detected by the system

RST: Pin function selection
SWBS: software reset startup selection
0: after the software is reset, execute the code from the user program area. The data in the user data area remains the same.

1: after the software reset, the code is executed from the system ISP area. The data in the user data area is initialized.

- **0**: RST Pin used as ordinary I/O port (P54)
- **1**: RST Pin used as reset pin

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

<table>
<thead>
<tr>
<th>LVDS[1:0]</th>
<th>Threshold voltage of low voltage detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>2.2V</td>
</tr>
<tr>
<td>01</td>
<td>2.4V</td>
</tr>
<tr>
<td>10</td>
<td>2.7V</td>
</tr>
<tr>
<td>11</td>
<td>3.0V</td>
</tr>
</tbody>
</table>

### 7.3 System power management

<table>
<thead>
<tr>
<th>Symbol</th>
<th>address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
<th>reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCON</td>
<td>87H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0011_0000</td>
</tr>
<tr>
<td>VOCTRL</td>
<td>BBH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0_0xx_xx00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Symbol</th>
<th>address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCON</td>
<td>87H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMOD</td>
<td>SMOD0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LVDF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GF1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GF0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IDL</td>
<td>0011_0000</td>
<td>0_0xx_xx00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**PCON (power control register)**

- **LVDF**: Low-voltage detection mark. When the system detects the low voltage event, the hardware automatically takes this position 1 and requests the CPU interrupt. This bit requires user software clearance.
- **POF**: Power on mark bit. When the hardware automatically places this position 1.
- **PD**: Power-down mode control bit
  - **0**: No effect
  - **1**: Single-chip microcomputer into power-off mode CPU and all peripherals are stopped working. The hardware is automatically cleared after wake-up.

(External pins that wake CPU from power off mode include: INT0/P3.2, INT1/P3.3, INT2/P3.6, INT3/P3.7, INT4/P3.0, T0/P3.4, T1/P3.5, T2/P1.2, T3/P0.4, T4/P0.6, CCP0/P1.7, CCP1/P1.6, CCP2/P1.5, CCP4/P1.4, CCP0_2/P2.3, CCP1_2/P2.4, CCP2_2/P2.5, CCP3_2/P2.6, CCP0_3/P7.0, CCP1_3/P7.1, CCP2_3/P7.2, CCP3_3/P7.3, CCP0_4/P3.3, CCP1_4/P3.2, CCP2_4/P3.1, CCP3_4/P3.0, RxD/P3.0, RxD_2/P3.6, RxD_3/P1.6, RxD_4/P4.3, RxD2/P1.0, RxD2_2/P4.0, RxD3/P0.0, RxD3_2/P5.0, RxD4/P0.2, RxD4_2/P5.2, There are also power-down wake-up timers, low-voltage interrupts, and comparator interrupts that also wake the CPU from power-off mode.)

**IDL**: IDLE (idle) Mode control bit
0: No effect
1: Microcontroller enters IDLE mode, other peripherals are still running, when CPU stops working. Automatic hardware clearance after wake-up

**VOCTRL (Voltage control register)**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>VOCTRL</td>
<td>BBH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SCC: Static current control bit

0: The static current control circuit is selected for static current control, and the static current is about 1.5uA.

1: Select the external static holding current control circuit, select this mode when the lower power consumption. The static current under the mode of STC8A8K series is generally less than 0.15uA; the static current of STC8F2K series is generally below 0.1uA. Note: this mode in power down mode, the VCC pin voltage can have greater volatility, or on MCU the kernel may have adverse effects.

[B1:B0]: Internal test bit, it must be written to 0

### 7.4 Sample program

#### 7.4.1 Select system clock source

**Assembly code**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_S_SW2 DATA 0BAH</td>
<td></td>
</tr>
<tr>
<td>CKSEL EQU 0FE00H</td>
<td></td>
</tr>
<tr>
<td>CLKDIV EQU 0FE01H</td>
<td></td>
</tr>
<tr>
<td>IRC24MCR EQU 0FE02H</td>
<td></td>
</tr>
<tr>
<td>XOSCCR EQU 0FE03H</td>
<td></td>
</tr>
<tr>
<td>IRC32KCR EQU 0FE04H</td>
<td></td>
</tr>
<tr>
<td>ORG 0000H</td>
<td></td>
</tr>
<tr>
<td>LJMP MAIN</td>
<td></td>
</tr>
<tr>
<td>ORG 0100H</td>
<td></td>
</tr>
</tbody>
</table>

**MAIN:**

```
MOV SP,#3FH
MOV P_S\_SW2,#80H ; Select the internal IRC (default)
MOV A,#00H
MOV DPTR,#CKSEL
MOVX @DPTR,A
MOVX A,@DPTR ; Waiting for the stability of the clock
CLR A ; Clock without frequency division
MOV DPTR,#CLKDIV
MOVX @DPTR,A
MOV P_S\_SW2,#00H ; Start the external oscillator
MOV A,#0C0H
MOV DPTR,#XOSCCR
MOVX @DPTR,A
MOVX A,@DPTR
JNB ACC.0,$-1 ; Clock without frequency division
```
; MOV A,#01H ; Selection of external oscillator
; MOV DPTR,#CKSEL
; MOVX @DPTR,A
; MOV P_SW2,#00H

; MOV P_SW2,#80H ; Start the internal 32K IRC
; MOV DPTR,#IRC32KCR
; MOVX @DPTR,A
; MOVX A,@DPTR
; JNB ACC.0,$-1 ; Waiting for the stability of the clock
; CLR A ; Clock without frequency division
; MOV DPTR,#CLKDIV
; MOVX @DPTR,A
; MOV A,#03H ; Select the internal 32K
; MOV DPTR,#CKSEL
; MOVX @DPTR,A
; MOV P_SW2,#00H

JMP $

END

C code
#include "reg51.h"
#include "intrins.h"
#define CKSEL (*(unsigned char volatile xdata *)0xfe00)
#define CLKDIV (*(unsigned char volatile xdata *)0xfe01)
#define IRC24MCR (*(unsigned char volatile xdata *)0xfe02)
#define XOSCCR (*(unsigned char volatile xdata *)0xfe03)
#define IRC32KCR (*(unsigned char volatile xdata *)0xfe04)
sfr P_SW2 = 0xba;

void main()
{
P_SW2 = 0x80;
CKSEL = 0x00; // Select the internal IRC (default)
P_SW2 = 0x00;

/*
P_SW2 = 0x80;
XOSCCR = 0xc0; // Start the external oscillator
while ((XOSCCR & 1)); // Waiting for the stability of the clock
CLKDIV = 0x00; // Clock without frequency division
CKSEL = 0x03; // Selection of external oscillator
P_SW2 = 0x00;
*/

/*
P_SW2 = 0x80;
IRC32KCR = 0x80; // Start the internal 32K IRC
while ((IRC32KCR & 1)); // Waiting for the stability of the clock
CLKDIV = 0x00; // Clock without frequency division
*/
CKSEL = 0x03; // Select the internal 32K
P_SW2 = 0x00;
*/
while (1);
;

7.4.2 Master clock frequency division output

Assembly code

```
P_SW2 DATA 0BAH
CKSEL EQU 0FE00H
CLKDIV EQU 0FE01H

ORG 0000H
LJMP MAIN

ORG 0100H
MAIN:

MOV SP,#3FH
MOV P_SW2,#80H ; MOV A,#10H ; The master clock output to the P5.4 port
; MOV A,#20H                   ; The 2 frequency division output of the master clock to the P5.4 port
MOV A,#40H ; The 4 frequency division output of the master clock to the P5.4 port
MOV A,#48H ; The 4 frequency division output of the master clock to the P1.6 port
MOV DPTR,#CKSEL
MOVX @DPTR,A
MOV P_SW2,#00H
JMP $

END
```

C code

```c
#include "reg51.h"
#include "intrins.h"
#define CKSEL (*(unsigned char volatile xdata *)0xfe00)
#define CLKDIV (*(unsigned char volatile xdata *)0xfe01)
sfr P_SW2 = 0xba;

void main()
{
    P_SW2 = 0x80;
    // CKSEL = 0x10; // The master clock output to the P5.4 port
    // CKSEL = 0x20; // The 2 frequency division output of the master clock to the P5.4 port
    CKSEL = 0x40; // The 4 frequency division output of the master clock to the P5.4 port
    // CKSEL = 0x48; // The 4 frequency division output of the master clock to the P1.6 port
    P_SW2 = 0x00;
    while (1);
}
```
7.4.3 Watchdog timer application

Assembly code

; The test frequency is 11.0592MHz

WDT_CONTR DATA 0C1H

ORG 0000H
LJMP MAIN

ORG 0100H
MAIN:

MOV SP,#3FH

; MOV WDT_CONTR,#23H ; Make a watchdog, the overflow time is about 0.5s
MOV WDT_CONTR,#24H ; Make a watchdog, the overflow time is about 1s
; MOV WDT_CONTR,#27H ; Make a watchdog, the overflow time is about 8s
CLR P3.2 ; Test port

LOOP:

; ORL WDT_CONTR,#10H ; Clear watchdog, otherwise the system reset
JMP LOOP

END

C code

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

// The test frequency is 11.0592MHz

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

void main()
{
    // WDT_CONTR = 0x23; // Make a watchdog, the overflow time is about 0.5s
    WDT_CONTR = 0x24; // Make a watchdog, the overflow time is about 1s
    // WDT_CONTR = 0x27; // Make a watchdog, the overflow time is about 8s
    P32 = 0; // Test port

    while (1)
    {
        // WDT_CONTR |= 0x10; // Clear watchdog, otherwise the system reset
    }
}

7.4.4 Soft reset to implement custom Downloads

Assembly code

; The test frequency is 11.0592MHz

IAP_CONTR DATA 0C7H
ORG 0000H
LJMP MAIN

ORG 0100H

MAIN:
MOV SP,#3FH
SETB P3.2
SETB P3.3

LOOP:
JB P3.2,LOOP
JB P3.3,LOOP
MOV IAP_CONTR,#60H ; Check P3.2 and P3.3 at the same time for 0 to reset ISP
JMP $

END

C code
#include "reg51.h"
#include "intrins.h"

// The test frequency is 11.0592MHz

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

void main()
{
    P32 = 1; // Test port
    P33 = 1; // Test port

    while (1)
    {
        if (!P32 & !P33)
        {
            IAP_CONTR |= 0x60; // Check P3.2 and P3.3 at the same time for 0 to reset ISP
        }
    }
}

7.4.5 Low voltage detection

assembly code

RSTCFG DATA 0FFH
ENLVR EQU 40H ;RSTCFG6
LVD2V2 EQU 00H ;LVD@2.2V
LVD2V4 EQU 01H ;LVD@2.4V
LVD2V7 EQU 02H ;LVD@2.7V
LVD3V0 EQU 03H ;LVD@3.0V
ELVD BIT IE.6
LVDF EQU 20H ;PCON.5

ORG 0000H
LJMP MAIN
ORG 0033H
LJMP LVDISR

ORG 0100H
LVDISR:
  ANL PCON,#NOT LVDF ; Clear interruption sign
  CPL P3.2 ; Test port
  RETI

MAIN:
  MOV SP,#3FH
  ANL PCON,#NOT LVDF ; Need to clear the LVDF logo after power-on
  MOV RSTCFG,#ENLVR | LVD3V0 ; Enables low voltage reset at 3.0V without LVD interrupt
  MOV RSTCFG,#LVD3V0 ; Low voltage interrupt when 3.0V is enabled
  SETB ELVD ; Enable LVD interrupt
  SETB EA
  JMP $

END

C code
#include "reg51.h"
#include "intrins.h"
sfr RSTCFG = 0xff;
#define ENLVR  0x40 //RSTCFG .6
#define LVD2V2 0x00 //LVD@2.2V
#define LVD2V4 0x01 //LVD@2.4V
#define LVD2V7 0x02 //LVD@2.7V
#define LVD3V0 0x03 //LVD@3.0V
sbit ELVD = IE^6;
#define LVDF  0x20 //PCON.5
sbit P32 = P3^2;

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

void main()
{
  PCON &= ~LVDF; // Test port
  // RSTCFG = ENLVR | LVD3V0; // Enables low voltage reset at 3.0V without LVD interrupt
  RSTCFG = LVD3V0; // Low voltage interrupt when 3.0V is enabled
  ELVD = 1; // Enable LVD interrupt
  EA = 1;

  while (1);
}

7.4.6 Power saving mode

assembly code
STC8F Series Manual

Include "reg51.h"
Include "intrins.h"

sfr VOCTRL = 0xbb;
define IDL 0x01 //PCON.0
define PD 0x02 //PCON.1

sbit P34 = P3^4;
sbit P35 = P3^5;

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

void main()
{
  VOCTRL = 0x00;
  //When the power down mode is used, the internal SCC module is used, and the power consumption is about 1.5uA
  VOCTRL = 0x80;
  // When the power down mode is used, the external SCC module is used, and the power consumption is about 0.15uA
}

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

while (1);
}

### 7.4.7 Wake-up MCU with INT0/INT1/INT2/INT3/INT4 interrupt

#### assembly code

<table>
<thead>
<tr>
<th>INTCLOKO</th>
<th>DATA</th>
<th>8FH</th>
</tr>
</thead>
<tbody>
<tr>
<td>EX2</td>
<td>EQU</td>
<td>10H</td>
</tr>
<tr>
<td>EX3</td>
<td>EQU</td>
<td>20H</td>
</tr>
<tr>
<td>EX4</td>
<td>EQU</td>
<td>40H</td>
</tr>
</tbody>
</table>

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:

```assembly
MOV SP,#3FH
CLR IT0 ; Enable INT0 rising edge and falling edge interruption
SETB IT0 ; Enable INT0 falling edge interruptio
SETB EX0 ; Enable INT0 interruption
CLR IT1 ; Enable INT1 rising edge and falling edge interruption
SETB IT1 ; Enable INT1 falling edge interruptio
SETB EX1 ; Enable INT1 interruption
MOV INTCLKO,#EX2 ; Enable INT2 falling edge interruptio
ORL INTCLKO,#EX3 ; Enable INT3 falling edge interruptio
ORL INTCLKO,#EX4 ; Enable INT4 falling edge interruptio
SETB EA
MOV PCON,#02H ; MCU enters the power down mode
NOP ; Enter interruption service program immediately after power down wake-up
```

```assembly
LOOP:
CPL P1.1
JMP LOOP
END
```

C code

```c
#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;

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
}
```
7.4.8 Wake-up MCU with T0/T1/T2/T3/T4 interrupts

assembly code

; The test frequency is 11.0592MHz

T2L    DATA    0D7H
T2H    DATA    0D6H
T3L    DATA    0D5H
T3H    DATA    0D4H
T4L    DATA    0D3H
T4H    DATA    0D2H
T4T3M  DATA    0D1H
AUXR   DATA    8EH
IE2    DATA    0AFH
ET2    EQU     04H
ET3    EQU     20H
ET4    EQU     40H
AUXINTIF DATA    0EFH
T2IF   EQU     01H
T3IF   EQU     02H
T4IF   EQU     04H
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
ANL AUXINTIF,#NOT T2IF ; Clear interruption sign
RETI

TM3ISR:
CPL P1.0 ; Test port
ANL AUXINTIF,#NOT T3IF ; Clear interruption sign
RETI

TM4ISR:
CPL P1.0 ; Test port
ANL AUXINTIF,#NOT T4IF ; Clear interruption sign
RETI

MAIN:
MOV SP,#3FH

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

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

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

MOV T3L,#66H ;65536-11.0592M/12/1000
MOV T3H,#0FCH
MOV T4T3M,#08H ; Start timer
ORL IE2,#ET3 ; Enable timer interruption
MOV T4L,#66H ; 65536-11.0592M/12/1000
MOV T4H,#0FCH
ORL T4T3M,#80H ; Start timer
ORL IE2,#ET4 ; Enable timer interruption
SETB EA
MOV PCON,#02H ; MCU enters the power down mode
N O P ; After the power down, it will not enter the interrupt service program immediately,
; It will wait for the timer overflow to enter the interrupt service program
N O P
LOOP:
CPL P1.1
JMP LOOP
END
C code
#include "reg51.h"
#include "intrins.h"

// The test frequency is 11.0592MHz:
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;

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
    AUXINTIF &= ~T2IF; // Clear interruption sign
}

void TM3_Isr() interrupt 19
{
    P10 = !P10; // Test port
    AUXINTIF &= ~T3IF; // Clear interruption sign
}

void TM4_Isr() interrupt 20
{
    P10 = !P10; // Test port
    AUXINTIF &= ~T4IF; // Clear interruption sign
}

void main()
{
    TMOD = 0x00;
    TL0 = 0x66; // 65536 - 11.0592MHz / 12 / 1000
    TH0 = 0xfc;
    TR0 = 1; // Start timer
    ET0 = 1; // Enable timer interruption
    TL1 = 0x66; // 65536 - 11.0592MHz / 12 / 1000
    TH1 = 0xfc;
    TR1 = 1; // Start timer
    ET1 = 1; // Enable timer interruption
    T2L = 0x66; // 65536 - 11.0592MHz / 12 / 1000
    T2H = 0xfc;
    AUXR = 0x10; // Start timer
    IE2 = ET2; // Enable timer interruption
    T3L = 0x66; // 65536 - 11.0592MHz / 12 / 1000
    T3H = 0xfc;
    T4T3M = 0x08; // Start timer
    IE2 |= ET3; // Enable timer interruption
    T4L = 0x66; // 65536 - 11.0592MHz / 12 / 1000
    T4H = 0xfc;
    T4T3M |= 0x80; // Start timer
    IE2 |= ET4; // Enable timer interruption
    EA = 1;
    PCON = 0x02; // MCU enters the power down mode
    _nop_();
    // After the power fail, it will not enter the interrupt service program immediately,
    // It will wait for the timer overflow to enter the interrupt service program
    _nop_();
    while (1)
7.4.9 Wake-up MCU with RxD/RxD2/RxD3/RxD4 interrupt

assembly code

; The test frequency is 11.0592MHz

IE2 DATA 0AFH
ES2 EQU 01H
ES3 EQU 08H
ES4 EQU 10H
P_SW1 DATA 0A2H
P_SW2 DATA 0BAH

ORG 0000H
LJMP MAIN

ORG 0023H
LJMP UART1ISR

ORG 0043H
LJMP UART2ISR

ORG 008BH
LJMP UART3ISR

ORG 0093H
LJMP UART4ISR

UART1ISR:
    RETI

UART2ISR:
    RETI

UART3ISR:
    RETI

UART4ISR:
    RETI

MAIN:
    MOV SP,#3FH
    MOV P_SW1,#00H           ; The falling edge of the RXD/P3.0 can wake up the program
    MOV P_SW1,#40H           ; The falling edge of the RXD_2/P3.6 can wake up the program
    MOV P_SW1,#80H           ; The falling edge of the RXD_3/P1.6 can wake up the program
    MOV P_SW1,#0C0H          ; The falling edge of the RXD_4/P4.3 can wake up the program
    MOV P_SW2,#00H          ; The falling edge of the RXD2/P1.0 can wake up the program
    MOV P_SW2,#01H          ; The falling edge of the RXD2_2/P4.0 can wake up the program
    MOV P_SW2,#00H          ; The falling edge of the RXD3/P0.0 can wake up the program
; MOV P_SW2,#02H ; The falling edge of the RXD3_2/P5.0 can wake up the program
; MOV P_SW2,#00H ; The falling edge of the RXD4/P0.2 can wake up the program
; MOV P_SW2,#04H ; The falling edge of the RXD4_2/P5.2 can wake up the program
SETB ES ; Enable the interruption of the serial port
MOV IE2,#ES2 ; Enable the interruption of the serial port
ORL IE2,#ES3 ; Enable the interruption of the serial port
ORL IE2,#ES4 ; Enable the interruption of the serial port
SETB EA
MOV PCON,#02H ; MCU enters the power down mode
NOP ; After the power down, it will not enter the
; interrupt service program
NOP
LOOP:
CPL P1.1
JMP LOOP
END

C code
#include "reg51.h"
#include "intrins.h"

// The test frequency is 11.0592MHz
sfr IE2 = 0xfaf;
#define ES2 0x101
#define ES3 0x108
#define ES4 0x110
sfr P_SW1 = 0xa2;
sfr P_SW2 = 0xba;
sbit P11 = P1^1;

void UART1_Isr() interrupt 4
{
}
void UART2_Isr() interrupt 8
{
}
void UART3_Isr() interrupt 17
{
}
void UART4_Isr() interrupt 18
{
}
void main()
{
P_SW1 = 0x00; // The falling edge of the RXD/P3.0 can wake up the program

}
// P_SW1 = 0x40;   // The falling edge of the RXD_2/P3.6 can wake up the program
// P_SW1 = 0x80;   // The falling edge of the RXD_3/P1.6 can wake up the program
// P_SW1 = 0xc0;   // The falling edge of the RXD_4/P4.3 can wake up the program

P_SW2 = 0x00;     // The falling edge of the RXD2/P1.0 can wake up the program
// P_SW2 = 0x01;   // The falling edge of the RXD2_2/P4.0 can wake up the program
P_SW2 = 0x00;     // The falling edge of the RXD3/P0.0 can wake up the program
// P_SW2 = 0x02;   // The falling edge of the RXD3_2/P5.0 can wake up the program
P_SW2 = 0x00;     // The falling edge of the RXD4/P0.2 can wake up the program
// P_SW2 = 0x04;   // The falling edge of the RXD4_2/P5.2 can wake up the program

ES = 1;          // Enable the interruption of the serial port
IE2 = ES2;       // Enable the interruption of the serial port
IE2 |= ES3;      // Enable the interruption of the serial port
IE2 |= ES4;      // Enable the interruption of the serial port
EA = 1;

PCON = 0x02;     // MCU enters the power down mode
_nop_();         // After the power fail, it will not enter the interrupt service program
_nop_();

while (1)
{
    P11 = ~P11;
}

7.4.10  Wake-up MCU with LVD interrupt

Assembly code

<table>
<thead>
<tr>
<th>RSTCFG</th>
<th>DATA</th>
<th>0FFH</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENLVR</td>
<td>EQU</td>
<td>40H</td>
</tr>
<tr>
<td>LVD2V2</td>
<td>EQU</td>
<td>00H</td>
</tr>
<tr>
<td>LVD2V4</td>
<td>EQU</td>
<td>01H</td>
</tr>
<tr>
<td>LVD2V7</td>
<td>EQU</td>
<td>02H</td>
</tr>
<tr>
<td>LVD3V0</td>
<td>EQU</td>
<td>03H</td>
</tr>
<tr>
<td>ELVD</td>
<td>BIT</td>
<td>IE.6</td>
</tr>
<tr>
<td>LVDF</td>
<td>EQU</td>
<td>20H</td>
</tr>
</tbody>
</table>

ORG 0000H
LJMP MAIN
ORG 0033H
LJMP LVDISR

ORG 0100H

LVDISR:

    ANL PCON,#NOT LVDF ; Clear interruption sign
    CPL P1.0 ; Test port

RETI

MAIN:

MOV SP,#3FH

Nantong guoxin Microelectronics Co., Ltd.  Tel: 0513-5501 2928/2929/2966  Fax: 0513-5501 2926/2956/2947 - 108 -
ANL PCON,#NOT LVDF ; Power on needs clear interruption signs
MOV RSTCFG,#LVD3V0 ; Set the LVD voltage to 3.0V
SETB ELVD ; Enable the interruption of LVD

MOV PCON,#02H ; MCU enters the power down mode
NOP ; Enter interruption service program
NOP
LOOP:
  CPL P1.1
  JMP LOOP

END

C code
#include "reg51.h"
#include "intrins.h"
sfr RSTCFG = 0xff;
#define ENLVR 0x40 // RSTCFG.6
#define LVD2V2 0x00 // LVD@2.2V
#define LVD2V4 0x01 // LVD@2.4V
#define LVD2V7 0x02 // LVD@2.7V
#define LVD3V0 0x03 // LVD@3.0V
#define LVDF 0x20 // PCON.5
sbit ELVD = IE^6;
sbit P10 = P1^0;
sbit P11 = P1^1;

void LVD_Isr() interrupt 6 {
  PCON &= ~LVDF; // Clear interruption sign
  P10 = !P10; // Test port
}

void main() {
  PCON &= ~LVDF; // Power on needs clear interruption signs
  RSTCFG = LVD3V0; // Set the LVD voltage to 3.0V
  ELVD = 1; // Enable the interruption of LVD
  EA = 1;

  PCON = 0x02; // MCU enters the power down mode
  _nop_(); // Enter interruption service program immediately after power down wake-up
  _nop_();

  while (1) {
    P11 = ~P11;
  }
}
7.4.11 Wake-up MCU with CCP0/CCP1/CCP2/CCP3 interrupt

assembly code

; The test frequency is 11.0592MHz

ORG 0000H
LJMP MAIN

ORG 003BH
LJMP PCAISR

ORG 0100H

PCAISR:
  ANL CCON,#NOT 8FH ; Clear interruption sign
  CPL P1.0 ; Test port
  RETI

MAIN:
  MOV SP,#3FH
  MOV P_SW1,#00H ; CCP0/P1.7, CCP1/P1.6, CCP2/P1.5, CCP3/P1.4
  MOV P_SW1,#10H ; CCP0_2/P2.3, CCP1_2/P2.4, CCP2_2/P2.5 CCP3_2/P2.6
  MOV P_SW1,#20H ; CCP0_3/P7.0, CCP1_3/P7.1, CCP2_3/P7.2, CCP3_3/P7.3
  MOV P_SW1,#30H ; CCP0_4/P3.3, CCP1_4/P3.2, CCP2_4/P3.1, CCP3_4/P3.0
  MOV CCON,#00H ; The PCA clock is a system clock
  MOV CMOD,#08H ; Enable the edge of wake-up function of the CCP0 port
MOV CCPM1,#31H ; Enable the edge of wake-up function of the CCP1 port
MOV CCPM2,#31H ; Enable the edge of wake-up function of the CCP2 port
MOV CCPM3,#31H ; Enable the edge of wake-up function of the CCP3 port
SETB CR ; Start the PCA timer
SETB EA
MOV PCON,#02H ; MCU enters the power down mode
NOP ; Enter interruption service program
; immediately after power down wake-up
NOP

LOOP:
CPL P1.1
JMP LOOP

END

C code
#include "reg51.h"
#include "intrins.h"

// The test frequency is 11.0592MHz

sfr CCON = 0xd8;
sbit CF = CCON^7;
sbit CR = CCON^6;
sbit CCF3 = CCON^3;
sbit CCF2 = CCON^2;
sbit CCF1 = CCON^1;
sbit CCF0 = CCON^0;
sfr CMOD = 0xd9;
sfr CL = 0xe9;
sfr CH = 0xf9;
sfr CCPM0 = 0xda;
sfr CCP0L = 0xea;
sfr CCP0H = 0xfa;
sfr PCA_PWM0 = 0xf2;
sfr CCPM1 = 0xdb;
sfr CCP1L = 0xeb;
sfr CCP1H = 0xfc;
sfr PCA_PWM1 = 0xf3;
sfr CCPM2 = 0xdc;
sfr CCP2L = 0xed;
sfr CCP2H = 0xfd;
sfr PCA_PWM2 = 0xf5;
sfr CCPM3 = 0xdd;
sfr CCP3L = 0xed;
sfr CCP3H = 0xfd;
sfr PCA_PWM3 = 0xf5;
sfr P_SW1 = 0xa2;
sbit P10 = P1^0;
sbit P11 = P1^1;
void PCA_Isr() interrupt 7
{
    CCON &= ~0x8f; // Clear interruption sign
    P10 = !P10; // Test port
}

void main()
{
    _nop_();

    P_SW1 = 0x00; // CCP0/P1.7, CCP1/P1.6, CCP2/P1.5, CCP3/P1.4
    // P_SW1 = 0x10; // CCP0_2/P2.3, CCP1_2/P2.4, CCP2_2/P2.5, CCP3_2/P2.6
    // P_SW1 = 0x20; // CCP0_3/P7.0, CCP1_3/P7.1, CCP2_3/P7.2, CCP3_3/P7.3
    // P_SW1 = 0x30; // CCP0_4/P3.3, CCP1_4/P3.2, CCP2_4/P3.1, CCP3_4/P3.0

    /*
     * The following sets the PCA clock as a system clock.
     * Enable the edge of wake-up function of the CCP ports.
     * Start the PCA timer.
     * Enter interruption service program immediately after power down wake-up.
     */
    _nop_();
    while (1)
    {
        P11 = ~P11;
    }
}

7.4.12 CMP interrupt wake-up MCU

**assembly code**

<table>
<thead>
<tr>
<th>CMPCR1</th>
<th>DATA</th>
<th>0E6H</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPCR2</td>
<td>DATA</td>
<td>0E7H</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Mnemonic</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000H</td>
<td>LJMP</td>
<td>MAIN</td>
</tr>
<tr>
<td>00ABH</td>
<td>LJMP</td>
<td>CMPISR</td>
</tr>
<tr>
<td>0100H</td>
<td>ORG</td>
<td></td>
</tr>
</tbody>
</table>

CMPISR:

<table>
<thead>
<tr>
<th>Address</th>
<th>Mnemonic</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANL</td>
<td>CMPCR1, #NOT 40H</td>
<td>Clear interruption sign</td>
</tr>
<tr>
<td>CPL</td>
<td>P1.0</td>
<td>Test port</td>
</tr>
</tbody>
</table>

MAIN:

<table>
<thead>
<tr>
<th>Address</th>
<th>Mnemonic</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV</td>
<td>SP, #3FH</td>
<td></td>
</tr>
</tbody>
</table>
MOV CMPCR2,#00H
MOV CMPCR1,#80H ; Enable comparator module
ORL CMPCR1,#030H ; Enable the edge of the comparator interruption
ANL CMPCR1,#NOT 08H ; P3.6 is the CMP+ input port
ORL CMPCR1,#04H ; P3.7 is the CMP- input port
ORL CMPCR1,#02H ; Enable comparator output
SETB EA

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

NOP

LOOP:
CPL P1.1
JMP LOOP

END

C code
#include "reg51.h"
#include "intrins.h"

sfr CMPCR1 = 0xe6;
sfr CMPCR2 = 0xe7;
sbit P10 = P1^0;
sbit P11 = P1^1;

void CMP_Isr() interrupt 21
{
  CMPCR1 &= ~0x40; // Clear interruption sign
  P10 = !P10; // Test port
}

void main()
{
  CMPCR2 = 0x00; // Enable comparator module
  CMPCR1 = 0x80; // Enable the edge of the comparator interruption
  CMPCR1 |= 0x30; // P3.6 is the CMP+ input port
  CMPCR1 |= 0x04; // P3.7 is the CMP- input port
  CMPCR1 |= 0x02; // Enable comparator output
  EA = 1;
  
  PCON = 0x02; // MCU enters the power down mode
  _nop_(); // Enter interruption service program immediately after power down wake-up
  _nop_();

  while (1)
  {
    P11 = ~P11;
  
  

Nantong guoxin Microelectronics Co., Ltd. Tel: 0513-5501 2928/2929/2966 Fax: 0513-5501 2926/2956/2947 - 113 -
7.4.13 Using LVD function to detect working Voltage(cell voltage)

If you need to use the LVD function to detect the battery voltage, you need to remove the low-voltage reset feature from the ISP download, as shown in the following figure "allow low-voltage reset (no low-voltage interruptions)" hardware options need to be removed.

![Diagram of LVD function settings]

assembly code

```
ORG 0000H
JMP MAIN

ORG 0100H

MAIN:
ANL PCON,#NOT LVDF
MOV RSTCFG,#LVD3V0
LOOP:
MOV A,B,#0FH
MOV RSTCFG,#LVD3V0
CALL DELAY
ANL PCON,#NOT LVDF
CALL DELAY
MOV A,B,#LVD2V7
JZ SKIP
CLR C
RRC A
MOV B,A
MOV RSTCFG,#LVD3V0
CALL DELAY
ANL PCON,#NOT LVDF
CALL DELAY
MOV A,B,
ANL A,#LVDF
JZ SKIP
CLR C
RRC A
MOV B,A
MOV RSTCFG,#LVD2V7
CALL DELAY
ANL PCON,#NOT LVDF
CALL DELAY
MOV A,B,
ANL A,#LVDF
JZ SKIP
```

Nantong guoxin Microelectronics Co., Ltd.  Tel: 0513-5501 2928/2929/2966  Fax: 0513-5501 2926/2956/2947  - 114 -
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 the battery power
JMP LOOP

DELAY:
MOV R0,#100
NEXT:
NOP
NOP
NOP
NOP
DJNZ R0,NEXT
RET

END

C code
#include "reg51.h"
#include "intrins.h"
#define FOSC 24000000UL
#define T1MS (65536 - FOSC/4/100)
#define FOSC 24000000UL
#define T1MS (65536 - FOSC/4/100)
sfr RSTCFG = 0xff;
#define LVD2V2 0x00 //LVD@2.2V
#define LVD2V4 0x01 //LVD@2.4V
#define LVD2V7 0x02 //LVD@2.7V
#define LVD3V0 0x03 //LVD@3.0V
#define LVDF 0x20 //PCON.5

void delay()
{
    int i;

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

void main()
{
    unsigned char power;

    PCON &= ~LVDF;
    RSTCFG = LVD3V0;

    while (1)
    {
        power = 0x0f;

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

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

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

        RSTCFG = LVD2V2;
        delay();
        PCON &= ~LVDF;
        delay();
        if (PCON & LVDF)
        {
            power >>= 1;
        }
    }
}
```c
RSTCFG = LVD2V2;
delay();
PCON & = ~LVDF;
delay();
if (PCON & LVDF)
{
    power >>= 1;
}
}
}
RSTCFG = LVD3V0;
P2 = ~power;  // P2.3~P2.0 display the battery power
```
8 Memory

STC8 Series MCU's program memory and data memory are individually addressable. Because no bus that accesses external program memory is provided, all program memory of all MCU is Flash memory on chip, and external program memory can not be accessed.

The STC8 series single chip microcomputer has integrated the large capacity data memory (STC8A8K64S4A12) inside the STC8A8K64S4A12 series single chip microcomputer with 8192 256-byte data memory and 4096 256-byte data memory inside the STC8A4K64S2A12 series single chip microcomputer. There are 2048 256words inside the STC8A8F2K64S4 series single chip microcomputer. STC8F2K64S2 Series single Chip Microcomputer with 2048 256bytes of data memory. The data memory of STC8 Series single Chip Microcomputer is physically and logically divided into two parts: STC8F2K64S2 series single-chip microcomputer and STC8F2K64S2 series single chip microcomputer. For two address spaces: internal RAM(256 bytes) and internal extension RAM. The high 128-byte data memory of the internal RAM overlaps with the special function register (SFRs) address, which is distinguished by different addressing methods in practical use. In addition, STC8 series microcontroller with 40 or more pins can also access the 64KB external data memory which is extended out of chip.

8.1 Program Memory

Program memory is used to store user programs, data, tables and other information. STC8 series monolithic integrated 64K bytes of Flash program memory.

```
<table>
<thead>
<tr>
<th>64K</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FFFFH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>001BH</td>
<td>0013H</td>
</tr>
<tr>
<td></td>
<td>000BH</td>
<td>0003H</td>
</tr>
<tr>
<td></td>
<td>0000H</td>
<td></td>
</tr>
</tbody>
</table>
```

After reset, the program counter (PC) is 0000H, and the program is executed from 0000H unit. In addition, the entry address of the interrupt service program (also called interrupt vector) is also located in the program memory unit. In the program memory, each interrupt has a fixed entry address. When the interrupt occurs and is responded to, the MCU will automatically jump to the corresponding interrupt entry address to execute the program. The entry address of the interrupt service program is 0003H, the entry address of the timer / counter 0 / TIMER0) interrupt service program is 000BH, and the external interrupt service program's entry address is 000BH. The entry address of interrupt service program is 0013H, and the entry address of interrupt service program of timer / counter 1 / timer is 001BH et al. For more entry addresses (interrupt vectors) for interrupt service programs, please refer to the interrupt introduction section.

Because the interval between adjacent interrupt entry addresses is only 8 bytes, it is generally impossible to save a complete interrupt service program, so an unconditional transfer instruction is stored in the address area of the
interrupt response. Points to the space where the interrupt service program is actually stored to execute.

STC8 series microcontroller contains Flash data memory. The data is read / written in bytes and erased with 512-byte as the page unit. It can be written more than 100000 times by programming repeatedly online, which improves the flexibility and convenience of use.

8.2 Data Memory

The interior-integrated RAM of the column microcontroller can be used to store intermediate results and process data of program execution. STC8A8K64S4A12 series and STC8F2K64S4 series internal integrated RAM have the following differences:

<table>
<thead>
<tr>
<th>the series of chip</th>
<th>Internal direct access RAM (DATA)</th>
<th>Internal direct access RAM (IDATA)</th>
<th>Internal expansion RAM (XDATA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>the series of STC8A8K64S4A12</td>
<td>128 byte</td>
<td>128 byte</td>
<td>8192 byte</td>
</tr>
<tr>
<td>the series of STC8A4K64S2A12</td>
<td>128 byte</td>
<td>128 byte</td>
<td>4096 byte</td>
</tr>
<tr>
<td>the series of STC8F2K64S4</td>
<td>128 byte</td>
<td>128 byte</td>
<td>2048 byte</td>
</tr>
<tr>
<td>the series of STC8F2K64S2</td>
<td>128 byte</td>
<td>128 byte</td>
<td>2048 byte</td>
</tr>
</tbody>
</table>

In addition, the STC8 series microcontroller with 40 or more pins can also access the 64KB external data memory which is extended out of chip.

8.2.1 Internal RAM

The internal RAM is 256-byte and can be divided into two parts: low 128-byte RAM and high 128-byte RAM. The 128-byte data memory is compatible with the traditional 8051 and can be addressed either directly or indirectly. The high 128-byte RAM (extended in 8052) shares the same logical address as the special function register area, using 80H / FFH, but is physically independent and is distinguished by different addressing methods. High 128-byte RAM can only be addressed indirectly, and special function register area can only be addressed directly.

The structure of the internal RAM is shown in the following figure:
A 128-byte low RAM is also known as a general RAM zone. The general RAM area can be divided into working register area, bit-addressable area, user RAM area and stack area. The address of the working register group is divided into four groups from the 32 byte unit of 00H~1FH. Each group is called a register group. Each group contains eight 8-bit working registers, all of which are numbered R0 ~ R7, but belong to different physical spaces. By using the working register group, it is possible to improve the operation speed. R0 / R7 is a common register, and four groups are provided because the first group is often not enough. RS1 and RS in the PSW register of program status word The set of working registers currently used is determined by the 0-0 combination, as described below in the PSW register.

**PSW (program status register)**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSW</td>
<td>D0H</td>
<td>CY</td>
<td>AC</td>
<td>F0</td>
<td>RS1</td>
<td>RS0</td>
<td>OV</td>
<td>-</td>
<td>P</td>
</tr>
</tbody>
</table>

RS1, RS0: Working register selection bit

<table>
<thead>
<tr>
<th>RS1</th>
<th>RS0</th>
<th>Work register group (R0~R7)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>The Zeroth groups (00H~07H)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>The first group (08H~0FH)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>The second group (10H~17H)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>The third group (18H~1FH)</td>
</tr>
</tbody>
</table>

The address of the addressable region is 16 byte units from 20H ~ 2FH. The 20H ~ 2FH 2FH unit can be accessed by bytes as ordinary RAM cells, or by any single bit in the unit. The address range of logical bit address is 00H / 7FH. The bit-address range is 00H~7FH, and the internal RAM address is 128-byte low, so the address is the same as the other. In fact, the two addresses are essentially different; the bit-address refers to a bit, and the byte address points to a byte unit. Use different instructions in a program Distinguish.

The 30H~FFH unit in the internal RAM is the user RAM and stack area. An 8-bit stack pointer is used to point to the stack area. After reset, the stack pointer SP is 07H, pointing to R7 in the working register group 0. Therefore,
the user initialization program should set the initial value to SP, which is suitable for the unit after 80H.

The stack pointer is an 8-bit special register. It indicates where the top of the stack is in the internal RAM block. After the system reset, the SP initialization bit 07H makes the stack actually start from 08H unit, considering that the 08H~1FH unit belongs to the working register group 1 / 3 respectively, if these areas are used in the program design, it is better to change the SP value to 80H or greater. The stack of STC8 series microcontroller is grown up, that is, the content of SP increases after the data is pressed onto the stack.

8.2.2 Internal extended RAM

STC8 series microcontroller chip in addition to the integration of 256-byte internal RAM, but also integrated the internal expansion RAM. The method of accessing the internal extended RAM is the same as that of the traditional 8051 single chip computer to access the external extended RAM, but it does not affect the P0 port (data bus and high 8-bit address bus / P2 port), as well as the signals on the ports such as RDWR and ALE.

In assembly language, the internal extended RAM is accessed by a move X instruction:

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

In C, xdata/pdata can be used to declare the storage type. Such as:

unsigned char xdata i;
unsigned int pdata j;

After subscribing to the variable of type pdata in C language, the compiler automatically allocates the variable to the 0000H~00FFH area of XDATA and accesses it by using MOVX 0000H~00FFH Riga and MOVX A@Ri.

Whether the RAM can be accessed is controlled by the EXTRAM bit in the auxiliary register AUXR.

**AUXR(auxiliary register)**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUXR</td>
<td>8EH</td>
<td></td>
<td></td>
<td>UART_M0x6</td>
<td>T2R</td>
<td>T2_C/T</td>
<td>T2x12</td>
<td>EXTRAM</td>
<td>S1ST2</td>
</tr>
</tbody>
</table>

EXTRAM: Extended RAM access control

- 0: Access the internal extension RAM. When the access address exceeds the address of the internal extension RAM, the system automatically switches to the external extension RAM.
- 1: Access to the external extension RAM, the internal extension RAM is disabled. Extended RAM access control.
8.2.3 External extended RAM

The STC8 series microcontroller with 40 or more pins has the ability to extend the 64KB external data memory. During access to external data memory, the WR / RD / ale signal must be effective. STC8 series single chip computers have added a special function register, bus SPEED which controls the speed of the external 64K byte data bus. The description is as follows:

**BUS_SPEED (Bus speed control register)**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>address</th>
<th>address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUS_SPEED</td>
<td>A1H</td>
<td>RW_S[1:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| RW_S[1:0]: RD/WR Control line selection bit
  00: P4.4 is RD, P4.3 is WR
  01: P3.7 is RD, P3.6 is WR
  10: P4.2 is RD, P4.0 is WR
  11: stay |
| SPEED[1:0]: Bus read-write speed control(Time of preparation and retention of control signals and data signals when reading and writing data) |
8.3 Special Parameters in Memory

Some special parameters related to the chip are stored in the data memory and program memory of STC8 series single chip microcomputer, including the frequency of the global unique ID number 32K shutdown wake-up timer, the internal Bandgap voltage value and the IRC parameter.

The location of these parameters in the program memory ROM is as follows:

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Save address</th>
<th>Parameter description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The only ID in the world</td>
<td>3FF9H<del>3FFFH 7FF9H</del>7FFFH 0EFF9H<del>0EFFFH 0FDF9H</del>0FDFFH</td>
<td>7byte</td>
</tr>
<tr>
<td>Bandgap voltage value</td>
<td>3FF7H<del>3FF8H 7FF7H</del>7FF8H 0EFF7H<del>0EFF8H 0FDF7H</del>0FDF8H</td>
<td>Voltage unit is mv</td>
</tr>
<tr>
<td>The frequency of 32K power down wake-up timer</td>
<td>3FF5H<del>3FF6H 7FF5H</del>7FF6H 0EFF5H<del>0EFF6H 0FDF5H</del>0FDF6H</td>
<td>Unit Hz</td>
</tr>
<tr>
<td>IRC parameters of 22.1184MHz</td>
<td>3FF4H 7FF4H 0EFF4H 0FDF4H</td>
<td>—</td>
</tr>
<tr>
<td>IRC parameters of 24MHz</td>
<td>3FF3H 7FF3H 0EFF3H 0FDF3H</td>
<td>—</td>
</tr>
</tbody>
</table>

The storage addresses of these parameters in the data memory RAM are as follows:

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Save address</th>
<th>Parameter description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandgap voltage value</td>
<td>iData: 0EFH~0F0H</td>
<td>Voltage unit is mv, high bytes are in the front</td>
</tr>
<tr>
<td>The only ID in the world</td>
<td>iData: 0F1H~0F7H</td>
<td>7byte</td>
</tr>
<tr>
<td>The frequency of 32K power down wake-up timer</td>
<td>iData: 0F8H~0F9H</td>
<td>Unit Hz, high bytes are in the front</td>
</tr>
<tr>
<td>IRC parameters of 22.1184MHz</td>
<td>iData: 0FAH</td>
<td>—</td>
</tr>
<tr>
<td>IRC parameters of 24MHz</td>
<td>iData: 0FBH</td>
<td>—</td>
</tr>
</tbody>
</table>
Special Explanation

1. Since the parameters in RAM may be modified, it is generally not recommended for users to use them, especially when they use ID numbers to encrypt, which is strongly recommended to read ID data in ROM.

Because STC8A8K64S4A10, STC8A4K64S2A10, STC8F2K64S4 and STC8F2K64S2, four types of EEPROM users can set their own size, it is possible to set the ROM space where important parameters are stored to EEPROM and artificially erase or modify important parameters. So use these 4 models for ID number encryption may need to consider this issue.

2. By default, the program memory has only global unique ID number data, while the Bandgap voltage value of 32K power-off timer frequency and IRC parameters are not available, you need to select the options shown in the following figure when downloading ISP. Since the parameters in RAM may be modified, it is generally not recommended for users to use them, especially when they use ID numbers to encrypt, which is strongly recommended to read ID data in ROM.

Because STC8A8K64S4A10, STC8A4K64S2A10, STC8F2K64S4 and STC8F2K64S2, four types of EEPROM users can set their own size, it is possible to set the ROM space where important parameters are stored to EEPROM and artificially erase or modify important parameters. So use these 4 models for ID number encryption may need to consider this issue.

3. By default, the program memory has only global unique ID number data, while the Bandgap voltage value of 32K power-off timer frequency and IRC parameters are not available, you need to select the options shown in the following figure when downloading ISP.
8.3.1  Read the Bandgap voltage (Read from ROM)

assembly code

<table>
<thead>
<tr>
<th>AUXR</th>
<th>DATA</th>
<th>8EH</th>
</tr>
</thead>
<tbody>
<tr>
<td>BGV</td>
<td>EQU</td>
<td>0FD7H</td>
</tr>
<tr>
<td>:BGV</td>
<td>EQU</td>
<td>0EFF7H</td>
</tr>
<tr>
<td>;BGV</td>
<td>EQU</td>
<td>07FF7H</td>
</tr>
<tr>
<td>;BGV</td>
<td>EQU</td>
<td>03FF7H</td>
</tr>
</tbody>
</table>

BUSY BIT 20H.0

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,S
SETB BUSY
MOV SBUF,A

RET

MAIN:
MOV SP,#3FH
LCALL UART_INIT
SETB ES
SETB EA

MOV DPTR,#BGV
CLR A
MOVC A,@A+DPTR ; // Read the high byte of the Bandgap voltage
LCALL UART_SEND
MOV A,#1
MOVC A,@A+DPTR ; // Read the low byte of the Bandgap voltage
LCALL UART_SEND

LOOP:
JMP LOOP

END

C code
#include "reg51.h"
#include "intrins.h"
#define FOSC 11059200UL
#define BRT (65536 - FOSC / 115200 / 4)
sfr AUXR = 0x8e;
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()
{
    BGV = (int code *)0xfdf7; // STC8A8K64S4A10
    // BGV = (int code *)0xeff7; // STC8A8K60S4A10
    // BGV = (int code *)0x7ff7; // STC8A8K32S4A10
    // BGV = (int code *)0x3ff7; // STC8A8K16S4A10
    UartInit();
    ES = 1;
    EA = 1;
    UARTsend(*BGV >> 8); // Read the high byte of the Bandgap voltage
    UARTsend(*BGV); // Read the low byte of the Bandgap voltage
    while (1);
}

8.3.2 Read the Bandgap voltage (Read from RAM)

assembly code

<table>
<thead>
<tr>
<th>AUXR</th>
<th>DATA</th>
<th>8EH</th>
</tr>
</thead>
<tbody>
<tr>
<td>BGV</td>
<td>DATA</td>
<td>0EFH</td>
</tr>
<tr>
<td>BUSY</td>
<td>BIT</td>
<td>20H 0</td>
</tr>
<tr>
<td>ORG</td>
<td>0000H</td>
<td></td>
</tr>
<tr>
<td>LJMP</td>
<td>MAIN</td>
<td></td>
</tr>
<tr>
<td>ORG</td>
<td>0023H</td>
<td></td>
</tr>
<tr>
<td>LJMP</td>
<td>UART_ISR</td>
<td></td>
</tr>
<tr>
<td>ORG</td>
<td>0100H</td>
<td></td>
</tr>
</tbody>
</table>

UART_ISR:

JNB T1, CHKRI
CLR T1
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, #3FH
LCALL UART_INIT
SETB ES
SETB EA
MOV R0, #BGV
MOV A, @R0 ;// Read the high byte of the Bandgap voltage
LCALL UART_SEND
INC R0
MOV A, @R0 ; // Read the low byte of the Bandgap voltage
LCALL UART_SEND
LOOP:
JMP LOOP
END

C code
#include "reg51.h"
#include "intrins.h"

#define FOSC 11059200UL
#define BRT (65536 - FOSC / 115200 / 4)
sfr AUXR = 0x8e;
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;
\[ TH1 = BRT >> 8; \]
\[ TR1 = 1; \]
\[ AUXR = 0x40; \]
\[ busy = 0; \]
\[
void UARTsend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}
\]
\[
void main()
{
    BGV = (int idata *)0xef;
    UartInit();
    ES = 1;
    EA = 1;
    UARTsend(*BGV >> 8); // Read the high byte of the Bandgap voltage
    UARTsend(*BGV); // Read the low byte of the Bandgap voltage
    while (1);
}
\]

8.3.3 Read the global unique ID number (Read from ROM)

assembly code

<table>
<thead>
<tr>
<th>AUXR</th>
<th>DATA</th>
<th>8EH</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>EQU</td>
<td>0FDF9H</td>
</tr>
<tr>
<td>;ID</td>
<td>EQU</td>
<td>0EFF9H</td>
</tr>
<tr>
<td>;ID</td>
<td>EQU</td>
<td>07FF9H</td>
</tr>
<tr>
<td>;ID</td>
<td>EQU</td>
<td>03FF9H</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BUSY</th>
<th>BIT</th>
<th>20H.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORG</td>
<td>0000H</td>
<td></td>
</tr>
<tr>
<td>LJMP</td>
<td>MAIN</td>
<td></td>
</tr>
<tr>
<td>ORG</td>
<td>0023H</td>
<td></td>
</tr>
<tr>
<td>LJMP</td>
<td>UART_ISR</td>
<td></td>
</tr>
<tr>
<td>ORG</td>
<td>0100H</td>
<td></td>
</tr>
</tbody>
</table>

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,#3FH
    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

C code
#include "reg51.h"
#include "intrins.h"

#define FOSC 11059200UL
#define BRT (65536 - FOSC / 115200 / 4)
sfr AUXR = 0x8e;
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;

    ID = (char code *)[0xfdf9]; // STC8A8K64S4A10
    // ID = (char code *)[0xef9]; // STC8A8K60S4A10
    // ID = (char code *)[0x7ff9]; // STC8A8K32S4A10
    // ID = (char code *)[0x3ff9]; // STC8A8K16S4A10

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

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

    while (1);
}

8.3.4 Read the global unique ID number (Read from RAM)

assembly code

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AUXR</td>
<td>DATA</td>
<td>8EH</td>
</tr>
<tr>
<td>ID</td>
<td>DATA</td>
<td>0F1H</td>
</tr>
</tbody>
</table>
BUSY  BIT  20H.0

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  ;65536-110200/115200/4=0FFE8H
MOV  TH1,#0FFH
SETB  TR1
MOV  AUXR,#40H
CLR  BUSY

RET

UART_SEND:
JB  BUSYS
SETB  BUSYS
MOV  SBUF,A

RET

MAIN:
MOV  SP,#3FH
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
C code

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

#define FOSC 11059200UL
#define BRT (65536 - FOSC / 115200 / 4)
sfr AUXR = 0x8e;

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;

    ID = (char idata *)0xf1;
    UartInit();
    ES = 1;
    EA = 1;
for (i=0; i<7; i++)
{
    UARTsend(ID[i]);
}

while (1);

8.3.5 Read the frequency of the 32K power down wake-up timer
(Read from ROM)

assembly code

<table>
<thead>
<tr>
<th>DATA</th>
<th>AUXR</th>
<th>F32K</th>
</tr>
</thead>
<tbody>
<tr>
<td>8EH</td>
<td>EQU</td>
<td>0FD5H</td>
</tr>
<tr>
<td></td>
<td>;STC8A8K64S4A10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EQU</td>
<td>0EFF5H</td>
</tr>
<tr>
<td></td>
<td>;STC8A8K60S4A10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EQU</td>
<td>07FF5H</td>
</tr>
<tr>
<td></td>
<td>;STC8A8K32S4A10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EQU</td>
<td>03FF5H</td>
</tr>
<tr>
<td></td>
<td>;STC8A8K16S4A10</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BUSY</th>
<th>BIT</th>
<th>20H.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORG</td>
<td>0000H</td>
<td></td>
</tr>
<tr>
<td>LJMP</td>
<td>MAIN</td>
<td></td>
</tr>
<tr>
<td>ORG</td>
<td>0023H</td>
<td></td>
</tr>
<tr>
<td>LJMP</td>
<td>UART_ISR</td>
<td></td>
</tr>
<tr>
<td>ORG</td>
<td>0100H</td>
<td></td>
</tr>
</tbody>
</table>

UART_ISR:

<table>
<thead>
<tr>
<th>JNB</th>
<th>TI,CHKRI</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>TI</td>
</tr>
<tr>
<td>CLR</td>
<td>BUSY</td>
</tr>
</tbody>
</table>

CHKRI:

<table>
<thead>
<tr>
<th>JNB</th>
<th>RI,UARTISR_EXIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLR</td>
<td>RI</td>
</tr>
</tbody>
</table>

UARTISR_EXIT:

<table>
<thead>
<tr>
<th>RETI</th>
<th></th>
<th></th>
</tr>
</thead>
</table>

UART_INIT:

<table>
<thead>
<tr>
<th>MOV</th>
<th>SCON,#50H</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV</td>
<td>TMOD,#00H</td>
</tr>
<tr>
<td>MOV</td>
<td>TL1,#0E8H</td>
</tr>
<tr>
<td>MOV</td>
<td>TH1,#0FFH</td>
</tr>
<tr>
<td>SETB</td>
<td>TR1</td>
</tr>
<tr>
<td>MOV</td>
<td>AUXR,#40H</td>
</tr>
<tr>
<td>CLR</td>
<td>BUSY</td>
</tr>
</tbody>
</table>

RET

UART_SEND:

<table>
<thead>
<tr>
<th>JB</th>
<th>BUSY,S</th>
</tr>
</thead>
<tbody>
<tr>
<td>SETB</td>
<td>BUSY</td>
</tr>
<tr>
<td>MOV</td>
<td>SBUF,A</td>
</tr>
</tbody>
</table>

RET
MAIN:

    MOV SP, #3FH
    LCALL UART_INIT
    SETB ES
    SETB EA
    MOV DPTR, #F32K
    CLR A
    MOVC A, @A + DPTR ; Reading high bytes of the 32K frequency
    LCALL UART_SEND
    INC DPTR
    CLR A
    MOVC A, @A + DPTR ; Reading low bytes of the 32K frequency
    LCALL UART_SEND

LOOP:

    JMP LOOP

END

C code

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

#define FOSC 11059200UL
#define BRT (65536 - FOSC / 115200 / 4)
sfr AUXR = 0x8e;
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;
```c
AUXR = 0x40;
busy = 0;
}

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

void main()
{
    F32K = (int code *)0xfdf5; // STC8A8K64S4A10
    // F32K = (int code *)0xeff5; // STC8A8K60S4A10
    // F32K = (int code *)0x7ff5; // STC8A8K32S4A10
    // F32K = (int code *)0x3ff5; // STC8A8K16S4A10
    UartInit();
    ES = 1;
    EA = 1;

    UARTsend(*F32K >> 8); // Reading high bytes of the 32K frequency
    UARTsend(*F32K); // Reading low bytes of the 32K frequency

    while (1);
}
```

### 8.3.6 Read the frequency of the 32K power down wake-up timer
(Read from RAM)

**Assembly code**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>DATA/Bit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUXR</td>
<td>DATA</td>
<td>8EH</td>
</tr>
<tr>
<td>F32K</td>
<td>DATA</td>
<td>0F8H</td>
</tr>
<tr>
<td>BUSY</td>
<td>BIT</td>
<td>20H.0</td>
</tr>
</tbody>
</table>

```
ORG 0000H
LJMP MAIN
ORG 0023H
LJMP UART_ISR

ORG 0100H

UART_ISR:
    JNB TL, 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,#3FH
    LCALL UART_INIT
    SETB ES
    SETB EA
    MOV R0,#F32K
    MOV A,@R0 ; Reading high bytes of the 32K frequency
    LCALL UART_SEND
    INC R0
    MOV A,@R0 ; Reading low bytes of the 32K frequency
    LCALL UART_SEND
    LOOP:
        JMP LOOP
    END

C code
#include "reg51.h"
#include "intrins.h"

#define FOSC 11059200UL
#define BRT (65536 - FOSC / 115200 / 4)
sfr AUXR = 0x8e;
bit busy;
int *F32K;

void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
```c
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()
{
    F32K = (intidata *)0xf8;
    UartInit();
    ES = 1;
    EA = 1;
    UARTsend(*F32K >> 8);  //Reading high bytes of the 32K frequency
    UARTsend(*F32K);      // Reading low bytes of the 32K frequency
    while (1);
}
```

### 8.3.7 Manually set the internal IRC frequency (Read from ROM assembly code)

<table>
<thead>
<tr>
<th>P_SW2</th>
<th>DATA</th>
<th>0BAH</th>
</tr>
</thead>
<tbody>
<tr>
<td>CKSEL</td>
<td>EQU</td>
<td>0FE00H</td>
</tr>
<tr>
<td>CLKDIV</td>
<td>EQU</td>
<td>0FE01H</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IRCCR</th>
<th>DATA</th>
<th>09FH</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRC22M</td>
<td>EQU</td>
<td>0FDF4H ; STC8A8K64S4A10</td>
</tr>
<tr>
<td>IRC24M</td>
<td>EQU</td>
<td>0FDF3H</td>
</tr>
<tr>
<td>IRC22M</td>
<td>EQU</td>
<td>0EFF4H ; STC8A8K60S4A10</td>
</tr>
<tr>
<td>IRC24M</td>
<td>EQU</td>
<td>0EFF3H</td>
</tr>
<tr>
<td>IRC22M</td>
<td>EQU</td>
<td>07FF4H ; STC8A8K32S4A10</td>
</tr>
<tr>
<td>IRC24M</td>
<td>EQU</td>
<td>07FF3H</td>
</tr>
</tbody>
</table>
;IRC22M EQU 03FF4H ; STC8A8K16S4A10
;IRC24M EQU 03FF3H

ORG 0000H
LJMP MAIN

ORG 0100H

MAIN:

MOV SP,#3FH
; MOV DPTR,#IRC22M ; loading IRC parameters for 22.1184MHz
; CLR A
; MOVC A,@A+DPTR
; MOV IRCCR,A
MOV DPTR,#IRC24M ; loading IRC parameters for 24MHz
CLR A
MOV A,@A+DPTR
MOV IRCCR,A

MOV P_SW2,#80H
MOV A,#0 ; Master clock without prescale
MOV DPTR,#CLKDIV
MOVX @DPTR,A
MOV A,#40H ; Master clock 4 frequency division output to P5.4 port
MOV DPTR,#CKSEL
MOVX @DPTR,A
MOV P_SW2,#00H
JMP $

END

C code
#include "reg51.h"
#include "intrins.h"

#define CKSEL (*(unsigned char volatile xdata *)0xfe00)
#define CLKDIV (*(unsigned char volatile xdata *)0xfe01)
sfr P_SW2 = 0xba;
sfr IRCCR = 0x9f;
char *IRC22M;
char *IRC24M;

void main()
{
    IRC22M = (char code *)0xdf4; // STC8A8K64S4A10
    IRC24M = (char code *) 0xdf3;
    // IRC22M = (char code *)0xff4; // STC8A8K60S4A10
    // IRC24M = (char code *) 0xff3;
    // IRC22M = (char code *)0xff4; // STC8A8K32S4A10
    // IRC24M = (char code *) 0xff3;
    // IRC22M = (char code *)0xff4; // STC8A8K16S4A10
    // IRC24M = (char code *) 0xff3;
}
8.3.8 Manually set the internal IRC frequency (Read from RAM)

**Assembly code**

```
P_SW2 DATA 0BAH
CKSEL EQU 0FE00H
CLKDIV EQU 0FE01H
IRCCR DATA 09FH
IRC22M DATA 0FAH
IRC24M DATA 0FBH

ORG 0000H
LJMP MAIN

ORG 0100H

MAIN:
MOV SP,#3FH
MOV R0,#IRC22M ; loading IRC parameters for 22.1184MHz
MOV IRCCR,@R0
MOV R0,#IRC24M ; loading IRC parameters for 24MHz
MOV IRCCR,@R0
MOV P_SW2,#80H ; Master clock without prescale
MOV A,#0
MOV DPTR,#CLKDIV
MOVX @DPTR,A
MOV A,#40H ; Master clock 4 frequency division output to P5.4 port
MOV DPTR,#CKSEL
MOVX @DPTR,A
MOV P_SW2,#00H

JMP $```

**C code**

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

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

#define P_SW2 (*volatile unsigned char *)0x80
#define CLKDIV (*volatile unsigned char *)0x40
```

Nantong guoxin Microelectronics Co., Ltd. Tel: 0513-5501 2928/2929/2966 Fax: 0513-5501 2926/2956/2947
sfr P_SW2 = 0xba;
sfr IRCCR = 0x9f;

char *IRC22M;
char *IRC24M;

void main()
{
  IRC22M = (char idata *)0xfa;
  IRC24M = (char idata *) 0xfb;
  // IRCCR = *IRC22M; // loading IRC parameters for 22.1184MHz
  IRCCR = *IRC24M; // loading IRC parameters for 24MHz

  P_SW2 = 0x80;
  CLKDIV = 0; // Master clock without prescale
  CKSEL = 0x40; // Master clock 4 frequency division output to P5.4 port
  P_SW2 = 0x00;

  while (1);
}
### 9 Special Function Register

#### 9.1 Series of STC8A8K64S4A12

<table>
<thead>
<tr>
<th>0/8</th>
<th>1/9</th>
<th>2/A</th>
<th>3/B</th>
<th>4/C</th>
<th>5/D</th>
<th>6/E</th>
<th>7/F</th>
</tr>
</thead>
<tbody>
<tr>
<td>F8H</td>
<td>P7</td>
<td>CH</td>
<td>CCAP0H</td>
<td>CCAP1H</td>
<td>CCAP2H</td>
<td>CCAP3H</td>
<td>PWMCR</td>
</tr>
<tr>
<td>F0H</td>
<td>B</td>
<td>PWMCFG</td>
<td>PCA_PWM0</td>
<td>PCA_PWM1</td>
<td>PCA_PWM2</td>
<td>PCA_PWM3</td>
<td>PWMIF</td>
</tr>
<tr>
<td>E8H</td>
<td>P6</td>
<td>CL</td>
<td>CCAP0L</td>
<td>CCAP1L</td>
<td>CCAP2L</td>
<td>CCAP3L</td>
<td>AUXINTF</td>
</tr>
<tr>
<td>E0H</td>
<td>ACC</td>
<td>P7M1</td>
<td>P7M0</td>
<td>DPS</td>
<td>DPL1</td>
<td>DPH1</td>
<td>CMPCR1</td>
</tr>
<tr>
<td>D8H</td>
<td>CCON</td>
<td>CMOD</td>
<td>CCAPM0</td>
<td>CCAPM1</td>
<td>CCAPM2</td>
<td>CCAPM3</td>
<td>ADCCFG</td>
</tr>
<tr>
<td>D0H</td>
<td>PSW</td>
<td>T4T3M</td>
<td>T4H</td>
<td>T4L</td>
<td>T3H</td>
<td>T3L</td>
<td>T2H</td>
</tr>
<tr>
<td>C8H</td>
<td>P5</td>
<td>P5M1</td>
<td>P5M0</td>
<td>P6M1</td>
<td>P6M0</td>
<td>SPSTAT</td>
<td>SPCTL</td>
</tr>
<tr>
<td>C0H</td>
<td>P4</td>
<td>WDT_CTRL</td>
<td>IAP_DATA</td>
<td>IAP_ADDRH</td>
<td>IAP_ADDRL</td>
<td>IAP_CMD</td>
<td>IAP_TRIG</td>
</tr>
<tr>
<td>B8H</td>
<td>IP</td>
<td>SADEN</td>
<td>P_SW2</td>
<td>VOCTRL</td>
<td>ADC_CONTR</td>
<td>ADC_RES</td>
<td>ADC_RESEL</td>
</tr>
<tr>
<td>B0H</td>
<td>P3</td>
<td>P3M1</td>
<td>P3M0</td>
<td>P4M1</td>
<td>P4M0</td>
<td>IP2</td>
<td>IP2H</td>
</tr>
<tr>
<td>A8H</td>
<td>IE</td>
<td>SADDR</td>
<td>WKTCL</td>
<td>WKTCH</td>
<td>S3CON</td>
<td>S3BUF</td>
<td>TA</td>
</tr>
<tr>
<td>A0H</td>
<td>P2</td>
<td>BUS_SPEED</td>
<td>P_SW1</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>98H</td>
<td>SCON</td>
<td>SBUF</td>
<td>S2CON</td>
<td>S2BUF</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>90H</td>
<td>P1</td>
<td>P1M1</td>
<td>P1M0</td>
<td>P0M1</td>
<td>P0M0</td>
<td>P2M1</td>
<td>P2M0</td>
</tr>
<tr>
<td>88H</td>
<td>TCON</td>
<td>TMOD</td>
<td>TL0</td>
<td>TL1</td>
<td>TH0</td>
<td>TH1</td>
<td>AUXR</td>
</tr>
<tr>
<td>80H</td>
<td>P0</td>
<td>SP</td>
<td>DPL</td>
<td>DPH</td>
<td>S4CON</td>
<td>S4BUF</td>
<td>PCON</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0/8</th>
<th>1/9</th>
<th>2/A</th>
<th>3/B</th>
<th>4/C</th>
<th>5/D</th>
<th>6/E</th>
<th>7/F</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFF0H</td>
<td>PWMCH</td>
<td>PWMCL</td>
<td>PWMCKS</td>
<td>TADCPH</td>
<td>TADCL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FF70H</td>
<td>PWM7T1H</td>
<td>PWM7T1L</td>
<td>PWM7T2H</td>
<td>PWM7T2L</td>
<td>PWM7CR</td>
<td>PWM7HLD</td>
<td></td>
</tr>
<tr>
<td>FF60H</td>
<td>PWM6T1H</td>
<td>PWM6T1L</td>
<td>PWM6T2H</td>
<td>PWM6T2L</td>
<td>PWM6CR</td>
<td>PWM6HLD</td>
<td></td>
</tr>
<tr>
<td>FF50H</td>
<td>PWM5T1H</td>
<td>PWM5T1L</td>
<td>PWM5T2H</td>
<td>PWM5T2L</td>
<td>PWM5CR</td>
<td>PWM5HLD</td>
<td></td>
</tr>
<tr>
<td>FF40H</td>
<td>PWM4T1H</td>
<td>PWM4T1L</td>
<td>PWM4T2H</td>
<td>PWM4T2L</td>
<td>PWM4CR</td>
<td>PWM4HLD</td>
<td></td>
</tr>
<tr>
<td>FF30H</td>
<td>PWM3T1H</td>
<td>PWM3T1L</td>
<td>PWM3T2H</td>
<td>PWM3T2L</td>
<td>PWM3CR</td>
<td>PWM3HLD</td>
<td></td>
</tr>
<tr>
<td>FF20H</td>
<td>PWM2T1H</td>
<td>PWM2T1L</td>
<td>PWM2T2H</td>
<td>PWM2T2L</td>
<td>PWM2CR</td>
<td>PWM2HLD</td>
<td></td>
</tr>
<tr>
<td>FF10H</td>
<td>PWM1T1H</td>
<td>PWM1T1L</td>
<td>PWM1T2H</td>
<td>PWM1T2L</td>
<td>PWM1CR</td>
<td>PWM1HLD</td>
<td></td>
</tr>
<tr>
<td>FF00H</td>
<td>PWM0T1H</td>
<td>PWM0T1L</td>
<td>PWM0T2H</td>
<td>PWM0T2L</td>
<td>PWM0CR</td>
<td>PWM0HLD</td>
<td></td>
</tr>
<tr>
<td>FE80H</td>
<td>I2CCFG</td>
<td>I2CMSCR</td>
<td>I2CMSST</td>
<td>I2CSLCR</td>
<td>I2CSLST</td>
<td>I2CSLADR</td>
<td>I2CTXD</td>
</tr>
<tr>
<td>FE18H</td>
<td>P0NCS</td>
<td>P1NCS</td>
<td>P2NCS</td>
<td>P3NCS</td>
<td>P4NCS</td>
<td>P5NCS</td>
<td>P6NCS</td>
</tr>
<tr>
<td>FE10H</td>
<td>POPU</td>
<td>P1PU</td>
<td>P2PU</td>
<td>P3PU</td>
<td>P4PU</td>
<td>P5PU</td>
<td>P6PU</td>
</tr>
<tr>
<td>FE00H</td>
<td>CKSEL</td>
<td>CLKDIV</td>
<td>IRC24MCR</td>
<td>XOSCRR</td>
<td>IRC32KCR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## 9.2 Series of STC8A4K64S2A12

<table>
<thead>
<tr>
<th>Location</th>
<th>Address</th>
<th>Register Name</th>
<th>Bit Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.2</td>
<td>0/8</td>
<td>F8H</td>
<td>P7 CH</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CCAP0H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CCAP1H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CCAP2H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CCAP3H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWMCR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RSTCFG</td>
</tr>
<tr>
<td></td>
<td>1/9</td>
<td>F0H</td>
<td>B PWMCFG</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PCA_PW0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PCA_PW1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PCA_PW2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PCA_PW3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PCA_PW4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PCA_PW5</td>
</tr>
<tr>
<td></td>
<td>2/A</td>
<td>E8H</td>
<td>P6 CL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CCAP0L</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CCAP1L</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CCAP2L</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CCAP3L</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWMSF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWMS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWMSFDCR</td>
</tr>
<tr>
<td></td>
<td>3/B</td>
<td>E0H</td>
<td>ACC P7M1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P7M0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DPL1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DPH1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CMP CR1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CMP CR2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AUXINTIF</td>
</tr>
<tr>
<td></td>
<td>4/C</td>
<td>D8H</td>
<td>CCON CMOD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CCAPM0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CCAPM1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CCAPM2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CCAPM3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ADCCFG</td>
</tr>
<tr>
<td></td>
<td>5/D</td>
<td>D0H</td>
<td>PSW</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>T4T3M</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>T4H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>T4L</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>T3H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>T3L</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>T2H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>T2L</td>
</tr>
<tr>
<td></td>
<td>6/E</td>
<td>C8H</td>
<td>P5 P5M1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P5M0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P6M1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P6M0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SPSTAT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SPCTL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SPDAT</td>
</tr>
<tr>
<td></td>
<td>7/F</td>
<td>C0H</td>
<td>P4 WDT_CONTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IAP_DATA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IAP ADDRH</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IAP ADDRDL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IAP_CMD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IAP TRIG</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IAP CONTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ADC_RESL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ADC RES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ADC RESL</td>
</tr>
<tr>
<td></td>
<td>0/8</td>
<td>B8H</td>
<td>IP SADV</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PW2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>VOCTRL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ADC CONTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ADC RES</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ADC RESL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ADC RESL</td>
</tr>
<tr>
<td></td>
<td>1/9</td>
<td>B0H</td>
<td>P3 P3M1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P3M0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P4M1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P4M0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IP2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IP2H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IPH</td>
</tr>
<tr>
<td></td>
<td>2/A</td>
<td>A8H</td>
<td>IE SADDR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WKTCI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WKTCI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IE2</td>
</tr>
<tr>
<td></td>
<td>3/B</td>
<td>A0H</td>
<td>P2 BUS_SPEED</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P_SW1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>4/C</td>
<td>98H</td>
<td>SCON SBUF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>S2CON</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>S2BUF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>5/D</td>
<td>90H</td>
<td>P1 P1M1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P1M0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P0M1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P0M0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P2M1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P2M0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AUXR2</td>
</tr>
<tr>
<td></td>
<td>6/E</td>
<td>88H</td>
<td>TCON TMOD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TL0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TL1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TH0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TH1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AUXR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INT CLK KO</td>
</tr>
<tr>
<td></td>
<td>7/F</td>
<td>80H</td>
<td>P0 SP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DPL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DPH</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PCON</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Location</th>
<th>Address</th>
<th>Register Name</th>
<th>Bit Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FFF0H</td>
<td>PWMCH</td>
<td>PWMCL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWMCKS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TADCPH</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TADCPH</td>
</tr>
<tr>
<td></td>
<td>FFF1H</td>
<td>PWM7T1H</td>
<td>PWM7T1L</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM7T2H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM7T2L</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM7CR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM7HLD</td>
</tr>
<tr>
<td></td>
<td>FFF2H</td>
<td>PWM6T1H</td>
<td>PWM6T1L</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM6T2H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM6T2L</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM6CR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM6HLD</td>
</tr>
<tr>
<td></td>
<td>FFF3H</td>
<td>PWM5T1H</td>
<td>PWM5T1L</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM5T2H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM5T2L</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM5CR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM5HLD</td>
</tr>
<tr>
<td></td>
<td>FFF4H</td>
<td>PWM4T1H</td>
<td>PWM4T1L</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM4T2H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM4T2L</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM4CR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM4HLD</td>
</tr>
<tr>
<td></td>
<td>FFF5H</td>
<td>PWM3T1H</td>
<td>PWM3T1L</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM3T2H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM3T2L</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM3CR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM3HLD</td>
</tr>
<tr>
<td></td>
<td>FFF6H</td>
<td>PWM2T1H</td>
<td>PWM2T1L</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM2T2H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM2T2L</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM2CR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM2HLD</td>
</tr>
<tr>
<td></td>
<td>FFF7H</td>
<td>PWM1T1H</td>
<td>PWM1T1L</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM1T2H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM1T2L</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM1CR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM1HLD</td>
</tr>
<tr>
<td></td>
<td>FFF8H</td>
<td>PWM0T1H</td>
<td>PWM0T1L</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM0T2H</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM0T2L</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM0CR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PWM0HLD</td>
</tr>
<tr>
<td></td>
<td>FFF9H</td>
<td>I2CCFG</td>
<td>I2CMSCTL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I2CMSST</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I2CSLCTL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I2CSLSTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I2CTXD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I2CRxD</td>
</tr>
<tr>
<td></td>
<td>FFE0H</td>
<td>P0NCS</td>
<td>P1NCS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P2NCS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P3NCS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P4NCS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P5NCS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P6NCS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P7NCS</td>
</tr>
<tr>
<td></td>
<td>FFE1H</td>
<td>P0PU</td>
<td>P1PU</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P2PU</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P3PU</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P4PU</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P5PU</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P6PU</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P7PU</td>
</tr>
<tr>
<td></td>
<td>FFE2H</td>
<td>CKSEL</td>
<td>CLKDIV</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IRC24MCR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IRC32KCR</td>
</tr>
</tbody>
</table>

Nantong guoxin Microelectronics Co., Ltd. Tel: 0513-5501 2928/2929/2966 Fax: 0513-5501 2926/2956/2947 - 143 -
## 9.3 Series of STC8F2K64S4

<table>
<thead>
<tr>
<th>Address</th>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F8H</td>
<td>P7</td>
<td>RSTCFG</td>
</tr>
<tr>
<td>F0H</td>
<td>B</td>
<td>PWMCFG</td>
</tr>
<tr>
<td>E8H</td>
<td>P6</td>
<td>AUXINTIF</td>
</tr>
<tr>
<td>E0H</td>
<td>ACC</td>
<td>CMPCR1</td>
</tr>
<tr>
<td>D8H</td>
<td></td>
<td>CMPCR2</td>
</tr>
<tr>
<td>D0H</td>
<td>PSW</td>
<td>IP3H</td>
</tr>
<tr>
<td>C8H</td>
<td>P5</td>
<td>AUXINTIF</td>
</tr>
<tr>
<td>C0H</td>
<td>P4</td>
<td>IAP_CONTR</td>
</tr>
<tr>
<td>B8H</td>
<td>IP</td>
<td>IAP_DATA</td>
</tr>
<tr>
<td>B0H</td>
<td>P3</td>
<td>IAP_ADDRH</td>
</tr>
<tr>
<td>A8H</td>
<td>IE</td>
<td>IAP_ADDRL</td>
</tr>
<tr>
<td>A0H</td>
<td>P2</td>
<td>IAP_CMD</td>
</tr>
<tr>
<td>98H</td>
<td>SCON</td>
<td>IAP_TRIG</td>
</tr>
<tr>
<td>90H</td>
<td>P1</td>
<td>IAP_CONTR</td>
</tr>
<tr>
<td>88H</td>
<td>TCON</td>
<td>BUS_SPEED</td>
</tr>
<tr>
<td>80H</td>
<td>P0</td>
<td>Reserved</td>
</tr>
<tr>
<td>FE80H</td>
<td>I2CCFG</td>
<td>Reserved</td>
</tr>
<tr>
<td>FE18H</td>
<td>P0NCS</td>
<td>Reserved</td>
</tr>
<tr>
<td>FE10H</td>
<td>P0PU</td>
<td>Reserved</td>
</tr>
<tr>
<td>FE00H</td>
<td>CKSEL</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address</th>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FE0H</td>
<td>CLKDIV</td>
<td>IRC24MCR</td>
</tr>
<tr>
<td></td>
<td>IRC32KCR</td>
<td>IRC32KCR</td>
</tr>
</tbody>
</table>

Nantong guoxin Microelectronics Co., Ltd.  
Tel: 0513-5501 2928/2929/2966  
Fax: 0513-5501 2926/2956/2947  
- 144 -
### 9.4 Series of STC8F2K64S2

<table>
<thead>
<tr>
<th></th>
<th>0/8</th>
<th>1/9</th>
<th>2/A</th>
<th>3/B</th>
<th>4/C</th>
<th>5/D</th>
<th>6/E</th>
<th>7/F</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>F8H</strong></td>
<td>P7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RSTCFG</td>
</tr>
<tr>
<td><strong>F0H</strong></td>
<td>B</td>
<td>PWMCFG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>E8H</strong></td>
<td>P6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>AUXINTIF</td>
</tr>
<tr>
<td><strong>E0H</strong></td>
<td>ACC</td>
<td>P7M1</td>
<td>P7M0</td>
<td>DPS</td>
<td>DPL1</td>
<td>DPH1</td>
<td>CMPCR1</td>
<td>CMPCR2</td>
</tr>
<tr>
<td><strong>D8H</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>D0H</strong></td>
<td>PSW</td>
<td>T4T3M</td>
<td>T4H</td>
<td>T4L</td>
<td>T3H</td>
<td>T3L</td>
<td>T2H</td>
<td>T2L</td>
</tr>
<tr>
<td><strong>C8H</strong></td>
<td>P5</td>
<td>P5M1</td>
<td>P5M0</td>
<td>P6M1</td>
<td>P6M0</td>
<td>SPSTAT</td>
<td>SPCTL</td>
<td>SPDAT</td>
</tr>
<tr>
<td><strong>C0H</strong></td>
<td>P4</td>
<td>WDT_CONTR</td>
<td>IAP_DATA</td>
<td>IAP_ADDRH</td>
<td>IAP_ADDRL</td>
<td>IAP_CMD</td>
<td>IAP_TRIG</td>
<td>IAP_CONTR</td>
</tr>
<tr>
<td><strong>B8H</strong></td>
<td>IP</td>
<td>SADEN</td>
<td>P_SW2</td>
<td>VOCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>B0H</strong></td>
<td>P3</td>
<td>P3M1</td>
<td>P3M0</td>
<td>P4M1</td>
<td>P4M0</td>
<td>IP2</td>
<td>IP2H</td>
<td>IPH</td>
</tr>
<tr>
<td><strong>A8H</strong></td>
<td>IE</td>
<td>SADDR</td>
<td>WKTC1</td>
<td>WKTC2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>A0H</strong></td>
<td>P2</td>
<td>BUS_SPEED</td>
<td>P_SW1</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>98H</strong></td>
<td>SCON</td>
<td>SBUF</td>
<td>S2CON</td>
<td>S2BUF</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td><strong>90H</strong></td>
<td>P1</td>
<td>P1M1</td>
<td>P1M0</td>
<td>P0M1</td>
<td>P0M0</td>
<td>P2M1</td>
<td>P2M0</td>
<td>AUXR2</td>
</tr>
<tr>
<td><strong>88H</strong></td>
<td>TCON</td>
<td>TMOD</td>
<td>TL0</td>
<td>TL1</td>
<td>TH0</td>
<td>TH1</td>
<td>AUXR</td>
<td>INTCLKO</td>
</tr>
<tr>
<td><strong>80H</strong></td>
<td>P0</td>
<td>SP</td>
<td>DPL</td>
<td>DPH</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0/8</th>
<th>1/9</th>
<th>2/A</th>
<th>3/B</th>
<th>4/C</th>
<th>5/D</th>
<th>6/E</th>
<th>7/F</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FE80H</strong></td>
<td>12CCFG</td>
<td>12CMSCR</td>
<td>12CMSST</td>
<td>12CSLCSR</td>
<td>12CSLLST</td>
<td>12CSLADR</td>
<td>12CTXD</td>
<td>12CRxD</td>
</tr>
<tr>
<td><strong>FE18H</strong></td>
<td>P0NCS</td>
<td>P1NCS</td>
<td>P2NCS</td>
<td>P3NCS</td>
<td>P4NCS</td>
<td>P5NCS</td>
<td>P6NCS</td>
<td>P7NCS</td>
</tr>
<tr>
<td><strong>FE10H</strong></td>
<td>P0PU</td>
<td>P1PU</td>
<td>P2PU</td>
<td>P3PU</td>
<td>P4PU</td>
<td>P5PU</td>
<td>P6PU</td>
<td>P7PU</td>
</tr>
<tr>
<td><strong>FE00H</strong></td>
<td>CKSEL</td>
<td>CLKDIV</td>
<td>IRC24MCR</td>
<td>XOSCCR</td>
<td>IRC32KCR</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 9.5 List of Special Function Registers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>description</th>
<th>addr ess</th>
<th>Bit address and symbol</th>
<th>reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>P0 port</td>
<td>80H</td>
<td>B7 B6 B5 B4 B3 B2 B1 B0</td>
<td>1111,1111</td>
</tr>
<tr>
<td>SP</td>
<td>Stack pointer</td>
<td>81H</td>
<td></td>
<td>0000,0111</td>
</tr>
<tr>
<td>DPL</td>
<td>Data pointer (low byte)</td>
<td>82H</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>DPH</td>
<td>Data pointer (high byte)</td>
<td>83H</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>S4CON</td>
<td>Serial port 4 control register</td>
<td>84H</td>
<td>S4SM0 S4ST4 S4SM2 S4REN S4TB8 S4RB8 S4TI S4RI</td>
<td>0000,0000</td>
</tr>
<tr>
<td>S4BUF</td>
<td>Serial port 4 data register</td>
<td>85H</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>PCON</td>
<td>Power control register</td>
<td>87H</td>
<td>S4SM0 S4ST4 S4SM2 S4REN S4TB8 S4RB8 S4TI S4RI</td>
<td>0000,0000</td>
</tr>
<tr>
<td>TCON</td>
<td>Timer control register</td>
<td>88H</td>
<td>TF1 TR1 TF0 TR0 IE1 IT1 IE0 IT0</td>
<td>0000,0000</td>
</tr>
<tr>
<td>TMOD</td>
<td>Timer mode register</td>
<td>89H</td>
<td>GATE C/T M1 M0 GATE C/T M1 M0</td>
<td>0000,0000</td>
</tr>
<tr>
<td>TL0</td>
<td>Timer 0 low 8 bit register</td>
<td>8AH</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>TL1</td>
<td>Timer 1 low 8 bit register</td>
<td>8BH</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>TH0</td>
<td>Timer 0 high 8 bit register</td>
<td>8CH</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>TH1</td>
<td>Timer 1 high 8 bit register</td>
<td>8DH</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>AUXR</td>
<td>Auxiliary register 1</td>
<td>8EH</td>
<td>T0x12 T1x12 UART_M0x6 T2R T2_C/T T2x12 EXTRAM S1ST2</td>
<td>0000,0000</td>
</tr>
<tr>
<td>INTCLK0</td>
<td>Interrupt and clock output control register</td>
<td>8FH</td>
<td>- EX4 EX3 EX2 - T2CLKO T1CLKO T0CLKO</td>
<td>0000,0000</td>
</tr>
<tr>
<td>P1</td>
<td>P1 port</td>
<td>90H</td>
<td></td>
<td>1111,1111</td>
</tr>
<tr>
<td>P1M1</td>
<td>P1 port configuration register 1</td>
<td>91H</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>P1M0</td>
<td>P1 port configuration register 0</td>
<td>92H</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>P0M1</td>
<td>P0 port configuration register 1</td>
<td>93H</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>P0M0</td>
<td>P0 port configuration register 0</td>
<td>94H</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>P2M1</td>
<td>P2 port configuration register 1</td>
<td>95H</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>P2M0</td>
<td>P2 port configuration register 0</td>
<td>96H</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>AUXR2</td>
<td>Auxiliary register 2</td>
<td>97H</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>SCON</td>
<td>Serial port 1 control register</td>
<td>98H</td>
<td>SM0/FE SM1 SM2 REN TB8 RB8 TI R1</td>
<td>0000,0000</td>
</tr>
<tr>
<td>SBUF</td>
<td>Serial port 1 data register</td>
<td>99H</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>S2CON</td>
<td>Serial port 2 control register</td>
<td>9AH</td>
<td>S2SM0 - S2SM2 S2REN S2TB8 S2RB8 S2TI S2RI</td>
<td>0100,0000</td>
</tr>
<tr>
<td>S2BUF</td>
<td>Serial port 2 data register</td>
<td>9BH</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>P2</td>
<td>P2 port</td>
<td>A0H</td>
<td></td>
<td>1111,1111</td>
</tr>
<tr>
<td>BUS_SPEED</td>
<td>Bus speed control register</td>
<td>A1H</td>
<td>RW_S[1:0] SPEED[1:0]</td>
<td>0000,0000</td>
</tr>
<tr>
<td>P_SW1</td>
<td>Peripheral port switching register 1</td>
<td>A2H</td>
<td>S1_S[1:0] CCP_S[1:0] S1_S[1:0]</td>
<td>0 - 0000,0000</td>
</tr>
<tr>
<td>IE</td>
<td>interrupt enable register</td>
<td>A8H</td>
<td>EA ELVD EADC ES ET1 EX1 ET0 EX0</td>
<td>0000,0000</td>
</tr>
<tr>
<td>SADDR</td>
<td>Serial port slave address register</td>
<td>A9H</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>WKTEC</td>
<td>Power down wake-up timer low byte</td>
<td>AAH</td>
<td></td>
<td>1111,1111</td>
</tr>
<tr>
<td>WKTCH</td>
<td>Power down wake-up timer high byte</td>
<td>ABH</td>
<td>WKTEC</td>
<td>0111,1111</td>
</tr>
<tr>
<td>Register</td>
<td>Description</td>
<td>Access</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------------------</td>
<td>--------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>S3CON</td>
<td>Serial port 3 control register A</td>
<td>B</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>S3BUF</td>
<td>Serial port 3 data register</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>TA</td>
<td>DPTR timing control register</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>IE2</td>
<td>Interrupt enable register 2</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>P3M1</td>
<td>P3 port configuration register 1</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>P4M1</td>
<td>P4 port configuration register 1</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>IP2</td>
<td>Interrupt priority control register 2</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>IP2H</td>
<td>High Interrupt priority control register 2</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>IPH</td>
<td>High Interrupt priority control register</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>P5M1</td>
<td>P5 port configuration register 1</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>P6M1</td>
<td>P6 port configuration register 1</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>P5M2</td>
<td>P5 port configuration register 2</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>P6M2</td>
<td>P6 port configuration register 2</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>SADEN</td>
<td>Serial port 1 slave address register</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>P_SW2</td>
<td>Peripheral port switching register 2</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>VOCTRL</td>
<td>Voltage control register</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>ADC_CONTR</td>
<td>ADC control register</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>ADC_RES</td>
<td>ADC conversion result high register</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>ADC_RESL</td>
<td>ADC conversion result low register</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>P4 port</td>
<td>H</td>
<td>1111,1111</td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>P4 port</td>
<td>H</td>
<td>1111,1111</td>
<td></td>
</tr>
<tr>
<td>WDT_CONTR</td>
<td>Watchdog control register</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>IAP_DATA</td>
<td>IAP data register</td>
<td>H</td>
<td>1111,1111</td>
<td></td>
</tr>
<tr>
<td>IAP_ADDRH</td>
<td>IAP high address register</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>IAP_ADDRL</td>
<td>IAP low address register</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>IAP_CMD</td>
<td>IAP command register</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>IAP_TRIG</td>
<td>IAP trigger register</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>IAP_CONTR</td>
<td>IAP control register</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>P5</td>
<td>P5 port</td>
<td>H</td>
<td>xx11,1111</td>
<td></td>
</tr>
<tr>
<td>P5M1</td>
<td>P5 port configuration register 1</td>
<td>H</td>
<td>xx11,1111</td>
<td></td>
</tr>
<tr>
<td>P5M2</td>
<td>P5 port configuration register 2</td>
<td>H</td>
<td>xx11,1111</td>
<td></td>
</tr>
<tr>
<td>P6M1</td>
<td>P6 port configuration register 1</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
<tr>
<td>P6M2</td>
<td>P6 port configuration register 2</td>
<td>H</td>
<td>x000,0000</td>
<td></td>
</tr>
</tbody>
</table>

Note: there is no P45~P47 in the STC8A series
<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
<th>Address</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPSTAT</td>
<td>SPI state register</td>
<td>CDH, SPIF, WCOL</td>
<td>-</td>
<td>00xxxx,xxxx</td>
</tr>
<tr>
<td>SPCTL</td>
<td>SPI control register</td>
<td>CEH, SSIG, SPEN, DORD</td>
<td>MSTR, CPOL, CPAH, SPR[1:0]</td>
<td>0000,0000</td>
</tr>
<tr>
<td>SPDAT</td>
<td>SPI data register</td>
<td>CFH</td>
<td>-</td>
<td>0000,0000</td>
</tr>
<tr>
<td>PSW</td>
<td>Program status word register</td>
<td>D0H, CY, AC</td>
<td>F0, RS1, RS0, OV, P</td>
<td>0000,0000</td>
</tr>
<tr>
<td>T4T3M</td>
<td>Timer 4/3 control register</td>
<td>D1H, T4R</td>
<td>T4x12, T4CLKO</td>
<td>T3R, T3_C/T, T3x12, T3CLKO</td>
</tr>
<tr>
<td>T4H</td>
<td>Timer 4 high byte</td>
<td>D2H</td>
<td>-</td>
<td>0000,0000</td>
</tr>
<tr>
<td>T4L</td>
<td>Timer 4 low byte</td>
<td>D3H</td>
<td>-</td>
<td>0000,0000</td>
</tr>
<tr>
<td>T3H</td>
<td>Timer 3 high byte</td>
<td>D4H</td>
<td>-</td>
<td>0000,0000</td>
</tr>
<tr>
<td>T3L</td>
<td>Timer 3 low byte</td>
<td>D5H</td>
<td>-</td>
<td>0000,0000</td>
</tr>
<tr>
<td>T2H</td>
<td>Timer 2 high byte</td>
<td>D6H</td>
<td>-</td>
<td>0000,0000</td>
</tr>
<tr>
<td>T2L</td>
<td>Timer 2 low byte</td>
<td>D7H</td>
<td>-</td>
<td>0000,0000</td>
</tr>
<tr>
<td>CCON</td>
<td>PCA control register</td>
<td>DBH, CF, CR</td>
<td>-</td>
<td>0000,0000</td>
</tr>
<tr>
<td>CMOD</td>
<td>PCA mode control register</td>
<td>D9H, CIDL</td>
<td>-</td>
<td>CPS[2:0], ECF</td>
</tr>
<tr>
<td>CCAPM0</td>
<td>PCA module0 mode control register</td>
<td>DAH, -</td>
<td>ECOM0, CCAP0, CCAPN0, MAT0, TOG0, PWM0, ECCF0</td>
<td>0000,0000</td>
</tr>
<tr>
<td>CCAPM1</td>
<td>PCA module1 mode control register</td>
<td>DBH, -</td>
<td>ECOM1, CCAP1, CCAPN1, MAT1, TOG1, PWM1, ECCF1</td>
<td>0000,0000</td>
</tr>
<tr>
<td>CCAPM2</td>
<td>PCA module2 mode control register</td>
<td>DCH, -</td>
<td>ECOM2, CCAP2, CCAPN2, MAT2, TOG2, PWM2, ECCF2</td>
<td>0000,0000</td>
</tr>
<tr>
<td>CCAPM3</td>
<td>PCA module3 mode control register</td>
<td>DDH, -</td>
<td>ECOM3, CCAP3, CCAPN3, MAT3, TOG3, PWM3, ECCF3</td>
<td>0000,0000</td>
</tr>
<tr>
<td>ACC</td>
<td>Accumulator</td>
<td>E0H</td>
<td>-</td>
<td>0000,0000</td>
</tr>
<tr>
<td>P7M1</td>
<td>P7 port configuration register 1</td>
<td>E1H</td>
<td>-</td>
<td>0000,0000</td>
</tr>
<tr>
<td>P7M0</td>
<td>P7 port configuration register 0</td>
<td>E2H</td>
<td>-</td>
<td>0000,0000</td>
</tr>
<tr>
<td>DPS</td>
<td>Pointer selector</td>
<td>E3H, ID1, ID0</td>
<td>TSL, AU1, AU0, SEL</td>
<td>0000,0000</td>
</tr>
<tr>
<td>DPL1</td>
<td>Second sets of data pointers (low byte)</td>
<td>E4H</td>
<td>-</td>
<td>0000,0000</td>
</tr>
<tr>
<td>DPH1</td>
<td>Second sets of data pointers (high byte)</td>
<td>E5H</td>
<td>-</td>
<td>0000,0000</td>
</tr>
<tr>
<td>CMPCR1</td>
<td>Comparator control register 1</td>
<td>E6H, CMPEN, CMPIF, PIE, NIE, PIS, NIS, CMPOE, CMPRES</td>
<td>LCDTY[5:0]</td>
<td>0000,0000</td>
</tr>
<tr>
<td>CMPCR2</td>
<td>Comparator control register 2</td>
<td>E7H, INVCMP0, DISFLT, LCDTY[5:0]</td>
<td>-</td>
<td>0000,0000</td>
</tr>
<tr>
<td>P6</td>
<td>P6 port</td>
<td>EBH</td>
<td>-</td>
<td>1111,1111</td>
</tr>
<tr>
<td>CL</td>
<td>PCA counter low byte</td>
<td>E9H</td>
<td>-</td>
<td>0000,0000</td>
</tr>
<tr>
<td>CCA0L</td>
<td>PCA module0 low byte</td>
<td>EAH</td>
<td>-</td>
<td>0000,0000</td>
</tr>
<tr>
<td>CCA1L</td>
<td>PCA module1 low byte</td>
<td>EBH</td>
<td>-</td>
<td>0000,0000</td>
</tr>
<tr>
<td>CCA2L</td>
<td>PCA module2 low byte</td>
<td>ECH</td>
<td>-</td>
<td>0000,0000</td>
</tr>
<tr>
<td>CCA3L</td>
<td>PCA module3 low byte</td>
<td>EDH</td>
<td>-</td>
<td>0000,0000</td>
</tr>
<tr>
<td>IP1H</td>
<td>High interrupt priority - control register</td>
<td>EEH</td>
<td>-</td>
<td>PS4H, PS3H</td>
</tr>
<tr>
<td>AUXINTIF</td>
<td>Extended external interrupt flag register</td>
<td>EFH</td>
<td>INT4IF, INT3IF, INT2IF, T4IF, T3IF, T2IF</td>
<td>0000,0000</td>
</tr>
</tbody>
</table>
The following special function registers are expanded SFR and logical address is located in the XDATA area. Before accessing, the P_SW2 (BAH) register's highest position (EAXFR) is placed to 1. Then MOVX A, @DPTR and MOVX @DPTR, A instruction are used to access.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Address</th>
<th>Bit of address and symbol</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>B7</td>
<td>B6</td>
</tr>
<tr>
<td>PWMCH</td>
<td>PWM counter high byte</td>
<td>FFF0H</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>PMWCL</td>
<td>PWM counter low byte</td>
<td>FFF1H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMWCKS</td>
<td>PWM clock selection</td>
<td>FFF2H</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TADCPH</td>
<td>触发 ADC count value high byte</td>
<td>FFF3H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TADCPL</td>
<td>触发 ADC count value byte</td>
<td>FFF4H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM0T1H</td>
<td>PWM0T1 count value high byte</td>
<td>FFF0H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM0T1L</td>
<td>PWM0T1 count value low byte</td>
<td>FF01H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM0T2H</td>
<td>PWM0T2 count value high byte</td>
<td>FF02H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM0T2L</td>
<td>PWM0T2 count value low byte</td>
<td>FF03H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM0CR</td>
<td>PWM0 control register</td>
<td>FF04H</td>
<td>ENC0</td>
<td>C0IN</td>
</tr>
<tr>
<td>PWM0HLD</td>
<td>PWM0 level hold control register</td>
<td>FF05H</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM1T1H</td>
<td>PWM1T1 count value high byte</td>
<td>FF10H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM1T1L</td>
<td>PWM1T1 count value low byte</td>
<td>FF11H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM1T2H</td>
<td>PWM1T2 count value high byte</td>
<td>FF12H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM1T2L</td>
<td>PWM1T2 count value low byte</td>
<td>FF13H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM1CR</td>
<td>PWM1 control register</td>
<td>FF14H</td>
<td>ENC1O</td>
<td>C1IN</td>
</tr>
<tr>
<td>Function</td>
<td>Address</td>
<td>Description</td>
<td>Bit 1</td>
<td>Bit 0</td>
</tr>
<tr>
<td>-----------</td>
<td>----------</td>
<td>-----------------------------------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>PWM1HLD</td>
<td>FF15H</td>
<td>PWM1 level hold control register</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM2T1H</td>
<td>FF20H</td>
<td>PWM2T1 count value high byte</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM2T1L</td>
<td>FF21H</td>
<td>PWM2T1 count value low byte</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM2T2H</td>
<td>FF22H</td>
<td>PWM2T2 count value high byte</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM2T2L</td>
<td>FF23H</td>
<td>PWM2T2 count value low byte</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM2CR</td>
<td>FF24H</td>
<td>PWM2 control register</td>
<td>ENC2O</td>
<td>C2IN1</td>
</tr>
<tr>
<td>PWM2HLD</td>
<td>FF25H</td>
<td>PWM2 level hold control register</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM3T1H</td>
<td>FF30H</td>
<td>PWM3T1 count value high byte</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM3T1L</td>
<td>FF31H</td>
<td>PWM3T1 count value low byte</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM3T2H</td>
<td>FF32H</td>
<td>PWM3T2 count value high byte</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM3T2L</td>
<td>FF33H</td>
<td>PWM3T2 count value low byte</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM3CR</td>
<td>FF34H</td>
<td>PWM3 control register</td>
<td>ENC3O</td>
<td>C3IN1</td>
</tr>
<tr>
<td>PWM3HLD</td>
<td>FF35H</td>
<td>PWM3 level hold control register</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM4T1H</td>
<td>FF40H</td>
<td>PWM4T1 count value high byte</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM4T1L</td>
<td>FF41H</td>
<td>PWM4T1 count value low byte</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM4T2H</td>
<td>FF42H</td>
<td>PWM4T2 count value high byte</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM4T2L</td>
<td>FF43H</td>
<td>PWM4T2 count value low byte</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM4CR</td>
<td>FF44H</td>
<td>PWM4 control register</td>
<td>ENC4O</td>
<td>C4IN1</td>
</tr>
<tr>
<td>PWM4HLD</td>
<td>FF45H</td>
<td>PWM4 level hold control register</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM5T1H</td>
<td>FF50H</td>
<td>PWM5T1 count value high byte</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM5T1L</td>
<td>FF51H</td>
<td>PWM5T1 count value low byte</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM5T2H</td>
<td>FF52H</td>
<td>PWM5T2 count value high byte</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM5T2L</td>
<td>FF53H</td>
<td>PWM5T2 count value low byte</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM5CR</td>
<td>FF54H</td>
<td>PWM5 control register</td>
<td>ENC5O</td>
<td>C5IN1</td>
</tr>
<tr>
<td>PWM5HLD</td>
<td>FF55H</td>
<td>PWM5 level hold control register</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM6T1H</td>
<td>FF60H</td>
<td>PWM6T1 count value high byte</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM6T1L</td>
<td>FF61H</td>
<td>PWM6T1 count value low byte</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM6T2H</td>
<td>FF62H</td>
<td>PWM6T2 count value high byte</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM6T2L</td>
<td>FF63H</td>
<td>PWM6T2 count value low byte</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM6CR</td>
<td>FF64H</td>
<td>PWM6 control register</td>
<td>ENC6O</td>
<td>C6IN1</td>
</tr>
<tr>
<td>PWM6HLD</td>
<td>FF65H</td>
<td>PWM6 level hold control register</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM7T1H</td>
<td>FF70H</td>
<td>PWM7T1 count value high byte</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM7T1L</td>
<td>FF71H</td>
<td>PWM7T1 count value low byte</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM7T2H</td>
<td>FF72H</td>
<td>PWM7T2 count value high byte</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM7T2L</td>
<td>FF73H</td>
<td>PWM7T2 count value low byte</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM7CR</td>
<td>FF74H</td>
<td>PWM7 control register</td>
<td>ENC7O</td>
<td>C7IN1</td>
</tr>
<tr>
<td>PWM7HLD</td>
<td>FF75H</td>
<td>PWM7 level hold</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Nantong guoxin Microelectronics Co., Ltd.  
Tel: 0513-5501 2928/2929/2966  
Fax: 0513-5501 2926/2956/2947
<table>
<thead>
<tr>
<th>Register Name</th>
<th>Description</th>
<th>Address</th>
<th>Value</th>
<th>Hardware Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2CCFG</td>
<td>I2C configuration register</td>
<td>FE80H</td>
<td>0000_0000</td>
<td>ENI2C, MSS, MSSPEED[6:1]</td>
</tr>
<tr>
<td>I2CMSCR</td>
<td>I2C host control register</td>
<td>FE81H</td>
<td>0000_0000</td>
<td>EMSI, MSSL, MSSL[6:1]</td>
</tr>
<tr>
<td>I2CMSST</td>
<td>I2C host state register</td>
<td>FE82H</td>
<td>0000_0000</td>
<td>MSBUSY, MSIF, MSACKI, MSACKO</td>
</tr>
<tr>
<td>I2CSLCR</td>
<td>I2C slave control register</td>
<td>FE83H</td>
<td>0000_0000</td>
<td>ESTAI, ERXI, ETXI, ESTOI, SLRST</td>
</tr>
<tr>
<td>I2CSLST</td>
<td>I2C slave state register</td>
<td>FE84H</td>
<td>0000_0000</td>
<td>SLBUSY, STAIF, RXIF, TXIF, STOIF, TXING, SLACKI, SLACKO</td>
</tr>
<tr>
<td>I2CSLADR</td>
<td>I2C slave address register</td>
<td>FE85H</td>
<td>0000_0000</td>
<td>SLADR[6:0], MA</td>
</tr>
<tr>
<td>I2CTXD</td>
<td>I2C data transmission register</td>
<td>FE86H</td>
<td>0000_0000</td>
<td></td>
</tr>
<tr>
<td>I2CRXD</td>
<td>I2C data receiving register</td>
<td>FE87H</td>
<td>0000_0000</td>
<td></td>
</tr>
<tr>
<td>I2CMSAUX</td>
<td>I2C host auxiliary control register</td>
<td>FE88H</td>
<td>0000_0000</td>
<td>WDTA</td>
</tr>
<tr>
<td>P0PU</td>
<td>P0 port pull-up resistance control register</td>
<td>FE10H</td>
<td>0000_0000</td>
<td></td>
</tr>
<tr>
<td>P1PU</td>
<td>P1 port pull-up resistance control register</td>
<td>FE11H</td>
<td>0000_0000</td>
<td></td>
</tr>
<tr>
<td>P2PU</td>
<td>P2 port pull-up resistance control register</td>
<td>FE12H</td>
<td>0000_0000</td>
<td></td>
</tr>
<tr>
<td>P3PU</td>
<td>P3 port pull-up resistance control register</td>
<td>FE13H</td>
<td>0000_0000</td>
<td></td>
</tr>
<tr>
<td>P4PU</td>
<td>P4 port pull-up resistance control register</td>
<td>FE14H</td>
<td>0000_0000</td>
<td></td>
</tr>
<tr>
<td>P5PU</td>
<td>P5 port pull-up resistance control register</td>
<td>FE15H</td>
<td>0000_0000</td>
<td></td>
</tr>
<tr>
<td>P6PU</td>
<td>P6 port pull-up resistance control register</td>
<td>FE16H</td>
<td>0000_0000</td>
<td></td>
</tr>
<tr>
<td>P7PU</td>
<td>P7 port pull-up resistance control register</td>
<td>FE17H</td>
<td>0000_0000</td>
<td></td>
</tr>
<tr>
<td>P0NCS</td>
<td>P0 port Schmidt trigger control register</td>
<td>FE18H</td>
<td>0000_0000</td>
<td></td>
</tr>
<tr>
<td>P1NCS</td>
<td>P1 port Schmidt trigger control register</td>
<td>FE19H</td>
<td>0000_0000</td>
<td></td>
</tr>
<tr>
<td>P2NCS</td>
<td>P2 port Schmidt trigger control register</td>
<td>FE1AH</td>
<td>0000_0000</td>
<td></td>
</tr>
<tr>
<td>P3NCS</td>
<td>P3 port Schmidt trigger control register</td>
<td>FE1BH</td>
<td>0000_0000</td>
<td></td>
</tr>
<tr>
<td>P4NCS</td>
<td>P4 port Schmidt trigger control register</td>
<td>FE1CH</td>
<td>0000_0000</td>
<td></td>
</tr>
<tr>
<td>P5NCS</td>
<td>P5 port Schmidt trigger control register</td>
<td>FE1DH</td>
<td>0000_0000</td>
<td></td>
</tr>
<tr>
<td>P6NCS</td>
<td>P6 port Schmidt trigger control register</td>
<td>FE1EH</td>
<td>0000_0000</td>
<td></td>
</tr>
<tr>
<td>P7NCS</td>
<td>P7 port Schmidt trigger control register</td>
<td>FE1FH</td>
<td>0000_0000</td>
<td></td>
</tr>
<tr>
<td>Clock register</td>
<td>Description</td>
<td>Address</td>
<td>MCLKO[3:0]</td>
<td>MCLKO_S</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------------------------------</td>
<td>---------</td>
<td>------------</td>
<td>----------</td>
</tr>
<tr>
<td>CKSEL</td>
<td>Clock selection register</td>
<td>FE00H</td>
<td>MCLKODIV[3:0]</td>
<td>-</td>
</tr>
<tr>
<td>CLKDIV</td>
<td>Clock frequency division register</td>
<td>FE01H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IRC24MCR</td>
<td>Internal 24M oscillator control register</td>
<td>FE02H</td>
<td>ENIRC24M</td>
<td>-</td>
</tr>
<tr>
<td>XOSCCR</td>
<td>External oscillator control register</td>
<td>FE03H</td>
<td>ENXOSC</td>
<td>XITYPE</td>
</tr>
<tr>
<td>IRC32KCR</td>
<td>Internal 32K oscillator control register</td>
<td>FE04H</td>
<td>ENIRC32K</td>
<td>-</td>
</tr>
</tbody>
</table>
10 I/O Ports

There are not more than 59 I/O ports in STC8 microcontrollers family. There are 4 modes for all GPIOs: quasi bidirectional or weak pull-up mode (standard 8051 output mode), push-pull output / strong pull-up mode, high-impedance input mode (where current can neither flow in nor out), open drain mode. It is easy to configure the I/O mode by software.

10.1 I/O port related registers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Address</th>
<th>Bit Address and Symbol</th>
<th>Value after reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>Port 0</td>
<td>80H</td>
<td>B7 B6 B5 B4 B3 B2 B1 B0</td>
<td>1111,1111</td>
</tr>
<tr>
<td>P1</td>
<td>Port 1</td>
<td>90H</td>
<td>1111,1111</td>
<td></td>
</tr>
<tr>
<td>P2</td>
<td>Port 2</td>
<td>A0H</td>
<td>1111,1111</td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>Port 3</td>
<td>B0H</td>
<td>1111,1111</td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>Port 4</td>
<td>C0H</td>
<td>1111,1111</td>
<td></td>
</tr>
<tr>
<td>P5</td>
<td>Port 5</td>
<td>C8H</td>
<td>- - xxx1,1111</td>
<td></td>
</tr>
<tr>
<td>P6</td>
<td>Port 6</td>
<td>E8H</td>
<td>1111,1111</td>
<td></td>
</tr>
<tr>
<td>P7</td>
<td>Port 7</td>
<td>F8H</td>
<td>1111,1111</td>
<td></td>
</tr>
<tr>
<td>P0M1</td>
<td>Port 0 mode register 1</td>
<td>93H</td>
<td>0000,0000</td>
<td></td>
</tr>
<tr>
<td>P0M0</td>
<td>Port 0 mode register 0</td>
<td>94H</td>
<td>0000,0000</td>
<td></td>
</tr>
<tr>
<td>P1M1</td>
<td>Port 1 mode register 1</td>
<td>91H</td>
<td>0000,0000</td>
<td></td>
</tr>
<tr>
<td>P1M0</td>
<td>Port 1 mode register 0</td>
<td>92H</td>
<td>0000,0000</td>
<td></td>
</tr>
<tr>
<td>P2M1</td>
<td>Port 2 mode register 1</td>
<td>B1H</td>
<td>n000,0000</td>
<td></td>
</tr>
<tr>
<td>P2M0</td>
<td>Port 2 mode register 0</td>
<td>B2H</td>
<td>n000,0000</td>
<td></td>
</tr>
<tr>
<td>P3M1</td>
<td>Port 3 mode register 1</td>
<td>B3H</td>
<td>0000,0000</td>
<td></td>
</tr>
<tr>
<td>P3M0</td>
<td>Port 3 mode register 0</td>
<td>B4H</td>
<td>0000,0000</td>
<td></td>
</tr>
<tr>
<td>P4M1</td>
<td>Port 4 mode register 1</td>
<td>C9H</td>
<td>- - xxx1,1111</td>
<td></td>
</tr>
<tr>
<td>P4M0</td>
<td>Port 4 mode register 0</td>
<td>CAH</td>
<td>- - xxx1,1111</td>
<td></td>
</tr>
<tr>
<td>P5M1</td>
<td>Port 5 mode register 1</td>
<td>E1H</td>
<td>0000,0000</td>
<td></td>
</tr>
<tr>
<td>P5M0</td>
<td>Port 5 mode register 0</td>
<td>E2H</td>
<td>0000,0000</td>
<td></td>
</tr>
<tr>
<td>P6M1</td>
<td>Port 6 mode register 1</td>
<td>FE10H</td>
<td>0000,0000</td>
<td></td>
</tr>
<tr>
<td>P6M0</td>
<td>Port 6 mode register 0</td>
<td>FE11H</td>
<td>0000,0000</td>
<td></td>
</tr>
<tr>
<td>P7M1</td>
<td>Port 7 mode register 1</td>
<td>E1H</td>
<td>0000,0000</td>
<td></td>
</tr>
<tr>
<td>P7M0</td>
<td>Port 7 mode register 0</td>
<td>E2H</td>
<td>0000,0000</td>
<td></td>
</tr>
<tr>
<td>P0PU</td>
<td>Port 0 register which is control the pull-up resistor</td>
<td>FE10H</td>
<td>0000,0000</td>
<td></td>
</tr>
<tr>
<td>P1PU</td>
<td>Port 1 register which is control</td>
<td>FE11H</td>
<td>0000,0000</td>
<td></td>
</tr>
</tbody>
</table>
the pull-up resistor

<table>
<thead>
<tr>
<th>Port</th>
<th>Register which is control</th>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2PU</td>
<td>Port2</td>
<td>FE12H</td>
<td>0000_0000</td>
</tr>
<tr>
<td>P3PU</td>
<td>Port 3</td>
<td>FE13H</td>
<td>0000_0000</td>
</tr>
<tr>
<td>P4PU</td>
<td>Port 4</td>
<td>FE14H</td>
<td>0000_0000</td>
</tr>
<tr>
<td>P5PU</td>
<td>Port 5</td>
<td>FE15H</td>
<td>0000_0000</td>
</tr>
<tr>
<td>P6PU</td>
<td>Port 6</td>
<td>FE16H</td>
<td>0000_0000</td>
</tr>
<tr>
<td>P7PU</td>
<td>Port 7</td>
<td>FE17H</td>
<td>0000_0000</td>
</tr>
<tr>
<td>P0NCS</td>
<td>Port 0 register which is control the Schmidt trigger</td>
<td>FE18H</td>
<td>0000_0000</td>
</tr>
<tr>
<td>P1NCS</td>
<td>Port 1 register which is control the Schmidt trigger</td>
<td>FE19H</td>
<td>0000_0000</td>
</tr>
<tr>
<td>P2NCS</td>
<td>Port 2 register which is control the Schmidt trigger</td>
<td>FE1AH</td>
<td>0000_0000</td>
</tr>
<tr>
<td>P3NCS</td>
<td>Port 3 register which is control the Schmidt trigger</td>
<td>FE1BH</td>
<td>0000_0000</td>
</tr>
<tr>
<td>P4NCS</td>
<td>Port 4 register which is control the Schmidt trigger</td>
<td>FE1CH</td>
<td>0000_0000</td>
</tr>
<tr>
<td>P5NCS</td>
<td>Port 5 register which is control the Schmidt trigger</td>
<td>FE1DH</td>
<td>0000_0000</td>
</tr>
<tr>
<td>P6NCS</td>
<td>Port 6 register which is control the Schmidt trigger</td>
<td>FE1EH</td>
<td>0000_0000</td>
</tr>
<tr>
<td>P7NCS</td>
<td>Port 7 register which is control the Schmidt trigger</td>
<td>FE1FH</td>
<td>0000_0000</td>
</tr>
</tbody>
</table>

Registers related to date of the ports

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>80H</td>
<td>P0.7</td>
<td>P0.6</td>
<td>P0.5</td>
<td>P0.4</td>
<td>P0.3</td>
<td>P0.2</td>
<td>P0.1</td>
<td>P0.0</td>
</tr>
<tr>
<td>P1</td>
<td>90H</td>
<td>P1.7</td>
<td>P1.6</td>
<td>P1.5</td>
<td>P1.4</td>
<td>P1.3</td>
<td>P1.2</td>
<td>P1.1</td>
<td>P1.0</td>
</tr>
<tr>
<td>P2</td>
<td>A0H</td>
<td>P2.7</td>
<td>P2.6</td>
<td>P2.5</td>
<td>P2.4</td>
<td>P2.3</td>
<td>P2.2</td>
<td>P2.1</td>
<td>P2.0</td>
</tr>
<tr>
<td>P3</td>
<td>B0H</td>
<td>P3.7</td>
<td>P3.6</td>
<td>P3.5</td>
<td>P3.4</td>
<td>P3.3</td>
<td>P3.2</td>
<td>P3.1</td>
<td>P3.0</td>
</tr>
<tr>
<td>P4</td>
<td>C0H</td>
<td>P4.7</td>
<td>P4.6</td>
<td>P4.5</td>
<td>P4.4</td>
<td>P4.3</td>
<td>P4.2</td>
<td>P4.1</td>
<td>P4.0</td>
</tr>
<tr>
<td>P5</td>
<td>C8H</td>
<td>-</td>
<td>-</td>
<td>P5.5</td>
<td>P5.4</td>
<td>P5.3</td>
<td>P5.2</td>
<td>P5.1</td>
<td>P5.0</td>
</tr>
<tr>
<td>P6</td>
<td>E8H</td>
<td>P6.7</td>
<td>P6.6</td>
<td>P6.5</td>
<td>P6.4</td>
<td>P6.3</td>
<td>P6.2</td>
<td>P6.1</td>
<td>P6.0</td>
</tr>
<tr>
<td>P7</td>
<td>F8H</td>
<td>P7.7</td>
<td>P7.6</td>
<td>P7.5</td>
<td>P7.4</td>
<td>P7.3</td>
<td>P7.2</td>
<td>P7.1</td>
<td>P7.0</td>
</tr>
</tbody>
</table>

Write and read the status of the port
Write 0: write low level to the buffer of ports
Write 1: write high level to the buffer of ports
Read: read the level of points
Registers which configure the mode of ports

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0M0</td>
<td>94H</td>
</tr>
<tr>
<td>P0M1</td>
<td>93H</td>
</tr>
<tr>
<td>P1M0</td>
<td>92H</td>
</tr>
<tr>
<td>P1M1</td>
<td>91H</td>
</tr>
<tr>
<td>P2M0</td>
<td>96H</td>
</tr>
<tr>
<td>P2M1</td>
<td>95H</td>
</tr>
<tr>
<td>P3M0</td>
<td>B2H</td>
</tr>
<tr>
<td>P3M1</td>
<td>B1H</td>
</tr>
<tr>
<td>P4M0</td>
<td>B4H</td>
</tr>
<tr>
<td>P4M1</td>
<td>B3H</td>
</tr>
<tr>
<td>P5M0</td>
<td>CAH</td>
</tr>
<tr>
<td>P5M1</td>
<td>C9H</td>
</tr>
<tr>
<td>P6M0</td>
<td>CCH</td>
</tr>
<tr>
<td>P6M1</td>
<td>CBH</td>
</tr>
<tr>
<td>P7M0</td>
<td>E2H</td>
</tr>
<tr>
<td>P7M1</td>
<td>E1H</td>
</tr>
</tbody>
</table>

The mode of Pn.x

- 0 0: quasi bidirectional
- 0 1: push-pull output
- 1 0: high-impedance
- 1 1: open drain mode

Register control the pull-up resistor in port

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0PU</td>
<td>FE10H</td>
</tr>
<tr>
<td>P1PU</td>
<td>FE11H</td>
</tr>
<tr>
<td>P2PU</td>
<td>FE12H</td>
</tr>
<tr>
<td>P3PU</td>
<td>FE13H</td>
</tr>
<tr>
<td>P4PU</td>
<td>FE14H</td>
</tr>
<tr>
<td>P5PU</td>
<td>FE15H</td>
</tr>
<tr>
<td>P6PU</td>
<td>FE16H</td>
</tr>
<tr>
<td>P7PU</td>
<td>FE17H</td>
</tr>
</tbody>
</table>

Control bit for internal 3.7K pull-up resistor

- 0: forbid internal 3.7K pull-up resistor (the measured is about 4.2K)
- 1: enable internal 3.7K pull-up resistor (the measured is about 4.2K)
register which is control the Schmidt trigger in port

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0NCS</td>
<td>FE18H</td>
</tr>
<tr>
<td>P1NCS</td>
<td>FE19H</td>
</tr>
<tr>
<td>P2NCS</td>
<td>FE1AH</td>
</tr>
<tr>
<td>P3NCS</td>
<td>FE1BH</td>
</tr>
<tr>
<td>P4NCS</td>
<td>FE1CH</td>
</tr>
<tr>
<td>P5NCS</td>
<td>FE1DH</td>
</tr>
<tr>
<td>P6NCS</td>
<td>FE1EH</td>
</tr>
<tr>
<td>P7NCS</td>
<td>FE1FH</td>
</tr>
</tbody>
</table>

Control bit for Schmidt trigger in port

0 : enable the function of Schmidt trigger in port
1 : forbid the function of Schmidt trigger in port

<table>
<thead>
<tr>
<th>VCC=5.0V</th>
<th>minimum value</th>
<th>Maxim value</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal I/O input high</td>
<td>2.2V</td>
<td>-</td>
<td>Open Schmidt trigger</td>
</tr>
<tr>
<td>Normal I/O input low</td>
<td>-</td>
<td>1.4V</td>
<td></td>
</tr>
<tr>
<td>Normal I/O input high</td>
<td>1.6V</td>
<td>-</td>
<td>Close Schmidt trigger</td>
</tr>
<tr>
<td>Normal I/O input low</td>
<td>-</td>
<td>1.5V</td>
<td></td>
</tr>
<tr>
<td>Reset pin input high</td>
<td>2.2V</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Reset pin input low</td>
<td>-</td>
<td>1.8V</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VCC=3.3V</th>
<th>Minimum value</th>
<th>Maxim value</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal I/O input high</td>
<td>1.6V</td>
<td>-</td>
<td>Open Schmidt trigger</td>
</tr>
<tr>
<td>Normal I/O input low</td>
<td>-</td>
<td>1.0V</td>
<td></td>
</tr>
<tr>
<td>Normal I/O input high</td>
<td>1.2V</td>
<td>1.1V</td>
<td>Close Schmidt trigger</td>
</tr>
<tr>
<td>Normal I/O input low</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset pin input high</td>
<td>1.7V</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Reset pin input low</td>
<td>-</td>
<td>1.3V</td>
<td></td>
</tr>
</tbody>
</table>

### 10.2 I/O Ports Configurations

Two registers are used to configure each I/O mode.

Taking Port 0 as an example, two registers, P0M0 and P0M1, are used to configure Port 0, as shown in the following figure:
The combination of bit 0 of P0M0 and bit 0 of P0M1 is used to configure the mode of P0.0.
The combination of bit 1 of P0M0 and bit 1 of P0M1 is used to configure the mode of P0.1.
All other I/O lines configurations are similar.

The combination of PnM0 and PnM1 to configure the I/O ports mode is as following.

<table>
<thead>
<tr>
<th>PnM1</th>
<th>PnM0</th>
<th>I/O ports Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Quasi bidirectional (traditional 8051 I/O port, weak pull-up)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sink Current up to 20mA, Pull-up Current is 270~150μA</td>
</tr>
<tr>
<td>(manufacturing error may be exist)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Push-pull output (strong pull-up output, current can be up to 20mA, resistors should be used to restrict current)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>high-impedance (where current can neither flow in nor out)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Open Drain mode. The internal pull-up resistors are disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The open drain mode can be used for both external status reading and output high or low. To read the external state correctly or output high level, the external pull-up resistors should be connected, otherwise the external state can not be read and the high level can not be output.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Note: n = 0,1,2,3,4,5,6,7

Note:

Any I/O port line can tolerate 20mA of sink current in weak pull-up mode (quasi-bidirectional mode) or strong push-pull output mode or open drain mode, and can output 20mA pull current in the strong push-pull output mode. Current limiting resistors should be connected in all I/O mode above, such as 1KΩ, 560Ω, 472Ω, etc. The entire chip operating current is recommended not to exceed 90mA, that is, the current flow in from the VCC should not exceed 90mA, the current flow out from the GND should not exceed 90mA, the overall inflow or outflow current is advised not to exceed 90mA.
10.3 I/O ports structure

10.3.1 Quasi-Bidirectional I/O (weak pull-up)

A quasi bidirectional port can be used as an input and output functions without the need to reconfigure the port. This is because the drive capability is weak when the port outputs a logic high level, allowing external devices to pull it low. When the pin outputs low, it's strong driving capability and able to sink a considerable current. There are three pull-up transistors in the quasi-bidirectional output to suit different needs.

One of the three pull-up transistors, called a "weak pull-up", is turned on when the port register is logic "1" and the pin itself is logic "1". This pull-up transistor provides the basic drive current to make the quasi-bidirectional port output logic "1". If one of the pin outputs logic "1" and the external device pulls it low, the weak pull-up transistor is off and the "very weak pull-up" maintains on. To pull the pin low, the external device must have sufficient sink capability to make the voltage on the pin drop below the threshold voltage. For a 5V microcontroller, the current of "weak pull-up" transistor is about 250uA; for a 3.3V microcontroller, the current of "weak pull-up" transistor is about 150uA.

The second pull-up transistor, called "very weak pull-up", turns on when the port latch is "1". When the pin is not connected, this very weak pull-up source produces a weak pull-up current that pulls the pin high. For a 5V microcontroller, the current of "weak pull-up" transistor is about 18uA; for 3.3V microcontrollers, the current of "weak pull-up" transistor is about 5uA.

The third pull-up transistor is called "strong pull-up". This pull-up transistor is used to speed up the low-to-high transition for quasi-bidirectional port pin when the port latch changes from logic "0" to logic "1". When this occurs, the strong pull-up transistor keeps on for about two clocks to quickly pull the pin high.

Quasi-bidirectional port (weak pull-up) has a Schmidt trigger and an interference suppression circuit. To read the correct external state, quasi-bidirectional port (weak pull-up) should latch to ‘1’ before reading.

The structure of quasi-bidirectional port (weak pull-up) output is shown below:

![Port latch data](image)

10.3.2 Push-Pull Output

The configuration of the strong push-pull output mode is the same as the pull-down configuration of the open-drain output mode and quasi-bidirectional mode. However, the push-pull output mode can provide a sustained strong pull-up when the latch is logic "1". Push-pull mode is generally used when more drive current is required.
10.3.3 High-Impedance

The current can neither flow in nor flow out.
The input port has a Schmidt trigger input and an interference suppression circuit.
The structure of high-impedance input pin configuration is shown below:

10.3.4 Open-Drain Output

The open-drain mode can be used for both reading external status and outputing high or low level. To read the external state correctly or output a high level, the external pull-up resistor should be connected.

The open-drain output configuration turns off all pull-up transistors when the port latch is logic "0". There must be an external pull-up in this configuration when the port outputs a logic high, typically the port pin is externally connected to VCC through a resistor. An open-drain I/O port pin can read the external state if the external pull-up resistor is connected. Here, the open-drain mode I/O port pin can be used as input mode. The pull-down in this way is the same as quasi-bidirectional mode.

The open drain port has a Schmidt trigger input and an interference suppression circuit.

The structure of output port configuration is shown below:
10.4 Instructions about special I / O ports

10.4.1 P2.0 / RSTCV

The initial level after powering on the STC8 series, P2.0 can be set by following hardware options in the ISP download software.

![P2.0 Setup Options](image)

Notions: when the operating voltage of the MCU is less than 1.6V, the P2.0 out level is high, only when the operating voltage of the MCU rising above 1.6V, P2.0 will output the level which set by user’s hardware option.

10.4.2 I / O ports related to PWM

All the I/O ports of the STC8 series have a week pull-up mode after reset when power-on. The users can configure open-drain mode on PWM related I/O ports through the following hardware options in the ISP download software.

![PWM Port Configuration](image)

The I/O ports related to PWM on STC8 series are P1.0~P1.7, P2.0~P2.7, P6.0~P6.7.

10.5 Sample programs

10.5.1 Mode configure of the ports

Assembly codes

<table>
<thead>
<tr>
<th>Port</th>
<th>Mode</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0M0</td>
<td>DATA</td>
<td>094H</td>
</tr>
<tr>
<td>P0M1</td>
<td>DATA</td>
<td>093H</td>
</tr>
<tr>
<td>P1M0</td>
<td>DATA</td>
<td>092H</td>
</tr>
<tr>
<td>P1M1</td>
<td>DATA</td>
<td>091H</td>
</tr>
<tr>
<td>P2M0</td>
<td>DATA</td>
<td>096H</td>
</tr>
<tr>
<td>P2M1</td>
<td>DATA</td>
<td>095H</td>
</tr>
<tr>
<td>P3M0</td>
<td>DATA</td>
<td>0B2H</td>
</tr>
<tr>
<td>P3M1</td>
<td>DATA</td>
<td>0B1H</td>
</tr>
<tr>
<td>P4M0</td>
<td>DATA</td>
<td>0B4H</td>
</tr>
<tr>
<td>P4M1</td>
<td>DATA</td>
<td>0B3H</td>
</tr>
<tr>
<td>P5M0</td>
<td>DATA</td>
<td>0CAH</td>
</tr>
<tr>
<td>P5M1</td>
<td>DATA</td>
<td>0C9H</td>
</tr>
<tr>
<td>P6M0</td>
<td>DATA</td>
<td>0CCH</td>
</tr>
<tr>
<td>P6M1</td>
<td>DATA</td>
<td>0CBH</td>
</tr>
<tr>
<td>P7M0</td>
<td>DATA</td>
<td>0E2H</td>
</tr>
<tr>
<td>P7M1</td>
<td>DATA</td>
<td>0E1H</td>
</tr>
</tbody>
</table>
ORG 0000H
LJMP MAIN

ORG 0100H

MAIN:
MOV SP,#3FH

MOV P0M0,#00H ;set P0.0~P0.7 as bidirectional mode
MOV P0M1,#00H
MOV P1M0,#0FFH ;set P1.0~P1.7 as output mode
MOV P1M1,#00H
MOV P2M0,#00H ;set P2.0~P2.7 as high impedance input mode
MOV P2M1,#0FFH
MOV P3M0,#0FFH ;set P3.0~P3.7 as drain mode
MOV P3M1,#0FFH

JMP $

END

Codes of C
#include "reg51.h"
#include "intrins.h"
sfr P0M0 = 0x94;
sfr P0M1 = 0x93;
sfr P1M0 = 0x92;
sfr P1M1 = 0x91;
sfr P2M0 = 0x96;
sfr P2M1 = 0x95;
sfr P3M0 = 0xb2;
sfr P3M1 = 0xb1;
sfr P4M0 = 0xb4;
sfr P4M1 = 0xb3;
sfr P5M0 = 0xca;
sfr P5M1 = 0xc9;
sfr P6M0 = 0xcc;
sfr P6M1 = 0xcb;
sfr P7M0 = 0xe2;
sfr P7M1 = 0xe1;

void main()
{
    P0M0 = 0x00; //set P0.0~P0.7 as bidirectional mode
    P0M1 = 0x00;
    P1M0 = 0xff; //set P0.0~P0.7 as bidirectional mode
    P1M1 = 0x00;
    P2M0 = 0x00; //set P2.0~P2.7 as high impedance input mode
    P2M1 = 0xff;
    P3M0 = 0xff; //set P3.0~P3.7 as drain mode
    P3M1 = 0xff;

    while (1);
}

10.5.2 Read and write operations on bidirectional port

Assembly codes

| Data     | P0M0 | 094H |
| Data     | P0M1 | 093H |

```plaintext
ORG 0000H
LJMP MAIN

ORG 0100H
MAIN:
MOV SP,#3FH
MOV P0M0,#00H ;set P0.0~P0.7 as bidirectional mode
MOV P0M1,#00H
SETB P0.0 ;P0.0 input high
CLR P0.0 ;P0.0 input low
SETB P0.0 ;enable internal weak pull-up resistor before reading port
NOP ;waiting for 2 clocks
MOV CY,P0.0 ;read status of the ports
JMP $

END
```

Codes of C

```c
#include "reg51.h"
#include "intrins.h"
sfr P0M0 = 0x94;
sfr P0M1 = 0x93;
sbit P00 = P0^0;

void main()
{
    P0M0 = 0x00; //set P0.0~P0.7 as bidirectional mode
    P0M1 = 0x00;

    P00 = 1; //P0.0 input high
    P00 = 0; ;P0.0 input low

    P00 = 1; //enable internal weak pull-up resistor before reading port
    _nop_(); //waiting for 2 clocks
    _nop_();
    CY = P00; ;//read status of the ports

    while (1);
}
```

## 11 Instruction Set

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Description</th>
<th>Bytes</th>
<th>Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD A,Rn</td>
<td>Add register to Accumulator</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ADD A,direct</td>
<td>Add direct byte to Accumulator</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ADD A,@Ri</td>
<td>Add indirect RAM to Accumulator</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ADD A,#data</td>
<td>Add immediate data to Accumulator</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ADDC A,Rn</td>
<td>Add register to Accumulator with Carry</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ADDC A,direct</td>
<td>Add direct byte to Accumulator with Carry</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ADDC A,@Ri</td>
<td>Add indirect RAM to Accumulator with Carry</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ADDC A,#data</td>
<td>Add immediate data to Accumulator with Carry</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>SUBB A,Rn</td>
<td>Subtract Register from Accumulator with borrow</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SUBB A,direct</td>
<td>Subtract direct byte from Accumulator with borrow</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>SUBB A,@Ri</td>
<td>Subtract indirect RAM from Accumulator with borrow</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SUBB A,#data</td>
<td>Subtract immediate data from Accumulator with borrow</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>INC A</td>
<td>Increment Accumulator</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>INC Rn</td>
<td>Increment register</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>INC direct</td>
<td>Increment direct byte</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>INC @Ri</td>
<td>Increment indirect RAM</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>DEC A</td>
<td>Decrement Accumulator</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>DEC Rn</td>
<td>Decrement Register</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>DEC direct</td>
<td>Decrement direct byte</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>DEC @Ri</td>
<td>Decrement indirect RAM</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>INC DPTR</td>
<td>Increment Data Pointer</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MUL AB</td>
<td>Multiply A &amp; B, high byte of result is in B, low byte in A</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>DIV AB</td>
<td>Divide A by B, quotient is in A, remainder is in B.</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>DA A</td>
<td>Decimal Adjust Accumulator</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>ANL A,Rn</td>
<td>AND Register to Accumulator</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ANL A,direct</td>
<td>AND direct byte to Accumulator</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ANL A,@Ri</td>
<td>AND indirect RAM to Accumulator</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ANL A,#data</td>
<td>AND immediate data to Accumulator</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ANL direct,A</td>
<td>AND Accumulator to direct byte</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ANL direct,#data</td>
<td>AND immediate data to direct byte</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>ORL A,Rn</td>
<td>OR register to Accumulator</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ORL A,direct</td>
<td>OR direct byte to Accumulator</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ORL A,@Ri</td>
<td>OR indirect RAM to Accumulator</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Instruction</td>
<td>Description</td>
<td>Mnemonic</td>
<td>Access</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------------------------------------</td>
<td>----------</td>
<td>---------</td>
</tr>
<tr>
<td>ORL A,#data</td>
<td>OR immediate data to Accumulator</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ORL direct,A</td>
<td>OR Accumulator to direct byte</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ORL direct,#data</td>
<td>OR immediate data to direct byte</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>XRL A,Rn</td>
<td>Exclusive-OR register to Accumulator</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XRL A,direct</td>
<td>Exclusive-OR direct byte to Accumulator</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>XRL A,@Ri</td>
<td>Exclusive-OR indirect RAM to Accumulator</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XRL A,#data</td>
<td>Exclusive-OR immediate data to Accumulator</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>XRL direct,A</td>
<td>Exclusive-OR Accumulator to direct byte</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>XRL direct,#data</td>
<td>Exclusive-OR immediate data to direct byte</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>CLR A</td>
<td>Clear Accumulator</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>CPL A</td>
<td>Complement Accumulator</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RL A</td>
<td>Rotate Accumulator Left</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RLC A</td>
<td>Rotate Accumulator Left through the Carry</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RR A</td>
<td>Rotate Accumulator Right</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RRC A</td>
<td>Rotate Accumulator Right through the Carry</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SWAP A</td>
<td>Swap nibbles within the Accumulator</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>CLR C</td>
<td>Clear Carry</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>CLR bit</td>
<td>Clear direct bit</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>SETB C</td>
<td>Set Carry</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>SETB bit</td>
<td>Set direct bit</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>CPL C</td>
<td>Complement Carry</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>CPL bit</td>
<td>Complement direct bit</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ANL C,bit</td>
<td>AND direct bit to Carry</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ANL C,/bit</td>
<td>AND complement of direct bit to Carry</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ORL C,bit</td>
<td>OR direct bit to Carry</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>ORL C,/bit</td>
<td>OR complement of direct bit to Carry</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>MOV C,bit</td>
<td>Move direct bit to Carry</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>MOV bit,C</td>
<td>Move Carry to direct bit</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>MOV A,Rn</td>
<td>Move register to Accumulator</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MOV A,direct</td>
<td>Move direct byte to Accumulator</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>MOV A,@Ri</td>
<td>Move indirect RAM to Accumulator</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MOV A,#data</td>
<td>Move immediate data to Accumulator</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>MOV Rn,A</td>
<td>Move Accumulator to register</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MOV Rn,direct</td>
<td>Move direct byte to register</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>MOV Rn,#data</td>
<td>Move immediate data to register</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>MOV direct,A</td>
<td>Move Accumulator to direct byte</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>MOV direct,Rn</td>
<td>Move register to direct byte</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Instruction</td>
<td>Description</td>
<td>Cycles</td>
<td>Flags</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>MOV direct, direct</td>
<td>Move direct byte to direct</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>MOV direct, @Ri</td>
<td>Move indirect RAM to direct byte</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>MOV direct, #data</td>
<td>Move immediate data to direct byte</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>MOV @Ri, A</td>
<td>Move Accumulator to indirect RAM</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MOV @Ri, direct</td>
<td>Move direct byte to indirect RAM</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>MOV @Ri, #data</td>
<td>Move immediate data to indirect RAM</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>MOV DPTR, #data16</td>
<td>Move 16-bit immediate data to indirect RAM</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>MOVC A, @A+DPTR</td>
<td>Move Code byte relative to DPTR to Accumulator</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>MOVC A, @A+PC</td>
<td>Move Code byte relative to PC to Accumulator</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>MOVX A, @Ri</td>
<td>Move extended RAM (8-bit addr) to Accumulator (Read)</td>
<td>1</td>
<td>3&lt;sup&gt;[1]&lt;/sup&gt;</td>
</tr>
<tr>
<td>MOVX A, @DPTR</td>
<td>Move extended RAM (16-bit addr) to Accumulator (Read)</td>
<td>1</td>
<td>2&lt;sup&gt;[1]&lt;/sup&gt;</td>
</tr>
<tr>
<td>MOVX @Ri, A</td>
<td>Move Accumulator to extended RAM (8-bit addr) (Write)</td>
<td>1</td>
<td>3&lt;sup&gt;[1]&lt;/sup&gt;</td>
</tr>
<tr>
<td>MOVX @DPTR, A</td>
<td>Move Accumulator to extended RAM (16-bit addr) (Write)</td>
<td>1</td>
<td>2&lt;sup&gt;[1]&lt;/sup&gt;</td>
</tr>
<tr>
<td>PUSH direct</td>
<td>Push direct byte onto stack</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>POP direct</td>
<td>POP direct byte from stack</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>XCH A, Rn</td>
<td>Exchange register with Accumulator</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XCH A, direct</td>
<td>Exchange direct byte with Accumulator</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>XCH A, @Ri</td>
<td>Exchange indirect RAM with Accumulator</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>XCHD A, @Ri</td>
<td>Exchange low-order Digit indirect RAM with Accumulator</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ACALL addr11</td>
<td>Absolute Subroutine Call</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>LCALL addr16</td>
<td>Long Subroutine Call</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>RET</td>
<td>Return from Subroutine</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>RETI</td>
<td>Return from interrupt</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>AJMP addr11</td>
<td>Absolute Jump</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>LJMP addr16</td>
<td>Long Jump</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>SJMP rel</td>
<td>Short Jump (relative addr)</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>JMP @A+DPTR</td>
<td>Jump indirect relative to the DPTR</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>JZ rel</td>
<td>Jump if Accumulator is Zero</td>
<td>2</td>
<td>1&lt;sup&gt;[2]&lt;/sup&gt;</td>
</tr>
<tr>
<td>JNZ rel</td>
<td>Jump if Accumulator is not Zero</td>
<td>2</td>
<td>1&lt;sup&gt;[2]&lt;/sup&gt;</td>
</tr>
<tr>
<td>JC rel</td>
<td>Jump if Carry is set</td>
<td>2</td>
<td>1&lt;sup&gt;[2]&lt;/sup&gt;</td>
</tr>
<tr>
<td>JNC rel</td>
<td>Jump if Carry not set</td>
<td>2</td>
<td>1&lt;sup&gt;[2]&lt;/sup&gt;</td>
</tr>
<tr>
<td>JB bit, rel</td>
<td>Jump if direct bit is set</td>
<td>3</td>
<td>1&lt;sup&gt;[2]&lt;/sup&gt;</td>
</tr>
<tr>
<td>JNB bit, rel</td>
<td>Jump if direct bit is not set</td>
<td>3</td>
<td>1&lt;sup&gt;[2]&lt;/sup&gt;</td>
</tr>
<tr>
<td>JBC bit, rel</td>
<td>Jump if direct bit is set &amp; clear bit</td>
<td>3</td>
<td>1&lt;sup&gt;[2]&lt;/sup&gt;</td>
</tr>
<tr>
<td>CJNE A, direct, rel</td>
<td>Compare direct byte to Accumulator and jump if not equal</td>
<td>3</td>
<td>1&lt;sup&gt;[2]&lt;/sup&gt;</td>
</tr>
<tr>
<td>CJNE A, #data, rel</td>
<td>Compare immediate data to Accumulator and Jump if not equal</td>
<td>3</td>
<td>1&lt;sup&gt;[2]&lt;/sup&gt;</td>
</tr>
<tr>
<td>Instruction</td>
<td>Description</td>
<td>Cycles</td>
<td>Machine Cycles</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>--------</td>
<td>----------------</td>
</tr>
<tr>
<td>CJNE Rn,#data,rel</td>
<td>Compare immediate data to register and Jump if not equal</td>
<td>3</td>
<td>1/3[^2]</td>
</tr>
<tr>
<td>CJNE @Ri,#data,rel</td>
<td>Compare immediate data to indirect and Jump if not equal</td>
<td>3</td>
<td>1/3[^2]</td>
</tr>
<tr>
<td>DJNZ Rn,rel</td>
<td>Decrement register and Jump if not Zero</td>
<td>2</td>
<td>1/3[^2]</td>
</tr>
<tr>
<td>DJNZ direct,rel</td>
<td>Decrement direct byte and Jump if not Zero</td>
<td>3</td>
<td>1/3[^2]</td>
</tr>
<tr>
<td>NOP</td>
<td>No Operation</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

[^1]: When accessing external extended RAM, the instruction execution cycle is related to the SPEED [1: 0] bits in the BUS_SPEED register.

[^2]: For the conditional jump statement, the execution cycle will be different based on whether the conditions are met or not. When the conditions are not met, the jump will not occur and continue to execute the next instruction, then execution cycle of the conditional jump statement is 1 machine cycle. When the conditions are met, the jump will occur, the execution cycle of the conditional jump statement is 3 machine cycles.
# 12 Interrupt System

The interrupt system is set up to give the CPU real-time processing capabilities for external emergencies.

If an emergency request occurs when CPU is dealing with something, and the CPU is required to suspend the current work to handle the emergency. After the emergency processing is completed, the CPU returns to the place where it was interrupted and continues the original work. This process is called interrupt. The component that implements this function is called the interrupt system, and the request source that makes the CPU interrupt to suspend the current work is called the interrupt source. Microcontroller interrupt system generally allows multiple interrupt sources. When several interrupt sources simultaneously require the CPU to handle the requests, the CPU should respond the interrupt source which has the highest priority. Usually the CPU handle the interrupt requests according to the priority of interrupt sources. The most urgent incidents have the highest priority. Each interrupt source has a priority level. The CPU always responds the highest priority interrupt request.

Another interrupt source request with a higher priority takes place when the CPU is processing an interrupt source request, that is, the CPU is executing the corresponding interrupt service routine. If the CPU can suspend the original interrupt service routine, and deal with the higher priority interrupt request source, and then return to the original low-level interrupt service routine after processing, this process is called interrupt nesting. Such an interrupt system is called a multi-level interrupt system, whereas an interrupt system without interrupt nesting is called a single-level interrupt system.

The corresponding interrupt request can be masked by turning off the general enable bit (EA / IE.7) or the corresponding interrupt enable bit. The CPU can be enabled to respond to the corresponding interrupt request by turning on the corresponding interrupt enable bit. Every interrupt source can be set independently by software to interrupt enabled or disabled state. The priority of some interrupts can be set by software. Higher priority interrupt requests can interrupt lower priority interrupts, whereas lower priority interrupt requests can not interrupt higher priority interrupts. When two interrupts with the same priority occur simultaneously, the inquiry order determines which interrupt the system responds firstly.

## 12.1 Interrupt Sources of STC8F family

The √ in the following table indicates that the corresponding series have the corresponding interrupt source.

<table>
<thead>
<tr>
<th>interrupt sources</th>
<th>STC8A8K64S4A12 series</th>
<th>STC8A4K64S2A12 series</th>
<th>STC8F2K64S4 series</th>
<th>STC8F8K64S2 series</th>
</tr>
</thead>
<tbody>
<tr>
<td>External interrupt 0 (INT0)</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Timer 0 interrupt (Timer0)</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>External interrupt 1 (INT1)</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Timer 1 interrupt (Timer1)</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
</tbody>
</table>
### 12.1.1 Interrupt Sources of STC8F8K64S4A10 series

STC8F8K64S4A10 series microcontrollers support 22 interrupt sources. They are external interrupt 0 (INT0), Timer 0 interrupt (Timer 0), external interrupt 1 (INT1), Timer 1 interrupt (Timer 1), serial port 1 interrupt (UART1), ADC interrupt, low voltage detection interrupt (LVD), CCP/PCA interrupt, serial port 2 interrupt (UART2), SPI interrupt, external interrupt 2 (INT2), external interrupt 3 (INT3), Timer 2 interrupt (Timer 2), external interrupt 4 (INT4), serial port 3 interrupt (UART3), serial port 4 interrupt (UART4), Timer 3 interrupt (Timer 3), Timer 4 interrupt (Timer 4), Comparator interrupt (CMP), PWM interrupt, PWM fault detection interrupt (PWMFD) and I2C interrupt.

Except for external interrupt 2 (INT2), external interrupt 3 (INT3), serial port 3 (UART3) interrupt, serial port 4 (UART4) interrupt, Timer 2 interrupt, Timer 3 interrupt, Timer 4 interrupt and comparator interrupt having the fixed lowest priority, all the other interrupts have four priority levels.
12.1.2 Interrupt Sources of STC8A8K64S4A12 series

STC8A8K64S4A12 series microcontrollers support 22 interrupt sources. They are external interrupt 0 (INT0), Timer 0 interrupt (Timer 0), external interrupt 1(INT1), Timer 1 interrupt (Timer 1), serial port 1 interrupt (UART1), ADC interrupt, low voltage detection interrupt (LVD), CCP/PCA interrupt, serial port 2 interrupt (UART2), SPI interrupt, external interrupt 2(INT2), external interrupt 3(INT3), Timer 2 interrupt (Timer 2), external interrupt 4 (INT4), serial port 3 interrupt (UART3), serial port 4 interrupt (UART4), Timer 3 interrupt (Timer 3), Timer 4 interrupt (Timer 4), Comparator interrupt (CMP), PWM interrupt, PWM fault detection interrupt (PWMFD) and I2C interrupt.

Except for external interrupt 2 (INT2), external interrupt 3 (INT3), serial port 3(UART3) interrupt, serial port 4(UART4) interrupt, Timer 2 interrupt, Timer 3 interrupt, Timer 4 interrupt and comparator interrupt having the fixed lowest priority, all the other interrupts have four priority levels.

12.1.3 Interrupt Sources of STC8F2K64S4 series

STC8F2K64S4 series microcontrollers support 19 interrupt sources. They are external interrupt 0 (INT0), Timer 0 interrupt (Timer 0), external interrupt 1(INT1), Timer 1 interrupt (Timer 1), serial port 1 interrupt (UART1), low voltage detection interrupt (LVD), CCP/PCA interrupt, serial port 2 interrupt (UART2), SPI interrupt, external interrupt 2(INT2), external interrupt 3(INT3), Timer 2 interrupt (Timer 2), external interrupt 4 (INT4), Timer 3 interrupt (Timer 3), Timer 4 interrupt (Timer 4), Comparator interrupt (CMP) and I2C interrupt.

Except for external interrupt 2 (INT2), external interrupt 3 (INT3), serial port 3(UART3) interrupt, serial port 4(UART4) interrupt, Timer 2 interrupt, Timer 3 interrupt, Timer 4 interrupt and comparator interrupt having the fixed lowest priority, all the other interrupts have four priority levels.

12.1.4 Interrupt Sources of STC8F2K64S4 series

STC8F2K64S4 series microcontrollers support 19 interrupt sources. They are external interrupt 0 (INT0), Timer 0 interrupt (Timer 0), external interrupt 1(INT1), Timer 1 interrupt (Timer 1), serial port 1 interrupt (UART1), low voltage detection interrupt (LVD), CCP/PCA interrupt, serial port 2 interrupt (UART2), SPI interrupt, external interrupt 2(INT2), external interrupt 3(INT3), Timer 2 interrupt (Timer 2), external interrupt 4 (INT4), serial port 3 interrupt (UART3), serial port 4 interrupt (UART4), Timer 3 interrupt (Timer 3), Timer 4 interrupt (Timer 4), Comparator interrupt (CMP) and I2C interrupt.

Except for external interrupt 2 (INT2), external interrupt 3 (INT3), Timer 2 interrupt, Timer 3 interrupt, Timer 4 interrupt and comparator interrupt having the fixed lowest priority, all the other interrupts have four priority levels.
12.2 Interrupt Structure Diagrams of STC8F family
### 12.3 Interrupt List of STC8F family microcontrollers

<table>
<thead>
<tr>
<th>Interrupt source</th>
<th>Interrupt vector</th>
<th>Order</th>
<th>Priority level setup bit</th>
<th>Priority level</th>
<th>Interrupt request flag</th>
<th>Interrupt enable bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT0</td>
<td>0003H</td>
<td>0</td>
<td>PX0PX0H</td>
<td>0/1/2/3</td>
<td>IE0</td>
<td>EX0</td>
</tr>
<tr>
<td>Timer0</td>
<td>000BH</td>
<td>1</td>
<td>PT0,PT0H</td>
<td>0/1/2/3</td>
<td>TF0</td>
<td>ET0</td>
</tr>
<tr>
<td>INT1</td>
<td>0013H</td>
<td>2</td>
<td>PX1,PX1H</td>
<td>0/1/2/3</td>
<td>IE1</td>
<td>EX1</td>
</tr>
<tr>
<td>Timer1</td>
<td>001BH</td>
<td>3</td>
<td>PT1,PT1H</td>
<td>0/1/2/3</td>
<td>TF1</td>
<td>ET1</td>
</tr>
<tr>
<td>UART1</td>
<td>0023H</td>
<td>4</td>
<td>PS,PSH</td>
<td>0/1/2/3</td>
<td>RI</td>
<td></td>
</tr>
<tr>
<td>ADC</td>
<td>002BH</td>
<td>5</td>
<td>PADC,PADCH</td>
<td>0/1/2/3</td>
<td>ADC_FLAG</td>
<td>EADC</td>
</tr>
<tr>
<td>LVD</td>
<td>0033H</td>
<td>6</td>
<td>PLVD,PLV DH</td>
<td>0/1/2/3</td>
<td>LVDF</td>
<td>ELVD</td>
</tr>
<tr>
<td>PCA</td>
<td>003BH</td>
<td>7</td>
<td>PPCA,PPCAH</td>
<td>0/1/2/3</td>
<td>CCF</td>
<td>CF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CCF0</td>
<td>ECCF0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CCF1</td>
<td>ECCF1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CCF2</td>
<td>ECCF2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CCF3</td>
<td>ECCF3</td>
</tr>
<tr>
<td>UART2</td>
<td>0043H</td>
<td>8</td>
<td>PS2,PS2H</td>
<td>0/1/2/3</td>
<td>S2RI</td>
<td></td>
</tr>
<tr>
<td>SPI</td>
<td>004BH</td>
<td>9</td>
<td>PSPI,PSPIH</td>
<td>0/1/2/3</td>
<td>SPIF</td>
<td>ESPI</td>
</tr>
<tr>
<td>INT2</td>
<td>0053H</td>
<td>10</td>
<td></td>
<td>0</td>
<td>INT2IF</td>
<td>EX2</td>
</tr>
<tr>
<td>INT3</td>
<td>005BH</td>
<td>11</td>
<td></td>
<td>0</td>
<td>INT3IF</td>
<td>EX3</td>
</tr>
<tr>
<td>Timer2</td>
<td>0063H</td>
<td>12</td>
<td></td>
<td>0</td>
<td>T2IF</td>
<td>ET2</td>
</tr>
<tr>
<td>INT4</td>
<td>0083H</td>
<td>16</td>
<td>PX4,PX4H</td>
<td>0/1/2/3</td>
<td>INT4IF</td>
<td>EX4</td>
</tr>
<tr>
<td>UART3</td>
<td>008BH</td>
<td>17</td>
<td></td>
<td>0</td>
<td>S3RI</td>
<td></td>
</tr>
<tr>
<td>UART4</td>
<td>0093H</td>
<td>18</td>
<td></td>
<td>0</td>
<td>S4RI</td>
<td></td>
</tr>
<tr>
<td>Timer3</td>
<td>009BH</td>
<td>19</td>
<td></td>
<td>0</td>
<td>T3IF</td>
<td>ET3</td>
</tr>
<tr>
<td>Timer4</td>
<td>00A3H</td>
<td>20</td>
<td></td>
<td>0</td>
<td>T4IF</td>
<td>ET4</td>
</tr>
<tr>
<td>CMP</td>
<td>00ABH</td>
<td>21</td>
<td>PCMP,PCMPH</td>
<td>0/1/2/3</td>
<td>CMPIF</td>
<td>PIE</td>
</tr>
<tr>
<td>interrupt source</td>
<td>interrupt vector</td>
<td>Order</td>
<td>Priority level setup bit</td>
<td>Priority level</td>
<td>interrupt request flag</td>
<td>interrupt enable bit</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------</td>
<td>-------</td>
<td>--------------------------</td>
<td>----------------</td>
<td>-----------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>PWM</td>
<td>00B3H</td>
<td>22</td>
<td>PPWM,PPWMH</td>
<td>0/1/2/3</td>
<td>CBIF</td>
<td>ECBI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C0IF</td>
<td>EC0I &amp;&amp; EC0T1SI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C1IF</td>
<td>EC0I &amp;&amp; EC0T2SI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C2IF</td>
<td>EC1I &amp;&amp; EC1T1SI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C3IF</td>
<td>EC1I &amp;&amp; EC1T2SI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C4IF</td>
<td>EC2I &amp;&amp; EC2T1SI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C5IF</td>
<td>EC2I &amp;&amp; EC2T2SI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C6IF</td>
<td>EC3I &amp;&amp; EC3T1SI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C7IF</td>
<td>EC3I &amp;&amp; EC3T2SI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EC4I &amp;&amp; EC4T1SI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EC4I &amp;&amp; EC4T2SI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EC5I &amp;&amp; EC5T1SI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EC5I &amp;&amp; EC5T2SI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EC6I &amp;&amp; EC6T1SI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EC6I &amp;&amp; EC6T2SI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EC7I &amp;&amp; EC7T1SI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EC7I &amp;&amp; EC7T2SI</td>
</tr>
<tr>
<td>PWMFD</td>
<td>00BBH</td>
<td>23</td>
<td>PPWMFD,PPWMFDH</td>
<td>0/1/2/3</td>
<td>FDIIF</td>
<td>EFDI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MSIF</td>
<td>EMSI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>STAIF</td>
<td>ESTAI</td>
</tr>
<tr>
<td>I2C</td>
<td>00C3H</td>
<td>24</td>
<td>PI2C,PI2CH</td>
<td>0/1/2/3</td>
<td>RXIF</td>
<td>ERXI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TXIF</td>
<td>ETXI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>STOIF</td>
<td>ESTOI</td>
</tr>
</tbody>
</table>

You may declare interrupt service routine in C language as the following,

```c
void INT0_Routine(void) interrupt 0;
void TM0_Routine(void) interrupt 1;
void INT1_Routine(void) interrupt 2;
void TM1_Routine(void) interrupt 3;
void UART1_Routine(void) interrupt 4;
void ADC_Routine(void) interrupt 5;
void LVD_Routine(void) interrupt 6;
void PCA_Routine(void) interrupt 7;
void UART2_Routine(void) interrupt 8;
void SPI_Routine(void) interrupt 9;
void INT2_Routine(void) interrupt 10;
void INT3_Routine(void) interrupt 11;
```
void TM2_Routine(void) interrupt 12;
void INT4_Routine(void) interrupt 16;
void UART3_Routine(void) interrupt 17;
void UART4_Routine(void) interrupt 18;
void TM3_Routine(void) interrupt 19;
void TM4_Routine(void) interrupt 20;
void CMP_Routine(void) interrupt 21;
void PWM_Routine(void) interrupt 22;
void PWMFD_Routine(void) interrupt 23;
void I2C_Routine(void) interrupt 24;

## 12.4 Interrupt Related Registers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Address</th>
<th>Bit Address and Symbol</th>
<th>Value after reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td>interrupt enable register</td>
<td>A8H</td>
<td>B7 B6 B5 B4 B3 B2 B1 B0</td>
<td>0000,0000</td>
</tr>
<tr>
<td>IE2</td>
<td>interrupt enable register 2</td>
<td>AFH</td>
<td></td>
<td>x000,0000</td>
</tr>
<tr>
<td></td>
<td>interrupt and clock output control register</td>
<td>8FH</td>
<td>B7 B6 B5 B4 B3 B2 B1 B0</td>
<td>x000,0000</td>
</tr>
<tr>
<td>INTCLKO</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IP</td>
<td>interrupt Priority Low</td>
<td>B8H</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>IPIH</td>
<td>interrupt Priority High</td>
<td>B7H</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>IP2</td>
<td>2nd interrupt Priority Low</td>
<td>B5H</td>
<td></td>
<td>x000,0000</td>
</tr>
<tr>
<td>IP2H</td>
<td>2nd interrupt Priority High</td>
<td>B6H</td>
<td></td>
<td>x000,0000</td>
</tr>
<tr>
<td>TCON</td>
<td>Timer 0 and 1 control</td>
<td>88H</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>AUXINTF</td>
<td>Extended external interrupt flag</td>
<td>EFH</td>
<td></td>
<td>x000,0000</td>
</tr>
<tr>
<td>SCON</td>
<td>Serial port 1 control</td>
<td>98H</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>S2CON</td>
<td>Serial port 2 control</td>
<td>9AH</td>
<td></td>
<td>0100,0000</td>
</tr>
<tr>
<td>S3CON</td>
<td>Serial port 3 control</td>
<td>ACH</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>S4CON</td>
<td>Serial port 4 control</td>
<td>84H</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>PCON</td>
<td>Power control register</td>
<td>87H</td>
<td></td>
<td>0011,0000</td>
</tr>
<tr>
<td>ADC_CONT</td>
<td>ADC control register</td>
<td>B7H</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>SPSTAT</td>
<td>SPI Status register</td>
<td>CDH</td>
<td></td>
<td>00xx,xxxx</td>
</tr>
<tr>
<td>CCON</td>
<td>PCA Control Register</td>
<td>D8H</td>
<td></td>
<td>00xx,xxxx</td>
</tr>
<tr>
<td>CMOD</td>
<td>PCA Mode Register</td>
<td>D9H</td>
<td></td>
<td>00xx,xxxx</td>
</tr>
<tr>
<td>CCAPM0</td>
<td>PCA 0 Mode Register</td>
<td>DAH</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>CCAPM1</td>
<td>PCA 1 Mode Register</td>
<td>DBH</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>CCAPM2</td>
<td>PCA 2 Mode Register</td>
<td>D7H</td>
<td></td>
<td>x000,0000</td>
</tr>
<tr>
<td>CCAPM3</td>
<td>PCA 3 Mode Register</td>
<td>DDH</td>
<td></td>
<td>x000,0000</td>
</tr>
<tr>
<td>CMPCR1</td>
<td>Comparator control register</td>
<td>E6H</td>
<td></td>
<td>0000,0000</td>
</tr>
</tbody>
</table>
12.4.1 Interrupt Enable Control Registers (interrupt Enable bits)

**IE (interrupt Enable Register)**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td>A8H</td>
<td>EA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ELVD</td>
<td>EADC</td>
<td>ES</td>
<td>ET1</td>
<td>EX1</td>
<td>ET0</td>
<td>EX0</td>
</tr>
</tbody>
</table>

EA: The general or global interrupt enable control bit. The function of EA is to allow interrupts to be multi-level controlled. That is, every interrupt source is controlled by EA firstly and then by its own interrupt enable control bit.

0: all interrupts are masked, and no interrupt would be acknowledged.
1: enable the CPU interrupt, every interrupt source would be individually enabled or disabled by setting or clearing its enable bit.

ELVD: Low voltage detection interrupt enable bit.

0: disable low voltage detection interrupt.
1: enable Low voltage detection interrupt.

EADC: ADC interrupt enable bit.

0: disable ADC interrupt.
1: enable ADC interrupt.
ES: Serial Port 1 (UART1) interrupt enable bit.
   0: disable UART1 interrupt.
   1: enable UART1 interrupt.

ET1: Timer 1 interrupt enable bit.
   0: disable Timer 1 interrupt.
   1: enable Timer 1 interrupt.

EX1: External interrupt 1 enable bit.
   0: disable external interrupt 1.
   1: enable external interrupt 1.

ET0: Timer 0 interrupt enable bit.
   0: disable Timer 0 interrupt.
   1: enable Timer 0 interrupt.

EX0: External interrupt 0 enable bit.
   0: disable external interrupt 0.
   1: enable external interrupt 0.

IE2 (interrupt Enable 2 Register) (Non bit-addressable)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE2</td>
<td>AFH</td>
<td>-</td>
<td>ET4</td>
<td>ET3</td>
<td>ES4</td>
<td>ES3</td>
<td>ET2</td>
<td>ESPI</td>
<td>ES2</td>
</tr>
</tbody>
</table>

ET4: Timer 4 interrupt enable bit.
   0: disable Timer 4 interrupt.
   1: enable Timer 4 interrupt.

ET3: Timer 3 interrupt enable bit.
   0: disable Timer 3 interrupt.
   1: enable Timer 3 interrupt.

ES4: Serial Port 4 (UART4) interrupt enable bit.
   0: disable UART4 interrupt.
   1: enable UART4 interrupt.

ES3: Serial Port 3 (UART3) interrupt enable bit.
   0: disable UART3 interrupt.
   1: enable UART3 interrupt.

ET2: Timer 2 interrupt enable bit.
   0: disable Timer 2 interrupt.
   1: enable Timer 2 interrupt.

ESPI: SPI interrupt enable bit.
   0: disable SPI interrupt.
   1: enable SPI interrupt.

ES2: Serial Port 2 (UART2) interrupt enable bit.
   0: disable UART2 interrupt.
   1: enable UART2 interrupt.

INTCLKO (External interrupt Enable and Clock Output register control)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTCLKO</td>
<td>8FH</td>
<td>-</td>
<td>EX4</td>
<td>EX3</td>
<td>EX2</td>
<td>-</td>
<td>T2CLKO</td>
<td>T1CLKO</td>
<td>T0CLKO</td>
</tr>
</tbody>
</table>

EX4: External interrupt 4 enable bit.
   0: disable External interrupt 4.
1: enable External interrupt 4.

EX3: External interrupt 3 enable bit.
0: disable External interrupt 3.
1: enable External interrupt 3.

EX2: External interrupt 2 enable bit.
0: disable External interrupt 2.
1: enable External interrupt 2.

### PCA/CCP interrupt control registers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMOD</td>
<td>D9H</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CPS[2:0] ECF</td>
</tr>
<tr>
<td>CCAPM0</td>
<td>DAH</td>
<td>-</td>
<td>ECOM0</td>
<td>CCAPP0</td>
<td>CCAPN0</td>
<td>MAT0</td>
<td>TOG0</td>
<td>PWM0</td>
<td>ECCF0</td>
</tr>
<tr>
<td>CCAPM1</td>
<td>DBH</td>
<td>-</td>
<td>ECOM1</td>
<td>CCAPP1</td>
<td>CCAPN1</td>
<td>MAT1</td>
<td>TOG1</td>
<td>PWM1</td>
<td>ECCF1</td>
</tr>
<tr>
<td>CCAPM2</td>
<td>DCH</td>
<td>-</td>
<td>ECOM2</td>
<td>CCAPP2</td>
<td>CCAPN2</td>
<td>MAT2</td>
<td>TOG2</td>
<td>PWM2</td>
<td>ECCF2</td>
</tr>
<tr>
<td>CCAPM3</td>
<td>DDH</td>
<td>-</td>
<td>ECOM3</td>
<td>CCAPP3</td>
<td>CCAPN3</td>
<td>MAT3</td>
<td>TOG3</td>
<td>PWM3</td>
<td>ECCF3</td>
</tr>
</tbody>
</table>

ECF: PCA counter interrupt enable bit.
0: disable PCA counter interrupt.
1: enable PCA counter interrupt.

ECCF0: PCA 0 interrupt enable bit.
0: disable PCA 0 interrupt.
1: enable PCA 0 interrupt.

ECCF1: PCA 1 interrupt enable bit.
0: disable PCA 1 interrupt.
1: enable PCA 1 interrupt.

ECCF2: PCA 2 interrupt enable bit.
0: disable PCA 2 interrupt.
1: enable PCA 2 interrupt.

ECCF3: PCA 3 interrupt enable bit.
0: disable PCA 3 interrupt.
1: enable PCA 3 interrupt.

### CMPCR1 (Comparator control register1)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPCR1</td>
<td>E6H</td>
<td>CMPEN</td>
<td>CMPIF</td>
<td>PIE</td>
<td>NIE</td>
<td>PIS</td>
<td>NIS</td>
<td>CMPOE</td>
<td>CMPRES</td>
</tr>
</tbody>
</table>

PIE: Comparator rising-edge interrupt enable bit.
0: disable comparator rising-edge interrupt.
1: enable comparator rising-edge interrupt.

NIE: Comparator falling-edge interrupt enable bit.
0: disable comparator falling-edge interrupt.
1: enable comparator falling-edge interrupt.

### PWMCR (PWM Control register)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWMCR</td>
<td>FEH</td>
<td>CMPEN</td>
<td>ENPWM</td>
<td>ECBI</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
ECBI: PWM counter interrupt enable bit.
  0: disable PWM counter interrupt.
  1: enable PWM counter interrupt.

PWMFDCR (PWM Fault Detection Control Register)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWMFDCR</td>
<td>F7H</td>
<td>INVCMP</td>
<td>INVIO</td>
<td>ENFD</td>
<td>FLTFLO</td>
<td>EFDI</td>
<td>FDCMP</td>
<td>FDIO</td>
<td>FDIF</td>
</tr>
</tbody>
</table>

EFDI: PWM external fault event interrupt enable bit.
  0: disable PWM external fault event interrupt.
  1: enable PWM external fault event interrupt.

Enhanced PWM control registers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWM0CR</td>
<td>FF04H</td>
<td>ENC0O</td>
<td>C0INI</td>
<td>-</td>
<td>C0_S[1:0]</td>
<td>EC0I</td>
<td>EC0T2SI</td>
<td>EC0T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM1CR</td>
<td>FF14H</td>
<td>ENC1O</td>
<td>C1INI</td>
<td>-</td>
<td>C1_S[1:0]</td>
<td>EC1I</td>
<td>EC1T2SI</td>
<td>EC1T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM2CR</td>
<td>FF24H</td>
<td>ENC2O</td>
<td>C2INI</td>
<td>-</td>
<td>C2_S[1:0]</td>
<td>EC2I</td>
<td>EC2T2SI</td>
<td>EC2T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM3CR</td>
<td>FF34H</td>
<td>ENC3O</td>
<td>C3INI</td>
<td>-</td>
<td>C3_S[1:0]</td>
<td>EC3I</td>
<td>EC3T2SI</td>
<td>EC3T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM4CR</td>
<td>FF44H</td>
<td>ENC4O</td>
<td>C4INI</td>
<td>-</td>
<td>C4_S[1:0]</td>
<td>EC4I</td>
<td>EC4T2SI</td>
<td>EC4T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM5CR</td>
<td>FF54H</td>
<td>ENC5O</td>
<td>C5INI</td>
<td>-</td>
<td>C5_S[1:0]</td>
<td>EC5I</td>
<td>EC5T2SI</td>
<td>EC5T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM6CR</td>
<td>FF64H</td>
<td>ENC6O</td>
<td>C6INI</td>
<td>-</td>
<td>C6_S[1:0]</td>
<td>EC6I</td>
<td>EC6T2SI</td>
<td>EC6T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM7CR</td>
<td>FF74H</td>
<td>ENC7O</td>
<td>C7INI</td>
<td>-</td>
<td>C7_S[1:0]</td>
<td>EC7I</td>
<td>EC7T2SI</td>
<td>EC7T1SI</td>
<td></td>
</tr>
</tbody>
</table>

ECnI: PWMn level flipping interrupt enable bit.
  0: disable PWMn interrupt.
  1: enable PWMn interrupt.

ECnT2SI: PWMn second flipping interrupt enable bit.
  0: disable PWMn second flipping interrupt.
  1: enable PWMn second flipping interrupt.

ECnT1SI: PWMn first flipping interrupt enable bit.
  0: disable PWMn first flipping interrupt.
  1: enable PWMn first flipping interrupt.

I2C control registers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2CMSCR</td>
<td>FE81H</td>
<td>EMSI</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>MSCMD[2:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I2CSLCR</td>
<td>FE83H</td>
<td>ESTAI</td>
<td>ERXI</td>
<td>ETXI</td>
<td>ESTOI</td>
<td>-</td>
<td>-</td>
<td>SLRST</td>
<td></td>
</tr>
</tbody>
</table>

EMSI: I2C master mode interrupt enable bit.
  0: disable I2C master mode interrupt.
  1: enable I2C master mode interrupt.

ESTAI: I2C slave receives the START event interrupt enable bit.
  0: disable I2C slave receives the START event interrupt.
  1: enable I2C slave receives the START event interrupt.

ERXI: I2C slave completes receiving data event interrupt enable bit.
  0: disable I2C slave completes receiving data event interrupt.
  1: enable I2C slave completes receiving data event interrupt.

ETXI: I2C slave completes transmitting data event interrupt enable bit.
  0: disable I2C slave completes transmitting data event interrupt.
  1: enable I2C slave completes transmitting data event interrupt.
1: enable I^2C slave completes transmitting data event interrupt.

ESTOI: I^2C slave receives a STOP event interrupt enable bit.

0: disable I^2C slave receives a STOP event interrupt.

1: enable I^2C slave receives a STOP event interrupt.

### 12.4.2 Interrupt Request Registers (interrupt flags)

**Timer control register**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCON</td>
<td>88H</td>
<td>TF1</td>
<td>TR1</td>
<td>TF0</td>
<td>TR0</td>
<td>IE1</td>
<td>IT1</td>
<td>IE0</td>
<td>IT0</td>
</tr>
</tbody>
</table>

TF1: Timer/Counter 1 Overflow Flag. Set by hardware on Timer/Counter 1 overflow. The flag can be cleared by software, however, it will be automatically cleared by the hardware when processor enters the Timer 1 interrupt service routine.

TF0: Timer/Counter 0 Overflow Flag. Set by hardware on Timer/Counter 0 overflow. The flag can be cleared by software, however, it will be automatically cleared by the hardware when processor enters the Timer 1 interrupt service routine.

IE1: External interrupt 1 request flag. Set by hardware when external interrupt rising or falling edge defined by IT1 is detected. The flag can be cleared by software, however, it will be automatically cleared when the processor enters the external interrupt 1 service routine.

IE0: External interrupt 0 request flag. Set by hardware when external interrupt rising or falling edge defined by IT0 is detected. The flag can be cleared by software, however, it will be automatically cleared when the processor enters the external interrupt 0 service routine.

**Auxiliary interrupt flag register**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUXINTIF</td>
<td>EFH</td>
<td>-</td>
<td>INT4IF</td>
<td>INT3IF</td>
<td>INT2IF</td>
<td>-</td>
<td>T4IF</td>
<td>T3IF</td>
<td>T2IF</td>
</tr>
</tbody>
</table>

INT4IF: external int 4 interrupt request flag, which should be cleared by software.

INT3IF: external int 3 interrupt request flag, which should be cleared by software.

INT2IF: external int 2 interrupt request flag, which should be cleared by software.

T4IF: timer 4 overflow interrupt flag, which should be cleared by software.

T3IF: timer 3 overflow interrupt flag, which should be cleared by software.

T2IF: timer 2 overflow interrupt flag, which should be cleared by software.

**Serial port control registers**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCON</td>
<td>98H</td>
<td>SM0/FE</td>
<td>SM1</td>
<td>SM2</td>
<td>REN</td>
<td>TB8</td>
<td>RB8</td>
<td>TI</td>
<td>RI</td>
</tr>
<tr>
<td>S2CON</td>
<td>9AH</td>
<td>S2SM0</td>
<td>-</td>
<td>S2SM2</td>
<td>S2REN</td>
<td>S2TB8</td>
<td>S2RB8</td>
<td>S2TI</td>
<td>S2RI</td>
</tr>
<tr>
<td>S3CON</td>
<td>ACH</td>
<td>S3SM0</td>
<td>S3ST3</td>
<td>S3SM2</td>
<td>S3REN</td>
<td>S3TB8</td>
<td>S3RB8</td>
<td>S3TI</td>
<td>S3RI</td>
</tr>
<tr>
<td>S4CON</td>
<td>84H</td>
<td>S4SM0</td>
<td>S4ST4</td>
<td>S4SM2</td>
<td>S4REN</td>
<td>S4TB8</td>
<td>S4RB8</td>
<td>S4TI</td>
<td>S4RI</td>
</tr>
</tbody>
</table>

TI: Transmit interrupt flag of UART1, which must be cleared manually by software.
RI: Receive interrupt flag of UART1, which must be cleared manually by software.
S2TI: Transmit interrupt flag of UART2, which must be cleared manually by software.
S2RI: Receive interrupt flag of UART2, which must be cleared manually by software.
S3TI: Transmit interrupt flag of UART3, which must be cleared manually by software.
S3RI: Receive interrupt flag of UART3, which must be cleared manually by software.
S4TI: Transmit interrupt flag of UART4, which must be cleared manually by software.
S4RI: Receive interrupt flag of UART4, which must be cleared manually by software.

**Power control register**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCON</td>
<td>87H</td>
<td>SMOD</td>
<td>SMOD0</td>
<td>LVDF</td>
<td>POF</td>
<td>GF1</td>
<td>GF0</td>
<td>PD</td>
<td>IDL</td>
</tr>
</tbody>
</table>

LVDF: Low voltage detection interrupt flag, which should be cleared by software.

**ADC control register**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC_CONTR</td>
<td>BCH</td>
<td>ADC_POWER</td>
<td>ADC_START</td>
<td>ADC_FLAG</td>
<td>-</td>
<td>ADC_CHS[3:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ADC_FLAG: ADC completes conversion interrupt request flag, which should be cleared by software.

**SPI status register**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPSTAT</td>
<td>CDH</td>
<td>SPIF</td>
<td>WCOL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

SPIF: SPI transfer completion interrupt request flag, which should be cleared by software.

**PCA control register**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCON</td>
<td>D8H</td>
<td>CF</td>
<td>CR</td>
<td>-</td>
<td>-</td>
<td>CCF3</td>
<td>CCF2</td>
<td>CCF1</td>
<td>CCF0</td>
</tr>
</tbody>
</table>

CF: PCA counter overflow interrupt request flag, which should be cleared by software.
CCF3: PCA3 interrupt request flag, which should be cleared by software.
CCF2: PCA2 interrupt request flag, which should be cleared by software.
CCF1: PCA1 interrupt request flag, which should be cleared by software.
CCF0: PCA0 interrupt request flag, which should be cleared by software.

**Comparator control register 1**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPPCR1</td>
<td>E6H</td>
<td>CMPEN</td>
<td>CMPIF</td>
<td>PIE</td>
<td>NIE</td>
<td>PIS</td>
<td>NIS</td>
<td>CMPOE</td>
<td>CMPRES</td>
</tr>
</tbody>
</table>

CMPIF: Comparator interrupt request flag, which should be cleared by software.

**PWM Configuration Register**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWMCFG</td>
<td>F1H</td>
<td>CBIF</td>
<td>ETADC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

CBIF: PWM counter interrupt request flag, which should be cleared by software.

**PWM interrupt flag register**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
</table>
C7IF: PWM7 interrupt request flag, which should be cleared by software.
C6IF: PWM6 interrupt request flag, which should be cleared by software.
C5IF: PWM5 interrupt request flag, which should be cleared by software.
C4IF: PWM4 interrupt request flag, which should be cleared by software.
C3IF: PWM3 interrupt request flag, which should be cleared by software.
C2IF: PWM2 interrupt request flag, which should be cleared by software.
C1IF: PWM1 interrupt request flag, which should be cleared by software.
C0IF: PWM0 interrupt request flag, which should be cleared by software.

### PWM fault detection control register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWMFDCR</td>
<td>F7H</td>
<td>INVCMP</td>
<td>INVIO</td>
<td>ENFD</td>
<td>FLTFLIO</td>
<td>EFDI</td>
<td>FDCMP</td>
<td>FDIO</td>
<td>FDIF</td>
</tr>
</tbody>
</table>

FDIF: PWM fault detection interrupt request flag, which should be cleared by software.

### I2C status registers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2CMSST</td>
<td>FE82H</td>
<td>MSBUSY</td>
<td>MSIF</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>MSACKI</td>
<td>MSACKO</td>
</tr>
<tr>
<td>I2CSSLST</td>
<td>FE84H</td>
<td>SLBUSY</td>
<td>STAIF</td>
<td>RXIF</td>
<td>TXIF</td>
<td>STOIF</td>
<td>TXING</td>
<td>SLACKI</td>
<td>SLACKO</td>
</tr>
</tbody>
</table>

MSIF: I2C master mode interrupt request flag, which should be cleared by software.

ESTAI: I2C slave receives the START event interrupt request flag, which should be cleared by software.

ERXI: I2C slave completes receiving data event interrupt request flag, which should be cleared by software.

ETXI: I2C slave completes transmitting data event interrupt request flag, which should be cleared by software.

ESTOI: I2C slave receives a STOP event interrupt request flag, which should be cleared by software.

### 12.4.3 Interrupt Priority Control Registers

#### interrupt priority control registers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP</td>
<td>B8H</td>
<td>PPCA</td>
<td>PLVD</td>
<td>PADC</td>
<td>PS</td>
<td>PT1</td>
<td>PX1</td>
<td>PT0</td>
<td>PX0</td>
</tr>
<tr>
<td>IPH</td>
<td>B7H</td>
<td>PPCAH</td>
<td>PLVDH</td>
<td>PADCH</td>
<td>PSH</td>
<td>PT1H</td>
<td>PX1H</td>
<td>PT0H</td>
<td>PX0H</td>
</tr>
<tr>
<td>IP2</td>
<td>B5H</td>
<td>-</td>
<td>PI2C</td>
<td>PCMP</td>
<td>PX4</td>
<td>PPWMFD</td>
<td>PPWM</td>
<td>PSPI</td>
<td>PS2</td>
</tr>
<tr>
<td>IP2H</td>
<td>B6H</td>
<td>-</td>
<td>PI2CH</td>
<td>PCMPH</td>
<td>PX4H</td>
<td>PPWMFDH</td>
<td>PPWMH</td>
<td>PSPIH</td>
<td>PS2H</td>
</tr>
</tbody>
</table>

PX0H, PX0: External interrupt 0 interrupt priority control bit.

00: INT0 interrupt priority level is 0 (lowest)
01: INT0 interrupt priority level is 1
10: INT0 interrupt priority level is 2
11: INT0 interrupt priority level is 3 (highest)

PT0H, PT0: Timer 0 interrupt priority control bit.

00: Timer 0 interrupt priority level is 0 (lowest)
01: Timer 0 interrupt priority level is 1
10: Timer 0 interrupt priority level is 2
11: Timer 0 interrupt priority level is 3 (highest)

PX1H,PX1: External interrupt 1 interrupt priority control bit.
  00: INT1 interrupt priority level is 0 (lowest)
  01: INT1 interrupt priority level is 1
  10: INT1 interrupt priority level is 2
  11: INT1 interrupt priority level is 3 (highest)

PT1H,PT1: Timer 1 interrupt priority control bit.
  00: Timer 1 interrupt priority level is 0 (lowest)
  01: Timer 1 interrupt priority level is 1
  10: Timer 1 interrupt priority level is 2
  11: Timer 1 interrupt priority level is 3 (highest)

PSH,PS: UART1 interrupt priority control bit.
  00: UART1 interrupt priority level is 0 (lowest)
  01: UART1 interrupt priority level is 1
  10: UART1 interrupt priority level is 2
  11: UART1 interrupt priority level is 3 (highest)

PADCH,PADC: ADC interrupt priority control bit.
  00: ADC interrupt priority level is 0 (lowest)
  01: ADC interrupt priority level is 1
  10: ADC interrupt priority level is 2
  11: ADC interrupt priority level is 3 (highest)

PLVDH,PLVD: Low voltage detection interrupt priority control bit.
  00: LVD interrupt priority level is 0 (lowest)
  01: LVD interrupt priority level is 1
  10: LVD interrupt priority level is 2
  11: LVD interrupt priority level is 3 (highest)

PPCAH,PPCA: CCP/PCA interrupt priority control bit.
  00: CCP/PCA interrupt priority level is 0 (lowest)
  01: CCP/PCA interrupt priority level is 1
  10: CCP/PCA interrupt priority level is 2
  11: CCP/PCA interrupt priority level is 3 (highest)

PS2H,PS2: UART2 interrupt priority control bit.
  00: UART2 interrupt priority level is 0 (lowest)
  01: UART2 interrupt priority level is 1
  10: UART2 interrupt priority level is 2
  11: UART2 interrupt priority level is 3 (highest)

PSPIH,PSPI: SPI interrupt priority control bit.
  00: SPI interrupt priority level is 0 (lowest)
  01: SPI interrupt priority level is 1
  10: SPI interrupt priority level is 2
  11: SPI interrupt priority level is 3 (highest)
PPWMH, PPWM: PWM interrupt priority control bit.
  00: PWM interrupt priority level is 0 (lowest)
  01: PWM interrupt priority level is 1
  10: PWM interrupt priority level is 2
  11: PWM interrupt priority level is 3 (highest)

PPWMFDH, PPWMFD: PWM fault detection interrupt priority control bit.
  00: PWMFD interrupt priority level is 0 (lowest)
  01: PWMFD interrupt priority level is 1
  10: PWMFD interrupt priority level is 2
  11: PWMFD interrupt priority level is 3 (highest)

PX4H, PX4: External interrupt 4 interrupt priority control bit.
  00: INT4 interrupt priority level is 0 (lowest)
  01: INT4 interrupt priority level is 1
  10: INT4 interrupt priority level is 2
  11: INT4 interrupt priority level is 3 (highest)

PCMPH, PCMP: Comparator interrupt priority control bit.
  00: CMP interrupt priority level is 0 (lowest)
  01: CMP interrupt priority level is 1
  10: CMP interrupt priority level is 2
  11: CMP interrupt priority level is 3 (highest)

PI2CH, PI2C: I2C interrupt priority control bit.
  00: I2C interrupt priority level is 0 (lowest)
  01: I2C interrupt priority level is 1
  10: I2C interrupt priority level is 2
  11: I2C interrupt priority level is 3 (highest)

12.5 Demo codes

12.5.1 INT0 interrupt (rising and falling edges)

Assembly code

```
ORG 0000H
LJMP MAIN

ORG 0003H
LJMP INT0ISR

INT0ISR:
    JB INT0,RISING          ;judging rising and falling edges
    CPL P1.0                ;test the port

RETI

RISING:
    CPL P1.1                ;test the port

RETI
```
MAIN:

```
MOV SP,#3FH
CLR IT0 ;enable the rising and falling interrupt of INT0
SETB EX0 ;enable the interrupt of INT0
SETB EA
JMP $
END
```

C code

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

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

void INT0_Isr() interrupt 0
{
  if (INT0)  //judging rising and falling edges
  {
    P10 = !P10; //test the port
  }
  else
  {
    P11 = !P11; //test the port
  }
}

void main()
{
  IT0 = 0; //enable the rising and falling interrupt of INT0
  EX0 = 1; //enable the interrupt of INT0
  EA = 1;

  while (1);
}
```

12.5.2 INT0 interrupt(falling edge)

Assembly code

```
ORG 0000H
LJMP MAIN
ORG 0003H
LJMP INT0ISR

ORG 0100H
INT0ISR:
  CPL P1.0 ;test the port
  RETI

MAIN:
  MOV SP,#3FH
```

SETB IT0 ;enable the falling interrupt of INT0
SETB EX0 ;enable the interrupt of INT0
SETB EA
JMP $  

END  

C code
#include "reg51.h"
#include "intrins.h"

sbit P10 = P1^0;

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

void main()
{
    IT0 = 1; //enable the falling interrupt of INT0
    EX0 = 1; //enable the interrupt of INT0
    EA = 1;

    while (1);
}

12.5.3 INT1 interrupt(rising and falling edges)

Assembly code
ORG 0000H
LJMP MAIN
ORG 0013H
LJMP INT1ISR

ORG 0100H
INT1ISR:
    JB INT1,RISING ;judging rising and falling edges
    CPL P1.0 ;test the port
    RETI
RISING:
    CPL P1.1 ;test the port
    RETI

MAIN:
    MOV SP,#3FH
    CLR IT1
    SETB EX1
    SETB EA
    JMP $
C code

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

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

void INT1_Isr() interrupt 2
{
    if (INT1)        //judging rising and falling edges
    {
        P10 = !P10;   //test the port
    }
    else
    {
        P11 = !P11;   //test the port
    }
}

void main()
{
    IT1 = 0;
    EX1 = 1;
    EA = 1;

    while (1);
}
```

12.5.4 INT1 interrupt(falling edge)

Assembly code

```
ORG 0000H
LJMP MAIN
ORG 0013H
LJMP INT1ISR

ORG 0100H

INT1ISR:
    CPL P1.0 ;test the port
    RETI

MAIN:
    MOV SP,#3FH
    SETB IT1 ;enable the rising and falling interrupt of INT1
    SETB EX1 ;enable INT1 interrupt
    SETB EA
    JMP $}
```

END
C code

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

sbit P10 = P1^0;

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

void main()
{
    IT1 = 1; // enable the falling interrupt of INT1
    EX1 = 1; // enable INT1 interrupt
    EA = 1;

    while (1);
}
```

12.5.5 INT2 interrupt(falling edge)

Assembly code

```assembly
ORG 0000H
LJMP MAIN

ORG 0053H
LJMP INT2ISR

INT2ISR:
    CPL P1.0 ; test the port
    RETI

MAIN:
    MOV SP,#3FH
    MOV INTCLKO,#EX2 ; enable INT2 interrupt
    JMP $ 

END
```

C code

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

sfr INTCLKO = 0x8f;
#define EX2 0x10
```
```c
#define EX3 0x20
#define EX4 0x40
sbit P10 = P1^0;

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

void main()
{
    INTCLKO = EX2;
    EA = 1;

    while (1);
}
```

### 12.5.6 INT3 interrupt (falling edge)

**Assembly code**

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

ORG 0000H
LJMP MAIN
ORG 005BH
LJMP INT3ISR

INT3ISR:
CPL P1.0 ; test the port
RETI

MAIN:
MOV SP,#3FH
MOV INTCLKO,#EX3 ; enable INT3 interrupt
SETB EA
JMP $

END
```

**C code**

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

sfr INTCLKO = 0x8f;
#define EX2 0x10
#define EX3 0x20
#define EX4 0x40
sbit P10 = P1^0;
```
void INT3_Isr() interrupt 11
{
    P10 = !P10; //test the port
}

void main()
{
    INTCLKO = EX3; //enable INT3 interrupt
    while (1);
}

### 12.5.7 INT4 interrupt (falling)

#### Assembly code

<table>
<thead>
<tr>
<th></th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTCLKO</td>
<td>8FH</td>
</tr>
<tr>
<td>EX2</td>
<td>10H</td>
</tr>
<tr>
<td>EX3</td>
<td>20H</td>
</tr>
<tr>
<td>EX4</td>
<td>40H</td>
</tr>
</tbody>
</table>

ORG 0000H
LJMP MAIN
ORG 0083H
LJMP INT4ISR

ORG 0100H
INT4ISR:
CPL P1.0 ;test the port
RETI

MAIN:
MOV SP,#3FH
MOV INTCLKO,#EX4 ;enable INT4 interrupt
SETB EA
JMP $

END

#### C code

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

sfr INTCLKO = 0x8f;
#define EX2  0x10
#define EX3  0x20
#define EX4  0x40
sbit P10 = P1^0;

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

Nantong guoxin Microelectronics Co., Ltd.  Tel: 0513-5501 2928/2929/2966  Fax: 0513-5501 2926/2956/2947  - 188 -
void main()
{
    INTCLKO = EX4; //enable INT4 interrupt
    EA = 1;

    while (1);
}

12.5.8 timer0 interrupt

Assembly code

```
ORG 0000H
LJMP MAIN
ORG 000BH
LJMP TM0ISR

TM0ISR:
    CPL P1.0 ;test the port
    RETI

MAIN:
    MOV SP,#3FH
    MOV TMOD,#00H
    MOV TL0,#66H ;65536-11.0592M/12/1000
    MOV TH0,#0FCH
    SETB TR0 ;start timer
    SETB ET0 ;enable timer interrupt
    SETB EA
    JMP $ END
```

C code

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

sbit P10 = P1^0;

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

void main()
{
    TMOD = 0x00;
    TL0 = 0x66; ;65536-11.0592M/12/1000
    TH0 = 0xfc;
```
TR0 = 1;  //start timer
ET0 = 1;  //enable timer interrupt
EA = 1;

while (1);
}

12.5.9 Timer1 interrupt

Assembly code

```assembly
ORG 0000H
LJMP MAIN

ORG 001BH
LJMP TM1ISR

ORG 0100H

TM1ISR:
CPL P1.0 ; test the port
RETI

MAIN:
MOVS SP,#3FH

MOV TMOD,#00H
MOV TL1,#66H ; 65536-11.0592M/12/1000
MOV TH1,#0FCH
SETB TR1 ; start timer
SETB ET1 ; enable timer interrupt
SETB EA

JMP $

END
```

C code

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

sbit P10 = P1^0;

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

void main()
{
    TMOD = 0x00;
    TL1 = 0x66; // 65536-11.0592M/12/1000
    TH1 = 0xfc;
    TR1 = 1; // start timer
    ET1 = 1; // enable timer interrupt
    EA = 1;
```
while (1);

12.5.10  Timer2 interrupt

Assembly code

```
while (1);

12.5.10  Timer2 interrupt

Assembly code

T2L  DATA  0D7H
T2H  DATA  0D6H
AUXR DATA  8EH
IE2  DATA  0AFH
ET2  EQU  04H
AUXINTIF DATA  0EFH
T2IF  EQU  01H

ORG  0000H
LJMP  MAIN

ORG  0063H
LJMP  TM2ISR

ORG  0100H

TM2ISR:
CPL P1.0 ;test the port
ANL AUXINTIF,#NOT T2IF ;clear the symbol of interrupt
RETI

MAIN:
MOV  SP,#3FH
MOV  T2L,#66H ;65536-11.0592M/12/1000
MOV  T2H,#0FCH
MOV  AUXR,#10H ;start timer
MOV  IE2,#ET2 ;enable timer interrupt
SETB EA

JMP  $
void TM2_Isr() interrupt 12
{
    P10 = !P10; //test the port
    AUXINTIF &= ~T2IF; //clear the symbol of interrupt
}

void main()
{
    T2L = 0x66; //65536-11.0592M/12/1000
    T2H = 0xfc;
    AUXR = 0x10; //start timer
    IE2 = ET2;
    EA = 1;

    while (1);
}

12.5.11 Timer3 interrupt

Assembly code

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>T3L</td>
<td>DATA</td>
<td>0D5H</td>
</tr>
<tr>
<td>T3H</td>
<td>DATA</td>
<td>0D4H</td>
</tr>
<tr>
<td>T4T3M</td>
<td>DATA</td>
<td>0D1H</td>
</tr>
<tr>
<td>IE2</td>
<td>DATA</td>
<td>0AFH</td>
</tr>
<tr>
<td>ET3</td>
<td>EQU</td>
<td>20H</td>
</tr>
<tr>
<td>AUXINTIF</td>
<td>DATA</td>
<td>0EFH</td>
</tr>
<tr>
<td>T3IF</td>
<td>EQU</td>
<td>02H</td>
</tr>
</tbody>
</table>

ORG 0000H
LJMP MAIN
ORG 009BH
LJMP TM3ISR

TM3ISR:
  CPL P1.0 ;test the port
  ANL AUXINTIF,#NOT T3IF ;clear the symbol of interrupt
  RETI

MAIN:
  MOV SP,#3FH
  MOV T3L,#66H ;65536-11.0592M/12/1000
  MOV T3H,#0FCH
  MOV T4T3M,#08H ;start timer
  MOV IE2,#ET3 ;enable timer interrupt
  SETB EA
  JMP $

END

C code
12.5.12 Timer4 interrupt

Assembly code

```
T4L DATA 0D3H
T4H DATA 0D2H
T4T3M DATA 0D1H
IE2 DATA 0AFH
ET4 EQU 40H
AUXINTIF DATA 0EFH
T4IF EQU 04H

ORG 0000H
LJMP MAIN
ORG 00A3H
LJMP TM4ISR

TM4ISR:
  CPL P1.0 ;test the port
  ANL AUXINTIF,#NOT T4IF ;clear the symbol of interrupt
  RETI

MAIN:
  MOV SP,#3FH
```
```c
#include "reg51.h"
#include "intrins.h"

sfr T4L = 0xd3;
sfr T4H = 0xd2;
sfr T4T3M = 0xd1;
sfr IE2 = 0xaf;
#define ET4 0x40
sfr AUXINTIF = 0xef;
#define T4IF 0x04

sbit P10 = P1^0;

void TM4_Isr() interrupt 20
{
    P10 = !P10; //test the port
    AUXINTIF &= ~T4IF; //clear the symbol of interrupt
}

void main()
{
    T4L = 0x66; //65536-11.0592M/12/1000
    T4H = 0xfc;
    T4T3M = 0x80; //start timer
    IE2 = ET4; //enable timer interrupt
    EA = 1;

    while (1);
}
```

12.5.13 UART1 interrupt

Assembly code

```assembly
ORG 0000H
LJMP MAIN
ORG 0023H
LJMP UART1ISR
```

```
T2L    DATA  0D7H
T2H    DATA  0D6H
AUXR   DATA  8EH

ORG    0000H
LJMP   MAIN
ORG    0023H
LJMP   UART1ISR
```
ORG 0100H

UART1ISR:

JNB TI, CHECKRI
CLR TI ; clear the symbol of interrupt
CPL P1.0 ; test the port

CHECKRI:

JNB RI, ISREXIT
CLR RI ; clear the symbol of interrupt
CPL P1.1 ; test the port

ISREXIT:

RETI

MAIN:

MOV SP,#3FH
MOV SCON,#50H
MOV T2L,#0E8H ; 65536-11059200/115200/4=0FFE8H
MOV T2H,#0FFH
MOV AUXR,#15H ; start timer
SETB ES ; enable serial port interrupt
SETB EA
MOV SBUF,#5AH ; send the date of test
JMP $  
END

C code

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

sfr T2L = 0xd7;
sfr T2H = 0xd6;
sfr AUXR = 0x8e;
sbit P10 = P1^0;
sbit P11 = P1^1;

void UART1_Isr() interrupt 4
{
    if (TI)
    {
        TI = 0; // clear the symbol of interrupt
        P10 = !P10; // test the port
    }
    if (RI)
    {
        RI = 0; // clear the symbol of interrupt
        P11 = !P11; // test the port
    }
}

void main()
{
    SCON = 0x50;
}
\begin{verbatim}
T2L = 0xe8;    //65536-11059200/115200/4=0FFE8H
T2H = 0xff;
AUXR = 0x15;  //start timer
ES = 1;       //enable serial port interrupt
EA = 1;
SBUF = 0x5a;  //send the date of test
while (1);
\end{verbatim}

12.5.14 UART2 interrupt

Assembly code

\begin{verbatim}
T2L DATA 0D7H
T2H DATA 0D6H
AUXR DATA 8EH
S2CON DATA 9AH
S2BUF DATA 9BH
IE2 DATA 0AFH
ES2 EQU 01H

ORG 0000H
LJMP MAIN

ORG 0043H
LJMP UART2ISR

ORG 0100H
UART2ISR:
PUSH ACC
PUSH PSW
MOV A,S2CON
JNB ACC.1,CHECKRI
ANL S2CON,#NOT 02H ;clear the symbol of interrupt
CPL P1.2 ;test the port
CHECKRI:
MOV A,S2CON
JNB ACC.0,ISREXIT
ANL S2CON,#NOT 01H ;clear the symbol of interrupt
CPL P1.3 ;test the port
ISREXIT:
POP PSW
POP ACC
RETI

MAIN:
MOV SP,#3FH

MOV S2CON,#10H
MOV T2L,#0E8H ;65536-11059200/115200/4=0FFE8H
MOV T2H,#0FFH
MOV AUXR,#14H ;start timer
MOV IE2,#ES2 ;enable serial port interrupt
SETB EA
MOV S2BUF,#5AH ;send the date of test
\end{verbatim}
C code

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

sfr T2L = 0xd7;
sfr T2H = 0xd6;
sfr AUXR = 0x8e;
sfr S2CON = 0x9a;
sfr S2BUF = 0x9b;
sfr IE2 = 0xaf;
#define ES2  0x01
sbit P12 = P1^2;
sbit P13 = P1^3;

void UART2_Isr() interrupt 8
{
    if (S2CON & 0x02)
    {
        S2CON &= ~0x02; //clear the symbol of interrupt
        P12 = !P12; //test the port
    }
    if (S2CON & 0x01)
    {
        S2CON &= ~0x01; //clear the symbol of interrupt
        P13 = !P13; //test the port
    }
}

void main()
{
    S2CON = 0x10; //65536-11059200/115200/4=0FFE8H
    T2L = 0xe8; //start timer
    T2H = 0xff;
    AUXR = 0x14; //enable serial port interrupt
    IE2 = ES2;
    EA = 1;
    S2BUF = 0x5a; //send the date of test

    while (1);
}
```

12.5.15 UART3 interrupt

Assembly code

```
T2L DATA 0D7H
T2H DATA 0D6H
AUXR DATA 8EH
S3CON DATA 0ACH
```
ORG 0000H
LJMP MAIN

ORG 008BH
LJMP UART3ISR

ORG 0100H

UART3ISR:
PUSH ACC
PUSH PSW
MOV A, S3CON
JNB ACC.1, CHECKRI
ANL S3CON, #NOT 02H ; clear the symbol of interrupt
CPL P1.0 ; test the port
CHECKRI:
MOV A, S3CON
JNB ACC.0, ISREXIT
ANL S3CON, #NOT 01H ; clear the symbol of interrupt
CPL P1.1 ; test the port
ISREXIT:
POP PSW
POP ACC
RETI

MAIN:
MOV SP, #3FH
MOV S3CON, #10H
MOV T2L, #0E8H ; 65536-11059200/115200/4=0FFE8H
MOV T2H, #0FFH
MOV AUXR, #14H ; start timer
MOV IE2, #ES3 ; enable serial port interrupt
SETH EA
MOV S3BUF, #5AH ; send the date of test
JMP $
void UART3_Isr() interrupt 17
{
    if (S3CON & 0x02) //clear the symbol of interrupt
    {
        S3CON &= ~0x02; //clear the symbol of interrupt
        P10 = !P10; //test the port
    }
    if (S3CON & 0x01) //clear the symbol of interrupt
    {
        S3CON &= ~0x01; //clear the symbol of interrupt
        P11 = !P11; //test the port
    }
}

void main()
{
    S3CON = 0x10; //65536-11059200/115200/4=0FFE8H
    T2L = 0xe8; //65536-11059200/115200/4=0FFE8H
    T2H = 0xff;
    AUXR = 0x14; //start timer
    IE2 = ES3; //enable serial port interrupt
    EA = 1;
    S3BUF = 0x5a; //send the date of test
    while (1);
}

12.5.16 UART4 interrupt

Assembly code

| T2L  | DATA    | 0D7H |
| T2H  | DATA    | 0D6H |
| AUXR | DATA    | 8EH  |
| S4CON| DATA    | 084H |
| S4BUF| DATA    | 085H |
| IE2  | DATA    | 0AFH |
| ES4  | EQU     | 10H  |

| ORG  | 0000H   |
| LJMP | MAIN    |
| ORG  | 0093H   |
| LJMP | UART4ISR|
| ORG  | 0100H   |

UART4ISR:
PUSH ACC
PUSH PSW
MOV AS4CON
JNB ACC.1, CHECKRI
ANL S4CON,#NOT 02H ;clear the symbol of interrupt
CPL P1.0 ;test the port

CHECKRI:
MOV AS4CON
JNB ACC.0, ISREXIT
ANL  S4CON,#NOT 01H ;clear the symbol of interrupt
CPL  P1.1 ;test the port

ISREXIT:
POP  PSW
POP  ACC
RETI

MAIN:
MOV  SP,#3FH
MOV  S4CON,#10H
MOV  T2L,#0E8H ;65536-11059200/115200/4=0FFE8H
MOV  T2H,#0FFH
MOV  AUXR,#14H ;start timer
SETB  EA
MOV  S4BUF,#5AH ;send the date of test
JMP  $  

END

C code

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

sfr  T2L  = 0xd7;
sfr  T2H  = 0xd6;
sfr  AUXR  = 0x8e;
sfr  S4CON  = 0x84;
sfr  S4BUF  = 0x85;
sfr  IE2  = 0xaf;
#define ES4  0x10

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

void UART4_Isr() interrupt 18
{
  if (S4CON & 0x02)
  {
      S4CON &= ~0x02; //clear the symbol of interrupt
      P10 = !P10; //test the port
  }
  if (S4CON & 0x01)
  {
      S4CON &= ~0x01; //clear the symbol of interrupt
      P11 = !P11; //test the port
  }
}

void main()
{
  S4CON = 0x10;
  T2L = 0xe8;  //65536-11059200/115200/4=0FFE8H
  T2H = 0xff;
}
AUXR = 0x14; //start timer
IE2 = ES4; //enable serial port interrupt
EA = 1;
S4BUF = 0x5a; //send the date of test

while (1);

12.5.17 ADC interrupt

Assembly code

<table>
<thead>
<tr>
<th></th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC_CONTR</td>
<td>0bch</td>
</tr>
<tr>
<td>ADC_RES</td>
<td>0bdh</td>
</tr>
<tr>
<td>ADC_RESL</td>
<td>0beh</td>
</tr>
<tr>
<td>ADCCFG</td>
<td>0deh</td>
</tr>
<tr>
<td>EADC</td>
<td>iE.5</td>
</tr>
</tbody>
</table>

ORG 0000H
LJMP MAIN
ORG 002BH
LJMP ADCISR

ORG 0100H
ADCISR:

ANL ADC_CONTR,#NOT 20H ;clear the symbol of interrupt
MOV P0,ADC_RES ;test the port
MOV P2,ADC_RESL ;test the port
RETI

MAIN:

MOV SP,#3FH
MOV ADCCFG,#00H
MOV ADC_CONTR,#0c0h ;start and enable the module of ADC
SETB EADC ;enable ADC interrupt
SETB EA

JMP $

END

C code

#include "reg51.h"
#include "intrins.h"
sfr ADC_CONTR = 0xbc;
sfr ADC_RES = 0xbd;
sfr ADC_RESL = 0xbe;
sfr ADCCFG = 0xde;
sbit EADC = IE^5

void ADC_Isr() interrupt 5
{
    ADC_CONTR &= ~0x20; //clear the symbol of interrupt
    P0 = ADC_RES; //test the port
}
P2 = ADC_RESL;  //test the port
}

void main()
{
    ADCCFG = 0x00;
    ADC_CONTR = 0xc0;  //start and enable the module of ADC
    EADC = 1;  //enable ADC interrupt
    EA = 1;

    while (1);
}

12.5.18  LVD interrupt

Assembly code

<table>
<thead>
<tr>
<th>RSTCFG</th>
<th>DATA</th>
<th>0FFH</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENLVR</td>
<td>EQU</td>
<td>40H</td>
</tr>
<tr>
<td>LVD2V2</td>
<td>EQU</td>
<td>00H</td>
</tr>
<tr>
<td>LVD2V4</td>
<td>EQU</td>
<td>01H</td>
</tr>
<tr>
<td>LVD2V7</td>
<td>EQU</td>
<td>02H</td>
</tr>
<tr>
<td>LVD3V0</td>
<td>EQU</td>
<td>03H</td>
</tr>
<tr>
<td>ELVD</td>
<td>BIT</td>
<td>1E6</td>
</tr>
<tr>
<td>LVDF</td>
<td>EQU</td>
<td>20H</td>
</tr>
<tr>
<td>RSTCFG</td>
<td>DATA</td>
<td>0FFH</td>
</tr>
<tr>
<td>ENLVR</td>
<td>EQU</td>
<td>40H</td>
</tr>
<tr>
<td>LVD2V2</td>
<td>EQU</td>
<td>00H</td>
</tr>
<tr>
<td>LVD2V4</td>
<td>EQU</td>
<td>01H</td>
</tr>
<tr>
<td>LVD2V7</td>
<td>EQU</td>
<td>02H</td>
</tr>
<tr>
<td>LVD3V0</td>
<td>EQU</td>
<td>03H</td>
</tr>
<tr>
<td>ELVD</td>
<td>BIT</td>
<td>1E6</td>
</tr>
<tr>
<td>LVDF</td>
<td>EQU</td>
<td>20H</td>
</tr>
<tr>
<td>ORG</td>
<td>0000H</td>
<td></td>
</tr>
<tr>
<td>LJMP</td>
<td>MAIN</td>
<td></td>
</tr>
<tr>
<td>ORG</td>
<td>0033H</td>
<td></td>
</tr>
<tr>
<td>LJMP</td>
<td>LVDISR</td>
<td></td>
</tr>
</tbody>
</table>

LVDISR:

| ANL       | PCON,#NOT LVDF ;clear the symbol of interrupt |
| CPL       | P1.0   ;test the port                          |
| RETI      |                                                 |

MAIN:

| MOV       | SP,#3FH |
| ANL       | PCON,#NOT LVDF ;clear the symbol of interrupt |
| MOV       | RSTCFG# LVD3V0 |
| SETB      | ELVD    ;enable LVD interrupt                  |
| SETB      | EA      |
| JMP       | S       |

END

C code

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

sfr RSTCFG = 0xff;  //RSTCFG6
#define ENLVR 0x40  //RSTCFG6
#define LVD2V2 0x00  //LVD@2.2V
#define LVD2V4 0x01  //LVD@2.4V
#define LVD2V7 0x02 //LVD@2.7V
#define LVD3V0 0x03 //LVD@3.0V
sbit ELVD = IE^6;
#define LVDF 0x20 //PCON.5
sbit P10 = P1^0;

void LVD_Isr() interrupt 6
{
    PCON &= ~LVDF; //clear the symbol of interrupt
    P10 = !P10; //test the port
}

void main()
{
    PCON &= ~LVDF; //clear the symbol of interrupt
    RSTCFG = LVD3V0; //
    ELVD = 1; //enable LVD interrupt
    EA = 1;

    while (1);
}

12.5.19   PCA interrupt

Assembly code

CON  DATA  0D8H
CF   BIT    CCON.7
CR   BIT    CCON.6
CCF3 BIT    CCON.3
CCF2 BIT    CCON.2
CCF1 BIT    CCON.1
CCF0 BIT    CCON.0
CMOD DATA  0D9H
CL   DATA  0E9H
CH   DATA  0F9H
CCAPM0 DATA  0DAH
CCAP0L DATA  0EAH
CCAP0H DATA  0FAH
PCA_PWM0 DATA  0F2H
CCAPM1 DATA  0DBH
CCAP1L DATA  0EBH
CCAP1H DATA  0FBH
PCA_PWM1 DATA  0F3H
CCAPM2 DATA  0DCH
CCAP2L DATA  0ECH
CCAP2H DATA  0FCH
PCA_PWM2 DATA  0F4H
CCAPM3 DATA  0DDH
CCAP3L DATA  0EDH
CCAP3H DATA  0FDH
PCA_PWM3 DATA  0F5H

ORG   0000H
LJMP  MAIN
ORG   003BH

LJMP  PCAISR

ORG  0100H

PCAISR:
JNB   CF,ISREXIT
CLR   CF    ;clear the symbol of interrupt
CPL   P1.0   ;test the port

ISREXIT:
RETI

MAIN:
MOV   SP,#3FH
MOV   CCON,#00H
MOV   CMOD,#09H
SETB  CR
SETB  EA
JMP   $  

END

C code

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

sfr  CCON     = 0xd8;
sbit  CF       = CCON^7;
sbit  CR       = CCON^6;
sbit  CCF3     = CCON^3;
sbit  CCF2     = CCON^2;
sbit  CCF1     = CCON^1;
sbit  CCF0     = CCON^0;
sfr  CMOD     = 0xd9;
sfr  CL       = 0xe9;
sfr  CH       = 0xfb;
sfr  CCAPM0   = 0xda;
sfr  CCAP0L   = 0xe9;
sfr  CCAP0H   = 0xfa;
sfr  PCA_PWM0 = 0xf2;
sfr  CCAPM1   = 0xe9;
sfr  CCAP1L   = 0xe9;
sfr  CCAP1H   = 0xfb;
sfr  PCA_PWM1 = 0xfb;
sfr  CCAPM2   = 0xdc;
sfr  CCAP2L   = 0xe9;
sfr  CCAP2H   = 0xfb;
sfr  PCA_PWM2 = 0xfc;
sfr  CCAPM3   = 0xdc;
sfr  CCAP3L   = 0xe9;
sfr  CCAP3H   = 0xfd;
sfr  PCA_PWM3 = 0xfb;
sbit  P10     = P1^0;

void PCA_Isr() interrupt 7
```c

void main()
{
    CMOD = 0x09;
    CCON = 0x00;
    CR = 1;
    EA = 1;

    while (1);
}

12.5.20 SPI interrupt

Assembly code

| SPSTAT   | DATA  | 0CDH |
| SPCTL    | DATA  | 0CEH |
| SPDAT    | DATA  | 0CFH |
| IE2      | DATA  | 0AFH |
| ESPI     | EQU   | 02H  |
|          | ORG   | 0000H|
|          | LJMP  | MAIN |
|          | ORG   | 004BH|
|          | LJMP  | SPIISR|
|          | ORG   | 0100H|

SPIISR:

| MOV      | SPSTAT,#0C0H ;clear the symbol of interrupt |
| CPL      | P1.0 ;test the port |
| RETI     | |

MAIN:

| MOV      | SP,#3FH |
| MOV      | SPCTL,#50H ;enable SPI |
| MOV      | SPSTAT,#0C0H ;clear the symbol of interrupt |
| MOV      | IE2,#ESPI ;enable SPI interrupt |
| SETB     | EA |
| MOV      | SPDAT,#5AH ;send the date of test |
| JMP      | $ |

END

C code

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

```
void SPI_Isr() interrupt 9
{
    SPSTAT = 0xc0; //clear the symbol of interrupt
    P10 = !P10; //test the port
}

void main()
{
    SPCTL = 0x50;
    SPSTAT = 0xc0; //clear the symbol of interrupt
    IE2 = ESPI; //enable SPI interrupt
    EA = 1;
    SPDAT = 0x5a; //send the date of test

    while (1);
}

12.5.21 CMP interrupt

Assembly code

<table>
<thead>
<tr>
<th>CMPCR1</th>
<th>CMPCR2</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA</td>
<td>DATA</td>
</tr>
<tr>
<td>0E6H</td>
<td>0E7H</td>
</tr>
</tbody>
</table>

ORG 0000H
LJMP MAIN
ORG 00ABH
LJMP CMPISR

ORG 0100H

CMPISR:
ANL CMPCR1,#NOT 40H ;clear the symbol of interrupt
CPL P1.0 ;test the port
RETI

MAIN:
MOV SP,#3FH
MOV CMPCR2,#00H
MOV CMPCR1,#80H
ORL CMPCR1,#30H
ANL CMPCR1,#NOT 08H
ORL CMPCR1,#04H
ORL CMPCR1,#02H
SETB EA

JMP $
C code

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

sfr CMPCR1 = 0xe6;
sfr CMPCR2 = 0xe7;
sbit P10 = P1^0;

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

void main()
{
    CMPCR2 = 0x00;
    CMPCR1 = 0x80;
    CMPCR1 |= 0x30;
    CMPCR1 &= ~0x08;
    CMPCR1 |= 0x04;
    CMPCR1 |= 0x02;
    EA = 1;

    while (1);
}
```

Assembly code

```assembly
P_SW2 DATA 0BAH
PWMCFG DATA 0F1H
PWMIF DATA 0F6H
PWMFDCR DATA 0F7H
PWMCR DATA 0FEH
PWMCH XDATA 0FFF0H
PWMCL XDATA 0FFF1H
PWMCKS XDATA 0FFF2H
TADCPH XDATA 0FFF3H
TADCPL XDATA 0FFF4H
PWM0T1H XDATA 0FF00H
PWM0T1L XDATA 0FF01H
PWM0T2H XDATA 0FF02H
PWM0T2L XDATA 0FF03H
PWM0CR XDATA 0FF04H
PWM0HLD XDATA 0FF05H
PWM1T1H XDATA 0FF10H
PWM1T1L XDATA 0FF11H
PWM1T2H XDATA 0FF12H
PWM1T2L XDATA 0FF13H
PWM1CR XDATA 0FF14H
```
PWM1HLD XDATA 0FF15H
PWM2T1H XDATA 0FF20H
PWM2T1L XDATA 0FF21H
PWM2T2H XDATA 0FF22H
PWM2T2L XDATA 0FF23H
PWM2CR XDATA 0FF24H
PWM2HLD XDATA 0FF25H
PWM3T1H XDATA 0FF30H
PWM3T1L XDATA 0FF31H
PWM3T2H XDATA 0FF32H
PWM3T2L XDATA 0FF33H
PWM3CR XDATA 0FF34H
PWM3HLD XDATA 0FF35H
PWM4T1H XDATA 0FF40H
PWM4T1L XDATA 0FF41H
PWM4T2H XDATA 0FF42H
PWM4T2L XDATA 0FF43H
PWM4CR XDATA 0FF44H
PWM4HLD XDATA 0FF45H
PWM5T1H XDATA 0FF50H
PWM5T1L XDATA 0FF51H
PWM5T2H XDATA 0FF52H
PWM5T2L XDATA 0FF53H
PWM5CR XDATA 0FF54H
PWM5HLD XDATA 0FF55H
PWM6T1H XDATA 0FF60H
PWM6T1L XDATA 0FF61H
PWM6T2H XDATA 0FF62H
PWM6T2L XDATA 0FF63H
PWM6CR XDATA 0FF64H
PWM6HLD XDATA 0FF65H
PWM7T1H XDATA 0FF70H
PWM7T1L XDATA 0FF71H
PWM7T2H XDATA 0FF72H
PWM7T2L XDATA 0FF73H
PWM7CR XDATA 0FF74H
PWM7HLD XDATA 0FF75H

ORG 0000H
LJMP MAIN
ORG 00B3H
LJMP PWMISR
ORG 00BBH
LJMP PWMFDISR

ORG 0100H

PWMISR:
PUSH ACC
PUSH PSW
MOV A,PWMCFG
JNB ACC.7,ISREXIT
ANL PWMCFG,#NOT 80H ;clear the symbol of interrupt
CPL P1.0 ;test the port
ISREXIT:
POP PSW
POP ACC
RETI
C code

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

sfr P_SW2 = 0xba;
sfr PWMCFG = 0xf1;
sfr PWMIF = 0xf6;
sfr PWMFDCR = 0xf7;
sfr PWMCR = 0xfe;
#define PWMC  (*(unsigned int volatile xdata *)0xfff0)
#define PWMCKS  (*(unsigned char volatile xdata *)0xfff2)
#define TADCP  (*(unsigned int volatile xdata *)0xfff3)
#define PWM0T1  (*(unsigned int volatile xdata *)0xff00)
#define PWM0T2  (*(unsigned int volatile xdata *)0xff02)
#define PWM0CR  (*(unsigned char volatile xdata *)0xff04)
#define PWM0HLD  (*(unsigned char volatile xdata *)0xff05)
#define PWM1T1  (*(unsigned int volatile xdata *)0xff10)
#define PWM1T2  (*(unsigned int volatile xdata *)0xff12)
#define PWM1CR  (*(unsigned char volatile xdata *)0xff14)
#define PWM1HLD  (*(unsigned char volatile xdata *)0xff15)
#define PWM2T1  (*(unsigned int volatile xdata *)0xff20)
#define PWM2T2  (*(unsigned int volatile xdata *)0xff22)
#define PWM2CR  (*(unsigned char volatile xdata *)0xff24)
#define PWM2HLD  (*(unsigned char volatile xdata *)0xff25)
#define PWM3T1  (*(unsigned int volatile xdata *)0xff30)
#define PWM3T2  (*(unsigned int volatile xdata *)0xff32)
#define PWM3CR  (*(unsigned char volatile xdata *)0xff34)

PWMFDISR:

ANL PWMFDCR,#NOT 01H ;clear the symbol of interrupt
CPL P1.1 ;test the port
RETI

MAIN:

MOV SP,#3FH
MOV P_SW2,#80H
MOV A,#0FH
MOV DPTR,#PWMCKS
MOVX @DPTR,A
MOV A,#01H
MOV DPTR,#PWMCH
MOVX @DPTR,A
MOV A,#00H
MOV DPTR,#PWMCL
MOVX @DPTR,A
MOV P_SW2,#00H
MOV PWMFDCR,#7AH
MOV PWMCR,#0C0H
SETB EA
JMP $
#define PWM3HLD  (*(unsigned char volatile xdata *)0xff35)
#define PWM4T1  (*(unsigned int volatile xdata *)0xff40)
#define PWM4CR  (*(unsigned char volatile xdata *)0xff44)
#define PWM4HLD  (*(unsigned char volatile xdata *)0xff45)
#define PWM5T1  (*(unsigned int volatile xdata *)0xff50)
#define PWM5T2  (*(unsigned int volatile xdata *)0xff52)
#define PWM5CR  (*(unsigned char volatile xdata *)0xff54)
#define PWM5HLD  (*(unsigned char volatile xdata *)0xff55)
#define PWM6T1  (*(unsigned int volatile xdata *)0xff60)
#define PWM6T2  (*(unsigned int volatile xdata *)0xff62)
#define PWM6CR  (*(unsigned char volatile xdata *)0xff64)
#define PWM6HLD  (*(unsigned char volatile xdata *)0xff65)
#define PWM7T1  (*(unsigned int volatile xdata *)0xff70)
#define PWM7T2  (*(unsigned int volatile xdata *)0xff72)
#define PWM7CR  (*(unsigned char volatile xdata *)0xff74)
#define PWM7HLD  (*(unsigned char volatile xdata *)0xff75)

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

void PWM_Isr() interrupt 22
{
    if (PWMCFG & 0x80)
    {
        PWMCFG &= ~0x80; //clear the symbol of interrupt
        P10 = !P10; //test the port
    }
}

void PWMFD_Isr() interrupt 23
{
    PWMFDCR &= ~0x01; //clear the symbol of interrupt
    P11 = !P11; //test the port
}

void main()
{
    P_SW2 = 0x80;
P_WMCKS = 0x0f;
P_WMC = 0x0100;
P_SW2 = 0x00;

    PWMFDCR = 0x7a;
P_WMCR = 0xc0;
    EA = 1;

    while (1);
}

12.5.23 I2C interrupt

Assembly code

<table>
<thead>
<tr>
<th>P_SW2</th>
<th>DATA</th>
<th>0BAH</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2CCFG</td>
<td>XDATA</td>
<td>0FE80H</td>
</tr>
</tbody>
</table>
I2CMSCR  XDATA 0FE81H
I2CMSST  XDATA 0FE82H
I2CSLCR  XDATA 0FE83H
I2CSLST  XDATA 0FE84H
I2CSLADR  XDATA 0FE85H
I2CTXD  XDATA 0FE86H
I2CRXD  XDATA 0FE87H

ORG 0000H
LJMP MAIN
ORG 00C3H
LJMP I2CISR

I2CISR:
PUSH ACC
PUSH DPL
PUSH DPH
PUSH P_SW2
MOV P_SW2,#80H
MOV DPTR,#I2CMSST
MOVX A,@DPTR
ANL A,#NOT 40H ;clear the symbol of interrupt
MOVX @DPTR,A
CPL P1.0 ;test the port
POP P_SW2
POP DPH
POP DPL
POP ACC
RETI

MAIN:
MOV SP,#3FH
MOV P_SW2,#80H
MOV A,#0C0H ;enable I2C
MOV DPTR,#I2CCFG
MOVX @DPTR,A
MOV A,#80H ;enable I2C interrupt
MOV DPTR,#I2CMSST
MOVX @DPTR,A
MOV A,#80H
MOV DPTR,#I2CMSRC
MOVX @DPTR,A
MOV A,#00H
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
SETB EA

MOV P_SW2,#80H
MOV A,#081H
MOV DPTR,#I2CMSRC
MOVX @DPTR,A
MOV P_SW2,#00H
JMP $

END

C code
#include "reg51.h"
#include "intrins.h"


```c
sfr P_SW2 = 0xba;
#define I2CCFG (*(unsigned char volatile xdata *)0xfe80)
#define I2CMSCR (*(unsigned char volatile xdata *)0xfe81)
#define I2CMSST (*(unsigned char volatile xdata *)0xfe82)
#define I2CSLCR (*(unsigned char volatile xdata *)0xfe83)
#define I2CSLST (*(unsigned char volatile xdata *)0xfe84)
#define I2CSDLADR (*(unsigned char volatile xdata *)0xfe85)
#define I2CTXD (*(unsigned char volatile xdata *)0xfe86)
#define I2CRXD (*(unsigned char volatile xdata *)0xfe87)

sbit P10 = P1^0;

void I2C_Isr() interrupt 24
{
    _push_(P_SW2);
    P_SW2 |= 0x80;
    if (I2CMSST & 0x40)
    {
        I2CMSST &= ~0x40; //clear the symbol of interrupt
        P10 = !P10; //test the port
    }
    _pop_(P_SW2);
}

void main()
{
    P_SW2 = 0x80;
    I2CCFG = 0xc0; //enable I2C
    I2CMSCR = 0x80; //enable I2C interrupt
    P_SW2 = 0x00;
    EA = 1;
    P_SW2 = 0x80;
    I2CMSCR = 0x81;
    P_SW2 = 0x00;

    while (1);
}
```
13 Timer/Counter

Five 16-bit Timer/Counters are integrated in STC8F family microcontrollers: T0, T1, T2, T3 and T4. All of them can be used as Timer or Counter. For T0 and T1, the "Timer" or "Counter" function is selected by the control bits C/T in the Special Function Register TMOD. For T2, the "Timer" or "Counter" function is selected by the control bit T2_C/T in the Special Function Register AUXR. For T3, the "Timer" or "Counter" function is selected by the control bit T3_C/T in the Special Function Register T4T3M. For T4, the "Timer" or "Counter" function is selected by the control bit T4_C/T in the Special Function Register T4T3M. The core of the timer / counter is an addition counter, the essence of which is counting pulses. The only difference of "Timer" mode and "Counter" mode is the different counting pulses sources. If the counting pulse is from the system clock, the timer/counter runs in the timing mode, it counts once every 12 clocks or one clock. If the counting pulse is from the microcontroller external reference pins (T0 is P3.4, T1 is P3.5, T2 is P1.2, T3 is P0.4, T4 is P0.6), the timer/counter runs in counting mode, it counts once every pulse.

When timer/counters T0, T1 and T2 are operating in "Timer" mode, T0x12, T1x12 and T2x12 in AUXR register are used to determine the clocks of T0, T1 and T2 are System Clock/12 or System Clock/1. When timer/ counters T3 and T4 are operating in "Timer" mode, T3x12 and T4x12 in the T4T3M Special Function Register determine the clocks of T3 and T4 are System Clock/12 or System Clock/1. When the timer/counters are operating in "Counter" mode, the frequency of the external pulse is not divided.

Timer/Counter 0 has four operating modes which are selected by bit-pairs (M1, M0) in TMOD. The four modes are mode 0 (16-bit auto-reload mode), mode 1 (16-bit non-auto-reload mode), mode 2 (8-bit auto-reload mode) and mode 3 (16-bit auto-reload mode whose interrupt can not be disabled). And for Timer/Counter 1, all modes except mode 3 are the same as Timer/Counter 0. The mode 3 of Timer/Counter 1 is invalid and stops counting. For T2, T3 and T4, they only have one mode: 16-bit auto-reload mode. Besides being used as Timer/Counters, T2, T3 and T4 also can be as the baud-rate generators of serial ports and programmable clock outputs.

### 13.1 Timer Related Registers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
<th>Value after reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCON</td>
<td>Timer 0 and 1 control register</td>
<td>88H</td>
<td>TF1</td>
<td>TR1</td>
<td>TF0</td>
<td>TR0</td>
<td>IE1</td>
<td>T1</td>
<td>IE0</td>
<td>T0</td>
<td>0000,0000</td>
</tr>
<tr>
<td>TMOD</td>
<td>Timer 0 and 1 mode</td>
<td>89H</td>
<td>GATE</td>
<td>C/T</td>
<td>M1</td>
<td>M0</td>
<td>GATE</td>
<td>C/T</td>
<td>M1</td>
<td>M0</td>
<td>0000,0000</td>
</tr>
<tr>
<td>TL0</td>
<td>Timer 0 low byte</td>
<td>8AH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>TL1</td>
<td>Timer 1 low byte</td>
<td>8BH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>TH0</td>
<td>Timer 0 high byte</td>
<td>8CH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>TH1</td>
<td>Timer 1 high byte</td>
<td>8DH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>AUXR</td>
<td>Auxiliary register 1</td>
<td>8EH</td>
<td>T0x12</td>
<td>T1x12</td>
<td>UART_M0x6</td>
<td>T2R</td>
<td>T2_C/T</td>
<td>T2x12</td>
<td>EXTRAM</td>
<td>S1ST2</td>
<td>0000,0000</td>
</tr>
<tr>
<td>INTCLOKO</td>
<td>interrupt and clock output control</td>
<td>8FH</td>
<td>-</td>
<td>EX4</td>
<td>EX3</td>
<td>EX2</td>
<td>-</td>
<td>t2CLKO</td>
<td>t1CLKO</td>
<td>t0CLKO</td>
<td>x000,x000</td>
</tr>
</tbody>
</table>
### Wake-up Timer Control register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>WKTCL</td>
<td>AAH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WKTCH</td>
<td>ABH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>WKTEN</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Wake-up Timer Control register

- **WKTCL**: Wake-up Timer Control register
- **WKTCH**: Wake-up Timer Control register

#### Timer4 and Timer 3 mode register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>T4T3M</td>
<td>D1H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>T4R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>T4_C/T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>T4x12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>T4CLKO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>T3R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>T3_C/T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>T3x12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>T3CLKO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Timer 4 high byte

- **D2H**: 0000,0000

#### Timer 4 low byte

- **D3H**: 0000,0000

#### Timer 3 high byte

- **D4H**: 0000,0000

#### Timer 3 low byte

- **D5H**: 0000,0000

#### Timer 2 high byte

- **D6H**: 0000,0000

#### Timer 2 low byte

- **D7H**: 0000,0000

---

### 13.2 Timer 0/1

#### Timer 0 and 1 control register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCON</td>
<td>88H</td>
<td>TR1</td>
<td>TR0</td>
<td>TF0</td>
<td>TF1</td>
<td>IE1</td>
<td>IT1</td>
<td>IE0</td>
<td>IT0</td>
</tr>
</tbody>
</table>

- **TF1**: Timer/Counter 1 Overflow Flag. After T1 is enabled to count, it performs adding 1 count from the initial value. TF1 is set by hardware on Timer/Counter 1 overflow and requests interrupt to CPU. It will keep the status until CPU responds the interrupt and is cleared by hardware automatically. It also can be cleared by software.

- **TR1**: Timer/Counter 1 run control bit. It is set or cleared by software to turn Timer/Counter on/off. When GATE (TMOD.7) = 0, T1 will start counting as soon as TR1=1 and stop counting when TR1 = 0. If GATE (TMOD.7) = 1, T1 count is enabled only if TR1 = 1 and INT1 is high.

- **TF0**: Timer/Counter 0 Overflow Flag. After T0 is enabled to count, it performs adding 1 count from the initial value. TF0 is set by hardware on Timer/Counter 1 overflow and requests interrupt to CPU. It will keep the status until CPU responds the interrupt and is cleared by hardware automatically. It also can be cleared by software.

- **TR0**: Timer/Counter 0 run control bit. It is set or cleared by software to turn Timer/Counter on/off. When GATE (TMOD.3) = 0, T0 will start counting as soon as TR0=1 and stop counting when TR0 = 0. If GATE (TMOD.3) = 1, T0 count is enabled only if TR0 = 1 and INT0 is high.

- **IE1**: External interrupt 1 (INT1/P3.3) request flag. It is set by hardware when external interrupt rising or falling edge defined by IT1 is detected. The flag can be cleared by software but is automatically cleared when the external interrupt 1 service routine has been processed.

- **IT1**: External Intenupt 1 edge trigger type select bit. It is set/cleared by software to specify rising / falling edges triggered external interrupt 1. If IT1 = 0, INT1 is triggered by both rising and falling edges. If IT1 = 1, INT1 is triggered only by falling edge.
IE0: External interrupt 0 (INT0/P3.2) request flag. It is set by hardware when external interrupt rising or falling edge defined by IT0 is detected. The flag can be cleared by software but is automatically cleared when the external interrupt 0 service routine has been processed.

IT0: External Interrupt 0 edge trigger type select bit. If IT0 = 0, INT0 is triggered by both rising and falling edges. If IT0 = 1, INT0 is triggered only by falling edge.

**Timer 0/1 mode register**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TMOD</td>
<td>89H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1_GATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1_C/T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1_M1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T1_M0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T0_GATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T0_C/T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T0_M1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T0_M0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

T1_GATE: Timer/Counter 1 gate control. If GATE/TMOD.7 = 1, Timer/Counter 1 enabled only when TR1 is set AND INT1 pin is high.

T0_GATE: Timer/Counter 0 gate control. If GATE/TMOD.3 = 1, Timer/Counter 0 enabled only when TR0 is set AND INTO pin is high.

T1_C/T: Timer/Counter 1 mode select bit.

If C/T / TMOD.6 = 0, Timer/Counter 1 is used as Timer (input pulse is from internal system clock);
If C/T / TMOD.6 = 1, Timer/Counter 1 is used as Counter (input pulse is from external T1/P3.5 pin).

T0_C/T: Timer/Counter 0 mode select bit.

If C/T / TMOD.2 = 0, Timer/Counter 0 is used as Timer (input pulse is from internal system clock);
If C/T / TMOD.2 = 1, Timer/Counter 0 is used as Counter (input pulse is from external T0/P3.4 pin).

**T1_M1/T1_M0:** Timer 1 mode select bits.

<table>
<thead>
<tr>
<th>T1_M1</th>
<th>T1_M0</th>
<th>Timer/Counter 1 operating mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>16-bit auto-reload mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When the 16-bit counter [TH1, TL1] overflows, the system automatically loads the reload value in the internal 16-bit reload register into [TH1, TL1].</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>16-bit non-auto-reload mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When the 16-bit counter [TH1, TL1] overflows, timer 1 will count from 0.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>8-bit auto-reload mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When the 8-bit counter TL1 overflows, the system automatically loads the reload value in TH1 into TL1.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>T1 is stoped.</td>
</tr>
</tbody>
</table>

**T0_M1/T0_M0:** Timer 0 mode select bits.

<table>
<thead>
<tr>
<th>T0_M1</th>
<th>T0_M0</th>
<th>Timer/Counter 0 operating mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>16-bit auto-reload mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When the 16-bit counter [TH0, TL0] overflows, the system automatically loads the reload value in the internal 16-bit reload register into [TH0, TL0].</td>
</tr>
</tbody>
</table>
16-bit non-auto-reload mode.

0 1  When the 16-bit counter [TH1, TL1] overflows, timer 1 will count from 0.

8-bit auto-reload mode.

1 0  When the 8-bit counter TL0 overflows, the system automatically loads the reload value in TH0 into TL0.

1 1  16-bit auto-reload mode. It is similar to mode 0, whose interrupt can not be disabled.

### Timer 0 counting register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>TL0</td>
<td>8AH</td>
</tr>
<tr>
<td>TH0</td>
<td>8CH</td>
</tr>
</tbody>
</table>

When Timer/Counter 0 is operating in 16-bit mode (Mode 0, Mode 1, Mode 3), TL0 and TH0 combine into a 16-bit register with TL0 as the low byte and TH0 as the high byte. For 8-bit mode (mode 2), TL0 and TH0 are two independent 8-bit registers.

### Timer 1 counting register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>TL1</td>
<td>8BH</td>
</tr>
<tr>
<td>TH1</td>
<td>8DH</td>
</tr>
</tbody>
</table>

When Timer/Counter 1 is operating in 16-bit mode (Mode 0, Mode 1, Mode 3), TL1 and TH1 combine into a 16-bit register with TL1 as the low byte and TH1 as the high byte. For 8-bit mode (mode 2), TL1 and TH1 are two independent 8-bit registers.

### Auxiliary register 1(AUXR)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUXR</td>
<td>8EH</td>
<td>T0x12</td>
<td>T1x12</td>
<td>UART_M0x6</td>
<td>T2R</td>
<td>T2_C/T</td>
<td>T2x12</td>
<td>EXTRAM</td>
<td>S1ST2</td>
</tr>
</tbody>
</table>

T0x12: Timer 0 speed control bit.

0: The clock source of Timer 0 is SYSclk/12.
1: The clock source of Timer 0 is SYSclk/1.

T1x12: Timer 1 speed control bit.

0: The clock source of Timer 1 is SYSclk/12.
1: The clock source of Timer 1 is SYSclk/1.

### Interrupt and clock out control register (INTCLKO)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTCLKO</td>
<td>8FH</td>
<td>-</td>
<td>EX4</td>
<td>EX3</td>
<td>EX2</td>
<td>-</td>
<td>T2CLKO</td>
<td>T1CLKO</td>
<td>T0CLKO</td>
</tr>
</tbody>
</table>

T0CLKO: Timer 0 clock out control bit.

0: Turn off the clock output.
1: P3.5 is configured for Timer0 programmable clock output T0CLKO. When Timer 0 overflows, P3.5...
T1CLKO: Timer 0 clock out control bit.
0: Turn off the clock output.
1: P3.4 is configured for Timer1 programmable clock output T1CLKO. When Timer 1 overflows, P3.4 will flip automatically.

### 13.3 Timer 2

**Auxiliary register 1 (AUXR)**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUXR</td>
<td>8EH</td>
<td>T0x12</td>
<td>T1x12</td>
<td>UART_M0x6</td>
<td>T2R</td>
<td>T2_C/T</td>
<td>T2x12</td>
<td>EXTRAM</td>
<td>S1ST2</td>
</tr>
</tbody>
</table>

TR2: Timer/Counter 2 run control bit.
0: Timer 2 stops counting.
1: Timer 2 start counting.

T2_C/T: Timer/Counter 2 mode select bit.
0: Timer/Counter 2 is used as Timer (input pulse is from internal system clock);
1: Timer/Counter 2 is used as Counter (input pulse is from external T2/P1.2 pin).

T2x12: Timer 2 speed control bit.
0: The clock source of Timer 2 is SYSclk/12.
1: The clock source of Timer 2 is SYSclk/1.

**interrupt and clock out control register (INTCLKO)**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTCLKO</td>
<td>8FH</td>
<td>-</td>
<td>EX4</td>
<td>EX3</td>
<td>EX2</td>
<td>-</td>
<td>T2CLKO</td>
<td>T1CLKO</td>
<td>T0CLKO</td>
</tr>
</tbody>
</table>

T2CLKO: Timer 2 clock out control bit.
0: Turn off the clock output.
1: P1.3 is configured for Timer2 programmable clock output T2CLKO. When Timer 2 overflows, P1.3 will flip automatically.

**Timer 2 counting register**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2L</td>
<td>D7H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2H</td>
<td>D6H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Timer/Counter 2 operates in 16-bit auto-reload mode. T2L and T2H combine into a 16-bit register with T2L as the low byte and T2H as the high byte. When the 16-bit counter [T2H, T2L] overflows, the system automatically loads the reload value in the internal 16-bit reload register into [T2H, T2L].

### 13.4 Timer 3/4

**Timer 4/3 control register (T4T3M)**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
</table>

Timer/Counter 2 operates in 16-bit auto-reload mode. T2L and T2H combine into a 16-bit register with T2L as the low byte and T2H as the high byte. When the 16-bit counter [T2H, T2L] overflows, the system automatically loads the reload value in the internal 16-bit reload register into [T2H, T2L].
T4T3M D1H T4R T4_C/T T4x12 T4CLKO T3R T3_C/T T3x12 T3CLKO

TR4: Timer/Counter 4 run control bit.
0: Timer 4 stops counting.
1: Timer 4 start counting.

T4_C/T: Timer/Counter 4 mode select bit.
0: Timer/Counter 4 is used as Timer (input pulse is from internal system clock);
1: Timer/Counter 4 is used as Counter (input pulse is from external T4/P0.6 pin).

T4x12: Timer 4 speed control bit.
0: The clock source of Timer 4 is SYSclk/12.
1: The clock source of Timer 4 is SYSclk/1.

T4CLKO: Timer 4 clock out control bit.
0: Turn off the clock output.
1: P0.7 is configured for Timer4 programmable clock output T4CLKO. When Timer 4 overflows, P0.7 will flip automatically.

TR3: Timer/Counter 3 run control bit.
0: Timer 3 stops counting.
1: Timer 3 start counting.

T3_C/T: Timer/Counter 3 mode select bit.
0: Timer/Counter 3 is used as Timer (input pulse is from internal system clock);
1: Timer/Counter 3 is used as Counter (input pulse is from external T3/P0.4 pin).

T3x12: Timer 3 speed control bit.
0: The clock source of Timer 3 is SYSclk/12.
1: The clock source of Timer 3 is SYSclk/1.

T3CLKO: Timer 3 clock out control bit.
0: Turn off the clock output.
1: P0.5 is configured for Timer3 programmable clock output T3CLKO. When Timer 3 overflows, P0.5 will flip automatically.

### Timer 3 counting register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>B7</td>
<td>B6</td>
</tr>
<tr>
<td>T3L</td>
<td>D5H</td>
</tr>
<tr>
<td>T3H</td>
<td>D4H</td>
</tr>
</tbody>
</table>

Timer/Counter 3 operates in 16-bit auto-reload mode. T3L and T3H combine into a 16-bit register with T3L as the low byte and T3H as the high byte. When the 16-bit counter [T3H, T3L] overflows, the system automatically loads the reload value in the internal 16-bit reload register into [T3H, T3L].

### Timer 4 counting register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>B7</td>
<td>B6</td>
</tr>
<tr>
<td>T4L</td>
<td>D3H</td>
</tr>
<tr>
<td>T4H</td>
<td>D2H</td>
</tr>
</tbody>
</table>

Timer/Counter 4 operates in 16-bit auto-reload mode. T4L and T4H combine into a 16-bit register with T4L as
the low byte and T4H as the high byte. When the 16-bit counter [T4H, T4L] overflows, the system automatically loads the reload value in the internal 16-bit reload register into [T4H, T4L].

13.5 Power-Down Wake-Up Special Timer

The internal power-down wake-up special Timer consists of a 15-bit timer \{WKTCH[6:0],WKTCL[7:0]\}, which is used to wake MCU in power-down mode.

### Power-down wake-up timer registers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>WKTCL</td>
<td>AAH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WKTCH</td>
<td>ABH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WKTEN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

WKTEN: internal power-down wake-up special Timer enable bit.

- 0: disable internal power-down wake-up special Timer.
- 1: enable internal power-down wake-up special Timer.

If the dedicated power-down wake-up timer integrated in STF8 family microcontrollers is enabled (the WKTEN bit in the WKTCH register is set by software), the wake-up timer starts counting when the MCU enters Power-down mode or stop mode. When the count value is equal to the value set by the user, the dedicated power-down wake-up timer will wake up the MCU. After the MCU wake-up, it will execute the next statement where MCU entered the power-down mode. After the MCU wakes up from power-down mode, we can read the contents of WKTCH and WKTCL to get the MCU sleep in power-down mode.

Please note here that the value written by user in registers \{WKTCH [6: 0], WKTCL [7: 0]\} must be one less than the actual count value. For example, if user needs to count 10 times, then 9 is written into the registers \{WKTCH [6: 0], WKTCL [7: 0]\}. Similarly, if user wants to count 32768 times, 7FFFH (ie 32767) should be written into \{WKTCH [6: 0], WKTCL [7: 0]\}.

Internal power-down wake-up timer has its own internal clock, where the wake-up timer’s count time is determined by the clock. The clock frequency of internal power-down wake-up timer is about 32KHz, of course, the error may be large. The user can read the contents of F8H and F9H in the RAM area (F8H high byte and F9H low byte) to get the clock frequency of the internal power-down wake-up timer marked by the factory.

The counting time of the dedicated power-down wake-up timer is calculated as follows: \( F_{wt} \) is the clock frequency of the internal wake-up timer we got from F8H and F9H of the RAM area.

\[
\text{counting time of the power-down wake-up timer} = \frac{10^6}{F_{wt}} \times 16 \times \text{counting timers} \text{ (us)}
\]

Assume \( F_{wt} = 32 \text{KHz} \), we have,

\[
\begin{array}{c|c}
{\text{WKTCH[6:0],WKTCL[7:0]}} & \text{counting time of the dedicated power-down wake-up timer} \\
0 & 10^6 \div 32 \times 16 \times (1+0) \approx 0.5 \text{ms} \\
9 & 10^6 \div 32 \times 16 \times (1+9) \approx 5 \text{ ms} \\
\end{array}
\]
13.6 Demo code

13.6.1 Timer 0 (Automatic reloading of mode 0 — 16 bit)

Assembly code

The operating frequency is 11.0592 MHz

```
ORG 0000H
LJMP MAIN

ORG 000BH
LJMP TM0ISR

ORG 0100H

TM0ISR:
CPL P1.0 ; port of the test
RETI

MAIN:
MOV SP,#3FH
MOV TMOD,#00H ; mode 0
MOV TL0,#66H ; 65536-11.0592M/12/1000
MOV TH0,#0FCH
SETB TR0 ; start the timer
SETB ET0 ; enable the interrupt of timer
SETB EA
JMP $ ;
END
```

C code

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

The operating frequency is 11.0592 MHz

sbit P10 = P1^0;

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

void main()
{
    TMOD = 0x00; // mode 0
    TL0 = 0x66; // 65536-11.0592M/12/1000
    TH0 = 0xfc;
```
TR0 = 1;  //start the timer
ET0 = 1;  //enable the interrupt of timer
EA = 1;

while (1);
}

13.6.2 Timer 0(Non automatic reloading of mode 1 — 16 bit)

Assembly code

The operating frequency is 11.0592 MHz

```
ORG 0000H
LJMP MAIN
ORG 000BH
LJMP TM0ISR

ORG 0100H

TM0ISR:
    MOV TL0,#66H ;reset the parameters of timer
    MOV TH0,#0FCH
    CPL P1.0 ;port of the test
    RETI

MAIN:
    MOV SP,#3FH
    MOV TMOD,#01H ;mode1
    MOV TL0,#66H ;65536-11.0592M/12/1000
    MOV TH0,#0FCH
    SETB TR0 ;start the timer
    SETB ET0 ;enable the interrupt of timer
    SETB EA
    JMP $    END
```

C code

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

//The operating frequency is 11.0592 MHz

sbit P10 = P1^0;

void TM0_Isr() interrupt 1
{
    TL0 = 0x66;  //reset the parameters of timer
    TH0 = 0xfc;
    P10 = !P10;  //port of the test
}

void main()
{
    TMOD = 0x01;  //mode 1
```
```
TL0 = 0x66;  //65536-11.0592M/12/1000
TH0 = 0xfc;
TR0 = 1;     //start the timer
ET0 = 1;     //enable the interrupt of timer
EA = 1;
while (1);
```

### 13.6.3 Timer 0 (Automatic reloading of mode 2 — 8 bit)

#### Assembly code

*The operating frequency is 11.0592 MHz*

```
ORG 0000H
LJMP MAIN

ORG 000BH
LJMP TM0ISR

TM0ISR:
CPL P1.0 ;port of the test
RETI

MAIN:
MOV SP,#3FH
MOV TMOD,#02H ;mode2
MOV TL0,#0F4H ;256-11.0592M/12/76K
MOV TH0,#0F4H
SETB TR0 ;start the timer
SETB ET0 ;enable the interrupt of timer
SETB EA
JMP $ 
```

#### C code

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

//The operating frequency is 11.0592 MHz

sbit P10 = P1^0;

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

void main()
{
    TMOD = 0x02; //mode2
    TL0 = 0xf4;  //256-11.0592M/12/76K
    TH0 = 0xf4;
```
13.6.4 Timer 0 (Automatic reloading of non-maskable interrupt for mode 2—8 bit)

Assembly code

`TR0 = 1; //start the timer
ET0 = 1; //enable the interrupt of timer
EA = 1;
while (1);`

```
13.6.4 Timer 0 (Automatic reloading of non-maskable interrupt for mode 2—8 bit)

Assembly code

The operating frequency is 11.0592 MHz
ORG 0000H
LJMP MAIN
ORG 000BH
LJMP TM0ISR

TM0ISR:
CPL P1.0 ;port of the test
RETI

MAIN:
MOV SP,#3FH
MOV TMOD,#03H ;mode3
MOV TL0,#66H ;65536-11.0592M/12/1000
MOV TH0,#0FCH
SETB TR0 ;start the timer
SETB ET0 ;enable the interrupt of timer
;
JMP $
END
```

C code

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

//The operating frequency is 11.0592 MHz

sbit P10 = P1^0;

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

void main()
{
    TMOD = 0x03; //mode3
    TL0 = 0x66; //65536-11.0592M/12/1000
```

Nantong guoxin Microelectronics Co., Ltd. Tel: 0513-5501 2928/2929/2966 Fax: 0513-5501 2926/2956/2947 - 223 -
13.6.5 Timer 0 (External counting — set T0 as external interrupt of falling edge)

Assembly code

The operating frequency is 11.0592 MHz

ORG 0000H
LJMP MAIN
ORG 000BH
LJMP TM0ISR

ORG 0100H
TM0ISR:
CPL P1.0 ; port of the test
RETI

MAIN:
MOV SP,#3FH
MOV TMOD,#04H ; External counting mode
MOV TL0,#0FFH
MOV TH0,#0FFH
SETB TR0 ; start the timer
SETB ET0 ; enable the interrupt of timer
SETB EA
JMP $
END

C code

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

// The operating frequency is 11.0592 MHz

sbit P10 = P1^0;

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

void main()
{
    TMOD = 0x04; // External counting mode
}
TL0 = 0xff;
TH0 = 0xff;
TR0 = 1;       //start the timer
ET0 = 1;       //enable the interrupt of timer
EA = 1;
while (1);

13.6.6 Timer 0(Test pulse width—the width of high level for INT0)

Assembly code

The operating frequency is 11.0592 MHz

<table>
<thead>
<tr>
<th>AUXR</th>
<th>DATA</th>
<th>8EH</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORG</td>
<td>0000H</td>
<td></td>
</tr>
<tr>
<td>LJMP</td>
<td>MAIN</td>
<td></td>
</tr>
<tr>
<td>ORG</td>
<td>0003H</td>
<td></td>
</tr>
<tr>
<td>LJMP</td>
<td>INT0ISR</td>
<td></td>
</tr>
<tr>
<td>ORG</td>
<td>0100H</td>
<td></td>
</tr>
</tbody>
</table>

INT0ISR:
MOV P0,TL0
MOV P1,TH0
RETI

MAIN:
MOV SP,#3FH
MOV AUXR,#80H ;1T mode
MOV TMOD,#08H
MOV TL0,#00H
MOV TH0,#00H
JB INT0,$ ;start the timer
SETB TR0
SETB IT0
SETB EX0
SETB EA
JMP $
END

C code

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

//The operating frequency is 11.0592 MHz

sfr AUXR = 0x8e;

void INT0_Isr() interrupt 0
void main()
{
    AUXR = 0x80; //IT mode
    TMOD = 0x08;
    TL0 = 0x00;
    TH0 = 0x00;
    while (INT0);
    TR0 = 1; //start the timer
    IT0 = 1;
    EX0 = 1;
    EA = 1;

    while (1);
}

13.6.7 Timer 0(Clock divider output)

Assembly code

The operating frequency is 11.0592 MHz

<table>
<thead>
<tr>
<th>INTCLKO</th>
<th>DATA</th>
<th>8FH</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORG</td>
<td>0000H</td>
<td></td>
</tr>
<tr>
<td>LJMP</td>
<td>MAIN</td>
<td></td>
</tr>
<tr>
<td>ORG</td>
<td>0100H</td>
<td></td>
</tr>
<tr>
<td>MAIN:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOV</td>
<td>SP,#3FH</td>
<td></td>
</tr>
<tr>
<td>MOV</td>
<td>TMOD,#00H; mode 0</td>
<td></td>
</tr>
<tr>
<td>MOV</td>
<td>TL0,#66H; 65536-11.0592MHz/12/1000</td>
<td></td>
</tr>
<tr>
<td>MOV</td>
<td>TH0,#0FCH</td>
<td></td>
</tr>
<tr>
<td>SETB</td>
<td>TR0</td>
<td>;start the timer</td>
</tr>
<tr>
<td>MOV</td>
<td>INTCLKO,#01H; enable the output of timer</td>
<td></td>
</tr>
<tr>
<td>JMP</td>
<td>$</td>
<td></td>
</tr>
<tr>
<td>END</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

C code

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

//The operating frequency is 11.0592 MHz

sfr INTCLKO = 0x8f;

void main()
{
    TMOD = 0x00; //mode 0
    TL0 = 0x66; //65536-11.0592MHz/12/1000
TH0 = 0xfc;
TR0 = 1;     //start the timer
INTCLKO = 0x01;    //enable the output of timer

while (1);

13.6.8  Timer 1(Automatic reloading of mode 0 — 16 bit)

Assembly code

The operating frequency is 11.0592 MHz:

```
ORG 0000H
LJMP MAIN
ORG 001BH
LJMP TM1ISR

ORG 0100H
TM1ISR:
CPL P1.0 ;port of the test
RETI

MAIN:
MOV SP,#3FH
MOV TMOD,#00H ;mode 0
MOV TL1,#66H ;65536-11.0592M/12/1000
MOV TH1,#0FCH
SETB TR1 ;start the timer
SETB ET1 ;enable the interrupt of timer
SETB EA
JMP $
END
```

C code

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

//The operating frequency is 11.0592 MHz

sbit P10 = P1^0;

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

void main()
{
    TMOD = 0x00;              //mode 0
    TL1 = 0x66;               //65536-11.0592M/12/1000
    TH1 = 0xfc;
    TR1 = 1;                 //start the timer
    ET1 = 1;                 //enable the interrupt of timer
```
EA = 1;

while (1);
}

13.6.9 Timer 1 (Non automatic reloading of mode 1—16 bit)

Assembly code

The operating frequency is 11.0592 MHz:

```
ORG 0000H
LJMP MAIN
ORG 001BH
LJMP TM1ISR
ORG 0100H

TM1ISR:
MOV TL1,#66H ;reset the parameters of timer
MOV TH1,#0FCH
CPL P1.0 ;port of the test
RETI

MAIN:
MOV SP,#3FH
MOV TMOD,#10H ;mode1
MOV TL1,#66H ;65536-11.0592MHz/12/1000
MOV TH1,#0FCH
SETB TR1 ;start the timer
SETB ET1 ;enable the interrupt of timer
SETB EA
JMP $  
END
```

C code

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

//The operating frequency is 11.0592 MHz

sbit P10 = P1^0;

void TM1_Isr() interrupt 3
{
    TL1 = 0x66; //reset the parameters of timer
    TH1 = 0xfc;
    P10 = !P10; //port of the test
}

void main()
{
```
TMOD = 0x10; //mode1
TL1 = 0x66; //65536-11.0592M/12/1000
TH1 = 0xfC;
TR1 = 1; //start the timer
ET1 = 1; //enable the interrupt of timer
EA = 1;
while (1);
}

13.6.10  Timer 1(Automatic reloading of mode 2—8 bit)

Assembly code

The operating frequency is 11.0592 MHz:

```
ORG 0000H
LJMP MAIN
ORG 001BH
LJMP TM1ISR

ORG 0100H
TM1ISR:
  CPL P1.0 ;port of the test
RETI

MAIN:
  MOV SP,#3FH
  MOV TMOD,#20H ;mode2
  MOV TL1,#0F4H ;256-11.0592M/12/76K
  MOV TH1,#0F4H
  SETB TR1 ;start the timer
  SETB ET1 ;enable the interrupt of timer
  SETB EA
  JMP $
END
```

C code

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

//The operating frequency is 11.0592 MHz

sbit P10 = P1^0;

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

void main()
{
  TMOD = 0x20; //mode2
  TL1 = 0xf4; //256-11.0592M/12/76K
```
TH1 = 0x4f;
TR1 = 1; //start the timer
ET1 = 1; //enable the interrupt of timer
EA = 1;

while (1);
}

13.6.11 Timer 1(External counting—set T1 as external interrupt of falling edge)

Assembly code

The operating frequency is 11.0592 MHz

ORG 0000H
LJMP MAIN
ORG 001BH
LJMP TM1ISR

ORG 0100H

TM1ISR:
CPL P1.0 ;port of the test
RETI

MAIN:
MOV SP,#3FH
MOV TMOD,#40H ;External counting mode
MOV TL1,#0FFH
MOV TH1,#0FFH
SETB TR1 ;start the timer
SETB ET1 ;enable the interrupt of timer
SETB EA

JMP $

END

C code

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

//The operating frequency is 11.0592 MHz

sbit P10 = P1^0;

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

void main()
{
    TMOD = 0x40; //External counting mode
TL1 = 0xff;
TH1 = 0xff;
TR1 = 1; //start the timer
ET1 = 1; //enable the interrupt of timer
EA = 1;
while (1);
}

13.6.12 Timer 1(Test pulse width—the width of high level for INT1)

Assembly code

The operating frequency is 11.0592 MHz

<table>
<thead>
<tr>
<th>AUXR</th>
<th>DATA</th>
<th>8EH</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORG</td>
<td>0000H</td>
<td></td>
</tr>
<tr>
<td>LJMP</td>
<td>MAIN</td>
<td></td>
</tr>
<tr>
<td>ORG</td>
<td>0013H</td>
<td></td>
</tr>
<tr>
<td>LJMP</td>
<td>INT1ISR</td>
<td></td>
</tr>
<tr>
<td>ORG</td>
<td>0100H</td>
<td></td>
</tr>
</tbody>
</table>

INT1ISR:
MOV P0,TL1
MOV P1,TH1
RETI

MAIN:
MOV SP,#3FH
MOV AUXR,#40H ;1Tmode
MOV TMOD,#80H
MOV TL1,#00H
MOV TH1,#00H
JB INT1,$
SETB TR1 ;start the timer
SETB IT1
SETB EX1
SETB EA

JMP $

END

C code

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

//The operating frequency is 11.0592 MHz

sfr AUXR = 0x8e;

void INT1_Isr() interrupt 2
```c
#include "reg51.h"
#include "intrins.h"

//The operating frequency is 11.0592 MHz

sfr INTCLKO = 0x8f;

void main()
{
    TMOD = 0x00; //mode 0
    TL1 = 0x66;  //65536-11.0592M/12/1000
    TH1 = 0x00;
    TR1 = 1;     //start the timer
    IT1 = 1;
    EX1 = 1;
    EA = 1;

    while (1);
}
```

**13.6.13  Timer 1(clock divider output)**

Assembly code

The operating frequency is 11.0592 MHz

<table>
<thead>
<tr>
<th>INTCLKO</th>
<th>DATA</th>
<th>8FH</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORG</td>
<td>0000H</td>
<td></td>
</tr>
<tr>
<td>LJMP</td>
<td>MAIN</td>
<td></td>
</tr>
<tr>
<td>ORG</td>
<td>0100H</td>
<td></td>
</tr>
</tbody>
</table>

MAIN:

MOV SP,#3FH
MOV TMOD,#00H ;mode 0
MOV TL1,#66H ;65536-11.0592M/12/1000
MOV TH1,#0FCH
SETB TR1 ;start the timer
MOV INTCLKO,#02H ;enable the output of timer
JMP $ ;

END

C code

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

//The operating frequency is 11.0592 MHz

sfr INTCLKO = 0x8f;

void main()
{
    TMOD = 0x00; //mode 0
    TL1 = 0x66;  //65536-11.0592M/12/1000
```
TH1 = 0xfc;
TR1 = 1;  //start the timer
INTCLKO = 0x02;  //enable the output of timer
while (1);
}

13.6.14  Configure Timer 1(mode 0)as Baud Rate
Generate of serial port 1

Assembly code

<table>
<thead>
<tr>
<th>AUXR</th>
<th>DATA</th>
<th>8EH</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUSY</td>
<td>BIT</td>
<td>20H.0</td>
</tr>
<tr>
<td>WPTR</td>
<td>DATA</td>
<td>21H</td>
</tr>
<tr>
<td>RPTR</td>
<td>DATA</td>
<td>22H</td>
</tr>
<tr>
<td>BUFFER</td>
<td>DATA</td>
<td>23H</td>
</tr>
</tbody>
</table>

;16 bytes

ORG 0000H
LJMP MAIN
ORG 0023H
LJMP UART_ISR

ORG 0100H

UART_ISR:
PUSH ACC
PUSH PSW
MOV PSW,#08H
JNB TI,CHKRI
CLR TI
CLR BUSY
CHKRI:
JNB TI,UARTISR_EXIT
CLR RI
MOV A,WPTR
ANL A,#0FH
ADD A,#BUFFER
MOV R0,A
MOV @R0,SBUF
INC WPTR
UARTISR_EXIT:
PUSH ACC
POP PSW
UART_INIT:
MOV SCON,#50H
MOV TMOD,#00H
MOV TL1,#0E8H ;65536-11059200/115200/4=0FFE8H
MOV TH1,#0FFH
SETB TR1
MOV AUXR,#40H

Nantong guoxin Microelectronics Co., Ltd.  Tel: 0513-5501 2928/2929/2966  Fax: 0513-5501 2926/2956/2947 - 233 -
CLR       BUSY
MOV       WPTR,#00H
MOV       RPTR,#00H
RET

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

UART_SENDSTR:
CLR       A
MOVC      A,@A+DPTR
JZ        SENDEND
LCALL     UART_SEND
INC       DPTR
JMP       UART_SENDSTR
SENDEND:
RET

MAIN:
MOV       SP,#3FH
LCALL     UART_INIT
SETB      ES
SETB      EA
MOV       DPTR,#STRING
LCALL     UART_SENDSTR
LOOP:
MOV       A,RPTR
XRL       A,WPTR
ANL       A,#0FH
JZ        LOOP
MOV       A,RPTR
ANL       A,#0FH
ADD       A,#BUFFER
MOV       R0,A
MOV       A,@R0
LCALL     UART_SEND
INC       RPTR
JMP       LOOP

STRING:   DB 'Uart Test !',0DH,0AH,00H
END

C code
#include "reg51.h"
#include "intrins.h"

#define FOSC 11059200UL
#define BRT (65536 - FOSC / 115200 / 4)
sfr AUXR = 0x8e;
bit busy;
char wptr;
char rptr;
char buffer[16];

void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
        buffer[wptr++] = SBUF;
        wptr &= 0x0f;
    }
}

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

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

void UARTsendStr(char *p)
{
    while (*p)
    {
        UARTsend(*p++);
    }
}

void main()
{
    UartInit();
    ES = 1;
    EA = 1;
    UARTsendStr("Uart Test \r\n");

    while (1)
    {  

```
if (rptr != wptr)
{
    UARTsend(buffer[rptr++]);
    rptr &= 0x0f;
}
}

13.6.15 Configure Timer 1(mode 2) as Baud Rate
Generate of serial port 1

Assembly code

<table>
<thead>
<tr>
<th>AUXR</th>
<th>DATA</th>
<th>8EH</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUSY</td>
<td>BIT</td>
<td>20H,0</td>
</tr>
<tr>
<td>WPTR</td>
<td>DATA</td>
<td>21H</td>
</tr>
<tr>
<td>RPTR</td>
<td>DATA</td>
<td>22H</td>
</tr>
<tr>
<td>BUFFER</td>
<td>DATA</td>
<td>23H</td>
</tr>
</tbody>
</table>

ORG 0000H
LJMP MAIN
ORG 0023H
LJMP UART_ISR

ORG 0100H

UART_ISR:
PUSH ACC
PUSH PSW
MOV PSW,#08H

JNB TI,CHKRI
CLR TI
CLR BUSY

CHKRI:
JNB RI,UARTISR_EXIT
CLR RI
MOV A,WPTR
ANL A,#0FH
ADD A,#BUFFER
MOV R0,A
MOV @R0,SBUF
INC WPTR

UARTISR_EXIT:
POP PSW
POP ACC
RETI

UART_INIT:
MOV SCON,#50H
MOV TMOD,#20H
MOV TL1,#0FDH ;256-11059200/115200/32=0FDH
MOV TH1,#0FDH  
SETB TR1  
MOV AUXR,#40H  
CLR BUSY  
MOV WPTR,#00H  
MOV RPTR,#00H  
RET

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

UART_SENDSTR:  
    CLR A  
    MOVC A,@A+DPTR  
    JZ SENDEND  
    LCALL UART_SEND  
    INC DPTR  
    JMP UART_SENDSTR  
SENDEND:  
    RET

MAIN:  
    MOV SP,#3FH  
    LCALL UART_INIT  
    SETB ES  
    SETB EA  
    MOV DPTR,#STRING  
    LCALL UART_SENDSTR  
LOOP:  
    MOV A,RPTR  
    XRL A,WPTR  
    ANL A,#0FH  
    JZ LOOP  
    MOV A,RPTR  
    ANL A,#0FH  
    ADD A,#BUFFER  
    MOV R0,A  
    MOV A,@R0  
    LCALL UART_SEND  
    INC RPTR  
    JMP LOOP

STRING: DB 'Uart Test !',0DH,0AH,00H  
END

C code
#include "reg51.h"  
#include "intrins.h"  
#define FOSC 11059200UL
#define BRT (256 - FOSC / 115200 / 32)

sfr AUXR = 0x8e;

bit busy;
char wpotr;
char rptr;
char buffer[16];

void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
        buffer[wpotr++] = SBUF;
        wpotr &= 0x0f;
    }
}

void UartInit()
{
    SCON = 0x50;
    TMOD = 0x20;
    TL1 = BRT;
    TH1 = BRT;
    TR1 = 1;
    AUXR = 0x40;
    wpotr = 0x00;
    rptr = 0x00;
    busy = 0;
}

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

void UARTsendStr(char *p)
{
    while (*p)
    {
        UARTsend(*p);
    }

void main()
{
    UartInit();
    ES = 1;
    EA = 1;
    UARTsendStr("Uart Test \r\n");
}
while (1) 
{
    if (rptr != wptr) 
    {
        UARTsend(buffer[rptr++]);
        rptr &= 0x0f;
    }
}

13.6.16 Timer 2 (Automatic reloading for 16 bits)

Assembly code

The operating frequency is 11.0592 MHz

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2L</td>
<td>DATA</td>
<td>0D7H</td>
</tr>
<tr>
<td>T2H</td>
<td>DATA</td>
<td>0D6H</td>
</tr>
<tr>
<td>AUXR</td>
<td>DATA</td>
<td>8EH</td>
</tr>
<tr>
<td>IE2</td>
<td>DATA</td>
<td>0AFH</td>
</tr>
<tr>
<td>ET2</td>
<td>EQU</td>
<td>04H</td>
</tr>
<tr>
<td>AUXINTIF</td>
<td>DATA</td>
<td>0EFH</td>
</tr>
<tr>
<td>T2IF</td>
<td>EQU</td>
<td>01H</td>
</tr>
</tbody>
</table>

ORG 0000H
LJMP MAIN
ORG 0063H
LJMP TM2ISR

ORG 0100H

TM2ISR:
CPL P1.0 ;port of the test
ANL AUXINTIF,#NOT T2IF ;clear the symbol of interrupt
RETI

MAIN:
MOV SP,#3FH
MOV T2L,#66H ;65536-11.0592M/12/1000
MOV T2H,#0FCH
MOV AUXR,#10H ;start the timer
MOV IE2,#ET2 ;enable the interrupt of timer
SETB EA
JMP $

END

C code

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

//The operating frequency is 11.0592 MHz

sfr T2L = 0xd7;
sfr T2H = 0xd6;
void TM2_Isr() interrupt 12
{
    P10 = !P10; //port of the test
    AUXINTIF &= ~T2IF; //clear the symbol of interrupt
}

void main()
{
    T2L = 0x66; //65536-11.0592M/12/1000
    T2H = 0xfc;
    AUXR = 0x10; //start the timer
    IE2 = ET2; //enable the interrupt of timer
    EA = 1;
    while (1);
}

13.6.17 Timer 2 (External counting—set T2 as external interrupt of falling edge)

Assembly code

The operating frequency is 11.0592 MHz.
T2L DATA 0D7H
T2H DATA 0D6H
AUXR DATA 8EH
IE2 DATA 0AFH
ET2 EQU 04H
AUXINTIF DATA 0EFH
T2IF EQU 01H

ORG 0000H
LJMP MAIN
ORG 0063H
LJMP TM2ISR

ORG 0100H
TM2ISR:
    CPL P1.0 ;port of the test
    ANL AUXINTIF,#NOT T2IF ;clear the symbol of interrupt
    RETI

MAIN:
    MOV SP,#3FH
    MOV T2L,#0FFH
    MOV T2H,#0FFH

Nantong guoxin Microelectronics Co., Ltd. Tel: 0513-5501 2928/2929/2966 Fax: 0513-5501 2926/2956/2947 - 240 -
MOV AUXR,#18H ;set External countingmode and start the timer
MOV IE2,#ET2 ;enable the interrupt of timer
SETB EA
JMP $  
END

C code

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

//The operating frequency is 11.0592 MHz
sfr T2L = 0xd7;
sfr T2H = 0xd6;
sfr AUXR = 0x8e;
sfr IE2 = 0xaf;
#define ET2  0x04
sfr AUXINTIF = 0xef;
#define T2IF  0x01
sbit P10 = P1^0;

void TM2_Isr() interrupt 12
{
P10 = !P10;          //port of the test
AUXINTIF &= ~T2IF;  //clear the symbol of interrupt
}

void main()
{
    T2L = 0xff;
    T2H = 0xff;
    AUXR = 0x18;        //set External countingmode and start the timer
    IE2 = ET2;          //enable the interrupt of timer
    EA = 1;

    while (1);
}
```

13.6.18  Timer 2(clock divider output)

Assembly code

```asm
ORG 0000H
LJMP MAIN

ORG 0100H
MAIN:
```

The operating frequency is 11.0592 MHz

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2L</td>
<td>0D7H</td>
</tr>
<tr>
<td>T2H</td>
<td>0D6H</td>
</tr>
<tr>
<td>AUXR</td>
<td>8EH</td>
</tr>
<tr>
<td>INTCLKO</td>
<td>8FH</td>
</tr>
</tbody>
</table>

ORG 0000H
LJMP MAIN

ORG 0100H
MAIN:
MOV SP, #3FH
MOV T2L, #66H ; 65536 - 11.0592M / 12 / 1000
MOV T2H, #0FCH
MOV AUXR, #10H ; start the timer
MOV INTCLKO, #04H ; enable the output of timer
JMP $
END

C code
#include "reg51.h"
#include "intrins.h"

// The operating frequency is 11.0592 MHz
sfr T2L = 0xd7;
sfr T2H = 0xd6;
sfr AUXR = 0x8e;
sfr INTCLKO = 0x8f;

void main()
{
T2L = 0x66; // 65536 - 11.0592M / 12 / 1000
T2H = 0xfc;
AUXR = 0x10; // start the timer
INTCLKO = 0x04; // enable the output of timer

while (1);
}

13.6.19 Configure Timer 2 as Baud Rate Generate of serial port 1

Assembly code
AUXR DATA 8EH
T2H DATA 0D6H
T2L DATA 0D7H
BUSY BIT 20H.0
WPTR DATA 21H
RPTR DATA 22H
BUFFER DATA 23H ; 16 bytes

ORG 0000H
LJMP MAIN
ORG 0023H
LJMP UART_ISR

ORG 0100H

UART_ISR:
PUSH ACC
PUSH PSW
MOV PSW,#08H
JNB TI,CHKRI
CLR TI
CLR BUSY

CHKRI:
JNB RI,UARTISR_EXIT
CLR RI
MOV A,WPTR
ANL A,#0FH
ADD A,#BUFFER
MOV R0,A
MOV @R0,SBUF
INC WPTR

UARTISR_EXIT:
POP PSW
POP ACC
RETI

UART_INIT:
MOV SCON,#50H
MOV T2L,#0E8H ;65536-11059200/115200/4=0FFE8H
MOV T2H,#0FFH
MOV AUXR,#15H
CLR BUSY
MOV WPTR,#00H
MOV RPTR,#00H
RET

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

UART_SENDSTR:
CLR A
MOVC A,@A+DPTR
JZ SENDEND
LCALL UART_SEND
INC DPTR
JMP UART_SENDSTR

SENDEND:
RET

MAIN:
MOV SP,#3FH
LCALL UART_INIT
SETB ES
SETB EA
MOV DPTR,#STRING
LCALL UART_SENDSTR

LOOP:
MOV  A,RPTR
XRL  A,WPTR
ANL  A,#0FH
JZ   LOOP
MOV  A,RPTR
ANL  A,#0FH
ADD  A,#BUFFER
MOV  R0,A
MOVL  A,@R0
LCALL UART_SEND
INC  RPTTR
JMP  LOOP

STRING: DB 'Uart Test !',0DH,0AH,00H

END

C code

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

#define FOSC 11059200UL
#define BRT (65536 - FOSC / 115200 / 4)
sfr AUXR = 0x8e;
sfr T2H = 0xd6;
sfr T2L = 0xd7;
bit busy;
char wptr;
char rptr;
char buffer[16];

void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
        buffer[wptr++] = SBUF;
        wptr &= 0x0f;
    }
}

void UartInit()
{
    SCON = 0x50;
    T2L = BRT;
    T2H = BRT >> 8;
    AUXR = 0x15;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}
void UARTsend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}

void UARTsendStr(char *p)
{
    while (*p)
    {
        UARTsend(*p++);
    }
}

void main()
{
    UartInit();
    ES = 1;
    EA = 1;
    UARTsendStr("Uart Test \r\n");

    while (1)
    {
        if (rptr != wptr)
        {
            UARTsend(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}

13.6.20 Configure Timer 2 as Baud Rate Generate of serial port 2

Assembly code

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUXR</td>
<td>DATA</td>
<td>8EH</td>
</tr>
<tr>
<td>T2H</td>
<td>DATA</td>
<td>0D6H</td>
</tr>
<tr>
<td>T2L</td>
<td>DATA</td>
<td>0D7H</td>
</tr>
<tr>
<td>S2CON</td>
<td>DATA</td>
<td>9AH</td>
</tr>
<tr>
<td>S2BUF</td>
<td>DATA</td>
<td>9BH</td>
</tr>
<tr>
<td>IE2</td>
<td>DATA</td>
<td>0AFH</td>
</tr>
<tr>
<td>BUSY</td>
<td>BIT</td>
<td>20H.0</td>
</tr>
<tr>
<td>WPTR</td>
<td>DATA</td>
<td>21H</td>
</tr>
<tr>
<td>RPTR</td>
<td>DATA</td>
<td>22H</td>
</tr>
<tr>
<td>BUFFER</td>
<td>DATA</td>
<td>23H</td>
</tr>
</tbody>
</table>

;16 bytes

ORG    0000H
LJMP   MAIN
ORG    0043H
LJMP   UART2_ISR
ORG 0100H

UART2_ISR:
PUSH ACC
PUSH PSW
MOV PSW,#08H

MOV A,S2CON
JNB ACC.1,CHKRI
ANL S2CON,#NOT 02H
CLR BUSY

CHKRI:
JNB ACC.0,UART2ISR_EXIT
ANL S2CON,#NOT 01H
MOV A,WPTR
ANL A,#0FH
ADD A,#BUFFER
MOV R0,A
MOV @R0,S2BUF
INC WPTR

UART2ISR_EXIT:
POP PSW
POP ACC
RETI

UART2_INIT:
MOV S2CON,#50H
MOV T2L,#0E8H ;65536-11059200/115200/4=0FFE8H
MOV T2H,#0FFH
MOV AUXR,#14H
CLR BUSY
MOV WPTR,#00H
MOV RPTR,#00H
RET

UART2_SEND:
JB BUSY,S
SETB BUSY
MOV S2BUF,A
RET

UART2_SENDSTR:
CLR A
MOVC A,@A+DPTR
JZ SEND2END
LCALL UART2_SEND
INC DPTR
JMP UART2_SENDSTR

SEND2END:
RET

MAIN:
MOV SP,#3FH
LCALL UART2_INIT
MOV IE2,#01H
SETB EA
MOV DPTR,#STRING
LCALL UART2_SENDSTR

LOOP:
MOV A,RPTR
XRL A,WPTR
ANL A,#0FH
JZ LOOP
MOV A,RPTR
ANL A,#0FH
ADD A,#BUFFER
MOV R0,A
MOV A,@R0
LCALL UART2_SEND
INC RPTR
JMP LOOP

STRING: DB 'Uart Test !',0DH,0AH,00H
END

C code

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

#define FOSC 11059200UL
#define BRT (65536 - FOSC / 115200 / 4)
sfr AUXR = 0x8e;
sfr T2H = 0xd6;
sfr T2L = 0xd7;
sfr S2CON = 0x9a;
sfr S2BUF = 0x9b;
sfr IE2 = 0xaf;
bit busy;
char wptr;
char rptr;
char buffer[16];

void Uart2Isr() interrupt 8
{
    if (S2CON & 0x02)
    {
        S2CON &= ~0x02;
        busy = 0;
    }
    if (S2CON & 0x01)
    {
        S2CON &= ~0x01;
        buffer[wptr++] = S2BUF;
        wptr &= 0x0f;
    }
}

void Uart2Init()
void Uart2Send(char dat)
{
    while (busy);
    busy = 1;
    S2BUF = dat;
}

void Uart2SendStr(char *p)
{
    while (*p)
    {
        Uart2Send(*p++);
    }
}

void main()
{
    Uart2Init();
    IE2 = 0x01;
    EA = 1;
    Uart2SendStr(“Uart Test !\r\n”);

    while (1)
    {
        if (rptr != wptr)
        {
            Uart2Send(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}

13.6.21 Configure Timer 2 as Baud Rate Generate of serial port 3

Assembly code

<table>
<thead>
<tr>
<th></th>
<th>DATA</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AUXR</td>
<td>DATA</td>
<td>8EH</td>
</tr>
<tr>
<td>T2H</td>
<td>DATA</td>
<td>0D6H</td>
</tr>
<tr>
<td>T2L</td>
<td>DATA</td>
<td>0D7H</td>
</tr>
<tr>
<td>S3CON</td>
<td>DATA</td>
<td>04CH</td>
</tr>
<tr>
<td>S3BUF</td>
<td>DATA</td>
<td>0ADH</td>
</tr>
<tr>
<td>IE2</td>
<td>DATA</td>
<td>0AFH</td>
</tr>
<tr>
<td>BUSY</td>
<td>BIT</td>
<td>20H.0</td>
</tr>
</tbody>
</table>
WPTR DATA 21H
RPTR DATA 22H
BUFFER DATA 23H ;16 bytes

ORG 0000H
LJMP MAIN
ORG 008BH
LJMP UART3_ISR
ORG 0100H

UART3_ISR:
PUSH ACC
PUSH PSW
MOV PSW,#08H

MOV A,S3CON
JNB ACC.1,CHKRI
ANL S3CON,#NOT 02H
CLR BUSY

CHKRI:
JNB ACC.0,UART3ISR_EXIT
ANL S3CON,#NOT 01H
CLR BUSY

UART3_ISR_EXIT:
PUSH ACC
POP PSW
POP ACC
RETI

UART3_INIT:
MOV S3CON,#10H
MOV T2L,#0E8H ;65536-11059200/115200/4=0FFE8H
MOV T2H,#0FFH
MOV AUXR,#14H
CLR BUSY
MOV WPTR,#00H
MOV RPTR,#00H
RET

UART3_SEND:
JB BUSY,$
SETB BUSY
MOV S3BUF,A
RET

UART3_SENDSTR:
CLR A
MOVC A,@A+DPTR
JZ SEND3END
LCALL UART3_SEND
INC DPTR
JMP UART3_SENDSTR

SEND3END:
RET

MAIN:
MOV SP,#3FH
LCALL UART3_INIT
MOV IE2,#08H
SETB EA
MOV DPTR,#STRING
LCALL UART3_SENDSTR

LOOP:
MOV A,RPTR
XRL A,WPTR
ANL A,#0FH
JZ LOOP
MOV A,RPTR
ANL A,#0FH
ADD A,#BUFFER
MOV R0,A
MOV A,@R0
LCALL UART3_SEND
INC RPTR
JMP LOOP

STRING:
DB 'Uart Test !',0DH,0AH,00H

END

C code
#include "reg51.h"
#include "intrins.h"
#define FOSC 11059200UL
#define BRT (65536 - FOSC / 115200 / 4)
sfr AUXR = 0x8e;
sfr T2H = 0xd6;
sfr T2L = 0xd7;
sfr S3CON = 0xac;
sfr S3BUF = 0xad;
sfr IE2 = 0xaf;

bit busy;
char wptr;
char rprr;
char buffer[16];

void Uart3Isr() interrupt 17
{
    if (S3CON & 0x02)
    {
        S3CON &= ~0x02;
        busy = 0;
    }
    if (S3CON & 0x01)
13.6.22 Configure Timer 2 as Baud Rate Generate of serial port 4

Assembly code
AUXR DATA 8EH
T2H DATA 0D6H
T2L DATA 0D7H
S4CON DATA 84H
S4BUF DATA 085H
IE2 DATA 0AFH

BUSY BIT 20H.0
WPTR DATA 21H
RPTR DATA 22H
BUFFER DATA 23H ;16 bytes

ORG 0000H
LJMP MAIN
ORG 0093H
LJMP UART4_ISR

ORG 0100H

UART4_ISR:
PUSH ACC
PUSH PSW
MOV PSW,#08H

MOV A,S4CON
JNB ACC.1,CHKRI
ANL S4CON,#NOT 02H
CLR BUSY

CHKRI:
JNB ACC.0,UART4_ISR_EXIT
ANL S4CON,#NOT 01H
MOV A,WPTR
ANL A,#0FH
ADD A,#BUFFER
MOV @R0,A
MOV @R0,S4BUF
INC WPTR

UART4_ISR_EXIT:
POP PSW
POP ACC
RETI

UART4_INIT:
MOV S4CON,#10H
MOV T2L,#0E8H ;65536-11059200/115200/4=0FFE8H
MOV T2H,#0FFH
MOV AUXR,#14H
CLR BUSY
MOV WPTR,#00H
MOV RPTR,#00H
RETI

UART4_SEND:
JB BUSYS
SETB BUSY
MOV S4BUF,A
UART4_SENDSTR:
    CLR     A
    MOVC   A,@A+DPTR
    JZ     SEND4END
    LCALL  UART4_SEND
    INC    DPTR
    JMP    UART4_SENDSTR
SEND4END:
    RET

MAIN:
    MOV     SP,#3FH
    LCALL  UART4_INIT
    MOV     IE2,#10H
    SETB   EA
    MOV     DPTR,#STRING
    LCALL  UART4_SENDSTR

LOOP:
    MOV     A,RPTR
    XRL    A,WPTR
    ANL    A,#0FH
    JZ     LOOP
    MOV     A,#0FH
    ADD    A,BUFFER
    MOV     R0,A
    MOV     A,R0
    LCALL  UART4_SEND
    INC    RPTR
    JMP    LOOP

STRING:    DB  'Uart Test !',0DH,0AH,00H

END

C code

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

#define FOSC 11059200UL
#define BRT (65536 - FOSC / 115200 / 4)
sfr AUXR = 0x8e;
sfr T2H = 0xd6;
sfr T2L = 0xd7;
sfr S4CON = 0x84;
sfr S4BUF = 0x85;
sfr IE2 = 0xaf;

bit     busy;
char    wptr;
char    rptr;
char    buffer[16];
void Uart4Isr() interrupt 18
{
    if (S4CON & 0x02)
    {
        S4CON &= ~0x02;
        busy = 0;
    }
    if (S4CON & 0x01)
    {
        S4CON &= ~0x01;
        buffer[wptr++] = S4BUF;
        wptr &= 0x0f;
    }
}

void Uart4Init()
{
    S4CON = 0x10;
    T2L = BRT;
    T2H = BRT >> 8;
    AUXR = 0x14;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}

void Uart4Send(char dat)
{
    while (busy);
    busy = 1;
    S4BUF = dat;
}

void Uart4SendStr(char *p)
{
    while (*p)
    {
        Uart4Send(*p+++);
    }
}

void main()
{
    Uart4Init();
    IE2 = 0x10;
    EA = 1;
    Uart4SendStr("Uart Test \r\n");

    while (1)
    {
        if (rptr != wptr)
        {
            Uart4Send(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}
13.6.23 Timer 3(Automatic reloading for 16 bits)

Assembly code

The operating frequency is 11.0592 MHz

<table>
<thead>
<tr>
<th>Register</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>T3L</td>
<td>0D5H</td>
</tr>
<tr>
<td>T3H</td>
<td>0D4H</td>
</tr>
<tr>
<td>T4T3M</td>
<td>0D1H</td>
</tr>
<tr>
<td>IE2</td>
<td>0AFH</td>
</tr>
<tr>
<td>ET3</td>
<td>20H</td>
</tr>
<tr>
<td>AUXINTIF</td>
<td>0EFH</td>
</tr>
<tr>
<td>T3IF</td>
<td>02H</td>
</tr>
</tbody>
</table>

ORG 0000H
LJMP MAIN
ORG 009BH
LJMP TM3ISR

TM3ISR:
CPL P1.0 ;port of the test
ANL AUXINTIF,#NOT T3IF ;clear the symbol of interrupt
RETI

MAIN:
MOV SP,#3FH
MOV T3L,#66H ;65536-11.0592M/12/1000
MOV T3H,#0FCH
MOV T4T3M,#08H ;start the timer
MOV IE2,#ET3 ;enable the interrupt of timer
SETB EA
JMP $
END

C code

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

//The operating frequency is 11.0592 MHz

sfr T3L = 0xd5;
sfr T3H = 0xd4;
sfr T4T3M = 0xd1;
sfr IE2 = 0xaf;
#define ET3 0x20
sfr AUXINTIF = 0xef;
#define T3IF 0x02
sbit P10 = P1^0;

void TM3_Isr() interrupt 19
{
}
P10 = !P10;  //port of the test
AUXINTIF &= ~T3IF;  //clear the symbol of interrupt

void main()
{
    T3L = 0x66;  //65536-11.0592M/12/1000
    T3H = 0xfc;
    T4T3M = 0x08;  //start the timer
    IE2 = ET3;  //enable the interrupt of timer
    EA = 1;

    while (1);
}

13.6.24 Timer 3(External counting — set T3 as the external interrupt for falling edge)

Assembly code

The operating frequency is 11.0592 MHz:

<table>
<thead>
<tr>
<th>Register</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>T3L</td>
<td>0D5H</td>
</tr>
<tr>
<td>T3H</td>
<td>0D4H</td>
</tr>
<tr>
<td>T4T3M</td>
<td>0D1H</td>
</tr>
<tr>
<td>IE2</td>
<td>0AFH</td>
</tr>
<tr>
<td>ET3</td>
<td>20H</td>
</tr>
<tr>
<td>AUXINTIF</td>
<td>0EFH</td>
</tr>
<tr>
<td>T3IF</td>
<td>02H</td>
</tr>
</tbody>
</table>

ORG 0000H
LJMP MAIN
ORG 009BH
LJMP TM3ISR

ORG 0100H

TM3ISR:
CPL P1.0 ;port of the test
ANL AUXINTIF,#NOT T3IF ;clear the symbol of interrupt
RETI

MAIN:
MOV SP,#3FH
MOV T3L,#0FFH
MOV T3H,#0FFH
MOV T4T3M,#0CH ;set External counting mode and start the timer
MOV IE2,#ET3 ;enable the interrupt of timer
SETB EA
JMP $ 

END

C code
#include "reg51.h"
#include "intrins.h"

//The operating frequency is 11.0592 MHz
sfr T3L = 0xd5;
sfr T3H = 0xd4;
sfr T4T3M = 0xd1;
sfr IE2 = 0xaf;
#define ET3 0x20
sfr AUXINTIF = 0xef;
#define T3IF 0x02
sb1t P10 = P1^0;

void TM3_Isr() interrupt 19
{
P10 = !P10; //port of the test
AUXINTIF &= ~T3IF; //clear the symbol of interrupt
}

void main()
{
T3L = 0xff;
T3H = 0xff;
T4T3M = 0x0c; //set External counting mode and start the timer
IE2 = ET3; //enable the interrupt of timer
EA = 1;

while (1);
}

13.6.25 Timer 3(clock divider output)

Assembly code

| The operating frequency is 11.0592 MHz |
| T3L DATA 0D5H |
| T3H DATA 0D4H |
| T4T3M DATA 0D1H |
| ORG 0000H |
| LJMP MAIN |
| ORG 0100H |
| MAIN: |
| MOV SP,#3FH |
| MOV T3L,#66H ;65536-11.0592M/12/1000 |
| MOV T3H,#0FCH |
| MOV T4T3M,#09H ;enable the output of timer #start the timer |
| JMP $ |

END
C code

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

//The operating frequency is 11.0592 MHz
sfr T3L = 0xd5;
sfr T3H = 0xd4;
sfr T4T3M = 0xd1;

void main()
{
    T3L = 0x66; //65536-11.0592M/12/1000
    T3H = 0xfc;
    T4T3M = 0x09; //enable the output of timer # start the timer

    while (1);
}
```

13.6.26 Configure Timer 3 as Baud Rate Generate of serial port 3

Assembly code

```assembly
T4T3M DATA 0D1H
T3H DATA 0D4H
T3L DATA 0D5H
S3CON DATA 0ACH
S3BUF DATA 0ADH
IE2 DATA 0AFH

BUSY BIT 20H.0
WPTR DATA 21H
RPTR DATA 22H
BUFFER DATA 23H ;16 bytes

ORG 0000H
LJMP MAIN
ORG 008BH
LJMP UART3_ISR

ORG 0100H

UART3_ISR:
PUSH ACC
PUSH PSW
MOV PSW,#08H

MOV A,S3CON
JNB ACC.1,CHKRI
ANL S3CON,#NOT 02H
CLR BUSY

CHKRI:
JNB ACC.0,UART3ISR_EXIT
ANL S3CON,#NOT 01H
```
MOV A,WPTR
ANL A,#0FH
ADD A,#BUFFER
MOV R0,A
MOV @R0,S3BUF
INC WPTR

UART3_ISR_EXIT:
POP PSW
POP ACC
RETI

UART3_INIT:
MOV S3CON,#50H
MOV T3L,#0E8H ;65536-11059200/115200/4=0FFE8H
MOV T3H,#0FFH
MOV T4T3M,#0AH
CLR BUSY
MOV WPTR,#00H
MOV RPTR,#00H
RET

UART3_SEND:
JB BUSY,$
SETB BUSY
MOV S3BUF,A
RET

UART3_SENDSTR:
CLR A
MOVC A,@A+DPTR
JZ SEND3END
LCALL UART3_SEND
INC DPTR
JMP UART3_SENDSTR
SEND3END:
RET

MAIN:
MOV SP,#3FH
LCALL UART3_INIT
MOV IE2,#08H
SETB EA
MOV DPTR,#STRING
LCALL UART3_SENDSTR

LOOP:
MOV A,RPTR
XRL A,WPTR
ANL A,#0FH
JZ LOOP
MOV A,RPTR
ANL A,#0FH
ADD A,#BUFFER
MOV R0,A
MOV A,R0
LCALL UART3_SEND
C code

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

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

sfr T4T3M = 0xd1;
sfr T3H = 0xd4;
sfr T3L = 0xd5;
sfr S3CON = 0xac;
sfr S3BUF = 0xad;
sfr IE2 = 0xaf;

bit busy;
char wptr;
char rptr;
char buffer[16];

void Uart3Isr() interrupt 17
{
    if (S3CON & 0x02)
        S3CON &= ~0x02;
    busy = 0;
}

void Uart3Init()
{
    S3CON = 0x50;
    T3L = BRT;
    T3H = BRT >> 8;
    T4T3M = 0x0a;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}

void Uart3Send(char dat)
{
    while (busy);
    busy = 1;
    S3BUF = dat;
```
void Uart3SendStr(char *p)
{
    while (*p)
    {
        Uart3Send(*p++);
    }
}

void main()
{
    Uart3Init();
    IE2 = 0x08;
    EA = 1;
    Uart3SendStr("Uart Test !\n");

    while (1)
    {
        if (rptr != wptr)
        {
            Uart3Send(buf[ｒptr++]);
            rptr &= 0x0f;
        }
    }
}

13.6.27 Timer4 (Automatic reloading for 16 bits)

Assembly code

The operating frequency is 11.0592 MHz

<table>
<thead>
<tr>
<th></th>
<th>DATA</th>
<th>0D3H</th>
</tr>
</thead>
<tbody>
<tr>
<td>T4L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T4H</td>
<td></td>
<td>0D2H</td>
</tr>
<tr>
<td>T4T3M</td>
<td></td>
<td>0D1H</td>
</tr>
<tr>
<td>IE2</td>
<td></td>
<td>0AFH</td>
</tr>
<tr>
<td>ET4</td>
<td>EQU</td>
<td>40H</td>
</tr>
<tr>
<td>AUXINTIF</td>
<td>DATA</td>
<td>0EFH</td>
</tr>
<tr>
<td>T4IF</td>
<td>EQU</td>
<td>04H</td>
</tr>
</tbody>
</table>

ORG 0000H
LJMP MAIN
ORG 00A3H
LJMP TM4ISR

ORG 0100H

TM4ISR:
CPL P1.0 ;port of the test
ANL AUXINTIF,#NOT T4IF ;clear the symbol of interrupt
RETI

MAIN:
MOV SP,#3FH
MOV T4L,#66H ;65536-11.0592M/12/1000
MOV T4H,#0FCH
MOV T4T3M,#80H ;start the timer
```c
#include "reg51.h"
#include "intrins.h"

//The operating frequency is 11.0592 MHz

sfr T4L = 0xd3;
sfr T4H = 0xd2;
sfr T4T3M = 0xd1;
sfr IE2 = 0xaf;
#define ET4 0x40
sfr AUXINTIF = 0xef;
#define T4IF 0x04
sbit P10 = P1^0;

void TM4_Isr() interrupt 20
{
P10 = !P10; //port of the test
AUXINTIF &= ~T4IF; //clear the symbol of interrupt
}

void main()
{
T4L = 0x66; //65536-11.0592M/12/1000
T4H = 0xfc;
T4T3M = 0x80; //start the timer
IE2 = ET4; //enable the interrupt of timer
EA = 1;

while (1);
}
```

### 13.6.28 Timer4 (External counting — set T4 as the external interrupt for falling edge)

**Assembly code**

The operating frequency is 11.0592 MHz

<table>
<thead>
<tr>
<th>Symbol</th>
<th>DATA</th>
</tr>
</thead>
<tbody>
<tr>
<td>T4L</td>
<td>0d3H</td>
</tr>
<tr>
<td>T4H</td>
<td>0d2H</td>
</tr>
<tr>
<td>T4T3M</td>
<td>0d1H</td>
</tr>
<tr>
<td>IE2</td>
<td>0afH</td>
</tr>
<tr>
<td>ET4</td>
<td>40H</td>
</tr>
<tr>
<td>AUXINTIF</td>
<td>0efH</td>
</tr>
<tr>
<td>T4IF</td>
<td>04H</td>
</tr>
</tbody>
</table>
ORG 0000H
LJMP MAIN

ORG 00A3H
LJMP TM4ISR

ORG 0100H

TM4ISR:
CPL P1.0 ;port of the test
ANL AUXINTIF, #NOT T4IF ;clear the symbol of interrupt
RETI

MAIN:
MOV SP, #3FH
MOV T4L, #0FFH
MOV T4H, #0FFH
MOV T4T3M, #0C0H ;set External counting mode and start the timer
MOV IE2, #ET4 ;enable the interrupt of timer
SETB EA
JMP $

END

C code

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

//The operating frequency is 11.0592 MHz

sfr T4L = 0xd3;
sfr T4H = 0xd2;
sfr T4T3M = 0xd1;
sfr IE2 = 0xaf;
#define ET4 0x40
sfr AUXINTIF = 0xef;
#define T4IF 0x04
sbit P10 = P1^0;

void TM4_Isr() interrupt 20
{
P10 = !P10; //port of the test
AUXINTIF &= ~T4IF; //clear the symbol of interrupt
}

void main()
{
T4L = 0xff;
T4H = 0xff;
T4T3M = 0xc0;
IE2 = ET4; //set External counting mode and start the timer
EA = 1;

while (1);
}
13.6.29 Timer4 (clock divider output)

Assembly code

The operating frequency is 11.0592 MHz

<table>
<thead>
<tr>
<th>Register</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>T4L</td>
<td>0xD3</td>
</tr>
<tr>
<td>T4H</td>
<td>0xD2</td>
</tr>
<tr>
<td>T4T3M</td>
<td>0xD1</td>
</tr>
<tr>
<td>ORG</td>
<td>0x0000</td>
</tr>
<tr>
<td>LJMP</td>
<td>MAIN</td>
</tr>
<tr>
<td>ORG</td>
<td>0x0100</td>
</tr>
<tr>
<td>MAIN:</td>
<td></td>
</tr>
<tr>
<td>MOV</td>
<td>SP,#3FH</td>
</tr>
<tr>
<td>MOV</td>
<td>T4L,#66H</td>
</tr>
<tr>
<td>MOV</td>
<td>T4H,#0FCH</td>
</tr>
<tr>
<td>MOV</td>
<td>T4T3M,#90H</td>
</tr>
<tr>
<td>JMP</td>
<td>$</td>
</tr>
</tbody>
</table>

C code

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

// The operating frequency is 11.0592 MHz

sfr T4L = 0xd3;
sfr T4H = 0xd2;
sfr T4T3M = 0xd1;

void main()
{
    T4L = 0x66; // 65536-11.0592MHz/12/1000
    T4H = 0xfc;
    T4T3M = 0x90; // enable the output of timer and start the timer

    while (1);
}
```

13.6.30 Configure Timer 4 as Baud Rate Generate of serial port 4

Assembly code

<table>
<thead>
<tr>
<th>Register</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>T4T3M</td>
<td>0xD1H</td>
</tr>
<tr>
<td>T4H</td>
<td>0xD2H</td>
</tr>
<tr>
<td>T4L</td>
<td>0xD3H</td>
</tr>
<tr>
<td>SCON</td>
<td>0x84</td>
</tr>
<tr>
<td>SBUF</td>
<td>0x085</td>
</tr>
</tbody>
</table>
IE2  DATA  0AFH
BUSY  BIT  20H.0
WPTR  DATA  21H
RPTR  DATA  22H
BUFFER  DATA  23H  ;16 bytes

ORG  0000H
LJMP MAIN
ORG  0093H
LJMP UART4_ISR

ORG  0100H
UART4_ISR:
PUSH ACC
PUSH PSW
MOV PSW,#08H
MOV A,S4CON
JNB ACC.1,CHKRI
ANL S4CON,#NOT 02H
CLR BUSY
CHKRI:
JNB ACC.0,UART4ISR_EXIT
ANL S4CON,#NOT 01H
MOV A,WPTR
ANL A,#0FH
ADD A,#BUFFER
MOV R0,A
MOV @R0,S4BUF
INC WPTR
UART4ISR_EXIT:
POP PSW
POP ACC
RETI

UART4_INIT:
MOV S4CON,#50H
MOV T4L,#0E8H  ;65536-11059200/115200/4=0FFE8H
MOV T4H,#0FFH
MOV T4T3M,#0A0H
CLR BUSY
MOV WPTR,#00H
MOV RPTR,#00H
RET

UART4_SEND:
JB BUSY,$
SETB BUSY
MOV S4BUF,A
RET

UART4_SENDSTR:
CLR A
MOVC A,@A+DPTR
JZ SEND4END
LCALL UART4_SEND
INC DPTR
JMP UART4_SENDSTR

SEND4END:
RET

MAIN:
MOV SP,#3FH
LCALL UART4_INIT
MOV IE2,#10H
SETB EA
MOV DPTR,#STRING
LCALL UART4_SENDSTR

LOOP:
MOV A,RPTR
XRL A,WPTR
ANL A,#0FH
JZ LOOP
MOV A,RPTR
ANL A,#0FH
ADD A,#BUFFER
MOV R0,A
MOV A,@R0
LCALL UART4_SEND
INC RPTR
JMP LOOP

STRING: DB 'Uart Test !',0DH,0AH,00H
END

C code
#include "reg51.h"
#include "intrins.h"
#define FOSC 11059200UL
#define BRT (65536 - FOSC / 115200 / 4)
sfr T4T3M = 0xd1;
sfr T4H = 0xd2;
sfr T4L = 0xd3;
sfr S4CON = 0x84;
sfr S4BUF = 0x85;
sfr IE2 = 0xaf;
bit busy;
char wprr;
char rprr;
char buffer[16];

void Uart4Isr() interrupt 18
{
    if (S4CON & 0x02)
    {
        S4CON &= ~0x02;
    }
```c
    busy = 0;
}  
if (S4CON & 0x01)  
{  
    S4CON &= ~0x01;
    buffer[wptr++] = S4BUF;
    wptr &= 0x0f;
}  

void Uart4Init()  
{  
    S4CON = 0x50;
    T4L = BRT;
    T4H = BRT >> 8;
    T4T3M = 0xa0;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}  

void Uart4Send(char dat)  
{  
    while (busy);  
    busy = 1;
    S4BUF = dat;  
}  

void Uart4SendStr(char *p)  
{  
    while (*p)  
    {  
        Uart4Send(*p++);
    }  
}  

void main()  
{  
    Uart4Init();
    IE2 = 0x10;
    EA = 1;
    Uart4SendStr("Uart Test !\r\n");
    while (1)  
    {  
        if (rptr != wptr)  
        {  
            Uart4Send(buffer[rptr++]);
            rptr &= 0x0f;
        }  
    }  
}  
```
14 Serial Port (UART) Communication

STC8F series microcontrollers have 4 full duplex asynchronous serial communication interfaces (serial port 1, serial port 2, serial port 3 and serial port 4). Each serial port consists of two data buffers, a shift register, a serial control register and a baud rate generator. Each serial port data buffer consists of two independent receive and transmit buffers, which can transmit and receive data simultaneously.

There are 4 modes for serial port 1 of STC8F series of microcontrollers, among them, the baud rates of two modes are variable, the baud rates of the other two modes are fixed, which can be chosen for different applications. Serial port 2, serial port 3, serial port 4 have only two modes, and their baud rates are variable. Different baud rates and different modes can be set by the software. It is flexible for the host to query the receiving or sending process, or use the interrupt method.

All the pins of serial port 1, serial port 2, serial port 3 and serial port 4 can be switched among multiple groups of ports, so that a serial port can be multiplexed into serial ports in a time-sharing manner.

14.1 Serial Port Related Registers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Address</th>
<th>Bit Address and Symbol</th>
<th>Value after reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCON</td>
<td>Serial port 1 control</td>
<td>98H</td>
<td>B7 B6 B5 B4 B3 B2 B1 B0</td>
<td>0000,0000000</td>
</tr>
<tr>
<td>SBUF</td>
<td>Serial port 1 data buffer register</td>
<td>99H</td>
<td>0000,0000000</td>
<td></td>
</tr>
<tr>
<td>S2CON</td>
<td>Serial port 2 control</td>
<td>9AH</td>
<td>0100,0000000</td>
<td></td>
</tr>
<tr>
<td>S2BUF</td>
<td>Serial port 2 data buffer register</td>
<td>9BH</td>
<td>0000,0000000</td>
<td></td>
</tr>
<tr>
<td>S3CON</td>
<td>Serial port 3 control</td>
<td>ACH</td>
<td>0000,0000000</td>
<td></td>
</tr>
<tr>
<td>S3BUF</td>
<td>Serial port 3 data buffer register</td>
<td>ADH</td>
<td>0000,0000000</td>
<td></td>
</tr>
<tr>
<td>S4CON</td>
<td>Serial port 4 control</td>
<td>84H</td>
<td>0000,0000000</td>
<td></td>
</tr>
<tr>
<td>S4BUF</td>
<td>Serial port 4 data buffer register</td>
<td>85H</td>
<td>0000,0000000</td>
<td></td>
</tr>
<tr>
<td>PCON</td>
<td>Power control register</td>
<td>87H</td>
<td>0011,0000000</td>
<td></td>
</tr>
<tr>
<td>AUXR</td>
<td>Auxiliary register 1</td>
<td>8EH</td>
<td>0000,0000001</td>
<td></td>
</tr>
<tr>
<td>AUXR2</td>
<td>Auxiliary register 2</td>
<td>97H</td>
<td>0000,0000000</td>
<td></td>
</tr>
<tr>
<td>SADDR</td>
<td>Serial port address register</td>
<td>A9H</td>
<td>0000,0000000</td>
<td></td>
</tr>
<tr>
<td>SADEN</td>
<td>Serial port address enable</td>
<td>B9H</td>
<td>0000,0000000</td>
<td></td>
</tr>
</tbody>
</table>

14.2 Serial Port 1

Serial port 1 control register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCON</td>
<td>98H</td>
<td>SM0/FE</td>
<td>SM1</td>
<td>SM2</td>
<td>REN</td>
<td>TB8</td>
<td>RB8</td>
<td>TI</td>
<td>RI</td>
</tr>
</tbody>
</table>

SM0/FE: If the SMOD0 bit in the PCON register is 1, this bit is the frame error detection flag. When the UART detects an invalid stop bit during reception, it is set by the UART receiver and must be cleared by...
software. If SMOD0 bit in PCON register is 0, this bit and SM1 specify the communication mode of serial port 1 as shown in the following table:

<table>
<thead>
<tr>
<th>SM0</th>
<th>SM1</th>
<th>the communication mode of serial port 1</th>
<th>Function description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Mode 0</td>
<td>synchronous shift serial mode</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Mode 1</td>
<td>8-bit UART, whose baud-rate is variable</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Mode 2</td>
<td>9-bit UART, whose baud-rate is fixed</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Mode 3</td>
<td>9-bit UART, whose baud-rate is variable</td>
</tr>
</tbody>
</table>

SM2: Mode 2 or mode 3 multi-machine communication enable control bit. When serial port 1 adopts mode 2 or mode 3, if the SM2 bit is 1 and the REN bit is 1, the receiver is in the Address Frame Filter state. In this case, the received 9th bit (RB8) can be used to filter the address frame. If RB8 = 1, it indicates that the frame is an address frame, the address information can enter SBUF and the RI is 1, and then the address information is compared in the interrupt service routine. If RB8 = 0, it indicates that the frame is not an address frame, which should be discarded and keep RI = 0. In mode 2 or mode 3, if the SM2 bit is 0 and the REN bit is 1, the receiver is in a state where the address frame filtering is disabled. The received message can enter SBUF regardless of whether RB8 is 0 or 1, and make RI = 1. Here, RB8 is usually used as a check bit. Mode 1 and mode 0 are non-multi-machine communication modes. In these two modes, SM2 should be set to 0.

REN: Receive enable control bit.
0: disable serial port receive data.
1: enable serial port receive data.

TB8: The 9th bit be transmitted for serial port 1 in mode 2 and 3. It can be set or cleared by software. It is not used in mode 0 and mode 1.

RB8: The 9th bit received for serial port 1 in mode 2 and 3 which is usually used as a check bit or address frame/data frame flag. It is not used in mode 0 and mode 1.

TI: Transmit interrupt request flag of serial port 1. In mode 0, when the transmission of the 8th bit is completed, TI is set by the hardware automatically and requests the interrupt to the CPU. After the CPU responds to the interrupt, TI must be cleared by software. In other modes, TI is set by the hardware automatically at the start of the stop bit transmission and requests interrupts to the CPU. TI must be cleared by software after the interrupt is serviced.

RI: Receive interrupt request flag of serial port 1. In mode 0, when the serial port receives the 8th bit of datum, RI is set by the hardware automatically and requests interrupt to the CPU. After the interrupt is serviced, RI must be cleared by software. In other modes, RI is set by hardware automatically at the middle of stop bit the serial port received, and requests the interrupt to the CPU. After the interrupt is serviced, RI must be cleared by software.

### Serial port 1 data buffer

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SBUF</td>
<td>99H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SBUF: It is used as the buffer in transmission and reception. SBUF is actually two buffers, read buffer and write buffer. Two operations correspond to two different registers, one is write-only register (write buffer), the other is read-only register (read buffer). Actually the CPU reads serial receive buffer when reads SBUF, and writes to the SBUF will trigger the serial port to start sending data.
Power control register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCON</td>
<td>87H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SMOD: double Baud rate of serial port 1 control bit.
- 0: disable double baud rate of the uart1.
- 1: enable double baud rate of the uart1.

SMOD0: Frame error detection control bit.
- 0: No frame error detection function, SCON.7 is SM0 function.
- 1: enable frame error detection function. The function of SM0/FE is FE.

Auxiliary register 1

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUXR</td>
<td>8EH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T0x12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UART_M0x6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2_C/T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2x12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXTRAM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1ST2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

UART_M0x6: Baud rate select bit of UART1 while it works in mode 0.
- 0: The baud-rate of UART in mode 0 is SYSclk/12.
- 1: The baud-rate of UART in mode 0 is SYSclk/2.

S1ST2: Serial port 1 baud rate generator select bit.
- 0: Select Timer 1 as the baud-rate generator of UART1.
- 1: Select Timer 2 as the baud-rate generator of UART1.

Auxiliary register 2

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUXR2</td>
<td>97H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TXLNRX: Serial port 1 broadcast mode control bit.
- 0: Serial port 1 is in normal mode.
- 1: Serial port 1 is in broadcast mode. That is, the RxD pin status is output to TxD pin in real time. The TxD external pin can amplify and output the RxD pin signal in real time.

14.2.1 Serial Port 1 Mode 0

When mode 0 is selected for serial port 1, the serial port 1 operates in synchronous shift register mode. When the serial port mode 0 communication speed setting bit UART_M0x6 is 0, the baud rate is fixed to SYSclk/12. When UART_M0x6 is 1, the baud rate is fixed to SYSclk/2. RxD is used as serial communication data pin, TxD is used as synchronous shift pulse output pin. 8-bit data are transmitted and received, LSB first.

Transmission process of mode 0: Transmission is initiated by any instruction that write data to SBUF. The 8-bit datum is output from the RxD pin at the baud rate of SYSclk/12 or SYSclk/2 (determined by the UART_M0x6 divided by 12 or 2), from LSB to MSB. The interrupt flag TI is set when transmission is completed. The TxD pin outputs the synchronous shift pulse signal. When the write signal is valid, the transmit control signal SEND is active (high) one clock apart, allowing RxD to send data while allowing the TxD output the synchronous shift pulse. When a frame (8 bits) of datum is sent, all control signals are restored to the original status, and only TI keeps high level and keeps the interrupt request status. TI must be cleared by
software before sending data again.

Receiving process of mode 0: Receiving is initiated by setting REN=1 and the receive interrupt request flag RI=0. After starting the receive process, RxD is the serial data input pin and TxD is the synchronous pulse output pin. The serial receiving baud rate is SYSclk/12 or SYSclk/2 (determined by UART_M0x6 is 12 or 2). After receiving a frame of data (8 bits), the control signal is reset and the interrupt flag RI is set to 1, and interrupt request status appears. RI must be cleared by software for the next receiving data.

![Diagram of transmitting data (Serial port 1 mode 0)](image1)

![Diagram of receiving data (Serial port 1 mode 0)](image2)

When operating in mode 0, SM2 must be cleared so that TB8 and RB8 bits are not affected. Since the baud rate is fixed at SYSclk/12 or SYSclk/2, no timer is required and the clock of the microcontroller is used as the synchronous shift pulse directly.

The baud rates of serial port 1 mode 0 are shown in the following table, where SYSclk is the system operating frequency:

<table>
<thead>
<tr>
<th>UART_M0x6</th>
<th>Baud rate calculation formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( \text{Baud rate} = \frac{\text{SYSclk}}{12} )</td>
</tr>
<tr>
<td>1</td>
<td>( \text{Baud rate} = \frac{\text{SYSclk}}{2} )</td>
</tr>
</tbody>
</table>
14.2.2 Serial Port 1 Mode 1

If SM0 and SM1 of SCON are set to "01" by the software, serial port 1 will work in mode 1. This is a 8-bit UART format, where a frame of information consists 10 bits: 1 start bit, 8 data bits (LSB first) and 1 stop bit. The baud rate is variable, which can be set by the software as needed. TxD is the data transmitting pin, and RxD is the data receiving pin, the serial port is a full duplex receiver/transmitter.

Transmission process of mode 1: TxD is used as data output pin when transmitting a datum. Transmission is initiated by writing SBUF. "1" is also written into the 9th bit of transmission shift register by the writing "SBUF" signal, and the TX control unit is notified to start sending. The shift register shifts the data right to TxD port to send, and shifts "0" in the left to supplement. When the highest bit of data is shifted to the output of the shift register, it is followed by the ninth bit "1", and all bits to the left of it are "0". This state condition causes the TX control unit to make the last shift output, and then disables the transmission signal "SEND" to complete the transmission of a frame of information and sets the interrupt request TI, and requests interrupt processing to CPU.

Receiving process of mode 1: After the software sets the reception enable flag REN, that is REN = 1, the receiver will detect the RxD pin signal. The receiver is ready to receive data when a "1" → "0" falling edge is detected at RxD pin, and resets the receiving counter of the baud rate generator immediately, loads 1FFH into the shift register. The received datum is shifted in from the right of the receiving shift register, the loaded 1FFH is shifted out to the left. When the start bit "0" is shifted to the left of the shift register, the RX controller shifts for the last time and completes a frame receiving. The received datum is valid only if the following two conditions are met:

· RI=0;
· SM2=0 or the stop bit received is 1.

The datum received is loaded into SBUF, the stop bit is loaded into RB8, RI flag is set to request interrupt to CPU. If the two conditions can not be met at the same time, the received data is invalid and is discarded. Regardless of the conditions are met or not, the receiver will re-test RxD pin of the "1" → "0" edge, and continue to receive the next frame. If the received datum is valid, the RI flag must be cleared by software in the interrupt service routine. Usually, SM2 is set to "0" when serial port is operating in mode 1.

![Transmitting data](image)

![Receiving data](image)
The baud rate of serial port 1 is variable. It can be generated by timer 1 or timer 2. If the timer is in 1T mode (12x speed), the corresponding baud rate is increased by 12 times.

The baud rate of serial port 1 mode 1 is calculated as follows, where SYSclk is the system operating frequency.

<table>
<thead>
<tr>
<th>Timer selected</th>
<th>Speed of timer</th>
<th>Baud rate calculation formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>1T</td>
<td></td>
<td>( \text{SYSclk} \div \text{baud rate} )</td>
</tr>
<tr>
<td>Timer 2</td>
<td></td>
<td>( \text{SYSclk} \div 12 \times \text{baud rate} )</td>
</tr>
<tr>
<td>12T</td>
<td></td>
<td>( \text{SYSclk} \div 12 \times 32 \times \text{baud rate} )</td>
</tr>
<tr>
<td>Timer 1 mode 0</td>
<td></td>
<td>( \text{SYSclk} \div 32 \times \text{baud rate} )</td>
</tr>
<tr>
<td>1T</td>
<td></td>
<td>( \text{SYSclk} \div 32 \times \text{baud rate} )</td>
</tr>
<tr>
<td>Timer 1 mode 2</td>
<td></td>
<td>( \text{SYSclk} \div 32 \times \text{baud rate} )</td>
</tr>
</tbody>
</table>

The reload value of the timers corresponding to the common frequency and the common baud rate are as following.

<table>
<thead>
<tr>
<th>Frequency (MHz)</th>
<th>Baud rate</th>
<th>Timer 2</th>
<th>Timer 1 mode 0</th>
<th>Timer 1 mode 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1T mode</td>
<td>12T mode</td>
<td>1T mode</td>
</tr>
<tr>
<td>115200</td>
<td>FF8E8H</td>
<td>FFEE8H</td>
<td>FFEE8H</td>
<td>FEE8H</td>
</tr>
<tr>
<td>57600</td>
<td>FFD0H</td>
<td>FFFCH</td>
<td>FFD0H</td>
<td>FFFCH</td>
</tr>
<tr>
<td>38400</td>
<td>FFB8H</td>
<td>FFFAH</td>
<td>FFB8H</td>
<td>FFFAH</td>
</tr>
<tr>
<td>19200</td>
<td>FF70H</td>
<td>FFF4H</td>
<td>FF70H</td>
<td>FFF4H</td>
</tr>
<tr>
<td>9600</td>
<td>FEE0H</td>
<td>FFE8H</td>
<td>FEE0H</td>
<td>FFE8H</td>
</tr>
<tr>
<td>115200</td>
<td>FFD8H</td>
<td>-</td>
<td>FFD8H</td>
<td>-</td>
</tr>
<tr>
<td>57600</td>
<td>FFB0H</td>
<td>-</td>
<td>FFB0H</td>
<td>-</td>
</tr>
<tr>
<td>38400</td>
<td>FF8E8H</td>
<td>FFF6H</td>
<td>FF8E8H</td>
<td>FFF6H</td>
</tr>
<tr>
<td>19200</td>
<td>FF10H</td>
<td>FFECH</td>
<td>FF10H</td>
<td>FFECH</td>
</tr>
<tr>
<td>9600</td>
<td>FE20H</td>
<td>FFD8H</td>
<td>FE20H</td>
<td>FFD8H</td>
</tr>
<tr>
<td>115200</td>
<td>FFD0H</td>
<td>FFFCH</td>
<td>FFD0H</td>
<td>FFFCH</td>
</tr>
<tr>
<td>57600</td>
<td>FFA0H</td>
<td>FFF8H</td>
<td>FFA0H</td>
<td>FFF8H</td>
</tr>
<tr>
<td>38400</td>
<td>FF70H</td>
<td>FFF4H</td>
<td>FF70H</td>
<td>FFF4H</td>
</tr>
<tr>
<td>19200</td>
<td>FEE0H</td>
<td>FFE8H</td>
<td>FEE0H</td>
<td>FFE8H</td>
</tr>
<tr>
<td>9600</td>
<td>FDC0H</td>
<td>FFD0H</td>
<td>FDC0H</td>
<td>FFD0H</td>
</tr>
</tbody>
</table>

Nantong guoxin Microelectronics Co., Ltd. Tel: 0513-5501 2928/2929/2966 Fax: 0513-5501 2926/2956/2947 - 273 -
14.2.3 Serial Port 1 Mode 2

If the two bits of SM0 and SM1 are 10, serial port 1 operates in mode 2. Serial port 1 operating mode 2 is a 9-bit data asynchronous communication UART. One frame information consists of 11 bits: 1 start bit, 8 data bits (LSB first), 1 programmable bit (9th bit) and 1 stop bit. The transmit programmable bit (9th bit) is supplied by TB8 in SCON, which can be configured as either 1 or 0 by software. The odd/even parity bit P in the PSW can be loaded into TB8. Not only can TB8 be used as either a multi-machine communication address/data flag, but also it can be used as datum parity check bit. The ninth bit is received into RB8 of SCON. TxD is the transmitting pin, and RxD is the receiving pin, the serial port is a full duplex receiver/transmitter.

The baud rate of mode 2 is fixed to the system clock divided by 64 or 32 depending on the value of SMOD in PCON.

The baud rate of serial port 1 mode 2 is shown in the following table, where SYSclk is the system operating frequency.

<table>
<thead>
<tr>
<th>SMOD</th>
<th>Baud rate calculation formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>( \text{baud rate} = \frac{\text{SYSclk}}{64} )</td>
</tr>
<tr>
<td>1</td>
<td>( \text{baud rate} = \frac{\text{SYSclk}}{32} )</td>
</tr>
</tbody>
</table>

Except that the source of the baud rate is slightly different, and the 9th bit of the shift register supplied by TB8 while being sent is different, the functional and structure of mode 2 and mode 1 are basically the same, the receiving/sending operation and timing of mode 2 and mode 1 are also basically the same.

After the receiver receives a frame of information, the following conditions must be met at the same time.

- RI=0
- SM2=0 or SM2=1 and the 9th bit received RB8=1.

Only when the two conditions above are satisfied at the same time, the data received in shift register is loaded into SBUF and RB8. The RI flag is set to 1, and the interrupt request processing is requested to CPU. If one of the above conditions is not satisfied, the data just received in the shift register is invalid and is discarded, and the RI is not set. Regardless of the above conditions are met or not, the receiver again begins to detect the RxD pin hopping information to receive the next frame of information. In mode 2, the received stop bit is not related to SBUF, RB8 and RI.

It provides for the convenience of multi-machine communication by setting SM2, TB8 of SCON and communication protocol using the software.

Write SBUF

<table>
<thead>
<tr>
<th>TxD</th>
<th>Start</th>
<th>D0</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
<th>D6</th>
<th>D7</th>
<th>TB8</th>
<th>Stop</th>
</tr>
</thead>
</table>

Transmitting data (Serial port 1 mode 2)
14.2.4 Serial Port 1 Mode 3

If the two bits of SM0 and SM1 are 11, serial port 1 operates in mode 3. Serial port 1 operating mode 3 is a 9-bit data asynchronous communication UART. One frame information consists of 11 bits: 1 start bit, 8 data bits (LSB first), 1 programmable bit (9th bit) and 1 stop bit. The transmit programmable bit (9th bit) is supplied by TB8 in SCON, which can be configured as either 1 or 0 by software. The odd/even parity bit P in the PSW can be loaded into TB8. Not only can TB8 be used as either a multi-machine communication address/data flag, but also it can be used as datum parity check bit. The ninth bit is received into RB8 of SCON. TxD is the transmitting pin, and RxD is the receiving pin, the serial port is a full duplex receiver/transmitter.

Except that the 9th bit of the shift register supplied by TB8 while is being sent is different, the functional and structure of mode 3 and mode 1 are basically the same, the receiving / sending operation and timing of mode 3 and mode 1 are also basically the same.

After the receiver receives a frame of information, the following conditions must be met at the same time.

- RI=0
- SM2=0 or SM2=1 and the 9th bit received RB8=1.

Only when the two conditions above are satisfied at the same time, the data received in shift register is loaded into SBUF and RB8. The RI flag is set to 1, and the interrupt request processing is requested to CPU. If one of the above conditions is not satisfied, the data just received in the shift register is invalid and is discarded, and the RI is not set. Regardless of the above conditions are met or not, the receiver again begins to detect the RxD pin hopping information to receive the next frame of information. In mode 3, the received stop bit is not related to SBUF, RB8 and RI.

It provides for the convenience of multi-machine communication by setting SM2, TB8 of SCON and communication protocol using the software.

Transmitting data (Serial port 1 mode 3)
14.2.5 Automatic Address Recognition

Serial port 1 slave address control registers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SADDR</td>
<td>A9H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SADEN</td>
<td>B9H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SADDR: Slave address register
SADEN: Slave address mask register

The automatic address recognition function is typically used in the field of multi-machine communications. Its main principle is that the slave system identifies the address information from the master serial port data stream through the hardware comparison function. The address of the slave is set by the registers SADDR and SADEN. The hardware filters the slave address automatically. The hardware will generate a serial port interrupt when the slave address information from the master matches the slave address set by the slave. Otherwise, the hardware will discard the serial port data automatically without any interruption. When a number of slaves in Idle mode are connected together, only the slave that matches the slave address will wake up from Idle mode. Then the power consumption of the slave MCU reduces greatly. Constantly entering the serial port interrupt which reduces the system execution efficiency can be avoided even if the slave is in normal operation.

To use the automatic address recognition feature of the serial port, mode 2 or mode 3 of the serial port of the MCU that participates in communication is selected. Usually the mode 3 with variable baud rate is selected because the baud rate of mode 2 is fixed, and it is inconvenient to adjust. And SM2 bit of slave SCON is set to 1. The 9th bit which is stored in RB8 of the 9-bit data in serial port working in mode 2 or 3 is the address/data flag. When the 9th bit is 1, it indicates the previous 8-bit datum stored in SBUF is the address information. If SM2 is set to 1, the slave MCU will filter out non-address data whose 9th bit is 0 automatically while the address data whose 9th bit is 1 in SBUF will automatically be matched with the address set in SADDR and SADEN. If the address matches, RI will be set to "1" and an interrupt will occur. Otherwise the received serial data is discarded.

The slave address is set by two registers, SADDR and SADEN. SADDR is the slave address register, where the slave address is stored. SADEN is the slave address mask register, which is used to set the ignore bit.
in the address information. The setting method is as follows.

For example
SADDR = 11001010
SADEN = 10000001
Then the matched address is 1xxxxxx0
That is, as long as bit 0 is 0 and bit 7 is 1 in the address data sent by the master, the address can be
matched with the local address.

Another example
SADDR = 11001010
SADEN = 00001111
Then the matched address is xxxx1010
That is, as long as the low 4 bits are 1010 in the address data sent by the master, the address can be
matched with the local address. The high 4 bits are ignored.

The Broadcast Address (FFH) can be used by the master select all the slaves simultaneously for
communication.

14.3 Serial Port 2

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2CON</td>
<td>9AH</td>
<td>-</td>
<td>S2SM2</td>
<td>S2REN</td>
<td>S2TB8</td>
<td>S2RB8</td>
<td>S2TI</td>
<td>S2RI</td>
<td></td>
</tr>
</tbody>
</table>

S2SM0: Serial port 2 mode select bit.

<table>
<thead>
<tr>
<th>S2SM0</th>
<th>Serial port 2 mode</th>
<th>Function description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Mode 0</td>
<td>8-bit UART, whose baud-rate is variable</td>
</tr>
<tr>
<td>1</td>
<td>Mode 1</td>
<td>9-bit UART, whose baud-rate is variable</td>
</tr>
</tbody>
</table>

S2SM2: Serial port 2 multi-machine communication control enable bit. In mode 1, if the S2SM2 bit is 1 and
the S2REN bit is 1, the receiver is in the address frame filter state. In this case, the received 9th bit
(S2RB8) can be used to filter the address frame. If S2RB8 = 1, the frame is the address frame, address
information can enter S2BUF, S2RI becomes 1, and then address can be compared in the interrupt
service routine. If S2RB8 = 0, it indicates that the frame is not an address frame and should be
discarded and keep S2RI = 0. In mode 1, if the S2SM2 bit is 0 and the S2REN bit is 1, the receiver is in
the address frame filter disabled state. Regardless of the received S2RB8 is 0 or 1, the information
received can enter into the S2BUF, and make S2RI = 1. Here, S2RB8 is usually used as check bit.
Mode 0 is non-multi-machine communication mode, where S2SM2 should be 0.

S2REN: Receive enable control bit.
0: disable serial port receive data.
1: enable serial port receive data.

S2TB8: S2TB8 is the 9th bit of datum to be sent when serial port 2 is in mode 1, which is usually used as a
parity check bit or an address frame / data frame flag. It can be set or cleared by software as required.
In mode 0, this bit is not used.

S2RB8: S2RB8 is the 9th bit of datum received when serial port 2 is in mode 1, which is usually used as a
parity check bit or an address frame / data frame flag. It can be set or cleared by software as required. In mode 0, this bit is not used.

S2TI: Transmit interrupt request flag of serial port 2. S2TI is set by the hardware automatically at the start of the stop bit transmission and requests interrupts to the CPU. S2TI must be cleared by software after the interrupt is serviced.

S2RI: Receive interrupt request flag of serial port 2. S2RI is set by hardware automatically at the middle of stop bit the serial port received, and requests the interrupt to the CPU. After the interrupt is serviced, S2RI must be cleared by software.

### Serial port 2 data register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2BUF</td>
<td>9BH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

S2BUF: It is used as the buffer in transmission and reception for serial port 2. S2BUF is actually two buffers, read buffer and write buffer. Two operations correspond to two different registers, one is write-only register (write buffer), the other is read-only register (read buffer). Actually the CPU reads serial receive buffer when reads S2BUF, and writes to the S2BUF will trigger the serial port to start sending data.

### 14.3.1 Serial Port 2 Mode 0

Serial port 2 mode 0 is 8-bit UART mode with variable baud rate, where a frame of information consists 10 bits: 1 start bit, 8 data bits (LSB first) and 1 stop bit. The baud rate is variable, which can be set by the software as needed. TxD2 is the data transmitting pin, and RxD2 is the data receiving pin, the serial port is a full duplex receiver/transmitter.

![Transmitting data (Serial port 2 mode 0)](image)

- **Write S2BUF**
- **TxD2**
- **S2TI**

**Transmitting data (Serial port 2 mode 0)**

- **Write S2CON**
- **S2REN=1, S2RI=0**
- **RxD2**
- **S2RI**

**Receiving data (Serial port 2 mode 0)**

The baud rate of serial port 2 is variable. It is generated by timer 2. If the timer is in 1T mode (12x speed), the corresponding baud rate is increased by 12 times.

The baud rate of serial port 2 mode 0 is calculated as follows, where SYSclk is the system operating frequency.
14.3.2 Serial Port 2 Mode 1

Serial port 2 operating mode 1 is a 9-bit data UART mode with variable baud rate. One frame information consists of 11 bits: 1 start bit, 8 data bits (LSB first), 1 programmable bit (9th bit) and 1 stop bit. The baud rate is variable, which can be set by the software as needed. TxD2 is the data transmitting pin, and RxD2 is the data receiving pin, the serial port is a full duplex receiver/transmitter.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3CON</td>
<td>ACH</td>
<td>S3SM0</td>
<td>S3ST3</td>
<td>S3SM2</td>
<td>S3REN</td>
<td>S3TB8</td>
<td>S3RB8</td>
<td>S3TI</td>
<td>S3RI</td>
</tr>
</tbody>
</table>

S3SM0: Serial port 3 mode select bit.

<table>
<thead>
<tr>
<th>S3SM0</th>
<th>Serial port 3 mode</th>
<th>Function description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Mode 0</td>
<td>8-bit UART, whose baud-rate is variable</td>
</tr>
<tr>
<td>1</td>
<td>Mode 1</td>
<td>9-bit UART, whose baud-rate is variable</td>
</tr>
</tbody>
</table>

The baud rate calculation formula of serial port 2 mode 1 is exactly the same as that of mode 0. Please refer to the mode 0 baud rate calculation formula.

14.4 Serial Port 3

Serial port 3 control register
S3ST3: Serial port 3 baud rate generator select bit.
  0: Select Timer 2 as the baud-rate generator of UART3.
  1: Select Timer 3 as the baud-rate generator of UART3.

S3SM2: Serial port 3 multi-machine communication control enable bit. In mode 1, if the S3SM2 bit is 1 and the S3REN bit is 1, the receiver is in the address frame filter state. In this case, the received 9th bit (S3RB8) can be used to filter the address frame. If S3RB8 = 1, the frame is the address frame, address information can enter S3BUF, S3RI becomes 1, and then address can be compared in the interrupt service routine. If S3RB8 = 0, it indicates that the frame is not an address frame and should be discarded and keep S3RI = 0. In mode 1, if the S3SM2 bit is 0 and the S3REN bit is 1, the receiver is in the address frame filter disabled state. Regardless of the received S3RB8 is 0 or 1, the information received can enter into the S3BUF, and make S3RI = 1. Here, S3RB8 is usually used as check bit. Mode 0 is non-multi-machine communication mode, where S3SM2 should be 0.

S3REN: Receive enable control bit.
  0: disable serial port receive data.
  1: enable serial port receive data.

S3TB8: S3TB8 is the 9th bit of datum to be sent when serial port 3 is in mode 1, which is usually used as a parity check bit or an address frame / data frame flag. It can be set or cleared by software as required. In mode 0, this bit is not used.

S3RB8: S3RB8 is the 9th bit of datum received when serial port 3 is in mode 1, which is usually used as a parity check bit or an address frame / data frame flag. It can be set or cleared by software as required. In mode 0, this bit is not used.

S3TI: Transmit interrupt request flag of serial port 3. S3TI is set by the hardware automatically at the start of the stop bit transmission and requests interrupts to the CPU. S3TI must be cleared by software after the interrupt is serviced.

S3RI: Receive interrupt request flag of serial port 3. S3RI is set by hardware automatically at the middle of stop bit the serial port received, and requests the interrupt to the CPU. After the interrupt is serviced, S3RI must be cleared by software.

**Serial port 3 data register**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3BUF</td>
<td>ADH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

S3BUF: It is used as the buffer in transmission and reception for serial port 3. S3BUF is actually two buffers, read buffer and write buffer. Two operations correspond to two different registers, one is write-only register (write buffer), the other is read-only register (read buffer). Actually the CPU reads serial receive buffer when reads S3BUF, and writes to the S3BUF will trigger the serial port to start sending data.

### 14.4.1 Serial Port 3 Mode 0

Serial port 3 mode 0 is 8-bit UART mode with variable baud rate, where a frame of information consists 10 bits: 1 start bit, 8 data bits (LSB first) and 1 stop bit. The baud rate is variable, which can be set by the software as needed. TxD3 is the data transmitting pin, and RxD3 is the data receiving pin, the serial port is a
full duplex receiver/transmitter.

Write
S3BUF

TxD3
Start D0 D1 D2 D3 D4 D5 D6 D7 Stop

S3TI

Transmitting data (Serial port 3 mode 0)

Write
S3CON

S3REN=1, S3RI=0

RxD3
Start D0 D1 D2 D3 D4 D5 D6 D7 Stop

S3RI

Receiving data (Serial port 3 mode 0)

The baud rate of serial port 3 is variable. It is generated by timer 2 or timer 3. If the timer is in 1T mode (12x speed), the corresponding baud rate is increased by 12 times.

The baud rate of serial port 3 mode 0 is calculated as follows, where SYSclk is the system operating frequency.

<table>
<thead>
<tr>
<th>Timer selected</th>
<th>Speed of timer</th>
<th>Baud rate calculation formula</th>
</tr>
</thead>
</table>
| 1T Timer 2     |                | \[
| Reload value of timer 2 = 65536 \] \[ \frac{SYSclk}{4 \times \text{baud rate}} \] |
| 12T Timer 2    |                | \[
| Reload value of timer 2 = 65536 \] \[ \frac{SYSclk}{12 \times 4 \times \text{baud rate}} \] |
| 1T Timer 3     |                | \[
| Reload value of timer 2 = 65536 \] \[ \frac{SYSclk}{4 \times \text{baud rate}} \] |
| 12T Timer 3    |                | \[
| Reload value of timer 2 = 65536 \] \[ \frac{SYSclk}{12 \times 4 \times \text{baud rate}} \] |

14.4.2 Serial Port 3 Mode 1

Serial port 3 operating mode 1 is a 9-bit data UART mode with variable baud rate. One frame information consists of 11 bits: 1 start bit, 8 data bits (LSB first), 1 programmable bit (9th bit) and 1 stop bit. The baud rate is variable, which can be set by the software as needed. TxD3 is the data transmitting pin, and RxD3 is the data receiving pin, the serial port is a full duplex receiver/transmitter.
The baud rate calculation formula of serial port 3 mode 1 is exactly the same as that of mode 0. Please refer to the mode 0 baud rate calculation formula.

### 14.5 Serial Port 4

#### Serial port 4 control register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>S4CON</td>
<td>84H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4SM0</td>
<td>S4SM0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4ST4</td>
<td>S4ST4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4SM2</td>
<td>S4SM2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4REN</td>
<td>S4REN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4TB8</td>
<td>S4TB8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4RB8</td>
<td>S4RB8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4TI</td>
<td>S4TI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4RI</td>
<td>S4RI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**S4SM0**: Serial port 4 mode select bit.

<table>
<thead>
<tr>
<th>S4SM0</th>
<th>Function description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Mode 0, 8-bit UART, whose baud-rate is variable</td>
</tr>
<tr>
<td>1</td>
<td>Mode 1, 9-bit UART, whose baud-rate is variable</td>
</tr>
</tbody>
</table>

**S4ST4**: Serial port 4 baud rate generator select bit.

- 0: Select Timer 2 as the baud-rate generator of UART4.
- 1: Select Timer 4 as the baud-rate generator of UART4.

**S4SM2**: Serial port 4 multi-machine communication control enable bit. In mode 1, if the S4SM2 bit is 1 and the S4REN bit is 1, the receiver is in the address frame filter state. In this case, the received 9th bit (S4RB8) can be used to filter the address frame. If S4RB8 = 1, the frame is the address frame, address information can enter S4BUF, S4RI becomes 1, and then address can be compared in the interrupt service routine. If S4RB8 = 0, it indicates that the frame is not an address frame and should be discarded and keep S4RI = 0. In mode 1, if the S4SM2 bit is 0 and the S4REN bit is 1, the receiver is in the address frame filter disabled state. Regardless of the received S4RB8 is 0 or 1, the information received can enter into the S4BUF, and make S4RI = 1. Here, S4RB8 is usually used as check bit. Mode 0 is non-multi-machine communication mode, where S4SM2 should be 0.

**S4REN**: Receive enable control bit.

- 0: disable serial port receive data.
- 1: enable serial port receive data.

**S4TB8**: S4TB8 is the 9th bit of datum to be sent when serial port 4 is in mode 1, which is usually used as a
parity check bit or an address frame / data frame flag. It can be set or cleared by software as required. In mode 0, this bit is not used.

S4RB8: S4RB8 is the 9th bit of datum received when serial port 4 is in mode 1, which is usually used as a parity check bit or an address frame / data frame flag. It can be set or cleared by software as required. In mode 0, this bit is not used.

S4TI: Transmit interrupt request flag of serial port 4. S4TI is set by the hardware automatically at the start of the stop bit transmission and requests interrupts to the CPU. S4TI must be cleared by software after the interrupt is serviced.

S4RI: Receive interrupt request flag of serial port 4. S4RI is set by hardware automatically at the middle of stop bit the serial port received, and requests the interrupt to the CPU. After the interrupt is serviced, S4RI must be cleared by software.

**Serial port 4 data register**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>S4BUF</td>
<td>85H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

S4BUF: It is used as the buffer in transmission and reception for serial port 4. S4BUF is actually two buffers, read buffer and write buffer. Two operations correspond to two different registers, one is write-only register (write buffer), the other is read-only register (read buffer). Actually the CPU reads serial receive buffer when reads S4BUF, and writes to the S4BUF will trigger the serial port to start sending data.

### 14.5.1 Serial Port 4 Mode 0

Serial port 4 mode 0 is 8-bit UART mode with variable baud rate, where a frame of information consists of 10 bits: 1 start bit, 8 data bits (LSB first) and 1 stop bit. The baud rate is variable, which can be set by the software as needed. TxD4 is the data transmitting pin, and RxD4 is the data receiving pin, the serial port is a full duplex receiver/transmitter.

![Transmitting data (Serial port 4 mode 0)](image)

The baud rate of serial port 4 is variable. It is generated by timer 2 or timer 4. If the timer is in 1T mode
The baud rate of serial port 4 mode 0 is calculated as follows, where SYSclk is the system operating frequency.

<table>
<thead>
<tr>
<th>Timer selected</th>
<th>Speed of timer</th>
<th>Baud rate calculation formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>1T</td>
<td>reload value of timer 2 = 65536 – SYSclk</td>
<td>4 \times \text{baud rate}</td>
</tr>
<tr>
<td>Timer 2</td>
<td>12T</td>
<td>reload value of timer 2 = 65536 – SYSclk</td>
</tr>
<tr>
<td>1T</td>
<td>reload value of timer 4 = 65536 – SYSclk</td>
<td>4 \times \text{baud rate}</td>
</tr>
<tr>
<td>Timer 4</td>
<td>12T</td>
<td>reload value of timer 4 = 65536 – SYSclk</td>
</tr>
</tbody>
</table>

14.5.2 Serial Port 4 Mode 1

Serial port 4 operating mode 1 is a 9-bit data UART mode with variable baud rate. One frame information consists of 11 bits: 1 start bit, 8 data bits (LSB first), 1 programmable bit (9th bit) and 1 stop bit. The baud rate is variable, which can be set by the software as needed. TxD4 is the data transmitting pin, and RxD4 is the data receiving pin, the serial port is a full duplex receiver/transmitter.

The baud rate calculation formula of serial port 4 mode 1 is exactly the same as that of mode 0. Please refer to the mode 0 baud rate calculation formula.
14.6 Precautions for Serial Port

There are some precautions for serial ports about request for interrupt:

When the mode is 8 bits data, there will be a TI interrupt after send out the whole stop status.

Write SBUF

```
| Start | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | Stop |
```

TI

Send data

When the mode is 8 bits data, there will be a RI interrupt after receive a half of the stop bit.

Write SCON

```
| Start | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | Stop |
```

REN=1, RI=0

RI

Receive data

When the mode is 9 bits data, there will be a TI interrupt after send out the whole stop status.

Write SBUF

```
| Start | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | TB8 | Stop |
```

TI

Send data

When the mode is 9 bits data, there will be a RI interrupt after receive a half of the stop bit.

Write SCON

```
| Start | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | RB8 | Stop |
```

REN=1, RI=0

RI

Receive data
14.7 Demo code

14.7.1 Serial port 1 use timer 2 as Baud Rate Generator

**Assembly code**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUXR</td>
<td>DATA</td>
<td>8EH</td>
</tr>
<tr>
<td>T2H</td>
<td>DATA</td>
<td>0D6H</td>
</tr>
<tr>
<td>T2L</td>
<td>DATA</td>
<td>0D7H</td>
</tr>
<tr>
<td>BUSY</td>
<td>BIT</td>
<td>20H.0</td>
</tr>
<tr>
<td>WPTR</td>
<td>DATA</td>
<td>21H</td>
</tr>
<tr>
<td>RPTR</td>
<td>DATA</td>
<td>22H</td>
</tr>
<tr>
<td>BUFFER</td>
<td>DATA</td>
<td>23H</td>
</tr>
</tbody>
</table>

```
ORG 0000H
LJMP MAIN

ORG 0023H
LJMP UART_ISR

ORG 0100H

UART_ISR:
    PUSH ACC
    PUSH PSW
    MOV PSW,#08H
    JNB TI,CHKRI
    CLR TI
    CLR BUSY
    CHKRI:
        JNB RI,UARTISR_EXIT
        CLR RI
        MOV A,WPTR
        ANL A,#0FH
        ADD A,#BUFFER
    INC WPTR
    UARTISR_EXIT:
        POP PSW
        POP ACC
        RETI

UART_INIT:
    MOV SCON,#50H
    MOV T2L,#0E8H
    MOV T2H,#0FFH
    MOV AUXR,#15H
    CLR BUSY
    MOV WPTR,#00H
    MOV RPTR,#00H
    RET

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

UART_SENDSTR:
CLR A  
MOVC A,@A+DPTR  
JZ SENDEND  
LCALL UART_SEND  
INC DPTR  
JMP UART_SENDSTR  
SENDEND:
RET

MAIN:
MOV SP,#3FH  
LCALL UART_INIT  
SETB ES  
SETB EA  
MOV DPTR,#STRING  
LCALL UART_SENDSTR

LOOP:
MOV A,RPTR  
XRL A,WPTR  
ANL A,#0FH  
JZ LOOP  
MOV A,RPTR  
ANL A,#0FH  
ADD A,#BUFFER  
MOV R0,A  
MOV A,@R0  
LCALL UART_SEND  
INC RPTR  
JMP LOOP

STRING:  DB 'Uart Test !',0DH,0AH,00H

END

C code
#include "reg51.h"
#include "intrins.h"
#define FOSC 11059200UL
#define BRT (65536 - FOSC / 115200 / 4)
sfr AUXR = 0x8e;
sfr T2H = 0xd6;
sic T2L = 0xd7;
bites:busy;
chars: wptr;
schar: rptr;
schar: buffer[16];
void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
        buffer[wptr++] = SBUF;
        wptr &= 0x0f;
    }
}

void UartInit()
{
    SCON = 0x50;
    T2L = BRT;
    T2H = BRT >> 8;
    AUXR = 0x15;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}

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

void UARTsendStr(char *p)
{
    while (*p)
    {
        UARTsend(*p++);
    }
}

void main()
{
    UartInit();
    ES = 1;
    EA = 1;
    UARTsendStr("Uart Test !\r\n");

    while (1)
    {
        if (rptr != wptr)
        {
            UARTsend(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}
14.7.2 Serial port 1 use timer 1 as Baud Rate Generator (MODE 0)

**Assembly code**

<table>
<thead>
<tr>
<th>Register</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUXR</td>
<td>DATA</td>
<td>8EH</td>
</tr>
<tr>
<td>BUSY</td>
<td>BIT</td>
<td>20H</td>
</tr>
<tr>
<td>WPTR</td>
<td>DATA</td>
<td>21H</td>
</tr>
<tr>
<td>RPTR</td>
<td>DATA</td>
<td>22H</td>
</tr>
<tr>
<td>BUFFER</td>
<td>DATA</td>
<td>23H</td>
</tr>
</tbody>
</table>

;16 bytes

ORG 0000H
LJMP MAIN
ORG 0023H
LJMP UART_ISR

ORG 0100H

**UART_ISR:**

PUSH ACC
PUSH PSW
MOV PSW,#08H

JNB TI,CHKRI
CLR TI
CLR BUSY

**CHKRI:**

JNB RI,UARTISR_EXIT
CLR RI
MOV A,WPTR
ANL A,#0FH
ADD A,#BUFFER
MOV R0,A
MOV @R0,SBUF
INC WPTR

UARTISR_EXIT:

POP PSW
POP ACC
RETI

**UART_INIT:**

MOV SCON,#50H
MOV TMOD,#00H
MOV TL1,#0E8H
MOV TH1,#0FFH
SETB TR1
MOV AUXR,#40H
CLR BUSY
MOV WPTR,#00H
MOV RPTR,#00H
RET

**UART_SEND:**

JB BUSY,S
SETB BUSY
MOV SBUF,A
RET

UART_SENDSTR:
CLR A
MOVC A,#A+DPTR
JZ SENDEND
LCALL UART_SEND
INC DPTR
JMP UART_SENDSTR
SENDEND:
RET

MAIN:
MOV SP,#3FH
LCALL UART_INIT
SETB ES
SETB EA
MOV DPTR,#STRING
LCALL UART_SENDSTR

LOOP:
MOV A,RPTR
XRL A,WPTR
ANL A,#0FH
JZ LOOP
MOV A,RPTR
ANL A,#0FH
ADD A,#BUFFER
MOV R0,A
MOV A,@R0
LCALL UART_SEND
INC RPTR
JMP LOOP

STRING: DB 'Uart Test !',0DH,0AH,00H

END

C code

#include "reg51.h"
#include "intrins.h"
#define FOSC 11059200UL
#define BRT (65536 - FOSC / 115200 / 4)
sfr AUXR = 0x8e;
bit busy;
char wptr;
char rptr;
char buffer[16];

void UartIsr() interrupt 4

{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
        buffer[wptr++] = SBUF;
        wptr &= 0x0f;
    }
}

void UartInit()
{
    SCON = 0x50;
    TMOD = 0x06;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}

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

void UARTsendStr(char *p)
{
    while (*p)
    {
        UARTsend(*p++);
    }
}

void main()
{
    UartInit();
    ES = 1;
    EA = 1;
    UARTsendStr("Uart Test !\r\n");

    while (1)
    {
        if (rptr != wptr)
        {
            UARTsend(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}
14.7.3 Serial port 1 use timer 1 as Baud Rate Generator (MODE 2)

Assembly code

```
| AUXR | DATA   | 8EH |
| BUSY | BIT    | 20H |
| WPTR | DATA   | 21H |
| RPTR | DATA   | 22H |
| BUFFER| DATA | 23H |

;16 bytes

ORG 0000H
LJMP MAIN
ORG 0023H
LJMP UART_ISR

ORG 0100H

UART_ISR:
PUSH ACC
PUSH PSW
MOV PSW,#08H
JNB TI,CHKRI
CLR TI
CLR BUSY
CHKRI:
JNB RI,UART_ISR_EXIT
CLR RI
MOV A,WPTR
ANL A,#0FH
ADD A,#BUFFER
MOV R0,A
MOV @R0,SBUF
INC WPTR
UART_ISR_EXIT:
POP PSW
POP ACC
RETI

UART_INIT:
MOV SCON,#50H
MOV TMOD,#20H
MOV TL1,#0FDH ;256-11059200/115200/32=0FDH
MOV TH1,#0FDH
SETB TR1
MOV AUXR,#40H
CLR BUSY
MOV WPTR,#00H
MOV RPTR,#00H
RET
```
UART_SEND:
    JB BUSY,
    SETB BUSY
    MOV SBUF, A
    RET

UART_SENDSTR:
    CLR A
    MOVC A, @A+DPTR
    JZ SENDEND
    LCALL UART_SEND
    INC DPTR
    JMP UART_SENDSTR
SENDEND:
    RET

MAIN:
    MOV SP, #3FH
    LCALL UART_INIT
    SETB ES
    SETB EA
    MOV DPTR, #STRING
    LCALL UART_SENDSTR

LOOP:
    MOV A, RPTR
    XRL A, WPTR
    ANL A, #0FH
    JZ LOOP
    MOV A, RPTR
    ANL A, #0FH
    ADD A, #BUFFER
    MOV R0, A
    MOV A, @R0
    LCALL UART_SEND
    INC RPTR
    JMP LOOP

STRING: DB 'Uart Test !', 0DH, 0AH, 00H
END

C code
#include "reg51.h"
#include "intrins.h"
#define FOSC 11059200UL
#define BRT (256 - FOSC / 115200 / 32)
sfr AUXR = 0x8e;
bit busy;
char wptr;
char rptr;
char buffer[16];
void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
        buffer[wptr++] = SBUF;
        wptr &= 0x0f;
    }
}

void UartInit()
{
    SCON = 0x50;
    TMOD = 0x20;
    TL1 = BRT;
    TH1 = BRT;
    TR1 = 1;
    AUXR = 0x40;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}

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

void UARTsendStr(char *p)
{
    while (*p)
    {
        UARTsend(*p++);
    }
}

void main()
{
    UartInit();
    ES = 1;
    EA = 1;
    UARTsendStr("Uart Test \r\n");

    while (1)
    {
        if (rptr != wptr)
        {
            UARTsend(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
14.7.4 Serial port 2 use timer 2 as Baud Rate Generator

Assembly code

<table>
<thead>
<tr>
<th>Data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUXR</td>
<td>DATA 8EH</td>
</tr>
<tr>
<td>T2H</td>
<td>DATA 0D6H</td>
</tr>
<tr>
<td>T2L</td>
<td>DATA 0D7H</td>
</tr>
<tr>
<td>S2CON</td>
<td>DATA 9AH</td>
</tr>
<tr>
<td>S2BUF</td>
<td>DATA 9BH</td>
</tr>
<tr>
<td>IE2</td>
<td>DATA 0AFH</td>
</tr>
<tr>
<td>BUSY</td>
<td>BIT 20H.0</td>
</tr>
<tr>
<td>WPTR</td>
<td>DATA 21H</td>
</tr>
<tr>
<td>RPTR</td>
<td>DATA 22H</td>
</tr>
<tr>
<td>BUFFER</td>
<td>DATA 23H</td>
</tr>
<tr>
<td>ORG</td>
<td>0000H</td>
</tr>
<tr>
<td>LJMP</td>
<td>MAIN</td>
</tr>
<tr>
<td>ORG</td>
<td>0043H</td>
</tr>
<tr>
<td>LJMP</td>
<td>UART2_ISR</td>
</tr>
<tr>
<td>ORG</td>
<td>0100H</td>
</tr>
</tbody>
</table>

UART2_ISR:

- PUSH ACC
- PUSH PSW
- MOV PSW,#08H
- MOV A,S2CON
- JNB ACC.1,CHKRI
- ANL S2CON,#NOT 02H
- CLR BUSY

CHKRI:

- JNB ACC.0,UART2ISR_EXIT
- ANL S2CON,#NOT 01H
- MOV A,WPTR
- ANL A,#0FH
- ADD A,#BUFFER
- MOV R0,A
- MOV @R0,S2BUF
- INC WPTR

UART2ISR_EXIT:

- POP PSW
- POP ACC
- RETI

UART2_INIT:

- MOV S2CON,#50H
- MOV T2L,#0E8H
- MOV T2H,#0FFH
- MOV AUXR,#14H
- CLR BUSY
- MOV WPTR,#00H
- MOV RPTR,#00H

;65536-11059200/115200/d=0FFE8H
RET

UART2_SEND:
JB BUSY,$
SETB BUSY
MOV S2BUF,A
RET

UART2_SENDSTR:
CLR A
MOVC A,@A+DPTR
JZ SEND2END
LCALL UART2_SEND
INC DPTR
JMP UART2_SENDSTR
SEND2END:
RET

MAIN:
MOV SP,#3FH
LCALL UART2_INIT
MOV IE2,#01H
SETB EA
MOV DPTR,#STRING
LCALL UART2_SENDSTR
LOOP:
MOV A,RPTR
XRL A,WPTR
ANL A,#0FH
JZ LOOP
MOV A,RPTR
ANL A,#0FH
ADD A,#BUFFER
MOV R0,A
MOV A,@R0
LCALL UART2_SEND
INC RPTR
JMP LOOP
STRING: DB 'Uart Test !',0DH,0AH,00H
END

C code
#include "reg51.h"
#include "intrins.h"
#define FOSC 11059200UL
#define BRT (65536 - FOSC / 115200 / 4)
sfr AUXR = 0x8e;
sfr T2H = 0xd6;
sfr T2L = 0xd7;
sfr S2CON = 0x9a;
sfr S2BUF = 0x9b;
sfr IE2 = 0xaf;

bit busy;
char wptr;
char rptr;
char buffer[16];

void Uart2Isr() interrupt 8
{   
    if (S2CON & 0x02)
    {
        S2CON &= ~0x02;
        busy = 0;
    }
    if (S2CON & 0x01)
    {
        S2CON &= ~0x01;
        buffer[wptr++] = S2BUF;
        wptr &= 0x0f;
    }
}

void Uart2Init()
{   
    S2CON = 0x50;
    T2L = BRT;
    T2H = BRT >> 8;
    AUXR = 0x14;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}

void Uart2Send(char dat)
{   
    while (busy);
    busy = 1;
    S2BUF = dat;
}

void Uart2SendStr(char *p)
{   
    while (*p)
    {
        Uart2Send(*p++);
    }
}

void main()
{   
    Uart2Init();
    IE2 = 0x01;
    EA = 1;
    Uart2SendStr("Uart Test !\r\n");

    while (1)
    {
    }
if (rptr != wptr)
{
    Uart2Send(buffer[rptr++]);
    rptr &= 0x0f;
}
}

14.7.5 Serial port 3 use timer 2 as Baud Rate Generator

Assembly code

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUXR</td>
<td>DATA</td>
<td>8EH</td>
</tr>
<tr>
<td>T2H</td>
<td>DATA</td>
<td>0D6H</td>
</tr>
<tr>
<td>T2L</td>
<td>DATA</td>
<td>0D7H</td>
</tr>
<tr>
<td>S3CON</td>
<td>DATA</td>
<td>0ACH</td>
</tr>
<tr>
<td>S3BUF</td>
<td>DATA</td>
<td>0ADH</td>
</tr>
<tr>
<td>IE2</td>
<td>DATA</td>
<td>0AFH</td>
</tr>
<tr>
<td>BUSY</td>
<td>BIT</td>
<td>20H.0</td>
</tr>
<tr>
<td>WPTR</td>
<td>DATA</td>
<td>21H</td>
</tr>
<tr>
<td>RPTR</td>
<td>DATA</td>
<td>22H</td>
</tr>
<tr>
<td>BUFFER</td>
<td>DATA</td>
<td>23H</td>
</tr>
</tbody>
</table>

;16 bytes

ORG 0000H
LJMP MAIN
ORG 008BH
LJMP UART3_ISR
ORG 0100H

UART3_ISR:

PUSH ACC
PUSH PSW
MOV PSW,#08H

MOV A,S3CON
JNB ACC.1,CHKRI
ANL S3CON,#NOT 02H
CLR BUSY

CHKRI:

JNB ACC.0,UART3ISR_EXIT
ANL S3CON,#NOT 02H
MOV A,WPTR
ANL A,#0FH
ADD A,#BUFFER
MOV R0,A
MOV @R0,S3BUF
INC WPTR

UART3ISR_EXIT:

POP PSW
POP ACC
RETI

UART3_INIT:

MOV S3CON,#10H
MOV T2L,#08EH

;65536-11059200/115200/4=0FFE8H
MOV T2H,#0FFH
MOV AUXR,#14H
CLR BUSY
MOV WPTR,#00H
MOV RPTR,#00H
RET

UART3_SEND:
JB BUSY,$
SETB BUSY
MOV S3BUF,A
RET

UART3_SENDSTR:
CLR A
MOVC A,@A+DPTR
JZ SEND3END
LCALL UART3_SEND
INC DPTR
JMP UART3_SENDSTR
SEND3END:
RET

MAIN:
MOV SP,#3FH
LCALL UART3_INIT
MOV IE2,#08H
SETB EA
MOV DPTR,#STRING
LCALL UART3_SENDSTR
LOOP:
MOV A,RPTR
XRL A,WPTR
ANL A,#0FH
JZ LOOP
MOV A,RPTR
ANL A,#0FH
ADD A,#BUFFER
MOV R0,A
MOV A,@R0
LCALL UART3_SEND
INC RPTR
JMP LOOP
STRING: DB 'Uart Test !',0DH,0AH,00H
END

C code
#include "reg51.h"
#include "intrins.h"
#define FOSC 11059200UL
#define BRT (65536 - FOSC / 115200 / 4)

Nantong guoxin Microelectronics Co., Ltd. Tel: 0513-5501 2928/2929/2966 Fax: 0513-5501 2926/2956/2947 - 299 -
sfr AUXR = 0x8e;
sfr T2H = 0xd6;
sfr T2L = 0xd7;
sfr S3CON = 0xac;
sfr S3BUF = 0xad;
sfr IE2 = 0xaf;

bit busy;
char wptr;
char rptr;
char buffer[16];

void Uart3Isr() interrupt 17
{
    if (S3CON & 0x02)
    {
        S3CON &= ~0x02;
        busy = 0;
    }
    if (S3CON & 0x01)
    {
        S3CON &= ~0x01;
        buffer[wptr++] = S3BUF;
        wptr &= 0x0f;
    }
}

void Uart3Init()
{
    S3CON = 0x10;
    T2L = BRT;
    T2H = BRT >> 8;
    AUXR = 0x14;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}

void Uart3Send(char dat)
{
    while (busy);
    busy = 1;
    S3BUF = dat;
}

void Uart3SendStr(char *p)
{
    while (*p)
    {
        Uart3Send(*p++);
    }
}

void main()
{
    Uart3Init();
    IE2 = 0x08;
}
EA = 1;
Uart3SendStr("Uart Test !n");

while (1)
{
    if (rptr != wptr)
    {
        Uart3Send(buffer[rptr++]);
        rptr &= 0x0f;
    }
}

14.7.6 Serial port 3 use timer 3 as Baud Rate Generator

Assembly code

<table>
<thead>
<tr>
<th>Register</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>T4T3M</td>
<td>DATA</td>
<td>0D1H</td>
</tr>
<tr>
<td>T4H</td>
<td>DATA</td>
<td>0D2H</td>
</tr>
<tr>
<td>T4L</td>
<td>DATA</td>
<td>0D3H</td>
</tr>
<tr>
<td>T3H</td>
<td>DATA</td>
<td>0D4H</td>
</tr>
<tr>
<td>T3L</td>
<td>DATA</td>
<td>0D5H</td>
</tr>
<tr>
<td>S3CON</td>
<td>DATA</td>
<td>0ACH</td>
</tr>
<tr>
<td>S3BUF</td>
<td>DATA</td>
<td>0ADH</td>
</tr>
<tr>
<td>IE2</td>
<td>DATA</td>
<td>0AFH</td>
</tr>
<tr>
<td>BUSY</td>
<td>BIT</td>
<td>20H.0</td>
</tr>
<tr>
<td>WPTR</td>
<td>DATA</td>
<td>21H</td>
</tr>
<tr>
<td>RPTR</td>
<td>DATA</td>
<td>22H</td>
</tr>
<tr>
<td>BUFFER</td>
<td>DATA</td>
<td>23H</td>
</tr>
</tbody>
</table>

ORG 0000H
LJMP MAIN
ORG 008BH
LJMP UART3_ISR

ORG 0100H

UART3_ISR:

PUSH ACC
PUSH PSW
MOV PSW,#08H

MOV A,S3CON
JNB ACC.1,CHKRI
ANL S3CON,#NOT 02H
CLR BUSY

CHKRI:

JNB ACC.0,UART3ISR_EXIT
ANL S3CON,#NOT 02H
MOV A,WPTR
ANL A,#0FH
ADD A,#BUFFER
MOV R0,A
MOV @R0,S3BUF
INC WPTR

UART3ISR_EXIT:
Technical support: 13922809991/13922805190

POP  PSW
POP  ACC
RETI

UART3_INIT:
MOV  S3CON,#50H
MOV  T3L,#0E8H ;65536-11059200/115200/4=0FFE8H
MOV  T3H,#0FFH
MOV  T4T3M,#0AH
CLR  BUSY
MOV  WPTR,#00H
MOV  RPTTR,#00H
RET

UART3_SEND:
JB  BUSY
SETB  BUSY
MOV  S3BUF,A
RET

UART3_SENDSTR:
CLR  A
MOVC  A,@A+DPTR
JZ  SEND3END
LCALL  UART3_SEND
INC  DPTR
JMP  UART3_SENDSTR
SEND3END:
RET

MAIN:
MOV  SP,#3FH
LCALL  UART3_INIT
MOV  IE2,#08H
SETB  EA
MOV  DPTR,#STRING
LCALL  UART3_SENDSTR

LOOP:
MOV  A,RPTTR
XRL  A,WPTR
ANL  A,#0FH
JZ  LOOP
MOV  A,RPTTR
ANL  A,#0FH
ADD  A,#BUFFER
MOV  R0,A
MOV  A,@R0
LCALL  UART3_SEND
INC  RPTR
JMP  LOOP

STRING:  DB  'Uart Test !',0DH,0AH,00H

END
C code

```c
#include "reg51.h"
#include "intrins.h"
#define FOSC 11059200UL
#define BRT (65536 - FOSC / 115200 / 4)
sfr T4T3M = 0xd1;
sfr T4H = 0xd2;
sfr T4L = 0xd3;
sfr T3H = 0xd4;
sfr T3L = 0xd5;
sfr S3CON = 0xac;
sfr S3BUF = 0xad;
sfr IE2 = 0xaf;

bit busy;
char wptr;
char rptr;
char buffer[16];

void Uart3Isr() interrupt 17
{
    if (S3CON & 0x02)
    {
        S3CON &= ~0x02;
        busy = 0;
    }
    if (S3CON & 0x01)
    {
        S3CON &= ~0x01;
        buffer[wptr++] = S3BUF;
        wptr &= 0x0f;
    }
}

void Uart3Init()
{
    S3CON = 0x50;
    T3L = BRT;
    T3H = BRT >> 8;
    T4T3M = 0x0a;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}

void Uart3Send(char dat)
{
    while (busy);
    busy = 1;
    S3BUF = dat;
}

void Uart3SendStr(char *p)
{
```

while (*p)
{
    Uart3Send(*p++);
}
}

void main()
{
    Uart3Init();
    IE2 = 0x08;
    EA = 1;
    Uart3SendStr("Uart Test !\r\n");

    while (1)
    {
        if (rptr != wptr)
        {
            Uart3Send(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}

14.7.7 Serial port 4 use timer 2 as Baud Rate Generator

Assembly code

<table>
<thead>
<tr>
<th>Label</th>
<th>Type</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUXR</td>
<td>DATA</td>
<td>8EH</td>
</tr>
<tr>
<td>T2H</td>
<td>DATA</td>
<td>0D6H</td>
</tr>
<tr>
<td>T2L</td>
<td>DATA</td>
<td>0D7H</td>
</tr>
<tr>
<td>S4CON</td>
<td>DATA</td>
<td>84H</td>
</tr>
<tr>
<td>S4BUF</td>
<td>DATA</td>
<td>085H</td>
</tr>
<tr>
<td>IE2</td>
<td>DATA</td>
<td>0AFH</td>
</tr>
<tr>
<td>BUSY</td>
<td>BIT</td>
<td>20H.0</td>
</tr>
<tr>
<td>WPTR</td>
<td>DATA</td>
<td>21H</td>
</tr>
<tr>
<td>RPTR</td>
<td>DATA</td>
<td>22H</td>
</tr>
<tr>
<td>BUFFER</td>
<td>DATA</td>
<td>23H</td>
</tr>
</tbody>
</table>

;16 bytes

ORG 0000H
LJMP MAIN
ORG 0093H
LJMP UART4_ISR

ORG 0100H

UART4_ISR:

PUSH ACC
PUSH PSW
MOV PSW,#08H

MOV A,S4CON
JNB ACC.1,CHKRI
ANL S4CON,#NOT 02H
CLR BUSY

CHKRI:

JNB ACC.0,UART4ISR_EXIT
ANL S4CON,#NOT 01H
MOV A,WPTR
ANL A,#0FH
ADD A,#BUFFER
MOV R0,A
MOV @R0,S4BUF
INC WPTR

UART4ISR_EXIT:
POP PSW
POP ACC
RETI

UART4_INIT:
MOV S4CON,#10H
MOV T2L,#0E8H ;65536-11059200/115200/4=0FFE8H
MOV T2H,#0FFH
MOV AUXR,#14H
CLR BUSY
MOV WPTR,#00H
MOV RPTR,#00H
RET

UART4_SEND:
JB BUSY,S
SETB BUSY
MOV S4BUF,A
RET

UART4_SENDSTR:
CLR A
MOVC A,@A+DPTR
JZ SEND4END
LCALL UART4_SEND
INC DPTR
JMP UART4_SENDSTR
SEND4END:
RET

MAIN:
MOV SP,#3FH
LCALL UART4_INIT
MOV IE2,#10H
SETB EA
MOV DPTR,#STRING
LCALL UART4_SENDSTR

LOOP:
MOV A,RPTR
XRL A,WPTR
ANL A,#0FH
JZ LOOP
MOV A,RPTR
ANL A,#0FH
ADD A,#BUFFER
MOV R0,A
MOV A,@R0
LCALL UART4_SEND
INC RPTR
JMP LOOP

STRING: DB 'Uart Test !',0DH,0AH,00H
END

C code

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

#define FOSC 11059200UL
#define BRT (65536 - FOSC / 115200 / 4)
sfr AUXR = 0x8e;
sfr T2H = 0xd6;
sfr T2L = 0xd7;
sfr S4CON = 0x84;
sfr S4BUF = 0x85;
sfr IE2 = 0xaf;

bit busy;
char wptr;
char rptr;
char buffer[16];

void Uart4Isr() interrupt 18
{
    if (S4CON & 0x02)
    {
        S4CON &= ~0x02;
        busy = 0;
    }
    if (S4CON & 0x01)
    {
        S4CON &= ~0x01;
        buffer[wptr++] = S4BUF;
        wptr &= 0x0f;
    }
}

void Uart4Init()
{
    S4CON = 0x10;
    T2L = BRT;
    T2H = BRT >> 8;
    AUXR = 0x14;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}

void Uart4Send(char dat)
{
    while (busy);
    busy = 1;
}
S4BUF = dat;
}

void Uart4SendStr(char *p)
{
    while (*p)
    {
        Uart4Send(*p++);
    }
}

void main()
{
    Uart4Init();
    IE2 = 0x10;
    EA = 1;
    Uart4SendStr("Uart Test !\r\n");

    while (1)
    {
        if (rptr != wptr)
        {
            Uart4Send(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}

14.7.8 Serial port 4 use timer 4 as Baud Rate Generator

Assembly code

<table>
<thead>
<tr>
<th>Register</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>T4T3M</td>
<td>DATA</td>
<td>0D1H</td>
</tr>
<tr>
<td>T4H</td>
<td>DATA</td>
<td>0D2H</td>
</tr>
<tr>
<td>T4L</td>
<td>DATA</td>
<td>0D3H</td>
</tr>
<tr>
<td>T3H</td>
<td>DATA</td>
<td>0D4H</td>
</tr>
<tr>
<td>T3L</td>
<td>DATA</td>
<td>0D5H</td>
</tr>
<tr>
<td>S4CON</td>
<td>DATA</td>
<td>84H</td>
</tr>
<tr>
<td>S4BUF</td>
<td>DATA</td>
<td>085H</td>
</tr>
<tr>
<td>IE2</td>
<td>DATA</td>
<td>0AFH</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Register</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUSY</td>
<td>BIT</td>
<td>20H.0</td>
</tr>
<tr>
<td>WPTR</td>
<td>DATA</td>
<td>21H</td>
</tr>
<tr>
<td>RPTR</td>
<td>DATA</td>
<td>22H</td>
</tr>
<tr>
<td>BUFFER</td>
<td>DATA</td>
<td>23H</td>
</tr>
</tbody>
</table>

;16 bytes

ORG 0000H
LJMP MAIN
ORG 0093H
LJMP UART4_ISR

ORG 0100H

UART4_ISR:

PUSH ACC
PUSH PSW
MOV PSW,#08H
Technical support: 13922809991/13922805190

MOV A,S4CON
JNB ACC.1.CHKRI
ANL S4CON,#NOT 02H
CLR BUSY

CHKRI:
JNB ACC.0,UART4ISR_EXIT
ANL S4CON,#NOT 01H
MOV A,WPTR
ANL A,#0FH
ADD A,#BUFFER
MOV R0,A
MOV @R0,S4BUF
INC WPTR

UART4ISR_EXIT:
POP PSW
POP ACC
RETI

UART4_INIT:
MOV S4CON,#50H
MOV T4L,#0E8H ;65536-11059200/115200/4=0FFE8H
MOV T4H,#0FFH
MOV T4T3M,#0A0H
CLR BUSY
MOV WPTR,#00H
MOV RPTR,#00H
RET

UART4_SEND:
JB BUSYS
SETB BUSY
MOV S4BUF,A
RET

UART4_SENDSTR:
CLR A
MOVC A,@A+DPTR
JZ SEND4END
LCALL UART4_SEND
INC DPTR
JMP UART4_SENDSTR

SEND4END:
RET

MAIN:
MOV SP,#3FH
LCALL UART4_INIT
MOV IE2,#10H
SETB EA
MOV DPTR,#STRING
LCALL UART4_SENDSTR

LOOP:
MOV A,RPTR
XRL A,WPTR
ANL A,#0FH
JZ LOOP
MOV A,RPTR
ANL A,#0FH
ADD A,#BUFFER
MOV R0,A
MOV A,@R0
LCALL UART4_SEND
INC RPTR
JMP LOOP

STRING: DB 'Uart Test !',0DH,0AH,00H
END

C code
#include "reg51.h"
#include "intrins.h"
#define FOSC 11059200UL
#define BRT (65536 - FOSC / 115200 / 4)
sfr T4T3M = 0xd1;
sfr T4H = 0xd2;
sfr T4L = 0xd3;
sfr T3H = 0xd4;
sfr T3L = 0xd5;
sfr S4CON = 0x84;
sfr S4BUF = 0x85;
sfr IE2 = 0xaf;
bit busy;
char wptr;
char rptr;
char buffer[16];

void Uart4Isr() interrupt 18
{
    if (S4CON & 0x02)
    {
        S4CON &= ~0x02;
        busy = 0;
    }
    if (S4CON & 0x01)
    {
        S4CON &= ~0x01;
        buffer[wptr++] = S4BUF;
        wptr &= 0x0f;
    }
}

void Uart4Init()
{
    S4CON = 0x50;
    T4L = BRT;
    T4H = BRT >> 8;
    T4T3M = 0xa0;
wptr = 0x00;
rtpr = 0x00;
busy = 0;
}

void Uart4Send(char dat)
{
    while (busy);  
    busy = 1;        
    S4BUF = dat;    
}

void Uart4SendStr(char *p)
{
    while (*p)
    {
        Uart4Send(*p++);
    }
}

void main()
{
    Uart4Init();
    IE2 = 0x10;    
    EA = 1;       
    Uart4SendStr("Uart Test !

while (1)
{
    if (rptr != wptr)
    {
        Uart4Send(buffer[rptr++]);
        rptr &= 0x0f;
    }
}
}
Comparator, brown-out detection, internal fixed comparison voltage

STC8 series microcontroller integrates a comparator. The positive side of the comparator can be either the P3.7 port or the ADC’s analog input channel, while the negative side can be P3.6 or the internal REFV voltage (internal fixed comparison voltage) after the internal BandGap passes the OP.

The comparator has two stages of programmable filter, analog filtering and digital filtering. Analog filtering can filter the glitches in the comparison input signal, and the digital filter can wait for the input signal to be more stable before comparing. The comparison result can be obtained directly by reading the internal register bits, and the comparator result can also be forwarded or inverted to the external port. Outputting the comparison result to the external port can be used as the trigger signal and feedback signal of external events, which can expand the application range of comparison.

15.1 Internal Structure of Comparator

![Comparator Diagram]

15.2 Registers related to comparator

<table>
<thead>
<tr>
<th>symbol</th>
<th>description</th>
<th>address</th>
<th>Bit address and sign</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>B7  B6  B5  B4  B3  B2  B1  B0</td>
<td></td>
</tr>
<tr>
<td>CMPCR1</td>
<td>Compare register 1</td>
<td>E6H</td>
<td>CMPEN  CMPIF  PIE  NIE  PIS  NIS</td>
<td>CMPOE  CMPRES</td>
</tr>
<tr>
<td>CMPCR2</td>
<td>Compare register 2</td>
<td>E7H</td>
<td>INVCMPRO  DISFLT</td>
<td>LCDTY[5-0]</td>
</tr>
</tbody>
</table>

**Compare register 1**

<table>
<thead>
<tr>
<th>symbol</th>
<th>address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPCR1</td>
<td>E6H</td>
<td>CMPEN</td>
<td>CMPIF</td>
<td>PIE</td>
<td>NIE</td>
<td>PIS</td>
<td>NIS</td>
<td>CMPOE</td>
<td>CMPRES</td>
</tr>
</tbody>
</table>

**CMPCR**: enable the module of comparator
- 0: close the function of comparision
- 1: open the function of comparision

**CMPIF**: symbol of comparator interruption. When PIE or NIE was enable, there will be the sign of interruption will automatically set as CMPIF 1, and request interruption to CPU. This flag should be cleared by users.

**PIE**: Enable the rising edge interrupt of comparator

Nantong guoxin Microelectronics Co., Ltd. Tel: 0513-5501 2928/2929/2966 Fax: 0513-5501 2926/2956/2947
0: Forbid the rising edge interrupt of comparator
1: Enable the rising edge interrupt of comparator

NIE: Enable the falling edge interrupt of comparator
0: Forbid the falling edge interrupt of comparator
1: Enable the falling edge interrupt of comparator

PIS: positive selection bit of comparator
0: choose external port P3.7 as positive input source of comparator
1: The analog input of the ADC is selected as the positive input source of the comparator through the
ADC_CHS bit in ADC_CONTR.

NIS: Comparator negative selection bit
0: The internal bandgap voltage REVF after OP is selected as the negative input source of the comparator
(REVF voltage value is 1.344V. Due to manufacturing error, the actual voltage value may be between
1.34V and 1.35V).
1: Select external port P3.6 as comparator negative input source.

CMPOE: Comparator result output control bit
0: Disable comparator output
1: Enable comparator output

CMPRES: the result of comparator (read only)
0: CMP+ lower than CMP-
1: CMP+ higher than CMP-

CMPRES is a digitally filtered output signal, not a direct output from the comparator.

The register control the comparator

<table>
<thead>
<tr>
<th>symbol</th>
<th>address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPCR2</td>
<td>E7H</td>
<td>INVCMPO</td>
<td>DISFLT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

INVCMP0: control the result of comparator
0: Comparator result positive output.
1: Comparator result negative output

DISFLT: Analog filter control
0: Enable 0.1us analog filtering
1: Turn off 0.1us analog filtering

LCDTY[5:0]: Digital filter control
The digital filter function is a digital signal debounce function. When the comparison result has a rising edge
or a falling edge, the comparator detects that the changed signal must maintain the CPU clock set by LCDTY
without changing, and then considers that the data change is valid; otherwise, it will regard the signal as
unchanged.

Nantong guoxin Microelectronics Co., Ltd.  Tel: 0513-5501 2928/2929/2966  Fax: 0513-5501 2926/2956/2947  - 312 -
15.3 Demo code

15.3.1 Use of comparators (interrupt way)

**Assembly code**

```assembly
CMPCR1 DATA 0E6H
CMPCR2 DATA 0E7H

ORG 0000H
LJMP MAIN
ORG 00ABH
LJMP CMPISR

ORG 0100H

CMPISR:
PUSH ACC
ANL CMPCR1,#NOT 40H ;clear the symbol of interrupt
MOV A,CMPCR1
JB ACC.0,RSING

FALLING:
PUSH ACC
ANL CMPCR1,#NOT 40H ;forbid 0.1us filtering
ORL CMPCR1,#40H ;enable 0.1us filtering
ANL CMPCR1,#NOT 3FH ;direct output the result
ORL CMPCR1,#10H
MOV CMPCR1,#00H
ORL CMPCR1,#30H ;Enable comparator edge interrupt

MAIN:
MOV SP,#3FH

MOV CMPCR2,#00H ;Comparator result positive output.
ANL CMPCR2,#NOT 80H ;Comparator result negative output.
ORL CMPCR2,#880H
ANL CMPCR2,#NOT 40H ;forbid 0.1us filtering
ORL CMPCR2,#40H ;enable 0.1us filtering
ANL CMPCR2,#NOT 3FH ;direct output the result
ORL CMPCR2,#10H
MOV CMPCR1,#00H
ORL CMPCR1,#30H ;Enable comparator edge interrupt
```

Nantong guoxin Microelectronics Co., Ltd.  Tel: 0513-5501 2928/2929/2966  Fax: 0513-5501 2926/2956/2947 - 313 -
; ANL CMPCR1,#NOT 20H ; Disable comparator rising edge interrupt
; ORL CMPCR1,#20H ; Enable comparator rising edge interrupt
; ANL CMPCR1,#NOT 10H ; Disable comparator falling edge interrupt
; ORL CMPCR1,#10H ; Enable comparator falling edge interrupt
; ANL CMPCR1,#NOT 08H ; Internal reference voltage is CMP-input pin
; ORL CMPCR1,#08H ; ADC input pin teaches CMP+ input
; ANL CMPCR1,#NOT 04H ; P3.6 is the CMP-input pin
; ORL CMPCR1,#04H ; P3.6 is the CMP-input pin
; ANL CMPCR1,#NOT 02H ; Disable comparator output
; ORL CMPCR1,#02H ; Enable comparator output
; ORL CMPCR1,#80H ; Enable comparator module

JMP $ END

C code
#include "reg51.h"
#include "intrins.h"
sfr CMPCR1 = 0xe6;
sfr CMPCR2 = 0xe7;
sbit P10 = P1^0;
sbit P11 = P1^1;
void CMP_Isr() interrupt 21
{  
    CMPCR1 &= ~0x40; //clear the symbol of interrupt
    if (CMPCR1 & 0x01)
    {  
        P10 = !P10;
    }
    else
    {  
        P11 = !P11;
    }
}

void main()
{
    CMPCR2 = 0x00;  // Comparator result positive output
    CMPCR2 &= ~0x80;  // Comparator result negative output.
    CMPCR2 &= ~0x40;  // forbid 0.1us filtering
    CMPCR2 &= ~0x04;  // enable 0.1us filtering
    CMPCR2 |= 0x10;  // direct output the result
    CMPCR2 |= 0x00;  
    CMPCR1 &= ~0x08;  // P3.7 Input for CMP+
}
// CMPCR1 |= 0x08; //ADC input pin teaches CMP+ input
// CMPCR1 &= ~0x04; //Internal reference voltage is CMP-input pin
CMPCR1 |= 0x04; //P3.6 is the CMP-input pin
// CMPCR1 &= ~0x02; //Disable comparator output
CMPCR1 |= 0x02; //Enable comparator output
CMPCR1 |= 0x80; //Enable comparator module

EA = 1;

while (1);
}

15.3.2 Use of comparators(search way)

Assembly code

| CMPCR1 | DATA | 0E6H |
| CMPCR2 | DATA | 0E7H |

ORG 0000H
LJMP MAIN

ORG 0100H

MAIN:

MOV SP,#3FH
MOV CMPCR2,#00H
ANL CMPCR2,#NOT 80H ;Comparator result positive output
; ORL CMPCR2,#80H ;Comparator result negative output.
; ANL CMPCR2,#NOT 40H ;forbid 0.1us filtering
; ORL CMPCR2,#40H ;enable 0.1us filtering
; ANL CMPCR2,#NOT 3FH ;direct output the result
ORL CMPCR2,#10H
ORL CMPCR2,#30H ;Enable comparator edge interrupt
; ANL CMPCR1,#NOT 20H ;Disable comparator rising edge interrupt
; ORL CMPCR1,#20H ;Enable comparator rising edge interrupt
; ANL CMPCR1,#NOT 10H ;Disable comparator falling edge interrupt
; ORL CMPCR1,#10H ;Enable comparator falling edge interrupt
ANL CMPCR1,#NOT 08H ;P3.7 Input for CMP+
; ORL CMPCR1,#08H ;ADC input pin teaches CMP+ input
; ANL CMPCR1,#NOT 04H ;Internal reference voltage is CMP-input pin
ORL CMPCR1,#04H ;P3.6 is the CMP-input pin
; ANL CMPCR1,#NOT 02H ;Disable comparator output
ORL CMPCR1,#02H ;Enable comparator output
ORL CMPCR1,#80H ;Enable comparator module

LOOP:

MOV A,CMPCR1
MOV C,ACC.0
MOV P1.0,C ;read the result of comparator
JMP LOOP

END

C code

#include "reg51.h"

Nantong guoxin Microelectronics Co., Ltd. Tel: 0513-5501 2928/2929/2966 Fax: 0513-5501 2926/2956/2947
```c
#include "intrins.h"

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

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

void main()
{
    CMPCR2 = 0x00;
    CMPCR2 &= ~0x80; //Comparator result positive output
    // CMPCR2 |= 0x80; //Comparator result negative output.
    CMPCR2 &= ~0x40; //forbid 0.1us filtering
    // CMPCR2 |= 0x40; //enable 0.1us filtering
    // CMPCR2 &= ~0x3f; //direct output the result
    CMPCR2 |= 0x10;
    CMPCR1 = 0x00;
    CMPCR1 |= 0x30; //Enable comparator edge interrupt
    // CMPCR1 &= ~0x20; //Disable comparator rising edge interrupt
    // CMPCR1 |= 0x20; //Enable comparator rising edge interrupt
    // CMPCR1 &= ~0x10; //Disable comparator falling edge interrupt
    // CMPCR1 |= 0x10; //Enable comparator falling edge interrupt
    CMPCR1 &= ~0x08; //P3.7 Input for CMP+
    // CMPCR1 |= 0x08; //ADC input pin teaches CMP+ input
    // CMPCR1 &= ~0x04; //Internal reference voltage is CMP-input pin
    CMPCR1 |= 0x04; //P3.6 is the CMP-input pin
    // CMPCR1 &= ~0x02; //Disable comparator output
    CMPCR1 |= 0x02; //Enable comparator output
    CMPCR1 |= 0x80; //Enable comparator module

    while (1)
    {
        P10 = CMPCR1 & 0x01; //read the result of comparator
    }
}
```
15.3.3 Configure Comparator as External Brown-out Detection

The resistors R1 and R2 in the figure above divide the voltage at the front end of the regulator block 7805. The divided voltage is compared with the external reference of the comparator CMP+ and the internal reference voltage (about 1.344V).

Generally, when the AC voltage is 220V, the DC voltage at the front end of the voltage regulator block 7805 is 11V, but when the AC voltage drops to 160V, the voltage at the front end of the voltage regulator block 7805 is 8.5V. When the direct voltage at the front end of the voltage regulator block 7805 is lower than or equal to 8.5V, the directly held voltage at the front end input is divided by the resistors R1 and R2 to the positive input terminal CMP+ of the comparator, and the input voltage at the CMP+ terminal is lower than the internal reference voltage. A comparator interrupt can be generated at this time so that there is sufficient time to save data to the EEPROM during brownout detection. When the direct voltage at the front end of the voltage regulator block 7805 is higher than 8.5V, the DC voltage input by the front end is divided by the resistors R1 and R2 to the positive input terminal CMP+ of the comparator, and the input voltage of the CMP+ terminal is higher than the internal reference voltage. The CPU can continue to work normally.

The internal reference voltage is the voltage REFV of the internal BandGap which passed the module OP. The voltage of the REFV is about 1.344V. Due to the manufacturing error, the actual voltage may be between 1.34V and 1.35V. The specific value is obtained by reading the value of the address occupied by the internal reference voltage in the internal RAM area or the ROM area. For the STC8 family, the internal reference voltages are stored in RAM and ROM as shown in the following table.

<table>
<thead>
<tr>
<th>Module of microcontroller</th>
<th>Address which store in RAM (High byte first)</th>
<th>Address which store in ROM (High byte first)</th>
</tr>
</thead>
<tbody>
<tr>
<td>STC8A8K16S4A12/STC8A4K16S4A12</td>
<td>0EFH-0F0H</td>
<td>3FF7H-3FF8H</td>
</tr>
</tbody>
</table>
Note: If you need to read the reference voltage from the ROM, you need to check the following options during ISP download.
15.3.4 Comparator detects operating voltage (battery voltage)

In the figure above, the operating voltage of the MCU can be measured approximately by the principle of resistive voltage division (the gate of the strobe, the IO output of the MCU is low, the voltage of the port is close to GND, the channel that has not been selected, and the IO port of the MCU Output open-drain mode high, does not affect other channels).

The negative terminal of the comparator selects the internal reference voltage (approximately 1.344V), and the positive terminal selects the voltage value that is input to the CMP+ pin after passing through the resistor divider.

During the initialization, P2.5~P2.0 are all set to open-drain mode and output high. Firstly, P2.0 output low level, with the same time, if the VCC voltage is lower than 2.5V, the comparator's comparison value is 0, whereas, the comparator's comparison value is 1; If it is determined that VCC is higher than 2.5V, the P2.0 output will be high and the P2.1 output will be low. At this time, if the VCC voltage is lower than 3.0V, the comparison value of the comparator is 0, whereas if the VCC voltage is higher than 3.0V compares the comparator to 1; If it is determined that VCC is higher than 3V, the P2.1 output will be high and the P2.2 output will be low. At this time, if the VCC voltage is lower than 3.5V, the comparison value of the comparator is 0, whereas if the VCC voltage is higher than 3.5V compares the comparator to 1; If it is determined that VCC is higher than 3.5V, the P2.2 output will be high and the P2.3 output will be low. At this time, if the VCC voltage is lower than 4.0V, the comparison value of the comparator is 0, whereas if the VCC voltage is higher than 4.0V compares the comparator to 1; If it is determined that VCC is higher than 4V, the P2.3 output will be high and the P2.4 output will be low. At this time, if the VCC voltage is lower than 4.5 V, the comparison value of the comparator is 0, whereas if the VCC voltage is higher than 4.5 V compares the comparator to 1; If it is determined that VCC is higher than 4.5 V, the P2.4 output will be high and the P2.5 output will be low. At this time, if the VCC voltage is lower than 5.0V, the comparison value of the comparator is 0, whereas if the VCC voltage is higher than 5.0V compares the comparator to 1;
### Assembly code

<table>
<thead>
<tr>
<th>Register</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMPCR1</td>
<td>DATA</td>
<td>0E6H</td>
</tr>
<tr>
<td>CMPCR2</td>
<td>DATA</td>
<td>0E7H</td>
</tr>
<tr>
<td>P2M0</td>
<td>DATA</td>
<td>96H</td>
</tr>
<tr>
<td>P2M1</td>
<td>DATA</td>
<td>95H</td>
</tr>
</tbody>
</table>

**ORG 0000H**
LJMP MAIN

**ORG 0100H**

```assembly
MAIN:
    MOV SP,#3FH
    MOV P2M0,#00111111B ;P2.5~P2.0 initialized to open-drain mode
    MOV P2M1,#00111111B
    MOV P2,#0FFH
    MOV CMPCR2,#10H ;Comparator output result after 16 debounced clocks
    MOV CMPCR1,#00H
    ANL CMPCR1,#NOT 08H ;P3.7 is the CMP+ input pin
    ANL CMPCR1,#NOT 04H  ;Internal reference voltage is CMP-input pin
    ANL CMPCR1,#NOT 02H  ;Disable comparator output
    ORL CMPCR1,#80H ;Enable comparator module

LOOP:
    MOV R0,#00000000B ;voltage<2.5V
    MOV P2,#11111110B ;P2.0 output 0
    CALL DELAY
    MOV A,CMPCR1
    JNB ACC.0,SKIP
    MOV R0,#00000001B ;voltage>2.5V
    MOV P2,#11111101B ;P2.1 output 0
    CALL DELAY
    MOV A,CMPCR1
    JNB ACC.0,SKIP
    MOV R0,#00000011B ;voltage>3.0V
    MOV P2,#11111011B ;P2.2 output 0
    CALL DELAY
    MOV A,CMPCR1
    JNB ACC.0,SKIP
    MOV R0,#00000111B ;voltage>3.5V
    MOV P2,#11110111B ;P2.3 output 0
    CALL DELAY
    MOV A,CMPCR1
    JNB ACC.0,SKIP
    MOV R0,#00001111B ;voltage>4.0V
    MOV P2,#11101111B ;P2.4 output 0
    CALL DELAY
    MOV A,CMPCR1
    JNB ACC.0,SKIP
    MOV R0,#00011111B ;voltage>4.5V
    MOV P2,#11011111B ;P2.5 output 0
    CALL DELAY
    MOV A,CMPCR1
    JNB ACC.0,SKIP
```

Nantong guoxin Microelectronics Co., Ltd.

Tel: 0513-5501 2928/2929/2966
Fax: 0513-5501 2926/2956/2947
MOV R0,#00111111B ; voltage > 5.0V

SKIP:
MOV P2,#11111111B
MOV A,R0
CPL A
MOV P0,A ; P0.5~P0.0 port display voltage
JMP LOOP

DELAY:
MOV R0,#20
DJNZ R0,$
RET

END

C code
#include "reg51.h"
#include "intrins.h"
sfr CMPCR1 = 0xe6;
sfr CMPCR2 = 0xe7;
sfr P2M0 = 0x96;
sfr P2M1 = 0x95;
void delay()
{
    char i;
    for (i=0; i<20; i++);
}
void main()
{
    unsigned char v;
    P2M0 = 0x3f; // P2.5~P2.0 initialized to open-drain mode
    P2M1 = 0x3f;
    P2 = 0xff;
    CMPCR2 = 0x10; // Comparator output result after 16 debounced clocks
    CMPCR1 = 0x00;
    CMPCR1 &= ~0x08; // P3.7 is the CMP+ input pin
    CMPCR1 &= ~0x04; // Internal reference voltage is CMP-input pin
    CMPCR1 &= ~0x02; // Disable comparator output
    CMPCR1 |= 0x80; // Enable comparator module

    while (1)
    {
        v = 0x00; // voltage < 2.5V
        P2 = 0xfe; // P2.0 output 0
        delay();
        if (!((CMPCR1 & 0x01)) goto ShowVol;
        v = 0x01; // voltage > 2.5V
        P2 = 0xfd; // P2.1 output 0
        delay();
if (!(CMPCR1 & 0x01)) goto ShowVol;
v = 0x03; //voltage>3.0V
P2 = 0xfb; //P2.2 output 0
delay();
if (!(CMPCR1 & 0x01)) goto ShowVol;
v = 0x07; //voltage>3.5V
P2 = 0xf7; //P2.3 output 0
delay();
if (!(CMPCR1 & 0x01)) goto ShowVol;
v = 0x0f; //voltage>4.0V
P2 = 0xef; //P2.4 output 0
delay();
if (!(CMPCR1 & 0x01)) goto ShowVol;
v = 0x1f; //voltage>4.5V
P2 = 0xdf; //P2.5 output 0
delay();
if (!(CMPCR1 & 0x01)) goto ShowVol;
v = 0x3f; //voltage>5.0V
ShowVol:
P2 = 0xff;
P0 = ~v;
}

}
16 IAP/EEPROM

STC8F family microcontrollers have integrated a large capacity of internal EEPROM. The internal Data Flash can be used as EEPROM by using ISP / IAP technology. And it can be repeatedly erased more than 100,000 times. EEPROM can be divided into several sectors, each sector contains 512 bytes. When EEPROM is used, it is recommended that the data modified at the same time be stored in the same sector, and data modified not at the same time be stored in different sectors, and not necessarily full. Data memory is erased sector by sector.

EEPROM can be used to save some parameters which need to be modified in the application process and need be kept when power down takes place. In the user program, byte read / byte programming / sector erase can be performed to the EEPROM. When the operating voltage is low, it is recommended not to carry out EEPROM operation to avoid data loss situation.

16.1 EEPROM Related Registers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Address</th>
<th>Bit Address and Symbol</th>
<th>Value after reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAP_DATA</td>
<td>ISP/IAP Flash Data Register</td>
<td>C2H</td>
<td>B7 B6 B5 B4 B3 B2 B1 B0</td>
<td>1111,1111</td>
</tr>
<tr>
<td>IAP_ADDRH</td>
<td>ISP/IAP Flash Address High</td>
<td>C3H</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>IAP_ADDRL</td>
<td>ISP/IAP Flash Address Low</td>
<td>C4H</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>IAP_CMD</td>
<td>ISP/IAP Flash Command Register</td>
<td>C5H</td>
<td>CMD[1:0]</td>
<td>xxxxx00</td>
</tr>
<tr>
<td>IAP_TRIG</td>
<td>ISP/IAP Flash Trigger register</td>
<td>C6H</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>IAP_CONTR</td>
<td>ISP/IAP Control Register</td>
<td>C7H</td>
<td>IAPEN</td>
<td>SWBS</td>
</tr>
</tbody>
</table>

EEPROM data register (IAP_DATA)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAP_DATA</td>
<td>C2H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

During EEPROM read operation, the EEPROM data be read after the command execution is completed is stored in the IAP_DATA register. When writing the EEPROM, the data to be written must be stored in the IAP_DATA register before the write command is sent. The erase EEPROM command is not related to the IAP_DATA register.

EEPROM address registers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAP_ADDRH</td>
<td>C3H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IAP_ADDRL</td>
<td>C4H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The target address register of EEPROM read, write, erase operation. IAP_ADDRH is the high byte address, and IAP_ADDRL is the low byte of the address.
EEPROM command register (IAP_CMD)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAP_CMD</td>
<td>C5H</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CMD[1:0]</td>
</tr>
</tbody>
</table>

CMD[1:0]: ISP/IAP operating mode selection.
- 00: No operation.
- 01: EEPROM read command. Read one byte from the destination address.
- 10: EEPROM write command. Write one byte from the destination address.
- 11: EEPROM erase command. Write one sector from the destination address.

EEPROM command trigger register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAP_TRIG</td>
<td>C6H</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

After setting the command register, address register, data register and control register of EEPROM read, write and erase operation, 5AH and A5H are written to the trigger register IAP_TRIG sequentially to trigger the corresponding read, write and erase operation. The order of 5AH and A5H can not be changed. After the operation is completed, the contents of the EEPROM address registers IAP_ADDRH, IAP_ADDRL and the EEPROM command register IAP_CMD do not change. The value of the IAP_ADDRH and IAP_ADDRL registers must be updated manually if the datum of the next address needs to be operated.

Note: For every EEPROM operation, we should write 5AH to IAP_TRIG first and then A5H to take effect the corresponding command. After the trigger command has been written, the CPU should wait in IDLE state until the corresponding IAP operation completes. The CPU will return to the normal state from the IDLE state and resume executing the CPU instructions.

EEPROM control register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAP_CONTR</td>
<td>C7H</td>
<td>IAPEN</td>
<td>SWBS</td>
<td>SWRST</td>
<td>CMD_FAIL</td>
<td>-</td>
<td>IAP_WT[2:0]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

IAPEN: ISP/IAP operation enable bit.
- 0: disable all ISP/IAP program/erase/read function.
- 1: Enable ISP/IAP program/erase/read function.

SWBS: Software boot selection control bit, which should be used with SWRST.
- 0: Execute the program from the user code area after the software reset.
- 1: Execute the program from the ISP memory area after the software reset.

SWRST: Software reset trigger control.
- 0: No operation.
- 1: Generate software reset.

CMD_FAIL: Command fail status bit for EEPROM operation which should be cleared by software.
- 0: EEPROM operation is right.
- 1: EEPROM operation fails.

IAP_WT[2:0]: Waiting time selection of EEPROM operation

<table>
<thead>
<tr>
<th>IAP_WT[2:0]</th>
<th>Read one byte (2 clocks)</th>
<th>Write one byte (about 55us)</th>
<th>Erase one sector (about 21ms)</th>
<th>Clock frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1</td>
<td>2 clocks</td>
<td>55 clocks</td>
<td>21012 clocks</td>
<td>≥ 1MHz</td>
</tr>
<tr>
<td>1 1 0</td>
<td>2 clocks</td>
<td>110 clocks</td>
<td>42024 clocks</td>
<td>≥ 2MHz</td>
</tr>
</tbody>
</table>
16.2 Important Notes on EEPROM Programming and Erase Waiting Time

Table 1 (Operation Time Requirements for STC8A Series and STC8F Series EEPROM)

<table>
<thead>
<tr>
<th>EEPROM operation</th>
<th>Shortest time</th>
<th>Longest time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program</td>
<td>6us</td>
<td>7.5us</td>
</tr>
<tr>
<td>Erase</td>
<td>4ms</td>
<td>6ms</td>
</tr>
</tbody>
</table>

Table 2 (Time Waiting Period for Waiting Parameters for STC8A Series and STC8F Series EEPROM Operations)

<table>
<thead>
<tr>
<th>IAP_WT[2:0]</th>
<th>Programming waiting clock</th>
<th>Erasing waiting clocks</th>
<th>Suitable frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1</td>
<td>7 clocks</td>
<td>5000 clocks</td>
<td>1MHz</td>
</tr>
<tr>
<td>1 1 0</td>
<td>14 clocks</td>
<td>10000 clocks</td>
<td>2MHz</td>
</tr>
<tr>
<td>1 0 1</td>
<td>21 clocks</td>
<td>15000 clocks</td>
<td>3MHz</td>
</tr>
<tr>
<td>1 0 0</td>
<td>42 clocks</td>
<td>30000 clocks</td>
<td>6MHz</td>
</tr>
<tr>
<td>0 1 1</td>
<td>84 clocks</td>
<td>60000 clocks</td>
<td>12MHz</td>
</tr>
<tr>
<td>0 1 0</td>
<td>140 clocks</td>
<td>100000 clocks</td>
<td>20MHz</td>
</tr>
<tr>
<td>0 0 1</td>
<td>168 clocks</td>
<td>120000 clocks</td>
<td>24MHz</td>
</tr>
<tr>
<td>0 0 0</td>
<td>301 clocks</td>
<td>215000 clocks</td>
<td>30MHz</td>
</tr>
</tbody>
</table>

The programming and erase wait times of the internal EEPROM of the STC8A series and STC8F series MCU must meet the requirements in Table I. The waiting time should not be either too short or too long.

The waiting time of the program must be between 6 us and 7.5 us. If the programming wait time is too short (less than the minimum time of 6 us), the data inside the programmed target memory unit may not be reliable (the data may not be stored for 25 years). If the waiting time is too long (more than 1.5 times the maximum time of 7.5 us), the data written may also be incorrect due to data interference. If we ensure the requests of waiting time for programming and finished the output data comparison after completion of the programming and get the correct checkout, the data will be programmed correctly.

The erase wait time must be between 4ms and 6ms, and the erase wait time is too small (less than the shortest time 4ms), then the erased target memory sector may not be erased cleanly; if the wait time is too long (greater than the maximum A long time of 1.5 times 6ms more than 9ms) will shorten the life of the EEPROM, that is, the original erase life of 100,000 times may be shortened to 50,000 times.

The programming and erasing waiting time should be properly selected according to the recommended...
frequency given in Table 2. If the working frequency is 12MHz, please set the waiting parameter to 011B according to Table 2. If the actual operating frequency of the CPU is not in Table 2, the list of recommended frequencies needs to be calculated based on the actual frequency and the actual number of waiting clocks in Table 2 to find out the waiting time parameters that satisfy the time requirements of Table 1.

For example: the operating frequency is 4MHz, if you choose to wait for the parameter is 101B, the programming time is 21/4MHz = 5.25us, the erase time is 15000/4MHz = 3.75ms, the time is obviously not enough, so you should choose to wait for the parameter is 100B, then programming The time is 42/4MHz = 10.5us, the erase time is 30000/4MHz = 7.5ms, and the time is between the shortest time and 1.5 times the longest time.

Note: The clock that the EEPROM waits for operation refers to the system clock after the main clock is divided, which means the actual working clock of the CPU. If the microcontroller uses an internal high-precision IRC, the EEPROM is waiting for the operating clock to use the ISP download software download frequency after adjustment; if the microcontroller uses an external crystal, the EEPROM waits for the operation of the external crystal frequency through the CLKDIV register points After the clock (for example, if the microcontroller uses an external crystal and the frequency of the external crystal is 24MHz and the value of the CLKDIV register is set to 4, the clock frequency of the EEPROM waiting for the operation is 24MHz/4 = 6MHz. At this time, the waiting parameter should be 100B, but can't choose 001B).

### 16.3 Demo code

#### 16.3.1 Basic operation for EEPROM

**assembly code**

```assembly
;Test operating frequency is 11.0592 MHz:

IAP_DATA   DATA   0C2H
IAP_ADDRH  DATA   0C3H
IAP_ADDRL  DATA   0C4H
IAP_CMD    DATA   0C5H
IAP_TRIG   DATA   0C6H
IAP_CONTR  DATA   0C7H

WT_30M     EQU     80H
WT_24M     EQU     81H
WT_20M     EQU     82H
WT_12M     EQU     83H
WT_6M      EQU     84H
WT_3M      EQU     85H
WT_2M      EQU     86H
WT_1M      EQU     87H

ORG        0000H
LJMP       MAIN

ORG        0100H

IAP_IDLE:

MOV        IAP_CONTR,#0  ;Turn off IAP function
MOV        IAP_CMD,#0   ;Clear command register
MOV        IAP_TRIG,#0  ;Clear trigger register
```

Nantong guoxin Microelectronics Co., Ltd.  Tel: 0513-5501 2928/2929/2966  Fax: 0513-5501 2926/2956/2947 - 326 -
MOV IAP_ADDRH,#80H ;Set address to area where not belong to IAP
MOV IAP_ADDRL,#0
RET

IAP_READ:
MOV IAP_CONTR,#WT_12M ;enable IAP
MOV IAP_CMD,#1 ;Set IAP read command
MOV IAP_ADDRL,DPL ;Set IAP low address
MOV IAP_ADDRH,DPH ;Set IAP high address
MOV IAP_TRIG,#5AH ;Write trigger command(0x5a)
NOP
MOV A,IAP_DATA ;Read IAP data
LCALL IAP_IDLE ;Turn off IAP function
RET

IAP_PROGRAM:
MOV IAP_CONTR,#WT_12M ;enable IAP
MOV IAP_CMD,#2 ;Set IAP read command
MOV IAP_ADDRL,DPL ;Set IAP low address
MOV IAP_ADDRH,DPH ;Set IAP high address
MOV IAP_DATA,A ;Write IAP data
MOV IAP_TRIG,#5AH ;Write trigger command(0x5a)
NOP
LCALL IAP_IDLE ;Turn off IAP function
RET

IAP_ERASE:
MOV IAP_CONTR,#WT_12M ;enable IAP
MOV IAP_CMD,#3 ;Set IAP wipe command
MOV IAP_ADDRL,DPL ;Set IAP low address
MOV IAP_ADDRH,DPH ;Set IAP high address
MOV IAP_TRIG,#5AH ;Write trigger command(0x5a)
NOP
LCALL IAP_IDLE ;Turn off IAP function
RET

MAIN:
MOV SP,#3FH
MOV DPTR,#0400H
LCALL IAP_ERASE
MOV DPTR,#0400H
LCALL IAP_READ
MOV P0,A ;P0=0FFH
MOV DPTR,#0400H
MOV A,#12H
LCALL IAP_PROGRAM
MOV DPTR,#0400H
LCALL IAP_READ
MOV P1,A ;P1=12H
SJMP$

END
C code

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

sfr IAP_DATA = 0xC2;
sfr IAP_ADDRH = 0xC3;
sfr IAP_ADDRL = 0xC4;
sfr IAP_CMD = 0xC5;
sfr IAP_TRIG = 0xC6;
sfr IAP_CONTR = 0xC7;

#define WT_30M 0x80
#define WT_24M 0x81
#define WT_20M 0x82
#define WT_12M 0x83
#define WT_6M 0x84
#define WT_3M 0x85
#define WT_2M 0x86
#define WT_1M 0x87

void IapIdle()
{
    IAP_CONTR = 0; //Turn off IAP function
    IAP_CMD = 0; //Clear command register
    IAP_TRIG = 0; //Clear trigger register
    IAP_ADDRH = 0x80; //Set address to area where not belong to IAP
    IAP_ADDRL = 0;
}

text IapRead(int addr)
{
    char dat;
    IAP_CONTR = WT_12M; //enable IAP
    IAP_CMD = 1; //Set IAP read command
    IAP_ADDRL = addr; //Set IAP low address
    IAP_ADDRH = addr >> 8; //Set IAP high address
    IAP_TRIG = 0x5a; //Write trigger command(0x5a)
    IAP_TRIG = 0xa5; //Write trigger command(0xa5)
    _nop_();
    dat = IAP_DATA; //read IAP data
    IapIdle(); //Turn off IAP function
    return dat;
}

void IapProgram(int addr, char dat)
{
    IAP_CONTR = WT_12M; //enable IAP
    IAP_CMD = 2; //Set IAP wipe command
    IAP_ADDRL = addr; //Set IAP low address
    IAP_ADDRH = addr >> 8; //Set IAP high address
    IAP_DATA = dat; //Write IAP data
    IAP_TRIG = 0x5a; //Write trigger command(0x5a)
    IAP_TRIG = 0xa5; //Write trigger command(0xa5)
    _nop_();
}
```

Nantong guoxin Microelectronics Co., Ltd. Tel: 0513-5501 2928/2929/2966 Fax: 0513-5501 2926/2956/2947 - 328 -
IapIdle(); //Turn off IAP function

void IapErase(int addr)
{
    IAP_CONTR = WT_12M; //enable IAP
    IAP_CMD = 3; //Set IAP wipe command
    IAP_ADDRL = addr; //Set IAP low address
    IAP_ADDRH = addr >> 8; //Set IAP high address
    IAP_TRIG = 0x5a; //Write trigger command(0x5a)
    IAP_TRIG = 0xa5; //Write trigger command(0xa5)
    _nop_(); //
    IapIdle(); //Turn off IAP function
}

void main()
{
    IapErase(0x0400); //P0=0xff
    P0 = IapRead(0x0400); //P1=0x12
    IapProgram(0x0400, 0x12); //P1=0x12
    while (1);
}

16.3.2 Using the Serial Port to Send EEPROM Data

assembly code

<table>
<thead>
<tr>
<th>AUXR</th>
<th>DATA</th>
<th>8EH</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2H</td>
<td>DATA</td>
<td>0D6H</td>
</tr>
<tr>
<td>T2L</td>
<td>DATA</td>
<td>0D7H</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IAP_DATA</th>
<th>DATA</th>
<th>0C2H</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAP_ADDRH</td>
<td>DATA</td>
<td>0C3H</td>
</tr>
<tr>
<td>IAP_ADDRL</td>
<td>DATA</td>
<td>0C4H</td>
</tr>
<tr>
<td>IAP_CMD</td>
<td>DATA</td>
<td>0C5H</td>
</tr>
<tr>
<td>IAP_TRIG</td>
<td>DATA</td>
<td>0C6H</td>
</tr>
<tr>
<td>IAP_CONTR</td>
<td>DATA</td>
<td>0C7H</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WT_30M</th>
<th>EQU</th>
<th>80H</th>
</tr>
</thead>
<tbody>
<tr>
<td>WT_24M</td>
<td>EQU</td>
<td>81H</td>
</tr>
<tr>
<td>WT_20M</td>
<td>EQU</td>
<td>82H</td>
</tr>
<tr>
<td>WT_12M</td>
<td>EQU</td>
<td>83H</td>
</tr>
<tr>
<td>WT_6M</td>
<td>EQU</td>
<td>84H</td>
</tr>
<tr>
<td>WT_3M</td>
<td>EQU</td>
<td>85H</td>
</tr>
<tr>
<td>WT_2M</td>
<td>EQU</td>
<td>86H</td>
</tr>
<tr>
<td>WT_1M</td>
<td>EQU</td>
<td>87H</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ORG</th>
<th>0000H</th>
</tr>
</thead>
<tbody>
<tr>
<td>LJMP</td>
<td>MAIN</td>
</tr>
</tbody>
</table>

| ORG      | 0100H|

| UART_INIT: |
| MOV SCON,#5AH |
| MOV T2L,#0E8H | ;65536-11059200/115200/d=0ffe8h |
MOV T2H,#0FFH
MOV AUXR,#15H
RET

UART_SEND:
JNB TI,$
CLR TI
MOV SBUF,A
RET

IAP_IDLE:
MOV IAP_CONTR,#0 ;Turn off IAP function
MOV IAP_CMD,#0 ;Clear command register
MOV IAP_TRIG,#0 ;Clear trigger register
MOV IAP_ADDRH,#80H ;Set address to area where not belong to IAP
MOV IAP_ADDRL,#0
RET

IAP_READ:
MOV IAP_CONTR,#WT_12M ;enable IAP
MOV IAP_CMD,#1 ;Set IAP read command
MOV IAP_ADDRL,DPL ;Set IAP low address
MOV IAP_ADDRH,DPH ;Set IAP high address
MOV IAP_TRIG,#5AH ;Write trigger command(0x5a)
MOV IAP_TRIG,#0A5H ;Write trigger command(0xa5)
NOP
MOV A,IAP_DATA ;read IAP data
LCALL IAP_IDLE ;Turn off IAP function
RET

IAP_PROGRAM:
MOV IAP_CONTR,#WT_12M ;enable IAP
MOV IAP_CMD,#2 ;Set IAP write command
MOV IAP_ADDRL,DPL ;Set IAP low address
MOV IAP_ADDRH,DPH ;Set IAP high address
MOV IAP_DATA,A ;Write IAP data
MOV IAP_TRIG,#5AH ;Write trigger command(0x5a)
MOV IAP_TRIG,#0A5H ;Write trigger command(0xa5)
NOP
LCALL IAP_IDLE ;Turn off IAP function
RET

IAP_ERASE:
MOV IAP_CONTR,#WT_12M ;enable IAP
MOV IAP_CMD,#3 ;Clear command register
MOV IAP_ADDRL,DPL ;Set IAP low address
MOV IAP_ADDRH,DPH ;Set IAP high address
MOV IAP_TRIG,#5AH ;Write trigger command(0x5a)
MOV IAP_TRIG,#0A5H ;Write trigger command(0xa5)
NOP
LCALL IAP_IDLE ;Turn off IAP function
RET

MAIN:
MOV SP,#3FH
LCALL UART_INIT
MOV DPTR,#0400H
LCALL IAP_ERASE  
MOV DPTR,#0400H  
LCALL IAP_READ  
LCALL UART_SEND  
MOV DPTR,#0400H  
MOV A,#12H  
LCALL IAP_PROGRAM  
MOV DPTR,#0400H  
LCALL IAP_READ  
LCALL UART_SEND  
SJMP $  
END

C code

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

#define FOSC 11059200UL
#define BRT (65536 - FOSC / 115200 / 4)
sfr AUXR = 0x8e;
sfr T2H = 0xd6;
sfr T2L = 0xd7;
sfr IAP_DA = 0xC2;
sfr IAP_ADDRH = 0xC3;
sfr IAP_ADDRL = 0xC4;
sfr IAP_CMD = 0xC5;
sfr IAP_TRIG = 0xC6;
sfr IAP_CONTR = 0xC7;

#define WT_30M 0x80
#define WT_24M 0x81
#define WT_20M 0x82
#define WT_12M 0x83
#define WT_6M 0x84
#define WT_3M 0x85
#define WT_2M 0x86
#define WT_1M 0x87

void UartInit()
{
    SCON = 0x5a;
    T2L = BRT;
    T2H = BRT >> 8;
    AUXR = 0x15;
}

void UARTsend(char dat)
{
    while (!TI);
    TI = 0;
    SBUF = dat;
}
void IapIdle()
{
    IAP_CONTR = 0;  // Turn off IAP function
    IAP_CMD = 0;  // Clear command register
    IAP_TRIG = 0;  // Clear trigger register
    IAP_ADDRH = 0x80;  // Set address to area where not belong to IAP
    IAP_ADDRL = 0;
}

char IapRead(int addr)
{
    char dat;
    IAP_CONTR = WT_12M;  // enable IAP
    IAP_CMD = 1;  // Set IAP read command
    IAP_ADDRL = addr;  // Set IAP low address
    IAP_ADDRH = addr >> 8;  // Set IAP high address
    IAP_TRIG = 0x5a;  // Write trigger command (0x5a)
    IAP_TRIG = 0xa5;  // Write trigger command (0xa5)
    _nop_();
    dat = IAP_DATA;  // read IAP data
    IapIdle();  // Turn off IAP function
    return dat;
}

void IapProgram(int addr, char dat)
{
    IAP_CONTR = WT_12M;  // enable IAP
    IAP_CMD = 2;  // Set IAP write command
    IAP_ADDRL = addr;  // Set IAP low address
    IAP_ADDRH = addr >> 8;  // Set IAP high address
    IAP_DATA = dat;
    IAP_TRIG = 0x5a;  // Write trigger command (0x5a)
    IAP_TRIG = 0xa5;  // Write trigger command (0xa5)
    _nop_();
    IapIdle();  // Turn off IAP function
}

void IapErase(int addr)
{
    IAP_CONTR = WT_12M;  // enable IAP
    IAP_CMD = 3;  // Clear command register
    IAP_ADDRL = addr;  // Set IAP low address
    IAP_ADDRH = addr >> 8;  // Set IAP high address
    IAP_TRIG = 0x5a;  // Write trigger command (0x5a)
    IAP_TRIG = 0xa5;  // Write trigger command (0xa5)
    _nop_();
    IapIdle();  // Turn off IAP function
}

void main()
{
    UartInit();
    IapErase(0x0400);
    UARTsend(IapRead(0x0400));
    IapProgram(0x0400, 0x12);
    UARTsend(IapRead(0x0400));
    while (1);
}
17 Analog to Digital Converter (ADC)

STC8F family of microcontrollers integrated a 15-channel 12-bit high-speed Analog to Digital Converter. (the 16th channel can only be used to detect the internal REFV reference voltage, the voltage value of which is 1.344V, due to manufacturing error, the actual voltage value may be between 1.34V~1.35V). The system frequency is divided by 2 and then divided again by the user-set division ratio as the clock frequency of the ADC. The range of ADC clock frequency is SYSclock/2/1 ~ SYSclock/2/16. An A/D conversion can complete every fixed 16 ADC clocks. The speed of ADC can be up to 800Ks (that is, 800000 analog-to-digital conversions per second).

There are two data formats for ADC conversion results: Align left and Align right. It is convenient for user program to read and reference.

17.1 ADC Related Registers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Address</th>
<th>Bit Address and Symbol</th>
<th>Value after reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC_CONTR</td>
<td>ADC control register</td>
<td>BCH</td>
<td>B7 B6 B5 B4 B3 B2 B1 B0</td>
<td>000x,0000</td>
</tr>
<tr>
<td>ADC_RES</td>
<td>ADC result high register</td>
<td>BDH</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>ADC_RESL</td>
<td>ADC result low register</td>
<td>BEH</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>ADCCFG</td>
<td>ADC configuration register</td>
<td>DEH</td>
<td></td>
<td>xx0x,0000</td>
</tr>
</tbody>
</table>

ADC control register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC_CONTR</td>
<td>BCH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ADC_POWER: ADC power control bit
0: Turn off the power of ADC
1: Turn on the power of ADC.

It is recommended to turn the ADC off before entering Idle mode and Power-down mode to reduce the power consumption.

ADC_START: ADC start bit. ADC conversion will start after write 1 to this bit. It will automatically cleared by the hardware after A/D conversion completes.
0: No effect. Writing 0 to this bit will not stop the A/D conversion if the ADC has already started.
1: Start the A/D conversion. It will automatically cleared by the hardware after A/D conversion completes.

ADC_FLAG: ADC conversion completement flag. It will be set by the hardware after the ADC conversion has finished, and requests interrupt to CPU. It should be cleared by software.

ADC_CHS[3:0]: ADC analog channel selection bits.
## ADC configure register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADCCFG</td>
<td>DEH</td>
<td>-</td>
<td>-</td>
<td></td>
<td>RESFMT</td>
<td>-</td>
<td>SPEED[3:0]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RESFMT: ADC conversion result format control bit.

0: The conversion result aligns left. ADC_RES is used to save the upper 8 bits of the result and ADC_RESL is used to save the lower 4 bits of the result. The format is as follows:

- **ADC_RES**: Fill 0 automatically
- **ADC_RESL**: D11 D10 D9 D8 D7 D6 D5 D4

1: The conversion result aligns right. ADC_RES is used to save the upper 4 bits of the result and ADC_RESL is used to save the lower 8 bits of the result. The format is as follows:

- **ADC_RES**: D3 D2 D1 D0 0 0 0 0
- **ADC_RESL**: D7 D6 D5 D4 D3 D2 D1 D0

SPEED[3:0]: ADC clock control bits($F_{\text{ADC}} = \frac{\text{SYSclk}}{2/16/\text{SPEED}}$)

<table>
<thead>
<tr>
<th>SPEED[3:0]</th>
<th>ADC conversion time (number of CPUclocks)</th>
<th>SPEED[3:0]</th>
<th>ADC conversion time (number of CPUclocks)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>32</td>
<td>1000</td>
<td>288</td>
</tr>
<tr>
<td>0001</td>
<td>64</td>
<td>1001</td>
<td>320</td>
</tr>
<tr>
<td>0010</td>
<td>96</td>
<td>1010</td>
<td>352</td>
</tr>
<tr>
<td>0011</td>
<td>128</td>
<td>1011</td>
<td>384</td>
</tr>
<tr>
<td>0100</td>
<td>160</td>
<td>1100</td>
<td>416</td>
</tr>
<tr>
<td>0101</td>
<td>192</td>
<td>1101</td>
<td>448</td>
</tr>
<tr>
<td>0110</td>
<td>224</td>
<td>1110</td>
<td>480</td>
</tr>
</tbody>
</table>
ADC conversion result registers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC_RES</td>
<td>BDH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC_RESL</td>
<td>BEH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When the A/D conversion is completed, the 12-bit conversion result is automatically saved to ADC_RES and ADC_RESL. Please refer to the RESFMT setting in the ADC_CFG register to see the result's data format.

### 17.2 ADC Typical application circuit diagram

#### 17.2.1 High precision ADC application

![ADC circuit diagram](image-url)
17.2.2 ADC General Application (Applications with Low Accuracy ADC Requirements)

17.3 Sample program

17.3.1 ADC basic operation (Query Mode)

Assembly code

;The test operating frequency is 11.0592 MHz

| ADC_CONTR | DATA | 0BCH |
| ADC_RES  | DATA | 0BDH |
| ADC_RESL | DATA | 0BEH |
| ADCCFG   | DATA | 0DEH |
| PIM0     | DATA | 092H |
| PIM1     | DATA | 091H |
| ORG      |      | 0000H |
| LJMP     |      | MAIN |

System clock<=10MHz

| C?        | 104(0.1uF) |

System clock>10MHz

| C?        | 103(0.01uF) |
ORG 0100H

MAIN:
MOV SP,#3FH

MOV P1M0,#00H ;Set P1.0 as ADC port
MOV P1M1,#01H
MOV ADCCFG,#0FH ;Set the ADC clock as the system clock / 2 / 16 / 16 / 16
MOV ADC_CONTR,#80H ;Enable ADC module

LOOP:
ORL ADC_CONTR,#40H ;Start AD conversion
NOP
NOP
MOV A,ADC_CONTR ;Query ADC Completement Flag
JNB ACC.5,$-2
ANL ADC_CONTR,#NOT 20H ;Clear Completement Flag
MOV P2,ADC_RES ;Read ADC results
SJMP LOOP

END

C code

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

//The test operating frequency is 11.0592 MHz

sfr ADC_CONTR = 0xbc;
sfr ADC_RES = 0xbd;
sfr ADC_RESL = 0xbe;
sfr ADCCFG = 0xde;
sfr P1M0 = 0x92;
sfr P1M1 = 0x91;

void main()
{
P1M0 = 0x00; //Set P1.0 as ADC port
P1M1 = 0x01;
ADCCFG = 0xdf; //Set the ADC clock as the system clock / 2 / 16 / 16 / 16
ADC_CONTR = 0x80; //Enable ADC module

while (1)
{
    ADC_CONTR |= 0x40; //Start AD conversion
    _nop_();
    _nop_();
    while (!(ADC_CONTR & 0x20)); //Query ADC Completement Flag
    ADC_CONTR &= ~0x20; //Clear Completement Flag
    P2 = ADC_RES; //Read ADC results
}
}
17.3.2 ADC basic operation (Interrupt Mode)

Assembly code

; The test operating frequency is 11.0592 MHz

<table>
<thead>
<tr>
<th>ADC_contr</th>
<th>DATA</th>
<th>0BCH</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC_res</td>
<td>DATA</td>
<td>0BDH</td>
</tr>
<tr>
<td>ADC_resl</td>
<td>DATA</td>
<td>0BEH</td>
</tr>
<tr>
<td>ADCCFG</td>
<td>DATA</td>
<td>0DEH</td>
</tr>
<tr>
<td>EADC</td>
<td>BIT</td>
<td>IE.5</td>
</tr>
<tr>
<td>P1M0</td>
<td>DATA</td>
<td>092H</td>
</tr>
<tr>
<td>P1M1</td>
<td>DATA</td>
<td>091H</td>
</tr>
</tbody>
</table>

ORG 0000H
LJMP MAIN
ORG 002BH
LJMP ADCISR

ADCISR:
ANL ADC_contr,#NOT 20H ; Clear Complentation Flag
MOV P2,ADC_res ; Read ADC results
ORL ADC_contr,#40H ; Continue AD conversion
RETI

MAIN:
MOV SP,#3FH ; Set P1.0 as ADC port
MOV P1M0,#00H ; Set the ADC clock as the system clock / 2 / 16 / 16 / 16
MOV P1M1,#01H
MOV ADCCFG,#0FH ; Enable ADC module
SETB EADC ; Enable ADC Interrupt
SETB EA
ORL ADC_contr,#40H ; Start AD conversion
SJMP $
END

C code

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

// The test operating frequency is 11.0592 MHz

sfr ADC_contr = 0xbc;
sfr ADC_res = 0xbd;
sfr ADC_resl = 0xbe;
sfr ADCCFG = 0xde;
sbit EADC = IE^5;
sfr P1M0 = 0x92;
sfr P1M1 = 0x91;

void ADC_Isr() interrupt 5
{
    ADC_CONTR &= ~0x20; //Clear Interrupt Flag
    P2 = ADC_RES; //Read ADC results
    ADC_CONTR |= 0x40;  //Continue AD conversion
}

void main()
{
    P1M0 = 0x00; //Set P1.0 as ADC port
    P1M1 = 0x01;
    ADCCFG = 0x0f; //Set the ADC clock as the system clock / 2 / 16 / 16 / 16
    ADC_CONTR = 0x80; //Enable ADC module
    EA = 1; //Enable ADC Interrupt
    EA = 1;
    ADC_CONTR |= 0x40; //Start AD conversion
    while (1);
}

17.3.3 Format the ADC conversion result

Assembly code

;The test operating frequency is 11.0592 MHz

ADC_CONTR DATA 0BCH
ADC_RES DATA 0BDH
ADC_RESL DATA 0BEH
ADCCFG DATA 0DEH

ORG 0000H
LJMP MAIN

ORG 0100H

MAIN:
MOV SP,#3FH

MOV P1M0,#00H ;Set P1.0 as ADC port
MOV P1M1,#01H
MOV ADCCFG,#0FH ;Set the ADC clock as the system clock / 2 / 16 / 16 / 16
MOV ADC_CONTR,#80H ;Enable ADC module
ORL ADC_CONTR,#NOT 20H ;Clear Complete Flag
MOV ADCCFG,#00H ;Set the results Align left
MOV A, ADC_RES ; A stores the 12-bit results of the ADC at higher 8 bits
MOV B, ADC_RESL ; B[7:4] stores the 12-bit results of the ADC at lower 4 bits, B[3:0] is 0

; MOV ADCCFG, #20H ; Set the results Align right
; MOV A, ADC_RES ; A[3:0] stores the 12-bit results of the ADC at higher 4 bits, A[7:4] is 0
; MOV B, ADC_RESL ; B stores the 12-bit results of the ADC at lower 8 bits

SJMP $
END

C code
#include "reg51.h"
#include "intrins.h"

// The test operating frequency is 11.0592 MHz

sfr ADC_CONTR = 0xbc;
sfr ADC_RES = 0xbd;
sfr ADC_RESL = 0xbe;
sfr ADCCFG = 0xde;
sfr P1M0 = 0x92;
sfr P1M1 = 0x91;

void main()
{
P1M0 = 0x00; // Set P1.0 as ADC port
P1M1 = 0x01;
ADCCFG = 0x0f; // Set the ADC clock as the system clock / 2 / 16 / 16 / 16
ADC_CONTR |= 0x40; // Start AD conversion
_nop_();
_nop_();
while (!(ADC_CONTR & 0x20)); // Query ADC Completment Flag
ADC_CONTR &= ~0x20; // Clear Completment Flag

ADCCFG = 0x00; // Set the results Align left
ACC = ADC_RES; // A stores the 12-bit results of the ADC at higher 8 bits
B = ADC_RESL; // B[7:4] stores the 12-bit results of the ADC at lower 4 bits, B[3:0] is 0

// ADCCFG = 0x20; // Set the results Align right
// ACC = ADC_RES; // A[3:0] stores the 12-bit results of the ADC at higher 4 bits, A[7:4] is 0
// B = ADC_RESL; // B stores the 12-bit results of the ADC at lower 8 bits

while (1);
}
17.3.4 Using ADC Channel 16 to Measure External Voltage or Battery Voltage

The 16th channel of STC8 series ADC is used to test the internal reference voltage of BandGap, due to the internal BandGap reference voltage is very stable, about 1.35V, and does not change with the working voltage of the chip, so it can be measured by the internal reference voltage of BandGap, then the ADC value can deduce the external voltage or the external battery voltage.

The following figure is a reference circuit diagram:

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

//The test operating frequency is 11.0592 MHz
#define FOSC 11059200UL
#define BRT (65536 - FOSC / 115200 / 4)
sfr AUXR = 0x8e;
sfr ADC_CONTR = 0xbc;
sfr ADC_RES = 0xbd;
sfr ADC_RESL = 0xbe;
```
sfr ADCCFG = 0xde;

sfr P1M0 = 0x92;
sfr P1M1 = 0x91;

int *BGV; //Internal Bandgap voltage values are stored in idata
//Idata EFH address is used to store high byte
//Idata F0H address is used to store the low byte
//The voltage is in millivolts (mV)
bit busy;

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

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

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

void ADCInit()
{
  ADCCFG = 0x2f; //Set the ADC clock as the system clock / 2 / 16 / 16 / 16
  ADC_CONTR = 0x8f; //Enable ADC module, and select Channel 16.
}

int ADCRead()
{
  int res;
  ADC_CONTR |= 0x40; //Start AD conversion
  _nop_();
  _nop_();
  while (!(ADC_CONTR & 0x20)); //Query ADC Completement Flag
  ADC_CONTR &= ~0x20; //Clear Completement Flag
  res = (ADC_RES << 8) | ADC_RESL; //Read ADC results
}
return res;
}

void main()
{
    int res;
    int vcc;
    int i;
    BGV = (int idata *)0xfe;
    ADCInit();  //ADC initialization
    UartInit();  //UART initialization
    ES = 1;
    EA = 1;

    ADCRead();    //Discard the first two data
    ADCRead();
    res = 0;
    for (i=0; i<8; i++)
    {
        res += ADCRead();  //Read 8 times data
    }
    res >>= 3;  //get average value

    vcc = (int)(4095L * *BGV / res);  //Calculate the VREF pin voltage, that is, the battery voltage
    UARTsend(vcc >> 8);  //The voltage is in millivolts(mV)
    UARTsend(vcc);
    while (1);
}
18 Application of PCA/CCP/PWM application

The STC8F family of microcontrollers integrate four groups of programmable counter array (PCA/CCP/PWM) modules, which can be used for software timer, external pulse capture, high-speed pulse output and pulse width modulation (PWM) output.

PCA contains a special 16-bit counter, with which four groups of PCA modules are connected. The structure of PCA counter is as follows:

18.1 PCA Related register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Address</th>
<th>Bit Address and Symbol</th>
<th>Value after reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCON</td>
<td>PCA Control Register</td>
<td>D8H</td>
<td>B7 B6 B5 B4 B3 B2 B1 B0</td>
<td>CF CR - - CCF3 CCF2 CCF1 CCF0 00xx,0000</td>
</tr>
<tr>
<td>CMOD</td>
<td>PCA Mode Register</td>
<td>D9H</td>
<td>CIDL - - - CPS[2:0] ECF</td>
<td>0xxx,0000</td>
</tr>
<tr>
<td>CCAPM0</td>
<td>PCA 0 Mode Register</td>
<td>DAH</td>
<td>ECOM0 CCAPP0 CCAPN0 MAT0 TOG0 PWM0 ECCF0 s000,0000</td>
<td></td>
</tr>
<tr>
<td>CCAPM1</td>
<td>PCA 1 Mode Register</td>
<td>DBH</td>
<td>ECOM1 CCAPP1 CCAPN1 MAT1 TOG1 PWM1 ECCF1 s000,0000</td>
<td></td>
</tr>
<tr>
<td>CCAPM2</td>
<td>PCA 2 Mode Register</td>
<td>DCH</td>
<td>ECOM2 CCAPP2 CCAPN2 MAT2 TOG2 PWM2 ECCF2 s000,0000</td>
<td></td>
</tr>
<tr>
<td>CCAPM3</td>
<td>PCA 3 Mode Register</td>
<td>DDH</td>
<td>ECOM3 CCAPP3 CCAPN3 MAT3 TOG3 PWM3 ECCF3 s000,0000</td>
<td></td>
</tr>
<tr>
<td>CL</td>
<td>PCA Base Timer Low</td>
<td>E9H</td>
<td>0000,0000</td>
<td></td>
</tr>
<tr>
<td>CCAP0L</td>
<td>PCA 0 capture register low</td>
<td>EAH</td>
<td>0000,0000</td>
<td></td>
</tr>
<tr>
<td>CCAP1L</td>
<td>PCA 1 capture register low</td>
<td>EBH</td>
<td>0000,0000</td>
<td></td>
</tr>
<tr>
<td>CCAP2L</td>
<td>PCA 2 capture register low</td>
<td>ECH</td>
<td>0000,0000</td>
<td></td>
</tr>
<tr>
<td>CCAP3L</td>
<td>PCA 3 capture register low</td>
<td>EDH</td>
<td>0000,0000</td>
<td></td>
</tr>
<tr>
<td>PCA_PWM0</td>
<td>PCA0 PWM Mode Register</td>
<td>F2H</td>
<td>EBS0[1:0] XXCAP0H[1:0] XXCAP0L[1:0] EPC0H EPC0L 0000,0000</td>
<td></td>
</tr>
<tr>
<td>PCA_PWM1</td>
<td>PCA1 PWM Mode Register</td>
<td>F3H</td>
<td>EBS1[1:0] XXCAP1H[1:0] XXCAP1L[1:0] EPC1H EPC1L 0000,0000</td>
<td></td>
</tr>
<tr>
<td>PCA_PWM2</td>
<td>PCA2 PWM Mode Register</td>
<td>F4H</td>
<td>EBS2[1:0] XXCAP2H[1:0] XXCAP2L[1:0] EPC2H EPC2L 0000,0000</td>
<td></td>
</tr>
<tr>
<td>PCA_PWM3</td>
<td>PCA3 PWM Mode Register</td>
<td>F5H</td>
<td>EBS3[1:0] XXCAP3H[1:0] XXCAP3L[1:0] EPC3H EPC3L 0000,0000</td>
<td></td>
</tr>
</tbody>
</table>
CH PCA Base Timer High F9H 0000,0000
CCAP0H PCA 0 capture register high FAH 0000,0000
CCAP1H PCA 1 capture register high FBH 0000,0000
CCAP2H PCA 2 capture register high FCH 0000,0000
CCAP3H PCA 3 capture register high FDH 0000,0000

### PCA control register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCON</td>
<td>D8H</td>
<td>CF</td>
<td>CR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CCF3</td>
<td>CCF2</td>
</tr>
</tbody>
</table>

CF: PCA Counter overflow flag. It is set by hardware when the 16-bit counter of PCA overflows, and requests interrupt to CPU. It must be cleared by software.

CR: PCA counter enable bit.
- 0: Stop PCA counting.
- 1: Start PCA counting.

CCFn(n=0,1,2,3): PCA module interrupt flag. When a match or a capture occurs on the PCA module, the corresponding flag bit is set by the hardware automatically and requests an interrupt to CPU. These flags should be cleared by software.

### PCA mode register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMOD</td>
<td>D9H</td>
<td>CIDL</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>CPS[2:0]</td>
<td>ECF</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CIDL: PCA Counter control bit in Idle mode.
- 0: the PCA counter will continue counting in idle mode.
- 1: the PCA counter will stop counting in idle mode.

CPS[2:0]: PCA Counter pulse source select bits.

<table>
<thead>
<tr>
<th>CPS[2:0]</th>
<th>Input clock source of PCA</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>System clock/12</td>
</tr>
<tr>
<td>001</td>
<td>System clock/2</td>
</tr>
<tr>
<td>010</td>
<td>Overflow pulse of timer 0</td>
</tr>
<tr>
<td>011</td>
<td>External clock input from ECI pin</td>
</tr>
<tr>
<td>100</td>
<td>System clock</td>
</tr>
<tr>
<td>101</td>
<td>System clock/4</td>
</tr>
<tr>
<td>110</td>
<td>System clock/6</td>
</tr>
<tr>
<td>111</td>
<td>System clock/8</td>
</tr>
</tbody>
</table>

ECF: PCA counter overflow interrupt enable bit
- 0: disable PCA counter overflow interrupt
- 1: enable PCA counter overflow interrupt

### PCA counter registers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CL</td>
<td>E9H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The 16-bit counter is the combination of CL and CH, where CL is the low 8-bit counter and CH is the high 8-bit counter. The 16-bit counter of PCA increments automatically every one PCA clock.

**PCA Mode Control Register**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCAPM0</td>
<td>DAH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCAPM1</td>
<td>DBH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCAPM2</td>
<td>DCH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCAPM3</td>
<td>DDH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ECOMn: PCAn Comparator enable bit
CCAPPn: PCA n Capture on rising edge enable bit
CCAPNn: PCA n Capture on falling edge enable bit
MATn: PCAn match function enable bit
TOGn: PCA n high speed pulse output function enable bit
PWMn: PCAn PWM function enable bit
ECCFn: PCAn match/capture interrupt enable bit

**PCA capture value/compare value registers**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCAP0L</td>
<td>EAH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCAP1L</td>
<td>EBH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCAP2L</td>
<td>ECH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCAP3L</td>
<td>EDH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCAP0H</td>
<td>FAH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCAP1H</td>
<td>FBH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCAP2H</td>
<td>FCH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCAP3H</td>
<td>FDH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When the PCA capture function is enabled, CCAPnL and CCAPnH are used to save the count value (CL and CH) of the PCA at the time of capture. When the PCA comparison function is enabled, the PCA controller compares the current value in [CH,CL] and the value in [CCAPnH, CCAPnL], and the comparison result is given. When the PCA match function is enabled, the PCA controller compares the current value in [CH, CL] with the value stored in [CCAPnH, CCAPnL], and checks if they match (equal), then gives a match result.

**PCA PWM Mode Registers**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA_PWM0</td>
<td>F2H</td>
<td>EBS0[1:0]</td>
<td>XCCAP0H[1:0]</td>
<td>XCCAP0L[1:0]</td>
<td>EPC0H</td>
<td>EPC0L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCA_PWM1</td>
<td>F3H</td>
<td>EBS1[1:0]</td>
<td>XCCAP1H[1:0]</td>
<td>XCCAP1L[1:0]</td>
<td>EPC1H</td>
<td>EPC1L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCA_PWM2</td>
<td>F4H</td>
<td>EBS2[1:0]</td>
<td>XCCAP2H[1:0]</td>
<td>XCCAP2L[1:0]</td>
<td>EPC2H</td>
<td>EPC2L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCA_PWM3</td>
<td>F5H</td>
<td>EBS3[1:0]</td>
<td>XCCAP3H[1:0]</td>
<td>XCCAP3L[1:0]</td>
<td>EPC3H</td>
<td>EPC3L</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

EBSn[1:0]: PCAn PWM number of bits control

<table>
<thead>
<tr>
<th>EBSn[1:0]</th>
<th>PWM bits</th>
<th>Reload value</th>
<th>Comparison value</th>
</tr>
</thead>
</table>

XCCAPnH[1:0]: The 9th bit and 10th bit of reload value of 10-bit PWM
XCCAPnL[1:0]: The 9th bit and 10th bit of comparison value of 10-bit PWM
EPCnH: The MSB of reload value in PWM mode (i.e. the 9th bit of 8-bit PWM, the 8th bit of 7-bit PWM, the 7th bit of 6-bit PWM, the 11th bit of 10-bit PWM)
EPCnL: The MSB of comparison value in PWM mode (i.e. the 9th bit of 8-bit PWM, the 8th bit of 7-bit PWM, the 7th bit of 6-bit PWM, the 11th bit of 10-bit PWM)

Note: When updating the reload value of 10-bit PWM, write the upper two bits of XCCAPnH[1:0] firstly and then the lower 8 bits of CCAPnH[7:0].

### 18.2 PCA Operation Mode

There are 4 groups of PCA modules in STC8F family microcontrollers, and operation mode of each module can be set independently. The mode settings are as follows:

<table>
<thead>
<tr>
<th>CCAPMn</th>
<th>-</th>
<th>ECOMn</th>
<th>CAPn</th>
<th>CAPn</th>
<th>MATn</th>
<th>TOGn</th>
<th>PWMn</th>
<th>ECCFn</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>-0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td>1</td>
</tr>
<tr>
<td>-0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>-0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>x</td>
<td>1</td>
</tr>
</tbody>
</table>

Function of module

- 00 8-bit PWM \{EPCnH, CCAPnH[7:0]\} \{EPCnL, CCAPnL[7:0]\}
- 01 7-bit PWM \{EPCnH, CCAPnH[6:0]\} \{EPCnL, CCAPnL[6:0]\}
- 10 6-bit PWM \{EPCnH, CCAPnH[5:0]\} \{EPCnL, CCAPnL[5:0]\}
- 11 10-bit PWM \{EPCnH, XCCAPnH[1:0], CCAPnH[7:0]\} \{EPCnL, XCCAPnL[1:0], CCAPnL[7:0]\}

18.2.1 Capture Mode

At least one of CAPn and CAPn in CCAPMn must be set (or all them are set) for a PCA module to operate in capture mode. When a PCA module is operating in capture mode, the input hoppings on the external CCP0 / CCP1 / CCP2 / CCP3 pins of the module are sampled. When a valid hopping is sampled, the PCA controller immediately loads the counter values in the PCA counters, CH and CL, into the module's capture registers, CCAPnL and CCAPnH, and sets the corresponding CCFn in the CCON register. If the ECCFn bit in CCAPMn is set to 1, an interrupt will be generated. Since all PCA module's interrupt entry addresses are shared, it is necessary to determine which module generated an interrupt in the interrupt service routine and
note that the interrupt flag bit must be cleared by software.

The structure of the PCA module working in capture mode is shown in the following figure:

![PCA capture mode diagram]

**18.2.2 Software Timer Mode**

The PCA module can be used as a software timer by setting the ECOM and MAT bits in the CCAPMn register. The PCA counter values in CL and CH are compared with the capture registers values in CCAPnL and CCAPnH. When they are equal, CCFn in CCON is set and an interrupt is generated if ECCFn in CCAPMn is set to 1. CCFn flag bit should be cleared by software.

The structure of PCA module working in software timer mode is shown in the following figure:

![PCA software timer mode diagram]

**18.2.3 High Speed pulse Output Mode**

When the count value of the PCA counter matches the value of the capture register, the CCPn output of the PCA module will hop. To activate the high speed pulse output mode, the TOGn, MATn, and ECOMn bits of the CCAPMn register must be set.

The structure of PCA module working in high-speed pulse output mode is shown below:
18.2.4 Pulse Width Modulator Mode (PWM mode)

18.2.4.1 8-bit PWM Mode

Pulse width modulation is a technique that uses a program to control the duty ratio, cycle and phase of a waveform. It is widely used in applications such as three-phase motor drive and D/A conversion. The PCA modules of the STC8F family of microcontrollers can be configured to operate in 8-bit, 7-bit, 6-bit or 10-bit PWM mode by setting corresponding PCA_PWMn registers. To enable the PWM function of the PCA module, the PWMn and ECOMn bits of the module register CCAPMn must be set.

When EBSn [1:0] in the PCA_PWMn register is set to 00, the PCAn operates in 8-bit PWM mode, where \{0, CL [7:0]\} will be compared with the capture registers \{EPCnL, CCAPnL [7:0]\}. When PCA modules are operating in 8-bit PWM mode, the output frequencies of them are the same because all the modules share a single PCA counter. The output duty ratio of each module is set using the registers \{EPCnL, CCAPnL [7:0]\}. The output is low when the value of \{0, CL [7:0]\} is less than \{EPCnL, CCAPnL [7:0]\}, and the output is high when the value of \{0, CL [7:0]\} is equal to or greater than \{EPCnL, CCAPnL [7:0]\}. When CL [7:0] overflows from FF to 00, the contents of \{EPCnH, CCAPnH [7:0]\} are reloaded into \{EPCnL, CCAPnL [7:0]\}. This makes it possible to update the PWM without interference.

The structure of PCA module working in 8-bit PWM mode is shown below:
18.2.4.2 7-bit PWM Mode

When EBSn [1:0] in the PCA_PWMn register is set to 01, the PCAn operates in 7-bit PWM mode, where \{0, CL [6: 0]\} will be compared with the capture registers \{EPCnL, CCAPnL [6: 0]\}. When PCA modules are operating in 6-bit PWM mode, the output frequencies of them are the same because all the modules share a single PCA counter. The output duty ratio of each module is set using the registers \{EPCnL, CCAPnL [6: 0]\}. The output is low when the value of \{0, CL [6: 0]\} is less than \{EPCnL, CCAPnL [6: 0]\}, and the output is high when the value of \{0, CL [6: 0]\} is equal to or greater than \{EPCnL, CCAPnL [6: 0]\}. When CL [6: 0] overflows from 7F to 00, the contents of \{EPCnH, CCAPnH [6: 0]\} are reloaded into \{EPCnL, CCAPnL [6: 0]\}. This makes it possible to update the PWM without interference.

The structure of PCA module working in 6-bit PWM mode is shown below:

18.2.4.3 6-bit PWM Mode

When EBSn [1: 0] in the PCA_PWMn register is set to 10, the PCAn operates in 6-bit PWM mode, where \{0, CL [5: 0]\} will be compared with the capture registers \{EPCnL, CCAPnL [5: 0]\}. When PCA modules are operating in 6-bit PWM mode, the output frequencies of them are the same because all the modules share a
single PCA counter. The output duty ratio of each module is set using the registers \{EPCnL, CCAPnL [5: 0]\}. The output is low when the value of \{0, CL [5: 0]\} is less than \{EPCnL, CCAPnL [5: 0]\}, and the output is high when the value of \{0, CL [5: 0]\} is equal to or greater than \{EPCnL, CCAPnL [5: 0]\}. When CL [5: 0] overflows from 3F to 00, the contents of \{EPCnH, CCAPnH [5: 0]\} are reloaded into \{EPCnL, CCAPnL [5: 0]\}. This makes it possible to update the PWM without interference.

The structure of PCA module working in 6-bit PWM mode is shown below:

### 18.2.4.4 10-bit PWM Mode

When EBSn [1: 0] in the PCA_PWMn register is set to 11, the PCAn operates in 10-bit PWM mode, where \{CH[1:0],CL[7:0]\} will be compared with the capture registers \{EPCnL,XCCAPnL[1:0],CCAPnL[7:0]\}. When PCA modules are operating in 10-bit PWM mode, the output frequencies of them are the same because all the modules share a single PCA counter. The output duty ratio of each module is set using the registers \{EPCnL,XCCAPnL[1:0],CCAPnL[7:0]\}. The output is low when the value of \{CH[1:0],CL[7:0]\} is less than \{EPCnL,XCCAPnL[1:0],CCAPnL[7:0]\}, and the output is high when the value of \{CH[1:0],CL[7:0]\} is equal to or greater than \{EPCnL,XCCAPnL[1:0],CCAPnL[7:0]\}. When \{CH[1:0],CL[7:0]\} overflows from 3FF to 00, the contents of \{EPCnH,XCCAPnH[1:0],CCAPnH[7:0]\} are reloaded into \{EPCnL,XCCAPnL[1:0], CCAPnL[7:0]\}. This makes it possible to update the PWM without interference.

The structure of PCA module working in 10-bit PWM mode is shown below:
18.3 Sample program

18.3.1 PCA outputs PWM(6/7/8/10 bit)

Assembly code

;The test operating frequency is 11.0592 MHz

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Type</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCON</td>
<td>DATA</td>
<td>0D8H</td>
</tr>
<tr>
<td>CF</td>
<td>BIT</td>
<td>CCON.7</td>
</tr>
<tr>
<td>CR</td>
<td>BIT</td>
<td>CCON.6</td>
</tr>
<tr>
<td>CCF3</td>
<td>BIT</td>
<td>CCON.3</td>
</tr>
<tr>
<td>CCF2</td>
<td>BIT</td>
<td>CCON.2</td>
</tr>
<tr>
<td>CCF1</td>
<td>BIT</td>
<td>CCON.1</td>
</tr>
<tr>
<td>CCF0</td>
<td>BIT</td>
<td>CCON.0</td>
</tr>
<tr>
<td>CMOD</td>
<td>DATA</td>
<td>0D9H</td>
</tr>
<tr>
<td>CL</td>
<td>DATA</td>
<td>0E9H</td>
</tr>
<tr>
<td>CH</td>
<td>DATA</td>
<td>0F9H</td>
</tr>
<tr>
<td>CCAPM0</td>
<td>DATA</td>
<td>0DAH</td>
</tr>
<tr>
<td>CCAP0L</td>
<td>DATA</td>
<td>0EAH</td>
</tr>
<tr>
<td>CCAP0H</td>
<td>DATA</td>
<td>0FAH</td>
</tr>
<tr>
<td>PCA_PWM0</td>
<td>DATA</td>
<td>0F2H</td>
</tr>
<tr>
<td>CCAPM1</td>
<td>DATA</td>
<td>0DBH</td>
</tr>
<tr>
<td>CCAP1L</td>
<td>DATA</td>
<td>0EBH</td>
</tr>
<tr>
<td>CCAP1H</td>
<td>DATA</td>
<td>0FBH</td>
</tr>
<tr>
<td>PCA_PWM1</td>
<td>DATA</td>
<td>0F3H</td>
</tr>
<tr>
<td>CCAPM2</td>
<td>DATA</td>
<td>0DCH</td>
</tr>
<tr>
<td>CCAP2L</td>
<td>DATA</td>
<td>0ECH</td>
</tr>
<tr>
<td>CCAP2H</td>
<td>DATA</td>
<td>0FCH</td>
</tr>
<tr>
<td>PCA_PWM2</td>
<td>DATA</td>
<td>0F4H</td>
</tr>
<tr>
<td>CCAPM3</td>
<td>DATA</td>
<td>0DDH</td>
</tr>
<tr>
<td>CCAP3L</td>
<td>DATA</td>
<td>0EDH</td>
</tr>
<tr>
<td>CCAP3H</td>
<td>DATA</td>
<td>0FDH</td>
</tr>
<tr>
<td>PCA_PWM3</td>
<td>DATA</td>
<td>0F5H</td>
</tr>
</tbody>
</table>

ORG 0000H
LJMP MAIN

ORG 0100H
MAIN:

```
MOV SP,#3FH
MOV CCON,#00H
MOV CMOD,#08H ;PCA clock is the system clock
MOV CL,#00H
MOV CH,#0H
MOV CCAPM0,#42H ;PCA module 0 is PWM mode
MOV PCA_PWM0,#080H ;PCA Module 0 outputs 6-bit PWM
MOV CCP0L,#20H ;PWM duty cycle is 50% (40H-20H)/40H
MOV CCP0H,#20H
MOV CCP1L,#0H ;PCA module 1 is PWM mode
MOV CCP1H,#00H ;PCA Module 1 outputs 7-bit PWM
MOV CCP2L,#00H ;PWM duty cycle is 75% (80H-20H)/80H
MOV CCP2H,#00H
MOV CCP3L,#00H ;PCA module 2 is PWM mode
MOV CCP3H,#00H ;PCA Module 2 outputs 8-bit PWM
MOV CCP4L,#00H ;PWM duty cycle is 87.5% (100H-20H)/100H
MOV CCP4H,#00H
MOV CCP5L,#00H ;PCA module 3 is PWM mode
MOV CCP5H,#00H ;PCA Module 3 outputs 10-bit PWM
MOV CCP6L,#00H ;PWM duty cycle is 96.875% (400H-20H)/400H
MOV CCP6H,#00H
MOV CR ;Start PCA timer
JMP $ ;
```

END

---

C code

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

//The test operating frequency is 11.0592 MHz

sfr CCON = 0xd8;
sbit CF = CCON^7;
sbit CR = CCON^6;
sbit CCF3 = CCON^3;
sbit CCF2 = CCON^2;
sbit CCF1 = CCON^1;
sbit CCF0 = CCON^0;
sfr CMOD = 0xd9;
sfr CL = 0xe9;
sfr CH = 0xf9;
sfr CCPM0 = 0xda;
sfr CCP0L = 0xea;
sfr CCP0H = 0xfa;
sfr PCA_PWM0 = 0xf2;
sfr CCP1L = 0xeb;
sfr CCP1H = 0xfc;
sfr CCP1H = 0xf3;
sfr CCP2L = 0xdc;
sfr CCP2H = 0xed;
sfr CCP3L = 0xf4;
sfr CCP3H = 0xef;
```
sfr CCAPM3 = 0xdd;
sfr CCAP3L = 0xed;
sfr CCAP3H = 0xfd;
sfr PCA_PWM3 = 0xf5;

void main()
{
    CCON = 0x00;
    CMOD = 0x08; //PCA clock is the system clock
    CL = 0x00;
    CH = 0x00;
    CCAPM0 = 0x42; //PCA module 0 is PWM mode
    PCA_PWM0 = 0x80; //PCA Module 0 outputs 6-bit PWM
    CCAP0L = 0x20; //PWM duty cycle is 50%[(40H-20H)/40H]
    CCAP0H = 0x20;
    CCAPM1 = 0x42; //PCA module 1 is PWM mode
    PCA_PWM1 = 0x40; //PCA Module 1 outputs 7-bit PWM
    CCAP1L = 0x20; //PWM duty cycle is 75%[(80H-20H)/80H]
    CCAP1H = 0x20;
    CCAPM2 = 0x42; //PCA module 2 is PWM mode
    PCA_PWM2 = 0x00; //PCA Module 1 outputs 8-bit PWM
    CCAP2L = 0x20; //PWM duty cycle is 87.5%[(100H-20H)/100H]
    CCAP2H = 0x20;
    CCAPM3 = 0x42; //PCA module 3 is PWM mode
    PCA_PWM3 = 0xc0; //PCA Module 1 outputs 10-bit PWM
    CCAP3L = 0x20; //PWM duty cycle is 96.875%[(400H-20H)/400H]
    CCAP3H = 0x20;
    CR = 1; //Start PCA timer

    while (1);
}

18.3.2 PCA Capture measurement pulse width

Assembly code

;The test operating frequency is 11.0592 MHz

<table>
<thead>
<tr>
<th>CCON</th>
<th>DATA</th>
<th>0D8H</th>
</tr>
</thead>
<tbody>
<tr>
<td>CF</td>
<td>BIT</td>
<td>CCON.7</td>
</tr>
<tr>
<td>CR</td>
<td>BIT</td>
<td>CCON.6</td>
</tr>
<tr>
<td>CCF3</td>
<td>BIT</td>
<td>CCON.3</td>
</tr>
<tr>
<td>CCF2</td>
<td>BIT</td>
<td>CCON.2</td>
</tr>
<tr>
<td>CCF1</td>
<td>BIT</td>
<td>CCON.1</td>
</tr>
<tr>
<td>CCF0</td>
<td>BIT</td>
<td>CCON.0</td>
</tr>
<tr>
<td>CMOD</td>
<td>DATA</td>
<td>0D9H</td>
</tr>
<tr>
<td>CL</td>
<td>DATA</td>
<td>0E9H</td>
</tr>
<tr>
<td>CH</td>
<td>DATA</td>
<td>0F9H</td>
</tr>
<tr>
<td>CCAPM0</td>
<td>DATA</td>
<td>0DAH</td>
</tr>
<tr>
<td>CCAP0L</td>
<td>DATA</td>
<td>0EAH</td>
</tr>
<tr>
<td>CCAP0H</td>
<td>DATA</td>
<td>0FAH</td>
</tr>
<tr>
<td>PCA_PWM0</td>
<td>DATA</td>
<td>0F2H</td>
</tr>
<tr>
<td>CCAPM1</td>
<td>DATA</td>
<td>0DBH</td>
</tr>
<tr>
<td>CCAP1L</td>
<td>DATA</td>
<td>0EBH</td>
</tr>
<tr>
<td>CCAP1H</td>
<td>DATA</td>
<td>0FBH</td>
</tr>
<tr>
<td>PCA_PWM1</td>
<td>DATA</td>
<td>0F3H</td>
</tr>
<tr>
<td>CCAPM2</td>
<td>DATA</td>
<td>0DCH</td>
</tr>
</tbody>
</table>

Nantong guoxin Microelectronics Co., Ltd.  Tel: 0513-5501 2928/2929/2966  Fax: 0513-5501 2926/2956/2947  - 354 -
CCAP2L DATA 0ECH
CCAP2H DATA 0FCH
PCA_PWM2 DATA 0F4H
CCAPM3 DATA 0DDH
CCAP3L DATA 0EDH
CCAP3H DATA 0FDH
PCA_PWM3 DATA 0F5H
CNT DATA 20H
COUNT0 DATA 21H ;3 bytes
COUNT1 DATA 24H ;3 bytes
LENGTH DATA 27H ;3 bytes, (COUNT1-COUNT0)

ORG 0000H
LJMP MAIN
ORG 003BH
LJMP PCAISR

ORG 0100H
PCAISR:
PUSH ACC
PUSH PSW
JNB CF,CHECKCCF0
CLR CF ;Clear interrupt flag
INC CNT ; PCA timing overflow times+1
CHECKCCF0:
JNB CCF0,ISREXIT
CLR CCF0
MOV COUNT0,COUNT1 ;Back up the last captured value
MOV COUNT0+1,COUNT1+1
MOV COUNT0+2,COUNT1+2
MOV COUNT1,CNT ;Save this captured value
MOV COUNT1+1,CCAP0H
MOV COUNT1+2,CCAP0L
CLR C ;Calculate twice the capture difference
SUBB A,COUNT1+2
MOV LENGTH+2,A
MOV A,COUNT1+1
SUBB A,COUNT0+1
MOV LENGTH+1,A
MOV A,COUNT1
SUBB A,COUNT0
MOV LENGTH,A ;LENGTH is the pulse width of the capture.
ISREXIT:
POP PSW
POP ACC
RETI

MAIN:
MOV SP,#3FH
CLR A
MOV CNT,A ;User variable initialization
MOV COUNT0,A
MOV COUNT0+1,A
MOV COUNT0+2,A
MOV COUNT1,A
MOV COUNT1+1,A  
MOV COUNT1+2,A  
MOV LENGTH,A  
MOV LENGTH+1,A  
MOV LENGTH+2,A  
MOV CCON,#00H  
MOV CMOD,#09H  ;PCA clock is system clock, enabling PCA timing interrupt  
MOV CL,#00H  
MOV CH,#0H  
MOV CCAPM0,#11H  ;PCA module 0 is 16 bit capture mode (descent edge capture)  
;MOV CCAPM0,#21H  ;PCA module 0 is 16 bit capture mode (rise edge capture)  
;MOV CCAPM0,#31H  ;PCA module 0 is 16-bit capture mode (edge capture)  
MOV CCAP0L,#00H  
MOV CCAP0H,#00H  
SETB CR ;Start PCA timer  
SETB EA  
JMP $  
END

C code

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

//The test operating frequency is 11.0592 MHz

sfr CCON = 0xd8;  
sbit CF = CCON^7;  
sbit CR = CCON^6;  
sbit CCF3 = CCON^3;  
sbit CCF2 = CCON^2;  
sbit CCF1 = CCON^1;  
sbit CCF0 = CCON^0;  
sfr CMOD = 0xd9;  
sfr CL = 0xe9;  
sfr CH = 0xf9;  
sfr CCAPM0 = 0xda;  
sfr CCAP0L = 0xea;  
sfr CCAP0H = 0xfa;  
sfr PCA_PWM0 = 0xf2;  
sfr CCAPM1 = 0xdb;  
sfr CCAP1L = 0xeb;  
sfr CCAP1H = 0xfc;  
sfr PCA_PWM1 = 0xf3;  
sfr CCAPM2 = 0xdc;  
sfr CCAP2L = 0xed;  
sfr CCAP2H = 0xfd;  
sfr PCA_PWM2 = 0xf4;  
sfr CCAPM3 = 0xdd;  
sfr CCAP3L = 0xff;  
sfr CCAP3H = 0xff;  
sfr PCA_PWM3 = 0xf5;  

unsigned char cnt;   //store PCA timing overflow times  
unsigned long count0;  //Record the previous capture value  

Nantong guoxin Microelectronics Co., Ltd.  
Tel: 0513-5501 2928/2929/2966  
Fax: 0513-5501 2926/2956/2947  
- 356 -
unsigned long count1; //Record the capture value for this time
unsigned long length; //store time length of signal

void PCA_Isr() interrupt 7
{
    if (CF)
    {
        CF = 0;
        cnt++;
        //PCA timing overflow times+1
    }
    else if (CCF0)
    {
        CCF0 = 0;
        count0 = count1; //Back up the last captured value
        ((unsigned char *)&count1)[3] = CCAP0L;
        ((unsigned char *)&count1)[2] = CCAP0H;
        ((unsigned char *)&count1)[1] = cnt;
        ((unsigned char *)&count1)[0] = 0;
        length = count1 - count0; //length is pulse width for capture
    }
}

void main()
{
    cnt = 0; //User variable initialization
    count0 = 0;
    count1 = 0;
    length = 0;
    CCON = 0x00;
    CMOD = 0x09; //PCA clock is system clock, enabling PCA timing interrupt
    CL = 0x00;
    CH = 0x00;
    CCAPM0 = 0x11; //PCA module 0 is 16 bit capture mode (descent edge capture)
    CCAP0L = 0x00;
    CCAP0H = 0x00;
    CR = 1; //Start PCA timer
    EA = 1;
    while (1);
}

### 18.3.3 PCA implements 16-bit software timing

#### Assembly code

;The test operating frequency is 11.0592 MHz

<table>
<thead>
<tr>
<th>Register</th>
<th>DATA</th>
<th>0D8H</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCON</td>
<td>DATA</td>
<td>0D8H</td>
</tr>
<tr>
<td>CF</td>
<td>BIT</td>
<td>CCON.7</td>
</tr>
<tr>
<td>CR</td>
<td>BIT</td>
<td>CCON.6</td>
</tr>
<tr>
<td>CCAPM0</td>
<td>BIT</td>
<td>CCON.3</td>
</tr>
<tr>
<td>CCAP0L</td>
<td>BIT</td>
<td>CCON.2</td>
</tr>
<tr>
<td>CCAP0H</td>
<td>BIT</td>
<td>CCON.1</td>
</tr>
<tr>
<td>CCF0</td>
<td>BIT</td>
<td>CCON.0</td>
</tr>
<tr>
<td>CMOD</td>
<td>DATA</td>
<td>0D9H</td>
</tr>
</tbody>
</table>
CL DATA 0F9H
CH DATA 0F9H
CCAPM0 DATA 0DAH
CCAP0L DATA 0EAH
CCAP0H DATA 0FAH
PCA_PWM0 DATA 0F2H
CCAPM1 DATA 0DBH
CCAP1L DATA 0EBH
CCAP1H DATA 0FBH
PCA_PWM1 DATA 0F3H
CCAPM2 DATA 0DCH
CCAP2L DATA 0ECH
CCAP2H DATA 0FCH
PCA_PWM2 DATA 0F4H
CCAPM3 DATA 0DDH
CCAP3L DATA 0EDH
CCAP3H DATA 0FDH
PCA_PWM3 DATA 0F5H

T50HZ EQU 2400H ;11059200/12/2/50

ORG 0000H
LJMP MAIN
ORG 003BH
LJMP PCAISR

ORG 0100H

PCAISR:
PUSH ACC
PUSH PSW
CLR CCF0
MOV A,CCAP0L
ADD A,#LOW T50HZ
MOV CCAP0L,A
MOV A,CCAP0H
ADDC A,#HIGH T50HZ
MOV CCAP0H,A
CPL P1.0 ;Test port, flashing frequency is 50Hz
POP PSW
POP ACC
RETI

MAIN:
MOV SP,#3FH
MOV CCON,#00H
MOV CMOD,#00H ;PCA clock is the system clock/12
MOV CL,#00H
MOV CH,#0H
MOV CCAPM0,#49H ;PCA module 0 is 16-bit timer mode
MOV CCAP0L,#LOW T50HZ
MOV CCAP0H,#HIGH T50HZ
SETB CR ;Start PCA timer
SETB EA

JMP $
C code

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

//The test operating frequency is 11.0592 MHz
#define T50HZ  (11059200L / 12 / 2 / 50)

sfr CCON = 0xd8;
sbit CF = CCON^7;
sbit CR = CCON^6;
sbit CCF3 = CCON^3;
sbit CCF2 = CCON^2;
sbit CCF1 = CCON^1;
sbit CCF0 = CCON^0;
sfr CMOD = 0xd9;
sfr CL = 0xe9;
sfr CH = 0xf9;
sfr CCAPM0 = 0xda;
sfr CCAP0L = 0xea;
sfr CCAP0H = 0xfa;
sfr PCA_PWM0 = 0xf2;
sfr CCAPM1 = 0xdb;
sfr CCAP1L = 0xeb;
sfr CCAP1H = 0xfc;
sfr PCA_PWM1 = 0xf3;
sfr CCAPM2 = 0xdc;
sfr CCAP2L = 0xed;
sfr CCAP2H = 0xfd;
sfr PCA_PWM2 = 0xf4;
sfr CCAPM3 = 0xdd;
sfr CCAP3L = 0xed;
sfr CCAP3H = 0xfd;
sfr PCA_PWM3 = 0xf5;

sbit P10 = P1^0;

unsigned int value;

void PCA_Isr() interrupt 7
{
    CCF0 = 0;
    CCAP0L = value;
    CCAP0H = value >> 8;
    value += T50HZ;
    P10 = !P10;  //Test port
}

void main()
{
    CCON = 0x00;
    CMOD = 0x00;  //The PCA clock is the system clock/12
    CL = 0x00;
    CH = 0x00;

    //The test operating frequency is 11.0592 MHz
    #define T50HZ  (11059200L / 12 / 2 / 50)
    sfr CCON = 0xd8;
    sbit CF = CCON^7;
    sbit CR = CCON^6;
    sbit CCF3 = CCON^3;
    sbit CCF2 = CCON^2;
    sbit CCF1 = CCON^1;
    sbit CCF0 = CCON^0;
    sfr CMOD = 0xd9;
    sfr CL = 0xe9;
    sfr CH = 0xf9;
    sfr CCAPM0 = 0xda;
    sfr CCAP0L = 0xea;
    sfr CCAP0H = 0xfa;
    sfr PCA_PWM0 = 0xf2;
    sfr CCAPM1 = 0xdb;
    sfr CCAP1L = 0xeb;
    sfr CCAP1H = 0xfc;
    sfr PCA_PWM1 = 0xf3;
    sfr CCAPM2 = 0xdc;
    sfr CCAP2L = 0xed;
    sfr CCAP2H = 0xfd;
    sfr PCA_PWM2 = 0xf4;
    sfr CCAPM3 = 0xdd;
    sfr CCAP3L = 0xed;
    sfr CCAP3H = 0xfd;
    sfr PCA_PWM3 = 0xf5;

    sbit P10 = P1^0;

    unsigned int value;

    void PCA_Isr() interrupt 7
    {
        CCF0 = 0;
        CCAP0L = value;
        CCAP0H = value >> 8;
        value += T50HZ;
        P10 = !P10;  //Test port
    }

    void main()
    {
        CCON = 0x00;
        CMOD = 0x00;  //The PCA clock is the system clock/12
        CL = 0x00;
        CH = 0x00;
```
CCAPM0 = 0x49; //PCA module 0 is 16-bit timer mode
tuple = T50HZ;
CCAP0L = value;
CCAP0H = value >> 8;
value += T50HZ;
CR = 1; //Start PCA timer
EA = 1;

while (1);
}

18.3.4 PCA Output high speed pulse

Assembly code

;The test operating frequency is 11.0592 MHz

CCON DATA 0D8H
CF BIT CCON.7
CR BIT CCON.6
CCF3 BIT CCON.3
CCF2 BIT CCON.2
CCF1 BIT CCON.1
CCF0 BIT CCON.0
CMOD DATA 0D9H
CL DATA 0E9H
CH DATA 0F9H
CCAPM0 DATA 0DAH
CCAP0L DATA 0EAH
CCAP0H DATA 0FAH
PCA_PWM0 DATA 0F2H
CCAPM1 DATA 0DBH
CCAP1L DATA 0EBH
CCAP1H DATA 0FBH
PCA_PWM1 DATA 0F3H
CCAPM2 DATA 0DCH
CCAP2L DATA 0ECH
CCAP2H DATA 0FCH
PCA_PWM2 DATA 0F4H
CCAPM3 DATA 0DDH
CCAP3L DATA 0EDH
CCAP3H DATA 0FDH
PCA_PWM3 DATA 0F5H

T38K4HZ EQU 90H ; 11059200/2/38400

ORG 0000H
LJMP MAIN
ORG 003BH
LJMP PCAISR

ORG 0100H

PCAISR:
PUSH ACC
PUSH PSW
CLR CCF0
MOV A, CCAP0L
ADD A,#LOW T38K4HZ
MOV CCAP0L,A
MOV A,CCAP0H
ADDC A,#HIGH T38K4HZ
MOV CCAP0H,A
POP PSW
POP ACC
RETI

MAIN:
MOV SP,#3FH
MOV CCON,#00H
MOV CMOD,#08H ;PCA clock is the system clock
MOV CL,#00H
MOV CH,#0H
MOV CCAPM0,#4DH          ;PCA module 0 is in 16 bit timer mode and enables pulse output
MOV CCAP0L,#LOW T38K4HZ
MOV CCAP0H,#HIGH T38K4HZ
SETB CR ;Start PCA timer
SETB EA
JMP $
END

C code
#include "reg51.h"
#include "intrins.h"

//The test operating frequency is 11.0592 MHz
#define T38K4HZ  (11059200L / 2 / 38400)
sfr CCON = 0xd8;
sbit CF = CCON^7;
sbit CR = CCON^6;
sbit CCF3 = CCON^3;
sbit CCF2 = CCON^2;
sbit CCF1 = CCON^1;
sbit CCF0 = CCON^0;
sfr CMOD = 0xd9;
sfr CL = 0xe9;
sfr CH = 0xf9;
sfr CCAPM0 = 0xea;
sfr CCAP0L = 0xfa;
sfr CCAP0H = 0xfb;
sfr PCA_PWM0 = 0xf2;
sfr CCAPM1 = 0xeb;
sfr CCAP1L = 0xfc;
sfr CCAP1H = 0xfd;
sfr PCA_PWM1 = 0xf3;
sfr CCAPM2 = 0xf4;
sfr CCAP2L = 0xfd;
sfr CCAP2H = 0xfe;
sfr PCA_PWM2 = 0xf5;
sfr CCAPM3 = 0xfd;
sfr CCAP3L = 0xed;
sfr CCAP3H = 0xfd;
sfr PCA_PWM3 = 0xf5;

unsigned int value;

void PCA_Isr() interrupt 7
{
    CCF0 = 0;
    CCAP0L = value;
    CCAP0H = value >> 8;
    value += T38K4HZ;
}

void main()
{
    CCON = 0x00;
    CMOD = 0x08; //PCA clock is the system clock
    CL = 0x00;
    CH = 0x00;
    CCAPM0 = 0x4d; //PCA module 0 is in 16 bit timer mode and enables pulse output
    value = T38K4HZ;
    CCAP0L = value;
    CCAP0H = value >> 8;
    value += T38K4HZ;
    CR = 1; //Start PCA timer
    EA = 1;

    while (1);
}

18.3.5 PCA extends external interrupt

Assembly code
;The test operating frequency is 11.0592 MHz

<table>
<thead>
<tr>
<th>CCON</th>
<th>DATA</th>
<th>0D8H</th>
</tr>
</thead>
<tbody>
<tr>
<td>CF</td>
<td>BIT</td>
<td>CCON.7</td>
</tr>
<tr>
<td>CR</td>
<td>BIT</td>
<td>CCON.6</td>
</tr>
<tr>
<td>CCF3</td>
<td>BIT</td>
<td>CCON.3</td>
</tr>
<tr>
<td>CCF2</td>
<td>BIT</td>
<td>CCON.2</td>
</tr>
<tr>
<td>CCF1</td>
<td>BIT</td>
<td>CCON.1</td>
</tr>
<tr>
<td>CCF0</td>
<td>BIT</td>
<td>CCON.0</td>
</tr>
<tr>
<td>CMOD</td>
<td>DATA</td>
<td>0D9H</td>
</tr>
<tr>
<td>CL</td>
<td>DATA</td>
<td>0E9H</td>
</tr>
<tr>
<td>CH</td>
<td>DATA</td>
<td>0F9H</td>
</tr>
<tr>
<td>CCAPM0</td>
<td>DATA</td>
<td>0DAH</td>
</tr>
<tr>
<td>CCAP0L</td>
<td>DATA</td>
<td>0EAH</td>
</tr>
<tr>
<td>CCAP0H</td>
<td>DATA</td>
<td>0FAH</td>
</tr>
<tr>
<td>PCA_PWM0</td>
<td>DATA</td>
<td>0F2H</td>
</tr>
<tr>
<td>CCAPM1</td>
<td>DATA</td>
<td>0BH</td>
</tr>
<tr>
<td>CCAP1L</td>
<td>DATA</td>
<td>0EBH</td>
</tr>
<tr>
<td>CCAP1H</td>
<td>DATA</td>
<td>0FBH</td>
</tr>
<tr>
<td>PCA_PWM1</td>
<td>DATA</td>
<td>03H</td>
</tr>
<tr>
<td>CCAPM2</td>
<td>DATA</td>
<td>0CH</td>
</tr>
<tr>
<td>CCAP2L</td>
<td>DATA</td>
<td>0ECH</td>
</tr>
</tbody>
</table>
CCAP2H DATA 0FCH
PCA_PWM2 DATA 0F4H
CCAPM3 DATA 0DDH
CCAP3L DATA 0EDH
CCAP3H DATA 0FDH
PCA_PWM3 DATA 0F5H

ORG 0000H
LJMP MAIN
ORG 003BH
LJMP PCAISR

ORG 0100H

PCAISR:
CLR CCF0
CPL P1.0
RETI

MAIN:
MOV SP,#3FH
MOV CCON,#00H
MOV CMOD,#08H ;PCA clock is the system clock
MOV CL,#00H
MOV CH,#0H
MOV CCAPM0,#11            ;Extend external port CCP0 to drop edge interrupt port
; MOV CCAPM0,#21H          ;Extend external port CCP0 to upper-edge interrupt port
; MOV CCAPM0,#31H          ;Extend external port CCP0 to edge interrupt port
MOV CCAP0L,#0
MOV CCAP0H,#0
SETB CR ;Start PCA timer
SETB EA
JMP $
END

C code
#include "reg51.h"
#include "intrins.h"

//The test operating frequency is 11.0592 MHz

sfr   CCON = 0xd8;
sbit   CF = CCON^7;
sbit   CR = CCON^6;
sbit   CCF3 = CCON^3;
sbit   CCF2 = CCON^2;
sbit   CCF1 = CCON^1;
sbit   CCF0 = CCON^0;
sfr   CMOD = 0xd9;
sfr   CL = 0xe9;
sfr   CH = 0xf9;
sfr   CCAPM0 = 0xda;
sfr   CCAP0L = 0xea;
sfr   CCAP0H = 0xfa;
sfr   PCA_PWM0 = 0xfb;
sfr CCAPM1 = 0xdb;
sfr CCAP1L = 0xeb;
sfr CCAP1H = 0xfb;
sfr PCA_PWM1 = 0xf3;
sfr CCAPM2 = 0xdc;
sfr CCAP2L = 0xec;
sfr CCAP2H = 0xfc;
sfr PCA_PWM2 = 0xf4;
sfr CCAPM3 = 0xdd;
sfr CCAP3L = 0xed;
sfr CCAP3H = 0xfd;
sfr PCA_PWM3 = 0xf5;

sbit P10 = P1^0;

void PCA_Isr() interrupt 7
{
    CCF0 = 0;
    P10 = !P10;
}

void main()
{
    CCON = 0x00;
    CMOD = 0x08; //PCA clock is the system clock
    CL = 0x00;
    CH = 0x00;
    CCAPM0 = 0x11; //Extend external port CCP0 to drop edge interrupt port
    // CCAPM0 = 0x21; //Extend external port CCP0 to upper-edge interrupt port
    // CCAPM0 = 0x31; //Extend external port CCP0 to edge interrupt port
    CCAP0L = 0;
    CCAP0H = 0;
    CR = 1; //Start PCA timer
    EA = 1;

    while (1);
}
19 Enhanced PWM

A set of individually enhanced 8-channel PWM waveform generators are integrated in the STC8F family of microcontrollers. There is a 15-bit PWM counter in the PWM waveform generator which is used for 8 channel PWMs. The initial level of each PWM can be set. In addition, two counters T1 and T2 are designed in the PWM waveform generator to control the waveform hopping for each PWM. The width of high and low level of each PWM can be set very flexibly, so that the PWM duty ratio and PWM output delay can be controlled. Because the 8 PWMs are independent and the initial state of each PWM can be set, any two of them can be used together to achieve complementary applications such as symmetrical output and dead band control.

The enhanced PWM waveform generators also feature the ability to monitor external abnormal events, such as external port P3.5 abnormal level and the abnormal comparator results. It can be used to shutdown PWM outputs immediately. The PWM waveform generator can also be associated with an ADC. Any point in the PWM cycle can be set to trigger the ADC conversion event.

19.1 PWM Related Registers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Address</th>
<th>Bit Address and Symbol</th>
<th>Value after reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWMCFG</td>
<td>PWM Configuration Register</td>
<td>F1H</td>
<td>B7 B6 B5 B4 B3 B2 B1 B0</td>
<td>00xx,xxxxxx</td>
</tr>
<tr>
<td>PWMIF</td>
<td>PWM Interrupt Flag register</td>
<td>F6H</td>
<td>C7IF C6IF C5IF C4IF C3IF C2IF C1IF C0IF</td>
<td>0000,00000000</td>
</tr>
<tr>
<td>PWMFDCR</td>
<td>PWM Fault Detection Control Register</td>
<td>F7H</td>
<td>INVCMP INVIO ENFD FLTFLIO EFDI FDCMP FDI0 FDIF</td>
<td>0000,00000000</td>
</tr>
<tr>
<td>PWMCR</td>
<td>PWM Control register</td>
<td>FEH</td>
<td>ENPWM ECBI</td>
<td>00xx,xxxxxx</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Address</th>
<th>Bit Address and Symbol</th>
<th>Value after reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWMCH</td>
<td>PWM Counter High</td>
<td>FFF0H</td>
<td>B7 B6 B5 B4 B3 B2 B1 B0</td>
<td>x000,00000000</td>
</tr>
<tr>
<td>PWML</td>
<td>PWM Counter low</td>
<td>FFF1H</td>
<td></td>
<td>0000,00000000</td>
</tr>
<tr>
<td>PWMCKS</td>
<td>PWM Clock Selection register</td>
<td>FFF2H</td>
<td>- - - - - - - - SELLT2</td>
<td>PWM_PS[3:0] xxx0,00000000</td>
</tr>
<tr>
<td>TADCPH</td>
<td>Trigger ADC count value high</td>
<td>FFF3H</td>
<td>-</td>
<td>x000,00000000</td>
</tr>
<tr>
<td>TADCPL</td>
<td>Trigger ADC count value low</td>
<td>FFF4H</td>
<td></td>
<td>0000,00000000</td>
</tr>
<tr>
<td>PWM0T1H</td>
<td>PWM0 Timer1 high</td>
<td>FF00H</td>
<td>B7 B6 B5 B4 B3 B2 B1 B0</td>
<td>x000,00000000</td>
</tr>
<tr>
<td>PWM0T1L</td>
<td>PWM0 Timer1 low</td>
<td>FF01H</td>
<td></td>
<td>0000,00000000</td>
</tr>
<tr>
<td>PWM0T2H</td>
<td>PWM0 Timer2 high</td>
<td>FF02H</td>
<td>- - - - - - - - - - -</td>
<td>0000,00000000</td>
</tr>
<tr>
<td>PWM0T2L</td>
<td>PWM0 Timer2 low</td>
<td>FF03H</td>
<td></td>
<td>0000,00000000</td>
</tr>
<tr>
<td>PWM0CR</td>
<td>PWM0 Control register</td>
<td>FF04H</td>
<td>ENC0O C0IN1 - - - - - - CO_S[1:0] CO_S[0:0] E0C0 E0T2S E0T1S</td>
<td>0000,00000000</td>
</tr>
<tr>
<td>PWM0HLD</td>
<td>PWM0 Level Hold Control Register</td>
<td>FF05H</td>
<td>- - - - - - - - - - -</td>
<td>0000,00000000</td>
</tr>
<tr>
<td>PWM0TH1</td>
<td>PWM0 Timer1 high</td>
<td>FF06H</td>
<td></td>
<td>x000,00000000</td>
</tr>
<tr>
<td>PWM0TH1L</td>
<td>PWM0 Timer1 low</td>
<td>FF07H</td>
<td></td>
<td>0000,00000000</td>
</tr>
<tr>
<td>PWM1T1H</td>
<td>PWM1 Timer1 high</td>
<td>FF10H</td>
<td>- - - - - -</td>
<td>x000,00000000</td>
</tr>
<tr>
<td>PWM1T1L</td>
<td>PWM1 Timer1 low</td>
<td>FF11H</td>
<td></td>
<td>0000,00000000</td>
</tr>
<tr>
<td>Register</td>
<td>Description</td>
<td>Offset</td>
<td>Data Type</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------------------</td>
<td>--------</td>
<td>-----------</td>
<td></td>
</tr>
<tr>
<td>PWM1T2H</td>
<td>PWM1 Timer2 high</td>
<td>FF12H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM1T2L</td>
<td>PWM1 Timer2 low</td>
<td>FF13H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM1CR</td>
<td>PWM1 Control register</td>
<td>FF14H</td>
<td>ENC1O, C1IN1, C1_S[1:0], EC1I, EC1T2SI, EC1T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM1HLD</td>
<td>PWM1 Level Hold Control Register</td>
<td>FF15H</td>
<td>HC1H, HC1L</td>
<td></td>
</tr>
<tr>
<td>PWM2T1H</td>
<td>PWM2 Timer1 high</td>
<td>FF20H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM2T1L</td>
<td>PWM2 Timer1 low</td>
<td>FF21H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM2T2H</td>
<td>PWM2 Timer2 high</td>
<td>FF22H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM2T2L</td>
<td>PWM2 Timer2 low</td>
<td>FF23H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM2CR</td>
<td>PWM2 Control register</td>
<td>FF24H</td>
<td>ENC2O, C2IN1, C2_S[1:0], EC2I, EC2T2SI, EC2T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM2HLD</td>
<td>PWM2 Level Hold Control Register</td>
<td>FF25H</td>
<td>HC2H, HC2L</td>
<td></td>
</tr>
<tr>
<td>PWM3T1H</td>
<td>PWM3 Timer1 high</td>
<td>FF30H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM3T1L</td>
<td>PWM3 Timer1 low</td>
<td>FF31H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM3T2H</td>
<td>PWM3 Timer2 high</td>
<td>FF32H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM3T2L</td>
<td>PWM3 Timer2 low</td>
<td>FF33H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM3CR</td>
<td>PWM3 Control register</td>
<td>FF34H</td>
<td>ENC3O, C3IN1, C3_S[1:0], EC3I, EC3T2SI, EC3T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM3HLD</td>
<td>PWM3 Level Hold Control Register</td>
<td>FF35H</td>
<td>HC3H, HC3L</td>
<td></td>
</tr>
<tr>
<td>PWM4T1H</td>
<td>PWM4 Timer1 high</td>
<td>FF40H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM4T1L</td>
<td>PWM4 Timer1 low</td>
<td>FF41H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM4T2H</td>
<td>PWM4 Timer2 high</td>
<td>FF42H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM4T2L</td>
<td>PWM4 Timer2 low</td>
<td>FF43H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM4CR</td>
<td>PWM4 Control register</td>
<td>FF44H</td>
<td>ENC4O, C4IN1, C4_S[1:0], EC4I, EC4T2SI, EC4T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM4HLD</td>
<td>PWM4 Level Hold Control Register</td>
<td>FF45H</td>
<td>HC4H, HC4L</td>
<td></td>
</tr>
<tr>
<td>PWM5T1H</td>
<td>PWM5 Timer1 high</td>
<td>FF50H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM5T1L</td>
<td>PWM5 Timer1 low</td>
<td>FF51H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM5T2H</td>
<td>PWM5 Timer2 high</td>
<td>FF52H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM5T2L</td>
<td>PWM5 Timer2 low</td>
<td>FF53H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM5CR</td>
<td>PWM5 Control register</td>
<td>FF54H</td>
<td>ENC5O, C5IN1, C5_S[1:0], EC5I, EC5T2SI, EC5T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM5HLD</td>
<td>PWM5 Level Hold Control Register</td>
<td>FF55H</td>
<td>HC5H, HC5L</td>
<td></td>
</tr>
<tr>
<td>PWM6T1H</td>
<td>PWM6 Timer1 high</td>
<td>FF60H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM6T1L</td>
<td>PWM6 Timer1 low</td>
<td>FF61H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM6T2H</td>
<td>PWM6 Timer2 high</td>
<td>FF62H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM6T2L</td>
<td>PWM6 Timer2 low</td>
<td>FF63H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM6CR</td>
<td>PWM6 Control register</td>
<td>FF64H</td>
<td>ENC6O, C6IN1, C6_S[1:0], EC6I, EC6T2SI, EC6T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM6HLD</td>
<td>PWM6 Level Hold Control Register</td>
<td>FF65H</td>
<td>HC6H, HC6L</td>
<td></td>
</tr>
<tr>
<td>PWM7T1H</td>
<td>PWM7 Timer1 high</td>
<td>FF70H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM7T1L</td>
<td>PWM7 Timer1 low</td>
<td>FF71H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM7T2H</td>
<td>PWM7 Timer2 high</td>
<td>FF72H</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PWM Configuration Register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWMCFG</td>
<td>F1H</td>
<td>CBIF</td>
<td>ETADC</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

CBIF: The flag bit of PWM interrupt happened when the PWM counter returns to zero. The bit will be set to 1 by hardware when the 15-bit PWM counter overflows and returns to zero, and requests interrupt tp CPU. It should be cleared by software.

ETADC: Whether the PWM is associated with the ADC or not.
0: PWM is not associated with ADC.
1: PWM is associated with ADC. A/D conversion is enabled to be triggered at a certain point in the PWM cycle. TADCPH and TADCPL are used to set the counter value.

PWM Interrupt Flag register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWMIF</td>
<td>F6H</td>
<td>C7IF</td>
<td>C6IF</td>
<td>C5IF</td>
<td>C4IF</td>
<td>C3IF</td>
<td>C2IF</td>
<td>C1IF</td>
<td>C0IF</td>
</tr>
</tbody>
</table>

CnIF: The interrupt flag bit of PWMn. Every PWM flip point 1 and flip point 2 can be set. When the flip event occurs, this bit is set by the hardware automatically, and requests an interrupt to CPU. It should be cleared by software.

PWM Fault Detection Control Register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWMFDCR</td>
<td>F7H</td>
<td>INVCMP</td>
<td>INVIO</td>
<td>ENFD</td>
<td>FLTFLIO</td>
<td>EFDI</td>
<td>FDCMP</td>
<td>FDIO</td>
<td>FDIF</td>
</tr>
</tbody>
</table>

INVCMP: Fault signal of comparator result selection bit
0: the fault signal is the comparator result changing from low to high.
1: the fault signal is the comparator result changing from high to low.

INVIO: Fault signal of external port P3.5 selection bit
0: the fault signal is the external port P3.5 signal changing from low to high.
1: the fault signal is the external port P3.5 signal changing from high to low.

ENFD: PWM external fault detection enable bit
0: disable the PWM external fault detection.
1: enable the PWM external fault detection.

FLTFLIO: PWM output port control bit when external PWM fault occurs
0: the PWM output port does not change when external PWM fault occurs.
1: the PWM output port is set as high impedance input mode when external PWM default occurs.

Note: Only the port whose corresponding ENCnO = 1 is forcibly in high impedance state.

EFDI: PWM fault detection interrupt enable bit
0: disable PWM fault detection interrupt (FDIF will still be set by hardware.)
1: enable PWM fault detection interrupt

FDCMP: fault detection of comparator output enable bit
0: the comparator is not associated with PWM.
1: the source of PWM fault detection is comparator output. (The fault type is set by INVCMP.)

FDIO: P3.5 level fault detection enable bit
0: P3.5 level is not associated with PWM
1: the source of PWM fault detection is P3.5. (The fault type is set by INVIO.)

FDIF: the interrupt flag bit of PWM fault detection
It is set automatically by the hardware when a PWM fault occurs. The fault can be comparator output changing from low to high or P2.4 changing high from low.
If EFDI=1, the program will jump to the corresponding interrupt entry to execute interrupt service routine.
It should be cleared by software.

### PWM Control register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWMCR</td>
<td>FEH</td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

ENPWM: Enhanced PWM waveform generator enable bit
0: disable PWM waveform generator.
1: enable PWM waveform generator, and the PWM counter starts counting.

Important notes on ENPWM control bit:
Once ENPWM is enabled, the internal PWM counter will start counting immediately and compare with the value of T1 / T2 two reversal points. ENPWM must be enabled after all other PWM settings are completed. These settings include T1 / T2 flip-flop settings, initial level settings, PWM fault detection settings, and PWM interrupt settings.
The ENPWM control bit is both the enable bit for the entire PWM module and the control bit for the PWM counter to start counting. If the ENPWM control bit is off during PWM counter counting, the PWM count stops immediately. And when the ENPWM control bit is enabled again, the PWM count starts counting from 0 and does not memorize the count value before the PWM stops counting.

ECBI: PWM counter return-to-zero interrupt enable bit
0: disable PWM counter return-to-zero interrupt. (CBIF will still be set by hardware.)
1: enable PWM counter return-to-zero interrupt.

### PWM Counter Registers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWMCH</td>
<td>FFF0H</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWMCL</td>
<td>FFF1H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The PWM counter is a 15-bit register that can be set to any value between 1 and 32767 as the PWM cycle. The counter inside the PWM waveform generator counts from 0 and increments by 1 every PWM clock cycle.
When the internal counter reaches the PWM cycle set by [PWMCH, PWMCL], the internal counter of the PWM waveform generator will count from 0 again, and the PWM return-to-zero interrupt flag bit CBIF will be set by the hardware automatically. If ECBI = 1, the program will jump to the corresponding interrupt entry address to execute the interrupt service routine.

### PWM Clock Selection register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
</table>
| PWMCKS   | FFF2H   | -  | -  | -  | SELT2 |    | PWM_PS[3:0] | ![](image)

SELT2: PWM clock source selection bit
0: The PWM clock source is the clock generated by the system clock being divided by the frequency divider.
1: The PWM clock source is the overflow pulse of timer 2.

<table>
<thead>
<tr>
<th>SELT2</th>
<th>PWM_PS[3:0]</th>
<th>PWM input clock frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>xxxx</td>
<td>Overflow pulse of timer 2</td>
</tr>
<tr>
<td>0</td>
<td>0000</td>
<td>SYSclk/1</td>
</tr>
<tr>
<td>0</td>
<td>0001</td>
<td>SYSclk/2</td>
</tr>
<tr>
<td>0</td>
<td>0010</td>
<td>SYSclk/3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0</td>
<td>x</td>
<td>SYSclk/(x+1)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0</td>
<td>1111</td>
<td>SYSclk/16</td>
</tr>
</tbody>
</table>

Trigger ADC counter registers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TADCPH</td>
<td>FFF3H</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TADCPL</td>
<td>FFF4H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If ETADC=1 and ADC_POWER=1, TADCPH, TADCPL forms a 15-bit register. In the PWM counting cycle, the hardware will trigger A/D conversion automatically when the internal PWM counting value equals to the value of TADCPH, TADCPL.

PWM Flipping point set count value registers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWM0T1H</td>
<td>FF00H</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM0T1L</td>
<td>FF01H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM0T2H</td>
<td>FF02H</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM0T2L</td>
<td>FF03H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM1T1H</td>
<td>FF10H</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM1T1L</td>
<td>FF11H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM1T2H</td>
<td>FF12H</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM1T2L</td>
<td>FF13H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM2T1H</td>
<td>FF20H</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM2T1L</td>
<td>FF21H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM2T2H</td>
<td>FF22H</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM2T2L</td>
<td>FF23H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM3T1H</td>
<td>FF30H</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM3T1L</td>
<td>FF31H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM3T2H</td>
<td>FF32H</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM3T2L</td>
<td>FF33H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM4T1H</td>
<td>FF40H</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PWM4T1L</td>
<td>FF41H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
{PWMnT1H, PWMnT1L} and {PWMnT2H, PWMnT2L} of every PWM are combined into two 15-bit registers, which are used to control the two flip points of the PWM output waveform in every PWM cycle of each PWM. During the counting cycle of PWM, the output waveform of PWM will be inverted to low level automatically when the internal counting value of PWM is equal to the value of the first set up point value in {PWMnT1H, PWMnT1L}. And the output waveform of the PWM will be inverted to high level automatically when the internal counting value of PWM is equal to the value of the second flip point set by {PWMnT2H, PWMnT2L}.

Note: When the values of {PWMnT1H, PWMnT1L} and {PWMnT2H, PWMnT2L} are set equal, the match of the 2nd set of flip-flops will be ignored and will only flip low.

### PWM channel control registers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWM0CR</td>
<td>FF04H</td>
<td>ENC0O</td>
<td>C0INI</td>
<td>-</td>
<td>C0_S[1:0]</td>
<td>EC0I</td>
<td>EC0T2SI</td>
<td>EC0T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM1CR</td>
<td>FF14H</td>
<td>ENC1O</td>
<td>C1INI</td>
<td>-</td>
<td>C1_S[1:0]</td>
<td>EC1I</td>
<td>EC1T2SI</td>
<td>EC1T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM2CR</td>
<td>FF24H</td>
<td>ENC2O</td>
<td>C2INI</td>
<td>-</td>
<td>C2_S[1:0]</td>
<td>EC2I</td>
<td>EC2T2SI</td>
<td>EC2T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM3CR</td>
<td>FF34H</td>
<td>ENC3O</td>
<td>C3INI</td>
<td>-</td>
<td>C3_S[1:0]</td>
<td>EC3I</td>
<td>EC3T2SI</td>
<td>EC3T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM4CR</td>
<td>FF44H</td>
<td>ENC4O</td>
<td>C4INI</td>
<td>-</td>
<td>C4_S[1:0]</td>
<td>EC4I</td>
<td>EC4T2SI</td>
<td>EC4T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM5CR</td>
<td>FF54H</td>
<td>ENC5O</td>
<td>C5INI</td>
<td>-</td>
<td>C5_S[1:0]</td>
<td>EC5I</td>
<td>EC5T2SI</td>
<td>EC5T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM6CR</td>
<td>FF64H</td>
<td>ENC6O</td>
<td>C6INI</td>
<td>-</td>
<td>C6_S[1:0]</td>
<td>EC6I</td>
<td>EC6T2SI</td>
<td>EC6T1SI</td>
<td></td>
</tr>
<tr>
<td>PWM7CR</td>
<td>FF74H</td>
<td>ENC7O</td>
<td>C7INI</td>
<td>-</td>
<td>C7_S[1:0]</td>
<td>EC7I</td>
<td>EC7T2SI</td>
<td>EC7T1SI</td>
<td></td>
</tr>
</tbody>
</table>

ENCnO: PWM output enable bit
- 0: the corresponding port of PWM channel is GPIO.
- 1: the corresponding port of PWM channel is PWM output port, which is controlled by the PWM waveform generator.

CnINI: the initial level of PWM output
- 0: the initial level of PWM n is low.
- 1: the initial level of PWM n is high.

Cn_S[1:0]: PWM output function pins switch selection, please refer to the function pin switching chapter.

ECnI: interrupt enable bit of PWM n
- 0: disable PWM n interrupt.
1: enable PWM n interrupt.

ECnT2SI: interrupt enable bit of the second flip point of PWM n.
0: disable the interrupt of the second flip point of PWM n.
1: enable the interrupt of the second flip point of PWM n.

ECnT1SI: interrupt enable bit of the first flip point of PWM n.
0: disable the interrupt of the first flip point of PWM n.
1: enable the interrupt of the first flip point of PWM n.

### PWM Level Hold Control Registers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWM0HLD</td>
<td>FF05H</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>HC0H</td>
<td>HC0L</td>
</tr>
<tr>
<td>PWM1HLD</td>
<td>FF15H</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>HC1H</td>
<td>HC1L</td>
</tr>
<tr>
<td>PWM2HLD</td>
<td>FF25H</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>HC2H</td>
<td>HC2L</td>
</tr>
<tr>
<td>PWM3HLD</td>
<td>FF35H</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>HC3H</td>
<td>HC3L</td>
</tr>
<tr>
<td>PWM4HLD</td>
<td>FF45H</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>HC4H</td>
<td>HC4L</td>
</tr>
<tr>
<td>PWM5HLD</td>
<td>FF55H</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>HC5H</td>
<td>HC5L</td>
</tr>
<tr>
<td>PWM6HLD</td>
<td>FF65H</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>HC6H</td>
<td>HC6L</td>
</tr>
<tr>
<td>PWM7HLD</td>
<td>FF75H</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>HC7H</td>
<td>HC7L</td>
</tr>
</tbody>
</table>

HCnH: PWM n outputs high compulsively control bit
0: PWM n output normally.
1: PWM n outputs high compulsively.

HCnL: PWM n outputs low compulsively control bit
0: PWM n output normally.
1: PWM n outputs low compulsively.

### 19.2 Sample program

#### 19.2.1 Output waveforms of any period and arbitrary duty cycle

**Assembly code**

;The test operating frequency is 11.0592MHz

```
P_SW2 DATA 0BAH

PWMCFG DATA 0F1H
PWMIF  DATA 0F6H
PWMFDCR DATA 0F7H
PWMCR  DATA 0FEH
PWMCH  XDATA 0FFF0H
PWMCL  XDATA 0FFF1H
PWMCKS XDATA 0FFF2H
TADCPH XDATA 0FFF3H
TADCPPL XDATA 0FFF4H
PWM0T1H XDATA 0FF00H
PWM0T1L XDATA 0FF01H
PWM0T2H XDATA 0FF02H
PWM0T2L XDATA 0FF03H
```
PWM0CR XDATA 0FF04H
PWM0HLD XDATA 0FF05H
PWM1T1H XDATA 0FF10H
PWM1T1L XDATA 0FF11H
PWM1T2H XDATA 0FF12H
PWM1T2L XDATA 0FF13H
PWM1CR XDATA 0FF14H
PWM1HLD XDATA 0FF15H
PWM2T1H XDATA 0FF20H
PWM2T1L XDATA 0FF21H
PWM2T2H XDATA 0FF22H
PWM2T2L XDATA 0FF23H
PWM2CR XDATA 0FF24H
PWM2HLD XDATA 0FF25H
PWM3T1H XDATA 0FF30H
PWM3T1L XDATA 0FF31H
PWM3T2H XDATA 0FF32H
PWM3T2L XDATA 0FF33H
PWM3CR XDATA 0FF34H
PWM3HLD XDATA 0FF35H
PWM4T1H XDATA 0FF40H
PWM4T1L XDATA 0FF41H
PWM4T2H XDATA 0FF42H
PWM4T2L XDATA 0FF43H
PWM4CR XDATA 0FF44H
PWM4HLD XDATA 0FF45H
PWM5T1H XDATA 0FF50H
PWM5T1L XDATA 0FF51H
PWM5T2H XDATA 0FF52H
PWM5T2L XDATA 0FF53H
PWM5CR XDATA 0FF54H
PWM5HLD XDATA 0FF55H
PWM6T1H XDATA 0FF60H
PWM6T1L XDATA 0FF61H
PWM6T2H XDATA 0FF62H
PWM6T2L XDATA 0FF63H
PWM6CR XDATA 0FF64H
PWM6HLD XDATA 0FF65H
PWM7T1H XDATA 0FF70H
PWM7T1L XDATA 0FF71H
PWM7T2H XDATA 0FF72H
PWM7T2L XDATA 0FF73H
PWM7CR XDATA 0FF74H
PWM7HLD XDATA 0FF75H

ORG 0000H
LJMP MAIN

ORG 0100H

MAIN:
MOV P_SW2,#80H
CLR A
MOV DPTR,#PWMCKS
MOVX @DPTR,A ;The PWM clock is a system clock
MOV A,#10H
MOV DPTR,#PWMCH ;Set the PWM Period to 1000H PWM Clocks
MOVX @DPTR,A
MOV A,#00H
Nantong guoxin Microelectronics Co., Ltd.  Tel: 0513-5501 2928/2929/2966  Fax: 0513-5501 2926/2956/2947  - 372 -
MOV DPTR,#PWMCL
MOVX @DPTR,A
MOV A,#01H
MOV DPTR,#PWM0T1H ;Output low level at count value of 100H
MOVX @DPTR,A
MOV A,#00H
MOV DPTR,#PWM0T1L
MOVX @DPTR,A
MOV A,#05H
MOV DPTR,#PWM0T2H ;Output a high level at a count value of 500H
MOVX @DPTR,A
MOV A,#00H
MOV DPTR,#PWM0T2L
MOVX @DPTR,A
MOV A,#80H
MOV DPTR,#PWM0CR ;Enable PWM0 output
MOVX @DPTR,A
MOV P_SW2,#00H
MOV PWMCR,#080H ;Start the PWM module
JMP $
END

C code

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

//The test operating frequency is 11.0592MHz
sfr P_SW2 = 0xba;
sfr PWMCFG = 0xf1;
sfr PWMIF = 0xf6;
sfr PWMFDCR = 0xf7;
sfr PWMCR = 0xfe;
#define PWMC  (*(unsigned int volatile xdata *)0xfff0)
#define PWMCKS  (*(unsigned char volatile xdata *)0xfff2)
#define TADCP  (*(unsigned int volatile xdata *)0xfff3)
#define PWM0T1  (*(unsigned int volatile xdata *)0xff00)
#define PWM0T2  (*(unsigned int volatile xdata *)0xff02)
#define PWM0CR  (*(unsigned char volatile xdata *)0xff04)
#define PWM0HLD  (*(unsigned char volatile xdata *)0xff05)
#define PWM1T1  (*(unsigned int volatile xdata *)0xff10)
#define PWM1T2  (*(unsigned int volatile xdata *)0xff12)
#define PWM1CR  (*(unsigned char volatile xdata *)0xff14)
#define PWM1HLD  (*(unsigned char volatile xdata *)0xff15)
#define PWM2T1  (*(unsigned int volatile xdata *)0xff20)
#define PWM2T2  (*(unsigned int volatile xdata *)0xff22)
#define PWM2CR  (*(unsigned char volatile xdata *)0xff24)
#define PWM2HLD  (*(unsigned char volatile xdata *)0xff25)
#define PWM3T1  (*(unsigned int volatile xdata *)0xff30)
#define PWM3T2  (*(unsigned int volatile xdata *)0xff32)
#define PWM3CR  (*(unsigned char volatile xdata *)0xff34)
#define PWM3HLD  (*(unsigned char volatile xdata *)0xff35)
#define PWM4T1  (*(unsigned int volatile xdata *)0xff40)
#define PWM4T2 (*(unsigned int volatile xdata *)0xff42)
#define PWM4CR (*(unsigned char volatile xdata *)0xff44)
#define PWM4HLD (*(unsigned char volatile xdata *)0xff45)
#define PWM5T1 (*(unsigned int volatile xdata *)0xff50)
#define PWM5T2 (*(unsigned int volatile xdata *)0xff52)
#define PWM5CR (*(unsigned char volatile xdata *)0xff54)
#define PWM5HLD (*(unsigned char volatile xdata *)0xff55)
#define PWM6T1 (*(unsigned int volatile xdata *)0xff60)
#define PWM6T2 (*(unsigned int volatile xdata *)0xff62)
#define PWM6CR (*(unsigned char volatile xdata *)0xff64)
#define PWM6HLD (*(unsigned char volatile xdata *)0xff65)
#define PWM7T1 (*(unsigned int volatile xdata *)0xff70)
#define PWM7T2 (*(unsigned int volatile xdata *)0xff72)
#define PWM7CR (*(unsigned char volatile xdata *)0xff74)
#define PWM7HLD (*(unsigned char volatile xdata *)0xff75)

void main()
{
    P_SW2 = 0x80;
    PWMCKS = 0x00; // The PWM clock is a system clock
    PWMCFG = 0x1000; //Set the PWM Period to 1000H PWM Clocks
    PWM0T1 = 0x0100; //Output low level at count value of 100H
    PWM0T2 = 0x0500; //Output a high level at a count value of 500H
    PWM0CR = 0x80; //Enable PWM0 output
    P_SW2 = 0x00;
    PWMCNCR = 0x80; //Start the PWM module

    while (1);
}

19.2.2 Two PWMs Complementary Symmetric Waveform with Dead-time Control

Assembly code
;The test operating frequency is 11.0592MHz

P_SW2 DATA 0BAH
PWMCFG DATA 0F1H
PWMIF DATA 0F6H
PWMFDCR DATA 0F7H
PWMCR DATA 0FEH
PWMCH XDATA 0FFF0H
PWML CL XDATA 0FFF1H
TADC(PH XDATA 0FFF2H
TADCPL XDATA 0FFF3H
PWM0T1H XDATA 0FF00H
PWM0T1L XDATA 0FF01H
PWM0T2H XDATA 0FF02H
PWM0T2L XDATA 0FF03H
PWM0CR XDATA 0FF04H
PWM0HLD XDATA 0FF05H
PWM1T1H XDATA 0FF10H
PWM1T1L XDATA 0FF11H
PWM1T2H XDATA 0FF12H
PWM1T2L XDATA 0FF13H
PWM1CR XDATA 0FF14H
PWM1HLD XDATA 0FF15H
PWM2T1H XDATA 0FF20H
PWM2T1L XDATA 0FF21H
PWM2T2H XDATA 0FF22H
PWM2T2L XDATA 0FF23H
PWM2CR XDATA 0FF24H
PWM2HLD XDATA 0FF25H
PWM3T1H XDATA 0FF30H
PWM3T1L XDATA 0FF31H
PWM3T2H XDATA 0FF32H
PWM3T2L XDATA 0FF33H
PWM3CR XDATA 0FF34H
PWM3HLD XDATA 0FF35H
PWM4T1H XDATA 0FF40H
PWM4T1L XDATA 0FF41H
PWM4T2H XDATA 0FF42H
PWM4T2L XDATA 0FF43H
PWM4CR XDATA 0FF44H
PWM4HLD XDATA 0FF45H
PWM5T1H XDATA 0FF50H
PWM5T1L XDATA 0FF51H
PWM5T2H XDATA 0FF52H
PWM5T2L XDATA 0FF53H
PWM5CR XDATA 0FF54H
PWM5HLD XDATA 0FF55H
PWM6T1H XDATA 0FF60H
PWM6T1L XDATA 0FF61H
PWM6T2H XDATA 0FF62H
PWM6T2L XDATA 0FF63H
PWM6CR XDATA 0FF64H
PWM6HLD XDATA 0FF65H
PWM7T1H XDATA 0FF70H
PWM7T1L XDATA 0FF71H
PWM7T2H XDATA 0FF72H
PWM7T2L XDATA 0FF73H
PWM7CR XDATA 0FF74H
PWM7HLD XDATA 0FF75H

ORG 0000H
LJMP MAIN

ORG 0100H

MAIN:
MOV P_SW2,#80H
CLR A
MOV DPTR,#PWMCKS
MOVX @DPTR,A ;The PWM clock is a system clock
MOV A,#08H
MOV DPTR,#PWMCH ;Set the PWM period to 0800H PWM clocks
MOVX @DPTR,A
MOV A,#00H
MOV DPTR,#PWMCL
MOVX @DPTR,A
MOV A,#01H
MOV DPTR,#PWM0T1H ;PWM0 outputs low level at count value 0100H
MOVX @DPTR,A
MOV A,#00H
MOV DPTR,#PWM0T1L
MOVX @DPTR,A
MOV A,#07H
MOV DPTR,#PWM0T2H ;PWM0 outputs high level at count value 0700H
MOVX @DPTR,A
MOV A,#00H
MOV DPTR,#PWM0T2L
MOVX @DPTR,A
MOV A,#00H
MOV DPTR,#PWM1T2H ;PWM1 outputs high level at count value 10080H
MOVX @DPTR,A
MOV A,#80H
MOV DPTR,#PWM1T2L
MOVX @DPTR,A
MOV A,#07H
MOV DPTR,#PWM1T1H ;PWM1 outputs low level at count value 10080H
MOVX @DPTR,A
MOV A,#80H
MOV DPTR,#PWM1T1L
MOVX @DPTR,A
MOV A,#080H
MOV DPTR,#PWM0CR ;Enable PWM0 output
MOVX @DPTR,A
MOV A,#80H
MOV DPTR,#PWM1CR ;Enable PWM1 output
MOVX @DPTR,A
MOV P_SW2,#00H
MOV PWMCR,#080H ;Start the PWM module
JMP $
END

C code

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

//The test operating frequency is 11.0592MHz

sfr P_SW2 = 0xba;
sfr PWMCFG = 0xf1;
sfr PWMIF = 0xf6;
sfr PWMFDCR = 0xf7;
sfr PWMCR = 0xfe;
#define PWMC  (*(unsigned int volatile xdata *)0xfff0)
#define PWMCKS  (*(unsigned char volatile xdata *)0xfff2)
#define TADCP  (*(unsigned int volatile xdata *)0xfff3)
#define PWM0T1  (*(unsigned int volatile xdata *)0xff00)
#define PWM0T2  (*(unsigned int volatile xdata *)0xff02)
#define PWM0CR  (*(unsigned char volatile xdata *)0xff04)
#define PWM0HLD  (*(unsigned char volatile xdata *)0xff05)
# define PWM1T1 (*(unsigned int volatile xdata *)0xff10)
# define PWM1T2 (*(unsigned int volatile xdata *)0xff12)
# define PWM1CR (*(unsigned char volatile xdata *)0xff14)
# define PWM1HLD (*(unsigned char volatile xdata *)0xff15)
# define PWM2T1 (*(unsigned int volatile xdata *)0xff20)
# define PWM2T2 (*(unsigned int volatile xdata *)0xff22)
# define PWM2CR (*(unsigned char volatile xdata *)0xff24)
# define PWM2HLD (*(unsigned char volatile xdata *)0xff25)
# define PWM3T1 (*(unsigned int volatile xdata *)0xff30)
# define PWM3T2 (*(unsigned int volatile xdata *)0xff32)
# define PWM3CR (*(unsigned char volatile xdata *)0xff34)
# define PWM3HLD (*(unsigned char volatile xdata *)0xff35)
# define PWM4T1 (*(unsigned int volatile xdata *)0xff40)
# define PWM4T2 (*(unsigned int volatile xdata *)0xff42)
# define PWM4CR (*(unsigned char volatile xdata *)0xff44)
# define PWM4HLD (*(unsigned char volatile xdata *)0xff45)
# define PWM5T1 (*(unsigned int volatile xdata *)0xff50)
# define PWM5T2 (*(unsigned int volatile xdata *)0xff52)
# define PWM5CR (*(unsigned char volatile xdata *)0xff54)
# define PWM5HLD (*(unsigned char volatile xdata *)0xff55)
# define PWM6T1 (*(unsigned int volatile xdata *)0xff60)
# define PWM6T2 (*(unsigned int volatile xdata *)0xff62)
# define PWM6CR (*(unsigned char volatile xdata *)0xff64)
# define PWM6HLD (*(unsigned char volatile xdata *)0xff65)
# define PWM7T1 (*(unsigned int volatile xdata *)0xff70)
# define PWM7T2 (*(unsigned int volatile xdata *)0xff72)
# define PWM7CR (*(unsigned char volatile xdata *)0xff74)
# define PWM7HLD (*(unsigned char volatile xdata *)0xff75)

void main()
{
    P_SW2 = 0x80;
P_PWMCKS = 0x00;              //The PWM clock is a system clock
PWMC = 0x0800;              //Set the PWM period to 0800H PWM clocks
PWM0T1= 0x0100;              //PWM0 outputs low level at count value 0100H
PWM0T2= 0x0700;              //PWM0 outputs high level at count value 700H
PWM1T1= 0x0080;              //PWM1 outputs high level at count value 0080H
PWM1T2= 0x0780;              //PWM1 outputs low level at count value 0780H
PWM0CR= 0x80;                //Enable PWM0 output
PWM1CR= 0x80;                //Enable PWM1 output
P_SW2 = 0x00;

    PWMCR = 0x80;              //Start the PWM module

    while (1);
}

19.2.3 PWM to achieve gradient light (breath light)

Assembly code

;The test operating frequency is 11.0592MHz

CYCLE EQU 1000H
P_SW2 DATA 0BAH
<table>
<thead>
<tr>
<th>Register Name</th>
<th>Data Type</th>
<th>Register Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWMCFG</td>
<td>DATA</td>
<td>0F1H</td>
</tr>
<tr>
<td>PWMIF</td>
<td>DATA</td>
<td>0F6H</td>
</tr>
<tr>
<td>PWMFDCR</td>
<td>DATA</td>
<td>0F7H</td>
</tr>
<tr>
<td>PWMCR</td>
<td>DATA</td>
<td>0FEH</td>
</tr>
<tr>
<td>PWMCH</td>
<td>XDATA</td>
<td>0FFF0H</td>
</tr>
<tr>
<td>PMMCL</td>
<td>XDATA</td>
<td>0FFF1H</td>
</tr>
<tr>
<td>PWMCKS</td>
<td>XDATA</td>
<td>0FFF2H</td>
</tr>
<tr>
<td>TADCPL</td>
<td>XDATA</td>
<td>0FFF3H</td>
</tr>
<tr>
<td>TADCPL</td>
<td>XDATA</td>
<td>0FFF4H</td>
</tr>
<tr>
<td>PWM0T1H</td>
<td>XDATA</td>
<td>0FF00H</td>
</tr>
<tr>
<td>PWM0T1L</td>
<td>XDATA</td>
<td>0FF01H</td>
</tr>
<tr>
<td>PWM0T2H</td>
<td>XDATA</td>
<td>0FF02H</td>
</tr>
<tr>
<td>PWM0T2L</td>
<td>XDATA</td>
<td>0FF03H</td>
</tr>
<tr>
<td>PWM0HLD</td>
<td>XDATA</td>
<td>0FF04H</td>
</tr>
<tr>
<td>PWM0LHLD</td>
<td>XDATA</td>
<td>0FF15H</td>
</tr>
<tr>
<td>PWM1T1H</td>
<td>XDATA</td>
<td>0FF20H</td>
</tr>
<tr>
<td>PWM1T1L</td>
<td>XDATA</td>
<td>0FF21H</td>
</tr>
<tr>
<td>PWM1T2H</td>
<td>XDATA</td>
<td>0FF22H</td>
</tr>
<tr>
<td>PWM1T2L</td>
<td>XDATA</td>
<td>0FF23H</td>
</tr>
<tr>
<td>PWM1HLD</td>
<td>XDATA</td>
<td>0FF25H</td>
</tr>
<tr>
<td>PWM2T1H</td>
<td>XDATA</td>
<td>0FF30H</td>
</tr>
<tr>
<td>PWM2T1L</td>
<td>XDATA</td>
<td>0FF31H</td>
</tr>
<tr>
<td>PWM2T2H</td>
<td>XDATA</td>
<td>0FF32H</td>
</tr>
<tr>
<td>PWM2T2L</td>
<td>XDATA</td>
<td>0FF33H</td>
</tr>
<tr>
<td>PWM2HLD</td>
<td>XDATA</td>
<td>0FF34H</td>
</tr>
<tr>
<td>PWM3T1H</td>
<td>XDATA</td>
<td>0FF35H</td>
</tr>
<tr>
<td>PWM3T1L</td>
<td>XDATA</td>
<td>0FF36H</td>
</tr>
<tr>
<td>PWM3T2H</td>
<td>XDATA</td>
<td>0FF37H</td>
</tr>
<tr>
<td>PWM3T2L</td>
<td>XDATA</td>
<td>0FF38H</td>
</tr>
<tr>
<td>PWM3HLD</td>
<td>XDATA</td>
<td>0FF39H</td>
</tr>
<tr>
<td>PWM4T1H</td>
<td>XDATA</td>
<td>0FF40H</td>
</tr>
<tr>
<td>PWM4T1L</td>
<td>XDATA</td>
<td>0FF41H</td>
</tr>
<tr>
<td>PWM4T2H</td>
<td>XDATA</td>
<td>0FF42H</td>
</tr>
<tr>
<td>PWM4T2L</td>
<td>XDATA</td>
<td>0FF43H</td>
</tr>
<tr>
<td>PWM4HLD</td>
<td>XDATA</td>
<td>0FF44H</td>
</tr>
<tr>
<td>PWM5T1H</td>
<td>XDATA</td>
<td>0FF45H</td>
</tr>
<tr>
<td>PWM5T1L</td>
<td>XDATA</td>
<td>0FF46H</td>
</tr>
<tr>
<td>PWM5T2H</td>
<td>XDATA</td>
<td>0FF47H</td>
</tr>
<tr>
<td>PWM5T2L</td>
<td>XDATA</td>
<td>0FF48H</td>
</tr>
<tr>
<td>PWM5HLD</td>
<td>XDATA</td>
<td>0FF49H</td>
</tr>
<tr>
<td>PWM6T1H</td>
<td>XDATA</td>
<td>0FF50H</td>
</tr>
<tr>
<td>PWM6T1L</td>
<td>XDATA</td>
<td>0FF51H</td>
</tr>
<tr>
<td>PWM6T2H</td>
<td>XDATA</td>
<td>0FF52H</td>
</tr>
<tr>
<td>PWM6T2L</td>
<td>XDATA</td>
<td>0FF53H</td>
</tr>
<tr>
<td>PWM6HLD</td>
<td>XDATA</td>
<td>0FF54H</td>
</tr>
<tr>
<td>PWM7T1H</td>
<td>XDATA</td>
<td>0FF55H</td>
</tr>
<tr>
<td>PWM7T1L</td>
<td>XDATA</td>
<td>0FF56H</td>
</tr>
<tr>
<td>PWM7T2H</td>
<td>XDATA</td>
<td>0FF57H</td>
</tr>
<tr>
<td>PWM7T2L</td>
<td>XDATA</td>
<td>0FF58H</td>
</tr>
<tr>
<td>PWM7HLD</td>
<td>XDATA</td>
<td>0FF59H</td>
</tr>
</tbody>
</table>

Nantong guoxin Microelectronics Co., Ltd.
Tel: 0513-5501 2928/2929/2966
Fax: 0513-5501 2926/2956/2947
DIR    BIT  20H.0
VALL   DATA  21H
VALH   DATA  22H

ORG  0000H
LJMP  MAIN
ORG  00B3H
LJMP  PWMISR

ORG  0100H

PWMISR:
PUSH   ACC
PUSH   PSW
PUSH   DPL
PUSH   DPH
PUSH   P_SW2

MOV    P_SW2,#80H
MOV    A,PWMCFG
JNB    ACC.7,ISREXIT
ANL    PWMCFG,#NOT 80H ;Clear interrupt flag
JNB    DIR,PWMDN

PWMUP:
MOV    A,VALL
ADD    A,#1
MOV    VALL,A
ADD     A,#0
MOV    VALH,A
CJNE    A,#HIGH CYCLE,SETPWM
MOV    A,VALL
CJNE    A,#LOW CYCLE,SETPWM
CLR    DIR
JMP    SETPWM

PWMDN:
MOV    A,VALL
ADD    A,#0FFH
MOV    VALL,A
MOV    A,VALH
ADD     A,#0FFH
MOV    VALH,A
JNZ     SETPWM
MOV    A,VALL
CJNE    A,#1,SETPWM
SETB    DIR

SETPWM:
MOV    A,VALH
MOV    DPTR,#PWM0T2H
MOVX    @DPTR,A
MOV    A,VALL
MOV    DPTR,#PWM0T2L
MOVX    @DPTR,A

ISREXIT:
POP    PSW
POP    DPH
POP    DPL
POP    P_SW2
POP    ACC
RETI

MAIN:

MOV SP,#3FH
SETB DIR
MOV VALH,#00H
MOV VALL,#01H
MOV P_SW2,#80H
CLR A
MOV DPTR,#PWMCKS
MOV A,#HIGH CYCLE
MOV DPTR,#PWMCH
MOV A,#LOW CYCLE
MOV DPTR,#PWMCL
MOVX @DPTR,A
MOV A,#00H
MOV DPTR,#PWM0T1H
MOVX @DPTR,A
MOV A,#00H
MOV DPTR,#PWM0T1L
MOVX @DPTR,A
MOV A,#VALH
MOV DPTR,#PWM0T2H
MOVX @DPTR,A
MOV A,#VALL
MOV DPTR,#PWM0T2L
MOVX @DPTR,A
MOV A,#80H
MOV DPTR,#PWM0CR
MOVX @DPTR,A
MOV P_SW2,#00H
MOV PWMCR,#0C0H
SETB EA
JMP $

END

C code

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

//The test operating frequency is 11.0592MHz

#define CYCLE 0x1000
#define PWMC (*(unsigned int volatile xdata *)0xfff0)

#define PWMC 0x1000
#define PWMC (*(unsigned int volatile xdata *)0xfff0)
#define PWMCKS (*(unsigned char volatile xdata *)0xfff2)
#define TADCP (*(unsigned int volatile xdata *)0xfff3)
#define PWM0T1 (*(unsigned int volatile xdata *)0xff00)
#define PWM0T2 (*(unsigned int volatile xdata *)0xff02)
#define PWM0CR (*(unsigned char volatile xdata *)0xfff0)
#define PWM0HLD (*(unsigned char volatile xdata *)0xfff5)
#define PWM1T1 (*(unsigned int volatile xdata *)0xfff1)
#define PWM1T2 (*(unsigned int volatile xdata *)0xfff2)
#define PWM1CR (*(unsigned char volatile xdata *)0xfff4)
#define PWM1HLD (*(unsigned char volatile xdata *)0xfff6)
#define PWM2T1 (*(unsigned int volatile xdata *)0xfff9)
#define PWM2T2 (*(unsigned int volatile xdata *)0xfff2)
#define PWM2CR (*(unsigned char volatile xdata *)0xfff7)
#define PWM2HLD (*(unsigned char volatile xdata *)0xfffa)
#define PWM3T1 (*(unsigned int volatile xdata *)0xfffb)
#define PWM3T2 (*(unsigned int volatile xdata *)0xfffd)
#define PWM3CR (*(unsigned char volatile xdata *)0xfff8)
#define PWM3HLD (*(unsigned char volatile xdata *)0xfff9)
#define PWM4T1 (*(unsigned int volatile xdata *)0xfffa)
#define PWM4T2 (*(unsigned int volatile xdata *)0xfffb)
#define PWM4CR (*(unsigned char volatile xdata *)0xfffc)
#define PWM4HLD (*(unsigned char volatile xdata *)0xfffd)
#define PWM5T1 (*(unsigned int volatile xdata *)0xfffd)
#define PWM5T2 (*(unsigned int volatile xdata *)0xfffe)
#define PWM5CR (*(unsigned char volatile xdata *)0xfff0)
#define PWM5HLD (*(unsigned char volatile xdata *)0xfff1)
#define PWM6T1 (*(unsigned int volatile xdata *)0xfff0)
#define PWM6T2 (*(unsigned int volatile xdata *)0xfff1)
#define PWM6CR (*(unsigned char volatile xdata *)0xfff2)
#define PWM6HLD (*(unsigned char volatile xdata *)0xfff3)
#define PWM7T1 (*(unsigned int volatile xdata *)0xfff3)
#define PWM7T2 (*(unsigned int volatile xdata *)0xfff4)
#define PWM7CR (*(unsigned char volatile xdata *)0xfff5)
#define PWM7HLD (*(unsigned char volatile xdata *)0xfff6)

void PWM_Isr() interrupt 22
{
    static bit dir = 1;
    static int val = 0;

    if (PWMCFG & 0x80) //Clear interrupt flag
    {
        PWMCFG &= ~0x80;
        if (dir)
        {
            val++;
            if (val >= CYCLE) dir = 0;
        }
        else
        {
            val--;
            if (val <= 1) dir = 1;
            _push_(P_SW2);
            P_SW2 |= 0x80;
            PWM0T2 = val;
            _pop_(P_SW2);
        }
    }
}
void main()
{
    P_SW2 = 0x80;
    PWMCKS = 0x00; //The PWM clock is a system clock
    PWMC = CYCLE; //Set the PWM period
    PWM0T1 = 0x0000;
    PWM0T2 = 0x0001;
    PWM0CR = 0x80; //Enable PWM0 output
    P_SW2 = 0x00;

    PWMCR = 0xc0; //Start the PWM module
    EA = 1;

    while (1);
}
20 SYNSCHRONOUS SERIAL PERIPHERAL INTERFACE (SPI)

A high-speed serial communication interface - SPI is integrated in STC8F family of microcontrollers. SPI is a full-duplex high-speed synchronous communication bus. SPI interface integrated in the STC8F family of microcontrollers offers two operation modes: master mode and slave mode.

20.1 SPI RELATED REGISTERS

**SPI Status Register**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPSTAT</td>
<td>CDH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SPIF: SPI transfer completion flag.

When SPI completes sending / receiving 1 byte of data, the hardware will automatically set this bit and request interrupt to CPU. When the SSIG bit is set to 0, this flag will also be automatically set by hardware to indicate a mode change of device when the master / slave mode of the device changes due to changes in the SS pin level.

Note: This bit must be cleared using software writing 1 to it.

WCOL: SPI write collision flag bit.

This bit is set by hardware when the SPI is writing to the SPDAT register during data transfer.

Note: This bit must be cleared using software by writing 1 to it.

**SPI Control Register**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPCTL</td>
<td>CEH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SSIG: Control bit of whether SS pin is ignored or not.

0: the SS pin decides whether the device is a master or slave.
1: the function of SS pin is ignored. MSTR decides whether the device is a master or slave.

SPEN: SPI enable bit.

0: the SPI is disabled.
1: the SPI is enabled.

DORD: Set the transmitted or received SPI data order.

0: The MSB of the data is transmitted first.
1: The LSB of the data is transmitted first.
MSTR: Master/Slave mode select bit.

To set the master mode:
If SSIG = 0, the SS pin must be high and set MSTR to 1.
If SSIG = 1, it only needs to set MSTR to 1 (ignoring the SS pin level).

To set the slave mode:
If SSIG = 0, the SS pin must be low (regardless of the MSTR bit).
If SSIG = 1, it only needs to set MSTR to 0 (ignoring the SS pin level).

CPOL: SPI clock polarity select bit.
0: SCLK is low when idle. The leading edge of SCLK is the rising edge and the trailing edge is the falling edge.
1: SCLK is high when idle. The leading edge of SCLK is the falling edge and the trailing edge is the rising edge.

CPHA: SPI clock phase select bit.
0: The first bit of datum is driven when SS pin is low. The datum changes on the trailing edge of SPICLK and is sampled on the leading edge of SPICLK. (SSIG must be 0.)
1: The datum is driven on the leading edge of SPICLK, and is sampled on the trailing edge.

SPR[1:0]: SPI clock frequency select bits

<table>
<thead>
<tr>
<th>SPR[1:0]</th>
<th>SCLK frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>SYSclick/4</td>
</tr>
<tr>
<td>01</td>
<td>SYSclick/8</td>
</tr>
<tr>
<td>10</td>
<td>SYSclick/16</td>
</tr>
<tr>
<td>11</td>
<td>SYSclick/32</td>
</tr>
</tbody>
</table>

SPI Data Register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPDAT</td>
<td>CFH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The SPDAT holds the data to be transmitted or the data received.

20.2 SPI Communication Modes

There are three SPI communication modes: single master and single slave mode, dual devices configuration mode (both can be a master or slave), single master and multiple slaves mode.

20.2.1 Single Master and Single Slave Mode

Two devices are connected, one of which is fixed as a master and the other as a slave.

Master settings: SSIG set to 1, MSTR set to 1, fixed to be master mode. The master can use any port to connect the slave SS pin, pull down the slave SS pin to enable the slave.

Slave settings: SSIG is set to 0, SS pin as the chip select signal of the slave.

Single master single slave connection configuration diagram is shown as follows:
### 20.2.2 Dual Devices Configuration Mode

Two devices are connected, the master and the slave are not fixed.

**Setting Method 1:** Both devices are initialized with SSIG set to 0, MSTR set to 1, and SS pin set to bi-directional mode and output high. Now the both devices are in master mode with not ignoring SS. When one of the devices needs to initiate a transfer, set its own SS pin to output mode and output low to pull down the other device's SS pin so that the other device is forcibly set to slave mode.

**Setting Method 2:** Both devices are initialized as slave mode with ignoring SS, where SIG is set to 1 and MSTR is set to 0. When one of the devices needs to initiate a transfer, detect the SS pin's level firstly. If SS is high, the device sets itself to master mode with ignoring SS, then starts the data transfer.

The connection configuration of dual devices configuration mode is shown as follows:

---

### 20.2.3 Single Master and Multiple Slaves Mode

Multiple devices are connected, one of which is fixed as a master and others are fixed as slaves.

**Master settings:** SSIG set to 1, MSTR set to 1, fixed to master mode. The master can use any port to connect with the SS pins of each slave respectively, and pull down the SS pin of one slave to enable the corresponding slave device.

**Slave settings:** SSIG is set to 0, SS pin is used as the chip select signal of the slave.

The configuration diagram of single master multiple slaves is as follows:
20.3 SPI Configuration

<table>
<thead>
<tr>
<th>Control bits</th>
<th>Communication port pins</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPEN SSIG MSTR S S MISO MOSI SCLK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 x x x</td>
<td>input input input</td>
<td>SPI is disabled, SS/MOSI/MISO/SCLK are used as general I/O ports</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>output input input</td>
<td><strong>Selected as slave</strong></td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>High impedance input input</td>
<td><strong>Selected as slave</strong>, not selected.</td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>output input input</td>
<td><strong>Slave mode</strong>, master mode with not ignoring SS and MSTR is 1. When SS pin is pulled low, MSTR will be automatically cleared by hardware and the operating mode will be passively set to slave mode.</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>input output output</td>
<td><strong>Master mode, idle state</strong></td>
</tr>
<tr>
<td>1 1 0 x</td>
<td>output output</td>
<td><strong>Master mode, active state</strong></td>
</tr>
<tr>
<td>1 1 1 x</td>
<td>input output output</td>
<td><strong>Master mode</strong></td>
</tr>
</tbody>
</table>

**Additional Considerations for a Slave**

When CPHA = 0, SSIG must be 0 (ie SS pin can not be ignored). The SS pin must be pulled low before each serial byte begins transfer and must be reset to high after the transfer has completed. The SPDAT register can not be written while the SS pin is low, otherwise a write collision error will occurs. Operation with CPHA = 0 and SSIG = 1 is undefined.

When CPHA = 1, SSIG may be set to 1 (ie, the SS pin can be ignored). If SSIG = 0, the SS pin may remain active low (ie, stay low all the way) for consecutive transfers. This method is suitable for fixed single
master single slave system.

**Additional Considerations for a Master**

In SPI, transfers are always initiated by the master. If the SPI is enabled (SPEN = 1) and selected as the master, the master will initiate a SPI clock generator and data transfer by writing to the SPI data register, SPDAT. The data will appear on the MOSI pin a half to one SPI bit-time later after the data is written to SPDAT. The data written to the SPDAT register of the master is shifted out from the MOSI pin and sent to the MOSI pin of the slave. And, at the same time the data in SPDAT register of the selected slave is shifted out on MISO pin to the MISO pin of the master.

After one byte has been transmitted, the SPI clock generator is stopped, the transfer completion flag (SPIF) is set, and an SPI interrupt is generated if the SPI interrupt is enabled. The two shift registers for the master and slave CPUs can be considered as a 16-bit cyclic shift register. As data is shifted from the master to the slave, data is also shifted in the opposite direction simultaneously. This means that the data of the master and the slave are exchanged with each other in one shift cycle.

**Mode Change on SS pin**

If SPEN = 1, SSIG = 0 and MSTR = 1, the SPI is enabled in master mode and the SS pin can be configured for input mode or quasi-bidirectional port mode. In this case, another master can drive this pin low to select the device as an SPI slave and send data to it. To avoid bus contention, the SPI system clears the slave's MSTR, forces MOSI and SCLK to be input mode, and MISO changes to output mode. The SPIF flag in SPSTAT is set, and if the SPI interrupt is enabled, an SPI interrupt will occur.

The user software must always detect the MSTR bit. If this bit is cleared by a slave selection action and the user wants to continue using the SPI as a master, the MSTR bit must be set again, otherwise it will remain in slave mode.

**Write Collision**

The SPI is single buffered in the transmit direction and double buffered in the receive direction. New data for transmission can not be written to the shift register until the previous transmission is complete. The WCOL bit will be set to indicate that a data write collision error has occurred when the data register SPDAT is written during transmission. In this case, the data currently being transmitted will continue to be transmitted, and the newly written data will be lost.

A write collision condition on the master is rare when write collision detection is performed on a master or slave because the master has full control of the data transfer. However, a write collision may occur on the slave because the slave can not control it when the master initiates the transfer.

When receiving data, the received data is transferred to a parallel read data buffer, which will release the shift register for the next data reception. However, the received data must be read from the data register before the next character is completely shifted in. Otherwise, the previous received data will be lost.

WCOL can be cleared by software by writing "1" to it.
20.4 Data Pattern

The clock phase control bit, CPHA, of the SPI allows the user to set the clock edge when the data is sampled and changed. The clock polarity bit CPOL allows the user to set the clock polarity. The following illustrations show the SPI communication timing under different clock phases and polarity settings.

SCLK (CPOL=0)  SCLK (CPOL=1)

Sample at the leading edge  Sample at the trailing edge

MOSI (input)  MISO (output)

DORD=0  DORD=1

MSB  LSB  MSB  LSB

1  6  5  4  3  2  1  6

1  6  5  4  3  2  1  6

The falling edge triggers the slave to send the 1st bit of data  Invalid data

SS (SSIG=0)

The falling edge triggers the slave to send the 1st bit of data  Invalid data

SPI slave transfer format with CPHA=0

SCLK (CPOL=0)  SCLK (CPOL=1)

Sample at the trailing edge  Sample at the leading edge

MOSI (input)  MISO (output)

DORD=0  DORD=1

MSB  LSB  MSB  LSB

1  6  5  4  3  2  1  6

1  6  5  4  3  2  1  6

Invalid data  Sample at the trailing edge  Drive at the leading edge

SS (SSIG=0)

Invalid data

SPI slave transfer format with CPHA=1

SCLK (CPOL=0)  SCLK (CPOL=1)

Change data at the trailing edge  Change data at the trailing edge

MOSI (output)  MISO (input)

DORD=0  DORD=1

MSB  LSB  MSB  LSB

1  6  5  4  3  2  1  6

1  6  5  4  3  2  1  6

Sample at the leading edge  Sample at the leading edge

SS (SSIG=0)

Invalid data

SPI master transfer format with CPHA=0
20.5 Sample program

20.5.1 SPI Single Master and Single Slave System host

program(interrupt mode)

Assembly code

<table>
<thead>
<tr>
<th></th>
<th>DATA</th>
<th>0CDH</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPSTAT</td>
<td>DATA</td>
<td>0CEH</td>
</tr>
<tr>
<td>SPCTL</td>
<td>DATA</td>
<td>0CFH</td>
</tr>
<tr>
<td>SPDAT</td>
<td>DATA</td>
<td>0AFH</td>
</tr>
<tr>
<td>IE2</td>
<td>EQU</td>
<td>02H</td>
</tr>
</tbody>
</table>

| BUSY  | BIT     | 20H.0  |
| SS    | BIT     | P1.0   |
| LED   | BIT     | P1.1   |

ORG 0000H
LJMP MAIN
ORG 004BH
LJMP SPIISR

OR 0100H

SPIISR:

MOV SPSTAT, #0C0H ;Clear interrupt flag
SETB SS ;pull up SS pin of slave
CLR BUSY
CPL LED
RETI

MAIN:

MOV SP,#3FH

SETB LED
SETB SS
CLR BUSY

MOV SPCTL,#50H ;Enable SPI master mode
MOV SPSTAT, #0C0H ;Clear interrupt flag
MOV IE2,#ESPI ;Enable SPI interrupt

Sample program

20.5.1 SPI Single Master and Single Slave System host

program(interrupt mode)
SETB EA

LOOP:
    JB BUSY,S
    SETB BUSY
    CLR SS ;Pull down SS pin of slave
    MOV SPDAT,#5AH ;Send test data
    JMP LOOP
END

C code
#include "reg51.h"
#include "intrins.h"
sfr SPSTAT = 0xcd;
sfr SPCTL = 0xce;
sfr SPDAT = 0xcf;
sfr IE2 = 0xaf;
#define ESPI 0x02
sbit SS = P1^0;
sbit LED = P1^1;
bit busy;

void SPI_Isr() interrupt 9
{
    SPSTAT = 0xc0; //Clear interrupt flag
    SS = 1; //pull up SS pin of slave
    busy = 0;
    LED = !LED; //Test port
}

void main()
{
    LED = 1;
    SS = 1;
    busy = 0;
    SPCTL = 0x50; //Enable SPI master mode
    SPSTAT = 0xc0; //Clear interrupt flag
    IE2 = ESPI; //Enable SPI interrupt
    EA = 1;
    while (1)
    {
        while (busy);
        busy = 1;
        SS = 0; //Pull down SS pin of slave
        SPDAT = 0x5a; //Send test data
    }
}
20.5.2 SPI Single Master and Single Slave System slave program (Single Master and Single Slave)

**Assembly code**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPSTAT</td>
<td>DATA</td>
<td>0CDH</td>
</tr>
<tr>
<td>SPCTL</td>
<td>DATA</td>
<td>0CEH</td>
</tr>
<tr>
<td>SPDAT</td>
<td>DATA</td>
<td>0CFH</td>
</tr>
<tr>
<td>IE2</td>
<td>DATA</td>
<td>0AFH</td>
</tr>
<tr>
<td>ESPI</td>
<td>EQU</td>
<td>02H</td>
</tr>
<tr>
<td>LED</td>
<td>BIT</td>
<td>P1.1</td>
</tr>
</tbody>
</table>

ORG 0000H
LJMP MAIN
ORG 004BH
LJMP SPIISR
ORG 0100H

**SPIISR:**

```asm
MOV SPSTAT,#0C0H ;Clear interrupt flag
MOV SPDAT,SPDAT ;Sending the received data back to the host
CPL LED
RETI
```

**MAIN:**

```asm
MOV SP,#3FH
MOV SPCTL,#40H ;Enable SPI slave mode
MOV SPSTAT,#0C0H ;Clear interrupt flag
MOV IE2,#ESPI ;Enable SPI interrupt
SETB EA
JMP $ END
```

**C code**

```c
#include "reg51.h"
#include "intrins.h"
sfr SPSTAT = 0xcd;
sfr SPCTL = 0xee;
sfr SPDAT = 0xcf;
sfr IE2 = 0xaf;
#define ESPI 0x02
sbit LED = P1^1;

void SPI_Isr() interrupt 9 {
    SPSTAT = 0x00; //Clear interrupt flag
    SPDAT = SPDAT; //Sending the received data back to the host
    LED = !LED; //Test port
}
```
void main()
{
    SPCTL = 0x40; //Enable SPI slave mode
    SPSTAT = 0xc0; //Clear interrupt flag
    IE2 = ESPI; //Enable SPI interrupt
    EA = 1;

    while (1);
}

20.5.3 SPI Single Master and Single Slave System host program(Query mode)

Assembly code

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPSTAT DATA 0CDH</td>
<td></td>
</tr>
<tr>
<td>SPCTL DATA 0CEH</td>
<td></td>
</tr>
<tr>
<td>SPDAT DATA 0CFH</td>
<td></td>
</tr>
<tr>
<td>IE2 DATA 0AFH</td>
<td></td>
</tr>
<tr>
<td>ESPI EQU 02H</td>
<td></td>
</tr>
<tr>
<td>SS BIT P1.0</td>
<td></td>
</tr>
<tr>
<td>LED BIT P1.1</td>
<td></td>
</tr>
<tr>
<td>ORG 0000H</td>
<td></td>
</tr>
<tr>
<td>LJMP MAIN</td>
<td></td>
</tr>
<tr>
<td>ORG 0100H</td>
<td></td>
</tr>
</tbody>
</table>

**MAIN:**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV SP,#3FH</td>
<td></td>
</tr>
<tr>
<td>SETB LED</td>
<td></td>
</tr>
<tr>
<td>SETB SS</td>
<td></td>
</tr>
<tr>
<td>MOV SPCTL,#50H</td>
<td>Enable SPI master mode</td>
</tr>
<tr>
<td>MOV SPSTAT,#0C0H</td>
<td>Clear interrupt flag</td>
</tr>
<tr>
<td>LOOP:</td>
<td></td>
</tr>
<tr>
<td>CLR SS</td>
<td>Pull down SS pin of slave</td>
</tr>
<tr>
<td>MOV SPDAT,#5AH</td>
<td>Send test data</td>
</tr>
<tr>
<td>MOV ASPSTAT</td>
<td>Query completion flag</td>
</tr>
<tr>
<td>JNB ACC.7,5-2</td>
<td></td>
</tr>
<tr>
<td>MOV SPSTAT,#0C0H</td>
<td>Clear interrupt flag</td>
</tr>
<tr>
<td>SETB SS</td>
<td></td>
</tr>
<tr>
<td>CPL LED</td>
<td></td>
</tr>
<tr>
<td>JMP LOOP</td>
<td></td>
</tr>
<tr>
<td>END</td>
<td></td>
</tr>
</tbody>
</table>

C code

```c
#include "reg51.h"
#include "intrins.h"
```
sfr SPSTAT = 0xcd;
sfr SPCTL = 0xce;
sfr SPDAT = 0xcf;
sfr IE2 = 0xaf;
#define ESPI 0x02
sbit SS = P1^0;
sbit LED = P1^1;

void main()
{
    LED = 1;
    SS = 1;

    SPCTL = 0x50; //Enable SPI master mode
    SPSTAT = 0xc0; //Clear interrupt flag

    while (1)
    {
        SS = 0; //Pull down SS pin of slave
        SPDAT = 0x5a; //Send test data
        while (!(SPSTAT & 0x80)); //Query completion flag
        SPSTAT = 0xc0; //Clear interrupt flag
        SS = 1; //pull up SS pin of slave
        LED = !LED; //Test port
    }
}

20.5.4 SPI Single Master and Single Slave System host program (Query mode)

<table>
<thead>
<tr>
<th>Assembly code</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPSTAT DATA 0CDH</td>
</tr>
<tr>
<td>SPCTL DATA 0CEH</td>
</tr>
<tr>
<td>SPDAT DATA 0CFH</td>
</tr>
<tr>
<td>IE2 DATA 0AFH</td>
</tr>
<tr>
<td>ESPI EQU 02H</td>
</tr>
<tr>
<td>LED BIT P1.1</td>
</tr>
<tr>
<td>ORG 0000H</td>
</tr>
<tr>
<td>LJMP MAIN</td>
</tr>
<tr>
<td>ORG 0100H</td>
</tr>
<tr>
<td>MAIN:</td>
</tr>
<tr>
<td>MOV SP,#3FH</td>
</tr>
<tr>
<td>MOV SPCTL,#40H ;Enable SPI slave mode</td>
</tr>
<tr>
<td>MOV SPSTAT,#0C0H ;Clear interrupt flag</td>
</tr>
<tr>
<td>LOOP:</td>
</tr>
<tr>
<td>MOV A,SPSTAT ;Query completion flag</td>
</tr>
<tr>
<td>JNB ACC.7-2</td>
</tr>
<tr>
<td>MOV SPSTAT,#0C0H ;Clear interrupt flag</td>
</tr>
</tbody>
</table>
MOV SPDAT,SPDAT ;Sending the received data back to the host
CPL LED
JMP LOOP
END

C code
#include "reg51.h"
#include "intrins.h"
sfr SPSTAT = 0xcd;
sfr SPCTL = 0xce;
sfr SPDAT = 0xcf;
sfr IE2 = 0xaf;
#define ESPI 0x02
sbit LED = P1^1;

void SPI_Isr() interrupt 9
{
    SPSTAT = 0xc0; //Clear interrupt flag
}

void main()
{
    SPCTL = 0x40; //Enable SPI slave mode
    SPSTAT = 0xc0; //Clear interrupt flag
    while (1)
    {
        while (!(SPSTAT & 0x80)); //Query completion flag
        SPSTAT = 0xc0; //Clear interrupt flag
        SPDAT = SPDAT; //Sending the received data back to the host
        LED = !LED; //Test port
    }
}

20.5.5 SPI Mutual master-slave system program(interrupt mode)

Assembly code

| SPSTAT | DATA | 0CDH |
| SPCTL  | DATA | 0CEH |
| SPDAT  | DATA | 0CFH |
| IE2    | DATA | 0AFH |
| ESPI   | EQU  | 02H  |
| SS     | BIT  | P1.0 |
| LED    | BIT  | P1.1 |
| KEY    | BIT  | P0.0 |
| ORG    | 0000H |
| LJMP   | MAIN |
ORG 004BH
LJMP SPIISR

ORG 0100H

SPIISR:
PUSH ACC
MOV SPSTAT,#0C0H ;Clear interrupt flag
MOV A,SPCTL
JB ACC.4,MASTER

SLAVE:
MOV SPDAT,SPDAT ;Sending the received data back to the host
JMP ISREXIT

MASTER:
SETB SS ;Pull up the SS pin of the slave
MOV SPCTL,#40H ;Reset to slave standby

ISREXIT:
CPL LED
POP ACC
RETI

MAIN:
MOV SP,#3FH
SETB SS
SETB LED
SETB KEY

MOV SPCTL,#40H ;Enable SPI slave mode to standby
MOV SPSTAT,#0C0H ;Clear interrupt flag
MOV IE2,#ESPI ;Enable SPI interrupt
SETB EA

LOOP:
JB KEY,LOOP ;Waiting key trigger
MOV SPCTL,#50H ;Enable SPI host mode
CLR SS ;Pull down the SS pin of slave
MOV SPDAT,#5AH ;Send test data
JNB KEY,$ ;Waiting key releases
JMP LOOP

END

C code
#include "reg51.h"
#include "intrins.h"
sfr SPSTAT = 0xcd; 
sfr SPCTL = 0xce; 
sfr SPDAT = 0xcf; 
sfr IE2 = 0xaf; 
#define ESPI 0xc02

sbit SS = P1^0; 
sbit LED = P1^1; 
sbit KEY = P0^0;

void SPI_Isr() interrupt 9
```c
{
    SPSTAT = 0xc0;  //Clear interrupt flag
    if (SPCTL & 0x10) //Master mode
    {
        SS = 1;       //Pull up the SS pin of the slave
        SPCTL = 0x40;  //Reset to slave standby
    }
    else //Slave mode
    {
        SPDAT = SPDAT; //Sending the received data back to the host
    }
    LED = !LED;    //Test port
}

void main()
{
    LED = 1;
    KEY = 1;
    SS = 1;

    SPCTL = 0x40;    //Enable SPI slave mode to standby
    SPSTAT = 0xc0;   //Clear interrupt flag
    IE2 = ESPI;      //Enable SPI interrupt
    EA = 1;

    while (1)
    {
        if (!KEY) //Waiting key trigger
        {
            SPCTL = 0x50;  //Enable SPI host mode
            SS = 0;        //pull down the SS pin of slave
            SPDAT = 0x5a;  //Send test data
            while (!KEY);  //waiting key releases
        }
    }
}
```

### 20.5.6 SPI Mutual master-slave system program (Query mode)

**Assembly code**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPSTAT</td>
<td>DATA</td>
<td>0CDH</td>
</tr>
<tr>
<td>SPCTL</td>
<td>DATA</td>
<td>0CEH</td>
</tr>
<tr>
<td>SPDAT</td>
<td>DATA</td>
<td>0CFH</td>
</tr>
<tr>
<td>IE2</td>
<td>DATA</td>
<td>0AFH</td>
</tr>
<tr>
<td>ESPI</td>
<td>EQU</td>
<td>02H</td>
</tr>
<tr>
<td>SS</td>
<td>BIT</td>
<td>P1.0</td>
</tr>
<tr>
<td>LED</td>
<td>BIT</td>
<td>P1.1</td>
</tr>
<tr>
<td>KEY</td>
<td>BIT</td>
<td>P0.0</td>
</tr>
<tr>
<td>ORG</td>
<td></td>
<td>0000H</td>
</tr>
<tr>
<td>LJMP</td>
<td></td>
<td>MAIN</td>
</tr>
</tbody>
</table>
ORG 0100H

MAIN:

MOV SP,#3FH

SETB SS
SETB LED
SETB KEY

MOV SPCTL,#40H
    ;Enable SPI slave mode to standby
MOV SPSTAT,#0C0H
    ;Clear interrupt flag

LOOP:

J B KEY,SKIP
    ;Waiting key trigger
MOV SPCTL,#50H
    ;Enable SPI host mode
CLR SS
    ;pull down the SS pin of slave
MOV SPDAT,#5AH
    ;Send test data
JNB KEY,$
    ;waiting key releases

SKIP:

MOV A,SPSTAT
JNB ACC.7,LOOP

MOV A,SPCTL
JB ACC.4,MASTER

SLAVE:

MOV SPDAT,SPDAT
    ;Sending the received data back to the host
CPL LED
JMP LOOP

MASTER:

SETB SS
    ;Pull up the SS pin of the slave
MOV SPCTL,#40H
    ;Reset to slave standby
CPL LED
JMP LOOP

END

C code

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

sfr SPSTAT = 0xcd;
sfr SPCTL = 0xce;
sfr SPDAT = 0xcf;
sfr IE2 = 0xaf;
#define ESPI 0x02
sbit SS = P1^0;
sbit LED = P1^1;
sbit KEY = P0^0;

void main()
{
    LED = 1;
    KEY = 1;
    SS = 1;

    SPCTL = 0x40;
    //Enable SPI slave mode to standby
    SPSTAT = 0x00;
    //Clear interrupt flag
while (1)
{
    if (!KEY) // Waiting key trigger
    {
        SPCTL = 0x50; // Enable SPI host mode
        SS = 0; // pull down the SS pin of slave
        SPDAT = 0x5a; // Send test data
        while (!KEY); // waiting key releases
    }
    if (SPSTAT & 0x80)
    {
        SPSTAT = 0xc0; // Clear interrupt flag
        if (SPCTL & 0x10) // Host mode
        {
            SS = 1; // Pull up the SS pin of the slave
            SPCTL = 0x40; // Reset to slave standby
        }
        else // Slave mode
        {
            SPDAT = SPDAT; // Sending the received data back to the host
        }
        LED = !LED; // Test port
    }
}
21 I²C Bus

An I²C serial bus controller is integrated in the STC8F family of microcontrollers. I²C is a high-speed synchronous communication bus, which uses SCL (clock line) and SDA (data line) to carry out two-wire synchronous communication. For port allocation of SCL and SDA, the STC8F family of microcontrollers provide a pin switchover mode that switches SCL and SDA to different I/O ports. So it is convenience to use a set of I²C as multiple sets of I²C buses through time sharing.

Compared with the standard I²C protocol, the following two mechanisms are ignored:
- No arbitration will be performed after the start signal (START) is sent.
- No timeout detection when the clock signal (SCL) stays at low level.

The I²C bus of the STC8F family microcontrollers offers two modes of operation: master mode (SCL is the output port, which is used to transmit synchronous clock signal) and slave mode (SCL is the input port, which is used to receive the synchronous clock signal).

21.1 I²C Related Registers

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Address</th>
<th>Bit Address and Symbol</th>
<th>Value after reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2CCFG</td>
<td>I²C configuration register</td>
<td>FE80H</td>
<td>ENI2C, MSSL, MSSPEED[6:1]</td>
<td>0000,0000</td>
</tr>
<tr>
<td>I2CMSCR</td>
<td>I²C Master Control Register</td>
<td>FE81H</td>
<td>EMSI, -,-,-,-, MSCMD[3:0]</td>
<td>0xxx,0000</td>
</tr>
<tr>
<td>I2CMSST</td>
<td>I²C Master Status Register</td>
<td>FE82H</td>
<td>MSBUSY, MSIF, -,-,-,-, MSACKI, MSACKO</td>
<td>0xxx,0000</td>
</tr>
<tr>
<td>I2CSLCR</td>
<td>I²C Slave Control Register</td>
<td>FE83H</td>
<td>ESTAI, ERXI, ETXI, ESTOI, -,-, SLRST</td>
<td>x000,0x00</td>
</tr>
<tr>
<td>I2CSLST</td>
<td>I²C Slave Status Register</td>
<td>FE84H</td>
<td>SLBUSY, STAIF, RXIF, TXIF, STOIF, TXING, SLACKI, SLACKO</td>
<td>0000,0000</td>
</tr>
<tr>
<td>I2CSLADR</td>
<td>I²C Slave Address Register</td>
<td>FE85H</td>
<td>SLADR[6:0]</td>
<td>MA 0000,0000</td>
</tr>
<tr>
<td>I2CTXD</td>
<td>I²C Data transmission register</td>
<td>FE86H</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>I2CRXD</td>
<td>I²C Data receive register</td>
<td>FE87H</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>I2CMSAUX</td>
<td>I²C Data receive register</td>
<td>FE88H</td>
<td></td>
<td>WDTA xxxx,xxxx</td>
</tr>
</tbody>
</table>

21.2 I²C Master Mode

I²C configuration register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2CCFG</td>
<td>FE80H</td>
<td>ENI2C</td>
<td>MSSL</td>
<td>MSSPEED[6:1]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ENI2C: I²C function enable bit
  0: disable I²C function
  1: enable I²C function
MSSL: I²C mode selection bit
  0: Slave mode
  1: Master mode
MSSPEED[6:1]: I²C bus speed control bits (clocks to wait)
MSSPEED[6:1] | Corresponding clocks
---|---
0 | 1
1 | 3
2 | 5
... | ...
x | 2x+1
... | ...
62 | 125
63 | 127

The waiting parameter set by the MSSPEED is valid only when the I2C module is operating in the master mode. The waiting parameter is mainly used for the following signals in master mode:

- **T_{SSTA}**: Setup Time of START
- **T_{HSTA}**: Hold Time of START
- **T_{SSTO}**: Setup Time of STOP
- **T_{HSTO}**: Hold Time of STOP
- **T_{HCKL}**: Hold Time of SCL Low

**Note:**

Due to the need to cooperate with the clock synchronization mechanism, the high-level hold time (THCKH) of the clock signal should be at least twice as long as the low-level hold time (THCKL) of the clock signal, and the exact length of THCKH depends on the pull-up speed of the SCL port.

The data retention time of SDA is fixed as 1 clock after the falling edge of SCL.

---

**I^2^C master control register**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2CMSCR</td>
<td>FE81H</td>
<td>EMSI</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>MSCMD[2:0]</td>
</tr>
</tbody>
</table>

- **EMSI**: Master mode interrupt enable control bit
  - 0: disable master mode interrupt
  - 1: enable master mode interrupt
- **MSCMD[2:0]**: master command bits
  - 000: Standby, no action
  - 001: START command

Send a START signal. If the I2C controller is in idle state currently, i.e. MSBUSY (I2CMSST.7) is 0, writing this command will make the controller enter the busy status, and the hardware will automatically set the MSBUSY status bit and start sending START signal. If the I2C controller is busy currently, writing this command is invalid. Sending the START signal waveform is shown...
010: Sending data command.

After this command is written, the I2C bus controller generates eight clocks on the SCL pin and sends the datum in the I2CTXD register to the SDA pin bit by bit (MSB first). Sending data waveform is shown below:

```
SCL  
SDA (output)
```

011: Receiving ACK command

After this command is written, the I2C bus controller generates a clock on the SCL pin and saves the datum read from the SDA port to MSACKI (I2CMSST.1). The waveform of receiving ACK is shown below:

```
SCL  
SDA (input)  ACK
```

100: Receiving data command

After this command is written, the I2C bus controller generates eight clocks on the SCL pin and shifts left the datum read from the SDA port to the I2CRXD register (MSB first). The waveform of receiving data is shown below:

```
SCL  
SDA (input)
```

101: Sending ACK command

After writing this command, the I2C bus controller generates a clock on the SCL pin and sends the datum in MSACKO (I2CMSST.0) to SDA. Sending ACK waveform is shown below:

```
SCL  
SDA (output)  ACK
```

110: STOP command

Send STOP signal. After writing this command, the I2C bus controller begins to send a STOP signal. After the signal is sent, the MSBUSY status bit will be cleared by the hardware automatically. STOP signal waveform is shown below:

```
SCL  
SDA (output)
```
0111:   Reserved.
1000:   Reserved.

注:  The following new expanded combination commands are only valid for STC8F2K64S4 Series C/D chips, STC8A8K64S4A12 Series E/F chips, STC8F2K64S2 Series C/D, and STC8A4K64S2A12 Series E/F chips.

1001:   Start command + send data command + receive ACK command.
   This command is a combination of command 0001, command 0010, and command 0011. After the command is executed, the controller executes these three commands in sequence.

1010:   send data command + receive ACK command.
   This command is the combination of the command 0010 and the command 0011. After the command is executed, the controller will execute the two commands in sequence.

1011:   Receive data command + send ACK(0) command.
   This command is a combination of the command 0100 and the command 0101. After the command is executed, the controller executes the two commands in sequence.

   Note: The response signal returned by this command is fixed as ACK(0), which is not affected by the MSACKO bit.

1100:   Receive data command + send NAK(1) command.
   This command is a combination of the command 0100 and the command 0101. After the command is executed, the controller executes the two commands in sequence.

   Note: The response signal returned by this command is fixed as ACK(1), which is not affected by the MSACKO bit.

### I²C master auxiliary control register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2CMSAUX</td>
<td>FE88H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>WDTA</td>
</tr>
</tbody>
</table>

WDTA: I²C data automatic send enable bit in master module

0:   Disable automatic sending
1:   Enable automatic sending

If the automatic transmission function is enabled, when the MCU performs a write operation to the I²C TXD data register, the I²C controller will automatically trigger the "1010" command, i.e., automatically send data and receive an ACK signal.

### I²C master status register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2CMSST</td>
<td>FE82H</td>
<td>MSBUSY</td>
<td>MSIF</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>MSACKI</td>
</tr>
</tbody>
</table>

MSBUSY: status bit of I²C controller in master mode. (Read-only)

0:   the controller is in idle state.
1:   the controller is in busy state.

When the I²C controller is in master mode, the controller will enter the busy state after sending the START signal in the idle state. The busy state will be maintained until the STOP signal is successfully transmitted, and the state will be restored to the idle state.

MSIF: master mode interrupt request bit (interrupt flag bit). After the interrupt signal is generated by the I²C controller in master mode, the hardware will automatically set this bit to 1 and request interrupt to CPU. After the interrupt is serviced, the MSIF bit must be cleared by software.
MSACKI: In master mode, it is the ACK datum received after sending the "011" command to the MSCMD bit in I2CMSCR.

MSACKO: In master mode, it is the ACK signal ready to be transmitted. When the "101" command is sent to the MSCMD bit of I2CMSCR, the controller will automatically read the datum of this bit and send it as ACK to SDA.

21.3 I^2C Slave Mode

I^2C slave control register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2CSLCR</td>
<td>FE83H</td>
<td>-</td>
<td>ESTAI</td>
<td>ERXI</td>
<td>ETXI</td>
<td>ESTOI</td>
<td>-</td>
<td>-</td>
<td>SLRCT</td>
</tr>
</tbody>
</table>

ESTAI: interrupt enable bit when receiving START signal in slave mode.
0: disable interrupt when receiving START signal in slave mode.
1: enable interrupt when receiving START signal in slave mode.

ERXI: Interrupt enable bit after 1 byte datum is received in Slave mode
0: disable interrupt after a datum is received in slave mode.
1: enable interrupt after 1 byte datum is received in slave mode.

ERXO: Interrupt enable bit after 1 byte datum is sent in Slave mode
0: disable interrupt after a datum is sent in slave mode.
1: enable interrupt after 1 byte datum is sent in slave mode.

ESTOI: interrupt enable bit after STOP signal is received in slave mode.
0: disable interrupt after STOP signal is received in slave mode.
1: enable interrupt after STOP signal is received in slave mode.

SLRCT: reset slave mode

I^2C slave status register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2CSLST</td>
<td>FE84H</td>
<td>SLBUSY</td>
<td>STAIF</td>
<td>RXIF</td>
<td>TXIF</td>
<td>STOIF</td>
<td>-</td>
<td>SLACKI</td>
<td>SLACKO</td>
</tr>
</tbody>
</table>

SLBUSY: status bit of I2C controller in slave mode. (Read-only)
0: the controller is in idle state.
1: the controller is in busy state.

When the I2C controller is in slave mode, the controller will continue to detect the subsequent device address data when it receives the START signal from the master in idle state. If the device address matches the slave address set in the current I2CSLADR register, the controller will enter the busy state. The busy state will be maintained until receives a STOP signal sent by the master successfully, and then the state will be restored to idle state.

STAIF: Interrupt request bit after START signal is received in slave mode. After the I2C controller in slave mode receives the START signal, the hardware will automatically set this bit and request interrupt to CPU. The STAIF bit must be cleared by software after the interrupt is responded. The time point of STAIF being set is shown below:
RXIF: Interrupt request bit after 1-byte datum is received in slave mode. After the I2C controller in slave mode receives a 1-byte datum, the hardware will automatically set this bit at the falling edge of the 8th clock and will request interrupt to CPU. The RXIF bit must be cleared by software after the interrupt is responded. The time point of RXIF being set is shown in the figure below:

TXIF: Interrupt request bit after 1-byte datum transmission is completed in slave mode. After the I2C controller in slave mode completes sending 1 byte of datum successfully and receives a 1-bit ACK signal, the hardware will automatically set this bit at the falling edge of the ninth clock and request an interrupt to CPU. TXIF bit must be cleared by software after the interrupt is responded. The time point of TXIF being set is shown below:

STOIF: Interrupt request bit after STOP signal is received in slave mode. After the I2C controller in slave mode receives the STOP signal, the hardware will automatically set this bit and request interrupt to CPU. The STOIF bit must be cleared by software after the interrupt is serviced. The time point of STOIF being set is shown below:

SLACKI: ACK data received in slave mode.
SLACKO: the ACK signal ready to send out in slave mode.
### I^2^C slave address register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2CSLADR</td>
<td>FE85H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SLADR[6:0]</td>
</tr>
</tbody>
</table>

**SLADR[6:0]: the slave device address**

When the I^2^C controller is in slave mode, the controller will continue to detect the device address and read/write signals sent by the master after it receives the START signal. If the device address sent by the master matches the slave device address set in SLADR [6: 0], the controller requests an interrupt to CPU requesting the CPU to process the I^2^C event. Otherwise, if the device address does not match, the I^2^C The controller continues to monitor, wait for the next START signal, and match the next device address.

**MA: Slave device address matching control bit**

- 0: The device address must continue to match SLADR [6: 0].
- 1: Ignore the settings in SLADR and match all device addresses.

### I^2^C data register

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2CTXD</td>
<td>FE86H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I2CRXD</td>
<td>FE87H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**I2CTXD** is the I^2^C transmit data register that holds the I^2^C data to be transmitted

**I2CRXD** is the I^2^C receive data register that holds the I^2^C data received.

---

## 21.4 Sample program

### 21.4.1 I^2^C master mode access AT24C256(interrupt mode)

**Assembly code**

```
.P_SW2   DATA    0BAH
.I2CCFG  XDATA   0FE80H
.I2CMSCR XDATA   0FE81H
.I2CMSST XDATA   0FE82H
.I2CSLCR XDATA   0FE83H
.I2CSLST XDATA   0FE84H
.I2CSLADR XDATA   0FE85H
.I2CTXD  XDATA   0FE86H
.I2CRXD  XDATA   0FE87H

SDA      BIT     P1.4
SCL      BIT     P1.5
```
BUSY BIT 20H.0
ORG 0000H
LJMP MAIN
ORG 00C3H
LJMP I2CISR
ORG 0100H

I2CISR:
PUSH ACC
PUSH DPL
PUSH DPH
MOV DPTR,#I2CMSST ;Clear interrupt flag
MOVX A,@DPTR
ANL A,#NOT 40H
MOV DPTR,#I2CMSST
MOVX @DPTR,A
CLR BUSY ;Reset busy flag
POP DPH
POP DPL
POP ACC
RETI

START:
SETB BUSY
MOV A,#10000001B ;Send START command
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

SENDDATA:
MOV DPTR,#I2CTXD ;Write data to the data buffer
MOVX @DPTR,A
SETB BUSY
MOV A,#10000010B ;Send SEND command
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

RECVACK:
SETB BUSY
MOV A,#10000011B ;Send read ACK command
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

RECVDATA:
SETB BUSY
MOV A,#10000100B ;Send RECV command
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
CALL WAIT
MOV DPTR,#I2CRXD ;Reading data from a data buffer
MOVX A,@DPTR
RET

SENDACK:
MOV A,#00000000B ;Setting ACK signal
MOV DPTR,#I2CMSST

Nantong guoxin Microelectronics Co., Ltd. Tel: 0513-5501 2928/2929/2966 Fax: 0513-5501 2926/2956/2947 - 406 -
MOVX   @DPTR,A
SETB   BUSY
MOV    A,#10000101B ;Send ACK command
MOV    DPTR,#I2CMSCR
MOVX   @DPTR,A
JMP    WAIT

SENDNAK:
MOV    A,#00000001B ;Setting NAK signal
MOV    DPTR,#I2CMSST
MOVX   @DPTR,A
SETB   BUSY
MOV    A,#10000101B ;Send ACK command
MOV    DPTR,#I2CMSCR
MOVX   @DPTR,A
JMP    WAIT

STOP:
SETB   BUSY
MOV    A,#10000110B ;Send STOP command
MOV    DPTR,#I2CMSCR
MOVX   @DPTR,A
JMP    WAIT

WAIT:
JB     BUSY,$ ;Waiting for the command to be sent
RET

DELAY:
DELAY1:
NOP
NOP
NOP
NOP
DJNZ  R1,DELAY1
DJNZ  R0,DELAY1
RET

MAIN:
MOV    SP,#3FH
MOV    P_SW2,#80H
MOV    A,#11100000B ;Setting I2C Module to Master Mode
MOV    DPTR,#I2CCFG
MOVX   @DPTR,A
MOV    A,#00000000B
MOV    DPTR,#I2CMSST
MOVX   @DPTR,A
SETB   EA
CALL   START ;Send start command
MOV    A,#0A0H
CALL   SENDDATA ;Send device address+write command
CALL   RECEVACK
MOV    A,#000H ;Send storage address high byte
CALL   SENDDATA
CALL   RECEVACK
MOV    A,#000H ;Send storage address low byte
CALL   SENDDATA

Nantong guoxin Microelectronics Co., Ltd.  Tel: 0513-5501 2928/2929/2966  Fax: 0513-5501 2926/2956/2947 - 407 -
CALL RECVACK
MOV A,#12H ;Write test data 1
CALL SENDDATA
CALL RECVACK
MOV A,#78H ;Write test data 2
CALL SENDDATA
CALL RECVACK
CALL STOP ;Send stop command
CALL DELAY ;Waiting for device to write data
CALL START ;Send start command
MOV A,#0A0H ;Send device address+Write command
CALL SENDDATA
CALL RECVACK
CALL SENDDATA
CALL RECVACK
CALL START ;Send start command
MOV A,#0A1H ;Send device address+read command
CALL SENDDATA
CALL RECVACK
CALL RECVACK
CALL RECVDATA ;Read data 1
MOV P0,A
CALL SENDACK
CALL RECVDATA ;Read Data 2
MOV P2,A
CALL SENDNAK
CALL STOP ;Send stop command
JMP $  
END

C code
#include "reg51.h"
#include "intrins.h"
sfr P_SW2 = 0xba;
#define I2CCFG (*(unsigned char volatile xdata *)0xfe80)
#define I2CMSCR (*(unsigned char volatile xdata *)0xfe81)
#define I2CMSST (*(unsigned char volatile xdata *)0xfe82)
#define I2CSLCR (*(unsigned char volatile xdata *)0xfe83)
#define I2CSLST (*(unsigned char volatile xdata *)0xfe84)
#define I2CSLADR (*(unsigned char volatile xdata *)0xfe85)
#define I2CTXD (*(unsigned char volatile xdata *)0xfe86)
#define I2CRXD (*(unsigned char volatile xdata *)0xfe87)
sbit SDA = P1^4;
sbit SCL = P1^5;
bit busy;
void I2C_Isr() interrupt 24
{
    _push_(P_SW2);
    P_SW2 |= 0x80;
    if (I2CMSST & 0x40)
    {
        I2CMSST &= ~0x40; //Clear interrupt flag
        busy = 0;
    }
    _pop_(P_SW2);
}

void Start()
{
    busy = 1;
    I2CMSCR = 0x81; //Send START command
    while (busy);
}

void SendData(char dat)
{
    I2CTXD = dat; //Write data to the data buffer
    busy = 1;
    I2CMSCR = 0x82; //Send SEND command
    while (busy);
}

void RecvACK()
{
    busy = 1;
    I2CMSCR = 0x83; //Send read ACK command
    while (busy);
}

char RecvData()
{
    busy = 1;
    I2CMSCR = 0x84; //Send RECV command
    while (busy);
    return I2CRXD;
}

void SendACK()
{
    I2CMSST = 0x00; //Setting ACK signal
    busy = 1;
    I2CMSCR = 0x85; //Send ACK command
    while (busy);
}

void SendNAK()
{
    I2CMSST = 0x01; //Setting NAK signal
    busy = 1;
    I2CMSCR = 0x85; //Send ACK command
    while (busy);
}
void Stop()
{
    busy = 1;
    I2CMSCR = 0x86; //Send STOP command
    while (busy);
}

void Delay()
{
    int i;
    for (i=0; i<3000; i++)
    {
        _nop_();
        _nop_();
        _nop_();
        _nop_();
    }
}

void main()
{
    P_SW2 = 0x80;
    I2CCFG = 0xe0; //Enable I2C master mode
    I2CMSST = 0x00;
    EA = 1;
    Start(); //Send start command
    SendData(0xa0); //Send device address+Write command
    RevcACK();
    SendData(0x00); //Send storage address high byte
    RevcACK();
    SendData(0x00); //Send storage address low byte
    RevcACK();
    SendData(0x12); //Write test data 1
    RevcACK();
    SendData(0x78); //Write test data 2
    RevcACK();
    Stop(); //Send stop command
    Delay(); //Waiting for device to write data
    Start(); //Send start command
    SendData(0xa0); //Send device address+Write command
    RevcACK();
    SendData(0x00); //Send storage address high byte
    RevcACK();
    SendData(0x00); //Send storage address low byte
    RevcACK();
    Start(); //Send start command
    SendData(0xa1); //Send device address+read command
    RevcACK();
P0 = RevcData(); //Read data 1
    SendACK();
P2 = RevcData(); //Read data 2
    SendNAK();
    Stop(); //Send stop command
P_SW2 = 0x00;

while (1);

21.4.2 I²C master mode access AT24C256(Query mode)

Assembly code

<table>
<thead>
<tr>
<th>P_SW2</th>
<th>DATA</th>
<th>0BAH</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2CCFG</td>
<td>XDATA</td>
<td>0FE80H</td>
</tr>
<tr>
<td>I2CMSCR</td>
<td>XDATA</td>
<td>0FE81H</td>
</tr>
<tr>
<td>I2CMSST</td>
<td>XDATA</td>
<td>0FE82H</td>
</tr>
<tr>
<td>I2CSLCR</td>
<td>XDATA</td>
<td>0FE83H</td>
</tr>
<tr>
<td>I2CSLST</td>
<td>XDATA</td>
<td>0FE84H</td>
</tr>
<tr>
<td>I2CSLADR</td>
<td>XDATA</td>
<td>0FE85H</td>
</tr>
<tr>
<td>I2CTXD</td>
<td>XDATA</td>
<td>0FE86H</td>
</tr>
<tr>
<td>I2CRXD</td>
<td>XDATA</td>
<td>0FE87H</td>
</tr>
</tbody>
</table>

SDA BIT P1.4
SCL BIT P1.5

ORG 0000H
LJMP MAIN

ORG 0100H

START:
MOV A,#00000001B ;Send START command
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

SENDDATA:
MOV DPTR,#I2CTXD ;Write data to the data buffer
MOVX @DPTR,A
MOV A,#00000010B ;Send SEND command
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

RECVACK:
MOV A,#00000011B ;Send read ACK command
MOV DPTR,#I2CMSST
MOVX @DPTR,A
CALL WAIT

RECVDATA:
MOV A,#00000100B ;Send RECV command
MOV DPTR,#I2CMSST
MOVX @DPTR,A
CALL WAIT
MOV DPTR,#I2CRXD ;Reading data from a data buffer
MOVX A,@DPTR
RET

SENDACK:
MOV A,#00000000B ;Setting ACK signal
MOV DPTR,#I2CMSST
MOVX @DPTR,A
MOV A,#00000101B ;Send ACK command
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

SENDNAK:
MOV A,#00000001B ;Setting NAK signal
MOV DPTR,#I2CMSST
MOVX @DPTR,A
MOV A,#00000101B ;Send ACK command
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

STOP:
MOV A,#00000110B ;Send STOP command
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

WAIT:
MOV DPTR,#I2CMSST ;Clear interrupt flag
MOVX A,@DPTR
JNB ACC.6,WAIT
ANL A,#NOT 40H
MOVX @DPTR,A
RET

DELAY:
MOV R0,#0
MOV R1,#0

DELAY1:
NOP
NOP
NOP
NOP
DJNZ R1,DELAY1
DJNZ R0,DELAY1
RET

MAIN:
MOV SP,#3FH
MOV P_SW2,#80H

MOV A,#11100000B ;Setting I2C Module to Master Mode
MOV DPTR,#I2CCFG
MOVX @DPTR,A
MOV A,#00000000B
MOV DPTR,#I2CMSST
MOVX @DPTR,A
CALL START ;Send start command
MOV A,#0A0H
CALL SENDDATA ;Send device address+Write command
CALL RECVACK
MOV A,#000H
CALL SENDDATA ;Send storage address high byte
CALL RECVACK
MOV A,#000H
CALL SENDDATA ;Send storage address low byte
CALL RECVACK
MOV A,#12H ;Write test data 1
CALL SENDDATA
CALL RECVACK
Mov A,#78H ;Write test data 2
CALL SENDDATA
CALL RECVACK
CALL STOP ;Send stop command
CALL DELAY ;Waiting for device to write data

CALL START ;Send start command
MOV A,#0A0H ;Send device address+Write command
CALL SENDDATA
CALL RECVACK
MOV A,#000H ;Send storage address high byte
CALL SENDDATA
CALL RECVACK
MOV A,#000H ;Send storage address low byte
CALL SENDDATA
CALL RECVACK
CALL START ;Send start command
MOV A,#0A1H ;Send device address+read command
CALL SENDDATA
CALL RECVACK
CALL RECVDATA ;Read data 1
MOV P0,A
CALL SENDACK
CALL RECVDATA ;Read Data 2
MOV P2,A
CALL SENDNAK
CALL STOP ;Send stop command
JMP $;
END

C code
#include "reg51.h"
#include "intrins.h"
sfr P_SW2 = 0xba;
#define I2CCFG (*(unsigned char volatile xdata *)0xfe80)
#define I2CMSCR (*(unsigned char volatile xdata *)0xfe81)
#define I2CMSST (*(unsigned char volatile xdata *)0xfe82)
#define I2CSLCR (*(unsigned char volatile xdata *)0xfe83)
#define I2CSLST (*(unsigned char volatile xdata *)0xfe84)
#define I2CSLADR (*(unsigned char volatile xdata *)0xfe85)
#define I2CTXD (*(unsigned char volatile xdata *)0xfe86)
#define I2CRXD (*(unsigned char volatile xdata *)0xfe87)
sbit SDA = P1^4;
sbit SCL = P1^5;

void Wait()
{
    while (!(I2CMSST & 0x40));
}

Nantong guoxin Microelectronics Co., Ltd. Tel: 0513-5501 2928/2929/2966 Fax: 0513-5501 2926/2956/2947 - 413 -
I2CMSST &= ~0x40;
}

void Start()
{
    I2CMSCR = 0x01; //Send START command
    Wait();
}

void SendData(char dat)
{
    I2CTXD = dat; //Write data to the data buffer
    I2CMSCR = 0x02; //Send SEND command
    Wait();
}

void RecvACK()
{
    I2CMSCR = 0x03; //Send read ACK command
    Wait();
}

char RecvData()
{
    I2CMSCR = 0x04; //Send RECV command
    Wait();
    return I2CRXD;
}

void SendACK()
{
    I2CMSST = 0x00; //Setting ACK signal
    I2CMSCR = 0x05; //Send ACK command
    Wait();
}

void SendNAK()
{
    I2CMSST = 0x01; //Setting NAK signal
    I2CMSCR = 0x05; //Send ACK command
    Wait();
}

void Stop()
{
    I2CMSCR = 0x06; //Send STOP command
    Wait();
}

void Delay()
{
    int i;

    for (i=0; i<3000; i++)
    {
        _nop_();
        _nop_();
        _nop_();
    }
}
21.4.3 I²C master mode access PCF8563

Assembly code

<table>
<thead>
<tr>
<th>P_SW2</th>
<th>DATA</th>
<th>0BAH</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2CCFG</td>
<td>XDATA</td>
<td>0FE80H</td>
</tr>
<tr>
<td>I2CMSCR</td>
<td>XDATA</td>
<td>0FE81H</td>
</tr>
<tr>
<td>I2CMSST</td>
<td>XDATA</td>
<td>0FE82H</td>
</tr>
<tr>
<td>I2CSLCR</td>
<td>XDATA</td>
<td>0FE83H</td>
</tr>
</tbody>
</table>
I2CSLST XDATA 0FE84H
I2CSLADR XDATA 0FE85H
I2CTXD XDATA 0FE86H
I2CRXD XDATA 0FE87H

SDA BIT P1.4
SCL BIT P1.5

ORG 0000H
LJMP MAIN

ORG 0100H

START:
MOV A,#00000001B ;Send START command
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

SENDDATA:
MOV DPTR,#I2CTXD ;Write data to the data buffer
MOVX @DPTR,A
MOV A,#00000010B ;Send SEND command
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

RECVACK:
MOV A,#00000011B ;Send read ACK command
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

RECVDATA:
MOV A,#00000100B ;Send RECV command
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
CALL WAIT
MOV DPTR,#I2CRXD ;Reading data from a data buffer
MOVX A,@DPTR
RET

SENDACK:
MOV A,#00000000B ;Setting ACK signal
MOV DPTR,#I2CMSST
MOVX @DPTR,A
MOV A,#000000101B ;Send ACK command
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

SENDNAK:
MOV A,#00000001B ;Setting NAK signal
MOV DPTR,#I2CMSST
MOVX @DPTR,A
MOV A,#00000101B ;Send ACK command
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

STOP:
MOV A,#00000110B ;Send STOP command
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT
WAIT:

```assembly
WAIT:
MOVDPTR,#I2CMSST ;Clear interrupt flag
MOVXA,@DPTR
JNBACC.6,WAIT
ANL A,#NOT40H
MOVX@DPTR,A
RET
```

DELAY:

```assembly
DELAY:
MOVXR0,#0
MOVXR1,#0
```

DELAY1:

```assembly
DELAY1:
NOP
NOP
NOP
NOP
DJNZR1,DELAY1
DJNZR0,DELAY1
RET
```

MAIN:

```assembly
MAIN:
MOVSPI#3FH
MOVQP_SW2,#80H
MOV A,#11100000B ;SettingI2CModuletoMasterMode
MOVDPTR,#I2CCFG
MOVX@DPTR,A
MOV A,#00000000B
MOVDPTR,#I2CMSST
MOVX@DPTR,A
CALLSTART;Sendstartcommand
MOV A,#0A2H
CALLSENDDATA;Senddeviceaddress+Writecommand
CALLERCVACK
MOV A,#002H ;sendstorageaddress
CALLENDATA
CALLERCVACK
MOV A,#00H ;Settingthesecondvalue
CALLENDATA
CALLERCVACK
MOV A,#00H ;Settheminutevalue
CALLENDATA
CALLERCVACK
MOV A,#12H ;Setthehourvalue
CALLENDATA
CALLERCVACK
CALLSTOP;Sendstopcommand
LOOP:
```

```assembly
CALLSTART;Sendstartcommand
MOV A,#0A2H
CALLSENDDATA;Senddeviceaddress+Writecommand
CALLERCVACK
MOV A,#002H ;sendstorageaddress
CALLENDATA
CALLERCVACK
CALLSTART;Sendstartcommand
MOV A,#0A3H ;Senddeviceaddress+readcommand
```
CALL SENDDATA
CALL RECVACK
CALL RECVDATA ;Read second value
MOV P0,A
CALL SENDACK
CALL RECVDATA ;Read minute value
MOV P2,A
CALL SENDACK
CALL RECVDATA ;Read hour value
MOV P3,A
CALL SENDNAK
CALL STOP ;Send stop command
CALL DELAY
JMP LOOP

END

C code

#include "reg51.h"
#include "intrins.h"
sfr P_SW2 = 0xba;
#define I2CCFG (*(unsigned char volatile xdata *)0xfe80)
#define I2CMSCR (*(unsigned char volatile xdata *)0xfe81)
#define I2CMSST (*(unsigned char volatile xdata *)0xfe82)
#define I2CSLCR (*(unsigned char volatile xdata *)0xfe83)
#define I2CSLST (*(unsigned char volatile xdata *)0xfe84)
#define I2CSLADR (*(unsigned char volatile xdata *)0xfe85)
#define I2CTXD (*(unsigned char volatile xdata *)0xfe86)
#define I2CRXD (*(unsigned char volatile xdata *)0xfe87)
sbit SDA = P1^4;
sbit SCL = P1^5;

void Wait()
{
    while (!(I2CMSST & 0x40));
    I2CMSST &= ~0x40;
}

void Start()
{
    I2CMSCR = 0x01; //Send START command
    Wait();
}

void SendData(char dat)
{
    I2CTXD = dat; //Write data to the data buffer
    I2CMSCR = 0x02; //Send SEND command
    Wait();
}

void RecvACK()
```c
void Delay()
{
    int i;

    for (i=0; i<3000; i++)
    {
        _nop_();
        _nop_();
        _nop_();
        _nop_();
    }
}

void main()
{
    P_SW2 = 0x80;

    I2CCFG = 0xe0; //Enable I2C master mode
    I2CMSST = 0x00;

    Start(); //Send start command
    SendData(0xa2); //Send device address+Write command
    RecvACK();
    SendData(0x02); //Send storage address
    RecvACK();
    SendData(0x00); //Setting the second value
```
_RECVACK();
SendData(0x00);  //Set the minute value
_RECVACK();
SendData(0x12);  //Set the hour value
_RECVACK();
Stop();  //Send stop command

while (1)
{
    Start();  //Send start command
    SendData(0xa2);  //Send device address+Write command
    __RECVACK();
    SendData(0x02);  //send storage address
    __RECVACK();
    Start();  //Send start command
    SendData(0xa3);  //Send device address+read command
    __RECVACK();
    P0 = _RecvData();  //Read second value
    _SENDACK();
    P2 = _RecvData();  //Read minute value
    _SENDACK();
    P3 = _RecvData();  //Read hour value
    _SENDNAK();
    Stop();  //Send stop command
    Delay();
}

21.4.4 I²C Slave Mode(interrupt mode)

Assembly code

<table>
<thead>
<tr>
<th>P_SW2</th>
<th>DATA</th>
<th>0BAH</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2CCFG</td>
<td>XDATA</td>
<td>0FE80H</td>
</tr>
<tr>
<td>I2CMSCR</td>
<td>XDATA</td>
<td>0FE81H</td>
</tr>
<tr>
<td>I2CMSST</td>
<td>XDATA</td>
<td>0FE82H</td>
</tr>
<tr>
<td>I2CSLCR</td>
<td>XDATA</td>
<td>0FE83H</td>
</tr>
<tr>
<td>I2CSLST</td>
<td>XDATA</td>
<td>0FE84H</td>
</tr>
<tr>
<td>I2CSLADR</td>
<td>XDATA</td>
<td>0FE85H</td>
</tr>
<tr>
<td>I2CTXD</td>
<td>XDATA</td>
<td>0FE86H</td>
</tr>
<tr>
<td>I2CRXD</td>
<td>XDATA</td>
<td>0FE87H</td>
</tr>
</tbody>
</table>

SDA BIT P1.4
SCL BIT P1.5
ISDA BIT 20H.0 ;Device address flag
ISMA BIT 20H.1 ;Storage address flag

ADDR DATA 21H

ORG 0000H
LJMP MAIN
ORG 00C3H
LJMP I2CISR

ORG 0100H
I2CISR:

```
PUSH ACC
PUSH PSW
PUSH DPL
PUSH DPH
MOV DPTR,#I2CSLST ;Detection of slave status
MOVX A,@DPTR
JB ACC.6,STARTIF
JB ACC.5,RXIF
JB ACC.4,TXIF
JB ACC.3,STOPIF

ISREXIT:
POP DPH
POP DPL
POP PSW
POP ACC
RETI

STARTIF:
ANL A,#NOT 40H ;Handling START events
MOVX @DPTR,A
JMP ISREXIT

RXIF:
ANL A,#NOT 20H ;Handling RECV events
MOVX @DPTR,A
MOV DPTR,#I2CRXD
MOVX A,@DPTR
JBC ISDA,RXDA
JBC ISMA,RXMA
MOV R0,ADDR ;Handling RECV events(RECV DATA)
INC ADDR
JMP ISREXIT

RXDA:
JMP ISREXIT ;Handling RECV events(RECV DEVICE ADDR)

RXMA:
MOV ADDR,A ;Handling RECV events(RECV MEMORY ADDR)
MOV R0,A
MOVX A,R0
MOV DPTR,#I2CTXD
MOVX @DPTR,A
JMP ISREXIT

TXIF:
ANL A,#NOT 10H ;Handling SEND events
MOVX @DPTR,A
JB ACC.1,RXNAK
INC ADDR
MOV R0,ADDR
MOVX A,R0
MOV DPTR,#I2CTXD
MOVX @DPTR,A
JMP ISREXIT

RXNAK:
MOVX A,#FFH
MOV DPTR,#I2CTXD
MOVX @DPTR,A
JMP ISREXIT

STOPIF:
ANL A,#NOT 08H ;Handling STOP events
```
MAIN:

MOV P_SW2,#80H
MOV A,#10000000B ;Enable I2C slave mode
MOV DPTR,#I2CCFG
MOVX @DPTR,A
MOV A,#01011010B ;Set slave device address to 5A
MOV DPTR,#I2CSLADR
MOVX @DPTR,A
MOV A,#00000000B
MOV DPTR,#I2CSLST
MOVX @DPTR,A
MOV A,#01111000B ;Enable slave mode interrupt
MOV DPTR,#I2CSLCR
MOVX @DPTR,A

SETB ISDA ;User variable initialization
SETB ISMA
CLR A
MOV ADDR,A
MOV R0,A
MOVX A,@R0
MOV DPTR,#I2CTXD
MOVX @DPTR,A

SETB EA
SJMP $

END

C code

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

sfr P_SW2 = 0xba;

#define I2CCFG (*(unsigned char volatile xdata *)0xfe80)
#define I2CMSCR (*(unsigned char volatile xdata *)0xfe81)
#define I2CMSST (*(unsigned char volatile xdata *)0xfe82)
#define I2CSLCR (*(unsigned char volatile xdata *)0xfe83)
#define I2CSLST (*(unsigned char volatile xdata *)0xfe84)
#define I2CSLADR (*(unsigned char volatile xdata *)0xfe85)
#define I2CTXD (*(unsigned char volatile xdata *)0xfe86)
#define I2CRXD (*(unsigned char volatile xdata *)0xfe87)

sbit SDA = P1^4; //Device address flag
sbit SCL = P1^5; //Storage address flag

bit isda;  //Device address flag
bit isma;  //Storage address flag
unsigned char addr;
unsigned char pdata buffer[256];

void I2C_Isr() interrupt 24
{
    _push_(P_SW2);
    P_SW2 |= 0x80;

    if (I2CSLST & 0x40) {
        I2CSLST &= ~0x40; //Handling START events
    } else if (I2CSLST & 0x20) {
        I2CSLST &= ~0x20; //Handling RECV events
        if (isda) {
            isda = 0; //Handling RECV events(RECV DEVICE ADDR)
        } else if (isma) {
            isma = 0; //Handling RECV events(RECV MEMORY ADDR)
            addr = I2CRXD;
            I2CTXD = buffer[addr];
        } else {
            buffer[addr++] = I2CRXD; //Handling RECV events(RECV DATA)
        }
    } else if (I2CSLST & 0x10) {
        I2CSLST &= ~0x10; //Handling SEND events
        if (I2CSLST & 0x02) {
            I2CTXD = 0xff; //Received NAK then stop reading data
        } else {
            I2CTXD = buffer[++addr]; //Receive ACK then continue reading data
        }
    } else if (I2CSLST & 0x08) {
        I2CSLST &= ~0x08; //Handling STOP events
        isda = 1;
        isma = 1;
    }

    _pop_(P_SW2);
}

void main()
{
    P_SW2 = 0x80;
    I2CCFG = 0x81; //Enable I2C slave mode
    I2CSLADR = 0x5a; //Set slave device address to 5A
    I2CSLST = 0x00;
}
I2CSLCR = 0x78;   //Enable slave mode interrupt
EA = 1;

isda = 1;       //User variable initialization
isma = 1;
addr = 0;
I2CTXD = buffer[addr];

while (1);

21.4.5 \textit{I}^2\textit{C} Slave Mode (Query mode)

\textbf{Assembly code}

\begin{verbatim}
P_SW2 DATA 0BAH
I2CCFG      XDATA 0FE80H
I2CMSCR     XDATA 0FE81H
I2CMSST     XDATA 0FE82H
I2CSLCR     XDATA 0FE83H
I2CSLST     XDATA 0FE84H
I2CSLADR    XDATA 0FE85H
I2CTXD      XDATA 0FE86H
I2CRXD      XDATA 0FE87H

SDA        BIT P1.4
SCL        BIT P1.5
ISDA       BIT 20H.0 ;Device address flag
ISMA       BIT 20H.1 ;Storage address flag

ADDR       DATA 21H

ORG 0000H
LJMP MAIN

ORG 0100H

MAIN:
  MOV P_SW2,#80H
  MOV A,#10000001B ;Enable I2C slave mode
  MOV DPTR,#I2CCFG
  MOVX @DPTR,A
  MOV A,#01011010B ;Set slave device address to 5A
  MOV DPTR,#I2CSLADR
  MOVX @DPTR,A
  MOV A,#00000000B
  MOV DPTR,#I2CSLST
  MOVX @DPTR,A
  MOV A,#00000000B ;Disable slave mode interrupt
  MOV DPTR,#I2CSLCR
  MOVX @DPTR,A
  SETB ISDA ;User variable initialization
  SETB ISMA
  CLR A
  MOV ADDR,A

Nantong guoxin Microelectronics Co., Ltd. Tel: 0513-5501 2928/2929/2966 Fax: 0513-5501 2926/2956/2947 - 424 -
MOV R0,A
MOVX A,@R0
MOV DPTR,#I2CTXD
MOVX @DPTR,A

LOOP:
MOV DPTR,#I2CSLST ;Detection of slave status
MOVX A,@DPTR
JB ACC.6,STARTIF
JB ACC.5,RXIF
JB ACC.4,TXIF
JB ACC.3,STOPIF
JMP LOOP

STARTIF:
ANL A,#NOT 40H ;Handling START events
MOVX @DPTR,A
JMP LOOP

RXIF:
ANL A,#NOT 20H ;Handling RECV events
MOVX @DPTR,A
MOV DPTR,#I2CRXD
MOVX A,@DPTR
JBC ISDA,RXDA
JBC ISMA,RXMA
MOV R0,ADDR ;Handling RECV events(RECV DATA)
MOVX @R0,A
INC ADDR
JMP LOOP

RXDA:
JMP LOOP ;Handling RECV events(RECV DEVICE ADDR)

RXMA:
MOV ADDR,A ;Handling RECV events(RECV MEMORY ADDR)
MOV R0,A
MOV DPTR,#I2CTXD
MOVX @DPTR,A
JMP LOOP

TXIF:
ANL A,#NOT 10H ;Handling SEND events
MOVX @DPTR,A
JB ACC.1,RXNAK
INC ADDR
MOV R0,ADDR
MOVX A,@R0
MOV DPTR,#I2CTXD
MOVX @DPTR,A
JMP LOOP

RXNAK:
MOVX A,#FFH
MOV DPTR,#I2CTXD
MOVX @DPTR,A
JMP LOOP

STOPIF:
ANL A,#NOT 08H ;Handling STOP events
MOVX @DPTR,A
SETB ISDA
SETB ISMA
JMP LOOP
C code

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

sfr P_SW2 = 0xba;

#define I2CCFG (*(unsigned char volatile xdata *)0xfe80)
#define I2CMSCCR (*(unsigned char volatile xdata *)0xfe81)
#define I2CMSST (*(unsigned char volatile xdata *)0xfe82)
#define I2CSLCR (*(unsigned char volatile xdata *)0xfe83)
#define I2CSLST (*(unsigned char volatile xdata *)0xfe84)
#define I2CSLADR (*(unsigned char volatile xdata *)0xfe85)
#define I2CTXD (*(unsigned char volatile xdata *)0xfe86)
#define I2CRXD (*(unsigned char volatile xdata *)0xfe87)

sbit SDA = P1^4;
sbit SCL = P1^5;

bit isda; //Device address flag
bit isma; //Storage address flag
unsigned char addr;
unsigned char pdata buffer[256];

void main()
{
P_SW2 = 0x80;
I2CCFG = 0x81; //Enable I2C slave mode
I2CSLADR = 0x5a; //Set slave device address to 5A
I2CSLCR = 0x00; //Disable slave mode interrupt
isda = 1; //User variable initialization
isma = 1;
addr = 0;
I2CTXD = buffer[addr];

while (1)
{
    if (I2CSLST & 0x40)
        I2CSLST &= ~0x40; //Handling START events
    else if (I2CSLST & 0x20)
        if (isda)
            isda = 0; //Handling RECV events(RECV DEVICE ADDR)
        else if (isma)
            isma = 0; //Handling RECV events(RECV MEMORY ADDR)
    addr = I2CRXD;
}
```
I2CTXD = buffer[addr];
}  
else
{
    buffer[addr++] = I2CRXD;  //Handling RECV events(RECV DATA)
}
}  
else if (I2CSLST & 0x10)
{
    I2CSLST &= ~0x10;    //Handling SEND events
    if (I2CSLST & 0x02)
    {
        I2CTXD = 0xff;    //Received NAK then stop reading data
    }
    else
    {
        I2CTXD = buffer[++addr];  //Receive ACK then continue reading data
    }
}
else if (I2CSLST & 0x08)
{
    I2CSLST &= ~0x08;    //Handling STOP events
    isda = 1;
    isma = 1;
}
}

21.4.6 Host code for testing I2C slave mode code

**Assembly code**

<table>
<thead>
<tr>
<th>P_SW2</th>
<th>DATA</th>
<th>0BAH</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2CCFG</td>
<td>XDATA</td>
<td>0FE80H</td>
</tr>
<tr>
<td>I2CMSCR</td>
<td>XDATA</td>
<td>0FE81H</td>
</tr>
<tr>
<td>I2CMSSST</td>
<td>XDATA</td>
<td>0FE82H</td>
</tr>
<tr>
<td>I2CSLSCR</td>
<td>XDATA</td>
<td>0FE83H</td>
</tr>
<tr>
<td>I2CSLST</td>
<td>XDATA</td>
<td>0FE84H</td>
</tr>
<tr>
<td>I2CSLADR</td>
<td>XDATA</td>
<td>0FE85H</td>
</tr>
<tr>
<td>I2CTXD</td>
<td>XDATA</td>
<td>0FE86H</td>
</tr>
<tr>
<td>I2CRXD</td>
<td>XDATA</td>
<td>0FE87H</td>
</tr>
</tbody>
</table>

SDA BIT P1.4  
SCL BIT P1.5

ORG 0000H  
LJMP MAIN

ORG 0100H

**START:**

MOV A,#00000001B ;Send START command
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

**SENDATA:**

MOV DPTR,#I2CTXD ;Write data to the data buffer
MOVX @DPTR,A ; Send SEND command
MOV A,#00000010B
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

RECVACK:
MOV A,#00000011B ; Send read ACK command
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

RECVDATA:
MOV A,#00000100B ; Send RECV command
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
CALL WAIT
MOV DPTR,#I2CRXD ; Reading data from a data buffer
MOVX A,@DPTR
RET

SENDACK:
MOV A,#00000000B ; Setting ACK signal
MOV DPTR,#I2CMSST
MOVX @DPTR,A
MOV A,#00000101B ; Send ACK command
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

SENDNAK:
MOV A,#00000001B ; Setting NAK signal
MOV DPTR,#I2CMSST
MOVX @DPTR,A
MOV A,#00000001B
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

STOP:
MOV A,#00000110B ; Send STOP command
MOV DPTR,#I2CMSCR
MOVX @DPTR,A
JMP WAIT

WAIT:
MOV DPTR,#I2CMSST ; Clear interrupt flag
MOVX A,@DPTR
JNB ACC.6,WAIT
ANL A,#40H
MOVX @DPTR,A
RET

DELAY:
MOV R0,#0
MOV R1,#0

DELAY1:
NOP
NOP
NOP
NOP
DJNZ R1,DELAY1
DJNZ R0,DELAY1
RET
```c
#include "reg51.h"
#include "intrins.h"

sfr P_SW2 = 0xba;

MAIN:
    MOV SP,#3FH
    MOV P_SW2,#80H
    
    MOV A,#11100000B ;Setting I2C Module to Host Mode
    MOV DPTR,#I2CCFG
    MOVX @DPTR,A
    MOV A,#00000000B
    MOV DPTR,#I2CMSST
    MOVX @DPTR,A
    CALL START ;Send start command
    MOV A,#5AH ;Slave address 5A
    CALL SENDDATA ;Send device address+Write command
    CALL RECVACK
    MOV A,#000H ;send storage address
    CALL SENDDATA
    CALL RECVACK
    MOV A,#12H ;Write test data 1
    CALL SENDDATA
    CALL RECVACK
    MOV A,#78H ;Write test data 2
    CALL SENDDATA
    CALL RECVACK
    CALL STOP ;Send stop command
    CALL DELAY ;Waiting for device to write data
    CALL START ;Send start command
    MOV A,#5AH ;Send device address+Write command
    CALL SENDDATA
    CALL RECVACK
    MOV A,#000H ; send storage address
    CALL SENDDATA
    CALL RECVACK
    CALL START ;Send start command
    MOV A,#5BH ;Send device address+read command
    CALL SENDDATA
    CALL RECVACK
    CALL RECVDATA ;Read data 1
    MOV P0,A
    CALL SENDACK
    CALL RECVDATA ;Read data 2
    MOV P2,A
    CALL SENDNAK
    CALL STOP ;Send stop command
    JMP $

END
```

C code

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

sfr P_SW2 = 0xba;
```
#define I2CCFG (*(unsigned char volatile xdata *)0xfe80)
#define I2CMSCR (*(unsigned char volatile xdata *)0xfe81)
#define I2CMSST (*(unsigned char volatile xdata *)0xfe82)
#define I2CSLSCR (*(unsigned char volatile xdata *)0xfe83)
#define I2CSLST (*(unsigned char volatile xdata *)0xfe84)
#define I2CSLADR (*(unsigned char volatile xdata *)0xfe85)
#define I2CTXD (*(unsigned char volatile xdata *)0xfe86)
#define I2CRXD (*(unsigned char volatile xdata *)0xfe87)

sbit SDA = P1^4;
sbit SCL = P1^5;

void Wait()
{
    while (!(I2CMSST & 0x40));
    I2CMSST &= ~0x40;
}

void Start()
{
    I2CMSCR = 0x01; //Send START command
    Wait();
}

void SendData(char dat)
{
    I2CTXD = dat; //Write data to the data buffer
    I2CMSCR = 0x02; //Send SEND command
    Wait();
}

void RecvACK()
{
    I2CMSCR = 0x03; //Send read ACK command
    Wait();
}

char RecvData()
{
    I2CMSCR = 0x04; //Send RECV command
    Wait();
    return I2CRXD;
}

void SendACK()
{
    I2CMSST = 0x00; //Setting ACK signal
    I2CMSCR = 0x05; //Send ACK command
    Wait();
}

void SendNAK()
{
    I2CMSST = 0x01; //Setting NAK signal
    I2CMSCR = 0x05; //Send ACK command
    Wait();
}
void Stop()
{
    I2CMSCR = 0x06; //Send STOP command
    Wait();
}

void Delay()
{
    int i;

    for (i=0; i<3000; i++)
    {
        _nop_();
        _nop_();
        _nop_();
        _nop_();
    }
}

void main()
{
    P_SW2 = 0x80;

    I2CCFG = 0xe0; //Enable I2C host mode
    I2CMSST = 0x00;

    Start();  //Send start command
    SendData(0x5a); //Send device address+Write command
    RecvACK();
    SendData(0x00); //send storage address
    RecvACK();
    SendData(0x12); //Write test data 1
    RecvACK();
    SendData(0x78); //Write test data 2
    RecvACK();
    Stop();  //Send stop command

    Start();  //Send start command
    SendData(0x5a); //Send device address+Write command
    RecvACK();
    SendData(0x00); //Send storage address high byte
    RecvACK();
    Start();  //Send start command
    SendData(0x5b); //Send device address+read command
    RecvACK();
    P0 = RecvData(); //Read data 1
    SendACK();
    P2 = RecvData(); //Read data 2
    SendNAK();
    Stop();  //Send stop command

    P_SW2 = 0x00;

    while (1);
}
22 Enhanced double data pointer

The STC8 series microcontrollers integrate two sets of 16-bit data pointers. Through program control, the data pointer can be automatically incremented or decremented and the two data pointers can be automatically switched.

Related special function register.

<table>
<thead>
<tr>
<th>symbol</th>
<th>description</th>
<th>addr</th>
<th>Bit address and symbol</th>
<th>Value after reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPL</td>
<td>Data pointer (low bytes)</td>
<td>82H</td>
<td>B7 B6 B5 B4 B3 B2 B1 B0</td>
<td>0000,0000</td>
</tr>
<tr>
<td>DPH</td>
<td>Data pointer (high bytes)</td>
<td>83H</td>
<td></td>
<td>0000,0000</td>
</tr>
<tr>
<td>DPL1</td>
<td>Second set of data pointers (low bytes)</td>
<td>E4H</td>
<td>ID1 ID0</td>
<td>TSL AU1 AU0</td>
</tr>
<tr>
<td>DPH1</td>
<td>second set of data pointers (high byte)</td>
<td>E5H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DPS</td>
<td>DPTR pointer selector</td>
<td>E3H</td>
<td>ID1 ID0 TSL AU1 AU0 - - SEL</td>
<td>0000,0000</td>
</tr>
<tr>
<td>TA</td>
<td>DPTR timing control register</td>
<td>AEH</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Group 1 16-bit data pointer register (DPTR0)**

<table>
<thead>
<tr>
<th>symbol</th>
<th>address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPL</td>
<td>82H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DPH</td>
<td>83H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DPL is low 8 bit data (low bytes)
DPH is high 8 bits of data (high bytes)
DPL and DPH are combined into the first set of 16-bit data pointer registers DPTR0

**Group 2 16-bit data pointer registers (DPTR1)**

<table>
<thead>
<tr>
<th>symbol</th>
<th>address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPL1</td>
<td>E4H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DPH1</td>
<td>E5H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DPL is low 8 bit data (low bytes)
DPH is high 8 bits of data (high bytes)
DPL1 and DPH1 combined into a second set of 16-bit data pointer registers DPTR1.

**Data pointer control register**

<table>
<thead>
<tr>
<th>symbol</th>
<th>address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPS</td>
<td>E3H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ID1: Control DPTR1 automatic increment mode
- 0: DPTR1 auto increment
- 1: DPTR1 auto decrement

ID0: Control DPTR0 automatic increment mode
- 0: DPTR0 auto increment
- 1: DPTR0 auto decrement
TSL: DPTR0/DPTR1 automatic switching control (automatic reverse of SEL)

0: Turn off automatic switching function
1: Enable automatic switching function

When the TSL bit is set, the system automatically negates the SEL bit each time the relevant instruction is executed.

TSL-related instructions include the following:

- MOV DPTR,#data16
- INC DPTR
- MOVC A,@A+DPTR
- MOVX A,@DPTR
- MOVX @DPTR,a

AU1/AU0: Enable DPTR1/DPTR0 to use the ID1/ID0 control bit for auto increment/decrement control

0: Turn off auto increment/decrement
1: Enable auto increment/decrement

Note: In the write protection mode, the AU0 and AU1 bits cannot be individually enabled. If the AU1 bit is enabled individually, the AU0 bit will be automatically enabled. If AU0 is enabled alone, it has no effect. If you need to enable AU1 or AU0 separately, you must use the TA register to trigger the DPS protection mechanism (refer to the description of the TA register).

In addition, DPTR0/DPTR1 is automatically incremented/decremented only after executing the following three instructions. 3 related instructions are as follows:

- MOVC A,@A+DPTR
- MOVX A,@DPTR
- MOVX @DPTR,a

SEL: Select DPTR0/DPTR1 as the current target DPTR

0: Select DPTR0 as Target DPTR
1: Select DPTR1 as Target DPTR

SEL: Select target DPTR is valid for the following instructions:

- MOV DPTR,#data16
- INC DPTR
- MOVC A,@A+DPTR
- MOVX A,@DPTR
- MOVX @DPTR,a
- JMP @A+DPTR

Data pointer control register

<table>
<thead>
<tr>
<th>symbol</th>
<th>address</th>
<th>B7</th>
<th>B6</th>
<th>B5</th>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
<th>B0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TA</td>
<td>AEH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The TA register write-protects AU1 and AU0 in the DPS register. Since the program cannot write AU1 and AU0 individually in DPS, when the AU1 or AU0 needs to be enabled separately, the TA register must be used for triggering. The TA register is a write-only register. When you need to enable AU1 or AU0 separately, you must follow the steps below:

- CLR EA ;close interruption (required)
- MOV TA,#0AAH ;Write trigger command sequence 1
STC8F Series Manual

Technical support: 13922809991/13922805190

MOV TA,#55H ;Write trigger command sequence 2

MOV DPS,#xxH ;Write protection is temporarily disabled and any value can be written to DPS

SETB EA ;Open interruptions (if necessary)

22.1 Sample program

22.1.1 Example code 1
Reversely copy the 4-byte data in program space 1000H to 1003H to 0100H to 0103H in the expansion RAM, that is

C:1000H -> X:0103H
C:1001H -> X:0102H
C:1002H -> X:0101H
C:1003H -> X:0100H

Assembly code

ORG 0000H
LJMP MAIN

ORG 0100H

MAIN:
MOV SP , #3FH
MOV DPS,#00100000B ;Enable TSL and select DPTR0
MOV DPTR,#1000H ;Write 1000H to DPTR0. Select DPTR1 as DPTR after the execution is completed.
MOV DPTR,#0103H ;Write 0103H to DPTR1
MOV DPS,#10111000B ;Set DPTR1 to decrement mode, DPTR0 to decrement mode, enable TSL and AU0 and AU1, and select DPTR0 as the current DPTR
MOV R7,#4 ;Setting the number of data replication

COPY_NEXT:
CLR A ;
MOVC A,@A+DPTR ;Reading data from the program space pointed to by DPTR0,
;After completion, DPTR0 is automatically incremented by 1 and
;DPTR1 is set to the next target DPTR.
MOVX @DPTR,A ;Write ACC data to XDATA pointed to by DPTR1,
;DPTR1 is automatically decremented after completion and DPTR0
;is set to the next target DPTR
DJNZ R7,COPY_NEXT ;
SJMP $ ;

END

22.1.2 Example code 2
Send data from 0100H to 0103H in the expansion RAM to the P0 port in sequence
Assembly code

ORG 0000H
LJMP MAIN

ORG 0100H

MAIN:

MOV SP, #3FH
CLR EA ; close interrupt
MOV TA, #0AAH ; Write DPS write protection trigger command 1
MOV TA, #55H ; Write to DPS write protection trigger command 2
MOV DPS, #00001000B ; DPTR0 increment, enable AU0 alone, and select DPTR0
SETB EA ; Open interrupt
MOV DPTR, #0100H ; Write 0100H to DPTR0
MOVX A, @DPTR ; Read data from the XRAM pointed to by DPTR0 and DPTR0 is automatically incremented after completion
MOV P0, A ; Data output to P0 port
MOVX A, @DPTR ; Read data from the XRAM pointed to by DPTR0 and DPTR0 is automatically incremented after completion
MOV P0, A ; Data output to P0 port
MOVX A, @DPTR ; Read data from the XRAM pointed to by DPTR0 and DPTR0 is automatically incremented after completion
MOV P0, A ; Data output to P0 port
MOVX A, @DPTR ; Read data from the XRAM pointed to by DPTR0 and DPTR0 is automatically incremented after completion
MOV P0, A ; Data output to P0 port
SJMP $ ; Data output to P0 port

END
Appendix A  Application Considerations

A.1  Important notes on EEPROM programming and erase wait times

Table 1(STC8A Series and STC8F Series EEPROM Operation Time Requirements)

<table>
<thead>
<tr>
<th>EEPROM operation</th>
<th>shortest time</th>
<th>longest time</th>
</tr>
</thead>
<tbody>
<tr>
<td>programming</td>
<td>6us</td>
<td>7.5us</td>
</tr>
<tr>
<td>abrasion</td>
<td>4ms</td>
<td>6ms</td>
</tr>
</tbody>
</table>

Table 2(STC8A Series and STC8F Series EEPROM Operation Wait Time Waiting for Parameters)

<table>
<thead>
<tr>
<th>IAP_WT[2:0]</th>
<th>Programming waiting clock</th>
<th>Erasing waiting clocks</th>
<th>Suitable frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1</td>
<td>7 clocks</td>
<td>5000 clocks</td>
<td>1MHz</td>
</tr>
<tr>
<td>1 1 0</td>
<td>14 clocks</td>
<td>10000 clocks</td>
<td>2MHz</td>
</tr>
<tr>
<td>1 0 1</td>
<td>21 clocks</td>
<td>15000 clocks</td>
<td>3MHz</td>
</tr>
<tr>
<td>1 0 0</td>
<td>42 clocks</td>
<td>30000 clocks</td>
<td>6MHz</td>
</tr>
<tr>
<td>0 1 1</td>
<td>84 clocks</td>
<td>60000 clocks</td>
<td>12MHz</td>
</tr>
<tr>
<td>0 1 0</td>
<td>140 clocks</td>
<td>100000 clocks</td>
<td>20MHz</td>
</tr>
<tr>
<td>0 0 1</td>
<td>168 clocks</td>
<td>120000 clocks</td>
<td>24MHz</td>
</tr>
<tr>
<td>0 0 0</td>
<td>301 clocks</td>
<td>215000 clocks</td>
<td>30MHz</td>
</tr>
</tbody>
</table>

The programming and erase wait time of the internal EEPROM of the STC8A series and STC8F series MCUs must meet the requirements in Table 1. The waiting time should not be too short or too long.

The programming wait time must be between 6 μs and 7.5 μs. If the programming wait time is too small (less than the minimum time of 6 μs), the data inside the programmed target memory unit may be unreliable (the data retention period may not reach 25 years). If the waiting time is too long (more than 1.5 times the maximum time of 7.5us, i.e., more than 11.25us), the data written may also be incorrect due to data interference. In order to ensure the waiting time for programming, and after the completion of programming, data reading and comparison verification are performed. If the verification is correct, the data will be correctly programmed.

The erase wait time must be between 4ms and 6ms, and the erase wait time is too small (less than the shortest time 4ms), then the erased target memory sector may not be erased cleanly; if the wait time is too long (greater than the maximum A long time of 1.5 times 6ms (i.e., more than 9ms) will shorten the life of the EEPROM, that is, the erase life of the original 100,000 times may be shortened to 50,000 times.

The waiting time for programming and erasing should be properly selected according to the recommended frequency given in Table 2. If the operating frequency is 12MHz, please set the waiting parameter to 011B according to Table 2. If the actual operating frequency of the CPU is not in Table 2 The list of recommended frequencies needs to be calculated based on the actual frequency and the actual number of waiting clocks in Table 2 to find out the waiting time parameters that satisfy the time requirements of Table 1.
For example: the operating frequency is 4MHz, if you choose to wait for the parameter is 101B, the programming time is 21/4MHz = 5.25us, the erase time is 15000/4MHz = 3.75ms, the time is obviously not enough, so you should choose to wait for the parameter is 100B, then programming The time is 42/4MHz = 10.5us, the erase time is 30000/4MHz = 7.5ms, and the time is between the shortest time and 1.5 times the longest time.

Note: The clock that the EEPROM waits for operation refers to the system clock after the main clock is divided, that is, the actual working clock of the CPU. If the microcontroller uses an internal high-precision IRC, then the EEPROM is waiting for the operating clock to use the ISP download software download frequency after adjustment; If the microcontroller uses an external crystal, the EEPROM waits for the operating clock is the external crystal frequency through the CLKDIV register points After the clock (for example, if the microcontroller uses an external crystal and the frequency of the external crystal is 24MHz and the value of the CLKDIV register is set to 4, the clock frequency of the EEPROM waiting for the operation is 24MHz/4 = 6MHz. At this time, the waiting parameter should be 100B, but can't choose 001B).

The following table shows the wrong parts of the previous version.

<table>
<thead>
<tr>
<th>IAP_WT[2:0]</th>
<th>Read byte (2 clocks)</th>
<th>Write Bytes (about 55us)</th>
<th>Erase sector (about 21ms)</th>
<th>clock frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1</td>
<td>2 clocks</td>
<td>55 clocks</td>
<td>21012 clocks</td>
<td>1MHz</td>
</tr>
<tr>
<td>1 1 0</td>
<td>2 clocks</td>
<td>110 clocks</td>
<td>42024 clocks</td>
<td>2MHz</td>
</tr>
<tr>
<td>1 0 1</td>
<td>2 clocks</td>
<td>465 clocks</td>
<td>63036 clocks</td>
<td>3MHz</td>
</tr>
<tr>
<td>1 0 0</td>
<td>2 clocks</td>
<td>330 clocks</td>
<td>126072 clocks</td>
<td>6MHz</td>
</tr>
<tr>
<td>0 1 1</td>
<td>2 clocks</td>
<td>660 clocks</td>
<td>252144 clocks</td>
<td>12MHz</td>
</tr>
<tr>
<td>0 1 0</td>
<td>2 clocks</td>
<td>1100 clocks</td>
<td>420240 clocks</td>
<td>20MHz</td>
</tr>
<tr>
<td>0 0 1</td>
<td>2 clocks</td>
<td>1320 clocks</td>
<td>504288 clocks</td>
<td>24MHz</td>
</tr>
<tr>
<td>0 0 0</td>
<td>2 clocks</td>
<td>1760 clocks</td>
<td>672384 clocks</td>
<td>30MHz</td>
</tr>
</tbody>
</table>

The following table is correct parameters after modifying the previous error.

<table>
<thead>
<tr>
<th>IAP_WT[2:0]</th>
<th>Read byte (2 clocks)</th>
<th>Write Bytes (about 6～7.5us)</th>
<th>Erase sector (about 4～6ms)</th>
<th>clock frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1</td>
<td>2 clocks</td>
<td>7 clocks</td>
<td>5000 clocks</td>
<td>1MHz</td>
</tr>
<tr>
<td>1 1 0</td>
<td>2 clocks</td>
<td>14 clocks</td>
<td>10000 clocks</td>
<td>2MHz</td>
</tr>
<tr>
<td>1 0 1</td>
<td>2 clocks</td>
<td>21 clocks</td>
<td>15000 clocks</td>
<td>3MHz</td>
</tr>
<tr>
<td>1 0 0</td>
<td>2 clocks</td>
<td>42 clocks</td>
<td>30000 clocks</td>
<td>6MHz</td>
</tr>
<tr>
<td>0 1 1</td>
<td>2 clocks</td>
<td>84 clocks</td>
<td>60000 clocks</td>
<td>12MHz</td>
</tr>
<tr>
<td>0 1 0</td>
<td>2 clocks</td>
<td>140 clocks</td>
<td>100000 clocks</td>
<td>20MHz</td>
</tr>
<tr>
<td>0 0 1</td>
<td>2 clocks</td>
<td>168 clocks</td>
<td>120000 clocks</td>
<td>24MHz</td>
</tr>
<tr>
<td>0 0 0</td>
<td>2 clocks</td>
<td>301 clocks</td>
<td>215000 clocks</td>
<td>30MHz</td>
</tr>
</tbody>
</table>

A.2 STC8F2K64S4 Series Application Notes

A.2.1 Important description of STC8F2K64S4 Series D Chip

When all serial ports (including serial port 1, serial port 2, serial port 3, and serial port 4) send serial port data to the serial port sender, the following settings must be made on the
sender port: (One of three methods is optional)

A.2.1.1 Set the I/O port to standard bi-directional mode and turn on the internal pull-up resistor

A.2.1.2 Set I/O port to standard bi-directional port mode and connect 3~10K pull-up resistor

A.2.1.3 Set the I/O port to push-pull mode

A.2.2 Important explanation of STC8F2K64S4 Series D Chip 2

When setting the ninth bit (TB8) of the transmission data in Mode 2 and Mode 3 of the serial port 1, it is necessary to set it twice in succession to be valid. Serial port 2, serial port 3, and serial port 4 do not have this problem.

A.2.3 STC8F2K64S4 Series C Edition Chip Important Notes

A.2.3.1 When all the serial ports (including serial port 1, serial port 2, serial port 3, and serial port 4) send serial port data, the send ports need to be set to open-drain mode and open internal pull-up resistors or external 3 to 10K. Pull-up resistor.

A.2.3.2 All I/O ports that are input only, it is recommended to set it to high impedance / input mode only, and open the internal pull-up resistor / 4.2K, or pull, can also be used on the weak traditional 8051 Pull mode to read the external state, as long as the external is set to "1" state, it can be used as input, but the new 8051 has a better high impedance / input mode only.

A.2.3.3 For all I/O ports that are output only, it is recommended that they be set to open-drain mode and open the internal pull-up resistor/4.2K, or an external pull-up resistor of 5 to 10K.

A.2.3.3.1 A: To output “1” externally, just set “1” externally. At this time, the internal pull-up resistor/4.2K is already open, or the external pull-up resistor is connected to 5~10K;

A.2.3.3.2 B: To output “0” externally, just clear “0” externally, and then turn off the internal pull-up resistor to reduce power consumption<5V/4.2K = 1.2mA, 3.3V/4.2K = 0.78mA>

A.2.3.4 For all I/O ports that are both input and output, it is recommended to set it to open-drain mode and open the internal pull-up resistor/4.2K, or an external pull-up resistor of 5 to 10K.

A.2.3.4.1 A: As an input, it is necessary to externally output "1" status. At this time, an internal pull-up resistor/4.2K is already turned on, or an external pull-up resistor of 5 to 10K is connected, and the 8051 P0 port is used;

A.2.3.4.2 B: To output “1” externally, just set “1” externally. At this time, the internal pull-up resistor/4.2K is already open or the external pull-up resistor is connected to 5~10K;

A.2.3.4.3 C: To output “0” externally, it is only necessary to clear “0” externally. At this time, the internal pull-up resistor can be turned off again to reduce power consumption<5V/4.2K = 1.2mA, 3.3V/4.2K = 0.78mA>

Open P0 port internal pull-up 4.2K resistor register address, P0PU, 0xFE10
Open P1 port internal pull-up 4.2K resistor register address, P1PU, 0xFE11
Open P2 port internal pull-up 4.2K resistor register address, P2PU, 0xFE12
Open P3 port internal pull-up 4.2K resistor register address, P3PU, 0xFE13
Open P4 port internal pull-up 4.2K resistor register address, P4PU, 0xFE14
Open P5 port internal pull-up 4.2K resistor register address, P5PU, 0xFE15
Open P6 port internal pull-up 4.2K resistor register address, P6PU, 0xFE16
Open P7 port internal pull-up 4.2K resistor register address, P7PU, 0xFE17

// The following special function registers are located in the expansion RAM area
// To access these registers, first set P_SW2's BIT7 to 1 to read and write normally.
#define P0PU (*(unsigned char volatile xdata *)0xFE10)
#define P1PU (*(unsigned char volatile xdata *)0xFE11)
#define P2PU (*(unsigned char volatile xdata *)0xFE12)
#define P3PU (*(unsigned char volatile xdata *)0xFE13)
#define P4PU (*(unsigned char volatile xdata *)0xFE14)
#define P5PU (*(unsigned char volatile xdata *)0xFE15)
#define P6PU (*(unsigned char volatile xdata *)0xFE16)
#define P7PU (*(unsigned char volatile xdata *)0xFE17)

A.2.4 STC8F2K64S4-LQFP44/LQFP32 version B chip, sample delivery, samples have the following issues: (All problems will be corrected in the C version of the chip)

======Serial port receiving requires 2 stop bits (including serial port 1, serial port 2, serial port 3, and serial port 4). How to solve in the system, if the sender is not an STC SCM, such as a 32-bit CPU/GPU/DSP, their UART transmission stops. Bits often have a 1-bit/1.5-bit/2-bit selection. The 2-bit stop bit can be selected directly. If the STC is a mass production type single-chip microcomputer, there is only one stop bit after the completion of the transmission and it takes a waiting bit time to wait. Send it again, but this version of the STC8F2K64S4 (version B) also makes a fixed 2 stop bits when sending, and it will change back to a stop bit in the next version.

======When the serial port 1 uses the timer 1 working in mode 2 as the baud rate generator of the serial port, the SMOD (PCON.7) bit must be set, that is, the baud rate must be doubled to make the serial port 1 work normally, otherwise the baud rate Incorrect. If Timer 2 or Timer 1 operating in Mode 0 is used as the serial baud rate generator, this is not a problem.

======When the user uses the special function register (XSFR) of the extended RAM area, the data is also written to the last 512-byte area of 2K bytes of the internal expansion RAM. If the user does not have external SRAM, the EXRAM can be set before accessing the XSFR. 1. After the completion of the access, set EXTRAM to 0 so that the XSFR can be correctly accessed without affecting the use of the internal expansion RAM.

======For STC8F2K series chips with firmware version 7.3.5U and earlier, when using the emulation function, the internal expansion RAM can only use 1K (0000H ~ 03FFH), that is, the simulation reserved area is (0400H ~ 07FFH), and the firmware version is For the STC8F2K series chips with 7.3.6U and newer firmware versions, when using the emulation function, the internal expansion RAM can use 1.25K (0000H ~ 04FFH), that is, the simulation reserved area is (0500H ~ 07FFH).

A.3 STC8F2K64S2 Series Application Notes

1. Important description of STC8F2K64S2 Series D Chip 1

When the serial port sending end of all serial ports (including serial port 1, serial port 2, serial port 3, and serial port 4) sends serial port data, the following settings must be made on the sending port:
(one of three modes is optional).

a. Set the I/O port to standard bi-directional mode and turn on the internal pull-up resistor

b. Set I/O port to standard bi-directional port mode and connect 3–10K pull-up resistor

c. Set the I/O port to push-pull mode

2. Important explanation of STC8F2K64S2 Series D Chip 2

When setting the ninth bit (TB8) of the transmission data in Mode 2 and Mode 3 of the serial port 1, it is necessary to set it twice in succession to be valid. Serial 2, Serial 3, and Serial 4 None

3. Important explanation of STC8F2K64S2 Series C Chip

a. When the serial port sender of all serial ports (including serial port 1, serial port 2, serial port 3, and serial port 4) sends serial port data, the transmit port requires software to be set to open-drain mode and turn on the internal pull-up resistor or an external 3 to 10K pull-up resistor.

b. All I/O ports that are input only, it is recommended to set it to high impedance / input mode only, and open the internal pull-up resistor / 4.2K, or pull, can also be used on the weak traditional 8051 Pull mode to read the external state, as long as the external is set to "1" state, it can be used as input, but the new 8051 has a better high impedance / input mode mode only.

c. For all I/O ports that are output only, it is recommended that they be set to open-drain mode and open the internal pull-up resistor/4.2K, or an external pull-up resistor of 5 to 10K.

1. A: To output “1” externally, just set “1” externally. At this time, the internal pull-up resistor/4.2K is already open or the external pull-up resistor is connected to 5–10K;

2. B: To output “0” externally, just clear “0” externally, and then turn off the internal pull-up resistor to reduce power consumption. <5V/4.2K = 1.2mA, 3.3V/4.2K = 0.78mA>

d. For all I/O ports that are both input and output, it is recommended to set it to open-drain mode and open the internal pull-up resistor/4.2K, or an external pull-up resistor of 5 to 10K.

1. A: As an input, it is necessary to externally output "1" status. At this time, an internal pull-up resistor/4.2K is already turned on, or an external pull-up resistor of 5 to 10K is connected, and the 8051 P0 port is used;

2. B: To output “1” externally, just set “1” externally. At this time, the internal pull-up resistor/4.2K is already open, or the external pull-up resistor is connected to 5–10K;

3. C: To output “0” externally, just clear “0” externally, and then turn off the internal pull-up resistor to reduce power consumption. <5V/4.2K = 1.2mA, 3.3V/4.2K = 0.78mA>

Open P0 port internal pull-up 4.2K resistor register address, P0PU, 0xFE10
Open P1 port internal pull-up 4.2K resistor register address, P1PU, 0xFE11
Open P2 port internal pull-up 4.2K resistor register address, P2PU, 0xFE12
Open P3 port internal pull-up 4.2K resistor register address, P3PU, 0xFE13
Open P4 port internal pull-up 4.2K resistor register address, P4PU, 0xFE14
Open P5 port internal pull-up 4.2K resistor register address, P5PU, 0xFE15
Open P6 port internal pull-up 4.2K resistor register address, P6PU, 0xFE16
Open P7 port internal pull-up 4.2K resistor register address, P7PU, 0xFE17

//The following special function registers are located in the expansion RAM area
//To access these registers, first set P_SW2’s BIT7 to 1 to read and write normally.
#define P0PU (*(unsigned char volatile xdata *)0xFE10)
A.4 STC8A8K64S4A12 Series Application Notes

1. Important description of STC8A8K64S4A12 Series F Chip 1
When all serial ports (including serial port 1, serial port 2, serial port 3, and serial port 4) send serial port data to the serial port sender, the following settings must be made on the sender port: (One of three methods is optional)

a. Set the I/O port to standard bi-directional mode and turn on the internal pull-up resistor
b. Set I/O port to standard bi-directional port mode and connect 3~10K pull-up resistor
c. Set the I/O port to push-pull mode

2. Important explanation of STC8A8K64S4A12 Series F Chip 2
When setting the ninth bit (TB8) of the transmission data in Mode 2 and Mode 3 of the serial port 1, it is necessary to set it twice in succession to be valid. Serial port 2, serial port 3, and serial port 4 do not have this problem.

3. Important description of TC8A8K64S4A12 Series E Chip
a. All ports on the serial port (including serial port 1, serial port 2, serial port 3, and serial port 4) require software to be set to open-drain mode and turn on internal pull-up resistors or an external 3 to 10K pull-up resistor.
b. All I/O ports that are input only, it is recommended to set it to high impedance / input mode only, and open the internal pull-up resistor / 4.2K, or pull, can also be used on the weak traditional 8051 Pull mode to read the external state, as long as the external is set to "1" state, it can be used as input, but the new 8051 has a better high impedance / input mode mode only.
c. For all I/O ports that are output only, it is recommended that they be set to open-drain mode and open the internal pull-up resistor/4.2K, or an external pull-up resistor of 5 to 10K.
1. A, To output “1” externally, just set “1” externally. At this time, the internal pull-up resistor/4.2K is already open or the external pull-up resistor is connected to 5~10K.
2. B, To output “0” externally, just clear “0” externally, and then turn off the internal pull-up resistor to reduce power consumption.<5V/4.2K = 1.2mA, 3.3V/4.2K = 0.78mA>
d. All I/O ports to be both input and output. It is recommended to set it to open-drain mode and turn on the newly added pull-up resistor/4.2K, or an external pull-up of 5 to 10K resistance.
1. A, As an input, it is necessary to externally output "1" status. At this time, an internal pull-up resistor/4.2K is already turned on, or an external pull-up resistor of 5 to 10K is connected, and the 8051 P0 port is used.
2. B, To output “1” externally, just set “1” externally. At this time, the internal pull-up resistor/4.2K is already open, or the external pull-up resistor is connected to 5~10K.
3. C, To output “0” externally, it is only necessary to clear “0” externally. At this time,
the internal pull-up resistor can be turned off again to reduce power consumption. \(<5\text{V}/4.2\text{K} = 1.2\text{mA}, 3.3\text{V}/4.2\text{K} = 0.78\text{mA}\> \\

Open P0 port internal pull-up 4.2K resistor register address, P0PU, 0xFE10 \\
Open P1 port internal pull-up 4.2K resistor register address, P1PU, 0xFE11 \\
Open P2 port internal pull-up 4.2K resistor register address, P2PU, 0xFE12 \\
Open P3 port internal pull-up 4.2K resistor register address, P3PU, 0xFE13 \\
Open P4 port internal pull-up 4.2K resistor register address, P4PU, 0xFE14 \\
Open P5 port internal pull-up 4.2K resistor register address, P5PU, 0xFE15 \\
Open P6 port internal pull-up 4.2K resistor register address, P6PU, 0xFE16 \\
Open P7 port internal pull-up 4.2K resistor register address, P7PU, 0xFE17 \\n
//The following special function registers are located in the expansion RAM area. \\
//To access these registers, first set BIT7 of P_SW2 to 1 to read and write normally. \\
#define P0PU (*(unsigned char volatile xdata *)0xfe10) \\
#define P1PU (*(unsigned char volatile xdata *)0xfe11) \\
#define P2PU (*(unsigned char volatile xdata *)0xfe12) \\
#define P3PU (*(unsigned char volatile xdata *)0xfe13) \\
#define P4PU (*(unsigned char volatile xdata *)0xfe14) \\
#define P5PU (*(unsigned char volatile xdata *)0xfe15) \\
#define P6PU (*(unsigned char volatile xdata *)0xfe16) \\
#define P7PU (*(unsigned char volatile xdata *)0xfe17) \\n
e. ADC related problems \\
1. Voltage difference between AVCC and VCC should be less than 0.3V \\
2. The software must set the corresponding ADC conversion port to input high impedance input mode or open drain. \\
3. To read 0, the ADC's conversion speed should use the fastest file. \\
4. When the ADC port is converted, some of the following ports are incorrectly set to high impedance by the IC. The software cannot be used or controlled. Do not use the port that has been mistakenly set as the high impedance input mode. \\
5. If ADC0/Channel 0 is used, the input port is P1.0 and P1.7 is set to high impedance by mistake. It is recommended to leave this port empty in the system. Do not use P1.7. \\
6. If ADC1/Channel 1 is used, the input port is P1.1. P0.0 is set to high impedance by mistake. It is recommended to leave this port empty in the system. Do not use P0.0. \\
7. If ADC2/Channel 2 is used, the input port is P1.2, P0.1 is set to high impedance by mistake. It is recommended to leave this port empty in the system. Do not use P0.1. \\
8. If ADC3/Channel 3 is used, the input port is P1.3. P0.2 is set to high impedance by mistake. It is recommended to leave this port empty in the system. Do not use P0.2. \\
9. If ADC4/Channel 4 is used, the input port is P1.4, and P0.3 is set to high impedance by mistake. It is recommended to leave this port empty in the system. Do not use P0.3. \\
10. If ADC5/Channel 5 is used, the input port is P1.5. P0.4 is set to high impedance by mistake. It is recommended to leave this port empty in the system. Do not use P0.4. \\
11. If ADC6/channel 6 is used, the input port is P1.6, P0.5 is set to high impedance by mistake. It is recommended to leave this port empty in the system. Do not use P0.5.
12. If ADC7/Channel 7 is used, the input port is P1.7, P0.6 is set to high impedance by mistake. It is recommended to leave this port empty in the system. Do not use P0.6.

13. If ADC8/Channel 8 is used, the input port is P0.0. P1.0 is set to high impedance by mistake. It is recommended to leave this port empty in the system. Do not use P1.0.

14. If ADC9/Channel 9 is used, the input port is P0.1, P1.1 is set to high impedance by mistake. It is recommended to leave this port empty in the system. Do not use P1.1.

15. If ADC10/Channel 10 is used, the input port is P0.2, and P1.2 is set to high impedance by mistake. It is recommended to leave this port empty in the system. Do not use P1.2.

16. If ADC11/Channel 11 is used, the input port is P0.3, and P1.3 is set to high impedance by mistake. It is recommended to leave this port empty in the system. Do not use P1.3.

17. If ADC12/Channel 12 is used, the input port is P0.4, P1.4 is set to high impedance by mistake. It is recommended to leave this port empty in the system. Do not use P1.4.

18. If ADC13/Channel 13 is used, the input port is P0.5, P1.5 is set to high impedance by mistake. It is recommended to leave this port empty in the system. Do not use P1.5.

19. If ADC14/Channel 14 is used, the input port is P0.6, and P1.6 is set to high impedance by mistake. It is recommended to leave this port empty in the system. Do not use P1.6.

20. If ADC15/Channel 15 is used, the input is internal Vref/1.344V, and P1.7 is set to high impedance by mistake. It is recommended to leave this port empty in the system. Do not use P1.7.

4. STC8A8K64S4A12-LQFP64S/LQFP48/44 version D chip, sample delivery, samples have the following problems: (All problems will be corrected in the E version of the chip)

======Serial port receiving requires 2 stop bits (including serial port 1, serial port 2, serial port 3, and serial port 4). How to solve in the system, if the sender is not an STC SCM, such as a 32-bit CPU/GPU/DSP, their UART transmission stops. Bits often have a 1-bit/1.5-bit/2-bit selection. The 2-bit stop bit can be selected directly. If the STC is a mass production type single-chip microcomputer, there is only one stop bit after the completion of the transmission and it takes a waiting bit time to wait. Send it again, but this version of the STC8F2K64S4 (version B) also makes a fixed 2 stop bits when sending, and it will change back to a stop bit in the next version.

======When the serial port 1 uses the timer 1 working in mode 2 as the baud rate generator of the serial port, the SMOD (PCON.7) bit must be set, that is, the baud rate must be doubled to make the serial port 1 work normally, otherwise the baud rate Incorrect. If Timer 2 or Timer 1 operating in Mode 0 is used as the serial baud rate generator, this is not a problem.

======When the user uses the special function register (XSFR) of the extended RAM area, the data is also written to the last 512-byte area of 8K bytes of the internal expansion RAM. If the user does not have external SRAM, the EXRAM can be set before accessing the XSFR. 1. After the completion of the access, set EXTRAM to 0 so that the XSFR can be correctly accessed without affecting the use of the internal expansion RAM.

======12-bit 16-channel ADC up to 11.5 bits (ADC7), the ADC7 channel closest to the AGnd pin is the best, followed by ADC6/ADC5/ADC4/ADC3/ADC2/ADC1/ADC0, ADC14/ADC13/ADC12/ADC11/ADC10/ADC9/ADC8, the ADC8 farthest away from the AGnd pin suggests that a 0.047uF - 0.1uF/0.2uF capacitor be used near the ADC input channel to the analog ground AGnd to reject the MCU’s digital power and ground disturbances.

======For STC8A8K series and STC8F8K series chips with firmware version 7.3.5U and earlier, when using the emulation function, the internal expansion RAM can only use 3K (0000H ~ 0FFFH), that is, the simulation reserved area is (0C00H ~ 0FFFH). For STC8A8K series and STC8F8K
series chips whose firmware version is 7.3.6U, when using the emulation function, the internal expansion RAM can use 7.25K (0000H ~ 0CFFH), (1000H ~ 1FFFH), that is, the simulation reserved area is (0D00H ~ 0FFFH). For STC8A8K series and STC8F8K series chips with firmware version 7.3.7U and newer firmware version, when using the emulation function, the internal expansion RAM can use 7.25K (0000H ~ 1CFFH), that is, the simulation reserved area is (1D00H ~ 1FFFH).

A.5  STC8A4K64S2A12 Series Application Notes

1. Important description of STC8A4K64S2A12 Series F Chip 1
   When all serial ports (including serial port 1, serial port 2, serial port 3, and serial port 4) send serial port data to the serial port sender, the following settings must be made on the sender port: (One of three methods is optional)
   a. Set the I/O port to standard bi-directional mode and turn on the internal pull-up resistor
   b. Set I/O port to standard bi-directional port mode and connect 3~10K pull-up resistor
   c. Set the I/O port to push-pull mode

2. Important explanation of STC8A4K64S2A12 Series F Chip 2
   When setting the ninth bit (TB8) of the transmission data in Mode 2 and Mode 3 of the serial port 1, it is necessary to set it twice in succession to be valid. Serial 2, Serial 3, and Serial 4 None
Appendix B STC Guide to the use of simulators

1. Installing the Keil version of the simulation driver

As shown above, first select "Keil simulation settings" page, click "Add MCU model to Keil", in the following
directory selection window appears, locate the Keil installation directory (usually may be "C:\Keil") , "OK"
appears on the right side of the figure below shows the message that the installation was successful. Adding the
header file also installs the STC Monitor51 emulation driver STCMON51.DLL. The driver and header files are
installed in the directory as shown above.
2. Create a project in Keil

If the driver installation of the first step is successful, when selecting the chip model when creating a new project in Keil, there will be a choice of “STC MCU Database”, as shown below

Select a CPU DataBase File

Then select the responding MCU model from the list. Here we select the model of "STC8A8K64S4A12" and click "OK" to complete the selection

Select Device for Target ‘Target 1’...

Add the source code file to the project, as shown below:
Save the project. If it is compiled correctly, you can set the following project.

An additional note:

When you create a C language project and add the startup file "STARTUP.A51" to your project, there is a macro definition named "IDATALEN". It is a macro that defines the size of IDATA. The default value is 128, which is hexadecimal 80H, and it is also the size of IDATA that needs to be initialized to 0 in the startup file.

So when IDATA is defined as 80H, the code in STARTUP.A51 will initialize the RAM of IDATA's 00-7F to 0; also if IDATA is defined as 0FFH, the RAM of 00-FF of IDATA will be initialized to 0.

Although the IDATA size of the STC8 series microcontroller is 256 bytes (DATA of 00-7F and IDATA of 80H-FFH), there is a write ID number and related test parameters in the last 17 bytes of RAM if the user is Need to use this part of the data in the program, you must not define IDATALEN as 256.

3. Project Settings, Select STC Simulation Drive
As shown above, first enter the project settings page, select the "Debug" settings page, the second step to select the right side of the hardware simulation "Use ...", the third step, in the simulation drive drop-down list, select "STC Monitor-51 Driver" Items, then click "Settings" button, enter the following setting screen, set the serial port number and baud rate, the baud rate is generally selected 115200. This setup is complete.

4. Create a simulation chip

Prepare a STC8A series or STC8F series chip and connect it to the computer's serial port via the download
board. Then, as shown above, select the correct chip model, and then enter the “Keil simulation settings” page, click the corresponding model button, when the program is downloaded. After the completion of the simulator is completed.

5. Start simulation

The completed simulation chip is connected to the computer through the serial port. After compiling the project we created earlier to no error, press "Ctrl+F5" to start debugging. If the hardware connection is correct, it will enter a debugging interface similar to the following, and the current simulation driver version number and the version number of the current emulation monitor code firmware are displayed in the command output window. A maximum of 20 breakpoints are allowed before the number of breakpoints is set (in theory, any number can be set, but setting too many breakpoints will affect the debugging speed).

Appendix C STC-USB Driver installation instructions

- Windows XP installation method
Open STC-ISP download software version V6.79 (or later). The download software will automatically copy the driver files to the relevant system directory.
Insert the USB device, the system automatically pops up the following dialog box after finding the device, select the "No, not for now" item.
In the following dialog box, select the "Install the software automatically (Recommended)" item.
In the following dialog box that pops up, select the "still continue" button.

![Hardware installation dialog box](image-url)
The system will automatically install the driver, as shown below.
The following dialog box appears indicating that the driver installation is complete.

![Dialog box showing completed installation]

完成找到新硬件向导

该向导已经完成了下列设备的软件安装。

STC USB Low Speed Writer

要关闭向导，请单击“完成”。
At this time, the serial number list in the previously opened STC-ISP download software will automatically select the inserted USB device and display the device name as “STC USB Writer (USB1)”, as shown below:
Windows 7(32 bit) installation method

Open STC-ISP download software version V6.79 (or later). The download software will automatically copy the driver files to the relevant system directory.
Insert the USB device and the system will automatically install the driver after it finds the device. After the installation is complete, the following prompt box will appear.
At this point, the serial number list in the previously opened STC-ISP download software will automatically select the inserted USB device and display the device name as "STC USB Writer (USB1)", as shown below:

![STC-ISP Software Screenshot]

Note: If Windows 7 does not automatically install the driver, please refer to the installation method for Windows 8 (32-bit) for how to install the driver.
● Windows 7(64 bit)installation method

Because the Windows 7 64-bit operating system is in the default state, drivers that do not have digital signatures cannot be installed successfully. Therefore, before installing the STC-USB driver, you need to follow the steps below to temporarily skip the digital signature and install it successfully.

First restart the computer and keep pressing F8 until the following splash screen appears

Select "Disable driver signature enforcement". After you start it, you can turn off the digital signature verification function temporarily.
Insert the USB device and open the Device Manager. Locate the USB device with a yellow exclamation mark in the device list. From the right-click menu of the device, select Update Driver Software.
In the following dialog box, select "Browse my computer for driver software."
Click the "Browse" button in the dialog below to find the directory where the previous STC-USB driver was stored. (For example, the previous example directory is "D:\STC-USB", and the user locates the path to the actual decompression directory.)
When the driver starts to install, the following dialog box will pop up and select "Always install this driver software".
Next, the system will automatically install the driver, as shown below.
The following dialog box appears indicating that the driver installation is complete.

![Driver installation complete dialog box](image-url)
At this point in the device manager, the device with the yellow exclamation point before it will now display the device name "STC USB Low Speed Writer."
The serial port number list in the previously opened STC-ISP download software will automatically select the inserted USB device and display the device name as “STC USB Writer (USB1)”, as shown below:
● Windows 8(32 bit) Installation method

Open STC-ISP download software for V6.79 (or later). (Due to permission reasons, downloading software in Windows 8 will not copy the driver files to the relevant system directory, requiring the user to manually install. First, download "stc-isp-15xx-v6.79.zip" (or later) from STC official website. After downloading and decompressing it to the local disk, the STC-USB driver file will also be extracted to the current unzipped directory “STC-USB Driver”. (For example, if you unzip the downloaded compressed file "stc-isp-15xx-v6.79.zip" to "F: \", the STC-USB driver is in the "F:\STC-USB Driver" directory.)
Insert the USB device and open the Device Manager. Locate the USB device with a yellow exclamation mark in the device list. From the right-click menu of the device, select Update Driver Software.
In the following dialog box, select "Browse my computer for driver software."
Click the "Browse" button in the dialog below to find the directory where the previous STC-USB driver was stored (For example, the previous example directory is "F:\STC-USB Driver". The user will locate the path to the actual decompressed directory.)
When the driver starts to install, the following dialog box will pop up and select "Always install this driver software".

![Driver install dialog box](image)

- **Windows 安全**

- **Windows 无法验证此驱动程序软件的发布者**

  - 不安装此驱动程序软件(N)
    应查看制造商的网站，获得设备的更新驱动程序软件。

  - 始终安装此驱动程序软件(Y)
    仅安装来自制造商网站或光盘的驱动程序软件。其他来源的未签名软件可能会损坏你的计算机或窃取信息。

  - 查看详细信息(D)
Next, the system will automatically install the driver, as shown below
The following dialog box appears indicating that the driver installation is complete.
At this point in the device manager, the device with the yellow exclamation point before it will now display the device name "STC USB Low Speed Writer."
The serial port number list in the previously opened STC-ISP download software will automatically select the inserted USB device and display the device name as “STC USB Writer (USB1)”, as shown below:
● Windows 8(64 bit) installation method

Because the Windows 8 64-bit operating system is in the default state, drivers that do not have digital signatures cannot be installed successfully. Therefore, before installing the STC-USB driver, you need to follow the steps below to temporarily skip the digital signature and install it successfully.

First move the mouse to the lower right corner of the screen and select the "Settings" button.
Then select "change computer settings" item in the setting interface.
In the PC settings, select the "Start Now" button under the "Advanced startup" item in the "General" property page.
In the following interface, select the "Troubleshoot" item.
Then select "Advanced Options" in "Troubleshooting."
In the "Advanced Options" interface below, select "Startup Settings".
In the following "Startup Settings" interface, click the "Restart" button to restart the computer.
After the computer restarts, it will automatically enter the "Startup Settings" interface as shown in the figure below. Press the number key "7" or press the function key "F7" to select "Disable Driver Force Signature" to start.

After booting to Windows 8, the installation of the driver can be completed according to the installation method of Windows 8 (32-bit).
Windows 8.1(64 bit) installation method

Windows 8.1 and Windows 8 have different methods for entering the advanced boot menu, and are specifically described here.

First move the mouse to the lower right corner of the screen and select the "Settings" button.
Then select "change computer settings" item in the setting interface.
In the computer settings, select "Update and Restore" (this is different from Windows 8 and Windows 8 selects "General").
Select the "Restore" property page on the Update and Recovery page and click the "Start Now" button under the "Advanced startup" item.
The following operations are the same as those of Window 8. In the following interface, select the "Troubleshoot" item.
Then select "Advanced Options" in "Troubleshooting."
In the "Advanced Options" interface below, select "Startup Settings".
In the following "Startup Settings" interface, click the "Restart" button to restart the computer.
After the computer restarts, it will automatically enter the "Startup Settings" interface as shown in the figure below. Press the number key "7" or press the function key "F7" to select "Disable Driver Force Signature" to start.

After booting to Windows 8, the installation of the driver can be completed according to the installation method of Windows 8 (32-bit).
# Appendix D Electrical Character

## Absolute maximum rated value

<table>
<thead>
<tr>
<th>parameter</th>
<th>Minimum</th>
<th>Maximum</th>
<th>unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage temperature</td>
<td>-55</td>
<td>+125</td>
<td>℃</td>
</tr>
<tr>
<td>working temperature</td>
<td>-40</td>
<td>+85</td>
<td>℃</td>
</tr>
<tr>
<td>working voltage</td>
<td>2.0</td>
<td>5.5</td>
<td>V</td>
</tr>
<tr>
<td>VDD voltage to earth</td>
<td>-0.3</td>
<td>+5.5</td>
<td>V</td>
</tr>
<tr>
<td>IO voltage to earth</td>
<td>-0.3</td>
<td>VDD+0.3</td>
<td>V</td>
</tr>
</tbody>
</table>

## direct-current characteristic (VSS=0V，VDD=5.0V，Test temperature=25℃)(STC8F2K series)

<table>
<thead>
<tr>
<th>Labeling</th>
<th>parameter</th>
<th>range</th>
<th>testing environment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Minimu m</td>
<td>Typical value</td>
</tr>
<tr>
<td>I&lt;sub&gt;PD&lt;/sub&gt;</td>
<td>Power-down mode current (SCC = 1)</td>
<td>-</td>
<td>0.08</td>
</tr>
<tr>
<td></td>
<td>Power-down mode current (SCC = 0)</td>
<td>-</td>
<td>1.5</td>
</tr>
<tr>
<td>I&lt;sub&gt;WKT&lt;/sub&gt;</td>
<td>Power-down wake-up timer</td>
<td>-</td>
<td>5</td>
</tr>
<tr>
<td>I&lt;sub&gt;LVD&lt;/sub&gt;</td>
<td>Low voltage detection module</td>
<td>-</td>
<td>260</td>
</tr>
<tr>
<td>I&lt;sub&gt;IDL&lt;/sub&gt;</td>
<td>Idle mode current(6MHz)</td>
<td>-</td>
<td>1.3</td>
</tr>
<tr>
<td></td>
<td>Idle mode current(11.0592MHz)</td>
<td>-</td>
<td>1.7</td>
</tr>
<tr>
<td></td>
<td>Idle mode current(20MHz)</td>
<td>-</td>
<td>2.3</td>
</tr>
<tr>
<td></td>
<td>Idle mode current(22.1184MHz)</td>
<td>-</td>
<td>2.5</td>
</tr>
<tr>
<td></td>
<td>Idle mode current(24MHz)</td>
<td>-</td>
<td>2.6</td>
</tr>
<tr>
<td></td>
<td>Idle mode current (internal 32KHz)</td>
<td>-</td>
<td>850</td>
</tr>
<tr>
<td>I&lt;sub&gt;NOR&lt;/sub&gt;</td>
<td>Normal mode current(6MHz)</td>
<td>-</td>
<td>2.7</td>
</tr>
<tr>
<td></td>
<td>Normal mode current(11.0592MHz)</td>
<td>-</td>
<td>3.8</td>
</tr>
<tr>
<td></td>
<td>Normal mode current(20MHz)</td>
<td>-</td>
<td>5.9</td>
</tr>
<tr>
<td></td>
<td>Normal mode current(22.1184MHz)</td>
<td>-</td>
<td>6.3</td>
</tr>
<tr>
<td></td>
<td>Normal mode current(24MHz)</td>
<td>-</td>
<td>6.5</td>
</tr>
<tr>
<td></td>
<td>Normal mode current (internal 32KHz)</td>
<td>-</td>
<td>950</td>
</tr>
<tr>
<td>I&lt;sub&gt;CC&lt;/sub&gt;</td>
<td>Normal operating mode current</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>V&lt;sub&gt;IL1&lt;/sub&gt;</td>
<td>Input low level</td>
<td>-</td>
<td>1.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-</td>
<td>1.5</td>
</tr>
</tbody>
</table>

Nantong guoxin Microelectronics Co., Ltd.  
Tel: 0513-5501 2928/2929/2966  
Fax: 0513-5501 2926/2956/2947
### Table 1: Electrical Characteristics

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
<th>Testing Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Power-down mode current (SCC = 1)</strong></td>
<td>-</td>
<td>0.08</td>
<td>-</td>
<td>μA</td>
<td>5.0V</td>
</tr>
<tr>
<td>Power-down mode current (SCC = 0)</td>
<td>-</td>
<td>1.6</td>
<td>-</td>
<td>μA</td>
<td>5.0V</td>
</tr>
<tr>
<td><strong>Power-down wake-up timer</strong></td>
<td>-</td>
<td>5</td>
<td>-</td>
<td>μA</td>
<td>5.0V</td>
</tr>
<tr>
<td>Low voltage detection module</td>
<td>-</td>
<td>260</td>
<td>-</td>
<td>μA</td>
<td>5.0V</td>
</tr>
<tr>
<td>Idle mode current (6MHz)</td>
<td>-</td>
<td>1.5</td>
<td>-</td>
<td>mA</td>
<td>5.0V</td>
</tr>
<tr>
<td>Idle mode current (11.0592MHz)</td>
<td></td>
<td>1.9</td>
<td>-</td>
<td>mA</td>
<td>5.0V</td>
</tr>
<tr>
<td>Idle mode current (20MHz)</td>
<td></td>
<td>2.7</td>
<td>-</td>
<td>mA</td>
<td>5.0V</td>
</tr>
<tr>
<td>Idle mode current (22.1184MHz)</td>
<td>-</td>
<td>3.0</td>
<td>-</td>
<td>mA</td>
<td>5.0V</td>
</tr>
<tr>
<td>Idle mode current (24MHz)</td>
<td>-</td>
<td>3.2</td>
<td>-</td>
<td>mA</td>
<td>5.0V</td>
</tr>
<tr>
<td>Idle mode current (internal 32KHz)</td>
<td>-</td>
<td>890</td>
<td>-</td>
<td>μA</td>
<td>5.0V</td>
</tr>
<tr>
<td>Normal mode current (6MHz)</td>
<td>-</td>
<td>3.0</td>
<td>-</td>
<td>mA</td>
<td>5.0V</td>
</tr>
<tr>
<td>Normal mode current (11.0592MHz)</td>
<td>-</td>
<td>4.3</td>
<td>-</td>
<td>mA</td>
<td>5.0V</td>
</tr>
<tr>
<td>Normal mode current (20MHz)</td>
<td>-</td>
<td>6.8</td>
<td>-</td>
<td>mA</td>
<td>5.0V</td>
</tr>
<tr>
<td>Normal mode current (22.1184MHz)</td>
<td>-</td>
<td>7.4</td>
<td>-</td>
<td>mA</td>
<td>5.0V</td>
</tr>
<tr>
<td>Normal mode current (24MHz)</td>
<td>-</td>
<td>7.8</td>
<td>-</td>
<td>mA</td>
<td>5.0V</td>
</tr>
</tbody>
</table>

#### Direct-current Characteristic

VSS=0V, VDD=5.0V, Test temperature=25°C (STC8A8K series)

**Input High Level (ordinary I/O)**
- **V_{IH1}**: Input high level
  - 1.7 V (Turn off Schmidt trigger)
  - 1.6 V (Open Schmidt trigger)

**Input High Level (Reset foot)**
- **V_{IH2}**: Input high level
  - 1.6 V (Turn off Schmidt trigger)

**Output Low Sink Current**
- **I_{OL1}**: Output low sink current
  - - 20 mA (5.0V, Port voltage 0.45V)

**Output High Level Current**
- **I_{OH1}**: Output high level current (Bidirectional mode)
  - 200 mA (5.0V)
- **I_{OH2}**: Output high level current (Push-pull mode)
  - - 20 mA (5.0V, Port voltage 2.4V)

**Logic 0 Input Current**
- **I_{IL}**: Logic 0 input current
  - - 50 μA (5.0V, Port voltage 0V)

**Transfer Current from Logic 1 to 0**
- **I_{TL}**: Transfer current from logic 1 to 0
  - 100 μA (5.0V, Port voltage 2.0V)

**IO Pull-up Resistor**
- **R_{PU}**: IO pull-up resistor
  - 4.1 KΩ (5.0V)
- **R_{PDU}**: IO pull-up resistor
  - 5.8 KΩ (3.3V)

**IO Pull-up Resistor**
- **R_{PU}**: IO pull-up resistor
  - 4.1 KΩ (5.0V)
- **R_{PDU}**: IO pull-up resistor
  - 5.8 KΩ (3.3V)
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range</th>
<th>Units</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>I&lt;sub&gt;CC&lt;/sub&gt;</td>
<td>0.45 mA</td>
<td>mA</td>
<td>Normal operating mode current</td>
</tr>
<tr>
<td>V&lt;sub&gt;IL&lt;/sub&gt;</td>
<td>1.4 V</td>
<td>V</td>
<td>Input low level</td>
</tr>
<tr>
<td>V&lt;sub&gt;IL&lt;/sub&gt;</td>
<td>1.5 V</td>
<td>V</td>
<td>Open Schmidt trigger</td>
</tr>
<tr>
<td>V&lt;sub&gt;IL&lt;/sub&gt;</td>
<td>2.2 V</td>
<td>V</td>
<td>Turn off Schmidt trigger</td>
</tr>
<tr>
<td>V&lt;sub&gt;OH&lt;/sub&gt;</td>
<td>2.2 V</td>
<td>V</td>
<td>Input high level (ordinary I/O)</td>
</tr>
<tr>
<td>V&lt;sub&gt;OH&lt;/sub&gt;</td>
<td>1.6 V</td>
<td>V</td>
<td>Open Schmidt trigger</td>
</tr>
<tr>
<td>V&lt;sub&gt;H&lt;/sub&gt;</td>
<td>1.4 V</td>
<td>V</td>
<td>Turn off Schmidt trigger</td>
</tr>
<tr>
<td>V&lt;sub&gt;H&lt;/sub&gt;</td>
<td>2.2 V</td>
<td>V</td>
<td>Input high level (Reset foot)</td>
</tr>
<tr>
<td>I&lt;sub&gt;OL&lt;/sub&gt;</td>
<td>20 mA</td>
<td>mA</td>
<td>Output low sink current</td>
</tr>
<tr>
<td>I&lt;sub&gt;OH&lt;/sub&gt;</td>
<td>200 uA</td>
<td>uA</td>
<td>Output high level current (Bidirectional mode)</td>
</tr>
<tr>
<td>I&lt;sub&gt;OH&lt;/sub&gt;</td>
<td>20 uA</td>
<td>uA</td>
<td>Output high level current (Push-pull mode)</td>
</tr>
<tr>
<td>I&lt;sub&gt;L&lt;/sub&gt;</td>
<td>50 uA</td>
<td>uA</td>
<td>Logic 0 input current</td>
</tr>
<tr>
<td>I&lt;sub&gt;T&lt;/sub&gt;</td>
<td>100 uA</td>
<td>uA</td>
<td>Transfer current from logic 1 to 0</td>
</tr>
<tr>
<td>R&lt;sub&gt;PU&lt;/sub&gt;</td>
<td>4.1 KΩ</td>
<td>KΩ</td>
<td>IO pull-up resistor</td>
</tr>
<tr>
<td>R&lt;sub&gt;PU&lt;/sub&gt;</td>
<td>5.8 KΩ</td>
<td>KΩ</td>
<td></td>
</tr>
</tbody>
</table>

Internal IRC temperature drift characteristics (reference temperature 25°C)

<table>
<thead>
<tr>
<th>Temperature</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>-40°C ~ 85°C</td>
<td>-1.8% ~ +0.8%</td>
</tr>
<tr>
<td>-20°C ~ 65°C</td>
<td>-1.0% ~ +0.5%</td>
</tr>
</tbody>
</table>
Appendix E Update Log

2018/3/20
1. Increase the sample program for reading important test parameters from ROM and RAM (8.3)
   a) Reading Bandgap voltage values from memory
   b) Read globally unique ID number from memory
   c) Read 32K Power-Down Wake-up Timer Frequency from Memory
   d) Reading IRC Parameters from Memory Manually Set Internal IRC Frequency

2018/3/13
1. Added pin diagram for STC8F1K08S2 model SOP16

2018/3/6
1. Increase the use of comparators for power-down detection example program (15.3.3)
2. Increase the use of comparators to detect operating voltage (battery voltage) example program (15.3.4)
3. Added example program for detecting operating voltage (battery voltage) using LVD (Low Voltage Detection) inside the chip (7.4.13)
4. Added STC8F1K08S2 feature descriptions and pinouts
5. Add STC8H1K08S2A10 features and pinouts

2018/1/30
1. Remove the use of "using, at_" in the sample program
2. Explain the RSTV function of P2.0
3. Indicates the voltage value of the internal reference voltage
4. Adjust part of reference circuit diagram

2017/11/27
1. Increase the power-down wake-up IO port and increase the power-down wakeup sample program

2017/11/7
1. Corrects the clock output by the main clock divided by the system clock divided by CLKDIV (The previous error description is that the main clock divided by the frequency output clock is the main clock before the CLKDIV frequency division)
2. STC8 Series Comparator Interrupt Sets 4-Level Interrupt Priority (The previous version was incorrectly described)
3. Increase the interrupt system block diagram
4. Increase the ADC 16th channel test sample code for external battery voltage testing (Chapter 17.3.4)
5. Added sample code for reading important parameters inside STC8 series MCUs (Chapter 8.3)

- 2017/11/2
  1. Added SOP16 pin diagram for STC8H1K08S2A10 series and STC8H1K08S2 series
  2. Correct EEPROM programming and erase clocks
  3. Important instructions for increasing the programming and erase latency settings of STC8F series and STC8A series EEPROMs

- 2017/10/31
  1. Update chip package diagram
  2. Update ADC Typical Application Circuit Diagram

- 2017/8/9
  1. Corrected the number of IO ports of the STC8A4K64S2A12 series to 59
  2. Update selection price list

- 2017/8/1
  1. Update application considerations

- 2017/7/27
  1. Update selection and Price list

- 2017/7/12
  1. Adding an important note to the STC8F2K64S4 Series D chip
  2. STC8F2K64S4 Series D Chip starts to send samples (optional package LQFP44N LQFP32U PDIP40)

- 2017/7/3
  1. STC8F2K Series Add TSSOP20 and SOP16 Pins

- 2017/6/30
  1. Increase the package size of QFN32, TSSOP20, SOP16
  2. Important notes for updating the revision
2017/5/17
1. Add STC8F2K64S2 Series
2. Add STC8A4K64S2A12 Series
3. Add STC8F1K08S2A10 Series
4. Add STC8F1K08S2 Series

2017/5/12
1. Added description of auxiliary commands for I2C master mode (The auxiliary commands are valid only for the C version of the STC8F2K64S4 series and the E version of the STC8A8K64S4A12 series.)
2. Added reference circuit diagram for ISP download using U8W, U8-Mini, and PL2303
3. Added description of port internal pull-up resistor and Schmitt trigger control
4. Updated STC8 series microcontroller selection and reference price

2017/3/1
1. Add chapters for important instructions and chapters used by the emulator

2016/12/22
1. STC8F2K64S4 Series C Chips without PCA/CCP/PWM Capability
2. STC8A8K64S4A12 Series B version of the chip has the following points need attention
   e) When the serial port 1 uses the mode 2 of the timer 1 as the baud rate generator, the SMOD (PCON.7) bit must be enabled.
   f) All serial ports (including serial port 1, serial port 2, serial port 3, and serial port 4) must have two stop bits when receiving data; otherwise, data loss may occur.
   g) When accessing the internal expansion XSFR, the EXTRAM (AUXR.1) bit needs to be set, otherwise the last 512 bytes of the internal XRAM are affected when the XSFR is written.

2016/11/22
1. Add sample programs
2. STC8F2K64S4 Series B version of the chip has the following points need attention
   a) When the serial port 1 uses the mode 2 of the timer 1 as the baud rate generator, the SMOD (PCON.7) bit must be enabled.
   b) All serial ports (including serial port 1, serial port 2, serial port 3, and serial port 4) must have two stop bits when receiving data; otherwise, data loss may occur.
   c) When accessing the internal expansion XSFR, the EXTRAM (AUXR.1) bit needs to be set, otherwise it will affect the last 512 bytes of the internal XRAM when writing XSFR.
- 2016/9/13 -
1. Increase CAN bus function (Define CAN pins, CAN interrupt related SFRs)
   CAN bus functional SFRs need to be defined after discussion
2. Add IP3 and IP3H to set the interrupt priority of serial port 3 and serial port 4
3. The above two functions are in the planning stage. The current chip does not have this function

- 2016/5/6 -
1. Add I2C Option to Selection Price List
2. The description of the port switching of the serial port 4 is wrong in the correction overview

- 2016/4/27 -
1. Modify some incorrect instruction execution time in the instruction list

- 2016/4/22 -
1. Modify the LQFP64S package diagram and pin arrangement of the STC8A8K64S4 series
2. Modify the package diagram and pin arrangement of the LQFP64S of STC8F8K64S4 series

- 2016/4/15 -
1. Modify the package diagram and pin arrangement of LQFP44 and LQFP48 of STC8A8K64S4 series
2. Modify the package drawing and pin arrangement of LQFP44 and LQFP48 of STC8F8K64S4 series
3. Modify the package diagram and pin arrangement of the STC8F2K64S4 series LQFP44