

STC8H family of Microcontrollers Reference Manual

STC MCU

Contents

1	Overview	20
2	Features, Price and Pins.....	21
2.1	STC8H1K08-36I-TSSOP20/QFN20 family.....	21
2.1.1	Features and Price.....	21
2.1.2	Pinouts	23
2.1.3	Pin descriptions	25
2.2	STC8H1K28-36I-LQFP32/QFN32 family.....	27
2.2.1	Features and Price.....	27
2.2.2	Pinouts	29
2.2.3	Pin descriptions	30
2.3	STC8H3K64S2-45I-LQFP48/32,QFN48/32,TSSOP20 family.....	34
2.3.1	Features and Price(Quasi 16-bit MCU with 16-bit hardware multiplier and divider MDU16)....	34
2.3.2	Pinouts	37
2.3.3	Pin descriptions	40
2.4	STC8H3K64S4-45I-LQFP48/32, QFN48/32, TSSOP20 family.....	44
2.4.1	Features and Price(Quasi 16-bit MCU with 16-bit hardware multiplier and divider MDU16)....	44
2.4.2	Pinouts	47
2.4.3	Pin descriptions	50
2.5	STC8H8K64U-45I-LQFP64/48,QFN64/48 (USB family)	54
2.5.1	Features and Price(Quasi 16-bit MCU with 16-bit hardware multiplier and divider MDU16)....	54
2.5.2	Pinouts	57
2.5.3	Pin descriptions	61
2.6	STC8H4K64TLR-45I-LQFP48/QFN48/LQFP32/TSSOP20 (touch key/LED/RTC family). 67	
2.6.1	Features and Price(Quasi 16-bit MCU with 16-bit hardware multiplier and divider MDU16)....	67
2.6.2	Pinouts	70
2.6.3	RTC reference circuit diagram (No VBAT pin).....	72
2.6.4	Pin descriptions	73
2.7	STC8H4K64TLCD-45I-LQFP64/QFN64/LQFP48/QFN48 (Touch key/LCD/RTC family). 77	
2.7.1	Features and Price(Quasi 16-bit MCU with 16-bit hardware multiplier and divider MDU16)....	77
2.7.2	Pinouts	81
2.7.3	RTC reference circuit diagram (No VBAT pin).....	83
2.7.4	Pin descriptions	85
2.8	STC8H1K08TR-36I-TSSOP20/QFN20(Touch key/RTC family).....	90
2.8.1	Features and Price(Quasi 16-bit MCU with 16-bit hardware multiplier and divider MDU16)....	90
2.8.2	Pinouts	93
2.8.3	Pin descriptions	94
2.9	STC32G12K128-LQFP64/48/32,PDIP40 (New product notice)	97
2.9.1	Features and Price.....	97
2.9.2	Pinouts	100
2.9.3	Pin descriptions	102

2.10	STC32G6K64-LQFP48/LQFP32/PDIP40 (New product notice)	108
2.10.1	Features and Price.....	108
2.10.2	Pinouts	110
2.10.3	Pin descriptions	111
2.11	STC32F12K60-LQFP48/LQFP32/PDIP40 (New product notice).....	116
2.11.1	Features and Price.....	116
2.11.2	Pinouts	119
2.11.3	Pin descriptions.....	120
3	Function pins switch.....	125
3.1	Register related to function pin switch	125
3.1.1	Peripheral port switch register 1(P_SW1)	125
3.1.2	Peripheral port switch register 2(P_SW2)	125
3.1.3	Clock selection register(MCLKOCR)	126
3.1.4	T3/T4 selection register(T3T4PIN)	126
3.1.5	Advanced PWM selection register(PWMx_PS).....	126
3.1.6	Advanced PWM Pin selection register(PWMx_ETRPS)	127
3.2	Example Routines.....	129
3.2.1	USART1 switch.....	129
3.2.2	USART2 switch.....	130
3.2.3	UART3 switch.....	132
3.2.4	UART4 switch.....	133
3.2.5	SPI switch.....	135
3.2.6	I2C switch	136
3.2.7	Comparator output switch	138
3.2.8	Main clock output switch	139
4	Package Dimensions	142
4.1	SOP8 Package mechanical data.....	142
4.2	DFN8 Package mechanical data (3mm*3mm)	143
4.3	SOP16 Package mechanical data.....	144
4.4	TSSOP20 Package mechanical data	145
4.5	QFN20 Package mechanical data (3mm*3mm)	146
4.6	LQFP32 Package mechanical data (9mm*9mm)	147
4.7	QFN32 Package mechanical data (4mm*4mm)	148
4.8	LQFP48 Package mechanical data (9mm*9mm)	149
4.9	QFN48 Package mechanical data (6mm*6mm)	150
4.10	LQFP64S Package mechanical data (12mm*12mm)	151
4.11	QFN64 Package mechanical data (8mm*8mm)	152
4.12	Naming rules of STC8 family	153
5	ISP Download and typical application circuit.....	154
5.1	STC8H series ISP download application circuit.....	154
5.1.1	Download using RS-232 converter (general precision ADC), Emulation supported	154
5.1.2	Download using RS-232 converter (High precision ADC), Emulation supported	155
5.1.3	High-precision ADC reference circuit diagram of STC8H3K64S4 series, Emulation supported	156

5.1.4 Download using PL2303-GL, Emulation supported	157
5.1.5 Download using universal USB to Serial Tool, Emulation supported.....	158
5.1.6 Download using U8-Mini tool, Support ISP online and offline download, also support emulation 159	
5.1.7 Download using U8W tool, Support ISP online and offline download, also support emulation.	160
5.1.8 Software simulation USB ISP download directly, which is not recommended, and does not support simulation	161
5.1.9 ISP Download using hardware USB directly.....	162
5.1.10 Microcontroller Power supply Control Reference Circuit.....	164
6. Clock, Reset and Power Management	165
6.1 System Clock Control	165
6.1.1 System clock selection register (CKSEL)	165
6.1.2 Clock Division register (CLKDIV)	165
6.1.3 Internal high speed high precision IRC control register (IRCCR)	166
6.1.4 External Oscillator control register (XOSCCR)	166
6.1.5 External 32KHz Oscillator control register (X32KCR)	166
6.1.6 Internal 32KHz low speed IRC control register (IRC32KCR).....	167
6.1.7 Main clock output control register (MCLKOCR)	167
6.2 STC8H series internal IRC frequency adjustment	168
6.2.1 IRC band selection register (IRCBAND)	168
6.2.2 Internal IRC Frequency Adjustment Register (IRTRIM)	168
6.2.3 Internal IRC frequency trim register (LIRTRIM).....	168
6.2.4 Clock Divide Register (CLKDIV).....	169
6.2.5 Example of fine-tuning to get a user frequency of 3MHz	170
6.3 System reset	173
6.3.1 Watch dog timer reset (WDT_CONTR)	173
6.3.2 Software reset (IAP_CONTR)	174
6.3.3 Low voltage reset (RSTCFG).....	175
6.3.4 Low voltage power-on reset reference circuit (generally not required).....	175
6.3.5 Low voltage button reset reference circuit	175
6.3.6 Traditional 8051 high voltage power-on reset reference circuit	176
6.4 External crystal oscillator and external clock circuit	177
6.4.1 External crystal input circuit.....	177
6.4.2 External clock input circuit (P1.6 cannot be used as general I/O).....	177
6.5 Clock stop / Power Saving Mode and System Power Management	177
6.5.1 Power control register (PCON)	177
6.6 Power-down wake-up timer	179
6.6.1 Power-down wake-up timer count register (WKTCL,WKTCH).....	179
6.7 Example Routines	180
6.7.1 System Clock Source Selection.....	180
6.7.2 Main Clock Output	182
6.7.3 Application of Watch-dog Timer	184
6.7.4 User Defined Downloading by Using Software Reset	186
6.7.5 Low Voltage Detection	188

6.7.6 Power Saving Mode	190
6.7.7 Wake up MCU from Power Saving Mode using INT0/INT1/INT2/INT3/INT4 interrupts.....	192
6.7.8 Wake up MCU from Power Saving Mode using T0/T1/T2/T3/T4 pin interrupts.....	195
6.7.9 Wake up MCU from Power Saving Mode using RxD/RxD2/RxD3/RxD4 pin interrupts.....	199
6.7.10 Wake up MCU from Power Saving Mode using I2C SDA pin	202
6.7.11 Wake up MCU from Power Saving Mode using Power-down wake-up timer	205
6.7.12 Wake up MCU from Power Saving Mode using LVD interrupt (Recommended for use with power-down wake-up timer)	207
6.7.13 Wake up MCU from Power Saving Mode using comparator interrupt (Recommended for use with power-down wake-up timer)	209
6.7.14 Detect the Operating Voltage (Battery Voltage) using LVD	212
7 Memory 217	
7.1 Program Memory	217
7.2 Data Memory	218
7.2.1 Internal RAM.....	218
7.2.2 PSW (program status word register).....	219
7.2.3 On-chip extended RAM.....	220
7.2.4 Auxiliary register (AUXR)	220
7.2.6 Bus speed control register (BUS_SPEED)	220
7.2.7 Bit Addressable Data Memory in 8051	221
7.3 Special parameters of memory.....	223
7.4 Unique ID number and important parameter (CHIPID) stored in read-only special function register	226
7.4.1 Interpretation of Global Unique ID Number in CHIP	227
7.4.2 Interpretation of the internal reference signal source in CHIP	227
7.4.3 Interpretation of internal 32K IRC oscillation frequency in CHIP	227
7.4.4 Interpretation of high precision IRC parameters in CHIP	228
7.4.5 Interpretation of test time parameters in CHIP.....	229
7.4.6 Interpretation of chip package form number in CHIP	229
7.5 Example Routines	230
7.5.1 Read Internal Reference Voltage Value (Read from CHIPID)	230
7.5.2 Read Internal Reference Voltage (from Flash)	233
~~7.5.3 Read Internal Reference Voltage (Read from RAM)~~	~~236~~
7.5.4 Read the Unique ID (Read from CHIPID)	238
7.5.5 Read the Unique ID (Read from Flash).....	242
~~7.5.6 Read the Unique ID (Read from RAM)~~	~~245~~
7.5.7 Read the Frequency of 32K Power-down Wake-up Timer (Read from CHIPID)	248
7.5.8 Read the Frequency of 32K Power-down Wake-up Timer (Read from Flash).....	251
~~7.5.9 Read the Frequency of 32K Power-down Wake-up Timer (Read from RAM)~~	~~254~~
7.5.10 Read the User-defined internal IRC Frequency (Read from CHIPID).....	257
7.5.11 Read the User-defined internal IRC Frequency (Read from Flash).....	265
~~7.5.12 Read the User-defined internal IRC Frequency (Read from RAM)~~	~~270~~
8 Special Function Registers.....	273
8.1 STC8H1K08 family	273
8.2 STC8H1K28-32PIN family	274

8.3 STC8H3K64S4 family	275
8.4 STC8H3K64S2 family	276
8.5 STC8H8K64U-64Pin/48Pin USB family	277
8.6 STC8H2K64T family	278
8.7 STC8H4K64TLR family	279
8.8 STC8H4K64TLCD family	281
8.9 STC8H4K64LCD family	283
8.10 List of Special Function Registers	285
9 I/O Ports 295	
9.1 Registers Related to I/O	295
9.1.1 Port Data Register	296
9.1.2 Ports Mode Registers	296
9.1.3 Pull-up Resistor Control Registers	297
9.1.4 Schmitt Trigger Control Registers	297
9.1.5 Level Shifting Speed Control Registers	297
9.1.6 Drive Current Control Registers	298
9.1.7 Port digital signal input enable control register (PxIE)	298
9.2 Configure I/O Ports	299
9.3 I/O Ports Structure	299
9.3.1 Quasi-Bidirectional I/O (weak pull-up)	299
9.3.2 Push-Pull Output	300
9.3.3 High Impedance Input	300
9.3.4 Open-Drain Output	300
9.3.5 4.1K Pull-up Resistor	301
9.3.6 How to set the external output speed of the I/O port	301
9.3.7 How to set I/O port current drive capability	302
9.3.8 How to reduce the external radiation of I/O ports	302
9.4 Example Routines	303
9.4.1 Port Mode Setting	303
9.4.2 Reading and Writing Operation of Bidirection Port	304
9.5 A Typical Circuit Controlled by Triode	307
9.6 Typical Control Circuit of LED	307
9.7 Interconnection of 3V/5V Devices in Mixed Voltage Power Supply System	307
9.8 Make I/O Port Output Low When Power on Reset	308
9.9 Circuit Diagram of Driving 8 Digital LEDs using 74HC595	309
9.10 Digital LEDs Driven Directly by I/O Port Circuit	310
9.11 LCD Segment LCD Driven Directly by I/O Port Circuit	310
10 Instruction Set	330
11 Interrupt System	333
11.1 Interrupt sources of STC8H series	333
11.2 Structure of STC8H Interrupt	336
11.3 Interrupt List of STC8H Series	337
11.4 Registers Related to Interrupt	339
11.4.1 Interrupt Enable Registers (Interrupt Enable bits)	340

11.4.2 Interrupt Request Registers (Interrupt flags)	344
11.4.3 Interrupt Priority Control Registers	347
11.5 Example Routines	352
11.5.1 INT0 Interrupt (Rising and Falling Edges).....	352
11.5.2 INT0 Interrupt (Falling Edge).....	354
11.5.3 INT1 Interrupt (Rising and Falling Edges).....	355
11.5.4 INT1 Interrupt (Falling Edge).....	357
11.5.5 INT2 Interrupt (Falling Edge).....	359
11.5.6 INT3 Interrupt (Falling Edge).....	361
11.5.7 INT4 Interrupt (Falling Edge).....	362
11.5.8 Timer0 Interrupt.....	364
11.5.9 Timer1 Interrupt.....	366
11.5.10 Timer2 Interrupt.....	368
11.5.11 Timer3 Interrupt.....	370
11.5.12 Timer4 Interrupt.....	372
11.5.13 UART1 Interrupt.....	374
11.5.14 UART2 Interrupt.....	376
11.5.15 UART3 Interrupt.....	379
11.5.16 UART4 Interrupt.....	381
11.5.17 ADC Interrupt	384
11.5.18 LVD Interrupt	386
11.5.19 Comparator Interrupt	388
11.5.20 SPI Interrupt	389
11.5.21 I2C Interrupt	391
12 I/O port interrupt.....	395
12.1 I/O port interrupt related registers.....	395
12.1.1 Port interrupt enable registers (PxINTE).....	397
12.1.2 Port interrupt flag registers (PxINTF)	397
12.1.3 Port interrupt mode configuration registers (PxIM0, PxIM1).....	397
12.1.4 Port interrupt priority control registers (PINIPL, PINIPH)	398
12.1.5 Port interrupt power-down wake-up enable registers (PxWKUE)	398
12.2 Example Routines	399
12.2.1 P0 Falling edge interrupt	399
12.2.2 P1 rising edge interrupt.....	402
12.2.3 P2 low level interrupt	406
12.2.4 Port3 high level interrupt.....	411
13 Timer/Counter	417
13.1 Registers Related to Timers	417
13.2 Timer 0/1.....	418
13.2.1 Timer 0 and 1 Control Register	418
13.2.2 Timer 0/1 Mode Register	418
13.2.3 Timer0 mode 0 (16-bit auto-reloadable mode).....	419
13.2.4 Timer0 mode 1 (16-bit non-autoreloadable mode)	420
13.2.5 Timer 0 mode 2 (8-bit auto-reloadable mode).....	421

13.2.6 Timer 0 mode 3 (16-bit auto-reloadable mode with non-maskable interrupt, which can be used as real-time operating system metronome)	421
13.2.7 Timer 1 mode 0 (16-bit auto-reloadable mode)	422
13.2.8 Timer1 mode 1 (16-bit non-autoreloadable mode)	423
13.2.9 Timer 1 mode 2 (8-bit auto-reloadable mode)	423
13.2.10 Timer 0 Counting Registers	424
13.2.11 Timer 1 Counting Registers	424
13.2.12 Auxiliary Register 1 (AUXR)	424
13.2.13 External Interrupt and Clock Output Control Register (INTCLKO)	425
13.2.14 Timer 0 calculation formula	425
13.2.15 Timer 1 calculation formula	426
13.3 Timer 2	427
13.3.1 Auxiliary Register 1 (AUXR)	427
13.3.2 External Interrupt and Clock Output Control Register (INTCLKO)	427
13.3.3 Timer 2 Counting Registers	427
13.3.4 Timer 2 8-bit Prescaler Register (TM2PS)	427
13.3.5 Timer 2 working mode	427
13.3.6 Timer 2 calculation formula	428
13.4 Timer 3/4	429
13.4.1 Timer4 and Timer 3 Control Register (T4T3M)	429
13.4.2 Timer 3 Counting Registers	429
13.4.3 Timer 4 Counting Registers	429
13.4.4 Timer 3 8-bit Prescaler Register (TM3PS)	430
13.4.5 Timer 4 8-bit Prescaler Register (TM4PS)	430
13.4.6 Timer 3 working mode	430
13.4.7 Timer 4 working mode	431
13.4.8 Timer 3 calculation formula	431
13.4.9 Timer 4 calculation formula	432
13.5 Example Routines	433
13.5.1 Timer 0 (Mode 0 – 16-bit auto reload)	433
13.5.2 Timer 0 (Mode 1 – 16-bit non-auto reload)	434
13.5.3 Timer 0 (Mode 2 - 8-bit auto reload)	436
13.5.4 Timer 0 (Mode 3 - 16-bit auto reload with non-maskable interrupt)	438
13.5.5 Timer 0 (External count - T0 is extended for external falling edge interrupt)	440
13.5.6 Timer 0 (Pulse width measurement for high-level width of INT0)	442
13.5.7 Timer 0 (Mode 0, Divided clock output)	444
13.5.8 Timer 1 (Mode 0 - 16-bit auto reload)	445
13.5.9 Timer 1 (Mode 1 - 16-bit non-auto reload)	447
13.5.10 Timer 1 (Mode 2 - 8-bit auto reload)	449
13.5.11 Timer 1 (External count – T1 is extended for external falling edge interrupt)	451
13.5.12 Timer 1 (Pulse width measurement for high-level width of INT1)	453
13.5.13 Timer 1 (Mode 0, Divided clock output)	455
13.5.14 Timer 1 (Mode 0) is used as baud rate generator of UART1	456
13.5.15 Timer 1 (Mode 2) is used as baud rate generator of UART1	460

13.5.16 Timer 2 (16-bit auto reload).....	464
13.5.17 Timer 2 (External count – T2 is extended for external falling edge interrupt)	466
13.5.18 Timer 2 (Divided clock output)	468
13.5.19 Timer 2 is used as baud rate generator of UART1.....	470
13.5.20 Timer 2 is used as baud rate generator of UART2.....	474
13.5.21 Timer 2 is used as baud rate generator of UART3.....	478
13.5.22 Timer 2 is used as baud rate generator of UART4.....	482
13.5.23 Timer 3 (16-bit auto reload).....	486
13.5.24 Timer 3 (External count – T3 is extended for external falling edge interrupt)	488
13.5.25 Timer 3 (Divided clock output)	490
13.5.26 Timer 3 is used as baud rate generator of UART3.....	492
13.5.27 Timer 4 (16-bit auto reload).....	496
13.5.28 Timer 4 (External count – T4 is extended for external falling edge interrupt)	499
13.5.29 Timer 4 (Divided clock output)	501
13.5.30 Timer 4 is used as baud rate generator of UART4.....	503
14 UART Communication	508
14.1 Registers Related to UARTs	508
14.2 UART1 509	
14.2.1 UART1 control register	509
14.2.2 UART1 data register	509
14.2.3 Power control register.....	510
14.2.4 Auxiliary register 1	510
14.2.5 UART1 Mode 0	510
14.2.6 UART1 Mode 1	511
14.2.7 UART1 Mode 2	513
14.2.8 UART1 Mode 3	514
14.2.9 Automatic Address Recognition	515
14.2.10 UART1 slave address control registers.....	515
14.3 UART2 517	
14.3.1 UART2 control register	517
14.3.2 UART2 data register	517
14.3.3 UART2 Mode 0	517
14.3.4 UART2 Mode 1	518
14.4 UART3 519	
14.4.1 UART3 control register	519
14.4.2 UART3 data register	519
14.4.3 UART3 Mode 0	519
14.4.4 UART3 Mode 1	520
14.5 UART4 522	
14.5.1 UART4 control register	522
14.5.2 UART4 data register	522
14.5.3 UART4 Mode 0	522
14.5.4 UART4 Mode 1	523
14.6 Precautions of UARTs.....	524

14.7 Example Routines	525
14.7.1 UART1 using T2 as baud rate generator	525
14.7.2 UART1 using T1 (Mode 0) as baud rate generator.....	528
14.7.3 UART1 using T1 (Mode 2) as baud rate generator.....	532
14.7.4 UART2 using T2 as baud rate generator	536
14.7.5 UART3 using T2 as baud rate generator	540
14.7.6 UART3 using T3 as baud rate generator	544
14.7.7 UART4 using T2 as baud rate generator	548
14.7.8 UART4 using T4 as baud rate generator	552
14.7.9 Serial multi-MCUs communication.....	556
14.7.10 UART to LIN bus	557
15 Comparator, Power-down Detection, Internal Reference Voltage	567
15.1 Internal Structure of Comparator	567
15.2 Registers Related to Comparator	568
15.2.1 Comparator control register 1.....	568
15.2.2 Comparator control register 2.....	569
15.2.3 Comparator Extended Configuration Register (CMPEXCFG)	569
15.3 Example Routines	571
15.3.1 Using Old Version Comparator (Interrupt Mode).....	571
15.3.2 Using Old Version Comparator (Polling Mode).....	573
15.3.3 Using New Version Comparator (Interrupt Mode)	576
15.3.4 Using New Version Comparator (Polling Mode).....	579
15.3.5 Multiplexing Application of Old Version Comparator (Comparator+ADC input channel).....	582
15.3.6 Multiplexing Application of New Version Comparator (Comparator+ADC input channel)	584
15.3.7 Comparator is Used for External Power-down Detection (User data should be saved to EEPROM in time during power down)	587
15.3.8 Comparator is Used to Detect the Operation Voltage (Battery Voltage).....	588
16 IAP/EEPROM/DATA-FLASH.....	592
16.1 EEPROM operation time.....	592
16.2 Registers Related to EEPROM	592
16.2.1 EEPROM data register (IAP_DATA)	592
16.2.2 EEPROM address registers	593
16.2.3 EEPROM command register (IAP_CMD)	593
16.2.4 EEPROM trigger register	593
16.2.5 EEPROM control register.....	593
16.2.6 EEPROM erase wait time control register.....	594
16.3 EEPROM Size and Address.....	594
16.4 Example Routines	597
16.4.1 EEPROM Basic Operation	597
16.4.2 Read EEPROM using MOVC	600
16.4.3 Send Out the Data in EEPROM Using UART	604
16.4.4 UART1 reads and writes EEPROM - Read using MOVC	608
16.4.5 Password erasing and writing - multi-sector backup - UART1 operation	615
17 ADC, Internal Reference Voltage.....	625

17.1 Registers Related to ADC	625
17.1.1 ADC control register.....	625
17.1.2 ADC configuration register	628
17.1.3 ADC result registers (ADC_RES, ADC_RES1)	629
17.1.4 ADC timing control register	629
17.2 ADC related calculation formula	630
17.2.1 ADC speed calculation formula.....	630
17.2.2 ADC conversion result calculation formula.....	631
17.2.3 Reverse calculation formula for ADC input voltage	631
17.2.4 Reverse working voltage calculation formula	631
17.3 10 BIT ADC Static Characteristics	632
17.4 12 BIT ADC Static Characteristics	632
17.5 ADC application reference circuit diagram.....	632
17.5.1 General precision ADC reference circuit diagram.....	632
17.5.2 High-precision ADC reference circuit diagram	633
17.6 Example Routines	634
17.6.1 ADC Basic Operation (Polling Mode).....	634
17.6.2 ADC Basic Operation (Interrupt Mode)	636
17.6.3 Format ADC Conversion Result.....	638
17.6.4 Detect External Voltage or Battery Voltage using ADC 15th Channel	641
17.6.5 Using ADC as Capacitive Sensing Touch Keys	644
17.6.6 Key-scan Application Circuit Diagram using ADC.....	656
17.6.7 Reference circuit diagram for detecting negative voltage	657
17.6.8 The application of common addition circuit in ADC.....	658
18 Synchronous Serial Peripheral Interface (SPI).....	659
18.1 Registers Related to SPI.....	659
18.1.1 SPI Status register.....	659
18.1.2 SPI Control Register	659
18.1.3 SPI Data Register	660
18.2 SPI Communication Modes	660
18.2.1 Single Master and Single Slave Mode.....	660
18.2.2 Dual Devices Configuration Mode.....	661
18.2.3 Single Master and Multiple Slaves Mode.....	661
18.3 Configure SPI	662
18.4 Data Format	663
18.5 Example Routines	665
18.5.1 Master Routine of Single Master Single Slave Mode (Interrupt Mode).....	665
18.5.2 Slave Routine of Single Master Single Slave Mode (Interrupt Mode).....	667
18.5.3 Master Routine of Single Master Single Slave Mode (Polling Mode)	669
18.5.4 Slave Routine of Single Master Single Slave Mode (Polling Mode)	671
18.5.5 Routine of Mutual Master-Slave Mode (Interrupt Mode)	674
18.5.6 Routine of Mutual Master-Slave Mode (Polling Mode).....	676
19 I2C Bus 680	
19.1 Registers Related to I ² C	680

19.2 I ² C Master Mode.....	681
19.2.1 I2C Configuration Register (I2CCFG)	681
19.2.2 I ² C Master Control Register (I2CMSCR).....	681
19.2.3 I ² C Master Auxiliary Control Register (I2CMSAUX)	683
19.2.4 I ² C Master Status Register (I2CMSST).....	683
19.3 I ² C Slave Mode	685
19.3.1 I ² C Slave Control Register (I2CSLCR).....	685
19.3.2 I ² C Slave Status Register (I2CSLST)	685
19.3.3 I ² C Slave Address Register (I2CSLADR)	686
19.3.4 I ² C data registers (I2CTXD, I2CRXD)	687
19.4 Example Routines	688
19.4.1 I ² C is Used to Access AT24C256 in Master Mode (Interrupt Mode)	688
19.4.2 I ² C is Used to Access AT24C256 in Master Mode AT24C256 (Polling Mode)	694
19.4.3 I ² C is Used to Access PCF8563 in Master Mode	700
19.4.4 I ² C Slave Mode (Polling Mode)	705
19.4.5 I ² C Slave Mode (Polling Mode)	710
19.4.6 Master Codes for testing I ² C Slave Mode	714
20 16-bit advanced PWM timer, support quadrature encoder	720
20.1 Introduction.....	723
20.2 Main features	723
20.3 Time base unit.....	724
20.3.1 Reading and writing 16-bit counter	724
20.3.2 Writing operation of 16-bit PWMA_ARR register.....	725
20.3.3 Prescaler	725
20.3.4 Up counting mode	725
20.3.5 Down counting mode.....	727
20.3.6 Center alignment mode (up/down count)	728
20.3.7 Repeat counter	730
20.4 Clock/Trigger Controller.....	731
20.4.1 Prescaler Clock (CK_PSC).....	731
20.4.2 Internal clock source (f _{MASTER}).....	731
20.4.3 External clock source mode 1.....	732
20.4.4 External clock source mode 2.....	733
20.4.5 Trigger synchronization.....	734
20.4.6 Synchronize with PWMB.....	736
20.5 Capture/Compare Channel	739
20.5.1 Writing process of 16-bit PWMA_CCRi register.....	741
20.5.2 Input Module	741
20.5.3 Input Capture Mode.....	741
20.5.4 Output Module.....	743
20.5.5 Forced output mode	744
20.5.6 Output Compare Mode	745
20.5.7 PWM Mode	745
20.5.8 Using the brake function (PWMFLT).....	750

20.5.9 Clear the OCIREF signal when an external event occurs.....	752
20.5.10 Encoder interface mode	752
20.6 Interrupt	754
20.7 PWMA/PWMB registers description.....	754
20.7.1 Output Enable Registers (PWMx_ENO).....	754
20.7.2 Output Additional Enable Registers (PWMx_IOAUX)	755
20.7.3 Control Registers 1 (PWMx_CR1).....	756
20.7.4 Control Registers 2 (PWMx_CR2), and trigger ADC in real time	757
20.7.5 Slave Mode Control Registers (PWMx_SMCR).....	758
20.7.6 External Trigger Registers (PWMx_ETR)	760
20.7.7 Interrupt Enable Registers (PWMx_IER).....	760
20.7.8 Status Registers 1 (PWMx_SR1).....	761
20.7.9 Status Registers 2 (PWMx_SR2).....	762
20.7.10 Event Generation Registers (PWMx_EGR)	762
20.7.11 Capture/Compare Mode Register 1 (PWMx_CCMR1).....	763
20.7.12 Capture/compare mode registers 2 (PWMx_CCMR2).....	765
20.7.13 Capture/Compare Mode Registers 3 (PWMx_CCMR3)	766
20.7.14 Capture/Compare Mode Registers 4 (PWMx_CCMR4)	767
20.7.15 Capture/Compare Enable Registers 1 (PWMx_CCER1)	769
20.7.16 Capture/Compare Enable Registers 2 (PWMx_CCER2)	770
20.7.17 Counters upper 8 bits (PWMx_CNTRH)	770
20.7.18 Counters low 8 bits (PWMx_CNTRL).....	770
20.7.19 Prescalers high 8 bits (PWMx_PSCRH), output frequency calculation formula	771
20.7.20 Prescalers low 8 bits (PWMx_PSCRL).....	771
20.7.21 The upper 8 bits of the auto-reload registers (PWMx_ARRH)	771
20.7.22 The lower 8 bits of the auto-reload registers (PWMx_ARRL).....	772
20.7.23 Repeat Counter Registers (PWMx_RCR)	772
20.7.24 Capture/Compare Registers 1/5 High 8 bits (PWMx_CCR1H)	772
20.7.25 Capture/compare registers 1/5 lower 8 bits (PWMx_CCR1L).....	772
20.7.26 Capture/Compare Registers 2/6 High 8-bit (PWMx_CCR2H)	773
20.7.27 Capture/compare registers 2/6 lower 8 bits (PWMx_CCR2L).....	773
20.7.28 Capture/Compare Registers 3/7 High 8 bits (PWMx_CCR3H)	773
20.7.29 Capture/compare registers 3/7 lower 8 bits (PWMx_CCR3L).....	773
20.7.30 Capture/compare registers 4/8 high 8 bits (PWMx_CCR4H)	773
20.7.31 Capture/compare registers 4/8 lower 8 bits (PWMx_CCR4L).....	773
20.7.32 Brake Registers (PWMx_BKR)	774
20.7.33 Dead Zone Registers (PWMx_DTR).....	775
20.7.34 Output Idle Status Registers (PWMx_OISR)	775
20.8 Sample Routines	776
20.8.1 Six-step BLDC brushless DC motor drive (with HALL)	776
20.8.2 BLDC brushless DC motor drive (without HALL)	788
20.8.3 Quadrature encoder mode.....	797
20.8.4 Single pulse mode (trigger control pulse output).....	798
20.8.5 Gating mode (input level enable counter)	801

20.8.6 External clock mode	803
20.8.7 Input capture mode to measure the pulse period (capture rising edge to rising edge or falling edge to falling edge)	805
20.8.8 Input capture mode to measure the pulse high level width (capture rising edge to falling edge)	807
20.8.9 Input capture mode to measure the pulse low-level width (capture falling edge to rising edge)	808
20.8.10 Simultaneous measurement of pulse period and duty cycle in input capture mode	809
20.8.11 PWM complementary output with dead zone control	811
20.8.12 PWM port as external interrupt (falling edge interrupt or rising edge interrupt).....	813
20.8.13 Output waveforms with any period and any duty cycle	814
20.8.14 Use PWM CEN to start PWMA timer and trigger ADC in real time	815
20.8.15 Reference circuit diagram for implementing 16-bit DAC using PWM	817
20.8.16 Using PWM to realize complementary SPWM	817
20.8.17 Advanced PWM Output - Frequency Adjustable - Pulse Count.....	821
21 USB Universal Serial Bus	827
21.1 USB related registers	827
21.1.1 USB Control Register (USBCON).....	828
21.1.2 USB clock control register (USBCLK)	828
21.1.3 USB Indirect Address Address Register (USBADDR)	829
21.1.4 USB indirect address data register (USBDATA)	829
21.2 USB Controller Registers (SIE).....	829
21.2.1 USB Function Address Register (FADDR)	831
21.2.2 USB Power Control Register (POWER).....	831
21.2.3 USB endpoint IN interrupt flag Register (INTRIN1).....	831
21.2.4 USB endpoint OUT interrupt flag Register (INTROUT1).....	832
21.2.5 USB power interruption flag (INTRUSB).....	832
21.2.6 USB endpoint IN interrupt enable register (INTRIN1E).....	833
21.2.7 USB endpoint OUT interrupt enable register (INTROUT1E).....	833
21.2.8 USB power interruption enable register (INTRUSB).....	833
21.2.9 USB data frame number registers (FRAMEn)	834
21.2.10 USB Endpoint Index Register (INDEX)	834
21.2.11 Maximum packet size of IN endpoint (INMAXP)	834
21.2.12 USB Endpoint 0 Control Status Register (CSR0).....	834
21.2.13 IN endpoint control status register 1 (INCSR1)	835
21.2.14 IN endpoint control status register 2 (INCSR2)	836
21.2.15 Maximum packet size of OUT endpoint (OUTMAXP)	836
21.2.16 OUT endpoint control status register 1 (OUTCSR1)	836
21.2.17 OUT endpoint control status register 2 (OUTCSR2)	837
21.2.18 OUT length of USB endpoint 0 (COUNT0).....	837
21.2.19 OUT length of USB endpoint (OUTCOUNTn)	838
21.2.20 FIFO Data Access Registers (FIFO _n) of USB Endpoint	838
21.3 USB Product Development Considerations	838
21.4 Sample Routines	839

21.4.1 Example of HID man-machine interface equipment	839
22 Touch Key Controller.....	850
22.1 Internal Structure Diagram of Touch Key Controller	852
22.2 Wake-up from Low Power Mode using Touch Key.....	852
22.3 Operation Steps When Touch Key Function is Used only	852
22.4 Operation Steps for Wake-up from Low Power Mode using Touch Key.....	853
22.5 Registers Related to Touch Key Controller.....	853
22.5.1 Touch Key Enable Registers.....	854
22.5.2 Touch Key Configuration Registers	854
22.5.3 Touch key power-down mode wake-up time control register.....	855
22.5.4 Touch Key Control Register	855
22.5.5 Touch Key Status Register 1.....	856
22.5.6 Touch Key Status Register 2.....	857
22.5.7 Touch Key Time Control Register	858
22.5.8 Touch key data registers	858
22.5.9 Touch key threshold register.....	858
22.6 Basic Reference Circuit and Precautions	859
22.7 Example Routines	859
22.7.1 Introduction of touch key configuration software	859
23 LED Driver 863	
23.1 Internal Structure Diagram of LED Driver	864
23.2 Registers Related to LED Driver	864
23.2.1 COM Enable Register.....	865
23.2.2 SEG Enable Registers.....	865
23.2.3 LED Control Register.....	865
23.2.4 LED Clock Divide Register.....	866
23.2.5 LED data registers of common anode mode.....	866
23.2.6 LED data registers of common cathod mode.....	866
23.3 LED Common Cathod Mode (LEDMODE = 00).....	868
23.4 LED Common Anode Mode (LEDMODE = 01).....	869
23.5 LED Common Cathode/ Common Anode Mode (LEDMODE = 10).....	870
23.6 Touch Keys and LED Driver Share I/O	871
23.7 Reference Circuit of Common Cathode Mode.....	873
23.8 Reference Circuit of Common Anode Mode	873
23.9 Reference Circuit of Common Cathode/Common Anode Mode	874
23.10 Example Routines	875
23.10.1 Common cathode/common anode mode drives 16 7-segment digital tubes.....	875
24 RTC real time clock.....	878
24.1 RTC related registers.....	878
24.1.1 RTC control register (RTCCR).....	879
24.1.2 RTC Configuration Register (RTCCFG).....	879
24.1.3 RTC Interrupt Enable Register (RTCIEN).....	879
24.1.4 RTC Interrupt Request Register (RTCIF).....	880
24.1.5 RTC Alarm Setting Registers	880

24.1.6 RTC real-time clock initial value setting registers.....	880
24.1.7 RTC real-time clock count registers	881
24.2 RTC reference circuit diagram (without VBAT pin).....	882
24.3 RTC practical circuit diagram.....	883
24.4 Example Routines	884
24.4.1 Serial port printing RTC clock example	884
25 LCD driver (Traditional segment/stroke LCD driver)	887
25.1 LCD function pins switch	887
25.2 LCD related registers	888
25.2.1 LCD configuration register (LCDCFG)	889
25.2.2 LCD configuration register 2 (LCDCFG2)	889
25.2.3 Dead Time Length Configuration Register (DBLEN).....	889
25.2.4 COM Time Length Configuration Registers (COMLENx)	890
25.2.5 Flicker Rate Configuration Register (BLANKRATE).....	890
25.2.6 LCD Control Register (LCDCR).....	890
25.2.7 COM Line Enable Register (COMON).....	890
25.2.8 SEG Line Enable Register (SEGONx)	890
25.2.9 LCD Data Registers (CxSEGVx)	890
25.3 LCD display related configuration.....	891
25.3.1 Configure LCD refresh rate (frame rate)	891
25.3.2 Configure LCD Flicker Rate	891
26 LCM interface (8/16-bit color screen module I8080/M6800 interface).....	892
26.1 LCM interface function pin switch	893
26.2 Registers Related to LCM.....	893
● LCM Interface Configuration Register (LCMIFCFG)	893
26.2.2 LCM Interface Configuration Register 2 (LCMIFCFG2)	894
26.2.3 LCM Interface Control Register (LCMIFCR)	894
26.2.4 LCM Interface Status Register (LCMIFSTA)	894
26.2.5 LCM interface Data Registers (LCMIFDATL, LCMIFDATH)	894
26.3 I8080/M6800 mode LCM interface timing diagram.....	896
26.3.1 I8080 mode.....	896
26.3.2 M6800 mode.....	898
27 DMA (batch data transfer)	900
27.1 Registers Related to DMA	900
27.2 Data read and write between memory and memory (M2M_DMA).....	902
27.2.1 M2M_DMA Configuration Register (DMA_M2M_CFG).....	902
27.2.2 M2M_DMA Control Register (DMA_M2M_CR)	902
27.2.3 M2M_DMA Status Register (DMA_M2M_STA).....	903
27.2.4 M2M_DMA transfer total byte register (DMA_M2M_AMT)	903
27.2.5 M2M_DMA transfer complete byte register (DMA_M2M_DONE)	903
27.2.6 M2M_DMA Send Address Registers (DMA_M2M_TXAx)	903
27.2.7 M2M_DMA Receive Address Registers (DMA_M2M_RXAx)	903
27.3 ADC Automatic Data Storage (ADC_DMA).....	904
27.3.1 ADC_DMA Configuration Register (DMA_ADC_CFG)	904

27.3.2 ADC_DMA Control Register (DMA_ADC_CR).....	904
27.3.3 ADC_DMA Status Register (DMA_ADC_STA)	904
27.3.4 ADC_DMA Receive Address Registers (DMA_ADC_RXAx).....	904
27.3.5 ADC_DMA Configuration Register 2 (DMA_ADC_CFG2)	904
27.3.6 ADC_DMA Channel Enable Registers (DMA_ADC_CHSWx).....	905
27.3.7 Data storage format of ADC_DMA.....	906
27.4 Data exchange between SPI and memory (SPI_DMA)	907
27.4.1 SPI_DMA Configuration Register (DMA_SPI_CFG)	907
27.4.2 SPI_DMA Control Register (DMA_SPI_CR).....	908
27.4.3 SPI_DMA Status Register (DMA_SPI_STA)	908
27.4.4 SPI_DMA transfer total byte register (DMA_SPI_AMT).....	909
27.4.5 SPI_DMA transfer complete byte register (DMA_SPI_DONE)	909
27.4.6 SPI_DMA Send Address Registers (DMA_SPI_TXAx).....	909
27.4.7 SPI_DMA Receive Address Registers (DMA_SPI_RXAx).....	909
27.4.8 SPI_DMA Configuration Register 2 (DMA_SPI_CFG2)	909
27.5 Data exchange between UART1 and memory (UR1T_DMA, UR1R_DMA).....	910
27.5.1 UR1T_DMA Configuration Register (DMA_UR1T_CFG).....	910
27.5.2 UR1T_DMA Control Register (DMA_UR1T_CR)	910
27.5.3 UR1T_DMA Status Register (DMA_UR1T_STA).....	910
27.5.4 UR1T_DMA transfer total byte register (DMA_UR1T_AMT)	910
27.5.5 UR1T_DMA transfer complete byte register (DMA_UR1T_DONE).....	911
27.5.6 UR1T_DMA Send Address Registers (DMA_UR1T_TXAx)	911
27.5.7 UR1R_DMA Configuration Register (DMA_UR1R_CFG)	911
27.5.8 UR1R_DMA Control Register (DMA_UR1R_CR).....	911
27.5.9 UR1R_DMA Status Register (DMA_UR1R_STA).....	911
27.5.10 UR1R_DMA transfer total byte register (DMA_UR1R_AMT).....	912
27.5.11 UR1R_DMA transfer complete byte register (DMA_UR1R_DONE)	912
27.5.12 UR1R_DMA Receive Address Registers (DMA_UR1T_RXAx)	912
27.6 Data exchange between UART2 and memory (UR2T_DMA, UR2R_DMA).....	913
27.6.1 UR2T_DMA Configuration Register (DMA_UR2T_CFG).....	913
27.6.2 UR2T_DMA Control Register (DMA_UR2T_CR)	913
27.6.3 UR2T_DMA Status Register (DMA_UR2T_STA)	913
27.6.4 UR2T_DMA transfer total byte register (DMA_UR2T_AMT)	913
27.6.5 UR2T_DMA transfer complete byte register (DMA_UR2T_DONE).....	914
27.6.6 UR2T_DMA Send Address Registers (DMA_UR2T_TXAx)	914
27.6.7 UR2R_DMA Configuration Register (DMA_UR2R_CFG)	914
27.6.8 UR2R_DMA Control Register (DMA_UR2R_CR).....	914
27.6.9 UR2R_DMA Status Register (DMA_UR2R_STA).....	914
27.6.10 UR2R_DMA transfer total byte register (DMA_UR2R_AMT).....	915
27.6.11 UR2R_DMA transfer complete byte register (DMA_UR2R_DONE)	915
27.6.12 UR2R_DMA Receive Address Registers (DMA_UR2T_RXAx)	915
27.7 Data exchange between UART3 and memory (UR3T_DMA, UR3R_DMA).....	916
27.7.1 UR3T_DMA Configuration Register (DMA_UR3T_CFG).....	916
27.7.2 UR3T_DMA Control Register (DMA_UR3T_CR)	916

27.7.3 UR3T_DMA Status Register (DMA_UR3T_STA)	916
27.7.4 UR3T_DMA transfer total byte register (DMA_UR3T_AMT)	916
27.7.5 UR3T_DMA transfer complete byte register (DMA_UR3T_DONE).....	917
27.7.6 UR3T_DMA Send Address Registers (DMA_UR3T_TXAx)	917
27.7.7 UR3R_DMA Configuration Register (DMA_UR3R_CFG)	917
27.7.8 UR3R_DMA Control Register (DMA_UR3R_CR)	917
27.7.9 UR3R_DMA Status Register (DMA_UR3R_STA).....	917
27.7.10 UR3R_DMA transfer total byte register (DMA_UR3R_AMT)	918
27.7.11 UR3R_DMA transfer complete byte register (DMA_UR3R_DONE)	918
27.7.12 UR3R_DMA Receive Address Registers (DMA_UR3T_RXAx)	918
27.8 Data exchange between UART4 and memory (UR4T_DMA, UR4R_DMA).....	919
27.8.1 UR4T_DMA Configuration Register (DMA_UR4T_CFG).....	919
27.8.2 UR4T_DMA Control Register (DMA_UR4T_CR)	919
27.8.3 UR4T_DMA Status Register (DMA_UR4T_STA)	919
27.8.4 UR4T_DMA transfer total byte register (DMA_UR4T_AMT)	919
27.8.5 UR4T_DMA transfer complete byte register (DMA_UR4T_DONE).....	920
27.8.6 UR4T_DMA Send Address Registers (DMA_UR4T_TXAx)	920
27.8.7 UR4R_DMA Configuration Register (DMA_UR4R_CFG)	920
27.8.8 UR4R_DMA Control Register (DMA_UR4R_CR)	920
27.8.9 UR4R_DMA Status Register (DMA_UR4R_STA).....	920
27.8.10 UR4R_DMA transfer total byte register (DMA_UR4R_AMT).....	921
27.8.11 UR4R_DMA transfer complete byte register (DMA_UR4R_DONE)	921
27.8.12 UR4R_DMA Receive Address Registers (DMA_UR4T_RXAx)	921
27.9 Data exchange between LCM and memory (LCM_DMA)	922
27.9.1 LCM_DMA Configuration Register (DMA_LCM_CFG)	922
27.9.2 LCM_DMA Control Register (DMA_LCM_CR)	922
27.9.3 LCM_DMA Status Register (DMA_LCM_STA).....	922
27.9.4 LCM_DMA transfer total byte register (DMA_LCM_AMT).....	923
27.9.5 LCM_DMA transfer complete byte register (DMA_LCM_DONE)	923
27.9.6 LCM_DMA Send Address Registers (DMA_LCM_TXAx)	923
27.9.7 LCM_DMA Receive Address Registers (DMA_LCM_RXAx)	923
27.10 Example Routines	924
27.10.1 UART1 interrupt mode and computer transceiver test - DMA receive timeout interrupt.....	924
27.10.2 UART1 interrupt mode and computer transceiver test - DMA data check	929
28 Enhanced Dual Data Pointer	937
28.1 Related special function registers	937
28.1.1 1st 16-bit Data Pointer Registers (DPTR0)	937
28.1.3 DPTR control register.....	937
28.1.4 Data Pointer control register (TA)	938
28.2 Example Routines	939
28.2.1 Example Routine 1	939
28.2.2 Example Routine 2	940
29 MDU16 Hardware 16-bit Multiplier and Divider	942
29.1 Registers Related to MDU16	942

29.1.1 Operand 1 Data Registers	942
29.1.2 Operand 2 Data Registers	942
29.1.3 MDU Mode Control Register (ARCON)	943
29.1.4 MDU Operation Control Register (OPCON)	943
29.2 Example Routines	945
Appendix A STC Emulator User Guide	946
Appendix B How to Test I/O Ports.....	953
Appendix C How to Make the Traditional 8051 MCU EVB Emulatable.....	954
Appendix D STC-USB Driver Installation Instructions	956
Appendix E Download Step Demo using USB	1019
Appendix F USB emulation step demonstration	1023
Appendix G RS485 Automatic control or I/O port control circuit diagram	1028
Appendix H STC tool instruction manual	1030
H.1 Overview.....	1030
H.2 System Programmable (ISP) Process Description	1030
H.3 USB type online/offline download tool U8W/U8W-Mini	1031
H3.1 Install U8W/U8W-Mini driver.....	1034
H3.2 U8W function introduction	1036
H3.3 U8W online download instructions.....	1037
F.3.4 U8W offline download instructions.....	1040
H.3.5 U8W-Mini's function introduction	1
H.3.6 U8W-Mini online download instructions	1049
H.3.7 U8W-Mini offline download instructions.....	1050
H.3.8 Make/Update U8W/U8W-Mini	1056
H.3.9 U8W/U8W-Mini set through mode (can be used for simulation).....	1058
H.3.10 Reference circuit of U8W/U8W-Mini	1059
H.4 STC Universal USB to Serial Tool	1061
H.4.1 Appearance of STC Universal USB to Serial Tool	1061
H.4.2 STC general USB to serial tool layout diagram.....	1062
H.4.3 STC Universal USB to Serial Tool Driver Installation.....	1063
H.4.4 Use STC universal USB to serial port tool to download program to MCU.....	1064
H.4.5 Use STC universal USB to serial port tool to simulate user code.....	1066
H.5 Application circuit diagram.....	1074
H.5.1 U8W tool application reference circuit diagram.....	1074
H.5.2 STC Universal USB to Serial Tool Application Reference Circuit Diagram	1075
Appendix I STC Emulation Instruction Manual.....	1076
I.1 Overview	1076
I.2 Install Keil software.....	1077
I.3 Install the emulation driver	1078
I.4 Serial port emulation directly	1081
I.4.1 Make serial port emulation chip.....	1081
I.4.2 Serial port emulation settings in Keil software	1084
I.4.3 Emulation using serial port in Keil software.....	1086
I.5 USB direct emulation (currently only supported by STC8H8K64U-B version chip)	1088

I.5.1 Making a USB emulation chip	1088
I.5.2 USB emulation settings in Keil software	1093
I.5.3 Emulation using USB in Keil software	1095
Appendix J Partial Circuit of RS485 in U8W Download Tool	1097
Appendix K ISP Download Starts Automatically After Receiving User Command While Running User Program (no Power-down)	1098
Appendix L Use STC's IAP series MCU to develop your own ISP program	1100
Appendix J The method of resetting the user program to the system area for ISP download (without power off) 1112	
Appendix N Example Routine of ISP download for STC8H series MCUs using third-party MCU.....	1118
Appendix O Use a third-party application program to call the STC release project program to download the ISP of the MCU	1126
Appendix P STC8H series orthogonal decoding example (courtesy of Chengdu Zhufei Technology) .	1130
Appendix Q Method for Creating Multi-file Projects in Keil	1134
Appendix R Handling of Compilation Error in Keil with Interrupt Numbers Greater Than 31	1138
Appendix S Electrical Characteristics.....	1147
S.1 Absolute Maximum Rating.....	1147
S.2 DC ELECTRICAL CHARACTERISTICS (3.3V).....	1147
S.3 DC ELECTRICAL CHARACTERISTICS (5V).....	1149
S.4 Internal IRC temperature drift characteristic (reference temperature 25 °C).....	1150
S.5 Low voltage reset threshold voltage (test temperature 25 °C).....	1151
Appendix T Application Considerations	1152
T.1 Notes on STC8H series IO ports.....	1152
T.2 Notes on STC8H2K64T series.....	1152
T.3 Notes on STC8H4K64TLR series	1152
T.4 Notes on STC8H8K64U series	1152
Appendix U PCB design guidance for touch keys	1154
Appendix V QFN/DFN packaged components welding method	1156
Appendix W About whether to bake before reflow soldering	1159
Appendix X Precautions for STC8H series MCU to replace STC15 series.....	1160
Appendix Y Precautions for STC8H series single-chip microcomputer to replace STC8A/8F series....	1162
Appendix Z Internally tested models.....	1163
Z.1 STC8H2K64T-35I-LQFP48/QFN48	1163
Z.1.1 Features and Price	1163
Z.1.2 Pinouts.....	1166
Z.2 STC8H4K64LCD-45I-LQFP64/QFN64/LQFP48/QFN48	1167
Z.2.1 Features and Price(Quasi 16-bit MCU with 16-bit hardware multiplier and divider MDU16)	1167
Z.2.2 Pinouts.....	1170
Appendix AA Update Records	1172
Appendix BB STC8 series naming tidbits	1176

1 Overview

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

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

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

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

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

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

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

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

2.1.1 Features and Price

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

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

✓ 1.9V~5.5V

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

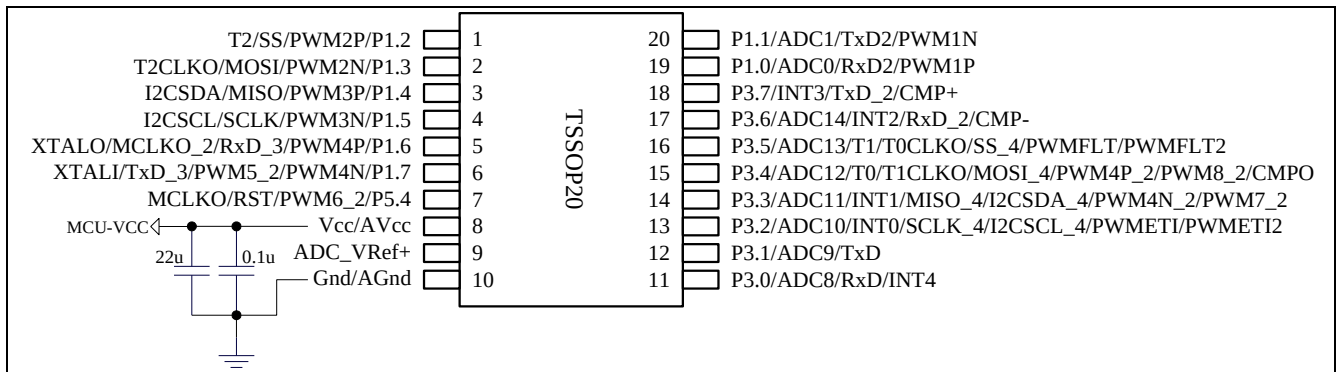
- ✓ Up to 17Kbytes of Flash memory to be used for storing user code
- ✓ Configurable size EEPROM, 512bytes single page for being erased, which can be repeatedly erased more than 100 thousand times.
- ✓ In-System-Programming, ISP in short, can be used to update the application code. No dedicated programmer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

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

- ✓ 128 bytes internal indirect access RAM (IDATA, use keyword *idata* to declare in C language program)
- ✓ 1024 bytes internal extended RAM (internal XDATA, use keyword *xdata* to declare in C language program)
- **Clock**
 - ✓ Internal high precise RC clock(IRC for short, ranges from 4MHz to 36MHz), adjustable while ISP and can be divided to lower frequency by user software, 100KHz for instance.
 - ✓ Error: $\pm 0.3\%$ (at the temperature 25°C)
 - ✓ $-1.35\% \sim +1.30\%$ temperature drift (at the temperature range of -40°C to $+85^{\circ}\text{C}$)
 - ✓ $-0.76\% \sim +0.98\%$ temperature drift (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 sources above can be selected freely by user code.
- **Reset**
 - ✓ Hardware reset
 - ✓ Power-on reset. **(Effective when the chip does not enable the low voltage reset function)**
 - ✓ Reset by reset pin. The default function of P5.4 is the I/O port. P5.4 pin can be set as the reset pin while ISP download. **(Note: When the P5.4 pin is set as the reset pin, the reset level is low.)**
 - ✓ Watch dog timer reset
 - ✓ Low voltage detection reset. 4 low voltage detection levels are provided, 2.0V, 2.4V, 2.7V, 3.0V.
 - ✓ Software reset
 - ✓ Writing the reset trigger register using software
- **Interrupts**
 - ✓ 17 interrupt sources: INT0(Supports rising edge and falling edge interrupt), INT1(Supports rising edge and falling edge interrupt), INT2(Supports falling edge interrupt only), INT3(Supports falling edge interrupt only), INT4(Supports falling edge interrupt only), timer 0, timer 1, timer 2, UART 1, UART 2, ADC, LVD, SPI, I²C, comparator, PWMA, PWMB
 - ✓ 4 interrupt priority levels
 - ✓ Interrupts that can wake up the CPU in clock stop mode: INT0(P3.2), INT1(P3.3), INT2(P3.6), INT3(P3.7), INT4(P3.0), T0(P3.4), T1(P3.5), T2(P1.2), RXD(P3.0/P3.6/P1.6), RXD2(P1.0), I2C_SDA(P1.4/P3.3), Comparator interrupt, LVD interrupt, Power-down wake-up timer.
- **Digital peripherals**
 - ✓ 3 16-bit timers: timer0, timer1, timer2, where the mode 3 of timer 0 has the Non-Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
 - ✓ 2 high speed UARTs: UART1, UART2, whose maximum baudrate clock may be FOSC/4
 - ✓ 8 channels/2 groups of enhanced PWMs, which can realize control signals with dead time, and support external fault detection function. In addition, it also supports 16-bit timers, 8 external interrupts, 8 channels of external capture and pulse width measurement functions.
 - ✓ SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
 - ✓ I²C: Master mode or slave mode are supported.
- **Analog peripherals**
 - ✓ 9 channels (channel 0 to channel 1, channel 8 to channel 14) ultra high speed ADC which supports 10-bit precision. The maximum speed can be 500K(Half a million ADC conversions per second)
 - ✓ Channel 15 of ADC is used to test the internal reference voltage. (The default internal reference voltage is 1.19V when the chip is shipped)
 - ✓ A set of comparator (the CMP+ port and all ADC input ports can be selected as the positive terminal of the comparator, so the comparator can be used as a multi-channel comparator for time division multiplexing)
 - ✓ DAC: 8 channels advanced PWMs timers can be used as 8 channels DAC
- **GPIO**
 - ✓ Up to 17 GPIOs: P1.0~P1.7, P3.0~P3.7, P5.4
 - ✓ 4 modes for all GPIOs: quasi_bidirectional mode, push-pull outputmode, open drain mode, high-impedance input mode
 - ✓ Except for P3.0 and P3.1, all other I/O ports are in a high-impedance state after power-on. User must configure the I/O ports mode before using them. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.
- **Package**
 - ✓ TSSOP20 <6.5mm*6.5mm>, QFN20 <3mm*3mm>

2.1.2 Pinouts



Note:

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



universal USB to UART tool

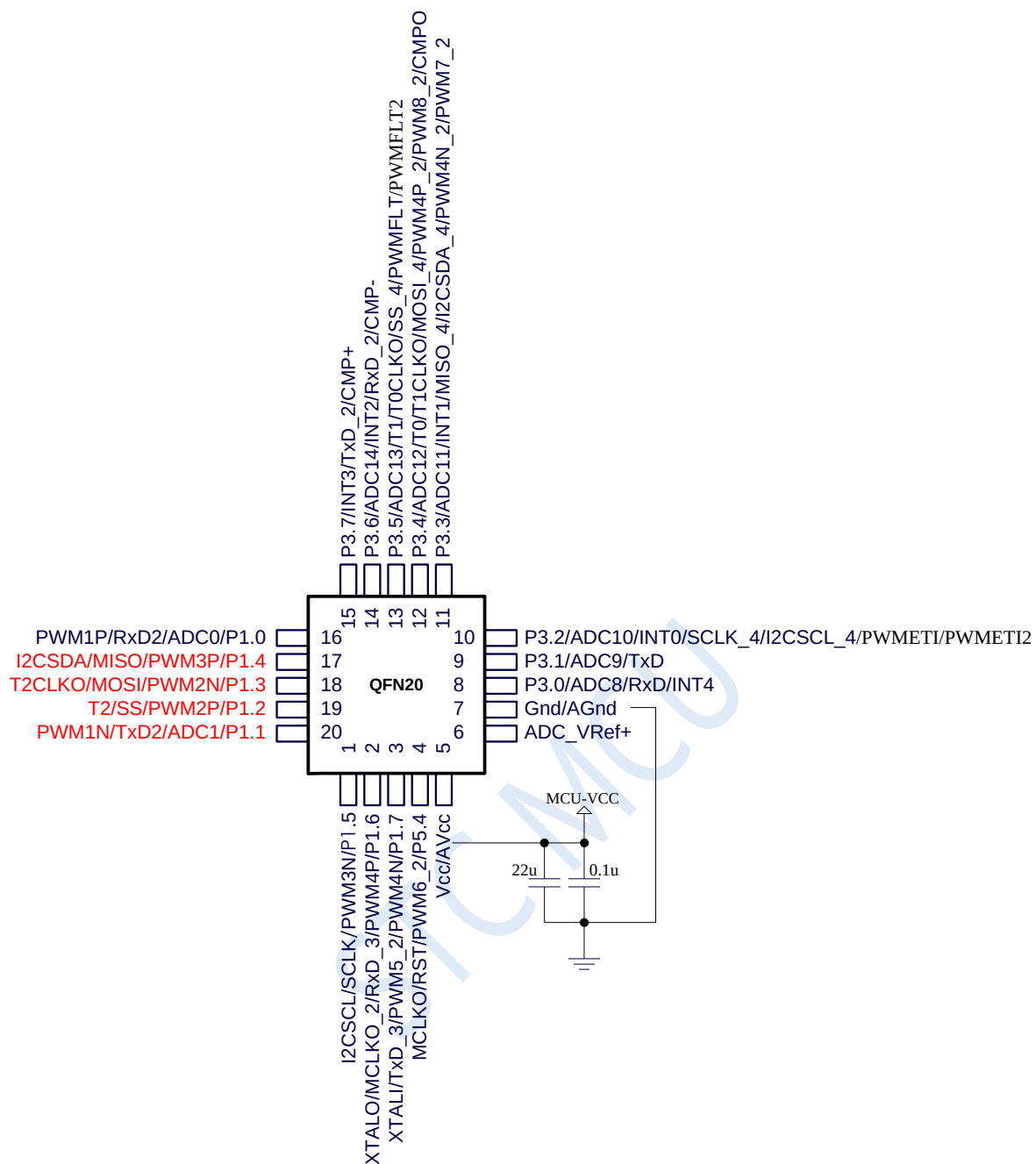
ISP download steps:

1. Connect the universal USB to UART tool to the target chip according to the connection method shown in the figure above.
2. Press the power button to confirm that the target chip is in a power-off state (the power-on LED is off).
Note: When the tool is powered on for the first time, there is no external power supply, so if it is the first time to use this tool, you can skip this step.
3. Click the "Download/Program" button in the STC-ISP download software.
4. Press the power button again to power on the target chip (the power-on LED is on).
5. Start ISP download.

Note: It has been found that when using the USB cable for ISP download, if the USB cable is too thin and the voltage drop on the USB cable is too large, this will result in insufficient power supply during the ISP download. Therefore, please be sure to use the booster USB cable for ISP download.

Note:

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



2.1.3 Pin descriptions

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

Pin number		name	type	description
TSSOP20	QFN20			
13	10	P3.2	I/O	Standard IO port
		ADC10	I	ADC analog input 10
		INT0	I	External interrupt 0
		SCLK_4	I/O	Serial Clock of SPI
		SCL_4	I/O	Serial Clock line of I2C
		PWMET1	I	PWM External trigger input pin
		PWMETI2	I	PWM External trigger input pin 2
14	11	P3.3	I/O	Standard IO port
		ADC11	I	ADC analog input 11
		INT1	I	External interrupt 1
		MISO_4	I/O	Master Input/Slave Output of SPI
		SDA_4	I/O	Serial data line of I2C
		PWM4N_4	I/O	Capture of external signal/ Negative of PWM4 pulse output
		PWM7_2	I/O	Capture of external signal/ Positive of PWM7 pulse output
15	12	P3.4	I/O	Standard IO port
		ADC12	I	ADC analog input 12
		T0	I	Timer0 external input
		T1CLKO	O	Clock out of timer 1
		MOSI_4	I/O	Master Output/Slave Input of SPI
		PWM4P_4	I/O	Capture of external signal/ Positive of PWM4 pulse output
		PWM8_2	I/O	Capture of external signal/ Positive of PWM8 pulse output
		CMPO	O	Comparator output
16	13	P3.5	I/O	Standard IO port
		ADC13	I	ADC analog input 13
		T1	I	Timer1 external input
		T0CLKO	O	Clock out of timer 0
		SS_4	I/O	Slave selection of SPI
		PWMFLT	I	PWMA external anomaly detection pin
		PWMFLT2	I	PWMB external anomaly detection pin
17	14	P3.6	I/O	Standard IO port
		ADC14	I	ADC analog input 14
		INT2	I	External interrupt 2
		RxD_2	I	Serial input of UART1
		CMP-	I	Comparator negative input
18	15	P3.7	I/O	Standard IO port
		INT3	I	External interrupt 3
		TxD_2	O	Serial Transmit pin of UART 1
		CMP+	I	Comparator positive input
19	16	P1.0	I/O	Standard IO port
		ADC0	I	ADC analog input 0
		RxD2	I	Serial input of UART2
		PWM1P	I/O	Capture of external signal/ Positive of PWMA pulse output
20	20	P1.1	I/O	Standard IO port
		ADC1	I	ADC analog input 1
		TxD2	O	Serial Transmit pin of UART 2
		PWM1N	I/O	Capture of external signal/ Negative of PWMA pulse output

2.2 STC8H1K28-36I-LQFP32/QFN32 family

2.2.1 Features and Price

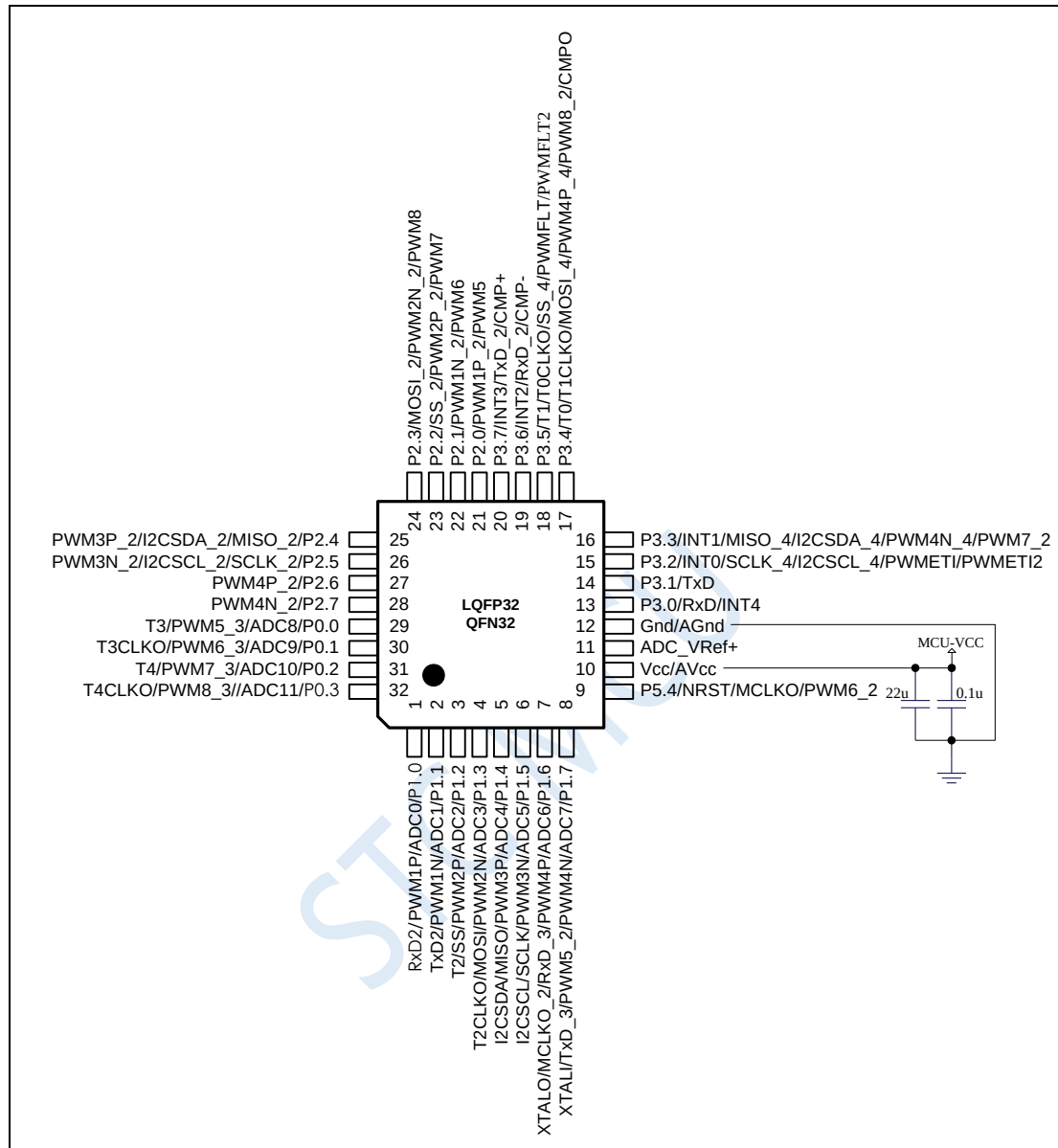
- Selection and price (No external crystal and external reset required)

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

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

- ✓ 128 bytes internal direct access RAM (DATA, use keyword *data* to declare in C language program)
- ✓ 128 bytes internal indirect access RAM (IDATA, use keyword *idata* to declare in C language program)
- ✓ 1024 bytes internal extended RAM (internal XDATA, use keyword *xdata* to declare in C language program)
- **Clock**
 - ✓ Internal high precise RC clock (IRC for short, ranges from 4MHz to 38MHz), adjustable while ISP and can be divided to lower frequency by user software, 100KHz for instance.
 - ✓ Error: $\pm 0.3\%$ (at the temperature 25°C)
 - ✓ -1.35%~+1.30% temperature drift (at the temperature range of -40 °C to +85 °C)
 - ✓ -0.76%~+0.98% temperature drift (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 sources above can be selected freely by user code.
- **Reset**
 - ✓ Hardware reset
 - ✓ Power-on reset. (**Effective when the chip does not enable the low voltage reset function**)
 - ✓ Reset by reset pin. The default function of P5.4 is the I/O port. P5.4 pin can be set as the reset pin while ISP download. (**Note: When the P5.4 pin is set as the reset pin, the reset level is low.**)
 - ✓ Watch dog timer reset
 - ✓ Low voltage detection reset. 4 low voltage detection levels are provided, 2.0V, 2.4V, 2.7V, 3.0V.
 - ✓ Software reset
 - ✓ Writing the reset trigger register using software
- **Interrupts**
 - ✓ 19 interrupt sources: INT0(Supports rising edge and falling edge interrupt), INT1(Supports rising edge and falling edge interrupt), INT2(Supports falling edge interrupt only), INT3(Supports falling edge interrupt only), INT4(Supports falling edge interrupt only), timer0, timer1, timer2, timer3, timer4, UART1, UART2, ADC, LVD, SPI, I²C, Comparator, PWMA, PWMB
 - ✓ 4 interrupt priority levels
 - ✓ Interrupts that can wake up the CPU in clock stop mode: INT0(P3.2), INT1(P3.3), INT2(P3.6), INT3(P3.7), INT4(P3.0), T0(P3.4), T1(P3.5), T2(P1.2), T3(P0.4), T4(P0.6), RXD(P3.0/P3.6/P1.6), RXD2(P1.0), I2C_SDA(P1.4/P2.4/P3.3), Comparator, LVD, Power-down wake-up timer.
- **Digital peripherals**
 - ✓ 5 16-bit timers: timer0, timer1, timer2, timer3, timer4, where the mode 3 of timer 0 has the Non-Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
 - ✓ 2 high speed UARTs: UART1, UART2, whose maximum baudrate may be FOSC/4.
 - ✓ 8 channels/2 groups of enhanced PWMs, which can realize control signals with dead time, and support external fault detection function. In addition, it also supports 16-bit timers, 8 external interrupts, 8 external captures and pulse width measurement functions.
 - ✓ SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
 - ✓ I²C: Master mode or slave mode are supported.
- **Analog peripherals**
 - ✓ 12 channels (channel 0 to channel 11) ultra high speed ADC which supports 10-bit precision. The maximum speed can be 500K(Half a million ADC conversions per second)
 - ✓ Channel 15 of ADC is used to test the internal reference voltage. (The default internal reference voltage is 1.19V when the chip is shipped)
 - ✓ A set of comparator (the CMP+ port and all ADC input ports can be selected as the positive terminal of the comparator, so the comparator can be used as a multi-channel comparator for time division multiplexing)
 - ✓ DAC: 8 channels advanced PWM timer can be used as 8 channels DAC
- **GPIO**
 - ✓ Up to 29 GPIOs: P0.0~P0.3, P1.0~P1.7, P2.0~P2.7, P3.0~P3.7, P5.4
 - ✓ 4 modes for all GPIOs: quasi_bidirectional mode, push-pull outputmode, open drain mode, high-impedance input mode
 - ✓ Except for P3.0 and P3.1, all other I/O ports are in a high-impedance state after power-on. User must configure the I/O ports mode before using them. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.
- **Package**
 - ✓ LQFP32 <9mm*9mm>, QFN32 <4mm*4mm>

2.2.2 Pinouts



Note:

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

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

Note:

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

2.2.3 Pin descriptions

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

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

Pin number	name	type	description

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

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

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

STC MCU

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

products supply information			Available				
Price & Package	TSSOP20 <6.5mm*6.5mm>			✓	✓	✓	✓
	QFN32<4mm*4mm>			✓	✓	✓	✓
	LQFP32<9mm*9mm>			✓	✓	✓	✓
	QFN48 <6mm*6mm>			✓	✓	✓	✓
	LQFP48 <9mm*9mm>			✓	✓	✓	✓
Online debug itself				✓	✓	✓	✓
Support software USB download directly				✓	✓	✓	✓
Support RS485 download				✓	✓	✓	✓
Password can be set for next update				✓	✓	✓	✓
Program encrypted transmission (Anti-blocking)				✓	✓	✓	✓
Clock output and Reset				✓	✓	✓	✓
Internal high precision Clock (adjustal under 45MHz)				✓	✓	✓	✓
Internal high reliable reset circuit with 4 levels optional reset threshold voltage				✓	✓	✓	✓
Watch-dog Timer				✓	✓	✓	✓
Internal LVD interrupt (can wake-up CPU)				✓	✓	✓	✓
Comparator (May be used as ADC to detect external power-down)				✓	✓	✓	✓
12-bit high speed ADC (8 PWMs can be used as 8 DACs)				12bit	✓	✓	✓
Power-down Wake-up timer				✓	✓	✓	✓
16-bit advanced PWM timer with Complementary symmetrical dead-time				✓	✓	✓	✓
Timers/Counters (T0-T4 Pin Can wake-up CPU)				5	✓	✓	✓
MDU16 (Hardware 16-bit Multiplier and Divider)				✓	✓	✓	✓
I²C which can wake-up CPU				✓	✓	✓	✓
SPI which can wake-up CPU				✓	✓	✓	✓
UARTs which can wake-up CPU				2	✓	✓	✓
All I/O ports support interrupts and can wake up MCU				✓	✓	✓	✓
Traditional I/O interrupt(INT0/INT1/INT2/INT3/INT4) (can wake-up CPU)				✓	✓	✓	✓
Maximum I/O Lines			43	✓	✓	✓	✓
EEPROM 100 thousand times) (Byte)			32K	✓	✓	✓	✓
Enhanced Dual DPTR increasing or decreasing			2	✓	✓	✓	✓
xdata Internal extended SRAM (Byte)			3K	✓	✓	✓	✓
idata Internal DATA RAM			256	✓	✓	✓	✓
Flash Code Memory (100 thousand times) (Byte)			48K	✓	✓	✓	✓
Operating voltage (V)			1.9-5.5	✓	✓	✓	✓
MCU			STC8H3K32S2	1.9-5.5	✓	✓	✓
			STC8H3K48S2	1.9-5.5	✓	✓	✓
			STC8H3K60S2	1.9-5.5	✓	✓	✓
			STC8H3K64S2	1.9-5.5	✓	✓	✓

- 34 -

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

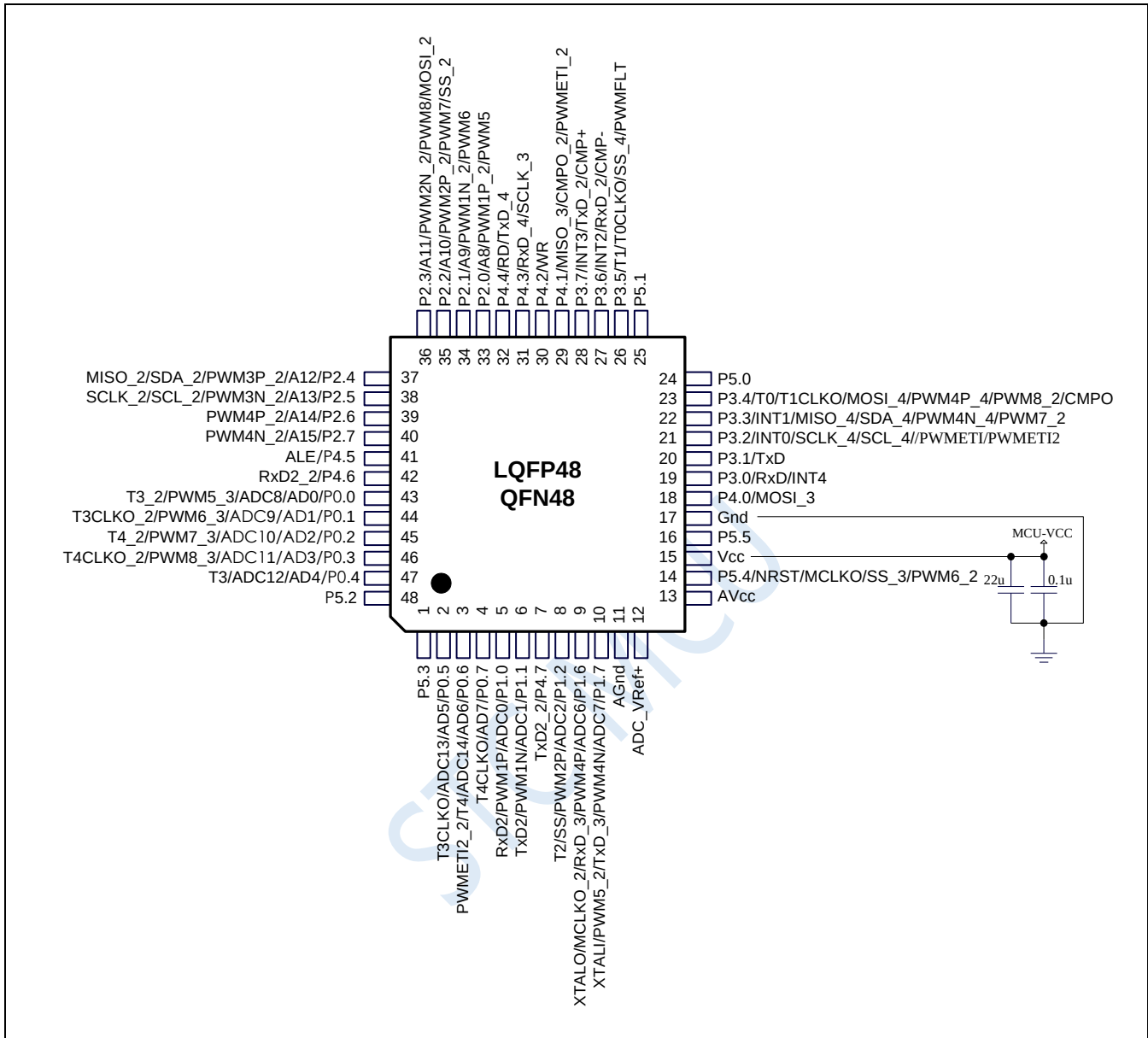
The three clock sources above can be selected freely by user code.

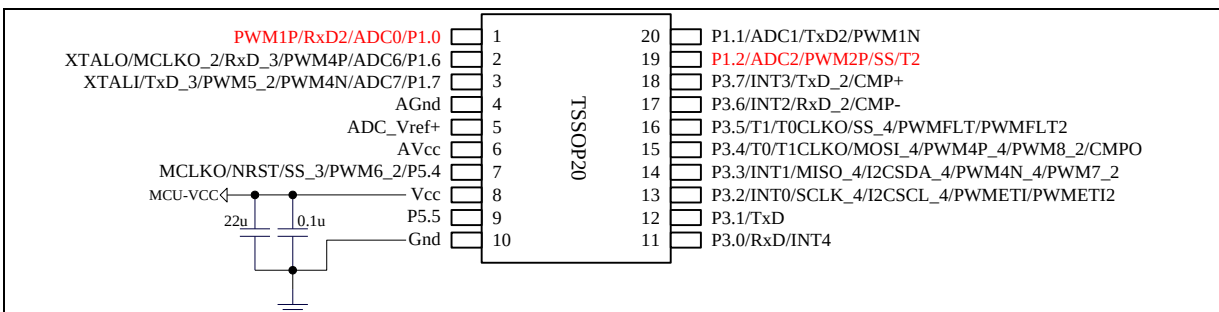
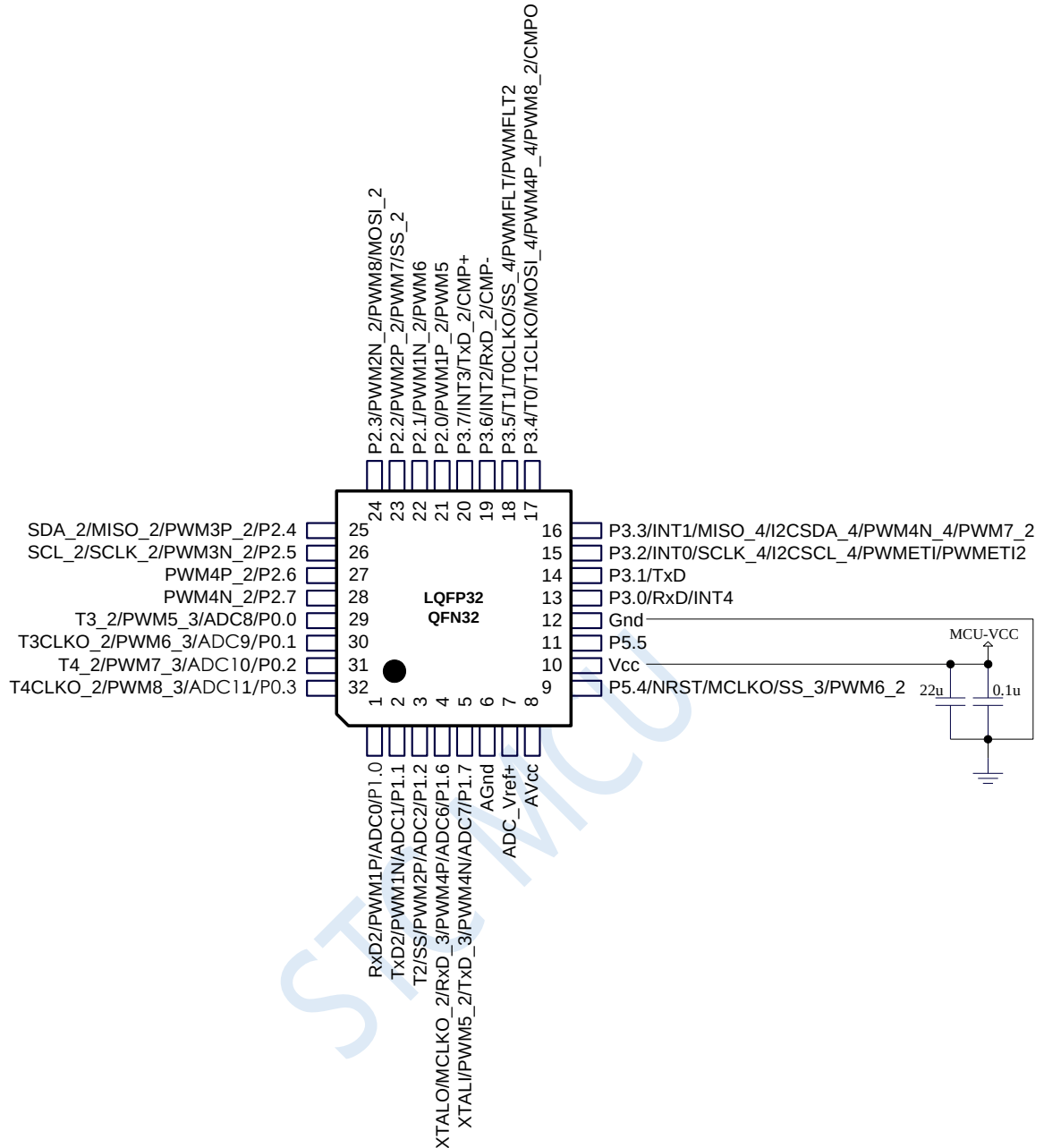
 - ✓ Important note about the internal high-speed IRC of STC8H3K64S2 series B version products
 - ✓ Due to manufacturing reasons, the internal high-speed IRC of some chips may have a blind area between 34MHz and 36MHz. It is recommended not to set the operating frequency in this area.
 - ✓ The temperature drift of the internal high-speed IRC at low temperature is larger than that at the higher temperature, and the temperature drift in the low frequency range is larger than that in the high frequency range. Generally, the operating frequency of 20MHz~40MHz, the temperature drift at 85℃ can be controlled within 0.8%.
- **Reset**
 - ✓ Hardware reset
 - ✓ Power-on reset. (Effective when the chip does not enable the low voltage reset function)
 - ✓ Reset by reset pin. The default function of P5.4 is the I/O port. The P5.4 pin can be set as the reset pin while ISP download. (Note: When the P5.4 pin is set as the reset pin, the reset level is low.)
 - ✓ Watch dog timer reset
 - ✓ Low voltage detection reset. 4 low voltage detection levels are provided, 2.0V, 2.4V, V2.7, V3.0.
 - ✓ Software reset
 - ✓ Writing the reset trigger register using software
- **Interrupts**
 - ✓ 19 interrupt sources: INT0(Supports rising edge and falling edge interrupt), INT1(Supports rising edge and falling edge interrupt), INT2(Supports falling edge interrupt only), INT3(Supports falling edge interrupt only), INT4(Supports falling edge interrupt only), timer 0, timer 1, timer 2, timer 3, timer 4, UART 1, UART 2, ADC, LVD, SPI, I²C, comparator, PWMA, PWMB
 - ✓ 4 interrupt priority levels
 - ✓ Interrupts that can wake up the CPU in clock stop mode: INT0(P3.2), INT1(P3.3), INT2(P3.6), INT3(P3.7), INT4(P3.0), T0(P3.4), T1(P3.5), T2(P1.2), T3(P0.4), T4(P0.6), RXD(P3.0/P3.6/P1.6/P4.3), RXD2(P1.0/P4.6), I2C_SDA(P1.4/P2.4/P3.3), Comparator interrupt, LVD interrupt, Power-down wake-up timer.
- **Digital peripherals**
 - ✓ 5 16-bit timers: timer0, timer1, timer2, timer3, timer4, where the mode 3 of timer 0 has the Non-Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
 - ✓ 2 high speed UARTs: UART1, UART2, whose maximum baudrate clock may be FOSC/4
 - ✓ 8 channels/2 groups of enhanced PWMs, which can realize control signals with dead time, and support external fault detection function. In addition, it also supports 16-bit timers, 8 external interrupts, 8 external captures and pulse width measurement functions.
 - ✓ SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
 - ✓ I²C: Master mode or slave mode are supported.
 - ✓ MDU16: Hardware 16-bit Multiplier and Divider which supports 32-bit divided by 16-bit, 16-bit divided by 16-bit, 16-bit multiplied by 16-bit, data shift, and data normalization operations.
 - ✓ I/O port interrupt: All I/Os support interrupts, each group of I/O interrupts has an independent interrupt entry address, all I/O interrupts can support 4 types interrupt mode: high level interrupt, low level interrupt, rising edge interrupt, falling edge interrupt. (Note: The I/O port interrupts of the STC8H3K64S2 series A version of the chip cannot wake up CPU from power-down. The I/O port interrupts of the B version chip can wake up CPU from power-down, but only have one level of interrupt priority)
- **Analog peripherals**
 - ✓ Ultra high speed ADC which supports 12-bit precision 12 channels analog-to-digital convertor (channel 0 to channel 2, channel 6 to channel 14. No channel 3 and channel 5 because P1.3, P1.4 and P1.5 do not exist). The maximum speed can be 800K(800K ADC conversions per second)
 - ✓ Channel 15 of ADC is used to test the internal reference voltage. (The default internal reference voltage is 1.19V when the chip is shipped)
 - ✓ A set of comparator (the CMP+ port and all ADC input ports can be selected as the positive terminal of the comparator, so the comparator can be used as a multi-channel comparator for time division multiplexing)
 - ✓ DAC: 8 channels advanced PWMs timers can be used as 8 channels DAC
- **GPIO**
 - ✓ Up to 43 GPIOs: P0.0~P0.7, P1.0~P1.2, P1.6~P1.7, P2.0~P2.7, P3.0~P3.7, P4.0~P4.7, P5.0~P5.5
 - ✓ 4 modes for all GPIOs: quasi_bidirectional mode, push-pull output mode, open drain mode, high-impedance input mode
 - ✓ Except for P3.0 and P3.1, all other I/O ports are in a high-impedance state after power-on. User must set the I/O ports mode before using them. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.
- **Package**
 - ✓ LQFP48 <9mm*9mm>, QFN48 <6mm*6mm>, LQFP32 <9mm*9mm>, QFN32 <4mm*4mm>, TSSOP20

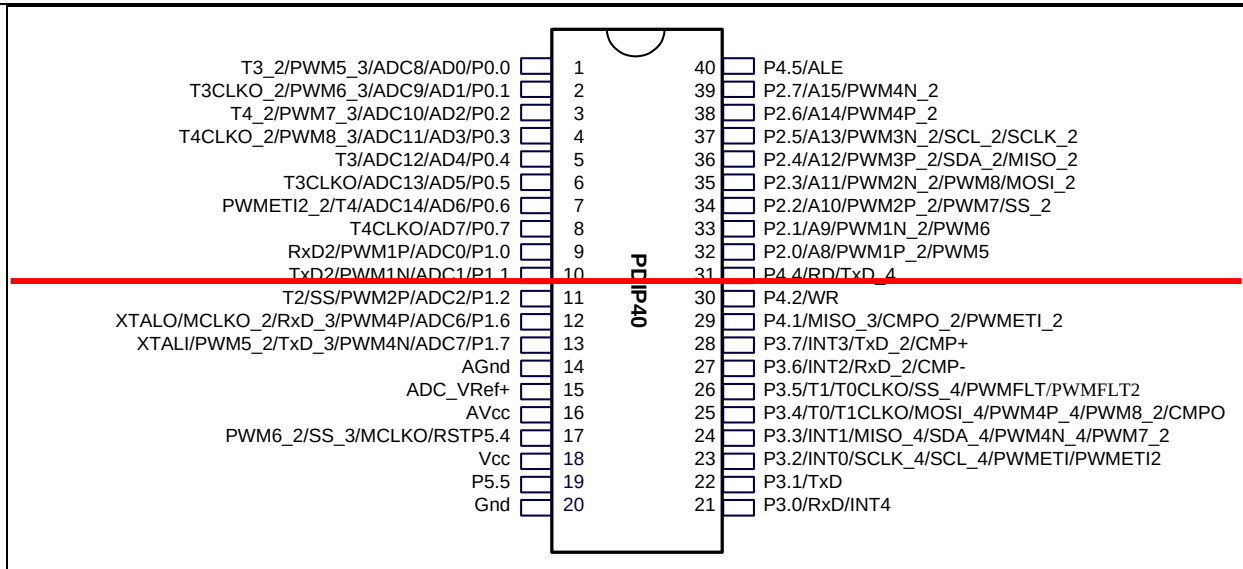
<6.5mm*6.5mm> (There are no samples for LQFP32, QFN32, TSSOP2 at the moment, there will be later, please order in advance if necessary.)

STC MCU

2.3.2 Pinouts





**Note:**

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

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

2.3.3 Pin descriptions

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

Pin number				name	type	description
LQFP48 QFN48	LQFP32 QFN32	TSSOP20				
		0				

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

Pin number				name	type	description
LQFP48 QFN48	LQFP32 QFN32	TSSOP20				
28	20	18		P3.7	I/O	Standard IO port
				INT3	I	External interrupt3
				TxD_2	O	Serial Transmit pin of UART 1

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

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

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

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

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

- 44 -

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

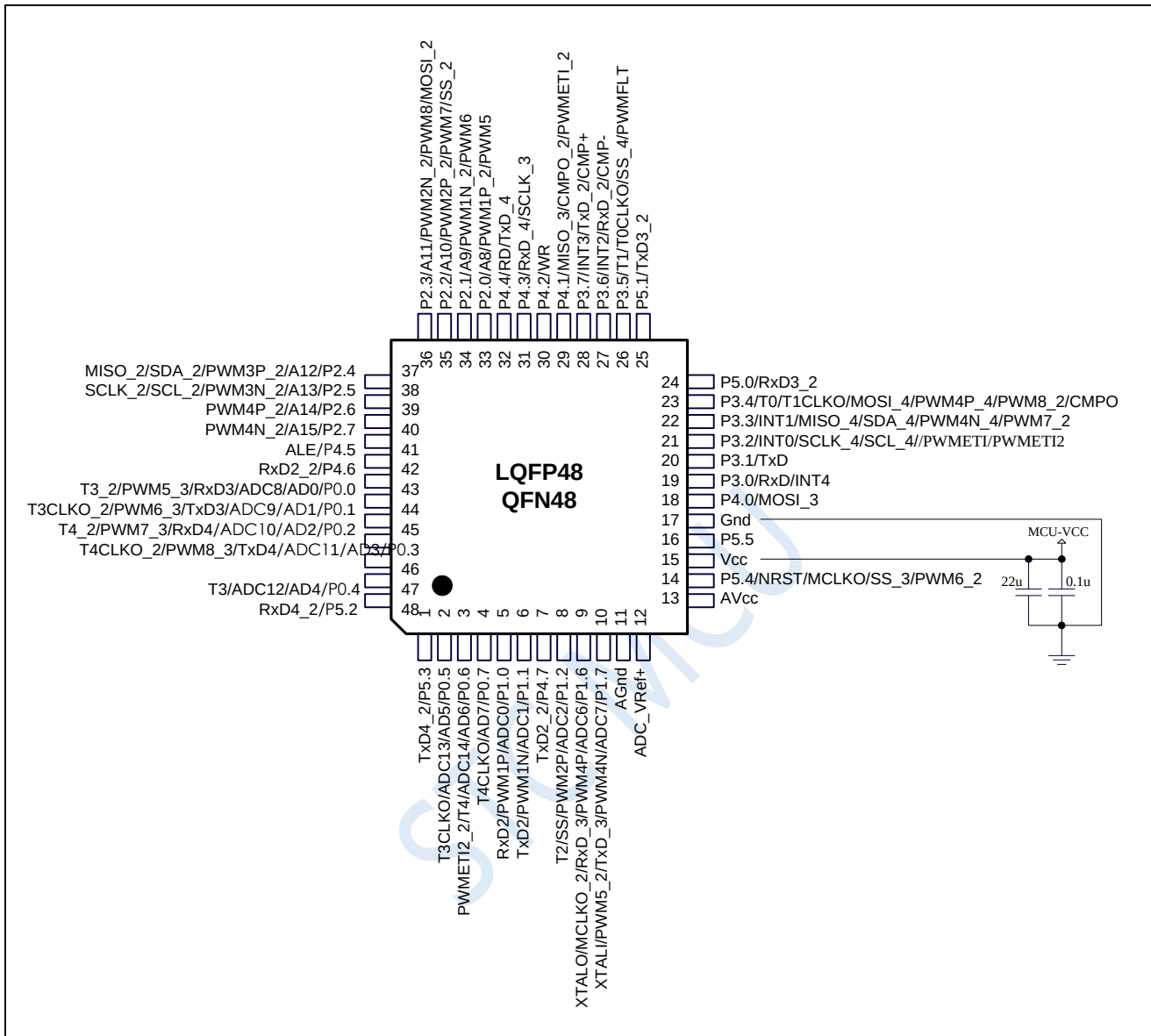
The three clock sources above can be selected freely by user code.

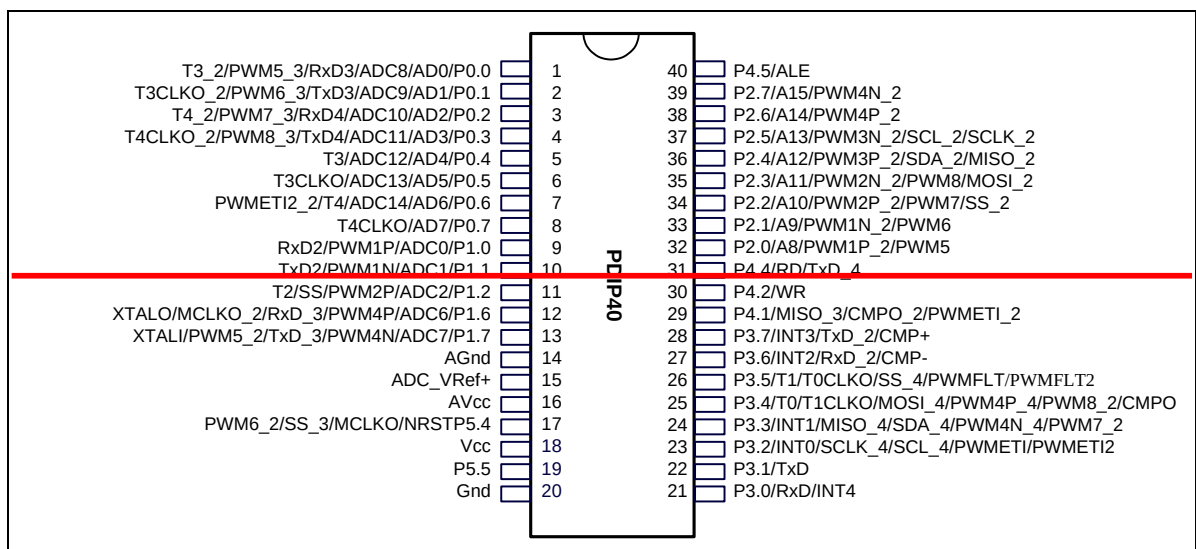
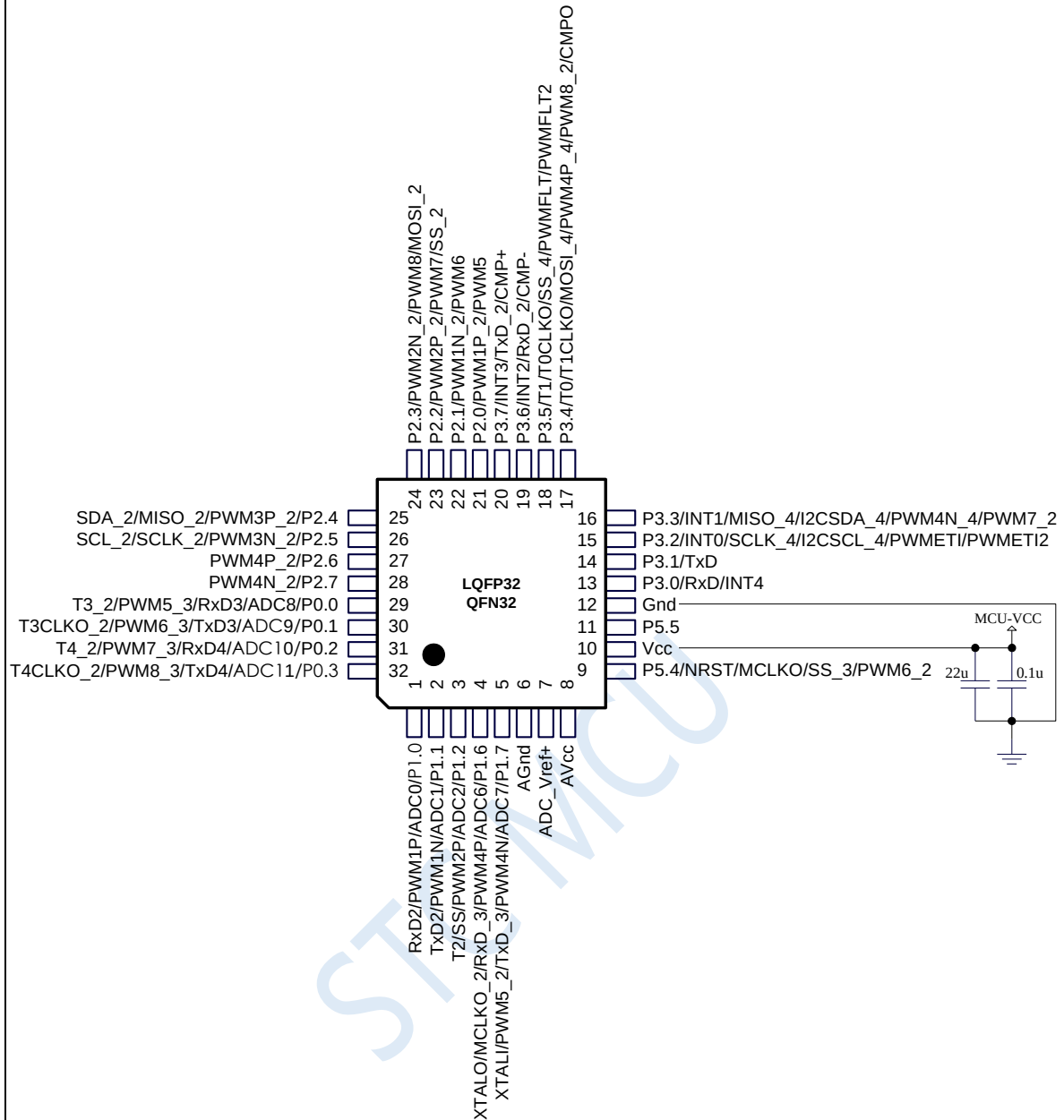
 - ✓ Important note about the internal high-speed IRC of STC8H3K64S4 series B version products
 - ✓ Due to manufacturing reasons, the internal high-speed IRC of some chips may have a blind area between 34MHz and 36MHz. It is recommended not to set the operating frequency in this area.
 - ✓ The temperature drift of the internal high-speed IRC at low temperature is larger than that at the higher temperature, and the temperature drift in the low frequency range is larger than that in the high frequency range. Generally, the operating frequency of 20MHz~40MHz, the temperature drift at 85℃ can be controlled within 0.8%.
- **Reset**
 - ✓ Hardware reset
 - ✓ Power-on reset.(Effective when the chip does not enable the low voltage reset function)
 - ✓ Reset by reset pin. The default function of P5.4 is the I/O port. The P5.4 pin can be set as the reset pin while ISP download. (Note: When the P5.4 pin is set as the reset pin, the reset level is low.)
 - ✓ Watch dog timer reset
 - ✓ Low voltage detection reset. 4 low voltage detection levels are provided, 2.0V, 2.4V, V2.7, V3.0.
 - ✓ Software reset
 - ✓ Writing the reset trigger register using software
- **Interrupts**
 - ✓ 21 interrupt sources: INT0(Supports rising edge and falling edge interrupt), INT1(Supports rising edge and falling edge interrupt), INT2(Supports falling edge interrupt only), INT3(Supports falling edge interrupt only), INT4(Supports falling edge interrupt only), timer 0, timer 1, timer 2, timer 3, timer 4, UART 1, UART 2, UART 3, UART 4, ADC, LVD, SPI, I²C, comparator, PWMA, PWMB
 - ✓ 4 interrupt priority levels
 - ✓ interruption that can wake up the CPU in clock stop mode: INT0(P3.2), INT1(P3.3), INT2(P3.6), INT3(P3.7), INT4(P3.0), T0(P3.4), T1(P3.5), T2(P1.2), T3(P0.4), T4(P0.6), RXD(P3.0/P3.6/P1.6/P4.3), RXD2(P1.0/P4.6), RXD3(P0.0/P5.0), RXD4(P0.2/P5.2), I2C_SDA(P1.4/P2.4/P3.3), Comparator interrupt, LVD interrupt, Power-down wake-up timer.
- **Digital peripherals**
 - ✓ 5 16-bit timers: timer0, timer1, timer2, timer3, timer4, where the mode 3 of timer 0 has the Non-Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
 - ✓ 4 high speed UARTs: UART1, UART 2, UART 3, UART 4, whose maximum baudrate may be FOSC/4.
 - ✓ 8 channels/2 groups of enhanced PWMs, which can realize control signals with dead time, and support external fault detection function. In addition, it also supports 16-bit timers, 8 external interrupts, 8 external captures and pulse width measurement functions.
 - ✓ SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
 - ✓ I²C: Master mode or slave mode are supported.
 - ✓ MDU16: Hardware 16-bit Multiplier and Divider which supports 32-bit divided by 16-bit, 16-bit divided by 16-bit, 16-bit multiplied by 16-bit, data shift, and data normalization operations.
 - ✓ I/O port interrupt: All I/Os support interrupts, each group of I/O interrupts has an independent interrupt entry address, all I/O interrupts can support 4 types interrupt mode: high level interrupt, low level interrupt, rising edge interrupt, falling edge interrupt. (Note: The I/O port interrupts of the STC8H3K64S4 series A version of the chip cannot wake up CPU from power-down. The I/O port interrupts of the B version chip can wake up CPU from power-down, but only have one level of interrupt priority)
- **Analog peripherals**
 - ✓ Ultra high speed ADC which supports 12-bit precision 12 channels analog-to-digital convertors (channel 0 to channel 2, channel 6 to channel 14. No channel 3 and channel 5 because P1.3, P1.4 and P1.5 do not exist). The maximum speed can reach 800K(800K ADC conversions per second)
 - ✓ Channel 15 of ADC is used to test the internal reference voltage. (The default internal reference voltage is 1.19V when the chip is shipped)
 - ✓ A set of comparator (the CMP+ port and all ADC input ports can be selected as the positive terminal of the comparator, so the comparator can be used as a multi-channel comparator for time division multiplexing)
 - ✓ DAC: 8 channels advanced PWMs timers can be used as 8 channels DAC
- **GPIO**
 - ✓ Up to 43 GPIOs: P0.0~P0.7, P1.0~ P1.2, P1.6~ P1.7, P2.0~P2.7, P3.0~P3.7, P4.0~P4.7, P5.0~P5.5
 - ✓ 4 modes for all GPIOs: quasi_bidirectional mode, push-pull outputmode, open drain mode, high-impedance input mode
 - ✓ Except for P3.0 and P3.1, all other I/O ports are in a high-impedance state after power-on. User must set the I/O ports mode before using them. In addition, , the internal 4K pull-up resistor of every I/O can be enabled independently.
- **Package**
 - ✓ LQFP48 <9mm*9mm>, QFN48 <6mm*6mm>, LQFP32 <9mm*9mm>, QFN32 <4mm*4mm> (There are

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

STC MCU

2.4.2 Pinouts





Note:

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

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

STC MCU

2.4.3 Pin descriptions

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

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

14	9	P5.4	I/O	Standard IO port
		NRST	I	Reset pin(low level reset)
		MCLKO	O	Main clock output
		SS_3	I	Slave selection of SPI (it is output with regard to master)
		PWM6_2	I/O	Capture of external signal/Pulse output of PWM6
15	10	Vcc	Vcc	Power supply
16	11	P5.5	I/O	Standard IO port
17	12	Gnd	Gnd	Ground
18		P4.0	I/O	Standard IO port
		MOSI_3	I/O	Master Output/Slave Input of SPI
19	13	P3.0	I/O	Standard IO port
		RxD	I	Input of UART1
		INT4	I	External interrupt 4
20	14	P3.1	I/O	Standard IO port
		TxD	O	Transmit pin of UART 1
21	15	P3.2	I/O	Standard IO port
		INT0	I	External interrupt 0
		SCLK_4	I/O	Serial Clock of SPI
		SCL_4	I/O	Serial Clock line of I2C
		PWMET1	I	PWM external trigger input pin
		PWMET2	I	PWM external trigger input pin 2
22	16	P3.3	I/O	Standard IO port
		INT1	I	External interrupt 1
		MISO_4	I/O	Master Input/Slave Output of SPI
		SDA_4	I/O	Serial data line of I2C
		PWM4N_4	I/O	Capture of external signal/Pulse negative output of PWM4
		PWM7_2	I/O	Capture of external signal/ Pulse output of PWM7
23	17	P3.4	I/O	Standard IO port
		T0	I	Timer0 external input
		T1CLKO	O	Clock out of timer 1
		MOSI_4	I/O	Master Output/Slave Input of SPI
		PWM4P_4	I/O	Capture of external signal/ Pulse positive output of PWM4
		PWM8_2	I/O	Capture of external signal/Pulse output of PWM8
		CMPO	O	Comparator output
24		P5.0	I/O	Standard IO port
		RxD3_2	I	Input of UART3
25		P5.1	I/O	Standard IO port
		TxD3_2	O	Transmit pin of UART 3
26	18	P3.5	I/O	Standard IO port
		T1	I	Timer1 external input
		T0CLKO	O	Clock out of timer 0
		SS_4	I	Slave selection of SPI (it is output with regard to master)
		PWMFLT	I	Enhance PWM external anomaly detection pin

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

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

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

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

- 54 -

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

➤ Clock

- ✓ Internal high precise RC clock IRC(IRC for short, ranges from 4MHz to 45MHz), adjustable while ISP and can be divided to lower frequency by user software, 100KHz for instance.
 - ✓ Error: $\pm 0.3\%$ (at the temperature 25℃)
 - ✓ $-1.35\% \sim +1.30\%$ temperature drift (at the temperature range of -40℃ to +85℃)
 - ✓ $-0.76\% \sim +0.98\%$ temperature drift (at the temperature range of -20℃ to 65℃)
 - ✓ Internal 32KHz low speed IRC with large error
 - ✓ External crystal (4MHz~33MHz) and external clock
- Users can freely choose the above 3 clock sources

➤ Reset

- ✓ Hardware reset
 - ✓ Power-on reset. Measured voltage is 1.69V~1.82V. (Effective when the chip does not enable the low voltage reset function)
The power-on reset voltage is a voltage range consisting of an upper limit voltage and a lower limit voltage. When the operating voltage drops from 5V / 3.3V to the lower limit threshold voltage of the power-on reset, the chip is in a reset state; when the voltage rises from 0V to the upper threshold voltage of power-on reset, the chip is released from the reset state.
 - ✓ Reset by reset pin. The default function of P5.4 is the I/O port. The P5.4 pin can be set as the reset pin while ISP download. (Note: When the P5.4 pin is set as the reset pin, the reset level is low.)
 - ✓ Watch dog timer reset
 - ✓ Low voltage detection reset. 4 low voltage detection levels are provided, 1.9V, 2.3V, 2.8V, 3.0V. Each level of low-voltage detection voltage is a voltage range consisting of an upper limit voltage and a lower limit voltage. When the operating voltage drops from 5V / 3.3V to the lower limit threshold voltage of low-voltage detection, the low-voltage detection takes effect. When the voltage rises from 0V to the upper threshold voltage, the low voltage detection becomes effective.
- ✓ Software reset
 - ✓ Writing the reset trigger register using software

➤ Interrupts

- ✓ 22 interrupt sources: INT0(Supports rising edge and falling edge interrupt), INT1(Supports rising edge and falling edge interrupt), INT2(Supports falling edge interrupt only), INT3(Supports falling edge interrupt only), INT4(Supports falling edge interrupt only), timer 0, timer 1, timer 2, timer 3, timer 4, UART 1, UART 2, UART 3, UART 4, ADC, LVD, SPI, I²C, comparator, PWMA, PWMB, USB
- ✓ 4 interrupt priority levels
- ✓ Interrupts that can wake up the CPU in clock stop mode: INT0(P3.2), INT1(P3.3), INT2(P3.6), INT3(P3.7), INT4(P3.0), T0(P3.4), T1(P3.5), T2(P1.2), T3(P0.4), T4(P0.6), RXD(P3.0/P3.6/P1.6/P4.3), RXD2(P1.0/P4.6), RXD3(P0.0/P5.0), RXD4(P0.2/P5.2), I2C_SDA(P1.4/P2.4/P3.3), Comparator interrupt, LVD interrupt, Power-down wake-up timer.

➤ Digital peripherals

- ✓ 5 16-bit timers: timer0, timer1, timer2, timer3, timer4, where the mode 3 of timer 0 has the Non Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
- ✓ 4 high speed UARTs: UART1, UART2, UART3, UART4, whose maximum baudrate clock may be FOSC/4
- ✓ 8 channels/2 groups of enhanced PWM, which can realize control signals with dead time, and support external fault detection function. In addition, supports 16-bit timers, 8 external interrupts, 8 external captures and pulse width measurement functions.
- ✓ SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
- ✓ I²C: Master mode or slave mode are supported.
- ✓ MDU16: Hardware 16-bit Multiplier and Divider which supports 32-bit divided by 16-bit, 16-bit divided by 16-bit, 16-bit multiplied by 16-bit, data shift, and data normalization operations.
- ✓ USB: USB2.0/USB1.1 compatible with full speed USB, 6 two-way endpoints, support 4 endpoint transmission modes (control transmission, interrupt transmission, batch Quantum and synchronous transfers), each endpoint has a 64 byte buffer
- ✓ RTC: Support year, month, day, hour, minute, second, sub-second (1/128 second). And supports clock interrupt and a set of alarm clocks (Note: A version of the chip does not have this function)
- ✓ I/O port interrupt: All I/Os support interrupts, each group of I/O interrupts has an independent interrupt entry address, all I/O interrupts can support 4 types interrupt mode: high level interrupt, low level interrupt, rising edge interrupt, falling edge interrupt. Provides 4 levels of interrupt priority and supports power-down wake-up function. (Note: A version of the chip does not have this function)
- ✓ DMA: support Memory-To-Memory, SPI, UART1TX/UART1RX, UART2TX/UART2RX, UART3TX/UART3RX, UART4TX/UART4RX, ADC(Automatically calculates the average of multiple ADC results), LCM (Note: A version of the chip does not have this function)
- ✓ LCM (TFT color screen) driver: support 8080 and 6800 interface, and support 8-bit and 16-bit data width (Note: A version of the chip does not have this function)
 - ✓ 8 bits 8080 data bus: 8 bits data lines (TD0~TD7), READ signal (TRD), WRITE signal (TWR), RS line (TRS)
 - ✓ 16 bits 8080 bus: 16 bits data lines (TD0~TD15), READ signal (TRD), WRITE signal (TWR), RS line (TRS)
 - ✓ 8 bits 6800 bus: 8 bits data lines (TD0~TD7), enable signal (TE), READ and WRITE signal (TRW), RS line (TRS)
 - ✓ 16 bits 6800 bus: 16 bits data lines (TD0~TD15), enable signal (TE), READ and WRITE signal (TRW), RS line (TRS)
 - ✓ Note: If you use 8-bit data lines to control the TFT screen, you generally need TD0~D7, TRD/TWR/TRS, 11 data and

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

➤ **Analog peripherals**

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

➤ **GPIO**

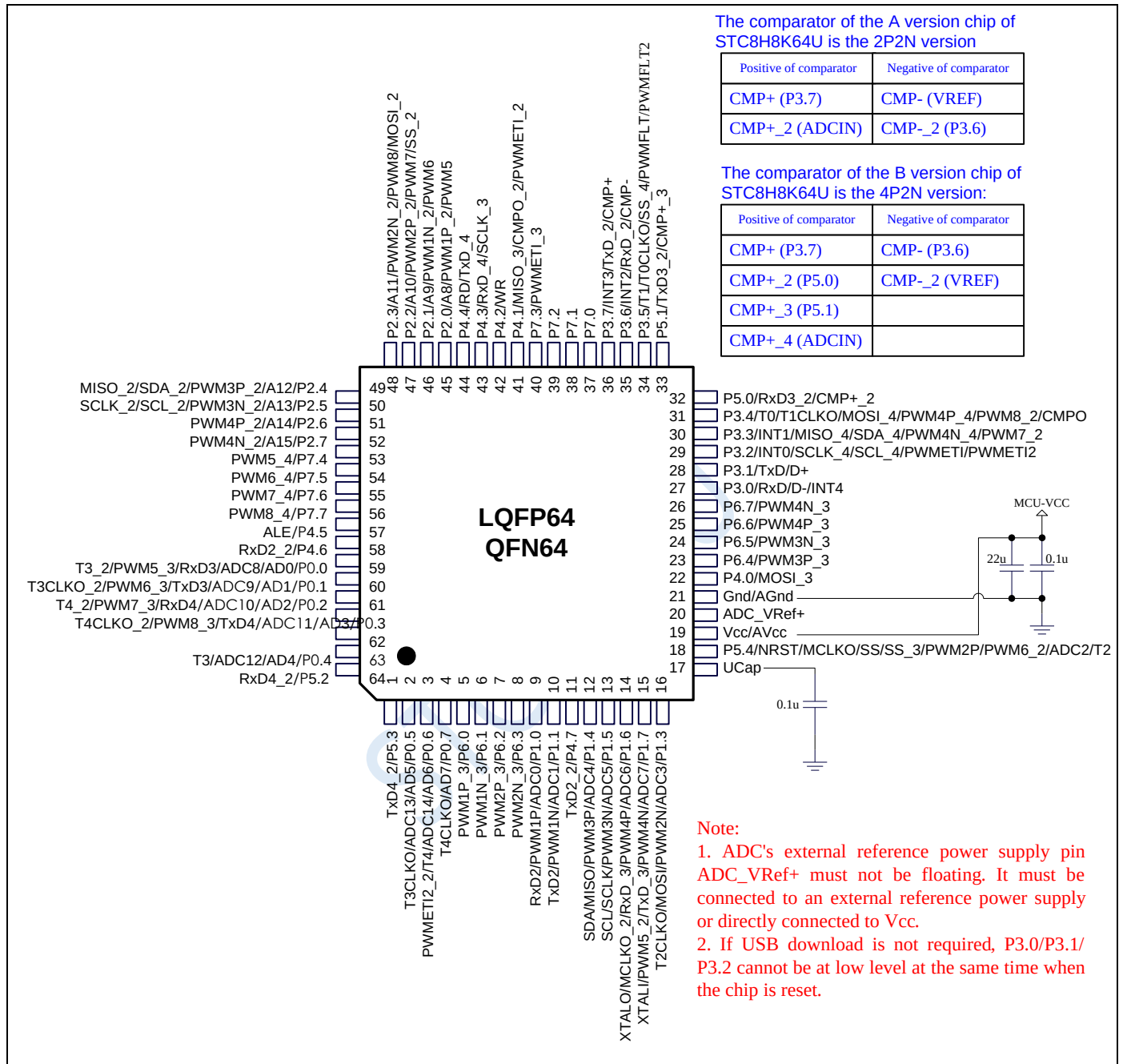
- ✓ Up to 61 GPIOs: P0.0~P0.7, P1.0~ P1.7(no P1.2), P2.0~P2.7, P3.0~P3.7, P4.0~P4.7, P5.0~P5.4, P6.0~P6.7, P7.0~P7.7
- ✓ 4 modes for all GPIOs: quasi_bidirectional mode, push-pull outputmode, open drain mode, high-impedance input mode
- ✓ Except for P3.0 and P3.1, all other I/O ports are in a high-impedance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

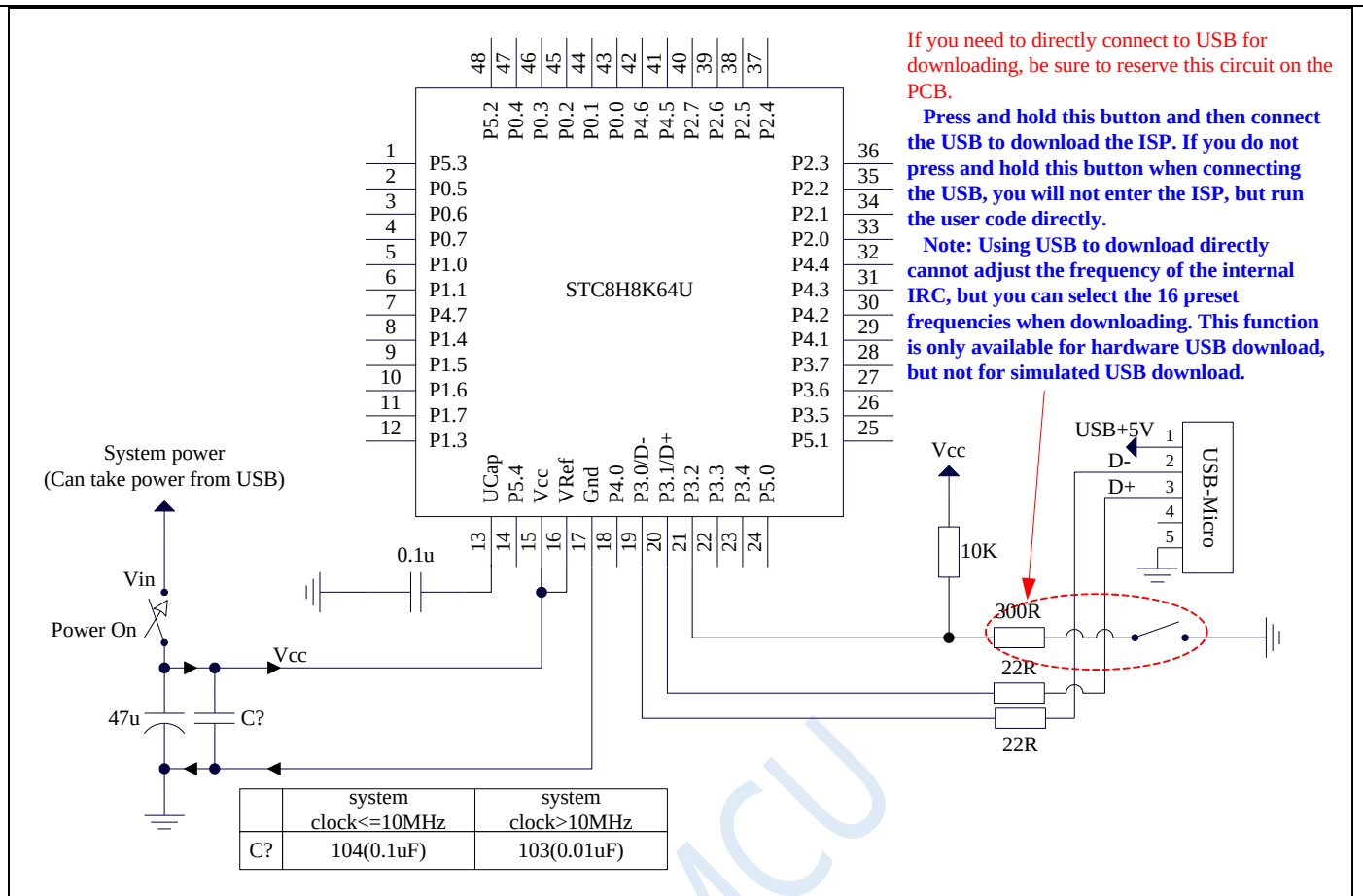
➤ **Package**

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

STC MCU

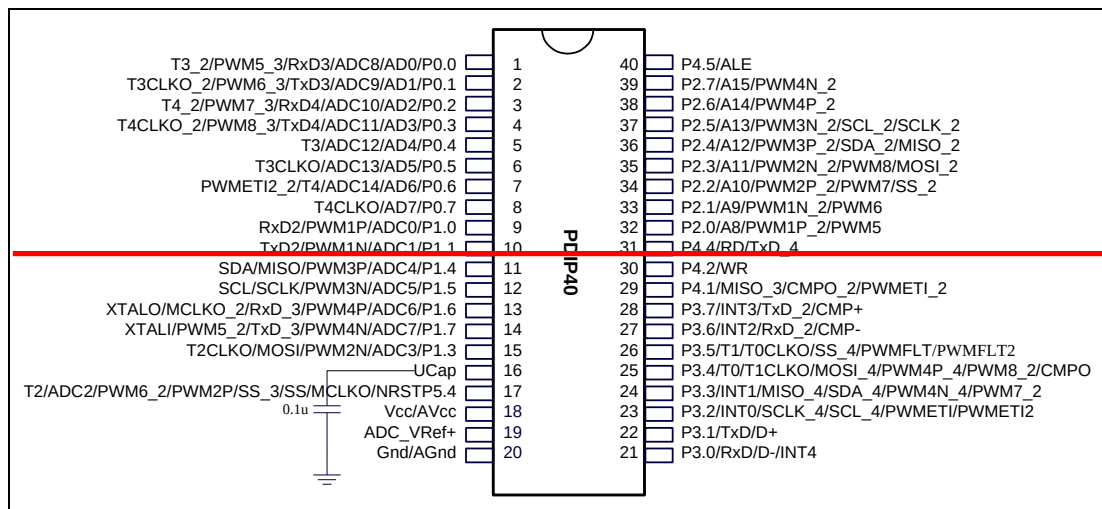
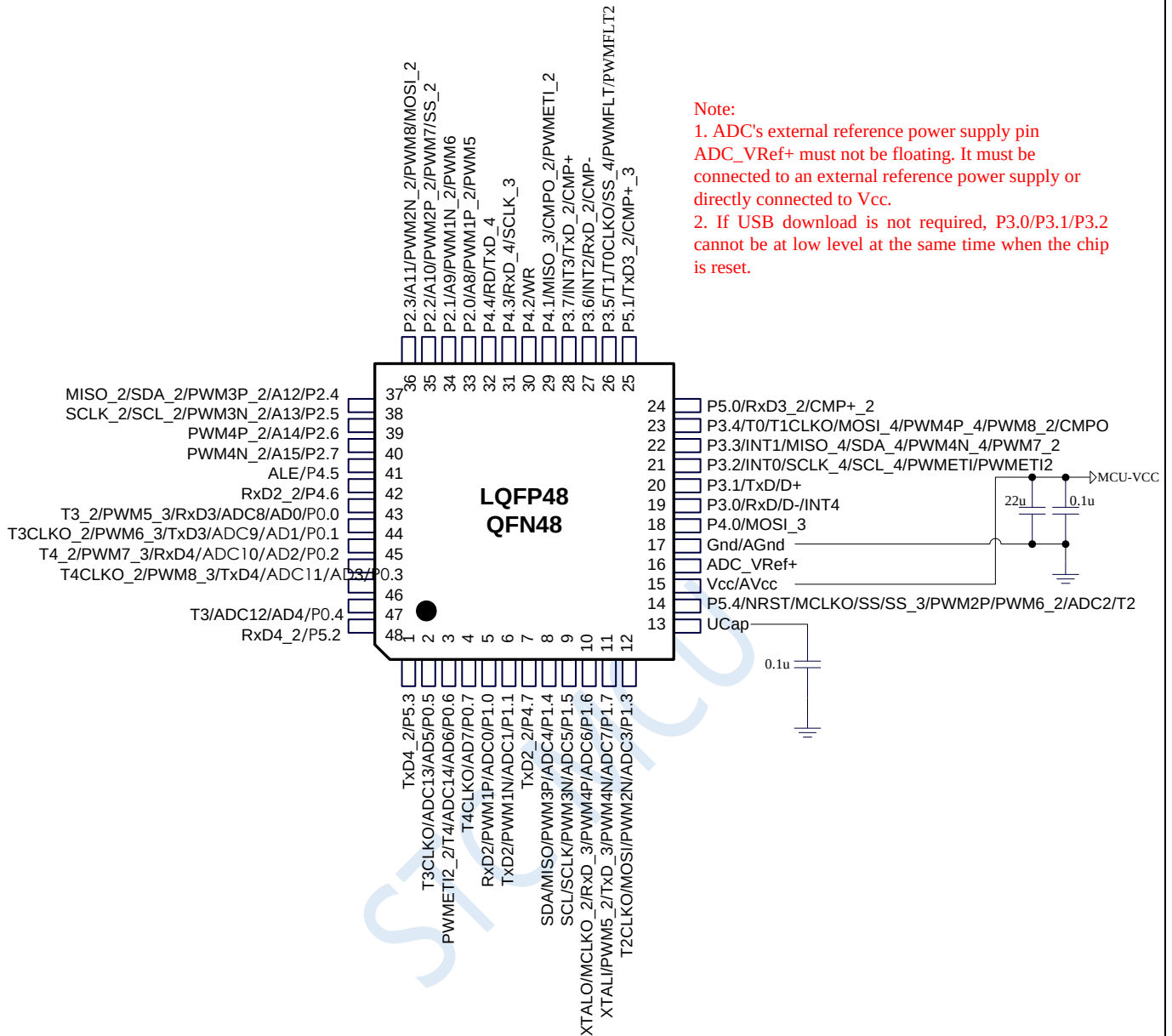
2.5.2 Pinouts

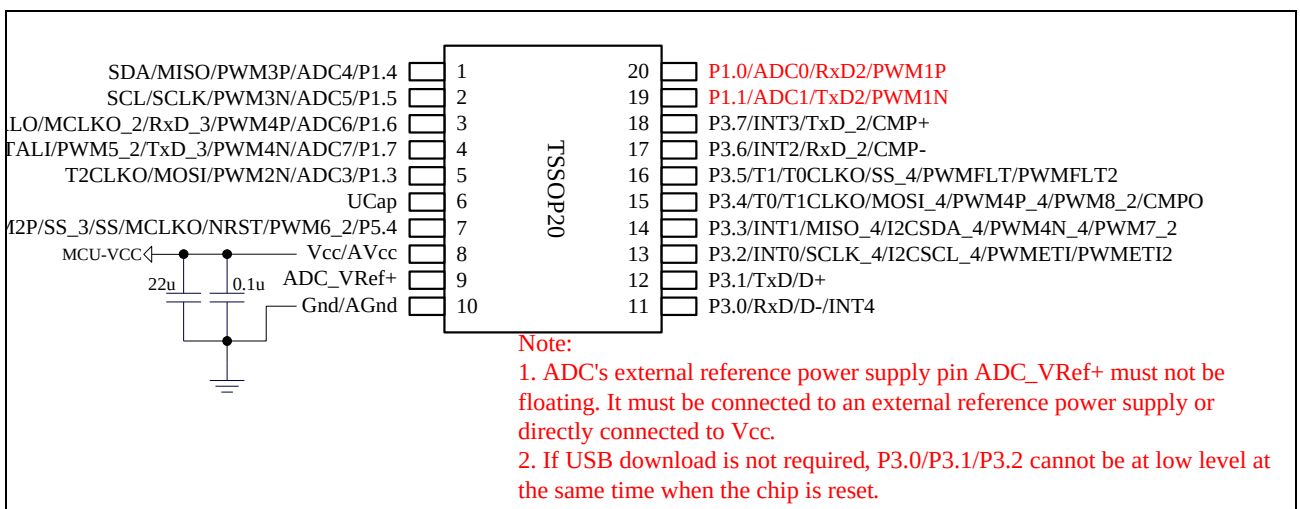
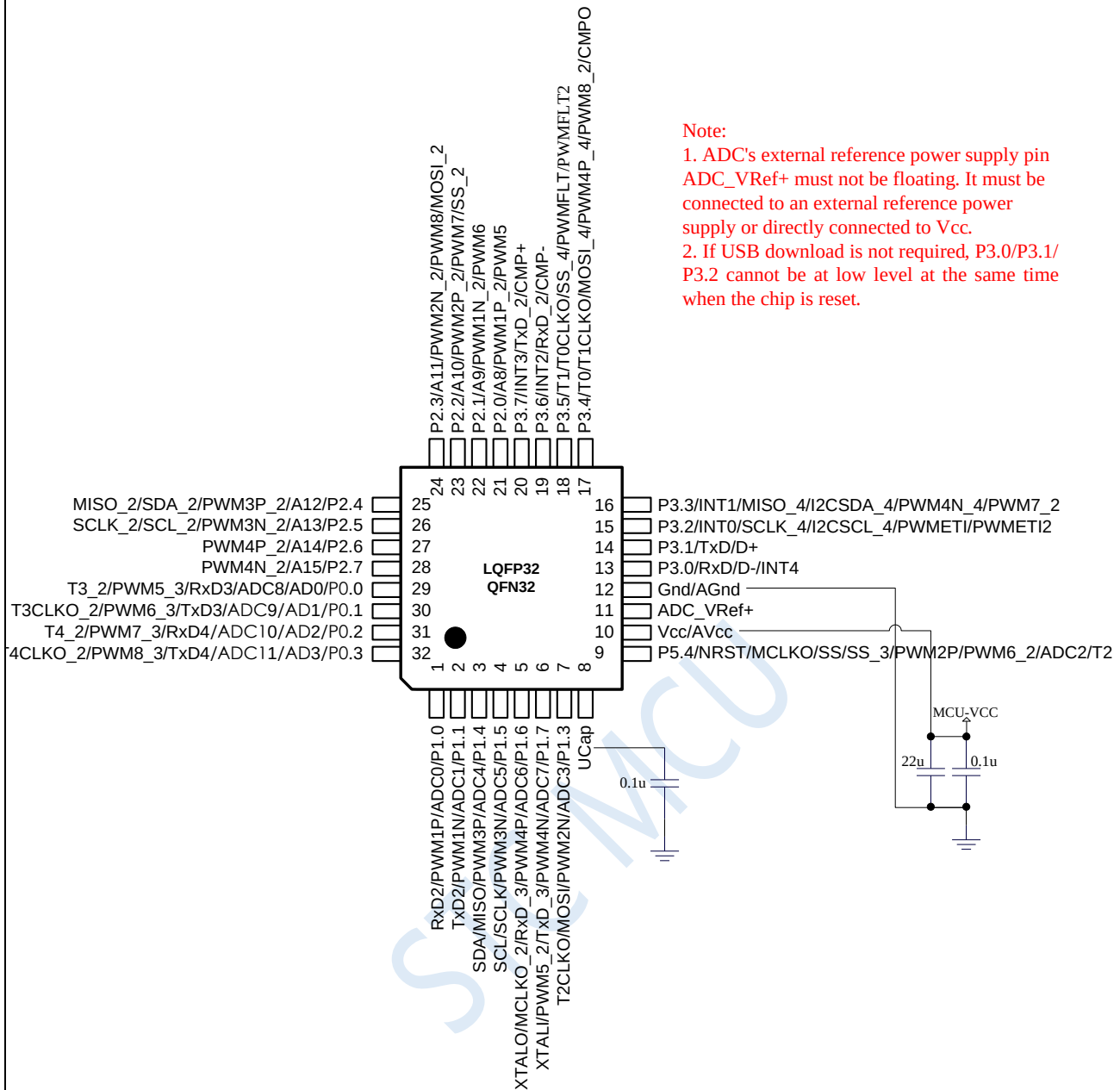




The download steps using ISP and notes are the same as the circumstances in 2.1.2.

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





2.5.3 Pin descriptions

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

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

Pin number	name	type	description

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

Pin number		name	type	description
LQFP64/QFN64	LQFP48/QFN48			

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

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

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

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

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

STC MCU

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

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

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

✓ 1.9V~5.5V

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

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

- 67 -

- ✓ 128 bytes internal direct access RAM (DATA, use keyword *data* to declare in C language program)
- ✓ 128 bytes internal indirect access RAM (IDATA, use keyword *idata* to declare in C language program)
- ✓ 4096 bytes internal extended RAM (internal XDATA, use keyword *xdata* to declare in C language program)
- **Clock**
 - ✓ Internal high precise RC clock IRC(IRC for short, ranges from 4MHz to 45MHz), adjustable while ISP and can be divided to lower frequency by user software, 100KHz for instance.
 - ✓ Error: $\pm 0.3\%$ (at the temperature 25℃)
 - ✓ $-1.35\% \sim +1.30\%$ temperature drift (at the temperature range of -40℃ to +85℃)
 - ✓ $-0.76\% \sim +0.98\%$ temperature drift (at the temperature range of -20℃ to 65℃)
 - ✓ Internal 32KHz low speed IRC with large error
 - ✓ External crystal (4MHz~33MHz) and external clock
Users can freely choose the above 3 clock sources
- **Reset**
 - ✓ Hardware reset
 - ✓ Power-on reset. Measured voltage is 1.69V~1.82V. (Effective when the chip does not enable the low voltage reset function)
The power-on reset voltage is a voltage range consisting of an upper limit voltage and a lower limit voltage. When the operating voltage drops from 5V / 3.3V to the lower limit threshold voltage of the power-on reset, the chip is in a reset state; when the voltage rises from 0V to the upper threshold voltage of power-on reset, the chip is released from the reset state.
 - ✓ Reset by reset pin. The default function of P5.4 is the I/O port. The P5.4 pin can be set as the reset pin while ISP download. (Note: When the P5.4 pin is set as the reset pin, the reset level is low.)
 - ✓ Watch dog timer reset
 - ✓ Low voltage detection reset. 4 low voltage detection levels are provided, 1.9V, 2.3V, 2.8V, 3.0V. Each level of low-voltage detection voltage is a voltage range consisting of an upper limit voltage and a lower limit voltage. When the operating voltage drops from 5V / 3.3V to the lower limit threshold voltage of low-voltage detection, the low-voltage detection takes effect. When the voltage rises from 0V to the upper threshold voltage, the low voltage detection becomes effective.
 - ✓ Software reset
 - ✓ Writing the reset trigger register using software
- **Interrupts**
 - ✓ 41 interrupt sources: INT0(Supports rising edge and falling edge interrupt), INT1(Supports rising edge and falling edge interrupt), INT2(Supports falling edge interrupt only), INT3(Supports falling edge interrupt only), INT4(Supports falling edge interrupt only), timer 0, timer 1, timer 2, timer 3, timer 4, UART 1, UART 2, UART 3, UART 4, ADC, LVD, SPI, I²C, comparator, PWMA, PWMB, RTC, TKS, P1, P2, P3, P4, P5, LCM driver, DMA receive and transmit interrupts of UART 1, DMA receive and transmit interrupts of UART 2, DMA receive and transmit interrupts of UART 3, DMA receive and transmit interrupts of UART 4, DMA interrupt of SPI, DMA interrupt of ADC, DMA interrupt of LCM driver and DMA interrupt of memory-to-memory.
 - ✓ 4 interrupt priority levels
 - ✓ Interrupts that can wake up the CPU in clock stop mode: INT0(P3.2), INT1(P3.3), INT2(P3.6), INT3(P3.7), INT4(P3.0), T0(P3.4), T1(P3.5), T2(P1.2), T3(P0.4), T4(P0.6), RXD(P3.0/P3.6/P1.6/P4.3), RXD2(P1.0/P4.6), RXD3(P0.0/P5.0), RXD4(P0.2/P5.2), I2C_SDA(P1.4/P2.4/P3.3), SPI_SS(P5.4/P2.2/P3.5), Comparator interrupt, LVD interrupt, Power-down wake-up timer and interrupts of all I/O ports.
- **Digital peripherals**
 - ✓ 5 16-bit timers: timer0, timer1, timer2, timer3, timer4, where the mode 3 of timer 0 has the Non-Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
 - ✓ 4 high speed UARTs: UART1, UART2, UART3, UART4, whose maximum baudrate clock may be FOSC/4
 - ✓ 8 channels/2 groups of enhanced PWM, which can realize control signals with dead time, and support external fault detection function. In addition, supports 16-bit timers, 8 external interrupts, 8 external captures and pulse width measurement functions.
 - ✓ SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
 - ✓ I²C: Master mode or slave mode are supported.
 - ✓ MDU16: Hardware 16-bit Multiplier and Divider which supports 32-bit divided by 16-bit, 16-bit divided by 16-bit, 16-bit multiplied by 16-bit, data shift, and data normalization operations.
 - ✓ RTC: Support year, month, day, hour, minute, second, sub-second (1/128 second). And supports clock interrupt and a set of alarm clocks (Note: A version of the chip does not have this function)
 - ✓ I/O port interrupt: All I/Os support interrupts, each group of I/O interrupts has an independent interrupt entry address, all I/O interrupts can support 4 types interrupt mode: high level interrupt, low level interrupt, rising edge interrupt, falling edge interrupt. Provides 4 levels of interrupt priority and supports power-down wake-up function.
 - ✓ DMA: support Memory-To-Memory, SPI, UART1TX/UART1RX, UART2TX/UART2RX, UART3TX/UART3RX, UART4TX/UART4RX, ADC(Automatically calculates the average of multiple ADC results), LCM
 - ✓ LCM (TFT color screen) driver: support 8080 and 6800 interface, and support 8-bit and 16-bit data width (Note: A version of the chip does not have this function)
 - ✓ 8 bits 8080 data bus: 8 bits data lines (TD0~TD7), READ signal (TRD)c WRITE signal (TWR), RS line (TRS)
 - ✓ 16 bits 8080 bus: 16 bits data lines (TD0~TD15), READ signal (TRD)c WRITE signal (TWR), RS line (TRS)
 - ✓ 8 bits 6800 bus: 8 bits data lines (TD0~TD7), enable signal (TE), READ and WRITE signal (TRW), RS line (TRS)
 - ✓ 16 bits 6800 bus: 16 bits data lines (TD0~TD15), enable signal (TE), READ and WRITE signal (TRW), RS line (TRS)
 - ✓ Note: If you use 8-bit data lines to control the TFT screen, you generally need TD0~D7, TRD/TWR/TRS, 11 data and

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

➤ **Analog peripherals**

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

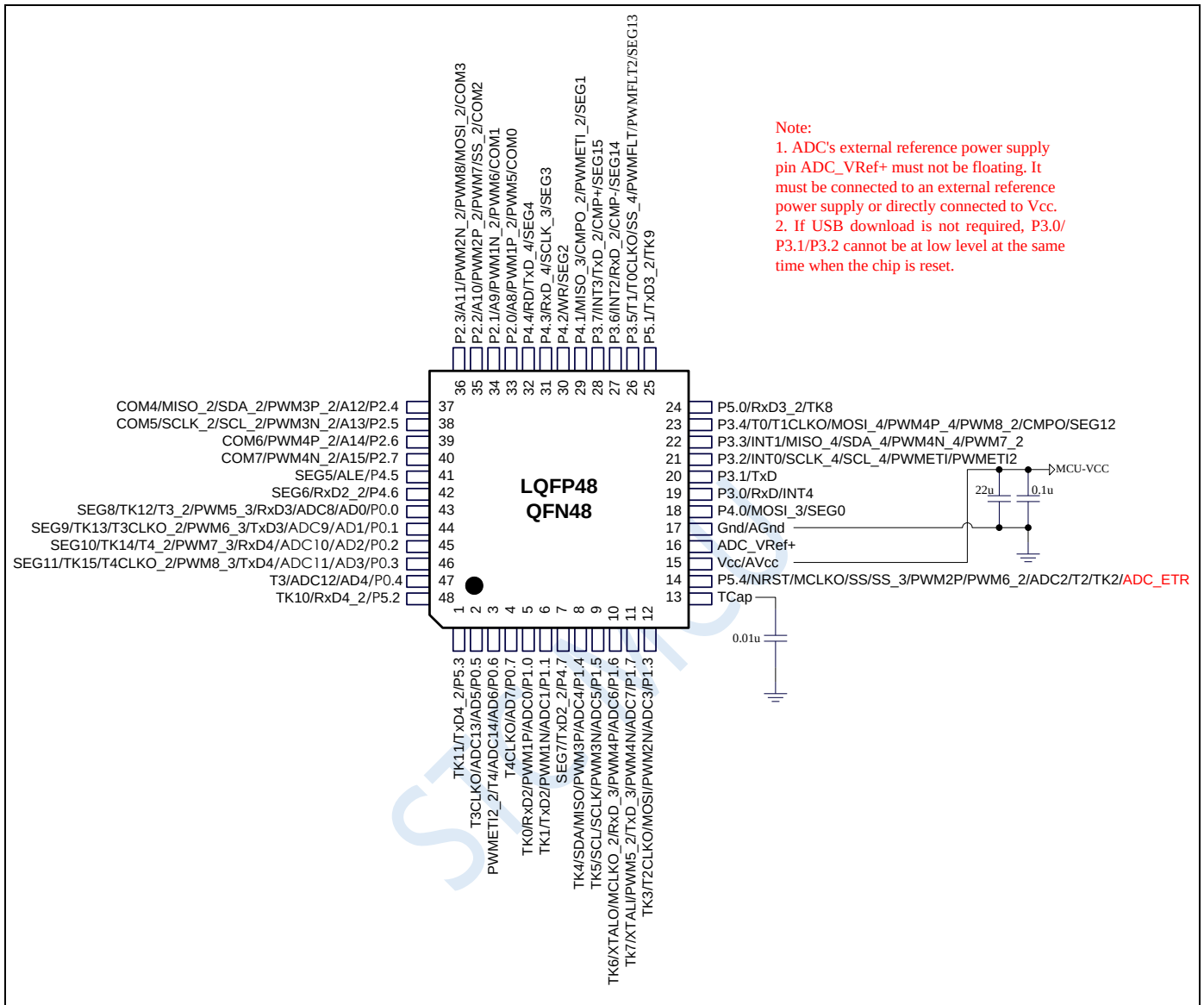
➤ **GPIO**

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

➤ **Package**

- ✓ LQFP48 <9mm*9mm>, QFN48 <6mm*6mm>

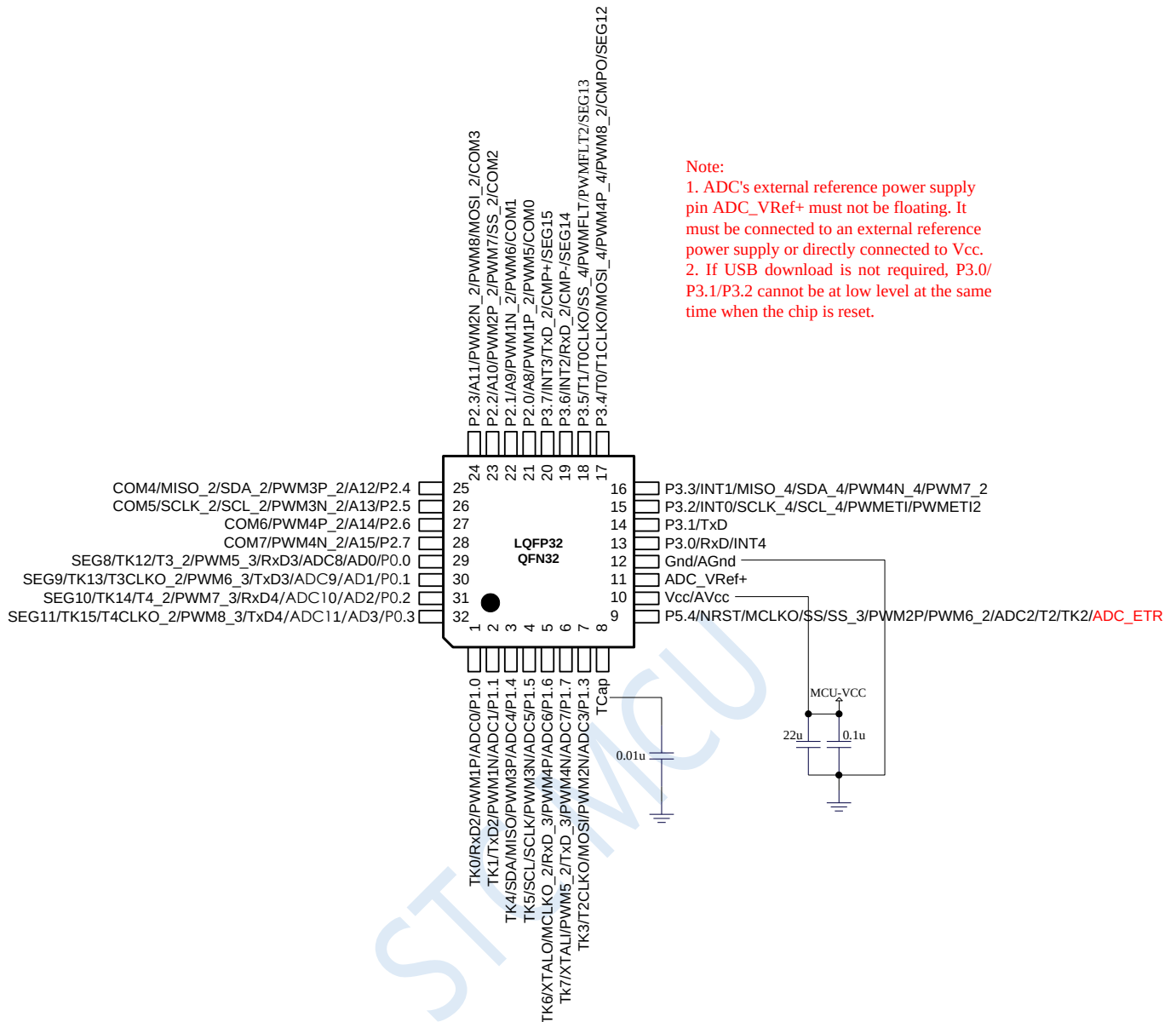
2.6.2 Pinouts



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

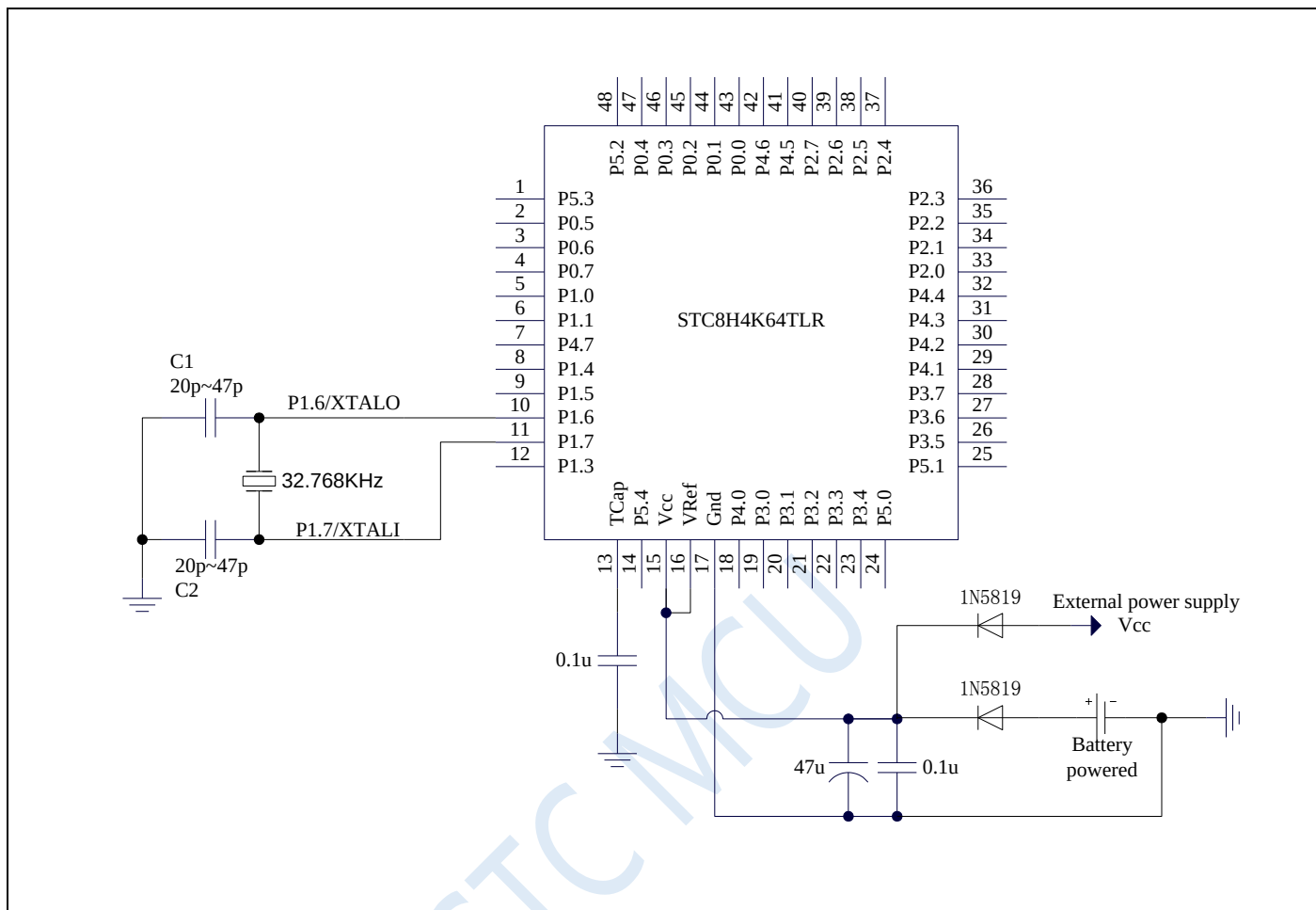
Note:

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



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

2.6.3 RTC reference circuit diagram (No VBAT pin)



2.6.4 Pin descriptions

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

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

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

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

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

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

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

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

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

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

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

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

✓ 1.9V~5.5V

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

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

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

- ✓ Internal high precise RC clock IRC(IRC for short, ranges from 4MHz to 45MHz), adjustable while ISP and can be divided to lower frequency by user software, 100KHz for instance.
 - ✓ Error: $\pm 0.3\%$ (at the temperature 25℃)
 - ✓ -1.35%~+1.30% temperature drift (at the temperature range of -40℃ to +85℃)
 - ✓ -0.76%~+0.98% temperature drift (at the temperature range of -20℃ to 65℃)
- ✓ Internal 32KHz low speed IRC with large error
- ✓ External crystal (4MHz~33MHz) and external clock
Users can freely choose the above 3 clock sources

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

function)

The power-on reset voltage is a voltage range consisting of an upper limit voltage and a lower limit voltage. When the operating voltage drops from 5V / 3.3V to the lower limit threshold voltage of the power-on reset, the chip is in a reset state; when the voltage rises from 0V to the upper threshold voltage of power-on reset, the chip is released from the reset state.

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

➤ Interrupts

- ✓ 43 interrupt sources: INT0(Supports rising edge and falling edge interrupt), INT1(Supports rising edge and falling edge interrupt), INT2(Supports falling edge interrupt only), INT3(Supports falling edge interrupt only), INT4(Supports falling edge interrupt only), timer 0, timer 1, timer 2, timer 3, timer 4, UART 1, UART 2, UART 3, UART 4, ADC, LVD, SPI, I²C, comparator, PWMA, PWMB, RTC, TKS, P1, P2, P3, P4, P5, P6, P7, LCM driver, DMA receive and transmit interrupts of UART 1, DMA receive and transmit interrupts of UART 2, DMA receive and transmit interrupts of UART 3, DMA receive and transmit interrupts of UART 4, DMA interrupt of SPI, DMA interrupt of ADC, DMA interrupt of LCM driver and DMA interrupt of memory-to-memory.
- ✓ 4 interrupt priority levels
- ✓ Interrupts that can wake up the CPU in clock stop mode: INT0(P3.2), INT1(P3.3), INT2(P3.6), INT3(P3.7), INT4(P3.0), T0(P3.4), T1(P3.5), T2(P1.2), T3(P0.4), T4(P0.6), RXD(P3.0/P3.6/P1.6/P4.3), RXD2(P1.0/P4.6), RXD3(P0.0/P5.0), RXD4(P0.2/P5.2), I2C_SDA(P1.4/P2.4/P3.3), SPI_SS(P5.4/P2.2/P3.5), Comparator interrupt, LVD interrupt, Power-down wake-up timer and interrupts of all I/O ports.

➤ Digital peripherals

- ✓ 5 16-bit timers: timer0, timer1, timer2, timer3, timer4, where the mode 3 of timer 0 has the Non-Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
- ✓ 4 high speed UARTs: UART1, UART2, UART3, UART4, whose maximum baudrate clock may be FOSC/4
- ✓ 8 channels/2 groups of enhanced PWM, which can realize control signals with dead time, and support external fault detection function. In addition, supports 16-bit timers, 8 external interrupts, 8 external captures and pulse width measurement functions.
- ✓ SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
- ✓ I²C: Master mode or slave mode are supported.
- ✓ MDU16: Hardware 16-bit Multiplier and Divider which supports 32-bit divided by 16-bit, 16-bit divided by 16-bit, 16-bit multiplied by 16-bit, data shift, and data normalization operations.
- ✓ RTC: Support year, month, day, hour, minute, second, sub-second (1/128 second). And supports clock interrupt and a set of alarm clocks (Note: A version of the chip does not have this function)
- ✓ I/O port interrupt: All I/Os support interrupts, each group of I/O interrupts has an independent interrupt entry address, all I/O interrupts can support 4 types interrupt mode: high level interrupt, low level interrupt, rising edge interrupt, falling edge interrupt. Provides 4 levels of interrupt priority and supports power-down wake-up function.
- ✓ DMA: support Memory-To-Memory, SPI, UART1TX/UART1RX, UART2TX/UART2RX, UART3TX/UART3RX, UART4TX/UART4RX, ADC(Automatically calculates the average of multiple ADC results), LCM
- ✓ LCM (TFT color screen) driver: support 8080 and 6800 interface, and support 8-bit and 16-bit data width (Note: A version of the chip does not have this function)
 - ✓ 8 bits 8080 data bus: 8 bits data lines (TD0~TD7), READ signal (TRD)c WRITE signal (TWR), RS line (TRS)
 - ✓ 16 bits 8080 bus: 16 bits data lines (TD0~TD15), READ signal (TRD)c WRITE signal (TWR), RS line (TRS)
 - ✓ 8 bits 6800 bus: 8 bits data lines (TD0~TD7), enable signal (TE) , READ and WRITE signal (TRW) , RS line (TRS)
 - ✓ 16 bits 6800 bus: 16 bits data lines (TD0~TD15), enable signal (TE) , READ and WRITE signal (TRW) , RS line (TRS)
 - ✓ Note: If you use 8-bit data lines to control the TFT screen, you generally need TD0~D7, TRD/TWR/TRS, 11 data and control lines, plus 2 common I/Os to control chip selection and reset (many TFT color screen chip selections and reset manufacturer has carried out automatic processing, does not need software control)
- ✓ LCD driver: support up to 4COM*40 SEGs and 8 levels grayscale adjustment

➤ Analog peripherals

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

➤ GPIO

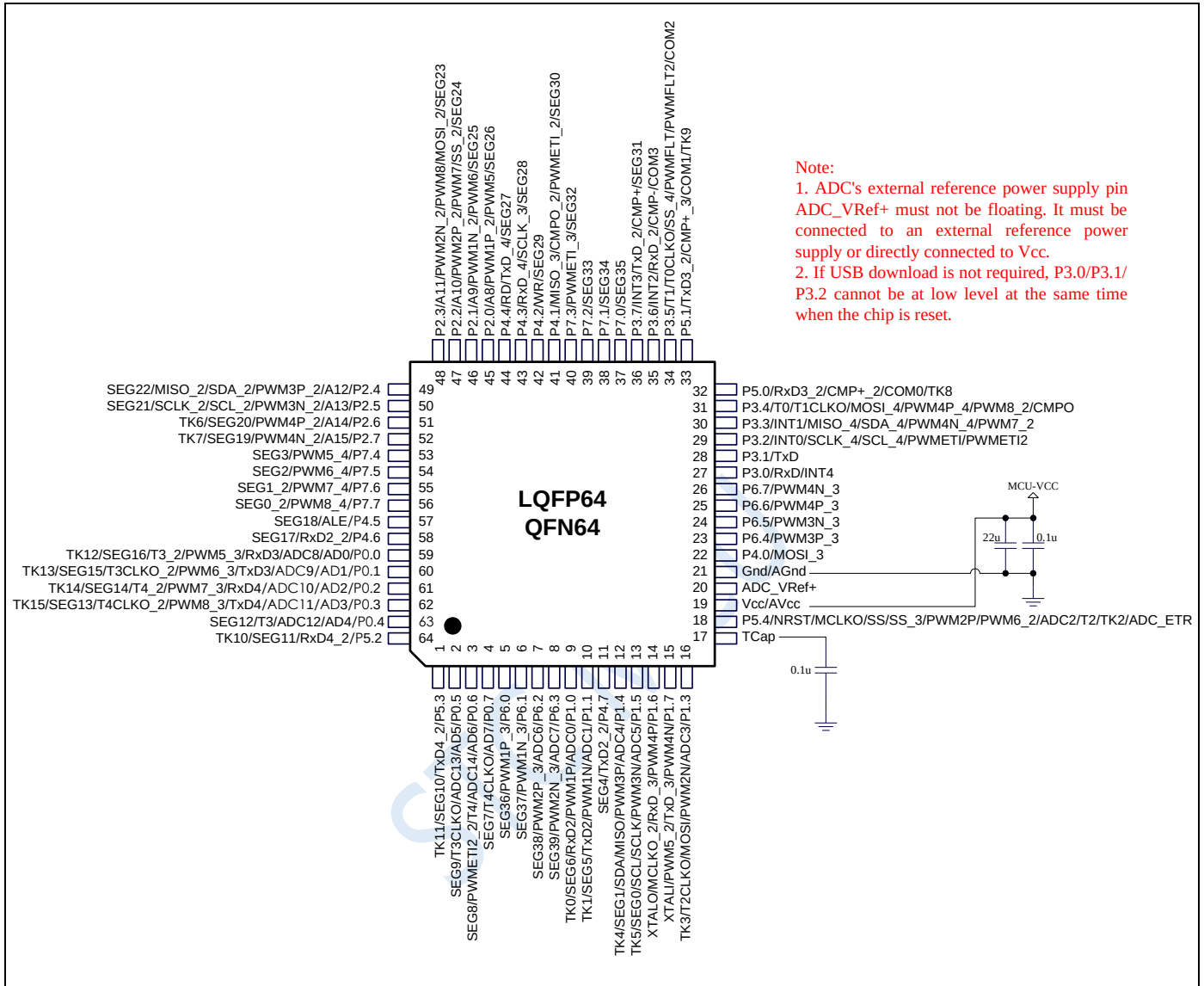
- ✓ Up to 60 GPIOs: P0.0~P0.7, P1.0~P1.7(No P1.2), P2.0~P2.7, P3.0~P3.7, P4.0~P4.7, P5.0~P5.4, P6.0~P6.7, P7.0~P7.7
- ✓ 4 modes for all GPIOs: quasi_bidirectional mode, push-pull outputmode, open drain mode, high-impedance input mode
- ✓ Except for P3.0 and P3.1, all other I/O ports are in a high-impedance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

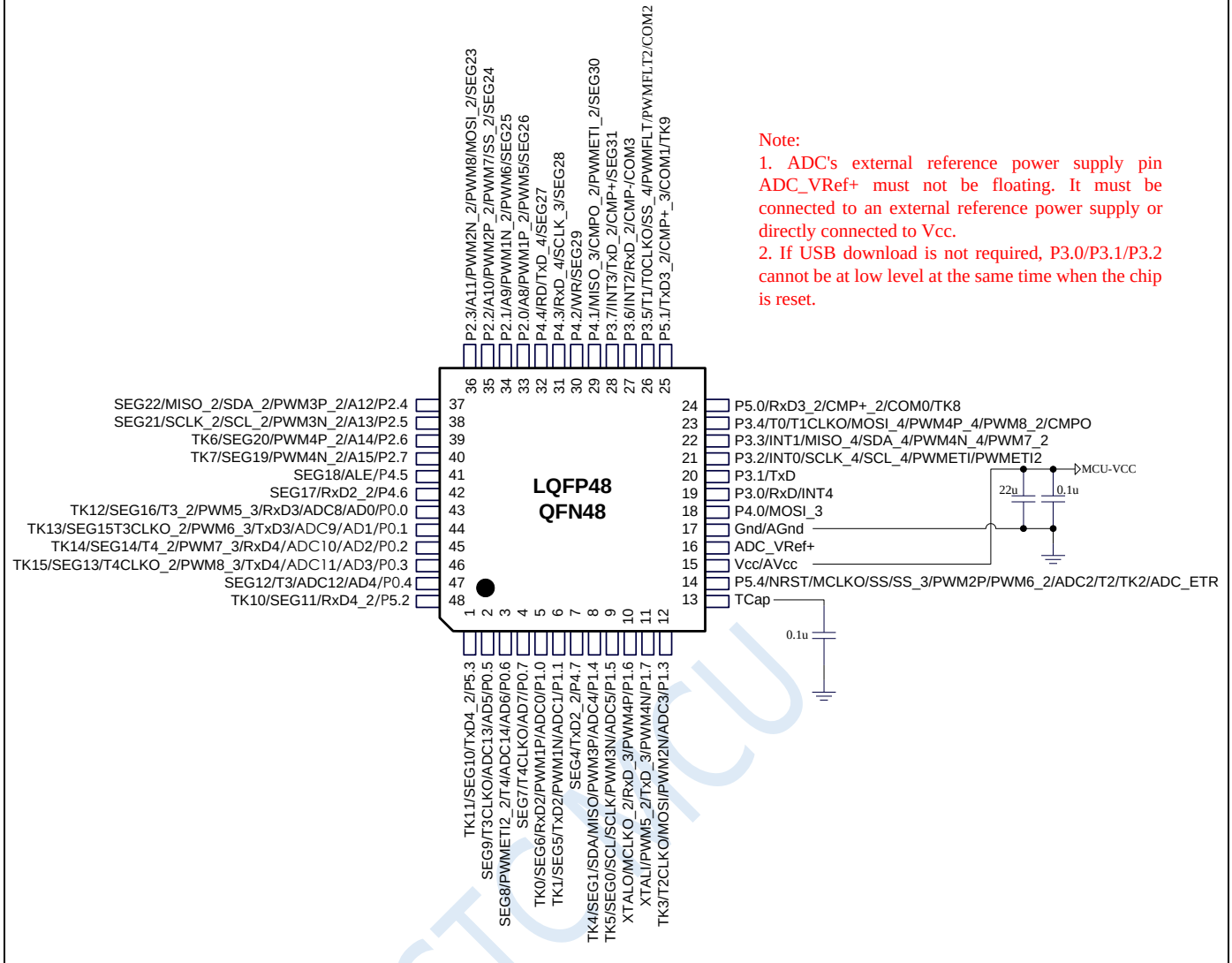
➤ **Package**

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

STC MCU

2.7.2 Pinouts



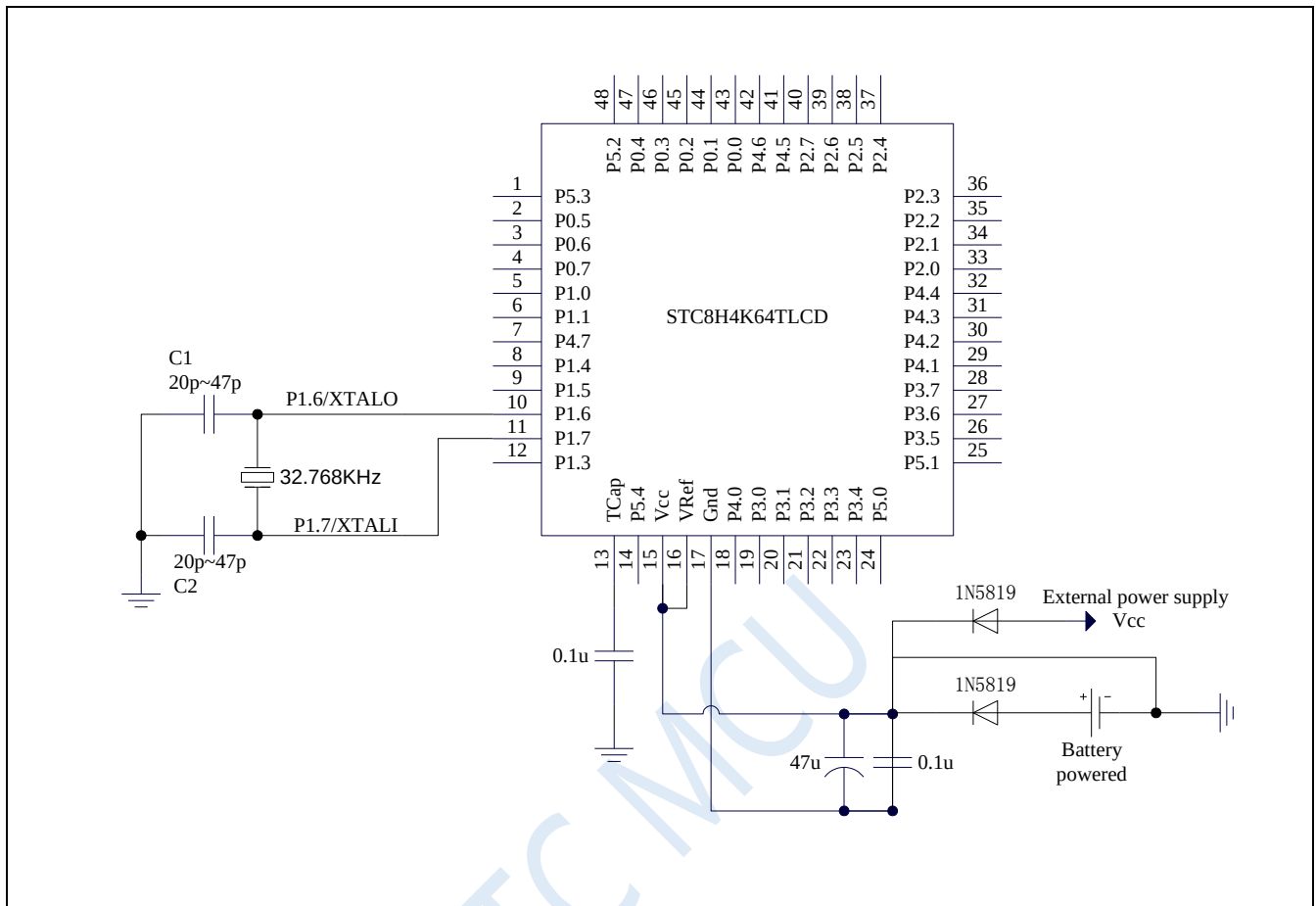


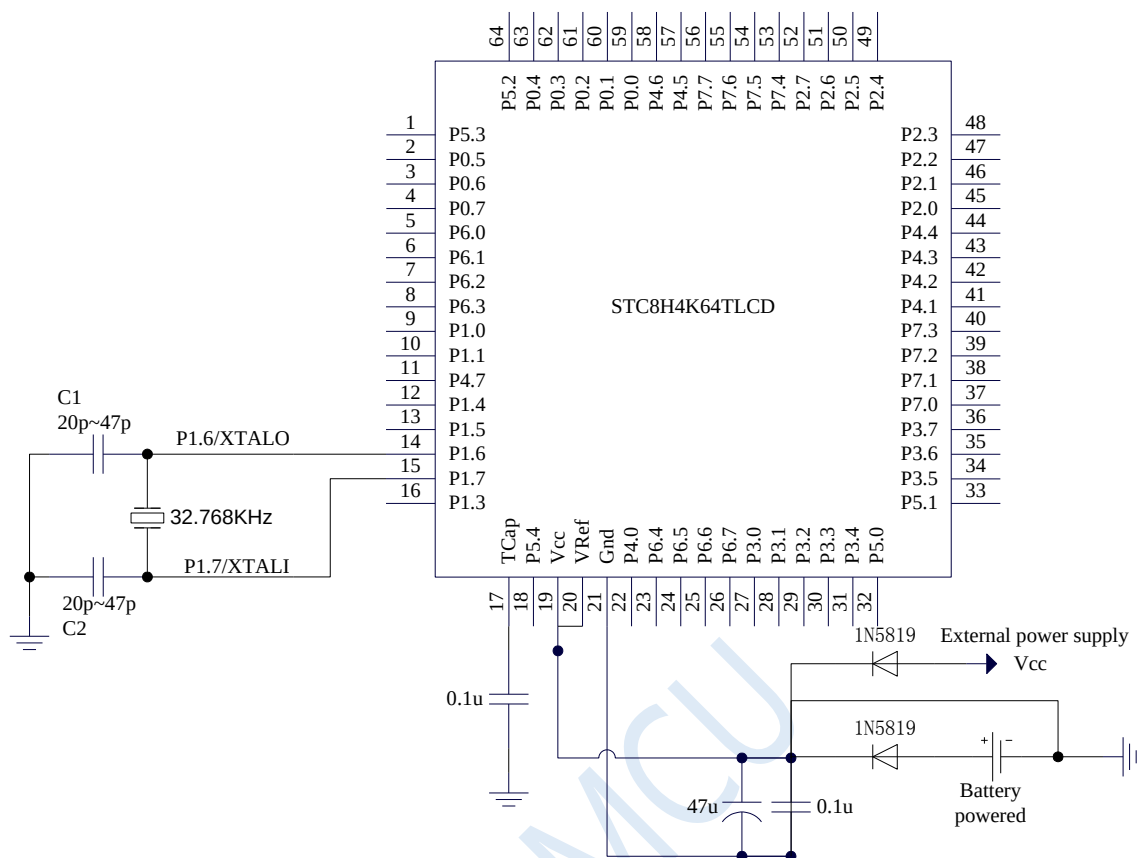
The download steps using ISP and notes are the same as the circumstances in 2.1.2.

Note:

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

2.7.3 RTC reference circuit diagram (No VBAT pin)





2.7.4 Pin descriptions

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

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

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

23		P6.4	I/O	Standard IO port
		PWM3P_3	I/O	Capture of external signal/Positive of PWM3 pulse output
24		P6.5	I/O	Standard IO port
		PWM3N_3	I/O	Capture of external signal/Negative of PWM3 pulse output
25		P6.6	I/O	Standard IO port
		PWM4P_3	O	Capture of external signal/Positive of PWM4 pulse output
26		P6.7	I/O	Standard IO port
		PWM4N_3	I/O	Capture of external signal/Negative of PWM4 pulse output
27	19	P3.0	I/O	Standard IO port
		RxD	I	Input of UART1
		INT4	I	External interrupt 4
28	20	P3.1	I/O	Standard IO port
		TxD	O	Transmit pin of UART 1
29	21	P3.2	I/O	Standard IO port
		INT0	I	External interrupt0
		SCLK_4	I/O	Serial Clock of SPI
		SCL_4	I/O	Serial Clock line of I2C
		PWMET1	I	PWM external trigger input pin
		PWMET2	I	PWM external trigger input pin 2
30	22	P3.3	I/O	Standard IO port
		INT1	I	External interrupt1
		MISO_4	I/O	Master Input/Slave Output of SPI
		SDA_4	I/O	Serial data line of I2C
		PWM4N_4	I/O	Capture of external signal/Negative of PWM4 pulse output
		PWM7_2	I/O	Capture of external signal/Pulse output of PWM7
31	23	P3.4	I/O	Standard IO port
		T0	I	Timer0 external input
		T1CLKO	O	Clock out of timer 1
		MOSI_4	I/O	Master Output/Slave Input of SPI
		PWM4P_4	I/O	Capture of external signal/Positive of PWM4 pulse output
		PWM8_2	I/O	Capture of external signal/Pulse output of PWM8
		CMPO	O	Output of comparator
32	24	P5.0	I/O	Standard IO port
		RxD3_2	I	Input of UART 3
		CMP+_2	I	Positive input of comparator
		COM0	O	LCD driver COM line
		TK8	I	Touch key
33	25	P5.1	I/O	Standard IO port
		TxD3_2	O	Transmit pin of UART 3
		CMP+_3	I	Positive input of comparator
		COM1	O	LCD driver COM line
		TK9	I	Touch key

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

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

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

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

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

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

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

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

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

MCU	products supply information
	Price & Package
	Online debug itself
	Support software USB download directly
	Support RS485 download
	Password can be set for next update
	Program encrypted transmission (Anti-block)
	Clock output and Reset
	Internal high precision Clock (adjustable under 36MHz)
	Internal high reliable reset circuit with 4 levels optional reset threshold voltage
	Watch-dog Timer
	Internal LVD interrupt (can wake-up CPU) Comparator (May be used as ADC to detect external power-down)
	DMA 15 channels high speed ADC (8 PWMs can be used as 8 DACs)
	Power-down Wake-up timer
	16-bit advanced PWM timer with General-purpose programmable dead time
	Timers/Counters (10-14 Pin can wake-up CPU)
	MDU16 (Hardware 16-bit Multiplier and Divider)
	I ² C which can wake-up CPU
	DMA SPI which can wake-up CPU
	RTC
	Touch key
	DMA UARTs which can wake-up CPU
	All I/O ports support interrupts and can wake
	Traditional I/O
	Maximum I/O Lines
	EEPROM 100 thousand times) (Byte)
	Enhanced Dual DPTR increasing or decreasing
	xdata Internal extended SRAM (Byte)
	idata Internal DATA RAM(Byte)
	Flash Code Memory (100 thousand times) (Byte)
	Operating voltage (V)

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

✓ 1.9V~5.5V

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

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

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

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

Users can freely choose the above 3 clock sources

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

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

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

interrupt), INT2(Supports falling edge interrupt only), INT3(Supports falling edge interrupt only), INT4(Supports falling edge interrupt only), timer 0, timer 1, timer 2, UART 1, UART 2, ADC, LVD, SPI, I²C, comparator, PWMA, PWMB, RTC, TKS, P1, P3, P5, DMA receive and transmit interrupts of UART 1, DMA receive and transmit interrupts of UART 2, DMA interrupt of SPI, DMA interrupt of ADC and DMA interrupt of memory-to-memory.

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

➤ Digital peripherals

- ✓ 3 16-bit timers: timer0, timer1, timer2, where the mode 3 of timer 0 has the Non-Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
- ✓ 2 high speed UARTs: UART1, UART2, whose maximum baudrate clock may be FOSC/4
- ✓ 8 channels/2 groups of enhanced PWM, which can realize control signals with dead time, and support external fault detection function. In addition, supports 16-bit timers, 8 external interrupts, 8 external captures and pulse width measurement functions.
- ✓ SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
- ✓ I²C: Master mode or slave mode are supported.
- ✓ MDU16: Hardware 16-bit Multiplier and Divider which supports 32-bit divided by 16-bit, 16-bit divided by 16-bit, 16-bit multiplied by 16-bit, data shift, and data normalization operations.
- ✓ RTC: Support year, month, day, hour, minute, second, sub-second (1/128 second). And supports clock interrupt and a set of alarm clocks (Note: A version of the chip does not have this function)
- ✓ I/O port interrupt: All I/Os support interrupts, each group of I/O interrupts has an independent interrupt entry address, all I/O interrupts can support 4 types interrupt mode: high level interrupt, low level interrupt, rising edge interrupt, falling edge interrupt. Provides 4 levels of interrupt priority and supports power-down wake-up function.
- ✓ DMA: support Memory-To-Memory, SPI, UART1TX/UART1RX, UART2TX/UART2RX, ADC(Automatically calculates the average of multiple ADC results)

➤ Analog peripherals

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

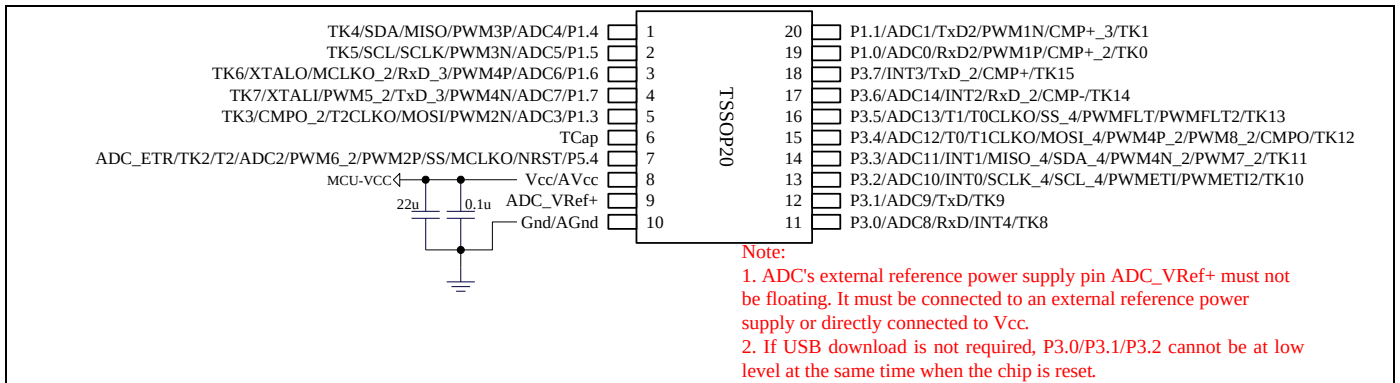
➤ GPIO

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

➤ Package

- ✓ TSSOP20, QFN20

2.8.2 Pinouts



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

2.8.3 Pin descriptions

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

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

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

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

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

2.9.1 Features and Price

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

products supply information		core avallia
Price & Package	PDP40	~
	LQFP32<9mm*9mm>	~
	LQFP48 <9mm*9mm>	~
	LQFP64 <12mm*12mm>	~
Online debug itself		Y
Support hardware USB download and debug directly		Y
Support RS485 download		Y
Password can be set for next update		Y
Program encrypted transmission (Anti-blocking)		Y
Clock output and Reset		Y
Internal high precision Clock (adjustal under 33MHz)		Y
Internal high reliable reset circuit with 4 levels optional reset threshold voltage		Y
Watch-dog Timer		Y
Internal LVD interrupt (can wake-up CPU)		Y
Comparator (May be used as ADC to detect external power-down)		Y
DMA 15 channels high speed ADC (8 PWMs can be used as 8 DACs)		12bit
Power-down Wake-up timer		Y
16-bit advanced PWM timer with Complementary symmetrical dead-time		8
Timers/Counters (T0-T4 pin Ccn wake-up CPU)		5
MDU32 (Hardware 32-bit Multiplier and Divider)		Y
I ² C which can wake-up CPU		Y
DMA SPI which can wake-up CPU		Y
Full-speed USB		Y
LIN		Y
CAN		2
DMA USARTs which can wake-up CPU		2
DMA UARTs which can wake-up CPU		2
RTC		Y
DMA 8080/6800 interface/ LCM driver(8-bit and 16-bit)		Y
All I/O ports support interrupts and can wake up MCU		Y
Traditional I/O interrupt(INT0/INT1/INT2/INT3/INT4) (can wake-up CPU)		Y
Maximum I/O Lines		60
EEPROM 100 thousand times) (Byte)		64K
Enhanced Dual DPTR increasing or decreasing		2
xdata Internal extended SRAM (Byte)		8K
edata Internal extended DATA RAM which can be used as stack(Byte)		4K
Flash Code Memory (100 thousand times) (Byte)		64K
Operating voltage (V)		1.9-5.5
MCU		STC32G12K64
		STC32G12K128

➤ Core

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

➤ Operating voltage

- ✓ 1.9V~5.5V
- ✓ Built-in LDO

➤ Operating temperature

- ✓ -40℃~85℃

➤ Flash memory

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

➤ SRAM

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

Notes on using xdata:

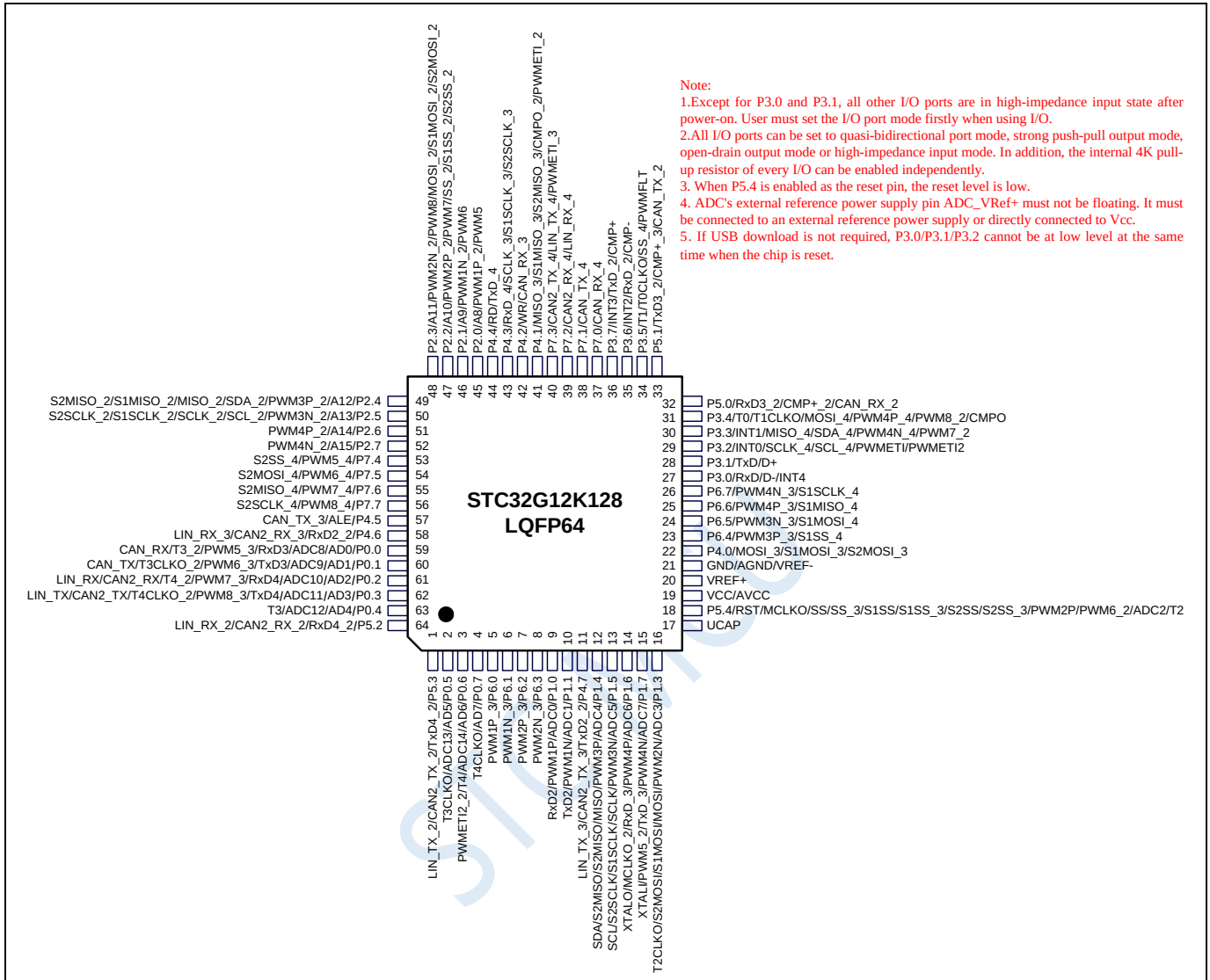
When defining variables, single-byte variables can be defined in xdata, and multi-byte (2-byte, 4-byte) variables need to be defined in edata.

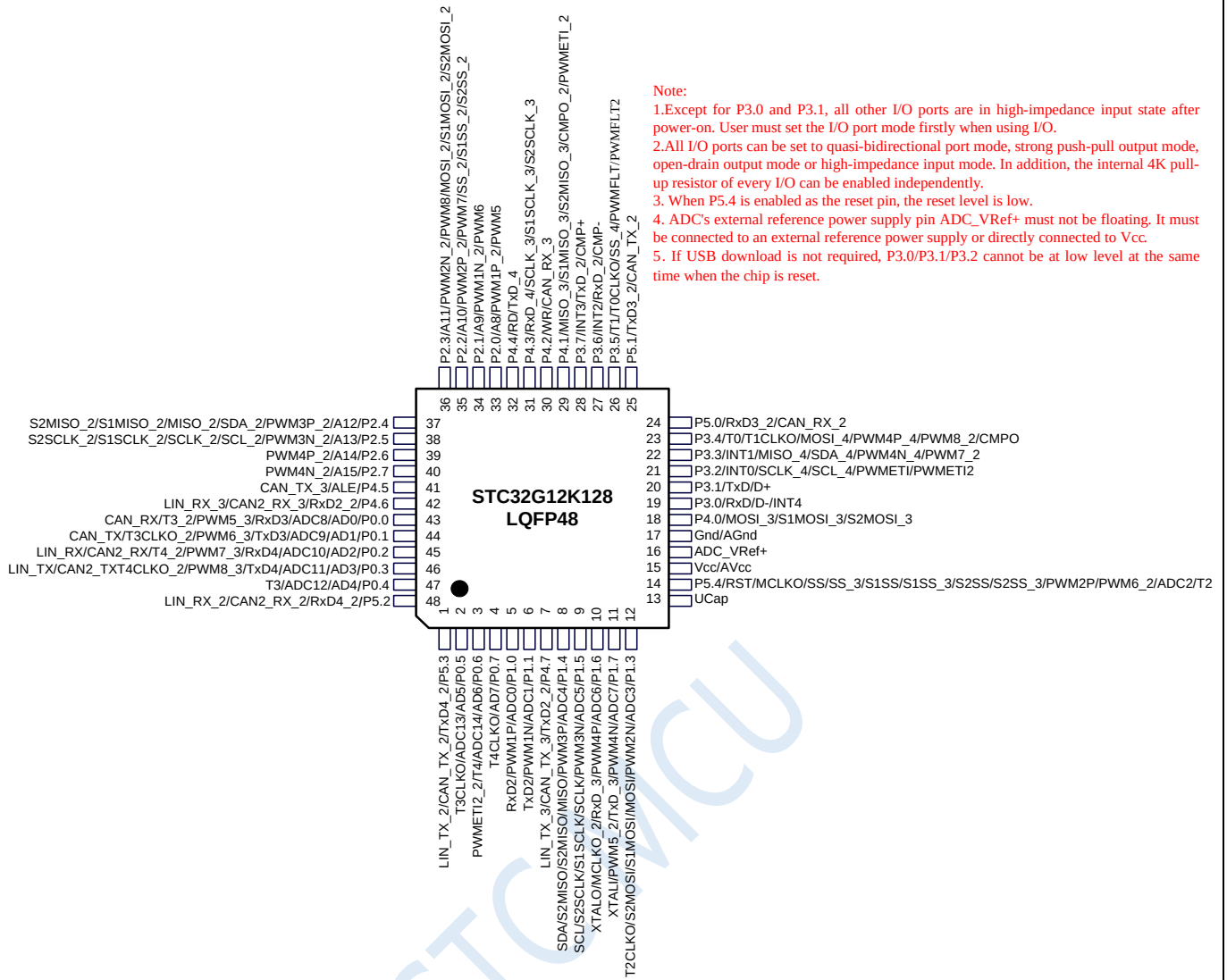
➤ Clock

- ✓ Internal high precise RC clock IRC(IRC for short), adjustable while ISP and can be divided to lower frequency by user software.
 - ✓ Error: $\pm 0.3\%$ (at the temperature 25°C)
 - ✓ $-1.35\% \sim +1.30\%$ temperature drift (at the temperature range of -40 °C to +85 °C)
 - ✓ $-0.76\% \sim +0.98\%$ temperature drift (at the temperature range of -20°C to 65°C)
- ✓ Internal 32KHz low speed IRC with large error
- ✓ External crystal (4MHz~33MHz) and external clock
- ✓ Internal PLL output clock
- Users can freely choose the above 4 clock sources
- **Reset**
 - ✓ Hardware reset
 - ✓ Power-on reset. Measured voltage is 1.7V~1.9V. (Effective when the chip does not enable the low voltage reset function)
 - ✓ Reset by reset pin. The default function of P5.4 is the I/O port. The P5.4 pin can be set as the reset pin while ISP download. (Note: When the P5.4 pin is set as the reset pin, the reset level is low.)
 - ✓ Watch dog timer reset
 - ✓ Low voltage detection reset. 4 low voltage detection levels are provided, 2.0V, 2.3V, 2.7V, 3.0V.
 - ✓ Software reset
 - ✓ Writing the reset trigger register using software
- **Interrupts**
 - ✓ 49 interrupt sources: INT0, INT1, INT2, INT3, INT4, timer 0, timer 1, timer 2, timer 3, timer 4, USART1, USART2, UART 3, UART 4, ADC, LVD, SPI, I²C, comparator, PWMA, PWMB, USB, CAN, CAN2, LIN, LCMIF color screen interface, RTC, all I/O interrupts (8 groups), DMA receive and transmit interrupts of USART 1, DMA receive and transmit interrupts of USART 2, DMA receive and transmit interrupts of UART 3, DMA receive and transmit interrupts of UART 4, DMA interrupt of I2C, DMA interrupt of SPI, DMA interrupt of ADC, DMA interrupt of LCD driver and DMA interrupt of memory-to-memory.
 - ✓ 4 interrupt priority levels
- **Digital peripherals**
 - ✓ 5 16-bit timers: timer0, timer1, timer2, timer3, timer4, where the mode 3 of timer 0 has the Non-Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
 - ✓ 2 high speed USARTs: USART1, USART2, whose maximum baudrate clock may be FOSC/4. The following modes are supported, synchronous serial port mode, asynchronous serial port mode, SPI mode, LIN mode, infrared mode (IrDA), smart card mode (ISO7816)
 - ✓ 2 high speed UARTs: UART3, UART4, whose maximum baudrate clock may be FOSC/4
 - ✓ 2 groups of enhanced PWM, which can realize 8 channels(4 groups complementary symmetry) control signals with dead time, and support external fault detection function.
 - ✓ SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
 - ✓ I²C: Master mode or slave mode are supported.
 - ✓ ICE: Hardware support emulation.
 - ✓ RTC: Support year, month, day, hour, minute, second, sub-second (1/128 second). And supports clock interrupt and a set of alarm clocks.
 - ✓ USB: USB2.0/USB1.1 compatible with full-speed USB, 6 bidirectional endpoints, support 4 endpoint transfer modes (control transfer, interrupt transfer, bulk transfer and isochronous transfer), each endpoint has a 64-byte buffer.
 - ✓ CAN: Two independent CAN 2.0 control units.
 - ✓ LIN: An independent LIN control unit (supports versions 1.3 and 2.1), USART1 and USART2 can support two sets of LIN.
 - ✓ MDU32: Hardware 32-bit Multiplier and Divider which supports 32-bit divided by 32-bit, 32-bit multiplied by 32-bit operations.
 - ✓ I/O port interrupt: All I/Os support interrupts, each group of I/O interrupts has an independent interrupt entry address, all I/O interrupts can support 4 types interrupt mode: high level interrupt, low level interrupt, rising edge interrupt, falling edge interrupt. Provides 4 levels of interrupt priority and supports power-down wake-up function.
 - ✓ LCD driver: support 8080 and 6800 interface, and support 8-bit and 16-bit data width.
 - ✓ DMA: Support SPI shift to receive data to memory, SPI shift to send data from memory, I2C send data from memory, I2C receive data to memory, USART 1/2 and UART 3/4 receive data to memory, USART 1/2 and UART 3/4 send data from memory, ADC automatically sample data to memory (calculate average value at the same time), LCD driver send data from memory, and copy data from memory to memory
 - ✓ Hardware digital ID: support 32 bytes
- **Analog peripherals**
 - ✓ Ultra high speed ADC which supports 12-bit precision 15 channels (channel 0 to channel 14) analog-to-digital conversion. ADC channel 15 is used to test the internal reference voltage. (The default internal reference voltage is 1.19V when the chip is shipped, the error is $\pm 1\%$)
 - ✓ Comparator. A set of comparator
- **GPIO**
 - ✓ Up to 60 GPIOs: P0.0~P0.7, P1.0~P1.7(No P1.2), P2.0~P2.7, P3.0~P3.7, P4.0~P4.7, P5.0~P5.4, P6.0~P6.7, P7.0~P7.7
 - ✓ 4 modes for all GPIOs: quasi_bidirectional mode, push-pull outputmode, open drain mode, high-impedance input mode
 - ✓ Except for P3.0 and P3.1, all other I/O ports are in a high-impedance state after power-on. User must set the I/O ports mode before using them. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.
- **Package**

STC MCU

2.9.2 Pinouts



**Note:**

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

2.9.3 Pin descriptions

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

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

Pin number		name	type	description
LQFP64	LQFP48			

26		P6.7	I/O	Standard IO port
		PWM4N_3	I/O	Capture of external signal/Negative of PWM4 pulse output
		S1SCLK_4	I/O	Serial Clock of USART1-SPI
27	19	P3.0	I/O	Standard IO port
		D-	I/O	USB data line
		RxD	I	Input of USART1
		INT4	I	External interrupt4
28	20	P3.1	I/O	Standard IO port
		D+	I/O	USB data line
		TxD	O	Transmit pin of USART1
29	21	P3.2	I/O	Standard IO port
		INT0	I	External interrupt0
		SCLK_4	I/O	Serial Clock of SPI
		SCL_4	I/O	Serial Clock line of I2C
		PWMET1	I	PWM external trigger input pin
		PWMET2	I	PWM external trigger input pin 2
30	22	P3.3	I/O	Standard IO port
		INT1	I	External interrupt1
		MISO_4	I/O	Master Input/Slave Output of SPI
		SDA_4	I/O	Serial data line of I2C
		PWM4N_4	I/O	Capture of external signal/Negative of PWM4 pulse output
		PWM7_2	I/O	Capture of external signal/Pulse output of PWM7
31	23	P3.4	I/O	Standard IO port
		T0	I	Timer0 external input
		T1CLKO	O	Clock out of timer 1
		MOSI_4	I/O	Master Output/Slave Input of SPI
		PWM4P_4	I/O	Capture of external signal/Positive of PWM4 pulse output
		PWM8_2	I/O	Capture of external signal/Pulse output of PWM8
		CMPO	O	Output of comparator
32	24	P5.0	I/O	Standard IO port
		RxD3_2	I	Input of UART3
		CMP+_2	I	Positive input of comparator
		CAN_RX_2	I	Receive pin of CAN
33	25	P5.1	I/O	Standard IO port
		TxD3_2	O	Transmit pin of UART 3
		CMP+_3	I	Positive input of comparator
		CAN_TX_2	I	Transmit pin of CAN
34	26	P3.5	I/O	Standard IO port
		T1	I	Timer1 external input
		T0CLKO	O	Clock out of timer 0
		SS_4	I	Slave selection of SPI (it is output with regard to master)
		PWMFLT	I	Enhance PWMA external anomaly detection pin
35	27	P3.6	I/O	Standard IO port
		INT2	I	External interrupt2
		RxD_2	I	Input of USART1
		CMP-	I	Negative input of comparator
36	28	P3.7	I/O	Standard IO port
		INT3	I	External interrupt3
		TxD_2	O	Transmit pin of USART 1
		CMP+	I	Positive input of comparator
37		P7.0	I/O	Standard IO port
		CAN_RX_4	I	Receive pin of CAN

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

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

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

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

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

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

STC MCU

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

2.10.1 Features and Price

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

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

➤ Core

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

➤ Operating voltage

- ✓ 1.9V~5.5V
- ✓ Built-in LDO

➤ Operating temperature

- ✓ -40℃~85℃

➤ Flash memory

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

➤ SRAM

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

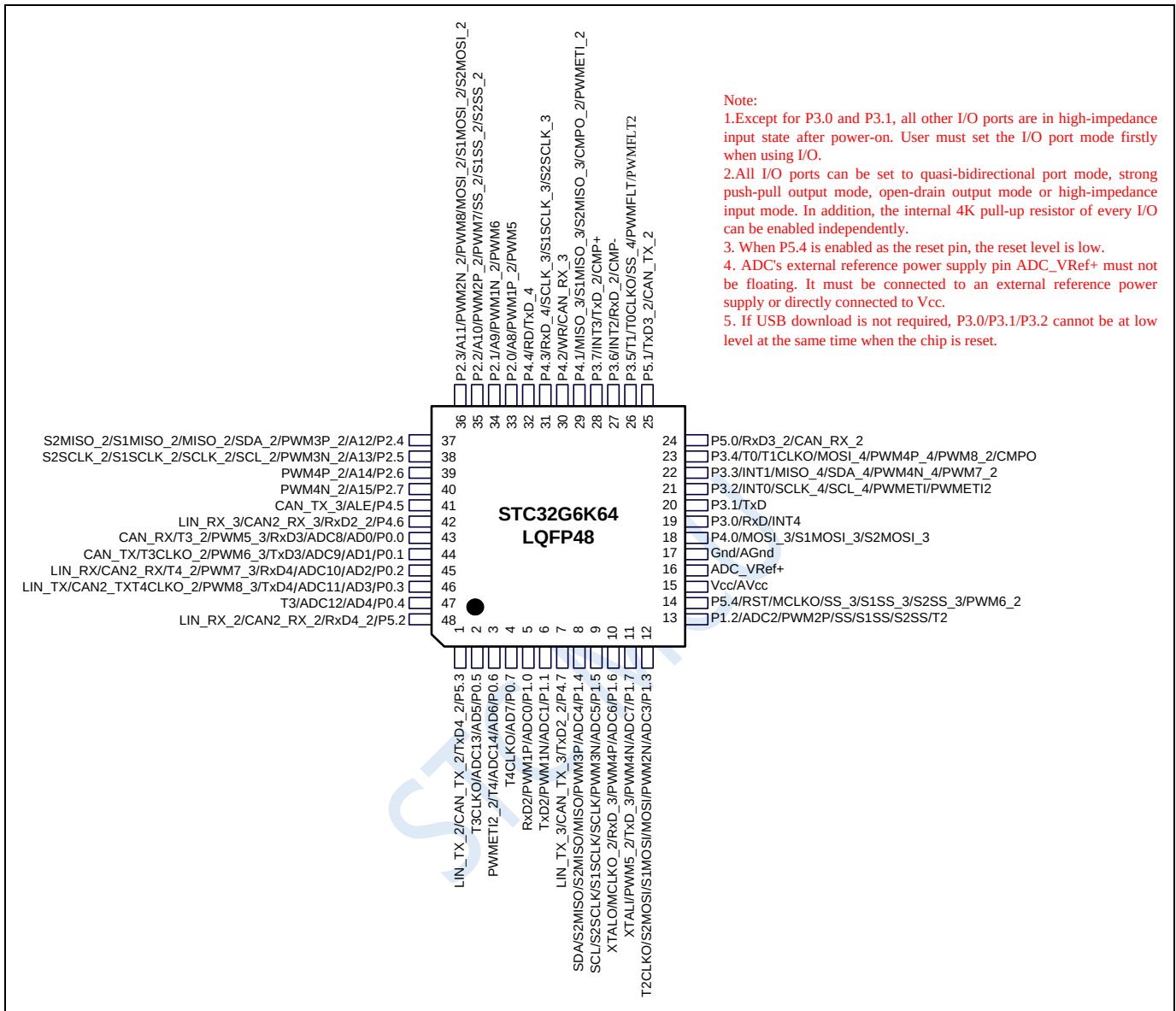
Notes on using xdata:

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

➤ Clock

- ✓ Internal high precise RC clock IRC(IRC for short), adjustable while ISP and can be divided to lower frequency by user software.
 - ✓ Error: $\pm 0.3\%$ (at the temperature 25℃)
 - ✓ $-1.35\% \sim +1.30\%$ temperature drift (at the temperature range of -40°C to $+85^{\circ}\text{C}$)
 - ✓ $-0.76\% \sim +0.98\%$ temperature drift (at the temperature range of -20°C to 65°C)
- ✓ Internal 32KHz low speed IRC with large error
- ✓ External crystal (4MHz~33MHz) and external clock
- ✓ Internal PLL output clock
- Users can freely choose the above 4 clock sources
- **Reset**
 - ✓ Hardware reset
 - ✓ Power-on reset. Measured voltage is 1.7V~1.9V. (Effective when the chip does not enable the low voltage reset function)
 - ✓ Reset by reset pin. The default function of P5.4 is the I/O port. The P5.4 pin can be set as the reset pin while ISP download. (Note: When the P5.4 pin is set as the reset pin, the reset level is low.)
 - ✓ Watch dog timer reset
 - ✓ Low voltage detection reset. 4 low voltage detection levels are provided, 2.0V, 2.3V, 2.7V, 3.0V.
 - ✓ Software reset
 - ✓ Writing the reset trigger register using software
- **Interrupts**
 - ✓ 48 interrupt sources: INT0, INT1, INT2, INT3, INT4, timer 0, timer 1, timer 2, timer 3, timer 4, USART1, USART2, UART 3, UART 4, ADC, LVD, SPI, I²C, comparator, PWMA, PWMB, CAN, CAN2, LIN, LCMIF color screen interface, RTC, all I/O interrupts (8 groups), DMA receive and transmit interrupts of USART 1, DMA receive and transmit interrupts of USART 2, DMA receive and transmit interrupts of UART 3, DMA receive and transmit interrupts of UART 4, DMA interrupt of I2C, DMA interrupt of SPI, DMA interrupt of ADC, DMA interrupt of LCD driver and DMA interrupt of memory-to-memory.
 - ✓ 4 interrupt priority levels
- **Digital peripherals**
 - ✓ 5 16-bit timers: timer0, timer1, timer2, timer3, timer4, where the mode 3 of timer 0 has the Non-Maskable Interrupt (NMI in short) function. Mode 0 of timer 0 and timer 1 is 16-bit Auto-reload mode.
 - ✓ 2 high speed USARTs: USART1, USART2, whose maximum baudrate clock may be FOSC/4. The following modes are supported, synchronous serial port mode, asynchronous serial port mode, SPI mode, LIN mode, infrared mode (IrDA), smart card mode (ISO7816)
 - ✓ 2 high speed UARTs: UART3, UART4, whose maximum baudrate clock may be FOSC/4
 - ✓ 2 groups of enhanced PWM, which can realize 8 channels(4 groups complementary symmetry) control signals with dead time, and support external fault detection function.
 - ✓ SPI: Master mode, slave mode or master/slave automatic switch mode are supported.
 - ✓ I²C: Master mode or slave mode are supported.
 - ✓ ICE: Hardware support emulation.
 - ✓ RTC: Support year, month, day, hour, minute, second, sub-second (1/128 second). And supports clock interrupt and a set of alarm clocks.
 - ✓ CAN: Two independent CAN 2.0 control units.
 - ✓ LIN: An independent LIN control unit (supports versions 1.3 and 2.1), USART1 and USART2 can support two sets of LIN.
 - ✓ MDU32: Hardware 32-bit Multiplier and Divider which supports 32-bit divided by 32-bit, 32-bit multiplied by 32-bit operations.
 - ✓ I/O port interrupt: All I/Os support interrupts, each group of I/O interrupts has an independent interrupt entry address, all I/O interrupts can support 4 types interrupt mode: high level interrupt, low level interrupt, rising edge interrupt, falling edge interrupt. Provides 4 levels of interrupt priority and supports power-down wake-up function.
 - ✓ LCD driver: support 8080 and 6800 interface, and support 8-bit and 16-bit data width.
 - ✓ DMA: Support SPI shift to receive data to memory, SPI shift to send data from memory, I2C send data from memory, I2C receive data to memory, USART 1/2 and UART 3/4 receive data to memory, USART 1/2 and UART 3/4 send data from memory, ADC automatically sample data to memory (calculate average value at the same time), LCD driver send data from memory, and copy data from memory to memory
 - ✓ Hardware digital ID: support 32 bytes
- **Analog peripherals**
 - ✓ Ultra high speed ADC which supports 12-bit precision 15 channels (channel 0 to channel 14) analog-to-digital conversion. ADC channel 15 is used to test the internal reference voltage. (The default internal reference voltage is 1.19V when the chip is shipped, the error is $\pm 1\%$)
 - ✓ Comparator: A set of comparator
- **GPIO**
 - ✓ Up to 45 GPIOs: P0.0~P0.7, P1.0~P1.7, P2.0~P2.7, P3.0~P3.7, P4.0~P4.7, P5.0~P5.4
 - ✓ 4 modes for all GPIOs: quasi_bidirectional mode, push-pull outputmode, open drain mode, high-impedance input mode
 - ✓ Except for P3.0 and P3.1, all other I/O ports are in a high-impedance state after power-on. User must set the I/O ports mode before using them. In addition, the internal 4K pull-up resistor of every I/O can be enabled independently.
- **Package**
 - ✓ LQFP48, LQFP32, PDIP40

2.10.2 Pinouts



2.10.3 Pin descriptions

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

Pin number		name	type	description
LQFP48				
11		P1.7	I/O	Standard IO port
		ADC7	I	ADC analog input 7
		TxD_3	O	Transmit pin of USART 1
		PWM4N	I/O	Capture of external signal/Negative of PWM4 pulse output
		PWM5_2	I/O	Capture of external signal/Pulse output of PWM5
		XTALI	I	Connect to external oscillator
12		P1.3	I/O	Standard IO port
		ADC3	I	ADC analog input 3
		MOSI	I/O	Master Output/Slave Input of SPI
		S1MOSI	I/O	Master Output/Slave Input of USART1-SPI
		S2MOSI	I/O	Master Output/Slave Input of USART2-SPI
		PWM2N	I/O	Capture of external signal/Negative of PWM2 pulse output
		T2CLKO	O	Clock out of timer 2
13		P1.2	I/O	Standard IO port
		ADC2	I	ADC analog input 2
		PWM2P	I/O	Capture of external signal/Positive of PWM2 pulse output
		SS	I	Slave selection of SPI (it is output with regard to master)
		S1SS	I	Slave selection of USART1-SPI (it is output with regard to master)
		S2SS	I	Slave selection of USART2-SPI (it is output with regard to master)
		T2	I	Timer2 external input
14		P5.4	I/O	Standard IO port
		RST	I	Reset pin (low level reset)
		MCLKO	O	Main clock output
		SS_3	I	Slave selection of SPI (it is output with regard to master)
		S1SS_3	I	Slave selection of USART1-SPI (it is output with regard to master)
		S2SS_3	I	Slave selection of USART2-SPI (it is output with regard to master)
		PWM6_2	I/O	Capture of external signal/Pulse output of PWM6
15		Vcc	Vcc	Power Supply
		AVcc	Vcc	ADC Power Supply
16		VRef+	I	Reference voltage pin of ADC
17		Gnd	Gnd	Ground
		AGnd	Gnd	ADC Ground
		VRef-	I	Reference voltage ground pin of ADC
18		P4.0	I/O	Standard IO port
		MOSI_3	I/O	Master Output/Slave Input of SPI
		S1MOSI_3	I/O	Master Output/Slave Input of USART1-SPI
		S2MOSI_3	I/O	Master Output/Slave Input of USART2-SPI
19		P3.0	I/O	Standard IO port
		RxD	I	Input of USART1
		INT4	I	External interrupt4
20		P3.1	I/O	Standard IO port
		TxD	O	Transmit pin of USART1
21		P3.2	I/O	Standard IO port
		INT0	I	External interrupt0
		SCLK_4	I/O	Serial Clock of SPI
		SCL_4	I/O	Serial Clock line of I2C
		PWMET1	I	PWM external trigger input pin
		PWMET2	I	PWM external trigger input pin 2

Pin number		name	type	description
LQFP48				

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

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

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

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

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

2.11.1 Features and Price

products supply information		Source	Availability
Price & Package	PDP40	✓	✓
	LQFP32<9mm*9mm>	✓	✓
	QFN48 <6mm*6mm>	✓	✓
	LQFP48 <9mm*9mm>	✓	✓
Online debug itself			
Full speed USB, support hardware USB download and debug directly			
Support RS485 download			
Password can be set for next update			
Program encrypted transmission (Anti-blocking)			
Clock output and Reset			
Internal high precision Clock (adjustabl under 33MHz)			
Internal high reliable reset circuit with 4 levels optional reset threshold voltage			
Watch-dog Timer			
Internal LVD interrupt (can wake-up CPU)			
Comparator (May be used as ADC to detect external power-down)			
DMA 15 channels high speed ADC (8 PWMs can be used as 8 DACs)			
Power-down Wake-up timer			
16-bit advanced PWM timer with Complementary symmetrical dead-time			
Timers/Counters (T0-T4 pin Cn wake-up CPU)			
FPMU(single-precision floating-point arithmetic)			
MDU32 (Hardware 32-bit Multiplier and Divider)			
I²C which can wake-up CPU			
DMA SPI which can wake-up CPU			
LIN			
CAN			
DMA USARTs which can wake-up CPU			
DMA UARTs which can wake-up CPU			
RTC			
DMA 8080/6800 interface/ LCM driver(8-bit and 16-bit)			
All I/O ports support interrupts and can wake up MCU			
Traditional I/O interrupt(1INT0/INT1/INT2/INT3/INT4) (can wake-up CPU)			
Maximum I/O Lines			
EEPROM 100 thousand times) (Byte)			
Enhanced Dual DPTR increasing or decreasing			
xdata Internal extended SRAM (Byte)			
edata Internal extended DATA RAM which can be used as stack(Byte)			
Flash Code Memory (100 thousand times) (Byte)			
Operating voltage (V)			
MCU			
STC32F12K48		1.9-5.5	
STC32F12K60		1.9-5.5	

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

- ✓ 1.9V~5.5V
- ✓ Built-in LDO

✓ -40°C~85°C

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

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

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

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

software.

- ✓ Error: $\pm 0.3\%$ (at the temperature 25°C)
 - ✓ $-1.35\% \sim +1.30\%$ temperature drift (at the temperature range of -40°C to $+85^{\circ}\text{C}$)
 - ✓ $-0.76\% \sim +0.98\%$ temperature drift (at the temperature range of -20°C to 65°C)
 - ✓ Internal 32KHz low speed IRC with large error
 - ✓ External crystal (4MHz~33MHz) and external clock
 - ✓ Internal PLL output clock
- Users can freely choose the above 4 clock sources

➤ Reset

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

➤ Interrupts

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

➤ Digital peripherals

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

➤ Analog peripherals

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

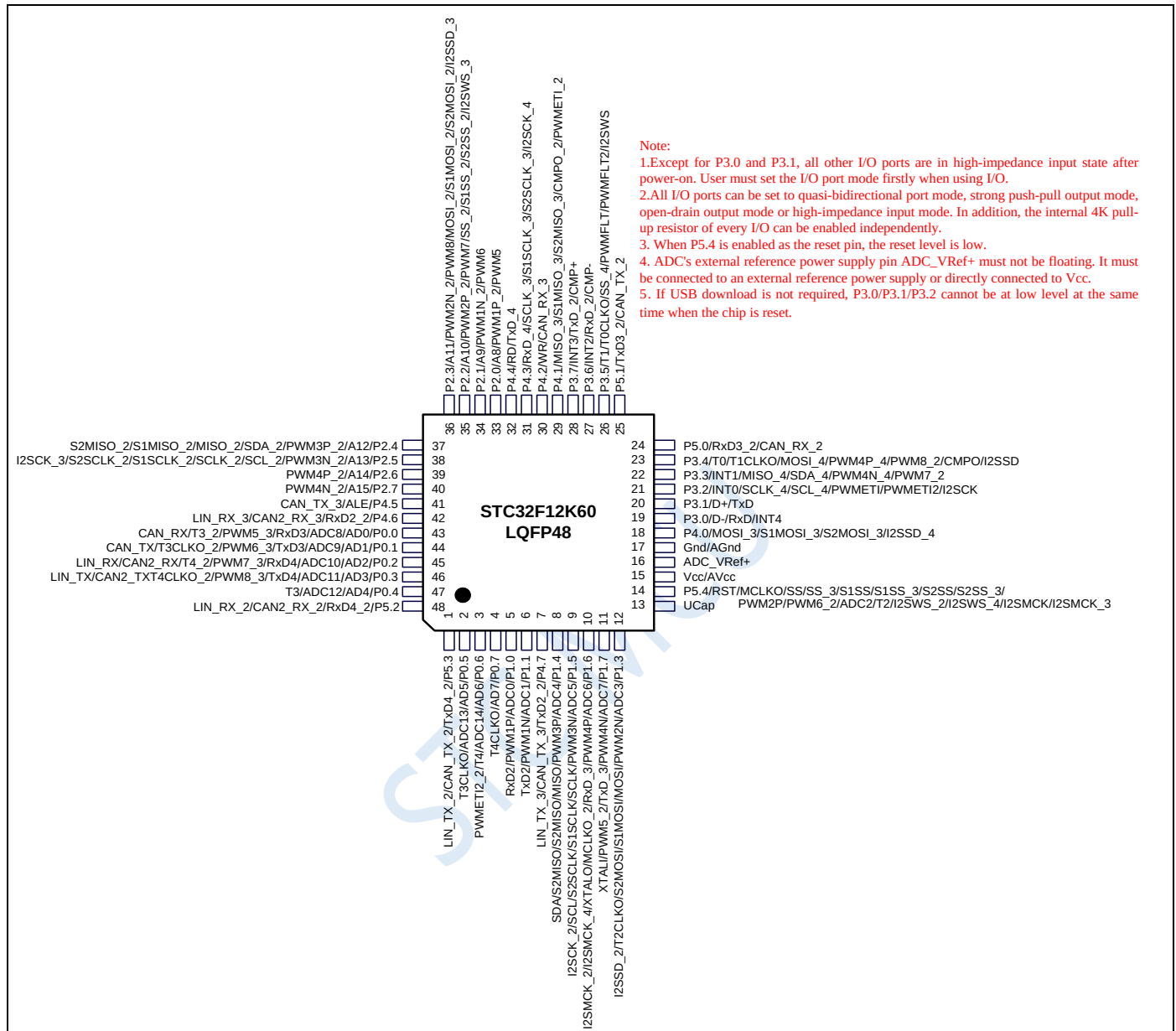
➤ GPIO

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

➤ Package

STC MCU

2.11.2 Pinouts



2.11.3 Pin descriptions

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

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

Pin number		name	type	description
LQFP48				

21		P3.2	I/O	Standard IO port
		INT0	I	External interrupt0
		SCLK_4	I/O	Serial Clock of SPI
		SCL_4	I/O	Serial Clock line of I2C
		PWMET1	I	PWM external trigger input pin
		PWMET2	I	PWM external trigger input pin 2
		I2SCK	I/O	Clock line of I2S
22		P3.3	I/O	Standard IO port
		INT1	I	External interrupt1
		MISO_4	I/O	Master Input/Slave Output of SPI
		SDA_4	I/O	Serial data line of I2C
		PWM4N_4	I/O	Capture of external signal/Negative of PWM4 pulse output
		PWM7_2	I/O	Capture of external signal/Pulse output of PWM7
23		P3.4	I/O	Standard IO port
		T0	I	Timer0 external input
		T1CLKO	O	Clock out of timer 1
		MOSI_4	I/O	Master Output/Slave Input of SPI
		PWM4P_4	I/O	Capture of external signal/Positive of PWM4 pulse output
		PWM8_2	I/O	Capture of external signal/Pulse output of PWM8
		CMPO	O	Output of comparator
24		I2SSD	I/O	Data line of I2S
		P5.0	I/O	Standard IO port
		RxD3_2	I	Input of UART3
		CMP+_2	I	Positive input of comparator
25		CAN_RX_2	I	Receive pin of CAN
		P5.1	I/O	Standard IO port
		TxD3_2	O	Transmit pin of UART 3
		CMP+_3	I	Positive input of comparator
26		CAN_TX_2	I	Transmit pin of CAN
		P3.5	I/O	Standard IO port
		T1	I	Timer1 external input
		T0CLKO	O	Clock out of timer 0
		SS_4	I	Slave selection of SPI (it is output with regard to master)
		PWMFLT	I	Enhance PWMA external anomaly detection pin
27		I2SWS	I/O	I2S channel selection line
		P3.6	I/O	Standard IO port
		INT2	I	External interrupt2
		RxD_2	I	Input of USART1
28		CMP-	I	Negative input of comparator
		P3.7	I/O	Standard IO port
		INT3	I	External interrupt3
		TxD_2	O	Transmit pin of USART 1
29		CMP+	I	Positive input of comparator
		P4.1	I/O	Standard IO port
		MISO_3	I/O	Master Input/Slave Output of SPI
		S1MISO_3	I/O	Master Input/Slave Output of USART1-SPI
		S2MISO_3	I/O	Master Input/Slave Output of USART2-SPI
		CMPO_2	O	Output of comparator
30		PWMET1_3	I	PWM external trigger input pin
		P4.2	I/O	Standard IO port
		WR	O	WRITE signal of external bus
		CAN_RX_3	I	Receive pin of CAN

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

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

3 Function pins switch

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

3.1 Register related to function pin switch

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

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

3.1.1 Peripheral port switch register 1(P_SW1)

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

S1_S[1:0]: USART1 pin selection bits

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

SPI_S[1:0]: SPI pin selection bits

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

3.1.2 Peripheral port switch register 2(P_SW2)

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

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

0: Access to XFR is prohibited

1: Enable access to XFR.

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

I2C_S[1:0]: I²C pin selection bits

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

CMPO_S: Comparator output pin selection bit

CMPO_S	CMPO
0	P3.4
1	P4.1

S4_S: UART4 pin selection bit

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

S3_S: UART3 pin selection bit

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

S2_S: USART2 pin selection bit

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

3.1.3 Clock selection register(MCLKOCR)

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

MCLKO_S: Main clock out pin selection bit

MCLKO_S	MCLKO
0	P5.4
1	P1.6

3.1.4 T3/T4 selection register(T3T4PIN)

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

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

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

3.1.5 Advanced PWM selection register(PWMx_PS)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.1.6 Advanced PWM Pin selection register(PWMx_ETRPS)

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

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

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

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

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

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

BRKAPS [1:0]	PWMFLT

00	P3.5
01	Comparator output

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

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

STC MCU

3.2 Example Routines

3.2.1 USART1 switch

C language code

// Operating frequency for test is 11.0592MHz

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

```
sfr      P_SW1      = 0xa2;
```

```
sfr      P0M1       = 0x93;
```

```
sfr      P0M0       = 0x94;
```

```
sfr      P1M1       = 0x91;
```

```
sfr      P1M0       = 0x92;
```

```
sfr      P2M1       = 0x95;
```

```
sfr      P2M0       = 0x96;
```

```
sfr      P3M1       = 0xb1;
```

```
sfr      P3M0       = 0xb2;
```

```
sfr      P4M1       = 0xb3;
```

```
sfr      P4M0       = 0xb4;
```

```
sfr      P5M1       = 0xc9;
```

```
sfr      P5M0       = 0xca;
```

```
void main()
```

```
{
```

```
    P0M0 = 0x00;
```

```
    P0M1 = 0x00;
```

```
    P1M0 = 0x00;
```

```
    P1M1 = 0x00;
```

```
    P2M0 = 0x00;
```

```
    P2M1 = 0x00;
```

```
    P3M0 = 0x00;
```

```
    P3M1 = 0x00;
```

```
    P4M0 = 0x00;
```

```
    P4M1 = 0x00;
```

```
    P5M0 = 0x00;
```

```
    P5M1 = 0x00;
```

```
    P_SW1 = 0x00;
```

```
//RXD/P3.0, TXD/P3.1
```

```
// P_SW1 = 0x40;
```

```
//RXD_2/P3.6, TXD_2/P3.7
```

```
// P_SW1 = 0x80;
```

```
//RXD_3/P1.6, TXD_3/P1.7
```

```
// P_SW1 = 0xc0;
```

```
//RXD_4/P4.3, TXD_4/P4.4
```

```
    while (1);
```

```
}
```

Assembly code

; Operating frequency for test is 11.0592MHz

```
P_SW1      DATA      0A2H
```

```
P0M1       DATA      093H
```

```
P0M0       DATA      094H
```

```
P1M1       DATA      091H
```

```
P1M0       DATA      092H
```

```
P2M1       DATA      095H
```

```

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

```

```

ORG        0000H
LJMP       MAIN

```

```

MAIN:      ORG        0100H

```

```

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

```

```

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

```

```

SJMP       $

```

```

END

```

3.2.2 USART2 switch

C language code

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

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

```
sfr        P_SW2      = 0xba;
```

```
sfr        P0M1       = 0x93;
```

```
sfr        P0M0       = 0x94;
```

```
sfr        P1M1       = 0x91;
```

```
sfr        P1M0       = 0x92;
```

```
sfr        P2M1       = 0x95;
```

```
sfr        P2M0       = 0x96;
```

```
sfr        P3M1       = 0xb1;
```

```
sfr        P3M0       = 0xb2;
```

```
sfr        P4M1       = 0xb3;
```

```
sfr        P4M0       = 0xb4;
```

```
sfr        P5M1       = 0xc9;
```

```
sfr        P5M0       = 0xca;
```

```

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

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

    while (1);
}

```

Assembly code

; Operating frequency for test is 11.0592MHz

P_SW2	DATA	0BAH	
P0M1	DATA	093H	
P0M0	DATA	094H	
P1M1	DATA	091H	
P1M0	DATA	092H	
P2M1	DATA	095H	
P2M0	DATA	096H	
P3M1	DATA	0B1H	
P3M0	DATA	0B2H	
P4M1	DATA	0B3H	
P4M0	DATA	0B4H	
P5M1	DATA	0C9H	
P5M0	DATA	0CAH	
	ORG	0000H	
	LJMP	MAIN	
	ORG	0100H	
MAIN:	MOV	SP, #5FH	
	MOV	P0M0, #00H	
	MOV	P0M1, #00H	
	MOV	P1M0, #00H	
	MOV	P1M1, #00H	
	MOV	P2M0, #00H	
	MOV	P2M1, #00H	
	MOV	P3M0, #00H	
	MOV	P3M1, #00H	
	MOV	P4M0, #00H	
	MOV	P4M1, #00H	
	MOV	P5M0, #00H	
	MOV	P5M1, #00H	
	MOV	P_SW2, #00H	;RXD2/P1.0, TXD2/P1.1

```

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

          SJMP          $

          END

```

3.2.3 UART3 switch

C language code

// Operating frequency for test is 11.0592MHz

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

```

sfr      P_SW2      = 0xba;

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

```

```
void main()
```

```

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

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

    while (1);
}

```

Assembly code

// Operating frequency for test is 11.0592MHz

```

P_SW2      DATA      0BAH

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

```

```

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

```

```

ORG        0000H
LJMP       MAIN

```

```

MAIN:      ORG        0100H

```

```

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

```

```

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

```

```

SJMP       $

```

```

END

```

3.2.4 UART4 switch

C language code

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

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

```
sfr        P_SW2      =    0xba;
```

```
sfr        P0M1       =    0x93;
```

```
sfr        P0M0       =    0x94;
```

```
sfr        P1M1       =    0x91;
```

```
sfr        P1M0       =    0x92;
```

```
sfr        P2M1       =    0x95;
```

```
sfr        P2M0       =    0x96;
```

```
sfr        P3M1       =    0xb1;
```

```
sfr        P3M0       =    0xb2;
```

```
sfr        P4M1       =    0xb3;
```

```
sfr        P4M0       =    0xb4;
```

```
sfr        P5M1       =    0xc9;
```

```
sfr        P5M0       =    0xca;
```

```

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

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

    while (1);
}

```

Assembly code

; Operating frequency for test is 11.0592MHz

P_SW2	DATA	0BAH	
P0M1	DATA	093H	
P0M0	DATA	094H	
P1M1	DATA	091H	
P1M0	DATA	092H	
P2M1	DATA	095H	
P2M0	DATA	096H	
P3M1	DATA	0B1H	
P3M0	DATA	0B2H	
P4M1	DATA	0B3H	
P4M0	DATA	0B4H	
P5M1	DATA	0C9H	
P5M0	DATA	0CAH	
	ORG	0000H	
	LJMP	MAIN	
	ORG	0100H	
MAIN:			
	MOV	SP, #5FH	
	MOV	P0M0, #00H	
	MOV	P0M1, #00H	
	MOV	P1M0, #00H	
	MOV	P1M1, #00H	
	MOV	P2M0, #00H	
	MOV	P2M1, #00H	
	MOV	P3M0, #00H	
	MOV	P3M1, #00H	
	MOV	P4M0, #00H	
	MOV	P4M1, #00H	
	MOV	P5M0, #00H	
	MOV	P5M1, #00H	
	MOV	P_SW2, #00H	;RXD4/P0.2, TXD4/P0.3


```

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

          SJMP          $

          END

```

3.2.5 SPI switch

C language code

// Operating frequency for test is 11.0592MHz

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

```

sfr      P_SW1      = 0xa2;

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

```

```
void main()
```

```

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

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

    while (1);
}

```

Assembly code

; Operating frequency for test is 11.0592MHz

```

P_SW1      DATA      0A2H

P0M1      DATA      093H

```

```

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

          ORG          0000H
          LJMP         MAIN

MAIN:     ORG          0100H

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

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

          SJMP         $

          END

```

3.2.6 I2C switch

C language code

// Operating frequency for test is 11.0592MHz

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

```

sfr      P_SW2      = 0xba;

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

```

```

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

```

```
void main()
```

```

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

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

    while (1);
}

```

Assembly code

; Operating frequency for test is 11.0592MHz

```

P_SW2      DATA      0BAH

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

            ORG         0000H
            LJMP        MAIN

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

```

```

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

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

SJMP     $
END

```

3.2.7 Comparator output switch

C language code

// Operating frequency for test is 11.0592MHz

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

```
sfr      P_SW2      = 0xba;
```

```
sfr      P0M1       = 0x93;
```

```
sfr      P0M0       = 0x94;
```

```
sfr      P1M1       = 0x91;
```

```
sfr      P1M0       = 0x92;
```

```
sfr      P2M1       = 0x95;
```

```
sfr      P2M0       = 0x96;
```

```
sfr      P3M1       = 0xb1;
```

```
sfr      P3M0       = 0xb2;
```

```
sfr      P4M1       = 0xb3;
```

```
sfr      P4M0       = 0xb4;
```

```
sfr      P5M1       = 0xc9;
```

```
sfr      P5M0       = 0xca;
```

```
void main()
```

```
{
```

```
    P0M0 = 0x00;
```

```
    P0M1 = 0x00;
```

```
    P1M0 = 0x00;
```

```
    P1M1 = 0x00;
```

```
    P2M0 = 0x00;
```

```
    P2M1 = 0x00;
```

```
    P3M0 = 0x00;
```

```
    P3M1 = 0x00;
```

```
    P4M0 = 0x00;
```

```
    P4M1 = 0x00;
```

```
    P5M0 = 0x00;
```

```
    P5M1 = 0x00;
```

```
    P_SW2 = 0x00;      //CMPO/P3.4
```

```
//    P_SW2 = 0x08;      //CMPO_2/P4.1
```

```
    while (1);
```

```
}
```

Assembly code

; Operating frequency for test is 11.0592MHz

```

P_SW2      DATA      0BAH

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

          ORG          0000H
          LJMP         MAIN

MAIN:      ORG          0100H

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

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

          SJMP         $

          END

```

3.2.8 Main clock output switch

C language code

// Operating frequency for test is 11.0592MHz

#include "reg51.h"

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

sfr P_SW2 = 0xba;

sfr P0M1 = 0x93;

sfr P0M0 = 0x94;

sfr P1M1 = 0x91;

sfr P1M0 = 0x92;

```
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;
```

```
void main()
```

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

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

    while (1);
}
```

Assembly code

; Operating frequency for test is 11.0592MHz

```

P_SW2      DATA      0BAH

CLKOCR      EQU        0FE05H

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

                ORG      0000H
                LJMP     MAIN

MAIN:          ORG      0100H

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

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

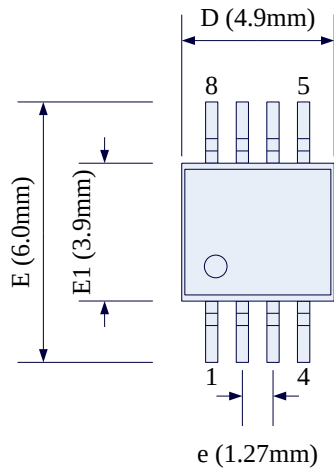
                SJMP     $

                END

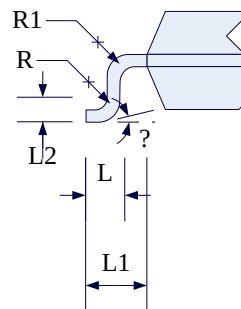
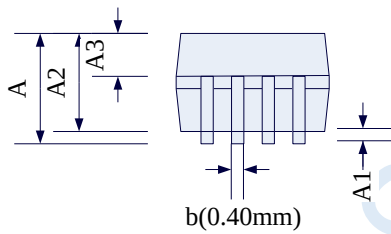
```

4 Package Dimensions

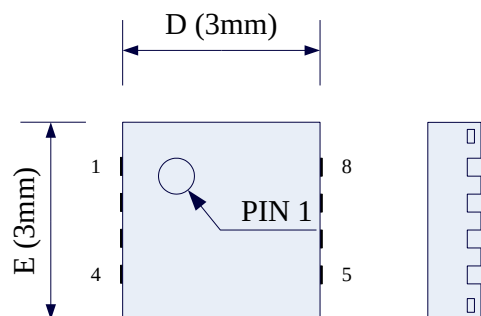
4.1 SOP8 Package mechanical data



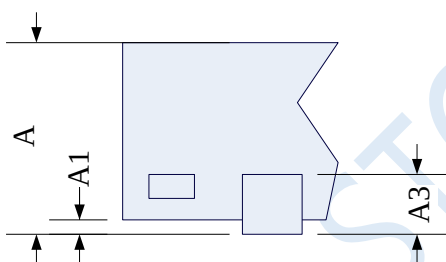
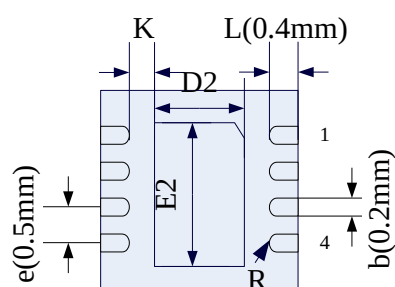
General size			
mm			
SYMBOL	MIN	TYP	MAX
A	1.35	1.60	1.75
A1	0.10	0.15	0.25
A2	1.25	1.45	1.65
A3	0.55	0.65	0.75
b	0.35	0.40	0.45
D	4.80	4.90	5.00
E	5.80	6.00	6.20
E1	3.80	3.90	4.00
e	1.27BSC		
L	0.45	0.60	0.80
L1	1.04REF		
L2	0.25BSC		
R1	0.07	-	-
R2	0.07	-	-



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

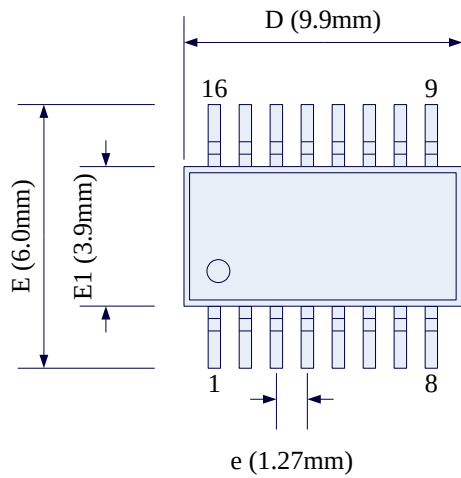


General size			
mm			
SYMBOL	MIN	TYP	MAX
A	0.70	0.75	0.80
A1	0	0.02	0.05
A2	0.50	0.55	0.60
A3	-	0.203REF	-
b	0.175	0.20	0.225
D	2.89	3.00	3.11
E	2.89	3.00	3.11
D2	1.65	1.70	1.75
E2	2.35	2.40	2.45
e	0.45	0.50	0.55
L	0.381	0.40	0.419
K	0.275REF		
R	0.125	-	-

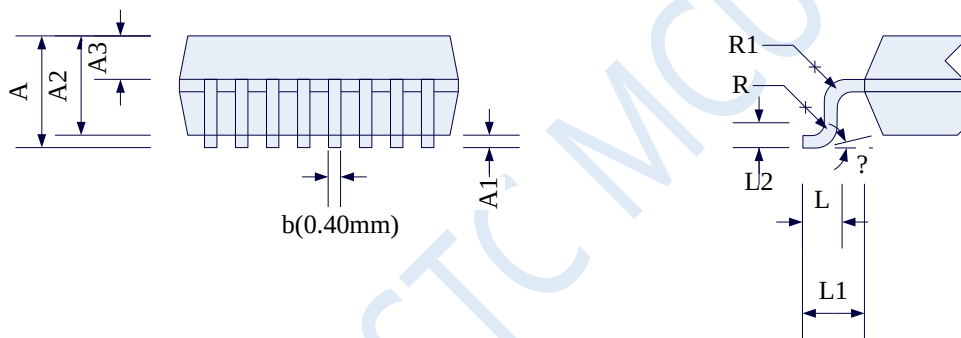


The back metal sheet (substrate) of STC's existing DFN8 packaged chip is not grounded inside the chip. It can be grounded or ungrounded on the user's PCB board, which will not affect the performance of the chip.

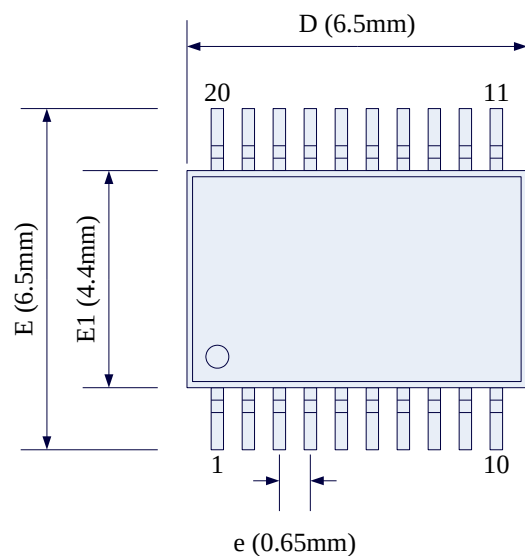
4.3 SOP16 Package mechanical data



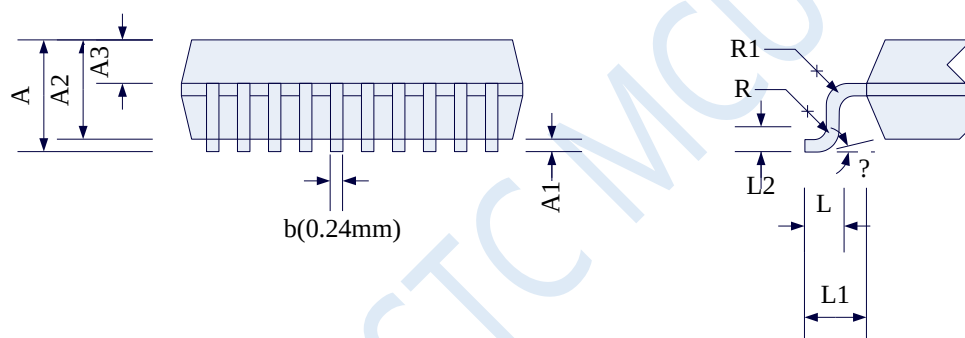
General size			
mm			
SYMBOL	MIN	TYP	MAX
A	1.35	1.60	1.75
A1	0.10	0.15	0.25
A2	1.25	1.45	1.65
A3	0.55	0.65	0.75
b	0.35	0.40	0.45
D	9.80	9.90	10.00
E	5.80	6.00	6.20
E1	3.80	3.90	4.00
e	1.27BSC		
L	0.45	0.60	0.80
L1	1.04REF		
L2	0.25BSC		
R1	0.07	-	-
R2	0.07	-	-



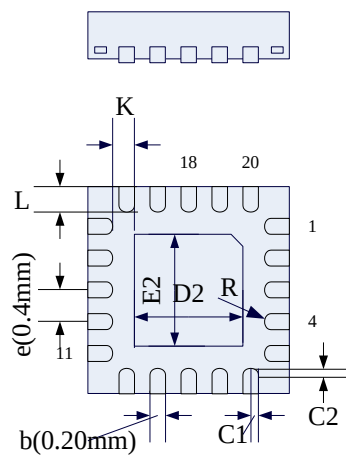
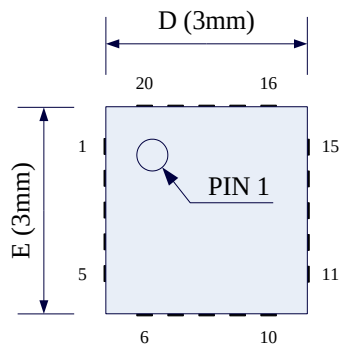
4.4 TSSOP20 Package mechanical data



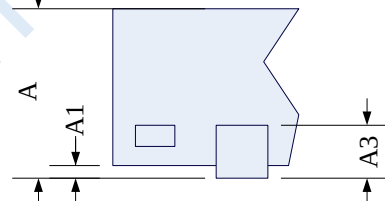
General size			
mm			
SYMBOL	MIN	TYP	MAX
A	-	-	1.20
A1	0.05	-	0.15
A2	0.90	1.00	1.05
A3	0.34	0.44	0.54
b	0.20	0.24	0.28
D	6.40	6.50	6.60
E	6.20	6.50	6.60
E1	4.30	4.40	4.50
e	0.65BSC		
L	0.45	0.60	0.75
L1	1.00REF		
L2	0.25BSC		
R1	0.09	-	-
R2	0.09	-	-



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

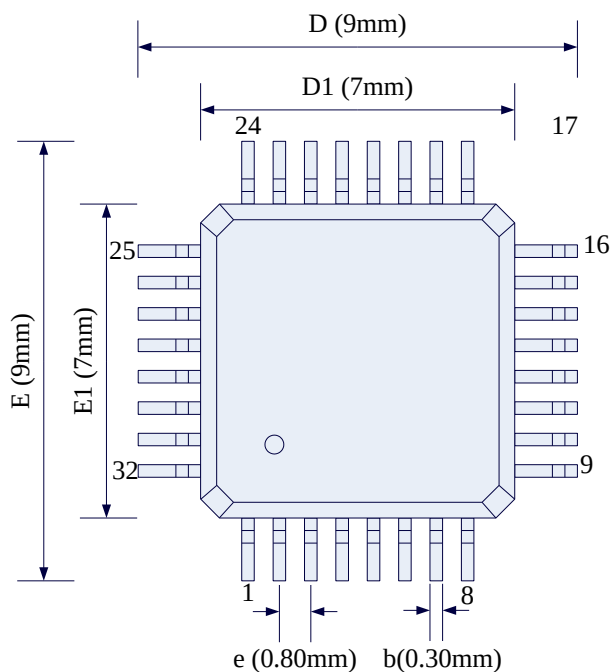


General size			
mm			
SYMBOL	MIN	TYP	MAX
A	0.70	0.75	0.80
A1	0	0.02	0.05
A2	0.50	0.55	0.60
A3	-	0.20REF	-
b	0.15	0.20	0.25
D	2.90	3.00	3.10
E	2.90	3.00	3.10
D2	1.40	1.50	1.60
E2	1.40	1.50	1.60
e	0.30	0.40	0.50
L	0.35	0.40	0.45
K	0.35REF		
R	0.085	-	-
C1	-	0.07	-
C2	-	0.07	-

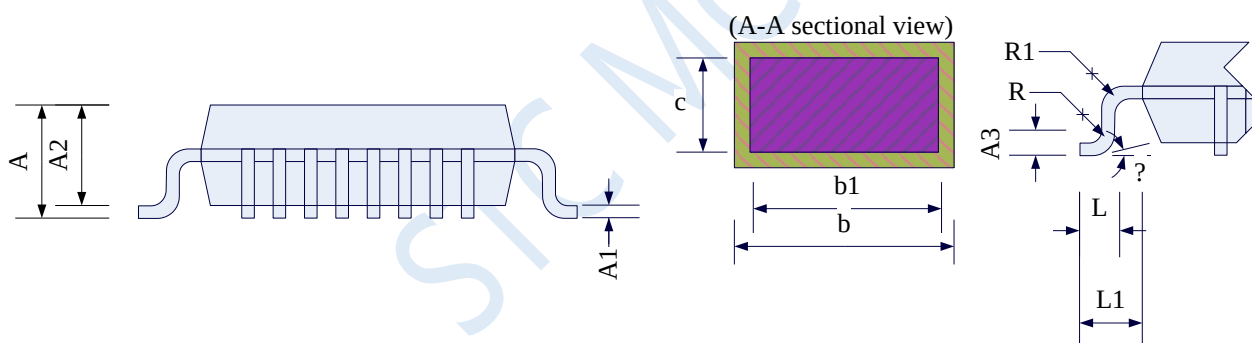


The back metal sheet (substrate) of STC's existing DFN8 packaged chip is not grounded inside the chip. It can be grounded or ungrounded on the user's PCB board, which will not affect the performance of the chip.

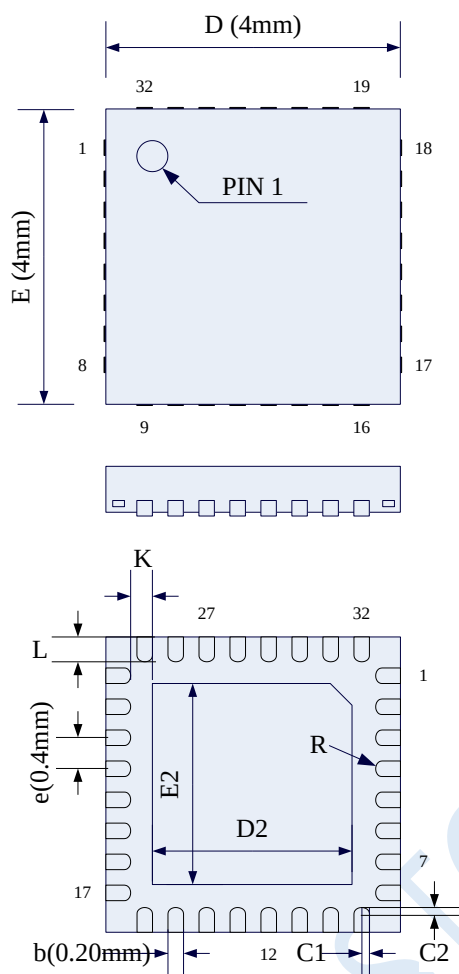
4.6 LQFP32 Package mechanical data (9mm*9mm)



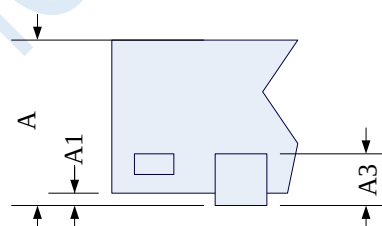
General size			
SYMBOL	MIN	TYP	MAX
A	1.45	1.55	1.65
A1	0.01	-	0.21
A2	1.35	1.40	1.45
A3	-	0.254	-
b1	0.30	0.35	0.40
b	0.31	0.37	0.43
c	-	0.127	-
D	8.80	9.00	9.20
D1	6.90	7.00	7.10
E	8.80	9.00	9.20
E1	6.90	7.00	7.10
e	0.70	0.80	0.90
L	0.43	-	0.71
L	1.00REF		
L1	0.25BSC		
R	0.1	-	0.25
R1	0.1	-	-
?	0°	-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



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

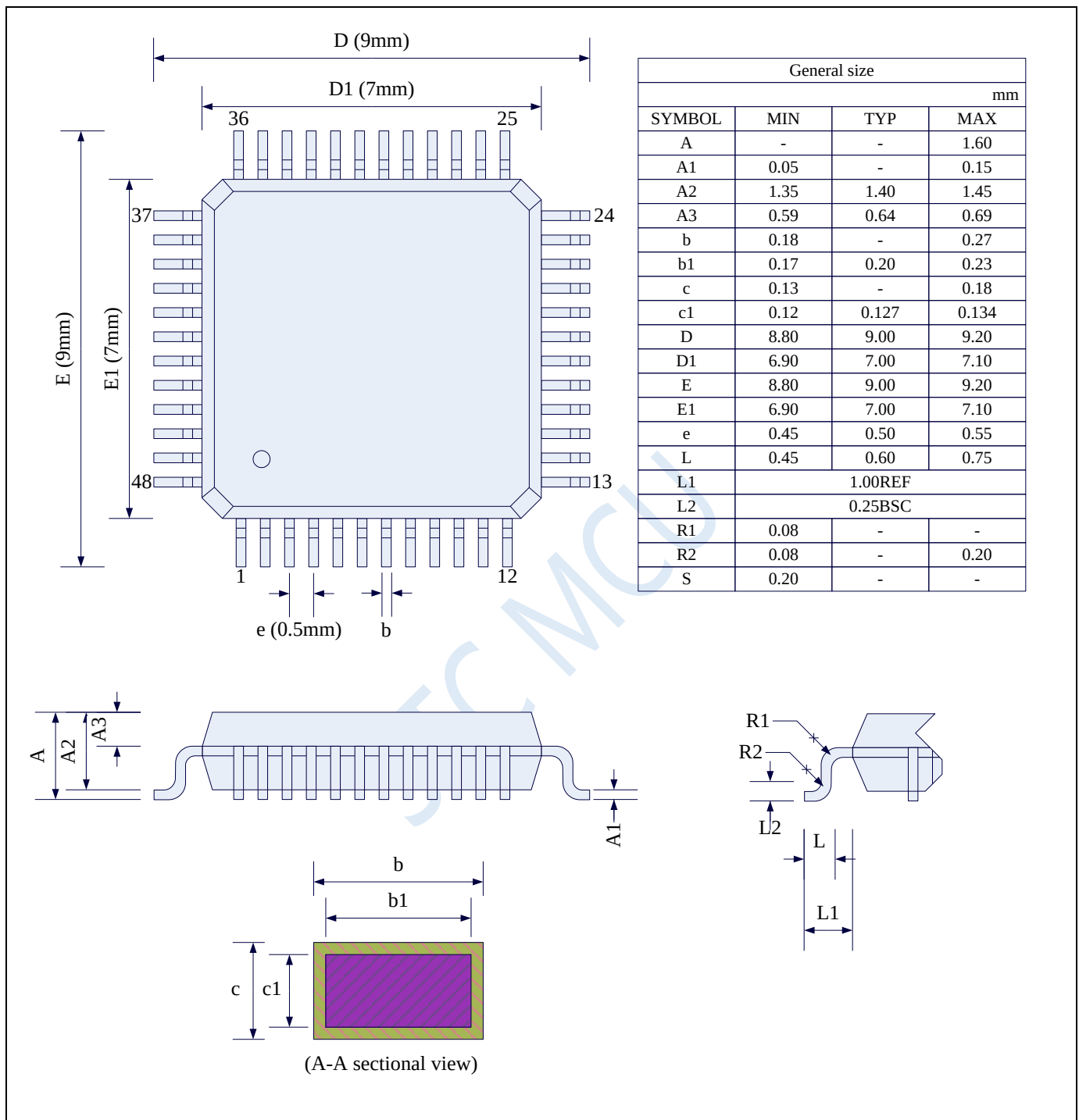


General size			
SYMBOL	MIN	TYP	MAX
A	0.70	0.75	0.80
A1	0	0.02	0.05
A2	0.50	0.55	0.60
A3	-	0.20REF	-
b	0.15	0.20	0.25
D	3.90	4.00	4.10
E	3.90	4.00	4.10
D2	2.60	2.70	2.80
E2	2.60	2.70	2.80
e	0.30	0.40	0.50
L	0.35	0.40	0.45
K	0.25REF		
R	0.09	-	-
C1	-	0.16	-
C2	-	0.16	-

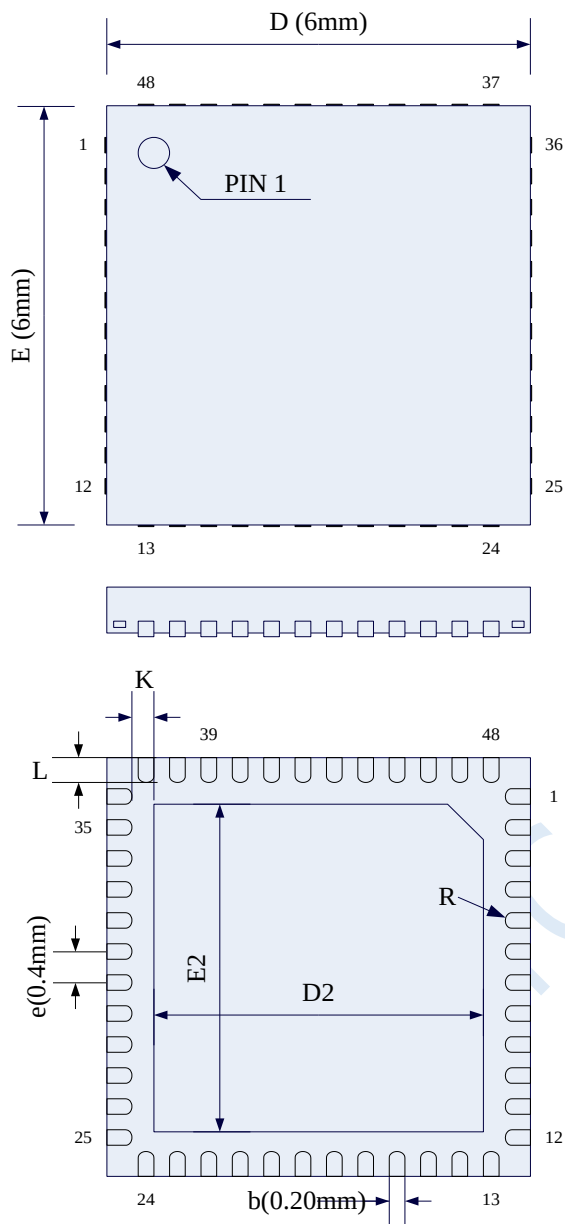


The back metal sheet (substrate) of STC's existing DFN8 packaged chip is not grounded inside the chip. It can be grounded or ungrounded on the user's PCB board, which will not affect the performance of the chip.

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



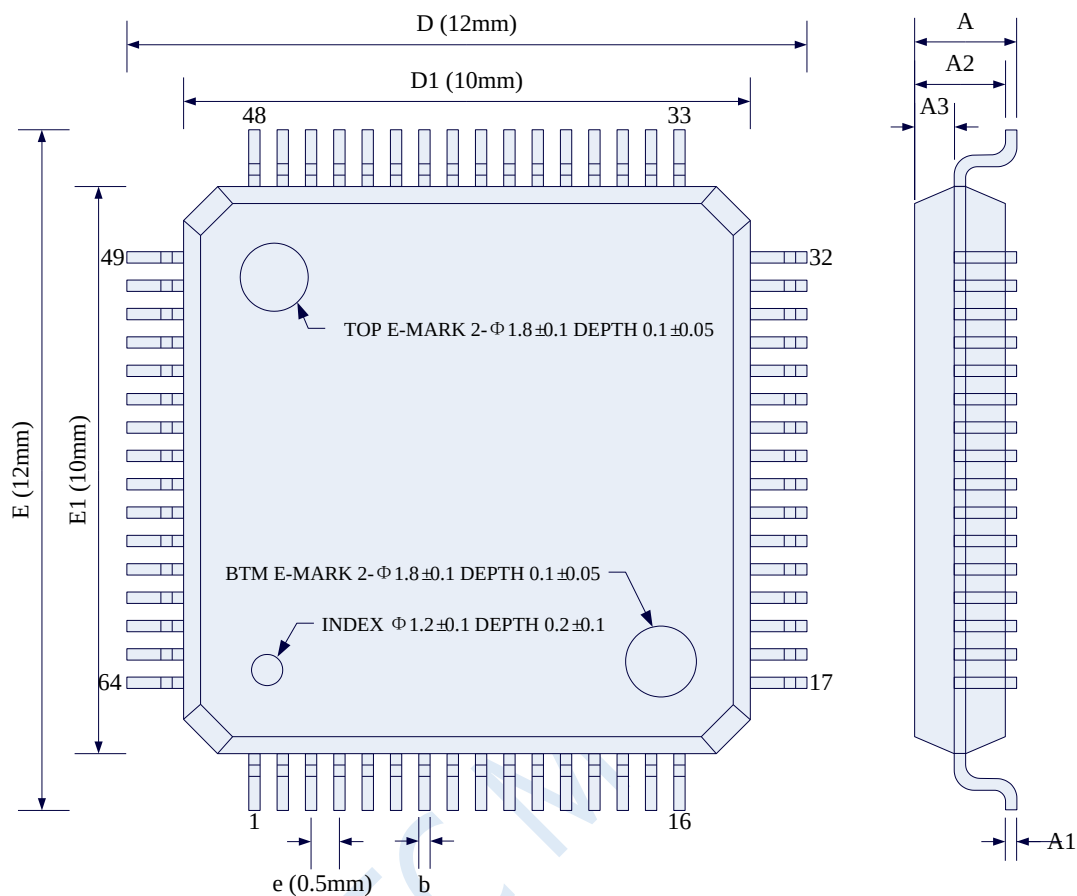
4.9 QFN48 Package mechanical data (6mm*6mm)



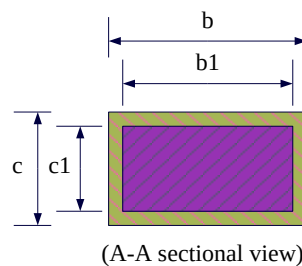
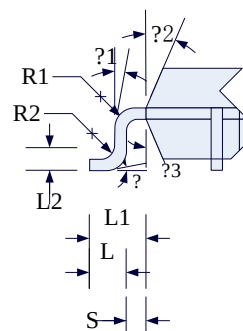
General size			
SYMBOL	MIN	TYP	MAX
A	0.70	0.75	0.80
A1	0.00	0.02	0.05
A3	0.20REF		
b	0.15	0.20	0.25
D	5.90	6.00	6.10
E	5.90	6.00	6.10
D2	3.95	4.05	4.15
E2	3.95	4.05	4.15
e	0.35	0.40	0.45
L	0.35	0.40	0.45
K	0.20		
R	0.09	-	-

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

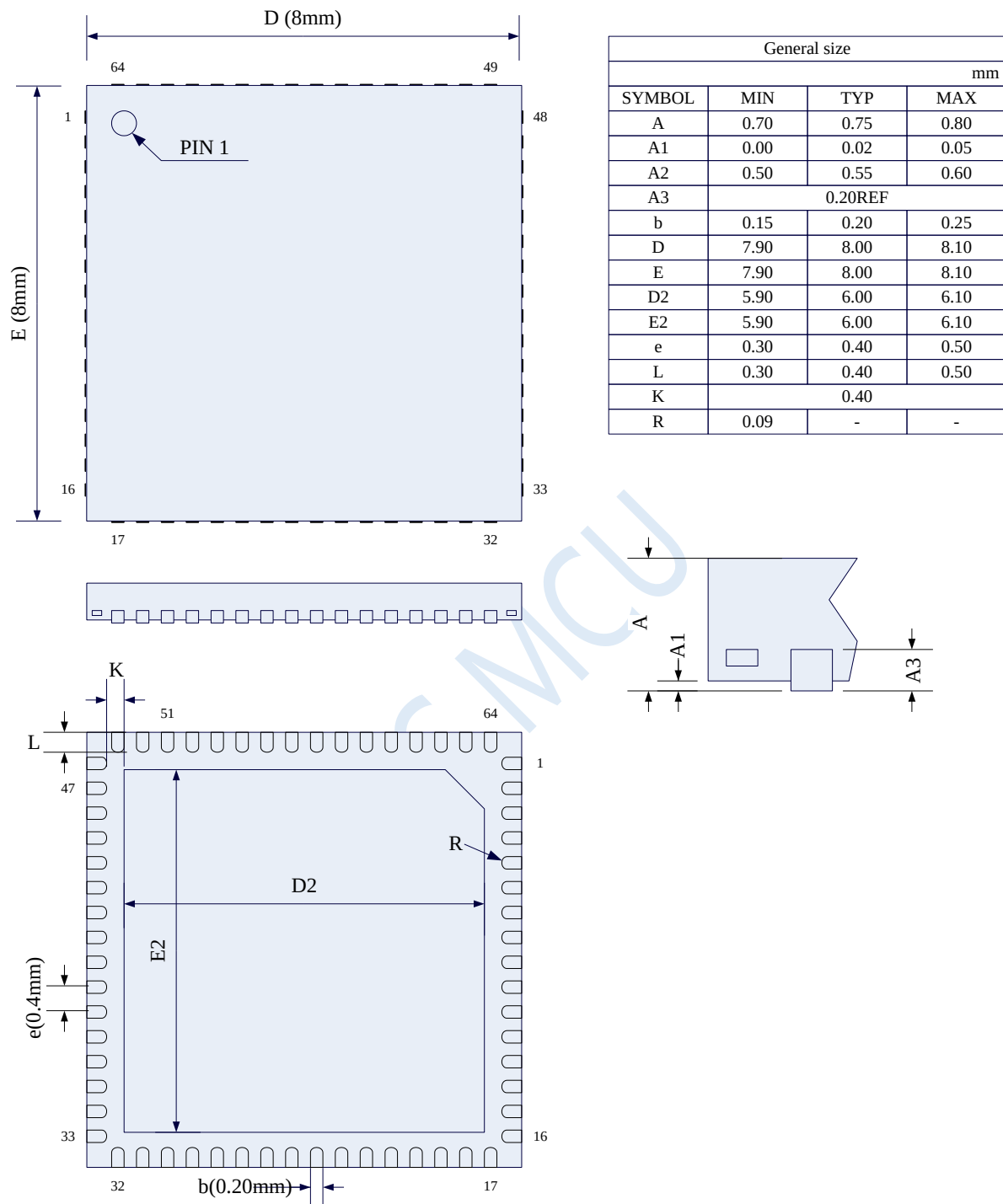
4.10 LQFP64S Package mechanical data (12mm*12mm)



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

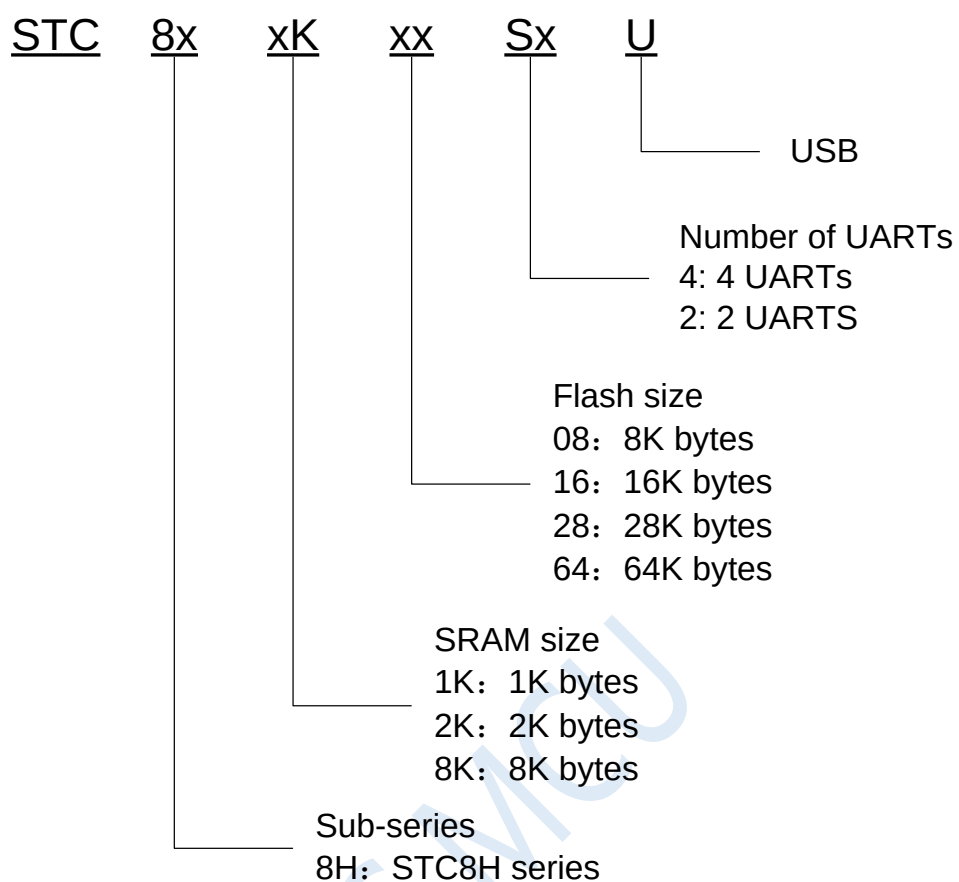


4.11 QFN64 Package mechanical data (8mm*8mm)



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

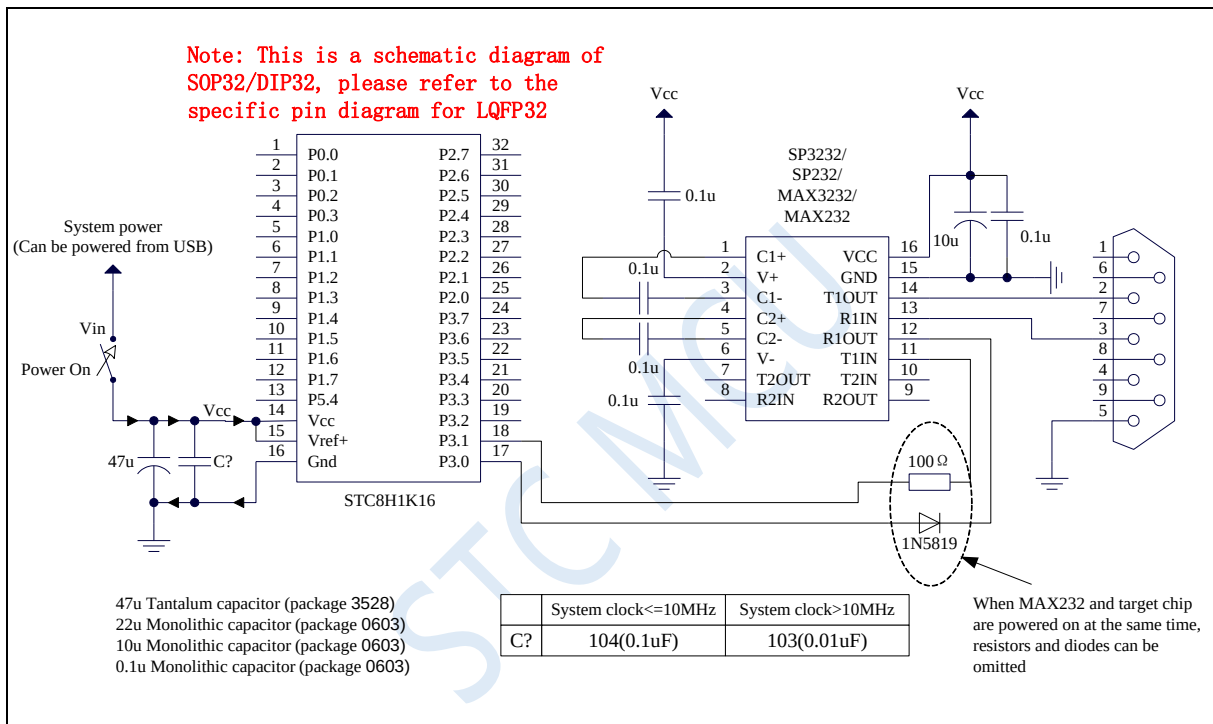
4.12 Naming rules of STC8 family



5 ISP Download and typical application circuit

5.1 STC8H series ISP download application circuit

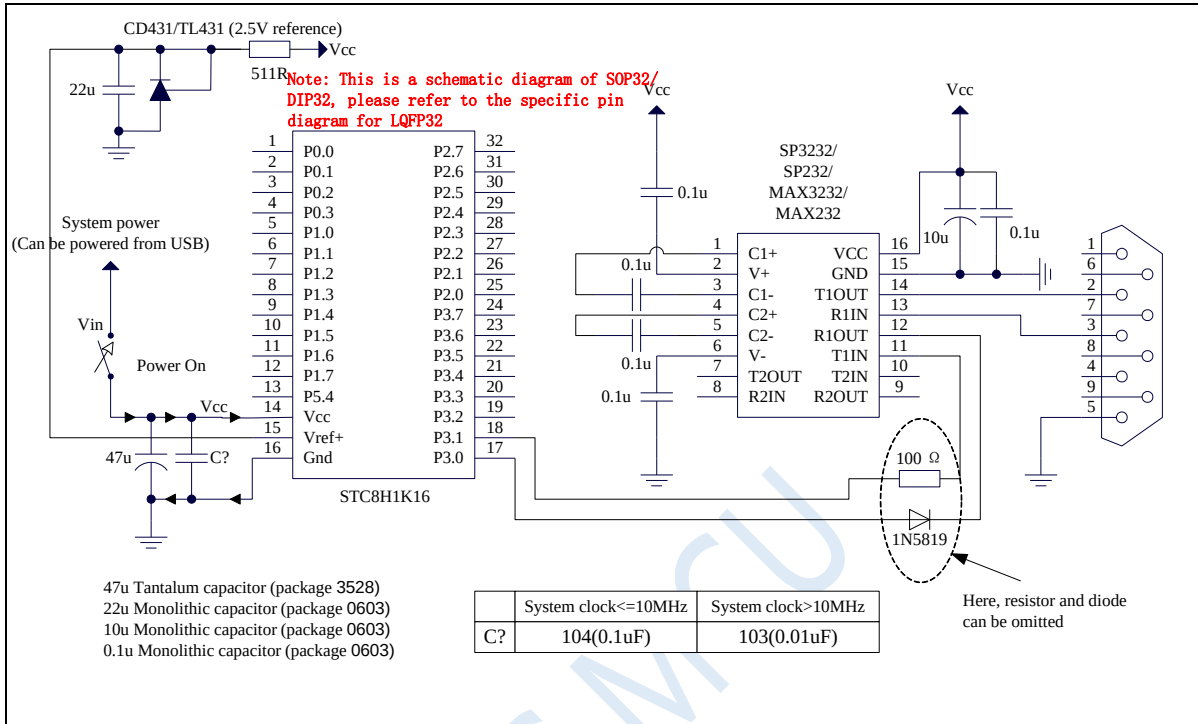
5.1.1 Download using RS-232 converter (general precision ADC), Emulation supported



ISP download steps:

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

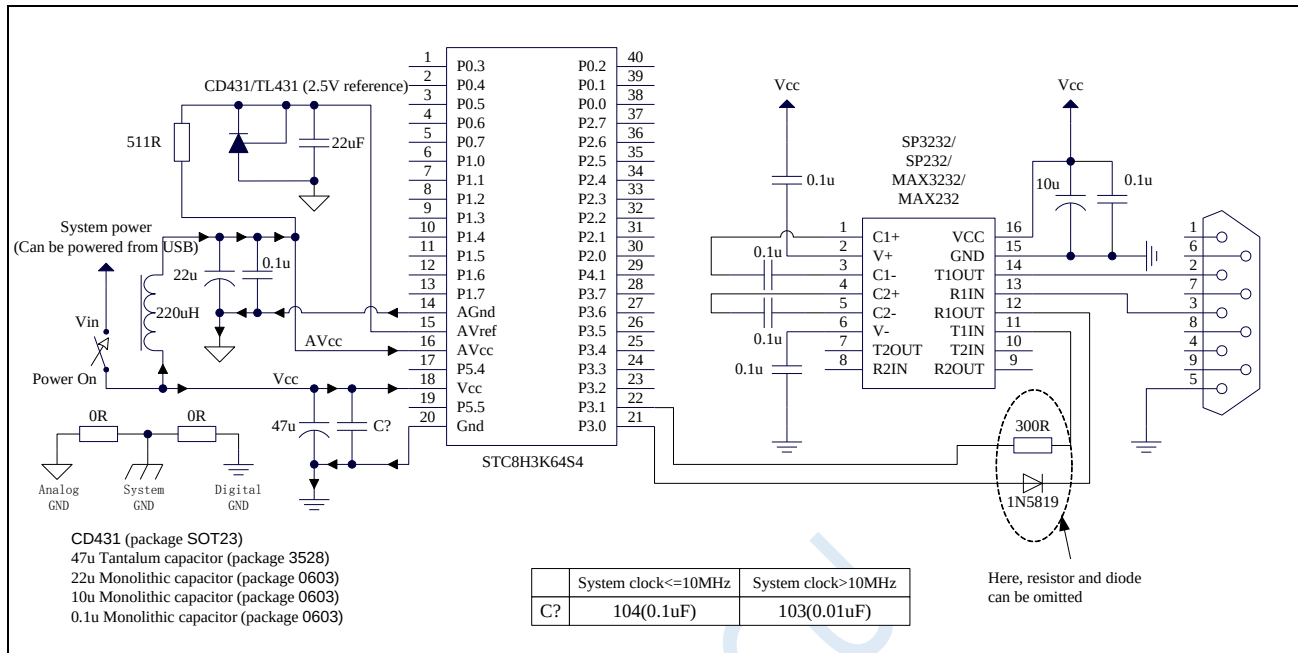
5.1.2 Download using RS-232 converter (High precision ADC), Emulation supported



ISP download steps:

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

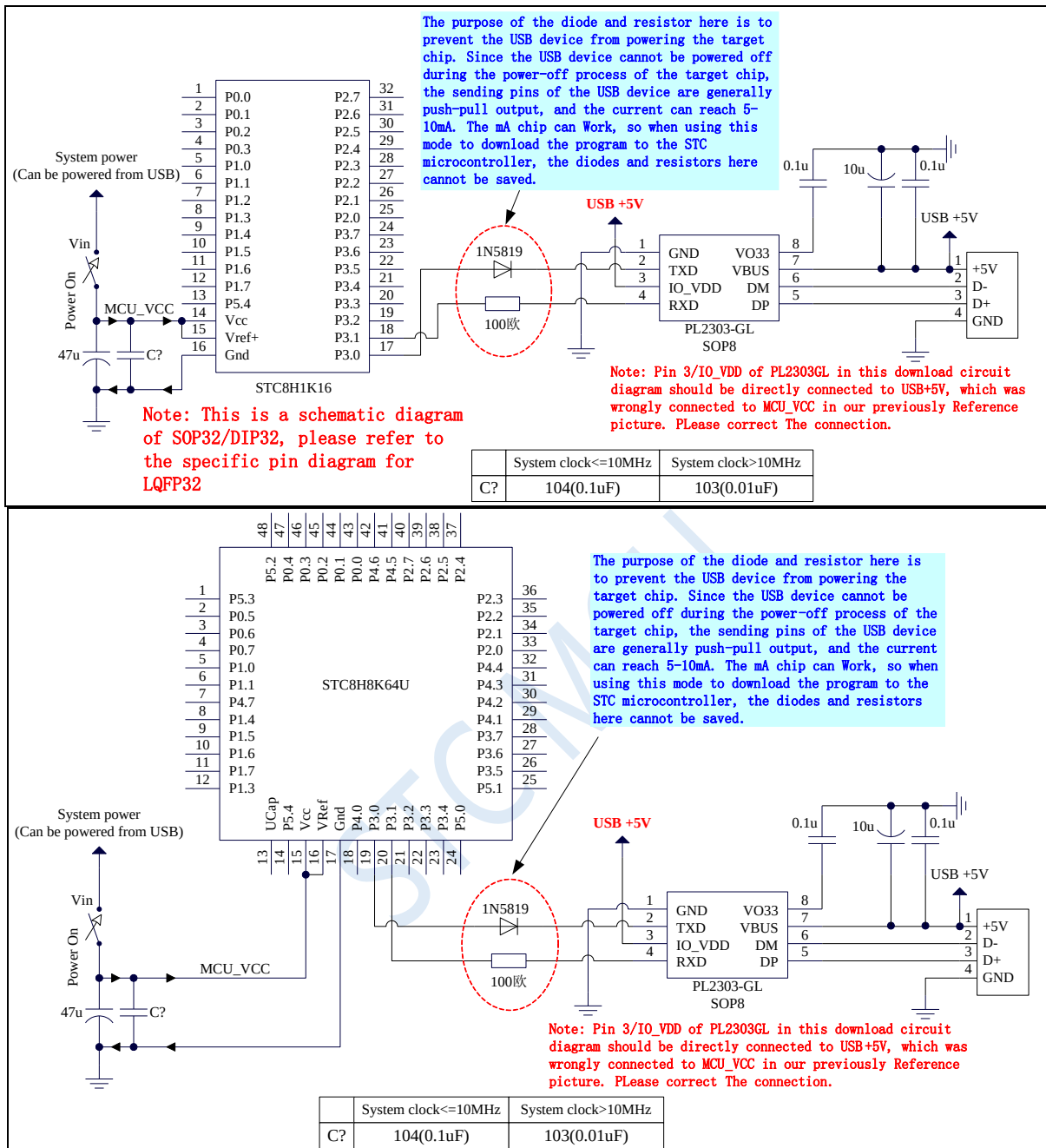
5.1.3 High-precision ADC reference circuit diagram of STC8H3K64S4 series, Emulation supported



ISP download steps:

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

5.1.4 Download using PL2303-GL, Emulation supported



ISP download steps:

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

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

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

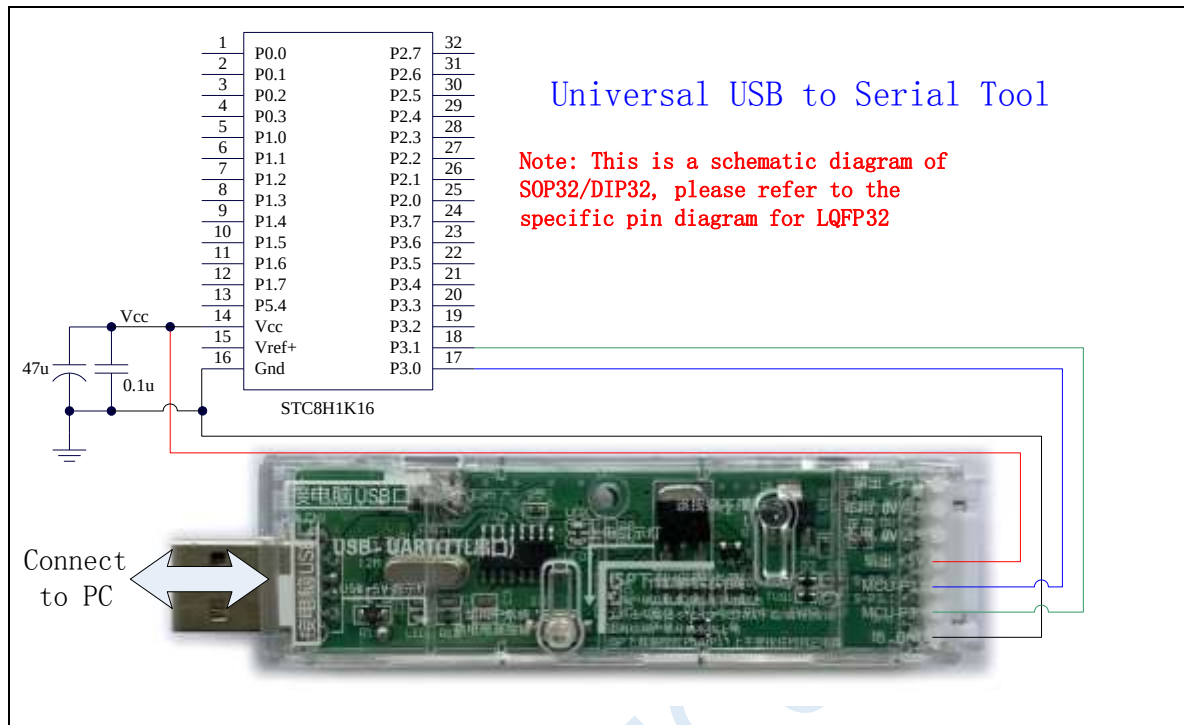
3. Click the "Download/Program" button in the STC-ISP download software

4. Power on the target chip

5. Start ISP download

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

5.1.5 Download using universal USB to Serial Tool, Emulation supported

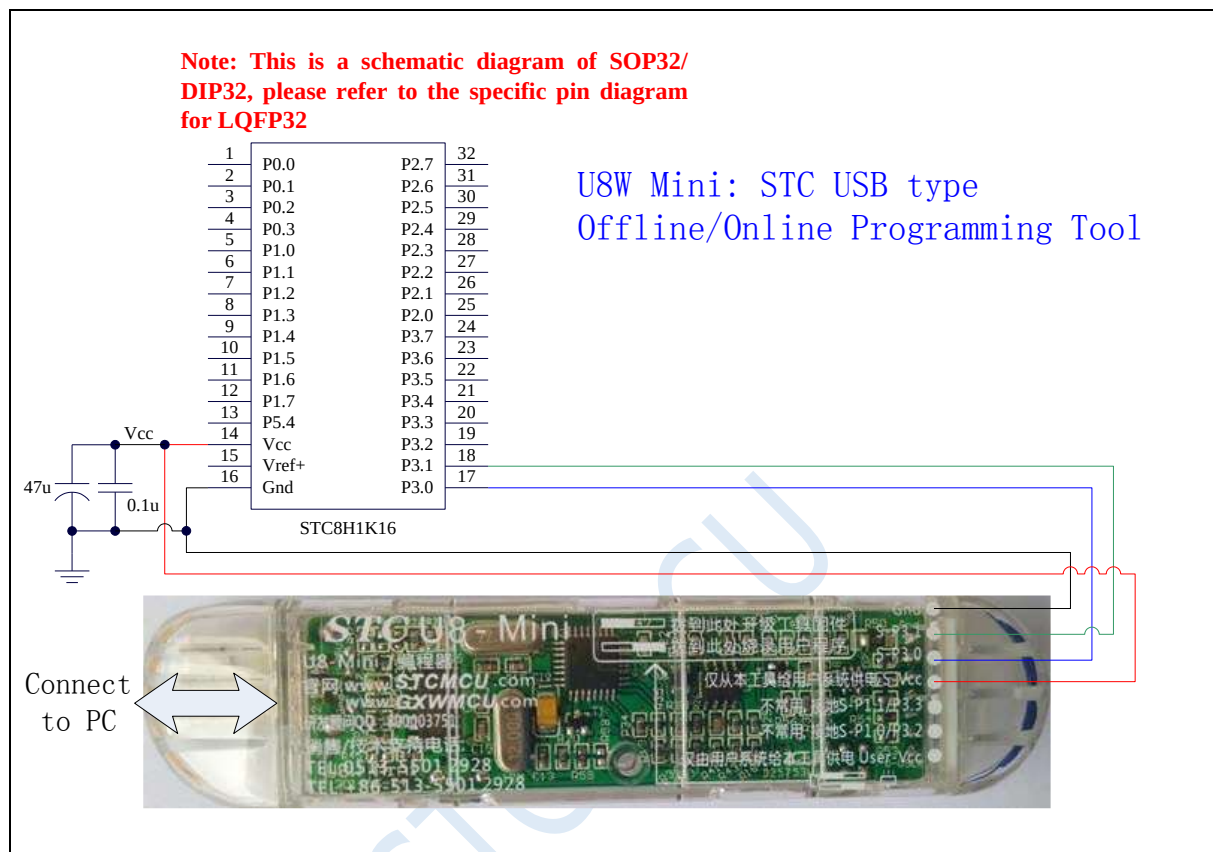


ISP download steps:

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

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

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



ISP download steps:

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

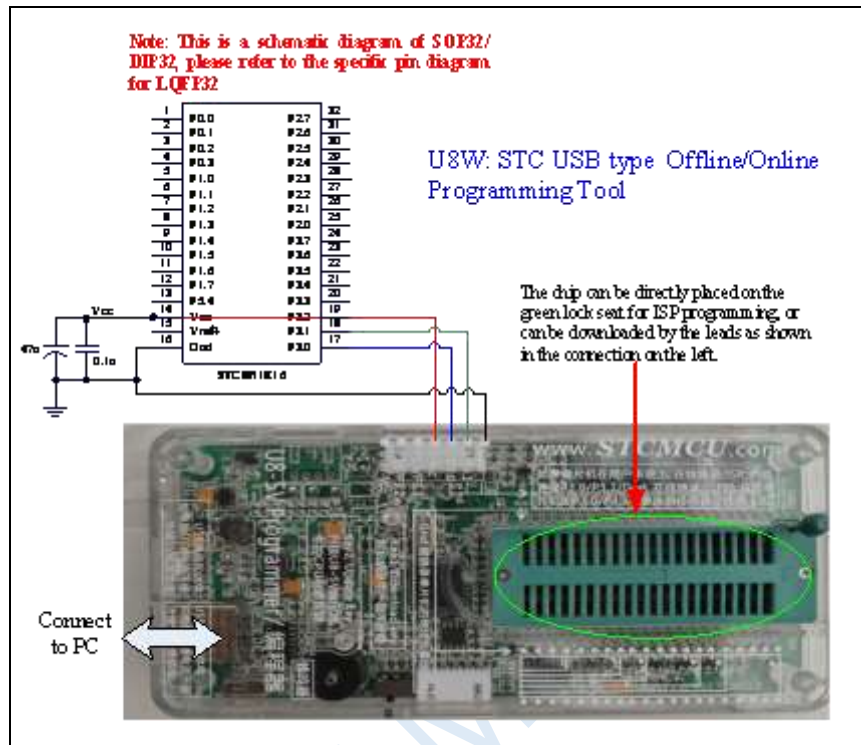
Note: If the U8-Mini is used to power the target system, the total current of the target system cannot be greater than 200mA, otherwise the download will fail.

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

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

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

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



ISP download steps(Connection method):

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

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

ISP download steps (on-board mode):

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

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

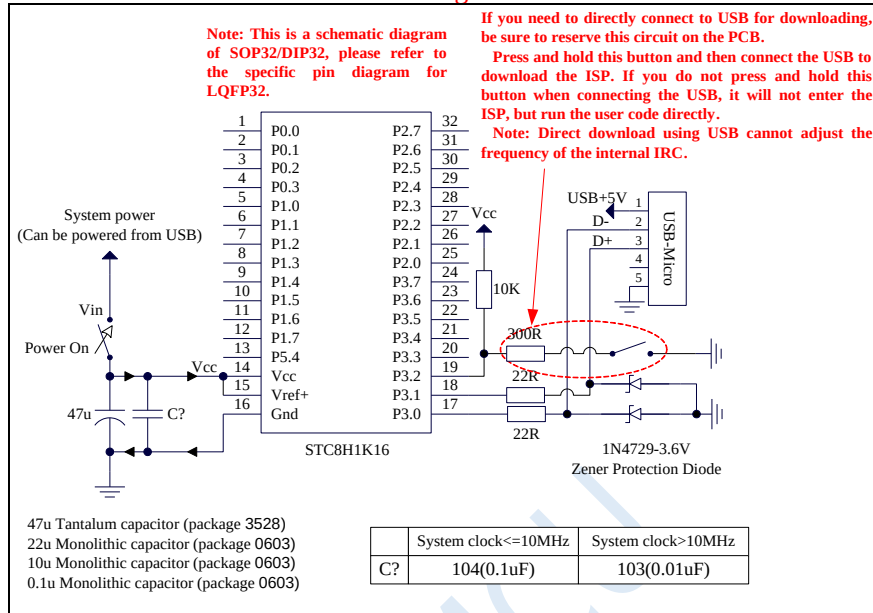
To emulate using the U8-Mini, you must set the U8-Mini to pass-through mode firstly. The method for U8W/U8W-Mini to realize the USB-to-serial pass-through mode is as follows:

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

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

Note 1: When using USB to download, you need to connect P3.2 to Gnd for normal download.

Note 2: If USB download is not required, P3.0/P3.1/P3.2 cannot be low at the same time when the chip is reset, otherwise the chip will always be in USB download mode without running user code.

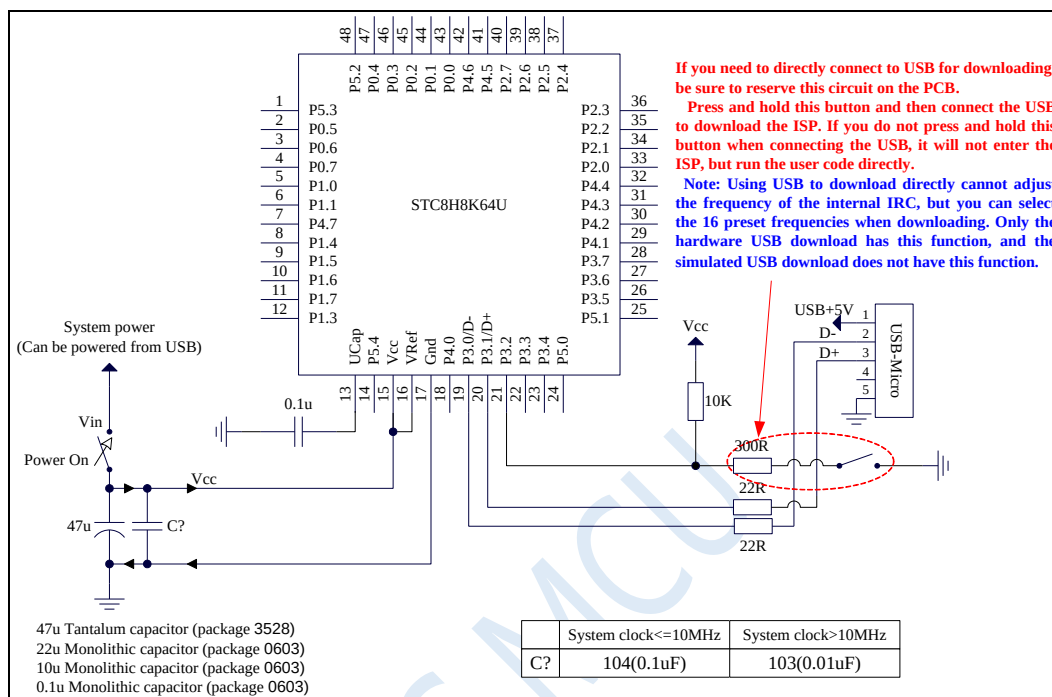


ISP download steps:

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

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

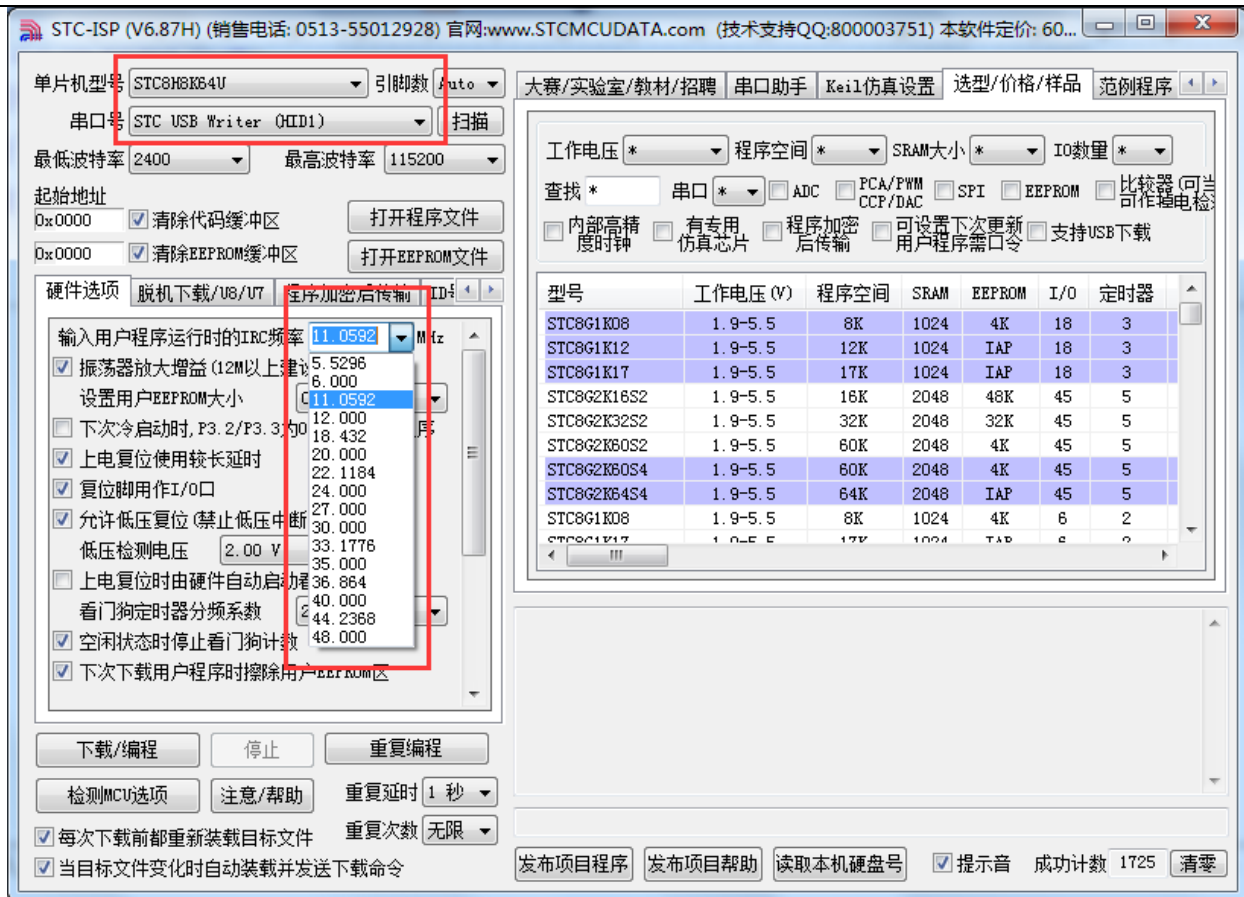
Note 3: If USB download is not required, P3.0/P3.1/P3.2 cannot be low at the same time when the chip is reset, otherwise the chip will always be in the USB download mode without running user code.



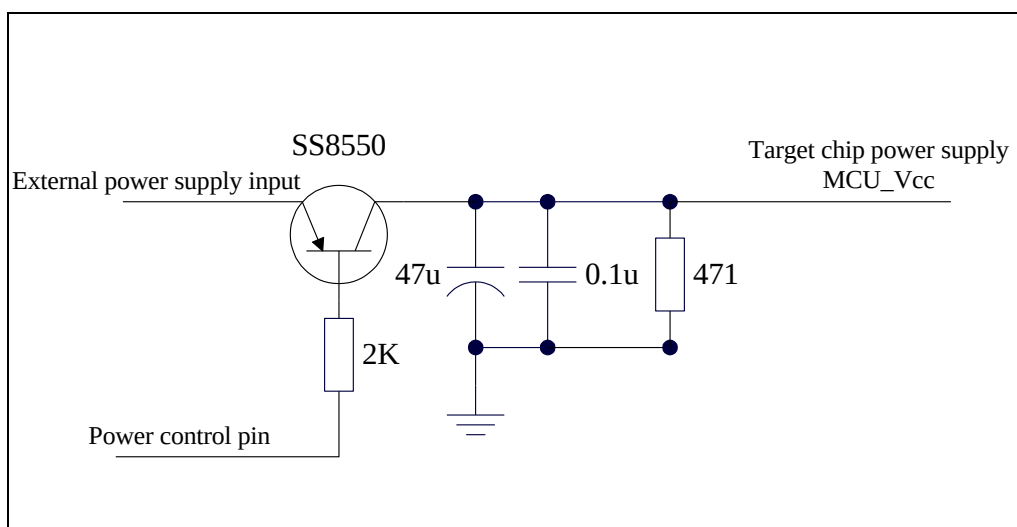
1. Power off the target chip.
2. Connect P3.0/P3.1 to the USB port according to the connection method shown in the figure.
3. Short-circuit P3.2 with GND.
4. Power on the target chip and wait for "STC USB Writer (HID1)" to be automatically recognized in the STC-ISP download software.
5. Click the "Download/Program"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

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

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



5.1.10 Microcontroller Power supply Control Reference Circuit

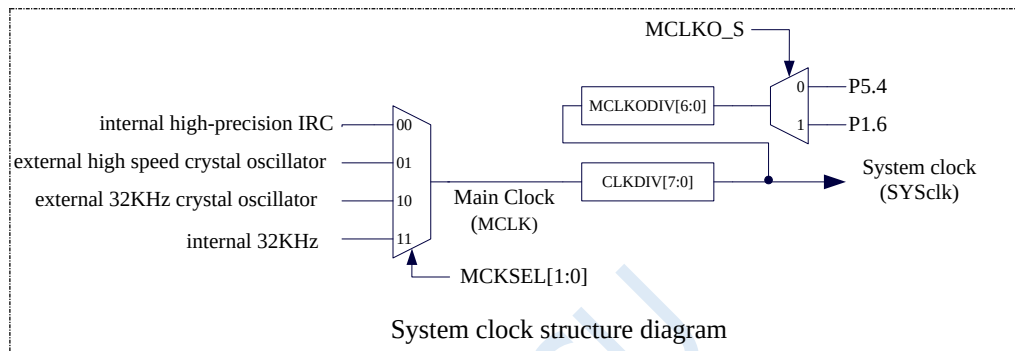


6. Clock, Reset and Power Management

6.1 System Clock Control

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

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



Related registers

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

6.1.1 System clock selection register (CKSEL)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
CKSEL	FE00H	-							

MCKSEL[1:0]: Main clock source selection

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

6.1.2 Clock Division register (CLKDIV)

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

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

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

...	...
255	MCLK/255

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

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

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

ENIRC: internal high speed high precision IRC enable bit

0: disable internal high-precision IRC

1: enable internal high-precision IRC

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

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

6.1.4 External Oscillator control register (XOSCCR)

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

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

STC8H other series

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

ENXOSC: external oscillator enable bit

0: disable external oscillator

1: enable external oscillator

XITYPE: external clock source type

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

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

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

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

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

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

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

GAIN: External crystal oscillator oscillator gain control bits

0: disable oscillator gain (low gain)

1: enable oscillator gain (high gain)

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

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

6.1.5 External 32KHz Oscillator control register (X32KCR)

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

ENX32K: External 32KHz Oscillator enable bit

0: disable external 32KHz Oscillator

1: enable external 32KHz Oscillator

GAIN32K: External 32KHz Oscillator gain control bit

0: disable external 32KHz Oscillator (low gain)

1: enable external 32KHz Oscillator (high gain)

X32KST: external 32KHz Oscillator frequency stability flag (read-only)

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

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

ENIRC32K: internal 32KHz low speed IRC enable bit

0: disable internal 32KHz low speed IRC

1: enable internal 32KHz low speed IRC

IRC32KST: internal 32KHz low speed IRC frequency stability flag (read-only)

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

6.1.7 Main clock output control register (MCLKOCR)

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

MCLKODIV[6:0]: Main clock output division factor

(Note: The clock source of main clock output is system clock divided by CLKDIV)

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

MCKO_S: Main clock output pin selection

0: Main clock output to P5.4

1: Main clock output to P1.6

6.2 STC8H series internal IRC frequency adjustment

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

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

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

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

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

Related registers

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

6.2.1 IRC band selection register (IRCBAND)

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

SEL: band selection

0: Select 20MHz band

1: Select 35MHz band

6.2.2 Internal IRC Frequency Adjustment Register (IRTRIM)

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

IRTRIM[7:0]: Internal high-precision IRC frequency adjustment register

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

6.2.3 Internal IRC frequency trim register (LIRTRIM)

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

LIRTRIM[1:0]: Internal high precision IRC frequency trim register

LIRTRIM can adjust the IRC frequency in 3 levels. The frequency range adjusted by the 3 levels is shown in the table below.

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

6.2.4 Clock Divide Register (CLKDIV)

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

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

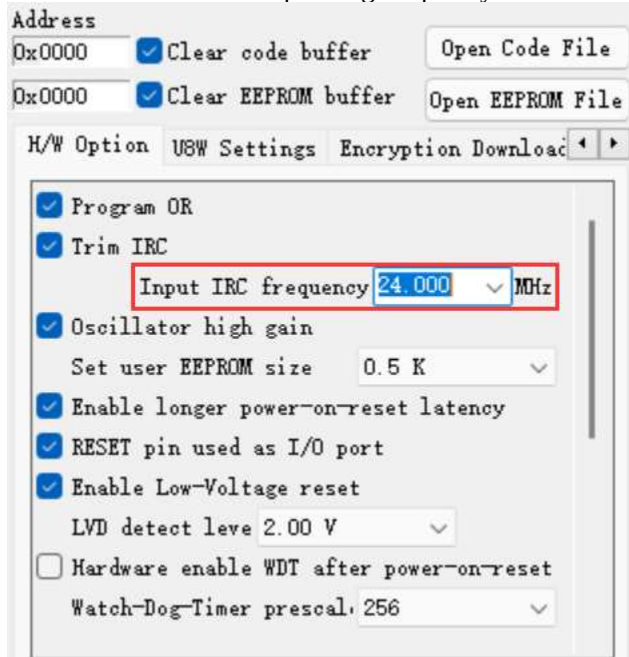
CLKDIV	System clock frequency
0	MCLK/1
1	MCLK/1
2	MCLK/2
3	MCLK/3
...	...
x	MCLK/x
...	...
255	MCLK/255

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

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

To get a frequency of 3MHz, you can use the method of 24MHz divided by 8.

Select the internal IRC operating frequency as 24MHz firstly when downloading the ISP, as shown in the figure below.



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

C language code

// Operating frequency for test is 24MHz

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

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

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

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

```
void main()
```

```
{
    P0M0 = 0x00;
    P0M1 = 0x00;
```

```

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

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

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

while (1);
}

```

Assembly code

; Operating frequency for test is 24MHz

P_SW2	DATA	0BAH
IRTRIM	DATA	09FH
CKSEL	EQU	0FE00H
CLKDIV	EQU	0FE01H
IRCCR	EQU	0FE02H
XOSCCR	EQU	0FE03H
IRC32KCR	EQU	0FE04H
P0M1	DATA	093H
P0M0	DATA	094H
P1M1	DATA	091H
P1M0	DATA	092H
P2M1	DATA	095H
P2M0	DATA	096H
P3M1	DATA	0B1H
P3M0	DATA	0B2H
P4M1	DATA	0B3H
P4M0	DATA	0B4H
P5M1	DATA	0C9H
P5M0	DATA	0CAH
	ORG	0000H
	LJMP	MAIN
	ORG	0100H
MAIN:	MOV	SP, #5FH
	MOV	P0M0, #00H
	MOV	P0M1, #00H
	MOV	P1M0, #00H
	MOV	P1M1, #00H
	MOV	P2M0, #00H
	MOV	P2M1, #00H

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

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

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

```
JMP    $
```

```
END
```

STC MCU

6.3 System reset

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

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

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

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

- Write SWRST bit in IAP_CONTR register to trigger reset

Related registers

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

6.3.1 Watch dog timer reset (WDT_CONTR)

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

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

WDT_CONTR (Watchdog control register)

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

WDT_FLAG : WDT reset flag.

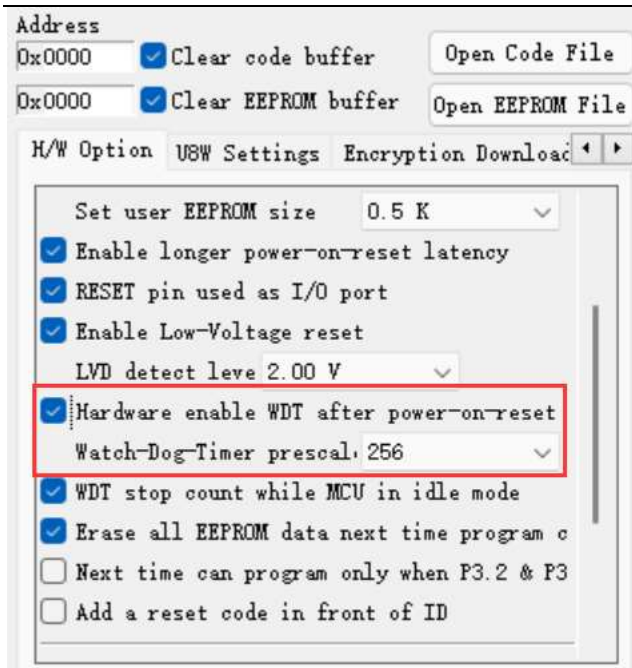
When WDT overflows, this bit is set by hardware automatically. This bit should be cleared by software.

EN_WDT: WDT enable bit.

0: No operation

1: WDT is started.

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



CLR_WDT: WDT clear bit.

0: No operation

1: WDT is cleared. This bit will be cleared by hardware automatically.

IDL_WDT: WDT control bit in IDLE mode.

0: WDT is disabled in IDLE mode.

1: WDT is enabled in IDLE mode, and the WDT will continue counting.

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

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

The WDT overflow time is determined by the following equation:

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

6.3.2 Software reset (IAP_CONTR)

IAP_CONTR (IAP Control Register)

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

SWBS: Software boot selection bit

0: The microcontroller executes the code from user program space (main flash memory) after the software reset. The data in the user data space remains unchanged.

1: The microcontroller executes the code from ISP space after the software reset. The data in the user data space is initialized.

SWRST: Software reset trigger bit.

0: No operation

1: Trigger software reset.

6.3.3 Low voltage reset (RSTCFG)

RSTCFG (Reset Configuration Register)

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

ENLVR: Low voltage detection reset enable bit

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

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

P54RST: RST pin function selection bit

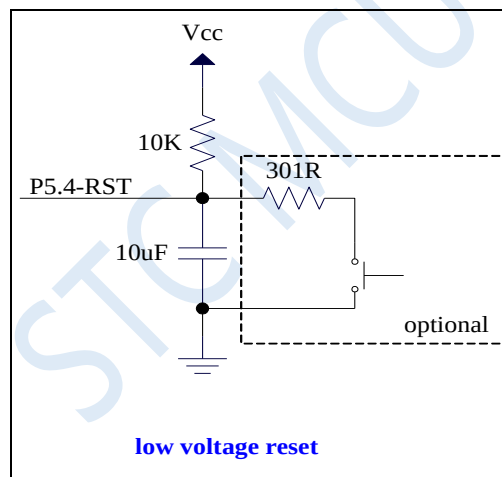
0: RST pin is used as common I/O (P5.4).

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

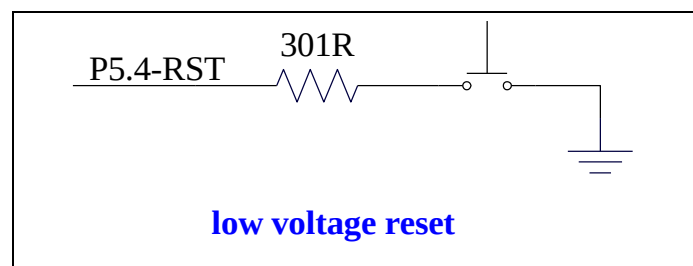
LVDS[1:0] : Low voltage detection threshold voltage setting bits

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

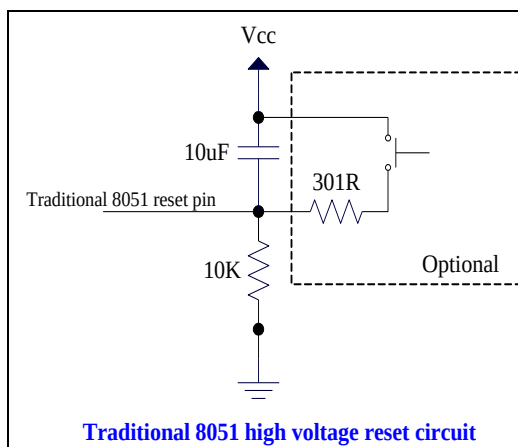
6.3.4 Low voltage power-on reset reference circuit (generally not required)



6.3.5 Low voltage button reset reference circuit



6.3.6 Traditional 8051 high voltage power-on reset reference circuit

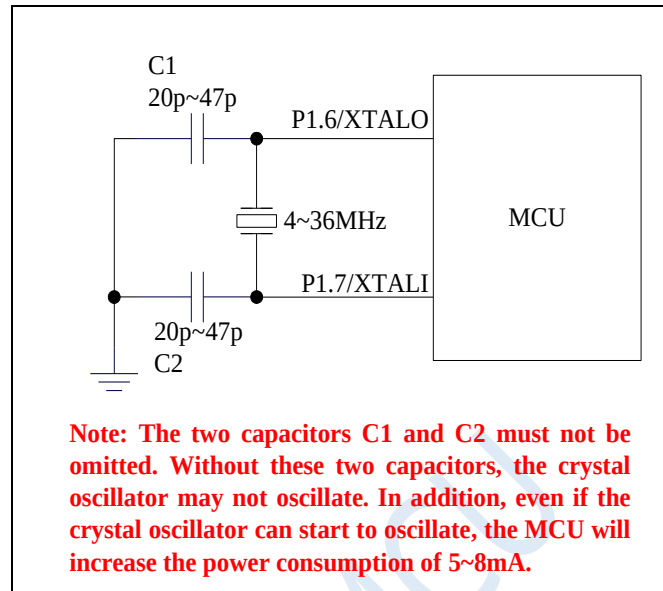


The picture above shows the high-level reset circuit of the traditional 8051. The reset of the STC8H is a low-level reset, which is different from the traditional reset circuit.

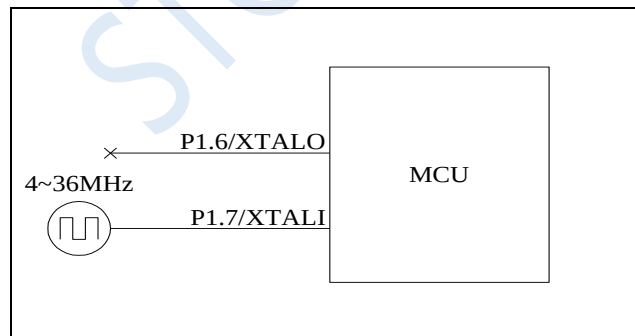
STC MCU

6.4 External crystal oscillator and external clock circuit

6.4.1 External crystal input circuit



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



6.5 Clock stop / Power Saving Mode and System Power Management

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

6.5.1 Power control register (PCON)

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

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

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

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

0: No operation.

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

IDL: IDLE mode control bit

0: No operation.

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

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

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

STC MCU

6.6 Power-down wake-up timer

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

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

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

WKTEN: wake-up timer enable bit

0: No operation

1: WKT is started.

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

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

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

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

Assuming Fwt=32KHz, there are:

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

6.7 Example Routines

6.7.1 System Clock Source Selection

C language code

// Operating frequency for test is 11.0592MHz

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

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

```
sfr      P_SW2      = 0xba;
```

```
sfr      P0M1       = 0x93;
```

```
sfr      P0M0       = 0x94;
```

```
sfr      P1M1       = 0x91;
```

```
sfr      P1M0       = 0x92;
```

```
sfr      P2M1       = 0x95;
```

```
sfr      P2M0       = 0x96;
```

```
sfr      P3M1       = 0xb1;
```

```
sfr      P3M0       = 0xb2;
```

```
sfr      P4M1       = 0xb3;
```

```
sfr      P4M0       = 0xb4;
```

```
sfr      P5M1       = 0xc9;
```

```
sfr      P5M0       = 0xca;
```

```
void main()
```

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

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

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

```

    CKSEL = 0x01;                //Select external crystal
    P_SW2 = 0x00;

*/

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

*/
    while (1);
}

```

Assembly code

; Operating frequency for test is 11.0592MHz

P_SW2	DATA	0BAH
CKSEL	EQU	0FE00H
CLKDIV	EQU	0FE01H
IRCCR	EQU	0FE02H
XOSCCR	EQU	0FE03H
IRC32KCR	EQU	0FE04H
P0M1	DATA	093H
P0M0	DATA	094H
P1M1	DATA	091H
P1M0	DATA	092H
P2M1	DATA	095H
P2M0	DATA	096H
P3M1	DATA	0B1H
P3M0	DATA	0B2H
P4M1	DATA	0B3H
P4M0	DATA	0B4H
P5M1	DATA	0C9H
P5M0	DATA	0CAH
	ORG	0000H
	LJMP	MAIN
	ORG	0100H
MAIN:	MOV	SP, #5FH
	MOV	P0M0, #00H
	MOV	P0M1, #00H
	MOV	P1M0, #00H
	MOV	P1M1, #00H
	MOV	P2M0, #00H
	MOV	P2M1, #00H
	MOV	P3M0, #00H
	MOV	P3M1, #00H
	MOV	P4M0, #00H
	MOV	P4M1, #00H
	MOV	P5M0, #00H
	MOV	P5M1, #00H
	MOV	P_SW2, #80H

```

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

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

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

JMP      $

END

```

6.7.2 Main Clock Output

C language code

// Operating frequency for test is 11.0592MHz

```

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

```

```

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

```

```

sfr      P_SW2      = 0xba;

```

```

sfr      P0M1       = 0x93;

```

```

sfr      P0M0       = 0x94;

```

```

sfr      P1M1       = 0x91;

```

```

sfr      P1M0       = 0x92;

```

```

sfr      P2M1       = 0x95;

```

```

sfr      P2M0       = 0x96;

```

```

sfr      P3M1       = 0xb1;

```

```

sfr      P3M0       = 0xb2;

```

```

sfr      P4M1       = 0xb3;

```



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

```
void main()
```

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

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

    while (1);
}
```

Assembly code

; Operating frequency for test is 11.0592MHz

```
P_SW2      DATA      0BAH

MCLKOCR     EQU        0FE05H

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

                ORG      0000H
                LJMP     MAIN

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

```

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

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

JMP      $

END

```

6.7.3 Application of Watch-dog Timer

C language code

// Operating frequency for test is 11.0592MHz

```

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

```

```

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

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

```

```
void main()
{

```

```

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

```

```

P5M0 = 0x00;
P5M1 = 0x00;

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

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

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

```

Assembly code

; Operating frequency for test is 11.0592MHz

```

WDT_CONTR DATA 0C1H

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

ORG 0000H
LJMP MAIN

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

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

```

```

;      MOV      WDT_CONTR,#27H      ; Watchdog enabled, overflow time is about 8s
;      CLR      P3.2                ; Test port

LOOP:
;      MOV      WDT_CONTR,#33H      ; Clear watchdog timer, otherwise system reset
;      MOV      WDT_CONTR,#34H      ; Clear watchdog timer, otherwise system reset
;      MOV      WDT_CONTR,#37H      ; Clear watchdog timer, otherwise system reset

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

      END

```

6.7.4 User Defined Downloading by Using Software Reset

C language code

// Operating frequency for test is 11.0592MHz

```

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

```

```

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

```

```

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

```

```
void main()
{

```

```

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

```

```

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

```

```

    while (1)

```

```

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

```

Assembly code

; Operating frequency for test is 11.0592MHz

```

IAP_CONTR    DATA    0C7H

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

                ORG     0000H
                LJMP    MAIN

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

    SETB       P3.2
    SETB       P3.3

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

    END

```

6.7.5 Low Voltage Detection

C language code

// Operating frequency for test is 11.0592MHz

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

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

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

void Lvd_Isr() interrupt 6

```
{
    PCON &= ~LVDF;          //Clear interrupt flag
    P32 = ~P32;             //Test port
}
```

void main()

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

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

    while (1);
}
```

}

Assembly code

; Operating frequency for test is 11.0592MHz

```

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

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

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

                ORG      0000H
                LJMP     MAIN
                ORG      0033H
                LJMP     LVDISR

LVDISR:       ORG      0100H

                ANL      PCON,#NOT LVDF    ;Clear interrupt flag
                CPL      P3.2              ;Test port
                RETI

MAIN:

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

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

```

END

6.7.6 Power Saving Mode

C language code

// Operating frequency for test is 11.0592MHz

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

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

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

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

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

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



```

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

while (1);
}

```

Assembly code

; Operating frequency for test is 11.0592MHz

```

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

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

        ORG      0000H
        LJMP     MAIN
        ORG      0003H
        LJMP     INT0ISR

INT0ISR: ORG      0100H

        CPL      P3.4      ;Test port
        RETI

MAIN:

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

        SETB     EX0      ;Enable INT0 interrupt to wake up MCU
        SETB     EA
        NOP
        NOP
;        MOV      PCON, #IDL      ;MCU enters IDLE mode
        MOV      PCON, #PD      ;MCU enters power down mode
        NOP
        NOP

```

```

NOP
NOP
CLR      P3.5      ;Test port
JMP      $

END

```

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

C language code

// Operating frequency for test is 11.0592MHz

```

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

```

```

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

```

```

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

```

```

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

```

void INT0_Isr() interrupt 0

```

{
    P10 = !P10;          //Test port
}

```

void INT1_Isr() interrupt 2

```

{
    P10 = !P10;          //Test port
}

```

void INT2_Isr() interrupt 10

```

{
    P10 = !P10;          //Test port
}

```

void INT3_Isr() interrupt 11

```

{
    P10 = !P10;          //Test port
}

```

```

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

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

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

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

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

    EA = 1;

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

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

```

Assembly code

; Operating frequency for test is 11.0592MHz

INTCLKO	DATA	8FH
EX2	EQU	10H
EX3	EQU	20H
EX4	EQU	40H
P0M1	DATA	093H
P0M0	DATA	094H
P1M1	DATA	091H

```

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

        ORG          0000H
        LJMP         MAIN

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

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

INT1ISR:  CPL          P1.0          ;Test port
        RETI

INT2ISR:  CPL          P1.0          ;Test port
        RETI

INT3ISR:  CPL          P1.0          ;Test port
        RETI

INT4ISR:  CPL          P1.0          ;Test port
        RETI

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

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

```

```

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

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

          SETB         EA

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

LOOP:     CPL          P1.1
          JMP          LOOP

          END

```

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

C language code

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

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

```

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

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

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

```

```

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

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

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

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

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

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

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

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

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

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

```

```

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

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

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

EA = 1;

PCON = 0x02;                                     //MCU enters power down mode
_nop_0; //Does not enter the interrupt service routine immediately after wake-up from power down mode
//Instead, wait for the timer to overflow before entering the interrupt service routine.
_nop_0;
_nop_0;
_nop_0;

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

```

Assembly code

; Operating frequency for test is 11.0592MHz

T2L	DATA	0D7H
T2H	DATA	0D6H
T3L	DATA	0D5H
T3H	DATA	0D4H
T4L	DATA	0D3H
T4H	DATA	0D2H
T4T3M	DATA	0D1H
AUXR	DATA	8EH
IE2	DATA	0AFH
ET2	EQU	04H
ET3	EQU	20H
ET4	EQU	40H
AUXINTIF	DATA	0EFH
T2IF	EQU	01H
T3IF	EQU	02H
T4IF	EQU	04H
P0M1	DATA	093H
P0M0	DATA	094H
P1M1	DATA	091H
P1M0	DATA	092H
P2M1	DATA	095H
P2M0	DATA	096H
P3M1	DATA	0B1H
P3M0	DATA	0B2H

```

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

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

TM0ISR:    ORG        0100H

CPL        P1.0          ;Test port
RETI

TM1ISR:    CPL        P1.0          ;Test port
RETI

TM2ISR:    CPL        P1.0          ;Test port
RETI

TM3ISR:    CPL        P1.0          ;Test port
RETI

TM4ISR:    CPL        P1.0          ;Test port
RETI

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

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

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

```



```

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

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

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

SETB   EA

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

END

```

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

C language code

// Operating frequency for test is 11.0592MHz

```

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

```

```

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

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

sbit   P11      = P1^1;

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

```

```

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

```

```

void UART1_Isr() interrupt 4
{
}

```

```

void UART2_Isr() interrupt 8
{
}

```

```

void UART3_Isr() interrupt 17
{
}

```

```

void UART4_Isr() interrupt 18
{
}

```

```

void main()
{

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

_nop_();
_nop_();

while (1)
{
    P11 = ~P11;
}

```

Assembly code

; Operating frequency for test is 11.0592MHz

IE2	DATA	0AFH
ES2	EQU	01H
ES3	EQU	08H
ES4	EQU	10H
P_SW1	DATA	0A2H
P_SW2	DATA	0BAH
P0M1	DATA	093H
P0M0	DATA	094H
P1M1	DATA	091H
P1M0	DATA	092H
P2M1	DATA	095H
P2M0	DATA	096H
P3M1	DATA	0B1H
P3M0	DATA	0B2H
P4M1	DATA	0B3H
P4M0	DATA	0B4H
P5M1	DATA	0C9H
P5M0	DATA	0CAH
	ORG	0000H
	LJMP	MAIN
	ORG	0023H
	LJMP	UART1ISR
	ORG	0043H
	LJMP	UART2ISR
	ORG	008BH
	LJMP	UART3ISR
	ORG	0093H
	LJMP	UART4ISR
	ORG	0100H
UART1ISR:		
	RETI	
UART2ISR:		
	RETI	
UART3ISR:		
	RETI	
UART4ISR:		
	RETI	
MAIN:		
	MOV	SP, #5FH
	MOV	P0M0, #00H
	MOV	P0M1, #00H
	MOV	P1M0, #00H

```

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

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

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

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

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

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

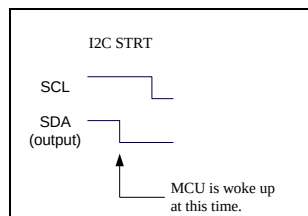
MOV      PCON, #02H        ;MCU enters power down mode
NOP      ;It will not enter the interrupt service routine after wake-up from power down mode.
NOP
NOP
NOP

LOOP:    CPL      P1.1
        JMP      LOOP

        END

```

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



C language code

// Operating frequency for test is 11.0592MHz

```

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

```

```

sfr      P_SW2      =    0xba;

```

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

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

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

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

```
void main()
{
```

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

```
    P_SW2 = 0x00;
//    P_SW2 = 0x10;
//    P_SW2 = 0x30;
    P_SW2 |= 0x80;
    I2CCFG = 0x80;
    I2CSLCR = 0x40;
    EA = 1;
```

```
//Wake up MCU on the falling edge of SDA/P1.4
//Wake up MCU on the falling edge of SDA_2/P2.4
//Wake up MCU on the falling edge of SDA_4/P3.3
```

```
//Enable slave mode of I2C module
//Enable start signal interrupt
```

```
    PCON = 0x02;
```

```
//MCU enters power-down mode, it will not enter the interrupt service
```

```
routine after power-down wake-up
```

```
    _nop_();
    _nop_();
    _nop_();
    _nop_();
```

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

Assembly code

; Operating frequency for test is 11.0592MHz

P_SW2 DATA 0BAH

I2CCFG XDATA 0FE80H

I2CSLCR XDATA 0FE83H

I2CSLST XDATA 0FE84H

P0M1 DATA 093H

P0M0 DATA 094H

P1M1 DATA 091H

P1M0 DATA 092H

P2M1 DATA 095H

P2M0 DATA 096H

P3M1 DATA 0B1H

P3M0 DATA 0B2H

P4M1 DATA 0B3H

P4M0 DATA 0B4H

P5M1 DATA 0C9H

P5M0 DATA 0CAH

ORG 0000H

LJMP MAIN

ORG 00C3H

LJMP I2CISR

I2CISR: ORG 0100H

PUSH ACC

PUSH DPH

PUSH DPL

ORL PSW2,#80H

MOV DPTR,#I2CSLST

MOVX A,@DPTR

ANL A,#NOT 40H

MOVX @DPTR,A

POP DPL

POP DPH

POP ACC

RETI

MAIN:

MOV SP, #5FH

MOV P0M0, #00H

MOV P0M1, #00H

MOV P1M0, #00H

MOV P1M1, #00H

MOV P2M0, #00H

MOV P2M1, #00H

MOV P3M0, #00H

MOV P3M1, #00H

MOV P4M0, #00H

MOV P4M1, #00H

MOV P5M0, #00H

MOV P5M1, #00H

MOV P_SW2,#00H

;Wake up MCU on the falling edge of SDA/P1.4

// **MOV P_SW2,#10H**

;Wake up MCU on the falling edge of SDA_2/P2.4

```
//      MOV      P_SW2,#30H      ;Wake up MCU on the falling edge of SDA_4/P3.3
      ORL      P_SW2,#80H
      MOV      DPTR,#I2CCFG
      MOV      A,#80H
      MOVX     @DPTR,A           ;Enable slave mode of I2C module
      MOV      DPTR,# I2CSLCR
      MOV      A,#40H           ;Enable start signal interrupt
      SETB     EA

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

LOOP:   CPL      P1.1
      JMP      LOOP

      END
```

6.7.11 Wake up MCU from Power Saving Mode using Power-down wake-up timer

C language code

// Operating frequency for test is 11.0592MHz

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

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

sbit     P11        = P1^1;
```

```
void main()
{
```

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

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

```
while (1)
{
    _nop_();
    _nop_();
    PCON = 0x02;    //MCU enters power-down mode
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    P11 = ~P11;
}
}
```

Assembly code

; Operating frequency for test is 11.0592MHz

```
WKTCL DATA 0AAH
WKTCH DATA 0ABH
```

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

```
ORG 0000H
LJMP MAIN
```

```
MAIN: ORG 0100H
```

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

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

LOOP:

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

END

6.7.12 Wake up MCU from Power Saving Mode using LVD interrupt (Recommended for use with power-down wake-up timer)

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

C language code

// Operating frequency for test is 11.0592MHz

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

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

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

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

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

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

```

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

PCON &= ~LVDF;           //Interrupt flag needs to be cleared after power-on
RSTCFG = LVD3V0;         //Set the LVD voltage to 3.0V
ELVD = 1;                //Enable LVD interrupt
EA = 1;

PCON = 0x02;             //MCU enters power-down mode
_nop_0; //Enter interrupt service routine immediately after wake-up from power mode
_nop_0;
_nop_0;
_nop_0;

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

```

Assembly code

; Operating frequency for test is 11.0592MHz

RSTCFG	DATA	0FFH	
ENLVR	EQU	40H	;RSTCFG.6
LVD2V0	EQU	00H	;LVD@2.0V
LVD2V4	EQU	01H	;LVD@2.4V
LVD2V7	EQU	02H	;LVD@2.7V
LVD3V0	EQU	03H	;LVD@3.0V
ELVD	BIT	IE.6	
LVDF	EQU	20H	;PCON.5
P0M1	DATA	093H	
P0M0	DATA	094H	
P1M1	DATA	091H	
P1M0	DATA	092H	
P2M1	DATA	095H	
P2M0	DATA	096H	
P3M1	DATA	0B1H	
P3M0	DATA	0B2H	
P4M1	DATA	0B3H	
P4M0	DATA	0B4H	
P5M1	DATA	0C9H	
P5M0	DATA	0CAH	
	ORG	0000H	
	LJMP	MAIN	
	ORG	0033H	
	LJMP	LVDISR	

```

LVDISR:      ORG          0100H

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

MAIN:

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

               ANL          PCON,#NOT LVDF      ;Interrupt flag needs to be cleared after power-on
               MOV          RSTCFG,# LVD3V0      ;Set the LVD voltage to 3.0V
               SETB          ELVD                ;Enable LVD interrupt
               SETB          EA

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

LOOP:

               CPL          P1.1
               JMP          LOOP

               END

```

6.7.13 Wake up MCU from Power Saving Mode using comparator interrupt (Recommended for use with power-down wake-up timer)

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

C language code

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

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

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

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

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

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

```
void CMP_Isr() interrupt 21
```

```
{
    CMPCR1 &= ~0x40;           //Clear interrupt flag
    P10 = !P10;                //Test port
}
```

```
void main()
```

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

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

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

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

}

Assembly code

; Operating frequency for test is 11.0592MHz

```

CMPCR1    DATA    0E6H
CMPCR2    DATA    0E7H

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

        ORG        0000H
        LJMP       MAIN
        ORG        00ABH
        LJMP       CMPISR

CMPISR:   ORG        0100H

        ANL        CMPCR1,#NOT 40H    ;Clear interrupt flag
        CPL        P1.0                ;Test port
        RETI

MAIN:

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

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

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

```

NOP

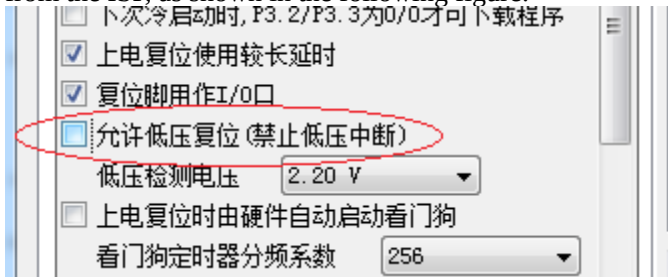
LOOP:

CPL P1.1
JMP LOOP

END

6.7.14 Detect the Operating Voltage (Battery Voltage) using LVD

If you need to use LVD to detect the battery voltage, you need to remove the low-voltage reset function when downloading from the ISP, as shown in the following figure.



C language code

// Operating frequency for test is 11.0592MHz

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

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

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

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

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

```
void delay()
{
    int i;

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

```
    _nop_0;
    _nop_0;
    _nop_0;
    _nop_0;
}
}

void main()
{
    unsigned char power;

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

    PCON &= ~LVDF;
    RSTCFG = LVD3V0;

    while (1)
    {
        power = 0x0f;

        RSTCFG = LVD3V0;
        delay();
        PCON &= ~LVDF;
        delay();
        if (PCON & LVDF)
        {
            power >>= 1;
            RSTCFG = LVD2V7;
            delay();
            PCON &= ~LVDF;
            delay();
            if (PCON & LVDF)
            {
                power >>= 1;
                RSTCFG = LVD2V4;
                delay();
                PCON &= ~LVDF;
                delay();
                if (PCON & LVDF)
                {
                    power >>= 1;
                    RSTCFG = LVD2V2;
                    delay();
                    PCON &= ~LVDF;
                    delay();
                    if (PCON & LVDF)
                    {
                        power >>= 1;
                    }
                }
            }
        }
    }
}
```

```

    }
}
}
RSTCFG = LVD3V0;
P2 = ~power;    // P2.3 ~ P2.0 are used to display battery level
}
}

```

Assembly code

; Operating frequency for test is 11.0592MHz

```

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

```

```

LVDF      EQU      20H           ;PCON.5

```

```

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

```

```

ORG       0000H
JMP       MAIN

```

```

MAIN:     ORG       0100H

```

```

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

```

```

ANL       PCON, #NOT LVDF
MOV       RSTCFG, #LVD3V0

```

LOOP:

```

MOV       B, #0FH

```

```

MOV       RSTCFG, #LVD3V0
CALL      DELAY

```



```

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

```

```

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

```

```

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

```

```

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

```

SKIP:

```

MOV      A,B
CPL      A
MOV      P2,A
JMP      LOOP

```

; P2.3 ~ P2.0 are used to display battery level

DELAY:

```

MOV      R0,#100

```

NEXT:

```

NOP
NOP
NOP
NOP
DJNZ     R0,NEXT

```

RET

END

STC MCU

7 Memory

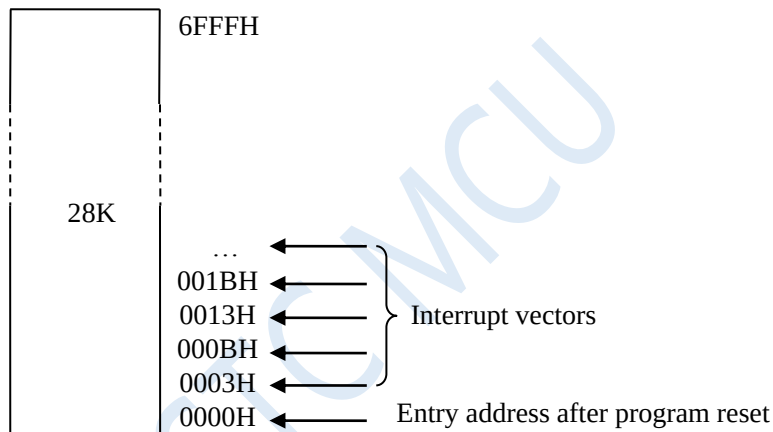
The STC8H series of microcontrollers have separate address spaces for Program Memory and Data Memory. Since no bus is provided for accessing external program memory, all program memory for all microcontrollers is on-chip Flash memory. The microcontrollers can not access external program memory.

Large-capacity data memory is integrated in STC8H series of microcontrollers. The data memory inside the STC8H series of microcontrollers is physically and logically separated into two address spaces: 256 bytes of internal RAM and internal extended RAM. The addresses of the high 128 bytes of internal RAM and special function registers (SFRs) overlap. They can be accessed through different addressing modes in actual use.

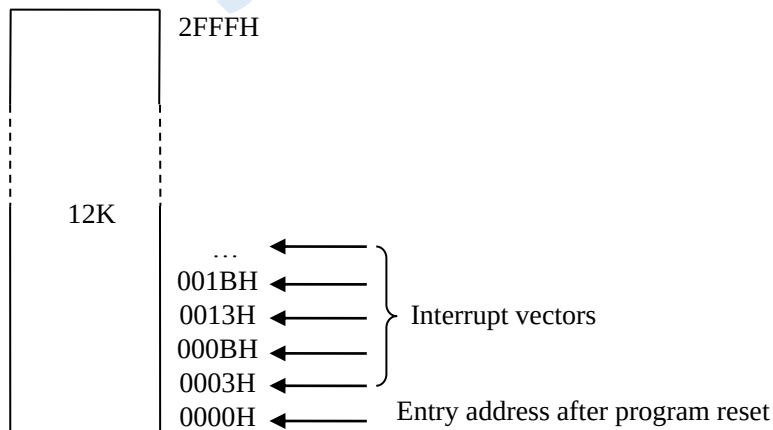
7.1 Program Memory

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

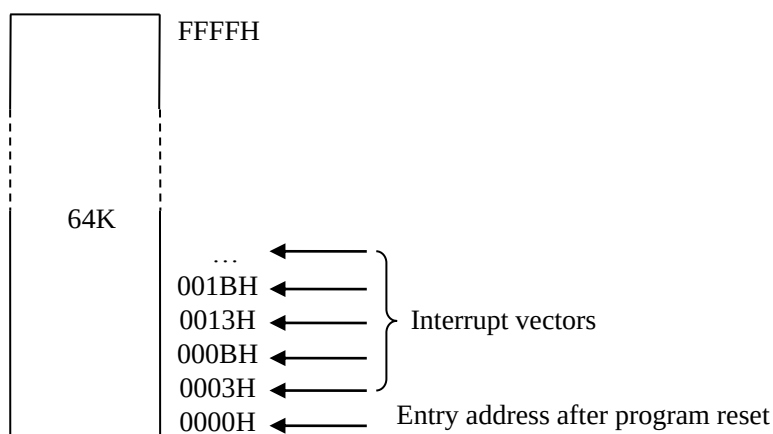
28K bytes of Flash program memory is integrated in STC8H1K28 family of microcontrollers.



12K bytes of Flash program memory is integrated in STC8H1K08 family of microcontrollers.



64K bytes of Flash program memory is integrated in STC8H3K64S4 family, STC8H3K64S2 family, STC8H8K64U family, STC8H2K64T family of microcontrollers.



After the microcontroller resets, the content of the Program Counter (PC) is 0000H, and the CPU begins to execute program from 0000H of Program Memory. The entry addresses of interrupt service routines, which are also called interrupt vectors, are also located in the program memory. Each interrupt has a fixed entry address in Program Memory. When an interrupt occurs and gets response, the microcontroller will automatically jump to its corresponding interrupt entry address to execute the service routine. The entry address of the interrupt service routine for the external interrupt 0 (INT0) is 0003H, the entry address for the timer / counter 0 (TIMER0) interrupt service routine is 000BH, and the entry address for the interrupt service routine for the external interrupt 1 (INT1) is 0013H. The counter/counter 1 (TIMER1) interrupt service routine's entry address is 001BH. More interrupt service routine entry address (interrupt vector), please refer to interrupt chapter.

The interval of adjacent interrupt entry addresses is only 8 bytes, which is not enough to save the complete interrupt service routine in general, so an unconditional jump instruction is stored in the the interrupt vector to jump to the space where the real interrupt service routine is stored, then execute interrupt service routine.

All STC8H series of microcontrollers integrate Flash data memory (EEPROM). The EEPROM is read or written in byte, and is erased in page of 512bytes. It can be repeatedly programmed and erased over 100,000 times, which improves the flexibility and convenience of use.

7.2 Data Memory

The RAM integrated in the STC8H series of microcontrollers can be used to store intermediate results and process data during program execution.

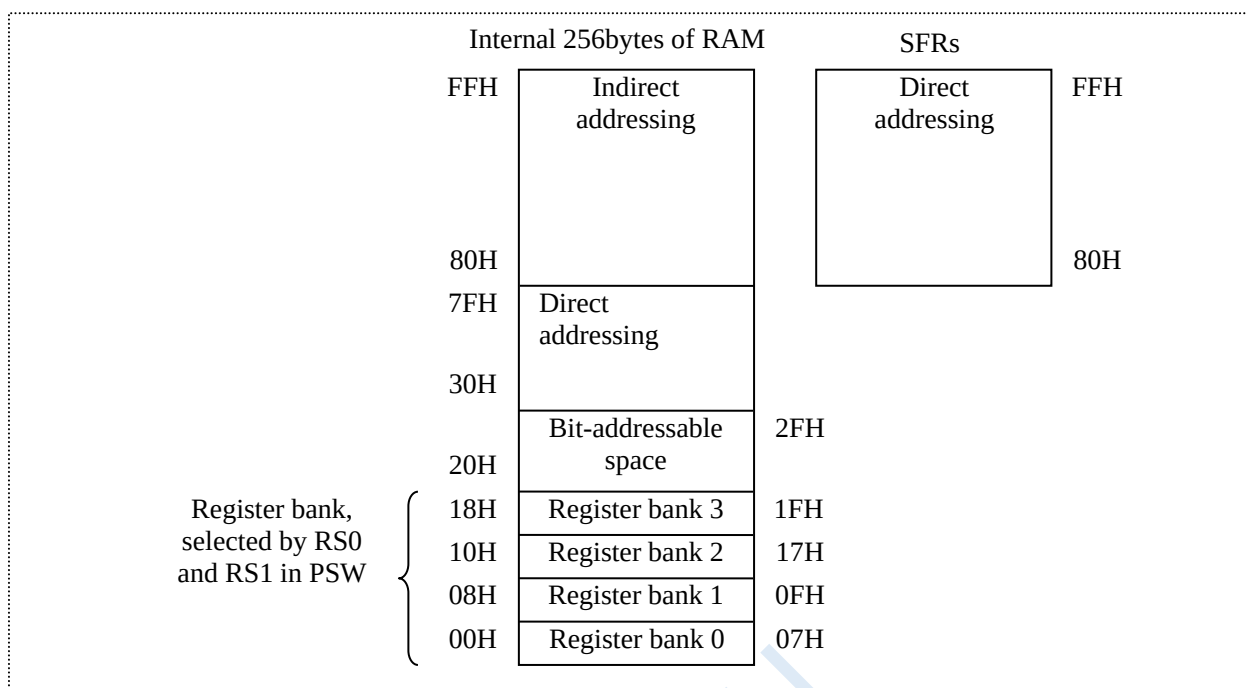
Family of microcontrollers	Internal direct access RAM (DATA)	Internal indirect access RAM (IDATA)	On-chip extended RAM (XDATA)
STC8H1K08 family	128 bytes	128 bytes	1024 bytes
STC8H1K28 family	128 bytes	128 bytes	1024 bytes
STC8H3K64S4 family	128 bytes	128 bytes	3072 bytes
STC8H3K64S2 family	128 bytes	128 bytes	3072 bytes
STC8H8K64U family	128 bytes	128 bytes	8192 bytes
STC8H2K64T family	128 bytes	128 bytes	2048 bytes
STC8H4K64TLR family	128 bytes	128 bytes	4096 bytes
STC8H4K64TLCD family	128 bytes	128 bytes	4096 bytes
STC8H4K64LCD family	128 bytes	128 bytes	4096 bytes

7.2.1 Internal RAM

A total of 256 bytes of internal RAM can be divided into two parts: Lower 128 bytes of RAM and Upper 128 bytes of RAM. The Lower 128 bytes of data memory are compatible with the traditional 8051 microcontroller, which can be access by either Direct addressing or Indirect addressing. The Upper 128 bytes of RAM (upper 128 bytes of RAM is extended in 8052) and special function registers, SFRs in short, occupy the same block of addresses, 80H to FFH, but they are physically separate entities and are accessed using different addressing modes. Upper 128 bytes of RAM can only be

accessed by Indirect addressing, SFRs area can only be accessed by Direct addressing.

Internal RAM is mapped in the following figure.



The Lower 128 bytes of RAM are also called as general purpose RAM space. The general purpose RAM space can be divided into working register banks space, bit addressable space, user RAM space and stack space. Total of 32 bytes of working register bank space, 00H to 1FH, are divided into 4 groups. Each group is called a register bank, which contains 8 8-bit working registers. All the numbers in different register bank are R0 through R7, but they belong to different Physical space. By using the working register registers, the operation speed can be increased. R0 ~ R7 are commonly used registers. Four bank sare provided because one bank is often not enough. The combination of RS1 and RS0 in the PSW register determines the working register bank currently used, see the introduction of PSW register below.

7.2.2 PSW (program status word register)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
PSW	D0H	CY	AC	F0	RS1	RS0	OV	F1	P

CY: Carry/borrow flag bit.

AC: Auxiliary carry/borrow flag bit.

F0: User flag bit 0.

RS1, RS0: Working register select bit

RS1	RS0	working register bank (R0~R7)
0	0	Bank 0 (00H~07H)
0	1	Bank 1 (08H~0FH)
1	0	Bank 2 (10H~17H)
1	1	Bank 3 (18H~1FH)

OV: Overflow flag bit.

F1: User flag bit 1.

P: Parity flag bit.

There are 16 bytes in the bit addressable space, 20H to 2FH. They can either be accessed by byte like ordinary RAM or be individually accessed by any one bit in the byte unit. There are totally 128 bits in this space, whose logic bit addresses are 00H to 7FH. From the appearance, bit addresses and the internal Lower 128 bytes RAM addresses are the same as 00H to 7FH, but in fact, they are essentially different: bit address points to a bit, and the byte address points to a byte unit. They are distinguished by using different instructions in programs.

The addresses 30H to FFH in the internal RAM are the user RAM and stack space. An 8-bit stack pointer, SP in short is used to point to the stack space. On reset, SP is 07H, which is R7 of register bank 0. Therefore, the initial value of SP should be set in the user initialization codes. You would better to set the initial value of SP at 80H or higher.

SP is an 8-bit dedicated register. It indicates the top of the stack in the internal RAM. On reset, SP is initialized to 07H, which makes the stack space begin from 08H. The addresses 08H to 1FH are also the addresses of working register bank 1 through 3. It is better to change the SP value to a value of 80H or more if these spaces are used in user application. The stack of STC8H series of microcontrollers grows upward, which means that when a datum is pushed into the stack, the content of SP will increase.

7.2.3 On-chip extended RAM

In addition to 256 bytes of internal RAM, on-chip extended RAM is integrated in STC8H series of microcontrollers. The method of accessing the on-chip extended RAM is the same as that of the traditional 8051 MCU accessing the external extended RAM. However, the P0 port (data bus and low-order address bus), P2 port (high-order address bus), RD, WR and ALE are not affected.

In assembly language, the on-chip extended RAM is accessed through the MOVX instruction,

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

In C language, xdata / pdata can be used to declare the storage type, such as,
 unsigned char xdata i;
 unsigned int pdata j;

Note that pdata is the lower 256 bytes of xdata. After declaring a variable as the pdata type in C program, the compiler will automatically allocate the variables in the 0000H to 00FFH of XDATA and use MOVX @ Ri, A and MOVX A @ Ri to access.

The control bit EXTRAM located in AUXR register is used to control the access of on-chip extended RAM can be used or not.

7.2.4 Auxiliary register (AUXR)

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

EXTRAM: on-chip extended RAM access control bit

0: On-chip extended RAM is enabled or can be accessed.

1: On-chip extended RAM is disabled.

7.2.5 External extended RAM, XRAM, XDATA

The STC8H series of packages with a pin count of 40 and above have the ability to expand 64KB of external data memory. During access to the external data memory, the WR/RD/ALE signal must be valid. A new special function register BUS_SPEED for controlling the speed of the external 64K byte data bus has been added to the STC8H series of single-chip microcomputers. The description is as follows:

7.2.6 Bus speed control register (BUS_SPEED)

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

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

00: P4.4 is RD, P4.2 is WR

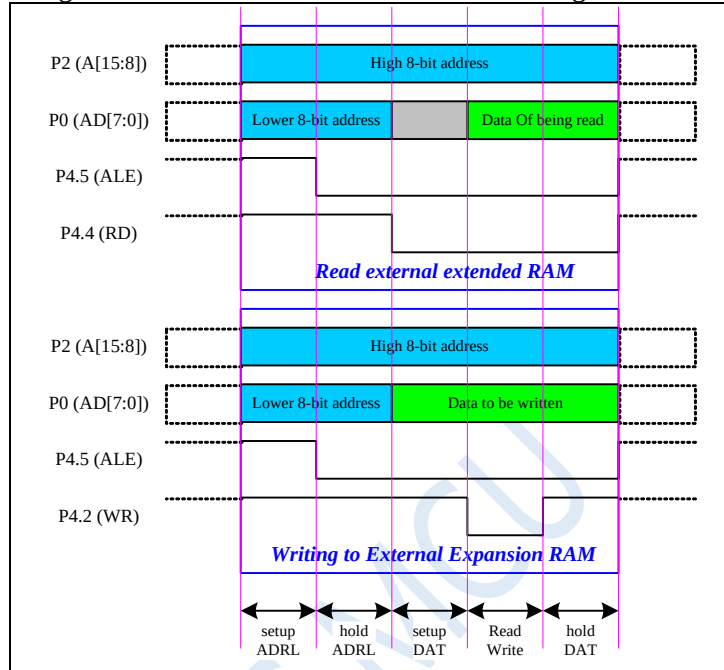
x1: Keep

SPEED[2:0]: Bus read and write speed control (preparation time and hold time of control signal and data signal when

reading and writing data)

instruction	Number of clocks	
	Access internal expansion RAM	Access external expansion RAM
MOVX A,@Ri	3	3+5* (SPEED+1)
MOVX @Ri,A	3	3+5* (SPEED+1)
MOVX A,@DPTR	2	3+5* (SPEED+1)
MOVX @DPTR,A	2	3+5* (SPEED+1)

The timing of reading and writing external extended RAM is shown in the figure below:



7.2.7 Bit Addressable Data Memory in 8051

Bit addressable data memory integrated in 8051 single-chip includes two parts: the address range of the first part is 00H ~ 7FH, and the address range of the second part is 80H ~ FFH. The 00H ~ 7FH bit addressing area is a mapping of the 16 bytes of the data area 20H ~ 2FH, and the 80H ~ FFh bit addressing area is the 16 special function registers whose addresses are divisible by 8. (Including 80H, 88H, 90H, 98H, A0H, A8H, B0H, B8H, C0H, C8H, D0H, D8H, E0H, E8H, F0H, F8H).

Address of Data Memory	Address of Bit-addressable							
	B7	B6	B5	B4	B3	B2	B1	B0
F8H (P7)	FFH F8H 7	FEH F8H 6	FDH F8H 5	FCH F8H 4	FBH F8H 3	FAH F8H 2	F9H F8H 1	F8H F8H 0
F0H (B)	F7H F0H 7	F6H F0H 6	F5H F0H 5	F4H F0H 4	F3H F0H 3	F2H F0H 2	F1H F0H 1	F0H F0H 0
E8H (P6)	EFH E8H 7	EEH E8H 6	EDH E8H 5	ECH E8H 4	EBH E8H 3	EAH E8H 2	E9H E8H 1	E8H E8H 0
E0H (ACC)	E7H E0H 7	E6H E0H 6	E5H E0H 5	E4H E0H 4	E3H E0H 3	E2H E0H 2	E1H E0H 1	E0H E0H 0
D8H (CCON)	DFH D8H 7	DEH D8H 6	DDH D8H 5	DCH D8H 4	DBH D8H 3	DAH D8H 2	D9H D8H 1	D8H D8H 0
D0H (PSW)	D7H D0H 7	D6H D0H 6	D5H D0H 5	D4H D0H 4	D3H D0H 3	D2H D0H 2	D1H D0H 1	D0H D0H 0
C8H (P5)	CFH C8H 7	CEH C8H 6	CDH C8H 5	CCH C8H 4	CBH C8H 3	CAH C8H 2	C9H C8H 1	C8H C8H 0
C0H (P4)	C7H C0H 7	C6H C0H 6	C5H C0H 5	C4H C0H 4	C3H C0H 3	C2H C0H 2	C1H C0H 1	C0H C0H 0
B8H (IP)	BFH B8H 7	BEH B8H 6	BDH B8H 5	BCH B8H 4	BBH B8H 3	BAH B8H 2	B9H B8H 1	B8H B8H 0

B0H (P3)	B7H B0H_7	B6H B0H_6	B5H B0H_5	B4H B0H_4	B3H B0H_3	B2H B0H_2	B1H B0H_1	B0H B0H_0
A8H (IE)	AFH A8H_7	AEH A8H_6	ADH A8H_5	ACH A8H_4	ABH A8H_3	AAH A8H_2	A9H A8H_1	A8H A8H_0
A0H (P2)	A7H A0H_7	A6H A0H_6	A5H A0H_5	A4H A0H_4	A3H A0H_3	A2H A0H_2	A1H A0H_1	A0H A0H_0
98H (SCON)	9FH 98H_7	9EH 98H_6	9DH 98H_5	9CH 98H_4	9BH 98H_3	9AH 98H_2	99H 98H_1	98H 98H_0
90H (P1)	97H 90H_7	96H 90H_6	95H 90H_5	94H 90H_4	93H 90H_3	92H 90H_2	91H 90H_1	90H 90H_0
88H (TCON)	8FH 88H_7	8EH 88H_6	8DH 88H_5	8CH 88H_4	8BH 88H_3	8AH 88H_2	89H 88H_1	88H 88H_0
80H (P0)	87H 80H_7	86H 80H_6	85H 80H_5	84H 80H_4	83H 80H_3	82H 80H_2	81H 80H_1	80H 80H_0
2FH	7FH 2FH_7	7EH 2FH_6	7DH 2FH_5	7CH 2FH_4	7BH 2FH_3	7AH 2FH_2	79H 2FH_1	78H 2FH_0
2EH	77H 2EH_7	76H 2EH_6	75H 2EH_5	74H 2EH_4	73H 2EH_3	72H 2EH_2	71H 2EH_1	70H 2EH_0
2DH	6FH 2DH_7	6EH 2DH_6	6DH 2DH_5	6CH 2DH_4	6BH 2DH_3	6AH 2DH_2	69H 2DH_1	68H 2DH_0
2CH	67H 2CH_7	66H 2CH_6	65H 2CH_5	64H 2CH_4	63H 2CH_3	62H 2CH_2	61H 2CH_1	60H 2CH_0
2BH	5FH 2BH_7	5EH 2BH_6	5DH 2BH_5	5CH 2BH_4	5BH 2BH_3	5AH 2BH_2	59H 2BH_1	58H 2BH_0
2AH	57H 2AH_7	56H 2AH_6	55H 2AH_5	54H 2AH_4	53H 2AH_3	52H 2AH_2	51H 2AH_1	50H 2AH_0
29H	4FH 29H_7	4EH 29H_6	4DH 29H_5	4CH 29H_4	4BH 29H_3	4AH 29H_2	49H 29H_1	48H 29H_0
28H	47H 28H_7	46H 28H_6	45H 28H_5	44H 28H_4	43H 28H_3	42H 28H_2	41H 28H_1	40H 28H_0
27H	3FH 27H_7	3EH 27H_6	3DH 27H_5	3CH 27H_4	3BH 27H_3	3AH 27H_2	39H 27H_1	38H 27H_0
26H	37H 26H_7	36H 26H_6	35H 26H_5	34H 26H_4	33H 26H_3	32H 26H_2	31H 26H_1	30H 26H_0
25H	2FH 25H_7	2EH 25H_6	2DH 25H_5	2CH 25H_4	2BH 25H_3	2AH 25H_2	29H 25H_1	28H 25H_0
24H	27H 24H_7	26H 24H_6	25H 24H_5	24H 24H_4	23H 24H_3	22H 24H_2	21H 24H_1	20H 24H_0
23H	1FH 23H_7	1EH 23H_6	1DH 23H_5	1CH 23H_4	1BH 23H_3	1AH 23H_2	19H 23H_1	18H 23H_0
22H	17H 22H_7	16H 22H_6	15H 22H_5	14H 22H_4	13H 22H_3	12H 22H_2	11H 22H_1	10H 22H_0
21H	0FH 21H_7	0EH 21H_6	0DH 21H_5	0CH 21H_4	0BH 21H_3	0AH 21H_2	09H 21H_1	08H 21H_0
20H	07H 20H_7	06H 20H_6	05H 20H_5	04H 20H_4	03H 20H_3	02H 20H_2	01H 20H_1	00H 20H_0

7.3 Special parameters of memory

The data memory and program memory of the STC8H series of microcontrollers store some special parameters related to the chip, including the global unique ID, the frequency of the 32K power-down wake-up timer, the internal reference voltage value, and the IRC parameters.

Note: For chips with firmware version 7.4.4 and later, all special parameters can only be read from the read-only special function register (CHIPID).

The addresses of these parameters in the Flash program memory (ROM) are as follows:

Parameters	Addresses			Parameter Description
	STC8H1K08	STC8H1K12	STC8H1K17	
global unique ID	1FF9H~1FFFH	2FF9H~2FFFH	43F9H~43FFFH	7 bytes
internal 1.19V reference voltage	1FF7H~1FF8H	2FF7H~2FF8H	43F7H~43F8H	mV (high byte first)
frequency of the 32K power-down wake-up timer	1FF5H~1FF6H	2FF5H~2FF6H	43F5H~43F6H	Hz (high byte first)
parameters of 22.1184MHz IRC(20M band)	1FF4H	2FF4H	43F4H	-
parameters of 24MHz IRC(20M band)	1FF3H	2FF3H	43F3H	-
parameters of 20MHz IRC(20M band)	1FF2H	2FF2H	43F2H	Valid for the version 7.3.12U or later firmware.
parameters of 27MHz IRC(35M band)	1FF1H	2FF1H	43F1H	
parameters of 30MHz IRC(35M band)	1FF0H	2FF0H	43F0H	
parameters of 33.1776MHz IRC(35M band)	1FEFH	2FEFH	43EFH	
parameters of 35MHz IRC(35M band)	1FEEH	2FEEH	43EEH	
parameters of 36.684MHz IRC(35M band)	1FEDH	2FEDH	43EDH	
Reserved	1FECH	2FECH	43ECH	
Reserved	1FEBH	2FEBH	43EBH	
parameters of VRTRIM in 20M band	1FEAH	2FEAH	43EAH	
parameters of VRTRIM in 35M band	1FE9H	2FE9H	43E9H	

Parameters	Addresses				Parameter Description
	STC8H1K16	STC8H1K24	STC8H1K28	STC8H1K33	
global unique ID	3FF9H~3FFFH	5FF9H~5FFFH	6FF9H~6FFFH	83F9H~83FFFH	7 bytes
internal 1.19V reference voltage	3FF7H~3FF8H	5FF7H~5FF8H	6FF7H~6FF8H	83F7H~83F8H	mV (high byte first)
frequency of the 32K power-down wake-up timer	3FF5H~3FF6H	5FF5H~5FF6H	6FF5H~6FF6H	83F5H~83F6H	Hz (high byte first)
parameters of 22.1184MHz IRC(20M band)	3FF4H	5FF4H	6FF4H	83F4H	-
parameters of 24MHz IRC(20M band)	3FF3H	5FF3H	6FF3H	83F3H	-
parameters of 20MHz IRC(20M band)	3FF2H	5FF2H	6FF2H	83F2H	Valid for the version 7.3.12U or later firmware.
parameters of 27MHz IRC(35M band)	3FF1H	5FF1H	6FF1H	83F1H	
parameters of 30MHz IRC(35M band)	3FF0H	5FF0H	6FF0H	83F0H	
parameters of 33.1776MHz IRC(35M band)	3FEFH	5FEFH	6FEFH	83EFH	
parameters of 35MHz IRC(35M band)	3FEEH	5FEEH	6FEEH	83EEH	
parameters of 36.684MHz IRC(35M band)	3FEDH	5FEDH	6FEDH	83EDH	
Reserved	3FECH	5FECH	6FECH	83ECH	
Reserved	3FEBH	5FEBH	6FEBH	83EBH	
parameters of VRTRIM in 20M band	3FEAH	5FEAH	6FEAH	83EAH	
parameters of VRTRIM in 35M band	3FE9H	5FE9H	6FE9H	83E9H	

Parameters	Addresses				Parameter Description
	STC8H3K32S4 STC8H3K32S2 STC8H2K32T STC8H4K32TLR STC8H4K32TLCD STC8H4K32LCD	STC8H3K48S4 STC8H3K48S2 STC8H2K48T STC8H4K48TLR STC8H4K48TLCD STC8H4K48LCD	STC8H3K60S4 STC8H3K60S2 STC8H2K60T STC8H4K60TLR STC8H4K60TLCD STC8H4K60LCD	STC8H3K64S4 STC8H3K64S2 STC8H2K64T STC8H4K64TLR STC8H4K64TLCD STC8H4K64LCD	
global unique ID	7FF9H~7FFFH	BFF9H~BFFFH	EFF9H~EFFFH	FDF9H~FDFFH	7 bytes
internal 1.19V reference voltage	7FF7H~7FF8H	BFF7H~BFF8H	EFF7H~EFF8H	FDF7H~FDF8H	mV (high byte first)
frequency of the 32K power-down wake-up timer	7FF5H~7FF6H	BFF5H~BFF6H	EFF5H~EFF6H	FDF5H~FDF6H	Hz (high byte first)
parameters of 22.1184MHz IRC(20M band)	7FF4H	BFF4H	EFF4H	FDF4H	-
parameters of 24MHz IRC(20M band)	7FF3H	BFF3H	EFF3H	FDF3H	-
parameters of 20MHz IRC(20M band)	7FF2H	BFF2H	EFF2H	FDF2H	Valid for the version 7.3.12U or later firmware.
parameters of 27MHz IRC(35M band)	7FF1H	BFF1H	EFF1H	FDF1H	
parameters of 30MHz IRC(35M band)	7FF0H	BFF0H	EFF0H	FDF0H	
parameters of 33.1776MHz IRC(35M band)	7FEFH	BFEFH	EF EFH	FDEFH	
parameters of 35MHz IRC(35M band)	7FEEH	BFEEH	EFEEH	FDEEH	
parameters of 36.684MHz IRC(35M band)	7FEDH	BFEDH	EFEDH	FDEDH	
Reserved	7FECH	BFECH	EFECH	FDECH	
Reserved	7FEBH	BFEBH	EFEBH	FDEBH	

STC8H Series Manual

parameters of VRTRIM in 20M band	7FEAH	BFEAH	EFEAH	FDEAH	
parameters of VRTRIM in 35M band	7FE9H	BFE9H	EFE9H	FDE9H	

Parameters	Addresses				Parameter Description
	STC8H8K32U	STC8H8K48U	STC8H8K60U	STC8H8K64U	
global unique ID	7FF9H~7FFFH	BFF9H~BFFFH	EFF9H~EFFFH	FDF9H~FDFFH	7 bytes
internal 1.19V reference voltage	7FF7H~7FF8H	BFF7H~BFF8H	EFF7H~EFF8H	FDF7H~FDF8H	mV (high byte first)
frequency of the 32K power-down wake-up timer	7FF5H~7FF6H	BFF5H~BFF6H	EFF5H~EFF6H	FDF5H~FDF6H	Hz (high byte first)
parameters of 22.1184MHz IRC(20M band)	7FF4H	BFF4H	EFF4H	FDF4H	-
parameters of 24MHz IRC(20M band)	7FF3H	BFF3H	EFF3H	FDF3H	-
parameters of 27MHz IRC(35M band)	7FF2H	BFF2H	EFF2H	FDF2H	Valid for the version 7.3.12U or later firmware.
parameters of 30MHz IRC(40M band)	7FF1H	BFF1H	EFF1H	FDF1H	
parameters of 33.1776MHz IRC(40M band)	7FF0H	BFF0H	EFF0H	FDF0H	
parameters of 35MHz IRC(40M band)	7FEFH	BFEFH	EFEFH	FDEFH	
parameters of 36.684MHz IRC(40M band)	7FEEH	BFEEH	EFEEH	FDEEH	
parameters of 40MHz IRC(40M band)	7FEDH	BFEDH	EFEDH	FDEDH	
parameters of 44.2368MHz IRC(40M band)	7FECH	BFECH	EFECH	FDECH	
parameters of 48MHz IRC(40M band)	7FEBH	BFEBH	EFEBH	FDEBH	
parameters of VRTRIM in 20M band	7FEAH	BFEAH	EFEAH	FDEAH	
parameters of VRTRIM in 40M band	7FE9H	BFE9H	EFE9H	FDE9H	

Note: The extended IRC parameter list of STC8H8K64U series is not compatible with other series.

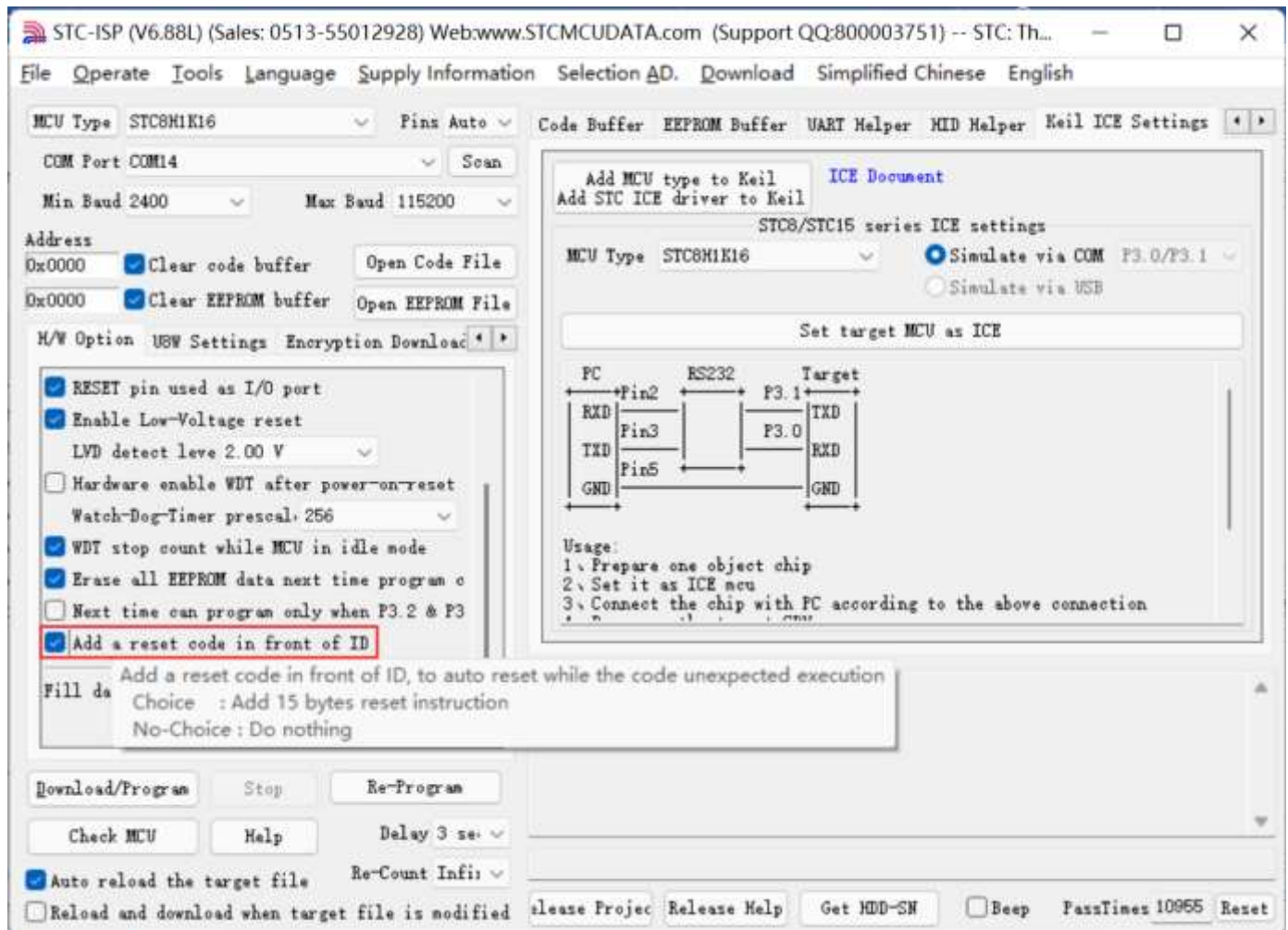
Note: For chips with firmware version 7.4.4 and later, all special parameters can only be read from the read-only special function register (CHIPID).

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

Parameters	Addresses	Parameter Description
global unique ID	idata: 0F1H~0F7H	7 bytes
internal reference voltage	idata: 0EFH~0F0H	mV (high byte first)
frequency of the 32K power-down wake-up timer	idata: 0F8H~0F9H	Hz (high byte first)
parameters of 22.1184MHz IRC	idata: 0FAH	-
parameters of 24MHz IRC	idata: 0FBH	-

Special Note

1. Since the parameters in RAM may be modified, it is generally not recommended to be used. Especially, it is strongly recommended to read ID data in FLASH program memory (ROM) when you use ID for encryption.
2. Due to the size of EEPROM in STC8H1K28, STC8H1K12, STC8H3K64S4, STC8H3K64S2, STC8H8K64U, STC8H2K64T, STC8H4K64TLR, STC8H4K64TLCD, STC8H4K64LCD can be set by user, important parameters stored in the FLASH program memory (ROM) space may be erased or modified when the FLASH is used as EEPROM. So this issue needs to be considered when using these microcontrollers for ID number encryption.
3. By default, only the global unique ID is in the Flash program memory (ROM), and the internal reference 1.19 voltage value, the frequency of the 32K power-down wake-up timer, and the IRC parameters are not available. You need to select the option in the following figure when downloading by ISP, and then the options shown are available.



7.4 Unique ID number and important parameter (CHIPID) stored in read-only special function register

Note: For chips with firmware version 7.4.4 and later, all special parameters can only be read from the read-only special function register (CHIPID).

Product line	CHIPID
STC8H1K08 family	
STC8H1K28 family	
STC8H3K64S4 family	
STC8H3K64S2 family	
STC8H8K64U family version A	
STC8H8K64U family version B	●
STC8H2K64T family	
STC8H4K64TLR family	●
STC8H4K64TLCD family	●
STC8H4K64LCD family	●

Some STC8H series microcontrollers have built-in 32-byte read-only special function register CHIPID. The content in CHIPID can only be read by the user program and cannot be modified. Using the data in CHIPID to encrypt user programs is the optimal solution officially recommended by STC.

Related registers

Symbol	Description	Address	Bit address and symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
CHIPID00	hardware digital ID00	FDE0H	Globally Unique ID Number (0th byte)								nnnn,nnnn
CHIPID01	hardware digital ID01	FDE1H	Globally Unique ID Number (1st byte)								nnnn,nnnn
CHIPID02	hardware digital ID02	FDE2H	Globally Unique ID Number (2nd byte)								nnnn,nnnn
CHIPID03	hardware digital ID03	FDE3H	Globally Unique ID Number (3rd byte)								nnnn,nnnn
CHIPID04	hardware digital ID04	FDE4H	Globally Unique ID Number (4th byte)								nnnn,nnnn
CHIPID05	hardware digital ID05	FDE5H	Globally Unique ID Number (5th byte)								nnnn,nnnn
CHIPID06	hardware digital ID06	FDE6H	Globally Unique ID Number (6th byte)								nnnn,nnnn
CHIPID07	hardware digital ID07	FDE7H	Internal 1.19V reference signal source (high byte)								nnnn,nnnn
CHIPID08	hardware digital ID08	FDE8H	Internal 1.19V reference signal source (low byte)								nnnn,nnnn
CHIPID09	hardware digital ID09	FDE9H	32K Power-down wake-up timer frequency(high byte)								nnnn,nnnn
CHIPID10	hardware digital ID10	FDEAH	32K Power-down wake-up timer frequency (low byte)								nnnn,nnnn
CHIPID11	hardware digital ID11	FDEBH	USB series (STC8H8K64U family)				Other series				nnnn,nnnn
			IRC parameter of 22.1184MHz (27M band)				IRC parameter of 22.1184MHz (27M band)				
CHIPID12	hardware digital ID12	FDECH	IRC parameter of 24MHz (27M band)				IRC parameter of 24MHz (27M band)				nnnn,nnnn
CHIPID13	hardware digital ID13	FDEDH	IRC parameter of 27MHz (27M band)				IRC parameter of 20MHz (27M band)				nnnn,nnnn
CHIPID14	hardware digital ID14	FDEEH	IRC parameter of 30MHz (27M band)				IRC parameter of 27MHz (27M band)				nnnn,nnnn
CHIPID15	hardware digital ID15	FDEFH	IRC parameter of 33.1776MHz (27M band)				IRC parameter of 30MHz (27M band)				nnnn,nnnn
CHIPID16	hardware digital ID16	FDF0H	IRC parameter of 35MHz (44M band)				IRC parameter of 33.1776MHz (27M band)				nnnn,nnnn
CHIPID17	hardware digital ID17	FDF1H	IRC parameter of 36.864MHz (44M band)				IRC parameter of 35MHz (44M band)				nnnn,nnnn
CHIPID18	hardware digital ID18	FDF2H	IRC parameter of 40MHz (44M band)				IRC parameter of 36.864MHz(44M band)				nnnn,nnnn
CHIPID19	hardware digital ID19	FDF3H	IRC parameter of 44.2368MHz (44M band)				IRC parameter of 40MHz(44M band)				nnnn,nnnn
CHIPID20	hardware digital ID20	FDF4H	IRC parameter of 48MHz (44M band)				IRC parameter of 45MHz(44M band)				nnnn,nnnn
CHIPID21	hardware digital ID21	FDF5H	VRTRIM parameter of 6M band								nnnn,nnnn
CHIPID22	hardware digital ID22	FDF6H	VRTRIM parameter of 10M band								nnnn,nnnn
CHIPID23	hardware digital ID23	FDF7H	VRTRIM parameter of 27M band								nnnn,nnnn
CHIPID24	hardware digital ID24	FDF8H	VRTRIM parameter of 44M band								nnnn,nnnn
CHIPID25	hardware digital ID25	FDF9H	00H								nnnn,nnnn
CHIPID26	hardware digital ID26	FDEAH	User program space end address (high byte)								nnnn,nnnn
CHIPID27	hardware digital ID27	FDFBH	Chip test time (year)								nnnn,nnnn
CHIPID28	hardware digital ID28	FDFCH	Chip test time (month)								nnnn,nnnn
CHIPID29	hardware digital ID29	FDFDH	Chip test time (day)								nnnn,nnnn
CHIPID30	hardware digital ID30	FDFEH	Chip package form number								nnnn,nnnn
CHIPID31	hardware digital ID31	FDFFH	5AH								nnnn,nnnn

7.4.1 Interpretation of Global Unique ID Number in CHIP

Symbol	Description	Address	Bit address and symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
CHIPID00	hardware digital ID00	FDE0H	Globally Unique ID Number (0th byte)								nnnn,nnnn
CHIPID01	hardware digital ID01	FDE1H	Globally Unique ID Number (1st byte)								nnnn,nnnn
CHIPID02	hardware digital ID02	FDE2H	Globally Unique ID Number (2nd byte)								nnnn,nnnn
CHIPID03	hardware digital ID03	FDE3H	Globally Unique ID Number (3rd byte)								nnnn,nnnn
CHIPID04	hardware digital ID04	FDE4H	Globally Unique ID Number (4th byte)								nnnn,nnnn
CHIPID05	hardware digital ID05	FDE5H	Globally Unique ID Number (5th byte)								nnnn,nnnn
CHIPID06	hardware digital ID06	FDE6H	Globally Unique ID Number (6th byte)								nnnn,nnnn

[CHIPID0, CHIPID1]: 16-bit MCU ID, which is used to distinguish different MCU models (higher bit first).

The commonly used MCU IDs of STC8 series are shown in the following table:

STC8H4K16TLC (F811)	STC8H4K16LCD (F821)	STC8H4K16TLR (F7C9)	STC8H3K16S4 (F741)	STC8H3K16S2 (F749)	STC8H8K16U (F781)
STC8H4K32TLC (F812)	STC8H4K32LCD (F822)	STC8H4K32TLR (F7CA)	STC8H3K32S4 (F742)	STC8H3K32S2 (F74A)	STC8H8K32U (F782)
STC8H4K48TLC (F815)	STC8H4K48LCD (F825)	STC8H4K48TLR (F7CD)	STC8H3K48S4 (F745)	STC8H3K48S2 (F74D)	STC8H8K48U (F785)
STC8H4K60TLC (F813)	STC8H4K60LCD (F823)	STC8H4K60TLR (F7CB)	STC8H3K60S4 (F743)	STC8H3K60S2 (F74B)	STC8H8K60U (F783)
STC8H4K64TLC (F814)	STC8H4K64LCD (F824)	STC8H4K64TLR (F7CC)	STC8H3K64S4 (F744)	STC8H3K64S2 (F74C)	STC8H8K64U (F784)
STC8H1K06 (F733)	STC8H1K16 (F721)	STC8G2K16S4 (F761)	STC8G2K16S2 (F769)	STC8G1K06A-8PIN (F793)	STC8G1K06-8PIN (F7A3)
STC8H1K08 (F734)	STC8H1K20 (F722)	STC8G2K32S4 (F762)	STC8G2K32S2 (F76A)	STC8G1K08A-8PIN (F794)	STC8G1K08-8PIN (F7A4)
STC8H1K10 (F735)	STC8H1K24 (F723)	STC8G2K48S4 (F765)	STC8G2K48S2 (F76D)	STC8G1K10A-8PIN (F795)	STC8G1K10-8PIN (F7A5)
STC8H1K12 (F736)	STC8H1K28 (F724)	STC8G2K60S4 (F763)	STC8G2K60S2 (F76B)	STC8G1K12A-8PIN (F796)	STC8G1K12-8PIN (F7A6)
STC8H1K17 (F737)	STC8H1K33 (F725)	STC8G2K64S4 (F764)	STC8G2K64S2 (F76C)	STC8G1K17A-8PIN (F797)	STC8G1K17-8PIN (F7A7)
STC8G1K06 (F753)	STC8C2K16S4 (F7D1)	STC8C2K16S2 (F7D9)	STC8A8K16D4 (F7F1)		
STC8G1K08 (F754)	STC8C2K32S4 (F7D2)	STC8C2K32S2 (F7DA)	STC8A8K32D4 (F7F2)		
STC8G1K10 (F755)	STC8C2K48S4 (F7D5)	STC8C2K48S2 (F7DB)	STC8A8K48D4 (F7F5)		
STC8G1K12 (F756)	STC8C2K60S4 (F7D3)	STC8C2K60S2 (F7DB)	STC8A8K60D4 (F7F3)		
STC8G1K17 (F757)	STC8C2K64S4 (F7D4)	STC8C2K64S2 (F7DC)	STC8A8K64D4 (F7F4)		

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

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

7.4.2 Interpretation of the internal reference signal source in CHIP

Symbol	Description	Address	Bit address and symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
CHIPID07	hardware digital ID07	FDE7H	Internal 1.19V reference signal source (high byte)								nnnn,nnnn
CHIPID08	hardware digital ID08	FDE8H	Internal 1.19V reference signal source (low byte)								nnnn,nnnn

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

The standard value is 1190 (04A6H), and the unit is mV, which is 1.19V. But there is error in the actual chip due to manufacturing errors. The voltage value of the internal reference signal source is not affected by the working voltage VCC, so the internal reference signal source can be combined with the ADC to calibrate the ADC, and can also be combined with the comparator to detect the operating voltage.

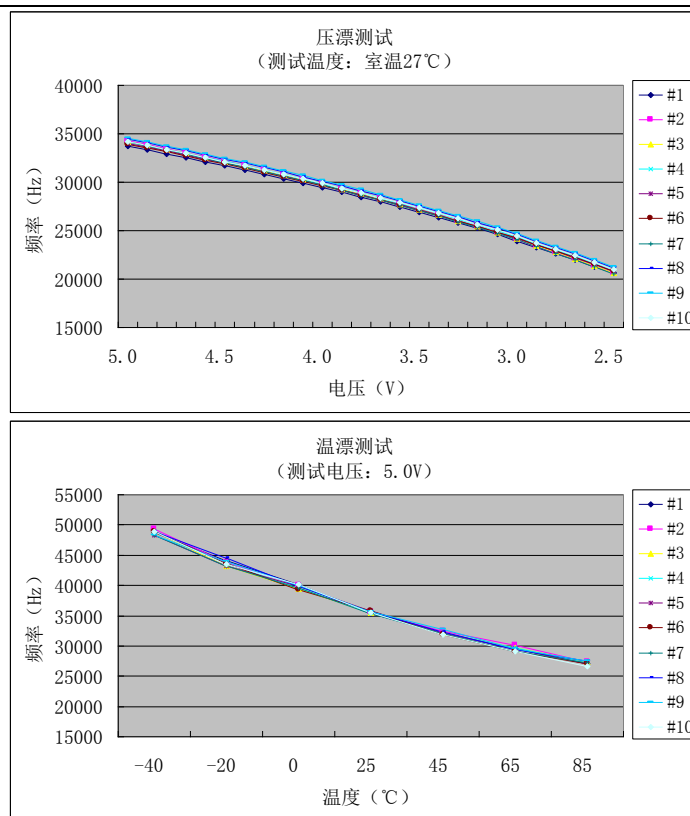
7.4.3 Interpretation of internal 32K IRC oscillation frequency in CHIP

Symbol	Description	Address	Bit address and symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
CHIPID09	hardware digital ID09	FDE9H	32K Power-down wake-up timer frequency (high byte)								nnnn,nnnn
CHIPID10	hardware digital ID10	FDEAH	32K Power-down wake-up timer frequency (low byte)								nnnn,nnnn

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

The standard value is 32768 (8000H), the unit is Hz, that is, 32.768KHz. However, the actual chip has manufacturing errors, and the temperature drift and pressure drift are relatively large.

The voltage drift test linear diagram and temperature drift linear diagram of the internal 32K oscillator are as follows:



7.4.4 Interpretation of high precision IRC parameters in CHIP

Symbol	Description	Address	Bit address and symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
CHIPID11	hardware digital ID11	FDEBH	USB series (STC8H8K64U family)				Other series				nnnn,nnnn
			IRC parameter of 22.1184MHz (27M band)				IRC parameter of 22.1184MHz (27M band)				
CHIPID12	hardware digital ID12	FDECH	IRC parameter of 24MHz (27M band)				IRC parameter of 24MHz (27M band)				nnnn,nnnn
CHIPID13	hardware digital ID13	FDEDH	IRC parameter of 27MHz (27M band)				IRC parameter of 20MHz (27M band)				nnnn,nnnn
CHIPID14	hardware digital ID14	FDEEH	IRC parameter of 30MHz (27M band)				IRC parameter of 27MHz (27M band)				nnnn,nnnn
CHIPID15	hardware digital ID15	FDEFH	IRC parameter of 33.1776MHz (27M band)				IRC parameter of 30MHz (27M band)				nnnn,nnnn
CHIPID16	hardware digital ID16	FDF0H	IRC parameter of 35MHz (44M band)				IRC parameter of 33.1776MHz (27M band)				nnnn,nnnn
CHIPID17	hardware digital ID17	FDF1H	IRC parameter of 36.864MHz (44M band)				IRC parameter of 35MHz (44M band)				nnnn,nnnn
CHIPID18	hardware digital ID18	FDF2H	IRC parameter of 40MHz (44M band)				IRC parameter of 36.864MHz(44M band)				nnnn,nnnn
CHIPID19	hardware digital ID19	FDF3H	IRC parameter of 44.2368MHz (44M band)				IRC parameter of 40MHz(44M band)				nnnn,nnnn
CHIPID20	hardware digital ID20	FDF4H	IRC parameter of 48MHz (44M band)				IRC parameter of 45MHz(44M band)				nnnn,nnnn
CHIPID21	hardware digital ID21	FDF5H	VRTRIM parameter of 6M band								
CHIPID22	hardware digital ID22	FDF6H	VRTRIM parameter of 10M band								
CHIPID23	hardware digital ID23	FDF7H	VRTRIM parameter of 27M band								
CHIPID24	hardware digital ID24	FDF8H	VRTRIM parameter of 44M band								

In the STC8H series MCUs that support the CHIPID function, the integrated high-precision IRC is divided into 4 frequency bands, and the reference voltage value corresponding to every frequency band has been calibrated at the factory. When selecting different frequency bands, you only need to fill the calibrated voltage value of the corresponding frequency band in the VRTRIM register. The center frequencies of the 4 frequency bands are 6MHz, 10MHz, 27MHz and 44MHz respectively. Due to manufacturing errors, the center frequency may generally have a deviation of $\pm 5\%$. In order to obtain accurate user frequencies, IRTRIM can be used to fine-tune the frequency. When using the download software provided by STC to download the user program, the system will automatically set the VRTRIM and IRTRIM registers according to the frequency set by the user. At the same time, the IRTRIM value of 10 common frequencies and the calibration value of the reference voltage value of 4 frequency bands are preset in CHIPID, so that the user can dynamically modify the working frequency during the running of the program.

[CHIPID11 : CHIPID20]: the IRTRIM value of 10 common frequencies. The annotations in parentheses are the corresponding frequency bands.

[CHIPID21 : CHIPID24]: the calibration value of the reference voltage value of 4 frequency bands.

When the user modifies the frequency dynamically, he only needs to read out a certain frequency calibration value in [CHIPID11 : CHIPID20] and write it into the IRTRIM register, and at the same time, according to the frequency band corresponding to the frequency, set a certain voltage calibration value in [CHIPID21 : CHIPID24] Just read and write

to the VRTRIM register. For detailed operations, please refer to the sample programs in the following chapters.

7.4.5 Interpretation of test time parameters in CHIP

Symbol	Description	Address	Bit address and symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
CHIPID27	hardware digital ID27	FDFBH	Chip test time (year)								nnnn,nnnn
CHIPID28	hardware digital ID28	FDFCH	Chip test time (month)								nnnn,nnnn
CHIPID29	hardware digital ID29	FDFDH	Chip test time (day)								nnnn,nnnn

The year, month, and day parameters of the test time are all BCD codes. For example, CHIPID27=0x21, CHIPID28=0x11, CHIPID29=0x18, then the production test date of the target chip is November 18, 2021.

7.4.6 Interpretation of chip package form number in CHIP

Symbol	Description	Address	Bit address and symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
CHIPID30	hardware digital ID30	FDFEH	Chip package form number								nnnn,nnnn

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

7.5 Example Routines

7.5.1 Read Internal Reference Voltage Value (Read from CHIPID)

C language code

//Operating frequency for test is 11.0592MHz

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

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

#define USBCHIPID
#define CPUIDBASE 0xfde0
#define VREF_ADDR (*(unsigned int volatile xdata *) (CPUIDBASE + 0x07))

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

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

bit busy;

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

void UartInit()
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
}
```



```

    TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}

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

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

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

    while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

CPUIDBASE EQU 0FDE0H
VREF_ADDR EQU CPUIDBASE + 07H

```

```
AUXR DATA 8EH
```

```
P_SW2 DATA 0BAH
```

```
BUSY BIT 20H.0
```

```
P0M1 DATA 093H
```

```
P0M0 DATA 094H
```

```
P1M1 DATA 091H
```

```
P1M0 DATA 092H
```

```
P2M1 DATA 095H
```

```
P2M0 DATA 096H
```

```
P3M1 DATA 0B1H
```

```
P3M0 DATA 0B2H
```

```
P4M1 DATA 0B3H
```

```
P4M0 DATA 0B4H
```

```
P5M1 DATA 0C9H
```

```

P5M0      DATA      0CAH

          ORG         0000H
          LJMP        MAIN
          ORG         0023H
          LJMP        UART_ISR

          ORG         0100H

UART_ISR:
          JNB         TI,CHKRI
          CLR         TI
          CLR         BUSY

CHKRI:
          JNB         RI,UARTISR_EXIT
          CLR         RI

UARTISR_EXIT:
          RETI

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

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

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

          LCALL        UART_INIT
          SETB        ES
          SETB        EA

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

```

```
MOVX    A,@DPTR
LCALL   UART_SEND
```

```
; Read the low byte of the internal 1.19V reference signal source
```

```
LOOP:
```

```
JMP     LOOP
```

```
END
```

7.5.2 Read Internal Reference Voltage (from Flash)

C language code

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

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

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

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

```
sfr      P0M1      = 0x93;
```

```
sfr      P0M0      = 0x94;
```

```
sfr      P1M1      = 0x91;
```

```
sfr      P1M0      = 0x92;
```

```
sfr      P2M1      = 0x95;
```

```
sfr      P2M0      = 0x96;
```

```
sfr      P3M1      = 0xb1;
```

```
sfr      P3M0      = 0xb2;
```

```
sfr      P4M1      = 0xb3;
```

```
sfr      P4M0      = 0xb4;
```

```
sfr      P5M1      = 0xc9;
```

```
sfr      P5M0      = 0xca;
```

```
bit      busy;
```

```
int      *BGV;
```

```
void UartIsr() interrupt 4
```

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

```
void UartInit()
```

```
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}
```

```

}

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

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

    BGV = (int code *)0x3ff7;           // STC8H1K16
    UartInit();
    ES = 1;
    EA = 1;
    UartSend(*BGV >> 8);               //Read the high byte of the internal reference voltage
    UartSend(*BGV);                   //Read the low byte of the internal reference voltage

    while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

AUXR	DATA	8EH	
BGV	EQU	01FF7H	;STC8H1K16
BUSY	BIT	20H.0	
P0M1	DATA	093H	
P0M0	DATA	094H	
P1M1	DATA	091H	
P1M0	DATA	092H	
P2M1	DATA	095H	
P2M0	DATA	096H	
P3M1	DATA	0B1H	
P3M0	DATA	0B2H	
P4M1	DATA	0B3H	
P4M0	DATA	0B4H	
P5M1	DATA	0C9H	
P5M0	DATA	0CAH	
	ORG	0000H	
	LJMP	MAIN	
	ORG	0023H	
	LJMP	UART_ISR	

```

    ORG        0100H

UART_ISR:
    JNB        TI,CHKRI
    CLR        TI
    CLR        BUSY

CHKRI:
    JNB        RI,UARTISR_EXIT
    CLR        RI

UARTISR_EXIT:
    RETI

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

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

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

    LCALL      UART_INIT
    SETB       ES
    SETB       EA

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

LOOP:
    JMP        LOOP

END

```

7.5.3 Read Internal Reference Voltage (Read from RAM)

C language code

//Operating frequency for test is 11.0592MHz

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

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

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

```
sfr P0M1 = 0x93;
```

```
sfr P0M0 = 0x94;
```

```
sfr P1M1 = 0x91;
```

```
sfr P1M0 = 0x92;
```

```
sfr P2M1 = 0x95;
```

```
sfr P2M0 = 0x96;
```

```
sfr P3M1 = 0xb1;
```

```
sfr P3M0 = 0xb2;
```

```
sfr P4M1 = 0xb3;
```

```
sfr P4M0 = 0xb4;
```

```
sfr P5M1 = 0xc9;
```

```
sfr P5M0 = 0xca;
```

```
bit busy;
```

```
int *BGV;
```

```
void UartIsr() interrupt 4
```

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

```
void UartInit()
```

```
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}
```

```
void UartSend(char dat)
```

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

```

    SBUF = dat;
}

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

    P5M0 = 0x00;
    P5M1 = 0x00;

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

    while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

AUXR	DATA	8EH
BGV	DATA	0EFH
BUSY	BIT	20H.0
P0M1	DATA	093H
P0M0	DATA	094H
P1M1	DATA	091H
P1M0	DATA	092H
P2M1	DATA	095H
P2M0	DATA	096H
P3M1	DATA	0B1H
P3M0	DATA	0B2H
P4M1	DATA	0B3H
P4M0	DATA	0B4H
P5M1	DATA	0C9H
P5M0	DATA	0CAH
	ORG	0000H
	LJMP	MAIN
	ORG	0023H
	LJMP	UART_ISR
	ORG	0100H
UART_ISR:		
	JNB	TI,CHKRI
	CLR	TI
	CLR	BUSY

CHKRI:

```
JNB    RI,UARTISR_EXIT
CLR    RI
```

UARTISR_EXIT:

```
RETI
```

UART_INIT:

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

UART_SEND:

```
JB     BUSY,$
SETB   BUSY
MOV    SBUF,A
RET
```

MAIN:

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

LCALL   UART_INIT
SETB    ES
SETB    EA

MOV     R0,#BGV
MOV     A,@R0    ;Read the high byte of the internal reference voltage
LCALL   UART_SEND
INC     R0
MOV     A,@R0    ;Read the low byte of the internal reference voltage
LCALL   UART_SEND
```

LOOP:

```
JMP     LOOP
```

```
END
```

7.5.4 Read the Unique ID (Read from CHIPID)

C language code

//Operating frequency for test is 11.0592MHz


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

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

#define CPUIDBASE 0xfde0
#define ID_ADDR   ((unsigned char volatile xdata *) (CPUIDBASE + 0x00))

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

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

bit busy;

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

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

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

void main()
```

```

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

    char i;

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

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

    while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

CPUIDBASE    EQU    0FDE0H
ID_ADDR      EQU    CPUIDBASE + 00H

AUXR         DATA    8EH
P_SW2        DATA    0BAH

BUSY         BIT      20H.0

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

                ORG      0000H
                LJMP     MAIN
                ORG      0023H
                LJMP     UART_ISR

                ORG      0100H

```

UART_ISR:

```
JNB    TI,CHKRI
CLR    TI
CLR    BUSY
```

CHKRI:

```
JNB    RI,UARTISR_EXIT
CLR    RI
```

UARTISR_EXIT:

```
RETI
```

UART_INIT:

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

UART_SEND:

```
JB     BUSY,$
SETB   BUSY
MOV    SBUF,A
RET
```

MAIN:

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

```
LCALL   UART_INIT
SETB    ES
SETB    EA
```

```
MOV    P_SW2,#80H
MOV    DPTR,#ID_ADDR
MOV    R1,#7
```

NEXT:

```
CLR    A
MOVX   A,@DPTR
LCALL  UART_SEND
INC    DPTR
DJNZ   R1,NEXT
```

LOOP:

```
JMP    LOOP
```

```
END
```

7.5.5 Read the Unique ID (Read from Flash)

C language code

//Operating frequency for test is 11.0592MHz

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

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

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

```
sfr      P0M1      = 0x93;
```

```
sfr      P0M0      = 0x94;
```

```
sfr      P1M1      = 0x91;
```

```
sfr      P1M0      = 0x92;
```

```
sfr      P2M1      = 0x95;
```

```
sfr      P2M0      = 0x96;
```

```
sfr      P3M1      = 0xb1;
```

```
sfr      P3M0      = 0xb2;
```

```
sfr      P4M1      = 0xb3;
```

```
sfr      P4M0      = 0xb4;
```

```
sfr      P5M1      = 0xc9;
```

```
sfr      P5M0      = 0xca;
```

```
bit      busy;
```

```
char     *ID;
```

```
void UartIsr() interrupt 4
```

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

```
void UartInit()
```

```
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}
```

```
void UartSend(char dat)
```

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

```

}

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

    char i;

    ID = (char code *)0x3ff9;                // STC8H1K16
    UartInit();
    ES = 1;
    EA = 1;

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

    while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

AUXR      DATA      8EH
ID         EQU        03FF9H                ; STC8H1K16

BUSY       BIT        20H.0

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

          ORG          0000H
          LJMP         MAIN
          ORG          0023H
          LJMP         UART_ISR

          ORG          0100H

```

UART_ISR:

```

JNB    TI,CHKRI
CLR    TI
CLR    BUSY

```

CHKRI:

```

JNB    RI,UARTISR_EXIT
CLR    RI

```

UARTISR_EXIT:

```

RETI

```

UART_INIT:

```

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

```

UART_SEND:

```

JB     BUSY,$
SETB   BUSY
MOV    SBUF,A
RET

```

MAIN:

```

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

```

```

LCALL   UART_INIT
SETB    ES
SETB    EA

```

```

MOV    DPTR,#ID
MOV    R1,#7

```

NEXT:

```

CLR    A
MOVC   A,@A+DPTR
LCALL  UART_SEND
INC    DPTR
DJNZ   R1,NEXT

```

LOOP:

```

JMP    LOOP

```

```

END

```

7.5.6 Read the Unique ID (Read from RAM)

C language code

//Operating frequency for test is 11.0592MHz

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

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

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

```
sfr P0M1 = 0x93;
```

```
sfr P0M0 = 0x94;
```

```
sfr P1M1 = 0x91;
```

```
sfr P1M0 = 0x92;
```

```
sfr P2M1 = 0x95;
```

```
sfr P2M0 = 0x96;
```

```
sfr P3M1 = 0xb1;
```

```
sfr P3M0 = 0xb2;
```

```
sfr P4M1 = 0xb3;
```

```
sfr P4M0 = 0xb4;
```

```
sfr P5M1 = 0xc9;
```

```
sfr P5M0 = 0xca;
```

```
bit busy;
```

```
char *ID;
```

```
void UartIsr() interrupt 4
```

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

```
void UartInit()
```

```
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}
```

```
void UartSend(char dat)
```

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

```

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

    char i;

    ID = (char idata *)0xf1;
    UartInit();
    ES = 1;
    EA = 1;

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

    while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

AUXR	DATA	8EH
ID	DATA	0F1H
BUSY	BIT	20H.0
P0M1	DATA	093H
P0M0	DATA	094H
P1M1	DATA	091H
P1M0	DATA	092H
P2M1	DATA	095H
P2M0	DATA	096H
P3M1	DATA	0B1H
P3M0	DATA	0B2H
P4M1	DATA	0B3H
P4M0	DATA	0B4H
P5M1	DATA	0C9H
P5M0	DATA	0CAH
	ORG	0000H
	LJMP	MAIN
	ORG	0023H
	LJMP	UART_ISR
	ORG	0100H

UART_ISR:

```

        JNB     TI,CHKRI
        CLR     TI
        CLR     BUSY

CHKRI:
        JNB     RI,UARTISR_EXIT
        CLR     RI

UARTISR_EXIT:
        RETI

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

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

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

        LCALL   UART_INIT
        SETB    ES
        SETB    EA

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

LOOP:
        JMP     LOOP

        END

```

7.5.7 Read the Frequency of 32K Power-down Wake-up Timer (Read from CHIPID)

C language code

//Operating frequency for test is 11.0592MHz

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

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

#define CPUIDBASE 0xfde0
#define F32K_ADDR (*(unsigned int volatile xdata *) (CPUIDBASE + 0x09))

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

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

bit busy;

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

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

```
void UartSend(char dat)
```

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

```
void main()
```

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

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

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

    while (1);
}
```

Assembly code

;Operating frequency for test is 11.0592MHz

CPUIDBASE	EQU	0FDE0H
F32K_ADDR	EQU	CPUIDBASE + 09H
AUXR	DATA	8EH
P_SW2	DATA	0BAH
BUSY	BIT	20H.0
P0M1	DATA	093H
P0M0	DATA	094H
P1M1	DATA	091H
P1M0	DATA	092H
P2M1	DATA	095H
P2M0	DATA	096H
P3M1	DATA	0B1H
P3M0	DATA	0B2H
P4M1	DATA	0B3H
P4M0	DATA	0B4H
P5M1	DATA	0C9H
P5M0	DATA	0CAH
	ORG	0000H
	LJMP	MAIN

```

    ORG      0023H
    LJMP     UART_ISR

    ORG      0100H

UART_ISR:
    JNB      TI,CHKRI
    CLR      TI
    CLR      BUSY

CHKRI:
    JNB      RI,UARTISR_EXIT
    CLR      RI

UARTISR_EXIT:
    RETI

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

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

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

    LCALL    UART_INIT
    SETB     ES
    SETB     EA

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

```

LOOP:
*JMP LOOP**END*

7.5.8 Read the Frequency of 32K Power-down Wake-up Timer (Read from Flash)

C language code

//Operating frequency for test is 11.0592MHz

#include "reg51.h"

#include "intrins.h"

#define FOSC 11059200UL

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

sfr AUXR = 0x8e;

sfr P0M1 = 0x93;

sfr P0M0 = 0x94;

sfr P1M1 = 0x91;

sfr P1M0 = 0x92;

sfr P2M1 = 0x95;

sfr P2M0 = 0x96;

sfr P3M1 = 0xb1;

sfr P3M0 = 0xb2;

sfr P4M1 = 0xb3;

sfr P4M0 = 0xb4;

sfr P5M1 = 0xc9;

sfr P5M0 = 0xca;

bit busy;

*int *F32K;*

void UartIsr() interrupt 4

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

void UartInit()

```
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
}
```

```

    busy = 0;
}

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

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

    F32K = (int code *)0x3ff5;           // STC8H1K16
    UartInit();
    ES = 1;
    EA = 1;

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

    while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

AUXR	DATA	8EH	
F32K	EQU	03FF5H	; STC8H1K16
BUSY	BIT	20H.0	
P0M1	DATA	093H	
P0M0	DATA	094H	
P1M1	DATA	091H	
P1M0	DATA	092H	
P2M1	DATA	095H	
P2M0	DATA	096H	
P3M1	DATA	0B1H	
P3M0	DATA	0B2H	
P4M1	DATA	0B3H	
P4M0	DATA	0B4H	
P5M1	DATA	0C9H	
P5M0	DATA	0CAH	
	ORG	0000H	
	LJMP	MAIN	
	ORG	0023H	

```

        LJMP      UART_ISR

        ORG      0100H

UART_ISR:
        JNB      TI,CHKRI
        CLR      TI
        CLR      BUSY

CHKRI:
        JNB      RI,UARTISR_EXIT
        CLR      RI

UARTISR_EXIT:
        RETI

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

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

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

        LCALL    UART_INIT
        SETB     ES
        SETB     EA

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

LOOP:
        JMP      LOOP

```

END

7.5.9 Read the Frequency of 32K Power-down Wake-up Timer (Read from RAM)

C language code

//Operating frequency for test is 11.0592MHz

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

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

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

```
sfr      P0M1      = 0x93;
```

```
sfr      P0M0      = 0x94;
```

```
sfr      P1M1      = 0x91;
```

```
sfr      P1M0      = 0x92;
```

```
sfr      P2M1      = 0x95;
```

```
sfr      P2M0      = 0x96;
```

```
sfr      P3M1      = 0xb1;
```

```
sfr      P3M0      = 0xb2;
```

```
sfr      P4M1      = 0xb3;
```

```
sfr      P4M0      = 0xb4;
```

```
sfr      P5M1      = 0xc9;
```

```
sfr      P5M0      = 0xca;
```

```
bit      busy;
```

```
int      *F32K;
```

```
void UartIsr() interrupt 4
```

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

```
void UartInit()
```

```
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
    busy = 0;
}
```



```
void UartSend(char dat)
```

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

```
void main()
```

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

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

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

    while (1);
}
```

Assembly code

;Operating frequency for test is 11.0592MHz

AUXR	DATA	8EH
F32K	DATA	0F8H
BUSY	BIT	20H.0
P0M1	DATA	093H
P0M0	DATA	094H
P1M1	DATA	091H
P1M0	DATA	092H
P2M1	DATA	095H
P2M0	DATA	096H
P3M1	DATA	0B1H
P3M0	DATA	0B2H
P4M1	DATA	0B3H
P4M0	DATA	0B4H
P5M1	DATA	0C9H
P5M0	DATA	0CAH
	ORG	0000H
	LJMP	MAIN
	ORG	0023H
	LJMP	UART_ISR

```

        ORG          0100H

UART_ISR:
        JNB          TI,CHKRI
        CLR          TI
        CLR          BUSY

CHKRI:
        JNB          RI,UARTISR_EXIT
        CLR          RI

UARTISR_EXIT:
        RETI

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

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

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

        LCALL        UART_INIT
        SETB         ES
        SETB         EA

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

LOOP:
        JMP          LOOP

        END

```

7.5.10 Read the User-defined internal IRC Frequency (Read from CHIPID)

C language code

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

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

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

```
#define USBCHIPID
#define CPUIDBASE    0xfde0
```

```
#define T22M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0b))    //22.1184MHz
#define T24M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0c))    //24MHz
#ifdef USBCHIPID
#define T27M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0d))    //27MHz
#define T30M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0e))    //30MHz
#define T33M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0f))    //33.1776MHz
#define T35M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x10))    //35MHz
#define T36M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x11))    //36.864MHz
#define T40M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x12))    //40MHz
#define T44M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x13))    //44.2368MHz
#define T48M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x14))    //48MHz
#else
#define T20M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0d))    //20MHz
#define T27M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0e))    //27MHz
#define T30M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x0f))    //30MHz
#define T33M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x10))    //33.1776MHz
#define T35M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x11))    //35MHz
#define T36M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x12))    //36.864MHz
#define T40M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x13))    //40MHz
#define T45M_ADDR    (*(unsigned char volatile xdata *)(CPUIDBASE + 0x14))    //45MHz
#endif
#define VRT6M_ADDR   (*(unsigned char volatile xdata *)(CPUIDBASE + 0x15))    //VRTRIM_6M
#define VRT10M_ADDR  (*(unsigned char volatile xdata *)(CPUIDBASE + 0x16))    //VRTRIM_10M
#define VRT27M_ADDR  (*(unsigned char volatile xdata *)(CPUIDBASE + 0x17))    //VRTRIM_27M
#define VRT44M_ADDR  (*(unsigned char volatile xdata *)(CPUIDBASE + 0x18))    //VRTRIM_44M
```

```
sfr P_SW2      = 0xba;
sfr IRCBAND    = 0x9d;
sfr IRTRIM     = 0x9f;
sfr VRTRIM     = 0xa6;
```

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

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

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

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

    // Select 24MHz
    P_SW2 = 0x80;
    CLKDIV = 0x04;
    IRTRIM = T24M_ADDR;
    VRTRIM = VRT27M_ADDR;
    IRCBAND = 0x02;
    CLKDIV = 0x00;

    //    // Select 27MHz
    //    P_SW2 = 0x80;
    //    CLKDIV = 0x04;
    //    IRTRIM = T27M_ADDR;
    //    VRTRIM = VRT27M_ADDR;
    //    IRCBAND = 0x02;
    //    CLKDIV = 0x00;

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

    //    // Select 33.1776MHz
    //    P_SW2 = 0x80;
    //    CLKDIV = 0x04;
```

```

//  IRTRIM = T33M_ADDR;
//  VRTRIM = VRT27M_ADDR;
//  IRCBAND = 0x02;
//  CLKDIV = 0x00;

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

#ifdef  USBCHIPID
//  // Select 44.2368MHz
//  P_SW2 = 0x80;
//  CLKDIV = 0x04;
//  IRTRIM = T44M_ADDR;
//  VRTRIM = VRT44M_ADDR;
//  IRCBAND = 0x03;
//  CLKDIV = 0x00;

//  // Select 48MHz
//  P_SW2 = 0x80;
//  CLKDIV = 0x04;
//  IRTRIM = T48M_ADDR;
//  VRTRIM = VRT44M_ADDR;
//  IRCBAND = 0x03;
//  CLKDIV = 0x00;
#else
//  // Select 40MHz
//  P_SW2 = 0x80;
//  CLKDIV = 0x04;
//  IRTRIM = T40M_ADDR;
//  VRTRIM = VRT44M_ADDR;
//  IRCBAND = 0x03;
//  CLKDIV = 0x00;

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

    while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

$SET(USBCHIPID = 1)
CPUIDBASE      EQU      0FDE0H

T22M_ADDR      EQU      CPUIDBASE + 0BH      ;22.1184MHz
T24M_ADDR      EQU      CPUIDBASE + 0CH      ;24MHz
$IF(USBCHIPID == 1)

```

T27M_ADDR	EQU	CPUIDBASE + 0DH	;27MHz
T30M_ADDR	EQU	CPUIDBASE + 0EH	;30MHz
T33M_ADDR	EQU	CPUIDBASE + 0FH	;33.1776MHz
T35M_ADDR	EQU	CPUIDBASE + 10H	;35MHz
T36M_ADDR	EQU	CPUIDBASE + 11H	;36.864MHz
T40M_ADDR	EQU	CPUIDBASE + 12H	;40MHz
T44M_ADDR	EQU	CPUIDBASE + 13H	;44.2368MHz
T48M_ADDR	EQU	CPUIDBASE + 14H	;20MHz
\$ELSE			
T20M_ADDR	EQU	CPUIDBASE + 0DH	;20MHz
T27M_ADDR	EQU	CPUIDBASE + 0EH	;27MHz
T30M_ADDR	EQU	CPUIDBASE + 0FH	;30MHz
T33M_ADDR	EQU	CPUIDBASE + 10H	;33.1776MHz
T35M_ADDR	EQU	CPUIDBASE + 11H	;35MHz
T36M_ADDR	EQU	CPUIDBASE + 12H	;36.864MHz
T40M_ADDR	EQU	CPUIDBASE + 13H	;40MHz
T45M_ADDR	EQU	CPUIDBASE + 14H	;45MHz
\$ENDIF			
VRT6M_ADDR	EQU	CPUIDBASE + 15H	;VRTRIM_6M
VRT10M_ADDR	EQU	CPUIDBASE + 16H	;VRTRIM_10M
VRT27M_ADDR	EQU	CPUIDBASE + 17H	;VRTRIM_27M
VRT44M_ADDR	EQU	CPUIDBASE + 18H	;VRTRIM_44M
P_SW2	DATA	0BAH	
CLKSEL	EQU	0FE00H	
CLKDIV	EQU	0FE01H	
IRCBAND	DATA	09DH	
IRCTRIM	DATA	09FH	
VRTRIM	DATA	0A6H	
P1M1	DATA	091H	
P1M0	DATA	092H	
P0M1	DATA	093H	
P0M0	DATA	094H	
P2M1	DATA	095H	
P2M0	DATA	096H	
P3M1	DATA	0B1H	
P3M0	DATA	0B2H	
P4M1	DATA	0B3H	
P4M0	DATA	0B4H	
P5M1	DATA	0C9H	
P5M0	DATA	0CAH	
	ORG	0000H	
	LJMP	MAIN	
	ORG	0100H	
MAIN:			
	MOV	SP, #5FH	
	MOV	P0M0, #00H	
	MOV	P0M1, #00H	
	MOV	P1M0, #00H	
	MOV	P1M1, #00H	
	MOV	P2M0, #00H	
	MOV	P2M1, #00H	
	MOV	P3M0, #00H	
	MOV	P3M1, #00H	
	MOV	P4M0, #00H	
	MOV	P4M1, #00H	

```

MOV    P5M0, #00H
MOV    P5M1, #00H

```

```

; $IF(USBCHIPID == 0)

```

```

;      ; Select 20MHz
;      MOV    P_SW2, #80H
;      MOV    A, #4
;      MOV    DPTR, #CLKDIV
;      MOVX   @DPTR, A
;      MOV    DPTR, #T20M_ADDR
;      CLR    A
;      MOVX   A, @DPTR
;      MOV    IRTRIM, A
;      MOV    DPTR, #VRT27M_ADDR
;      CLR    A
;      MOVX   A, @DPTR
;      MOV    VRTRIM, A
;      MOV    IRCBAND, #02H
;      MOV    A, #0
;      MOV    DPTR, #CLKDIV
;      MOVX   @DPTR, A
;      MOV    P_SW2, #00H

```

```

; $ENDIF

```

```

;      ; Select 22.1184MHz

```

```

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

```

```

;      ; Select 24MHz

```

```

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

```

MOV P_SW2,#00H

```

;      ; Select 27MHz
;      MOV      P_SW2,#80H
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A
;      MOV      DPTR,#T27M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT27M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#02H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A
;      MOV      P_SW2,#00H

```

```

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

```

```

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

```

```

;      ; Select 35MHz

```

```

;      MOV      P_SW2,#80H
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A
;      MOV      DPTR,#T35M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT44M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#03H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A
;      MOV      P_SW2,#00H

```

```

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

```

```

;$IF(USBCHIPID == 1)

```

```

;      ; Select 44.2368MHz
;      MOV      P_SW2,#80H
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A
;      MOV      DPTR,#T44M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT44M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#03H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A
;      MOV      P_SW2,#00H

```

```

;      ; Select 48MHz
;      MOV      P_SW2,#80H
;      MOV      A,#4

```

```

;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A
;      MOV      DPTR,#T48M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT44M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#03H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A
;      MOV      P_SW2,#00H
;ELSE
;      ; Select 40MHz
;      MOV      P_SW2,#80H
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A
;      MOV      DPTR,#T40M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT44M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#03H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A
;      MOV      P_SW2,#00H
;
;      ; Select 45MHz
;      MOV      P_SW2,#80H
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A
;      MOV      DPTR,#T45M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT44M_ADDR
;      CLR      A
;      MOVX     A,@DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#03H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV
;      MOVX     @DPTR,A
;      MOV      P_SW2,#00H
;ENDIF

```

JMP \$

END

7.5.11 Read the User-defined internal IRC Frequency (Read from Flash)

C language code

//Operating frequency for test is 11.0592MHz

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

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

// The following table is the parameter list of STC8H1K08-20Pin

```
#define ID_ROMADDR      ((unsigned char code *)0x1ff9)
#define VREF_ROMADDR    (*(unsigned int code *)0x1ff7)
#define F32K_ROMADDR    (*(unsigned int code *)0x1ff5)
#define T22M_ROMADDR    (*(unsigned char code *)0x1ff4)           //22.1184MHz
#define T24M_ROMADDR    (*(unsigned char code *)0x1ff3)           //24MHz
#define T20M_ROMADDR    (*(unsigned char code *)0x1ff2)           //20MHz
#define T27M_ROMADDR    (*(unsigned char code *)0x1ff1)           //27MHz
#define T30M_ROMADDR    (*(unsigned char code *)0x1ff0)           //30MHz
#define T33M_ROMADDR    (*(unsigned char code *)0x1fef)           //33.1776MHz
#define T35M_ROMADDR    (*(unsigned char code *)0x1fee)           //35MHz
#define T36M_ROMADDR    (*(unsigned char code *)0x1fed)           //36.864MHz
#define VRT20M_ROMADDR  (*(unsigned char code *)0x1fea)           //VRTRIM_20M
#define VRT35M_ROMADDR  (*(unsigned char code *)0x1fe9)           //VRTRIM_35M
```

```
sfr      P_SW2      = 0xba;
sfr      IRCBAND    = 0x9d;
sfr      IRTIM      = 0x9f;
sfr      VRTRIM     = 0xa6;
```

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

void main()

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

```
// //Select 20MHz
// P_SW2 = 0x80;
// CLKDIV = 0x04;
// IRTRIM = T20M_ROMADDR;
// VRTRIM = VRT20M_ROMADDR;
// IRCBAND = 0x00;
// CLKDIV = 0x00;

// // Select 22.1184MHz
// P_SW2 = 0x80;
// CLKDIV = 0x04;
// IRTRIM = T22M_ROMADDR;
// VRTRIM = VRT20M_ROMADDR;
// IRCBAND = 0x00;
// CLKDIV = 0x00;

// Select 24MHz
P_SW2 = 0x80;
CLKDIV = 0x04;
IRTRIM = T24M_ROMADDR;
VRTRIM = VRT20M_ROMADDR;
IRCBAND = 0x00;
CLKDIV = 0x00;

// // Select 27MHz
// P_SW2 = 0x80;
// CLKDIV = 0x04;
// IRTRIM = T27M_ROMADDR;
// VRTRIM = VRT35M_ROMADDR;
// IRCBAND = 0x01;
// CLKDIV = 0x00;

// // Select 30MHz
// P_SW2 = 0x80;
// CLKDIV = 0x04;
// IRTRIM = T30M_ROMADDR;
// VRTRIM = VRT35M_ROMADDR;
// IRCBAND = 0x01;
// CLKDIV = 0x00;

// // Select 33.1776MHz
// P_SW2 = 0x80;
// CLKDIV = 0x04;
// IRTRIM = T33M_ROMADDR;
// VRTRIM = VRT35M_ROMADDR;
// IRCBAND = 0x01;
// CLKDIV = 0x00;

// // Select 35MHz
// P_SW2 = 0x80;
// CLKDIV = 0x04;
// IRTRIM = T35M_ROMADDR;
// VRTRIM = VRT35M_ROMADDR;
// IRCBAND = 0x01;
// CLKDIV = 0x00;

while (1);
}
```

Assembly code

;Operating frequency for test is 11.0592MHz

; The following table is the parameter list of STC8H1K08-20Pin

ID_ROMADDR	EQU	01FF9H	
VREF_ROMADDR	EQU	01FF7H	
F32K_ROMADDR	EQU	01FF5H	
T22M_ROMADDR	EQU	01FF4H	//22.1184MHz
T24M_ROMADDR	EQU	01FF3H	//24MHz
T20M_ROMADDR	EQU	01FF2H	//20MHz
T27M_ROMADDR	EQU	01FF1H	//27MHz
T30M_ROMADDR	EQU	01FF0H	//30MHz
T33M_ROMADDR	EQU	01FEFH	//33.1776MHz
T35M_ROMADDR	EQU	01FEEH	//35MHz
T36M_ROMADDR	EQU	01FEDH	//36.864MHz
VRT20M_ROMADDR	EQU	01FEAH	//VRTRIM_20M
VRT35M_ROMADDR	EQU	01FE9H	//VRTRIM_35M

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

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

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

ORG 0000H
LJMP MAIN

ORG 0100H

MAIN:

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

```
; ;Select 20MHz
; MOV P_SW2, #80H
```

```

;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOV      DPTR,#T20M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT20M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#00H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV
;      MOV      P_SW2,#00H

```

```

;      ; Select 22.1184MHz
;      MOV      P_SW2,#80H
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOV      DPTR,#T22M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT20M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#00H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV
;      MOV      P_SW2,#00H

```

```

;      ; Select 24MHz
;      MOV      P_SW2,#80H
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOV      DPTR,#T24M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT20M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#00H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV
;      MOV      P_SW2,#00H

```

```

;      ; Select 27MHz
;      MOV      P_SW2,#80H
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOV      DPTR,#T27M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT35M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR

```

```

;      MOV      VRTRIM,A
;      MOV      IRCBAND,#01H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV
;      MOV      P_SW2,#00H

;      ; Select 30MHz
;      MOV      P_SW2,#80H
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOV      DPTR,#T30M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT35M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#01H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV
;      MOV      P_SW2,#00H

;      ; Select 33.1776MHz
;      MOV      P_SW2,#80H
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOV      DPTR,#T33M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT35M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#01H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV
;      MOV      P_SW2,#00H

;      ; Select 35MHz
;      MOV      P_SW2,#80H
;      MOV      A,#4
;      MOV      DPTR,#CLKDIV
;      MOV      DPTR,#T35M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT35M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#01H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV
;      MOV      P_SW2,#00H

;      ; Select 36.864MHz
;      MOV      P_SW2,#80H
;      MOV      A,#4

```

```

;      MOV      DPTR,#CLKDIV
;      MOV      DPTR,#T36M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      IRTRIM,A
;      MOV      DPTR,#VRT35M_ROMADDR
;      CLR      A
;      MOVC     A,@A+DPTR
;      MOV      VRTRIM,A
;      MOV      IRCBAND,#01H
;      MOV      A,#0
;      MOV      DPTR,#CLKDIV
;      MOV      P_SW2,#00H

```

```

JMP      $

```

```

END

```

7.5.12 Read the User-defined internal IRC Frequency (Read from RAM)

C language code

//Operating frequency for test is 11.0592MHz

```

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

```

```

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

```

```

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

```

```

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

```

```

char      *IRC22M;
char      *IRC24M;

```

```

void main()

```

```

{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;

```



```

P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

IRC22M = (char idata *)0xfa;
IRC24M = (char idata *) 0xfb;
// IRCCR = *IRC22M;           //Load 22.1184MHz IRC parameters
IRCCR = *IRC24M;             //Load 24MHz IRC parameters

P_SW2 = 0x80;
CLKDIV = 0;                  //No division to main clock
P_SW2 = 0x00;

while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

P_SW2	DATA	0BAH
CLKDIV	EQU	0FE01H
IRCCR	DATA	09FH
IRC22M	DATA	0FAH
IRC24M	DATA	0FBH
P0M1	DATA	093H
P0M0	DATA	094H
P1M1	DATA	091H
P1M0	DATA	092H
P2M1	DATA	095H
P2M0	DATA	096H
P3M1	DATA	0B1H
P3M0	DATA	0B2H
P4M1	DATA	0B3H
P4M0	DATA	0B4H
P5M1	DATA	0C9H
P5M0	DATA	0CAH
	ORG	0000H
	LJMP	MAIN
	ORG	0100H
MAIN:		
	MOV	SP, #5FH
	MOV	P0M0, #00H
	MOV	P0M1, #00H
	MOV	P1M0, #00H
	MOV	P1M1, #00H
	MOV	P2M0, #00H
	MOV	P2M1, #00H
	MOV	P3M0, #00H
	MOV	P3M1, #00H
	MOV	P4M0, #00H
	MOV	P4M1, #00H
	MOV	P5M0, #00H
	MOV	P5M1, #00H

```
;      MOV      R0,#IRC22M      ;Load 22.1184MHz IRC parameters
;      MOV      IRCCR,@R0
      MOV      R0,#IRC24M      ;Load 24MHz IRC parameters
      MOV      IRCCR,@R0

      MOV      P_SW2,#80H
      MOV      A,#0            ;No division to main clock
      MOV      DPTR,#CLKDIV
      MOVX     @DPTR,A
      MOV      P_SW2,#00H

      JMP      $

      END
```

8 Special Function Registers

8.1 STC8H1K08 family

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8H								RSTCFG
F0H	B					IAP_TPS		
E8H								AUXINTIF
E0H	ACC			DPS	DPL1	DPH1	CMPCR1	CMPCR2
D8H							ADCCFG	
D0H	PSW						T2H	T2L
C8H	P5	P5M1	P5M0			SPSTAT	SPCTL	SPDAT
C0H		WDT_CONTR	IAP_DATA	IAP_ADDRH	IAP_ADDRL	IAP_CMD	IAP_TRIG	IAP_CONTR
B8H	IP	SADEN	P_SW2		ADC_CONTR	ADC_RES	ADC_RESL	
B0H	P3	P3M1	P3M0			IP2	IP2H	IPH
A8H	IE	SADDR	WKTCL	WKTCH			TA	IE2
A0H			P_SW1					
98H	SCON	SBUF	S2CON	S2BUF		IRCBAND	LIRTRIM	IRTRIM
90H	P1	P1M1	P1M0					
88H	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR	INTCLKO
80H		SP	DPL	DPH				PCON

↑
Bit addressable

Not bit addressable

Note: The register address can be bit-addressable only if it is divisible by 8, and it cannot be bit-addressable if it is not divisible by 8.

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
FEF8H	PWMB_CCR6L	PWMB_CCR7H	PWMB_CCR7L	PWMB_CCR8H	PWMB_CCR8L	PWMB_BKR	PWMB_DTR	PWMB_OISR
FEF0H	PWMB_PSCRH	PWMB_PSCRL	PWMB_ARRH	PWMB_ARRL	PWMB_RCR	PWMB_CCR5H	PWMB_CCR5L	PWMB_CCR6H
FEE8H	PWMB_CCMR1	PWMB_CCMR2	PWMB_CCMR3	PWMB_CCMR4	PWMB_CCER1	PWMB_CCER2	PWMB_CNTRH	PWMB_CNTRL
FEE0H	PWMB_CR1	PWMB_CR2	PWMB_SMCR	PWMB_ETR	PWMB_IER	PWMB_SR1	PWMB_SR2	PWMB_EGR
FED8H	PWMA_CCR2L	PWMA_CCR3H	PWMA_CCR3L	PWMA_CCR4H	PWMA_CCR4L	PWMA_BKR	PWMA_DTR	PWMA_OISR
FED0H	PWMA_PSCRH	PWMA_PSCRL	PWMA_ARRH	PWMA_ARRL	PWMA_RCR	PWMA_CCR1H	PWMA_CCR1L	PWMA_CCR2H
FEC8H	PWMA_CCMR1	PWMA_CCMR2	PWMA_CCMR3	PWMA_CCMR4	PWMA_CCER1	PWMA_CCER2	PWMA_CNTRH	PWMA_CNTRL
FEC0H	PWMA_CR1	PWMA_CR2	PWMA_SMCR	PWMA_ETR	PWMA_IER	PWMA_SR1	PWMA_SR2	PWMA_EGR
FEB0H	PWMA_ETRPS	PWMA_ENO	PWMA_PS	PWMA_IOAUX	PWMB_ETRPS	PWMB_ENO	PWMB_PS	PWMB_IOAUX
FEA8H	ADCTIM							
FEA0H			TM2PS					
FE88H	I2CMSAUX							
FE80H	I2CCFG	I2CMSCR	I2CMSST	I2CSLCR	I2CSLST	I2CSLADR	I2CTxD	I2CRxD
FE30H		P1IE		P3IE				
FE28H		P1DR		P3DR		P5DR		
FE20H		P1SR		P3SR		P5SR		
FE18H		P1NCS		P3NCS		P5NCS		
FE10H		P1PU		P3PU		P5PU		
FE00H	CKSEL	CLKDIV	HIRCCR	XOSCCR	IRC32KCR	MCLKOCR	IRCDDB	

8.2 STC8H1K28-32PIN family

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8H								RSTCFG
F0H	B					IAP_TPS		
E8H								AUXINTIF
E0H	ACC			DPS	DPL1	DPH1	CMPCR1	CMPCR2
D8H							ADCCFG	
D0H	PSW	T4T3M	T4H	T4L	T3H	T3L	T2H	T2L
C8H	P5	P5M1	P5M0			SPSTAT	SPCTL	SPDAT
C0H		WDT_CONTR	IAP_DATA	IAP_ADDRH	IAP_ADDRL	IAP_CMD	IAP_TRIG	IAP_CONTR
B8H	IP	SADEN	P_SW2		ADC_CONTR	ADC_RES	ADC_RESL	
B0H	P3	P3M1	P3M0			IP2	IP2H	IPH
A8H	IE	SADDR	WKTCL	WKTCH			TA	IE2
A0H	P2		P_SW1					
98H	SCON	SBUF	S2CON	S2BUF		IRCBAND	LIRTRIM	IRTRIM
90H	P1	P1M1	P1M0	P0M1	P0M0	P2M1	P2M0	
88H	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR	INTCLKO
80H	P0	SP	DPL	DPH				PCON

Note: The register address can be bit-addressable only if it is divisible by 8, and it cannot be bit-addressable if it is not divisible by 8.

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
FEF8H	PWMB_CCR6L	PWMB_CCR7H	PWMB_CCR7L	PWMB_CCR8H	PWMB_CCR8L	PWMB_BKR	PWMB_DTR	PWMB_OISR
FEF0H	PWMB_PSCRH	PWMB_PSCRL	PWMB_ARRH	PWMB_ARRL	PWMB_RCR	PWMB_CCR5H	PWMB_CCR5L	PWMB_CCR6H
FEE8H	PWMB_CCMR1	PWMB_CCMR2	PWMB_CCMR3	PWMB_CCMR4	PWMB_CCER1	PWMB_CCER2	PWMB_CNTRH	PWMB_CNTRL
FEE0H	PWMB_CR1	PWMB_CR2	PWMB_SMCR	PWMB_ETR	PWMB_IER	PWMB_SR1	PWMB_SR2	PWMB_EGR
FED8H	PWMA_CCR2L	PWMA_CCR3H	PWMA_CCR3L	PWMA_CCR4H	PWMA_CCR4L	PWMA_BKR	PWMA_DTR	PWMA_OISR
FED0H	PWMA_PSCRH	PWMA_PSCRL	PWMA_ARRH	PWMA_ARRL	PWMA_RCR	PWMA_CCR1H	PWMA_CCR1L	PWMA_CCR2H
FEC8H	PWMA_CCMR1	PWMA_CCMR2	PWMA_CCMR3	PWMA_CCMR4	PWMA_CCER1	PWMA_CCER2	PWMA_CNTRH	PWMA_CNTRL
FEC0H	PWMA_CR1	PWMA_CR2	PWMA_SMCR	PWMA_ETR	PWMA_IER	PWMA_SR1	PWMA_SR2	PWMA_EGR
FEB0H	PWMA_ETRPS	PWMA_ENO	PWMA_PS	PWMA_IOAUX	PWMB_ETRPS	PWMB_ENO	PWMB_PS	PWMB_IOAUX
FEA8H	ADCTIM							
FEA0H			TM2PS	TM3PS	TM4PS			
FE88H	I2CMSAUX							
FE80H	I2CCFG	I2CMSCR	I2CMSST	I2CSLCR	I2CSLST	I2CSLADR	I2CTxD	I2CRxD
FE30H	P0IE	P1IE						
FE28H	P0DR	P1DR	P2DR	P3DR		P5DR		
FE20H	P0SR	P1SR	P2SR	P3SR		P5SR		
FE18H	P0NCS	P1NCS	P2NCS	P3NCS		P5NCS		
FE10H	P0PU	P1PU	P2PU	P3PU		P5PU		
FE00H	CKSEL	CLKDIV	HIRCCR	XOSCCR	IRC32KCR	MCLKOCR	IRCDDB	

8.3 STC8H3K64S4 family

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8H	P7							RSTCFG
F0H	B					IAP_TPS		
E8H	P6						IP3H	AUXINTIF
E0H	ACC	P7M1	P7M0	DPS	DPL1	DPH1	CMPCR1	CMPCR2
D8H							ADCCFG	IP3
D0H	PSW	T4T3M	T4H	T4L	T3H	T3L	T2H	T2L
C8H	P5	P5M1	P5M0	P6M1	P6M0	SPSTAT	SPCTL	SPDAT
C0H	P4	WDT_CONTR	IAP_DATA	IAP_ADDRH	IAP_ADDRL	IAP_CMD	IAP_TRIG	IAP_CONTR
B8H	IP	SADEN	P_SW2		ADC_CONTR	ADC_RES	ADC_RESL	
B0H	P3	P3M1	P3M0	P4M1	P4M0	IP2	IP2H	IPH
A8H	IE	SADDR	WKTCL	WKTCH	S3CON	S3BUF	TA	IE2
A0H	P2	BUS_SPEED	P_SW1					
98H	SCON	SBUF	S2CON	S2BUF		IRCBAND	LIRTRIM	IRTRIM
90H	P1	P1M1	P1M0	P0M1	P0M0	P2M1	P2M0	
88H	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR	INTCLKO
80H	P0	SP	DPL	DPH	S4CON	S4BUF		PCON

Note: The register address can be bit-addressable only if it is divisible by 8, and it cannot be bit-addressable if it is not divisible by 8.

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
FEF8H	PWMB_CCR6L	PWMB_CCR7H	PWMB_CCR7L	PWMB_CCR8H	PWMB_CCR8L	PWMB_BKR	PWMB_DTR	PWMB_OISR
FEF0H	PWMB_PSCRH	PWMB_PSCRL	PWMB_ARRH	PWMB_ARRL	PWMB_RCR	PWMB_CCR5H	PWMB_CCR5L	PWMB_CCR6H
FEE8H	PWMB_CCMR1	PWMB_CCMR2	PWMB_CCMR3	PWMB_CCMR4	PWMB_CCER1	PWMB_CCER2	PWMB_CNTRH	PWMB_CNTRL
FEE0H	PWMB_CR1	PWMB_CR2	PWMB_SMCRL	PWMB_ETR	PWMB_IER	PWMB_SR1	PWMB_SR2	PWMB_EGR
FED8H	PWMA_CCR2L	PWMA_CCR3H	PWMA_CCR3L	PWMA_CCR4H	PWMA_CCR4L	PWMA_BKR	PWMA_DTR	PWMA_OISR
FED0H	PWMA_PSCRH	PWMA_PSCRL	PWMA_ARRH	PWMA_ARRL	PWMA_RCR	PWMA_CCR1H	PWMA_CCR1L	PWMA_CCR2H
FEC8H	PWMA_CCMR1	PWMA_CCMR2	PWMA_CCMR3	PWMA_CCMR4	PWMA_CCER1	PWMA_CCER2	PWMA_CNTRH	PWMA_CNTRL
FEC0H	PWMA_CR1	PWMA_CR2	PWMA_SMCRL	PWMA_ETR	PWMA_IER	PWMA_SR1	PWMA_SR2	PWMA_EGR
FEB0H	PWMA_ETRPS	PWMA_ENO	PWMA_PS	PWMA_IOAUX	PWMB_ETRPS	PWMB_ENO	PWMB_PS	PWMB_IOAUX
FEA8H	ADCTIM				T3T4PIN			
FEA0H			TM2PS	TM3PS	TM4PS			
FE88H	I2CMSAUX							
FE80H	I2CCFG	I2CMSCLR	I2CMSST	I2CSLCR	I2CSLST	I2CSLADR	I2CTxD	I2CRxD
FE30H	P0IE	P1IE						
FE28H	P0DR	P1DR	P2DR	P3DR	P4DR	P5DR	P6DR	P7DR
FE20H	P0SR	P1SR	P2SR	P3SR	P4SR	P5SR	P6SR	P7SR
FE18H	P0NCS	P1NCS	P2NCS	P3NCS	P4NCS	P5NCS	P6NCS	P7NCS
FE10H	P0PU	P1PU	P2PU	P3PU	P4PU	P5PU	P6PU	P7PU
FE08H	SPFUNC	RSTFLAG						
FE00H	CKSEL	CLKDIV	HIRCCR	XOSCCR	IRC32KCR	MCLKOCR	IRCD	IRC48MCR
FD40H	P0WKUE	P1WKUE	P2WKUE	P3WKUE	P4WKUE	P5WKUE	P6WKUE	P7WKUE
FD30H	P0IM1	P1IM1	P2IM1	P3IM1	P4IM1	P5IM1	P6IM1	P7IM1
FD20H	P0IM0	P1IM0	P2IM0	P3IM0	P4IM0	P5IM0	P6IM0	P7IM0
FD10H	P0INTF	P1INTF	P2INTF	P3INTF	P4INTF	P5INTF	P6INTF	P7INTF
FD00H	P0INTE	P1INTE	P2INTE	P3INTE	P4INTE	P5INTE	P6INTE	P7INTE
FCF0H	MD3	MD2	MD1	MD0	MD5	MD4	ARCON	OPCON

8.4 STC8H3K64S2 family

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8H	P7							RSTCFG
F0H	B					IAP_TPS		
E8H	P6						IP3H	AUXINTIF
E0H	ACC	P7M1	P7M0	DPS	DPL1	DPH1	CMPCR1	CMPCR2
D8H							ADCCFG	IP3
D0H	PSW	T4T3M	T4H	T4L	T3H	T3L	T2H	T2L
C8H	P5	P5M1	P5M0	P6M1	P6M0	SPSTAT	SPCTL	SPDAT
C0H	P4	WDT_CONTR	IAP_DATA	IAP_ADDRH	IAP_ADDRL	IAP_CMD	IAP_TRIG	IAP_CONTR
B8H	IP	SADEN	P_SW2		ADC_CONTR	ADC_RES	ADC_RESL	
B0H	P3	P3M1	P3M0	P4M1	P4M0	IP2	IP2H	IPH
A8H	IE	SADDR	WKTCL	WKTCH			TA	IE2
A0H	P2	BUS_SPEED	P_SW1					
98H	SCON	SBUF	S2CON	S2BUF		IRCBAND	LIRTRIM	IRTRIM
90H	P1	P1M1	P1M0	P0M1	P0M0	P2M1	P2M0	
88H	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR	INTCLKO
80H	P0	SP	DPL	DPH				PCON

Note: The register address can be bit-addressable only if it is divisible by 8, and it cannot be bit-addressable if it is not divisible by 8.

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
FEF8H	PWMB_CCR6L	PWMB_CCR7H	PWMB_CCR7L	PWMB_CCR8H	PWMB_CCR8L	PWMB_BKR	PWMB_DTR	PWMB_OISR
FEF0H	PWMB_PSCRH	PWMB_PSCRL	PWMB_ARRH	PWMB_ARRL	PWMB_RCR	PWMB_CCR5H	PWMB_CCR5L	PWMB_CCR6H
FEE8H	PWMB_CCMR1	PWMB_CCMR2	PWMB_CCMR3	PWMB_CCMR4	PWMB_CCER1	PWMB_CCER2	PWMB_CNTRH	PWMB_CNTRL
FEE0H	PWMB_CR1	PWMB_CR2	PWMB_SMC	PWMB_ETR	PWMB_IER	PWMB_SR1	PWMB_SR2	PWMB_EGR
FED8H	PWMA_CCR2L	PWMA_CCR3H	PWMA_CCR3L	PWMA_CCR4H	PWMA_CCR4L	PWMA_BKR	PWMA_DTR	PWMA_OISR
FED0H	PWMA_PSCRH	PWMA_PSCRL	PWMA_ARRH	PWMA_ARRL	PWMA_RCR	PWMA_CCR1H	PWMA_CCR1L	PWMA_CCR2H
FEC8H	PWMA_CCMR1	PWMA_CCMR2	PWMA_CCMR3	PWMA_CCMR4	PWMA_CCER1	PWMA_CCER2	PWMA_CNTRH	PWMA_CNTRL
FEC0H	PWMA_CR1	PWMA_CR2	PWMA_SMC	PWMA_ETR	PWMA_IER	PWMA_SR1	PWMA_SR2	PWMA_EGR
FEB0H	PWMA_ETRPS	PWMA_ENO	PWMA_PS	PWMA_IOAUX	PWMB_ETRPS	PWMB_ENO	PWMB_PS	PWMB_IOAUX
FEA8H	ADCTIM				T3T4PIN			
FEA0H			TM2PS	TM3PS	TM4PS			
FE88H	I2CMSAUX							
FE80H	I2CCFG	I2CMSCR	I2CMSST	I2CSLCR	I2CSLST	I2CSLADR	I2CTxD	I2CRxD
FE30H	P0IE	P1IE						
FE28H	P0DR	P1DR	P2DR	P3DR	P4DR	P5DR	P6DR	P7DR
FE20H	P0SR	P1SR	P2SR	P3SR	P4SR	P5SR	P6SR	P7SR
FE18H	P0NCS	P1NCS	P2NCS	P3NCS	P4NCS	P5NCS	P6NCS	P7NCS
FE10H	P0PU	P1PU	P2PU	P3PU	P4PU	P5PU	P6PU	P7PU
FE08H	SPFUNC	RSTFLAG						
FE00H	CKSEL	CLKDIV	HIRCCR	XOSCCR	IRC32KCR	MCLKOCR	IRCD	IRC48MCR
FD40H	POWKUE	P1WKUE	P2WKUE	P3WKUE	P4WKUE	P5WKUE	P6WKUE	P7WKUE
FD30H	P0IM1	P1IM1	P2IM1	P3IM1	P4IM1	P5IM1	P6IM1	P7IM1
FD20H	P0IM0	P1IM0	P2IM0	P3IM0	P4IM0	P5IM0	P6IM0	P7IM0
FD10H	P0INTF	P1INTF	P2INTF	P3INTF	P4INTF	P5INTF	P6INTF	P7INTF
FD00H	P0INTE	P1INTE	P2INTE	P3INTE	P4INTE	P5INTE	P6INTE	P7INTE
FCF0H	MD3	MD2	MD1	MD0	MD5	MD4	ARCON	OPCON

8.5 STC8H8K64U-64Pin/48Pin USB family

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8H	P7				USBADR			RSTCFG
F0H	B				USBCON	IAP_TPS		
E8H	P6				USBDAT		IP3H	AUXINTIF
E0H	ACC	P7M1	P7M0	DPS	DPL1	DPH1	CMPCR1	CMPCR2
D8H					USBCLK		ADCCFG	IP3
D0H	PSW	T4T3M	T4H	T4L	T3H	T3L	T2H	T2L
C8H	P5	P5M1	P5M0	P6M1	P6M0	SPSTAT	SPCTL	SPDAT
C0H	P4	WDT_CONTR	IAP_DATA	IAP_ADDRH	IAP_ADDRL	IAP_CMD	IAP_TRIG	IAP_CONTR
B8H	IP	SADEN	P_SW2		ADC_CONTR	ADC_RES	ADC_RESL	
B0H	P3	P3M1	P3M0	P4M1	P4M0	IP2	IP2H	IPH
A8H	IE	SADDR	WKTCL	WKTCH	S3CON	S3BUF	TA	IE2
A0H	P2	BUS_SPEED	P_SW1					
98H	SCON	SBUF	S2CON	S2BUF		IRCBAND	LIRTRIM	IRTRIM
90H	P1	P1M1	P1M0	P0M1	P0M0	P2M1	P2M0	
88H	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR	INTCLKO
80H	P0	SP	DPL	DPH	S4CON	S4BUF		PCON

Note: The register address can be bit-addressable only if it is divisible by 8, and it cannot be bit-addressable if it is not divisible by 8.

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
FEF8H	PWMB_CCR6L	PWMB_CCR7H	PWMB_CCR7L	PWMB_CCR8H	PWMB_CCR8L	PWMB_BKR	PWMB_DTR	PWMB_OISR
FEF0H	PWMB_PSCRH	PWMB_PSCRL	PWMB_ARRH	PWMB_ARRL	PWMB_RCR	PWMB_CCR5H	PWMB_CCR5L	PWMB_CCR6H
FEE8H	PWMB_CCMR1	PWMB_CCMR2	PWMB_CCMR3	PWMB_CCMR4	PWMB_CCER1	PWMB_CCER2	PWMB_CNTRH	PWMB_CNTRL
FEE0H	PWMB_CR1	PWMB_CR2	PWMB_SMCRL	PWMB_ETR	PWMB_IER	PWMB_SR1	PWMB_SR2	PWMB_EGR
FED8H	PWMA_CCR2L	PWMA_CCR3H	PWMA_CCR3L	PWMA_CCR4H	PWMA_CCR4L	PWMA_BKR	PWMA_DTR	PWMA_OISR
FED0H	PWMA_PSCRH	PWMA_PSCRL	PWMA_ARRH	PWMA_ARRL	PWMA_RCR	PWMA_CCR1H	PWMA_CCR1L	PWMA_CCR2H
FEC8H	PWMA_CCMR1	PWMA_CCMR2	PWMA_CCMR3	PWMA_CCMR4	PWMA_CCER1	PWMA_CCER2	PWMA_CNTRH	PWMA_CNTRL
FEC0H	PWMA_CR1	PWMA_CR2	PWMA_SMCRL	PWMA_ETR	PWMA_IER	PWMA_SR1	PWMA_SR2	PWMA_EGR
FEB0H	PWMA_ETRPS	PWMA_ENO	PWMA_PS	PWMA_IOAUX	PWMB_ETRPS	PWMB_ENO	PWMB_PS	PWMB_IOAUX
FEA8H	ADCTIM				T3T4PIN			
FEA0H			TM2PS	TM3PS	TM4PS			
FE88H	I2CMSAUX							
FE80H	I2CCFG	I2CMSCLR	I2CMSST	I2CSLCR	I2CSLST	I2CSLADR	I2CTxD	I2CRxD
FE30H	P0IE	P1IE						
FE28H	P0DR	P1DR	P2DR	P3DR	P4DR	P5DR	P6DR	P7DR
FE20H	P0SR	P1SR	P2SR	P3SR	P4SR	P5SR	P6SR	P7SR
FE18H	P0NCS	P1NCS	P2NCS	P3NCS	P4NCS	P5NCS	P6NCS	P7NCS
FE10H	P0PU	P1PU	P2PU	P3PU	P4PU	P5PU	P6PU	P7PU
FE08H	SPFUNC	RSTFLAG						
FE00H	CKSEL	CLKDIV	HIRCCR	XOSCCR	IRC32KCR	MCLKOCR	IRCDDB	IRC48MCR
FCF0H	MD3	MD2	MD1	MD0	MD5	MD4	ARCON	OPCON

8.6 STC8H2K64T family

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8H								RSTCFG
F0H	B					IAP_TPS		
E8H							IP3H	AUXINTIF
E0H	ACC			DPS	DPL1	DPH1	CMPCR1	CMPCR2
D8H							ADCCFG	IP3
D0H	PSW	T4T3M	T4H	T4L	T3H	T3L	T2H	T2L
C8H	P5	P5M1	P5M0			SPSTAT	SPCTL	SPDAT
C0H	P4	WDT_CONTR	IAP_DATA	IAP_ADDRH	IAP_ADDRL	IAP_CMD	IAP_TRIG	IAP_CONTR
B8H	IP	SADEN	P_SW2		ADC_CONTR	ADC_RES	ADC_RESL	
B0H	P3	P3M1	P3M0	P4M1	P4M0	IP2	IP2H	IPH
A8H	IE	SADDR	WKTCL	WKTCH	S3CON	S3BUF	TA	IE2
A0H	P2	BUS_SPEED	P_SW1					
98H	SCON	SBUF	S2CON	S2BUF		IRCBAND	LIRTRIM	IRTRIM
90H	P1	P1M1	P1M0	P0M1	P0M0	P2M1	P2M0	
88H	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR	INTCLKO
80H	P0	SP	DPL	DPH	S4CON	S4BUF		PCON

Note: The register address can be bit-addressable only if it is divisible by 8, and it cannot be bit-addressable if it is not divisible by 8.

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
FEF8H	PWMB_CCR6L	PWMB_CCR7H	PWMB_CCR7L	PWMB_CCR8H	PWMB_CCR8L	PWMB_BKR	PWMB_DTR	PWMB_OISR
FEF0H	PWMB_PSCRH	PWMB_PSCRL	PWMB_ARRH	PWMB_ARRL	PWMB_RCR	PWMB_CCR5H	PWMB_CCR5L	PWMB_CCR6H
FEE8H	PWMB_CCMR1	PWMB_CCMR2	PWMB_CCMR3	PWMB_CCMR4	PWMB_CCER1	PWMB_CCER2	PWMB_CNTRH	PWMB_CNTRL
FEE0H	PWMB_CR1	PWMB_CR2	PWMB_SMC	PWMB_ETR	PWMB_IER	PWMB_SR1	PWMB_SR2	PWMB_EGR
FED8H	PWMA_CCR2L	PWMA_CCR3H	PWMA_CCR3L	PWMA_CCR4H	PWMA_CCR4L	PWMA_BKR	PWMA_DTR	PWMA_OISR
FED0H	PWMA_PSCRH	PWMA_PSCRL	PWMA_ARRH	PWMA_ARRL	PWMA_RCR	PWMA_CCR1H	PWMA_CCR1L	PWMA_CCR2H
FEC8H	PWMA_CCMR1	PWMA_CCMR2	PWMA_CCMR3	PWMA_CCMR4	PWMA_CCER1	PWMA_CCER2	PWMA_CNTRH	PWMA_CNTRL
FEC0H	PWMA_CR1	PWMA_CR2	PWMA_SMC	PWMA_ETR	PWMA_IER	PWMA_SR1	PWMA_SR2	PWMA_EGR
FEB0H	PWMA_ETRPS	PWMA_ENO	PWMA_PS	PWMA_IOAUX	PWMB_ETRPS	PWMB_ENO	PWMB_PS	PWMB_IOAUX
FEA8H	ADCTIM				T3T4PIN			
FEA0H			TM2PS	TM3PS	TM4PS			
FE88H	I2CMSAUX							
FE80H	I2CCFG	I2CMSCR	I2CMSST	I2CSLCR	I2CSLST	I2CSLADR	I2CTxD	I2CRxD
FE70H	YEAR	MONTH	DAY	HOURL	MIN	SEC	SSEC	
FE68H	INIYEAR	INIMONTH	INIDAY	INIHOUR	INIMIN	INISEC	INISSEC	
FE60H	RTCCR	RTCCFG	RTCEN	RTCIF	ALAHOUR	ALAMIN	ALASEC	ALASSEC
FE30H	P0IE	P1IE						
FE28H	P0DR	P1DR	P2DR	P3DR	P4DR	P5DR		
FE20H	P0SR	P1SR	P2SR	P3SR	P4SR	P5SR		
FE18H	P0NCS	P1NCS	P2NCS	P3NCS	P4NCS	P5NCS		
FE10H	P0PU	P1PU	P2PU	P3PU	P4PU	P5PU		
FE08H	X32KCR							
FE00H	CKSEL	CLKDIV	HIRCCR	XOSCCR	IRC32KCR	MCLKOCR	IRCD	
FD30H	P0IM1	P1IM1	P2IM1	P3IM1	P4IM1	P5IM1		
FD20H	P0IM0	P1IM0	P2IM0	P3IM0	P4IM0	P5IM0		
FD10H	P0INTF	P1INTF	P2INTF	P3INTF	P4INTF	P5INTF		
FD00H	P0INTE	P1INTE	P2INTE	P3INTE	P4INTE	P5INTE		
FCF0H	MD3	MD2	MD1	MD0	MD5	MD4	ARCON	OPCON
FB68H	TSTH12H	TSTH12L	TSTH13H	TSTH13L	TSTH14H	TSTH14L	TSTH15H	TSTH15L
FB60H	TSTH08H	TSTH08L	TSTH09H	TSTH09L	TSTH10H	TSTH10L	TSTH11H	TSTH11L
FB58H	TSTH04H	TSTH04L	TSTH05H	TSTH05L	TSTH06H	TSTH06L	TSTH07H	TSTH07L
FB50H	TSTH00H	TSTH00L	TSTH01H	TSTH01L	TSTH02H	TSTH02L	TSTH03H	TSTH03L
FB48H	TSRT	TSRATH	TSRATL					
FB40H	TSCHEN1	TSCHEN2	TSFCG1	TSFCG2	TSWUTC	TSCTRL	TSSTA1	TSSTA2
FB28H	COM0_DC_H	COM1_DC_H	COM2_DC_H	COM3_DC_H	COM4_DC_H	COM5_DC_H	COM6_DC_H	COM7_DC_H
FB20H	COM0_DC_L	COM1_DC_L	COM2_DC_L	COM3_DC_L	COM4_DC_L	COM5_DC_L	COM6_DC_L	COM7_DC_L
FB18H	COM0_DA_H	COM1_DA_H	COM2_DA_H	COM3DA_H	COM4_DA_H	COM5_DA_H	COM6_DA_H	COM7_DA_H
FB10H	COM0_DA_L	COM1_DA_L	COM2_DA_L	COM3DA_L	COM4_DA_L	COM5_DA_L	COM6_DA_L	COM7_DA_L
FB00H	COMEN	SEGENL	SEGENH	LEDCTRL	LEDCKS			

8.7 STC8H4K64TLR family

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8H								RSTCFG
F0H	B					IAP_TPS		
E8H							IP3H	AUXINTIF
E0H	ACC			DPS	DPL1	DPH1	CMPCR1	CMPCR2
D8H							ADCCFG	IP3
D0H	PSW	T4T3M	T4H	T4L	T3H	T3L	T2H	T2L
C8H	P5	P5M1	P5M0			SPSTAT	SPCTL	SPDAT
C0H	P4	WDT_CONTR	IAP_DATA	IAP_ADDRH	IAP_ADDRL	IAP_CMD	IAP_TRIG	IAP_CONTR
B8H	IP	SADEN	P_SW2		ADC_CONTR	ADC_RES	ADC_RESL	
B0H	P3	P3M1	P3M0	P4M1	P4M0	IP2	IP2H	IPH
A8H	IE	SADDR	WKTCL	WKTCH	S3CON	S3BUF	TA	IE2
A0H	P2	BUS_SPEED	P_SW1					
98H	SCON	SBUF	S2CON	S2BUF		IRCBAND	LIRTRIM	IRTRIM
90H	P1	P1M1	P1M0	P0M1	P0M0	P2M1	P2M0	
88H	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR	INTCLKO
80H	P0	SP	DPL	DPH	S4CON	S4BUF		PCON

↑
Bit addressable

Not bit addressable

Note: The register address can be bit-addressable only if it is divisible by 8, and it cannot be bit-addressable if it is not divisible by 8.

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
FEF8H	PWMB_CCR6L	PWMB_CCR7H	PWMB_CCR7L	PWMB_CCR8H	PWMB_CCR8L	PWMB_BKR	PWMB_DTR	PWMB_OISR
FEF0H	PWMB_PSCRH	PWMB_PSCRL	PWMB_ARRH	PWMB_ARRL	PWMB_RCR	PWMB_CCR5H	PWMB_CCR5L	PWMB_CCR6H
FEE8H	PWMB_CCMR1	PWMB_CCMR2	PWMB_CCMR3	PWMB_CCMR4	PWMB_CCER1	PWMB_CCER2	PWMB_CNTRH	PWMB_CNTRL
FEE0H	PWMB_CR1	PWMB_CR2	PWMB_SMC	PWMB_ETR	PWMB_IER	PWMB_SR1	PWMB_SR2	PWMB_EGR
FED8H	PWMA_CCR2L	PWMA_CCR3H	PWMA_CCR3L	PWMA_CCR4H	PWMA_CCR4L	PWMA_BKR	PWMA_DTR	PWMA_OISR
FED0H	PWMA_PSCRH	PWMA_PSCRL	PWMA_ARRH	PWMA_ARRL	PWMA_RCR	PWMA_CCR1H	PWMA_CCR1L	PWMA_CCR2H
FEC8H	PWMA_CCMR1	PWMA_CCMR2	PWMA_CCMR3	PWMA_CCMR4	PWMA_CCER1	PWMA_CCER2	PWMA_CNTRH	PWMA_CNTRL
FEC0H	PWMA_CR1	PWMA_CR2	PWMA_SMC	PWMA_ETR	PWMA_IER	PWMA_SR1	PWMA_SR2	PWMA_EGR
FEB0H	PWMA_ETRPS	PWMA_ENO	PWMA_PS	PWMA_IOAUX	PWMB_ETRPS	PWMB_ENO	PWMB_PS	PWMB_IOAUX
FEA8H	ADCTIM				T3T4PIN	ADCEXCFG	CMPEXCFG	
FEA0H			TM2PS	TM3PS	TM4PS			
FE88H	I2CMSAUX							
FE80H	I2CCFG	I2CMSCR	I2CMSST	I2CSLCR	I2CSLST	I2CSLADR	I2CTxD	I2CRxD
FE70H	YEAR	MONTH	DAY	HOUR	MIN	SEC	SSEC	
FE68H	INIYEAR	INIMONTH	INIDAY	INIHOURL	INIMIN	INISEC	INISSEC	
FE60H	RTCCR	RTCCFG	RTCIE	RTCF	ALAHOUR	ALAMIN	ALASEC	ALASSEC
FE50H	LCMIFCFG	LCMIFCFG2	LCMIFCR	LCMIFSTA	LCMIFDATH	LCMIFDATH		
FE30H	P0IE	P1IE				P5IE		
FE28H	P0DR	P1DR	P2DR	P3DR	P4DR	P5DR		
FE20H	P0SR	P1SR	P2SR	P3SR	P4SR	P5SR		
FE18H	P0NCS	P1NCS	P2NCS	P3NCS	P4NCS	P5NCS		
FE10H	P0PU	P1PU	P2PU	P3PU	P4PU	P5PU		
FE08H	X32KCR							
FE00H	CKSEL	CLKDIV	HIRCCR	XOSCCR	IRC32KCR	MCLKOCR	IRCDDB	
FD60H	PINPL	PINIPH						
FD40H	P0WKUE	P1WKUE	P2WKUE	P3WKUE	P4WKUE	P5WKUE		
FD30H	P0IM1	P1IM1	P2IM1	P3IM1	P4IM1	P5IM1		
FD20H	P0IM0	P1IM0	P2IM0	P3IM0	P4IM0	P5IM0		
FD10H	P0INTF	P1INTF	P2INTF	P3INTF	P4INTF	P5INTF		
FD00H	P0INTE	P1INTE	P2INTE	P3INTE	P4INTE	P5INTE		
FCF0H	MD3	MD2	MD1	MD0	MD5	MD4	ARCON	OPCON
FB68H	TSTH12H	TSTH12L	TSTH13H	TSTH13L	TSTH14H	TSTH14L	TSTH15H	TSTH15L
FB60H	TSTH08H	TSTH08L	TSTH09H	TSTH09L	TSTH10H	TSTH10L	TSTH11H	TSTH11L
FB58H	TSTH04H	TSTH04L	TSTH05H	TSTH05L	TSTH06H	TSTH06L	TSTH07H	TSTH07L
FB50H	TSTH00H	TSTH00L	TSTH01H	TSTH01L	TSTH02H	TSTH02L	TSTH03H	TSTH03L
FB48H	TSRT	TSDATH	TSDATL					
FB40H	TSCHEN1	TSCHEN2	TSCFG1	TSCFG2	TSWUTC	TSCTRL	TSSTA1	TSSTA2
FB28H	COM0_DC_H	COM1_DC_H	COM2_DC_H	COM3_DC_H	COM4_DC_H	COM5_DC_H	COM6_DC_H	COM7_DC_H
FB20H	COM0_DC_L	COM1_DC_L	COM2_DC_L	COM3_DC_L	COM4_DC_L	COM5_DC_L	COM6_DC_L	COM7_DC_L
FB18H	COM0_DA_H	COM1_DA_H	COM2_DA_H	COM3DA_H	COM4_DA_H	COM5_DA_H	COM6_DA_H	COM7_DA_H
FB10H	COM0_DA_L	COM1_DA_L	COM2_DA_L	COM3DA_L	COM4_DA_L	COM5_DA_L	COM6_DA_L	COM7_DA_L
FB00H	COMEN	SEGENL	SEGENH	LEDCTRL	LEDCKS			
FA78H	DMA_LCM_RXAL							
FA70H	DMA_LCM_CFG	DMA_LCM_CR	DMA_LCM_STA	DMA_LCM_AMT	DMA_LCM_DONE	DMA_LCM_TXAH	DMA_LCM_TXAL	DMA_LCM_RXAH
FA68H	DMA_UR4R_CFG	DMA_UR4R_CR	DMA_UR4R_STA	DMA_UR4R_AMT	DMA_UR4R_DONE	DMA_UR4R_RXAH	DMA_UR4R_RXAL	
FA60H	DMA_UR4T_CFG	DMA_UR4T_CR	DMA_UR4T_STA	DMA_UR4T_AMT	DMA_UR4T_DONE	DMA_UR4T_TXAH	DMA_UR4T_TXAL	
FA58H	DMA_UR3R_CFG	DMA_UR3R_CR	DMA_UR3R_STA	DMA_UR3R_AMT	DMA_UR3R_DONE	DMA_UR3R_RXAH	DMA_UR3R_RXAL	
FA50H	DMA_UR3T_CFG	DMA_UR3T_CR	DMA_UR3T_STA	DMA_UR3T_AMT	DMA_UR3T_DONE	DMA_UR3T_TXAH	DMA_UR3T_TXAL	

STC8H Series Manual

FA48H	DMA_UR2R_CFG	DMA_UR2R_CR	DMA_UR2R_STA	DMA_UR2R_AMT	DMA_UR2R_DONE	DMA_UR2R_RXAH	DMA_UR2R_RXAL	
FA40H	DMA_UR2T_CFG	DMA_UR2T_CR	DMA_UR2T_STA	DMA_UR2T_AMT	DMA_UR2T_DONE	DMA_UR2T_TXAH	DMA_UR2T_TXAL	
FA38H	DMA_UR1R_CFG	DMA_UR1R_CR	DMA_UR1R_STA	DMA_UR1R_AMT	DMA_UR1R_DONE	DMA_UR1R_RXAH	DMA_UR1R_RXAL	
FA30H	DMA_UR1T_CFG	DMA_UR1T_CR	DMA_UR1T_STA	DMA_UR1T_AMT	DMA_UR1T_DONE	DMA_UR1T_TXAH	DMA_UR1T_TXAL	
FA28H	DMA_SPI_RXAL	DMA_SPI_CFG2						
FA20H	DMA_SPI_CFG	DMA_SPI_CR	DMA_SPI_STA	DMA_SPI_AMT	DMA_SPI_DONE	DMA_SPI_TXAH	DMA_SPI_TXAL	DMA_SPI_RXAH
FA18H	DMA_ADC_RXAL	DMA_ADC_CFG2	DMA_ADC_CHSW0	DMA_ADC_CHSW1				
FA10H	DMA_ADC_CFG	DMA_ADC_CR	DMA_ADC_STA					DMA_ADC_RXAH
FA08H	DMA_M2M_RXAL							
FA00H	DMA_M2M_CFG	DMA_M2M_CR	DMA_M2M_STA	DMA_M2M_AMT	DMA_M2M_DONE	DMA_M2M_TXAH	DMA_M2M_TXAL	DMA_M2M_RXAH

STC MCU

8.8 STC8H4K64TLCD family

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8H	P7							RSTCFG
F0H	B					IAP_TPS		
E8H	P6						IP3H	AUXINTIF
E0H	ACC	P7M1	P7M0	DPS	DPL1	DPH1	CMPCR1	CMPCR2
D8H							ADCCFG	IP3
D0H	PSW	T4T3M	T4H	T4L	T3H	T3L	T2H	T2L
C8H	P5	P5M1	P5M0	P6M1	P6M0	SPSTAT	SPCTL	SPDAT
C0H	P4	WDT_CONTR	IAP_DATA	IAP_ADDRH	IAP_ADDRL	IAP_CMD	IAP_TRIG	IAP_CONTR
B8H	IP	SADEN	P_SW2		ADC_CONTR	ADC_RES	ADC_RESL	
B0H	P3	P3M1	P3M0	P4M1	P4M0	IP2	IP2H	IPH
A8H	IE	SADDR	WKTCL	WKTCH	S3CON	S3BUF	TA	IE2
A0H	P2	BUS_SPEED	P_SW1					
98H	SCON	SBUF	S2CON	S2BUF		IRCBAND	LIRTRIM	IRTRIM
90H	P1	P1M1	P1M0	P0M1	P0M0	P2M1	P2M0	
88H	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR	INTCLKO
80H	P0	SP	DPL	DPH	S4CON	S4BUF		PCON

Note: The register address can be bit-addressable only if it is divisible by 8, and it cannot be bit-addressable if it is not divisible by 8.

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
FEF8H	PWMB_CCR6L	PWMB_CCR7H	PWMB_CCR7L	PWMB_CCR8H	PWMB_CCR8L	PWMB_BKR	PWMB_DTR	PWMB_OISR
FEF0H	PWMB_PSCRH	PWMB_PSCRL	PWMB_ARRH	PWMB_ARRL	PWMB_RCR	PWMB_CCR5H	PWMB_CCR5L	PWMB_CCR6H
FEE8H	PWMB_CCMR1	PWMB_CCMR2	PWMB_CCMR3	PWMB_CCMR4	PWMB_CCER1	PWMB_CCER2	PWMB_CNTRH	PWMB_CNTRL
FEE0H	PWMB_CR1	PWMB_CR2	PWMB_SMC	PWMB_ETR	PWMB_IER	PWMB_SR1	PWMB_SR2	PWMB_EGR
FED8H	PWMA_CCR2L	PWMA_CCR3H	PWMA_CCR3L	PWMA_CCR4H	PWMA_CCR4L	PWMA_BKR	PWMA_DTR	PWMA_OISR
FED0H	PWMA_PSCRH	PWMA_PSCRL	PWMA_ARRH	PWMA_ARRL	PWMA_RCR	PWMA_CCR1H	PWMA_CCR1L	PWMA_CCR2H
FEC8H	PWMA_CCMR1	PWMA_CCMR2	PWMA_CCMR3	PWMA_CCMR4	PWMA_CCER1	PWMA_CCER2	PWMA_CNTRH	PWMA_CNTRL
FEC0H	PWMA_CR1	PWMA_CR2	PWMA_SMC	PWMA_ETR	PWMA_IER	PWMA_SR1	PWMA_SR2	PWMA_EGR
FEB0H	PWMA_ETRPS	PWMA_ENO	PWMA_PS	PWMA_IOAUX	PWMB_ETRPS	PWMB_ENO	PWMB_PS	PWMB_IOAUX
FEA8H	ADCTIM				T3T4PIN	ADCEXCFG	CMPEXCFG	
FEA0H			TM2PS	TM3PS	TM4PS			
FE88H	I2CMSAUX							
FE80H	I2CCFG	I2CMSCR	I2CMSST	I2CSLCR	I2CSLST	I2CSLADR	I2CTxD	I2CRxD
FE70H	YEAR	MONTH	DAY	HOURL	MIN	SEC	SSEC	
FE68H	INIYEAR	INIMONTH	INIDAY	INIHOUR	INIMIN	INISEC	INISSEC	
FE60H	RTCCR	RTCCFG	RTCEN	RTCIF	ALAHOUR	ALAMIN	ALASEC	ALASSEC
FE30H	P0IE	P1IE	P2IE	P3IE	P4IE	P5IE	P6IE	P7IE
FE28H	P0DR	P1DR	P2DR	P3DR	P4DR	P5DR	P6DR	P7DR
FE20H	P0SR	P1SR	P2SR	P3SR	P4SR	P5SR	P6SR	P7SR
FE18H	P0NCS	P1NCS	P2NCS	P3NCS	P4NCS	P5NCS	P6NCS	P7NCS
FE10H	P0PU	P1PU	P2PU	P3PU	P4PU	P5PU	P6PU	P7PU
FE08H	X32KCR							
FE00H	CKSEL	CLKDIV	HIRCCR	XOSCCR	IRC32KCR	MCLKOCR	IRCDDB	
FD60H	PINPL	PINPH						
FD40H	P0WKUE	P1WKUE	P2WKUE	P3WKUE	P4WKUE	P5WKUE	P6WKUE	P7WKUE
FD30H	P0IM1	P1IM1	P2IM1	P3IM1	P4IM1	P5IM1	P6IM1	P7IM1
FD20H	P0IM0	P1IM0	P2IM0	P3IM0	P4IM0	P5IM0	P6IM0	P7IM0
FD10H	P0INTF	P1INTF	P2INTF	P3INTF	P4INTF	P5INTF	P6INTF	P7INTF
FD00H	P0INTE	P1INTE	P2INTE	P3INTE	P4INTE	P5INTE	P6INTE	P7INTE
FCF0H	MD3	MD2	MD1	MD0	MD5	MD4	ARCON	OPCON
FBA8H	C3SEGV0	C3SEGV1	C3SEGV2	C3SEGV3	C3SEGV4			
FBA0H	C2SEGV0	C2SEGV1	C2SEGV2	C2SEGV3	C2SEGV4			
FB98H	C1SEGV0	C1SEGV1	C1SEGV2	C1SEGV3	C1SEGV4			
FB90H	C0SEGV0	C0SEGV1	C0SEGV2	C0SEGV3	C0SEGV4			
FB88H	COMON		SEGON1	SEGON2	SEGON3	SEGON4	SEGON5	
FB80H	LCDCFG	LCDCFG2	DBLEN	COMLENL	COMLENM	COMLENH	BLINKRATE	LCDCR
FB68H	TSTH12H	TSTH12L	TSTH13H	TSTH13L	TSTH14H	TSTH14L	TSTH15H	TSTH15L
FB60H	TSTH08H	TSTH08L	TSTH09H	TSTH09L	TSTH10H	TSTH10L	TSTH11H	TSTH11L
FB58H	TSTH04H	TSTH04L	TSTH05H	TSTH05L	TSTH06H	TSTH06L	TSTH07H	TSTH07L
FB50H	TSTH00H	TSTH00L	TSTH01H	TSTH01L	TSTH02H	TSTH02L	TSTH03H	TSTH03L
FB48H	TSRT	TSDATH	TSDATL					
FB40H	TSCHEN1	TSCHEN2	TSCFG1	TSCFG2	TSWUTC	TSCTRL	TSSTA1	TSSTA2
FA78H	DMA_LCM_RXAL							
FA70H	DMA_LCM_CFG	DMA_LCM_CR	DMA_LCM_STA	DMA_LCM_AMT	DMA_LCM_DONE	DMA_LCM_TXAH	DMA_LCM_TXAL	DMA_LCM_RXAH
FA68H	DMA_UR4R_CFG	DMA_UR4R_CR	DMA_UR4R_STA	DMA_UR4R_AMT	DMA_UR4R_DONE	DMA_UR4R_RXAH	DMA_UR4R_RXAL	
FA60H	DMA_UR4T_CFG	DMA_UR4T_CR	DMA_UR4T_STA	DMA_UR4T_AMT	DMA_UR4T_DONE	DMA_UR4T_TXAH	DMA_UR4T_TXAL	
FA58H	DMA_UR3R_CFG	DMA_UR3R_CR	DMA_UR3R_STA	DMA_UR3R_AMT	DMA_UR3R_DONE	DMA_UR3R_RXAH	DMA_UR3R_RXAL	

STC8H Series Manual

FA50H	DMA_UR3T_CFG	DMA_UR3T_CR	DMA_UR3T_STA	DMA_UR3T_AMT	DMA_UR3T_DONE	DMA_UR3T_TXAH	DMA_UR3T_TXAL	
FA48H	DMA_UR2R_CFG	DMA_UR2R_CR	DMA_UR2R_STA	DMA_UR2R_AMT	DMA_UR2R_DONE	DMA_UR2R_RXAH	DMA_UR2R_RXAL	
FA40H	DMA_UR2T_CFG	DMA_UR2T_CR	DMA_UR2T_STA	DMA_UR2T_AMT	DMA_UR2T_DONE	DMA_UR2T_TXAH	DMA_UR2T_TXAL	
FA38H	DMA_UR1R_CFG	DMA_UR1R_CR	DMA_UR1R_STA	DMA_UR1R_AMT	DMA_UR1R_DONE	DMA_UR1R_RXAH	DMA_UR1R_RXAL	
FA30H	DMA_UR1T_CFG	DMA_UR1T_CR	DMA_UR1T_STA	DMA_UR1T_AMT	DMA_UR1T_DONE	DMA_UR1T_TXAH	DMA_UR1T_TXAL	
FA28H	DMA_SPI_RXAL	DMA_SPI_CFG2						
FA20H	DMA_SPI_CFG	DMA_SPI_CR	DMA_SPI_STA	DMA_SPI_AMT	DMA_SPI_DONE	DMA_SPI_TXAH	DMA_SPI_TXAL	DMA_SPI_RXAH
FA18H	DMA_ADC_RXAL	DMA_ADC_CFG2	DMA_ADC_CHSW0	DMA_ADC_CHSW1				
FA10H	DMA_ADC_CFG	DMA_ADC_CR	DMA_ADC_STA					DMA_ADC_RXAH
FA08H	DMA_M2M_RXAL							
FA00H	DMA_M2M_CFG	DMA_M2M_CR	DMA_M2M_STA	DMA_M2M_AMT	DMA_M2M_DONE	DMA_M2M_TXAH	DMA_M2M_TXAL	DMA_M2M_RXAH

STC MCU

8.9 STC8H4K64LCD family

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
F8H	P7							RSTCFG
F0H	B					IAP_TPS		
E8H	P6						IP3H	AUXINTIF
E0H	ACC	P7M1	P7M0	DPS	DPL1	DPH1	CMPCR1	CMPCR2
D8H							ADCCFG	IP3
D0H	PSW	T4T3M	T4H	T4L	T3H	T3L	T2H	T2L
C8H	P5	P5M1	P5M0	P6M1	P6M0	SPSTAT	SPCTL	SPDAT
C0H	P4	WDT_CONTR	IAP_DATA	IAP_ADDRH	IAP_ADDRL	IAP_CMD	IAP_TRIG	IAP_CONTR
B8H	IP	SADEN	P_SW2		ADC_CONTR	ADC_RES	ADC_RESL	
B0H	P3	P3M1	P3M0	P4M1	P4M0	IP2	IP2H	IPH
A8H	IE	SADDR	WKTCL	WKTCH	S3CON	S3BUF	TA	IE2
A0H	P2	BUS_SPEED	P_SW1					
98H	SCON	SBUF	S2CON	S2BUF		IRCBAND	LIRTRIM	IRTRIM
90H	P1	P1M1	P1M0	P0M1	P0M0	P2M1	P2M0	
88H	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR	INTCLKO
80H	P0	SP	DPL	DPH	S4CON	S4BUF		PCON

Note: The register address can be bit-addressable only if it is divisible by 8, and it cannot be bit-addressable if it is not divisible by 8.

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
FEF8H	PWMB_CCR6L	PWMB_CCR7H	PWMB_CCR7L	PWMB_CCR8H	PWMB_CCR8L	PWMB_BKR	PWMB_DTR	PWMB_OISR
FEF0H	PWMB_PSCRH	PWMB_PSCRL	PWMB_ARRH	PWMB_ARRL	PWMB_RCR	PWMB_CCR5H	PWMB_CCR5L	PWMB_CCR6H
FEE8H	PWMB_CCMR1	PWMB_CCMR2	PWMB_CCMR3	PWMB_CCMR4	PWMB_CCER1	PWMB_CCER2	PWMB_CNTRH	PWMB_CNTRL
FEE0H	PWMB_CR1	PWMB_CR2	PWMB_SMC	PWMB_ETR	PWMB_IER	PWMB_SR1	PWMB_SR2	PWMB_EGR
FED8H	PWMA_CCR2L	PWMA_CCR3H	PWMA_CCR3L	PWMA_CCR4H	PWMA_CCR4L	PWMA_BKR	PWMA_DTR	PWMA_OISR
FED0H	PWMA_PSCRH	PWMA_PSCRL	PWMA_ARRH	PWMA_ARRL	PWMA_RCR	PWMA_CCR1H	PWMA_CCR1L	PWMA_CCR2H
FEC8H	PWMA_CCMR1	PWMA_CCMR2	PWMA_CCMR3	PWMA_CCMR4	PWMA_CCER1	PWMA_CCER2	PWMA_CNTRH	PWMA_CNTRL
FEC0H	PWMA_CR1	PWMA_CR2	PWMA_SMC	PWMA_ETR	PWMA_IER	PWMA_SR1	PWMA_SR2	PWMA_EGR
FEB0H	PWMA_ETRPS	PWMA_ENO	PWMA_PS	PWMA_IOAUX	PWMB_ETRPS	PWMB_ENO	PWMB_PS	PWMB_IOAUX
FEA8H	ADCTIM				T3T4PIN	ADCEXCFG	CMPEXCFG	
FEA0H			TM2PS	TM3PS	TM4PS			
FE88H	I2CMSAUX							
FE80H	I2CCFG	I2CMSCR	I2CMSST	I2CSLCL	I2CSLST	I2CSLADR	I2CTxD	I2CRxD
FE70H	YEAR	MONTH	DAY	HOURL	MIN	SEC	SSEC	
FE68H	INIYEAR	INIMONTH	INIDAY	INIHOUR	INIMIN	INISEC	INISSEC	
FE60H	RTCCR	RTCCFG	RTCEN	RTCIF	ALAHOUR	ALAMIN	ALASEC	ALASSEC
FE30H	P0IE	P1IE	P2IE	P3IE	P4IE	P5IE	P6IE	P7IE
FE28H	P0DR	P1DR	P2DR	P3DR	P4DR	P5DR	P6DR	P7DR
FE20H	P0SR	P1SR	P2SR	P3SR	P4SR	P5SR	P6SR	P7SR
FE18H	P0NCS	P1NCS	P2NCS	P3NCS	P4NCS	P5NCS	P6NCS	P7NCS
FE10H	P0PU	P1PU	P2PU	P3PU	P4PU	P5PU	P6PU	P7PU
FE08H	X32KCR							
FE00H	CKSEL	CLKDIV	HIRCCR	XOSCCR	IRC32KCR	MCLKOCR	IRCDDB	
FD60H	PINPL	PINPH						
FD40H	POWKUE	P1WKUE	P2WKUE	P3WKUE	P4WKUE	P5WKUE	P6WKUE	P7WKUE
FD30H	P0IM1	P1IM1	P2IM1	P3IM1	P4IM1	P5IM1	P6IM1	P7IM1
FD20H	P0IM0	P1IM0	P2IM0	P3IM0	P4IM0	P5IM0	P6IM0	P7IM0
FD10H	P0INTF	P1INTF	P2INTF	P3INTF	P4INTF	P5INTF	P6INTF	P7INTF
FD00H	P0INTE	P1INTE	P2INTE	P3INTE	P4INTE	P5INTE	P6INTE	P7INTE
FCF0H	MD3	MD2	MD1	MD0	MD5	MD4	ARCON	OPCON
FBA8H	C3SEGV0	C3SEGV1	C3SEGV2	C3SEGV3	C3SEGV4			
FBA0H	C2SEGV0	C2SEGV1	C2SEGV2	C2SEGV3	C2SEGV4			
FB98H	C1SEGV0	C1SEGV1	C1SEGV2	C1SEGV3	C1SEGV4			
FB90H	C0SEGV0	C0SEGV1	C0SEGV2	C0SEGV3	C0SEGV4			
FB88H	COMON		SEGON1	SEGON2	SEGON3	SEGON4	SEGON5	
FB80H	LCDCFG	LCDCFG2	DBLEN	COMLENL	COMLENM	COMLENH	BLINKRATE	LCDCR
FA78H	DMA_LCM_RXAL							
FA70H	DMA_LCM_CFG	DMA_LCM_CR	DMA_LCM_STA	DMA_LCM_AMT	DMA_LCM_DONE	DMA_LCM_TXAH	DMA_LCM_TXAL	DMA_LCM_RXAH
FA68H	DMA_UR4R_CFG	DMA_UR4R_CR	DMA_UR4R_STA	DMA_UR4R_AMT	DMA_UR4R_DONE	DMA_UR4R_RXAH	DMA_UR4R_RXAL	
FA60H	DMA_UR4T_CFG	DMA_UR4T_CR	DMA_UR4T_STA	DMA_UR4T_AMT	DMA_UR4T_DONE	DMA_UR4T_TXAH	DMA_UR4T_TXAL	
FA58H	DMA_UR3R_CFG	DMA_UR3R_CR	DMA_UR3R_STA	DMA_UR3R_AMT	DMA_UR3R_DONE	DMA_UR3R_RXAH	DMA_UR3R_RXAL	
FA50H	DMA_UR3T_CFG	DMA_UR3T_CR	DMA_UR3T_STA	DMA_UR3T_AMT	DMA_UR3T_DONE	DMA_UR3T_TXAH	DMA_UR3T_TXAL	
FA48H	DMA_UR2R_CFG	DMA_UR2R_CR	DMA_UR2R_STA	DMA_UR2R_AMT	DMA_UR2R_DONE	DMA_UR2R_RXAH	DMA_UR2R_RXAL	
FA40H	DMA_UR2T_CFG	DMA_UR2T_CR	DMA_UR2T_STA	DMA_UR2T_AMT	DMA_UR2T_DONE	DMA_UR2T_TXAH	DMA_UR2T_TXAL	
FA38H	DMA_UR1R_CFG	DMA_UR1R_CR	DMA_UR1R_STA	DMA_UR1R_AMT	DMA_UR1R_DONE	DMA_UR1R_RXAH	DMA_UR1R_RXAL	
FA30H	DMA_UR1T_CFG	DMA_UR1T_CR	DMA_UR1T_STA	DMA_UR1T_AMT	DMA_UR1T_DONE	DMA_UR1T_TXAH	DMA_UR1T_TXAL	
FA28H	DMA_SPI_RXAL	DMA_SPI_CFG2						

STC8H Series Manual

FA20H	DMA_SPI_CFG	DMA_SPI_CR	DMA_SPI_STA	DMA_SPI_AMT	DMA_SPI_DONE	DMA_SPI_TXAH	DMA_SPI_TXAL	DMA_SPI_RXAH
FA18H	DMA_ADC_RXAL	DMA_ADC_CFG2	DMA_ADC_CHSW0	DMA_ADC_CHSW1				
FA10H	DMA_ADC_CFG	DMA_ADC_CR	DMA_ADC_STA					DMA_ADC_RXAH
FA08H	DMA_M2M_RXAL							
FA00H	DMA_M2M_CFG	DMA_M2M_CR	DMA_M2M_STA	DMA_M2M_AMT	DMA_M2M_DONE	DMA_M2M_TXAH	DMA_M2M_TXAL	DMA_M2M_RXAH

STC MCU

8.10 List of Special Function Registers

Note: The register address can be bit-addressable only if it is divisible by 8, and it cannot be bit-addressable if it is not divisible by 8.

STC8H can be bit-addressable registers: P0 (80H), TCON (88H), P1 (90H), SCON (98H), P2 (A0H), IE (A8H), P3 (B0H), IP (B8H), P4 (C0H), P5 (C8H), PSW (D0H), ACC (E0H), P6 (E8H), B (F0H), P7 (F8H)

Symbol	Description	Address	Bit Address and Symbol								Value after Reset	
			B7	B6	B5	B4	B3	B2	B1	B0		
P0	Port 0	80H	P07	P06	P05	P04	P03	P02	P01	P00	1111,1111	
SP	Stack Pointer	81H									0000,0111	
DPL	Data pointer low byte register	82H									0000,0000	
DPH	Data pointer high byte register	83H									0000,0000	
S4CON	UART 4 control register	84H	S4SM0	S4ST4	S4SM2	S4REN	S4TB8	S4RB8	S4TI	S4RI	0000,0000	
S4BUF	UART 4 data buffer register	85H									0000,0000	
PCON	Power control register	87H	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000	
TCON	Timer 0 and 1 control register	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0000,0000	
TMOD	Timer 0 and 1 mode register	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	0000,0000	
TL0	Timer 0 low byte register	8AH									0000,0000	
TL1	Timer 1 low byte register	8BH									0000,0000	
TH0	Timer 0 high byte register	8CH									0000,0000	
TH1	Timer 1 high byte register	8DH									0000,0000	
AUXR	Auxiliary register 1	8EH	T0x12	T1x12	UART_M0x6	T2R	T2_C/T	T2x12	EXTRAM	S1ST2	0000,0001	
INTCLKO	External interrupt and clock output control register	8FH	-	EX4	EX3	EX2	-	T2CLKO	T1CLKO	T0CLKO	x000,x000	
P1	Port 1	90H	P17	P16	P15	P14	P13	P12	P11	P10	1111,1111	
P1M1	Port 1 mode register 1	91H	P17M1	P16M1	P15M1	P14M1	P13M1	P12M1	P11M1	P10M1	1111,1111	
P1M0	Port 1 mode register 0	92H	P17M0	P16M0	P15M0	P14M0	P13M0	P12M0	P11M0	P10M0	0000,0000	
P0M1	Port 0 mode register 1	93H	P07M1	P06M1	P05M1	P04M1	P03M1	P02M1	P01M1	P00M1	1111,1111	
P0M0	Port 0 mode register 0	94H	P07M0	P06M0	P05M0	P04M0	P03M0	P02M0	P01M0	P00M0	0000,0000	
P2M1	Port 2 mode register 1	95H	P27M1	P26M1	P25M1	P24M1	P23M1	P22M1	P21M1	P20M1	1111,1111	
P2M0	Port 2 mode register 0	96H	P27M0	P26M0	P25M0	P24M0	P23M0	P22M0	P21M0	P20M0	0000,0000	
SCON	UART1 control register	98H	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	0000,0000	
SBUF	UART1 data buffer register	99H									0000,0000	
S2CON	UART2 control register	9AH	S2SM0	-	S2SM2	S2REN	S2TB8	S2RB8	S2TI	S2RI	0x00,0000	
S2BUF	UART2 data buffer registe	9BH									0000,0000	
IRCBAND	IRC band selection detection	9DH	-	-	-	-	-	-	-	SEL	xxxx,xxxn	
LIRTRIM	IRC frequency trim register	9EH	-	-	-	-	-	-	LIRTRIM[1:0]		xxxx,xxnn	
IRTRIM	IRC frequency adjustment register	9FH	IRTRIM[7:0]								nnnn,nnnn	
P2	Port 2	A0H	P27	P26	P25	P24	P23	P22	P21	P20	1111,1111	
BUS_SPEED	Bus speed control register	A1H	RW_S[1:0]						SPEED[2:0]			00xx,x000
P_SW1	Peripheral port switch register 1	A2H	S1_S[1:0]			CCP_S[1:0]		SPI_S[1:0]		0	-	nn00,000x
IE	Interrupt enable register	A8H	EA	ELVD	EADC	ES	ET1	EX1	ET0	EX0	0000,0000	
SADDR	UART1 slave address register	A9H									0000,0000	
WKTCL	Wake-up Timer Control Register Low Byte	AAH									1111,1111	
WKTCH	Wake-up Timer Control Register High Byte	ABH	WKTEN								0111,1111	
S3CON	UART3 control register	ACH	S3SM0	S3ST4	S3SM2	S3REN	S3TB8	S3RB8	S3TI	S3RI	0000,0000	
S3BUF	UART3 data buffer register	ADH									0000,0000	
TA	DPTR Timing control register	AEH									0000,0000	
IE2	Interrupt enable register 2	AFH	EUSB	ET4	ET3	ES4	ES3	ET2	ESPI	ES2	x000,0000	
P3	Port 3	B0H	P37	P36	P35	P34	P33	P32	P31	P30	1111,1111	
P3M1	Port 3 mode register 1	B1H	P37M1	P36M1	P35M1	P34M1	P33M1	P32M1	P31M1	P30M1	1111,1100	
P3M0	Port 3 mode register 0	B2H	P37M0	P36M0	P35M0	P34M0	P33M0	P32M0	P31M0	P30M0	0000,0000	
P4M1	Port 4 mode register 1	B3H	P47M1	P46M1	P45M1	P44M1	P43M1	P42M1	P41M1	P40M1	1111,1111	
P4M0	Port 4 mode register 0	B4H	P47M0	P46M0	P45M0	P44M0	P43M0	P42M0	P41M0	P40M0	0000,0000	
IP2	2nd Interrupt Priority register low byte	B5H	PUSB PTKSU	PI2C	PCMP	PX4	PPWMB	PPWMA	PSPI	PS2	0000,0000	
IP2H	2nd Interrupt Priority register high byte	B6H	PUSBH PTKSUH	PI2CH	PCMPH	PX4H	PPWMBH	PPWMAH	PSPIH	PS2H	0000,0000	
IPH	Interrupt Priority High Byte	B7H	-	PLVDH	PADCH	PSH	PT1H	PX1H	PT0H	PX0H	x000,0000	
IP	Interrupt Priority Low Byte	B8H	-	PLVD	PADC	PS	PT1	PX1	PT0	PX0	x000,0000	
SADEN	UART1 slave address enable register	B9H									0000,0000	
P_SW2	Peripheral port switch register 2	BAH	EAXFR	-	I2C_S[1:0]		CMPO_S	S4_S	S3_S	S2_S	0x00,0000	
ADC_CONTR	ADC control register	BCH	ADC_POWER	ADC_START	ADC_FLAG	ADC_EPWMT	ADC_CHS[3:0]				0000,0000	
ADC_RES	ADC Result High Byte	BDH									0000,0000	
ADC_RESL	ADC Result Low Byte	BEH									0000,0000	
P4	Port 4	C0H	P47	P46	P45	P44	P43	P42	P41	P40	1111,1111	
WDT_CONTR	Watchdog control register	C1H	WDT_FLAG	-	EN_WDT	CLR_WDT	IDL_WDT	WDT_PS[2:0]			0x00,0000	
IAP_DATA	IAP Flash Data Register	C2H									1111,1111	
IAP_ADDRH	IAP Flash Address High Byte	C3H									0000,0000	
IAP_ADDRL	IAP Flash Address Low Byte	C4H									0000,0000	
IAP_CMD	IAP Flash Command Register	C5H	-	-	-	-	-	-	CMD[1:0]		xxxx,xx00	
IAP_TRIG	IAP Flash Trigger register	C6H									0000,0000	
IAP_CONTR	IAP Control Register	C7H	IAPEN	SWBS	SWRST	CMD_FAIL	-	-	-	-	0000,xxxx	
P5	Port 5	C8H	-	-	P55	P54	P53	P52	P51	P50	xx11,1111	
P5M1	Port 5 mode register 1	C9H	-	-	P55M1	P54M1	P53M1	P52M1	P51M1	P50M1	xx11,1111	
P5M0	Port 5 mode register 0	CAH	-	-	P55M0	P54M0	P53M0	P52M0	P51M0	P50M0	xx00,0000	

STC8H Series Manual

P6M1	Port 6 mode register 1	CBH	P67M1	P66M1	P65M1	P64M1	P63M1	P62M1	P61M1	P60M1	1111,1111	
P6M0	Port 6 mode register 0	CCH	P67M0	P66M0	P65M0	P64M0	P63M0	P62M0	P61M0	P60M0	0000,0000	
SPSTAT	SPI Status register	CDH	SPIF	WCOL	-	-	-	-	-	-	00xx,xxxx	
SPCTL	SPI Control Register	CEH	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR[1:0]		0000,0100	
SPDAT	SPI Data Register	CFH										0000,0000
PSW	Program Status Word Register	D0H	CY	AC	F0	RS1	RS0	OV	F1	P	0000,00x0	
T4T3M	Timer4 and Timer 3 Control Register	D1H	T4R	T4_C/T	T4x12	T4CLKO	T3R	T3_C/T	T3x12	T3CLKO	0000,0000	
T4H	Timer 4 high byte register	D2H										0000,0000
T4L	Timer 4 low byte register	D3H										0000,0000
T3H	Timer 3 high byte register	D4H										0000,0000
T3L	Timer 3 low byte register	D5H										0000,0000
T2H	Timer 2 high byte register	D6H										0000,0000
T2L	Timer 2 low byte register	D7H										0000,0000
USBCLK	USB clock control register	DCH	ENCKM	PCKI[1:0]		CRE	TST_USB	TST_PHY	PHYTST[1:0]		0010,0000	
ADCCFG	ADC Configuration Register	DEH	-	-	RESFMT	-	SPEED[3:0]					xx0x,0000
IP3	3nd Interrupt Priority register low byte	DFH	-	-	-	-	-	PRTC	PS4	PS3	xxxx,x000	
ACC	Accumulator	E0H										0000,0000
P7M1	Port 7 mode register 1	E1H	P77M1	P76M1	P75M1	P74M1	P73M1	P72M1	P71M1	P70M1	1111,1111	
P7M0	Port 7 mode register 0	E2H	P77M0	P76M0	P75M0	P74M0	P73M0	P72M0	P71M0	P70M0	0000,0000	
DPS	DPTR Selection Register	E3H	ID1	ID0	TSL	AU1	AU0	-	-	SEL	0000,0xx0	
DPL1	2nd Data pointer low byte	E4H										0000,0000
DPH1	2nd Data pointer high byte	E5H										0000,0000
CMPCR1	Comparator Control Register 1	E6H	CMPEM	CMPIF	PIE	NIE	PIS	NIS	CMPOE	COMPRES	0000,0000	
CMPCR2	Comparator Control Register 2	E7H	INVCMP O	DISFLT	LCDTY[5:0]							0000,0000
P6	Port 6	E8H	P67	P66	P65	P64	P63	P62	P61	P60	1111,1111	
USB DAT	USB Data register	ECH										0000,0000
IP3H	3nd Interrupt Priority Register High Byte	EEH	-	-	-	-	-	PRTCH	PS4H	PS3H	xxxx,x000	
AUXINTIF	Extended External Interrupt Flag Register	EFH	-	INT4IF	INT3IF	INT2IF	-	T4IF	T3IF	T2IF	x000,x000	
B	B register	F0H										0000,0000
USBCON	USB Control Register	F4H	ENUSB	USBRST	PS2M	PUEN	PDEN	DFREC	DP	DM	0000,0000	
IAP_TPS	IAP Waiting Time Control Register	F5H	-	-	IAPTPS[5:0]							xx00,0000
P7	Port 7	F8H	P77	P76	P75	P74	P73	P72	P71	P70	1111,1111	
USBADR	USB Address register	FCH	BUSY	AUTORD	UADR[5:0]							0000,0000
RSTCFG	Reset Configuration Register	FFH	-	ENLVR	-	P54RST	-	-	LVDS[1:0]		xxnx,xxnn	

The following special function registers are extended SFRs whose logical addresses are in the XDATA area. Before access them, the highest bit (EAXFR) of the P_SW2 (BAH) register needs to be set, and they can accessed by using the MOVX A, @DPTR and MOVX @ DPTR, A instructions.

Symbol	Description	Address	Bit Address and Symbol								Value after Reset
			B7	B6	B5	B4	B3	B2	B1	B0	
CKSEL	Clock Select Register	FE00H	-	-	-	-	-	-	MCKSEL[1:0]		xxxx,xx00
CLKDIV	Clock Divide Register	FE01H									nnnn,nnnn
HIRCCR	Internal high-speed oscillator Control Register	FE02H	ENHIRC	-	-	-	-	-	-	HIRCST	1xxx,xxx0
XOSCCR	External Oscillator Control Register	FE03H	ENXOSC	XITYPE	-	-	-	-	-	XOSCST	00xx,xxx0
IRC32KCR	Internal 32K Oscillator Control Register	FE04H	ENIRC32K	-	-	-	-	-	-	IRC32KST	0xxx,xxx0
MCLKOCR	Main Clock Output Control Register	FE05H	MCLKO_S	MCLKODIV[6:0]							0000,0000
IRCDDB	Internal high-speed oscillator debounce control	FE06H	IRCDDB_PAR[7:0]								1000,0000
IRC48MCR	Internal 48M Oscillator Control Register	FE07H	ENIRC48M	-	-	-	-	-	-	IRC48MST	1xxx,xxx0
SPFUNC	Special Function Control Register	FE08H								BKSWR	xxxx,xxx0
RSTFLAG	Reset flag register	FE09H						SWR	ROMOV	EXRST	xxxx,x000
P0PU	P0 Pull-up Resistor Control Register	FE10H	P07PU	P06PU	P05PU	P04PU	P03PU	P02PU	P01PU	P00PU	0000,0000
P1PU	P1 Pull-up Resistor Control Register	FE11H	P17PU	P16PU	P15PU	P14PU	P13PU	P12PU	P11PU	P10PU	0000,0000
P2PU	P2 Pull-up Resistor Control Register	FE12H	P27PU	P26PU	P25PU	P24PU	P23PU	P22PU	P21PU	P20PU	0000,0000
P3PU	P3 Pull-up Resistor Control Register	FE13H	P37PU	P36PU	P35PU	P34PU	P33PU	P32PU	P31PU	P30PU	0000,0000
P4PU	P4 Pull-up Resistor Control Register	FE14H	P47PU	P46PU	P45PU	P44PU	P43PU	P42PU	P41PU	P40PU	0000,0000
P5PU	P5 Pull-up Resistor Control Register	FE15H	-	-	-	P54PU	P53PU	P52PU	P51PU	P50PU	xxx0,0000
P6PU	P6 Pull-up Resistor Control Register	FE16H	P67PU	P66PU	P65PU	P64PU	P63PU	P62PU	P61PU	P60PU	0000,0000
P7PU	P7 Pull-up Resistor Control Register	FE17H	P77PU	P76PU	P75PU	P74PU	P73PU	P72PU	P71PU	P70PU	0000,0000
P0NCS	P0 Schmitt Trigger Control Register	FE18H	P07NCS	P06NCS	P05NCS	P04NCS	P03NCS	P02NCS	P01NCS	P00NCS	0000,0000
P1NCS	P1 Schmitt Trigger Control Register	FE19H	P17NCS	P16NCS	P15NCS	P14NCS	P13NCS	P12NCS	P11NCS	P10NCS	0000,0000
P2NCS	P2 Schmitt Trigger Control Register	FE1AH	P27NCS	P26NCS	P25NCS	P24NCS	P23NCS	P22NCS	P21NCS	P20NCS	0000,0000
P3NCS	P3 Schmitt Trigger Control	FE1BH	P37NCS	P36NCS	P35NCS	P34NCS	P33NCS	P32NCS	P31NCS	P30NCS	0000,0000

STC8H Series Manual

	Register										
P4NCS	P4 Schmitt Trigger Control Register	FE1CH	P47NCS	P46NCS	P45NCS	P44NCS	P43NCS	P42NCS	P41NCS	P40NCS	0000,0000
P5NCS	P5 Schmitt Trigger Control Register	FE1DH	-	-	P55NCS	P54NCS	P53NCS	P52NCS	P51NCS	P50NCS	xx00,0000
P6NCS	P6 Schmitt Trigger Control Register	FE1EH	P67NCS	P66NCS	P65NCS	P64NCS	P63NCS	P62NCS	P61NCS	P60NCS	0000,0000
P7NCS	P7 Schmitt Trigger Control Register	FE1FH	P77NCS	P76NCS	P75NCS	P74NCS	P73NCS	P72NCS	P71NCS	P70NCS	0000,0000
P0SR	P0 Level Shift Rate Register	FE20H	P07SR	P06SR	P05SR	P04SR	P03SR	P02SR	P01SR	P00SR	1111,1111
P1SR	P1 Level Shift Rate Register	FE21H	P17SR	P16SR	P15SR	P14SR	P13SR	P12SR	P11SR	P10SR	1111,1111
P2SR	P2 Level Shift Rate Register	FE22H	P27SR	P26SR	P25SR	P24SR	P23SR	P22SR	P21SR	P20SR	1111,1111
P3SR	P3 Level Shift Rate Register	FE23H	P37SR	P36SR	P35SR	P34SR	P33SR	P32SR	P31SR	P30SR	1111,1111
P4SR	P4 Level Shift Rate Register	FE24H	P47SR	P46SR	P45SR	P44SR	P43SR	P42SR	P41SR	P40SR	1111,1111
P5SR	P5 Level Shift Rate Register	FE25H	-	-	P55SR	P54SR	P53SR	P52SR	P51SR	P50SR	xx11,1111
P6SR	P6 Level Shift Rate Register	FE26H	P57SR	P66SR	P65SR	P64SR	P63SR	P62SR	P61SR	P60SR	1111,1111
P7SR	P7 Level Shift Rate Register	FE27H	P77SR	P76SR	P75SR	P74SR	P73SR	P72SR	P71SR	P70SR	1111,1111
P0DR	P0 Drive Current Control Register	FE28H	P07DR	P06DR	P05DR	P04DR	P03DR	P02DR	P01DR	P00DR	1111,1111
P1DR	P1 Drive Current Control Register	FE29H	P17DR	P16DR	P15DR	P14DR	P13DR	P12DR	P11DR	P10DR	1111,1111
P2DR	P2 Drive Current Control Register	FE2AH	P27DR	P26DR	P25DR	P24DR	P23DR	P22DR	P21DR	P20DR	1111,1111
P3DR	P3 Drive Current Control Register	FE2BH	P37DR	P36DR	P35DR	P34DR	P33DR	P32DR	P31DR	P30DR	1111,1111
P4DR	P4 Drive Current Control Register	FE2CH	P47DR	P46DR	P45DR	P44DR	P43DR	P42DR	P41DR	P40DR	1111,1111
P5DR	P5 Drive Current Control Register	FE2DH	-	-	P55DR	P54DR	P53DR	P52DR	P51DR	P50DR	xx11,1111
P6DR	P6 Drive Current Control Register	FE2EH	P67DR	P66DR	P65DR	P64DR	P63DR	P62DR	P61DR	P60DR	1111,1111
P7DR	P7 Drive Current Control Register	FE2FH	P77DR	P76DR	P75DR	P74DR	P73DR	P72DR	P71DR	P70DR	1111,1111
P0IE	P0 Input Enable Control Register	FE30H	P07IE	P06IE	P05IE	P04IE	P03IE	P02IE	P11IE	P00IE	1111,1111
P1IE	P1 Input Enable Control Register	FE31H	P17IE	P16IE	P15IE	P14IE	P13IE	P12IE	P11IE	P10IE	1111,1111
P2IE	P2 Input Enable Control Register	FE32H	P27IE	P26IE	P25IE	P24IE	P23IE	P22IE	P21IE	P20IE	1111,1111
P3IE	P3 Input Enable Control Register	FE33H	P37IE	P36IE	P35IE	P34IE	P33IE	P32IE	P31IE	P30IE	1111,1111
P4IE	P4 Input Enable Control Register	FE34H	P47IE	P46IE	P45IE	P44IE	P43IE	P42IE	P41IE	P40IE	1111,1111
P5IE	P5 Input Enable Control Register	FE35H	-	-	P55IE	P54IE	P53IE	P52IE	P51IE	P50IE	xx11,1111
P6IE	P6 Input Enable Control Register	FE36H	P67IE	P66IE	P65IE	P64IE	P63IE	P62IE	P61IE	P60IE	1111,1111
P7IE	P7 Input Enable Control Register	FE37H	P77IE	P76IE	P75IE	P74IE	P73IE	P72IE	P71IE	P70IE	1111,1111

LCMIFCFG	LCM Interface Configuration Register	FE50H	LCMIFIE	-	LCMIFIP[1:0]		LCMIFDPS[1:0]		D16_D8	M68_I80	0x00,0000	
LCMIFCFG2	LCM Interface Configuration Register 2	FE51H	-	LCMIFCPS[1:0]		SETUPT[2:0]			HOLDT[1:0]		x000,0000	
LCMIFCR	LCM Interface Control Register	FE52H	ENLCMIF	-	-	-	-	CMD[2:0]			0xxx,x000	
LCMIFSTA	LCM Interface Status Register	FE53H	-	-	-	-	-	-	LCMIFIF	xxxx,xxx0		
LCMIDDATL	LCM interface low byte data	FE54H	LCMIFDAT[7:0]								0000,0000	
LCMIDDATH	LCM interface high byte data	FE55H	LCMIFDAT[15:8]								0000,0000	
RTCCR	RTC Control Register	FE60H	-	-	-	-	-	-	-	RUNRTC	xxxx,xxx0	
RTCCFG	RTC Configuration Register	FE61H	-	-	-	-	-	-	RTCCKS	SETRTC	xxxx,xx00	
RTCIEN	RTC Interrupt Enable Register	FE62H	EALAI	EDAYI	EHOURI	EMINI	ESECI	ESEC2I	ESEC8I	ESEC32I	0000,0000	
RTCIF	RTC Interrupt Request Register	FE63H	ALAIF	DAYIF	HOURIF	MINIF	SECIF	SEC2IF	SEC8IF	SEC32IF	0000,0000	
ALAHOUR	the hour value of the RTC alarm	FE64H	-	-	-						xxx0,0000	
ALAMIN	the minute value of the RTC alarm	FE65H	-	-							xx00,0000	
ALASEC	the second value of the RTC alarm	FE66H	-	-							xx00,0000	
ALASSEC	1/128 second value of RTC alarm	FE67H	-								x000,0000	
INIYEAR	initialization of RTC year	FE68H	-								x000,0000	
INIMONTH	initialization of RTC month	FE69H	-	-	-	-					xxxx,0000	
INIDAY	initialization of RTC day	FE6AH	-	-	-						xxx0,0000	
INIHOUR	initialization of RTC hour	FE6BH	-	-	-						xxx0,0000	
INIMIN	initialization of RTC minute	FE6CH	-	-							xx00,0000	
INISEC	initialization of RTC second	FE6DH	-	-							xx00,0000	
INISSEC	initialization of RTC 1/128 second	FE6EH	-								x000,0000	
YEAR	Year count value of RTC	FE70H	-								x000,0000	
MONTH	Month count value of RTC	FE71H	-	-	-	-					xxxx,0000	
DAY	Day count value of RTC	FE72H	-	-	-						xxx0,0000	
HOURL	Hour count value of RTC	FE73H	-	-	-						xxx0,0000	
MIN	Minute count value of RTC	FE74H	-	-							xx00,0000	
SEC	Second count value of RTC	FE75H	-	-							xx00,0000	
SSEC	1/128 second count value of RTC	FE76H	-								x000,0000	
I2CCFG	I ² C Configuration Register	FE80H	ENI2C	MSSL	MSSPEED[6:1]					0000,0000		
I2CMSCR	I ² C Master Control Register	FE81H	EMSI	-	-	-	MSCMD[3:0]			0xxx,0000		
I2CMSST	I ² C Master Status Register	FE82H	MSBUSY	MSIF	-	-	-	-	MSACKI	MSACKO	00xx,xx00	
I2CSLCR	I ² C Slave Control Register	FE83H	-	ESTAI	ERXI	ETXI	ESTOI	-	-	SLRST	x000,0xx0	
I2CSLST	I ² C Slave Status Register	FE84H	SLBUSY	STAIF	RXIF	TXIF	STOIF	TXING	SLACKI	SLACKO	0000,0000	
I2CSLADR	I ² C Slave Address Register	FE85H	SLADR[6:0]							MA		0000,0000
I2CTXD	I ² C Data Transmission Register	FE86H										0000,0000
I2CRXD	I ² C Data Receive Register	FE87H										0000,0000
I2CMSAUX	I ² C Master Auxiliary Control Register	FE88H	-	-	-	-	-	-	-	WDTA	xxxx,xxx0	
TM2PS	Timer2 Clock Prescaler Register	FEA2H										0000,0000
TM3PS	Timer3 Clock Prescaler Register	FEA3H										0000,0000
TM4PS	Timer4 Clock Prescaler Register	FEA4H										0000,0000
ADCTIM	ADC Timing Control Register	FEA8H	CSSETUP	CSHOLD[1:0]		SMPDUTY[4:0]					0010,1010	
ADCEXCFG	ADC Extended Configuration Register	FEADH	-	-	ADCETRS [1:0]		-	CVTIMESEL[2:0]			xx00,x000	
CMPEXCFG	Comparator Extended Configuration Register	FEAEH	CHYS[1:0]		-	-	-	CMPNS	CMPPS[1:0]		00xx,x000	
T3T4PIN	T3/T4 Select register	FEACH	-	-	-	-	-	-	-	T3T4SEL	xxxx,xxx0	
PWMA_ETRPS	PWMA ETR Select register	FEB0H					BRKAPS		ETRAPS[1:0]		xxxx,x000	
PWMA_ENO	PWMA Output enable control	FEB1H	ENO4N	ENO4P	ENO3N	ENO3P	ENO2N	ENO2P	ENO1N	ENO1P	0000,0000	
PWMA_PS	PWMA Output pin selection register	FEB2H	C4PS[1:0]		C3PS[1:0]		C2PS[1:0]		C1PS[1:0]		0000,0000	
PWMA_IOAUX	PWMAAuxiliary register	FEB3H	AUX4N	AUX4P	AUX3N	AUX3P	AUX2N	AUX2P	AUX1N	AUX1P	0000,0000	
PWMB_ETRPS	PWMB ETR Select register	FEB4H					BRKBPS		ETRBPS[1:0]		xxxx,x000	
PWMB_ENO	PWMB Output enable control	FEB5H	-	ENO8P	-	ENO7P	-	ENO6P	-	ENO5P	x0x0,x0x0	
PWMB_PS	PWMB Output pin selection register	FEB6H	C8PS[1:0]		C7PS[1:0]		C6PS[1:0]		C5PS[1:0]		0000,0000	
PWMB_IOAUX	PWM2 Auxiliary register	FEB7H	-	AUX8P	-	AUX7P	-	AUX6P	-	AUX5P	x0x0,x0x0	
PWMA_CR1	PWMA Control register 1	FEC0H	ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN	0000,0000	
PWMA_CR2	PWMA Control register 2	FEC1H	-	MMS[2:0]			-	COMS	-	CCPC	x000,x0x0	
PWMA_SMCR	PWMA Slave mode control register	FEC2H	MSM	TS[2:0]			-	SMS[2:0]			0000,x000	
PWMA_ETR	PWMA External trigger register	FEC3H	ETP	ECE	ETPS[1:0]		ETF[3:0]				0000,0000	
PWMA_IER	PWMA Interrupt enable register	FEC4H	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE	0000,0000	
PWMA_SR1	PWMA Status register1	FEC5H	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF	0000,0000	
PWMA_SR2	PWMA Status register2	FEC6H	-	-	-	CC4OF	CC3OF	CC2OF	CC1OF	-	xxx0,000x	
PWMA_EGR	PWMA Event occurrence register	FEC7H	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG	0000,0000	
PWMA_CCMR1	PWMA Capture mode register1	FEC8H	OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]		0000,0000	
	PWMA Compare mode register1		IC1F[3:0]			IC1PSC[1:0]		CC1S[1:0]		0000,0000		
PWMA_CCMR2	PWMA Capture mode register2	FEC9H	OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		0000,0000	
	PWMA Compare mode register2		IC2F[3:0]			IC2PSC[1:0]		CC2S[1:0]		0000,0000		
PWMA_CCMR3	PWMA Capture mode register3	FECAH	OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]		0000,0000	
	PWMA Compare mode register3		IC3F[3:0]			IC3PSC[1:0]		CC3S[1:0]		0000,0000		
PWMA_CCMR4	PWMA Capture mode register4	FECBH	OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		0000,0000	
	PWMA Compare mode register4		IC4F[3:0]			IC4PSC[1:0]		CC4S[1:0]		0000,0000		
PWMA_CCER1	PWMA Capture compare enable register 1	FECCH	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E	0000,0000	
PWMA_CCER2	PWMA Capture compare enable register 2	FECDH	CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	0000,0000	
PWMA_CNTRH	PWMA Counter high byte	FECEH	CNT[15:8]								0000,0000	
PWMA_CNTRL	PWMA Counter low byte	FECFH	CNT[7:0]								0000,0000	

STC8H Series Manual

PWMA_PSCRH	PWMA Prescaler high byte	FED0H	PSC[15:8]									0000,0000
PWMA_PSCRL	PWMA Prescaler low byte	FED1H	PSC[7:0]									0000,0000
PWM1_ARRH	PWMA Auto reload register high byte	FED2H	ARR[15:8]									0000,0000
PWMA_ARRL	PWMA Auto reload register low byte	FED3H	ARR[7:0]									0000,0000
PWMA_RCR	PWMA Repeat counter register	FED4H	REP[7:0]									0000,0000
PWMA_CCR1H	PWMA Compare capture register1 high bit	FED5H	CCR1[15:8]									0000,0000
PWMA_CCR1L	PWMA Compare capture register1 low bit	FED6H	CCR1[7:0]									0000,0000
PWMA_CCR2H	PWMA Compare capture register2 high bit	FED7H	CCR2[15:8]									0000,0000
PWMA_CCR2L	PWMA Compare capture register2 low bit	FED8H	CCR2[7:0]									0000,0000
PWMA_CCR3H	PWMA Compare capture register3 high bit	FED9H	CCR3[15:8]									0000,0000
PWMA_CCR3L	PWMA Compare capture register3 low bit	FEDA H	CCR3[7:0]									0000,0000
PWMA_CCR4H	PWMA Compare capture register4 high bit	FEDB H	CCR4[15:8]									0000,0000
PWMA_CCR4L	PWMA Compare capture register4 low bit	FEDC H	CCR4[7:0]									0000,0000
PWMA_BKR	PWMA Brake register	FEDD H	MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		0000,000x	
PWMA_DTR	PWMA Dead zone control register	FEDE H	DTG[7:0]									0000,0000
PWMA_OISR	PWMA Output idle status register	FEDF H	OIS4N	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	0000,0000	
PWMB_CR1	PWMB Control register 1	FEEH	ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN	0000,0000	
PWMB_CR2	PWMB Control register 2	FEE1 H	-	MMS[2:0]			-	COMS	-	CCPC	x000,x0x0	
PWMB_SMCR	PWMB Slave mode control register	FEE2 H	MSM	TS[2:0]			-	SMS[2:0]			0000,x000	
PWMB_ETR	PWMB External trigger register	FEE3 H	ETP	ECE	ETPS[1:0]		ETF[3:0]				0000,0000	
PWMB_IER	PWMB Interrupt enable register	FEE4 H	BIE	TIE	COMIE	CC8IE	CC7IE	CC6IE	CC5IE	UIE	0000,0000	
PWMB_SR1	PWMB Status register1	FEE5 H	BIF	TIF	COMIF	CC8IF	CC7IF	CC6IF	CC5IF	UIF	0000,0000	
PWMB_SR2	PWMB Status register2	FEE6 H	-	-	-	CC8OF	CC7OF	CC6OF	CC5OF	-	xxx0,000x	
PWMB_EGR	PWMB Event occurrence register	FEE7 H	BG	TG	COMG	CC8G	CC7G	CC6G	CC5G	UG	0000,0000	
PWMB_CCMR1	PWMB Capture mode register1	FEE8 H	OC5CE	OC5M[2:0]			OC5PE	OC5FE	CC5S[1:0]		0000,0000	
	PWMB Compare mode register1		IC5F[3:0]			IC5PSC[1:0]		CC5S[1:0]		0000,0000		
PWMB_CCMR2	PWMB Capture mode register2	FEE9 H	OC6CE	OC6M[2:0]			OC6PE	OC6FE	CC6S[1:0]		0000,0000	
	PWMB Compare mode register2		IC6F[3:0]			IC6PSC[1:0]		CC6S[1:0]		0000,0000		
PWMB_CCMR3	PWMB Capture mode register3	FEEA H	OC7CE	OC7M[2:0]			OC7PE	OC7FE	CC7S[1:0]		0000,0000	
	PWMB Compare mode register3		IC7F[3:0]			IC7PSC[1:0]		CC7S[1:0]		0000,0000		
PWMB_CCMR4	PWMB Capture mode register4	FEEB H	OC8CE	OC8M[2:0]			OC8PE	OC8FE	CC8S[1:0]		0000,0000	
	PWMB Compare mode register4		IC8F[3:0]			IC8PSC[1:0]		CC8S[1:0]		0000,0000		
PWMB_CCER1	PWMB Capture compare enable register 1	FEECH	-	-	CC6P	CC6E	-	-	CC5P	CC5E	xx00,xx00	
PWMB_CCER2	PWMB Capture compare enable register 2	FEEDH	-	-	CC8P	CC8E	-	-	CC7P	CC7E	xx00,xx00	
PWMB_CNTRH	PWMB Counter high byte	FEEEH	CNT[15:8]									0000,0000
PWMB_CNTRL	PWMB Counter low byte	FEEFH	CNT[7:0]									0000,0000
PWMB_PSCRH	PWMB Prescaler high byte	FEF0H	PSC[15:8]									0000,0000
PWMB_PSCRL	PWMB Prescaler low byte	FEF1H	PSC[7:0]									0000,0000
PWMB_ARRH	PWMB Auto reload register high byte	FEF2H	ARR[15:8]									0000,0000
PWMB_ARRL	PWMB Auto reload register low byte	FEF3H	ARR[7:0]									0000,0000
PWMB_RCR	PWMB Repeat counter register	FEF4H	REP[7:0]									0000,0000
PWMB_CCR5H	PWMB Compare capture register5 high bit	FEF5H	CCR1[15:8]									0000,0000
PWMB_CCR5L	PWMB Compare capture register5 low bit	FEF6H	CCR1[7:0]									0000,0000
PWMB_CCR6H	PWMB Compare capture register6 high bit	FEF7H	CCR2[15:8]									0000,0000
PWM2_CCR6L	PWMB Compare capture register6 low bit	FEF8H	CCR2[7:0]									0000,0000
PWMB_CCR7H	PWMB Compare capture register7 high bit	FEF9H	CCR3[15:8]									0000,0000
PWMB_CCR7L	PWMB Compare capture register7 low bit	FEFAH	CCR3[7:0]									0000,0000
PWMB_CCR8H	PWMB Compare capture register8 high bit	FEFBH	CCR4[15:8]									0000,0000
PWMB_CCR8L	PWMB Compare capture register8 low bit	FEFCH	CCR4[7:0]									0000,0000
PWMB_BKR	PWMB Brake register	FEFDH	MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]	-	0000,000x	
PWMB_DTR	PWMB2 Dead zone control register	FEFEH	DTG[7:0]									0000,0000
PWMB_OISR	PWMB Output idle status register	FEFFH	-	OIS8	-	OIS7	-	OIS6	-	OIS5	x0x0,x0x0	
MD3	MDU Data register	FCF0H	MD3[7:0]									0000,0000
MD2	MDU Data register	FCF1H	MD2[7:0]									0000,0000
MD1	MDU Data register	FCF2H	MD1[7:0]									0000,0000
MD0	MDU Data register	FCF3H	MD0[7:0]									0000,0000
MD5	MDU Data register	FCF4H	MD5[7:0]									0000,0000
MD4	MDU Data register	FCF5H	MD4[7:0]									0000,0000
ARCON	MDU Mode Control Register	FCF6H	MODE[2:0]			SC[4:0]						0000,0000
OPCON	MDU Operation Control Register	FCF7H	-	MDOV	-	-	-	-	RST	ENOP	0000,0000	
POINTE	P0 Interrupt enable register	FD00H	P07INTE	P06INTE	P05INTE	P04INTE	P03INTE	P02INTE	P01INTE	P00INTE	0000,0000	

STC8H Series Manual

P1INTE	P1 Interrupt enable register	FD01H	P17INTE	P16INTE	P15INTE	P14INTE	P13INTE	P12INTE	P11INTE	P10INTE	0000,0000
P2INTE	P2 Interrupt enable register	FD02H	P27INTE	P26INTE	P25INTE	P24INTE	P23INTE	P22INTE	P21INTE	P20INTE	0000,0000
P3INTE	P3 Interrupt enable register	FD03H	P37INTE	P36INTE	P35INTE	P34INTE	P33INTE	P32INTE	P31INTE	P30INTE	0000,0000
P4INTE	P4 Interrupt enable register	FD04H	P47INTE	P46INTE	P45INTE	P44INTE	P43INTE	P42INTE	P41INTE	P40INTE	0000,0000
P5INTE	P5 Interrupt enable register	FD05H	-	-	P55INTE	P54INTE	P53INTE	P52INTE	P51INTE	P50INTE	xx00,0000
P6INTE	P6 Interrupt enable register	FD06H	P67INTE	P66INTE	P65INTE	P64INTE	P63INTE	P62INTE	P61INTE	P60INTE	0000,0000
P7INTE	P7 Interrupt enable register	FD07H	P77INTE	P76INTE	P75INTE	P74INTE	P73INTE	P72INTE	P71INTE	P70INTE	0000,0000
P0INTF	P0 Interrupt flag register	FD10H	P07INTF	P06INTF	P05INTF	P04INTF	P03INTF	P02INTF	P01INTF	P00INTF	0000,0000
P1INTF	P1 Interrupt flag register	FD11H	P17INTF	P16INTF	P15INTF	P14INTF	P13INTF	P12INTF	P11INTF	P10INTF	0000,0000
P2INTF	P2 Interrupt flag register	FD12H	P27INTF	P26INTF	P25INTF	P24INTF	P23INTF	P22INTF	P21INTF	P20INTF	0000,0000
P3INTF	P3 Interrupt flag register	FD13H	P37INTF	P36INTF	P35INTF	P34INTF	P33INTF	P32INTF	P31INTF	P30INTF	0000,0000
P4INTF	P4 Interrupt flag register	FD14H	P47INTF	P46INTF	P45INTF	P44INTF	P43INTF	P42INTF	P41INTF	P40INTF	0000,0000
P5INTF	P5 Interrupt flag register	FD15H	-	-	P55INTF	P54INTF	P53INTF	P52INTF	P51INTF	P50INTF	xx00,0000
P6INTF	P6 Interrupt flag register	FD16H	P67INTF	P66INTF	P65INTF	P64INTF	P63INTF	P62INTF	P61INTF	P60INTF	0000,0000
P7INTF	P7 Interrupt flag register	FD17H	P77INTF	P76INTF	P75INTF	P74INTF	P73INTF	P72INTF	P71INTF	P70INTF	0000,0000
P0IM0	P0 Interrupt mode register0	FD20H	P07IM0	P06IM0	P05IM0	P04IM0	P03IM0	P02IM0	P01IM0	P00IM0	0000,0000
P1IM0	P1 Interrupt mode register0	FD21H	P17IM0	P16IM0	P15IM0	P14IM0	P13IM0	P12IM0	P11IM0	P10IM0	0000,0000
P2IM0	P2 Interrupt mode register0	FD22H	P27IM0	P26IM0	P25IM0	P24IM0	P23IM0	P22IM0	P21IM0	P20IM0	0000,0000
P3IM0	P3 Interrupt mode register0	FD23H	P37IM0	P36IM0	P35IM0	P34IM0	P33IM0	P32IM0	P31IM0	P30IM0	0000,0000
P4IM0	P4 Interrupt mode register0	FD24H	P47IM0	P46IM0	P45IM0	P44IM0	P43IM0	P42IM0	P41IM0	P40IM0	0000,0000
P5IM0	P5 Interrupt mode register0	FD25H	-	-	P55IM0	P54IM0	P53IM0	P52IM0	P51IM0	P50IM0	xx00,0000
P6IM0	P6 Interrupt mode register0	FD26H	P67IM0	P66IM0	P65IM0	P64IM0	P63IM0	P62IM0	P61IM0	P60IM0	0000,0000
P7IM0	P7 Interrupt mode register0	FD27H	P77IM0	P76IM0	P75IM0	P74IM0	P73IM0	P72IM0	P71IM0	P70IM0	0000,0000
P0IM1	P0 Interrupt mode register1	FD30H	P07IM1	P06IM1	P05IM1	P04IM1	P03IM1	P02IM1	P01IM1	P00IM1	0000,0000
P1IM1	P1 Interrupt mode register1	FD31H	P17IM1	P16IM1	P15IM1	P14IM1	P13IM1	P12IM1	P11IM1	P10IM1	0000,0000
P2IM1	P2 Interrupt mode register1	FD32H	P27IM1	P26IM1	P25IM1	P24IM1	P23IM1	P22IM1	P21IM1	P20IM1	0000,0000
P3IM1	P3 Interrupt mode register1	FD33H	P37IM1	P36IM1	P35IM1	P34IM1	P33IM1	P32IM1	P31IM1	P30IM1	0000,0000
P4IM1	P4 Interrupt mode register1	FD34H	P47IM1	P46IM1	P45IM1	P44IM1	P43IM1	P42IM1	P41IM1	P40IM1	0000,0000
P5IM1	P5 Interrupt mode register1	FD35H	-	-	P55IM1	P54IM1	P53IM1	P52IM1	P51IM1	P50IM1	xx00,0000
P6IM1	P6 Interrupt mode register1	FD36H	P67IM1	P66IM1	P65IM1	P64IM1	P63IM1	P62IM1	P61IM1	P60IM1	0000,0000
P7IM1	P7 Interrupt mode register1	FD37H	P77IM1	P76IM1	P75IM1	P74IM1	P73IM1	P72IM1	P71IM1	P70IM1	0000,0000
P0WKUE	P0 Interrupt Wake-Up Enable Register	FD40H	P07WKUE	P06WKUE	P05WKUE	P04WKUE	P03WKUE	P02WKUE	P01WKUE	P00WKUE	0000,0000
P1WKUE	P1 Interrupt Wake-Up Enable Register	FD41H	P17WKUE	P16WKUE	P15WKUE	P14WKUE	P13WKUE	P12WKUE	P11WKUE	P10WKUE	0000,0000
P2WKUE	P2 Interrupt Wake-Up Enable Register	FD42H	P27WKUE	P26WKUE	P25WKUE	P24WKUE	P23WKUE	P22WKUE	P21WKUE	P20WKUE	0000,0000
P3WKUE	P3 Interrupt Wake-Up Enable Register	FD43H	P37WKUE	P36WKUE	P35WKUE	P34WKUE	P33WKUE	P32WKUE	P31WKUE	P30WKUE	0000,0000
P4WKUE	P4 Interrupt Wake-Up Enable Register	FD44H	P47WKUE	P46WKUE	P45WKUE	P44WKUE	P43WKUE	P42WKUE	P41WKUE	P40WKUE	0000,0000
P5WKUE	P5 Interrupt Wake-Up Enable Register	FD45H	-	-	P55WKUE	P54WKUE	P53WKUE	P52WKUE	P51WKUE	P50WKUE	xx00,0000
P6WKUE	P6 Interrupt Wake-Up Enable Register	FD46H	P67WKUE	P66WKUE	P65WKUE	P64WKUE	P63WKUE	P62WKUE	P61WKUE	P60WKUE	0000,0000
P7WKUE	P7 Interrupt Wake-Up Enable Register	FD47H	P77WKUE	P76WKUE	P75WKUE	P74WKUE	P73WKUE	P72WKUE	P71WKUE	P70WKUE	0000,0000
PINIPL	I/O Interrupt Priority Low Register	FD60H	P7IP	P6IP	P5IP	P4IP	P3IP	P2IP	P1IP	P0IP	0000,0000
PINIPH	I/O Interrupt Priority High Register	FD61H	P7IPH	P6IPH	P5IPH	P4IPH	P3IPH	P2IPH	P1IPH	P0IPH	0000,0000
COMEN	COM Enable Register	FB00H	C7EN	C6EN	C5EN	C4EN	C3EN	C2EN	C1EN	C0EN	0000,0000
SEGENL	SEG Enable Register	FB01H	S7EN	S6EN	S5EN	S4EN	S3EN	S2EN	S1EN	S0EN	0000,0000
SEGENH	SEG Enable Register	FB02H	S15EN	S14EN	S13EN	S12EN	S11EN	S10EN	S9EN	S8EN	0000,0000
LEDCTRL	LED Control Register	FB03H	LEDON	-	LEDMODE[1:0]			LEDDUTY[2:0]			0000,0000
LEDCKS	LED Clock Divide Register	FB04H									0000,0001
COM0_DA_L	Common Anode Mode Disply	FB10H									0000,0000
COM1_DA_L	Common Anode Mode Disply	FB11H									0000,0000
COM2_DA_L	Common Anode Mode Disply	FB12H									0000,0000
COM3_DA_L	Common Anode Mode Disply	FB13H									0000,0000
COM4_DA_L	Common Anode Mode Disply	FB14H									0000,0000
COM5_DA_L	Common Anode Mode Disply	FB15H									0000,0000
COM6_DA_L	Common Anode Mode Disply	FB16H									0000,0000
COM7_DA_L	Common Anode Mode Disply	FB17H									0000,0000
COM0_DA_H	Common Anode Mode Disply	FB18H									0000,0000
COM1_DA_H	Common Anode Mode Disply	FB19H									0000,0000
COM2_DA_H	Common Anode Mode Disply	FB1AH									0000,0000
COM3_DA_H	Common Anode Mode Disply	FB1BH									0000,0000
COM4_DA_H	Common Anode Mode Disply	FB1CH									0000,0000
COM5_DA_H	Common Anode Mode Disply	FB1DH									0000,0000
COM6_DA_H	Common Anode Mode Disply	FB1EH									0000,0000
COM7_DA_H	Common Anode Mode Disply	FB1FH									0000,0000
COM0_DC_L	Common Cathode Mode Disply	FB20H									0000,0000
COM1_DC_L	Common Cathode Mode Disply	FB21H									0000,0000
COM2_DC_L	Common Cathode Mode Disply	FB22H									0000,0000
COM3_DC_L	Common Cathode Mode Disply	FB23H									0000,0000
COM4_DC_L	Common Cathode Mode Disply	FB24H									0000,0000
COM5_DC_L	Common Cathode Mode Disply	FB25H									0000,0000
COM6_DC_L	Common Cathode Mode Disply	FB26H									0000,0000
COM7_DC_L	Common Cathode Mode Disply	FB27H									0000,0000
COM0_DC_H	Common Cathode Mode Disply	FB28H									0000,0000
COM1_DC_H	Common Cathode Mode Disply	FB29H									0000,0000
COM2_DC_H	Common Cathode Mode Disply	FB2AH									0000,0000
COM3_DC_H	Common Cathode Mode Disply	FB2BH									0000,0000
COM4_DC_H	Common Cathode Mode Disply	FB2CH									0000,0000

STC8H Series Manual

COM5_DC_H	Common Cathode Mode Disply	FB2DH									0000,0000
COM6_DC_H	Common Cathode Mode Disply	FB2EH									0000,0000
COM7_DC_H	Common Cathode Mode Disply	FB2FH									0000,0000
TSCHEN1	Touch Key Enable Register 1	FB40H	TKEN7	TKEN6	TKEN5	TKEN4	TKEN3	TKEN2	TKEN1	TKEN0	0000,0000
TSCHEN2	Touch Key Enable Register 2	FB41H	TKEN15	TKEN14	TKEN13	TKEN12	TKEN11	TKEN10	TKEN9	TKEN8	0000,0000
TSCFG1	Touch Key Configuration Register 1	FB42H	-	SCR[2:0]			-	DT[2:0]			0000,0000
TSCFG2	Touch Key Configuration Register 2	FB43H	-	-	-	-	-	-	TSVR[1:0]		0000,0000
TSWUTC	Touch Key Wakeup Control Register	FB44H									0000,0000
TSCTRL	Touch Key Control Register	FB45H	TSGO	SINGLE	TSWAIT	TSWUCS	TSDCEN	TSWUEN	TSSAMP[1:0]		0000,0000
TSSTA1	Touch Key Status Register 1	FB46H	LEDWK	-	-	-	TSWKCHN[3:0]				0000,0000
TSSTA2	Touch Key Status Register 2	FB47H	TSIF	TSDOV	-	-	TSDNCHN[3:0]				0000,0000
TSRT	Touch Key Time Control Register	FB48H									0000,0000
TSDATH	Touch Key Data High Byte	FB49H									0000,0000
TSDATL	Touch Key Data Low Byte	FB4AH									0000,0000
TSTH00H	Touch Key0 Threshold High Byte	FB50H									0000,0000
TSTH00L	Touch Key0 Threshold Low Byte	FB51H									0000,0000
TSTH01H	Touch Key1 Threshold High Byte	FB52H									0000,0000
TSTH01L	Touch Key1 Threshold Low Byte	FB53H									0000,0000
TSTH02H	Touch Key2 Threshold High Byte	FB54H									0000,0000
TSTH02L	Touch Key2 Threshold Low Byte	FB55H									0000,0000
TSTH03H	Touch Key3 Threshold High Byte	FB56H									0000,0000
TSTH03L	Touch Key3 Threshold Low Byte	FB57H									0000,0000
TSTH04H	Touch Key4 Threshold High Byte	FB58H									0000,0000
TSTH04L	Touch Key4 Threshold Low Byte	FB59H									0000,0000
TSTH05H	Touch Key5 Threshold High Byte	FB5AH									0000,0000
TSTH05L	Touch Key5 Threshold Low Byte	FB5BH									0000,0000
TSTH06H	Touch Key6 Threshold High Byte	FB5CH									0000,0000
TSTH06L	Touch Key6 Threshold Low Byte	FB5DH									0000,0000
TSTH07H	Touch Key7 Threshold High Byte	FB5EH									0000,0000
TSTH07L	Touch Key7 Threshold Low Byte	FB5FH									0000,0000
TSTH08H	Touch Key8 Threshold High Byte	FB60H									0000,0000
TSTH08L	Touch Key8 Threshold Low Byte	FB61H									0000,0000
TSTH09H	Touch Key9 Threshold High Byte	FB62H									0000,0000
TSTH09L	Touch Key9 Threshold Low Byte	FB63H									0000,0000
TSTH10H	Touch Key10 Threshold High Byte	FB64H									0000,0000
TSTH10L	Touch Key10 Threshold Low Byte	FB65H									0000,0000
TSTH11H	Touch Key11 Threshold High Byte	FB66H									0000,0000
TSTH11L	Touch Key11 Threshold Low Byte	FB67H									0000,0000
TSTH12H	Touch Key12 Threshold High Byte	FB68H									0000,0000
TSTH12L	Touch Key12 Threshold Low Byte	FB69H									0000,0000
TSTH13H	Touch Key13 Threshold High Byte	FB6AH									0000,0000
TSTH13L	Touch Key13 Threshold Low Byte	FB6BH									0000,0000
TSTH14H	Touch Key14 Threshold High Byte	FB6CH									0000,0000
TSTH14L	Touch Key14 Threshold Low Byte	FB6DH									0000,0000
TSTH15H	Touch Key15 Threshold High Byte	FB6EH									0000,0000
TSTH15L	Touch Key15 Threshold Low Byte	FB6FH									0000,0000

LCDFCFG	LCD configuration register	FB80H	CKSEL	-	-	-	VRLPSEL	VLCDSEL[2:0]				0xxx,0000
LCDFCFG2	LCD configuration register 2	FB81H	-	-	-	-	SEG3PS	SEG2PS	SEG1PS	SEG0PS	xxxx,0000	
DBLEN	Dead time length configuration	FB82H	-	-	-	-	-	DBLEN[2:0]				xxxx,x000
COMLENL	COM time length configuration low Bit	FB83H	COMLEN[7:0]								0000,0000	
COMLENM	COM time length configuration median	FB84H	COMLEN[15:8]								0000,0000	
COMLENH	COM time length configuration high bit	FB85H	-	-	-	-	COMLEN[19:16]				xxxx,0000	
BLINKRATE	Flicker rate configuration register	FB86H	BLANKRATE[7:0]								1000,0000	
LCDCR	LCD control register	FB87H	-	-	-	-	-	ACTMODE[1:0]			ENLCD	xxxx,x000
COMON	COM Line Enable Register	FB88H	-	-	-	-	COM3	COM2	COM1	COM0	xxxx,0000	
SEGON1	SEG Line Enable Register 1	FB8AH	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0	0000,0000	
SEGON2	SEG Line Enable Register 2	FB8BH	SEG15	SEG14	SEG13	SEG12	SEG11	SEG10	SEG9	SEG8	0000,0000	
SEGON3	SEG Line Enable Register 3	FB8CH	SEG23	SEG22	SEG21	SEG20	SEG19	SEG18	SEG17	SEG16	0000,0000	
SEGON4	SEG Line Enable Register 4	FB8DH	SEG31	SEG30	SEG29	SEG28	SEG27	SEG26	SEG25	SEG24	0000,0000	
SEGON5	SEG Line Enable Register 5	FB8EH	SEG39	SEG38	SEG37	SEG36	SEG35	SEG34	SEG33	SEG32	0000,0000	
C0SEGV0	C0SEG7_0 data register	FB90H	C0S7	C0S6	C0S5	C0S4	C0S3	C0S2	C0S1	C0S0	0000,0000	
C0SEGV1	C0SEG15_8 data register	FB91H	C0S15	C0S14	C0S13	C0S12	C0S11	C0S10	C0S9	C0S8	0000,0000	
C0SEGV2	C0SEG23_16 data register	FB92H	C0S23	C0S22	C0S21	C0S20	C0S19	C0S18	C0S17	C0S16	0000,0000	
C0SEGV3	C0SEG31_24 data register	FB93H	C0S31	C0S30	C0S29	C0S28	C0S27	C0S26	C0S25	C0S24	0000,0000	
C0SEGV4	C0SEG39_32 data register	FB94H	C0S39	C0S38	C0S37	C0S36	C0S35	C0S34	C0S33	C0S32	0000,0000	
C1SEGV0	C1SEG7_0 data register	FB98H	C1S7	C1S6	C1S5	C1S4	C1S3	C1S2	C1S1	C1S0	0000,0000	
C1SEGV1	C1SEG15_8 data register	FB99H	C1S15	C1S14	C1S13	C1S12	C1S11	C1S10	C1S9	C1S8	0000,0000	
C1SEGV2	C1SEG23_16 data register	FB9AH	C1S23	C1S22	C1S21	C1S20	C1S19	C1S18	C1S17	C1S16	0000,0000	
C1SEGV3	C1SEG31_24 data register	FB9BH	C1S31	C1S30	C1S29	C1S28	C1S27	C1S26	C1S25	C1S24	0000,0000	
C1SEGV4	C1SEG39_32 data register	FB9CH	C1S39	C1S38	C1S37	C1S36	C1S35	C1S34	C1S33	C1S32	0000,0000	
C2SEGV0	C2SEG7_0 data register	FBA0H	C2S7	C2S6	C2S5	C2S4	C2S3	C2S2	C2S1	C2S0	0000,0000	
C2SEGV1	C2SEG15_8 data register	FBA1H	C2S15	C2S14	C2S13	C2S12	C2S11	C2S10	C2S9	C2S8	0000,0000	
C2SEGV2	C2SEG23_16 data register	FBA2H	C2S23	C2S22	C2S21	C2S20	C2S19	C2S18	C2S17	C2S16	0000,0000	
C2SEGV3	C2SEG31_24 data register	FBA3H	C2S31	C2S30	C2S29	C2S28	C2S27	C2S26	C2S25	C2S24	0000,0000	
C2SEGV4	C2SEG39_32 data register	FBA4H	C2S39	C2S38	C2S37	C2S36	C2S35	C2S34	C2S33	C2S32	0000,0000	
C3SEGV0	C3SEG7_0 data register	FBA8H	C3S7	C3S6	C3S5	C3S4	C3S3	C3S2	C3S1	C3S0	0000,0000	
C3SEGV1	C3SEG15_8 data register	FBA9H	C3S15	C3S14	C3S13	C3S12	C3S11	C3S10	C3S9	C3S8	0000,0000	
C3SEGV2	C3SEG23_16 data register	FBAAH	C3S23	C3S22	C3S21	C3S20	C3S19	C3S18	C3S17	C3S16	0000,0000	
C3SEGV3	C3SEG31_24 data register	FBABH	C3S31	C3S30	C3S29	C3S28	C3S27	C3S26	C3S25	C3S24	0000,0000	
C3SEGV4	C3SEG39_32 data register	FBACH	C3S39	C3S38	C3S37	C3S36	C3S35	C3S34	C3S33	C3S32	0000,0000	
DMA_M2M_CFG	M2M_DMA configuration register	FA00H	M2MIE	-	TXACO	RXACO	M2MIP[1:0]		M2MPTY[1:0]		0x00,0000	
DMA_M2M_CR	M2M_DMA control register	FA01H	ENM2M	TRIG	-	-	-	-	-	-	00xx,xxxx	
DMA_M2M_STA	M2M_DMA status register	FA02H	-	-	-	-	-	-	-	M2MIF	xxxx,xxx0	
DMA_M2M_AMT	M2M_DMA total bytes to be transferred	FA03H									0000,0000	
DMA_M2M_DONE	M2M_DMA transfer completed bytes	FA04H									0000,0000	
DMA_M2M_TXAH	M2M_DMA send address high byte	FA05H									0000,0000	
DMA_M2M_TXAL	M2M_DMA send address low byte	FA06H									0000,0000	
DMA_M2M_RXAH	M2M_DMA receive address high byte	FA07H									0000,0000	
DMA_M2M_RXAL	M2M_DMA receive address low byte	FA08H									0000,0000	
DMA_ADC_CFG	ADC_DMA configuration register	FA10H	ADCIE	-	-	-	ADCMIP[1:0]		ADCPITY[1:0]		0xxx,0000	
DMA_ADC_CR	ADC_DMA control register	FA11H	ENADC	TRIG	-	-	-	-	-	-	00xx,xxxx	
DMA_ADC_STA	ADC_DMA status register	FA12H	-	-	-	-	-	-	-	ADCIF	xxxx,xxx0	
DMA_ADC_RXAH	ADC_DMA receive address high byte	FA17H									0000,0000	
DMA_ADC_RXAL	ADC_DMA receive address low byte	FA18H									0000,0000	
DMA_ADC_CFG2	ADC_DMA configuration register 2	FA19H	-	-	-	-	CVTIMESEL[3:0]				xxxx,0000	
DMA_ADC_CHSW0	ADC_DMA channel enable 0	FA1AH	CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	1000,0000	
DMA_ADC_CHSW1	ADC_DMA channel enable 1	FA1BH	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0	0000,0001	
DMA_SPI_CFG	SPI_DMA configuration register	FA20H	SPIIE	ACT_TX	ACT_RX	-	SPIIP[1:0]		SPIPTY[1:0]		000x,0000	
DMA_SPI_CR	SPI_DMA control register	FA21H	ENSPI	TRIG_M	TRIG_S	-	-	-	-	CLRFIFO	000x,xxx0	
DMA_SPI_STA	SPI_DMA status register	FA22H	-	-	-	-	-	TXOVW	RXLOSS	SPIIF	xxxx,x000	
DMA_SPI_AMT	SPI_DMA total bytes to be transferred	FA23H									0000,0000	
DMA_SPI_DONE	SPI_DMA transfer completed bytes	FA24H									0000,0000	
DMA_SPI_TXAH	SPI_DMA send address high byte	FA25H									0000,0000	
DMA_SPI_TXAL	SPI_DMA send address low byte	FA26H									0000,0000	
DMA_SPI_RXAH	SPI_DMA receive address high byte	FA27H									0000,0000	
DMA_SPI_RXAL	SPI_DMA receive address low byte	FA28H									0000,0000	
DMA_SPI_CFG2	SPI_DMA configuration register 2	FA29H	-	-	-	-	-	WRPSS	SSS[1:0]		xxxx,x000	
DMA_UR1T_CFG	UR1T_DMA configuration register	FA30H	UR1TIE	-	-	-	UR1TIP[1:0]		UR1TPTY[1:0]		0xxx,0000	
DMA_UR1T_CR	UR1T_DMA control register	FA31H	ENUR1T	TRIG	-	-	-	-	-	-	00xx,xxxx	
DMA_UR1T_STA	UR1T_DMA status register	FA32H	-	-	-	-	-	TXOVW	-	UR1TIF	xxxx,x0x0	
DMA_UR1T_AMT	UR1T_DMA total bytes to be transferred	FA33H									0000,0000	
DMA_UR1T_DONE	UR1T_DMA transfer completed bytes	FA34H									0000,0000	
DMA_UR1T_TXAH	UR1T_DMA send address high	FA35H									0000,0000	

STC8H Series Manual

	byte										
DMA_UR1T_TXAL	UR1T_DMA send address low byte	FA36H									0000,0000
DMA_UR1R_CFG	UR1R_DMA configuration register	FA38H	UR1RIE	-	-	-	UR1RIP[1:0]		UR1RPTY[1:0]		0xxx,0000
DMA_UR1R_CR	UR1R_DMA control register	FA39H	ENUR1R	-	TRIG	-	-	-	-	CLRFIFO	0x0x,xxx0
DMA_UR1R_STA	UR1R_DMA status register	FA3AH	-	-	-	-	-	-	RXLOSS	UR1RIF	xxxx,xx00
DMA_UR1R_AMT	UR1R_DMA total bytes to be transferred	FA3BH									0000,0000
DMA_UR1R_DONE	UR1R_DMA transfer completed bytes	FA3CH									0000,0000
DMA_UR1R_TXAH	UR1R_DMA send address high byte	FA3DH									0000,0000
DMA_UR1R_TXAL	UR1R_DMA send address low byte	FA3EH									0000,0000
DMA_UR2T_CFG	UR2T_DMA configuration register	FA40H	UR2TIE	-	-	-	UR2TIP[1:0]		UR2TPTY[1:0]		0xxx,0000
DMA_UR2T_CR	UR2T_DMA control register	FA41H	ENUR2T	TRIG	-	-	-	-	-	-	00xx,xxxx
DMA_UR2T_STA	UR2T_DMA status register	FA42H	-	-	-	-	-	TXOVW	-	UR2TIF	xxxx,x0x0
DMA_UR2T_AMT	UR2T_DMA total bytes to be transferred	FA43H									0000,0000
DMA_UR2T_DONE	UR2T_DMA transfer completed bytes	FA44H									0000,0000
DMA_UR2T_TXAH	UR2T_DMA send address high byte	FA45H									0000,0000
DMA_UR2T_TXAL	UR2T_DMA send address low byte	FA46H									0000,0000
DMA_UR2R_CFG	UR2R_DMA configuration register	FA48H	UR2RIE	-	-	-	UR2RIP[1:0]		UR2RPTY[1:0]		0xxx,0000
DMA_UR2R_CR	UR2R_DMA control register	FA49H	ENUR2R	-	TRIG	-	-	-	-	CLRFIFO	0x0x,xxx0
DMA_UR2R_STA	UR2R_DMA status register	FA4AH	-	-	-	-	-	-	RXLOSS	UR2RIF	xxxx,xx00
DMA_UR2R_AMT	UR2R_DMA total bytes to be transferred	FA4BH									0000,0000
DMA_UR2R_DONE	UR2R_DMA transfer completed bytes	FA4CH									0000,0000
DMA_UR2R_TXAH	UR2R_DMA send address high byte	FA4DH									0000,0000
DMA_UR2R_TXAL	UR2R_DMA send address low byte	FA4EH									0000,0000
DMA_UR3T_CFG	UR3T_DMA configuration register	FA50H	UR3TIE	-	-	-	UR3TIP[1:0]		UR3TPTY[1:0]		0xxx,0000
DMA_UR3T_CR	UR3T_DMA control register	FA51H	ENUR3T	TRIG	-	-	-	-	-	-	00xx,xxxx
DMA_UR3T_STA	UR3T_DMA status register	FA52H	-	-	-	-	-	TXOVW	-	UR3TIF	xxxx,x0x0
DMA_UR3T_AMT	UR3T_DMA total bytes to be transferred	FA53H									0000,0000
DMA_UR3T_DONE	UR3T_DMA transfer completed bytes	FA54H									0000,0000
DMA_UR3T_TXAH	UR3T_DMA send address high byte	FA55H									0000,0000
DMA_UR3T_TXAL	UR3T_DMA send address low byte	FA56H									0000,0000
DMA_UR3R_CFG	UR3R_DMA configuration register	FA58H	UR3RIE	-	-	-	UR3RIP[1:0]		UR3RPTY[1:0]		0xxx,0000
DMA_UR3R_CR	UR3R_DMA control register	FA59H	ENUR3R	-	TRIG	-	-	-	-	CLRFIFO	0x0x,xxx0
DMA_UR3R_STA	UR3R_DMA status register	FA5AH	-	-	-	-	-	-	RXLOSS	UR3RIF	xxxx,xx00
DMA_UR3R_AMT	UR3R_DMA total bytes to be transferred	FA5BH									0000,0000
DMA_UR3R_DONE	UR3R_DMA transfer completed bytes	FA5CH									0000,0000
DMA_UR3R_TXAH	UR3R_DMA send address high byte	FA5DH									0000,0000
DMA_UR3R_TXAL	UR3R_DMA send address low byte	FA5EH									0000,0000
DMA_UR4T_CFG	UR4T_DMA configuration register	FA60H	UR4TIE	-	-	-	UR4TIP[1:0]		UR4TPTY[1:0]		0xxx,0000
DMA_UR4T_CR	UR4T_DMA control register	FA61H	ENUR4T	TRIG	-	-	-	-	-	-	00xx,xxxx
DMA_UR4T_STA	UR4T_DMA status register	FA62H	-	-	-	-	-	TXOVW	-	UR4TIF	xxxx,x0x0
DMA_UR4T_AMT	UR4T_DMA total bytes to be transferred	FA63H									0000,0000
DMA_UR4T_DONE	UR4T_DMA transfer completed bytes	FA64H									0000,0000
DMA_UR4T_TXAH	UR4T_DMA send address high byte	FA65H									0000,0000
DMA_UR4T_TXAL	UR4T_DMA send address low byte	FA66H									0000,0000
DMA_UR4R_CFG	UR4R_DMA configuration register	FA68H	UR4RIE	-	-	-	UR4RIP[1:0]		UR4RPTY[1:0]		0xxx,0000
DMA_UR4R_CR	UR4R_DMA control register	FA69H	ENUR4R	-	TRIG	-	-	-	-	CLRFIFO	0x0x,xxx0
DMA_UR4R_STA	UR4R_DMA status register	FA6AH	-	-	-	-	-	-	RXLOSS	UR4RIF	xxxx,xx00
DMA_UR4R_AMT	UR4R_DMA total bytes to be transferred	FA6BH									0000,0000
DMA_UR4R_DONE	UR4R_DMA transfer completed bytes	FA6CH									0000,0000
DMA_UR4R_TXAH	UR4R_DMA send address high byte	FA6DH									0000,0000
DMA_UR4R_TXAL	UR4R_DMA send address low byte	FA6EH									0000,0000
DMA_LCM_CFG	LCM_DMA configuration register	FA70H	LCMIE	-	-	-	LCMIP[1:0]		LCMPTY[1:0]		0xxx,0000
DMA_LCM_CR	LCM_DMA control register	FA71H	ENLCM	TRIGWC	TRIGWD	TRIGRC	TRIGRD	-	-	-	0000,0xxx
DMA_LCM_STA	LCM_DMA status register	FA72H	-	-	-	-	-	-	TXOVW	LCMIF	xxxx,xx00
DMA_LCM_AMT	LCM_DMA total bytes to be transferred	FA73H									0000,0000
DMA_LCM_DONE	LCM_DMA transfer completed bytes	FA74H									0000,0000
DMA_LCM_TXAH	LCM_DMA send address high byte	FA75H									0000,0000
DMA_LCM_TXAL	LCM_DMA send address low byte	FA76H									0000,0000
DMA_LCM_RXAH	LCM_DMA receive address high	FA77H									0000,0000

STC8H Series Manual

	byte			
DMA_LCM_RXAL	LCM_DMA receive address low byte	FA78H		0000,0000

Note: The meaning of the initial value of the special function register:

0: The initial value is 0;

1: The initial value is 1;

n: The initial value is related to the hardware options when the ISP downloads;

x: This bit does not exist, the initial value is undefined.

STC MCU

9 I/O Ports

Product line	Maximum I/O lines
STC8H1K08 family	17
STC8H1K28 family	29
STC8H3K64S4 family	45
STC8H3K64S2 family	45
STC8H8K64U family	60
STC8H2K64T family	44
STC8H4K64TLR family	44
STC8H4K64TLCD family	60
STC8H4K64LCD family	61

There are 4 modes for all GPIOs of STC8H series MCU, quasi bidirectional or weak pull-up mode (standard 8051 output mode), push-pull output / strong pull-up mode, high-impedance input mode (where current can neither flow in nor out), open drain mode. It is easy to configure the I/O mode using software.

Note: All I/O ports except for P3.0 and P3.1 are in high-impedance input state after power-on. You must set the I/O port mode before using it.

9.1 Registers Related to I/O

Symbol	Description	Address	Bit Address and Symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
P0	Port 0	80H	P07	P06	P05	P04	P03	P02	P01	P00	1111,1111
P1	Port 1	90H	P17	P16	P15	P14	P13	P12	P11	P10	1111,1111
P2	Port 2	A0H	P27	P26	P25	P24	P23	P22	P21	P20	1111,1111
P3	Port 3	B0H	P37	P36	P35	P34	P33	P32	P31	P30	1111,1111
P4	Port 4	C0H	P47	P46	P45	P44	P43	P42	P41	P40	1111,1111
P5	Port 5	C8H	-	-	P55	P54	P53	P52	P51	P50	xx11,1111
P6	Port 6	E8H	P67	P66	P65	P64	P63	P62	P61	P60	1111,1111
P7	Port 7	F8H	P77	P76	P75	P74	P73	P72	P71	P70	1111,1111
P0M1	Port 0 mode register 1	93H	P07M1	P06M1	P05M1	P04M1	P03M1	P02M1	P01M1	P00M1	1111,1111
P0M0	Port 0 mode register 0	94H	P07M0	P06M0	P05M0	P04M0	P03M0	P02M0	P01M0	P00M0	0000,0000
P1M1	Port 1 mode register 1	91H	P17M1	P16M1	P15M1	P14M1	P13M1	P12M1	P11M1	P10M1	1111,1111
P1M0	Port 1 mode register 0	92H	P17M0	P16M0	P15M0	P14M0	P13M0	P12M0	P11M0	P10M0	0000,0000
P2M1	Port 2 mode register 1	95H	P27M1	P26M1	P25M1	P24M1	P23M1	P22M1	P21M1	P20M1	1111,1111
P2M0	Port 2 mode register 0	96H	P27M0	P26M0	P25M0	P24M0	P23M0	P22M0	P21M0	P20M0	0000,0000
P3M1	Port 3 mode register 1	B1H	P37M1	P36M1	P35M1	P34M1	P33M1	P32M1	P31M1	P30M1	n111,1100
P3M0	Port 3 mode register 0	B2H	P37M0	P36M0	P35M0	P34M0	P33M0	P32M0	P31M0	P30M0	n000,0000
P4M1	Port 4 mode register 1	B3H	P47M1	P46M1	P45M1	P44M1	P43M1	P42M1	P41M1	P40M1	1111,1111
P4M0	Port 4 mode register 0	B4H	P47M0	P46M0	P45M0	P44M0	P43M0	P42M0	P41M0	P40M0	0000,0000
P5M1	Port 5 mode register 1	C9H	-	-	P55M1	P54M1	P53M1	P52M1	P51M1	P50M1	xx11,1111
P5M0	Port 5 mode register 0	CAH	-	-	P55M0	P54M0	P53M0	P52M0	P51M0	P50M0	xx00,0000
P6M1	P6 mode registe 1	CBH	P67M1	P66M1	P65M1	P64M1	P63M1	P62M1	P61M1	P60M1	0000,0000
P6M0	P6 mode registe 0	CCH	P67M0	P66M0	P65M0	P64M0	P63M0	P62M0	P61M0	P60M0	0000,0000
P7M1	P7 mode registe 1	E1H	P77M1	P76M1	P75M1	P74M1	P73M1	P72M1	P71M1	P70M1	0000,0000
P7M0	P7 mode registe 0	E2H	P77M0	P76M0	P75M0	P74M0	P73M0	P72M0	P71M0	P70M0	0000,0000

Symbol	Description	Address	Bit Address and Symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
P0PU	P0 Pull-up resistor control register	FE10H	P07PU	P06PU	P05PU	P04PU	P03PU	P02PU	P01PU	P00PU	0000,0000
P1PU	P1 Pull-up resistor control register	FE11H	P17PU	P16PU	P15PU	P14PU	P13PU	P12PU	P11PU	P10PU	0000,0000
P2PU	P2 Pull-up resistor control register	FE12H	P27PU	P26PU	P25PU	P24PU	P23PU	P22PU	P21PU	P20PU	0000,0000
P3PU	P3 Pull-up resistor control register	FE13H	P37PU	P36PU	P35PU	P34PU	P33PU	P32PU	P31PU	P30PU	0000,0000
P4PU	P4 Pull-up resistor control register	FE14H	P47PU	P46PU	P45PU	P44PU	P43PU	P42PU	P41PU	P40PU	0000,0000
P5PU	P5 Pull-up resistor control register	FE15H	-	-	P55PU	P54PU	P53PU	P52PU	P51PU	P50PU	xx00,0000
P6PU	P6 Pull-up resistor control register	FE16H	P67PU	P66PU	P65PU	P64PU	P63PU	P62PU	P61PU	P60PU	0000,0000
P7PU	P7 Pull-up resistor control register	FE17H	P77PU	P76PU	P75PU	P74PU	P73PU	P72PU	P71PU	P70PU	0000,0000
P0NCS	P0 Schmitt trigger control register	FE18H	P07NCS	P06NCS	P05NCS	P04NCS	P03NCS	P02NCS	P01NCS	P00NCS	0000,0000
P1NCS	P1 Schmitt trigger control registe	FE19H	P17NCS	P16NCS	P15NCS	P14NCS	P13NCS	P12NCS	P11NCS	P10NCS	0000,0000
P2NCS	P2 Schmitt trigger control registe	FE1AH	P27NCS	P26NCS	P25NCS	P24NCS	P23NCS	P22NCS	P21NCS	P20NCS	0000,0000
P3NCS	P3 Schmitt trigger control registe	FE1BH	P37NCS	P36NCS	P35NCS	P34NCS	P33NCS	P32NCS	P31NCS	P30NCS	0000,0000
P4NCS	P4 Schmitt trigger control registe	FE1CH	P47NCS	P46NCS	P45NCS	P44NCS	P43NCS	P42NCS	P41NCS	P40NCS	0000,0000
P5NCS	P5 Schmitt trigger control registe	FE1DH	-	-	P55NCS	P54NCS	P53NCS	P52NCS	P51NCS	P50NCS	xx00,0000
P6NCS	P6 Schmitt trigger control register	FE1EH	P67NCS	P66NCS	P65NCS	P64NCS	P63NCS	P62NCS	P61NCS	P60NCS	0000,0000
P7NCS	P7 Schmitt trigger control register	FE1FH	P77NCS	P76NCS	P75NCS	P74NCS	P73NCS	P72NCS	P71NCS	P70NCS	0000,0000
P0SR	Port0 Level Shift Rate Register	FE20H	P07SR	P06SR	P05SR	P04SR	P03SR	P02SR	P01SR	P00SR	1111,1111
P1SR	Port1 Level Shift Rate Register	FE21H	P17SR	P16SR	P15SR	P14SR	P13SR	P12SR	P11SR	P10SR	1111,1111

STC8H Series Manual

P2SR	Port2 Level Shift Rate Register	FE22H	P27SR	P26SR	P25SR	P24SR	P23SR	P22SR	P21SR	P20SR	1111,1111
P3SR	Port3 Level Shift Rate Register	FE23H	P37SR	P36SR	P35SR	P34SR	P33SR	P32SR	P31SR	P30SR	1111,1111
P4SR	Port4 Level Shift Rate Register	FE24H	P47SR	P46SR	P45SR	P44SR	P43SR	P42SR	P41SR	P40SR	1111,1111
P5SR	Port5 Level Shift Rate Register	FE25H	-	-	-	P54SR	P53SR	P52SR	P51SR	P50SR	xx11,1111
P6SR	Port6 Level Shift Rate Register	FE26H	P67SR	P66SR	P65SR	P64SR	P63SR	P62SR	P61SR	P60SR	1111,1111
P7SR	Port7 Level Shift Rate Register	FE27H	P77SR	P76SR	P75SR	P74SR	P73SR	P72SR	P71SR	P70SR	1111,1111
P0DR	P0 Drive Current Control Register	FE28H	P07DR	P06DR	P05DR	P04DR	P03DR	P02DR	P01DR	P00DR	1111,1111
P1DR	P1 Drive Current Control Register	FE29H	P17DR	P16DR	P15DR	P14DR	P13DR	P12DR	P11DR	P10DR	1111,1111
P2DR	P2 Drive Current Control Register	FE2AH	P27DR	P26DR	P25DR	P24DR	P23DR	P22DR	P21DR	P20DR	1111,1111
P3DR	P3 Drive Current Control Register	FE2BH	P37DR	P36DR	P35DR	P34DR	P33DR	P32DR	P31DR	P30DR	1111,1111
P4DR	P4 Drive Current Control Register	FE2CH	P47DR	P46DR	P45DR	P44DR	P43DR	P42DR	P41DR	P40DR	1111,1111
P5DR	P5 Drive Current Control Register	FE2DH	-	-	P55DR	P54DR	P53DR	P52DR	P51DR	P50DR	xx11,1111
P6DR	P6 Drive Current Control Register	FE2EH	P67DR	P66DR	P65DR	P64DR	P63DR	P62DR	P61DR	P60DR	1111,1111
P7DR	P7 Drive Current Control Register	FE2FH	P77DR	P76DR	P75DR	P74DR	P73DR	P72DR	P71DR	P70DR	1111,1111
P0IE	P0 Input Enable Control Register	FE30H	P07IE	P06IE	P05IE	P04IE	P03IE	P02IE	P01IE	P00IE	1111,1111
P1IE	P1 Input Enable Control Register	FE31H	P17IE	P16IE	P15IE	P14IE	P13IE	P12IE	P11IE	P10IE	1111,1111
P2IE	P2 Input Enable Control Register	FE32H	P27IE	P26IE	P25IE	P24IE	P23IE	P22IE	P21IE	P20IE	1111,1111
P3IE	P3 Input Enable Control Register	FE33H	P37IE	P36IE	P35IE	P34IE	P33IE	P32IE	P31IE	P30IE	1111,1111
P4IE	P4 Input Enable Control Register	FE34H	P47IE	P46IE	P45IE	P44IE	P43IE	P42IE	P41IE	P40IE	1111,1111
P5IE	P5 Input Enable Control Register	FE35H	-	-	P55IE	P54IE	P53IE	P52IE	P41IE	P50IE	xx11,1111
P6IE	P6 Input Enable Control Register	FE36H	P67IE	P66IE	P65IE	P64IE	P63IE	P62IE	P41IE	P60IE	1111,1111
P7IE	P7 Input Enable Control Register	FE37H	P77IE	P76IE	P75IE	P74IE	P73IE	P72IE	P41IE	P70IE	1111,1111

9.1.1 Port Data Register

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

Read and write port status

Write 0: Output low to port buffer.

Write 1: Output high to port buffer.

Read: Read the level on the port pin directly.

9.1.2 Ports Mode Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P0M0	94H	P07M1	P06M1	P05M1	P04M1	P03M1	P02M1	P01M1	P00M1
P0M1	93H	P07M0	P06M0	P05M0	P04M0	P03M0	P02M0	P01M0	P00M0
P1M0	92H	P17M1	P16M1	P15M1	P14M1	P13M1	P12M1	P11M1	P10M1
P1M1	91H	P17M0	P16M0	P15M0	P14M0	P13M0	P12M0	P11M0	P10M0
P2M0	96H	P27M1	P26M1	P25M1	P24M1	P23M1	P22M1	P21M1	P20M1
P2M1	95H	P27M0	P26M0	P25M0	P24M0	P23M0	P22M0	P21M0	P20M0
P3M0	B2H	P37M1	P36M1	P35M1	P34M1	P33M1	P32M1	P31M1	P30M1
P3M1	B1H	P37M0	P36M0	P35M0	P34M0	P33M0	P32M0	P31M0	P30M0
P4M0	B4H	P47M1	P46M1	P45M1	P44M1	P43M1	P42M1	P41M1	P40M1
P4M1	B3H	P47M0	P46M0	P45M0	P44M0	P43M0	P42M0	P41M0	P40M0
P5M0	CAH	-	-	P55M1	P54M1	P53M1	P52M1	P51M1	P50M1
P5M1	C9H	-	-	P55M0	P54M0	P53M0	P52M0	P51M0	P50M0
P6M0	CCH	P67M1	P66M1	P65M1	P64M1	P63M1	P62M1	P61M1	P60M1
P6M1	CBH	P67M0	P66M0	P65M0	P64M0	P63M0	P62M0	P61M0	P60M0
P7M0	E2H	P77M1	P76M1	P75M1	P74M1	P73M1	P72M1	P71M1	P70M1
P7M1	E1H	P77M0	P76M0	P75M0	P74M0	P73M0	P72M0	P71M0	P70M0

Configure the mode of the ports as shown below.

PnM1.x	PnM0.x	Pn.x mode
0	0	quasi bidirectional mode

0	1	push-pull output mode
1	0	high-impedance input mode
1	1	open drain mode

Note: When an I/O port is selected as the ADC input channel, the PxM0/PxM1 register must be set to set the I/O port mode to input mode. In addition, if the ADC channel still needs to be enabled after the MCU enters the power-down mode/clock stop mode, you need to set the PxIE register to close the digital input to ensure that there will be no additional power consumption.

9.1.3 Pull-up Resistor Control Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P0PU	FE10H	P07PU	P06PU	P05PU	P04PU	P03PU	P02PU	P01PU	P00PU
P1PU	FE11H	P17PU	P16PU	P15PU	P14PU	P13PU	P12PU	P11PU	P10PU
P2PU	FE12H	P27PU	P26PU	P25PU	P24PU	P23PU	P22PU	P21PU	P20PU
P3PU	FE13H	P37PU	P36PU	P35PU	P34PU	P33PU	P32PU	P31PU	P30PU
P4PU	FE14H	P47PU	P46PU	P45PU	P44PU	P43PU	P42PU	P41PU	P40PU
P5PU	FE15H	-	-	P55PU	P54PU	P53PU	P52PU	P51PU	P50PU
P6PU	FE16H	P67PU	P66PU	P65PU	P64PU	P63PU	P62PU	P61PU	P60PU
P7PU	FE17H	P77PU	P76PU	P75PU	P74PU	P73PU	P72PU	P71PU	P70PU

Internal 4.1K pull-up resistor control bit. (Note: The pull-up resistors on the P3.0 and P3.1 ports may be slightly smaller.)

0: Disable 4.1K pull-up resistor inside the port

1: Enable 4.1K pull-up resistor inside the port

9.1.4 Schmitt Trigger Control Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P0NCS	FE18H	P07NCS	P06NCS	P05NCS	P04NCS	P03NCS	P02NCS	P01NCS	P00NCS
P1NCS	FE19H	P17NCS	P16NCS	P15NCS	P14NCS	P13NCS	P12NCS	P11NCS	P10NCS
P2NCS	FE1AH	P27NCS	P26NCS	P25NCS	P24NCS	P23NCS	P22NCS	P21NCS	P20NCS
P3NCS	FE1BH	P37NCS	P36NCS	P35NCS	P34NCS	P33NCS	P32NCS	P31NCS	P30NCS
P4NCS	FE1CH	P47NCS	P46NCS	P45NCS	P44NCS	P43NCS	P42NCS	P41NCS	P40NCS
P5NCS	FE1DH	-	-	P55NCS	P54NCS	P53NCS	P52NCS	P51NCS	P50NCS
P6NCS	FE1EH	P67NCS	P66NCS	P65NCS	P64NCS	P63NCS	P62NCS	P61NCS	P60NCS
P7NCS	FE1FH	P77NCS	P76NCS	P75NCS	P74NCS	P73NCS	P72NCS	P71NCS	P70NCS

Schmitt trigger control bit:

0: Enable schmitt trigger function on the port. (Schmitt trigger is enabled by default after power-on reset.)

1: Disable schmitt trigger function on the port.

9.1.5 Level Shifting Speed Control Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0	Reset value
P0SR	FE20H	P07SR	P06SR	P05SR	P04SR	P03SR	P02SR	P01SR	P00SR	1111,1111
P1SR	FE21H	P17SR	P16SR	P15SR	P14SR	P13SR	P12SR	P11SR	P10SR	1111,1111
P2SR	FE22H	P27SR	P26SR	P25SR	P24SR	P23SR	P22SR	P21SR	P20SR	1111,1111
P3SR	FE23H	P37SR	P36SR	P35SR	P34SR	P33SR	P32SR	P31SR	P30SR	1111,1111
P4SR	FE24H	P47SR	P46SR	P45SR	P44SR	P43SR	P42SR	P41SR	P40SR	1111,1111
P5SR	FE25H	-	-	P55SR	P54SR	P53SR	P52SR	P51SR	P50SR	xx11,1111
P6SR	FE26H	P57SR	P66SR	P65SR	P64SR	P63SR	P62SR	P61SR	P60SR	1111,1111
P7SR	FE27H	P77SR	P76SR	P75SR	P74SR	P73SR	P72SR	P71SR	P70SR	1111,1111

Level shifting speed control bits:

0: Fast level shifting, and the corresponding up and down impact will be relatively large.

1: Slow level shifting, and the corresponding up and down impact will be relatively small.

9.1.6 Drive Current Control Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0	Reset value
P0DR	FE28H	P07DR	P06DR	P05DR	P04DR	P03DR	P02DR	P01DR	P00DR	1111,1111
P1DR	FE29H	P17DR	P16DR	P15DR	P14DR	P13DR	P12DR	P11DR	P10DR	1111,1111
P2DR	FE2AH	P27DR	P26DR	P25DR	P24DR	P23DR	P22DR	P21DR	P20DR	1111,1111
P3DR	FE2BH	P37DR	P36DR	P35DR	P34DR	P33DR	P32DR	P31DR	P30DR	1111,1111
P4DR	FE2CH	P47DR	P46DR	P45DR	P44DR	P43DR	P42DR	P41DR	P40DR	1111,1111
P5DR	FE2DH	-	-	P55DR	P54DR	P53DR	P52DR	P51DR	P50DR	xx11,1111
P6DR	FE2EH	P67DR	P66DR	P65DR	P64DR	P63DR	P62DR	P61DR	P60DR	1111,1111
P7DR	FE2FH	P77DR	P76DR	P75DR	P74DR	P73DR	P72DR	P71DR	P70DR	1111,1111

Drive capability control bit:

0: Enhanced drive ability

1: General drive ability

9.1.7 Port digital signal input enable control register (PxIE)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P0IE	FE30H	P07IE	P06IE	P05IE	P04IE	P03IE	P02IE	P01IE	P00IE
P1IE	FE31H	P17IE	P16IE	P15IE	P14IE	P13IE	P12IE	P11IE	P10IE
P2IE	FE32H	P27IE	P26IE	P25IE	P24IE	P23IE	P22IE	P21IE	P20IE
P3IE	FE33H	P37IE	P36IE	P35IE	P34IE	P33IE	P32IE	P31IE	P30IE
P4IE	FE34H	P47IE	P46IE	P45IE	P44IE	P43IE	P42IE	P41IE	P40IE
P5IE	FE35H	-	-	P55IE	P54IE	P53IE	P52IE	P41IE	P50IE
P6IE	FE36H	P67IE	P66IE	P65IE	P64IE	P63IE	P62IE	P41IE	P60IE
P7IE	FE37H	P77IE	P76IE	P75IE	P74IE	P73IE	P72IE	P41IE	P70IE

Digital signal input enable control:

0: Disable digital signal input. If the I/O is used as an analog port such as a comparator input port, ADC input port, touch key input port or external crystal oscillator input pin, it must be set to 0 before entering the clock stop mode, otherwise there will be additional power consumption.

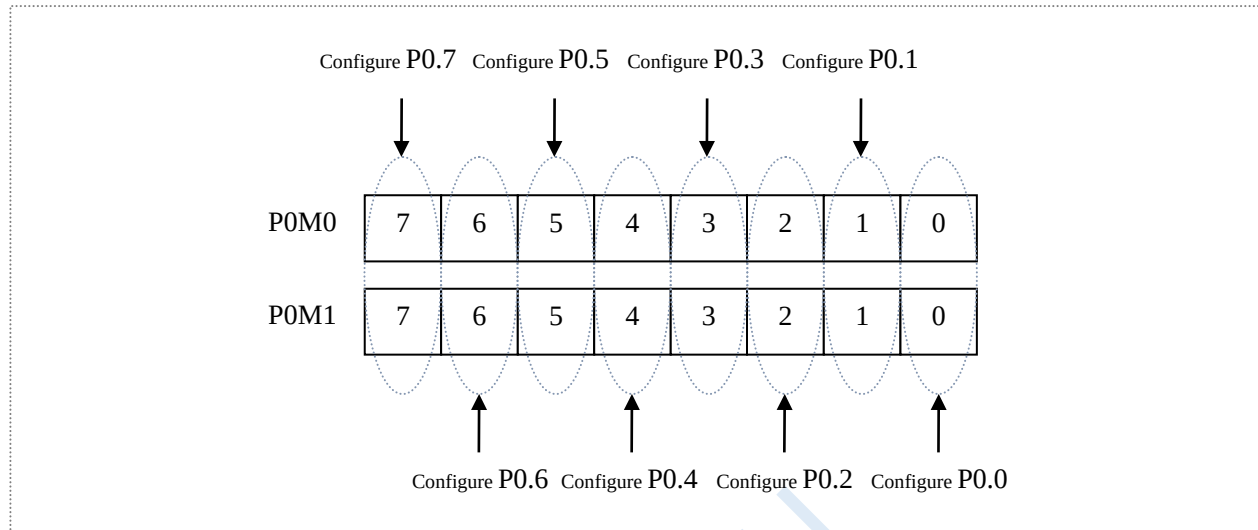
1: Enable digital signal input. If the I/O is used as a digital port, it must be set to 1, otherwise the MCU cannot read the level of the external port.

Special attention: For MCU with RTC function, when the clock source of RTC selects an external 32.768K crystal oscillator, it is necessary to close the digital channels of pins P1.6 and P1.7 which are connected with the crystal oscillator. Otherwise there will be additional leakage. (Set both bits 6 and 7 of register P1IE to 0 to close the digital channels of P1.6 and P1.7)

9.2 Configure I/O Ports

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

Taking Port 0 as an example, two registers, P0M0 and P0M1, are used to configure Port 0, as shown in the following figure:



The combination of bit 0 of P0M0 and bit 0 of P0M1 is used to configure the mode of P0.0.

The combination of bit 1 of P0M0 and bit 1 of P0M1 is used to configure the mode of P0.1.

All other I/O lines configuration method is similar.

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

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

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

Note:

Any I/O port line can tolerate 20mA of sink current in weak pull-up mode (quasi-bidirectional mode) or strong push-pull output mode or open drain mode, and can output 20mA pull current in the strong push-pull output mode. Current limiting resistors should be connected in all I/O mode above, such as 1KΩ, 560Ω, 472Ω, etc. **But the working current of the whole chip is recommended not to exceed 70mA, that is, the current flowing in from Vcc is not recommended to exceed 70mA, the current flowing from Gnd is not to exceed 70mA, and the overall current flowing in/out of it is not recommended to exceed 70mA.**

9.3 I/O Ports Structure

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

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

One of the three pull-up transistors, called “weak pull-up”, is turned on when the port register is logic “1” and the pin

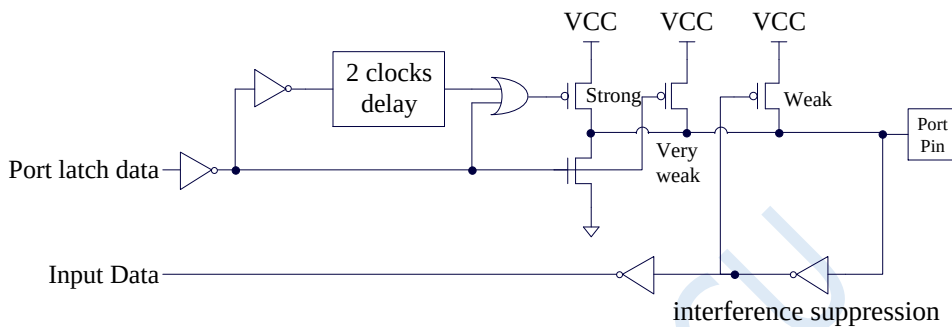
itself is logic “1”. This pull-up transistor provides the basic drive current to make the quasi-bidirectional port output logic “1”. If one of the pin outputs logic “1” and the external device pulls it low, the weak pull-up transistor is off and the “very weak pull-up” maintains on. To pull the pin low,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

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

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

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

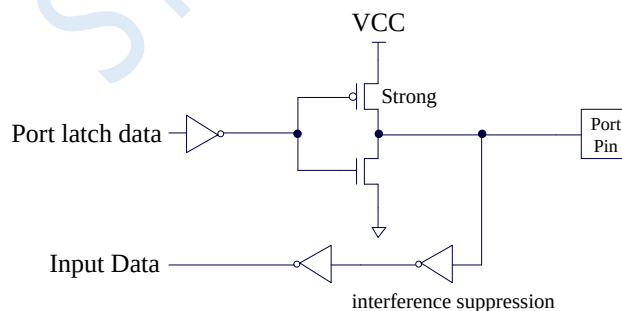
The structure of quasi-bidirectional port (weak pull-up) output is shown below:



9.3.2 Push-Pull Output

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

The structure of strong push-pull pin configuration is shown below:

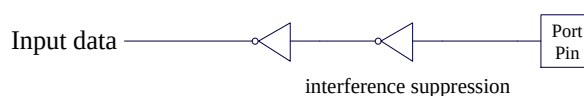


9.3.3 High Impedance Input

The current can neither flow in nor flow out.

The input port has a Schmitt trigger input and an interference suppression circuit.

The structure of high impedance input pin configuration is shown below:



9.3.4 Open-Drain Output

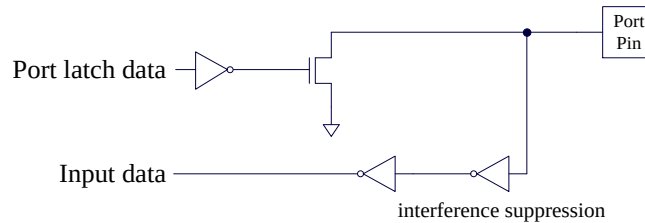
The open-drain mode can be used for both reading external status and outputting high or low level. To read the

external state correctly or output a high level, the external pull-up resistor should be connected.

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

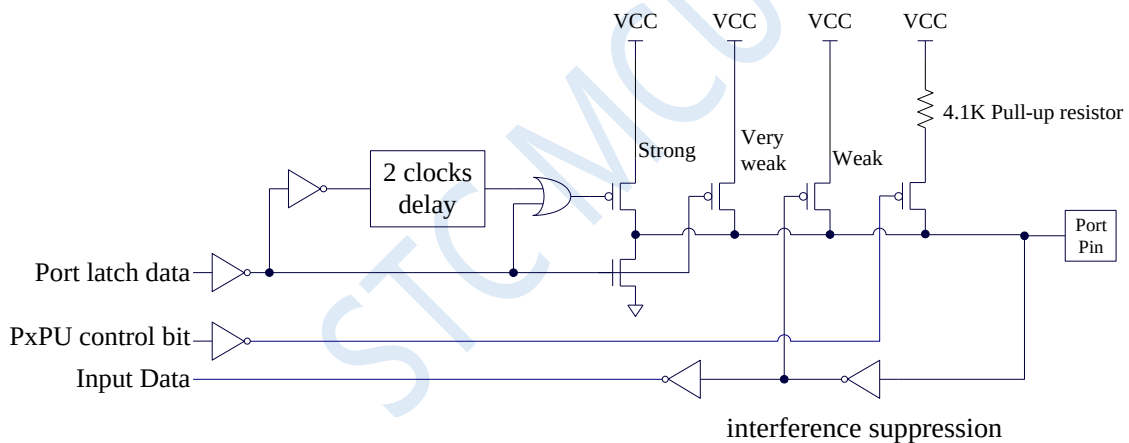
The open drain port has a Schmitt trigger input and an interference suppression circuit.

The structure of open drain port configuration is shown below:



9.3.5 4.1K Pull-up Resistor

A pull-up resistor of approximately 4.1K can be enabled internally in all I/O ports of the STC8 series (due to manufacturing errors, the range of the pull-up resistor may be 3K to 5K).



Pull-up Resistor Control Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P0PU	FE10H	P07PU	P06PU	P05PU	P04PU	P03PU	P02PU	P01PU	P00PU
P1PU	FE11H	P17PU	P16PU	P15PU	P14PU	P13PU	P12PU	P11PU	P10PU
P2PU	FE12H	P27PU	P26PU	P25PU	P24PU	P23PU	P22PU	P21PU	P20PU
P3PU	FE13H	P37PU	P36PU	P35PU	P34PU	P33PU	P32PU	P31PU	P30PU
P4PU	FE14H	P47PU	P46PU	P45PU	P44PU	P43PU	P42PU	P41PU	P40PU
P5PU	FE15H	-	-	-	P54PU	P53PU	P52PU	P51PU	P50PU
P6PU	FE16H	P67PU	P66PU	P65PU	P64PU	P63PU	P62PU	P61PU	P60PU
P7PU	FE17H	P77PU	P76PU	P75PU	P74PU	P73PU	P72PU	P71PU	P70PU

Internal 4.1K pull-up resistor control bit (Note: The pull-up resistors on the P3.0 and P3.1 ports may be slightly smaller)

0: Disable 4.1K pull-up resistor inside the port

1: Enable 4.1K pull-up resistor inside the port

9.3.6 How to set the external output speed of the I/O port

When users need the I/O port to output a faster frequency, they can increase the I/O port drive current and increase the I/O port level conversion speed to increase the I/O port output speed.

Setting the PxSR register can be used to control the I/O port level conversion speed. When it is set to 0, the corresponding I/O port is fast flipping, and when it is set to 1, it is slow flipping.

Set the PxDR register, which can be used to control the drive current of the I/O port. When set to 1, I/O output is general drive current, and when set to 0, it is strong drive current.

9.3.7 How to set I/O port current drive capability

If you need to change the current drive capability of the I/O port, you can do so by setting the PxDR register

Set the PxDR register, which can be used to control the drive current of the I/O port. When set to 1, I/O output is general drive current, when set to 0, it is strong drive current

9.3.8 How to reduce the external radiation of I/O ports

Because the PxSR register is set, it can be used to control the I/O port level conversion speed, and the PxDR register can be used to control the I/O port drive current.

When the external radiation of the I/O port needs to be reduced, the PxSR register needs to be set to 1 to reduce the I/O port level conversion speed, and the PxDR register needs to be set to 1 to reduce the I/O drive current, and finally reduce I /O port external radiation

STC MCU

9.4 Example Routines

9.4.1 Port Mode Setting

C language code

//Operating frequency for test is 11.0592MHz

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

```
sfr      P0M0      =    0x94;
sfr      P0M1      =    0x93;
sfr      P1M0      =    0x92;
sfr      P1M1      =    0x91;
sfr      P2M0      =    0x96;
sfr      P2M1      =    0x95;
sfr      P3M0      =    0xb2;
sfr      P3M1      =    0xb1;
sfr      P4M0      =    0xb4;
sfr      P4M1      =    0xb3;
sfr      P5M0      =    0xca;
sfr      P5M1      =    0xc9;
sfr      P6M0      =    0xcc;
sfr      P6M1      =    0xcb;
sfr      P7M0      =    0xe2;
sfr      P7M1      =    0xe1;
```

```
void main()
{
```

```
    P0M0 = 0x00;           //Set P0.0 ~ P0.7 as bidirectional port mode
    P0M1 = 0x00;
    P1M0 = 0xff;           //Set P1.0 ~ P1.7 as push-pull output mode
    P1M1 = 0x00;
    P2M0 = 0x00;           //Set P2.0 ~ P2.7 as high impedance input mode
    P2M1 = 0xff;
    P3M0 = 0xff;           //Set P3.0 ~ P3.7 as open-drain mode
    P3M1 = 0xff;
```

```
    while (1);
```

```
}
```

Assembly code

;Operating frequency for test is 11.0592MHz

```
P0M0      DATA      094H
P0M1      DATA      093H
P1M0      DATA      092H
P1M1      DATA      091H
P2M0      DATA      096H
P2M1      DATA      095H
P3M0      DATA      0B2H
P3M1      DATA      0B1H
P4M0      DATA      0B4H
P4M1      DATA      0B3H
P5M0      DATA      0CAH
P5M1      DATA      0C9H
P6M0      DATA      0CCH
```

```

P6M1      DATA      0CBH
P7M0      DATA      0E2H
P7M1      DATA      0E1H

          ORG          0000H
          LJMP         MAIN

MAIN:      ORG          0100H

          MOV          SP, #5FH

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

          JMP          $

          END

```

9.4.2 Reading and Writing Operation of Bidirection Port

C language code

//Operating frequency for test is 11.0592MHz

```

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

```

```

sfr      P0M0      = 0x94;
sfr      P0M1      = 0x93;
sfr      P0M1      = 0x93;
sfr      P0M0      = 0x94;
sfr      P1M1      = 0x91;
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;
sbit     P00       = P0^0;

```

```

void main()
{

```

```

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

```

```

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

P0M0 = 0x00;           //Set P0.0 ~ P0.7 as bidirectional port mode
P0M1 = 0x00;

P00 = 1;               //P0.0 output high level
P00 = 0;               //P0.0 output low level

P00 = 1;               //Enable the internal weak pull-up resistor before reading the port
_nop_0;                //Wait for two clocks
_nop_0;                //
CY = P00;              //Read port status

while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

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

        ORG       0000H
        LJMP      MAIN

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

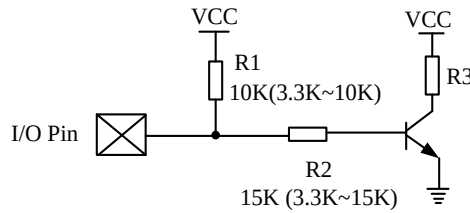
        MOV       P0M0, #00H           ;Set P0.0 ~ P0.7 as bidirectional port mode
        MOV       P0M1, #00H

```

<i>SETB</i>	<i>P0.0</i>	<i>;P0.0 output high level</i>
<i>CLR</i>	<i>P0.0</i>	<i>;P0.0 output low level</i>
<i>SETB</i>	<i>P0.0</i>	<i>;Enable the internal weak pull-up resistor before reading the port</i>
<i>NOP</i>		<i>;Wait for two clocks</i>
<i>NOP</i>		
<i>MOV</i>	<i>C,P0.0</i>	<i>;Read port status</i>
<i>JMP</i>	<i>\$</i>	
<i>END</i>		

STC MCU

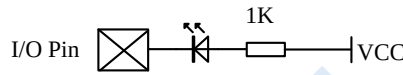
9.5 A Typical Circuit Controlled by Triode



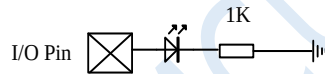
For pull-up control, it is recommended to add a pull-up resistor R1 (3.3K ~ 10K). If pull-up resistor R1 (3.3K ~ 10K) is not connected, it is recommended that the value of R2 be above 15K, or use strong push-pull output mode.

9.6 Typical Control Circuit of LED

For Quasi-Bidirectional (weak pull-up) I/O, you can drive the light-emitting diode using sink current mode, where the current limiting resistance should be greater than 1K oms, preferably not less than 470Ω.

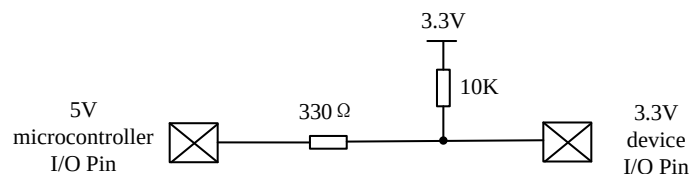


For push-pull (strong pull-up) I/O, you can drive the light-emitting diode with pull current mode.

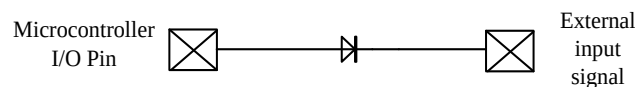


9.7 Interconnection of 3V/5V Devices in Mixed Voltage Power Supply System

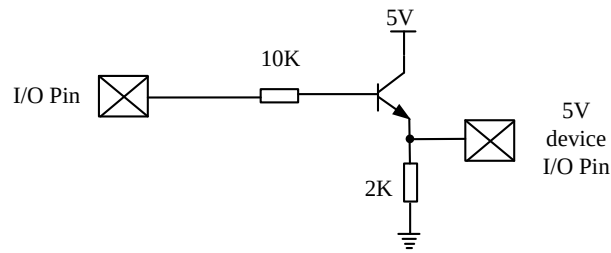
When STC's 5V microcontroller is connected to a 3.3V device, the corresponding I/O port of the 5V microcontroller can be connected with a 330Ω current limiting resistor to the 3.3V device I/O port in order to prevent the 3.3V device from withstanding 5V. The I/O port of the microcontroller is set to open-drain mode, and the internal pull-up resistor is disconnected. The corresponding 3.3V device I/O port is connected to 3.3V via with a 10K pull-up resistor. Then the high level is 3.3V, and the low level is 0V.



When STC's 3V microcontroller is connected to a 5V device and the corresponding I/O port is used as an input, an isolation diode can be connected in series to the I/O port to isolate the high voltage part in order to prevent the 3V microcontroller from bearing 5V. When the external signal voltage is higher than the microcontroller operating voltage, the isolation diode will cutoff, the state of the read I/O port is high because the I/O port is pulled up to a high level internally. When the external signal voltage is low, the isolation diode will turn on and the I/O port is clamped at 0.7V. The microcontroller reads I/O port low status as the voltage is less than 0.8V.



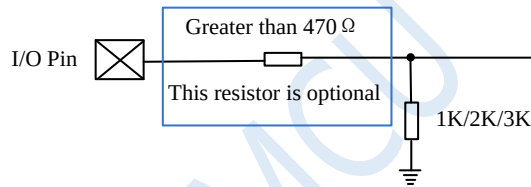
When STC's 3V microcontroller is connected to a 5V device and the corresponding I/O port is used as an output, it can be isolated with an NPN transistor in order to prevent the 3V microcontroller from bearing 5V. The circuit is as follows.



9.8 Make I/O Port Output Low When Power on Reset

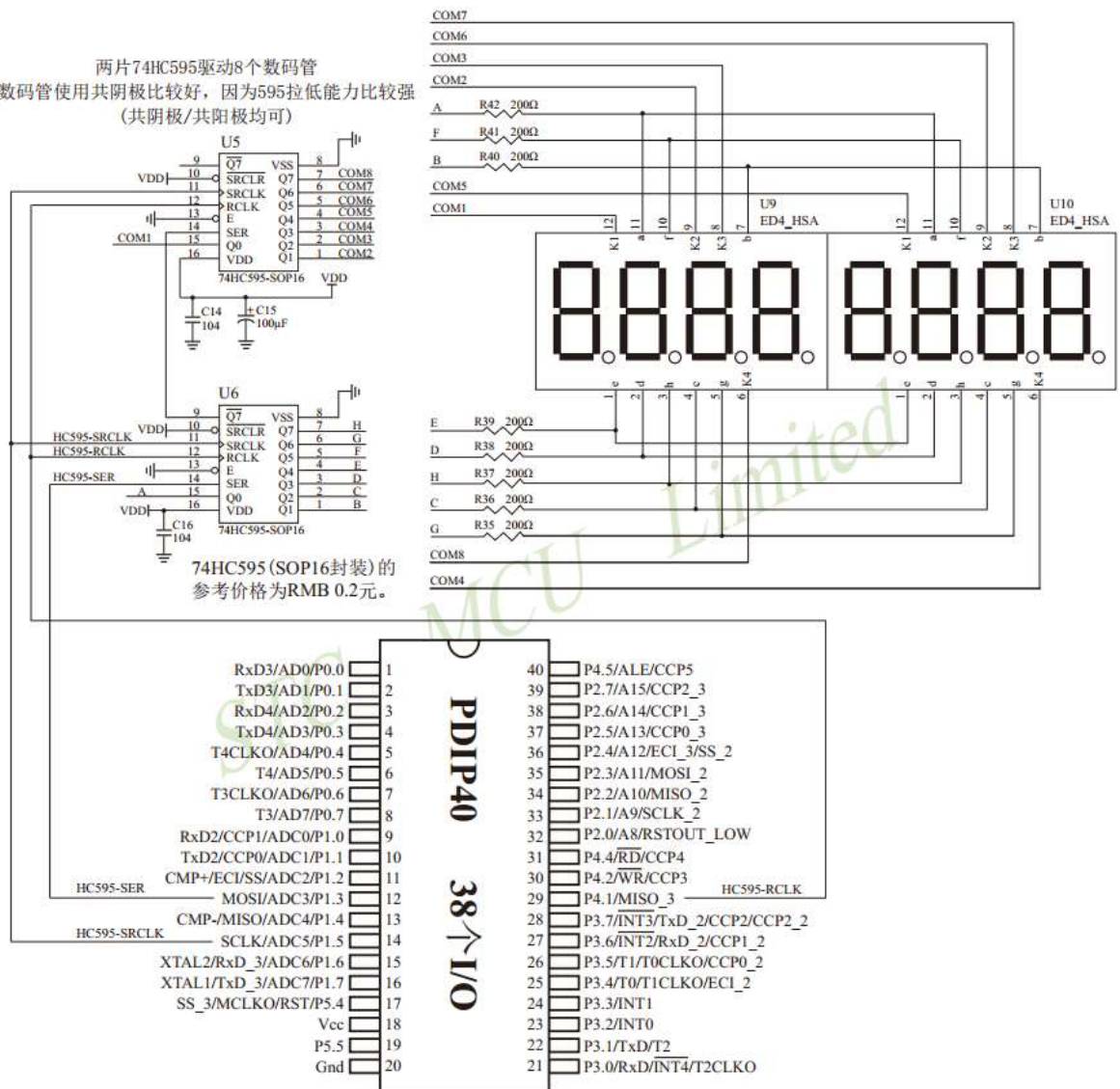
The general I/O port which is a weak pull-up (quasi-bidirectional) port will output high level when the traditional 8051 microcontroller is powered on and reset. Many practical applications require that certain I/O ports be low level output when powered on, otherwise the system (such as the motor) controlled by the microcontroller will malfunction. The STC microcontroller's I/O ports have weak pull-up output mode and strong push-pull output mode, which can easily solve this problem.

Now you can connect a pull-down resistor (1K, 2K or 3K) to the I/O port of STC microcontroller. The I/O port of the microcontroller is internal weak pull-up (quasi-bidirectional)/high-level output when the microcontroller is powered on reset. It cannot be pulled high because the internal pull-up capability is limited and the external pull-down resistor is small. Therefore the I/O port is externally low when power is reset. If you want to drive this I/O port to a high level, you can set this I/O port in strong push-pull output mode, under this circumstance, the I/O port drive current can reach 20mA, so the port can be definitely driven high.

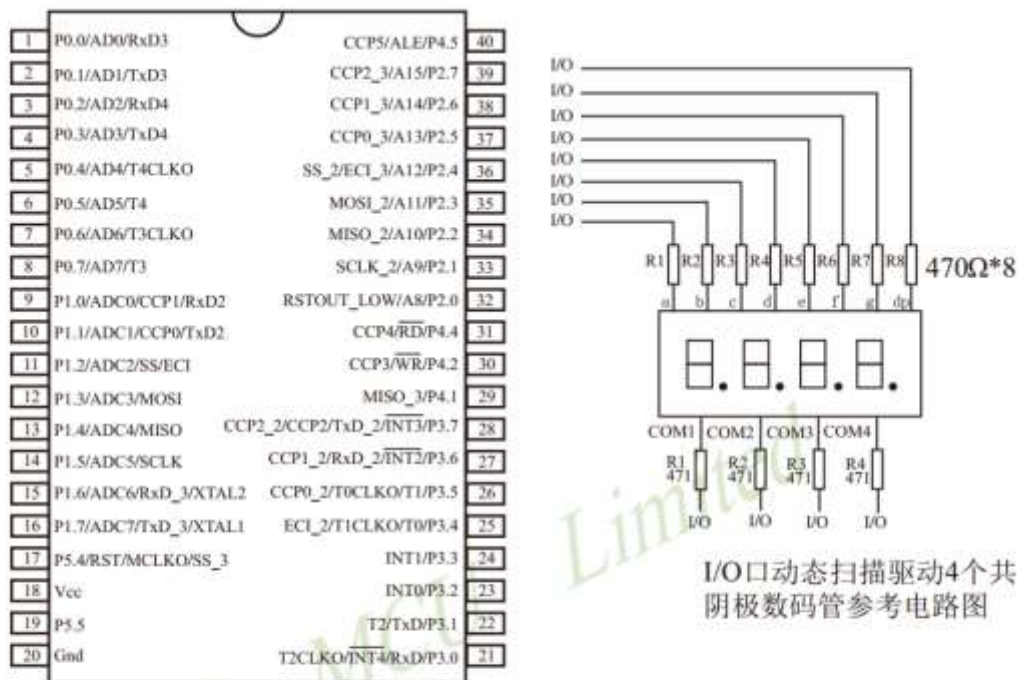


9.9 Circuit Diagram of Driving 8 Digital LEDs using 74HC595

两片74HC595驱动8个数码管
数码管使用共阴极比较好，因为595拉低能力比较强
(共阴极/共阳极均可)

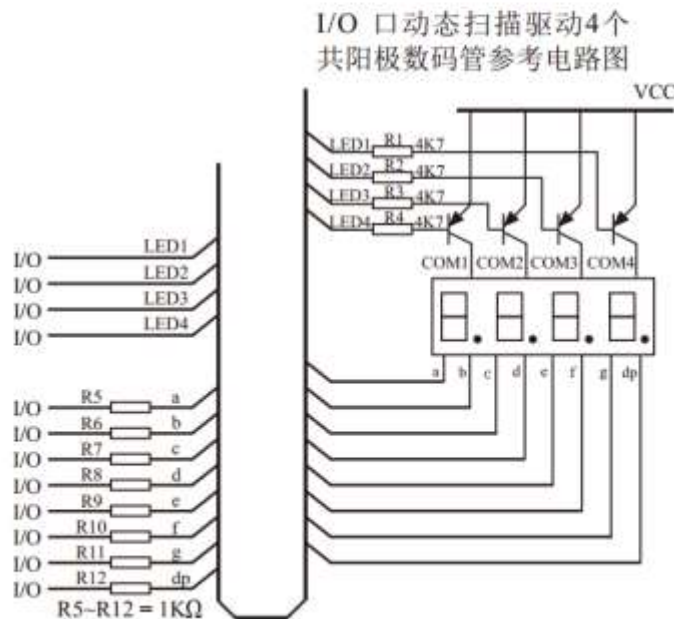


9.10 Digital LEDs Driven Directly by I/O Port Circuit



I/O口动态扫描驱动4个共阴极数码管参考电路图

I/O 口动态扫描驱动数码时，可以一次点亮一个数码管中的8段，但为降低功耗，建议可以一次只点亮其中的4段或者2段



I/O 口动态扫描驱动4个共阳极数码管参考电路图

9.11 LCD Segment LCD Driven Directly by I/O Port Circuit

An external LCD driver IC is needed when you design a product having segment LCD display requirement using MCU without LCD driver is used, which will increase cost. In fact, many small projects, such as a large number of small appliances, do not need to display a lot of segment codes. There are usually four 8 colons with a decimal point or clock: ":", so if you use the IO port to scan and display directly, it will reduce costs and work more reliably.

However, this solution is not suitable for driving too many segments (occupying too many IOs), and it is not suitable

for very low power consumption occasions (driving will have hundreds of μA current).

The simple principle of segment code LCD driving is shown in Figure 1.

LCD is a special kind of liquid crystal. The arrangement direction of the crystal will be reversed under the action of an electric field, which changes its light transmittance, so that the display content can be seen. LCD has a torsional voltage threshold. The content will be displayed if the voltage across the LCD is higher than this threshold, and the content will not be displayed if the voltage is lower than this threshold. There are usually 3 parameters in LCD: working voltage, DUTY (corresponding to COM number) and BIAS (ie bias, corresponding threshold). For example, '3.0V, 1/4 DUTY, 1/3 BIAS' means that the LCD display voltage is 3.0V, 4 COM, the threshold is about 1.5V. The content will be displayed if the voltage across a certain LCD segment is 3.0V, and the content will not be displayed if the voltage is 1.0V. However, the LCD's response to the driving voltage is not very sensitive. For example, the display may be faint if the voltage is 2V. This is usually called a 'ghost image'. Therefore, it is necessary to ensure that the voltage is larger than the threshold value if you want to display something, and the voltage is smaller than the threshold value if you do not display anything.

Note: The LCD should be driven by AC, and DC voltage cannot be applied to the two ends of the LCD. Otherwise it will be damaged for a long time DC applying. The average voltage of the driving voltage applied to the LCD must be 0. Time-sharing is used to LCD. At any time, if one COM scan is valid, the other COM is invalid.

The scheme circuit for driving '1/4Duty, 1/2BIAS, 3V' is shown in Figure 1. The scanning principle of LCD is shown in Figure 3. The MCU works at 3.0V or 3.3V. Each COM is connected with a 20K resistor in series to a capacitor C1 aiming to obtain the midpoint voltage of $1/2V_{DD}$ after RC filtering. When it is the turn of a COM scan, the connected IO is set to push-pull output mode, and the remaining COMs are set to high impedance. If the SEG connected to this COM is not used to display, the SEG output is in phase with the COM, and if it is not used to display, it is inverted. After scanning is finished, the I/O corresponding to this COM is set to high impedance. Each COM is connected to the $1/2V_{DD}$ voltage on the capacitor C1 through a 20K resistor. The SEG outputs high or low level according to whether it is used to display or not. The voltage applied to the LCD segment is $\pm V_{DD}$ when displayed, and $\pm 1/2V_{DD}$ when not displayed, which can ensure that the average DC voltage across the LCD is 0.

The circuit for driving the '1/4Duty, 1/3BIAS, 3V' is shown in Figure 4. The scanning principle of LCD is shown in Figure 5. The MCU works at 5V. The IOs connected to SEG output 1.5V and 3.5V through the resistor divider. The IOs connected to COM output 0.5V, 2.5V (at high impedance), 4.5V. The common point of the voltage-dividing resistor is connected to a capacitor C1 to obtain a mid-point voltage of $1/2V_{DD}$ after RC filtering. When it is the turn of a certain COM scan, the IO is set to push-pull output mode. If the SEG connected to this COM is not used to display, the SEG output is in phase with COM, and if it is used to display, it is inverted. After scanning is finished, the I/O corresponding to this COM is set to high impedance. This COM is connected to a 2.5V voltage through a 47K resistor. The SEG outputs high or low level according to whether it is used to display or not. The voltage applied to the LCD is $\pm 3.0V$ when displayed, and $\pm 1.0V$ when not displaying, which can meet the LCD scanning requirements.

When sleep of power saving is required, all IOs used to drive COMs and SEGs output low level, and the extra current will not appear in LCD drive part.

Figure 1 Circuit for driving a '1/4Duty, 1/2BIAS, 3V' LCD

本电路MCU工作于3.3V驱动1/4 Duty, 1/2 bias, 3V的段码LCD。
 本电路适用于STC8系列, IO都是普通IO操作。
 C1用于中点电压滤波, 4.7~47uF。
 MCU睡眠后LCD驱动部分电路不会耗电。

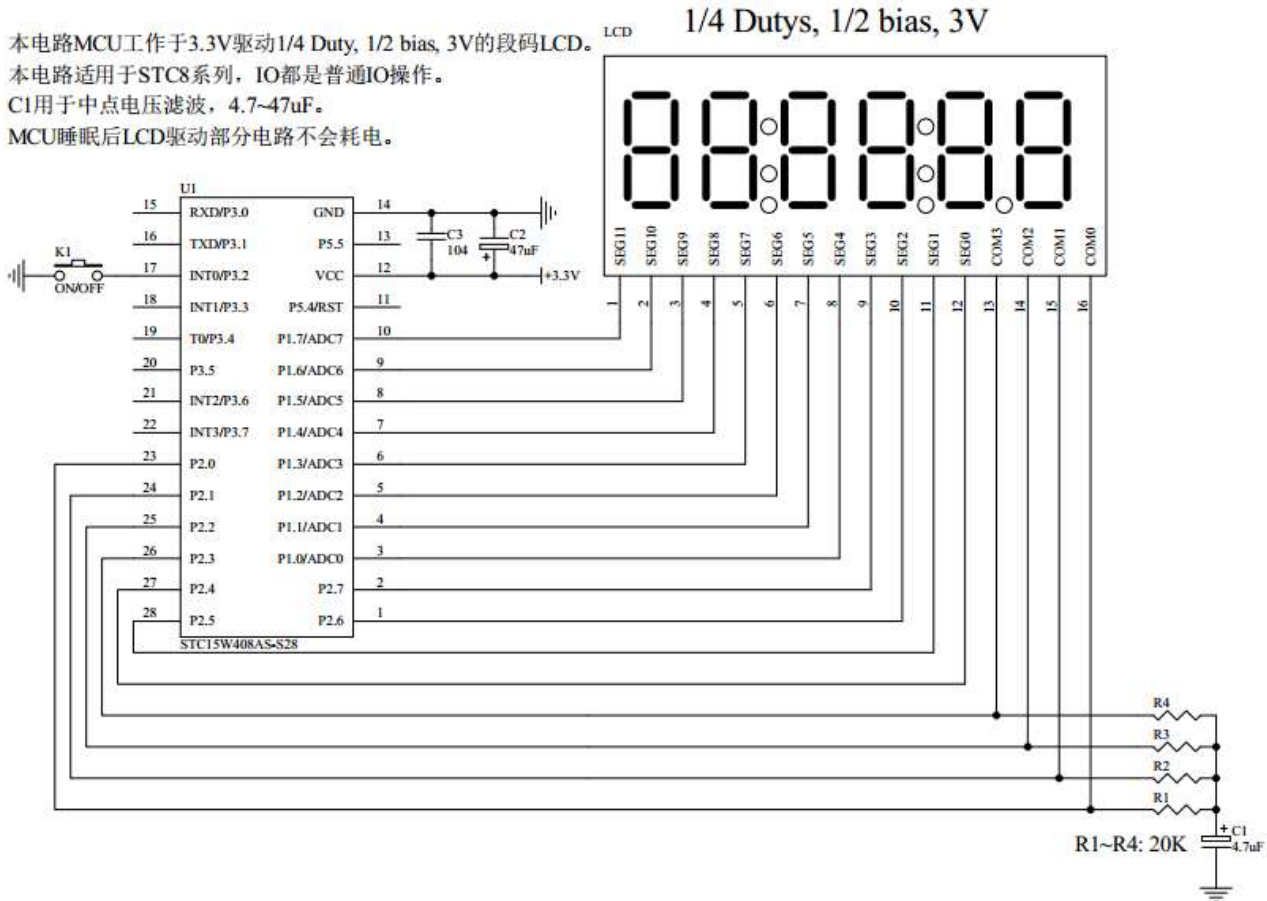


Figure 2 Segment name

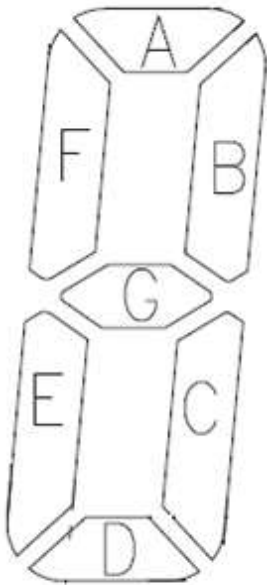


Figure 3 Principle of '1/4Duty, 1/2BIAS' scanning

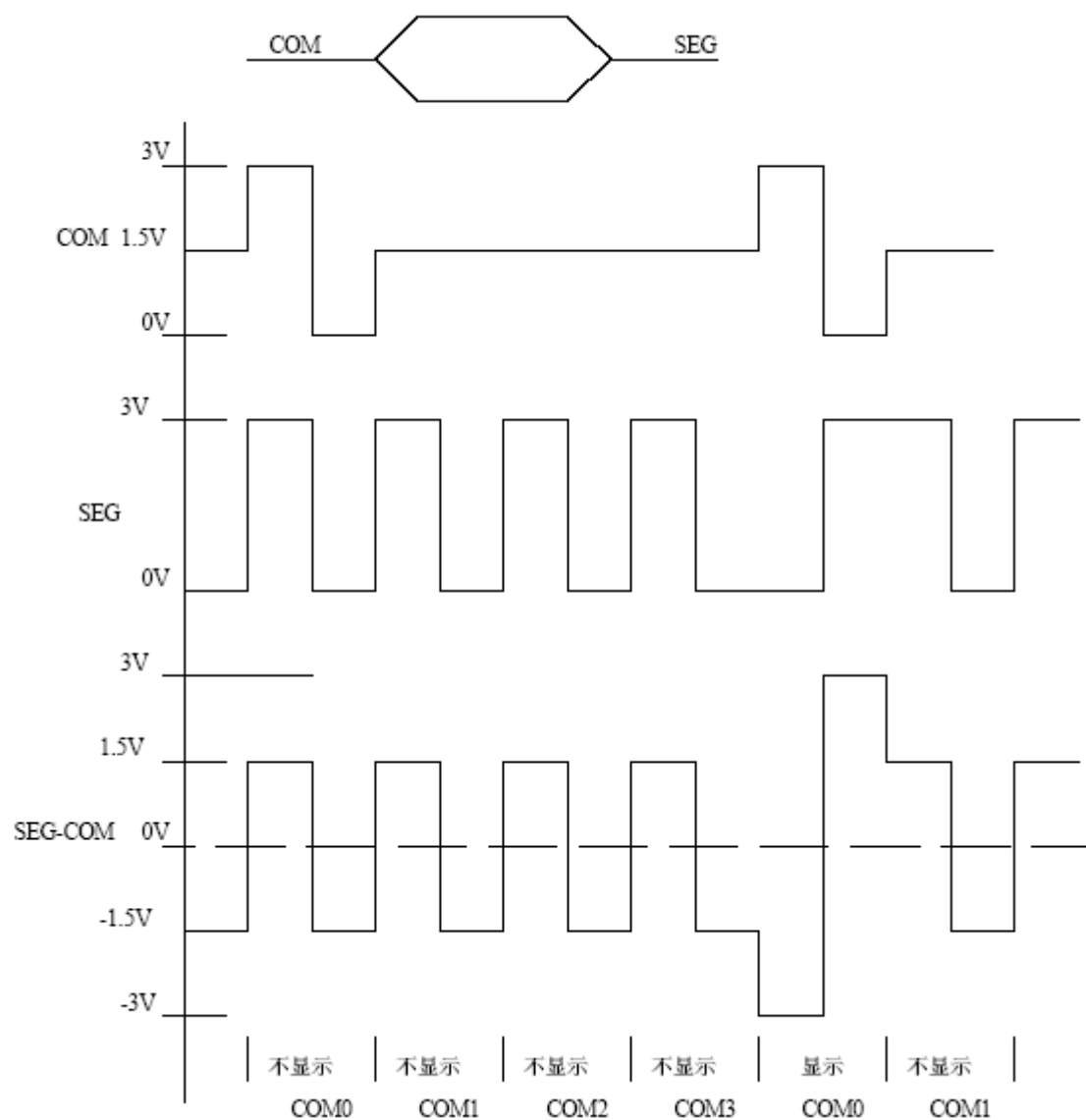


Figure 4 Driving circuit of '1/4Duty, 1/3BIAS, 3V' LCD

本电路MCU工作于5V驱动1/4 Duty, 1/3 bias, 3V的段码LCD。

本电路适用于STC8系列，IO都是普通IO操作。

C1用于中点电压滤波，4.7~47uF。

MCU睡眠后LCD驱动部分电路不会耗电。

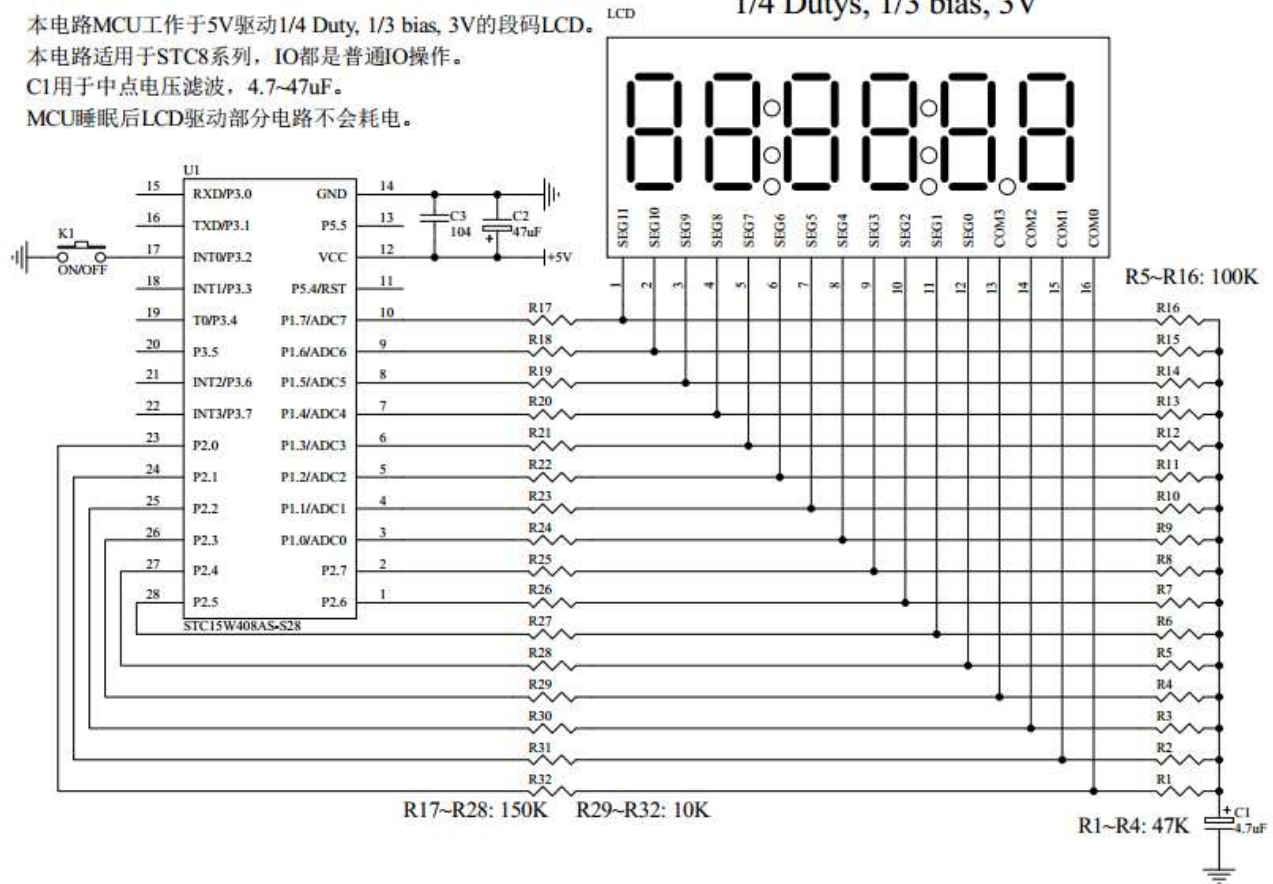
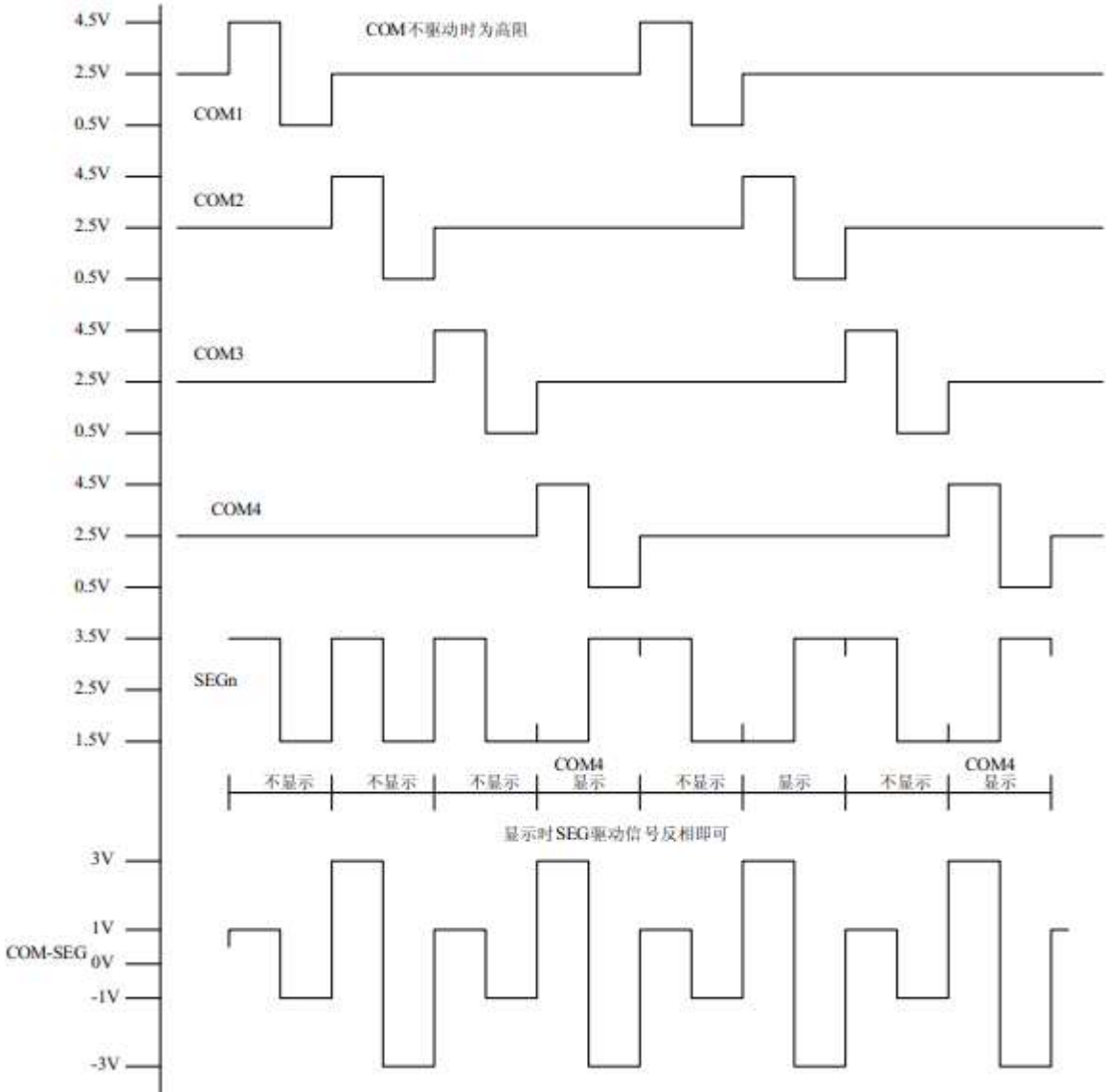


Figure 5 Principle of '1/4Duty, 1/3BIAS' scanning



For ease of use, the display contents are stored in display memory where each bit corresponds to the LCD segment one by one, as shown in Figure 6.

Figure 6 LCD truth table and memory mapping table

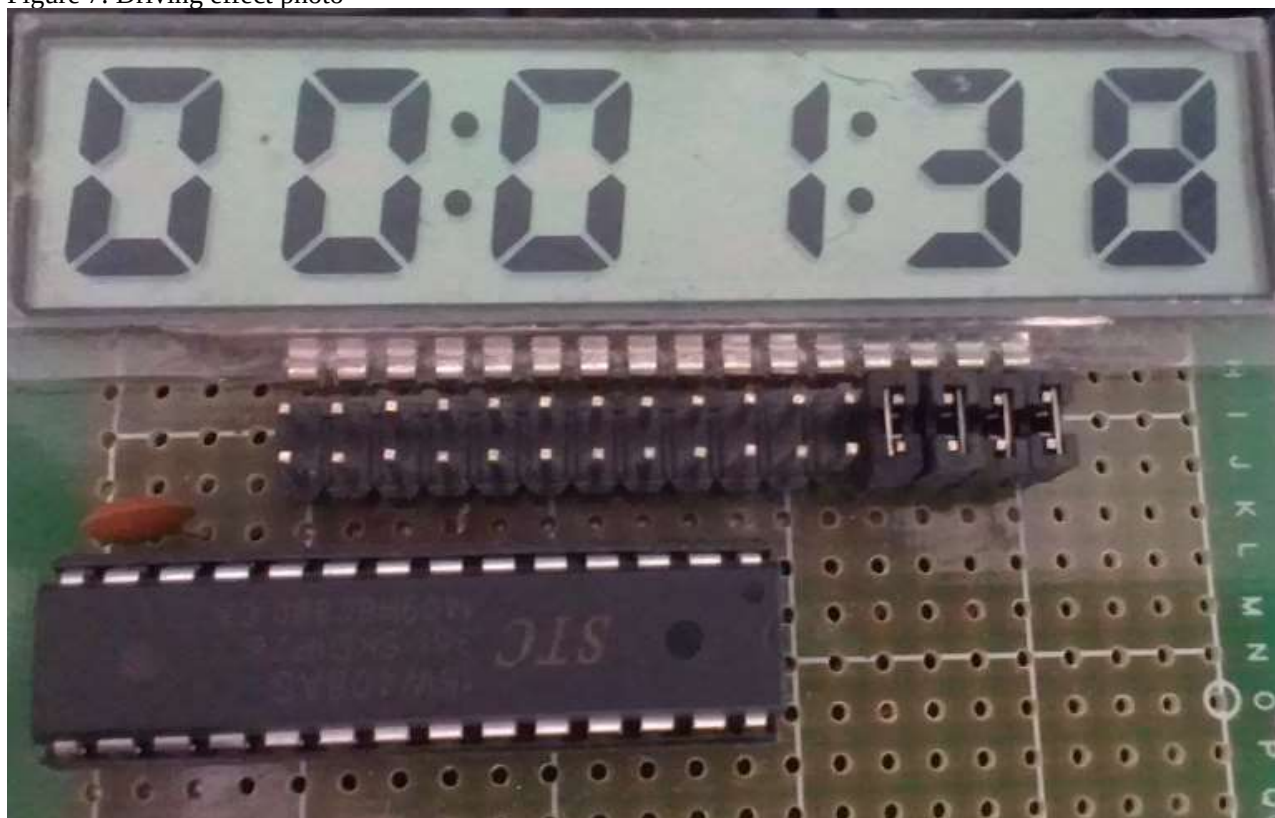
LCD真值表:

MCU PIN	P17	P16	P15	P14	P13	P12	P11	P10	P27	P26	P25	P24	P23	P22	P21	P20
LCD PIN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
LCD PIN name	SEG11	SEG10	SEG9	SEG8	SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0	COM3	COM2	COM1	COM0
--	1D	2:	2D	2.	3D	4:	4D	4.	5D	5.	6D		COM3			
1E	1C	2E	2C	3E	3C	4E	4C	5E	5C	6E	6C			COM2		
1G	1B	2G	2B	3G	3B	4G	4B	5G	5B	6G	6B				COM1	
1F	1A	2F	2A	3F	3A	4F	4A	5F	5A	6F	6A					COM0

显存影射表: |

	B7	B6	B5	B4	B3	B2	B1	B0
buff[0]:	--	1D	2:	2D	2.	3D	4:	4D
buff[1]:	1E	1C	2E	2C	3E	3C	4E	4C
buff[2]:	1G	1B	2G	2B	3G	3B	4G	4B
buff[3]:	1F	1A	2F	2A	3F	3A	4F	4A
buff[4]:	4.	5D	5.	6D	--	--	--	--
buff[5]:	5E	5C	6E	6C	--	--	--	--
buff[6]:	5G	5B	6G	6B	--	--	--	--
buff[7]:	5F	5A	6F	6A	--	--	--	--

Figure 7: Driving effect photo



Only two functions are required in the LCD scanning program:

1. LCD segment code scan function

```
void LCD_scan(void)
```

The program calls this function at a certain interval, and it will display the contents of the LCD display buffer on the LCD. It takes 8 calling cycles to scan all of them. The calling interval is generally 1~2ms. The scanning cycle is 8ms if 1ms is used. The refresh rate is 125Hz.

2. LCD segment code display buffer loading function

```
void LCD_load(u8 n,u8 dat)
```

This function is used to put the displayed numbers or characters in the LCD display buffer. For example, LCD_load (1,6) is to display the number 6 at the first digit position. It supports the display of 0 ~ 9, A ~ F. You can add them by yourself if you need other characters.

In addition, macros are used to display, extinguish, or flash colons or decimal points.

C language code

```
/******Function description*****
```

STC15 series of microcontrollers are used to test segment LCD driven by I/O directly (6 8-word LCDs, 1/4 Dutys, 1/3 bias).

Time (hours, minutes and seconds) is displayed after power-on.

P3.2 is connected to ground via a switch to enter sleep or wake up.

```
*****/
```

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

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

```
typedef unsigned char u8;
```

```
typedef unsigned int u16;
```

```
typedef unsigned long u32;
```

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

```
sfr P1M1 = 0x91;
```

```
sfr P1M0 = 0x92;
```

```
sfr P2M1 = 0x95;
```

```
sfr P2M0 = 0x96;
```

```
/******Local constant declaration******/
```

```
#define MAIN_Fosc          11059200L           //Define the main clock
```

```
#define DIS_BLACK          0x10
```

```
#define DIS_                0x11
```

```
#define DIS_A              0x0A
```

```
#define DIS_B              0x0B
```

```
#define DIS_C              0x0C
```

```
#define DIS_D              0x0D
```

```
#define DIS_E              0x0E
```

```
#define DIS_F              0x0F
```

```
#define LCD_SET_DP2        LCD_buff[0] |= 0x08
```

```
#define LCD_CLR_DP2        LCD_buff[0] &= ~0x08
```

```
#define LCD_FLASH_DP2      LCD_buff[0] ^= 0x08
```

```
#define LCD_SET_DP4        LCD_buff[4] |= 0x80
```

```
#define LCD_CLR_DP4        LCD_buff[4] &= ~0x80
```

```
#define LCD_FLASH_DP4      LCD_buff[4] ^= 0x80
```

```
#define LCD_SET_2M         LCD_buff[0] |= 0x20
```

```
#define LCD_CLR_2M         LCD_buff[0] &= ~0x20
```

```
#define LCD_FLASH_2M       LCD_buff[0] ^= 0x20
```

```
#define LCD_SET_4M         LCD_buff[0] |= 0x02
```

```
#define LCD_CLR_4M         LCD_buff[0] &= ~0x02
```

```
#define LCD_FLASH_4M       LCD_buff[0] ^= 0x02
```

```
#define LCD_SET_DP5        LCD_buff[4] |= 0x20
```

```
#define LCD_CLR_DP5        LCD_buff[4] &= ~0x20
```

```
#define LCD_FLASH_DP5      LCD_buff[4] ^= 0x20
```

```
#define P1n_standard(bitn) P1M1 &= ~(bitn), P1M0 &= ~(bitn)
```

```
#define P1n_push_pull(bitn) P1M1 &= ~(bitn), P1M0 |= (bitn)
```

```
#define P1n_pure_input(bitn) P1M1 |= (bitn), P1M0 &= ~(bitn)
```

```
#define P1n_open_drain(bitn) P1M1 |= (bitn), P1M0 |= (bitn)
```

```
#define P2n_standard(bitn) P2M1 &= ~(bitn), P2M0 &= ~(bitn)
```

```
#define P2n_push_pull(bitn) P2M1 &= ~(bitn), P2M0 |= (bitn)
```

```
#define P2n_pure_input(bitn) P2M1 |= (bitn), P2M0 &= ~(bitn)
```

```
#define P2n_open_drain(bitn) P2M1 |= (bitn), P2M0 |= (bitn)
```

```
/******Local variable declaration******/
```

```
u8 cnt_500ms;
```

```
u8 second,minute,hour;
```

```
bit B_Second;
```

```
bit B_2ms;
```

```
u8 LCD_buff[8];
```

```
u8 scan_index;
```

```
/******Local function declaration******/
```

```
void LCD_load(u8 n,u8 dat);
```

```
void LCD_scan(void);
```

```
void LoadRTC(void);
```

```
void delay_ms(u8 ms);
```

```
/******main function******/
```

```
void main(void)
```

```

{
    u8 i;

    AUXR = 0x80;
    TMOD = 0x00;
    TL0 = (65536 - (MAIN_Fosc / 500));
    TH0 = (65536 - (MAIN_Fosc / 500)) >> 8;
    TR0 = 1;
    ET0 = 1;
    EA = 1;

    //Initialize LCD memory
    for(i=0; i<8; i++) LCD_buff[i] = 0;
    P2n_push_pull(0xf0);
    P1n_push_pull(0xff); //segment is set as push-pull output mode

    LCD_SET_2M; //Display hour-minute division interval:
    LCD_SET_4M; //Display minute-second division interval:
    LoadRTC(); //Display time

    while (1)
    {
        PCON |= 0x01; //Enter Idle mode, wake up and exit by Timer0 2ms
        _nop_0;
        _nop_0;
        _nop_0;

        if(B_2ms) //2ms beat
        {
            B_2ms = 0;

            if(++cnt_500ms >= 250) //reach to 500ms
            {
                cnt_500ms = 0;
                // LCD_FLASH_2M; //Flashing hour-minute interval:
                // LCD_FLASH_4M; //Flashing minute-second interval:

                B_Second = ~B_Second;
                if(B_Second)
                {
                    if(++second >= 60) //reach to 1 minute
                    {
                        second = 0;
                        if(++minute >= 60) //reach to 1 hour
                        {
                            minute = 0;
                            if(++hour >= 24) hour = 0; //reach to 24 hours
                        }
                    }
                    LoadRTC(); //Display time
                }
            }
        }

        if(!INT0) //key is pressed, ready to sleep
        {
            LCD_CLR_2M; //Display hour-minute division interval:
            LCD_CLR_4M; //Display minute-second division interval:
            LCD_load(1,DIS_BLACK);
            LCD_load(2,DIS_BLACK);
            LCD_load(3,0);
        }
    }
}

```



```

LCD_load(4,0x0F);
LCD_load(5,0x0F);
LCD_load(6,DIS_BLACK);

while(!INT0) delay_ms(10);           //Waiting for the key to be released
delay_ms(50);
while(!INT0) delay_ms(10);           //Waiting for the key to be released once more

TR0 = 0;                             //关闭定时器
IE0 = 0;                             //外中断0 标志位
EX0 = 1;                             //INT0 Enable
IT0 = 1;                             //INT0 下降沿中断

P1n_push_pull(0xff);                //com 和 seg 全部输出 0
P2n_push_pull(0xff);
P1 = 0;
P2 = 0;

PCON |= 0x02;                        //Sleep
_nop_0;
_nop_0;
_nop_0;

LCD_SET_2M;                          //Display hour-minute division interval:
LCD_SET_4M;                          //Display minute-second division interval:
LoadRTC();                          //Display time
TR0 = 1;                             //Open the timer
while(!INT0) delay_ms(10);           //Waiting for the key to be released
delay_ms(50);
while(!INT0) delay_ms(10);           //Waiting for the key to be released once more
    }
}
}

/*****delay function*****/
void delay_ms(u8 ms)
{
    unsigned int i;
    do{
        i = MAIN_Fosc / 13000;
        while(--i);                  //14T per loop
    }while(--ms);
}

/***** Timer0 interrupt function *****/
void timer0_int (void) interrupt 1
{
    LCD_scan();
    B_2ms = 1;
}

/***** INT0 interrupt function *****/
void INT0_int (void) interrupt 0
{
    EX0 = 0;
    IE0 = 0;
}

/***** LCD Segment code scan function *****/

```

```

void LCD_scan(void)                                     //5us @22.1184MHZ
{
    u8 code T_COM[4]={0x08,0x04,0x02,0x01};
    u8 j;

    j = scan_index >> 1;
    P2n_pure_input(0x0f);    //All COM outputs are high impedance, and COM 's voltage is the midpoint
    if(scan_index & 1)       // Reverse scan
    {
        P1 = ~LCD_buff[j];
        P2 = ~(LCD_buff[j]>4] & 0xf0);
    }
    else                     // Normal phase scan
    {
        P1 = LCD_buff[j];
        P2 = LCD_buff[j>4] & 0xf0;
    }
    P2n_push_pull(T_COM[j]); //A COM is set as push-pull output mode
    if(++scan_index >= 8) scan_index = 0;
}

/***** Load display functions for numbers 1 to 6 *****/
void LCD_load(u8 n, u8 dat)                             // n is the number, dat is the number to be displayed
{
    u8 code t_display[]={                                // Standard font
        // 0 1 2 3 4 5 6 7 8 9 A B C D E F
        0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F,0x77,0x7C,0x39,0x5E,0x79,0x71,
        //black -
        0x00,0x40
    };
    u8 code T_LCD_mask[4] = {~0xc0,~0x30,~0x0c,~0x03};
    u8 code T_LCD_mask4[4] = {~0x40,~0x10,~0x04,~0x01};
    u8 i,k;
    u8 *p;

    if((n == 0) || (n > 6)) return;
    i = t_display[dat];

    if(n <= 4)                                           //1~4
    {
        n--;
        p = LCD_buff;
    }
    else
    {
        n = n - 5;
        p = &LCD_buff[4];
    }

    k = 0;
    if(i & 0x08) k |= 0x40;                               //D
    *p = (*p & T_LCD_mask4[n]) | (k>>2*n);
    p++;

    k = 0;
    if(i & 0x04) k |= 0x40;                               //C
    if(i & 0x10) k |= 0x80;                               //E
    *p = (*p & T_LCD_mask4[n]) | (k>>2*n);
    p++;
}

```

```

    k = 0;
    if(i & 0x02) k |= 0x40;           //B
    if(i & 0x40) k |= 0x80;           //G
    *p = (*p & T_LCD_mask[n]) | (k>>2*n);
    p++;

    k = 0;
    if(i & 0x01) k |= 0x40;           //A
    if(i & 0x20) k |= 0x80;           //F
    *p = (*p & T_LCD_mask[n]) | (k>>2*n);
}

/*****Display time *****/
void LoadRTC(void)
{
    LCD_load(1,hour/10);
    LCD_load(2,hour%10);
    LCD_load(3,minute/10);
    LCD_load(4,minute%10);
    LCD_load(5,second/10);
    LCD_load(6,second%10);
}

```

Assembly code

;STC8 series of microcontrollers are used to test segment LCD driven by I/O directly (6 8-word LCDs, 1/4 Dutys, 1/3 bias).
;Time (hours, minutes and seconds) is displayed after power-on.

```

;*****
P0M1      DATA      0x93
P0M0      DATA      0x94
P1M1      DATA      0x91
P1M0      DATA      0x92
P2M1      DATA      0x95
P2M0      DATA      0x96
P3M1      DATA      0xB1
P3M0      DATA      0xB2
P4M1      DATA      0xB3
P4M0      DATA      0xB4
P5M1      DATA      0xC9
P5M0      DATA      0xC
P6M1      DATA      0xCB
P6M0      DATA      0xCC
P7M1      DATA      0xE1
P7M0      DATA      0xE2
AUXR      DATA      0x8E
INT_CLKO  DATA      0x8F
IE2       DATA      0xAF
P4        DATA      0xC0
T2H       DATA      0xD6
T2L       DATA      0xD7

;*****
DIS_BLACK EQU         010H
DIS_      EQU         011H
DIS_A     EQU         00AH
DIS_B     EQU         00BH
DIS_C     EQU         00CH

```

```
DIS_D EQU 00DH
DIS_E EQU 00EH
DIS_F EQU 00FH
```

```
B_2ms BIT 20H.0 ;2ms signal
B_Second BIT 20H.1 ;second signal
cnt_500ms DATA 30H
second DATA 31H
minute DATA 32H
hour DATA 33H
scan_index DATA 34H
```

```
LCD_buff DATA 40H ;40H~47H
```

```
*****
```

```
ORG 0000H
LJMP F_Main
```

```
ORG 000BH
LJMP F_Timer0_Interrupt
```

```
*****
```

```
F_Main: ORG 0100H
```

```
CLR A
MOV P3M1, A ;Set as a quasi-bidirectional port
MOV P3M0, A
MOV P5M1, A ;Set as a quasi-bidirectional port
MOV P5M0, A

MOV P1M1, #0 ; segments are set as push-pull output mode
MOV P1M0, #0ffh
ANL P2M1, #NOT 0f0h ; segments are set as push-pull output mode
ORL P2M0, #0f0h
ORL P2M1, #00fH ; All COM outputs are high impedance, and COM's voltage is the midpoint
ANL P2M0, #0f0H
MOV SP, #0D0H
MOV PSW, #0
USING 0 ;Select bank0 R0 ~ R7
```

```
*****
```

```
MOV R2, #8
MOV R0, #LCD_buff
```

```
L_ClearLcdRam:
```

```
MOV @R0, #0
INC R0
DJNZ R2, L_ClearLcdRam
```

```
LCALL F_Timer0_init
SETB EA
```

```
; ORL LCD_buff, #020H ;Display hour-minute division interval:
; ORL LCD_buff, #002H ;Display minute-second division interval:
```

```
MOV hour, #12
MOV minute, #00
MOV second, #00
LCALL F_LoadRTC ;Display time
```

```
*****
```

L_Main_Loop:

```

JNB      B_2ms, L_Main_Loop      ;2ms beat
CLR      B_2ms

INC      cnt_500ms
MOV      A, cnt_500ms
CJNE     A, #250, L_Main_Loop
;reach to 500ms

MOV      cnt_500ms, #0;

XRL      LCD_buff, #020H          ;Flashing hour-minute interval:
XRL      LCD_buff, #002H          ;Flashing minute-second interval:

CPL      B_Second
JNB      B_Second, L_Main_Loop

INC      second
MOV      A, second
CJNE     A, #60, L_Main_Load
MOV      second, #0              ; reach to 1 minute
INC      minute
MOV      A, minute
CJNE     A, #60, L_Main_Load
MOV      minute, #0;
INC      hour
MOV      A, hour
CJNE     A, #24, L_Main_Load
MOV      hour, #0                ;reach to 24 hours

```

L_Main_Load:

```

LCALL    F_LoadRTC                ;Display time
LJMP     L_Main_Loop

```

F_Timer0_init:

```

CLR      TR0                      ; Stop counting
ANL      TMOD, #0f0H
SETB     ET0                      ; Enable interrupt
ORL      TMOD, #0                ; Working mode 0: 16-bit auto-reload
ANL      INT_CLKO, #NOT 0x01      ; Does not output clock
ORL      AUXR, #0x80             ; 1T mode
MOV      TH0, #HIGH (-22118)     ; 2ms
MOV      TL0, #LOW (-22118)      ;
SETB     TR0                    ; Start operation
RET

```

F_Timer0_Interrupt: ;Timer0 1ms interrupt function

```

PUSH     PSW                      ;push PSW into stack
PUSH     ACC                      ;push ACC into stack
PUSH     AR0
PUSH     AR7
PUSH     DPH
PUSH     DPL

LCALL    F_LCD_scan
SETB     B_2ms

POP      DPL
POP      DPH

```

```

POP      AR7
POP      AR0
POP      ACC      ;pop ACC from stack
POP      PSW      ;pop PSW from stack
RETI

```

***** Display time *****

F_LoadRTC:

```

MOV      R6, #1      ;LCD_load(1,hour/10);
MOV      A, hour
MOV      B, #10
DIV      AB
MOV      R7, A
LCALL    F_LCD_load   ;R6 is the number, which is 1~6, R7 is the number to be displayed

MOV      R6, #2      ;LCD_load(2,hour%10);
MOV      A, hour
MOV      B, #10
DIV      AB
MOV      R7, B
LCALL    F_LCD_load   ;R6 is the number, which is 1~6, R7 is the number to be displayed

MOV      R6, #3      ;LCD_load(3,minute/10);
MOV      A, minute
MOV      B, #10
DIV      AB
MOV      R7, A
LCALL    F_LCD_load   ;R6 is the number, which is 1~6, R7 is the number to be displayed

MOV      R6, #4      ;LCD_load(4,minute%10);
MOV      A, minute
MOV      B, #10
DIV      AB
MOV      R7, B
LCALL    F_LCD_load   ;R6 is the number, which is 1~6, R7 is the number to be displayed

MOV      R6, #5      ;LCD_load(5,second/10);
MOV      A, second
MOV      B, #10
DIV      AB
MOV      R7, A
LCALL    F_LCD_load   ;R6 is the number, which is 1~6, R7 is the number to be displayed

MOV      R6, #6      ;LCD_load(6,second%10);
MOV      A, second
MOV      B, #10
DIV      AB
MOV      R7, B
LCALL    F_LCD_load   ;R6 is the number, which is 1~6, R7 is the number to be displayed

RET

```

T_COM:

```

DB      008H, 004H, 002H, 001H

```

F_LCD_scan:

```

MOV      A, scan_index ;j = scan_index >> 1;
CLR      C
RRC      A

```

```

MOV      R7, A                      ;R7 = j
ADD      A, #LCD_buff
MOV      R0, A                      ;R0 = LCD_buff[j]
ORL      P2M1, #00fH                ;All COM outputs are high impedance, and COM 's voltage is the
midpoint
ANL      P2M0, #0f0H

MOV      A, scan_index
JNB      ACC.0, L_LCD_Scan2          ;if(scan_index & 1) // Reverse scan
MOV      A, @R0                     ;P1 = ~LCD_buff[j];
CPL      A
MOV      P1, A
MOV      A, R0                      ;P2 = ~(LCD_buff[j|4] & 0xf0);
ADD      A, #4
MOV      R0, A
MOV      A, @R0
ANL      A, #0f0H
CPL      A
MOV      P2, A
SJMP     L_LCD_Scan3

L_LCD_Scan2:                          ; Normal phase scan
MOV      A, @R0                     ;P1 = LCD_buff[j];
MOV      P1, A
MOV      A, R0                      ;P2 = (LCD_buff[j|4] & 0xf0);
ADD      A, #4
MOV      R0, A
MOV      A, @R0
ANL      A, #0f0H
MOV      P2, A

L_LCD_Scan3:
MOV      DPTR, #T_COM                ;A COM is set as push-pull output mode
MOV      A, R7
MOVC     A, @A+DPTR
ORL      P2M0, A
CPL      A
ANL      P2M1, A

INC      scan_index                  ;if(++scan_index == 8)    scan_index = 0;
MOV      A, scan_index
CJNE     A, #8, L_QuitLcdScan
MOV      scan_index, #0

L_QuitLcdScan:
RET

;***** Standard font*****
T_Display:
;      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
DB      03FH,006H,05BH,04FH,066H,06DH,07DH,007H,07FH,06FH,077H,07CH,039H,05EH,079H,071H
;      black -
DB      000H,040H

;***** Load display functions for numbers 1 to 6, and the algorithm is simple *****
F_LCD_load:
MOV      DPTR, #T_Display            ;R6 is the number, which is 1~6, R7 is the number to be displayed
MOV      A, R7                      ;i = t_display[dat];
MOVC     A, @A+DPTR
MOV      B, A                        ;the number to be displayed

```

```

MOV      A, R6
CJNE     A, #1, L_NotLoadChar1
MOV      R0, #LCD_buff
MOV      A, @R0
MOV      C, B.3      ;D
MOV      ACC.6, C
MOV      @R0, A

INC      R0
MOV      A, @R0
MOV      C, B.2      ;C
MOV      ACC.6, C
MOV      C, B.4      ;E
MOV      ACC.7, C
MOV      @R0, A

INC      R0
MOV      A, @R0
MOV      C, B.1      ;B
MOV      ACC.6, C
MOV      C, B.6      ;G
MOV      ACC.7, C
MOV      @R0, A

INC      R0
MOV      A, @R0
MOV      C, B.0      ;A
MOV      ACC.6, C
MOV      C, B.5      ;F
MOV      ACC.7, C
MOV      @R0, A
RET

```

L_NotLoadChar1:

```

CJNE     A, #2, L_NotLoadChar2
MOV      R0, #LCD_buff
MOV      A, @R0
MOV      C, B.3      ;D
MOV      ACC.4, C
MOV      @R0, A

INC      R0
MOV      A, @R0
MOV      C, B.2      ;C
MOV      ACC.4, C
MOV      C, B.4      ;E
MOV      ACC.5, C
MOV      @R0, A

INC      R0
MOV      A, @R0
MOV      C, B.1      ;B
MOV      ACC.4, C
MOV      C, B.6      ;G
MOV      ACC.5, C
MOV      @R0, A

INC      R0
MOV      A, @R0

```



```

MOV      C, B.0                ;A
MOV      ACC.4, C
MOV      C, B.5                ;F
MOV      ACC.5, C
MOV      @R0, A
RET

```

L_NotLoadChar2:

```

CJNE     A, #3, L_NotLoadChar3
MOV      R0, #LCD_buff
MOV      A, @R0
MOV      C, B.3                ;D
MOV      ACC.2, C
MOV      @R0, A

INC      R0
MOV      A, @R0
MOV      C, B.2                ;C
MOV      ACC.2, C
MOV      C, B.4                ;E
MOV      ACC.3, C
MOV      @R0, A

INC      R0
MOV      A, @R0
MOV      C, B.1                ;B
MOV      ACC.2, C
MOV      C, B.6                ;G
MOV      ACC.3, C
MOV      @R0, A

INC      R0
MOV      A, @R0
MOV      C, B.0                ;A
MOV      ACC.2, C
MOV      C, B.5                ;F
MOV      ACC.3, C
MOV      @R0, A
RET

```

L_NotLoadChar3:

```

CJNE     A, #4, L_NotLoadChar4
MOV      R0, #LCD_buff
MOV      A, @R0
MOV      C, B.3                ;D
MOV      ACC.0, C
MOV      @R0, A

INC      R0
MOV      A, @R0
MOV      C, B.2                ;C
MOV      ACC.0, C
MOV      C, B.4                ;E
MOV      ACC.1, C
MOV      @R0, A

INC      R0
MOV      A, @R0
MOV      C, B.1                ;B
MOV      ACC.0, C

```

```

MOV      C, B.6                ;G
MOV      ACC.1, C
MOV      @R0, A

INC      R0
MOV      A, @R0
MOV      C, B.0                ;A
MOV      ACC.0, C
MOV      C, B.5                ;F
MOV      ACC.1, C
MOV      @R0, A
RET

```

L_NotLoadChar4:

```

CJNE     A, #5, L_NotLoadChar5
MOV      R0, #LCD_buff+4
MOV      A, @R0
MOV      C, B.3                ;D
MOV      ACC.6, C
MOV      @R0, A

INC      R0
MOV      A, @R0
MOV      C, B.2                ;C
MOV      ACC.6, C
MOV      C, B.4                ;E
MOV      ACC.7, C
MOV      @R0, A

INC      R0
MOV      A, @R0
MOV      C, B.1                ;B
MOV      ACC.6, C
MOV      C, B.6                ;G
MOV      ACC.7, C
MOV      @R0, A

INC      R0
MOV      A, @R0
MOV      C, B.0                ;A
MOV      ACC.6, C
MOV      C, B.5                ;F
MOV      ACC.7, C
MOV      @R0, A
RET

```

L_NotLoadChar5:

```

CJNE     A, #6, L_NotLoadChar6
MOV      R0, #LCD_buff+4
MOV      A, @R0
MOV      C, B.3                ;D
MOV      ACC.4, C
MOV      @R0, A

INC      R0
MOV      A, @R0
MOV      C, B.2                ;C
MOV      ACC.4, C
MOV      C, B.4                ;E
MOV      ACC.5, C

```

MOV @R0, A

INC R0

MOV A, @R0

MOV C, B.1 ;B

MOV ACC.4, C

MOV C, B.6 ;G

MOV ACC.5, C

MOV @R0, A

INC R0

MOV A, @R0

MOV C, B.0 ;A

MOV ACC.4, C

MOV C, B.5 ;F

MOV ACC.5, C

MOV @R0, A

RET

L_NotLoadChar6:

RET

E

N

D

STC MCU

10 Instruction Set

Mnemonic	Description	Bytes	Cycle
ADD A,Rn	Add register to Accumulator	1	1
ADD A,direct	Add direct byte to Accumulator	2	1
ADD A,@Ri	Add indirect RAM to Accumulator	1	1
ADD A,#data	Add immediate data to Accumulator	2	1
ADDC A,Rn	Add register to Accumulator with Carry	1	1
ADDC A,direct	Add direct byte to Accumulator with Carry	2	1
ADDC A,@Ri	Add indirect RAM to Accumulator with Carry	1	1
ADDC A,#data	Add immediate data to Accumulator with Carry	2	1
SUBB A,Rn	Subtract Register from Accumulator with borrow	1	1
SUBB A,direct	Subtract direct byte from Accumulator with borrow	2	1
SUBB A,@Ri	Subtract indirect RAM from Accumulator with borrow	1	1
SUBB A,#data	Subtract immediate data from Accumulator with borrow	2	1
INC A	Increment Accumulator	1	1
INC Rn	Increment register	1	1
INC direct	Increment direct byte	2	1
INC @Ri	Increment indirect RAM	1	1
DEC A	Decrement Accumulator	1	1
DEC Rn	Decrement Register	1	1
DEC direct	Decrement direct byte	2	1
DEC @Ri	Decrement indirect RAM	1	1
INC DPTR	Increment Data Pointer	1	1
MUL AB	Multiply A & B, high byte of result is in B, low byte in A	1	2
DIV AB	Divide A by B, quotient is in A, remainder is in B.	1	6
DA A	Decimal Adjust Accumulator	1	3
ANL A,Rn	AND Register to Accumulator	1	1
ANL A,direct	AND direct byte to Accumulator	2	1
ANL A,@Ri	AND indirect RAM to Accumulator	1	1
ANL A,#data	AND immediate data to Accumulator	2	1
ANL direct,A	AND Accumulator to direct byte	2	1
ANL direct,#data	AND immediate data to direct byte	3	1
ORL A,Rn	OR register to Accumulator	1	1
ORL A,direct	OR direct byte to Accumulator	2	1
ORL A,@Ri	OR indirect RAM to Accumulator	1	1
ORL A,#data	OR immediate data to Accumulator	2	1
ORL direct,A	OR Accumulator to direct byte	2	1
ORL direct,#data	OR immediate data to direct byte	3	1
XRL A,Rn	Exclusive-OR register to Accumulator	1	1
XRL A,direct	Exclusive-OR direct byte to Accumulator	2	1
XRL A,@Ri	Exclusive-OR indirect RAM to Accumulator	1	1
XRL A,#data	Exclusive-OR immediate data to Accumulator	2	1

XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	1
XRL	direct,#data	Exclusive-OR immediate data to direct byte	3	1
CLR	A	Clear Accumulator	1	1
CPL	A	Complement Accumulator	1	1
RL	A	Rotate Accumulator Left	1	1
RLC	A	Rotate Accumulator Left through the Carry	1	1
RR	A	Rotate Accumulator Right	1	1
RRC	A	Rotate Accumulator Right through the Carry	1	1
SWAP	A	Swap nibbles within the Accumulator	1	1
CLR	C	Clear Carry	1	1
CLR	bit	Clear direct bit	2	1
SETB	C	Set Carry	1	1
SETB	bit	Set direct bit	2	1
CPL	C	Complement Carry	1	1
CPL	bit	Complement direct bit	2	1
ANL	C,bit	AND direct bit to Carry	2	1
ANL	C,/bit	AND complement of direct bit to Carry	2	1
ORL	C,bit	OR direct bit to Carry	2	1
ORL	C,/bit	OR complement of direct bit to Carry	2	1
MOV	C,bit	Move direct bit to Carry	2	1
MOV	bit,C	Move Carry to direct bit	2	1
MOV	A,Rn	Move register to Accumulator	1	1
MOV	A,direct	Move direct byte to Accumulator	2	1
MOV	A,@Ri	Move indirect RAM to Accumulator	1	1
MOV	A,#data	Move immediate data to Accumulator	2	1
MOV	Rn,A	Move Accumulator to register	1	1
MOV	Rn,direct	Move direct byte to register	2	1
MOV	Rn,#data	Move immediate data to register	2	1
MOV	direct,A	Move Accumulator to direct byte	2	1
MOV	direct,Rn	Move register to direct byte	2	1
MOV	direct,direct	Move direct byte to direct	3	1
MOV	direct,@Ri	Move indirect RAM to direct byte	2	1
MOV	direct,#data	Move immediate data to direct byte	3	1
MOV	@Ri,A	Move Accumulator to indirect RAM	1	1
MOV	@Ri,direct	Move direct byte to indirect RAM	2	1
MOV	@Ri,#data	Move immediate data to indirect RAM	2	1
MOV	DPTR,#data16	Move 16-bit immediate data to indirect RAM	3	1
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to Accumulator	1	4
MOVC	A,@A+PC	Move Code byte relative to PC to Accumulator	1	3
MOVX	A,@Ri	Move extended RAM(8-bit addr) to Accumulator (Read)	1	3 ^[1]
MOVX	A,@DPTR	Move extended RAM(16-bit addr) to Accumulator (Read)	1	2 ^[1]
MOVX	@Ri,A	Move Accumulator to extended RAM(8-bit addr) (Write)	1	3 ^[1]
MOVX	@DPTR,A	Move Accumulator to extended RAM(16-bit addr) (Write)	1	2 ^[1]
PUSH	direct	Push direct byte onto stack	2	1
POP	direct	POP direct byte from stack	2	1
XCH	A,Rn	Exchange register with Accumulator	1	1
XCH	A,direct	Exchange direct byte with Accumulator	2	1
XCH	A,@Ri	Exchange indirect RAM with Accumulator	1	1

XCHD	A,@Ri	Exchange low-order Digit indirect RAM with Accumulator	1	1
ACALL	addr11	Absolute Subroutine Call	2	3
LCALL	addr16	Long Subroutine Call	3	3
RET		Return from Subroutine	1	3
RETI		Return from interrupt	1	3
AJMP	addr11	Absolute Jump	2	3
LJMP	addr16	Long Jump	3	3
SJMP	rel	Short Jump (relative addr)	2	3
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	4
JZ	rel	Jump if Accumulator is Zero	2	1/3 ^[2]
JNZ	rel	Jump if Accumulator is not Zero	2	1/3 ^[2]
JC	rel	Jump if Carry is set	2	1/3 ^[2]
JNC	rel	Jump if Carry not set	2	1/3 ^[2]
JB	bit,rel	Jump if direct bit is set	3	1/3 ^[2]
JNB	bit,rel	Jump if direct bit is not set	3	1/3 ^[2]
JBC	bit,rel	Jump if direct bit is set & clear bit	3	1/3 ^[2]
CJNE	A,direct,rel	Compare direct byte to Accumulator and jump if not equal	3	2/3 ^[3]
CJNE	A,#data,rel	Compare immediate data to Accumulator and Jump if not equal	3	1/3 ^[2]
CJNE	Rn,#data,rel	Compare immediate data to register and Jump if not equal	3	2/3 ^[3]
CJNE	@Ri,#data,rel	Compare immediate data to indirect and jump if not equal	3	2/3 ^[3]
DJNZ	Rn,rel	Decrement register and jump if not Zero	2	2/3 ^[3]
DJNZ	direct,rel	Decrement direct byte and Jump if not Zero	3	2/3 ^[3]
NOP		No Operation	1	1

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

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

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

11 Interrupt System

(An error will be reported when compiled in Keil when using an interrupt with an interrupt number greater than 31 in a C program. Please refer to Appendix for the solution.)

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

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

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

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

11.1 Interrupt sources of STC8H series

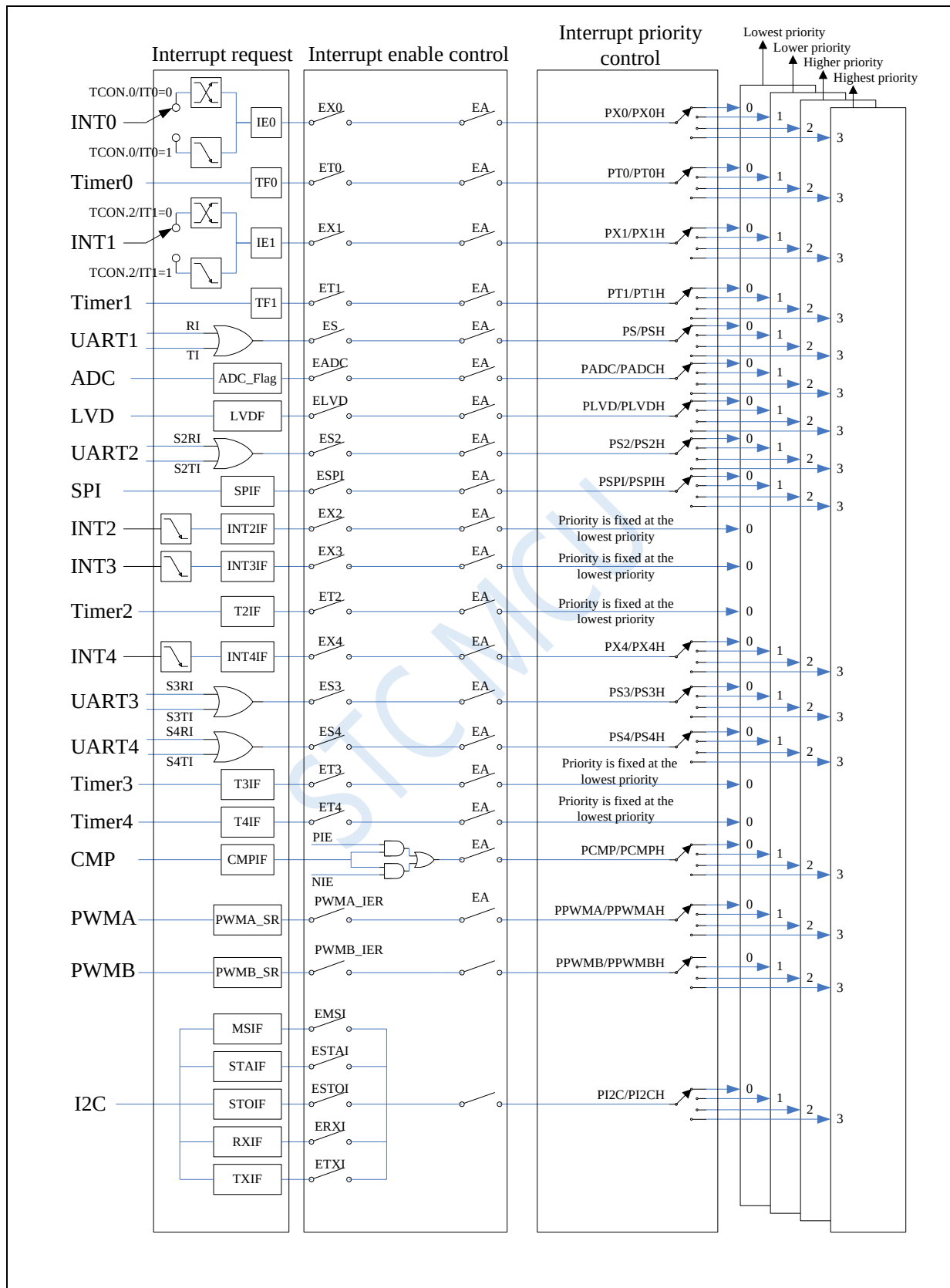
The ✓ in the following table indicates that the corresponding series have the corresponding interrupt source.

Interrupt sources	STC8H1K16 family	STC8H1K08 family	STC8H3K64S4 family	STC8H3K64S2 family	STC8H8K64U -A family	STC8H8K64U -B family	STC8H4K64TLR family	STC8H4K64TLC family
External interrupt 0 (INT0) Supports falling and edges interrupts	✓	✓	✓	✓	✓	✓	✓	✓
Timer 0 interrupt (Timer0)	✓	✓	✓	✓	✓	✓	✓	✓
External interrupt 1 (INT1) Supports falling and edges interrupts	✓	✓	✓	✓	✓	✓	✓	✓
Timer 1 interrupt (Timer1)	✓	✓	✓	✓	✓	✓	✓	✓
UART1 interrupt (UART1)	✓	✓	✓	✓	✓	✓	✓	✓
ADC interrupt (ADC)	✓	✓	✓	✓	✓	✓	✓	✓
Low voltage detection interrupt (LVD)	✓	✓	✓	✓	✓	✓	✓	✓
CCP/PCA/PWM interrupt (CCP/PCA)	✓	✓	✓	✓	✓	✓	✓	✓
UART2 interrupt (UART2)	✓	✓	✓	✓	✓	✓	✓	✓
SPI interrupt (SPI)	✓	✓	✓	✓	✓	✓	✓	✓
External interrupt 2 (INT2) Supports falling edge interrupts	✓	✓	✓	✓	✓	✓	✓	✓
External interrupt 3 (INT3) Supports falling edge interrupts	✓	✓	✓	✓	✓	✓	✓	✓
Timer 2 interrupt (Timer2)	✓	✓	✓	✓	✓	✓	✓	✓

External interrupt 4 (INT4)	✓	✓	✓	✓	✓	✓	✓	✓
UART3 interrupt (UART3)			✓		✓	✓	✓	✓
UART4 interrupt (UART4)			✓		✓	✓	✓	✓
Timer 3 interrupt (Timer3)	✓		✓	✓	✓	✓	✓	✓
Timer 4 interrupt (Timer4)	✓		✓	✓	✓	✓	✓	✓
Comparator interrupt (CMP)	✓	✓	✓	✓	✓	✓	✓	✓
I2C interrupt	✓	✓	✓	✓	✓	✓	✓	✓
PWMA	✓	✓	✓	✓	✓	✓	✓	✓
PWMB	✓	✓	✓	✓	✓	✓	✓	✓
USB interrupt					✓	✓		
Touch Key interrupt							✓	✓
RTC interrupt Supports falling edge, rising edge, high level and low level interrupts						✓	✓	✓
Port0 interrupt Supports falling edge, rising edge, high level and low level interrupts			✓	✓		✓	✓	✓
Port1 interrupt Supports falling edge, rising edge, high level and low level interrupts			✓	✓		✓	✓	✓
Port2 interrupt Supports falling edge, rising edge, high level and low level interrupts			✓	✓		✓	✓	✓
Port3 interrupt Supports falling edge, rising edge, high level and low level interrupts			✓	✓		✓	✓	✓
Port4 interrupt Supports falling edge, rising edge, high level and low level interrupts			✓	✓		✓	✓	✓
Port5 interrupt Supports falling edge, rising edge, high level and low level interrupts			✓	✓		✓	✓	✓
Port6 interrupt Supports falling edge, rising edge, high level and low level interrupts			✓	✓		✓		
Port7 interrupt Supports falling edge, rising edge, high level and low level interrupts			✓	✓		✓		
DMA_M2M interrupt						✓	✓	✓
DMA_ADC interrupt						✓	✓	✓
DMA_SPI interrupt						✓	✓	✓
DMA_UR1T interrupt						✓	✓	✓
DMA_UR1R interrupt						✓	✓	✓
DMA_UR2T interrupt						✓	✓	✓
DMA_UR2R interrupt						✓	✓	✓
DMA_UR3T interrupt						✓	✓	✓
DMA_UR3R interrupt						✓	✓	✓
DMA_UR4T interrupt						✓	✓	✓
DMA_UR4R interrupt						✓	✓	✓
DMA_LCM interrupt						✓	✓	✓
LCM interrupt						✓	✓	✓

STC MCU

11.2 Structure of STC8H Interrupt



11.3 Interrupt List of STC8H Series

Interrupt source	Interrupt vector	Order	Priority level setup bit	Priority level	Interrupt request flag	Interrupt enable bit
INT0	0003H	0	PX0,PX0H	0/1/2/3	IE0	EX0
Timer0	000BH	1	PT0,PT0H	0/1/2/3	TF0	ET0
INT1	0013H	2	PX1,PX1H	0/1/2/3	IE1	EX1
Timer1	001BH	3	PT1,PT1H	0/1/2/3	TF1	ET1
UART1	0023H	4	PS,PSH	0/1/2/3	RI TI	ES
ADC	002BH	5	PADC,PADCH	0/1/2/3	ADC_FLAG	EADC
LVD	0033H	6	PLVD,PLVDH	0/1/2/3	LVDF	ELVD
PCA	003BH	7	PPCA,PPCAH	0/1/2/3	CF	ECF
					CCF0	ECCF0
					CCF1	ECCF1
					CCF2	ECCF2
					CCF3	ECCF3
UART2	0043H	8	PS2,PS2H	0/1/2/3	S2RI S2TI	ES2
SPI	004BH	9	PSPI,PSPIH	0/1/2/3	SPIF	ESPI
INT2	0053H	10		0	INT2IF	EX2
INT3	005BH	11		0	INT3IF	EX3
Timer2	0063H	12		0	T2IF	ET2
INT4	0083H	16	PX4,PX4H	0/1/2/3	INT4IF	EX4
UART3	008BH	17	PS3,PS3H	0/1/2/3	S3RI S3TI	ES3
UART4	0093H	18	PS4,PS4H	0/1/2/3	S4RI S4TI	ES4
Timer3	009BH	19		0	T3IF	ET3
Timer4	00A3H	20		0	T4IF	ET4
CMP	00ABH	21	PCMP,PCMPH	0/1/2/3	CMPIF	PIE NIE
I2C	00C3H	24	PI2C,PI2CH	0/1/2/3	MSIF	EMSI
					STAIF	ESTAI
					RXIF	ERXI
					TXIF	ETXI
					STOIF	ESTOI
USB	00CBH	25	PUSB,PUSBH	0/1/2/3	USB Events	EUSB
PWMA	00D3H	26	PPWMA,PPWMAH	0/1/2/3	PWMA_SR	PWMA_IER
PWMB	00DBH	27	PPWMB,PPWMBH	0/1/2/3	PWMB_SR	PWMB_IER
TKSU	011BH	35	PTKSU,PTKSUH	0/1/2/3	TKIF	ETKSUI
RTC	0123H	36	PRTC,PRTCH	0/1/2/3	ALAI	EALAI
					DAYIF	EDAYI
					HOURLIF	EHOURI
					MINIF	EMINI
					SECIF	ESECI
					SEC2IF	ESEC2I
					SEC8IF	ESEC8I
					SEC32IF	ESEC32I

Interrupt source	Interrupt vector	Order	Priority level setup bit	Priority level	Interrupt request flag	Interrupt enable bit
P0 interrupt	012BH	37	PINIPL[0], PINIPH[0]	0/1/2/3	P0INTF	P0INTE
P1 interrupt	0133H	38	PINIPL[1], PINIPH[1]	0/1/2/3	P1INTF	P1INTE
P2 interrupt	013BH	39	PINIPL[2], PINIPH[2]	0/1/2/3	P2INTF	P2INTE
P3 interrupt	0143H	40	PINIPL[3], PINIPH[3]	0/1/2/3	P3INTF	P3INTE
P4 interrupt	014BH	41	PINIPL[4], PINIPH[4]	0/1/2/3	P4INTF	P4INTE
P5 interrupt	0153H	42	PINIPL[5], PINIPH[5]	0/1/2/3	P5INTF	P5INTE
P6 interrupt	015BH	43	PINIPL[6], PINIPH[6]	0/1/2/3	P6INTF	P6INTE
P7 interrupt	0163H	44	PINIPL[7], PINIPH[7]	0/1/2/3	P7INTF	P7INTE
DMA_M2M interrupt	017BH	47	M2MIP[1:0]	0/1/2/3	M2MIF	M2MIE
DMA_ADC interrupt	0183H	48	ADCIP[1:0]	0/1/2/3	ADCIF	ADCIE
DMA_SPI interrupt	018BH	49	SPIIP[1:0]	0/1/2/3	SPIIF	SPIIE
DMA_UR1T interrupt	0193H	50	UR1TIP[1:0]	0/1/2/3	UR1TIF	UR1TIE
DMA_UR1R interrupt	019BH	51	UR1RIP[1:0]	0/1/2/3	UR1RIF	UR1RIE
DMA_UR2T interrupt	01A3H	52	UR2TIP[1:0]	0/1/2/3	UR2TIF	UR2TIE
DMA_UR2R interrupt	01ABH	53	UR2RIP[1:0]	0/1/2/3	UR2RIF	UR2RIE
DMA_UR3T interrupt	01B3H	54	UR3TIP[1:0]	0/1/2/3	UR3TIF	UR3TIE
DMA_UR3R interrupt	01BBH	55	UR3RIP[1:0]	0/1/2/3	UR3RIF	UR3RIE
DMA_UR4T interrupt	01C3H	56	UR4TIP[1:0]	0/1/2/3	UR4TIF	UR4TIE
DMA_UR4R interrupt	01CBH	57	UR4RIP[1:0]	0/1/2/3	UR4RIF	UR3RIE
DMA_LCM interrupt	01D3H	58	LCMIP[1:0]	0/1/2/3	LCMIF	LCMIE
LCM interrupt	01DBH	59	LCMIFIP[1:0]	0/1/2/3	LCMIFIF	LCMIFIE

Interrupt service routine may be declared in C language as the following,

```

void INT0_Routine(void)    interrupt 0;
void TM0_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 I2C_Routine(void)     interrupt 24;
void USB_Routine(void)     interrupt 25;
void PWMA_Routine(void)    interrupt 26;
void PWMB_Routine(void)    interrupt 27;
//void TKSU_Routine(void) interrupt 35;
//void RTC_Routine(void) interrupt 36;
//void P0Int_Routine(void) interrupt 37;
//void P0Int_Routine(void) interrupt 37;
//void P0Int_Routine(void) interrupt 37;
//void P1Int_Routine(void) interrupt 38;
//void P2Int_Routine(void) interrupt 39;
//void P3Int_Routine(void) interrupt 40;
//void P4Int_Routine(void) interrupt 41;
//void P5Int_Routine(void) interrupt 42;
//void P6Int_Routine(void) interrupt 43;
//void P7Int_Routine(void) interrupt 44;

```

Interrupt service routines with interrupt numbers greater than 31 cannot be directly declared in C language. Please refer to the processing method in "Appendix J". Assembly language is not affected

11.4 Registers Related to Interrupt

Symbol	Description	Address	Bit Address and Symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
IE	Interrupt enable register	A8H	EA	ELVD	EADC	ES	ET1	EX1	ET0	EX0	0000,0000
IE2	Interrupt enable register2	AFH	EUSB	ET4	ET3	ES4	ES3	ET2	ESPI	ES2	0000,0000
INTCLKO	External interrupt and clock output control register	8FH	-	EX4	EX3	EX2	-	T2CLKO	T1CLKO	T0CLKO	x000,x000
IP	Interrupt Priority Low Byte	B8H	-	PLVD	PADC	PS	PT1	PX1	PT0	PX0	x000,0000
IPH	Interrupt Priority High Byte	B7H	-	PLVDH	PADCH	PSH	PT1H	PX1H	PT0H	PX0H	x000,0000
IP2	2nd Interrupt Priority register low byte	B5H	PUSB PTKSU	PI2C	PCMP	PX4	PPWMB	PPWMA	PSPI	PS2	0000,0000
IP2H	2nd Interrupt Priority register high byte	B6H	PUSBH PTKSUH	PI2CH	PCMPH	PX4H	PPWMBH	PPWMAH	PSPIH	PS2H	0000,0000
IP3	3nd Interrupt Priority register low byte	DFH	-	-	-	-	-	PRTC	PS4	PS3	xxxx,x000
IP3H	3nd Interrupt Priority Register High Byte	EEH	-	-	-	-	-	PRTCH	PS4H	PS3H	xxxx,x000
TCON	Timer 0 and 1 control register	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0000,0000
AUXINTIF	Extended External Interrupt Flag Register	EFH	-	INT4IF	INT3IF	INT2IF	-	T4IF	T3IF	T2IF	x000,x000
SCON	UART1 control register	98H	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	0000,0000
S2CON	UART2 control register	9AH	S2SM0	-	S2SM2	S2REN	S2TB8	S2RB8	S2TI	S2RI	0100,0000
S3CON	UART3 control register	ACH	S3SM0	S3ST3	S3SM2	S3REN	S3TB8	S3RB8	S3TI	S3RI	0000,0000
S4CON	Serial port 4 control register	84H	S4SM0	S4ST4	S4SM2	S4REN	S4TB8	S4RB8	S4TI	S4RI	0000,0000
PCON	Power control register	87H	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000
ADC_CONTR	ADC control register	BCH	ADC_POWER	ADC_START	ADC_FLAG	ADC_EPWMT	ADC_CHS[3:0]				000x,0000
SPSTAT	SPI Status register	CDH	SPIF	WCOL	-	-	-	-	-	-	00xx,xxxx
CMPCR1	Comparator Control Register 1	E6H	CMPEN	CMPIF	PIE	NIE	PIS	NIS	CMPOE	CMPRES	0000,0000
LCMIFCFG	LCM Interface Configuration Register	FE50H	LCMIFIE	-	LCMIFIP[1:0]		LCMIFDPS[1:0]		D16_D8	M68_I80	0x00,0000
LCMIFSTA	LCM Interface Status Register	FE53H	-	-	-	-	-	-	-	LCMIFIF	xxxx,xxx0
I2CMSCR	I ² C Master Control Register	FE81H	EMSI	-	-	-	MSCMD[3:0]				0xxx,0000
I2CMSST	I ² C Master Status Register	FE82H	MSBUSY	MSIF	-	-	-	-	MSACKI	MSACKO	00xx,xx00
I2CSLCR	I ² C Slave Control Register	FE83H	-	ESTAI	ERXI	ETXI	ESTOI	-	-	SLRST	x000,0xx0
I2CSLST	I ² C Slave Status Register	FE84H	SLBUSY	STAIF	RXIF	TXIF	STOIF	TXING	SLACKI	SLACKO	0000,0000
PWMA_IER	PWMA Interrupt enable register	FEC4H	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE	0000,0000
PWMA_SR1	PWMA Status register 1	FEC5H	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF	0000,0000
PWMA_SR2	PWMA Status register 2	FEC6H	-	-	-	CC4OF	CC3OF	CC2OF	CC1OF	-	xxx0,000x
PWMB_IER	PWMB Interrupt enable register	FEE4H	BIE	TIE	COMIE	CC8IE	CC7IE	CC6IE	CC5IE	UIE	0000,0000
PWMB_SR1	PWMB Status register 1	FEE5H	BIF	TIF	COMIF	CC8IF	CC7IF	CC6IF	CC5IF	UIF	0000,0000
PWMB_SR2	PWMB Status register 2	FEE6H	-	-	-	CC8OF	CC7OF	CC6OF	CC5OF	-	xxx0,000x
P0INTE	P0 Interrupt enable register	FD00H	P07INTE	P06INTE	P05INTE	P04INTE	P03INTE	P02INTE	P01INTE	P00INTE	0000,0000
P1INTE	P1 Interrupt enable register	FD01H	P17INTE	P16INTE	P15INTE	P14INTE	P13INTE	P12INTE	P11INTE	P10INTE	0000,0000
P2INTE	P2 Interrupt enable register	FD02H	P27INTE	P26INTE	P25INTE	P24INTE	P23INTE	P22INTE	P21INTE	P20INTE	0000,0000
P3INTE	P3 Interrupt enable register	FD03H	P37INTE	P36INTE	P35INTE	P34INTE	P33INTE	P32INTE	P31INTE	P30INTE	0000,0000
P4INTE	P4 Interrupt enable register	FD04H	P47INTE	P46INTE	P45INTE	P44INTE	P43INTE	P42INTE	P41INTE	P40INTE	0000,0000
P5INTE	P5 Interrupt enable register	FD05H	-	-	P55INTE	P54INTE	P53INTE	P52INTE	P51INTE	P50INTE	xx00,0000
P6INTE	P6 Interrupt enable register	FD06H	P67INTE	P66INTE	P65INTE	P64INTE	P63INTE	P62INTE	P61INTE	P60INTE	0000,0000
P7INTE	P7 Interrupt enable register	FD07H	P77INTE	P76INTE	P75INTE	P74INTE	P73INTE	P72INTE	P71INTE	P70INTE	0000,0000
P0INTF	P0 Interrupt flag register	FD10H	P07INTF	P06INTF	P05INTF	P04INTF	P03INTF	P02INTF	P01INTF	P00INTF	0000,0000
P1INTF	P1 Interrupt flag register	FD11H	P17INTF	P16INTF	P15INTF	P14INTF	P13INTF	P12INTF	P11INTF	P10INTF	0000,0000
P2INTF	P2 Interrupt flag register	FD12H	P27INTF	P26INTF	P25INTF	P24INTF	P23INTF	P22INTF	P21INTF	P20INTF	0000,0000
P3INTF	P3 Interrupt flag register	FD13H	P37INTF	P36INTF	P35INTF	P34INTF	P33INTF	P32INTF	P31INTF	P30INTF	0000,0000
P4INTF	P4 Interrupt flag register	FD14H	P47INTF	P46INTF	P45INTF	P44INTF	P43INTF	P42INTF	P41INTF	P40INTF	0000,0000
P5INTF	P5 Interrupt flag register	FD15H	-	-	P55INTF	P54INTF	P53INTF	P52INTF	P51INTF	P50INTF	xx00,0000
P6INTF	P6 Interrupt flag register	FD16H	P67INTF	P66INTF	P65INTF	P64INTF	P63INTF	P62INTF	P61INTF	P60INTF	0000,0000
P7INTF	P7 Interrupt flag register	FD17H	P77INTF	P76INTF	P75INTF	P74INTF	P73INTF	P72INTF	P71INTF	P70INTF	0000,0000
PIN1PL	I/O port interrupt priority low register	FD60H	P7IP	P6IP	P5IP	P4IP	P3IP	P2IP	P1IP	POIP	0000,0000
PIN1PH	I/O port interrupt priority high register	FD61H	P7IPH	P6IPH	P5IPH	P4IPH	P3IPH	P2IPH	P1IPH	POIPH	0000,0000
DMA_M2M_CFG	M2M_DMA configuration register	FA00H	M2MIE	-	TXACO	RXACO	M2MIP[1:0]		M2MPTY[1:0]		0x00,0000
DMA_ADC_CFG	ADC_DMA configuration register	FA10H	ADCIE	-	-	-	ADCMIP[1:0]		ADCPPTY[1:0]		0xxx,0000
DMA_SPL_CFG	SPI_DMA configuration register	FA20H	SPIIE	ACT_TX	ACT_RX	-	SPIIP[1:0]		SPIPTY[1:0]		000x,0000
DMA_UR1T_CFG	UR1T_DMA configuration register	FA30H	UR1TIE	-	-	-	UR1TIP[1:0]		UR1TPTY[1:0]		0xxx,0000
DMA_UR1R_CFG	UR1R_DMA configuration register	FA38H	UR1RIE	-	-	-	UR1RIP[1:0]		UR1RPTY[1:0]		0xxx,0000
DMA_UR2T_CFG	UR2T_DMA configuration register	FA40H	UR2TIE	-	-	-	UR2TIP[1:0]		UR2TPTY[1:0]		0xxx,0000
DMA_UR2R_CFG	UR2R_DMA configuration register	FA48H	UR2RIE	-	-	-	UR2RIP[1:0]		UR2RPTY[1:0]		0xxx,0000
DMA_UR3T_CFG	UR3T_DMA configuration register	FA50H	UR3TIE	-	-	-	UR3TIP[1:0]		UR3TPTY[1:0]		0xxx,0000
DMA_UR3R_CFG	UR3R_DMA configuration register	FA58H	UR3RIE	-	-	-	UR3RIP[1:0]		UR3RPTY[1:0]		0xxx,0000
DMA_UR4T_CFG	UR4T_DMA configuration register	FA60H	UR4TIE	-	-	-	UR4TIP[1:0]		UR4TPTY[1:0]		0xxx,0000
DMA_UR4R_CFG	UR4R_DMA configuration register	FA68H	UR4RIE	-	-	-	UR4RIP[1:0]		UR4RPTY[1:0]		0xxx,0000
DMA_LCM_CFG	LCM_DMA configuration register	FA70H	LCMIE	-	-	-	LCMIP[1:0]		LCMPTY[1:0]		0xxx,0000
DMA_M2M_STA	M2M_DMA status register	FA02H	-	-	-	-	-	-	-	M2MIF	xxxx,xxx0
DMA_ADC_STA	ADC_DMA status register	FA12H	-	-	-	-	-	-	-	ADCIF	xxxx,xxx0
DMA_SPL_STA	SPI_DMA status register	FA22H	-	-	-	-	-	TXOVW	RXLOSS	SPIIF	xxxx,x000
DMA_UR1T_STA	UR1T_DMA status register	FA32H	-	-	-	-	-	TXOVW	-	UR1TIF	xxxx,x0x0
DMA_UR1R_STA	UR1R_DMA status register	FA3AH	-	-	-	-	-	-	RXLOSS	UR1RIF	xxxx,xx00
DMA_UR2T_STA	UR2T_DMA status register	FA42H	-	-	-	-	-	TXOVW	-	UR2TIF	xxxx,x0x0
DMA_UR2R_STA	UR2R_DMA status register	FA4AH	-	-	-	-	-	-	RXLOSS	UR2RIF	xxxx,xx00
DMA_UR3T_STA	UR3T_DMA status register	FA52H	-	-	-	-	-	TXOVW	-	UR3TIF	xxxx,x0x0
DMA_UR3R_STA	UR3R_DMA status register	FA5AH	-	-	-	-	-	-	RXLOSS	UR3RIF	xxxx,xx00
DMA_UR4T_STA	UR4T_DMA status register	FA62H	-	-	-	-	-	TXOVW	-	UR4TIF	xxxx,x0x0
DMA_UR4R_STA	UR4R_DMA status register	FA6AH	-	-	-	-	-	-	RXLOSS	UR4RIF	xxxx,xx00

DMA_LCM_STA	LCM_DMA status register	FA72H	-	-	-	-	-	-	TXOVW	LCMIF	xxxx,xx00

11.4.1 Interrupt Enable Registers (Interrupt Enable bits)

IE (Interrupt Enable Register)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
IE	A8H	EA	ELVD	EADC	ES	ET1	EX1	ET0	EX0

EA: The general or global interrupt enable control bit. The function of EA is to allow interrupts to be multi-level controlled.

That is, every interrupt source is controlled by EA firstly and then by its own interrupt enable control bit.

0: All interrupts are masked.

1: Enable the CPU interrupt, every interrupt source would be individually enabled or disabled by setting or clearing its enable bit.

ELVD: Low 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: 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 Register 2)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
IE2	AFH	EUSB	ET4	ET3	ES4	ES3	ET2	ESPI	ES2

EUSB: USB interrupt enable bit.

0: disable USB interrupt.

1: enable USB interrupt.

ET4: Timer 4 interrupt enable bit.

0: disable Timer 4 interrupt.

1: enable Timer 4 interrupt.

ET3: Timer 3 interrupt enable bit.

0: disable Timer 3 interrupt.

1: enable Timer 3 interrupt.

ES4: UART4 interrupt enable bit.

0: disable UART4 interrupt.

1: enable UART4 interrupt.

ES3: UART3 interrupt enable bit.

0: disable UART3 interrupt.

1: enable UART3 interrupt.

ET2: Timer 2 interrupt enable bit.

0: disable Timer 2 interrupt.

1: enable Timer 2 interrupt.

ESPI: SPI interrupt enable bit.

0: disable SPI interrupt.

1: enable SPI interrupt.

ES2: UART2 interrupt enable bit.

0: disable UART2 interrupt.

1: enable UART2 interrupt.

INTCLKO (External interrupt and clock output control register)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
INTCLKO	8FH	-	EX4	EX3	EX2	-	T2CLKO	T1CLKO	T0CLKO

EX4: External Interrupt 4 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.

CMPCR1 (Comparator Control Register 1)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
CMPCR1	E6H	CM PEN	CMPIF	PIE	NIE	PIS	NIS	CMPOE	CMPRES

PIE: Comparator rising-edge interrupt enable bit.

0: disable comparator rising-edge interrupt.

1: enable comparator rising-edge interrupt.

NIE: Comparator falling-edge interrupt enable bit.

0: disable comparator falling-edge interrupt.

1: enable comparator falling-edge interrupt.

I2C Control Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
I2CMSCR	FE81H	EMSI	-	-	-	MSCMD[3:0]			
I2CSLCR	FE83H	-	ESTAI	ERXI	ETXI	ESTOI	-	-	SLRST

EMSI: 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.

ESTOI: I²C slave receives STOP event interrupt enable bit.

0: disable I²C slave receives STOP event interrupt.

1: enable I²C slave receives STOP event interrupt.

PWMA interrupt enable register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_IER	FEC4H	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE

BIE: PWMA brake interrupt enable bit.

0: Disable PWMA brake interrupt

1: Enable PWMA brake interrupt

TIE: PWMA trigger interrupt enable bit.

0: Disable PWMA trigger interrupt

1: Allow PWMA to trigger interrupt

COMIE: PWMA compare interrupt enable bit.

0: Disable PWMA compare interrupt

1: Enable PWMA compare interrupt
 CC4IE: PWMA capture compare channel 4 interrupt enable bit.
 0: Disable PWMA capture compare channel 4 interrupt
 1: Allow PWMA to capture and compare channel 4 interrupt
 CC3IE: PWMA capture compare channel 3 interrupt enable bit.
 0: Disable PWMA capture compare channel 3 interrupt
 1: Allow PWMA to capture and compare channel 3 interrupt
 CC2IE: PWMA capture compare channel 2 interrupt enable bit.
 0: Disable PWMA capture compare channel 2 interrupt
 1: Allow PWMA to capture and compare channel 2 interrupt
 CC1IE: PWMA capture compare channel 1 interrupt enable bit.
 0: Disable PWMA capture compare channel 1 interrupt
 1: Allow PWMA to capture and compare channel 1 interrupt
 UIE: PWMA update interrupt enable bit.
 0: Disable PWMA update interrupt
 1: Enable PWMA update interrupt

PWMB interrupt enable register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
PWMB_IER	FEE4H	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE

BIE: PWMB brake interrupt enable bit.
 0: Disable PWMB brake interrupt
 1: Enable PWMB brake interrupt
 TIE: PWMB trigger interrupt enable bit.
 0: Disable PWMB trigger interrupt
 1: Allow PWMB to trigger interrupt
 COMIE: PWMB compare interrupt enable bit.
 0: Disable PWMB compare interrupt
 1: Enable PWMB compare interrupt
 CC4IE: PWMB capture compare channel 4 interrupt enable bit.
 0: Disable PWMB capture compare channel 4 interrupt
 1: Allow PWMB to capture and compare channel 4 interrupt
 CC3IE: PWMB capture compare channel 3 interrupt enable bit.
 0: Disable PWMB capture compare channel 3 interrupt
 1: Allow PWMB to capture and compare channel 3 interrupt
 CC2IE: PWMB capture compare channel 2 interrupt enable bit.
 0: Disable PWMB capture compare channel 2 interrupt
 1: Allow PWMB to capture and compare channel 2 interrupt
 CC1IE: PWMB capture compare channel 1 interrupt enable bit.
 0: Disable PWMB capture compare channel 1 interrupt
 1: Allow PWMB to capture and compare channel 1 interrupt
 UIE: PWMB update interrupt enable bit.
 0: Disable PWMB update interrupt
 1: Enable PWMB update interrupt

Port interrupt enable registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P0INTE	FD00H	P07INTE	P06INTE	P05INTE	P04INTE	P03INTE	P02INTE	P01INTE	P00INTE
P1INTE	FD01H	P17INTE	P16INTE	P15INTE	P14INTE	P13INTE	P12INTE	P11INTE	P10INTE
P2INTE	FD02H	P27INTE	P26INTE	P25INTE	P24INTE	P23INTE	P22INTE	P21INTE	P20INTE
P3INTE	FD03H	P37INTE	P36INTE	P35INTE	P34INTE	P33INTE	P32INTE	P31INTE	P30INTE
P4INTE	FD04H	P47INTE	P46INTE	P45INTE	P44INTE	P43INTE	P42INTE	P41INTE	P40INTE
P5INTE	FD05H	-	-	P55INTE	P54INTE	P53INTE	P52INTE	P51INTE	P50INTE
P6INTE	FD06H	P67INTE	P66INTE	P65INTE	P64INTE	P63INTE	P62INTE	P61INTE	P60INTE
P7INTE	FD07H	P77INTE	P76INTE	P75INTE	P74INTE	P73INTE	P72INTE	P71INTE	P70INTE

PnINTE.x: Port interrupt enable control bit (n=0~7, x=0~7)

0: disable Pn.x port interrupt function
 1: Enable Pn.x port interrupt function

LCM interface configuration register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
LCMIFCFG	FE50H	LCMIFIE	-	LCMIFIP[1:0]		LCMIFDPS[1:0]		D16_D8	M68_I80

LCMIFIE: LCM interface interrupt enable bit.

0: disable LCM interface interrupt

1: Enable LCM interface interrupt

DMA interrupt enable registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_M2M_CFG	FA00H	M2MIE	-	TXACO	RXACO	M2MIP[1:0]		M2MPTY[1:0]	
DMA_ADC_CFG	FA10H	ADCIE	-	-	-	ADCMIP[1:0]		ADCPTY[1:0]	
DMA_SPI_CFG	FA20H	SPIIE	ACT_TX	ACT_RX	-	SPIIP[1:0]		SPIPTY[1:0]	
DMA_UR1T_CFG	FA30H	UR1TIE	-	-	-	UR1TIP[1:0]		UR1TPTY[1:0]	
DMA_UR1R_CFG	FA38H	UR1RIE	-	-	-	UR1RIP[1:0]		UR1RPTY[1:0]	
DMA_UR2T_CFG	FA40H	UR2TIE	-	-	-	UR2TIP[1:0]		UR2TPTY[1:0]	
DMA_UR2R_CFG	FA48H	UR2RIE	-	-	-	UR2RIP[1:0]		UR2RPTY[1:0]	
DMA_UR3T_CFG	FA50H	UR3TIE	-	-	-	UR3TIP[1:0]		UR3TPTY[1:0]	
DMA_UR3R_CFG	FA58H	UR3RIE	-	-	-	UR3RIP[1:0]		UR3RPTY[1:0]	
DMA_UR3R_CFG	FA60H	UR4TIE	-	-	-	UR4TIP[1:0]		UR4TPTY[1:0]	
DMA_UR4R_CFG	FA68H	UR4RIE	-	-	-	UR4RIP[1:0]		UR4RPTY[1:0]	
DMA_LCM_CFG	FA70H	LCMIE	-	-	-	LCMIP[1:0]		LCMPTY[1:0]	

M2MIE: DMA_M2M (Memory-to-Memory DMA) interrupt enable bit.

0: disable DMA_M2M interrupt

1: enable DMA_M2M interrupt

ADCIE: DMA_ADC (ADC DMA) interrupt enable bit.

0: disable DMA_ADC interrupt

1: enable DMA_ADC interrupt

SPIIE: DMA_SPI (SPI DMA) interrupt enable bit.

0: disable DMA_SPI interrupt

1: enable DMA_SPI interrupt

UR1TIE: DMA_UR1T (UART 1 send DMA) interrupt enable bit.

0: disable DMA_UR1T interrupt

1: enable DMA_UR1T interrupt

UR1RIE: DMA_UR1R (UART 1 receive DMA) interrupt enable bit.

0: disable DMA_UR1R interrupt

1: enable DMA_UR1R interrupt

UR2TIE: DMA_UR2T (UART 2 send DMA) interrupt enable bit.

0: disable DMA_UR2T interrupt

1: enable DMA_UR2T interrupt

UR2RIE: DMA_UR2R (UART 2 receive DMA) interrupt enable bit.

0: disable DMA_UR2R interrupt

1: enable DMA_UR2R interrupt

UR3TIE: DMA_UR3T (UART 3 send DMA) interrupt enable bit.

0: disable DMA_UR3T interrupt

1: enable DMA_UR3T interrupt

UR3RIE: DMA_UR3R (UART 3 receive DMA) interrupt enable bit.

0: disable DMA_UR3R interrupt

1: enable DMA_UR3R interrupt

UR4TIE: DMA_UR4T (UART 4 send DMA) interrupt enable bit.

0: disable DMA_UR4T interrupt

1: enable DMA_UR4T interrupt

UR4RIE: DMA_UR4R (UART 4 receive DMA) interrupt enable bit.

0: disable DMA_UR4R interrupt

1: enable DMA_UR4R interrupt

LCMIE: DMA_LCM (LCM interface DMA) interrupt enable bit.

0: disable DMA_LCM interrupt

1: enable DMA_LCM interrupt

11.4.2 Interrupt Request Registers (Interrupt flags)

Timer 0 and 1 Control Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
TCON	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1: Timer/Counter 1 Overflow Flag. Set by hardware on Timer/Counter 1 overflow. It will be automatically cleared by the hardware when processor enters the Timer 1 interrupt service routine.

TF0: Timer/Counter 0 Overflow Flag. Set by hardware on Timer/Counter 0 overflow. It will be automatically cleared by the hardware when processor enters the Timer 1 interrupt service routine.

IE1: External Interrupt 1 request flag. It will be automatically cleared when the processor enters the external interrupt 1 service routine.

IE0: External Interrupt 0 request flag. It will be automatically cleared when the processor enters the external interrupt 0 service routine.

Auxiliary Interrupt Flag Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
AUXINTIF	EFH	-	INT4IF	INT3IF	INT2IF	-	T4IF	T3IF	T2IF

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

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

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

T4IF: timer 4 overflow interrupt flag, which must be cleared by software (Note: This bit is a write-only register, not readable).

T3IF: timer 3 overflow interrupt flag, which must be cleared by software (Note: This bit is a write-only register, not readable).

T2IF: timer 2 overflow interrupt flag, which must be cleared by software (Note: This bit is a write-only register, not readable).

Notice:

Early 1T 8051 MCU using 0.35um process, STC15 series added a 16-bit reload timer which was the world's first big innovation of 8051. Due to high manufacturing cost, STC 16-bit reloadable timer 2/3/4 did not design the interrupt request flag registers for users to access. The interrupt request flag register has only internal hidden flags. The method provided to the user software to clear the internal hidden flags is: when the user software disables the timer 2/3/4 interrupt, the hardware automatically clears the internal timer 2/3/4. Hidden interrupt request flags.

For product consistency:

The STC8A/ STC8F and subsequent STC8G/STC8H/ STC8C/ STC12H series which adopt 0.18um process add an interrupt request flag register accessible by the timer 2/3/4 user, but when the timer 2/3/4 interrupt is disabled, the function of the internal hidden interrupt request flag bit of the hardware automatic clear timer 2/3/4 is still retained. Therefore, do not arbitrarily disable the timer 2/3/4 interrupt when the timer 2/3/4 does not stop counting, otherwise the hidden interrupt request flag that actually works will be cleared. It is possible that after the counter overflows again, there is also a case that after the hidden interrupt request flag is set to 1, and request an interrupt and wait, it is mistakenly cleared by the user.

This is different from the traditional INTEL8048, 8051, but INTEL has been discontinued, so the new design of STC does not consider the specifications compatible with traditional INTEL.

This is the further development of 8051 by China STC.

UARTs Control Registers

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

TI: Transmit interrupt flag of UART1, which must be cleared by software.

RI: Receive interrupt flag of UART1, which must be cleared by software.

S2TI: Transmit interrupt flag of UART2, which must be cleared by software.

S2RI: Receive interrupt flag of UART2, which must be cleared by software.

S3TI: Transmit interrupt flag of UART3, which must be cleared by software.

S3RI: Receive interrupt flag of UART3, which must be cleared by software.

S4TI: Transmit interrupt flag of UART4, which must be cleared by software.

S4RI: Receive interrupt flag of UART4, which must be cleared by software.

Power Control Register

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

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

ADC Control Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
ADC_CONTR	BCH	ADC_POWER	ADC_START	ADC_FLAG	ADC_EPWMT	ADC_CHS[3:0]			

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

SPI Status Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
SPSTAT	CDH	SPIF	WCOL	-	-	-	-	-	-

SPIF: SPI transmission completion interrupt request flag, which must be cleared by software.

Comparator Control Register 1

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
CMPCR1	E6H	CMPEN	CMPIF	PIE	NIE	PIS	NIS	CMPOE	CMPRES

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

I2C Status Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
I2CMSST	FE82H	MSBUSY	MSIF	-	-	-	-	MSACKI	MSACKO
I2CSLST	FE84H	SLBUSY	STAIF	RXIF	TXIF	STOIF	TXING	SLACKI	SLACKO

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

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

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

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

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

PWMA Status Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
PWMA_SR1	FEC5H	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
PWMA_SR2	FEC6H	-	-	-	CC4OF	CC3OF	CC2OF	CC1OF	-

BIF: PWMA brake interrupt request flag. Need to be cleared by software.

TIF: PWMA triggers the interrupt request flag. Need to be cleared by software.

COMIF: PWMA compare interrupt request flag. Need to be cleared by software.

CC4IF: A capture compare interrupt request flag occurred in PWMA channel 4. Need to be cleared by software.

CC3IF: A capture compare interrupt request flag occurred in PWMA channel 3. Need to be cleared by software.

CC2IF: A capture compare interrupt request flag occurred in PWMA channel 2. Need to be cleared by software.

CC1IF: A capture compare interrupt request flag occurred in PWMA channel 1. Need to be cleared by software.

TIF: PWMA update interrupt request flag. Need to be cleared by software.

CC4OF: PWMA channel 4 has repeated capture interrupt request flag. Need to be cleared by software.

CC3OF: PWMA channel 3 has repeated capture interrupt request flag. Need to be cleared by software.

CC2OF: PWMA channel 2 has repeated capture interrupt request flag. Need to be cleared by software.

CC1OF: PWMA channel 1 repeated capture interrupt request flag. Need to be cleared by software.

PWMB Status Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
PWMB_SR1	FEC5H	BIF	TIF	COMIF	CC8IF	CC7IF	CC6IF	CC5IF	UIF
PWMB_SR2	FEC6H	-	-	-	CC8OF	CC7OF	CC6OF	CC5OF	-

BIF: PWMB brake interrupt request flag. Need to be cleared by software.

TIF: PWMB triggers the interrupt request flag. Need to be cleared by software.

COMIF: PWMB compare interrupt request flag. Need to be cleared by software.

CC8IF: A capture compare interrupt request flag occurred in PWMB channel 8. Need to be cleared by software.

CC7IF: A capture compare interrupt request flag occurred in PWMB channel 7. Need to be cleared by software.

CC6IF: A capture compare interrupt request flag occurred in PWMB channel 6. Need to be cleared by software.

CC5IF: A capture compare interrupt request flag occurred in PWMB channel 5. Need to be cleared by software.

TIF: PWMB update interrupt request flag. Need to be cleared by software.

CC8OF: PWMB channel 8 has repeated capture interrupt request flag. Need to be cleared by software.

CC7OF: PWMB channel 7 has repeated capture interrupt request flag. Need to be cleared by software.

CC6OF: PWMB channel 6 has repeated capture interrupt request flag. Need to be cleared by software.

CC5OF: PWMB channel 5 repeated capture interrupt request flag. Need to be cleared by software.

Port interrupt flag registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P0INTF	FD10H	P07INTF	P06INTF	P05INTF	P04INTF	P03INTF	P02INTF	P01INTF	P00INTF
P1INTF	FD11H	P17INTF	P16INTF	P15INTF	P14INTF	P13INTF	P12INTF	P11INTF	P10INTF
P2INTF	FD12H	P27INTF	P26INTF	P25INTF	P24INTF	P23INTF	P22INTF	P21INTF	P20INTF
P3INTF	FD13H	P37INTF	P36INTF	P35INTF	P34INTF	P33INTF	P32INTF	P31INTF	P30INTF
P4INTF	FD14H	P47INTF	P46INTF	P45INTF	P44INTF	P43INTF	P42INTF	P41INTF	P40INTF
P5INTF	FD15H	-	-	P55INTF	P54INTF	P53INTF	P52INTF	P51INTF	P50INTF
P6INTF	FD16H	P67INTF	P66INTF	P65INTF	P64INTF	P63INTF	P62INTF	P61INTF	P60INTF
P7INTF	FD17H	P77INTF	P76INTF	P75INTF	P74INTF	P73INTF	P72INTF	P71INTF	P70INTF

PnINTF.x: Port interrupt request flag (n=0~7, x=0~7)

0: No interrupt request for Pn.x port

1: Pn.x port has an interrupt request, if the interrupt is enabled, it will enter the interrupt service routine. **The flag bit needs to be cleared by software.**

LCM Interface Status Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
LCMIFSTA	FE53H	-	-	-	-	-	-	-	LCMIFIF

LCMIFIF: LCM interface interrupt request flag. Need to be cleared by software.

DMA interrupt flag registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_M2M_STA	FA02H	-	-	-	-	-	-	-	M2MIF
DMA_ADC_STA	FA12H	-	-	-	-	-	-	-	ADCIF
DMA_SPI_STA	FA22H	-	-	-	-	-	TXOVW	RXLOSS	SPIIF
DMA_UR1T_STA	FA32H	-	-	-	-	-	TXOVW	-	UR1TIF
DMA_UR1R_STA	FA3AH	-	-	-	-	-	-	RXLOSS	UR1RIF
DMA_UR2T_STA	FA42H	-	-	-	-	-	TXOVW	-	UR2TIF
DMA_UR2R_STA	FA4AH	-	-	-	-	-	-	RXLOSS	UR2RIF
DMA_UR3T_STA	FA52H	-	-	-	-	-	TXOVW	-	UR3TIF
DMA_UR3R_STA	FA5AH	-	-	-	-	-	-	RXLOSS	UR3RIF
DMA_UR4T_STA	FA62H	-	-	-	-	-	TXOVW	-	UR4TIF
DMA_UR4R_STA	FA6AH	-	-	-	-	-	-	RXLOSS	UR4RIF
DMA_LCM_STA	FA72H	-	-	-	-	-	-	TXOVW	LCMIF

M2MIF: DMA_M2M(Memory-to-Memory DMA) interrupt request flag. Need to be cleared by software.

ADCIF: DMA_ADC(ADC DMA) interrupt request flag. Need to be cleared by software.

SPIIF: DMA_SPI(SPI DMA) interrupt request flag. Need to be cleared by software.。

UR1TIF: DMA_UR1T(UART1 send DMA) interrupt request flag. Need to be cleared by software.

UR1RIF: DMA_UR1R(UART1 receive DMA) interrupt request flag. Need to be cleared by software.

UR2TIF: DMA_UR2T(UART2 send DMA) interrupt request flag. Need to be cleared by software.

UR2RIF: DMA_UR2R(UART2 receive DMA) interrupt request flag. Need to be cleared by software.

UR3TIF: DMA_UR3T(UART3 send DMA) interrupt request flag. Need to be cleared by software.

UR3RIF: DMA_UR3R(UART3 receive DMA) interrupt request flag. Need to be cleared by software.

UR4TIF: DMA_UR4T(UART4 send DMA) interrupt request flag. Need to be cleared by software.

UR4RIF: DMA_UR4R(UART4 receive DMA) interrupt request flag. Need to be cleared by software.

LCMIF: DMA_LCM(LCM interface DMA) interrupt request flag. Need to be cleared by software.

11.4.3 Interrupt Priority Control Registers

Except INT2, INT3, Timer 2, Timer 3, Timer 4 and all port interrupts, all other interrupts have 4 levels of interrupt priority that can be set.

Interrupt Priority Control Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
IP	B8H	-	PLVD	PADC	PS	PT1	PX1	PT0	PX0
IPH	B7H	-	PLVDH	PADCH	PSH	PT1H	PX1H	PT0H	PX0H
IP2	B5H	PUSB PTKSU	PI2C	PCMP	PX4	PPWMB	PPWMA	PSPI	PS2
IP2H	B6H	PUSBH PTKSUH	PI2CH	PCMPH	PX4H	PPWMBH	PPWMAH	PSPIH	PS2H
IP3	DFH	-	-	-	-	-	PRTC	PS4	PS3
IP3H	EEH	-	-	-	-	-	PRTCH	PS4H	PS3H

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

00: INT0 interrupt priority level is 0 (lowest)

01: INT0 interrupt priority level is 1 (lower)

10: INT0 interrupt priority level is 2 (higher)

11: INT0 interrupt priority level is 3 (highest)

PT0H,PT0: Timer 0 interrupt priority control bit.

00: Timer 0 interrupt priority level is 0 (lowest)

01: Timer 0 interrupt priority level is 1 (lower)

10: Timer 0 interrupt priority level is 2 (higher)

11: Timer 0 interrupt priority level is 3 (highest)

PX1H,PX1: External interrupt 1 interrupt priority control bit.

00: INT1 interrupt priority level is 0 (lowest)

01: INT1 interrupt priority level is 1 (lower)

10: INT1 interrupt priority level is 2 (higher)

11: INT1 interrupt priority level is 3 (highest)

PT1H,PT1: Timer 1 interrupt priority control bit.

00: Timer 1 interrupt priority level is 0 (lowest)

01: Timer 1 interrupt priority level is 1 (lower)

10: Timer 1 interrupt priority level is 2 (higher)

11: Timer 1 interrupt priority level is 3 (highest)

PSH,PS: UART1 interrupt priority control bit.

00: UART1 interrupt priority level is 0 (lowest)

01: UART1 interrupt priority level is 1 (lower)

10: UART1 interrupt priority level is 2 (higher)

11: UART1 interrupt priority level is 3 (highest)

PADCH,PADC: ADC interrupt priority control bit.

00: ADC interrupt priority level is 0 (lowest)

01: ADC interrupt priority level is 1 (lower)

10: ADC interrupt priority level is 2 (higher)

11: ADC interrupt priority level is 3 (highest)

PLVDH,PLVD: Low voltage detection interrupt priority control bit.

00: LVD interrupt priority level is 0 (lowest)
01: LVD interrupt priority level is 1 (lower)
10: LVD interrupt priority level is 2 (higher)
11: LVD interrupt priority level is 3 (highest)
PS2H,PS2: UART2 interrupt priority control bit.
00: UART2 interrupt priority level is 0 (lowest)
01: UART2 interrupt priority level is 1 (lower)
10: UART2 interrupt priority level is 2 (higher)
11: UART2 interrupt priority level is 3 (highest)
PS3H,PS3: UART3 interrupt priority control bit.
00: UART3 interrupt priority level is 0 (lowest)
01: UART3 interrupt priority level is 1 (lower)
10: UART3 interrupt priority level is 2 (higher)
11: UART3 interrupt priority level is 3 (highest)
PS4H,PS4: UART4 interrupt priority control bit.
00: UART4 interrupt priority level is 0 (lowest)
01: UART4 interrupt priority level is 1 (lower)
10: UART4 interrupt priority level is 2 (higher)
11: UART4 interrupt priority level is 3 (highest)
PSPIH,PSPI: SPI interrupt priority control bit.
00: SPI interrupt priority level is 0 (lowest)
01: SPI interrupt priority level is 1 (lower)
10: SPI interrupt priority level is 2 (higher)
11: SPI interrupt priority level is 3 (highest)
PPWMAH,PPWMA: Advanced PWMA interrupt priority control bit.
00: Advanced PWMA interrupt priority level is 0 (lowest)
01: Advanced PWMA interrupt priority level is 1 (lower)
10: Advanced PWMA interrupt priority level is 2 (higher)
11: Advanced PWMA interrupt priority level is 3 (highest)
PPWMBH,PPWMB: Advanced PWMB interrupt priority control bit.
00: Advanced PWMB interrupt priority level is 0 (lowest)
01: Advanced PWMB interrupt priority level is 1 (lower)
10: Advanced PWMB interrupt priority level is 2 (higher)
11: Advanced PWMB interrupt priority level is 3 (highest)
PX4H,PX4: External interrupt 4 interrupt priority control bit.
00: INT4 interrupt priority level is 0 (lowest)
01: INT4 interrupt priority level is 1 (lower)
10: INT4 interrupt priority level is 2 (higher)
11: INT4 interrupt priority level is 3 (highest)
PCMPH,PCMP: Comparator interrupt priority control bit.
00: CMP interrupt priority level is 0 (lowest)
01: CMP interrupt priority level is 1 (lower)
10: CMP interrupt priority level is 2 (higher)
11: CMP interrupt priority level is 3 (highest)
PI2CH,PI2C: I2C interrupt priority control bit.
00: I2C interrupt priority level is 0 (lowest)
01: I2C interrupt priority level is 1 (lower)
10: I2C interrupt priority level is 2 (higher)
11: I2C interrupt priority level is 3 (highest)
PUSBH, PUSB: USB interrupt priority control bit
 00: USB interrupt priority is level 0 (the lowest level)
 01: USB interrupt priority is level 1 (lower level)
 10: USB interrupt priority is level 2 (higher level)
 11: USB interrupt priority is level 3 (the highest level)
PTKSUH, PTKSU: touch key interrupt priority control bit
 00: Touch key interrupt priority is 0 (the lowest level)

- 01: Touch key interrupt priority is level 1 (lower level)
 10: Touch button interrupt priority is level 2 (higher level)
 11: Touch key interrupt priority is level 3 (the highest level)

PRTCH, PRTC: RTC interrupt priority control bit

- 00: RTC interrupt priority level is 0 (lowest level)
 01: RTC interrupt priority is level 1 (lower level)
 10: RTC interrupt priority is level 2 (higher level)
 11: RTC interrupt priority level is 3 (the highest level)

LCM Interface Configuration Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
LCMIFCFG	FE50H	LCMIFIE	-	LCMIFIP[1:0]		LCMIFDPS[1:0]		D16_D8	M68_I80

LCMIFIP[1:0]: LCM interface interrupt priority control bits

- 00: LCM interface interrupt priority level is 0 (lowest level)
 01: LCM interface interrupt priority is level 1 (lower level)
 10: LCM interface interrupt priority is level 2 (higher level)
 11: LCM interface interrupt priority level is 3 (the highest level)

Port Interrupt Priority Control Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
PINIPL	FD60H	P7IP	P6IP	P5IP	P4IP	P3IP	P2IP	P1IP	P0IP
PINIPH	FD61H	P7IPH	P6IPH	P5IPH	P4IPH	P3IPH	P2IPH	P1IPH	P0IPH

P0IPH,P0IP: P0 interrupt priority control bits

- 00: P0 interrupt priority level is 0 (lowest level)
 01: P0 interrupt priority is level 1 (lower level)
 10: P0 interrupt priority is level 2 (higher level)
 11: P0 interrupt priority level is 3 (the highest level)

P1IPH,P1IP: P1 interrupt priority control bits

- 00: P1 interrupt priority level is 0 (lowest level)
 01: P1 interrupt priority is level 1 (lower level)
 10: P1 interrupt priority is level 2 (higher level)
 11: P1 interrupt priority level is 3 (the highest level)

P2IPH,P2IP: P2 interrupt priority control bits

- 00: P2 interrupt priority level is 0 (lowest level)
 01: P2 interrupt priority is level 1 (lower level)
 10: P2 interrupt priority is level 2 (higher level)
 11: P2 interrupt priority level is 3 (the highest level)

P3IPH,P3IP: P3 interrupt priority control bits

- 00: P3 interrupt priority level is 0 (lowest level)
 01: P3 interrupt priority is level 1 (lower level)
 10: P3 interrupt priority is level 2 (higher level)
 11: P3 interrupt priority level is 3 (the highest level)

P4IPH,P4IP: P4 interrupt priority control bits

- 00: P4 interrupt priority level is 0 (lowest level)
 01: P4 interrupt priority is level 1 (lower level)
 10: P4 interrupt priority is level 2 (higher level)
 11: P4 interrupt priority level is 3 (the highest level)

P5IPH,P5IP: P5 interrupt priority control bits

- 00: P5 interrupt priority level is 0 (lowest level)
 01: P5 interrupt priority is level 1 (lower level)
 10: P5 interrupt priority is level 2 (higher level)
 11: P5 interrupt priority level is 3 (the highest level)

P6IPH,P6IP: P6 interrupt priority control bits

- 00: P6 interrupt priority level is 0 (lowest level)
 01: P6 interrupt priority is level 1 (lower level)
 10: P6 interrupt priority is level 2 (higher level)
 11: P6 interrupt priority level is 3 (the highest level)

P7IPH,P7IP: P7 interrupt priority control bits

- 00: P7 interrupt priority level is 0 (lowest level)
- 01: P7 interrupt priority is level 1 (lower level)
- 10: P7 interrupt priority is level 2 (higher level)
- 11: P7 interrupt priority level is 3 (the highest level)

DMA Interrupt Priority Control Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
DMA_M2M_CFG	FA00H	M2MIE	-	TXACO	RXACO	M2MIP[1:0]		M2MPTY[1:0]	
DMA_ADC_CFG	FA10H	ADCIE	-	-	-	ADCIP[1:0]		ADCPTY[1:0]	
DMA_SPI_CFG	FA20H	SPIIE	ACT_TX	ACT_RX	-	SPIIP[1:0]		SPIPTY[1:0]	
DMA_UR1T_CFG	FA30H	UR1TIE	-	-	-	UR1TIP[1:0]		UR1TPTY[1:0]	
DMA_UR1R_CFG	FA38H	UR1RIE	-	-	-	UR1RIP[1:0]		UR1RPTY[1:0]	
DMA_UR2T_CFG	FA40H	UR2TIE	-	-	-	UR2TIP[1:0]		UR2TPTY[1:0]	
DMA_UR2R_CFG	FA48H	UR2RIE	-	-	-	UR2RIP[1:0]		UR2RPTY[1:0]	
DMA_UR3T_CFG	FA50H	UR3TIE	-	-	-	UR3TIP[1:0]		UR3TPTY[1:0]	
DMA_UR3R_CFG	FA58H	UR3RIE	-	-	-	UR3RIP[1:0]		UR3RPTY[1:0]	
DMA_UR3R_CFG	FA60H	UR4TIE	-	-	-	UR4TIP[1:0]		UR4TPTY[1:0]	
DMA_UR4R_CFG	FA68H	UR4RIE	-	-	-	UR4RIP[1:0]		UR4RPTY[1:0]	
DMA_LCM_CFG	FA70H	LCMIE	-	-	-	LCMIP[1:0]		LCMPTY[1:0]	

M2MIP: DMA_M2M(Memory-to Memory DMA)interrupt priority control bits

- 00: DMA_M2M interrupt priority level is 0 (lowest level)
- 01: DMA_M2M interrupt priority is level 1 (lower level)
- 10: DMA_M2M interrupt priority is level 2 (higher level)
- 11: DMA_M2M interrupt priority level is 3 (the highest level)

ADCIP: DMA_ADC(ADC DMA)interrupt priority control bits

- 00: DMA_ADC interrupt priority level is 0 (lowest level)
- 01: DMA_ADC interrupt priority is level 1 (lower level)
- 10: DMA_ADC interrupt priority is level 2 (higher level)
- 11: DMA_ADC interrupt priority level is 3 (the highest level)

SPIIP: DMA_SPI(SPI DMA)interrupt priority control bits

- 00: DMA_SPI interrupt priority level is 0 (lowest level)
- 01: DMA_SPI interrupt priority is level 1 (lower level)
- 10: DMA_SPI interrupt priority is level 2 (higher level)
- 11: DMA_SPI interrupt priority level is 3 (the highest level)

UR1TIP: DMA_UR1T(UART 1 send DMA)interrupt priority control bits

- 00: DMA_UR1T interrupt priority level is 0 (lowest level)
- 01: DMA_UR1T interrupt priority is level 1 (lower level)
- 10: DMA_UR1T interrupt priority is level 2 (higher level)
- 11: DMA_UR1T interrupt priority level is 3 (the highest level)

UR1RIP: DMA_UR1R(UART 1 receive DMA)interrupt priority control bits

- 00: DMA_UR1R interrupt priority level is 0 (lowest level)
- 01: DMA_UR1R interrupt priority is level 1 (lower level)
- 10: DMA_UR1R interrupt priority is level 2 (higher level)
- 11: DMA_UR1R interrupt priority level is 3 (the highest level)

UR2TIP: DMA_UR2T(UART 2 send DMA)interrupt priority control bits

- 00: DMA_UR2T interrupt priority level is 0 (lowest level)
- 01: DMA_UR2T interrupt priority is level 1 (lower level)
- 10: DMA_UR2T interrupt priority is level 2 (higher level)
- 11: DMA_UR2T interrupt priority level is 3 (the highest level)

UR2RIP: DMA_UR2R(UART 2 receive DMA)interrupt priority control bits

- 00: DMA_UR2R interrupt priority level is 0 (lowest level)
- 01: DMA_UR2R interrupt priority is level 1 (lower level)
- 10: DMA_UR2R interrupt priority is level 2 (higher level)
- 11: DMA_UR2R interrupt priority level is 3 (the highest level)

UR3TIP: DMA_UR3T(UART 3 send DMA)interrupt priority control bits

- 00: DMA_UR3T interrupt priority level is 0 (lowest level)
- 01: DMA_UR3T interrupt priority is level 1 (lower level)

10: DMA_UR3T interrupt priority is level 2 (higher level)
11: DMA_UR3T interrupt priority level is 3 (the highest level)
UR3RIP: DMA_UR3R(UART 3 receive DMA)interrupt priority control bits
00: DMA_UR3R interrupt priority level is 0 (lowest level)
01: DMA_UR3R interrupt priority is level 1 (lower level)
10: DMA_UR3R interrupt priority is level 2 (higher level)
11: DMA_UR3R interrupt priority level is 3 (the highest level)
UR4TIP: DMA_UR4T(UART 4 send DMA)interrupt priority control bits
00: DMA_UR3R interrupt priority level is 0 (lowest level)
01: DMA_UR3R interrupt priority is level 1 (lower level)
10: DMA_UR3R interrupt priority is level 2 (higher level)
11: DMA_UR3R interrupt priority level is 3 (the highest level)
UR4RIP: DMA_UR4R(UART 4 receive DMA)interrupt priority control bits
00: DMA_UR4R interrupt priority level is 0 (lowest level)
01: DMA_UR4R interrupt priority is level 1 (lower level)
10: DMA_UR4R interrupt priority is level 2 (higher level)
11: DMA_UR4R interrupt priority level is 3 (the highest level)
LCMIP: DMA_LCM(LCM interfaceDMA)interrupt priority control bits
00: DMA_LCM interrupt priority level is 0 (lowest level)
01: DMA_LCM interrupt priority is level 1 (lower level)
10: DMA_LCM interrupt priority is level 2 (higher level)
11: DMA_LCM interrupt priority level is 3 (the highest level)

STC MCU

11.5 Example Routines

11.5.1 INT0 Interrupt (Rising and Falling Edges)

C language code

//Operating frequency for test is 11.0592MHz

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

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

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

```
void INT0_Isr() interrupt 0
```

```
{
    if (INT0)                                //Judging rising and falling edges
    {
        P10 = !P10;                          //Test port
    }
    else
    {
        P11 = !P11;                          //Test port
    }
}
```

```
void main()
```

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

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

```

while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

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

          ORG          0000H
          LJMP         MAIN
          ORG          0003H
          LJMP         INT0ISR

INT0ISR:  ORG          0100H

          JB           INT0,RISING      ;Judging rising and falling edges
          CPL          P1.0             ;Test port
          RETI

RISING:   CPL          P1.1             ;Test port
          RETI

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

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

```

E

N

D

11.5.2 INT0 Interrupt (Falling Edge)

C language code

//Operating frequency for test is 11.0592MHz

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

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

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

```
void INT0_Isr() interrupt 0
```

```
{
    P10 = !P10;                                //Test port
}
```

```
void main()
```

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

    IT0 = 1;                                    //Enable INT0 falling edge interrupt
    EX0 = 1;                                    //Enable INT0 interrupt
    EA = 1;

    while (1);
}
```

Assembly code

;Operating frequency for test is 11.0592MHz

```
P0M1      DATA    093H
P0M0      DATA    094H
P1M1      DATA    091H
P1M0      DATA    092H
P2M1      DATA    095H
P2M0      DATA    096H
```

```

P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          0003H
          LJMP         INT0ISR

INT0ISR:   ORG          0100H

          CPL          P1.0          ;Test port
          RETI

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

          SETB         IT0          ;Enable INT0 falling edge interrupt
          SETB         EX0          ;Enable INT0 interrupt
          SETB         EA
          JMP          $

          END

```

11.5.3 INT1 Interrupt (Rising and Falling Edges)

C language code

//Operating frequency for test is 11.0592MHz

```

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

```

```

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

```

```

sfr      P5M0      = 0xca;

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

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

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

    IT1 = 0;                  //Enable INT1 rising edge and falling edge interrupts
    EX1 = 1;                  //Enable INT1 interrupt
    EA = 1;

    while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

P0M1      DATA      093H
P0M0      DATA      094H
P1M1      DATA      091H
P1M0      DATA      092H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG         0000H
          LJMP        MAIN
          ORG         0013H
          LJMP        INT1ISR

```

```

INT1ISR:      ORG          0100H

               JB          INT1,RISING      ;Judging rising and falling edges
               CPL          P1.0           ;Test port
               RETI

RISING:

               CPL          P1.1           ;Test port
               RETI

MAIN:

               MOV          SP, #5FH
               MOV          P0M0, #00H
               MOV          P0M1, #00H
               MOV          P1M0, #00H
               MOV          P1M1, #00H
               MOV          P2M0, #00H
               MOV          P2M1, #00H
               MOV          P3M0, #00H
               MOV          P3M1, #00H
               MOV          P4M0, #00H
               MOV          P4M1, #00H
               MOV          P5M0, #00H
               MOV          P5M1, #00H

               CLR          IT1           ;Enable INT1 rising edge and falling edge interrupts
               SETB         EX1           ;Enable INT1 interrupt
               SETB         EA
               JMP          $

               END

```

11.5.4 INT1 Interrupt (Falling Edge)

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

sfr      P0M1      = 0x93;
sfr      P0M0      = 0x94;
sfr      P1M1      = 0x91;
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;

sbit     P10       = P1^0;

```

```

void INT1_Isr() interrupt 2
{

```

```

    P10 = !P10;           //Test port

```

```

}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    IT1 = 1;                //Enable INT1 falling edge interrupt
    EX1 = 1;                //Enable INT1 interrupt
    EA = 1;

    while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

P0M1      DATA      093H
P0M0      DATA      094H
P1M1      DATA      091H
P1M0      DATA      092H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

            ORG        0000H
            LJMP       MAIN
            ORG        0013H
            LJMP       INT1ISR

            ORG        0100H
INT1ISR:
            CPL        P1.0                ;Test port
            RETI

MAIN:
            MOV        SP, #5FH
            MOV        P0M0, #00H
            MOV        P0M1, #00H
            MOV        P1M0, #00H
            MOV        P1M1, #00H
            MOV        P2M0, #00H
            MOV        P2M1, #00H
            MOV        P3M0, #00H

```

```

MOV      P3M1, #00H
MOV      P4M0, #00H
MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

SETB     IT1                      ;Enable INT1 falling edge interrupt
SETB     EX1                      ;Enable INT1 interrupt
SETB     EA
JMP      $

END

```

11.5.5 INT2 Interrupt (Falling Edge)

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

sfr      P0M1      = 0x93;
sfr      P0M0      = 0x94;
sfr      P1M1      = 0x91;
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;

```

```

sfr      INTCLKO    = 0x8f;
#define    EX2        0x10
#define    EX3        0x20
#define    EX4        0x40
sbit     P10        = P1^0;

```

```
void INT2_Isr() interrupt 10
```

```

{
    P10 = !P10;                      //Test port
}

```

```
void main()
```

```

{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;

```

```

P5M1 = 0x00;

INTCLKO = EX2;           //Enable INT2 interrupt
EA = 1;

while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

INTCLKO    DATA    8FH
EX2        EQU      10H
EX3        EQU      20H
EX4        EQU      40H

P0M1       DATA    093H
P0M0       DATA    094H
P1M1       DATA    091H
P1M0       DATA    092H
P2M1       DATA    095H
P2M0       DATA    096H
P3M1       DATA    0B1H
P3M0       DATA    0B2H
P4M1       DATA    0B3H
P4M0       DATA    0B4H
P5M1       DATA    0C9H
P5M0       DATA    0CAH

                ORG      0000H
                LJMP     MAIN
                ORG      0053H
                LJMP     INT2ISR

INT2ISR:      ORG      0100H

                CPL      P1.0           ;Test port
                RETI

MAIN:

                MOV      SP, #5FH
                MOV      P0M0, #00H
                MOV      P0M1, #00H
                MOV      P1M0, #00H
                MOV      P1M1, #00H
                MOV      P2M0, #00H
                MOV      P2M1, #00H
                MOV      P3M0, #00H
                MOV      P3M1, #00H
                MOV      P4M0, #00H
                MOV      P4M1, #00H
                MOV      P5M0, #00H
                MOV      P5M1, #00H

                MOV      INTCLKO, #EX2   ;Enable INT2 interrupt
                SETB     EA
                JMP      $

                END

```

11.5.6 INT3 Interrupt (Falling Edge)

C language code

//Operating frequency for test is 11.0592MHz

```
#include "reg51.h"
#include "intrins.h"
```

```
sfr      P0M1      = 0x93;
sfr      P0M0      = 0x94;
sfr      P1M1      = 0x91;
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;
```

```
sfr      INTCLKO    = 0x8f;
#define   EX2        0x10
#define   EX3        0x20
#define   EX4        0x40
sbit     P10        = P1^0;
```

```
void INT3_Isr() interrupt 11
```

```
{
    P10 = !P10;           //Test port
}
```

```
void main()
```

```
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    INTCLKO = EX3;        //Enable INT3 interrupt
    EA = 1;

    while (1);
}
```

Assembly code

;Operating frequency for test is 11.0592MHz

```
INTCLKO    DATA    8FH
```

```

EX2      EQU      10H
EX3      EQU      20H
EX4      EQU      40H

P0M1     DATA    093H
P0M0     DATA    094H
P1M1     DATA    091H
P1M0     DATA    092H
P2M1     DATA    095H
P2M0     DATA    096H
P3M1     DATA    0B1H
P3M0     DATA    0B2H
P4M1     DATA    0B3H
P4M0     DATA    0B4H
P5M1     DATA    0C9H
P5M0     DATA    0CAH

ORG       0000H
LJMP     MAIN
ORG       005BH
LJMP     INT3ISR

INT3ISR:  ORG       0100H
          CPL       P1.0          ;Test port
          RETI

MAIN:
          MOV       SP, #5FH
          MOV       P0M0, #00H
          MOV       P0M1, #00H
          MOV       P1M0, #00H
          MOV       P1M1, #00H
          MOV       P2M0, #00H
          MOV       P2M1, #00H
          MOV       P3M0, #00H
          MOV       P3M1, #00H
          MOV       P4M0, #00H
          MOV       P4M1, #00H
          MOV       P5M0, #00H
          MOV       P5M1, #00H

          MOV       INTCLKO, #EX3 ;Enable INT3 interrupt
          SETB      EA
          JMP       $

          END

```

11.5.7 INT4 Interrupt (Falling Edge)

C language code

```
//Operating frequency for test is 11.0592MHz
```

```
#include "reg51.h"
#include "intrins.h"
```

```
sfr      P0M1      =    0x93;
sfr      P0M0      =    0x94;
```

```

sfr      P1M1      = 0x91;
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;

sfr      INTCLKO    = 0x8f;
#define   EX2        0x10
#define   EX3        0x20
#define   EX4        0x40
sbit     P10        = P1^0;

```

```
void INT4_Isr() interrupt 16
```

```

{
    P10 = !P10;                //Test port
}

```

```
void main()
```

```

{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    INTCLKO = EX4;             //Enable INT4 interrupt
    EA = 1;

    while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

INTCLKO    DATA    8FH
EX2         EQU     10H
EX3         EQU     20H
EX4         EQU     40H

```

```

P0M1       DATA    093H
P0M0       DATA    094H
P1M1       DATA    091H
P1M0       DATA    092H
P2M1       DATA    095H
P2M0       DATA    096H
P3M1       DATA    0B1H
P3M0       DATA    0B2H

```

```

P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          0083H
          LJMP         INT4ISR

INT4ISR:   ORG          0100H

          CPL          P1.0          ;Test port
          RETI

MAIN:
          MOV          SP, #5FH
          MOV          P0M0, #00H
          MOV          P0M1, #00H
          MOV          P1M0, #00H
          MOV          P1M1, #00H
          MOV          P2M0, #00H
          MOV          P2M1, #00H
          MOV          P3M0, #00H
          MOV          P3M1, #00H
          MOV          P4M0, #00H
          MOV          P4M1, #00H
          MOV          P5M0, #00H
          MOV          P5M1, #00H

          MOV          INTCLKO, #EX4 ;Enable INT4 interrupt
          SETB         EA
          JMP          $

          END

```

11.5.8 Timer0 Interrupt

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

sfr      P0M1      = 0x93;
sfr      P0M0      = 0x94;
sfr      P1M1      = 0x91;
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;

sbit     P10       = P1^0;

```

```

void TM0_Isr() interrupt 1
{
    P10 = !P10;                //Test port
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    TMOD = 0x00;
    TL0 = 0x66;                //65536-11.0592M/12/1000
    TH0 = 0xfc;
    TR0 = 1;                   //Start timer
    ET0 = 1;                   //Enable timer0 interrupt
    EA = 1;

    while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

P0M1      DATA      093H
P0M0      DATA      094H
P1M1      DATA      091H
P1M0      DATA      092H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

            ORG        0000H
            LJMP       MAIN
            ORG        000BH
            LJMP       TM0ISR

            ORG        0100H
TM0ISR:
            CPL        P1.0        ;Test port
            RETI

MAIN:
            MOV        SP, #5FH

```

```

MOV      P0M0, #00H
MOV      P0M1, #00H
MOV      P1M0, #00H
MOV      P1M1, #00H
MOV      P2M0, #00H
MOV      P2M1, #00H
MOV      P3M0, #00H
MOV      P3M1, #00H
MOV      P4M0, #00H
MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

MOV      TMOD, #00H
MOV      TL0, #66H           ;65536-11.0592M/12/1000
MOV      TH0, #0FCH
SETB     TR0                 ;Start timer
SETB     ET0                 ;Enable timer0 interrupt
SETB     EA

JMP      $

END

```

11.5.9 Timer1 Interrupt

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

sfr      P0M1      = 0x93;
sfr      P0M0      = 0x94;
sfr      P1M1      = 0x91;
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;

```

```

sbit     P10       = P1^0;

```

```

void TM1_Isr() interrupt 3

```

```

{
    P10 = !P10;           //Test port
}

```

```

void main()

```

```

{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
}

```



```

P2M0 = 0x00;
P2M1 = 0x00;
P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

TMOD = 0x00;
TL1 = 0x66;           //65536-11.0592M/12/1000
TH1 = 0xfc;
TR1 = 1;              //Start timer
ET1 = 1;              //Enable timer1 interrupt
EA = 1;

while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

P0M1      DATA      093H
P0M0      DATA      094H
P1M1      DATA      091H
P1M0      DATA      092H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          001BH
          LJMP         TM1ISR

          ORG          0100H
TM1ISR:
          CPL          P1.0           ;Test port
          RETI

MAIN:
          MOV          SP, #5FH
          MOV          P0M0, #00H
          MOV          P0M1, #00H
          MOV          P1M0, #00H
          MOV          P1M1, #00H
          MOV          P2M0, #00H
          MOV          P2M1, #00H
          MOV          P3M0, #00H
          MOV          P3M1, #00H
          MOV          P4M0, #00H
          MOV          P4M1, #00H
          MOV          P5M0, #00H
          MOV          P5M1, #00H

```

```

MOV      TMOD,#00H
MOV      TL1,#66H           ;65536-11.0592M/12/1000
MOV      TH1,#0FCH
SETB     TR1                ;Start timer
SETB     ET1                ;Enable timer1 interrupt
SETB     EA

JMP      $

END

```

11.5.10 Timer2 Interrupt

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

sfr      T2L      = 0xd7;
sfr      T2H      = 0xd6;
sfr      AUXR     = 0x8e;
sfr      IE2      = 0xaf;
#define    ET2      0x04
sfr      AUXINTIF  = 0xef;
#define    T2IF     0x01

```

```

sfr      P0M1     = 0x93;
sfr      P0M0     = 0x94;
sfr      P1M1     = 0x91;
sfr      P1M0     = 0x92;
sfr      P2M1     = 0x95;
sfr      P2M0     = 0x96;
sfr      P3M1     = 0xb1;
sfr      P3M0     = 0xb2;
sfr      P4M1     = 0xb3;
sfr      P4M0     = 0xb4;
sfr      P5M1     = 0xc9;
sfr      P5M0     = 0xca;

```

```

sbit     P10      = P1^0;

```

```

void TM2_Isr() interrupt 12

```

```

{
    P10 = !P10;           //Test port
}

```

```

void main()

```

```

{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
}

```

```

P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

T2L = 0x66; //65536-11.0592M/12/1000
T2H = 0xfc;
AUXR = 0x10; //Start timer
IE2 = ET2; //Enable timer2 interrupt
EA = 1;

while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

T2L      DATA      0D7H
T2H      DATA      0D6H
AUXR     DATA      8EH
IE2      DATA      0AFH
ET2      EQU        04H
AUXINTIF DATA      0EFH
T2IF     EQU        01H

P0M1     DATA      093H
P0M0     DATA      094H
P1M1     DATA      091H
P1M0     DATA      092H
P2M1     DATA      095H
P2M0     DATA      096H
P3M1     DATA      0B1H
P3M0     DATA      0B2H
P4M1     DATA      0B3H
P4M0     DATA      0B4H
P5M1     DATA      0C9H
P5M0     DATA      0CAH

        ORG         0000H
        LJMP        MAIN
        ORG         0063H
        LJMP        TM2ISR

        ORG         0100H
TM2ISR:  CPL         P1.0      ;Test port
        RETI

MAIN:    MOV         SP, #5FH
        MOV         P0M0, #00H
        MOV         P0M1, #00H
        MOV         P1M0, #00H
        MOV         P1M1, #00H
        MOV         P2M0, #00H
        MOV         P2M1, #00H
        MOV         P3M0, #00H
        MOV         P3M1, #00H
        MOV         P4M0, #00H

```

```

MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

MOV      T2L, #66H           ;65536-11.0592M/12/1000
MOV      T2H, #0FCH
MOV      AUXR, #10H         ;Start timer
MOV      IE2, #ET2          ;Enable timer2 interrupt
SETB     EA

JMP      $

END

```

11.5.11 Timer3 Interrupt

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

sfr      T3L      = 0xd5;
sfr      T3H      = 0xd4;
sfr      T4T3M    = 0xd1;
sfr      IE2      = 0xaf;
#define    ET3      0x20
sfr      AUXINTIF  = 0xef;
#define    T3IF     0x02

```

```

sfr      P0M1     = 0x93;
sfr      P0M0     = 0x94;
sfr      P1M1     = 0x91;
sfr      P1M0     = 0x92;
sfr      P2M1     = 0x95;
sfr      P2M0     = 0x96;
sfr      P3M1     = 0xb1;
sfr      P3M0     = 0xb2;
sfr      P4M1     = 0xb3;
sfr      P4M0     = 0xb4;
sfr      P5M1     = 0xc9;
sfr      P5M0     = 0xca;

```

```

sbit     P10      = P1^0;

```

```

void TM3_Isr() interrupt 19

```

```

{
    P10 = !P10;           //Test port
}

```

```

void main()

```

```

{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
}

```

```

P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

T3L = 0x66;           //65536-11.0592M/12/1000
T3H = 0xfc;
T4T3M = 0x08;         //Start timer
IE2 = ET3;            //Enable timer3 interrupt
EA = 1;

while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

T3L      DATA      0D5H
T3H      DATA      0D4H
T4T3M    DATA      0D1H
IE2       DATA      0AFH
ET3       EQU        20H
AUXINTIF  DATA      0EFH
T3IF      EQU        02H

P0M1     DATA      093H
P0M0     DATA      094H
P1M1     DATA      091H
P1M0     DATA      092H
P2M1     DATA      095H
P2M0     DATA      096H
P3M1     DATA      0B1H
P3M0     DATA      0B2H
P4M1     DATA      0B3H
P4M0     DATA      0B4H
P5M1     DATA      0C9H
P5M0     DATA      0CAH

        ORG          0000H
        LJMP         MAIN
        ORG          009BH
        LJMP         TM3ISR

        ORG          0100H
TM3ISR:  CPL          P1.0           ;Test port
        RETI

MAIN:    MOV          SP, #5FH
        MOV          P0M0, #00H
        MOV          P0M1, #00H
        MOV          P1M0, #00H
        MOV          P1M1, #00H
        MOV          P2M0, #00H
        MOV          P2M1, #00H
        MOV          P3M0, #00H

```

```

MOV      P3M1, #00H
MOV      P4M0, #00H
MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

MOV      T3L, #66H           ;65536-11.0592M/12/1000
MOV      T3H, #0FCH
MOV      T4T3M, #08H        ;Start timer
MOV      IE2, #ET3          ;Enable timer3 interrupt
SETB     EA

JMP      $

END

```

11.5.12 Timer4 Interrupt

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

sfr      T3L      = 0xd5;
sfr      T3H      = 0xd4;
sfr      T4L      = 0xd3;
sfr      T4H      = 0xd2;
sfr      T4T3M    = 0xd1;
sfr      IE2      = 0xaf;
#define   ET3      0x20
#define   ET4      0x40
sfr      AUXINTIF  = 0xef;
#define   T3IF     0x02
#define   T4IF     0x04

```

```

sfr      P0M1     = 0x93;
sfr      P0M0     = 0x94;
sfr      P1M1     = 0x91;
sfr      P1M0     = 0x92;
sfr      P2M1     = 0x95;
sfr      P2M0     = 0x96;
sfr      P3M1     = 0xb1;
sfr      P3M0     = 0xb2;
sfr      P4M1     = 0xb3;
sfr      P4M0     = 0xb4;
sfr      P5M1     = 0xc9;
sfr      P5M0     = 0xca;

```

```

sbit     P10      = P1^0;

```

```

void TM4_Isr() interrupt 20

```

```

{
    P10 = !P10;           //Test port
}

```

```

void main()

```

```

{

```

```

P0M0 = 0x00;
P0M1 = 0x00;
P1M0 = 0x00;
P1M1 = 0x00;
P2M0 = 0x00;
P2M1 = 0x00;
P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

T4L = 0x66;           //65536-11.0592M/12/1000
T4H = 0xfc;
T4T3M = 0x80;         //Start timer
IE2 = ET4;            //Enable timer4 interrupt
EA = 1;

while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

T3L      DATA      0D5H
T3H      DATA      0D4H
T4L      DATA      0D3H
T4H      DATA      0D2H
T4T3M    DATA      0D1H
IE2       DATA      0AFH
ET3      EQU        20H
ET4      EQU        40H
AUXINTIF DATA      0EFH
T3IF     EQU        02H
T4IF     EQU        04H

P0M1     DATA      093H
P0M0     DATA      094H
P1M1     DATA      091H
P1M0     DATA      092H
P2M1     DATA      095H
P2M0     DATA      096H
P3M1     DATA      0B1H
P3M0     DATA      0B2H
P4M1     DATA      0B3H
P4M0     DATA      0B4H
P5M1     DATA      0C9H
P5M0     DATA      0CAH

        ORG         0000H
        LJMP        MAIN
        ORG         00A3H
        LJMP        TM4ISR

        ORG         0100H
TM4ISR:  CPL         P1.0           ;Test port
        RETI

```

MAIN:

```

MOV     SP, #5FH
MOV     P0M0, #00H
MOV     P0M1, #00H
MOV     P1M0, #00H
MOV     P1M1, #00H
MOV     P2M0, #00H
MOV     P2M1, #00H
MOV     P3M0, #00H
MOV     P3M1, #00H
MOV     P4M0, #00H
MOV     P4M1, #00H
MOV     P5M0, #00H
MOV     P5M1, #00H

MOV     T4L, #66H           ;65536-11.0592M/12/1000
MOV     T4H, #0FCH
MOV     T4T3M, #80H        ;Start timer
MOV     IE2, #ET4          ;Enable timer4 interrupt
SETB    EA

JMP     $

END

```

11.5.13 UART1 Interrupt

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

sfr      T2L      = 0xd7;
sfr      T2H      = 0xd6;
sfr      AUXR     = 0x8e;

```

```

sfr      P0M1     = 0x93;
sfr      P0M0     = 0x94;
sfr      P1M1     = 0x91;
sfr      P1M0     = 0x92;
sfr      P2M1     = 0x95;
sfr      P2M0     = 0x96;
sfr      P3M1     = 0xb1;
sfr      P3M0     = 0xb2;
sfr      P4M1     = 0xb3;
sfr      P4M0     = 0xb4;
sfr      P5M1     = 0xc9;
sfr      P5M0     = 0xca;

```

```

sbit     P10      = P1^0;
sbit     P11      = P1^1;

```

```

void UART1_Isr() interrupt 4
{
    if (TI)
    {

```



```

    TI = 0;                                     //Clear interrupt flag
    P10 = !P10;                                 //Test port
}
if (RI)
{
    RI = 0;                                     //Clear interrupt flag
    P11 = !P11;                                 //Test port
}
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    SCON = 0x50;
    T2L = 0xe8;                                 //65536-11059200/115200/4=0FFE8H
    T2H = 0xff;
    AUXR = 0x15;                                //Start timer
    ES = 1;                                     //Enable UART1 interrupt
    EA = 1;
    SBUF = 0x5a;                                // Send test data

    while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

T2L	DATA	0D7H
T2H	DATA	0D6H
AUXR	DATA	8EH
P0M1	DATA	093H
P0M0	DATA	094H
P1M1	DATA	091H
P1M0	DATA	092H
P2M1	DATA	095H
P2M0	DATA	096H
P3M1	DATA	0B1H
P3M0	DATA	0B2H
P4M1	DATA	0B3H
P4M0	DATA	0B4H
P5M1	DATA	0C9H
P5M0	DATA	0CAH
	ORG	0000H
	LJMP	MAIN
	ORG	0023H

```

        LJMP      UART1ISR

UART1ISR:
        ORG      0100H

        JNB      TI,CHECKRI
        CLR      TI                ;Clear interrupt flag
        CPL      P1.0             ;Test port

CHECKRI:
        JNB      RI,ISREXIT
        CLR      RI                ;Clear interrupt flag
        CPL      P1.1             ;Test port

ISREXIT:
        RETI

MAIN:
        MOV      SP, #5FH
        MOV      P0M0, #00H
        MOV      P0M1, #00H
        MOV      P1M0, #00H
        MOV      P1M1, #00H
        MOV      P2M0, #00H
        MOV      P2M1, #00H
        MOV      P3M0, #00H
        MOV      P3M1, #00H
        MOV      P4M0, #00H
        MOV      P4M1, #00H
        MOV      P5M0, #00H
        MOV      P5M1, #00H

        MOV      SCON, #50H
        MOV      T2L, #0E8H        ;65536-11059200/115200/4=0FFE8H
        MOV      T2H, #0FFH
        MOV      AUXR, #15H        ;Start timer
        SETB     ES                ;Enable UART1 interrupt
        SETB     EA
        MOV      SBUF, #5AH        ; Send test data

        JMP      $

END

```

11.5.14 UART2 Interrupt

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

sfr      T2L      = 0xd7;
sfr      T2H      = 0xd6;
sfr      AUXR     = 0x8e;
sfr      S2CON    = 0x9a;
sfr      S2BUF    = 0x9b;
sfr      IE2      = 0xaf;
#define     ES2      0x01

```

```

sfr      P0M1     = 0x93;

```

```

sfr      P0M0      = 0x94;
sfr      P1M1      = 0x91;
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;

sbit     P12       = P1^2;
sbit     P13       = P1^3;

```

```
void UART2_Isr() interrupt 8
```

```

{
    if (S2CON & 0x02)
    {
        S2CON &= ~0x02;           //Clear interrupt flag
        P12 = !P12;               //Test port
    }
    if (S2CON & 0x01)
    {
        S2CON &= ~0x01;           //Clear interrupt flag
        P13 = !P13;               //Test port
    }
}

```

```
void main()
```

```

{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    S2CON = 0x10;
    T2L = 0xe8;                //65536-11059200/115200/4=0FFE8H
    T2H = 0xff;
    AUXR = 0x14;               //Start timer
    IE2 = ES2;                 //Enable UART2 interrupt
    EA = 1;
    S2BUF = 0x5a;              // Send test data

    while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```
T2L      DATA      0D7H
```

T2H	DATA	0D6H	
AUXR	DATA	8EH	
S2CON	DATA	9AH	
S2BUF	DATA	9BH	
IE2	DATA	0AFH	
ES2	EQU	01H	
P0M1	DATA	093H	
P0M0	DATA	094H	
P1M1	DATA	091H	
P1M0	DATA	092H	
P2M1	DATA	095H	
P2M0	DATA	096H	
P3M1	DATA	0B1H	
P3M0	DATA	0B2H	
P4M1	DATA	0B3H	
P4M0	DATA	0B4H	
P5M1	DATA	0C9H	
P5M0	DATA	0CAH	
	ORG	0000H	
	LJMP	MAIN	
	ORG	0043H	
	LJMP	UART2ISR	
UART2ISR:	ORG	0100H	
	PUSH	ACC	
	PUSH	PSW	
	MOV	A,S2CON	
	JNB	ACC.1,CHECKRI	
	ANL	S2CON,#NOT 02H	;Clear interrupt flag
	CPL	P1.2	;Test port
CHECKRI:			
	MOV	A,S2CON	
	JNB	ACC.0,ISREXIT	
	ANL	S2CON,#NOT 01H	;Clear interrupt flag
	CPL	P1.3	;Test port
ISREXIT:			
	POP	PSW	
	POP	ACC	
	RETI		
MAIN:			
	MOV	SP, #5FH	
	MOV	P0M0, #00H	
	MOV	P0M1, #00H	
	MOV	P1M0, #00H	
	MOV	P1M1, #00H	
	MOV	P2M0, #00H	
	MOV	P2M1, #00H	
	MOV	P3M0, #00H	
	MOV	P3M1, #00H	
	MOV	P4M0, #00H	
	MOV	P4M1, #00H	
	MOV	P5M0, #00H	
	MOV	P5M1, #00H	
	MOV	S2CON,#10H	
	MOV	T2L,#0E8H	;65536-11059200/115200/4=0FFE8H

```

MOV      T2H,#0FFH
MOV      AUXR,#14H      ;Start timer
MOV      IE2,#ES2      ;Enable UART2 interrupt
SETB     EA
MOV      S2BUF,#5AH      ; Send test data

JMP      $

END

```

11.5.15 UART3 Interrupt

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

sfr      T2L      = 0xd7;
sfr      T2H      = 0xd6;
sfr      AUXR     = 0x8e;
sfr      S3CON    = 0xac;
sfr      S3BUF    = 0xad;
sfr      IE2      = 0xaf;
#define    ES3     0x08

```

```

sfr      P0M1     = 0x93;
sfr      P0M0     = 0x94;
sfr      P1M1     = 0x91;
sfr      P1M0     = 0x92;
sfr      P2M1     = 0x95;
sfr      P2M0     = 0x96;
sfr      P3M1     = 0xb1;
sfr      P3M0     = 0xb2;
sfr      P4M1     = 0xb3;
sfr      P4M0     = 0xb4;
sfr      P5M1     = 0xc9;
sfr      P5M0     = 0xca;

```

```

sbit     P12      = P1^2;
sbit     P13      = P1^3;

```

```
void UART3_Isr() interrupt 17
```

```

{
    if (S3CON & 0x02)
    {
        S3CON &= ~0x02;      //Clear interrupt flag
        P12 = !P12;          //Test port
    }
    if (S3CON & 0x01)
    {
        S3CON &= ~0x01;      //Clear interrupt flag
        P13 = !P13;          //Test port
    }
}

```

```
void main()
```

```
{
```

```

P0M0 = 0x00;
P0M1 = 0x00;
P1M0 = 0x00;
P1M1 = 0x00;
P2M0 = 0x00;
P2M1 = 0x00;
P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

S3CON = 0x10;
T2L = 0xe8; //65536-11059200/115200/4=0FFE8H
T2H = 0xff;
AUXR = 0x14; //Start timer
IE2 = ES3; //Enable UART3 interrupt
EA = 1;
S3BUF = 0x5a; //Send test data

while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

T2L	DATA	0D7H
T2H	DATA	0D6H
AUXR	DATA	8EH
S3CON	DATA	0ACH
S3BUF	DATA	0ADH
IE2	DATA	0AFH
ES3	EQU	08H
P0M1	DATA	093H
P0M0	DATA	094H
P1M1	DATA	091H
P1M0	DATA	092H
P2M1	DATA	095H
P2M0	DATA	096H
P3M1	DATA	0B1H
P3M0	DATA	0B2H
P4M1	DATA	0B3H
P4M0	DATA	0B4H
P5M1	DATA	0C9H
P5M0	DATA	0CAH
	ORG	0000H
	LJMP	MAIN
	ORG	008BH
	LJMP	UART3ISR
	ORG	0100H
UART3ISR:		
	PUSH	ACC
	PUSH	PSW
	MOV	A,S3CON
	JNB	ACC.1,CHECKRI

```

        ANL      S3CON,#NOT 02H      ;Clear interrupt flag
        CPL      P1.2                ;Test port
CHECKRI:
        MOV      A,S3CON
        JNB      ACC.0,ISREXIT
        ANL      S3CON,#NOT 01H      ;Clear interrupt flag
        CPL      P1.3                ;Test port
ISREXIT:
        POP      PSW
        POP      ACC
        RETI

MAIN:
        MOV      SP,#5FH
        MOV      P0M0,#00H
        MOV      P0M1,#00H
        MOV      P1M0,#00H
        MOV      P1M1,#00H
        MOV      P2M0,#00H
        MOV      P2M1,#00H
        MOV      P3M0,#00H
        MOV      P3M1,#00H
        MOV      P4M0,#00H
        MOV      P4M1,#00H
        MOV      P5M0,#00H
        MOV      P5M1,#00H

        MOV      S3CON,#10H
        MOV      T2L,#0E8H          ;65536-11059200/115200/4=0FFE8H
        MOV      T2H,#0FFH
        MOV      AUXR,#14H          ;Start timer
        MOV      IE2,#ES3           ;Enable UART3 interrupt
        SETB     EA
        MOV      S3BUF,#5AH         ;Send test data

        JMP      $

        END

```

11.5.16 UART4 Interrupt

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

sfr      T2L      = 0xd7;
sfr      T2H      = 0xd6;
sfr      AUXR     = 0x8e;
sfr      S4CON    = 0x84;
sfr      S4BUF    = 0x85;
sfr      IE2      = 0xaf;
#define     ES4      0x10

```

```

sfr      P0M1     = 0x93;
sfr      P0M0     = 0x94;
sfr      P1M1     = 0x91;

```

```

sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;

```

```

sbit     P12       = P1^2;
sbit     P13       = P1^3;

```

```
void UART4_Isr() interrupt 18
```

```

{
    if (S4CON & 0x02)
    {
        S4CON &= ~0x02;           //Clear interrupt flag
        P12 = !P12;               //Test port
    }
    if (S4CON & 0x01)
    {
        S4CON &= ~0x01;           //Clear interrupt flag
        P13 = !P13;               //Test port
    }
}

```

```
void main()
```

```

{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    S4CON = 0x10;
    T2L = 0xe8;           //65536-11059200/115200/4=0FFE8H
    T2H = 0xff;
    AUXR = 0x14;          //Start timer
    IE2 = ES4;            //Enable UART4 interrupt
    EA = 1;
    S4BUF = 0x5a;         //Send test data

    while (1);
}

```

Assembly code

```
;Operating frequency for test is 11.0592MHz
```

```

T2L      DATA      0D7H
T2H      DATA      0D6H
AUXR     DATA      8EH

```


S4CON	DATA	84H	
S4BUF	DATA	85H	
IE2	DATA	0AFH	
ES4	EQU	10H	
P0M1	DATA	093H	
P0M0	DATA	094H	
P1M1	DATA	091H	
P1M0	DATA	092H	
P2M1	DATA	095H	
P2M0	DATA	096H	
P3M1	DATA	0B1H	
P3M0	DATA	0B2H	
P4M1	DATA	0B3H	
P4M0	DATA	0B4H	
P5M1	DATA	0C9H	
P5M0	DATA	0CAH	
	ORG	0000H	
	LJMP	MAIN	
	ORG	0093H	
	LJMP	UART4ISR	
UART4ISR:	ORG	0100H	
	PUSH	ACC	
	PUSH	PSW	
	MOV	A,S4CON	
	JNB	ACC.1,CHECKRI	
	ANL	S4CON,#NOT 02H	;Clear interrupt flag
	CPL	P1.2	;Test port
CHECKRI:	MOV	A,S4CON	
	JNB	ACC.0,ISREXIT	
	ANL	S4CON,#NOT 01H	;Clear interrupt flag
	CPL	P1.3	;Test port
ISREXIT:	POP	PSW	
	POP	ACC	
	RETI		
MAIN:	MOV	SP, #5FH	
	MOV	P0M0, #00H	
	MOV	P0M1, #00H	
	MOV	P1M0, #00H	
	MOV	P1M1, #00H	
	MOV	P2M0, #00H	
	MOV	P2M1, #00H	
	MOV	P3M0, #00H	
	MOV	P3M1, #00H	
	MOV	P4M0, #00H	
	MOV	P4M1, #00H	
	MOV	P5M0, #00H	
	MOV	P5M1, #00H	
	MOV	S4CON,#10H	
	MOV	T2L,#0E8H	;65536-11059200/115200/4=0FFE8H
	MOV	T2H,#0FFH	
	MOV	AUXR,#14H	;Start timer

```

MOV    IE2,#ES4           ;Enable UART4 interrupt
SETB   EA
MOV     S4BUF,#5AH        ;Send test data

JMP     $

END

```

11.5.17 ADC Interrupt

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

sfr     ADC_CONTR = 0xbc;
sfr     ADC_RES   = 0xbd;
sfr     ADC_RESL  = 0xbe;
sfr     ADCCFG    = 0xde;
sbit    EADC      = IE^5;

```

```

sfr     P0M1      = 0x93;
sfr     P0M0      = 0x94;
sfr     P1M1      = 0x91;
sfr     P1M0      = 0x92;
sfr     P2M1      = 0x95;
sfr     P2M0      = 0x96;
sfr     P3M1      = 0xb1;
sfr     P3M0      = 0xb2;
sfr     P4M1      = 0xb3;
sfr     P4M0      = 0xb4;
sfr     P5M1      = 0xc9;
sfr     P5M0      = 0xca;

```

```
void ADC_Isr() interrupt 5
```

```

{
    ADC_CONTR &= ~0x20;           //Clear interrupt flag
    P0 = ADC_RES;                 //Test port
    P2 = ADC_RESL;                //Test port
}

```

```
void main()
```

```

{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

```

```
    ADCCFG = 0x00;
```

```

ADC_CONTR = 0xc0;           //Enable and start the ADC module
EADC = 1;                   //Enable ADC interrupt
EA = 1;

while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

ADC_CONTR  DATA    0BCH
ADC_RES     DATA    0BDH
ADC_RESL    DATA    0BEH
ADCCFG      DATA    0DEH
EADC        BIT      IE.5

P0M1        DATA    093H
P0M0        DATA    094H
P1M1        DATA    091H
P1M0        DATA    092H
P2M1        DATA    095H
P2M0        DATA    096H
P3M1        DATA    0B1H
P3M0        DATA    0B2H
P4M1        DATA    0B3H
P4M0        DATA    0B4H
P5M1        DATA    0C9H
P5M0        DATA    0CAH

                ORG      0000H
                LJMP     MAIN
                ORG      002BH
                LJMP     ADCISR

ADCISR:        ORG      0100H

                ANL      ADC_CONTR,#NOT 20H    ;Clear interrupt flag
                MOV      P0,ADC_RES           ;Test port
                MOV      P2,ADC_RESL          ;Test port
                RETI

MAIN:

                MOV      SP, #5FH
                MOV      P0M0, #00H
                MOV      P0M1, #00H
                MOV      P1M0, #00H
                MOV      P1M1, #00H
                MOV      P2M0, #00H
                MOV      P2M1, #00H
                MOV      P3M0, #00H
                MOV      P3M1, #00H
                MOV      P4M0, #00H
                MOV      P4M1, #00H
                MOV      P5M0, #00H
                MOV      P5M1, #00H

                MOV      ADCCFG,#00H
                MOV      ADC_CONTR,#0C0H      ;Enable and start the ADC module
                SETB     EADC                 ;Enable ADC interrupt

```

SETB *EA*

JMP *\$*

END

11.5.18 LVD Interrupt

C language code

//Operating frequency for test is 11.0592MHz

#include "reg51.h"

#include "intrins.h"

```
sfr      RSTCFG      = 0xff;
#define    ENLVR      0x40          //RSTCFG.6
#define    LVD2V2     0x00          //LVD@2.2V
#define    LVD2V4     0x01          //LVD@2.4V
#define    LVD2V7     0x02          //LVD@2.7V
#define    LVD3V0     0x03          //LVD@3.0V
sbit      ELVD       = IE^6;
#define    LVDF       0x20          //PCON.5
```

```
sfr      P0M1        = 0x93;
sfr      P0M0        = 0x94;
sfr      P1M1        = 0x91;
sfr      P1M0        = 0x92;
sfr      P2M1        = 0x95;
sfr      P2M0        = 0x96;
sfr      P3M1        = 0xb1;
sfr      P3M0        = 0xb2;
sfr      P4M1        = 0xb3;
sfr      P4M0        = 0xb4;
sfr      P5M1        = 0xc9;
sfr      P5M0        = 0xca;
sbit     P10         = P1^0;
```

void LVD_Isr() interrupt 6

```
{
    PCON &= ~LVDF;          //Clear interrupt flag
    P10 = !P10;             //Test port
}
```

void main()

```
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;
```

```

PCON &= ~LVDF;           //Interrupt flag needs to be cleared at power-on
RSTCFG = LVD3V0;          //Set the LVD voltage to 3.0V
ELVD = 1;                 //Enable LVD interrupt
EA = 1;

while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

RSTCFG      DATA      0FFH
ENLVR       EQU        40H           ;RSTCFG.6
LVD2V2      EQU        00H           ;LVD@2.2V
LVD2V4      EQU        01H           ;LVD@2.4V
LVD2V7      EQU        02H           ;LVD@2.7V
LVD3V0      EQU        03H           ;LVD@3.0V
ELVD        BIT        IE.6
LVDF        EQU        20H           ;PCON.5

P0M1        DATA      093H
P0M0        DATA      094H
P1M1        DATA      091H
P1M0        DATA      092H
P2M1        DATA      095H
P2M0        DATA      096H
P3M1        DATA      0B1H
P3M0        DATA      0B2H
P4M1        DATA      0B3H
P4M0        DATA      0B4H
P5M1        DATA      0C9H
P5M0        DATA      0CAH

                ORG      0000H
                LJMP     MAIN
                ORG      0033H
                LJMP     LVDISR

LVDISR:       ORG      0100H

                ANL      PCON,#NOT LVDF      ;Clear interrupt flag
                CPL      P1.0                ;Test port
                RETI

MAIN:
                MOV      SP, #5FH
                MOV      P0M0, #00H
                MOV      P0M1, #00H
                MOV      P1M0, #00H
                MOV      P1M1, #00H
                MOV      P2M0, #00H
                MOV      P2M1, #00H
                MOV      P3M0, #00H
                MOV      P3M1, #00H
                MOV      P4M0, #00H
                MOV      P4M1, #00H
                MOV      P5M0, #00H
                MOV      P5M1, #00H

```

ANL	PCON,#NOT LVDF	<i>;Interrupt flag needs to be cleared at power-on</i>
MOV	RSTCFG,# LVD3V0	<i>;Set the LVD voltage to 3.0V</i>
SETB	ELVD	<i>;Enable LVD interrupt</i>
SETB	EA	
JMP	\$	
 END		

11.5.19 Comparator Interrupt

C language code

//Operating frequency for test is 11.0592MHz

```
#include "reg51.h"
#include "intrins.h"

sfr      CMPCR1 = 0xe6;
sfr      CMPCR2 = 0xe7;

sfr      P0M1 = 0x93;
sfr      P0M0 = 0x94;
sfr      P1M1 = 0x91;
sfr      P1M0 = 0x92;
sfr      P2M1 = 0x95;
sfr      P2M0 = 0x96;
sfr      P3M1 = 0xb1;
sfr      P3M0 = 0xb2;
sfr      P4M1 = 0xb3;
sfr      P4M0 = 0xb4;
sfr      P5M1 = 0xc9;
sfr      P5M0 = 0xca;

sbit      P10 = P1^0;

void CMP_Isr() interrupt 21
{
    CMPCR1 &= ~0x40;    //clear interrupt flag
    P10 = !P10;         //test port
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    CMPCR2 = 0x00;
    CMPCR1 = 0x80;      // Enable comparator module
    CMPCR1 |= 0x30;     // Enable comparator edge interrupt
    CMPCR1 &= ~0x08;    //P3.6 is CMP+ input pin
    CMPCR1 |= 0x04;     //P3.7 is CMP- input pin
    CMPCR1 |= 0x02;     // Enable comparator output
    EA = 1;

    while (1);
}
```

Assembly code

;Operating frequency for test is 11.0592MHz

```
CMPCR1    DATA 0E6H
CMPCR2    DATA 0E7H
```

```
P0M1      DATA 093H
P0M0      DATA 094H
P1M1      DATA 091H
P1M0      DATA 092H
P2M1      DATA 095H
P2M0      DATA 096H
P3M1      DATA 0B1H
P3M0      DATA 0B2H
P4M1      DATA 0B3H
P4M0      DATA 0B4H
P5M1      DATA 0C9H
P5M0      DATA 0CAH
```

```
ORG       0000H
LJMP      MAIN
ORG       00ABH
LJMP      CMPISR
```

```
ORG       0100H
CMPISR:
ANL       CMPCR1,#NOT 40H    ; clear interrupt flag
CPL       P1.0              ; test port
RETI
```

MAIN:

```
MOV       SP, #5FH
MOV       P0M0, #00H
MOV       P0M1, #00H
MOV       P1M0, #00H
MOV       P1M1, #00H
MOV       P2M0, #00H
MOV       P2M1, #00H
MOV       P3M0, #00H
MOV       P3M1, #00H
MOV       P4M0, #00H
MOV       P4M1, #00H
MOV       P5M0, #00H
MOV       P5M1, #00H

MOV       CMPCR2,#00H
MOV       CMPCR1,#80H      ; Enable comparator module
ORL       CMPCR1,#30H      ; Enable comparator edge interrupt
ANL       CMPCR1,#NOT 08H   ; P3.6 is CMP+ input pin
ORL       CMPCR1,#04H      ; P3.7 is CMP- input pin
ORL       CMPCR1,#02H      ; Enable comparator output
SETB     EA

JMP       $
```

END

11.5.20 SPI Interrupt

C language code

//Operating frequency for test is 11.0592MHz

```
#include "reg51.h"
```

```
#include "intrins.h"
```

```
sfr      SPSTAT      = 0xcd;
sfr      SPCTL       = 0xce;
sfr      SPDAT       = 0xcf;
sfr      IE2         = 0xaf;
#define   ESPI        0x02
```

```
sfr      P0M1        = 0x93;
sfr      P0M0        = 0x94;
sfr      P1M1        = 0x91;
sfr      P1M0        = 0x92;
sfr      P2M1        = 0x95;
sfr      P2M0        = 0x96;
sfr      P3M1        = 0xb1;
sfr      P3M0        = 0xb2;
sfr      P4M1        = 0xb3;
sfr      P4M0        = 0xb4;
sfr      P5M1        = 0xc9;
sfr      P5M0        = 0xca;
```

```
sbit     P10         = P1^0;
```

```
void SPI_Isr() interrupt 9
```

```
{
    SPSTAT = 0xc0;           //Clear interrupt flag
    P10 = !P10;             //Test port
}
```

```
void main()
```

```
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    SPCTL = 0x50;           //Enable SPI master mode
    SPSTAT = 0xc0;         //Clear interrupt flag
    IE2 = ESPI;             //Enable SPI interrupt
    EA = 1;
    SPDAT = 0x5a;           //Send test data

    while (1);
}
```

Assembly code

;Operating frequency for test is 11.0592MHz

```
SPSTAT      DATA      0CDH
SPCTL       DATA      0CEH
SPDAT       DATA      0CFH
```

```

IE2      DATA      0AFH
ESPI      EQU        02H

P0M1      DATA      093H
P0M0      DATA      094H
P1M1      DATA      091H
P1M0      DATA      092H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

ORG        0000H
LJMP       MAIN
ORG        004BH
LJMP       SPIISR

SPIISR:    ORG        0100H

MOV        SPSTAT,#0C0H      ;Clear interrupt flag
CPL        P1.0              ;Test port
RETI

MAIN:

MOV        SP, #5FH
MOV        P0M0, #00H
MOV        P0M1, #00H
MOV        P1M0, #00H
MOV        P1M1, #00H
MOV        P2M0, #00H
MOV        P2M1, #00H
MOV        P3M0, #00H
MOV        P3M1, #00H
MOV        P4M0, #00H
MOV        P4M1, #00H
MOV        P5M0, #00H
MOV        P5M1, #00H

MOV        SPCTL,#50H        ;Enable SPI master mode
MOV        SPSTAT,#0C0H      ;Clear interrupt flag
MOV        IE2,#ESPI         ;Enable SPI interrupt
SETB       EA
MOV        SPDAT,#5AH        ;Send test data

JMP        $

END

```

11.5.21 I2C Interrupt

C language code

```
//Operating frequency for test is 11.0592MHz
```

```
#include "reg51.h"
```

```
#include "intrins.h"
```

```
sfr      P_SW2      = 0xba;
```

```
#define I2CCFG      (*(unsigned char volatile xdata *)0xfe80)
#define I2CMSCR      (*(unsigned char volatile xdata *)0xfe81)
#define I2CMSST      (*(unsigned char volatile xdata *)0xfe82)
#define I2CSLCR      (*(unsigned char volatile xdata *)0xfe83)
#define I2CSLST      (*(unsigned char volatile xdata *)0xfe84)
#define I2CSLADR      (*(unsigned char volatile xdata *)0xfe85)
#define I2CTXD      (*(unsigned char volatile xdata *)0xfe86)
#define I2CRXD      (*(unsigned char volatile xdata *)0xfe87)
```

```
sfr      P0M1      = 0x93;
sfr      P0M0      = 0x94;
sfr      P1M1      = 0x91;
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;
sbit     P10       = P1^0;
```

```
void I2C_Isr() interrupt 24
```

```
{
    _push_(P_SW2);
    P_SW2 |= 0x80;
    if (I2CMSST & 0x40)
    {
        I2CMSST &= ~0x40;          //Clear interrupt flag
        P10 = !P10;                //Test port
    }
    _pop_(P_SW2);
}
```

```
void main()
```

```
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P_SW2 = 0x80;
    I2CCFG = 0xc0;          //Enable I2C master mode
    I2CMSCR = 0x80;         //Enable I2C interrupt;
    P_SW2 = 0x00;
    EA = 1;
}
```

```

P_SW2 = 0x80;
I2CMSCR = 0x81;                                     //Send start command
P_SW2 = 0x00;

while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

P_SW2 DATA 0BAH

I2CCFG XDATA 0FE80H

I2CMSCR XDATA 0FE81H

I2CMSST XDATA 0FE82H

I2CSLCR XDATA 0FE83H

I2CSLST XDATA 0FE84H

I2CSLADR XDATA 0FE85H

I2CTXD XDATA 0FE86H

I2CRXD XDATA 0FE87H

P0M1 DATA 093H

P0M0 DATA 094H

P1M1 DATA 091H

P1M0 DATA 092H

P2M1 DATA 095H

P2M0 DATA 096H

P3M1 DATA 0B1H

P3M0 DATA 0B2H

P4M1 DATA 0B3H

P4M0 DATA 0B4H

P5M1 DATA 0C9H

P5M0 DATA 0CAH

ORG 0000H

LJMP MAIN

ORG 00C3H

LJMP I2CISR

ORG 0100H

I2CISR:

PUSH ACC

PUSH DPL

PUSH DPH

PUSH P_SW2

MOV P_SW2,#80H

MOV DPTR,#I2CMSST

MOVX A,@DPTR

ANL A,#NOT 40H

;Clear interrupt flag

MOVX @DPTR,A

CPL P1.0

;Test port

POP P_SW2

POP DPH

POP DPL

POP ACC

RETI

MAIN:

MOV SP, #5FH

```
MOV    P0M0, #00H
MOV    P0M1, #00H
MOV    P1M0, #00H
MOV    P1M1, #00H
MOV    P2M0, #00H
MOV    P2M1, #00H
MOV    P3M0, #00H
MOV    P3M1, #00H
MOV    P4M0, #00H
MOV    P4M1, #00H
MOV    P5M0, #00H
MOV    P5M1, #00H

MOV    P_SW2, #80H
MOV    A, #0C0H           ;Enable I2C master mode
MOV    DPTR, #I2CCFG
MOVX   @DPTR, A
MOV    A, #80H           ;Enable I2C interrupt
MOV    DPTR, #I2CMSCR
MOVX   @DPTR, A
MOV    P_SW2, #00H
SETB   EA

MOV    P_SW2, #80H
MOV    A, #081H          ;Send start command
MOV    DPTR, #I2CMSCR
MOVX   @DPTR, A
MOV    P_SW2, #00H

JMP    $

END
```

12 I/O port interrupt

Product line	I/O interrupt	I/O interrupt priority	I/O interrupt wake-up function
STC8H1K08 family			
STC8H1K28 family			
STC8H3K64S4 family A version	●	1	
STC8H3K64S4 family A version	●	1	
STC8H3K64S2 family B version	●	1	●
STC8H3K64S2 family B version	●	1	●
STC8H8K64U family A version			
STC8H8K64U family B version	●	4	●
STC8H2K64T family	●	4	●
STC8H4K64TLR family	●	4	●
STC8H4K64TLCD family	●	4	●
STC8H4K64LCD family	●	4	●

The subsequent families of the STC8H series support all I/O interrupts, and support 4 interrupt modes: falling edge interrupt, rising edge interrupt, low-level interrupt, and high-level interrupt. Every group of I/O ports has an independent interrupt entry address, and each I/O can independently set the interrupt mode.

Note: The I/O interrupts of A-version chips of STC8H3K64S4 and STC8H3K64S2 series cannot wake up CPU from power-off, and only have level 1 interrupt priority. The I/O interrupts of B-version chips of STC8H3K64S4 and STC8H3K64S2 series can wake up CPU from power-down, and only have 1 level interrupt priority. For other series with I/O interrupts, I/O interrupts can wake up CPU from power-down, and there are 4 levels of interrupt priority.

12.1 I/O port interrupt related registers

Symbol	Description	Address	Bit Address and Symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
P0INTE	P0 interrupt enable register	FD00H	P07INTE	P06INTE	P05INTE	P04INTE	P03INTE	P02INTE	P01INTE	P00INTE	0000,0000
P1INTE	P1 interrupt enable register	FD01H	P17INTE	P16INTE	P15INTE	P14INTE	P13INTE	P12INTE	P11INTE	P10INTE	0000,0000
P2INTE	P2 interrupt enable register	FD02H	P27INTE	P26INTE	P25INTE	P24INTE	P23INTE	P22INTE	P21INTE	P20INTE	0000,0000
P3INTE	P3 interrupt enable register	FD03H	P37INTE	P36INTE	P35INTE	P34INTE	P33INTE	P32INTE	P31INTE	P30INTE	0000,0000
P4INTE	P4 interrupt enable register	FD04H	P47INTE	P46INTE	P45INTE	P44INTE	P43INTE	P42INTE	P41INTE	P40INTE	0000,0000
P5INTE	P5 interrupt enable register	FD05H	-	-	P55INTE	P54INTE	P53INTE	P52INTE	P51INTE	P50INTE	xx00,0000
P6INTE	P6 interrupt enable register	FD06H	P67INTE	P66INTE	P65INTE	P64INTE	P63INTE	P62INTE	P61INTE	P60INTE	0000,0000
P7INTE	P7 interrupt enable register	FD07H	P77INTE	P76INTE	P75INTE	P74INTE	P73INTE	P72INTE	P71INTE	P70INTE	0000,0000
P0INTF	P0 interrupt flag register	FD10H	P07INTF	P06INTF	P05INTF	P04INTF	P03INTF	P02INTF	P01INTF	P00INTF	0000,0000
P1INTF	P1 interrupt flag register	FD11H	P17INTF	P16INTF	P15INTF	P14INTF	P13INTF	P12INTF	P11INTF	P10INTF	0000,0000
P2INTF	P2 interrupt flag register	FD12H	P27INTF	P26INTF	P25INTF	P24INTF	P23INTF	P22INTF	P21INTF	P20INTF	0000,0000
P3INTF	P3 interrupt flag register	FD13H	P37INTF	P36INTF	P35INTF	P34INTF	P33INTF	P32INTF	P31INTF	P30INTF	0000,0000
P4INTF	P4 interrupt flag register	FD14H	P47INTF	P46INTF	P45INTF	P44INTF	P43INTF	P42INTF	P41INTF	P40INTF	0000,0000
P5INTF	P5 interrupt flag register	FD15H	-	-	P55INTF	P54INTF	P53INTF	P52INTF	P51INTF	P50INTF	xx00,0000
P6INTF	P6 interrupt flag register	FD16H	P67INTF	P66INTF	P65INTF	P64INTF	P63INTF	P62INTF	P61INTF	P60INTF	0000,0000
P7INTF	P7 interrupt flag register	FD17H	P77INTF	P76INTF	P75INTF	P74INTF	P73INTF	P72INTF	P71INTF	P70INTF	0000,0000

P0IM0	P0 Interrupt mode register 0	FD20H	P07IM0	P06IM0	P05IM0	P04IM0	P03IM0	P02IM0	P01IM0	P00IM0	0000,0000
P1IM0	P1 Interrupt mode register 0	FD21H	P17IM0	P16IM0	P15IM0	P14IM0	P13IM0	P12IM0	P11IM0	P10IM0	0000,0000
P2IM0	P2 Interrupt mode register 0	FD22H	P27IM0	P26IM0	P25IM0	P24IM0	P23IM0	P22IM0	P21IM0	P20IM0	0000,0000
P3IM0	P3 Interrupt mode register 0	FD23H	P37IM0	P36IM0	P35IM0	P34IM0	P33IM0	P32IM0	P31IM0	P30IM0	0000,0000
P4IM0	P4 Interrupt mode register 0	FD24H	P47IM0	P46IM0	P45IM0	P44IM0	P43IM0	P42IM0	P41IM0	P40IM0	0000,0000
P5IM0	P5 Interrupt mode register 0	FD25H	-	-	P55IM0	P54IM0	P53IM0	P52IM0	P51IM0	P50IM0	xx00,0000
P6IM0	P6 Interrupt mode register 0	FD26H	P67IM0	P66IM0	P65IM0	P64IM0	P63IM0	P62IM0	P61IM0	P60IM0	0000,0000
P7IM0	P7 Interrupt mode register 0	FD27H	P77IM0	P76IM0	P75IM0	P74IM0	P73IM0	P72IM0	P71IM0	P70IM0	0000,0000
P0IM1	P0 Interrupt mode register 1	FD30H	P07IM1	P06IM1	P05IM1	P04IM1	P03IM1	P02IM1	P01IM1	P00IM1	0000,0000
P1IM1	P1 Interrupt mode register 1	FD31H	P17IM1	P16IM1	P15IM1	P14IM1	P13IM1	P12IM1	P11IM1	P10IM1	0000,0000
P2IM1	P2 Interrupt mode register 1	FD32H	P27IM1	P26IM1	P25IM1	P24IM1	P23IM1	P22IM1	P21IM1	P20IM1	0000,0000
P3IM1	P3 Interrupt mode register 1	FD33H	P37IM1	P36IM1	P35IM1	P34IM1	P33IM1	P32IM1	P31IM1	P30IM1	0000,0000
P4IM1	P4 Interrupt mode register 1	FD34H	P47IM1	P46IM1	P45IM1	P44IM1	P43IM1	P42IM1	P41IM1	P40IM1	0000,0000
P5IM1	P5 Interrupt mode register 1	FD35H	-	-	P55IM1	P54IM1	P53IM1	P52IM1	P51IM1	P50IM1	xx00,0000
P6IM1	P6 Interrupt mode register 1	FD36H	P67IM1	P66IM1	P65IM1	P64IM1	P63IM1	P62IM1	P61IM1	P60IM1	0000,0000
P7IM1	P7 Interrupt mode register 1	FD37H	P77IM1	P76IM1	P75IM1	P74IM1	P73IM1	P72IM1	P71IM1	P70IM1	0000,0000
PINIPL	I/O port interrupt priority low register	FD60H	P7IP	P6IP	P5IP	P4IP	P3IP	P2IP	P1IP	P0IP	0000,0000
PINIPH	I/O port interrupt priority high register	FD61H	P7IPH	P6IPH	P5IPH	P4IPH	P3IPH	P2IPH	P1IPH	P0IPH	0000,0000
P0WKUE	P0 interrupt wake-up enable register	FD40H	P07WKUE	P06WKUE	P05WKUE	P04WKUE	P03WKUE	P02WKUE	P01WKUE	P00WKUE	0000,0000
P1WKUE	P1 interrupt wake-up enable register	FD41H	P17WKUE	P16WKUE	P15WKUE	P14WKUE	P13WKUE	P12WKUE	P11WKUE	P10WKUE	0000,0000
P2WKUE	P2 interrupt wake-up enable register	FD42H	P27WKUE	P26WKUE	P25WKUE	P24WKUE	P23WKUE	P22WKUE	P21WKUE	P20WKUE	0000,0000
P3WKUE	P3 interrupt wake-up enable register	FD43H	P37WKUE	P36WKUE	P35WKUE	P34WKUE	P33WKUE	P32WKUE	P31WKUE	P30WKUE	0000,0000
P4WKUE	P4 interrupt wake-up enable register	FD44H	P47WKUE	P46WKUE	P45WKUE	P44WKUE	P43WKUE	P42WKUE	P41WKUE	P40WKUE	0000,0000
P5WKUE	P5 interrupt wake-up enable register	FD45H	-	-	P55WKUE	P54WKUE	P53WKUE	P52WKUE	P51WKUE	P50WKUE	xx00,0000
P6WKUE	P6 interrupt wake-up enable register	FD46H	P67WKUE	P66WKUE	P65WKUE	P64WKUE	P63WKUE	P62WKUE	P61WKUE	P60WKUE	0000,0000
P7WKUE	P7 interrupt wake-up enable register	FD47H	P77WKUE	P76WKUE	P75WKUE	P74WKUE	P73WKUE	P72WKUE	P71WKUE	P70WKUE	0000,0000

12.1.1 Port interrupt enable registers (PxINTE)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P0INTE	FD00H	P07INTE	P06INTE	P05INTE	P04INTE	P03INTE	P02INTE	P01INTE	P00INTE
P1INTE	FD01H	P17INTE	P16INTE	P15INTE	P14INTE	P13INTE	P12INTE	P11INTE	P10INTE
P2INTE	FD02H	P27INTE	P26INTE	P25INTE	P24INTE	P23INTE	P22INTE	P21INTE	P20INTE
P3INTE	FD03H	P37INTE	P36INTE	P35INTE	P34INTE	P33INTE	P32INTE	P31INTE	P30INTE
P4INTE	FD04H	P47INTE	P46INTE	P45INTE	P44INTE	P43INTE	P42INTE	P41INTE	P40INTE
P5INTE	FD05H	-	-	P55INTE	P54INTE	P53INTE	P52INTE	P51INTE	P50INTE
P6INTE	FD06H	P67INTE	P66INTE	P65INTE	P64INTE	P63INTE	P62INTE	P61INTE	P60INTE
P7INTE	FD07H	P77INTE	P76INTE	P75INTE	P74INTE	P73INTE	P72INTE	P71INTE	P70INTE

PnINTE.x: Port interrupt enable control bit (n=0~7, x=0~7)

0: disable Pn.x port interrupt function

1: Enable Pn.x port interrupt function

12.1.2 Port interrupt flag registers (PxINTF)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P0INTF	FD10H	P07INTF	P06INTF	P05INTF	P04INTF	P03INTF	P02INTF	P01INTF	P00INTF
P1INTF	FD11H	P17INTF	P16INTF	P15INTF	P14INTF	P13INTF	P12INTF	P11INTF	P10INTF
P2INTF	FD12H	P27INTF	P26INTF	P25INTF	P24INTF	P23INTF	P22INTF	P21INTF	P20INTF
P3INTF	FD13H	P37INTF	P36INTF	P35INTF	P34INTF	P33INTF	P32INTF	P31INTF	P30INTF
P4INTF	FD14H	P47INTF	P46INTF	P45INTF	P44INTF	P43INTF	P42INTF	P41INTF	P40INTF
P5INTF	FD15H	-	-	P55INTF	P54INTF	P53INTF	P52INTF	P51INTF	P50INTF
P6INTF	FD16H	P67INTF	P66INTF	P65INTF	P64INTF	P63INTF	P62INTF	P61INTF	P60INTF
P7INTF	FD17H	P77INTF	P76INTF	P75INTF	P74INTF	P73INTF	P72INTF	P71INTF	P70INTF

PnINTF.x: Port interrupt request flag (n=0~7, x=0~7)

0: No interrupt request for Pn.x port

1: Pn.x port has an interrupt request, if the interrupt is enabled, it will enter the interrupt service routine. **The flag bit needs to be cleared by software.**

12.1.3 Port interrupt mode configuration registers (PxIM0, PxIM1)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P0IM0	FD20H	P07IM0	P06IM0	P05IM0	P04IM0	P03IM0	P02IM0	P01IM0	P00IM0
P0IM1	FD30H	P07IM1	P06IM1	P05IM1	P04IM1	P03IM1	P02IM1	P01IM1	P00IM1
P1IM0	FD21H	P17IM0	P16IM0	P15IM0	P14IM0	P13IM0	P12IM0	P11IM0	P10IM0
P1IM1	FD31H	P17IM1	P16IM1	P15IM1	P14IM1	P13IM1	P12IM1	P11IM1	P10IM1
P2IM0	FD22H	P27IM0	P26IM0	P25IM0	P24IM0	P23IM0	P22IM0	P21IM0	P20IM0
P2IM1	FD32H	P27IM1	P26IM1	P25IM1	P24IM1	P23IM1	P22IM1	P21IM1	P20IM1
P3IM0	FD23H	P37IM0	P36IM0	P35IM0	P34IM0	P33IM0	P32IM0	P31IM0	P30IM0

P3IM1	FD33H	P37IM1	P36IM1	P35IM1	P34IM1	P33IM1	P32IM1	P31IM1	P30IM1
P4IM0	FD24H	P47IM0	P46IM0	P45IM0	P44IM0	P43IM0	P42IM0	P41IM0	P40IM0
P4IM1	FD34H	P47IM1	P46IM1	P45IM1	P44IM1	P43IM1	P42IM1	P41IM1	P40IM1
P5IM0	FD25H	-	-	P55IM0	P54IM0	P53IM0	P52IM0	P51IM0	P50IM0
P5IM1	FD35H	-	-	P55IM1	P54IM1	P53IM1	P52IM1	P51IM1	P50IM1
P6IM0	FD26H	P67IM0	P66IM0	P65IM0	P64IM0	P63IM0	P62IM0	P61IM0	P60IM0
P6IM1	FD36H	P67IM1	P66IM1	P65IM1	P64IM1	P63IM1	P62IM1	P61IM1	P60IM1
P7IM0	FD27H	P77IM0	P76IM0	P75IM0	P74IM0	P73IM0	P72IM0	P71IM0	P70IM0
P7IM1	FD37H	P77IM1	P76IM1	P75IM1	P74IM1	P73IM1	P72IM1	P71IM1	P70IM1

Configure port mode

PnIM1.x	PnIM0.x	Pn.x Interrupt mode
0	0	Falling edge interrupt
0	1	Rising edge interrupt
1	0	Low level interrupt
1	1	High level interrupt

12.1.4 Port interrupt priority control registers (PINIPL, PINIPH)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
PINIPL	FD60H	P7IP	P6IP	P5IP	P4IP	P3IP	P2IP	P1IP	P0IP
PINIPH	FD61H	P7IPH	P6IPH	P5IPH	P4IPH	P3IPH	P2IPH	P1IPH	P0IPH

PxIPH, PxIP: Px interrupt priority control bit

- 00: Px interrupt priority level is 0 (lowest)
- 01: Px interrupt priority level is 1 (lower)
- 10: Px interrupt priority level is 2 (higher)
- 11: Px interrupt priority level is 3 (highest)

12.1.5 Port interrupt power-down wake-up enable registers (PxWKUE)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
P0WKUE	FD40H	P07WKUE	P06WKUE	P05WKUE	P04WKUE	P03WKUE	P02WKUE	P01WKUE	P00WKUE
P1WKUE	FD41H	P17WKUE	P16WKUE	P15WKUE	P14WKUE	P13WKUE	P12WKUE	P11WKUE	P10WKUE
P2WKUE	FD42H	P27WKUE	P26WKUE	P25WKUE	P24WKUE	P23WKUE	P22WKUE	P21WKUE	P20WKUE
P3WKUE	FD43H	P37WKUE	P36WKUE	P35WKUE	P34WKUE	P33WKUE	P32WKUE	P31WKUE	P30WKUE
P4WKUE	FD44H	P47WKUE	P46WKUE	P45WKUE	P44WKUE	P43WKUE	P42WKUE	P41WKUE	P40WKUE
P5WKUE	FD45H	-	-	P55WKUE	P54WKUE	P53WKUE	P52WKUE	P51WKUE	P50WKUE
P6WKUE	FD46H	P67WKUE	P66WKUE	P65WKUE	P64WKUE	P63WKUE	P62WKUE	P61WKUE	P60WKUE
P7WKUE	FD47H	P77WKUE	P76WKUE	P75WKUE	P74WKUE	P73WKUE	P72WKUE	P71WKUE	P70WKUE

PnxWKUE: Port interrupt power-down wake-up enable bit(n=0~7, x=0~7)

- 0: disable Pn.x interrupt power-down wake-up function
- 1: enable Pn.x interrupt power-down wake-up function

12.2 Example Routines

12.2.1 P0 Falling edge interrupt

C language code

// Operating frequency for test is 11.0592MHz

```
#include "reg51.h"
#include "intrins.h"

sfr      P0M0      = 0x94;
sfr      P0M1      = 0x93;
sfr      P1M0      = 0x92;
sfr      P1M1      = 0x91;
sfr      P2M0      = 0x96;
sfr      P2M1      = 0x95;
sfr      P3M0      = 0xb2;
sfr      P3M1      = 0xb1;
sfr      P4M0      = 0xb4;
sfr      P4M1      = 0xb3;
sfr      P5M0      = 0xca;
sfr      P5M1      = 0xc9;
sfr      P6M0      = 0xcc;
sfr      P6M1      = 0xcb;
sfr      P7M0      = 0xe2;
sfr      P7M1      = 0xe1;

sfr      P_SW2     = 0xba;

#define P0INTE      (*(unsigned char volatile xdata *)0xfd00)
#define P0INTF      (*(unsigned char volatile xdata *)0xfd10)
#define P0IM0       (*(unsigned char volatile xdata *)0xfd20)
#define P0IM1       (*(unsigned char volatile xdata *)0xfd30)

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P_SW2 |= 0x80;
    P0IM0 = 0x00;
    P0IM1 = 0x00;
    P0INTE = 0xff;
    P_SW2 &= ~0x80;

    EA = 1;

    while (1);
}
```

// Falling edge interrupt

// Enable P0 port interrupt

//Because the interrupt vector is greater than 31, it cannot be directly compiled in KEIL

//The 13th interrupt entry address must be borrowed

void common_isr() interrupt 13

```
{
    unsigned char psw2_st;
    unsigned char intf;

    psw2_st = P_SW2;
    P_SW2 |= 0x80;
    intf = P0INTF;
    if (intf)
    {
        P0INTF = 0x00;
        if (intf & 0x01)
        {
            //P0.0 interrupt
        }
        if (intf & 0x02)
        {
            //P0.1 interrupt
        }
        if (intf & 0x04)
        {
            //P0.2 interrupt
        }
        if (intf & 0x08)
        {
            //P0.3 interrupt
        }
        if (intf & 0x10)
        {
            //P0.4 interrupt
        }
        if (intf & 0x20)
        {
            //P0.5 interrupt
        }
        if (intf & 0x40)
        {
            //P0.6 interrupt
        }
        if (intf & 0x80)
        {
            //P0.7 interrupt
        }
    }
    P_SW2 = psw2_st;
}
```

// ISR.ASM

// Save the following code as ISP.ASM, and then add the file to the project

CSEG	AT 012BH	;P0 interrupt entry address
JMP	P0INT_ISR	
P0INT_ISR:		
JMP	006BH	; Borrow the entry address of the 13th interrupt
END		

Assembly code

;Operating frequency for test is 11.0592MHz

```

P0M0      DATA      094H
P0M1      DATA      093H
P1M0      DATA      092H
P1M1      DATA      091H
P2M0      DATA      096H
P2M1      DATA      095H
P3M0      DATA      0B2H
P3M1      DATA      0B1H
P4M0      DATA      0B4H
P4M1      DATA      0B3H
P5M0      DATA      0CAH
P5M1      DATA      0C9H
P6M0      DATA      0CCH
P6M1      DATA      0CBH
P7M0      DATA      0E2H
P7M1      DATA      0E1H

P_SW2     DATA      0BAH

P0INTE     XDATA      0FD00H
P0INTF     XDATA      0FD10H
P0IM0      XDATA      0FD20H
P0IM1      XDATA      0FD30H

            ORG        0000H
            LJMP       MAIN

P0INT_ISR:  ORG        012BH                ;P0 interrupt entry address

            PUSH       ACC
            PUSH       B
            PUSH       DPL
            PUSH       DPH
            PUSH       P_SW2

            MOV        DPTR,#P0INTF
            MOVX       A,@DPTR
            MOV        B,A
            CLR        A
            MOVX       @DPTR,A
            MOV        A,B

CHECKP00:   JNB        ACC.0,CHECKP01
            NOP

CHECKP01:   JNB        ACC.1,CHECKP02
            NOP                ;P0.0 interrupt

CHECKP02:   JNB        ACC.2,CHECKP03
            NOP                ;P0.1 interrupt

CHECKP03:   JNB        ACC.3,CHECKP04
            NOP                ;P0.2 interrupt

CHECKP04:   JNB        ACC.4,CHECKP05
            NOP                ;P0.3 interrupt

CHECKP05:   JNB        ACC.5,CHECKP06
            NOP                ;P0.4 interrupt

```

```

NOP                                     ;P0.5 interrupt
CHECKP06:
JNB ACC.6,CHECKP07
NOP                                     ;P0.6 interrupt
CHECKP07:
JNB ACC.7,P0ISREXIT
NOP                                     ;P0.7 interrupt

P0ISREXIT:
POP P_SW2
POP DPH
POP DPL
POP B
POP ACC
RETI

MAIN:
ORG 0200H

MOV SP, #5FH

MOV P0M0, #00H
MOV P0M1, #00H
MOV P1M0, #00H
MOV P1M1, #00H
MOV P2M0, #00H
MOV P2M1, #00H
MOV P3M0, #00H
MOV P3M1, #00H

ORL P_SW2, #80H
CLR A
MOV DPTR, # P0IM0                       ; Falling edge interrupt
MOVX @DPTR, A
MOV DPTR, # P0IM1
MOVX @DPTR, A
MOV DPTR, # P0INTE
MOV A, #0FFH
MOVX @DPTR, A                           ; Enable P0 interrupt
ANL P_SW2, #7FH

SETB EA

JMP $

END

```

12.2.2 P1 rising edge interrupt

C language code

//Operating frequency for test is 11.0592MHz

```
#include "reg51.h"
```

```
#include "intrins.h"
```

```

sfr P0M0    = 0x94;
sfr P0M1    = 0x93;
sfr P1M0    = 0x92;

```

```

sfr      P1M1      = 0x91;
sfr      P2M0      = 0x96;
sfr      P2M1      = 0x95;
sfr      P3M0      = 0xb2;
sfr      P3M1      = 0xb1;
sfr      P4M0      = 0xb4;
sfr      P4M1      = 0xb3;
sfr      P5M0      = 0xca;
sfr      P5M1      = 0xc9;
sfr      P6M0      = 0xcc;
sfr      P6M1      = 0xcb;
sfr      P7M0      = 0xe2;
sfr      P7M1      = 0xe1;

```

```

sfr      P_SW2      = 0xba;

```

```

#define P1INTE      (*(unsigned char volatile xdata *)0xfd01)
#define P1INTF      (*(unsigned char volatile xdata *)0xfd11)
#define P1IM0       (*(unsigned char volatile xdata *)0xfd21)
#define P1IM1       (*(unsigned char volatile xdata *)0xfd31)

```

```

void main()

```

```

{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P_SW2 |= 0x80;
    P1IM0 = 0xff;           // Rising edge interrupt
    P1IM1 = 0x00;
    P1INTE = 0xff;         // Enable P1 interrupt
    P_SW2 &= ~0x80;

    EA = 1;

    while (1);
}

```

//Because the interrupt vector is greater than 31, it cannot be directly compiled in KEIL

//The 13th interrupt entry address must be borrowed

```

void common_isr() interrupt 13

```

```

{
    unsigned char psw2_st;
    unsigned char intf;

    psw2_st = P_SW2;
    P_SW2 |= 0x80;
    intf = P1INTF;
    if (intf)
    {
        P1INTF = 0x00;
    }
}

```

```

    if (intf & 0x01)
    {
                                                //P1.0 interrupt
    }
    if (intf & 0x02)
    {
                                                //P1.1 interrupt
    }
    if (intf & 0x04)
    {
                                                //P1.2 interrupt
    }
    if (intf & 0x08)
    {
                                                //P1.3 interrupt
    }
    if (intf & 0x10)
    {
                                                //P1.4 interrupt
    }
    if (intf & 0x20)
    {
                                                //P1.5 interrupt
    }
    if (intf & 0x40)
    {
                                                //P1.6 interrupt
    }
    if (intf & 0x80)
    {
                                                //P1.7 interrupt
    }
}
P_SW2 = psw2_st;
}

```

// ISR.ASM

// Save the following code as ISP.ASM, and then add the file to the project

```

        CSEG            AT 0133H                ;P1 interrupt entry address
        JMP             P1INT_ISR

P1INT_ISR:
        JMP             006BH                ; Borrow the entry address of the 13th interrupt
        END

```

Assembly code

;Operating frequency for test is 11.0592MHz

P0M0	DATA	094H
P0M1	DATA	093H
P1M0	DATA	092H
P1M1	DATA	091H
P2M0	DATA	096H
P2M1	DATA	095H
P3M0	DATA	0B2H
P3M1	DATA	0B1H
P4M0	DATA	0B4H
P4M1	DATA	0B3H

P5M0	DATA	0CAH	
P5M1	DATA	0C9H	
P6M0	DATA	0CCH	
P6M1	DATA	0CBH	
P7M0	DATA	0E2H	
P7M1	DATA	0E1H	
P_SW2	DATA	0BAH	
P1INTE	XDATA	0FD01H	
P1INTF	XDATA	0FD11H	
P1IM0	XDATA	0FD21H	
P1IM1	XDATA	0FD31H	
	ORG	0000H	
	LJMP	MAIN	
	ORG	0133H	;P1 interrupt entry address
P1INT_ISR:			
	PUSH	ACC	
	PUSH	B	
	PUSH	DPL	
	PUSH	DPH	
	PUSH	P_SW2	
	MOV	DPTR,#P1INTF	
	MOVX	A,@DPTR	
	MOV	B,A	
	CLR	A	
	MOVX	@DPTR,A	
	MOV	A,B	
CHECKP10:			
	JNB	ACC.0,CHECKP11	
	NOP		;P1.0 interrupt
CHECKP11:			
	JNB	ACC.1,CHECKP12	
	NOP		;P1.1 interrupt
CHECKP12:			
	JNB	ACC.2,CHECKP13	
	NOP		;P1.2 interrupt
CHECKP13:			
	JNB	ACC.3,CHECKP14	
	NOP		;P1.3 interrupt
CHECKP14:			
	JNB	ACC.4,CHECKP15	
	NOP		;P1.4 interrupt
CHECKP15:			
	JNB	ACC.5,CHECKP16	
	NOP		;P1.5 interrupt
CHECKP16:			
	JNB	ACC.6,CHECKP17	
	NOP		;P1.6 interrupt
CHECKP17:			
	JNB	ACC.7,P1ISREXIT	
	NOP		;P1.7 interrupt
P1ISREXIT:			
	POP	P_SW2	
	POP	DPH	
	POP	DPL	
	POP	B	

```

    POP        ACC
    RETI

MAIN:
    ORG        0200H

    MOV        SP, #5FH

    MOV        P0M0, #00H
    MOV        P0M1, #00H
    MOV        P1M0, #00H
    MOV        P1M1, #00H
    MOV        P2M0, #00H
    MOV        P2M1, #00H
    MOV        P3M0, #00H
    MOV        P3M1, #00H

    ORL        P_SW2, #80H
    CLR        A
    MOV        DPTR, #P1IM0           ; Falling edge interrupt
    MOVX       @DPTR, A
    MOV        DPTR, #P1IM1
    MOVX       @DPTR, A
    MOV        DPTR, #P1INTE
    MOV        A, #0FFH
    MOVX       @DPTR, A           ; Enable P1 interrupt
    ANL        P_SW2, #7FH

    SETB       EA

    JMP        $

    END

```

12.2.3 P2 low level interrupt

C language code

//Operating frequency for test is 11.0592MHz

```
#include "reg51.h"
```

```
#include "intrins.h"
```

```

sfr    P0M0    =    0x94;
sfr    P0M1    =    0x93;
sfr    P1M0    =    0x92;
sfr    P1M1    =    0x91;
sfr    P2M0    =    0x96;
sfr    P2M1    =    0x95;
sfr    P3M0    =    0xb2;
sfr    P3M1    =    0xb1;
sfr    P4M0    =    0xb4;
sfr    P4M1    =    0xb3;
sfr    P5M0    =    0xca;
sfr    P5M1    =    0xc9;
sfr    P6M0    =    0xcc;
sfr    P6M1    =    0xcb;
sfr    P7M0    =    0xe2;

```



```
sfr      P7M1      =    0xe1;
```

```
sfr      P_SW2     =    0xba;
```

```
#define   P2INTE     (*(unsigned char volatile xdata *)0xfd02)
```

```
#define   P2INTF     (*(unsigned char volatile xdata *)0xfd12)
```

```
#define   P2IM0      (*(unsigned char volatile xdata *)0xfd22)
```

```
#define   P2IM1      (*(unsigned char volatile xdata *)0xfd32)
```

```
void main()
```

```
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    P_SW2 |= 0x80;
    P2IM0 = 0x00;           // low level interrupt
    P2IM1 = 0xff;
    P2INTE = 0xff;         // Enable P2 port interrupt
    P_SW2 &= ~0x80;

    EA = 1;

    while (1);
}
```

//Because the interrupt vector is greater than 31, it cannot be directly compiled in KEIL

//The 13th interrupt entry address must be borrowed

```
void common_isr() interrupt 13
```

```
{
    unsigned char psw2_st;
    unsigned char intf;

    psw2_st = P_SW2;
    P_SW2 |= 0x80;
    intf = P2INTF;
    if (intf)
    {
        P2INTF = 0x00;
        if (intf & 0x01)
        {
            //P2.0 interrupt
        }
        if (intf & 0x02)
        {
            //P2.1 interrupt
        }
        if (intf & 0x04)
        {
            //P2.2 interrupt
        }
    }
}
```

```

    if (intf & 0x08)
    {
                                                //P0.3 interrupt
    }
    if (intf & 0x10)
    {
                                                //P2.4 interrupt
    }
    if (intf & 0x20)
    {
                                                //P2.5 interrupt
    }
    if (intf & 0x40)
    {
                                                //P2.6 interrupt
    }
    if (intf & 0x80)
    {
                                                //P2.7 interrupt
    }
}
P_SW2 = psw2_st;
}

```

// ISR.ASM

// Save the following code as ISP.ASM, and then add the file to the project

```

CSEG                AT 013BH                ;P2 interrupt entry address
JMP                 P2INT_ISR

P2INT_ISR:
JMP                 006BH                ; Borrow the entry address of the 13th interrupt
END

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

P0M0                DATA                094H
P0M1                DATA                093H
P1M0                DATA                092H
P1M1                DATA                091H
P2M0                DATA                096H
P2M1                DATA                095H
P3M0                DATA                0B2H
P3M1                DATA                0B1H
P4M0                DATA                0B4H
P4M1                DATA                0B3H
P5M0                DATA                0CAH
P5M1                DATA                0C9H
P6M0                DATA                0CCH
P6M1                DATA                0CBH
P7M0                DATA                0E2H
P7M1                DATA                0E1H

P_SW2               DATA                0BAH
P2INTE              XDATA                0FD02H
P2INTF              XDATA                0FD12H
P2IM0               XDATA                0FD22H
P2IM1               XDATA                0FD32H

ORG                 0000H
LJMP                MAIN

```

```

P2INT_ISR:    ORG        013BH                ;P2 interrupt entry address

                PUSH      ACC
                PUSH      B
                PUSH      DPL
                PUSH      DPH
                PUSH      P_SW2
                MOV        DPTR,#P2INTF
                MOVX      A,@DPTR
                MOV        B,A
                CLR        A
                MOVX      @DPTR,A
                MOV        A,B

CHECKP20:     JNB        ACC.0,CHECKP21
                NOP                                ;P2.0 interrupt

CHECKP21:     JNB        ACC.1,CHECKP22
                NOP                                ;P2.1 interrupt

CHECKP22:     JNB        ACC.2,CHECKP23
                NOP                                ;P2.2 interrupt

CHECKP23:     JNB        ACC.3,CHECKP24
                NOP                                ;P2.3 interrupt

CHECKP24:     JNB        ACC.4,CHECKP25
                NOP                                ;P2.4 interrupt

CHECKP25:     JNB        ACC.5,CHECKP26
                NOP                                ;P2.5 interrupt

CHECKP26:     JNB        ACC.6,CHECKP27
                NOP                                ;P2.6 interrupt

CHECKP27:     JNB        ACC.7,P2ISREXIT
                NOP                                ;P2.7 interrupt

P2ISREXIT:    POP        P_SW2
                POP        DPH
                POP        DPL
                POP        B
                POP        ACC
                RETI

MAIN:        ORG        0200H

                MOV        SP, #5FH

                MOV        P0M0,#00H
                MOV        P0M1,#00H
                MOV        P1M0,#00H
                MOV        P1M1,#00H
                MOV        P2M0,#00H
                MOV        P2M1,#00H
                MOV        P3M0,#00H
                MOV        P3M1,#00H

                ORL        P_SW2,#80H
                CLR        A
                MOV        DPTR,# P2IM0                ; low level interrupt
                MOVX      @DPTR,A
                MOV        DPTR,# P2IM1
                MOVX      @DPTR,A
                MOV        DPTR,# P2INTE
                MOV        A,#0FFH
                MOVX      @DPTR,A                ;enable P2 interrupt
                ANL        P_SW2,#7FH

                SETB      EA

```

JMP \$

END

STC MCU

12.2.4 Port3 high level interrupt

C language code

//Operating frequency for test is 11.0592MHz

#include "reg51.h"

#include "intrins.h"

sfr P0M0 = 0x94;

sfr P0M1 = 0x93;

sfr P1M0 = 0x92;

sfr P1M1 = 0x91;

sfr P2M0 = 0x96;

sfr P2M1 = 0x95;

sfr P3M0 = 0xb2;

sfr P3M1 = 0xb1;

sfr P4M0 = 0xb4;

sfr P4M1 = 0xb3;

sfr P5M0 = 0xca;

sfr P5M1 = 0xc9;

sfr P6M0 = 0xcc;

sfr P6M1 = 0xcb;

sfr P7M0 = 0xe2;

sfr P7M1 = 0xe1;

sfr P_SW2 = 0xba;

#define P3INTE ((unsigned char volatile xdata *)0xfd03)*

#define P3INTF ((unsigned char volatile xdata *)0xfd13)*

#define P3IM0 ((unsigned char volatile xdata *)0xfd23)*

#define P3IM1 ((unsigned char volatile xdata *)0xfd33)*

void main()

{

P0M0 = 0x00;

P0M1 = 0x00;

P1M0 = 0x00;

P1M1 = 0x00;

P2M0 = 0x00;

P2M1 = 0x00;

P3M0 = 0x00;

P3M1 = 0x00;

P4M0 = 0x00;

P4M1 = 0x00;

```
P5M0 = 0x00;
P5M1 = 0x00;

P_SW2 |= 0x80;

P3IM0 = 0xff;           // high level interrupt
P3IM1 = 0xff;

P3INTE = 0xff;          //Enable P3 interrupt

P_SW2 &= ~0x80;

EA = 1;
while (1);
}
```

//Because the interrupt vector is greater than 31, it cannot be directly compiled in KEIL

//The 13th interrupt entry address must be borrowed

void common_isr() interrupt 13

```
{
    unsigned char psw2_st;
    unsigned char intf;

    psw2_st = P_SW2;
    P_SW2 |= 0x80;
    intf = P3INTF;
    if (intf)
    {
        P3INTF = 0x00;
        if (intf & 0x01)
        {
            //P3.0 interrupt
        }
        if (intf & 0x02)
        {
            //P3.1 interrupt
        }
        if (intf & 0x04)
        {
            //P3.2 interrupt
        }
        if (intf & 0x08)
        {
            //P3.3 interrupt
        }
    }
}
```

```

        if (intf & 0x10)
        {
                                                    //P3.4 interrupt
        }
        if (intf & 0x20)
        {
                                                    //P3.5 interrupt
        }
        if (intf & 0x40)
        {
                                                    //P3.6 interrupt
        }
        if (intf & 0x80)
        {
                                                    //P3.7 interrupt
        }
    }
    P_SW2 = psw2_st;
}
// ISR.ASM
//Save the following code as ISP.ASM, and then add the file to the project
        CSEG          AT 0143H          ; P3 port interrupt entry address
        JMP          P3INT_ISR
P3INT_ISR:
        JMP          006BH          ; Borrow the entry address of the 13th interrupt
        END

```

Assembly code

;Operating frequency for test is 11.0592MHz

P0M0	DATA	094H
P0M1	DATA	093H
P1M0	DATA	092H
P1M1	DATA	091H
P2M0	DATA	096H
P2M1	DATA	095H
P3M0	DATA	0B2H
P3M1	DATA	0B1H
P4M0	DATA	0B4H
P4M1	DATA	0B3H
P5M0	DATA	0CAH
P5M1	DATA	0C9H
P6M0	DATA	0CCH

<i>P6M1</i>	<i>DATA</i>	<i>0CBH</i>
<i>P7M0</i>	<i>DATA</i>	<i>0E2H</i>
<i>P7M1</i>	<i>DATA</i>	<i>0E1H</i>
<i>P_SW2</i>	<i>DATA</i>	<i>0BAH</i>
<i>P3INTE</i>	<i>XDATA</i>	<i>0FD03H</i>
<i>P3INTF</i>	<i>XDATA</i>	<i>0FD13H</i>
<i>P3IM0</i>	<i>XDATA</i>	<i>0FD23H</i>
<i>P3IM1</i>	<i>XDATA</i>	<i>0FD33H</i>

ORG 0000H

LJMP MAIN

ORG 0143H ;P3 interrupt entry address

P3INT_ISR:

PUSH ACC
PUSH B
PUSH DPL
PUSH DPH
PUSH P_SW2

MOV DPTR,#P3INTF
MOVB A,@DPTR
MOVB A,B
CLRA
MOVB A,@DPTR
MOVB A,B

CHECKP30:

JNB ACC.0,CHECKP31
NOP ;P3.0 interrupt

CHECKP31:

JNB ACC.1,CHECKP32
NOP ;P3.1 interrupt

CHECKP32:

JNB ACC.2,CHECKP33
NOP ;P3.2 interrupt

CHECKP33:

JNB ACC.3,CHECKP34
NOP ;P3.3 interrupt

CHECKP34:

JNB ACC.4,CHECKP35
NOP ;P3.4 interrupt

CHECKP35:

JNB ACC.5,CHECKP36


```

        NOP                                ;P3.5 interrupt
CHECKP36:
        JNB      ACC.6,CHECKP37
        NOP                                ;P3.6 interrupt
CHECKP37:
        JNB      ACC.7,P3ISREXIT
        NOP                                ;P3.7 interrupt
P3ISREXIT:
        POP      P_SW2
        POP      DPH
        POP      DPL
        POP      B
        POP      ACC
        RETI

MAIN:
        ORG      0200H

        MOV      SP, #5FH

        MOV      P0M0,#00H
        MOV      P0M1,#00H
        MOV      P1M0,#00H
        MOV      P1M1,#00H
        MOV      P2M0,#00H
        MOV      P2M1,#00H
        MOV      P3M0,#00H
        MOV      P3M1,#00H

        ORL      P_SW2,#80H
        CLR A
        MOV      DPTR,# P3IM0                ; high level interrupt
        MOVX     @DPTR,A
        MOV      DPTR,# P3IM1
        MOVX     @DPTR,A
        MOV      DPTR,# P3INTE
        MOV      A,#0FFH
        MOVX     @DPTR,A                ;enable P3 interrupt
        ANL      P_SW2,#7FH
        SETB     EA
        JMP      $

        END

```

STC MCU

13 Timer/Counter

Product line	Number of timers
STC8H1K08 family	3
STC8H1K28 family	5
STC8H3K64S4 family	5
STC8H3K64S2 family	5
STC8H8K64U family	5
STC8H2K64T family	5
STC8H4K64TLR family	5
STC8H4K64TLCD family	5
STC8H4K64LCD family	5

Five 16-bit Timers/Counters are integrated in STC8H series of microcontrollers: T0, T1, T2, T3 and T4. All of them can be used as timer or counter. For T0 and T1, the ‘timer’ or ‘counter’ function is selected by the control bits C/T in the special function register TMOD. For T2, the ‘timer’ or ‘counter’ function is selected by the control bit T2_C/T in the special function register AUXR. For T3, the ‘timer’ or ‘counter’ function is selected by the control bit T3_C/T in the special function register T4T3M. For T4, the ‘timer’ or ‘counter’ function is selected by the control bit T4_C/T in the special function register T4T3M. The core of the timer/counter is a up counter, the essence of which is counting pulses. The only difference of ‘timer’ mode and ‘counter’ mode is the different counting pulses sources. If the counting pulse is from the system clock, the timer/counter runs in the timing mode, it counts once every 12 clocks or one clock. If the counting pulse is from the microcontroller external pins, the timer/counter runs in counting mode, it counts once every pulse.

When T0, T1 and T2 are operating in ‘timer’ mode, T0x12, T1x12 and T2x12 in AUXR register are used to determine the clocks of T0, T1 and T2 are system clock/12 or system clock/1. When T3 and T4 are operating in ‘timer’ mode, T3x12 and T4x12 in the T4T3M register determine the clocks of T3 and T4 are system clock/12 or system clock/1. When the timer/counters are operating in ‘counter’ mode, the frequency of the external pulse is not divided.

T0 has four operating modes which are selected by bit-pairs (M1, M0) in TMOD. The four modes are mode 0 (16-bit auto-reload mode), mode 1 (16-bit non-auto-reload mode), mode 2 (8-bit auto-reload mode) and mode 3 (16-bit auto-reload mode whose interrupt can not be disabled). And for T1, all modes except mode 3 are the same as T0. The mode 3 of T1 is invalid and stops counting. For T2, T3 and T4, they only have one mode: 16-bit auto-reload mode. Besides being used as timer/counters, T2, T3 and T4 can also be as the baud-rate generators of UARTs and programmable clock outputs.

13.1 Registers Related to Timers

Symbol	Description	Address	Bit Address and Symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
TCON	Timer 0 and 1 control register	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0000,0000
TMOD	Timer 0 and 1 mode register	89H	GATE	C/T	M1	M0	GATE	C/T	M1	M0	0000,0000
TL0	Timer 0 low byte register	8AH									0000,0000
TL1	Timer 1 low byte register	8BH									0000,0000
TH0	Timer 0 high byte register	8CH									0000,0000
TH1	Timer 1 high byte register	8DH									0000,0000
AUXR	Auxiliary register 1	8EH	T0x12	T1x12	UART_M0x6	T2R	T2_C/T	T2x12	EXTRAM	S1ST2	0000,0001
INTCLKO	External interrupt and clock output control register	8FH	-	EX4	EX3	EX2	-	T2CLKO	T1CLKO	T0CLKO	x000,x000
WKTCL	Wake-up Timer Control Register low	AAH									1111,1111
WKTCH	Wake-up Timer Control Register high byte	ABH	WKTEN								0111,1111
T4T3M	Timer4 and Timer 3 Control Register	D1H	T4R	T4_C/T	T4x12	T4CLKO	T3R	T3_C/T	T3x12	T3CLKO	0000,0000
T4H	Timer 4 high byte register	D2H									0000,0000
T4L	Timer 4 low byte register	D3H									0000,0000
T3H	Timer 3 high byte register	D4H									0000,0000
T3L	Timer 3 low byte register	D5H									0000,0000

T2H	Timer 2 high byte register	D6H		0000,0000
T2L	Timer 2 low byte register	D7H		0000,0000

Symbol	Description	Address	Bit Address and Symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
TM2PS	Timer 2 clock prescaler register	FEA2H									0000,0000
TM3PS	Timer 3 clock prescaler register	FEA3H									0000,0000
TM4PS	Timer 4 clock prescaler register	FEA4H									0000,0000

13.2 Timer 0/1

13.2.1 Timer 0 and 1 Control Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
TCON	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1: T1 overflow flag. After T1 is enabled to count, it performs adding 1 count from the initial value. TF1 is set by hardware on T1 overflow and requests interrupt to CPU. It will keep the status until CPU responds the interrupt and is cleared by hardware automatically. It also can be cleared by software.

TR1: T1 run control bit. It is set or cleared by software to turn the timer on/off. If GATE (TMOD.7) = 0, T1 will start counting as soon as TR1=1 and stop counting when TR1=0. If GATE (TMOD.7) = 1, T1 is enabled to count only if TR1 = 1 and INT1 is high.

TF0: T0 overflow flag. After T0 is enabled to count, it performs adding 1 count from the initial value. TF0 is set by hardware on T0 overflow and requests interrupt to CPU. It will keep the status until CPU responds the interrupt and is cleared by hardware automatically. It also can be cleared by software.

TR0: T0 run control bit. It is set or cleared by software to turn the timer on/off. If GATE (TMOD.3) = 0, T0 will start counting as soon as TR0=1 and stop counting when TR0=0. If GATE (TMOD.0) = 1, T0 is enabled to count only if TR0 = 1 and INT0 is high.

IE1: External Interrupt 1 (INT1/P3.3) request flag. IE1=1 means external interrupt requests interrupt to CPU. It is cleared by hardware automatically when the CPU responds to the interrupt.

IT1: External Interrupt 1 trigger edge type control bit. If IT1 = 0, INT1 can be triggered by both rising and falling edges. If IT1 = 1, INT1 can be triggered only by falling edge.

IE0: External Interrupt 0 (INT0/P3.2) request flag. IE0=1 means external interrupt requests interrupt to CPU. It is cleared by hardware automatically when the CPU responds to the interrupt.

IT0: External Interrupt 0 trigger edge type control bit. If IT0 = 0, INT0 can be triggered by both rising and falling edges. If IT0 = 1, INT0 can be triggered only by falling edge.

13.2.2 Timer 0/1 Mode Register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
TMOD	89H	T1_GATE	T1_C/T	T1_M1	T1_M0	T0_GATE	T0_C/T	T0_M1	T0_M0

T1_GATE: T1 gate control. If GATE/TMOD.7=1, T1 starts only when TR1 is set AND INT1 pin is high.

T0_GATE: T0 gate control. If GATE/TMOD.3 =1, T0 starts only when TR0 is set AND INT0 pin is high.

T1_C/T: T1 mode select bit. If it is reset, T1 is used as a timer (input pulse is from internal system clock). If it is set, T1 is used as a counter (input pulse is from external T1/P3.5 pin).

T0_C/T: T0 mode select bit. If it is reset, T0 is used as a timer (input pulse is from internal system clock). If it is set, T0 is used as a counter (input pulse is from external T0/P3.4 pin).

T1_M1/T1_M0: T1 mode select bits.

T1_M1	T1_M0	T1 operating mode
0	0	16-bit auto-reload mode.

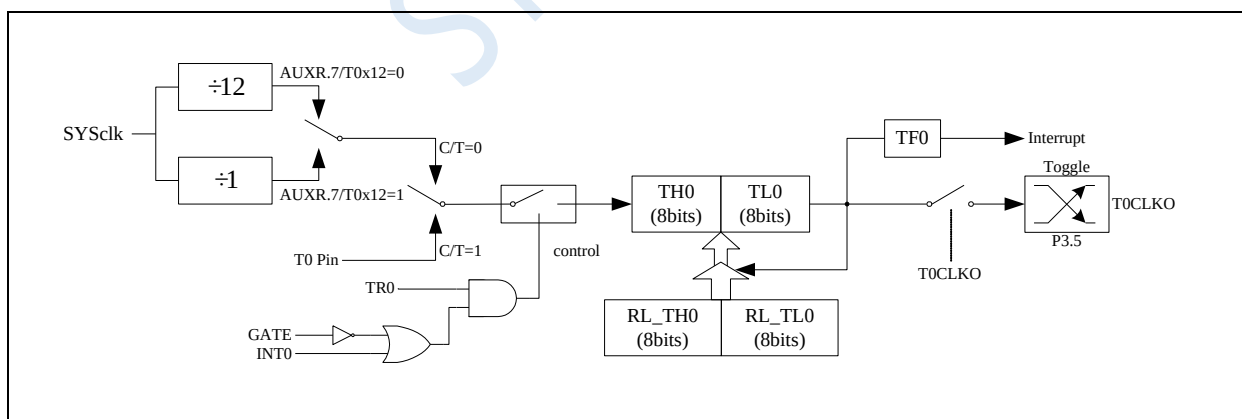
		When the 16-bit counter [TH1, TL1] overflows, the system loads the reload value in the internal 16-bit reload register into [TH1, TL1] automatically.
0	1	16-bit non-auto-reload mode. When the 16-bit counter [TH1, TL1] overflows, timer 1 will count from 0.
1	0	8-bit auto-reload mode. When the 8-bit counter TL1 overflows, the system loads the reload value in TH1 into TL1 automatically.
1	1	T1 stops working.

T0_M1/T0_M0: T0 mode select bits.

T0_M1	T0_M0	T0 operating mode
0	0	16-bit auto-reload mode. When the 16-bit counter [TH0, TL0] overflows, the system loads the reload value in the internal 16-bit reload register into [TH0, TL0] automatically.
0	1	16-bit non-auto-reload mode. When the 16-bit counter [TH1, TL1] overflows, timer 1 will count from 0.
1	0	8-bit auto-reload mode. When the 8-bit counter TL0 overflows, the system loads the reload value in TH0 into TL0 automatically.
1	1	16-bit auto-reload mode. It is similar to mode 0, whose interrupt can not be disabled. The interrupt has the highest priority, higher than the priority of all other interrupts, and cannot be turned off. It can be used as the system tick timer of the operating system or the system monitoring timer. The only way to stop is to turn off the TR0 bit in the TCON register and stop supplying the clock to Timer 0.

13.2.3 Timer0 mode 0 (16-bit auto-reloadable mode)

In this mode, Timer/Counter 0 is used as a 16-bit counter that can be automatically reloaded, as shown in the figure below.



Timer/Counter0 mode 0: 16-bit auto-reload mode

When GATE=0 (TMOD.3), the timer will count if TR0=1. When GATE=1, it is allowed to control timer0 by external input INT0, so that pulse width measurement can be realized. TR0 is the control bit in the TCON register. For the specific function description of each bit of the TCON register, see the introduction of the TCON register in the previous section.

When C/T=0, the multiplexer is connected to the frequency division output of the system clock. T0 counts the internal system clock, and works in timing mode. When C/T=1, the multiplexer is connected to the external pulse input P3.4/T0, and T0 works in counting mode.

Timer0 of STC microcontroller has two counting rates: one is 12T mode, which is increased by 1 for every 12 clocks, which is the same as traditional 8051 microcontroller, the other is 1T mode, which is increased by 1

for each clock, and the speed is 12 times of traditional 8051. The rate of T0 is determined by T0x12 in the special function register AUXR. If T0x12=0, T0 works in 12T mode, and if T0x12=1, T0 works in 1T mode.

Timer0 has two hidden registers RL_TH0 and RL_TL0. RL_TH0 and TH0 share the same address, and RL_TL0 and TL0 share the same address. When TR0=0, that is, when Timer/Counter0 is disabled, the content written to TL0 will be written to RL_TL0 at the same time, and the content written to TH0 will also be written to RL_TH0 at the same time. When TR0=1, that is, when Timer/Counter0 is allowed to work, writing content to TL0 is not actually written to the current register TL0, but written to the hidden register RL_TL0, and writing content to TH0 is actually also it is not written into the current register TH0, but into the hidden register RL_TH0, which can cleverly realize the 16-bit reload timer. When reading the contents of TH0 and TL0, the contents be read are the contents of TH0 and TL0, not the contents of RL_TH0 and RL_TL0.

When Timer0 is working in mode 0 (TMOD[1:0]/[M1,M0]=00B), the overflow of [TH0,TL0] not only sets TF0, but also automatically reloads the contents of [RL_TH0,RL_TL0] to [TH0,TL0].

If T0CLKO/INT_CLKO.0=1, the P3.5/T1 pin is configured as timer 0's clock output T0CLKO. The output clock frequency is $T0 \text{ overflow rate}/2$.

If C/T=0, the timer/counter 0 counts the internal system clock, then:

if T0 works in 1T mode (AUXR.7/T0x12=1), the output clock frequency = $(SYSclk)/(65536-[RL\_TH0, RL\_TL0])/2$

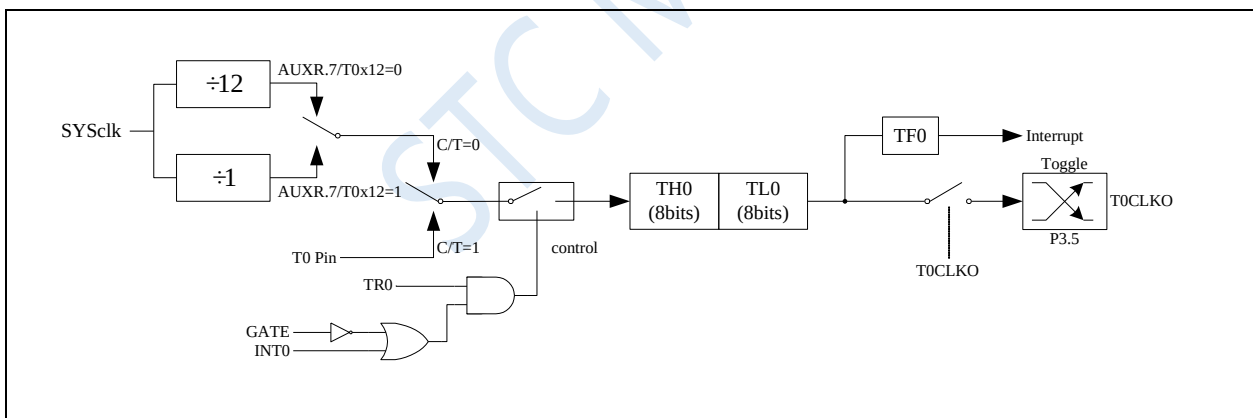
if T0 works in 12T mode (AUXR.7/T0x12=0), the output clock frequency = $(SYSclk)/12/(65536-[RL\_TH0, RL\_TL0])/2$

If C/T=1, the timer/counter 0 counts the external pulse input (P3.4/T0), then:

the output clock frequency = $(T0\_Pin\_CLK) / (65536-[RL\_TH0, RL\_TL0])/2$

13.2.4 Timer0 mode 1 (16-bit non-autoreloadable mode)

In this mode, Timer/Counter 0 works in 16-bit non-reloadable mode, as shown in the figure below.



Timer/counter 0 mode 1: 16-bit non-reloadable mode

In this mode, Timer/Counter 0 is configured as a 16-bit non-reloadable mode, which is composed of 8 bits of TL0 and 8 bits of TH0. The 8-bit overflow of TL0 carries over to TH0, and the overflow of TH0 counts the overflow flag TF0 in TCON.

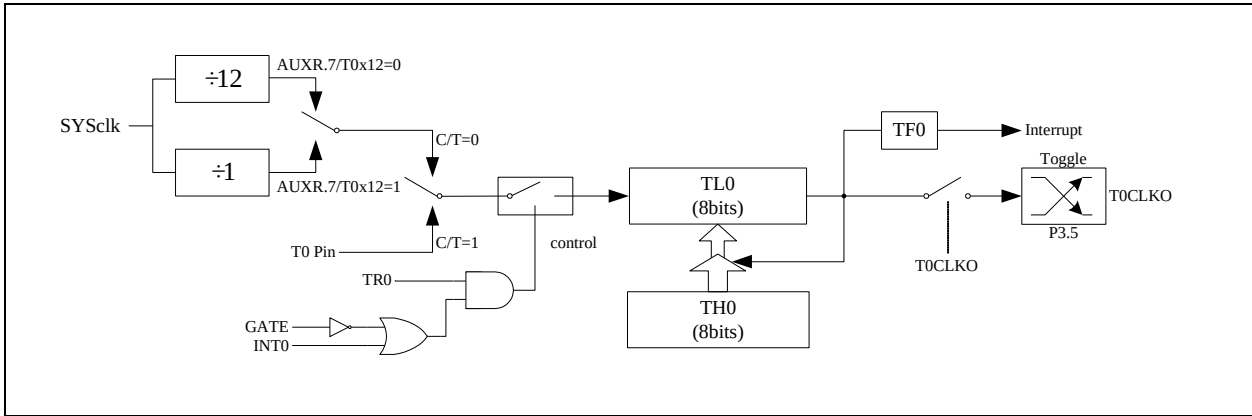
When GATE=0 (TMOD.3), the timer will count if TR0=1. When GATE=1, it is allowed to control timer 0 by external input INT0, so that pulse width measurement can be realized. TR0 is the control bit in the TCON register. For the specific function description of each bit of the TCON register, see the introduction of the TCON register in the previous section.

When C/T=0, the multiplexer is connected to the frequency division output of the system clock, T0 counts the internal system clock, and works in timing mode. When C/T=1, the multiplexer is connected to the external pulse input P3.4/T0, and T0 works in counting mode.

Timer0 of STC microcontroller has two counting rates: one is 12T mode, which is increased by 1 for every 12 clocks, which is the same as traditional 8051 microcontroller, the other is 1T mode, which is increased by 1 for each clock, and the speed is 12 times of traditional 8051. The rate of T0 is determined by T0x12 in the special function register AUXR. If T0x12=0, T0 works in 12T mode, and if T0x12=1, T0 works in 1T mode.

13.2.5 Timer 0 mode 2 (8-bit auto-reloadable mode)

In this mode, Timer/Counter 0 is an 8-bit counter that can be automatically reloaded, as shown in the figure below.



Timer/counter 0 mode 2: 8-bit auto-reloadable mode

The overflow of TL0 not only sets TF0, but also reloads the content of TH0 into TL0. The content of TH0 is preset by software, and its content remains unchanged during reloading.

When T0CLKO/INT_CLKO.0=1, the P3.5/T1 pin is configured as timer 0's clock output T0CLKO. The output clock frequency is [T0 overflow rate/2](#).

If C/T=0, the timer/counter 0 counts the internal system clock, then:

if T0 works in 1T mode (AUXR.7/T0x12=1), the output clock frequency = [\(SYSClk\)/\(256-TH0\)/2](#)

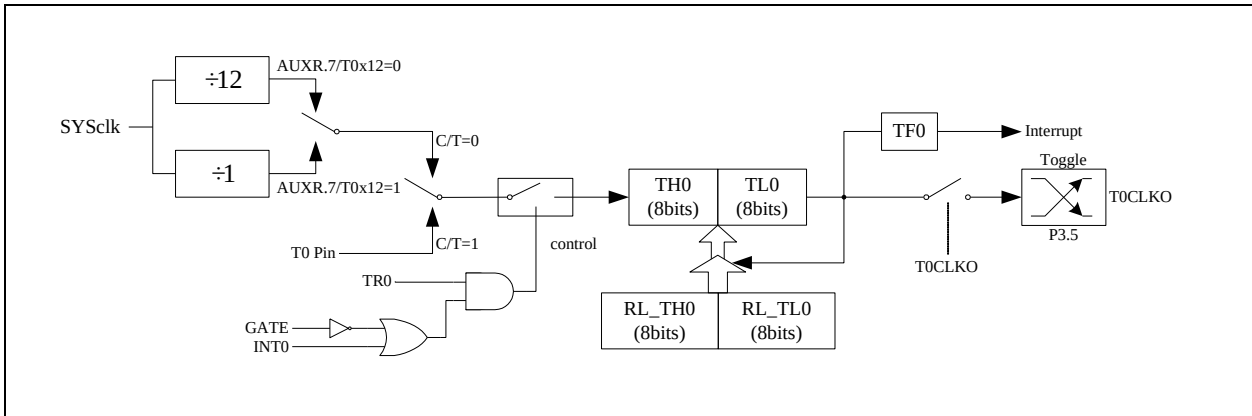
if T0 works in 12T mode (AUXR.7/T0x12=0), the output clock frequency = [\(SYSClk\)/12/\(256-TH0\)/2](#)

If C/T=1, the timer/counter T0 counts the external pulse input (P3.4/T0), then:

Output clock frequency = [\(T0_Pin_CLK\) / \(256-TH0\)/2](#)

13.2.6 Timer 0 mode 3 (16-bit auto-reloadable mode with non-maskable interrupt, which can be used as real-time operating system metronome)

For timer/counter 0, its working mode 3 is the same as working mode 0 (the schematic diagram of timer mode 3 in the figure below is the same as working mode 0). The only difference is: when timer/counter 0 is working in mode 3, its interrupt can be enabled just setting ET0/IE.1 (timer/counter 0 interrupt enable bit), and EA/IE.7 (total interrupt enable bit) is not required. The timer/counter 0 interrupt in this mode has nothing to do with the total interrupt enable bit EA. Once the timer/counter 0 interrupt working in mode 3 is enabled (ET0=1), then the interrupt is non-maskable, and the priority of the interrupt is the highest, that is, the interrupt cannot be interrupted by any interrupt, and the interrupt is neither controlled by EA/IE.7 nor controlled by ET0 after it is enabled. When EA=0 or ET0=0, this interrupt cannot be disabled. This mode is so called the 16-bit automatic reload mode with non-maskable interrupt.

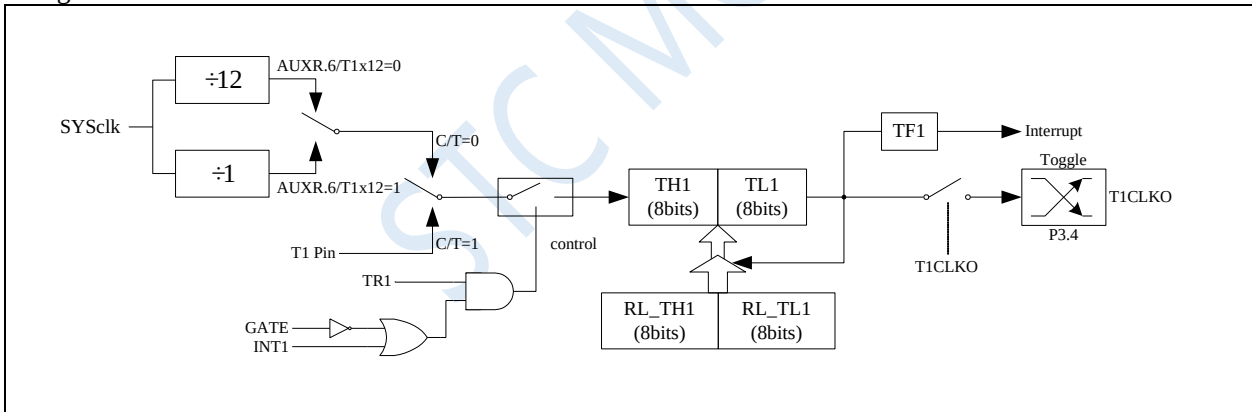


Timer/counter 0 mode 3: 16-bit auto-reload mode with non-maskable interrupt

Note: When Timer/Counter 0 works in mode 3 (16-bit auto-reload mode with non-maskable interrupt), it is not necessary to enable EA/IE.7 (total interrupt enable bit), only ET0/IE.1 is required. (Timer/counter 0 interrupt enable bit) can turn on the timer/counter 0 interrupt. The timer/counter 0 interrupt in this mode has nothing to do with the total interrupt enable bit EA. Once the timer/counter 0 interrupt in this mode is enabled, the timer/counter 0 interrupt priority is the highest, and it cannot be interrupted by any other interrupt (no matter it is lower than the timer/counter 0 interrupt priority). After the interrupt in this mode is enabled, it is neither controlled by EA/IE.7 nor controlled by ET0. Clearing EA nor ET0 can not disable this interrupt.

13.2.7 Timer 1 mode 0 (16-bit auto-reloadable mode)

In this mode, Timer/Counter 1 is used as a 16-bit counter that can be automatically reloaded, as shown in the figure below.



Timer/Counter 1 mode 0: 16-bit auto-reload mode

When GATE=0 (TMOD.7), the timer will count if TR1=1. When GATE=1, it is allowed to control timer1 by external input INT1, so that pulse width measurement can be realized. TR1 is the control bit in the TCON register. For the specific function description of each bit of the TCON register, see the introduction of the TCON register in the previous section.

When C/T=0, the multiplexer is connected to the frequency division output of the system clock. T1 counts the internal system clock, and works in timing mode. When C/T=1, the multiplexer is connected to the external pulse input P3.5/T1, and T1 works in counting mode.

Timer1 of STC microcontroller has two counting rates: one is 12T mode, which is increased by 1 for every 12 clocks, which is the same as traditional 8051 microcontroller, the other is 1T mode, which is increased by 1 for each clock, and the speed is 12 times of traditional 8051. The rate of T1 is determined by T1x12 in the special function register AUXR. If T1x12=0, T1 works in 12T mode, and if T1x12=1, T1 works in 1T mode.

Timer1 has two hidden registers RL_TH1 and RL_TL1. RL_TH1 and TH1 share the same address, and RL_TL1 and TL1 share the same address. When TR1=0, that is, when Timer/Counter1 is disabled, the content written to TL1 will be written to RL_TL1 at the same time, and the content written to TH1 will also be written to RL_TH1 at the same time. When TR1=1, that is, when Timer/Counter1 starts to work, writing content to TL1

is not actually written to the current register TL1, but written to the hidden register RL_TL1, and writing content to TH1 is actually also it is not written into the current register TH1, but into the hidden register RL_TH1, which can cleverly realize the 16-bit reload timer. When reading the contents of TH1 and TL1, the contents be read are the contents of TH1 and TL1, not the contents of RL_TH1 and RL_TL1.

When Timer1 is working in mode 0 (TMOD[5:4]/[M1,M0]=00B), the overflow of [TH1,TL1] not only sets TF1, but also automatically reloads the contents of [RL_TH1,RL_TL1] to [TH1,TL1].

If T1CLKO/INT_CLKO.1=1, the P3.4/T1 pin is configured as timer 1's clock output T1CLKO. The output clock frequency is $T1 \text{ overflow rate}/2$.

If C/T=0, the timer/counter 1 counts the internal system clock, then:

if T1 works in 1T mode (AUXR.6/T1x12=1), the output clock frequency = $(SYSclk)/(65536-[RL\_TH1, RL\_TL1])/2$

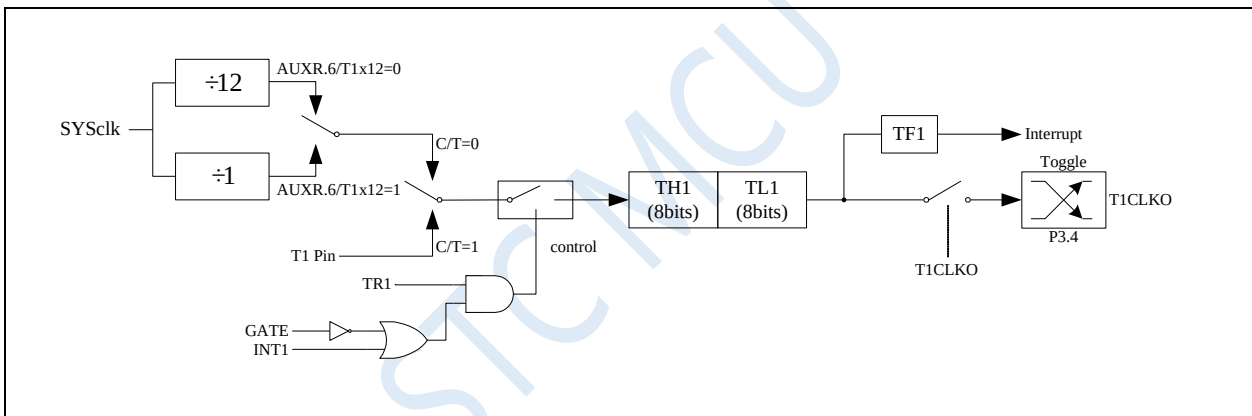
if T1 works in 12T mode (AUXR.6/T1x12=0), the output clock frequency = $(SYSclk)/12/(65536-[RL\_TH1, RL\_TL1])/2$

If C/T=1, the timer/counter 1 counts the external pulse input (P3.5/T1), then:

the output clock frequency = $(T1\_Pin\_CLK) / (65536-[RL\_TH1, RL\_TL1])/2$

13.2.8 Timer1 mode 1 (16-bit non-autoreloadable mode)

In this mode, Timer/Counter 1 works in 16-bit non-reloadable mode, as shown in the figure below.



Timer/counter 1 mode 1: 16-bit non-reloadable mode

In this mode, Timer/Counter 1 is configured as a 16-bit non-reloadable mode, which is composed of 8 bits of TL1 and 8 bits of TH1. The 8-bit overflow of TL1 carries over to TH1, and the overflow of TH1 counts the overflow flag TF1 in TCON.

When GATE=0 (TMOD.7), the timer will count if TR1=1. When GATE=1, it is allowed to control timer 1 by external input INT1, so that pulse width measurement can be realized. TR1 is the control bit in the TCON register. For the specific function description of each bit of the TCON register, see the introduction of the TCON register in the previous section.

When C/T=0, the multiplexer is connected to the frequency division output of the system clock, T1 counts the internal system clock, and works in timing mode. When C/T=1, the multiplexer is connected to the external pulse input P3.5/T1, and T1 works in counting mode.

Timer1 of STC microcontroller has two counting rates: one is 12T mode, which is increased by 1 for every 12 clocks, which is the same as traditional 8051 microcontroller, the other is 1T mode, which is increased by 1 for each clock, and the speed is 12 times of traditional 8051. The rate of T1 is determined by T1x12 in the special function register AUXR. If T1x12=0, T1 works in 12T mode, and if T1x12=1, T1 works in 1T mode.

13.2.9 Timer 1 mode 2 (8-bit auto-reloadable mode)

In this mode, Timer/Counter 1 is an 8-bit counter that can be automatically reloaded, as shown in the figure

0: The clock source of T0 is SYSclk/12.

1: The clock source of T0 is SYSclk/1.

T1x12: T1 speed control bit.

0: The clock source of T1 is SYSclk/12.

1: The clock source of T1 is SYSclk/1.

13.2.13 External Interrupt and Clock Output Control Register (INTCLKO)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
INTCLKO	8FH	-	EX4	EX3	EX2	-	T2CLKO	T1CLKO	T0CLKO

T0CLKO: T0 clock out control bit.

0: Turn off the clock output.

1: P3.5 is configured for T0 clock output pin. When T0 overflows, P3.5 will flip automatically.

T1CLKO: T0 clock out control bit.

0: Turn off the clock output.

1: P3.4 is configured for T1 clock output pin. When T1 overflows, P3.4 will flip automatically.

13.2.14 Timer 0 calculation formula

Mode of Timer	Speed	Period calculation formula
Mode 0/3 (16-bit automatic reload)	1T	$\text{Timer period} = \frac{65536 \div TH0, TLO}{SYSclk}$ (Auto-reload)
	12T	$\text{Timer period} = \frac{65536 \div TH0, TLO}{SYSclk}$ (Auto-reload)
Mode 1 (16-bit does not automatically reload)	1T	$\text{Timer period} = \frac{65536 \div TH0, TLO}{SYSclk}$ (Soft reload)
	12T	$\text{Timer period} = \frac{65536 \div TH0, TLO}{SYSclk}$ (Soft reload)
Mode 2 (8-bit automatic reload)	1T	$\text{Timer period} = \frac{256 \div TH0}{SYSclk}$

	12T	(Auto-reload)

13.2.15 Timer 1 calculation formula

Mode of Timer	Speed	Period calculation formula
Mode 0 (16-bit automatic reload)	1T	$\text{Timer period} = \frac{65536 \mid TH1, TL1}{SYSclk}$ (Auto-reload)
	12T	(Auto-reload)
Mode 1 (16-bit does not automatically reload)	1T	$\text{Timer period} = \frac{65536 \mid TH1, TL1}{SYSclk}$ (Soft reload)
	12T	(Soft reload)
Mode 2 (8-bit automatic reload)	1T	$\text{Timer period} = \frac{256 \mid TH1}{SYSclk}$ (Auto-reload)
	12T	(Auto-reload)

13.3 Timer 2

13.3.1 Auxiliary Register 1 (AUXR)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
AUXR	8EH	T0x12	T1x12	UART_M0x6	T2R	T2_C/T	T2x12	EXTRAM	S1ST2

TR2: T2 run control bit.

0: T2 stops counting.

1: T2 start counting.

T2_C/T: T2 mode select bit.

0: T2 is used as a timer (input pulse is from internal system clock);

1: T2 is used as a counter (input pulse is from external T2/P1.2 pin).

T2x12: T2 speed control bit.

0: The clock source of T2 is SYSclk/12.

1: The clock source of T2 is SYSclk/1.

13.3.2 External Interrupt and Clock Output Control Register (INTCLKO)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
INTCLKO	8FH	-	EX4	EX3	EX2	-	T2CLKO	T1CLKO	T0CLKO

T2CLKO: T2 clock out control bit.

0: Turn off the clock output.

1: P1.3 is configured for T2 clock output pin. When T2 overflows, P1.3 will flip automatically.

13.3.3 Timer 2 Counting Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
T2L	D7H								
T2H	D6H								

T2 operates in 16-bit auto-reload mode. T2L and T2H combine into a 16-bit register with T2L as the low byte and T2H as the high byte. When the 16-bit counter [T2H, T2L] overflows, the system loads the reload value in the internal 16-bit reload register into [T2H, T2L] automatically.

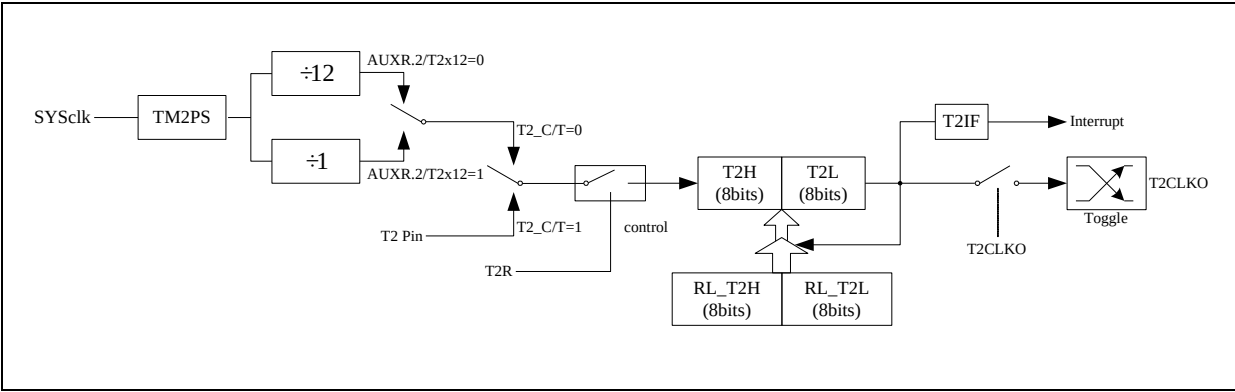
13.3.4 Timer 2 8-bit Prescaler Register (TM2PS)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
TM2PS	FEA2H								

Timer 2 clock = SYSclk ÷ (TM2PS + 1)

13.3.5 Timer 2 working mode

The functional block diagram of Timer/Counter 2 is as follows.



Timer/counter 2 working mode: 16-bit auto-reload mode

T2R/AUXR.4 is the control bit in the AUXR register. For the specific function description of each bit of the AUXR register, see the introduction of the AUXR register in the previous section.

When T2_C/T=0, the multiplexer is connected to the frequency division output of the system clock, T2 counts the internal system clock, and works in timing mode. When T2_C/T=1, the multiplexer is connected to the external pulse input T2, and T2 works in counting mode.

Timer2 of STC microcontroller has two counting rates: one is 12T mode, which is increased by 1 for every 12 clocks, which is the same as traditional 8051 microcontroller, the other is 1T mode, which is increased by 1 for each clock, and the speed is 12 times of traditional 8051. The rate of T2 is determined by T2x12 in the special function register AUXR. If T2x12=0, T2 works in 12T mode, and if T2x12=1, T1 works in 1T mode.

Timer2 has two hidden registers RL_T2H and RL_T2L. RL_T2H and T2H share the same address, and RL_T2L and T2L share the same address. When T2R=0, that is, when Timer/Counter2 is disabled, the content written to T2L will be written to RL_T2L at the same time, and the content written to T2H will also be written to RL_T2H at the same time. When T2R=1, that is, when Timer/Counter2 starts to work, writing content to T2L is not actually written to the current register T2L, but written to the hidden register RL_T2L, and writing content to T2H is actually also it is not written into the current register T2H, but into the hidden register RL_T2H, which can cleverly realize the 16-bit reload timer. When reading the contents of T2H and T2L, the contents be read are the contents of T2H and T2L, not the contents of RL_T2H and RL_T2L.

The overflow of [T2H, T2L] not only sets the interrupt request flag (T2IF), which causes the CPU to switch to the timer 2 interrupt routine, but also automatically reloads the contents of [RL_T2H, RL_T2L] into [T2H, T2L].

13.3.6 Timer 2 calculation formula

Speed	Period calculation formula
1T	$\text{Timer period} = \frac{65536 \div T2H, T2L}{\text{SYSclk}}$ (Auto-reload)
12T	(Auto-reload)

13.4 Timer 3/4

13.4.1 Timer4 and Timer 3 Control Register (T4T3M)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
T4T3M	D1H	T4R	T4_C/T	T4x12	T4CLKO	T3R	T3_C/T	T3x12	T3CLKO

TR4: T4 run control bit.

0: T4 stops counting.

1: T4 start counting.

T4_C/T: T4 mode select bit.

0: T4 is used as a timer (input pulse is from internal system clock);

1: T4 is used as a counter (input pulse is from external T4/P0.6 pin).

T4x12: T4 speed control bit.

0: The clock source of T4 is SYSclk/12.

1: The clock source of T4 is SYSclk/1.

T4CLKO: T4 clock out control bit.

0: Turn off the clock output.

1: P0.7 is configured for T4 clock output pin. When T4 overflows, P0.7 will flip automatically.

TR3: T3 run control bit.

0: T3 stops counting.

1: T3 start counting.

T3_C/T: T3 mode select bit.

0: T3 is used as a timer (input pulse is from internal system clock);

1: T3 is used as a counter (input pulse is from external T3/P0.4 pin).

T3x12: T3 speed control bit.

0: The clock source of T3 is SYSclk/12.

1: The clock source of T3 is SYSclk/1.

T3CLKO: T3 clock out control bit.

0: Turn off the clock output.

1: P0.5 is configured for T3 clock output pin. When T3 overflows, P0.5 will flip automatically.

13.4.2 Timer 3 Counting Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
T3L	D5H								
T3H	D4H								

T3 operates in 16-bit auto-reload mode. T3L and T3H combine into a 16-bit register with T3L as the low byte and T3H as the high byte. When the 16-bit counter [T3H, T3L] overflows, the system loads the reload value in the internal 16-bit reload register into [T3H, T3L] automatically.

13.4.3 Timer 4 Counting Registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
T4L	D3H								
T4H	D2H								

T4 operates in 16-bit auto-reload mode. T4L and T4H combine into a 16-bit register with T4L as the low byte and T4H as the high byte. When the 16-bit counter [T4H, T4L] overflows, the system loads the reload value in the internal 16-bit reload register into [T4H, T4L] automatically.

13.4.4 Timer 3 8-bit Prescaler Register (TM3PS)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
TM3PS	FEA3H								

Timer 3 clock = SYSclk $\div$ (TM3PS + 1)

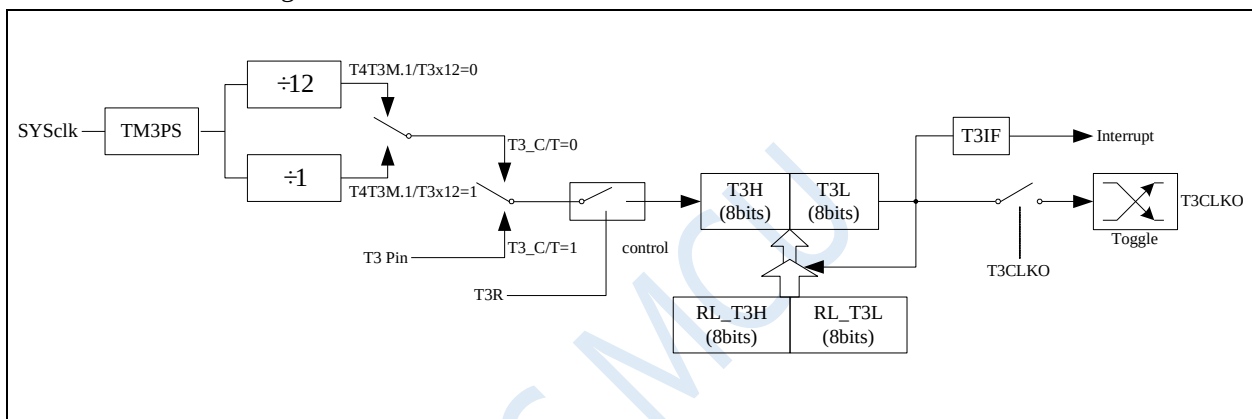
13.4.5 Timer 4 8-bit Prescaler Register (TM4PS)

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
TM4PS	FEA4H								

Timer 4 clock = SYSclk $\div$ (TM4PS + 1)

13.4.6 Timer 3 working mode

The functional block diagram of Timer/Counter 3 is as follows.



Timer/counter 3 working mode: 16-bit auto-reload mode

T3R/T4T3M.3 is the control bit in the T4T3M register. For the specific function description of each bit of the T4T3M register, see the introduction of the T4T3M register in the previous section.

When T3_C/T=0, the multiplexer is connected to the frequency division output of the system clock, T3 counts the internal system clock, and works in timing mode. When T3_C/T=1, the multiplexer is connected to the external pulse input T3, and T3 works in counting mode.

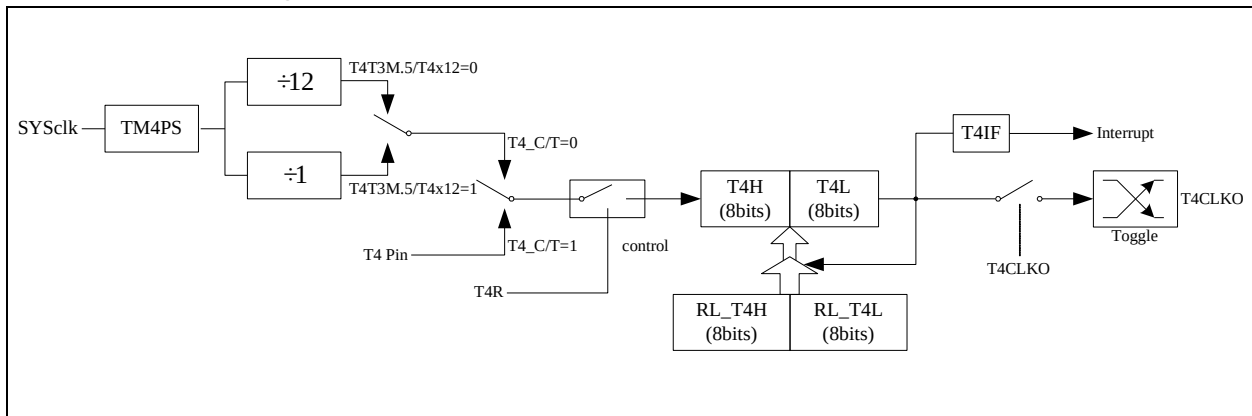
Timer3 of STC microcontroller has two counting rates: one is 12T mode, which is increased by 1 for every 12 clocks, which is the same as traditional 8051 microcontroller, the other is 1T mode, which is increased by 1 for each clock, and the speed is 12 times of traditional 8051. The rate of T3 is determined by T3x12 in the special function register T4T3M. If T3x12=0, T3 works in 12T mode, and if T3x12=1, T3 works in 1T mode.

Timer3 has two hidden registers RL_T3H and RL_T3L. RL_T3H and T3H share the same address, and RL_T3L and T3L share the same address. When T3R=0, that is, when Timer/Counter3 is disabled, the content written to T3L will be written to RL_T3L at the same time, and the content written to T3H will also be written to RL_T3H at the same time. When T3R=1, that is, when Timer/Counter3 starts to work, writing content to T3L is not actually written to the current register T3L, but written to the hidden register RL_T3L, and writing content to T3H is actually also it is not written into the current register T3H, but into the hidden register RL_T3H, which can cleverly realize the 16-bit reload timer. When reading the contents of T3H and T3L, the contents be read are the contents of T3H and T3L, not the contents of RL_T3H and RL_T3L.

The overflow of [T3H, T3L] not only sets the interrupt request flag (T3IF), which causes the CPU to switch to the timer 3 interrupt routine, but also automatically reloads the contents of [RL_T3H, RL_T3L] into [T3H, T3L].

13.4.7 Timer 4 working mode

The functional block diagram of Timer/Counter 4 is as follows.



Timer/counter 4 working mode: 16-bit auto-reload mode

T4R/T4T3M.7 is the control bit in the T4T3M register. For the specific function description of each bit of the T4T3M register, see the introduction of the T4T3M register in the previous section.

When T4_C/T=0, the multiplexer is connected to the frequency division output of the system clock, T4 counts the internal system clock, and works in timing mode. When T4_C/T=1, the multiplexer is connected to the external pulse input T4, and T4 works in counting mode.

Timer4 of STC microcontroller has two counting rates: one is 12T mode, which is increased by 1 for every 12 clocks, which is the same as traditional 8051 microcontroller, the other is 1T mode, which is increased by 1 for each clock, and the speed is 12 times of traditional 8051. The rate of T4 is determined by T4x12 in the special function register T4T3M. If T4x12=0, T4 works in 12T mode, and if T4x12=1, T4 works in 1T mode.

Timer4 has two hidden registers RL_T4H and RL_T4L. RL_T4H and T4H share the same address, and RL_T4L and T4L share the same address. When T4R=0, that is, when Timer/Counter4 is disabled, the content written to T4L will be written to RL_T4L at the same time, and the content written to T4H will also be written to RL_T4H at the same time. When T4R=1, that is, when Timer/Counter4 starts to work, writing content to T4L is not actually written to the current register T4L, but written to the hidden register RL_T4L, and writing content to T4H is actually also it is not written into the current register T4H, but into the hidden register RL_T4H, which can cleverly realize the 16-bit reload timer. When reading the contents of T4H and T4L, the contents be read are the contents of T4H and T4L, not the contents of RL_T4H and RL_T4L.

The overflow of [T4H, T4L] not only sets the interrupt request flag (T4IF), which causes the CPU to switch to the timer 4 interrupt routine, but also automatically reloads the contents of [RL_T4H, RL_T4L] into [T4H, T4L].

13.4.8 Timer 3 calculation formula

Speed of Timer	Period calculation formula
1T	$\text{Timer period} = \frac{65536 - T3H, T3L}{\text{SYSclk}}$ (Auto-reload)
12T	(Auto-reload)

13.4.9 Timer 4 calculation formula

Speed of Timer	Period calculation formula
1T	$\text{Timer period} = \frac{65536 \mid T4H, T4L}{SYSclk}$ (Auto-reload)
12T	

STC MCU

13.5 Example Routines

13.5.1 Timer 0 (Mode 0 – 16-bit auto reload)

C language code

//Operating frequency for test is 11.0592MHz

```
#include "reg51.h"
#include "intrins.h"
```

```
sfr      P0M1      = 0x93;
sfr      P0M0      = 0x94;
sfr      P1M1      = 0x91;
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;
```

```
sbit     P10       = P1^0;
```

```
void TM0_Isr() interrupt 1
```

```
{
    P10 = !P10;           //Test port
}
```

```
void main()
```

```
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    TMOD = 0x00;           //Mode 0
    TL0 = 0x66;           //65536-11.0592M/12/1000
    TH0 = 0xfc;
    TR0 = 1;              //Start timer
    ET0 = 1;              //Enable timer interrupt
    EA = 1;

    while (1);
}
```

Assembly code

;Operating frequency for test is 11.0592MHz

```

P0M1      DATA      093H
P0M0      DATA      094H
P1M1      DATA      091H
P1M0      DATA      092H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          000BH
          LJMP         TM0ISR

TM0ISR:   ORG          0100H

          CPL          P1.0          ;Test port
          RETI

MAIN:

          MOV          SP, #5FH
          MOV          P0M0, #00H
          MOV          P0M1, #00H
          MOV          P1M0, #00H
          MOV          P1M1, #00H
          MOV          P2M0, #00H
          MOV          P2M1, #00H
          MOV          P3M0, #00H
          MOV          P3M1, #00H
          MOV          P4M0, #00H
          MOV          P4M1, #00H
          MOV          P5M0, #00H
          MOV          P5M1, #00H

          MOV          TMOD, #00H    ;Mode 0
          MOV          TL0, #66H     ;65536-11.0592M/12/1000
          MOV          TH0, #0FCH
          SETB         TR0           ;Start timer
          SETB         ET0           ;Enable timer interrupt
          SETB         EA

          JMP          $

          END

```

13.5.2 Timer 0 (Mode 1 – 16-bit non-auto reload)

C language code

//Operating frequency for test is 11.0592MHz

#include "reg51.h"

```
#include "intrins.h"
```

```
sfr      P0M1      = 0x93;
sfr      P0M0      = 0x94;
sfr      P1M1      = 0x91;
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;
```

```
sbit     P10       = P1^0;
```

```
void TM0_Isr() interrupt 1
```

```
{
    TL0 = 0x66;           //Reset parameters
    TH0 = 0xfc;
    P10 = !P10;          //Test port
}
```

```
void main()
```

```
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    TMOD = 0x01;          //Mode 1
    TL0 = 0x66;           //65536-11.0592M/12/1000
    TH0 = 0xfc;
    TR0 = 1;              //Start timer
    ET0 = 1;              //Enable timer interrupt
    EA = 1;

    while (1);
}
```

Assembly code

;Operating frequency for test is 11.0592MHz

P0M1	DATA	093H
P0M0	DATA	094H
P1M1	DATA	091H
P1M0	DATA	092H
P2M1	DATA	095H
P2M0	DATA	096H
P3M1	DATA	0B1H

```
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          000BH
          LJMP         TM0ISR

          ORG          0100H
TM0ISR:
          MOV          TL0,#66H          ;Reset parameters
          MOV          TH0,#0FCH
          CPL          P1.0             ;Test port
          RETI

MAIN:
          MOV          SP,#5FH
          MOV          P0M0,#00H
          MOV          P0M1,#00H
          MOV          P1M0,#00H
          MOV          P1M1,#00H
          MOV          P2M0,#00H
          MOV          P2M1,#00H
          MOV          P3M0,#00H
          MOV          P3M1,#00H
          MOV          P4M0,#00H
          MOV          P4M1,#00H
          MOV          P5M0,#00H
          MOV          P5M1,#00H

          MOV          TMOD,#01H        ;Mode 1
          MOV          TL0,#66H        ;65536-11.0592M/12/1000
          MOV          TH0,#0FCH
          SETB         TR0             ;Start timer
          SETB         ET0             ;Enable timer interrupt
          SETB         EA

          JMP          $

          END
```

13.5.3 Timer 0 (Mode 2 - 8-bit auto reload)

C language code

//Operating frequency for test is 11.0592MHz

```
#include "reg51.h"
#include "intrins.h"
```

```
sfr      P0M1      = 0x93;
sfr      P0M0      = 0x94;
sfr      P1M1      = 0x91;
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
```

```
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;
```

```
sbit     P10       = P1^0;
```

```
void TM0_Isr() interrupt 1
```

```
{
    P10 = !P10;                      //Test port
}
```

```
void main()
```

```
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    TMOD = 0x02;                      //Mode 2
    TL0 = 0xf4;                       //256-11.0592M/12/76K
    TH0 = 0xf4;
    TR0 = 1;                          //Start timer
    ET0 = 1;                          //Enable timer interrupt
    EA = 1;

    while (1);
}
```

Assembly code

;Operating frequency for test is 11.0592MHz

```
P0M1      DATA      093H
P0M0      DATA      094H
P1M1      DATA      091H
P1M0      DATA      092H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          000BH
```

```

        LJMP      TM0ISR

        ORG      0100H

TM0ISR:
        CPL      P1.0          ;Test port
        RETI

MAIN:
        MOV      SP, #5FH
        MOV      P0M0, #00H
        MOV      P0M1, #00H
        MOV      P1M0, #00H
        MOV      P1M1, #00H
        MOV      P2M0, #00H
        MOV      P2M1, #00H
        MOV      P3M0, #00H
        MOV      P3M1, #00H
        MOV      P4M0, #00H
        MOV      P4M1, #00H
        MOV      P5M0, #00H
        MOV      P5M1, #00H

        MOV      TMOD, #02H    ;Mode 2
        MOV      TL0, #0F4H    ;256-11.0592M/12/76K
        MOV      TH0, #0F4H
        SETB     TR0           ;Start timer
        SETB     ET0           ;Enable timer interrupt
        SETB     EA

        JMP      $

        END

```

13.5.4 Timer 0 (Mode 3 - 16-bit auto reload with non-maskable interrupt)

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

sfr      P0M1      = 0x93;
sfr      P0M0      = 0x94;
sfr      P1M1      = 0x91;
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;

```



```

sbit      P10      = P1^0;

void TM0_Isr() interrupt 1
{
    P10 = !P10;                //Test port
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    TMOD = 0x03;                //Mode 3
    TL0 = 0x66;                 //65536-11.0592M/12/1000
    TH0 = 0xfc;
    TR0 = 1;                    //Start timer
    ET0 = 1;                    //Enable timer interrupt
    // EA = 1;                  // Not controlled by EA

    while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

P0M1      DATA      093H
P0M0      DATA      094H
P1M1      DATA      091H
P1M0      DATA      092H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

            ORG        0000H
            LJMP       MAIN
            ORG        000BH
            LJMP       TM0ISR

            ORG        0100H
TM0ISR:
            CPL        P1.0        ;Test port
            RETI

```

MAIN:

```

MOV      SP, #5FH
MOV      P0M0, #00H
MOV      P0M1, #00H
MOV      P1M0, #00H
MOV      P1M1, #00H
MOV      P2M0, #00H
MOV      P2M1, #00H
MOV      P3M0, #00H
MOV      P3M1, #00H
MOV      P4M0, #00H
MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

MOV      TMOD, #03H           ;Mode 3
MOV      TL0, #66H           ;65536-11.0592M/12/1000
MOV      TH0, #0FCH
SETB     TR0                 ;Start timer
SETB     ET0                 ;Enable timer interrupt
; SETB     EA                 ; Not controlled by EA

JMP      $

END

```

13.5.5 Timer 0 (External count - T0 is extended for external falling edge interrupt)

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

sfr      P0M1      = 0x93;
sfr      P0M0      = 0x94;
sfr      P1M1      = 0x91;
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;

```

```

sbit     P10       = P1^0;

```

```

void TM0_Isr() interrupt 1

```

```

{
    P10 = !P10;           //Test port
}

```

```

void main()

```

```

{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    TMOD = 0x04;                //External counting mode
    TL0 = 0xff;
    TH0 = 0xff;
    TR0 = 1;                    //Start timer
    ET0 = 1;                    //Enable timer interrupt
    EA = 1;

    while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

P0M1      DATA      093H
P0M0      DATA      094H
P1M1      DATA      091H
P1M0      DATA      092H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

            ORG        0000H
            LJMP       MAIN
            ORG        000BH
            LJMP       TM0ISR

            ORG        0100H
TM0ISR:
            CPL        P1.0                ;Test port
            RETI

MAIN:
            MOV        SP, #5FH
            MOV        P0M0, #00H
            MOV        P0M1, #00H
            MOV        P1M0, #00H
            MOV        P1M1, #00H
            MOV        P2M0, #00H
            MOV        P2M1, #00H
            MOV        P3M0, #00H

```

```

MOV      P3M1, #00H
MOV      P4M0, #00H
MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

MOV      TMOD, #04H           ;External counting mode
MOV      TL0, #0FFH
MOV      TH0, #0FFH
SETB     TR0                 ;Start timer
SETB     ET0                 ;Enable timer interrupt
SETB     EA

JMP      $

END

```

13.5.6 Timer 0 (Pulse width measurement for high-level width of INT0)

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

sfr      AUXR      = 0x8e;

sfr      P0M1      = 0x93;
sfr      P0M0      = 0x94;
sfr      P1M1      = 0x91;
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;

```

```

void INT0_Isr() interrupt 0
{
    P0 = TL0;
    P1 = TH0;
}

```

//TL0 is the low byte of the measured value
//TH0 is the high byte of the measured value

```

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
}

```

```

P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

AUXR = 0x80;
TMOD = 0x08;
TL0 = 0x00;
TH0 = 0x00;
while (INT0);
TR0 = 1;
IT0 = 1;
EX0 = 1;
EA = 1;

while (1);
}

```

```

//1T mode
//Enable GATE, and enable timing when INT0 is 1

//Wait for INT0 to be low
//Start timer
//Enable INT0 falling edge interrupt

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

AUXR      DATA      8EH
P0M1      DATA      093H
P0M0      DATA      094H
P1M1      DATA      091H
P1M0      DATA      092H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

```

```

ORG      0000H
LJMP     MAIN
ORG      0003H
LJMP     INT0ISR

```

```

INT0ISR:  ORG      0100H

```

```

MOV      P0,TL0
MOV      P1,TH0
RETI

```

```

;TL0 is the low byte of the measured value
;TH0 is the high byte of the measured value

```

MAIN:

```

MOV      SP, #5FH
MOV      P0M0, #00H
MOV      P0M1, #00H
MOV      P1M0, #00H
MOV      P1M1, #00H
MOV      P2M0, #00H
MOV      P2M1, #00H
MOV      P3M0, #00H
MOV      P3M1, #00H
MOV      P4M0, #00H

```

```

MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

MOV      AUXR, #80H           ;1T mode
MOV      TMOD, #08H          ;Enable GATE, and enable timing when INT0 is 1
MOV      TL0, #00H
MOV      TH0, #00H
JB       INT0, $              ;Wait for INT0 to be low
SETB     TR0                  ;Start timer
SETB     IT0                  ;Enable INT0 falling edge interrupt
SETB     EX0
SETB     EA

JMP      $

END

```

13.5.7 Timer 0 (Mode 0, Divided clock output)

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

sfr      INTCLKO    = 0x8f;

sfr      P0M1       = 0x93;
sfr      P0M0       = 0x94;
sfr      P1M1       = 0x91;
sfr      P1M0       = 0x92;
sfr      P2M1       = 0x95;
sfr      P2M0       = 0x96;
sfr      P3M1       = 0xb1;
sfr      P3M0       = 0xb2;
sfr      P4M1       = 0xb3;
sfr      P4M0       = 0xb4;
sfr      P5M1       = 0xc9;
sfr      P5M0       = 0xca;

```

```

void main()
{

```

```

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

```

```

    TMOD = 0x00;

```

```

    //Mode 0

```

```

    TL0 = 0x66;                //65536-11.0592M/12/1000
    TH0 = 0xfc;
    TR0 = 1;                   //Start timer
    INTCLKO = 0x01;            //Enable clock output

    while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

INTCLKO    DATA    8FH
P0M1       DATA    093H
P0M0       DATA    094H
P1M1       DATA    091H
P1M0       DATA    092H
P2M1       DATA    095H
P2M0       DATA    096H
P3M1       DATA    0B1H
P3M0       DATA    0B2H
P4M1       DATA    0B3H
P4M0       DATA    0B4H
P5M1       DATA    0C9H
P5M0       DATA    0CAH

                ORG    0000H
                LJMP   MAIN

MAIN:         ORG    0100H

                MOV     SP, #5FH
                MOV     P0M0, #00H
                MOV     P0M1, #00H
                MOV     P1M0, #00H
                MOV     P1M1, #00H
                MOV     P2M0, #00H
                MOV     P2M1, #00H
                MOV     P3M0, #00H
                MOV     P3M1, #00H
                MOV     P4M0, #00H
                MOV     P4M1, #00H
                MOV     P5M0, #00H
                MOV     P5M1, #00H

                MOV     TMOD, #00H    ;Mode 0
                MOV     TL0, #66H     ;65536-11.0592M/12/1000
                MOV     TH0, #0FCH
                SETB    TR0           ;Start timer
                MOV     INTCLKO, #01H ;Enable clock output

                JMP     $

                END

```

13.5.8 Timer 1 (Mode 0 - 16-bit auto reload)

C language code

//Operating frequency for test is 11.0592MHz

#include "reg51.h"

#include "intrins.h"

```
sfr      P0M1      = 0x93;
sfr      P0M0      = 0x94;
sfr      P1M1      = 0x91;
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;
```

```
sbit     P10       = P1^0;
```

void TM1_Isr() interrupt 3

```
{
    P10 = !P10;                                //Test port
}
```

void main()

```
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    TMOD = 0x00;                                //Mode 0
    TL1 = 0x66;                                //65536-11.0592M/12/1000
    TH1 = 0xfc;
    TR1 = 1;                                    //Start timer
    ET1 = 1;                                    //Enable timer interrupt
    EA = 1;

    while (1);
}
```

Assembly code

;Operating frequency for test is 11.0592MHz

<i>P0M1</i>	<i>DATA</i>	<i>093H</i>
<i>P0M0</i>	<i>DATA</i>	<i>094H</i>
<i>P1M1</i>	<i>DATA</i>	<i>091H</i>
<i>P1M0</i>	<i>DATA</i>	<i>092H</i>
<i>P2M1</i>	<i>DATA</i>	<i>095H</i>
<i>P2M0</i>	<i>DATA</i>	<i>096H</i>


```

P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          001BH
          LJMP         TM1ISR

          ORG          0100H
TM1ISR:
          CPL          P1.0          ;Test port
          RETI

MAIN:
          MOV          SP, #5FH
          MOV          P0M0, #00H
          MOV          P0M1, #00H
          MOV          P1M0, #00H
          MOV          P1M1, #00H
          MOV          P2M0, #00H
          MOV          P2M1, #00H
          MOV          P3M0, #00H
          MOV          P3M1, #00H
          MOV          P4M0, #00H
          MOV          P4M1, #00H
          MOV          P5M0, #00H
          MOV          P5M1, #00H

          MOV          TMOD, #00H      ;Mode 0
          MOV          TL1, #66H      ;65536-11.0592M/12/1000
          MOV          TH1, #0FCH
          SETB         TR1            ;Start timer
          SETB         ET1            ;Enable timer interrupt
          SETB         EA

          JMP          $

          END

```

13.5.9 Timer 1 (Mode 1 - 16-bit non-auto reload)

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

sfr      P0M1      =    0x93;
sfr      P0M0      =    0x94;
sfr      P1M1      =    0x91;
sfr      P1M0      =    0x92;
sfr      P2M1      =    0x95;
sfr      P2M0      =    0x96;

```

```
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;

sbit     P10        = P1^0;

void TM1_Isr() interrupt 3
{
    TL1 = 0x66;                //Reset parameters
    TH1 = 0xfc;
    P10 = !P10;                //Test port
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    TMOD = 0x10;                //Mode 1
    TL1 = 0x66;                //65536-11.0592M/12/1000
    TH1 = 0xfc;
    TR1 = 1;                    //Start timer
    ET1 = 1;                    //Enable timer interrupt
    EA = 1;

    while (1);
}
```

Assembly code

;Operating frequency for test is 11.0592MHz

```
P0M1      DATA      093H
P0M0      DATA      094H
P1M1      DATA      091H
P1M0      DATA      092H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
```

```

        ORG      001BH
        LJMP     TM1ISR

TM1ISR:  ORG      0100H

        MOV      TL1,#66H           ;Reset parameters
        MOV      TH1,#0FCH
        CPL      P1.0              ;Test port
        RETI

MAIN:

        MOV      SP,#5FH
        MOV      P0M0,#00H
        MOV      P0M1,#00H
        MOV      P1M0,#00H
        MOV      P1M1,#00H
        MOV      P2M0,#00H
        MOV      P2M1,#00H
        MOV      P3M0,#00H
        MOV      P3M1,#00H
        MOV      P4M0,#00H
        MOV      P4M1,#00H
        MOV      P5M0,#00H
        MOV      P5M1,#00H

        MOV      TMOD,#10H         ;Mode 1
        MOV      TL1,#66H         ;65536-11.0592M/12/1000
        MOV      TH1,#0FCH
        SETB     TR1              ;Start timer
        SETB     ET1              ;Enable timer interrupt
        SETB     EA

        JMP      $

END

```

13.5.10 Timer 1 (Mode 2 - 8-bit auto reload)

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

sfr      P0M1      = 0x93;
sfr      P0M0      = 0x94;
sfr      P1M1      = 0x91;
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;

```

```

sbit      P10      =    P1^0;

void TM1_Isr() interrupt 3
{
    P10 = !P10;                      //Test port
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    TMOD = 0x20;                      //Mode 2
    TL1 = 0xf4;                       //256-11.0592M/12/76K
    TH1 = 0xf4;
    TR1 = 1;                          //Start timer
    ET1 = 1;                          //Enable timer interrupt
    EA = 1;

    while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

P0M1      DATA      093H
P0M0      DATA      094H
P1M1      DATA      091H
P1M0      DATA      092H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

            ORG        0000H
            LJMP       MAIN
            ORG        001BH
            LJMP       TM1ISR

            ORG        0100H
TM1ISR:
            CPL        P1.0           ;Test port
            RETI

```

MAIN:

```

MOV      SP, #5FH
MOV      P0M0, #00H
MOV      P0M1, #00H
MOV      P1M0, #00H
MOV      P1M1, #00H
MOV      P2M0, #00H
MOV      P2M1, #00H
MOV      P3M0, #00H
MOV      P3M1, #00H
MOV      P4M0, #00H
MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

MOV      TMOD, #20H           ;Mode 2
MOV      TL1, #0F4H           ;256-11.0592M/12/76K
MOV      TH1, #0F4H
SETB     TR1                   ;Start timer
SETB     ET1                   ;Enable timer interrupt
SETB     EA

JMP      $

END

```

13.5.11 Timer 1 (External count – T1 is extended for external falling edge interrupt)

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

sfr      P0M1      = 0x93;
sfr      P0M0      = 0x94;
sfr      P1M1      = 0x91;
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;

```

```

sbit     P10       = P1^0;

```

```

void TM1_Isr() interrupt 3

```

```

{
    P10 = !P10;           //Test port
}

```

```

void main()

```

```

{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    TMOD = 0x40;                //External counting mode
    TL1 = 0xff;
    TH1 = 0xff;
    TR1 = 1;                    //Start timer
    ET1 = 1;                    //Enable timer interrupt
    EA = 1;

    while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

P0M1      DATA      093H
P0M0      DATA      094H
P1M1      DATA      091H
P1M0      DATA      092H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

            ORG        0000H
            LJMP       MAIN
            ORG        001BH
            LJMP       TM1ISR

            ORG        0100H
TM1ISR:
            CPL        P1.0                ;Test port
            RETI

MAIN:
            MOV        SP, #5FH
            MOV        P0M0, #00H
            MOV        P0M1, #00H
            MOV        P1M0, #00H
            MOV        P1M1, #00H
            MOV        P2M0, #00H
            MOV        P2M1, #00H
            MOV        P3M0, #00H

```

```

MOV      P3M1, #00H
MOV      P4M0, #00H
MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

MOV      TMOD, #40H           ;External counting mode
MOV      TL1, #0FFH
MOV      TH1, #0FFH
SETB     TR1                 ;Start timer
SETB     ET1                 ;Enable timer interrupt
SETB     EA

JMP      $

END

```

13.5.12 Timer 1 (Pulse width measurement for high-level width of INT1)

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

sfr      P0M1      = 0x93;
sfr      P0M0      = 0x94;
sfr      P1M1      = 0x91;
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;

sfr      AUXR      = 0x8e;

```

```

void INT1_Isr() interrupt 2
{
    P0 = TL1;
    P1 = TH1;
}

```

//TL1 is the low byte of the measured value
//TH1 is the high byte of the measured value

```

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
}

```

```

P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

AUXR = 0x40;
TMOD = 0x80;
TL1 = 0x00;
TH1 = 0x00;
while (INT1);
TR1 = 1;
IT1 = 1;
EX1 = 1;
EA = 1;

while (1);
}

```

```

//1T mode
//Enable GATE, and enable timing when INT1 is 1

//Wait for INT1 to be low
//Start timer
//Enable INT1 falling edge interrupt

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

AUXR      DATA      8EH
P0M1      DATA      093H
P0M0      DATA      094H
P1M1      DATA      091H
P1M0      DATA      092H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

```

```

ORG      0000H
LJMP     MAIN
ORG      0013H
LJMP     INT1ISR

```

```

INT1ISR:  ORG      0100H

```

```

MOV      P0,TL1
MOV      P1,TH1
RET

```

```

;TL1 is the low byte of the measured value
;TH1 is the high byte of the measured value

```

MAIN:

```

MOV      SP,#5FH
MOV      P0M0,#00H
MOV      P0M1,#00H
MOV      P1M0,#00H
MOV      P1M1,#00H
MOV      P2M0,#00H
MOV      P2M1,#00H
MOV      P3M0,#00H
MOV      P3M1,#00H
MOV      P4M0,#00H

```



```

MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

MOV      AUXR, #40H           ;1T mode
MOV      TMOD, #80H          ;Enable GATE, and enable timing when INT1 is 1
MOV      TL1, #00H
MOV      TH1, #00H
JB       INT1, $              ;Wait for INT1 to be low
SETB     TR1                  ;Start timer
SETB     IT1                  ;Enable INT1 falling edge interrupt
SETB     EX1
SETB     EA

JMP      $

END

```

13.5.13 Timer 1 (Mode 0, Divided clock output)

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

sfr      INTCLKO    =    0x8f;

sfr      P0M1       =    0x93;
sfr      P0M0       =    0x94;
sfr      P1M1       =    0x91;
sfr      P1M0       =    0x92;
sfr      P2M1       =    0x95;
sfr      P2M0       =    0x96;
sfr      P3M1       =    0xb1;
sfr      P3M0       =    0xb2;
sfr      P4M1       =    0xb3;
sfr      P4M0       =    0xb4;
sfr      P5M1       =    0xc9;
sfr      P5M0       =    0xca;

```

```

void main()
{

```

```

    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

```

```

    TMOD = 0x00;           //Mode 0

```

```

    TL1 = 0x66;                      //65536-11.0592M/12/1000
    TH1 = 0xfc;
    TR1 = 1;                          //Start timer
    INTCLKO = 0x02;                   //Enable clock output

    while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

INTCLKO    DATA    8FH
P0M1       DATA    093H
P0M0       DATA    094H
P1M1       DATA    091H
P1M0       DATA    092H
P2M1       DATA    095H
P2M0       DATA    096H
P3M1       DATA    0B1H
P3M0       DATA    0B2H
P4M1       DATA    0B3H
P4M0       DATA    0B4H
P5M1       DATA    0C9H
P5M0       DATA    0CAH

                ORG    0000H
                LJMP   MAIN

MAIN:         ORG    0100H

                MOV    SP, #5FH
                MOV    P0M0, #00H
                MOV    P0M1, #00H
                MOV    P1M0, #00H
                MOV    P1M1, #00H
                MOV    P2M0, #00H
                MOV    P2M1, #00H
                MOV    P3M0, #00H
                MOV    P3M1, #00H
                MOV    P4M0, #00H
                MOV    P4M1, #00H
                MOV    P5M0, #00H
                MOV    P5M1, #00H

                MOV    TMOD, #00H           ;Mode 0
                MOV    TL1, #66H           ;65536-11.0592M/12/1000
                MOV    TH1, #0FCH
                SETB   TR1                  ;Start timer
                MOV    INTCLKO, #02H       ;Enable clock output

                JMP    $

                END

```

13.5.14 Timer 1 (Mode 0) is used as baud rate generator of UART1

C language code

//Operating frequency for test is 11.0592MHz

```
#include "reg51.h"
#include "intrins.h"
```

```
#define FOSC      11059200UL
#define BRT      (65536 - FOSC / 115200 / 4)
```

```
sfr      AUXR      = 0x8e;
```

```
sfr      P0M1      = 0x93;
```

```
sfr      P0M0      = 0x94;
```

```
sfr      P1M1      = 0x91;
```

```
sfr      P1M0      = 0x92;
```

```
sfr      P2M1      = 0x95;
```

```
sfr      P2M0      = 0x96;
```

```
sfr      P3M1      = 0xb1;
```

```
sfr      P3M0      = 0xb2;
```

```
sfr      P4M1      = 0xb3;
```

```
sfr      P4M0      = 0xb4;
```

```
sfr      P5M1      = 0xc9;
```

```
sfr      P5M0      = 0xca;
```

```
bit      busy;
```

```
char      wptr;
```

```
char      rptr;
```

```
char      buffer[16];
```

```
void UartIsr() interrupt 4
```

```
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
        buffer[wptr++] = SBUF;
        wptr &= 0x0f;
    }
}
```

```
void UartInit()
```

```
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}
```

```
void UartSend(char dat)
```

```
{
    while (busy);
```

```
    busy = 1;
    SBUF = dat;
}

void UartSendStr(char *p)
{
    while (*p)
    {
        UartSEND(*p++);
    }
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    UartInit();
    ES = 1;
    EA = 1;
    UartSENDStr("Uart Test !\r\n");

    while (1)
    {
        if (rptr != wptr)
        {
            UartSEND(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}
```

Assembly code

;Operating frequency for test is 11.0592MHz

AUXR	DATA	8EH	
BUSY	BIT	20H.0	
WPTR	DATA	21H	
RPTR	DATA	22H	
BUFFER	DATA	23H	;16 bytes
P0M1	DATA	093H	
P0M0	DATA	094H	
P1M1	DATA	091H	
P1M0	DATA	092H	
P2M1	DATA	095H	
P2M0	DATA	096H	
P3M1	DATA	0B1H	

```

P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          0023H
          LJMP         UART_ISR

          ORG          0100H

UART_ISR:
          PUSH         ACC
          PUSH         PSW
          MOV          PSW,#08H

          JNB          TI,CHKRI
          CLR          TI
          CLR          BUSY

CHKRI:
          JNB          RI,UARTISR_EXIT
          CLR          RI
          MOV          A,WPTR
          ANL          A,#0FH
          ADD          A,#BUFFER
          MOV          R0,A
          MOV          @R0,SBUF
          INC          WPTR

UARTISR_EXIT:
          POP          PSW
          POP          ACC
          RETI

UART_INIT:
          MOV          SCON,#50H
          MOV          TMOD,#00H
          MOV          TL1,#0E8H
          MOV          TH1,#0FFH
          SETB         TR1
          MOV          AUXR,#40H
          CLR          BUSY
          MOV          WPTR,#00H
          MOV          RPTR,#00H
          RET

UART_SEND:
          JB           BUSY,$
          SETB         BUSY
          MOV          SBUF,A
          RET

UART_SENDSTR:
          CLR          A
          MOVC         A,@A+DPTR
          JZ           SENDEND
          LCALL        UART_SEND
          INC          DPTR

```

;65536-11059200/115200/4=0FFE8H

```
                JMP          UART_SENDSTR
SENDEND:
                RET

MAIN:
                MOV          SP, #5FH
                MOV          P0M0, #00H
                MOV          P0M1, #00H
                MOV          P1M0, #00H
                MOV          P1M1, #00H
                MOV          P2M0, #00H
                MOV          P2M1, #00H
                MOV          P3M0, #00H
                MOV          P3M1, #00H
                MOV          P4M0, #00H
                MOV          P4M1, #00H
                MOV          P5M0, #00H
                MOV          P5M1, #00H

                LCALL        UART_INIT
                SETB         ES
                SETB         EA

                MOV          DPTR, #STRING
                LCALL        UART_SENDSTR

LOOP:
                MOV          A, RPTR
                XRL          A, WPTR
                ANL          A, #0FH
                JZ           LOOP
                MOV          A, RPTR
                ANL          A, #0FH
                ADD          A, #BUFFER
                MOV          R0, A
                MOV          A, @R0
                LCALL        UART_SEND
                INC          RPTR
                JMP          LOOP

STRING:         DB          'Uart Test !', 0DH, 0AH, 00H

                END
```

13.5.15 Timer 1 (Mode 2) is used as baud rate generator of UART1

C language code

//Operating frequency for test is 11.0592MHz

```
#include "reg51.h"
#include "intrins.h"
```

```
#define FOSC          11059200UL
#define BRT           (256 - FOSC / 115200 / 32)
```

```
sfr      AUXR        =    0x8e;
```

```
sfr      P0M1      = 0x93;
sfr      P0M0      = 0x94;
sfr      P1M1      = 0x91;
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;
```

```
bit      busy;
char     wptr;
char     rptr;
char     buffer[16];
```

```
void UartIsr() interrupt 4
```

```
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
        buffer[wptr++] = SBUF;
        wptr &= 0x0f;
    }
}
```

```
void UartInit()
```

```
{
    SCON = 0x50;
    TMOD = 0x20;
    TL1 = BRT;
    TH1 = BRT;
    TR1 = 1;
    AUXR = 0x40;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}
```

```
void UartSend(char dat)
```

```
{
    while (busy);
    busy = 1;
    SBUF = dat;
}
```

```
void UartSendStr(char *p)
```

```
{
    while (*p)
    {
        UartSEND(*p++);
    }
}
```

```

}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    UartInit();
    ES = 1;
    EA = 1;
    UartSENDStr("Uart Test !\r\n");

    while (1)
    {
        if (rptr != wptr)
        {
            UartSEND(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

AUXR	DATA	8EH	
BUSY	BIT	20H.0	
WPTR	DATA	21H	
RPTR	DATA	22H	
BUFFER	DATA	23H	;16 bytes
P0M1	DATA	093H	
P0M0	DATA	094H	
P1M1	DATA	091H	
P1M0	DATA	092H	
P2M1	DATA	095H	
P2M0	DATA	096H	
P3M1	DATA	0B1H	
P3M0	DATA	0B2H	
P4M1	DATA	0B3H	
P4M0	DATA	0B4H	
P5M1	DATA	0C9H	
P5M0	DATA	0CAH	
	ORG	0000H	
	LJMP	MAIN	
	ORG	0023H	
	LJMP	UART_ISR	


```

        ORG        0100H

UART_ISR:
        PUSH      ACC
        PUSH      PSW
        MOV       PSW,#08H

        JNB       TI,CHKRI
        CLR       TI
        CLR       BUSY

CHKRI:
        JNB       RI,UARTISR_EXIT
        CLR       RI
        MOV       A,WPTR
        ANL       A,#0FH
        ADD       A,#BUFFER
        MOV       R0,A
        MOV       @R0,SBUF
        INC       WPTR

UARTISR_EXIT:
        POP       PSW
        POP       ACC
        RETI

UART_INIT:
        MOV       SCON,#50H
        MOV       TMOD,#20H
        MOV       TL1,#0FDH      ;256-11059200/115200/32=0FDH
        MOV       TH1,#0FDH
        SETB      TR1
        MOV       AUXR,#40H
        CLR       BUSY
        MOV       WPTR,#00H
        MOV       RPTR,#00H
        RET

UART_SEND:
        JB        BUSY,$
        SETB      BUSY
        MOV       SBUF,A
        RET

UART_SENDSTR:
        CLR       A
        MOVC      A,@A+DPTR
        JZ        SENDEND
        LCALL     UART_SEND
        INC       DPTR
        JMP       UART_SENDSTR

SENDEND:
        RET

MAIN:
        MOV       SP,#5FH
        MOV       P0M0,#00H
        MOV       P0M1,#00H
        MOV       P1M0,#00H
        MOV       P1M1,#00H

```

```
MOV      P2M0, #00H
MOV      P2M1, #00H
MOV      P3M0, #00H
MOV      P3M1, #00H
MOV      P4M0, #00H
MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

LCALL    UART_INIT
SETB     ES
SETB     EA

MOV      DPTR, #STRING
LCALL    UART_SENDSTR
```

LOOP:

```
MOV      A, RPTR
XRL      A, WPTR
ANL      A, #0FH
JZ       LOOP
MOV      A, RPTR
ANL      A, #0FH
ADD      A, #BUFFER
MOV      R0, A
MOV      A, @R0
LCALL    UART_SEND
INC      RPTR
JMP      LOOP
```

```
STRING:  DB      'Uart Test !', 0DH, 0AH, 00H

END
```

13.5.16 Timer 2 (16-bit auto reload)

C language code

//Operating frequency for test is 11.0592MHz

```
#include "reg51.h"
#include "intrins.h"
```

```
sfr      T2L      = 0xd7;
sfr      T2H      = 0xd6;
sfr      AUXR     = 0x8e;
sfr      IE2      = 0xaf;
#define    ET2      0x04
sfr      AUXINTIF  = 0xef;
#define    T2IF     0x01

sfr      P0M1     = 0x93;
sfr      P0M0     = 0x94;
sfr      P1M1     = 0x91;
sfr      P1M0     = 0x92;
sfr      P2M1     = 0x95;
sfr      P2M0     = 0x96;
sfr      P3M1     = 0xb1;
```

```

sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;

```

```

sbit     P10       = P1^0;

```

```

void TM2_Isr() interrupt 12

```

```

{
    P10 = !P10;                      //Test port
}

```

```

void main()

```

```

{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    T2L = 0x66;                      //65536-11.0592M/12/1000
    T2H = 0xfc;
    AUXR = 0x10;                     //Start timer
    IE2 = ET2;                       //Enable timer interrupt
    EA = 1;

    while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

T2L      DATA    0D7H
T2H      DATA    0D6H
AUXR     DATA    8EH
IE2      DATA    0AFH
ET2      EQU      04H
AUXINTIF DATA    0EFH
T2IF     EQU      01H

```

```

P0M1     DATA    093H
P0M0     DATA    094H
P1M1     DATA    091H
P1M0     DATA    092H
P2M1     DATA    095H
P2M0     DATA    096H
P3M1     DATA    0B1H
P3M0     DATA    0B2H
P4M1     DATA    0B3H
P4M0     DATA    0B4H
P5M1     DATA    0C9H

```

```
P5M0      DATA      0CAH

           ORG         0000H
           LJMP        MAIN
           ORG         0063H
           LJMP        TM2ISR

           ORG         0100H
TM2ISR:    CPL         P1.0           ;Test port
           RETI

MAIN:
           MOV         SP, #5FH
           MOV         P0M0, #00H
           MOV         P0M1, #00H
           MOV         P1M0, #00H
           MOV         P1M1, #00H
           MOV         P2M0, #00H
           MOV         P2M1, #00H
           MOV         P3M0, #00H
           MOV         P3M1, #00H
           MOV         P4M0, #00H
           MOV         P4M1, #00H
           MOV         P5M0, #00H
           MOV         P5M1, #00H

           MOV         T2L, #66H      ;65536-11.0592M/12/1000
           MOV         T2H, #0FCH
           MOV         AUXR, #10H     ;Start timer
           MOV         IE2, #ET2      ;Enable timer interrupt
           SETB        EA

           JMP         $

           END
```

13.5.17 Timer 2 (External count – T2 is extended for external falling edge interrupt)

C language code

//Operating frequency for test is 11.0592MHz

```
#include "reg51.h"
#include "intrins.h"
```

```
sfr      T2L      = 0xd7;
sfr      T2H      = 0xd6;
sfr      AUXR     = 0x8e;
sfr      IE2      = 0xaf;
#define   ET2      0x04
sfr      AUXINTIF  = 0xef;
#define   T2IF     0x01

sfr      P0M1     = 0x93;
```

```
sfr      P0M0      = 0x94;
sfr      P1M1      = 0x91;
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;

sbit     P10       = P1^0;
```

```
void TM2_Isr() interrupt 12
```

```
{
    P10 = !P10;                //Test port
}
```

```
void main()
```

```
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    T2L = 0xff;
    T2H = 0xff;
    AUXR = 0x18;                //Set external counting mode and start timer
    IE2 = ET2;                  //Enable timer interrupt
    EA = 1;

    while (1);
}
```

Assembly code

;Operating frequency for test is 11.0592MHz

T2L	DATA	0D7H
T2H	DATA	0D6H
AUXR	DATA	8EH
IE2	DATA	0AFH
ET2	EQU	04H
AUXINTIF	DATA	0EFH
T2IF	EQU	01H
P0M1	DATA	093H
P0M0	DATA	094H
P1M1	DATA	091H
P1M0	DATA	092H
P2M1	DATA	095H

```
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          0063H
          LJMP         TM2ISR

TM2ISR:   ORG          0100H

          CPL          P1.0          ;Test port
          RETI

MAIN:

          MOV          SP, #5FH
          MOV          P0M0, #00H
          MOV          P0M1, #00H
          MOV          P1M0, #00H
          MOV          P1M1, #00H
          MOV          P2M0, #00H
          MOV          P2M1, #00H
          MOV          P3M0, #00H
          MOV          P3M1, #00H
          MOV          P4M0, #00H
          MOV          P4M1, #00H
          MOV          P5M0, #00H
          MOV          P5M1, #00H

          MOV          T2L, #0FFH
          MOV          T2H, #0FFH
          MOV          AUXR, #18H    ;Set external counting mode and start timer
          MOV          IE2, #ET2    ;Enable timer interrupt
          SETB         EA

          JMP          $

          END
```

13.5.18 Timer 2 (Divided clock output)

C language code

//Operating frequency for test is 11.0592MHz

```
#include "reg51.h"
#include "intrins.h"
```

```
sfr      T2L      = 0xd7;
sfr      T2H      = 0xd6;
sfr      AUXR     = 0x8e;
sfr      INTCLKO  = 0x8f;

sfr      P0M1     = 0x93;
```

```
sfr      P0M0      = 0x94;
sfr      P1M1      = 0x91;
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;
```

```
void main()
```

```
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    T2L = 0x66;           //65536-11.0592M/12/1000
    T2H = 0xfc;
    AUXR = 0x10;          //Start timer
    INTCLKO = 0x04;       //Enable clock output

    while (1);
}
```

Assembly code

;Operating frequency for test is 11.0592MHz

T2L	DATA	0D7H
T2H	DATA	0D6H
AUXR	DATA	8EH
INTCLKO	DATA	8FH
P0M1	DATA	093H
P0M0	DATA	094H
P1M1	DATA	091H
P1M0	DATA	092H
P2M1	DATA	095H
P2M0	DATA	096H
P3M1	DATA	0B1H
P3M0	DATA	0B2H
P4M1	DATA	0B3H
P4M0	DATA	0B4H
P5M1	DATA	0C9H
P5M0	DATA	0CAH
	ORG	0000H
	LJMP	MAIN

```

ORG          0100H

MAIN:

  MOV          SP, #5FH
  MOV          P0M0, #00H
  MOV          P0M1, #00H
  MOV          P1M0, #00H
  MOV          P1M1, #00H
  MOV          P2M0, #00H
  MOV          P2M1, #00H
  MOV          P3M0, #00H
  MOV          P3M1, #00H
  MOV          P4M0, #00H
  MOV          P4M1, #00H
  MOV          P5M0, #00H
  MOV          P5M1, #00H

  MOV          T2L, #66H                ;65536-11.0592M/12/1000
  MOV          T2H, #0FCH
  MOV          AUXR, #10H                ;Start timer
  MOV          INTCLKO, #04H            ;Enable clock output

  JMP          $

END

```

13.5.19 Timer 2 is used as baud rate generator of UART1

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

#define FOSC      11059200UL
#define BRT       (65536 - FOSC / 115200 / 4)

```

```

sfr  AUXR      = 0x8e;
sfr  T2H       = 0xd6;
sfr  T2L       = 0xd7;

```

```

sfr  P0M1      = 0x93;
sfr  P0M0      = 0x94;
sfr  P1M1      = 0x91;
sfr  P1M0      = 0x92;
sfr  P2M1      = 0x95;
sfr  P2M0      = 0x96;
sfr  P3M1      = 0xb1;
sfr  P3M0      = 0xb2;
sfr  P4M1      = 0xb3;
sfr  P4M0      = 0xb4;
sfr  P5M1      = 0xc9;
sfr  P5M0      = 0xca;

```

```

bit   busy;
char  wptr;
char  rptr;
char  buffer[16];

```



```
void UartIsr() interrupt 4
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
        buffer[wptr++] = SBUF;
        wptr &= 0x0f;
    }
}

void UartInit()
{
    SCON = 0x50;
    T2L = BRT;
    T2H = BRT >> 8;
    AUXR = 0x15;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}

void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}

void UartSendStr(char *p)
{
    while (*p)
    {
        UartSEND(*p++);
    }
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    UartInit();
    ES = 1;
}
```

```

EA = 1;
UartSENDStr("Uart Test !\r\n");

while (1)
{
    if (rptr != wptr)
    {
        UartSEND(buffer[rptr++]);
        rptr &= 0x0f;
    }
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

AUXR	DATA	8EH
T2H	DATA	0D6H
T2L	DATA	0D7H

BUSY	BIT	20H.0
WPTR	DATA	21H
RPTR	DATA	22H
BUFFER	DATA	23H ;16 bytes

P0M1	DATA	093H
P0M0	DATA	094H
P1M1	DATA	091H
P1M0	DATA	092H
P2M1	DATA	095H
P2M0	DATA	096H
P3M1	DATA	0B1H
P3M0	DATA	0B2H
P4M1	DATA	0B3H
P4M0	DATA	0B4H
P5M1	DATA	0C9H
P5M0	DATA	0CAH

ORG	0000H
LJMP	MAIN
ORG	0023H
LJMP	UART_ISR

ORG	0100H

UART_ISR:		
	PUSH	ACC
	PUSH	PSW
	MOV	PSW,#08H

	JNB	TI,CHKRI
	CLR	TI
	CLR	BUSY

CHKRI:		
	JNB	RI,UARTISR_EXIT
	CLR	RI
	MOV	A,WPTR
	ANL	A,#0FH
	ADD	A,#BUFFER

```

        MOV     R0,A
        MOV     @R0,SBUF
        INC     WPTR
UARTISR_EXIT:
        POP     PSW
        POP     ACC
        RETI

UART_INIT:
        MOV     SCON,#50H
        MOV     T2L,#0E8H                ;65536-11059200/115200/4=0FFE8H
        MOV     T2H,#0FFH
        MOV     AUXR,#15H
        CLR     BUSY
        MOV     WPTR,#00H
        MOV     RPTR,#00H
        RET

UART_SEND:
        JB      BUSY,$
        SETB    BUSY
        MOV     SBUF,A
        RET

UART_SENDSTR:
        CLR     A
        MOVC    A,@A+DPTR
        JZ      SENDEND
        LCALL   UART_SEND
        INC     DPTR
        JMP     UART_SENDSTR
SENDEND:
        RET

MAIN:
        MOV     SP,#5FH
        MOV     P0M0,#00H
        MOV     P0M1,#00H
        MOV     P1M0,#00H
        MOV     P1M1,#00H
        MOV     P2M0,#00H
        MOV     P2M1,#00H
        MOV     P3M0,#00H
        MOV     P3M1,#00H
        MOV     P4M0,#00H
        MOV     P4M1,#00H
        MOV     P5M0,#00H
        MOV     P5M1,#00H

        LCALL   UART_INIT
        SETB    ES
        SETB    EA

        MOV     DPTR,#STRING
        LCALL   UART_SENDSTR

LOOP:
        MOV     A,RPTR
        XRL     A,WPTR

```

```
        ANL          A,#0FH
        JZ           LOOP
        MOV          A,RPTR
        ANL          A,#0FH
        ADD          A,#BUFFER
        MOV          R0,A
        MOV          A,@R0
        LCALL        UART_SEND
        INC          RPTR
        JMP          LOOP

STRING:  DB          'Uart Test !',0DH,0AH,00H

        END
```

13.5.20 Timer 2 is used as baud rate generator of UART2

C language code

//Operating frequency for test is 11.0592MHz

```
#include "reg51.h"
#include "intrins.h"
```

```
#define FOSC          11059200UL
#define BRT           (65536 - FOSC / 115200 / 4)
```

```
sfr  AUXR      = 0x8e;
sfr  T2H       = 0xd6;
sfr  T2L       = 0xd7;
sfr  S2CON     = 0x9a;
sfr  S2BUF     = 0x9b;
sfr  IE2       = 0xaf;
```

```
sfr  P0M1      = 0x93;
sfr  P0M0      = 0x94;
sfr  P1M1      = 0x91;
sfr  P1M0      = 0x92;
sfr  P2M1      = 0x95;
sfr  P2M0      = 0x96;
sfr  P3M1      = 0xb1;
sfr  P3M0      = 0xb2;
sfr  P4M1      = 0xb3;
sfr  P4M0      = 0xb4;
sfr  P5M1      = 0xc9;
sfr  P5M0      = 0xca;
```

```
bit   busy;
char  wptr;
char  rptr;
char  buffer[16];
```

```
void Uart2Isr() interrupt 8
{
    if (S2CON & 0x02)
    {
        S2CON &= ~0x02;
        busy = 0;
    }
}
```

```
    }
    if (S2CON & 0x01)
    {
        S2CON &= ~0x01;
        buffer[wptr++] = S2BUF;
        wptr &= 0x0f;
    }
}

void Uart2Init()
{
    S2CON = 0x10;
    T2L = BRT;
    T2H = BRT >> 8;
    AUXR = 0x14;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}

void Uart2Send(char dat)
{
    while (busy);
    busy = 1;
    S2BUF = dat;
}

void Uart2SendStr(char *p)
{
    while (*p)
    {
        Uart2SEND(*p++);
    }
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    Uart2Init();
    IE2 = 0x01;
    EA = 1;
    Uart2SENDStr("Uart Test !\r\n");

    while (1)
    {
        if (rptr != wptr)
        {

```

```

        Uart2SEND(buffer[rptr++]);
        rptr &= 0x0f;
    }
}
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

AUXR	DATA	8EH	
T2H	DATA	0D6H	
T2L	DATA	0D7H	
S2CON	DATA	9AH	
S2BUF	DATA	9BH	
IE2	DATA	0AFH	
BUSY	BIT	20H.0	
WPTR	DATA	21H	
RPTR	DATA	22H	
BUFFER	DATA	23H	;16 bytes
P0M1	DATA	093H	
P0M0	DATA	094H	
P1M1	DATA	091H	
P1M0	DATA	092H	
P2M1	DATA	095H	
P2M0	DATA	096H	
P3M1	DATA	0B1H	
P3M0	DATA	0B2H	
P4M1	DATA	0B3H	
P4M0	DATA	0B4H	
P5M1	DATA	0C9H	
P5M0	DATA	0CAH	
	ORG	0000H	
	LJMP	MAIN	
	ORG	0043H	
	LJMP	UART2_ISR	
	ORG	0100H	
UART2_ISR:			
	PUSH	ACC	
	PUSH	PSW	
	MOV	PSW,#08H	
	MOV	A,S2CON	
	JNB	ACC.1,CHKRI	
	ANL	S2CON,#NOT 02H	
	CLR	BUSY	
CHKRI:			
	JNB	ACC.0,UART2ISR_EXIT	
	ANL	S2CON,#NOT 01H	
	MOV	A,WPTR	
	ANL	A,#0FH	
	ADD	A,#BUFFER	
	MOV	R0,A	
	MOV	@R0,S2BUF	
	INC	WPTR	

UART2ISR_EXIT:

```
POP     PSW
POP     ACC
RETI
```

UART2_INIT:

```
MOV     S2CON,#10H
MOV     T2L,#0E8H           ;65536-11059200/115200/4=0FFE8H
MOV     T2H,#0FFH
MOV     AUXR,#14H
CLR     BUSY
MOV     WPTR,#00H
MOV     RPTR,#00H
RET
```

UART2_SEND:

```
JB      BUSY,$
SETB    BUSY
MOV     S2BUF,A
RET
```

UART2_SENDSTR:

```
CLR     A
MOVC    A,@A+DPTR
JZ      SEND2END
LCALL   UART2_SEND
INC     DPTR
JMP     UART2_SENDSTR
```

SEND2END:

```
RET
```

MAIN:

```
MOV     SP,#5FH
MOV     P0M0,#00H
MOV     P0M1,#00H
MOV     P1M0,#00H
MOV     P1M1,#00H
MOV     P2M0,#00H
MOV     P2M1,#00H
MOV     P3M0,#00H
MOV     P3M1,#00H
MOV     P4M0,#00H
MOV     P4M1,#00H
MOV     P5M0,#00H
MOV     P5M1,#00H

LCALL   UART2_INIT
MOV     IE2,#01H
SETB    EA

MOV     DPTR,#STRING
LCALL   UART2_SENDSTR
```

LOOP:

```
MOV     A,RPTR
XRL     A,WPTR
ANL     A,#0FH
JZ      LOOP
MOV     A,RPTR
```

```
ANL      A,#0FH
ADD      A,#BUFFER
MOV      R0,A
MOV      A,@R0
LCALL    UART2_SEND
INC      RPTR
JMP      LOOP
```

```
STRING:  DB      'Uart Test !',0DH,0AH,00H
```

```
END
```

13.5.21 Timer 2 is used as baud rate generator of UART3

C language code

//Operating frequency for test is 11.0592MHz

```
#include "reg51.h"
#include "intrins.h"
```

```
#define FOSC      11059200UL
#define BRT      (65536 - FOSC / 115200 / 4)
```

```
sfr AUXR      = 0x8e;
sfr T2H       = 0xd6;
sfr T2L       = 0xd7;
sfr S3CON     = 0xac;
sfr S3BUF     = 0xad;
sfr IE2       = 0xaf;
```

```
sfr P0M1      = 0x93;
sfr P0M0      = 0x94;
sfr P1M1      = 0x91;
sfr P1M0      = 0x92;
sfr P2M1      = 0x95;
sfr P2M0      = 0x96;
sfr P3M1      = 0xb1;
sfr P3M0      = 0xb2;
sfr P4M1      = 0xb3;
sfr P4M0      = 0xb4;
sfr P5M1      = 0xc9;
sfr P5M0      = 0xca;
```

```
bit busy;
char wptr;
char rptr;
char buffer[16];
```

```
void Uart3Isr() interrupt 17
```

```
{
    if (S3CON & 0x02)
    {
        S3CON &= ~0x02;
        busy = 0;
    }
    if (S3CON & 0x01)
    {
```



```
        S3CON &= ~0x01;
        buffer[wptr++] = S3BUF;
        wptr &= 0x0f;
    }
}

void Uart3Init()
{
    S3CON = 0x10;
    T2L = BRT;
    T2H = BRT >> 8;
    AUXR = 0x14;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}

void Uart3Send(char dat)
{
    while (busy);
    busy = 1;
    S3BUF = dat;
}

void Uart3SendStr(char *p)
{
    while (*p)
    {
        Uart3SEND(*p++);
    }
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    Uart3Init();
    IE2 = 0x08;
    EA = 1;
    Uart3SENDStr("Uart Test !\r\n");

    while (1)
    {
        if (rptr != wptr)
        {
            Uart3SEND(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}
```

```

    }
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

AUXR      DATA      8EH
T2H       DATA      0D6H
T2L       DATA      0D7H
S3CON     DATA      0ACH
S3BUF     DATA      0ADH
IE2       DATA      0AFH

BUSY      BIT         20H.0
WPTR      DATA      21H
RPTR      DATA      22H
BUFFER    DATA      23H                ;16 bytes

P0M1      DATA      093H
P0M0      DATA      094H
P1M1      DATA      091H
P1M0      DATA      092H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

        ORG          0000H
        LJMP         MAIN
        ORG          008BH
        LJMP         UART3_ISR

        ORG          0100H

UART3_ISR:
        PUSH         ACC
        PUSH         PSW
        MOV          PSW,#08H

        MOV          A,S3CON
        JNB          ACC.1,CHKRI
        ANL          S3CON,#NOT 02H
        CLR          BUSY

CHKRI:
        JNB          ACC.0,UART3ISR_EXIT
        ANL          S3CON,#NOT 01H
        MOV          A,WPTR
        ANL          A,#0FH
        ADD          A,#BUFFER
        MOV          R0,A
        MOV          @R0,S3BUF
        INC          WPTR

UART3ISR_EXIT:
        POP          PSW
        POP          ACC

```

```

    RETI

UART3_INIT:
    MOV     S3CON,#10H
    MOV     T2L,#0E8H                ;65536-11059200/115200/4=0FFE8H
    MOV     T2H,#0FFH
    MOV     AUXR,#14H
    CLR     BUSY
    MOV     WPTR,#00H
    MOV     RPTR,#00H
    RET

UART3_SEND:
    JB      BUSY,$
    SETB    BUSY
    MOV     S3BUF,A
    RET

UART3_SENDSTR:
    CLR     A
    MOV     A,@A+DPTR
    JZ      SEND3END
    LCALL   UART3_SEND
    INC     DPTR
    JMP     UART3_SENDSTR

SEND3END:
    RET

MAIN:
    MOV     SP,#5FH
    MOV     P0M0,#00H
    MOV     P0M1,#00H
    MOV     P1M0,#00H
    MOV     P1M1,#00H
    MOV     P2M0,#00H
    MOV     P2M1,#00H
    MOV     P3M0,#00H
    MOV     P3M1,#00H
    MOV     P4M0,#00H
    MOV     P4M1,#00H
    MOV     P5M0,#00H
    MOV     P5M1,#00H

    LCALL   UART3_INIT
    MOV     IE2,#08H
    SETB    EA

    MOV     DPTR,#STRING
    LCALL   UART3_SENDSTR

LOOP:
    MOV     A,RPTR
    XRL     A,WPTR
    ANL     A,#0FH
    JZ      LOOP
    MOV     A,RPTR
    ANL     A,#0FH
    ADD     A,#BUFFER
    MOV     R0,A

```

```
        MOV        A,@R0
        LCALL      UART3_SEND
        INC        RPTR
        JMP        LOOP

STRING:  DB         'Uart Test !',0DH,0AH,00H

        END
```

13.5.22 Timer 2 is used as baud rate generator of UART4

C language code

//Operating frequency for test is 11.0592MHz

```
#include "reg51.h"
#include "intrins.h"

#define  FOSC      11059200UL
#define  BRT       (65536 - FOSC / 115200 / 4)

sfr  AUXR      = 0x8e;
sfr  T2H       = 0xd6;
sfr  T2L       = 0xd7;
sfr  S4CON     = 0x84;
sfr  S4BUF     = 0x85;
sfr  IE2       = 0xaf;

sfr  P0M1      = 0x93;
sfr  P0M0      = 0x94;
sfr  P1M1      = 0x91;
sfr  P1M0      = 0x92;
sfr  P2M1      = 0x95;
sfr  P2M0      = 0x96;
sfr  P3M1      = 0xb1;
sfr  P3M0      = 0xb2;
sfr  P4M1      = 0xb3;
sfr  P4M0      = 0xb4;
sfr  P5M1      = 0xc9;
sfr  P5M0      = 0xca;

bit   busy;
char  wptr;
char  rptr;
char  buffer[16];

void Uart4Isr() interrupt 18
{
    if (S4CON & 0x02)
    {
        S4CON &= ~0x02;
        busy = 0;
    }
    if (S4CON & 0x01)
    {
        S4CON &= ~0x01;
        buffer[wptr++] = S4BUF;
        wptr &= 0x0f;
    }
}
```

```
    }  
}  
  
void Uart4Init()  
{  
    S4CON = 0x10;  
    T2L = BRT;  
    T2H = BRT >> 8;  
    AUXR = 0x14;  
    wptr = 0x00;  
    rptr = 0x00;  
    busy = 0;  
}  
  
void Uart4Send(char dat)  
{  
    while (busy);  
    busy = 1;  
    S4BUF = dat;  
}  
  
void Uart4SendStr(char *p)  
{  
    while (*p)  
    {  
        Uart4SEND(*p++);  
    }  
}  
  
void main()  
{  
    P0M0 = 0x00;  
    P0M1 = 0x00;  
    P1M0 = 0x00;  
    P1M1 = 0x00;  
    P2M0 = 0x00;  
    P2M1 = 0x00;  
    P3M0 = 0x00;  
    P3M1 = 0x00;  
    P4M0 = 0x00;  
    P4M1 = 0x00;  
    P5M0 = 0x00;  
    P5M1 = 0x00;  
  
    Uart4Init();  
    IE2 = 0x10;  
    EA = 1;  
    Uart4SENDStr("Uart Test !\r\n");  
  
    while (1)  
    {  
        if (rptr != wptr)  
        {  
            Uart4SEND(buffer[rptr++]);  
            rptr &= 0x0f;  
        }  
    }  
}
```

Assembly code

;Operating frequency for test is 11.0592MHz

```

AUXR      DATA      8EH
T2H       DATA      0D6H
T2L       DATA      0D7H
S4CON     DATA      84H
S4BUF     DATA      85H
IE2       DATA      0AFH

BUSY      BIT         20H.0
WPTR      DATA      21H
RPTR      DATA      22H
BUFFER    DATA      23H                ;16 bytes

P0M1      DATA      093H
P0M0      DATA      094H
P1M1      DATA      091H
P1M0      DATA      092H
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

ORG        0000H
LJMP      MAIN
ORG        0093H
LJMP      UART4_ISR

ORG        0100H

UART4_ISR:
    PUSH   ACC
    PUSH   PSW
    MOV     PSW,#08H

    MOV     A,S4CON
    JNB     ACC.1,CHKRI
    ANL     S4CON,#NOT 02H
    CLR     BUSY

CHKRI:
    JNB     ACC.0,UART4ISR_EXIT
    ANL     S4CON,#NOT 01H
    MOV     A,WPTR
    ANL     A,#0FH
    ADD     A,#BUFFER
    MOV     R0,A
    MOV     @R0,S4BUF
    INC     WPTR

UART4ISR_EXIT:
    POP     PSW
    POP     ACC
    RETI

UART4_INIT:

```

```

        MOV     S4CON,#10H
        MOV     T2L,#0E8H                ;65536-11059200/115200/4=0FFE8H
        MOV     T2H,#0FFH
        MOV     AUXR,#14H
        CLR     BUSY
        MOV     WPTR,#00H
        MOV     RPTR,#00H
        RET

UART4_SEND:
        JB      BUSY,$
        SETB    BUSY
        MOV     S4BUF,A
        RET

UART4_SENDSTR:
        CLR     A
        MOVC    A,@A+DPTR
        JZ      SEND4END
        LCALL   UART4_SEND
        INC     DPTR
        JMP     UART4_SENDSTR

SEND4END:
        RET

MAIN:
        MOV     SP,#5FH
        MOV     P0M0,#00H
        MOV     P0M1,#00H
        MOV     P1M0,#00H
        MOV     P1M1,#00H
        MOV     P2M0,#00H
        MOV     P2M1,#00H
        MOV     P3M0,#00H
        MOV     P3M1,#00H
        MOV     P4M0,#00H
        MOV     P4M1,#00H
        MOV     P5M0,#00H
        MOV     P5M1,#00H

        LCALL   UART4_INIT
        MOV     IE2,#10H
        SETB    EA

        MOV     DPTR,#STRING
        LCALL   UART4_SENDSTR

LOOP:
        MOV     A,RPTR
        XRL     A,WPTR
        ANL     A,#0FH
        JZ      LOOP
        MOV     A,RPTR
        ANL     A,#0FH
        ADD     A,#BUFFER
        MOV     R0,A
        MOV     A,@R0
        LCALL   UART4_SEND
        INC     RPTR

```

```

                JMP      LOOP

STRING:        DB      'Uart Test !',0DH,0AH,00H

                END

```

13.5.23 Timer 3 (16-bit auto reload)

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

sfr      T4T3M      =    0xd1;
sfr      T4L        =    0xd3;
sfr      T4H        =    0xd2;
sfr      T3L        =    0xd5;
sfr      T3H        =    0xd4;
sfr      T2L        =    0xd7;
sfr      T2H        =    0xd6;
sfr      AUXR       =    0x8e;
sfr      IE2        =    0xaf;
#define   ET2        0x04
#define   ET3        0x20
#define   ET4        0x40
sfr      AUXINTIF    =    0xef;
#define   T2IF       0x01
#define   T3IF       0x02
#define   T4IF       0x04

```

```

sfr      P0M1       =    0x93;
sfr      P0M0       =    0x94;
sfr      P1M1       =    0x91;
sfr      P1M0       =    0x92;
sfr      P2M1       =    0x95;
sfr      P2M0       =    0x96;
sfr      P3M1       =    0xb1;
sfr      P3M0       =    0xb2;
sfr      P4M1       =    0xb3;
sfr      P4M0       =    0xb4;
sfr      P5M1       =    0xc9;
sfr      P5M0       =    0xca;

```

```

sbit      P10       =    P1^0;

```

```

void TM3_Isr() interrupt 19

```

```

{
    P10 = !P10;           //Test port
}

```

```

void main()

```

```

{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
}

```



```

P2M0 = 0x00;
P2M1 = 0x00;
P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

T3L = 0x66;           //65536-11.0592M/12/1000
T3H = 0xfc;
T4T3M = 0x08;         //Start timer
IE2 = ET3;            //Enable timer interrupt
EA = 1;

while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

T4T3M      DATA      0D1H
T4L         DATA      0D3H
T4H         DATA      0D2H
T3L         DATA      0D5H
T3H         DATA      0D4H
T2L         DATA      0D7H
T2H         DATA      0D6H
AUXR        DATA      8EH
IE2         DATA      0AFH
ET2         EQU         04H
ET3         EQU         20H
ET4         EQU         40H
AUXINTIF    DATA      0EFH
T2IF        EQU         01H
T3IF        EQU         02H
T4IF        EQU         04H

P0M1        DATA      093H
P0M0        DATA      094H
P1M1        DATA      091H
P1M0        DATA      092H
P2M1        DATA      095H
P2M0        DATA      096H
P3M1        DATA      0B1H
P3M0        DATA      0B2H
P4M1        DATA      0B3H
P4M0        DATA      0B4H
P5M1        DATA      0C9H
P5M0        DATA      0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         009BH
            LJMP        TM3ISR

            ORG         0100H
TM3ISR:
            CPL          P1.0

```

;Test port

RETI**MAIN:**

```

MOV     SP, #5FH
MOV     P0M0, #00H
MOV     P0M1, #00H
MOV     P1M0, #00H
MOV     P1M1, #00H
MOV     P2M0, #00H
MOV     P2M1, #00H
MOV     P3M0, #00H
MOV     P3M1, #00H
MOV     P4M0, #00H
MOV     P4M1, #00H
MOV     P5M0, #00H
MOV     P5M1, #00H

MOV     T3L, #66H           ;65536-11.0592M/12/1000
MOV     T3H, #0FCH
MOV     T4T3M, #08H        ;Start timer
MOV     IE2, #ET3          ;Enable timer interrupt
SETB    EA

JMP     $

END

```

13.5.24 Timer 3 (External count – T3 is extended for external falling edge interrupt)

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

sfr      T4T3M      = 0xd1;
sfr      T4L        = 0xd3;
sfr      T4H        = 0xd2;
sfr      T3L        = 0xd5;
sfr      T3H        = 0xd4;
sfr      T2L        = 0xd7;
sfr      T2H        = 0xd6;
sfr      AUXR       = 0x8e;
sfr      IE2        = 0xaf;
#define   ET2        0x04
#define   ET3        0x20
#define   ET4        0x40
sfr      AUXINTIF    = 0xef;
#define   T2IF       0x01
#define   T3IF       0x02
#define   T4IF       0x04

sfr      P0M1       = 0x93;
sfr      P0M0       = 0x94;

```

```

sfr      P1M1      = 0x91;
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;

sbit     P10       = P1^0;

```

```
void TM3_Isr() interrupt 19
```

```

{
    P10 = !P10;           //Test port
}

```

```
void main()
```

```

{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    T3L = 0x66;           //65536-11.0592M/12/1000
    T3H = 0xfc;
    T4T3M = 0x0c;         //Set external counting mode and start timer
    IE2 = ET3;             //Enable timer interrupt
    EA = 1;

    while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

T4T3M      DATA      0D1H
T4L         DATA      0D3H
T4H         DATA      0D2H
T3L         DATA      0D5H
T3H         DATA      0D4H
T2L         DATA      0D7H
T2H         DATA      0D6H
AUXR        DATA      8EH
IE2         DATA      0AFH
ET2         EQU         04H
ET3         EQU         20H
ET4         EQU         40H
AUXINTIF    DATA      0EFH
T2IF        EQU         01H
T3IF        EQU         02H

```

```

T4IF      EQU      04H

P0M1      DATA    093H
P0M0      DATA    094H
P1M1      DATA    091H
P1M0      DATA    092H
P2M1      DATA    095H
P2M0      DATA    096H
P3M1      DATA    0B1H
P3M0      DATA    0B2H
P4M1      DATA    0B3H
P4M0      DATA    0B4H
P5M1      DATA    0C9H
P5M0      DATA    0CAH

          ORG      0000H
          LJMP     MAIN
          ORG      009BH
          LJMP     TM3ISR

TM3ISR:    ORG      0100H

          CPL      P1.0          ;Test port
          RETI

MAIN:

          MOV      SP, #5FH
          MOV      P0M0, #00H
          MOV      P0M1, #00H
          MOV      P1M0, #00H
          MOV      P1M1, #00H
          MOV      P2M0, #00H
          MOV      P2M1, #00H
          MOV      P3M0, #00H
          MOV      P3M1, #00H
          MOV      P4M0, #00H
          MOV      P4M1, #00H
          MOV      P5M0, #00H
          MOV      P5M1, #00H

          MOV      T3L, #66H      ;65536-11.0592M/12/1000
          MOV      T3H, #0FCH
          MOV      T4T3M, #0CH    ;Set external counting mode and start timer
          MOV      IE2, #ET3      ;Enable timer interrupt
          SETB     EA

          JMP      $

          END

```

13.5.25 Timer 3 (Divided clock output)

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```
sfr      T4T3M      =    0xd1;
sfr      T4L        =    0xd3;
sfr      T4H        =    0xd2;
sfr      T3L        =    0xd5;
sfr      T3H        =    0xd4;
sfr      T2L        =    0xd7;
sfr      T2H        =    0xd6;
```

```
sfr      P0M1      =    0x93;
sfr      P0M0      =    0x94;
sfr      P1M1      =    0x91;
sfr      P1M0      =    0x92;
sfr      P2M1      =    0x95;
sfr      P2M0      =    0x96;
sfr      P3M1      =    0xb1;
sfr      P3M0      =    0xb2;
sfr      P4M1      =    0xb3;
sfr      P4M0      =    0xb4;
sfr      P5M1      =    0xc9;
sfr      P5M0      =    0xca;
```

```
void main()
```

```
{
```

```
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;
```

```
    T3L = 0x66;
    T3H = 0xfc;
    T4T3M = 0x09;
```

```
//65536-11.0592M/12/1000
```

```
//Enable clock output and start timer
```

```
    while (1);
```

```
}
```

Assembly code

;Operating frequency for test is 11.0592MHz

T4T3M	DATA	0D1H
T4L	DATA	0D3H
T4H	DATA	0D2H
T3L	DATA	0D5H
T3H	DATA	0D4H
T2L	DATA	0D7H
T2H	DATA	0D6H

P0M1	DATA	093H
P0M0	DATA	094H
P1M1	DATA	091H
P1M0	DATA	092H

```
P2M1      DATA      095H
P2M0      DATA      096H
P3M1      DATA      0B1H
P3M0      DATA      0B2H
P4M1      DATA      0B3H
P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN

MAIN:     ORG          0100H

          MOV          SP, #5FH
          MOV          P0M0, #00H
          MOV          P0M1, #00H
          MOV          P1M0, #00H
          MOV          P1M1, #00H
          MOV          P2M0, #00H
          MOV          P2M1, #00H
          MOV          P3M0, #00H
          MOV          P3M1, #00H
          MOV          P4M0, #00H
          MOV          P4M1, #00H
          MOV          P5M0, #00H
          MOV          P5M1, #00H

          MOV          T3L, #66H           ;65536-11.0592M/12/1000
          MOV          T3H, #0FCH
          MOV          T4T3M, #09H        ;Enable clock output and start timer

          JMP          $

          END
```

13.5.26 Timer 3 is used as baud rate generator of UART3

C language code

//Operating frequency for test is 11.0592MHz

```
#include "reg51.h"
#include "intrins.h"
```

```
#define FOSC      11059200UL
#define BRT      (65536 - FOSC / 115200 / 4)
```

```
sfr      T4T3M    = 0xd1;
sfr      T4L      = 0xd3;
sfr      T4H      = 0xd2;
sfr      T3L      = 0xd5;
sfr      T3H      = 0xd4;
sfr      T2L      = 0xd7;
sfr      T2H      = 0xd6;
sfr      S3CON     = 0xac;
sfr      S3BUF     = 0xad;
sfr      IE2       = 0xaf;
```

```
sfr      P0M1      = 0x93;
sfr      P0M0      = 0x94;
sfr      P1M1      = 0x91;
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;
```

```
bit      busy;
char     wptr;
char     rptr;
char     buffer[16];
```

```
void Uart3Isr() interrupt 17
```

```
{
    if (S3CON & 0x02)
    {
        S3CON &= ~0x02;
        busy = 0;
    }
    if (S3CON & 0x01)
    {
        S3CON &= ~0x01;
        buffer[wptr++] = S3BUF;
        wptr &= 0x0f;
    }
}
```

```
void Uart3Init()
```

```
{
    S3CON = 0x50;
    T3L = BRT;
    T3H = BRT >> 8;
    T4T3M = 0x0a;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}
```

```
void Uart3Send(char dat)
```

```
{
    while (busy);
    busy = 1;
    S3BUF = dat;
}
```

```
void Uart3SendStr(char *p)
```

```
{
    while (*p)
    {
        Uart3SEND(*p++);
    }
}
```

```

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    Uart3Init();
    IE2 = 0x08;
    EA = 1;
    Uart3SENDStr("Uart Test !\r\n");

    while (1)
    {
        if (rptr != wptr)
        {
            Uart3SEND(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

T4T3M	DATA	0D1H	
T4L	DATA	0D3H	
T4H	DATA	0D2H	
T3L	DATA	0D5H	
T3H	DATA	0D4H	
T2L	DATA	0D7H	
T2H	DATA	0D6H	
S3CON	DATA	0ACH	
S3BUF	DATA	0ADH	
IE2	DATA	0AFH	
BUSY	BIT	20H.0	
WPTR	DATA	21H	
RPTR	DATA	22H	
BUFFER	DATA	23H	;16 bytes
P0M1	DATA	093H	
P0M0	DATA	094H	
P1M1	DATA	091H	
P1M0	DATA	092H	
P2M1	DATA	095H	
P2M0	DATA	096H	
P3M1	DATA	0B1H	
P3M0	DATA	0B2H	
P4M1	DATA	0B3H	


```

P4M0      DATA      0B4H
P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          008BH
          LJMP         UART3_ISR

          ORG          0100H

UART3_ISR:
          PUSH         ACC
          PUSH         PSW
          MOV          PSW,#08H

          MOV          A,S3CON
          JNB          ACC.1,CHKRI
          ANL          S3CON,#NOT 02H
          CLR          BUSY

CHKRI:
          JNB          ACC.0,UART3ISR_EXIT
          ANL          S3CON,#NOT 01H
          MOV          A,WPTR
          ANL          A,#0FH
          ADD          A,#BUFFER
          MOV          R0,A
          MOV          @R0,S3BUF
          INC          WPTR

UART3ISR_EXIT:
          POP          PSW
          POP          ACC
          RETI

UART3_INIT:
          MOV          S3CON,#50H
          MOV          T3L,#0E8H
          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

```

;65536-11059200/115200/4=0FFE8H

MAIN:

```
MOV     SP, #5FH
MOV     P0M0, #00H
MOV     P0M1, #00H
MOV     P1M0, #00H
MOV     P1M1, #00H
MOV     P2M0, #00H
MOV     P2M1, #00H
MOV     P3M0, #00H
MOV     P3M1, #00H
MOV     P4M0, #00H
MOV     P4M1, #00H
MOV     P5M0, #00H
MOV     P5M1, #00H

LCALL   UART3_INIT
MOV     IE2, #08H
SETB    EA

MOV     DPTR, #STRING
LCALL   UART3_SENDSTR
```

LOOP:

```
MOV     A, RPTR
XRL     A, WPTR
ANL     A, #0FH
JZ      LOOP
MOV     A, RPTR
ANL     A, #0FH
ADD     A, #BUFFER
MOV     R0, A
MOV     A, @R0
LCALL   UART3_SEND
INC     RPTR
JMP     LOOP
```

```
STRING:  DB      'Uart Test !', 0DH, 0AH, 00H
```

```
END
```

13.5.27 Timer 4 (16-bit auto reload)

C language code

//Operating frequency for test is 11.0592MHz

```
#include "reg51.h"
#include "intrins.h"
```

```
sfr     T4T3M    = 0xd1;
sfr     T4L      = 0xd3;
sfr     T4H      = 0xd2;
sfr     T3L      = 0xd5;
sfr     T3H      = 0xd4;
sfr     T2L      = 0xd7;
sfr     T2H      = 0xd6;
sfr     AUXR     = 0x8e;
```

```

sfr      IE2          = 0xaf;
#define   ET2          0x04
#define   ET3          0x20
#define   ET4          0x40
sfr      AUXINTIF     = 0xef;
#define   T2IF         0x01
#define   T3IF         0x02
#define   T4IF         0x04

```

```

sfr      P0M1         = 0x93;
sfr      P0M0         = 0x94;
sfr      P1M1         = 0x91;
sfr      P1M0         = 0x92;
sfr      P2M1         = 0x95;
sfr      P2M0         = 0x96;
sfr      P3M1         = 0xb1;
sfr      P3M0         = 0xb2;
sfr      P4M1         = 0xb3;
sfr      P4M0         = 0xb4;
sfr      P5M1         = 0xc9;
sfr      P5M0         = 0xca;

```

```

sbit     P10          = P1^0;

```

```

void TM4_Isr() interrupt 20

```

```

{
    P10 = !P10;           //Test port
}

```

```

void main()

```

```

{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    T4L = 0x66;           //65536-11.0592M/12/1000
    T4H = 0xfc;
    T4T3M = 0x80;         //Start timer
    IE2 = ET4;            //Enable timer interrupt
    EA = 1;

    while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

```

T4T3M    DATA    0D1H
T4L      DATA    0D3H
T4H      DATA    0D2H

```

T3L	DATA	0D5H	
T3H	DATA	0D4H	
T2L	DATA	0D7H	
T2H	DATA	0D6H	
AUXR	DATA	8EH	
IE2	DATA	0AFH	
ET2	EQU	04H	
ET3	EQU	20H	
ET4	EQU	40H	
AUXINTIF	DATA	0EFH	
T2IF	EQU	01H	
T3IF	EQU	02H	
T4IF	EQU	04H	
P0M1	DATA	093H	
P0M0	DATA	094H	
P1M1	DATA	091H	
P1M0	DATA	092H	
P2M1	DATA	095H	
P2M0	DATA	096H	
P3M1	DATA	0B1H	
P3M0	DATA	0B2H	
P4M1	DATA	0B3H	
P4M0	DATA	0B4H	
P5M1	DATA	0C9H	
P5M0	DATA	0CAH	
	ORG	0000H	
	LJMP	MAIN	
	ORG	00A3H	
	LJMP	TM4ISR	
	ORG	0100H	
TM4ISR:			
	CPL	P1.0	;Test port
	RETI		
MAIN:			
	MOV	SP, #5FH	
	MOV	P0M0, #00H	
	MOV	P0M1, #00H	
	MOV	P1M0, #00H	
	MOV	P1M1, #00H	
	MOV	P2M0, #00H	
	MOV	P2M1, #00H	
	MOV	P3M0, #00H	
	MOV	P3M1, #00H	
	MOV	P4M0, #00H	
	MOV	P4M1, #00H	
	MOV	P5M0, #00H	
	MOV	P5M1, #00H	
	MOV	T4L, #66H	;65536-11.0592M/12/1000
	MOV	T4H, #0FCH	
	MOV	T4T3M, #80H	;Start timer
	MOV	IE2, #ET4	;Enable timer interrupt
	SETB	EA	
	JMP	\$	

END

13.5.28 Timer 4 (External count – T4 is extended for external falling edge interrupt)

C language code

//Operating frequency for test is 11.0592MHz

```
#include "reg51.h"
#include "intrins.h"
```

```
sfr      T4T3M      = 0xd1;
sfr      T4L        = 0xd3;
sfr      T4H        = 0xd2;
sfr      T3L        = 0xd5;
sfr      T3H        = 0xd4;
sfr      T2L        = 0xd7;
sfr      T2H        = 0xd6;
sfr      AUXR       = 0x8e;
sfr      IE2        = 0xaf;
#define   ET2        0x04
#define   ET3        0x20
#define   ET4        0x40
sfr      AUXINTIF    = 0xef;
#define   T2IF       0x01
#define   T3IF       0x02
#define   T4IF       0x04
```

```
sfr      P0M1       = 0x93;
sfr      P0M0       = 0x94;
sfr      P1M1       = 0x91;
sfr      P1M0       = 0x92;
sfr      P2M1       = 0x95;
sfr      P2M0       = 0x96;
sfr      P3M1       = 0xb1;
sfr      P3M0       = 0xb2;
sfr      P4M1       = 0xb3;
sfr      P4M0       = 0xb4;
sfr      P5M1       = 0xc9;
sfr      P5M0       = 0xca;
```

```
sbit     P10        = P1^0;
```

```
void TM4_Isr() interrupt 20
```

```
{
    P10 = !P10;           //Test port
}
```

```
void main()
```

```
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
```

```

P2M0 = 0x00;
P2M1 = 0x00;
P3M0 = 0x00;
P3M1 = 0x00;
P4M0 = 0x00;
P4M1 = 0x00;
P5M0 = 0x00;
P5M1 = 0x00;

```

```

T4L = 0x66;           //65536-11.0592M/12/1000
T4H = 0xfc;
T4T3M = 0xc0;         //Set external counting mode and start timer
IE2 = ET4;            //Enable timer interrupt
EA = 1;

```

```

while (1);

```

```

}

```

Assembly code

;Operating frequency for test is 11.0592MHz

T4T3M	DATA	0D1H
T4L	DATA	0D3H
T4H	DATA	0D2H
T3L	DATA	0D5H
T3H	DATA	0D4H
T2L	DATA	0D7H
T2H	DATA	0D6H
AUXR	DATA	8EH
IE2	DATA	0AFH
ET2	EQU	04H
ET3	EQU	20H
ET4	EQU	40H
AUXINTIF	DATA	0EFH
T2IF	EQU	01H
T3IF	EQU	02H
T4IF	EQU	04H
P0M1	DATA	093H
P0M0	DATA	094H
P1M1	DATA	091H
P1M0	DATA	092H
P2M1	DATA	095H
P2M0	DATA	096H
P3M1	DATA	0B1H
P3M0	DATA	0B2H
P4M1	DATA	0B3H
P4M0	DATA	0B4H
P5M1	DATA	0C9H
P5M0	DATA	0CAH
	ORG	0000H
	LJMP	MAIN
	ORG	00A3H
	LJMP	TM4ISR
	ORG	0100H
TM4ISR:		
	CPL	P1.0

;Test port

RETI**MAIN:**

```

MOV     SP, #5FH
MOV     P0M0, #00H
MOV     P0M1, #00H
MOV     P1M0, #00H
MOV     P1M1, #00H
MOV     P2M0, #00H
MOV     P2M1, #00H
MOV     P3M0, #00H
MOV     P3M1, #00H
MOV     P4M0, #00H
MOV     P4M1, #00H
MOV     P5M0, #00H
MOV     P5M1, #00H

MOV     T4L, #66H           ;65536-11.0592M/12/1000
MOV     T4H, #0FCH
MOV     T4T3M, #0C0H       ;Set external counting mode and start timer
MOV     IE2, #ET4          ;Enable timer interrupt
SETB    EA

JMP     $

END

```

13.5.29 Timer 4 (Divided clock output)

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

sfr     T4T3M    = 0xd1;
sfr     T4L      = 0xd3;
sfr     T4H      = 0xd2;
sfr     T3L      = 0xd5;
sfr     T3H      = 0xd4;
sfr     T2L      = 0xd7;
sfr     T2H      = 0xd6;

```

```

sfr     P0M1     = 0x93;
sfr     P0M0     = 0x94;
sfr     P1M1     = 0x91;
sfr     P1M0     = 0x92;
sfr     P2M1     = 0x95;
sfr     P2M0     = 0x96;
sfr     P3M1     = 0xb1;
sfr     P3M0     = 0xb2;
sfr     P4M1     = 0xb3;
sfr     P4M0     = 0xb4;
sfr     P5M1     = 0xc9;
sfr     P5M0     = 0xca;

```

```
void main()
```

```

{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P3M1 = 0x00;

    T4L = 0x66; //65536-11.0592M/12/1000
    T4H = 0xfc;
    T4T3M = 0x90; //Enable clock output and start timer

    while (1);
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

T4T3M	DATA	0D1H
T4L	DATA	0D3H
T4H	DATA	0D2H
T3L	DATA	0D5H
T3H	DATA	0D4H
T2L	DATA	0D7H
T2H	DATA	0D6H
P0M1	DATA	093H
P0M0	DATA	094H
P1M1	DATA	091H
P1M0	DATA	092H
P2M1	DATA	095H
P2M0	DATA	096H
P3M1	DATA	0B1H
P3M0	DATA	0B2H
P4M1	DATA	0B3H
P4M0	DATA	0B4H
P5M1	DATA	0C9H
P5M0	DATA	0CAH
	ORG	0000H
	LJMP	MAIN
	ORG	0100H
MAIN:	MOV	SP, #5FH
	MOV	P0M0, #00H
	MOV	P0M1, #00H
	MOV	P1M0, #00H
	MOV	P1M1, #00H
	MOV	P2M0, #00H
	MOV	P2M1, #00H
	MOV	P3M0, #00H
	MOV	P3M1, #00H


```

MOV      P4M0, #00H
MOV      P4M1, #00H
MOV      P5M0, #00H
MOV      P5M1, #00H

MOV      T4L, #66H           ;65536-11.0592M/12/1000
MOV      T4H, #0FCH
MOV      T4T3M, #90H        ;Enable clock output and start timer

JMP      $

END

```

13.5.30 Timer 4 is used as baud rate generator of UART4

C language code

//Operating frequency for test is 11.0592MHz

```

#include "reg51.h"
#include "intrins.h"

```

```

#define FOSC      11059200UL
#define BRT       (65536 - FOSC / 115200 / 4)

```

```

sfr      T4T3M    = 0xd1;
sfr      T4L      = 0xd3;
sfr      T4H      = 0xd2;
sfr      T3L      = 0xd5;
sfr      T3H      = 0xd4;
sfr      T2L      = 0xd7;
sfr      T2H      = 0xd6;
sfr      S4CON     = 0x84;
sfr      S4BUF     = 0x85;
sfr      IE2       = 0xaf;

```

```

sfr      P0M1     = 0x93;
sfr      P0M0     = 0x94;
sfr      P1M1     = 0x91;
sfr      P1M0     = 0x92;
sfr      P2M1     = 0x95;
sfr      P2M0     = 0x96;
sfr      P3M1     = 0xb1;
sfr      P3M0     = 0xb2;
sfr      P4M1     = 0xb3;
sfr      P4M0     = 0xb4;
sfr      P5M1     = 0xc9;
sfr      P5M0     = 0xca;

```

```

bit      busy;
char     wptr;
char     rprr;
char     buffer[16];

```

```

void Uart4Isr() interrupt 18
{
    if (S4CON & 0x02)
    {

```

```
        S4CON &= ~0x02;
        busy = 0;
    }
    if (S4CON & 0x01)
    {
        S4CON &= ~0x01;
        buffer[wptr++] = S4BUF;
        wptr &= 0x0f;
    }
}

void Uart4Init()
{
    S4CON = 0x50;
    T4L = BRT;
    T4H = BRT >> 8;
    T4T3M = 0xa0;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}

void Uart4Send(char dat)
{
    while (busy);
    busy = 1;
    S4BUF = dat;
}

void Uart4SendStr(char *p)
{
    while (*p)
    {
        Uart4SEND(*p++);
    }
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    Uart4Init();
    IE2 = 0x10;
    EA = 1;
    Uart4SENDStr("Uart Test !\r\n");

    while (1)
    {
```

```

        if (rptr != wptr)
        {
            Uart4SEND(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

T4T3M	DATA	0D1H
T4L	DATA	0D3H
T4H	DATA	0D2H
T3L	DATA	0D5H
T3H	DATA	0D4H
T2L	DATA	0D7H
T2H	DATA	0D6H
S4CON	DATA	84H
S4BUF	DATA	85H
IE2	DATA	0AFH

BUSY	BIT	20H.0
WPTR	DATA	21H
RPTR	DATA	22H
BUFFER	DATA	23H ;16 bytes

P0M1	DATA	093H
P0M0	DATA	094H
P1M1	DATA	091H
P1M0	DATA	092H
P2M1	DATA	095H
P2M0	DATA	096H
P3M1	DATA	0B1H
P3M0	DATA	0B2H
P4M1	DATA	0B3H
P4M0	DATA	0B4H
P5M1	DATA	0C9H
P5M0	DATA	0CAH

```

ORG      0000H
LJMP     MAIN
ORG      0093H
LJMP     UART4_ISR

```

```

ORG      0100H

```

UART4_ISR:

```

    PUSH    ACC
    PUSH    PSW
    MOV     PSW,#08H

```

```

    MOV     A,S4CON
    JNB     ACC.1,CHKRI
    ANL     S4CON,#NOT 02H
    CLR     BUSY

```

CHKRI:

```

    JNB     ACC.0,UART4ISR_EXIT
    ANL     S4CON,#NOT 01H

```

```

        MOV     A,WPTR
        ANL     A,#0FH
        ADD     A,#BUFFER
        MOV     R0,A
        MOV     @R0,S4BUF
        INC     WPTR
UART4ISR_EXIT:
        POP     PSW
        POP     ACC
        RETI

UART4_INIT:
        MOV     S4CON,#50H
        MOV     T4L,#0E8H
        MOV     T4H,#0FFH
        MOV     T4T3M,#0A0H
        CLR     BUSY
        MOV     WPTR,#00H
        MOV     RPTR,#00H
        RET

UART4_SEND:
        JB      BUSY,$
        SETB    BUSY
        MOV     S4BUF,A
        RET

UART4_SENDSTR:
        CLR     A
        MOVC    A,@A+DPTR
        JZ      SEND4END
        LCALL   UART4_SEND
        INC     DPTR
        JMP     UART4_SENDSTR
SEND4END:
        RET

MAIN:
        MOV     SP,#5FH
        MOV     P0M0,#00H
        MOV     P0M1,#00H
        MOV     P1M0,#00H
        MOV     P1M1,#00H
        MOV     P2M0,#00H
        MOV     P2M1,#00H
        MOV     P3M0,#00H
        MOV     P3M1,#00H
        MOV     P4M0,#00H
        MOV     P4M1,#00H
        MOV     P5M0,#00H
        MOV     P5M1,#00H

        LCALL   UART4_INIT
        MOV     IE2,#10H
        SETB    EA

        MOV     DPTR,#STRING
        LCALL   UART4_SENDSTR

```

LOOP:

```
MOV    A,RPTR
XRL    A,WPTR
ANL    A,#0FH
JZ     LOOP
MOV    A,RPTR
ANL    A,#0FH
ADD    A,#BUFFER
MOV    R0,A
MOV    A,@R0
LCALL  UART4_SEND
INC    RPTR
JMP    LOOP
```

STRING: **DB** 'Uart Test !',0DH,0AH,00H

END

STC MCU

14 UART Communication

Product line	Number of UART
STC8H1K08 family	2
STC8H1K28 family	2
STC8H3K64S4 family	4
STC8H3K64S2 family	2
STC8H8K64U family	4
STC8H2K64T family	4
STC8H4K64TLR family	4
STC8H4K64TLCD family	4
STC8H4K64LCD family	4

There are 4 full duplex asynchronous serial communication ports (UART in short) in STC8H series of microcontrollers. Each UART consists of two data buffers, a shift register, a serial control register and a baud rate generator. Each UART data buffer consists of two independent receive and transmit buffers, which can transmit and receive data simultaneously.

There are 4 modes for UART1 of STC8H series of microcontrollers, the baud rates of two modes of them are variable, and the baud rates of the other two modes are fixed. They can be chosen for different applications. There are only two modes in UART2, UART3 and UART4, and their baud rates are variable. Different baud rates and different modes can be set by software. It is flexible for the host to query the receiving or sending process, or use the interrupt method.

All the pins of UART1, UART2, UART3 and UART4 can be switched among multiple groups of ports using the pin switching function, so that a serial port can be multiplexed into several serial ports in a time-sharing manner.

14.1 Registers Related to UARTs

Symbol	Description	Address	Bit Address and Symbol								Reset value
			B7	B6	B5	B4	B3	B2	B1	B0	
SCON	UART1 control register	98H	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	0000,0000
SBUF	UART1 data buffer register	99H									0000,0000
S2CON	UART2 control register	9AH	S2SM0	-	S2SM2	S2REN	S2TB8	S2RB8	S2TI	S2RI	0100,0000
S2BUF	UART2 data buffer register	9BH									0000,0000
S3CON	UART3 control register	ACH	S3SM0	S3ST3	S3SM2	S3REN	S3TB8	S3RB8	S3TI	S3RI	0000,0000
S3BUF	UART3 data buffer register	ADH									0000,0000
S4CON	Serial port 4 control register	84H	S4SM0	S4ST4	S4SM2	S4REN	S4TB8	S4RB8	S4TI	S4RI	0000,0000
S4BUF	Serial port 4 data buffer register	85H									0000,0000
PCON	Power control register	87H	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000
AUXR	Auxiliary register 1	8EH	T0x12	T1x12	UART_M0x6	T2R	T2_C/T	T2x12	EXTRAM	S1ST2	0000,0001
SADDR	UART1 slave address register	A9H									0000,0000
SADEN	UART1 slave address enable register	B9H									0000,0000

14.2 UART1

14.2.1 UART1 control register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
SCON	98H	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI

SM0/FE: If the SMOD0 bit in the PCON register is 1, this bit is the frame error detection flag. When the UART detects an invalid stop bit during reception, it is set by the UART receiver and must be cleared by software. If SMOD0 bit in PCON register is 0, this bit and SM1 specify the communication mode of UART1 as shown in the following table:

SM0	SM1	Mode of UART1	Function description
0	0	Mode 0	synchronous shift serial mode
0	1	Mode 1	8-bit UART, whose baud-rate is variable
1	0	Mode 2	9-bit UART, whose baud-rate is fixed
1	1	Mode 3	9-bit UART, whose baud-rate is variable

SM2: Mode 2 or mode 3 multi-machine communication enable control bit. When UART1 adopts mode 2 or mode 3, if the SM2 bit is 1 and the REN bit is 1, the receiver is in the Address Frame Filter state. In this case, the received 9th bit (RB8) can be used to filter the address frame. If RB8 = 1, it indicates that the frame is an address frame, the address information can enter SBUF and set RI bit. The address information is compared in the interrupt service routine. If RB8 = 0, it indicates that the frame is not an address frame, which should be discarded and keep RI = 0. In mode 2 or mode 3, if the SM2 bit is 0 and the REN bit is 1, the receiver is in a state where the address frame filtering is disabled. The received message can enter SBUF regardless of whether RB8 is 0 or 1, and make RI = 1. Here, RB8 is usually used as a check bit. Mode 1 and mode 0 are non-multi-machine communication modes. In these two modes, SM2 should be set to 0.

REN: Receive enable control bit.

0: disable UART1 receive data.

1: enable UART1 receive data.

TB8: The 9th bit to be transmitted for UART1 in mode 2 and 3. It can be set or cleared by software. It is not used in mode 0 and mode 1.

RB8: The 9th bit received for UART1 in mode 2 and 3 which is usually used as a check bit or address frame/data frame flag. It is not used in mode 0 and mode 1.

TI: Transmit interrupt request flag of UART1. In mode 0, when the transmission of the 8th bit completes, TI is set by the hardware automatically and requests the interrupt to the CPU. After the CPU responds the interrupt, TI must be cleared by software. In other modes, TI is set by the hardware automatically at the start of the stop bit transmission and requests interrupts to the CPU. TI must be cleared by software after the interrupt is responded.

RI: Receive interrupt request flag of UART1. In mode 0, when the serial port receives the 8th bit of datum, RI is set by the hardware automatically and requests interrupt to the CPU. After the interrupt is responded, RI must be cleared by software. In other modes, RI is set by hardware automatically at the middle of stop bit the serial port received, and requests the interrupt to the CPU. After the interrupt is responded, RI must be cleared by software.

14.2.2 UART1 data register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
SBUF	99H								

SBUF: It is used as the buffer in transmission and receiving of UART1. SBUF is actually two buffers, reading buffer and writing buffer. Two operations correspond to two different registers, one is write-only register (writing buffer), the other is read-only register (reading buffer). In fact, the CPU reads serial receive

buffer when reads SBUF. When CPU writes to the SBUF will trigger the serial port to start sending data.

14.2.3 Power control register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
PCON	87H	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL

SMOD: double Baud rate of UART1 control bit.

0: disable double baud rate of the UART1.

1: enable double baud rate of the UART1.

SMOD0: Frame error detection control bit.

0: No frame error detection function, SCON.7 is SM0 function.

1: enable frame error detection function. The function of SM0/FE is FE.

14.2.4 Auxiliary register 1

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
AUXR	8EH	T0x12	T1x12	UART_M0x6	T2R	T2_C/T	T2x12	EXTRAM	S1ST2

UART_M0x6: Baud rate select bit of UART1 while it works in mode 0.

0: The baud-rate of UART in mode 0 is SYSclk/12.

1: The baud-rate of UART in mode 0 is SYSclk/2.

S1ST2: UART1 baud rate generator select bit.

0: Select Timer 1 as the baud-rate generator of UART1.

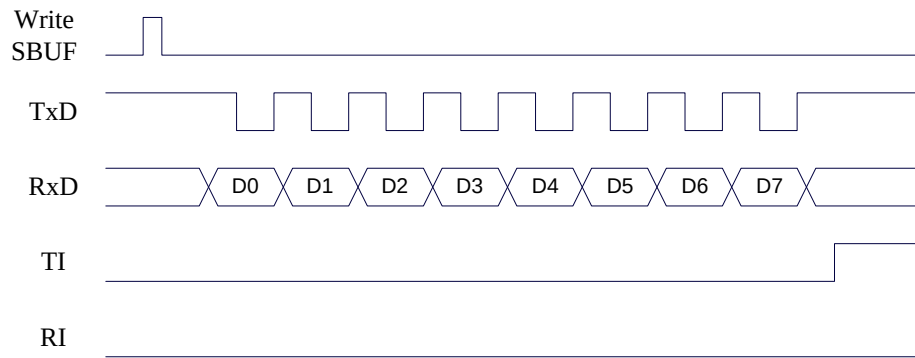
1: Select Timer 2 as the baud-rate generator of UART1.

14.2.5 UART1 Mode 0

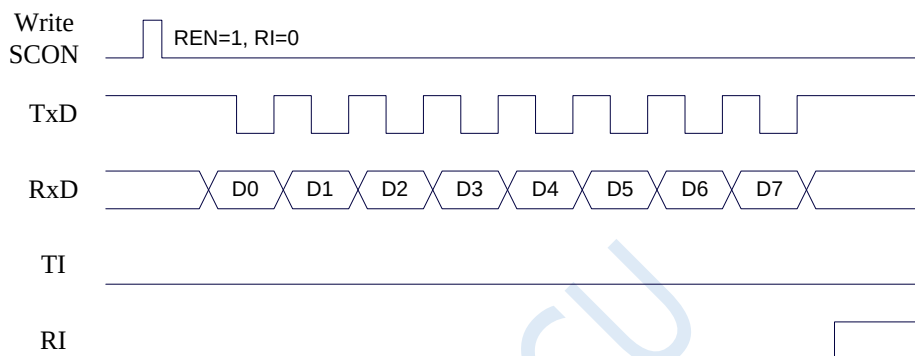
When mode 0 is selected for UART1, UART1 operates in synchronous shift register mode. When the serial port mode 0 communication speed setting bit UART_M0x6 is 0, the baud rate is fixed to SYSclk/12. When UART_M0x6 is 1, the baud rate is fixed to SYSclk/2. RxD is used as serial communication data pin, TxD is used as synchronous shift pulse output pin. 8-bit data are transmitted and received, LSB first.

Transmission process of mode 0: Transmission is initiated by any instruction that write data to SBUF. The 8-bit datum is output from the RxD pin at the baud rate of SYSclk/12 or SYSclk/2 (determined by the UART_M0x6 divided by 12 or 2), from LSB to MSB. The TxD pin outputs the synchronous shift pulse signal. The interrupt flag TI will be set when transmission is completed. When the write signal is valid, the transmit control signal SEND is active (high) one clock apart, allowing RxD to send data while allowing the TxD output the synchronous shift pulse. When a frame (8 bits) of datum is sent, all control signals are reset to the original status, and only TI keeps high level and keeps the interrupt request status. TI must be cleared by software before sending data again.

Receiving process of mode 0: Receiving is initiated by setting REN and the receive interrupt request flag RI=0. After starting the receive process, RxD is the serial data input pin and TxD is the synchronous pulse output pin. The serial receiving baud rate is SYSclk/12 or SYSclk/2 (determined by UART_M0x6 is 12 or 2). After receiving a frame of datum (8 bits), the control signal is reset and the interrupt flag RI is set to 1, the interrupt request status appears. RI must be cleared by software for the next receiving data.



Transmitting data (UART1 mode 0)



Receiving data (UART1 mode 0)

In mode 0, SM2 must be cleared so that TB8 and RB8 bits are not affected. Since the baud rate is fixed at SYSclk/12 or SYSclk/2, no timer is required and the clock of the microcontroller is used as the synchronous shift pulse directly.

The baud rates of UART1 mode 0 are shown in the following table, where SYSclk is the system operating frequency:

UART_M0x6	Baud rate calculation formula
0	$\text{Baud rate} = \frac{\text{SYSclk}}{12}$
1	$\text{Baud rate} = \frac{\text{SYSclk}}{2}$

14.2.6 UART1 Mode 1

If SM0 and SM1 of SCON are set to '01' by the software, UART1 will work in mode 1, which is a 8-bit UART mode. In mode 1, a frame of information consists 10 bits: 1 start bit, 8 data bits (LSB first) and 1 stop bit. The baud rate is variable, which can be set by the software as needed. Tx/D is the data transmitting pin, and Rx/D is the data receiving pin, the UART is a full duplex receiver/transmitter.

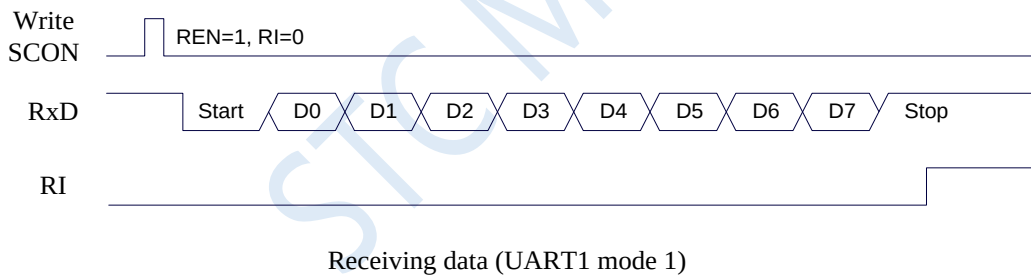
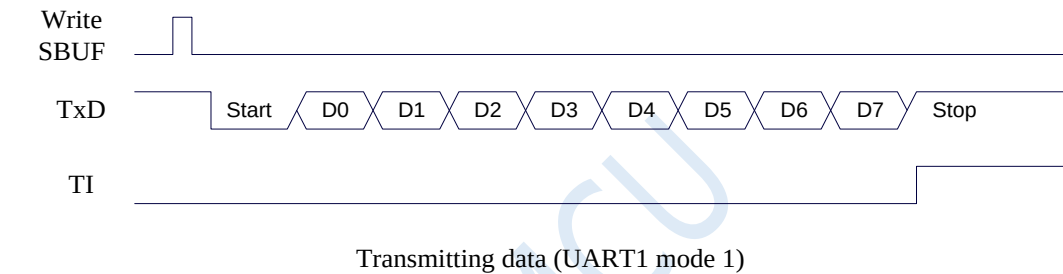
Transmission process of mode 1: Tx/D is used as data output pin when transmitting a datum. Transmission is initiated by writing SBUF. "1" is also written into the 9th bit of transmission shift register by the writing "SBUF"

signal, and the TX control unit is notified to start sending. The shift register shifts the data right to TxD to send, and shifts "0" in the left to supplement. When the highest bit of data is shifted to the output of the shift register, it is followed by the 9th bit "1", and all bits to the left of it are "0". This state causes the TX control unit to make the last shift output, and then disables the transmission signal "SEND" to complete the transmission of a frame and sets TI, and requests interrupt processing to CPU.

Receiving process of mode 1: After the software sets the reception enable flag REN, i.e. REN = 1, the receiver will detect the RxD pin signal. The receiver is ready to receive data when a "1" → "0" falling edge is detected at RxD pin, and resets the receiving counter of the baud rate generator immediately, loads 1FFH into the shift register. The received datum is shifted in from the right of the receiving shift register, the loaded 1FFH is shifted out to the left. When the start bit "0" is shifted to the far left of the shift register, the RX controller shifts for the last time and completes a frame receiving. The received datum is valid only if the following two conditions are met:

- RI=0;
- SM2=0 or the stop bit received is 1.

The datum received is loaded into SBUF, the stop bit is loaded into RB8, RI flag is set to request interrupt to CPU. If the two conditions can not be met at the same time, the received data is invalid and is discarded. Regardless of the conditions are met or not, the receiver will re-test RxD pin of the "1" → "0" edge, and continue to receive the next frame. If the received datum is valid, the RI flag must be cleared by software in the interrupt service routine. Usually, SM2 is set to "0" when serial port is operating in mode 1.



The baud rate of UART1 is variable. It can be generated by T1 or T2. If the timer is in 1T mode (12x speed), the corresponding baud rate is increased by 12 times.

The baud rate of UART1 mode 1 is calculated as follows, where SYSclk is the system operating frequency.

Timer selected	Speed of timer	Baud rate calculation formula
T2	1T	$\text{reload value of T2} = 65536 \div \frac{\text{SYSclk}}{4 \times \text{baud rate}}$
	12T	$\text{reload value of Timer 2} = 65536 \div \frac{\text{SYSclk}}{12 \times 4 \times \text{baud rate}}$
T1 mode 0	1T	$\text{reload value of T1} = 65536 \div \frac{\text{SYSclk}}{4 \times \text{baud rate}}$
	12T	$\text{reload value of T1} = 65536 \div \frac{\text{SYSclk}}{12 \times 4 \times \text{baud rate}}$
T1 mode 2	1T	$\text{reload value of T1} = 256 \div \frac{2^{\text{SMOD}} \times \text{SYSclk}}{32 \times \text{baud rate}}$
	12T	$\text{reload value of T1} = 256 \div \frac{2^{\text{SMOD}} \times \text{SYSclk}}{12 \times 32 \times \text{baud rate}}$

The reload value of the timers corresponding to the common frequency and the common baud rate are as

following.

Frequency (MHz)	Baud rate	T2		T1 mode 0		T1 mode 2			
		1T mode	12T mode	1T mode	12T mode	SMOD=1		1T mode	
						1T mode	12T mode	1T mode	12T mode
11.0592	115200	FFE8H	FFFEH	FFE8H	FFFEH	FAH	-	FDH	-
	57600	FFD0H	FFFCH	FFD0H	FFFCH	F4H	FFH	FAH	-
	38400	FFB8H	FFFAH	FFB8H	FFFAH	EEH	-	F7H	-
	19200	FF70H	FFF4H	FF70H	FFF4H	DCH	FDH	EEH	-
	9600	FEE0H	FFE8H	FEE0H	FFE8H	B8H	FAH	DCH	FDH
18.432	115200	FFD8H	-	FFD8H	-	F6H	-	FBH	-
	57600	FFB0H	-	FFB0H	-	ECH	-	F6H	-
	38400	FF88H	FFF6H	FF88H	FFF6H	E2H	-	F1H	-
	19200	FF10H	FFECH	FF10H	FFECH	C4H	FBH	E2H	-
	9600	FE20H	FFD8H	FE20H	FFD8H	88H	F6H	C4H	FBH
22.1184	115200	FFD0H	FFFCH	FFD0H	FFFCH	F4H	FFH	FAH	-
	57600	FFA0H	FFF8H	FFA0H	FFF8H	E8H	FEH	F4H	FFH
	38400	FF70H	FFF4H	FF70H	FFF4H	DCH	FDH	EEH	-
	19200	FEE0H	FFE8H	FEE0H	FFE8H	B8H	FAH	DCH	FDH
	9600	FDC0H	FFD0H	FDC0H	FFD0H	70H	F4H	B8H	FAH

14.2.7 UART1 Mode 2

If the two bits of SM0 and SM1 are '10', UART1 operates in mode 2. UART1 operating in mode 2 is a 9-bit data asynchronous communication UART. One frame of data consists of 11 bits: 1 start bit, 8 data bits (LSB first), 1 programmable bit (9th bit) and 1 stop bit. The transmitted programmable bit (9th bit) is supplied by TB8 in SCON, which can be configured as either 1 or 0 by software. Or, the odd/even parity bit P in the PSW can be loaded into TB8. Not only can TB8 be used as a multi-machine communication address/data flag, but also it can be used as datum parity check bit. The 9th bit is received into RB8 of SCON. TxD is the transmitting pin, and RxD is the receiving pin, the serial port is a full duplex receiver/transmitter.

The baud rate of mode 2 is fixed to the system clock divided by 64 or 32 depending on the value of SMOD in PCON.

The baud rate of UART1 mode 2 is shown in the following table, where SYSclk is the system operating frequency.

SMOD	Baud rate calculation formula
0	$\text{baud rate} = \frac{\text{SYSclk}}{64}$
1	$\text{baud rate} = \frac{\text{SYSclk}}{32}$

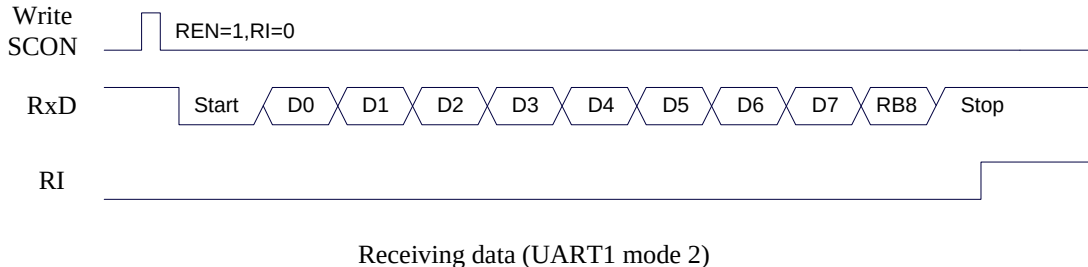
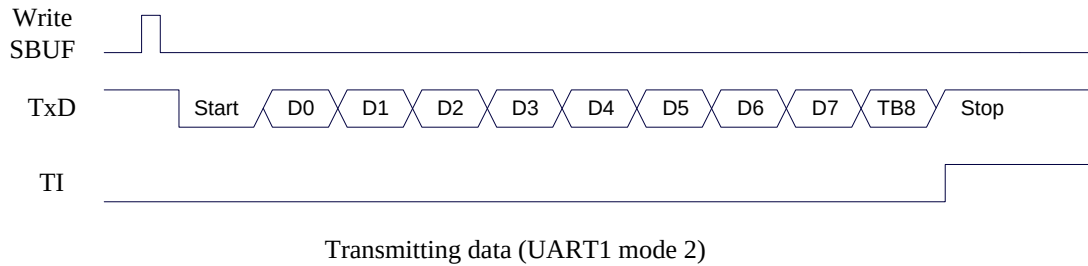
Except that the source of the baud rate is slightly different, and the 9th bit of the shift register supplied by TB8 while being sent is different, the functional and structure of mode 2 and mode 1 are basically the same, the receiving / sending operation and timing of mode 2 and mode 1 are also basically the same.

After the receiver receives a frame of information, the following conditions must be met at the same time.

- RI=0
- SM2=0 or SM2=1 and the 9th bit received RB8=1.

Only when the two conditions above are met at the same time, the data received in shift register is loaded into SBUF and RB8. The RI flag is set to 1, and the interrupt is requested to CPU. If one of the above conditions is not met, the data received in the shift register is invalid and is discarded, and RI is not set. Regardless of the above conditions are met or not, the receiver begins to detect the RxD pin hopping information again to receive the next frame of information. In mode 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.



14.2.8 UART1 Mode 3

If the two bits of SM0 and SM1 are '11', UART1 operates in mode 3. UART1 operating in mode 3 is a 9-bit data asynchronous communication UART. One frame of data consists of 11 bits: 1 start bit, 8 data bits (LSB first), 1 programmable bit (9th bit) and 1 stop bit. The transmitted programmable bit (9th bit) is supplied by TB8 in SCON, which can be configured as either 1 or 0 by software. Or, the odd/even parity bit P in the PSW can be loaded into TB8. Not only can TB8 be used as a multi-machine communication address/data flag, but also it can be used as datum parity check bit. The 9th bit is received into RB8 of SCON. TxD is the transmitting pin, and RxD is the receiving pin, the serial port is a full duplex receiver/transmitter.

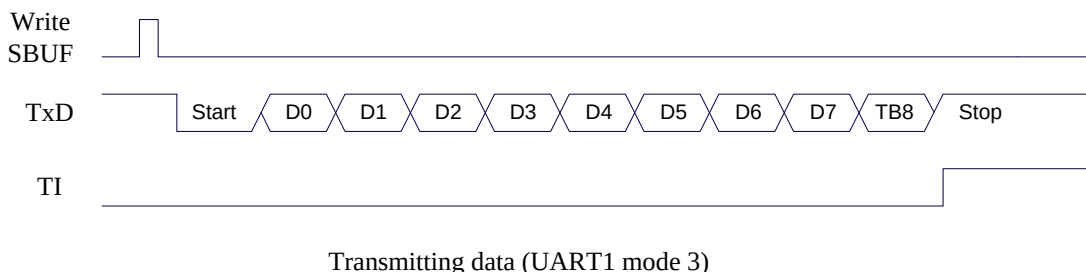
Except that the 9th bit of the shift register supplied by TB8 while being sent is different, the functional and structure of mode 3 and mode 1 are basically the same, the receiving / sending operation and timing of mode 3 and mode 1 are also basically the same.

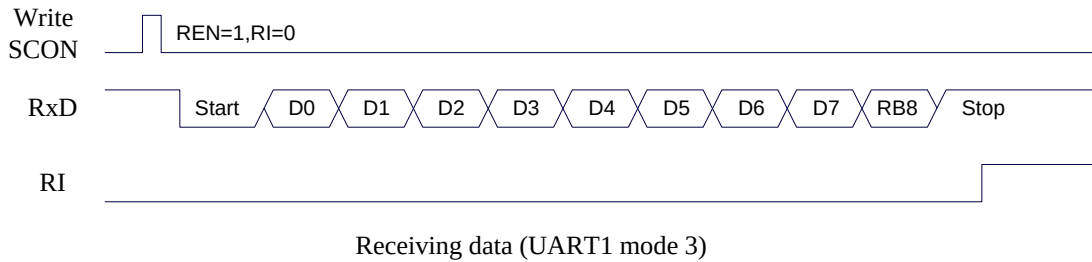
After the receiver receives a frame of information, the following conditions must be met at the same time.

- RI=0
- SM2=0 or SM2=1 and the 9th bit received RB8=1.

Only when the two conditions above are met at the same time, the data received in shift register is loaded into SBUF and RB8. The RI flag is set to 1, and the interrupt is requested to CPU. If one of the above conditions is not met, the data received in the shift register is invalid and is discarded, and RI is not set. Regardless of the above conditions are met or not, the receiver begins to detect the RxD pin hopping information again to receive the next frame of information. In mode 3, the received stop bit is not related to SBUF, RB8 and RI.

It provides for the convenience of multi-machine communication by setting SM2, TB8 of SCON and communication protocol using the software.





The baud rate calculation formula of UART1 mode 3 is exactly the same as that of mode 1. Please refer to the mode 1 baud rate calculation formula.

14.2.9 Automatic Address Recognition

14.2.10 UART1 slave address control registers

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
SADDR	A9H								
SADEN	B9H								

SADDR: Slave address register

SADEN: Slave address mask register

The automatic address recognition function is typically used in the field of multi-machine communications. Its main principle is that the slave system identifies the address information from the master serial port data stream through the hardware comparison function. The address of the slave is set by the registers SADDR and SADEN. The hardware filters the slave address automatically. The hardware will generate a serial port interrupt when the slave address information from the master matches the slave address set by the slave. Otherwise, the hardware will discard the serial port data automatically without any interruption. When a number of slaves in Idle mode are connected together, only the slave that matches the slave address will wake up from Idle mode. Then the power consumption of the slave MCU reduces greatly. Constantly entering the serial port interrupt which reduces the system execution efficiency can be avoided even if the slave is in normal operation.

To use the automatic address recognition feature of the serial port, mode 2 or mode 3 of the serial port of the MCU that participates in communication is selected. Usually the mode 3 with variable baud rate is selected because the baud rate of mode 2 is fixed, and it is inconvenient to adjust. SM2 bit of slave in SCON is set to 1. The 9th bit which is stored in RB8 is the address/data flag in mode 2 or 3. When the 9th bit is 1, it indicates the previous 8-bit datum stored in SBUF is the address information. If SM2 is set to 1, the slave MCU will filter out non-address data whose 9th bit is 0 automatically while the address data whose 9th bit is 1 in SBUF will automatically be matched with the address set in SADDR and SADEN. If the address matches, RI will be set to "1" and an interrupt will occur. Otherwise the received data is discarded.

The slave address is set by two registers, SADDR and SADEN. SADDR is the slave address register, where the slave address is stored. SADEN is the slave address mask register, which is used to set the ignore bit in the address information. The setting method is as follows.

For example

SADDR = 11001010

SADEN = 10000001

Then the matched address is 1xxxxxx0

That is, as long as bit 0 is 0 and bit 7 is 1 in the address data sent by the master, the address can be matched with the local address.

Another example

SADDR = 11001010

SADEN = 00001111

Then the matched address is xxxx1010

That is, as long as the low 4 bits are 1010 in the address data sent by the master, the address can be

matched with the local address. The high 4 bits can be any value and are ignored.

The Broadcast Address (FFH) can be used by the master to select all the slaves simultaneously for communication.

STC MCU

14.3 UART2

14.3.1 UART2 control register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
S2CON	9AH	S2SM0	-	S2SM2	S2REN	S2TB8	S2RB8	S2TI	S2RI

S2SM0: Serial port 2 mode select bit.

S2SM0	UART2 mode	Function description
0	Mode 0	8-bit UART, whose baud-rate is variable
1	Mode 1	9-bit UART, whose baud-rate is variable

S2SM2: UART2 multi-machine communication control enable bit. In mode 1, if the S2SM2 bit is 1 and the S2REN bit is 1, the receiver is in the address frame filter state. In this case, the received 9th bit (S2RB8) can be used to filter the address frame. If S2RB8 = 1, the frame is the address frame, address information can enter S2BUF, S2RI becomes 1, and then address can be compared in the interrupt service routine. If S2RB8 = 0, it indicates that the frame is not an address frame and should be discarded and keep S2RI = 0. In mode 1, if the S2SM2 bit is 0 and the S2REN bit is 1, the receiver is in the address frame filter disabled state. Regardless of the received S2RB8 is 0 or 1, the information received can enter into the S2BUF, and make S2RI = 1. Here, S2RB8 is usually used as check bit. Mode 0 is non-multi-machine communication mode, where S2SM2 should be 0.

S2REN: Receive enable control bit.

0: disable UART2 receive data.

1: enable UART2 receive data.

S2TB8: S2TB8 is the 9th bit of datum to be sent when UART2 is in mode 1, which is usually used as a parity check bit or an address frame / data frame flag. It can be set or cleared by software as required. In mode 0, this bit is not used.

S2RB8: S2RB8 is the 9th bit of datum received when UART2 is in mode 1, which is usually used as a parity check bit or an address frame / data frame flag. It can be set or cleared by software as required. In mode 0, this bit is not used.

S2TI: Transmit interrupt request flag of UART2. S2TI is set by the hardware automatically at the beginning of the stop bit transmission and requests interrupts to the CPU. S2TI must be cleared by software after the interrupt is responded.

S2RI: Receive interrupt request flag of UART2. S2RI is set by hardware automatically at the middle of stop bit received, and requests the interrupt to the CPU. After the interrupt is responded, S2RI must be cleared by software.

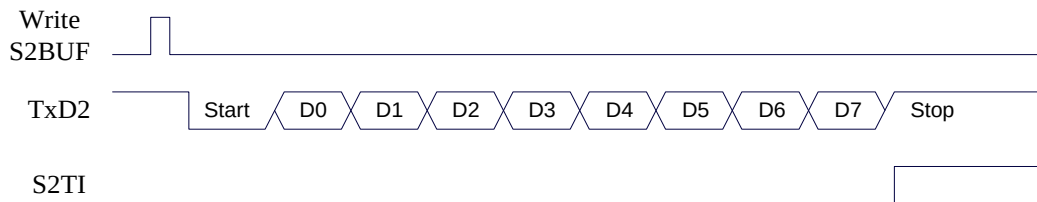
14.3.2 UART2 data register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
S2BUF	9BH								

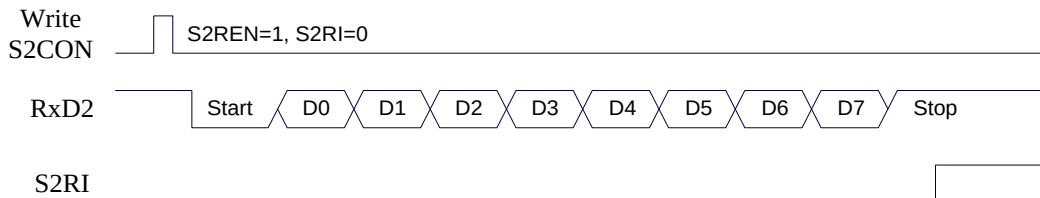
S2BUF: It is used as the buffer in transmission and receiving for UART2. S2BUF is actually two buffers, reading buffer and writing buffer. Two operations correspond to two different registers, one is write-only register (writing buffer), the other is read-only register (reading buffer). The CPU reads serial receiving buffer when reads S2BUF, and writes to the S2BUF will trigger the serial port to start sending data.

14.3.3 UART2 Mode 0

Serial port 2 mode 0 is 8-bit UART mode with variable baud rate. In this mode, a frame of data consists 10 bits: 1 start bit, 8 data bits (LSB first) and 1 stop bit. The baud rate is variable, which can be set by the software as needed. Tx2 is the data transmitting pin, and Rx2 is the data receiving pin, the serial port is a full duplex receiver/transmitter.



Transmitting data (UART2 mode 0)



Receiving data (UART2 mode 0)

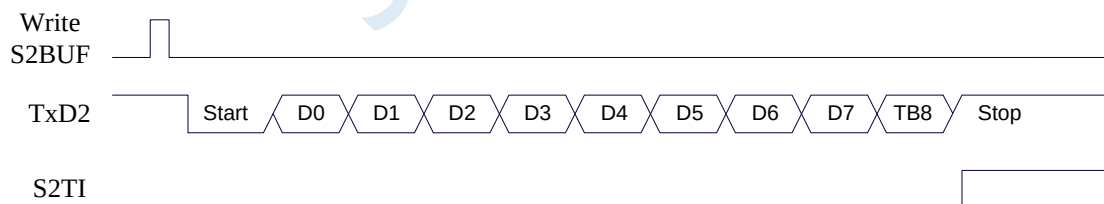
The baud rate of UART2 is variable. It is generated by T2. If the timer is in 1T mode (12x speed), the corresponding baud rate is increased by 12 times.

The baud rate of UART2 mode 0 is calculated as follows, where SYSclk is the system operating frequency.

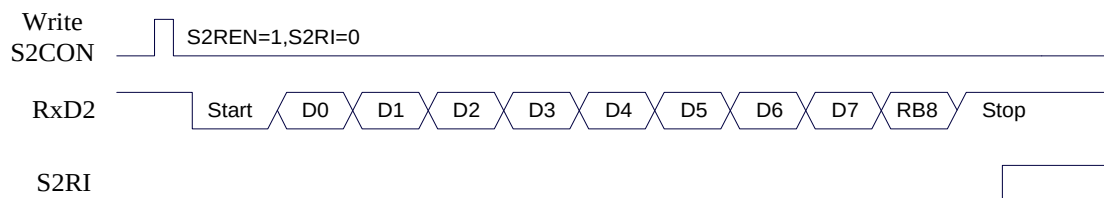
Timer selected	Speed of timer	Baud rate calculation formula
T2	1T	$\text{reload value of timer 2} = 65536 \div \frac{\text{SYSclk}}{4 \times \text{baud rate}}$
	12T	$\text{reload value of timer 2} = 65536 \div \frac{\text{SYSclk}}{12 \times 4 \times \text{baud rate}}$

14.3.4 UART2 Mode 1

UART2 operating in mode 1 is a 9-bit data UART mode with variable baud rate. One frame data consists of 11 bits: 1 start bit, 8 data bits (LSB first), 1 programmable bit (9th bit) and 1 stop bit. The baud rate is variable, which can be set by the software as needed. TxD2 is the data transmitting pin, and RxD2 is the data receiving pin, the serial port is a full duplex receiver/transmitter.



Transmitting data (UART2 mode 1)



Receiving data (UART2 mode 1)

The baud rate calculation formula of UART2 mode 1 is exactly the same as that of mode 0. Please refer to the mode 0 baud rate calculation formula.

14.4 UART3

14.4.1 UART3 control register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
S3CON	ACH	S3SM0	S3ST3	S3SM2	S3REN	S3TB8	S3RB8	S3TI	S3RI

S3SM0: UART3 mode select bit.

S3SM0	UART3 mode	Function description
0	Mode 0	8-bit UART, whose baud-rate is variable
1	Mode 1	9-bit UART, whose baud-rate is variable

S3ST3: UART3 baud rate generator select bit.

0: Select T2 as the baud-rate generator of UART3.

1: Select T3 as the baud-rate generator of UART3.

S3SM2: UART3 multi-machine communication control bit. In mode 1, if the S3SM2 bit is 1 and the S3REN bit is 1, the receiver is in the address frame filter state. In this case, the received 9th bit (S3RB8) can be used to filter the address frame. If S3RB8 = 1, the frame is the address frame, address information can enter S3BUF, S3RI becomes 1, and then the address is compared with the slave address in the interrupt service routine. If S3RB8 = 0, it indicates that the frame is not an address frame and should be discarded and keep S3RI = 0. In mode 1, if the S3SM2 bit is 0 and the S3REN bit is 1, the receiver is in the address frame filter disabled state. Regardless of the received S3RB8 is 0 or 1, the information received can enter into the S3BUF, and make S3RI = 1. Here, S3RB8 is usually used as parity check bit. Mode 0 is non-multi-machine communication mode, where S3SM2 should be 0.

S3REN: Receive enable control bit.

0: disable UART3 receive data.

1: enable UART3 receive data.

S3TB8: S3TB8 is the 9th bit of datum to be sent when UART3 is in mode 1, which is usually used as a parity check bit or an address frame / data frame flag. It can be set or cleared by software as required. In mode 0, this bit is not used.

S3RB8: S3RB8 is the 9th bit of datum received when UART3 is in mode 1, which is usually used as a parity check bit or an address frame / data frame flag. It can be set or cleared by software as required. In mode 0, this bit is not used.

S3TI: Transmit interrupt request flag of UART3. S3TI is set by the hardware automatically at the beginning of the stop bit transmission and requests interrupt to the CPU. S3TI must be cleared by software after the interrupt is responded.

S3RI: Receive interrupt request flag of UART3. S3RI is set by hardware automatically at the middle of stop bit the serial port received, and requests interrupt to the CPU. After the interrupt is responded, S3RI must be cleared by software.

14.4.2 UART3 data register

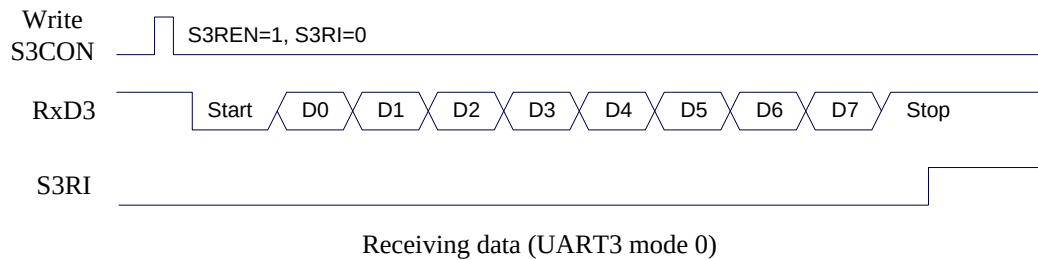
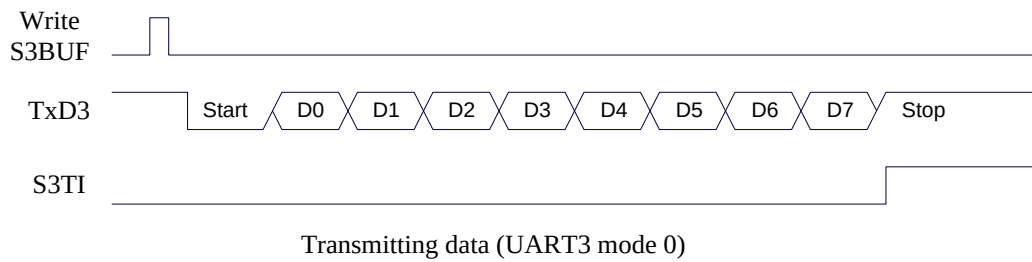
Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
S3BUF	ADH								

S3BUF: It is used as the buffer in transmission and receiving for UART3. S3BUF is actually two buffers, reading buffer and writing buffer. Two operations correspond to two different registers, one is write-only register (writing buffer), the other is read-only register (reading buffer). The CPU reads serial receive buffer when reads S3BUF, and writes to the S3BUF will trigger the serial port to start sending data.

14.4.3 UART3 Mode 0

UART3 mode 0 is a 8-bit UART mode with variable baud rate, where a frame of data consists 10 bits: 1 start bit, 8 data bits (LSB first) and 1 stop bit. The baud rate is variable, which can be set by the software as

needed. TxD3 is the data transmitting pin, and RxD3 is the data receiving pin, the serial port is a full duplex receiver/transmitter.



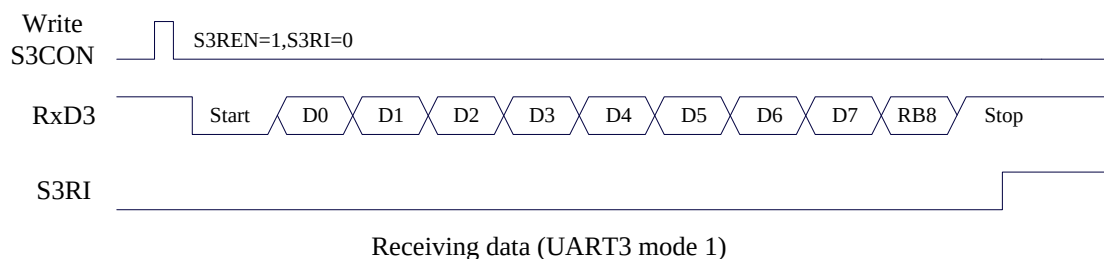
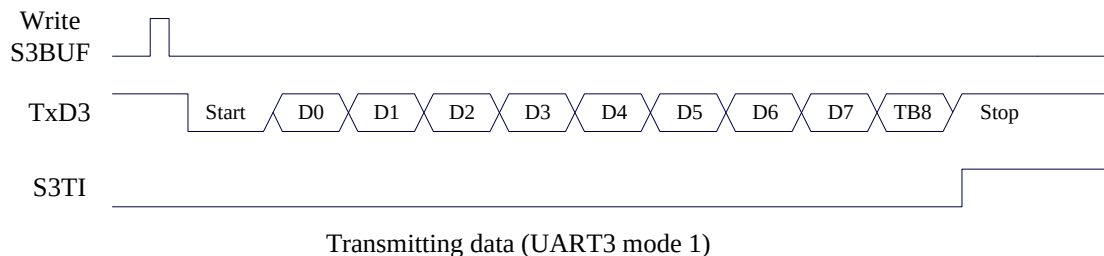
The baud rate of UART3 is variable. It is generated by T2 or T3. If the timer is in 1T mode (12x speed), the corresponding baud rate is increased by 12 times.

The baud rate of UART3 mode 0 is calculated as follows, where SYSclk is the system operating frequency.

Timer selected	Speed of timer	Baud rate calculation formula
T2	1T	$\text{reload value of T2} = 65536 \div \frac{\text{SYSclk}}{4 \times \text{baud rate}}$
	12T	$\text{reload value of T2} = 65536 \div \frac{\text{SYSclk}}{12 \times 4 \times \text{baud rate}}$
T3	1T	$\text{reload value of T3} = 65536 \div \frac{\text{SYSclk}}{4 \times \text{baud rate}}$
	12T	$\text{reload value of T3} = 65536 \div \frac{\text{SYSclk}}{12 \times 4 \times \text{baud rate}}$

14.4.4 UART3 Mode 1

UART3 operating in mode 1 is a 9-bit data UART mode with variable baud rate. One frame data consists of 11 bits: 1 start bit, 8 data bits (LSB first), 1 programmable bit (9th bit) and 1 stop bit. The baud rate is variable, which can be set by the software as needed. TxD3 is the data transmitting pin, and RxD3 is the data receiving pin, the serial port is a full duplex receiver/transmitter.



The baud rate calculation formula of UART3 mode 1 is exactly the same as that of mode 0. Please refer to the mode 0 baud rate calculation formula.

STC MCU

14.5 UART4

14.5.1 UART4 control register

Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
S4CON	84H	S4SM0	S4ST4	S4SM2	S4REN	S4TB8	S4RB8	S4TI	S4RI

S4SM0: UART4 mode select bit.

S4SM0	UART4 mode	Function description
0	Mode 0	8-bit UART, whose baud-rate is variable
1	Mode 1	9-bit UART, whose baud-rate is variable

S4ST4: UART4 baud rate generator select bit.

0: Select T2 as the baud-rate generator of UART4.

1: Select T4 as the baud-rate generator of UART4.

S4SM2: UART4 multi-machine communication control bit. In mode 1, if the S4SM2 bit is 1 and the S4REN bit is 1, the receiver is in the address frame filter state. In this case, the received 9th bit (S4RB8) can be used to filter the address frame. If S4RB8 = 1, the frame is the address frame, address information can enter S4BUF, S4RI becomes 1, and then address can be compared with the slave address in the interrupt service routine. If S4RB8 = 0, it indicates that the frame is not an address frame and should be discarded and keep S4RI = 0. In mode 1, if the S4SM2 bit is 0 and the S4REN bit is 1, the receiver is in the address frame filter disabled state. Regardless of the received S4RB8 is 0 or 1, the information received can enter into the S4BUF, and make S4RI = 1. Here, S4RB8 is usually used as parity check bit. Mode 0 is non-multi-machine communication mode, where S4SM2 should be 0.

S4REN: Receive enable control bit.

0: disable UART4 receive data.

1: enable UART4 receive data.

S4TB8: S4TB8 is the 9th bit of datum to be sent when UART4 is in mode 1, which is usually used as a parity check bit or an address frame / data frame flag. It can be set or cleared by software as required. In mode 0, this bit is not used.

S4RB8: S4RB8 is the 9th bit of datum received when UART4 is in mode 1, which is usually used as a parity check bit or an address frame / data frame flag. It can be set or cleared by software as required. In mode 0, this bit is not used.

S4TI: Transmit interrupt request flag of UART4. S4TI is set by the hardware automatically at the beginning of the stop bit transmission and requests interrupt to the CPU. S4TI must be cleared by software after the interrupt is responded.

S4RI: Receive interrupt request flag of UART4. S4RI is set by hardware automatically at the middle of stop bit the serial port received, and requests interrupt to the CPU. After the interrupt is responded, S4RI must be cleared by software.

14.5.2 UART4 data register

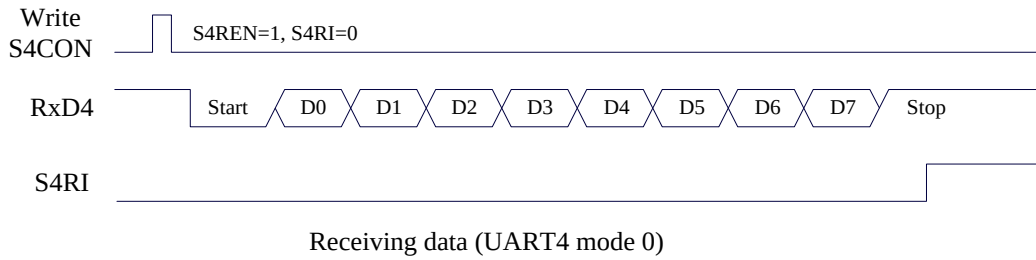
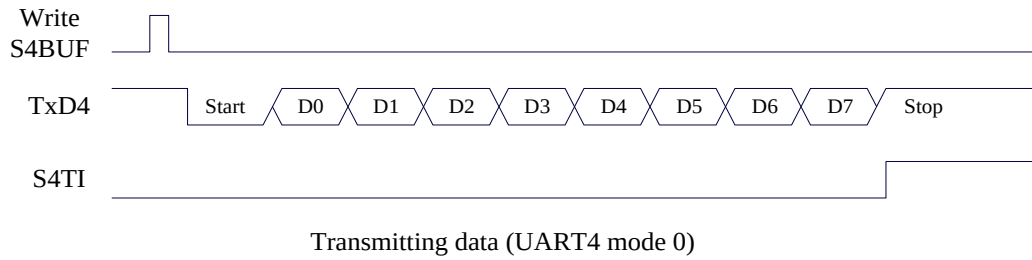
Symbol	Address	B7	B6	B5	B4	B3	B2	B1	B0
S4BUF	85H								

S4BUF: It is used as the buffer in transmission and receiving for UART4. S4BUF is actually two buffers, reading buffer and writing buffer. Two operations correspond to two different registers, one is write-only register (writing buffer), the other is read-only register (reading buffer). The CPU reads serial receive buffer when reads S4BUF, and writes to the S4BUF will trigger the serial port to start sending data.

14.5.3 UART4 Mode 0

UART4 mode 0 is an 8-bit UART mode with variable baud rate, where a frame of data consists 10 bits: 1 start bit, 8 data bits (LSB first) and 1 stop bit. The baud rate is variable, which can be set by the software as

needed. TxD4 is the data transmitting pin, and RxD4 is the data receiving pin, the serial port is a full duplex receiver/transmitter.



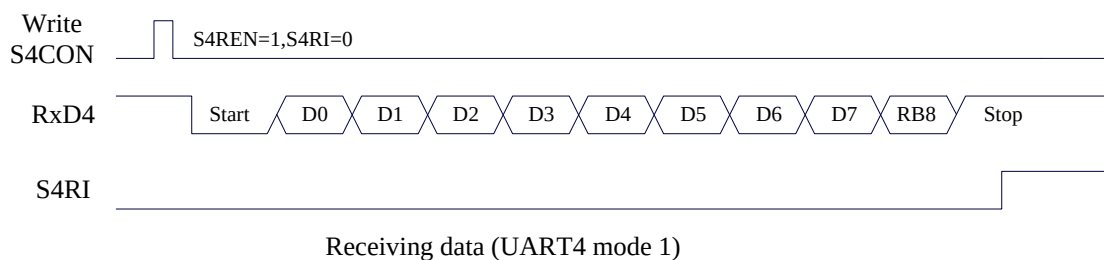
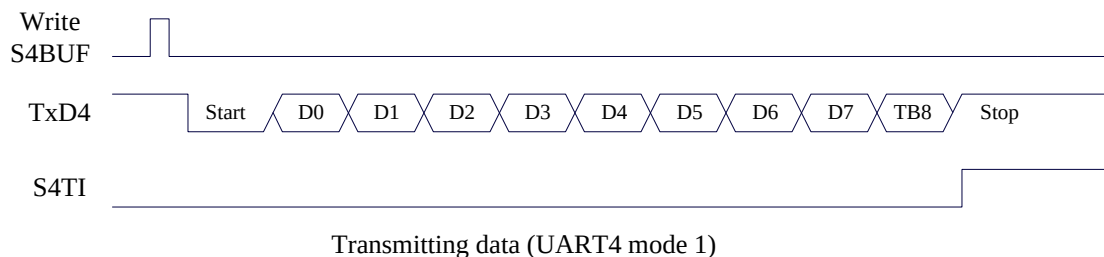
The baud rate of UART4 is variable. It is generated by T2 or T4. If the timer is in 1T mode (12x speed), the corresponding baud rate is increased by 12 times.

The baud rate of UART4 mode 0 is calculated as follows, where SYSclk is the system operating frequency.

Timer selected	Speed of timer	Baud rate calculation formula
T2	1T	$\text{reload value of T2} = 65536 \div \frac{\text{SYSclk}}{4 \times \text{baud rate}}$
	12T	$\text{reload value of T2} = 65536 \div \frac{\text{SYSclk}}{12 \times 4 \times \text{baud rate}}$
T4	1T	$\text{reload value of T4} = 65536 \div \frac{\text{SYSclk}}{4 \times \text{baud rate}}$
	12T	$\text{reload value of T4} = 65536 \div \frac{\text{SYSclk}}{12 \times 4 \times \text{baud rate}}$

14.5.4 UART4 Mode 1

UART4 operating in mode 1 is a 9-bit data UART mode with variable baud rate. One frame data consists of 11 bits: 1 start bit, 8 data bits (LSB first), 1 programmable bit (9th bit) and 1 stop bit. The baud rate is variable, which can be set by the software as needed. TxD4 is the data transmitting pin, and RxD4 is the data receiving pin, the serial port is a full duplex receiver/transmitter.



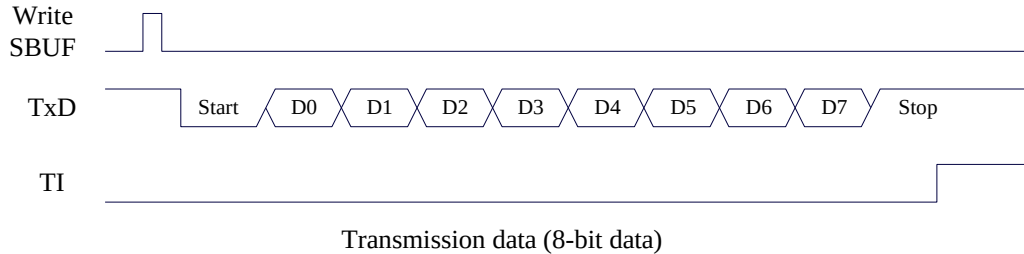
The baud rate calculation formula of UART4 mode 1 is exactly the same as that of mode 0. Please refer to

the mode 0 baud rate calculation formula.

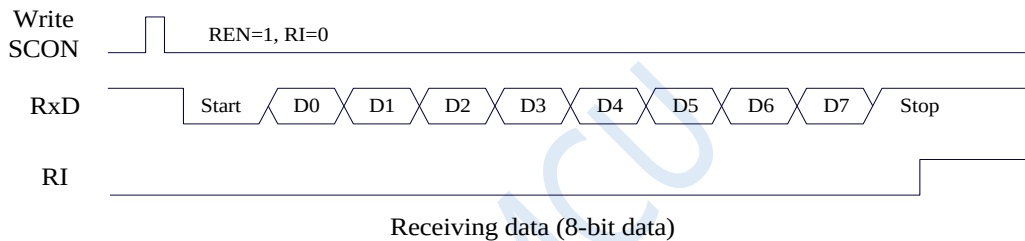
14.6 Precautions of UARTs

Regarding the UART interrupts requests, the following issues need to be noted. UART1, UART2, UART3, and UART4 are all similar, and serial port 1 is used as an example below.

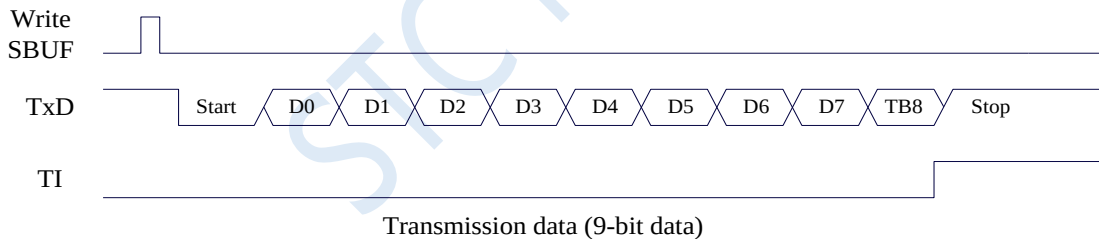
In 8-bit data mode, TI interrupt request is generated after the entire stop bit is transmitted, as shown in the following figure:



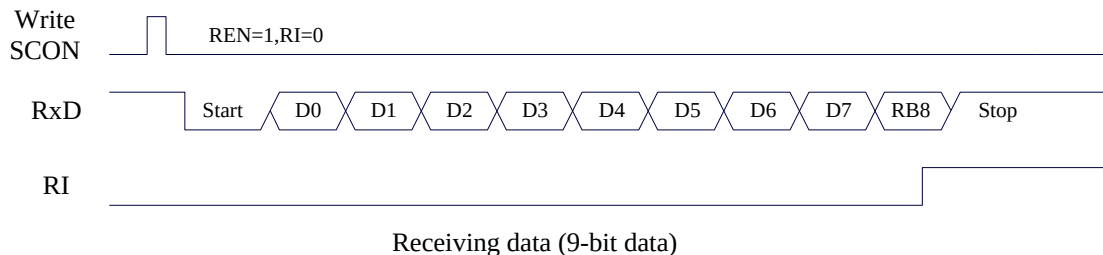
In 8-bit data mode, RI interrupt request is generated after half of the stop bit is received, as shown in the following figure:



In 9-bit data mode, TI interrupt request is generated after the entire 9th data bit is transmitted, as shown in the following figure:



In 9-bit data mode, RI interrupt request is generated after receiving half of the 9th bit, as shown in the following figure:



14.7 Example Routines

14.7.1 UART1 using T2 as baud rate generator

C language code

//Operating frequency for test is 11.0592MHz

```
#include "reg51.h"
#include "intrins.h"
```

```
#define FOSC 11059200UL
#define BRT (65536 - FOSC / 115200 / 4)
```

```
sfr AUXR = 0x8e;
sfr T2H = 0xd6;
sfr T2L = 0xd7;
```

```
sfr P0M1 = 0x93;
sfr P0M0 = 0x94;
sfr P1M1 = 0x91;
sfr P1M0 = 0x92;
sfr P2M1 = 0x95;
sfr P2M0 = 0x96;
sfr P3M1 = 0xb1;
sfr P3M0 = 0xb2;
sfr P4M1 = 0xb3;
sfr P4M0 = 0xb4;
sfr P5M1 = 0xc9;
sfr P5M0 = 0xca;
```

```
bit busy;
char wptr;
char rptr;
char buffer[16];
```

```
void UartIsr() interrupt 4
```

```
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
        buffer[wptr++] = SBUF;
        wptr &= 0x0f;
    }
}
```

```
void UartInit()
```

```
{
    SCON = 0x50;
    T2L = BRT;
    T2H = BRT >> 8;
    AUXR = 0x15;
    wptr = 0x00;
```

```
    rptr = 0x00;
    busy = 0;
}

void UartSend(char dat)
{
    while (busy);
    busy = 1;
    SBUF = dat;
}

void UartSendStr(char *p)
{
    while (*p)
    {
        UartSEND(*p++);
    }
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    UartInit();
    ES = 1;
    EA = 1;
    UartSENDStr("Uart Test !\r\n");

    while (1)
    {
        if (rptr != wptr)
        {
            UartSEND(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}
```

Assembly code

;Operating frequency for test is 11.0592MHz

AUXR	DATA	8EH
T2H	DATA	0D6H
T2L	DATA	0D7H
BUSY	BIT	20H.0
WPTR	DATA	21H
RPTR	DATA	22H

```

BUFFER      DATA      23H                                ;16 bytes

P0M1        DATA      093H
P0M0        DATA      094H
P1M1        DATA      091H
P1M0        DATA      092H
P2M1        DATA      095H
P2M0        DATA      096H
P3M1        DATA      0B1H
P3M0        DATA      0B2H
P4M1        DATA      0B3H
P4M0        DATA      0B4H
P5M1        DATA      0C9H
P5M0        DATA      0CAH

            ORG         0000H
            LJMP        MAIN
            ORG         0023H
            LJMP        UART_ISR

            ORG         0100H

UART_ISR:
            PUSH        ACC
            PUSH        PSW
            MOV         PSW,#08H

            JNB         TI,CHKRI
            CLR         TI
            CLR         BUSY

CHKRI:
            JNB         RI,UARTISR_EXIT
            CLR         RI
            MOV         A,WPTR
            ANL         A,#0FH
            ADD         A,#BUFFER
            MOV         R0,A
            MOV         @R0,SBUF
            INC         WPTR

UARTISR_EXIT:
            POP         PSW
            POP         ACC
            RETI

UART_INIT:
            MOV         SCON,#50H
            MOV         T2L,#0E8H
            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

```

;65536-11059200/115200/4=0FFE8H

```
UART_SENDSTR:
    CLR        A
    MOVC       A,@A+DPTR
    JZ         SENDEND
    LCALL      UART_SEND
    INC        DPTR
    JMP        UART_SENDSTR

SENDEND:
    RET

MAIN:
    MOV        SP,#5FH
    MOV        P0M0,#00H
    MOV        P0M1,#00H
    MOV        P1M0,#00H
    MOV        P1M1,#00H
    MOV        P2M0,#00H
    MOV        P2M1,#00H
    MOV        P3M0,#00H
    MOV        P3M1,#00H
    MOV        P4M0,#00H
    MOV        P4M1,#00H
    MOV        P5M0,#00H
    MOV        P5M1,#00H

    LCALL      UART_INIT
    SETB       ES
    SETB       EA

    MOV        DPTR,#STRING
    LCALL      UART_SENDSTR

LOOP:
    MOV        A,RPTR
    XRL        A,WPTR
    ANL        A,#0FH
    JZ         LOOP
    MOV        A,RPTR
    ANL        A,#0FH
    ADD        A,#BUFFER
    MOV        R0,A
    MOV        A,@R0
    LCALL      UART_SEND
    INC        RPTR
    JMP        LOOP

STRING:      DB      'Uart Test !',0DH,0AH,00H

    END
```

14.7.2 UART1 using T1 (Mode 0) as baud rate generator

C language code

//Operating frequency for test is 11.0592MHz

#include "reg51.h"

```
#include "intrins.h"
```

```
#define FOSC      11059200UL
#define BRT      (65536 - FOSC / 115200 / 4)
```

```
sfr      AUXR      = 0x8e;
```

```
sfr      P0M1      = 0x93;
```

```
sfr      P0M0      = 0x94;
```

```
sfr      P1M1      = 0x91;
```

```
sfr      P1M0      = 0x92;
```

```
sfr      P2M1      = 0x95;
```

```
sfr      P2M0      = 0x96;
```

```
sfr      P3M1      = 0xb1;
```

```
sfr      P3M0      = 0xb2;
```

```
sfr      P4M1      = 0xb3;
```

```
sfr      P4M0      = 0xb4;
```

```
sfr      P5M1      = 0xc9;
```

```
sfr      P5M0      = 0xca;
```

```
bit      busy;
```

```
char     wptr;
```

```
char     rptr;
```

```
char     buffer[16];
```

```
void UartIsr() interrupt 4
```

```
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
        buffer[wptr++] = SBUF;
        wptr &= 0x0f;
    }
}
```

```
void UartInit()
```

```
{
    SCON = 0x50;
    TMOD = 0x00;
    TL1 = BRT;
    TH1 = BRT >> 8;
    TR1 = 1;
    AUXR = 0x40;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}
```

```
void UartSend(char dat)
```

```
{
    while (busy);
    busy = 1;
    SBUF = dat;
}
```

```
void UartSendStr(char *p)
{
    while (*p)
    {
        UartSEND(*p++);
    }
}

void main()
{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    UartInit();
    ES = 1;
    EA = 1;
    UartSENDStr("Uart Test !\r\n");

    while (1)
    {
        if (rptr != wptr)
        {
            UartSEND(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}
```

Assembly code

;Operating frequency for test is 11.0592MHz

AUXR	DATA	8EH	
BUSY	BIT	20H.0	
WPTR	DATA	21H	
RPTR	DATA	22H	
BUFFER	DATA	23H	;16 bytes
P0M1	DATA	093H	
P0M0	DATA	094H	
P1M1	DATA	091H	
P1M0	DATA	092H	
P2M1	DATA	095H	
P2M0	DATA	096H	
P3M1	DATA	0B1H	
P3M0	DATA	0B2H	
P4M1	DATA	0B3H	
P4M0	DATA	0B4H	

```

P5M1      DATA      0C9H
P5M0      DATA      0CAH

          ORG          0000H
          LJMP         MAIN
          ORG          0023H
          LJMP         UART_ISR

          ORG          0100H

UART_ISR:
          PUSH         ACC
          PUSH         PSW
          MOV          PSW,#08H

          JNB          TI,CHKRI
          CLR          TI
          CLR          BUSY

CHKRI:
          JNB          RI,UARTISR_EXIT
          CLR          RI
          MOV          A,WPTR
          ANL          A,#0FH
          ADD          A,#BUFFER
          MOV          R0,A
          MOV          @R0,SBUF
          INC          WPTR

UARTISR_EXIT:
          POP          PSW
          POP          ACC
          RETI

UART_INIT:
          MOV          SCON,#50H
          MOV          TMOD,#00H
          MOV          TL1,#0E8H
          MOV          TH1,#0FFH
          SETB         TR1
          MOV          AUXR,#40H
          CLR          BUSY
          MOV          WPTR,#00H
          MOV          RPTR,#00H
          RET

UART_SEND:
          JB           BUSY,$
          SETB         BUSY
          MOV          SBUF,A
          RET

UART_SENDSTR:
          CLR          A
          MOVC         A,@A+DPTR
          JZ           SENDEND
          LCALL        UART_SEND
          INC          DPTR
          JMP          UART_SENDSTR

SENDEND:
          RET

```

;65536-11059200/115200/4=0FFE8H

MAIN:

```
MOV     SP, #5FH
MOV     P0M0, #00H
MOV     P0M1, #00H
MOV     P1M0, #00H
MOV     P1M1, #00H
MOV     P2M0, #00H
MOV     P2M1, #00H
MOV     P3M0, #00H
MOV     P3M1, #00H
MOV     P4M0, #00H
MOV     P4M1, #00H
MOV     P5M0, #00H
MOV     P5M1, #00H

LCALL   UART_INIT
SETB    ES
SETB    EA

MOV     DPTR, #STRING
LCALL   UART_SENDSTR
```

LOOP:

```
MOV     A, RPTR
XRL     A, WPTR
ANL     A, #0FH
JZ      LOOP
MOV     A, RPTR
ANL     A, #0FH
ADD     A, #BUFFER
MOV     R0, A
MOV     A, @R0
LCALL   UART_SEND
INC     RPTR
JMP     LOOP
```

```
STRING:  DB      'Uart Test !', 0DH, 0AH, 00H
```

```
END
```

14.7.3 UART1 using T1 (Mode 2) as baud rate generator

C language code

//Operating frequency for test is 11.0592MHz

```
#include "reg51.h"
#include "intrins.h"
```

```
#define FOSC      11059200UL
#define BRT       (256 - FOSC / 115200 / 32)
```

```
sfr      AUXR      = 0x8e;
```

```
sfr      P0M1      = 0x93;
```

```
sfr      P0M0      = 0x94;
```

```
sfr      P1M1      = 0x91;
```

```
sfr      P1M0      = 0x92;
sfr      P2M1      = 0x95;
sfr      P2M0      = 0x96;
sfr      P3M1      = 0xb1;
sfr      P3M0      = 0xb2;
sfr      P4M1      = 0xb3;
sfr      P4M0      = 0xb4;
sfr      P5M1      = 0xc9;
sfr      P5M0      = 0xca;
```

```
bit      busy;
char     wptr;
char     rptr;
char     buffer[16];
```

```
void UartIsr() interrupt 4
```

```
{
    if (TI)
    {
        TI = 0;
        busy = 0;
    }
    if (RI)
    {
        RI = 0;
        buffer[wptr++] = SBUF;
        wptr &= 0x0f;
    }
}
```

```
void UartInit()
```

```
{
    SCON = 0x50;
    TMOD = 0x20;
    TL1 = BRT;
    TH1 = BRT;
    TR1 = 1;
    AUXR = 0x40;
    wptr = 0x00;
    rptr = 0x00;
    busy = 0;
}
```

```
void UartSend(char dat)
```

```
{
    while (busy);
    busy = 1;
    SBUF = dat;
}
```

```
void UartSendStr(char *p)
```

```
{
    while (*p)
    {
        UartSEND(*p++);
    }
}
```

```
void main()
```

```

{
    P0M0 = 0x00;
    P0M1 = 0x00;
    P1M0 = 0x00;
    P1M1 = 0x00;
    P2M0 = 0x00;
    P2M1 = 0x00;
    P3M0 = 0x00;
    P3M1 = 0x00;
    P4M0 = 0x00;
    P4M1 = 0x00;
    P5M0 = 0x00;
    P5M1 = 0x00;

    UartInit();
    ES = 1;
    EA = 1;
    UartSENDStr("Uart Test !\r\n");

    while (1)
    {
        if (rptr != wptr)
        {
            UartSEND(buffer[rptr++]);
            rptr &= 0x0f;
        }
    }
}

```

Assembly code

;Operating frequency for test is 11.0592MHz

AUXR	DATA	8EH	
BUSY	BIT	20H.0	
WPTR	DATA	21H	
RPTR	DATA	22H	
BUFFER	DATA	23H	;16 bytes
P0M1	DATA	093H	
P0M0	DATA	094H	
P1M1	DATA	091H	
P1M0	DATA	092H	
P2M1	DATA	095H	
P2M0	DATA	096H	
P3M1	DATA	0B1H	
P3M0	DATA	0B2H	
P4M1	DATA	0B3H	
P4M0	DATA	0B4H	
P5M1	DATA	0C9H	
P5M0	DATA	0CAH	
	ORG	0000H	
	LJMP	MAIN	
	ORG	0023H	
	LJMP	UART_ISR	
	ORG	0100H	

UART_ISR:

PUSH ACC
PUSH PSW
MOV PSW,#08H

JNB TI,CHKRI
CLR TI
CLR BUSY

CHKRI:

JNB RI,UARTISR_EXIT
CLR RI
MOV A,WPTR
ANL A,#0FH
ADD A,#BUFFER
MOV R0,A
MOV @R0,SBUF
INC WPTR

UARTISR_EXIT:

POP PSW
POP ACC
RETI

UART_INIT:

MOV SCON,#50H
MOV TMOD,#20H
MOV TL1,#0FDH ;256-11059200/115200/32=0FDH
MOV TH1,#0FDH
SETB TR1
MOV AUXR,#40H
CLR BUSY
MOV WPTR,#00H
MOV RPTR,#00H
RET

UART_SEND:

JB BUSY,\$
SETB BUSY
MOV SBUF,A
RET

UART_SENDSTR:

CLR A
MOVC A,@A+DPTR
JZ SENDEND
LCALL UART_SEND
INC DPTR
JMP UART_SENDSTR

SENDEND:

RET

MAIN:

MOV SP,#5FH
MOV P0M0,#00H
MOV P0M1,#00H
MOV P1M0,#00H
MOV P1M1,#00H
MOV P2M0,#00H
MOV P2M1,#00H
MOV P3M0,#00H