S3F82HB

8-BIT COMOS

Revision 1.20

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.

S3F82HB 8-BIT COMOS
用户手册, Revision 1.20

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	Sep, 2006	Preliminary Spec for internal release only.	bumsun.Hong
1.00	Jan, 2007	First edition.	bumsun.Hong
1.10	Mar, 2007	Second edition	bumsun.Hong
1.20	Oct, 2008	Third edition	bumsun.Hong

Table of Contents

List of Figures

List of Tables

List of Examples

1 产品概述

1.1 S3F8- 系列 MCU

三星S3F8-系列单片机向用户提供了高效快速的 CPU，丰富的外设接口，以及各种大小的可编程 ROM。
其中重要的 CPU 特性包括：

- 高效的寄存器结构
- 可选的 CPU 时钟源
- 可由中断唤醒的 IDLE 和 STOP 低功耗模式
- 内置有看门狗功能的Basic Timer

它的中断结构复杂，可提供高达8个中断优先级。每个优先级又有1个或多个中断源和中断向量。特定的中断优先级还可设置成快速中断 (最短4个CPU时钟周期)。

1.2 S3F82HB MCU

S3F82HB 单片机采用了三星最新的 CPU 构架和先进的 CMOS 工艺。

S3F82HB 是一款内置64K 字节 Flash ROM 的微控制器。

三星工程师采用先进的模块化设计，成功的将如下外设模块集成到强大的 SAM8 核中：

- 11个可编程 I/O口，包括9个8位 I/O口，1个6位I/O口和1个5位I/O 口，共83个管脚
- 12个供外部中断使用的位可编程管脚
- 1个可用于振荡稳定和看门狗 (系统复位) 的8位 basic timer
- 2个8位timer/counter 和2个可选工作模式的16位 timer/counter
- 可用作实时时钟的钟表定时器 (Watch timer)
- LCD 控制器/驱动器
- 9通道模数转换通道
- 同步 SIO 模块
- 16位频率计数器
- 2个模拟比较器
- 电池电压检测

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)
  - 64K × 8 位程序存储器 (S3F82HB)
  - 内部 flash 存储器 (程序存储器)
    - 扇区 (Sector)：128 字节
    - 10年数据保留
    - 快速编程
    - 用户可编程及扇区擦除支持
    - 寿命：10,000 次可擦除/编程
    - 支持外部串行编程
    - 可扩展的 OBPTM (on board program) 扇区
- 据存储器 (RAM)
  - 包括 LCD 显示数据存储器
  - 2634 × 8 位数据存储器

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 83 I/O 口

- I/O：23 个
- I/O：60 个 (LCD 信号复用)

1.3.5 中断

- 8 个中断级，29 个中断源
- 快速中断处理支持

1.3.6 8 位 BASIC TIMER

- 看门狗功能
- 4种时钟源可选

1.3.7 8 位 TIMER/COUNTER A

- 可编程8位内部 timer
- 外部事件计数功能
- PWM 和捕获 (capture) 功能

1.3.8 8 位 TIMER/COUNTER B

- 可编程8位内部 timer
- 载波发生器

1.3.9 2 个 16 位 TIMER/COUNTERS (T0，T1)

- 可编程16位内部 timer
- 外部事件计数功能
- PWM 和捕获 (capture) 功能

1.3.10 钟表定时器 (WATCH TIMER)

- 时间间隔：3.91mS，0.25S，0.5S 或1S (32.768kHz)
- 0.5/1/2/4kHz 可选的蜂鸣器输出

1.3.11 16 位 FREQUENCY COUNTER

- 外部事件计数功能
- 门控制功能

1.3.12 LCD 控制器/驱动器

- 52 个段，8 个公共端
- 1/2，1/3，1/4和1/8占空比可选
- 电容或电阻偏置可选
- 内置 LCD 偏置需要的稳压器和升压电路

1.3.13 A/D 转换器

- 8通道模拟输入
- 10位转换精度
- 25uS 转换时间

1.3.14 2 个模拟比较器

- 2路
- 每路都有正负输入端和输出端

1.3.15 2 路 UART

- 全双工串行 I/O 接口
- 4个可编程工作模式
- 自动生成奇偶校验位

1.3.16 8 位 SIO 接口

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

1.3.17 电池电压检测

- 3种阈值电压可选 (2.2V，2.4V，2.8V)
- 根据功耗要求，软件使能或禁止

1.3.18 低电压复位 (LVR)

- 阈值电压：2.2V
- 通过smart option (ROM 地址：3FH) 使能/禁止

1.3.19 2 种低功耗模式

- Idle：仅停止 CPU 时钟
- Stop：停止被选中的系统时钟和 CPU 时钟

1.3.20 振荡源

- 主时钟可选石英晶振，陶瓷振荡器或者 RC
- 主时钟频率：0.4MHz – 12.0MHz
- 副时钟选用32.768kHz 晶体振荡电路

1.3.21 指令执行时间

- fx = 12.0MHz 时，最快333nS
- fxt = 32.768kHz 时，最快122.1uS

1.3.22 工作电压范围

- 2.0V - 3.6V (0.4 – 4.2MHz)
- 2.7V - 3.6V (0.4 – 12.0MHz)

1.3.23 低功耗模式

- sub-idle 模式下，VDD = 3.0V 时电流为 0.9uA

1.3.24 工作温度范围

- –25°C 到 + 85°C

1.3.25 封装形式

- 100-QFP-1420C，100-TQFP-1414

1.3.26 SMART OPTION

- 低电压复位 (LVR) 的使能/禁止及复位电压可由硬件选择 (ROM 地址3FH)
- ISP 相关设置选择 (ROM 地址3EH)

1.4 内部模块框图

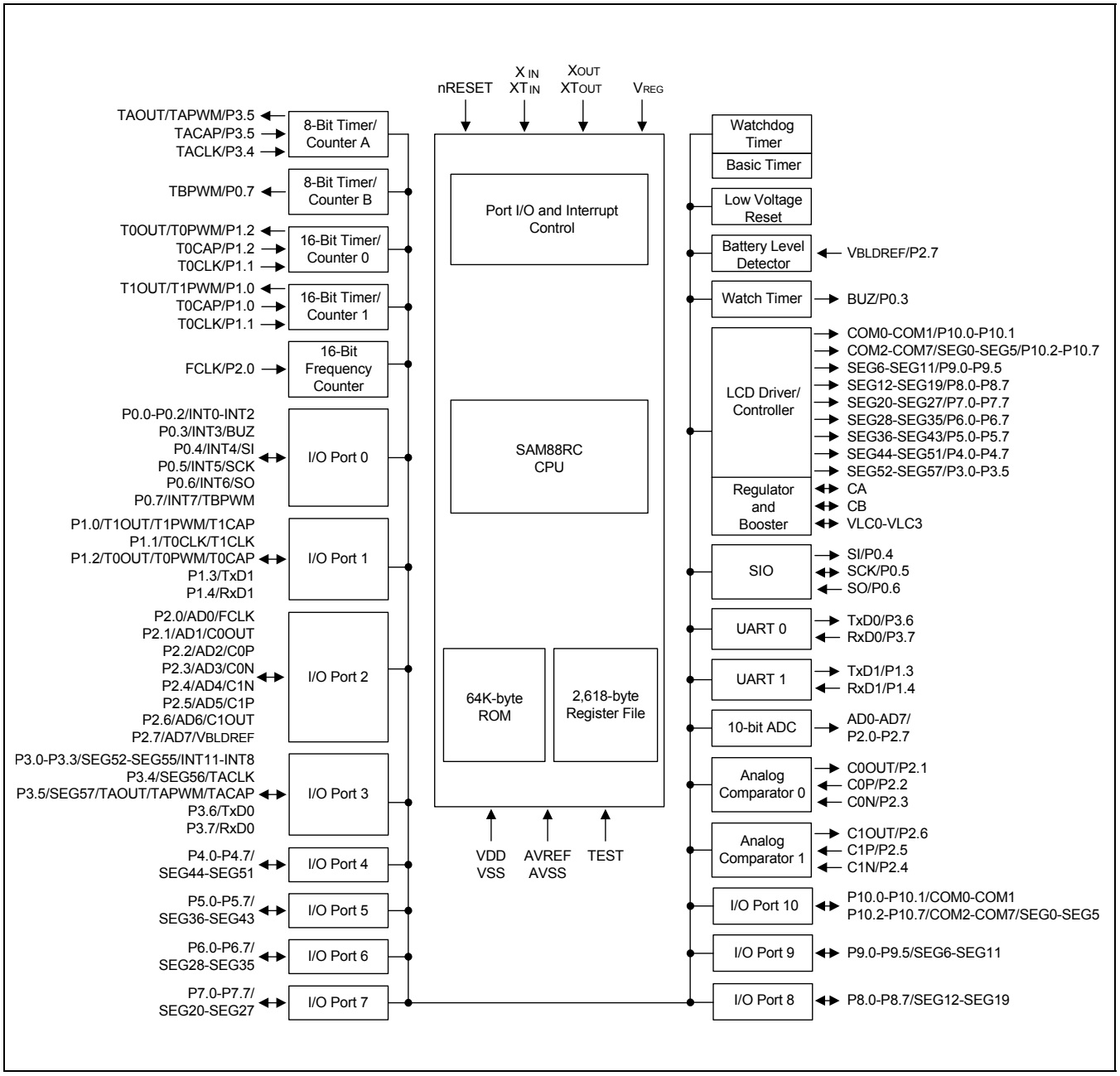


图 1-1 系统框图

1.5 管脚分布图

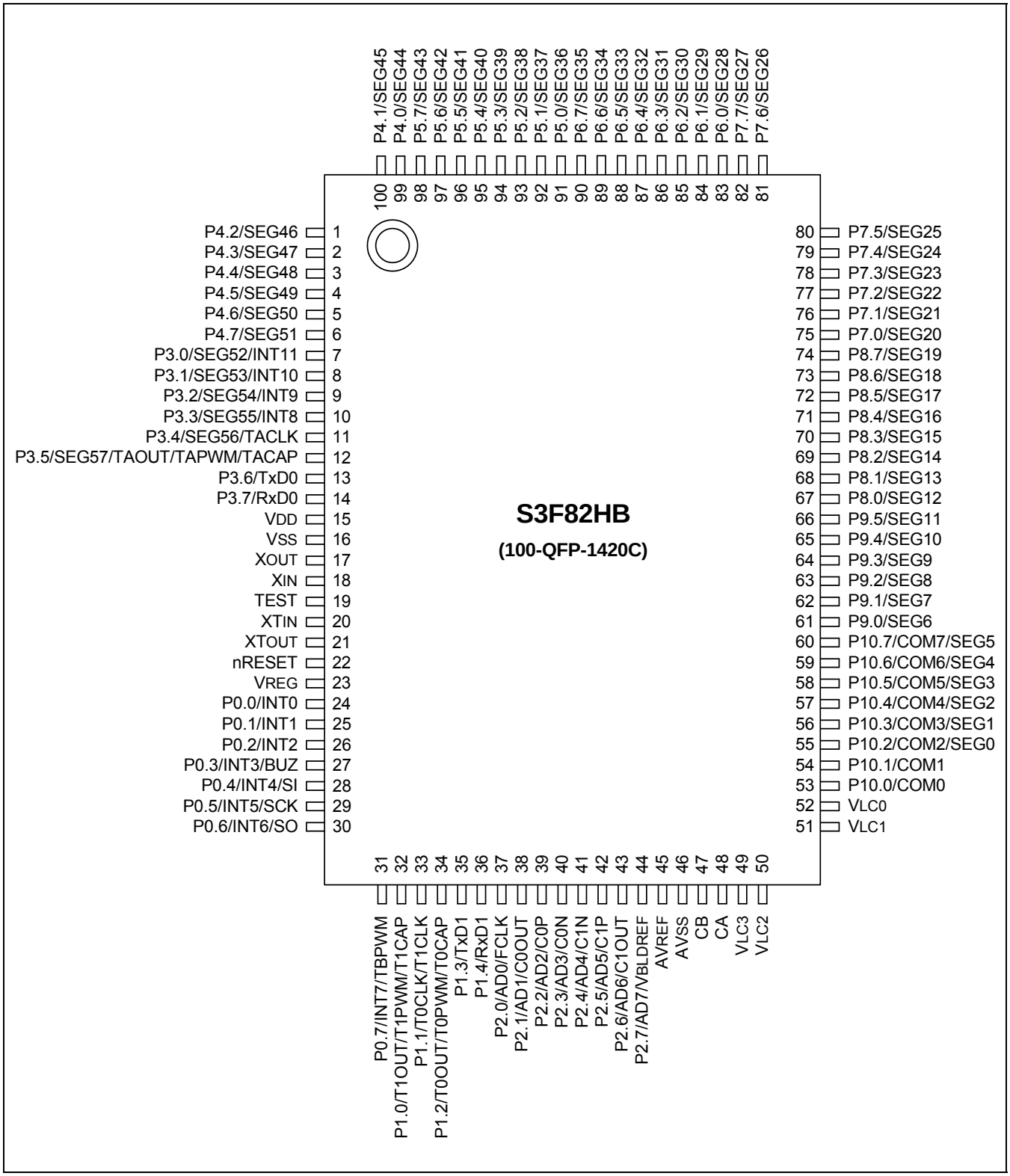


图 1-2 S3F82HB 管脚分布图 (100-QFP-1420C 封装)

NOTE: The sequence of pins in TQFP package is disagreement with that in QFP package.

图 1-3 S3F82HB 管脚分布图 (100-TQFP-1414 封装)

1.6 管脚特性描述

表 1-1 S3F82HB 管脚特性

管脚名称	输入/输出	管脚特性描述	管脚类型	管脚号 (注释)	复用管脚
P0.0-P0.2 P0.3 P0.4 P0.5 P0.6 P0.7	I/O	可对该口的每一位进行功能设定，可以设定为施密特触发器输入，推挽式输出或开漏输出。通过软件设定上拉电阻。 P0.0-P0.7 也可用作外部中断输入(噪声滤波器，中断使能和标志位控制)。	E-4	24-26(22-24) 27(25) 28(26) 29(27) 30(28) 31(29)	INT0-INT2 INT3/BUZ INT4/SI INT5/SCK INT6/SO INT7/TBPWM
P1.0 P1.1 P1.2 P1.3 P1.4	I/O	可对该口的每一位进行功能设定，可以设定为施密特触发器输入，推挽式输出或开漏输出。通过软件设定上拉电阻。	E-4	32(30) 33(31) 34(32) 35(33) 36(34)	T1OUT/T1PWM/ T1CAPT0CLK/ T1CLK T0OUT/T0PWM/ T0CAPTxD1 RxD1
P2.0 P2.1 P2.2 P2.3 P2.4 P2.5 P2.6 P2.7	I/O	可对该口的每一位进行功能设定，可以设定为输入或推挽式输出。通过软件设定上拉电阻。	F-1 F-3 F-1 F-2	37(35) 38(36) 39(37) 40(38) 41(39) 42(40) 43(41) 44(42)	AD0/FCLK AD1/C0OUT AD2/C0P AD3/C0N AD4/C1N AD5/C1P AD6/C1OUT AD7/VBLDREF
P3.0 P3.1 P3.2 P3.3 P3.4 P3.5 P3.6 P3.7	I/O	可对该口的每一位进行功能设定，可以设定为输入或推挽式输出。通过软件设定上拉电阻。 P3.0-P3.3 也可用作外部中断输入(噪声滤波器，中断使能和标志位控制)。	H-9 D-1	7(5) 8(6) 9(7) 10(8) 11(9) 12(10) 13(11) 14(12)	SEG52/INT11 SEG53/INT10 SEG54/INT9 SEG55/INT8 SEG56/TACLK SEG57/TAOUT/ TAPWM/TACAP TxD0 RxD0
P4.0-P4.7	I/O	可对该口的每一位进行功能设定，可以设定为输入，推挽式输出或开漏输出。通过软件设定上拉电阻。	H-8	99-100,1-6 (97-100,1-4)	SEG44-SEG51
P5.0-P5.7	I/O	可对该口的每一位进行功能设定，可以设定为输入，推挽式输出或开漏输出。通过软件设定上拉电阻。	H-8	91-98 (89-96)	SEG36-SEG43
P6.0-P6.7	I/O	可对该口的每一位进行功能设定，可以设定为输入或推挽式输出。通过软件设定上拉电阻。	H-9	83-90 (81-88)	SEG28-SEG35
P7.0-P7.7	I/O	可对该口2位一组分别进行功能设定，可以设定为输入或推挽式输出。通过软件设定上拉电阻。	H-9	75-82 (73-80)	SEG20-SEG27
P8.0 - P8.7	I/O	可对该口2位或4位一组分别进行功能设	H-9	67-74	SEG12-SEG19

管脚名称	输入/输出	管脚特性描述	管脚类型	管脚号 (注释)	复用管脚
		定，可以设定为输入或推挽式输出。通过软件设定上拉电阻。		(65-72)	
P9.0-P9.5	I/O	可对该口6位一组分别进行功能设定，可以设定为输入或推挽式输出。通过软件设定上拉电阻。	H-9	61-66 (59-64)	SEG6-SEG11
P10.0-P10.1 P10.2-P10.7	I/O	可对该口1位，2位或4位一组分别进行功能设定，可以设定为输入或推挽式输出。通过软件设定上拉电阻。	H-9	53-54(51-52) 55-60(53-58)	COM0-COM1 COM2-COM7/ SEG0-SEG5
INT0-INT2 INT3 INT4 INT5 INT6 INT7 INT8 - INT11	I/O	外部中断输入管脚	E-4 H-9	24-26(22-24) 27(25) 28(26) 29(27) 30(28) 31(29) 10-7 (8-5)	P0.0-P0.2 P0.3/BUZ P0.4/SI P0.5/SCK P0.6/SO P0.7/TBPWM P3.3-P3.0/ SEG55-SEG52
T0CAP	I/O	Timer 0 capture 输入	E-4	34(32)	P1.2/T0OUT/ T0PWM
T0CLK	I/O	Timer 0 外部时钟源输入	E-4	33(31)	P1.1/T1CLK
T0OUT/T0PWM	I/O	Timer 0 时钟输出及 PWM 输出	E-4	34(32)	P1.2/T0CAP
T1CAP	I/O	Timer 1 capture 输入	E-4	32(30)	P1.0/T1OUT/ T1PWM
T1CLK	I/O	Timer 1 外部时钟源输入	E-4	33(31)	P1.1/T0CLK
T1OUT/T1PWM	I/O	Timer 1 时钟输出及 PWM 输出	E-4	32(30)	P1.0/T1CAP
TACAP	I/O	Timer A capture (捕获)输入	H-9	12(10)	P3.5/SEG57/ TAOUT/TAPWM
TACLK	I/O	Timer A 外部时钟源输入	H-9	11(9)	P3.4/SEG56
TAOUT/ TAPWM	I/O	Timer A 时钟输出及 PWM 输出	H-9	12(10)	P3.5/SEG57/ TACAP
TBPWM	I/O	Timer B PWM 输出	E-4	31(29)	P0.7/INT7
FCLK	I/O	频率计数器外部时钟源输入	F-1	37(35)	P2.0/AD0
BUZ	I/O	蜂鸣器信号输出管脚	E-4	27(25)	P0.3/INT3
SI SCK SO	I/O	串行时钟，串行数据输出，串行数据输入	E-4	28-30 (26-28)	P0.4/INT4 P0.5/INT5 P0.6/INT6
TxD0 RxD0	I/O	Uart 0 数据输出，输入	D-1	13(11) 14(12)	P3.6 P3.7
TxD1 RxD1	I/O	Uart 1 数据输出，输入	E-4	35(33) 36(34)	P1.3 P1.4
AD0 AD1 AD2	I/O	模数转换器模拟输入通道	F-1 F-3	37(35) 38(36) 39(37)	P2.0/FCLK P2.1/C0OUT P2.2/C0P

管脚名称	输入/输出	管脚特性描述	管脚类型	管脚号 (注释)	复用管脚
AD3 AD4 AD5 AD6 AD7			 F-1 F-2	40(38) 41(39) 42(40) 43(41) 44(42)	P2.3/C0N P2.4/C1N P2.5/C1P P2.6/C1OUT P2.7/VBLDREF
V_{BLDREF}	I/O	电池电压检测参考电压	F-2	44(42)	P2.7/AD7
COM0 - COM1 COM2 - COM7	I/O	LCD 公共端信号输出	H-9	53-54(51-52) 55-60(53-58)	P10.0-P10.1 P10.2-P10.7/ SEG0-SEG5
SEG0-SEG5 SEG6-SEG11 SEG12-SEG19 SEG20-SEG27 SEG28-SEG35 SEG36-SEG43 SEG44-SEG45 SEG46-SEG47 SEG48-SEG51 SEG52 SEG53 SEG54 SEG55 SEG56 SEG57	I/O	LCD 段信号输出	H-9 H-8 H-9	55-60 (53-58) 61-66(59-64) 67-74(65-72) 75-82(73-80) 83-90(81-88) 91-98(89-96) 99,100(97,98) 1,2(99,100) 3-6(1-4) 7(5) 8(6) 9(7) 10(8) 11(9) 12(10)	P10.2-P10.7/ COM2-COM7 P9.0-P9.5 P8.0-P8.7 P7.0-P7.7 P6.0-P6.7 P5.0-P5.7 P4.0-P4.1 P4.2-P4.3 P4.4-P4.7 P3.0/INT11 P3.1/INT10 P3.2/INT9 P3.3/INT8 P3.4/TACLK P3.5/TAOUT/ TAPWM/TACAP
C0OUT C1OUT	I/O	比较器0/1输出	F-1	38(36) 43(41)	P2.1/AD1 P2.6/AD6
C0P C1P	I/O	比较器0/1正向输入端	F-3	39(37) 42(40)	P2.2/AD2 P2.5/AD5
C0N C1N	I/O	比较器0/1反向输入端	F-3	40(38) 41(39)	P2.3/AD3 P2.4/AD4
VLC0-VLC3	–	LCD 电源	–	52-49 (50-47)	–
CA CB	–	升压器电容终端	–	48(46) 47(45)	–
AV_{REF}	–	模数转换器参考电压	–	45(43)	–
AV_{SS}	–	模数转换器地	–	46(44)	–
V_{REG}	O	使用副时钟时，稳压器电压输出 (需接0.1μF电容)	–	23(21)	–
nRESET	I	系统复位管脚	B	22(20)	–
XT_{IN}，XT_{OUT}	–	副时钟晶振管脚	–	20,21(18,19)	–
X_{IN}，X_{OUT}	–	主振荡器管脚	–	18,17(16,15)	–

管脚名称	输入/输出	管脚特性描述	管脚类型	管脚号 (注释)	复用管脚
TEST	I	测试输入：必须接到 V_{SS}	–	19(17)	–
V_{DD}，V_{SS}	–	电源输入管脚	–	15,16(13,14)	–

注释: 括号内的数字为 100-TQFP-1414 封装对应的管脚号。

1.7 管脚电路

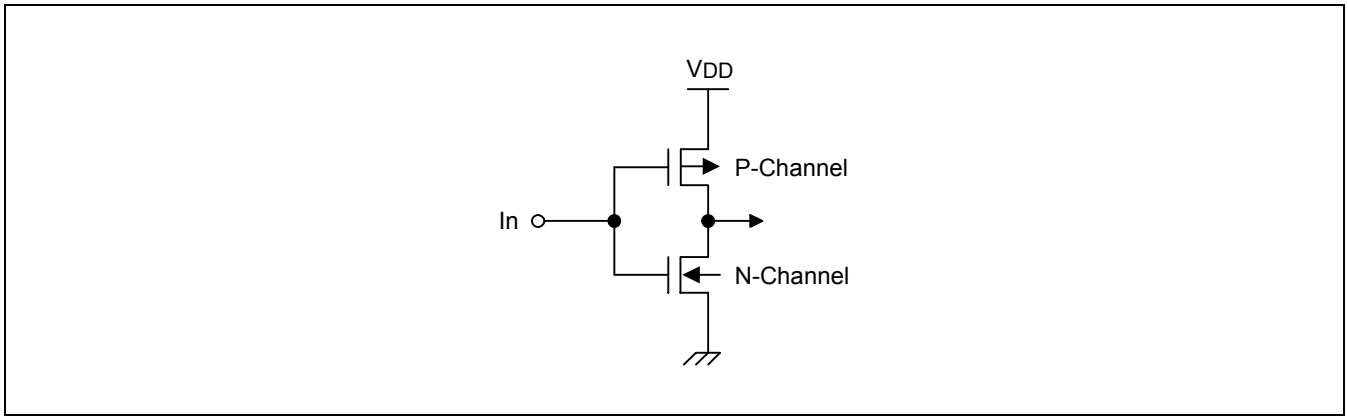


图 1-4 管脚电路类型 A

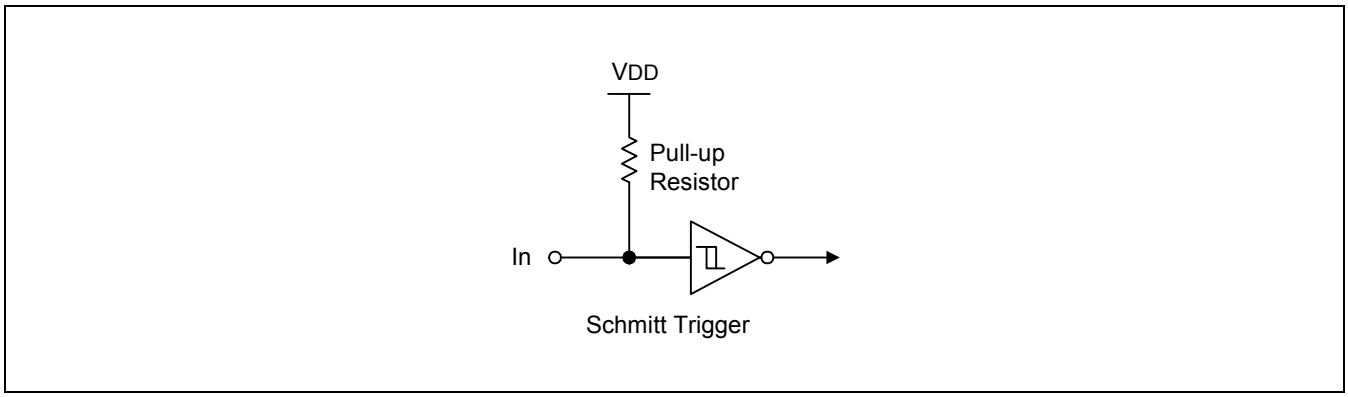


图 1-5 管脚电路类型 B

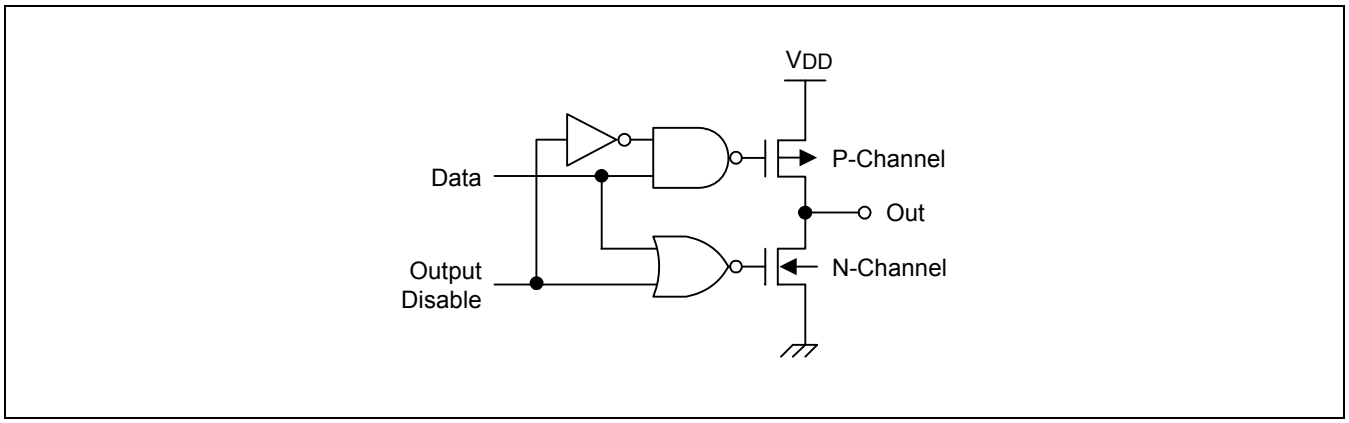


图 1-6 管脚电路类型 C

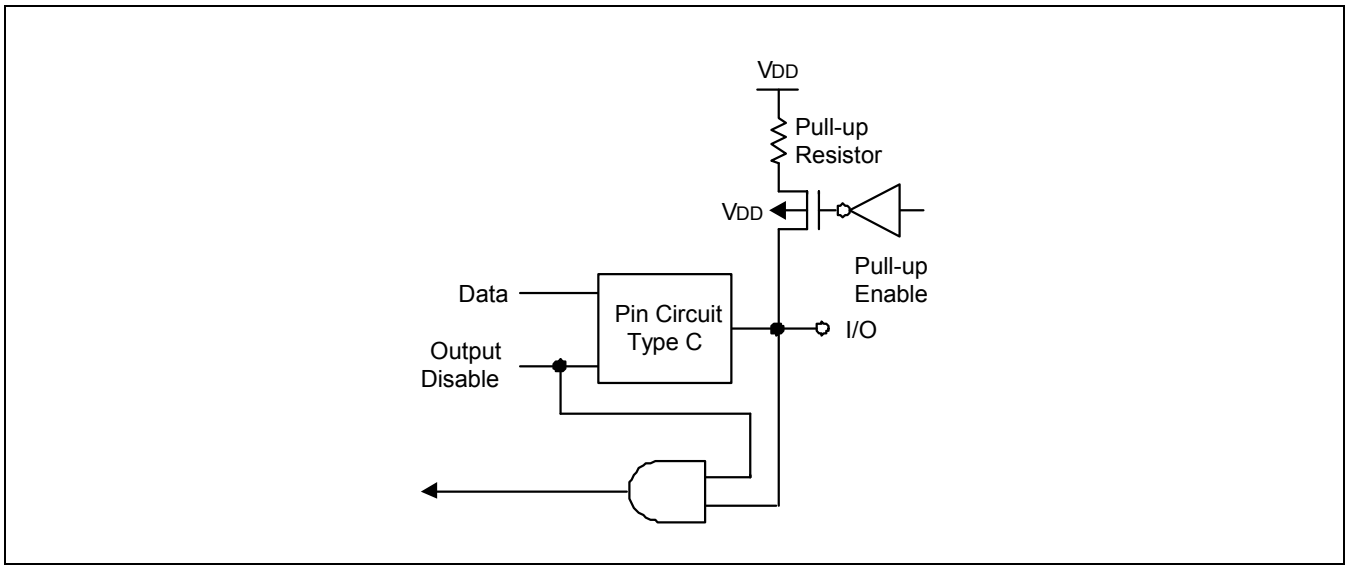


图 1-7 管脚电路类型 D-1 (P3.6，P3.7)

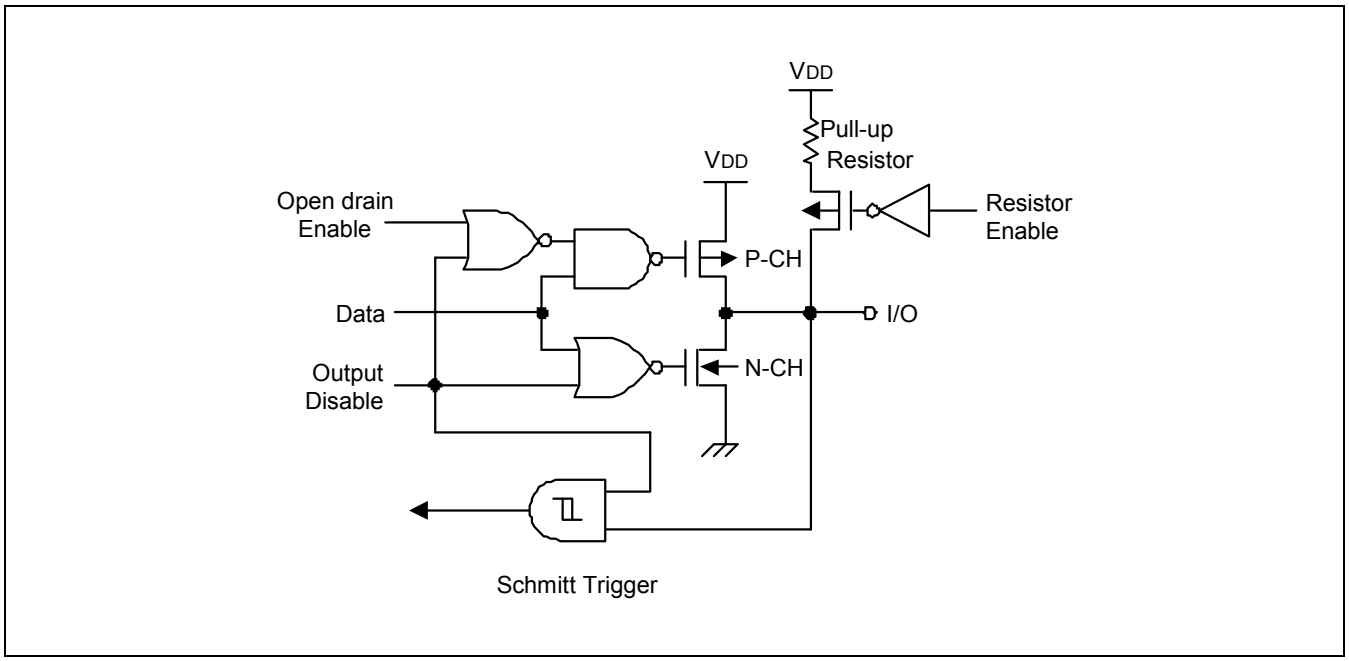


图 1-8 管脚电路类型 E-4 (P0，P1)

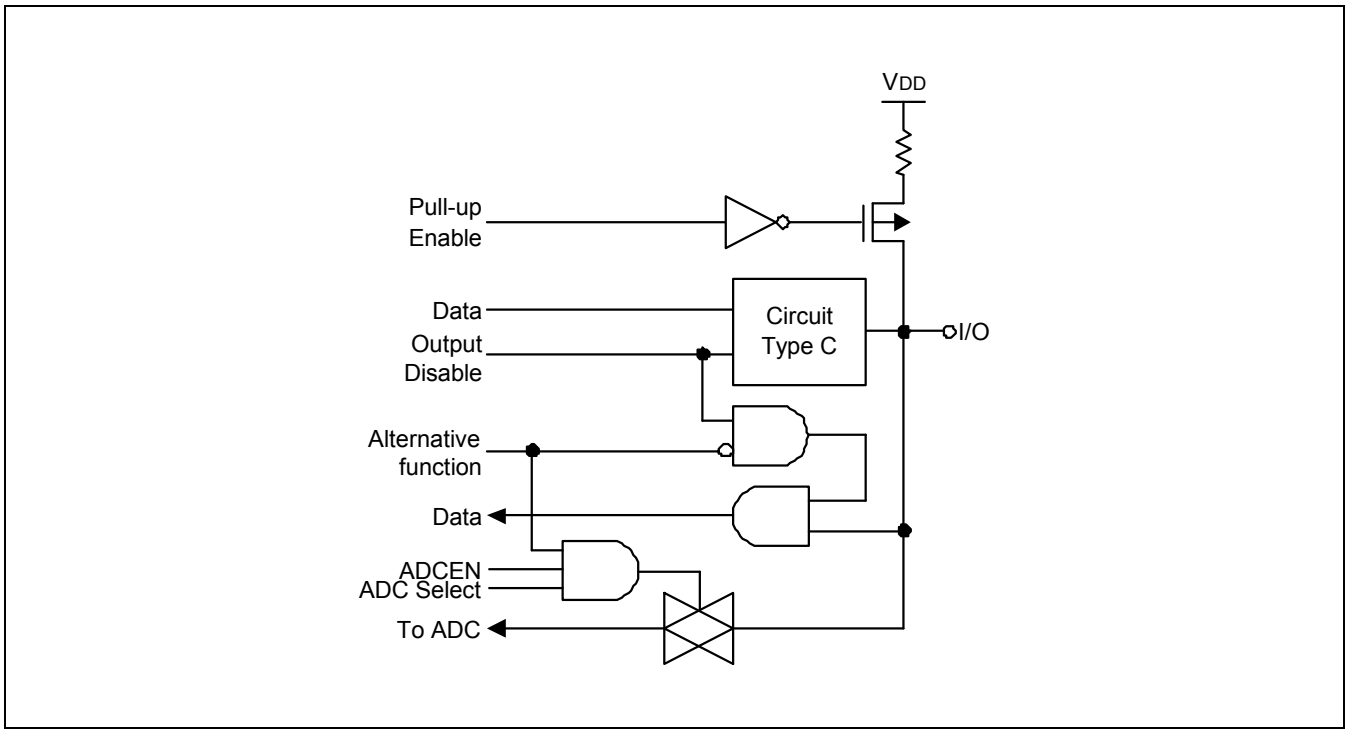


图 1-9 管脚电路类型 F-1 (P2.0，P2.1，P2.6)

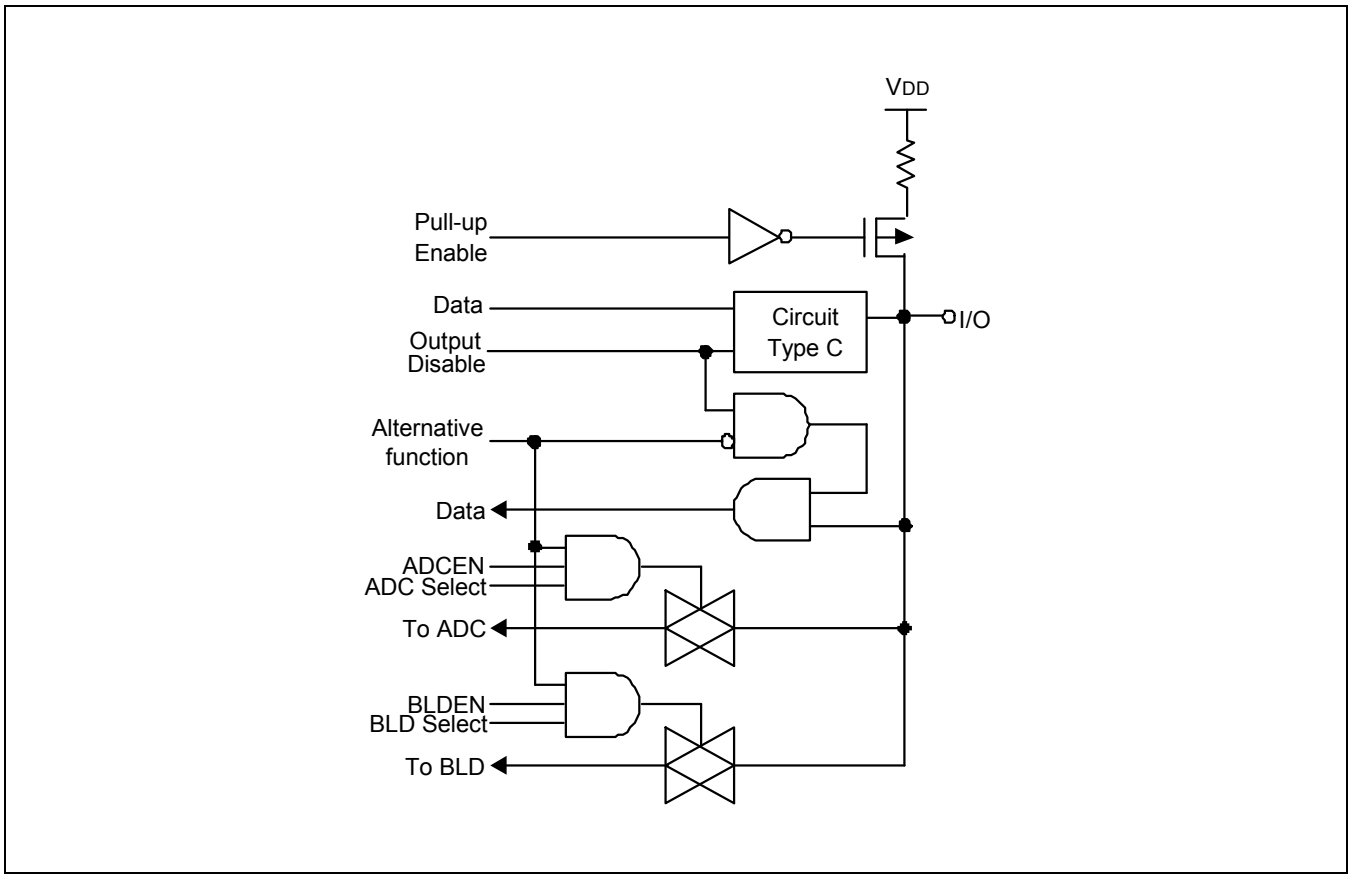


图 1-10 管脚电路类型 F-2 (P2.7)

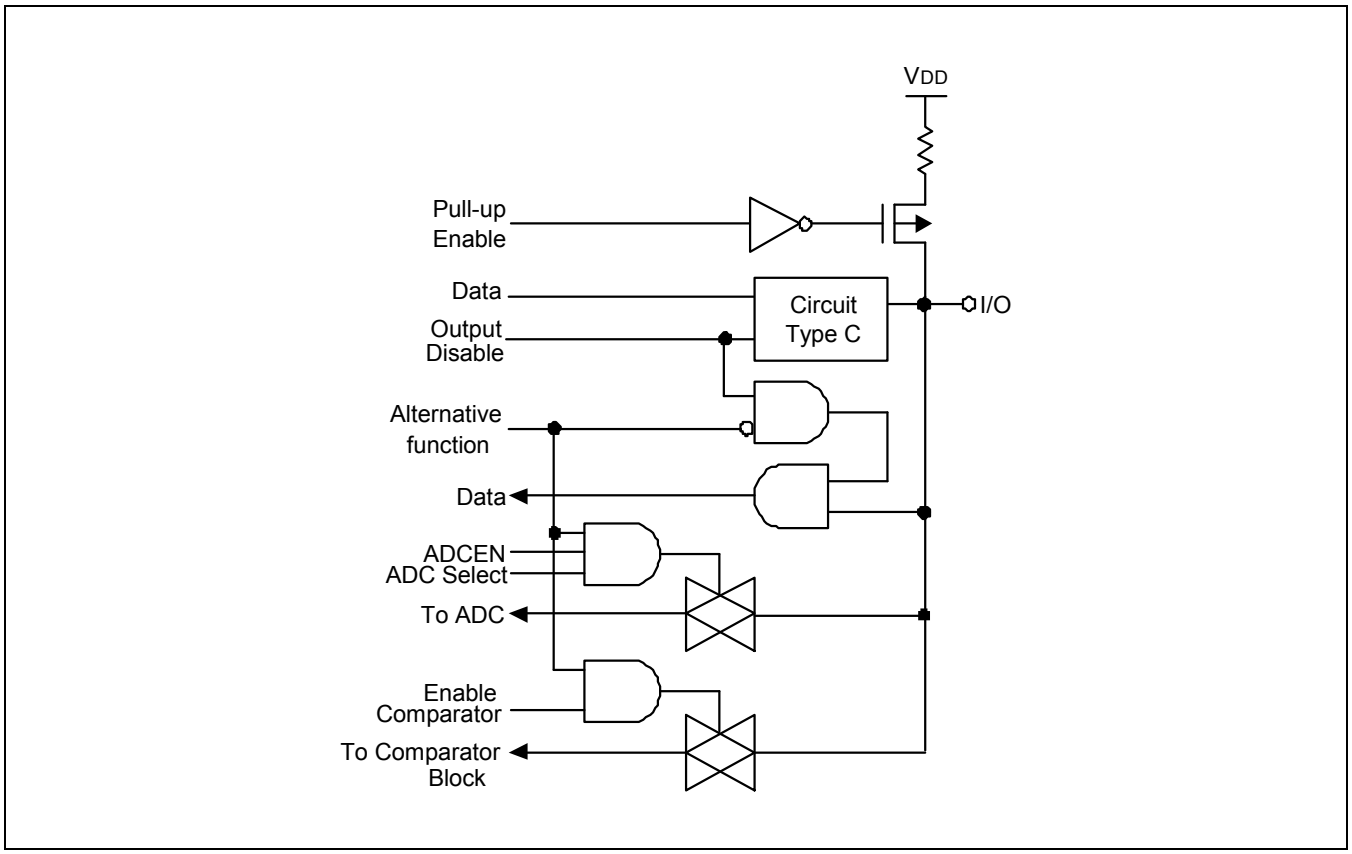


图 1-11 管脚电路类型 F-3 (P2.2-P2.5)

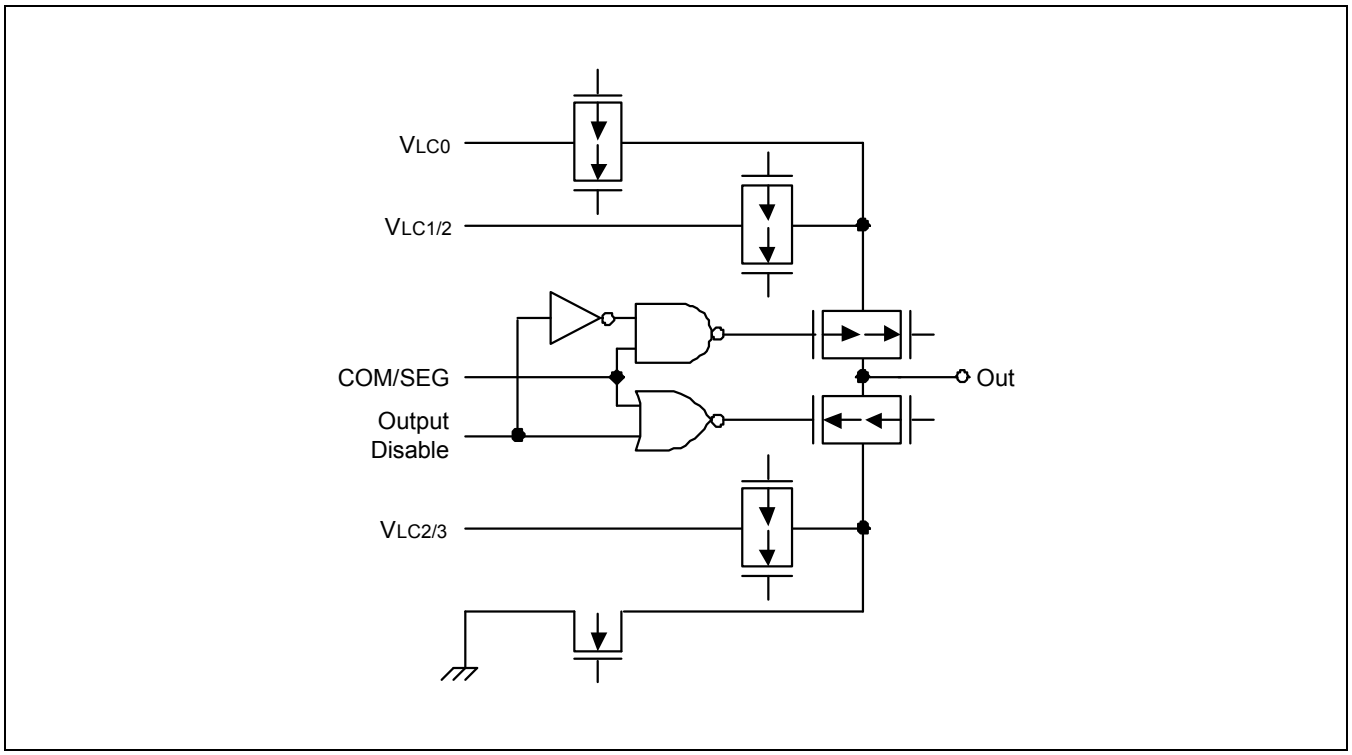


图 1-12 管脚电路类型 H-4

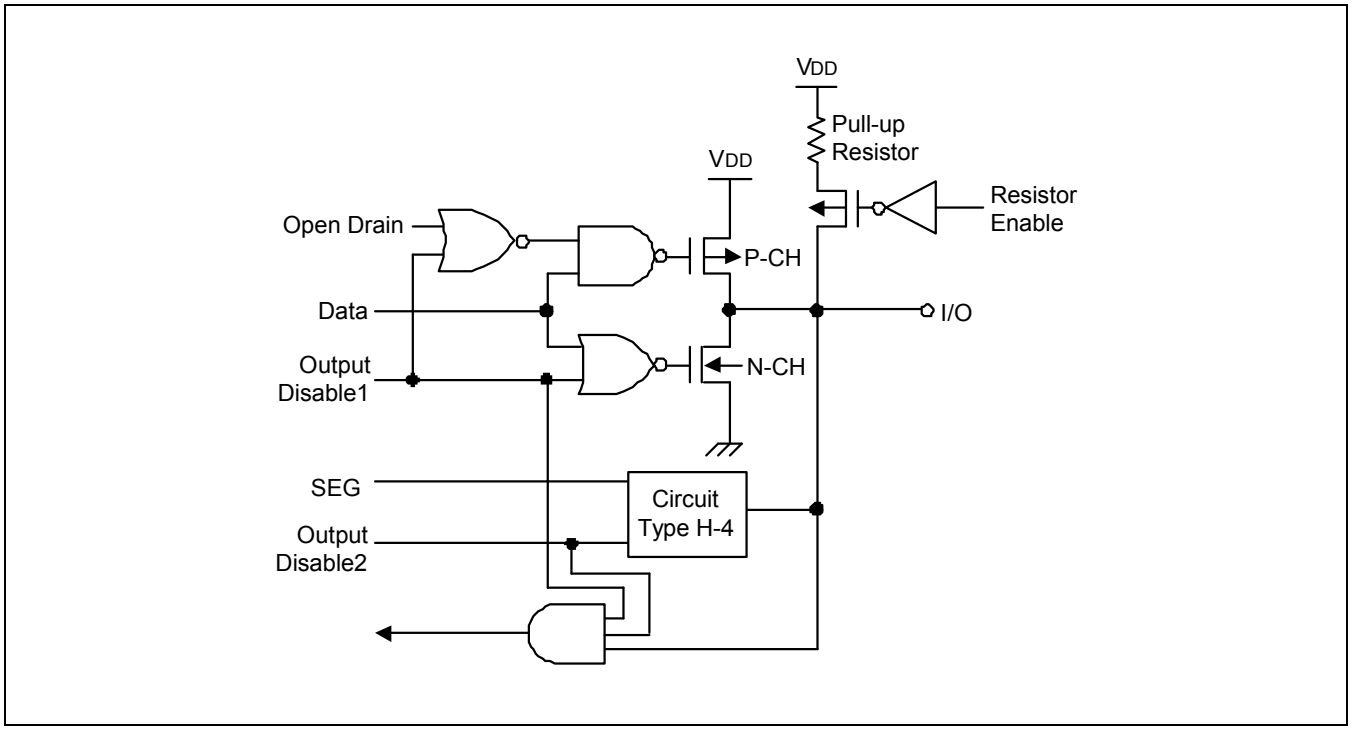


图 1-13 管脚电路类型 H-8 (P4，P5)

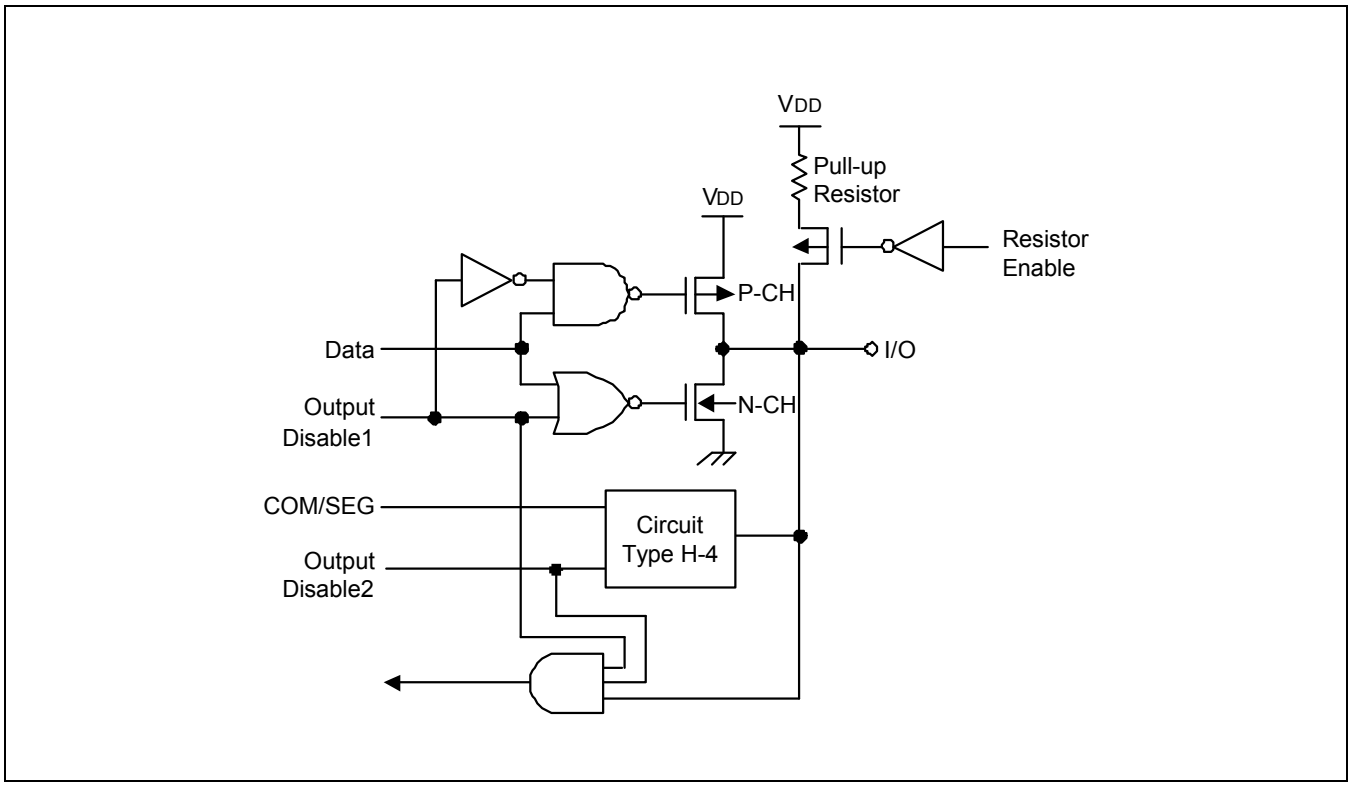


图 1-14 管脚电路类型 H-9 (P3.0-P3.5，P6，P7，P8，P9，P10)

2 地址空间

2.1 概述

S3F82HB MCU 有两类地址空间：

- 内部 full flash 程序存储空间 (ROM)
- 内部寄存器卷

MCU 通过16位的地址总线访问程序存储空间，通过独立的8位地址线和数据线访问内部寄存器卷。

S3F82HB 内集成了64K 字节 Flash 存储器。

256字节的物理寄存器空间通过不同的寻址方式被扩展成可寻址的320字节的空间。内建一个58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。S3F82HB 有64K 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，程序从0100H 开始执行。ROM 的复位地址可以在 smart option 中更改。具体详见 第19章 嵌入式闪存接口。

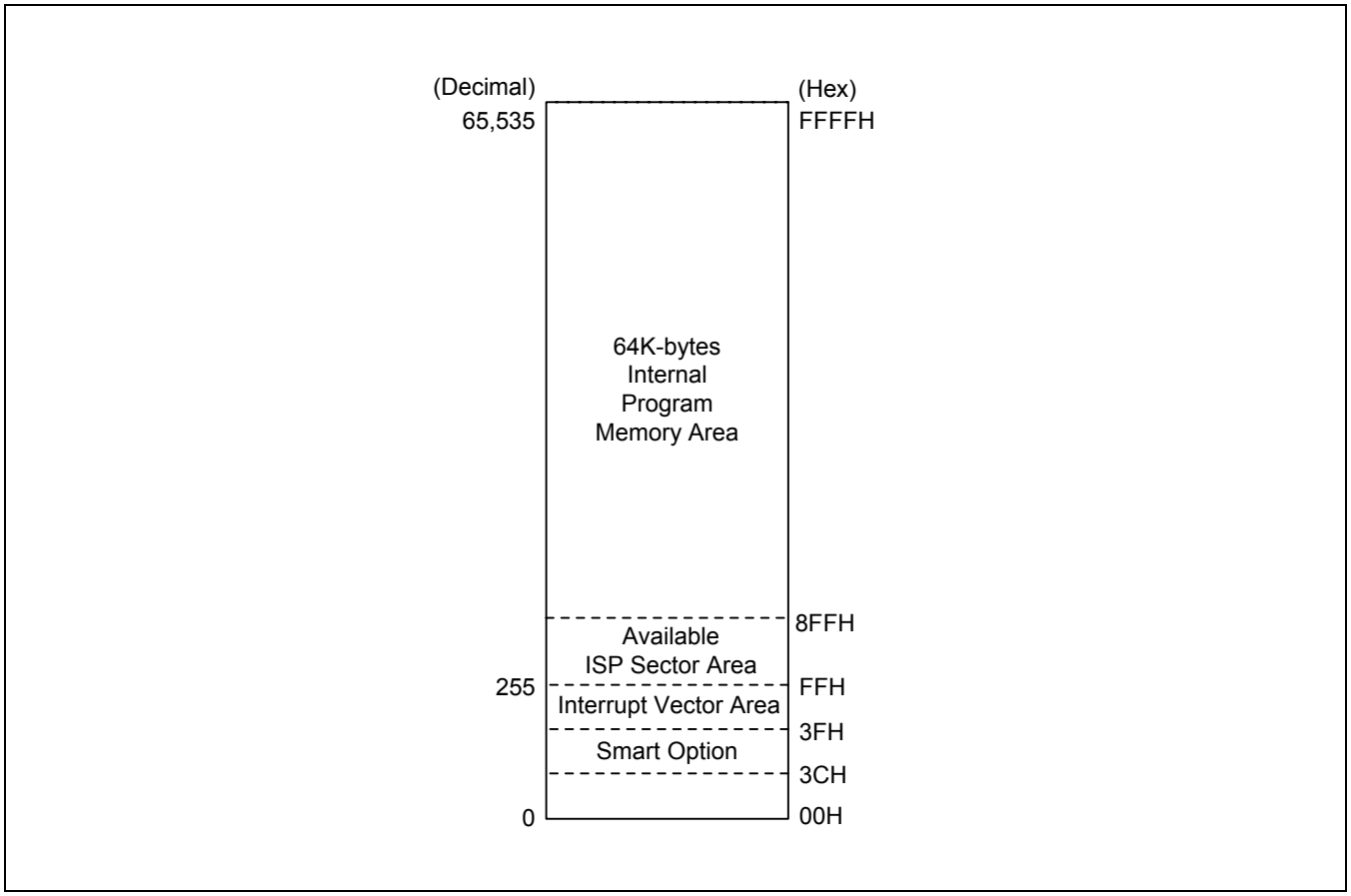
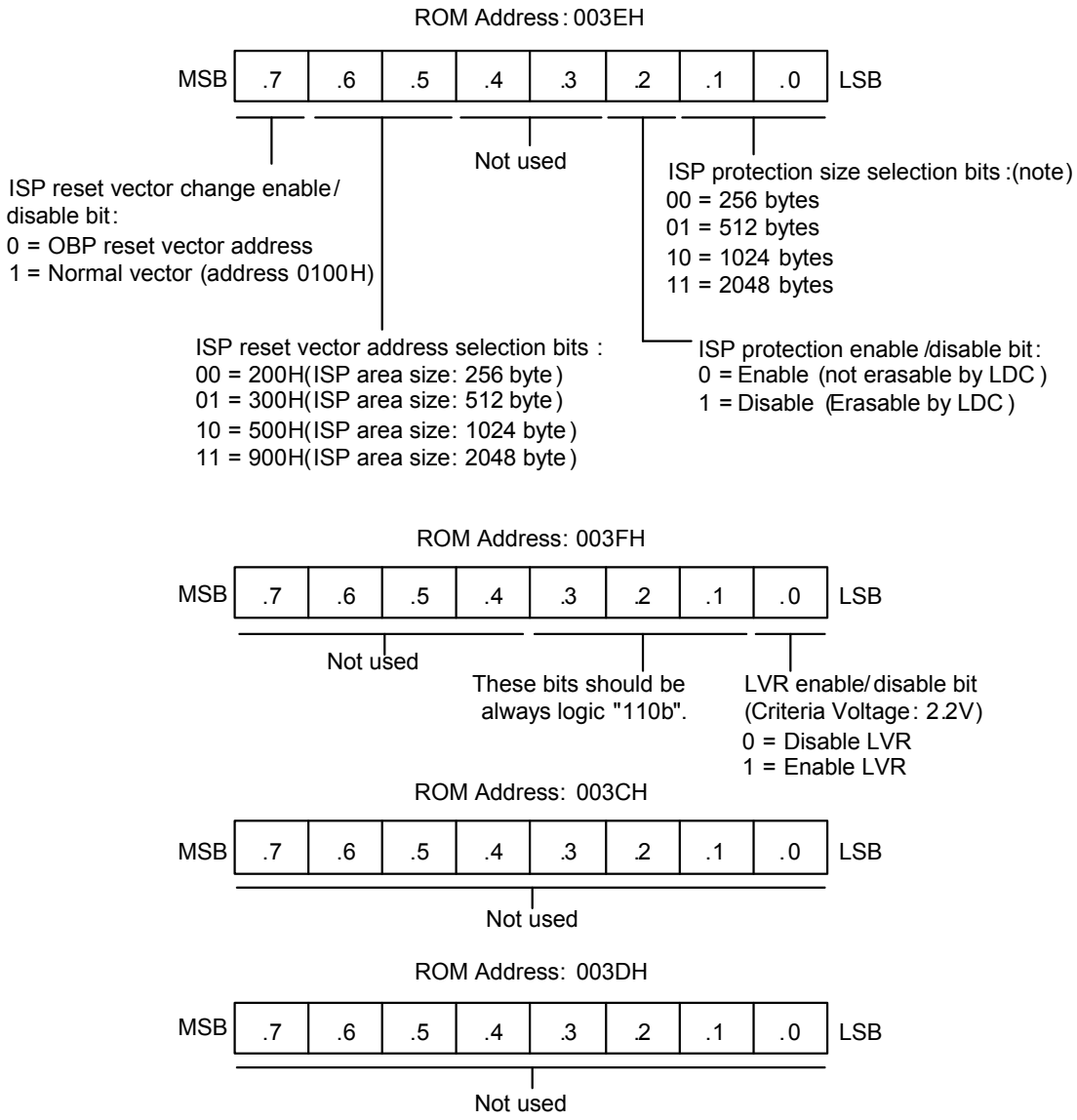


图 2-1 程序存储地址空间

2.2.1 SMART OPTION



注释：
1. 在选择ISP保护区间和ISP复位向量地址后，不要选择高于ISP区间的地址。
2. 无论用LDC指令对 Smart Option 区域 (003CH-003FH) 写入任何值，都不能改变 Smart Option 的值，Smart Option 区域 (003CH-003FH) 的值应该通过 OTP/MTP 工具 (SPW2+ 单芯片烧写器或者GW-PRO2 多芯片烧写器) 烧写。

图 2-2 Smart Option

Smart Option 是用来初始化芯片的ROM选项，决定了芯片的启动状态。ROM 中用于 Smart Option 的地址范围为003CH ~ 003FH。

2.3 寄存器结构（RAM）

S3F82HB 内部寄存器卷中的高64位被扩展成2个64字节区域，称为set 1和set 2。
Set1的高32字节区域再被扩展为2个32字节的寄存器 bank (bank 0 和 bank 1)，低32字节是单32字节通用区域。

S3F82HB 可寻址的8位寄存器个数为2,715。其中包含，13字节的 CPU 和系统控制寄存器，58字节的 LCD 数据寄存器，68字节的外设控制和数据寄存器，16字节的共享工作寄存器以及 page 0-page 9中的2,560个通用寄存器 (包括16字节的外设控制寄存器)。

不管当前选的是哪个寄存器页，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。

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

表 2-1 中总结了内部寄存器卷中特殊功能寄存器类型和所占字节数。

表 2-1 S3F82HB 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字节的通用工作寄存器卷，10个192字节的主寄存器卷 (包括外设控制寄存器)，以及10个64字节的 set 2卷) LCD 数据寄存器 CPU 和系统控制寄存器 时钟，外设，I/O 控制和数据映射寄存器	2,576 58 13 68
所有可寻址的字节数	2,715

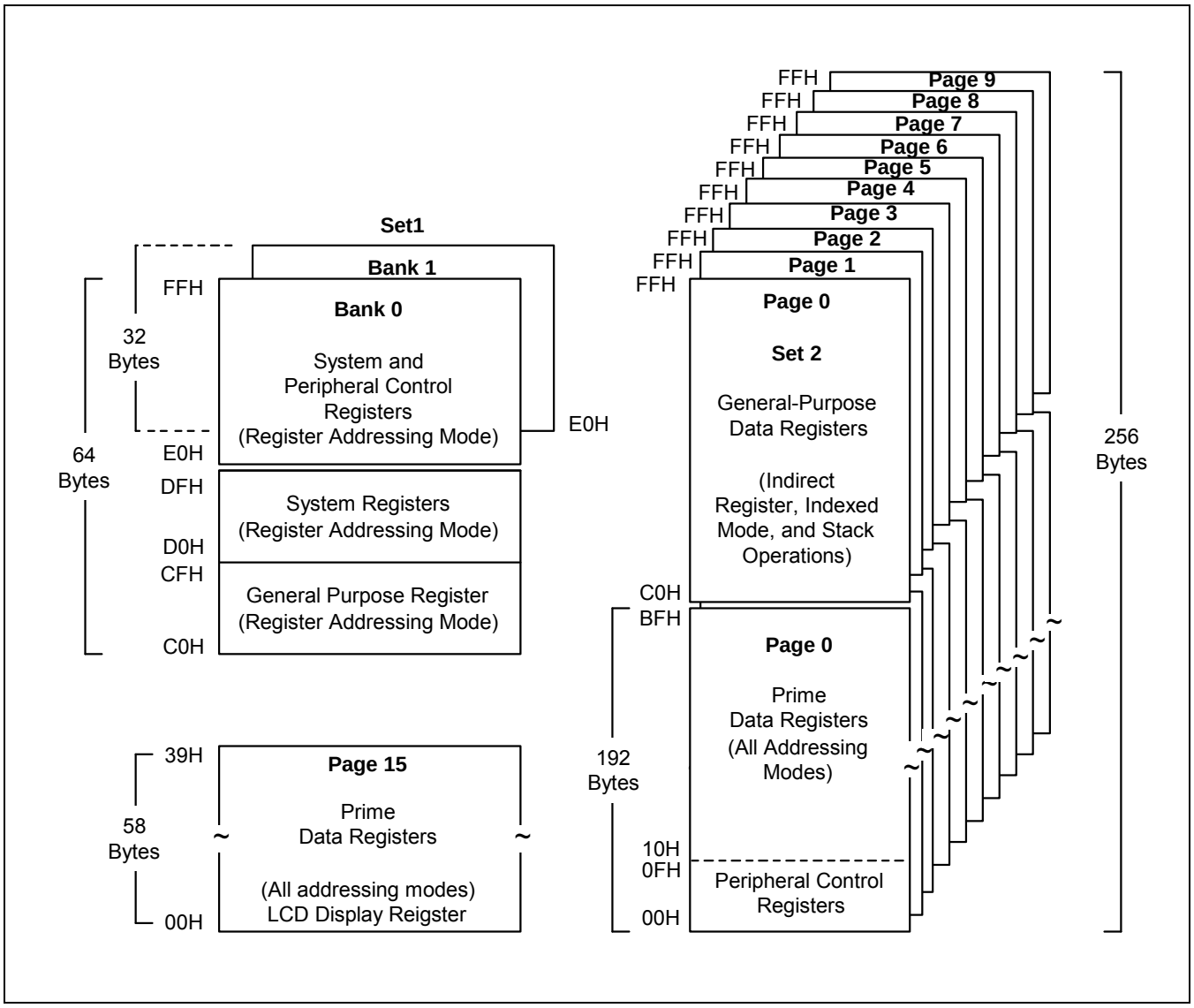


图 2-3 内部寄存器卷的地址空间 (S3F82HB)

2.3.1 寄存器页面指针(PP)

S3F8- 系列 MCU 支持对物理上256字节的内部寄存器页进行逻辑扩展 (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。页面寻址由寄存器页面指针 (PP，地址：DFH) 来控制。
S3F82HB 中将 LCD 数据寄存器扩展成页文件，所以访问其他页面时要改变寄存器页面指针。

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。

Register Page Pointer （PP）
DFH , Set 1, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB

Destination register page selection bits :

0000	Destination : Page 0
0001	Destination : Page 1
0010	Destination : Page 2
0011	Destination : Page 3
0100	Destination : Page 4
0101	Destination : Page 5
0110	Destination : Page 6
0111	Destination : Page 7
1000	Destination : Page 8
1001	Destination : Page 9
1111	Destination : Page 15
Others	Not used for the S3F82HB

Source register page selection bits :

0000	Source : page 0
0001	Source : page 1
0010	Source : page 2
0011	Source : page 3
0100	Source : page 4
0101	Source : page 5
0110	Source : page 6
0111	Source : page 7
1000	Source : page 8
1001	Source : page 9
1111	Source : page 15
Others	Not used for the S3F82HB

注释 ：
1. S3F82HB 将内部寄存器文件分为 12 页 (第0-9页, 第15页)。第0-9 页为通用寄存器 。
2. 第15 页可用作 LCD 数据寄存器或者通用寄存器 。

图 2-4 寄存器页面指针 (PP)

编程实例 2-1 在清RAM时使用页面指针 (Page 0，Page 1)

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

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

注释: DJNZ 指令的用法详见第 6-39 页。

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个系统和外设控制寄存器，
低32字节区域包含16个系统寄存器 (D0H-DFH) 和16字节的通用工作寄存器 (C0H-CFH)。
在其它寄存器空间执行的数据操作，也可以通过工作寄存器。

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，“寻址方式“)。

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 S3F82HB 中，page 0-9 可对 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 主寄存器区

S3F82HB 有10个寄存器页，每页有256个字节，其中低192字节 (00H–BFH) 称为主寄存器区。
主寄存器区可通过七种寻址模式中的任何一种进行访问 (详见第3章，“寻址方式”)。

第0页的主寄存器区在复位之后就可寻址。为访问第0，1，2，3，4，5，6，7，8，9，15页的主寄存器区，必须对寄存器页面指针 (PP) 的源和目标进行正确设置。

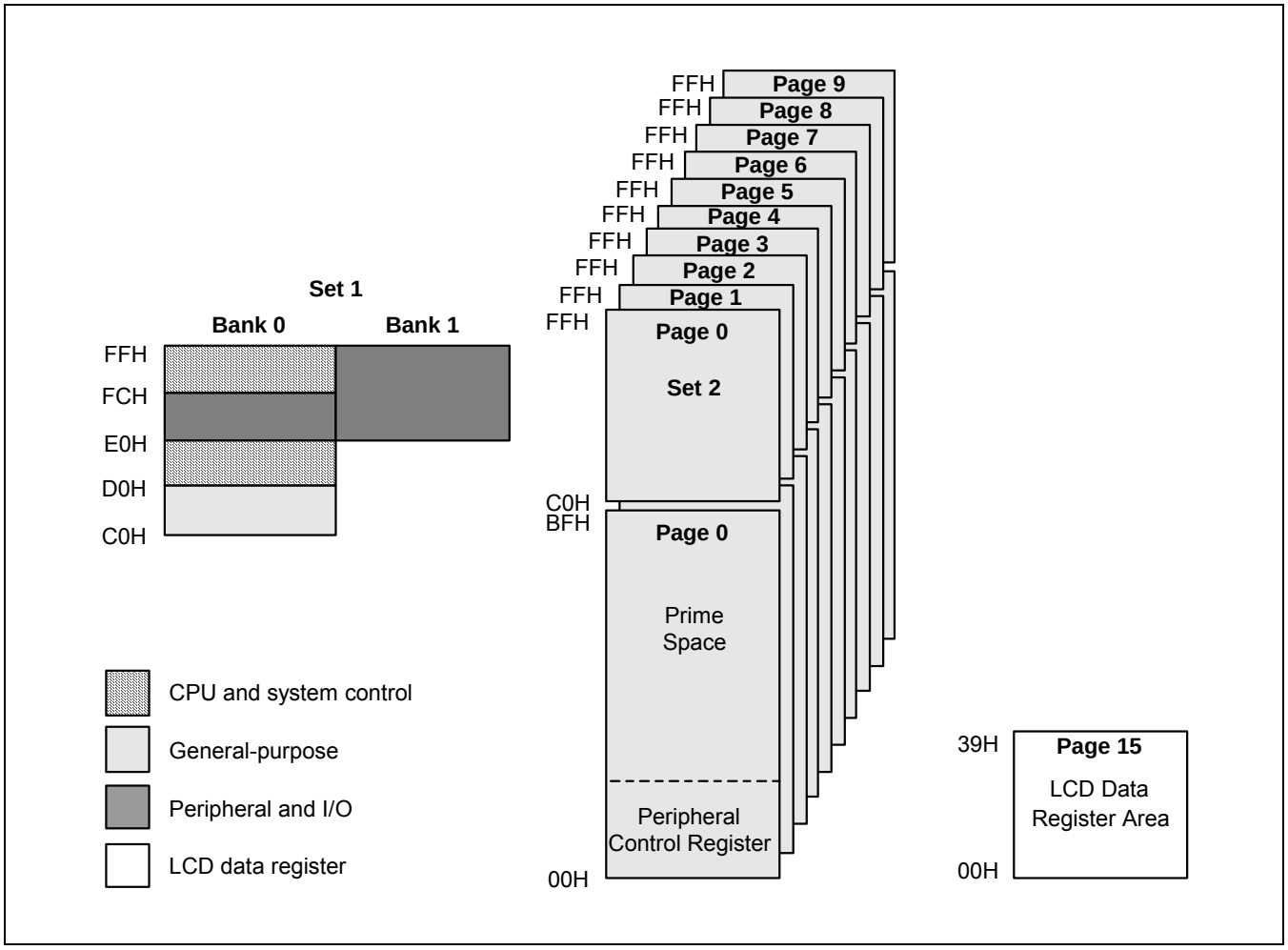


图 2-5 Set 1，Set 2，主寄存器区和 LCD 数据寄存器

2.3.5 工作寄存器

指令可通过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，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除 set 2以外的32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字节公共空间 (C0H-CFH)。

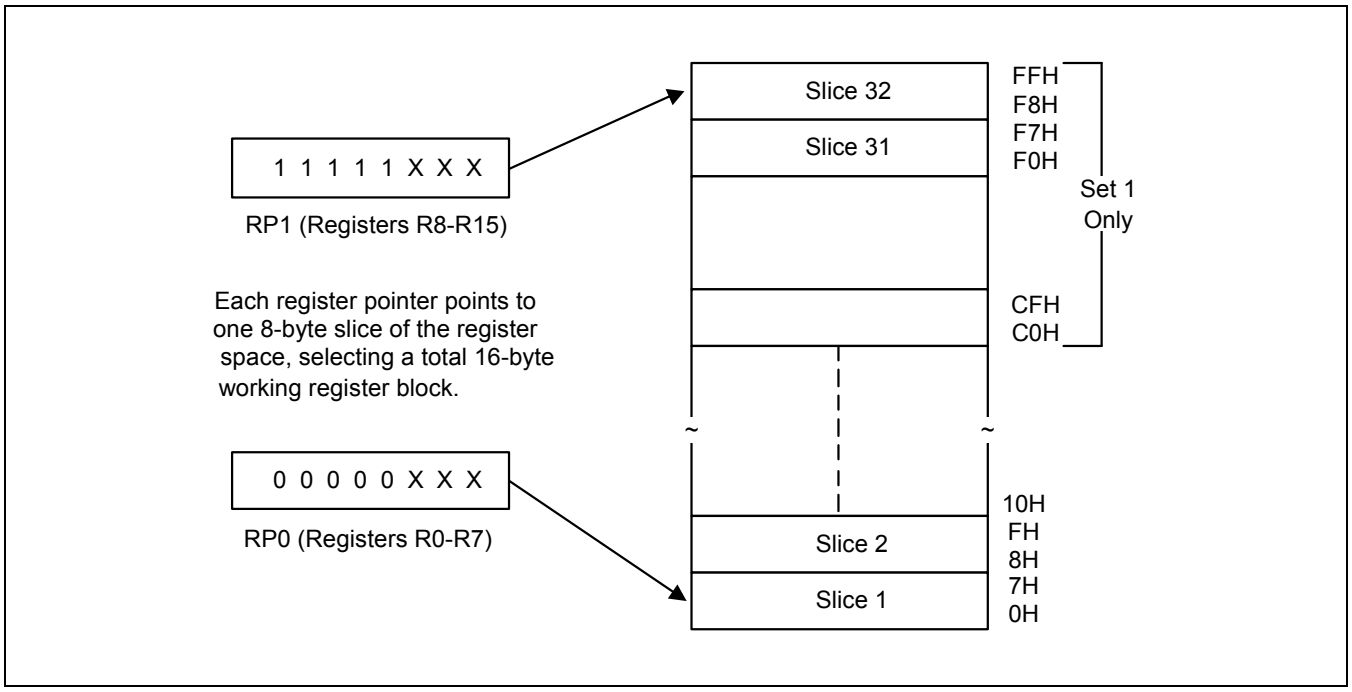


图 2-6 8字节工作寄存器区 (Slices)

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，RP# 指向工作寄存器的公共空间：RP0指向地址 C0H-C7H，RP1指向 C8H-CFH。

通过 SRP 或 LD 指令可以改变寄存器指针 RP0/ RP1的值 (*图 2-7* 和 *图 2-8*)。

用工作寄存器寻址方式，只能访问当前 RP0和 RP1指定的两个8字节的工作寄存器片。但不能用寄存器指针来选择 set 2 (C0-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和偏址寻址模式。

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 2-7*)。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 2-8*)，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两个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 ← no change, RP1 ← 48H,
SRP0  #0A0H       ; RP0 ← A0H, RP1 ← no change
CLR   RP0         ; RP0 ← 00H, RP1 ← no change
LD    RP1,#0F8H   ; RP0 ← no change, RP1 ← 0F8H
```

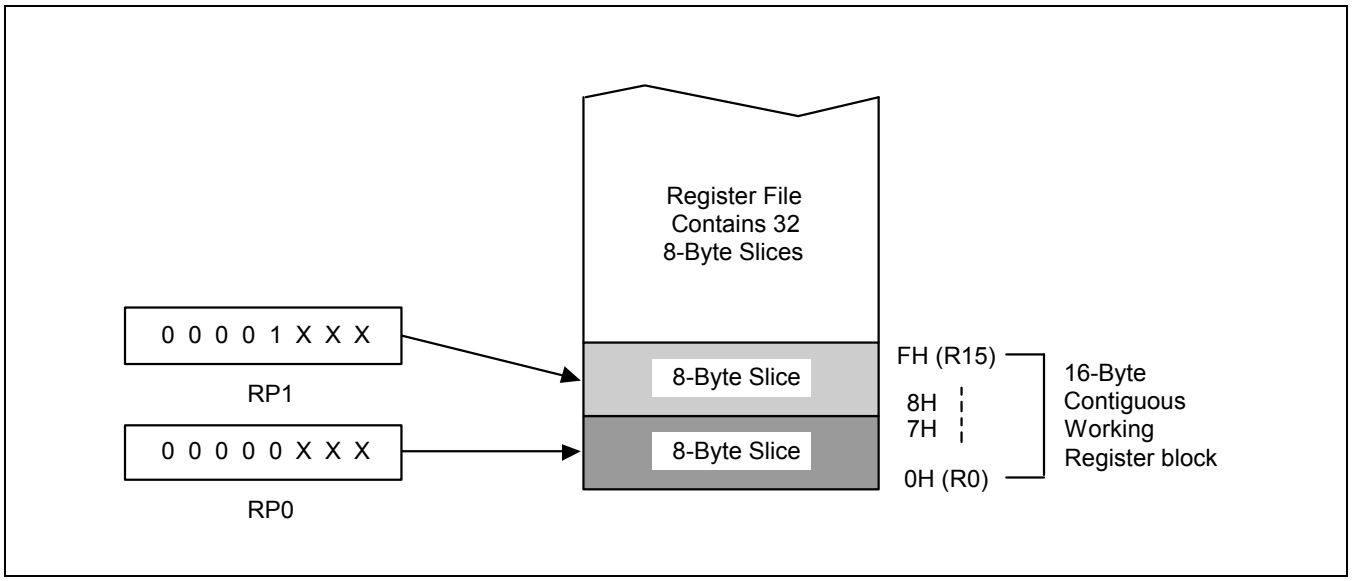


图 2-7 相邻的16字节工作寄存器块

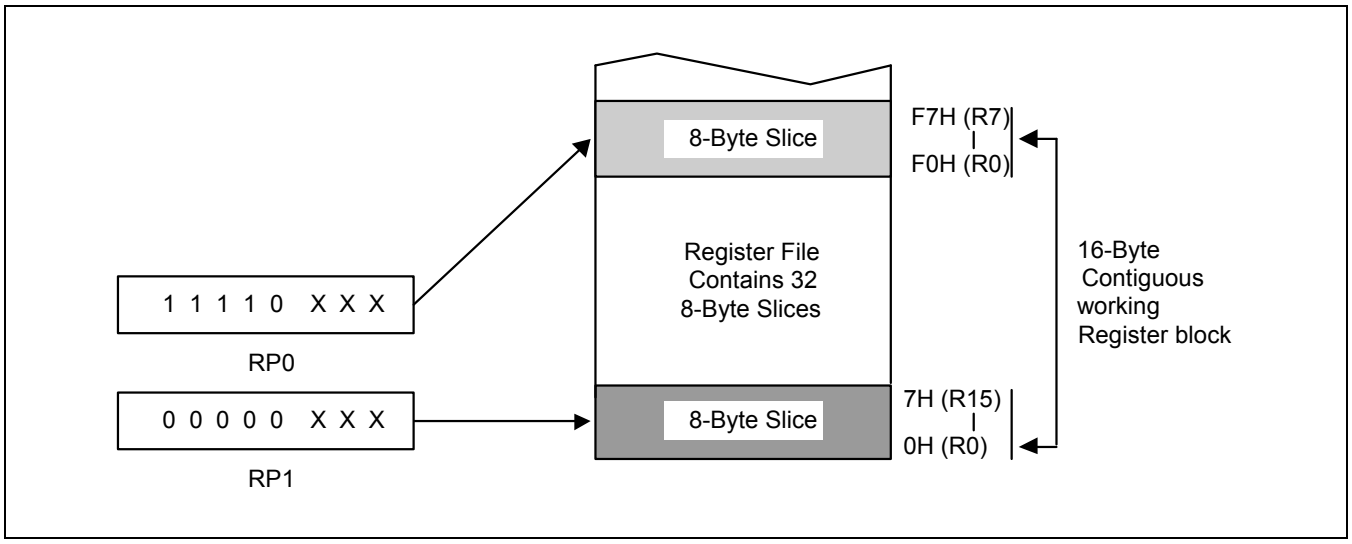


图 2-8 非相邻的16字节工作寄存器块

编程实例 2-3 使用 RP 对寄存器80H-85H 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

S3F8- 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，而另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。

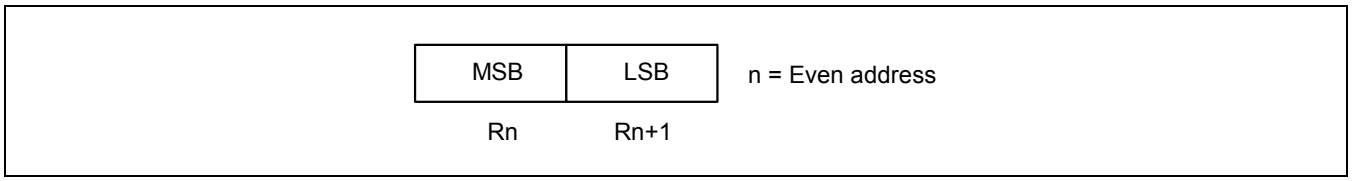


图 2-9 16位寄存器结构

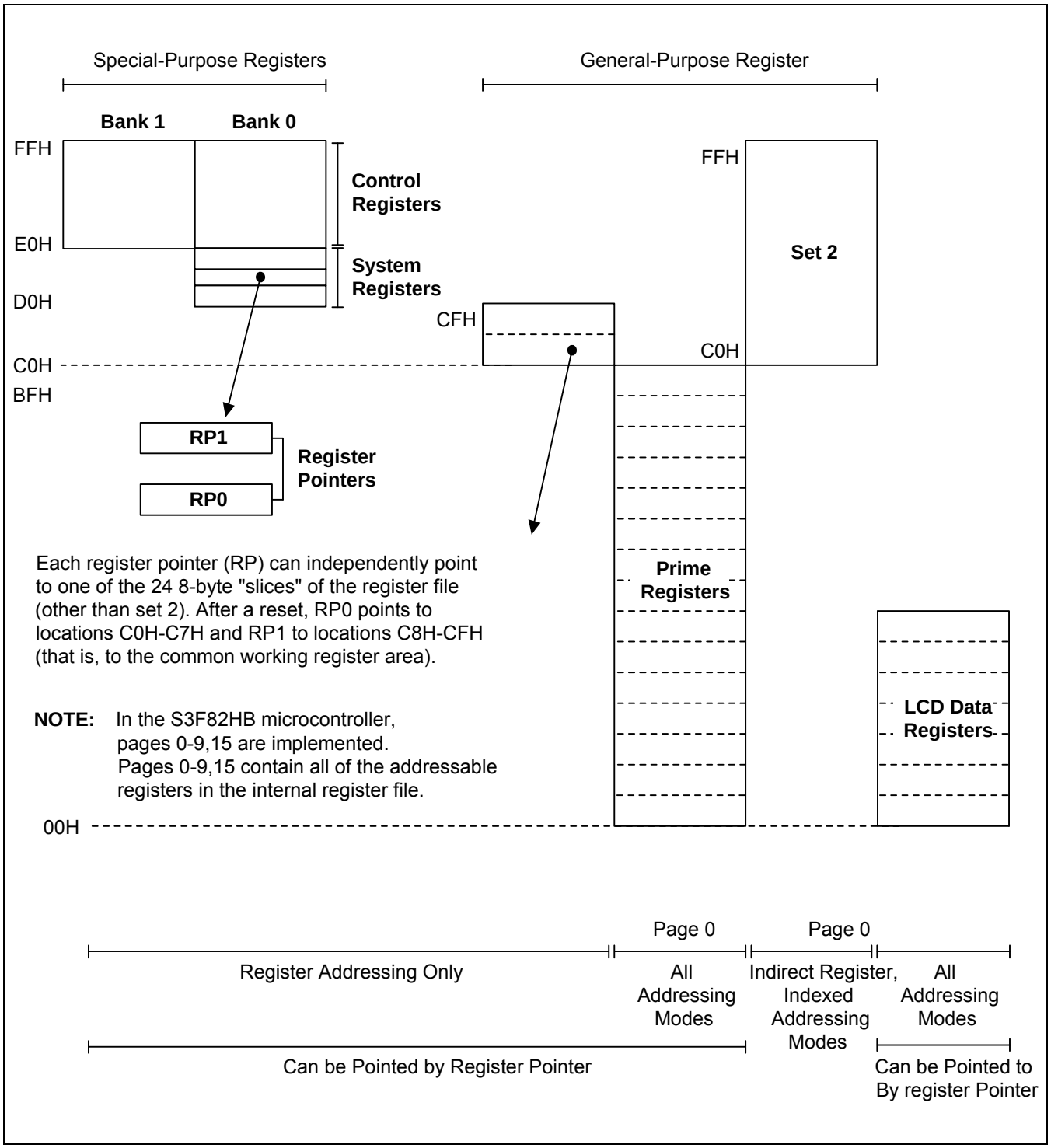


图 2-10 寄存器卷寻址模式

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。

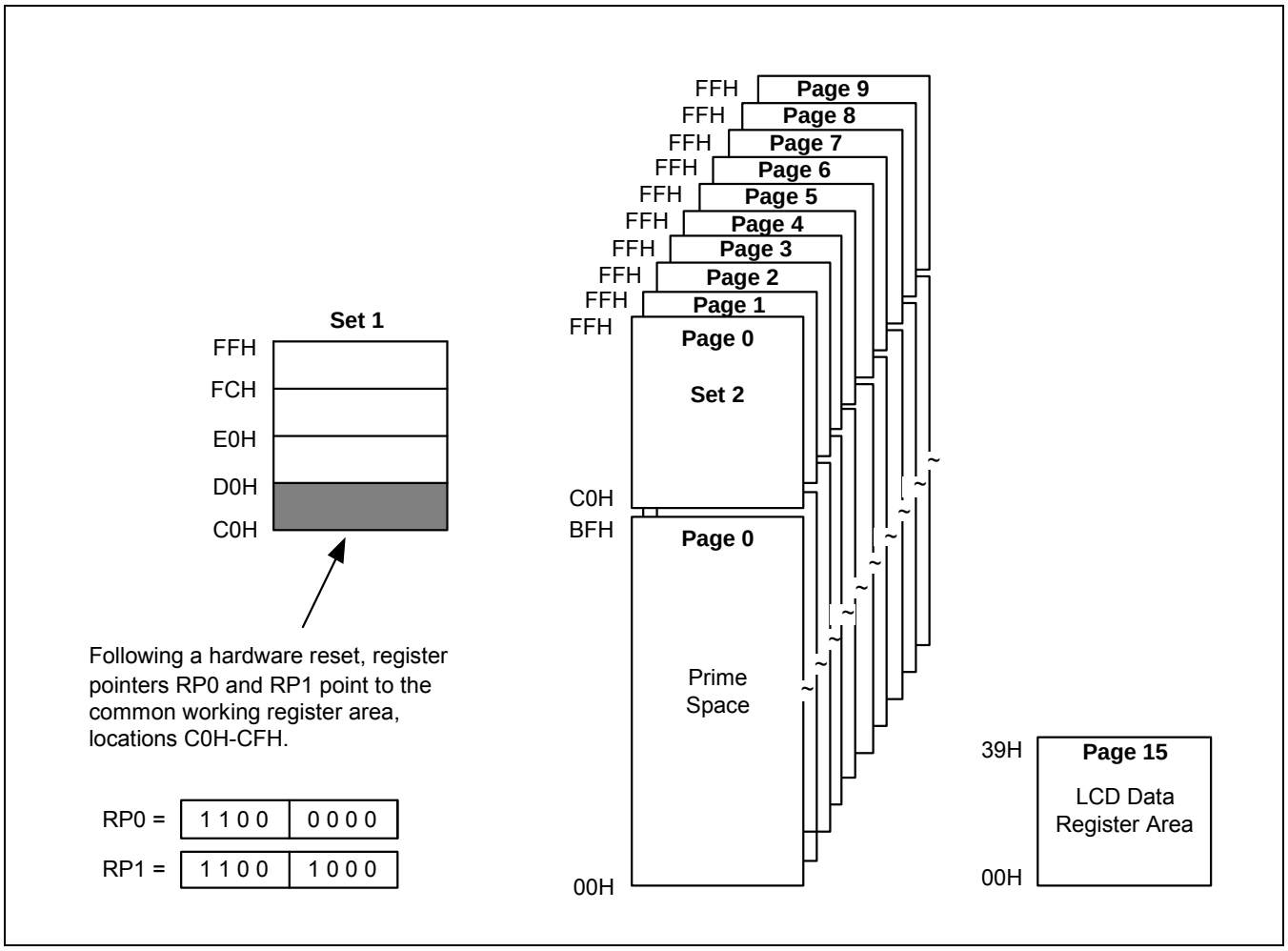


图 2-11 通用工作寄存器区

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
```

例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 位工作寄存器寻址方式

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字节寄存器片中的一个

图 2-12，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 2-13 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)。

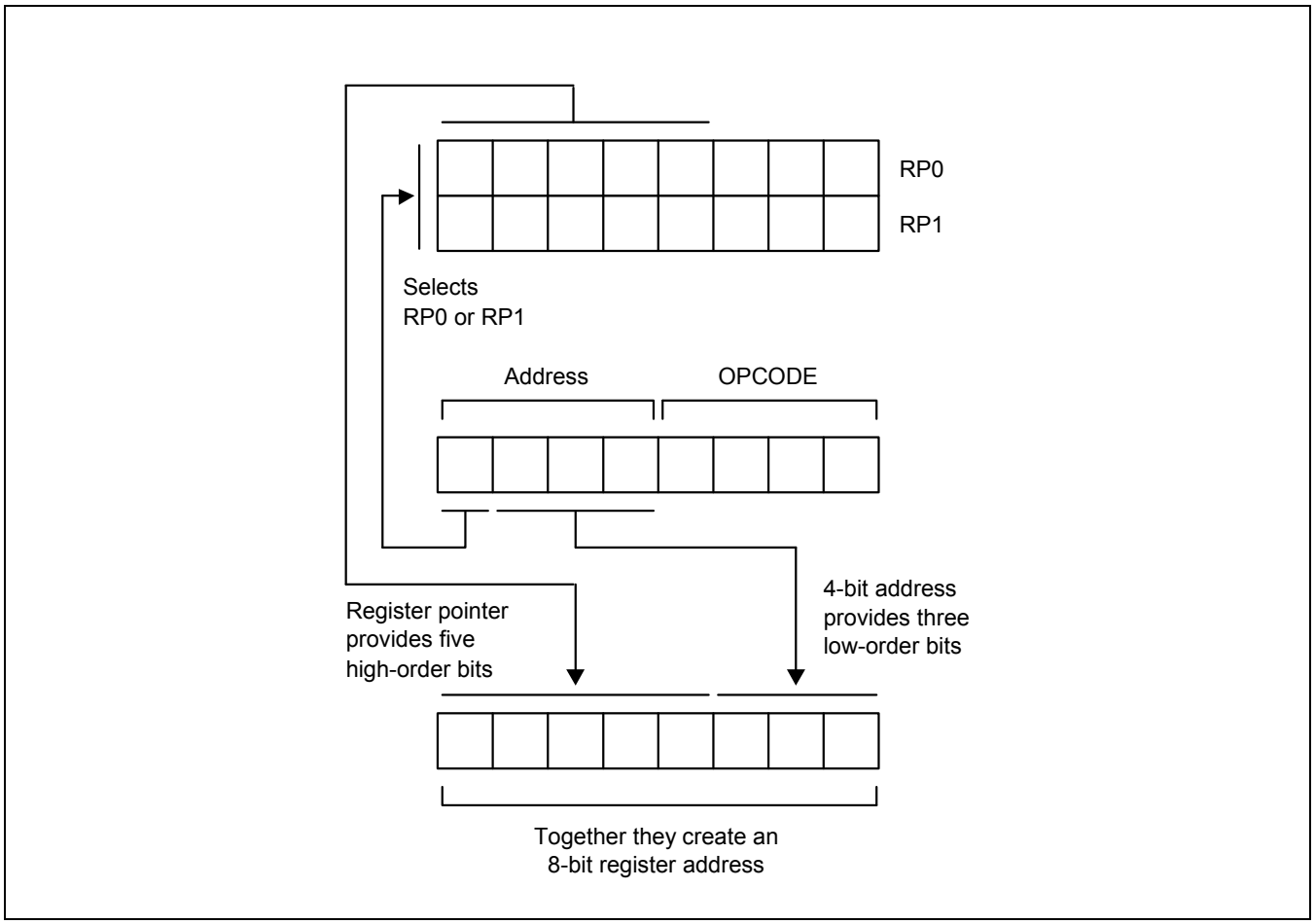


图 2-12 4位工作寄存器寻址方式

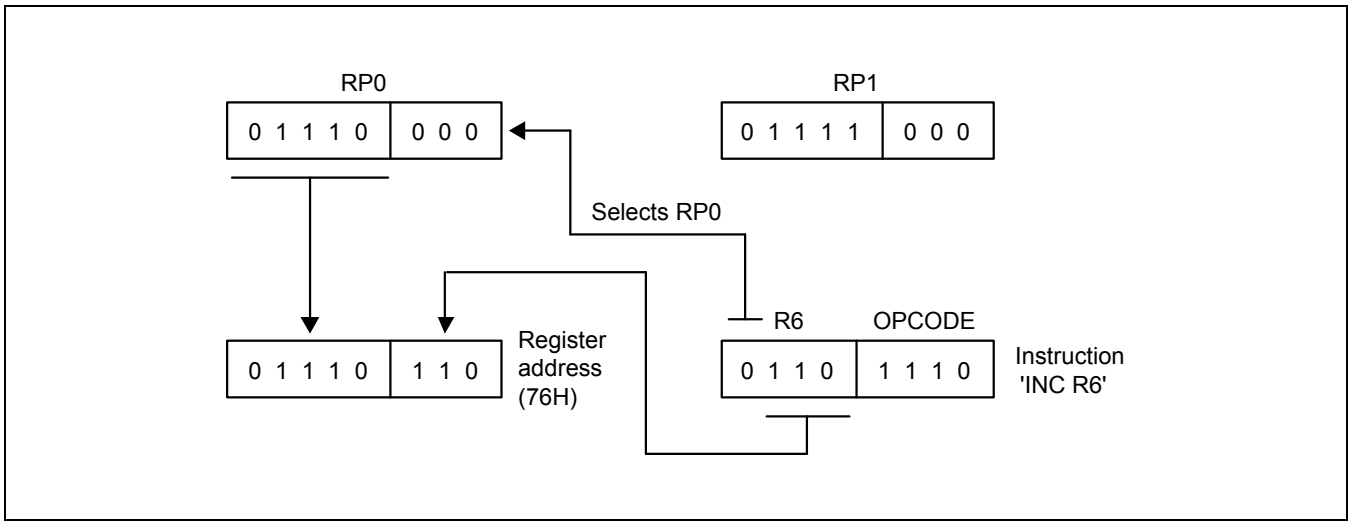


图 2-13 4位工作寄存器寻址实例

2.4.3 8 位工作寄存器寻址方式

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 2-14，8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 2-15 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3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)。

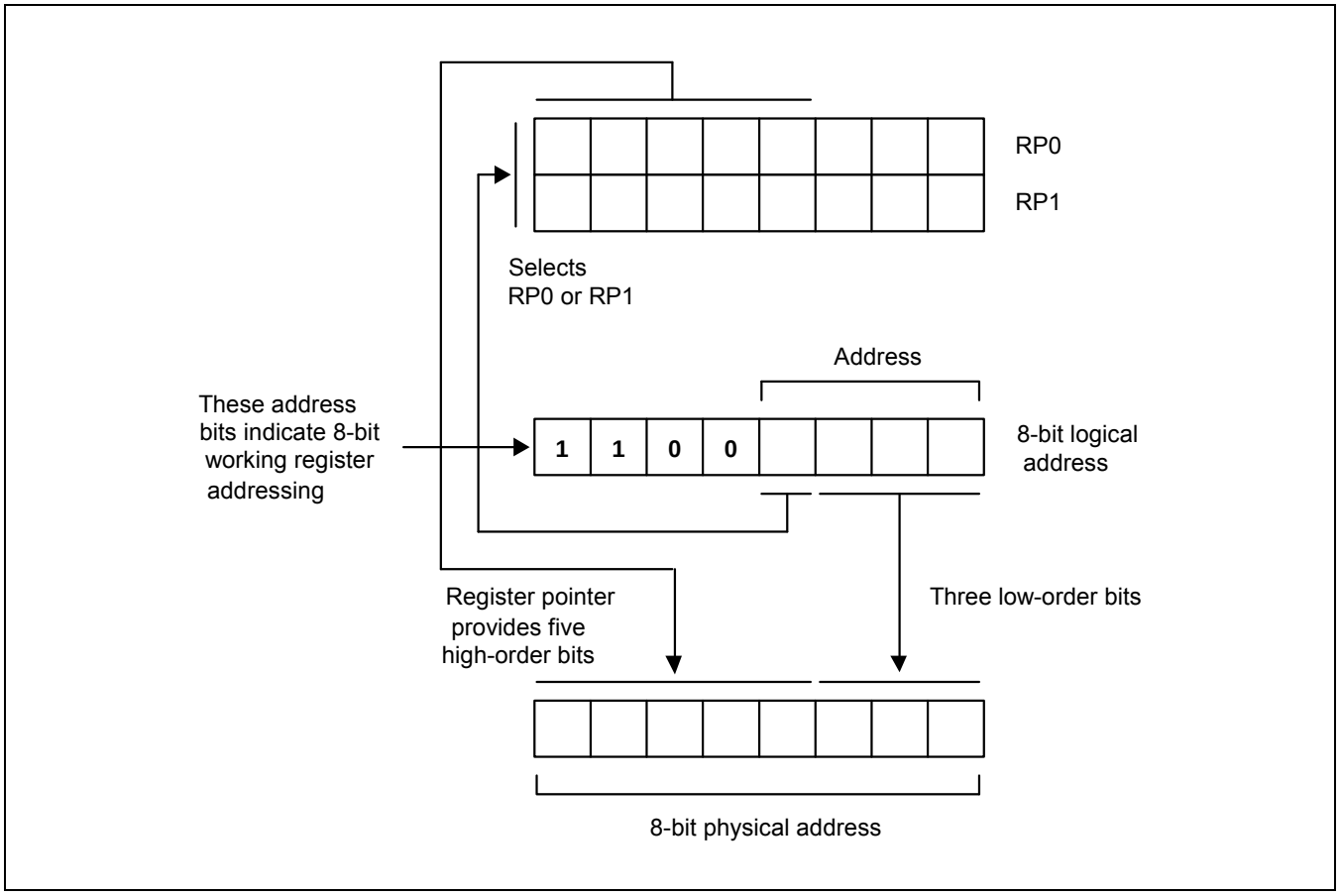


图 2-14 8位工作寄存器寻址方式

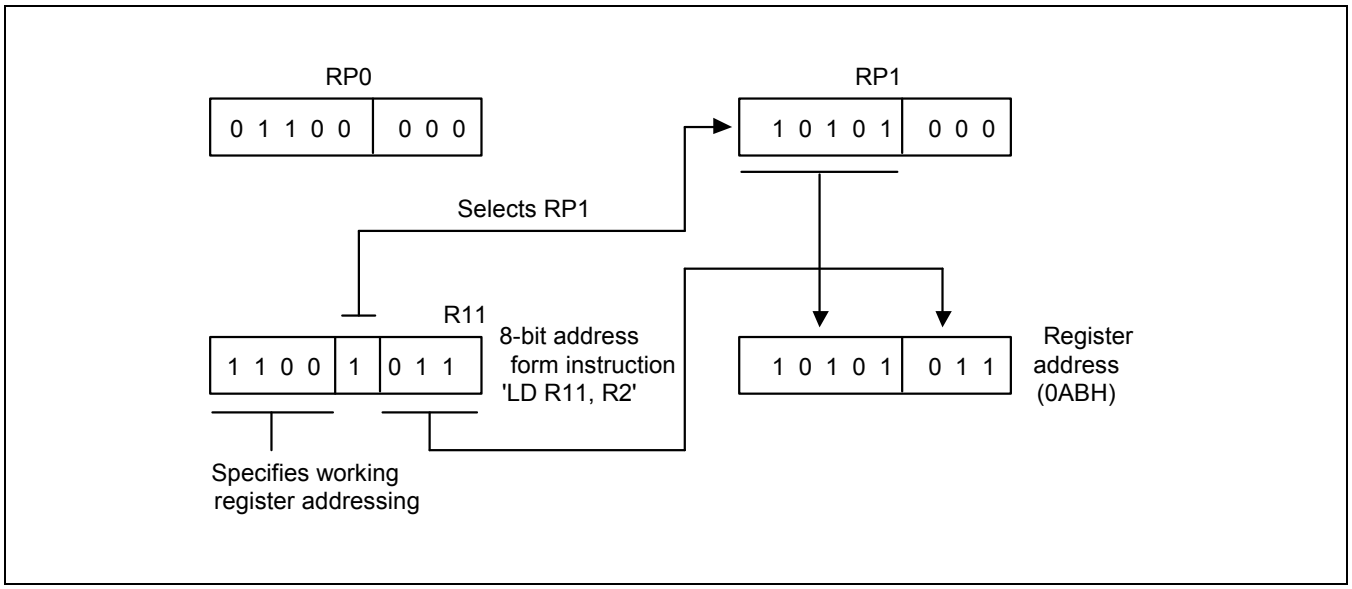


图 2-15 8位工作寄存器寻址实例

2.5 系统和用户栈

S3F8- 系列单片机通过栈来处理数据存储，子程序调用和返回。PUSH 和 POP 指令用于控制栈操作。S3F82HB 支持在内部寄存器卷中进行栈操作。

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 2-16* 所示。

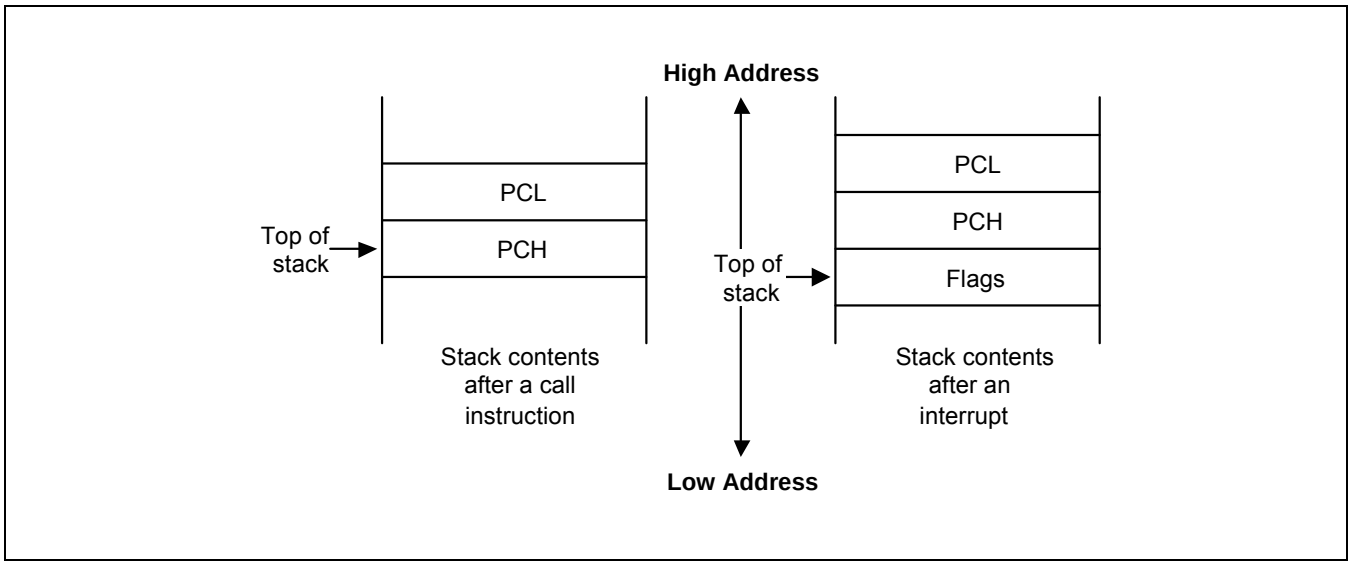


图 2-16 栈操作

2.5.2 用户自定义栈

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，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 S3F82HB 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。SAM88RC 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。

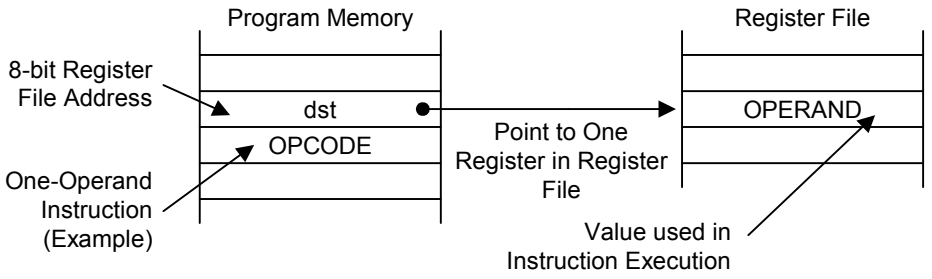
S3F8- 系列的指令集支持7种寻址模式，但这些寻址模式并非适用于所有指令。7种寻址模式和它们的符号表示：

- 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*。

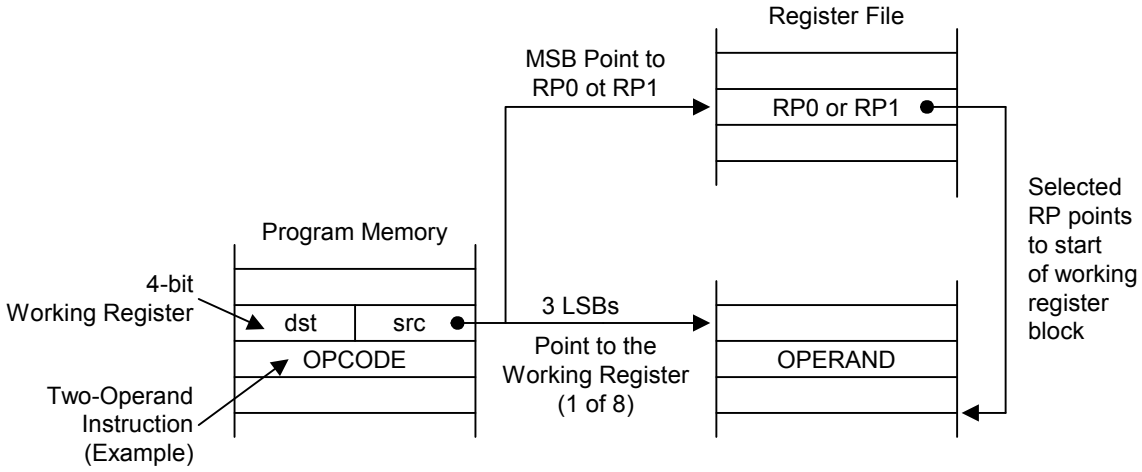
工作寄存器寻址模式与寄存器寻址模式不同。因为工作寄存器是通过一个16位的寄存器指针，来指定8字节的工作寄存器空间，和空间内的某个8位寄存器，*图 3-2*。



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位寄存器来间接访问其它的寄存器，也可以用任意的16位寄存器组来间接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。

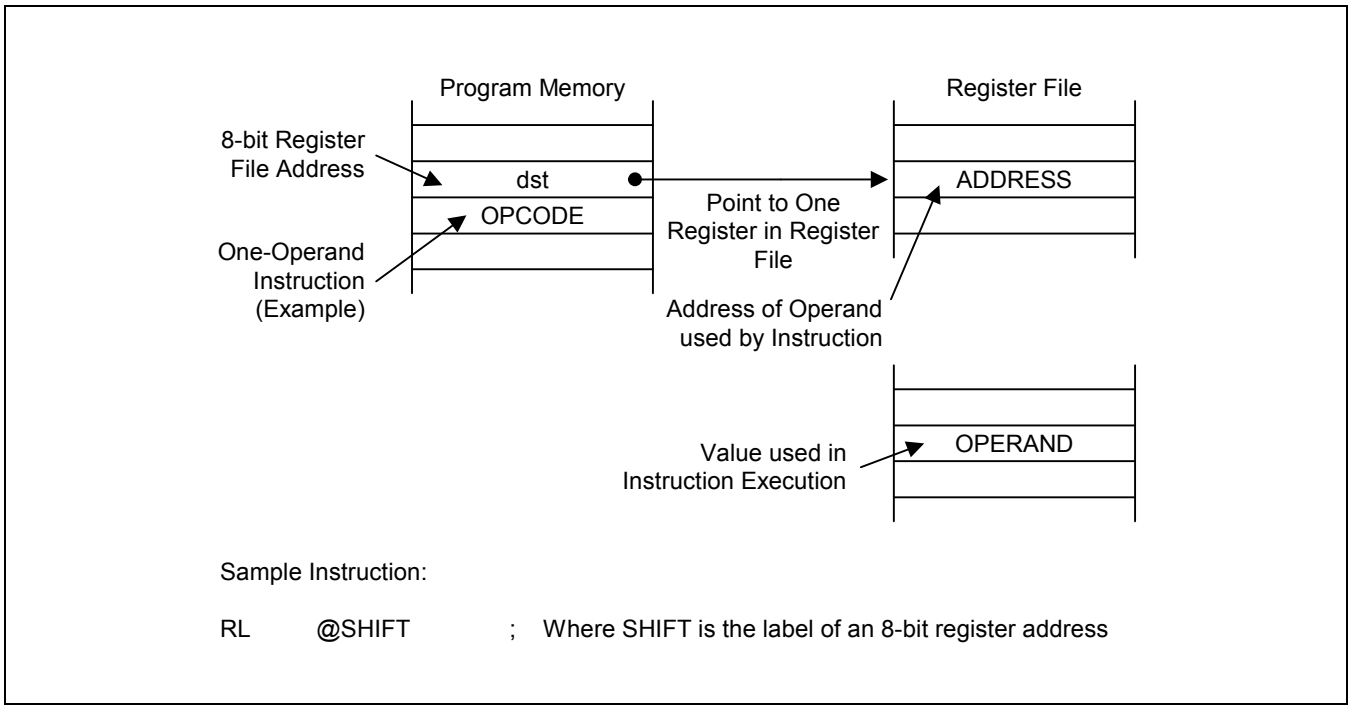


图 3-3 间接寄存器寻址寄存器卷

3.1.2.1 间接寄存器寻址模式 (续)

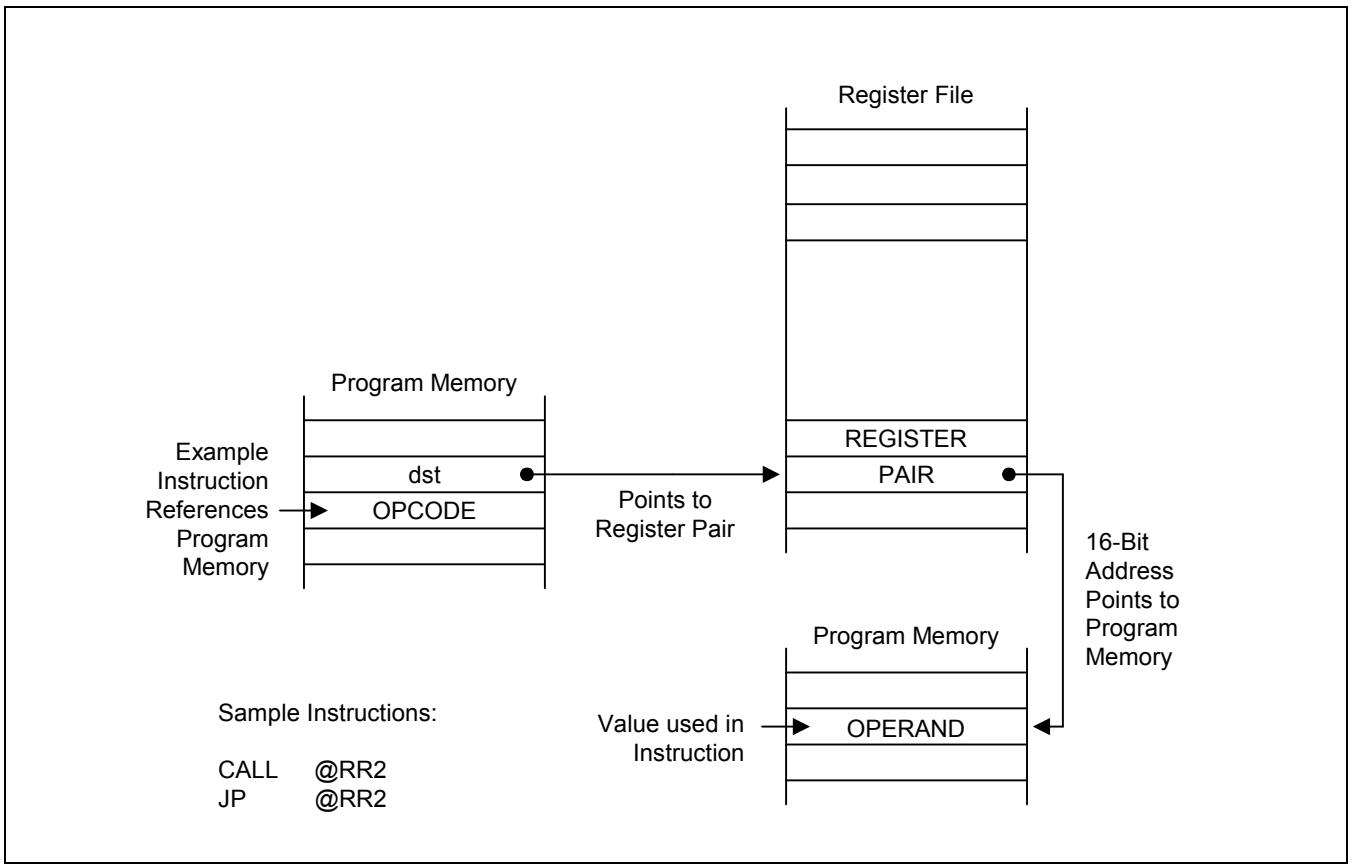


图 3-4 间接寄存器寻址程序存储空间

3.1.2.2 间接寄存器寻址模式 (续)

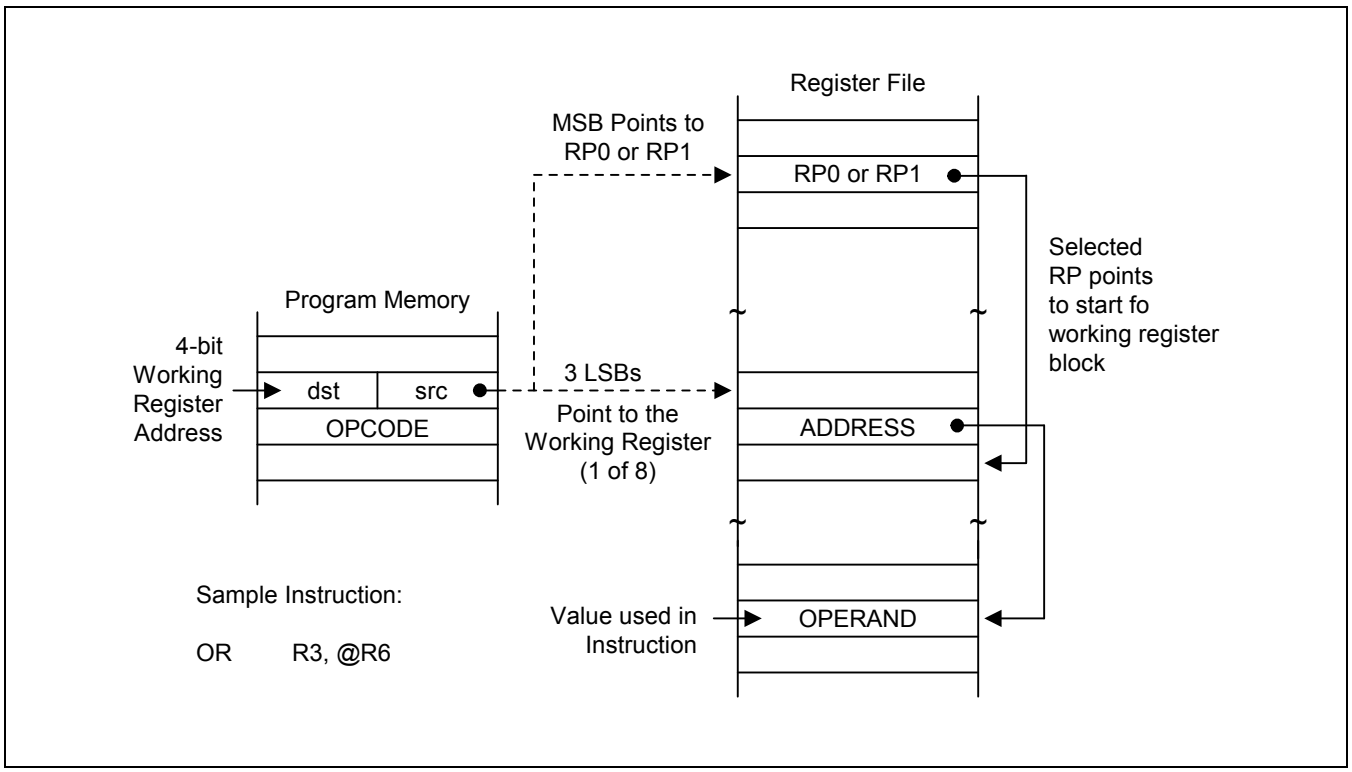


图 3-5 间接工作寄存器寻址寄存器卷

3.1.2.3 间接寄存器寻址模式 (续)

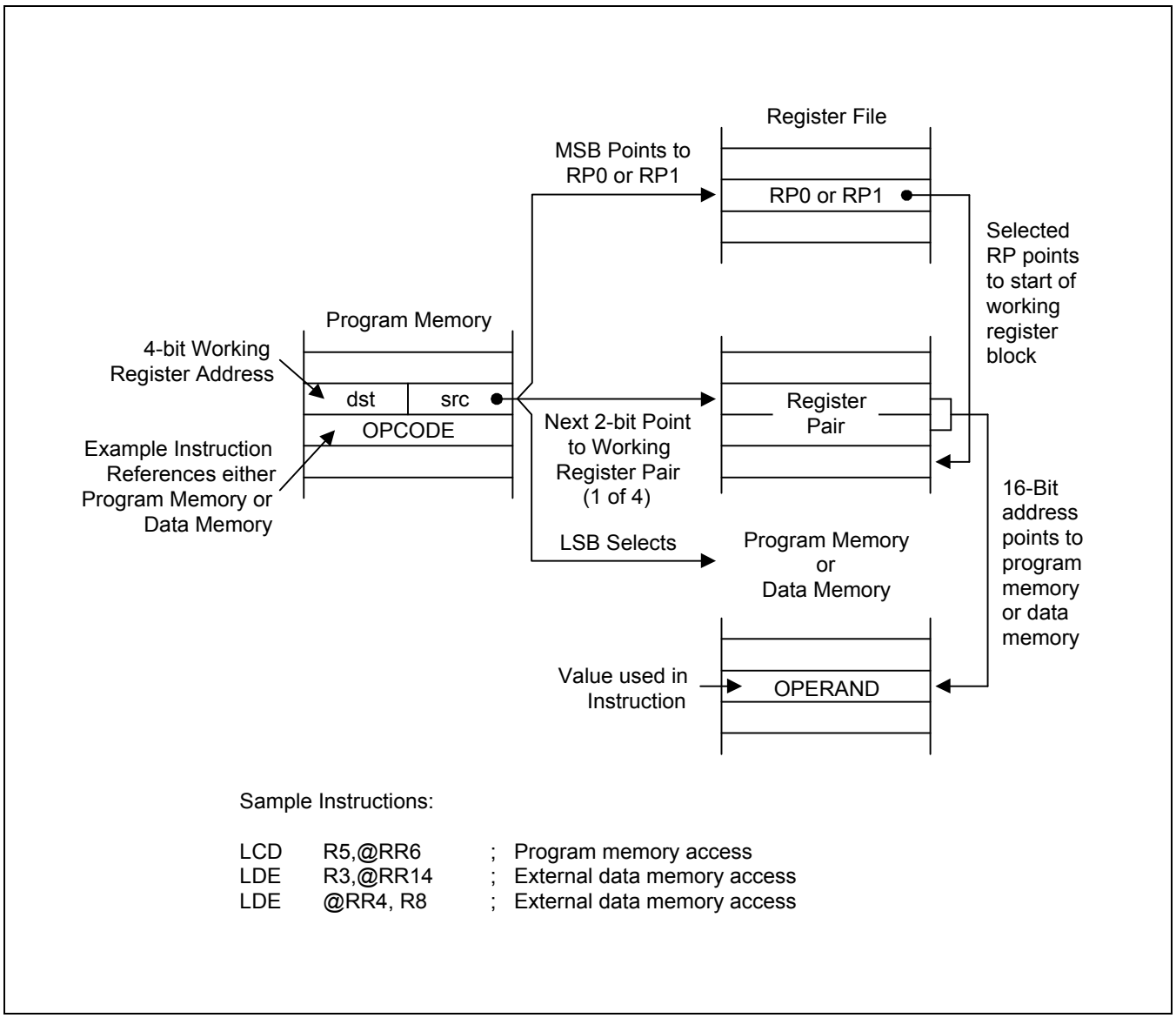


图 3-6 工作寄存器间接寻址程序存储器或数据存储器

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。

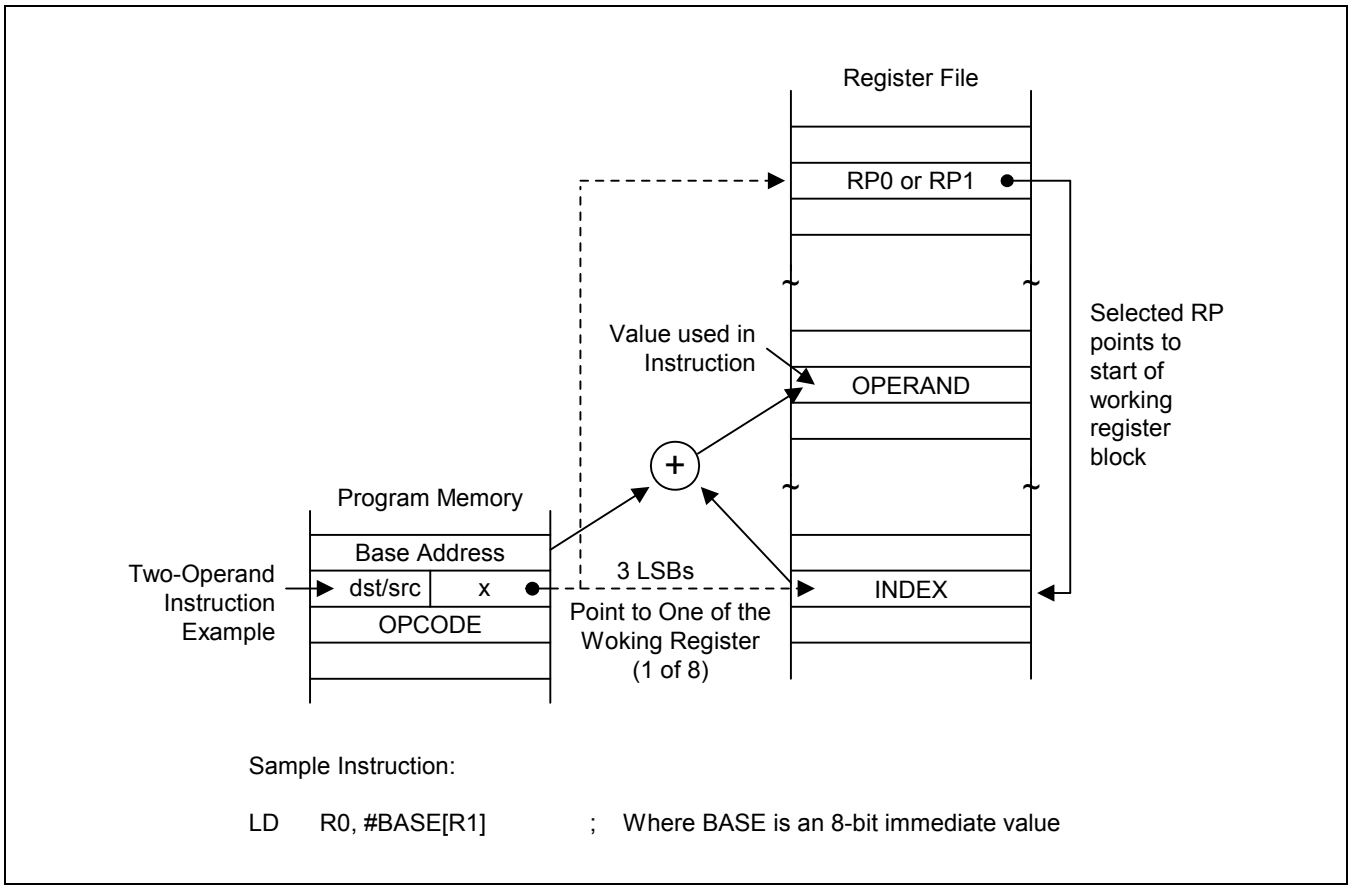


图 3-7 寄存器卷的偏址寻址模式

3.1.3.1 偏址寻址模式 (续)

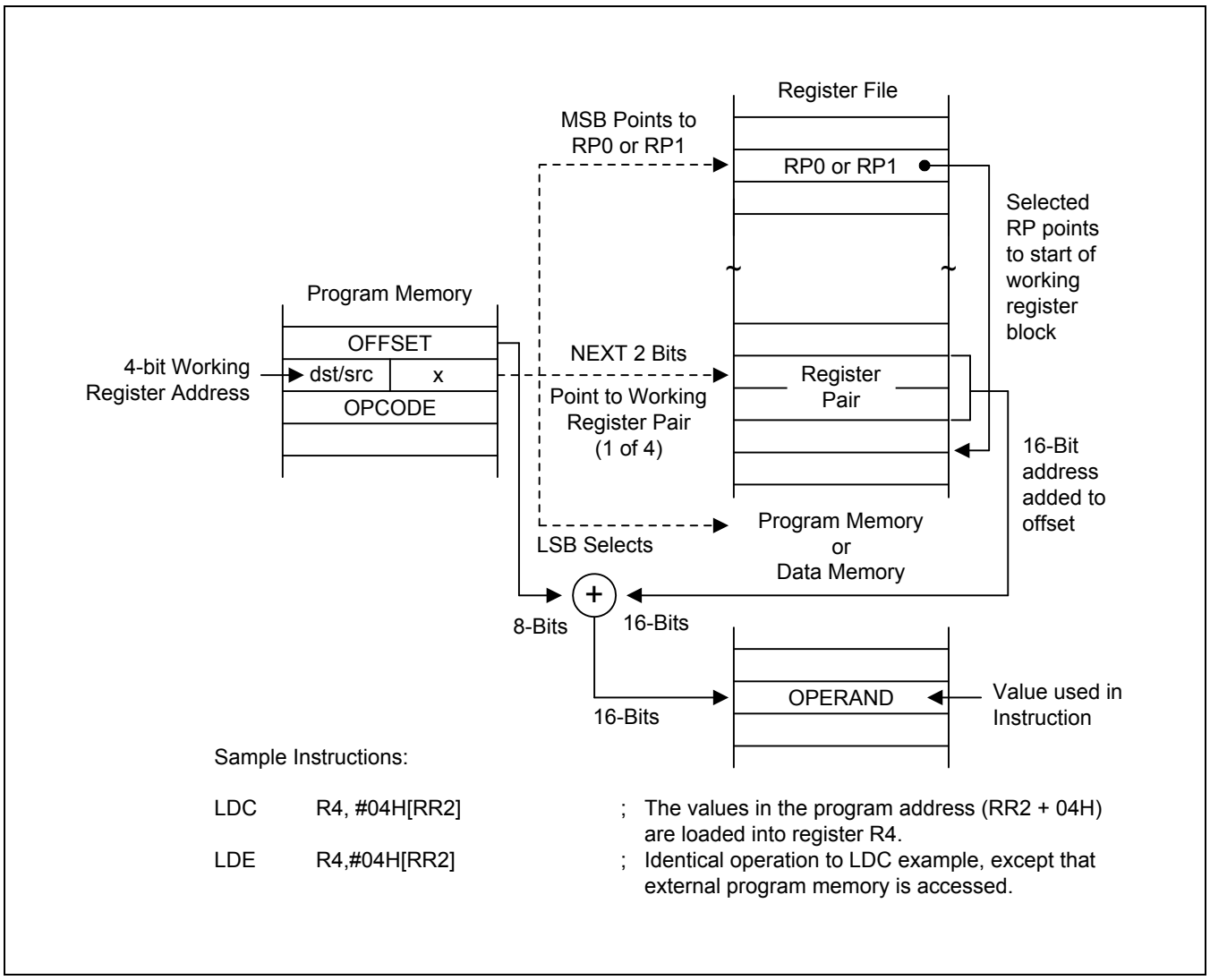
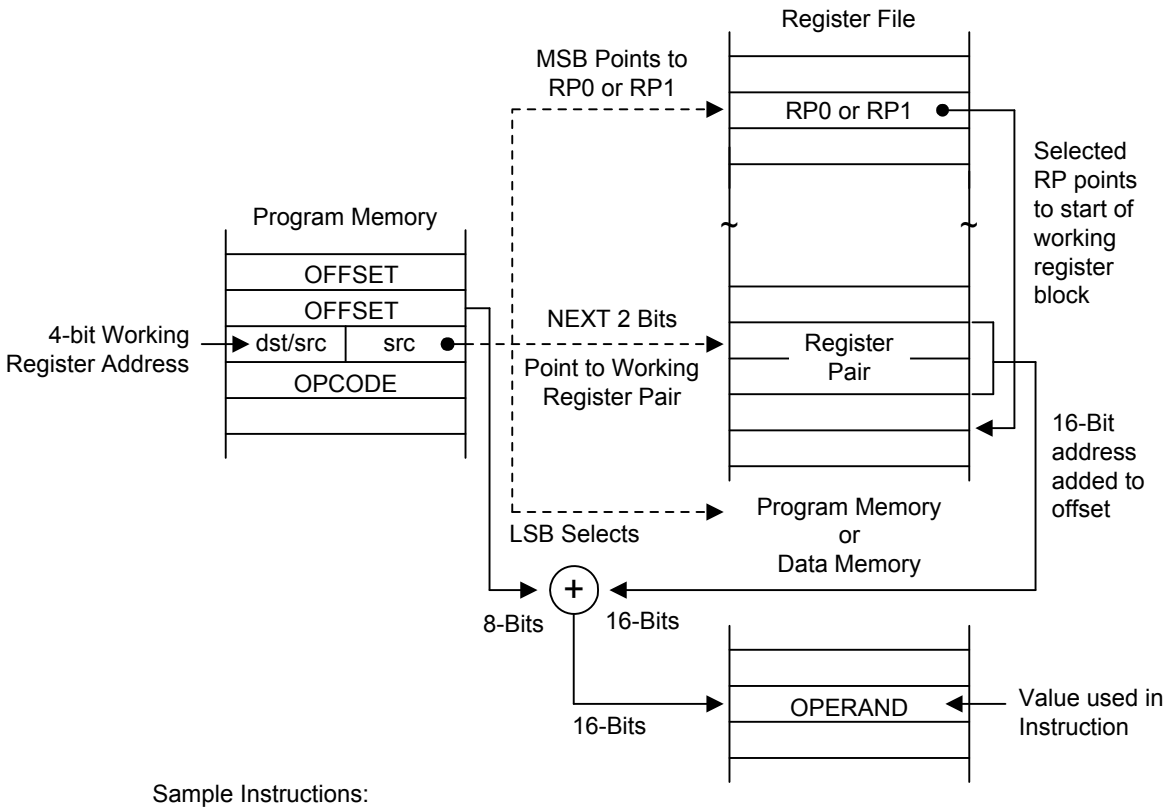


图 3-8 程序或数据存储器的短偏址寻址模式

3.1.3.2 偏址寻址模式 (续)



Sample Instructions:

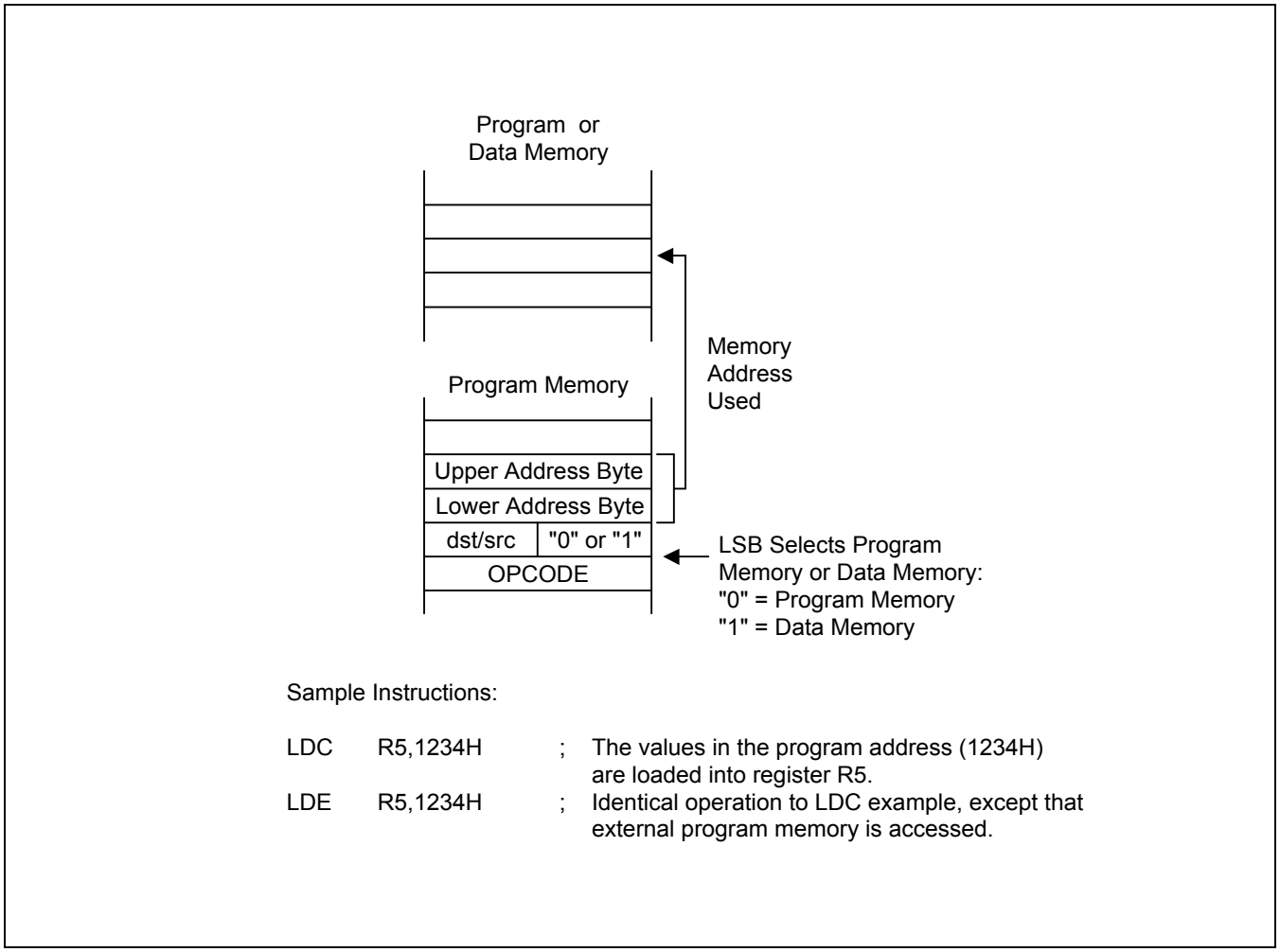
```
LDC     R4, #1000H[RR2]     ; The values in the program address (RR2 + 1000H)
                            ; are loaded into register R4.
LDE     R4,#1000H[RR2]      ; Identical operation to LDC example, except that
                            ; external program memory is accessed.
```

图 3-9 程序或数据存储器的长偏址寻址模式

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。

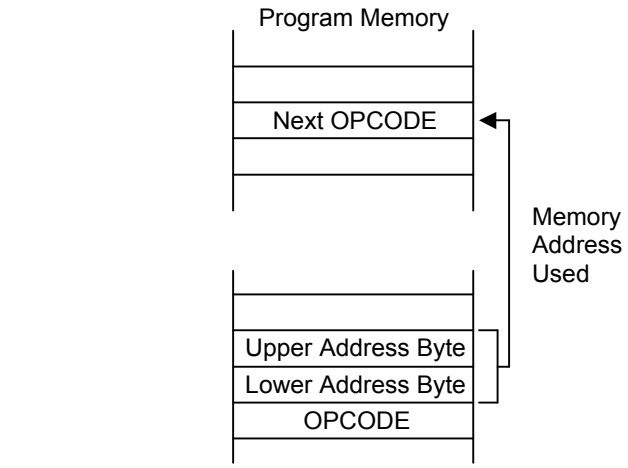


Sample Instructions:

```
LDC     R5,1234H    ;  The values in the program address (1234H)
                       are loaded into register R5.
LDE     R5,1234H    ;  Identical operation to LDC example, except that
                       external program memory is accessed.
```

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 指令的直接寻址

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。

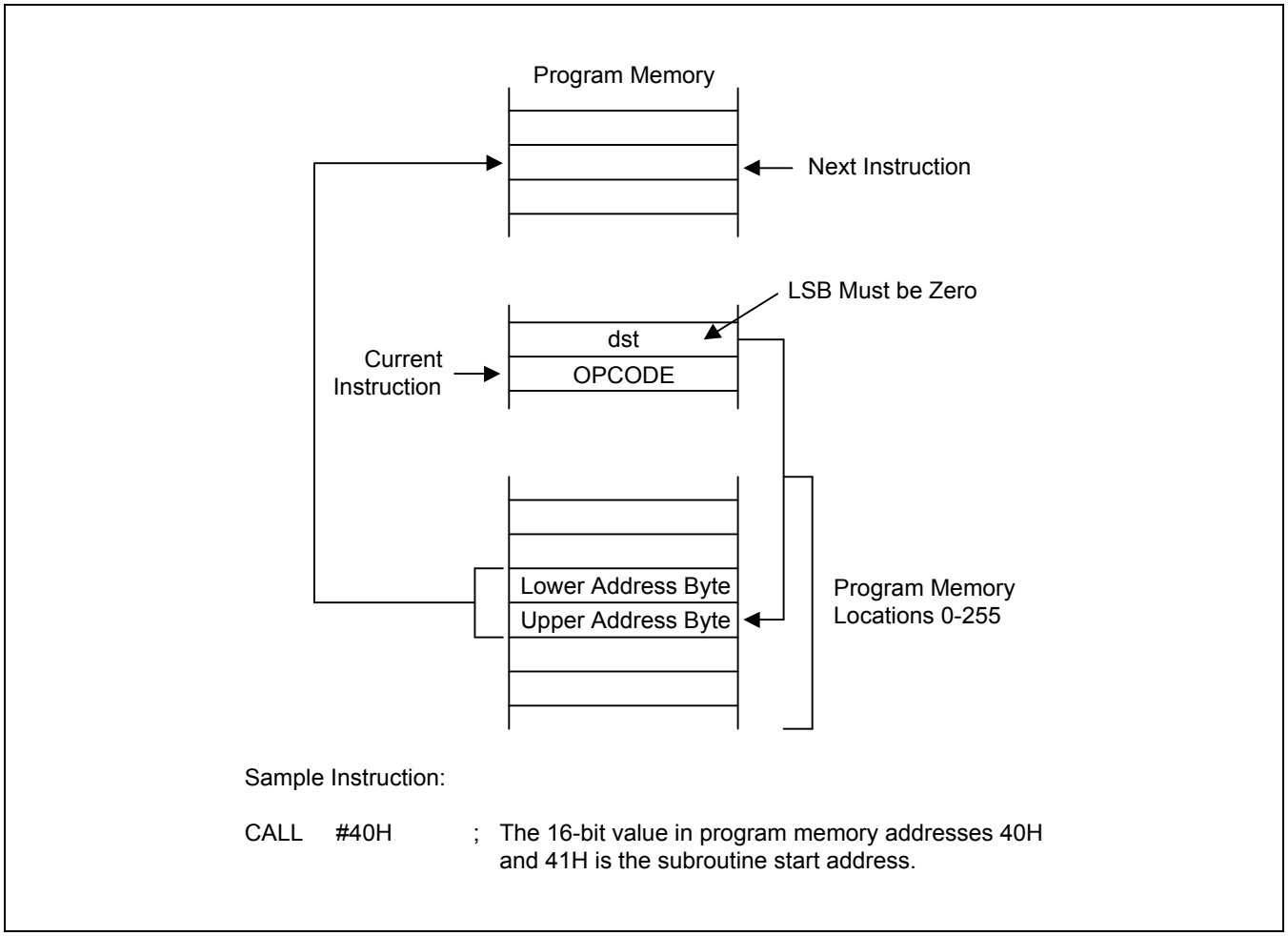


图 3-12 间接寻址模式

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。

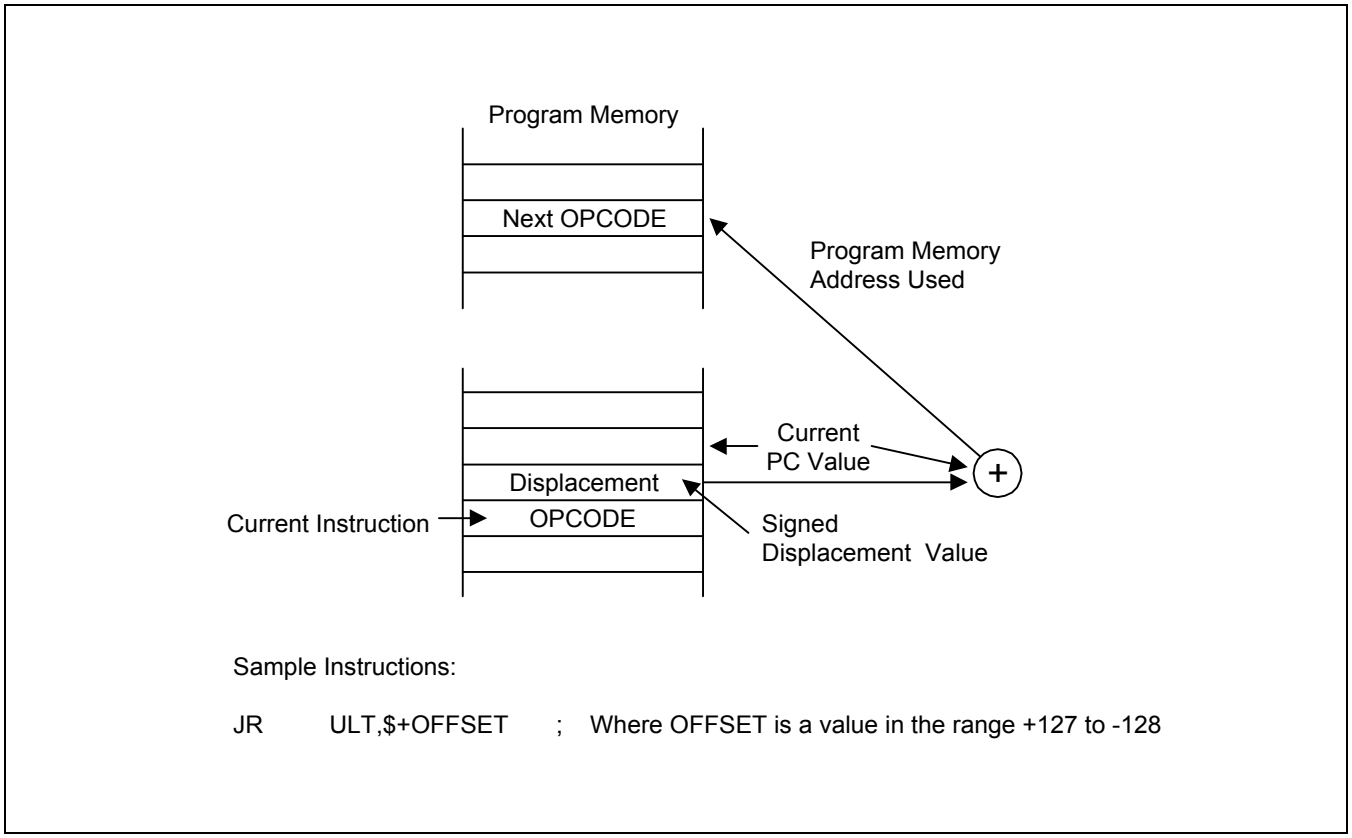


图 3-13 相对地址寻址

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。

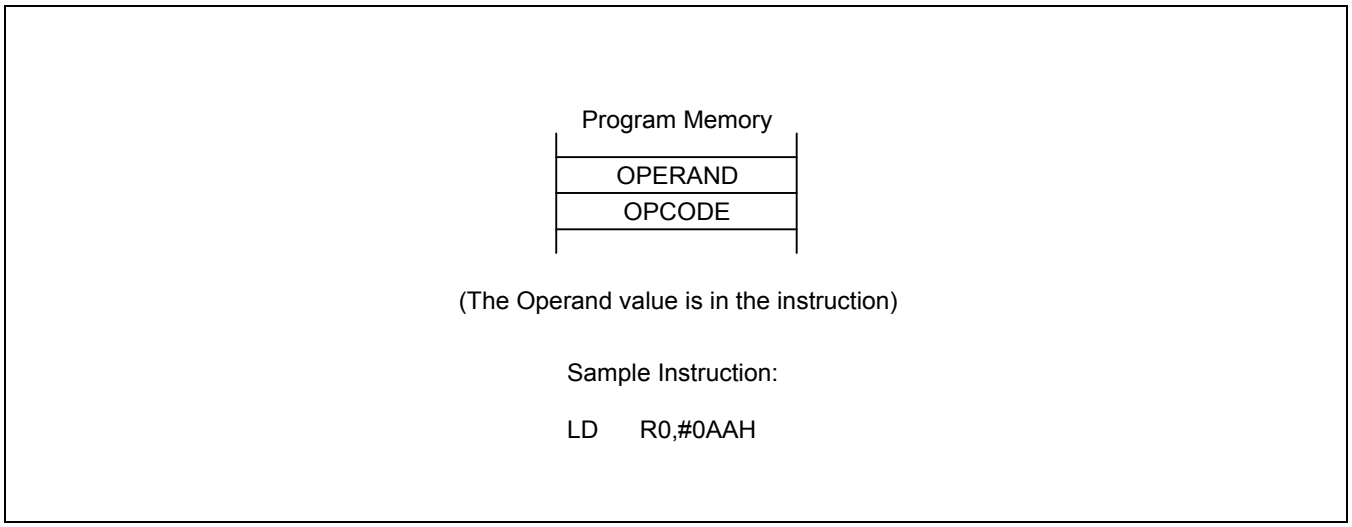


图 3-14 立即数寻址模式

4 控制寄存器

4.1 概述

本章中，S3F82HB 控制寄存器的详细描述以易读方式排列。可以将本章视为编程时的快速参考资源。*表 4-1* 列出了标准寄存器描述的重要参数。

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。

数据和计数寄存器没有在本章中详细列出。关于外设寄存器的更多内容详见本手册第二部分的相关外设章节。

表 4-1 列举了 S3F82HB 所有映射寄存器的位置和读/写特性。
每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	助记标号	地址 (Dec)	地址 (Hex)	R/W
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	助记标号	地址 (Dec)	地址 (Hex)	R/W
中断标志位寄存器	INTPND	208	D0H	R/W
钟表定时器控制寄存器	WTCON	209	D1H	R/W
电池电压检测控制寄存器	BLDCON	210	D2H	R/W
SIO 控制寄存器	SIOCON	224	E0H	R/W
SIO 数据寄存器	SIODATA	225	E1H	R/W
SIO 预处理寄存器	SIOPS	226	E2H	R/W
Timer 0 控制寄存器	T0CON	227	E3H	R/W
Timer 0 计数器 (高字节)	T0CNTH	228	E4H	R
Timer 0 计数器 (低字节)	T0CNTL	229	E5H	R
Timer 0 数据寄存器 (高字节)	T0DATAH	230	E6H	R/W
Timer 0 数据寄存器 (低字节)	T0DATAL	231	E7H	R/W
Timer A 控制寄存器	TACON	232	E8H	R/W
Timer A 计数器	TACNT	233	E9H	R
Timer A 数据寄存器	TADATA	234	EAH	R/W
Timer 1 控制寄存器	T1CON	235	EBH	R/W
Timer 1 控制寄存器 (高字节)	T1CNTH	236	ECH	R
Timer 1 控制寄存器 (低字节)	T1CNTL	237	EDH	R
Timer 1 数据寄存器 (高字节)	T1DATAH	238	EEH	R/W
Timer 1 数据寄存器 (低字节)	T1DATAL	239	EFH	R/W
Timer B 数据寄存器 (高字节)	TBDATAH	240	F0H	R/W
Timer B 数据寄存器 (低字节)	TBDATAL	241	F1H	R/W
Timer B 控制寄存器	TBCON	242	F2H	R/W
频率计数器控制寄存器	FCCON	243	F3H	R/W
频率计数器计数器 (高字节)	FCNTH	244	F4H	R
频率计数器计数器 (低字节)	FCNTL	245	F5H	R
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
振荡器控制寄存器	OSCCON	250	FAH	R/W
STOP 控制寄存器	STPCON	251	FBH	R/W
地址空间 FCH 未映射				
Basic Timer 计数器	BTCNT	253	FDH	R
地址空间 FEH 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	助记标号	地址 (Dec)	地址 (Hex)	R/W
P7 口控制寄存器	P7CON	208	D0H	R/W
P8, 9 口控制寄存器	P89CON	209	D1H	R/W
P10 口控制寄存器	P10CON	210	D2H	R/W
P0 口控制寄存器 (高字节)	P0CONH	224	E0H	R/W
P0 口控制寄存器 (低字节)	P0CONL	225	E1H	R/W
P0 口中断控制寄存器 (高字节)	P0INTH	226	E2H	R/W
P0 口中断控制寄存器 (低字节)	P0INTL	227	E3H	R/W
P0 口中断标志位寄存器	P0PND	228	E4H	R/W
P0 口上拉电阻使能控制寄存器	P0PUR	229	E5H	R/W
P1 口控制寄存器 (高字节)	P1CONH	230	E6H	R/W
P1 口控制寄存器 (低字节)	P1CONL	231	E7H	R/W
P2 口控制寄存器 (高字节)	P2CONH	232	E8H	R/W
P2 口控制寄存器 (低字节)	P2CONL	233	E9H	R/W
P3 口控制寄存器 (高字节)	P3CONH	234	EAH	R/W
P3 口控制寄存器 (低字节)	P3CONL	235	EBH	R/W
P3 口中断控制寄存器	P3INT	236	ECH	R/W
P4 口上拉电阻使能控制寄存器	P4PUR	237	EDH	R/W
P4 口控制寄存器 (高字节)	P4CONH	238	EEH	R/W
P4 口控制寄存器 (低字节)	P4CONL	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
P5 口数据寄存器	P5	245	F5H	R/W
P6 口数据寄存器	P6	246	F6H	R/W
P7 口数据寄存器	P7	247	F7H	R/W
P8 口数据寄存器	P8	248	F8H	R/W
P9 口数据寄存器	P9	249	F9H	R/W
P10 口数据寄存器	P10	250	FAH	R/W
P5 口上拉电阻使能控制寄存器	P5PUR	251	FBH	R/W
P5 口控制寄存器 (高字节)	P5CONH	252	FCH	R/W
P5 口控制寄存器 (低字节)	P5CONL	253	FDH	R/W
P6 口控制寄存器 (高字节)	P6CONH	254	FEH	R/W
P6 口控制寄存器 (低字节)	P6CONL	255	FFH	R/W

表 4-4 Page 0 寄存器

寄存器名	助记标号	地址 (Dec)	地址 (Hex)	R/W
UART 0 控制寄存器 (高字节)	UART0CONH	0	00H	R/W
UART 0 控制寄存器 (低字节)	UART0CONL	1	01H	R/W
UART 0 数据寄存器	UART0DATA	2	02H	R/W
UART 0 波特率数据寄存器	BR0DATA	3	03H	R/W
UART 1 控制寄存器 (高字节)	UART1CONH	4	04H	R/W
UART 1 控制寄存器 (低字节)	UART1CONL	5	05H	R/W
UART 1 数据寄存器	UART1DATA	6	06H	R/W
UART 1 波特率数据寄存器	BR1DATA	7	07H	R/W
比较器 0 控制寄存器	CMP0CON	8	08H	R/W
比较器 1 控制寄存器	CMP1CON	9	09H	R/W
A/D 转换数据寄存器 (高字节)	ADDATAH	10	0AH	R
A/D 转换数据寄存器 (低字节)	ADDATAL	11	0BH	R
A/D 转换控制寄存器	ADCON	12	0CH	R/W
LCD 控制寄存器	LCON	13	0DH	R/W
LCD 对比度控制寄存器	LCONST	14	0EH	R/W
比较器输出控制寄存器	COUTCON	15	0FH	R/W

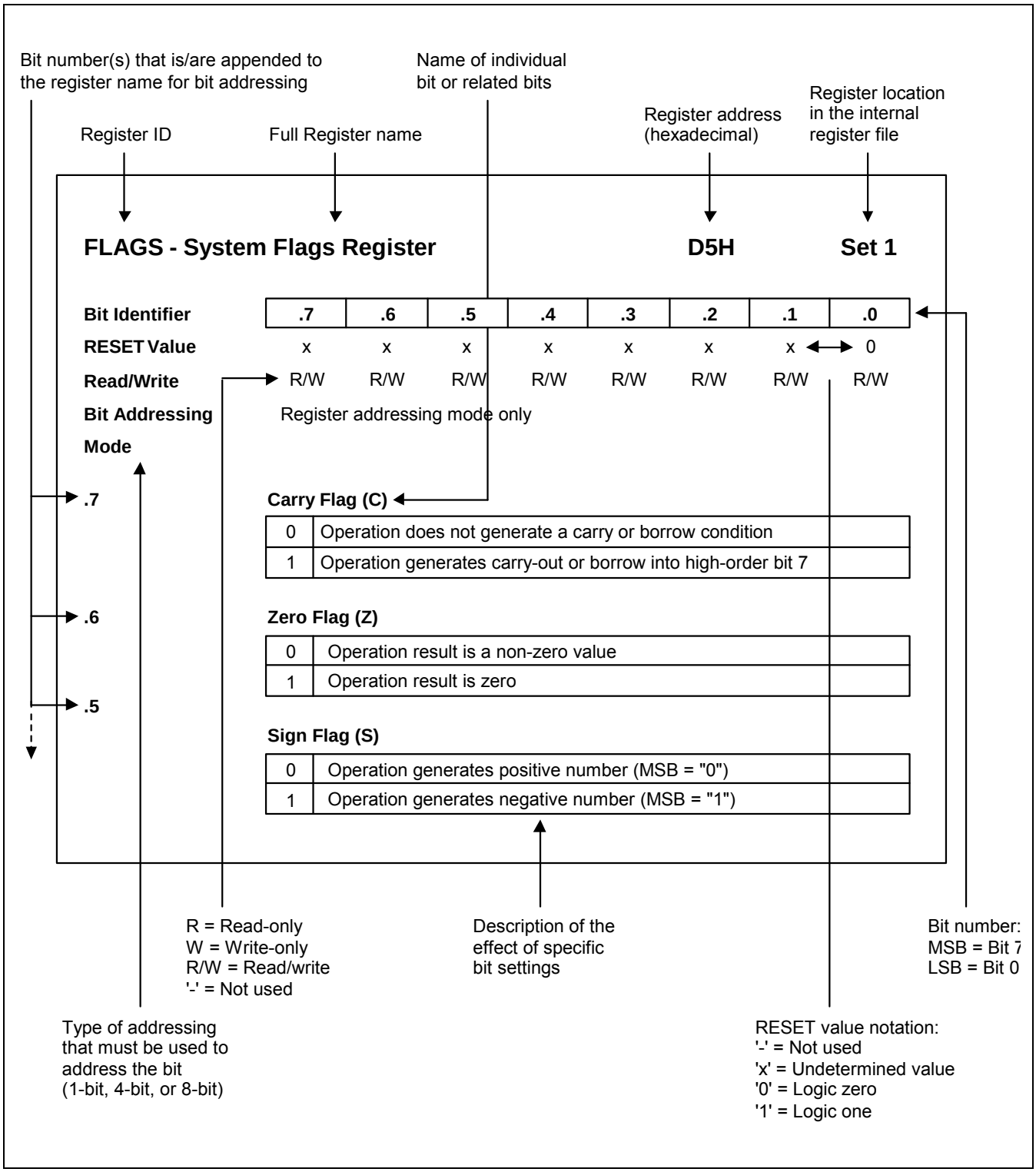

Bit number(s) that is/are appended to the register name for bit addressing
Name of individual bit or related bits
Register location in the internal register file
Register address (hexadecimal)
Register ID
Full Register name
FLAGS - System Flags Register
D5H
Set 1
Bit Identifier
RESET Value
Read/Write
Bit Addressing Mode
.7 .6 .5 .4 .3 .2 .1 .0
x x x x x x x 0
R/W R/W R/W R/W R/W R/W R/W R/W
Register addressing mode only
.7
.6
.5
Carry Flag (C)
0 Operation does not generate a carry or borrow condition
1 Operation generates carry-out or borrow into high-order bit 7
Zero Flag (Z)
0 Operation result is a non-zero value
1 Operation result is zero
Sign Flag (S)
0 Operation generates positive number (MSB = "0")
1 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: 0CH PAGE 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
寻址模式	仅寄存器寻址模式							

.7

S3F82HB 不使用

.6–.4 **A/D 转换输入管脚选择位**

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 (EOC) 状态位 (只读)**

0	转换正在进行
1	转换结束

.2–.1 **A/D 转换时钟源选择位**

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

.0 **A/D 转换启动位**

0	禁止转换
1	启动转换

4.1.2 BLDCON—电池电压检测控制寄存器: D2H SET 1, BANK 0

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

.7–.6

S3F82HB 不使用

.5 **VIN 源选择位**

0	内部源
1	外部源 (当 P2CONH.7-.6 = “11”时，VBLDREF 用作复用功能)

.4 **BLD 输出位 (只读)**

0	$V_{IN} > V_{REF}$ (BLD 使能时)
1	$V_{IN} < V_{REF}$ (BLD 禁止时)

.3 **BLD 使能/禁止位**

0	禁止 BLD
1	使能 BLD

.2–.0 **检测电压选择位**

0	0	0	V_{BLD} = 2.2V
1	0	1	V_{BLD} = 2.4V
0	1	1	V_{BLD} = 2.8V
其他			无

4.1.3 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
寻址模式	仅寄存器寻址模式							

.7–.4 **看门狗时钟功能使能位 (系统复位)**

1	0	1	0	禁止看门狗时钟功能
其他				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 0 控制位(1)**

0	无效
1	清除 Basic Timer 计数器值

.0 **Basic Timer 和 Timer/Counters 分频器清 0(2)**

0	无效
1	2者分频器清 0

注释:

1. 当写"1"到 BTCON.1 时，Basic Timer 计数器被清0，之后该位也自动清0。
2. 当写"1"到 BTCON.0 时，相应的分频器被清0，之后该位也自动清0。
3. fxx 为选定的系统时钟 (主时钟或副时钟)。

4.1.4 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	–	–	–
寻址模式	仅寄存器寻址模式							

.7 **振荡器 IRQ 唤醒功能位**

0	使能省电模式下的 IRQ 唤醒主时钟
1	禁止省电模式下的 IRQ 唤醒主时钟

.6–.5 S3F82HB 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 S3F82HB 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。

4.1.5 CMP0CON—比较器 0 控制寄存器: 08H PAGE 0

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

.7 C0N/AD3 复用功能选择位 (仅当 P2CONL.7-.6 = “11”时)

0	选择 AD3 功能
1	选择 C0N 功能

.6 C0P/AD2 复用功能选择位 (仅当 P2CONL.5-.4 = “11”时)

0	选择 AD2 功能
1	选择 C0P 功能

.5 C0OUT/AD1 复用功能选择位 (仅当 P2CONL.3-.2 = “11”时)

0	选择 AD1 功能
1	选择 C0OUT 功能

.4 比较器 0 输出状态位

0	当 C0P ≤ C0N 时 比较器 0 (C0OUT) 输出低电平
1	当 C0P ＞ C0N 时 比较器 0 (C0OUT) 输出高电平

.3 比较器 0 工作使能位

0	禁止工作 (若 C0OUT 管脚被设置为 比较器 输出，则输出为高阻状态)
1	使能工作

.2–.1 比较器 0 中断使能位

0	0	禁止 比较器 0 中断
0	1	使能 比较器 0 C0OUT 下降沿中断
1	0	使能 比较器 0 C0OUT 上升沿中断
1	1	使能 比较器 0 C0OUT 上升/下降沿中断

.0 比较器 0 中断标志位

0	无中断发生 (读)，标志位清 0 (写)
1	中断发生 (读)

4.1.6 CMP1CON—比较器 1 控制寄存器: 09H PAGE 0

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

.7 C1N/AD4 复用功能选择位 (仅当 P2CONH.1-.0 = “11”时)

0	选择 AD4 功能
1	选择 C1N 功能

.6 C1P/AD5 复用功能选择位 (仅当 P2CONH.3-.2 = “11”时)

0	选择 AD5 功能
1	选择 C1P 功能

.5 C1OUT/AD6 复用功能选择位 (仅当 P2CONH.5-.4 = “11”时)

0	选择 AD6 功能
1	选择 C1OUT 功能

.4 比较器 1 输出状态位

0	当 C1P ≤ C1N 时 比较器 1 (C1OUT) 输出低电平
1	当 C1P > C1N 时 比较器 1 (C1OUT) 输出高电平

.3 比较器 1 工作使能位

0	禁止工作 (若 C1OUT 管脚被设置为比较器输出，则输出为高阻状态)
1	使能工作

.2–.1 比较器 1 中断使能位

0	0	禁止 比较器 1 中断
0	1	使能 比较器 1 C1OUT 下降沿中断
1	0	使能 比较器 1 C1OUT 上升沿中断
1	1	使能 比较器 1 C1OUT 上升/下降沿中断

.0 比较器 1 中断标志位

0	无中断发生 (读)，标志位清 0 (写)
1	中断发生 (读)

4.1.7 COUTCON—比较器输出控制寄存器: 0FH PAGE 0

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

.7–.2

S3F82HB 不使用

.1 **比较器 1 输出 (C1OUT) 控制位**

0	推挽输出
1	N-沟道开漏输出

注释: 此位仅当 P2.6 设置为 C1OUT 功能时有效。

.0 **比较器 0 输出 (C0OUT) 控制位**

0	推挽输出
1	N-沟道开漏输出

注释: 此位仅当 P2.1 设置为 C0OUT 功能时有效。

4.1.8 FCCON—频率计数器控制寄存器: 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
寻址模式	仅寄存器寻址模式							

.7–.6 **频率计数器时钟源选择位**

0	0	禁止计数器计数
0	1	外部时钟源 (FCLK) 下降沿
1	0	C0OUT 下降沿
1	1	C1OUT 下降沿

.5–.3 **门开启时间选择位**

0	0	0	门关闭
0	0	1	门始终开启
0	1	0	门开启时间: $fw/2^{10}$ (31.25mS，fw = 32.768kHz)
0	1	1	门开启时间: $fw/2^{11}$ (62.5mS，fw = 32.768kHz)
1	0	0	门开启时间: $fw/2^{12}$ (125mS，fw = 32.768kHz)
1	0	1	门开启时间: $fw/2^{13}$ (250mS，fw = 32.768kHz)
1	1	0	门开启时间: $fw/2^{14}$ (500mS，fw = 32.768kHz)
1	1	1	门开启时间: $fw/2^{15}$ (1000mS，fw = 32.768kHz)

.2 **16 位递增计数器清0并启动位**

0	无效
1	寄存器清0并启动计数器 (当写入时，清计数器后自动清0)

.1 **频率计数器门周期中断使能位**

0	禁止中断
1	使能中断

.0 **频率计数器溢出中断使能位**

0	禁止中断
1	使能中断

注释: 关于频率计数器的中断标志位，请参考中断标志寄存器 (INTPND)。

4.1.9 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
寻址模式	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	操作产生进位或者借位

.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未产生进位或减法操作时第3位未产生借位

.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

4.1.10 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
寻址模式	仅寄存器寻址模式							

.7–.4 **Flash 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				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

S3F82HB 不使用

.0 **Flash 操作启动位**

0	操作停止
1	操作开始

注释: 相应的操作完成后，FMCON.0 将自动清 0。

4.1.11 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
寻址模式	仅寄存器寻址模式							

.7–.0 **Flash 扇区地址位 (高字节)**

Flash ROM 扇区选择高8位 (第15 ~ 8位)。

注释: 高字节 flash 扇区地址指针指向 16 位指针地址的高 8 位。

4.1.12 FMSECL—FLASH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
寻址模式	仅寄存器寻址模式							

.7 **Flash 扇区地址位 (低字节)**

Flash ROM 扇区选择第 7 位。

.6–.0

S3F82HB 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。

4.1.13 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
寻址模式	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								禁止用户编程模式

4.1.14 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
寻址模式	仅寄存器寻址模式							

.7 **中断优先级7 (IRQ7) 使能位；外部中断 P0.4–0.7, P3.0–3.3**

0	禁止 (屏蔽)
1	使能 (未屏蔽)

.6 **中断优先级6 (IRQ6) 使能位；外部中断 P0.0–0.3**

0	禁止 (屏蔽)
1	使能 (未屏蔽)

.5 **中断优先级5 (IRQ5) 使能位；频率计数器溢出，频率计数器门周期，比较器0/1**

0	禁止 (屏蔽)
1	使能 (未屏蔽)

.4 **中断优先级4 (IRQ4) 使能位；SIO**

0	禁止 (屏蔽)
1	使能 (未屏蔽)

.3 **中断优先级3 (IRQ3) 使能位；UART 0/1 发送, UART 0/1 接收**

0	禁止 (屏蔽)
1	使能 (未屏蔽)

.2 **中断优先级2 (IRQ2) 使能位；Timer A 匹配/捕获, Timer A 溢出，钟表定时器**

0	禁止 (屏蔽)
1	使能 (未屏蔽)

.1 **中断优先级1 (IRQ1) 使能位；Timer B 匹配**

0	禁止 (屏蔽)
1	使能 (未屏蔽)

.0 **中断优先级0 (IRQ0) 使能位；Timer 0 匹配/捕获, Timer 0 溢出, Timer 1 匹配/捕获, Timer 1 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不能被 CPU 识别。

4.1.15 INTPND—中断标志位寄存器: D0H 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
寻址模式	仅寄存器寻址模式							

.7 频率计数器门周期中断标志位

0	没有中断 (读此位时)，中断标志清0 (写此位时)
1	中断标志位 (读此位时)

.6 频率计数器溢出中断标志位

0	没有中断 (读此位时)，中断标志清0 (写此位时)
1	中断标志位 (读此位时)

.5 Timer 1 匹配/捕获中断标志位

0	没有中断 (读此位时)，中断标志清0 (写此位时)
1	中断标志位 (读此位时)

.4 Timer 1 溢出中断标志位

0	没有中断 (读此位时)，中断标志清0 (写此位时)
1	中断标志位 (读此位时)

.3 Timer 0 匹配/捕获中断标志位

0	没有中断 (读此位时)，中断标志清0 (写此位时)
1	中断标志位 (读此位时)

.2 Timer 0 溢出中断标志位

0	没有中断 (读此位时)，中断标志清0 (写此位时)
1	中断标志位 (读此位时)

.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0 (写此位时)
1	中断标志位 (读此位时)

.0 Timer A 溢出中断标志位

0	没有中断 (读此位时)，中断标志清0 (写此位时)
1	中断标志位 (读此位时)

4.1.16 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
寻址模式	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。

4.1.17 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
寻址模式	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。

4.1.18 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
寻址模式	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

4.1.19 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 P0.4–0.7, P3.0–3.3**

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 P0.0–0.3**

0	没有中断
1	标志位置起

.5 **中断级 5 (IRQ5) 请求标志位；频率计数器溢出，频率计数器门周期，比较器0/1**

0	没有中断
1	标志位置起

.4 **中断级 4 (IRQ4) 请求标志位；SIO**

0	没有中断
1	标志位置起

.3 **中断级 3 (IRQ3) 请求标志位；UART 0/1 发送，UART 0/1 接收**

0	没有中断
1	标志位置起

.2 **中断级 2 (IRQ2) 请求标志位；Timer A 匹配/捕捉，Timer A 溢出，钟表定时器**

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 0 匹配/捕捉，Timer 0 溢出，Timer 1 匹配/捕捉，Timer 1 溢出**

0	没有中断
1	标志位置起

4.1.20 LCON—LCD 控制寄存器: 0DH PAGE 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
寻址模式	仅寄存器寻址模式							

.7–.6 **LCD 时钟源选择位**

0	0	fw/2^8 (128Hz)
0	1	fw/2^7 (256Hz)
1	0	fw/2^6 (512Hz)
1	1	fw/2^5 (1024Hz)

.5–.3 **LCD 占空比和偏置选择位(注释)**

0	0	0	1/8 占空比，1/4 偏置
0	0	1	1/4 占空比，1/3 偏置
0	1	0	1/3 占空比，1/3 偏置
0	1	1	1/3 占空比，1/2 偏置
1	x	x	1/2 占空比，1/2 偏置

.2–.1 **LCD 偏置类型选择位**

0	x	电容式偏置
1	0	内部电阻式偏置 (升压器始终停止并断开)
1	1	外部电阻式偏置 (升压器始终停止并断开)

.0 **LCD 显示控制位**

0	所有 LCD 信号低电平 (升压器始终停止并断开)
1	启动显示 (当 LCON.2 = “0”时，启动并连接升压器)

注释:

1. "x" 表示任意值
2. 选择 1/3 偏置时，偏置电压被设置为 V_{LC0} (V_{LC1})，V_{LC2}，V_{LC3} 和 V_{SS}。
3. 选择 1/2 偏置时，偏置电压被设置为 V_{LC0} (V_{LC1}，V_{LC2})，V_{LC3} 和 V_{SS}。

4.1.21 LCONST—LCD 对比度控制寄存器: 0EH PAGE 0

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
寻址模式	仅寄存器寻址模式							

.7 **复用功能选择位 (仅当 P3CONH.3-.2 = “11”时)**

0	选择 SEG57 功能
1	选择 TAOUT/TAPWM 功能

.6–.3

不使用，必须保持‘0’

.2–.0 **VLCD 电压选择位 (仅当选择电容式偏置时)**

值			1/2 偏置 [V]	1/3 偏置 [V]	1/4 偏置 [V]
0	0	0	2.450	2.310	2.410
0	0	1	2.570	2.490	2.530
0	1	0	2.690	2.670	2.645
0	1	1	2.815	2.845	2.765
1	0	0	2.925	3.015	2.875
1	0	1	3.045	3.190	2.995
1	1	0	3.165	3.370	3.115
1	1	1	3.285	3.545	3.230

4.1.22 OSCCON—振荡器控制寄存器: FAH SET 1, BANK 0

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

.7 **副时钟电路选择位**

0	初始状态
1	副振荡器省电模式选择 (当副振荡器停止时自动清“0”)

注释
1. 副振荡器工作时 OSCCON.7 必须保持“1”。
2. 在 VREG 和 GND 之间需要加一个电容 (0.1uF)。

.6–.4

不使用，必须保持‘0’

.3 **主振荡器控制位**

0	主振荡器工作
1	主振荡器停止

.2 **副振荡器控制位**

0	副振荡器工作
1	副振荡器停止

.1

S3F82HB 不使用

.0 **系统时钟选择位**

0	使用主振荡器作系统时钟
1	使用副振荡器作系统时钟

4.1.23 P0CONH—P0 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
寻址模式	仅寄存器寻址模式							

.7–.6 P0.7/INT7/TBPWM 设置位

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (TBPWM)

.5–.4 P0.6/INT6/SO 设置位

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (SO)

.3–.2 P0.5/INT5/SCK 设置位

0	0	输入模式 (SCK 输入)
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (SCK 输出)

.1–.0 P0.4/INT4/SI 设置位

0	0	输入模式 (SI)
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	不使用

4.1.24 P0CONL—P0 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
寻址模式	仅寄存器寻址模式							

.7–.6 **P0.3/INT3/BUZ 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (BUZ)

.5–.4 **P0.2/INT2 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	不使用

.3–.2 **P0.1/INT1 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	不使用

.1–.0 **P0.0/INT0 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	不使用

4.1.25 P0INTH—P0 口中断控制寄存器 (高字节): E2H SET 1, BANK1

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
寻址模式	仅寄存器寻址模式							

.7–.6 **P0.7 外部中断 (INT7) 使能位**

0	0	禁止中断
0	1	下降沿中断使能
1	0	上升沿中断使能
1	1	上升/下降沿中断使能

.5–.4 **P0.6 外部中断 (INT6) 使能位**

0	0	禁止中断
0	1	下降沿中断使能
1	0	上升沿中断使能
1	1	上升/下降沿中断使能

.3–.2 **P0.5 外部中断 (INT5) 使能位**

0	0	禁止中断
0	1	下降沿中断使能
1	0	上升沿中断使能
1	1	上升/下降沿中断使能

.1–.0 **P0.4 外部中断 (INT4) 使能位**

0	0	禁止中断
0	1	下降沿中断使能
1	0	上升沿中断使能
1	1	上升/下降沿中断使能

4.1.26 P0INTL—P0 口中断控制寄存器 (低字节): E3H SET 1, BANK1

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
寻址模式	仅寄存器寻址模式							

.7–.6 **P0.3 外部中断 (INT3) 使能位**

0	0	禁止中断
0	1	下降沿中断使能
1	0	上升沿中断使能
1	1	上升/下降沿中断使能

.5–.4 **P0.2 外部中断 (INT2) 使能位**

0	0	禁止中断
0	1	下降沿中断使能
1	0	上升沿中断使能
1	1	上升/下降沿中断使能

.3–.2 **P0.1 外部中断 (INT1) 使能位**

0	0	禁止中断
0	1	下降沿中断使能
1	0	上升沿中断使能
1	1	上升/下降沿中断使能

.1–.0 **P0.0 外部中断 (INT0) 使能位**

0	0	禁止中断
0	1	下降沿中断使能
1	0	上升沿中断使能
1	1	上升/下降沿中断使能

4.1.27 P0PND—P0 口中断标志位寄存器: E4H SET 1, BANK1

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
寻址模式	仅寄存器寻址模式							

.7 P0.7 外部中断 (INT7) 标志位

0	清除标志位 (写此位时)
1	P0.7 中断请求 (读此位时)

.6 P0.6 外部中断 (INT6) 标志位

0	清除标志位 (写此位时)
1	P0.6 中断请求 (读此位时)

.5 P0.5 外部中断 (INT5) 标志位

0	清除标志位 (写此位时)
1	P0.5 中断请求 (读此位时)

.4 P0.4 外部中断 (INT4) 标志位

0	清除标志位 (写此位时)
1	P0.4 中断请求 (读此位时)

.3 P0.3 外部中断 (INT3) 标志位

0	清除标志位 (写此位时)
1	P0.3 中断请求 (读此位时)

.2 P0.2 外部中断 (INT2) 标志位

0	清除标志位 (写此位时)
1	P0.2 中断请求 (读此位时)

.1 P0.1 外部中断 (INT1) 标志位

0	清除标志位 (写此位时)
1	P0.1 中断请求 (读此位时)

.0 P0.0 外部中断 (INT0) 标志位

0	清除标志位 (写此位时)
1	P0.0 中断请求 (读此位时)

4.1.28 P0PUR—P0 口上拉电阻使能寄存器: E5H SET 1, BANK1

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
寻址模式	仅寄存器寻址模式							

.7 **P0.7 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.6 **P0.6 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.5 **P0.5 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.4 **P0.4 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.3 **P0.3 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.2 **P0.2 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.1 **P0.1 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.0 **P0.0 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

注释: 当对应的管脚选择为推挽输出或复用功能时，P0 口的上拉电阻自动禁止。

4.1.29 P1CONH—P1 口控制寄存器 (高字节): E6H SET 1, BANK1

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/W	R/W	–	R/W	R/W
寻址模式	仅寄存器寻址模式							

.7 **P1.4 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.6 **P1.3 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.5 **P1.2 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.4 **P1.1 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.3 **P1.0 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

S3F82HB 不使用

.1–.0 **P1.4/RxD1 设置位**

0	0	输入模式 (RxD1)
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (RxD1 输出)

注释: 当对应的管脚选择为推挽输出或复用功能时，P1 口的上拉电阻自动禁止。

4.1.30 P1CONL—P1 口控制寄存器 (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
寻址模式	仅寄存器寻址模式							

.7–.6 P1.3/TxD1 设置位

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (TxD1)

.5–.4 P1.2/T0OUT/T0PWM/T0CAP 设置位

0	0	输入模式 (T0CAP)
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (T0OUT/T0PWM)

.3–.2 P1.1/T0CLK/T1CLK 设置位

0	0	输入模式 (T0CLK/T1CLK)
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	无效

.1–.0 P1.0/T1OUT/T1PWM/T1CAP 设置位

0	0	输入模式 (T1CAP)
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (T1OUT/T1PWM)

4.1.31 P2CONH—P2 口控制寄存器 (高字节): 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
寻址模式	仅寄存器寻址模式							

.7–.6 **P2.7/AD7/VBLDREF 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (AD7/VBLDREF)

.5-.4 **P2.6/AD6/C1OUT 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (AD6/C1OUT)

.3–.2 **P2.5/AD5/C1P 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (AD5/C1P)

.1–.0 **P2.4/AD4/C1N 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (AD4/C1N)

注释: P2.7 和 P2.4-P2.6 的复用功能，请相应参考电池电压检测控制寄存器 (BLDCON) 和 比较器 1 控制寄存器 (CMP1CON)。

4.1.32 P2CONL—P2 口控制寄存器 (低字节): 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
寻址模式	仅寄存器寻址模式							

.7–.6 **P2.3/AD3/C0N 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (AD3/C0N)

.5–.4 **P2.2/AD2/C0P 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (AD2/C0P)

.3–.2 **P2.1/AD1/C0OUT 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (AD1/C0OUT)

.1–.0 **P2.0/AD0/FCLK 设置位**

0	0	输入模式 (FCLK)
0	1	带上拉电阻的输入模式 (FCLK)
1	0	推挽输出模式
1	1	复用功能 (AD0)

注释: P2.1-P2.3 的复用功能，请参考 比较器 0 控制寄存器 (CMP0CON)。

4.1.33 P3CONH—P3 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
寻址模式	仅寄存器寻址模式							

.7–.6 **P3.7/R×D0 设置位**

0	0	输入模式 (R×D0)
0	1	带上拉电阻的输入模式 (R×D0)
1	0	推挽输出模式
1	1	复用功能 (R×D0 out)

.5 **P3.6/T×D0 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (T×D0 out)

.3–.2 **P3.5/SEG57/TAOUT/TAPWM/TACAP 设置位**

0	0	输入模式 (TAOUT)
0	1	带上拉电阻的输入模式 (TAOUT)
1	0	推挽输出模式
1	1	复用功能 (TAOUT/TAPWM or SEG57)

.1–.0 **P3.4/SEG56/TACLK 设置位**

0	0	输入模式 (TACLK)
0	1	带上拉电阻的输入模式 (TACLK)
1	0	推挽输出模式
1	1	复用功能 (SEG56)

注释: P3.5 的复用功能，请参考 LCD 控制寄存器 (LCONST)。

4.1.34 P3CONL—P3 口控制寄存器 (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
寻址模式	仅寄存器寻址模式							

.7–.6 **P3.3/SEG55/INT8 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (SEG55)

.5–.4 **P3.2/SEG54/INT9 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (SEG54)

.3–.2 **P3.1/SEG53/INT10 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (SEG53)

.1–.0 **P3.0/SEG52/INT11 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (SEG52)

4.1.35 P3INT—P3 口中断控制寄存器: 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
寻址模式	仅寄存器寻址模式							

.7–.6 **P3.3 外部中断 (INT8) 使能位**

0	0	禁止中断
0	1	下降沿中断使能
1	0	上升沿中断使能
1	1	上升/下降沿中断使能

.5–.4 **P3.2 外部中断 (INT9) 使能位**

0	0	禁止中断
0	1	下降沿中断使能
1	0	上升沿中断使能
1	1	上升/下降沿中断使能

.3–.2 **P3.1 外部中断 (INT10) 使能位**

0	0	禁止中断
0	1	下降沿中断使能
1	0	上升沿中断使能
1	1	上升/下降沿中断使能

.1–.0 **P3.0 外部中断 (INT11) 使能位**

0	0	禁止中断
0	1	下降沿中断使能
1	0	上升沿中断使能
1	1	上升/下降沿中断使能

4.1.36 P4CONH—P4 口控制寄存器 (高字节): EEH 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
寻址模式	仅寄存器寻址模式							

.7–.6 **P4.7/SEG51 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (SEG51)

.5–.4 **P4.6/SEG50 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (SEG50)

.3–.2 **P4.5/SEG49 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (SEG49)

.1–.0 **P4.4/SEG48 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (SEG48)

4.1.37 P4CONL—P4 口控制寄存器 (低字节): EFH 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
寻址模式	仅寄存器寻址模式							

.7–.6 **P4.3/SEG47 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (SEG47)

.5–.4 **P4.2/SEG46 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (SEG46)

.3–.2 **P4.1/SEG45 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (SEG45)

.1–.0 **P4.0/SEG44 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (SEG44)

4.1.38 P4PUR—P4 口上拉电阻使能寄存器: 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
寻址模式	仅寄存器寻址模式							

.7 **P4.7 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.6 **P4.6 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.5 **P4.5 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.4 **P4.4 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.3 **P4.3 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.2 **P4.2 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.1 **P4.1 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

.0 **P4.0 上拉电阻使能位**

0	禁止上拉电阻
1	使能上拉电阻

注释: 当对应的管脚选择为推挽输出或复用功能时，P4 口的上拉电阻自动禁止。

4.1.39 P5CONH—P5 口控制寄存器 (高字节): FCH 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
寻址模式	仅寄存器寻址模式							

.7–.6 **P5.7/SEG43 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (SEG43)

.5–.4 **P5.6/SEG42 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (SEG42)

.3–.2 **P5.5/SEG41 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (SEG41)

.1–.0 **P5.4/SEG40 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (SEG40)

4.1.40 P5CONL—P5 口控制寄存器 (低字节): FDH 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
寻址模式	仅寄存器寻址模式							

.7–.6 **P5.3/SEG39 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (SEG39)

.5–.4 **P5.2/SEG38 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (SEG38)

.3–.2 **P5.1/SEG37 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (SEG37)

.1–.0 **P5.0/SEG36 设置位**

0	0	输入模式
0	1	N-沟道开漏输出模式
1	0	推挽输出模式
1	1	复用功能 (SEG36)

4.1.41 P5PUR—P5 口上拉电阻使能寄存器: 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
寻址模式	仅寄存器寻址模式							

.7 P5.7 上拉电阻使能位

0	禁止上拉电阻
1	使能上拉电阻

.6 P5.6 上拉电阻使能位

0	禁止上拉电阻
1	使能上拉电阻

.5 P5.5 上拉电阻使能位

0	禁止上拉电阻
1	使能上拉电阻

.4 P5.4 上拉电阻使能位

0	禁止上拉电阻
1	使能上拉电阻

.3 P5.3 上拉电阻使能位

0	禁止上拉电阻
1	使能上拉电阻

.2 P5.2 上拉电阻使能位

0	禁止上拉电阻
1	使能上拉电阻

.1 P5.1 上拉电阻使能位

0	禁止上拉电阻
1	使能上拉电阻

.0 P5.0 上拉电阻使能位

0	禁止上拉电阻
1	使能上拉电阻

注释: 当对应的管脚选择为推挽输出或复用功能时，P5 口的上拉电阻自动禁止。

4.1.42 P6CONH—P6 口控制寄存器 (高字节): FEH 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
寻址模式	仅寄存器寻址模式							

.7–.6 **P6.7/SEG35 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (SEG35)

.5–.4 **P6.6/SEG34 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (SEG34)

.3–.2 **P6.5/SEG33 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (SEG33)

.1–.0 **P6.4/SEG32 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (SEG32)

4.1.43 P6CONL—P6 口控制寄存器 (低字节): FFH 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
寻址模式	仅寄存器寻址模式							

.7–.6 **P6.3/SEG31 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (SEG31)

.5–.4 **P6.2/SEG30 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (SEG30)

.3–.2 **P6.1/SEG29 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (SEG29)

.1–.0 **P6.0/SEG28 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (SEG28)

4.1.44 P7CON—P7 口控制寄存器: 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
寻址模式	仅寄存器寻址模式							

.7–.6 **P7.7–P7.6/SEG27–SEG26 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (SEG27-SEG26)

.5–.4 **P7.5–P7.4/SEG25–SEG24 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (SEG25-SEG24)

.3–.2 **P7.3–P7.2/SEG23–SEG22 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (SEG23–SEG22)

.1–.0 **P7.1–P7.0/SEG21–SEG20 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (SEG21–SEG20)

4.1.45 P89CON—P8, 9 口控制寄存器: 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
寻址模式	仅寄存器寻址模式							

.7–.6 **P9.5–P9.0/SEG11–SEG6 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (SEG11–SEG6)

.5–.4 **P8.7–P8.6/SEG19–SEG18 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (SEG19–SEG18)

.3–.2 **P8.5–P8.4/SEG17–SEG16 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (SEG17–SEG16)

.1–.0 **P8.3–P8.0/SEG15–SEG12 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (SEG15–SEG12)

4.1.46 P10CON—P10 口控制寄存器: 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
寻址模式	仅寄存器寻址模式							

.7–.6 **P10.7–P10.4/COM7–COM4/SEG5-SEG2 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (COM7–COM4/SEG5–SEG2)

注释: 当LCD控制器占空比为1/8时作COM7-COM4，其他作 SEG5-SEG2

.5–.4 **P10.3/COM3/SEG1 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (COM3/SEG1)

注释: 当LCD控制器占空比为1/8或1/4时作 COM3，其他作 SEG1

.3–.2 **P10.2/COM2/SEG0 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (COM2/SEG0)

注释: 当LCD控制器占空比为 1/8，1/4或1/3 时作COM2，其他作 SEG0

.1–.0 **P10.1–P10.0/COM1–COM0 设置位**

0	0	输入模式
0	1	带上拉电阻的输入模式
1	0	推挽输出模式
1	1	复用功能 (COM1–COM0)

4.1.47 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
寻址模式	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
0	0	1	0	目的寄存器：第 2 页
0	0	1	1	目的寄存器：第 3 页
0	1	0	0	目的寄存器：第 4 页
0	1	0	1	目的寄存器：第 5 页
0	1	1	0	目的寄存器：第 6 页
0	1	1	1	目的寄存器：第 7 页
1	0	0	0	目的寄存器：第 8 页
1	0	0	1	目的寄存器：第 9 页
1	1	1	1	目的寄存器：第 15 页
其他				S3F82HB 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
0	0	1	0	源寄存器：第 2 页
0	0	1	1	源寄存器：第 3 页
0	1	0	0	源寄存器：第 4 页
0	1	0	1	源寄存器：第 5 页
0	1	1	0	源寄存器：第 6 页
0	1	1	1	源寄存器：第 7 页
1	0	0	0	源寄存器：第 8 页
1	0	0	1	源寄存器：第 9 页
1	1	1	1	源寄存器：第 15 页
其他				S3F82HB 不使用

注释:

1. S3F82HB 内部寄存器卷分为11页 (0-9,15页)。第0-9页作为通用寄存器卷使用。
2. S3F82HB 的第15页可用作 LCD 数据寄存器或通用寄存器卷。

4.1.48 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	–	–	–
寻址模式	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 S3F82HB 不使用

4.1.49 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	–	–	–
寻址模式	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 S3F82HB 不使用

4.1.50 SIOCON—SIO 控制寄存器: E0H 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
寻址模式	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 优先模式
1	LSB 优先模式

.5 SIO 模式选择位

0	只接收模式
1	发送/接收模式

.4 移位时钟边沿选择位

0	下降沿 Tx，上升沿 Rx
1	上升沿 Tx，下降沿 Rx

.3 SIO 计数器清0和移位启动位

0	无效
1	3位计数器清0，并启动移位

.2 SIO 移位操作使能位

0	禁止移位和时钟计数
1	使能移位和时钟计数

.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	无中断标志 (读此位时)，中断状态清除 (写此位时)
1	中断发生

4.1.51 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
寻址模式	仅寄存器寻址模式							

.7–.0 **堆栈指针地址 (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 (D9H) 中。复位后，堆栈指针 (SP) 的值不确定。

4.1.52 SPL—堆栈指针 (低字节): 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
寻址模式	仅寄存器寻址模式							

.7–.0 **堆栈指针地址 (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 (D8H) 中。复位后，堆栈指针 (SP) 的值不确定。

4.1.53 STPCON—STOP 控制寄存器: FBH SET 1, BANK 0

位	.7	.6	.5	.4	.3	.2	.1	.0
复位值	0	0	0	0	0	0	0	0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
寻址模式	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	禁止使用 STOP 指令

注释: 在执行 STOP 指令前，需要将 STPCON 寄存器设置为“10100101B”，否则，STOP 指令不会被执行。

4.1.54 SYM—系统模式寄存器: DEH SET 1

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

.7

始终保持逻辑“0”

.6–.5

S3F82HB 不使用

.4–.2 **快速中断优先级选择位**(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
1	使能快速中断

.0 **全局中断使能位**(3)

0	禁止所有中断
1	使能所有中断

注释:

1. 一次只能选择一个中断优先级作为快速中断。
2. 将 SYM.1 设为“1”使能 SYM.2-SYM.4 选择的快速中断。
3. 复位后，必须通过执行 EI 指令使能全局中断 (而不是往 SYM.0 写“1”)。

4.1.55 T0CON—TIMER 0 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
寻址模式	仅寄存器寻址模式							

.7–.5 Timer 0 输入时钟选择位

0	0	0	TBOF
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 (T0CLK) 下降沿
1	1	0	外部时钟 (T0CLK) 上升沿
1	1	1	停止计数器

.4–.3 Timer 0 工作模式选择位

0	0	定时 (Interval) 模式 (T0OUT)
0	1	捕获 (Capture) 模式 (上升沿捕获，计数器工作，溢出中断可发生)
1	0	捕获 (Capture) 模式 (下降沿捕获，计数器工作，溢出中断可发生)
1	1	PWM 模式 (溢出和匹配中断可发生)

.2 Timer 0 计数器使能位

0	无效
1	timer 0 计数器清0 (写此位时，计数器清0后此位自动清0)

.1 Timer 0 匹配 (Match)/捕获 (Capture) 中断使能位

0	禁止中断
1	使能中断

.0 Timer 0 溢出 (Overflow) 中断使能位

0	禁止溢出 (overflow) 中断
1	使能溢出 (overflow) 中断

注释: timer 0 的中断标志位请参考中断标志位寄存器 (INTPND)。

4.1.56 T1CON—TIMER 1 控制寄存器: EBH 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
寻址模式	仅寄存器寻址模式							

.7–.5 **Timer 1 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 (T1CLK) 下降沿
1	1	0	外部时钟 (T1CLK) 上升沿
1	1	1	停止计数器

.4–.3 **Timer 1 工作模式选择位**

0	0	定时 (Interval) 模式 (T1OUT)
0	1	捕获 (Capture) 模式 (上升沿捕获，计数器工作，OVF 中断可发生)
1	0	捕获 (Capture) 模式 (下降沿捕获，计数器工作，OVF 中断可发生)
1	1	PWM 模式 (OVF 和匹配中断可发生)

.2 **Timer 1 计数器使能位**

0	无效
1	timer 1 计数器清0 (写此位时，计数器清0后此位自动清0)

.1 **Timer 1 匹配 (Match)/捕获 (Capture) 中断使能位**

0	禁止中断
1	使能中断

.0 **Timer 1 溢出 (Overflow) 中断使能位**

0	禁止溢出 (overflow) 中断
1	使能溢出 (overflow) 中断

注释: timer 1 的中断标志位请参考中断标志位寄存器 (INTPND)。

4.1.57 TACON—TIMER A 控制寄存器: E8H 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
寻址模式	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 (系统时钟)
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	停止计数器

.4–.3 **Timer A 工作模式选择位**

0	0	定时 (Interval) 模式 (TAOUT)
0	1	捕获 (Capture) 模式 (上升沿捕获，计数器工作，OVF 中断可发生)
1	0	捕获 (Capture) 模式 (下降沿捕获，计数器工作，OVF 中断可发生)
1	1	PWM 模式 (OVF 和匹配中断可发生)

.2 **Timer A 计数器使能位**

0	无效
1	timer A 计数器清0 (写此位时，计数器清0后此位自动清0)

.1 **Timer A 匹配 (Match)/捕获 (Capture) 中断使能位**

0	禁止中断
1	使能中断

.0 **Timer A 溢出 (Overflow) 中断使能位**

0	禁止溢出 (overflow) 中断
1	使能溢出 (overflow) 中断

注释: timer A 的中断标志位请参考中断标志位寄存器 (INTPND)。

4.1.58 TBCON—TIMER B 控制寄存器: 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
寻址模式	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 中断时间选择位**

0	0	低字节数据借位产生后生成
0	1	高字节数据借位产生后生成
1	0	高低字节数据借位产生后生成
1	1	无效

.3 **Timer B 中断使能位**

0	禁止中断
1	使能中断

.2 **Timer B 启动/停止为**

0	停止 timer B
1	启动 timer B

.1 **Timer B 模式选择位**

0	单次模式
1	循环模式

.0 **Timer B 输出 flip-flop 控制位**

0	TBOF 低 (TBPWM：低字节数据低电平，高字节数据高电平)
1	TBOF 高 (TBPWM：低字节数据高电平，高字节数据低电平)

4.1.59 UART0CONH—UART 0 控制寄存器 (高字节): 00H PAGE 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
寻址模式	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×(BRDATA+1)))
0	1	模式 1：8位 UART (fU/(16×(BRDATA+1)))
1	0	模式 2：9位 UART (fU/16)
1	1	模式 3：9位 UART (fU/(16×(BRDATA+1)))

.5 多处理器通讯使能位 (仅供模式 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)

UART 0 的模式2 或模式3 中待发送数据的第9 位位置 (“0”或“1”)

注释: 若 UART0CONL.7 = 1，则此位无效

.2 RB8 (仅当 UART0CONL.7 = 0)

UART 0 的模式2 或模式3 中接受到数据的第9 位位置 (“0”或“1”)

注释: 若 UART0CONL.7 = 1，则此位无效

.1 Uart 0 接收中断使能位

0	禁止 Rx 中断
1	使能 Rx 中断

.0 Uart 0 接收中断标志位

0	无中断标志 (写此位时)，标志位清0 (写此位时)
1	中断请求 (写此位时)

注释:

1. 模式2和模式3中，若 MCE 位设为“1”，则当接收到的第9位数据为“0”时接收中断不被激活。
   模式1中，若 MCE =“1”，则接收到有效的停止位后接收中断才被激活。模式0中，MCE 应设为“0”。
2. 8位和9位 UART 模式的说法中不包括串行发送/接收的起始位和停止位。

4.1.60 UART0CONL—UART 0 控制寄存器 (低字节): 01H PAGE 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
寻址模式	仅寄存器寻址模式							

.7 UART 0 发送奇偶校验位自动生成使能位

0	禁止奇偶校验自动生成
1	使能奇偶校验自动生成

.6 UART 0 发送奇偶校验选择位 (仅供模式2和模式3)

0	偶校验
1	奇校验

注释: 若 UART0CONL.7 = 0，此位无效。

.5 UART 0 接收奇偶校验位自动生成使能位 (仅供模式2和模式3)

0	偶校验
1	奇校验

注释: 若 UART0CONL.7 = 0，此位无效。

.4 UART 0 接收机偶校验错误状态位 (仅供模式2和模式3)

0	无奇偶校验错误
1	奇偶校验错误

注释: 若 UART0CONL.7 = 0，此位无效。

.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 Tx 中断
1	使能 Tx 中断

.0 Uart 0 发送中断标志位

0	无中断标志 (写此位时)，标志位清0 (写此位时)
1	中断请求 (写此位时)

4.1.61 UART1CONH—UART 1 控制寄存器 (高字节): 04H PAGE 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
寻址模式	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×(BRDATA+1)))
0	1 模式 1: 8-bit UART (fU/(16×(BRDATA+1)))
1	0 模式 2: 9-bit UART (fU/16)
1	1 模式 3: 9-bit UART (fU/(16×(BRDATA+1)))

.5 多处理器通讯使能位 (仅供模式 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)

UART 1 的模式2 或模式3 中待发送数据的第9 位位置

注释: 若 UART1CONL.7 = 1，则此位无效

.2 RB8 (仅当 UART1CONL.7 = 0)

UART 1 的模式2 或模式3 中接受到数据的第9 位位置

注释: 若 UART1CONL.7 = 1，则此位无效

.1 Uart 1 接收中断使能位

0	禁止 Rx 中断
1	使能 Rx 中断

.0 Uart 1接收中断标志位

0	无中断标志 (写此位时)，标志位清0 (写此位时)
1	中断请求 (写此位时)

注释:

1. 模式2和模式3中，若 MCE 位设为“1”，则当接收到的第9位数据为“0”时接收中断不被激活。
   模式1中，若 MCE = “1”，则接收到有效的停止位后接收中断才被激活。模式0中，MCE 应设为“0”。
2. 8位和9位 UART 模式的说法中不包括串行发送/接收的起始位和停止位。

4.1.62 UART1CONL—UART 1 控制寄存器 (低字节): 05H PAGE 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
寻址模式	仅寄存器寻址模式							

.7 UART 1 发送奇偶校验位自动生成使能位

0	禁止奇偶校验自动生成
1	使能奇偶校验自动生成

.6 UART 1 发送奇偶校验选择位 (仅供模式2和模式3)

0	偶校验
1	奇校验

注释: 若 UART1CONL.7 = 0，此位无效。

.5 UART 1 接收校验选择位 (仅供模式2和模式3)

0	偶校验
1	奇校验

注释: 若 UART1CONL.7 = 0，此位无效。

.4 UART 1 接收机偶校验错误状态位 (仅供模式2和模式3)

0	无奇偶校验错误
1	奇偶校验错误

注释: 若 UART1CONL.7 = 0，此位无效。

.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 Tx 中断
1	使能 Tx 中断

.0 Uart 1 发送中断标志位

0	无中断标志 (写此位时)，标志位清0 (写此位时)
1	中断请求 (写此位时)

4.1.63 WTCON—钟表定时器 控制寄存器: D1H 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
寻址模式	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 27 分频 (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 速度选择位**

0	0	钟表定时器 中断时间 1.0s
0	1	钟表定时器 中断时间 0.5s
1	0	钟表定时器 中断时间 0.25s
1	1	钟表定时器 中断时间 3.91ms

.1 **钟表定时器 使能位**

0	禁止 钟表定时器，分频电路清0
1	使能 钟表定时器

.0 **钟表定时器 中断标志位**

0	无中断标志 (写此位时)，标志位清0 (写此位时)
1	中断请求 (写此位时)

注释: 假设 钟表定时器 的时钟频率 (fw) 为 32.768 kHz。

5 中断结构

5.1 概述

S3F8- 系列的中断结构由三个基本部分组成：中断级，中断向量和中断源。
SAM8 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硬件决定。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)

中断级是优先级分配和识别的主要单元。所有的外设和 I/O 模块都可以发出中断请求。换句话说，外设和 I/O 模块的操作都是中断驱动的 (interrupt-driven)。一共有8个可能的中断级：IRQ0–IRQ7，也被称为优先级0 ~ 优先级 7。每个中断级直接与中断请求号 (IRQn) 关联。每款芯片的中断级数量不同。S3F82HB 的中断结构中就有8个中断级。

中断级序号0~7仅仅是一个标号，并不直接表示优先级。相对优先级由中断优先级寄存器 IPR 中的设置决定。IPR中，中断组和子分组结构更细化了各个中断级之间的优先级定义。

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地址。每个中断级最多支持128个中断向量(但 S3F8- 系列产品中实际用到的向量个数往往远小于128)。当一个中断级有多个中断地址时，优先级由硬件决定。S3F82HB 里有29个中断向量。

5.1.3 中断源 (SOURCES)

中断源可以是任何产生中断的外设。可以是外部管脚或者计数器溢出。多个中断源可以共用一个中断向量。S3F82HB 的中断结构中有29个潜在的中断源。

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 S3F8- 系列中断结构的三个组成部分 — 中断级，中断向量和中断源 — 充分使用了芯片的中断逻辑，一起决定了芯片的中断结构。中断结构一共有三种可能的组合，分别称为 Type1，2，3。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 ($S_1 – S_n$, $S_{n+1} – S_{n+m}$)

S3F82HB 只使用了上述两种 Type。

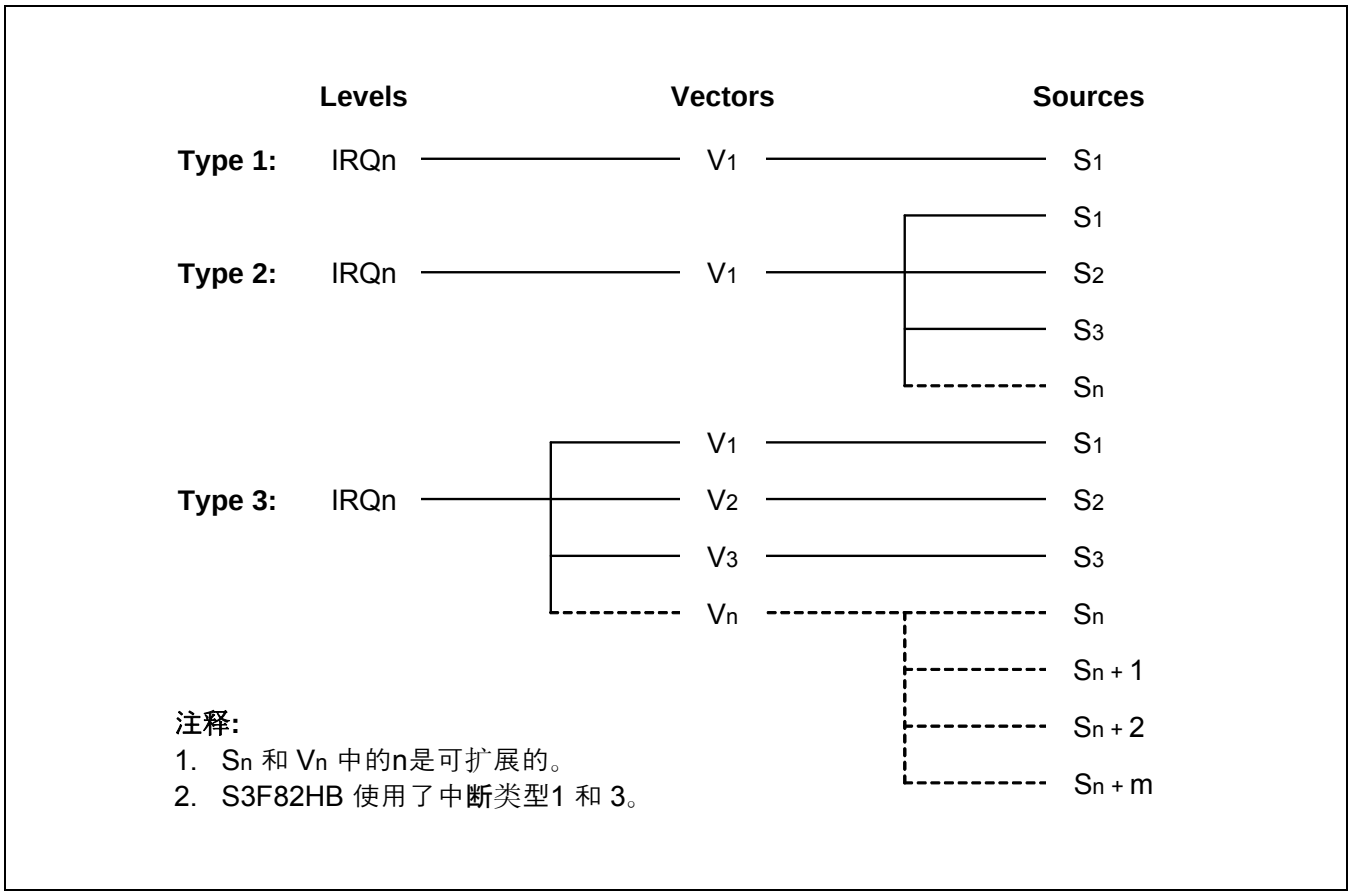


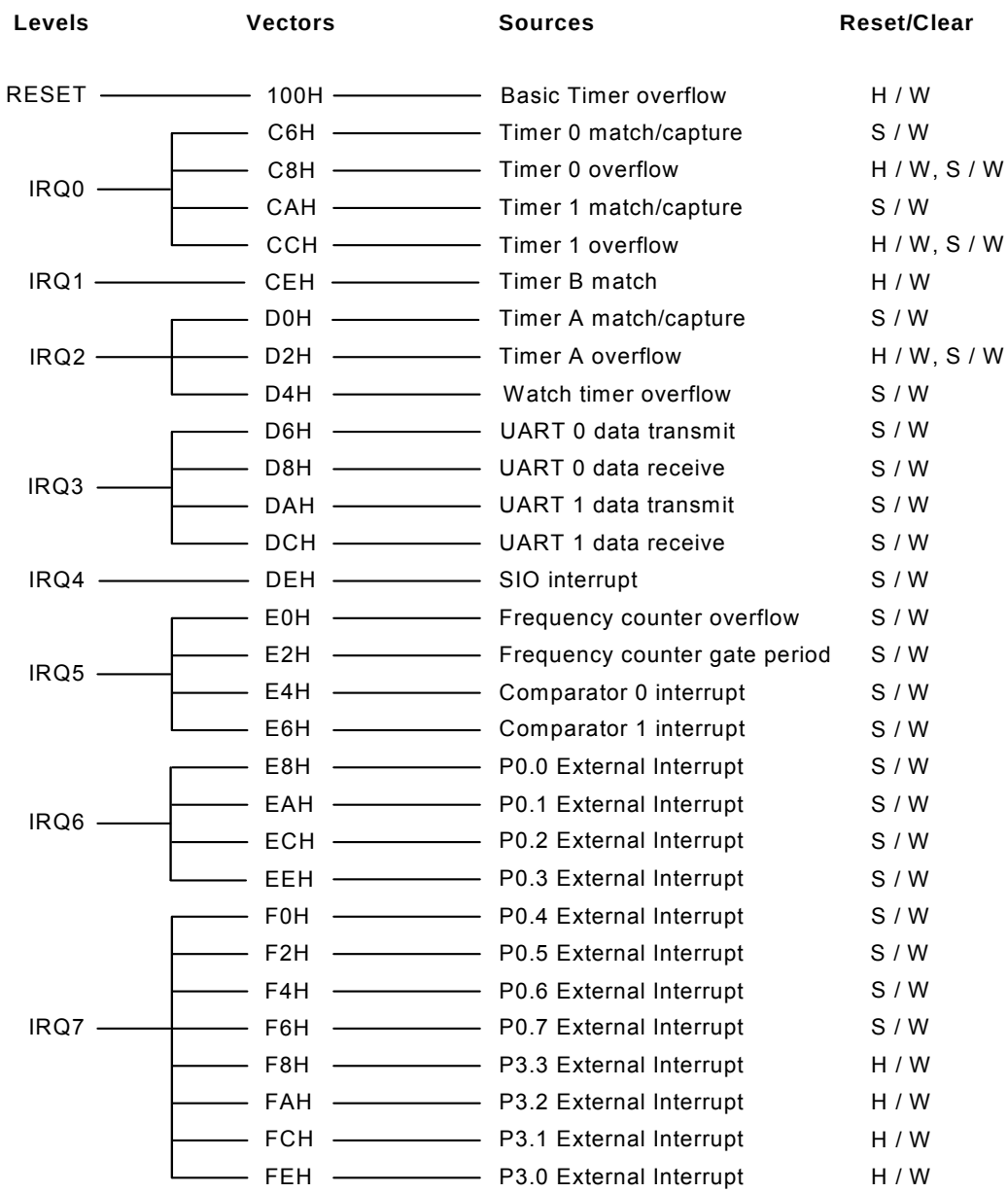
图 5-1 S3F8-系列的中断类型

5.1.5 S3F82HB 中断结构

S3F82HB 支持29个中断源，每一个中断源都有对应的中断向量地址。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 (IPR) 中的值决定了哪一个竞争中的中断级先被 CPU 响应。如果同一个中断级中的多个中断源同时发出中断请求，中断向量地址最小的那个中断源拥有最高优先级，首先被响应 (同一中断级内中断源优先级顺序由硬件决定)。

当 CPU 响应某个中断请求以后，中断处理就开始了。此时所有的其他中断都处于禁止状态，PC 和 FLAGS 的值被压入堆栈。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)，程序跳转而后开始执行中断服务程序。



注释：

1. 在给定的中断级中，越低的中断向量地址中断优先级越高
   例如：在IRQ5中，E0H 比 E2H 优先级高，
   同一中断级内部的优先级顺序出厂时决定。
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 S3F82HB 中断结构

5.1.5.1 中断向量地址

S3F82HB 中断结构的所有中断向量地址都位于内部64K 字节程序存储空间 (0H–FFFFH，*图 5-3*) 的向量地址区。

没有被用作中断向量的地址空间可以当作普通的程序存储空间。此时要小心千万不能与中断向量区交叠 (*表 5-1* 列出了所有的向量地址)。

ROM中的程序复位地址为 0100H。ROM 中的复位地址可在 smart option 中修改。详情参见第19章 嵌入式闪存接口。

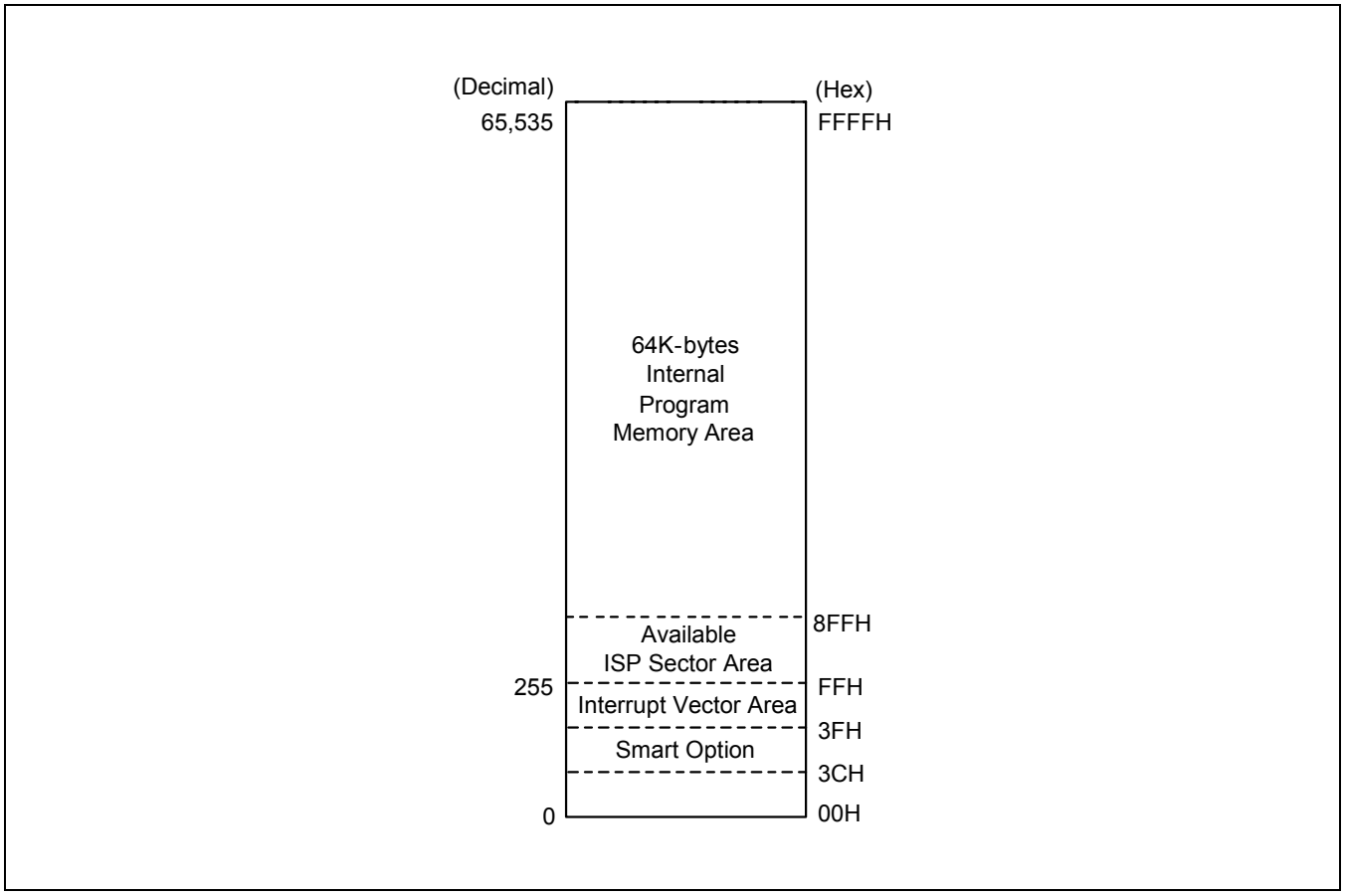


图 5-3 ROM 中断向量地址区

表 5-1 中断向量

中断地址		中断源	中断请求		复位/清除	
十进制值	十六进制值		中断级	级内优先级	硬件	软件
256	100H	Basic timer 溢出	Reset	–	√	
198	C6H	Timer 0 匹配 (match) /捕捉 (capture)	IRQ0	0		√
200	C8H	Timer 0 溢出		1	√	√
202	CAH	Timer 1 匹配 (match) /捕捉 (capture)		2		√
204	CCH	Timer 1 溢出		3	√	√
206	CEH	Timer B 匹配 (match)	IRQ1	–	√	
208	D0H	Timer A 匹配 (match) /捕捉 (capture)	IRQ2	0		√
210	D2H	Timer A 溢出		1	√	√
212	D4H	钟表定时器溢出		2		√
214	D6H	UART 0 数据发送	IRQ3	0		√
216	D8H	UART 0 数据接收		1		√
218	DAH	UART 1 数据发送		2		√
220	DCH	UART 1 数据接收		3		
222	DEH	SIO 中断	IRQ4	0		√
224	E0H	频率计数器溢出	IRQ5	0		√
226	E2H	频率计数器门周期		1		√
228	E4H	比较器0 中断		2		√
230	E6H	比较器1 中断		3		√
232	E8H	P0.0 外部中断	IRQ6	0		√
234	EAH	P0.1 外部中断		1		√
236	ECH	P0.2 外部中断		2		√
238	EEH	P0.3 外部中断		3		√
240	F0H	P0.4 外部中断	IRQ7	0		√
242	F2H	P0.5 外部中断		1		√
244	F4H	P0.6 外部中断		2		√
246	F6H	P0.7 外部中断		3		√
248	F8H	P3.3 外部中断		4	√	
250	FAH	P3.2 外部中断		5	√	
252	FCH	P3.1 外部中断		6	√	
254	FEH	P3.0 外部中断		7	√	

注释:

1. 中断优先级标注是反向的：“0”的优先级最高，“1”第二高，依次往下。
2. 若2个或多个同级中断同时发生，则向量地址较低的中断具有较高的优先级。同级中断的优先级由硬件决定。

5.1.5.2 使能/禁止中断的指令 (EI, DI)

执行使能中断指令 (EI) 使能全局中断结构。然后，所有的中断按照既定的优先级顺序执行。

注释: 复位后，在初始化程序中应该包含 EI 指令来使能全局中断结构。

在正常的操作中，可以执行 DI (禁止中断) 指令随时禁止全局中断处理。EI 和 DI 指令会改变寄存器 SYM 的第0位状态。

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个系统级寄存器也控制中断处理：

- 中断屏蔽寄存器，IMR，使能 (解除屏蔽) 或者禁止 (屏蔽) 中断级。
- 中断优先级控制寄存器，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 (不同于中断源的标志位)。
- 系统模式控制寄存器，SYM，使能或者禁止全局中断处理 (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 中断控制寄存器描述

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 中任意一个中断级的中断处理。
中断优先级控制寄存器	IPR	R/W	控制各个中断级间的相对优先级。S3F82HB 中的7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	包含每个中断级的中断标志位
系统模式控制寄存器	SYM	R/W	使能或者禁止快速中断处理，动态全局中断处理以及外部接口控制 (S3F82HB 内置外部存储器接口)。

注释: IMR 寄存器改变成任何值前，都必须禁止所有中断。建议使用 DI 指令。

5.1.6 中断处理控制要点

中断处理控制可以通过两种方式来完成：全局或者特定中断级和中断源控制。系统级的中断结构控制要点如下：

- 全局中断的使能/禁止 (通过 EI 和 DI 指令或直接操作 SYM.0)
- 中断级使能/禁止 (通过寄存器 IMR)
- 中断级优先级设置 (通过寄存器 IPR)
- 通过相应的外设控制寄存器使能/禁止中断源

注释: 包含中断处理的应用程序中，务必确保包含必须的寄存器文件地址 (寄存器指针) 信息。

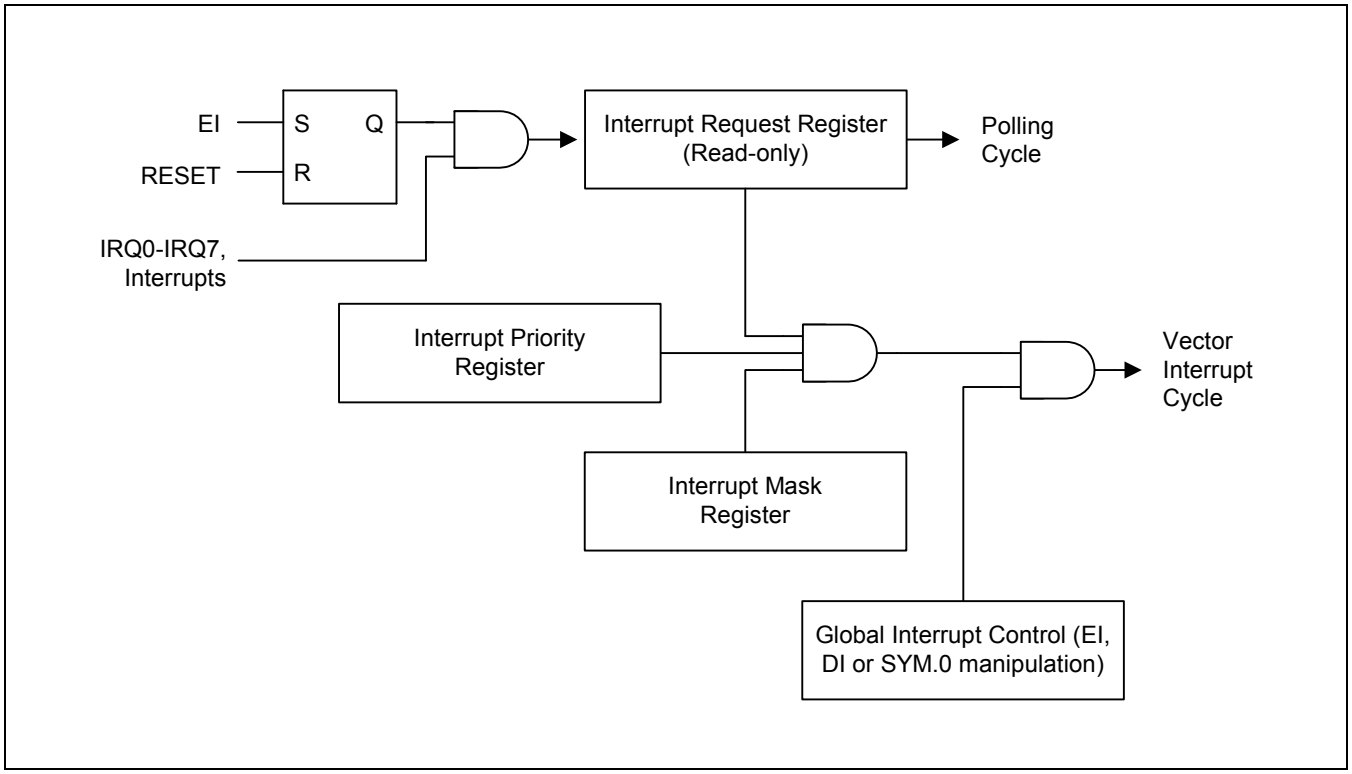


图 5-4 中断功能框图

5.1.7 外设中断控制寄存器

每一个中断源都有一个或多个对应的外设控制寄存器来控制相关外设产生的中断。(*表 5-3*)。

表 5-3 中断源控制和数据寄存器

中断源	中断级	寄存器	Set 1 中寄存器地址
Timer 0 匹配/捕获中断 Timer 0 溢出中断	IRQ0	T0CON, INTPND T0CNTH, T0CNTL, T0DATAH, T0DATAL	E3H, D0H, bank 0 E4H, E5H, bank 0 E6H, E7H, bank 0
Timer 1 匹配/捕获中断 Timer 1 溢出中断		T1CON, INTPND T1CNTH, T1CNTL T1DATAH, T1DATAL	EBH, D0H, bank 0 ECH, EDH, bank 0 EEH, EFH, bank 0
Timer B 匹配中断	IRQ1	TBCON TBDATAH, TBDATAL	F2H, bank 0 F0H, F1H, bank 0
Timer A 匹配/捕获中断 Timer A 溢出中断	IRQ2	TACON TACNT, TADATA INTPND	E8H, bank 0 F9H, EAH, bank 0 D0H, bank 0
钟表定时器溢出中断		WTCON	D1H, bank 0
UART 0 数据发送中断 UART 0 数据接收中断	IRQ3	UART0CONH, UART0CONL UART0DATA, BR0DATA	00H, 01H page 0 02H, 03H page 0
UART 1 数据发送中断 UART 1 数据接收中断		UART1CONH, UART1CONL UART1DATA, BR1DATA	04H, 05H page 0 06H, 07H page 0
SIO 中断	IRQ4	SIOCON SIODATA, SIOPS	E0H, bank 0 E1H, E2H, bank 0
频率计数器溢出中断 频率计数器门周期中断	IRQ5	FCCON, FCNTH, FCNTL INTPND	F3H, F4H, F5H, bank 0 D0H, bank 0
比较器0 中断 比较器1 中断		CMP0CON CMP1CON COUTCON	08H, page 0 09H, page 0 0FH, page 0
P0.0 外部中断 P0.1 外部中断 P0.2 外部中断 P0.3 外部中断	IRQ6	P0CONL P0INTL P0PND	E1H, bank 1 E3H, bank 1 E4H, bank 1
P0.4 外部中断 P0.5 外部中断 P0.6 外部中断 P0.7 外部中断	IRQ7	P0CONH P0INTH P0PND	E0H, bank 1 E2H, bank 1 E4H, bank 1
P3.3 外部中断 P3.2 外部中断 P3.1 外部中断 P3.0 外部中断		P3CONL P3INT	EBH, bank 1 ECH, bank 1

注释: 若某一中断在 IMR 寄存器中未被屏蔽 (使能中断级)，则其中断标志位和使能位必须在 DI 指令之后写入。

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，控制快速中断处理。(*图 5-5*)。

复位时清零 SYM.1 和 SYM.0，快速中断级选择位 SYM.4–SYM.2 的值不确定。

EI 和 DI 指令可以使能或禁止全局中断处理，相应的，也可以通过修改 SYM.0 得到同样效果。
为使能中断处理，复位后应该在初始化程序中包含 EI 指令。尽管可以通过操作 SYM.3 位直接使能或禁止中断，但还是建议用 EI 指令或 DI 指令。

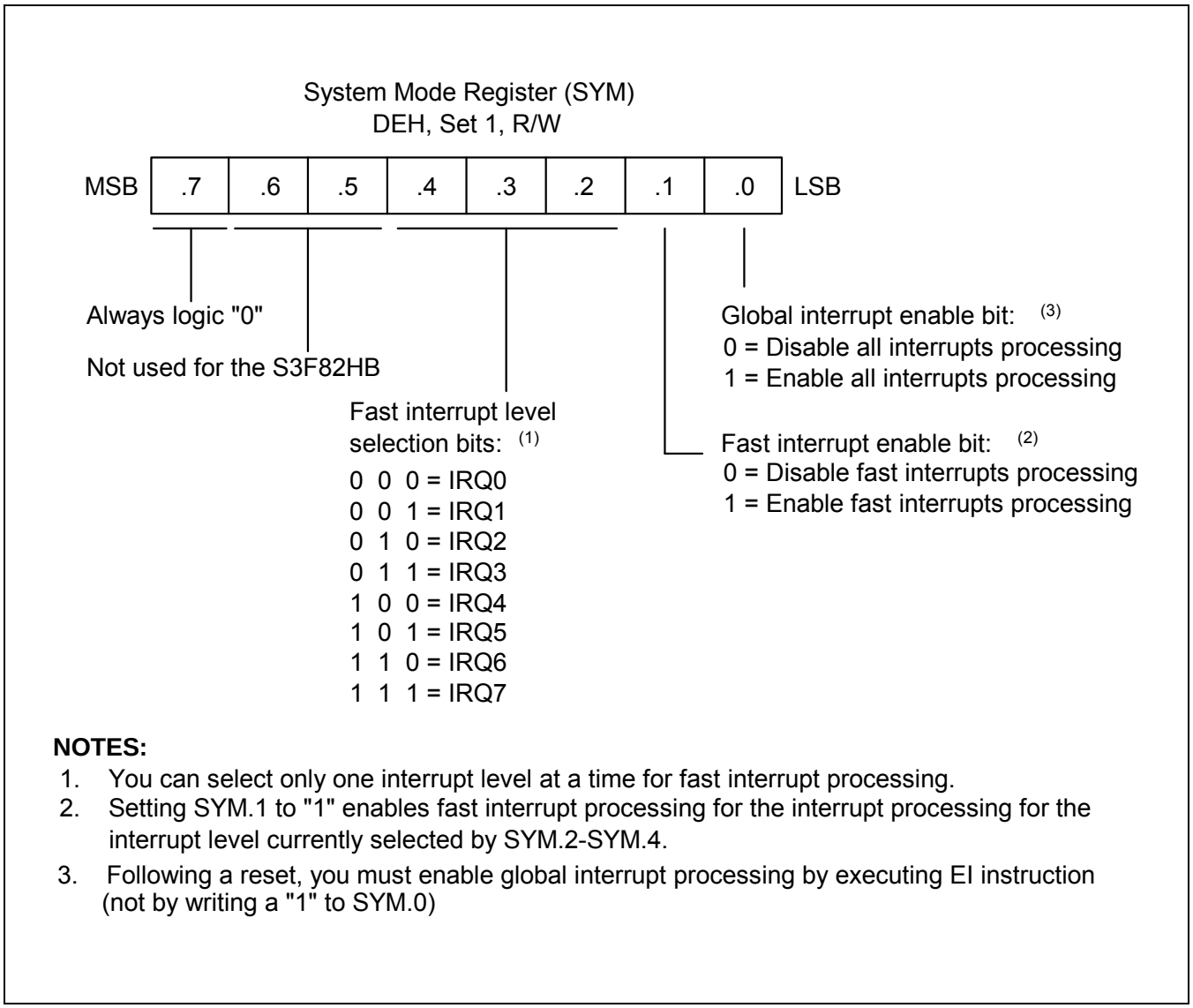


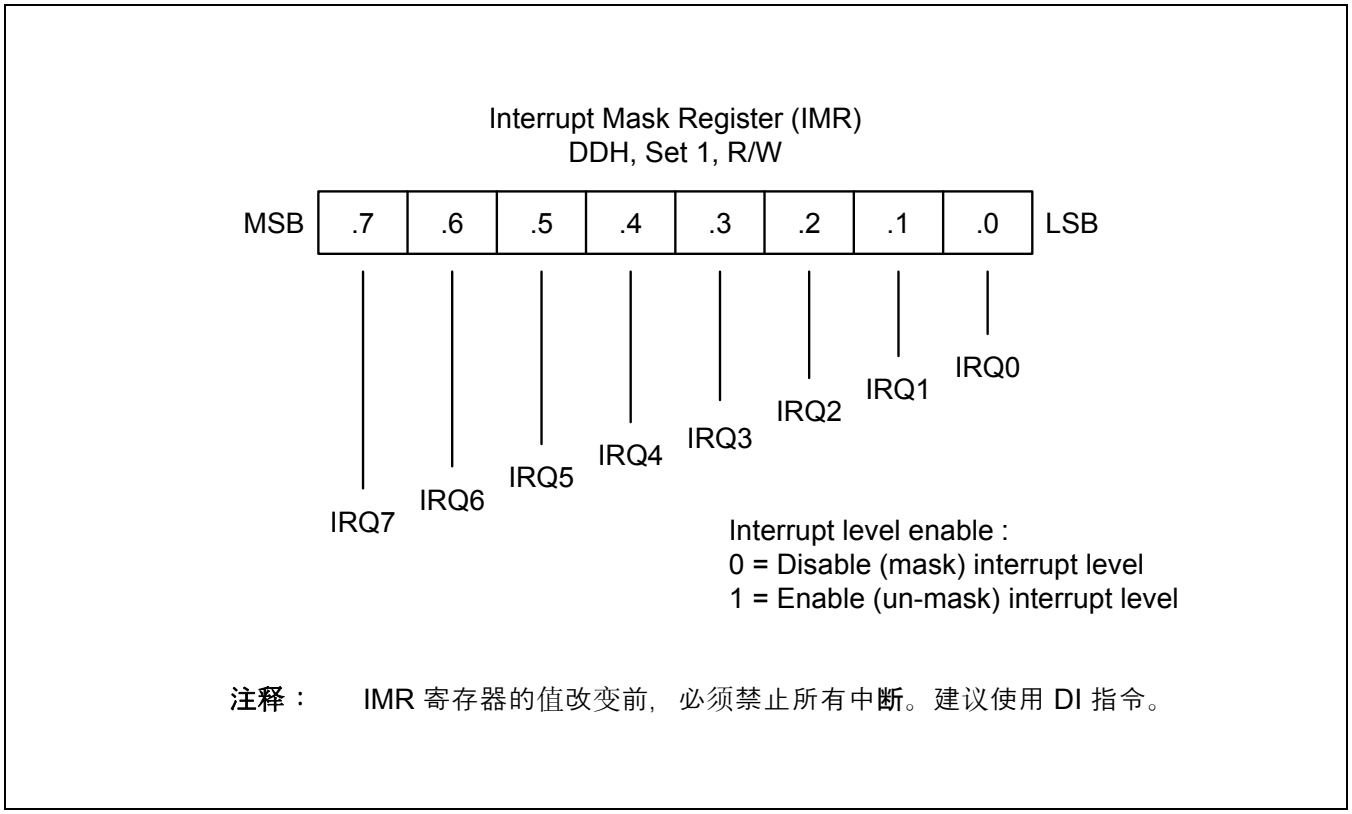
图 5-5 系统模式寄存器 (SYM)

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，所以必须在初始化程序中根据应用设置该寄存器。

IMR 中的每一位都对应一个中断级，第1位对应于 IRQ1，第2位对应于 IRQ2，以此类推。
当 IMR 中的某位被清“0”后，与之对应的中断级的中断处理就被禁止 (屏蔽) 了。
当设置为“1”时，相应中断级的中断处理就被使能(解除屏蔽) 了。

IMR 寄存器映射于 set 1 的 DDH 地址。可通过指令使用寄存器寻址模式对每一位进行读写。



注释： IMR 寄存器的值改变前，必须禁止所有中断。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)

中断优先级控制寄存器 IPR (地址：FFH，Set 1，Bank 0) 可以用来设置 MCU 中断结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，所以必须在初始化程序中根据应用设置该寄存器。

当多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 CPU 响应。
如果同一中断级中的多个中断源同时发出中断请求，中断向量地址最小的那个最先被响应。

为支持相对中断优先级编程，中断逻辑将中断级分为组和子组。请注意，这些组 (子组) 仅供 IPR 逻辑定义 IP 寄存器的优先级。(*图 5-7*)：

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

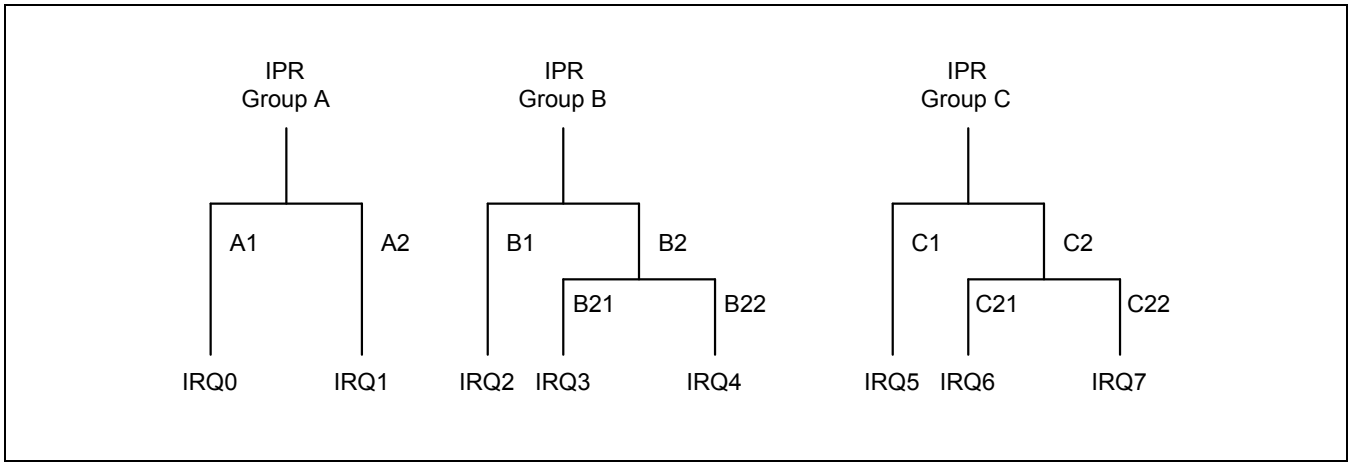


图 5-7 中断请求优先级组

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

IPR 其他位控制功能描述如下：

- IPR.5 控制中断组 C 内中断级的相对优先级。
- 中断组 C 包含一个子组，内部的中断级5，6，7有额外的优先级。子组的优先级由 IPR.6 的值决定。IPR.5 控制中断组 C。
- IPR.0 控制 IRQ0 和 IRQ1 的相对优先级。

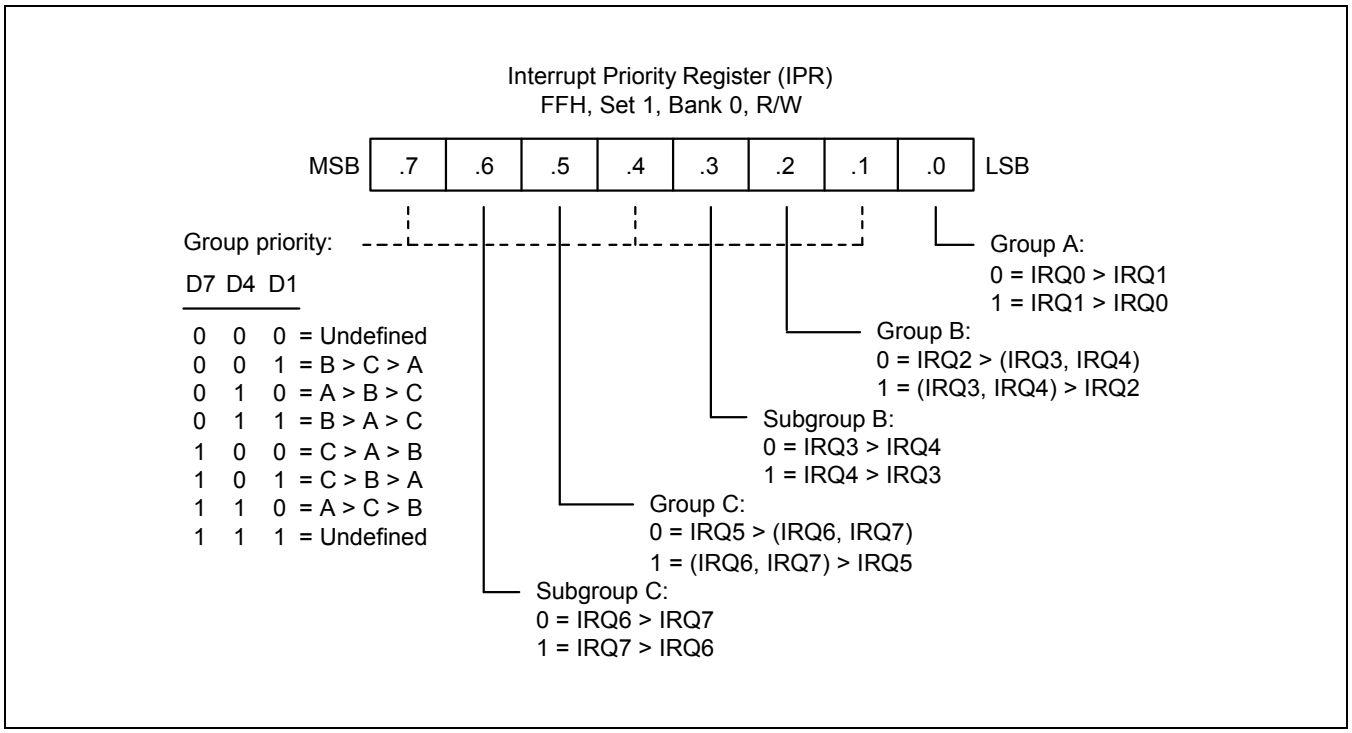


图 5-8 中断优先级寄存器 (IPR)

5.1.11 中断请求寄存器 (IRQ)

可以通过查询中断请求寄存器 IRQ (地址：DCH，Set 1) 的每一位监视所有中断级的中断请求情况。
寄存器中的每一位都对应于一个同序号的中断级：第0位对应于 IRQ0，第1位对应于 IRQ1，以此类推。
“0”代表目前该中断级没有中断发生，“1”代表该中断级中的某个中断源发出了中断请求。

IRQ 寄存器是只读的。可随时使用位或字节方式访问 IRQ 寄存器来读取 (测试) 某一中断级当前的中断请求。
复位后，所有状态位清0。

即使是执行 DI 指令后，即全局中断处理禁止的情况下，仍然可以查询 IRQ 的内容。 此时如果有中断发生，CPU 并不会做出任何响应。但是仍然可以通过查询 IRQ 确定中断发生情况。
可在全局中断屏蔽的情况下，使用上述方法判断事件的发生。

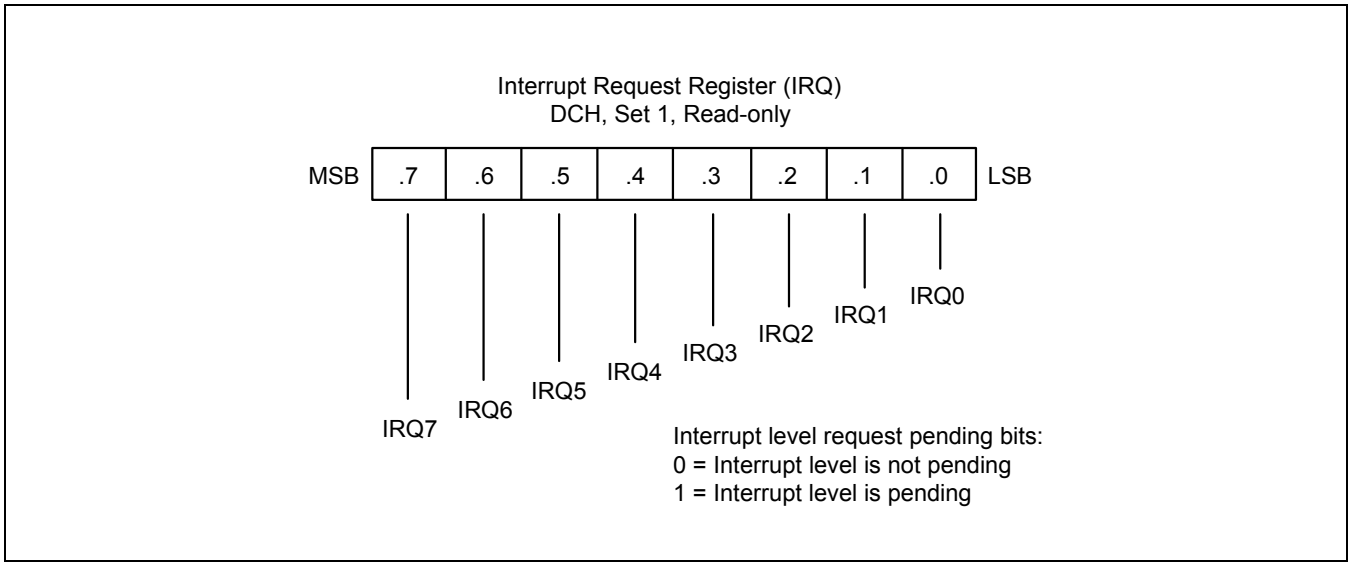


图 5-9 中断请求寄存器 (IRQ)

5.1.12 中断标志位类型

5.1.12.1 概述

有两种类型的中断标志位：一种会在中断服务程序执行完毕后硬件自动清零，一种必须在中断服务程序中软件清零.

5.1.12.2 硬件自动清零的标志位

对于硬件自动清零的标志位来说，当有中断请求时，中断逻辑置高相应的标志位。然后触发 IRQ 脉冲告知 CPU 有一个中断正等待处理。CPU 随后发送 IACK 给中断源确认接受请求，然后执行服务程序，最后清除标志位。这种类型的标志位没有被映射，所以不能在软件中对其进行读写。

在 S3F82HB 的中断结构中，timer 0 溢出中断 (IRQ0)，timer 1 溢出中断 (IRQ0)，timer B 匹配中断 (IRQ1)，timer A 溢出中断 (IRQ2)，以及 P3.3~P3.0 口外部中断 (IRQ7) 就属于这个类型的中断，标志位硬件自动清零。

5.1.12.3 中断服务程序中清零的标志位

第二类标志位必须用软件清零。中断服务程序必须在中断返回 (IRET) 前清除标志位。
可通过对中断源的模式或控制寄存器中的相应标志位清“0”来实现。

编程实例 5-1 如何清除中断标志位

如下例所示，应使用 load 指令来清除中断标志位。

例：

```
1.   SB1
     LD    P0PND, #11111011B        ; P0.2 中断标志位清0
     •
     •
     •
     IRET

2.   SB0
     LD    INTPND, #11111101B       ; timer A 匹配/捕获中断标志位清0
     •
     •
     •
     IRET
```

5.1.13 中断源查询顺序

中断请求查询和响应的步骤如下：

1. 某一中断源通过置位中断请求位产生中断请求。
2. CPU 查询程序识别该中断源的中断挂起状态。
3. CPU 检查该中断源的中断级。
4. CPU 产生中断确认信号。
5. 中断逻辑决定中断的向量地址。
6. 中断服务程序启动并将该中断源的标志位清0。(硬件清0或软件清0)
7. CPU 继续查询中断请求。

5.1.14 中断服务程序

中断处理前，必须满足以下条件：

- 全局中断处理使能 (EI)
- 该中断源所处的中断级处于使能状态 (IMR 寄存器)
- 若有多个中断请求，该中断级必须具有最高优先级
- 该中断源处于使能状态 (外设控制寄存器)

当所有的条件都满足之后，中断请求会在指令周期的最后被确认。随后，CPU 将开始中断处理并完成下列操作：

1. SYM 寄存器的 SYM.0 位清零，以禁止所有后续的中断。
2. 把程序计数器 PC 和状态寄存器压入堆栈。
3. 跳转到中断向量，获取中断服务程序地址。
4. 跳转执行中断服务程序。

中断服务程序完成以后，CPU 发出中断返回指令 (IRET)。中断返回指令 IRET 会将堆栈中的 PC 和状态寄存器重置，同时置位SYM.0，使 CPU 能响应后续的中断请求。

5.1.15 中断向量地址的生成

ROM (00H–FFH) 中的中断向量区域，包含了中断结构中的每一级对应的中断服务程序的入口地址。
向量化的中断处理流程如下：

1. 程序计数器 PC 低字节压入堆栈。
2. 程序计数器 PC 高字节压入堆栈。
3. FLAGS 寄存器压入堆栈。
4. 从中断向量地址中取出中断服务程序入口地址 (高字节)。
5. 从中断向量地址中取出中断服务程序入口地址 (低字节)。
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 位向量地址始终起始于 ROM 空间 00H ~ FFH 中的某个偶数地址。

5.1.16 中断嵌套

可以在一个低优先级中断请求的处理过程中，嵌套一个高优先级的中断请求。为此，必须遵循如下步骤：

1. 将当前的8位 IMR 寄存器压栈。
2. 重置 IMR，仅使能那个有较高优先级的中断。
3. 执行 EI 使能中断处理 (较高优先级的中断一旦发生就可被响应)
4. 在较低优先级的中断服务处理程序最后，将 IMR 寄存器弹栈，恢复进入中断前的状态。
5. 执行中断返回指令 IRET。

可根据不同的应用，简化上述步骤。

5.1.17 指令指针 (IP)

S3F8- 全系列都采用了指令指针 (IP) 以支持高速中断处理，又称作快速中断 (fast interrupts)。
IP实际是个寄存器对，高8位 IPH (IP15–IP8) 和低8位 IPL (IP7–IP0) 分别位于地址单元 DAH 和 DBH。

5.1.18 快速中断处理

一般的中断处理需要16个CPU 时钟周期，而一旦将给定中断级设为快速中断，只需要大概6个 CPU 时钟周期就可以完成整个中断处理。通过写 SYM.4–SYM.2 可以将快速中断处理应用于某一个中断级。然后需要置位 SYM.1 使能快速中断处理。

5.1.18.1 快速中断处理 (续)

快速中断处理用到其他2个系统寄存器：

指令指针 (IP) 包含了快速中断服务程序的起始地址 (中断处理后会和 PC 交换完成中断返回)

当快速中断发生时，FLAGS 寄存器的内容会自动载入一个没有地址映射的专用寄存器，称为 FLAGS'

注释: 对于 S3F82HB，8 个中断级 IRQ0–IRQ7 中的任何一个都可以选择为快速中断处理的对象。

5.1.18.2 快速中断初始化步骤

以下步骤初始化快速中断：

1. 将中断服务程序的起始地址载入指令指针 (IP)。
2. 将中断级编号 (IRQn) 载入快速中断选择区 (SYM.4–SYM.2)。
3. 在 SYM 寄存器的快速中断使能位中写“1”。

5.1.18.3 快速中断服务程序

当指定为快速中断的中断级中出现中断请求时，如下事件发生：

1. PC 和 IP 交换内容。
2. FLAGS 寄存器值载入 FLAGS'。
3. FLAGS 寄存器中的快速中断状态位置高。
4. 开始中断服务。
5. 若快速中断标志位被置高，当中断服务程序结束后，PC 和 IP 再次交换内容。
6. FLAGS' 寄存器的值自动复制回 FLAGS 寄存器。
7. FLAGS 寄存器中的快速中断标志位自动清0。

5.1.19 中断标志位类型间关系

如前所述，有2种中断标志位：1种在中断服务程序被识别并执行之后由硬件自动清0；另一种必须由中断服务程序软件清0。可为快速中断处理选择任一种类型的中断标志位清0–硬件清0或软件清0。

5.1.20 编程指导

切记使能/禁止快速中断的唯一方法是置位/清0 SYM 寄存器中的快速中断使能位 SYM.1。执行 EI 或 DI 指令使能或禁止全局中断处理，包括快速中断。若使用快速中断，务必要在快速中断服务程序的最后，在IP 中写入一个新的起始地址。

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 64 K 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 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。

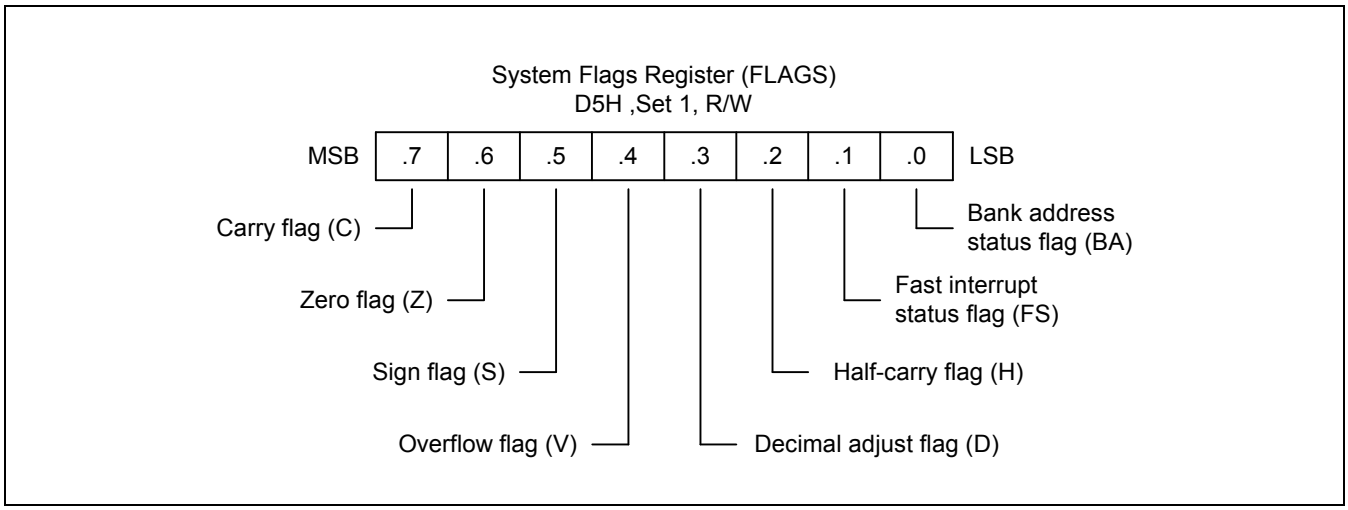


图 6-1 系统标志寄存器 (FLAGS)

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

符号	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

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 6-8*。

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	寻址模式	
			(Hex)	dst	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:

	字节数	时钟周期	指令代码	寻址模式	
(注释)			(Hex)	dst	src
opc | src | b | 0 | dst	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:

	字节数	时钟周期	指令代码	寻址模式	
(注释)			(Hex)	dst	src
opc | src | b | 1 | dst	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:

	字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc | 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	寻址模式	
					(Hex)	dst	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:

opc	dst

字节数	时钟周期	指令代码 (Hex)	寻址模式 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       115
+     0 0 1 0    0 1 1 1       127
      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:

opc	dst

字节数	时钟周期	指令代码 (Hex)	寻址模式 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 ≥ 28，
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。

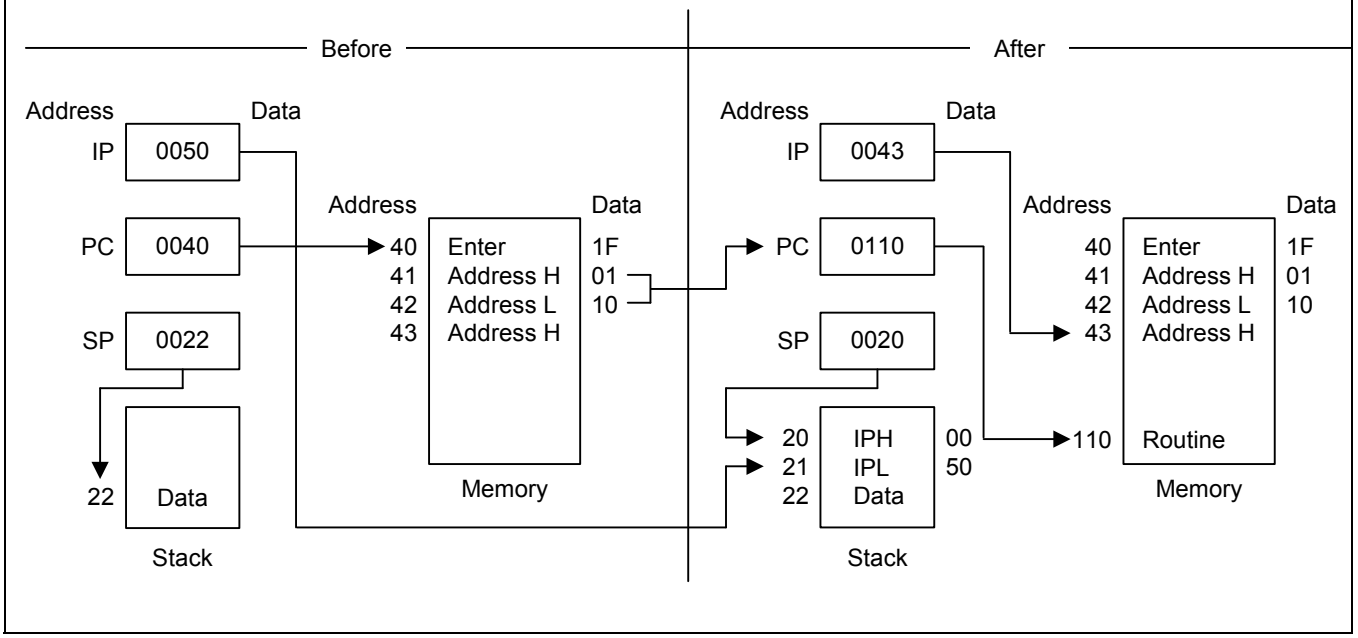
标志位: 没有标志位受影响。

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。

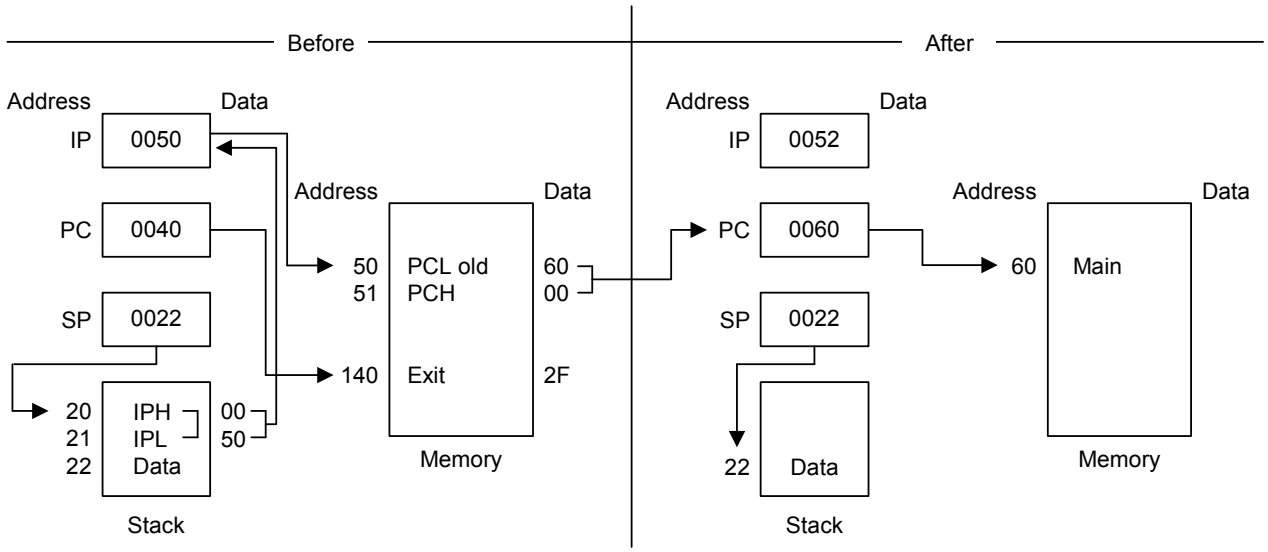
标志位: 没有标志位受影响。

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 (Hex)	寻址模式 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)

```
标志位  ←  @SP
SP      ←  SP + 1
PC      ←  @SP
SP      ←  SP + 2
SYM(0)  ←  1
```

IRET (快速)

```
PC  ↔  IP
标志位  ←  标志位'
FIS  ←  0
```

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)。

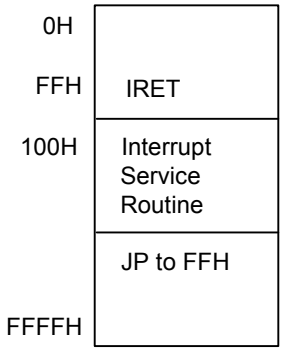
格式:

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

IRET (快速)	字节数	时钟周期	指令代码 (Hex)
opc	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:

(注释)		字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
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“lrr”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	lrr	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“lrr”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	lrr	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:

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	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。

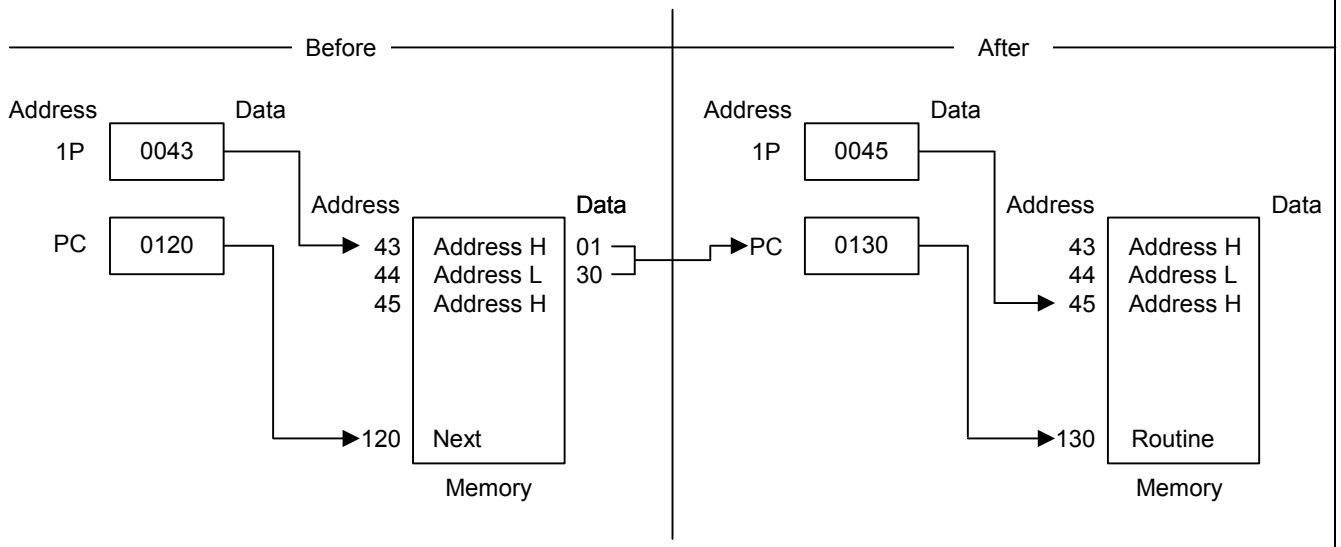
标志位: 没有标志位受影响。

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

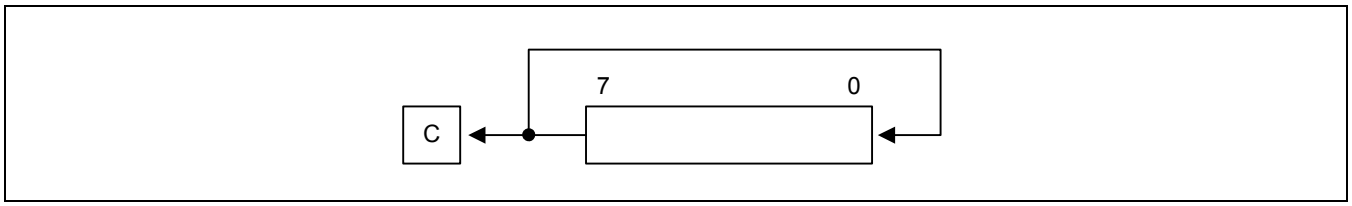
指令“RET”将堆栈指针 00FCH 中的内容(10H)弹出到程序计数器的高字节中，00FDH 中的内容(1AH)弹出到 PC 的低字节，地址 101AH 中的指令被执行。堆栈指针现在指向地址 00FEH。

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

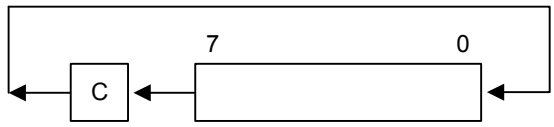
```
假如：寄存器 00H = 0AAH, 寄存器 01H = 02H, 寄存器 02H = 17H:

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:

	字节数	时钟周期	指令代码 (Hex)	寻址模式 dst
opc | 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。

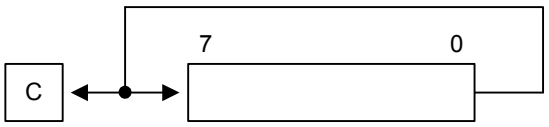
6.3.1.60 RR—右移 (Rotate Right)

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

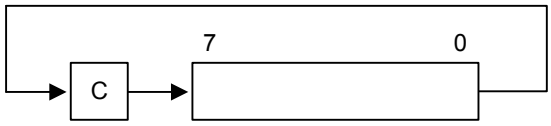
```
假如：寄存器 00H = 31H, 寄存器 01H = 02H, 寄存器 02H = 17H:

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)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 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)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 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

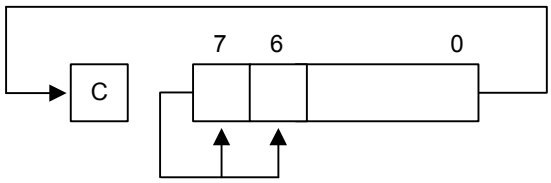
语句 SCF 将进位标志置 1。

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

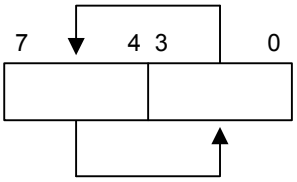
```
假如：R1 = 12H, R2 = 03H, 寄存器 01H = 21H, 寄存器 02H = 03H, 寄存器 03H = 0AH:

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。

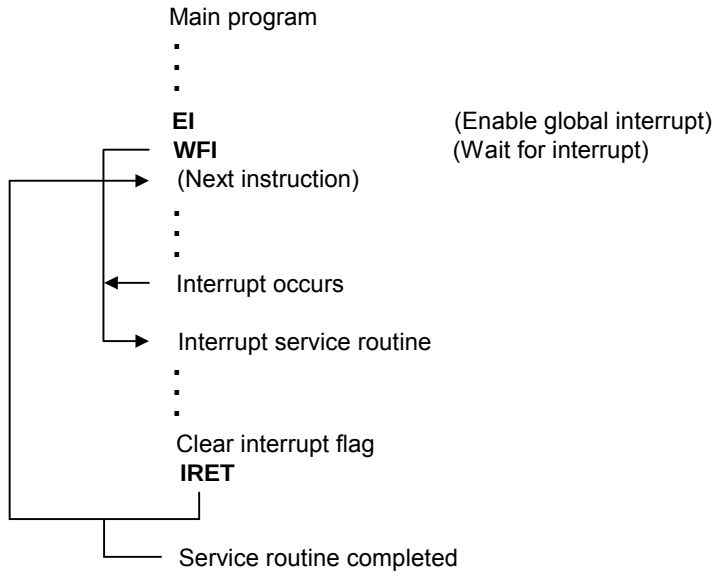
标志位: 没有标志位受影响

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

S3F82HB 有2个振荡器电路：主时钟电路和副时钟电路。
CPU，外设等硬件在此电路提供的系统时钟频率下工作。S3F82HB 的最大 CPU 时钟频率由 CLKCON 寄存器的设置决定。

系统时钟电路

系统时钟电路由下列部分组成：

- 外部晶振，陶瓷谐振器，RC 振荡源或外部时钟源
- 振荡器停止和唤醒功能
- CPU 时钟可通过编程进行分频 (fxx 1, 2, 8, 16分频)
- 系统时钟控制寄存器 CLKCON
- 振荡器控制寄存器 OSCCON 和 STOP 控制寄存器 STPCON

7.1.1 CPU 时钟助记符

本文档中，下列助记符用来描述 CPU 时钟：

fx: 主时钟

fxt: 副时钟

fxx: 选定的系统时钟

7.1.1.1 主振荡器电路

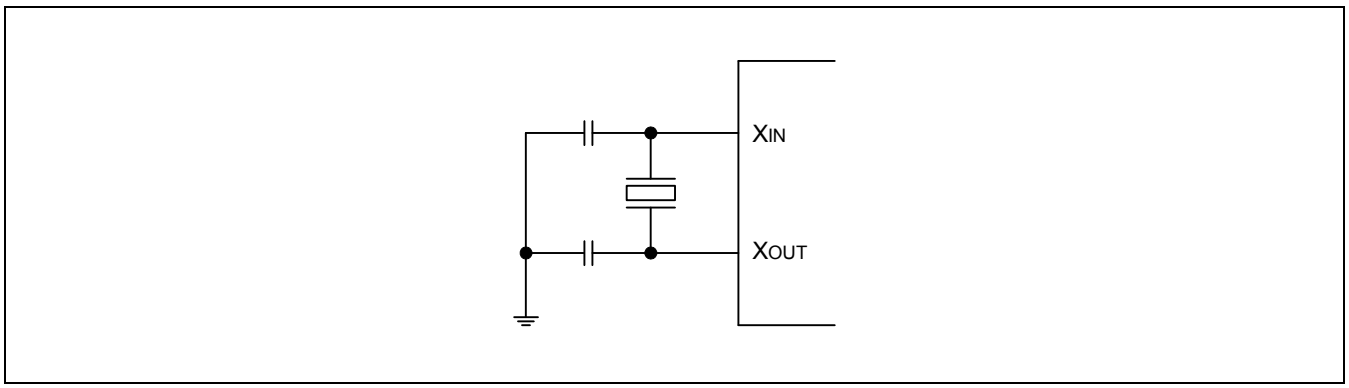


图 7-1 晶体/陶瓷振荡器 (fX)

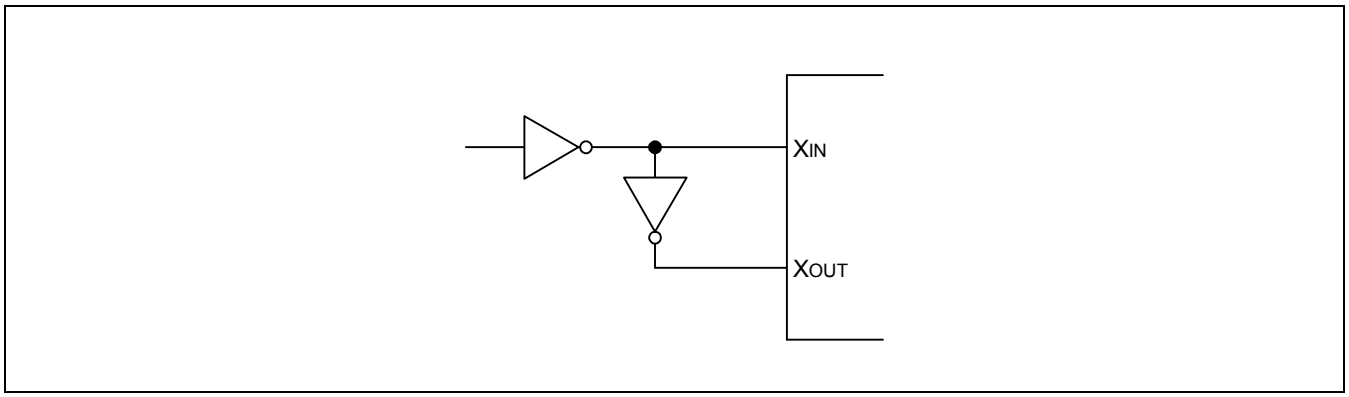


图 7-2 外部振荡器 (fX)

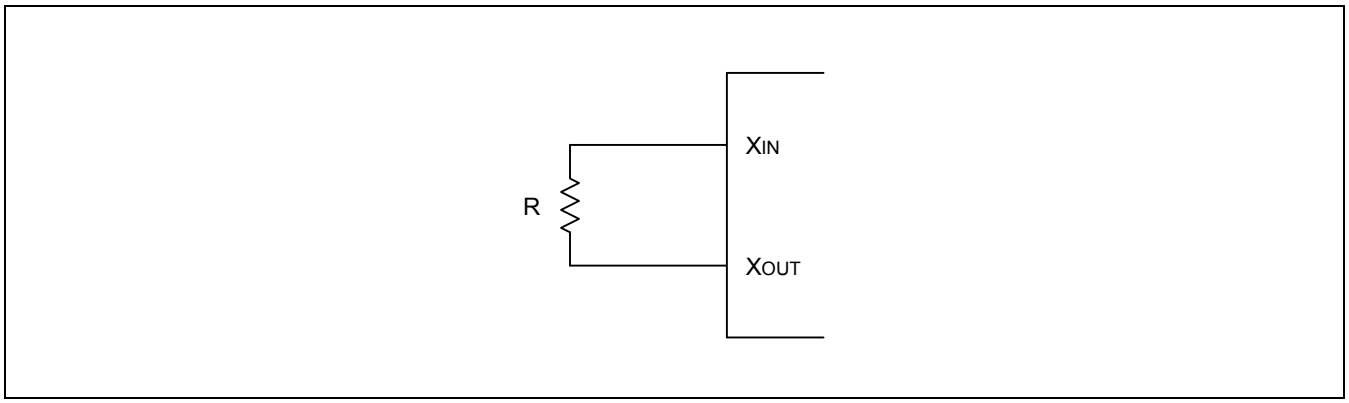


图 7-3 RC 振荡器 (fX)

7.1.1.2 副振荡器电路

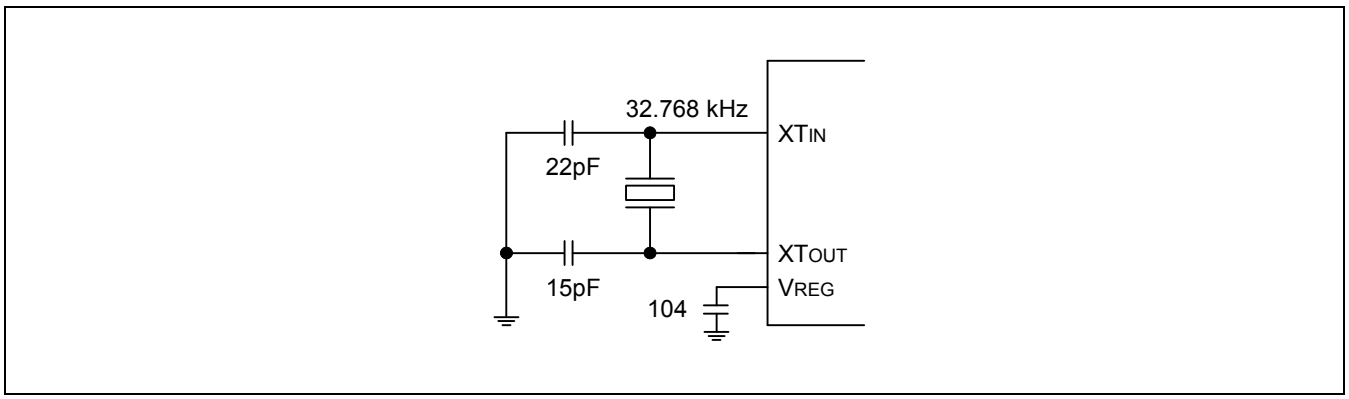


图 7-4 晶振 (fxt)

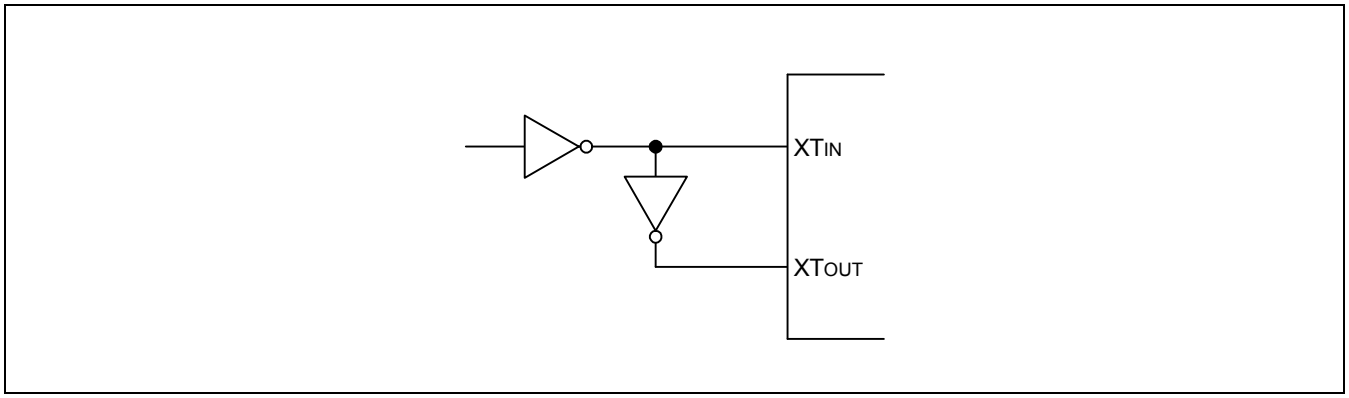


图 7-5 外部振荡器 (fxt)

7.1.1.3 省电模式下时钟状态

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。在复位操作或具有 RC 延时噪声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。副振荡器工作且钟表定时器的时钟源是副时钟时，CPU 也能退出STOP 模式。
- 在 IDLE 模式下，内部时钟信号进入 CPU 的通路被切断，但中断及 Timer 仍然工作。复位或中断 (外部中断或内部中断) 都可以使 CPU 退出 IDLE 模式。

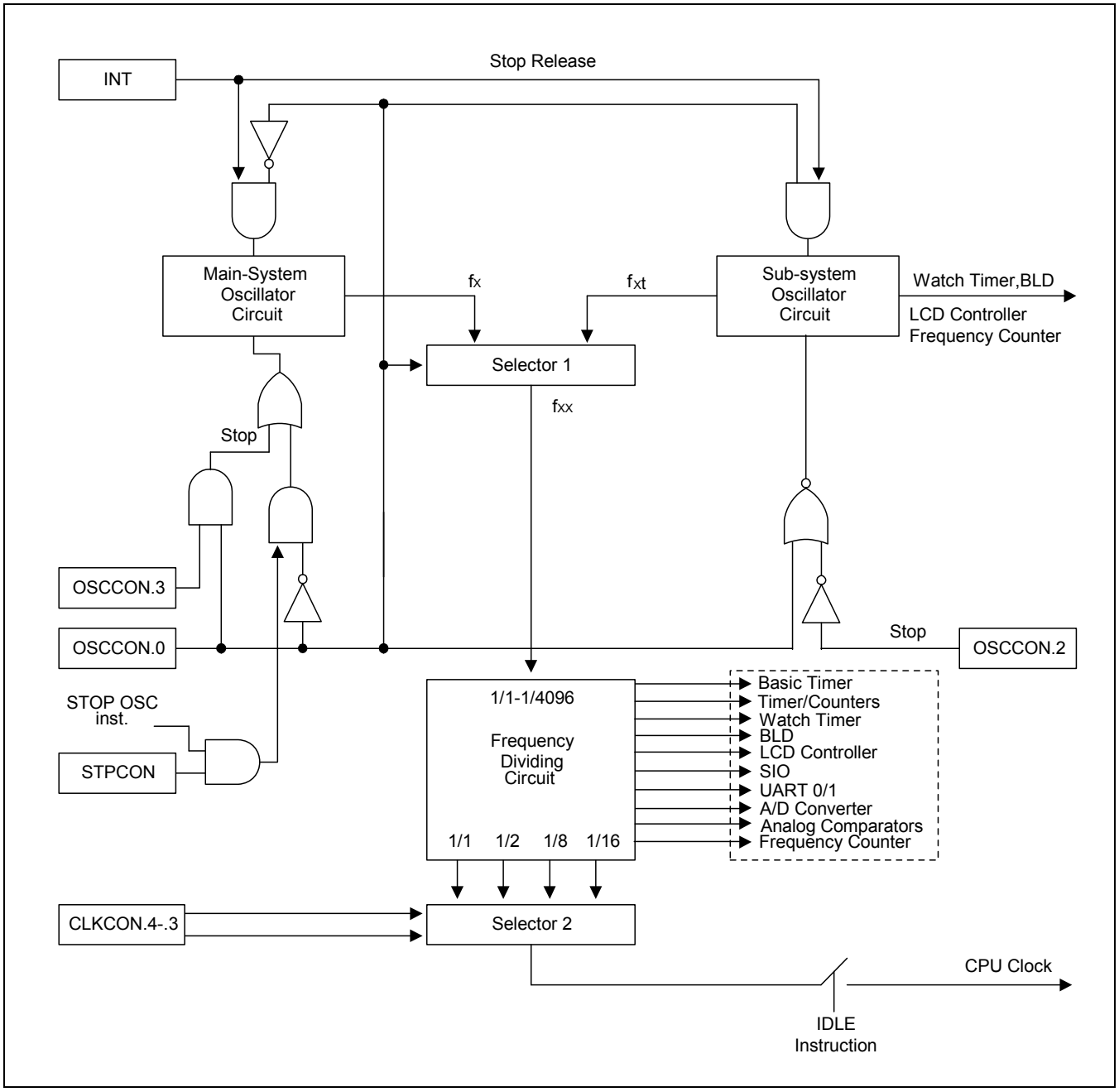


图 7-6 系统时钟电路框图

7.1.1.4 系统时钟控制寄存器 (CLKCON)

系统时钟控制寄存器 CLKCON 的地址是 set 1的 D4H，可读/写，有如下功能：

- 系统时钟分频系数选择

主振荡器建立以后，f_{OSC} /16 (最慢时钟速度) 被选择为 CPU 时钟。如有需要，也可提高 CPU 时钟速度至 f_{OSC}，f_{OSC}/2 或 f_{OSC} /8。

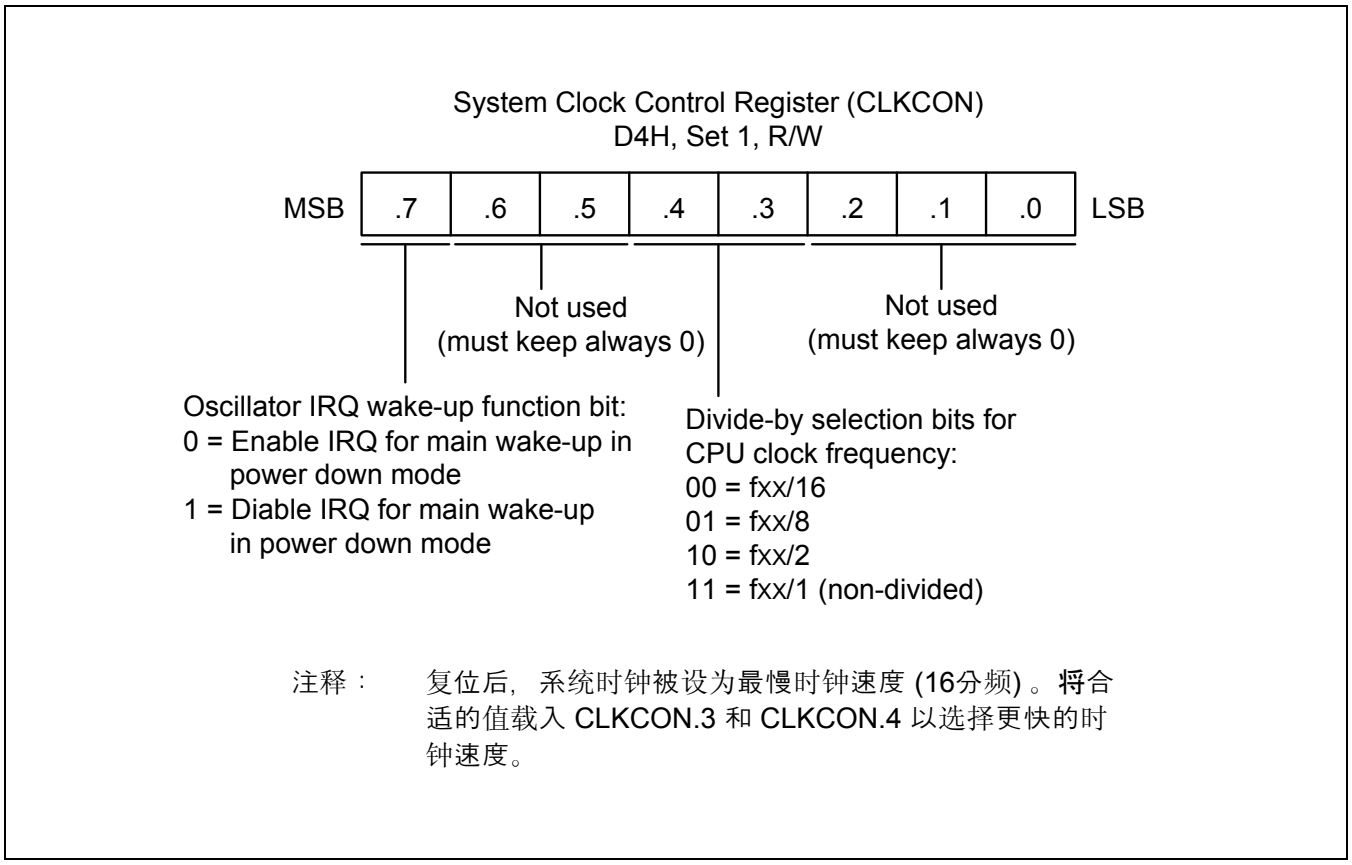


图 7-7 系统时钟控制寄存器 (CLKCON)

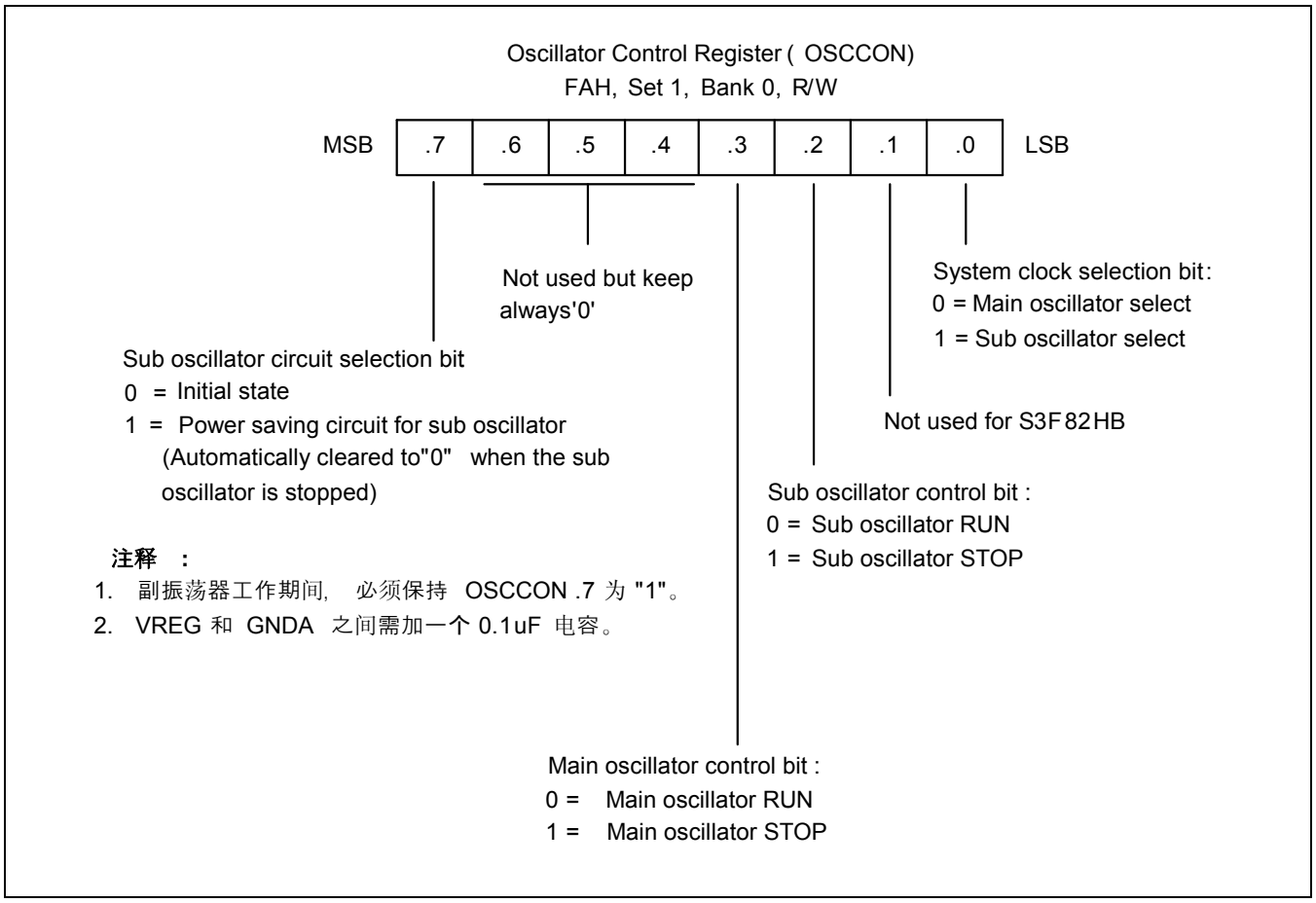


图 7-8 振荡器控制寄存器 (OSCCON)

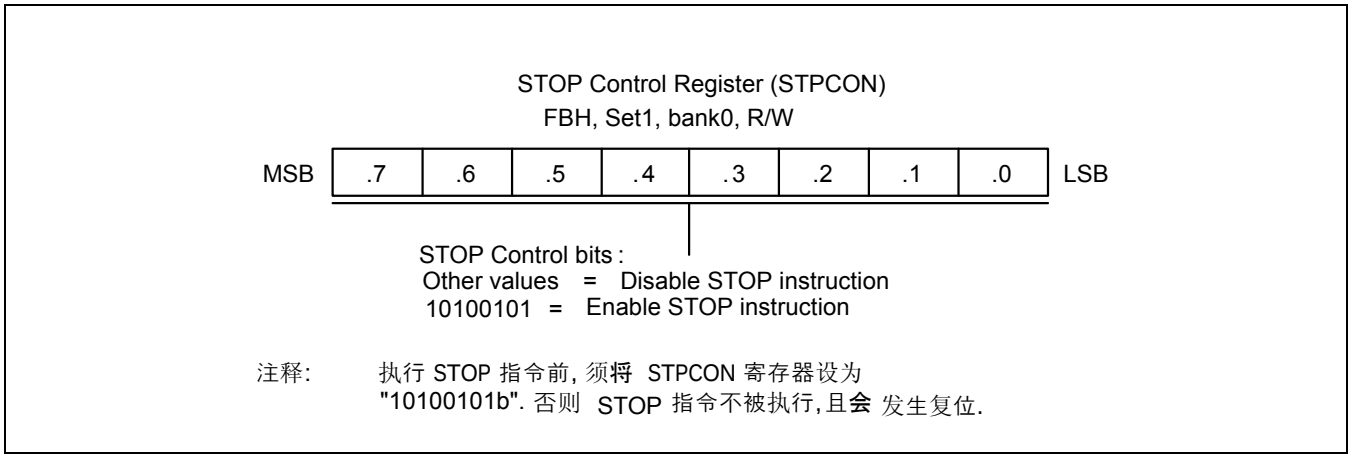


图 7-9 STOP 控制寄存器 (STPCON)

7.1.1.5 CPU 时钟切换

载入振荡器控制寄存器OSCCON 的值，决定了CPU时钟选择主时钟或副时钟；设置CLKCON寄存器，决定时钟分频。这就实现了在主副时钟之间动态切换以及工作频率的改变。

OSCCON.0 为 CPU 时钟选择主时钟 (fx) 或副时钟 (fxt)。OSCCON .3 启动或停止主时钟振荡，OSCCON.2 启动或停止副时钟振荡。CLKCON.4–.3 控制分频电路，将选定的 fxx 时钟1, 2, 8, 16分频。

例如，需要将 CPU 时钟从默认值 fx (普通工作模式，主时钟为fx/16)，切换到副时钟并停止主时钟。为此，可将CLKCON.4-.3 设置为“11”，将OSCCON.0 设置为“1”，同时将 OSCCON.3 设置为“1”。这就把时钟从 fx切换到 fxt，并停止了主时钟振荡。

从副时钟切换到主时钟时，必须遵循以下步骤：首先，将 OSCCON.3 设为“0”以使能主时钟振荡。然后，在特定的机器周期之后，将 OSCCON.0 设置为“0”来选择主时钟。

编程实例 7-1 CPU时钟切换

1. 本例为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

            RET
```

2. 本例为如何从副时钟切换到主时钟：

```
SUB2MA      AND   OSCCON,#07H ; 启动主时钟振荡
            CALL  DLY16       ; 延时 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

上电复位时，V_{DD} 电压上升到高电平而 RESET 管脚被强制拉到低电平。RESET 信号通过一个施密特触发器电路输入，并与 CPU 时钟同步。这个过程将 S3F82HB 带入既定的工作状态。

上电后到误差范围内的最小时间间隔里，RESET 管脚必须保持低电平，来为 CPU 时钟振荡提供稳定时间。振荡稳定所需的最小复位操作时间是1ms。

正常工作状态下 (V_{DD} 和 RESET 都是高电平)，无论何时发生复位，nRESET 管脚都将被强制拉到低电平并启动复位操作。所有的系统和外设控制寄存器将同时被复位成默认值。

总而言之，复位操作时如下事件依次发生：

- 禁止所有中断。
- 看门狗功能启动 (basic timer)。
- P0-10 设为输入模式，所有 I/O 口的上拉电阻禁止。
- 外设控制和数据寄存器设置被禁止，并复位成默认值。
- 程序计数器 (PC) 被置为 ROM 中的程序复位地址 0100H。
- 当设定的振荡稳定时间结束以后，正常模式下 ROM 地址 0100H (以及 0101H) 中的指令被取值和执行。
- ROM 中的复位地址可在 Smart Option 中改变。详情请参考第19章 嵌入式闪存接口。

8.1.2 正常模式复位操作

正常模式下，Test 管脚接到 V_{SS}。复位后，64K 字节片上 ROM 访问被使能 (外部接口非自动配置)。

注释: 为改变振荡稳定时间，可在进入 STOP 模式前，将合适的配置写入 basic timer 控制寄存器 BTCON 中。同样，若不使用 basic timer 的看门狗功能 (当 basic timer 计数器溢出时，系统复位)，可在 BTCON 的高位写入“1010B”来禁止该功能。

8.1.3 硬件复位值

表 8-1，*表 8-2*，*表 8-3*，*表 8-4* 列出了复位操作后 CPU 和系统寄存器，外设控制寄存器，以及外设数据寄存器的复位值。用以下的助记符来代表复位值：

- “1”或“0”代表复位后该位的值相应的为逻辑1或逻辑0。
- “x”代表复位后该位的值不确定。
- 横线（“–”）代表该位未使用或者未映射，但读取的值为0。

表 8-1 S3F82HB Set 1 寄存器及其复位后初始值

寄存器名	助记标号	地址		复位后各位的值							
		Dec	Hex	7	6	5	4	3	2	1	0
地址空间 D0H–D2H 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	0	–	–	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’表示该位复位后的值未定义。
2. 横线（‘-’）表示该位未使用或未映射，但读取的值为“0”。

表 8-2 S3F82HB Set 1, Bank 0 寄存器及其复位后初始值

寄存器名	助记标号	地址		复位后各位的值							
		Dec	Hex	7	6	5	4	3	2	1	0
中断标志位寄存器	INTPND	208	D0H	0	0	0	0	0	0	0	0
钟表定时器控制寄存器	WTCON	209	D1H	0	0	0	0	0	0	0	0
电池电压检测控制寄存器	BLDCON	210	D2H	–	–	0	0	0	0	0	0
SIO 控制寄存器	SIOCON	224	E0H	0	0	0	0	0	0	0	0
SIO 数据寄存器	SIODATA	225	E1H	0	0	0	0	0	0	0	0
SIO 预处理寄存器	SIOPS	226	E2H	0	0	0	0	0	0	0	0
Timer 0 控制寄存器	T0CON	227	E3H	0	0	0	0	0	0	0	0
Timer 0 计数器 (高字节)	T0CNTH	228	E4H	0	0	0	0	0	0	0	0
Timer 0 计数器 (低字节)	T0CNTL	229	E5H	0	0	0	0	0	0	0	0
Timer 0 数据寄存器 (高字节)	T0DATAH	230	E6H	1	1	1	1	1	1	1	1
Timer 0 数据寄存器 (低字节)	T0DATAL	231	E7H	1	1	1	1	1	1	1	1
Timer A 控制寄存器	TACON	232	E8H	0	0	0	0	0	0	0	0
Timer A 计数器	TACNT	233	E9H	0	0	0	0	0	0	0	0
Timer A 数据寄存器	TADATA	234	EAH	1	1	1	1	1	1	1	1
Timer 1 控制寄存器	T1CON	235	EBH	0	0	0	0	0	0	0	0
Timer 1 控制寄存器 (高字节)	T1CNTH	236	ECH	0	0	0	0	0	0	0	0
Timer 1 控制寄存器 (低字节)	T1CNTL	237	EDH	0	0	0	0	0	0	0	0
Timer 1 数据寄存器 (高字节)	T1DATAH	238	EEH	1	1	1	1	1	1	1	1
Timer 1 数据寄存器 (低字节)	T1DATAL	239	EFH	1	1	1	1	1	1	1	1
Timer B 数据寄存器 (高字节)	TBDATAH	240	F0H	1	1	1	1	1	1	1	1
Timer B 数据寄存器 (低字节)	TBDATAL	241	F1H	1	1	1	1	1	1	1	1
Timer B 控制寄存器	TBCON	242	F2H	0	0	0	0	0	0	0	0
频率计数器控制寄存器	FCCON	243	F3H	0	0	0	0	0	0	0	0
频率计数器计数器 (高字节)	FCNTH	244	F4H	0	0	0	0	0	0	0	0
频率计数器计数器 (低字节)	FCNTL	245	F5H	0	0	0	0	0	0	0	0
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
振荡控制寄存器	OSCCON	250	FAH	0	–	–	–	0	0	–	0
STOP 控制寄存器	STPCON	251	FBH	0	0	0	0	0	0	0	0
地址空间 FCH 未映射											
Basic Timer 计数器	BTCNT	253	FDH	0	0	0	0	0	0	0	0
地址空间 FEH 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 S3F82HB Set 1, Bank 1 寄存器及其复位后初始值

寄存器名	助记标号	地址		复位后各位的值							
		Dec	Hex	7	6	5	4	3	2	1	0
P7 口控制寄存器	P7CON	208	D0H	0	0	0	0	0	0	0	0
P8, 9 口控制寄存器	P89CON	209	D1H	0	0	0	0	0	0	0	0
P10 口控制寄存器	P10CON	210	D2H	0	0	0	0	0	0	0	0
P0 口控制寄存器 (高字节)	P0CONH	224	E0H	0	0	0	0	0	0	0	0
P0 口控制寄存器 (低字节)	P0CONL	225	E1H	0	0	0	0	0	0	0	0
P0 口中断控制寄存器 (高字节)	P0INTH	226	E2H	0	0	0	0	0	0	0	0
P0 口中断控制寄存器 (低字节)	P0INTL	227	E3H	0	0	0	0	0	0	0	0
P0 口中断标志位寄存器	P0PND	228	E4H	0	0	0	0	0	0	0	0
P0 口上拉电阻使能控制寄存器	P0PUR	229	E5H	0	0	0	0	0	0	0	0
P1 口控制寄存器 (高字节)	P1CONH	230	E6H	0	0	0	0	0	–	0	0
P1 口控制寄存器 (低字节)	P1CONL	231	E7H	0	0	0	0	0	0	0	0
P2 口控制寄存器 (高字节)	P2CONH	232	E8H	0	0	0	0	0	0	0	0
P2 口控制寄存器 (低字节)	P2CONL	233	E9H	0	0	0	0	0	0	0	0
P3 口控制寄存器 (高字节)	P3CONH	234	EAH	0	0	0	0	0	0	0	0
P3 口控制寄存器 (低字节)	P3CONL	235	EBH	0	0	0	0	0	0	0	0
P3 口中断控制寄存器	P3INT	236	ECH	0	0	0	0	0	0	0	0
P4 口上拉电阻使能控制寄存器	P4PUR	237	EDH	0	0	0	0	0	0	0	0
P4 口控制寄存器 (高字节)	P4CONH	238	EEH	0	0	0	0	0	0	0	0
P4 口控制寄存器 (低字节)	P4CONL	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	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
P5 口数据寄存器	P5	245	F5H	0	0	0	0	0	0	0	0
P6 口数据寄存器	P6	246	F6H	0	0	0	0	0	0	0	0
P7 口数据寄存器	P7	247	F7H	0	0	0	0	0	0	0	0
P8 口数据寄存器	P8	248	F8H	0	0	0	0	0	0	0	0
P9 口数据寄存器	P9	249	F9H	–	–	0	0	0	0	0	0
P10 口数据寄存器	P10	250	FAH	0	0	0	0	0	0	0	0
P5 口上拉电阻使能控制寄存器	P5PUR	251	FBH	0	0	0	0	0	0	0	0
P5 口控制寄存器 (高字节)	P5CONH	252	FCH	0	0	0	0	0	0	0	0
P5 口控制寄存器 (低字节)	P5CONL	253	FDH	0	0	0	0	0	0	0	0
P6 口控制寄存器 (高字节)	P6CONH	254	FEH	0	0	0	0	0	0	0	0
P6 口控制寄存器 (低字节)	P6CONL	255	FFH	0	0	0	0	0	0	0	0

表 8-4 S3F82HB Page 0 寄存器及其复位后初始值

寄存器名	助记标号	地址		复位后各位的值							
		Dec	Hex	7	6	5	4	3	2	1	0
UART 0 控制寄存器 (高字节)	UART0CONH	0	00H	0	0	0	0	0	0	0	0
UART 0 控制寄存器 (低字节)	UART0CONL	1	01H	0	0	0	0	0	0	0	0
UART 0 数据寄存器	UART0DATA	2	02H	x	x	x	x	x	x	x	x
UART 0 波特率数据寄存器	BR0DATA	3	03H	1	1	1	1	1	1	1	1
UART 1 控制寄存器 (高字节)	UART1CONH	4	04H	0	0	0	0	0	0	0	0
UART 1 控制寄存器 (低字节)	UART1CONL	5	05H	0	0	0	0	0	0	0	0
UART 1 数据寄存器	UART1DATA	6	06H	x	x	x	x	x	x	x	x
UART 1 波特率数据寄存器	BR1DATA	7	07H	1	1	1	1	1	1	1	1
比较器0控制寄存器	CMP0CON	8	08H	0	0	0	x	0	0	0	0
比较器1控制寄存器	CMP1CON	9	09H	0	0	0	x	0	0	0	0
A/D 转换数据寄存器 (高字节)	ADDATAH	10	0AH	x	x	x	x	x	x	x	x
A/D 转换数据寄存器 (低字节)	ADDATAL	11	0BH	–	–	–	–	–	–	x	x
A/D 转换控制寄存器	ADCON	12	0CH	–	0	0	0	0	0	0	0
LCD 控制寄存器	LCON	13	0DH	0	0	0	0	0	0	0	0
LCD 对比度控制寄存器	LCONST	14	0EH	0	–	–	–	–	0	0	0
比较器输出控制寄存器	COUTCON	15	0FH	–	–	–	–	–	–	0	0

注释:

1. 'x' 表示该位复位后的值未定义。
2. 横线 ('-') 表示该位未使用或未映射，但读取的值为“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 和所有外设将停止工作。也就是说，片上主振荡器会停止，芯片消耗电流将减小到3μA 以下。
当时钟冻结时，所有的系统功能停止，但存储在内部寄存器卷中的数据仍然保留。可以通过2种方式退出 STOP 模式：复位或中断 (详见图7-6)。

注释: 使用外部时钟源时，不能使用 STOP 模式。因为 X_{IN} 或 XT_{IN} 管脚必须内部连接到 V_{SS} 以减小漏电流。

8.2.1.1 使用 nRESET 退出 STOP 模式

nRESET 信号被释放并恢复高电平时，退出 STOP 模式：所有系统和外设控制寄存器恢复为默认值，数据寄存器仍被保持。因为 CLKCON.3和 CLKCON.4被清除为“00B”，CPU 时钟选择最慢频率 (f_{xx}/16)。振荡稳定时间间隔之后，CPU 调入 ROM 地址0100H (以及0101H) 的指令开始系统初始化。

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以让系统退出 STOP 模式。CPU 当前的内部工作模式决定了退出STOP 可用的中断类型。S3F82HB 中断结构中可用作的退出 STOP 模式的外部中断为：

- 外部中断 P0.0–P0.7, P3.0–P3.3 (INT0–INT11)

关于 STOP 模式的退出请注意以下条件：

- 若使用外部中断退出 STOP 模式，除了 STPCON 寄存器之外的所有系统和外设控制寄存器都不发生改变。
- 若使用内部或外部中断退出 STOP 模式，也可对振荡稳定时间进行编程。但必须在进入 STOP 模式之前将控制和时钟设置成合适的值。
- 当使用外部中断退出 STOP 模式时，CLKCON.4 和 CLKCON.3 的设置不发生改变，当前选择的时钟值也将保留。
- 在退出 STOP 模式时，系统处理外部中断服务程序。中断返回后，执行紧接着 STOP 令的下一条指令。

8.2.1.3 使用内部中断退出STOP 模式

激活任意已使能的中断，可退出 STOP 模式。其他内容和外部中断一样。

如何进入STOP 模式

先写STPCON 寄存器然后写 STOP 指令 (保持这个顺序) 。

```
LD    STPCON,#10100101B
STOP
NOP
NOP
NOP
```

8.2.2 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 停止但某些外设仍然工作。IDLE 模式下，内部时钟信号无法进入 CPU，但所有外设的时钟仍然有效。I/O 口仍然保持进入 IDLE 模式以前的状态(输入或输出)。

有2种方式可以退出 IDLE 模式：

1. 执行复位操作。所有的系统和外设控制寄存器复位至默认值，但所有数据寄存器的值保持不变。因为 CLKCON.3 和 CLKCON.4 清除为“00B”，复位自动选择慢时钟 (f_{xx}/16)。
   如果中断被屏蔽，则复位是退出 IDLE 模式的唯一方法。

2. 激活任意使能的中断，退出 IDLE 模式。当用中断退出 IDLE 模式，CLKCON.3 和 CLKCON.4 寄存器的值保持不变，当前选择的时钟值也不变。系统会先处理中断服务程序。
   中断返回后，执行紧接着 IDLE 指令的下一条指令。

9 I/O 口

9.1 概述

S3F82HB 有11个位可编程的 I/O 口，P0–P10。其中，P1 口是5位口，P9 口是6位口，其他都是8位口。一共有83个 I/O 管脚。每个口都可以灵活配置以满足应用的需要。CPU 通过直接读写口寄存器访问 I/O 口。不需要特别的 I/O 口指令。

表 9-1 给出了S3F82HB I/O 口的功能概述。

表 9-1 S3F82HB 口配置概述

I/O 口	设置选项
0	可对该口的每一位进行功能设定，可以设定为施密特触发器输入，推挽式输出或开漏输出。通过软件设定上拉电阻。P0.0 - P0.7 也可用作外部中断输入 INT0–INT7 (噪声滤波器，中断使能和标志位控制)，BUZ，SI，SCK，SO，和TBPWM。
1	可对该口的每一位进行功能设定，可以设定为施密特触发器输入，推挽式输出或开漏输出。通过软件设定上拉电阻。P1.0–P1.4 也可用作 T1OUT/T1PWM/T1CAP，T0CLK/T1CLK，T0OUT/T0PWM/T0CAP，TxD1，RxD1。
2	可对该口的每一位进行功能设定，可以设定为输入或推挽式输出。通过软件设定上拉电阻。P2.0-P2.7 也可用作AD0–AD7/FCLK，C0OUT，C0P，C0N，C1N，C1P，C1OUT，VBLDREF。
3	可对该口的每一位进行功能设定，可以设定为输入或推挽式输出。通过软件设定上拉电阻。P3.0 - P3.3也可用作外部中断输入 INT8–INT11 (噪声滤波器, 中断使能和标志位控制)，TACLK，TAOUT，TAPWM，TACAP，TxD0，RxD0 或 LCD SEG。
4	可对该口的每一位进行功能设定，可以设定为输入，推挽式输出或开漏输出。通过软件设定上拉电阻。P4.0–P4.7 也可用作 LCD SEG 输出。
5	可对该口的每一位进行功能设定，可以设定为输入，推挽式输出或开漏输出。通过软件设定上拉电阻。P5.0–P5.7 也可用作 LCD SEG 输出。
6	可对该口的每一位进行功能设定，可以设定为输入或推挽式输出。通过软件设定上拉电阻。P6.0–P6.7 也可用作LCD SEG 输出。
7	可对该口2位一组分别进行功能设定，可以设定为输入或推挽式输出。通过软件设定上拉电阻。P7.0–P7.7 也可用作 LCD SEG 输出。
8	可对该口2位或4位一组分别进行功能设定，可以设定为输入或推挽式输出。通过软件设定上拉电阻。P8.0–P8.7 也可用作 LCD SEG 输出。
9	可对该口6位一组分别进行功能设定，可以设定为输入或推挽式输出。通过软件设定上拉电阻。P9.0–P9.5 也可用作 LCD SEG 输出。
10	可对该口1位，2位或4位一组分别进行功能设定，可以设定为输入或推挽式输出。通过软件设定上拉电阻。P10.0–P10.7 也可用作 LCD COM/SEG 输出。

9.1.1 口数据寄存器

表 9-2 总结了 S3F82HB 所有 I/O 口数据寄存器的位置。P0, 1, 2, 3, 4, 5, 6, 7, 8, 9,10 口的数据寄存器大致格式如图 图 9-1 所示。

表 9-2 Port Data Register Summary

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
P5 口数据寄存器	P5	245	F5H	Set 1, Bank 1	R/W
P6 口数据寄存器	P6	246	F6H	Set 1, Bank 1	R/W
P7 口数据寄存器	P7	247	F7H	Set 1, Bank 1	R/W
P8 口数据寄存器	P8	248	F8H	Set 1, Bank 1	R/W
P9 口数据寄存器	P9	249	F9H	Set 1, Bank 1	R/W
P10 口数据寄存器	P10	250	FAH	Set 1, Bank 1	R/W

9.1.2 P0 口

P0 口是一个8位都可单独配置的I/O口。可通过直接读写P0口数据寄存器P0 (地址：F0H set 1, bank 1)来访问P0口。P0.0–P0.7 可用作输入，输出(推挽或开漏)或设置成如下的复用功能：

- 低字节 (P0.0–P0.3)：INT0-INT3/BUZ
- 高字节 (P0.4–P0.7)：INT4-INT7/SI，SCK，SO，TBPWM

9.1.2.1 P0 口控制寄存器 (P0CONH, P0CONL)

P0 有2个8位控制寄存器：控制 P0.4-P0.7的 P0CONH 寄存器和控制 P0.0-P0.3的 P0CONL 寄存器。复位后，P0CONH 和 P0CONL 寄存器被清除为“00H”，所有管脚输入模式。
输入模式下，存在3种不同的选择：

- 施密特触发器输入，信号下降沿触发中断
- 施密特触发器输入，信号上升沿触发中断
- 施密特触发器输入，信号上升/下降沿触发中断

对该口编程时，请记住在 P0口控制寄存器中设置外设 I/O
复用功能的同时，必须在相应的外设模块中使能该功能。

9.1.2.2 P0 口上拉电阻使能寄存器 (P0PUR)

使用 P0口上拉电阻使能寄存器 P0PUR (E5H，set 1，bank 1) 可以设置 P0口各位的上拉电阻。

9.1.2.3 P0 口中断使能和标志位寄存器 (P0INTH, P0INTL, P0PND)

为实现 P0口外部中断，提供下列附加的寄存器：P0中断使能寄存器 P0INTH (高字节，E2H，set 1，bank 1)，P0INTL (低字节，E3H，set1，bank1) 和 P0中断标志寄存器 P0PND (E4H，set 1，bank 1)。

P0 中断标志寄存器 P0PND 可以检查中断挂起状态并且当中断服务程序进入之后清除标志位。
应用程序在一定的时间间隔查询标志位以检测中断请求。

当 P0口任意位的中断使能位是“1”时，该管脚上信号的上升或下降沿触发中断请求。相应的 P0PND 位也自动置“1”，同时 IRQ 也变成低电平告知 CPU 有中断请求在等待。
当 CPU 响应中断请求时，应用软件必须通过在 P0PND 寄存器的相应位写“0”来清除标志位状态。

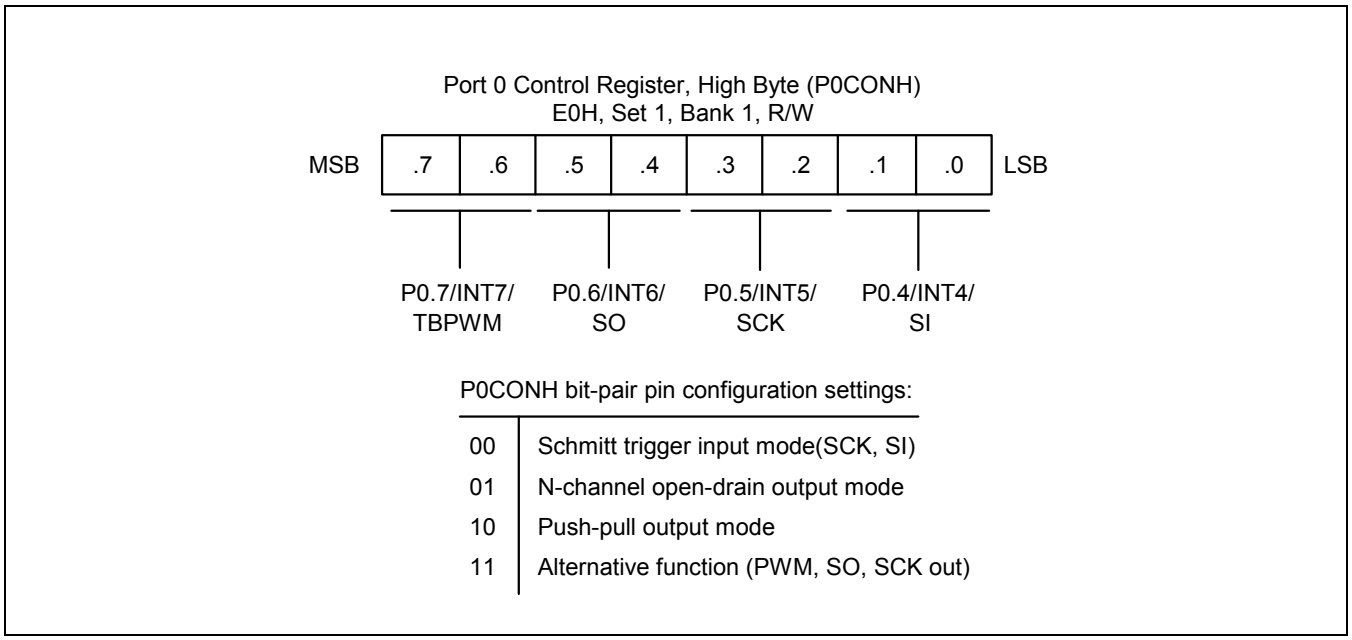


图 9-1 P0 口高字节控制寄存器 (P0CONH)

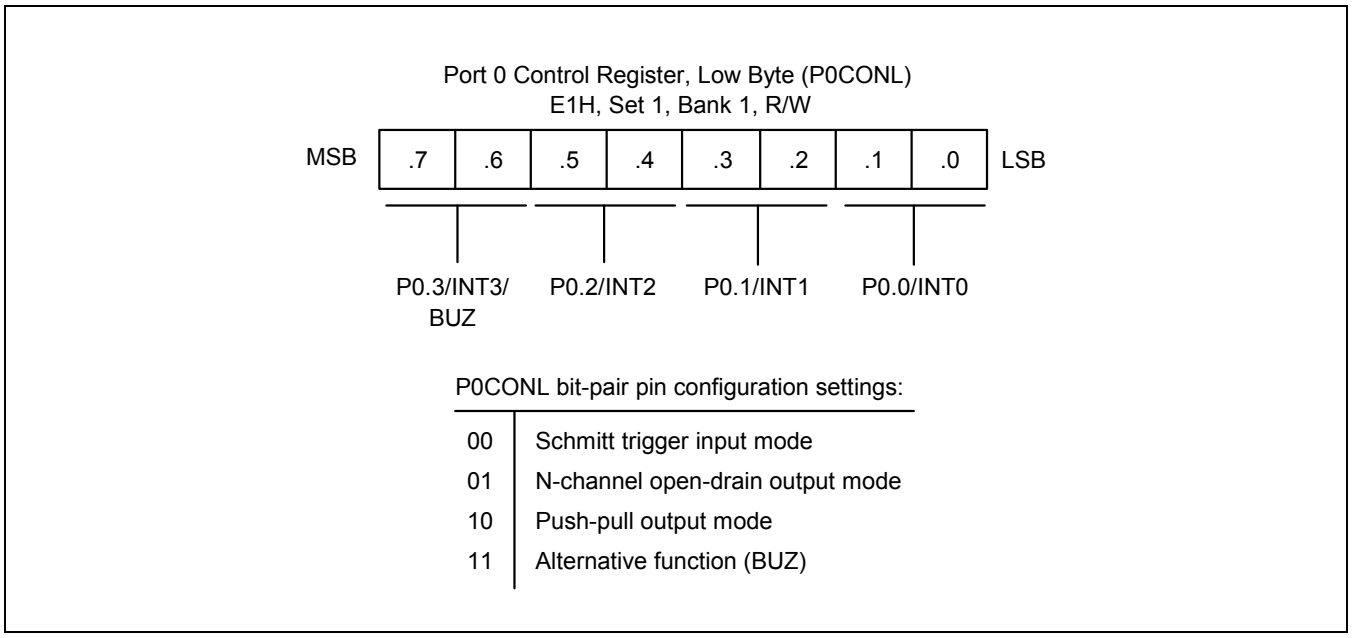
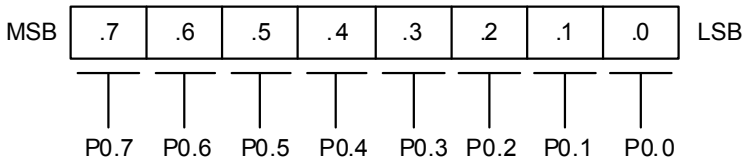


图 9-2 P0 口低字节控制寄存器 (P0CONL)

Port 0 Pull -up Resistor Enable Register (P0PUR)
E5H, Set 1, Bank 1, R/W



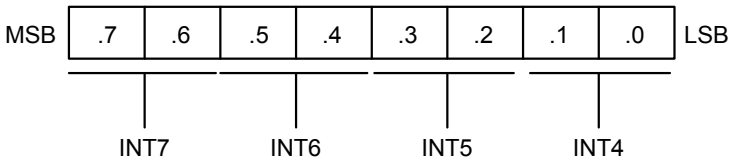
P0PUR bit configuration settings:

0	Disable Pull - up Resistor
1	Enable Pull - up Resistor

注释： P0口的上拉电阻自动禁止仅当相应管脚被设置成推挽输出或复用功能。

图 9-3 P0 口上拉电阻使能寄存器 (P0PUR)

Port 0 Interrupt Control Register, High Byte (P0INTH)
E2H, Set 1, Bank 1, R/W

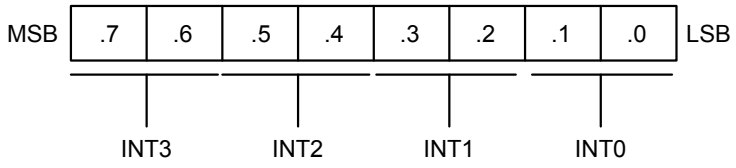


P0INTH bit-pair pin configuration settings:

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 9-4 P0 口高字节中断控制寄存器 (P0INTH)

Port 0 Interrupt Control Register, Low Byte (P0INTL)
E3H, Set 1, Bank 1, R/W

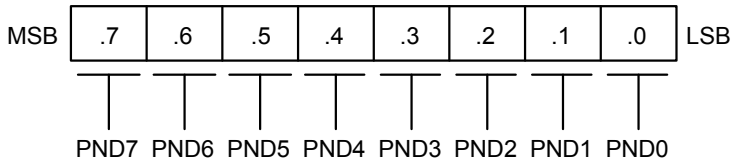


P0INTL bit-pair pin configuration settings:

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 9-5 P0 口低字节中断控制寄存器 (P0INTL)

Port 0 Interrupt Pending Register (P0PND)
E4H, Set 1, Bank 1, R/W



P0PND bit configuration settings:

0	Interrupt request is not pending, pending bit clear when write 0
1	Interrupt request is pending

图 9-6 P0 口中断标志寄存器 (P0PND)

9.1.3 P1 口

P1 口是一个5位都可单独配置的I/O口。可通过直接读写P1口数据寄存器P1 (地址：F1H set 1, bank 1)来访问P1口。P1.0–P1.4 可用作输入，输出(推挽或开漏)或设置成如下的复用功能：

- 低字节管脚 (P1.0-P1.3)：T1OUT/T1PWM/T1CAP，T0CLK/T1CLK，T0OUT/T0PWM/T0CAP，TxD1
- 高字节管脚 (P1.4)：RxD1

9.1.3.1 P1 口控制寄存器 (P1CONH, P1CONL)

P1 有2个8位控制寄存器：控制 P1.4的 P1CONH 寄存器和控制 P1.0-P1.3的 P1CONL 寄存器。复位后，P1CONH 和 P1CONL 寄存器被清除为“00H”，所有管脚输入模式。
使用控制寄存器设置来选择输入或输出模式 (推挽或开漏)，使能上拉电阻以及复用功能。

对该口编程时，请记住在 P1口控制寄存器中设置外设 I/O 复用功能的同时，必须在相应的外设模块中使能该功能。

Port Control Register, High Byte (P1 CONH)
E6H, Set 1, Bank 1, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB
	P1.4 PUR	P1.3 PUR	P1.2 PUR	P1.1 PUR	P1.0 PUR		P1.4/ RxD 1		

Not used for S3F82HB

P1 PUR bit configuration settings

0	Disable Pull - up Resistor
1	Enable Pull - up Resistor

P1 CONH bit -pair pin configuration settings :

00	Schmitt trigger input mode (RxD 1)
01	N - channel open - drain output mode
10	Push - pull output mode
11	Alternative function (RxD 1 out)

注释 P1口的上拉电阻被自动禁止仅当相应位被设置为推挽输出或复用功能。

图 9-7 P1 高字节控制寄存器 (P1CONH)

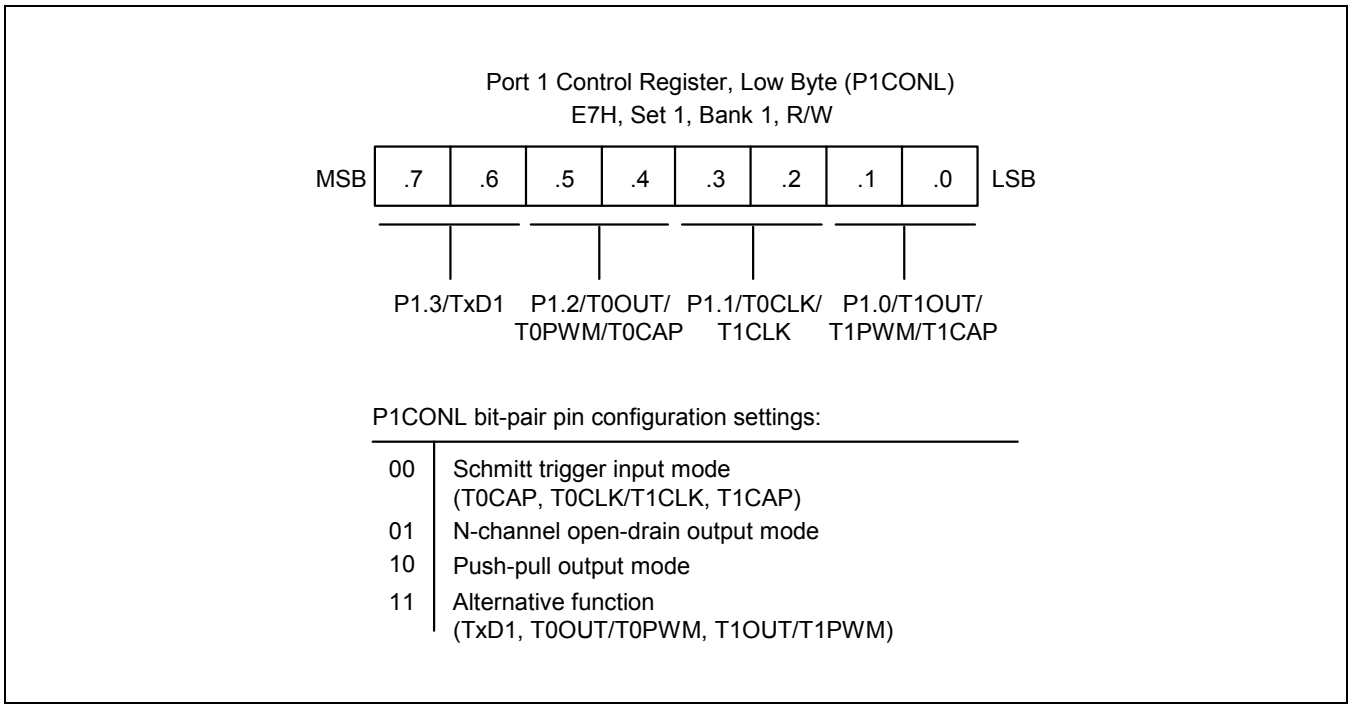


图 9-8 P1 低字节控制寄存器 (P1CONL)

9.1.4 P2 口

P2 口是一个8位 I/O 口，可用作通用 I/O 或 A/D 转换器输入，AD0–AD7。可通过直接读写 P2口数据寄存器 P2 (地址：F2H set 1, bank 1) 来访问 P2口。P2.0–P2.7 可用作输入，推挽输出或设置成如下的复用功能：

- 低字节 (P2.0–P2.3)：AD0–AD3/FCLK, C0OUT, C0P, C0N
- 高字节 (P2.4–P2.7)：AD4–AD7/C1N, C1P, C1OUT, V_{BLDREF}

9.1.4.1 P2 口控制寄存器 (P2CONH, P2CONL)

P2 有2个8位控制寄存器：控制 P2.4-P2.7的 P2CONH 寄存器和控制 P2.0-P2.3的 P2CONL 寄存器。
复位后，P2CONH 和 P2CONL 寄存器被清除为“00H”，所有管脚输入模式。
使用控制寄存器设置来选择输入或输出模式 (推挽)，使能上拉电阻以及复用功能。

对该口编程时，请记住在 P2口控制寄存器中设置外设 I/O 复用功能的同时，必须在相应的外设模块中使能该功能。

Port 2 Control Register, High Byte (P2CONH)
E8H, Set 1, Bank 1, R/W

MSB	.7	.6	.5	.4	.3	.2	.1	.0	LSB

P2.7/AD7/ V_{BLDREF}	P2.6/AD6/ C1OUT	P2.5/AD5/ C1P	P2.4/AD4/ C1N

P2CONH bit-pair pin configuration settings:

00	Input mode
01	Input mode with pull-up resistor
10	Push-pull output mode
11	Alternative function (AD7-AD4/V_{BLDREF}, C1OUT, C1P, C1N)

注释： 关于P2.7和P2.4-P2.6的复用功能，请参考电池电压检测控制寄存器 (BLDCON) 和 comparator 1 控制寄存器 (CMP1CON)。

图 9-9 P2 口高字节控制寄存器 (P2CONH)

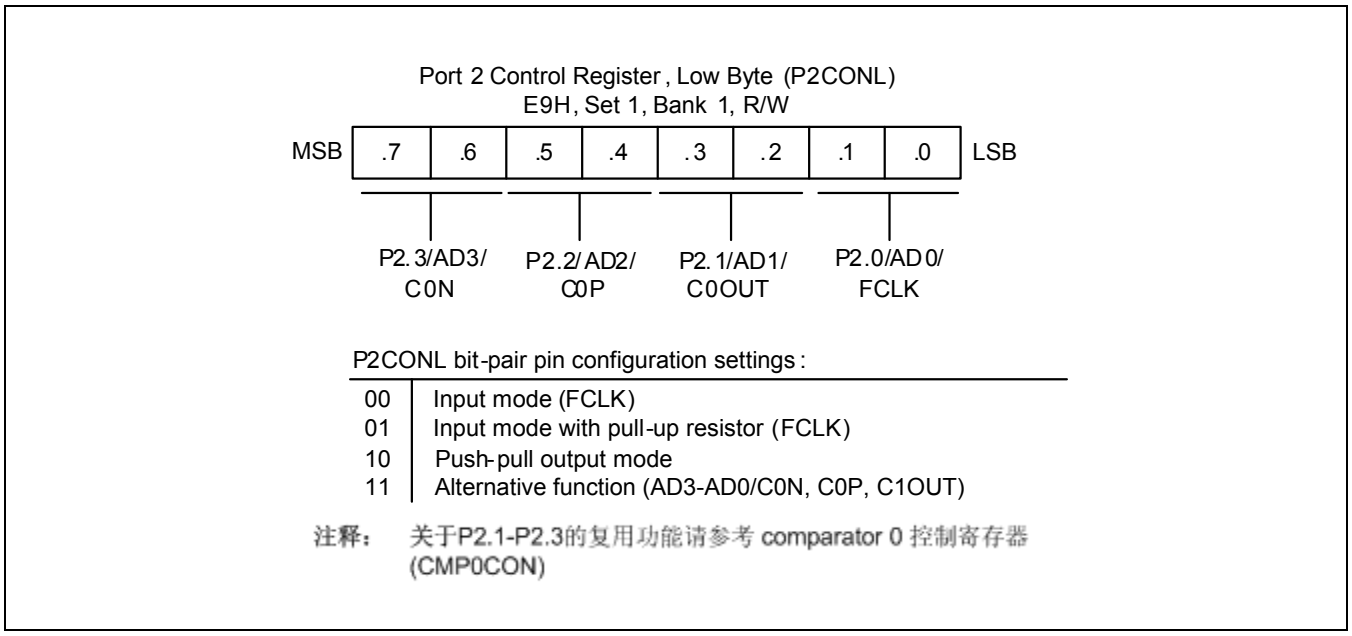


图 9-10 P2 口低字节控制寄存器 (P2CONL)

9.1.5 P3 口

P3 口是一个8位都可单独配置的I/O口。可通过直接读写P3口数据寄存器P3 (地址：F3H set 1, bank 1)来访问P3口。P3.0–P3.7 可用作输入，推挽输出或设置成如下的复用功能：

- 低字节管脚 (P3.0-P3.3)：INT11-INT8/SEG52-SEG55
- 高字节管脚 (P3.4–P3.7)：SEG56-SEG57/TACLK, TAOUT/TAPWM/TACAP, TxD0, RxD0

9.1.5.1 P3 口控制寄存器 (P3CONH, P3CONL)

P3 有2个8位控制寄存器：控制 P3.4-P3.7的 P3CONH 寄存器和控制 P3.0-P3.3的 P3CONL 寄存器。复位后，P3CONH 和 P3CONL 寄存器被清除为“00H”，所有管脚输入模式。
使用控制寄存器设置来选择输入或输出模式 (推挽)，使能上拉电阻以及复用功能。

对该口编程时，请记住在 P3口控制寄存器中设置外设 I/O 复用功能的同时，必须在相应的外设模块中使能该功能。

9.1.5.2 P3 口中断使能寄存器 (P3INT)

为实现 P3口外部中断，提供下列附加的寄存器：P3 中断使能寄存器 P3INT (ECH, set 1, bank 1)。

INT8-INT11 的挂起状态由硬件自动清除。

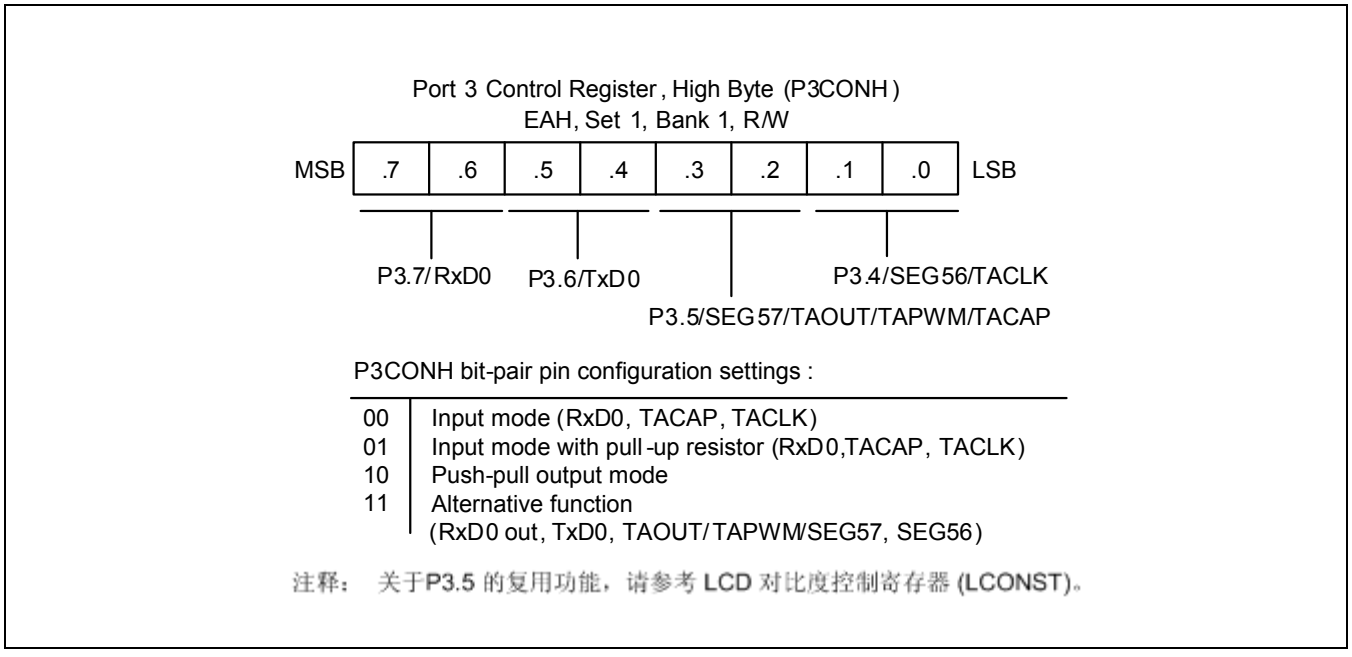


图 9-11 P3 高字节控制寄存器 (P3CONH)

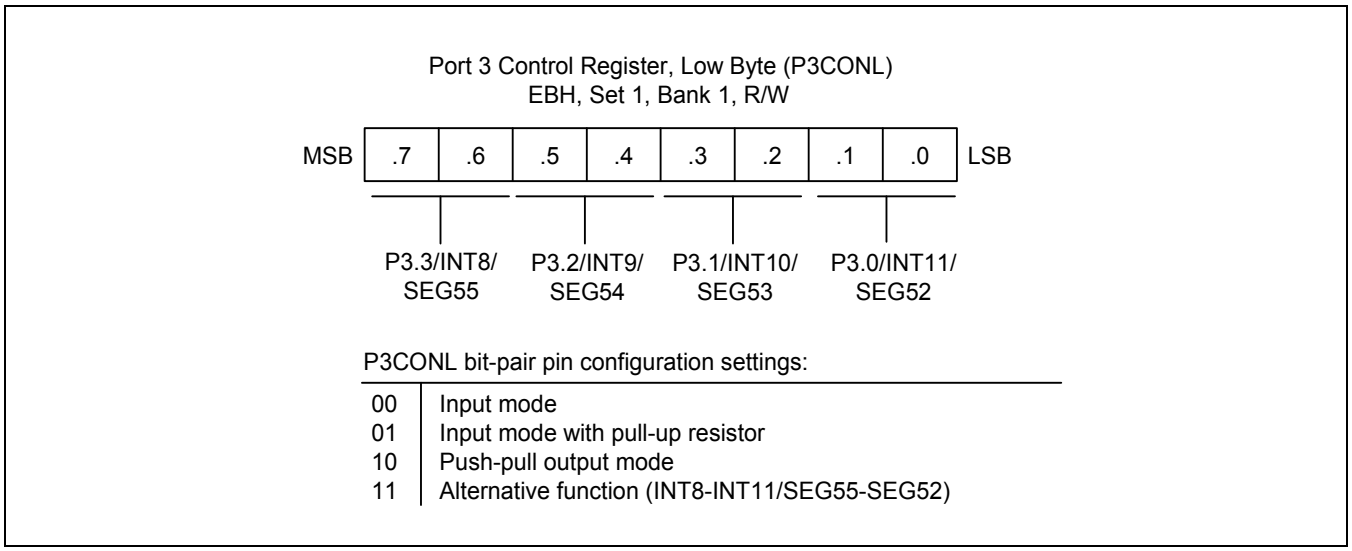


图 9-12 P3 低字节控制寄存器 (P3CONL)

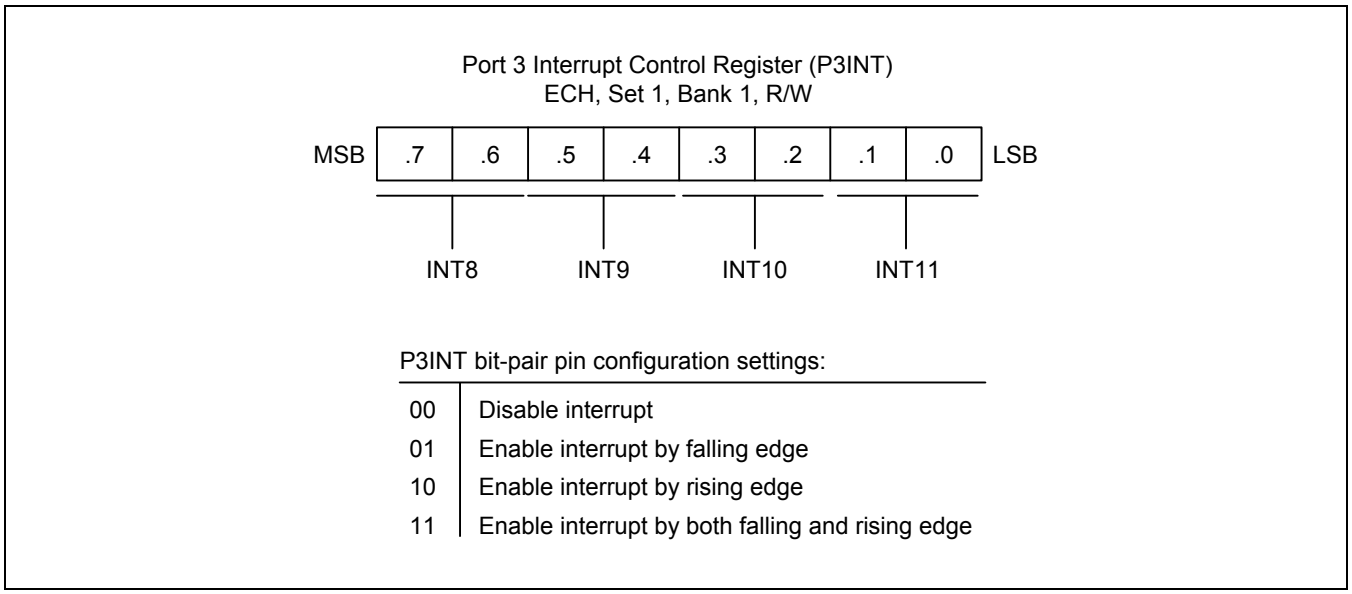


图 9-13 P3 中断控制寄存器 (P3INT)

9.1.6 P4 口

P4 口是一个8位都可单独配置的 I/O 口。可通过直接读写 P4口数据寄存器 P4 (地址：F4H set 1, bank 1) 来访问P4口。P4.0–P4.7 可用作输入 (带或不带上拉) 和输出 (推挽或开漏)。也可用作 LCD 模块的段。

9.1.6.1 P4 口控制寄存器 (P4CONH, P4CONL)

P4 有2个8位控制寄存器：控制 P4.4-P4.7的 P4CONH 寄存器和控制 P4.0-P4.3的 P4CONL 寄存器。
复位后，P4CONH 和 P4CONL 寄存器被清除为“00H”，所有管脚输入模式。

9.1.6.2 P4 口上拉电阻使能寄存器 (P4PUR)

使用 P4口上拉电阻使能寄存器 P4PUR (EDH, set 1, bank 1) 可以设置 P4口各位的上拉电阻。

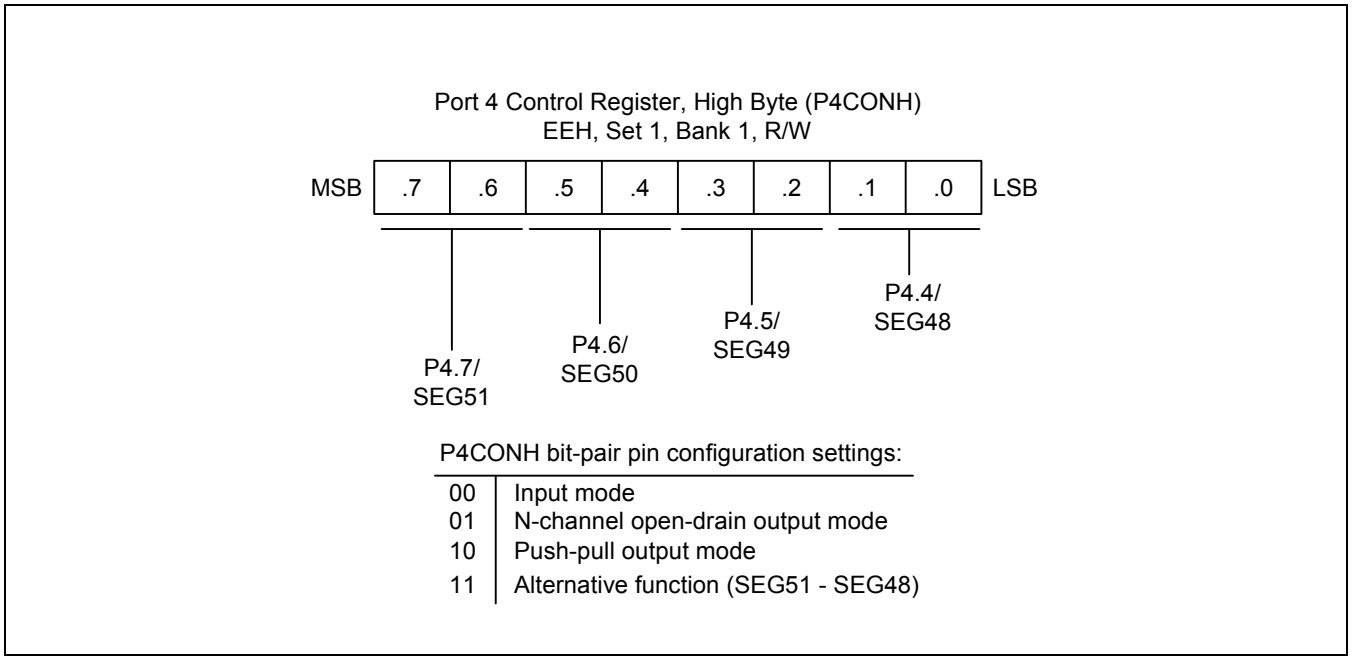
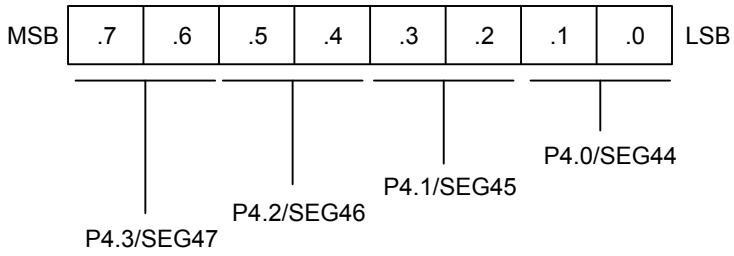


图 9-14 P4 口高字节控制寄存器 (P4CONH)

Port 4 Control Register, Low Byte (P4CONL)
EFH, Set 1, Bank 1, R/W

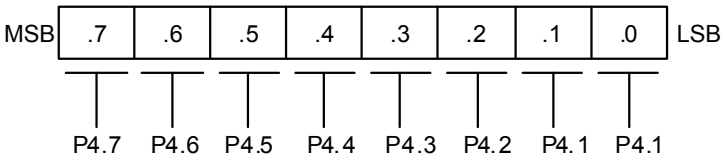


P4CONL bit-pair pin configuration settings:

00	Input mode
01	N-channel open-drain output mode
10	Push-pull output mode
11	Alternative function (SEG47-SEG44)

图 9-15 P4 口低字节控制寄存器 (P4CONL)

Port 4 Pull-up Resistor Enable Register (P4PUR)
EDH, Set 1, Bank 1, R/W



P4PUR bit configuration settings:

0	Disable Pull-up Resistor
1	Enable Pull-up Resistor

注释： P4 口的上拉电阻自动禁止，仅当相应的管脚被设置为推挽输出或复用功能。

图 9-16 P4 口上拉电阻使能寄存器 (P4PUR)

9.1.7 P5 口

P5 口是一个8位都可单独配置的 I/O 口。可通过直接读写 P5口数据寄存器P5 (地址：F5H set 1, bank 1) 来访问P5口。P5.0–P5.7 可用作输入 (带或不带上拉) 和输出 (推挽或开漏)。也可用作 LCD 模块的段。

9.1.7.1 P5 口控制寄存器 (P5CONH, P5CONL)

P5 有2个8位控制寄存器：控制 P5.4-P5.7的 P5CONH 寄存器和控制 P5.0-P5.3的 P5CONL 寄存器。复位后，P5CONH 和 P5CONL 寄存器被清除为“00H”，所有管脚输入模式。

9.1.7.2 P5 口上拉电阻使能寄存器 (P5PUR)

使用 P5口上拉电阻使能寄存器 P5PUR (FBH, set 1, bank 1) 可以设置 P5口各位的上拉电阻。

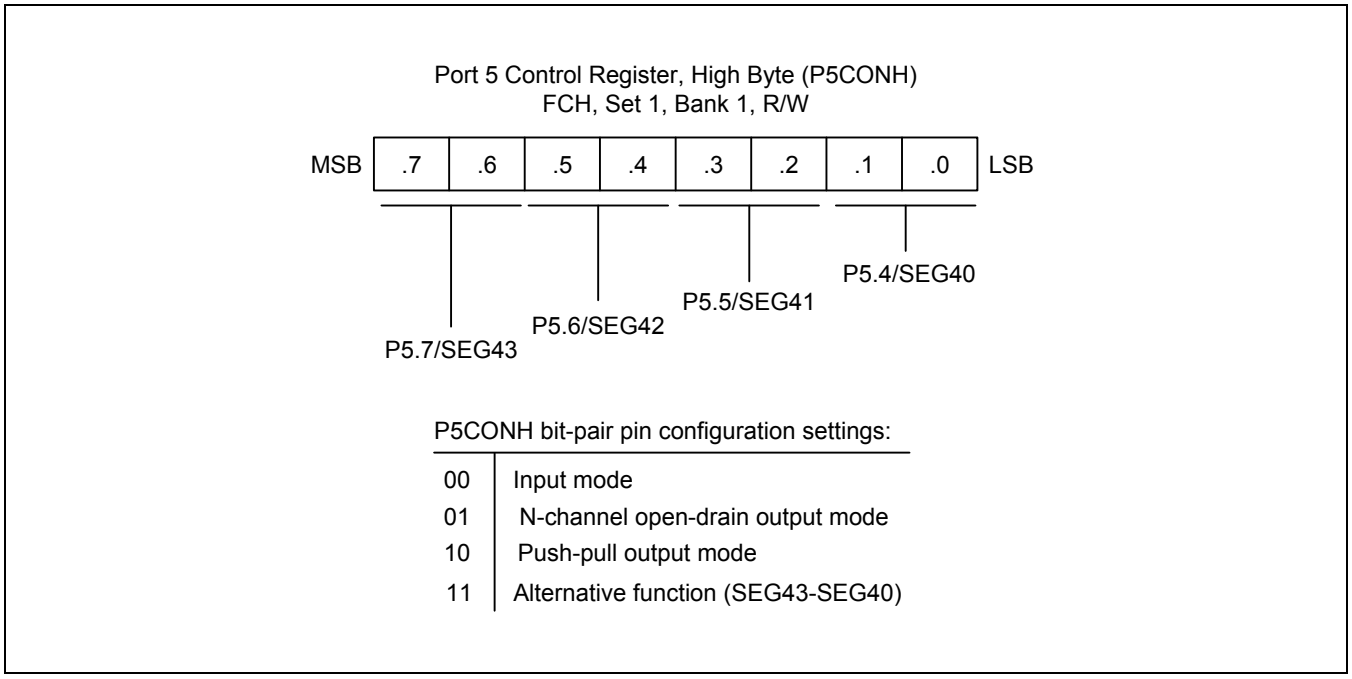


图 9-17 P5 口高字节控制寄存器 (P5CONH)

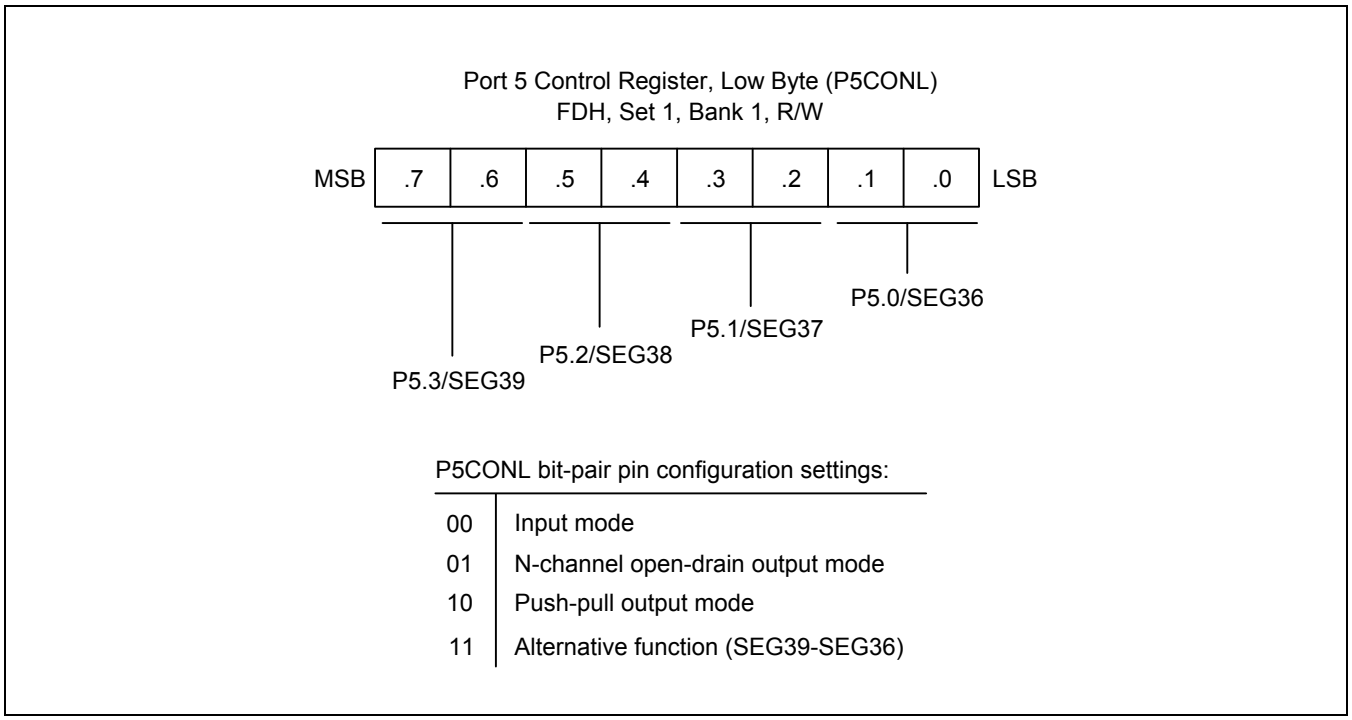
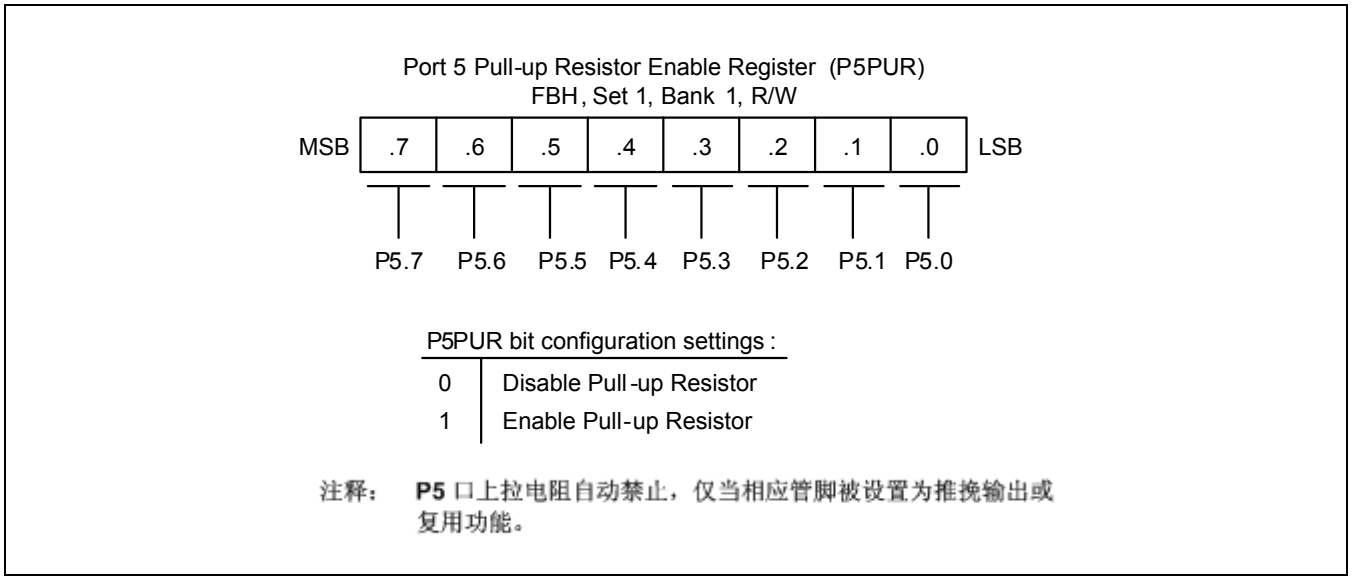


图 9-18 P5 口低字节控制寄存器 (P5CONL)



注释： **P5** 口上拉电阻自动禁止，仅当相应管脚被设置为推挽输出或复用功能。

图 9-19 P5 口上拉电阻使能寄存器 (P5PUR)

9.1.8 P6 口

P6 口是一个8位都可单独配置的 I/O 口。可通过直接读写P6口数据寄存器 P6 (地址：F6H set 1, bank 1) 来访问P6口。P6.0–P6.7 可用作输入 (带或不带上拉) 和输出 (推挽或开漏)。也可用作 LCD 模块的段。

9.1.8.1 P6 口控制寄存器 (P6CONH, P6CONL)

P6 有2个8位控制寄存器：控制 P6.4-P6.7的 P6CONH 寄存器和控制 P6.0-P6.3的 P6CONL 寄存器。复位后，P6CONH 和 P6CONL 寄存器被清除为“00H”，所有管脚输入模式。

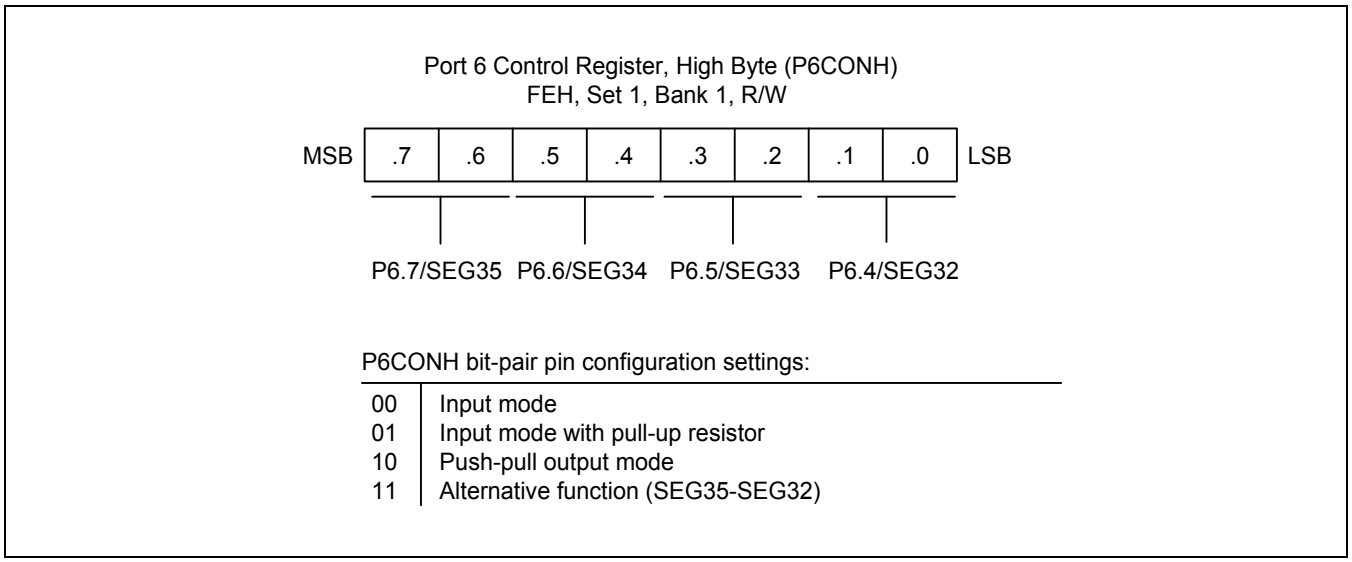


图 9-20 P6 口高字节控制寄存器 (P6CONH)

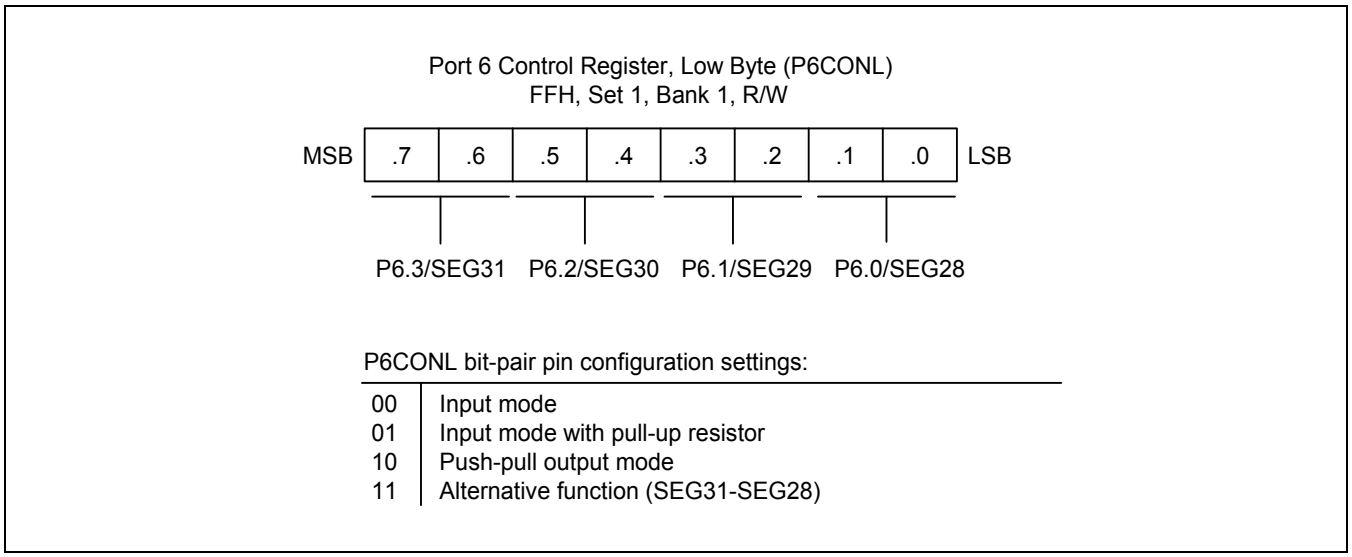


图 9-21 P6 口低字节控制寄存器 (P6CONL)

9.1.9 P7 口

P7 口是一个可2个一组配置的8位 I/O 口。可通过直接读写 P7口数据寄存器 P7 (地址：F7H set 1, bank 1) 来访问 P7口。P7.0–P7.7 可用作输入 (带或不带上拉) 和推挽输出。也可用作 LCD 模块的段。

9.1.9.1 P7 口控制寄存器 (P7CON)

P7 有1个8位控制寄存器：控制 P7.0-P7.7的 P7CON 寄存器。复位后，P7CON 寄存器被清除为 “00H”，所有管脚输入模式。

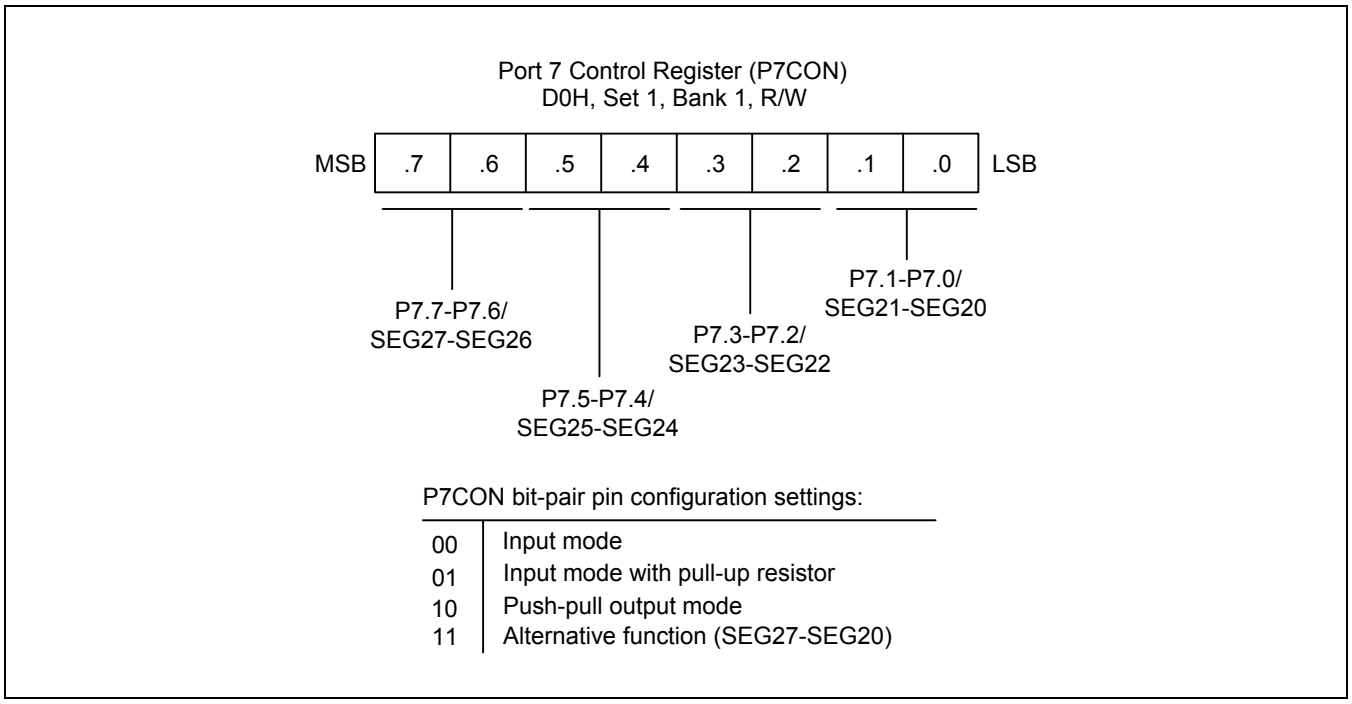


图 9-22 P7 口控制寄存器 (P7CON)

9.1.10 P8, 9 口

P8 口是一个可2个一组或4个一组配置的8位 I/O 口。可通过直接读写P8口数据寄存器 P8 (地址：F8H set 1, bank 1) 来访问 P8口。P9 口是一个可6个一组配置的6位 I/O 口。可通过直接读写 P9口数据寄存器 P9 (地址：F9H set 1, bank 1) 来访问 P9口。P8.0–P8.7和 P9.0–P9.5可用作输入 (带或不带上拉) 和推挽输出。也可用作LCD模块的段。

9.1.10.1 P8, 9 口控制寄存器 (P89CON)

P8，9有1个8位控制寄存器：控制 P8.0-P8.7的 P89CON 寄存器。复位后，P89CON 寄存器被清除为“00H”，所有管脚输入模式。

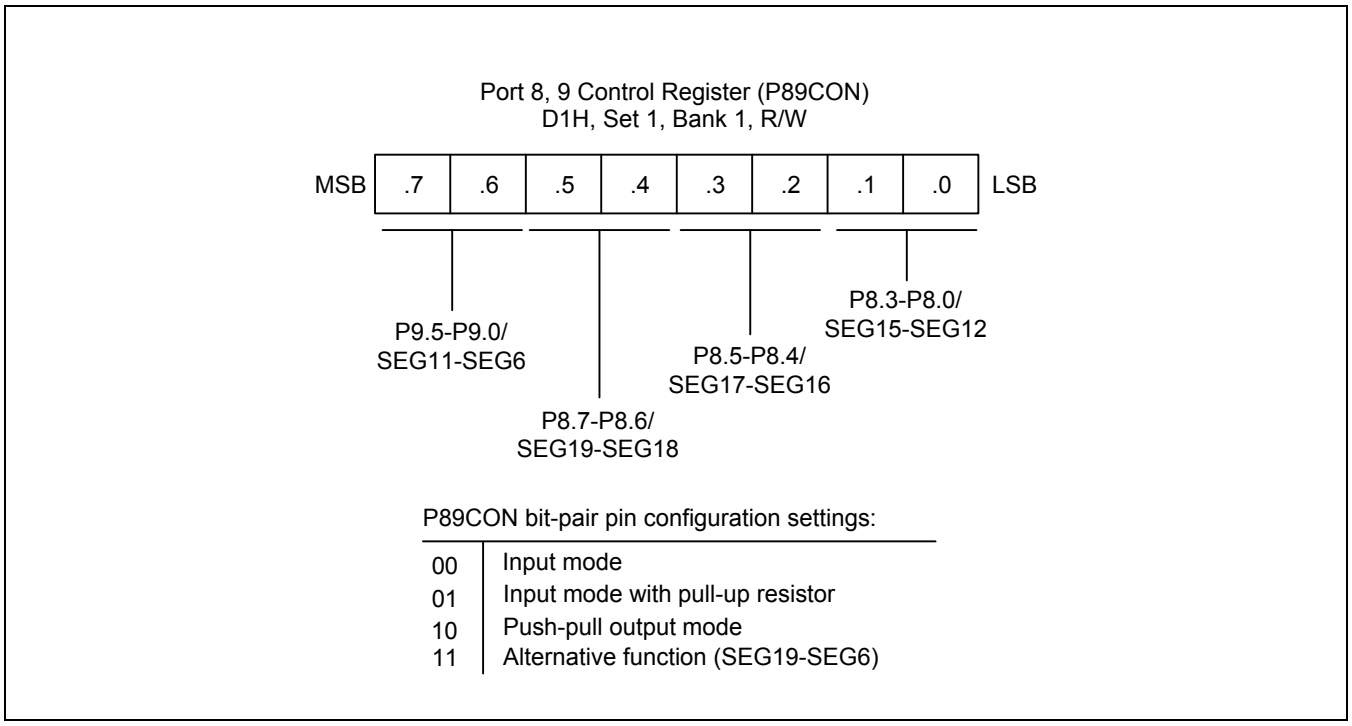


图 9-23 P8, 9 口控制寄存器 (P89CON)

9.1.11 P10 口

P10 口是一个可2个一组或4个一组或单个位配置的8位 I/O 口。可通过直接读写 P10口数据寄存器 P10 (地址：FAH set 1, bank 1) 来访问 P10口。P10.0–P10.7可用作输入 (带或不带上拉) 和推挽输出。也可用作 LCD 模块的段。

9.1.11.1 P10 口控制寄存器 (P10CON)

P10 口有1个8位控制寄存器：控制 P10.0-P10.7的 P10CON 寄存器。复位后，P10CON 寄存器被清除为“00H”，所有管脚输入模式。

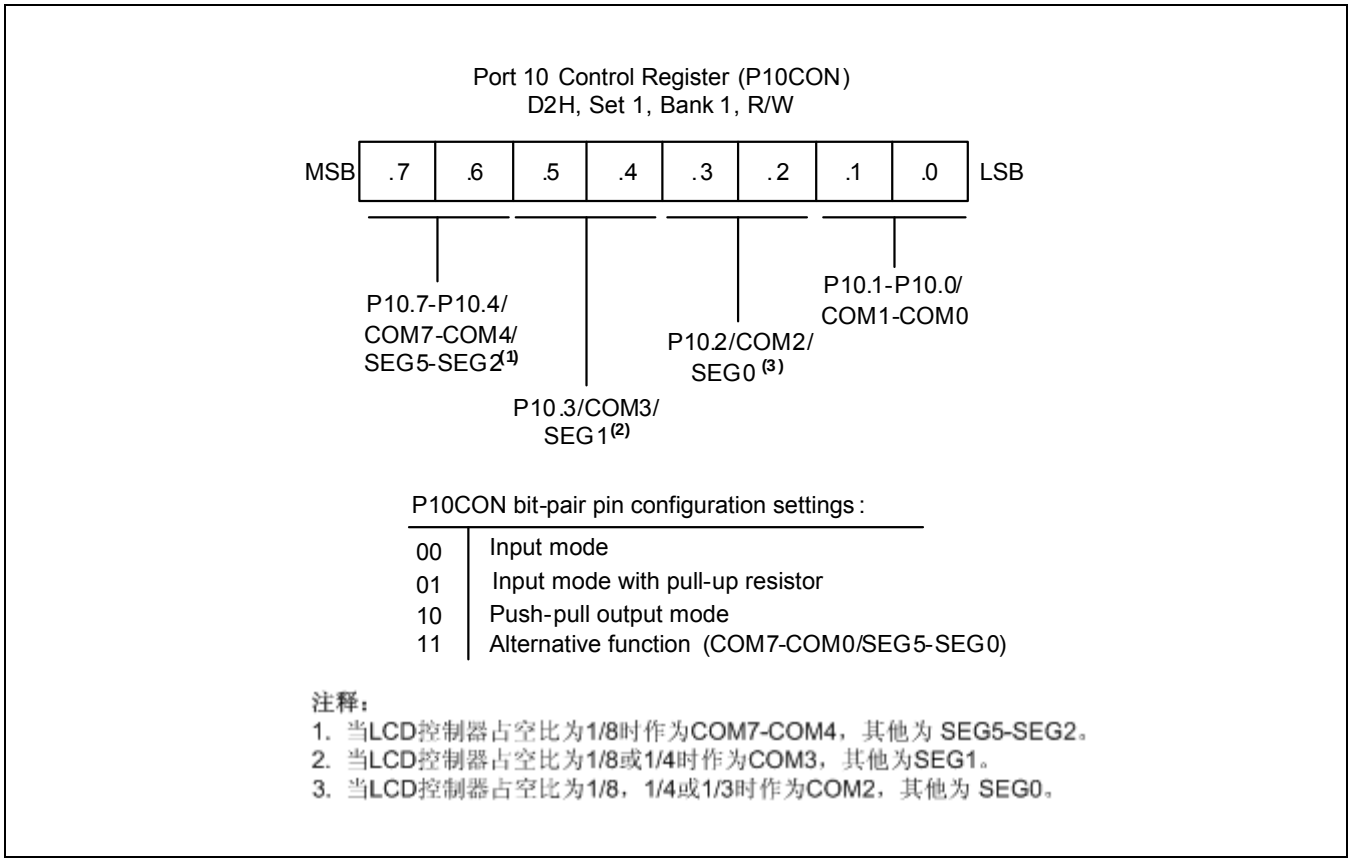


图 9-24 P10 口控制寄存器 (P10CON)

10 BASIC TIMER

10.1 概述

S3F82HB 内置一个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：

- 时钟分频器 (fosc/4096/1024/128/16)
- 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)

10.1.2 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 BT 计数器和分频器，禁止或使能看门狗功能。它的物理地址位于D3H，set 1，可通过寄存器寻址模式进行读/写。

复位后，BTCON 被清零“00H”，该选项将使能看门狗功能，并选择 BT 的时钟为 fxx/4096。
如果想禁用看门狗功能，必须将 Basic Timer 控制寄存器的高四位 BTCON.7–BTCON.4 设置为“1010B”。

在正常操作情况下，可以写“1”到 BTCON.1 来清零 8 位 Basic Timer 计数器 BTCNT (set 1, bank 0, FDH)。也可以写“1”到 BTCON.0 位，来清除 Basic Timer 输入时钟的分频器。

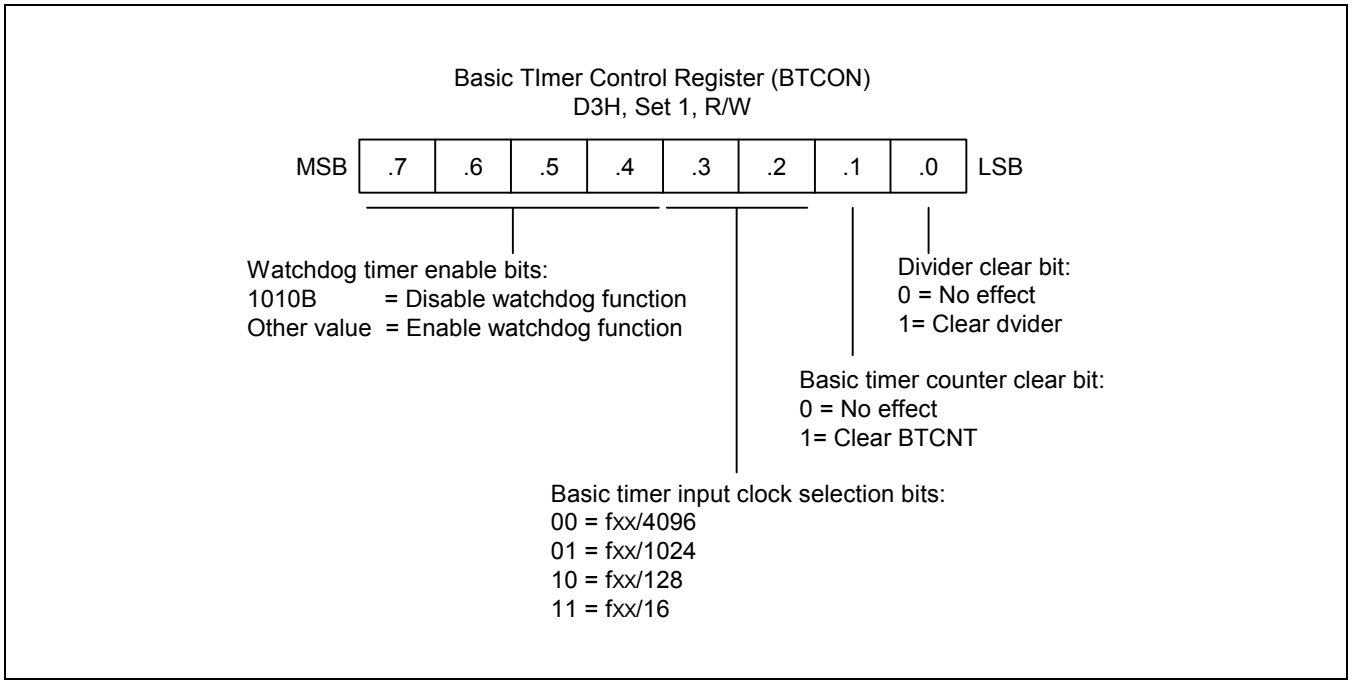


图 10-1 Basic Timer 控制寄存器 (BTCON)

10.1.3 BASIC TIMER 功能描述

10.1.3.1 看门狗定时器功能

可通过向 BTCON.7–BTCON.4 写入非“1010B”的任意值 (“1010B”将禁用看门狗功能)，来对 basic timer 溢出信号 (BTOVF) 进行编程控制，以产生系统复位。复位时 BTCON 清零，自动允许看门狗功能。同时选择 Basic Timer 的时钟信号为 fxx/4096。

无论什么时候 BT
计数器溢出，都将产生复位。正常情况下，应用程序须防止计数器溢出，为此，每隔一段时间要清除看门狗计数器 (写“1”到 BTCON.1)。

如果由于电路噪声或其它原因造成系统失灵，Basic Timer 的清零操作不会被执行，看门狗将复位芯片。
换言之，在正常运行期间，必须通过 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，自动产生复位。

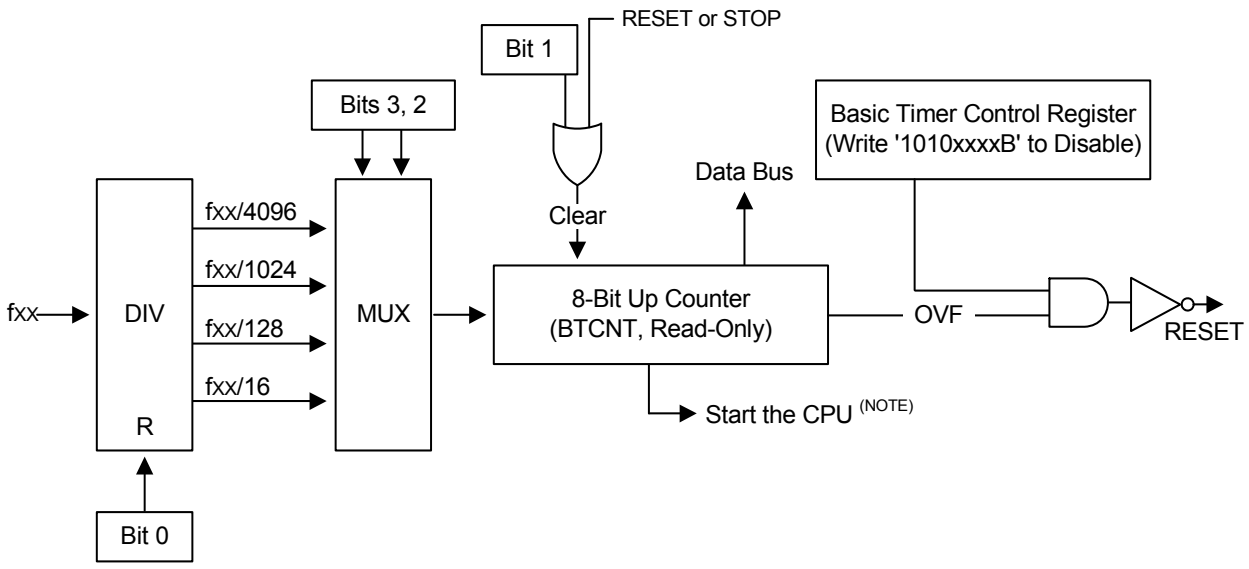
10.1.3.2 振荡稳定功能

还可以用 Basic Timer 对复位或 STOP 模式唤醒时的振荡稳定时间进行定时。

在 STOP 模式下，每当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 溢出时，
将产生一个信号表明振荡稳定时间到，CPU开始在稳定的时钟下开始工作。

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，系统从 STOP 模式唤醒，振荡器起振。
2. 如果是复位操作，BTCNT 将以默认的时钟频率 fxx/4096 进行计数。
   如果是外部中断，BTCNT 寄存器按照预设时钟频率计数。
3. 振荡稳定间隔计时开始，直到 BTCNT 第4位溢出为止。
4. 当BTCNT.4 溢出发生时，CPU 开始正常工作。



注释: 上电复位期间，CPU在振荡稳定时间到来之前处于IDLE状态，直到basic timer计数器第4位溢出。

图 10-2 Basic Timer 方框图

11 8 位 TIMER A/B

11.1 8 位 TIMER A

11.1.1 概述

Timer A 是一个8位通用 Timer/Counter。它有三种工作模式，可通过配置寄存器 TACON 来选择：

- Interval (定时) 模式 (TAOUT 管脚反转输出)
- Capture (捕获) 模式，TACAP 管脚上升沿或下降沿触发
- PWM 模式 (TAPWM)

Timer A 包括以下几个功能模块：

- 带多路复用的时钟分频器 (fxx/1024/256/64/8/1)
- 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)
- 用于捕获输入 (TACAP)，PWM 输出或匹配输出 (TAPWM, TAOUT) 的I/O管脚
- Timer A 溢出中断 (IRQ2，中断向量地址：D2H) 和匹配/捕获中断 (IRQ2，中断向量地址：D0H)
- Timer A 控制寄存器 TACON (地址：E8H，Set 1，Bank 0，可读/写)

11.1.2 TIMER A 控制寄存器 (TACON)

Timer A 控制寄存器 TACON 可用于：

- 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
- 清零 Timer A 计数器 TACNT
- 使能 Timer A 溢出中断或 Timer A 匹配/捕获中断

TACON 地址为：E8H，Set 1，Bank 0。可读/写，支持寄存器寻址模式。

复位时，清零 TACON，将会把 Timer A 设置为定时模式，输入时钟频率选择为 fxx/1024，并禁止所有 Timer A 中断。正常工作时，可随时通过向 TACON.2 位写“1”来清零 Timer A 计数器。

Timer A 溢出中断 (TAOVF) 的优先级为 IRQ2，中断向量地址为 D2H。当 Timer A 溢出中断发生后，CPU 响应此中断，中断标志位被硬件自动清零或必须由软件清零。

将 TACON.1 位置“1”可使能 Timer A 匹配/捕获中断 (IRQ2，中断向量地址：D0H)。
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.1 位。检测到“1”时，Timer A 的匹配/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 A 的匹配/捕获中断标志位 INTPND.1。

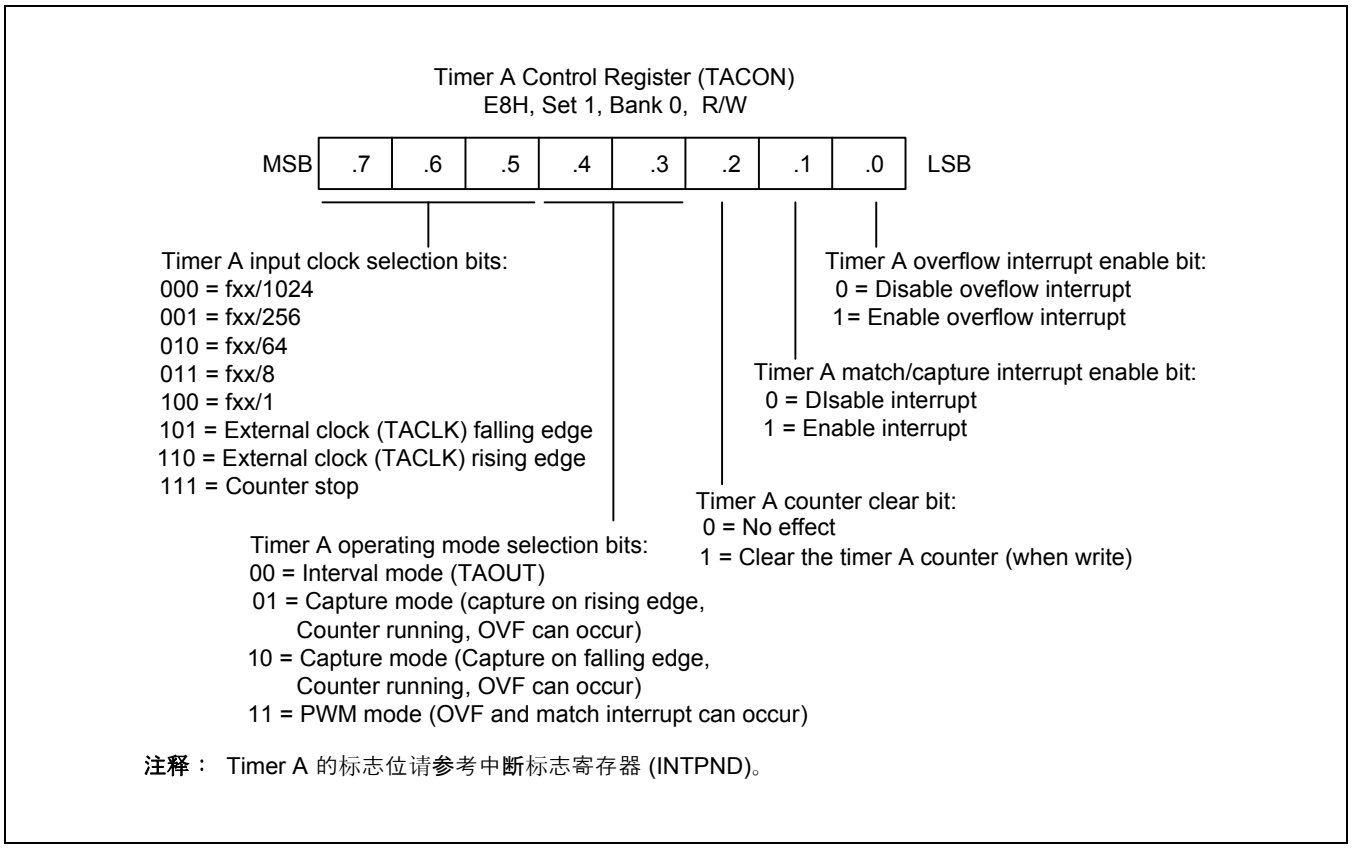


图 11-1 Timer A 控制寄存器 (TACON)

11.1.3 TIMER A 功能描述

11.1.3.1 Timer A 中断 (IRQ2, 中断向量地址：D0H 和 D2H)

Timer A 模块能产生2种中断：Timer A 溢出中断 (TAOVF)，以及 Timer A 匹配/捕获中断 (TAINT)。TAOVF 的中断优先级为 IRQ2，中断向量地址为 D2H。TAINT 的中断优先级也为 IRQ2，不同的是中断向量地址为 D0H。

当 Timer A 溢出中断发生后，CPU响应此中断，中断标志位被硬件自动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.0。但 Timer A 匹配/捕获中断标志位必须通过在中断服务程序中写“0”到中断标志位 INTPND.1 来清零。

11.1.3.2 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 的写入值相等时，将产生一个匹配信号。这个匹配信号产生一个 Timer A 匹配中断 (TAINT，中断向量地址：D0H) 并清零计数器。

例如，若写“10H”到 TADATA，计数器将自增到“10H”。此时，Timer A 产生中断请求，计数器复位，之后重新计数。每次匹配都将 Timer A 的输出电平反转 (*图 11-2*)。

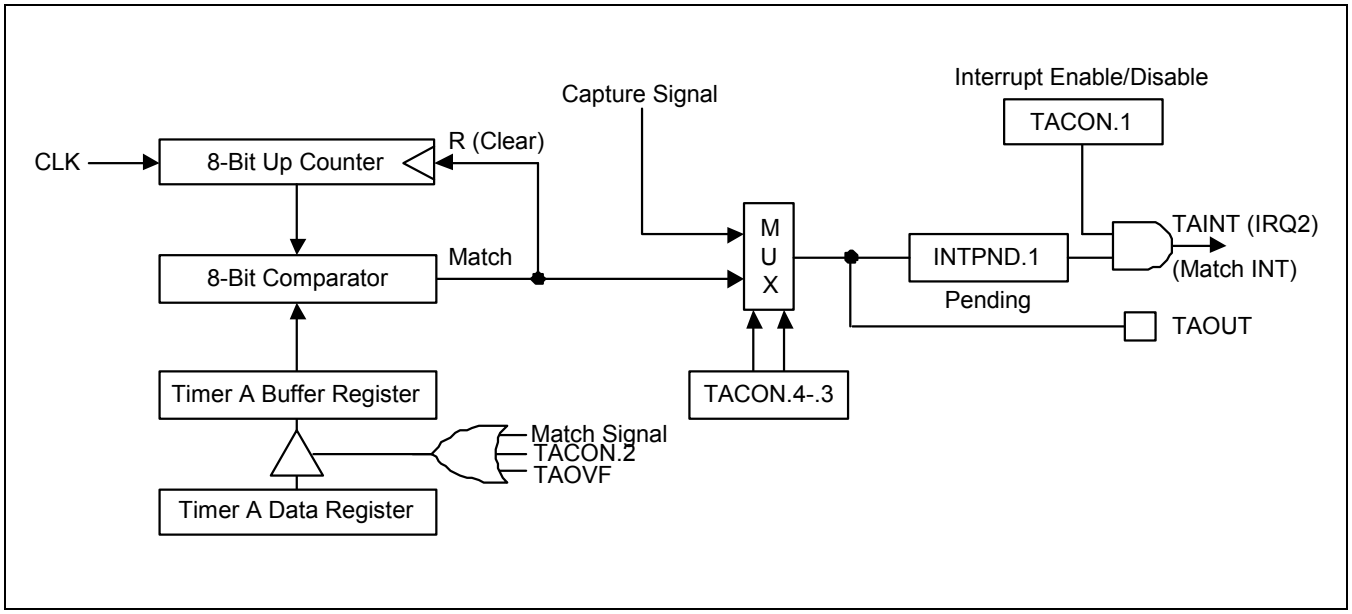


图 11-2 简化的 Timer A 功能框图：Interval (定时) 模式

11.1.4 PWM 模式

脉冲宽度调制 (PWM) 模块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，这个匹配信号并不将计数器清零，而是继续计数直到“FFH”溢出，之后从“00H”重新开始向上计数。

虽然在 PWM 模式下可以使用 Timer A 的匹配信号产生一个 Timer A 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，TAPWM 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，TAPWM 管脚上的脉冲保持高电平。一个脉冲周期等于256 个 t_{CLK} (*图 11-3*)。

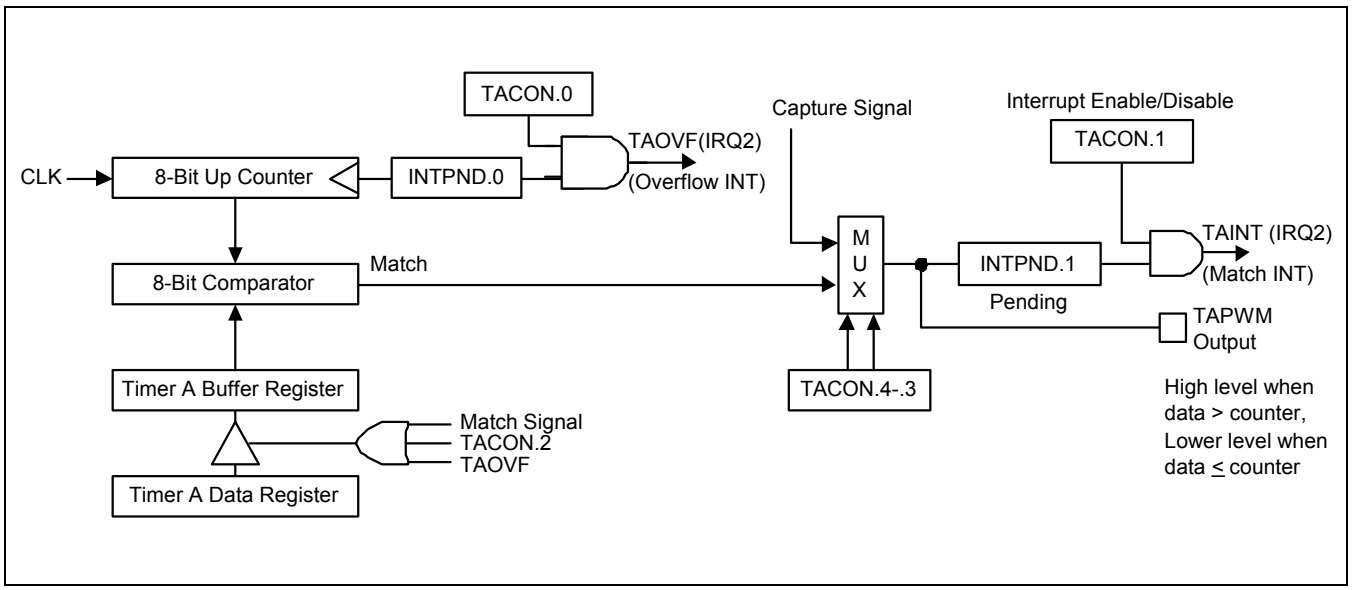


图 11-3 简化的 Timer A 功能框图：PWM 模式

11.1.5 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，把当前计数器的值载入 Timer A 数据寄存器中。可以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 P3 口控制寄存器 P3CONH.3–.2 (地址：EAH，Set 1，Bank 1) 的 Timer A 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 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 中的捕获数据值，并为 Timer A 假定一个特定的时钟频率，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*。

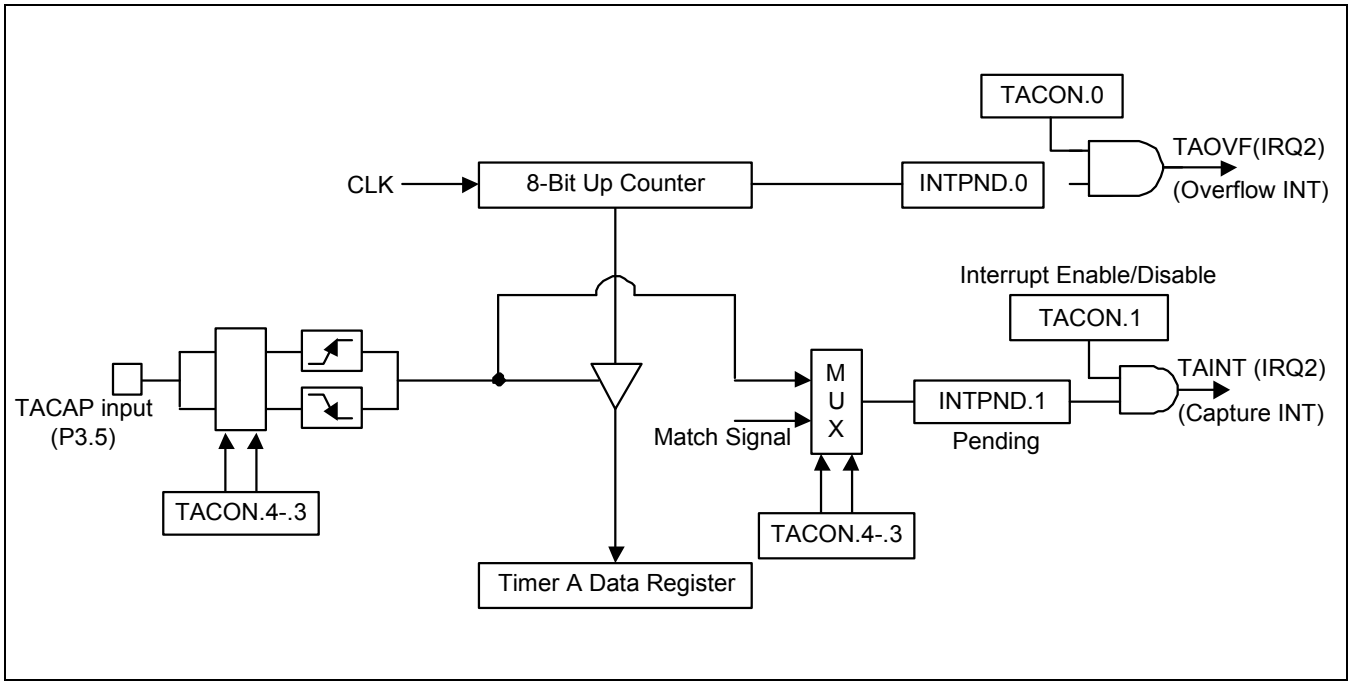


图 11-4 简化的 Timer A 功能框图：Capture (捕获) 模式

11.1.6 模块框图

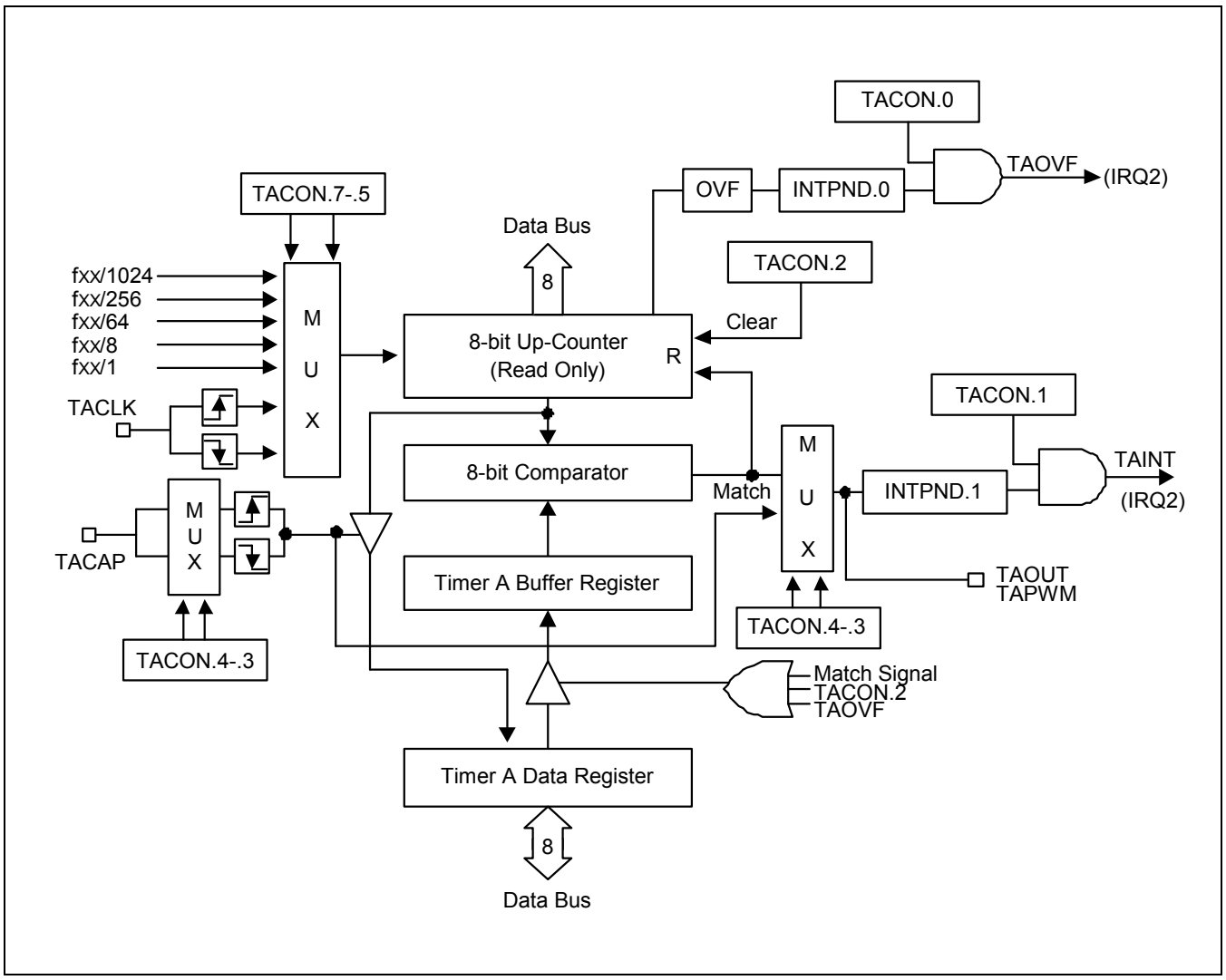


图 11-5 Timer A 功能模块框图

11.2 8 位 TIMER B

11.2.1 概述

S3F82HB MCU 有一个8位计数器 Timer B，用于产生遥控器输出信号的载波频率。

Timer B 有 2 个功能：

- 作为普通的定时时钟，在编程的时间间隔内产生 Timer B 中断
- 给8位 Timer B 提供时钟源，用于产生 Timer B 溢出中断

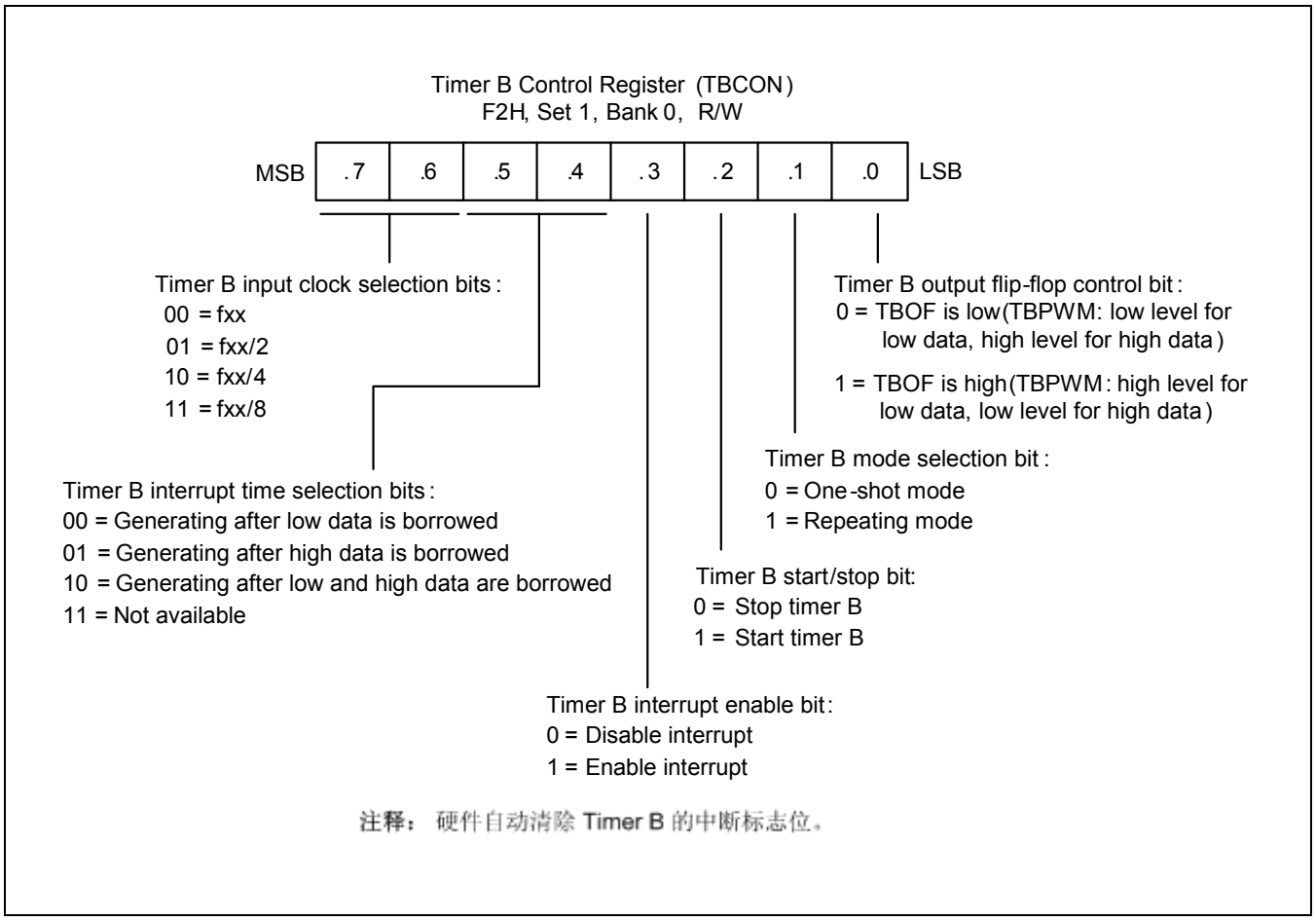
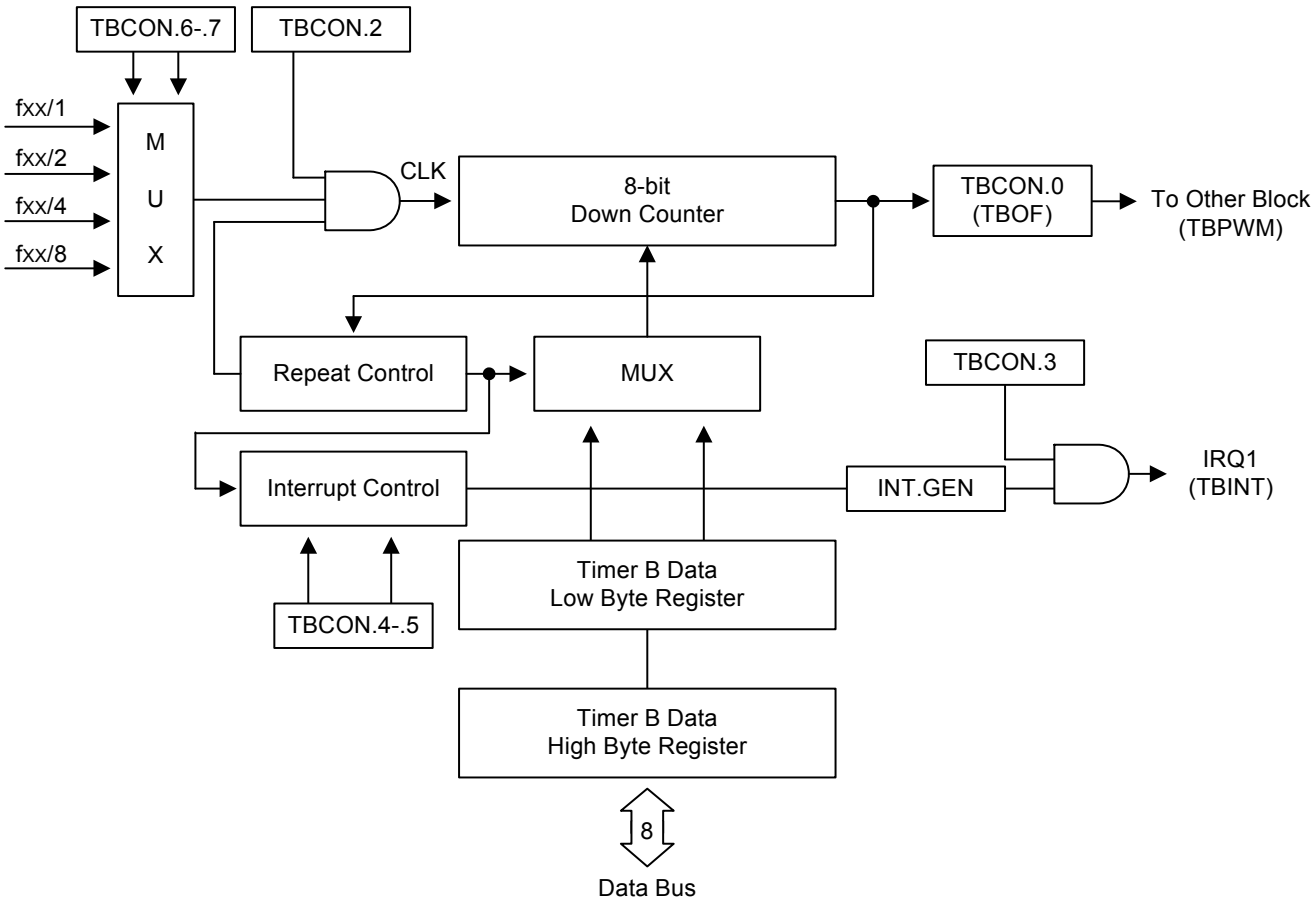


图 11-6 Timer B 控制寄存器

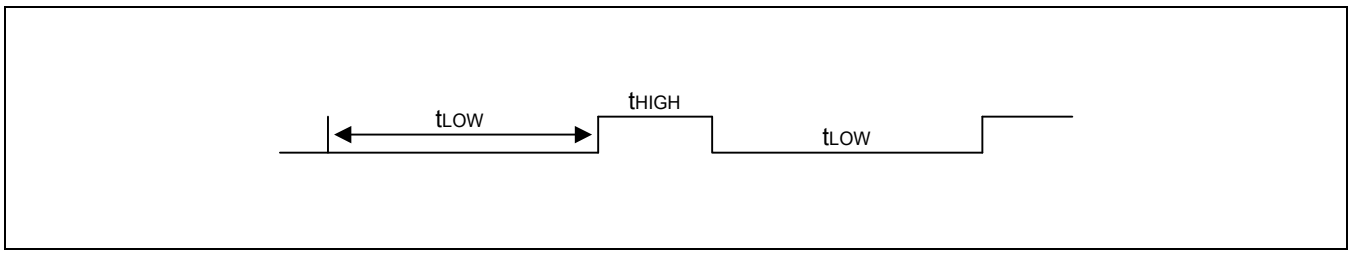
11.2.2 模块框图



注释： 当 Timer B 开始工作时，TBDATAL 寄存器的值装载到 8 位计数器中。如果计数器发生借位，TBDATAH 寄存器的值装载到该 8 位计数器中。但在下一次借位发生时，TBDATAL 寄存器的值又装载到这个 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 (由低电平时间 t_{LOW} 和高电平时间 t_{HIGH} 组成)，

当 TBOF = 0 时，

t_{LOW} = (TBDATAL + 2) × 1/fx，0H < TBDATAL < 100H，其中 fx = 已选的时钟。

t_{HIGH} = (TBDATAH + 2) × 1/fx，0H < TBDATAH < 100H，其中 fx = 已选的时钟。

当 TBOF = 1 时，

t_{LOW} = (TBDATAH + 2) × 1/fx，0H < TBDATAH < 100H，其中 fx = 已选的时钟。

t_{HIGH} = (TBDATAL + 2) × 1/fx，0H < TBDATAL < 100H，其中 fx = 已选的时钟。

为了获得 t_{LOW} = 24us，t_{HIGH} = 15us。f_{OSC} = 4MHz，fx = 4MHz/4 = 1MHz

当 TBOF = 0 时，

t_{LOW} = 24us = (TBDATAL + 2) /fx = (TBDATAL + 2) × 1us，TBDATAL = 22。

t_{HIGH} = 15us = (TBDATAH + 2) /fx = (TBDATAH + 2) × 1us，TBDATAH = 13。

当 TBOF = 1 时，

t_{HIGH} = 15us = (TBDATAL + 2) /fx = (TBDATAL + 2) × 1us，TBDATAL = 13。

t_{LOW} = 24us = (TBDATAH + 2) /fx = (TBDATAH + 2) × 1us，TBDATAH = 22。

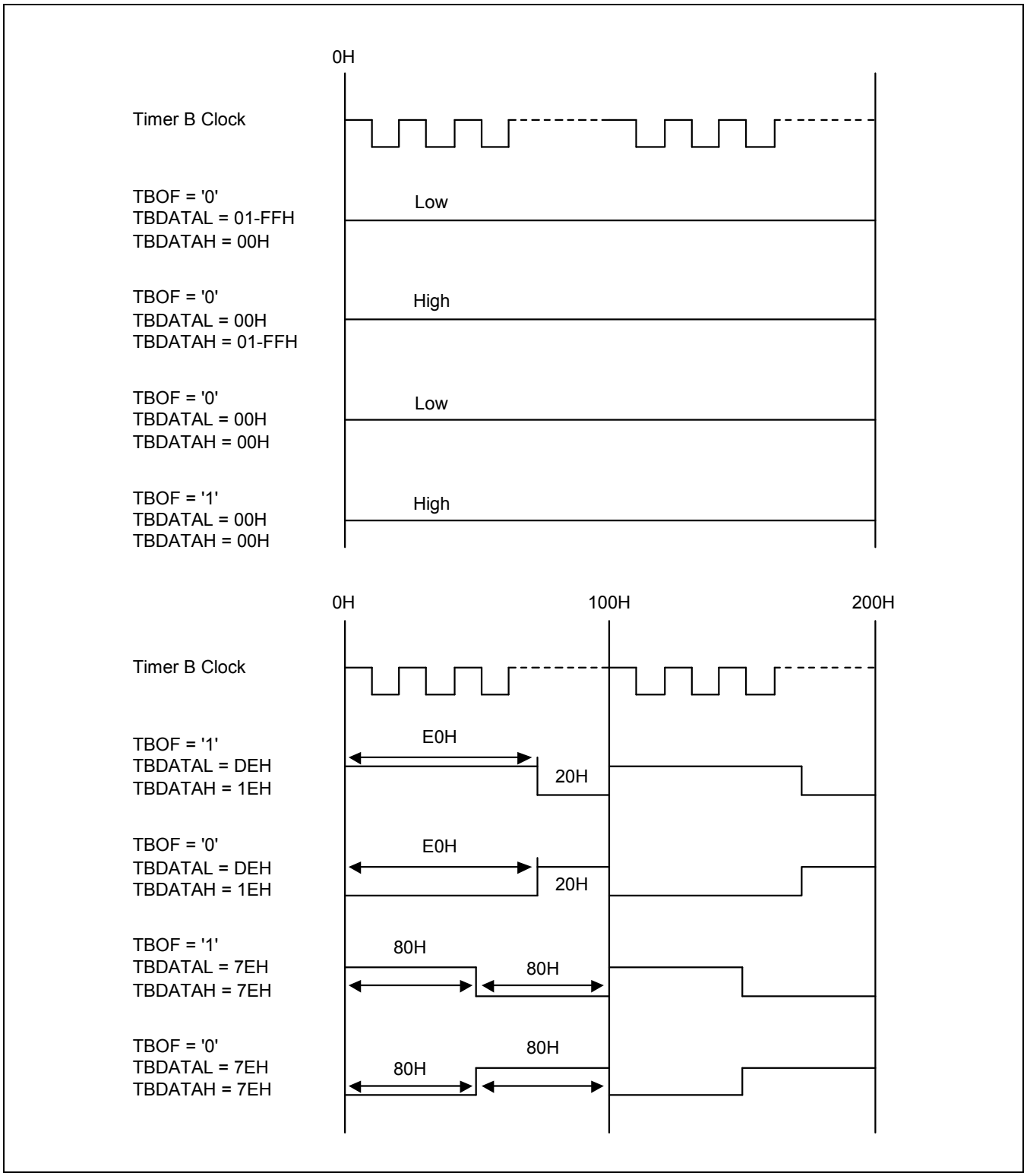
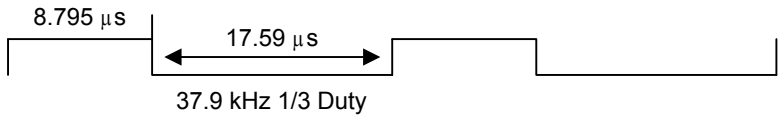


图 11-8 Timer B 重复模式输出波形

编程实例 11-1 如何在 P0.7 处产生 38kHz, 1/3 占空比信号

本例中设置 Timer B 为重复模式，并且设置时钟频率为 Timer B 时钟源，以及设置 TBDATAH 和 TBDATAL 获得一个 38 kHz，1/3占空比得载波频率。该编程参量如下：

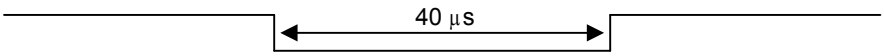


- Timer B 用于重复模式
- 时钟频率为 4MHz (0.25µs)
- TBDATAH = 8.795µs /0.25µs = 35.18，TBDATAL = 17.59µs /0.25µs = 70.36
- 设置 P0.7 为 TBPWM 模式

```
            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 为重复模式
                                      ; 启动 Timer B 工作
                                      ; 设置 Timer B 输出触发 (TBOF) 高

            OR     P0CONH,#0C0H       ; 设置 P0.7 为 TBPWM 模式
                                      ; 该指令在 P0.7 处产生38kHz，1/3占空比脉冲信号
            •
            •
            •
```

编程实例 11-2 如何在 P0.7 处产生一个脉冲

本例中设置 Timer B 为单次模式，并且设置时钟频率为 Timer B 时钟源，以及设置 TBDATAH 和 TBDATAL 以获得一个40µs 宽度脉冲。该编程参量如下：



- Timer B 用于单次模式
- 时钟频率为 4MHz (1 clock = 0.25µs)
- TBDATAH = 40µs / 0.25µs = 160，TBDATAL = 1
- 设置 P0.7 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 为单次模式
                                      ; 停止 Timer B
                                      ; 设置 Timer B 输出触发 (TBOF) 高
            OR     P0CONH, #0C0H      ; 设置 P0.7 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 16 位 TIMER 0/1

12.1 16 位 TIMER 0

12.1.1 概述

Timer0 是一个16位的通用 timer/counter。
Timer 0 有3种工作模式，通过设置 T0CON 寄存器可以选择任一工作模式：

- 定时模式 (T0OUT 输出)
- 捕获输入模式 (T0CAP 的上升沿或下降沿触发)
- PWM 模式 (T0PWM)

Timer 0 有如下的功能模块：

- 带多路复用的时钟分频器 (fxx /256，fxx $_C$/64，fxx $_C$/8，fxx /1)
- 外部时钟输入管脚 (T0CLK，TBOF)
- 16位计数器 (T0CNTH/L)，16 位比较器以及16位数据寄存器 (T0DATAH/L)
- 捕获输入管脚 (T0CAP)，或者匹配输出管脚 (T0PWM，T0OUT)
- 产生Timer 0 溢出中断 (IRQ0，中断向量地址 C8H) 和匹配/捕获中断 (IRQ0，中断向量地址 C6H)
- Timer 0 控制寄存器 T0CON (set 1，bank 0，E3H，可读/写)

12.1.1.1 Timer 0 控制寄存器 (T0CON)

Timer 0 控制寄存器 (T0CON) 可以：

- 选择 Timer 0 工作模式 (定时模式，捕获模式，PWM模式)
- 选择 Timer 0 输入时钟频率
- 清零 Timer 0 计数器 T0CNTH/T0CNTL
- 使能 Timer 0 溢出中断或者 匹配/捕获中断

T0CON 位于 set 1，Bank 0的地址 E3H，可以通过寄存器寻址模式读/写。

复位操作将 T0CON 置为“00H”，这将设置 Timer 0 为定时模式，选择 TBOF 的输入时钟频率，
并禁止所有Timer 0 中断。可以通过设置 T0CON .7 - .5 为“111B”来禁止计数器操作。
可以通过写“1” 到T0CON.2随时清零Timer 0计数器。

Timer 0 溢出中断 (T0OVF) 的中断优先级为IRQ0，中断向量地址为C8H。当Timer 0溢出中断发生时，
响应中断(IRQ0, 向量 C8H)，必须写“1”到T0CON.0. 当Timer 0溢出中断发生并由CPU响应时，
标志位自动硬件清零或者软件清零。

要是能Timer 0 匹配/捕获中断(IRQ0, 向量 C6H)，必须写“1”到T0CON.1.要检测中断标志位，
应用程序可以轮询INTPND.3, 检测到标志位为“1”时，Timer 0匹配/捕获中断挂起。
当中断响应后，可以通过软件对中断标志位INTPND.3写“0”来清除中断标志。

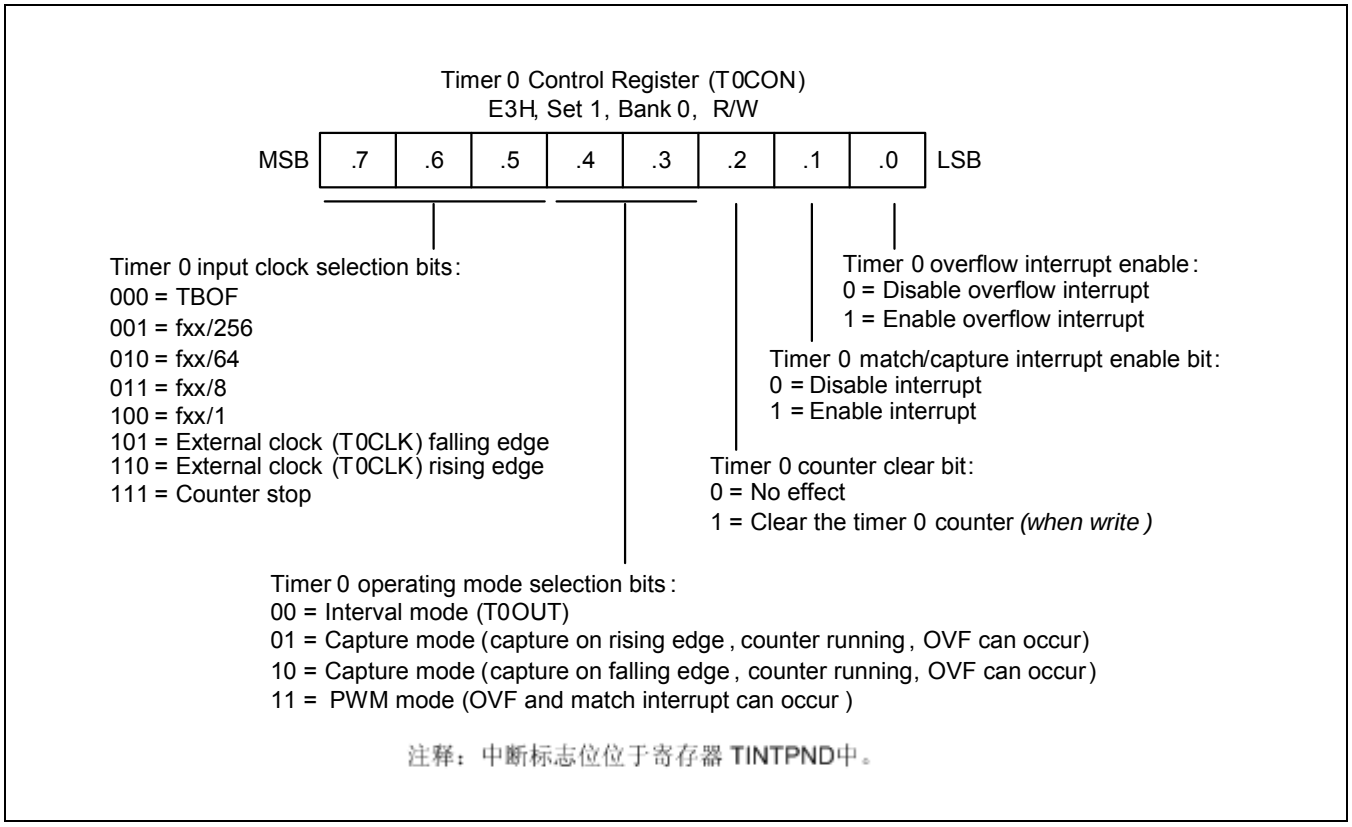


图 12-1 Timer 0 控制寄存器 (T0CON)

12.1.2 TIMER 0 功能描述

12.1.2.1 Timer 0 中断 (IRQ0, 中断向量地址 C6H 和 C8H)

Timer 0 模块能产生2个中断，Timer 0 溢出中断 (T0OVF) 以及Timer 0 匹配/捕获中断 (T0INT)。T0OVF的中断优先级为IRQ0，向量地址为C8H。T0INT的中断优先级也为IRQ0，不同的是向量地址为C6H。

Timer 0 溢出中断标志在中断响应后由硬件自动清除或者在应用程序中写“0”到 INTPND.2软件清零。Timer 0匹配/捕获中断必须通过在应用程序中写“0”到 INTPND.3软件清零。

12.1.2.2 定时模式

在定时模式中，当计数器的值与Timer 0参考数据寄存器T0DATAH/T0DATAL的值相等时，生成一个匹配信号。匹配信号生成一个Timer 0匹配中断 (T0INT，向量地址 C6H) 同时计数器清零。

例如，如果写"1087H" 到 T0DATAH/T0DATAL，计数器计数到“1087H”，产生Timer 0中断请求，计数器值复位，开始重新计数。每次匹配，Timer 0输出的信号发生反转(*图 12-2*)。

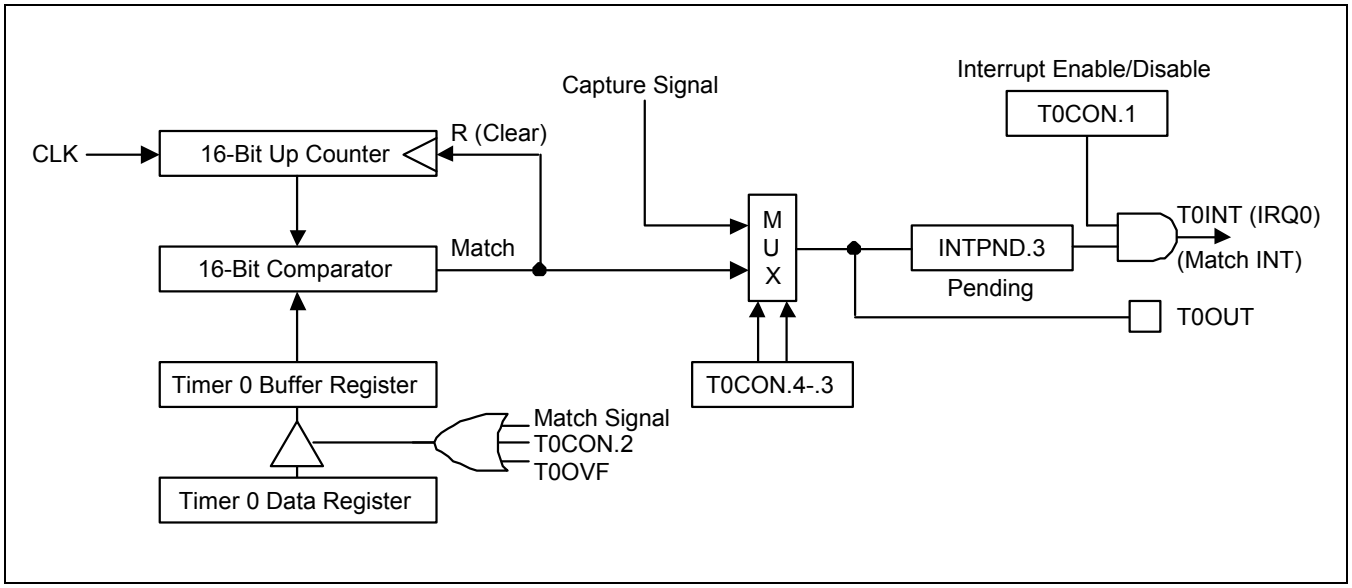


图 12-2 Timer 0 功能简图: 定时模式

12.1.2.3 PWM 模式

脉宽调制模块 (PWM)，用户可以控制管脚T0PWM输出的脉冲宽度 (持续时间)。
在PWM模式下，当计数器的值与Timer 0 数据寄存器中的值相等时，产生一个匹配信号，但不清零计数器，它会连续工作，直到FFFFH时溢出，然后从0000H继续向上计数。

虽然在PWM模式下可以使用匹配或者溢出中断，但是这些中断在PWM应用中并不普遍。常用到的功能是，管脚T0 PWM 上的脉冲，在参考数据寄存器中的值小于等于计数器的值时保持低电平；
而在参考数据寄存器中的值大于计数器的值时保持高电平。一个脉冲宽度等于 t_{CLK} 65536 (*图 12-3*)。

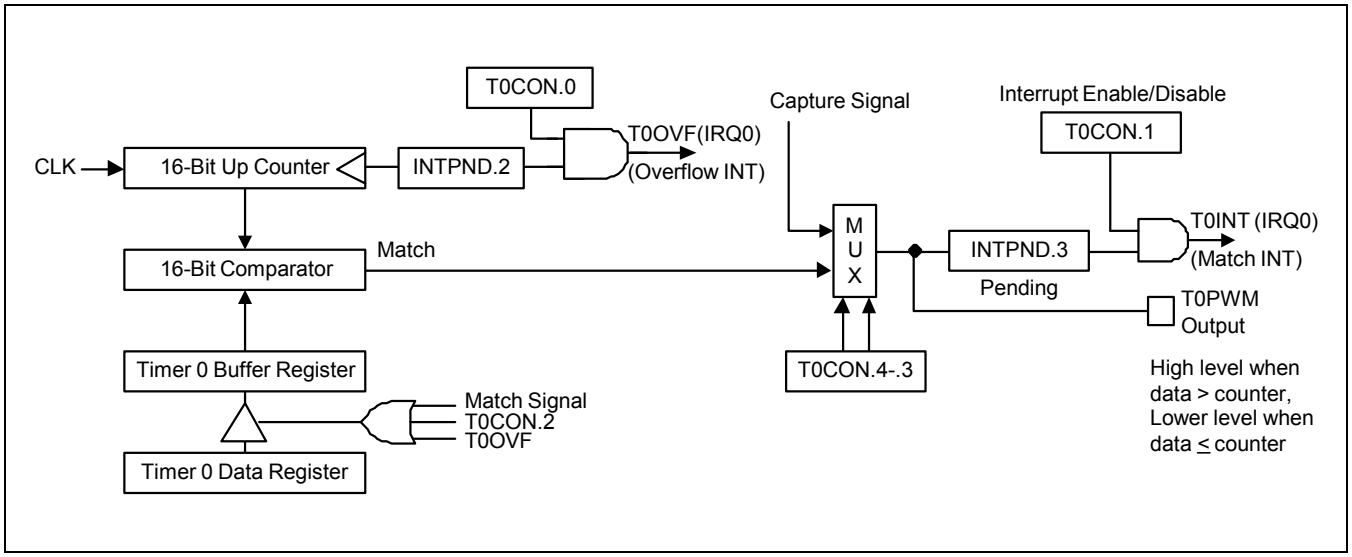


图 12-3 Timer 0 功能简图: PWM 模式

12.1.3 捕获模式

Timer 0 的捕获模式下，一旦管脚T0CAP上检测到信号边沿就将当前计数器的值载入Timer 0数据寄存器。用户可以选择上升沿或者下降沿来触发此操作。

Timer 0捕获输入源：管脚T0CAP上的信号边沿。可以通过设置P3高字节控制寄存器P1CONH.5–.4 (set 1 bank1，E7H) 来选择捕获输入。当P1CONH.5–.4为"00"时，选择T0CAP输入。

Timer 0 的两种中断均可用于捕获模式，当计数器溢出时产生Timer 0溢出中断；当计数器值载入Timer 0数据寄存器时产生Timer 0 捕获中断。

通过读取T0DATAH/T0DATAL,寄存器中的值，并为Timer 0假设一个特定的时钟频率，就可以计算出管脚T0CAP上输入信号的脉冲宽度(持续时间) (*图 12-4*)。

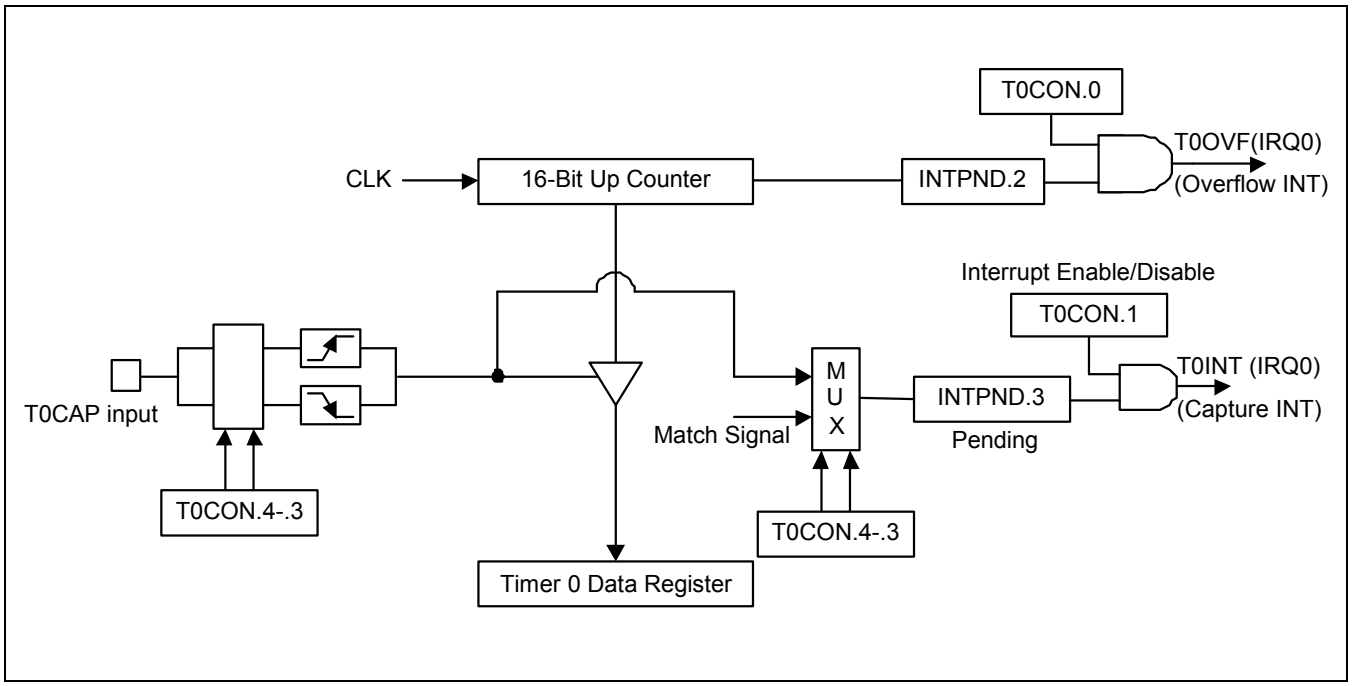


图 12-4 Timer 0 功能简图: 捕获模式

12.1.4 框图

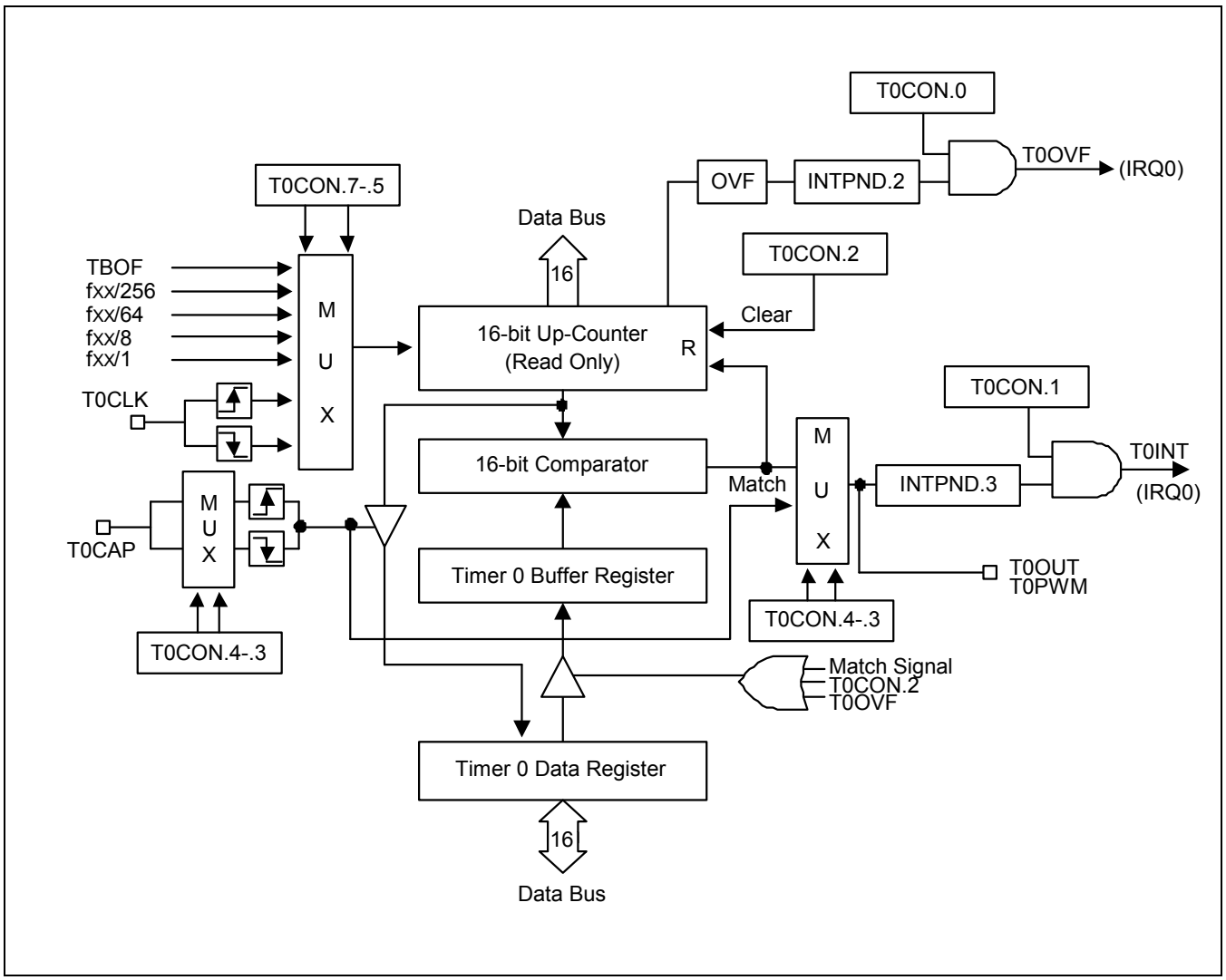


图 12-5 Timer 0 功能框图

12.2 16 位TIMER 1

12.2.1 概述

16位Timer 1是一个16位的通用timer。Timer 1 有3种工作模式，通过设置T1CON寄存器可以选择任一工作模式：

- 定时模式 (T1OUT 输出)
- 捕获输入模式 (T1CAP的上升沿或下降沿触发)
- PWM 模式 (T1PWM)；PWM 输出与T1OUT输出管脚复用

Timer 1 有如下的功能模块：

- 带多路复用的时钟分频器 (fxx /1024，fxx /256，fxx $_C$/64, fxx $_C$/8，fxx /1)
- 外部时钟输入管脚 (T1CLK)
- 1个16位计数器 (T1CNTH/L)，1个16 位比较器以及1个16位数据寄存器(T1DATAH/L)
- 捕获输入管脚 (T1CAP)，或者匹配输出管脚 (T1OUT)
- 产生Timer 1溢出中断 (IRQ0，中断向量地址 CCH) 和匹配/捕获中断 (IRQ0，中断向量地址 CAH)
- Timer 1控制寄存器 T1CON (set 1, Bank 0, EBH，读/写)

12.2.1.1 Timer 1 控制寄存器 (T1CON)

Timer 1 控制寄存器T1CON可以：

- 选择Timer 1工作模式 (定时模式, 捕获模式, PWM模式)
- 选择Timer 1 输入时钟频率
- 清零Timer 1 计数器 T1CNTH/T1CNTL
- 使能Timer 1 溢出中断或者使能Timer 1 匹配/捕获中断

T1CON 位于set 1，Bank 0的地址EBH，可以通过寄存器寻址模式读/写。

复位操作将置T1CON为“00H”，这将设置Timer 1 为定时模式，选择输入的时钟频率为fxx/1024，并禁止所有Timer 1 中断。可以通过设置T1CON .7 - .5 为“111B”来禁止计数器操作。可以通过写“1”到T1CON.2随时清l零Timer 1计数器。

Timer 1溢出中断 (T1OVF) 的中断优先级为IRQ0，中断向量地址为CCH。当Timer 1溢出中断发生时，响应中断(IRQ0, 向量 CCH)，必须写“1”到T1CON.0. 当Timer 1溢出中断发生并由CPU响应时，标志位自动硬件清零或者软件清零。

要是能Timer 1 匹配/捕获中断(IRQ0, 向量 CAH)，必须写“1”到T1CON.1.要检测中断标志位，应用程序可以轮询INTPND.5, 检测到标志位为“1”时，Timer 1匹配/捕获中断挂起。
当中断响应后，可以通过软件对中断标志位INTPND.5写“0”来清除中断标志。

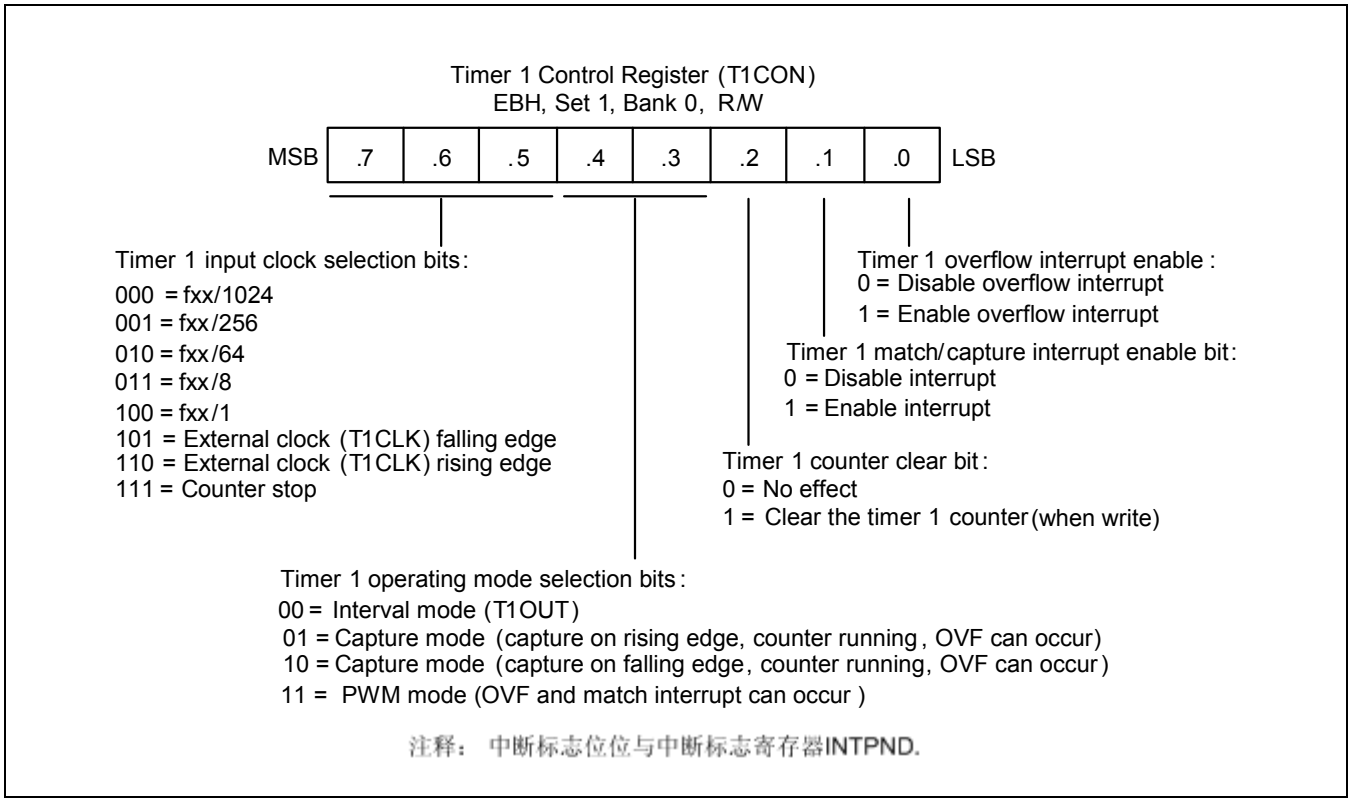


图 12-6 Timer 1 控制寄存器 (T1CON)

12.2.2 TIMER 1 功能描述

12.2.2.1 Timer 1 中断 (IRQ0, 中断向量地址 CAH 和 CCH)

Timer 1 模块能产生2个中断，Timer 1 溢出中断 (T1OVF) 以及Timer 1 匹配/捕获中断 (T1INT)。T1OVF的中断优先级为IRQ0，向量地址为CCH。T1INT的中断优先级也为IRQ0，不同的是向量地址为CAH。

Timer 1 溢出中断标志在中断响应后由硬件自动清除或者在应用程序中写“0”到 INTPND.4软件清零。Timer 1匹配/捕获中断必须通过在应用程序中写“0”到 INTPND.5软件清零。

12.2.2.2 定时模式

在定时模式中，当计数器的值与Timer 1参考数据寄存器T1DATAH/TD0DATAL的值相等时，生成一个匹配信号。匹配信号生成一个Timer 1匹配中断 (T1INT，向量地址 CAH) 同时计数器清零。

例如，如果写"1087H" 到 T1DATAH/T1DATAL，计数器计数到“1087H”，产生timer 1中断请求，计数器值复位，开始重新计数。每次匹配，timer 1输出的信号发生反转(*图 12-7*)。

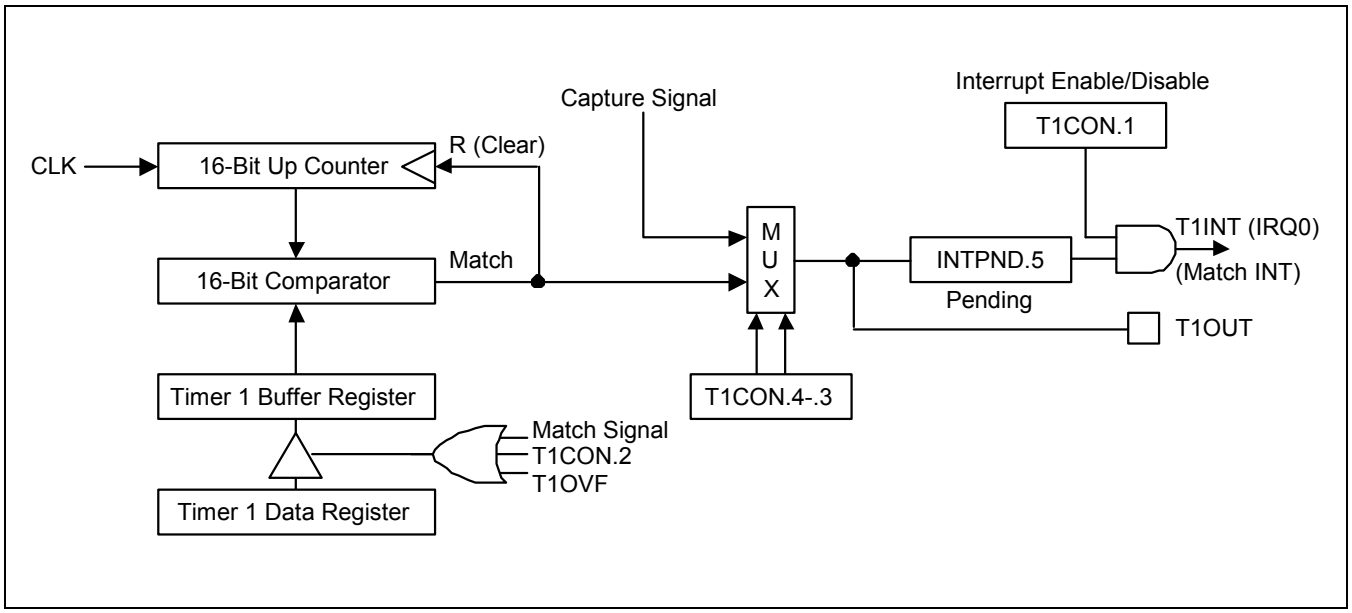


图 12-7 Timer 1 功能简图: 定时模式

12.2.2.3 PWM 模式

脉宽调制模块 (PWM) ，用户可以控制管脚T1PWM输出的脉冲宽度 (持续时间)。
在在PWM模式下，当计数器的值与Timer 1 数据寄存器中的值相等时，产生一个匹配信号，但不清零计数器，它会连续工作，直到FFFFH时溢出，然后从0000H继续向上计数。

虽然在PWM模式下可以使用匹配或者溢出中断，但是这些中断在PWM应用中并不普遍。常用到的功能是，管脚T1PWM 上的脉冲，在参考数据寄存器中的值小于等于计数器的值时保持低电平；
而在参考数据寄存器中的值大于计数器的值时保持高电平。一个脉冲宽度等于 t_{CLK} 65536 (*图 12-8*)。

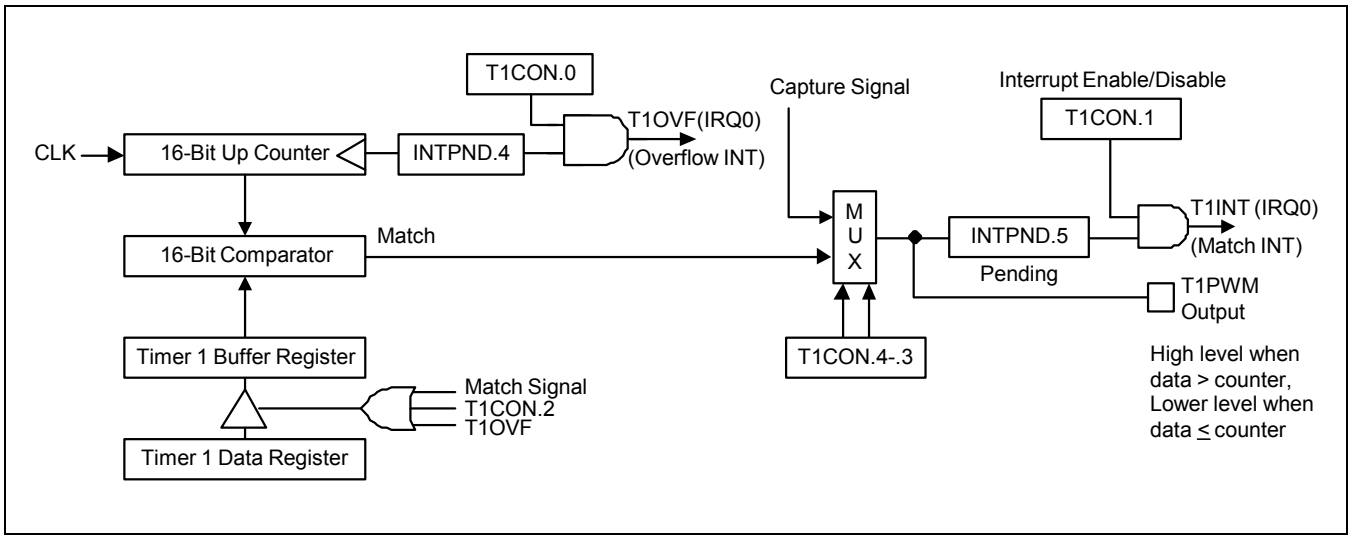


图 12-8 Timer 1 功能简图: PWM 模式

12.2.3 捕获模式

Timer 1 的捕获模式下，一旦管脚T1CAP上检测到信号边沿就将当前计数器的值载入Timer 1数据寄存器。用户可以选择上升沿或者下降沿来触发此操作。

Timer 1捕获输入源：管脚T1CAP上的信号边沿。可以通过设置P3高字节控制寄存器P1CONL.1–.0, (set 1 bank0，E4H) 来选择捕获输入。当P1CONL.1–.0, 为"00"时，选择T1CAP输入。

Timer 1 的两种中断 均可用于捕获模式，当计数器溢出时产生Timer 1溢出中断；当计数器值载入Timer 1数据寄存器时产生Timer 1捕获中断。

通过读取T1DATAH/T1DATAL,寄存器中的值，并为Timer 1假设一个特定的时钟频率，就可以计算出管脚T1CAP上输入信号的脉冲宽度（持续时间）(*图 12-9*)。

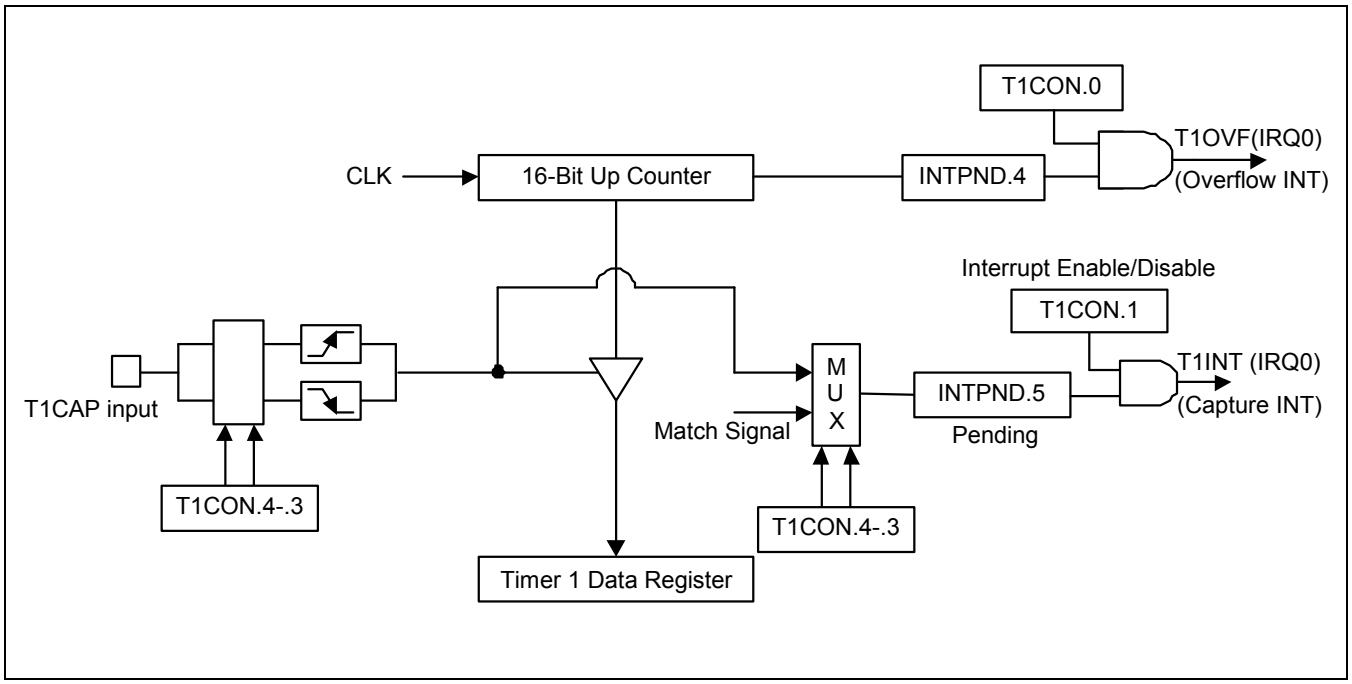


图 12-9 Timer 1 功能简图: 捕获模式

12.2.4 框图

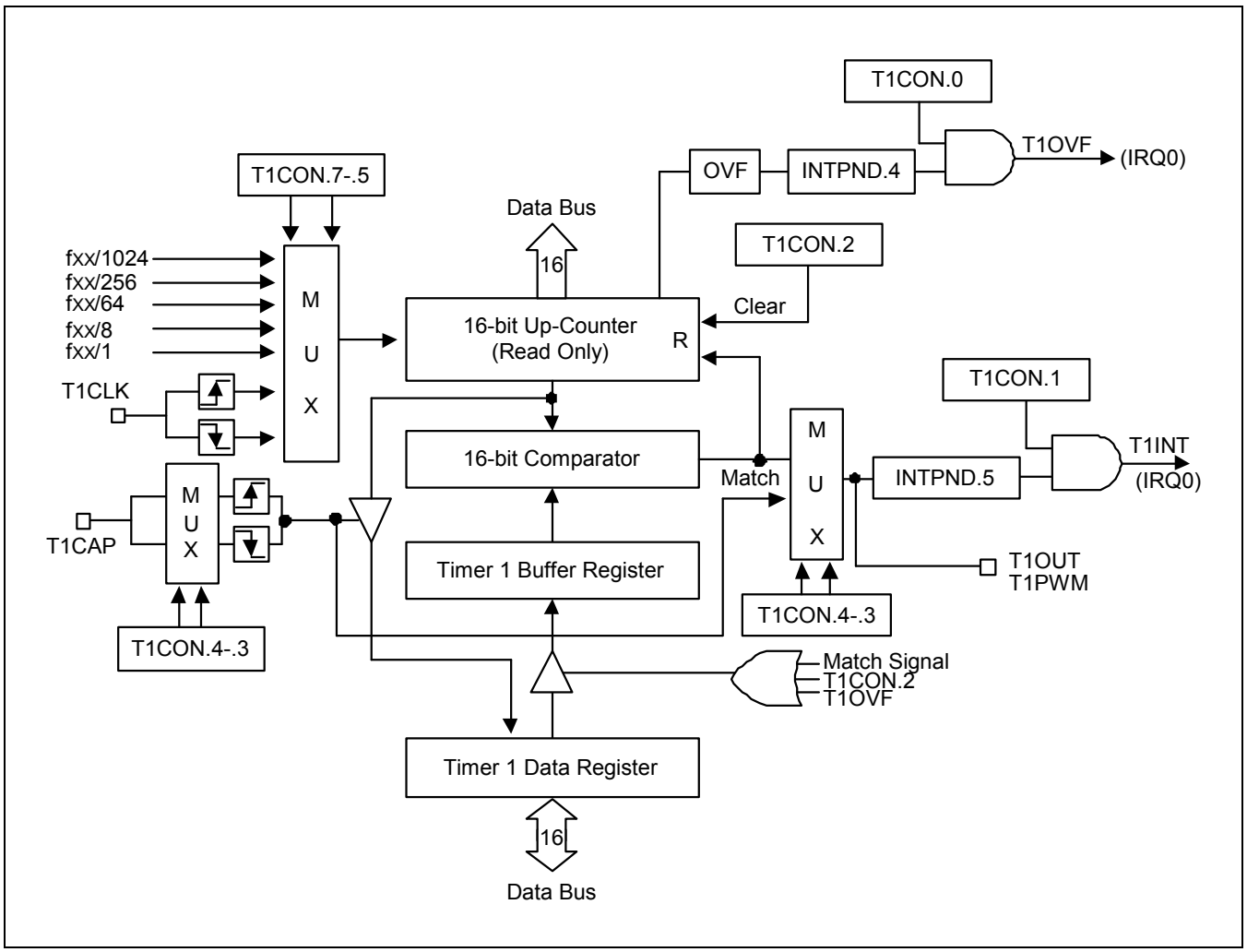


图 12-10 Timer 1 功能框图

13 钟表定时器

13.1 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, 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3.91ms，0.25s，0.5s 或 1s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 (IRQ2, 地址D4H) 发生
- 钟表定时器控制寄存器，WTCON (set 1，bank 0，D1H，可读/写)

13.1.1 钟表定时器控制寄存器 (WTCON: 可读写)

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。
它位于set 1，bank 0，地址D1H，可通过寄存器寻址模式进行读/写。复位后 WTCON 清零，
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。

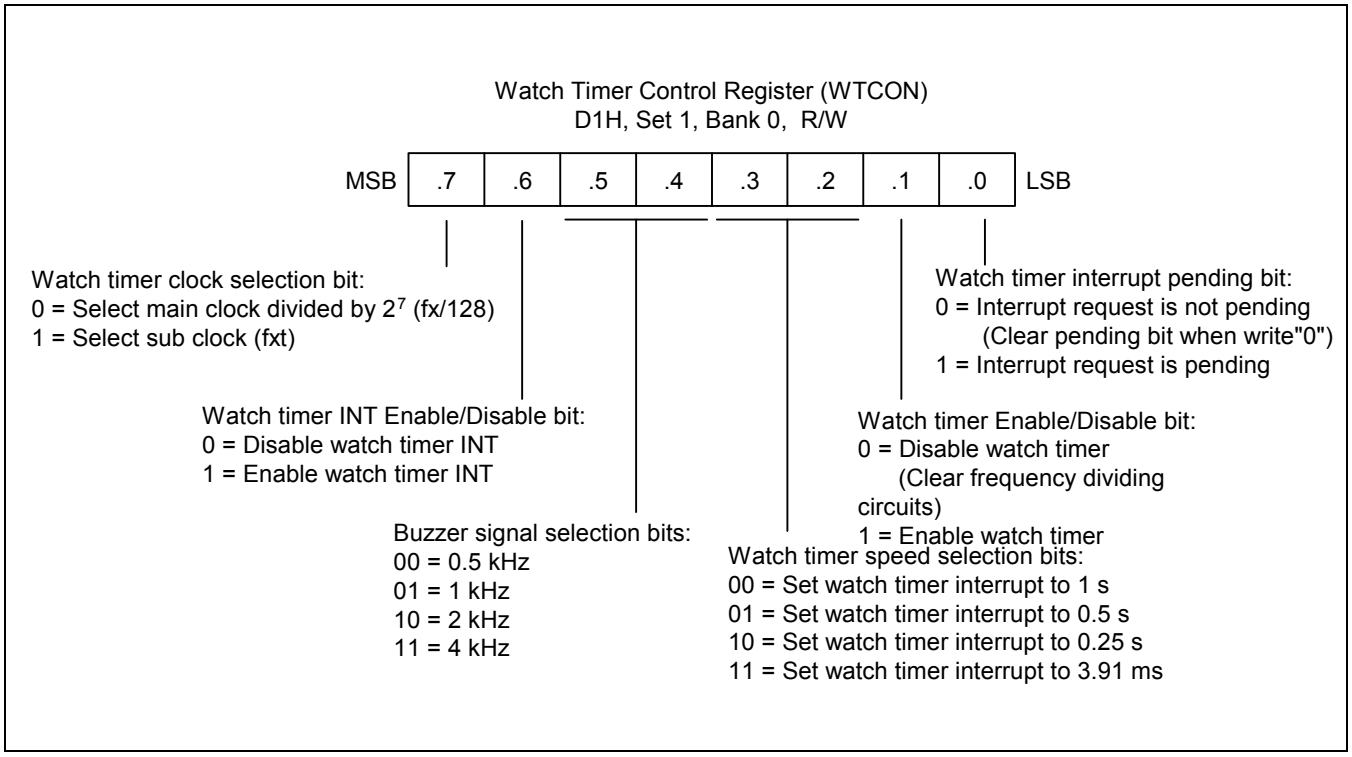


图 13-1 钟表定时器控制寄存器 (WTCON)

13.1.2 钟表定时器电路图

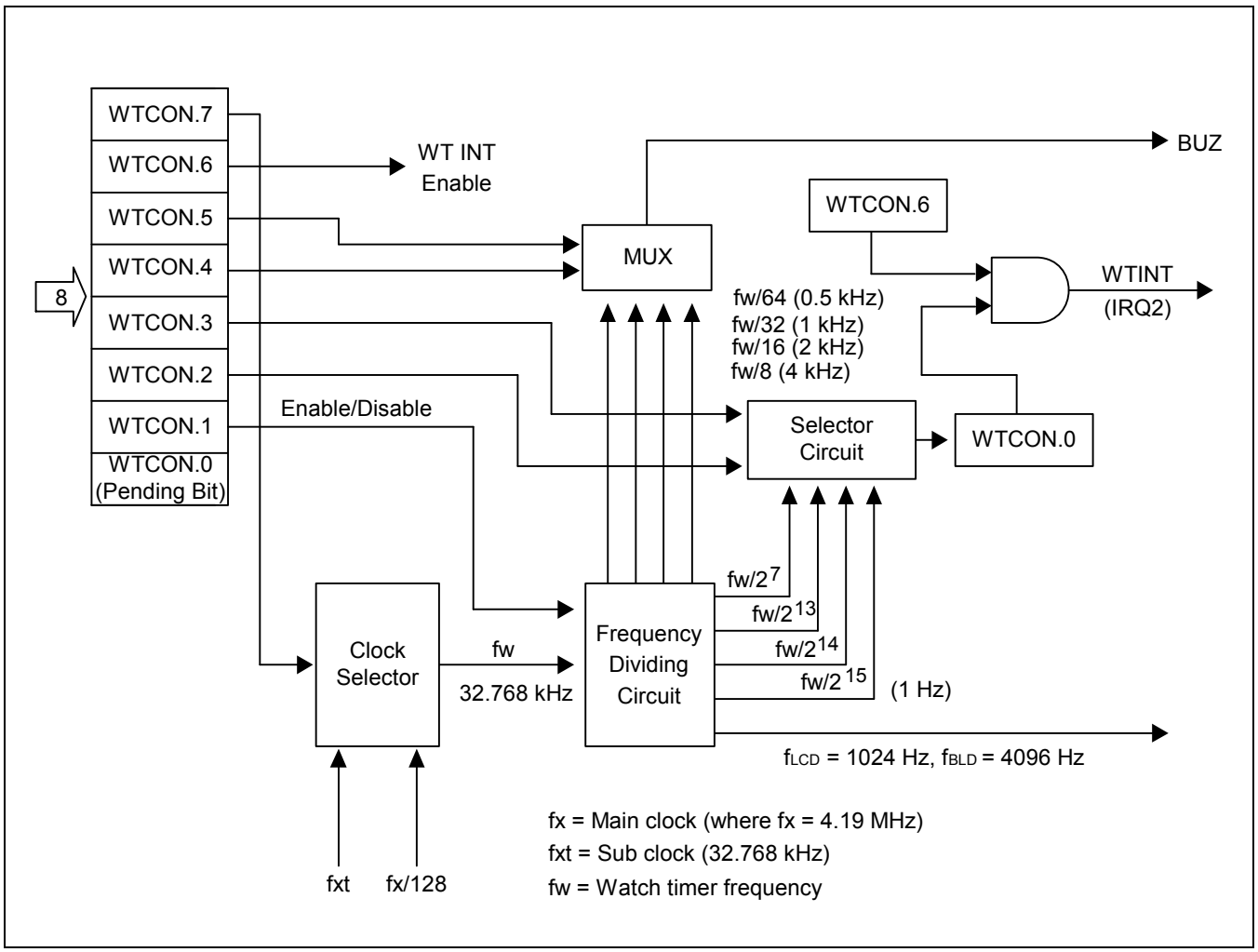


图 13-2 钟表定时器电路框图

14 LCD 控制器/驱动器

14.1 概述

S3F82HB 微控制器可以直接驱动最多416点阵 (52 segments x 8 commons) 的LCD面板。
LCD模块包括以下部分：

- LCD 控制器/驱动器
- 存储显示数据的显示RAM
- 6个common/segment 输出管脚 (COM2/SEG0–COM7/SEG5)
- 52个 segment 输出管脚 (SEG6–SEG57)
- 2个 common 输出管脚 (COM0–COM1)
- 4个LCD 供电电压管脚 (V_{LC0}–V_{LC3})
- LCD 内部/外部电阻偏压和电容偏压

LCD 控制寄存器 LCON 用来打开关断 LCD 显示，选择帧频率，选择占空比和偏压比，LCD偏压类型。
LCD 对比度控制寄存器用来选择 VLCD 电压。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。

当选择副振作为 LCD 时钟源时，即使在主时钟 STOP 或者 IDLE 模式下 LCD 仍可以显示。

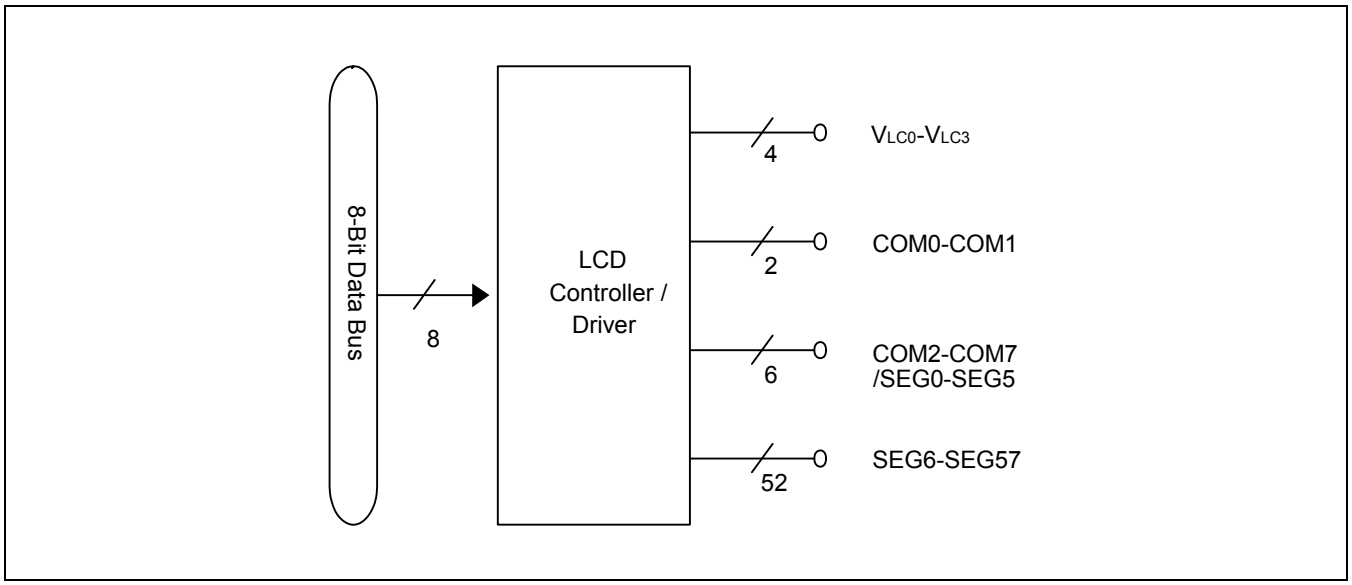


图 14-1 LCD 功能框图

14.1.1 LCD 电路框图

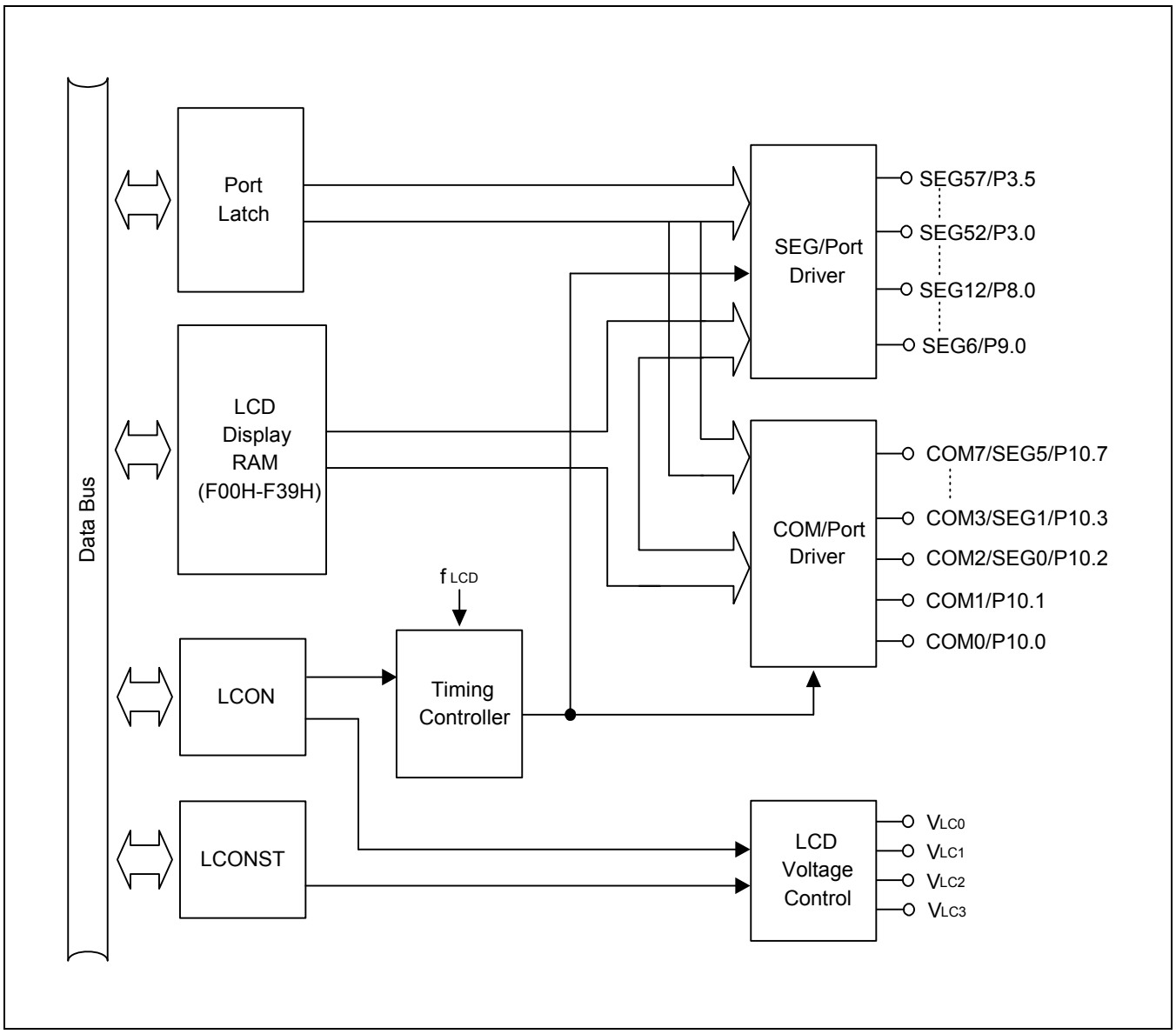


图 14-2 LCD 电路框图

14.1.2 LCD RAM 地址空间

page 15中的 RAM 空间被用作 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”，LCD 显示打开；当某个显示段的值为“0”，LCD 显示关闭。

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，通过管脚 SEG0-SEG57 送出。如果这些 RAM 地址没有用作 LCD 显示，也可以用作一般用途。

COM	Bit	SEG0	SEG1	SEG2	SEG3	SEG4	------	SEG56	SEG57
COM0 COM1 COM2 COM3 COM4 COM5 COM6 COM7	.0 .1 .2 .3 .4 .5 .6 .7	F00H	F01H	F02H	F03H	F04H	------	F38H	F39H

图 14-3 LCD 显示数据 RAM 结构图

14.1.3 LCD 控制寄存器 (LCON)

LCON 位于 0DH，Page 0，可读写，寄存器模式寻址。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
- 内部/外部分压电阻或者电容偏压选择

LCON 寄存器可以控制打开关断 LCD 显示，可以选择占空比和偏压比，可以选择 LCD 时钟和 LCD 偏压类型。复位后，LCON 的值被清零。LCD 显示关断，占空比选择1/8，偏压比选择1/4，LCD 时钟选择128Hz 并选择电容偏压。

LCD 时钟频率决定了每个 segment 输出 COM 信号的扫描频率，也即 LCD 的帧频。因为 LCD 时钟由钟表定时器 (fw) 产生，LCD 显示打开的时候必须使能钟表定时器 (watch timer)。

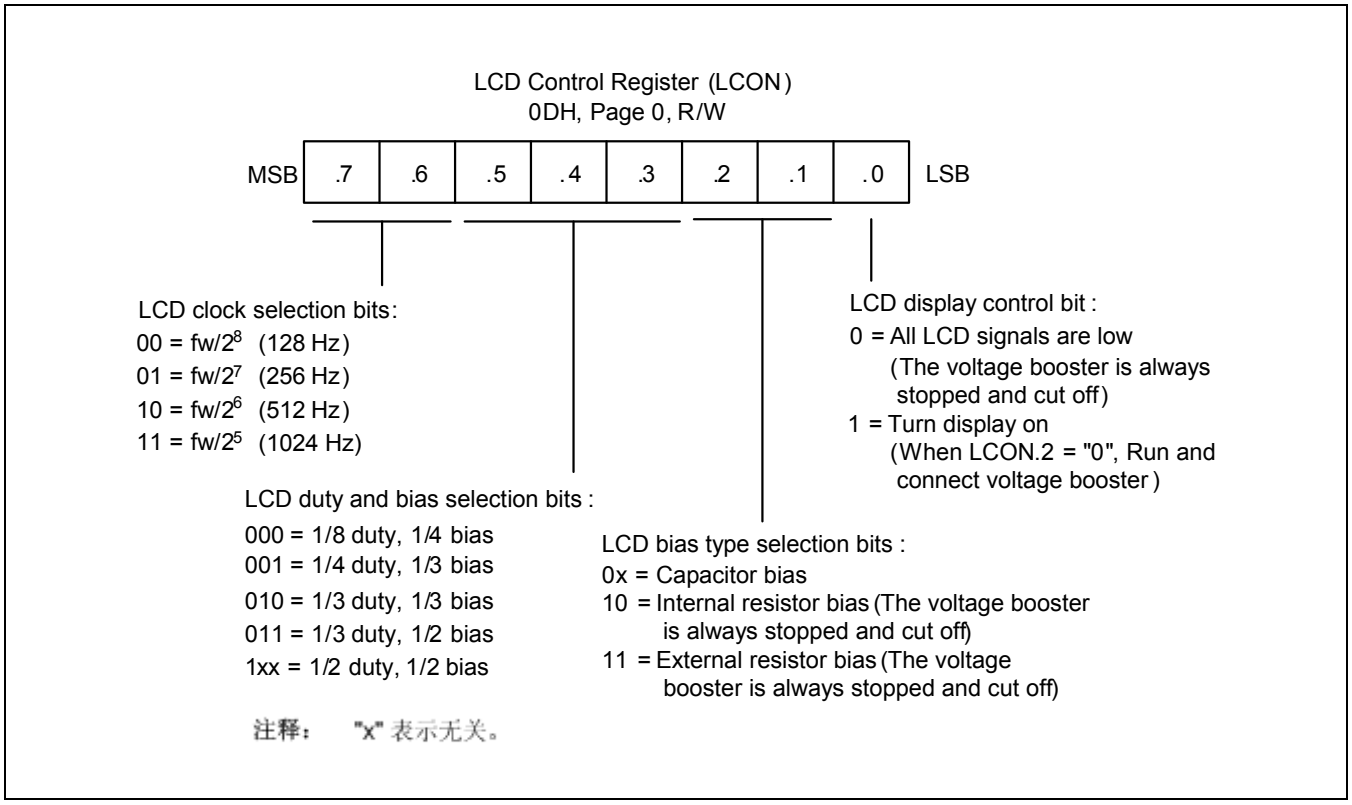


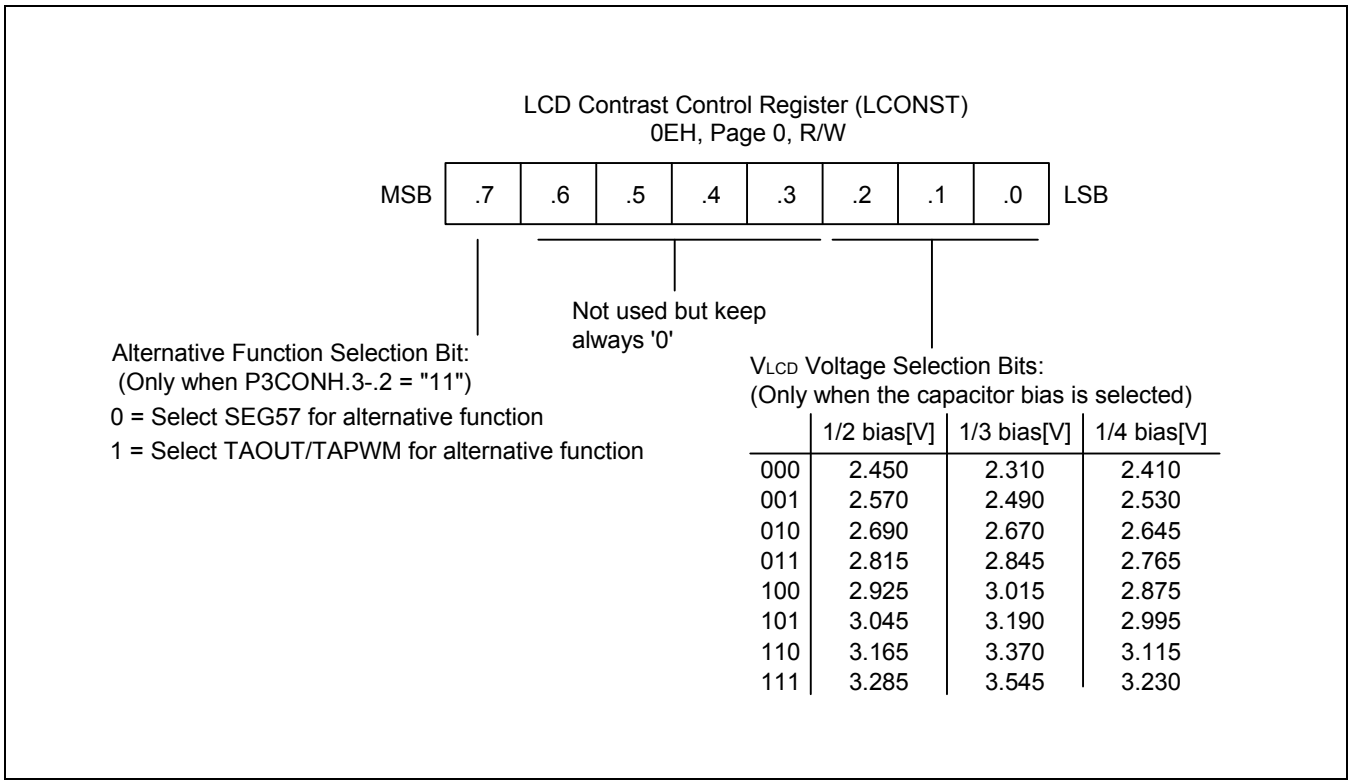
图 14-4 LCD 控制寄存器 (LCON)

14.1.4 LCD 对比度控制寄存器 (LCONST)

LCONST 地址为0EH，page0。可读/写，寄存器寻址模式。它包含以下功能：

- 选择复用功能 (SEG57 或 TAOUT/TAPWM)
- 选择 V_{LCD} 电压

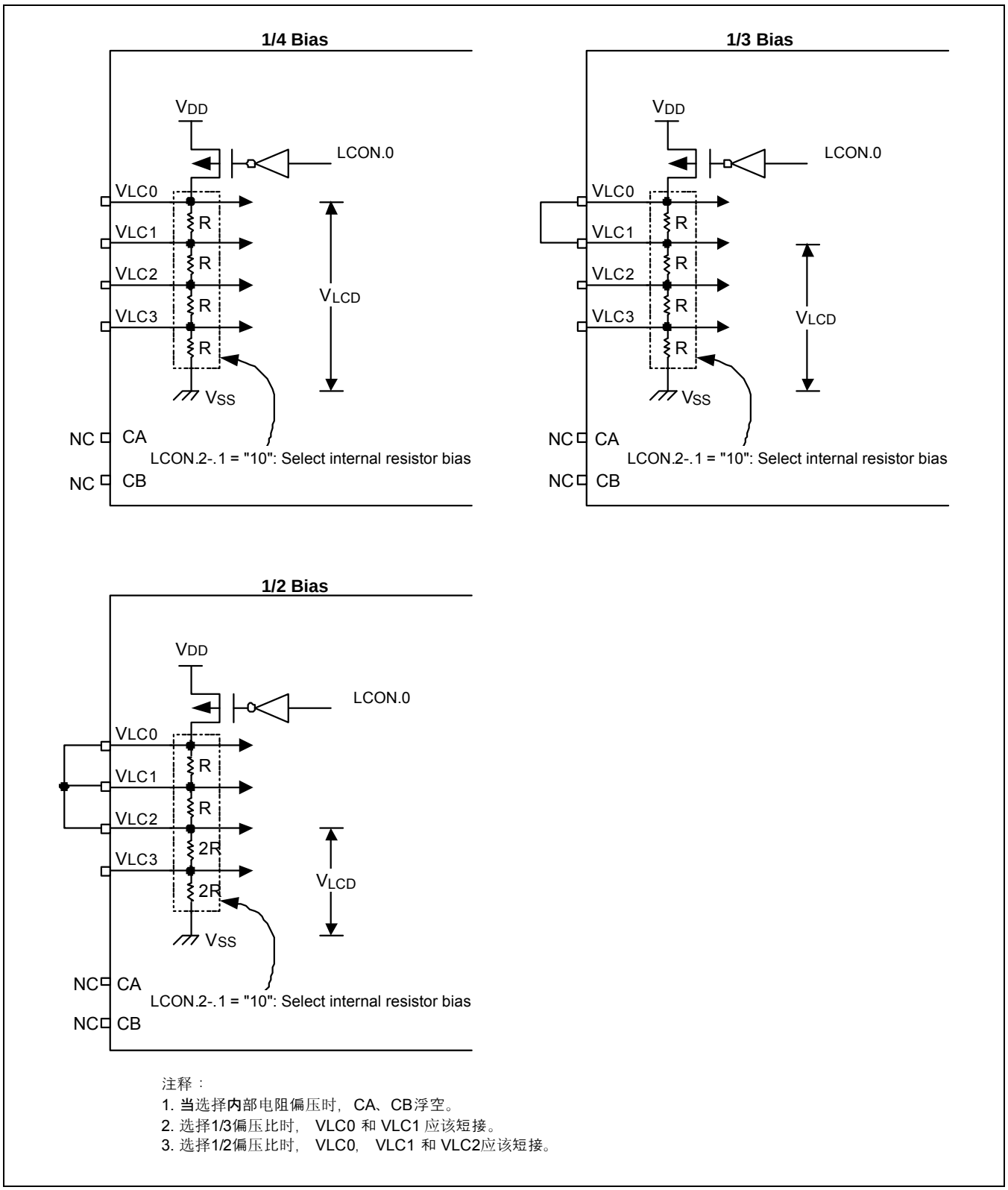
LCONST 用来选择复用功能 (SEG57 或 TAOUT/TAPWM) 和 V_{LCD} 电压。复位后 LCONST 寄存器清零，复用功能选择 SEG57 (当 P3CONH.3-.2=“11”时) 并选择 V_{LCD} 电压为2.41V (1/4偏压比时) 。



	1/2 bias[V]	1/3 bias[V]	1/4 bias[V]
000	2.450	2.310	2.410
001	2.570	2.490	2.530
010	2.690	2.670	2.645
011	2.815	2.845	2.765
100	2.925	3.015	2.875
101	3.045	3.190	2.995
110	3.165	3.370	3.115
111	3.285	3.545	3.230

图 14-5 LCD 对比度控制寄存器 (LCONST)

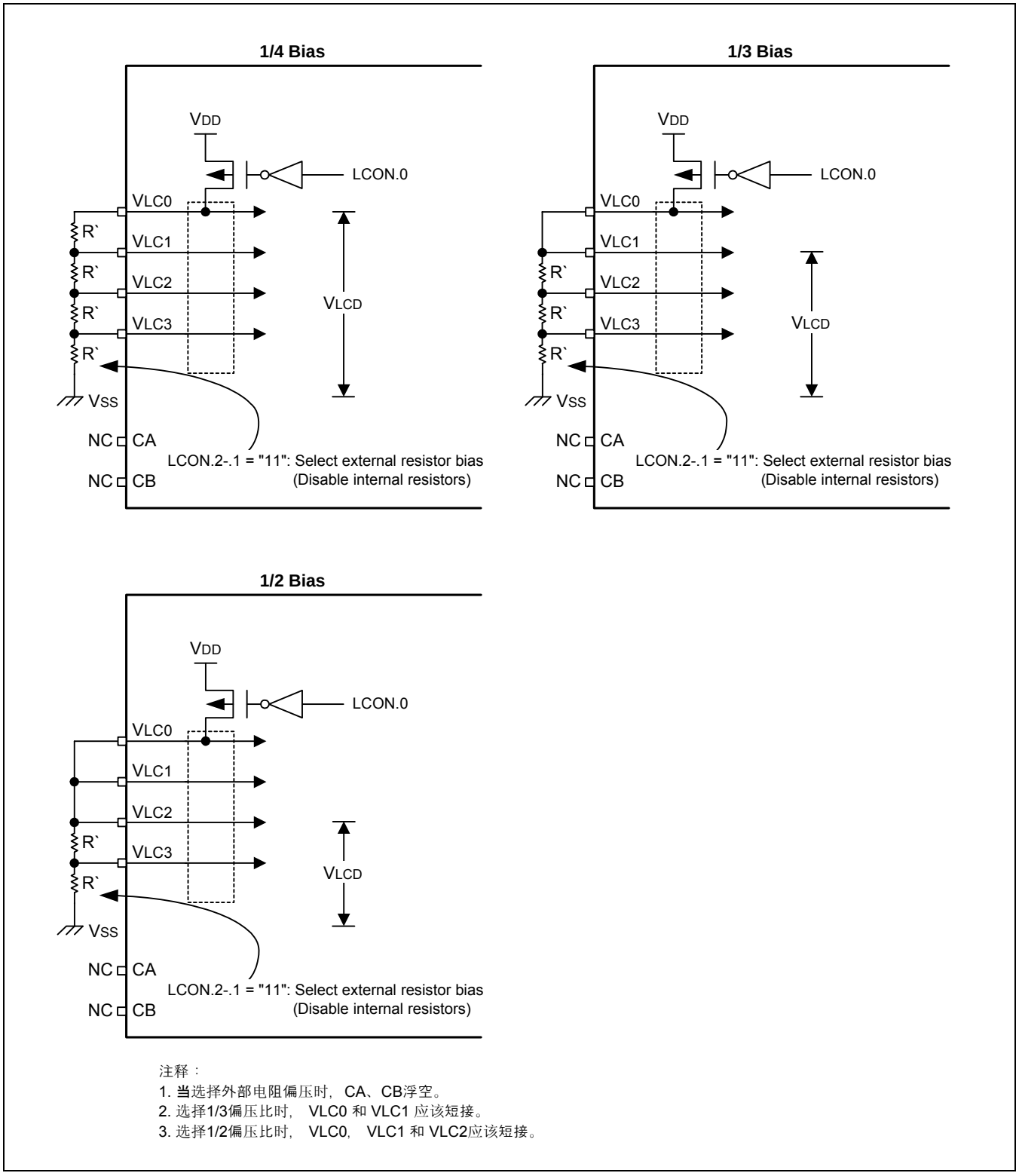
14.1.5 内部电阻偏压管脚连接



注释：
1. 当选择内部电阻偏压时，CA、CB浮空。
2. 选择1/3偏压比时， VLC0 和 VLC1 应该短接。
3. 选择1/2偏压比时， VLC0， VLC1 和 VLC2应该短接。

图 14-6 内部电阻偏压管脚连接

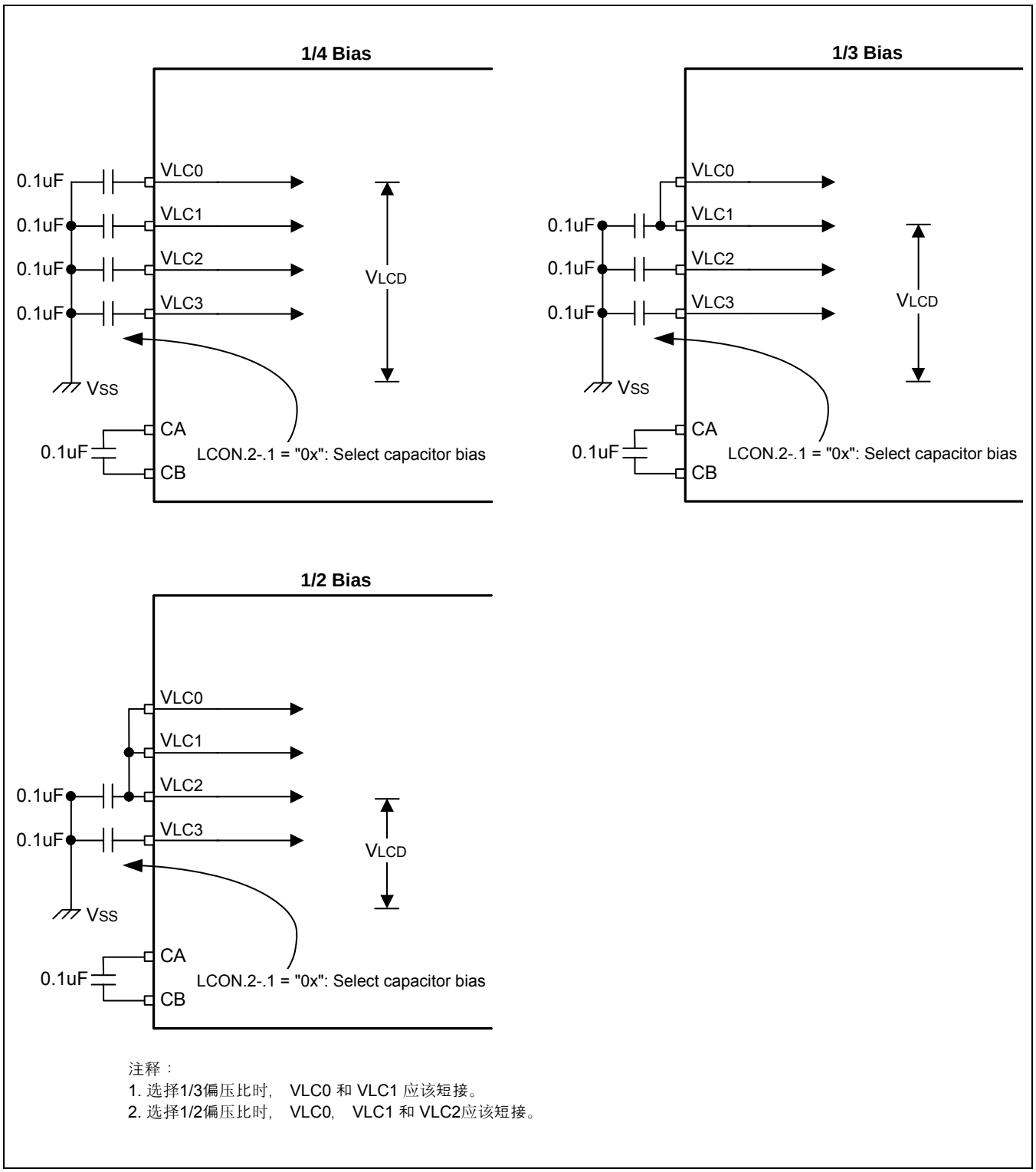
14.1.6 外部电阻偏压管脚连接



注释：
1. 当选择外部电阻偏压时， CA、CB浮空。
2. 选择1/3偏压比时， VLC0 和 VLC1 应该短接。
3. 选择1/2偏压比时， VLC0， VLC1 和 VLC2应该短接。

图 14-7 外部电阻偏压管脚连接

14.1.7 电容偏压管脚连接



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 14-8 电容偏压管脚连接

14.1.8 COMMON (COM) 信号

根据占空比选择不同的COM信号。

- 1/8 占空比，选择 COM0-COM7 (SEG6–SEG57)
- 1/4 占空比，选择 COM0-COM3 (SEG2–SEG57)
- 1/3 占空比，选择 COM0-COM2 (SEG1–SEG57)
- 1/2 占空比，选择 COM0-COM1 (SEG0–SEG57)

14.1.9 SEGMENT (SEG) 信号

58个 LCD segment 信号管脚连到相应的 page 15显示 RAM 区。
RAM 区的显示位与 COM 信号输出管脚同步。

当 RAM 区的显示位为“1”时，相应的 segment 被选中。
当显示位为“0”时，说明相应的 segment 没有被选中。

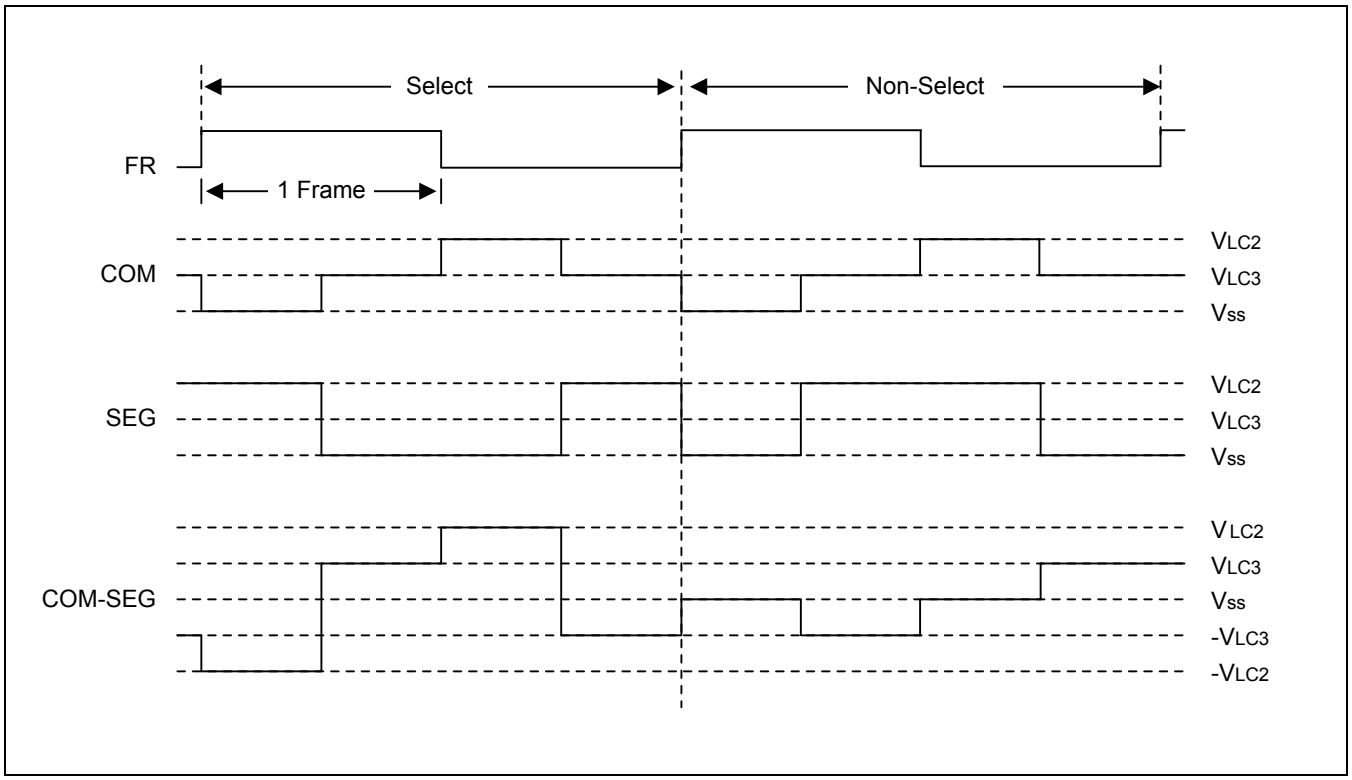


图 14-9 1/2占空比, 1/2偏压比显示模式下选中/非选中信号

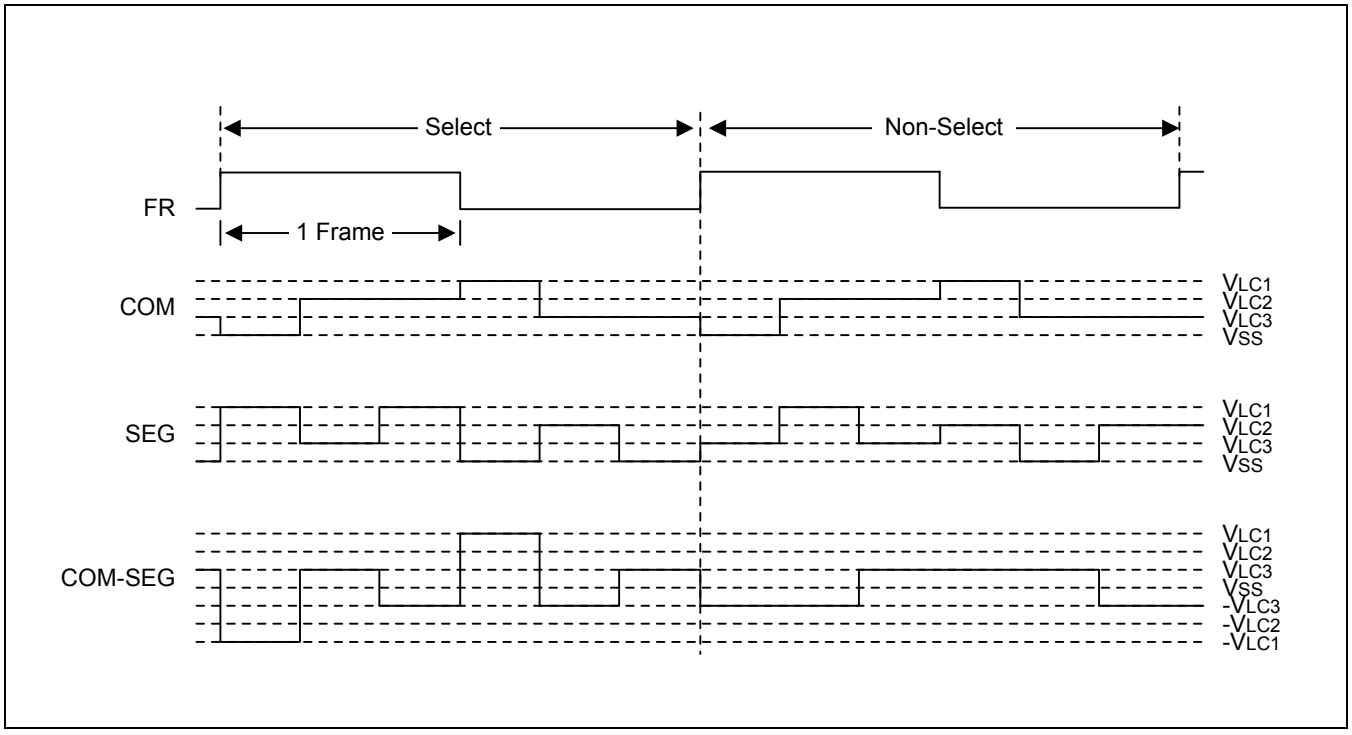


图 14-10 1/3占空比, 1/3偏压比显示模式下选中/非选中信号

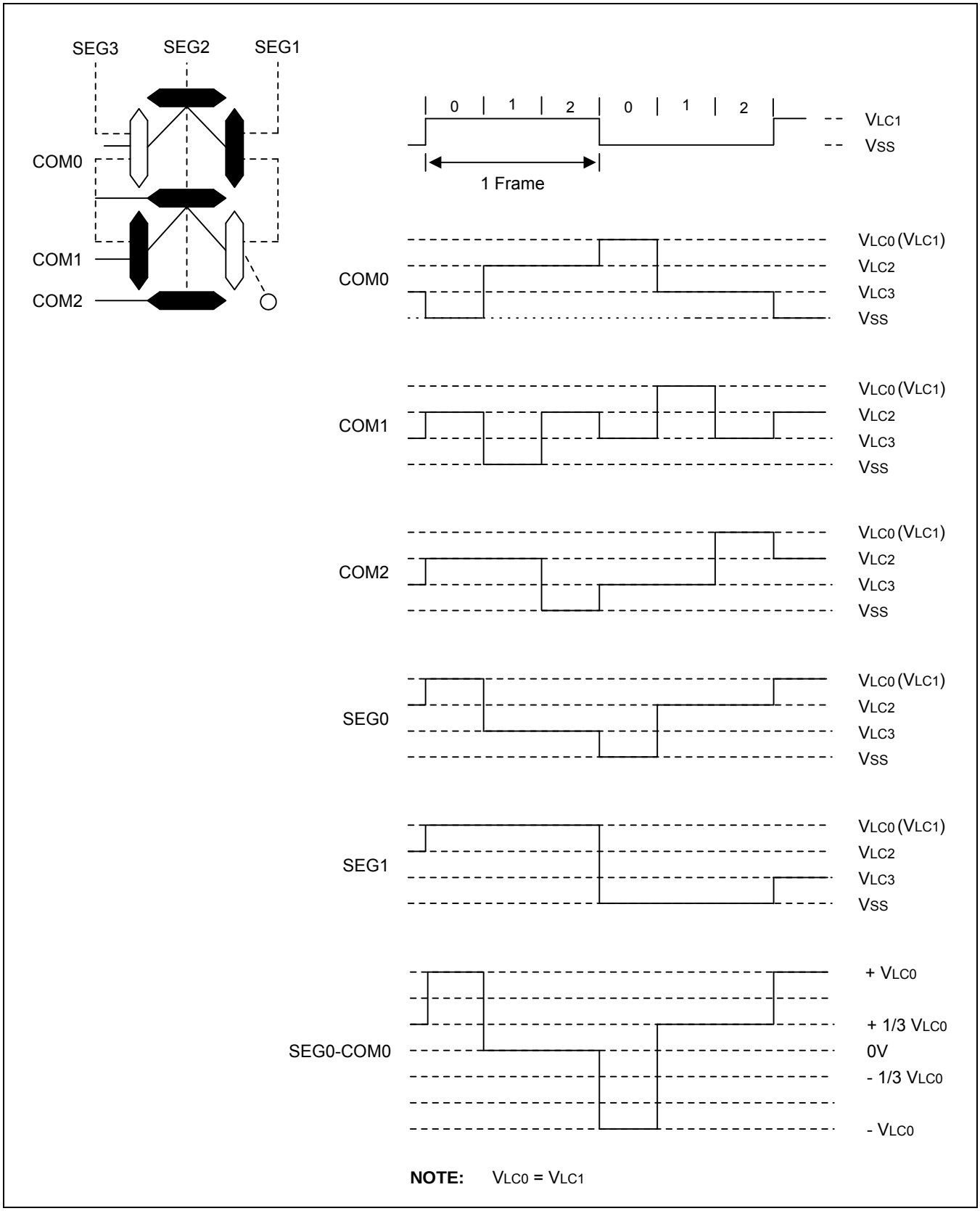


图 14-11 LCD 信号波形 (1/3 占空比, 1/3 偏压比)

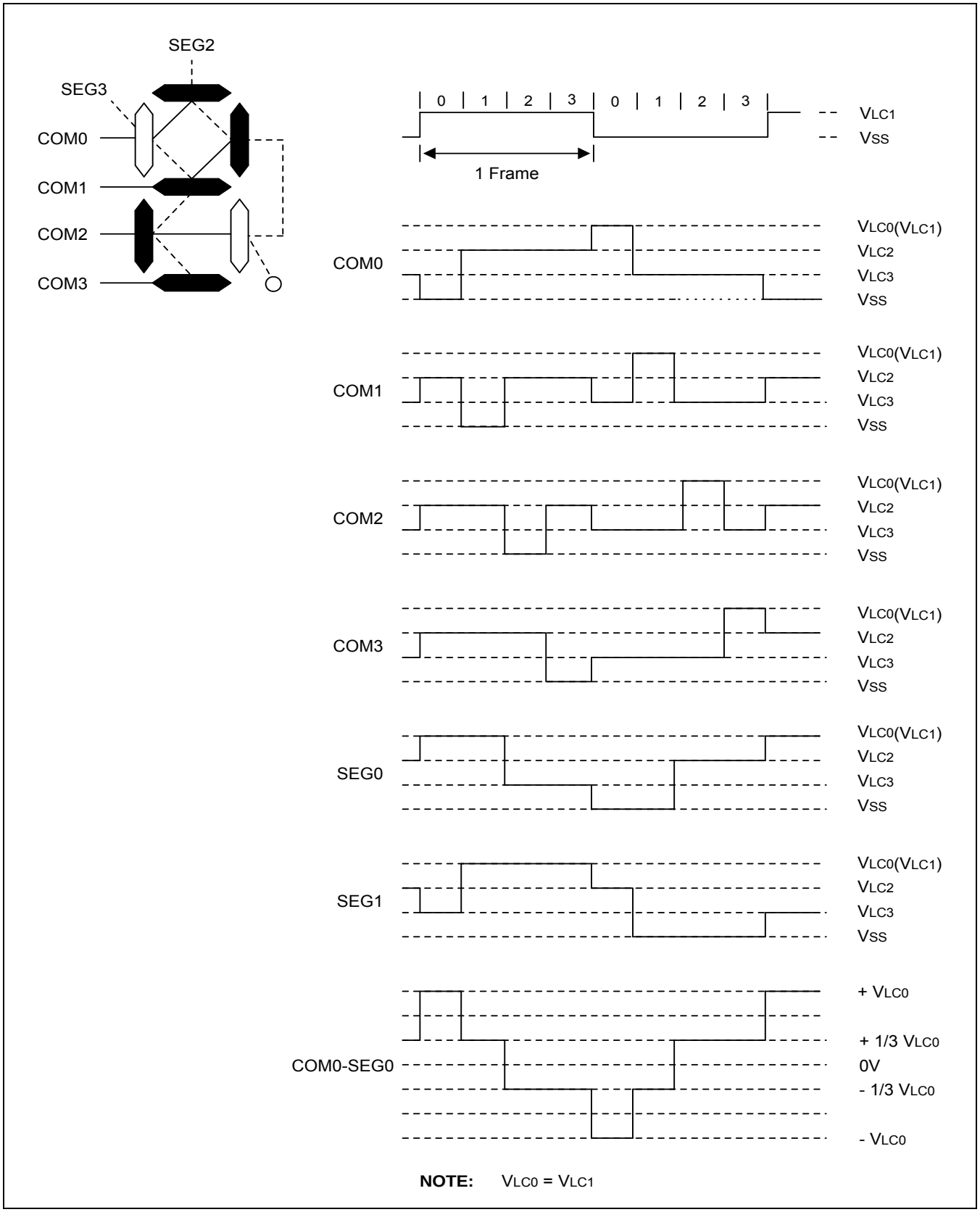


图 14-12 LCD 信号波形 (1/4 占空比, 1/3 偏压比)

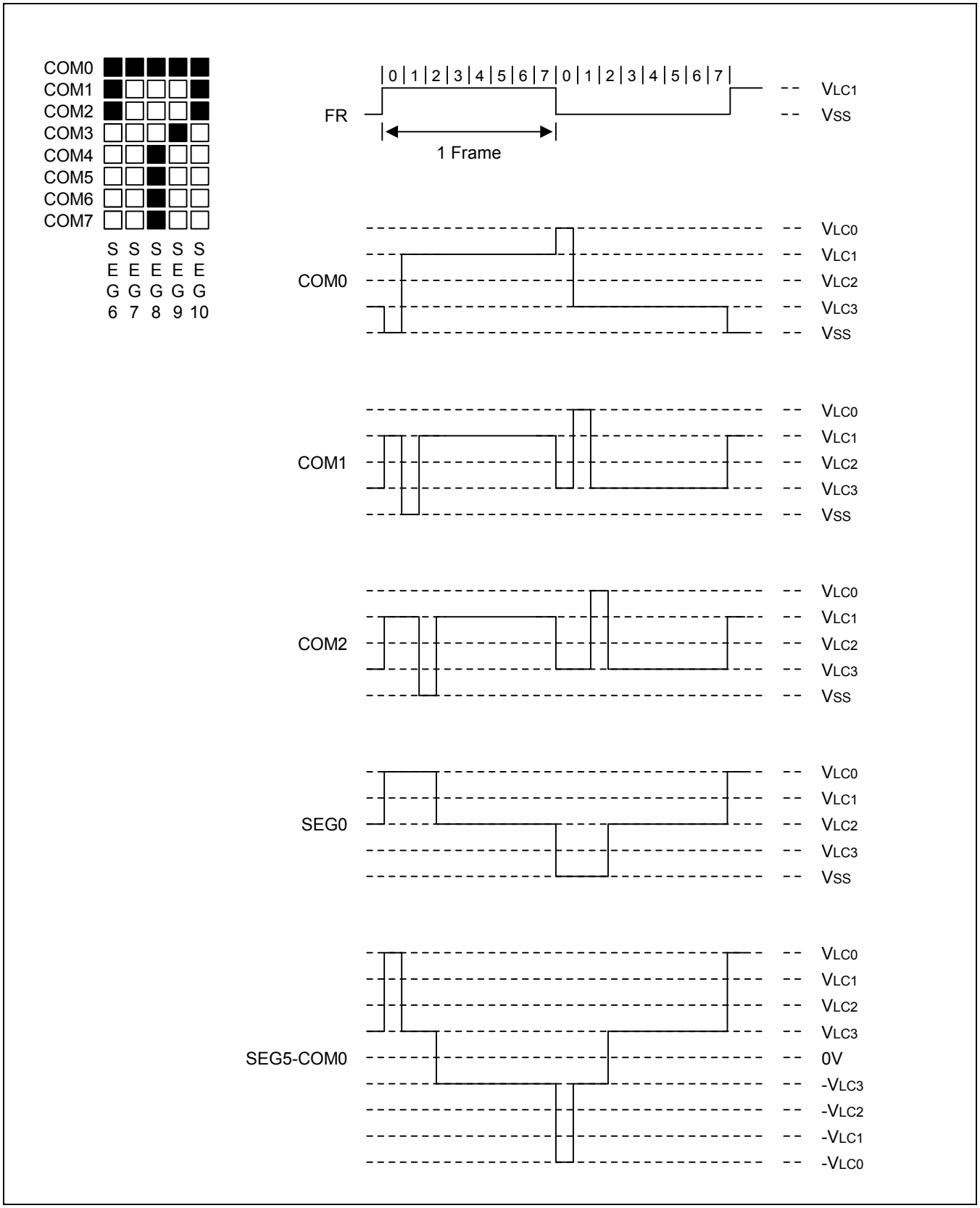


图 14-13 LCD 信号波形 (1/8 占空比, 1/4 偏压比)

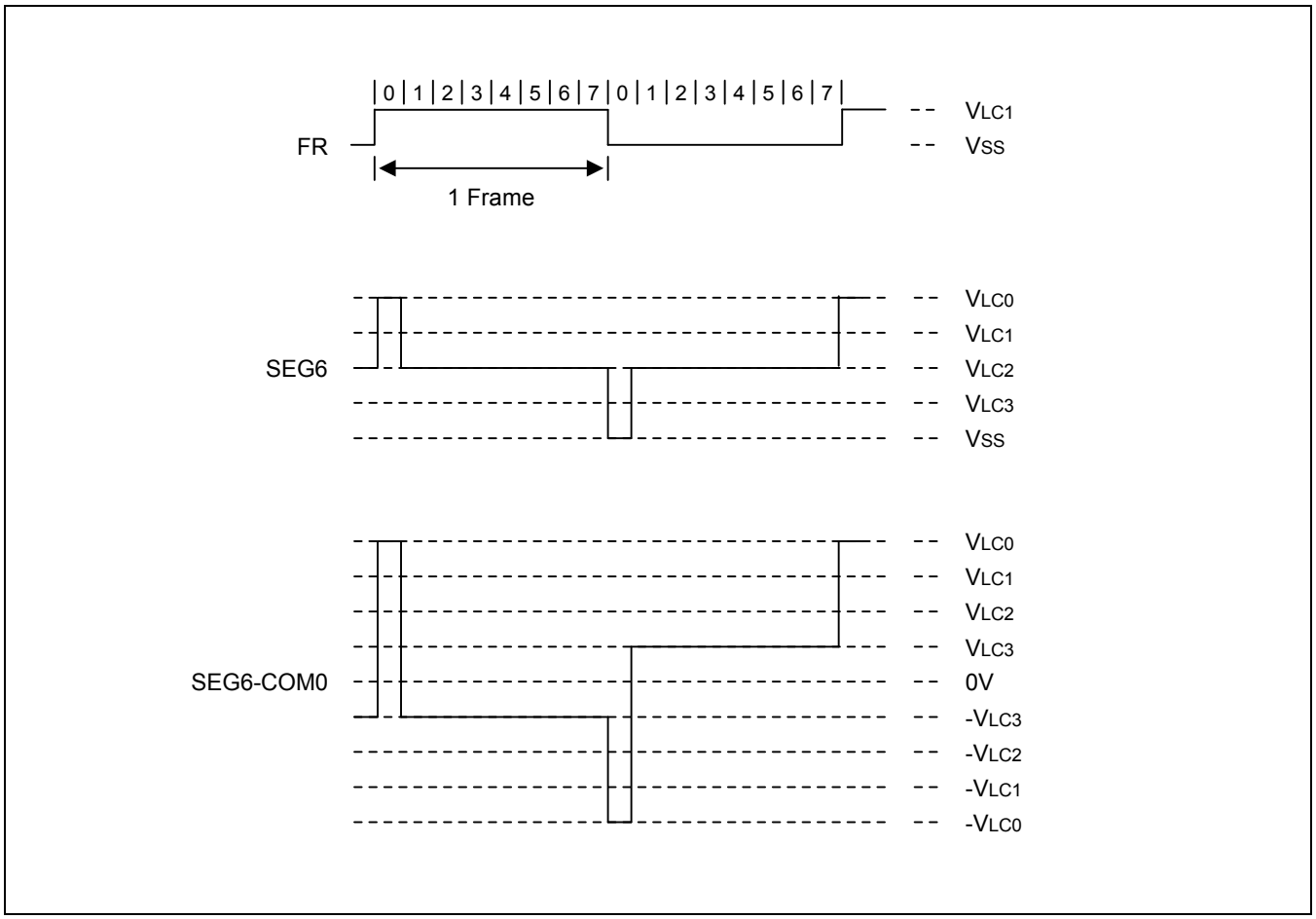


图 14-14 LCD 信号波形 (1/8 占空比, 1/4 偏压比) (续)

15 10 位 AD 转换器

15.1 概述

S3F82HB 内部集成了一个8通道10位 A/D 转换器 (ADC)，通过逐次逼近逻辑把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 AV_{SS} 之间。A/D 转换器由以下几个部分组成：

- 逐次逼近型模拟比较器
- D/A 转换逻辑 (电阻串型)
- ADC 控制寄存器 (ADCON)
- 8通道模拟信号输入端 (AD0–AD7)
- 10位 A/D 转换结果寄存器 (ADDATAH/L)
- 8位数字输入端口 (功能复用，I/O 口)
- AV_{REF} 和 AV_{SS} 管脚，AV_{SS} 内部连接到 V_{SS}

15.2 功能描述

启动一次模数转换，首先必须置位 ADCEN 信号来使能位于 P2口的 A/D 输入，设置为“1”则使能 ADC 模拟输入。通过设置 A/D 控制寄存器 ADCON.4–.6 选择8路模拟输入通道中的1路作为模拟输入，并置位转换使能位 ADCON.0。可读写的 ADCON 寄存器地址是0CH，page 0。未使用作 A/D 输入的管脚可用作一般 I/O。

在一个正常转换过程中，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 AD 转换，但是可以通过操作 ADCON 寄存器的 ADCON.6-.4 动态分时选择多路 A/D 输入。设置使能位 ADCON.0 位为“1”，则启动 A/D 转换。当 A/D 转换结束时，控制寄存器 ADCON.3 自动置“1”，即 EOC 被置“1”。A/D 转换结果保存于 ADDATAH/L 寄存器，A/D 转换电路随即进入空闲状态。
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: ADC 电路内部没有采样保存电路，所以在每次 A/D 转换时，此路模拟信号输入端的波动要非常小，否则，转换结果可能会不正确。如果在转换过程中进入 STOP 或者 IDLE 模式，A/D 模块会产生漏电流。必须在 A/D 转换结束后再使用 STOP 或者 IDLE 模式。

15.2.1 转换时间

A/D 转换1位需要4个 ADC 时钟周期，建立 A/D 转换需要10个时钟周期，因此 A/D 转换完10位一共需要50个时钟周期。在8MHz 频率，A/D 转换始终选择八分频的情况下，一个时钟周期为1us (8/fxx)。
则 A/D 转换所需要的时间为：

4 时钟周期/位 × 10位 + 建立时间 = 50 个时钟周期，50 个时钟周期×1us = 50μs 当 1MHz 时

15.2.2 A/D 转换控制寄存器 (ADCON)

A/D 转换控制寄存器 ADCON 的地址为 0CH，page 0，该寄存器可以实现以下4个功能：

- 选择模拟输入通道 (ADCON.6–.4)
- 转换结束状态检测 (ADCON.3)
- 选择 A/D 转换速度 (ADCON.2–.1)
- A/D 转换使能位 (ADCON.0)

复位后，使能位被关闭。一次只能转换一路模拟量。通过操作 ADCON 寄存器的 ADCON.6-.4 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一般 I/O。

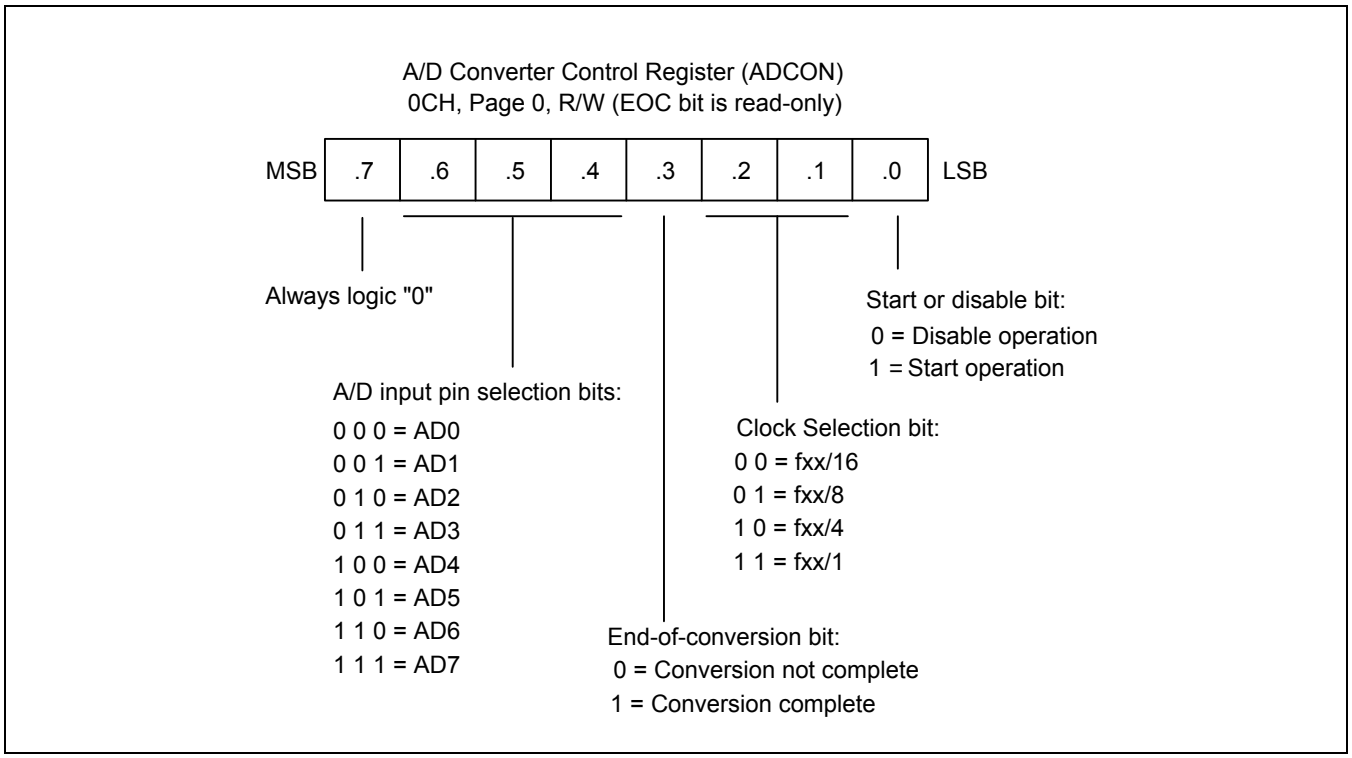


图 15-1 A/D 转换控制寄存器 (ADCON)

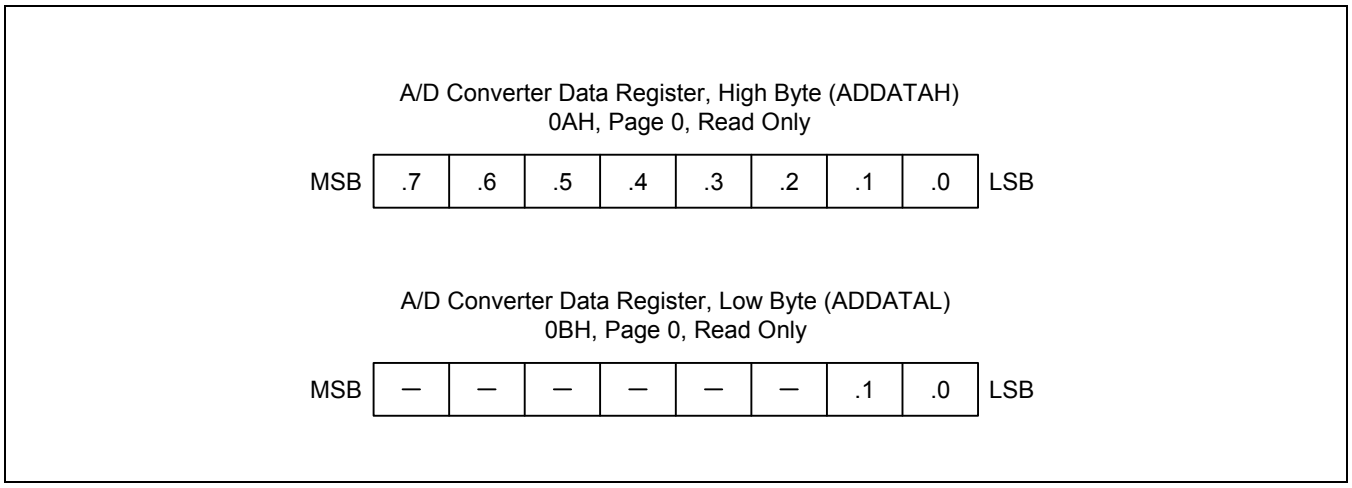


图 15-2 A/D 转换数据寄存器 (ADDATAH/L)

15.2.3 内部参考电压

在 A/D 转换中，输入模拟信号要与内部参考电压进行比较。输入模拟信号的电压范围应在 AV_{SS} 至 AV_{REF} 之间(通常, $AV_{REF} \leq V_{DD}$)。

不同的比较电压是由内部电阻网络分压产生的，开始的比较电压一般为1/2 AV_{REF}。

15.3 模块图

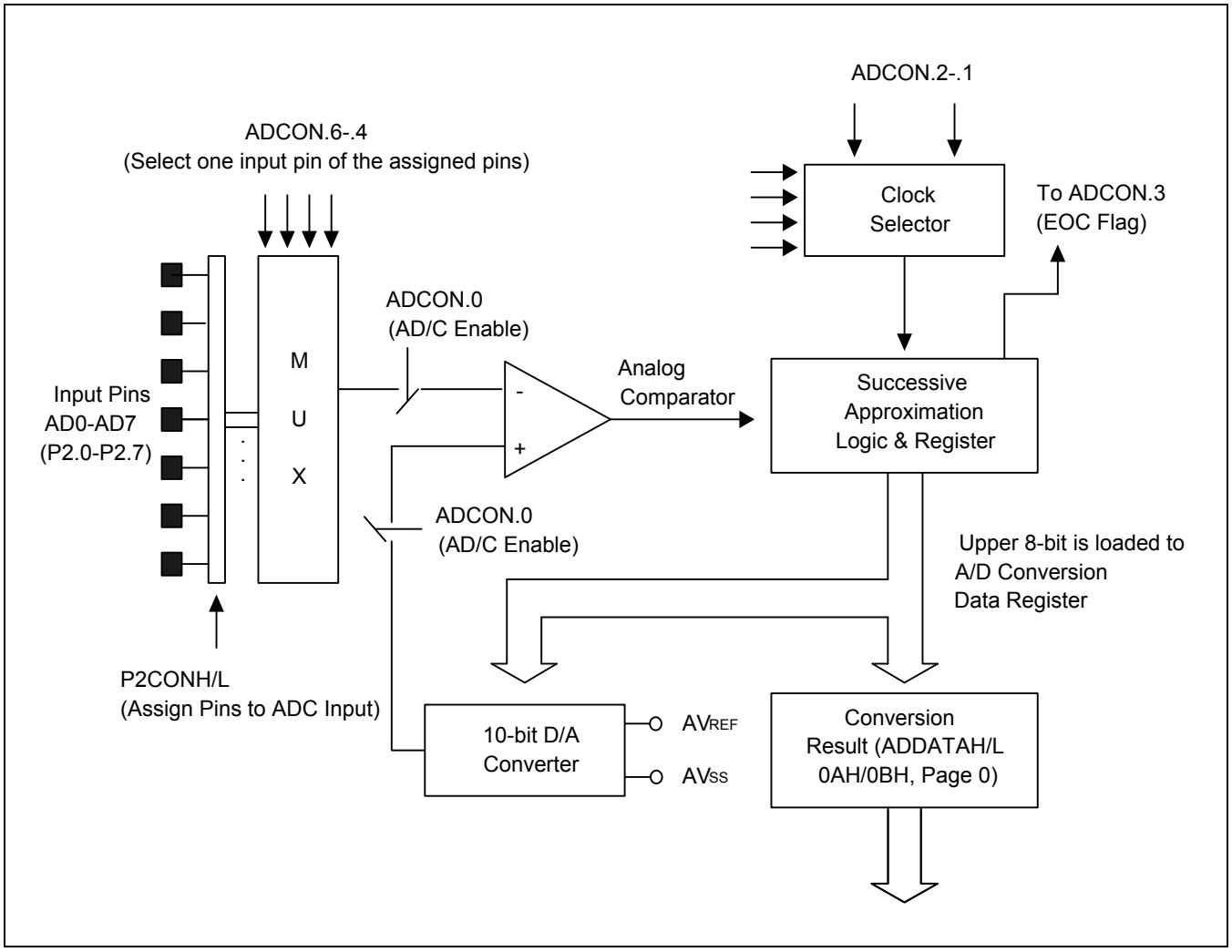


图 15-2 A/D 转换功能模块图

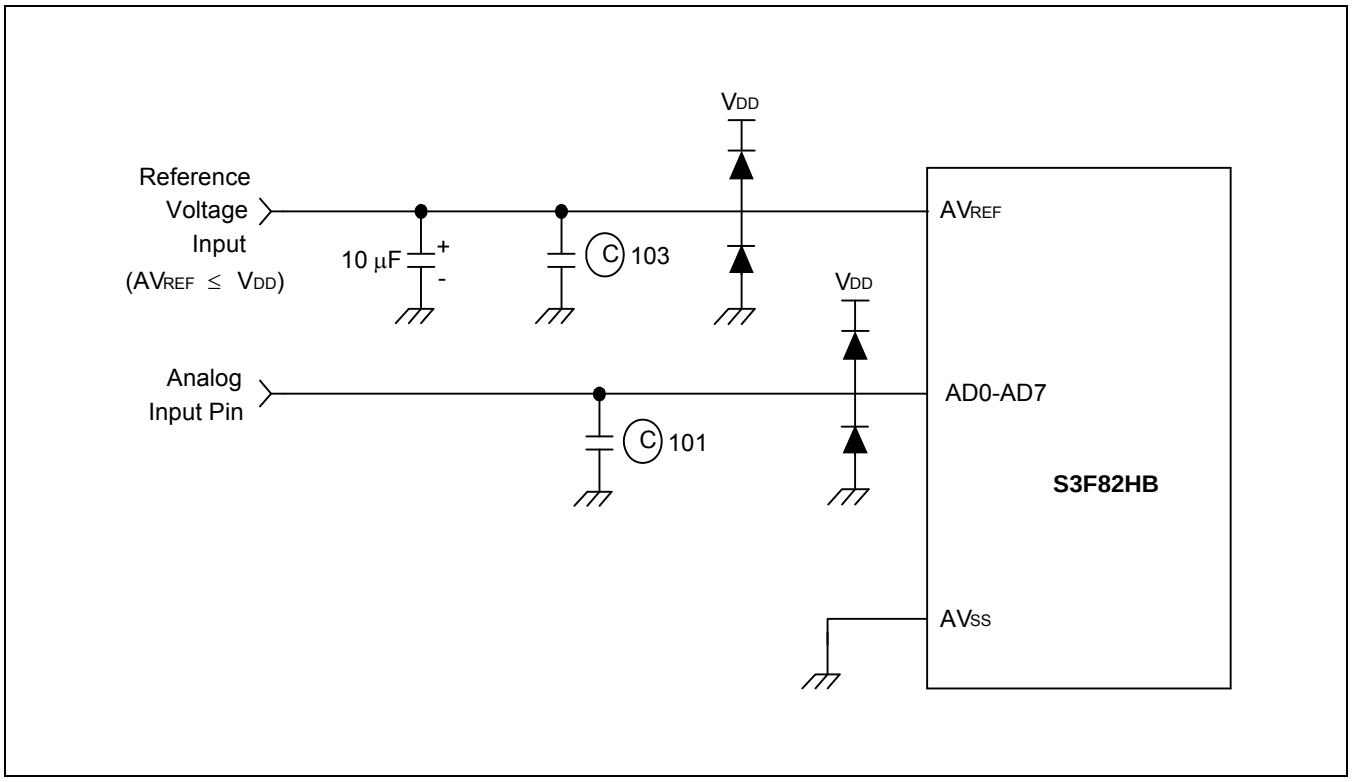


图 15-3 推荐高精度 A/D 转换电路

16 串行输入输出接口

16.1 概述

串行输入输出接口 (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)
- 外部时钟输入引脚 (SCK)

SIO 模块可以接收和发送8位数据，其接收，发送速率由寄存器设定。
为了实现灵活的数据传输速率，可以选择内部或者外部时钟源。

16.1.1 编程流程

SIO 模块的编程遵循以下基本的步骤：

1. 通过设置寄存器 P0CONH 寄存器，设置 P0口中 SIO 对应的 I/O 口为 SO，SCK，SI。
2. 设置寄存器 SIOCON，正确的配置 SIO 模块。SIOCON.2 必须设置为“1”以使能数据的移位。
3. 需要中断时，将中断使能位 (SIOCON.1) 置为“1”。
4. 发送数据时，将要发送的数据写到数据缓冲器 SIODATA，并将 SIOCON.3 设为“1”，则开始发送数据。
5. 当数据接收/发送完成后，SIO中断标志位 (SIOCON.0) 置为“1”，同时产生 SIO 中断请求。

16.1.1.1 SIO 控制寄存器 (SIOCON)

SIO 控制寄存器 SIOCON，地址为 E0H，Set 1，Bank 0。设置 SIO 模块的以下功能：

- 时钟源选择 (内部或者外部)
- 中断使能控制
- 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 (高位在前或低位在前)

复位后，SIOCON 的初始值为“00H”，默认 SIO 选择内部时钟，工作在仅接收模式下，3位时钟计数器清零。禁止数据移位操作 (发送) 和中断，选择数据方向为高位在前。

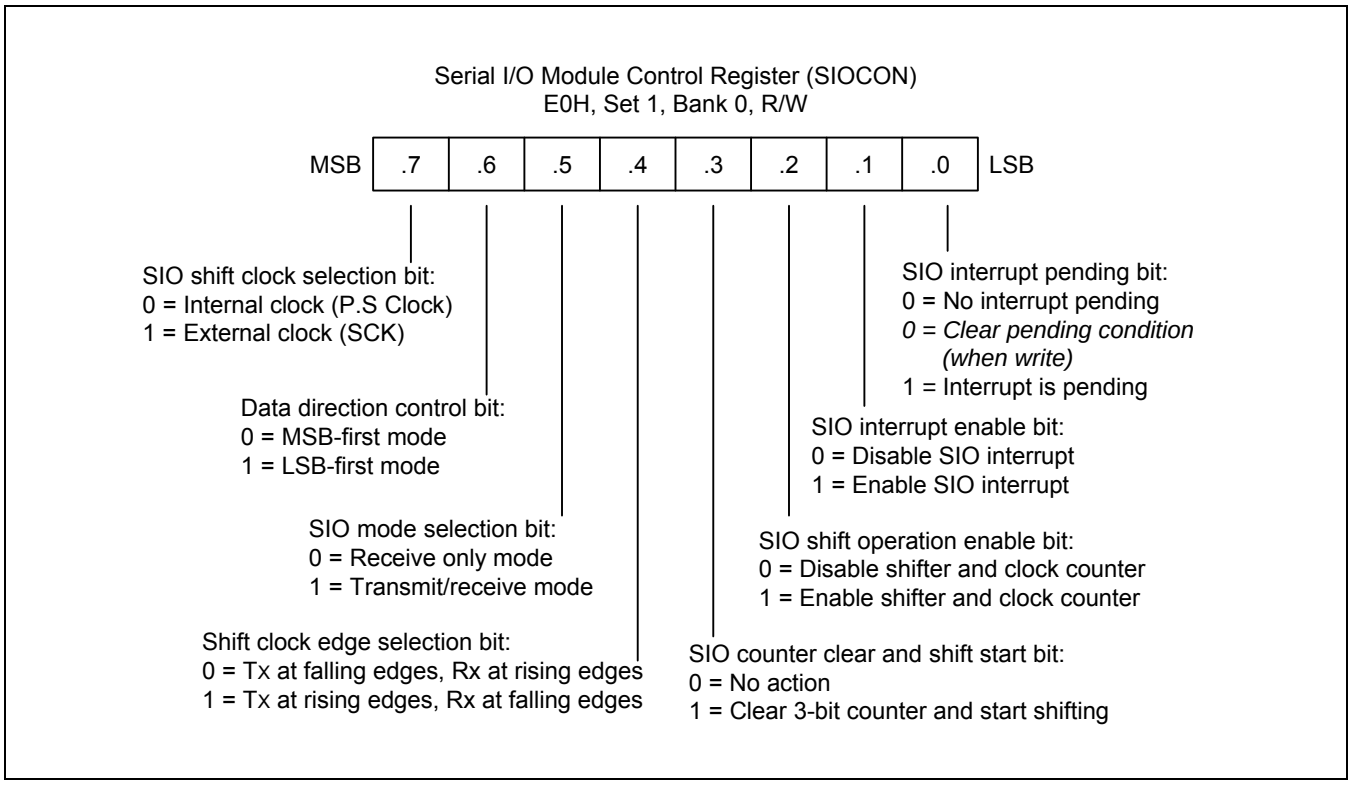


图 16-1 串行输入输出接口控制寄存器 (SIOCON)

16.1.1.2 SIO 预分频寄存器 (SIOPS)

SIO 预分频寄存器 SIOPS，地址为 F0H，Set 1， Bank 1。SIOPS 寄存器的值决定了 SIO 数据传输的波特率：

波特率 = 输入时钟频率(Xin)/4 / (预分频值+ 1)，或外部输入时钟频率，这里输入时钟为fxx/4。

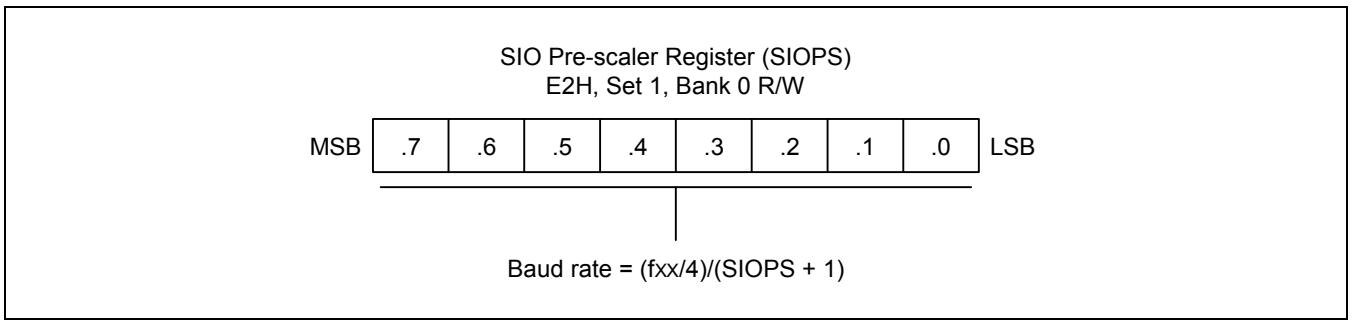


图 16-2 SIO 预分频寄存器 (SIOPS)

16.2 模块框图

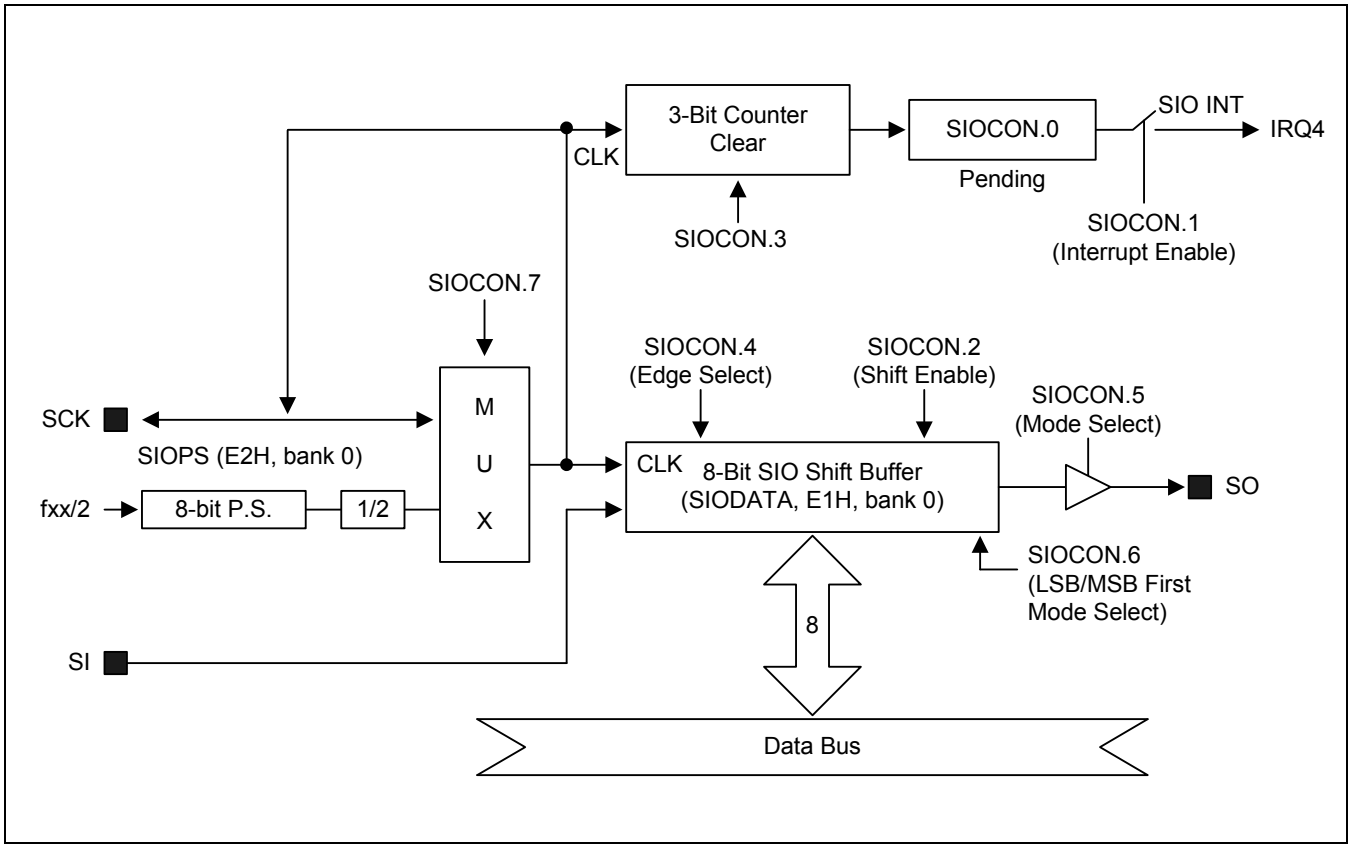


图 16-3 SIO 功能模块框图

16.2.1 SIO 时序图

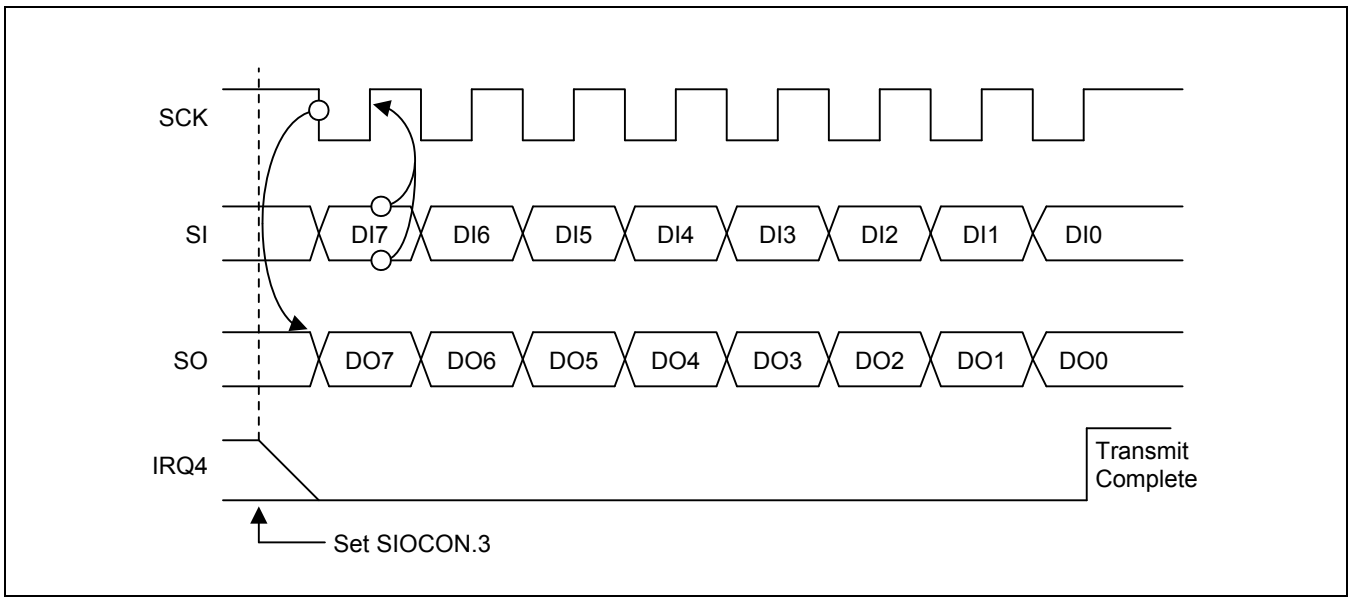


图 16-4 SIO串行发送-接收模式 (下降沿发送，SIOCON.4 = 0)

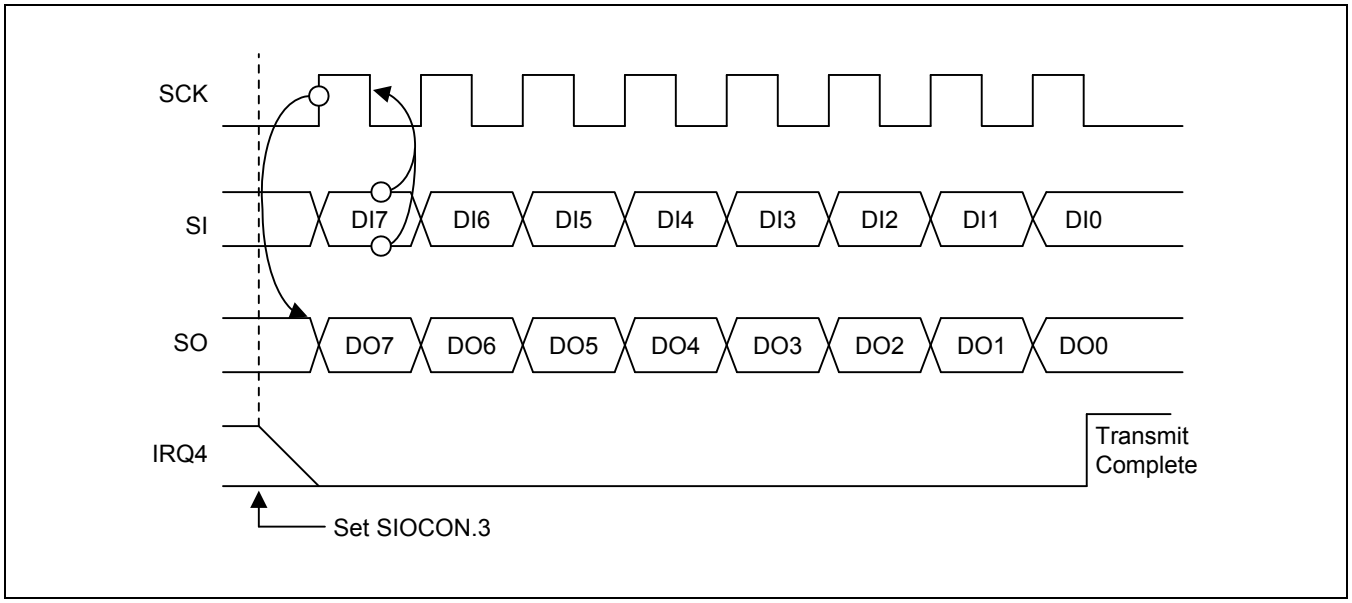


图 16-5 SIO串行发送-接收模式 (上升沿发送，SIOCON.4 = 1)

17 UART 0

17.1 概述

UART 0 模块是运行模式可设置的全双工串行通讯口。其运行模式包括一个并行模式和三个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 的接收和发送都是通过 UART 数据寄存器 UART0DATA (02H，page 0) 来实现的。写 UART 0 数据寄存器时，数据装载到 UART 0 的发送缓冲器中；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。

接收缓冲器为移位寄存器，在上一个接收的数据还未读出前，下一个数据已经开始接收。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，当任何指令将 UART0DATA 作为目标地址写入数据时，则立即开始发送。在 mode 0 模式下，只有当接收中断标志位 (UART0CONH.0) 为“0”和接收使能位 (UART0CONH.4) 为“1”时，才开始接收串行数据。在 mode 1，2和3模式下，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。

17.1.1 UART 0 模块编程的基本步骤为：

1. 通过设置 P3CONH 为相应的值，将 P3.7和 P3.6设为 UART 0 功能引脚 (RXD0 (P3.7)，TXD0 (P3.6))。
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 或 UART0CONL.1) 设为“1”。
4. 需要传送数据时，将需要传送的数据写入到 UART0DATA，则开始传送。
5. 当发送/接收完一个数据后，UART 0 标志位 (UART0CONH.0 或 UART0CONL.0) 被置为“1”，同时产生一个相应的发送/接收中断。

17.1.2 UART 0 控制寄存器高字节 (UART0CONH)

UART 0 的控制寄存器的高字节是 UART0CONH，地址为00H，Page 0。主要实现以下功能：

- 运行模式和波特率选择
- 多节点通讯和中断控制
- 串行接收使能/禁止控制
- Mode 2 和 Mode 3 模式下发送和接收时第9位数据位的位置
- UART 0 接收中断控制

复位后，UART0CONH 的值被清为“00H”，因此，如果需要使用 UART 0 模块，应设定 UART0CONH 为适当的值。

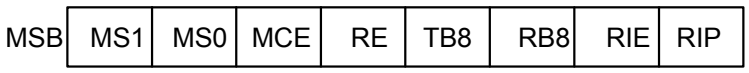
17.1.3 UART 0 控制寄存器低字节 (UARTCONL)

UART 0 的控制寄存器的低字节是 UART0CONL，地址为01H，Page 0。主要实现以下功能：

- UART 0 发送和接收的奇偶校验位选择
- UART 0 时钟选择
- UART 0 发送中断控制

复位后，UART0CONL 的值被清为“00H”，因此，如果需要使用 UART 0 模块，应设定 UART0CONL 为适当的值。

UART 0 Control Register, High Byte (UART0CONH)
00H, Page 0, R/W



Operating mode and baud rate selection bits: (see table below)

Multiprocessor communication (1) enable bit (for modes 2 and 3 only):
0 = Disable
1 = Enable

Serial data receive enable bit:
0 = Disable
1 = Enable

Uart 0 receive interrupt pending bit:
0 = No interrupt pending(when read), clear pending bit(when write)
1 = Interrupt is pending(when read)

Uart 0 receive interrupt enable bit:
0 = Disable Rx interrupt
1 = Enable Rx interrupt

RB8(3) (Only when UART0CONL.7 = 0):
Location of the 9th data bit that was received in UART 0 mode 2 or 3 ("0" or "1")

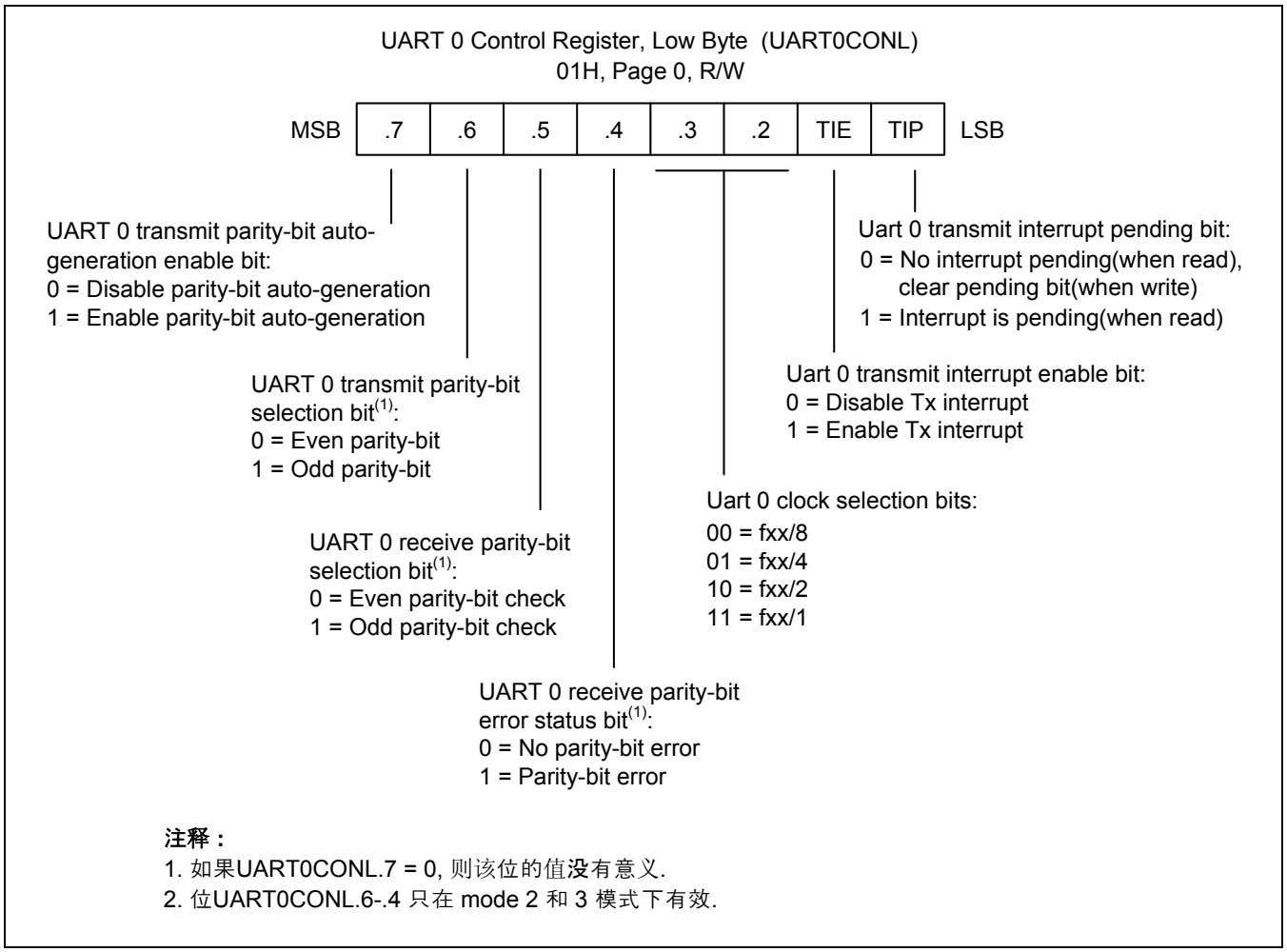
TB8(3) (Only when UART0CONL.7 = 0):
Location of the 9th data bit to be transmitted in UART 0 mode 2 or 3 ("0" or "1")

MS1	MS0	Mode	Description(2)	Baud Rate
0	0	0	Shift register	(fu/(16 x (BR0DATA + 1)))
0	1	1	8-bit UART	(fu/(16 x (BR0DATA + 1)))
1	0	2	9-bit UART	(fu /16)
1	1	3	9-bit UART	(fu/(16 x (BR0DATA + 1)))

注释:

1. 在 mode 2 或者 mode 3 模式下，如果设置 UART0CONH.5 位的值为“ 1”，则当收到的第9位数据位为“ 0”时，不会产生接收中断。在 mode 1 模式下，如果设置 UART0CONH.5 位的值为“ 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 mode 0 模式下，应将 UART0CONH.5 设为“ 0”
2. 有关UART的8位和9位的描述中，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, 表示该位的值可以为任意值。

图 17-1 UART 0 控制寄存器高字节 (UART0CONH)



注释：

1. 如果UART0CONL.7 = 0, 则该位的值没有意义.
2. 位UART0CONL.6-.4 只在 mode 2 和 3 模式下有效.

图 17-2 UART 0 控制寄存器低字节 (UART0CONL)

17.1.4 UART 0 中断标志位

在 mode 0 模式下，当收到第8位数据位后，接收中断标志位 UART0CONH.0 被置为“1”。在 mode 1模式下，UART0CONH.0 在停止位的中间点置“1”。在 mode 2或 mode 3模式下，在接收到 RB8位的中间点时，UART0CONH.0 置“1”。当 CPU 响应了接收中断后，UART0CONH.0 必须在中断服务程序中软件清零。

在 UART 0 的 mode 0模式下，当发送完第8位数据后，发送中断标志位 UART0CONL.0被置为“1”。在 mode 1，mode 2或 mode 3模式下， UART0CONL.0在停止位开始发送时置“1”。
当CPU响应了发送中断后，UART0CONL.0必须在中断服务程序中软件清零。

17.1.5 UART 0 数据寄存器 (UART0DATA)

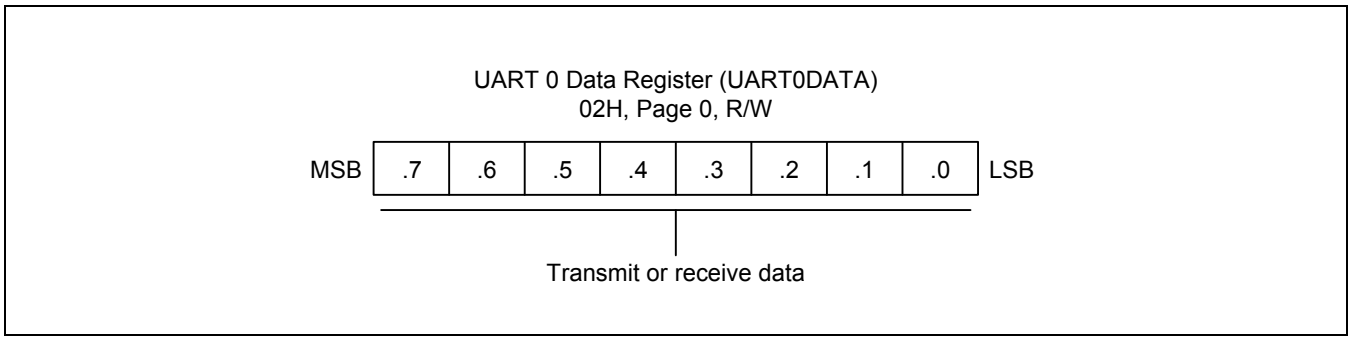


图 17-3 UART 0 数据寄存器 (UART0DATA)

17.1.6 UART 0 波特率数据寄存器 (BR0DATA)

通过设置 UART 0 波特率数据寄存器，BR0DATA，可以设定 UART 0 的时钟速率 (即波特率)。

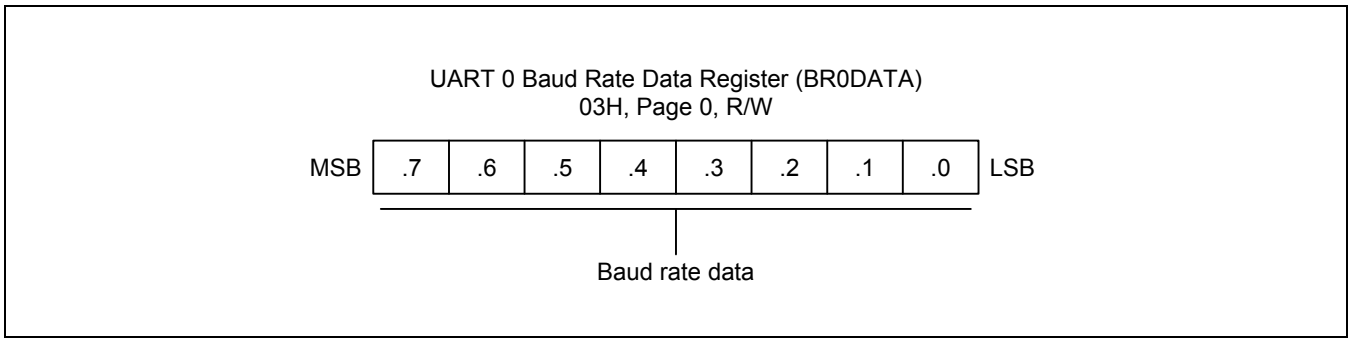


图 17-4 UART 0 波特率数据寄存器 (BR0DATA)

17.1.7 波特率计算

17.1.7.1 Mode 0 模式下波特率的计算

在 mode 0 模式下，波特率由波特率数据寄存器，BR0DATA (03H，Page 0) 的值决定，mode 0模式下波特率为：$f_U/(16\times(BR0DATA+1))$

17.1.7.2 Mode 2 模式下波特率的计算

Mode 2 模式下的波特率固定为 $f_U/16$，即 f_U 信号16分频。

17.1.7.3 Mode 1 和Mode 3 模式下波特率的计算

在 mode 1和 mode 3模式下，波特率由波特率数据寄存器，BR0DATA (03H，Page 0) 的值决定，mode 1和mode 3模式下波特率为：$f_U/(16\times(BR0DATA+1))$

表 17-1 8位 BR0DATA可以产生的常用波特率表

模式	波特率	UART 时钟 (fU)	BR0DATA	
			十进制	十六进制
Mode 2	0.5MHz	8MHz	x	x
Mode 0 Mode 1 Mode 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

17.1.8 模块框图

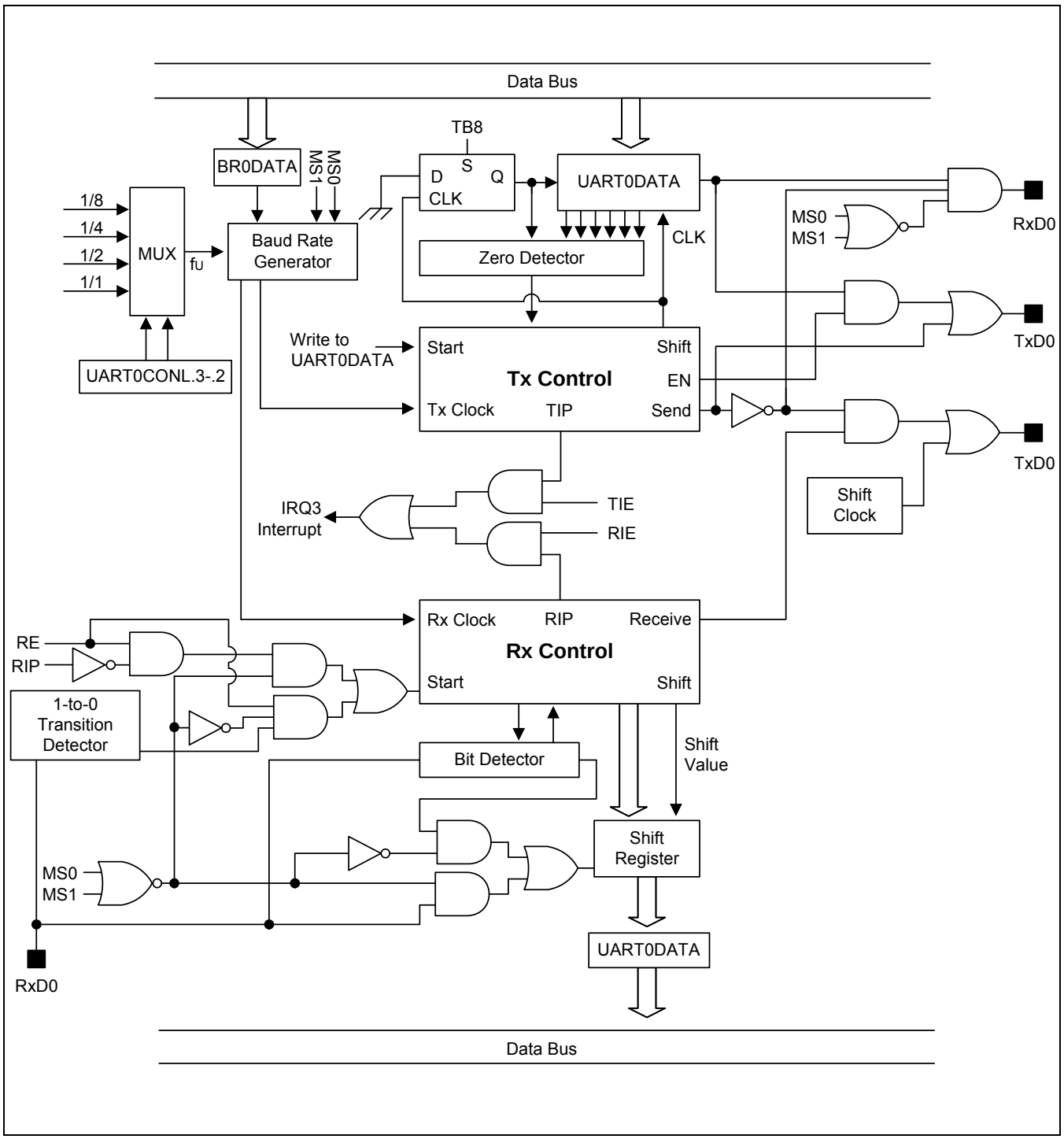


图 17-5 UART 0 功能模块框图

17.1.9 UART 0 MODE 0 模式功能描述

在 mode 0 模式下，UART 0 通过RxD0 (P3.7) 接收和发送数据，TxD0 (P3.6) 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低位 (LSB)。

17.1.9.1 Mode 0 模式数据发送流程：

1. 设置 UART0CONL.3和.2，选择 UART 0 时钟源。
2. 设置 UART 0 禁止或使能发送时奇偶校验自动产生功能 (UART0CONL.7)。
3. 设置 UART0CONH.7-.6 为“00B”，选择 mode 0。
4. 将需要发送的数据写入 UART0DATA (02H，page0)，数据开始发送。

17.1.9.2 Mode 0 模式数据接收流程：

1. 设置 UART0CONL.3 和.2，选择 UART 0 时钟源。
2. 设置 UART 0禁止或使能发送时奇偶校验自动产生功能 (UART0CONL.7)。
3. 设置 UART0CONH.7-.6为“00B”，选择 mode 0。
4. 写“0”到 UART0CONH.0，清除接收中断标志位。
5. 设置 UART 0 接收使能位 (UART0CONH.4) 为“1”，使能UART 0 接收。
6. 时钟移位信号开始输出到 TxD0 (P3.6)，同时开始从RxD0 (P3.7) 读取数据。
   当 UART0CONH.1置为“1”时，产生UART 0 接收中断。

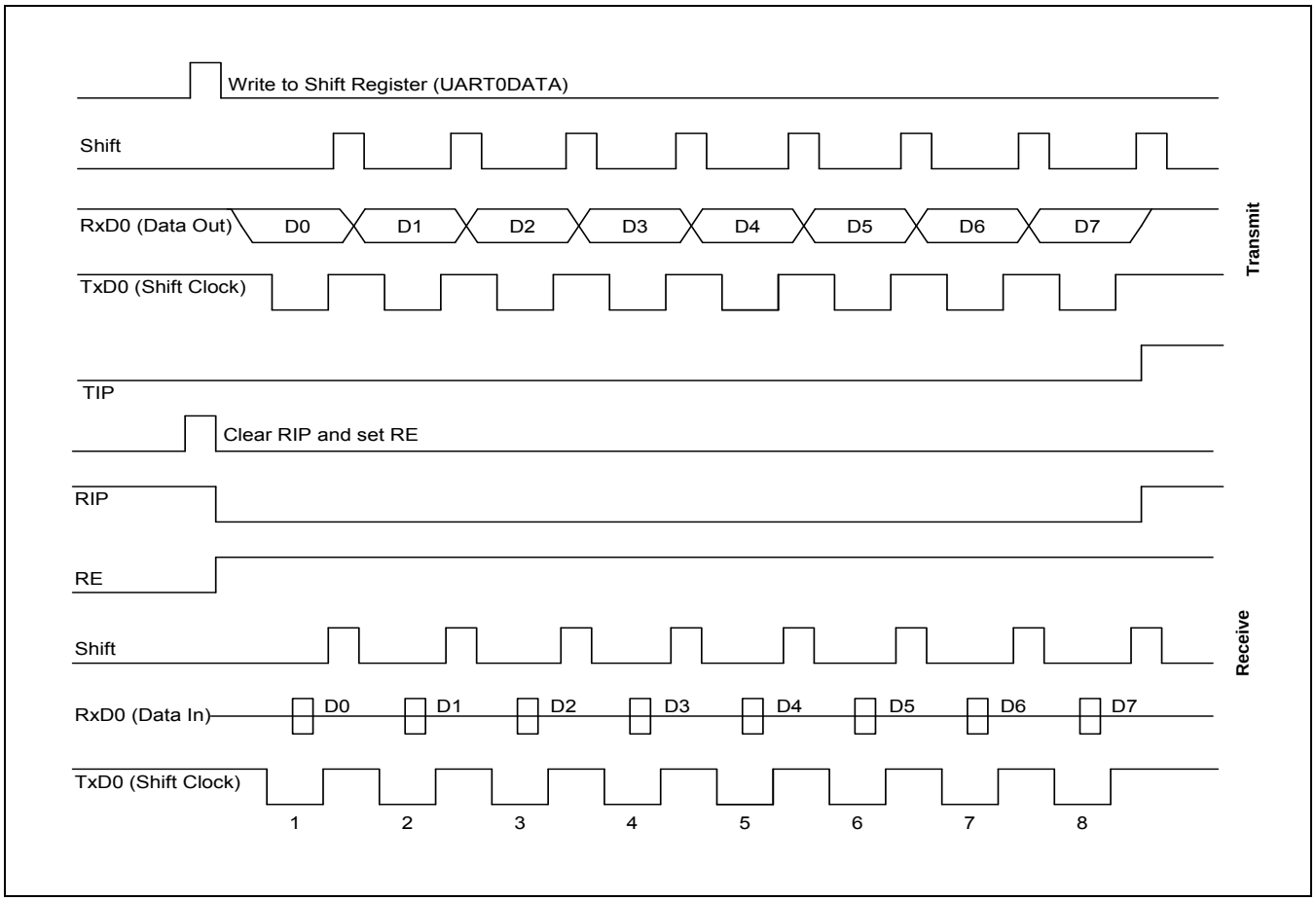


图 17-6 UART 0 Mode 0 模式运行时序图

17.1.10 UART 0 MODE 1 模式功能描述

在 mode 1 模式下，发送 (通过TxD0 (P3.6)) 或者接收 (通过RxD0 (P3.7))
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 (低位在前)
- 停止位（“1”）

Mode 1模式的波特率是可变的。

17.1.10.1 Mode 1 模式数据发送流程

1. 设置 UART0CONL.3 和.2，选择 UART 0 时钟源。
2. 设置 UART 0 禁止或使能发送时奇偶校验自动产生功能 (UART0CONL.7)。
3. 通过设置8位 BR0DATA 寄存器设定波特率。
4. 设置 UART0CONH.7-.6为“01B”，选择 mode 1 (8位 UART)。
5. 将需要发送的数据写入 UART0DATA (02H，page0)，数据开始发送。起始位和停止位由硬件自动产生。

17.1.10.2 Mode 1 模式数据接收流程

1. 设置 UART0CONL.3 和.2，选择 UART 0 时钟源。
2. 设置 UART 0禁止或使能发送时奇偶校验自动产生功能 (UART0CONL.7)。
3. 通过设置8位 BR0DATA 寄存器设定波特率。
4. 选择 mode 1 模式并设置 UART0CONH 的 RE (接收使能) 位为“1”。
5. 当 RxD0 (P3.7) 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。

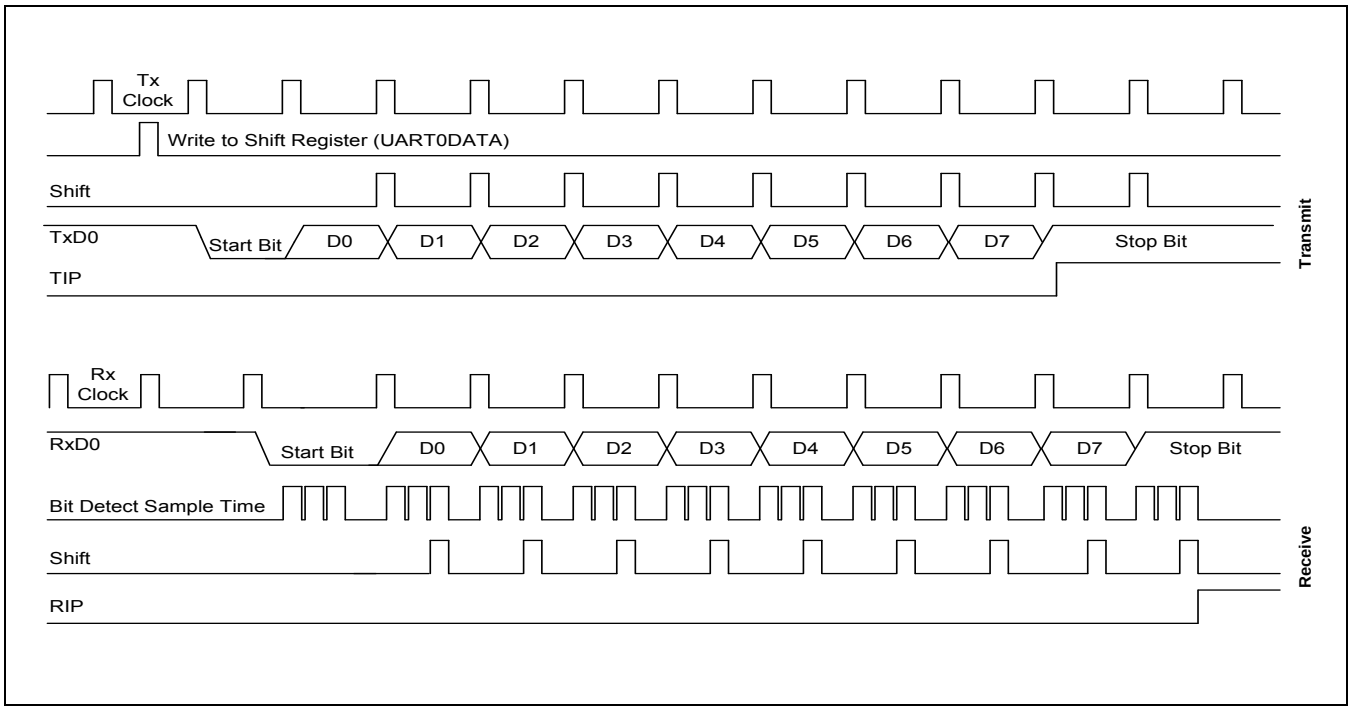


图 17-7 UART 0 mode 1 模式运行时序图

17.1.11 UART 0 MODE 2 模式功能描述

在 mode 2 模式下，发送 (通过 TxD0 (P3.6)) 或者接收 (通过 RxD0 (P3.7)) 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 (低位在前)
- 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”的写入到 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。Mode 2 模式的波特率为 $f_U/16$。

17.1.11.1 Mode 2 模式发送数据流程

1. 设置 UART0CONL.3 和.2，选择 UART 0 时钟源。
2. 设置 UART 0 禁止或使能发送时奇偶校验自动产生功能 (UART0CONL.7)。
3. 设置 UART0CONH.7-.6为“10B”，选择 mode 2 (9位 UART)，同时将需要发送的第9位数据写入 TB8。
4. 将需要发送的数据写入 UART0DATA (02H，page0)，数据开始发送。

17.1.11.2 Mode 2 模式接收数据流程

1. 设置 UART0CONL.3 和.2，选择 UART 0 时钟源。
2. 设置 UART 0 禁止或使能发送时奇偶校验自动产生功能 (UART0CONL.7)。
3. 选择 mode 2 模式并设置 UART0CONH的 RE (接收使能) 位为“1”。
4. 当 RxD0 (P3.7) 管脚上的电平变为低时，开始接收数据。

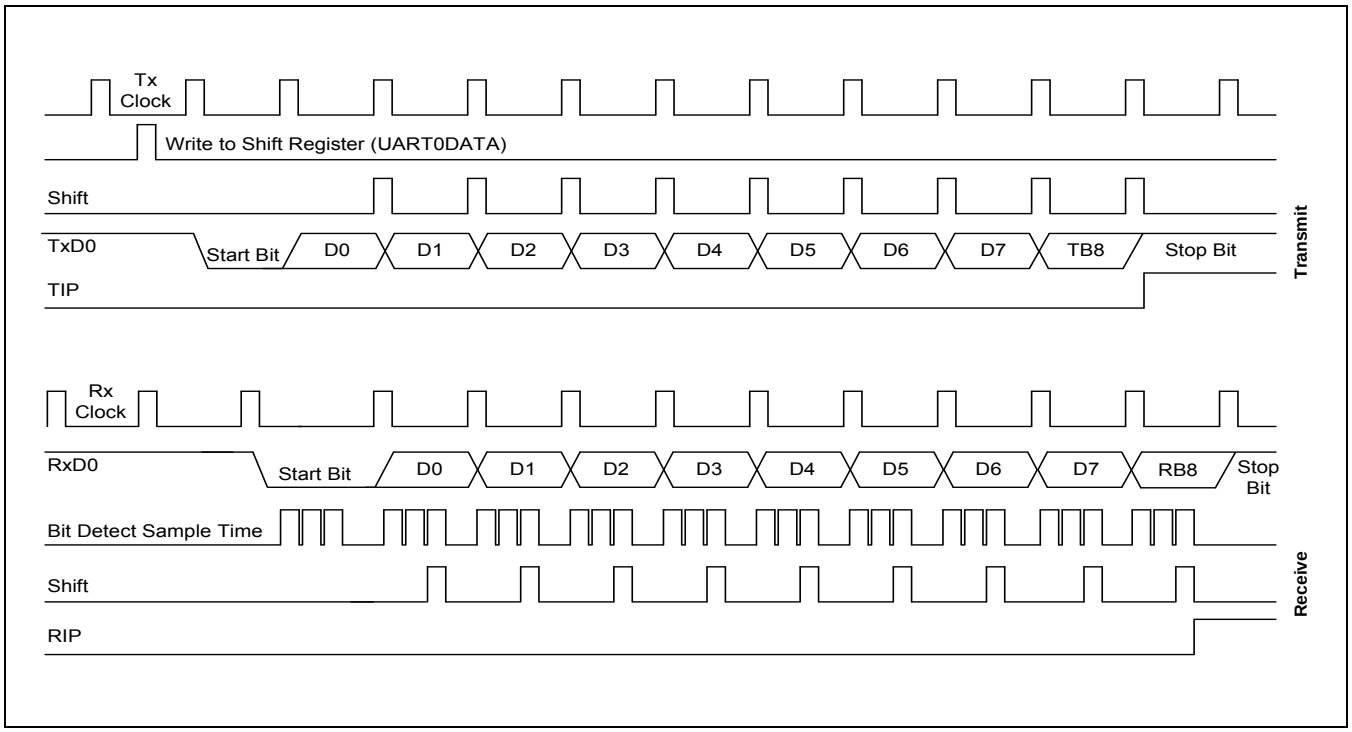


图 17-8 UART 0 mode 2 模式运行时序图

17.1.12 UART 0 MODE 3 模式功能描述

在 mode 3 模式下，发送 (通过TxD0 (P3.6)) 或者接收 (通过RxD0 (P3.7)) 的数据帧总长度是11位。除了mode 3 模式的波特率可以调节之外，其余的功能 mode 3 和 mode 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 (低位在前)
- 可编程设置的第9位数据位
- 停止位 (“1”)

17.1.12.1 Mode 3 模式发送数据流程

1. 设置 UART0CONL.3 和.2，选择 UART 0 时钟源。
2. 设置 UART 0 禁止或使能发送时奇偶校验自动产生功能 (UART0CONL.7)。
3. 设置 UART0CONH.7-.6为“11B”，选择mode 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 UART0DATA (02H，page0)，数据开始发送。

17.1.12.2 Mode 2 模式接受数据流程

1. 设置 UART0CONL.3 和.2，选择 UART 0 时钟源。
2. 设置 UART 0 禁止或使能发送时奇偶校验自动产生功能 (UART0CONL.7)。
4. 选择 mode 3 模式并设置 UART0CONH的 RE (接收使能) 位为“1”。
5. 当 RxD0 (P3.7) 管脚上的电平变为低时，开始接收数据。

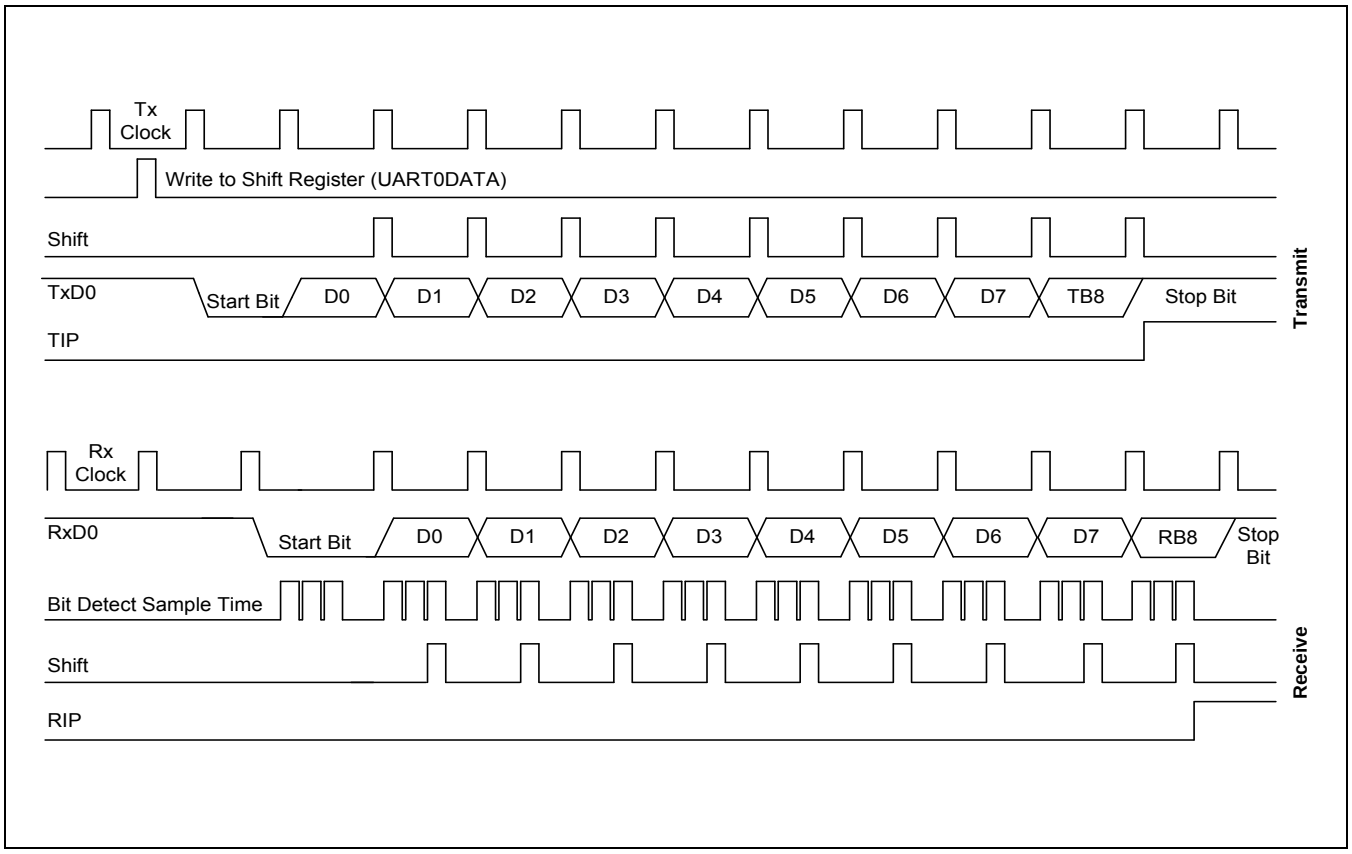


图 17-9 UART 0 mode 3 模式运行时序图

17.1.13 多节点串行通讯

S3F8 系列的多节点通讯功能的特点是：在一个由多个 S3F82HB 组成的串行通信系统中，允许一个S3F82HB作为主机向另一个特定的从机设备发送串行数据，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 mode 2 和 mode 3 模式下，禁止奇偶校验时才可以实现。在 mode 2 和 mode 3 模式下，收到的数据供9位，第9位写入了 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。

17.1.13.1 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 mode 0 模式下没有用处。在 mode 1 模式下，它可以用来检测停止位的有效性。在 mode 1模式下接收数据时，如果设置 MCE 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。

17.1.13.2 多节点通讯的建立流程

多节点通讯的建立流程如下：

1. 设置所有的串行总线的 S3F82HB设备的 UART 模式为 mode 2 或者 mode 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”)
   - 下一个字节：数据
     (第9位 =“0”)
4. 由于第9位为“1”，所有的从机都会产生中断来接收第一个字节。当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 位清为“0”来准备接受数据。其他的从机继续正常运行。

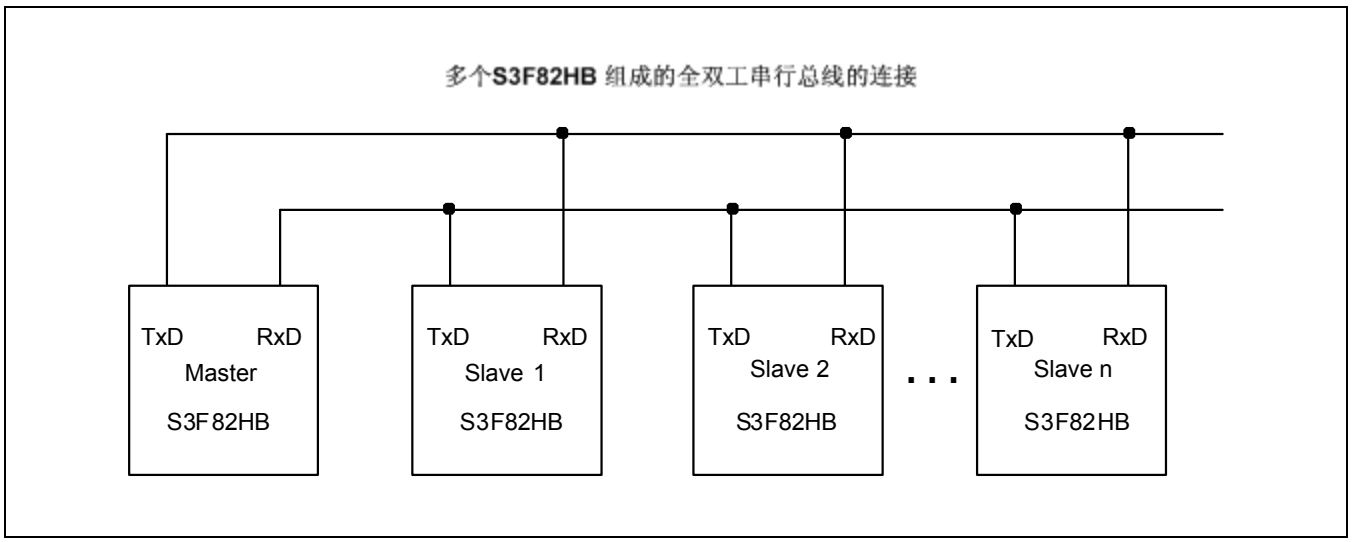


图 17-10 多节点串行通讯连接示例

18 UART 1

18.1 概述

UART 1 模块是运行模式可设置的全双工串行通讯口。其运行模式包括一个并行模式和三个 UART (通用异步串行接收/发送) 模式：

- 串行 I/O 口，波特率为 $f_U/(16\times(BR1DATA+1))$
- 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 的接收和发送都是通过 UART 数据寄存器 UART1DATA (06H，Page 0) 来实现的。写 UART 1 数据寄存器时，数据装载到 UART 1 的发送缓冲器中；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。

接收缓冲器为移位寄存器，在上一个接收的数据还未读出前，下一个数据已经开始接收。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，当任何指令将 UART1DATA 作为目标地址写入数据时，则立即开始发送。在 mode 0 模式下，只有当接收中断标志位 (UART1CONH.0) 为“0”和接收使能位 (UART1CONH.4) 为“1”时，才开始接收串行数据。在 mode 1，2和3模式下，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。

18.1.1 UART 1 模块编程的基本步骤为：

1. 通过设置 P1CONH/L 为相应的值，将 P1.4和 P1.3设为 UART 1功能引脚 (RXD1 (P1.4)，TXD1 (P1.3))。
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或 UART1CONL.1) 设为“1”。
4. 需要传送数据时，将需要传送的数据写入到 UART1DATA，则开始传送。
5. 当发送/接收完一个数据后，UART 1 标志位 (UART1CONH.0或UART1CONL.0) 被置为‘1’，同时产生一个相应的发送/接收中断。

18.1.2 UART 1 控制寄存器高字节 (UART1CONH)

UART 1的控制寄存器的高字节是 UART1CONH，地址为04H，Page 0。主要实现以下功能：

- 运行模式和波特率选择
- 多节点通讯和中断控制
- 串行接收使能/禁止控制
- Mode 2和 Mode 3模式下发送和接收时第9位数据位的位置。
- UART 1接收中断控制

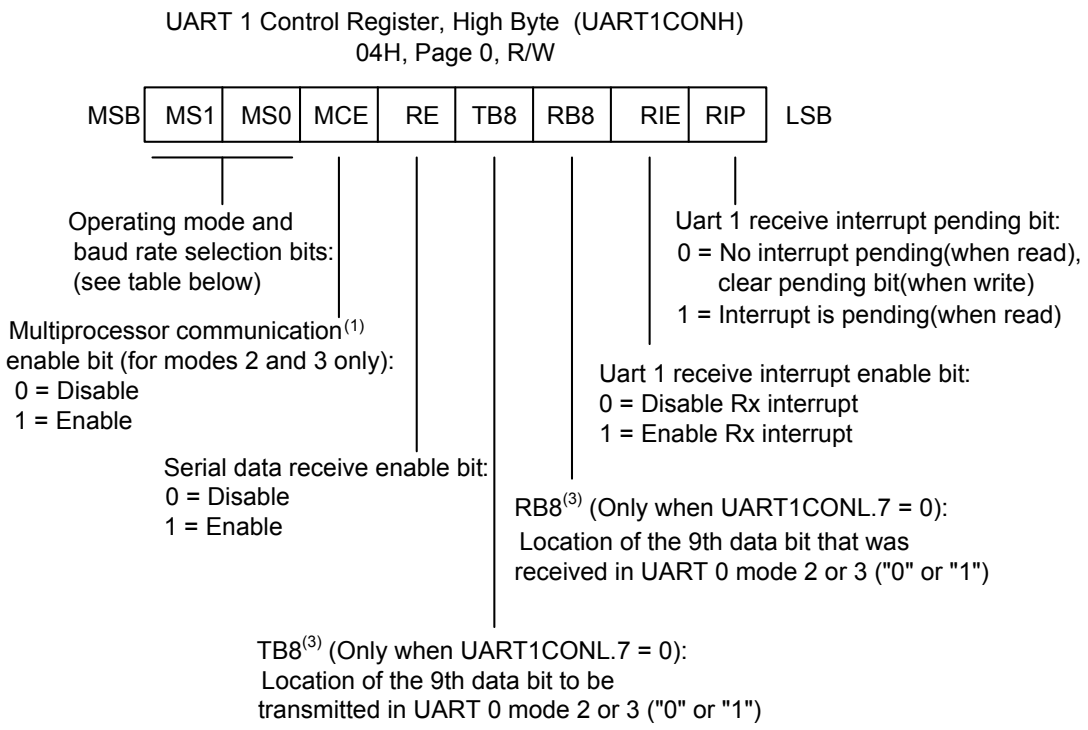
复位后，UART1CONH 的值被清为“00H”，因此，如果需要使用 UART 1模块，应设定 UART1CONH 为适当的值。

18.1.3 UART 1 控制寄存器低字节 (UART1CONL)

UART 1的控制寄存器的低字节是 UART1CONL，地址为05H，Page 0。主要实现以下功能：

- UART 1发送和接收的奇偶校验位选择
- UART 1时钟选择
- UART 1发送中断控制

复位后，UART1CONL 的值被清为“00H”，因此，如果需要使用 UART 1模块，应设定 UART1CONL 为适当的值。

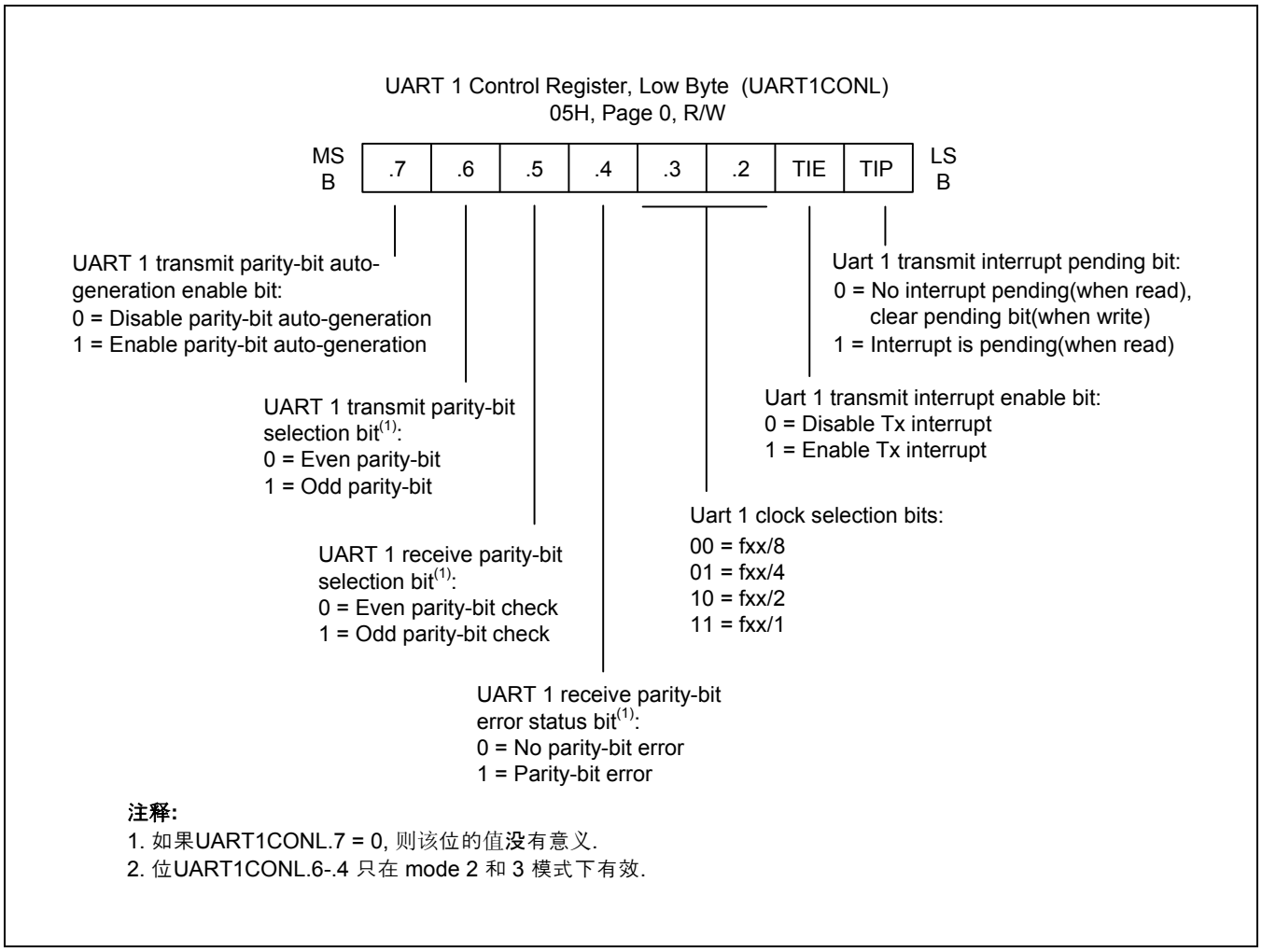


MS1	MS0	Mode	Description(2)	Baud Rate
0	0	0	Shift register	(fu/(16 x (BR0DATA + 1)))
0	1	1	8-bit UART	(fu/(16 x (BR0DATA + 1)))
1	0	2	9-bit UART	(fu /16)
1	1	3	9-bit UART	(fu/(16 x (BR0DATA + 1)))

注释:

1. 在 mode 2 或者 mode 3 模式下，如果设置 UART1CONH.5位的值为“ 1”，则当收到的第9位数据位为“ 0”时，不会产生接收中断。在 mode 1 模式下，如果设置 UART1CONH.5位的值为“ 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 mode 0 模式下，应将 UART1CONH.5设为“ 0”。
2. 有关 UART 的8位和9位的描述中，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，表示该位的值可以为任意值。

图 18-1 UART 1 控制寄存器高字节 (UART1CONH)



注释:
1. 如果UART1CONL.7 = 0, 则该位的值没有意义.
2. 位UART1CONL.6-.4 只在 mode 2 和 3 模式下有效.

图 18-2 UART 1 控制寄存器低字节 (UART1CONL)

18.1.4 UART 1 中断标志位

在 mode 0 模式下，当收到第8位数据位后，接收中断标志位 UART1CONH.0 被置为“1”。在 mode 1模式下，UART1CONH.0在停止位的中间点置“1”。在 mode 2或 mode 3模式下，在接收到RB8位的中间点时，UART1CONH.0置“1”。
当 CPU 响应了接收中断后，UART1CONH.0必须在中断服务程序中软件清零。

在 UART 1的 mode 0模式下，当发送完第8位数据后，发送中断标志位 UART1CONL.0被置为“1”。在 mode 1，mode 2或 mode 3模式下，UART1CONL.0在停止位开始发送时置“1”。
当 CPU 响应了发送中断后，UART1CONL.0必须在中断服务程序中软件清零。

18.1.5 UART 1 数据寄存器 (UART1DATA)

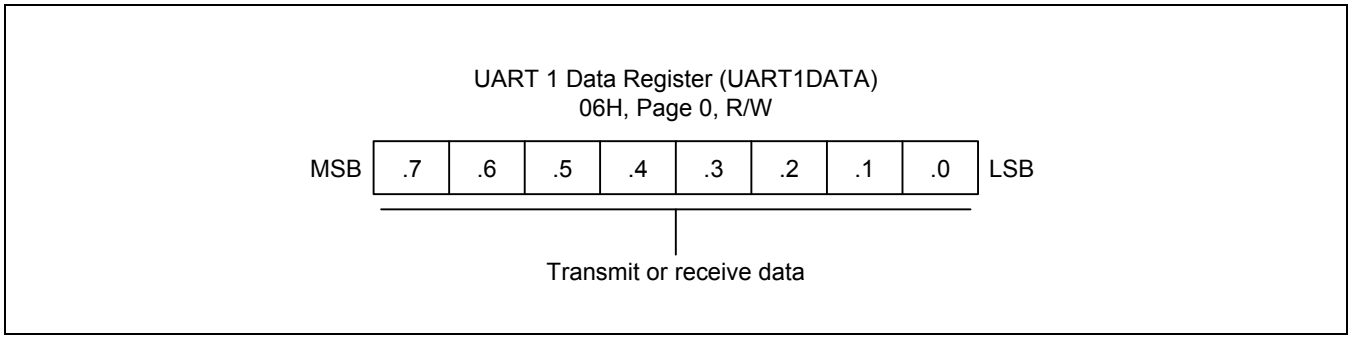


图 18-3 UART 1 数据寄存器 (UART1DATA)

18.1.6 UART 1 波特率数据寄存器 (BR1DATA)

通过设置 UART 1波特率数据寄存器 BR1DATA，可以设定 UART 1的时钟速率 (即波特率)。

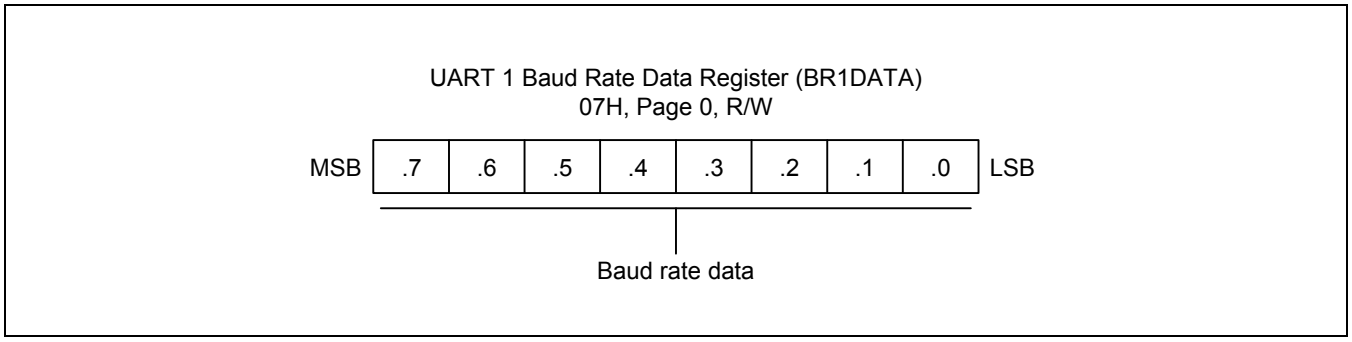


图 18-4 UART 1 波特率数据寄存器 (BR1DATA)

18.1.7 波特率计算

18.1.7.1 Mode 0 模式下波特率的计算

在 mode 0模式下，波特率由波特率数据寄存器，BR1DATA (03H，Page 0) 的值决定，mode 0模式下波特率为：$f_U/(16\times(BR1DATA+1))$

18.1.7.2 Mode 2 模式下波特率的计算

Mode 2模式下的波特率固定为 $f_U/16$，即 f_U 信号16分频。

18.1.7.3 Mode 1 和Mode 3 模式下波特率的计算

在 mode 1和 mode 3模式下，波特率由波特率数据寄存器，BR1DATA (03H，Page 0) 的值决定，mode 1和mode 3模式下波特率为：$f_U/(16\times(BR1DATA+1))$

表 18-1 8位 BR1DATA 可以产生的常用波特率表

模式	波特率	UART 时钟 (fU)	BR1DATA	
			十进制	十六进制
Mode 2	0.5MHz	8MHz	x	x
Mode 0 Mode 1 Mode 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

18.2 模块框图

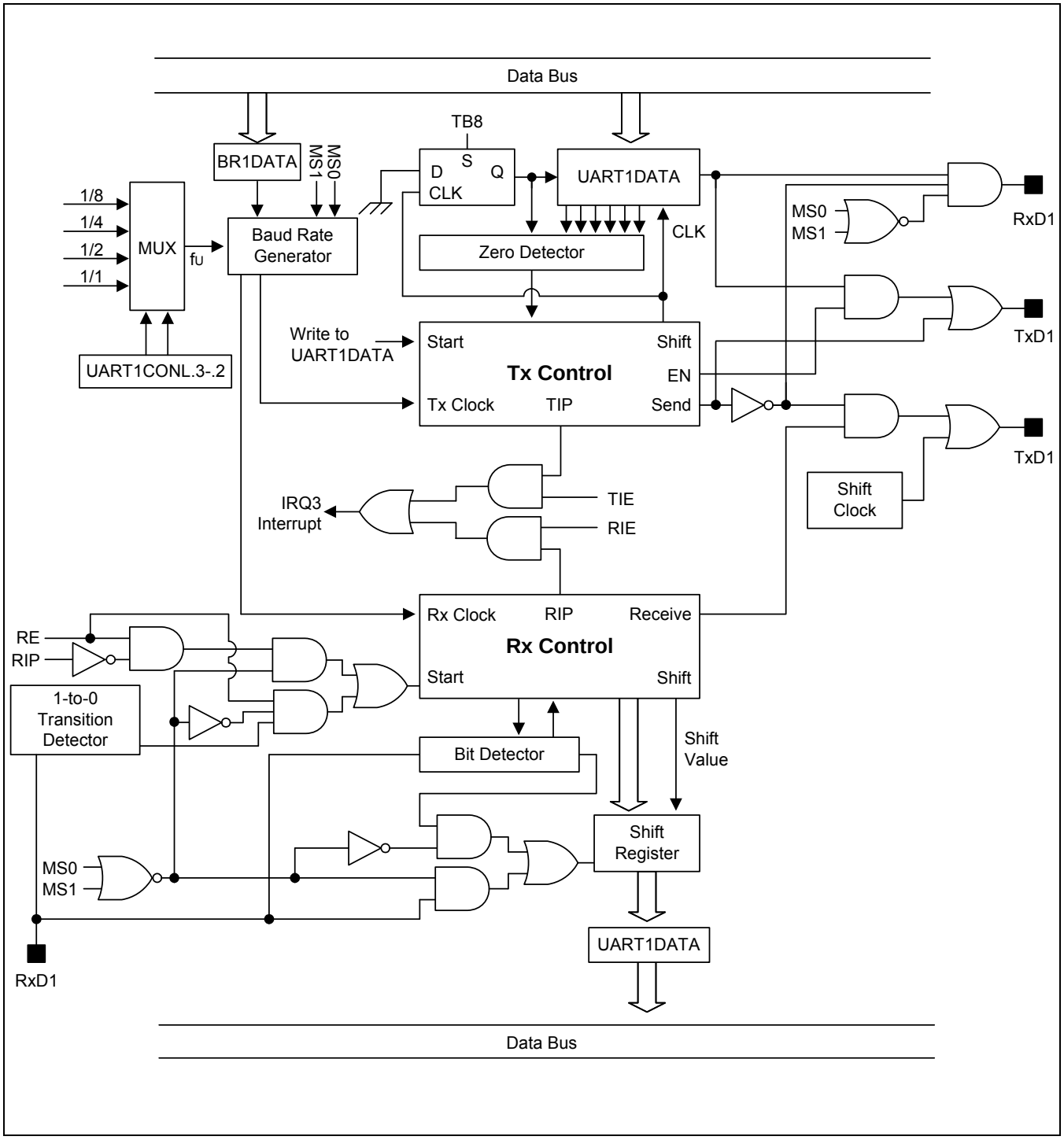


图 18-5 UART 1 功能模块框图

18.2.1 UART 1 MODE 0 模式功能描述

在 mode 0模式下，UART 1通过 RxD1 (P1.4) 接收和发送数据，TxD1 (P1.3) 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低位 (LSB)。

18.2.1.1 Mode 0 模式数据发送流程：

1. 设置 UART1CONL.3 和.2，选择 UART 1 时钟源。
2. 设置 UART 1 禁止或使能发送时奇偶校验自动产生功能 (UART1CONL.7)。
3. 设置 UART1CONH.7-.6为“00B”，选择 mode 0。
4. 将需要发送的数据写入 UART1DATA (06H，page0)，数据开始发送。

18.2.1.2 Mode 0 模式数据接收流程：

1. 设置 UART1CONL.3 和.2，选择 UART 1时钟源。
2. 设置 UART 1禁止或使能发送时奇偶校验自动产生功能 (UART1CONL.7)。
3. 设置 UART1CONH.7-.6为“00B”，选择 mode 0。
4. 写“0”到 UART1CONH.0，清除接收中断标志位。
5. 设置 UART 1 接收使能位 (UART1CONH.4) 为“1”，使能 UART 1接收。
6. 时钟移位信号开始输出到 TxD1 (P1.3)，同时开始从 RxD1 (P1.4) 读取数据。
   当 UART1CONH.1置为“1”时，产生 UART 1接收中断。

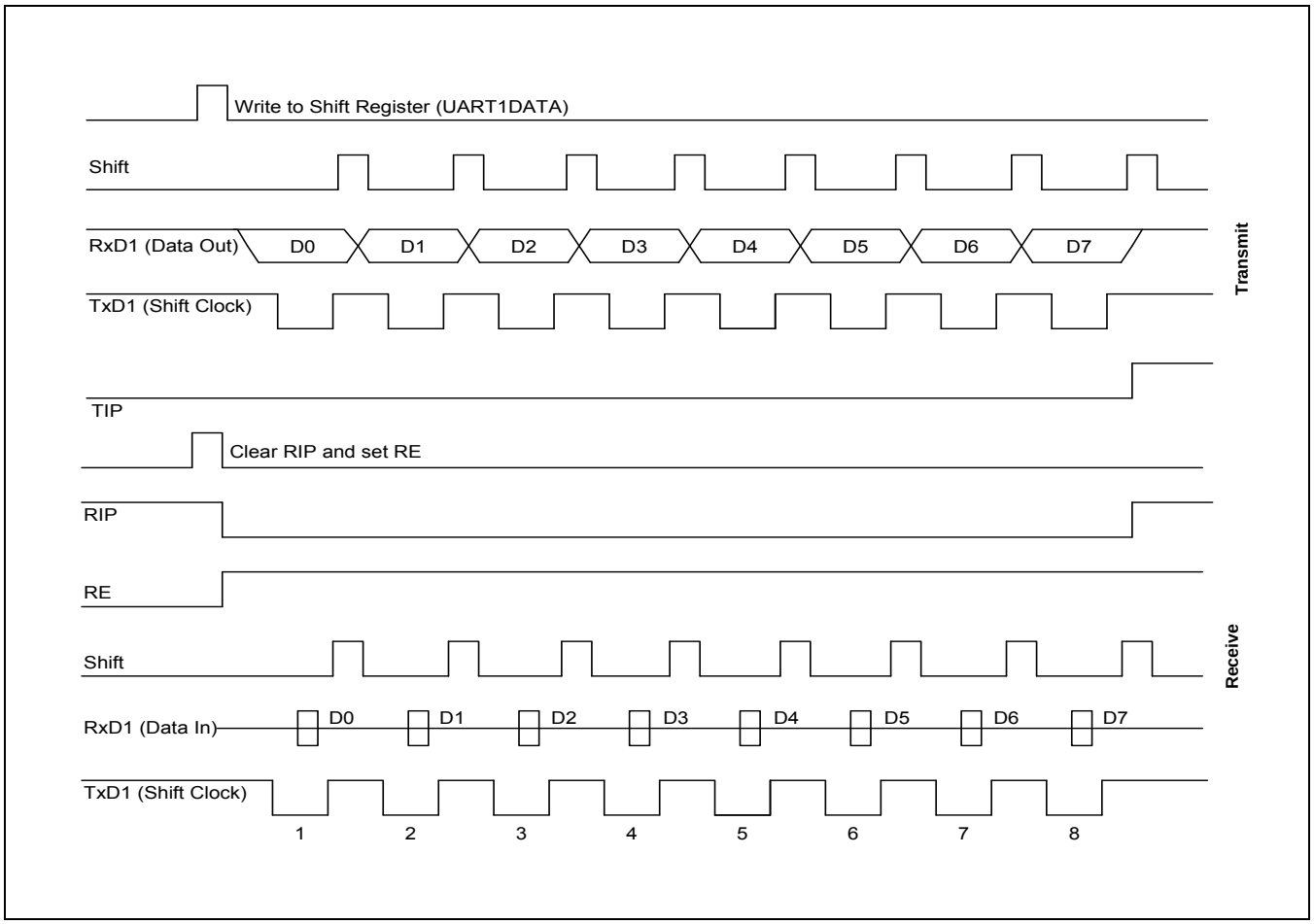


图 18-6 UART 1 Mode 0 模式运行时序图

18.2.2 UART 1 MODE 1 功能描述

在 mode 1模式下，发送 (通过 TxD1 (P1.3)) 或者接收 (通过 RxD1 (P1.4)) 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 (低位在前)
- 停止位（“1”）

Mode 1模式的波特率是可变的。

18.2.2.1 Mode 1 模式数据发送流程

1. 设置 UART1CONL.3 和.2，选择 UART 1时钟源。
2. 设置 UART 1禁止或使能发送时奇偶校验自动产生功能 (UART1CONL.7)。
3. 通过设置8位 BR1DATA 寄存器设定波特率。
4. 设置 UART1CONH.7-.6为“01B”，选择 mode 1(8位 UART)。
5. 将需要发送的数据写入 UART1DATA (06H，page0)，数据开始发送。起始位和停止位由硬件自动产生。

18.2.2.2 Mode 1 模式数据接收流程

1. 设置 UART1CONL.3 和.2，选择 UART 1 时钟源。
2. 设置 UART 1禁止或使能发送时奇偶校验自动产生功能 (UART1CONL.7)。
3. 通过设置8位 BR1DATA 寄存器设定波特率。
4. 选择 mode 1 模式并设置 UART1CONH 的 RE (接收使能) 位为“1”。
5. 当 RxD1 (P1.4) 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。

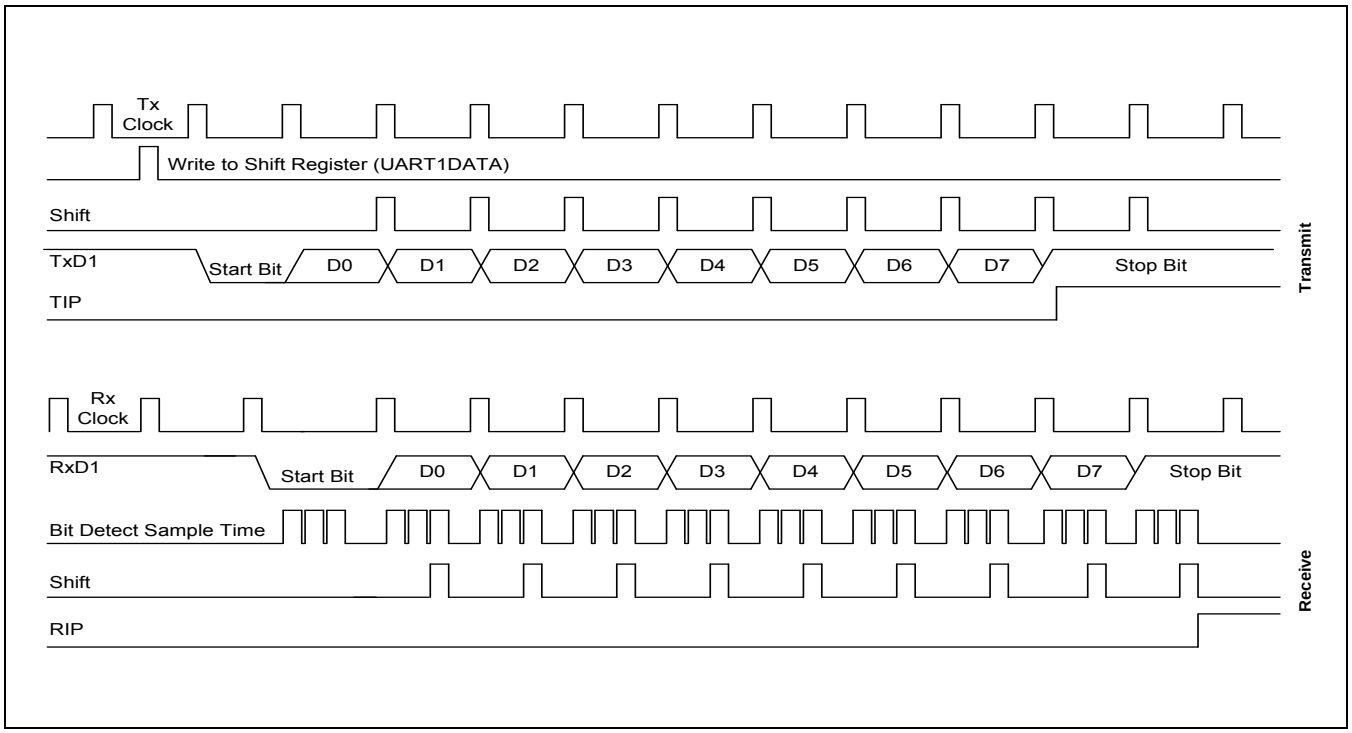


图 18-7 UART 1 Mode 1 模式运行时序图

18.2.3 UART 1 MODE 2 功能描述

在 mode 2 模式下，发送 (通过 TxD1 (P1.3)) 或者接收 (通过 RxD1 (P1.4)) 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 (低位在前)
- 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”的写入到 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。Mode 2模式的波特率为 $f_U/16$。

18.2.3.1 Mode 2 模式发送数据流程

1. 设置 UART1CONL.3 和.2，选择 UART 1时钟源。
2. 设置 UART 1禁止或使能发送时奇偶校验自动产生功能 (UART1CONL.7)。
3. 设置 UART1CONH.7-.6为“10B”，选择 mode 2 (9位 UART)，同时将需要发送的第9位数据写入 TB8。
4. 将需要发送的数据写入 UART1DATA (06H，page0)，数据开始发送。

18.2.3.2 Mode 2 模式接收数据流程

1. 设置 UART1CONL.3 和.2，选择 UART 1时钟源。
2. 设置 UART 1禁止或使能发送时奇偶校验自动产生功能 (UART1CONL.7)。
3. 选择 mode 2模式并设置 UART1CONH 的 RE (接收使能) 位为“1”。
4. 当 RxD1 (P1.4) 管脚上的电平变为低时，开始接收数据。

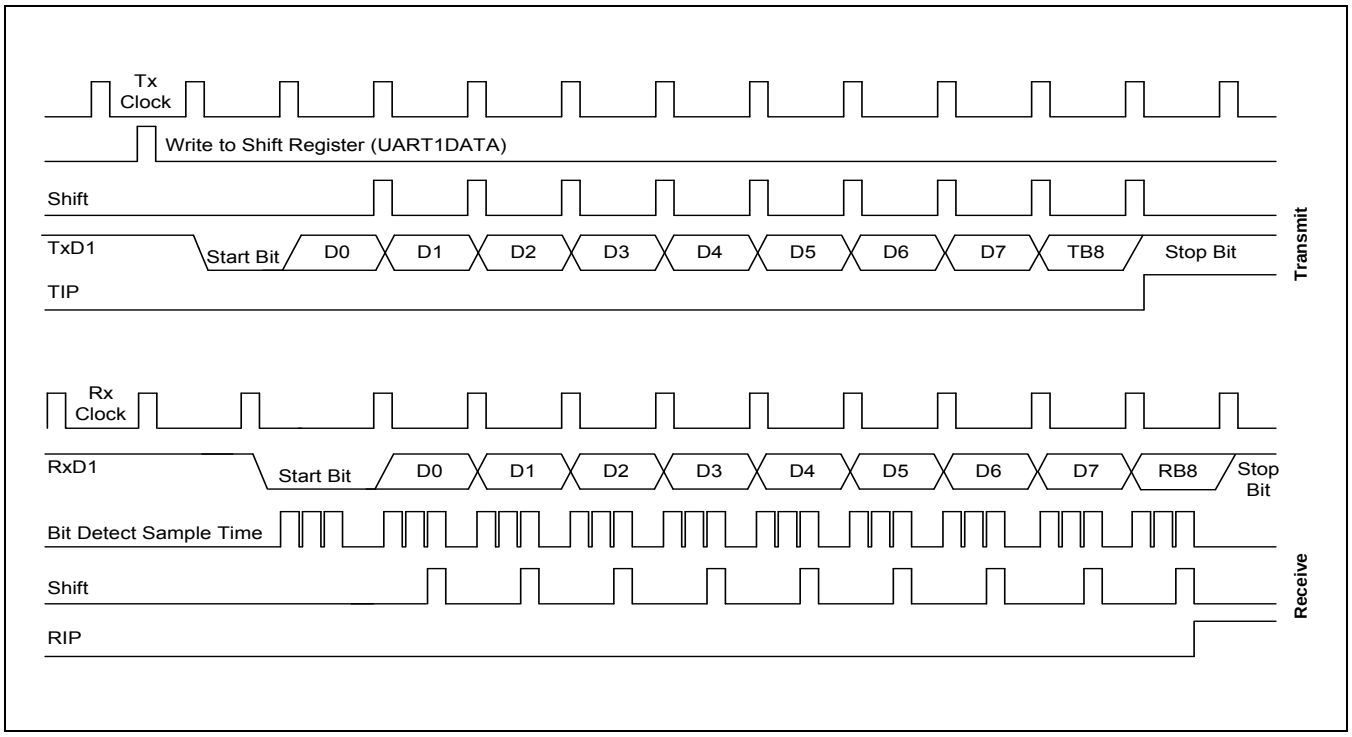


图 18-8 UART 1 Mode 2 模式运行时序图

18.2.4 UART 1 MODE 3 模式功能描述

在 mode 3模式下，发送 (通过 TxD1 (P1.3)) 或者接收 (通过RxD1 (P1.4)) 的数据帧总长度是11位。除了 mode 3模式的波特率可以调节之外，其余的功能 mode 3和 mode 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 (低位在前)
- 可编程设置的第9位数据位
- 停止位 (“1”)

18.2.4.1 Mode 3 模式发送数据流程

1. 设置 UART1CONL.3 和.2，选择 UART 1时钟源。
2. 设置 UART 1禁止或使能发送时奇偶校验自动产生功能 (UART1CONL.7)。
3. 设置 UART1CONH.7-.6为“11B”，选择 mode 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 UART1DATA (06H，page0)，数据开始发送。

18.2.4.2 Mode 2 模式接收数据流程

1. 设置 UART1CONL.3 和.2，选择 UART 1时钟源。
2. 设置 UART 1禁止或使能发送时奇偶校验自动产生功能 (UART1CONL.7)。
4. 选择 mode 3模式并设置 UART1CONH 的 RE (接收使能) 位为“1”。
5. 当 RxD1 (P1.4) 管脚上的电平变为低时，开始接收数据。

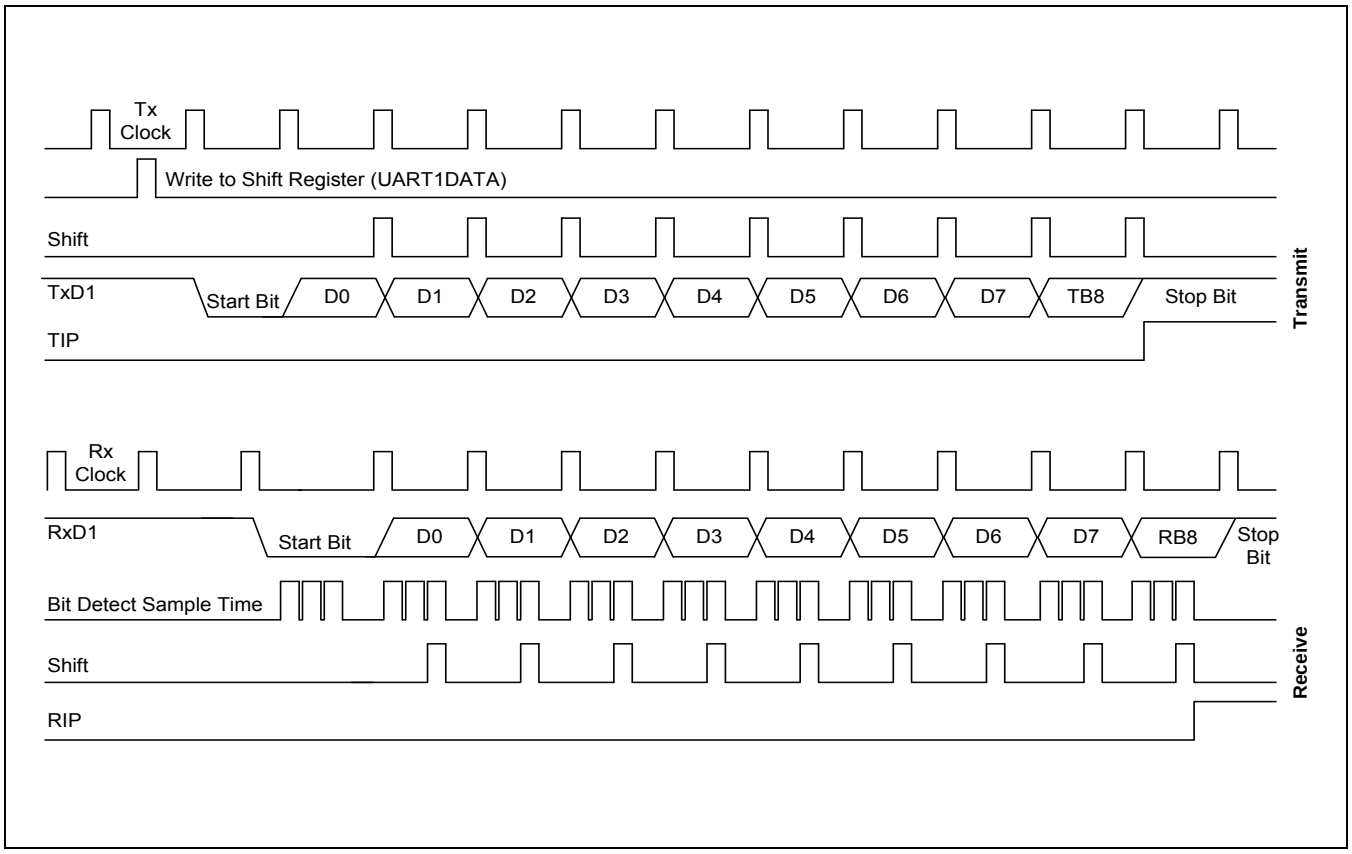


图 18-9 UART 1 Mode 3 模式运行时序图

18.2.5 多节点串行通讯

S3F8 系列的多节点通讯功能的特点是：在一个由多个 S3F82HB 组成的串行通信系统中，允许一个 S3F82HB 作为主机向另一个特定的从机设备发送串行数据，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 mode 2和 mode 3模式下，禁止奇偶校验时才可以实现。在 mode 2 和 mode 3 模式下，收到的数据供9位，第9位写入了 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。

18.2.5.1 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 mode 0模式下没有用处。在 mode 1模式下，它可以用来检测停止位的有效性。在 mode 1模式下接收数据时，如果设置MCE 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。

18.2.6 多节点通讯的建立流程

多节点通讯的建立流程如下：

1. 设置所有的串行总线的 S3F82HB 设备的 UART 1模式为 mode 2或者mode 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”)
   - 下一个字节：数据
     (第9位 =“0”)
4. 由于第9位为“1”，所有的从机都会产生中断来接收第一个字节。当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 位清为“0”来准备接受数据。其他的从机继续正常运行。

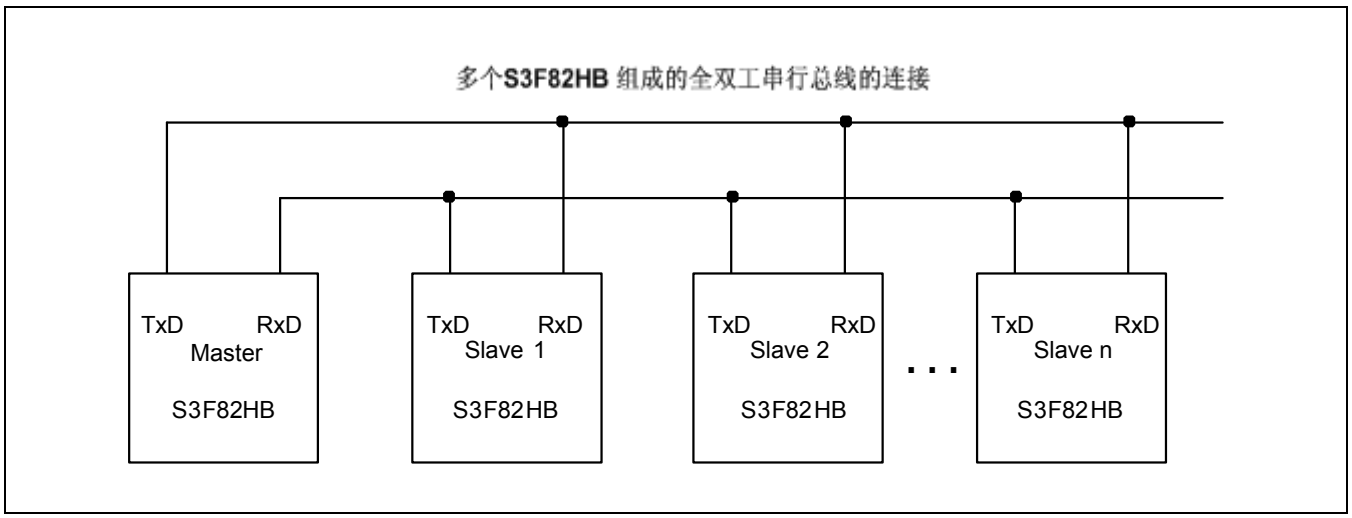


图 18-10 多节点串行通讯连接示例

19 嵌入式闪存接口

19.1 概述

S3F82HB 内部有一个片上闪存，可代替掩膜 ROM。这个闪存支持“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编写数据。
S3F82HB 中嵌入的64K 字节闪存有两种工作模式：

- 工具编程模式
- 用户编程模式：参考第25章，S3F82HB FLASH MCU

19.1.1 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)。
读保护功能只能在工具模式下实现。所以为了芯片在编程后处于读保护状态，需要在编程工具提供的操作界面上勾选读保护。

S3F82HB 内部集成了自升压电路可以产生高电压。 因此，使用芯片内部逻辑电平就可以完成对闪存的操作。
用户编程模式下的片上闪存编程是通过几个专用控制寄存器来实现的，可是实现4种功能 – 编程，读，扇区擦除和hard lock (硬锁)。

注释:

1. 当 CPU 工作在副时钟下时，用户编程模式不可用。
2. 在进入用户编程模式的前，请确保用 DI 指令禁止全局中断。因为用户模式下会一直检查中断请求寄存器 (IRQ) 的状态。如果一旦发生中断请求，闪存会立即停止用户模式的任何操作。
3. 即使 DI 以后，中断请求寄存器的状态仍然可能因为中断请求的产生而改变。为了避免这种情况的发生，务必禁止每个中断源的中断使能位。

19.2 闪存控制寄存器 (用户编程模式)

19.2.1 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，字节编程和设置硬件锁。

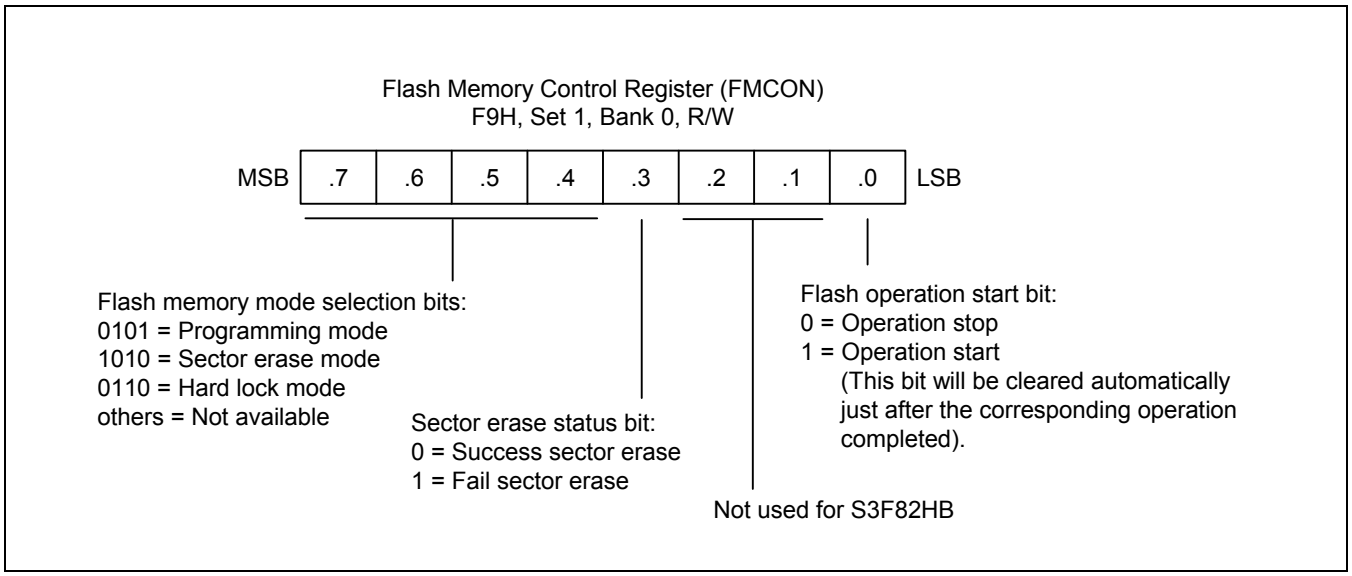


图 19-1 闪存控制寄存器 (FMCON)

FMCON.0是擦除和 Hard Lock 保护操作的开始位。因此，设置 FMCON.0为“1”，则开始擦除或 Hard Lock 保护操作。开始擦除或 Hard Lock 操作后，必须等待一段时间后才能进行同一扇区的读写操作。
闪存的读写则不需要用到这一控制位。

扇区擦除标志位 (FMCON.0) 是只读的。即使 IMR = 0，只要具体中断源的中断使能位处于使能的状态，
在扇区擦除的过程中 CPU 仍然可能响应中断。此时，扇区擦除操作被强行终止，CPU 进行中断处理。
所以一定要在擦除操作之后判断扇区擦除标志位，从而确定擦除是否成功。如果该位为逻辑“1”则说明擦除失败，反之，擦除成功。

注释: 当向 FMUSR 寄存器中写入“A5H”后，将会立即根据 FMCON 的设置进行扇区擦除，编程或者 Hard Lock 操作。但这个操作可能不是希望发生的。因此，为了不发生误操作，当执行扇区擦除或 Hard Lock 操作时用户应注意 FMUSR 的设置顺序。

19.2.2 用户编程使能寄存器 (FMUSR)

FMUSR 寄存器是为闪存的安全操作设置的。当 CPU 由于干扰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。

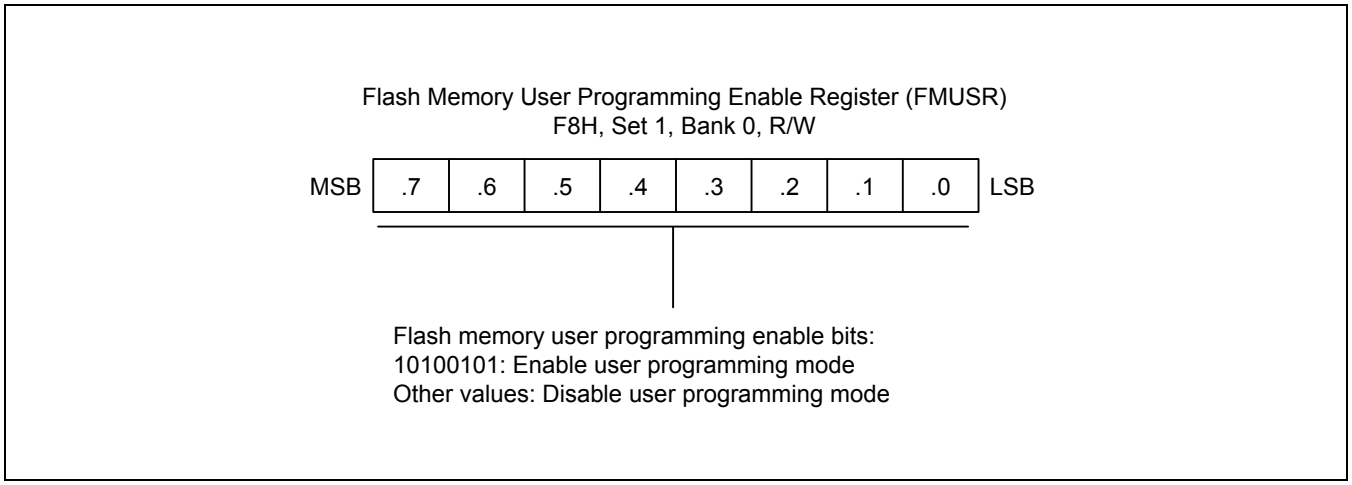


图 19-2 闪存用户编程使能寄存器 (FMUSR)

19.2.3 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8位地址，FMSECH (高字节闪存扇区地址寄存器) 存储高8位地址。

因为 S3F82HB 有512个扇区，所以同时需要 FMSECL 和 FMSECH 来进行扇区定位。一个扇区由128个字节组成。每个扇区的起始地址是 XX00H 或 XX80H，也就是说每个扇区的基址不是 XX00H 就是 XX80H。所以 FMSECL 的低7位数据没有意义。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。

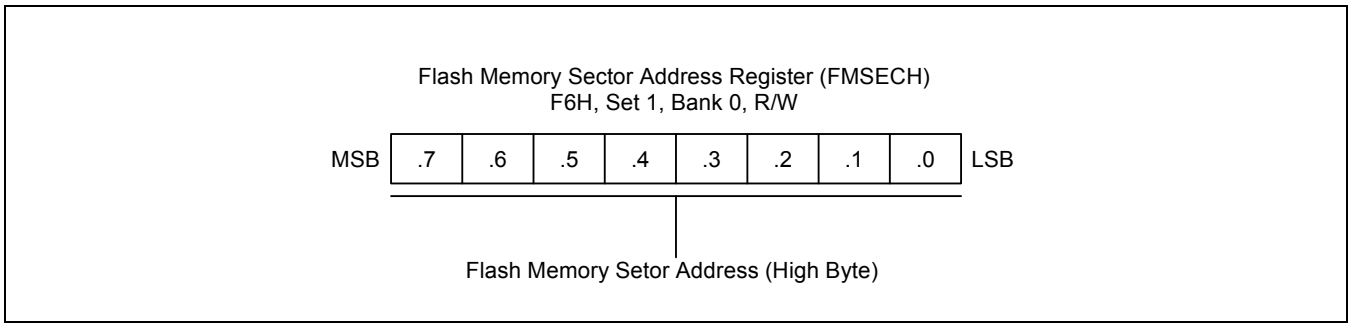


图 19-3 闪存扇区地址寄存器 (高字节)

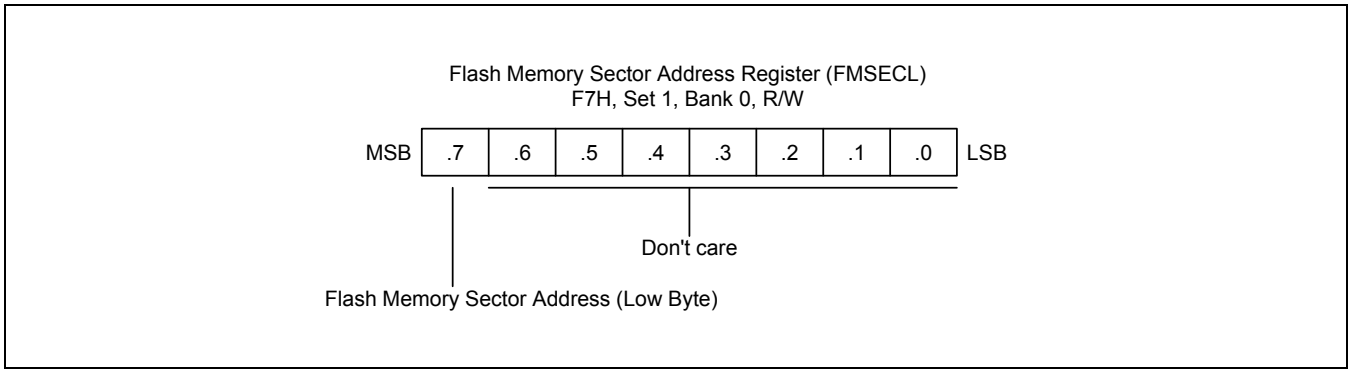


图 19-4 闪存扇区地址寄存器 (低字节)

19.3 ISP™ (在板编程) 扇区

ISP™ 扇区可以用于存放在板编程代码 (通过 I/O 接口实现软件升级的 Boot 代码)。
出于安全考虑，“LDC”指令擦除或编程操作不会影响 ISP™ 扇区的内容，但是这要通过使能 ISP 保护来实现。

用户可以通过设置 Smart Option 中 ISP 使能禁止位为“0”来使能芯片的 ISP 功能，而 ISP™ 扇区的保护功能只有在用户使能 ISP 功能时才有效。如果不想使用 ISP 扇区，只要将 Smart Option 中 ISP 使能禁止位设为“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。

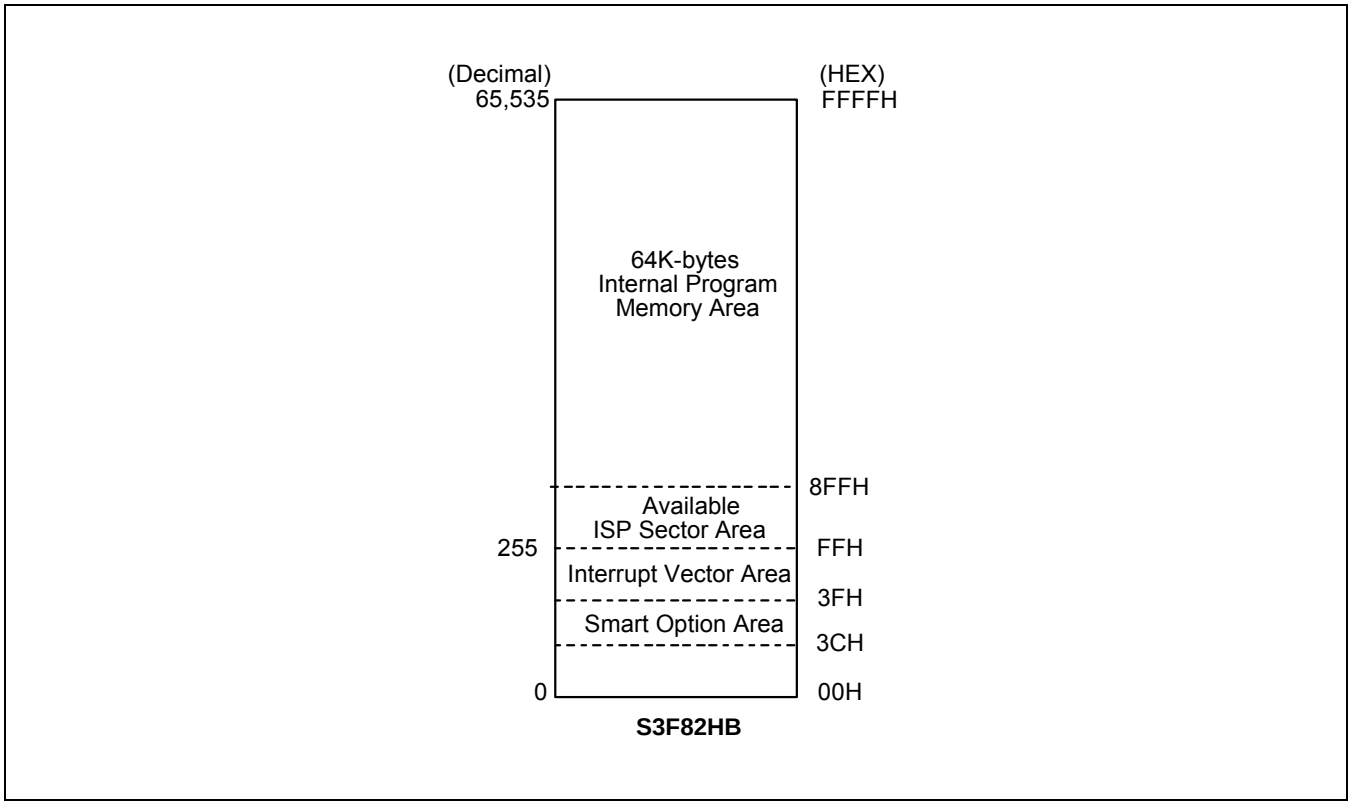


图 19-5 程序存储地址空间

表 19-1 ISP 扇区大小

Smart Option (003EH) ISP 扇区大小选择位			ISP 扇区	ISP 扇区大小
位2	位1	位0		
1	x	x	–	0
0	0	0	100H – 1FFH (256 Byte)	256 Bytes
0	0	1	100H – 2FFH (512 Byte)	512 Bytes
0	1	0	100H – 4FFH (1024 Byte)	1024 Bytes
0	1	1	100H – 8FFH (2048 Byte)	2048 Bytes

注释: 在用户模式下，一旦使能 ISP 的保护，Smart Option (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。

19.3.1 ISP 复位向量和 ISP 扇区大小的关系

如果用户使能了 ISP 保护功能 (3EH.2 = 0) 并且选择了 OBP 复位向量 (3EH.7 = 0)，需要通过设置 ISP 复位向量地址选择位 (3E.7-.5) 来选择恰当的 CPU 复位地址 (*表 19-2*)。

表 19-2 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
位7	位6	位5			
1	x	x	0100H	–	–
0	0	0	0200H	100H – 1FFH	256 Bytes
0	0	1	0300H	100H – 2FFH	512 Bytes
0	1	0	0500H	100H – 4FFH	1024 Bytes
0	1	1	0900H	100H – 8FFH	2048 Bytes

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。

19.4 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。

S3F82HB 的64K 字节闪存空间分为512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触发擦除起始位 (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)。

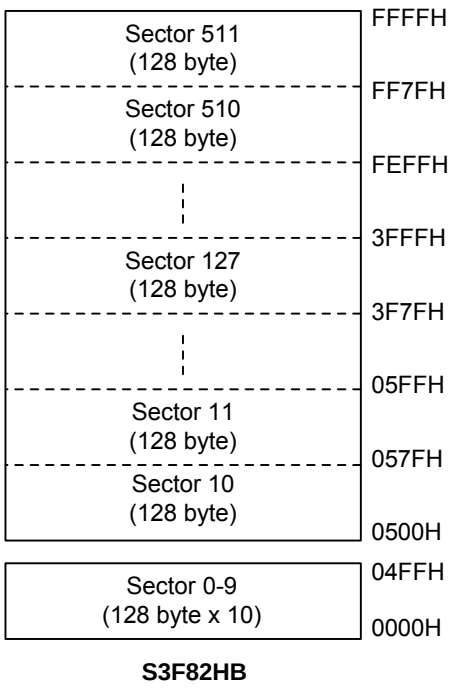


图 19-6 用户编程模式下的扇区

19.4.1 用户编程模式下扇区擦除流程

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 (用户自设值)。
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. 检查“扇区擦除状态位”判断此次擦除操作是否成功。

编程实例 19-1 扇区擦除

```
            •
            •

            SB0
reErase:    LD     FMUSR,Temp0              ; 使能用户编程模式
                                             ; Temp0 = #0A5H
                                             ; Temp0 的值必须在其他代码段中设置
            LD     FMSECH,#10H
            LD     FMSECL,#00H              ; 设置扇区地址 (1000H–107FH)
            CP     UserID_Code,#User_value  ; 检查用户ID (用户自设置)
                                             ; User_value 可以是用户自设的任意值
            JR     NE,Not_ID_Code           ; 如果不相等，则跳转到 Not_ID_Code
            LD     FMCON,Temp1              ; 开始扇区擦除操作
                                             ; Temp1 = #0A1H
                                             ; Temp1 的值必须在其他代码段中设置
            NOP                             ; 必需的空指令
            NOP                             ; 必需的空指令
            LD     FMUSR,#0                 ; 关闭用户编程模式
            TM     FMCON,#00001000B         ; 检查 “扇区擦除状态位”
            JR     NZ,reErase               ; 如果擦除失败则跳转到 reErase
            •
            •
            •
            •

Not_ID_Code:
            SB0
            LD     FMUSR,#0                 ; 关闭用户编程模式
            LD     FMCON,#0                 ; 关闭扇区擦除模式
            •
            •
```

注释: 为使用上述代码段，必须在其他代码段中事前设置好 Temp0~Temp(n) 的值

19.5 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。为了闪存内容的安全性，必须设置 FMSECH 和FMSECL 为闪存中欲操作区间的扇区地址。

由“LDC”指令开始编程写操作，在改变 FMSECH 和 FMSECL 的之前最多可以连续写入128 字节，因为一个扇区的大小就是128字节。

19.5.1 用户编程模式下的编程流程

1. 编程前必须保证编程地址中的值为“FFH”(可以通过扇区擦除来实现)。
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一个工作寄存器中。
7. 检查用户 ID (用户自设)。
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 19-2 编程

```
            •
            •

            SB0
            LD          FMUSR,Temp0             ; 使能用户编程模式
                                                ; Temp0 = #0A5H
                                                ; T emp0 的值必须在其他代码段中设置
            LD          FMSECH,#17H
            LD          FMSECL,#80H             ; 设置扇区地址 (1780H-17FFH)
            LD          R2,#17H                 ; 设置位于同一扇区的目标操作地址

            LD          R3,#84H
            LD          R4,#78H                 ; 临时数据
            CP          UserID_Code,#User_value; 检查用户 ID (用户自设)
                                                ; User_value可以是用户设置的任意值
            JR          NE,Not_ID_Code          ; 如果不相等，则跳转 Not_ID_Code
            LD          FMCON,Temp1             ; 开始编程
                                                ; Temp1 = #51H
                                                ; Temp1的值必须在其他代码段中设置
            LDC   @RR2,R4                       ; 往1784H单元写数据
            NOP                                 ; 必需的空指令
            LD          FMUSR,#0                ; 关闭用户编程模式

            •
            •
            •
            •
Not_ID_Code:
            SB0
            LD          FMUSR,#0                ; 关闭用户编程模式
            LD          FMCON,#0                ; 关闭编程模式

            •
            •
            •
            •
```

注释: 为使用上述代码段，必须在其他代码段中事前设置好 Temp0~Temp(n) 的值。

19.6 读

由“LDC”指令开始读操作。

19.6.1 用户编程模式下编程流程

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 19-3 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           ; 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
                                    ; (在300H 和3FFH 之间)
            INC     R3
            CP      R3,#0H
            JP      NZ,LOOP

            •
            •
            •
            •
```

19.7 HARD LOCK 保护

用户可通过写“0110B”到 FMCON7-4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，执行片擦除操作来接触这种保护。

在用户编程模式下，设置 Hard Lock 保护的流程如下。工具模式下，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。

19.7.1 用户编程模式下的编程流程

1. 设置闪存用户编程使能寄存器 (FMUSR) 为“10100101B”。
2. 检查设置闪存控制寄存器 (FMCON) 为“01100001B”。
3. 检查用户 ID (用户自设)。
4. 设置闪存用户编程使能寄存器 (FMUSR) 为“00000000B”。

编程实例 19-4 Hard Lock 保护

```
            •
            •

            SB0
            LD     FMUSR,Temp0              ; 使能用户编程模式
                                            ; Temp0 = #0A5H
                                            ; Temp0 必须在其他代码段中赋值
            CP     UserID_Code,#User_value  ; 检查用户ID (用户自设)
                                            ; User_value 可以是用户设定的任何值
            JR     NE,Not_ID_Code           ; 如果不相等，则跳转 Not_ID_Code
            LD     FMCON,Temp1              ; 设置并开始Hard Lock
                                            ; Temp1 = #61H
                                            ; Temp1必须在其他代码段中赋值
            NOP                             ; 必需的空指令
            LD     FMUSR,#0                 ; 关闭用户编程模式

            •
            •
            •
            •
Not_ID_Code:
            SB0
            LD     FMUSR,#0                 ; 关闭用户编程模式
            LD     FMCON,#0                 ; 关闭Hard Lock 保护模式
            •
            •
            •
            •
```

注释: 为使用上述代码段，必须在其他代码段中事前设置好 Temp0~Temp(n) 的值。

20 电池电压检测

20.1 概述

S3F82HB 内建一个BLD (Battery Level Detector, 电池电压检测)电路，可用于软件检测电源电压跌落或者外部输入电平。BLD工作可通过软件启动或关闭。因为当BLD工作时，芯片消耗很大的电流，所以建议非必要时保持BLD模块关闭。BLD的阈值电压也可以通软件设置成下列3种电压之一：2.2V，2.4V，2.8V(V_{DD} 参考电压) 或外部输入电平(外部参考电压)。

当$B_{LDCON.3}$置位时，BLD模块工作。若V_{DD} 电压低于BLDCON.2–.0选择的参考电压，则BLDCON.4位将被置位。若V_{DD} 电压更高，则BLDCON.4位将被清“0”。若需减小功耗，关闭BLD模块。

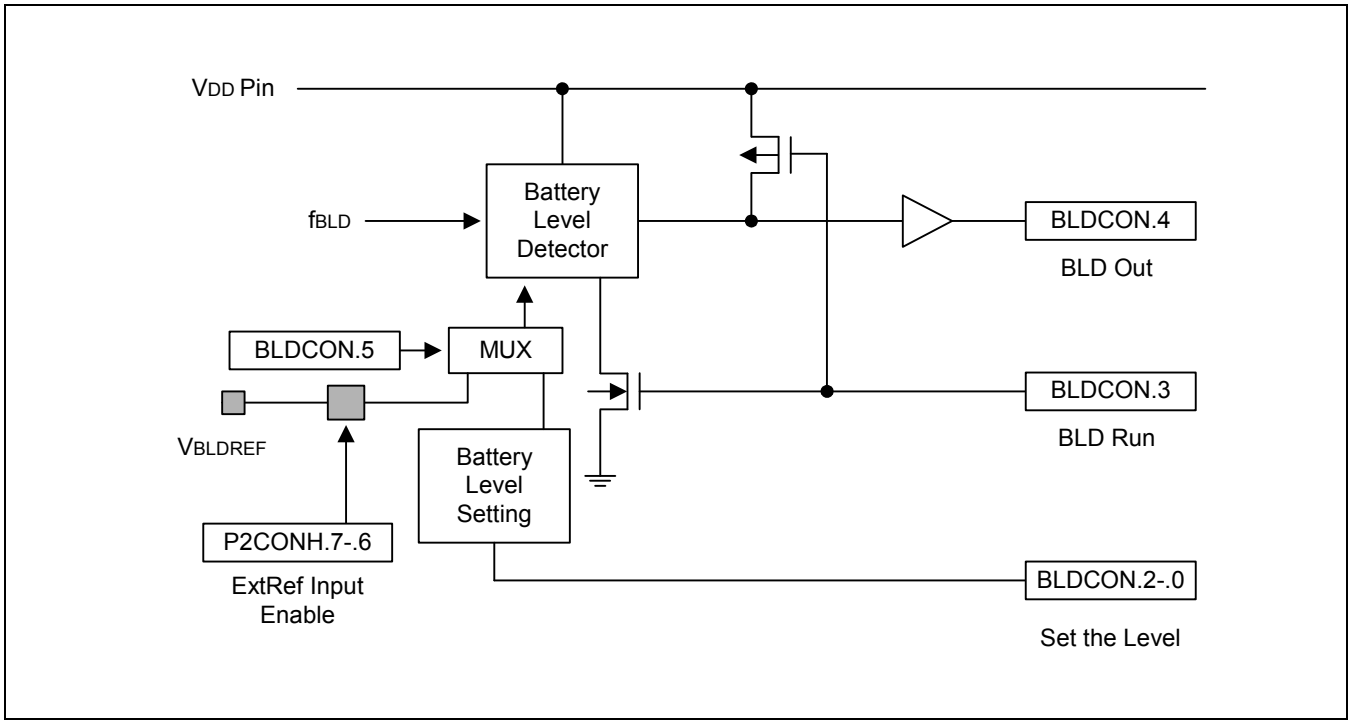
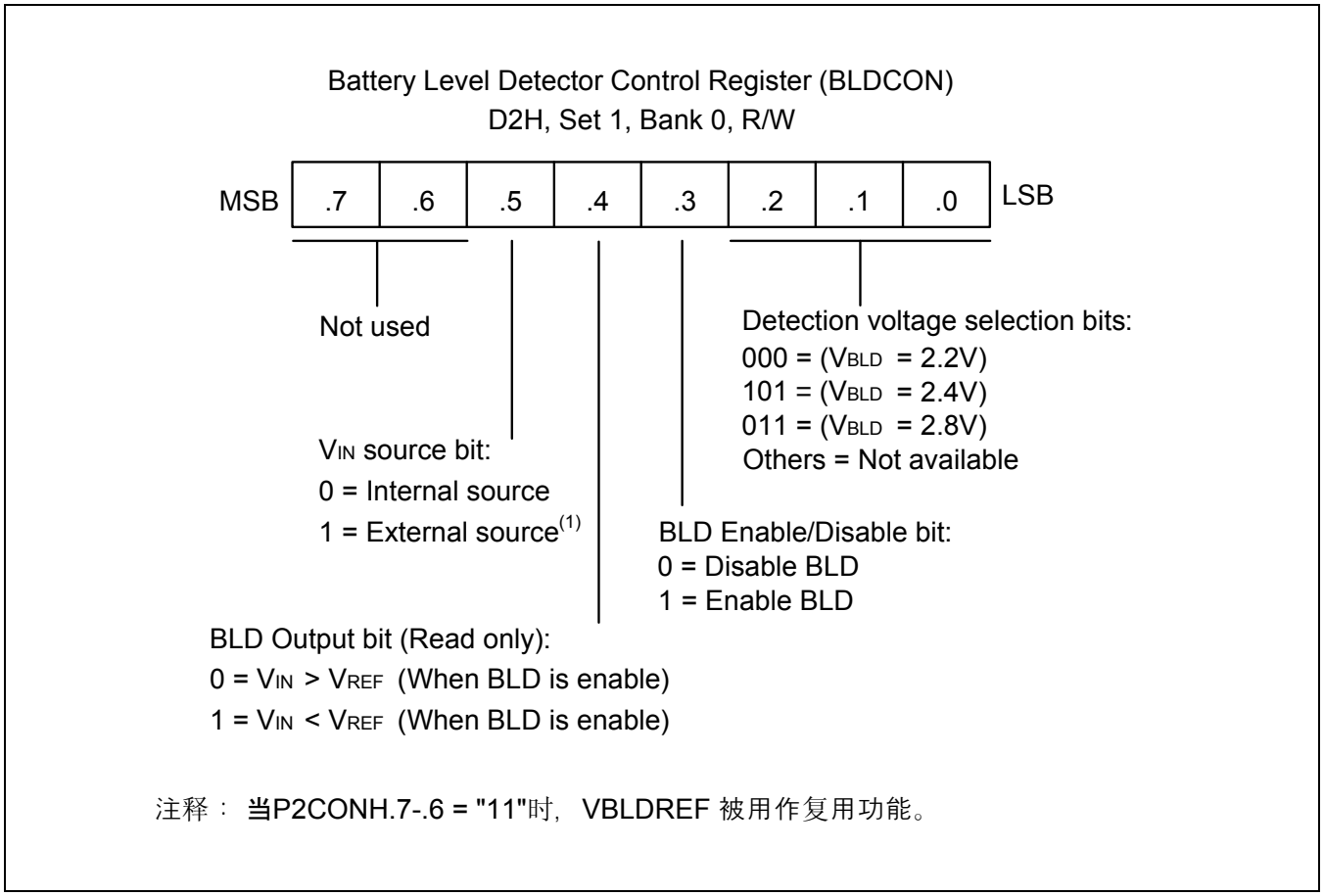


图 20-1 电池电压检测框图

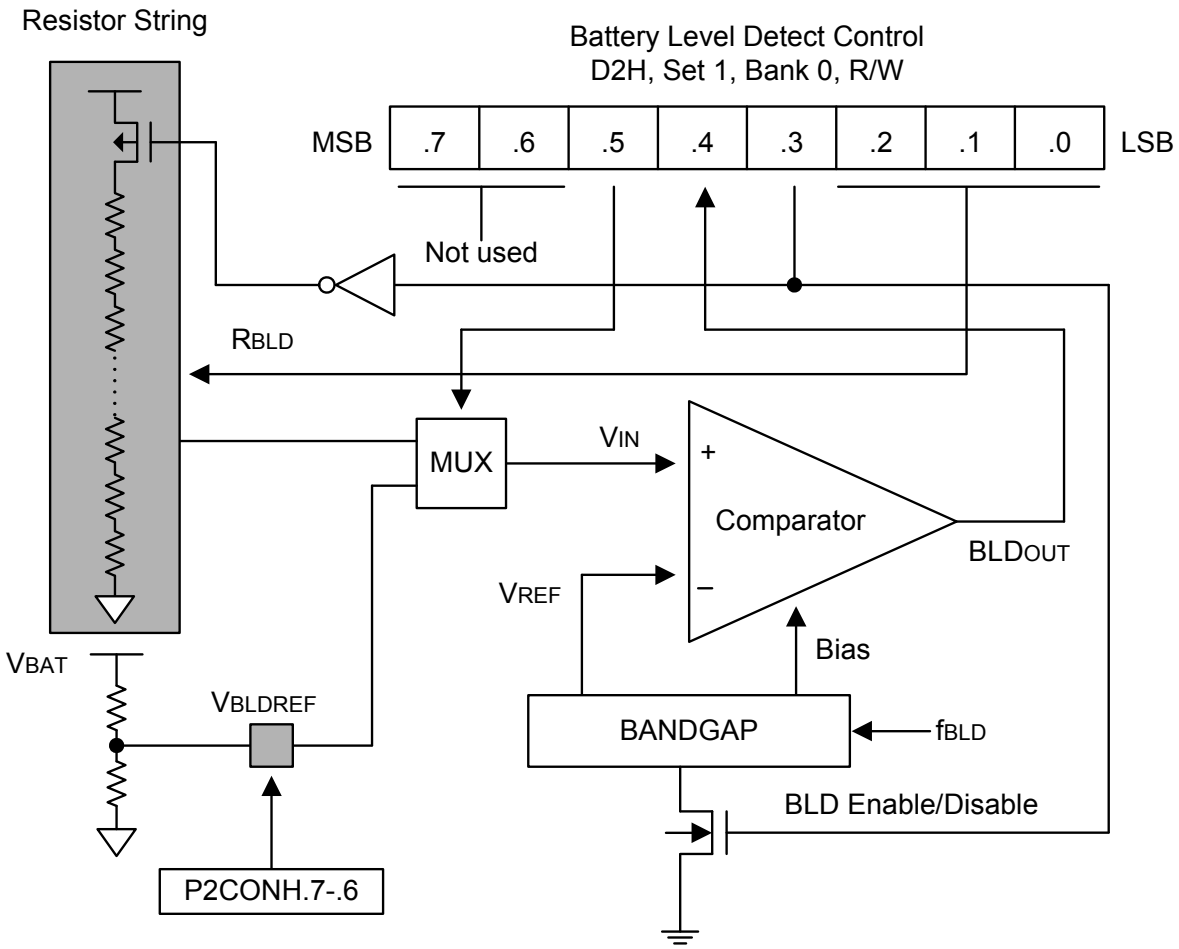
20.1.1 电池电压检测控制寄存器 (BLDCON)

BLDCON的第3位控制电池电压检测模块的开关。基本上，系统复位后V_{BLD}被设为2.2 V，并可通过改变电池电压检测控制寄存器(BLDCON) 选择3个值之一。在BLDCON寄存器中写入3位值之后，
内建的电阻串被选中并固定V_{BLD}的值。*图 20-2* 显示了V_{BLD}的3个值。



注释： 当P2CONH.7-.6 = "11"时, VBLDREF 被用作复用功能。

图 20-2 电池电压检测控制寄存器 (BLDCON)



注释：

1. BLDCON 的复位值为 #00H。
2. VREF 约 1V。

图 20-3 电池电压检测电路和框图

表 20-1 BLDCON 值和检测电平

BLDCON .2–.0	V_{BLD}
0 0 0	2.2V
1 0 1	2.4V
0 1 1	2.8V
其他	无效

21 频率计数器

21.1 概述

16位频率计数器 (frequency counter，简称 FC) 由3个功能模块组成：

- 8位模式寄存器 (FCCON)
- 16位计数器 (FCNTH/FCNTL)
- 门控制逻辑

频率计数器用精确的时间间隔测量输入频率。门打开后，脉冲被输入频率计数器。可通过 FCCON 寄存器设置门控制电路，来控制频率计数时间。FCCON 寄存器可设置6种不同的门打开时间。

门打开时间内，16位频率计数器对从 FCLK，C0OUT 或 C1OUT 管脚输入的频率信号进行计数。16位频率计数器的计数输入是由 FCCON 寄存器设置的。

16位二进制计数器 (FCNTH/FCNTL) 仅可由8位的 RAM 控制指令读取。

通过设置 FCCON 寄存器，门在选定的时间段打开并将门标志位清0。
在门的打开时间内，输入频率由16位计数器计数。当门关闭后，计数操作完成，并产生一个中断。

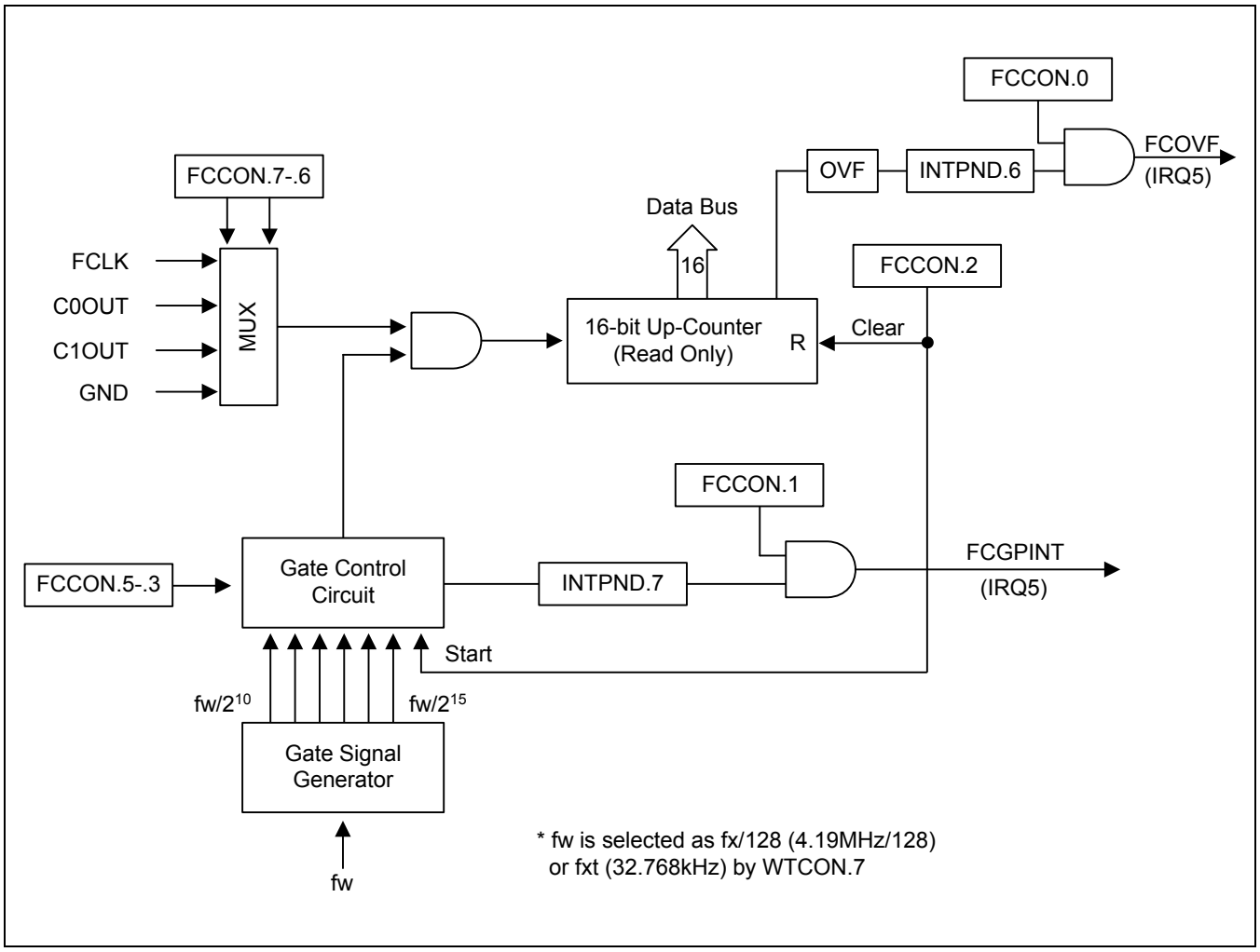


图 21-1 频率计数器框图

21.1.1 频率计数器控制寄存器 (FCCON)

FCCON 是一个8位寄存器。可用来设置门打开时间，并检查频率计数器计数操作是否完成。

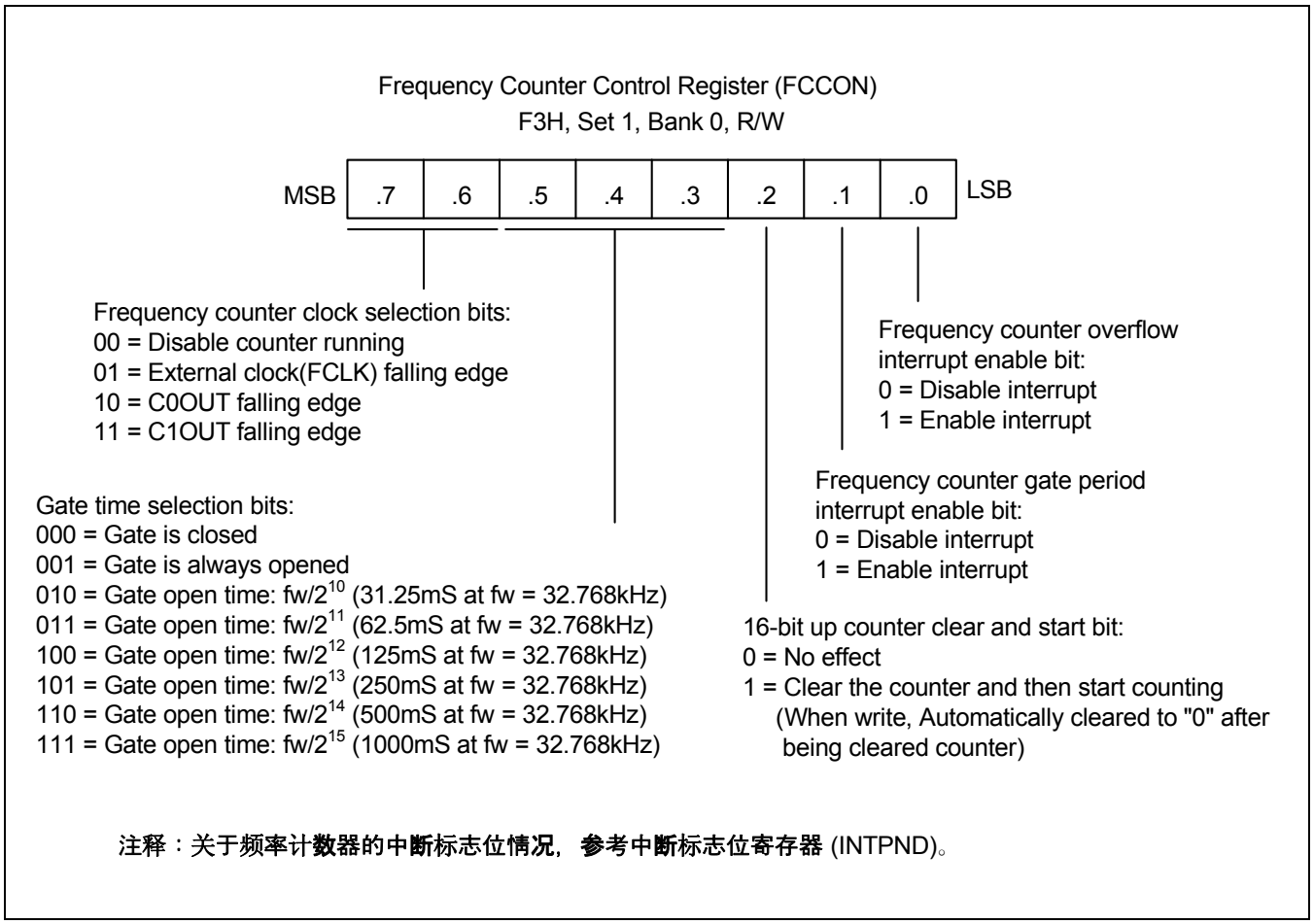


图 21-2 频率计数器控制寄存器 (FCCON)

21.1.2 门打开时间

对 FCCON 写入值后，频率计数器的门就在选定的时间段中打开，开始于一个时钟下降沿。当门打开时，FCLK，C0OUT 或 C1OUT 管脚的频率被16位计数器计数。当门关闭后，生成一个门周期中断并将门周期中断标志位 (INTPND.7) 置1。

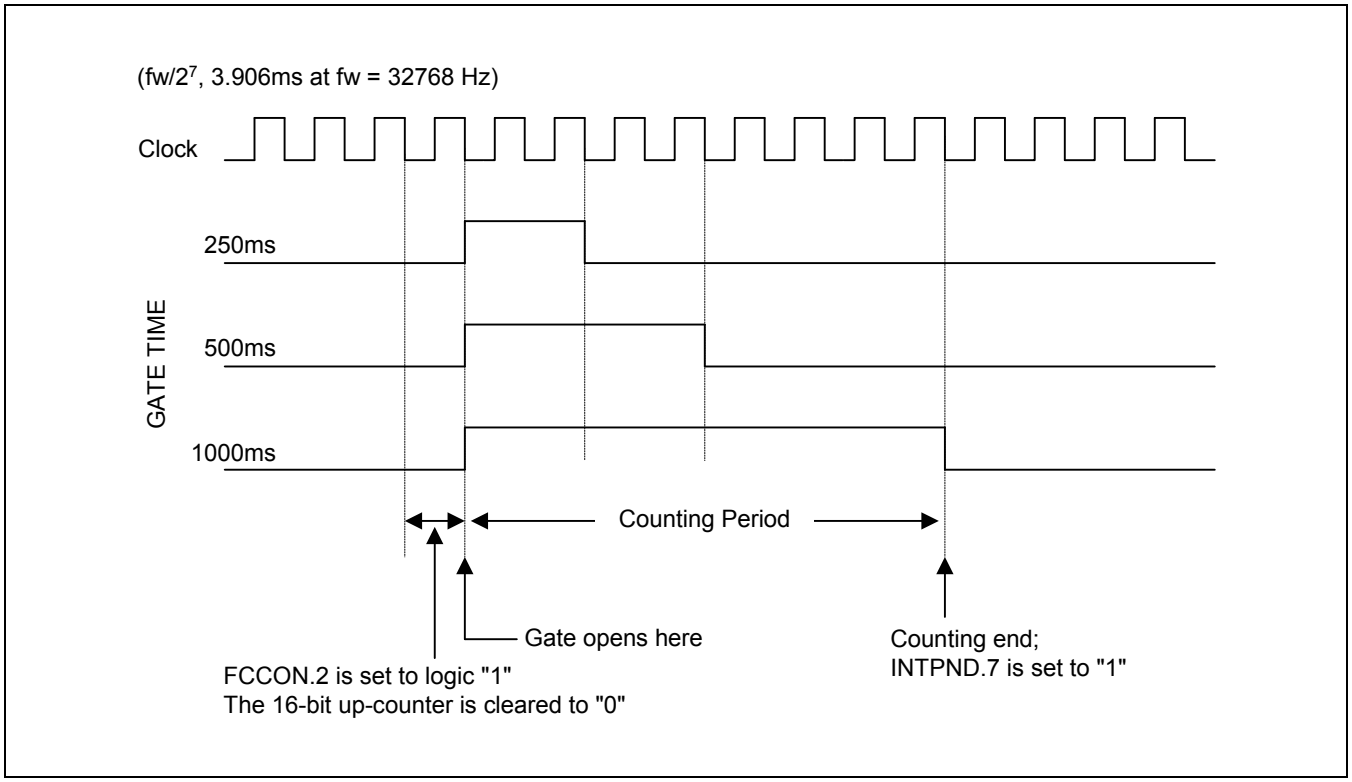


图 21-3 门时序图

22 比较器 0/1

22.1 概述

S3F82HB 内部集成了两个比较器模块，比较器模块的输出会随正向和反向输入信号的改变而改变。
在温控器的应用场合，可以用做 RCF 转换时频率信号的发生器。RCF 转换将电阻值转换为频率值，
用另一种方式实现了 AD 转换。

比较器 0/1只有在使能位 CMP0CON.3/CMP1CON.3 置高的时候才处于工作状态。
如果正向输入端的电平小于反向输入端 CMP0CON.4/CMP1CON.4会清零。
反之，CMP0CON.4/CMP1CON.4会被置高。比较器的输出可以设置为频率计数器 (frequency counter)
模块的时钟源。因为比较器会有功耗，所以在对功耗要求比较的应用场合，不建议使用。

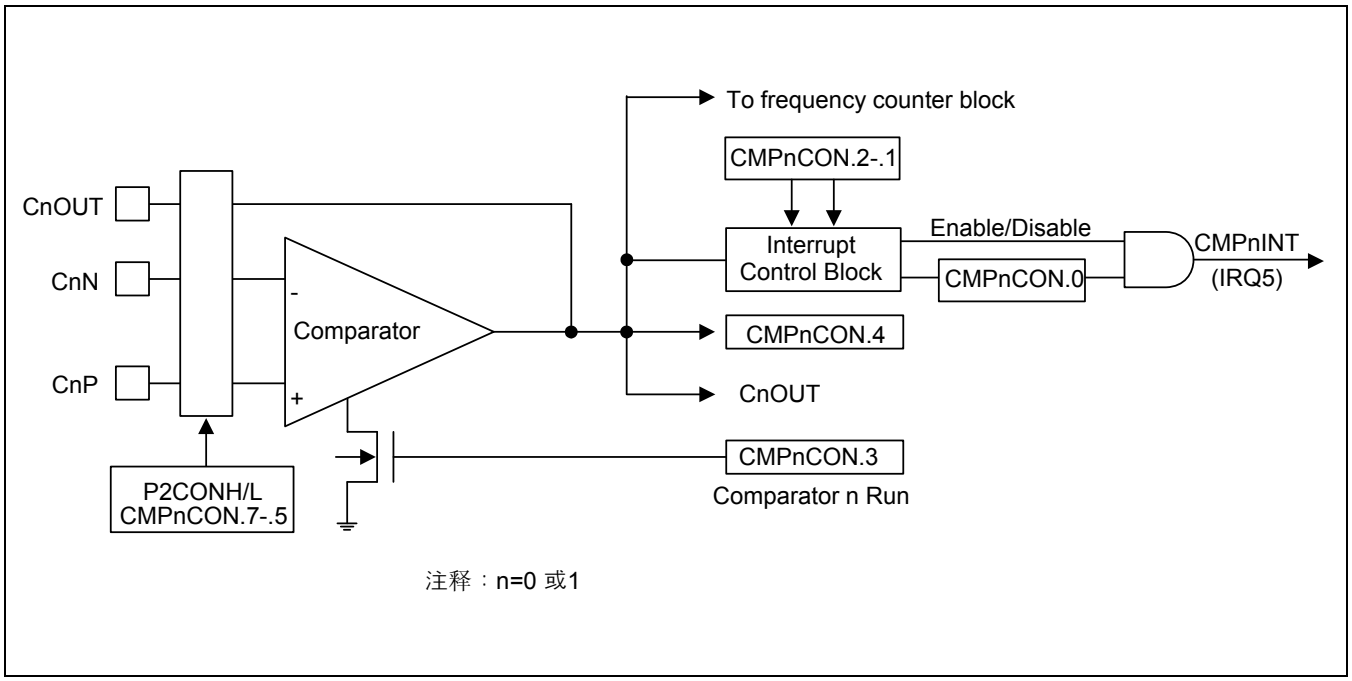


图 22-1 比较器模块框图

22.1.1 比较器 0 控制寄存器 (CMP0CON)

比较器0受控于 CMP0CON (位于 page 0的08H 单元)。可以通过配置 CMP0CON 实现以下功能：

- 选择复用功能 (AD3/AD2/AD1或 C0N/C0P/C0OUT)
- 比较器0工作使能/禁止控制
- 比较器0中断控制

复位后，CMP0CON的值为“00H”。 所以如果要使用比较器0模块，必须先配置寄存器 CMP0CON。

22.1.2 比较器 1 控制寄存器 (CMP1CON)

比较器1受控于 CMP1CON (位于 page 0的09H 单元)。通过配置 CMP1CON 可以实现以下功能：

- 选择复用功能 (AD4/AD5/AD6 和 C1N/C1P/C1OUT)
- 比较器1工作使能/禁止控制
- 比较器1中断控制

复位后，CMP1CON的值为“00H”。 所以如果要使用比较器1模块，必须先配置寄存器 CMP1CON。

22.1.3 比较输出控制寄存器 (COUTCON)

通过配置寄存器 COUTCON 可以实现以下功能：

- 比较器0/1的输出控制

复位后，COUTCON 的值为“00H”。所以如果要使用比较器0/1模块，必须先配置寄存器 COUTCON。

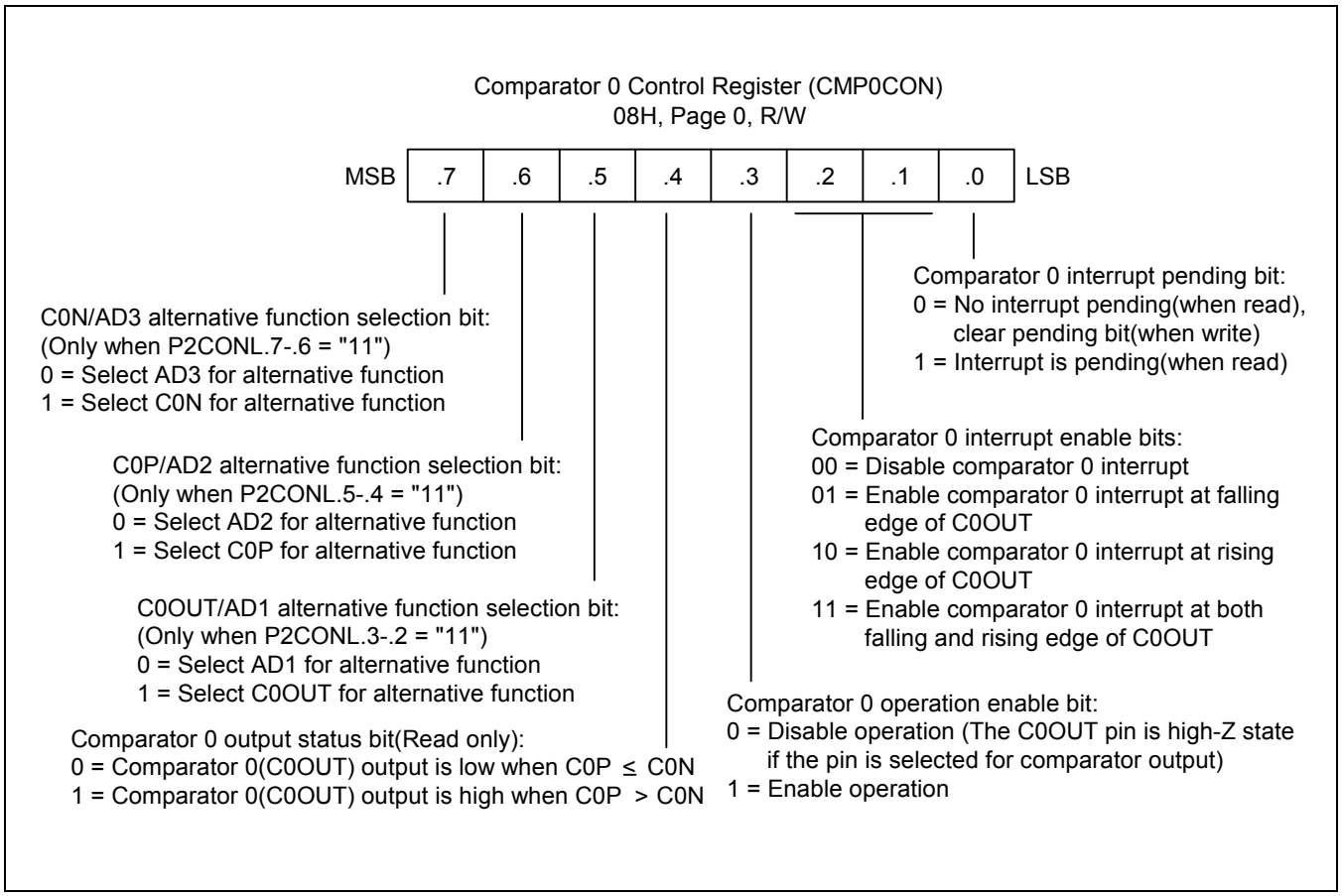


图 22-2 比较器0控制寄存器 (CMP0CON)

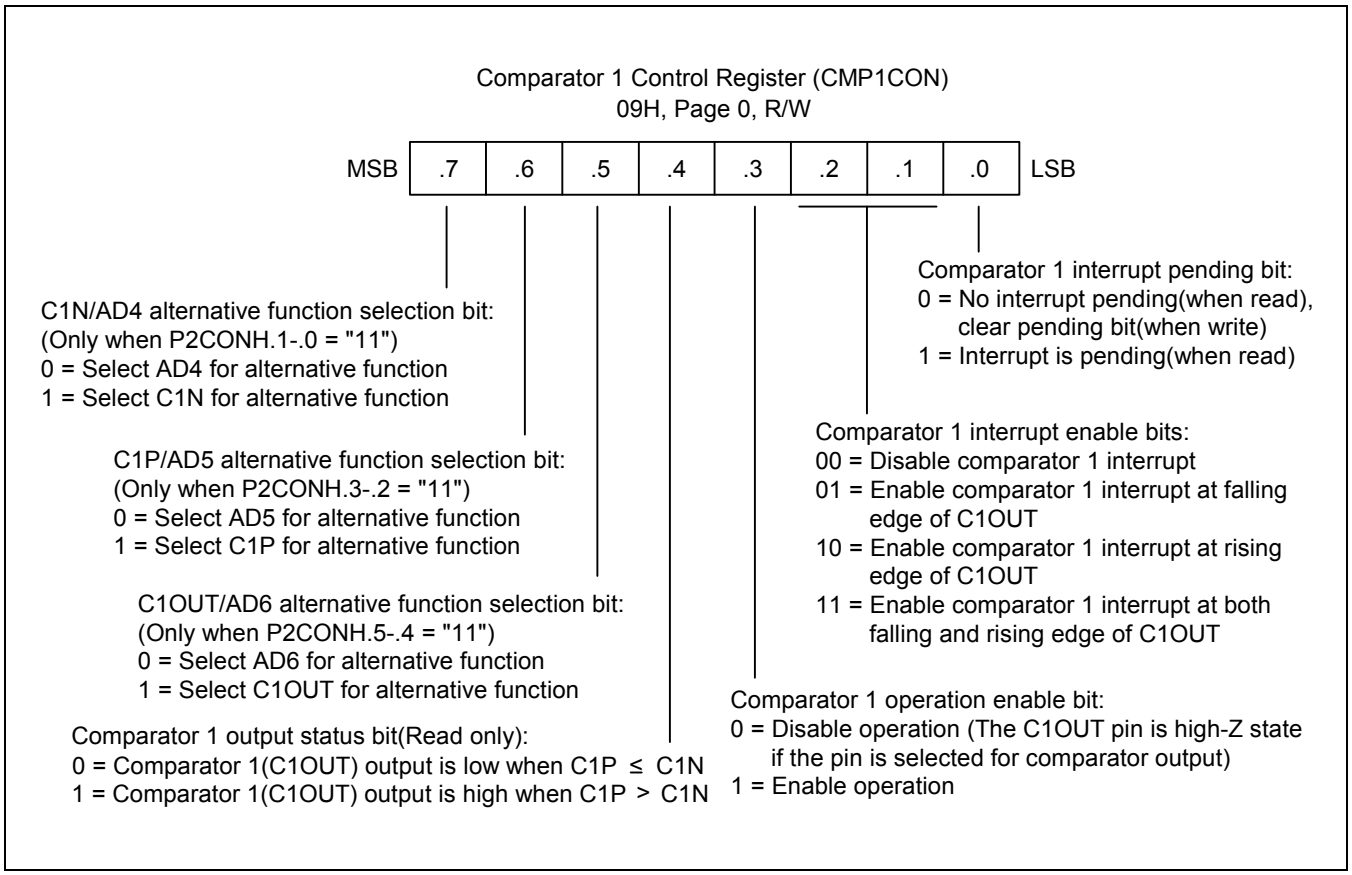


图 22-3 比较器1控制寄存器 (CMP1CON)

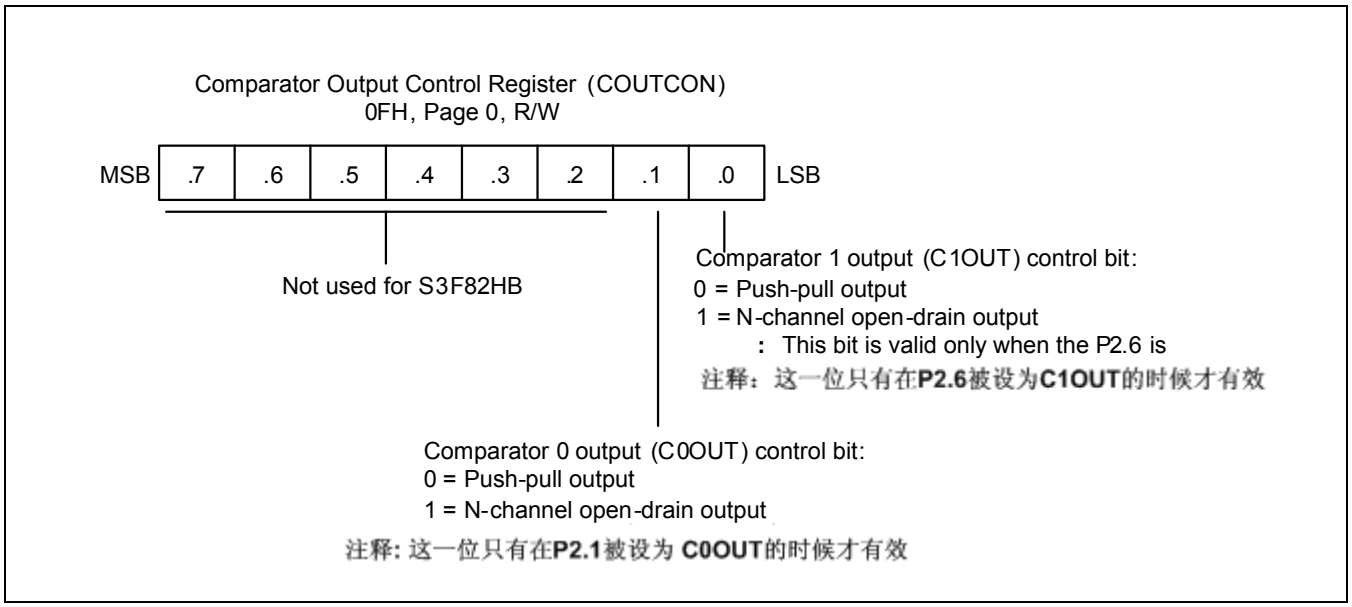


图 22-4 比较器输出控制寄存器 (COUTCON)

23 电气参数

23.1 概述

本章将以表格或图表的方式提供 S3F82HB 的电气参数。按如下顺序排列：

- 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
- 振荡稳定时间
- STOP 模式下，RAM数据保持电压
- LVR 复位时序
- BLD 电气参数
- SIO 时序
- A/D 转换电气特性
- Comparator 电气特性
- LCD 电容偏置电气特性
- UART 时序
- 内部 Flash ROM 电气特性
- 工作电压范围

表 23-1 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 + 4.6	V
输入电压	V_I	P0-10 口	– 0.3 to VDD + 0.3	
输出电压	V_O	–	– 0.3 to VDD + 0.3	
I/O 口输出电流	I_{OH}	单个 I/O 口工作	– 15	mA
		所有 I/O 口工作	– 60	
I/O 口输入电流	I_{OL}	单个 I/O 口工作	+ 30 (Peak value)	
		口所有管脚工作	+ 100 (Peak value)	
工作温度	T_A	–	– 25 to + 85	°C
储藏温度	T_{STG}	–	– 65 to + 150	

表 23-2 直流电气特性

(T_A = – 25°C to + 85°C, V_{DD} = 2.0V to 3.6V)

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	–	3.6	V
		fx = 0.4–12.0MHz	2.7	–	3.6	
输入高电平	VIH1	除 V_{IH2}, V_{IH3} 外所以输入管脚	$0.7V_{DD}$	–	V_{DD}	
	V_{IH2}	P0–1口，nRESET	$0.8V_{DD}$		V_{DD}	
	V_{IH3}	X_{IN}, X_{OUT} 以及 XT_{IN}, XT_{OUT}	V_{DD}–0.1		V_{DD}	
输入低电平	V_{IL1}	除 V_{IL2}, V_{IL3} 外所以输入管脚	–	–	$0.3V_{DD}$	
	V_{IL2}	P0–1口，nRESET			$0.2V_{DD}$	
	V_{IL3}	X_{IN}, X_{OUT} 以及 XT_{IN}, XT_{OUT}			0.1	
输出高电平	V_{OH1}	V_{DD} = 2.7V to 3.6V I_{OH} = –10 mA P2.1, P2.6	V_{DD}–1.3	–	–	V
	V_{OH2}	V_{DD} = 2.7V to 3.6V I_{OH} = –1 mA 除 V_{OH1} 外所有输出管脚	V_{DD}–1.0	–	–	
输出低电平	V_{OL1}	V_{DD} = 2.7V to 3.6V I_{OL} = 15 mA Ports 1–2	–	–	1.0	
	V_{OL2}	V_{DD} = 2.7V to 3.6V I_{OL} = 10 mA 除 V_{OL1} 外所有输出管脚	–	–	1.5	
输入高电平漏电流	I_{LIH1}	V_{IN} = V_{DD} 除 I_{LIH2} 外所有输入管脚	–	–	3	μA
	I_{LIH2}	V_{IN} = V_{DD} ,X_{IN}, X_{OUT}, XT_{IN}, XT_{OUT}			20	
输入低电平漏电流	I_{LIL1}	V_{IN} = 0V 除 nRESET, I_{LIL2} 外所有输入管脚	–	–	–3	
	I_{LIL2}	V_{IN} = 0V, X_{IN}, X_{OUT}, XT_{IN}, XT_{OUT}			–20	
输出高电平漏电流	I_{LOH}	V_{OUT} = V_{DD} 所有输出管脚	–	–	3	
输出低电平漏电流	I_{LOL}	V_{OUT} = 0V 所有输出管脚	–	–	–3	
LCD 电压分频电阻	R_{LCD}	T_A = 25°C	30	60	90	kΩ
振荡器反馈电阻	R_{OSC1}	V_{DD} = 3V, T_A = 25°C X_{IN} = V_{DD}, X_{OUT} = 0V	600	1600	3000	
	R_{OSC2}	V_{DD} = 3V, T_A = 25°C XT_{IN} = V_{DD}, XT_{OUT} = 0V OSCCON.7=0	3000	6000	9000	
上拉电阻	R_{L1}	V_{IN} = 0V; V_{DD} = 3V Ports 0–10, T_A = 25°C	40	70	100	
	R_{L2}	V_{IN} = 0V; V_{DD} = 3V T_A = 25°C, nRESET	220	360	500	
中间输出电压(1)	V_{LC1}	V_{DD} = 2.7V to 3.6V, 1/4 偏置 内部电阻偏置	$0.75V_{DD}$–0.2	$0.75V_{DD}$	$0.75V_{DD}$+0.2	V
	V_{LC2}		$0.50V_{DD}$–0.2	$0.5V_{DD}$	$0.5V_{DD}$+0.2	

参数	标号	条件	最小值	典型值	最大值	单位
	V_{LC3}	LCD 时钟 = 0Hz, $V_{LC0} = V_{DD}$	$0.25V_{DD}–0.2$	$0.25V_{DD}$	$0.25V_{DD}+0.2$	
|VLCD – COMi| 电压跌落 (i = 0 – 7)	V_{DC}	–15μA 每个公共端	–	–	120	mV
|VLCD – SEGx| 电压跌落 (x = 0 – 57)	V_{DS}	–15μA 每个段	–	–	120	

注释: 当 V_{DD} 和 V_{LC0} 管脚连接时的电压是中间输出电压。

表 23-3 直流电气特性

(T_A = – 25°C to + 85°C, V_{DD} = 2.0V to 3.6V)

参数	标号	条件		最小值	典型值	最大值	单位
供电电流(1)	I_{DD1}(2)	Run 模式：V_{DD} = 3.3V ± 0.3V 石英振荡器 C1 = C2 = 22pF	12.0MHz	–	4.5	9.0	mA
			4.0MHz	–	1.8	3.6	
	I_{DD2}(2)	Idle 模式：V_{DD} = 3.3V ± 0.3V 石英振荡器 C1 = C2 = 22pF	12.0MHz	–	1.2	2.4	
			4.0MHz		0.5	1.0	
	I_{DD3}(3)	Run 模式：V_{DD} = 3.3V ± 0.3V, T_A = 25°C 32kHz 石英振荡器		–	13.0	26.0	μA
		Run 模式：V_{DD} = 3.3V ± 0.3V, T_A = 25°C 32kHz 石英振荡器 C = 0.1uF，不加 LCD 面板，电容偏置 LCD 使能		–	16.0	32.0	
	I_{DD4}(3)	Idle 模式：(5) V_{DD} = 3.0V, T_A = 25°C 32kHz 石英振荡器 CLKCON.4-.3 = 00b UART0CONL.3-.2 = 00b UART1CONL.3-.2 = 00b		–	0.9	3.0	
		Idle 模式：(6) V_{DD} = 3.0V, T_A = 25°C 32kHz 石英振荡器 C=0.1uF，不加 LCD 面板，电容偏置 LCD 使能 CLKCON.4-.3 = 00b UART0CONL.3-.2 = 00b UART1CONL.3-.2 = 00b		–	3.9	9.0	
	I_{DD5}(4)	Stop 模式：V_{DD} = 3.0V	T_A = 25°C	–	0.2	2.0	
			T_A = -25°C to + 85°C	–	–	10	

注释:

1. 供电电流不包括内部上拉电阻，LCD分压电阻，LVR模块消耗电流以及外部输出电流负载。
2. I_{DD1} 和 I_{DD2} 包含副时钟振荡的功耗。
3. I_{DD3} 和 I_{DD4} 是主时钟振荡停止，副时钟使用 (OSCCON.7 = 1) 时的电流。
4. I_{DD5} 是主时钟和副时钟振荡都停止时的电流。
5. 所有外设模块停止。
6. 仅 watch timer 工作。
7. 表中除 IDD4 外的所有值都是在系统时钟控制寄存器 (CLKCON.4–.3) 的位4-3设置成11B时的结果。

表 23-4 交流电气特性

(T_A = – 25°C to + 85°C, V_{DD} = 2.0V to 3.6V)

参数	标号	条件	最小值	典型值	最大值	单位
TACLK, T0CLK, T1CLK, 和 FCLK 的周期	t_{CLK}	不带上拉电阻的输入	400	–	–	ns
TACLK, T0CLK, T1CLK, 和 FCLK 输入高，低电平宽度	t_{CLKH}, t_{CLKL}		200			
中断输入高，低电平宽度	t_{INTH}, t_{INTL}	所有中断，V_{DD} = 3V	500	–	–	ns
nRESET 输入低电平宽度	t_{RSL}	V_{DD} = 3V	10	–	–	µs

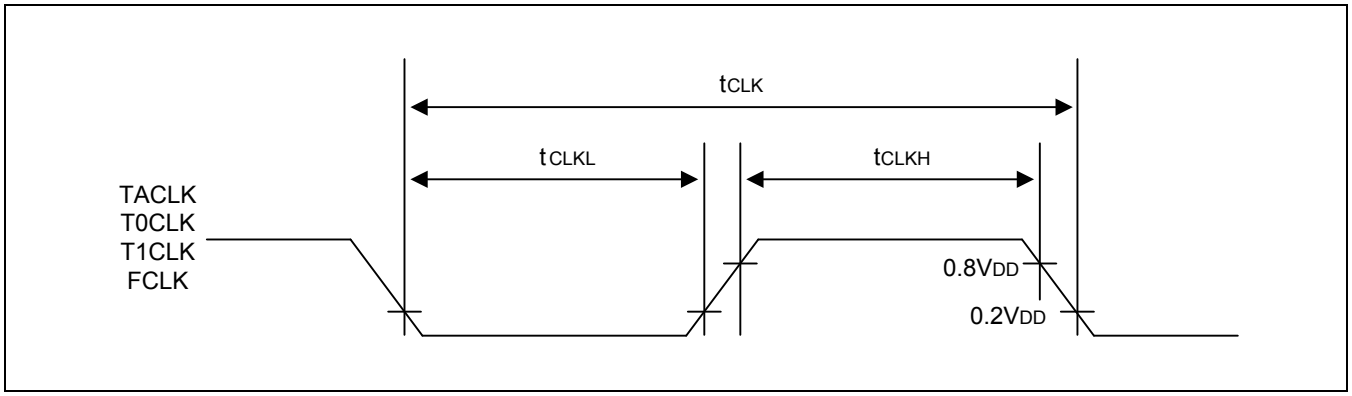


图 23-1 TACLK/T0CLK/T1CLK/FCLK 输入时序

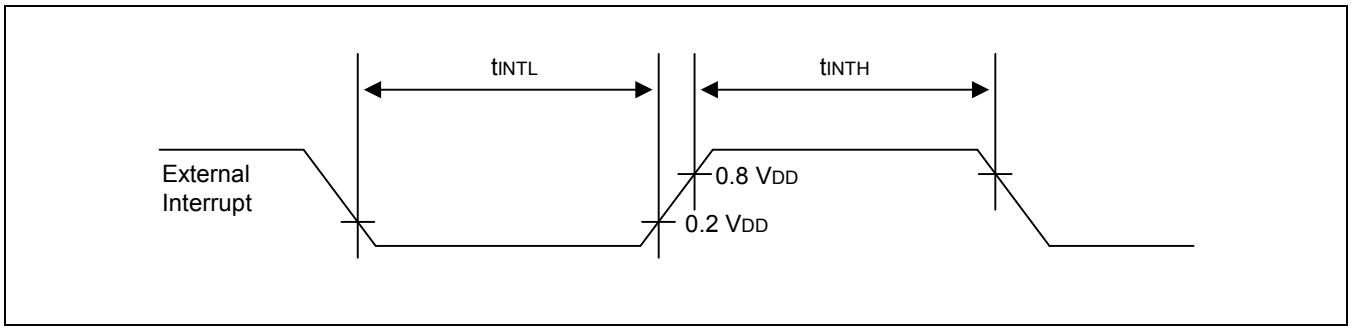


图 23-2 外部中断输入时序

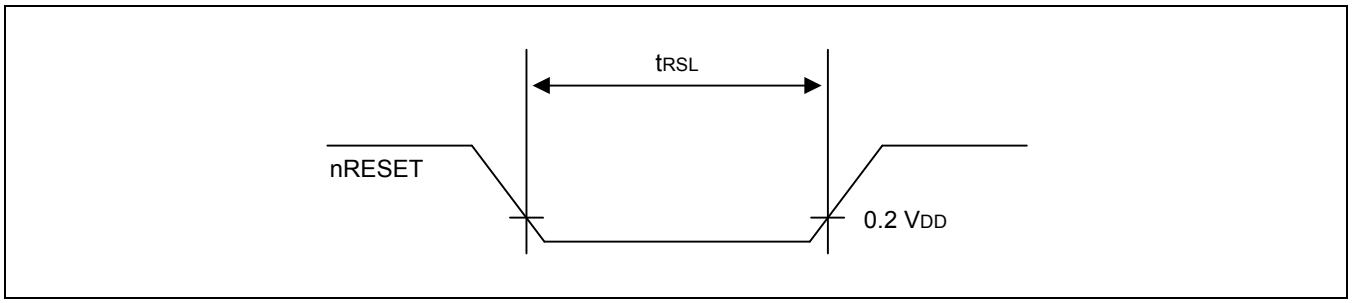


图 23-3 nRESET 输入时序

表 23-5 输入/输出电容

(T_A = – 25°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 = 1MHz; 不测试的管脚接 V_{SS}	–	–	10	pF
输出电容	C_{OUT}					
I/O 口电容	C_{IO}					

表 23-6 STOP 模式下数据保持电压

(T_A = – 25°C to + 85°C)

参数	标号	条件	最小值	典型值	最大值	单位
数据保持供电电压	V_{DDDR}		2.0	–	3.6	V
数据保持供电电流	I_{DDDR}	Stop 模式，T_A = 25°C V_{DDDR} = 2.0V 禁止 LVR 模块	–	–	1	μA

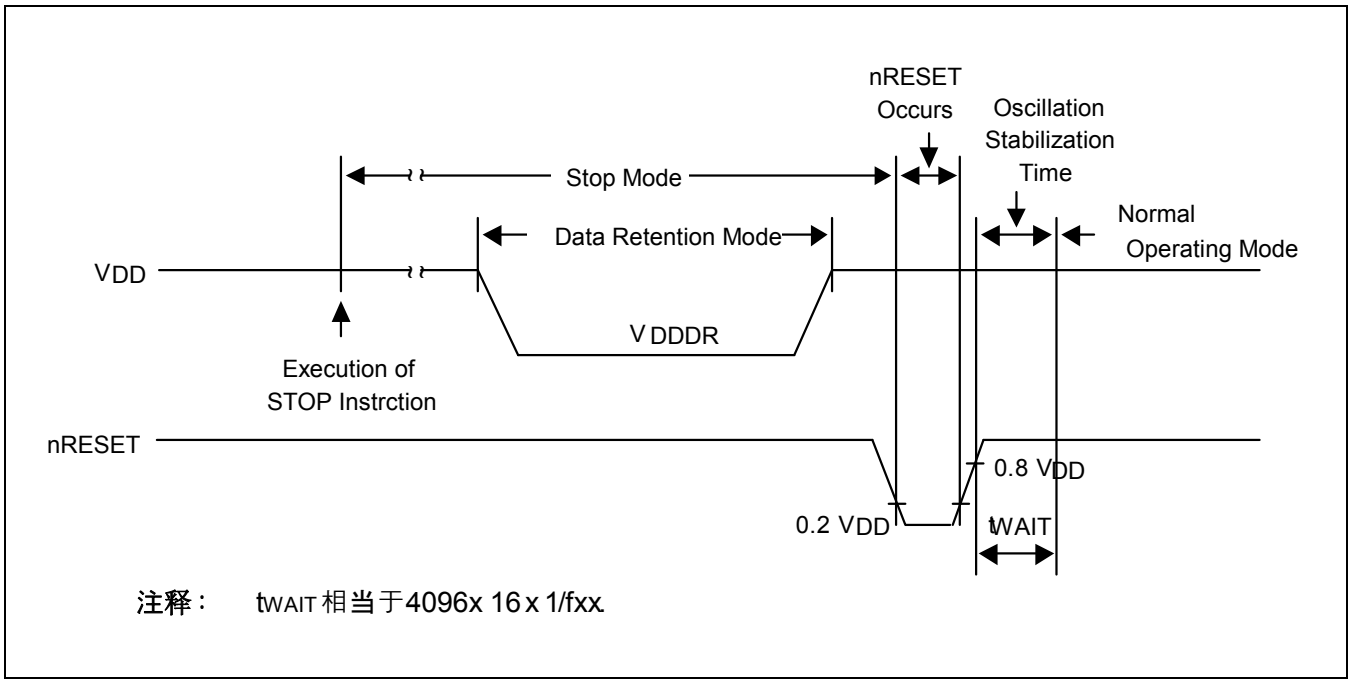


图 23-4 RESET 退出STOP模式时序

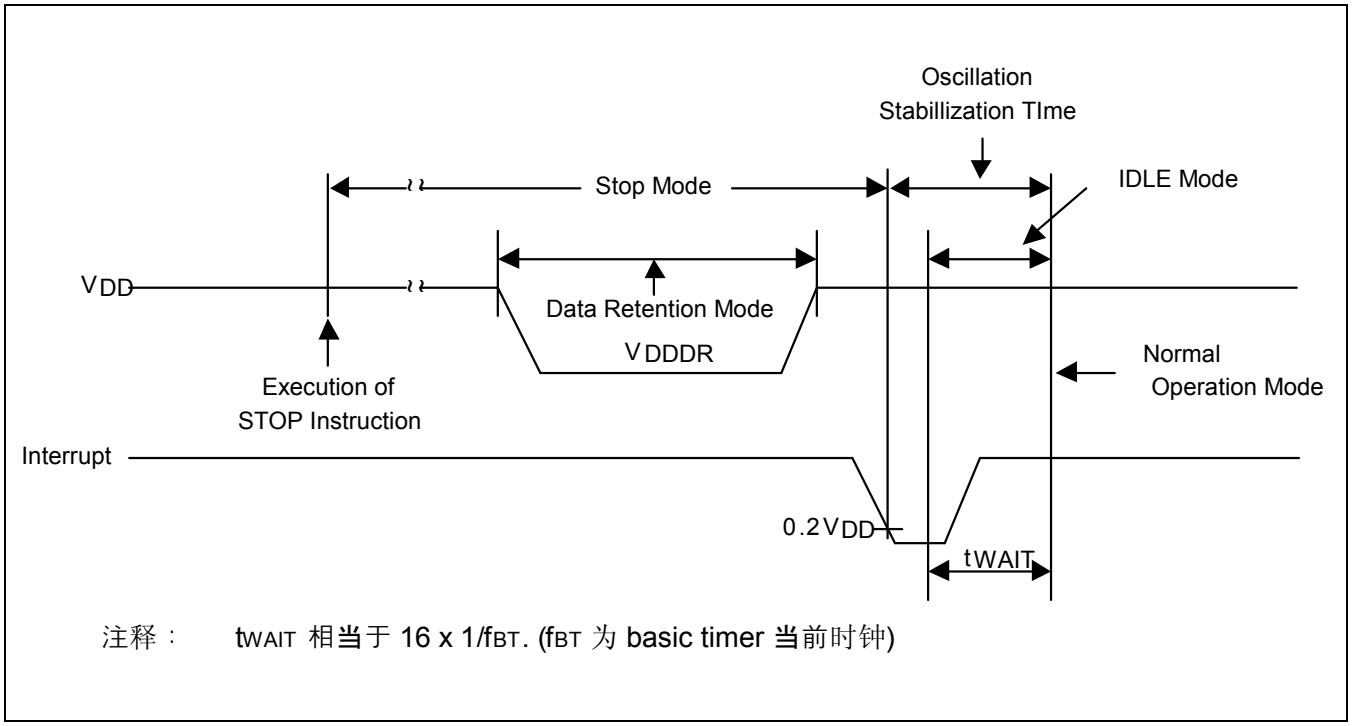


图 23-5 中断退出STOP模式时序

表 23-7 A/D 转换电气特性

(T_A = – 25°C to + 85°C, V_{DD} = 2.0V to 3.6V)

参数	标号	条件	最小值	典型值	最大值	单位
精度		–	–	10	–	bit
总误差		–	–	–	± 3	LSB
积分线性误差	ILE	V_{DD} = 3.072V V_{SS} = 0 V CPU 时钟 = 8.0MHz	–	–	± 2	
微分线性误差	DLE			–	± 1	
顶部偏移误差	EOT			± 1	± 3	
底部偏移误差	EOB			± 1	±3	
转换时间(1)	T_{CON}	10位精度 50 x fxx/4, fxx = 8MHz	25	–	–	μS
模拟输入电压	V_{IAN}	–	AV_{SS}	–	AV_{REF}	V
模拟输入阻抗	R_{AN}	–	2	1000	–	M
模拟参考电压	AV_{REF}	–	2.0	–	V_{DD}	V
模拟地	AV_{SS}	–	V_{SS}	–	V_{SS} + 0.3	
模拟输入电流	I_{ADIN}	V_{DD} = 3.3V	–	–	5	μA
模拟模块电流(2)	I_{ADC}	V_{DD} = 3.3V	–	0.5	1.5	mA
		V_{DD} = 3.3V 省电模式		100	500	nA

注释:

1. “转换时间”是启动转换到转换结束所需时间。
2. I_{ADC} 是 A/D 转换的工作电流。

表 23-8 Comparator 电气特性

(T_A = – 25°C to + 85°C, V_{DD} = 2.0V to 3.6V)

参数	标号	条件	最小值	典型值	最大值	单位
Comparator 输入电压	V_{CP}, V_{CN}	正向，反向输入	V_{SS}	–	V_{DD}	V
Comparator 偏置电压	V_{OF}	V_{IN} = 0.5V to 2.5V 当 V_{DD} = 3.0V	–	–	15	mV
响应时间	T_{DA}	当 V_{DD} = 3.0V V_{CP} = 1.5V V_{CN} = V_{CP} ± 15mV	–	–	3	μS
Comparator 模块电流	I_{CMP}	V_{DD} = 3.3V，使能时	–	0.3	0.6	mA
		V_{DD} = 3.3V，禁止时		100	500	μA
输入频率	F_{IN}	当 V_{DD} = 3.0V V_{CP} = 1.5V V_{CN} = F_{IN}	–	–	1	MHz

表 23-9 低电压复位电气特性

(T_A = 25°C)

参数	标号	条件	最小值	典型值	最大值	单位
LVR 电压	V_{LVR}	T_A = 25°C	2.0	2.2	2.4	V
VDD 电压上升时间	t_R	–	10	–	–	μS
VDD 电压关断时间	t_{OFF}	–	0.5	–	–	S
LVR 时滞电压	ΔV	–	–	10	100	mV
电源消耗	I_{DDPR}	V_{DD} = 3.3V	–	70	120	μA

注释: 在 “Smart Option” 中使能 LVR 后，LVR 电路就有电流消耗。

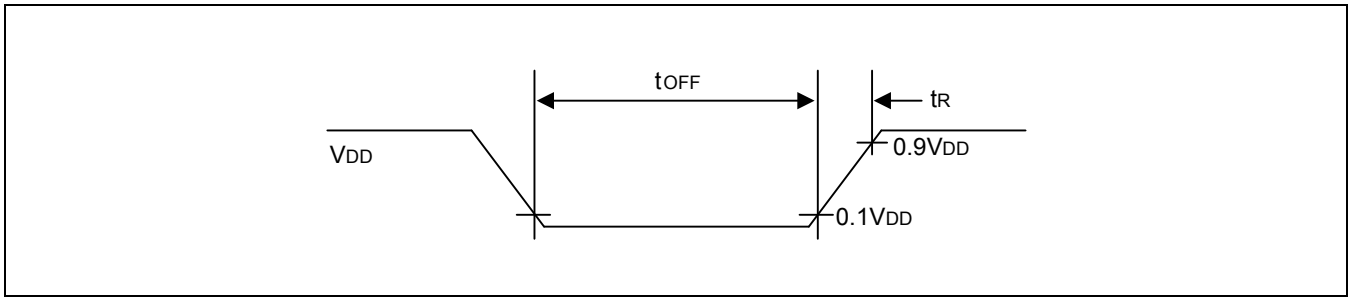


图 23-6 LVR (低电压复位) 时序

表 23-10 电池电压检测电气特性

(T_A = 25°C, V_{DD} = 2.0V to 3.6V)

参数	标号	条件	最小值	典型值	最大值	单位
BLD工 作电压	V_{DDBLD}	–	2.0	–	3.6	V
BLD 电压	V_{BLD}	BLDCON.2–.0 = 000b	2.0	2.2	2.4	V
		BLDCON.2–.0 = 101b	2.15	2.4	2.65	
		BLDCON.2–.0 = 011b	2.5	2.8	3.1	
BLD 时滞电压	ΔV	BLDCON.2–.0 = 000, 101, 011b	–	10	100	mV
电流消耗	I_{BLD}	V_{DD} = 3.3V	–	70	120	μA
		V_{DD} = 2.2V	–	50	100	
BLD 电路响应时间	T_B	fw = 32.768kHz	–	–	1	ms

表 23-11 同步 SIO 电气特性

(T_A = – 25°C to + 85°C, V_{DD} = 2.0V to 3.6V)

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	400	–	–	ns
		内部 SCK 源	400			
SCK 高，低电平宽度	t_{KH}, t_{KL}	外部 SCK 源	200			
		内部 SCK 源	t_{KCY}/2-20			
相对于SCK 高电平的SI建立时间	t_{SIK}	外部 SCK 源	100			
		内部 SCK 源	100			
相对于SCK 高电平的SI保持时间	t_{KSI}	外部 SCK 源	160			
		内部 SCK 源	160			
相对于SO的SCK 输出延时	t_{KSO}	外部 SCK 源	–		120	
		内部 SCK 源			100	

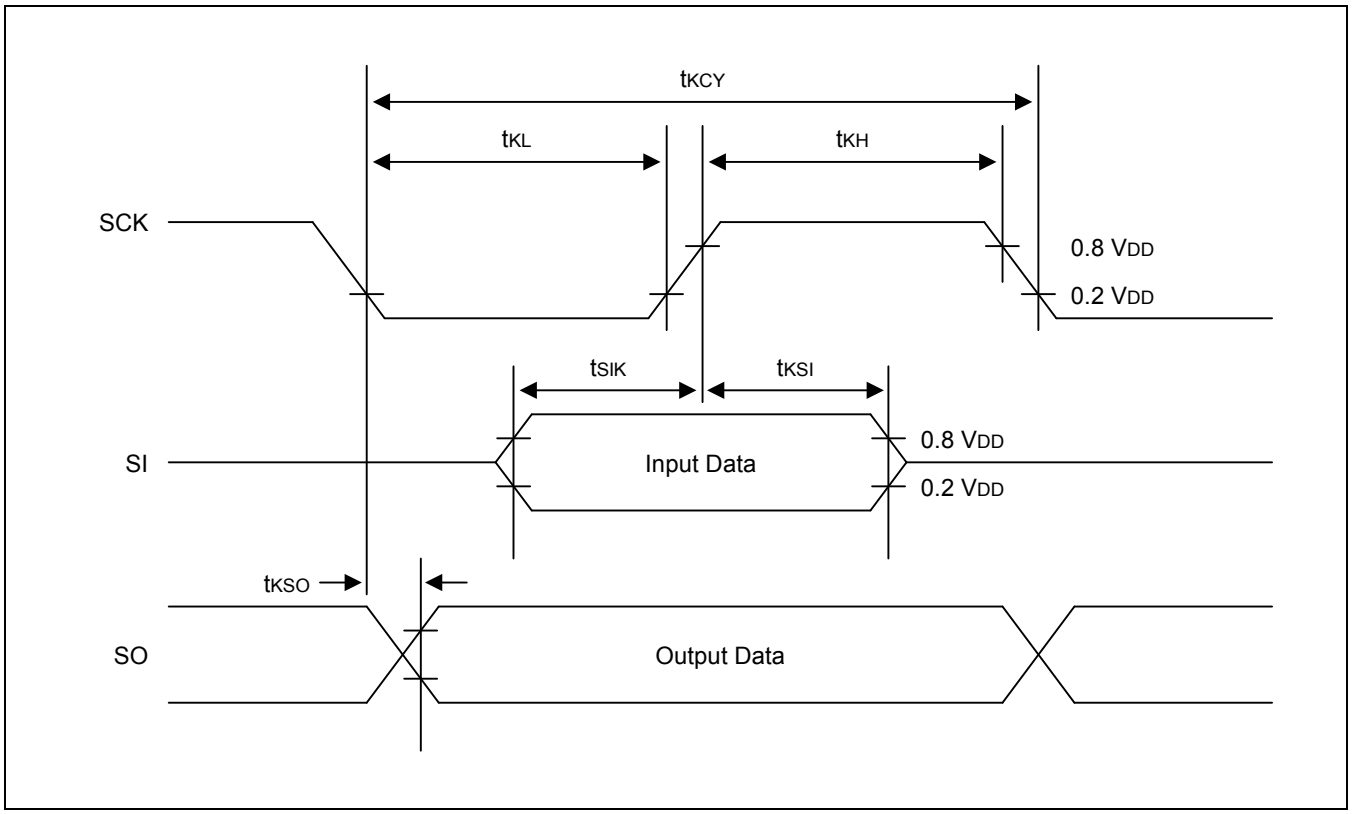


图 23-7 串行数据传输时序

表 23-12 UART 模式0 时序特性 (12.0MHz)

(T_A = – 25°C to + 85°C, V_{DD} = 2.0V to 3.6V, 负载电容 = 80pF)

参数	标号	条件	最小值	典型值	最大值
串行口时钟周期	t_{SCK}	1,160	$t_U \times 16$	1,500	ns
相对于时钟上升沿的输出数据建立时间	t_{S1}	500	$t_U \times 13$	–	
相对于输入数据有效的时钟上升沿	t_{S2}	–	–	500	
时钟上升沿后的输出数据保持时间	t_{H1}	t_{CPU} – 50	t_U	–	
时钟上升沿后的输入数据保持时间	t_{H2}	0	–	–	
串行口时钟高、低电平宽度	t_{HIGH}, t_{LOW}	450	$t_U \times 8$	890	

注释:

1. 所有的时序都是纳秒 (ns) 级的，并且CPU时钟为 12.0 MHz。
2. 单位 t_U 表示1个 UART 时钟周期。

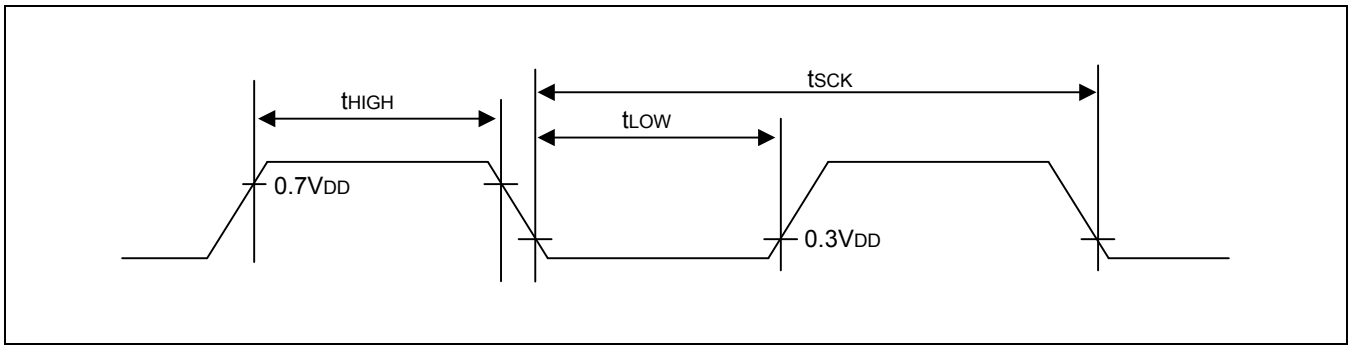
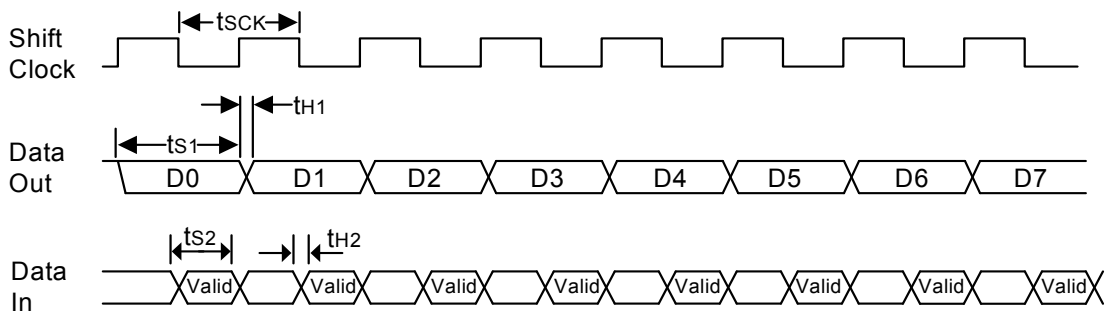


图 23-8 UART 时序特性波形



注释： 本图中的符号定义如下：

f_{SCK}	串行口时钟周期
t_{S1}	相对于时钟上升沿的输出数据建立时间
t_{S2}	相对于输入数据有效的时钟上升沿时间
t_{H1}	时钟上升沿后的输出数据保持时间
t_{H2}	时钟上升沿后的输入数据保持时间

图 23-9 UART 模块的时序波形

表 23-13 LCD 电容偏置电气特性

(T_A = – 25°C to + 85°C, V_{DD} = 2.0V to 3.6V)

参数	标号	条件		最小值	典型值	最大值	单位
液晶驱动电压	V_{LC3}	V_{LC3} 和 V_{SS} 之间接一个1MΩ的负载电阻 (无LCD面板) T_A = 25°C V_{DD} = 3.0V	LCONST.2-.0 = 0	Typ. × 0.85	0.62	Typ. × 1.15	V
			LCONST.2-.0 = 1		0.65		
			LCONST.2-.0 = 2		0.68		
			LCONST.2-.0 = 3		0.71		
			LCONST.2-.0 = 4		0.74		
			LCONST.2-.0 = 5		0.77		
			LCONST.2-.0 = 6		0.80		
			LCONST.2-.0 = 7		0.83		
	V_{LC2}	V_{LC2} 和 V_{SS}之间接一个1MΩ 的负载电阻 (无LCD面板)		$2 \times V_{LC3} \times 0.9$	–	$2 \times V_{LC3} \times 1.1$	
	V_{LC1}	V_{LC1} 和 V_{SS}之间接一个1MΩ 的负载电阻 (无LCD面板)		$3 \times V_{LC3} \times 0.9$	–	$3 \times V_{LC3} \times 1.1$	
	V_{LC0}	V_{LC0} 和 V_{SS}之间接一个1MΩ 的负载电阻 (无LCD面板)		$4 \times V_{LC3} \times 0.9$	–	$4 \times V_{LC3} \times 1.1$	

注释: 这是 1MΩ负载电阻只连接到一个信号点 (V_{LC0} - V_{LC3}) 时的特性。V_{LCN}(N = 0-2)的值取决于 VLC3 的实测值。

表 23-14 主振荡器特性

(T_A = – 25°C to + 85°C)

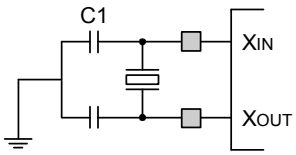
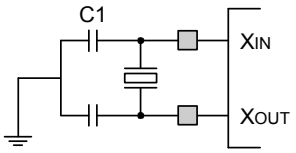
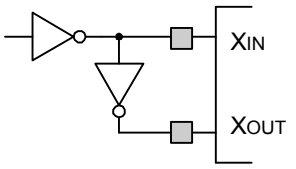
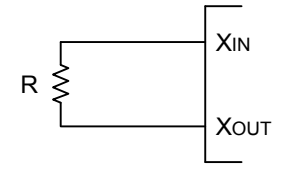
振荡器	时钟配置	参数	测试条件	最小值	典型值	最大值	单位
石英振荡器		主振荡频率	2.7V – 3.6V	0.4	–	12.0	MHz
			2.0V – 3.6V	0.4	–	4.2	
陶瓷振荡器		主振荡频率	2.7V – 3.6V	0.4	–	12.0	
			2.0V – 3.6V	0.4	–	4.2	
外部时钟源		X_{IN} 管脚输入频率	2.7V – 3.6V	0.4	–	12.0	
			2.0V – 3.6V	0.4	–	4.2	
RC 振荡器		频率	3.3V	0.4	–	1.0	MHz

表 23-15 副振荡器特性

(T_A = – 25°C to + 85°C)

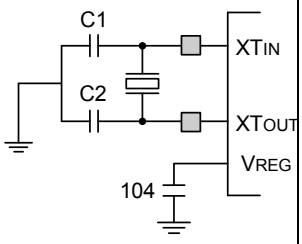
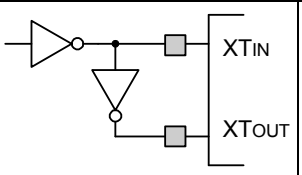
振荡器	时钟配置	参数	测试条件	最小值	典型值	最大值	单位
石英振荡器		副振荡频率 C1 = 22pF C2 = 15pF	2.0V – 3.6V	32	32.768	35	kHz
外部时钟源		XT_{IN}管脚输入频率	2.0V – 3.6V	32	–	100	

表 23-16 主振荡稳定时间

(T_A = – 25°C to + 85°C, V_{DD} = 2.0V to 3.6V)

振荡器	测试条件	最小值	典型值	最大值	单位
石英振荡器	fx > 1MHz	–	–	40	ms
陶瓷振荡器	V_{DD} 满足最小起振电压时振荡稳定。	–	–	10	ms
外部时钟源	X_{IN} 输入高低电平宽度 (t_{XH}, t_{XL})	62.5	–	1250	ns

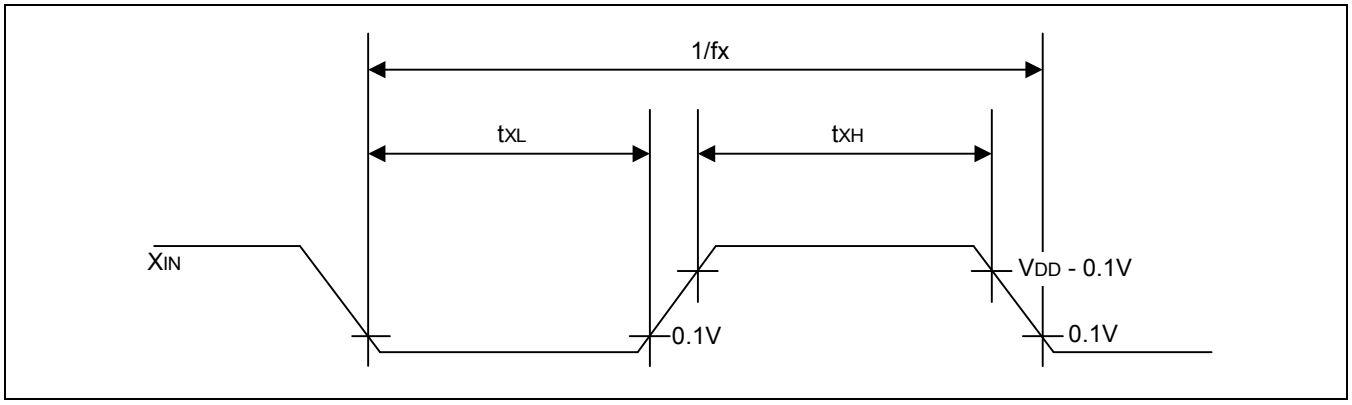


图 23-10 X_{IN} 管脚测得的时钟时序

表 23-17 副振荡稳定时间

(T_A = – 25°C to + 85°C, V_{DD} = 2.0V to 3.6V)

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
外部时钟源	XT_{IN} 输入高低电平宽度 (t_{XTH}, t_{XTL})	5	–	15	μs

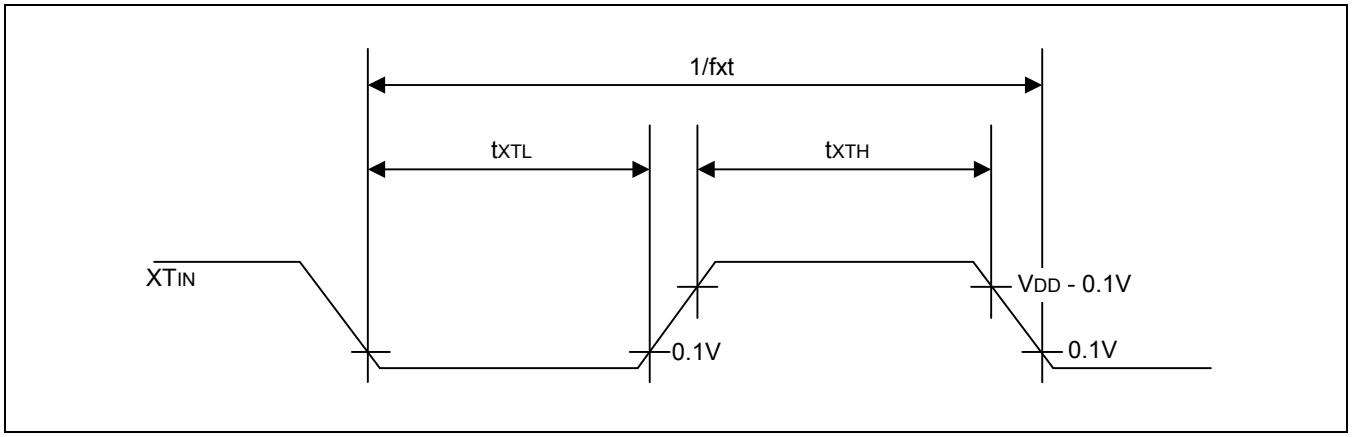


图 23-11 XT_{IN}管脚测得的时钟时序

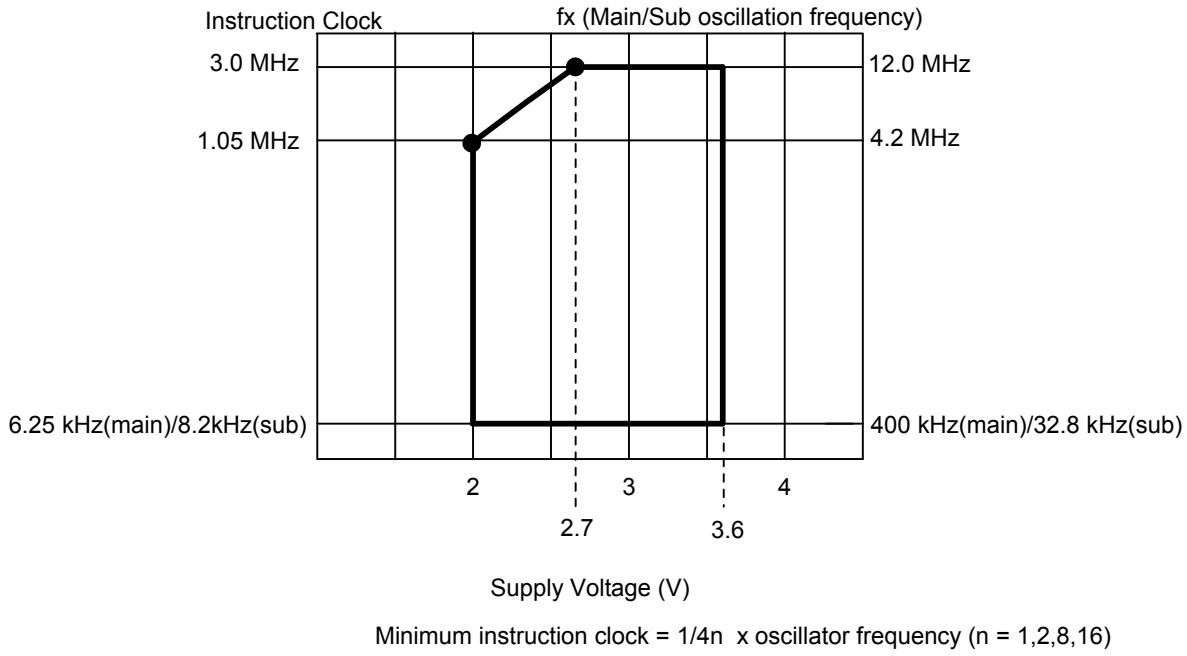


图 23-12 工作电压范围

表 23-18 内部 Flash ROM 电气特性

(T_A = – 25°C to + 85°C, V_{DD} = 2.2V to 3.6V)

参数	标号	条件	最小值	典型值	最大值	单位
编程时间(1)	Ftp	–	30	–	–	μs
芯片擦除时间(2)	Ftp1		50	–	–	ms
扇区擦除时间(3)	Ftp2		10	–	–	ms
读频率	f_R	–	–	–	12	MHz
写/擦除次数	FN_{WE}	–	–	–	10,000(4)	Times

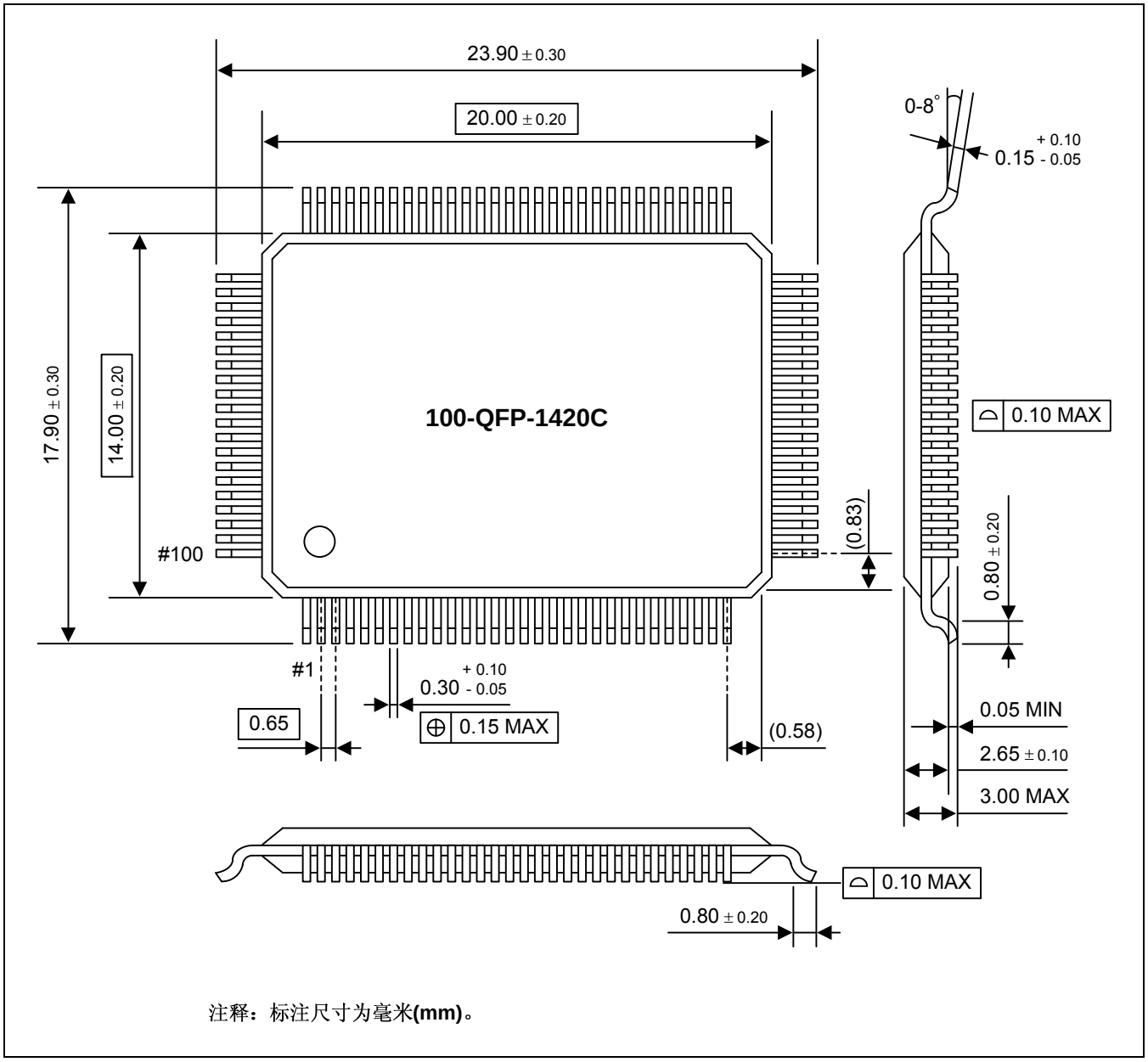
注释:

1. 编程时间是编程一字节(8 位)所需要的时间。
2. 芯片擦除时间是擦除整个64K 字节所需要的时间。
3. 扇区擦除时间是擦除一个扇区内所有128 字节所需要的时间。
4. 芯片擦除只在工具编程模式下可用。

24 机械尺寸

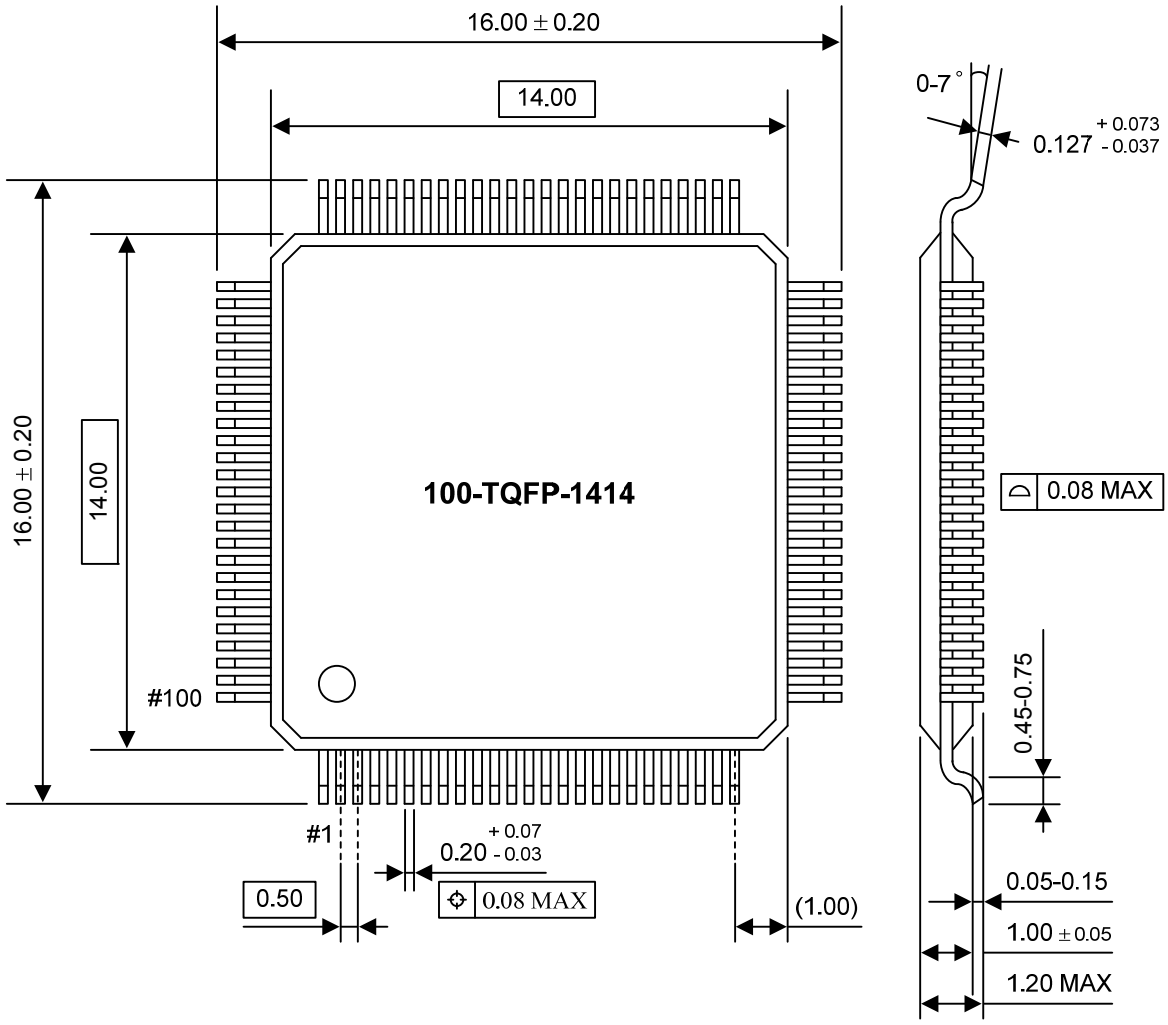
24.1 概述

S3F82HB 现有的封装为 100-pin-QFP/TQFP 封装。



注释：标注尺寸为毫米**(mm)**。

图 24-1 封装尺寸 (100-QFP-1420C)



注释：标注尺寸为毫米**(mm)**。

图 **24-2** 封装尺寸 **(100-TQFP-1414)**

25 S3F82HB FLASH MCU

25.1 概述

S3F82HB单片 CMOS MCU 是一款 Flash MCU。
它具有片上 64K 字节闪存空间，可以通过串行数据格式访问。

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 19 章”嵌入式闪存接口”。

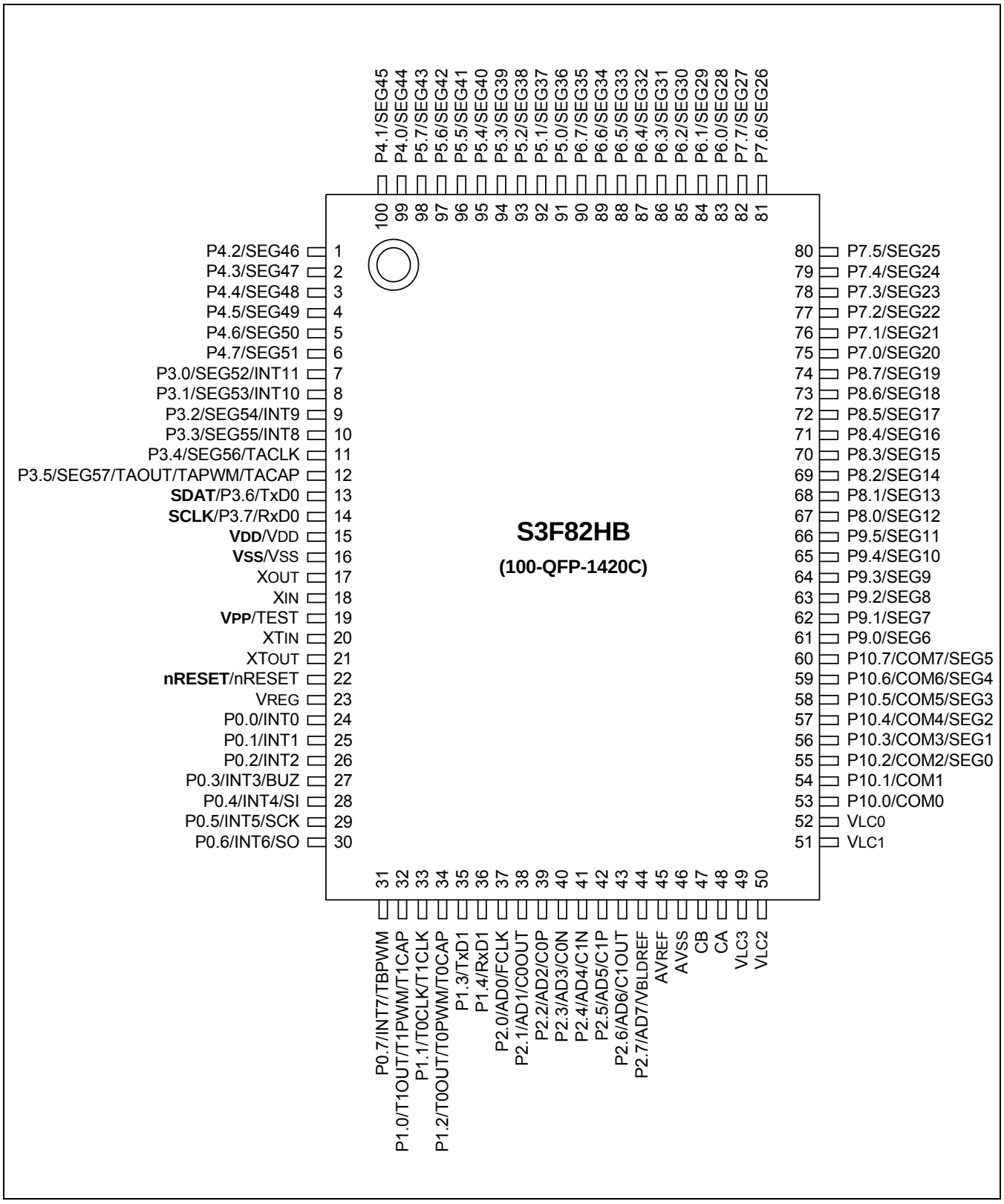
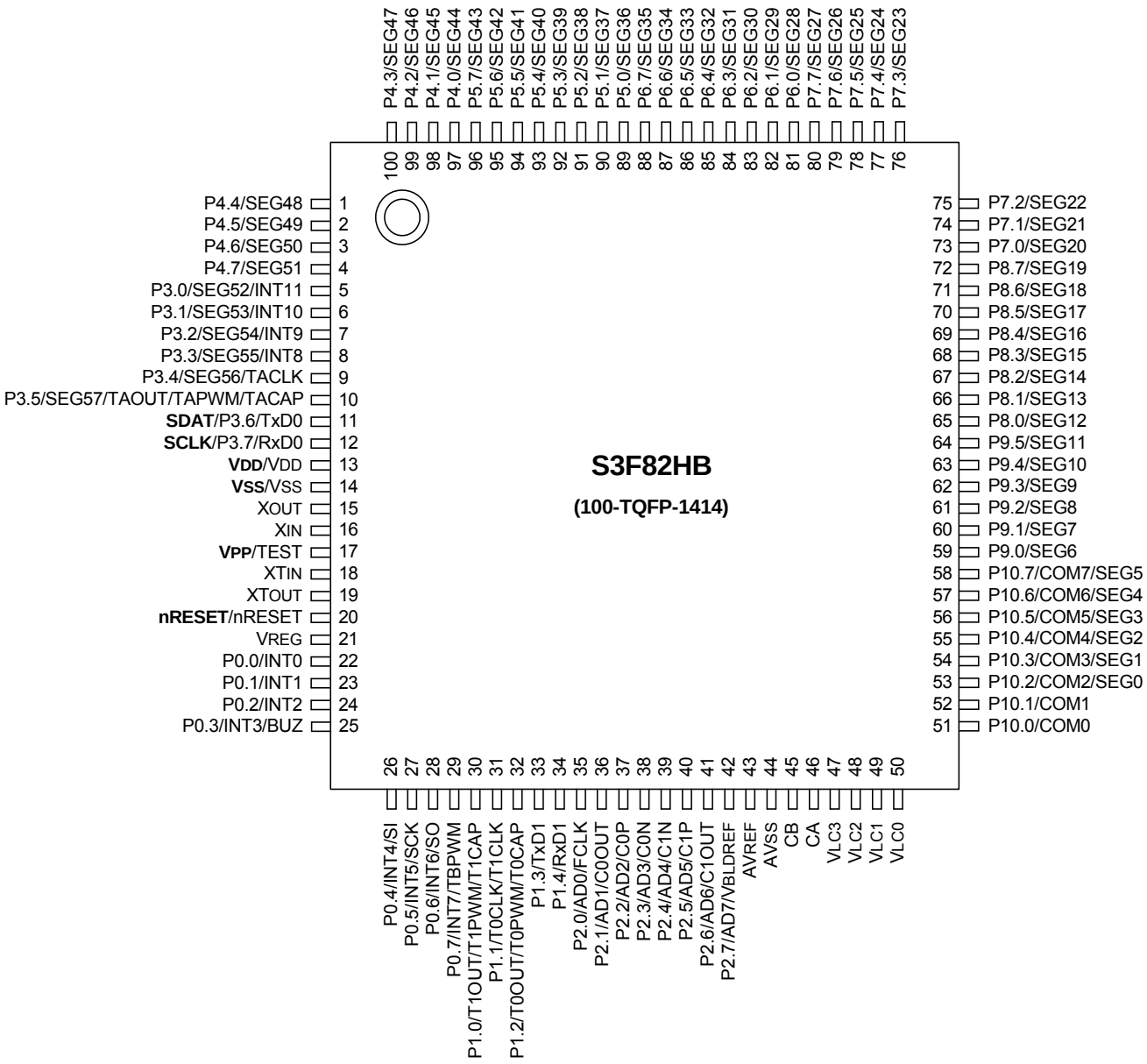


图 25-1 S3F82HB 管脚分配 (100-QFP-1420C)



NOTE: The sequence of pins in TQFP package is disagreement with that in QFP package.

图 25-2 S3F82HB 管脚分配 (100-TQFP-1414)

表 25-1 闪存读写管脚描述

主芯片	编程过程中			
管脚名	管脚名称	管脚 No	I/O	功能
P3.6	SDAT	13(11)	I/O	串行数据管脚 (读时为输出脚，写入时为输入脚)，管脚可设置为输入和推挽式输出模式。
P3.7	SCLK	14(12)	I/O	串行时钟管脚 (仅为输入管脚)。
TEST	V_{PP}	19(17)	I	用于Tool模式下flash单元编程的供电管脚 (指示MTP进入编程模式)。VPP为逻辑高电平，MTP即处于编程模式。 当使用编程工具时(如spw2+)必须将TEST/VPP 管脚连到 VDD. (S3F82HB 内部用自升压电路可以产生12.5V 的编程电压)。
nRESET	nRESET	22(20)	I	芯片初始化。
V_{DD}, V_{SS}	V_{DD}, V_{SS}	15(13) 16(14)	–	逻辑供电管脚。

注释: 括号内是 100-TQFP-1414 封装的芯片管脚 No.

25.2 在板编程（ON BOARD WRITING）

S3F82HB 只需要包括 VDD 和 GND在内的6 根线就可以用编程工具通过串行协议进行内部闪存的编程。

25.2.1 电路设计指导

flash编程时，编程工具需要的6 条信号线为：GND，VDD，RESET，VPP，SDA和 SCL 。当进行 PCB 电路设计时，应该考虑为这些信号线留编程接口。

TEST 管脚在一般应用中都可以直接连到GND，但是为了支持工具烧写，
必须在TEST管脚和GND之间串接一个电阻。RESET，SDA和SCL管脚也应该做同样的处理。*图 25-3*。

请仔细设计和这些信号管脚相关的电路，因为 VPP, SCL 和 SDA的上升/下降时序可能直接决定编程是否成功。

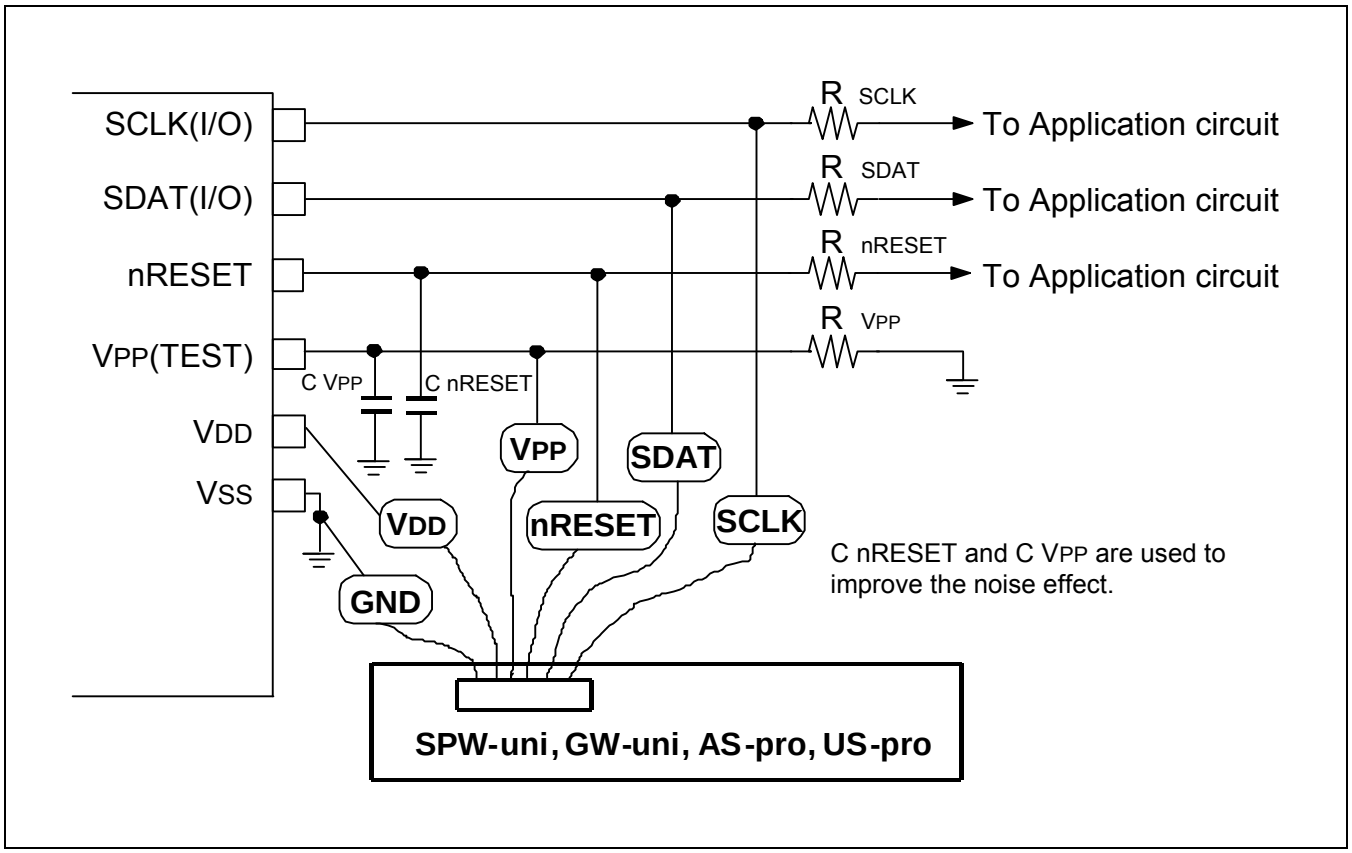


图 25-3 编程接口(在板编程) PCB 设计指导

25.2.2 连接参照表

表 25-2 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} 为 10kΩ ~ 50kΩ C_{Vpp} 为 0.01uF ~ 0.02uF
nRESET	输入	需要	R_{nRESET} 为 2kΩ ~ 5kΩ C_{nRESET} 为 0.01uF ~ 0.02uF
SDAT(I/O)	输出	不需要(注释)	R_{SDAT} 为 2kΩ ~ 5kΩ
	输入	需要	–
SCLK(I/O)	输出	不需要(注释)	R_{SCLK} 为 2kΩ~ 5kΩ
	输入	需要	–

注释:

1. 在板编程模式中，管脚 SCL 和 SDA上通过的是高速信号。所以，如果应用电路设计为高速响应，例如延迟控制电路；那么因为编程接口的存在，管脚时序可能发生偏移，从而对连接到 SCLK和SDAT 口的应用电路就有可能造成一定的干扰。 所以如果可能，最好在设计应用电路时将SDAT, SCLK设计为输入管脚。
2. 表中 R, C 的值仅为推荐值。在不同的电路系统中它们的值也不同。

26 开发工具

26.1 概述

三星提供了一套强大易用的开发工具，该开发工具由一个主系统，一套调试工具和相应的支持软件组成。所谓主系统，即任何一台使用 Win95/98/2000/XP 操作系统的标准电脑。 成熟的调试工具包含了硬件和软件：一个强大的电路仿真器，OPENice-i500 或SK-1200，支持所有的 S3F7-, S3F9- 和 S3F8- MCU 家族。
三星同时可提供各种支持仿真器的软件， 包括调试器，汇编编译器和仿真器安装设置程序。

26.1.1 目标板

S3C9/S3F9- 系列 MCU 调试时需要目标板，该板上提供了所有需要的目标系统连线和调试接口。
TB82HB 是为开发S3F82HB 专用的目标板。

26.1.2 编程插座适配器

当用仿真器或 OTP/MTP 编程器对 S3F82HB 的闪存进行编程时，S3F82HB 需要专用的编程适配器。

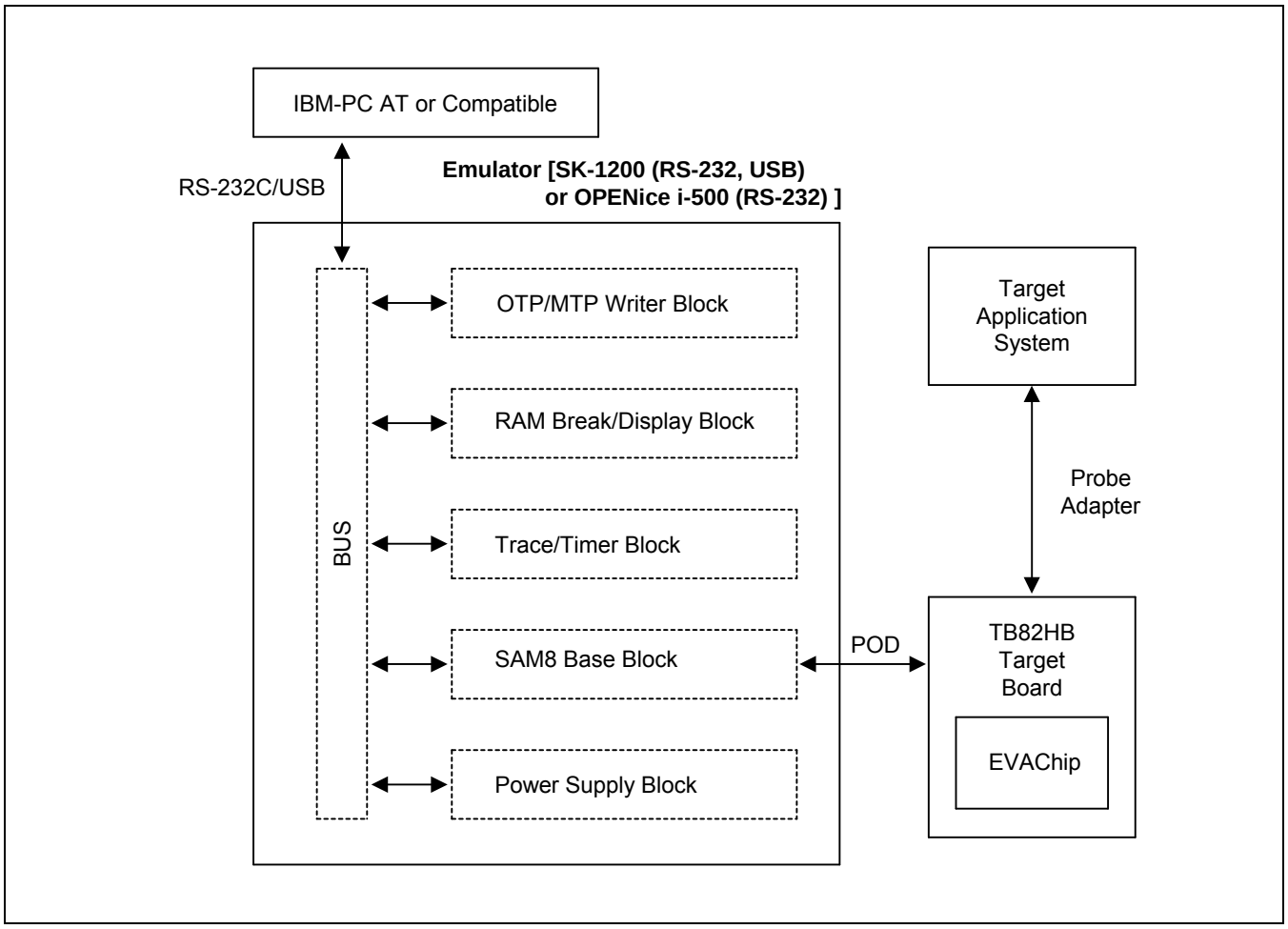

IBM-PC AT or Compatible
RS-232C/USB
Emulator [SK-1200 (RS-232, USB)
or OPENice i-500 (RS-232)]
OTP/MTP Writer Block
RAM Break/Display Block
Trace/Timer Block
SAM8 Base Block
Power Supply Block
BUS
POD
Target
Application
System
Probe
Adapter
TB82HB
Target
Board
EVAChip

图 26-1 开发系统配置

26.1.3 TB82HB 目标板

TB82HB 目标板可用于 S3F82HB MCU的开发。TB82HB 目标板与仿真器 (SK-1200, OPENice-i500) 相连时作为目标CPU工作。

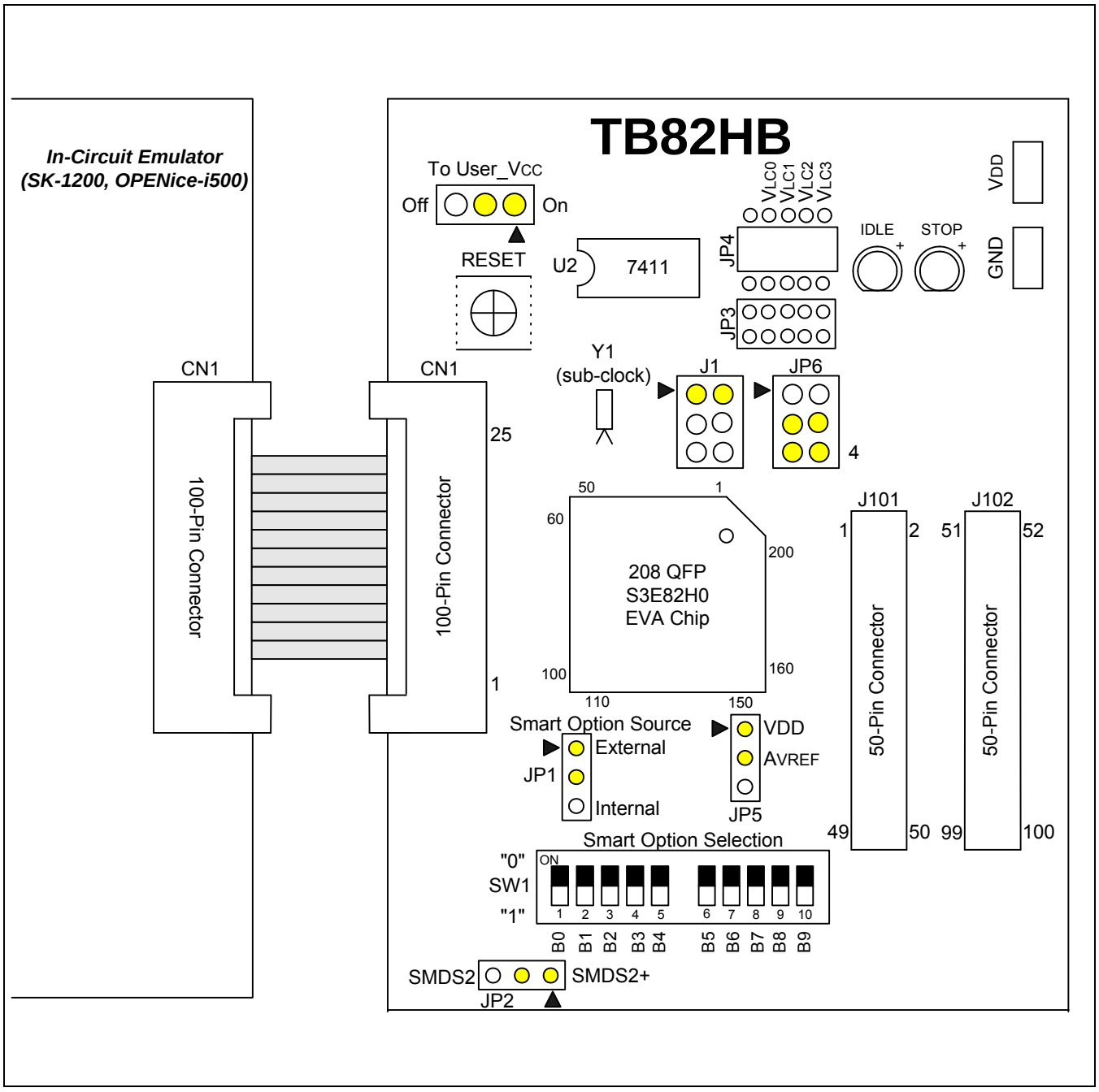


图 26-2 TB82HB 目标板配置

注释: 符号 “◄ ” 代表跳线的起始点。

表 26-1 TB82HB 构件

标号	用途	描述
CN1	100-pin 接口	连接仿真器和 TB82HB 目标板。
J101, J102	50-pin 接口	连接目标板和用户应用系统。
RESET	按键	产生S3F82HB EVA 芯片的低电平复位信号。
VCC, GND	电源接口	TB82HB的外部电源接口。
STOP, IDLE LED	STOP/IDLE 指示	指示 TB82HB 目标板上 S3F82HB EVA-芯片处于STOP 或 IDLE 状态。

表 26-2 TB82HB 跳线设置

JP#	描述	1-2 连接	2-3 连接	默认设置
JP5	AVREF 电源	VDD	用户电源	连接 1-2
JP6	时钟源选择	当使用仿真器提供的内部时钟源时，连接 2-3 和 4-5。若使用外部时钟源如石英晶振时，必须将跳线设置从 1-2 变成 5-6，并将 J1 与外部时钟源相连。		仿真器 2-3 4-5
J1	外部时钟源	外部时钟源连接点		
JP1	Smart option 源选择	Smart Option 由外部 smart option 开关 (SW1) 选择	Smart Option 由内部 smart option 区 (ROM空间 003EH–0003FH) 选择。但该选项无效。	连接 1-2
SW1	Smart option 选择	当 Smart Option 源由外部提供时，Smart Option 可由该开关选择。B2–B0 相当于 003EH.2–.0。B7–B5 相当于 003EH.7–.5。B8 相当于 003FH.0。B4–B3 无连接。B9 不使用。参考第2-3页。		
JP3	LCD 偏置	电容偏置模式时需要电容		
JP4		外部电阻偏置模式时需要电阻		
JP2	MDS 版本	SMDS2+,SK-1200, OPENIce I-500	SMDS2	连接 1-2
To User_Vcc	Target System is supplied V_{DD}	目标板不向用户系统提供 V_{DD}。	目标板向用户系统提供 V_{DD}。	连接 2-3

26.1.3.1 IDLE LED

当EVA 芯片 (S3E82H0) 处于IDLE 模式时，该 LED 亮。

26.1.3.2 STOP LED

当EVA 芯片 (S3E82H0) 处于STOP 模式时，该 LED 亮。

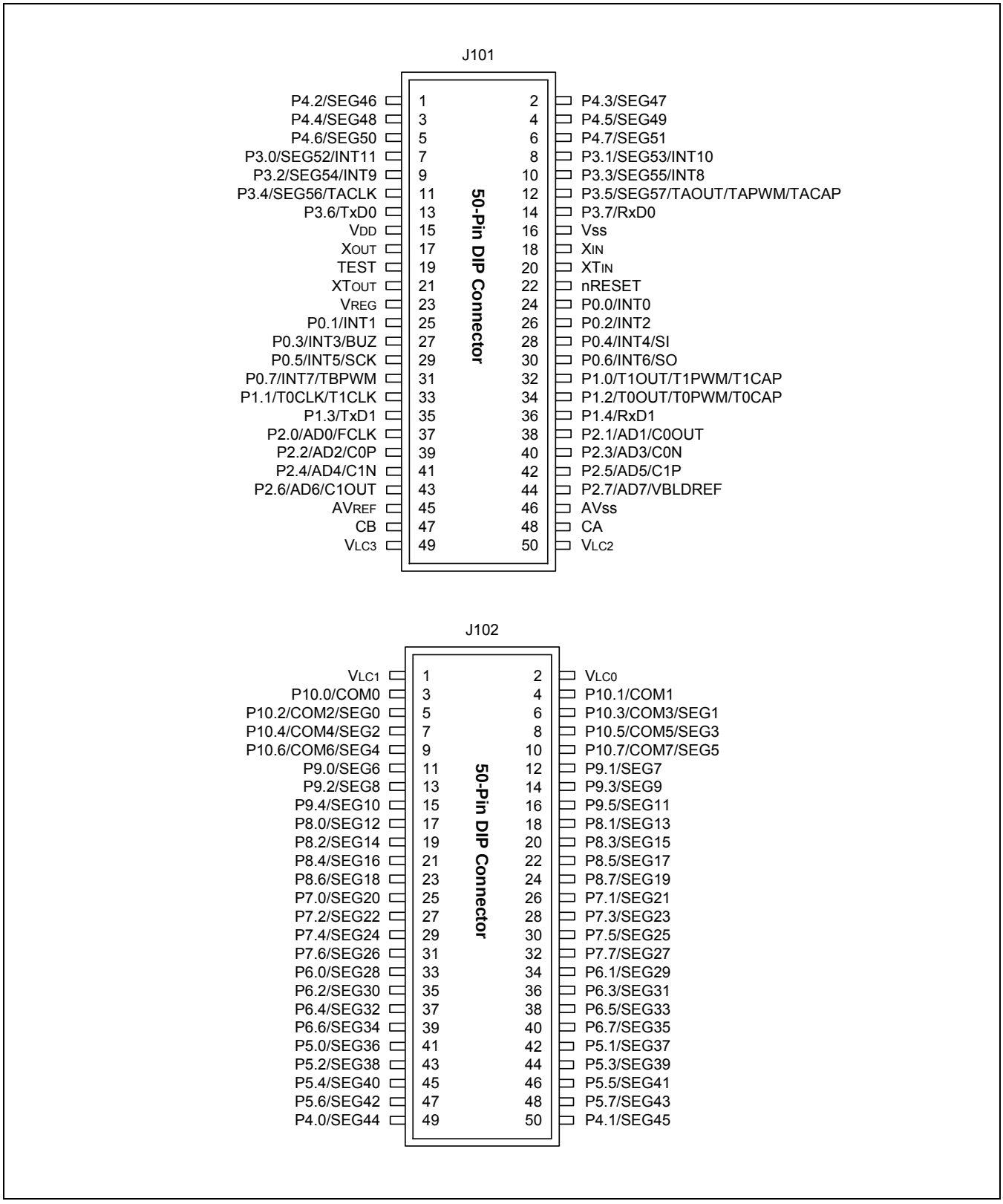


图 26-3 TB82HB 50-Pin 接口 (J101, J102)

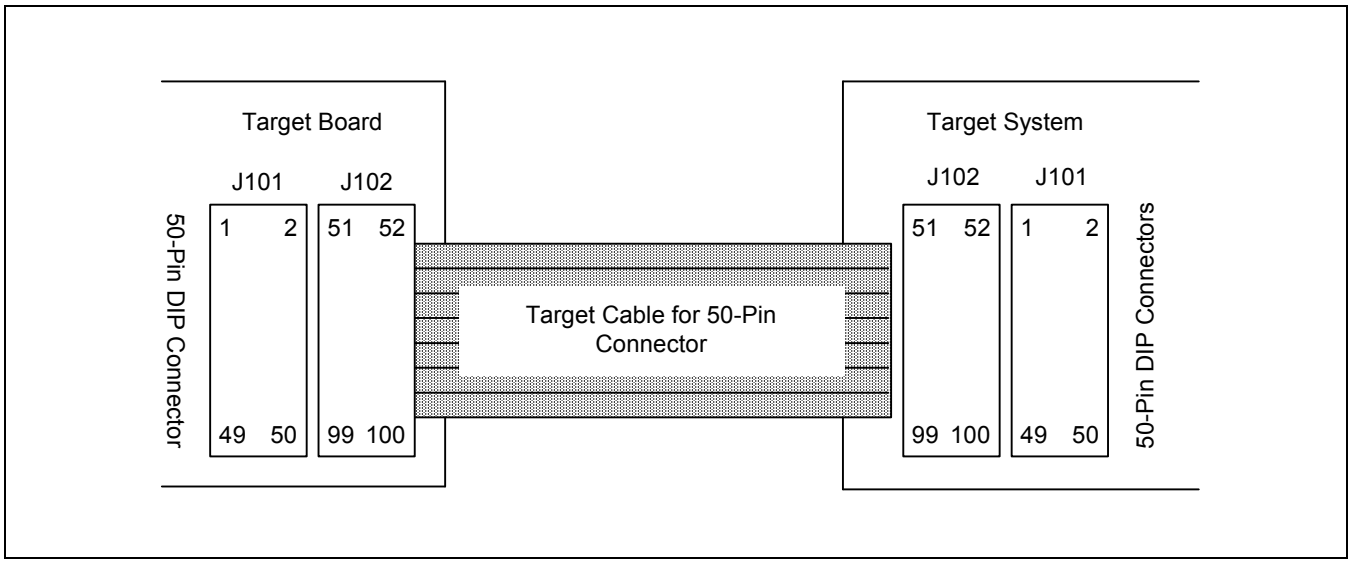


图 26-4 100-QFP 封装的 S3E82H0 数据线

26.1.4 第三方开发工具

三星提供开发三星 MCU 用的全系列工具。凭借着在 MCU 系统开发方面长时间的经验积累，
我们的第三方合作伙伴都是工具开发领域的佼佼者。三星的电路仿真器无论在价格还是在工具性能方面都为用户提供了很多选择，从低廉的 ICE 到完整的带有 OTP/MTP 编程功能的仿真器。

26.1.4.1 SAM8 系列在电路仿真器

- OPENice-i500
- SmartKit SK-1200

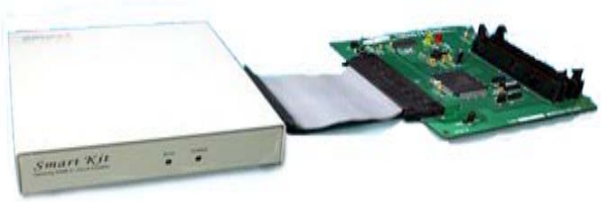
26.1.4.2 OTP/MTP 编程器

- SPW-uni
- GW-uni
- AS-pro
- US-pro

26.1.4.3 开发工具供应商

可以联系我们的当地销售机构，或者下边所示的第三方工具供应商。

26.1.4.4 8 位在电路仿真器

OPENice - i500 	AIJI 系统 • TEL: 82-31-223-6611 • FAX: 82-331-223-6613 • E-mail : openice@aijisystem.com • URL : http://www.aijisystem.com
SK-1200 	Seminix • TEL: 82-2-539-7891 • FAX: 82-2-539-7819 • E-mail: sales@seminix.com • URL: http://www.seminix.com

26.1.4.5 OTP/MTP 编程器 (烧写器)

	SPW-uni 单片 OTP/ MTP/FLASH 编程器 • 下载/上传数据及数据编辑功能 • 带 USB 口的基于 PC 的操作 • 全功能 OTP/MTP/FLASH MCU 编程器 (读，写，验证，擦除，保护) • 快速烧写 (4K字节/秒) • 支持所有的三星 OTP/MTP/FLASH MCU 芯片 • 低价 • NOR FLASH (SST, Samsung…) • NAND FLASH (SLC) • 增加设备文件或升级软件来支持新设备	SEMINIX • 电话：82-2-539-7891 • 传真：82-2-539-7819. • 邮箱：sales@seminix.com • 网站：http://www.seminix.com
	GW-uni OTP/MTP/FLASH MCU 的 Gang 编程器 • 8 芯片同时烧写 • 下载/上传数据及数据编辑功能 • 带 USB 口的基于 PC 的操作 • 全功能 OTP/MTP/FLASH MCU 编程器 (读，写，验证，擦除，保护) • 快速烧写 (4K字节/秒) • 支持所有的三星 OTP/MTP/FLASH MCU 芯片 • 低价 • NOR FLASH (SST, Samsung…) • NAND FLASH (SLC) • 增加设备文件或升级软件来支持新设备 • 将于2008年3月推出	SEMINIX • 电话：82-2-539-7891 • 传真：82-2-539-7819. • 邮箱：sales@seminix.com • 网站：http://www.seminix.com
	AS-pro Samsung Flash MCU 的在板编程器 • 适用于售后服务的便携式独立 Samsung OTP/MTP/FLASH 编程器 • 尺寸小，轻巧，便于携带 • 支持所有 SAMSUNG OTP/MTP/FLASH 芯片 • 通过 PC 的 USB 口下载 HEX 文件 • 非常快速的编程和验证 (OTP：2Kbytes 每秒，MTP：10Kbytes 每秒) • 内置大容量缓存 (118M 字节) • 驱动程序可工作在多种 O/S (Windows 95/98/2000/XP) • 全功能 OTP/MTP 编程器 (读，写，验证，擦除，保护) • 两种供电模式 (用户系统供电或 USB 电源适配器) • 支持固件升级	SEMINIX • 电话：82-2-539-7891 • 传真：82-2-539-7819. • 邮箱：sales@seminix.com • 网站：http://www.seminix.com

	US-pro 便携式 Samsung OTP/MTP/FLASH 编程器 • 便携式 Samsung OTP/MTP/FLASH 编程器 • 尺寸小，轻巧，便于携带 • 支持所有 SAMSUNG OTP/MTP/FLASH 芯片 • 可通过 USB 方便的连接到任何 IBM 兼容PC 或笔记本电脑 • 由PC 的 USB 口供电 • 便捷的基于 PC 的菜单式软件操作 • 非常快速的编程和验证 (OTP：2Kbytes 每秒，MTP：10Kbytes 每秒) • 支持 Samsung 标准 Hex 或 Intel Hex 标准 • 驱动程序可工作在多种 O/S (Windows 95/98/2000/XP) • 全功能 OTP/MTP 编程器 (读，写，验证，擦除，保护) • 支持固件升级	SEMINIX • 电话：82-2-539-7891 • 传真：82-2-539-7819. • 邮箱：sales@seminix.com • 网站：http://www.seminix.com
	Flash 编程适配器 • S3F82HB 专用 flash 烧写座 - 100QFP, 100TQFP	SEMINIX • 电话：82-2-539-7891 • 传真：82-2-539-7819. • 邮箱：sales@seminix.com • 网站：http://www.seminix.com