S3F84UA/F84U8

8-BIT CMOS MCU

Revision 1.10

August 2009

用户手册

SAMSUNG ELECTRONICS

SAMSUNG

Important Notice

The information in this publication has been carefully checked and is believed to be entirely accurate at the time of publication. Samsung assumes no responsibility, however, for possible errors or omissions, or for any consequences resulting from the use of the information contained herein.

Samsung reserves the right to make changes in its products or product specifications with the intent to improve function or design at any time and without notice and is not required to update this documentation to reflect such changes.

This publication does not convey to a purchaser of semiconductor devices described herein any license under the patent rights of Samsung or others.

Samsung makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Samsung assume any liability arising out of the application or use of any product or circuit and specifically disclaims any and all liability, including without limitation any consequential or incidental damages.

"Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by the customer's technical experts.

Samsung products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, for other applications intended to support or sustain life, or for any other application in which the failure of the Samsung product could create a situation where personal injury or death may occur.

Should the Buyer purchase or use a Samsung product for any such unintended or unauthorized application, the Buyer shall indemnify and hold Samsung and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, expenses, and reasonable attorney fees arising out of, either directly or indirectly, any claim of personal injury or death that may be associated with such unintended or unauthorized use, even if such claim alleges that Samsung was negligent regarding the design or manufacture of said product.

S3F84UA/F84U8 8-BIT CMOS MCU
用户手册, Revision 1.10

Samsung Electronics Co., Ltd.
San #24 Nongseo-Dong, Giheung-Gu
Yongin-City, Gyeonggi-Do, Korea 446-711

TEL : (82)-(31)-209-3107
FAX : (82)-(31)-209-3262

Home Page: *http://www.samsungsemi.com*

Printed in the Republic of Korea

Revision History

Revision No.	Date	Description	Author(s)
0.00	2008年6月	仅供内部参考 spec	SM. Lee
1.00	2008年7月	第一版	SM. Lee
1.10	2009年7月	第二版	SM. Lee
1.10	2009年12月	中文版第一版 1.10	Hui. Xu

Table of Contents

1 产品概述 1-1

1.1 S3C8- 系列 MCU 1-1
1.2 S3F84UA/F84U8 MCU 1-1
1.3 特性 1-2
1.3.1 CPU 1-2
1.3.2 存储器 1-2
1.3.3 指令集 1-2
1.3.4 34 个 I/O 管脚 1-2
1.3.5 中断 1-2
1.3.6 8-位 Basic Timer 1-3
1.3.7 8 位 Timer/Counter A 1-3
1.3.8 8 位 Timer/Counter B 1-3
1.3.9 8 位 Timer/Counter C 1-3
1.3.10 两个 16 位 Timer/Counter (D0/D1) 1-3
1.3.11 钟表定时器 1-3
1.3.12 LCD 控制器/驱动器 1-3
1.3.13 模拟数字转换器 1-3
1.3.14 两个通道 UART 1-4
1.3.15 8 位串行 I/O 接口 1-4
1.3.16 Pattern Generation 模块 1-4
1.3.17 低电压复位 (LVR) 1-4
1.3.18 两种省电模式 1-4
1.3.19 时钟源 1-4
1.3.20 指令执行时间 1-4
1.3.21 工作电压范围 1-4
1.3.22 工作温度范围 1-5
1.3.23 封装类型 1-5
1.3.24 IVC 1-5
1.3.25 Smart Option 1-5
1.4 模块框图 1-6
1.5 管脚分布图 1-7
1.6 管脚描述 1-9
1.7 管脚电路 1-12

2 地址空间 2-1

2.1 概述 2-1
2.2 程序存储空间 (ROM) 2-2
2.2.1 Smart Option 2-3
2.3 寄存器结构 2-4
2.3.1 寄存器页 (PP) 2-7
2.3.2 寄存器 SET 1 2-9
2.3.3 寄存器 SET 2 2-9
2.3.4 主寄存器空间 2-10

2.3.5 工作寄存器 2-11
2.3.6 使用寄存器指针 2-12
2.4 寄存器寻址 2-14
2.4.1 通用工作寄存器区 (C0H–CFH) 2-16
2.4.2 4 位工作寄存器寻址模式 2-18
2.4.3 8 位工作寄存器寻址模式 2-20
2.5 系统和用户堆栈 2-22
2.5.1 栈操作 2-22
2.5.2 用户自定义栈 2-22
2.5.2.1 栈指针 (SPL, SPH) 2-22

3 寻址模式 3-1

3.1 概述 3-1
3.1.1 寄存器寻址模式 (R) 3-2
3.1.2 间接寄存器寻址模式 (IR) 3-3
3.1.2.1 间接寄存器寻址模式 (续) 3-4
3.1.2.2 间接寄存器寻址模式 (续) 3-5
3.1.2.3 间接寄存器寻址模式 (续) 3-6
3.1.3 偏址寻址模式 (X) 3-7
3.1.3.1 偏址寻址模式 (续) 3-8
3.1.3.2 偏址寻址模式 (续) 3-9
3.1.4 直接寻址模式 (DA) 3-10
3.1.4.1 直接寻址模式 (续) 3-11
3.1.5 间接寻址模式 (IA) 3-12
3.1.6 相对地址寻址模式 (RA) 3-13
3.1.7 立即数寻址模式 (IM) 3-14

4 控制寄存器 4-1

4.1 概述 4-1
4.1.1 ADCON—A/D 转换控制寄存器: D2H Set 1, Bank 0 4-5
4.1.2 BTCON—Basic Timer 控制寄存器: D3H Set 1 4-6
4.1.3 CLKCON—系统时钟控制寄存器: D4H Set 1 4-7
4.1.4 FLAGS—系统标志寄存器: D5H Set 1 4-8
4.1.5 FMCON—闪存控制寄存器: F9H Set 1, Bank 0 4-9
4.1.6 FMSECH—闪存扇区地址寄存器 (高字节): F6H Set 1, Bank 0 4-10
4.1.7 FMSECL—闪存扇区地址寄存器 (低字节): F7H Set 1, Bank 0 4-10
4.1.8 FMUSR—闪存用户可编程使能寄存器: F8H Set 1, Bank 0 4-10
4.1.9 IMR—中断屏蔽寄存器: DDH Set 1 4-11
4.1.10 INTPND—中断标志位寄存器: FBH Set 1, Bank 0 4-12
4.1.11 IPH—指令指针 (高字节): DAH Set 1 4-13
4.1.12 IPL—指令指针 (低字节): DBH Set 1 4-13
4.1.13 IPR—中断优先级寄存器: FFH Set 1, Bank 0 4-14
4.1.14 IRQ—中断请求寄存器: DCH Set 1 4-15
4.1.15 LCON—LCD 控制寄存器: F5H Set 1, Bank 1 4-16
4.1.16 OSCCON—时钟控制寄存器: FAH Set 1, Bank 0 4-17
4.1.17 P0CONH—P0 口控制寄存器 (高字节): D0H Set 1, Bank 1 4-18
4.1.18 P0CONL—P0 口控制寄存器 (低字节): D1H Set 1, Bank 1 4-19

4.1.19 P0PUR—P0 口上拉电阻使能控制寄存器: D2H Set 1, Bank 1 4-20
4.1.20 P1CON—P1 口控制寄存器: E2H Set 1, Bank 1 4-21
4.1.21 P2CONH—P2 口控制寄存器 (高字节): E0H Set 1, Bank 1 4-22
4.1.22 P2CONL—P2 口控制寄存器 (低字节): E1H Set 1, Bank 1 4-23
4.1.23 P3CONH—P3 口控制寄存器 (高字节): E4H Set 1, Bank 1 4-24
4.1.24 P3CONL—P3 口控制寄存器 (低字节): E5H Set 1, Bank 1 4-25
4.1.25 P3INTH—P3 口中断控制寄存器 (高字节): E6H Set 1, Bank 1 4-26
4.1.26 P3INTL—P3 口中断控制寄存器 (低字节): E7H Set 1, Bank 1 4-27
4.1.27 P3PND—P3 口中断标志寄存器: E8H Set 1, Bank 1 4-28
4.1.28 P3PUR—P3 口上拉电阻使能控制寄存器: E9H Set 1, Bank 1 4-29
4.1.29 PNE3—P3 口 N 沟道开漏模式寄存器: E3H Set 1, Bank 1 4-30
4.1.30 P4CONH—P4 口控制寄存器 (高字节): EAH Set 1, Bank 1 4-31
4.1.31 P4CONL—P4 控制寄存器 (低字节): EBH Set 1, Bank 1 4-32
4.1.32 P4PUR—P4 上拉电阻使能控制寄存器: ECH Set 1, Bank 1 4-33
4.1.33 PNE4—P4 口 N 沟道开漏模式寄存器: EDH Set 1, Bank 1 4-34
4.1.34 PGCON—Pattern Generation 模块控制寄存器: EEH Set 1, Bank 1 4-35
4.1.35 PP—寄存器页指针: DFH Set 1 4-36
4.1.36 RP0—寄存器指针 0: D6H Set 1 4-37
4.1.37 RP1—寄存器指针 1: D7H Set 1 4-37
4.1.38 SIOCON—SIO 控制寄存器: E7H Set 1, Bank 0 4-38
4.1.39 SPH—堆栈指针 (高字节): D8H Set 1 4-39
4.1.40 SPL—Stack Pointer (Low Byte): D9H Set 1 4-39
4.1.41 STPCON—STOP 控制寄存器: EDH Set 1, Bank 0 4-39
4.1.42 SYM—系统模式寄存器: DEH Set 1 4-40
4.1.43 TACON—Timer A 控制寄存器: E2H Set 1, Bank 0 4-41
4.1.44 TBCON—Timer B 控制寄存器: E3H Set 1, Bank 0 4-42
4.1.45 TCCON—Timer C0 控制寄存器: ECH Set 1, Bank 0 4-43
4.1.46 TD0CON—Timer D0 控制寄存器: FAH Set 1, Bank 1 4-44
4.1.47 TD1CON—Timer D1 控制寄存器: FBH Set 1, Bank 1 4-45
4.1.48 UART0CONH—UART 0 控制寄存器 (高字节): EEH Set 1, Bank 0 4-46
4.1.49 UART0CONL—UART 0 控制寄存器 (低字节): EFH Set 1, Bank 0 4-47
4.1.50 UART1CONH—UART 1 控制寄存器 (高字节): F2H Set 1, Bank 0 4-48
4.1.51 UART1CONL—UART 1 控制寄存器 (低字节): F3H Set 1, Bank 0 4-49
4.1.52 WTCON—钟表定时器控制寄存器: E6H Set 1, Bank 0 4-50

5 中断结构 5-1

5.1 概述 5-1
5.1.1 中断级 (Levels) 5-1
5.1.2 中断向量 (Vectors) 5-1
5.1.3 中断源 (Sources) 5-1
5.1.4 中断类型 5-2
5.1.5 S3F84UA/F84U8 中断结构 5-3
5.1.5.1 中断向量地址 5-5
5.1.5.2 使能/禁止中断指令 (EI, DI) 5-7
5.1.5.3 系统级中断控制寄存器 5-7
5.1.6 中断处理控制要点 5-8
5.1.7 外围中断控制寄存器 5-9
5.1.8 系统模式寄存器 (SYM) 5-11

5.1.9 中断屏蔽寄存器 (IMR) 5-12
5.1.10 中断优先级寄存器 (IPR) 5-13
5.1.11 中断请求寄存器 (IRQ) 5-15
5.1.12 中断标志位类型 5-16
5.1.12.1 概述 5-16
5.1.12.2 硬件自动清零的标志位 5-16
5.1.12.3 需要在中断服务程序里(软件)清零的标志位 5-16
5.1.13 中断源响应步骤 5-17
5.1.14 中断服务程序 5-17
5.1.15 中断向量地址的生成 5-18
5.1.16 向量化的中断嵌套 5-18
5.1.17 指令指针 (IP) 5-18
5.1.18 快速中断处理 5-18
5.1.18.1 快速中断处理 (续) 5-19
5.1.18.2 快速中断处理的初始化 5-19
5.1.18.3 快速中断服务程序 5-19
5.1.19 中断标志位类型之间的关系 5-19
5.1.20 编程指导 5-19

6 指令集 6-1

6.1 概述 6-1
6.1.1 数据类型 6-1
6.1.2 寄存器访问 6-1
6.1.3 寻址模式 6-1
6.2 标志寄存器(FLAGS) 6-5
6.2.1 标志位描述 6-6
6.2.2 指令集符号 6-7
6.3 条件码 6-11
6.3.1 指令集描述 6-12
6.3.1.1 ADC—带进位加法 (Add with Carry) 6-13
6.3.1.2 ADD—加法 (Add) 6-14
6.3.1.3 AND—逻辑与 (Logical AND) 6-15
6.3.1.4 BAND—位与 (Bit AND) 6-16
6.3.1.5 BCP—位比较 (Bit Compare) 6-17
6.3.1.6 BITC—位反 (Bit Complement) 6-18
6.3.1.7 BITR—位清零 (Bit Reset) 6-19
6.3.1.8 BITS—位置1 (Bit Set) 6-20
6.3.1.9 BOR—位或 (Bit OR) 6-21
6.3.1.10 BTJRF—位测试，若为假相对跳转 (Bit Test, Jump Relative on False) 6-22
6.3.1.11 BTJRT—位测试，若为真，相对跳转 (Bit Test, Jump Relative on True) 6-23
6.3.1.12 BXOR — 位异或 (Bit XOR) 6-24
6.3.1.13 CALL—程序调用 (Call Procedure) 6-25
6.3.1.14 CCF—进位标志位取反 (Complement Carry Flag) 6-26
6.3.1.15 CLR—清零 (Clear) 6-27
6.3.1.16 COM—取反 (complement) 6-28
6.3.1.17 CP—比较 (Compare) 6-29
6.3.1.18 CPIJE—比较，增加一，若相等跳转 (Compare, Increment, and Jump on Equal) 6-30
6.3.1.19 CPIJNE—比较，增加一，不等跳转 (Compare, Increment, Jump on Non-Equal) 6-31

6.3.1.20 DA — 十进制调整 (Decimal Adjust) 6-32
6.3.1.21 DA—十进制调整 (Decimal Adjust) 6-33
6.3.1.22 DEC—字节减 1 (Decrement) 6-34
6.3.1.23 DECW—字减 1 (Decrement Word) 6-35
6.3.1.24 DI—屏蔽全局中断 (Disable Interrupts) 6-36
6.3.1.25 DIV—无符号除法 (Unsigned Divide) 6-37
6.3.1.26 DJNZ—减 1，如果非零，跳转 (Decrement and Jump if Non-Zero) 6-38
6.3.1.27 EI—使能全局中断 (Enable Interrupts) 6-39
6.3.1.28 ENTER—进入 (Enter) 6-40
6.3.1.29 EXIT—退出 (Exit) 6-41
6.3.1.30 IDLE—空闲指令 (Idle Operation) 6-42
6.3.1.31 INC—加1 (Increment) 6-43
6.3.1.32 INCW—字加 1 (Increment Word) 6-44
6.3.1.33 IRET—中断返回 (Interrupt Return) 6-45
6.3.1.34 JP—跳转 (Jump) 6-46
6.3.1.35 JR—相对跳转指令 (Jump Relative) 6-47
6.3.1.36 LD—传送数据 (Load) 6-48
6.3.1.37 LD—传送数据 (Load) 6-49
6.3.1.38 LDB—传送位数据 (Load Bit) 6-50
6.3.1.39 LDC/LDE—传送程序/外部数据存储器数据 (Load Memory) 6-51
6.3.1.40 LDC/LDE—传送程序/外部数据存储器数据 (Load Memory) 6-52
6.3.1.41 LDCD/LDED—传送数据之后地址减1 (Load Memory and Decrement) 6-53
6.3.1.42 LDCI/LDEI—传送数据后地址加 1 (Load Memory and Increment) 6-54
6.3.1.43 LDCPD/LDEPD—传送数据前地址减 1 (Load Memory with Pre-Decrement) 6-55
6.3.1.44 LDCPI/LDEPI—传送数据前地址加 1 (Load Memory with Pre-Increment) 6-56
6.3.1.45 LDW—传送字数据 (Load Word) 6-57
6.3.1.46 MULT—无符号数乘法 (Unisigned Multiply) 6-58
6.3.1.47 NEXT—Next 指令 6-59
6.3.1.48 NOP—空操作 (No Operation) 6-60
6.3.1.49 OR—逻辑或 (Logical OR) 6-61
6.3.1.50 POP—出栈 (Pop from Stack) 6-62
6.3.1.51 POPUD—弹出用户栈 (自减) (Pop User Stack (Drementing)) 6-63
6.3.1.52 POPUI—弹出用户栈 (自增) (Pop User Stack (Incrementing)) 6-64
6.3.1.53 PUSH—压栈 (Push to Stack) 6-65
6.3.1.54 PUSHUD—压入用户栈 (自减) Push User Stack (Decrementing) 6-66
6.3.1.55 PUSHUI—压入用户栈 (自增) Push User Stack (Incrementing) 6-67
6.3.1.56 RCF—C清0 (Reset Carry Flag) 6-68
6.3.1.57 RET—子程序返回 (Return) 6-69
6.3.1.58 RL—左移 (Rotate Left) 6-70
6.3.1.59 RLC—带进位左移 (Rotate Left Through Carry) 6-71
6.3.1.60 RR—右移 (Rotate Right) 6-72
6.3.1.61 RRC—带进位右移 (Rotate Right Through Carry) 6-73
6.3.1.62 SB0—选择 Bank 0 (Select Bank 0) 6-74
6.3.1.63 SB1—选择 Bank 1 (Select Bank 1) 6-75
6.3.1.64 SBC—带进位减法 (Subtract with Carry) 6-76
6.3.1.65 SCF—C 置 1 (Set Carry Flag) 6-77
6.3.1.66 SRA—算术右移 (Shift Right Arithmetic) 6-78
6.3.1.67 SRP/SRP0/SRP1—设置寄存器指针 (Set Register Pointer) 6-79

6.3.1.68 STOP—Stop 操作 (Stop Operation) 6-80
6.3.1.69 SUB—减法 (Subtract) 6-81
6.3.1.70 SWAP—交换 (Swap Nibbles) 6-82
6.3.1.71 TCM—取反后位测试 (Test Complement Under Mask) 6-83
6.3.1.72 TM—位测试 (Test Under Mask) 6-84
6.3.1.73 WFI—等待中断 (Wait for Interrupt) 6-85
6.3.1.74 XOR—逻辑异或 (Logical Exclusive OR) 6-86

7 时钟电路 7-1

7.1 概述 7-1
7.1.1 系统时钟电路 7-1
7.1.2 CPU 时钟符号 7-1
7.1.3 主振荡电路 7-2
7.1.4 副振荡H电路 7-3
7.1.5 省电模式下的时钟电路状态 7-4
7.1.6 系统时钟控制寄存器 (CLKCON) 7-5
7.1.7 时钟控制寄存器 (OSCCON) 7-6
7.1.8 STOP 控制寄存器 (STPCON) 7-7
7.1.9 切换 CPU 时钟 7-8

8 复位和省电模式 8-1

8.1 系统复位 8-1
8.1.1 概述 8-1
8.1.2 正常模式下的复位操作 8-1
8.1.3 硬件复位值 8-2
8.2 省电模式 8-6
8.2.1 STOP 模式 8-6
8.2.1.1 使用复位操作退出 STOP 模式 8-6
8.2.1.2 使用外部中断退出 STOP 模式 8-6
8.2.1.3 使用内部中断退出 STOP 模式 8-7
8.2.1.4 IDLE 模式 8-7

9 I/O 口 9-1

9.1 概述 9-1
9.2 端口数据寄存器 9-2
9.2.1 P0 口 9-3
9.2.1.1 P0 口控制寄存器 (P0CONH, P0CONL) 9-3
9.2.1.2 P0 口上拉电阻使能控制寄存器 (P0PUR) 9-3
9.2.2 P1 口 9-5
9.2.2.1 P1 口控制寄存器 (P1CON) 9-5
9.2.3 P2 口 9-6
9.2.3.1 P2 口控制寄存器 (P2CONH, P2CONL) 9-6
9.2.4 P3 口 9-8
9.2.4.1 INT7/TD0OUT/TD0PWM/TD0CAP 9-8
9.2.4.2 P3 口中断控制和标志位寄存器 (P3INTH, P3INTL, P3PND) 9-8
9.2.4.3 P3 口上拉电阻使能控制寄存器 (P3PUR) 9-8
9.2.4.4 P3 口 N 沟道开漏模式寄存器 (PNE3) 9-8

9.2.5 P4 口 9-13
9.2.5.1 P4 口控制寄存器 (P4CONH, P4CONL) 9-13
9.2.5.2 P4 口上拉电阻使能控制寄存器 (P4PUR) 9-13
9.2.5.3 P4 口 N 沟道开漏模式寄存器 (PNE4) 9-13

10 Basic Timer 10-1

10.1 概述 10-1
10.1.1 Basic Timer (BT) 10-1
10.1.1.1 Basic Timer 控制寄存器 (BTCON) 10-2
10.1.2 Basic Timer 功能描述 10-3
10.1.2.1 看门狗功能 10-3
10.1.2.2 振荡器稳定功能 10-3

11 8位 Timer A/B 11-1

11.1 8位 Timer A 11-1
11.1.1 概述 11-1
11.1.1.1 Timer A 控制寄存器 (TACON) 11-2
11.1.2 Timer A 功能描述 11-3
11.1.2.1 Timer A 中断 (IRQ0，中断向量地址：CEH 和 D0H) 11-3
11.1.2.2 Interval (定时) 模式 11-3
11.1.2.3 PWM 模式 11-4
11.1.2.4 Capture (捕获) 模式 11-5
11.1.3 模块框图 11-6
11.2 8位 Timer B 11-7
11.2.1 概述 11-7
11.2.2 模块框图 11-8
11.2.2.1 Timer B 脉冲宽度计算 11-9

12 8位 Timer C 12-1

12.1 8位 Timer C 12-1
12.1.1 概述 12-1
12.1.1.1 Timer C 控制寄存器 (TCCON) 12-2
12.1.2 模块框图 12-3

13 16位 Timer D0/D1 13-1

13.1 16位 Timer D0 13-1
13.1.1 概述 13-1
13.1.1.1 Timer D0 控制寄存器 (TD0CON) 13-2
13.1.2 Timer D0 功能描述 13-3
13.1.2.1 Timer D0 中断 (IRQ3, 中断向量地址 D8H 和 DAH) 13-3
13.1.2.2 Interval (定时) 模式 13-3
13.1.2.3 PWM 模式 13-4
13.1.2.4 Capture (捕获) 模式 13-5
13.1.3 模块框图 13-6

13.2 16位 Timer D1 13-7
13.2.1 概述 13-7
13.2.1.1 Timer D1 控制寄存器 (TD1CON) 13-8
13.2.2 Timer D1 功能描述 13-9
13.2.2.1 Timer D1 中断 (IRQ3, 中断向量地址 DCH 和 DEH) 13-9
13.2.2.2 Interval (定时) 模式 13-9
13.2.2.3 PWM 模式 13-10
13.2.2.4 Capture (捕获) 模式 13-11
13.2.3 模块框图 13-12

14 钟表定时器 14-1

14.1 概述 14-1
14.1.1 钟表定时器控制寄存器 (WTCON) 14-2
14.1.2 钟表定时器电路图 14-3

15 LCD 控制器/驱动器 15-1

15.1 概述 15-1
15.1.1 LCD 电路框图 15-2
15.1.2 LCD RAM 地址空间 15-3
15.1.3 LCD 控制寄存器 (lCON) 15-4
15.1.4 内部电阻偏压管脚连接 15-5
15.1.5 Common (COM) 信号 15-6
15.1.6 Segment (SEG) 信号 15-6

16 10 位 A/D 转换器 16-1

16.1 概述 16-1
16.2 功能描述 16-1
16.2.1 转换时序 16-2
16.2.1.1 A/D 转换控制寄存器 (ADCON) 16-2
16.2.1.2 内部参考电压 16-3
16.3 模块框图 16-4

17 串行输入/输出接口 17-1

17.1 概述 17-1
17.2 编程步骤 17-1
17.2.1 SIO 控制寄存器 (SIOCON) 17-2
17.2.2 SIO 预分频寄存器 (SIOPS) 17-3
17.3 模块框图 17-4
17.3.1 串行输入/输出时序图 17-5

18 UART 0 18-1

18.1 概述 18-1
18.1.1 UART 0 高字节控制寄存器 (UART0CONH) 18-2
18.1.2 UART 0 低字节控制寄存器低字节 (UARTCONL) 18-2
18.1.3 UART 0 中断标志位 18-5
18.1.4 UART 0 数据寄存器 (UDATA0) 18-5

18.1.5 uart 0 波特率数据寄存器 (BRDATA0) 18-5
18.1.6 波特率计算 18-6
18.1.7 模块框图 18-7
18.1.8 UART 0 模式 0 功能描述 18-8
18.1.8.1 模式 0 数据发送流程 18-8
18.1.8.2 模式 0 数据接收流程 18-8
18.1.9 UART 0 模式 1 功能描述 18-10
18.1.9.1 模式 1 数据发送流程 18-10
18.1.9.2 模式 1 数据接收流程 18-10
18.1.10 UART 0 模式 2 功能描述 18-12
18.1.10.1 模式 2 数据发送流程 18-12
18.1.10.2 模式 2 数据接收流程 18-12
18.1.11 UART 0 模式 3 功能描述 18-14
18.1.11.1 模式 3 数据发送流程 18-14
18.1.11.2 模式 2 数据接收流程 18-14
18.1.12 多机串行通讯的配置 18-16
18.1.13 主机/从机交互协议的一个例子 18-16
18.1.14 多机通讯的配置流程 18-17

19 UART 1 19-1

19.1 概述 19-1
19.1.1 UART 1 高字节控制寄存器 (UART1CONH) 19-2
19.1.2 UART 1 低字节控制寄存器 (UART1CONL) 19-2
19.1.3 UART 1 中断标志位 19-5
19.1.4 UART 1 数据寄存器 (UDATA1) 19-5
19.1.5 uart 1 波特率数据寄存器 (BRDATA1) 19-5
19.1.6 波特率计算 19-6
19.1.7 模块框图 19-7
19.1.8 UART 1 模式 0 功能描述 19-8
19.1.9 UART 1 模式 1 功能描述 19-10
19.1.9.1 模式 1 数据发送流程 19-10
19.1.9.2 模式 1 数据接收流程 19-10
19.1.10 UART 1 模式 2 功能描述 19-12
19.1.10.1 模式 2 数据发送流程 19-12
19.1.10.2 模式 2 数据接收流程 19-12
19.1.11 UART 1 模式 3 功能描述 19-14
19.1.11.1 模式 3 数据发送流程 19-14
19.1.11.2 模式 3 数据接收流程 19-14
19.1.12 多机串行通讯的配置 19-16
19.1.13 主机/从机交互协议的一个例子 19-16
19.1.14 多机通讯的配置流程 19-17

20 Pattern Generation 模块 20-1

20.1 概述 20-1
20.1.1 Pattern Gneration 流程 20-1

21 嵌入式闪存接口 21-1

21.1 概述 21-1
21.2 用户编程模式 21-2
21.2.1 闪存控制寄存器 (用户编程模式) 21-3
21.2.1.1 闪存控制寄存器 (FMCON) 21-3
21.2.1.2 闪存用户编程使能寄存器 21-4
21.2.1.3 闪存扇区地址寄存器 21-5
21.3 ISP™ (在板编程) 扇区 21-6
21.3.1.1 ISP 复位向量和 ISP 扇区大小的关系 21-7
21.4 扇区擦除 21-8
21.4.1 用户编程模式下的扇区擦除流程 21-9
21.5 编程 21-10
21.5.1 用户编程模式下编程的流程 21-10
21.6 读 21-12
21.6.1 用户编程模式下读操作的编程流程 21-12
21.7 Hard Lock 保护 21-13
21.7.1 用户编程模式下 Hard Lock 保护的编程流程 21-13

22 电气参数 22-1

22.1 概述 22-1

23 机械尺寸 23-1

23.1 概述 23-1

24 S3F84UA/F84U8 Flash MCU 24-1

24.1 概述 24-1
24.2 在板编程 24-5
24.2.1 电路设计指导 24-5
24.2.2 连接参照表 24-6

List of Figures

Figure Number	Title	Page Number
图 1-1	模块框图	1-6
图 1-2	S3F84UA/F84U8 管脚分布 (44-QFP-1010B)	1-7
图 1-3	S3F84UA/F84U8 管脚分布 (42-SDIP-600)	1-8
图 1-4	管脚电路类型 A	1-12
图 1-5	管脚电路类型 B	1-12
图 1-6	管脚电路类型 C	1-13
图 1-7	管脚电路类型 D-2 (P1.0–P1.1)	1-13
图 1-8	管脚电路类型 F-16 (P0)	1-14
图 1-9	管脚电路类型H-39	1-14
图 1-10	管脚电路类型 H-44 (P2)	1-15
图 1-11	管脚电路类型 H-41 (P3)	1-15
图 1-12	管脚电路类型 H-42 (P4)	1-16
图 2-1	程序存储地址空间	2-2
图 2-2	Smart Option	2-3
图 2-3	内部寄存器卷的组织结构 (S3F84UA)	2-5
图 2-4	内部寄存器卷的组织结构 (S3F84U8)	2-6
图 2-5	寄存器页指针 (PP)	2-7
图 2-6	Set 1, Set 2, 主寄存器空间和 LCD 数据寄存器映射图	2-10
图 2-7	8 字节工作寄存器区 (Slices)	2-11
图 2-8	相邻的 16 字节工作寄存器块	2-12
图 2-9	非相邻的 16 字节工作寄存器块	2-13
图 2-10	16 位寄存器对	2-14
图 2-11	寄存器卷寻址模式	2-15
图 2-12	通用工作寄存器区	2-16
图 2-13	4 位工作寄存器寻址模式	2-19
图 2-14	4 位工作寄存器寻址实例	2-19
图 2-15	8 位工作寄存器寻址模式	2-20
图 2-16	8 位工作寄存器寻址实例	2-21
图 2-17	栈操作	2-22
图 3-1	寄存器寻址模式	3-2
图 3-2	工作寄存器寻址模式	3-2
图 3-3	寄存器卷中的间接寄存器寻址模式	3-3
图 3-4	程序存储空间的间接寄存器寻址	3-4
图 3-5	寄存器卷中的间接寄存器寻址	3-5
图 3-6	工作寄存器间接访问程序存储器或数据存储器	3-6
图 3-7	寄存器空间的偏址寻址模式	3-7
图 3-8	偏址寻址模式中短格式访问程序或数据存储空间	3-8
图 3-9	偏址寻址模式中长格式访问程序或数据存储空间	3-9
图 3-10	Load 指令的直接寻址	3-10
图 3-11	Call, Jump 的直接寻址	3-11

图 3-12 间接寻址模式 3-12
图 3-13 相对寻址 3-13
图 3-14 立即数寻址模式 3-14

图 4-1 寄存器描述格式 4-4

图 5-1 S3C8- 系列的中断类型 5-2
图 5-2 S3F84UA/F84U8 中断结构 5-4
图 5-3 ROM 中断向量地址区 5-5
图 5-4 中断功能框图 5-8
图 5-5 系统模式寄存器 (SYM) 5-11
图 5-6 中断屏蔽寄存器 (IMR) 5-12
图 5-7 中断请求优先级组 5-13
图 5-8 中断优先级寄存器 (IPR) 5-14
图 5-9 中断请求寄存器 (IRQ) 5-15

图 6-1 系统标志寄存器 (FLAGS) 6-5

图 7-1 石英/陶瓷晶振 (fX) 7-2
图 7-2 外部振荡器 (fX) 7-2
图 7-3 RC 振荡器 (fX) 7-2
图 7-4 石英振荡器 (fxt) 7-3
图 7-5 外部振荡器 (fxt) 7-3
图 7-6 系统时钟电路图 7-4
图 7-7 系统时钟控制寄存器 (CLKCON) 7-5
图 7-8 时钟控制寄存器 (OSCCON) 7-6
图 7-9 STOP 控制寄存器 (STPCON) 7-7

图 9-1 P0 口高字节控制寄存器 (P0CONH) 9-3
图 9-2 P0 口低字节控制寄存器 (P0CONL) 9-4
图 9-3 P0 口上拉电阻使能控制寄存器 (P0PUR) 9-4
图 9-4 P1 口控制寄存器 9-5
图 9-5 P2 口高字节控制寄存器 (P2CONH) 9-6
图 9-6 P2 口低字节控制寄存器 (P2CONL) 9-7
图 9-7 P3 口高字节控制寄存器 (P3CONH) 9-9
图 9-8 P3 口低字节控制寄存器 (P3CONL) 9-9
图 9-9 P3 口高字节中断控制寄存器 (P3INTH) 9-10
图 9-10 P3 口低字节中断控制寄存器 (P3INTL) 9-10
图 9-11 P3 口中断标志位寄存器 (P3PND) 9-11
图 9-12 P3 口上拉电阻使能控制寄存器 (P3PUR) 9-11
图 9-13 P3 口 N 沟道开漏模式寄存器 (PNE3) 9-12
图 9-14 P4 口高字节控制寄存器 (P4CONH) 9-13
图 9-15 P4 口低字节控制寄存器 (P4CONL) 9-14
图 9-16 P4 口上拉电阻使能控制寄存器 (P4PUR) 9-14
图 9-17 P4 口 N 沟道开漏模式寄存器 (PNE4) 9-15

图 10-1 Basic Timer 控制寄存器 (BTCON) 10-2
图 10-2 Basic Timer 功能框图 10-4

图 11-1 Timer A 控制寄存器 (TACON) 11-2
图 11-2 简化的 Timer A 功能图：Interval (定时) 模式 11-3
图 11-3 简化的 Timer A 功能图：PWM 模式 11-4
图 11-4 简化的 Timer A 功能图：Capture (捕获) 模式 11-5
图 11-5 Timer A 功能模块框图 11-6
图 11-6 Timer B 控制寄存器 11-7
图 11-7 Timer B 功能模块框图 11-8
图 11-8 Timer B 在 Repeat (重复) 模式下输出的触发器波形 11-10

图 12-1 Timer C 控制寄存器 (TCCON) 12-2
图 12-2 Timer C 功能模块框图 12-3

图 13-1 Timer D0 控制寄存器 (TD0CON) 13-2
图 13-2 简化的 Timer D0 功能框图：Interval (定时) 模式 13-3
图 13-3 简化的 Timer D0 功能简图: PWM 模式 13-4
图 13-4 简化的 Timer D0 功能简图：Capture (捕获) 模式 13-5
图 13-5 Timer D0 功能模块框图 13-6
图 13-6 Timer D1 控制寄存器 (TD1CON) 13-8
图 13-7 简化的 Timer D1 功能框图：Interval (定时) 模式 13-9
图 13-8 简化的 Timer D1 功能框图：PWM 模式 13-10
图 13-9 简化的 Timer D1 功能框图：Capture (捕获) 模式 13-11
图 13-10 Timer D1 功能模块框图 13-12

图 14-1 钟表定时器控制寄存器 (WTCON) 14-2
图 14-2 钟表定时器电路框图 14-3

图 15-1 LCD 功能框图 15-1
图 15-2 LCD 电路框图 15-2
图 15-3 LCD 显示数据 RAM 结构图 15-3
图 15-4 LCD 控制寄存器 (LCON) 15-4
图 15-5 内部电阻偏压管脚连接图 15-5
图 15-6 1/2 占空比, 1/2 偏压比显示模式下的选定/未选定信号 15-7
图 15-7 1/3 占空比, 1/3 偏压比显示模式下的选定/未选定信号 15-7
图 15-8 LCD 信号波形 (1/2 占空比, 1/2 偏压比) 15-8
图 15-9 LCD 信号波形 (1/3 占空比, 1/3 偏压比) 15-9
图 15-10 LCD 信号波形 (1/4 占空比, 1/3 偏压比) 15-10
图 15-11 LCD 信号波形 (1/8 占空比, 1/4 偏压比) 15-11
图 15-12 LCD 信号波形 (1/8 占空比, 1/4 偏压比) (续) 15-12

图 16-1 A/D 转换控制寄存器 (ADCON) 16-2
图 16-2 A/D 转换数据寄存器 (ADDATAH/L) 16-3
图 16-3 A/D 转换功能模块图 16-4
图 16-4 推荐高精度 A/D 转换电路 16-5

图 17-1 串行输入/输出模块控制寄存器 (SIOCON) 17-2
图 17-2 SIO 预分频寄存器 (SIOPS) 17-3
图 17-3 SIO 功能模块框图 17-4
图 17-4 串行输入/输出发送/接收模式的时序 (下降沿发送, SIOCON.4 = 0) 17-5
图 17-5 串行输入/输出发送/接收模式的时序 (上升沿发送，SIOCON.4 = 1) 17-5

图 18-1 UART 0 高字节控制寄存器 (UART0CONH) 18-3
图 18-2 UART 0 低字节控制寄存器 (UART0CONL) 18-4
图 18-3 UART 0 数据寄存器 (UDATA0) 18-5
图 18-4 UART 0 波特率数据寄存器 (BRDATA0) 18-5
图 18-5 UART 0 功能模块框图 18-7
图 18-6 UART0 模式 0 运行时序图 18-9
图 18-7 UART 0 模式 1 运行时序图 18-11
图 18-8 UART 0 模式 2 运行时序图 18-13
图 18-9 UART 0 模式 3 运行时序图 18-15
图 18-10 多机串行通讯连接示例 18-17

图 19-1 UART 1 高字节控制寄存器 (UART1CONH) 19-3
图 19-2 UART 1 低字节控制寄存器 (UART1CONL) 19-4
图 19-3 UART 1 数据寄存器 (UDATA1) 19-5
图 19-4 UART 1 波特率数据寄存器 (BRDATA1) 19-5
图 19-5 UART 1 功能模块框图 19-7
图 19-6 UART 1 模式 1 运行时序图 19-9
图 19-7 UART 1 模式 1 运行时序图 19-11
图 19-8 UART 1 模式 2 运行时序图 19-13
图 19-9 UART 1 模式 3 运行时序图 19-15
图 19-10 多机串行通讯连接示例 19-17

图 20-1 Pattern Generation 流程 20-1
图 20-2 Pattern Generation 控制寄存器(PGCON) 20-2
图 20-3 Pattern Generation 控制电路 20-2

图 21-1 闪存控制寄存器 (FMCON) 21-3
图 21-2 闪存用户编程使能寄存器 (FMUSR) 21-4
图 21-3 闪存扇区地址寄存器高字节 (FMSECH) 21-5
图 21-4 闪存扇区地址寄存器低字节 (FMSECL) 21-5
图 21-5 程序存储地址空间 21-6
图 21-6 用户编程模式下的扇区 21-8

图 22-1 外部中断输入时序 22-5
图 22-2 nRESET 输入时序 22-5
图 22-3 nRESET 使系统退出 STOP 模式时序图 22-6
图 22-4 中断使系统退出 STOP 模式时序图 22-7
图 22-5 LVR (低电压复位) 时序 22-8
图 22-6 串行数据发送时序 22-9
图 22-7 UART 时序特性波形图 22-10
图 22-8 UART 模块的时序波形图 22-11
图 22-9 X_{IN} 上测量到的时钟时序 22-13
图 22-10 XT_{IN} 上测量到的时钟时序 22-13
图 22-11 工作电压范围 22-14

图 23-1 封装尺寸 (44-QFP-1010B) 23-1
图 23-2 封装尺寸 (42-SDIP-600) 23-2

图 24-1 S3F84UA/F84U8 管脚分布 (44-QFP-1010B) 24-2
图 24-2 S3F84UA/F84U8 管脚分布 (42-SDIP-600) 24-3
图 24-3 RC 时延电路 24-4
图 24-4 编程接口(在板编程) PCB 设计指导 24-5

List of Tables

Table Number	Title	Page Number
表 1-1	S3F84UA/F84U8 管脚描述	1-9
表 2-1	S3F84UA 寄存器类型总结	2-4
表 2-2	S3F84U8 寄存器类型总结	2-4
表 4-1	Set 1 寄存器	4-1
表 4-2	Set 1, Bank 0 寄存器	4-2
表 4-3	Set 1, Bank 1 寄存器	4-3
表 5-1	中断向量	5-6
表 5-2	中断控制寄存器概述	5-7
表 5-3	中断源控制寄存器和数据寄存器	5-9
表 6-1	指令集简介	6-2
表 6-2	指令集简介	6-3
表 6-3	标志位符号	6-7
表 6-4	指令集符号	6-7
表 6-5	指令符号的定义	6-8
表 6-6	指令代码快速参考	6-9
表 6-7	指令代码快速参考 (续)	6-10
表 6-8	条件码	6-11
表 8-1	S3F84UA/F84U8 Set 1 复位后的寄存器值	8-2
表 8-2	S3F84UA/F84U8 Set 1, Bank 0 复位后的寄存器值	8-3
表 8-3	S3F84UA/F84U8 Set 1, Bank 1 复位后的寄存器值	8-4
表 9-1	S3F84UA/F84U8 口设置概述	9-1
表 9-2	端口 数据寄存器概述	9-2
表 18-1	利用 BRDATA0 可以产生的常用波特率表	18-6
表 19-1	利用 BRDATA1 可以产生的常用波特率表	19-6
表 21-1	ISP 扇区大小	21-7
表 21-2	复位向量地址	21-7
表 22-1	芯片极限物理特性	22-2
表 22-2	直流电气特性	22-3
表 22-3	交流电气特性	22-5
表 22-4	输入/输出电容	22-6
表 22-5	STOP 模式下 数据保持电压	22-6
表 22-6	A/D 转换电气特性	22-7
表 22-7	LVR (低电压复位) 电气特性	22-8

表 22-8 同步 SIO 电气特性 22-9
表 22-9 UART 在模式 0 下的时序特性 (12.0MHz) 22-10
表 22-10 主振荡器特性 22-12
表 22-11 副振荡器特性 22-12
表 22-12 主振荡器稳定时间 22-13
表 22-13 副振荡器稳定时间 22-13
表 22-14 内部闪存电气特性 22-14

表 24-1 Flash ROM 读/写管脚特性描述 24-4
表 24-2 连接参照表 24-6

List of Examples

Example Number	Title	Page Number
编程实例 2-1	在清 RAM 时使用页指针 (第 0 页, 第 1 页)	2-8
编程实例 2-2	设置寄存器指针	2-12
编程实例 2-3	使用 RPs 对多个寄存器的内容求和	2-13
编程实例 2-4	访问通用工作寄存器区	2-17
编程实例 2-5	用 PUSH 和 POP 实现标准栈操作	2-23
编程实例 5-1	如何避免不希望的外部中断发生	5-10
编程实例 5-2	如何清除中断标志位	5-16
编程实例 7-1	如何使用 STOP 指令	7-7
编程实例 7-2	切换 CPU 时钟	7-8
编程实例 11-1	如何在 P4.1 处产生一个 38 kHz, 1/3 占空比的信号	11-11
编程实例 11-2	在 P4.1 处产生一个脉冲信号	11-12
编程实例 20-1	如何使用 Pattern Generation	20-3
编程实例 21-1	扇区擦除	21-9
编程实例 21-2	编程	21-11
编程实例 21-3	读	21-12
编程实例 21-4	Hard Lock 保护	21-13

1 产品概述

1.1 S3C8- 系列 MCU

三星 S3C8- 系列 8 位 CMOS 单片机向用户提供了高效快速的 CPU，丰富的外围接口，
以及各种大小的可编程掩模 ROM。该 CPU 主要特性如下：

- 高效的面向寄存器架构
- 可供选择的 CPU 时钟源
- 可中断唤醒的 IDLE 和 STOP 省电模式
- 带看门狗功能的 Basic Timer

复杂中断结构最高支持 8 个中断级。每个中断级有一个或多个中断源和中断向量。
针对特定的中断级，可指定为快速中断操作 (最快 4 个 CPU 时钟)。

1.2 S3F84UA/F84U8 MCU

S3F84UA/F84U8 单片 CMOS MCU 由先进 CMOS 工艺制造，并基于三星最新的 CPU 架构。

S3F84UA，S3F84U8 内嵌 48K，8K字节的 Flash ROM。

S3F84UA 是一款内嵌 48K Flash ROM 的MCU。S3F84U8 是一款内嵌 8K Flash ROM 的 MCU。

通过成熟的模块化设计方法，和基于强大的 SAM8 内核，在 S3F84UA/F84U8 中成功地集成了以下外围模块：

- 五个可编程 I/O 口，包括四个 8 位端口和一个 4 位端口，总共 36 个管脚
- 八个用于外部中断的位可编程管脚
- 一个用于时钟稳定以及看门狗功能(系统复位)的 8 位 Basic Timer
- 三个 8 位 Timer/Counter 和两个可选工作模式的 16 位 Timer/Counter
- 用作实时时钟的钟表定时器
- LCD 控制器/驱动器
- 带 8 个可选择输入管脚的 A/D 转换器
- 同步 SIO 模块
- 两个异步 UART 模块
- Pattern Generation 模块

目前可提供的封装类型有 44-管脚-QFP 和 42-管脚-SDIP。

1.3 特性

1.3.1 CPU

- SAM88 RC CPU 内核

1.3.2 存储器

- 程序存储器 (ROM)
- 48K × 8 位程序存储器 (S3F84UA)
- 8K × 8 位程序存储器 (S3F84U8)
  - 内部 Flash 存储器 (程序存储器)
  - Sector(扇区)大小：128 字节
  - 10 年数据保留
  - 快速编程时间
  - 支持用户编程和扇区可擦除
  - 寿命：10,000 次可擦除/编程
  - 支持外部串行编程
  - 可扩展 OBPTM (在板编程) 扇区
- 数据存储器 (RAM)
  - 包括 LCD 显示数据内存空间
  - 550 × 8 位数据存储器(S3F84UA)
  - 294 × 8 位数据存储器(S3F84U8)

1.3.3 指令集

- 78 条指令
- 用于低功耗模式的 IDLE 和 STOP 指令

1.3.4 34 个 I/O 管脚

- 输出：2 个管脚 (仅 44-QFP)
- I/O：10 个管脚 (与其他信号管脚复用)
- I/O：24 个管脚 (与 LCD 输出信号复用)

1.3.5 中断

- 8 个中断级和 22 个中断源
- 支持快速中断处理

1.3.6 8-位 BASIC TIMER

- 看门狗功能
- 4 种时钟源

1.3.7 8 位 TIMER/COUNTER A

- 可编程的 8 位内部 Timer
- 外部事件 Counter 功能
- PWM 和 Capture(捕获)功能

1.3.8 8 位 TIMER/COUNTER B

- 可编程的 8 位内部 Timer
- 载波频率发生器

1.3.9 8 位 TIMER/COUNTER C

- 可编程的 8 位 内部 Timer
- PWM 功能

1.3.10 两个 16 位 TIMER/COUNTER (D0/D1)

- 可编程的 16-位内部 Timer
- 外部事件 Counter 功能
- PWM 和 Capture(捕获)功能

1.3.11 钟表定时器

- Interval(定时)时钟: 在 32.768kHz 频率时，1.995ms, 0.125s, 0.25s, 和 0.5s
- 可选择的 0.5/1/2/4kHz 蜂鸣器输出

1.3.12 LCD 控制器/驱动器

- 16 个 Segments 和 8 个 Commons 输出
- 可选择的 1/2, 1/3, 1/4, 和 1/8 占空比
- 可选择的电阻串偏压

1.3.13 模拟数字转换器

- 8 通道模拟输入
- 10 位转换精度
- 25us 转换时间

1.3.14 两个通道 UART

- 全双工串行 I/O 接口
- 四种可编程的工作模式
- 自动生成校验位

1.3.15 8 位串行 I/O 接口

- 8 位发送/接收模式
- 8 位接收模式
- 可选择 LSB 最先或 MSB 最先发送
- 内部或外部时钟源

1.3.16 PATTERN GENERATION 模块

- Timer 匹配信号或者软件可触发 Pattern Generation 模块

1.3.17 低电压复位 (LVR)

- 标准电压：2.2V
- 可由 Smart Option 使能/禁止(ROM 地址：3FH)

1.3.18 两种省电模式

- IDLE：仅 CPU 时钟停止
- STOP：选定的系统时钟和 CPU 时钟停止

1.3.19 时钟源

- 用作主时钟的晶振，陶振或 RC
- 主时钟频率：0.4MHz ~ 12.0MHz
- 32.768 kHz 时钟振荡电路可作为副时钟

1.3.20 指令执行时间

- fx 为 12.0MHz 时最小 333ns
- fxt 为 32.768kHz 时最小 122.1us

1.3.21 工作电压范围

- 2.0V 至 5.5V @ 0.4 ~ 4.2MHz
- 2.7V 至 5.5V @ 0.4 ~ 12.0MHz

1.3.22 工作温度范围

- –40°C 至 + 85°C

1.3.23 封装类型

- 40-QFP-1010B, 42-SDIP-600

1.3.24 IVC

- 用作 5V 工作的内部电压转换器

1.3.25 SMART OPTION

- 低电压复位 (LVR) 电平值选择，以及使能/禁止 (ROM 地址 3FH)
- ISP 相关选项的选择 (ROM 地址 3EH)

1.4 模块框图

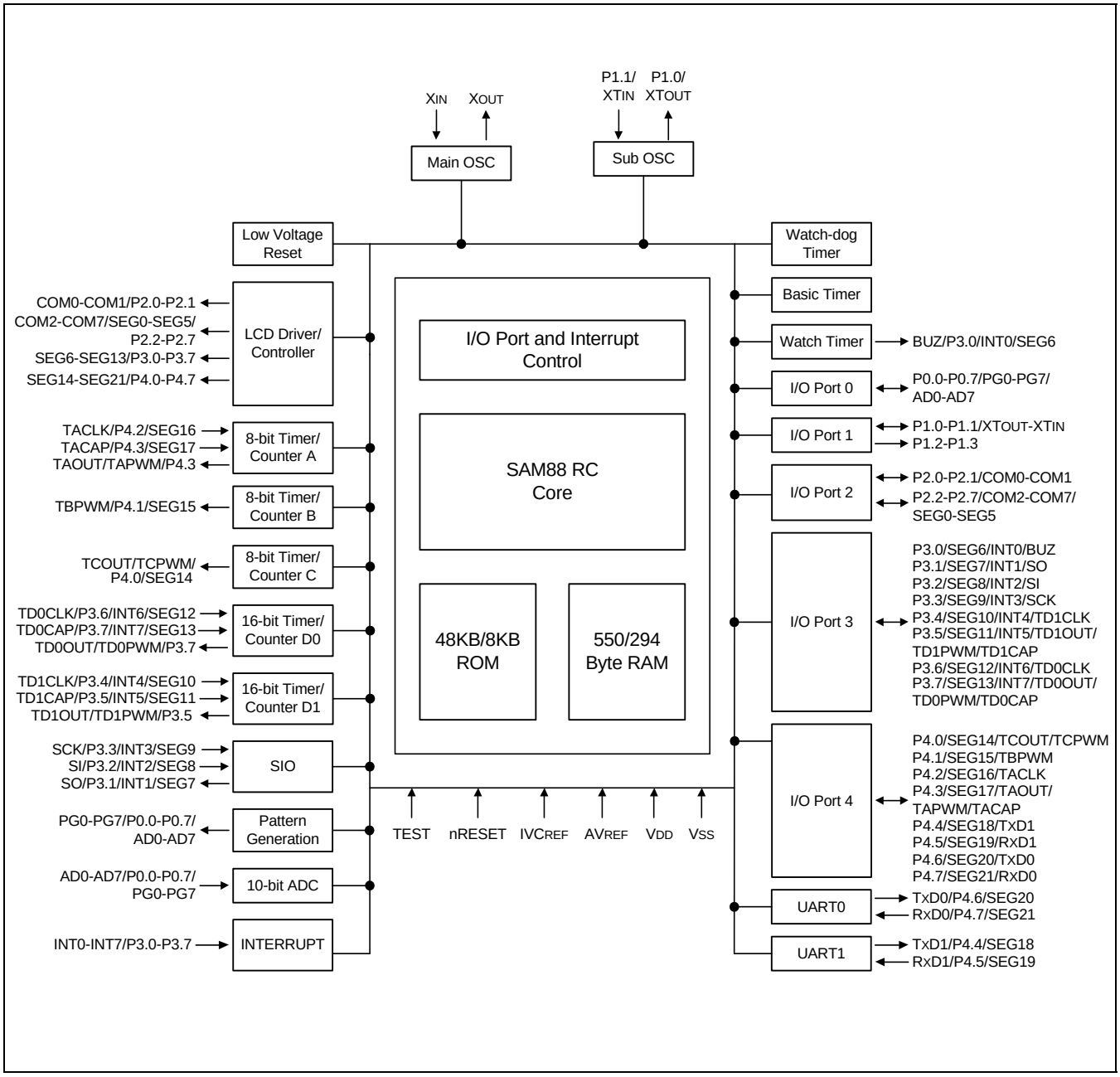


图 1-1 模块框图

1.5 管脚分布图

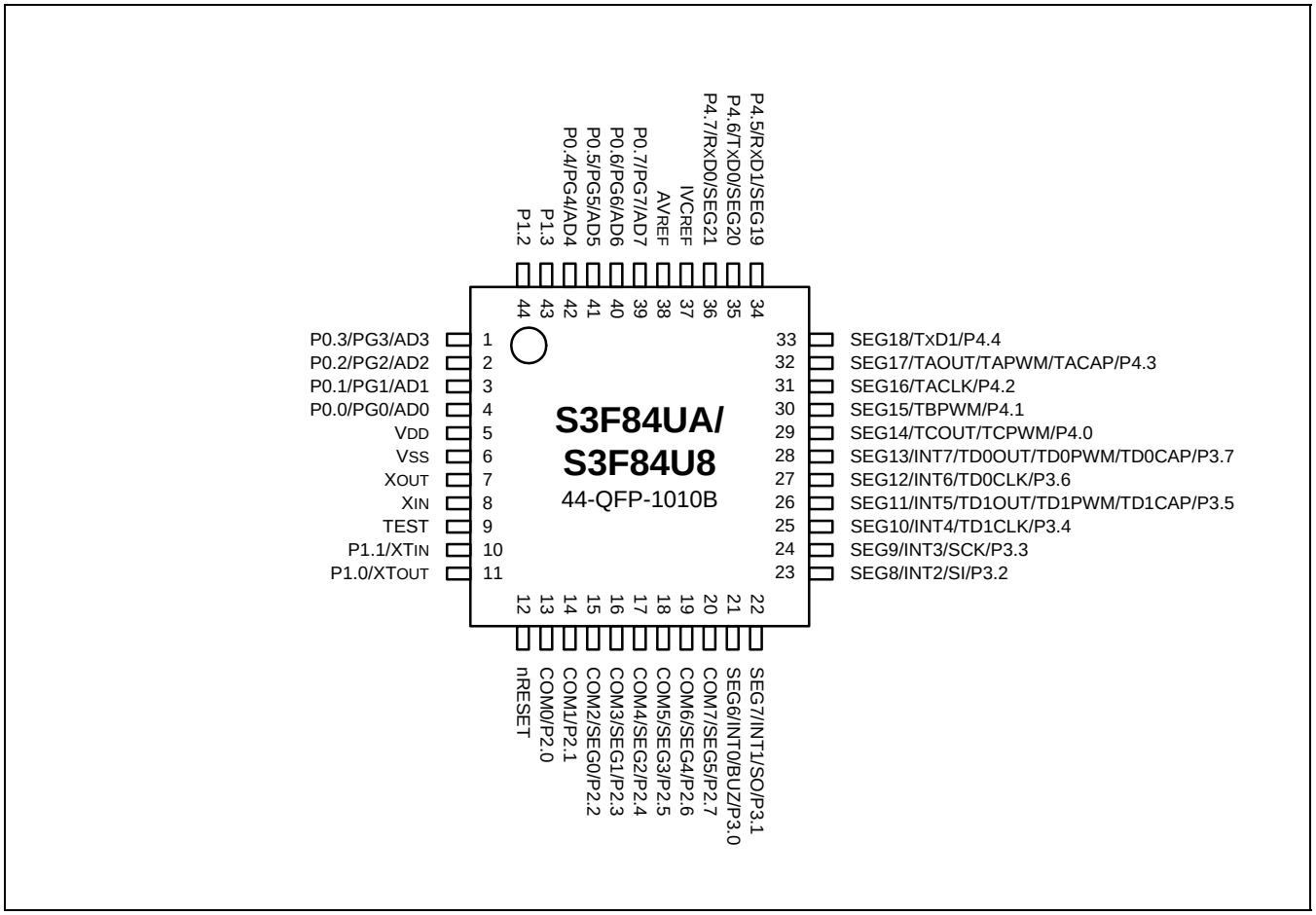


图 1-2 S3F84UA/F84U8 管脚分布 (44-QFP-1010B)

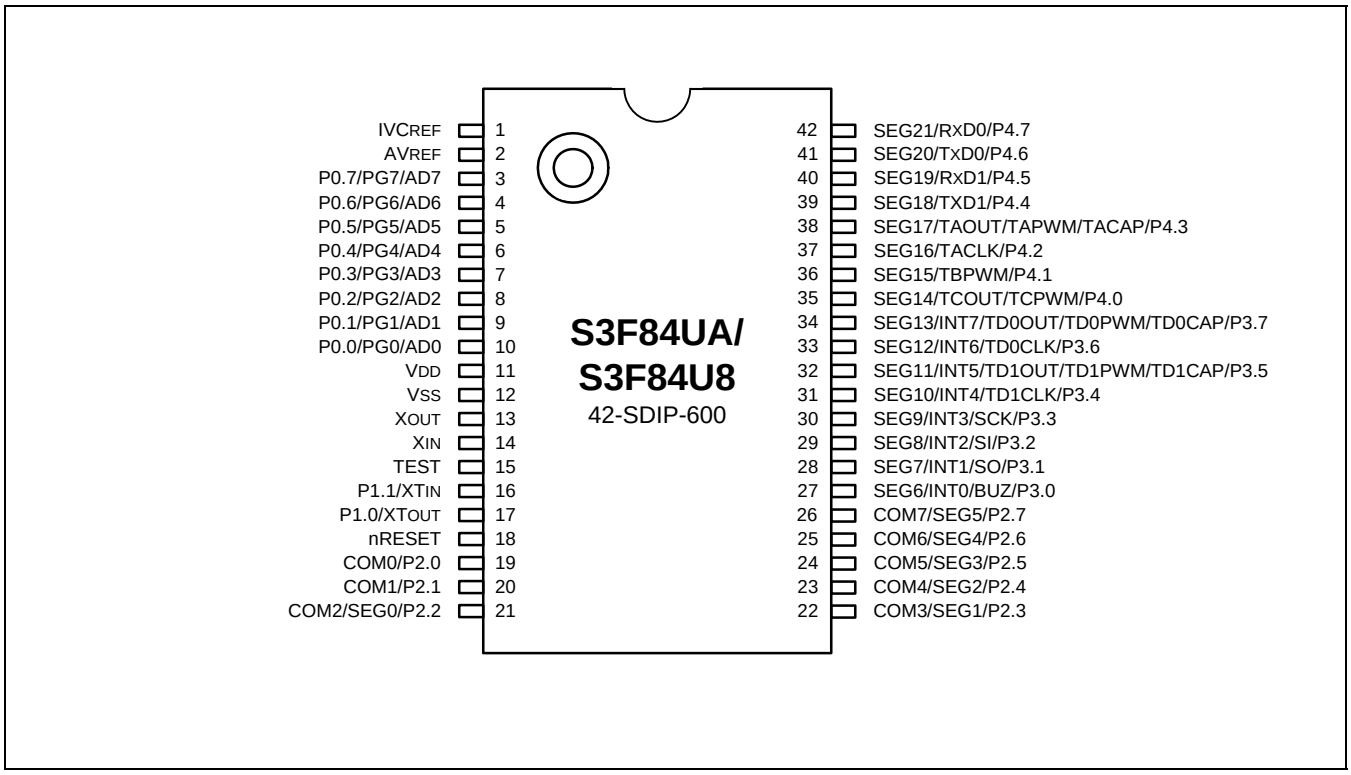


图 1-3 S3F84UA/F84U8 管脚分布 (42-SDIP-600)

1.6 管脚描述

表 1-1 S3F84UA/F84U8 管脚描述

管脚名称	管脚类型	管脚描述	电路类型	管脚序号 (备注)	共用管脚
P0.0–P0.3 P0.4–P0.7	I/O	位可编程 I/O 口管脚；可设定为输入或推挽式输出以及软件设定上拉电阻。	F-16	4–1 (10–7) 42–39 (6–3)	AD0–AD3/ PG0–PG3 AD4–AD7/ PG4–PG7
P1.0 P1.1	I/O	位可编程 I/O 口管脚；可设定为输入或推挽式输出以及软件设定上拉电阻。	D-2	11(17) 10(16)	XTOUT XTIN
P1.2 P1.3	O	1 位可编程输出口。	C	44 43	– –
P2.0–P2.1 P2.2–P2.7	I/O	位可编程 I/O 口管脚；可设定为输入或推挽式输出以及软件设定上拉电阻。	H-44	13–14 (19–20) 15–20 (21–26)	COM0–COM1 COM2–COM7/ SEG0–SEG5
P3.0 P3.1 P3.2 P3.3 P3.4 P3.5 P3.6 P3.7	I/O	位可编程 I/O 口管脚；可设定为施密特触发输入或推挽输出，开漏输出以及软件设定上拉电阻。	H-41	21(27) 22(28) 23(29) 24(30) 25(31) 26(32) 27(33) 28(34)	INT0/BUZ/SEG6 INT1/SO/SEG7 INT2/SI/SEG8 INT3/SCK/SEG9 INT4/TD1CLK/ SEG10 INT5/TD1OUT/ TD1PWM/ TD1CAP/SEG11 INT6/TD0CLK/ SEG12 INT7/TD0OUT/ TD0PWM/ TD0CAP/SEG13
P4.0 P4.1 P4.2 P4.3 P4.4 P4.5 P4.6 P4.7	I/O	位可编程 I/O 口管脚；可设定为施密特触发输入或推挽输出，开漏输出以及软件设定上拉电阻。	H-42	29(35) 30(36) 31(37) 32(38) 33(39) 34(40) 35(41) 36(42)	TCOUT/ TCPWM/SEG14 TBPWM/SEG15 TACLK/SEG16 TAOUT/TAPWM/ TACAP/SEG17 TxD1/SEG18 RxD1/SEG19 TxD0/SEG20 RxD0/SEG21
COM0–COM1 COM2–COM7	I/O	LCD Common 信号输出。	H-44	13–14 (19–20) 15–20 (21–26)	P2.0–P2.1 P2.2–P2.7/ SEG0–SEG5
SEG6	I/O	LCD Segment 信号输出。	H-41	21(27)	P3.0/INT0/BUZ

管脚 名称	管脚 类型	管脚描述	电路类型	管脚序号 (备注)	共用管脚
SEG7 SEG8 SEG9 SEG10 SEG11 SEG12 SEG13				22(28) 23(29) 24(30) 25(31) 26(32) 27(33) 28(34)	P3.1/INT1/SO P3.2/INT2/SI P3.3/INT3/SCK P3.4/INT4/ TD1CLK P3.5/INT5/ TD1OUT/TD1PW M/TD1CAP P3.6/INT6/ TD0CLK P3.7/INT7/ TD0OUT/TD0PW M/TD0CAP
SEG14 SEG15 SEG16 SEG17 SEG18 SEG19 SEG20 SEG21	I/O	LCD Segment 信号输出。	H-42	29(35) 30(36) 31(37) 32(38) 33(39) 34(40) 35(41) 36(42)	P4.0/TCOUT/ TCPWM P4.1/TBPWM P4.2/TACLK P4.3/TAOUT/ TAPWM/TACAP P4.4/TxD1 P4.5/RxD1 P4.6/TxD0 P4.7/RxD0
AD0–AD3 AD4–AD7	I/O	A/D 转换器模拟输入通道。	F-16	4–1 (10–7) 42–39 (6–3)	P0.0–P0.3/ PG0–PG3 P0.4–P0.7/ PG4–PG7
AV_{REF}	–	A/D 转换器参考电压。	–	38(2)	–
PG0–PG3 PG4–PG7	I/O	Pattern Generation 输出。	F-16	4–1 (10–7) 42–39 (6–3)	P0.0–P0.3/ AD0–AD3 P0.4–P0.7/ AD4–AD7
TxD0 RxD0	I/O	UART 0 数据输出，输入。	H-42	35(41) 36(42)	P4.6/SEG20 P4.7/SEG21
TxD1 RxD1	I/O	UART 1 数据输出，输入。	H-42	33(39) 34(40)	P4.4/SEG18 P4.5/SEG19
TAOUT/ TAPWM	I/O	Timer A 时钟输出和 PWM 输出口。	H-42	32(38)	P4.3/SEG17/ TACAP
TACAP	I/O	Timer A Capture(捕获)输入口。	H-42	32(38)	P4.3/SEG17/ TAOUT/TAPWM
TACLK	I/O	Timer A 外部时钟输入口。	H-42	31(37)	P4.2/SEG16
TBPWM	I/O	Timer B 载波频率输出口。	H-42	30(36)	P4.1/SEG15
TCOUT/ TCPWM	I/O	Timer C 时钟输出和 PWM 输出口。	H-42	29(35)	P4.0/SEG14
TD0OUT/	I/O	Timer D0 时钟输出和 PWM 输出口。	H-41	28(34)	P3.7/SEG13/

管脚名称	管脚类型	管脚描述	电路类型	管脚序号 (备注)	共用管脚
TD0PWM					INT7/TD0CAP
TD0CAP	I/O	Timer D0 Capture(捕获)输入口。	H-41	28(34)	P3.7/SEG13/ INT7/TD0OUT/ TD0PWM
TD0CLK	I/O	Timer D0 外部时钟输入口。	H-41	27(33)	P3.6/SEG12/ INT6
TD1OUT/ TD1PWM	I/O	Timer D1 时钟输出和 PWM 输出口。	H-41	26(32)	P3.5/SEG11/ INT5/TD1CAP
TD1CAP	I/O	Timer D1 Capture(捕获)输入口。	H-41	26(32)	P3.5/SEG11/ INT5/TD1OUT/ TD1PWM
TD1CLK	I/O	Timer D1 外部时钟输入口。	H-41	25(31)	P3.4/SEG10/ INT4
BUZ	I/O	蜂鸣器信号输出管脚。	H-41	21(27)	P3.0/SEG6/INT0
SCK	I/O	串行接口时钟。	H-41	24(30)	P3.3/SEG9
SI	I/O	串行接口数据输入口。	H-41	23(29)	P3.2/SEG8
SO	I/O	串行接口数据输出口。	H-41	22(28)	P3.1/SEG7
INT0–INT7	I/O	外部中断输入管脚。	H-41	21–28 (27–34)	P3.0–P3.7/ SEG6–SEG13
nRESET	I	系统复位管脚。	B	12(18)	–
X_{IN}, X_{OUT}	–	主时钟管脚。	–	8(14), 7(13)	–
XT_{IN}, XT_{OUT}	–	副时钟晶振管脚。	–	10(16) 11(17)	P1.0 P1.1
TEST	I	TEST 输入：必须接 V_{SS}。	–	9(15)	–
V_{DD}	–	供电电源输入管脚。	–	5(11)	–
V_{SS}	–	电源地管脚。	–	6(12)	–
IVC_{REF}	–	内部电压转换参考输入管脚。	–	37(1)	–

注释: 圆括号中管脚号为 42-SDIP-600 封装。

1.7 管脚电路

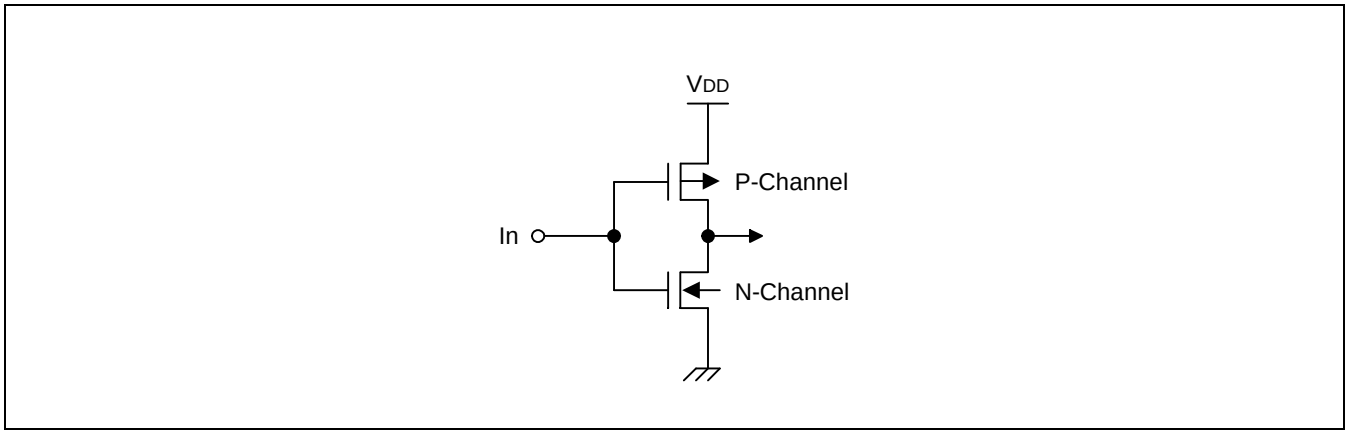


图 1-4 管脚电路类型 A

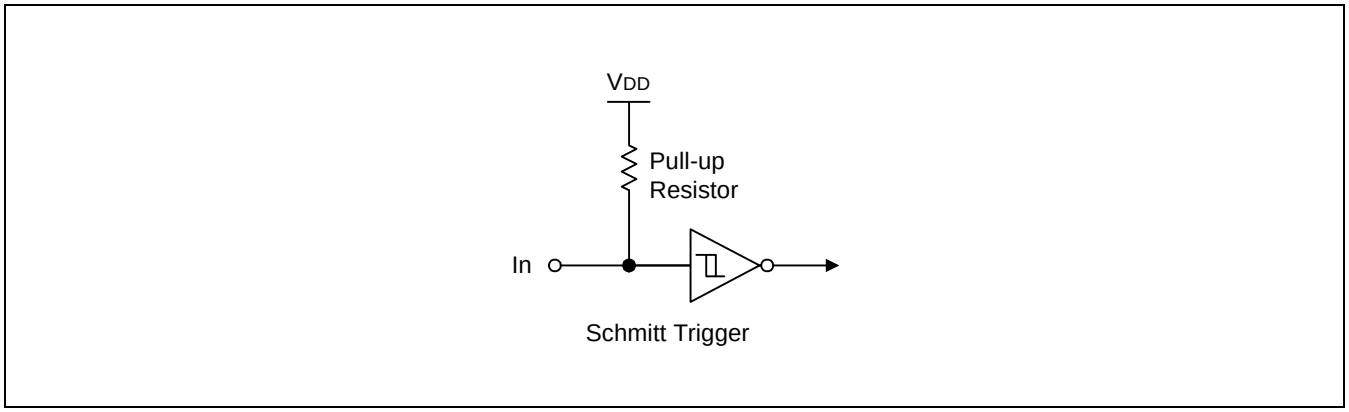


图 1-5 管脚电路类型 B

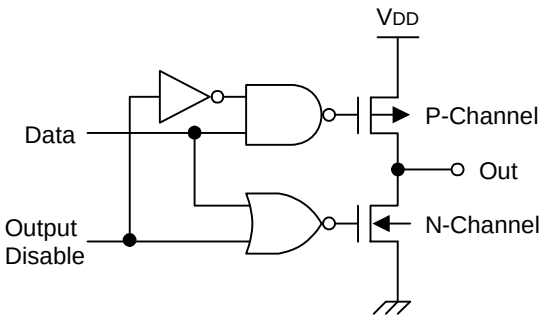


图 1-6 管脚电路类型 C

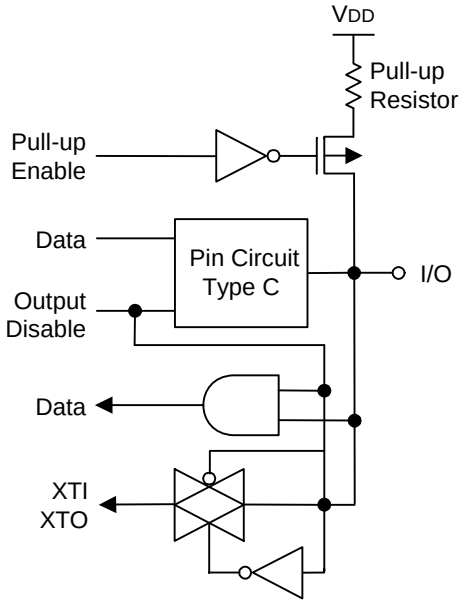


图 1-7 管脚电路类型 D-2 (P1.0–P1.1)

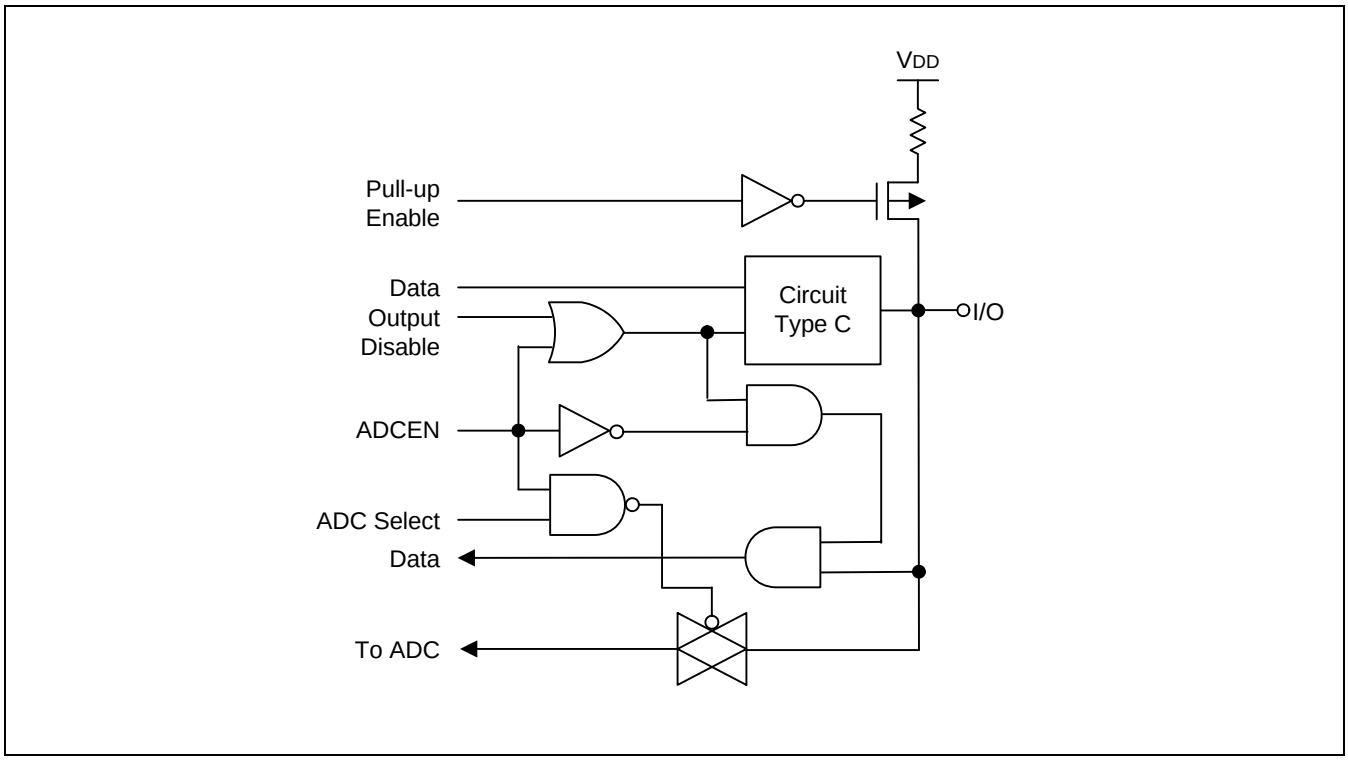


图 1-8 管脚电路类型 F-16 (P0)

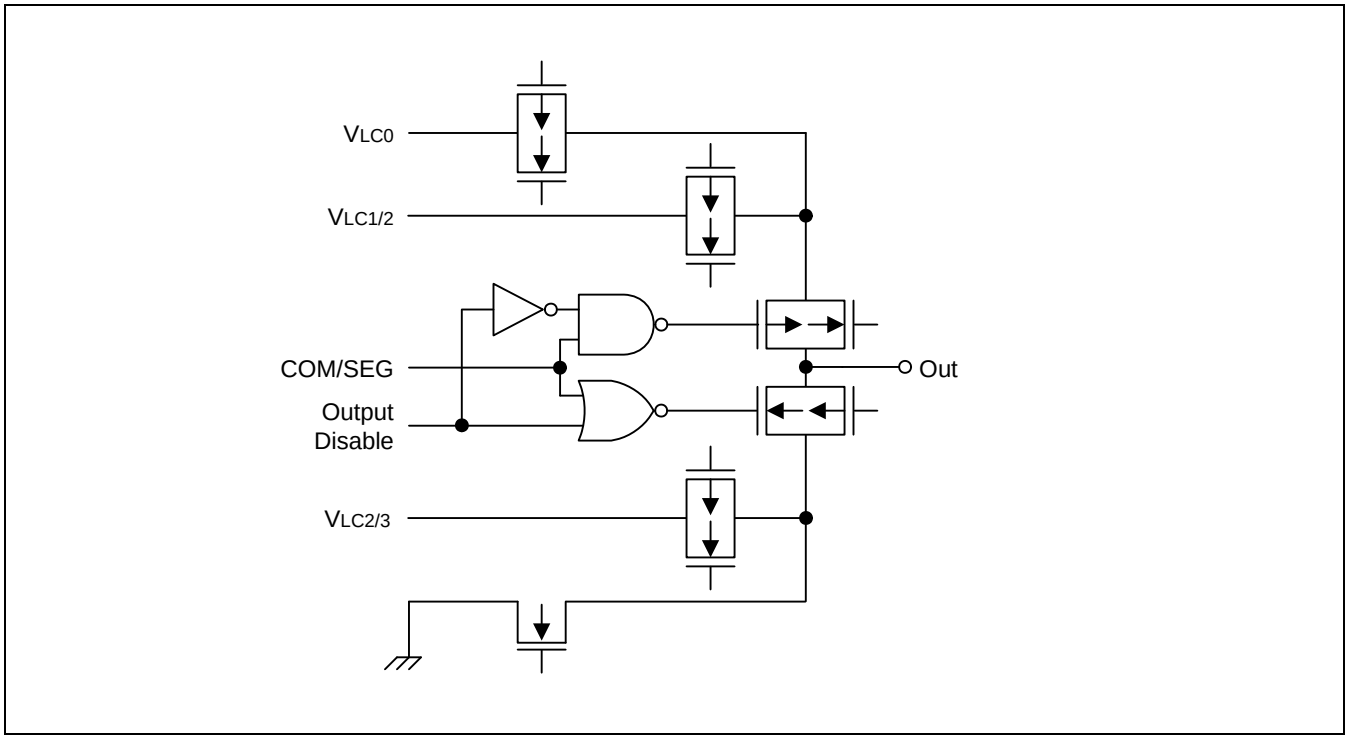


图 1-9 管脚电路类型H-39

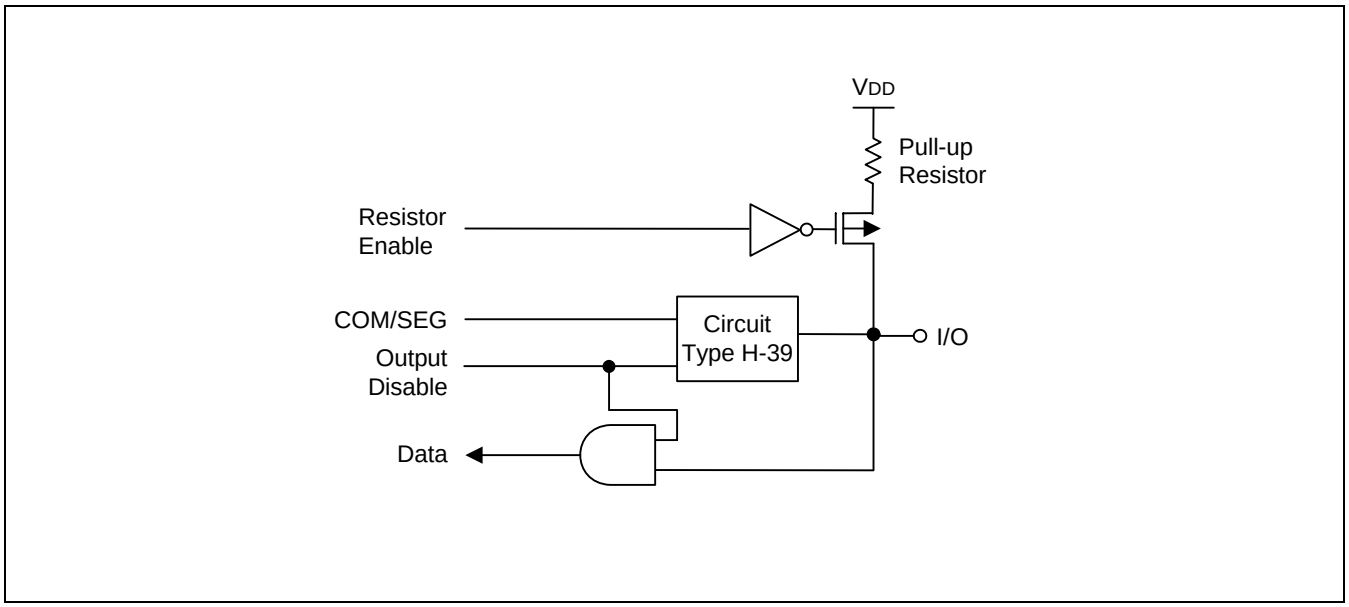


图 1-10 管脚电路类型 H-44 (P2)

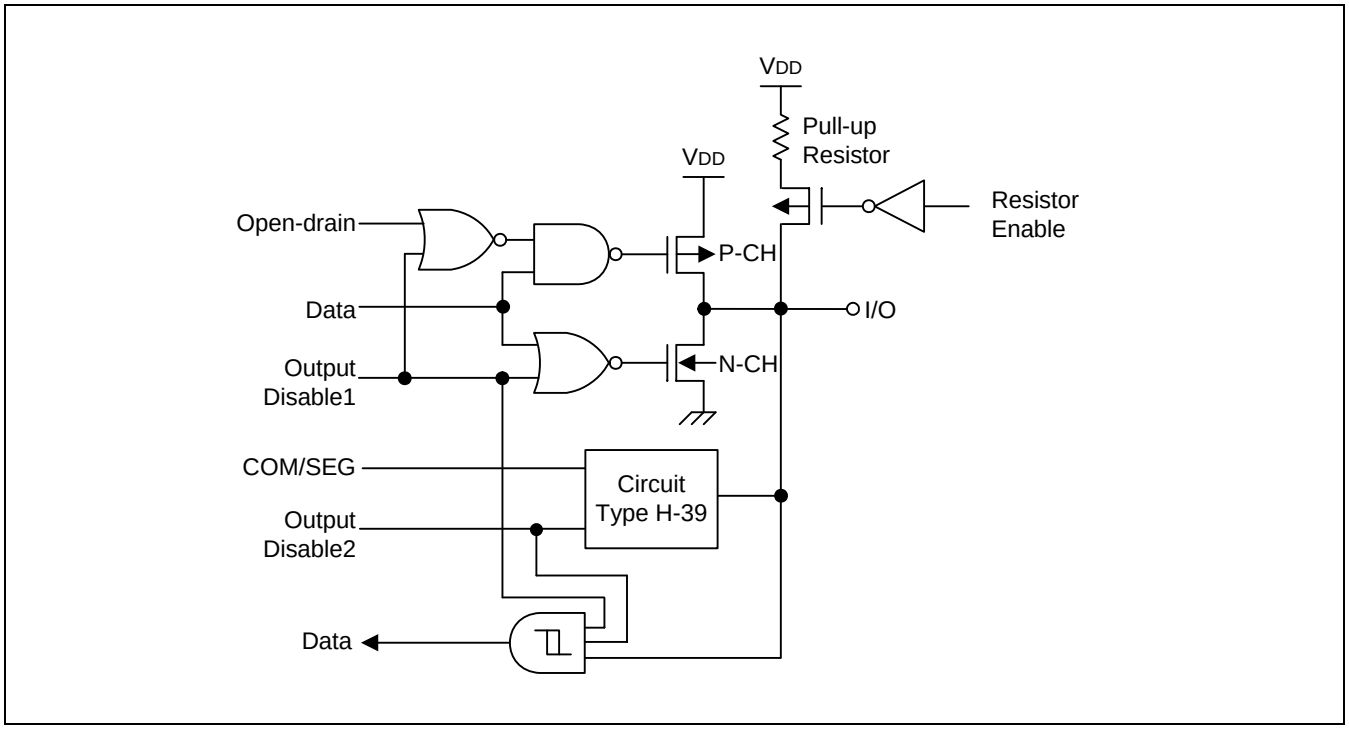


图 1-11 管脚电路类型 H-41 (P3)

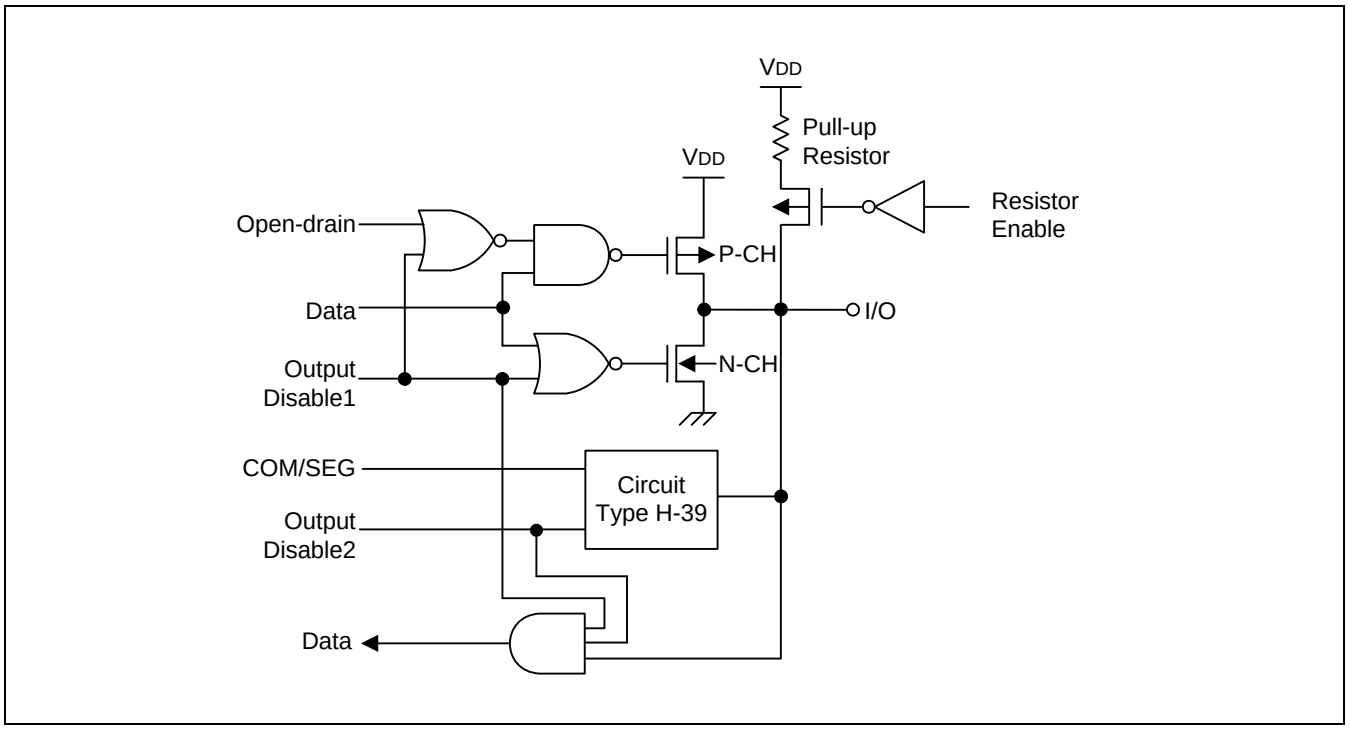


图 1-12 管脚电路类型 H-42 (P4)

2 地址空间

2.1 概述

S3F84UA/F84U8 MCU 有两类地址空间：

- 内部程序存储空间(ROM)
- 内部寄存器卷(RAM)

MCU 通过 16 位的地址总线访问程序存储空间。CPU 和内部寄存器卷之间通过独立的 8 位地址线和数据线联系。

S3F84UA 有内部 48K 字节的 Flash ROM。S3F84U8 有内部 8K 字节的 Flash ROM。

这 256 字节的物理寄存器空间扩展为支持寄存器寻址模式的 320 字节可寻址空间，另外还包括 22 字节的 LCD 显示寄存器卷。

2.2 程序存储空间 (ROM)

程序存储空间 (ROM) 保存程序代码或表格数据。S3F84UA 有 48K 字节的内部 Flash 程序存储空间，S3F84U8 有 8K 字节的内部 Flash 程序存储空间。

ROM (0H–0FFH)中初始 256 字节预留作中断入口地址。除去 3CH，3DH，3EH 和 3FH 外，
未使用的地址(0002H–00FFH)
都可用作普通的程序存储空间。在使用中断地址区域作为程序代码空间时，注意不可覆盖中断入口地址。

复位后，S3F84UA/F84U8 的 ROM 地址中程序从 0100H 处开始执行。

S3F84UA/F84U8(Full-Flash 芯片)的 ROM 复位地址可通过 Smart Option 修改。详细内容请参考第 21 章：嵌入式闪存接口。

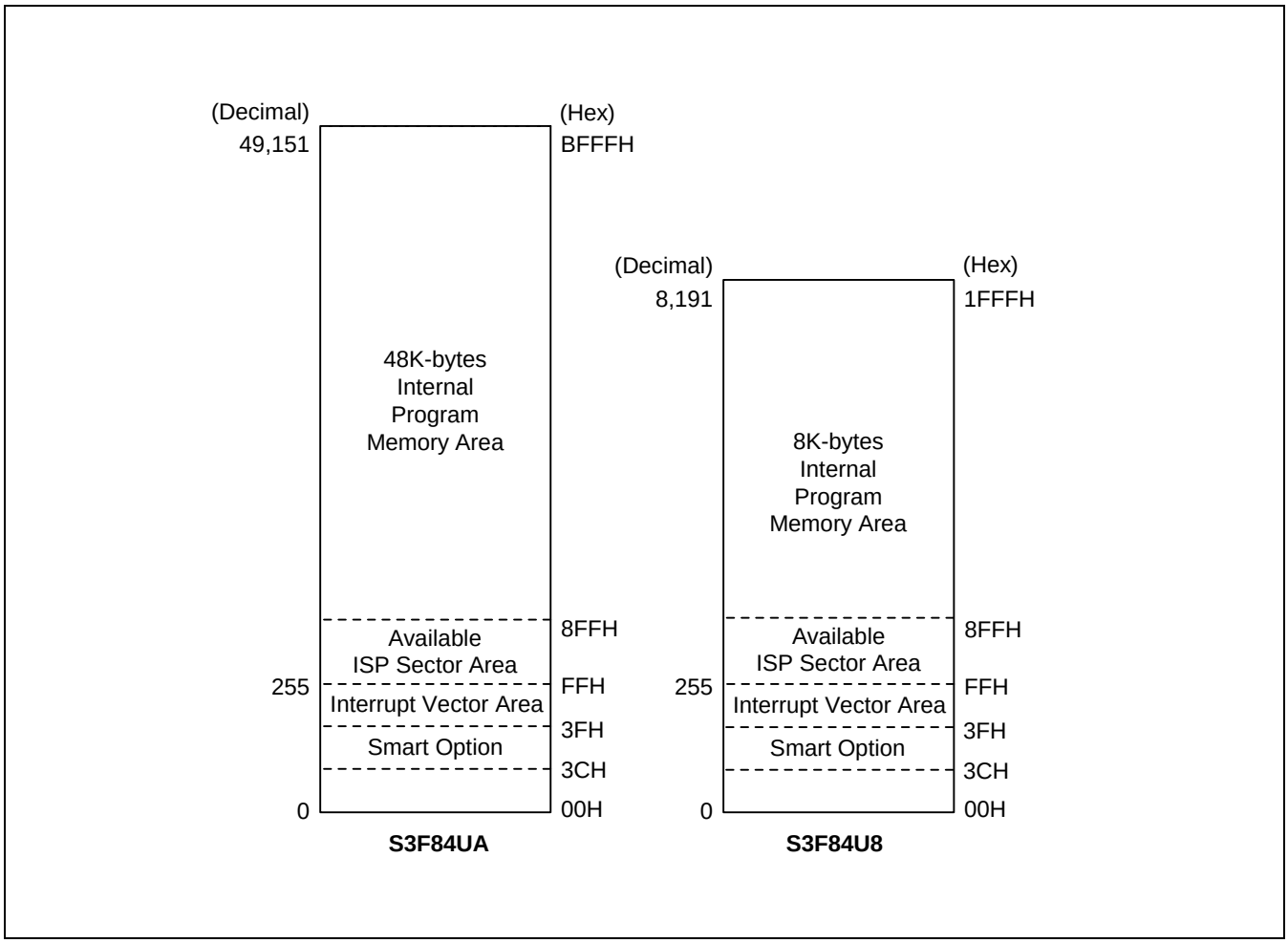
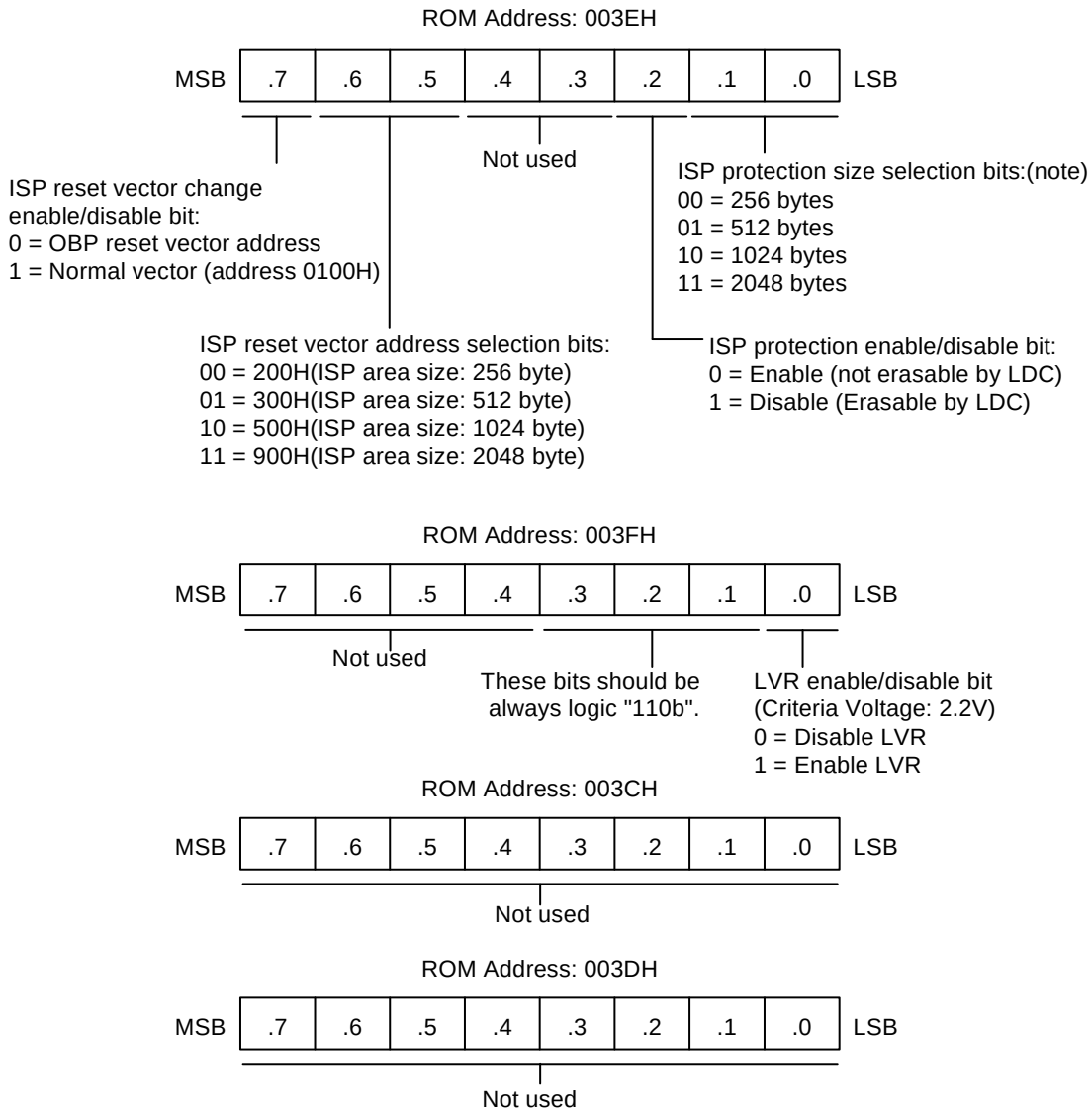


图 2-1 程序存储地址空间

2.2.1 SMART OPTION



注释：

1. 在选择 ISP 保护区域选择好 ISP 复位向量地址后，不要选择超过 ISP 空间大小。
2. 当通过 LDC 指令往 Smart Option 空间（003CH-003FH)）写任何值时，该空间的值可能会被更改，但 Smart Option 不受影响。需要使用 OTP/MTP 工具（SPW2+ 单编程器或 GW-PRO2 群编程器）往 Smart Option 空间写 Smart Option 的数据。

图 2-2 Smart Option

Smart Option 是 ROM 中用于芯片启动的选项。ROM 中的 003CH 至 003FH 地址为 Smart Option。S3F84UA/F84U8 仅使用 003EH 至 003FH。

2.3 寄存器结构

在 S3F84UA/F84U8 中，寄存器卷的高 64 字节被扩展为两个 64 字节区域，叫做 set 1 和 set 2。Set 1 的高 32 字节区域又进一步扩展为两个 32 字节寄存器块 (Bank 0 和 Bank 1)，低 32 字节是独立的公共存储区域。

S3F84UA 共有 631 个可寻址的 8 位寄存器。其中 13 个字节用于 CPU 和系统控制寄存器，22 个字节用于 LCD 数据寄存器，68 个字节用于外设控制和数据寄存器，16 个字节用于工作寄存器，还有 512 个寄存器作为通用寄存器(Page 0-Page 1)来使用(对 S3F84U8 仅有 Page 0 可供使用)。

不管当前选择的是这 10 个寄存器页中的那一页，用户都可以对 Set 1 的寄存器进行寻址。但只能通过寄存器寻址模式进行寻址。

通过不同寻址模式的限制，Bank 选择指令 SB0 和 SB1 以及寄存器页指针(PP)，得以将寄存器空间扩展为各个独立的可寻址区域(Sets，Banks 和 Pages)。

表 2-1 中总结了内部寄存器卷中特殊寄存器类型和所占字节数。

表 2-1 S3F84UA 寄存器类型总结

寄存器类型	所占字节数
通用寄存器 (包括 16 字节公共工作寄存器空间，两个192 字节基本寄存器空间和两个 64 字节 Set 2 空间) LCD 数据寄存器 CPU 和系统控制寄存器 映射的时钟，外设，I/O 控制和数据寄存器	528 22 13 68
所有可寻址的字节数	631

表 2-2 S3F84U8 寄存器类型总结

寄存器类型	所占字节数
通用寄存器 (包括 16 字节公共工作寄存器空间，一个192 字节基本寄存器空间和一个 64 字节 Set 2 空间) LCD 数据寄存器 CPU 和系统控制寄存器 映射的时钟，外设，I/O 控制和数据寄存器	272 22 13 68
所有可寻址的字节数	375

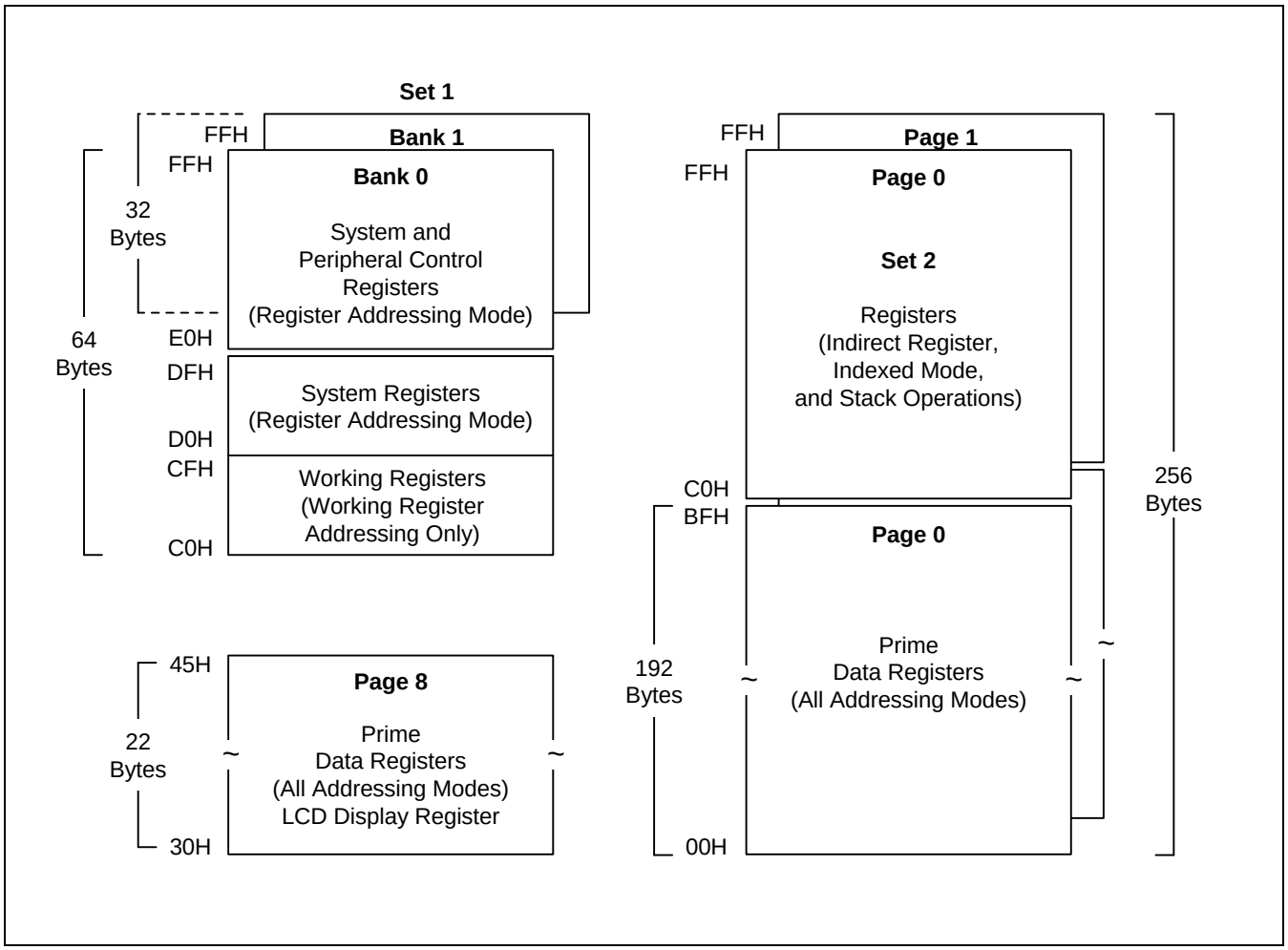


图 2-3 内部寄存器卷的组织结构 (S3F84UA)

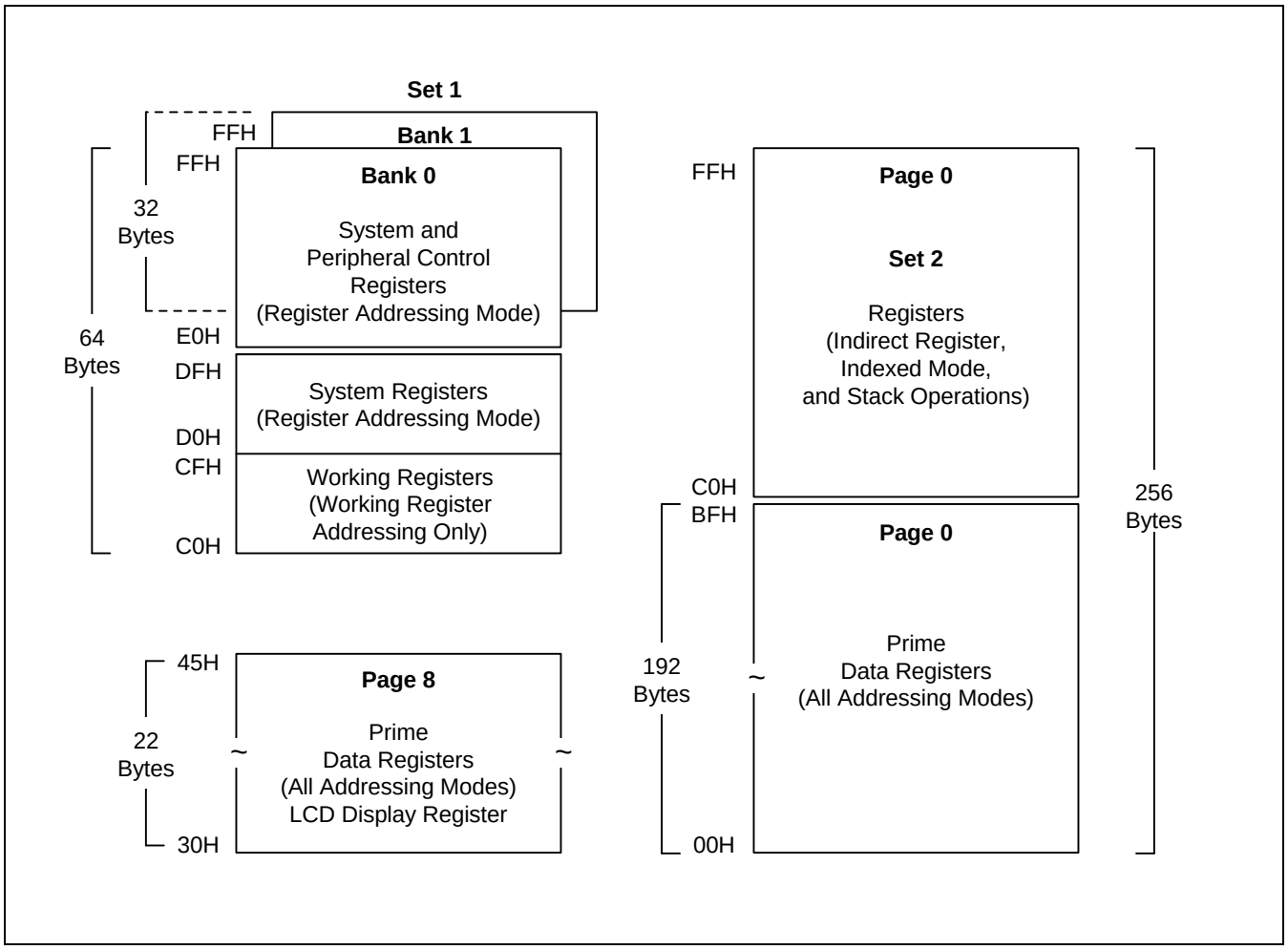


图 2-4 内部寄存器卷的组织结构 (S3F84U8)

2.3.1 寄存器页 (PP)

S3C8- 系列 MCU 架构支持物理上 256 字节的内部寄存器卷进行逻辑扩展(通过 8 位数据总线)，最多可实现16个可独立寻址的寄存器页。页寻址由寄存器页指针(PP，地址：DFH)来控制。S3F84UA/F84U8 MCU 中专门页中的寄存器卷用作扩展 LCD 数据寄存器，此时寄存器页指针必须从其他页改到指向该页。

复位之后，页面指针的源页(低四位)和目标页(高四位)数值缺省为“0000B”，自动选择页面 0 作为源和目标页来进行寄存器寻址。

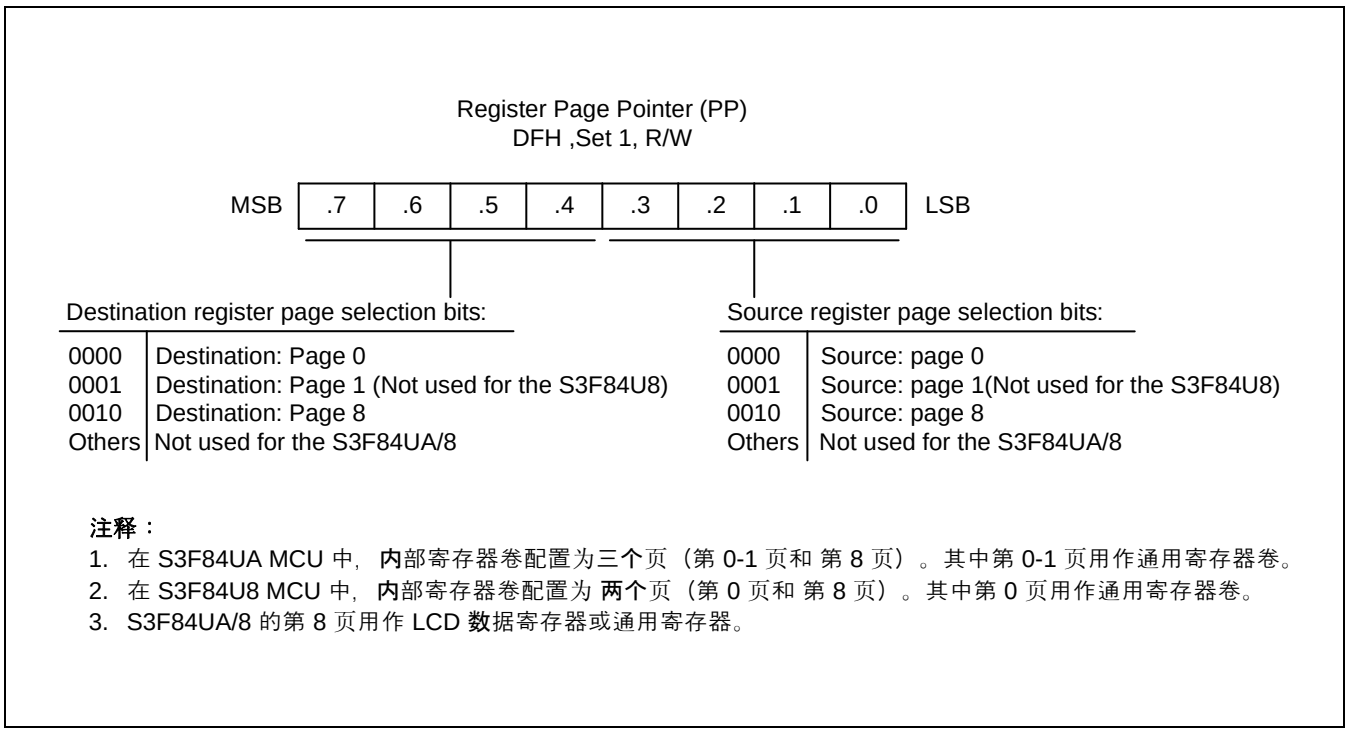


图 2-5 寄存器页指针 (PP)

编程实例 2-1 在清 RAM 时使用页指针 (第 0 页, 第 1 页)

```
            LD      PP,#00H             ; 目标 ← 0, 源 ← 0
            SRP     #0C0H
            LD      R0,#0FFH            ; 开始对第 0 页的 RAM 清零
RAMCL0:     CLR     @R0
            DJNZ    R0,RAMCL0
            CLR     @R0                 ; R0 = 00H

            LD      PP,#10H             ; 目标 ← 1, 源 ← 0
            LD      R0,#0FFH            ; 开始对第 1 页的 RAM 清零
RAMCL1:     CLR     @R0
            DJNZ    R0,RAMCL1
            CLR     @R0                 ; R0 = 00H
```

注释: 当程序中使用到 DJNZ 指令时，用户必须参考第 6-39 页内容正确使用 DJNZ 指令。

2.3.2 寄存器 SET 1

Set 1 指的是寄存器卷中的高 64 字节，地址为 C0H–FFH。

这 64 字节空间(E0H–FFH)的高 32 字节又被扩展为两个 32 字节的寄存器块，Bank 0 和 Bank 1。通过指令 SB0 或 SB1 来访问 Bank 0 或 Bank 1。硬件复位后默认选择 Bank 0 进行寻址。

Set 1(E0H–FFH)的两个高 32 字节区域(Bank0 和 Bank1)包含 64 个系统寄存器(D0H–DFH)和 16 个公共工作寄存器(C0H–CFH)。在其它寄存器空间执行的数据操作，用户也可以使用工作寄存器作为“扩展的”空间。

Set 1 中的寄存器可以随时通过寄存器寻址模式进行访问。
16 字节的工作寄存器区域，则只能使用工作寄存器寻址模式(更多关于工作寄存器寻址的信息，请查阅第 3 章,“寻址模式”)。

2.3.3 寄存器 SET 2

对 Set 1 的 64 字节地址空间(C0H-FFH)进行逻辑复制,以增加额外的 64 字节寄存器空间。
这个扩展的空间被称为 Set 2。在 S3F84UA 中,Page 0-1 可对 Set 2 的地址空间(C0H-FFH)进行寻址。
在 S3F84U8 中仅Page 0 对 Set 2 的地址空间(C0H-FFH)进行寻址。

通过寻址模式的限制，得以实现 Set 1 和 Set 2 的逻辑分割。Set 1 只支持寄存器寻址模式，而对 Set 2 的寻址只能采用寄存器间接寻址模式或偏址寻址模式。

Set 2 寄存器区常用作堆栈操作。

2.3.4 主寄存器空间

在 S3F84UA/F84U8 中，两个或一个 256 字节寄存器页的低 192 字节(00H–BFH)称为主寄存器空间。主寄存器空间可通过七种寻址模式中的任何一种进行访问(见第 3 章，“寻址模式”)。

芯片复位后，可立即对 Page 0 的主寄存器空间进行寻址。为了在其它页(Page 0-1 和 Page 8)访问主寄存器空间，必须对寄存器页面指针(PP)的源页和目标页进行正确设置。

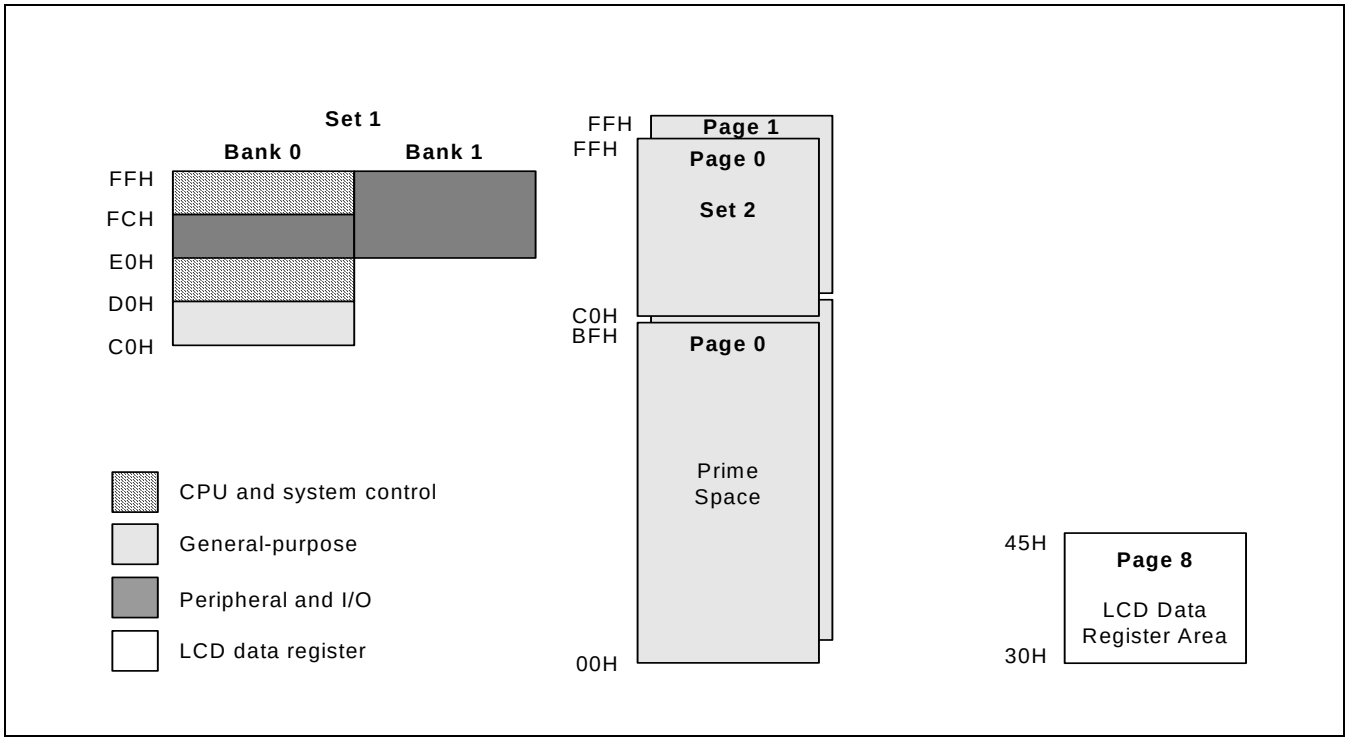


图 2-6 Set 1, Set 2, 主寄存器空间和 LCD 数据寄存器映射图

2.3.5 工作寄存器

指令可以使用 4 位或 8 位地址来访问指定的 8 位寄存器或者 16 位寄存器对。
当使用 4 位工作寄存器寻址模式时，256 字节的寄存器卷可以被看成是由 32 个 8 字节寄存器组或“片(Slice)”组成的。每个片包含 8 个 8 位寄存器。

通过两个 8 位寄存器指针 RP1 和 RP0，可以随时选择两个工作寄存器片，组成一个 16 字节的工作寄存器块。使用这两个指针，用户可以将 16 字节的寄存器块移动到除 Set 2 以外的任何可寻址空间。

在本手册中，术语“片(Slice)”和“块(Block)”用来帮助理解工作寄存器空间的大小和相对位置:

- 一个工作寄存器“片”是 8 个字节(8 个 8 位工作寄存器，R0–R7 或 R8–R15)
- 一个工作寄存器“块”是 16 个字节(16 个 8 位工作寄存器，R0–R15)

所有 8 字节的工作寄存器“片”中，寄存器地址的高 5 位数值完全相同。
这使得各个寄存器指针都可以指向寄存器卷中的 24 个寄存器“片”中任何一个。寄存器指针 RP0 和 RP1 中存放的是选定的两个 8 位寄存器“片”的起始地址。

系统复位后，RP0 和 RP1 总是指向 Set 1 中的 16 字节公用空间(C0H-CFH)。

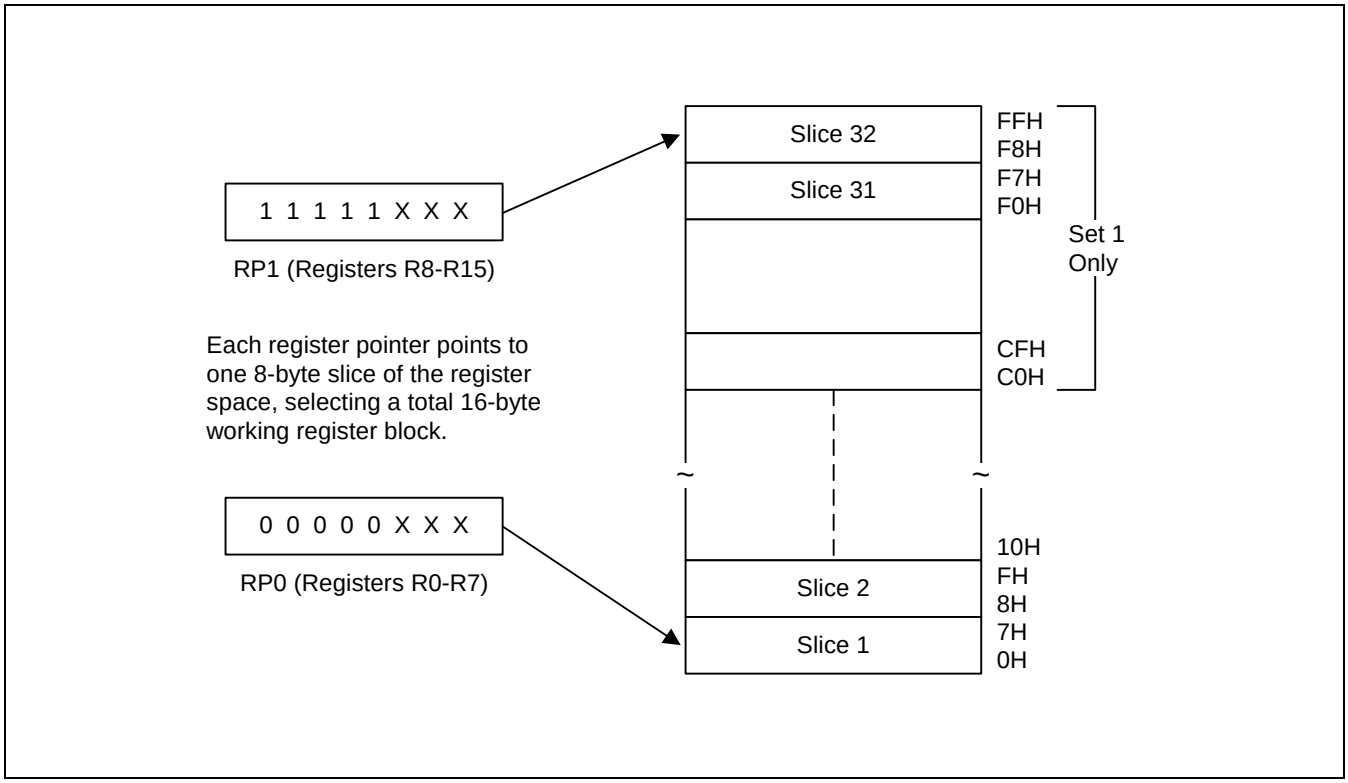


图 2-7 8 字节工作寄存器区 (Slices)

2.3.6 使用寄存器指针

寄存器指针 RP0 和 RP1(地址：D6H 和 D7H，Set 1)，用于选择寄存器卷中两个可移动的8字节工作寄存器片。复位后，他们指向工作寄存器的公共空间：RP0 指向地址 C0H-C7H，RP1指向C8H-CFH。

用户通过 SRP 或者 LD 指令可以改变寄存器 RP0/RP1 的值(*图 2-8* 和 *图 2-9*)。

用工作寄存器寻址方式，用户只能访问当前 RP0 和 RP1 指定的寄存器卷中的两个 8 字节的工作寄存器片。但不能用寄存器指针来选择 Set 2 (C0H–FFH) 中的工作寄存器，
因为这些地址空间只支持间接寄存器寻址模式或偏址寻址模式。

16 字节的工作寄存器块通常由两个相邻的 8 字节工作寄存器片组成。编程时，一般建议将 RP0 指向“低”地址的片，而 RP1 指向“高”地址的片(*图 2-8*)。
某些情况下， 可能需要将工作寄存器定义在不同的(不相邻的)寄存器空间中(*图 2-9*)，RP0 指向“高”地址的片，而 RP1 指向“低”地址的片。

由于寄存器指针可以指向工作寄存器块中的两个 8 位工作寄存器片中的任意一个，
用户可根据程序需求灵活定义工作寄存器空间。

编程实例 2-2 设置寄存器指针

```
SRP    #70H             ; RP0 ← 70H, RP1 ← 78H
SRP1   #48H             ; RP0 ← 不变, RP1 ← 48H,
SRP0   #0A0H            ; RP0 ← A0H, RP1 ← 不变
CLR    RP0              ; RP0 ← 00H, RP1 ← 不变
LD     RP1,#0F8H        ; RP0 ← 不变, RP1 ← 0F8H
```

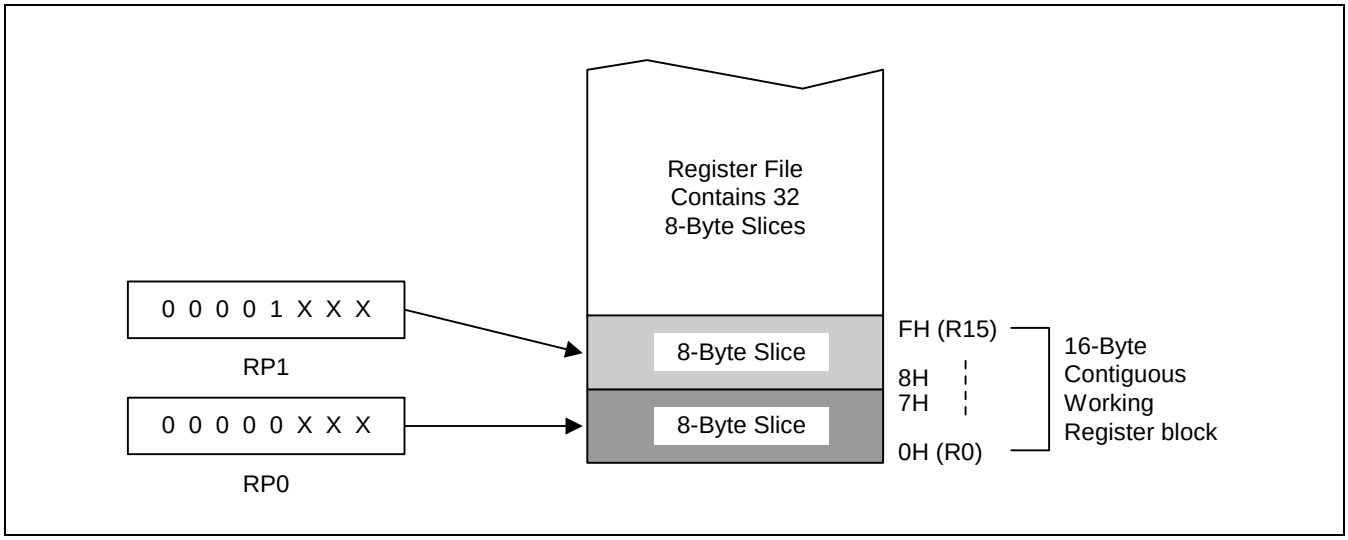


图 2-8 相邻的 16 字节工作寄存器块

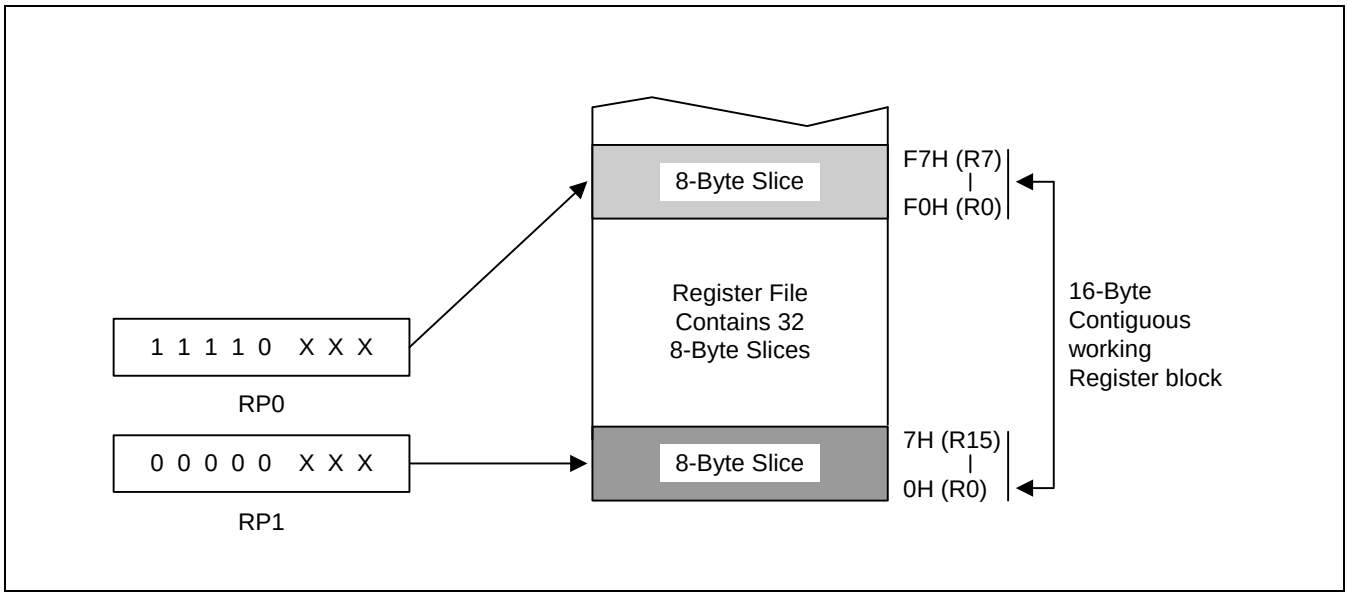


图 2-9 非相邻的 16 字节工作寄存器块

编程实例 2-3 使用 RPs 对多个寄存器的内容求和

用寄存器指针来对寄存器 80H–85H 中的内容求和。寄存器 80H-85H 中的存放数据分别为 10H，11H，12H，13H，14H和15H：

```
SRP0  #80H        ; RP0 ← 80H
ADD   R0,R1       ; R0 ← R0 + R1
ADC   R0,R2       ; R0 ← R0 + R2 + C
ADC   R0,R3       ; R0 ← R0 + R3 + C
ADC   R0,R4       ; R0 ← R0 + R4 + C
ADC   R0,R5       ; R0 ← R0 + R5 + C
```

6 个寄存器的和 (6FH) 存放在 R0（80H）中。例子中的指令共占用 12 字节的代码长度，执行时间为 36 个时钟周期。如果不用寄存器指针，那就得按照下面的指令顺序来做：

```
ADD   80H,81H     ; 80H ← (80H) + (81H)
ADC   80H,82H     ; 80H ← (80H) + (82H) + C
ADC   80H,83H     ; 80H ← (80H) + (83H) + C
ADC   80H,84H     ; 80H ← (80H) + (84H) + C
ADC   80H,85H     ; 80H ← (80H) + (85H) + C
```

现在，6 个寄存器的和依然放在 80H 当中，但是所有指令总共占用了 15 字节的代码长度而非 12 字节，且执行时间为 50 个时钟周期而非 36 个。

2.4 寄存器寻址

S3C8- 系列单片机的寄存器结构提供了一种效率高的工作寄存器寻址方式，
该方式充分利用了短指令格式以缩短程序执行时间。

在寄存器寻址模式中，操作数存放在某个特定的寄存器或寄存器对中。用寄存器寻址模式可以访问寄存器空间中除 Set 2 以外的的任何地址。使用工作寄存器寻址时，通过寄存器指针指定一个 8 字节的工作寄存器空间和该空间内的一个 8 位寄存器。

寄存器寻址时，既可以视其为单独的 8 位寄存器，也可以视为成对的 16位寄存器空间。在 16位寄存器对中，第一个 8 位寄存器的地址总是偶数，而下一个寄存器地址则是奇数。16位数据 的高位字节存放在偶地址寄存器中，低位字节存放在邻近的 (+1) 奇地址寄存器中。

工作寄存器寻址模式与寄存器寻址模式的不同之处在于，它通过寄存器指针来指定一个 8 字节工作寄存器空间，以及其中的某个 8 位工作寄存器。

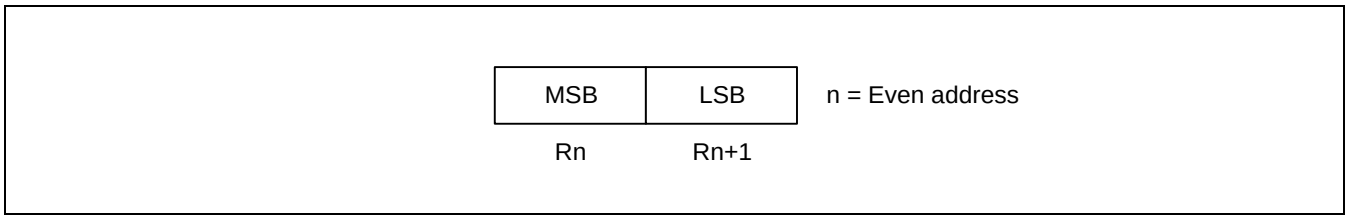


图 2-10 16 位寄存器对

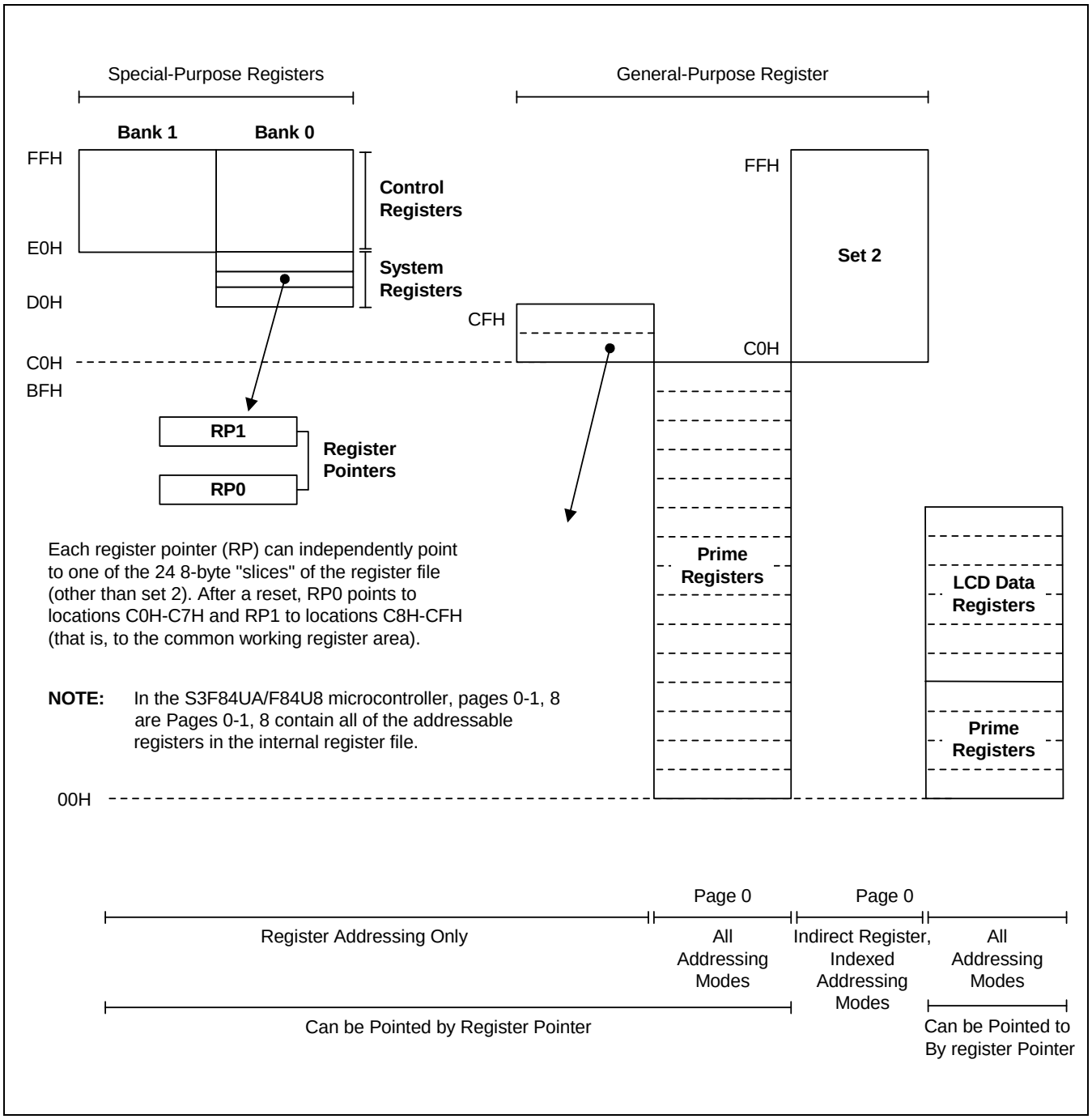


图 2-11 寄存器卷寻址模式

2.4.1 通用工作寄存器区 (C0H–CFH)

系统复位后，寄存器指针 RP0 和 RP1 自动指向 Set 1 中两个 8 字节的寄存器片(C0H-CFH)，组成 16 字节的寄存器块：

RP0 → C0H–C7H

RP1 → C8H–CFH

这 16 字节的地址空间称为通用工作寄存器区。就是说无论当前操作所要访问的是哪个页面，都可以把该空间的寄存器作为工作寄存器使用。典型地，在不同页面间进行数据交换操作时，可使用工作寄存器作为临时存储。

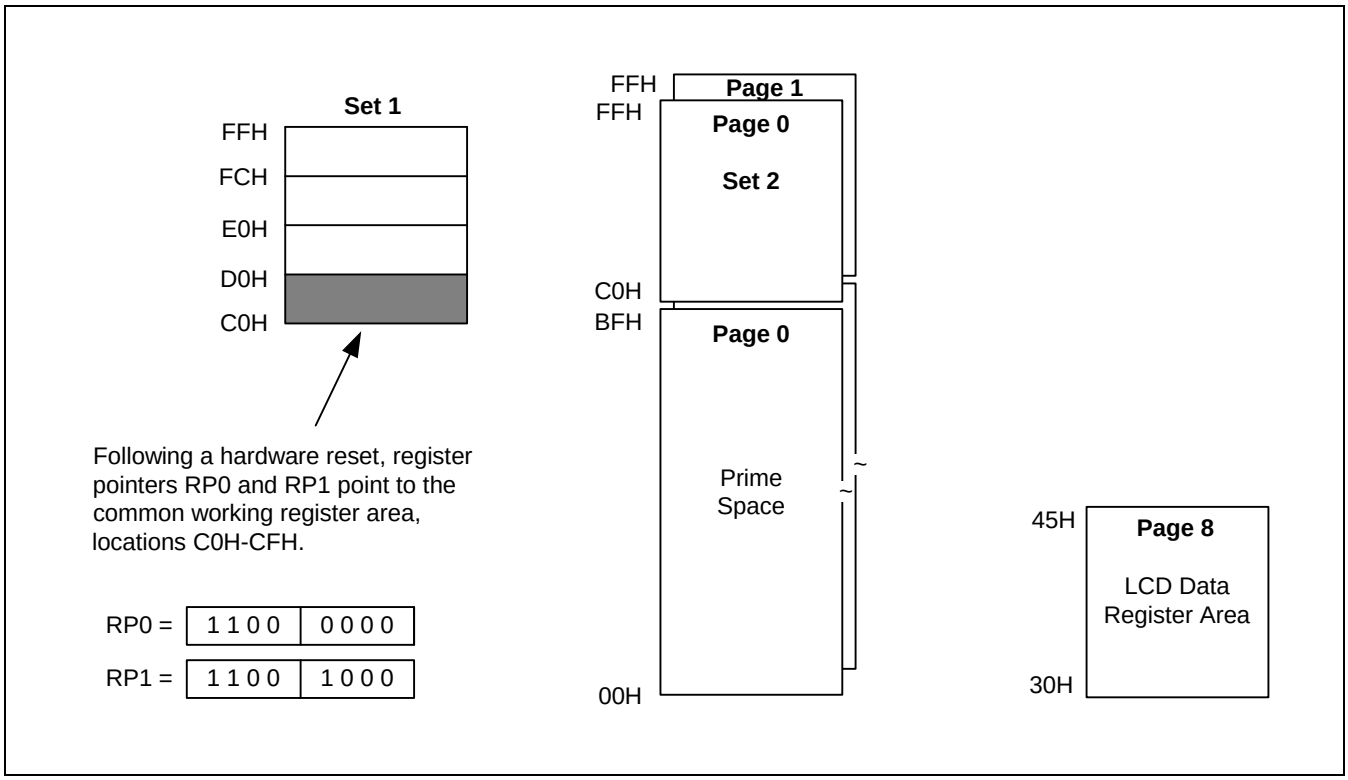


图 2-12 通用工作寄存器区

编程实例 2-4 访问通用工作寄存器区

如下例所示，要访问通用工作寄存器(C0H-CFH)，只能采用工作寄存器寻址模式：

```
例    1.    LD     0C2H,40H          ; 非法寻址模式!

            改为工作寄存器寻址模式：

            SRP    #0C0H
            LD     R2,40H            ; R2 (C2H) ← 地址 40H 中存放的值

      2.    ADD    0C3H,#45H         ; 非法寻址模式!

            改为工作寄存器寻址模式：

            SRP    #0C0H
            ADD    R3,#45H           ; R3 (C3H) → R3 + 45H
```

2.4.2 4 位工作寄存器寻址模式

每个寄存器指针定义了一个可移动的 8 字节寄存器片，其中寄存器指针中存储的地址信息作为一扇寻址的“窗”，使得指令只须 4 位地址就可以实现对工作寄存器的有效访问。
在工作寄存器寻址时，8 位地址是采用下述方法构成的：

- 4 位地址的最高位选择一个寄存器指针(“0”选择 RP0，“1”选择 RP1)
- 寄存器指针的高 5 位选择寄存器卷中的某个 8 字节寄存器片
- 指令中 4 位地址的低 3 位选定 8 个寄存器片中的一个

图 2-13，操作的结果是，寄存器指针的高 5 位和指令地址的低 3 位一起组成完整的 8 位寄存器地址。
只要寄存器指针中保存的地址不变，指令中 4 位地址的低 3 位总是指向同一个 8 字节寄存器片。

图 2-14 给出了一个典型的 4 位工作寄存器寻址实例。指令“INC R6”的最高位是“0”，这将选择 RP0。RP0 的高 5 位(01110B)与指令中 4 位地址的低 3 位(110B)一起组成了寄存器地址 76H(01110110B)。

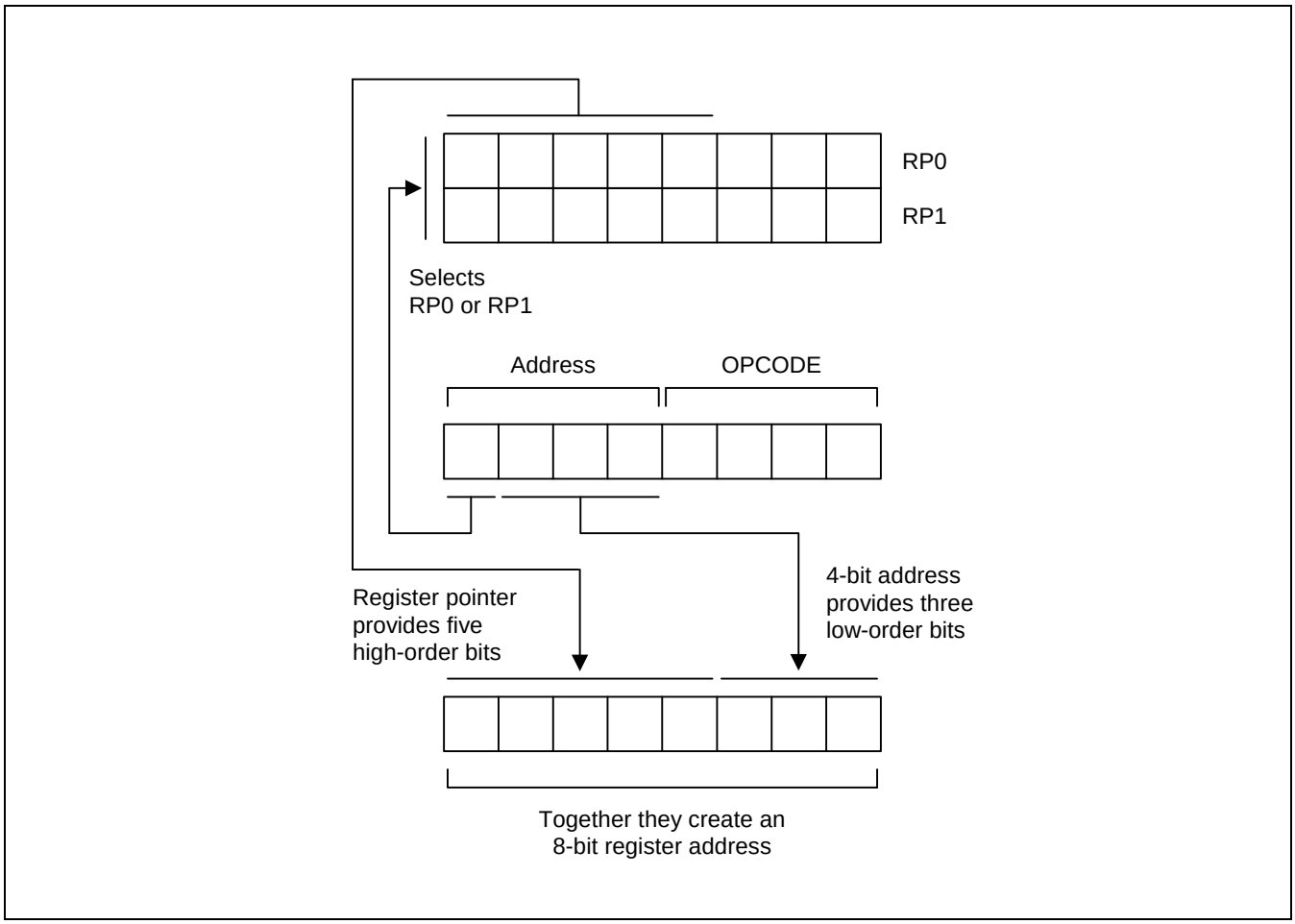


图 2-13 4 位工作寄存器寻址模式

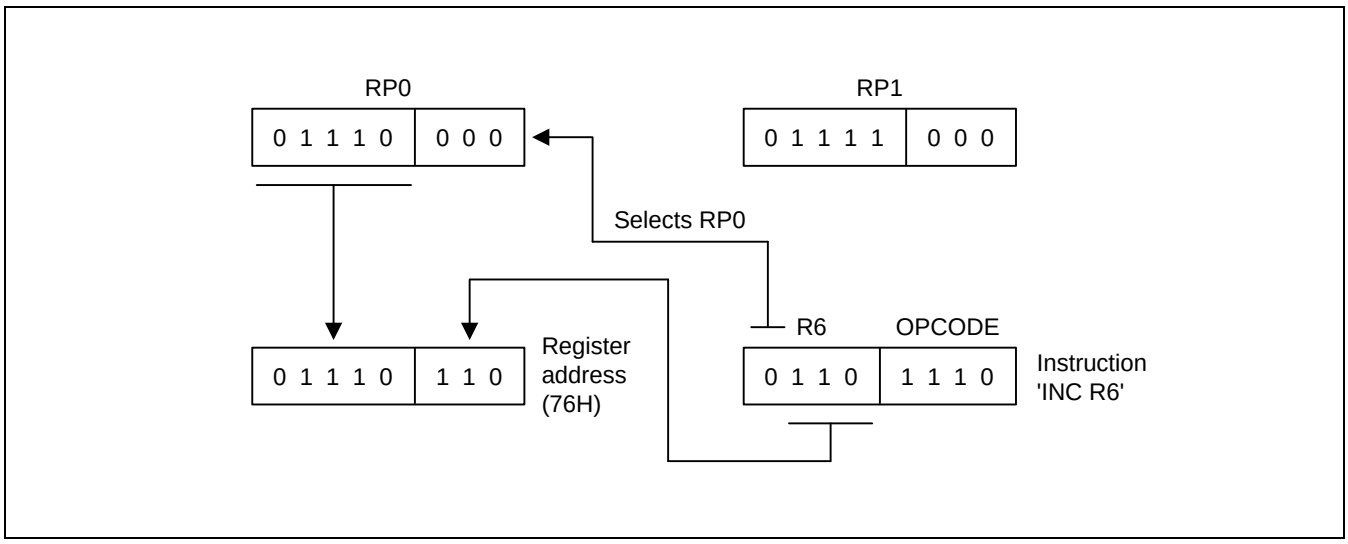


图 2-14 4 位工作寄存器寻址实例

2.4.3 8 位工作寄存器寻址模式

用户还可以通过 8 位工作寄存器寻址方式来访问工作寄存器区。首先，指令地址的高 4 位必须是“1100B”，这 4 位数据 (1100B) 表示余下的 4 位与 4 位工作寄存器寻址方式相同。

图 2-15，8 位地址的低阶位形成机制与 4 位工作寄存器寻址时类似：第 3 位选择 RP0 或 RP1，用来产生最终地址的高 5 位；而最终地址的低 3 位则由原始指令提供。

图 2-16 给出了一个典型的 8 位工作寄存器寻址实例。指令地址的高 4 位 (1100B) 表明寻址方式为 8 位寄存器寻址。第 4 位(“1”)选择 RP1，所以 RP1 中的高 5 位(10101B)成为寄存器地址的高 5 位，寄存器地址的低 3 位(011B)则由 8 位指令地址的低 3 位提供。RP1 中的 5 个地址位和指令中的 3 个地址位组合，形成了完整的寄存器地址，0ABH(10101011B)。

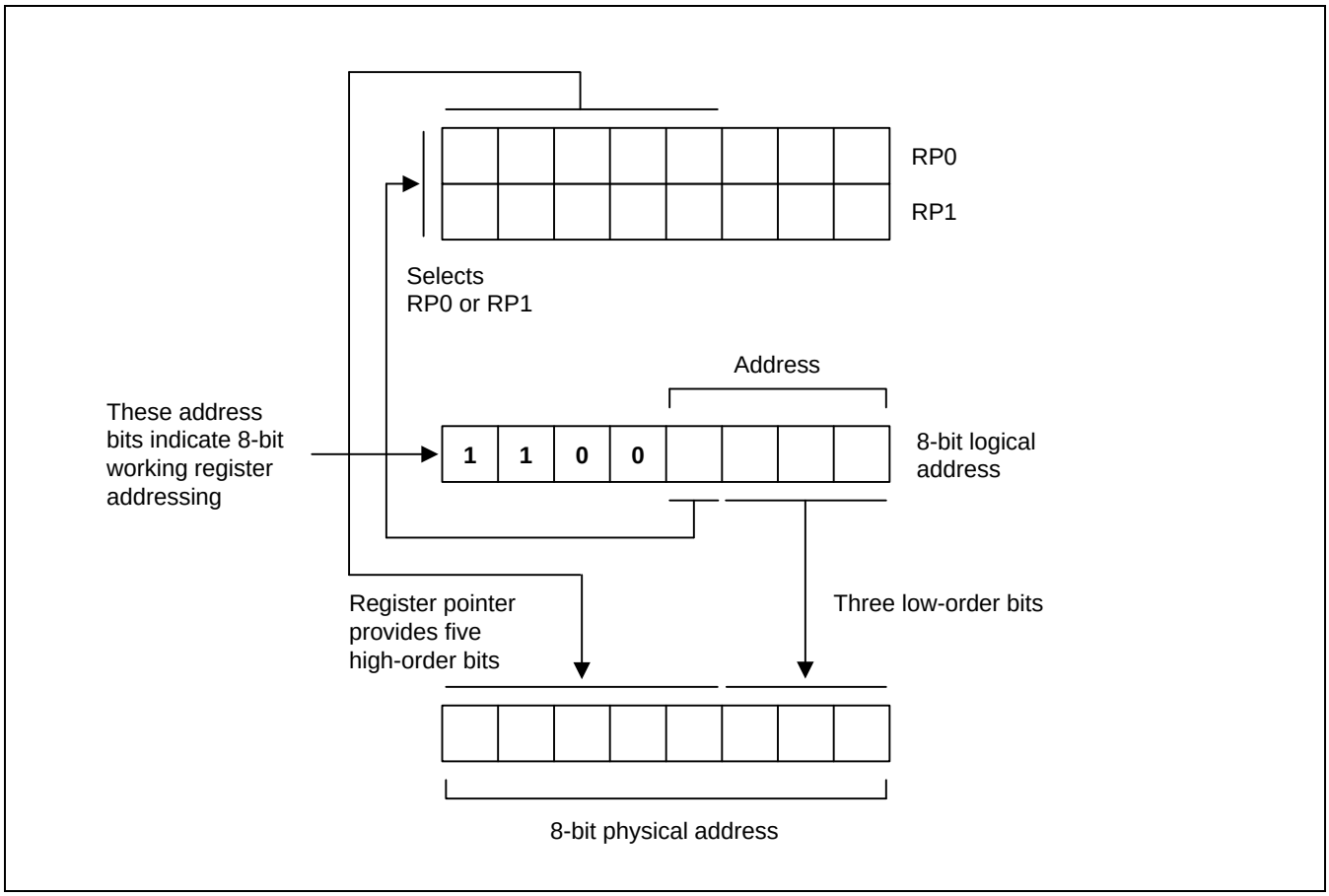


图 2-15 8 位工作寄存器寻址模式

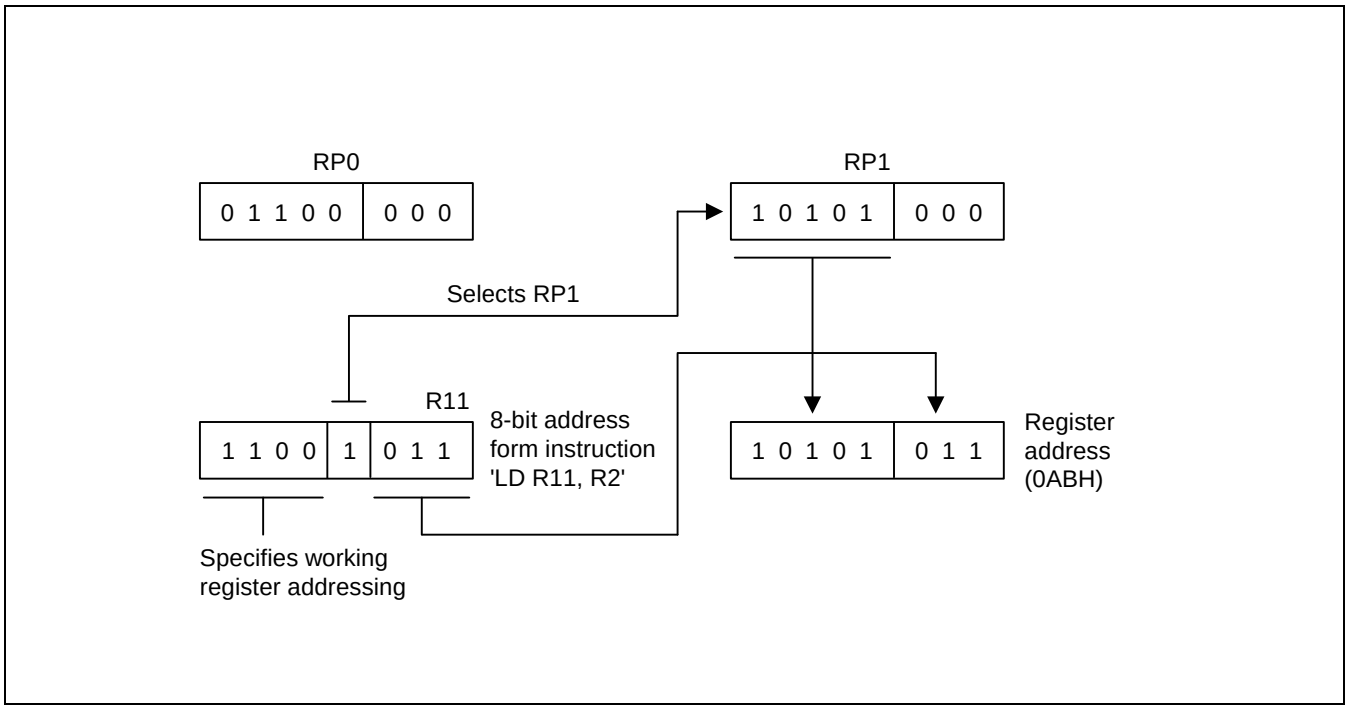


图 2-16 8 位工作寄存器寻址实例

2.5 系统和用户堆栈

S3C8- 系列 MCU 通过系统栈来处理数据存储，子程序调用和返回。PUSH 和 POP 指令用于控制系统栈操作。S3F84UA/F84U8 架构支持在内部寄存器卷中的栈操作。

2.5.1 栈操作

栈用来储存程序调用以及中断的返回地址，也用来储存数据。CALL 指令将 PC 的值压入栈，而 RET 指令将其从栈中弹出。当中断发生时，PC 和 FLAGS 寄存器的值被压入栈，指令 IRET 则将这些值弹出到它们原来的地址。栈指针总是在 PUSH 操作之前先减“1”，在 POP 操作之后加“1”。栈指针(SP)总是指向栈的顶端，*图 2-17* 所示。

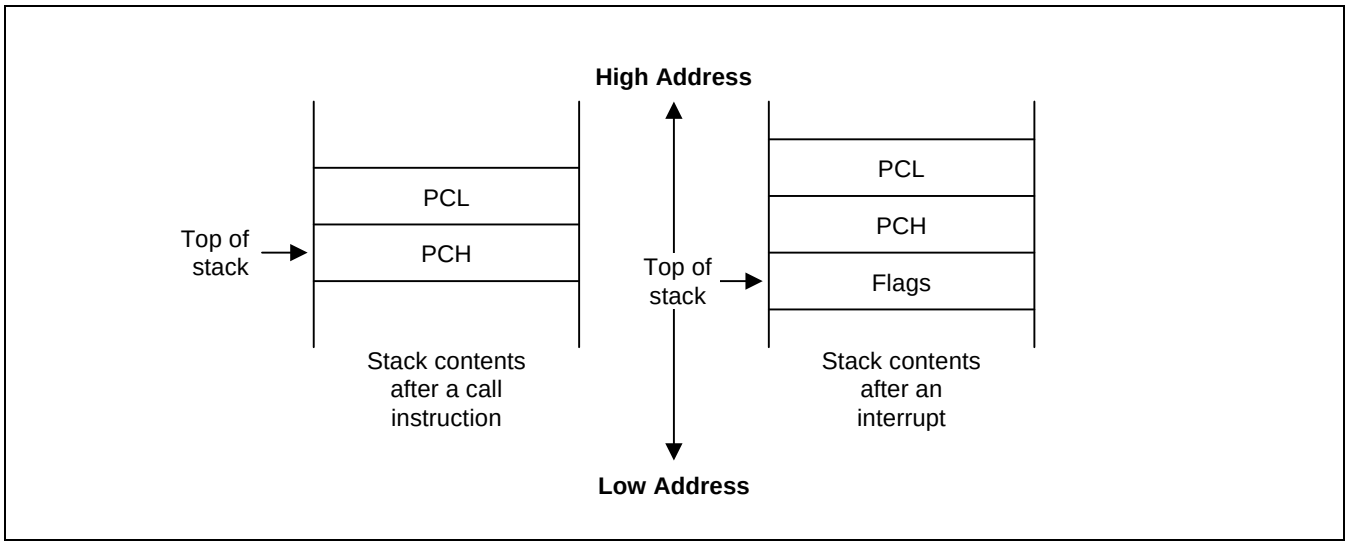


图 2-17 栈操作

2.5.2 用户自定义栈

用户可以在内部寄存器卷中堆栈自由定义栈数据储存区域。指令 PUSHUI，PUSHUD，POPUI 和 POPUD 支持用户自定义栈操作。

2.5.2.1 栈指针 (SPL, SPH)

寄存器地址 D8H 和 D9H 存放用于系统栈操作的 16 位栈指针(SP)。SP 的高字节地址，SP15–SP8，存放在 SPH 寄存器（D8H）中；低字节地址，SP7–SP0，存放在 SPL 寄存器（D9H）中。系统复位后，SP 的值不确定。

由于 S3F84UA/F84U8 只有内部存储空间，SPL 必须被初始化成介于 00H–FFH 之间的 8 位数值，而 SPH 则用不到，需要时可以把它作为通用寄存器使用。

当 SPH 寄存器用作通用数据寄存器时，如果因为 SPL 寄存器中正常的栈操作，如栈地址增加或减少而导致溢出发生，将会影响到 SPH 寄存器，并覆盖当前存储的数据。为了避免 SPH 寄存器被覆盖的情况发生，最好把 SPL 的初始值设为“FFH”而非“00H”。

编程实例 2-5 用 PUSH 和 POP 实现标准栈操作

下面的例子演示了在内部寄存器卷中如何通过 PUSH 和 POP 指令进行栈操作：

```
LD     SPL,#0FFH          ; SPL ← FFH
                          ; (通常 SPL 被程序初始化为 0FFH)
•
•
•
PUSH   PP                 ; 栈地址 0FEH ← PP
PUSH   RP0                ; 栈地址 0FDH ← RP0
PUSH   RP1                ; 栈地址 0FCH ← RP1
PUSH   R3                 ; 栈地址 0FBH ← R3
•
•
•
POP    R3                 ; R3 ← 栈地址 0FBH
POP    RP1                ; RP1 ← 栈地址 0FCH
POP    RP0                ; RP0 ← 栈地址 0FDH
POP    PP                 ; PP ← 栈地址 0FEH
```

3 寻址模式

3.1 概述

通过程序指针 (PC) 读取程序存储空间的指令然后执行。指令隐含着要进行的操作和操作数。
寻址模式是用于确定操作数地址的一种方法。SAM8RC 指令中的操作数可以是条件代码，立即数，或者寄存器卷，程序存储区，数据存储区的地址。

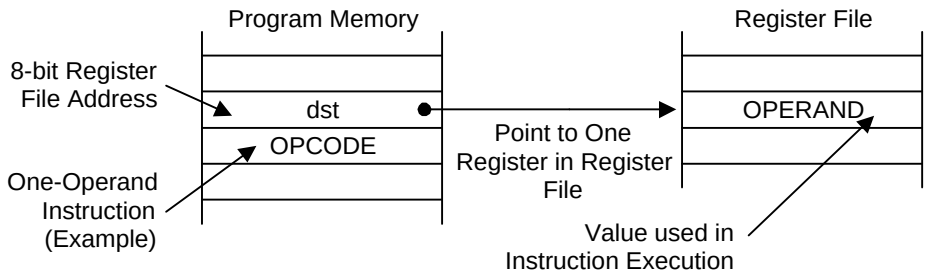
S3F8- 系列的指令集支持 7 种寻址模式，但这些寻址模式并非适用于所有指令。7 种寻址模式和它们的符号表示：

- 寄存器寻址模式 (R)
- 间接寄存器寻址模式 (IR)
- 偏址寻址模式 (X)
- 直接寻址模式 (DA)
- 间接寻址模式 (IA)
- 相对地址寻址模式 (RA)
- 立即数寻址模式 (IM)

3.1.1 寄存器寻址模式 (R)

寄存器寻址模式(R)中，操作数是具体寄存器或寄存器对中的内容，*图 3-1*。

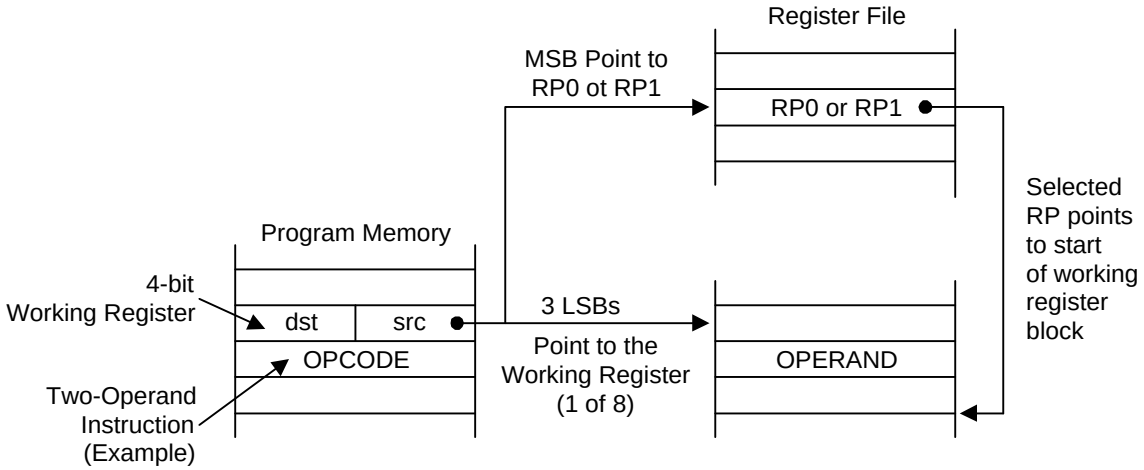
工作寄存器寻址模式与寄存器寻址模式不同。因为工作寄存器是通过一个 16 位的寄存器指针，来指定 8 字节的工作寄存器空间，和空间内的某个 8 位寄存器，*图 3-2*。



Sample Instruction:

```
DEC     CNTR            ;   Where CNTR is the label of an 8-bit register address
```

图 3-1 寄存器寻址模式



Sample Instruction:

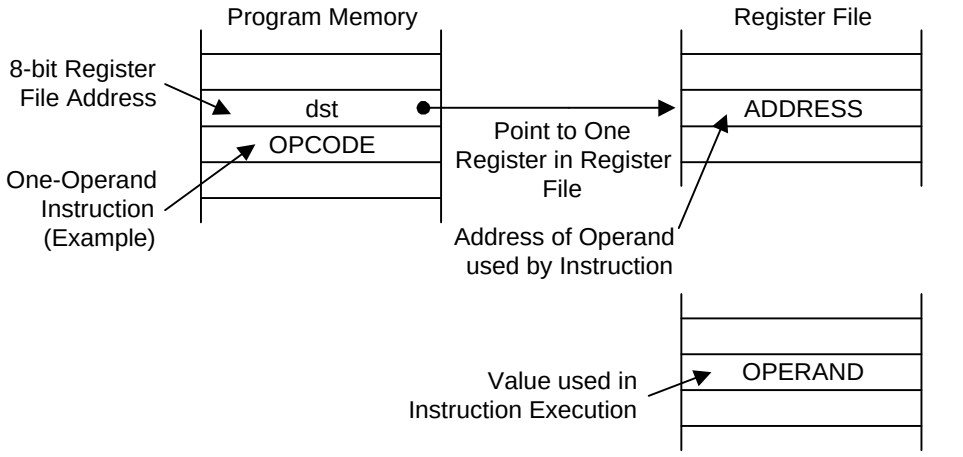
```
ADD     R1, R2          ;   Where R1 and R2 are registers in the currently
                            selected working register area.
```

图 3-2 工作寄存器寻址模式

3.1.2 间接寄存器寻址模式 (IR)

在间接寄存器寻址模式中，指定寄存器或寄存器对中存放的是操作数的地址。根据所用的指令，物理地址可能是寄存器卷中的寄存器、程序存储器(ROM)，或者外部数据存储器(*图 3-3* 至 *图 3-6*)。

可以用任意的 8 位寄存器来访问其它的寄存器，也可以用任意的 16 位寄存器组来访问其它的存储空间。
但要注意，不能通过间接寄存器寻址模式对 Set 1 中地址段 C0H-FFH 进行访问。



Sample Instruction:

```
RL      @SHIFT          ;   Where SHIFT is the label of an 8-bit register address
```

图 3-3 寄存器卷中的间接寄存器寻址模式

3.1.2.1 间接寄存器寻址模式 (续)

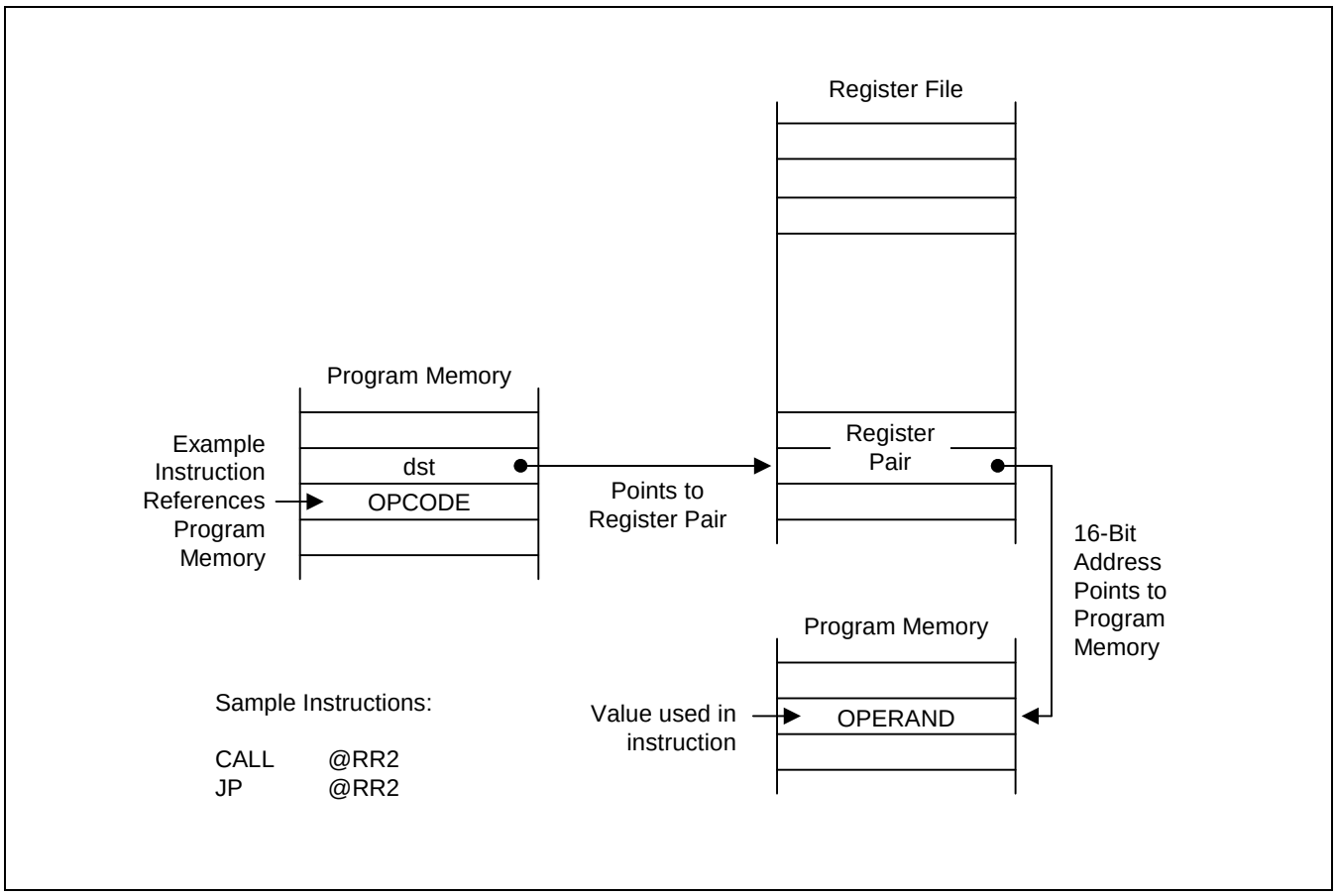


图 3-4 程序存储空间的间接寄存器寻址

3.1.2.2 间接寄存器寻址模式 (续)

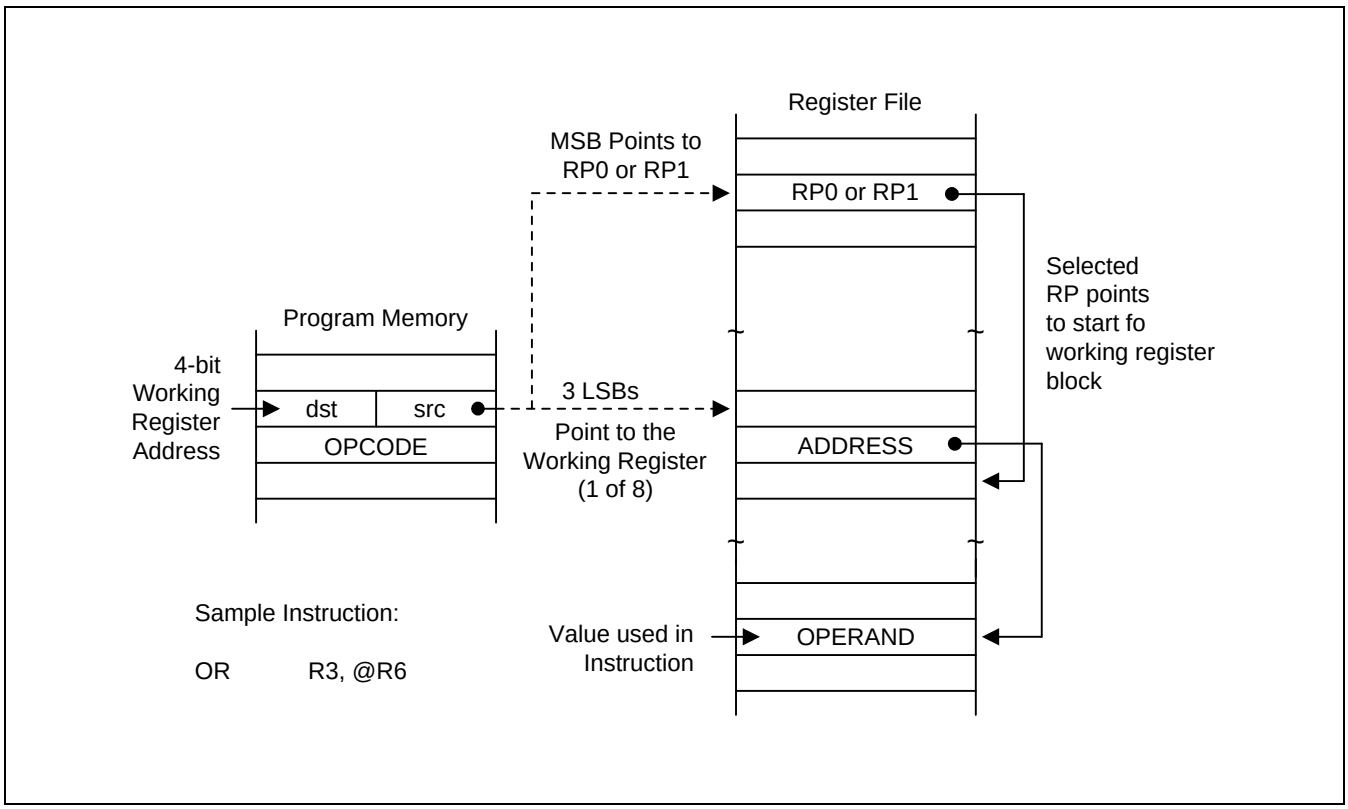


图 3-5 寄存器卷中的间接寄存器寻址

3.1.2.3 间接寄存器寻址模式 (续)

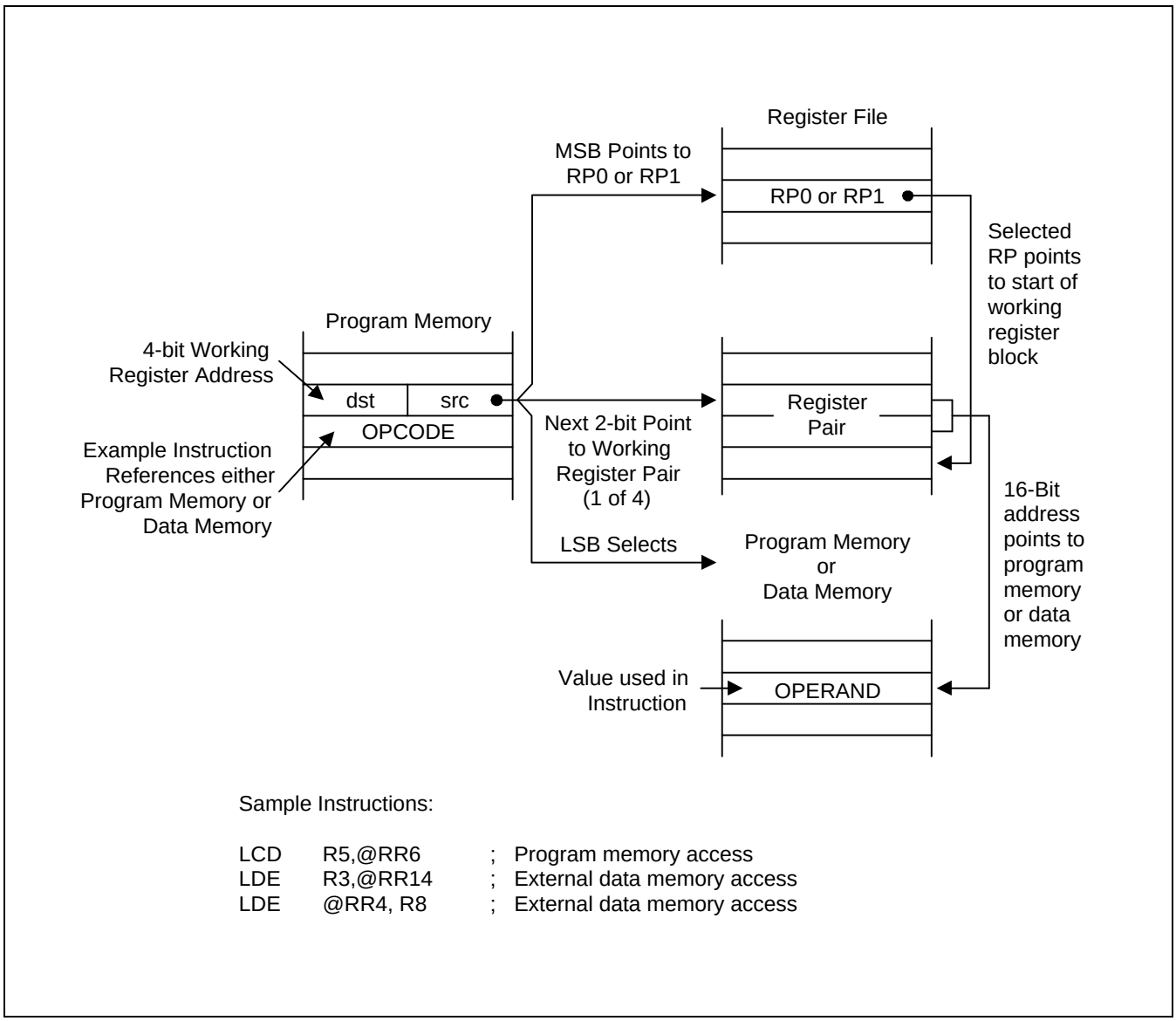


图 3-6 工作寄存器间接访问程序存储器或数据存储器

3.1.3 偏址寻址模式 (X)

在指令执行时，偏址寻址模式(X)是在基地址的基础上加上一个偏移地址量，计算出有效的操作数地址(*图 3-7*)。偏址寻址可以用来访问内部寄存器卷或外部数据存储器空间。但要注意，不能通过它对 Set 1 中地址段 C0H-FFH 进行寻址。

在短指令寻址模式下，8 位的偏移量被认为是介于 -128 到 +127 之间的一个有符号整数，
这只用于外部存储器访问(*图 3-8*)。

对寄存器卷寻址时，指令提供的 8 位基地址与工作寄存器中的 8 位偏移地址相加得到操作数地址。
对外部存储器访问时，基地址存放在指令指示的 16 位工作寄存器中，指令中给出的 8 位或 16 位偏移地址加到基地址上，得到操作数地址(*图 3-9*)。

支持对寄存器卷进行偏址寻址的指令只有 Load(LD)。
LDC 和 LDE 支持内部程序存储器和外部数据存储器(如果存在)的偏址寻址模式。

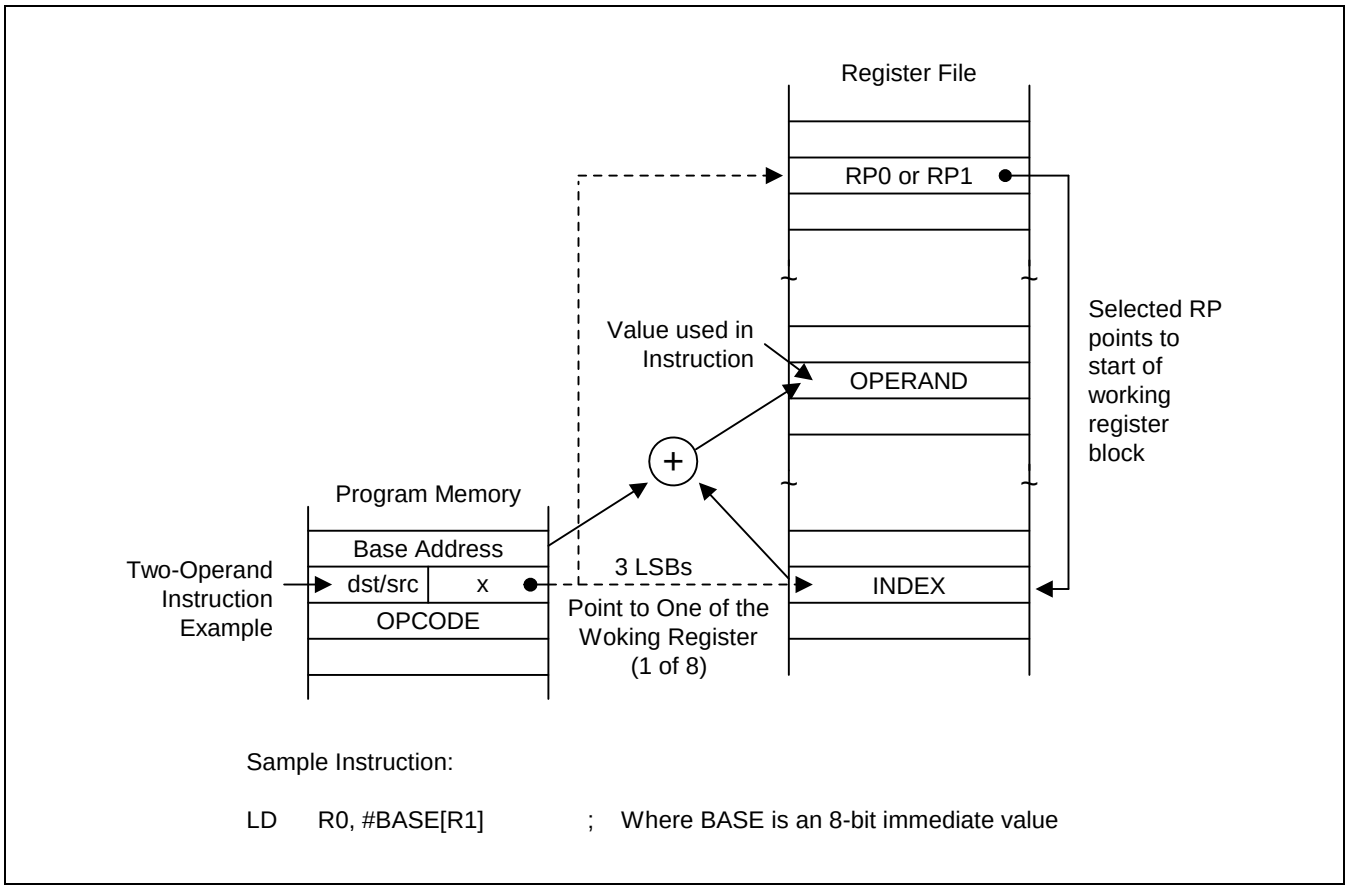


图 3-7 寄存器空间的偏址寻址模式

3.1.3.1 偏址寻址模式 (续)

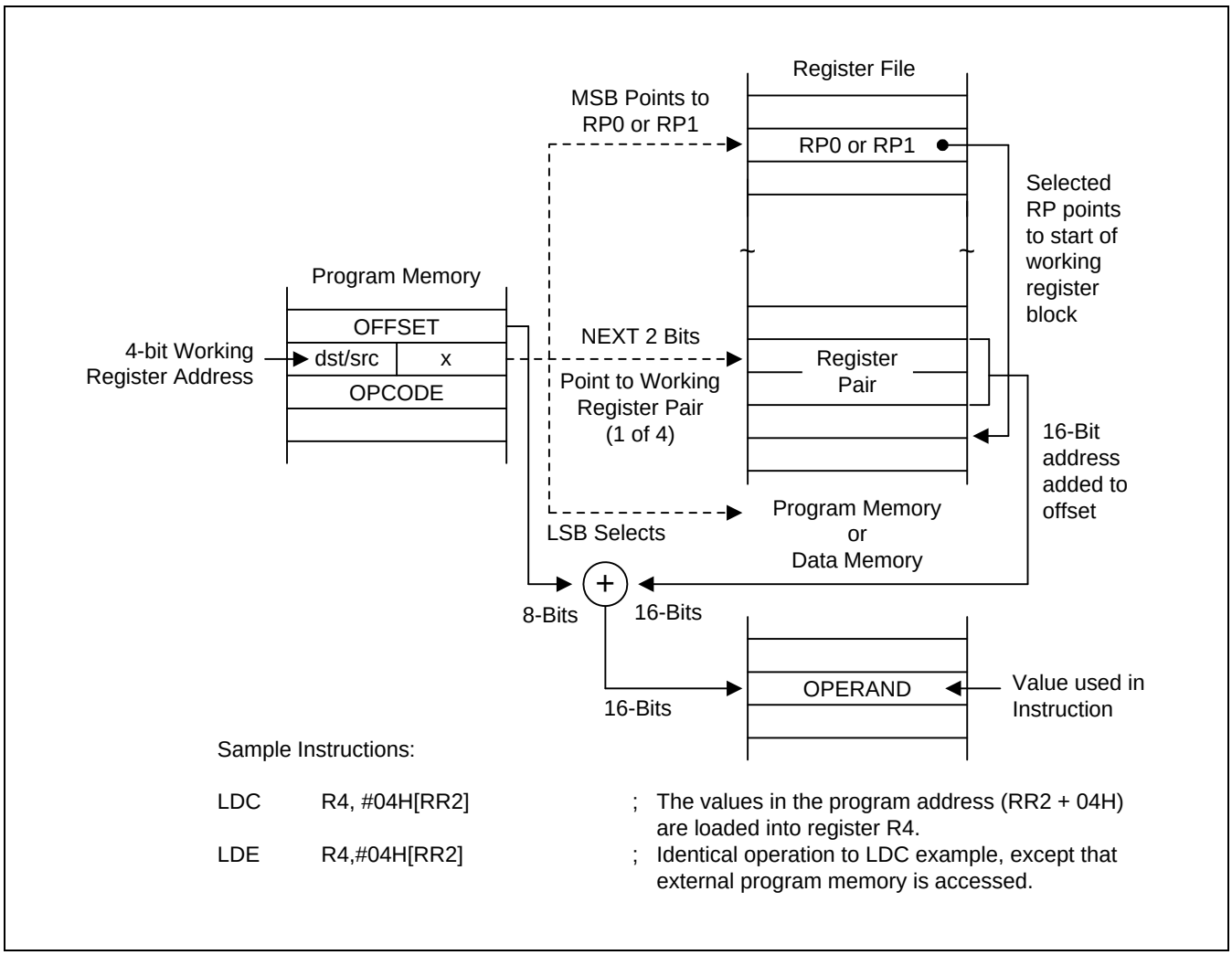
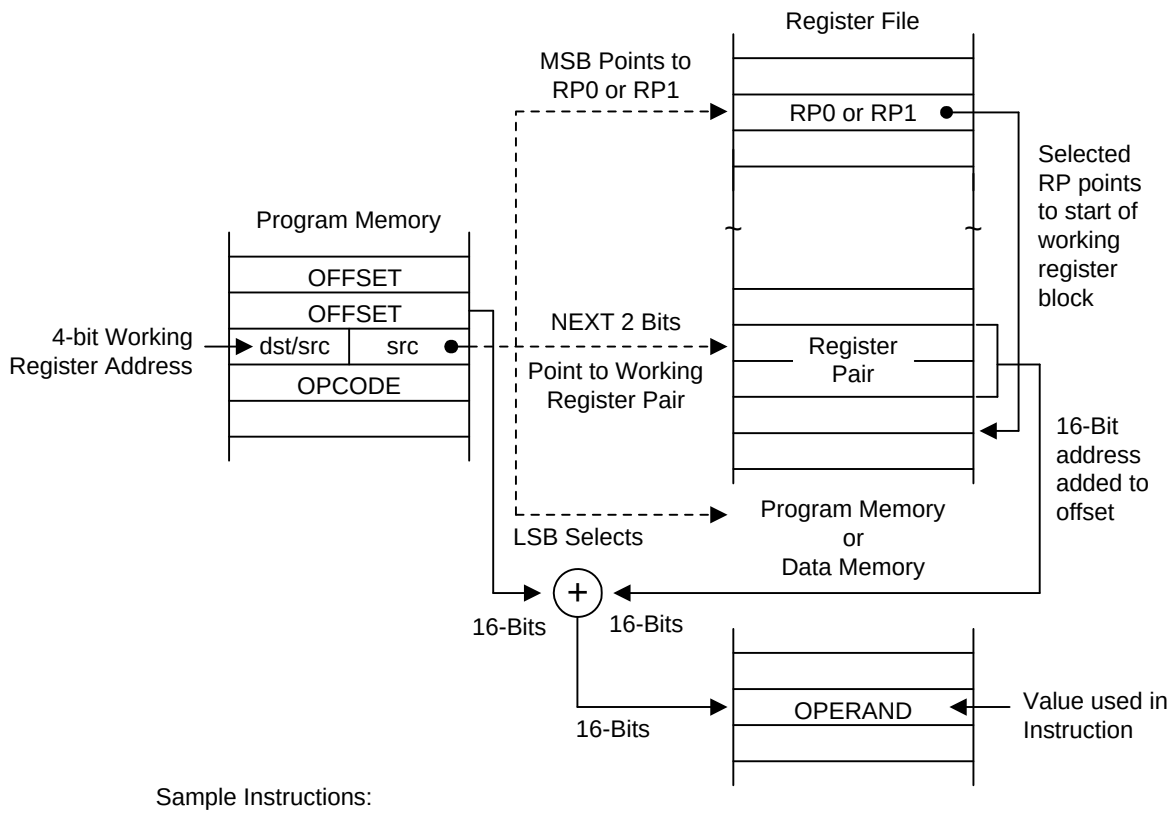


图 3-8 偏址寻址模式中短格式访问程序或数据存储空间

3.1.3.2 偏址寻址模式 (续)



Sample Instructions:

```
LDC     R4, #1000H[RR2]     ; The values in the program address (RR2 + 1000H)
                            ; are loaded into register R4.
LDE     R4,#1000H[RR2]      ; Identical operation to LDC example, except that
                            ; external program memory is accessed.
```

图 3-9 偏址寻址模式中长格式访问程序或数据存储空间

3.1.4 直接寻址模式 (DA)

在直接寻址模式中，指令提供操作数的 16 位存储器地址。执行 Jump(JP) 和 Call(CALL) 指令时，就是采用这种寻址模式指定 16 位的目标地址并将其装入PC。

LDC 和 LDE 指令即运用直接寻址模式为数据传送操作提供源地址或目标地址，LDC 访问程序存储空间，LDE 访问外部数据存储空间。

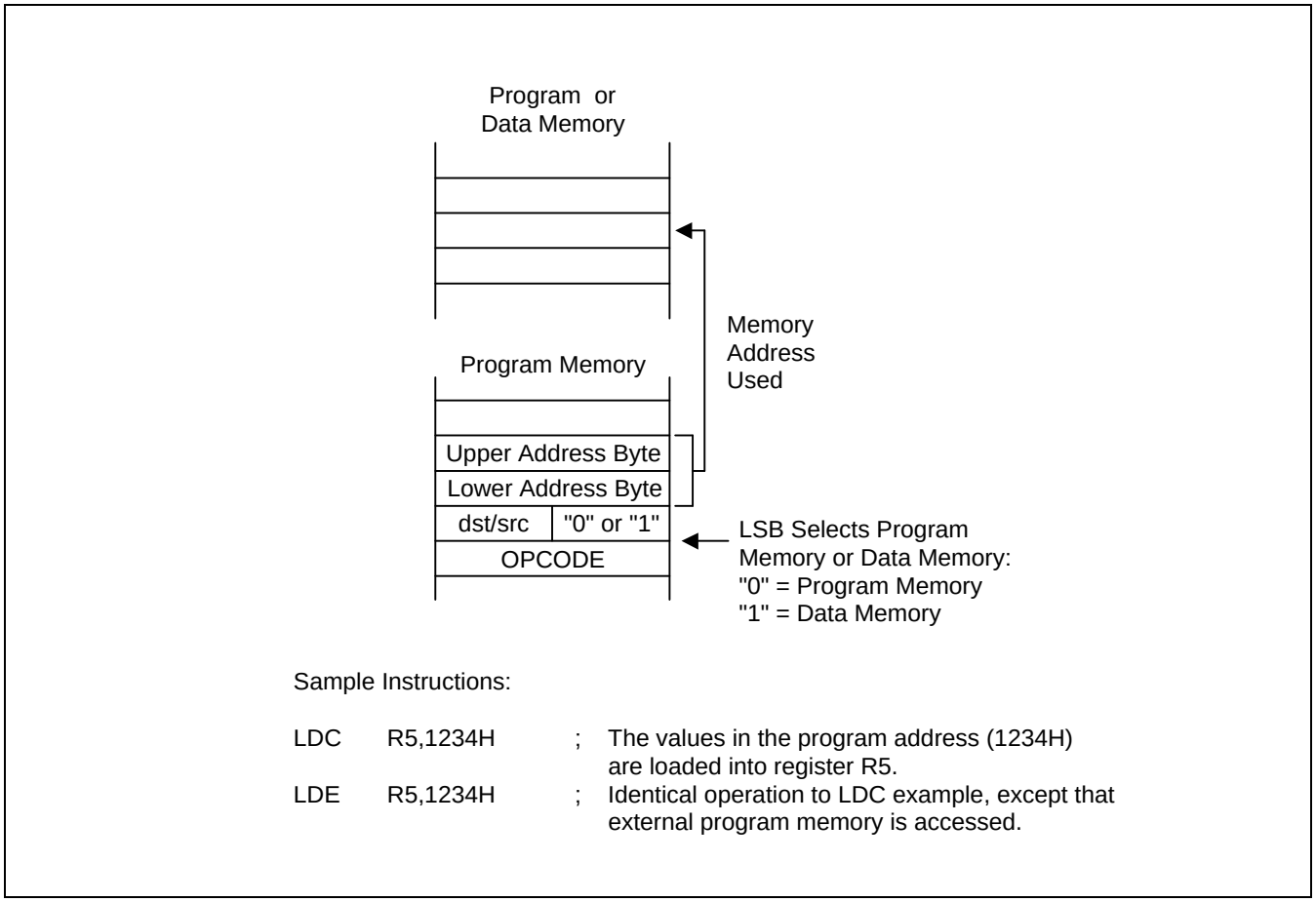
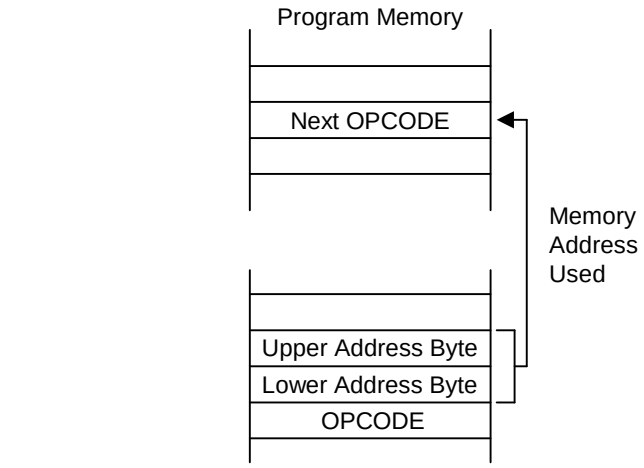


图 3-10 Load 指令的直接寻址

3.1.4.1 直接寻址模式 (续)



Sample Instructions:

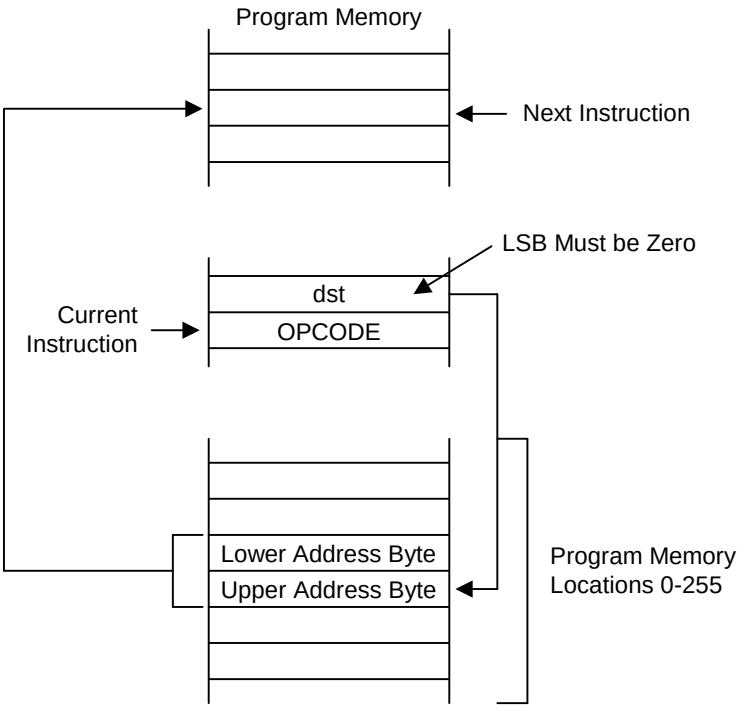
```
JP      C,JOB1   ;  Where JOB1 is a 16-bit immediate address
CALL    DISPLAY  ;  Where DISPLAY is a 16-bit immediate address
```

图 3-11 Call, Jump 的直接寻址

3.1.5 间接寻址模式 (IA)

在间接寻址模式中，须指定程序存储空间中低 256 字节中的某个地址，其中放有要执行的下一条指令地址。只有CALL 指令支持间接寻址模式。

由于间接寻址模式规定操作数只能存放在程序存储空间的低 256 字节中，所以指令中只有一个 8 位地址；目标地址的高 8 位全部为 0。



Sample Instruction:

```
CALL    #40H        ; The 16-bit value in program memory addresses 40H
                    and 41H is the subroutine start address.
```

图 3-12 间接寻址模式

3.1.6 相对地址寻址模式 (RA)

在相对寻址模式中，指令的调转范围只能在有符号数 –128 到 +127 之间。偏移量加上当前 PC 值，即为下一条要执行指令的地址。在加偏移量之前，PC 中的内容是紧跟着当前指令的后一条指令地址。

程序控制指令用相对寻址模式来实现条件跳转。支持相对寻址模式的指令有：BTJRF, BTJRT, DJNZ, CPIJE, CPIJNE 和 JR。

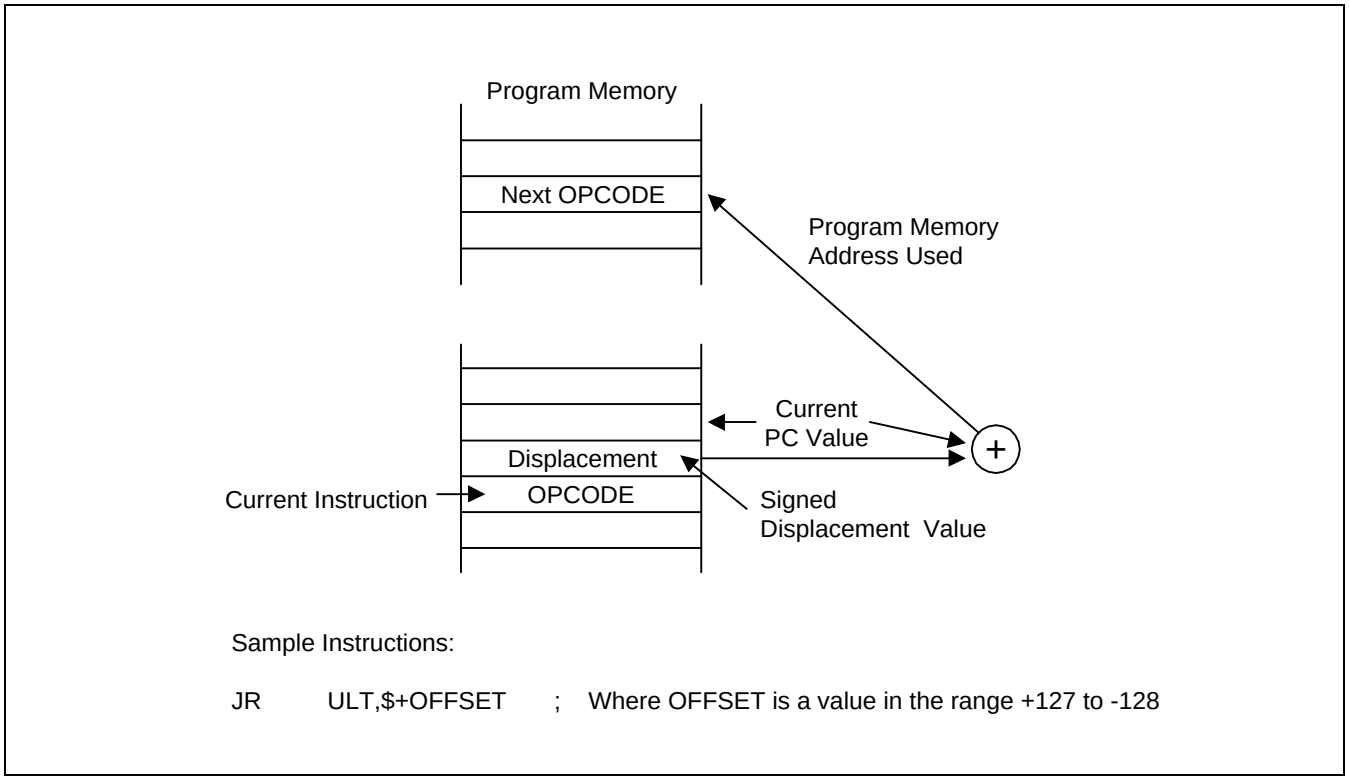


图 3-13 相对寻址

3.1.7 立即数寻址模式 (IM)

立即数寻址模式中，操作数本身就包含在指令当中。根据指令的不同，操作数的长度可以是一个字节或一个字。立即数寻址模式常用来将常量赋值给寄存器。

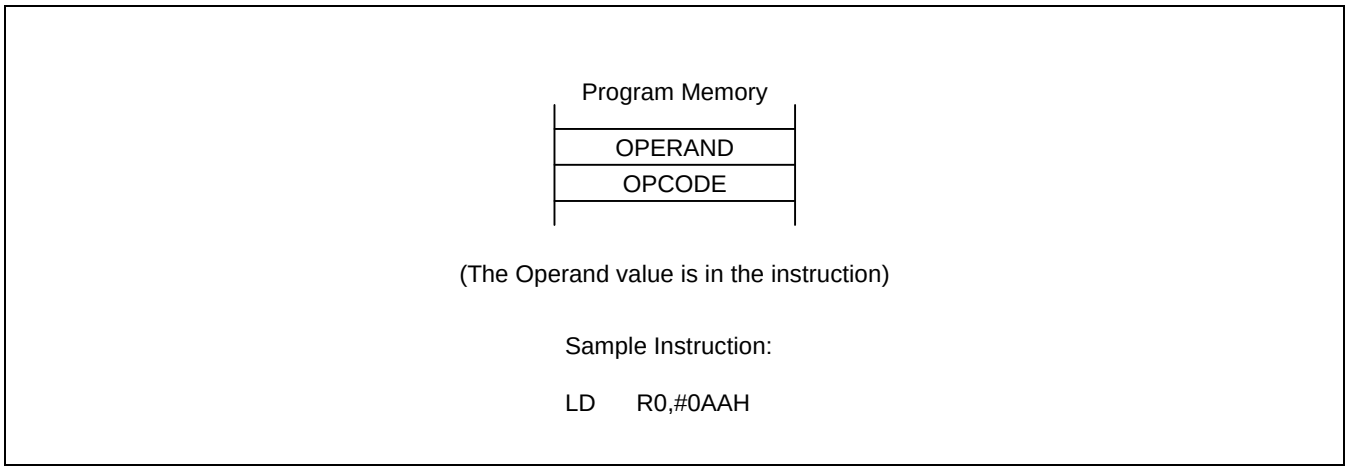


图 3-14 立即数寻址模式

4 控制寄存器

4.1 概述

本章中，S3F84UA/F84U8 控制寄存器的细节描述以简单易读的形式表达。
用户在编写应用程序时，可将本章节作为一个快速参考手册。*图 4-1* 说明了标准寄存器描述格式的重要特点。

控制寄存器描述按照寄存器代表符号的字母顺序排列。更多有关控制寄存器的信息在本手册第二部分硬件资源描述中阐述。

数据和计数器寄存器未在本参考章节中详细描述。更多有关外围设备的寄存器信息在本手册第二部分的相应外围设备描述中阐述。

表 4-1 列举了 S3F84UA/F84U8 所有映射寄存器的位置和读/写特性。 每一个映射寄存器的硬件复位值在第八章“复位和省电模式”中描述。

表 4-1 Set 1 寄存器

寄存器名字	助记标号	地址	Hex	R/W
D0H – D2H 地址空间未映射				
Basic Timer 控制寄存器	BTCON	211	D3H	R/W
系统时钟控制寄存器	CLKCON	212	D4H	R/W
系统标志寄存器	FLAGS	213	D5H	R/W
寄存器指针 0	RP0	214	D6H	R/W
寄存器指针 1	RP1	215	D7H	R/W
堆栈指针 (高字节)	SPH	216	D8H	R/W
堆栈指针 (低字节)	SPL	217	D9H	R/W
指令指针 (高字节)	IPH	218	DAH	R/W
指令指针 (低字节)	IPL	219	DBH	R/W
中断请求寄存器	IRQ	220	DCH	R
中断屏蔽寄存器	IMR	221	DDH	R/W
系统模式寄存器	SYM	222	DEH	R/W
寄存器页指针	PP	223	DFH	R/W

表 4-2 Set 1, Bank 0 寄存器

寄存器名字	助记标号	Decimal	Hex	R/W
A/D 转换数据寄存器 (高字节)	ADDATAH	208	D0H	R
A/D 转换数据寄存器 (低字节)	ADDATAL	209	D1H	R
A/D 转换控制寄存器	ADCON	210	D2H	R/W
Timer A 计数器	TACNT	224	E0H	R
Timer A 数据寄存器	TADATA	225	E1H	R/W
Timer A 控制寄存器	TACON	226	E2H	R/W
Timer B 控制寄存器	TBCON	227	E3H	R/W
Timer B 数据寄存器 (高字节)	TBDATAH	228	E4H	R/W
Timer B 数据寄存器 (低字节)	TBDATAL	229	E5H	R/W
钟表定时器控制寄存器	WTCON	230	E6H	R/W
SIO 控制寄存器	SIOCON	231	E7H	R/W
SIO 数据寄存器	SIODATA	232	E8H	R/W
SIO 预分频寄存器	SIOPS	233	E9H	R/W
Timer C 计数器	TCCNT	234	EAH	R
Timer C 数据寄存器	TCDATA	235	EBH	R/W
Timer C 控制寄存器	TCCON	236	ECH	R/W
STOP 控制寄存器	STPCON	237	EDH	R/W
UART 0 控制寄存器 (高字节)	UART0CONH	238	EEH	R/W
UART 0 控制寄存器 (低字节)	UART0CONL	239	EFH	R/W
UART 0 数据寄存器	UDATA0	240	F0H	R/W
UART 0 波特率数据寄存器	BRDATA0	241	F1H	R/W
UART 1 控制寄存器 (高字节)	UART1CONH	242	F2H	R/W
UART 1 控制寄存器 (低字节)	UART1CONL	243	F3H	R/W
UART 1 数据寄存器	UDATA1	244	F4H	R/W
UART 1 波特率数据寄存器	BRDATA1	245	F5H	R/W
闪存扇区地址寄存器 (高字节)	FMSECH	246	F6H	R/W
闪存扇区地址寄存器 (低字节)	FMSECL	247	F7H	R/W
闪存用户可编程使能寄存器	FMUSR	248	F8H	R/W
闪存控制寄存器	FMCON	249	F9H	R/W
时钟控制寄存器	OSCCON	250	FAH	R/W
中断标志位寄存器	INTPND	251	FBH	R/W
FCH 地址空间未映射				
Basic Timer 计数器	BTCNT	253	FDH	R
FEH 地址空间未映射				
中断优先级寄存器	IPR	255	FFH	R/W

表 4-3 Set 1, Bank 1 寄存器

寄存器名字	助记标号	Decimal	Hex	R/W
P0 口控制寄存器 (高字节)	P0CONH	208	D0H	R/W
P0 口控制寄存器 (低字节)	P0CONL	209	D1H	R/W
P0 口上拉电阻使能控制寄存器	P0PUR	210	D2H	R/W
P2 口控制寄存器 (高字节)	P2CONH	224	E0H	R/W
P2 口控制寄存器 (低字节)	P2CONL	225	E1H	R/W
P1 口控制寄存器	P1CON	226	E2H	R/W
P3 口 N 沟道开漏模式寄存器	PNE3	227	E3H	R/W
P3 口控制寄存器 (高字节)	P3CONH	228	E4H	R/W
P3 口控制寄存器 (低字节)	P3CONL	229	E5H	R/W
P3 口中断控制寄存器 (高字节)	P3INTH	230	E6H	R/W
P3 口中断控制寄存器 (低字节)	P3INTL	231	E7H	R/W
P3 口中断标志位寄存器	P3PND	232	E8H	R/W
P3 口上拉电阻使能控制寄存器	P3PUR	233	E9H	R/W
P4 口中断控制寄存器 (高字节)	P4CONH	234	EAH	R/W
P4 口中断控制寄存器 (低字节)	P4CONL	235	EBH	R/W
P4 口上拉电阻使能控制寄存器	P4PUR	236	ECH	R/W
P4 口 N 沟道开漏模式寄存器	PNE4	237	EDH	R/W
Pattern Generation 模块控制寄存器	PGCON	238	EEH	R/W
Pattern Generation 数据寄存器	PGDATA	239	EFH	R/W
P0 口数据寄存器	P0	240	F0H	R/W
P1 口数据寄存器	P1	241	F1H	R/W
P2 口数据寄存器	P2	242	F2H	R/W
P3 口数据寄存器	P3	243	F3H	R/W
P4 口数据寄存器	P4	244	F4H	R/W
LCD 控制寄存器	LCON	245	F5H	R/W
Timer D0 计数器 (高字节)	TD0CNTH	246	F6H	R
Timer D0 计数器 (低字节)	TD0CNTL	247	F7H	R
Timer D0 数据寄存器 (高字节)	TD0DATAH	248	F8H	R/W
Timer D0 数据寄存器 (低字节)	TD0DATAL	249	F9H	R/W
Timer D0 控制寄存器	TD0CON	250	FAH	R/W
Timer D1 控制寄存器	TD1CON	251	FBH	R/W
Timer D1 计数器 (高字节)	TD1CNTH	252	FCH	R
Timer D1 计数器 (低字节)	TD1CNTL	253	FDH	R
Timer D1 数据寄存器 (高字节)	TD1DATAH	254	FEH	R/W
Timer D1 数据寄存器 (低字节)	TD1DATAL	255	FFH	R/W

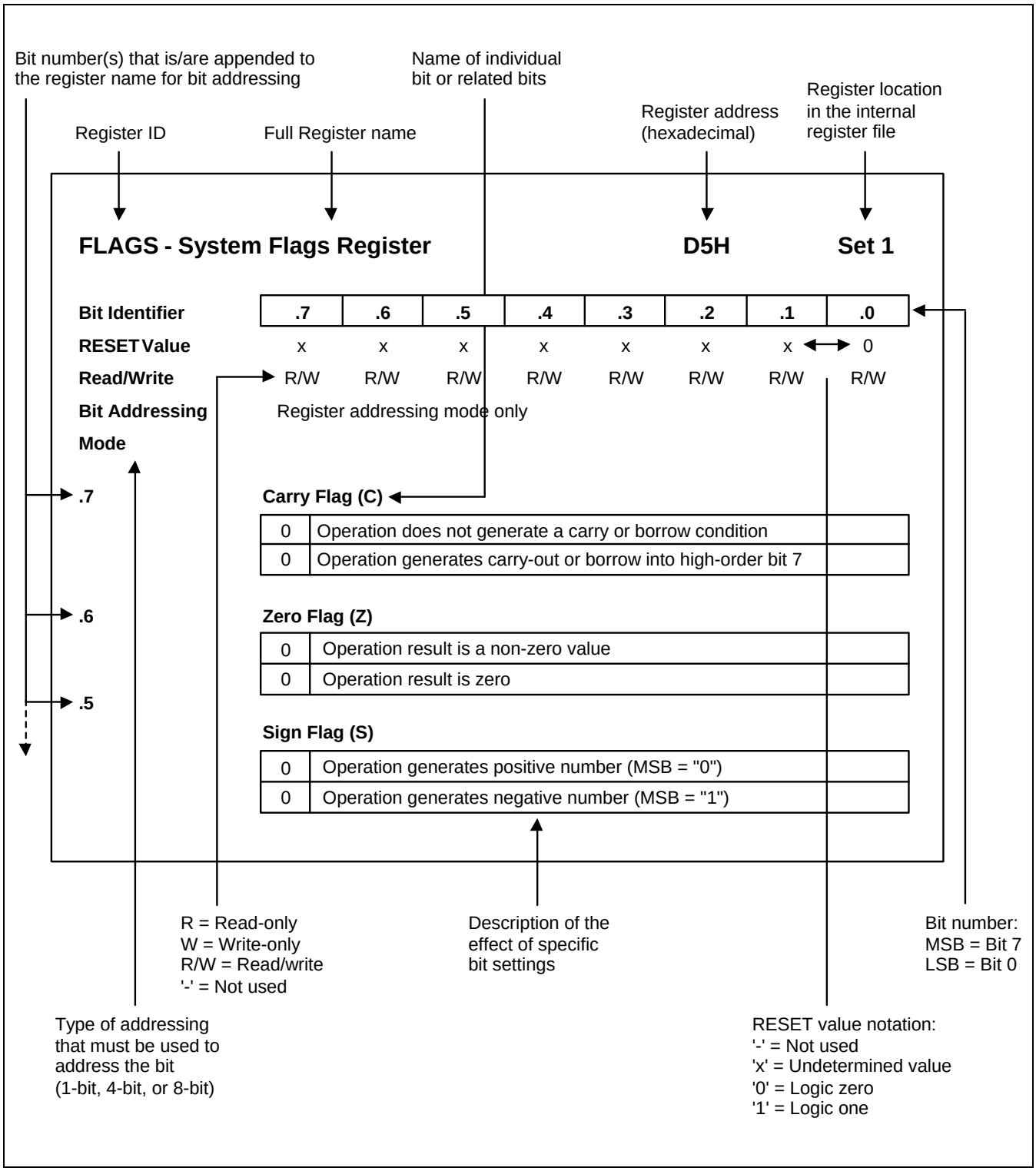

Bit number(s) that is/are appended to the register name for bit addressing
Name of individual bit or related bits
Register location in the internal register file
Register address (hexadecimal)
Register ID
Full Register name
FLAGS - System Flags Register
D5H
Set 1
Bit Identifier .7 .6 .5 .4 .3 .2 .1 .0
RESET Value x x x x x x x 0
Read/Write R/W R/W R/W R/W R/W R/W R/W R/W
Bit Addressing Mode
Register addressing mode only
.7
.6
.5
Carry Flag (C)
0 Operation does not generate a carry or borrow condition
0 Operation generates carry-out or borrow into high-order bit 7
Zero Flag (Z)
0 Operation result is a non-zero value
0 Operation result is zero
Sign Flag (S)
0 Operation generates positive number (MSB = "0")
0 Operation generates negative number (MSB = "1")
R = Read-only
W = Write-only
R/W = Read/write
'-' = Not used
Description of the effect of specific bit settings
Bit number:
MSB = Bit 7
LSB = Bit 0
Type of addressing that must be used to address the bit (1-bit, 4-bit, or 8-bit)
RESET value notation:
'-' = Not used
'x' = Undetermined value
'0' = Logic zero
'1' = Logic one

图 4-1 寄存器描述格式

4.1.1 ADCON—A/D 转换控制寄存器: D2H SET 1, BANK 0

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	–	0	0	0	0	0	0	0
读/写	–	R/W	R/W	R/W	R	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7

S3F84UA/F84U8 不使用

.6–.4 **A/D 输入管脚选择位**

0	0	0	AD0
0	0	1	AD1
0	1	0	AD2
0	1	1	AD3
1	0	0	AD4
1	0	1	AD5
1	1	0	AD6
1	1	1	AD7

.3 **转换结束位 (只读)**

0	转换未完成
1	转换结束

.2–.1 **时钟源选择位**

0	0	fxx/16
0	1	fxx/8
1	0	fxx/4
1	1	fxx/1

.0 **启动或使能位**

0	禁止转换
1	启动转换

4.1.2 BTCON—BASIC TIMER 控制寄存器: D3H SET 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.4 **看门狗时钟功能禁止位 (用于系统复位)**

1	0	1	0	禁止看门狗时钟功能
其他值				使能看门狗时钟功能

.3–.2 **Basic Timer 输入时钟选择位**(3)

0	0	fxx/4096
0	1	fxx/1024
1	0	fxx/128
1	1	fxx/16

.1 **Basic Timer 计数器清零位**(1)

0	没有作用
1	清除 Basic Timer 的计数器值

.0 **Basic Timer 和 Timer/Counter 时钟分频器清除位**(2)

0	没有作用
1	清除两个时钟分频器

注释:

1. 当写“1”到 BTCON.1 位时，Basic Timer 计数器的值被清零。紧跟此写操作，BTCON.1 位的值自动清为“0”。
2. 当写“1”到 BTCON.0 位时，对应的分频器被清为“00H”。紧跟此写操作，BTCON.0 位的值自动清为“0”。
3. fxx 作为系统时钟 (主时钟或副时钟)。

4.1.3 CLKCON—系统时钟控制寄存器: D4H SET 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	–	–	0	0	–	–	–
读/写	R/W	–	–	R/W	R/W	–	–	–
寻址方式	仅寄存器寻址模式							

.7 **时钟 IRQ 唤醒功能位**

0	省电模式下，使能 IRQ 唤醒主时钟
1	省电模式下，禁止 IRQ 唤醒主时钟

.6–.5

S3F84UA/F84U8 不使用

.4–.3 **CPU 时钟 (系统时钟) 选择位**(注释)

0	0	fxx/16
0	1	fxx/8
1	0	fxx/2
1	1	fxx/1

.2–.0

S3F84UA/F84U8 不使用

注释: 复位后，选择最慢时钟(16 分频)作为系统时钟。可通过向 CLKCON.3 和 CLKCON.4 位写合适的值选择更快的时钟速率。

4.1.4 FLAGS—系统标志寄存器: D5H SET 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	x	x	x	x	x	x	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
寻址方式	仅寄存器寻址模式							

.7 Carry Flag (C)

0	操作没有产生进位或借位
1	操作产生进位或向第 7 位的借位

.6 Zero Flag (Z)

0	操作结果不是“0”
1	操作结果是“0”

.5 Sign Flag (S)

0	操作产生正数 (MSB = “0”)
1	操作产生负数 (MSB = “1”)

.4 Overflow Flag (V)

0	操作结果在 –128 ~ +127 之间
1	操作结果不在 –128 ~ +127 之间，即溢出

.3 Decimal Adjust Flag (D)

0	加操作完成
1	减操作完成

.2 Half-Carry Flag (H)

0	加法操作时第 3 位未产生进位或减法操作时第 3 位未产生借位
1	加法操作时第 3 位产生进位或减法操作时第 3 位产生借位

.1 Fast Interrupt Status Flag (FIS)

0	中断返回(IRET)正在进行(读此位时)
1	快速中断服务程序正在进行(读此位时)

.0 Bank Address Selection Flag (BA)

0	选择 Bank 0
1	选择 Bank 1

4.1.5 FMCON—闪存控制寄存器: F9H SET 1, BANK 0

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	–	–	0
读/写	R/W	R/W	R/W	R/W	R	–	–	R/W
寻址方式	仅寄存器寻址模式							

.7–.4 **闪存模式选择位**

0	1	0	1	(字节) 编程模式
1	0	1	0	扇区擦除模式
0	1	1	0	Hard Lock 模式
其他值				不可用

.3 **扇区擦除状态位 (只读)**

0	扇区擦除成功
1	扇区擦除失败

.2–.1

S3F84UA/F84U8 不使用

.0 **闪存操作启动位**

0	操作停止
1	操作开始

注释: 对应的操作完成后，FMCON.0 位将立即自动清零。

4.1.6 FMSECH—闪存扇区地址寄存器 (高字节): F6H SET 1, BANK 0

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.0 **闪存扇区地址位 (高字节)**

闪存扇区选择高 8 位 (第 15 ~ 8 位)。

注释: 高字节闪存扇区地址指针指向 16 位指针地址的高 8 位。

4.1.7 FMSECL—闪存扇区地址寄存器 (低字节): F7H SET 1, BANK 0

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
寻址方式	仅寄存器寻址模式							

.7 **闪存扇区地址位 (低字节)**

闪存扇区选择第 7 位

.6–.0

S3F84UA/F84U8 不使用

注释: 低字节闪存扇区地址指针指向 16 位指针地址的低 8 位。

4.1.8 FMUSR—闪存用户可编程使能寄存器: F8H SET 1, BANK 0

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.0 **闪存用户可编程使能位**

1	0	1	0	0	1	0	1	使能用户编程模式
其他值								禁止用户编程模式

4.1.9 IMR—中断屏蔽寄存器: DDH SET 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	x	x	x	x	x	x	x	x
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7 **中断级 7 (IRQ7) 使能位; 外部中断 P3.4–P3.7**

0	禁止 (屏蔽)
1	使能 (未屏蔽)

.6 **中断级 6 (IRQ6) 使能位; 外部中断 P3.0–P3.3**

0	禁止 (屏蔽)
1	使能 (未屏蔽)

.5 **中断级 5 (IRQ5) 使能位; UART0/1 发送, UART0/1 接收**

0	禁止 (屏蔽)
1	使能 (未屏蔽)

.4 **中断级 4 (IRQ4) 使能位; 钟表定时器, SIO**

0	禁止 (屏蔽)
1	使能 (未屏蔽)

.3 **中断级 3 (IRQ3) 使能位; Timer D0/1 匹配/捕获或溢出**

0	禁止 (屏蔽)
1	使能 (未屏蔽)

.2 **中断级 2 (IRQ2) 使能位; Timer C 匹配/溢出**

0	禁止 (屏蔽)
1	使能 (未屏蔽)

.1 **中断级 1 (IRQ1) 使能位; Timer B 匹配**

0	禁止 (屏蔽)
1	使能 (未屏蔽)

.0 **中断级 0 (IRQ0) 使能位; Timer A 匹配/捕获或溢出**

0	禁止 (屏蔽)
1	使能 (未屏蔽)

注释: 当某个中断级被屏蔽，任何发起的中断请求都被不能被 CPU 识别。

4.1.10 INTPND—中断标志位寄存器: FBH SET 1, BANK 0

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	–	–	0	0	0	0	0	0
读/写	–	–	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址							

.7–.6

S3F84UA/F84U8 不使用

.5 **Timer D1 匹配/捕获中断标志位**

0	没有中断 (读此位时)，中断标志清除 (写 0 时清除中断标志)
1	中断标志位 (读此位时，如果是1则有中断)

.4 **Timer D1 溢出中断标志位**

0	没有中断 (读此位时)，中断标志清除 (写 0 时清除中断标志)
1	中断标志位 (读此位时，如果是1则有中断)

.3 **Timer D0 匹配/捕获中断标志位**

0	没有中断 (读此位时)，中断标志清除 (写 0 时清除中断标志)
1	中断标志位 (读此位时，如果是1则有中断)

.2 **Timer D0 溢出中断标志位**

0	没有中断 (读此位时)，中断标志清除 (写 0 时清除中断标志)
1	中断标志位 (读此位时，如果是1则有中断)

.1 **Timer A 匹配/捕获中断标志位**

0	没有中断 (读此位时)，中断标志清除 (写 0 时清除中断标志)
1	中断标志位 (读此位时，如果是1则有中断)

.0 **Timer A 溢出中断标志位**

0	没有中断 (读此位时)，中断标志清除 (写 0 时清除中断标志)
1	中断标志位 (读此位时，如果是1则有中断)

4.1.11 IPH—指令指针 (高字节): DAH SET 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	x	x	x	x	x	x	x	x
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.0 **指令指针地址 (高字节)**

高字节指令指针指向 16 位指令指针地址的高 8 位 (IP15–IP8)。
IP 地址的低字节在 IPL 寄存器 (地址：DBH) 中。

4.1.12 IPL—指令指针 (低字节): DBH SET 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	x	x	x	x	x	x	x	x
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.0 **指令指针地址 (低字节)**

低字节指令指针指向 16 位指令指针地址的低 8 位 (IP7–IP0)。
IP 地址的高字节在 IPH 寄存器 (地址：DAH) 中。

4.1.13 IPR—中断优先级寄存器: FFH SET 1, BANK 0

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	x	x	x	x	x	x	x	x
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7, .4, and .1 **中断 A, B 和 C 组优先级控制位**

0	0	0	组优先级未定义
0	0	1	B > C > A
0	1	0	A > B > C
0	1	1	B > A > C
1	0	0	C > A > B
1	0	1	C > B > A
1	1	0	A > C > B
1	1	1	组优先级未定义

.6 **中断 C 子分组优先级控制位**

0	IRQ6 > IRQ7
1	IRQ7 > IRQ6

.5 **中断 C 组优先级控制位**

0	IRQ5 > (IRQ6, IRQ7)
1	(IRQ6, IRQ7) > IRQ5

.3 **中断 B 子分组优先级控制位**

0	IRQ3 > IRQ4
1	IRQ4 > IRQ3

.2 **中断 B 组优先级控制位**

0	IRQ2 > (IRQ3, IRQ4)
1	(IRQ3, IRQ4) > IRQ2

.0 **中断 A 组优先级控制位**

0	IRQ0 > IRQ1
1	IRQ1 > IRQ0

注释: 中断 A 组 - IRQ0, IRQ1
中断 B 组 - IRQ2, IRQ3, IRQ4
中断 C 组 -IRQ5, IRQ6, IRQ7

4.1.14 IRQ—中断请求寄存器: DCH SET 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R	R	R	R	R	R	R	R
寻址模式	仅寄存器寻址模式							

.7 **中断级 7 (IRQ7) 请求标志位; 外部中断 P3.4–P3.7**

0	没有中断
1	标志位置起

.6 **中断级 6 (IRQ6) 请求标志位; 外部中断 P3.0–P3.3**

0	没有中断
1	标志位置起

.5 **中断级 5 (IRQ5) 请求标志位; UART0/1 发送, UART0/1 接收**

0	没有中断
1	标志位置起

.4 **中断级 4 (IRQ4) 请求标志位; 钟表定时器, SIO**

0	没有中断
1	标志位置起

.3 **中断级 3 (IRQ3) 请求标志位; Timer D0/1 匹配/捕获或溢出**

0	没有中断
1	标志位置起

.2 **中断级 2 (IRQ2) 请求标志位; Timer C 匹配/溢出**

0	没有中断
1	标志位置起

.1 **中断级 1 (IRQ1) 请求标志位; Timer B 匹配**

0	没有中断
1	标志位置起

.0 **中断级 0 (IRQ0) 请求标志位; Timer A 匹配/捕获或溢出**

0	没有中断
1	标志位置起

4.1.15 LCON—LCD 控制寄存器: F5H SET 1, BANK 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	–	–	0
读/写	R/W	R/W	R/W	R/W	R/W	–	–	R/W
寻址方式	仅寄存器寻址模式							

.7–.6 **LCD 时钟选择位**

0	0	fw/28 (128Hz)
0	1	fw/27 (256Hz)
1	0	fw/26 (512Hz)
1	1	fw/25 (1024Hz)

.5–.3 **LCD 占空比和偏压比选择位**

0	0	0	1/8 占空比, 1/4 偏压比
0	0	1	1/4 占空比, 1/3 偏压比
0	1	0	1/3 占空比, 1/3 偏压比
0	1	1	1/3 占空比, 1/2 偏压比
1	x	x	1/2 占空比, 1/2 偏压比

.2–.1

S3F84UA/F84U8 不使用

.0 **LCD 显示控制位**

0	关闭显示
1	打开显示

注释: 无论 LCON 寄存器数据值如何改写，LCD 控制器/驱动器的时钟和占空比由硬件自动初始化。因此，LCON 寄存器不要经常改写。

4.1.16 OSCCON—时钟控制寄存器: FAH SET 1, BANK 0

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	–	–	–	–	0	0	–	0
读/写	–	–	–	–	R/W	R/W	–	R/W
寻址方式	仅寄存器寻址模式							

.7–.4

S3F84UA/F84U8 不使用

.3 **主振荡器控制寄存器**

0	主振荡器运行
1	主振荡器停止

.2 **副振荡器控制寄存器**

0	副振荡器运行
1	副振荡器停止

.1

S3F84UA/F84U8 不使用

.0 **系统时钟选择位**

0	选择主振荡器作为系统时钟
1	选择副振荡器作为系统时钟

4.1.17 P0CONH—P0 口控制寄存器 (高字节): D0H SET 1, BANK 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.6 **P0.7/PG7/AD7**

0	0	输入模式
0	1	复用功能 (PG7)
1	0	复用功能 (AD7)
1	1	推挽式输出

.5–.4 **P0.6/PG6/AD6**

0	0	输入模式
0	1	复用功能 (PG6)
1	0	复用功能 (AD6)
1	1	推挽式输出

.3–.2 **P0.5/PG5/AD5**

0	0	输入模式
0	1	复用功能 (PG5)
1	0	复用功能 (AD5)
1	1	推挽式输出

.1–.0 **P0.4/PG4/AD4**

0	0	输入模式
0	1	复用功能 (PG4)
1	0	复用功能 (AD4)
1	1	推挽式输出

4.1.18 P0CONL—P0 口控制寄存器 (低字节): D1H SET 1, BANK 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.6 **P0.3/PG3/AD3**

0	0	输入模式
0	1	复用功能 (PG3)
1	0	复用功能 (AD3)
1	1	推挽式输出

.5–.4 **P0.2/PG2/AD2**

0	0	输入模式
0	1	复用功能 (PG2)
1	0	复用功能 (AD2)
1	1	推挽式输出

.3–.2 **P0.1/PG1/AD1**

0	0	输入模式
0	1	复用功能 (PG1)
1	0	复用功能 (AD1)
1	1	推挽式输出

.1–.0 **P0.0/PG0/AD0**

0	0	输入模式
0	1	复用功能 (PG0)
1	0	复用功能 (AD0)
1	1	推挽式输出

4.1.19 P0PUR—P0 口上拉电阻使能控制寄存器: D2H SET 1, BANK 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7 P0.7 上拉电阻使能位

0	上拉电阻禁止
1	上拉电阻使能

.6 P0.6 上拉电阻使能位

0	上拉电阻禁止
1	上拉电阻使能

.5 P0.5 上拉电阻使能位

0	上拉电阻禁止
1	上拉电阻使能

.4 P0.4 上拉电阻使能位

0	上拉电阻禁止
1	上拉电阻使能

.3 P0.3 上拉电阻使能位

0	上拉电阻禁止
1	上拉电阻使能

.2 P0.2 上拉电阻使能位

0	上拉电阻禁止
1	上拉电阻使能

.1 P0.1 上拉电阻使能位

0	上拉电阻禁止
1	上拉电阻使能

.0 P0.0 上拉电阻使能位

0	上拉电阻禁止
1	上拉电阻使能

注释: P0 口只有在对应管脚选择作为推挽式输出或复用功能时，上拉电阻才自动禁止。

4.1.20 P1CON—P1 口控制寄存器: E2H SET 1, BANK 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	–	–	–	–	0	0	0	0
读/写	–	–	–	–	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.4

S3F84UA/F84U8 不使用

.3–.2 **P1.1/ XT_{IN}**

0	0	输入模式
0	1	输入模式，带上拉电阻
1	0	复用功能 (XT_{IN})
1	1	推挽式输出

.1–.0 **P1.0/ XT_{OUT}**

0	0	输入模式
0	1	输入模式，带上拉电阻
1	0	复用功能 (XT_{OUT})
1	1	推挽式输出

4.1.21 P2CONH—P2 口控制寄存器 (高字节): E0H SET 1, BANK 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.6 **P2.7/SEG5/COM7**

0	0	输入模式
0	1	输入模式，带上拉电阻
1	0	复用功能 (LCD 信号)
1	1	推挽式输出

.5-.4 **P2.6/SEG4/COM6**

0	0	输入模式
0	1	输入模式，带上拉电阻
1	0	复用功能 (LCD 信号)
1	1	推挽式输出

.3–.2 **P2.5/SEG3/COM5**

0	0	输入模式
0	1	输入模式，带上拉电阻
1	0	复用功能 (LCD 信号)
1	1	推挽式输出

.1–.0 **P2.4/SEG2/COM4**

0	0	输入模式
0	1	输入模式，带上拉电阻
1	0	复用功能 (LCD 信号)
1	1	推挽式输出

4.1.22 P2CONL—P2 口控制寄存器 (低字节): E1H SET 1, BANK 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.6 P2.3/SEG1/COM3

0	0	输入模式
0	1	输入模式，带上拉电阻
1	0	复用功能 (LCD 信号)
1	1	推挽式输出

.5–.4 P2.2/SEG0/COM2

0	0	输入模式
0	1	输入模式，带上拉电阻
1	0	复用功能 (LCD 信号)
1	1	推挽式输出

.3–.2 P2.1/COM1

0	0	输入模式
0	1	输入模式，带上拉电阻
1	0	复用功能 (LCD 信号)
1	1	推挽式输出

.1–.0 P2.0/COM0

0	0	输入模式
0	1	输入模式，带上拉电阻
1	0	复用功能 (LCD 信号)
1	1	推挽式输出

4.1.23 P3CONH—P3 口控制寄存器 (高字节): E4H SET 1, BANK 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.6 **P3.7/INT7/TD0OUT/TD0PWM/TD0CAP/SEG13**

0	0	施密特输入模式 (TD0CAP)
0	1	复用功能 (TD0OUT/TD0PWM)
1	0	复用功能 (LCD 信号)
1	1	输出模式

.5–.4 **P3.6/INT6/TD0CLK/SEG12**

0	0	施密特输入模式 (TD0CLK)
0	1	不可用
1	0	复用功能 (LCD 信号)
1	1	输出模式

.3–.2 **P3.5/INT5/TD1OUT/TD1PWM/TD1CAP/SEG11**

0	0	施密特输入模式 (TD1CAP)
0	1	复用功能 (TD1OUT/TD1PWM)
1	0	复用功能 (LCD 信号)
1	1	输出模式

.1–.0 **P3.4/INT4/TD1CLK/SGE10**

0	0	施密特输入模式 (TD1CLK)
0	1	不可用
1	0	复用功能 (LCD 信号)
1	1	输出模式

4.1.24 P3CONL—P3 口控制寄存器 (低字节): E5H SET 1, BANK 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.6 **P3.3/INT3/SCK/SEG9**

0	0	施密特输入模式 (SCK)
0	1	复用功能 (SCK 输出)
1	0	复用功能 (LCD 信号)
1	1	输出模式

.5–.4 **P3.2/INT2/SI/SEG8**

0	0	施密特输入模式 (SI)
0	1	不可用
1	0	复用功能 (LCD 信号)
1	1	输出模式

.3–.2 **P3.1/INT1/SO/SEG7**

0	0	施密特输入模式
0	1	复用功能 (SO)
1	0	复用功能 (LCD 信号)
1	1	输出模式

.1–.0 **P3.0/INT0/BUZ/SEG6**

0	0	输入模式
0	1	复用功能 (BUZ)
1	0	复用功能 (LCD 信号)
1	1	输出模式

4.1.25 P3INTH—P3 口中断控制寄存器 (高字节): E6H SET 1, BANK 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.6 **P3.7/ 外部中断 (INT7) 使能位**

0	0	禁止中断
0	1	使能下降沿中断
1	0	使能上升沿中断
1	1	同时使能上升沿和下降沿中断

.5–.4 **P3.6/ 外部中断 (INT6) 使能位**

0	0	禁止中断
0	1	使能下降沿中断
1	0	使能上升沿中断
1	1	同时使能上升沿和下降沿中断

.3–.2 **P3.5/ 外部中断 (INT5) 使能位**

0	0	禁止中断
0	1	使能下降沿中断
1	0	使能上升沿中断
1	1	同时使能上升沿和下降沿中断

.1–.0 **P3.4/ 外部中断 (INT4) 使能位**

0	0	禁止中断
0	1	使能下降沿中断
1	0	使能上升沿中断
1	1	同时使能上升沿和下降沿中断

4.1.26 P3INTL—P3 口中断控制寄存器 (低字节): E7H SET 1, BANK 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.6 **P3.3/ 外部中断 (INT3) 使能位**

0	0	禁止中断
0	1	使能下降沿中断
1	0	使能上升沿中断
1	1	同时使能上升沿和下降沿中断

.5–.4 **P3.2/ 外部中断 (INT2) 使能位**

0	0	禁止中断
0	1	使能下降沿中断
1	0	使能上升沿中断
1	1	同时使能上升沿和下降沿中断

.3–.2 **P3.1/ 外部中断 (INT1) 使能位**

0	0	禁止中断
0	1	使能下降沿中断
1	0	使能上升沿中断
1	1	同时使能上升沿和下降沿中断

.1–.0 **P3.0/ 外部中断 (INT0) 使能位**

0	0	禁止中断
0	1	使能下降沿中断
1	0	使能上升沿中断
1	1	同时使能上升沿和下降沿中断

4.1.27 P3PND—P3 口中断标志寄存器: E8H SET 1, BANK 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7 P3.7/ 外部中断 (INT7) 标志位

0	中断标志清除 (写 0 时清除中断标志)
1	有 P3.7/INT7 中断请求 (读此位时，如果是 1 则有中断)

.6 P3.6/ 外部中断 (INT6) 标志位

0	中断标志清除 (写 0 时清除中断标志)
1	有 P3.6/INT6 中断请求 (读此位时，如果是 1 则有中断)

.5 P3.5/ 外部中断 (INT5) 标志位

0	中断标志清除 (写 0 时清除中断标志)
1	有 P3.5/INT5 中断请求 (读此位时，如果是 1 则有中断)

.4 P3.4/ 外部中断 (INT4) 标志位

0	中断标志清除 (写 0 时清除中断标志)
1	有 P3.4/INT4 中断请求 (读此位时，如果是 1 则有中断)

.3 P3.3/ 外部中断 (INT3) 标志位

0	中断标志清除 (写 0 时清除中断标志)
1	有 P3.3/INT3 中断请求 (读此位时，如果是 1 则有中断)

.2 P3.2/ 外部中断 (INT2) 标志位

0	中断标志清除 (写 0 时清除中断标志)
1	有 P3.2/INT2 中断请求 (读此位时，如果是 1 则有中断)

.1 P3.1/ 外部中断 (INT1) 标志位

0	中断标志清除 (写 0 时清除中断标志)
1	有 P3.1/INT1 中断请求 (读此位时，如果是 1 则有中断)

.0 P3.0/ 外部中断 (INT0) 标志位

0	中断标志清除 (写 0 时清除中断标志)
1	有 P3.0/INT0 中断请求 (读此位时，如果是 1 则有中断)

4.1.28 P3PUR—P3 口上拉电阻使能控制寄存器: E9H SET 1, BANK 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7 P3.7 上拉电阻使能位

0	上拉电阻禁止
1	上拉电阻使能

.6 P3.6 上拉电阻使能位

0	上拉电阻禁止
1	上拉电阻使能

.5 P3.5 上拉电阻使能位

0	上拉电阻禁止
1	上拉电阻使能

.4 P3.4 上拉电阻使能位

0	上拉电阻禁止
1	上拉电阻使能

.3 P3.3 上拉电阻使能位

0	上拉电阻禁止
1	上拉电阻使能

.2 P3.2 上拉电阻使能位

0	上拉电阻禁止
1	上拉电阻使能

.1 P3.1 上拉电阻使能位

0	上拉电阻禁止
1	上拉电阻使能

.0 P3.0 上拉电阻使能位

0	上拉电阻禁止
1	上拉电阻使能

注释: P3 口只有在对应管脚选择作为推挽式输出或复用功能时，上拉电阻才自动禁止。

4.1.29 PNE3—P3 口 N 沟道开漏模式寄存器: E3H SET 1, BANK 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7 P3.7 输出模式选择位

0	推挽式输出
1	开漏输出

.6 P3.6 输出模式选择位

0	推挽式输出
1	开漏输出

.5 P3.5 输出模式选择位

0	推挽式输出
1	开漏输出

.4 P3.4 输出模式选择位

0	推挽式输出
1	开漏输出

.3 P3.3 输出模式选择位

0	推挽式输出
1	开漏输出

.2 P3.2 输出模式选择位

0	推挽式输出
1	开漏输出

.1 P3.1 输出模式选择位

0	推挽式输出
1	开漏输出

.0 P3.0 输出模式选择位

0	推挽式输出
1	开漏输出

4.1.30 P4CONH—P4 口控制寄存器 (高字节): EAH SET 1, BANK 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.6 **P4.7/RxD0/SEG21**

0	0	输入模式 (RxD0)
0	1	复用功能 (RxD0 输出)
1	0	复用功能 (LCD 信号)
1	1	输出模式

.5–.4 **P4.6/TxD0/SEG20**

0	0	输入模式
0	1	复用功能 (TxD0)
1	0	复用功能 (LCD 信号)
1	1	输出模式

.3–.2 **P4.5/RxD1/SEG19**

0	0	输入模式 (RxD1)
0	1	复用功能 (RxD1 输出)
1	0	复用功能 (LCD 信号)
1	1	输出模式

.1–.0 **P4.4/TxD1/SEG18**

0	0	输入模式
0	1	复用功能 (TxD1)
1	0	复用功能 (LCD 信号)
1	1	输出模式

4.1.31 P4CONL—P4 控制寄存器 (低字节): EBH SET 1, BANK 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.6 **P4.3/TAOUT/TAPWM/TACAP/SEG17**

0	0	输入模式 (TACAP)
0	1	复用功能 (TAOUT/TAPWM)
1	0	复用功能 (LCD 信号)
1	1	输出模式

.5–.4 **P4.2/TACLK/SEG16**

0	0	输入模式 (TACLK)
0	1	不可用
1	0	复用功能 (LCD 信号)
1	1	输出模式

.3–.2 **P4.1/TBPWM/SEG15**

0	0	输入模式
0	1	复用功能 (TBPWM)
1	0	复用功能 (LCD 信号)
1	1	输出模式

.1–.0 **P4.0/TCOUT/TCPWM/SEG14**

0	0	输入模式
0	1	复用功能 (TCOUT/TCPWM)
1	0	复用功能 (LCD 信号)
1	1	输出模式

4.1.32 P4PUR—P4 上拉电阻使能控制寄存器: ECH SET 1, BANK 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7 P4.7 上拉电阻使能位

0	上拉电阻禁止
1	上拉电阻使能

.6 P4.6 上拉电阻使能位

0	上拉电阻禁止
1	上拉电阻使能

.5 P4.5 上拉电阻使能位

0	上拉电阻禁止
1	上拉电阻使能

.4 P4.4 上拉电阻使能位

0	上拉电阻禁止
1	上拉电阻使能

.3 P4.3 上拉电阻使能位

0	上拉电阻禁止
1	上拉电阻使能

.2 P4.2 上拉电阻使能位

0	上拉电阻禁止
1	上拉电阻使能

.1 P4.1 上拉电阻使能位

0	上拉电阻禁止
1	上拉电阻使能

.0 P4.0 上拉电阻使能位

0	上拉电阻禁止
1	上拉电阻使能

注释: P4 口只有在对应管脚选择作为推挽式输出或复用功能时，上拉电阻才自动禁止。

4.1.33 PNE4—P4 口 N 沟道开漏模式寄存器: EDH SET 1, BANK 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7 **P4.7 输出模式选择位**

0	推挽式输出
1	开漏输出

.6 **P4.6 输出模式选择位**

0	推挽式输出
1	开漏输出

.5 **P4.5 输出模式选择位**

0	推挽式输出
1	开漏输出

.4 **P4.4 输出模式选择位**

0	推挽式输出
1	开漏输出

.3 **P4.3 输出模式选择位**

0	推挽式输出
1	开漏输出

.2 **P4.2 输出模式选择位**

0	推挽式输出
1	开漏输出

.1 **P4.1 输出模式选择位**

0	推挽式输出
1	开漏输出

.0 **P4.0 输出模式选择位**

0	推挽式输出
1	开漏输出

4.1.34 PGCON—PATTERN GENERATION 模块控制寄存器: EEH SET 1, BANK 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	–	–	–	–	0	0	0	0
读/写	–	–	–	–	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.4

S3F84UA/F84U8 不使用

.3 **软件触发开始位**

0	无效
1	软件触发开始 (自动清零)

.2 **PG 操作禁止/使能选择位**

0	PG 操作禁止
1	PG 操作使能

.1–.0 **检测电压选择位**

0	0	Timer A 匹配信号触发
0	1	Timer B 溢出信号触发
1	0	Timer D0 匹配信号触发
1	1	软件触发

4.1.35 PP—寄存器页指针: DFH SET 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.4 **目的寄存器页选择位**

0	0	0	0	目的寄存器：第 0 页
0	0	0	1	目的寄存器：第 1 页 (S3F84U8 不使用)
1	0	0	0	目的寄存器：第 8 页
其他值				S3F84UA/F84U8 不使用

.3 – .0 **源寄存器页选择位**

0	0	0	0	源寄存器：第 0 页
0	0	0	1	源寄存器：第 1 页 (S3F84U8 不使用)
1	0	0	0	源寄存器：第 8 页
其他值				S3F84UA/F84U8 不使用

注释:

1. 对于 S3F84UA MCU 来说，内部寄存器卷设置为 3 页(第 0-1 页和第 8 页), 其中第 0-1 页可作为通用寄存器卷。
2. 对于 S3F84U8 MCU 来说，内部寄存器卷设置为 2 页(第 0 页和第 8 页), 其中第 0 页可作为通用寄存器卷。
3. S3F84UA/F84U8 的第 8 页作为 LCD 数据寄存器 (30H-45H)。

4.1.36 RP0—寄存器指针 0: D6H SET 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	1	1	0	0	0	–	–	–
读/写	R/W	R/W	R/W	R/W	R/W	–	–	–
寻址方式	仅寄存器寻址模式							

.7–.3 **寄存器指针 0 地址值**

寄存器指针 0 可以单独指向寄存器卷中的某个 256 字节工作寄存器区域。通过使用寄存器指针 RP0 和 RP1 同时选择 2 个 8 字节寄存器组作为有效的工作寄存器空间。复位后，RP0 指向寄存器 Set 1 的 C0H 地址，选择从 C0H 到 C7H 的 8 位工作寄存器。

.2–.0

S3F84UA/F84U8 不使用

4.1.37 RP1—寄存器指针 1: D7H SET 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	1	1	0	0	1	–	–	–
读/写	R/W	R/W	R/W	R/W	R/W	–	–	–
寻址方式	仅寄存器寻址模式							

.7 – .3 **寄存器指针 1 地址值**

寄存器指针 1 可以单独指向寄存器卷中的某个 256 字节工作寄存器区域。通过使用寄存器指针 RP0 和 RP1 同时选择 2 个 8 字节寄存器组作为有效的工作寄存器空间。复位后，RP1 指向寄存器 Set 1 的 C8H 地址，选择从 C8H 到 CFH 的 8 位工作寄存器。

.2 – .0

S3F84UA/F84U8 不使用

4.1.38 SIOCON—SIO 控制寄存器: E7H SET 1, BANK 0

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7 SIO 移位时钟选择位

0	内部时钟 (P.S 时钟)
1	外部时钟 (SCK)

.6 数据方向控制位

0	MSB (最高位) 优先模式
1	LSB (最低位) 优先模式

.5 SIO 模式选择位

0	仅接收模式
1	发送/接收模式

.4 移位时钟边沿选择位

0	下降沿发送，上升沿接收
1	上升沿发送，下降沿接收

.3 SIO 计数器清零和移位启动位

0	无动作
1	清除 3 位计数器并开始移位

.2 SIO 移位操作使能位

0	禁止移位器和时钟计数器
1	使能移位器和时钟计数器

.1 SIO 中断使能位

0	禁止 SIO 中断
1	使能 SIO 中断

.0 SIO 中断标志位

0	无中断 (读此位时)，清除中断标志 (写此位时)
1	有中断 (读此位时，如果是 1 则有中断)

4.1.39 SPH—堆栈指针 (高字节): D8H SET 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	x	x	x	x	x	x	x	x
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.0 **堆栈指针 (高字节)**

高字节堆栈指针的值是 16 位堆栈指针地址 (SP15–SP8) 的高 8 位。低字节堆栈指针的值位于寄存器 SPL (地址：D9H) 中。复位后，堆栈指针 (SP) 的值不确定。

4.1.40 SPL—STACK POINTER (LOW BYTE): D9H SET 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	x	x	x	x	x	x	x	x
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.0 **堆栈指针 (低字节)**

低字节堆栈指针的值是 16 位堆栈指针地址 (SP7–SP0) 的低 8 位。高字节堆栈指针的值位于寄存器 SPH (地址：D8H) 中。复位后，堆栈指针 (SP) 的值不确定。

4.1.41 STPCON—STOP 控制寄存器: EDH SET 1, BANK 0

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.0 **STOP 控制位**

1 0 1 0 0 1 0 1	使能使用 STOP 指令
其他值	禁止使用 STOP 指令

注释: 在执行 STOP 指令前，用户必须将 STPCON 寄存器设置为“10100101B”，否则，STOP 指令不会被执行，同时产生复位。

4.1.42 SYM—系统模式寄存器: DEH SET 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	–	–	–	x	x	x	0	0
读/写	–	–	–	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.5

S3F84UA/F84U8 不使用

.4–.2 **快速中断级选择位**(1)

0	0	0	IRQ0
0	0	1	IRQ1
0	1	0	IRQ2
0	1	1	IRQ3
1	0	0	IRQ4
1	0	1	IRQ5
1	1	0	IRQ6
1	1	1	IRQ7

.1 **快速中断使能位**(2)

0	禁止快速中断处理
1	使能快速中断处理

.0 **全局中断使能位**(3)

0	禁止所有中断处理
1	使能所有中断处理

注释:

1. 用户一次只能选择一个中断级作为快速中断处理。
2. 将 SYM.1 位设置为“1”使能快速中断处理，通过 SYM.2–SYM.4 位选择当前中断级。
3. 复位后，用户必须通过执行 EI 指令使能全局中断处理 (而不是往 SYM.0 位写“1”)。

4.1.43 TACON—TIMER A 控制寄存器: E2H SET 1, BANK 0

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.5 Timer A 输入时钟选择位

0	0	0	fxx/1024
0	0	1	fxx/256
0	1	0	fxx/64
0	1	1	fxx/8
1	0	0	fxx/1
1	0	1	外部时钟 (TACLK) 下降沿
1	1	0	外部时钟 (TACLK) 上升沿
1	1	1	计数器停止

.4–.3 Timer A 工作模式选择位

0	0	Interval (定时) 模式 (TAOUT)
0	1	Capture (捕获) 模式 (上升沿时捕获，计数器运行，允许溢出发生)
1	0	Capture (捕获) 模式 (下降沿时捕获，计数器运行，允许溢出发生)
1	1	PWM 模式 (允许溢出和匹配中断发生)

.2 Timer A 计数器清零位

0	没有作用
1	清除 Timer A 计数器 (写此位时)

.1 Timer A 匹配/捕获中断使能位

0	禁止中断
1	使能中断

.0 Timer A 溢出中断使能位

0	禁止溢出中断
1	使能溢出中断

注释: TACON.2 的值在计数器清零后自动清除为“0”。

4.1.44 TBCON—TIMER B 控制寄存器: E3H SET 1, BANK 0

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.6 **Timer B 输入时钟选择位**

0	0	fxx/1
0	1	fxx/2
1	0	fxx/4
1	1	fxx/8

.5–.4 **Timer B 中断时钟选择位**

0	0	低位数据借位后产生
0	1	高位数据借位后产生
1	0	低或高位数据借位后均产生
1	1	不使用

.3 **Timer B 中断使能位**

0	禁止中断
1	使能中断

.2 **Timer B 开始/停止位**

0	停止 Timer B
1	开始 Timer B

.1 **Timer B 模式选择位**

0	One-shot (单次触发) 模式
1	Repeat (重复) 模式

.0 **Timer B 输出触发器控制位**

0	TBOF 为低 (TBPWM：低位数据时输出低电平，高位数时输出据高电平)
1	TBOF 为高 (TBPWM：低位数据时输出高电平，高位数时输出据低电平)

4.1.45 TCCON—TIMER C0 控制寄存器: ECH SET 1, BANK 0

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7 **Timer C 开始/停止位**

0	停止 Timer C
1	开始 Timer C

.6–.4 **Timer C 3 位预设位**

0	0	0	不分频
0	0	1	2 分频
0	1	0	3 分频
0	1	1	4 分频
1	0	0	5 分频
1	0	1	6 分频
1	1	0	7 分频
1	1	1	8 分频

.3 **Timer C 计数器清零位**

0	没有作用
1	清除 Timer C 计数器 (写此位时)

.2 **Timer C 模式选择位**

0	fxx/1 & PWM 模式
1	fxx/64 & Interval (定时) 模式

.1 **Timer C 中断使能位**

0	禁止中断
1	使能中断

.0 **Timer C 中断标志位**

0	无中断 (读此位时)，清除中断标志位 (写 0 时清除中断标志)
1	有中断 (读此位时，如果是 1 则有中断)

注释: TCCON.3 的值在计数器清零后自动清除为“0”。

4.1.46 TD0CON—TIMER D0 控制寄存器: FAH SET 1, BANK 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.5 **Timer D0 输入时钟选择位**

0	0	0	fxx/1024
0	0	1	fxx/256
0	1	0	fxx/64
0	1	1	fxx/8
1	0	0	fxx/1
1	0	1	外部时钟 (TD0CLK) 下降沿
1	1	0	外部时钟 (TD0CLK) 上升沿
1	1	1	计数器停止

.4–.3 **Timer D0 工作模式选择位**

0	0	Interval (定时) 模式 (TD0OUT)
0	1	Capture (捕获) 模式 (上升沿时捕获，计数器运行，允许溢出发生)
1	0	Capture (捕获) 模式 (下降沿时捕获，计数器运行，允许溢出发生)
1	1	PWM 模式 (允许溢出或匹配中断发生)

.2 **Timer D0 计数器清零位**

0	没有作用
1	清除 Timer D0 计数器 (写此位时)

.1 **Timer D0 匹配/捕获中断使能位**

0	禁止中断
1	使能中断

.0 **Timer D0 溢出中断使能位**

0	禁止溢出中断
1	使能溢出中断

注释: TD0CON.2 的值在计数器清零后自动清除为“0”。

4.1.47 TD1CON—TIMER D1 控制寄存器: FBH SET 1, BANK 1

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.5 **Timer D1 输入时钟选择位**

0	0	0	fxx/1024
0	0	1	fxx/256
0	1	0	fxx/64
0	1	1	fxx/8
1	0	0	fxx/1
1	0	1	外部时钟 (TD1CLK) 下降沿
1	1	0	外部时钟 (TD1CLK) 上升沿
1	1	1	计数器停止

.4–.3 **Timer D1 工作模式选择位**

0	0	Interval (定时) 模式 (TD0OUT)
0	1	Capture (捕获) 模式 (上升沿时捕获，计数器运行，允许溢出发生)
1	0	Capture (捕获) 模式 (下降沿时捕获，计数器运行，允许溢出发生)
1	1	PWM 模式 (允许溢出或匹配中断发生)

.2 **Timer D1 计数器清零位**

0	没有作用
1	清除 Timer D1 计数器 (写此位时)

.1 **Timer D1 匹配/捕获中断使能位**

0	禁止中断
1	使能中断

.0 **Timer D1 溢出中断使能位**

0	禁止溢出中断
1	使能溢出中断

注释: The TD1CON.2 的值在计数器清零后自动清除为“0”。

4.1.48 UART0CONH—UART 0 控制寄存器 (高字节): EEH SET 1, BANK 0

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.6 UART 0 模式选择位

0	0	模式 0：移位寄存器 (fU/(16 × (BRDATA0+1)))
0	1	模式 1：8 位 UART (fU/(16 × (BRDATA0+1)))
1	0	模式 2：9 位 UART (fU/16)
1	1	模式 3：9 位 UART (fU/(16 × (BRDATA0+1)))

.5 多芯片通讯使能位 (仅对模式 2 和模式 3)

0	禁止
1	使能

.4 串行数据接收使能位

0	禁止
1	使能

.3 TB8 (仅当 UART0CONL.7 = 0 时)

UART 0 模式 2 或模式 3 下待发送的第 9 个数据位 (“0”或“1”)

注释：如果 UART0CONL.7 = 1，则此位为“无关位”。

.2 RB8 (仅当 UART0CONL.7 = 0 时)

UART 0 模式 2 或模式 3 下接收到的第 9 个数据位 (“0”或“1”)

注释：如果 UART0CONL.7 = 1，则此位为“无关位”。

.1 UART 0 接收中断使能位

0	禁止接收中断
1	使能接收中断

.0 UART 0 接收中断标志位

0	无中断 (读此位时)，清除中断标志位 (写 0 时清除中断标志)
1	有中断 (读此位时，如果是 1 则有中断)

注释:

1. 在模式 2 或模式 3 下，如果 MCE 位设置为“1”，那么当接收到的第 9 位数据位为“0”时，接收中断不起作用；在模式 1 下，当 MCE = “1”，如果没有接收到有效的停止位，则接收中断也不起作用；在模式 0 下，MCE应当设置为“0”。
2. 对 8 位和 9 位 UART 模式的描述，不包括串行数据接收和发送的起始位和停止位。

4.1.49 UART0CONL—UART 0 控制寄存器 (低字节): EFH SET 1, BANK 0

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7 UART 0 发送校验位自动生成使能位

0	禁止校验位自动生成
1	使能校验位自动生成

.6 UART 0 发送校验位选择位 (仅模式 2 和模式 3)

0	偶校验位
1	奇校验位

注释：如果 UART0CONL.7 = 0，则此位为“无关位”。

.5 UART 0 接收校验位选择位 (仅模式 2 和模式 3)

0	偶校验位检查
1	奇校验位检查

注释：如果 UART0CONL.7 = 0，则此位为“无关位”。

.4 UART 0 接收校验位错误状态位 (仅模式 2 和模式 3)

0	校验无错误
1	校验错误

注释：如果 UART0CONL.7 = 0，则此位为“无关位”。

.3–.2 UART 0 时钟选择位

0	0	fxx/8
0	1	fxx/4
1	0	fxx/2
1	1	fxx/1

.1 UART 0 发送中断使能位

0	禁止发送中断
1	使能发送中断

.0 UART 0 发送中断标志位

0	无中断 (读此位时) 清除中断标志位 (写 0 时清除中断标志)
1	有中断(读此位时，如果是 1 则有中断)

4.1.50 UART1CONH—UART 1 控制寄存器 (高字节): F2H SET 1, BANK 0

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7–.6 UART 1 模式选择位

0	0	模式 0：移位寄存器 (fU/(16 × (BRDATA1+1)))
0	1	模式 1：8 位 UART (fU/(16 × (BRDATA1+1)))
1	0	模式 2：9 位 UART (fU/16)
1	1	模式 3：9 位 UART (fU/(16 × (BRDATA1+1)))

.5 多机通讯使能位 (仅对模式 2 和模式 3)

0	禁止
1	使能

.4 串行数据接收使能位

0	禁止
1	使能

.3 TB8 (仅当 UART1CONL.7 = 0 时)

UART 1 模式 2 或模式 3 下待发送的第 9 个数据位 (“0”或“1”)

注释：如果 UART1CONL.7 = 1，则此位为“无关位”。

.2 RB8 (仅当 UART1CONL.7 = 0 时)

UART 1 模式 2 或模式 3 下接收到的第 9 个数据位 (“0”或“1”)

注释：如果 UART1CONL.7 = 1，则此位为“无关位”。

.1 UART 1 接收中断使能位

0	禁止接收中断
1	使能接收中断

.0 UART 1 接收中断标志位

0	无中断 (读此位时), 清除中断标志位 (写 0 时清除中断标志)
1	有中断 (读此位时, 如果是 1 则有中断)

注释:

1. 在模式 2 或模式 3 下，如果 MCE 位设置为“1”，那么当接收到的第 9 位数据位为“0”时，接收中断不起作用；在模式 1 下，当 MCE = “1”，如果没有接收到有效的停止位，则接收中断也不起作用；在模式 0 下，MCE应当设置为“0”。
2. 对 8 位和 9 位 UART 模式的描述，不包括串行数据接收和发送的起始位和停止位。

4.1.51 UART1CONL—UART 1 控制寄存器 (低字节): F3H SET 1, BANK 0

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7 UART 1 发送校验位自动生成使能位

0	禁止校验位自动生成
1	使能校验位自动生成

.6 UART 1 发送校验位选择位 (仅模式 2 和模式 3)

0	偶校验位
1	奇校验位

注释：如果 UART1CONL.7 = 0，则此位为“无关位”。

.5 UART 1 接收校验位选择位 (仅模式 2 和模式 3)

0	偶校验位检查
1	奇校验位检查

注释：如果 UART1CONL.7 = 0，则此位为“无关位”。

.4 UART 1 接收校验位错误状态位 (仅模式 2 和模式 3)

0	校验无错误
1	校验错误

注释：如果 UART1CONL.7 = 0，则此位为“无关位”。

.3–.2 UART 1 时钟选择位

0	0	fxx/8
0	1	fxx/4
1	0	fxx/2
1	1	fxx/1

.1 UART 1 发送中断使能位

0	禁止发送中断
1	使能发送中断

.0 UART 1 发送中断标志位

0	无中断 (读此位时), 清除中断标志位 (写 0 时清除中断标志)
1	有中断 (读此位时, 如果是 1 则有中断)

4.1.52 WTCON—钟表定时器控制寄存器: E6H SET 1, BANK 0

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址方式	仅寄存器寻址模式							

.7 **钟表定时器时钟选择位**

0	主系统时钟除以 27 (fxx/128)
1	副系统时钟 (fxt)

.6 **钟表定时器中断使能位**

0	禁止钟表定时器中断
1	使能钟表定时器中断

.5–.4 **蜂鸣器信号选择位**

0	0	0.5kHz
0	1	1kHz
1	0	2kHz
1	1	4kHz

.3–.2 **钟表定时器速率选择位**

0	0	设置钟表定时器的中断为 0.5s
0	1	设置钟表定时器的中断为 0.25s
1	0	设置钟表定时器的中断为 0.125s
1	1	设置钟表定时器的中断为 1.995ms

.1 **钟表定时器使能位**

0	禁止钟表定时器，分频器清零
1	使能钟表定时器

.0 **钟表定时器中断标志位**

0	无中断 (读此位时)，清除中断标志位 (写 0 时清除中断标志)
1	有中断 (读此位时，如果是 1 则有中断)

注释: 假定钟表定时器的时钟分频 (fw) 为 32.768kHz。

5 中断结构

5.1 概述

S3C8- 系列的中断结构由三个部分组成：中断级，中断向量和中断源。SAM8 的 CPU 最多可以识别 8 个中断级和 128 个中断向量。当多个中断向量同属于一个中断级时，这几个向量的优先级由向量地址的前后决定. 从芯片设计的角度来看，单个或多个中断源可以共用同一个中断向量地址。

5.1.1 中断级 (LEVELS)

中断级是中断优先级分配和识别的主要操作单元。所有的外围和 I/O 模块都可以发出中断请求。
换句话说，所有外围和 I/O 模块的操作都可以是中断驱动的(interrupt-driven)。一共有 8 个可能的中断级：IRQ0–IRQ7，也被称为中断级 0 ~ 中断级 7。每个中断级直接与中断请求序号(IRQn)关联。
但是每款芯片中，中断级的数量可以相同也可以不同。S3F84UA/F84U8 的中断结构中就有 8 个中断级。

中断级序号 0~7 不直接表示中断级的优先顺序，仅仅是让 CPU 识别中断级的一个区分标号。
不同中断级对应的优先级由中断优先级寄存器 IPR 中的设置决定。通过 IPR 的设置，可以让用户对中断组和子分组结构定义更细化了各个级之间的优先级关系。

5.1.2 中断向量 (VECTORS)

每一个中断级中可以包括一个或多个中断向量，又或者没有任何中断向量。最多支持 128 个中断向量(但 S3C8-系列产品中实际用到的向量个数往往远小于 128)。当多个中断向量同属于一个优先级时，
这几个向量的优先级由硬件决定。S3F84UA/F84U8 里有 22 个中断向量。

5.1.3 中断源 (SOURCES)

中断源可以是任何外围产生的中断，也可以是外部管脚或者计数器的溢出等。多个中断源可以共用一个中断向量地址。S3F84UA/F84U8 的中断结构中有 22 个中断源。

当中断服务程序开始之后，必须清除相应的中断标志位，如果不是硬件自动清零，就必须软件“手动”清零。标志位类型是由中断源的标志位产生/清除机制决定的。

5.1.4 中断类型

之前介绍的 S3C8- 系列中断结构的三个组成部分 — 中断级，中断向量和中断源 — 一起决定了各个产品的中断结构，目的为了充分利用中断逻辑。中断结构一共有三种可能的组合，称为中断 Type 1，2 和 3。
三者之间的区别在于分配给每个中断级的中断向量和中断源的个数不同(*图 5-1*)。

- Type 1: 一个中断级 (IRQn) + 一个中断向量 (V_1) + 一个中断源 (S_1)
- Type 2: 一个中断级 (IRQn) + 一个中断向量 (V_1) + 多个中断源 ($S_1 – S_n$)
- Type 3: 一个中断级 (IRQn) + 多个中断向量 ($V_1 – V_n$) + 多个中断源 ($S_1 – S_n$, $S_{n+1} – S_{n+m}$)

S3F84UA/F84U8 MCU 中用到以上两种类型。

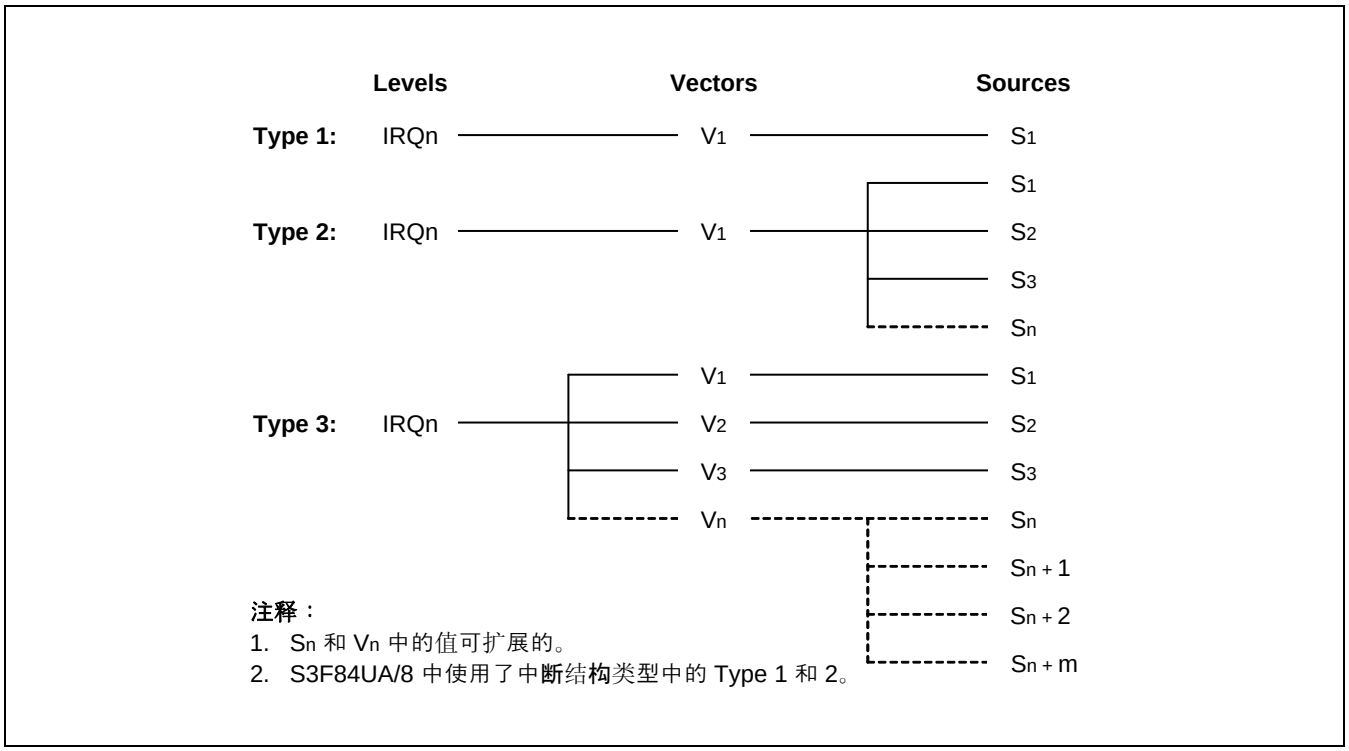


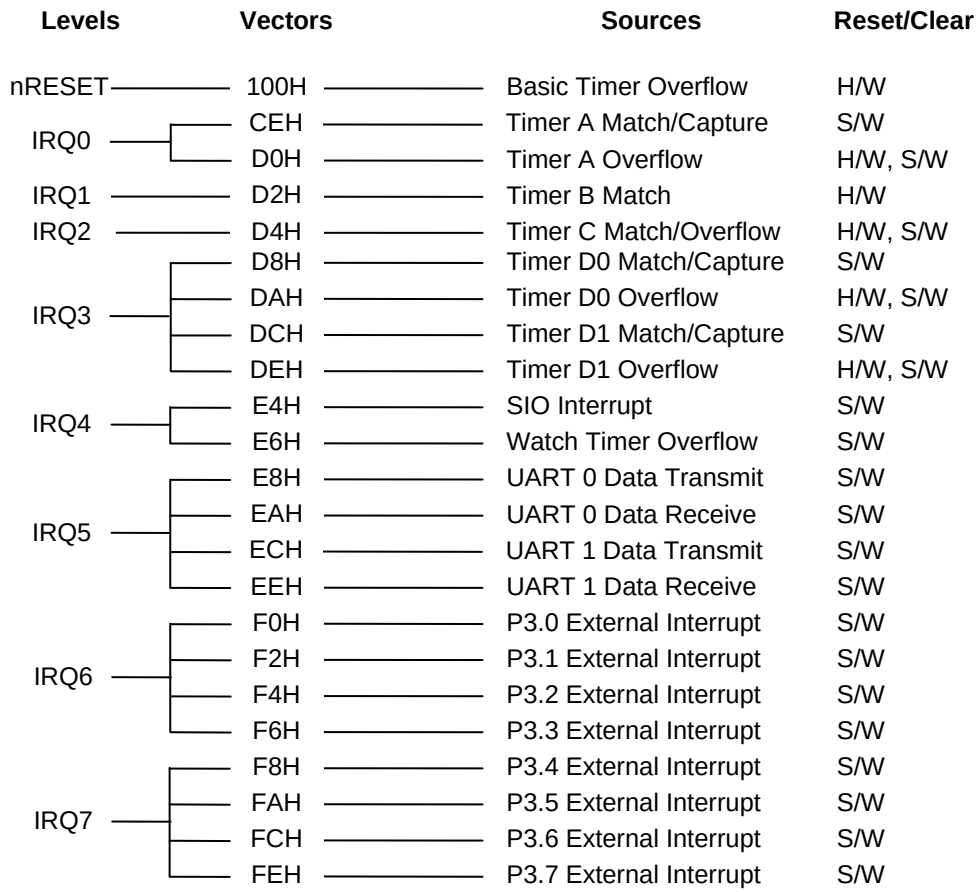
图 5-1 S3C8- 系列的中断类型

5.1.5 S3F84UA/F84U8 中断结构

S3F84UA/F84U8 MCU 支持 22 个中断源，每一个中断源都有唯一的中断向量地址。在这款芯片中，CPU 可以响应 8 个中断级(*图 5-2*)。

多个中断级同时发出中断请求时，中断优先级寄存器(IRP)决定哪一个竞争中的中断级先被 CPU 响应的顺序。如果同一个中断级中的多个中断源同时发出中断请求，通常来说，中断向量地址最小的那个中断源最先被处理(同一个中断级内中断源优先顺序由硬件决定)。

当 CPU 响应了某个中断请求后，中断处理就开始了。此时所有的其他中断都处于禁止状态，程序计数器的值和状态标志位的值被压入栈。从中断向量地址中获取中断服务程序的起始地址 (16 位地址由中断向量地址处的 8 位和向量地址之后的 8 位数据一起构成)，程序跳转之后开始执行中断服务程序。



注释:

1. 在一个给定的中断级中，向量地址越小，中断优先级越高。
   例如, IRQ0 中，CEH 比 D0H 优先级高。同一优先级中各个中断源的优先级顺序由芯片设计者决定。
2. 外部中断由上升沿或下降沿触发，触发方式由相应的控制寄存器决定。

图 5-2 S3F84UA/F84U8 中断结构

5.1.5.1 中断向量地址

S3F84UA/F84U8 中断结构中的所有的中断向量地址都位于内部 48K ROM(0H–BFFFH)或 8K ROM(0H–1FFFH)的向量地址区(*图 5-3*)。

用户可将向量地址区中未使用的地址空间当作普通的程序空间来用。此时需小心不能和已储存的中断向量地址重叠(*表 5-1* 列出了所有的中断向量地址)。

默认的程序复位地址为 ROM 区 0100H。

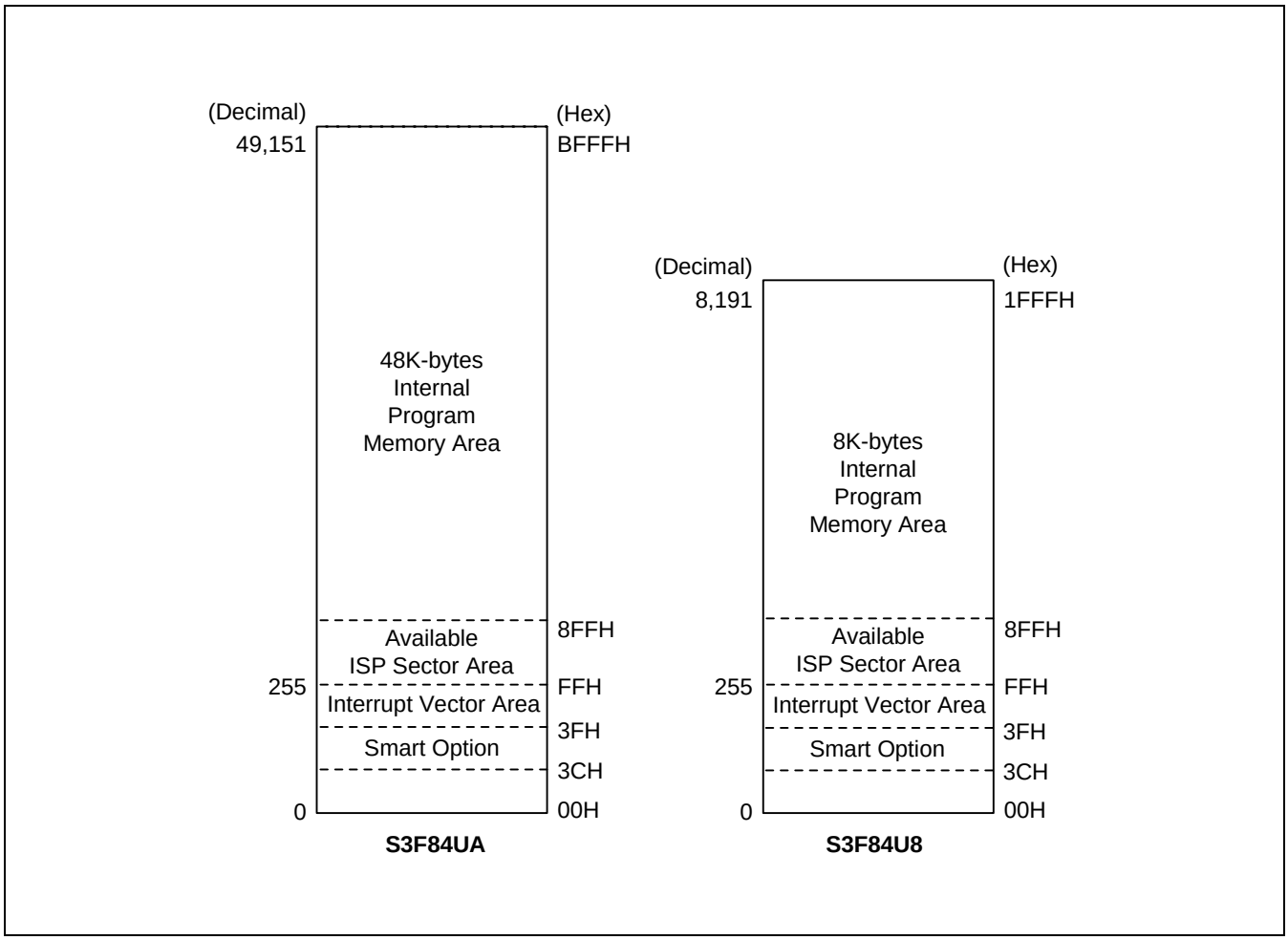


图 5-3 ROM 中断向量地址区

表 5-1 中断向量

向量地址		中断源	中断请求		复位/清零	
十进制值	十六进制值		中断级	优先级	H/W	S/W
256	100H	Basic Timer 溢出中断	Reset	–	√	
206	CEH	Timer A 匹配/捕获中断	IRQ0	0		√
208	D0H	Timer A 溢出中断		1	√	√
210	D2H	Timer B 匹配中断	IRQ1	–	√	
212	D4H	Timer C 匹配/溢出中断	IRQ2	–	√	√
216	D8H	Timer D0 匹配/捕获中断	IRQ3	0		√
218	DAH	Timer D0 溢出中断		1	√	√
220	DCH	Timer D1 匹配/捕获中断		2		√
222	DEH	Timer D1 溢出中断		3	√	√
228	E4H	SIO 中断	IRQ4	0		√
230	E6H	钟表定时器溢出中断		1		√
232	E8H	UART 0 数据发送中断	IRQ5	0		√
234	EAH	UART 0 数据接收中断		1		√
236	ECH	UART 1 数据发送中断		2		√
238	EEH	UART 1 数据接收中断		3		√
240	F0H	P3.0 外部中断	IRQ6	0		√
242	F2H	P3.1 外部中断		1		√
244	F4H	P3.2 外部中断		2		√
246	F6H	P3.3 外部中断		3		√
248	F8H	P3.4 外部中断	IRQ7	0		√
250	FAH	P3.5 外部中断		1		√
252	FCH	P3.6 外部中断		2		√
254	FEH	P3.7 外部中断		3		√

注释:

1. 中断优先级反向排序：“0”是最高优先级，“1”是次高，依次类推。
2. 同一个中断级中有竞争的两个或者更多的中断源同时发出中断请求时，
   中断向量地址最小的那个中断源比其他任何一个中断源拥有更高的优先级。给定的中断级中的优先级顺序由硬件决定。

5.1.5.2 使能/禁止中断指令 (EI, DI)

执行使能中断指令(EI)使能全局中断结构。在随后的操作中，根据已有的优先级顺序响应所有的中断。

注释: 复位后，为了使能全局中断处理，应该在系统初始化程序中包含 EI 指令。

正常工作时时，用户可在任何时间执行 DI(禁止中断)来禁止全局中断处理。EI 和 DI 指令将改变 SYM 寄存器0位的值。

5.1.5.3 系统级中断控制寄存器

除了特定中断源的控制寄存器外，4 个系统级的寄存器页参与中断处理控制：

- 中断屏蔽寄存器，IMR，使能(解除屏蔽)或禁止(屏蔽)中断级
- 中断优先级寄存器，IPR，控制各个中断级之间的相对优先级
- 中断请求寄存器，IRQ，包含每个中断级的中断标志位(不同于中断源标志位)
- 系统模式寄存器，SYM，使能或禁止全局中断处理(SYM 的设置还可以使能快速中断并在硬件支持的情况下控制外部接口)

表 5-2 中断控制寄存器概述

控制寄存器	ID	R/W	功能描述
中断屏蔽寄存器	IMR	R/W	IMR 寄存器中的位设置可以使能或禁止 IRQ0–IRQ7 8 个中断级中任意一个的中断处理。
中断优先级寄存器	IPR	R/W	控制各个中断级之间的相对处理优先级。S3F84UA/F84U8 中的 8 个中断级被分作 3 个组：A，B 和C 组。A 组包括 IRQ0 和 IRQ1，B 组包括IRQ2，IRQ3 和 IRQ4，C 组包括 IRQ5，IRQ6 和 IRQ7。
中断请求寄存器	IRQ	R	该寄存器包含每个中断级的中断请求标志位。
系统模式寄存器	SYM	R/W	该寄存器使能或禁止快速中断处理，动态全局中断处理和外部接口控制(S3F84UA/F84U8 MCU 中支持外部存储接口)。

注释: 在改变 IMR 寄存器的内容前，必须禁止所有的中断。建议使用 DI 指令。

5.1.6 中断处理控制要点

中断处理控制可通过两种方式完成：全局或特定中断级和中断源。中断结构中系统级控制要点如下：

- 全局中断的使能和禁止(通过 EI 和 DI 指令或者直接操作 SYM.0)
- 中断级的使能/禁止设置(通过 IMR 寄存器)
- 中断级优先级设置(通过 IPR 寄存器)
- 通过相应的外围控制寄存器中的中断源使能/禁止设置

注释: 当为处理中断而编写应用程序时，务必包含必要的寄存器卷地址(寄存器指针)信息。

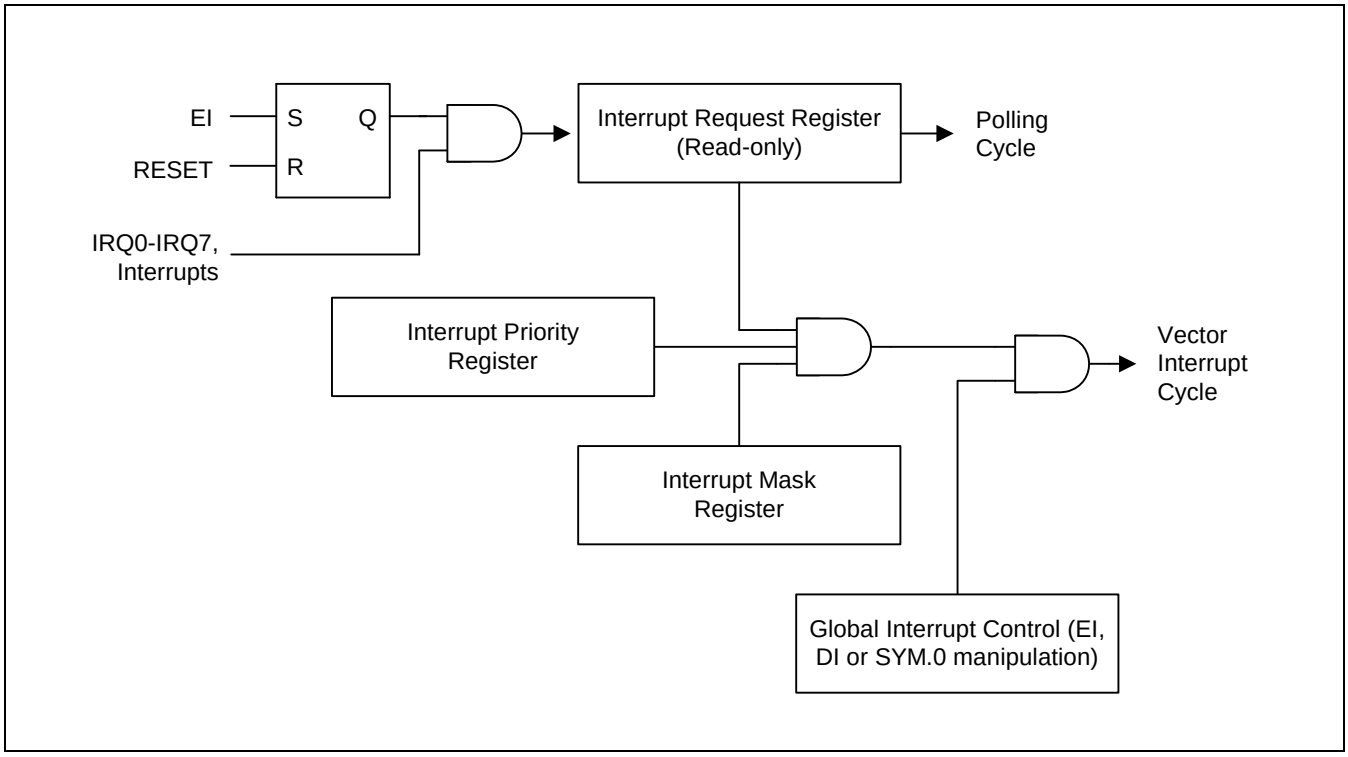


图 5-4 中断功能框图

5.1.7 外围中断控制寄存器

每一个中断源都受控于一个或多个外围控制寄存器，可让用户控制对应外围设备产生的中断(*表 5-3*)

表 5-3 中断源控制寄存器和数据寄存器

中断源	中断级	寄存器	在 Set 1 中的位置
Timer A 匹配/捕获中断 Timer A 溢出中断	IRQ0	TACON TACNT TADATA	E2H, bank 0 E0H, bank 0 E1H, bank 0
Timer B 匹配中断	IRQ1	TBCON TBDATAH TBDATAL	E3H, bank 0 E4H, bank 0 E5H, bank 0
Timer C 匹配/溢出中断	IRQ2	TCCON TCCNT TCDATA	ECH, bank 0 EAH, bank 0 EBH, bank 0
Timer D0 匹配/捕获中断 Timer D0 溢出中断 Timer D1 匹配/捕获中断 Timer D1 溢出中断	IRQ3	TD0CON TD0CNTH TD0CNTL TD0DATAH TD0DATAL TD1CON TD1CNTH TD1CNTL TD1DATAH TD1DATAL	FAH, bank 1 F6H, bank 1 F7H, bank 1 F8H, bank 1 F9H, bank 1 FBH, bank 1 FCH, bank 1 FDH, bank 1 FEH, bank 1 FFH, bank 1
SIO 中断 钟表定时器溢出中断	IRQ4	SIOCON SIODATA SIOPS WTCON	E7H, bank 0 E8H, bank 0 E9H, bank 0 E6H, bank 0
UART 0 数据发送中断 UART 0 数据接收中断 UART 1 数据发送中断 UART 1 数据接收中断	IRQ5	UART0CONH UART0CONL UDATA0, BRDATA0 UART1CONH UART1CONL UDATA1, BRDATA1	EEH, bank 0 EFH, bank 0 F0H, F1H, bank 0 F2H, bank 0 F3H, bank 0 F4H, F5H, bank 0
P3.0 外部中断 P3.1 外部中断 P3.2 外部中断 P3.3 外部中断	IRQ6	P3CONL P3INTL P3PND	E5H, bank 1 E7H, bank 1 E8H, bank 1
P3.4 外部中断 P3.5 外部中断 P3.6 外部中断 P3.7 外部中断	IRQ7	P3CONH P3INTH P3PND	E4H, bank 1 E6H, bank 1 E8H, bank 1

注释: 如果 IMR 寄存器中的中断未屏蔽(使能中断级)，必须在 DI 指令执行后写中断中的标志位和使能位。

P3 口状态改变时，将产生不希望发生的外部中断。这些不希望发生的中断因 P3 口状态改变而产生。因此，在 P3 口状态变为任何值前，用户需要执行 DI，EI 指令以及清除标志位。

按照以下步骤进行改变：

1. 执行 DI 指令。
2. 修改 P3CONH/L，P3INTH/L 和 P3PUR。
3. 将 P3 口中断标志位寄存器(P3PND)清除为“00000000B”。
4. 执行 EI 指令。

编程实例 5-1 如何避免不希望的外部中断发生

例：

1．本例示范如何从普通端口模式切换到中断端口模式

```
SB1
LD    P3CONH,#10101010B          ; P3.7~.4 复用功能 (LCD 信号)
LD    P3CONL,#10101010B          ; P3.3~.0 复用功能 (LCD 信号)
•
•
•
DI                               ; 设置为中断设置模式
LD    P3CONH,#00000000B          ; P3.7~.4 用于中断的输入模式
LD    P3CONL,# 00000000B         ; P3.3~.0 用于中断的输入模式
LD    P3INTH,#01010101B          ; P3.7~.4 使能下降沿中断
LD    P3INTL,# 01010101B         ; P3.3~.0 使能下降沿中断
LD    P3PUR,#11111111B           ; P3.7~.0 使能上拉电阻
LD    P3PND,#00000000B           ; P3.7~.0 清除中断标志位
EI
•
•
```

2．本例示范如何从中断端口模式切换到普通端口模式

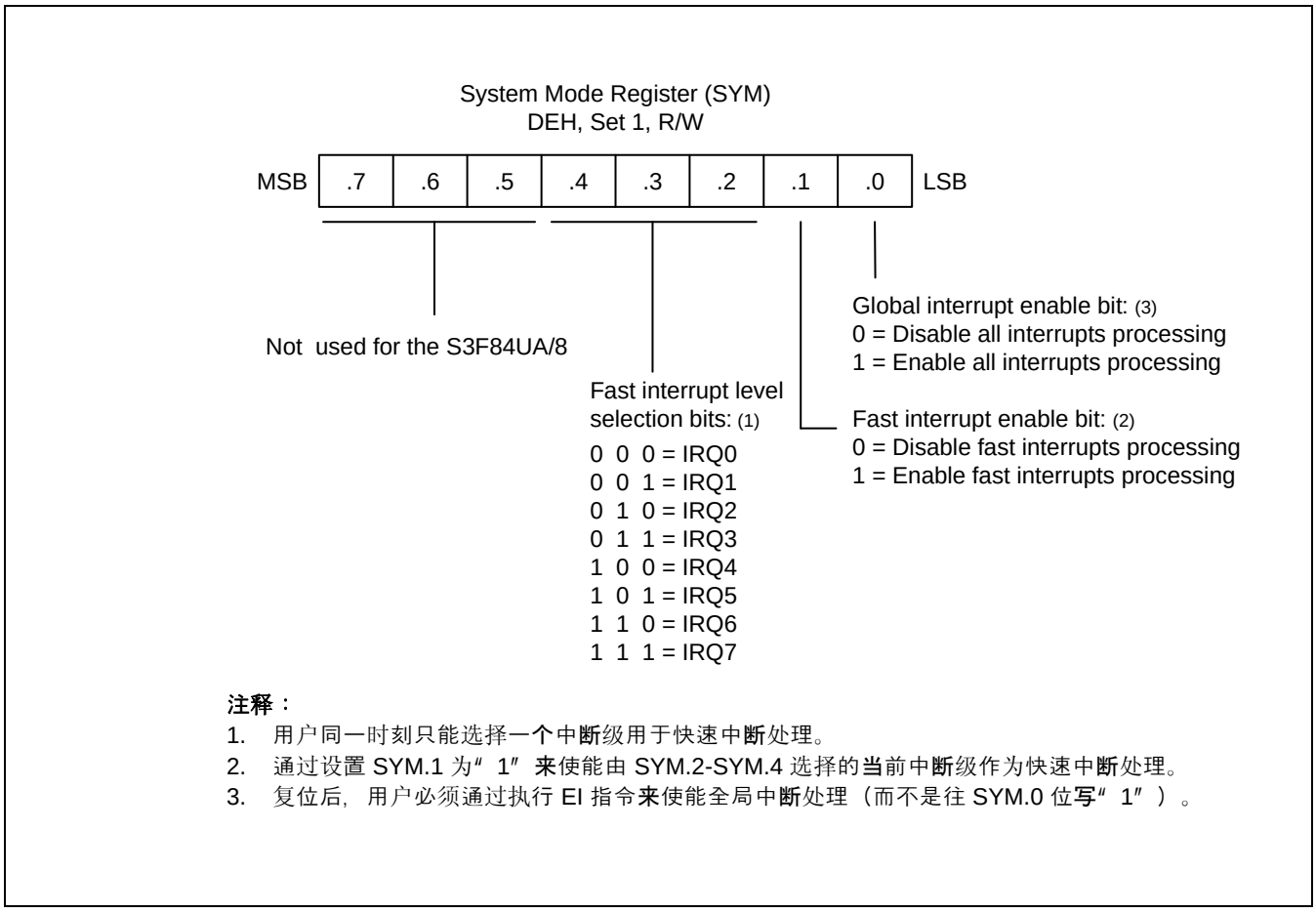
```
SB1
LD    P3CONH,# 00000000B         ; P3.7~.4 用于中断的输入模式
LD    P3CONL,# 00000000B         ; P3.3~.0 用于中断的输入模式
•
•
•
DI                               ; 设置为普通端口
LD    P3CONH,# 10101010B         ; P3.7~.4 复用功能 (LCD 信号)
LD    P3CONL,# 10101010B         ; P3.3~.0 复用功能 (LCD 信号)
LD    P3INTH,# 00000000B         ; P3.7~.4 禁止下降沿中断
LD    P3INTL,# 00000000B         ; P3.3~.0 禁止下降沿中断
LD    P3PUR,# 00000000B          ; P3.7~.0 禁止上拉电阻
LD    P3PND,# 00000000B          ; P3.7~.0 清除中断标志位
EI
•
•
```

5.1.8 系统模式寄存器 (SYM)

系统模式寄存器 SYM(地址：DEH，Set 1)可以用来使能/禁止全局中断处理，以及控制快速中断处理(*图 5-5*)。

复位将 SYM.1 和 SYM.0 清零，用于快速中断级选择的 3 位 SYM.4–SYM.2 的值不确定。

EI 和 DI 指令可通过修改 SYM 寄存器的第 0 位来分别使能或禁止全局中断处理。
复位操作后，为了使能中断处理，必须在初始化程序中使用 EI 指令。尽管在正常工作中，用户可以直接通过操作 SYM.0 位来使能或禁止中断，但还是建议尽量使用 EI 和 DI 指令。



注释：
1. 用户同一时刻只能选择一个中断级用于快速中断处理。
2. 通过设置 SYM.1 为“ 1” 来使能由 SYM.2-SYM.4 选择的当前中断级作为快速中断处理。
3. 复位后，用户必须通过执行 EI 指令来使能全局中断处理（而不是往 SYM.0 位写“ 1” ）。

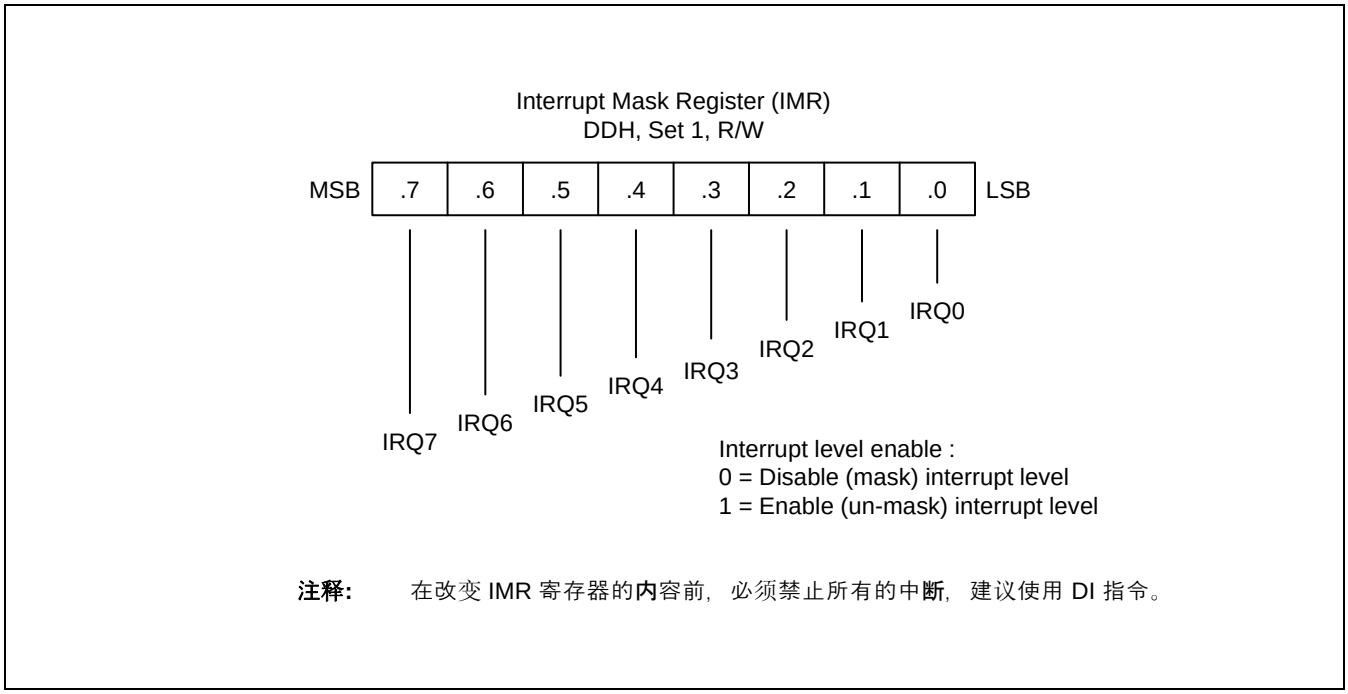
图 5-5 系统模式寄存器 (SYM)

5.1.9 中断屏蔽寄存器 (IMR)

中断屏蔽寄存器 IMR(地址：DDH，Set 1)可以用来使能或禁止各个中断级的中断处理。复位后，IMR 的所有位都是不确定的，因此必须在初始化程序中根据需要设置该寄存器。

IMR 中的每一位都对应一个特定的中断级：第 1 位对应于 IRQ1，第 2 位对应于 IRQ2，依次类推。当 IMR 中某个中断级位被清为“0”后，与之对应的中断处理就被禁止(屏蔽)了。当用户将 IMR 中某个级的位设置为“1”时，相应中断级的中断级被使能(解除屏蔽)了。

IMR 寄存器映射地址为：DDH，Set 1。通过寄存器寻址模式支持对位进行读写操作指令。



注释: 在改变 IMR 寄存器的内容前，必须禁止所有的中断，建议使用 DI 指令。

图 5-6 中断屏蔽寄存器 (IMR)

5.1.10 中断优先级寄存器 (IPR)

中断优先级寄存器 IPR(地址：FFH，Set 1，Bank 0)可以用来设置 MCU 中断机构中各个中断级的相对优先级。复位后，IPR 的所有位都是不确定的， 因此在初始化程序中根据需要设置该寄存器。

当超过一个中断源同时发出中断请求时，优先级最高的那个中断源首先被响应。如果同一个中断级的两个中断源同时发出中断请求时，通常中断向量地址较小的那个具有较高的优先级(优先级由硬件决定)。

为支持对相对中断优先级的编程，根据中断逻辑分为组和子分组。请注意 IPR 寄存器优先级定义的 IPR 逻辑中使用到的组和子分组(*图 5-7*)：

Group A IRQ0, IRQ1
Group B IRQ2, IRQ3, IRQ4
Group C IRQ5, IRQ6, IRQ7

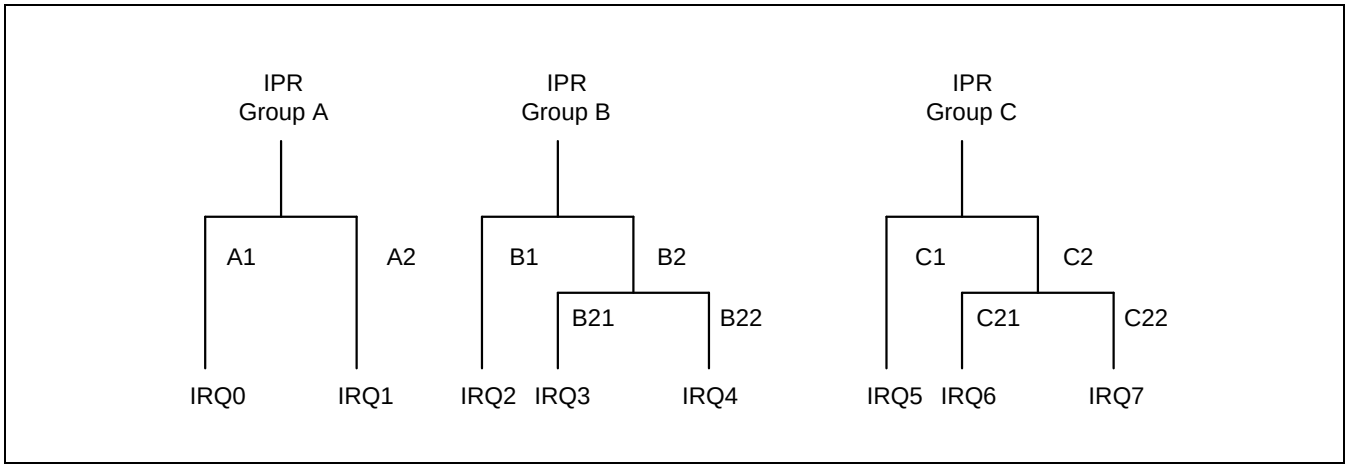


图 5-7 中断请求优先级组

图 5-8 所示，IPR.7，IPR.4 和 IPR.1 控制中断 ABC 组的相对优先级。例如，当这 3 为设置为“001B”时，此时中断组优先顺序为 B > C > A；设置为“101B”时，优先顺序为 C > B > A。

IPR 其他位控制功能如下：

- IPR.5 位控制中断 C 组内中断级的相对优先级
- 中断 C 组包含一个由中断级 5，6 和 7 之间额外优先级关系的子分组，其中 IPR.6 位定义中断 C 子分组的关系，IPR.5 控制中断 C 组
- IPR.0 位控制 IRQ0 和 IRQ1 的相对优先级

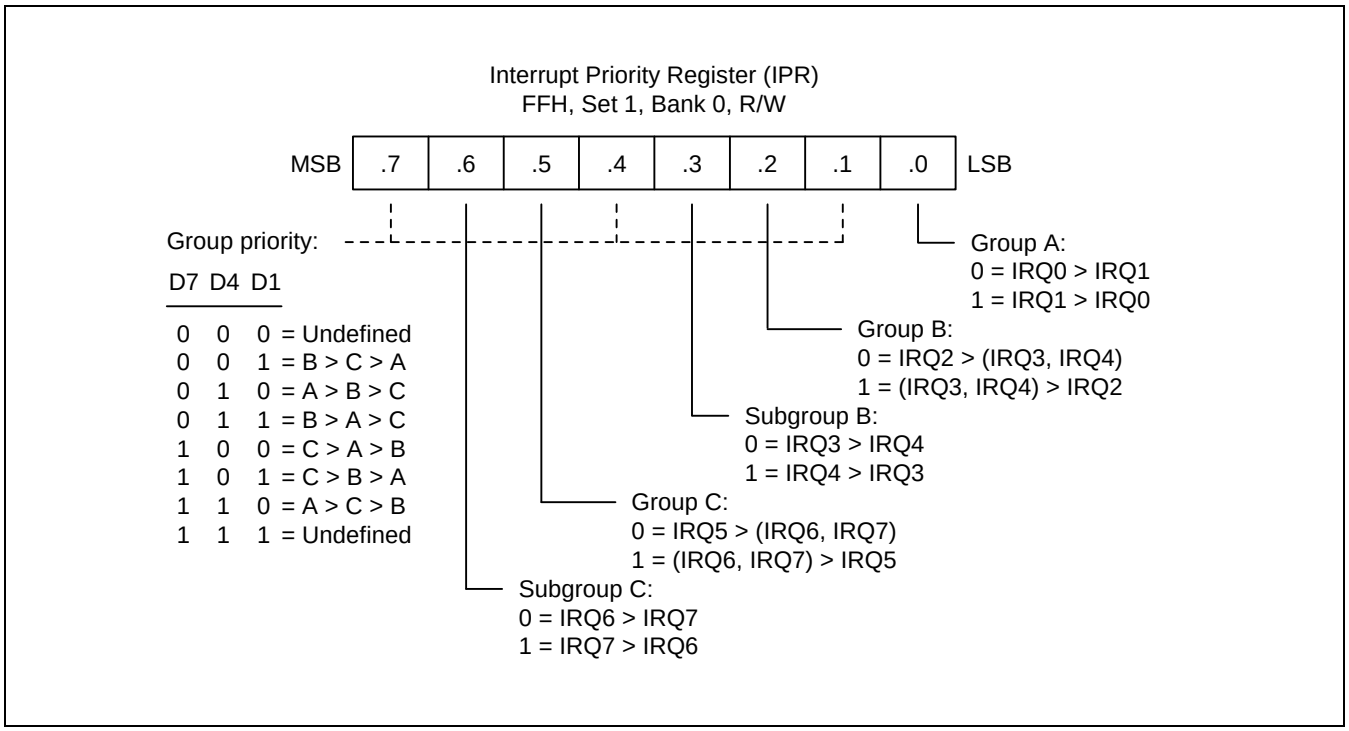


图 5-8 中断优先级寄存器 (IPR)

5.1.11 中断请求寄存器 (IRQ)

用户可以通过查询中断请求寄存器 IRQ(地址：DCH，Set 1)的每一位监视 MCU 中断结构中所有中断级的中断请求状况。每一位都对应于一个序号相同的中断级：第 0 位对应于 IRQ0，第 1 位对应于 IRQ1，依此类推。“0”表示当前的中断级没有中断请求发生；“1”表示当前的中断级有中断请求。

IRQ 位只读，且支持寄存器寻址。用户可以在任何时刻使用位或字节方式访问来读取(测试)IRQ 寄存器的内容，以判断特定中断级的当前中断请求状态。复位后，IRQ 中所有的状态位清除为“0”。

即使是执行 DI 指令后(即全局中断处理禁止)，用户仍可以查询 IRQ 寄存器的值。此时如果中断发生，CPU 不会响应。当用户仍可以查询 IRQ 寄存器来检测中断请求。通过这种方法，用户可以在全局中断结构禁止时也能判断中断发生请况。

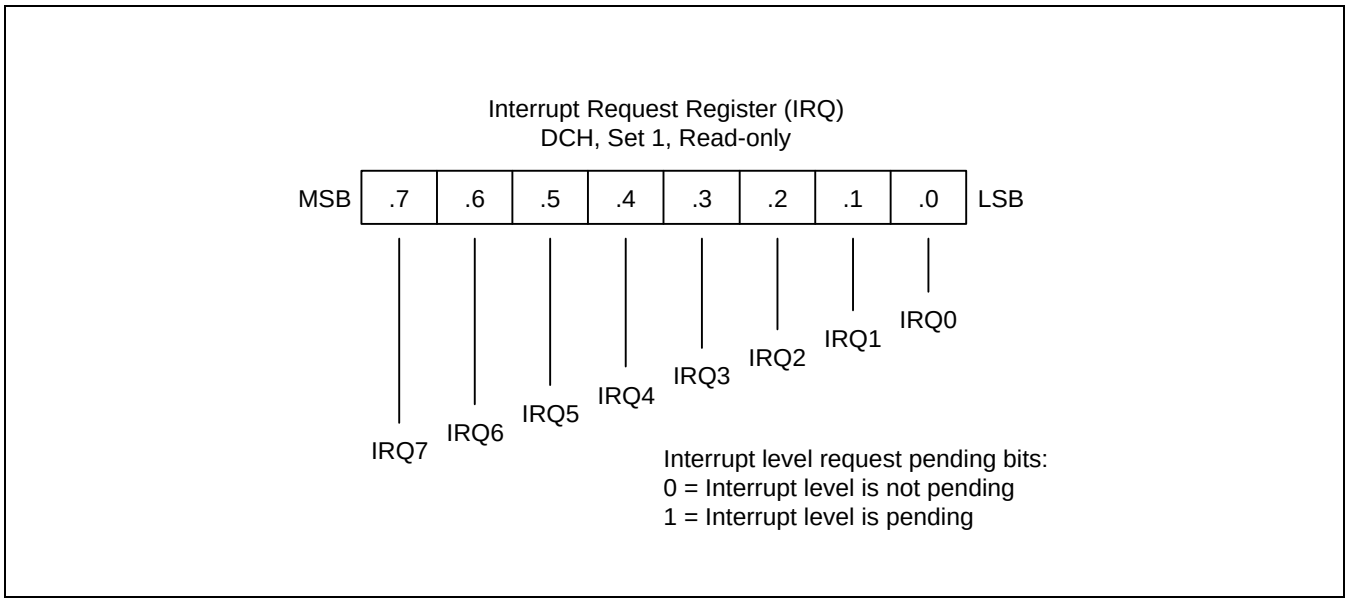


图 5-9 中断请求寄存器 (IRQ)

5.1.12 中断标志位类型

5.1.12.1 概述

有两种中断标志位：一种会在中断服务程序被响应并执行后硬件自动清零，另一种必须在中断服务程序中由软件清零。

5.1.12.2 硬件自动清零的标志位

对于硬件自动清零的中断标志位来说，当中断请求发生时，中断逻辑自动将相应的标志位设置为“1”。之后，触发一个 IRQ 脉冲来通知 CPU 有一个中断正等待处理。CPU 随后发送一个 IACK 给中断源确认接收请求，然后执行服务程序，最后将标志位清为“0”。这种类型的标志位未映射，因此也无法被应用软件读或写。

在 S3F84UA/F84U8 中断结构中，Timer A 溢出中断(IRQ0) ，Timer B 匹配中断(IRQ1) ，Timer C 溢出中断(IRQ2)和 Timer D0/D1 溢出中断(IRQ3)隶属于此类中断，硬件会自动清除中断标志位。

5.1.12.3 需要在中断服务程序里(软件)清零的标志位

第二种中断标志位必须由软件清零。中断服务程序返回(IRET)之前，服务程序必须清除相应的标志位。可通过向位于中断源模式或控制寄存器的中断标志位写“0”来实现。

编程实例 5-2 如何清除中断标志位

下面例子中示范使用 Load 指令来清除中断标志位

例：

```
1.   SB1
     LD    P3PND, #11111011B            ; 清除 P3.2 的中断标志位
     •
     •
     •
     IRET

2.   SB0
     LD    INTPND, #11111101B           ; 清除 Timer A 的匹配/捕获中断标志位
     •
     •
     •
     IRET
```

5.1.13 中断源响应步骤

中断请求的查询和中断服务的步骤如下：

1. 某个中断源通过将中断请求位设置为“1”产生一个中断申请。
2. CPU 查询并认出该中断源已被挂起。
3. CPU 检查该中断源的中断级。
4. CPU 发出一个中断确认信号。
5. 中断逻辑决定中断的向量地址。
6. 开始执行中断服务程序，中断源的标志位被清为“0”(通过硬件或软件)。
7. 该中断服务程序结束后，CPU 继续查询中断请求。

5.1.14 中断服务程序

满足以下条件时，CPU 才会接收一个中断请求：

- 使能全局中断处理 (EI, SYM.0 = "1")
- 使能中断级 (IMR 寄存器)
- 如果当前有多个中断级发出请求，该中断级必须具有最高的优先级
- 该中断源必须使能 (外围控制寄存器)

待上述所有条件均满足时，指令周期结束时对中断请求进行确认。
之后，CPU 初始化一个中断机器周期以完成如下处理过程：

1. 重置(清零) SYM 寄存器的中断使能位(SYM.0)，禁止所有的子中断。
2. 将程序计数器(PC)和状态寄存器压入系统栈进行保留。
3. 跳转到中断向量地址，获取中断服务程序地址。
4. 将控制权移交给中断服务程序。

当中断服务程序执行完毕，CPU 产生一个中断返回(IRET)。
IRET 指令将 PC 和状态标志重新装入当前寄存器，同时将SYM.0 位置“1”，使能 CPU 响应之后的中断请求。

5.1.15 中断向量地址的生成

中断向量位于 ROM (00H–FFH)，包含中断结构中每个中断级对应的中断服务程序的地址。
向量化的中断处理流程如下：

1. 将程序计数器(PC)低字节的值压入栈。
2. 将程序计数器(PC)高字节的值压入栈。
3. 将 FLAG 寄存器的值压入栈。
4. 从向量地址中获取中断服务程序的高字节地址。
5. 从向量地址中获取中断服务程序的低字节地址。
6. 跳转执行 16 位中断向量地址所确定的中断服务程序。

注释: 16 位向量地址的起始地址始终位于 ROM 地址空间 00H–FFH 中的某个偶数地址。

5.1.16 向量化的中断嵌套

在一个低优先级中断请求正在处理时，可以嵌入一个更高的优先级的中断请求。为了实现这个目的，用户需按如下步骤：

1. 将当前的 8 位中断屏蔽寄存器(IMR)的值压入栈(指令：PUSH IMR)。
2. 载入一个新的屏蔽值到 IMR 寄存器，除了希望嵌套的那个高优先级的中断外，屏蔽其他所有中断级。
3. 执行 EI 指令使能中断处理(如果更高优先级的中断产生时将可以被响应) 。
4. 当低优先级的中断服务程序执行完毕后，通过将先前的屏蔽值从栈中弹出(指令：POP IMR)，重新载入到 IMR 中。
5. 执行 IRET。

根据不同的应用，用户可能可以简化上述一些步骤。

5.1.17 指令指针 (IP)

S3C8- 系列的所有 MCU 都采用指令指针(IP)以支持高速中断处理特性，也就快速中断。IP 由寄存器对 DAH 和 DBH 组成。IP 寄存器的名称为 IPH(高字节，IP15–IP8)和 IPL(低字节，IP7–IP0)。

5.1.18 快速中断处理

快速中断处理允许给定中断级的一个中断处理完毕大概需要 6 个时钟周期，普通的中断处理需要 16 个时钟周期。用户可以通过往 SYM.4–SYM.2 写合适的 3 位值来选择某个特定的中断级作为快速中断处理。之后，用户通过将 SYM.1 位设置为“1”来使能选定中断级作为快速中断处理。

5.1.18.1 快速中断处理 (续)

支持快速中断处理的两种其他系统寄存器：

- 指令指针(IP)包含了服务程序的起始地址(中断处理后会和程序计数器的值交换)
- 当一个快速中断发生时，FLAGS 寄存器的内容会载入到一个未映射的专用寄存器，称为 FLAGS' ("FLAGS prime")

注释: 对于 S3F84UA/F84U8 MCU 来说，所有 8 个中断级（IRQ0–IRQ7）中的任何一个都可以选择作为快速中断处理。

5.1.18.2 快速中断处理的初始化

按照以下步骤对快速中断处理进行初始化：

1. 将中断服务程序的起始地址载入指令指针(IP)。
2. 将中断级序号(IRQn)载入快速中断选择区域(SYM.4–SYM.2)。
3. 将 SYM 寄存器中的快速中断使能位置"1" 。

5.1.18.3 快速中断服务程序

当选作快速中断处理的那个中断级中的任何一个中断发生时，将按以下步骤：

1. 将指令指针和 PC 的内容交换。
2. 将 FLAG 寄存器的置写入 FLAGS' ("FLAGS prime") 寄存器。
3. FLAGS 寄存器中的快速中断状态位置高。
4. 开始中断服务。
5. 假设快速中断状态位已置高，快速中断服务程序结束后，指令指针和 PC 的值再次交换。
6. FLAGS' ("FLAGS prime") 的内容自动复制回 FLAGS 寄存器。
7. FLAGS 寄存器中的快速中断状态位自动清零。

5.1.19 中断标志位类型之间的关系

如之前描述的那样，有两种类型的中断标志位：一种会在中断服务程序被响应并执行后硬件自动清零，另一种必须在应用程序的中断服务程序中由软件清零。用户也选择这两种标志位清除方式(通过硬件或软件)作为快速中断处理.

5.1.20 编程指导

切记，设置/清除 SYM 寄存器中的快速中断使能位是使能/禁止快速中断的唯一方法。执行 EI 或 DI 指令使能或禁止全局中断处理，也包括快速中断。如果用户要使用快速中断，切记当快速中断服务程序执行完毕后重新装载一个新的开始地址到 IP 寄存器中。

6 指令集

6.1 概述

SAM8RC 指令集支持对大容量寄存器卷的操作，一共包含 78 条指令，具有强大的数据处理能力。
指令集特性如下：

- 实现了所有 8 位算术和逻辑操作，包括乘法和除法
- 没有专门的输入/输出指令 (输入/输出控制寄存器和数据寄存器直接映射到寄存器卷中)
- 十进制调整，包括 BCD 操作
- 16 位(字)数据可自增或自减
- 灵活的位寻址，循环和移位指令

6.1.1 数据类型

SAM8 CPU 可以实现位操作，字节操作，BCD 数字操作和双字节操作。寄存器的每个位可以被置 1，清 0，取反和测试。一个字节的各位按从 7 到 0 编号，其中 0 位是最低位 (在最右边)。

6.1.2 寄存器访问

为访问寄存器，应指定寄存器卷中地址为 0 – 255 之间 的 8 位地址或工作寄存器的 4 位地址。
工作寄存器中，寄存器对可以访问 16 位程序存储空间和数据存储空间。关于寄存器访问的详细描述，请参考第 2 章“地址空间”。

6.1.3 寻址模式

有 7 种寻址模式：寄存器寻址(R)，间接寄存器寻址(IR)，偏址寻址(X)，直接寻址(DA)，相对地址寻址(RA)，立即数寻址(IM)和间接寻址(IA)。有关寻址模式的详细描述，请参考第 3 章“寻址模式”。

表 6-1 指令集简介

助记符	操作数	指令介绍
数据传送类指令		
CLR	dst	清零
LD	dst,src	传送数据
LDB	dst,src	传送位数据
LDE	dst,src	传送数据 (访问外部数据存储空间)
LDC	dst,src	传送数据 (访问程序存储空间)
LDED	dst,src	传送数据后地址减 1 (访问外部数据存储空间)
LDCD	dst,src	传送数据后地址减 1 (访问程序存储空间)
LDEI	dst,src	传送数据后地址加 1 (访问外部数据存储空间)
LDCI	dst,src	传送数据后地址加 1 (访问程序存储空间)
LDEPD	dst,src	传送数据前地址减 1 (访问外部数据存储空间)
LDCPD	dst,src	传送数据前地址减 1 (访问程序存储空间)
LDEPI	dst,src	传送数据前地址加 1 (访问外部数据存储空间)
LDCPI	dst,src	传送数据前地址加 1 (访问程序存储空间)
LDW	dst,src	传送字数据
POP	dst	出栈
POPUD	dst,src	弹出用户栈 (减 1)
POPUI	dst,src	弹出用户栈 (加 1)
PUSH	src	压栈
PUSHUD	dst,src	压入用户栈 (减 1)
PUSHUI	dst,src	压入用户栈 (加 1)

注释: LDE, LDED, LDEI, LDEPP 和 LDEPI 指令可用来读/写 64K 字节的外部数据存储空间。

表 6-2 指令集简介

助记符	操作数	指令描述
算术操作类指令		
ADC	dst,src	带进位加法
ADD	dst,src	不带进位加法
CP	dst,src	比较指令
DA	dst	十进制调整
DEC	dst	字节减 1
DECW	dst	字减 1
DIV	dst,src	除法指令
INC	dst	字节加 1
INCW	dst	字加 1
MULT	dst,src	乘法指令
SBC	dst,src	带借位减法
SUB	dst,src	不带借位减法
逻辑操作类指令		
AND	dst,src	逻辑与
COM	dst	取反
OR	dst,src	逻辑或
XOR	dst,src	逻辑异或
程序控制指令		
BTJRF	dst,src	位测试，如果为 0 跳转
BTJRT	dst,src	位测试，如果为 1 跳转
CALL	dst	调用子程序
CPIJE	dst,src	比较，如果相等跳转
CPIJNE	dst,src	比较，如果不等跳转
DJNZ	r,dst	寄存器减 1，不为 0 跳转
ENTER		进入
EXIT		跳出
IRET		中断返回
JP	cc,dst	有条件跳转
JP	dst	无条件跳转
JR	cc,dst	有条件相对跳转
NEXT		Next 指令
RET		子程序返回
WFI		等待中断
位操作指令		

助记符	操作数	指令描述
BAND	dst,src	位与
BCP	dst,src	位比较
BITC	dst	位取反
BITR	dst	位清零
BITS	dst	位置 1
BOR	dst,src	位或
BXOR	dst,src	位异或
TCM	dst,src	取反后位测试
TM	dst,src	位测试指令
循环和移位指令		
RL	dst	循环左移
RLC	dst	带进位循环左移
RR	dst	循环右移
RRC	dst	带进位循环右移
SRA	dst	算术右移
SWAP	dst	交换
CPU控制指令		
CCF		进位标志取反
DI		屏蔽全局中断
EI		使能全局中断
IDLE		进入 IDLE 模式
NOP		空操作
RCF		进位标志清零
SB0		选择寄存器块 Bank 0
SB1		选择寄存器块 Bank 1
SCF		进位标志置 1
SRP	src	设置寄存器指针
SRP0	src	设置寄存器指针 0
SRP1	src	设置寄存器指针 1
STOP		进入 STOP 模式

6.2 标志寄存器(FLAGS)

8 位标志寄存器 FLAGS 描述当前 CPU 的操作状态。其中的 4 位(FLAGS.4–FLAGS.7)可以用于测试和条件转移指令；另外两个标志位 FLAGS.3 和标志位 FLAGS.2 用于 BCD 算术操作。

还有一个标志位 FLAGS.1 用于指示快速中断处理；FLAGS.0 位表示当前正在寻址的 Bank 地址状态，是 Bank 0 还是Bank 1。

只要结果不影响到状态位，FLAGS 寄存器可以通过指令(如 LOAD 指令)置 1 或者清 0。逻辑和算术类操作指令，例如与，或，异或，加法和减法操作，会影响到标志位寄存器。例如，AND 指令会根据结果改变 Zero, Sign 和 Overflow (Z，S，O)标志。如果 AND 指令使用 FLAGS 作为目标寄存器，将会同时发生两次 FLAGS 寄存器的写操作，造成不可预知的后果。

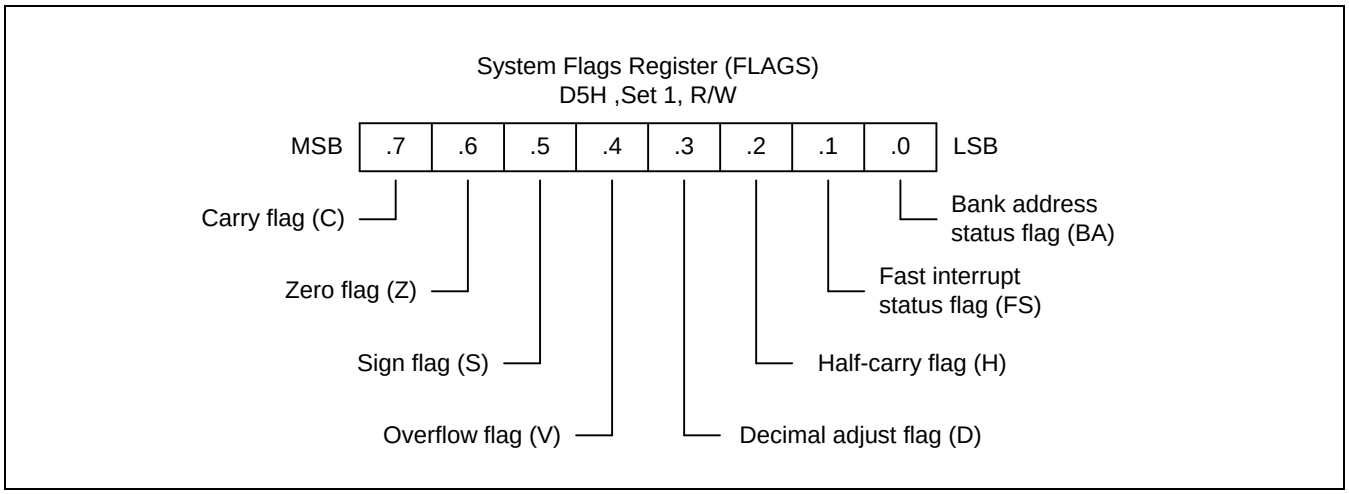


图 6-1 系统标志寄存器 (FLAGS)

6.2.1 标志位描述

C 进(借)位标志 (FLAGS.7)

如果算术操作后，最高位产生进位或借位，则此标志位为 1。移位，循环移位操作之后，此位保存最后移出的那一位值。指令可以对此位置 1，清 0，取反操作。

Z 零标志位 (FLAGS.6)

如果算术，逻辑操作的结果为 0 ，则此标志位为 1。位测试指令，移位指令，循环移位指令都会影响此标志位，如果结果为逻辑 0，则此标志位为 1。

S 符号标志位 (FLAGS.5)

算术，逻辑，循环，移位操作之后，操作结果最高位的状态反映在符号位。逻辑 0 表示操作结果是正数，逻辑 1 表示操作数结果是负数。

V 溢出标志 (FLAGS.4)

当操作结果大于 +127 或 小于 –128 时，溢出标志将会置 1。逻辑操作之后，它将会被清 0。

D 十进制调整标志 (FLAGS.3)

DA 标志用于指明 BCD
操作中何种类型的指令是最后被执行的，因此随后的十进制调整指令可以正确执行。编程者通常不能访问 DA 位，也不能用来做测试的条件码。

H 半字节进(借)位标志 (FLAGS.2)

当加法操作时第 3 位产生进位或减法操作时向第 4 位发生借位，H标志将被置 1。
通常用在DA指令中，将前一次的加法或减法结果(二进制码)转化为十进制结果(BCD)。程序通常不会直接用到 H 标志。

FIS 快速中断状态标志 (FLAGS.1)

在快速中断执行期间，FIS 位被置 1；快速中断返回时，FIS 位被清 0。当 FIS 被置 1 时，将禁止所有中断，直到 IRET 指令被执行后，才重新打开快速中断。

BA 寄存器块地址标志 (FLAGS.0)

BA 标志表明内部寄存器卷 Set 1 中哪个寄存器块被选中，是 Bank 0 还是 Bank 1。当执行 SB0 指令，BA标志被清零(选择 Bank 0)；当执行 SB1 指令，BA 标志被置 1(选择 Bank 1)。

6.2.2 指令集符号

表 6-3 标志位符号

标志位	描述
C	进(借)位标志
Z	零标志
S	符号标志
V	溢出标志
D	十进制调整标志
H	半字节进(借)位标志
0	清为逻辑 0
1	置为逻辑 1
*	根据相应操作置 1 或清 0
–	不受影响
x	不确定

表 6-4 指令集符号

标志位	描述
dst	目的操作数
src	源操作数
@	间接寄存器地址前缀
PC	程序计数器
IP	指令指针
FLAGS	标志寄存器 (D5H)
RP	寄存器指针
#	立即数或寄存器访问的前缀
H	十六进制后缀
D	十进制后缀
B	二进制后追
opc	指令代码

表 6-5 指令符号的定义

符号	描述	实际操作范围
cc	条件码	参考表格 6-6
r	工作寄存器	Rn (n = 0–15)
rb	工作寄存器的位 b	Rn.b (n = 0–15, b = 0–7)
r0	工作寄存器的位 0，最低位	Rn (n = 0–15)
rr	工作寄存器对	RRp (p = 0, 2, 4, ..., 14)
R	寄存器或者工作寄存器	reg or Rn (reg = 0–255, n = 0–15)
Rb	寄存器或者工作寄存器的位 b	reg.b (reg = 0–255, b = 0–7)
RR	寄存器对或工作寄存器对	reg or RRp (reg = 0–254, 只能是偶数 p = 0, 2, ..., 14)
IA	间接寻址模式	addr (addr = 0–254, 只能是偶数)
Ir	间接工作寄存器寻址	@Rn (n = 0–15)
IR	间接工作寄存器寻址或间接寄存器寻址	@Rn or @reg (reg = 0–255, n = 0–15)
Irr	间接工作寄存器对	@RRp (p = 0, 2, ..., 14)
IRR	间接寄存器对或间接工作寄存器对	@RRp or @reg (reg = 0–254, 只能是偶数 p = 0, 2, ..., 14)
X	基址寻址模式	#reg [Rn] (reg = 0–255, n = 0–15)
XS	短偏址寻址模式	#addr [RRp] (addr = 范围 –128 to +127, p = 0, 2, ..., 14)
xl	长偏址寻址模式	#addr [RRp] (addr = 范围 0–65535, p = 0, 2, ..., 14)
da	直接寻址模式	addr (addr = 范围 0–65535)
ra	相对地址寻址模式	addr (addr = 在 +127 到 -128 范围内的数字)
im	立即数寻址模式	#data (data = 0–255)
iml	长立即数寻址模式	#data (data = 范围 0–65535)

表 6-6 指令代码快速参考

指令代码对照图									
	LOWER NIBBLE (HEX)								
	–	0	1	2	3	4	5	6	7
U	0	DEC R1	DEC IR1	ADD r1,r2	ADD r1,Ir2	ADD R2,R1	ADD IR2,R1	ADD R1,IM	BOR r0–Rb
P	1	RLC R1	RLC IR1	ADC r1,r2	ADC r1,Ir2	ADC R2,R1	ADC IR2,R1	ADC R1,IM	BCP r1.b, R2
P	2	INC R1	INC IR1	SUB r1,r2	SUB r1,Ir2	SUB R2,R1	SUB IR2,R1	SUB R1,IM	BXOR r0–Rb
E	3	JP IRR1	SRP/0/1 IM	SBC r1,r2	SBC r1,Ir2	SBC R2,R1	SBC IR2,R1	SBC R1,IM	BTJR r2.b, RA
R	4	DA R1	DA IR1	OR r1,r2	OR r1,Ir2	OR R2,R1	OR IR2,R1	OR R1,IM	LDB r0–Rb
	5	POP R1	POP IR1	AND r1,r2	AND r1,Ir2	AND R2,R1	AND IR2,R1	AND R1,IM	BITC r1.b
N	6	COM R1	COM IR1	TCM r1,r2	TCM r1,Ir2	TCM R2,R1	TCM IR2,R1	TCM R1,IM	BAND r0–Rb
I	7	PUSH R2	PUSH IR2	TM r1,r2	TM r1,Ir2	TM R2,R1	TM IR2,R1	TM R1,IM	BIT r1.b
B	8	DECW RR1	DECW IR1	PUSHUD IR1,R2	PUSHUI IR1,R2	MULT R2,RR1	MULT IR2,RR1	MULT IM,RR1	LD r1, x, r2
B	9	RL R1	RL IR1	POPUD IR2,R1	POPUI IR2,R1	DIV R2,RR1	DIV IR2,RR1	DIV IM,RR1	LD r2, x, r1
L	A	INCW RR1	INCW IR1	CP r1,r2	CP r1,Ir2	CP R2,R1	CP IR2,R1	CP R1,IM	LDC r1, Irr2, xL
E	B	CLR R1	CLR IR1	XOR r1,r2	XOR r1,Ir2	XOR R2,R1	XOR IR2,R1	XOR R1,IM	LDC r2, Irr2, xL
	C	RRC R1	RRC IR1	CPIJE Ir,r2,RA	LDC r1,Irr2	LDW RR2,RR1	LDW IR2,RR1	LDW RR1,IML	LD r1, Ir2
H	D	SRA R1	SRA IR1	CPIJNE Irr,r2,RA	LDC r2,Irr1	CALL IA1		LD IR1,IM	LD Ir1, r2
E	E	RR R1	RR IR1	LDCD r1,Irr2	LDCI r1,Irr2	LD R2,R1	LD R2,IR1	LD R1,IM	LDC r1, Irr2, xs
X	F	SWAP R1	SWAP IR1	LDCPD r2,Irr1	LDCPI r2,Irr1	CALL IRR1	LD IR2,R1	CALL DA1	LDC r2, Irr1, xs

表 6-7 指令代码快速参考 (续)

指令代码对照图									
LOWER NIBBLE (HEX)									
	–	8	9	A	B	C	D	E	F
U	0	LD r1,R2	LD r2,R1	DJNZ r1,RA	JR cc,RA	LD r1,IM	JP cc,DA	INC r1	NEXT
P	1	↓	↓	↓	↓	↓	↓	↓	ENTER
P	2								EXIT
E	3								WFI
R	4								SB0
	5								SB1
N	6								IDLE
I	7	↓	↓	↓	↓	↓	↓	↓	STOP
B	8								DI
B	9								EI
L	A								RET
E	B								IRET
	C								RCF
H	D	↓	↓	↓	↓	↓	↓	↓	SCF
E	E								CCF
X	F	LD r1,R2	LD r2,R1	DJNZ r1,RA	JR cc,RA	LD r1,IM	JP cc,DA	INC r1	NOP

6.3 条件码

条件转移指令通常包含 4 位条件转移操作判断(cc)。这些条件转移操作判断的结果将决定程序的跳转方向。例如，比较操作后的“相等”条件转移，只有在两个操作数相等时该条件转移操作才会跳转。条件码列举在表格 *表 6-6*。

C，Z，S，V 等标志位用作条件转移判断位。指令将会根据这些标志位决定跳转方向。

表 6-8 条件码

二进制	助记符	描述	标志位设置
0000	F	逻辑假	–
1000	T	逻辑真	–
0111 (1)	C	有进位或借位	C = 1
1111 (1)	NC	无进位或借位	C = 0
0110 (1)	Z	结果为 0	Z = 1
1110 (1)	NZ	结果不为 0	Z = 0
1101	PL	正数	S = 0
0101	MI	负数	S = 1
0100	OV	溢出	V = 1
1100	NOV	没有溢出	V = 0
0110 (1)	EQ	相等	Z = 1
1110 (1)	NE	不相等	Z = 0
1001	GE	大于等于	(S XOR V) = 0
0001	LT	小于	(S XOR V) = 1
1010	GT	大于	(Z OR (S XOR V)) = 0
0010	LE	小于等于	(Z OR (S XOR V)) = 1
1111 (1)	UGE	无符号大于等于	C = 0
0111 (1)	ULT	无符号小于	C = 1
1011	UGT	无符号大于	(C = 0 AND Z = 0) = 1
0011	ULE	无符号小于等于	(C OR Z) = 1

注释:

1. 一次算术操作的结果可能同时影响两个标志位。例如，Z 符号位被置起时，Z, EQ 都为真。但是 ADD 指令操作之后，可能会用到 Z ；而 CP 指令操作之后，EQ 可能被用到。
2. 如果操作数涉及到无符号数，必须使用 UGE, ULT, UGT, ULE 等条件代码。

6.3.1 指令集描述

本章详细介绍了 S3F8- 系列单片机的指令操作，同时给出了具体的编程实例。
为便于参阅和快速查找，在介绍指令时采用了统一的格式。对每条指令，采用了如下的描述方法：

- 指令名称 (标号)
- 指令全称
- 源操作数/目的操作数的格式
- 具体指令的解释
- 每条指令操作的具体描述
- 每条指令对标志寄存器的影响
- 指令格式，执行周期和访问模式的详细介绍
- 每条指令的编程实例

6.3.1.1 ADC—带进位加法 (Add with Carry)

ADC dst, src

操作: dst ← dst + src + c

目的操作数加上源操作数和 C 位，所得结果保存在目的操作数。源操作数不受影响。
在多字节加法中，该指令允许把低字节的进位加到高字节的加法运算中。

标志位:

C: 如果加法运算中产生进位，则此位置 1；否则，清 0
Z: 如果运算结果为 0，则该位置 1；否则，清 0
S: 如果运算结果为负，则该位置 1；否则，清 0
V: 如果运算结果产生溢出，该位置 1；否则，清 0
D: 总是被清 0
H: 如果运算结果的低4位有进位，该位置 1；否则，清 0

格式:

	字节数	时钟周期	指令代码	寻址模式	
			(Hex)	dst	src
opc | dst | src	2	4	12	r	r
		6	13	r	lr
opc | src | dst	3	6	14	R	R
		6	15	R	IR
opc | dst | src	3	6	16	R	IM

编程实例

```
假设：R1 = 10H，R2 = 03H，C flag =“1”，寄存器 01H = 20H，
寄存器 02H = 03H，和寄存器 03H = 0AH：

ADC   R1,R2        → R1 = 14H, R2 = 03H
ADC   R1,@R2       → R1 = 1BH, R2 = 03H
ADC   01H,02H      → 寄存器 01H = 24H, 寄存器 02H = 03H
ADC   01H,@02H     → 寄存器 01H = 2BH, 寄存器 02H = 03H
ADC   01H,#11H     → 寄存器 01H = 32H

在第一个例子中，目的寄存器R1内容为 10H，进位标志位为 1，源寄存器 R2 为 03H。语句“ADC
R1,R2”把03H和进位位(“1”)累加到目的数 10H，结果为 14H，保存在 R1。
```

6.3.1.2 ADD—加法 (Add)

ADD dst, src

操作: dst ← dst + src

源操作数和目的操作数相加，结果保存在目的操作数，源操作数不受影响。

标志位:

C: 如果结果的最高位有进位，被置 1；否则，被清 0
Z: 如果结果为 0，被置 1；否则，被清 0
S: 如果结果是负数，被置 1；否则，被清 0
V: 如果有溢出，被置 1，也就是说，两个操作数符号相同，运算结果是相反的符号；否则，被清0
D: 总是被清 0
H: 如果结果的低4位有进位，被置 1；否则，被清 0

格式:

	字节数	时钟周期	指令代码	寻址模式	
			(Hex)	dst	src
opc | dst | src	2	4	02	r	r
		6	03	r	lr
opc | src | dst	3	6	04	R	R
		6	05	R	IR
opc | dst | src	3	6	06	R	IM

编程实例

```
假设： R1 = 12H，R2 = 03H，寄存器 01H = 21H，寄存器 02H = 03H，寄存器 03H = 0AH：

ADD  R1,R2       → R1 = 15H, R2 = 03H
ADD  R1,@R2      → R1 = 1CH, R2 = 03H
ADD  01H,02H     → 寄存器 01H = 24H, 寄存器 02H = 03H
ADD  01H,@02H    → 寄存器 01H = 2BH, 寄存器 02H = 03H
ADD  01H,#25H    → 寄存器 01H = 46H

在第一个例子中，目的工作寄存器 R1 内容为12H，源工作寄存器 R2 内容为 03H。语句“ADD R1,R2”
执行 03H+12H，结果为 15H，保存在寄存器 R1。
```

6.3.1.3 AND—逻辑与 (Logical AND)

AND dst, src

操作: dst ← dst AND src

源操作数与目的操作数执行逻辑与操作，结果保存在目的操作数。只有当两个操作数的对应位都为1时，结果的相应位才是 1；否则，该位为 0。源操作数不受影响。

标志位:

C: 不受影响
Z: 如果结果为 0，被置 1；否则，被清 0
S: 如果结果是负数，被置 1；否则，被清 0
V: 总是被清零
D: 不受影响
H: 不受影响

格式:

	字节数	时钟周期	指令代码	寻址模式	
			(Hex)	dst	src
opc | dst | src	2	4	52	r	r
		6	53	r	lr
opc | src | dst	3	6	54	R	R
		6	55	R	IR
opc | dst | src	3	6	56	R	IM

编程实例

```
假设：R1 = 12H，R2 = 03H，寄存器 01H = 21H，寄存器 02H = 03H，寄存器 03H = 0AH:

AND  R1,R2       → R1 = 02H, R2 = 03H
AND  R1,@R2      → R1 = 02H, R2 = 03H
AND  01H,02H     → 寄存器 01H = 01H, 寄存器 02H = 03H
AND  01H,@02H    → 寄存器 01H = 00H, 寄存器 02H = 03H
AND  01H,#25H    → 寄存器 01H = 21H

在第一个例子中，目的操作数 R1为 12H，源操作数 R2 为 03H，指令“AND R1,R2”对 03H 和12H
进行逻辑与操作，结果为 02H，保存在寄存器 R1。
```

6.3.1.4 BAND—位与 (Bit AND)

BAND dst, src.b

BAND dst.b, src

操作: dst(0) ← dst(0) AND src(b)
or
dst(b) ← dst(b) AND src(0)

源操作数(或者目的操作数)的特定位与目的操作数(或者源操作数)的最低位进行逻辑与操作，结果保存在目的操作数的特定位。目的操作数的其他位不受影响。源操作数不受影响。

标志位:

C: 不受影响
Z: 如果结果为 0，被置 1；否则，被清 0
S: 总是被清零
V: 不确定
D: 不受影响
H: 不受影响

格式:

			字节数	时钟周期	指令代码	寻址模式	
					(Hex)	dst	src
opc	dst | b | 0	src	3	6	67	r0	Rb
opc	src | b | 1	dst	3	6	67	Rb	r0

注释: 在 3 字节指令各式的第二个字节中，目的或者源操作数地址是 4 位，位地址是 3 位。

编程实例

假设：R1 = 07H 和寄存器 01H = 05H：

```
BAND R1,01H.1     → R1 = 06H, 寄存器 01H = 05H
BAND 01H.1,R1     → 寄存器 01H = 05H, R1 = 07H
```

在第一个例子中，源寄存器 01H 的内容是 05H，目的寄存器 R1 的内容是 07H，指令“BAND R1,01H.1”将源寄存器的位 1 和目的寄存器 R1 的位 0 进行逻辑与操作，结果为 06H，保存在寄存器 R1。

6.3.1.5 BCP—位比较 (Bit Compare)

BCP dst, src.b

操作: dst(0) – src(b)

源操作数的特定位与目的操作数的最低位做比较，如果相等，零标志位被置 1，否则被清 0。两个操作数都不受影响。

标志位:

C: 不受影响
Z: 如果结果为 0，被置 1；否则，被清 0
S: 总是被清零
V: 不确定
D: 不受影响
H: 不受影响

格式:

	字节数	时钟周期	指令代码 (Hex)	寻址模式 dst	寻址模式 src
opc | dst | b | 0 | src	3	6	17	r0	Rb

注释: 在 3 字节指令各式的第二个字节中，目的操作数地址是 4 位，位地址是 3 位。

编程实例

假设：R1 = 07H 和寄存器 01H = 01H：

```
BCP  R1,01H.1    → R1 = 07H, 寄存器 01H = 01H
```

如果目的寄存器 R1 内容为 07H，源寄存器 01H 的内容为 01H，指令“BCP R1,01H.1” 比较源寄存器(01H)的位 1 和目的寄存器 R1 的位 0。因为两个位是不相等的，所以标志位寄存器的 Z 标志位被清零。

6.3.1.6 BITC—位反 (Bit Complement)

BITC dst.b

操作: dst(b) ← NOT dst(b)

这个指令对目的操作数的特定位取反而不影响其他位。

标志位:
C: 不受影响
Z: 如果结果为 0，被置 1；否则，被清 0
S: 总是被清零
V: 不确定
D: 不受影响
H: 不受影响

格式:

	字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc | dst | b | 0	2	4	57	rb

注释: 在指令的第二个字节中，目的操作数地址是 4 位，位地址是 3 位。

编程实例

```
假设：R1 = 07H：

BITC R1.1          → R1 = 05H

工作寄存器 R1 内容为 07H，指令“BITC R1.1“对 R1 的位 1 取反，并将结果(05H)保存在 R1。
因为结果不是“0”，所以标志位寄存器的 Z 标志被清零。
```

6.3.1.7 BITR—位清零 (Bit Reset)

BITR dst.b

操作: dst(b) ← 0

BITR 指令将目的操作数的特定位清零，而不影响其它的位。

标志位: 没有任何标志位受影响

格式:

	字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc | dst | b | 0	2	4	77	rb

注释: 在指令的第二个字节中，目的操作数地址是 4 位，位地址是 3 位。

编程实例

```
假设：R1 = 07H：

BITR R1.1          → R1 = 05H

工作寄存器 R1 的内容是 07H，指令“BITR R1.1”对目标寄存器 R1 的位 1 清零，结果为 05H。
```

6.3.1.8 BITS—位置 1 (Bit Set)

BITS dst.b

操作: dst(b) ← 1

BITS 指令将目的操作数的特定位设置为 1，而不影响其它的位。

标志位: 没有标志位受影响

格式:

	字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc | dst | b | 1	2	4	77	rb

注释: 在指令的第二个字节中，目的操作数地址是 4 位，位地址是 3 位。

编程实例

```
假设：R1 = 07H：

BITS R1.3          → R1 = 0FH

工作寄存器 R1 的内容是 07H，指令“BITS R1.3”将目标寄存器 R1 的位 3 置为 1，结果为 0FH。
```

6.3.1.9 BOR—位或 (Bit OR)

BOR dst, src.b

BOR dst.b, src

操作: dst(0) ← dst(0) OR src(b)
或
dst(b) ← dst(b) OR src(0)

源操作数(或者目的操作数)的特定位与目的操作数(源操作数)的最低位进行逻辑或，
运算结果保存在目的操作数的特定位，目的操作数的其它位不受影响。源操作数不受影响。

标志位:
C: 不受影响
Z: 如果结果为 0，被置 1；否则，被清 0
S: 总是被清零
V: 不确定
D: 不受影响
H: 不受影响

格式:

			字节数	时钟周期	指令代码	寻址模式	
					(Hex)	dst	src
opc	dst | b | 0	src	3	6	07	r0	Rb
opc	src | b | 1	dst	3	6	07	Rb	r0

注释: 在 3 字节指令格式的第二个字节中，目的操作数地址是 4 位，位地址是 3 位。

编程实例

```
假设：R1 = 07H 和寄存器 01H = 03H：

BOR  R1, 01H.1   → R1 = 07H, 寄存器  01H = 03H
BOR  01H.2, R1   → 寄存器  01H = 07H, R1 = 07H

在第一个例子里面，目的寄存器 R1 的内容为 07H，源寄存器 01H 的内容是 03H，指令“BOR
R1,01H.1”将寄存器 01H 的位 1 与 R1 的位 0 进行逻辑或运算，结果(还是 07H )保存在 R1 中。

在第二个例子里面，目的寄存器 01H 的内容为 03H，源寄存器 R1 的内容是 07H，指令“BOR
01H.2,R2”将 01H 的位 2 与寄存器 R1 的位 0 进行逻辑或运算，结果 (07H) 保存在寄存器 01H 中。
```

6.3.1.10 BTJRF—位测试，若为假相对跳转 (Bit Test, Jump Relative on False)

BTJRF dst, src.b

操作: 如果目标操作数的位 b 为“0”，那么 PC ← PC + dst

测试源原操作数的特定位，如果为“0”，则将相对地址累加到程序计数器(PC)，并从新的 PC 地址开始执行程序；否则，执行 BTJRF 后面的指令。

标志位: 没有标志受影响。

格式:

(注释)

opc	src | b | 0	dst

字节数	时钟周期	指令代码	寻址模式	
		(Hex)	dst	src
3	10	37	RA	rb

注释: 在 3 字节指令各式的第二个字节中，目的操作数地址是 4 位，位地址是 3 位。

编程实例

假设：R1 = 07H：

```
BTJRF SKIP,R1.3          → PC 跳转到 SKIP 地址处
```

如果工作寄存器 R1 的内容是 07H，指令“BTJRF SKIP,R1.3”测试 R1 的位 3。
因其为“0”，相对地址将累加到 PC，然后 PC 跳转到 SKIP 地址处。(记住地址跳转的范围必须在 +127 至 -128之间)。

6.3.1.11 BTJRT—位测试，若为真，相对跳转 (Bit Test, Jump Relative on True)

BTJRT dst, src.b

操作: 如果目标操作数的位 b 为“1”，那么 PC ← PC + dst

测试源操作数的特定位，如果为“1”，则将相对地址累加到程序计数器(PC)，并从新的 PC 地址开始执行程序；如果为“0”，执行 BTJRT 后面的指令。

标志位: 没有标志位受影响。

格式:

(注释)

opc	src | b | 1	dst

字节数	时钟周期	指令代码 (Hex)	寻址模式 dst	寻址模式 src
3	10	37	RA	rb

注释: 在 3 字节指令各式的第二个字节中，目的操作数地址是 4 位，位地址是 3 位。

编程实例

```
假设：R1 = 07H：

BTJRT SKIP,R1.1

工作寄存器 R1 的内容是 07H，指令“BTJRT SKIP,R1.1”测试寄存器 R1 的位 1。
因该位为“1”，相对地址将被叠加到 PC，并跳转到新的 PC 地址 SKIP 处。(记住跳转范围必须在 +127至
-128 之间)。
```

6.3.1.12 BXOR — 位异或 (Bit XOR)

BXOR dst, src.b

BXOR dst.b, src

操作: dst(0) ← dst(0) XOR src(b)
或
dst(b) ← dst(b) XOR src(0)

源操作数(或者目的操作数)的特定位与目的操作数(源操作数)的最低位进行逻辑异或运算。结果保存在目的寄存器的特定位中。其他位不受影响。源操作数不受影响。

标志位:

C: 不受影响
Z: 如果结果为 0，被置 1；否则，被清 0
S: 总是被清零
V: 不确定
D: 不受影响
H: 不受影响

格式:

			字节数	时钟周期	指令代码	寻址模式	
					(Hex)	dst	src
opc	dst | b | 0	src	3	6	27	r0	Rb
opc	src | b | 1	dst	3	6	27	Rb	r0

注释: 在 3 字节指令各式的第二个字节中，目的操作数地址是 4 位，位地址是 3 位。

编程实例

```
假设：R1 = 07H (00000111B) 和寄存器 01H = 03H (00000011B)：

BXOR R1,01H.1     → R1 = 06H, 寄存器 01H = 03H
BXOR 01H.2,R1     → 寄存器 01H = 07H, R1 = 07H

在第一个例子中，目的寄存器 R1 的内容是 07H，源寄存器 01H 的内容是 03H，指令“BXOR
R1,01H.1”将源寄存器 01H 的位 1 和 R1 的位 0 进行逻辑异或，结果保存在 R1 的位 0，将 R1 的值从
07H 改写为  06H。源寄存器 01H 的值不受影响。
```

6.3.1.13 CALL—程序调用 (Call Procedure)

CALL dst

操作:

SP ← SP – 1
@SP ← PCL
SP ← SP –1
@SP ← PCH
PC ← dst

PC 的当前内容被压入堆栈，也就是紧跟 CALL 指令之后的指令地址。
然后，指定的目标地址被送给PC，指向子程序的第一条指令地址。在子程序末尾，
用返回指令 RET 返回到原来的程序流程，继续执行主程序。RET 指令的执行，将 PC
值从堆栈顶部弹出。

标志位: 没有标志位受影响

格式:

	字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc | dst	3	14	F6	DA
opc | dst	2	12	F4	IRR
opc | dst	2	14	D4	IA

编程实例

假设：R0 = 35H，R1 = 21H，PC = 1A47H，和SP = 0002H：

```
CALL 3521H      → SP = 0000H(存储器 0000H = 1AH, 0001H = 4AH, 4AH 是指令后面的地址)
CALL @RR0       → SP = 0000H (0000H = 1AH, 0001H = 49H)
CALL #40H       → SP = 0000H (0000H = 1AH, 0001H = 49H)
```

在第一个例子中，如果 PC 值为 1A47H，堆栈指针内容为 0002H，指令“CALL 3521H”将当前的PC
数值压入堆栈顶，堆栈指针现在指向 0000H，然后 PC 装载入 3521H，
从子程序的第一条指令地址开始顺序执行。

如果 PC 和 SP 的内容与第一个例子相同，指令“CALL @RP0”运行的结果基本相同，除了堆栈0001H
的内容是 49H(因为是 2 字节指令)，然后 PC 装载数值 3521H，并顺序执行指令。如果PC
和堆栈指针的内容与第一个例子相同，如果程序抵制 0040H 的内容为 35H，41H 的内容为21H，
指令“CALL #40H”产生和第二个例子相同的结果。

6.3.1.14 CCF—进位标志位取反 (Complement Carry Flag)

CCF

操作: C ← NOT C

进位标志 C 取反。如果C=“1”，进位标志变成 0；如果C=“0”，进位标志变成 1。

标志位: **C:** 取反

其他的标志位不受影响。

格式:

	字节数	时钟周期	指令代码 (Hex)
opc	1	4	EF

编程实例

假设：进位标志C=“0”：

```
CCF
```

如果进位标志 C=“0”，指令 CCF 对该标志为取反，在标志位寄存器中，该位从“0”变成“1”。

6.3.1.15 CLR—清零 (Clear)

CLR dst

操作: dst ← "0"

目的操作数被清零。

标志位: 没有标志位受影响。

格式:

opc	dst

字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
2	4	B0	R
	4	B1	IR

编程实例

假设：寄存器 00H = 4FH，寄存器 01H = 02H，和寄存器 02H = 5EH：

```
CLR  00H   → 寄存器 00H = 00H
CLR  @01H  → 寄存器 01H = 02H, 寄存器 02H = 00H
```

在寄存器寻址模式中，指令“CLR 00H”把目的寄存器 00H 的内容清零。在第二个例子中，指令“CLR @01H”使用间接寄存器寻址模式，把寄存器 02H 清零。

6.3.1.16 COM—取反 (complement)

COM dst

操作: dst ← NOT dst

对目标地址的内容取反，所有的“1”变成“0”，所有的“0”变成“1”。

标志位:
C: 不受影响
Z: 如果结果为 0，被置 1；否则，被清 0
S: 如果结果是负数，被置 1；否则，被清 0
V: 总是被清 0
D: 不受影响
H: 不受影响

格式:

	字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc | dst	2	4	60	R
		4	61	IR

编程实例

```
假设：R1 = 07H 和寄存器 07H = 0F1H：

COM  R1    → R1 = 0F8H
COM  @R1   → R1 = 07H, 寄存器 07H = 0EH

在第一个例子中，目的寄存器 R1 的内容是 07H，指令“COM
R1”对R1的所有位取反，所有的逻辑“1”变成逻辑“0”，所有的逻辑“0”变成逻辑“1”，结果为 0F8H。

在第二个例子中，应用间接寄存器寻址模式对目的寄存器 07H 的内容取反，结果为 0EH。
```

6.3.1.17 CP—比较 (Compare)

CP dst, src

操作: dst – src

源操作数和目的操作数作比较(相减)，根据结果设置适当的标志位。
源操作数和目的操作数均不受影响。

标志位:

C: 如果源操作数大于目的操作数，被置 1；否则，被清 0
Z: 如果结果为 0，被置 1；否则，被清 0
S: 如果结果是负数，被置 1；否则，被清 0
V: 总是被清零
D: 不受影响
H: 不受影响

格式:

	字节数	时钟周期	指令代码	寻址模式	
			(Hex)	dst	src
opc | dst | src	2	4	A2	r	r
		6	A3	r	lr
opc | src | dst	3	6	A4	R	R
		6	A5	R	IR
opc | dst | src	3	6	A6	R	IM

编程实例

```
1.    假设：R1 = 02H 和 R2 = 03H：

            CP     R1,R2 → 对 C 和 S 标志位置 1

目的寄存器 R1 的内容为 02H，源寄存器 R2 的内容为 03H，指令“CP R1,R2”把 R1 减去
R2，因为产生了借位，结果是负数，所以 C 和 S 被置 1。

2.    假设：  R1 = 05H 和 R2 = 0AH：
            CP     R1,R2
            JP     UGE,SKIP
            INC    R1
SKIP LD     R3,R1

在这个例子中，目的寄存器 R1 的内容是 05H，小于源寄存器 R2 的内容 0AH。指令“CP R1,R2”使得 C
=“1”，JP 指令没有跳转到 SKIP 处。当执行完指令“LD R3,R1”，寄存器 R3的内容为 06H。
```

6.3.1.18 CPIJE—比较，增加一，若相等跳转 (Compare, Increment, and Jump on Equal)

CPIJE dst, src, RA

操作: If dst – src = "0", PC ← PC + RA

Ir ← Ir + 1

源操作数与目的操作数作比较 (相减)。若结果为“0”，则相对地址被叠加到 PC 上，从 PC 指向的新地址开始执行程序。否则，继续执行 CPIJE 后面的指令。无论哪种情况下，在执行下一条指令前，源指针增加 1。

标志位: 没有标志位受影响。

格式:

			字节数	时钟周期	指令代码 (Hex)	寻址模式 dst	寻址模式 src
opc	src | dst	RA	3	12	C2	r	Ir

注释: 执行时间是 18 个时钟周期(跳转)或者 16 个时钟周期(无跳转)。

编程实例

```
假设：R1 = 02H，R2 = 03H，和寄存器 03H = 02H：

CPIJE R1,@R2,SKIP              → R2 = 04H, PC 跳转到地址 SKIP 处

在这个例子中，工作寄存器 R1 的内容是 02H，工作寄存器 R2 的内容是 03H，寄存器 03H 的内容是
02H，指令“CPIJE R1,@R2,SKIP”比较@ R2 的内容 02H 和 02H，
因为比较的结果是相等的，跳转到新的 PC 地址 SKIP 处。源寄存器 R2 加 1 变成 04H。

记住 CPIJE 指令跳转的地址范围须介于 +127 至 -128 之间。
```

6.3.1.19 CPIJNE—比较，增加一，不等跳转 (Compare, Increment, Jump on Non-Equal)

CPIJNE dst, src, RA

操作: If dst – src "0", PC ← PC + RA

Ir ← Ir + 1

源操作数与目的操作数作比较(相减)，如果结果不为“0”，则将相对地址叠加到 PC 上，从 PC 指向的新地址开始执行程序。否则，继续执行 CPIJE 后面的指令。无论在哪种情况下，在执行下一条指令前，源指针增加 1。

标志位: 没有标志位受影响。

格式:

	字节数	时钟周期	指令代码	寻址模式	
			(Hex)	dst	src
opc | src | dst | RA	3	12	D2	r	Ir

注释: 执行时间是 18 个时钟周期(跳转)或者 16 个时钟周期(无跳转)。

编程实例

假设：R1 = 02H，R2 = 03H，和寄存器 03H = 04H：

```
CPIJNE R1,@R2,SKIP      → R2 = 04H, PC 跳转到地址 SKIP
```

在这个例子中，工作寄存器 R1 的内容是 02H，工作寄存器 R2 的内容是 03H，寄存器 03H 的内容是 04H，指令“CPIJE R1,@R2,SKIP”比较 @R2 的内容 04H 和 02H，因为比较的结果是不相等的，跳转到新的 PC 地址 SKIP 处。源寄存器 R2 加 1 变成 04H。

记住 CPIJNE 指令跳转的地址范围须介于 +127 至 -128 之间。

6.3.1.20 DA — 十进制调整 (Decimal Adjust)

DA dst

操作: dst ← DA dst

在加法或者减法运算后，将结果调整为 2 个 4 位 BCD 码。对于加法(ADD, ADC)或者减法(SUB, SBC)下面的表格指明了操作如何进行。

指令	DA前的进位标志	位7-4 (十六进制)	DA前H标志	位3-0 (十六进制)	要增加的数字	DA后的进位标志
	0	0–9	0	0–9	00	0
	0	0–8	0	A–F	06	0
	0	0–9	1	0–3	06	0
ADD	0	A–F	0	0–9	60	1
ADC	0	9–F	0	A–F	66	1
	0	A–F	1	0–3	66	1
	1	0–2	0	0–9	60	1
	1	0–2	0	A–F	66	1
	1	0–3	1	0–3	66	1
	0	0–9	0	0–9	00 = – 00	0
SUB	0	0–8	1	6–F	FA = – 06	0
SBC	1	7–F	0	0–9	A0 = – 60	1
	1	6–F	1	6–F	9A = – 66	1

标志位:

C: 如果发生进位，被置 1；否则，被清 0
Z: 如果结果为 0，被置 1；否则，被清 0
S: 如果结果是负数，被置 1；否则，被清 0
V: 没有定义
D: 不受影响
H: 不受影响

格式:

	字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc dst	2	4	40	R
		4	41	IR

6.3.1.21 DA—十进制调整 (Decimal Adjust)

DA (续)

编程实例

假设：工作寄存器R0的内容是15(BCD)，工作寄存器 R1 的内容是 27(BCD)，地址 27H 的内容是 46(BCD)：

```
ADD   R1,R0        ; C ← "0", H ← "0", Bits 4–7 = 3, bits 0–3 = C, R1 ← 3CH
DA    R1           ; R1 ← 3CH + 06
```

如果用 BCD 数值 15 和 27 进行加法运算，结果是 27。然而结果是不对的，当加法的时候，使用的是标准的二进制运算：

```
          0 0 0 1    0 1 0 1   15
     +    0 0 1 0    0 1 1 1   27
          0 0 1 1    1 1 0 0 = 3CH
```

DA 指令调整了运算结果，进而得到正确的结果：

```
          0 0 1 1    1 1 0 0
     +    0 0 0 0    0 1 1 0
          0 1 0 0    0 0 1 0 = 42
```

假设同样的数值，指令：

```
SUB   27H,R0       ; C ← "0", H ← "0", Bits 4–7 = 3, bits 0–3 = 1
DA    @R1          ; @R1 ← 31–0
```

执行的结果为 31(BCD)，保存在地址 27H(@R1)。

6.3.1.22 DEC—字节减 1 (Decrement)

DEC dst

操作: dst ← dst – 1

目的操作数的内容减 1

标志位:
C: 不受影响
Z: 如果结果为 0，被置 1；否则，被清 0
S: 如果结果是负数，被置 1；否则，被清 0
V: 如果结果溢出，被置 1；否则，被清 0
D: 不受影响
H: 不受影响

格式:

	字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc | dst	2	4	00	R
		4	01	IR

编程实例

假设：R1 = 03H 和寄存器 03H = 10H：

```
DEC  R1    → R1 = 02H
DEC  @R1   → 寄存器 03H = 0FH
```

在第一个例子中，如果工作寄存器 R1 的内容是 03H，值令“DEC R1”将该 16 进制数减 1，结果为 02H。在第二个例子中，“DEC @R1”将数值 10H 减 1，得到 0FH，保存在地址 03H。

6.3.1.23 DECW—字减 1 (Decrement Word)

DECW dst

操作: dst ← dst – 1

目的地址(须为偶数)的内容减 1，把操作数视为 16 位数据。

标志位:
- **C:** 不受影响
- **Z:** 如果结果为 0，被置 1；否则，被清 0
- **S:** 如果结果是负数，被置 1；否则，被清 0
- **V:** 如果有溢出，被置 1；否则，被清 0
- **D:** 不受影响
- **H:** 不受影响

格式:

	字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc | dst	2	8	80	RR
		8	81	IR

编程实例

假设：R0 = 12H，R1 = 34H，R2 = 30H，寄存器 30H = 0FH，和寄存器 31H = 21H：

```
DECW RR0    →    R0 = 12H, R1 = 33H
DECW @R2    →    寄存器 30H = 0FH, 寄存器 31H = 20H
```

在第一个例子中，目的寄存器 R0 内容是 12H，寄存器 R1 内容 34H，指令“DECW RR0”把R1，R0 当作 16 位数，减 1 以后，得到的结果为 33H。

注释: 当 DECW 指令与 Z 标志一起使用时，可能导致系统错误。为避免这个问题，推荐按照如下方法来使用 DECW 指令：

```
LOOP:   DECW    RR0
        LD      R2,R1
        OR      R2,R0
        JR      NZ,LOOP
```

6.3.1.24 DI—屏蔽全局中断 (Disable Interrupts)

DI

操作: SYM (0) ← 0

系统模式控制寄存器的位 0，SYM.0 被清零，将禁止全局中断。中断请求仍然会置起相应的中断悬挂标志，但 CPU 不会响应中断服务程序，因为中断处理被屏蔽了。

标志位: 没有标志位受影响。

格式:

	字节数	时钟周期	指令代码 (Hex)
opc	1	4	8F

编程实例

假设：SYM = 01H：

```
DI
```

如果 SYM 寄存器的值是 01H，指令 DI 将清零 SYM.0，屏蔽所有的中断处理。

在改变 IMR，中断标志位，中断源控制寄存器之前，确保 DI 状态。

6.3.1.25 DIV—无符号除法 (Unsigned Divide)

DIV dst, src

操作: dst ÷ src

dst (UPPER) ← REMAINDER
dst (LOWER) ← QUOTIENT

目的操作数(16 位)除以源操作数(8 位)，商(8 位)保存在目的操作数的低字节，余数(8位)则保存在目的操作数的高字节。如果商 ≥ 2^8，保存在目的操作数高低字节的商和余数是不正确的。所有的操作数都是无符号整数。

标志位:

C: 如果V标志被设置并且商在 2^8 和 2^9 –1范围之间，被置 1；否则，被清 0
Z: 如果除数或者商 = “0”，被置 1；否则，被清 0
S: 如果商的最高位 = “1”，被置 1；否则，被清 0
V: 如果商 ≥ 2^8 或者除数 = “0”，被置 1；否则，被清 0
D: 不受影响
H: 不受影响

格式:

opc	src	dst

字节数	时钟周期	指令代码 (Hex)	寻址模式 dst	寻址模式 src
3	26/10(注释)	94	RR	R
	26/10(注释)	95	RR	IR
	26/10(注释)	96	RR	IM

注释: 如果除数为 0，运行时间需要 10 个时钟周期；其他情况下，执行时间为 26 个时钟周期。

编程实例

假设：R0 = 10H，R1 = 03H，R2 = 40H，寄存器 40H = 80H：

```
DIV   RR0,R2       → R0 = 03H, R1 = 40H
DIV   RR0,@R2      → R0 = 03H, R1 = 20H
DIV   RR0,#20H     → R0 = 03H, R1 = 80H
```

在第一个例子中，目的寄存器 RR0 的内容是 10H(R0)和 03H(R1)，寄存器 R2 的内容是40H。 指令“DIV RR0,R2”将 16 位 RR0 被除数除以 8 位除数 R2，除法运算后，R0 内容为03H，R1 的内容为 40H。8 位余数保存在目的寄存器 RR0 的上半部分(R0)，商则保存在下半部分(R1)。

6.3.1.26 DJNZ—减 1，如果非零，跳转 (Decrement and Jump if Non-Zero)

DJNZ r, dst

操作: r ← r – 1
If r ≠ 0, PC ← PC + dst

作为计数器的工作寄存器值首先减 1，如果结果不为“0”，相对地址被累加到 PC，然后程序跳转到新的地址执行。相对地址的范围是 +127 至 –128，PC 的初始值是紧跟在 DJNZ 指令后面的指令地址。

注释: 使用 DJNZ 指令的时候，被用来做计数器的工作寄存器必须通过 SRP,SRP0,SRP1 指令设置为 0C0H~0C1H 中的一个。

标志位: 没有标志位受影响。

格式:

	字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
r | opc | dst	2	8 (jump taken) 8 (no jump)	rA r = 0 to F	RA

编程实例

假设：R1 = 02H 和 LOOP 是相对地址的地址标志符：

```
LOOP:       SRP    #0C0H
            DJNZ   R1,LOOP
```

指令 DJNZ 经常被用在循环程序中。大多数情况下，用地址标号而非数字相对地址来做目的操作数。在这个例子中，工作寄存器 R1 内容是 02H，LOOP 是相对地址的标号。

指令“DJNZ R1, LOOP”首先将寄存器 R1 减 1，得到结果 01H，因为减 1 后 R1 的内容非“0”，程序将跳转到标号 LOOP 指向的地址执行。

6.3.1.27 EI—使能全局中断 (Enable Interrupts)

EI

操作: SYM (0) ← 1

EI 指令对系统模式寄存器(SYM)的最低位 SYM.0 置 1，这将允许响应中断服务程序(假定该中断具有最高优先级)。如果中断处理被屏蔽(通过执行 DI)时，又发生中断，可通过执行EI指令，来执行中断服务程序。

标志位: 没有标志位受影响。

格式:

	字节数	时钟周期	指令代码 (Hex)
Opc	1	4	9F

编程实例

```
假设：SYM = 00H：

EI

如果 SYM 寄存器的内容是 00H，也就是说，全局中断被屏蔽，执行指令“EI”将把SYM寄存器设置为
01H，使能所有中断(SYM.0 是全局中断的使能位)。
```

6.3.1.28 ENTER—进入 (Enter)

ENTER

操作:

SP ← SP – 2
@SP ← IP
IP ← PC
PC ← @IP
IP ← IP + 2

这条指令在执行线性代码语言时非常有用。指令指针的内容压入堆栈，然后把程序计数器(PC)的值写入指令指针，程序存储器中指令指针所指向的字数据载入 PC，并且指令指针的值增加 2。

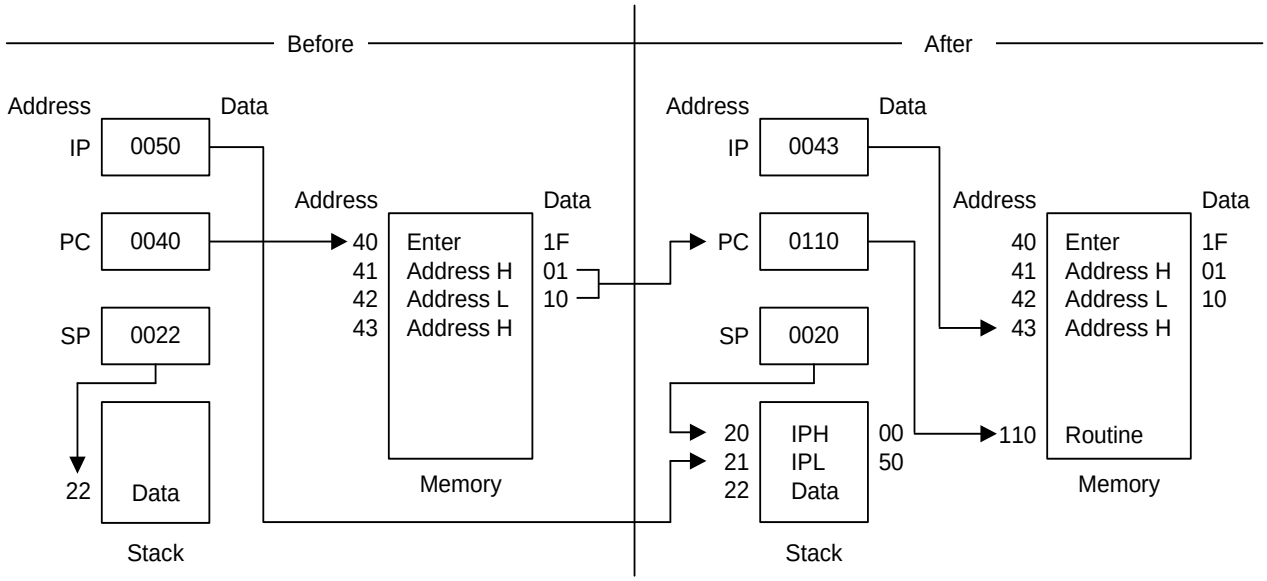
标志位: 没有标志位受影响。

格式:

	字节数	时钟周期	指令代码 (Hex)
opc	1	14	1F

编程实例

下图给出了一个如何使用 ENTER 指令的例子。



6.3.1.29 EXIT—退出 (Exit)

EXIT

操作:

IP ← @SP
SP ← SP + 2
PC ← @IP
IP ← IP + 2

这条指令在执行线性代码语言时非常有用。被压入堆栈的值弹出并载入指令指针，程序存储器中被指令指针指向的字数据载入程序计数器，并且指令指针的值增加 2。

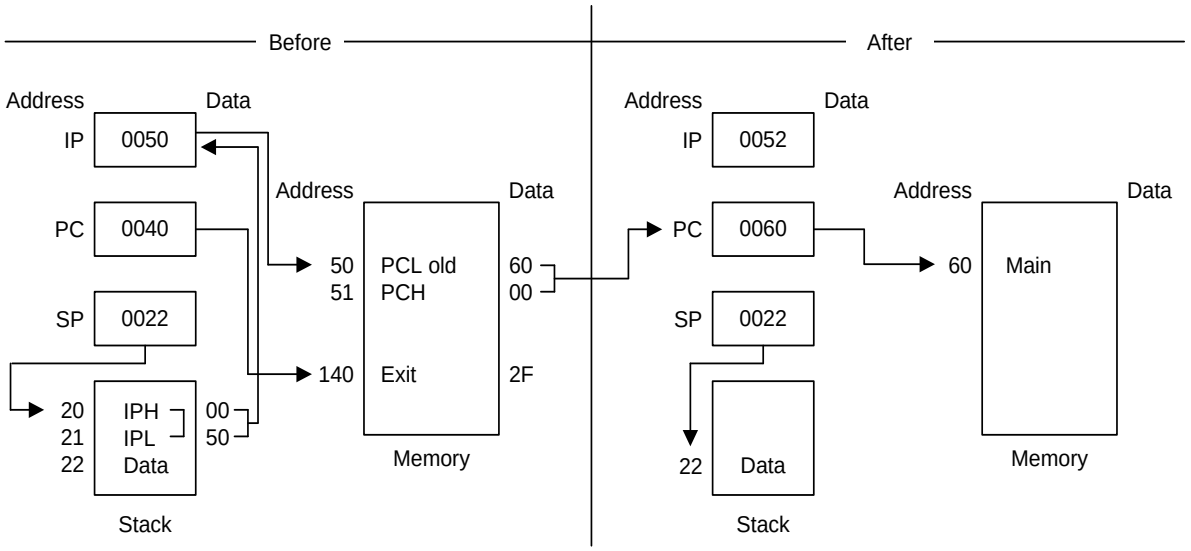
标志位: 没有标志位受影响。

格式:

	字节数	时钟周期	指令代码 (Hex)
opc	1	14 (内部堆栈) 16 (外部堆栈)	2F

编程实例

下图给出了一个如何使用 EXIT 指令的例子。



6.3.1.30 IDLE—空闲指令 (Idle Operation)

IDLE

操作: (见描述)

IDLE 指令将停止 CPU 时钟但允许系统时钟继续工作。IDLE 模式可以被中断请求(IRQ)或者是外部复位操作唤醒。在应用程序中，IDLE 指令后必须立即执行至少 3 个 NOP 指令。这是为了保证在下一条指令执行之前，系统时钟有足够的时间间隔来稳定时钟信号。如果在 IDLE 指令后没有 3 个或更多个的 NOP 指令，内部总线的悬浮状态将导致漏电流的产生。

标志位: 没有标志位受影响。

格式:

	字节数	时钟周期	指令代码	寻址模式	
			(Hex)	dst	src
opc	1	4	6F	–	–

编程实例

指令

```
IDLE ; 停止 CPU 时钟但不停系统时钟
NOP
NOP
NOP
```

6.3.1.31 INC—加 1 (Increment)

INC dst

操作: dst ← dst + 1

目标操作数的内容增加1

标志位:
- **C:** 没有影响
- **Z:** 如果结果为“0”则置 1；否则清零
- **S:** 如果结果为负数则置 1；否则清零
- **V:** 如果发生溢出则置 1；否则清零
- **D:** 没有影响
- **H:** 没有影响

格式:

	字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
dst | opc	1	4	rE r = 0 到 F	r
opc | dst	2	4	20	R
		4	21	IR

编程实例

假如：R0 = 1BH，寄存器 00H = 0CH，寄存器 1BH = 0FH：

```
INC  R0    → R0 = 1CH
INC  00H   → 寄存器 00H = 0DH
INC  @R0   → R0 = 1BH, 寄存器 01H = 10H
```

在第一个例子中，如果目标工作寄存器R0的值为1BH，那么语句“INC R0”将 R0 的值变为1CH。

第二个例子演示了 INC 指令对寄存器 00H 产生的效果，假定寄存器的值为 0CH。

第三个例子中，INC 指令用在间接寄存器(IR)寻址模式中，将寄存器 1BH 的值由 0FH 变为10H。

6.3.1.32 INCW—字加 1 (Increment Word)

INCW dst

操作: dst ← dst + 1

目标操作数(必须是偶地址)中的一个字节和下一地址中的字节数据合在一起作为一个 16 位数据，整个 16 位的数据增加 1。

标志位:

C: 没有影响
Z: 如果结果为“0”则置 1；否则清零
S: 如果结果为负数则置 1；否则清零
V: 如果发生溢出则置 1；否则清零
D: 没有影响
H: 没有影响

格式:

	字节数	时钟周期	指令代码	寻址模式
			(Hex)	**dst**
opc | dst	2	8	A0	RR
		8	A1	IR

编程实例

假如：R0 = 1AH，R1 = 02H，寄存器 02H = 0FH，寄存器 03H = 0FFH：

```
INCW RR0    → R0 = 1AH, R1 = 03H
INCW @R1    → 寄存器 02H = 10H, 寄存器 03H = 00H
```

在第一个例子中，工作寄存器对 RR0 的内容是 1AH(R0) 和 02H(R1)。语句“INCW RR0”将16位的目标数据加 1，使寄存器 R1 的值变为 03H。第二个例子中，语句“INCW @R1”用间接寄存器(IR)寻址模式将寄存器 03H 的值从 0FFH 变为 00H，寄存器 02H 的值从 0FH 变为 10H。

注释: 如果 Zero(Z)标志位(标志位.6)与 INCW 指令一起使用，可能会发生冲突而导致系统错误。为了避免这个问题，建议按下面的方法使用 INCW 指令：

```
LOOP:   INCW    RR0
        LD      R2,R1
        OR      R2,R0
        JR      NZ,LOOP
```

6.3.1.33 IRET—中断返回 (Interrupt Return)

IRET IRET (正常) IRET (快速)

操作:

IRET (正常)	IRET (快速)
标志位 ← @SP	PC ↔ IP
SP ← SP + 1	标志位 ← 标志位'
PC ← @SP	FIS ← 0
SP ← SP + 2	
SYM(0) ← 1	

该指令用在中断服务程序的末尾。该指令将恢复标志寄存器和程序计数器的值，同时重新使能全局中断。只有在快速中断状态位(FIS，标志位寄存器位 1，0D5H)被清零时(=“0”)，才执行“正常IRET”。快速中断产生时，对于在中断服务程序开始时被置 1 的 FIS 位，IRET 会将其清零。

标志位: 所有标志位恢复到初始状态(即中断发生以前的状态)。

格式:

IRET (正常)

Opc

字节数	时钟周期	指令代码 (Hex)
1	10 (内部堆栈) 12 (外部堆栈)	BF

IRET (快速)

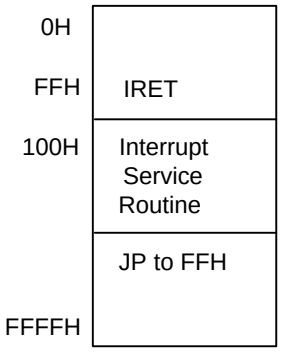
opc

字节数	时钟周期	指令代码 (Hex)
1	6	BF

编程实例

在下图中，中断使能之前，在主程序中将指令指针初始化为 100H。中断发生时，程序计数器和指令指针的内容相交换，这使得 PC 跳转到地址 100H
而IP则保存返回地址。中断服务程序的最后一条指令通常是跳转到地址 FFH 处的 IRET。

这将使指令指针“重新”被赋值为 100H，且程序计数器跳回到主程序。现在，下一个中断可以发生了，IP 寄存器的值仍然为 100H。



注释: 上面快速中断的例子中，如果最后的指令不是跳转到 IRET，那么必须注意最后两条指令的顺序。在 IRET 指令之后，不可紧跟用于中断状态清除的指令(例如 IPR 寄存器清 0)。

6.3.1.34 JP—跳转 (Jump)

JP cc,dst (条件跳转)

JP dst (无条件跳转)

操作: 如果 cc 为真，PC ← dst

如果转移条件为真，那么条件跳转指令把系统控制交给目标地址；否则执行紧跟 JP 指令之后的指令。无条件跳转指令只是简单的用目标地址替换 PC 的内容，然后程序控制交给由 PC 指定的语句。

标志位: 没有标志位受影响。

格式: (1)

		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
(2) cc | opc	dst	3	8	ccD cc = 0 to F	DA
opc	dst	2	8	30	IRR

注释:

1. 3 字节格式用于条件跳转，2 字节格式用于无条件跳转。
2. 在 3 字节指令格式(条件跳转)的第一字节中，条件码和指令代码各占 4 位。

编程实例:

假如：进位标志 (C) = “1”，寄存器 00 = 01H，寄存器 01 = 20H：

```
JP   C,LABEL_W          → LABEL_W = 1000H, PC = 1000H
JP   @00H               → PC = 0120H
```

第一个例子是条件跳转 JP。假定进位标志为“1”，指令“JP C,LABEL_W”将 PC 的值替换为1000H 并且跳转到该地址。如果标志位没有被置 1，那么程序将立即转到紧跟 JP 后的那条指令。

第二个例子为无条件跳转 JP。指令“JP @00”将 PC 的内容替换为寄存器对 00H 和 01H 的值，亦即 0120H。

6.3.1.35 JR—相对跳转指令 (Jump Relative)

JR cc, dst

操作: 如果 cc 为真，PC ← PC + dst

如果转移条件为真，那么程序计数器值加上相对地址，并把程序控制转到该地址处的指令；否则执行紧跟 JR 指令后的那条指令(见本章：转移条件码列表)。

相对地址的范围是 –128 ~ +127，并且程序计数器的初值被认为是紧跟JR指令之后的第一条指令地址。

标志位: 没有标志位受影响。

格式:

	字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
(1) cc | opc | dst	2	6	ccB cc = 0 到 F	RA

注释: 2 字节指令格式中的第一个字节中，条件码和指令代码各占 4 位。

编程实例

假如：进位标志位 = “1” 并且 LABEL_X = 1FF7H：

```
JR    C,LABEL_X    → PC = 1FF7H
```

如果进位标志位为“1”(也就是，转移条件为真)，指令“JR C,LABEL_X”会将程序控制交给当前 PC 指向的地址；否则，执行紧跟 JR 后的程序指令。

6.3.1.36 LD—传送数据 (Load)

LD dst, src

操作: dst ← src

源操作数的内容将赋给目标操作数。源操作数的内容不受影响。

标志位: 没有标志位受影响。

格式:

			字节数	时钟周期	指令代码 (Hex)	寻址模式 dst	寻址模式 src
dst | opc	src		2	4	rC	r	IM
				4	r8	r	R
src | opc	dst		2	4	r9	R	r
					r = 0 到 F		
opc	dst | src		2	4	C7	r	Ir
				4	D7	Ir	r
opc	src	dst	3	6	E4	R	R
				6	E5	R	IR
opc	dst	src	3	6	E6	R	IM
				6	D6	IR	IM
opc	src	dst	3	6	F5	IR	R
opc	dst | src	x	3	6	87	r	x [r]
opc	src | dst	x	3	6	97	x [r]	r

6.3.1.37 LD—传送数据 (Load)

LD (续)

编程实例

```
假如：R0 = 01H，R1 = 0AH，寄存器 00H = 01H，寄存器 01H = 20H，
寄存器 02H  = 02H，LOOP = 30H，寄存器 3AH = 0FFH：

LD   R0,#10H          → R0 = 10H
LD   R0,01H           → R0 = 20H, 寄存器 01H = 20H
LD   01H,R0           → 寄存器 01H = 01H, R0 = 01H
LD   R1,@R0           → R1 = 20H, R0 = 01H
LD   @R0,R1           → R0 = 01H, R1 = 0AH, 寄存器 01H = 0AH
LD   00H,01H          → 寄存器 00H = 20H, 寄存器 01H = 20H
LD   02H,@00H         → 寄存器 02H = 20H, 寄存器 00H = 01H
LD   00H,#0AH         → 寄存器 00H = 0AH
LD   @00H,#10H        → 寄存器 00H = 01H, 寄存器 01H = 10H
LD   @00H,02H         → 寄存器 00H = 01H, 寄存器 01H = 02, 寄存器 02H = 02H
LD   R0,#LOOP[R1]     → R0 = 0FFH, R1 = 0AH
LD   #LOOP[R0],R1     → 寄存器 31H = 0AH, R0 = 01H, R1 = 0AH
```

6.3.1.38 LDB—传送位数据 (Load Bit)

LDB dst, src.b

LDB dst.b, src

操作: dst(0) ← src(b)
或
dst(b) ← src(0)

源操作数的指定位载入目标操作数的最低位，或者源操作数的最低位载入目标操作数的指定位。目标操作数的其它位都不受影响，源操作数也不受影响。

标志位: 没有标志位受影响。

格式:

	字节数	时钟周期	指令代码	寻址模式	
			(Hex)	dst	src
opc | dst | b | 0 | src	3	6	47	r0	Rb
opc | src | b | 1 | dst	3	6	47	Rb	r0

注释: 指令格式的第二个字节中，目标(或源)地址占 4 位，位地址“b”占 3 位，LSB 地址值占 1 位。

编程实例

```
假如：R0 = 06H ，通用寄存器 00H = 05H：

LDB  R0,00H.2    → R0 = 07H, 寄存器 00H = 05H
LDB  00H.0,R0    → R0 = 06H, 寄存器 00H = 04H

第一个例子中，目标工作寄存器 R0 的值为 06H，源寄存器 00H 的值为 05H。
指令“ R0,00h.2”将寄存器 00H 中的第二位(bit 2)载入寄存器 R0 的最低位，R0 的值变为 07H。

第二个例子中，目标寄存器是 00H。指令“LD 00H.0,R0”将工作寄存器 R0 的最低位载入目标寄存器 00H
的指定位(bit 0)，寄存器 00H 的值变为 04H。
```

6.3.1.39 LDC/LDE—传送程序/外部数据存储器数据 (Load Memory)

LDC/LDE dst, src

操作: dst ← src

这条指令从程序或外部数据存储器中装载一个字节数据到工作寄存器，或者相反。
源操作数不受影响。指令 LDC 用作程序存储器，LDE 用作外部数据存储器。
对于程序存储器，编译器将“Irr”或“rr”的值编译成偶地址，而对于外部数据存储器则编译成奇地址。

标志位: 没有标志位受影响。

格式:

					字节数	时钟周期	指令代码	寻址模式	
							(Hex)	dst	src
1.	opc	dst | src			2	10	C3	r	Irr
2.	opc	src | dst			2	10	D3	Irr	r
3.	opc	dst | src	XS		3	12	E7	r	XS [rr]
4.	opc	src | dst	XS		3	12	F7	XS [rr]	r
5.	opc	dst | src	XL_L	XL_H	4	14	A7	r	XL [rr]
6.	opc	src | dst	XL_L	XL_H	4	14	B7	XL [rr]	r
7.	opc	dst | 0000	DA_L	DA_H	4	14	A7	r	DA
8.	opc	src | 0000	DA_L	DA_H	4	14	B7	DA	r
9.	opc	dst | 0001	DA_L	DA_H	4	14	A7	r	DA
10.	opc	src | 0001	DA_L	DA_H	4	14	B7	DA	r

注释:
1. 格式 5 和 6 的源操作数 [src] 或工作寄存器对 [rr] 不能使用工作寄存器对 0–1。
2. 格式 3 和 4 的目标地址“XS[rr]”和源地址“XS[rr]”均为一个字节。
3. 格式 3 和 4 的目标地址“XL[rr]”和源地址“XL[rr]”均为两个字节。
4. 格式 7 和 8 的 DA 和 r 源操作数值用于访问程序存储器，格式 9 和 10 则用于访问外部数据存储器。
5. LDE 指令可用于读/写 64K 字节的外部数据存储器。

6.3.1.40 LDC/LDE—传送程序/外部数据存储器数据 (Load Memory)

LDC/LDE (续)

编程实例

```
假如：R0 = 11H, R1 = 34H, R2 = 01H, R3 = 04H; 程序存储器空间中
0103H = 4FH, 0104H = 1A, 0105H = 6DH, 1104H = 88H. 外部数据存储空间中
0103H = 5FH, 0104H = 2AH, 0105H = 7DH, 1104H = 98H:

LDC  R0,@RR2                  ; R0 ← 程序存储器地址 0104H 的内容
                              ; R0 = 1AH, R2 = 01H, R3 = 04H
LDE  R0,@RR2                  ; R0 ← 外部数据存储器地址 0104H 的内容
                              ; R0 = 2AH, R2 = 01H, R3 = 04H
LDC (注释)@RR2,R0             ; 11H (R0 的内容) 载入程序存储器地址 0104H (RR2)
                              ; 工作寄存器 R0, R2, R3 → 没有变化
LDE  @RR2,R0                  ; 11H (R0 的内容) 载入外部数据存储器地址 0104H (RR2)
                              ; 工作寄存器 R0, R2, R3 → 没有变化
LDC  R0,#01H[RR2]             ; R0  ← 程序存储器地址 0105H 的内容
                              ; (01H + RR2),
                              ; R0 = 6DH, R2 = 01H, R3 = 04H
LDE  R0,#01H[RR2]             ; R0 ← 外部数据存储器地址 0105H 的内容
                              ; (01H + RR2), R0 = 7DH, R2 = 01H, R3 = 04H
LDC (注释) #01H[RR2],R0       ; 11H (R0 的内容) 载入程序存储器地址
                              ; 0105H (01H + 0104H)
LDE  #01H[RR2],R0             ; 11H (R0 的内容) 载入外部数据存储器地址
                              ; 0105H (01H + 0104H)
LDC  R0,#1000H[RR2]           ; R0 ← 程序存储器地址 1104H 的内容
                              ; (1000H + 0104H), R0 = 88H, R2 = 01H, R3 = 04H
LDE  R0,#1000H[RR2]           ; R0 ← 外部数据存储器地址 1104H 的内容
                              ; (1000H + 0104H), R0 = 98H, R2 = 01H, R3 = 04H
LDC  R0,1104H                 ; R0 ← 程序存储器地址 1104H 的内容, R0  =  88H
LDE  R0,1104H                 ; R0 ← 外部数据存储器地址 1104H 的内容
                              ; R0 = 98H
LDC (注释)1105H,R0            ; 11H (R0 的内容) 载入程序存储器地址
                              ; 1105H, (1105H) ← 11H
LDE  1105H,R0                 ; 11H (R0 的内容) 载入外部数据存储器地址
                              ; 1105H, (1105H) ← 11H
```

注释: 掩膜 ROM 类型的器件不支持 LDC/LDE 指令。

6.3.1.41 LDCD/LDED—传送数据之后地址减 1 (Load Memory and Decrement)

LDCD/LDED dst, src

操作: dst ← src
rr ← rr – 1

该指令用于用户栈或在程序/数据存储器与寄存器卷之间批量传送数据。
存储器的地址由工作寄存器对指定。源地址的内容载入目标地址，然后存储器地址自动减1，源操作数的内容不变。

LDCD 对应程序存储器，而 LDED 对应外部数据存储器。对于程序存储器，编译器将“Irr”或“rr”的值编译成偶地址，对于数据存储器则编译成奇地址。

标志位: 没有标志位受影响。

格式:

	字节数	时钟周期	指令代码	寻址模式	
			(Hex)	dst	src
opc | dst | src	2	10	E2	r	Irr

编程实例

```
假如：R6 = 10H，R7 = 33H，R8 = 12H，程序存储器地址 1033H = 0CDH，外部数据存储地址 1033H = 0DDH：

LDCD R8,@RR6      ; 0CDH (程序存储器 1033H 的内容) 载入R8
                  ; RR6 减1
                  ; R8 = 0CDH, R6 = 10H, R7 = 32H (RR6 ← RR6 – 1)

LDED R8,@RR6      ; 0DDH (数据存储器地址 1033H 的内容) 载入R8
                  ; RR6 减1 (RR6 ← RR6 – 1)
                  ; R8 = 0DDH, R6 = 10H, R7 = 32H
```

6.3.1.42 LDCI/LDEI—传送数据后地址加 1 (Load Memory and Increment)

LDCI/LDEI dst, src

操作: dst ← src
rr ← rr + 1

该指令用于用户栈或在程序/数据存储器与寄存器卷之间批量传送数据。
存储器的地址由工作寄存器对指定。源地址的内容载入目标地址，然后存储器地址自动加 1，源操作数的内容不变。

LDCI 对应程序存储器，而 LDEI 对应外部数据存储器。对于程序存储器，编译器将“Irr”或“rr”的值编译成偶地址，对于数据存储器则编译成奇地址。

标志位: 没有标志位受影响。

格式:

	字节数	时钟周期	指令代码	寻址模式	
			(Hex)	dst	src
opc | dst | src	2	10	E3	r	Irr

编程实例

```
假如：R6 = 10H，R7 = 33H，R8 = 12H，程序存储器地址 1033H = 0CDH，1034H =
0C5H；外部数据存储器地址 1033H = 0DDH，1034H = 0D5H：

LDCI R8,@RR6      ; 0CDH (程序存储器地址 1033H 的内容) 载入R8
                  ; RR6 加 1 (RR6 ← RR6 + 1)
                  ; R8 = 0CDH, R6 = 10H, R7 = 34H

LDEI R8,@RR6      ; 0DDH (数据存储器地址 1033H 的内容) 载入R8
                  ; RR6 加 1 (RR6 ← RR6 + 1)
                  ; R8 = 0DDH, R6 = 10H, R7 = 34H
```

6.3.1.43 LDCPD/LDEPD—传送数据前地址减 1 (Load Memory with Pre-Decrement)

LDCPD/
LDEPD dst, src

操作: rr ← rr – 1
dst ← src

该指令用于用户栈或在程序/数据存储器和寄存器卷之间批量传送数据。
存储器的地址由工作寄存器对指定并首先减 1。之后源地址的内容送入目标地址，
源操作数的内容不变。

LDCPD 对应程序存储器，LDEPD 对应外部数据存储器。对于程序存储器，编译器将“Irr”或“rr”的值编译成偶地址，而对于数据存储器则编译成奇地址。

标志位: 没有标志位受影响。

格式:

		字节数	时钟周期	指令代码	寻址模式	
				(Hex)	dst	src
opc	src | dst	2	14	F2	Irr	r

编程实例

```
假如：R0 = 77H，R6 = 30H，R7 = 00H：

LDCPD @RR6,R0     ; (RR6 ← RR6 – 1)
                  ; 77H (R0 的内容) 载入程序存储器地址 2FFFH (3000H – 1H)
                  ; R0 = 77H, R6 = 2FH, R7 = 0FFH

LDEPD @RR6,R0     ; (RR6 ← RR6 – 1)
                  ; 77H (R0 的内容) 载入外部数据存储器地址 2FFFH (3000H – 1H)
                  ; R0 = 77H, R6 = 2FH, R7 = 0FFH
```

6.3.1.44 LDCPI/LDEPI—传送数据前地址加 1 (Load Memory with Pre-Increment)

LDCPI/
LDEPI dst, src

操作: rr ← rr + 1
dst ← src

该指令用于用户栈或在程序/数据存储器与寄存器卷之间批量传送数据。
存储器的地址由工作寄存器对指定并首先加 1。之后源地址的内容载入目标地址，
源操作数的内容不变。

LDCPI 对应程序存储器，LDEPI 对应外部数据存储器。对于程序存储器，编译器将“Irr”或“rr”的值编译成偶地址，而对于数据存储器则编译成奇地址。

标志位: 没有标志位受影响。

格式:

		字节数	时钟周期	指令代码	寻址模式	
				(Hex)	dst	src
opc	src | dst	2	14	F3	Irr	r

编程实例

```
假如：R0 = 7FH，R6 = 21H，R7 = 0FFH：

LDCPI @RR6,R0     ; (RR6 ← RR6 + 1)
                  ; 7FH (R0 的内容) 载入程序存储器地址 2200H (21FFH + 1H)
                  ; R0 = 7FH, R6 = 22H, R7 = 00H

LDEPI @RR6,R0     ; (RR6 ← RR6 + 1)
                  ; 7FH (R0 的内容) 载入外部数据存储器地址 2200H (21FFH + 1H)
                  ; R0 = 7FH, R6 = 22H, R7 = 00H
```

6.3.1.45 LDW—传送字数据 (Load Word)

LDW dst, src

操作: dst ← src

源操作数的内容(字)载入目标操作数。源操作数内容不变。

标志位: 没有标志位受影响。

格式:

	字节数	时钟周期	指令代码	寻址模式	
			(Hex)	dst	src
opc | src | dst	3	8	C4	RR	RR
		8	C5	RR	IR
opc | dst | src	4	8	C6	RR	IML

编程实例

```
假如：R4 = 06H，R5 = 1CH，R6 = 05H，R7 = 02H，寄存器 00H = 1AH，
寄存器 01H = 02H，寄存器 02H = 03H，寄存器 03H = 0FH：

LDW  RR6,RR4      → R6 = 06H, R7 = 1CH, R4 = 06H, R5 = 1CH

LDW  00H,02H      → 寄存器 00H = 03H, 寄存器 01H = 0FH, 寄存器 02H = 03H,
                    寄存器 03H = 0FH

LDW  RR2,@R7      → R2 = 03H, R3 = 0FH,

LDW  04H,@01H     → 寄存器 04H = 03H, 寄存器 05H = 0FH

LDW  RR6,#1234H   → R6 = 12H, R7 = 34H

LDW  02H,#0FEDH   → 寄存器 02H = 0FH, 寄存器 03H = 0EDH

第二个例子中，请注意指令“LDW 00H,02H”将源寄存器 02H 和 03H 中的内容载入目标寄存器00H和
01H，使通用寄存器 00H 中的值变为 03H，01H 中的值变为 0FH。

其它的例子演示了如何通过不用的寻址模式和格式来使用 LDW 指令。
```

6.3.1.46 MULT—无符号数乘法 (Unisigned Multiply)

MULT dst, src

操作: dst ← dst × src

8 位目标操作数(寄存器对中的偶地址寄存器)与源操作数(8位)相乘，乘积(16位)保存在目标地址指定的寄存器对中。两个操作数都为无符号整型数据。

标志位:
- **C:** 如果结果 >255 则置 1；否则清零
- **Z:** 如果结果为“0”则置 1；否则清零
- **S:** 如果结果的最高为“1”则置1；否则清零
- **V:** 清零
- **D:** 不受影响
- **H:** 不受影响

格式:

	字节数	时钟周期	指令代码	寻址模式	
			(Hex)	dst	src
opc | src | dst	3	22	84	RR	R
		22	85	RR	IR
		22	86	RR	IM

编程实例

```
假如：寄存器 00H = 20H，寄存器 01H = 03H，寄存器02H = 09H，寄存器 03H = 06H：

MULT 00H, 02H    → 寄存器 00H = 01H, 寄存器 01H = 20H, 寄存器 02H = 09H
MULT 00H, @01H   → 寄存器 00H = 00H, 寄存器 01H = 0C0H
MULT 00H, #30H   → 寄存器 00H = 06H, 寄存器 01H = 00H

第一个例子中，指令“MULT 00H,02H”将 8 位目标操作数(寄存器对 00H，01H 中的 00H)与源寄存器
02H 中的操作数(09H)相乘。16 位的乘积，0120H，保存在寄存器对 00H，01H中。
```

6.3.1.47 NEXT—Next 指令

NEXT

操作: PC ← @ IP
IP ← IP + 2

NEXT 指令在执行线性代码语言时非常有用。程序存储器中指令指针所指向的字送入 PC，并且指令指针增加 2。

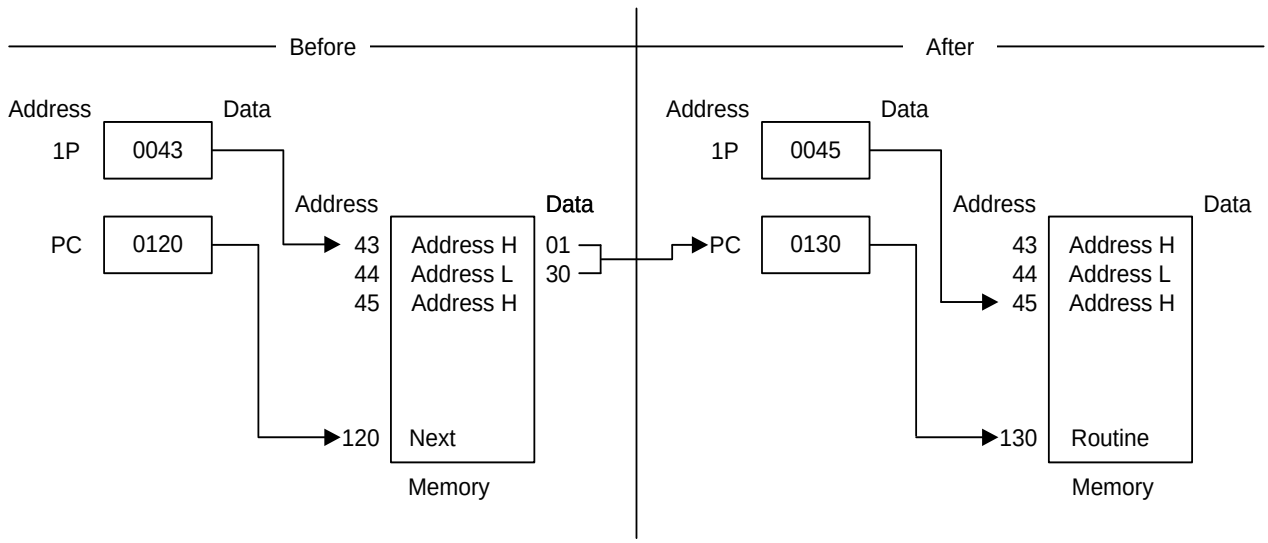
标志位: 没有标志位受影响。

格式:

	字节数	时钟周期	指令代码 (Hex)
opc	1	10	0F

编程实例

下图是一个如何使用 NEXT 指令的例子。



6.3.1.48 NOP—空操作 (No Operation)

NOP

操作: CPU 执行这条指令时，不做任何操作。通常用顺序执行一个或多个 NOP 指令来实现一定时长的延时。

标志位: 没有标志位受影响。

格式:

	字节数	时钟周期	指令代码 (Hex)
opc	1	4	FF

编程实例

在程序中执行 NOP 指令时，没有任何操作发生，只是一个指令执行时间的延时。

6.3.1.49 OR—逻辑或 (Logical OR)

OR dst, src

操作: dst ← dst OR src

源操作数与目标操作数进行逻辑或，结果存放在目标操作数。源操作数的值不受影响。
只要两个操作数中任一个的相应位为“1”，那么或的结果就是“1”，否则为“0”。

标志位:

C: 不受影响
Z: 如果结果是“0”则置 1；否则清零
S: 如果结果的第 7 位(bit 7)为“1”则置 1；否则清零
V: 总是清零
D: 不受影响
H: 不受影响

格式:

	字节数	时钟周期	指令代码	寻址模式	
			(Hex)	dst	src
opc | dst | src	2	4	42	r	r
		6	43	r	lr
opc | src | dst	3	6	44	R	R
		6	45	R	IR
opc | dst | src	3	6	46	R	IM

编程实例

```
假如：R0 = 15H，R1 = 2AH，R2 = 01H，寄存器 00H = 08H，寄存器 01H = 37H，
      寄存器 08H = 8AH：

OR    R0,R1        → R0 = 3FH, R1 = 2AH
OR    R0,@R2       → R0 = 37H, R2 = 01H, 寄存器 01H = 37H
OR    00H,01H      → 寄存器 00H = 3FH, 寄存器 01H = 37H
OR    01H,@00H     → 寄存器 00H = 08H, 寄存器 01H = 0BFH
OR    00H,#02H     → 寄存器 00H = 0AH

第一个例子中，如果工作寄存器 R0 的值是 15H，寄存器 R1 的值是 2AH，指令“OR R0,R1”将寄存器 R0
和 R1 的内容进行逻辑或，并将结果(3FH)存在目标寄存器 R0。

其他的例子演示了如何通过不同的寻址模式和格式来使用逻辑或指令。
```

6.3.1.50 POP—出栈 (Pop from Stack)

POP dst

操作: dst ← @SP
SP ← SP + 1

堆栈指针指定地址的内容被装入目标操作数，然后堆栈指针加1。

标志位: 没有标志位受影响。

格式:

	字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc | dst	2	8	50	R
		8	51	IR

编程实例

```
假如：寄存器 00H = 01H，寄存器 01H = 1BH，SPH (0D8H) = 00H，SPL (0D9H) = 0FBH，
堆栈寄存器 0FBH = 55H：

POP  00H   → 寄存器 00H = 55H, SP = 00FCH
POP  @00H  → 寄存器 00H = 01H, 寄存器 01H = 55H, SP = 00FCH

第一个例子中，通用寄存器 00H 的值为 01H。指令“POP 00H”将地址 00FBH 中的值 (55H)
送入目标寄存器 00H，然后堆栈指针加 1。寄存器 00H 的值变为 55H，并且 SP 指向地址 00FCH。
```

6.3.1.51 POPUD—弹出用户栈 (自减) (Pop User Stack (Drementing))

POPUD dst, src

操作: dst ← src
IR ← IR – 1

该指令用于寄存器卷中的用户自定义栈。用户栈指针指定地址的内容被载入目标寄存器，然后用户栈指针减 1。

标志位: 没有标志位受影响。

格式:

	字节数	时钟周期	指令代码	寻址模式	
			(Hex)	dst	src
opc src dst	3	8	92	R	IR

编程实例

假如：寄存器 00H = 42H (用户栈指针寄存器)，寄存器 42H = 6FH，寄存器 02H = 70H：

```
POPUD 02H,@00H    → 寄存器 00H = 41H, 寄存器 02H = 6FH, 寄存器 42H = 6FH
```

如果通用寄存器 00H 的值为42H，寄存器 42H 的值为 6FH，那么指令“POPUD 02H,@00H”将寄存器 42H 的内容载入目标寄存器 02H，然后用户栈指针减 1，变成 41H。

6.3.1.52 POPUI—弹出用户栈 (自增) (Pop User Stack (Incrementing))

POPUI dst, src

操作: dst ← src
IR ← IR + 1

该指令用于寄存器卷中的用户自定义栈。用户栈指针指定地址的内容载入目标寄存器，然后用户栈指针加1。

标志位: 没有标志位受影响。

格式:

	字节数	时钟周期	指令代码	寻址模式	
			(Hex)	dst	src
opc src dst	3	8	93	R	IR

编程实例

假如：寄存器 00H = 01H，寄存器 01H = 70H：

```
POPUI 02H,@00H    → 寄存器 00H = 02H, 寄存器 01H = 70H, 寄存器 02H = 70H
```

如果通用寄存器 00H 的值为 01H，寄存器 01H 的值为 70H，那么语句“POPUI 02H,@00H”将值70H载入目标通用寄存器 02H，然后用户栈指针(寄存器00H)加1，从 01H 变为 02H。

6.3.1.53 PUSH—压栈 (Push to Stack)

PUSH src

操作: SP ← SP – 1
@SP ← src

PUSH 指令先将栈指针减 1，然后再将源操作数的内容载入减 1 后的栈地址。此操作将在栈顶置入新的数值。

标志位: 没有标志位受影响。

格式:

	字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc | src	2	8 (内部时钟) 8 (外部时钟)	70	R
		8 (内部时钟) 8 (外部时钟)	71	IR

编程实例

假如：寄存器 40H = 4FH，寄存器 4FH = 0AAH，SPH = 00H，SPL = 00H：

```
PUSH 40H    → 寄存器 40H = 4FH, 堆栈寄存器 0FFH = 4FH, SPH = 0FFH, SPL = 0FFH
PUSH @40H   → 寄存器 40H = 4FH, 寄存器 4FH = 0AAH,
              堆栈寄存器 0FFH = 0AAH, SPH = 0FFH, SPL = 0FFH
```

第一个例子中，如果栈指针包含的内容为 0000H，通用寄存器 40H 的内容为 4FH，语句“PUSH 40H”使栈指针从 0000H 减为 0FFFFH，然后将寄存器 40H 的值载入地址 0FFFFH 中，即将这个新的数值加入栈顶。

6.3.1.54 PUSHUD—压入用户栈 (自减) Push User Stack (Decrementing)

PUSHUD dst, src

操作: IR ← IR – 1
dst ← src

该指令用于寄存器卷中的用户自定义栈。PUSHUD 先将用户栈指针减 1，然后将源操作数的内容载入减1后的栈指针指向的寄存器。

标志位: 没有标志位受影响。

格式:

			字节数	时钟周期	指令代码	寻址模式	
					(Hex)	dst	src
opc	dst	src	3	8	82	IR	R

编程实例

假如：寄存器 00H = 03H，寄存器 01H = 05H，寄存器 02H = 1AH：

```
PUSHUD @00H,01H          → 寄存器 00H = 02H, 寄存器 01H = 05H, 寄存器 02H = 05H
```

如果用户栈指针(例如寄存器 00H)的值为 03H，语句“PUSHUD @00H,01H”使用户栈指针减1，变为02H。而寄存器 01H 的值，05H，则载入减 1 后的用户栈指针所指向的寄存器。

6.3.1.55 PUSHUI—压入用户栈 (自增) Push User Stack (Incrementing)

PUSHUI dst, src

操作: IR ← IR + 1
dst ← src

该指令用于寄存器卷中的用户自定义栈。PUSHUI 先将用户栈指针加 1，然后将源操作数的内容载入加1后的栈指针指向的寄存器。

标志位: 没有标志位受影响。

格式:

			字节数	时钟周期	指令代码	寻址模式	
					(Hex)	dst	src
opc	dst	src	3	8	83	IR	R

编程实例

假如：寄存器 00H = 03H，寄存器 01H = 05H，寄存器 04H = 2AH：

```
PUSHUI @00H,01H          → 寄存器 00H = 04H, 寄存器 01H = 05H, 寄存器 04H = 05H
```

如果用户栈指针(例如寄存器 00H)的值为 03H，语句“PUSHUI @00H,01H”使用户栈指针加1，变为 04H。而寄存器 01H 的值，05H，则载入加 1 后的用户栈指针所指向的寄存器。

6.3.1.56 RCF—C 清 0 (Reset Carry Flag)

RCF RCF

操作: C ← 0

进/借位标志位被无条件清零。

标志位: **C:** 清除为“0”

其他标志位都不受影响。

格式:

	字节数	时钟周期	指令代码 (Hex)
opc	1	4	CF

编程实例

假如：C =“1” 或 “0”：

指令 RCF 将进/借位标志位(C)清零。

6.3.1.57 RET—子程序返回 (Return)

RET

操作: PC ← @SP
SP ← SP + 2

RET 指令通常被用来从 CALL 指令所调用的子程序中返回。栈指针指向的地址内容被弹出到程序计数器中，下一条要执行的指令地址由程序计数器的新值指定。

标志位: 没有标志位被影响。

格式:

	字节数	时钟周期	指令代码 (Hex)
opc	1	8 (内部堆栈) 10 (外部堆栈)	AF

编程实例

假如：SP = 00FCH，(SP) = 101AH，PC = 1234：

```
RET          → PC = 101AH, SP = 00FEH
```

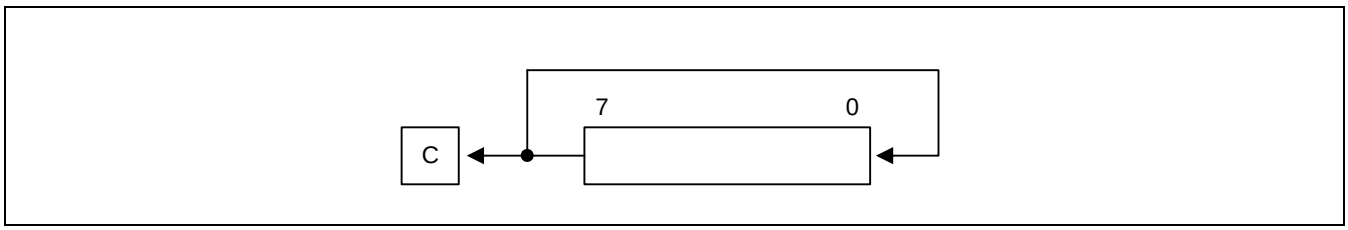
指令“RET”将堆栈指针 00FCH 中的内容(10H)弹出到程序计数器的高字节中，00FDH 中的内容(1AH)弹出到 PC 的低字节，地址 101AH 中的指令被执行。堆栈指针现在指向地址 00FEH。

6.3.1.58 RL—左移 (Rotate Left)

RL dst

操作: C ← dst (7)
dst (0) ← dst (7)
dst (n + 1) ← dst (n), n = 0–6

目标操作数的内容左移一位，原来的第 7 位(bit 7)移到第0位(LSB)和C标志位中，如下图示：



标志位:

C: 如果从最高位(bit 7)移出的数为“1”，则置 1
Z: 如果结果为“0”则置1；否则清零
S: 如果结果的第 7 位(bit 7)为“1”则置 1；否则清零
V: 如果发生溢出则置 1；否则清零
D: 不受影响
H: 不受影响

格式:

	字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc dst	2	4	90	R
		4	91	IR

编程实例

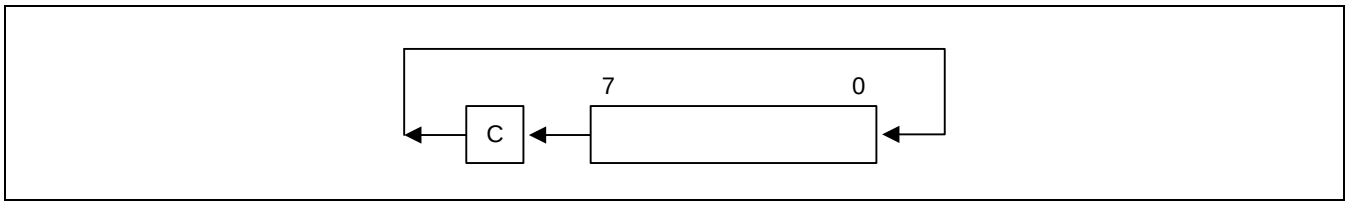
```
假如：寄存器 00H = 0AAH，寄存器 01H = 02H，寄存器 02H = 17H：

RL   00H   → 寄存器 00H = 55H, C =“1”
RL   @01H  → 寄存器 01H = 02H, 寄存器 02H = 2EH, C =“0”

第一个例子中，如果通用寄存器 00H 的值为 0AAH(10101010B)，语句“RL 00H”将 0AAH
左移一位，变为 55H(01010101B)，并将进位和溢出标志置 1。
```

6.3.1.59 RLC—带进位左移 (Rotate Left Through Carry)

RLC dst

操作: dst (0) ← C
C ← dst (7)
dst (n + 1) ← dst (n), n = 0–6

目标操作数的内容通过 C 标志位左移一位，原来的第 7 位(bit 7)移到 C 标志位中，原来的 C 标志位则移至第 0 位(LSB)。



标志位:
C: 如果从最高位(bit 7)移出的为“1”则置 1
Z: 如果结果为“0”则置 1；否则清零
S: 如果结果的第 7 位(bit 7)为“1”则置 1；否则清零
V: 如果发生溢出，也就是目标操作数的符号在移位中发生改变，则置 1；否则清零
D: 不受影响
H: 不受影响

格式:

opc	dst

字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
2	4	10	R
	4	11	IR

编程实例

假如：寄存器 00H = 0AAH，寄存器 01H = 02H，寄存器 02H = 17H，C =“0”：

```
RLC  00H   → 寄存器 00H = 54H, C =“1”
RLC  @01H  → 寄存器 01H = 02H, 寄存器 02H = 2EH, C =“0”
```

第一个例子中，如果通用寄存器的值为 0AAH(10101010B)，语句“RLC 00H”将 0AAH 左移一位。原来的第 7 位(bit 7)将进位标志置 1，C 标志位原来的值则移至寄存器 00H 的第 0 位(LSB)，使寄存器的值变为 55H(01010101B)。寄存器 00H 的最高位 MSB 重新置 C 标志位为“1”，并且将溢出标志位置 1。

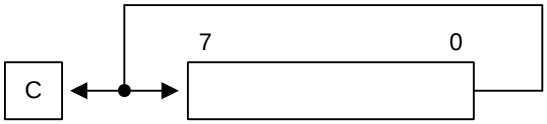
6.3.1.60 RR—右移 (Rotate Right)

RR dst

操作:

C ← dst (0)
dst (7) ← dst (0)
dst (n ← dst (n + 1), n = 0–6

目标操作数的内容右移一位，原来的第 0 位(LSB)移到第 7 位(MSB)和 C 标志位中。



标志位:

C: 如果从第 0 位(LSB)移出的为“1”，则置 1
Z: 如果结果为“0”则置 1；否则清零
S: 如果结果的第 7 位(bit 7)为“1”则置 1；否则清零
V: 如果发生溢出，也就是目标操作数的符号在移位中发生改变，则置 1；否则清零
D: 不受影响
H: 不受影响

格式:

	字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc | dst	2	4	E0	R
		4	E1	IR

编程实例

假如：寄存器 00H = 31H，寄存器 01H = 02H，寄存器 02H = 17H：

```
RR 00H      → 寄存器 00H = 98H, C =“1”
RR @01H     → 寄存器 01H = 02H, 寄存器 02H = 8BH, C =“1”
```

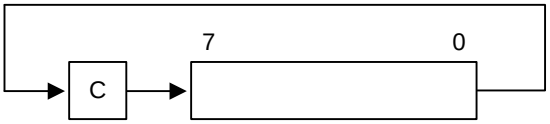
第一个例子中，如果通用寄存器 00H 的值为 31H(00110001B)，语句“RR 00H”将 00H 的值右移一位，原来第 0 位移至第 7 位，目标寄存器的值变为 98H(10011000B)。原来的第 0 位将 C 标志位置“1”，同时符号标志和溢出标志也被置“1”。

6.3.1.61 RRC—带进位右移 (Rotate Right Through Carry)

RRC dst

操作: dst (7) ← C
C ← dst (0)
dst (n) ← dst (n + 1), n = 0–6

目标操作数的内容通过 C 标志位右移一位，原来的第 0 位(LSB)移到C标志位中，原来的 C 标志位则移至第 7 位(MSB)。



标志位:
C: 如果从最低位(bit 0)移出的为“1”，则置 1
Z: 如果结果为“0”则置 1；否则清零
S: 如果结果的第 7 位(bit 7)为“1”则置 1；否则清零
V: 如果发生溢出，也就是说目标操作数的符号在移位中发生改变，则置 1；否则清零
D: 不受影响
H: 不受影响

格式:

	字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc | dst	2	4	C0	R
		4	C1	IR

编程实例

假如：寄存器 00H = 55H，寄存器 01H = 02H，寄存器 02H = 17H，C =“0”：

```
RRC 00H    → 寄存器 00H = 2AH, C =“1”
RRC  @01H  → 寄存器 01H = 02H, 寄存器 02H = 0BH, C =“1”
```

第一个例子中，如果通用寄存器 00H 的值为 55H(01010101B)，语句“RRC 00H”将 00H 的值右移一位，原来第 0 位(“1”)移至进位标志位，原来的 C 标志位(“1”)移至第 7 位，将目标寄存器 00H 的变为 2AH(00101010B)。符号标志和溢出标志都被清零。

6.3.1.62 SB0—选择 Bank 0 (Select Bank 0)

SB0

操作: BANK ← 0

SB0 指令将标志位寄存器 (FLAGS) 中的 Bank 地址标志 (FLAGS.0 位) 清零，在寄存器卷的 Set 1区域选择 Bank 0 。

标志位: 没有标志位受影响。

格式:

	字节数	时钟周期	指令代码 (Hex)
opc	1	4	4F

编程实例

```
语句 SB0 将 FLAGS.0 位清零，选择 Bank 0 中的寄存器进行寻址。
```

6.3.1.63 SB1—选择 Bank 1 (Select Bank 1)

SB1

操作: BANK ← 1

SB1 指令将标志位寄存器 (FLAGS) 中的 Bank 地址标志 (FLAGS.0 位) 置 1，在寄存器卷的 Set 1区域中选择 Bank 1。

注释: 某些 S3F8- 系列的单片机没有 Bank 1。

标志位: 没有标志位受影响。

格式:

	字节数	时钟周期	指令代码 (Hex)
opc	1	4	5F

编程实例

```
指令 SB1 将 FLAGS.0 位清零，选择 Bank 1 的寄存器进行寻址 (如果存在 Bank 1)。
```

6.3.1.64 SBC—带进位减法 (Subtract with Carry)

SBC dst, src

操作: dst ← dst – src – c

目标操作数减去源操作数和当前 C 标志位的值，结果存在目标操作数中。源操作数不受影响。减法操作是通过将源操作数的补码加到目标操作数来完成的。在多次进行的精确运算中，指令允许低阶的操作数向高阶的操作数“借位”。

标志位:

C: 如果借位操作发生 (src > dst) 则置 1；否则清零
Z: 如果结果是“0”则置 1；否则清零
S: 如果结果为负则置 1；否则清零
V: 如果发生溢出，也就是说如果操作数符号相反而差值与源操作数符号相同，则置1；否则清零
D: 总是被置 1
H: 如果从结果低4位的最高位有进位则清零；否则置 1，表示有“借位”发生

格式:

	字节数	时钟周期	指令代码	寻址模式	
			(Hex)	dst	src
opc | dst | src	2	4	32	r	r
		6	33	r	lr
opc | src | dst	3	6	34	R	R
		6	35	R	IR
opc | dst | src	3	6	36	R	IM

编程实例

```
假如：R1 = 10H，R2 = 03H，C =“1”，寄存器 01H = 20H，寄存器 02H = 03H，
      寄存器 03H = 0AH：

SBC   R1,R2        → R1 = 0CH, R2 = 03H
SBC   R1,@R2       → R1 = 05H, R2 = 03H, 寄存器 03H = 0AH
SBC   01H,02H      → 寄存器 01H = 1CH, 寄存器 02H = 03H
SBC   01H,@02H     → 寄存器 01H = 15H, 寄存器 02H = 03H, 寄存器 03H = 0AH
SBC   01H,#8AH     → 寄存器 01H = 95H; C, S, V =“1”

第一个例子中，如果工作寄存器 R1 的值为 10H，R2 的值为 03H，语句“SBC R1,R2”
从目标操作数(10H)中减去源操作数(03H)和 C 标志的值(“1”)，然后将结果(0CH)存放在寄存器 R1 中。
```

6.3.1.65 SCF—C 置 1 (Set Carry Flag)

SCF

操作: C ← 1

将进位标志位 (C) 无条件置 1。

标志位: C: 置为“1”

其它标志位都不受影响。

格式:

	字节数	时钟周期	指令代码 (Hex)
opc	1	4	DF

编程实例

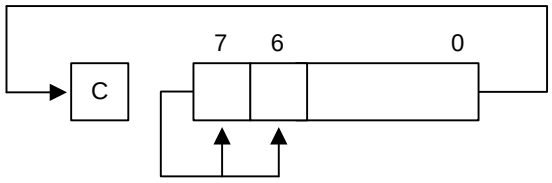
语句 SCF 将进位标志置 1。

6.3.1.66 SRA—算术右移 (Shift Right Arithmetic)

SRA dst

操作: dst (7) ← dst (7)
C ← dst (0)
dst (n) ← dst (n + 1), n = 0–6

将目标操作数算数右移一位，第 0 位(LSB)移至 C 标志，第 7 位(bit 7，符号位)不改变并且移到第 6 位。



标志位:
C: 如果从第 0 位 (LSB) 移出的是“1”则置 1；否则清零
Z: 如果结果是“0”则置 1；否则清零
S: 如果结果是负数则置 1；否则清零
V: 总是清零
D: 不受影响
H: 不受影响

格式:

	字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc | dst	2	4	D0	R
		4	D1	IR

编程实例

假如：寄存器 00H = 9AH，寄存器 02H = 03H，寄存器 03H = 0BCH，C =“1”：

```
SRA   00H    → 寄存器 00H = 0CD, C =“0”
SRA   @02H   → 寄存器 02H = 03H, 寄存器 03H = 0DEH, C =“0”
```

第一个例子中，如果通用寄存器 00H 的值为 9AH(10011010B)，语句“SRA 00H”将寄存器00H右移一位，第 0 位(“0”)清除 C 标志，第7位(“1”)移至第 6 位(第 7 位保持不变)，目标寄存器 00H 的值变为 0CDH(11001101B)。

6.3.1.67 SRP/SRP0/SRP1—设置寄存器指针 (Set Register Pointer)

SRP src

SRP0 src

SRP1 src

操作:

如果 src (1) = 1 且 src (0) = 0 那么: RP0 (3–7) ← src (3–7)
如果 src (1) = 0 且 src (0) = 1 那么: RP1 (3–7) ← src (3–7)
如果 src (1) = 0 且 src (0) = 0 那么: RP0 (4–7) ← src (4–7),
RP0 (3) ← 0
RP1 (4–7) ← src (4–7),
RP1 (3) ← 1

源操作数的第 1 位和第 0 位(LSB)决定写入 2 个寄存器指针中的 1 个还是 2 个一起写入。如果两个寄存器指针都被选择，那么选择的寄存器指针的 3 到 7 位被写入，然后 RP0.3 清零，RP1.3 置 1。

标志位: 没有标志位受影响。

格式:

	字节数	时钟周期	指令代码 (Hex)	寻址模式 src
opc | src	2	4	31	IM

编程实例

语句

```
SRP #40H
```

将地址为 0D6H 的寄存器指针 0(RP0)设定在 40H，地址为 0D7H 的寄存器指针 1(RP1)设定在 48H。

语句“SRP0 #50H”将 RP0 设为 50H，而语句“SRP1 #68H”将 RP1 设为 68H。

6.3.1.68 STOP—Stop 操作 (Stop Operation)

STOP

操作: STOP 指令同时停止 CPU 时钟和系统时钟，使单片机进入 STOP 模式。在 STOP 模式下，CPU 寄存器，外设寄存器，I/O 口的控制和数据寄存器内容都保持不变。
STOP 模式可以被外部的复位操作或者外部中断唤醒。对于复位操作，RESET 引脚的低电平必须保持足够长的时间，以保证晶振稳定所需的时间间隔。在应用程序中，STOP 指令后必须跟有 3 个 NOP 指令，以保证在下条指令执行之前有足够长的时间间隔来稳定晶振。如果 STOP 后没有使用 3 个或者多个 NOP 指令，内部总线的悬浮状态将会产生漏电流。

标志位: 没有标志位受影响。

格式:

	字节数	时钟周期	指令代码	寻址模式	
			(Hex)	dst	src
opc	1	4	7F	–	–

编程实例

语句

```
STOP ;  停止所有单片机操作
NOP
NOP
NOP
```

注释: 在执行 STOP 指令之前，必须设置 STPCON 寄存器值为“10100101B”，否则 STOP 指令不会执行。

6.3.1.69 SUB—减法 (Subtract)

SUB dst, src

操作: dst ← dst – src

目标操作数减去源操作数，结果存放在目标操作数中。源操作数的内容不受影响。
减法操作是将源操作数的补码加到目标操作数来完成的。

标志位:

C: 如果“借位”发生则置 1；否则清零
Z: 如果结果是“0”则置 1；否则清零
S: 如果结果是负数则置 1；否则清零
V: 如果发生溢出，也就是如果操作数符号相反而结果与源操作数的符号相同，则置 1；否则清零
D: 总是被置 1
H: 如果从结果低 4 位的最高位有进位则清零；否则置 1，表示有“借位”发生

格式:

	字节数	时钟周期	指令代码	寻址模式	
			(Hex)	dst	src
opc | dst | src	2	4	22	r	r
		6	23	r	lr
opc | src | dst	3	6	24	R	R
		6	25	R	IR
opc | dst | src	3	6	26	R	IM

编程实例

假如：R1 = 12H，R2 = 03H，寄存器 01H = 21H，寄存器 02H = 03H，寄存器 03H = 0AH：

```
SUB  R1,R2       → R1 = 0FH, R2 = 03H
SUB  R1,@R2      → R1 = 08H, R2 = 03H
SUB  01H,02H     → 寄存器 01H = 1EH, 寄存器 02H = 03H
SUB  01H,@02H    → 寄存器 01H = 17H, 寄存器 02H = 03H
SUB  01H,#90H    → 寄存器 01H = 91H; C, S, V =“1”
SUB  01H,#65H    → 寄存器 01H = 0BCH; C 和 S =“1”, V =“0”
```

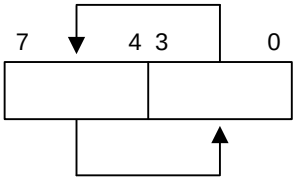
在第一个例子中，如果工作寄存器 R1 的值为12H，R2 为 03H，语句“SUB R1,R2”将源操作数(03H)从目标操作数(12H)中减去，并将结果(0FH)存在目标寄存器 R1 中。

6.3.1.70 SWAP—交换 (Swap Nibbles)

SWAP dst

操作: dst (0 – 3) ↔ dst (4 – 7)

目标操作数的低四位和高四位互相交换。



标志位:

C: 没有定义
Z: 如果结果是“0”则置 1；否则清零
S: 如果结果的第 7 位(bit 7)为“1”则置 1；否则清零
V: 没有定义
D: 不受影响
H: 不受影响

格式:

	字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc | dst	2	4	F0	R
		4	F1	IR

编程实例

```
假如：寄存器 00H = 3EH，寄存器 02H = 03H，寄存器 03H = 0A4H：

SWAP 00H          → 寄存器 00H = 0E3H
SWAP @02H         → 寄存器 02H = 03H, 寄存器 03H = 4AH

第一个例子中，如果通用寄存器 00H 的值为 3EH(00111110B)，语句“SWAP 00H”
将其高四位和低四位交换，使寄存器 00H 的值变为 0E3H(11100011B)。
```

6.3.1.71 TCM—取反后位测试 (Test Complement Under Mask)

TCM dst, src

操作: (NOT dst) AND src

该指令用于测试目标操作数中特定位是否为逻辑1。被测试的位通过将源操作数中相应位设为“1”(屏蔽)来指定。TCM 语句将目标操作数取反，然后与源操作数进行与操作。
通过检查零标志位(Z)来确定结果。目标操作数和源操作数不受影响。

标志位:

C: 不受影响
Z: 如果结果是“0”则置 1；否则清零
S: 如果结果的第 7 位(bit 7)为“1”则置 1；否则清零
V: 总是清零
D: 不受影响
H: 不受影响

格式:

	字节数	时钟周期	指令代码	寻址模式	
			(Hex)	dst	src
opc | dst | src	2	4	62	r	r
		6	63	r	lr
opc | src | dst	3	6	64	R	R
		6	65	R	IR
opc | dst | src	3	6	66	R	IM

编程实例

```
假如：R0 = 0C7H，R1 = 02H，R2 = 12H，寄存器 00H = 2BH，寄存器 01H = 02H，
      寄存器 02H = 23H：

TCM   R0,R1        → R0 = 0C7H, R1 = 02H, Z = “1”
TCM   R0,@R1       → R0 = 0C7H, R1 = 02H, 寄存器 02H = 23H, Z = “0”
TCM   00H,01H      → 寄存器 00H = 2BH, 寄存器 01H = 02H, Z = “1”
TCM   00H,@01H     → 寄存器 00H = 2BH, 寄存器 01H = 02H, 寄存器 02H = 23H, Z  = “1”
TCM   00H,#34      → 寄存器 00H = 2BH, Z = “0”

第一个例子中，如果工作寄存器 R0 的值为 0C7H(11000111B)，寄存器 R1 为 02H(00000010B)，
语句“TCM R0,R1”测试目标寄存器的第 1 位(bit 1)是否为“1”。由于屏蔽值与对应的测试位一致，Z
标志被置 1，可用来确定 TCM 操作的结果。
```

6.3.1.72 TM—位测试 (Test Under Mask)

TM dst, src

操作: dst AND src

该指令用于测试目标操作数中特定位是否为逻辑 0。被测试的位通过将源操作数中相应位设为“1”(屏蔽)来指定，然后目标操作数与源操作数进行与操作。通过检查零标志位(Z)来确定结果。目标操作数和源操作数不受影响。

标志位:

C: 不受影响
Z: 如果结果是“0”则置 1；否则清零
S: 如果结果的第 7 位为“1”则置 1；否则清零
V: 总是清零
D: 不受影响
H: 不受影响

格式:

			字节数	时钟周期	指令代码	寻址模式	
					(Hex)	dst	src
opc	dst | src		2	4	72	r	r
				6	73	r	lr
opc	src	dst	3	6	74	R	R
				6	75	R	IR
opc	dst	src	3	6	76	R	IM

编程实例

假如：R0 = 0C7H，R1 = 02H，R2 = 18H，寄存器 00H = 2BH，寄存器 01H = 02H，
寄存器 02H = 23H：

```
TM    R0,R1       → R0 = 0C7H, R1 = 02H, Z = “0”
TM    R0,@R1      → R0 = 0C7H, R1 = 02H, 寄存器 02H = 23H, Z = “0”
TM    00H,01H     → 寄存器 00H = 2BH, 寄存器 01H = 02H, Z = “0”
TM    00H,@01H    → 寄存器 00H = 2BH, 寄存器 01H = 02H, 寄存器 02H = 23H, Z = “0”
TM    00H,#54H    → 寄存器 00H = 2BH, Z = “1”
```

第一个例子中，如果工作寄存器 R0 的值为 0C7H(11000111B)，寄存器 R1 的值为 02H(00000010B)，语句“TM R0,R1” 测试目标寄存器的第1位是否为“0”。由于屏蔽值与对应的的测试位不匹配，Z 标志被清零，可以用来确定 TM 操作的结果。

6.3.1.73 WFI—等待中断 (Wait for Interrupt)

WFI

操作: 在等待状态时，CPU 被暂停直到中断发生，只有 DMA 传送仍然可以工作。
WFI 状态可以由内部中断，包括快速中断唤醒。

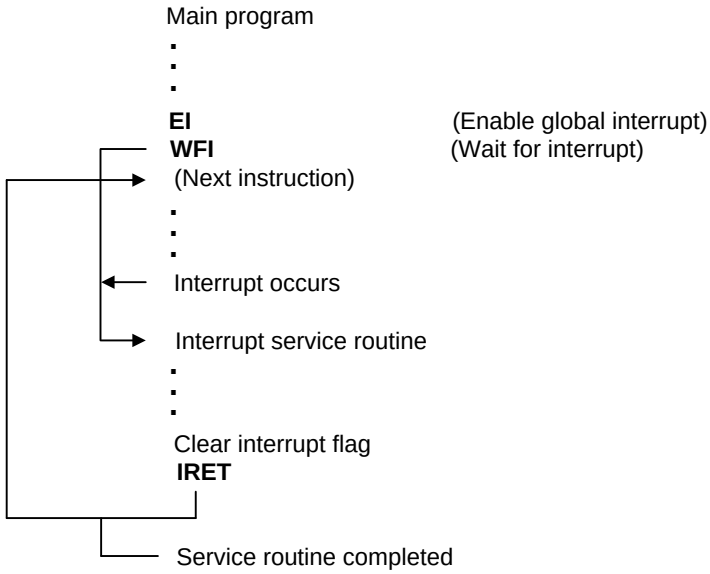
标志位: 没有标志位受影响。

格式:

	字节数	时钟周期	指令代码 (Hex)
opc	1	4n (n = 1, 2, 3, …)	3F

编程实例

以下实例程序的结构表示了“WFI”指令执行后的一系列操作：



6.3.1.74 XOR—逻辑异或 (Logical Exclusive OR)

XOR dst, src

操作: dst ← dst XOR src

源操作数与目标操作数进行逻辑异或，结果存在目标操作数中。
如果两个操作数中对应的位不相同，那么异或的结果为“1”；否则为“0”。

标志位:

C: 不受影响
Z: 如果结果是“0”则置 1；否则清零
S: 如果结果的第 7 位(bit 7)为“1”则置 1；否则清零
V: 总是清零
D: 不受影响
H: 不受影响

格式:

	字节数	时钟周期	指令代码	寻址模式	
			(Hex)	dst	src
opc | dst | src	2	4	B2	r	r
		6	B3	r	lr
opc | src | dst	3	6	B4	R	R
		6	B5	R	IR
opc | dst | src	3	6	B6	R	IM

编程实例

假如：R0 = 0C7H，R1 = 02H，R2 = 18H，寄存器 00H = 2BH，寄存器 01H = 02H，
寄存器 02H = 23H：

```
XOR  R0,R1        → R0 = 0C5H, R1 = 02H
XOR  R0,@R1       → R0 = 0E4H, R1 = 02H, 寄存器 02H = 23H
XOR  00H,01H      → 寄存器 00H = 29H, 寄存器 01H = 02H
XOR  00H,@01H     → 寄存器 00H = 08H, 寄存器 01H = 02H, 寄存器 02H = 23H
XOR  00H,#54H     → 寄存器 00H = 7FH
```

第一个例子中，如果工作寄存器 R0 的值为 0C7H，R1 的值为 02H，语句“XOR R0,R2”将R1和R0的值进行逻辑异或，并将结果(0C5H)存在目标寄存器 R0 中。

7 时钟电路

7.1 概述

S3F84UA/F84U8 MCU 有两种时钟振荡电路：主时钟和副时钟电路。
CPU 和外围硬件工作在由这些电路提供的系统时钟频率上。S3F84UA/F84U8 的最大 CPU 时钟频率由 CLKCON 寄存器设置决定。

7.1.1 系统时钟电路

系统时钟电路有以下几个部分组成：

- 外部石英晶振，陶瓷晶振，RC 振荡源或外部时钟源
- 时钟停止和唤醒功能
- CPU 时钟(fxx 除以1，2，8 或 16)可编程频率分频器
- 系统时钟控制寄存器，CLKCON
- 时钟控制寄存器，OSCCON 和 STOP 控制寄存器，STPCON

7.1.2 CPU 时钟符号

本文档中，采用如下符号描述 CPU 时钟：

fx: 主时钟

fxt: 副时钟

fxx: 已选择的系统时钟

7.1.3 主振荡电路

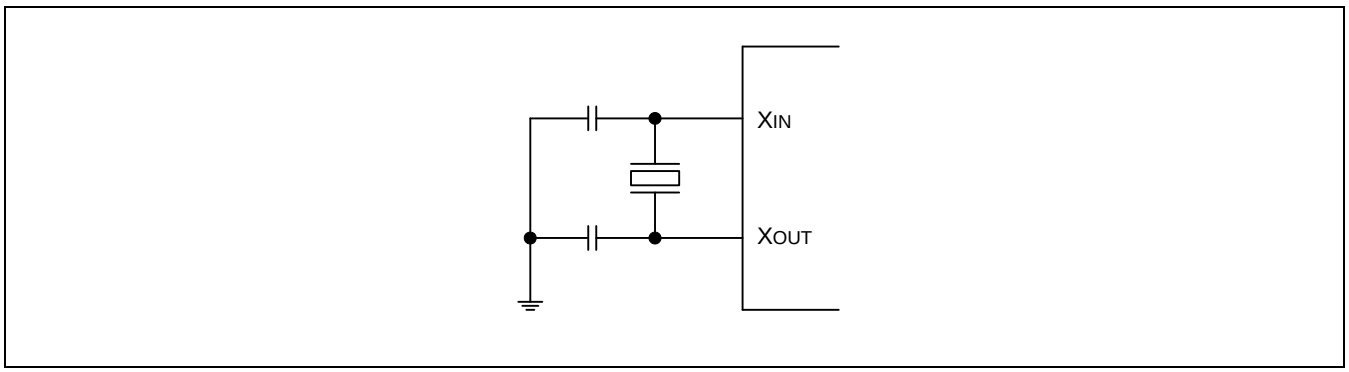


图 7-1 石英/陶瓷晶振 (fX)

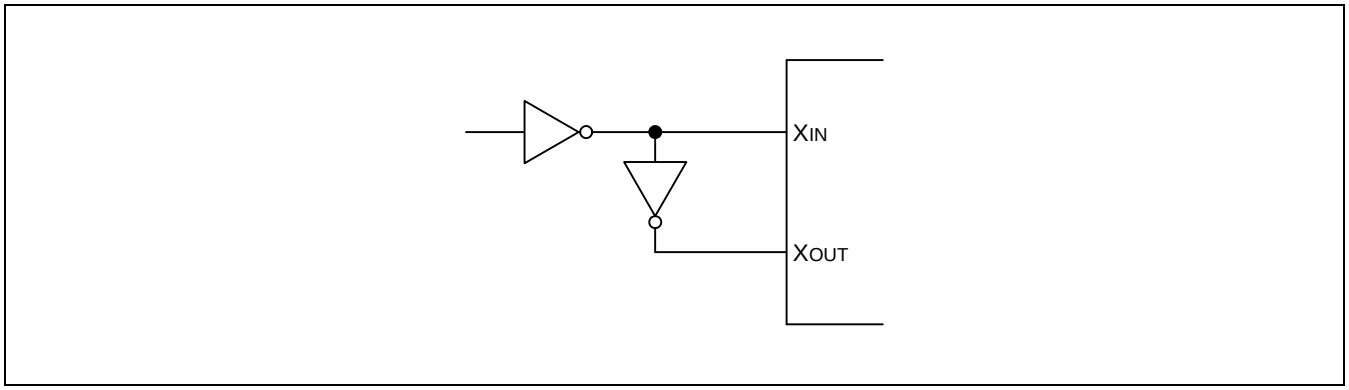


图 7-2 外部振荡器 (fX)

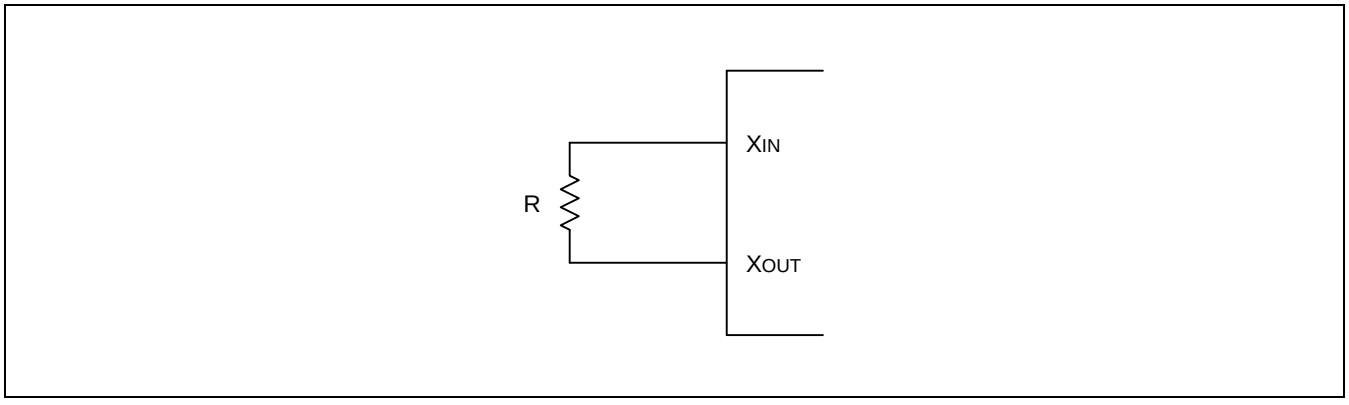


图 7-3 RC 振荡器 (fX)

7.1.4 副振荡 H 电路

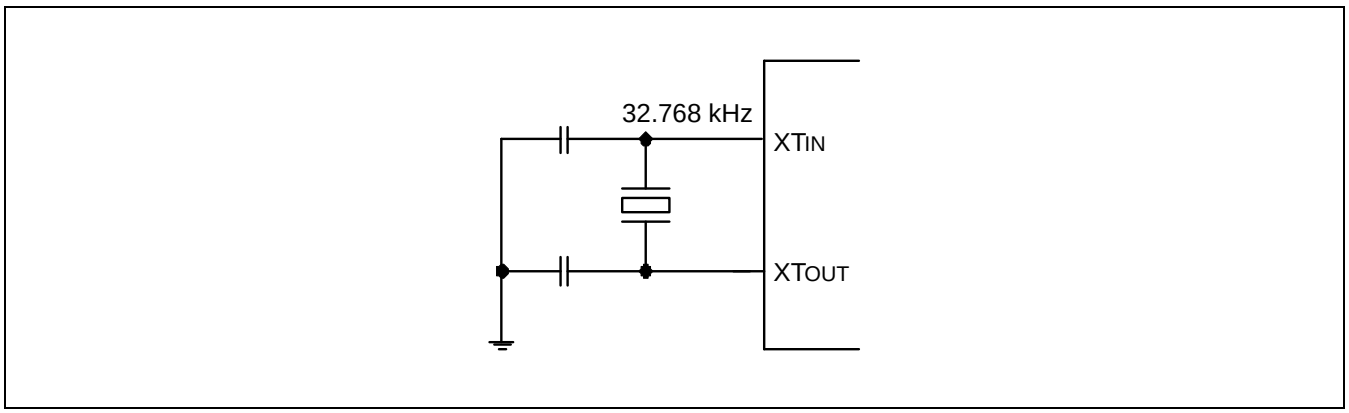


图 7-4 石英振荡器 (fxt)

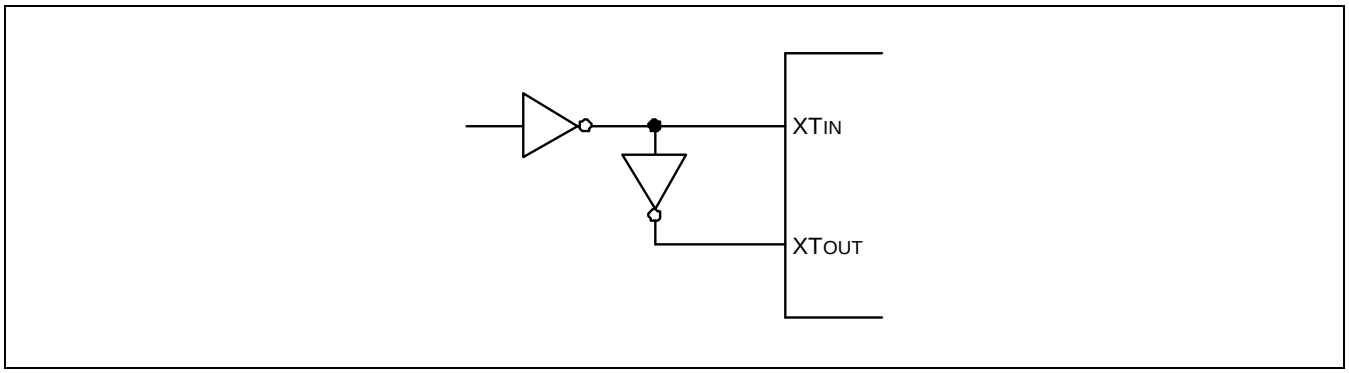


图 7-5 外部振荡器 (fxt)

7.1.5 省电模式下的时钟电路状态

有两种省电模式，STOP 模式和 IDLE 模式，分别对时钟振荡产生如下影响：

- 在 STOP 模式下，主振荡器停止工作，CPU 和外设停止。寄存器卷中的值和当前系统寄存器的值都保持原值。在复位操作或具有 RC 延迟噪音滤波器的外部中断被触发下，CPU 会退出 STOP 模式，振荡器重新起振。当副系统振荡器工作且钟表定时器时钟工作时，内部中断也能唤醒退出 STOP 模式。
- 在 IDLE 模式下，进入 CPU 的内部时钟信号被切断，但内部中断，Timers 和 Timer/Counters 仍然工作。复位，外部或内部中断都可以唤醒 CPU 退出 IDLE 模式。

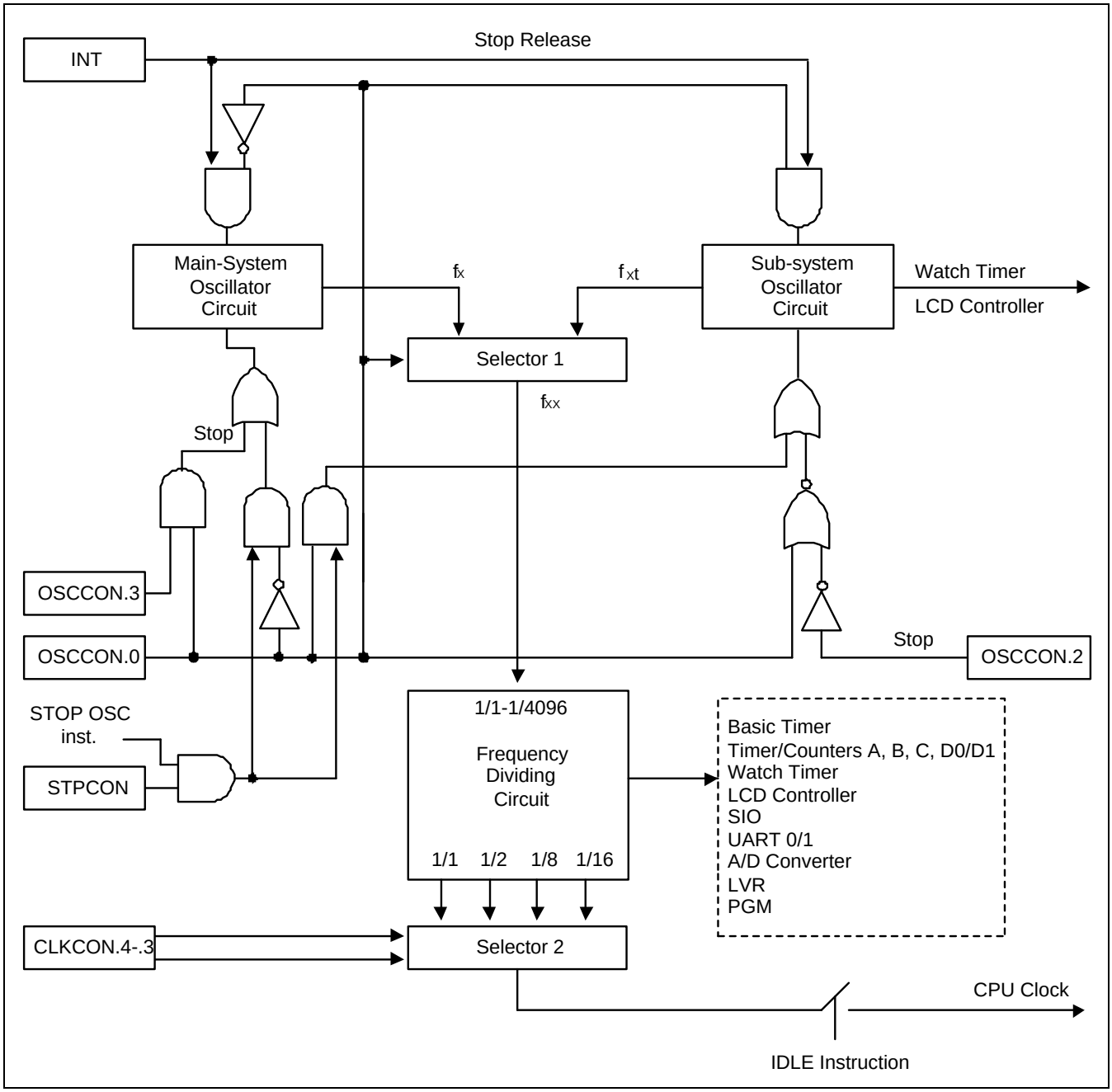


图 7-6 系统时钟电路图

7.1.6 系统时钟控制寄存器 (CLKCON)

系统时钟控制寄存器 CLKCON 的地址是 D4H，Set 1，可读/写，有以下功能：

- 时钟分频系数选择

主振荡器起振后，选择 fxx/16(最慢时钟速率)作为 CPU 时钟。如果需要，用户可提高 CPU 时钟速率至 fxx/8，fxx/2 或 fxx/1。

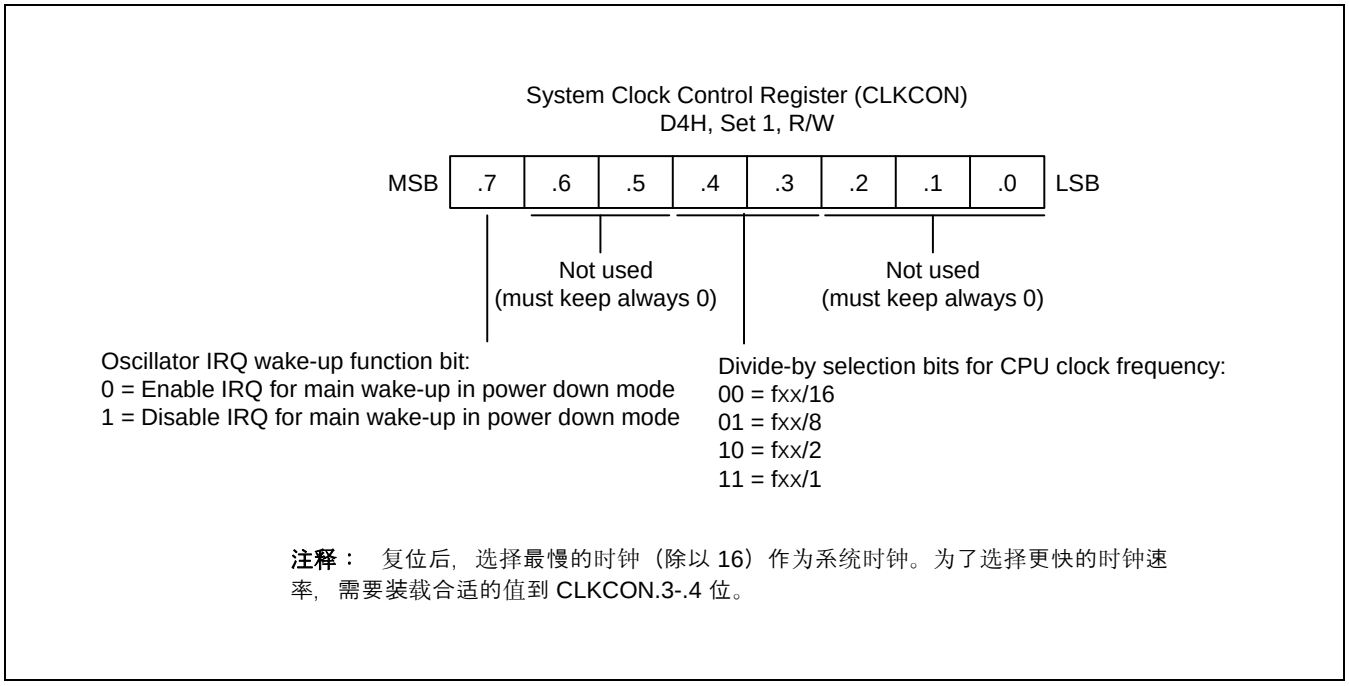


图 7-7 系统时钟控制寄存器 (CLKCON)

7.1.7 时钟控制寄存器 (OSCCON)

时钟控制寄存器 OSCCON 的地址是 FAH，Bank 0，Set 1，可读/写，有以下功能：

- 系统时钟选择
- 主振荡器控制
- 副振荡器控制

OSCCON.0 位选择主时钟或者副时钟作为系统时钟。
复位后，选择主时钟作为系统时钟，原因是 OSCCON.0 位的复位值为“0”。

可通过设置 OSCCON.3 位停止或开启主振荡器。

可通过设置 OSCCON.2 位停止或开启副振荡器。

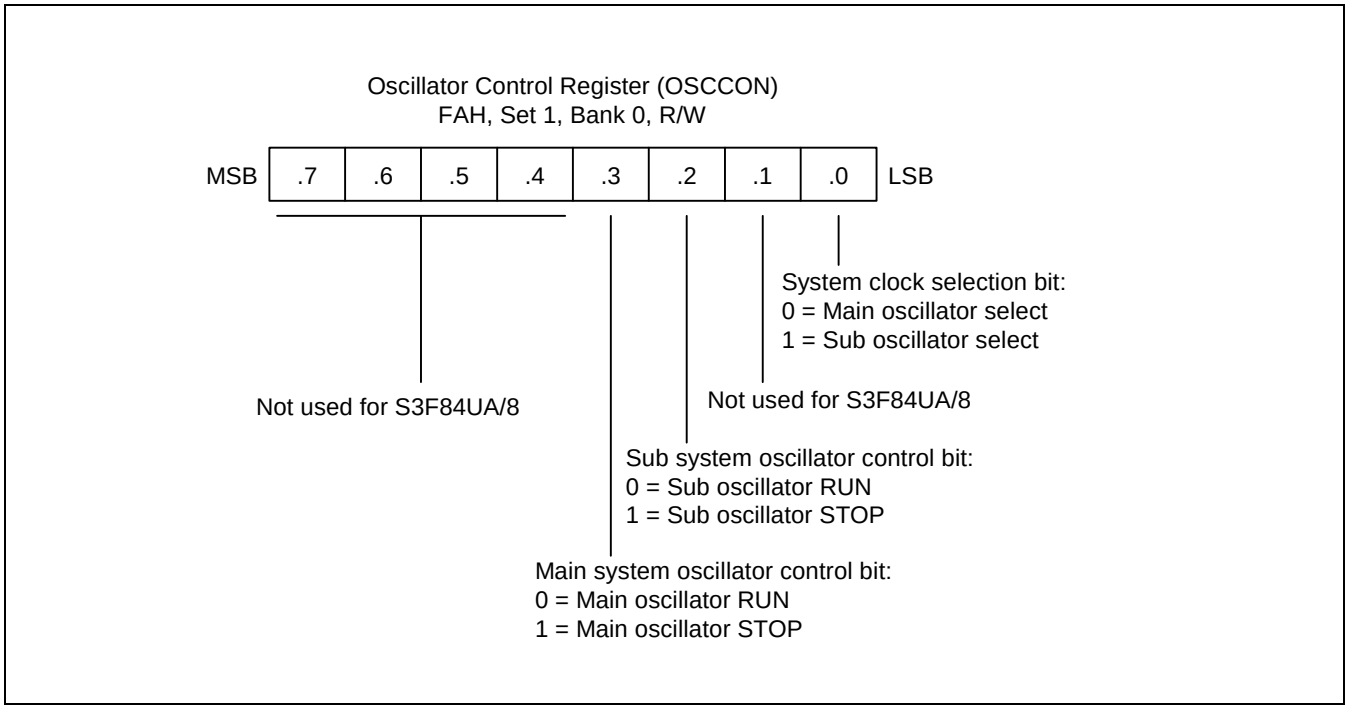


图 7-8 时钟控制寄存器 (OSCCON)

7.1.8 STOP 控制寄存器 (STPCON)

STOP 控制寄存器 STPCON 地址为 EDH，Bank 0，Set 1，可读/写，有以下功能：

- 使能/禁止 STOP 指令
- 复位后，禁止 STOP 指令，原因是 STPCON 的值为“其他值”。如果需要，用户可通过将 STPCON 的值设置为“10100101B”来使能 STOP 指令。

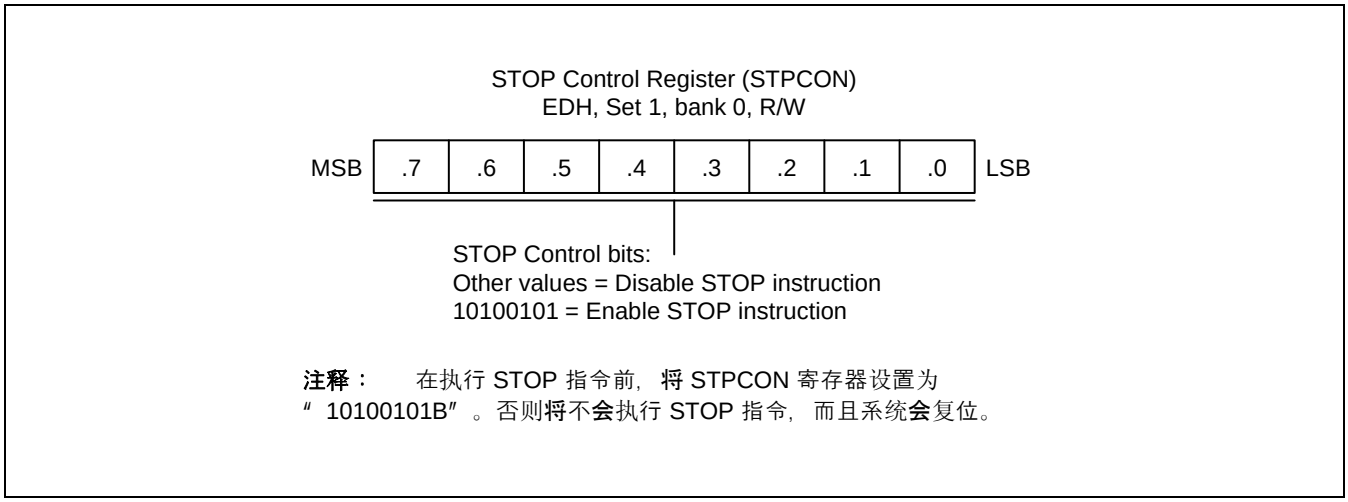


图 7-9 STOP 控制寄存器 (STPCON)

编程实例 7-1 如何使用 STOP 指令

本例介绍当系统时钟选择主时钟时，如何进入 STOP 模式：

```
LD     STOPCON,#1010010B        ; 使能 STOP 指令
STOP                            ; 进入 STOP 模式

NOP
NOP
NOP                             ; 退出 STOP 模式
LD     STOPCON,#00000000B       ; 禁止 STOP 指令
```

7.1.9 切换 CPU 时钟

时钟控制寄存器 OSCCON 中的数据决定选择主或副时钟作为 CPU 时钟，以及由 CLKCON 设置的分频系数。这使得在主副时钟之间的动态切换和修改工作频率成为可能。

OSCCON.0 位选择主时钟 (fx) 或副时钟 (fxt) 作为 CPU 时钟。OSCCON .3 位开启或停止主时钟振荡器，OSCCON.2 位开启或停止副时钟振荡器。CLKCON.4–.3 位控制分频器电路以及将所选 fxx 时钟频率除以 1，2，8 或 16。

举个例子，用户使用默认 CPU 时钟(正常工作模式以及 fxx/16 的主时钟频率)，如果想从 fx 切换到副时钟以及停止主时钟，为了实现这个目的，用户需要将 CLKCON.4-.3 位设置为“11B”，OSCCON.0 位设置为“1B”以及同时将 OSCCON.3 设置为“1”。这样就将时钟从 fx 切换到 fxt，同时停止主时钟振荡器。

实现从副时钟切换到主时钟需进行以下几步：首先，将 OSCCON.3 位设置为“0B”以使能主时钟振荡器；之后，经过一定数目的机器周期后，再将 OSCCON.0 位设置为“0B”来选择主时钟。

编程实例 7-2 切换 CPU 时钟

1. 本例介绍如何从主时钟切换到副时钟：

```
MA2SUB:     LD    OSCCON,#01H       ; 切换到副时钟
                                    ; 停止主时钟振荡器

            RET
```

2. 本例介绍如何从副时钟切换到主时钟：

```
SUB2MA:     AND   OSCCON,#07H       ; 开始主时钟振荡器
            CALL  DLY16             ; 延迟 16 ms
            AND   OSCCON,#06H       ; 切换到主时钟
            RET
DLY16:      SRP   #0C0H
            LD    R0,#20H
DEL:        NOP
            DJNZ  R0,DEL
            RET
```

8 复位和省电模式

8.1 系统复位

8.1.1 概述

外部上电复位时，V_{DD} 管脚电压变为高电平，同时 nRESET 管脚拉低。复位信号通过施密特触发电路与CPU时钟同步，继而复位系统。复位操作使得 S3F84UA/F84U8 处于已知的操作状态。
为了确保芯片正常开始工作，必须维持复位信号为低电平直到 V_{DD} 爬升充分。

接电源后，复位管脚必须维持低电平一段时间，以允许内部 CPU 时钟振荡达到稳定。
复位操作的最小振荡稳定时间为 1ms。

正常工作模式下，V_{DD} 和 nRESET 均为高电平。当 nRESET 管脚变为低电平时，将会导致系统复位。
复位后，系统和外围控制寄存器均恢复为默认值。

总的来说，复位后，各事件会按如下顺序依次发生：

- 禁止所有中断
- 允许看门狗功能(Basic Timer)
- 将 P0-P4 口设为输入模式，并禁止所有 I/O 口的上拉电阻
- 禁止外围控制寄存器和数据寄存器设置，并且恢复到硬件默认值
- 程序计数器(PC)装入 ROM 程序复位地址 0100H 中
- 当编程的时钟振荡稳定后，调入 ROM 中 0100H(和 0101H)单元中的指令并根据 Smart Option 执行
- S3F84UA/F84U8(Full-Flash 器件)的 Smart Option 可更改 ROM 中的复位地址，详细内容请参考第 21 章，嵌入式闪存接口

8.1.2 正常模式下的复位操作

在正常模式下，Test 管脚接到 V_{SS}。复位时可对片上 64K ROM 进行访问(外部接口不会自动设置)。

注释: 如果希望改变振荡稳定的等待时间，用户可以在进入 STOP 状态之前设置 Basic Timer 的控制寄存器 BTCON。如果不希望使能 Basic Timer 看门狗功能(Basic Timer 计数器溢出，会导致系统复位)，用户可以往 BTCON 的高半字节中写“1010B”来禁止看门狗功能。

8.1.3 硬件复位值

表 8-1 至表 *表 8-3* 列出了 CPU，系统寄存器，外围控制寄存器和外围数据寄存器复位后的值。
以下标号用作表示复位值：

- “1”或“0”表示复位后此位对应地为逻辑 1 或逻辑 0
- “x”表示复位后此位值不确定
- “–”表示此位没有用到或未映射，但读此位为 0

表 8-1 S3F84UA/F84U8 Set 1 复位后的寄存器值

寄存器名字	助记标号	地址		复位值 (位)							
		Dec	Hex	7	6	5	4	3	2	1	0
D0H–D2H 地址空间未映射											
Basic Timer 控制寄存器	BTCON	211	D3H	0	0	0	0	0	0	0	0
系统时钟控制寄存器	CLKCON	212	D4H	0	–	–	0	0	–	–	–
系统标志寄存器	FLAGS	213	D5H	x	x	x	x	x	x	0	0
寄存器指针 0	RP0	214	D6H	1	1	0	0	0	–	–	–
寄存器指针 1	RP1	215	D7H	1	1	0	0	1	–	–	–
堆栈指针(高字节)	SPH	216	D8H	x	x	x	x	x	x	x	x
堆栈指针(低字节)	SPL	217	D9H	x	x	x	x	x	x	x	x
指令指针(高字节)	IPH	218	DAH	x	x	x	x	x	x	x	x
指令指针(低字节)	IPL	219	DBH	x	x	x	x	x	x	x	x
中断请求寄存器	IRQ	220	DCH	0	0	0	0	0	0	0	0
中断屏蔽寄存器	IMR	221	DDH	x	x	x	x	x	x	x	x
系统模式寄存器	SYM	222	DEH	–	–	–	x	x	x	0	0
寄存器页指针	PP	223	DFH	0	0	0	0	0	0	0	0

注释:

1. “x”表示复位后此位不确定。
2. “-”表示此位没有用到或未映射，但读此位为 0。

表 8-2 S3F84UA/F84U8 Set 1, Bank 0 复位后的寄存器值

寄存器名字	助记标号	地址		复位值 (位)							
		Dec	Hex	7	6	5	4	3	2	1	0
A/D 转换数据寄存器(高字节)	ADDATAH	208	D0H	x	x	x	x	x	x	x	x
A/D 转换数据寄存器(低字节)	ADDATAL	209	D1H	–	–	–	–	–	–	x	x
A/D 转换控制寄存器	ADCON	210	D2H	–	0	0	0	0	0	0	0
Timer A 计数器	TACNT	224	E0H	0	0	0	0	0	0	0	0
Timer A 数据寄存器	TADATA	225	E1H	1	1	1	1	1	1	1	1
Timer A 控制寄存器	TACON	226	E2H	0	0	0	0	0	0	0	0
Timer B 控制寄存器	TBCON	227	E3H	0	0	0	0	0	0	0	0
Timer B 数据寄存器(高字节)	TBDATAH	228	E4H	1	1	1	1	1	1	1	1
Timer B 数据寄存器(低字节)	TBDATAL	229	E5H	1	1	1	1	1	1	1	1
钟表定时器控制寄存器	WTCON	230	E6H	0	0	0	0	0	0	0	0
SIO 控制寄存器	SIOCON	231	E7H	0	0	0	0	0	0	0	0
SIO 数据寄存器	SIODATA	232	E8H	0	0	0	0	0	0	0	0
SIO 预分频寄存器	SIOPS	233	E9H	0	0	0	0	0	0	0	0
Timer C 计数器	TCCNT	234	EAH	0	0	0	0	0	0	0	0
Timer C 数据寄存器	TCDATA	235	EBH	1	1	1	1	1	1	1	1
Timer C 控制寄存器	TCCON	236	ECH	0	0	0	0	0	0	0	0
STOP 控制寄存器	STPCON	237	EDH	0	0	0	0	0	0	0	0
UART 0 控制寄存器(高字节)	UART0CONH	238	EEH	0	0	0	0	0	0	0	0
UART 0 控制寄存器(低字节)	UART0CONL	239	EFH	0	0	0	0	0	0	0	0
UART 0 数据寄存器	UDATA0	240	F0H	x	x	x	x	x	x	x	x
UART 0 波特率数据寄存器	BRDATA0	241	F1H	1	1	1	1	1	1	1	1
UART 1 控制寄存器(高字节)	UART1CONH	242	F2H	0	0	0	0	0	0	0	0
UART 1 控制寄存器(低字节)	UART1CONL	243	F3H	0	0	0	0	0	0	0	0
UART 1 数据寄存器	UDATA1	244	F4H	x	x	x	x	x	x	x	x
UART 1 波特率数据寄存器	BRDATA1	245	F5H	1	1	1	1	1	1	1	1
闪存扇区地址寄存器(高字节)	FMSECH	246	F6H	0	0	0	0	0	0	0	0
闪存扇区地址寄存器(低字节)	FMSECL	247	F7H	0	0	0	0	0	0	0	0
闪存用户可编程使能寄存器	FMUSR	248	F8H	0	0	0	0	0	0	0	0
闪存控制寄存器	FMCON	249	F9H	0	0	0	0	0	–	–	0
时钟控制寄存器	OSCCON	250	FAH	–	–	–	–	0	0	–	0
中断标志位寄存器	INTPND	251	FBH	–	–	0	0	0	0	0	0
FCH 地址空间未映射											
Basic Timer 计数器	BTCNT	253	FDH	0	0	0	0	0	0	0	0
FEH 地址空间未映射											
中断优先级寄存器	IPR	255	FFH	x	x	x	x	x	x	x	x

表 8-3 S3F84UA/F84U8 Set 1, Bank 1 复位后的寄存器值

寄存器名字	助记标号	地址		复位值 (位)							
		Dec	Hex	7	6	5	4	3	2	1	0
P0 口控制寄存器(高字节)	P0CONH	208	D0H	0	0	0	0	0	0	0	0
P0 口控制寄存器(低字节)	P0CONL	209	D1H	0	0	0	0	0	0	0	0
P0 口上拉电阻使能控制寄存器	P0PUR	210	D2H	0	0	0	0	0	0	0	0
P2 口控制寄存器(高字节)	P2CONH	224	E0H	0	0	0	0	0	0	0	0
P2 口控制寄存器(低字节)	P2CONL	225	E1H	0	0	0	0	0	0	0	0
P1 口控制寄存器	P1CON	226	E2H	–	–	–	–	0	0	0	0
P3 口 N 沟道开漏模式寄存器	PNE3	227	E3H	0	0	0	0	0	0	0	0
P3 口控制寄存器(高字节)	P3CONH	228	E4H	0	0	0	0	0	0	0	0
P3 口控制寄存器(低字节)	P3CONL	229	E5H	0	0	0	0	0	0	0	0
P3 口中断控制寄存器(高字节)	P3INTH	230	E6H	0	0	0	0	0	0	0	0
P3 口中断控制寄存器(低字节)	P3INTL	231	E7H	0	0	0	0	0	0	0	0
P3 口中断标志位寄存器	P3PND	232	E8H	0	0	0	0	0	0	0	0
P3 口上拉电阻使能控制寄存器	P3PUR	233	E9H	0	0	0	0	0	0	0	0
P4 口控制寄存器(高字节)	P4CONH	234	EAH	0	0	0	0	0	0	0	0
P4 口控制寄存器(低字节)	P4CONL	235	EBH	0	0	0	0	0	0	0	0
P4 口上拉电阻使能控制寄存器	P4PUR	236	ECH	0	0	0	0	0	0	0	0
P4 口 N 沟道开漏模式寄存器	PNE4	237	EDH	–	–	0	0	0	0	0	0
Pattern Generation 控制寄存器	PGCON	238	EEH	–	–	–	0	0	0	0	0
Pattern Generation 数据寄存器	PGDATA	239	EFH	0	0	0	0	0	0	0	0
P0 口数据寄存器	P0	240	F0H	0	0	0	0	0	0	0	0
P1 口数据寄存器	P1	241	F1H	–	–	–	–	0	0	0	0
P2 口数据寄存器	P2	242	F2H	0	0	0	0	0	0	0	0
P3 口数据寄存器	P3	243	F3H	0	0	0	0	0	0	0	0
P4 口数据寄存器	P4	244	F4H	0	0	0	0	0	0	0	0
LCD 控制寄存器	LCON	245	F5H	0	0	0	0	0	–	–	0
Timer D0 计数器(高字节)	TD0CNTH	246	F6H	0	0	0	0	0	0	0	0
Timer D0 计数器(低字节)	TD0CNTL	247	F7H	0	0	0	0	0	0	0	0
Timer D0 数据寄存器(高字节)	TD0DATAH	248	F8H	1	1	1	1	1	1	1	1
Timer D0 数据寄存器(低字节)	TD0DATAL	249	F9H	1	1	1	1	1	1	1	1
Timer D0 控制寄存器	TD0CON	250	FAH	0	0	0	0	0	0	0	0
Timer D1 控制寄存器	TD1CON	251	FBH	0	0	0	0	0	0	0	0
Timer D1 计数器(高字节)	TD1CNTH	252	FCH	0	0	0	0	0	0	0	0
Timer D1 计数器(低字节)	TD1CNTL	253	FDH	0	0	0	0	0	0	0	0

寄存器名字	助记标号	地址		复位值 (位)							
		Dec	Hex	7	6	5	4	3	2	1	0
Timer D1 数据寄存器(高字节)	TD1DATAH	254	FEH	1	1	1	1	1	1	1	1
Timer D1 数据寄存器(低字节)	TD1DATAL	255	FFH	1	1	1	1	1	1	1	1

注释:

1. “x”表示复位后此位不确定。
2. “-”表示此位没有用到或未映射，但读此位为 0。

8.2 省电模式

8.2.1 STOP 模式

STOP 指令(指令码 7FH)将会使系统进入 STOP 模式。在 STOP 模式下，CPU 和所有的外围设备停止工作。也就是说，片上的主振荡器停止，供电电流将少于 5μA。当时钟冻结时，所有的系统功能停止，当内部寄存器卷的数据仍保留。可通过 2 种方式退出 STOP 模式：通过复位或中断，详细内容见图 7-6。

注释: 如果使用外部时钟源时，建议不要使用 STOP 模式，原因是 X_{IN} 或 XT_{IN} 输入必须严格地内部接到 V_{SS} 以减少漏电流。

8.2.1.1 使用复位操作退出 STOP 模式

当复位信号恢复到高电平时，系统将退出 STOP 模式，所有的系统和外围控制寄存器恢复为默认硬件值，但数据寄存器的值保持不变。复位操作自动选择最慢时钟 fxx/16 ，原因是 CLKCON.3 和 CLKCON.4 位被清为“00B”。当编程设定的时钟振荡稳定后，CPU 调入 ROM 中 0100H(和 0101H)单元中的程序指令指令执行系统初始化。

8.2.1.2 使用外部中断退出 STOP 模式

具有 RC 延迟噪声滤波电路的外部中断可让系统退出 STOP 模式。根据 MCU 当前的工作模式给定的状态，这些中断可以用来退出 STOP 模式。S3F84UA/F84U8 中断结构中的可用来退出 STOP 模式的外部中断有：

- 外部中断 P3.0–P3.7 (INT0–INT7)

若想退出 STOP 模式，请注意如下条件：

- 若用户想使用外部中断退出 STOP 模式，除 STPCON 寄存器外的系统和外围控制寄存器的当前值不能更改
- 若用户想使用内部或者外部中断退出 STOP 模式，可编程设定时钟振荡的稳定时间间隔。为了实现这个目的，在进入 STOP 模式之前设置合适的控制和时钟
- 当使用外部中断退出 STOP 模式时，CLKCON.4 和 CLKCON.3 位对设置保持不变，选择当前的时钟分频
- 退出 STOP 模式后，执行外部中断。中断服务程序退出后，返回执行 STOP 指令的下一条指令

8.2.1.3 使用内部中断退出 STOP 模式

激活任意一个已经使能的中断，将退出 STOP 模式。其他事情和使用外部中断一致。

如何进入 STOP 模式。

先操作 STPCON 寄存器，之后再执行 STOP 指令(保持这个顺序)。

```
LD    STPCON,#10100101B
STOP
NOP
NOP
NOP
```

8.2.1.4 IDLE 模式

IDLE 指令(指令码 6FH)将会使系统进入 IDLE 模式。在 IDLE 模式下，CPU 操作停止，但一些外围设备仍工作. IDLE 模式下，主振荡器进入 CPU 的时钟被切断，但所有的外围设备仍工作，各 I/O 口仍然保持进入 IDLE 模式前的状态(输入或输出)。

可以通过 2 种方式退出 IDLE 模式：

1. 执行复位操作。所有的系统和外围控制寄存器复位至默认值，但所有数据寄存器的内容保持不变。
   复位操作自动选择最慢时钟 fxx/16 ，原因是 CLKCON.3 和 CLKCON.4 位被清为“00B”。
   如果中断被屏蔽，执行复位操作是唯一可以退出 IDLE 模式的方式。

2. 激活任意一个已使能的中断，退出 IDLE 模式。当用户使用中断退出 IDLE 模式，CLKCON.4 和 CLKCON.3 寄存器的值保持不变，选择当前的时钟分频值。之后响应中断。中断处理完毕后，返回执行 IDLE 指令的下一条指令。

9 I/O 口

9.1 概述

S3F84UA/F84U8 MCU 有 5 个位可编程 I/O 口，P0–P4。P1 口有 4 位，其余均为 8 位，一共 36 个 I/O 管脚。每个口可灵活地设置以满足应用设计的需要。

CPU 可以通过直接读写 I/O 口寄存器来访问这些端口，不需要的 I/O 指令。除 P1.2 和 P1.3 外，S3F84UA/F84U8 的所有端口都可设置为输入或输出模式。所有的 LCD 信号管脚与普通 I/O 口复用。

表 9-1 表述 S3F84UA/F84U8 I/O 口的功能概述。

表 9-1 S3F84UA/F84U8 口设置概述

I/O 口	设置选项
0	8 位可编程 I/O 口。可由软件设定为输入或推挽式输出模式。通过软件设定上拉电阻。P0.0-P0.7 还可作为 AD0–AD7 或 PG0–PG7。
1	4 位可编程 I/O 口。可由软件设定为输入或推挽式输出模式。通过软件设定上拉电阻。 P1.2 和 P1.3 只能设置为推挽式输出模式端口。P1.0–P1.1 可复用为 XT_{OUT}, XT_{IN}.
2	8 位可编程 I/O 口。可由软件设定为输入或推挽式输出模式。通过软件设定上拉电阻。P2.0-P2.7 可作为 LCD COM 和 SEG 信号使用。
3	8 位可编程 I/O 口。可由软件设定为施密特触发输入，推挽式输出或开漏输出模式。 通过软件设定上拉电阻。P3.0–P3.7 可作为外部中断 INT0–INT7 (带噪声滤波器，中断使能和标志位控制) 输入。另外，P3.0–P3.7 可复用为 BUZ, SO, SI, SCK, TD1CLK, TD1OUT/TD1PWM/TD1CAP, TD0CLK, TD0OUT/TD0PWM/TD0CAP 或 LCD SEG。
4	8 位可编程 I/O 口。可由软件设定为施密特触发输入，推挽式输出或开漏输出模式。 通过软件设定上拉电阻。P4.0–P4.7 可复用为 TCOUT/TCPWM, TBPWM, TACLK, TAOUT/TAPWM/TACAP, TXD1, RXD1, TXD0, RXD0。

9.2 端口数据寄存器

表 9-2 列出了 S3F84UA/F84U8 5 个 I/O 口数据寄存器的概述。P0-P4 口数据寄存器的一般格式如图 *图 9-2* 所示：

表 9-2 端口 数据寄存器概述

寄存器名称	标号	十进制	十六进制	位置	R/W
P0 口数据寄存器	P0	240	F0H	Set 1，Bank 1	R/W
P1 口数据寄存器	P1	241	F1H	Set 1，Bank 1	R/W
P2 口数据寄存器	P2	242	F2H	Set 1，Bank 1	R/W
P3 口数据寄存器	P3	243	F3H	Set 1，Bank 1	R/W
P4 口数据寄存器	P4	244	F4H	Set 1，Bank 1	R/W

9.2.1 P0 口

P0 口为一个可独立配置的 8 位 I/O 口管脚。通过读写 P0 口数据寄存器 P0(地址：F0H，Set 1，Bank 1)可直接访问 P0 管脚。P0.0–P0.7 可设置为输入(带上拉或不带上拉电阻)和推挽式输出或者其他如下复用功能：

- 低字节管脚 (P0.0-P0.3)：PG0-PG3/AD0-AD3
- 高字节管脚 (P0.4-P0.7)：PG4-PG7/AD4-AD7

9.2.1.1 P0 口控制寄存器 (P0CONH, P0CONL)

P0 口有两个 8 位控制寄存器：P0CONH 用于 P0.4-P0.7，P0CONL 用作 P0.0-P0.3。复位后，P0CONH 和 P0CONL 寄存器清为“00H”，将所有管脚设置为输入模式。
用户也可使用控制设置来选择输入(带上来电阻或不带上拉电阻)或推挽式输出模式以及使能复用功能。

9.2.1.2 P0 口上拉电阻使能控制寄存器 (P0PUR)

使用 P0 口上拉电阻使能控制寄存器 P0PUR(地址：D2H，Set 1，Bank 0)可对 P0 管脚每个口单独设置上拉电阻。

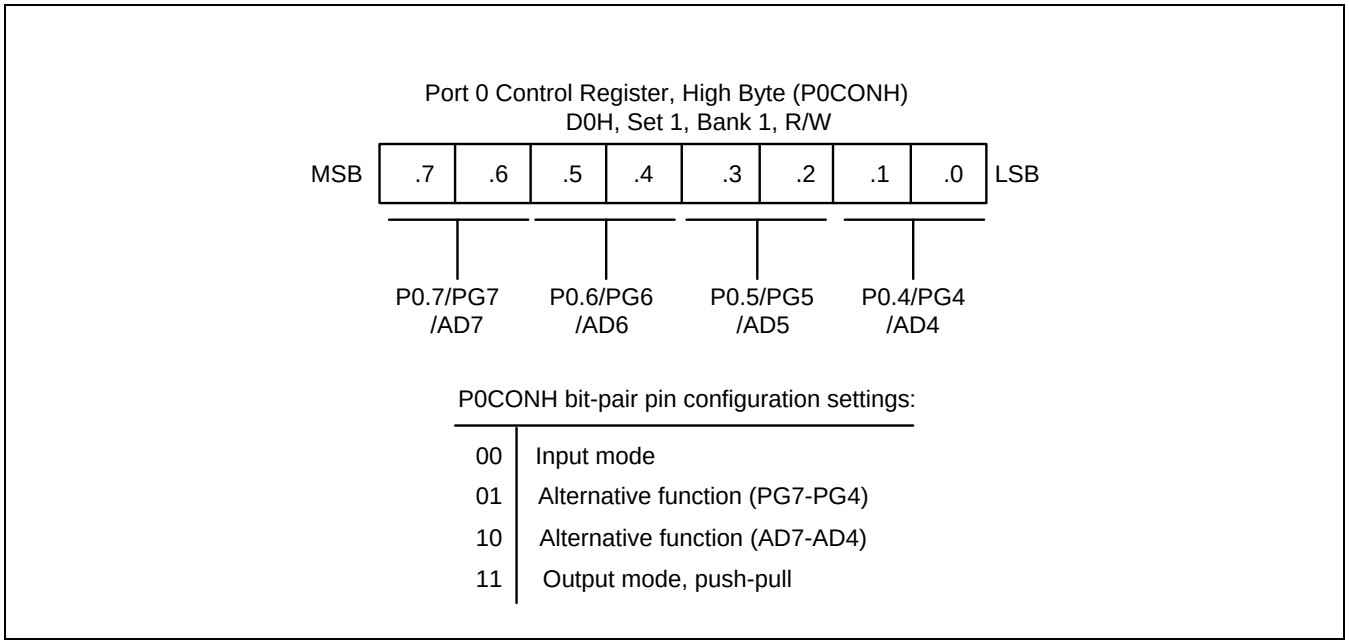


图 9-1 P0 口高字节控制寄存器 (P0CONH)

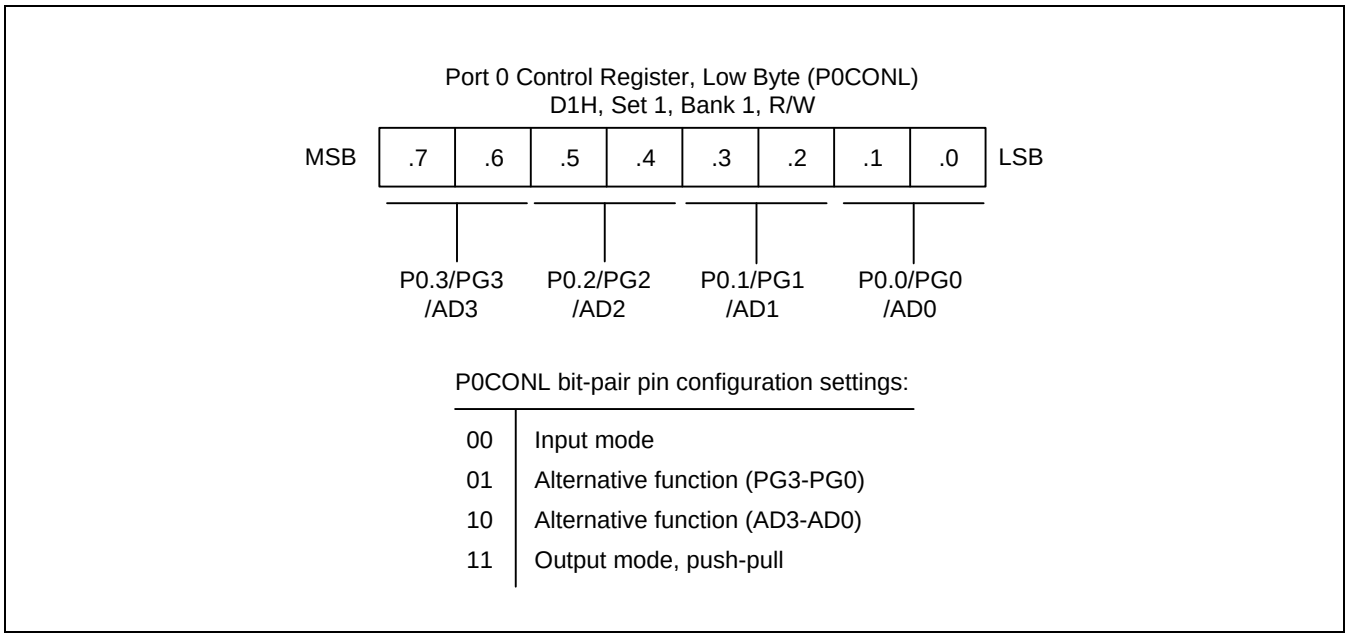


图 9-2 P0 口低字节控制寄存器 (P0CONL)

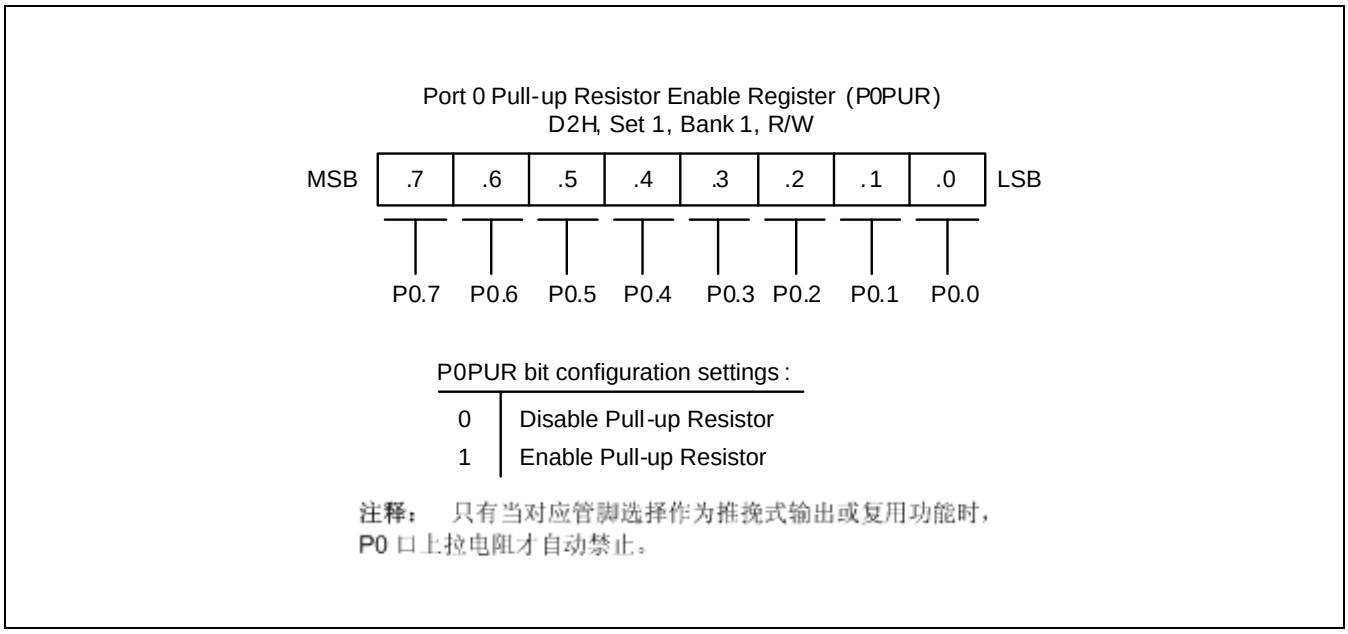


图 9-3 P0 口上拉电阻使能控制寄存器 (P0PUR)

9.2.2 P1 口

P1 口为一个可独立配置的 4 位 I/O 口。通过读写 P1 口数据寄存器 P1(地址：F1H，Set 1，Bank 1)可直接访问 P1 管脚。P1.0–P1.1 可设置为输入(带上拉或不带上拉电阻)和推挽式输出或者其他如下复用功能：

- 低半字节管脚 (P1.0-P1.1)：XT_{OUT}, XT_{IN}

9.2.2.1 P1 口控制寄存器 (P1CON)

P1 口有一个 4 位控制寄存器：P1CON 用于 P1.1-P1.0。复位后，P1CON 寄存器清为“00H”，将所有管脚设置为输入模式。用户也可使用用控制设置来选择输入，输出模式，使能上拉电阻以及使能复用功能。

当对这个端口编程时，确保任何使用 P1 口控制寄存器设置的外设 I/O口复用功能必须在对应的外设模块中使能。

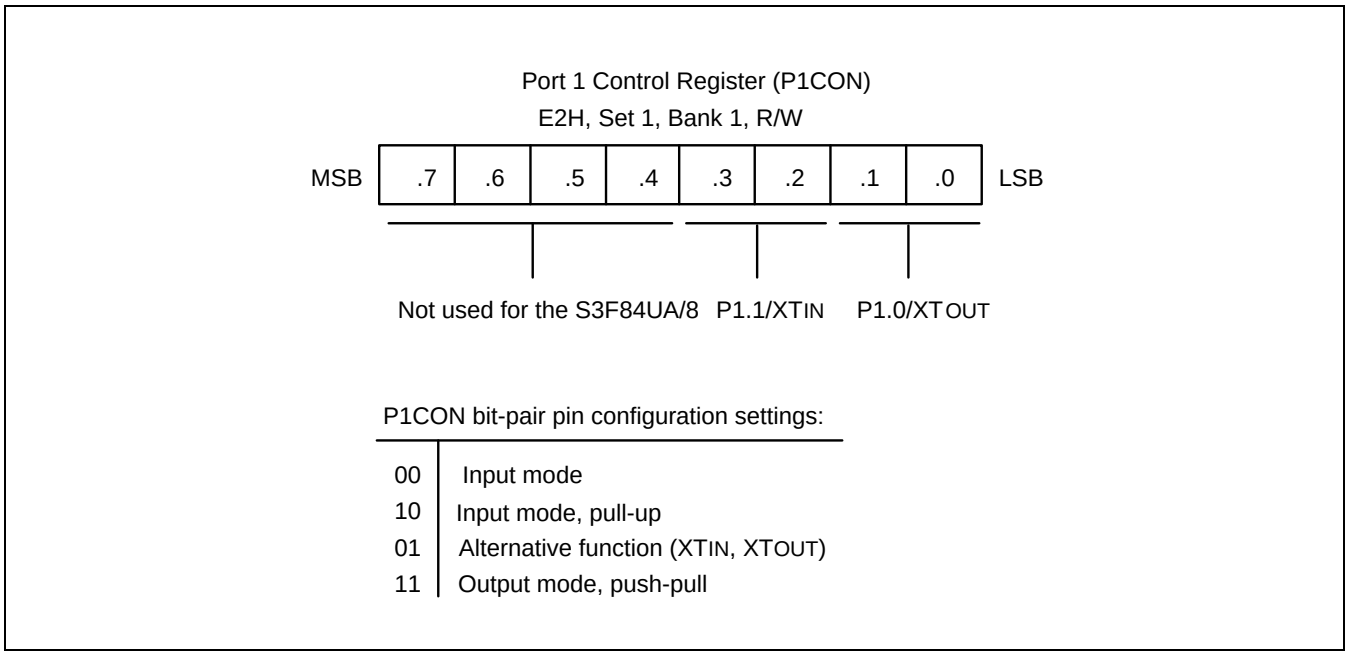


图 9-4 P1 口控制寄存器

9.2.3 P2 口

P2 口为一个可独立配置的 8 位 I/O 口管脚。通过读写 P2 口数据寄存器 P2(地址：F2H，Set 1，Bank 1)可直接访问 P2 管脚。P2.0–P2.7 可设置为输入(带上拉或不带上拉电阻)和推挽式输出或者其他如下复用功能：

- 低字节管脚 (P2.0–P2.3)：COM0, COM1, COM2/SEG0, COM3/SEG1
- 高字节管脚 (P2.4–P2.7)：COM4/SEG2, COM5/SEG3, COM6/SEG4, COM7/SEG5

9.2.3.1 P2 口控制寄存器 (P2CONH, P2CONL)

P2 口有两个 8 位控制寄存器：P2CONH 用于 P2.4-P2.7，P2CONL 用作 P2.0-P2.3。复位后，P2CONH 和 P2CONL 寄存器清为“00H”，将所有管脚设置为输入模式。
用户也可使用控制设置来选择输入（带上来电阻或不带上拉电阻）或推挽式输出模式以及使能复用功能。

当对这个端口编程时，确保任何使用 P2 口控制寄存器设置的外设 I/O口复用功能必须在对应的外设模块中使能。

Port 2 Control Register, High Byte (P2CONH)
E0H, Set 1, Bank 1, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB

P2.7/SEG5 /COM7 | P2.6/SEG4 /COM6 | P2.5/SEG3 /COM5 | P2.4/SEG2 /COM4

P2CONH bit-pair pin configuration settings:

00	Input mode
01	Input mode, pull-up
10	Alternative function (LCD signal)
11	Output mode, push-pull

图 9-5 P2 口高字节控制寄存器 (P2CONH)

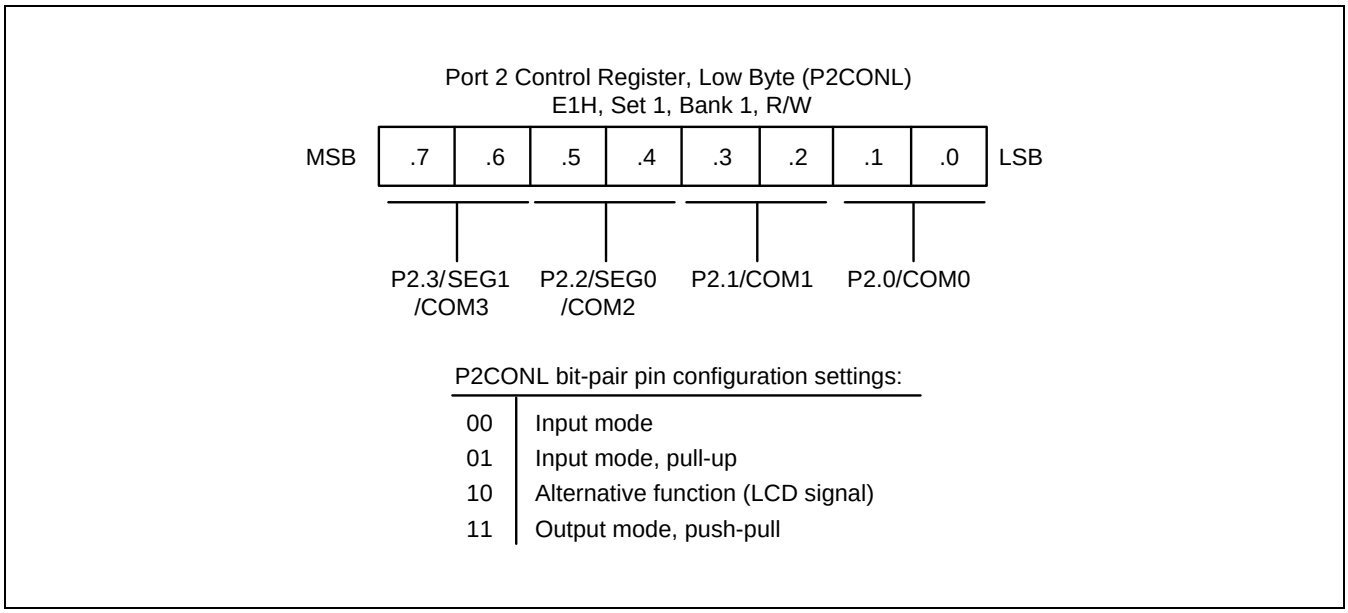


图 9-6 P2 口低字节控制寄存器 (P2CONL)

9.2.4 P3 口

P3 口为一个可独立配置的 8 位 I/O 口管脚。通过读写 P3 口数据寄存器 P3(地址：F3H，Set 1，Bank 1)可直接访问 P3 管脚。P3.0–P3.7 可设置为输入(带上拉或不带上拉电阻)，推挽式输出，LCD 的 SEG 管脚或者其他如下复用功能：

- 低字节管脚 (P3.0–P3.3)：INT0/BUZ, INT1/SO, INT2/SI, INT3/SCK
- 高字节管脚 (P3.4–P3.7)：INT4/TD1CLK, INT5/TD1OUT/TD1PWM/TD1CAP, INT6/TD0CLK,

9.2.4.1 INT7/TD0OUT/TD0PWM/TD0CAP

P3 口控制寄存器 (P3CONH, P3CONL)

P3 口有两个 8 位控制寄存器：P3CONH 用于 P3.4-P3.7，P3CONL 用作 P3.0-P3.3。复位后，P3CONH 和 P3CONL 寄存器清为“00H”，将所有管脚设置为输入模式。在输入模式下，有三种不同的选择：

- 施密特触发输入(下降沿触发中断)
- 施密特触发输入(上升沿触发中断)
- 施密特触发输入(下降沿和上升沿触发中断)

当对这个端口编程时，确保任何使用 P3 口控制寄存器设置的外设 I/O 口复用功能必须在对应的外设模块中使能。

9.2.4.2 P3 口中断控制和标志位寄存器 (P3INTH, P3INTL, P3PND)

为了处理 P3 口管脚上的外部中断，特提供额外的控制寄存器：P3 口中断控制寄存器 P3INTH (高字节，地址：E6H，Set 1，Bank 1)，P3INTL(低字节，地址：E7H，Set 1，Bank 1)和 P3口中断标志位寄存器 P3PND(地址：E8H，Set 1，Bank 1)。

当执行中断服务程序时，用户可通过查询 P3 口中断标志寄存器种的中断标志条件以及清除中断标志条件。
应用程序通过定期时间内查询 P3PND 寄存器检测到中断请求。

当任何 P3 管脚的中断使能位为“1”时，管脚上的上升沿或者下降沿信号将产生一个中断请求，对应的 P3PND 位自动设置为“1”，同时 IRQ 级变为低电平通知 CPU 有个中断请求等待处理。
当 CPU 响应到该中断请求时，应用程序必须通过往对应的 P3PND 位写“0”来清除中断标志条件。

9.2.4.3 P3 口上拉电阻使能控制寄存器 (P3PUR)

通过 P3 口上拉电阻使能控制寄存器 P3PUR(地址：E9H，Set 1，Bank 1)，用户可对 P3 口管脚单独设置上拉电阻。

9.2.4.4 P3 口 N 沟道开漏模式寄存器 (PNE3)

通过 P3 口 N 沟道开漏模式寄存器 PNE3(地址：E3H，Set 1，Bank 1)，用户可对 P3 口管脚单独设置推挽式输出或开漏输出模式。

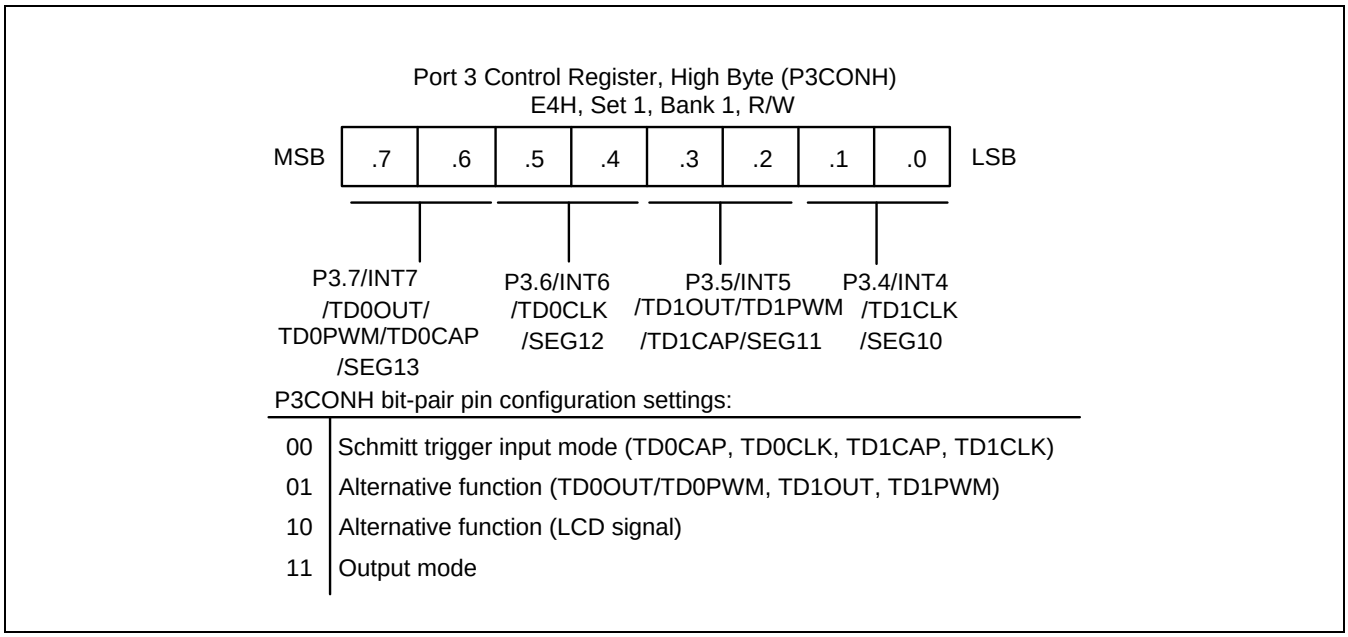


图 9-7 P3 口高字节控制寄存器 (P3CONH)

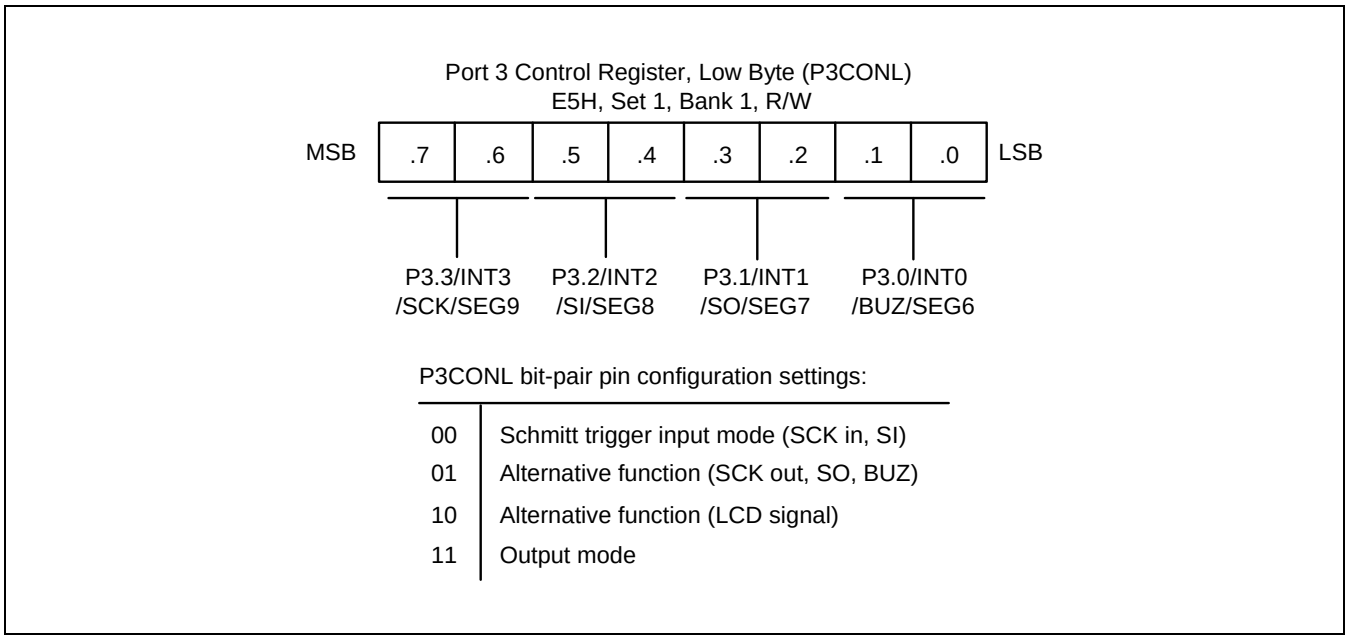
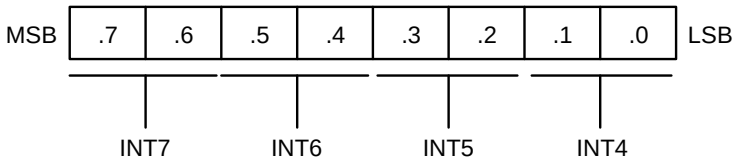


图 9-8 P3 口低字节控制寄存器 (P3CONL)

Port 3 Interrupt Control Register, High Byte (P3INTH)
E6H, Set 1, Bank 1, R/W

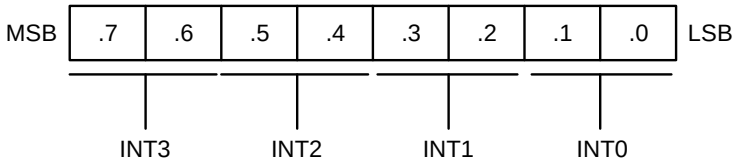


P3INTH bit-pair pin configuration settings:

00	Disable interrupt
01	Enable interrupt by falling edge
10	Enable interrupt by rising edge
11	Enable interrupt by both falling and rising edge

图 9-9 P3 口高字节中断控制寄存器 (P3INTH)

Port 3 Interrupt Control Register, Low Byte (P3INTL)
E7H, Set 1, Bank 1, R/W

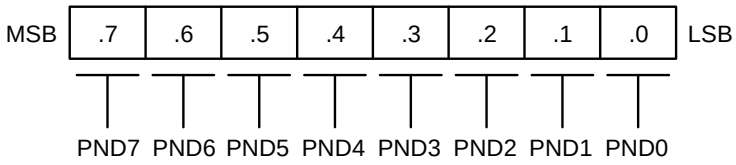


P3INTL bit-pair pin configuration settings:

00	Disable interrupt
01	Enable interrupt by falling edge
10	Enable interrupt by rising edge
11	Enable interrupt by both falling and rising edge

图 9-10 P3 口低字节中断控制寄存器 (P3INTL)

Port 3 Interrupt Pending Register (P3PND)
E8H, Set 1, Bank 1, R/W

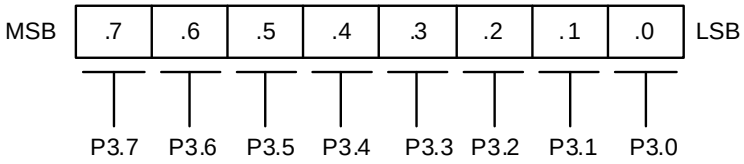


P3PND bit configuration settings:

0	Interrupt request is not pending, pending bit clear when write 0
1	Interrupt request is pending

图 9-11 P3 口中断标志位寄存器 (P3PND)

Port 3 Pull-up Resistor Enable Register (P3PUR)
E9H, Set 1, Bank 1, R/W



P3PUR bit configuration settings :

0	Disable Pull-up Resistor
1	Enable Pull-up Resistor

注释： 只有当对应管脚选择作为推挽式输出或复用功能时，P3 口上拉电阻才自动禁止。

图 9-12 P3 口上拉电阻使能控制寄存器 (P3PUR)

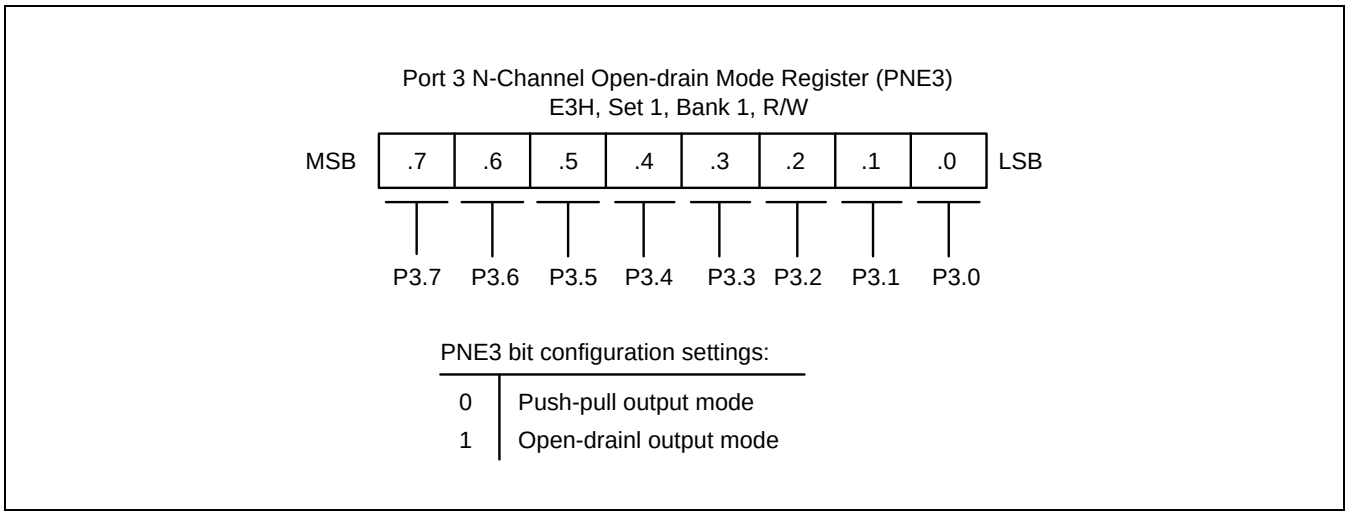


图 9-13 P3 口 N 沟道开漏模式寄存器 (PNE3)

9.2.5 P4 口

P4 口为一个可独立配置的 8 位 I/O 口管脚。通过读写 P4 口数据寄存器 P4(地址：F4H，Set 1，Bank 1)可直接访问 P4 管脚。P4.0–P4.7 可设置为输入(带上拉或不带上拉电阻)或输出(推挽式或开漏)。另外，P4.7–P4.0 也能作为 LCD 的 SEG 管脚或者其他如下复用功能：

- 低字节管脚 (P4.0-P4.3)：TCOUT, TBPWM, TACLK, TACAP, TAOUT/TAPWM
- 高字节管脚 (P4.4-P4.7)：TxD1, RxD1, TxD0, RxD0

9.2.5.1 P4 口控制寄存器 (P4CONH, P4CONL)

P4 口有两个 8 位控制寄存器：P4CONH 用于 P4.4-P4.7，P4CONL 用作 P4.0-P4.3。复位后，P4CONH 和 P4CONL 寄存器清为“00H”，将所有管脚设置为输入模式。用户也可使用控制设置来选择输入或输出模式，使能上拉电阻，选择推挽式输出或开漏输出模式以及使能复用功能。

当对这个端口编程时，确保任何使用 P4 口控制寄存器设置的外设 I/O口复用功能必须在对应的外设模块中使能。

9.2.5.2 P4 口上拉电阻使能控制寄存器 (P4PUR)

通过 P4 口上拉电阻使能控制寄存器 P4PUR(地址：ECH，Set 1，Bank 1)，用户可对 P4 口管脚单独设置上拉电阻。

9.2.5.3 P4 口 N 沟道开漏模式寄存器 (PNE4)

通过 P4 口 N 沟道开漏模式寄存器 PNE4(地址：EDH，Set 1，Bank 1)，用户可对 P4 口管脚单独设置推挽式输出或开漏输出模式。

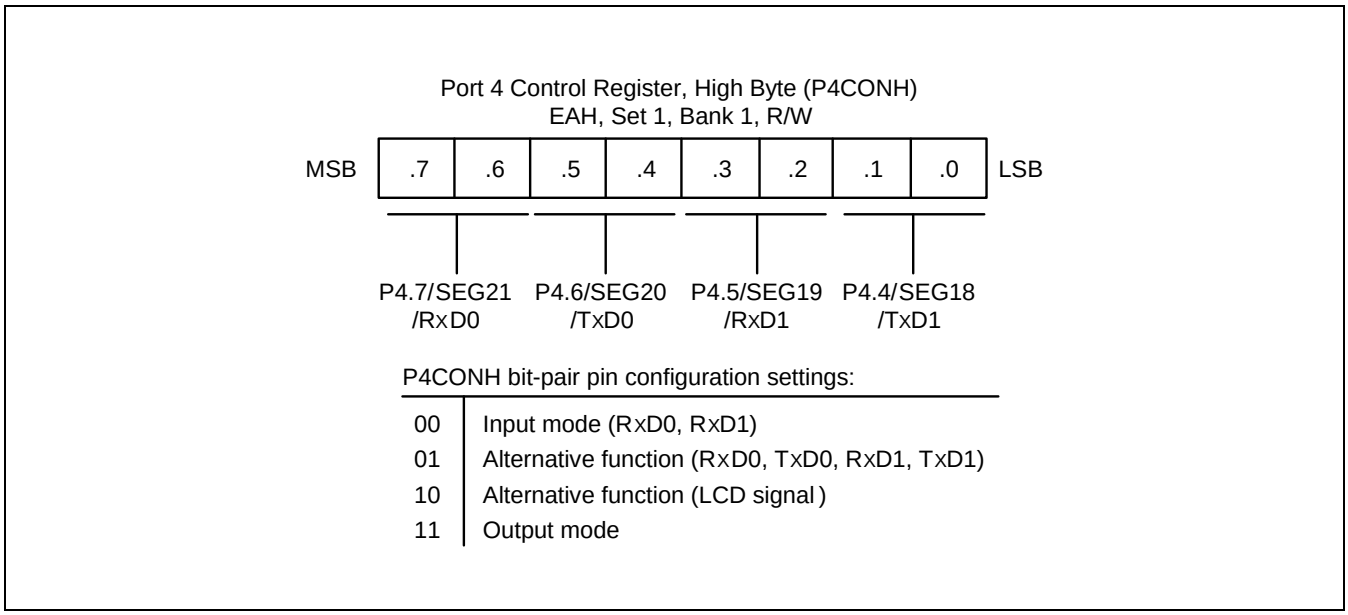
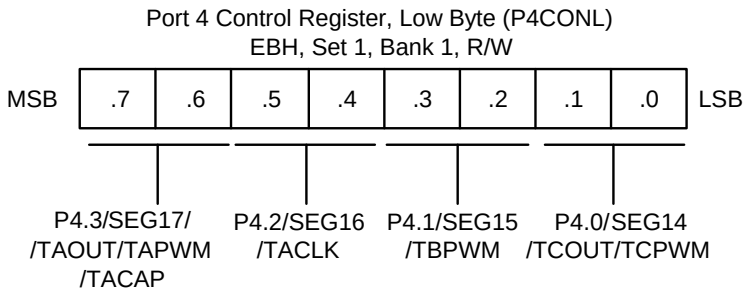


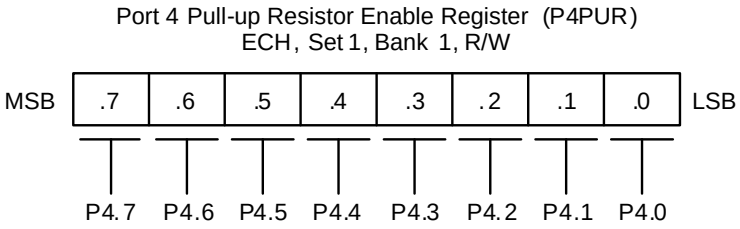
图 9-14 P4 口高字节控制寄存器 (P4CONH)



P4CONL bit-pair pin configuration settings:

00	Input mode (TACAP, TACLK)
01	Alternative function (TAOUT/TAPWM, TBPWM, TCOUT/TCPWM)
10	Alternative function (LCD signal)
11	Output mode

图 9-15 P4 口低字节控制寄存器 (P4CONL)



P4PUR bit configuration settings:

0	Disable Pull-up Resistor
1	Enable Pull-up Resistor

注释： 只有当对应管脚选择作为推挽式输出或复用功能时，P4 口上拉电阻才自动禁止。

图 9-16 P4 口上拉电阻使能控制寄存器 (P4PUR)

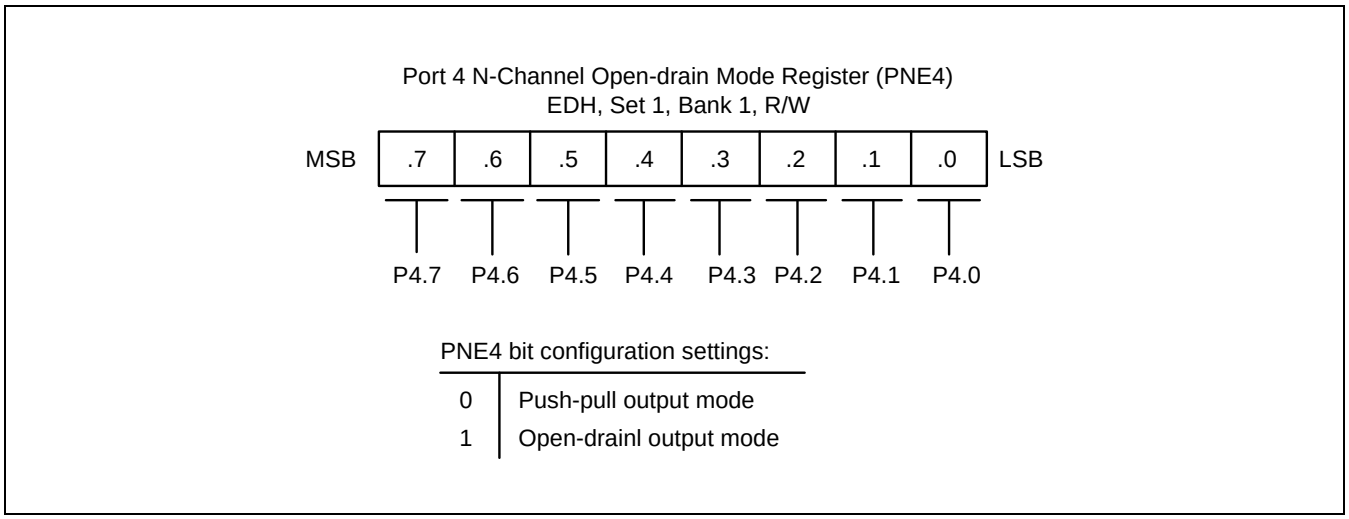


图 9-17 P4 口 N 沟道开漏模式寄存器 (PNE4)

10 BASIC TIMER

10.1 概述

S3F84UA/F84U8 有一个 8 位 Basic Timer。

10.1.1 BASIC TIMER (BT)

Basic Timer 主要应用在两个方面：

- 当系统出现异常时，Basic Timer 作为看门狗定时器，自动复位芯片
- 在复位或退出 STOP 模式后，用于延时以稳定振荡

Basic Timer 单元包含以下几个部分：

- 带多路复用的时钟分频器 (fosc/4096/1024/128)
- 一个 8 位 Basic Timer 计数器 BTCNT (地址：FDH，Set 1，Bank 0，只读)
- Basic Timer 控制寄存器 BTCON (地址：D3H，Set 1，可读/写)

10.1.1.1 Basic Timer 控制寄存器 (BTCON)

Basic Timer 控制寄存器 BTCON 用于选择输入时钟频率，清除 Basic Timer 的计数器和分频器，以及使能或禁止看门狗时钟功能。它的地址为：D3H，Set 1。可读/写，支持寄存器寻址模式。

复位后，将 BTCON 清至“00H”，将使能看门狗功能，并选择 fxx/4096 作为 Basic Timer 的时钟频率。如果想禁止看门狗功能，用户必须将 Basic Timer 控制寄存器的高四位 BTCON.7–BTCON.4 位设置为“1010B”。

正常工作时，任何时刻都可通过往 BTCON.1 位写“1”来将 此 8 位 Basic Timer 计数器 BTCNT (地址：FDH，Set 1，Bank 0) 清零。写“1”到 BTCON.0 位可将分频器清零。

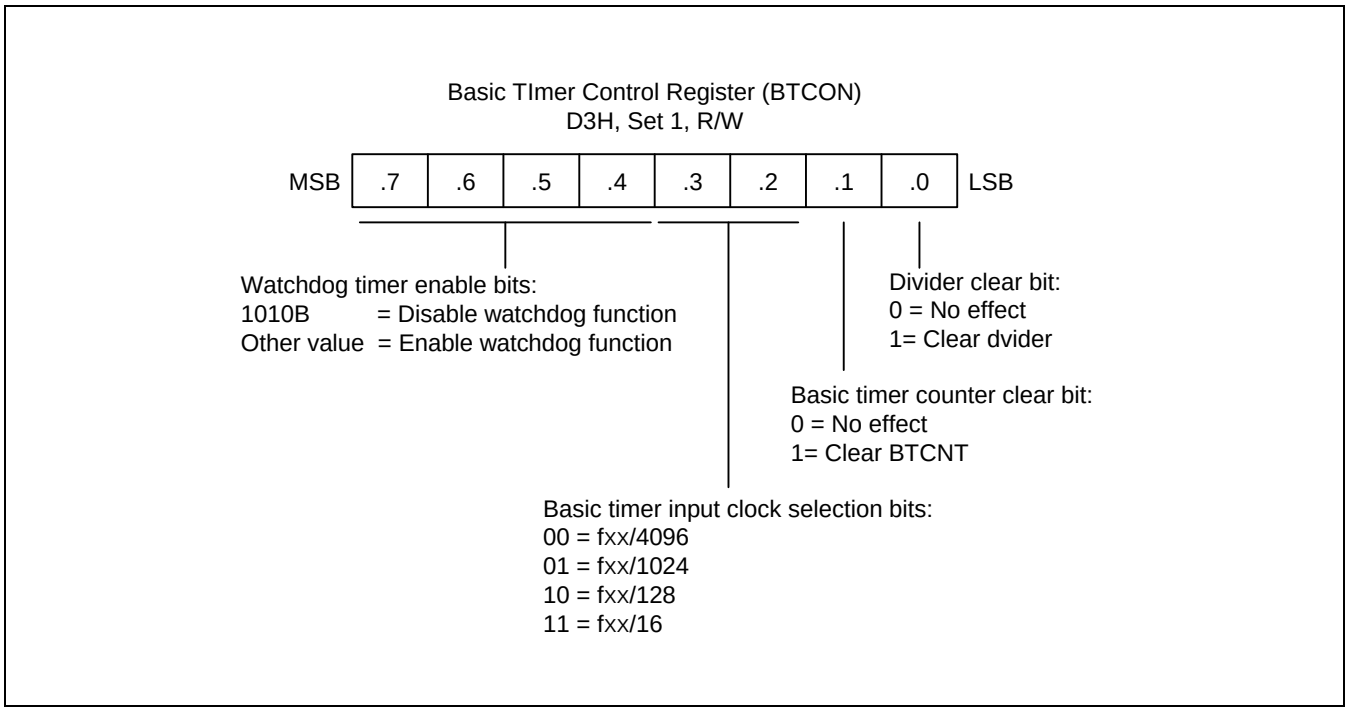


图 10-1 Basic Timer 控制寄存器 (BTCON)

10.1.2 BASIC TIMER 功能描述

10.1.2.1 看门狗功能

通过将 BTCON.7–BTCON.4 位设置为除“1010B”(“1010B”将禁止看门狗功能)外的值，
用户可以对能产生复位的 Basic Timer 溢出信号 (BTOVF) 进行编程。复位时，将 BTCON 清至“00H”，
自动使能看门狗功能。同时选择 fxx/4096 作为 CPU 时钟 (取决于当前的 CLKCON 寄存器设置)。

无论什么时候 Basic Timer 计数器溢出，MCU
都将产生复位信号。在正常情况下，应用程序须防止计数器溢出，以及随之而来的复位的发生，为此，每隔一段时间要清除 BTCNT 值 (写“1”到 BTCON.1 位)。

如果由于电路噪声或其它原因造成系统失灵，Basic Timer 计数器的清零操作将不会被执行，Basic Timer
将溢出并启动复位。换言之，在正常运行期间，总是通过 BTCNT 的清零操作来打破 Basic Timer 的溢出循环 (8 位 Basic Timer 计数器 BTCNT 的第 7 位溢出)。如果发生异常，复位自动产生。

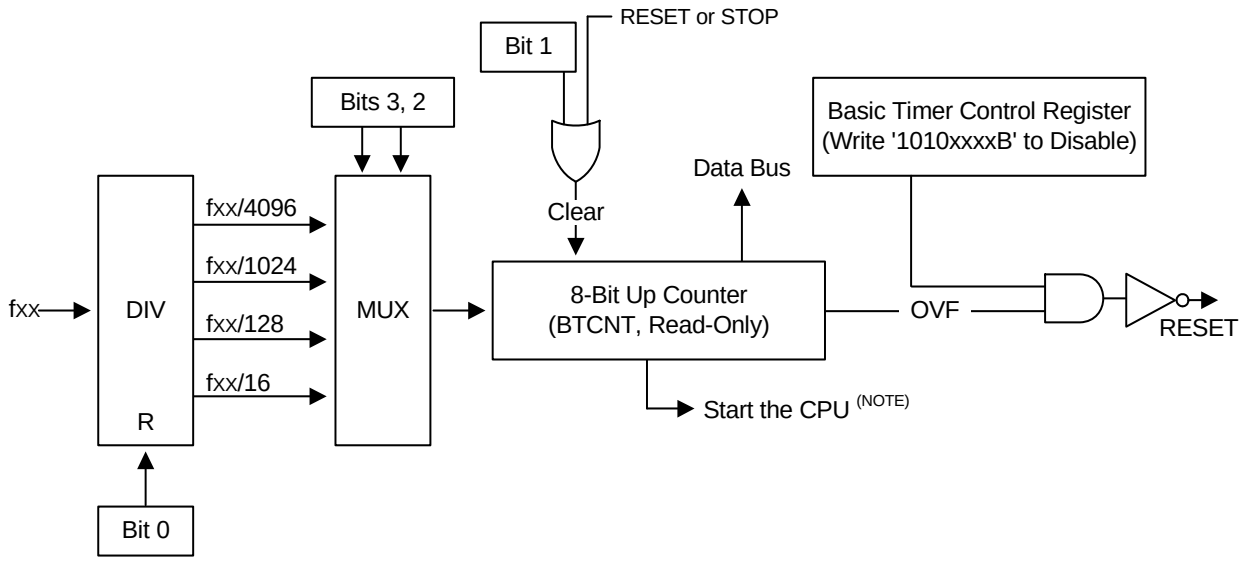
10.1.2.2 振荡器稳定功能

复位或通过外部中断退出 STOP 模式后，用户还可以用 Basic Timer 对时钟振荡器的稳定时间进行编程。

在 STOP 模式下，无论复位或者外部中断发生，振荡电路开始运行。然后 BTCNT 寄存器按照 fxx/4096
的频率(复位时)或者按照预设时钟 (对外部中断) 开始计数。当 BTCNT.4 位溢出，
将产生一个信号说明振荡已经稳定，并且切断 CPU 的时钟信号，目的是让系统恢复正常工作。

总之，当系统从 STOP 模式退出时，以下的事件会依次发生：

1. 上电复位或外部中断产生，系统退出 STOP 状态，振荡器起振。
2. 如果是上电复位操作，Basic Timer 计数器将以 fxx/4096 的时钟频率增计数。
   如果是中断，BTCNT 值就按照预先设置的时钟增计数。
3. 时钟振荡稳定间隔计时开始，一直持续到 Basic Timer 计数器的第 4 位溢出为止。
4. 当 BTCNT.4 位溢出发生时，CPU 恢复正常工作。



注释: 上电复位期间，CPU 在振荡稳定时间间隔到来之前处于 IDLE 状态，直到 Basic Timer 计数器的第 4 位溢出。

图 10-2 Basic Timer 功能框图

11 8 位 TIMER A/B

11.1 8 位 TIMER A

11.1.1 概述

8位 Timer A 是一个 8 位通用 Timer/Counter。它有三种工作模式，用户可通过配置 TACON 来选择其中的一种：

- Interval(定时)模式 (TAOUT 反转输出)
- Capture(捕获)输入模式，TACAP 管脚上升沿或下降沿触发
- PWM 模式 (TAPWM)

Timer A 包括以下几个功能部分：

带多路复用的时钟分频器 (fxx/1024, 256, 64, 8 或 1)

- 外部时钟输入管脚 (TACLK)
- 一个 8 位计数器 (TACNT)，一个 8 位比较器和一个 8 位参考数据寄存器 (TADATA)
- 捕获输入 I/O 管脚 (TACAP) 或 PWM 输出或匹配输出 (TAPWM, TAOUT)
- Timer A 溢出中断 (IRQ0，中断向量地址：D0H) 和匹配/捕获中断 (IRQ0，中断向量地址：CEH) 产生
- Timer A 控制寄存器 TACON (地址：E2H，Set 1，Bank 0，可读/写)

11.1.1.1 Timer A 控制寄存器 (TACON)

Timer A 控制寄存器 TACON 可以：

- 选择 Timer A 工作模式 (定时模式，捕获模式或 PWM 模式)
- 选择 Timer A 输入时钟频率
- 将 Timer A 计数器 TACNT 清零
- 使能 Timer A 溢出中断或 Timer A 匹配/捕获中断

TACON 地址为：E2H，Set 1，Bank 0。可读/写，支持寄存器寻址模式。

复位时，将清除 TACON 至“00H”，将设置 Timer A 为普通定时模式，选择 fxx/1024 作为输入时钟频率，并且禁止所有的 Timer A 中断。正常工作时，任何时刻都可通过往 TACON.2 位写“1”来将 Timer A 计数器清零。

Timer A 的溢出中断 (TAOVF) 中断级为 IRQ0，向量地址为 D0H。当 Timer A 溢出中断发生后，CPU 响应此中断，硬件自动清除中断标志位，或者必须通过软件来清零。

往 TACON.1 位写“1”可使能 Timer A 的匹配/捕获中断(IRQ0，中断向量地址：CEH)。应用程序通过查询 INTPND.1 位来检测匹配/捕获中断标志位。当检测到“1”时，Timer A 匹配或捕获中断产生。
执行中断服务程序时，中断标志位必须通过软件发放清除，即写“0”到 Timer A 匹配/捕获中断标志位 INTPND.1。

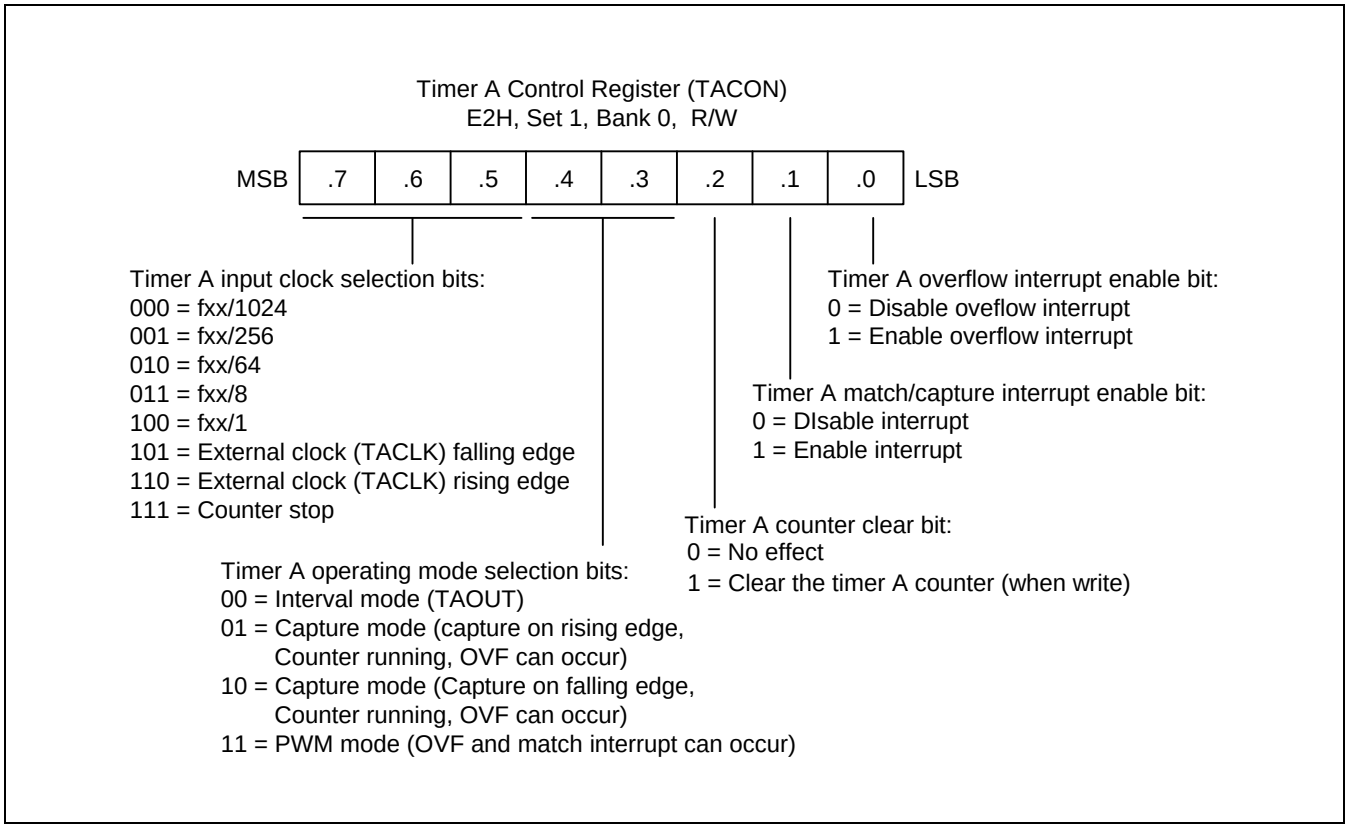


图 11-1 Timer A 控制寄存器 (TACON)

11.1.2 TIMER A 功能描述

11.1.2.1 Timer A 中断 (IRQ0，中断向量地址：CEH 和 D0H)

Timer A 能产生 2 种中断：Timer A 溢出中断 (TAOVF)，和 Timer A 匹配/捕获中断 (TAINT)。TAOVF 的中断优先级为 IRQ0，中断向量地址为 D0H。TAINT 的中断优先级也为 IRQ0，但不同的中断向量地址为 CEH。

Timer A 溢出中断标志在中断响应后，硬件自动清零，或者在中断服务程序中通过往 INTPND.0 中断标志位中写“0”的软件方式进行清除。而 Timer A 匹配/捕获中断标志位必须通过往 INTPND.1 中断标志位中写“0”的软件方式进行清除。

11.1.2.2 Interval (定时) 模式

在定时模式中，当计数器的值与 Timer A 参考数据寄存器 TADATA 写入的值相等时，将产生一个匹配信号。这个匹配信号产生一个 Timer A 的匹配中断 (TAINT，中断向量地址：CEH) 并将计数器清零。

举个例子，如果写“10H”到 TADATA，计数器增计数直到“10H”。此时，Timer A 中断请求产生，计数器的值复位，之后重新计数。每次匹配时，Timer A 输出管脚的信号电平将反转(*图 11-2*)。

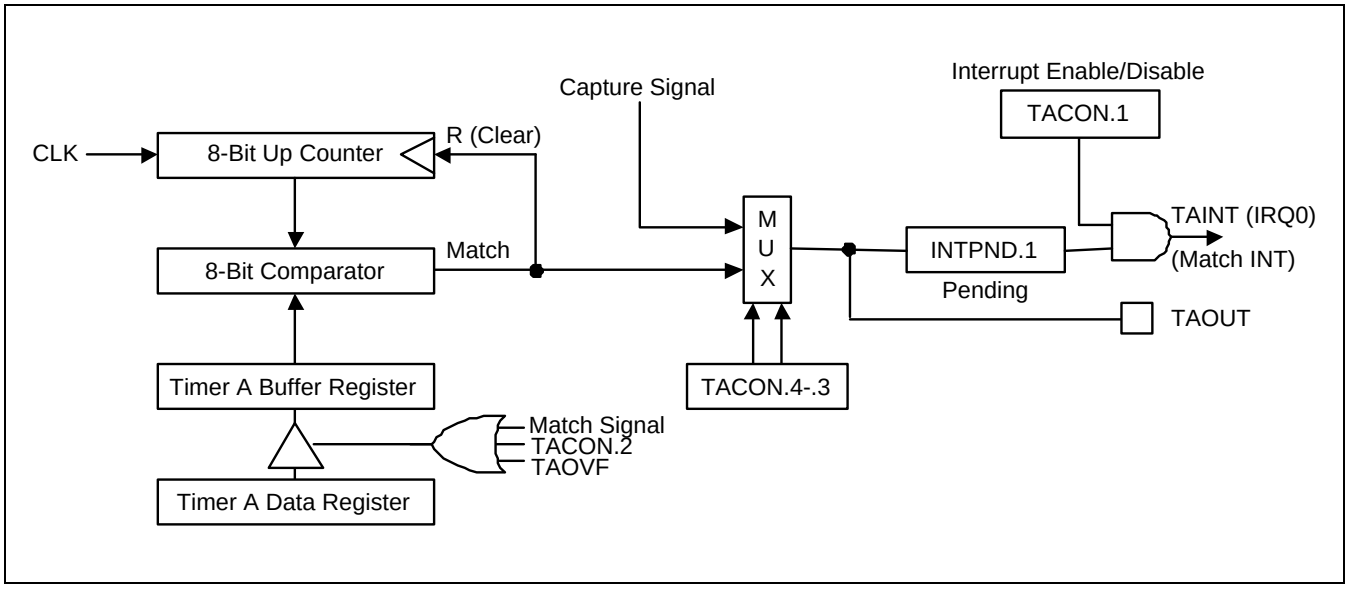


图 11-2 简化的 Timer A 功能图：Interval (定时) 模式

11.1.2.3 PWM 模式

脉冲宽度调制 (PWM) 模式可以让用户编程控制 TAPWM 管脚上输出脉冲宽度(持续时间)。
和定时模式一样，当计数器的值和 Timer A 数据寄存器写入的值一致时，产生一个匹配信号。不同的是，在 PWM 模式中，这个匹配信号不会将计数器清零，而是继续计数直到“FFH”菜溢出，之后从“00H”重新开始计数。

虽然可以使用匹配信号去产生一个 Timer A 的溢出中断，但这些中断在 PWM 类型的应用中并不普遍。相反，只要参考数据值小于或等于(≤)计数器的值时，TAPWM 管脚上保持低电平；而当数据值大于(>)计数器的值时，TAPWM 管脚保持高电平。一个脉冲宽度等于 256 个 t_{CLK} (*图 11-3*)。

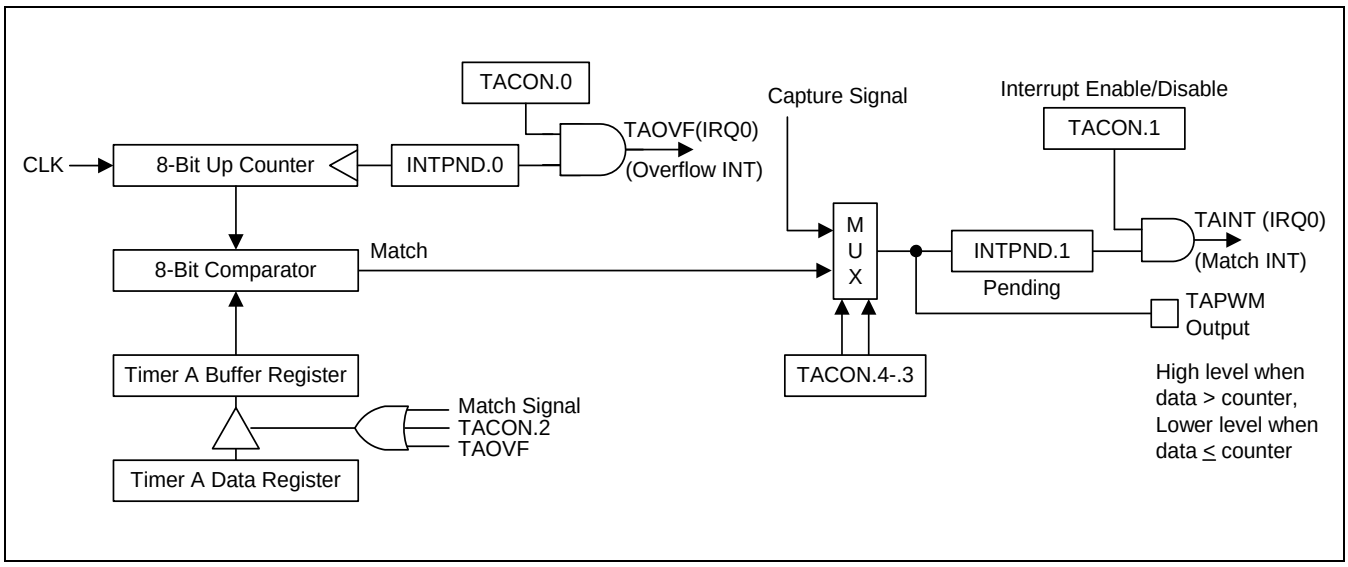


图 11-3 简化的 Timer A 功能图：PWM 模式

11.1.2.4 Capture (捕获) 模式

在捕获模式中，当 TACAP 管脚上检测到一个信号沿时，会把当前计数器的值载入 Timer A数据寄存器中。用户可选择上升沿或者下降沿触发。

Timer A 也提供了捕获输入源：TACAP 管脚上的信号沿。通过设置 P4 控制寄存器 P4CONL（地址：EBH，Set 1，Bank 1）的 P4CONL.7–.6 位 Timer A 捕获输入选择位来选择捕获输入。当 P4CONL.7–.6 位为“00B”时，选择作为 TACAP 输入。

Timer A 的两种中断均可用在捕获模式：当计数器溢出时产生 Timer A 溢出中断；而当计数器值载入到 Timer A 数据寄存器时产生 Timer A 匹配/捕获中断。

通过读取 TADATA 中捕获的数据值，并为 Timer A 假定一个特定的时钟频率，用户就可以计算出 TACAP 管脚上输入信号的脉冲宽度(持续时间)(*图 11-4*)。

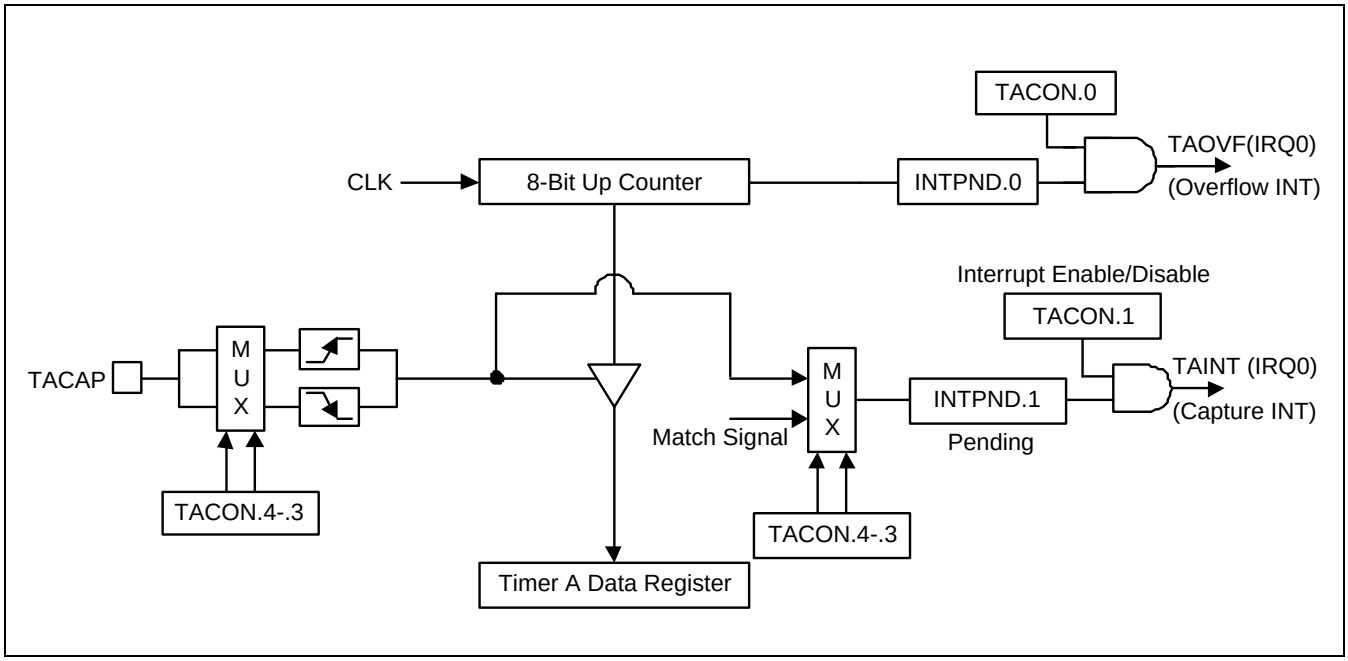


图 11-4 简化的 Timer A 功能图：Capture (捕获) 模式

11.1.3 模块框图

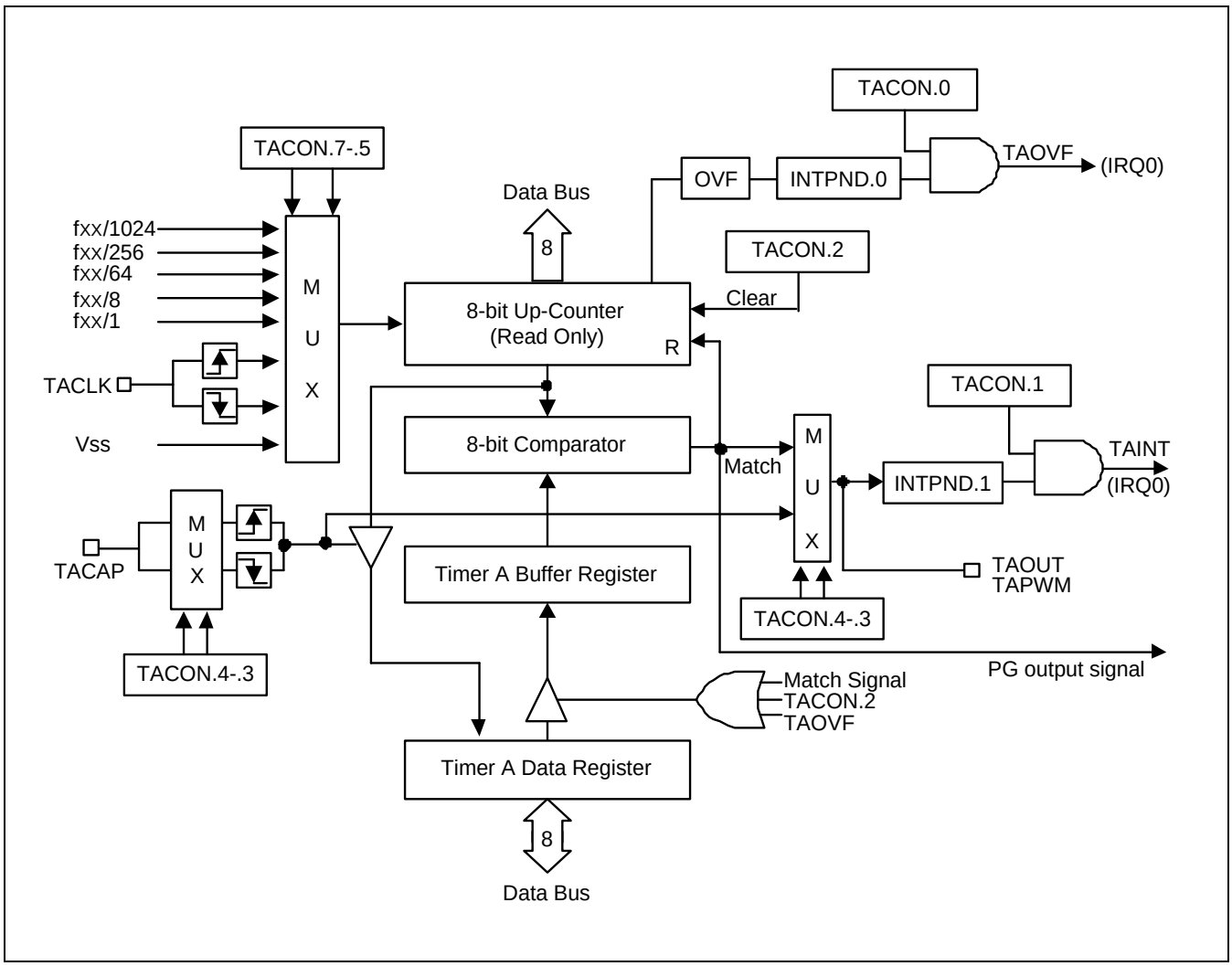


图 11-5 Timer A 功能模块框图

11.2 8 位 TIMER B

11.2.1 概述

S3F84UA/F84U8 MCU 有一个 8 位 Counter 称为 Timer B。可用于产生遥控器信号中的载波频率。

Timer B 有两个功能：

- 普通的定时时钟，在编程设定的时间间隔内产生一个 Timer B 中断
- 提供时钟给 8 位 Timer/Counter 模块 Timer B，目的产生 Timer B 溢出中断

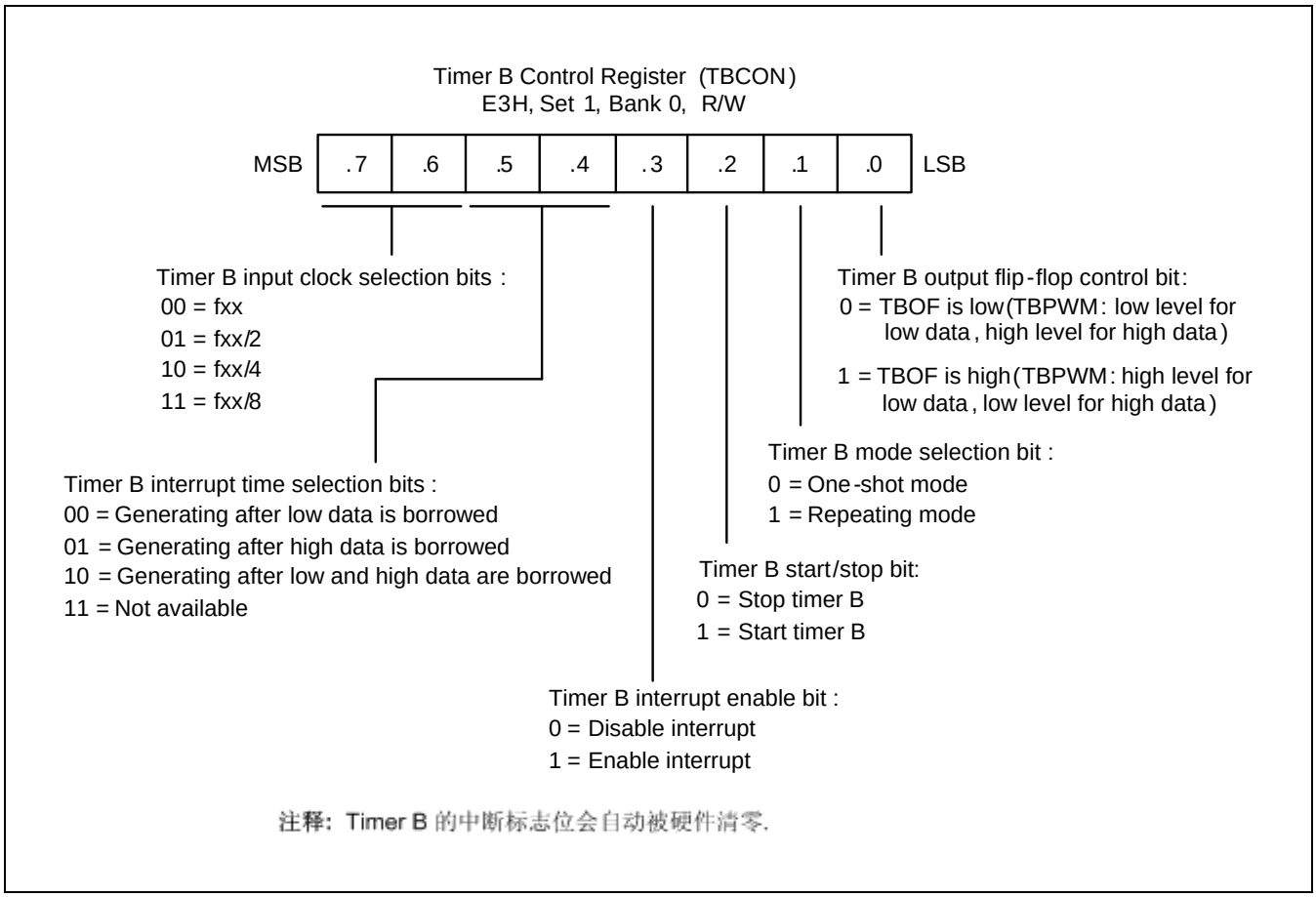
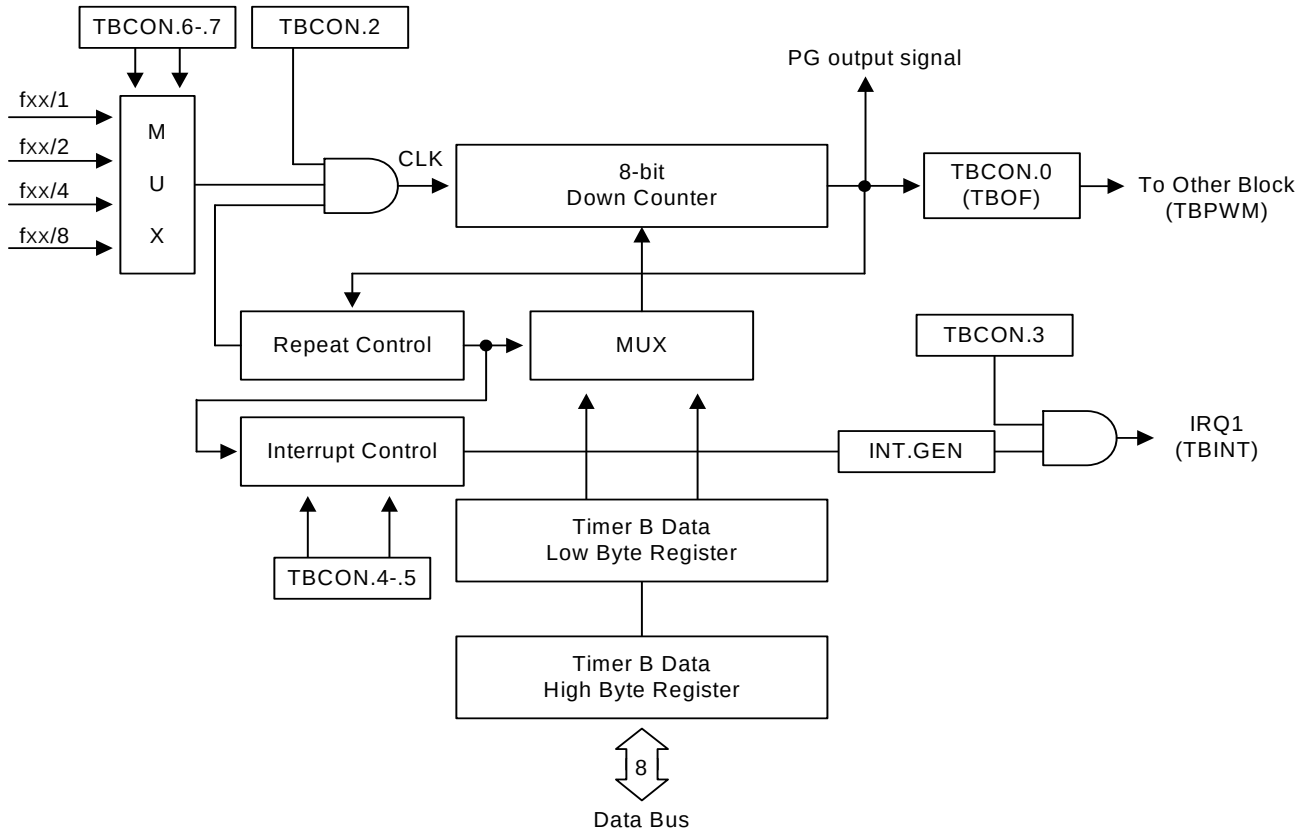


图 11-6 Timer B 控制寄存器

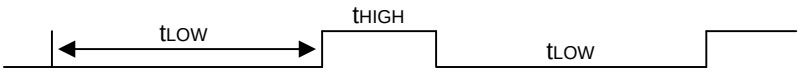
11.2.2 模块框图



注释: 当 Timer B 开始工作时，TBDATAL 寄存器的值载入到 8 位计数器。如果计数器发生借位，TBDATAH 寄存器的值载入到 8 位计数器。但，如果下次借位发生时，TBDATAL 寄存器的值又重新载入到 8 位计数器中。

图 11-7 Timer B 功能模块框图

11.2.2.1 Timer B 脉冲宽度计算



为产生如上所示的重复波形，需要低脉宽时间 t_{LOW} 和高脉宽时间 t_{HIGH}。

当 TBOF = 0 时，

t_{LOW} = (TBDATAL + 2) x 1/fx，0H < TBDATAL < 100H，此时 fx = 已选的时钟。
t_{HIGH} = (TBDATAH + 2) x 1/fx，0H < TBDATAH < 100H，此时 fx = 已选的时钟。

当 TBOF = 1 时，

t_{LOW} = (TBDATAH + 2) x 1/fx，0H < TBDATAH < 100H，此时 fx = 已选的时钟。
t_{HIGH} = (TBDATAL + 2) x 1/fx，0H < TBDATAL < 100H，此时 fx = 已选的时钟。

为获得 t_{LOW} = 24us 以及 t_{HIGH} = 15us。f_{OSC} = 4MHz，fx = 4 MHz/4 = 1MHz

当 TBOF = 0 时，

t_{LOW} = 24us = (TBDATAL + 2) /fx = (TBDATAL + 2) x 1us，TBDATAL = 22。
t_{HIGH} = 15us = (TBDATAH + 2) /fx = (TBDATAH + 2) x 1us，TBDATAH = 13。

当 TBOF = 1 时，

t_{HIGH} = 15us = (TBDATAL + 2) /fx = (TBDATAL + 2) x 1us，TBDATAL = 13。
t_{LOW} = 24us = (TBDATAH + 2) /fx = (TBDATAH + 2) x 1us，TBDATAH = 22。

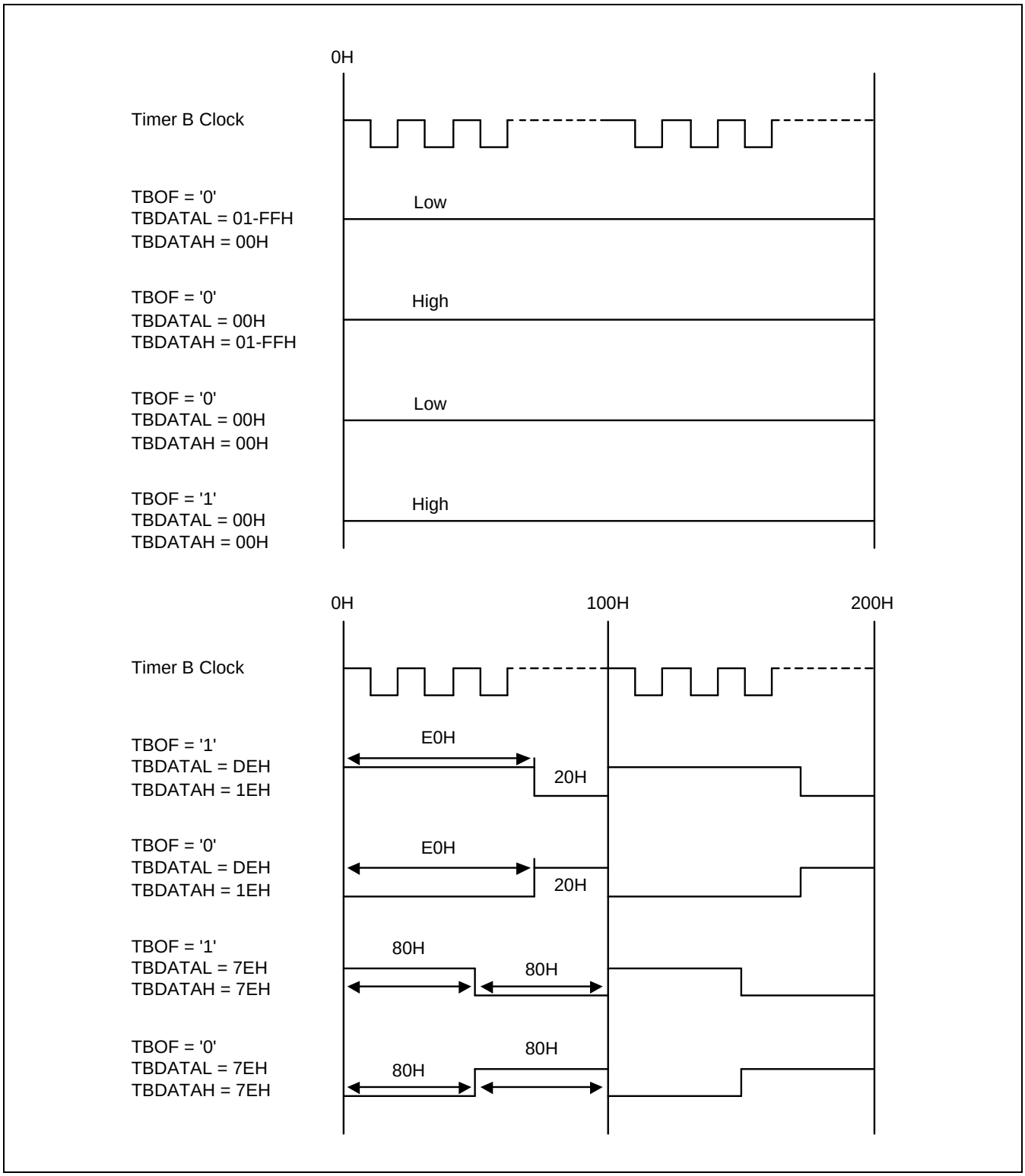
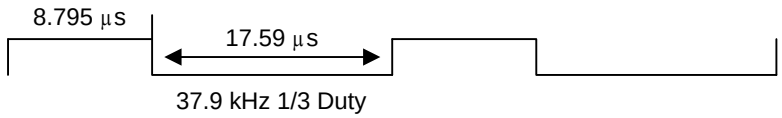


图 11-8 Timer B 在 Repeat (重复) 模式下输出的触发器波形

编程实例 11-1 如何在 P4.1 处产生一个 38 kHz, 1/3 占空比的信号

本例中，设置 Timer B为 Repeat（重复）模式，并且设置系统时钟作为 Timer B 的时钟源，另外设置 TBDATAH 和 TBDATAL 以获得一个 38kHz, 1/3 占空比的载波频率信号。编程参数如下：

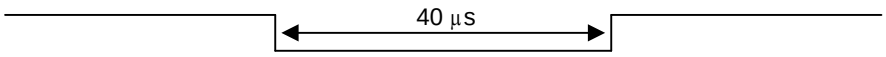


- Timer B 用于 Repeat（重复）模式
- 时钟频率为 4MHz (0.25µs)
- TBDATAH = 8.795µs/0.25µs = 35.18，TBDATAL = 17.59µs/0.25µs = 70.36
- 设置 P4.1 为 TBPWM 模式

```
            ORG    0100H                    ; 复位地址
START       DI
            •
            •
            •
            LD     TBDATAL,#(70-2)          ; 设置 17.5µs
            LD     TBDATAH,#(35-2)          ; 设置 8.75µs
            LD     TBCON,#00000111B         ; 时钟源 ← fxx
                                            ; 禁止 Timer B 中断
                                            ; 选择 Timer B 为 Repeat（重复）模式
                                            ; 启动 Timer B 工作
                                            ; 设置 Timer B 输出触发 (TBOF) 高


            OR     P4CONL,#00000100B        ; 设置 P4.1 为 TBPWM 模式
                                            ; 该指令在 P4.1 处产生一个 38kHz,
                                              1/3 占空比脉冲信号
            •
            •
            •
```

编程实例 11-2 在 P4.1 处产生一个脉冲信号

本例中设置 Timer B 为 One-shot（单次）触发模式，并且设置系统时钟作为 Timer B 时钟源，另外设置 TBDATAH 和 TBDATAL 以获得一个 40μs 宽度的脉冲。编程参数如下：




```
–    Timer B 用于 One-shot（单次）触发模式
–    时钟频率为 4MHz (1 个时钟 = 0.25μs)
–    TBDATAH = 40μs / 0.25μs = 160，TBDATAL = 1
–    设置 P4.1 为 TBPWM 模式

            ORG    0100H                    ; 复位地址
START       DI
            •
            •
            •
            LD     TBDATAH,# (160-2)        ; 设置 40μs
            LD     TBDATAL,# 1              ; 设置为除 00H 外的任何值
            LD     TBCON,#00000001B         ; 时钟源 ← fOSC
                                            ; 禁止 Timer B 中断
                                            ; 设置 Timer B 为 One-shot（单次）触发模式
                                            ; 停止 Timer B 工作
                                            ; 设置 Timer B 输出触发 (TBOF) 高
            OR     P4CONL, #00000100B       ; 设置 P4.1 为 TBPWM 模式
            •
            •
PULSE_OUT:LD       TBCON,#00000101B         ; 启动 Timer B 工作
                                            ; 为在此刻获得该脉冲，
                                            ; 该指令执行后，在脉冲下降沿开始前需要 0.75μs
            •
            •
```

12 8 位 TIMER C

12.1 8 位 TIMER C

12.1.1 概述

8位 Timer C 是一个 8 位通用 Timer/Counter。它有两种工作模式，用户可通过配置 TCCON 来选择其中的一种：

- Interval (定时) 模式 (TCOUT/TCPWM 反转输出)，仅产生匹配中断
- PWM 模式 (TCOUT/TCPWM 管脚)，能产生匹配和溢出中断

Timer C 有以下几个功能部分：

- 带多路复用的时钟分频器
- 一个 8 位计数器，一个 8 位比较器和一个 8 位参考数据寄存器 (TCDATA)
- PWM 或匹配输出 (TCOUT/TCPWM)
- Timer C 匹配/溢出中断 (IRQ2，中断向量地址：D4H) 产生
- Timer C 控制寄存器 TCCON (地址：ECH，Set 1，Bank 0，可读/写)

12.1.1.1 Timer C 控制寄存器 (TCCON)

Timer C 控制寄存器 TCCON 可以：

- 选择 Timer C 工作模式 (fxx/1 & PWM 模式或 fxx/64 & 定时模式)
- 选择 Timer C 3 位预分频
- 将 Timer C 计数器 TCCNT 清零
- 使能 Timer C 匹配/溢出中断
- 启动 Timer C

TCCON 地址为：ECH，Set 1，Bank 0。可读/写，支持寄存器寻址模式。

复位时，将清除 TCCON 至“00H”，设置 Timer C 为 fxx/1 & PWM Timer 模式，将 3 位预分频设置选择不分频，停止 Timer C，并且禁止所有的 Timer C 中断。正常工作时，任何时刻都可以通过往 TCCON.3 位写“1”来将 Timer C 计数器清零。

往 TCCON.7 和 TCCON.1 位写“1”可使能 Timer C 的匹配/溢出中断（IRQ2，中断向量地址：D4H）。为产生精确的定时间隔，应往 TCCON.3 和 .0 位写“1”来清除计数器以及中断标志位。当 TCINT 禁止时，应用程序通过查询标志位 TCCON.0 来检测中断标志。当检测到“1”时，此时有一个 Timer C 的匹配/溢出中断。当 执行 TCINT 的中断服务时，中断标志位必须通过软件的方法清除，即写“0”到 Timer C 中断标志位 TCCON.0。

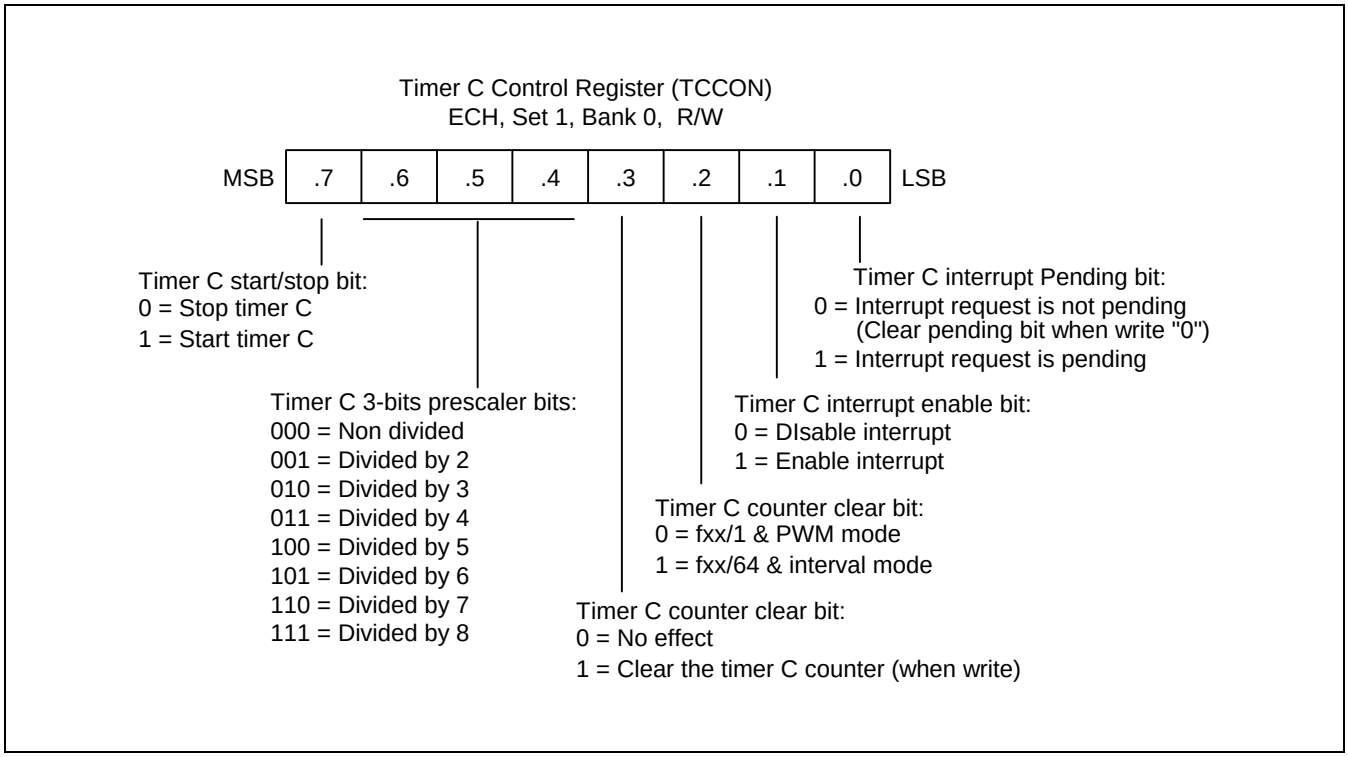
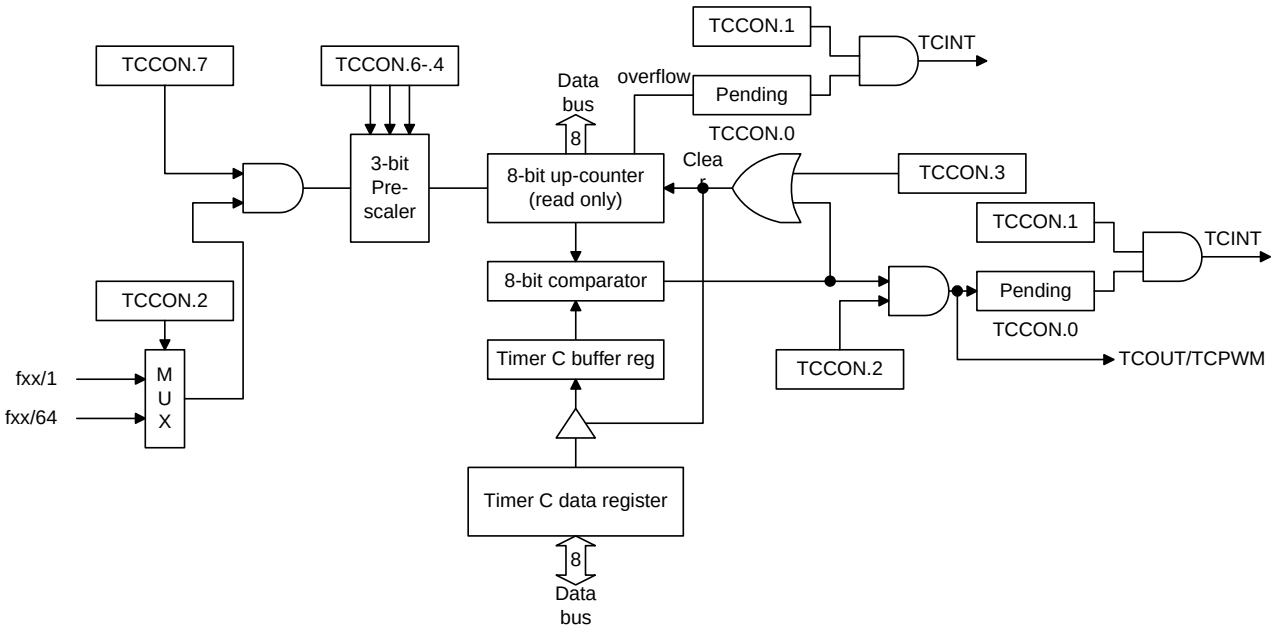


图 12-1 Timer C 控制寄存器 (TCCON)

12.1.2 模块框图



注释： 在 PWM 模式中，匹配信号不能将计数器清零。

图 12-2 Timer C 功能模块框图

13 16 位 TIMER D0/D1

13.1 16 位 TIMER D0

13.1.1 概述

16位 Timer D0 是一个 16 位通用 Timer。Timer D0 有三种工作模式，用户可通过配置 TD0CON 来选择其中的一种：

- Interval (定时) 模式 (TD0OUT 反转输出)
- Capture (捕获) 输入模式，TD0CAP 管脚上升沿或下降沿触发
- PWM 模式 (TD0PWM)；PWM 输出与 TD0OUT 输出管脚复用

Timer D0 包括以下几个功能模块：

- 带多路复用的时钟分频器 (fxx /1024，fxx /256，fxx /64, fxx /8，fxx /1)
- 外部时钟输入管脚 (TD0CLK)
- 一个 16 位计数器 (TD0CNTH/L)，一个 16 位比较器以及一个 16 位参考数据寄存器 (TD0DATAH/L)
- 捕获输入 I/O 管脚 (TD0CAP)，或者匹配输出(TD0OUT)
- Timer D0 溢出中断 (IRQ3，中断向量地址：DAH) 和匹配/捕获中断 (IRQ3，中断向量地址：D8H) 产生
- Timer D0 控制寄存器 TD0CON (地址：FAH，Set 1，Bank 1，可读/写)

13.1.1.1 Timer D0 控制寄存器 (TD0CON)

Timer D0 控制寄存器 TD0CON 可以：

- 选择 Timer D0 工作模式 (定时模式，捕获模式或 PWM 模式)
- 选择 Timer D0 输入时钟频率
- 将 Timer D0 计数器 TD0CNTH/TD0CNTL 清零
- 使能 Timer D0 溢出中断或者 Timer D0 匹配/捕获中断

TD0CON 地址为：FAH，Set 1，Bank 1。可读/写，支持寄存器寻址模式。

复位时，将清除 TD0CON 至“00H”，将设置 Timer D0 为普通定时模式，选择 fxx/1024 作为输入时钟频率，并且禁止所有的 Timer D0 中断。可通过将 TD0CON.7-.5 位设置为“111B”来禁止计数器操作。
正常工作时，任何时刻都可通过往 TD0CON.2 位写“1”来将 Timer D0 计数器清零。

Timer D0 的溢出中断 (TD0OVF) 中断级为 IRQ3，中断向量地址为 DAH。当 Timer D0 溢出中断发生后，CPU 响应此中断 (IRQ3，中断向量地址：DAH)，中断标志位自动被硬件清零或必须由软件清零。

往 TD0CON.1 位写“1”可使能 Timer D0 的匹配/捕获中断 (IRQ3，中断向量地址：D8H)。
为了检测到一个匹配/捕获中断标志条件，应用程序需查询 INTPND.3 位。当检测到“1”时，Timer D0 的匹配或捕获中断产生。执行中断服务程序时，中断标志位必须通过软件的方法清除，即写“0”到 Timer D0 的匹配/捕获中断标志位 INTPND.3。

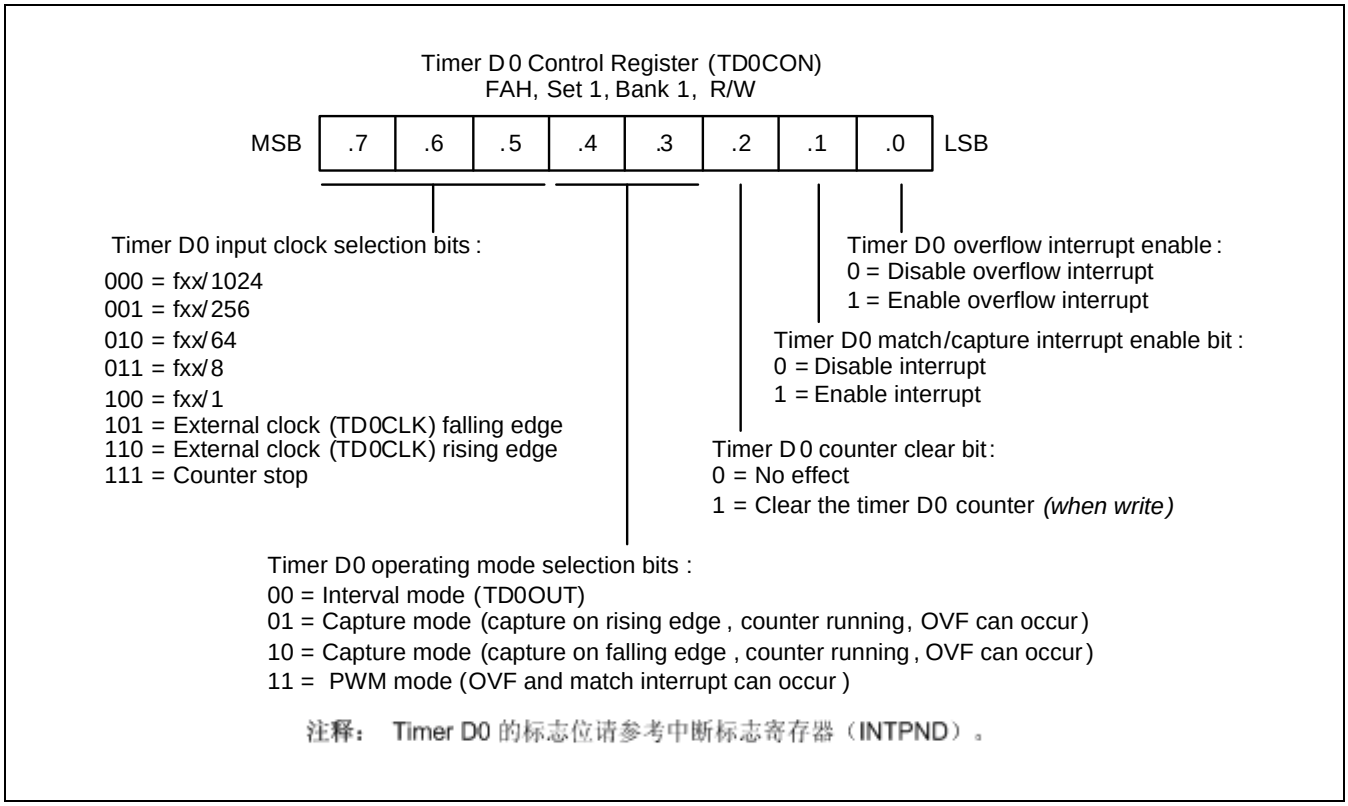


图 13-1 Timer D0 控制寄存器 (TD0CON)

13.1.2 TIMER D0 功能描述

13.1.2.1 Timer D0 中断 (IRQ3，中断向量地址 D8H 和 DAH)

Timer D0 模块能产生 2 种中断：Timer D0 溢出中断 (TD0OVF)，以及 Timer D0 匹配/捕获中断 (TD0INT)。TD0OVF 的中断优先级为 IRQ3，中断向量地址为 DAH。TD0INT 的中断优先级也为 IRQ3，但不同的是中断向量地址为 D8H。

当 Timer D0 溢出中断发生后，CPU 响应此中断，中断标志位自动被硬件清零或必须由软件清零，即在中断服务程序中写“0”到 中断标志位 INTPND.2。但对于 Timer D0 匹配/捕获中断标志位来说，必须通过在应用程序的中断服务程序中写“0”到中断标志位 INTPND.3 来清零。

13.1.2.2 Interval (定时) 模式

在定时模式中，当计数器的值与 Timer D0 参考数据寄存器 TD0DATAH/TD0DATAL 写入的值相等时，将产生一个匹配信号。这个匹配信号产生一个 Timer D0 的 匹配中断(TD0INT，中断向量地址：D8H)并将计数器清零。

举个例子，如果写“1087H”到 TD0DATAH/TD0DATAL，计数器将增计数直到值为“1087H”。当计数器值达到“1087H”时，Timer D0 中断请求产生，计数器的值复位，之后重新计数。每一次匹配时，Timer D0 的输出管脚上的信号电平将反转(*图 13-2*)。

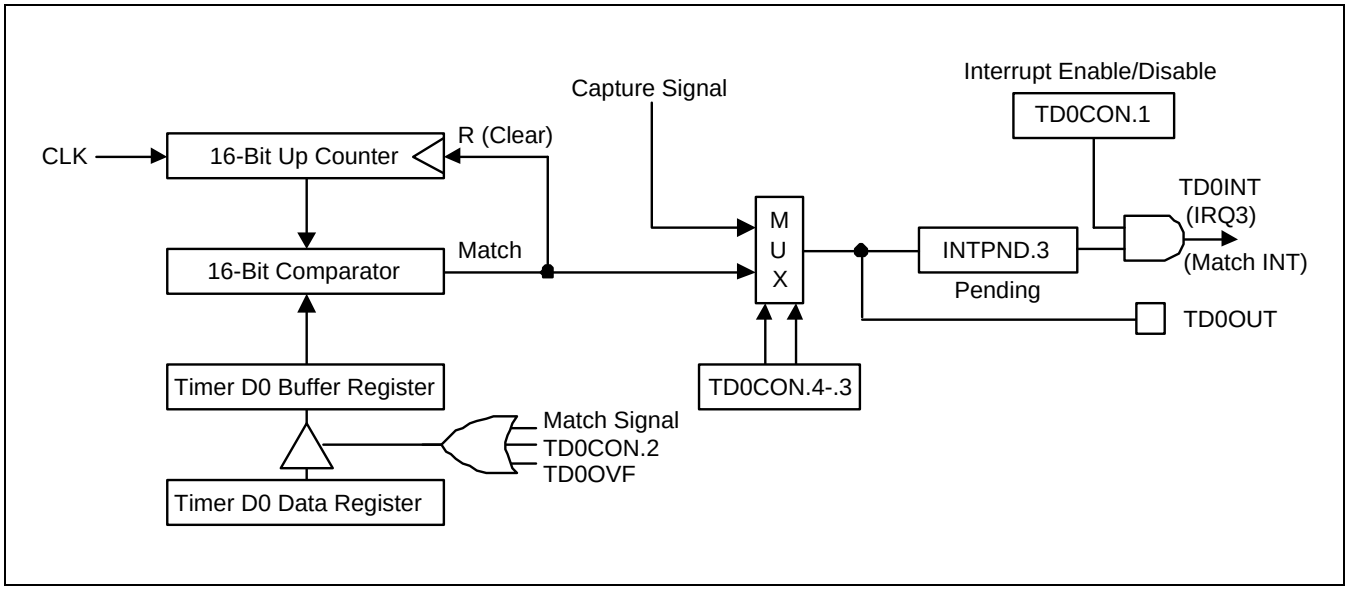


图 13-2 简化的 Timer D0 功能框图：Interval (定时) 模式

13.1.2.3 PWM 模式

脉冲宽度调制 (PWM) 模块可以让用户编程控制 TD0PWM 管脚上输出脉冲宽度(持续时间)。
和定时模式一样，当计数器的值和 Timer D0 数据寄存器写入的值一致时，产生一个匹配信号。不同的是，在 PWM 模式中，这个匹配信号并不将计数器清零，而是继续计数直到“FFFFH”溢出，之后从“0000H”重新开始向上计数。

虽然在 PWM 模式下可以使用 Timer D0 的匹配信号产生一个 Timer D0 溢出中断，但这些中断在 PWM 类型的应用中并不普遍。当参考数据值小于或者等于(≤)计数器值时，TD0PWM 管脚上的脉冲保持低电平；而当参考数据值大于(>)计数器值时，TD0PWM 管脚上的脉冲保持高电平。一个脉冲周期等于 65536 个 t_{CLK} (*图 13-3*)。

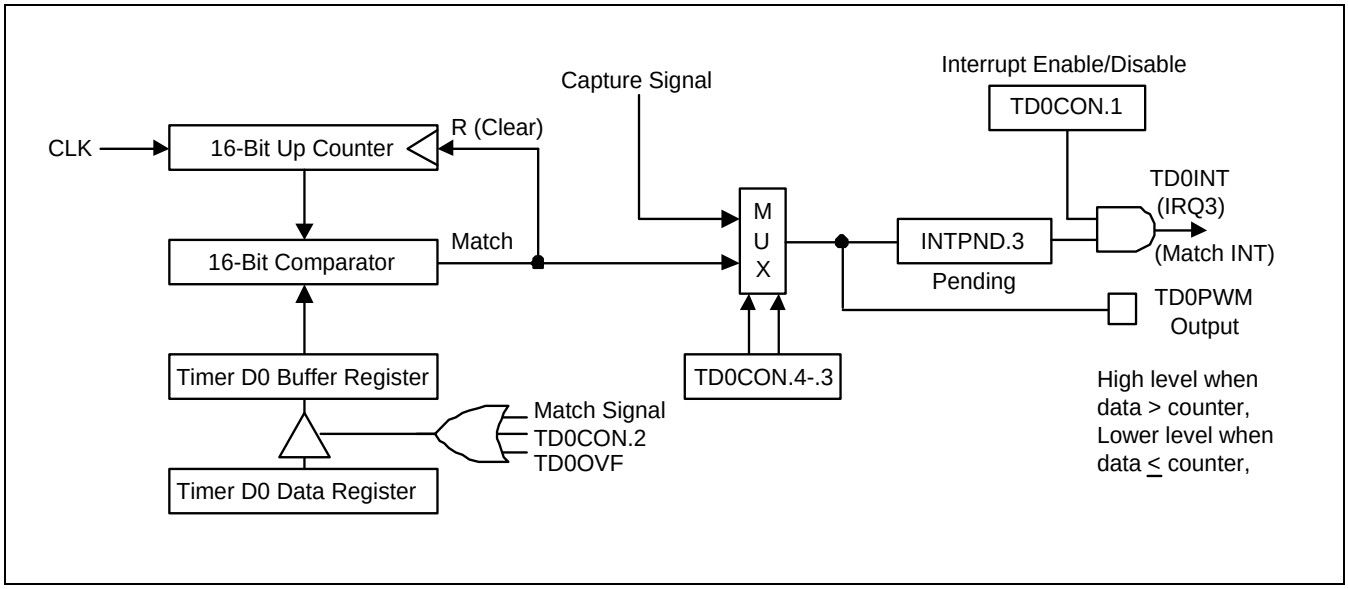


图 13-3 简化的 Timer D0 功能简图: PWM 模式

13.1.2.4 Capture (捕获) 模式

在捕获模式中，当 TD0CAP 管脚上检测到一个信号沿时，把当前计数器的值载入 Timer D0 数据寄存器中。可以选择上升沿或者下降沿触发。

Timer D0 也提供了捕获输入源：TD0CAP 管脚上的信号沿。通过设置 P3 口高字节控制寄存器 P3CONH.7–.6(地址：E4H，Set 1，Bank 1)的 Timer D0 捕获输入选择位来选择捕获输入。当 P3CONH.7–.6 位为“00B”时，选择作为 TD0CAP 输入。

Timer D0 的两种中断均可用在捕获模式：当计数器溢出时产生 Timer D0 溢出中断；而当计数器值载入到 Timer D0 数据寄存器时产生 Timer D0 匹配/捕获中断。

通过读取 TD0DATAH/TD0DATAL 中的捕获数据值，并为 Timer D0 假定一个特定的时钟频率，就可以计算出 TD0CAP 管脚上输入信号的脉冲宽度(持续时间)，*图 13-4*。

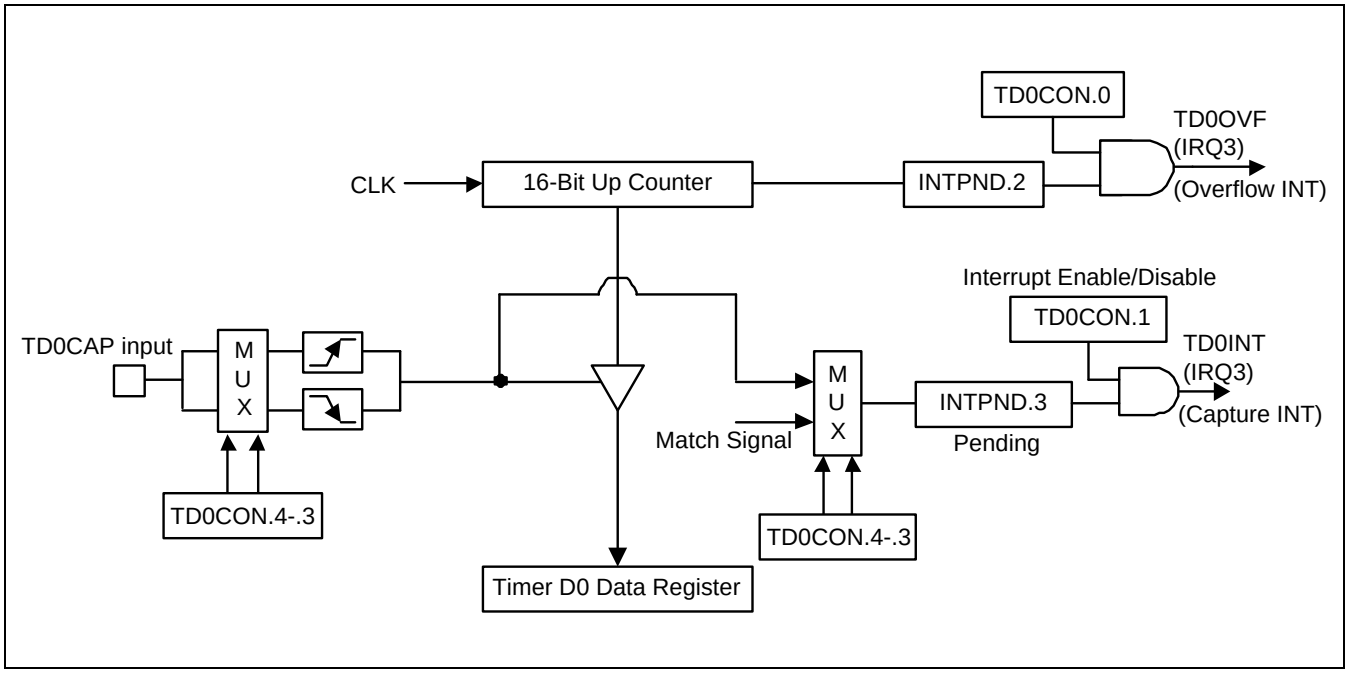


图 13-4 简化的 Timer D0 功能简图：Capture (捕获) 模式

13.1.3 模块框图

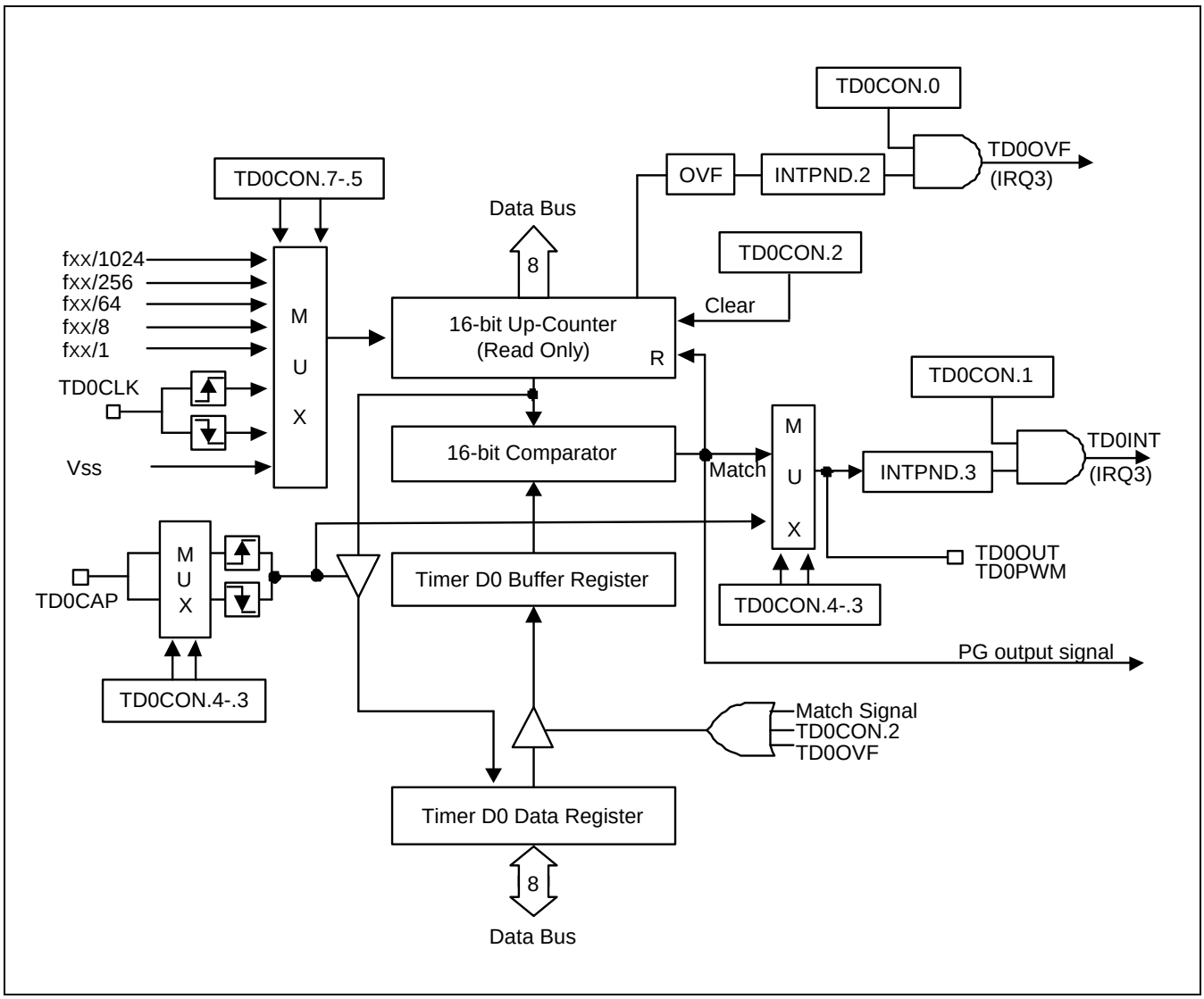


图 13-5 Timer D0 功能模块框图

13.2 16 位 TIMER D1

13.2.1 概述

16位 Timer D1 是一个 16 位通用 Timer。Timer D1 有三种工作模式，用户可通过配置 TD1CON 来选择其中的一种：

- Interval (定时) 模式 (TD1OUT 反转输出)
- Capture (捕获) 输入模式，TD1CAP 管脚上升沿或下降沿触发
- PWM 模式 (TD1PWM)；PWM 输出与 TD1OUT 输出管脚复用

Timer D1 包括以下几个功能模块：

- 带多路复用的时钟分频器 (fxx /1024，fxx /256，fxx /64, fxx /8，fxx /1)
- 外部时钟输入管脚 (TD1CLK)
- 一个 16 位计数器 (TD1CNTH/L)，一个 16 位比较器以及一个 16 位参考数据寄存器 (TD1DATAH/L)
- 捕获输入 I/O 管脚 (TD1CAP)，或者匹配输出(TD1OUT)
- Timer D1 溢出中断 (IRQ3，中断向量地址：DEH) 和匹配/捕获中断 (IRQ3，中断向量地址：DCH) 产生
- Timer D1 控制寄存器 TD1CON (地址：FBH，Set 1，Bank 1，可读/写)

13.2.1.1 Timer D1 控制寄存器 (TD1CON)

Timer D1 控制寄存器 TD1CON 可以：

- 选择 Timer D1 工作模式 (定时器模式，捕获模式或 PWM 模式)
- 选择 Timer D1 输入时钟频率
- 将 Timer D1 计数器 TD1CNTH/TD1CNTL 清零
- 使能 Timer D1 溢出中断或者 Timer D1 匹配/捕获中断

TD1CON 地址为：FBH，Set 1，Bank 1。可读/写，支持寄存器寻址模式。

复位时，将清除 TD1CON 至“00H”，将设置 Timer D1 为普通定时模式，选择 fxx/1024 作为输入时钟频率，并且禁止所有的 Timer D1 中断。可通过将 TD1CON.7-.5 位设置为“111B”来禁止计数器操作。
正常工作时，任何时刻都可通过往 TD1CON.2 位写“1”来将 Timer D1 计数器清零。

Timer D1 的溢出中断 (TD1OVF) 中断级为 IRQ3，中断向量地址为 DEH。当 Timer D1 溢出中断发生后，CPU 响应此中断 (IRQ3，中断向量地址：DEH)，用户必须写“1”到 TD1CON.0 位。当 Timer D1 溢出中断发生后，CPU 响应此中断，中断标志位自动被硬件清零或必须由软件清零。

往 TD1CON.1 位写“1”可使能 Timer D1 的匹配/捕获中断 (IRQ3，中断向量地址：DCH)。
为了检测到一个匹配/捕获中断标志条件，应用程序需查询 INTPND.3 位。当检测到“1”时，Timer D1 的匹配或捕获中断产生。执行中断服务程序时，中断标志位必须通过软件的方法清除，即写“0”到 Timer D1 的匹配/捕获中断标志位 INTPND.5。

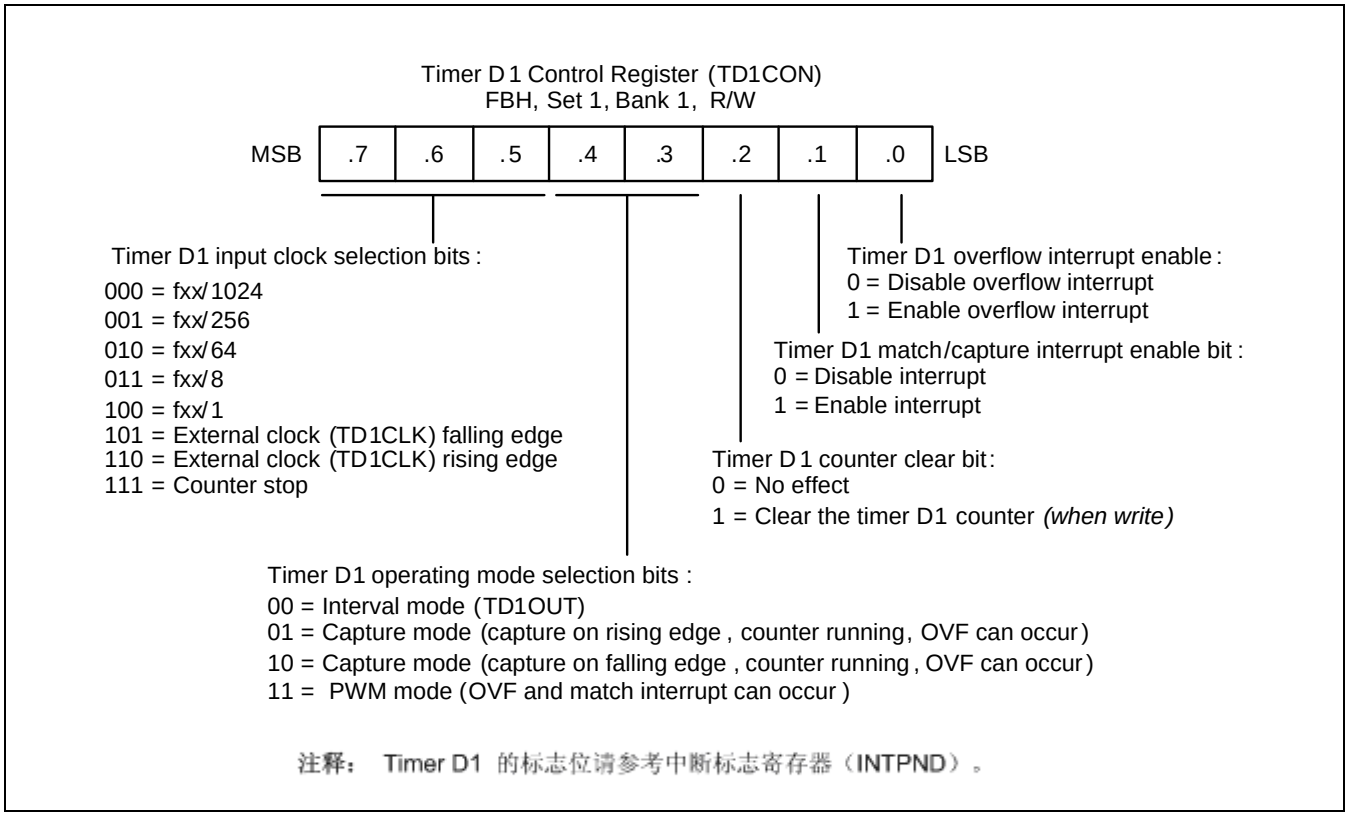


图 13-6 Timer D1 控制寄存器 (TD1CON)

13.2.2 TIMER D1 功能描述

13.2.2.1 Timer D1 中断 (IRQ3，中断向量地址 DCH 和 DEH)

Timer D1 模块能产生 2 种中断：Timer D1 溢出中断 (TD1OVF)，以及 Timer D1 匹配/捕获中断 (TD1INT)。TD1OVF 的中断优先级为 IRQ3，中断向量地址为 DEH。TD1INT 的中断优先级也为 IRQ3，但不同的是中断向量地址为 DCH 。

当 Timer D1 溢出中断发生后，CPU 响应此中断，中断标志位自动被硬件清零或必须由软件清零，即在中断服务程序中写“0”到 中断标志位 INTPND.4。但对于 Timer D1 匹配/捕获中断标志位来说，必须通过在应用程序的中断服务程序中写“0”到中断标志位 INTPND.5 来清零。

13.2.2.2 Interval (定时) 模式

在定时模式中，当计数器的值与 Timer D1 参考数据寄存器 TD1DATAH/TD1DATAL 写入的值相等时，将产生一个匹配信号。这个匹配信号产生一个 Timer D1 的 匹配中断(TD1INT，中断向量地址：DCH)并清将计数器清零。

举个例子，如果写“1087H”到 TD1DATAH/TD1DATAL ，计数器将增计数直到值为“1087H”。当计数器值达到“1087H”时，Timer D1 中断请求产生，计数器的值复位，之后重新计数。每一次匹配时，Timer D1 的输出管脚上的信号电平将反转(*图 13-7*)。

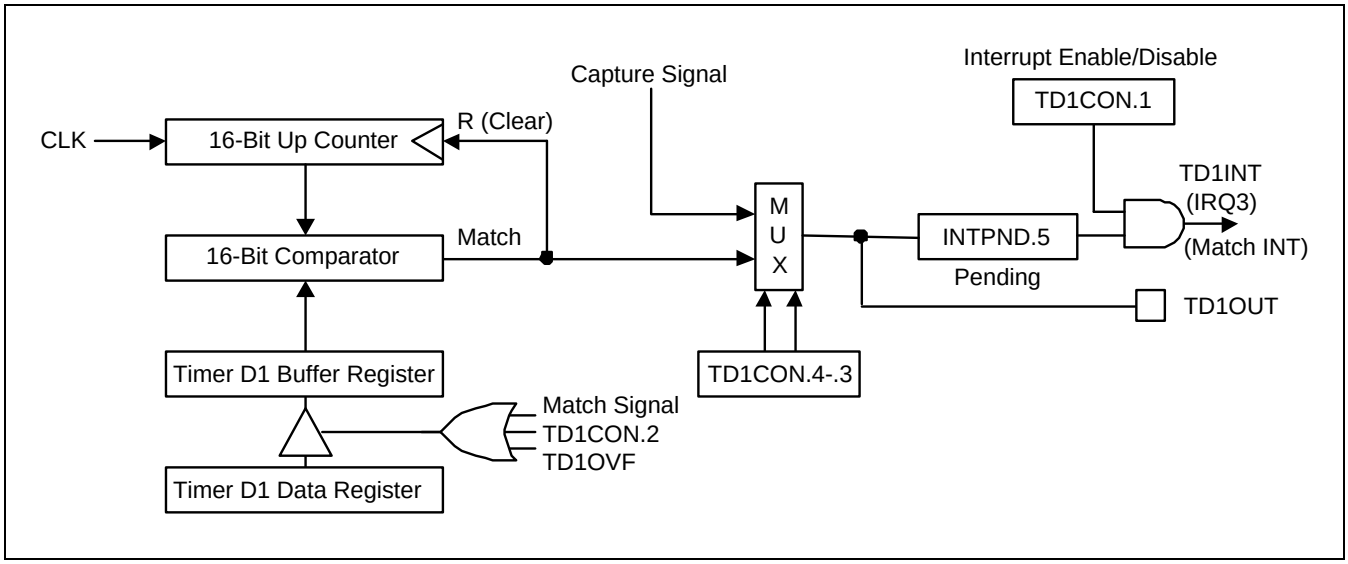


图 13-7 简化的 Timer D1 功能框图：Interval (定时) 模式

13.2.2.3 PWM 模式

脉冲宽度调制 (PWM) 模块可以让用户编程控制 TD1PWM 管脚上输出脉冲宽度(持续时间)。
和定时模式一样，当计数器的值和 Timer D1 数据寄存器写入的值一致时，产生一个匹配信号。不同的是，在 PWM 模式中，这个匹配信号并不将计数器清零，而是继续计数直到“FFFFH”溢出，之后从“0000H”重新开始向上计数。

虽然在 PWM 模式下可以使用 Timer D1 的匹配信号产生一个 Timer D1 溢出中断，但这些中断在 PWM 类型的应用中并不普遍。当参考数据值小于或者等于(≤)计数器值时，TD1PWM 管脚上的脉冲保持低电平；而当参考数据值大于(>)计数器值时，TD1PWM 管脚上的脉冲保持高电平。一个脉冲周期等于 65536 个 t_{CLK} (*图 13-8*)。

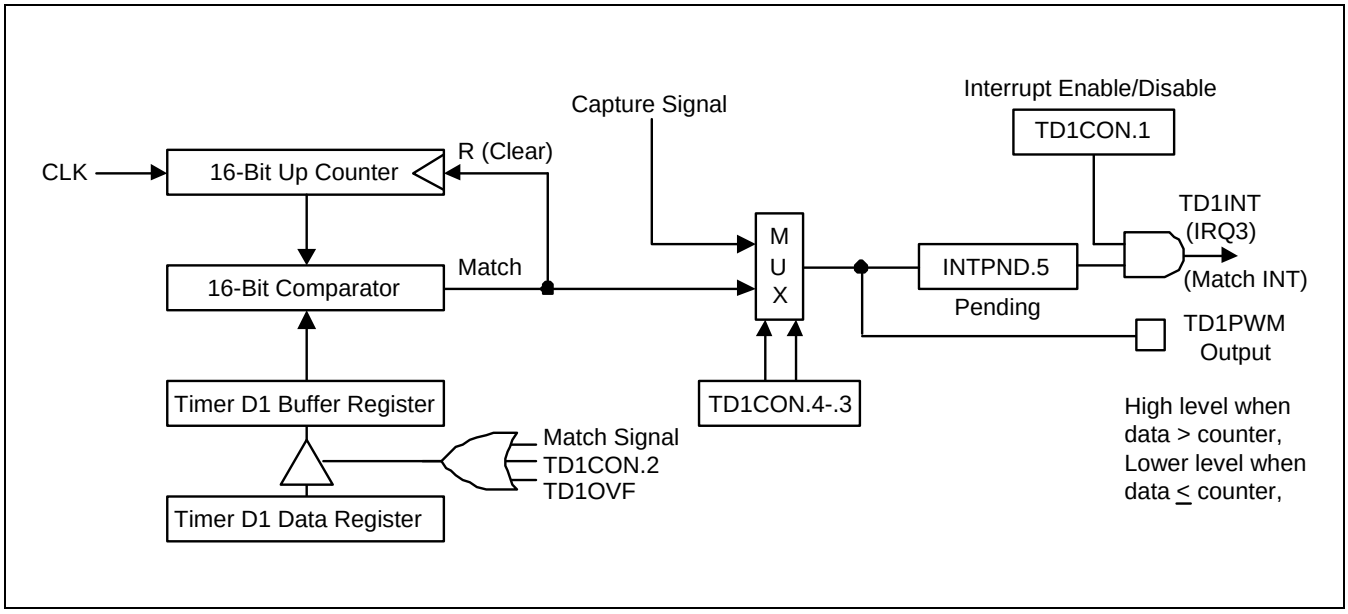


图 13-8 简化的 Timer D1 功能框图：PWM 模式

13.2.2.4 Capture (捕获) 模式

在捕获模式中，当 TD1CAP 管脚上检测到一个信号沿时，把当前计数器的值载入 Timer D1 数据寄存器中。可以选择上升沿或者下降沿触发。

Timer D1 也提供了捕获输入源：TD1CAP 管脚上的信号沿。通过设置 P3 口高字节控制寄存器 P3CONH.3–.2(地址：E4H，Set 1，Bank 1)的 Timer D1 捕获输入选择位来选择捕获输入。当 P3CONH.3–.2 位为“00B”时，选择作为 TD1CAP 输入。

Timer D1 的两种中断均可用在捕获模式：当计数器溢出时产生 Timer D1 溢出中断；而当计数器值载入到 Timer D1 数据寄存器时产生 Timer D1 匹配/捕获中断。

通过读取 TD1DATAH/TD1DATAL 中的捕获数据值，并为 Timer D1 假定一个特定的时钟频率，就可以计算出 TD1CAP 管脚上输入信号的脉冲宽度(持续时间)，*图 13-9*。

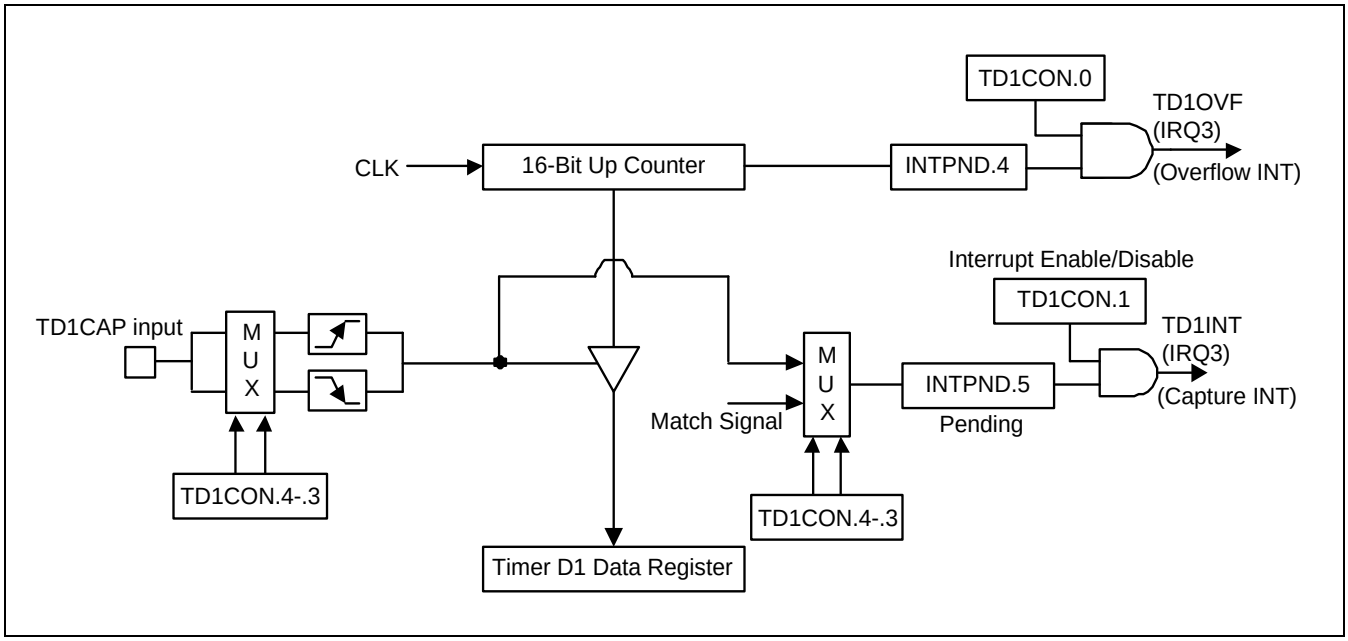


图 13-9 简化的 Timer D1 功能框图：Capture (捕获) 模式

13.2.3 模块框图

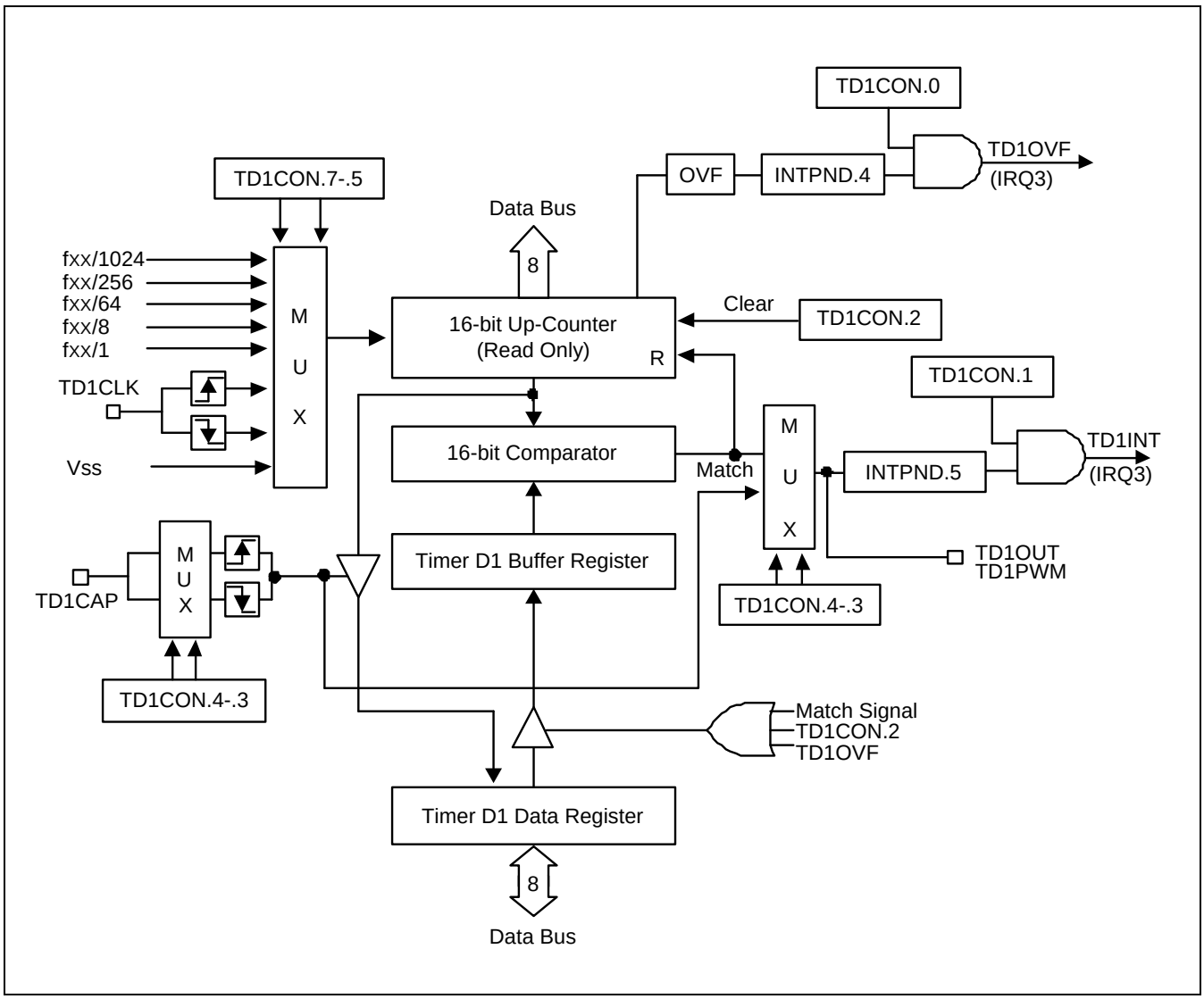


图 13-10 Timer D1 功能模块框图

14 钟表定时器

14.1 概述

钟表定时器的功能包括实时时钟，时间测量以及为系统时钟提供间隔定时。为了启动钟表定时器，将控制寄存器的第 1 位(WTCON.1)设置为“1”。如果希望使用钟表定时器的溢出中断(IRQ4，中断向量地址：E6H)，还要将WTCON.1设置为“1”。

钟表定时器溢出中断标志 (WTCON.0) 必须在应用程序中进行软件清零，也就是向 WTCON.0 写入“0”。

钟表定时器启动之后经过一段时间，中断标志会被自动置“1”，中断请求以 1.955ms，0.125s，0.25s 或 0.5s 的间隔发生，这由钟表定时器的速度设置 (WTCON.3–.2) 决定。

钟表定时器可以产生 稳定的 0.5kHz，1kHz，2kHz 或 4kHz 信号，送到蜂鸣器输出管脚 (BUZ)。将 WTCON.3 和 WTCON.2 设置为“11B”，钟表定时器将工作于高速模式，每隔 1.955ms 产生一次中断。在程序调试时可使用高速模式来对事件定时。

钟表定时器为 LCD 控制器提供时钟频率 (f_{LCD})。因此，如果禁止钟表定时器，LCD 控制器将不能工作。

钟表定时器包含以下组成部分：

- 实时时钟和时间测量
- 时钟源可使用主系统时钟或副系统时钟
- LCD 控制器时钟 (f_{LCD}) 生成
- 蜂鸣器频率输出引脚(BUZ)
- 高速模式下的时序测试
- 钟表定时器溢出中断(IRQ4，中断向量地址：E6H) 发生
- 钟表定时器控制寄存器, WTCON (地址：E6H，Set 1，Bank 0，可读/写)

14.1.1 钟表定时器控制寄存器 (WTCON)

钟表定时器控制寄存器，WTCON，用于选择定时器中断间隔和蜂鸣器信号，使能或禁止钟表定时器功能。它的地址为：E2H，Set 1，Bank 0。可读/写，支持寄存器寻址模式。复位时，将 WTCON 清至“00H”，这将禁用钟表定时器功能。所以，如果想使用钟表定时器，必须对 WTCON 寄存器写入适当的值。

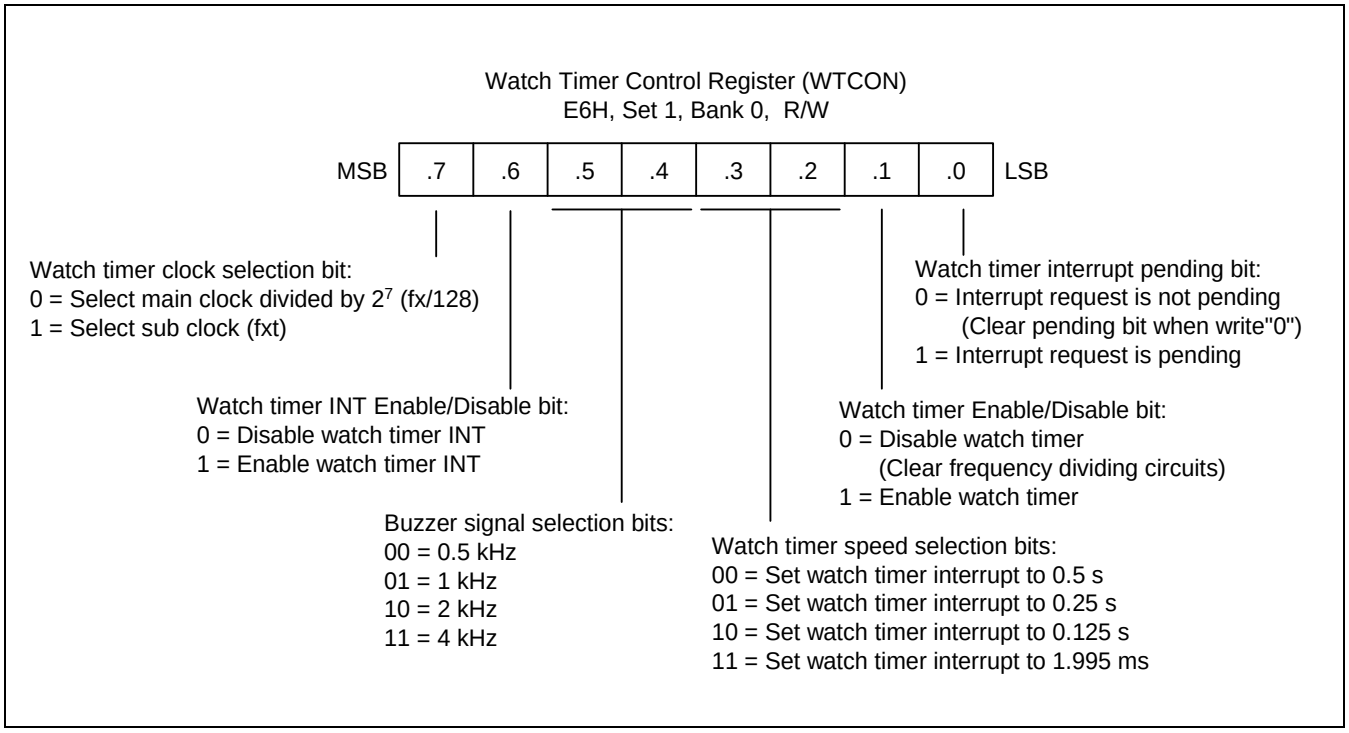


图 14-1 钟表定时器控制寄存器 (WTCON)

14.1.2 钟表定时器电路图

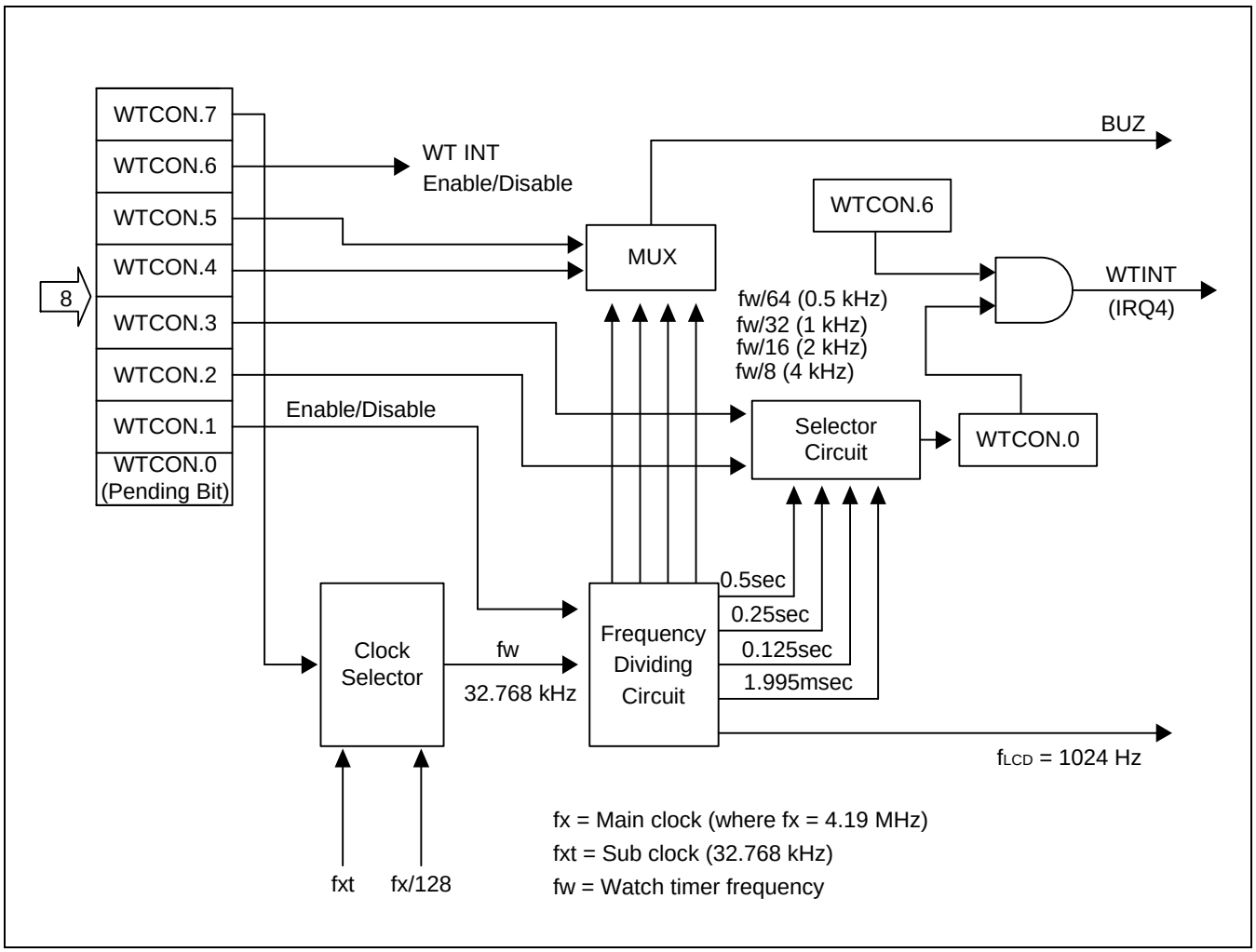


图 14-2 钟表定时器电路框图

15 LCD 控制器/驱动器

15.1 概述

S3F84UA/F84U8 MCU 最多可以直接驱动128 点阵 (16 Segments x 8 Commons) 的 LCD 面板。
LCD 模块包括以下部分：

- LCD 控制器/驱动器
- Page 8 中的显示 RAM(30H–45H) 用于存储显示数据
- 6 个 Common/Segment 输出管脚 (COM2/SEG0–COM7/SEG5)
- 16 个 Segment 输出管脚 (SEG6–SEG21)
- 2 个 Common 输出管脚 (COM0–COM1)
- LCD 内部电阻偏压电路

LCD 控制寄存器 LCON 用来打开和关闭 LCD 显示，选择帧频率，LCD 占空比和偏压比。写到 LCD 显示 RAM 里的数据自动传输到 Segment 信号管脚，不需要任何程序控制。当选择副时钟作为 LCD 时钟源时，即使在主时钟 STOP 或者 IDLE 模式下 LCD 仍可以显示。

写到 LCD 显示 RAM 里的数据自动传输到 Segment 信号管脚，不需要程序控制。

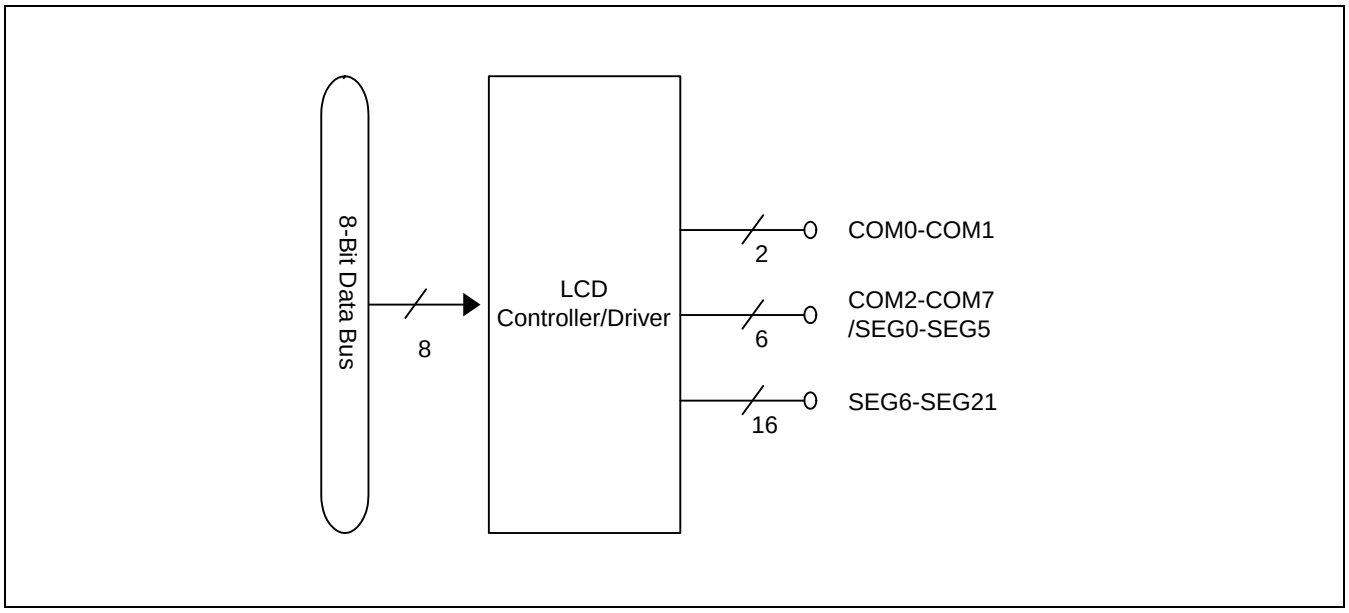


图 15-1 LCD 功能框图

15.1.1 LCD 电路框图

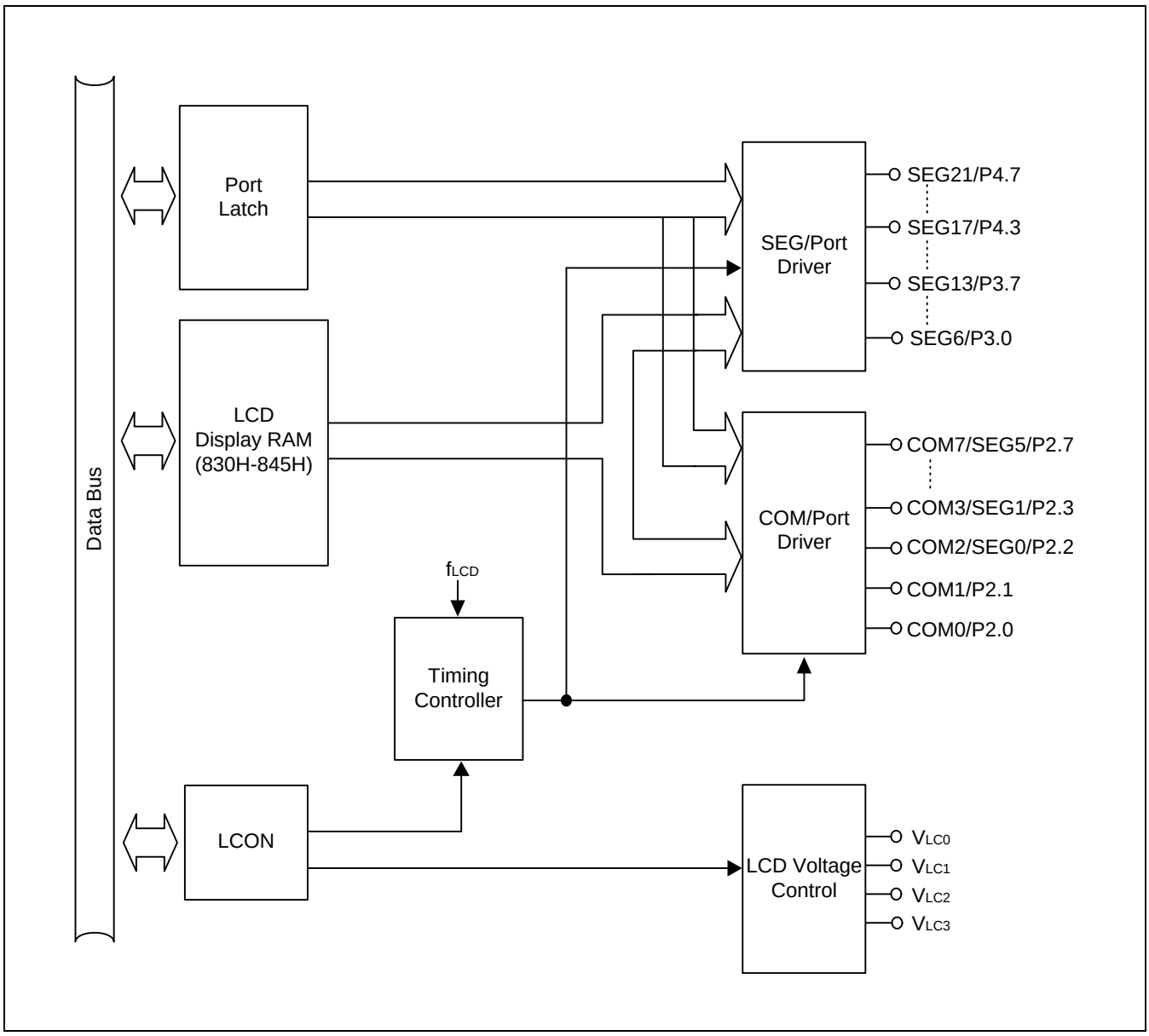


图 15-2 LCD 电路框图

15.1.2 LCD RAM 地址空间

Page 8 中的 RAM 空间 30H - 45H 被用作 LCD 数据存储。这些地址可以由 1 位或者 8 位指令寻址。当某个显示段的值为“1”时，打开 LCD 显示；当某个显示段的值为“0”，关闭 LCD 显示。

显示 RAM 区的数据采用与信号 f_{LCD} 同步的直接内存访问(DMA)方式，通过 Segment 管脚 SEG0-SEG21 送出。该地址空间中未用作 LCD 显示的 RAM 地址可作为一般地址空间使用。

COM	Bit	SEG0	SEG1	SEG2	SEG3	SEG4	------	SEG20	SEG21
COM0 COM1 COM2 COM3 COM4 COM5 COM6 COM7	.0 .1 .2 .3 .4 .5 .6 .7	830H	831H	832H	833H	834H	------	844H	845H

图 15-3 LCD 显示数据 RAM 结构图

15.1.3 LCD 控制寄存器 (LCON)

LCON 地址为：F5H，Set 1，Bank 1。可读/写，支持寄存器寻址模式。它包含以下功能：

- LCD 占空比，偏压比选择
- LCD 时钟选择
- LCD 显示控制

LCON 寄存器可以打开/关闭 LCD 显示，可以选择占空比和偏压比，可以选择 LCD 时钟。复位后，将清除LCON 的值为“00H”，关闭 LCD 显示，选择 1/8 的占空比和 1/4 的偏压比，并选择 128Hz 频率作为 LCD 的时钟。

LCD 时钟信号决定了扫描每个 Segment 输出的 COM 信号频率。也被认为是 LCD 的帧频。既然 LCD 时钟由钟表定时器 (fw) 产生，当 LCD 显示需要打开的时候必须先使能钟表定时器。

注释: 每当改写 LCON 寄存器时，LCD 控制器、驱动器的时钟和占空比都会由硬件自动初始化。所以，不要频繁改写 LCON 寄存器。

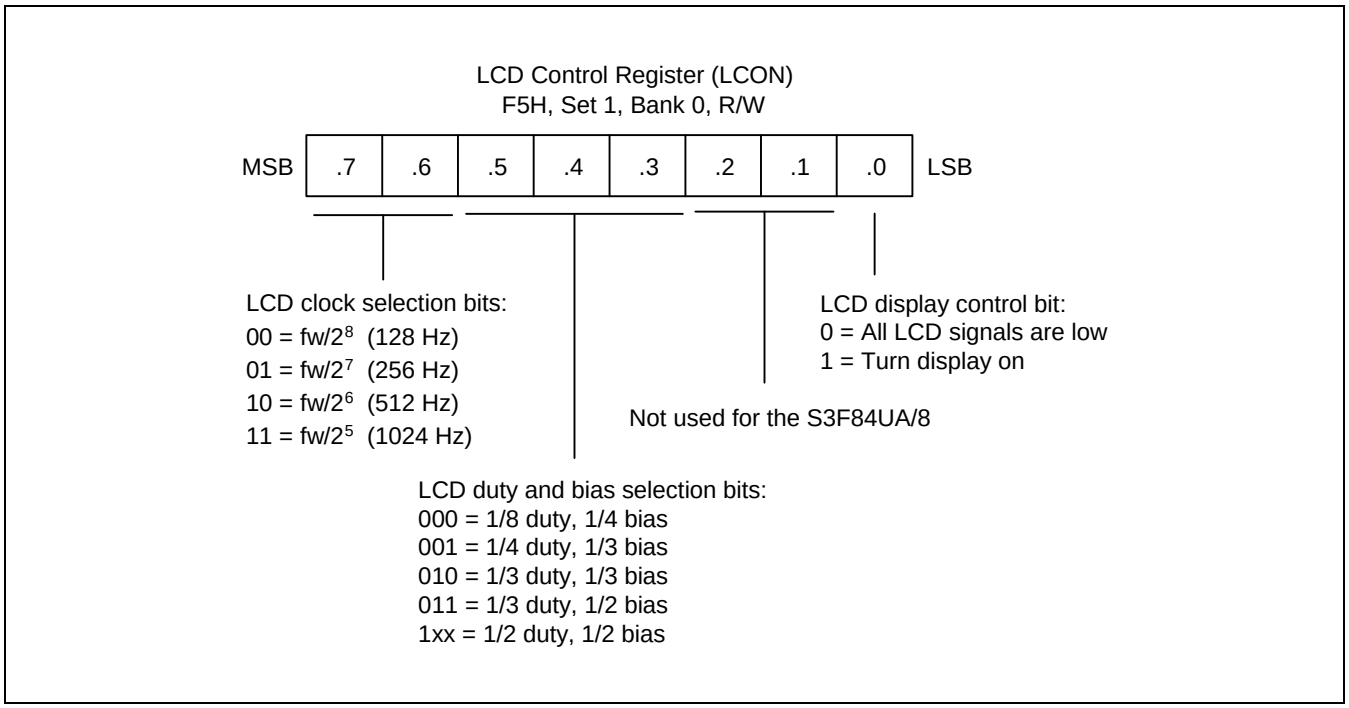
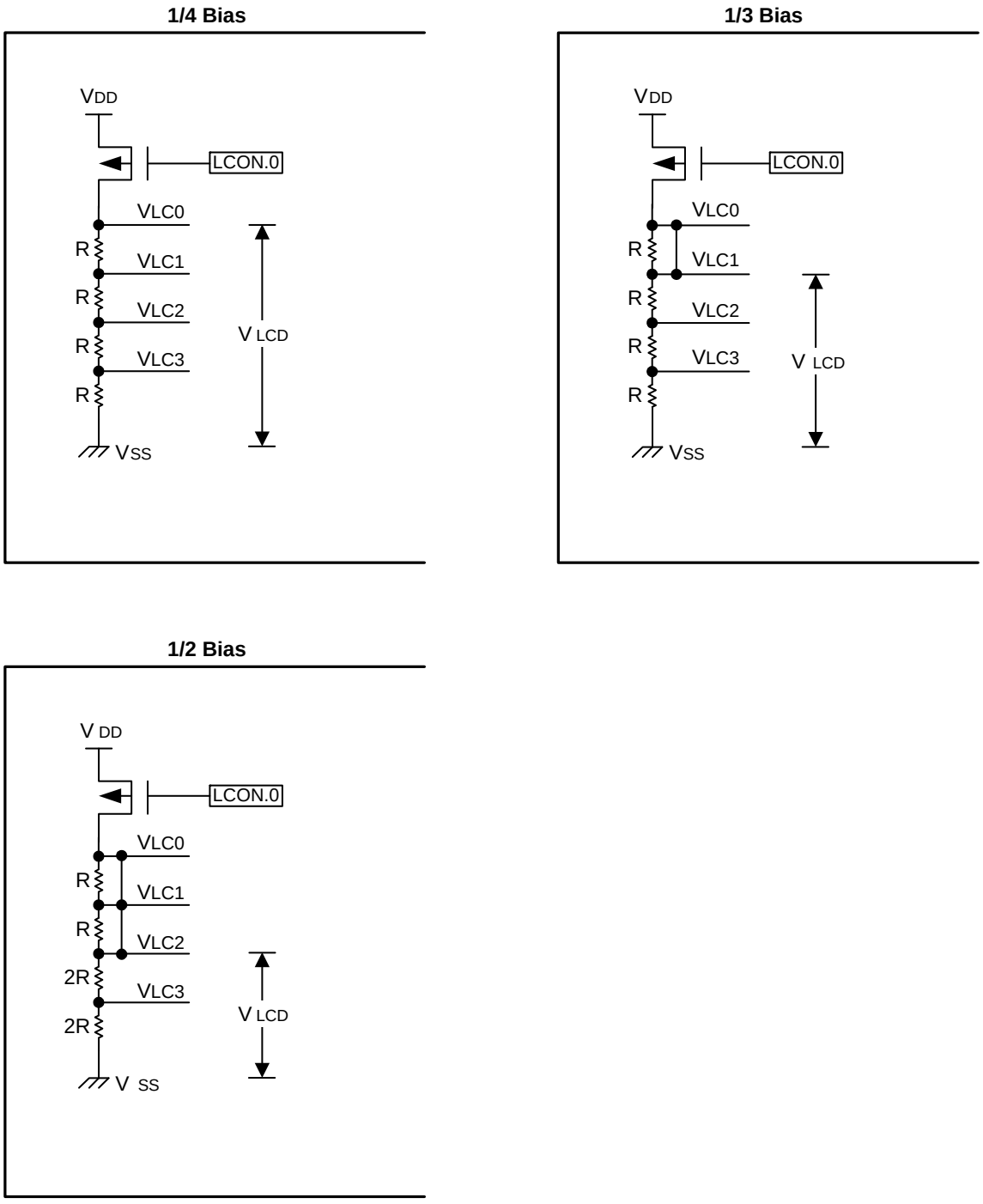


图 15-4 LCD 控制寄存器 (LCON)

15.1.4 内部电阻偏压管脚连接



注释：
1. 选择 1/3 偏压比时， VLC0 和 VLC1 应该短接。
2. 选择 1/2 偏压比时， VLC0， VLC1 和 VLC2 应该短接。

图 15-5 内部电阻偏压管脚连接图

15.1.5 COMMON (COM) 信号

Common 信号输出选择 (COM 管脚选择) 根据所选的占空比：

- 1/8 占空比模式时，选择 COM0-COM7 (SEG6–SEG21) 管脚
- 1/4 占空比模式时，选择 COM0-COM3 (SEG2–SEG21) 管脚
- 1/3 占空比模式时，选择 COM0-COM2 (SEG1–SEG21) 管脚
- 1/2 占空比模式时，选择 COM0-COM1 (SEG0–SEG21) 管脚

15.1.6 SEGMENT (SEG) 信号

22 个 LCD 的 Segment 信号管脚连到 Page 8 中的对应显示 RAM 区。显示 RAM 区的各位与 COM 信号输出管脚同步。

当显示 RAM 区的位为“1”时，所选信号通过对应的 Segment 管脚发送；当显示位为“0”时，将向对应的 Segment 管脚发送一个“未选定”信号。

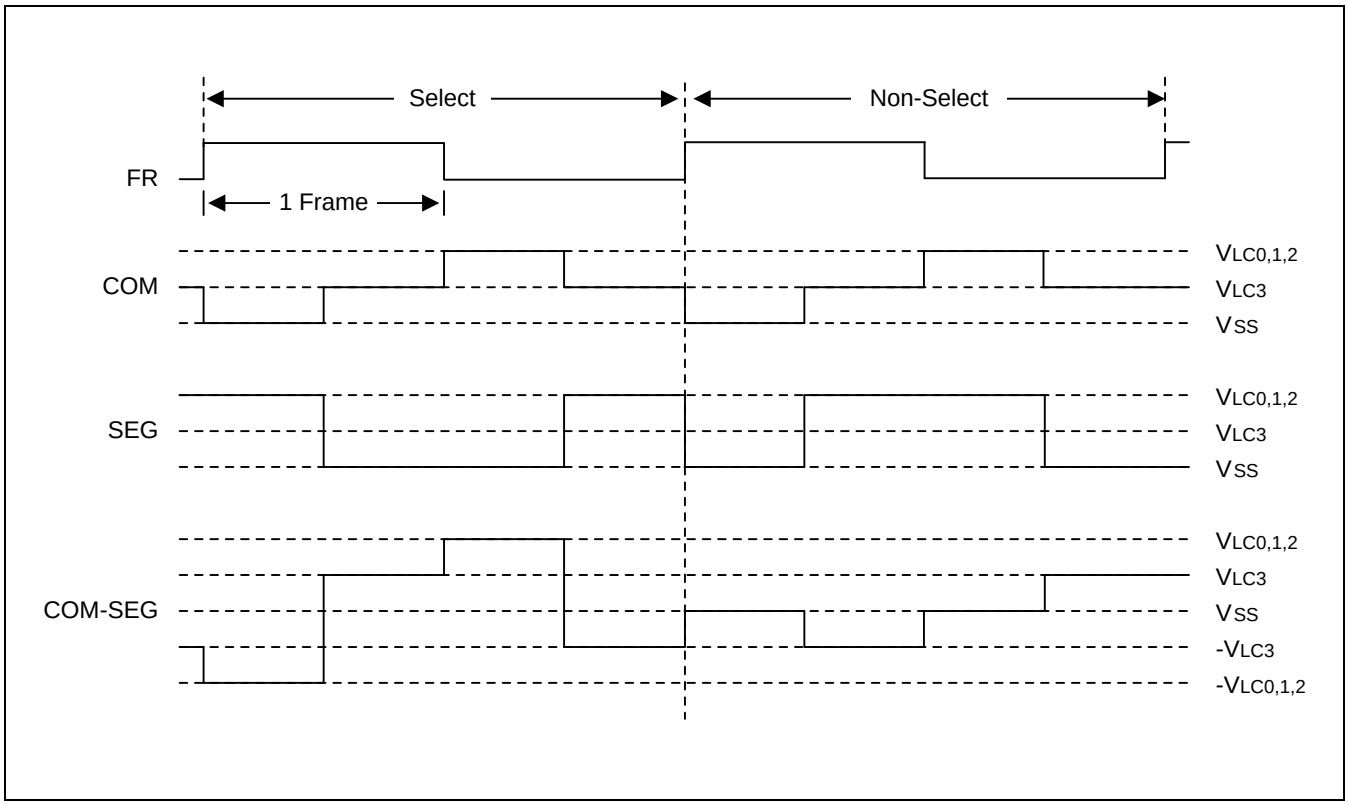


图 15-6 1/2 占空比, 1/2 偏压比显示模式下的选定/未选定信号

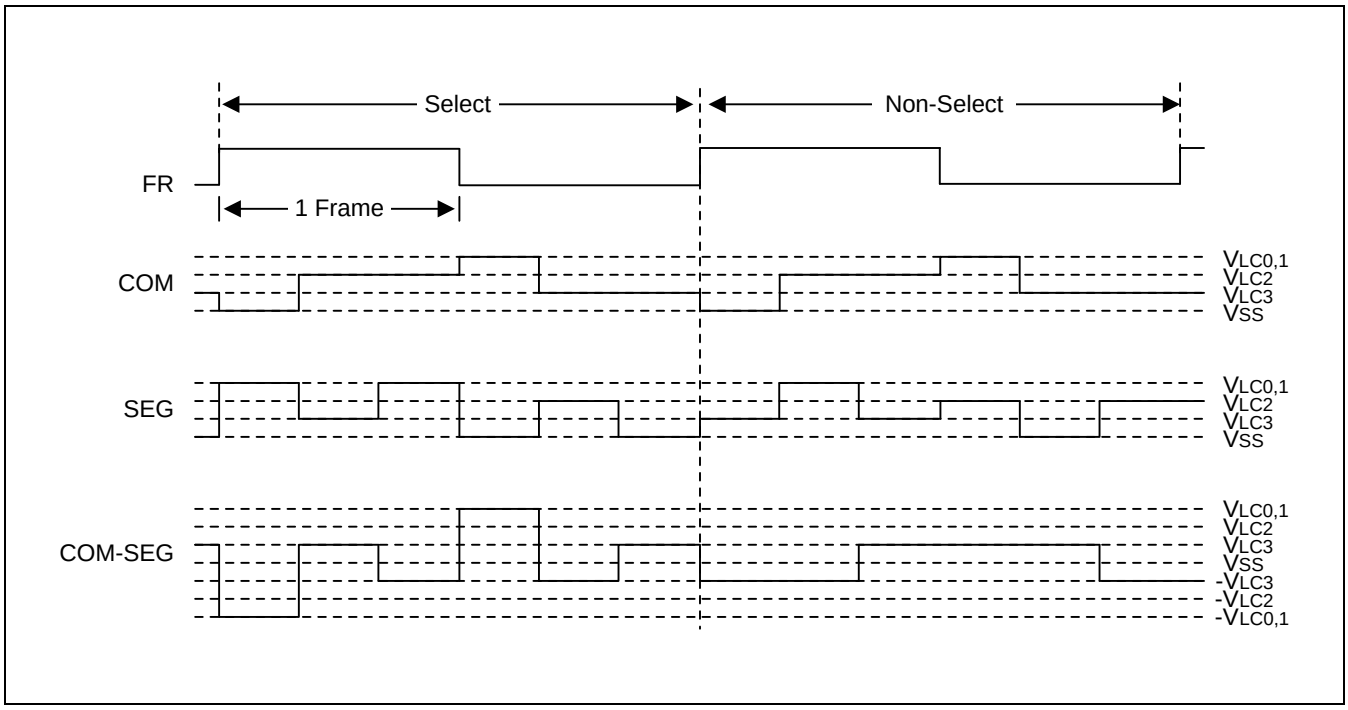


图 15-7 1/3 占空比, 1/3 偏压比显示模式下的选定/未选定信号

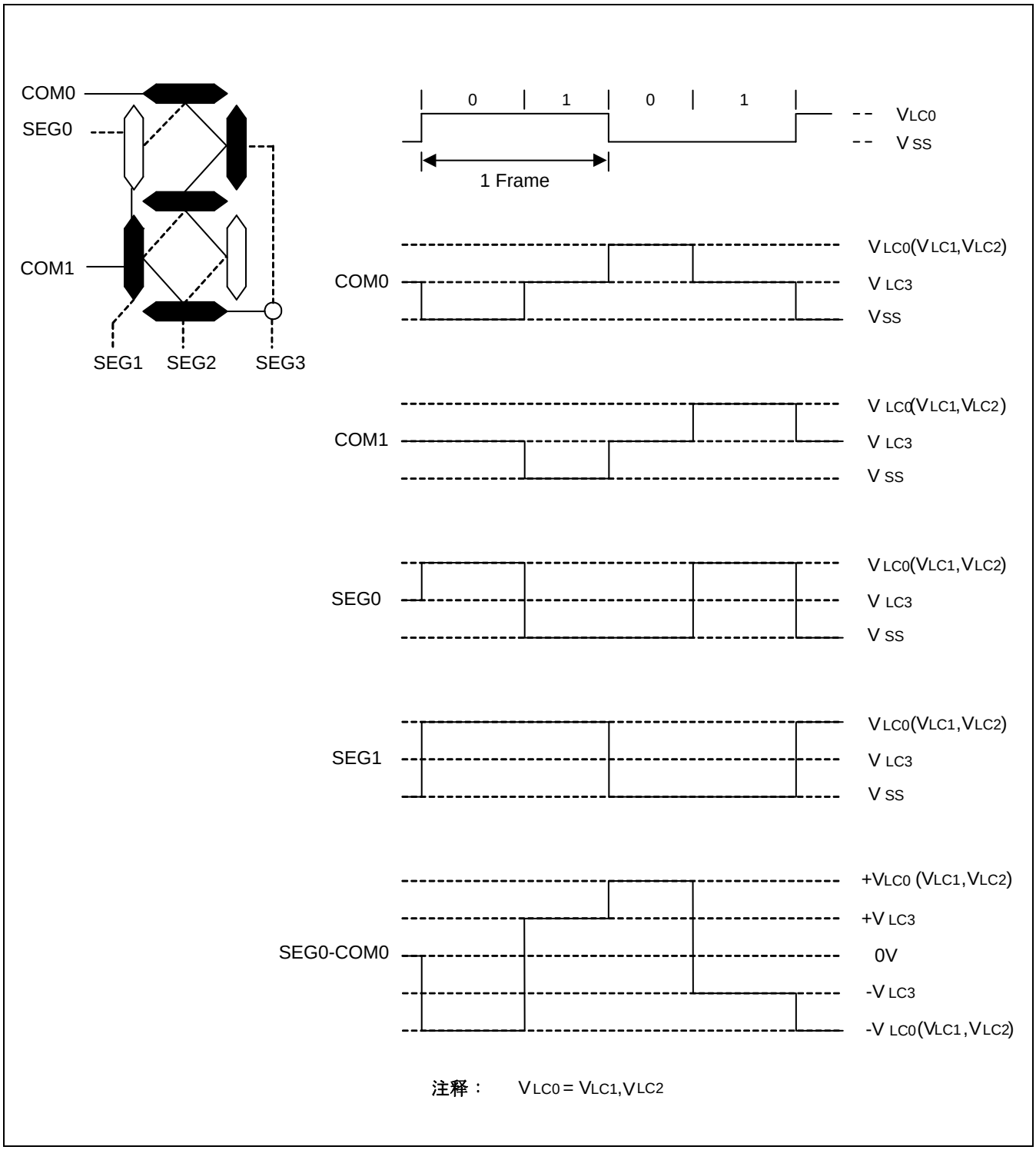


图 15-8 LCD 信号波形 (1/2 占空比, 1/2 偏压比)

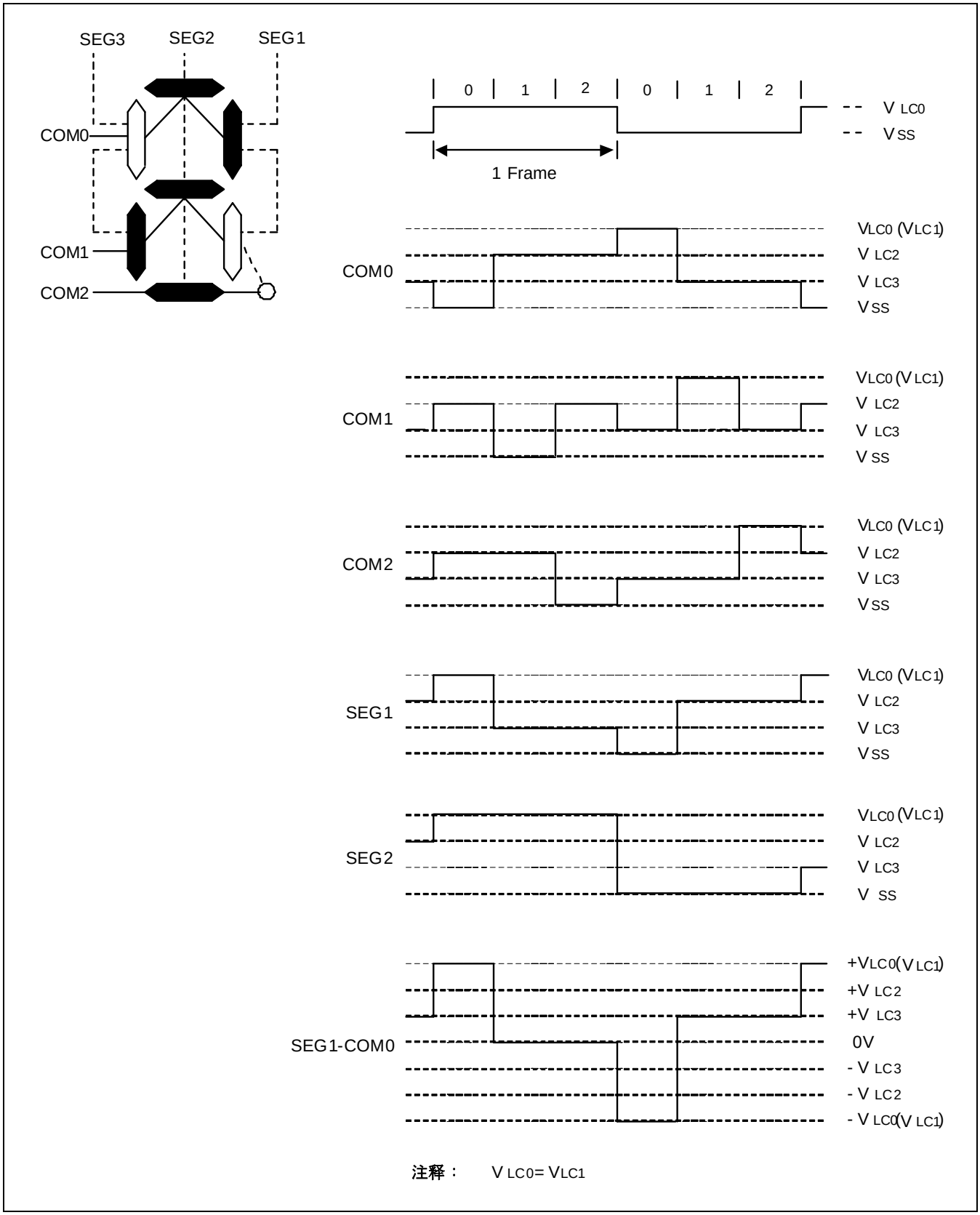


图 15-9 LCD 信号波形 (1/3 占空比, 1/3 偏压比)

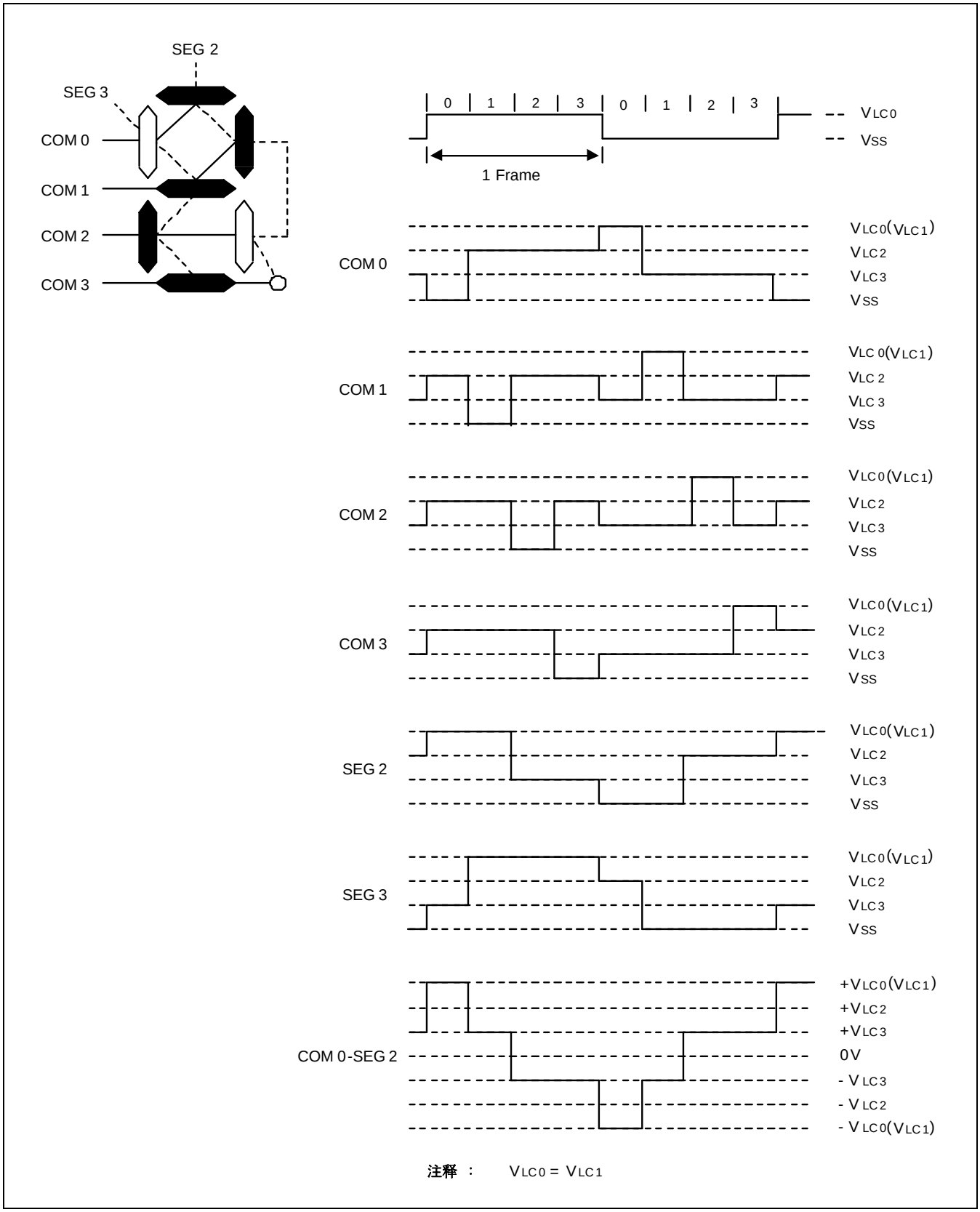


图 15-10 LCD 信号波形 (1/4 占空比, 1/3 偏压比)

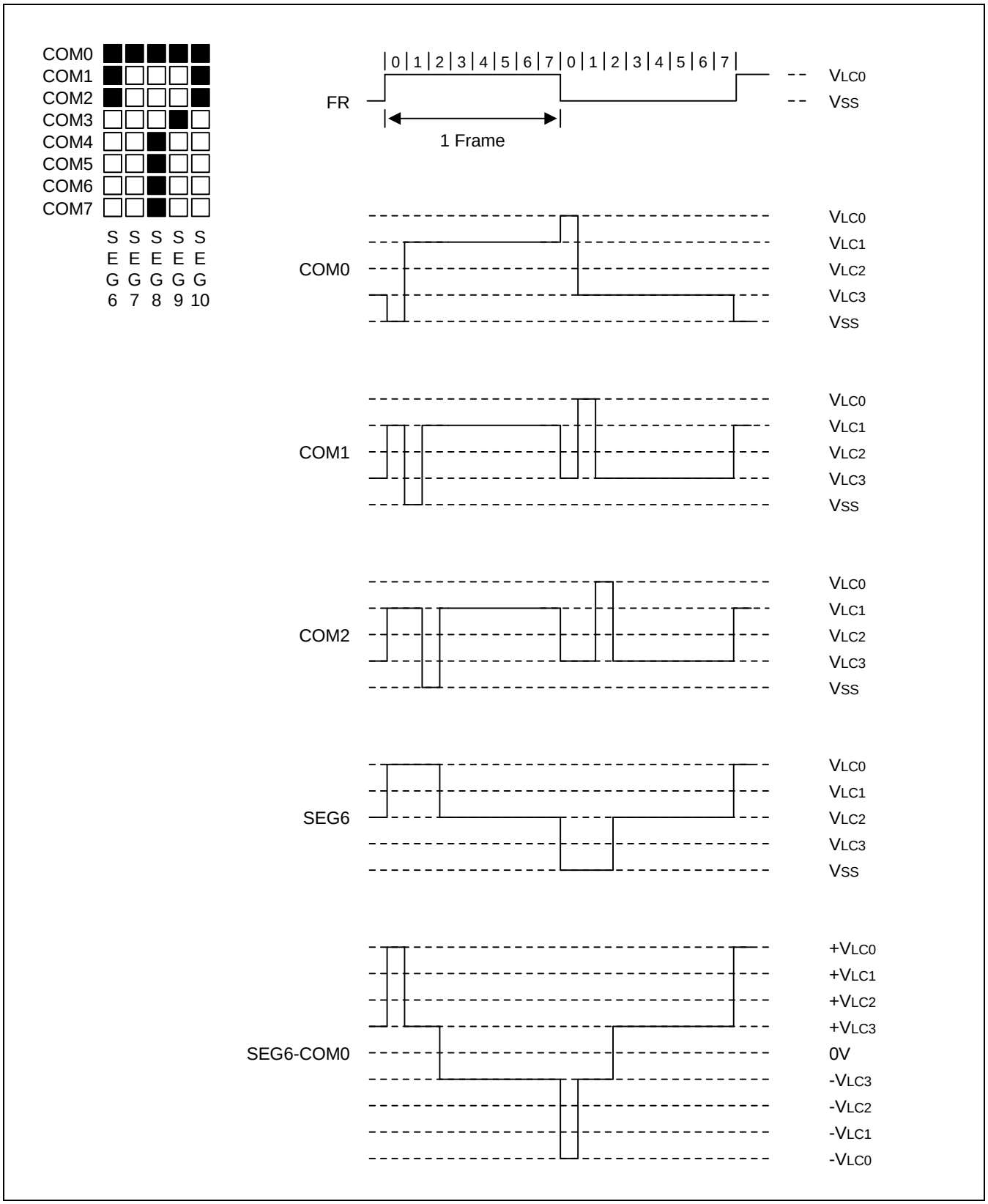


图 15-11 LCD 信号波形 (1/8 占空比, 1/4 偏压比)

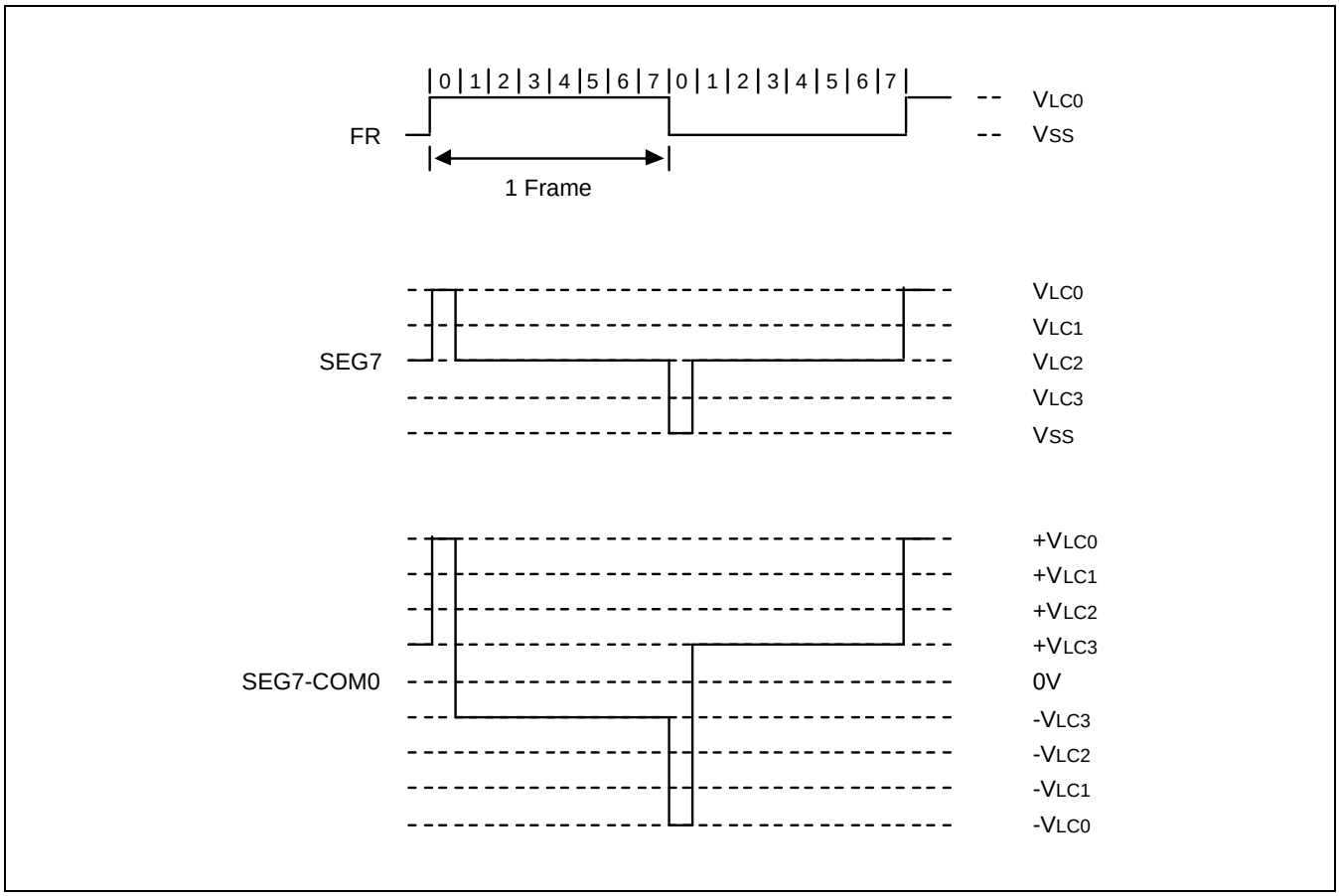


图 15-12 LCD 信号波形 (1/8 占空比, 1/4 偏压比) (续)

16 10 位 A/D 转换器

16.1 概述

10 位 A/D 转换器 (ADC) 模块通过逐次逼近逻辑把由八个输入通道进来的一路模拟信号转换为相应的 10 位数字量。模拟输入量的值必须在 AV_{REF} 和 V_{SS} 之间。A/D 转换器由以下几个部分组成：

- 带逐次逼近逻辑的模拟比较器
- D/A 转换逻辑 (电阻串型)
- ADC 控制寄存器 (ADCON)
- 八路模拟信号输入管脚 (AD0–AD7)
- 10 位 A/D 转换结果寄存器 (ADDATAH/L)
- 8 位数字输入端口 (与 I/O 口复用)
- AV_{REF} 和 V_{SS} 管脚

16.2 功能描述

为了开始一次模数转换，用户首先需要置位 ADCEN 信号来使能位于端口 0 的 ADC 输入，设置为“1”的管脚可用作 ADC 模拟输入。然后通过设置 A/D 转换器控制寄存器(ADCON.4–.6) 来选择八路模拟输入通道 (AD0–AD7) 中的一路，同时启动 A/D 转换即置高使能位 ADCON.0。可读/写的 A/DCON 寄存器的地址为：F7H，Set 1，Bank 0。未使用的 ADC 输入管脚可用作普通 I/O 口。

在一个普通转换过程中，ADC 逻辑电路先将逐次逼近寄存器的值设为 200H(10 位满量程转换结果的一半)。随后这个寄存器在每次转换时会自动更新。逐次逼近电路每次只能实现一个通道的 10 位转换。用户可以通过操作 ADCON 寄存器中的通道选择位 (ADCON.6–.4) 的值来动态选择不同的输入。将转换开始位 ADCON.0 位设置为“1”，可启动 A/D 转换。当一次转换结束时，转换结束 (EOC) 位 ADCON.3 将自动置“1”，同时转换结果保存到 ADDATAH/L 寄存器供用户读取。之后，A/D 转换电路进入空闲状态。
在开始下一次转换前，上一次 A/D 转换的数据必须被读出。否则，前面的结果将被下一次的转换结果覆盖。

注释: 由于 A/D 转换电路内部没有采样/保持电路，转换过程中 AD0–AD7 输入管脚上的模拟信号的波动必须非常小，这点很重要。输入端上的任何变化都可能引入噪声，将会使结果不正确。如果在转换过程中芯片进入 STOP 或 IDLE 模式，A/D 模块中将产生漏电流。故用户必须在 ADC 转换完成之后方可使用 STOP 或 IDLE 模式。

16.2.1 转换时序

A/D 转换处理时，每一位需要 4 步 (4 个时钟边沿)，还需要 10 个时钟周期用于建立 A/D 转换。因此，完成一次 10 位转换需要总共 50 个时钟周期：在 8MHz 时钟频率时，A/D 转换时钟选择 fxx/8，一个时钟周期为 1 us。每一位转换需要 4 个时钟周期，则转换速率计算如下：

4 个时钟周期/位 × 10 位 + 建立时间 = 50 个时钟周期，50 个时钟周期× 1 us = 50 us @1 MHz

16.2.1.1 A/D 转换控制寄存器 (ADCON)

- A/D 转换控制寄存器 ADCON 的地址为：D2H，Set 1，Bank 0，该寄存器 可以实现以下4个功能：
- 模拟输入通道选择 (ADCON.6–.4)
- 转换结束状态检测 (ADCON.3)
- ADC 时钟选择 (ADCON.2–.1)
- A/D 转换开始或禁止 (ADCON.0)

复位后，转换开始位被关闭。用户一次只能选择一路模拟输入。通过操作 ADCON.4–.6 位可以动态分时选择多路 A/D 转换通道 (AD0–AD7)。未使用的模拟输入管脚可用作普通 I/O 口。

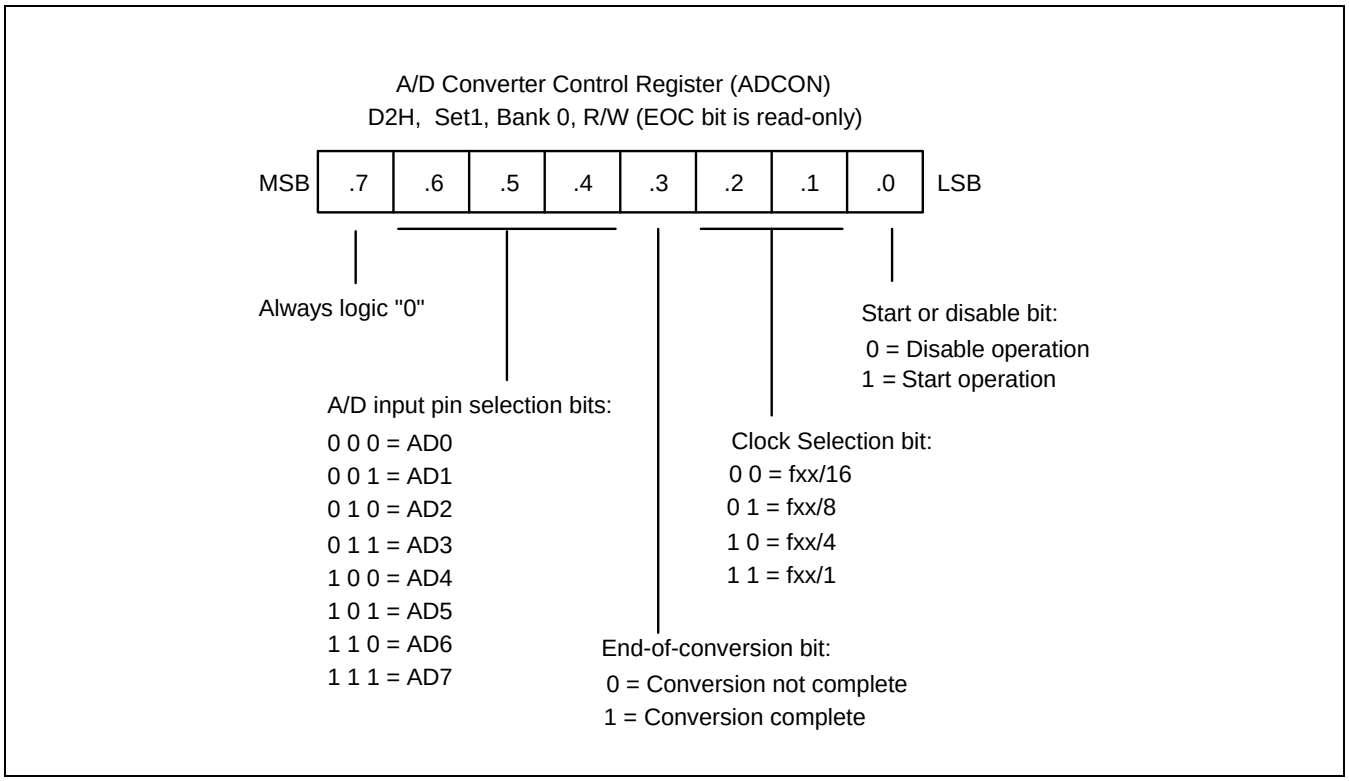


图 16-1 A/D 转换控制寄存器 (ADCON)

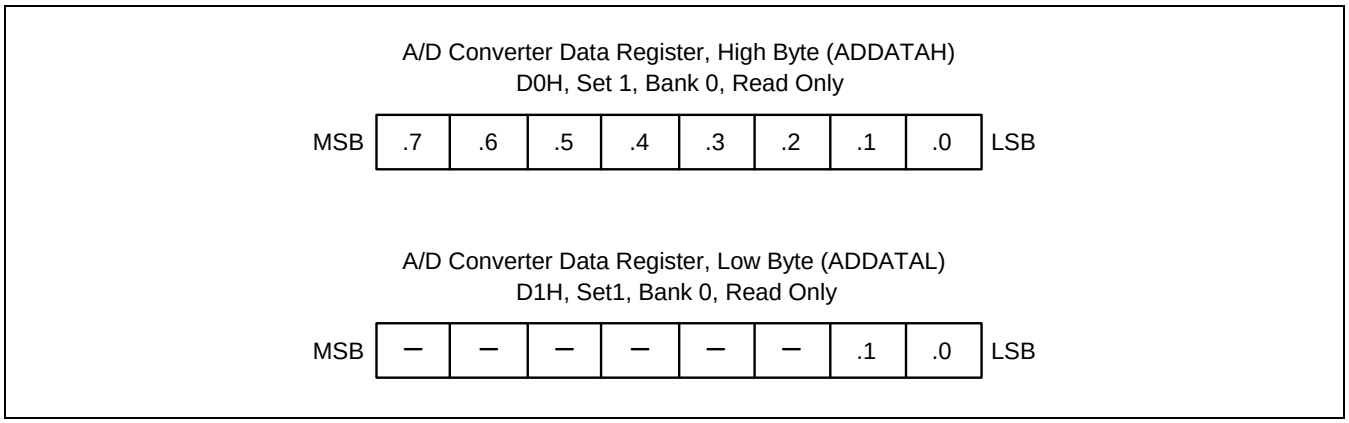


图 16-2 A/D 转换数据寄存器 (ADDATAH/L)

16.2.1.2 内部参考电压

在 ADC 模块中，模拟输入电压范围要与参考电压进行比较。模拟输入信号的电压范围应在 V_{SS} 至 AV_{REF} 之间(通常，$AV_{REF} \leq V_{DD}$)。

在模拟转换过程的每一步转换中，内部电阻网络产生不同的参考电压。第一个转换位的参考电压总是 1/2 AV_{REF}。

16.3 模块框图

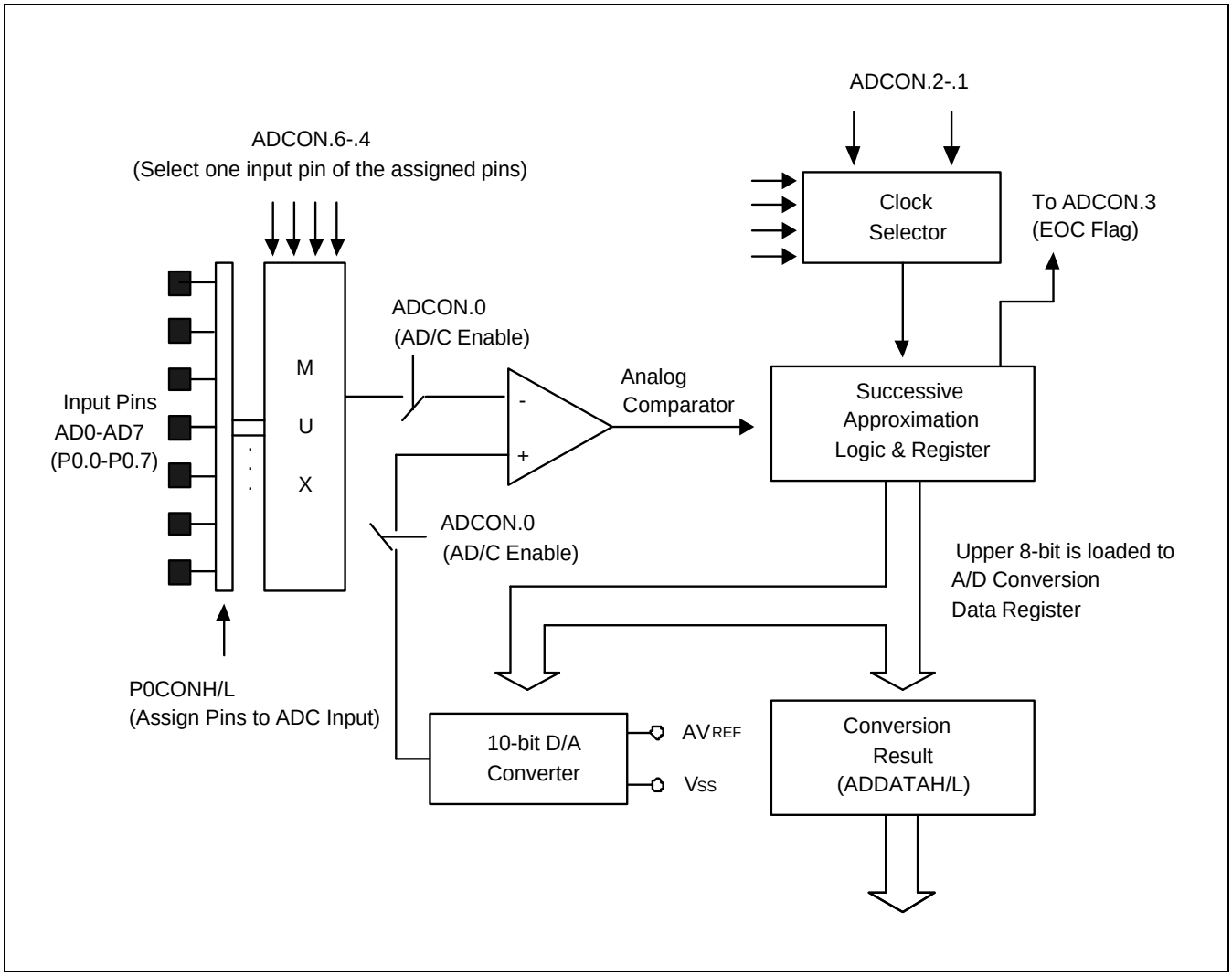


图 16-3 A/D 转换功能模块图

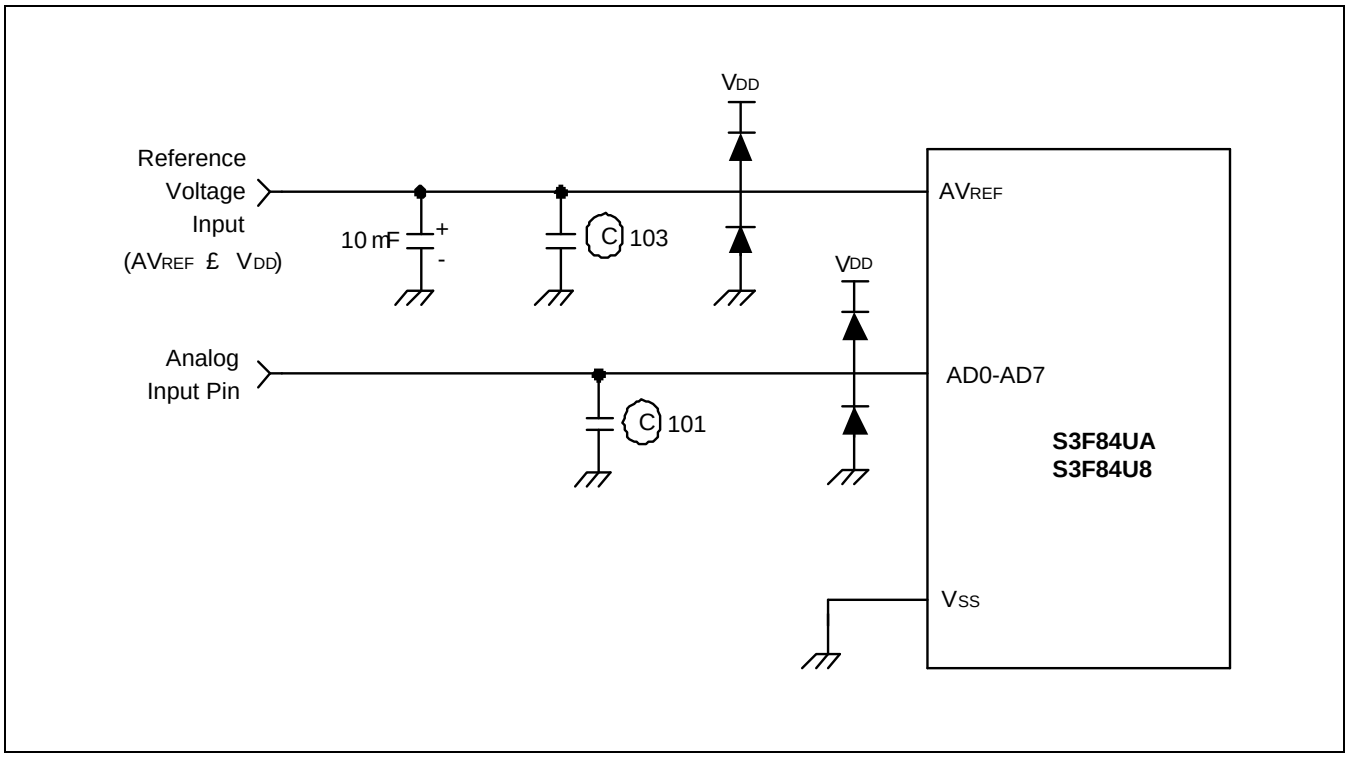


图 16-4 推荐高精度 A/D 转换电路

17 串行输入/输出接口

17.1 概述

串行输入/输出模块 (SIO) 可以与各种外设进行串行数据通讯。SIO 模块主要包括：

- 8 位控制寄存器 (SIOCON)
- 时钟选择逻辑
- 8 位数据缓冲器 (SIODATA)
- 8 位预分频器
- 3 位串行时钟计数器
- 串行数据输入/输出引脚 (SI，SO)
- 外部时钟输入/输出管脚 (SCK)

SIO 模块可以发送或接收 8 位串行数据，其发送，接收频率由它对应的控制寄存器设置。
为了实现灵活的数据传输速率，用户可以选择内部或者外部时钟源。

17.2 编程步骤

SIO 模块的编程遵循以下基本的步骤：

如果需要，通过 P3CONL 寄存器设置端口上的 I/O 管脚 (SO，SCK 和 SI)

写一个 8 位数据值到寄存器 SIOCON 中，正确的配置 SIO 模块。SIOCON.2 位必须设置为“1”
以使能数据的移位。

需要中断时，将 SIO 中断使能位 (SIOCON.1) 设置为“1”。

用户发送数据到串行缓冲器，写输入到 SIODATA，并将 SIOCON.3 位设为“1”，则启动移位操作。

当移位操作 (发送/接收) 完成后，SIO 中断标志位 (SIOCON.0) 置为“1”，同时产生 SIO 中断请求。

17.2.1 SIO 控制寄存器 (SIOCON)

串行输入/输出接口模块的控制寄存器 SIOCON 的地址为：E7H，Set 1，Bank 0。设置 SIO 模块的以下功能：

- 移位时钟的时钟源选择 (内部或者外部)
- 中断使能
- 移位操作的边沿选择
- 清除 3 位计数器并开始移位操作
- 使能移位操作 (发送)
- 模式选择 (发送/接收或只接收)
- 数据方向选择 (最高位优先或最低位优先)

复位后，将 SIOCON 的值清除为“00H”，设置对应模块在 SCK 上选择内部时钟，选择只接收的工作模式，并将 3 位计数器清零。禁止数据移位操作和中断。所选的数据方向为最高位优先。

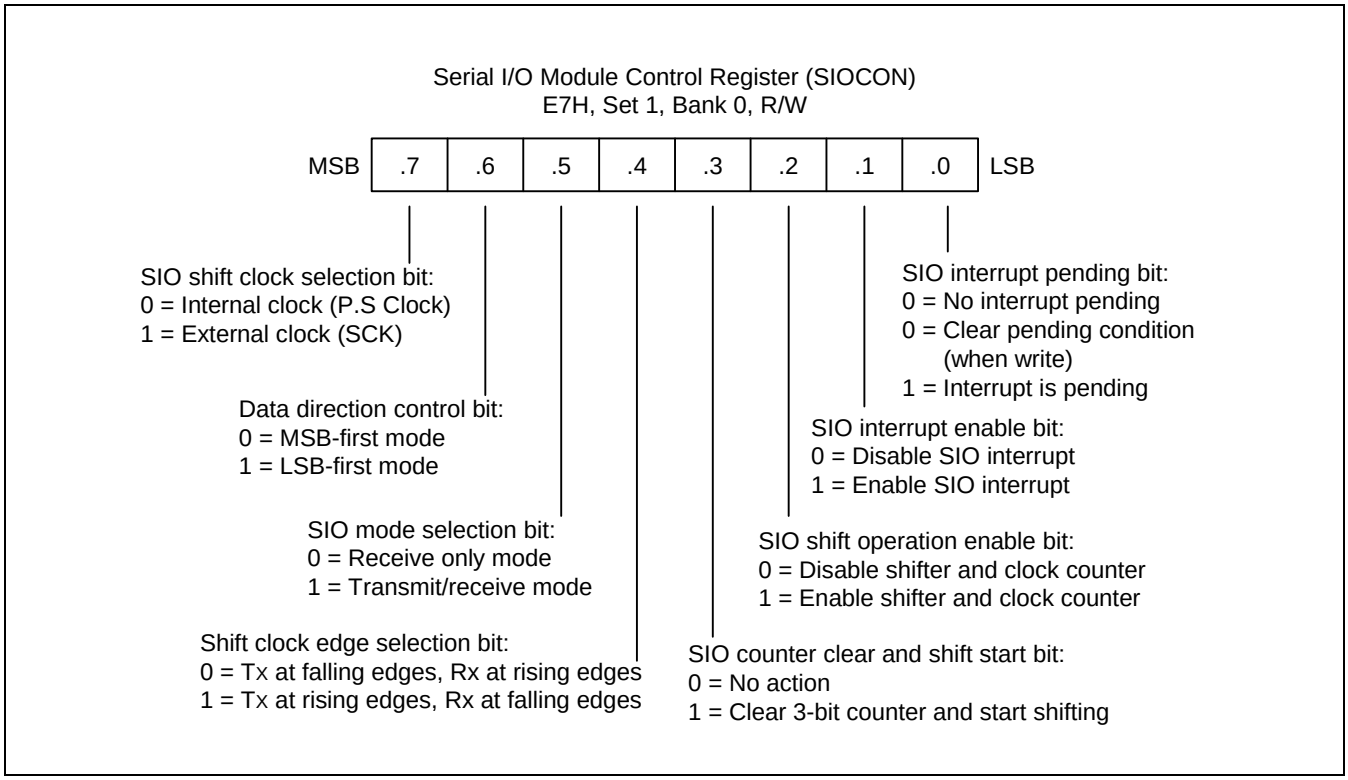


图 17-1 串行输入/输出模块控制寄存器 (SIOCON)

17.2.2 SIO 预分频寄存器 (SIOPS)

串行输入/输出接口模块的控制寄存器 SIOPS 的地址为：E9H，Set 1，Bank 0。SIO 预分频寄存器 SIOPS 中存储的值决定了 SIO 时钟速率 (波特率)，如下所示：

波特率 = 输入时钟频率 (fxx/4) / (预分频值+ 1)， 或 SCK 输入时钟频率，此时输入时钟为 fxx/4

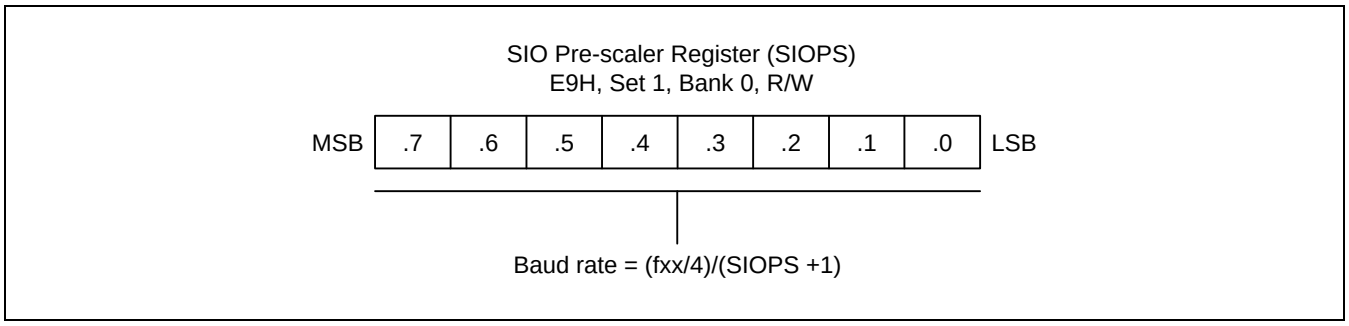


图 17-2 SIO 预分频寄存器 (SIOPS)

17.3 模块框图

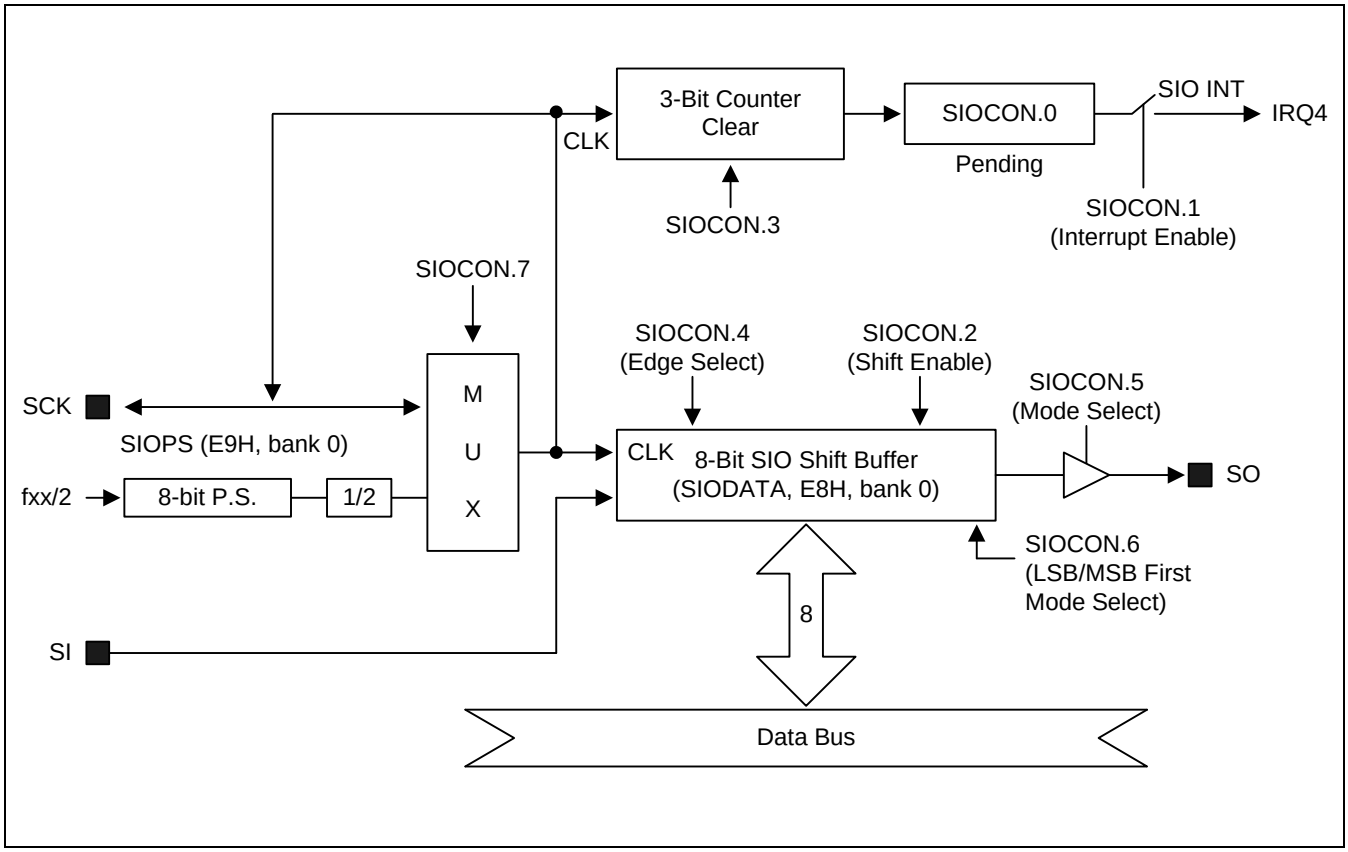


图 17-3 SIO 功能模块框图

17.3.1 串行输入/输出时序图

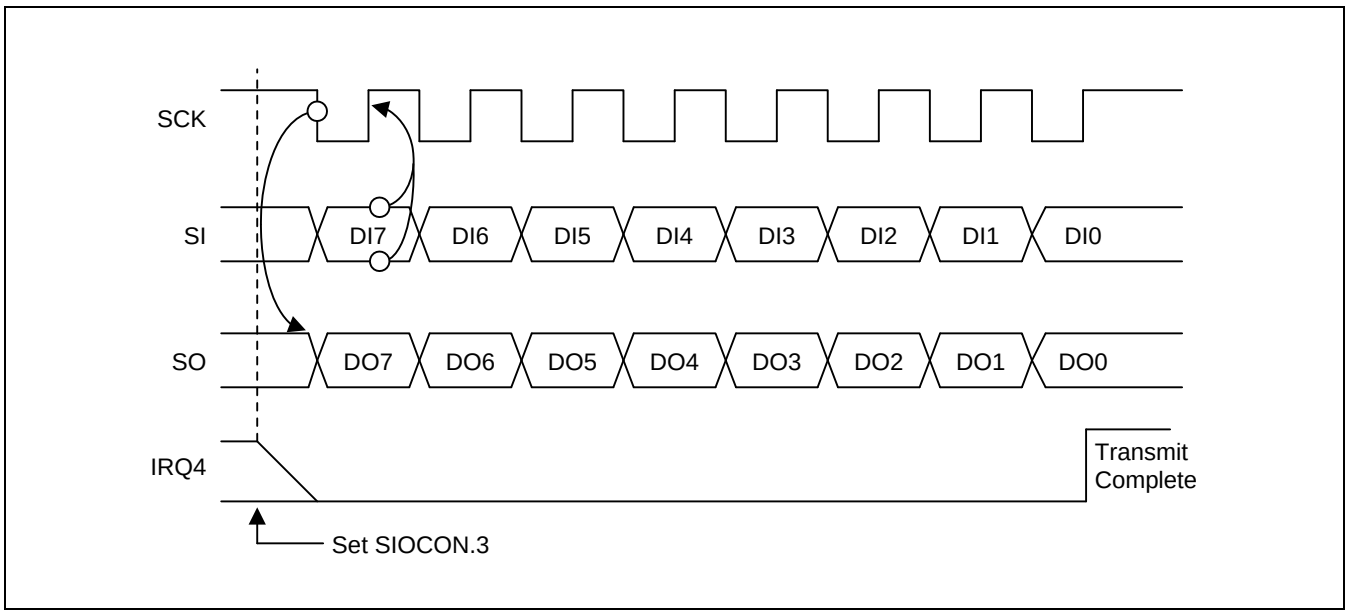


图 17-4 串行输入/输出发送/接收模式的时序 (下降沿发送, SIOCON.4 = 0)

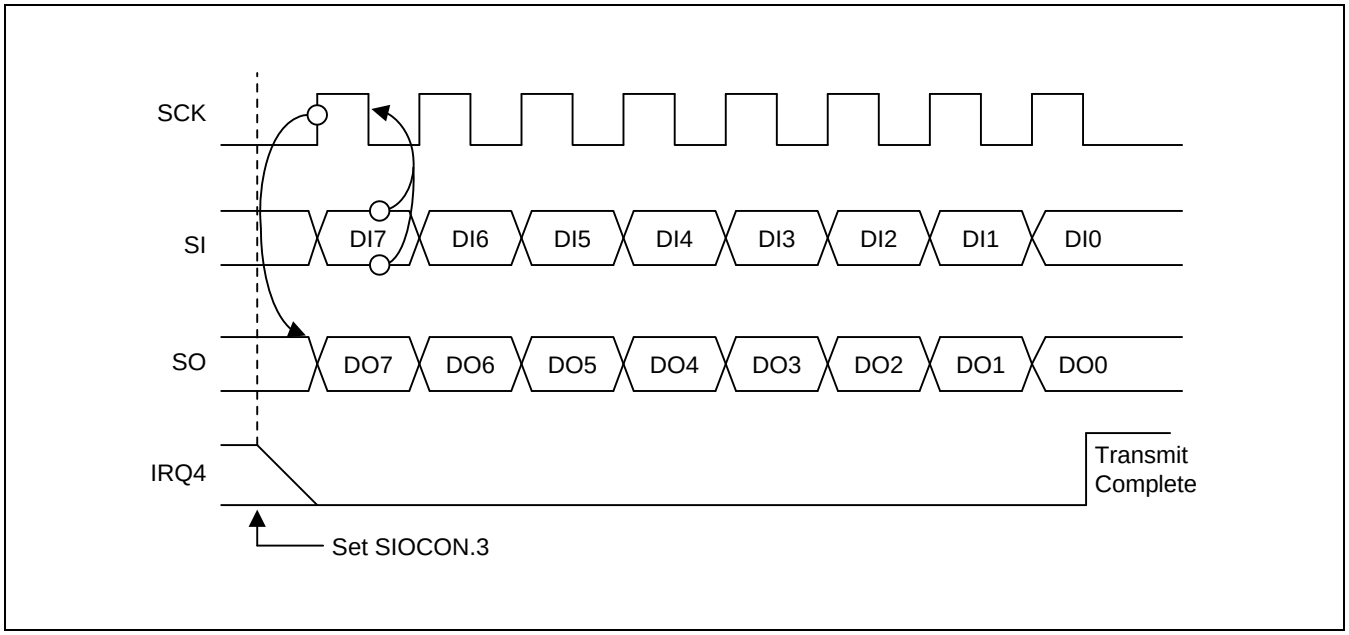


图 17-5 串行输入/输出发送/接收模式的时序 (上升沿发送，SIOCON.4 = 1)

18 UART 0

18.1 概述

UART 0 模块是一个全双工的串行通讯口，其运行模式可编程。
包括一个同步模式和三个 UART(通用异步串行接收/发送)模式：

- 串行 I/O 口，波特率为 $f_U/(16 \times (BRDATA0+1))$
- 8 位 UART 模式，波特率可设置
- 9 位 UART 模式，波特率固定为：$f_U/16$
- 9 位 UART 模式，波特率可设置

UART 0 的接收和发送缓冲都是通过数据寄存器 UDATA0 (地址：F0H，Set 1，Bank 0) 来实现的。写 UART 数据寄存器时，数据装载到发送缓冲中；读 UART 0 数据寄存器时，数据从 UART 0 的接收缓冲器中读出。
发送缓冲器和接收缓冲器在物理上是分开的。

访问接收数据缓冲器(移位寄存器)时，在上一个接收的数据还读出之前，可以开始下一个数据的接收。因此，如果在下一个数据完成接收之前，上一个接收的数据还没有读出的话，则上一个数据会丢失(过载错误)。

在所有运行模式下，当任何指令(通常是一个写指令)将 UDATA0 作为目标地址写入数据时，则立即开始发送。
在模式 0 下，只有当接收中断标志位 (UART0CONH.0) 为“0”且接收使能位 (UART0CONH.4) 为“1”时，才开始接收串行数据。在模式 1，2 和 3 下，当接收使能位 (UART0CONH.4) 为“1”时，
任何时刻只要收到数据起始位(“0”)则立即开始接收数据。

UART 0 模块编程的基本步骤为：

1. 通过设置 P4CONH 为相应的值，将 P4.7 和 P4.6 设为 UART 0 功能引脚(RXD0 (P4.7)，TXD0 (P4.6))。
2. 设置 UART0CONH/L 控制寄存器来选择 UART 0 的 I/O 口模块。
3. 需要中断时，将 UART 0 中断使能位 (UART0CONH.1 位或 UART0CONL.1 位)设置为“1”。
4. 需要传送数据时，将需要传送的数据写入到 UDATA 0，则开始传送。
5. 当发送/接收完一个数据后，UART 0 标志位 (UART0CONH.0 位或 UART0CONL.0 位)被置为“1”，同时产生一个相应的发送/接收中断。

18.1.1 UART 0 高字节控制寄存器 (UART0CONH)

UART 0 的高字节控制寄存器是 UART0CONH，地址为：EEH，Set 1，Bank 0。主要实现以下功能：

- 运行模式和波特率选择
- 多机通讯和中断控制
- 串行接收使能/禁止控制
- 模式 2 和模式 3 下发送和接收时第 9 位数据位的位置
- UART 0 接收中断控制

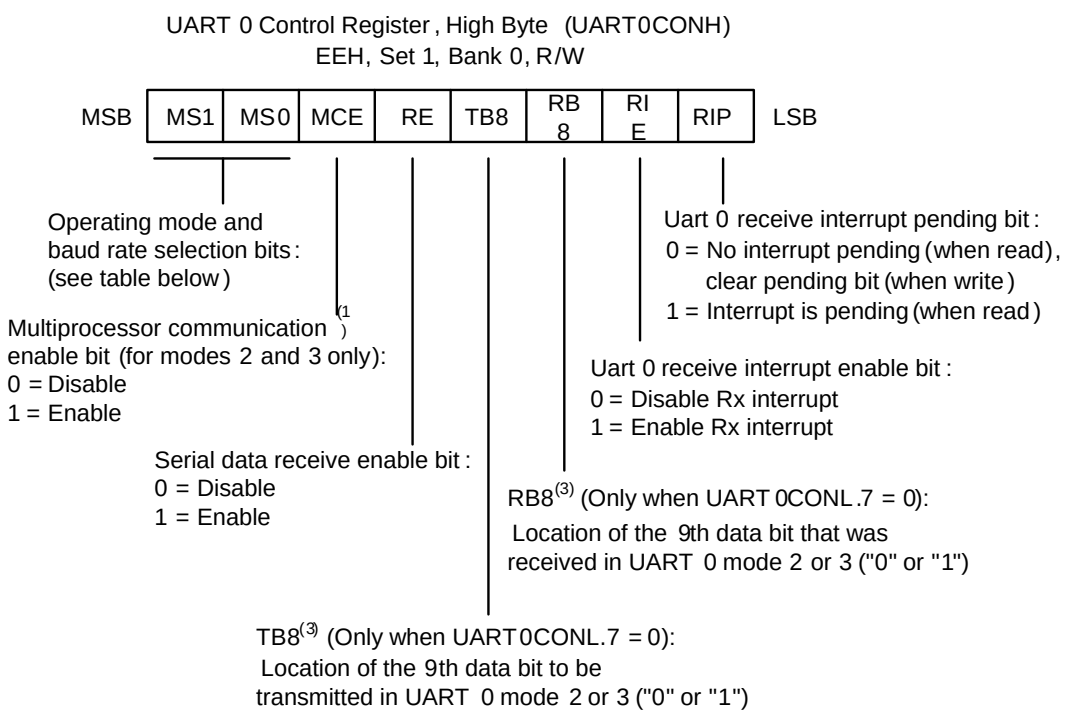
复位后，将清除 UART0CONH 至“00H”。因此，用户如果需要使用 UART 0 模块，应设定 UART0CONH 为适当的值。

18.1.2 UART 0 低字节控制寄存器低字节 (UARTCONL)

UART 0 的低字节控制寄存器是 UART0CONL，地址为：EFH，Set 1，Bank 0。主要实现以下功能：

- UART 0 发送和接收的校验位选择
- UART 0 时钟选择
- UART 0 发送中断控制

复位后，将清除 UART0CONL 至“00H”。因此，用户如果需要使用 UART 0 模块，应设定 UART0CONL 为适当的值。

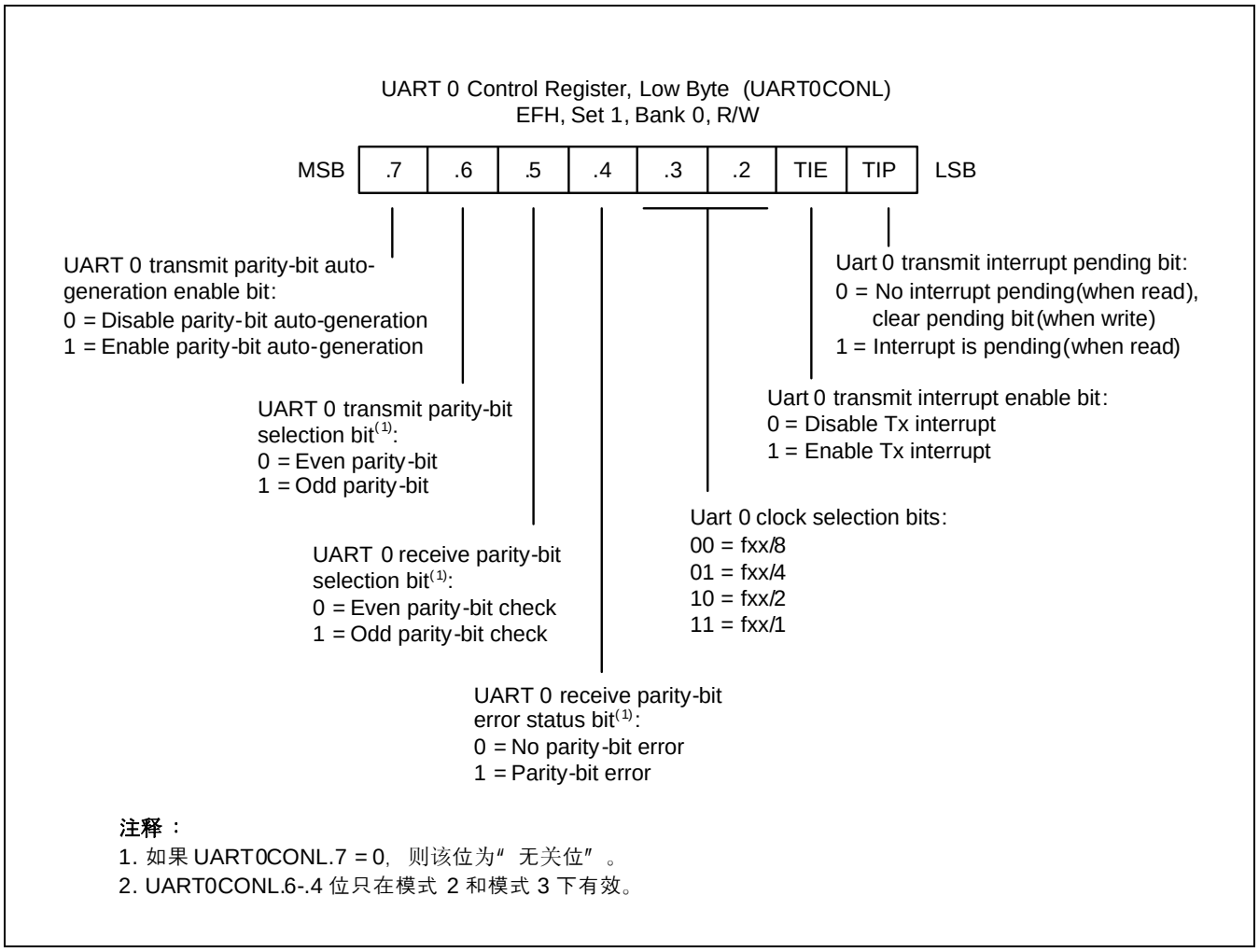


MS1	MS0	Mode	Description (2)	Baud Rate
0	0	0	Shift register	(fu/(16 x (BRDATA0 + 1)))
0	1	1	8-bit UART	(fu/(16 x (BRDATA0 + 1)))
1	0	2	9-bit UART	(fu/16)
1	1	3	9-bit UART	(fu/(16 x (BRDATA0 + 1)))

注释:

1. 在模式 2 或者模式 3 下，如果 UART0CONH.5 位设置为“1”，则当收到的第 9 位数据位为“0”时，不会产生接收中断。在模式 1 模式下，如果 UART0CONH.5 位的值设置为“1”，则在没有收到有效的停止位的时候，不会产出接收中断。在模式 0 下，应将 UART0CONH.5 位设为“0”。
2. 有关 UART 的 8 位和 9 位的的描述中，其中的 8 位和 9 位是指有效数据的位数，不包括发送和接收时的开始位和停止位。
3. 如果 UART0CONL.7 = 1，表示该位为“无关位”。

图 18-1 UART 0 高字节控制寄存器 (UART0CONH)



注释：

1. 如果 UART0CONL.7 = 0，则该位为"无关位"。
2. UART0CONL.6-.4 位只在模式 2 和模式 3 下有效。

图 18-2 UART 0 低字节控制寄存器 (UART0CONL)

18.1.3 UART 0 中断标志位

在模式 0 下，当收到第 8 位数据位移位后，接收中断标志位 UART0CONH.0 将置为“1”。在模式 1 下，UART0CONH.0 在停止位的移位时间的中间点处置“1”。在模式 2 或模式 3 下，在接收到 RB8 位的移位时间的中间点处，UART0CONH.0 位将置为“1”。当 CPU 响应了接收中断后，UART0CONH.0 位必须在中断服务程序中由软件清零。

在模式 0 下，当第 8 位发送数据位移位后，发送中断标志位 UART0CONL.0 位被置为“1”。在模式 1，2 或 3 下，UART0CONL.0 位在停止位开始发送时置位。当 CPU 响应了发送中断后，UART0CONL.0 位必须在中断服务程序中由软件清零。

18.1.4 UART 0 数据寄存器 (UDATA0)

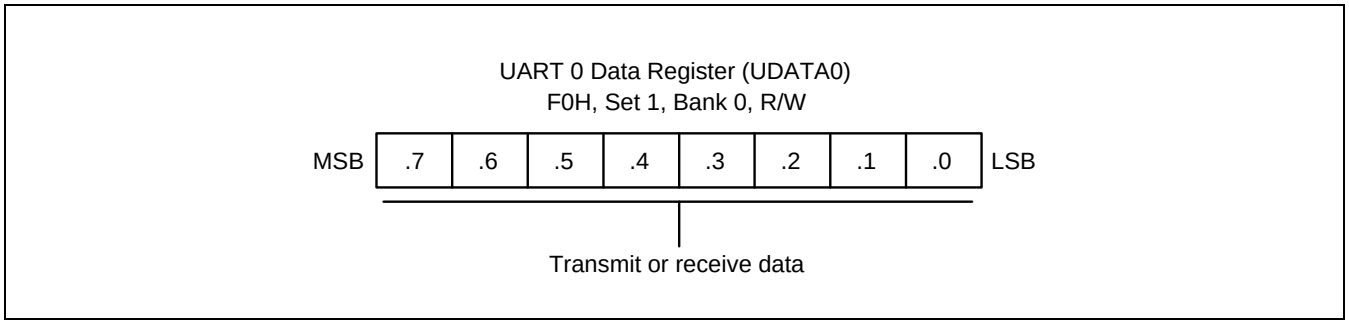


图 18-3 UART 0 数据寄存器 (UDATA0)

18.1.5 UART 0 波特率数据寄存器 (BRDATA0)

通过设置 UART 0 波特率数据寄存器 BRDATA0，可以设定 UART 0 的时钟速率(即波特率)。

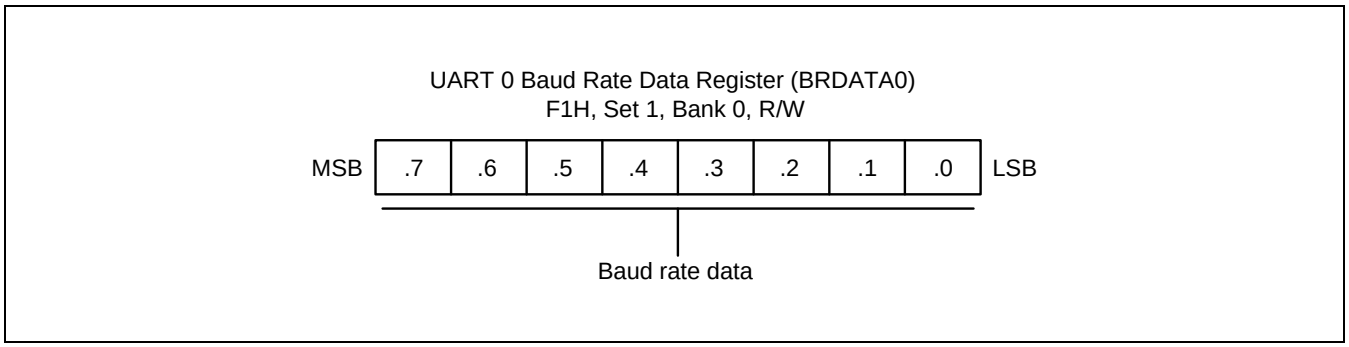


图 18-4 UART 0 波特率数据寄存器 (BRDATA0)

18.1.6 波特率计算

模式 0 的波特率计算

在模式 0 下，波特率由波特率数据寄存器 BRDATA0 (地址：F1H，Set 1，Bank 0) 的值决定，模式 0 的波特率为：$f_U/(16 \times (BRDATA0 + 1))$

模式 2 的波特率计算

模式 2 的波特率固定为 $f_U/16$，即 f_U 信号的 16 分频。

模式 1 和模式 3 的波特率计算

在模式 1 和模式 3 下，波特率由波特率数据寄存器，BRDATA0 (地址：F1H，Set 1，Bank 0) 的值决定，模式 1 和模式 3 的波特率为：$f_U/(16 \times (BRDATA0 + 1))$

表 18-1 利用 BRDATA0 可以产生的常用波特率表

模式	波特率	UART 时钟 (f_U)	BRDATA0	
			十进制	十六进制
模式 2	0.5MHz	8MHz	x	x
模式 0 模式 1 模式 3	230,400Hz	11.0592MHz	02	02H
	115,200Hz	11.0592MHz	05	05H
	57,600Hz	11.0592MHz	11	0BH
	38,400Hz	11.0592MHz	17	11H
	19,200Hz	11.0592MHz	35	23H
	9,600Hz	11.0592MHz	71	47H
	4,800Hz	11.0592MHz	143	8FH
	62,500Hz	10MHz	09	09H
	9,615Hz	10MHz	64	40H
	38,461Hz	8MHz	12	0CH
	12,500Hz	8MHz	39	27H
	19,230Hz	4MHz	12	0CH
	9,615Hz	4MHz	25	19H

18.1.7 模块框图

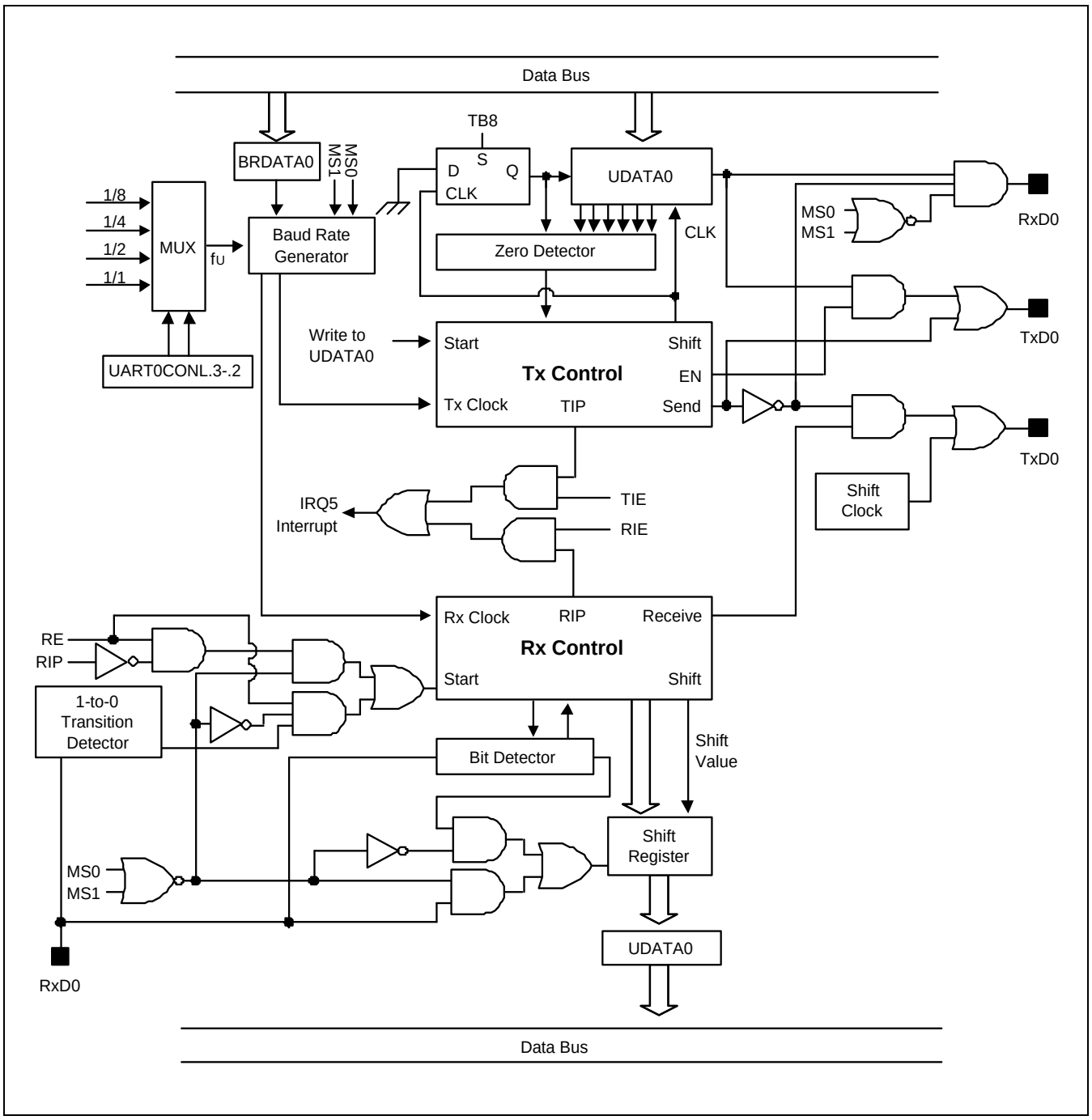


图 18-5 UART 0 功能模块框图

18.1.8 UART 0 模式 0 功能描述

在模式 0 下，UART 0 通过 RxD0 (P4.7) 管脚进行接收和发送数据，TxD0 (P4.6) 输出数据移位时钟信号。数据接收和发送的长度都是 8 位，首先发送(或接收)的是数据的最低位 (LSB)。

18.1.8.1 模式 0 数据发送流程

1. 设置 UART0CONL.3 和 .2 位，选择 UART 0 的时钟源。
2. 设置 UART 0 发送时奇偶校验自动生成使能或禁止位 (UART0CONL.7)。
3. 设置 UART0CONH.7 和 .6 位为“00B”，选择模式 0。
4. 将需要发送的数据写入移位寄存器 UDATA0 (地址：F0H，Set1，Bank 0)，数据开始发送。

18.1.8.2 模式 0 数据接收流程

1. 设置 UART0CONL.3 和 .2 位，选择 UART 0 的时钟源。
2. 设置 UART 0 发送时奇偶校验自动生成使能或禁止位 (UART0CONL.7)。
3. 设置 UART0CONH.7 和 .6 位为“00B”，选择模式 0。
4. 往 UART0CONH.0 位写“0”清除接收中断标志位 (UART0CONH.0)。
5. 设置 UART 0 接收使能位 (UART0CONH.4) 为“1”，使能 UART 0 接收。
6. 移位时钟开始输出到 TxD0 (P4.6) 管脚，同时开始从 RxD0 (P4.7) 管脚上读取数据。
   当 UART0CONH.1 位置为“1”"时，产生一个 UART 0 接收中断。

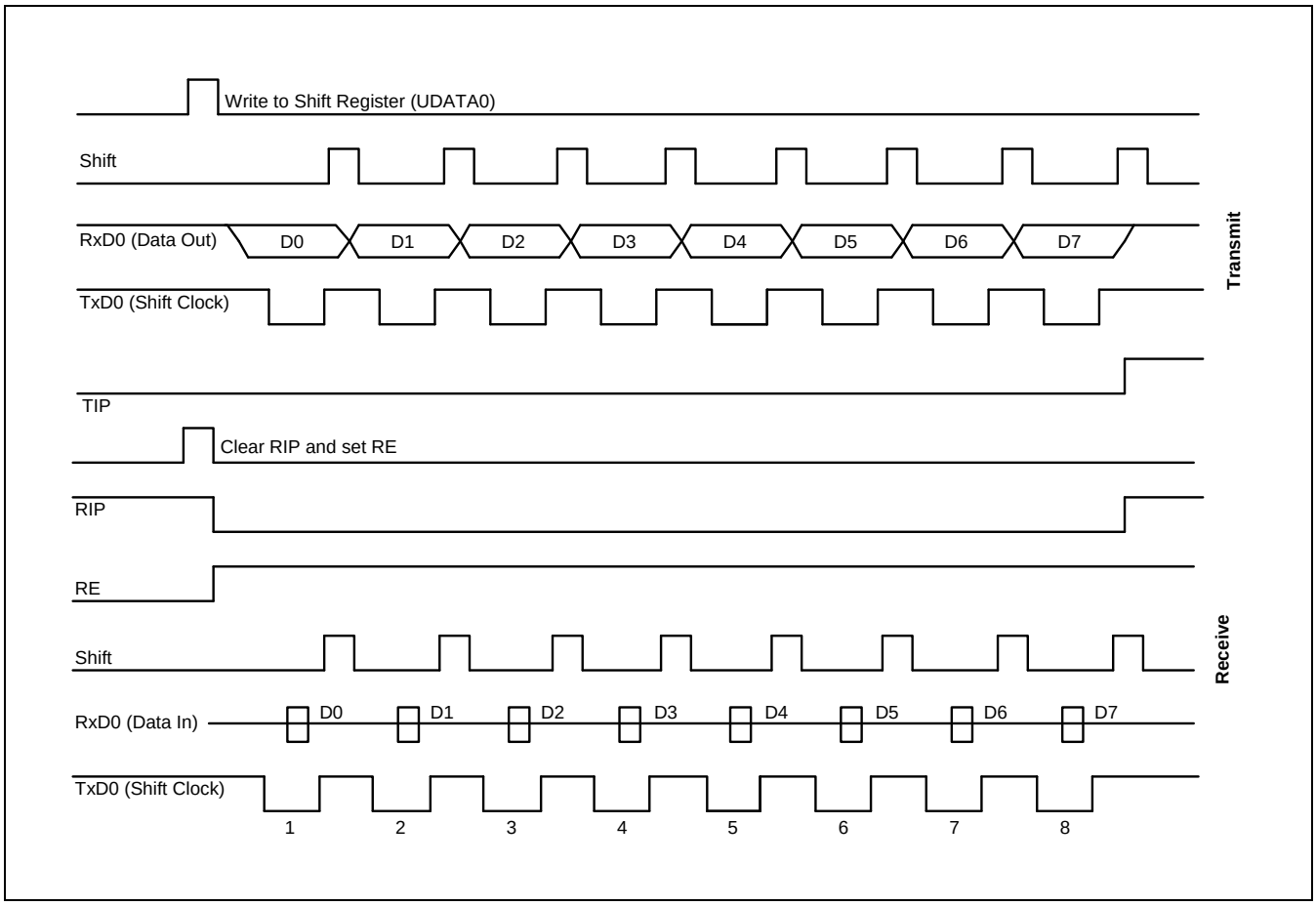


图 18-6 UART0 模式 0 运行时序图

18.1.9 UART 0 模式 1 功能描述

在模式 1 下，发送(通过 TxD0 (P4.6) 管脚) 或者接收 (通过 RxD0 (P4.7) 管脚)的数据帧总长度是 10 位。每一个数据帧由三部分组成：

- 起始位 (“0”)
- 8 位有效数据 (最低位优先)
- 停止位 (“1”)

模式 1 的波特率是可变的。

18.1.9.1 模式 1 数据发送流程

1. 设置 UART0CONL.3 和 .2 位，选择 UART 0 时钟源。
2. 设置 UART 0 发送时奇偶校验自动生成使能或禁止位 (UART0CONL.7)。
3. 通过设置 8 位 BRDATA0 寄存器设定波特率。
4. 设置 UART0CONH.7-.6 位为“01B”，选择模式 1(8 位 UART)。
5. 将需要发送的数据写入移位寄存器 UDATA0 (地址：F0H，Set 1，Bank 0)，数据开始发送。
   起始位和停止位由硬件自动产生。

18.1.9.2 模式 1 数据接收流程

1. 设置 UART0CONL.3 和 .2 位，选择 UART 0 时钟源。
2. 设置 UART 0 发送时奇偶校验自动生成使能或禁止位 (UART0CONL.7)。
3. 通过设置 8 位 BRDATA0 寄存器设定波特率。
4. 选择模式 1 模式并设置 UART0CONH 中的 RE (接收使能) 位为“1”。
5. 当 RxD0 (P4.7) 管脚上检测到起始条件(低电平“0”)时，UART 0 模块开始进行串行数据接收操作。

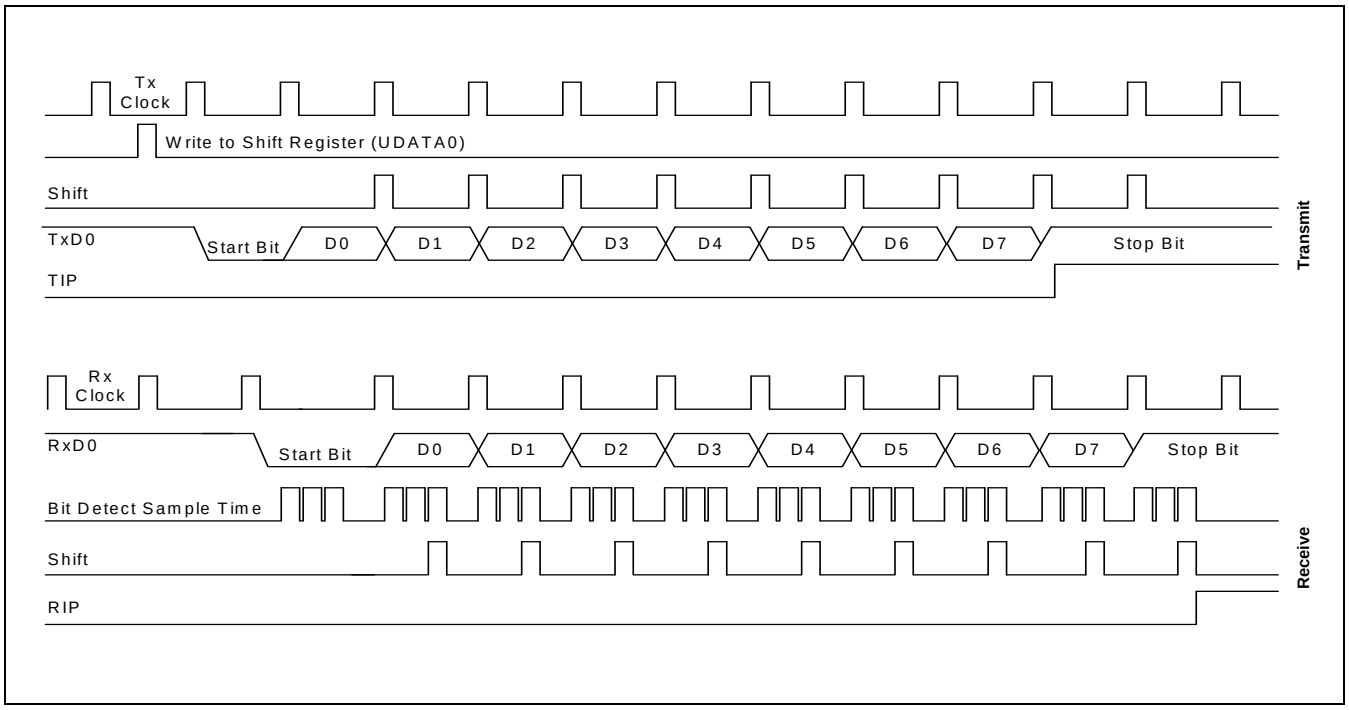


图 18-7 UART 0 模式 1 运行时序图

18.1.10 UART 0 模式 2 功能描述

在模式 2 下，发送(通过 TxD0 (P4.6)) 或者接收 (通过 RxD0 (P4.7))的数据帧总长度是 11 位。
每一个数据帧由四部分组成:

- 起始位 (“0”)
- 8 位有效数据 (最低位优先)
- 可编程设置的第 9 位数据位
- 停止位 (“1”)

发送时，将需要发送的第 9 位数据的值“0”或“1”写入到 TB8 位 (UART0CONH.3)。接收时，接收到的第9位数据存放在 RB8 位 (UART0CONH.2)，同时停止位被忽略。 模式 2 的波特率为 $f_U/16$。

18.1.10.1 模式 2 数据发送流程

1. 设置 UART0CONL.3 和 .2 位， 选择 UART 0 时钟源。
2. 设置 UART 0 发送时奇偶校验自动生成使能或禁止位 (UART0CONL.7)。
3. 设置 UART0CONH.7-.6 位为“10B”，选择模式 2 (9 位 UART)，同时将需要发送的第 9 位数据“0”或“1”写入TB8。
4. 将需要发送的数据写入移位寄存器 UDATA0 (地址：F0H，Set 1，Bank 0)，数据开始发送。

18.1.10.2 模式 2 数据接收流程

1. 设置 UART0CONL.3 和 .2位， 选择 UART 0 时钟源。
2. 设置 UART 0 发送时奇偶校验自动生成使能或禁止位 (UART0CONL.7)。
3. 选择模式 2 并设置 UART0CONH 中的 接收使能位 (RE) 为“1”。
4. 当 RxD0 (P4.7) 管脚上的电平变为低时，开始接收数据。

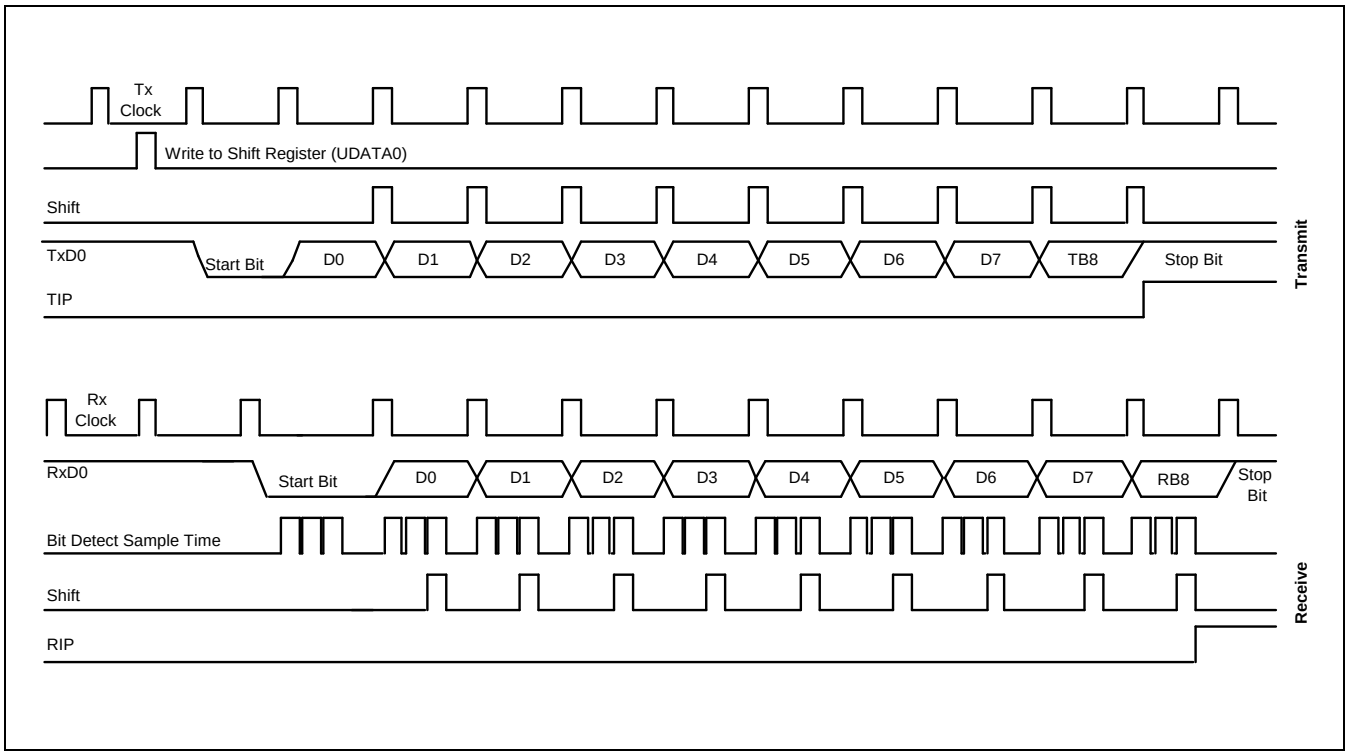


图 18-8 UART 0 模式 2 运行时序图

18.1.11 UART 0 模式 3 功能描述

在模式 3 下，发送(通过 TxD0 (P4.6) 管脚) 或者接收 (通过 RxD0 (P4.7) 管脚)的数据帧总长度是 11 位。除了模式 3 的波特率可以调节之外，其余的功能模式 3 和模式 2 是完全相同的。每一个数据帧由四部分组成:

- 起始位 (“0”)
- 8 位有效数据 (最低位优先)
- 可编程设置的第 9 位数据位
- 停止位 (“1”)

18.1.11.1 模式 3 数据发送流程

1. 设置 UART0CONL.3 和 .2 位，选择 UART 0 时钟源。
2. 设置 UART 0 发送时奇偶校验自动生成使能或禁止位 (UART0CONL.7)。
3. 设置 UART0CONH.7-.6 位为“11B”，选择模式 3 (9 位 UART)，同时将需要发送的第 9 位数据写入 TB8 (UART0CONH.3)。
4. 将需要发送的数据写入移位寄存器 UDATA0 (地址：F0H，Set 1，Bank 0)，数据开始发送。

18.1.11.2 模式 2 数据接收流程

1. 设置 UART0CONL.3 和 .2 位，选择 UART 0 时钟源。
2. 设置 UART 0 发送时奇偶校验自动生成使能或禁止位 (UART0CONL.7)。
4. 选择模式 3 并设置 UART0CONH 寄存器中的 RE (接收使能) 位为“1”。
5. 当 RxD0 (P4.7) 管脚上的电平变为低时，开始接收数据。

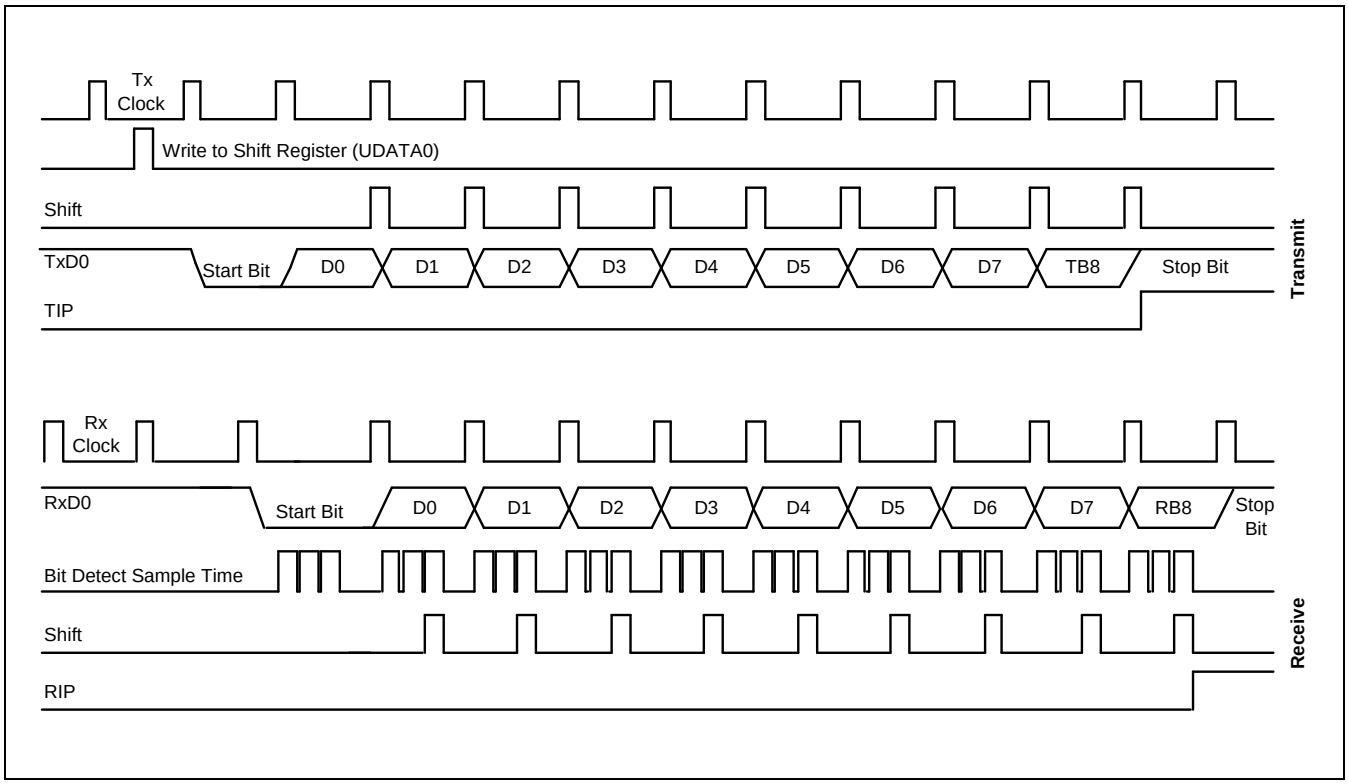


图 18-9 UART 0 模式 3 运行时序图

18.1.12 多机串行通讯的配置

S3F8- 系列的多机串行通讯功能的特点是：在一个由多个 S3F84UA/F84U8 组成的串行通信系统中，
允许一个S3F84UA/F84U8 作为“主机”发送一组多帧串行数据给另一个的“从机”设备，
而不会影响连接在同一个串行线上的其他从机设备。

这个功能只有在 UART 的模式 2 和模式 3 下才有。在模式 2 和模式 3 下，收到的数据共 9 位。
第 9 位写入到 RB8 (UART0CONH.2)。接收操作在收到停止位后完成。
因此可以编程实现这样的功能：收到停止位后，只有当 RB8 =“1”时才产生中断。

为了实现这个功能，需要设置 UART0CONH 寄存器中的 MCE 位，当设置 MCE 位为“1”时，
接收的串行数据帧中的第 9 位数据 =“0”时不会产生中断。在这种情况下，就可以使用第 9
位来简单的区分地址和数据。

18.1.13 主机/从机交互协议的一个例子

当主机希望给串接在同一条串行总线上的很多从机中的一个从机发送数据块时，它首先发送地址信息来选中目标从机。注意在这种情况下，一个地址帧和一个数据帧是不同的，地址帧的第 9 位数据为“1”，而数据帧的第 9 位数据为“0”。

地址帧会让所有的从机都产生中断，并被每个从机都接收到，则每个从机就可以通过检查接收到的数据来判断是否被选中。 被选中的目标从机会将 MCE 位清为“0”并准备开始接收数据。

没有被选中的从机的 MCE 位继续保持为“1”，保持正常运行，并忽略串行总线上的数据帧。

MCE 位在模式 0 下没有用处。在模式 1 下，它可以用来检测停止位的有效性。在模式 1
下接收数据时，如果设置 MCE 位为“1”，只有接收到的停止位有效(为“1”)时，才可以产生接收中断。

18.1.14 多机通讯的配置流程

多机通讯的配置流程如下：

1. 设置所有的串行总线上的 S3F84UA/F84U8 设备 (主机和从机) 的 UART 0 模式为模式 2 或者模式 3。
2. 设置所有从机设备的 MCE 位为“1”。
3. 主机的传输协议为:
   - 第一个字节：地址
     - 标识目标从机设备地址 (第 9 位 =“1”)
   - 后续字节：数据
     - (第 9 位 =“0”)
4. 由于第 9 位为“1”，当所有的目标从机设备接收到第一个字节时都会产生中断。
   目标从机收到第一个字节后，将收到的地址与自己的地址进行比较，确认被选中后，则将 MCE 位清除为“0”来准备接受数据。其他的从机继续正常运行。

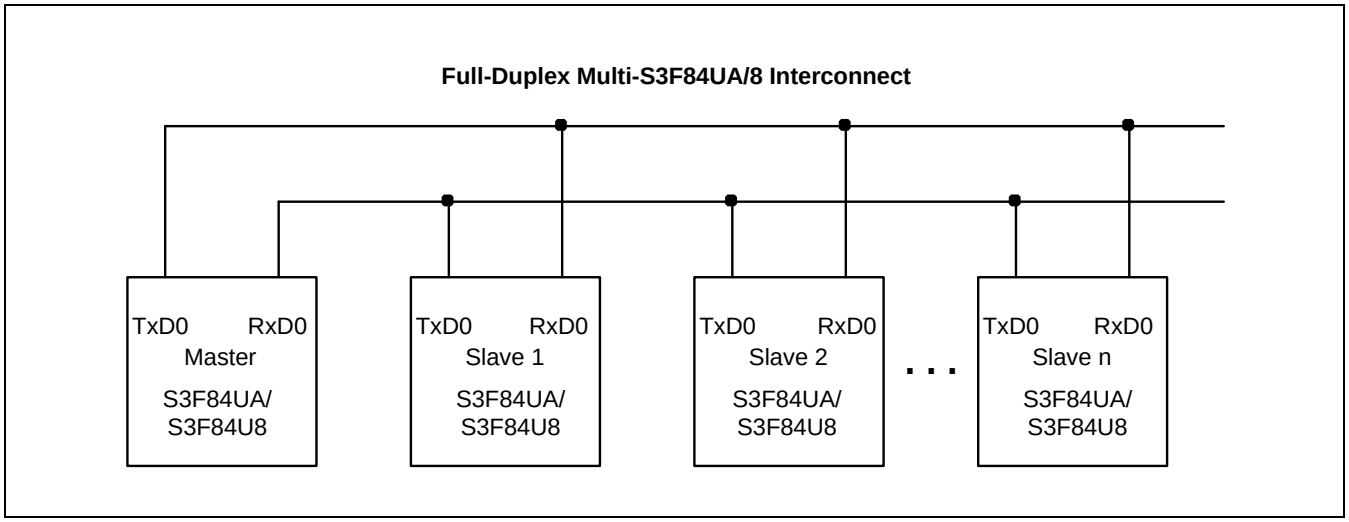


图 18-10 多机串行通讯连接示例

19 UART 1

19.1 概述

UART 1 模块是一个全双工的串行通讯口，其运行模式可编程。
包括一个同步模式和三个 UART (通用异步串行接收/发送) 模式：

- 串行I/O口，波特率为 $f_U/(16 \times (BRDATA0+1))$
- 8 位 UART 模式，波特率可设置
- 9 位 UART 模式，波特率固定为：$f_U/16$
- 9 位 UART 模式，波特率可设置

UART 1 的接收和发送缓冲都是通过数据寄存器 UDATA1 (地址：F4H，Set 1，Bank 0) 来实现的。写 UART 数据寄存器时，数据装载到发送缓冲中；读 UART 1 数据寄存器时，数据从 UART 1 的接收缓冲器中读出。发送缓冲器和接收缓冲器在物理上是分开的。

访问接收数据缓冲器(移位寄存器)时，在上一个接收的数据还读出之前，可以开始下一个数据的接收。因此，如果在下一个数据完成接收之前，上一个接收的数据还没有读出的话，则上一个数据会丢失(过载错误)。

在所有运行模式下，当任何指令(通常是一个写指令)将 UDATA1 作为目标地址写入数据时，则立即开始发送。
在模式 0 下，只有当接收中断标志位 (UART1CONH.0) 为“0”且接收使能位 (UART1CONH.4) 为“1”时，才开始接收串行数据。在模式 1，2 和 3 下，当接收使能位 (UART1CONH.4) 为“1”时，任何时刻只要收到数据起始位(“0”)则立即开始接收数据。

UART 1 模块编程的基本步骤为：

1. 通过设置 P4CONH/L 为相应的值，将 P4.5 和 P4.4 设为 UART 1 功能引脚(RXD1 (P4.5)，TXD1 (P4.4))。
2. 设置 UART1CONH/L 控制寄存器来选择 UART 1 的 I/O 口模块。
3. 需要中断时，将 UART 1 中断使能位 (UART1CONH.1 位或 UART1CONL.1 位)设置为“1”。
4. 需要传送数据时，将需要传送的数据写入到 UDATA 1，则开始传送。
5. 当发送/接收完一个数据后，UART 1 标志位 (UART1CONH.0 位或 UART1CONL.0 位)被置为“1”，同时产生一个相应的发送/接收中断。

19.1.1 UART 1 高字节控制寄存器 (UART1CONH)

UART 0 的高字节控制寄存器是 UART1CONH，地址为：F2H，Set 1，Bank 0。主要实现以下功能：

- 运行模式和波特率选择
- 多机通讯和中断控制
- 串行接收使能/禁止控制
- 模式 2 和模式 3 下发送和接收时第 9 位数据位的位置。
- UART 1 接收中断控制

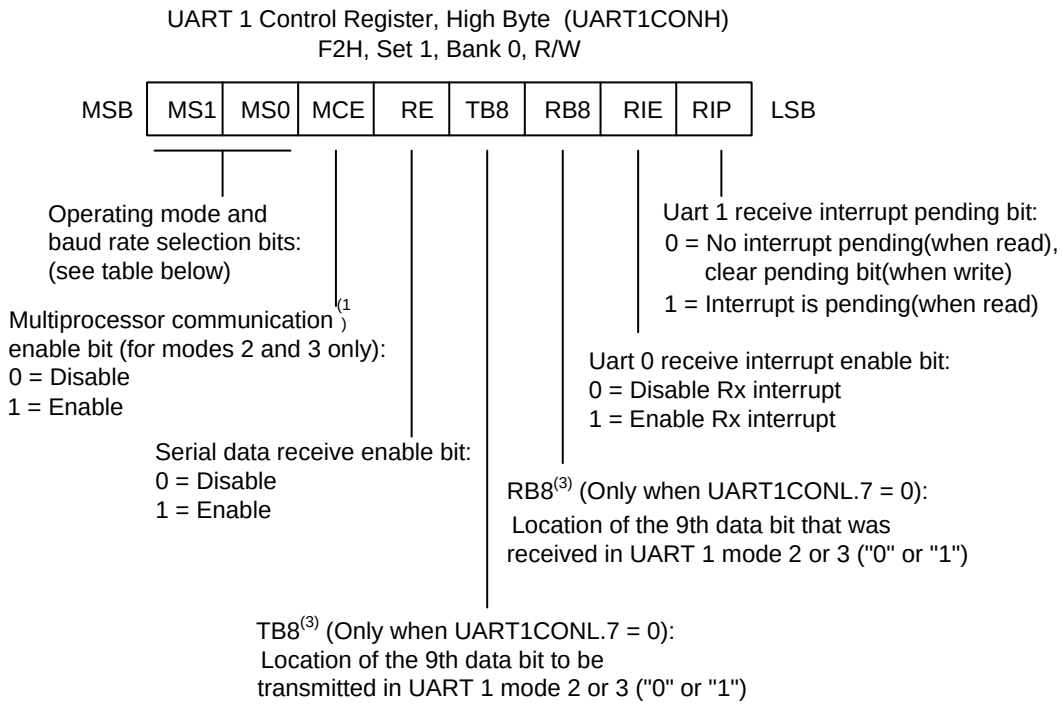
复位后，将清除 UART1CONH 至“00H”。因此，用户如果需要使用 UART 1 模块，应设定 UART1CONH 为适当的值。

19.1.2 UART 1 低字节控制寄存器 (UART1CONL)

UART 1 的低字节控制寄存器是 UART1CONL，地址为：F3H，Set 1，Bank 0。主要实现以下功能：

- UART 1 发送和接收的校验位选择
- UART 1 时钟选择
- UART 1 发送中断控制

复位后，将清除 UART1CONL 至“00H”。因此，用户如果需要使用 UART 1 模块，应设定 UART1CONL 为适当的值。

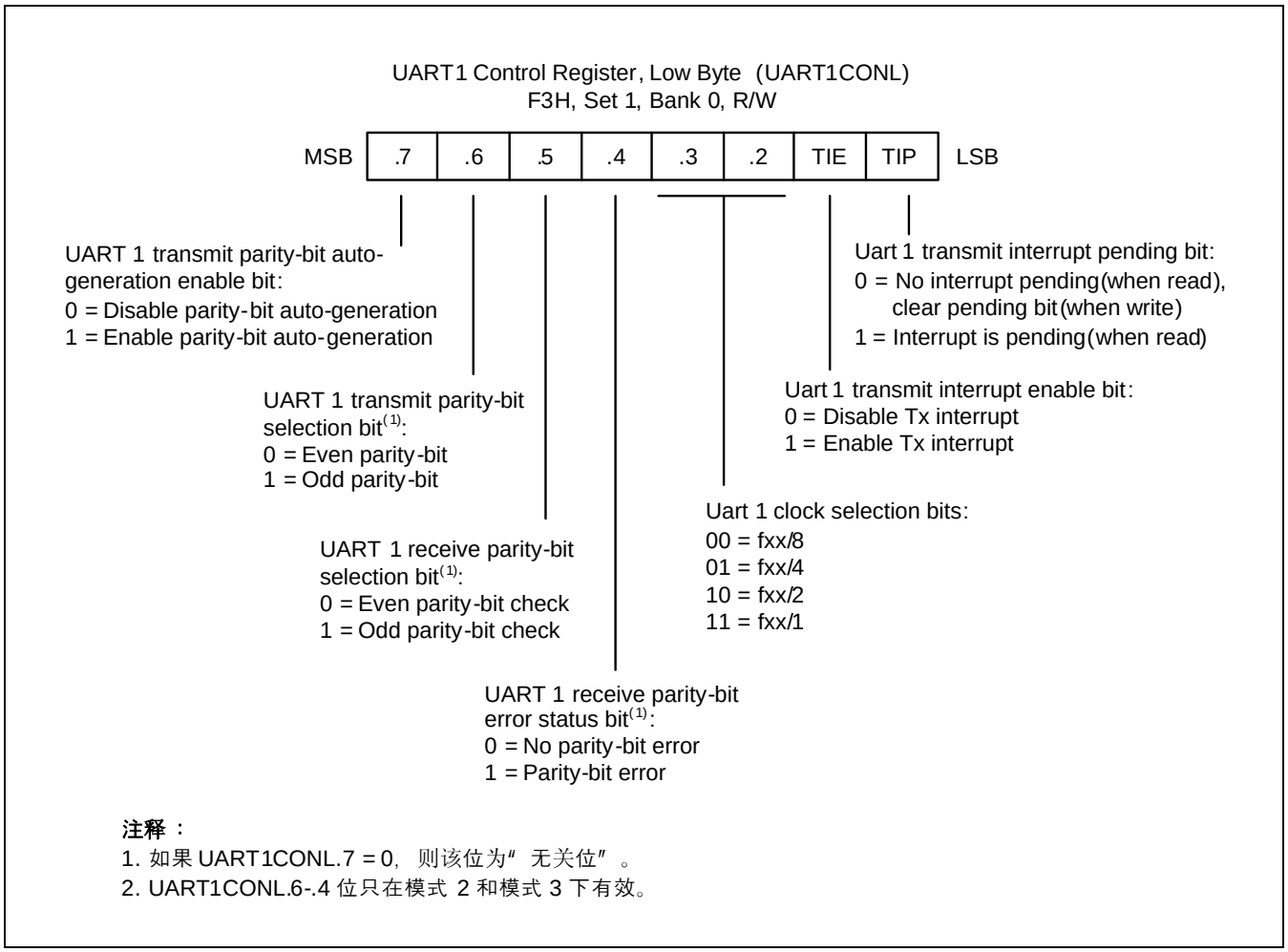


MS1	MS0	Mode	Description(2)	Baud Rate
0	0	0	Shift register	(fu/(16 x (BRDATA0 + 1)))
0	1	1	8-bit UART	(fu/(16 x (BRDATA0 + 1)))
1	0	2	9-bit UART	(fu /16)
1	1	3	9-bit UART	(fu/(16 x (BRDATA0 + 1)))

注释:

1. 在模式 2 或者模式 3 下，如果 UART1CONH.5 位设置为“ 1”，则当收到的第 9 位数据位为“ 0”时，不会产生接收中断。在模式 1 模式下，如果 UART1CONH.5 位的值设置为“ 1”，则在没有收到有效的停止位的时候，不会产出接收中断。在模式 0 下，应将 UART1CONH.5 位设为“ 0”。
2. 有关 UART 的 8 位和 9 位的的描述中，其中的 8 位和 9 位是指有效数据的位数，不包括发送和接收时的开始位和停止位。
3. 如果 UART1CONL.7 = 1，表示该位为“ 无关位”。

图 19-1 UART 1 高字节控制寄存器 (UART1CONH)



注释：

1. 如果 UART1CONL.7 = 0，则该位为"无关位"。
2. UART1CONL.6-.4 位只在模式 2 和模式 3 下有效。

图 19-2 UART 1 低字节控制寄存器 (UART1CONL)

19.1.3 UART 1 中断标志位

在模式 0 下，当收到第 8 位数据位移位后，接收中断标志位 UART1CONH.0 将置为“1”。在模式 1 下，UART1CONH.0 位在停止位的移位时间的中间点处置“1”。在模式 2 或模式 3 下，在接收到 RB8 位的移位时间的中间点处，UART1CONH.0 位将置为“1”。当 CPU 响应了接收中断后，UART1CONH.0 位必须在中断服务程序中由软件清零。

在模式 0 下，当第 8 位发送数据位移位后，发送中断标志位 UART1CONL.0 位被置为“1”。在模式 1，2 或 3 下，UART1CONL.0 位在停止位开始发送时置位。当 CPU 响应了发送中断后，UART1CONL.0 位必须在中断服务程序中由软件清零。

19.1.4 UART 1 数据寄存器 (UDATA1)

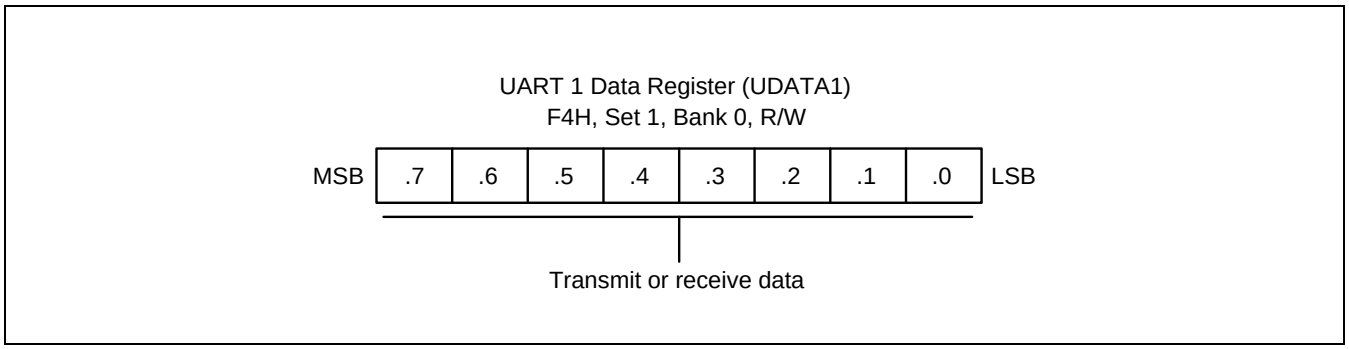


图 19-3 UART 1 数据寄存器 (UDATA1)

19.1.5 UART 1 波特率数据寄存器 (BRDATA1)

通过设置 UART 1 波特率数据寄存器 BRDATA1，可以设定 UART 1 的时钟速率(即波特率)。

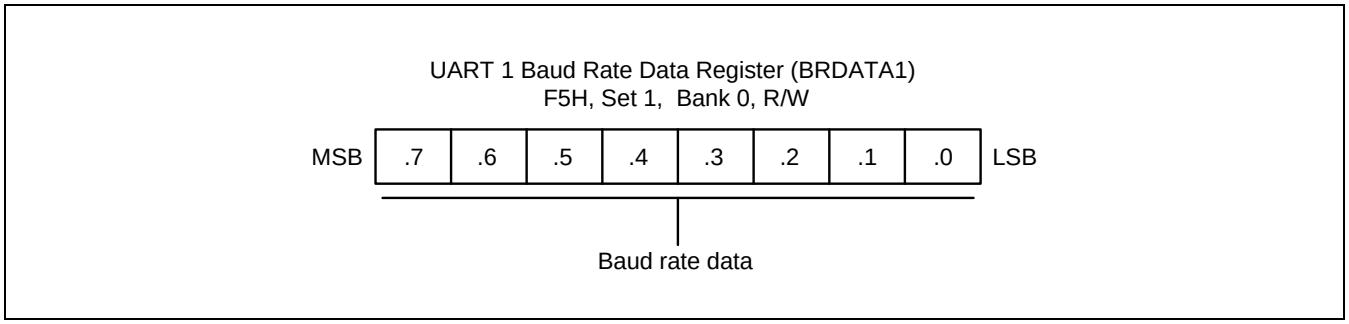


图 19-4 UART 1 波特率数据寄存器 (BRDATA1)

19.1.6 波特率计算

模式 0 的波特率计算

在模式 0 下，波特率由波特率数据寄存器 BRDATA1 (地址：F5H，Set 1，Bank 0) 的值决定，模式 0 的波特率为：$f_U/(16 \times (BRDATA1 + 1))$

模式 2 下波特率的计算

模式 2 下的波特率固定为 $f_U/16$，即f_U 信号16分频。

模式 1 和模式 3 模式下波特率的计算

在模式 1和模式 3 下，波特率由波特率数据寄存器，BRDATA1 (地址：F5H，Set 1，Bank 0) 的值决定，模式 1和模式 3 下的波特率为：$f_U/(16 \times (BRDATA1 + 1))$

表 19-1 利用 BRDATA1 可以产生的常用波特率表

模式	波特率	UART 时钟(f_U)	BRDATA1	
			十进制	十六进制
模式 2	0.5MHz	8MHz	x	x
模式 0 模式 1 模式 3	230,400Hz	11.0592MHz	02	02H
	115,200Hz	11.0592MHz	05	05H
	57,600Hz	11.0592MHz	11	0BH
	38,400Hz	11.0592MHz	17	11H
	19,200Hz	11.0592MHz	35	23H
	9,600Hz	11.0592MHz	71	47H
	4,800Hz	11.0592MHz	143	8FH
	62,500Hz	10MHz	09	09H
	9,615Hz	10MHz	64	40H
	38,461Hz	8MHz	12	0CH
	12,500Hz	8MHz	39	27H
	19,230Hz	4MHz	12	0CH
	9,615Hz	4MHz	25	19H

19.1.7 模块框图

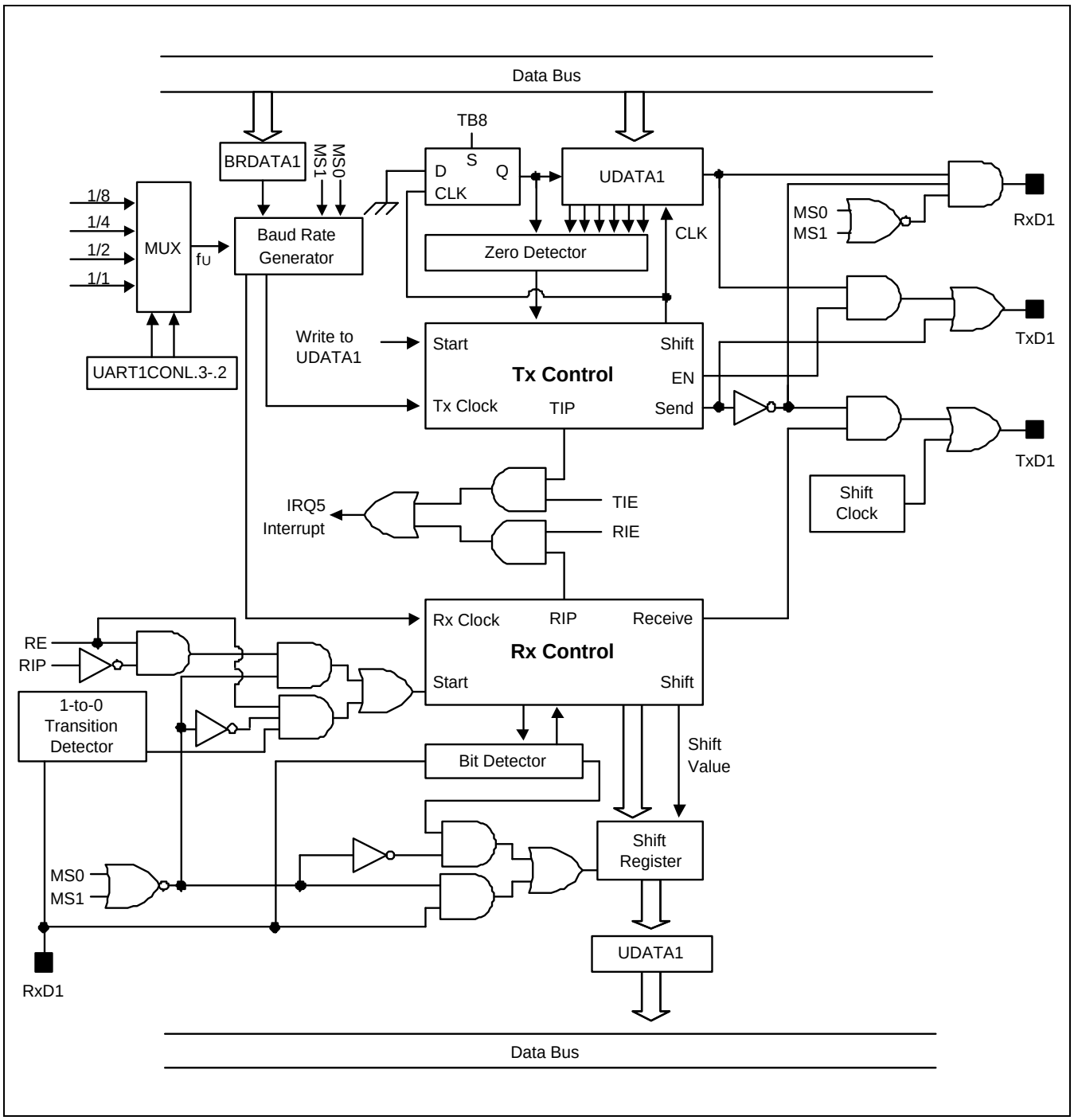


图 19-5 UART 1 功能模块框图

19.1.8 UART 1 模式 0 功能描述

在模式 0 下，UART 1 通过 R×D1 (P4.5) 管脚进行接收和发送数据，T×D1 (P4.4) 输出数据移位时钟信号。数据接收和发送的长度都是 8 位，首先发送(或接收)的是数据的最低位 (LSB)。

模式 0 数据发送流程：

1. 设置 UART1CONL.3 和 .2 位，选择 UART 1 的时钟源。
2. 设置 UART 1 发送时奇偶校验自动生成使能或禁止位 (UART1CONL.7)。
3. 设置 UART1CONH.7 和 .6 位为“00B”，选择模式 0。
4. 将需要发送的数据写入移位寄存器 UDATA1 (地址：F4H，Set1，Bank 0)，数据开始发送。

模式 0 数据接收流程：

1. 设置 UART1CONL.3 和 .2 位，选择 UART 1 的时钟源。
2. 设置 UART 1 发送时奇偶校验自动生成使能或禁止位 (UART1CONL.7)。
3. 设置 UART1CONH.7 和 .6 位为“00B”，选择模式 0。
4. 往 UART1CONH.0 位写“0”清除接收中断标志位 (UART1CONH.0)。
5. 设置 UART 1 接收使能位 (UART1CONH.4) 为“1”，使能 UART 1 接收。
6. 移位时钟开始输出到 TxD1 (P4.4) 管脚，同时开始从 RxD1 (P4.5) 管脚上读取数据。
   当 UART1CONH.1 位置为“1”"时，产生一个 UART 1 接收中断。

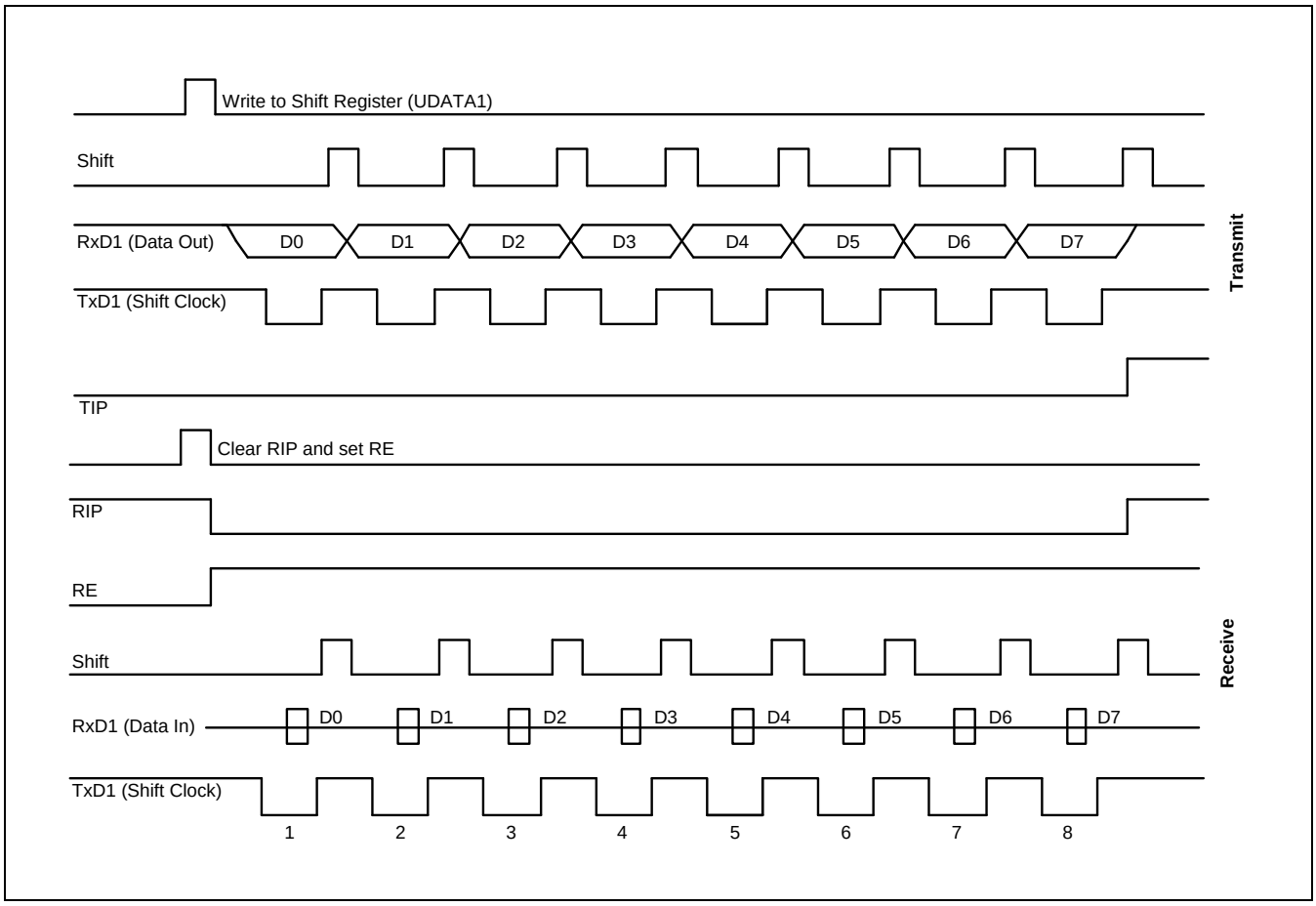


图 19-6 UART 1 模式 1 运行时序图

19.1.9 UART 1 模式 1 功能描述

在模式 1 下，发送(通过 TxD1 (P4.4) 管脚) 或者接收 (通过 RxD1 (P4.5) 管脚)的数据帧总长度是 10 位。每一个数据帧由三部分组成：

- 起始位 (“0”)
- 8 位有效数据 (最低位优先)
- 停止位 (“1”)

模式 1 的波特率是可变的。

19.1.9.1 模式 1 数据发送流程

1. 设置 UART1CONL.3 和 .2 位，选择 UART 1 时钟源。
2. 设置 UART 1 发送时奇偶校验自动生成使能或禁止位 (UART1CONL.7)。
3. 通过设置 8 位 BRDATA1 寄存器设定波特率。
4. 设置 UART1CONH.7-.6 位为“01B”，选择模式 1(8 位 UART)。
5. 将需要发送的数据写入移位寄存器 UDATA1 (地址：F4H，Set 1，Bank 0)，数据开始发送。
   起始位和停止位由硬件自动产生。

19.1.9.2 模式 1 数据接收流程

1. 设置 UART1CONL.3 和 .2 位，选择 UART 1 时钟源。
2. 设置 UART 1 发送时奇偶校验自动生成使能或禁止位 (UART1CONL.7)。
3. 通过设置 8 位 BRDATA1 寄存器设定波特率。
4. 选择模式 1 模式并设置 UART1CONH 中的 RE (接收使能) 位为“1”。
5. 当 RxD1 (P4.5) 管脚上检测到起始条件(低电平“0”)时，UART 1 模块开始进行串行数据接收操作。

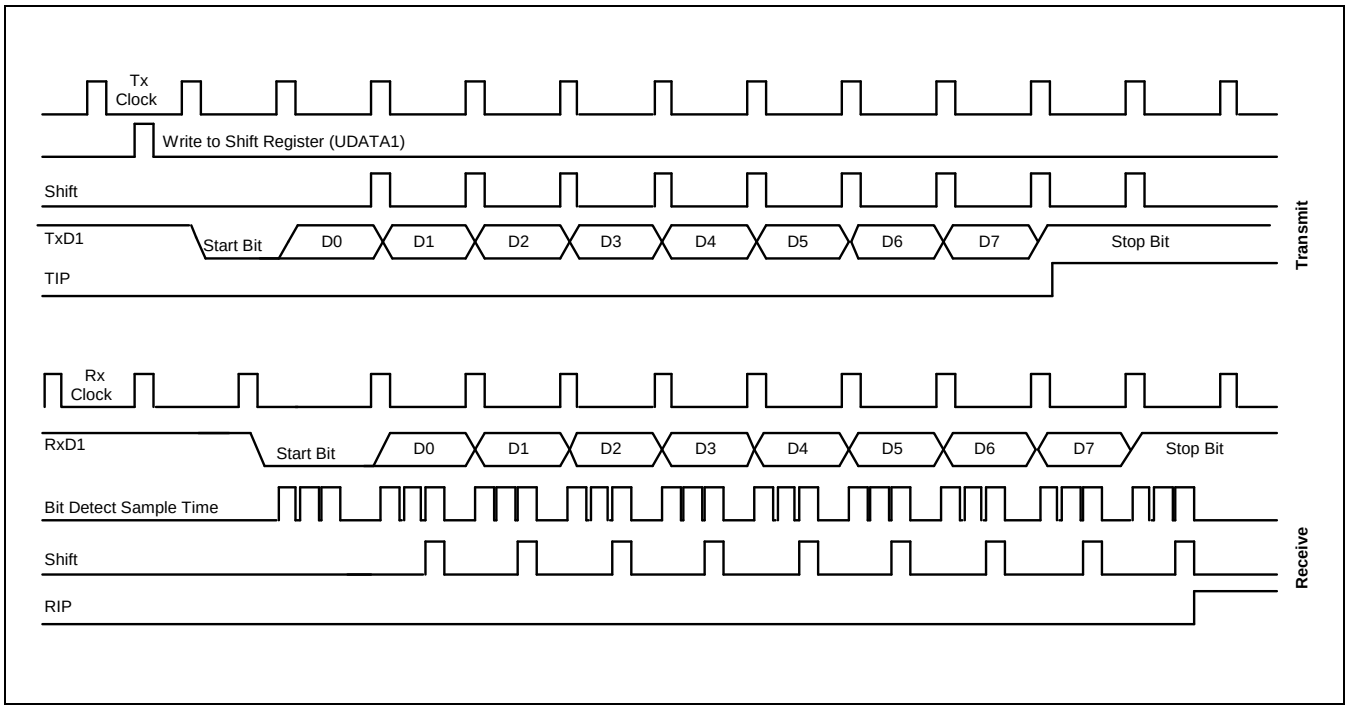


图 19-7 UART 1 模式 1 运行时序图

19.1.10 UART 1 模式 2 功能描述

在模式 2 下，发送(通过 T×D1 (P4.4)) 或者接收 (通过 R×D1 (P4.5))的数据帧总长度是 11 位。
每一个数据帧由四部分组成：

- 起始位 (“0”)
- 8 位有效数据 (最低位优先)
- 可编程设置的第 9 位数据位
- 停止位 (“1”)

发送时，将需要发送的第 9 位数据的值“0”或“1”写入到 TB8 位 (UART1CONH.3)。接收时，接收到的第9位数据存放在 RB8 位 (UART1CONH.2)，同时停止位被忽略。模式 2 的波特率为 $f_U/16$。

19.1.10.1 模式 2 数据发送流程

1. 设置 UART1CONL.3 和 .2 位，选择 UART 1 时钟源。
2. 设置 UART 1 发送时奇偶校验自动生成使能或禁止位 (UART1CONL.7)。
3. 设置 UART1CONH.7-.6 位为“10B”，选择模式 2 (9 位 UART)，同时将需要发送的第 9 位数据“0”或“1”写入TB8。
4. 将需要发送的数据写入移位寄存器 UDATA1 (地址：F4H，Set 1，Bank 0)，数据开始发送。

19.1.10.2 模式 2 数据接收流程

1. 设置 UART1CONL.3 和 .2位，选择 UART 1 时钟源。
2. 设置 UART 1 发送时奇偶校验自动生成使能或禁止位 (UART1CONL.7)。
3. 选择模式 2 并设置 UART1CONH 中的 接收使能位 (RE) 为“1”。
4. 当 RxD1 (P4.5) 管脚上的电平变为低时，开始接收数据。

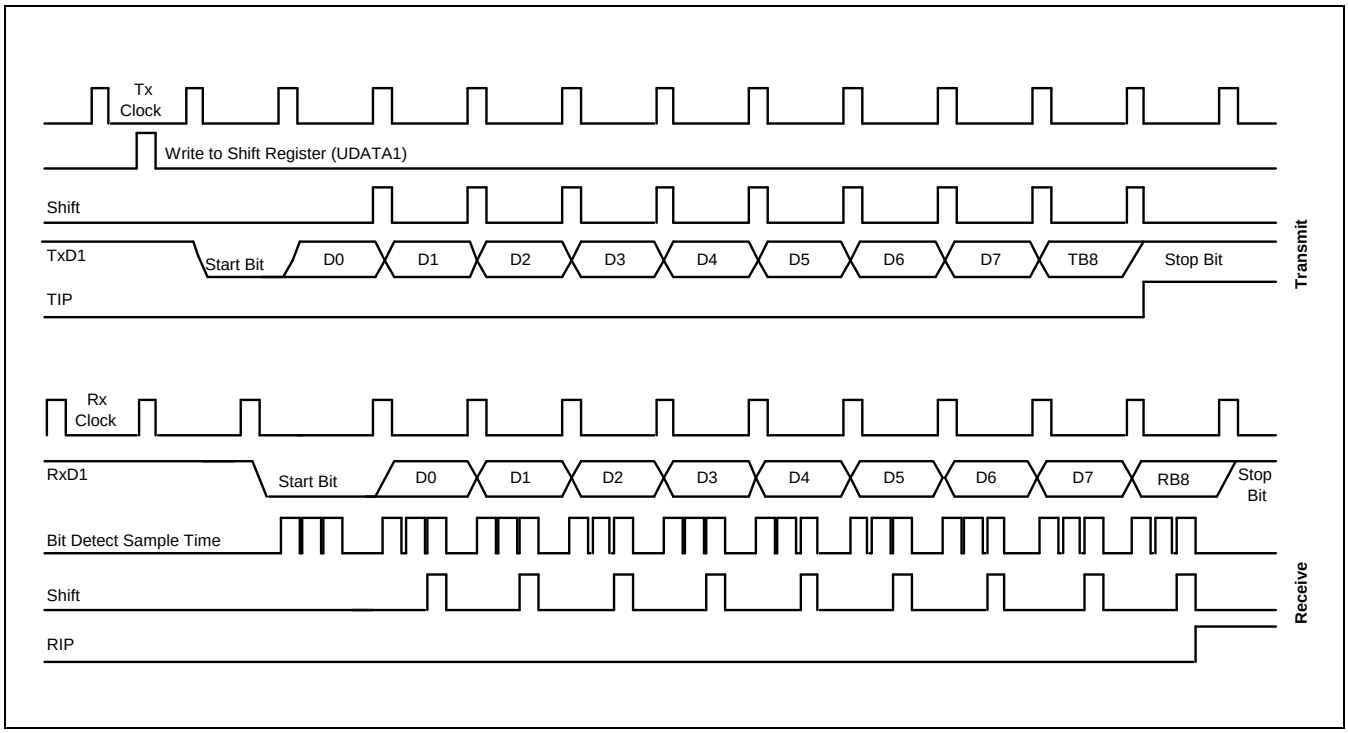


图 19-8 UART 1 模式 2 运行时序图

19.1.11 UART 1 模式 3 功能描述

在模式 3 下，发送(通过 T×D1 (P4.4) 管脚) 或者接收 (通过 R×D1 (P4.5) 管脚)的数据帧总长度是 11 位。
除了模式 3 的波特率可以调节之外，其余的功能模式 3 和模式 2 是完全相同的。每一个数据帧由四部分组成：

- 起始位 (“0”)
- 8 位有效数据 (最低位优先)
- 可编程设置的第 9 位数据位
- 停止位 (“1”)

19.1.11.1 模式 3 数据发送流程

1. 设置 UART1CONL.3 和 .2 位，选择 UART 1 时钟源。
2. 设置 UART 1 发送时奇偶校验自动生成使能或禁止位 (UART1CONL.7)。
3. 设置 UART1CONH.7-.6 位为“11B”，选择模式 3 (9 位 UART)，同时将需要发送的第 9 位数据写入 TB8 (UART1CONH.3)。
4. 将需要发送的数据写入移位寄存器 UDATA0 (地址：F4H，Set 1，Bank 0)，数据开始发送。

19.1.11.2 模式 3 数据接收流程

1. 设置 UART1CONL.3 和 .2 位，选择 UART 1 时钟源。
2. 设置 UART 1 发送时奇偶校验自动生成使能或禁止位 (UART1CONL.7)。
4. 选择模式 3 并设置 UART1CONH 寄存器中的 RE (接收使能) 位为“1”。
5. 当 RxD1 (P4.5) 管脚上的电平变为低时，开始接收数据。

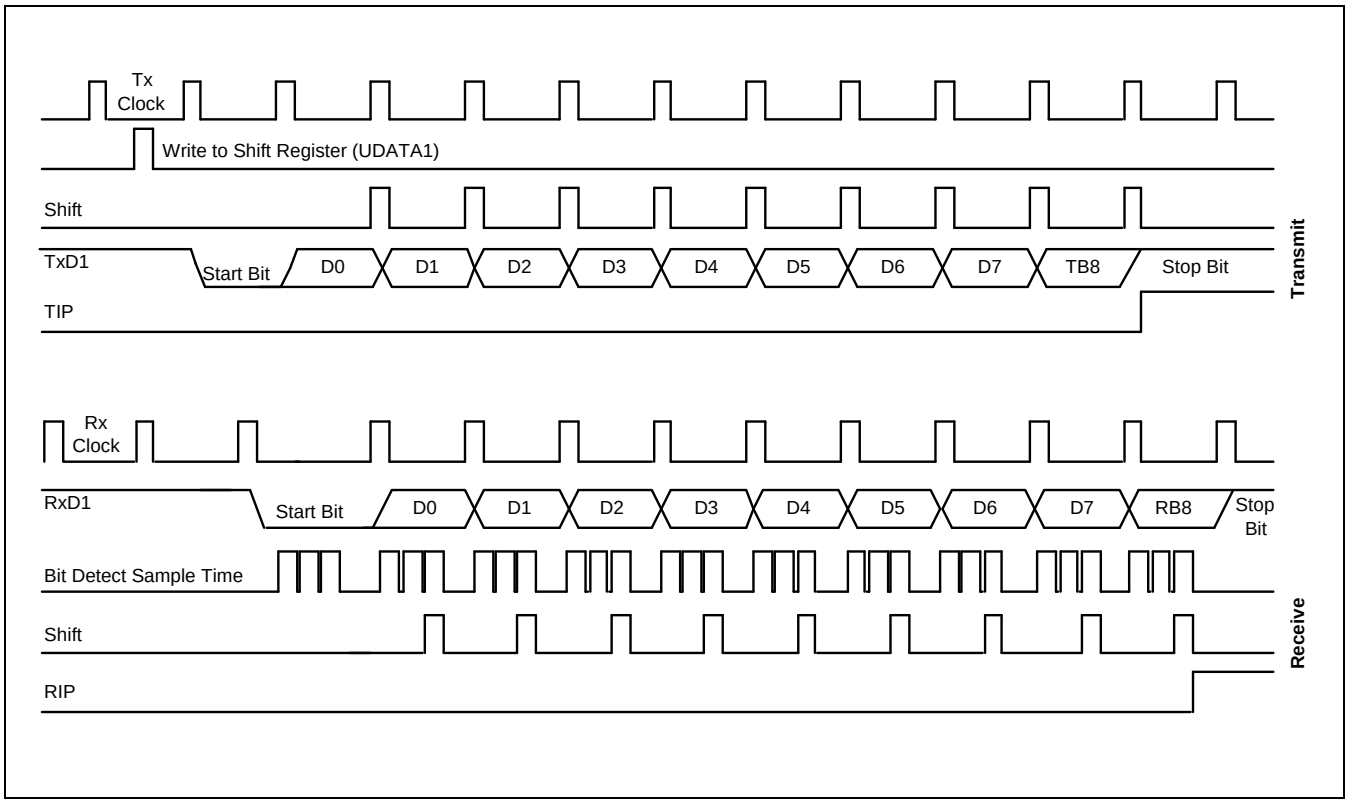


图 19-9 UART 1 模式 3 运行时序图

19.1.12 多机串行通讯的配置

S3F8- 系列的多机串行通讯功能的特点是：在一个由多个 S3F84UA/F84U8 组成的串行通信系统中，
允许一个S3F84UA/F84U8 作为“主机”发送一组多帧串行数据给另一个的“从机”设备，
而不会影响连接在同一个串行线上的其他从机设备。

这个功能只有在 UART 的模式 2 和模式 3 下才有。在模式 2 和模式 3 下，收到的数据共 9 位。
第 9 位写入到 RB8 (UART1CONH.2)。接收操作在收到停止位后完成。
因此可以编程实现这样的功能：收到停止位后，只有当 RB8 =“1”时才产生中断。

为了实现这个功能，需要设置 UART1CONH 寄存器中的 MCE 位，当设置 MCE 位为“1”时，
接收的串行数据帧中的第 9 位数据 =“0”时不会产生中断。在这种情况下，就可以使用第 9
位来简单的区分地址和数据。

19.1.13 主机/从机交互协议的一个例子

当主机希望给串接在同一条串行总线上的很多从机中的一个从机发送数据块时，它首先发送地址信息来选中目标从机。注意在这种情况下，一个地址帧和一个数据帧是不同的，地址帧的第 9 位数据为“1”，而数据帧的第 9 位数据为“0”。

地址帧会让所有的从机都产生中断，并被每个从机都接收到，则每个从机就可以通过检查接收到的数据来判断是否被选中。被选中的目标从机会将 MCE 位清为“0”并准备开始接收数据。

没有被选中的从机的 MCE 位继续保持为“1”，保持正常运行，并忽略串行总线上的数据帧。

MCE 位在模式 0 下没有用处。在模式 1 下，它可以用来检测停止位的有效性。在模式 1 下接收数据时，
如果设置 MCE 位为“1”，只有接收到的停止位有效(为“1”)时，才可以产生接收中断。

19.1.14 多机通讯的配置流程

多机通讯的配置流程如下：

1. 设置所有的串行总线上的 S3F84UA/F84U8 设备 (主机和从机) 的 UART 1 模式为模式 2 或者模式 3。
2. 设置所有从机设备的 MCE 位为“1”。
3. 主机的传输协议为:
   - 第一个字节：地址
     - 标识目标从机设备地址 (第 9 位 =“1”)
   - 后续字节：数据
     - (第 9 位 =“0”)
4. 由于第 9 位为“1”，当所有的目标从机设备接收到第一个字节时都会产生中断。
   目标从机收到第一个字节后，将收到的地址与自己的地址进行比较，确认被选中后，则将 MCE 位清除为“0”来准备接受数据。其他的从机继续正常运行。

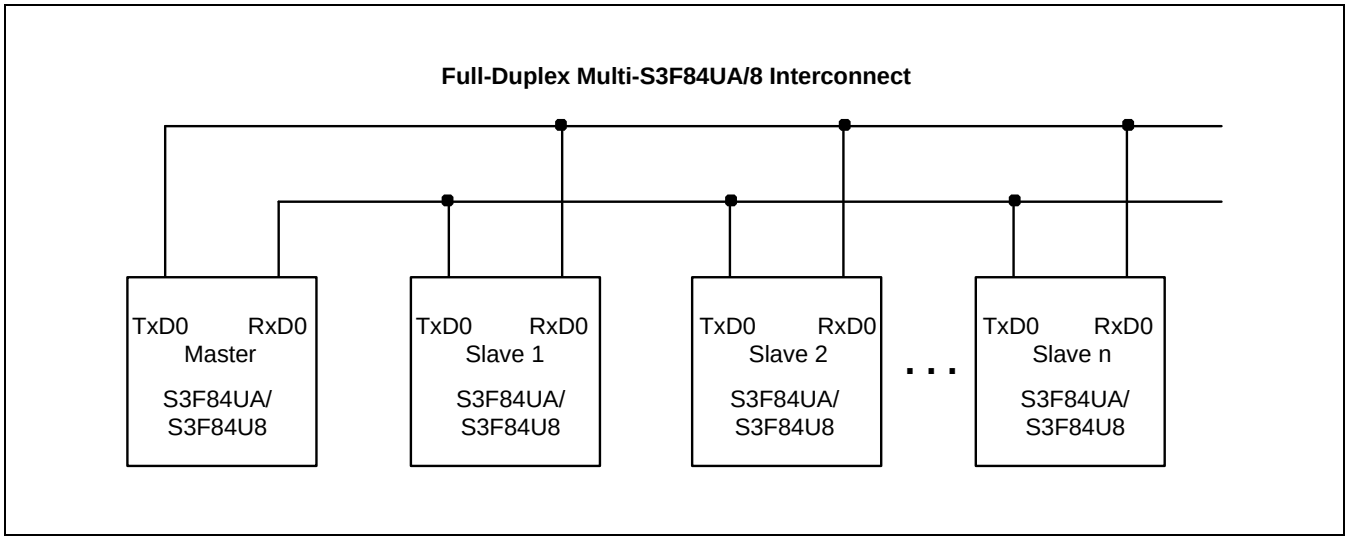


图 19-10 多机串行通讯连接示例

20 PATTERN GENERATION 模块

20.1 概述

20.1.1 PATTERN GNERATION 流程

按照下面的流程，可以通过 P0.0–P0.7 输出最多 8 位的 Pattern 序列。首先，将 PGDATA 修改为期望输出的值，然后设置 PGCON 以使能 Pattern Generation 模块，并选择触发信号。
自此，每当所选的触发信号产生时，PGDATA 的各位将自动出现在 P0.0–P0.7 端口上。

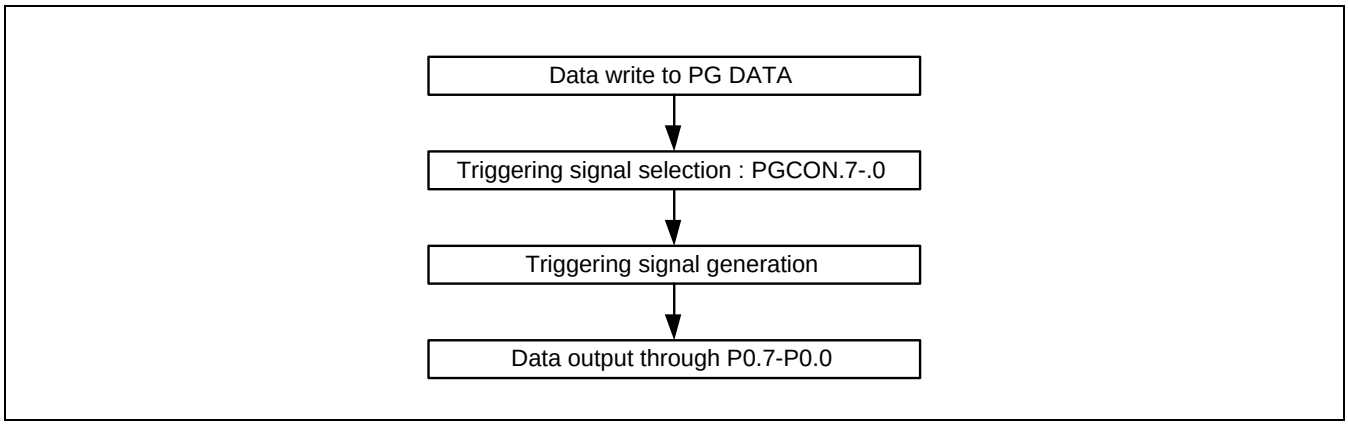


图 20-1 Pattern Generation 流程

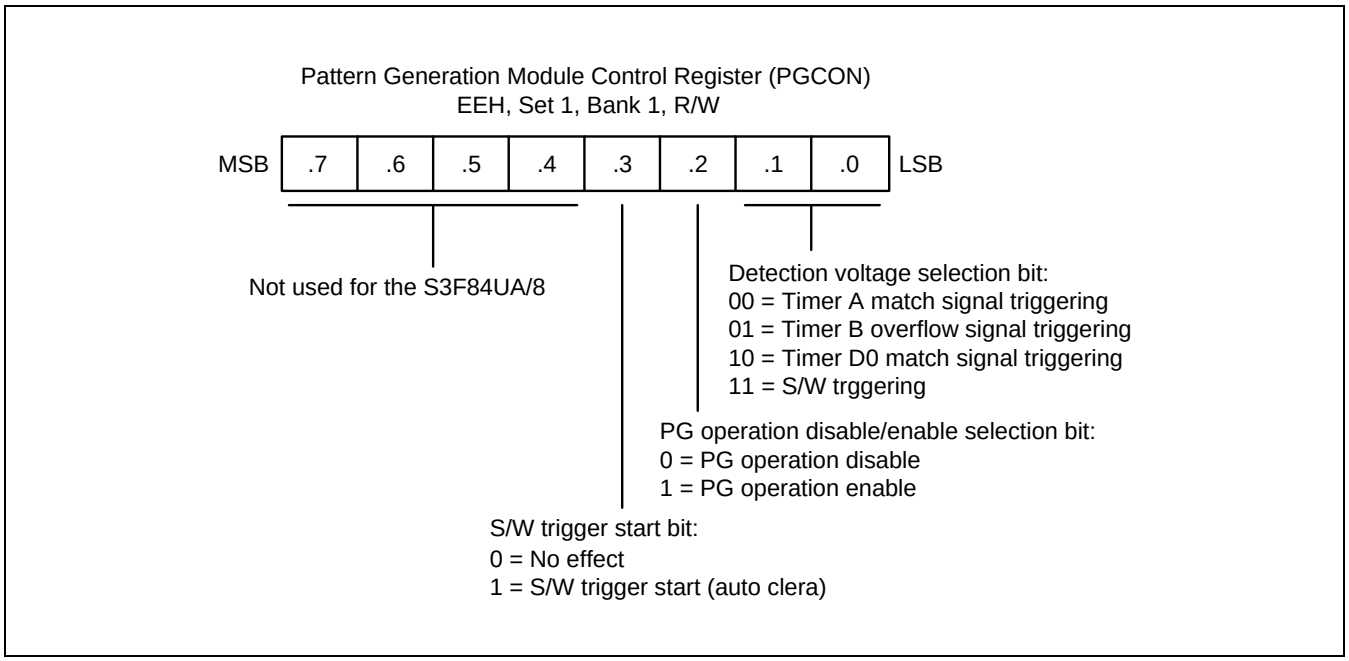


图 20-2 Pattern Generation 控制寄存器(PGCON)

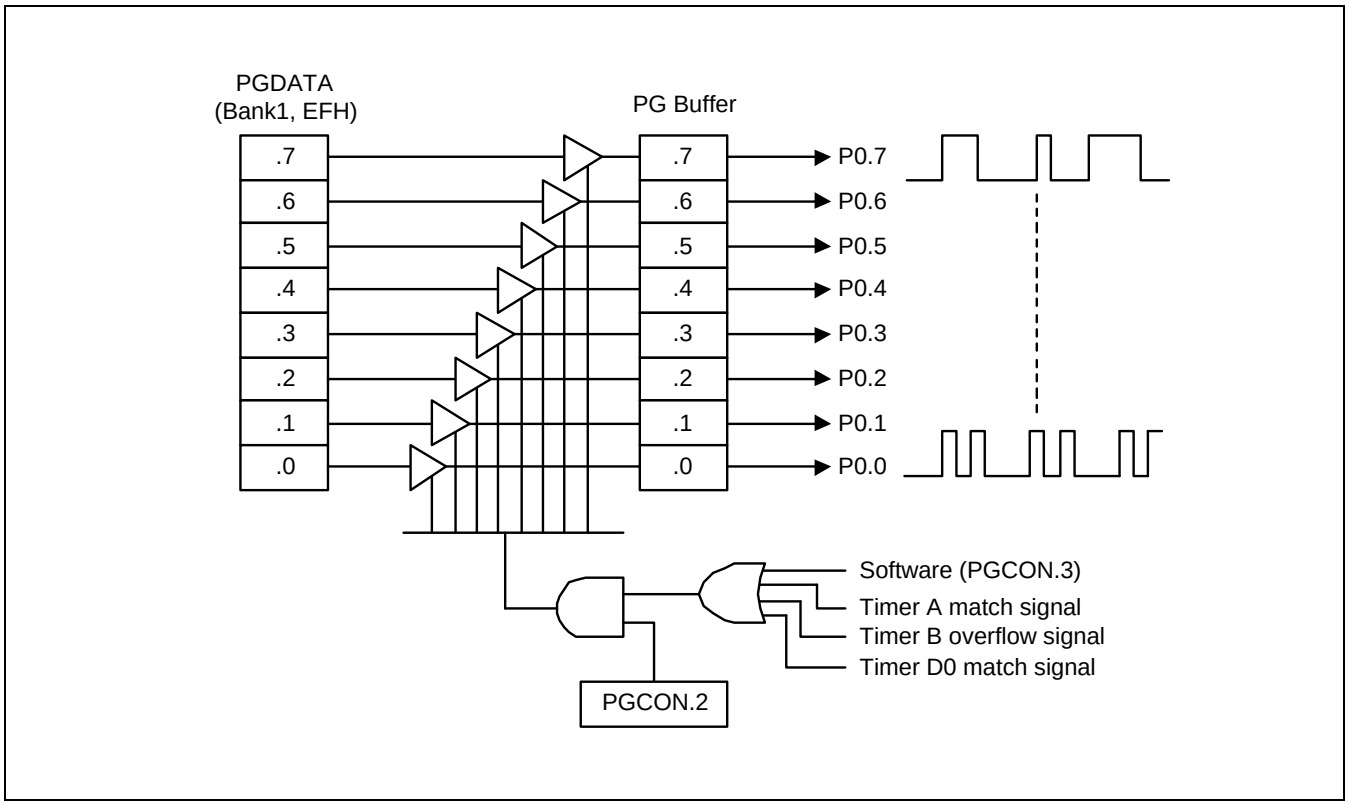


图 20-3 Pattern Generation 控制电路

编程实例 20-1 如何使用 Pattern Generation

```
            ORG    0000h

            ORG    0100h

INITIAL:
            SB0
            LD     SYM, #00h                ; 禁止全局/快速中断 → SYM
            LD     IMR, #01h                ; 使能 IRQ0
            LD     SPH, #0h                 ; 堆栈指针高字节 → SPH
            LD     SPL, #0FFh               ; 堆栈指针低字节 → SPL
            LD     BTCON, #10100011b        ; 禁止看门狗
            LD     CLKCON, #00011000b       ; 不分频(fxx)

            SB1
            LD     P0CONH,#01010101b        ; 使能 PG 输出
            LD     P0CONL,# 01010101b       ; 使能 PG 输出
            SB0

            EI

MAIN:
            NOP
            NOP

            SB1
            LD     PGDATA, #10101010b       ; PG 数据设置
            OR     PGCON, #00000100b        ; 被 Timer A 匹配信号触发，Pattern 序列输出
            SB0

            NOP
            NOP

            JR     T,MAIN

            .END
```

21 嵌入式闪存接口

21.1 概述

S3F84UA/ F84U8 内部有一个片上闪存，可代替掩膜 ROM。该闪存支持“LDC”指令，编程的最小单位是字节，擦除的最小单位是扇区。用户可在任何时间往闪存里写入数据。S3F84UA/ F84U8中 内嵌的 48K/8K 字节闪存有两种编程模式：

- 工具编程模式
- 用户编程模式：参考第 24 章，S3F84UA/F84U8 FLASH MCU。

21.2 用户编程模式

用户编程模式支持扇区擦除，字节编程，字节读取和一种保护模式(Hard Lock 保护)。
只有在工具编程模式下才可以启动读保护，所以，为了使芯片进入独保护状态，用户需要在使用编程工具对芯片采用工具编程模式进行烧写代码时选择读保护选项。

S3F84UA/F84U8 内部集成了自升压电路，可以产生高电压。因此，不需要在 V_{PP} (Test) 管脚上另加 12.5V 的编程电压，只是利用芯片内部逻辑电平就可以完成对闪存的操作。这种模式下对闪存进行编程需要用到一些控制寄存器。用户编程模式下有四种功能 – 编程，读，扇区擦除和 Hard Lock 保护。

注释:

1. 当 CPU 工作在副时钟下时，不能使用用户编程模式。
2. 必须在开始用户编程前执行 DI 指令禁止中断。因为用户编程模式会检查中断请求寄存器（IRQ），一旦检测到有中断发生，即会终止用户编程模式。
3. 即使在 DI 状态下屏蔽掉的中断，一旦产生中断请求时，用户编程模式也会被终止。为了避免这种情况发生，需要通过设置各个外围设备模块的中断使能位来禁止这些中断。

21.2.1 闪存控制寄存器 (用户编程模式)

21.2.1.1 闪存控制寄存器 (FMCON)

FMCON 寄存器只支持用户编程模式下选择闪存操作模式：扇区擦除，字节编程，和对闪存设置 Hard Lock 保护。

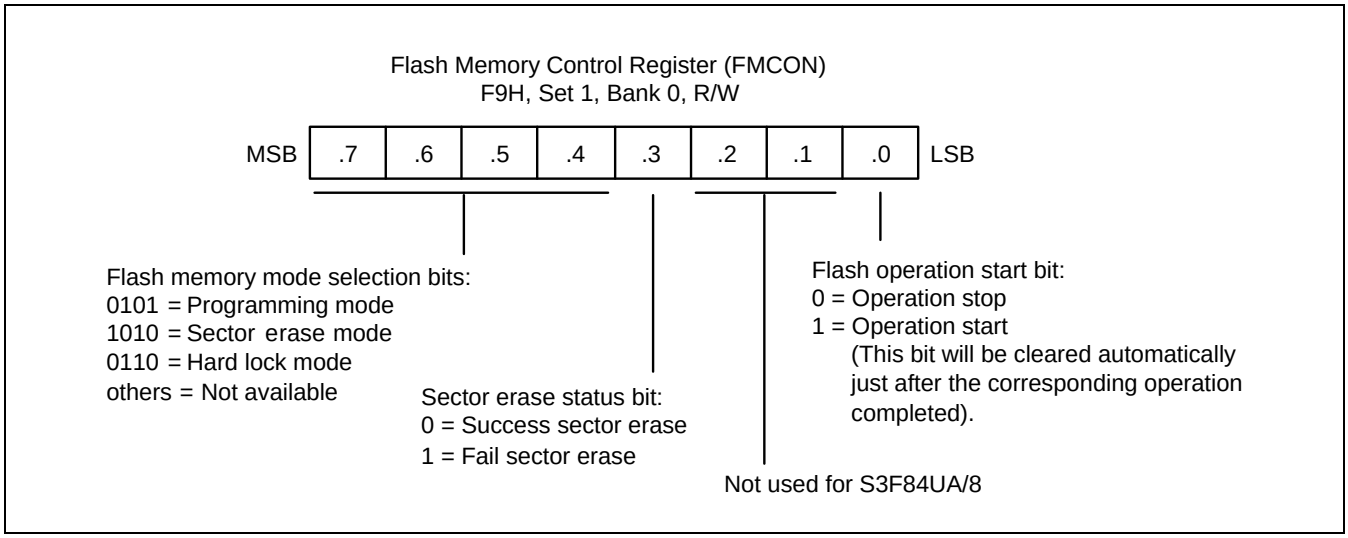


图 21-1 闪存控制寄存器 (FMCON)

FMCON 寄存器中的第 0 位 (FMCON.0 位) 是擦除和 Hard Lock 保护操作的开始位。因此，当用户将 FMCON.0 位设置为“1”时，则开始擦除或 Hard Lock 保护操作。在执行扇区擦除或 Hard Lock 保护后，需要等待一段时间才可以用“LDC”指令对同一扇区区域进行读字节或者写字节的操作。对闪存的字节读写则不需要用到这一控制位。

扇区擦除状态位是只读的。在执行扇区擦除的过程中，即使在 IMR 寄存器中禁止了所有中断源，但是只要具体模块的中断处于使能状态，并且满足中断触发条件而置起中断标志位，仍然会向 CPU 发出中断请求，此时扇区擦除操作停止，CPU 响应中断请求。所以，为了操作的可靠性，必须在擦除操作结束后检查扇区擦除状态位，如果为“0”，则表示操作成功，为“1”则表示操作失败，需要重新擦除。

注释: 如果用户执行扇区擦除操作，而此时 FMUSR 的值不是“A5H”，FMCON.0 位会一直保持“1”。当随后 FMUSR 被置“A5H”，就会立即开始执行擦除操作，而此时的操作扇区可能与之前不同，也就是说会启动一次误擦除操作。因此，为了不影响其它 flash 扇区，当执行扇区擦除时用户应注意 FMUSR 的设置顺序。

21.2.1.2 闪存用户编程使能寄存器

FMUSR 寄存器是为闪存的安全操作设置的。当 CPU 由于电噪声或其它原因致使程序跑飞时，这个寄存器的状态可以避免闪存的误操作。

复位后，由于 FMUSR 的值为“00000000B”，用户编程模式处于禁止状态。如需操作闪存，可通过设置 FMUSR 的值为“10100101B”来使能用户编程模式。除“10100101B”外的其它值，将禁止用户编程模式。

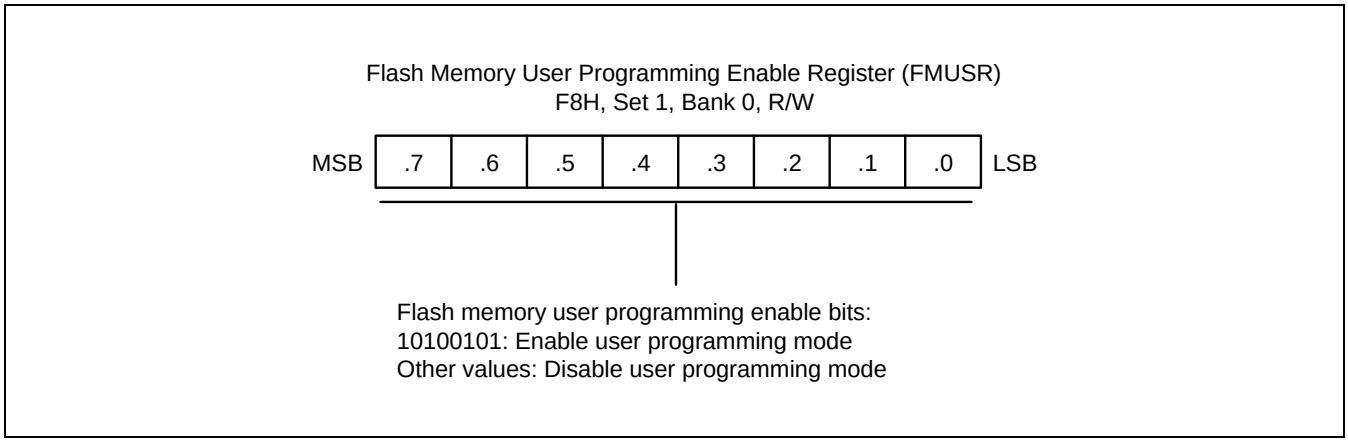


图 21-2 闪存用户编程使能寄存器 (FMUSR)

21.2.1.3 闪存扇区地址寄存器

擦除或编程闪存需要两个 8 位扇区地址寄存器来指定操作地址。
FMSECL(低字节闪存扇区地址寄存器)存储被操作扇区基址的低字节地址，FMSECH(高字节闪存扇区地址寄存器)存储高字节地址。

S3F84UA/F84U8 里分别有 384 和 64 个扇区，所以需要用到 FMSECH。一个扇区由 128 个字节组成。每个扇区的起始地址不是 XX00H 就是 XX80H。所以 FMSECL 寄存器的第 6-0 位不管“1”或“0”都没有意义。 但从操作的简单性来考虑，建议用户还是直接向 FMSECH 和 FMSECL 寄存器写入完整的扇区基址。

当编程闪存时，用户应该在确定扇区基址后再进行编程。如果下一个操作也是写入一字节数据，用户应该核对下一个目的地址是否位于同一个扇区内。如果不在同一扇区，用户应该重新向 FMSECH 和 FMSECL 寄存器中写入新的扇区基址。

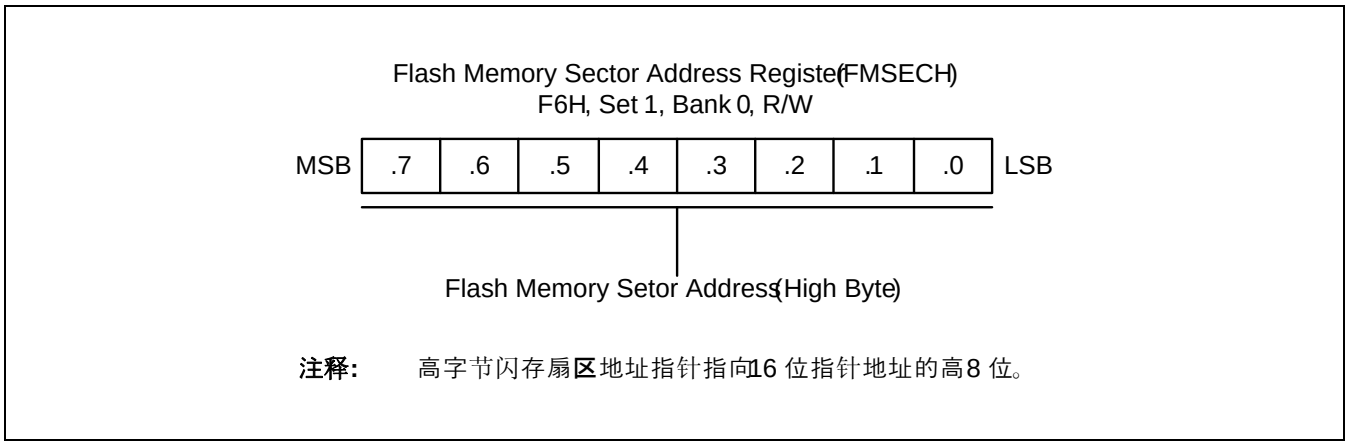


图 21-3 闪存扇区地址寄存器高字节 (FMSECH)

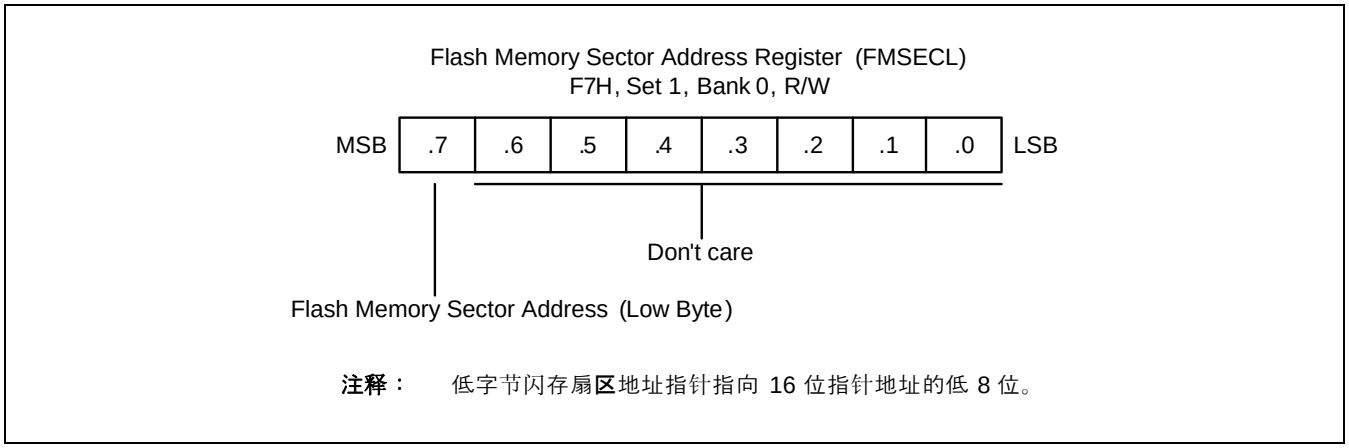


图 21-4 闪存扇区地址寄存器低字节 (FMSECL)

21.3 ISP™ (在板编程) 扇区

位于程序空间的 ISP™ 扇区可以用于存放在板编程程序(通过 I/O 接口实现软件升级的 Boot 程序代码)。出于在板编程程序的安全考虑，ISP™ 扇区的内容不能通过“LDC” 指令擦除或编程操作改变，但是这要通过使能ISP 保护来实现。

当 ISP 使能/禁止位设置为“0”时，即在 Smart Option 中使能 ISP，ISP 扇区才有效。如果用户不想使用 ISP 扇区，只要将 Smart Option 中的 ISP 禁止位设置为“1”，此时这个区域可以用作普通的程序存储空间。工具模式下，ISP 扇区会失去保护，即通过串行编程工具还是可以擦除或编程该区域。

ISP 扇区的大小可以通过 Smart Option 来设置。用户可以根据在板编程代码的大小选择合适的 ISP 扇区尺寸。

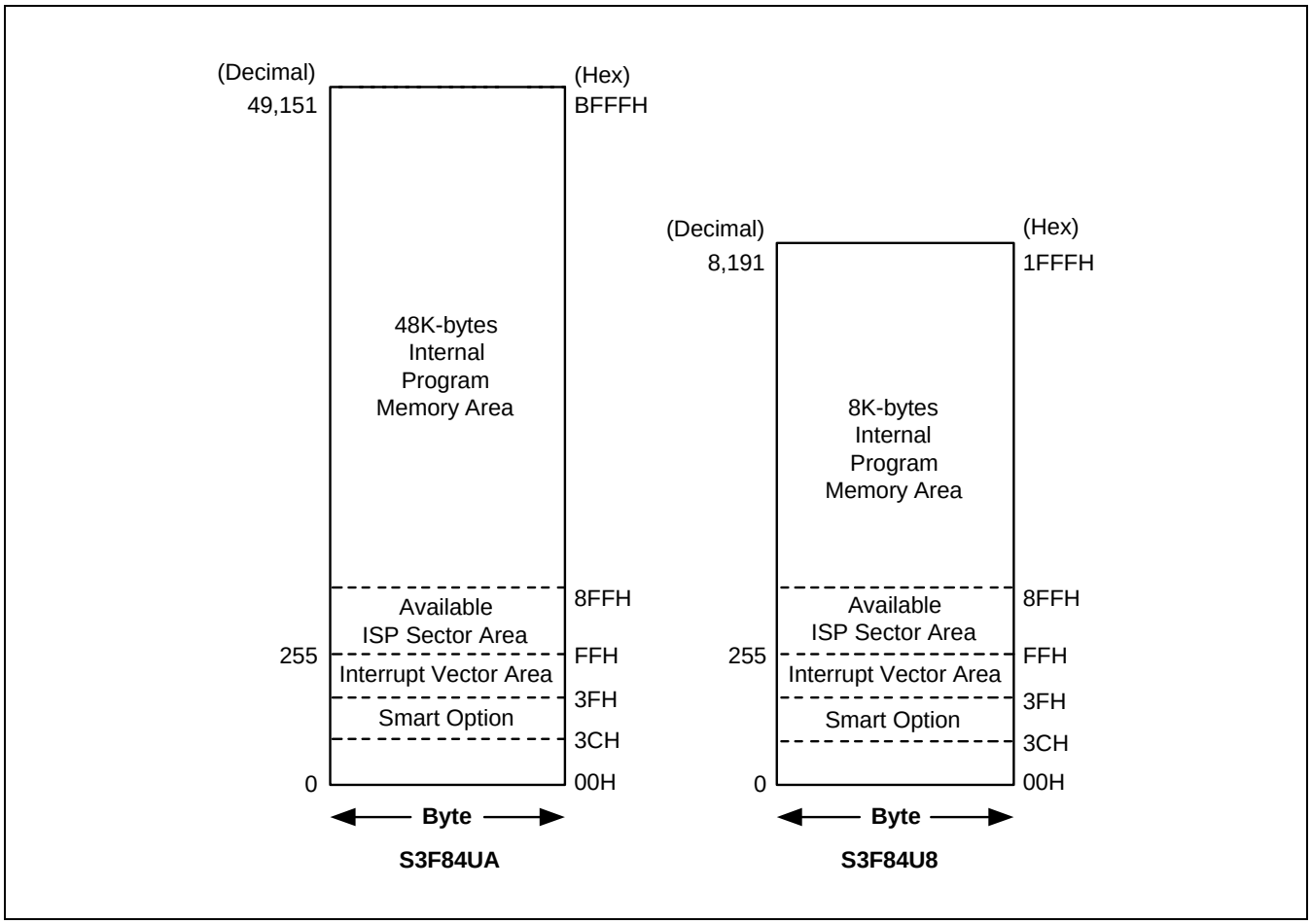


图 21-5 程序存储地址空间

表 21-1 ISP 扇区大小

Smart Option(003EH) ISP 扇区大小选择位			ISP 扇区	ISP 扇区大小
Bit 2	Bit 1	Bit 0		
1	x	x	–	0
0	0	0	100H – 1FFH (256 字节)	256 字节
0	0	1	100H – 2FFH (512 字节)	512 字节
0	1	0	100H – 4FFH (1024 字节)	1024 字节
0	1	1	100H – 8FFH (2048 字节)	2048 字节

注释: 在用户模式下，通过 Smart Option 位（3E.2-3E.0）设置的 ISP 扇区空间内，用 LDC 指令实现的擦除和编程操作都是无效的。

21.3.1.1 ISP 复位向量和 ISP 扇区大小的关系

如果用户将 Smart Option 的 ISP 使能/禁止位设为“0”且复位向量选择位设为“0”，以此使用 ISP 扇区，此时可通过设置 ISP 复位向量地址选择位来选择 CPU 的复位向量地址，*表 21-2* 所示。

表 21-2 复位向量地址

Smart Option (003EH) ISP 复位向量地址选择			上电复位后的复位向量地址	存放 ISP 代码的空间	ISP 扇区大小
Bit 7	Bit 6	Bit 5			
1	x	x	0100H	–	–
0	0	0	0200H	100H – 1FFH	256 字节
0	0	1	0300H	100H – 2FFH	512 字节
0	1	0	0500H	100H – 4FFH	1024 字节
0	1	1	0900H	100H – 8FFH	2048 字节

注释: Smart Option 中 ISP 复位向量选择位（003EH.7 – 003EH.5）和 ISP 保护扇区选择（003EH.2 – 003EH.0）是相互独立的。所以设置的时候要注意一致性。

21.4 扇区擦除

用户只能在用户编程模式下利用扇区擦除操作来部分擦除闪存。在用户编程模式下，擦除闪存的唯一单元是扇区。

S3F84UA/F84U8 的 48K/8K 字节闪存空间被分为 384/64 个扇区。 每个扇区大小为 128 个字节。
因此，在对闪存内单个或多个字节进行编程前，必须先擦除目的地址所在的扇区。

在设置完被擦扇区的扇区地址和置高擦除起始位(FMCON.0)开始擦除后，至少需要 10 ms 来完成擦除操作。
工具编程模式不支持扇区擦除(MDS 模式工具或编程工具)。

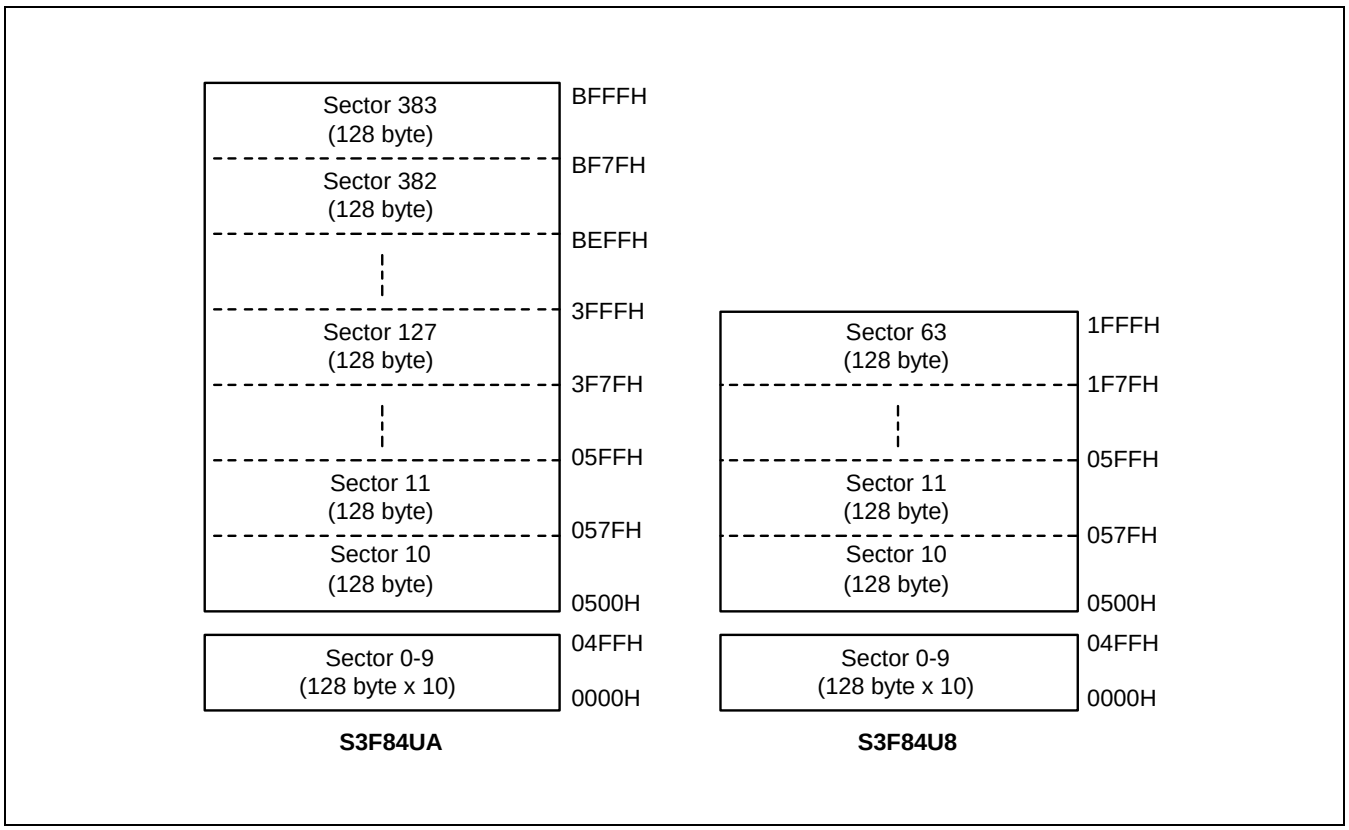


图 21-6 用户编程模式下的扇区

21.4.1 用户编程模式下的扇区擦除流程

1. 设置闪存用户编程使能寄存器 (FMUSR) 为“10100101B”。
2. 设置闪存扇区地址寄存器 (FMSECH 和 FMSECL)。
3. 检查用户 ID (用户设置的任意值，和 FMUSR 功能类似，用来保护闪存区的安全，可选)。
4. 设置闪存控制寄存器 (FMCON) 为“10100001B”。
5. 设置闪存用户编程使能寄存器 (FMUSR) 为“00000000B”。
6. 通过检查“扇区擦除状态位”判断“扇区擦除”操作是否成功。

编程实例 21-1 扇区擦除

```
                •
                •
                SB0
reErase:        LD    FMUSR,Temp0           ; 使能用户编程模式
                                            ; Temp0 =
                                             #0A5H，必须在其他代码段中事先设置
                LD    FMSECH,#10H
                LD    FMSECL,#00H           ; 设置扇区地址 (1000H-107FH)
                CP    UserID_Code,#User_value; 确认用户 ID (用户自设定的任意值)
                JR    NE,Not_ID_Code        ; 不相等的话，跳转到 Not_ID_Code
                LD    FMCON,Temp1           ; 使能擦除模式 & 开始扇区擦除
                                            ; Temp1 =
                                              #0A1H，必须在其他代码段中事先设置
                NOP                         ; 必须增加的空操作
                NOP                         ; 必须增加的空操作
                LD    FMUSR,#0              ; 禁止用户编程模式
                TM    FMCON,#00001000B      ; 检查扇区擦除状态位
                JR    NZ,reErase            ; 如果失败则跳转到 reErase
                •
                •
                •
                •
Not_ID_Code:
                SB0
                LD    FMUSR,#0              ; 禁止用户编程模式
                LD    FMCON,#0              ; 关闭扇区擦除模式
                •
                •
                •
                •
```

注释: 在用户编程模式下，Temp0 ~ Temp1 的数据值必须在其他代码段中事先设置。
Temp0 ~ Temp(n) 变量必须由用户定义。

21.5 编程

扇区擦除后，闪存编程操作是以一个字节为单位来进行的。为了能安全编程，必须将 FMSECH 和 FMSECL 设置为闪存扇区的值。

由“LDC”指令开始编程的写操作。

因为扇区大小为 128 字节，所以在不重写扇区基址的情况下，一次最多只能连续写入 128 个字节，超过部分需要重新设置 FMSECH 和 FMSECL，然后写入。

21.5.1 用户编程模式下编程的流程

1. 编程前必须进行扇区擦除。
2. 设置闪存用户编程使能寄存器 (FMUSR) 为“10100101B”。
3. 设置闪存扇区地址寄存器 (FMSECH 和 FMSECL) 为要写数据的扇区基地址的值。
4. 把闪存的高位地址放进对工作寄存器的高地址寄存器中。
5. 把闪存的低位地址放进对工作寄存器的低地址寄存器中。
6. 把要发送的数据放到工作寄存器中。
7. 检查用户 ID (用户设置的任意值，和 FMUSR功 能类似，用来保护闪存区的安全，可选)。
8. 设置闪存控制寄存器 (FMCON) 为“01010001B”。
9. 用“LDC”指令，通过间接寻址方式把要发送的数据写入闪存地址区。
10. 设置闪存用户编程使能寄存器 (FMUSR) 为“00000000B”，以关闭用户编程模式。

编程实例 21-2 编程

```
                    •
                    •

                    SB0
                    LD      FMUSR,Temp0             ; 使能用户编程模式
                                                    ; Temp0 =
                                                      #0A5H，必须在其他代码段中事先设置
                    LD      FMSECH,#17H
                    LD      FMSECL,#80H             ; 设置扇区地址 (1780H-17FFH)
                    LD      R2,#17H                 ; 设置目标操作地址
                                                      (位于扇区1780H–17FFH 中)
                    LD      R3,#84H
                    LD      R4,#78H                 ; 临时数据
                    CP      UserID_Code,#User_value; 确认用户 ID (用户自设定的任意值)
                    JR      NE,Not_ID_Code          ; 如果不相等，跳转到
                                                      Not_ID_Code
                    LD      FMCON,Temp1             ; 开始编程
                                                    ; Temp1 =
                                                      #51H，必须在其他代码段中事先设置
                    LDC     @RR2,R4                 ; 将 78H 写入 1784H 地址单元中
                    NOP                             ; 必须增加的空操作
                    LD      FMUSR,#0                ; 禁止用户编程模式

                    •
                    •
                    •
                    •
Not_ID_Code:
                    SB0
                    LD      FMUSR,#0                ; 禁止用户编程模式

                    LD      FMCON,#0                ; 关闭用户编程模式

                    •
                    •
                    •
                    •
```

注释: 在用户编程模式下，Temp0 ~ Temp1 的数据值必须在其他代码段中事先设置。
Temp0 ~ Temp(n) 变量必须由用户定义。

21.6 读

由“LDC”指令开始读操作。

21.6.1 用户编程模式下读操作的编程流程

1. 把闪存的高位地址放进对工作寄存器的高地址寄存器中。
2. 把闪存的低位地址放进对工作寄存器的低地址寄存器中。
3. 用“LDC”指令，通过间接寻址方式从闪存区读取数据。

编程实例 21-3 读

```
            •
            •

            LD     R2,#3H              ; 把闪存的高位地址放进工作寄存器对的高地址寄存器中
            LD     R3,#0               ; 把闪存的低位地址放进工作寄存器对的低地址寄存器中
LOOP:       LDC    R0,@RR2             ; 从闪存区读数据 (从300H 到 3FFH)
            INC    R3
            CP     R3,#0H
            JP     NZ,LOOP

            •
            •
            •
            •
```

21.7 HARD LOCK 保护

用户可通过写“0110B”到 FMCON7-4 位开启 Hard Lock 保护。这个功能可以防止闪存区数据的变化。
如果启用这个功能，用户就不能往闪存里写或擦除数据。可通过在工具编程模式下，执行片擦除操作来解除这种保护。

在用户编程模式下，设置 Hard Lock 保护的流程如下。在工具编程模式下，
串行烧写工具的制造商通过编程选项支持硬件保护。具体请参照制造商提供的编程工具用户说明书。

21.7.1 用户编程模式下 HARD LOCK 保护的编程流程

1. 设置闪存用户编程使能寄存器 (FMUSR) 为“10100101B”。
2. 检查用户 ID 代码 (用户设置的任意值，和 FMUSR 功能类似，用来保护闪存区的安全，可选)。
3. 设置闪存控制寄存器 (FMCON) 为“01100001B”。
4. 设置闪存用户编程使能寄存器 (FMUSR) 为“00000000B”。

编程实例 21-4 Hard Lock 保护

```
                •
                •
                SB0
                LD      FMUSR,Temp0             ; 使能用户编程模式
                                                ; Temp0 =
                                                  #0A5H，必须在其他代码段中事先设置
                CP      UserID_Code,#User_value; 确认用户 ID
                                                  (用户自设定的任意值)
                JR      NE,Not_ID_Code          ; 如果不相等，则跳转到 Not_ID_Code
                LD      FMCON,Temp1             ; 设置 Hard Lock 保护模式并开始
                                                ; Temp1 =
                                                  #61H，必须在其他代码段中事先设置
                NOP                             ; 必须增加的空操作
                LD      FMUSR,#0                ; 禁止用户编程模式
                •
                •
                •
                •
Not_ID_Code:
                SB0
                LD      FMUSR,#0                ; 禁止用户编程模式
                LD      FMCON,#0                ; 关闭 Hard Lock 保护模式
                •
                •
                •
                •
```

注释: 在用户编程模式下，Temp0 ~ Temp1 的数据值必须在其他代码段中事先设置。
Temp0 ~ Temp(n) 变量必须由用户定义。

22 电气参数

22.1 概述

本章将以表格或者图表的方式提供 S3F84UA/F84U8 的电气参数。该信息按以下顺序排列：

- 芯片极限物理特性
- 输入/输出电容
- 直流电气特性
- 交流电气特性
- 振荡器特性
- 振荡器稳定时间
- STOP 模式下，数据保持电压
- LVR 时序特性
- 串行 I/O 时序特性
- A/D 转换电气特性
- UART 时序特性
- 内部闪存电气特性
- 工作电压范围

表 22-1 芯片极限物理特性

(T_A = 25°C)

参数	标号	条件	范围	单位
供电电压	V_{DD}	–	– 0.3 to + 6.5	V
输入电压	V_I	P0-P4	– 0.3 to V_{DD} + 0.3	
输出电压	V_O	–	– 0.3 to V_{DD} + 0.3	
输出电流高电平	I_{OH}	单个 I/O 口工作时	– 15	mA
		所有 I/O 口工作时	– 60	
输出电流低电平	I_{OL}	单个 I/O 口工作时	+ 30 (最大值)	
		所有的 I/O 口的电流和	+ 100 (最大值)	
工作温度	T_A	–	– 40 to + 85	°C
储藏温度	T_{STG}	–	– 65 to + 150	

表 22-2 直流电气特性

(T_A = – 40°C to + 85°C, V_{DD} = 2.0V to 5.5V)

参数	标号	条件	最小值	典型值	最大值	单位
工作电压	V_{DD}	fx = 0.4 – 4.2MHz, fxt = 32.768kHz	2.0	–	5.5	V
		fx = 0.4 – 12.0MHz	2.7	–	5.5	
输入高电压	V_{IH1}	除了 V_{IH2}，所有输入	0.7V_{DD}	–	V_{DD}	
	V_{IH2}	P3	0.8V_{DD}		V_{DD}	
	V_{IH3}	nRESET	0.8V_{DD}		V_{DD}	
	V_{IH4}	X_{IN}, X_{OUT}	V_{DD} – 0.1		V_{DD}	
输入低电压	V_{IL1}	除了 V_{IL2}，所有输入	–	–	0.3V_{DD}	
	V_{IL2}	P3			0.2V_{DD}	
	V_{IL3}	nRESET			0.2V_{DD}	
	V_{IL4}	X_{IN}, X_{OUT}			0.1	
输出高电压	V_{OH}	V_{DD} = 4.5V to 5.5V 所有输出管脚, I_{OH} = –1 mA	V_{DD}–1.5	–	–	V
输出低电压	V_{OL}	V_{DD} = 4.5V to 5.5V 所有输出管脚, I_{OL} = 10 mA	–	–	2.0	
输入高漏电流	I_{LIH1}	V_{IN} = V_{DD} 除了 I_{LIH2}，所有输入管脚	–	–	3	μA
	I_{LIH2}	V_{IN} = V_{DD} X_{IN}, X_{OUT}, XT_{IN}, XT_{OUT}			20	
输入低漏电流	I_{LIL1}	V_{IN} = 0 V 除了nRESET, I_{LIL2}，所有输入管脚	–	–	–3	
	I_{LIL2}	V_{IN} = 0 V X_{IN}, X_{OUT}, XT_{IN}, XT_{OUT}			–20	
输出高漏电流	I_{LOH}	V_{OUT} = V_{DD} 所有输出管脚	–	–	3	
输出低漏电流	I_{LOL}	V_{OUT} = 0V 所有输出管脚	–	–	–3	
LCD 分压电阻	R_{LCD}	T_A = 25°C	40	65	90	kΩ
时钟反馈电阻	R_{OSC1}	V_{DD} = 5V, T_A= 25°C X_{IN} = V_{DD}, X_{OUT} = 0 V	420	850	1700	
	R_{OSC2}	V_{DD} = 5V, T_A= 25°C XT_{IN} = V_{DD}, XT_{OUT} = 0 V	2200	4500	9000	
上拉电阻	R_{L1}	V_{IN} = 0V; V_{DD} = 5V Ports 0–4, T_A = 25°C	25	50	100	
		V_{IN} = 0V; V_{DD} = 3V Ports 0–4, T_A = 25°C	50	100	150	

参数	标号	条件		最小值	典型值	最大值	单位
	R_{L2}	V_{IN} = 0V; V_{DD} = 5V T_A = 25°C, nRESET		150	250	400	
		V_{IN} = 0V; V_{DD} = 3V T_A = 25°C, nRESET		300	500	700	
$\|V_{LCD} - COMi\|$ 压降 (i = 0 – 7)	V_{DC}	每个 Common 管脚 –15μA		–	–	120	mV
$\|V_{LCD} - SEGx\|$ 压降 (x = 0 – 21)	V_{DS}	每个 Segment 管脚 –15μA		–	–	120	
供电电流(1)	I_{DD1}(2)	RUN 模式：V_{DD} = 5.0V 晶体振荡器 C1 = C2 = 22pF	12.0MHz	–	3.0	6.0	mA
			4.2MHz		1.5	3.0	
		V_{DD} = 3.0V	4.2MHz		1.0	2.0	
	I_{DD2}(2)	IDLE 模式：V_{DD} = 5.0V 晶体振荡器 C1 = C2 = 22pF	12.0MHz	–	1.3	2.6	
			4.2MHz		0.8	1.6	
		V_{DD} = 3.0V	4.2MHz		0.4	0.8	
	I_{DD3}(3)	副时钟工作模式： V_{DD} = 3.0V, T_A = 25°C 32kHz 晶体振荡器		–	70.0	140.0	μA
	I_{DD4}(3)	副时钟 IDLE 模式： V_{DD} = 3.0V, T_A = 25°C 32kHz 晶体振荡器		–	8.0	20.0	
	I_{DD5}(4)	STOP 模式：T_A= 25°C, V_{DD} = 5.0V		–	2.5	6.0	
		T_A = 85°C, V_{DD} = 5.0V			5.0	10.0	
		T_A= – 40°C to + 85°C, V_{DD} = 5.0V			–	10.0	

注释:

1. 供电电流不包括内部上拉电阻、LCD 分压电阻、LVR 模块和外部输出电流负载消耗的电流。
2. I_{DD1} 和 I_{DD2} 包括副时钟振荡器的功耗。
3. I_{DD3} 和 I_{DD4} 为主时钟振荡器停止并且使用副时钟时的电流。
4. I_{DD5} 是主时钟和副时钟振荡器都停止时的电流。
5. 表中的每个值均为系统时钟控制寄存器 (CLKCON.4–.3) 的 4-3 位设置为“11B”时测量而得。

表 22-3 交流电气特性

(T_A = – 40°C to + 85°C, V_{DD} = 2.0V to 5.5V)

参数	标号	条件	最小值	典型值	最大值	单位
中断输入高/低电平脉冲宽度 (P3.0 – P3.7)	t_{INTH}, t_{INTL}	所有中断, V_{DD} = 5V	500	–	–	ns
nRESET 输入低电平脉冲宽度	t_{RSL}	输入, V_{DD} = 5V	10	–	–	µs

注释: 如果中断或复位脉冲的宽度大于最小值，则总会被视为有效脉冲。

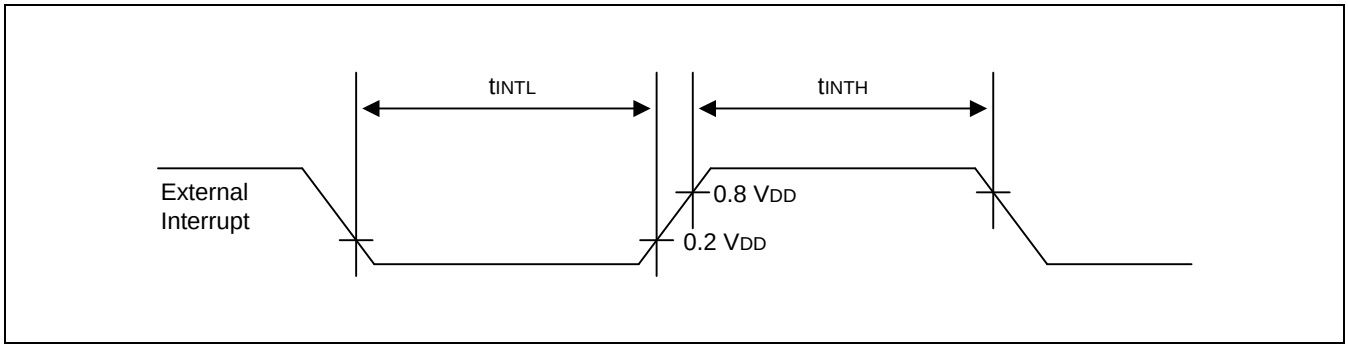


图 22-1 外部中断输入时序

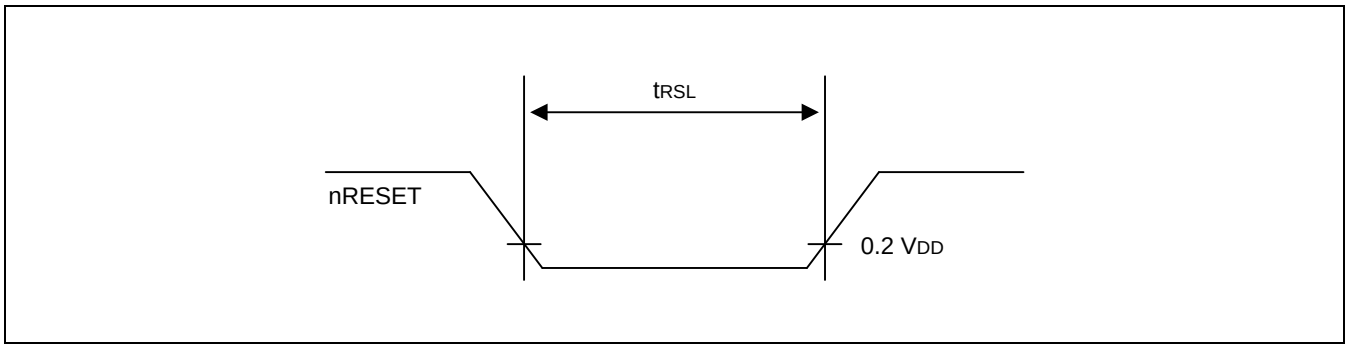


图 22-2 nRESET 输入时序

表 22-4 输入/输出电容

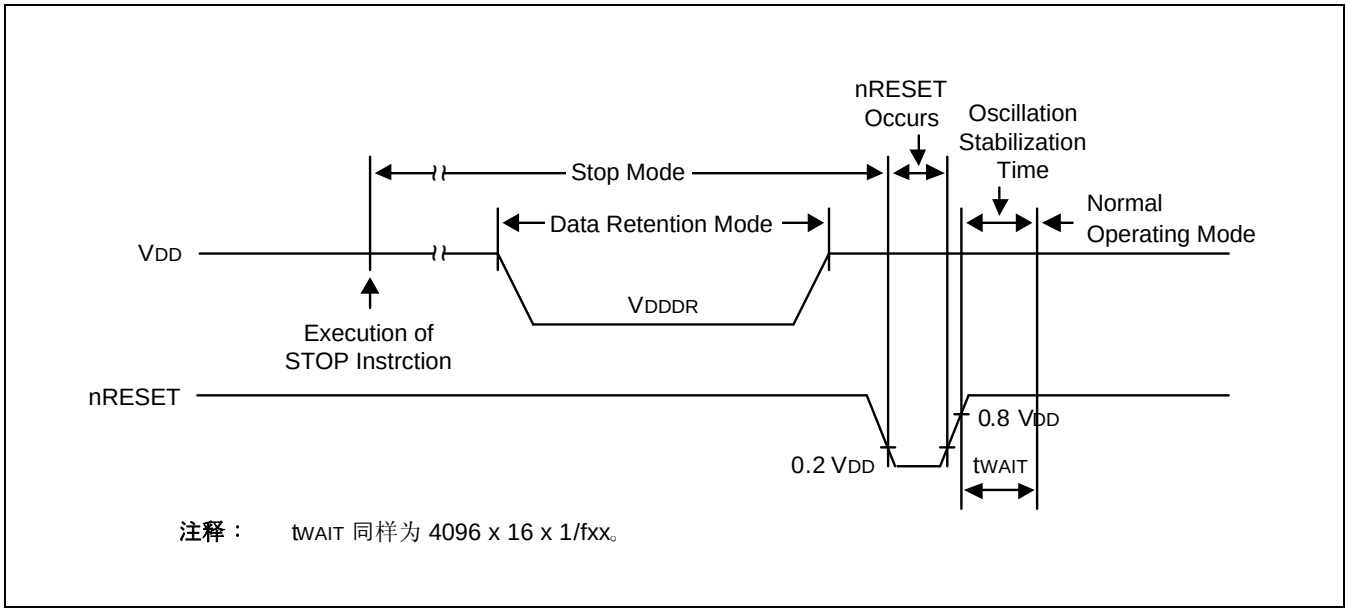
(T_A = – 40°C to + 85°C, V_{DD} = 0V)

参数	标号	条件	最小值	典型值	最大值	单位
输入电容	C_{IN}	f = 1 MHz; 为测量管脚均接到 V_{SS}	–	–	10	pF
输出电容	C_{OUT}					
I/O 电容	C_{IO}					

表 22-5 STOP 模式下 数据保持电压

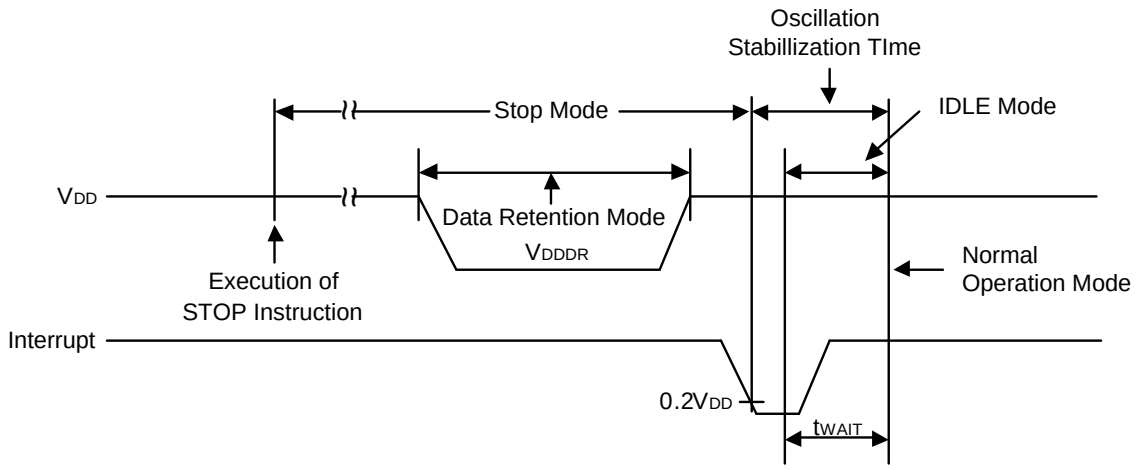
(T_A = – 40°C to + 85°C)

参数	标号	条件	最小值	典型值	最大值	单位
保持数据所需的供电电压	V_{DDDR}		2.0	–	5.5	V
保持数据所需的供电电流	I_{DDDR}	STOP 模式, TA = 25°C V_{DDDR} = 2.0V	–	–	1	µA



注释： tWAIT 同样为 4096 x 16 x 1/fxx。

图 22-3 nRESET 使系统退出 STOP 模式时序图



注释： tWAIT 同样为 16 x 1/fBT。(fBT 是 Basic Timer 所选的时钟)

图 22-4 中断使系统退出 STOP 模式时序图

表 22-6 A/D 转换电气特性

(TA = – 40°C to + 85°C, V_{DD} = 2.7V to 5.5V)

参数	标号	条件	最小值	典型值	最大值	单位
分辨率		–	–	10	–	bit
总精度		–	–	–	± 3	LSB
积分线性误差	ILE	V_{DD} = 5.120V V_{SS} = 0V CPU 时钟 = 12.0MHz	–	–	± 2	
微分线性误差	DLE			–	± 1	
最高点偏移误差	EOT			± 1	± 3	
最低点偏移误差	EOB			± 1	± 3	
转换时间(1)	T_{CON}	–	25	–	–	μS
模拟信号输入电压	V_{IAN}	–	V_{SS}	–	AVREF	V
模拟信号输入阻抗	R_{AN}	–	2	1000	–	M
模拟参考电压	AV_{REF}	–	2.0	–	VDD	V
模拟信号输入电流	I_{ADIN}	V_{DD} = 5.0V	–	–	10	μA
模拟模块电流(2)	I_{ADC}	V_{DD} = 5.0V	–	0.5	1.5	mA
		V_{DD} = 5.0V 省电模式下		100	500	nA

注释:

1. “转换时间”是从启动转换到结束转换所需要的时间。
2. I_{ADC} 是 A/D 转换器的工作电流。

表 22-7 LVR (低电压复位) 电气特性

(TA = – 40°C to + 85°C, V_{DD} = 2.0V to 5.5V)

Parameter	Symbol	Test Condition	Min	Typ	Max	Unit
LVR 电压	V_{LVR}	T_A = 25°C, V_{DD} = 2.2V (下降)	2.0	2.2	2.4	V
		T_A = 25°C, V_{DD} = 2.4V (上升)	2.2	2.4	2.6	
V_{DD} 电压上升时间	t_R	–	10	–	–	μS
V_{DD} 电压关闭时间	t_{OFF}	–	0.5	–	–	S
电流功耗	I_{LVR}	V_{DD} = 3.3V	–	60	120	μA
		V_{DD} = 5.5V		80	160	

注释: LVR 电路的电流在通过“Smart Option”使能 LVR 时测量而得。

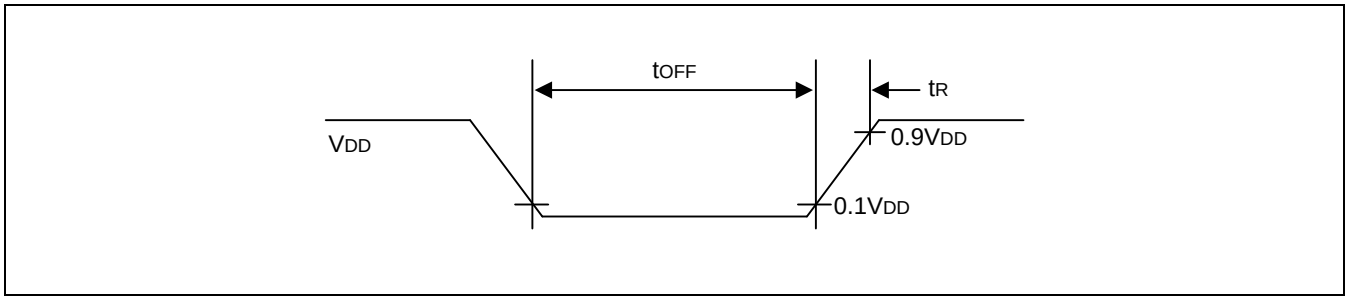


图 22-5 LVR (低电压复位) 时序

表 22-8 同步 SIO 电气特性

(T_A = – 40°C to + 85°C, V_{DD} = 2.0V to 5.5V)

参数	标号	条件	最小值	典型值	最大值	单位
SCK 周期	t_{KCY}	外部 SCK 源	1,000	–	–	ns
		内部 SCK 源	1,000			
SCK 高/低宽度	t_{KH}, t_{KL}	外部 SCK 源	500			
		内部 SCK 源	tKCY/2-50			
SI 至 SCK 高电平的建立时间	t_{SIK}	外部 SCK 源	250			
		内部 SCK 源	250			
SI 至 SCK 高电平的保持时间	t_{KSI}	外部 SCK 源	400			
		内部 SCK 源	400			
SCK 至 SO 的输出延时	t_{KSO}	外部 SCK 源	–		300	
		内部 SCK 源			250	

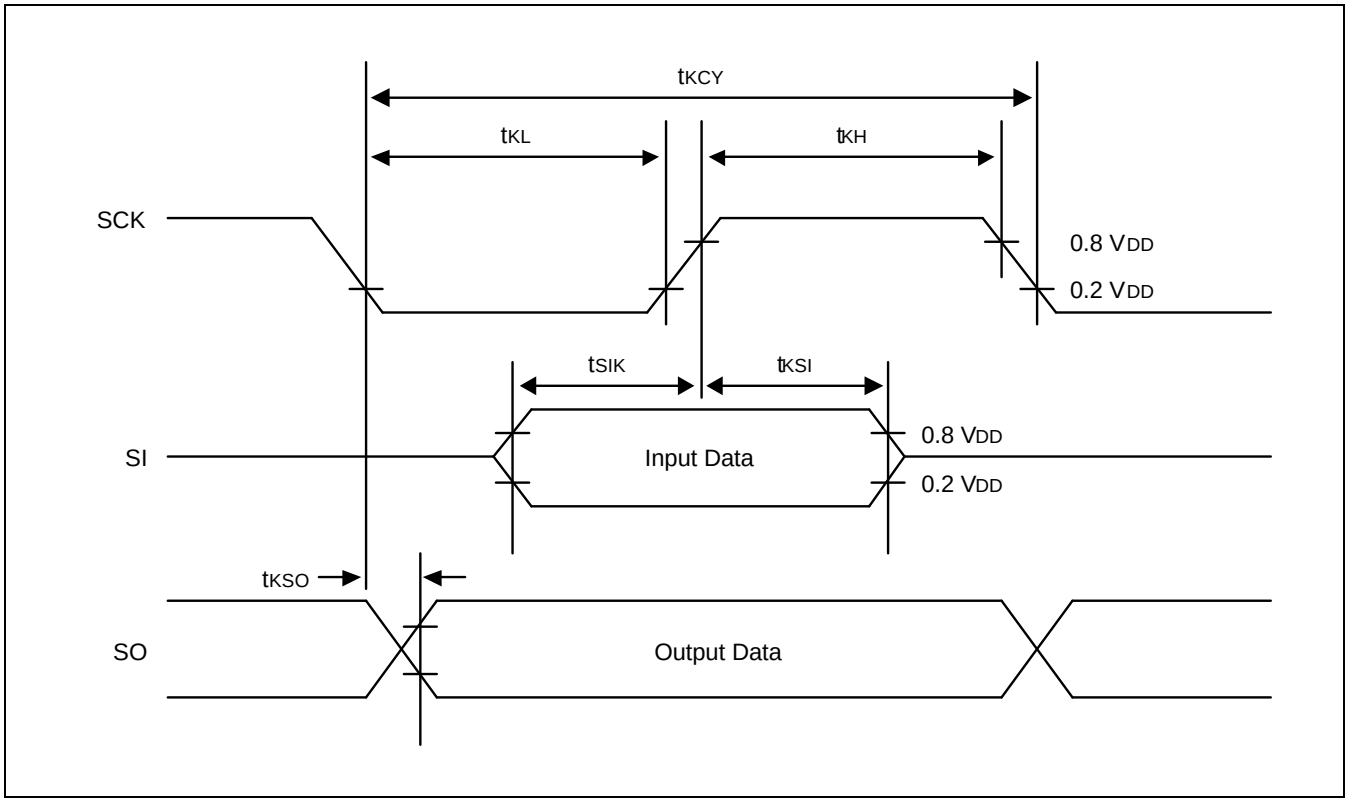


图 22-6 串行数据发送时序

表 22-9 UART 在模式 0 下的时序特性 (12.0MHz)

(T_A = – 40°C to + 85°C, V_{DD} = 2.0V to 5.5V, 负载电容值 = 80pF)

参数	标号	最小值	典型值	最大值	单位
串口时钟周期时间	t_{SCK}	1,160	$T_{cpu} \times 16$	1,500	ns
输出数据建立至时钟上升沿的时间	t_{S1}	500	$t_{CPU} \times 13$	–	
时钟上升沿至输入数据有效的时间	t_{S2}	–	–	500	
时钟上升沿后输出数据保持时间	t_{H1}	$t_{CPU} – 50$	t_{CPU}	–	
时钟上升沿后输入数据保持时间	t_{H2}	0	–	–	
串口时钟高/低电平宽度	t_{HIGH}, t_{LOW}	450	$t_{CPU} \times 8$	890	

注释:

1. 所有时序均以纳秒 (ns) 为单位，并假设工作在 12.0-MHz CPU 时钟频率。
2. 单位 t_{CPU} 为一个 时钟周期。

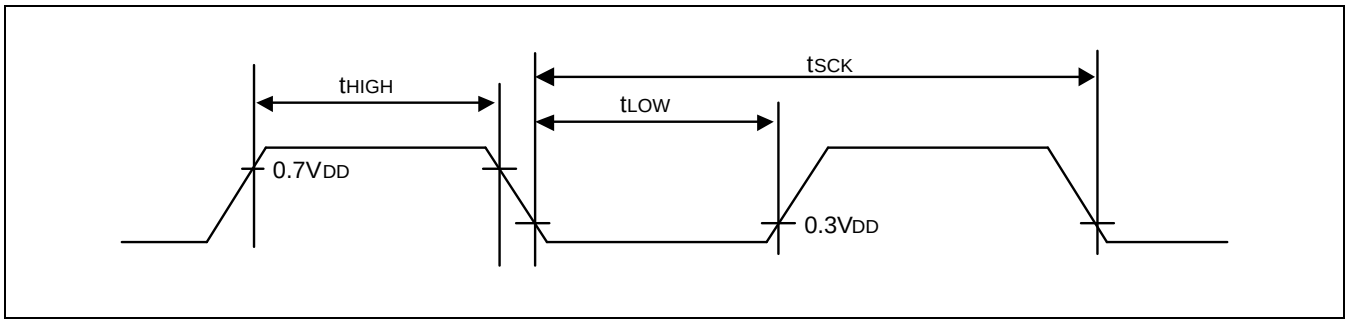
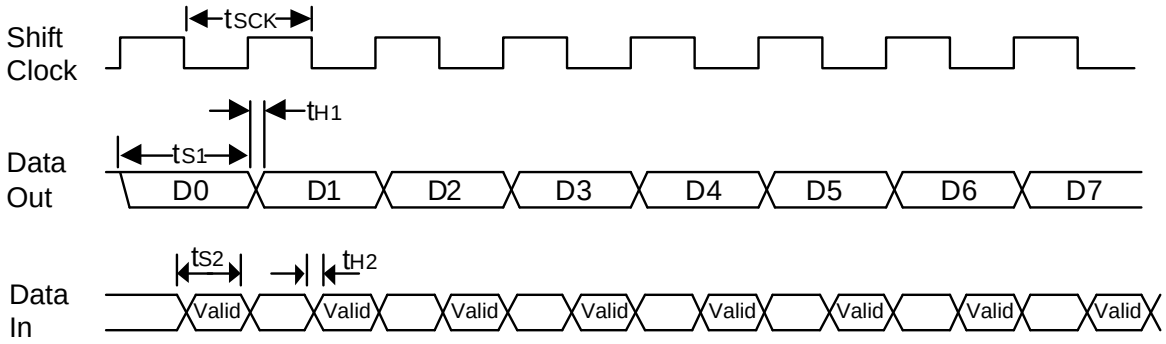


图 22-7 UART 时序特性波形图



注释： 图中的符号如下定义为：

f_{SCK}	串行口时钟周期时间
t_{S1}	输出数据建立至时钟上升沿的时间
t_{S2}	时钟上升沿至输入数据有效的时间
t_{H1}	时钟上升沿后输出数据保持时间
t_{H2}	时钟上升沿后输入数据保持时间

图 22-8 UART 模块的时序波形图

表 22-10 主振荡器特性

(T_A = – 40°C to + 85°C, V_{DD} = 2.0V to 5.5V)

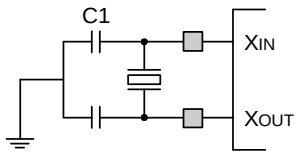
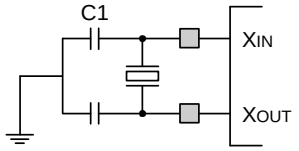
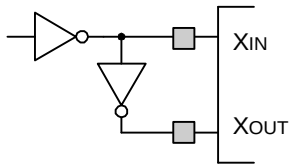
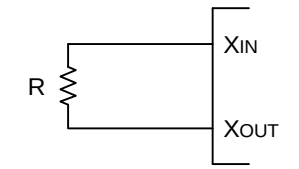
振荡器	时钟结构	参数	测试条件	最小值	典型值	最大值	单位
石英振荡器		主振荡器频率	2.7V – 5.5V	0.4	–	12.0	MHz
			2.0V – 5.5V	0.4	–	4.2	
陶瓷振荡器		主振荡器频率	2.7V – 5.5V	0.4	–	12.0	
			2.0V – 5.5V	0.4	–	4.2	
外部时钟		X_{IN} 输入频率	2.7V – 5.5V	0.4	–	12.0	
			2.0V – 5.5V	0.4	–	4.2	
RC 振荡器		频率	3.3V	0.4	–	1.0	MHz

表 22-11 副振荡器特性

(T_A = –25°C to + 85°C)

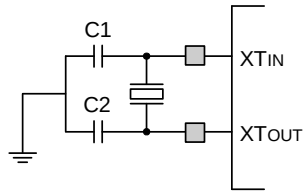
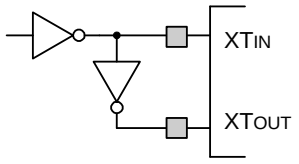
振荡器	时钟结构	参数	测试条件	最小值	典型值	最大值	单位
石英振荡器		副振荡器频率	2.0V – 5.5V	–	32.768	–	kHz
外部时钟		XT_{IN} 输入频率	2.0V – 5.5V	32	–	100	

表 22-12 主振荡器稳定时间

(T_A = – 40°C to + 85°C, V_{DD} = 2.0V to 5.5V)

振荡器	测试条件	最小值	典型值	最大值	单位
石英振荡器	fx > 1 MHz 当 V_{DD} 满足最小的起振电压时，振荡器开始稳定。	–	–	40	ms
陶瓷振荡器		–	–	10	ms
外部时钟	X_{IN} 输入的高/低电平宽度 (t_{XH}, t_{XL})	62.5	–	1250	ns

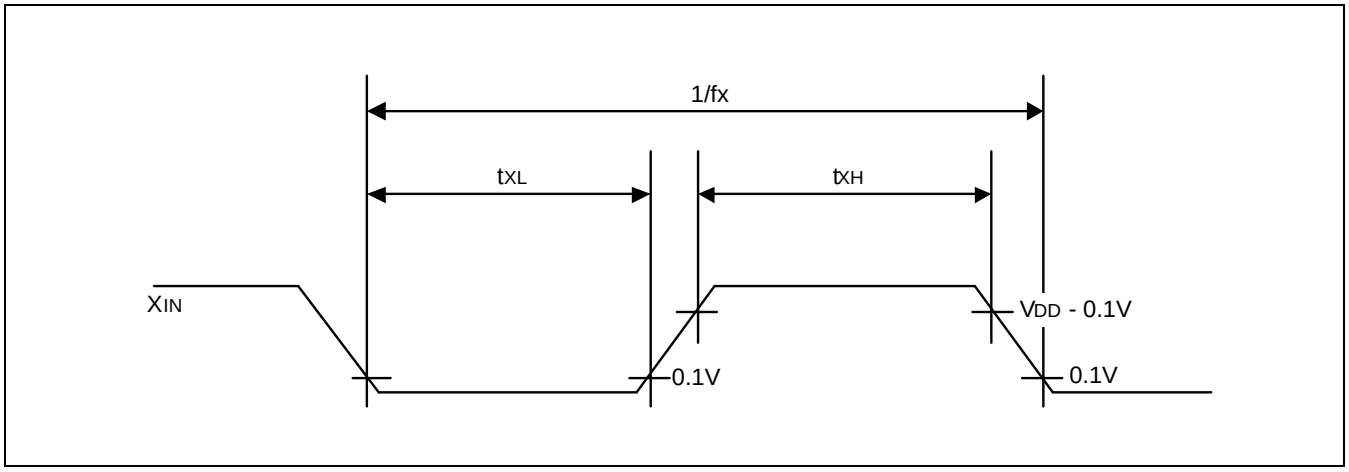


图 22-9 X_{IN} 上测量到的时钟时序

表 22-13 副振荡器稳定时间

(T_A = – 40°C to + 85°C, V_{DD} = 2.0V to 5.5V)

振荡器	测试条件	最小值	典型值	最大值	单位
石英振荡器	–	–	–	10	s
外部时钟	XT_{IN} 输入的高/低电平宽度 (t_{XTH}, t_{XTL})	5	–	15	µs

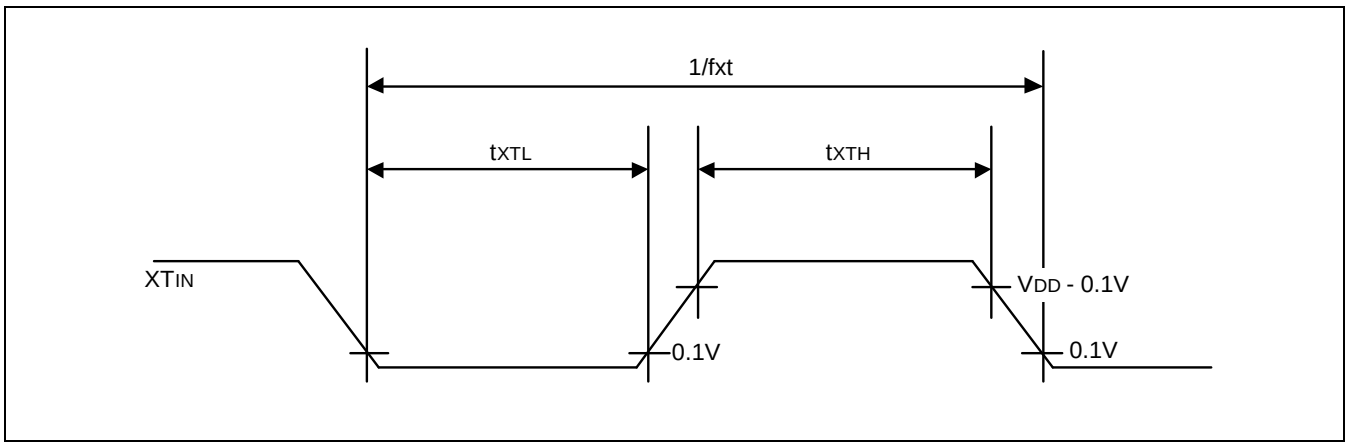


图 22-10 XT_{IN} 上测量到的时钟时序

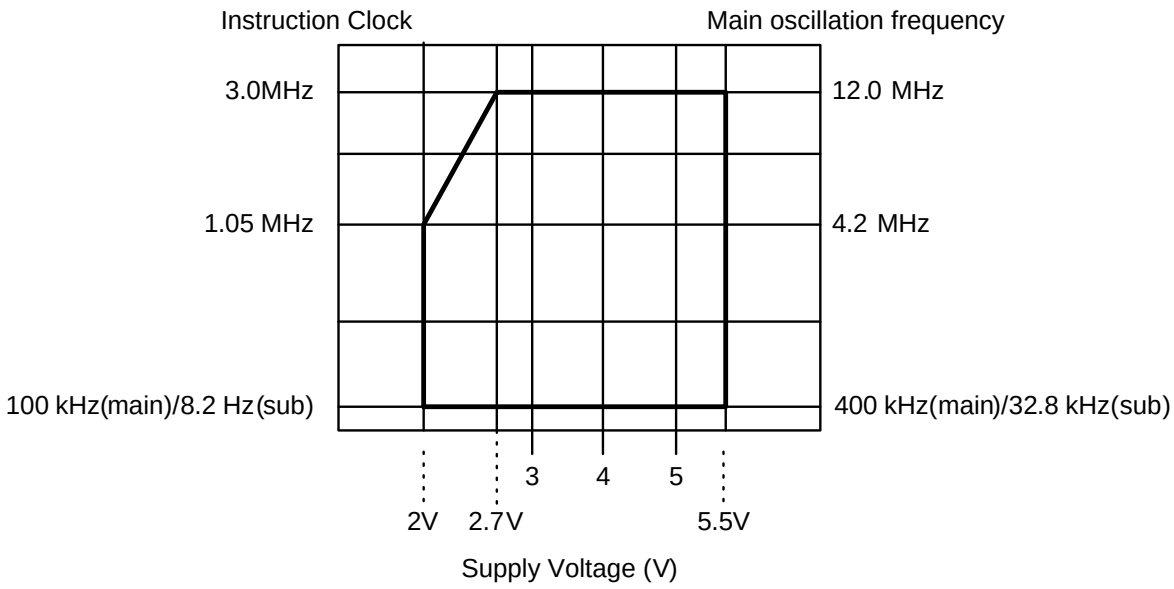


图 22-11 工作电压范围

表 22-14 内部闪存电气特性

(T_A = – 40°C to + 85°C, V_{DD} = 2.0V to 5.5V)

参数	标号	条件	最小值	典型值	最大值	单位
编程时间(1)	Ftp	–	20	30	40	μs
芯片擦除时间(2)	Ftp1		30	50	70	ms
扇区擦除时间(3)	Ftp2		10	15	20	ms
读取数据频率	f_R		–	–	12	MHz
写/擦除次数	FN_{WE}	–	–	–	10,000(4)	次数

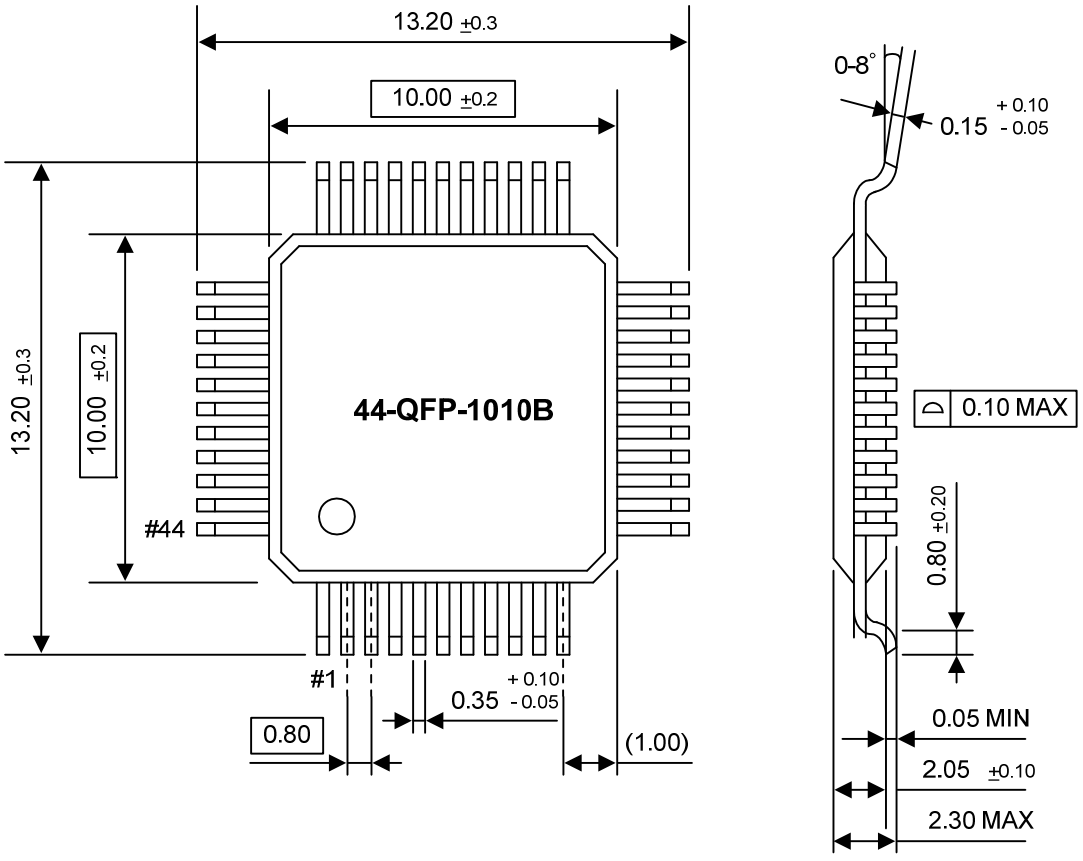
注释:

1. 编程时间是编程一个字节 (8位) 所需的时间。
2. 芯片擦除时间是擦除所有 64K 字节所需的时间。
3. 扇区擦除时间是擦除一个扇区内所有 128 字节所需的时间。
4. 芯片擦除仅在工具编程模式下可用。

23 机械尺寸

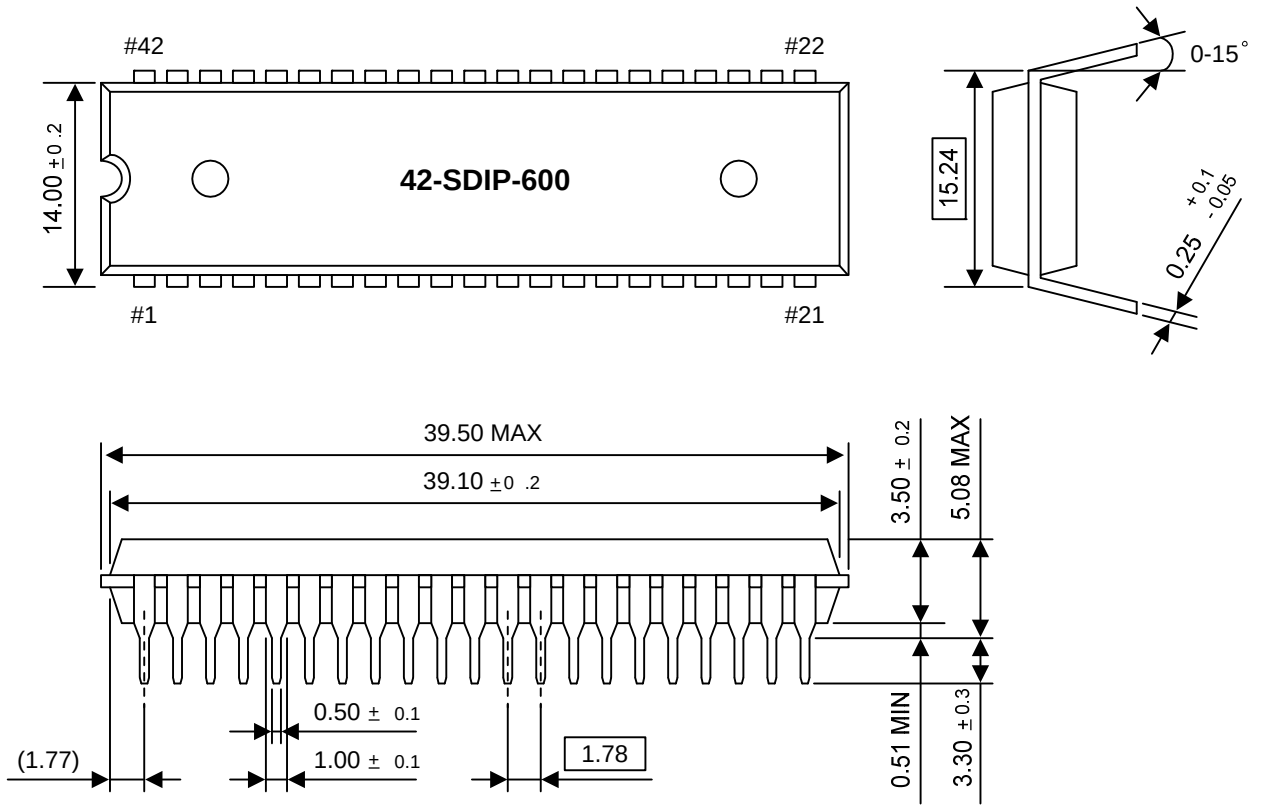
23.1 概述

S3F84UA/F84U8 MCU 提供 44-管脚-QFP 和 42-管脚-SDIP 封装。



注释： 标注单位为毫米。

图 23-1 封装尺寸 (44-QFP-1010B)



注释： 标注单位为毫米。

图 23-2 封装尺寸 (42-SDIP-600)

24 S3F84UA/F84U8 FLASH MCU

24.1 概述

S3F84UA/F84U8 单芯片 CMOS MCU 是一款 Flash MCU。它具有片上 48K/8K 字节闪存空间，可通过串行数据格式进行访问。

注释: 本章介绍 Flash MCU 的工具编程模式。如需了解用户编程模式，请参考第 21 章，嵌入式闪存接口。

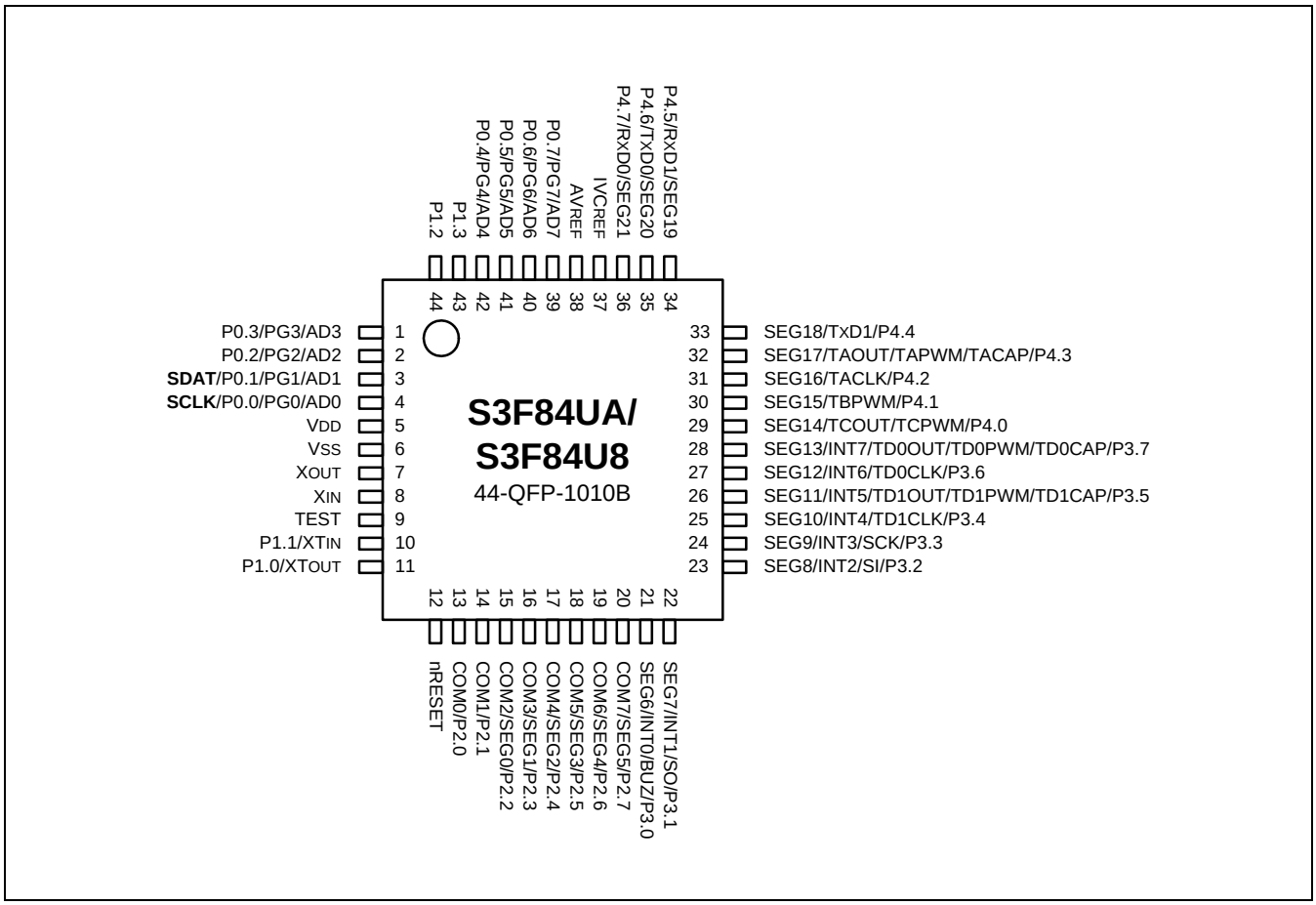


图 24-1 S3F84UA/F84U8 管脚分布 (44-QFP-1010B)

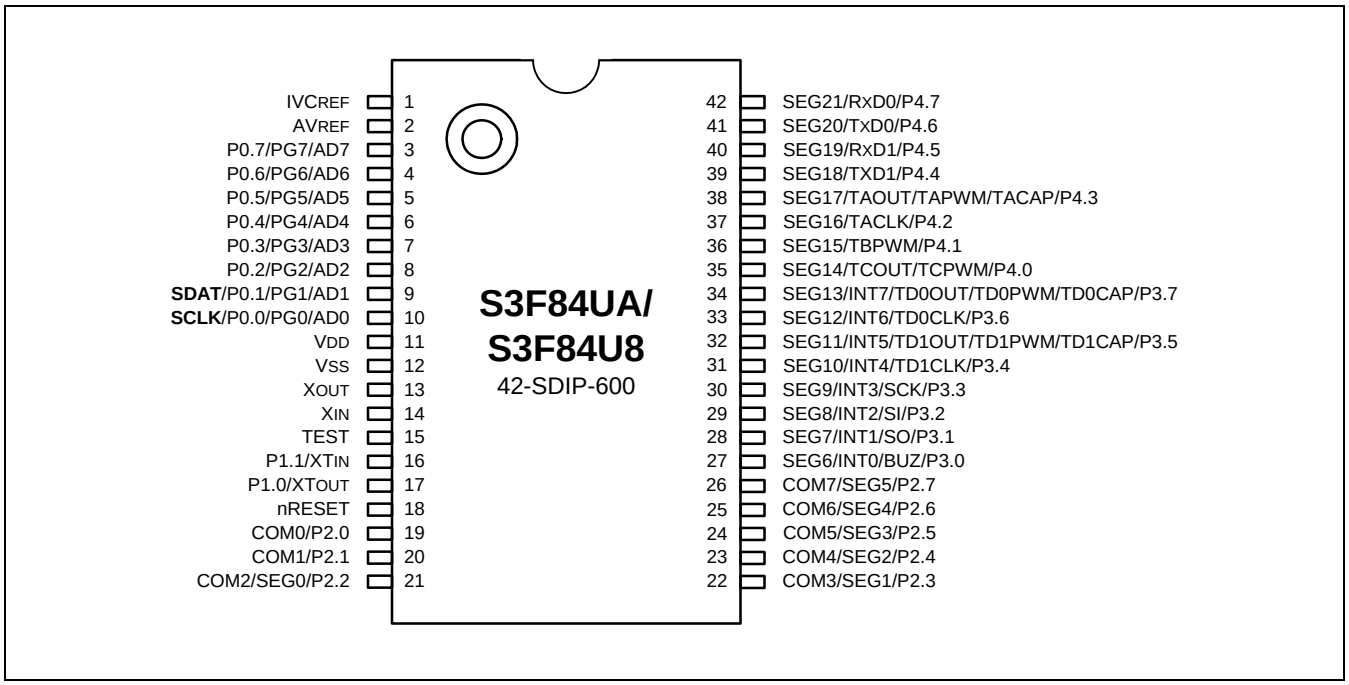


图 24-2 S3F84UA/F84U8 管脚分布 (42-SDIP-600)

表 24-1 Flash ROM 读/写管脚特性描述

主芯片	编程过程中			
管脚名称	管脚名称	管脚序号	I/O	功能
P0.1	SDAT	3(9)	I/O	串行数据管脚(读出时为输出脚，写入时为输入脚)，管脚可设置为输入和推挽式输出口。
P0.0	SCLK	4(10)	I/O	串行时钟管脚(仅为输入管脚)。
TEST	V_{PP}	9(15)	I	工具模式选择位。当 TEST/ V_{PP} 管脚检测到逻辑高即进入工具模式。用户如果使用 Flash 编程工具(如 spw2+ 等)时，应当将 TEST/ V_{PP} 管脚应该连到 V_{DD}。(S3F84UA/F84U8 内部有自升压电路 可以产生 12.5V 的编程电压)
nRESET	nRESET	12(18)	I	芯片初始化。
IVC_{REF}	IVC_{REF}	37(1)	–	必须在 IVC_{REF} 和 V_{SS} 之间接一个 0.1uF 电容。
V_{DD}, V_{SS}	V_{DD}, V_{SS}	5(11) 6(12)	–	逻辑电路的供电管脚 (编程时 V_{DD} 应接到 5V 电源)。

注释:

1. 括号内是 42-SDIP-600 封装的管脚序号。
2. V_{PP} (TEST) 管脚最好连到 V_{DD}。

Test 管脚电压

仅当使用 SPW 2+ 和 GW-pro2 等 MTP 烧写器时，转接板上的 TEST 管脚必须采用如图 *图 24-3* 所示的 RC 延迟电路连到 V_{DD} (5.0V)。

任何情况下，TEST 管脚都不能直接连到 MTP 烧写器提供的 Vpp (12.5V) 上。 当使用 MTP 烧写器进行烧写或擦除芯片时，必须使用专门的 S3F84UA/F84U8 转接板。

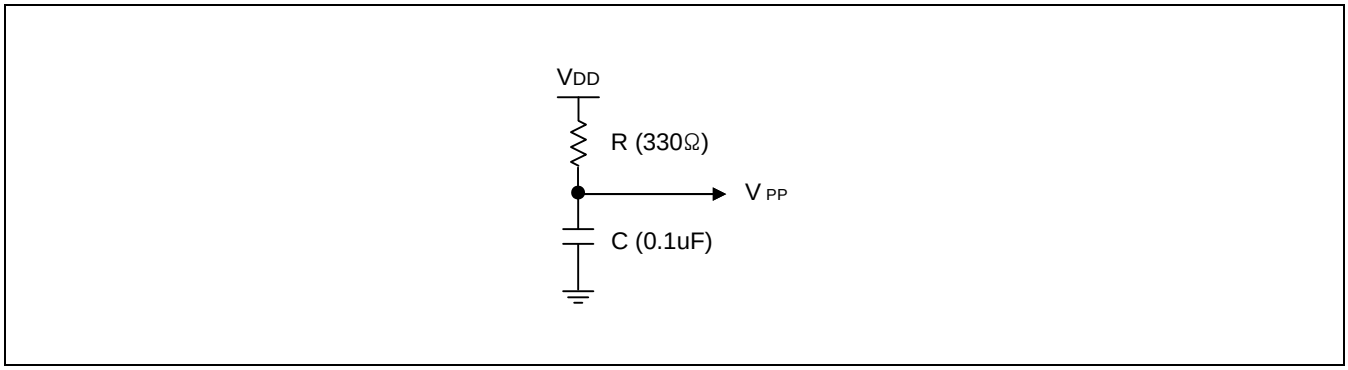


图 24-3 RC 时延电路

24.2 在板编程

S3F84UA/F84U8 仅需要包含 V_{DD} 和 V_{SS} 在内的 6 根信号线，用来通过串口协议对内部闪存进行烧写。因此，在应用板的 PCB 设计之初，应考虑烧写信号线，这样就能实现在板编程。

24.2.1 电路设计指导

在 Flash 烧写时，烧写工具需要的 6 条信号线为：V_{SS}，V_{DD}，nRESET，TEST，SDAT 和 SCLK。用户在进行 PCB 电路设计时，应该考虑在板烧写时这些信号线的用途。对于 TEST 管脚，正常使用时应该连接到 V_{SS}，但是在编程过程中必须保持高电平，所以需要在 TEST 和 V_{SS} 之间连一个电阻。nRESET，SDAT 和 SCLK 也需要做同样的考量。

请仔细设计和这些信号管脚相关的电路，因为 V_{PP}，SCLK 和 SDAT 的上升/下降时序可能直接决定编程是否成功。

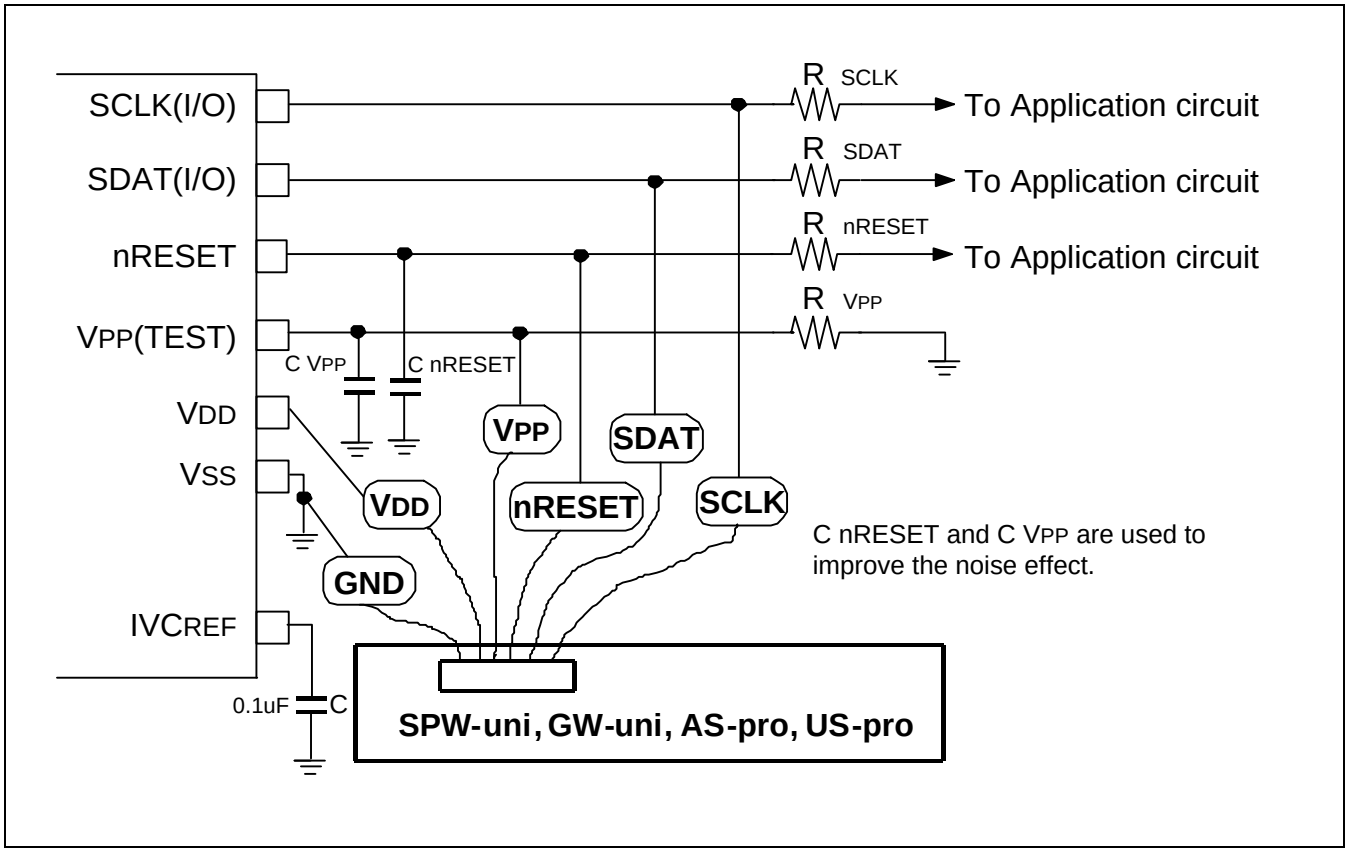


图 24-4 编程接口(在板编程) PCB 设计指导

24.2.2 连接参照表

表 24-2 连接参照表

管脚名称	应用时 I/O 模式	电阻(需要)	推荐值
VPP (TEST)	输入	需要	R_{Vpp} = 10kΩ ~ 50kΩ。 C_{Vpp} = 0.01uF ~ 0.02uF。
nRESET	输入	需要	R_{nRESET} = 2kΩ ~ 5kΩ。 C_{nRESET} = .01uF ~ 0.02uF。
SDAT(I/O)	输入	需要	R_{SDAT} = 2kΩ ~ 5kΩ。
	输出	不需要(注释)	–
SCLK(I/O)	输入	需要	R_{SCLK} = 2kΩ ~ 5kΩ。
	输出	不需要(注释)	–

注释:

1. 下，管脚 SCLK 和 SDAT 上为非常快速信号。如果应用电路设计为告诉响应时，例如延迟控制电路，那么如果将对连到 SCLK 和 SDAT 口上的应用电路造成一定的损坏。如果可能，最好将 SDAT 和 SCLK 管脚设置为输入模式。
2. 本表中的 R 和 C 的值仅为推荐值，在不同的电路系统中值也不同。